Viz Artist User Guide

Version 5.6



Viz Artist

Copyright ©2026 Vizrt. All rights reserved.

No part of this software, documentation or publication may be reproduced, transcribed, stored in a retrieval system, translated into any language, computer language, or transmitted in any form or by any means, electronically, mechanically, magnetically, optically, chemically, photocopied, manually, or otherwise, without prior written permission from Vizrt. Vizrt specifically retains title to all Vizrt software. This software is supplied under a license agreement and may only be installed, used or copied in accordance to that agreement.

Disclaimer

Vizrt provides this publication “as is” without warranty of any kind, either expressed or implied. This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document.

Vizrt’s policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time. Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Antivirus Considerations

Vizrt advises customers to use an AV solution that allows for custom exclusions and granular performance tuning to prevent unnecessary interference with our products. If interference is encountered:

- **Real-Time Scanning:** Keep it enabled, but exclude any performance-sensitive operations involving Vizrt-specific folders, files, and processes. For example:
  - C:\Program Files\[Product Name]
  - C:\ProgramData\[Product Name]
  - Any custom directory where [Product Name] stores data, and any specific process related to [Product Name].
- **Risk Acknowledgment:** Excluding certain folders/processes may improve performance, but also create an attack vector.
- **Scan Scheduling:** Run full system scans during off-peak hours.
- **False Positives:** If behavior-based detection flags a false positive, mark that executable as a trusted application.

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Created on

2026/11/19

Contents

1 Introduction to Viz Artist 26
1.1 Related Documents 26
1.2 Keyboard Shortcuts 26
1.3 Feedback and Suggestions 26
2 New in Viz Artist 27
2.1 Gaussian Splatting - Reality, Rendered in Real Time 27
2.2 Introducing Maps - Every story happens somewhere 28
2.3 Get started - Introducing Template Library 28
2.4 New Logicmaker 29
3 Getting Started with Viz Artist 30
3.1 General Concepts of Viz Artist 30
3.1.1 Reset and Undo/Redo 31
3.1.2 Default Values 32
3.1.3 Drag and Drop 33
3.1.4 To Change Values 34
3.1.5 Timecodes 35
3.1.6 Conflicting Values 35
3.2 Viz Artist Startup and Close 36
3.2.1 Starting Viz Artist 36
3.2.2 Viz Configuration 36
3.2.3 To Add a Viz Artist Startup Option 37
3.2.4 Close and Restart 38
3.3 Viz Artist Folders 39
3.3.1 Installation Folders 39
3.3.2 Data Folders 39
3.4 Viz Artist Command Line Options 40
3.4.1 Systems with Two or More GPUs 42
3.5 Welcome Screen 44
3.5.1 Using Templates 45
4 Viz Artist Interface Overview 46
4.1 Main Window 46
4.2 Main Menu 47
4.2.1 Default 47

4.2.2 Workspaces 47
4.2.3 Import 48
4.2.4 Archive 48
4.2.5 Post 48
4.2.6 On Air 48
4.2.7 Config 49
4.3 Button Bar 49
4.4 Docking and Workspaces 51
4.4.1 Workspace Manager 51
4.4.2 Working with Docks 53
4.4.3 Documentation 54
4.4.4 Render Views 55
4.5 Keyboard and Mouse Shortcuts 58
4.5.1 Shortcut Help 58
4.5.2 To Edit a Shortcut 59
4.5.3 To Export and Import Key Mappings 60
4.5.4 Application Controls and Shortcuts 61
4.5.5 Asset View Shortcuts 64
4.5.6 Import Shortcuts 66
4.5.7 Integer Float Controls 68
4.5.8 On Air Shortcuts 69
4.5.9 Polygon Plug-in Editor Shortcuts 70
4.5.10 Scene Editor Shortcuts 72
4.5.11 Scene Tree Shortcuts 77
4.5.12 Script Editor Shortcuts 80
4.5.13 Stage Shortcuts 81
4.5.14 Logicmaker Shortcuts 88
4.6 Color Picker 90
4.6.1 General Functions 90
4.6.2 Working with Different Color Spaces 91
4.6.3 Color Picker for BT.709 Linear Mode 91
4.6.4 Color Picker for High Definition Color spaces 92
5 Introduction to Viz Engine Rendering 94
5.1 Comparison of Render Pipelines 95
5.1.1 Rendering Technology 95
5.1.2 Framebuffer Format 95
5.1.3 Color Spaces 95

5.1.4 Sorting 95
5.1.5 Performance 95
5.1.6 Compositing of Media Assets 96
5.2 Moving from the Classic to Viz Engine Pipeline 98
5.2.1 Conversion Tools 98
5.2.2 Feature Replacement Table 99
5.2.3 Tonemapper, Gamma Correction and Color Spaces 100
5.3 From 2D Logo to AR Object 101
5.3.1 Step 1: Preparing Data for Use in a 3D Environment 101
5.3.2 Step 2: Bringing the Object into the 3D World 102
5.3.3 Step 3: Texturing Using Substance Painter 104
5.3.4 Step 4: Creating the AR Object 106
5.4 Postprocessing Effects 111
5.4.1 Postprocessing Effects 111
5.4.2 Cinematic Depth of Field 112
5.4.3 Screen Space Reflection 117
5.4.4 Bloom 124
5.4.5 Deep Learning Super Sampling 127
5.4.6 SSAO 131
5.4.7 Fog 134
5.4.8 Chromatic Aberration 139
5.4.9 Light Glares 141
5.5 Color Space 143
5.5.1 Color Limits 144
5.5.2 Tonemapping 147
5.6 Working with Phong Materials 154
5.6.1 Color 154
5.6.2 Ambient 156
5.6.3 Diffuse 157
5.6.4 Specular 157
5.6.5 Emissive 159
5.6.6 Final Result 160
5.6.7 Unlit 164
5.6.8 Reflectivity 166
5.6.9 Alpha 166
5.6.10 Texture Generation 169
5.6.11 Clamping 170

5.7 Working with PBR Materials....172
5.7.1 Color or Albedo Map....173
5.7.2 Emissive Color....173
5.7.3 Normal Maps....174
5.7.4 Roughness....175
5.7.5 Metallic....176
5.7.6 Ambient Occlusion....176
5.7.7 Height Maps....177
5.7.8 Environment Maps....177
5.7.9 Improved Image Import for PBR Materials....178
5.7.10 Using Transparency Effects....179
5.7.11 Texture Manipulation....180
5.7.12 Clamping....181
5.7.13 Working with Material Definitions on Graphic Hub....182
5.7.14 Environment Maps....184
5.7.15 Substance Materials....186
5.8 Light and Shadows....188
5.8.1 Lighting in the Viz Engine Render Pipeline....188
5.8.2 Working with Environment Maps....189
5.8.3 Directional Lights....192
5.8.4 Area Lights....192
5.8.5 Point Lights....193
5.8.6 Spot Lights....194
5.8.7 Working with IES Files....194
5.8.8 Volumes....196
5.8.9 Shadows....197
5.8.10 Advanced Light Falloff (Attenuation)....199
5.8.11 Troubleshooting Shadows....201
5.8.12 Concepts of Lighting....203
5.8.13 Volumetric Lighting....207
5.9 Global Illumination....211
5.9.1 Precomputation....211
5.9.2 Baking....214
5.9.3 Rendering....217
5.9.4 Geometry Requirements....220
5.9.5 Material Texture....220
5.10 Gaussian Splatting....222

5.10.1 How Gaussian Splatting works 222
5.10.2 Why it looks real 223
5.10.3 Benefits and Limitations 224
5.10.4 To use Splat files for Viz Engine 224
5.10.5 Importing Splat Files 229
5.10.6 Rendering 230
5.11 Using Multiple Textures in Viz Engine 242
5.11.1 To Setup Multiple Textures 242
5.11.2 To Use Multiple Textures on Objects 249
5.11.3 Simply Add some colors 254
5.11.4 To Use Screenspace Mapping 255
5.12 Working with Text and New Font System 257
5.12.1 Getting Started 257
5.12.2 Supported OpenType Features 264
5.13 Viz Engine Shaders 266
5.13.1 Introduction to Viz Engine Shaders 267
5.13.2 Properties for Viz Engine Shaders 272
5.13.3 Viz Engine Fragment Shader 273
5.13.4 Viz Engine Vertex and Fragment Shader 277
5.13.5 User Parameters 280
5.13.6 Transition Shaders 282
5.13.7 Shader Samples Library 285
5.13.8 Built-in Features 286
5.14 To Improve Render Quality 291
5.14.1 Anti-aliasing 291
5.15 Scene Compositing 296
5.15.1 Render Sequence 296
5.15.2 Precision Keyer 298
5.15.3 Matte Scene 298
5.15.4 Multi-Layer External VS 299
5.15.5 Multi-Layer Virtual Studio 306
5.15.6 Multi-Layer XR Set 310
5.15.7 Multi-Layer External XR 315
5.16 Virtual Studio, Augmented Reality and Extended Reality 320
5.16.1 To Setup Talent Shadow in a Virtual Studio 321
5.16.2 Setup Precision Keyer 326
5.16.3 To Setup a Reflection for Augmented Reality 343

5.16.4 To Setup a Talent Reflection in a Virtual Studio 345
5.16.5 Working with Holdout and Trash Mattes 351
5.16.6 Projected Virtual Window 356
5.17 Instancing 377
5.17.1 Example 378
5.18 Particle System 380
5.18.1 Creating a Particle System 380
5.18.2 Particles and Instancing 382
5.18.3 Properties 383
5.19 Working with Dynamic Geometries 394
5.19.1 Convert a Geometry into a Dynamic Geometry 394
5.19.2 Node Modifiers in Detail 394
5.19.3 Combining Multiple Modifiers 397
5.19.4 Use Blendshapes 398
5.20 Stylesheets 401
5.20.1 Introduction 401
5.20.2 Enabling and Using Stylesheets 402
5.20.3 Subscene Workflow 407
5.20.4 Scripting 407
6 Working with Maps 411
6.1 Note about searching Locations 412
6.2 Maps Prerequisites 413
6.2.1 Configuration 413
6.2.2 Creating Styles 413
6.2.3 Attribution Settings 414
6.3 Start Using Maps 415
6.3.1 Projections 415
6.3.2 GeoCanvas and GeoCamera 417
6.3.3 Geo Transformation 418
6.3.4 Caching and Optimizations 422
6.4 Working with GeoJSON 426
6.4.1 What is a GeoJSON 427
6.4.2 Importing GeoJSON 428
6.4.3 Rendering GeoJSON 434
6.4.4 Textures on GeoJSON 440
6.4.5 Adding GeoJSON data 445
6.5 Working with Animations 447

6.5.1 How a flight is built 449
6.5.2 Example 449
6.5.3 The settings in detail 450
6.5.4 Per-hop overrides 453
6.5.5 Driving it from a template 455
6.5.6 Tips and troubleshooting 456
6.5.7 Using GeoExpression 458
6.6 Template based workflow 465
6.6.1 Adding Labeldesigns 466
6.6.2 Adding Shapedesigns 473
6.6.3 Adding a Locator 476
6.6.4 Controlling the flight 478
7 Manage Items and Plug-ins 483
7.1 Working with Graphic Hub 483
7.1.1 To Switch Active Graphic Hub 484
7.1.2 To Log on a New User after Viz Artist has Started 486
7.2 Asset View 487
7.2.1 Asset View Window 488
7.2.2 Server Tree 488
7.2.3 Items View 492
7.2.4 Quick Look 503
7.2.5 Properties 506
7.2.6 Search 513
7.2.7 Shortcuts 519
7.3 Working with Items 520
7.3.1 Working with Fontstyle Items 521
7.3.2 Working with Various Materials 535
7.3.3 Working with Geometry Items 544
7.3.4 Working with Scene Items 552
7.3.5 Working with Image Items 557
7.3.6 Working with Audio Items 570
7.3.7 Working with File Collections 572
7.4 Background Loading 577
7.4.1 Automatic Background Loading 577
7.4.2 Manual Background Loading 578
7.4.3 Background Loading Limitations 578
7.5 Plug-ins Tab 580

7.5.1 Plug-in Color Coding 580
7.5.2 Plug-in Context Menu 582
7.5.3 View and Search for Plug-ins 583
7.5.4 Add Plug-ins to Containers 584
8 Working in HDR Color Space 588
8.1 Introduction to HDR 589
8.1.1 Color Spaces, Brightness and Contrast 589
8.2 Prerequisites and Configuration Details 592
8.2.1 Prerequisites 592
8.2.2 Setup HDR Preview 592
8.3 Usecases for HDR 595
8.3.1 SDR Scenes in an HDR Environment 596
8.3.2 HDR Scenes in an HDR Environment 596
8.3.3 HDR Scenes in an SDR Environment 596
8.3.4 Limitations and Known Issues 596
8.4 Mediatypes in HDR Workflows 597
8.4.1 Using Live Video 597
8.4.2 Clip Playback in HDR Workflows 598
9 Adaptive Scene Design 603
9.1 Introduction to Adaptive Scene Design 604
9.1.1 So, How Does It Work? 605
9.2 Box Transformation and 3D Transformation 606
9.2.1 3D Transformation 606
9.2.2 Box Transformation 606
9.2.3 To Switch between Box- and 3D Transformation 607
9.2.4 3D Objects Support for Box Transformation 608
9.3 Working with Different Formats 610
9.3.1 To Create a New Format 610
9.3.2 To Preview Formats 612
9.3.3 Format Overview 620
9.4 Introduction to Flexbox 623
9.5 Flexbox in Detail 626
9.5.1 Flex Wrap 627
9.5.2 Flex Direction 629
9.5.3 Direction 631
9.5.4 Flex Basis 631
9.5.5 Growing and Shrinking 631

9.5.6 Handling of 3D Objects....633
9.5.7 Aspect Ratio....633
9.5.8 Width, Height and Limits....635
9.5.9 Margins, Padding and Borders....636
9.5.10 Positioning....636
9.5.11 Justify Content....637
9.5.12 Align Items....640
9.5.13 Align Self....643
9.5.14 Align Content....643
9.6 Designing for Adaptive Storytelling....648
9.6.1 Planning Phase....648
9.6.2 Layout....650
9.6.3 Tutorial....651
9.7 Transition Logic and Adaptive Scenes....669
9.8 Parallel Outputs and Adaptive Scenes....670
9.8.1 To Setup Parallel Outputs....670
9.9 Adaptive Scene Design Tips and Tricks....673
9.9.1 Building TextBG Functionality....673
9.9.2 Building AutoFollow Functionality....674
9.10 Adaptive Scene Design Troubleshooting....676
9.10.1 Wrong Aspect Ratio....676
9.10.2 Verify Plug-in Settings for Different Formats....676
9.10.3 Text Limitations....677
10 Scene Tree....679
10.1 Scene Tree in General....681
10.1.1 Color Style....681
10.1.2 Plug-in Overlay....681
10.1.3 Smart Scrollbar....682
10.1.4 Zooming....684
10.2 Toolbars....686
10.3 Containers....688
10.3.1 Container Components....689
10.3.2 Container Names....690
10.3.3 Basic Concept of Using a Container....691
10.3.4 Selecting Multiple Containers....692
10.3.5 Copying Containers....693
10.3.6 Deleting Containers....695

10.3.7 Grouping Containers 695
10.3.8 Merging and Splitting Containers 696
10.3.9 Container Context Menu 697
10.3.10 Container Properties and Editors 699
10.3.11 Plug-ins and their order 704
10.4 Plug-in Overlay 705
10.4.1 To Add Plug-ins 706
10.4.2 To Manage Favorites 707
10.4.3 Recently Used Items 708
10.5 Searching 709
10.5.1 To Search for Container Names 711
10.5.2 To Search for Plug-ins 712
10.5.3 To Search for Color Labels 713
10.5.4 To Search for Containers Status 713
10.5.5 To Search for Scripts Fragments 713
10.6 Sorting 715
10.6.1 To Optimize Render Time 717
10.6.2 To Optimize Your Geometries 718
10.6.3 To Reduce Texture Size 719
10.7 Subscenes 721
10.7.1 To Add a Subscene 722
10.7.2 Grouping and Ungrouping 723
10.7.3 Save to Database 723
10.7.4 Re-Establish a Subscene 723
10.7.5 To Remove a Subscene 724
10.7.6 To Select a Container of Subscene 724
10.7.7 Accessing "Main" Containers from within a Subscene 724
10.7.8 Adaptive Scene Design Formats Support for Subscene 724
10.7.9 Shared Memory Support in Subscenes 725
10.7.10 Workspace Items Support in Subscenes 725
10.7.11 Known Limitations 725
10.8 Layer Manager 726
10.8.1 Layer Support in the Viz Engine Render Pipeline 728
10.8.2 To Add Scenes to the Front or Back Layer 728
10.8.3 To Edit a Scene Used as a Front or Back Layer 728
10.8.4 Self Layers 728
11 Scene Management 730

11.1 Working with the Scene Editor 731
11.1.1 Scene Editor Functions 731
11.1.2 Scene Editor Context Menu 734
11.2 Scene Overview 742
11.2.1 Sample View 742
11.2.2 In-Scene Editing 743
11.2.3 To Switch Back to the Main Scene 744
11.3 Scene Settings 746
11.3.1 Rendering Panel 748
11.3.2 Global Settings Panel 759
11.3.3 Global Media Assets Settings 770
11.3.4 Plug-in Panel 772
11.3.5 Clipper Panel 775
11.3.6 High Dynamic Range Panel 777
11.3.7 Real Time Global Illumination 779
11.3.8 Classic SSAO 796
11.3.9 Virtual Studio Panel 798
11.3.10 Script Panel 801
11.3.11 Adaptive Scene Design Formats 802
11.3.12 Unreal Scene Settings 804
11.4 Scene Editor 806
11.4.1 Shortcuts 806
11.4.2 Scene Editor Snapshot 808
11.4.3 Grid Tool-bar 810
11.4.4 Performance Bar 813
11.4.5 Scene Editor Buttons 817
11.5 Scene Editor Views 819
12 Editors 821
12.1 Transformation Editor 822
12.1.1 Transformation Editor Properties 823
12.1.2 Transformation Principles 826
12.1.3 Multiple Select Transformation 826
12.1.4 Copy Transformation Properties to Another Container 827
12.2 Texture Editor 828
12.2.1 Image Properties 830
12.2.2 Transformation 832
12.2.3 Mapping 832

12.2.4 Multitexturing 833
12.2.5 Quality Properties 833
12.2.6 Environment Properties 835
12.2.7 Stereo Mode Properties 835
12.3 Classic Text Editor 837
12.3.1 To Open the Classic Text Editor 838
12.3.2 To Create a Classic Text Object 839
12.3.3 Classic Text Editor 839
12.3.4 Inline Text Editor 851
12.3.5 Special Fonts 851
12.4 Viz Engine Text Editor 854
12.4.1 Text Input 855
12.4.2 Font Favorites 855
12.4.3 Font Family / Font Style 856
12.4.4 Characters 856
12.4.5 Basic Styling 857
12.4.6 Layout 859
12.4.7 Advanced Styling 863
12.4.8 Font Effects & Variables 865
12.4.9 Rendering 866
12.4.10 Razor 866
12.4.11 Geometry 868
12.4.12 Debug 870
12.5 Workspace Editor 871
12.5.1 Shaders 871
12.5.2 Textures 871
12.5.3 Texture Renderer 872
12.5.4 Pools Inspector 890
13 Media Assets 891
13.1 Video Clips 893
13.1.1 Video Clip Storage 893
13.1.2 Matrox Clip Player information 895
13.1.3 VML Clip Player information 895
13.1.4 Video Clip Playback 896
13.1.5 Video Clip Playback Considerations 909
13.1.6 Transfer Clips From Viz One 918
13.2 Media Asset Manager 922

13.2.1 Global Settings for Media Assets 922
13.2.2 Working with the Media Asset Overview 923
13.3 Media Asset Panel 930
13.3.1 Transformation Settings 931
13.3.2 Media Asset Keying Settings 932
13.3.3 Color Correction 933
13.3.4 Talent Reflection 934
13.3.5 Shared Memory (SHM) Aux 934
13.4 Media Asset Channel Types 936
13.4.1 AUX Channels 937
13.4.2 Channel Folder Media Assets 938
13.4.3 Clip Channel Media Asset 939
13.4.4 Container Folder Media Assets 941
13.4.5 GFX Channels 947
13.4.6 Image Channels 953
13.4.7 Live Video Media Asset 957
13.4.8 Super Channels 959
13.4.9 Web Channels 972
13.5 Playback of Media Assets 974
13.5.1 DVE 974
13.5.2 Texture 974
13.6 Keying Mode 975
13.6.1 Linear Key 975
13.6.2 Multi-Zone Key 975
13.6.3 External Key 975
13.6.4 External Processed Foreground 975
13.6.5 Precision Keyer 976
13.6.6 Keying Best Practices 977
13.6.7 To Apply Keying Parameters via SHM and Scripting 978
13.7 Viz Artist Supported Codecs 980
13.7.1 Bluefish Codecs 980
13.7.2 Supported Codecs for VML Clip Player 981
13.7.3 Advanced Issues with Video Codecs 990
13.7.4 Supported Codecs for Matrox Clip Playback 991
14 Lights 1027
14.1 Types Of Light 1028
14.2 Light Source Animation 1029

14.3 Light Editor 1030
14.3.1 Light Editor 1030
14.3.2 Light Editor Properties 1032
14.3.3 Light Editor Procedures 1034
14.4 Light Visualization 1037
14.4.1 To Modify the Light Direction with Light Handles 1038
14.5 Shadow Maps 1040
14.5.1 About Shadow Maps 1040
14.5.2 To Use Shadows in a Scene 1042
15 Cameras 1043
15.1 Track Objects with a Camera 1044
15.1.1 To Track an Object 1044
15.2 Camera Editor 1046
15.2.1 Parameters for Perspective View 1048
15.2.2 Parameters for Orthogonal View 1054
15.2.3 Parameters for Window View 1056
15.3 Stereo Settings 1059
15.3.1 Stereoscopy Best Practices 1062
15.3.2 Stereoscopic Output Using Shutter Glasses 1063
15.4 Change Camera Parameters in Orthogonal Views 1065
15.4.1 To View the Predefined Orthogonal Views 1065
15.4.2 To Change the Camera Views 1065
15.5 Receive Tracking Data from a Real Camera 1067
15.5.1 To Receive Tracking Data from a Real Camera 1067
15.6 Copy Properties from One Camera to Another 1068
15.6.1 To Copy Camera Properties 1068
15.7 Camera Selection 1069
15.7.1 Camera Selection Panel Properties 1069
15.8 Camera Animation 1072
15.9 Advanced Lens Distortion 1073
15.9.1 Important Changes Compared to Previous Versions 1073
15.9.2 Distortion Algorithms 1074
15.9.3 Parameter Providers 1075
15.9.4 Configuration File Flags 1075
16 The Stage for Animation 1077
16.1 Stage Elements 1077

16.2 Channel Editor 1079
16.2.1 Common Channel Editor Properties 1079
16.2.2 Position Channel Editor Properties 1079
16.2.3 Action Channel Editor Properties 1081
16.3 Time-line Marker 1083
16.3.1 To Manually Move the Time-line Marker 1083
16.3.2 To Move the Time-line Marker a Set Amount of Fields 1083
16.3.3 To Move the Time-line Marker a Set Amount of Fields with Shortcuts 1084
16.4 Key Frame Editors 1085
16.4.1 Common Key Frame Editor Properties 1085
16.4.2 Locked and Unlocked Key Frames 1086
16.4.3 Action Key Frame Editor Properties 1086
16.4.4 Position Key Frame Editor Properties 1087
16.4.5 Rotation Key Frame Editor Properties 1089
16.4.6 Scale Key Frame Editor Properties 1090
16.4.7 Material Key Frame Editor Properties 1091
16.4.8 Clip Key Frame Editor Properties 1093
16.4.9 Adding Camera Events Using Key Frames 1095
16.5 Event Editor 1097
16.6 Stage Tree Area 1098
16.6.1 Stage Tree 1098
16.6.2 Stage Tree Bar 1099
16.6.3 Stage Tree Menu 1100
16.7 Stage Editor 1103
16.7.1 Stage Editor Bar 1103
16.8 Time-line Editor 1105
16.8.1 Layer/ Director Controls 1105
16.8.2 Play Controls 1105
16.8.3 Keyframe and Timeline Controls 1105
16.9 Director Editor 1107
16.9.1 Director Editor Properties 1107
16.10 Director Control Panel 1109
16.10.1 Working with the Director Control Panel 1111
16.11 Master Clip 1112
16.12 Actor Editor 1113
16.12.1 Actor Editor for Integer Channels 1113
16.13 Dopesheet Editor 1114

16.14 Spline Editor 1117
16.14.1 To Resize the Height or Width of the Spline Editor 1117
16.14.2 Zooming 1118
16.14.3 To Switch Between the Various Splines 1118
16.14.4 Example of Axes X, Y, and Z in the Spline Editor 1119
16.14.5 To Move Key Frames in the Spline Editor 1120
16.14.6 Editing Handles 1120
16.14.7 Position in the Spline Editor 1121
16.14.8 Rotation in the Spline Editor 1121
16.14.9 Scale in the Spline Editor 1122
16.14.10 Material in the Spline Editor 1122
16.14.11 Additional Best Practices for the Spline Editor 1123
16.15 Stage Object Editor 1125
17 Create Animations 1126
17.1 Actors 1127
17.1.1 Actors in the Stage Tree 1128
17.1.2 Actors in the Dopesheet Editor 1129
17.2 Channels 1131
17.2.1 Channels in the Stage Tree 1131
17.2.2 Channels in the Dopesheet Editor 1132
17.3 Basic Animation Functions 1134
17.3.1 To Play All Animations in a Director from the Start of the Time-line 1134
17.3.2 To Play Multiple or All Animations in a Scene 1134
17.3.3 To Play an Animation in Reverse 1135
17.3.4 To Stop a Director 1135
17.3.5 To Stop an Animation 1136
17.3.6 To Stop All Animations 1136
17.3.7 To Continue an Animation 1136
17.3.8 To Jump to Start of an Animation 1136
17.3.9 To Jump to the End of an Animation 1136
17.3.10 To Delay Animations in a Director with Offsets 1136
17.3.11 Scale Animations 1136
17.3.12 To Scale Animations in a Director 1137
17.4 Create a Basic Animation 1138
17.4.1 Animation Task 1138
17.4.2 To Create the Basic Animation Scene 1138
17.4.3 To Create a Basic Animation 1138

17.4.4 To Create Key Frames with the Transformation Editor....1139
17.5 Key Frames....1141
17.5.1 To Add a Key Frame....1141
17.5.2 To Select One or More Key Frames....1142
17.5.3 To Move One or More Key Frames....1142
17.5.4 To Move a Key Frame to a Specific Time....1143
17.5.5 To Copy a Key Frame....1143
17.5.6 To Rename a Key Frame....1143
17.5.7 To Delete a Key Frame....1144
17.5.8 Reverse the Order of Selected Key Frames....1144
17.6 Directors....1145
17.6.1 Directors in the Stage Tree....1145
17.6.2 Directors in the Dopesheet Editor....1147
17.6.3 Events in a Director....1149
17.7 Create a Stand-alone Scene....1153
17.7.1 To Create a Scene....1153
17.7.2 To Add a Background....1154
17.7.3 To Add Text....1154
17.7.4 To Create an In and Out Animation....1155
17.7.5 To Add Stop Tags....1158
17.7.6 To Add Key Functions to the Container....1159
17.7.7 To Add Expose Properties....1159
17.7.8 To Edit Multiple Items with a Single Value....1163
17.8 Create an Over the Shoulder Scene....1164
17.8.1 To Create a Scene....1164
17.8.2 To Add an Image and Basic Geometries....1164
17.8.3 To Add Materials....1165
17.8.4 To Scale Geometries and Place Them in Z Space....1166
17.8.5 To Add Text....1167
17.8.6 To Add Background Items....1168
17.8.7 To Animate....1169
17.8.8 To Animate Sub Containers....1171
17.8.9 To Animate Single Containers....1172
17.8.10 To Animate Single Objects....1172
17.8.11 To Animate out....1174
17.8.12 To Fade out the Graphic with the Alpha Plug-in....1174
17.8.13 To Add Controls....1174

17.9 Action Channels....1177
17.9.1 Action Channels in the Stage Tree....1177
17.9.2 Action Channels in the Dopesheet Editor....1178
17.9.3 To Use Action Channels....1178
17.10 Create an Advanced Animation....1181
17.10.1 To Edit the First Key Frame....1181
17.10.2 To Edit the Second Key Frame....1182
17.10.3 To Create an Additional Key Frame....1182
17.10.4 To Add a Stop Point....1183
17.10.5 To Create Key Frames for Rotation....1183
17.10.6 To Modify the Rotation Settings....1184
17.10.7 To Run the Animations Independently....1184
17.10.8 To Add and Animate a Material....1185
17.11 Advanced Animation Functions....1186
17.11.1 Animating Channels....1186
17.11.2 Path Control....1188
17.11.3 Spline Control....1189
17.11.4 Animate Audio....1191
17.11.5 Animate Cameras....1192
17.11.6 Animate Light....1194
17.11.7 Merging Animations....1194
17.11.8 Multi Select Animation....1194
18 Import, Post and Archive....1196
18.1 Post Render Scenes....1197
18.1.1 Post Render Screen....1197
18.1.2 To Post Render a Scene....1198
18.1.3 Post Render Templates....1201
18.1.4 Post Rendering Plug-ins....1202
18.2 Archives....1204
18.2.1 Archive Workspace....1204
18.2.2 To Create an Archive....1206
18.2.3 To Use Path Aliases....1207
18.3 Import of Files and Archives....1209
18.3.1 Import Using Asset View....1210
18.3.2 Import Window....1212
18.3.3 To Import Various File Types....1214
18.3.4 To Import Archives....1223

18.4 Deploy Items....1235
18.4.1 To Configure Deploy Destinations....1235
18.4.2 To Deploy Items....1236
19 Viz Artist-On Air Mode....1240
19.1 Viz Artist On Air Control Buttons....1242
19.1.1 Animation Controls....1242
19.1.2 Preview Controls....1242
19.1.3 Tools....1242
19.2 Viz Artist On Air Director Control Panel....1243
19.3 Viz Engine Performance....1244
19.3.1 Performance Monitor....1244
19.3.2 Renderer Overlays....1245
19.4 On Air Information....1250
19.4.1 Basic Tab....1250
19.4.2 Clients Tab....1251
19.5 Unreal Control Panel....1252
19.6 Viz Artist Information....1253
20 Transition Logic....1254
20.1 How a take works.....1255
20.2 Architecture Overview....1255
20.3 The Take Lifecycle (TCP)....1256
20.3.1 First Take: Layer Goes from $O to $IN....1256
20.3.2 Update On Air (No Animation)....1257
20.3.3 Swap: A/B Switch to a Different Template....1257
20.3.4 Take Out: Layer Goes from $IN to $O....1257
20.3.5 Multi-Layer Concurrent Operation....1257
20.3.6 Session Cleanup....1258
20.4 Standalone versus Transition Logic Scene Design....1259
20.4.1 What Transition Logic Solves....1259
20.4.2 Take Lower Third....1260
20.4.3 Take OTS....1260
20.4.4 Take Out OTS....1260
20.4.5 Take Out Lower Third....1261
20.5 Adaptive Scenes and Subscene Mode....1262
20.5.1 Merged Geometry vs Subscene Mode....1262
20.5.2 Toggle Adaptive Properties....1262
20.5.3 Subscene Mode TCP Workflow....1263

20.6 Logicmaker 1264
20.6.1 Quick start 1264
20.6.2 User Interface 1265
20.6.3 Creating a package 1266
20.6.4 Messages and troubleshooting 1271
20.6.5 Reference 1273
20.6.6 Layers pane 1275
20.6.7 The State Graph 1277
20.6.8 Scenes Pane 1280
20.6.9 Control Objects Pane 1283
20.6.10 Health Check 1284
20.7 Toggle Layer 1288
20.8 State Transition Animation 1290
20.8.1 Toggle Layer States 1290
20.8.2 Defining States 1290
20.8.3 Identifying States 1291
20.9 Cross Animation 1297
20.9.1 Custom Transitions 1297
20.10 Master Scene 1299
20.11 Object/Leaf Scene 1300
20.12 Troubleshooting 1302
21 External Control 1304
21.1 Control Channels 1305
21.1.1 Control Channels Properties 1307
21.1.2 Control Channels Procedures 1308
21.2 Control Objects 1310
21.2.1 Control Object Parameters 1310
21.2.2 Search 1311
22 Scripting 1313
22.1 Script Editor 1314
22.1.1 Scene Scripts 1314
22.1.2 Script Editor Properties 1315
22.2 Create and Run Scripts 1324
22.2.1 Example I 1324
22.2.2 Example II 1325
22.3 Create Script-based Plug-ins 1327
22.3.1 To Name Your Script 1327

22.3.2 To Create a Script-based Plug-in 1328
22.3.3 To Retrieve the Original Code 1329
22.3.4 To Edit a Script-based Plug-in 1330
22.4 Integration Guide 1331
22.4.1 Overview 1331
22.4.2 Language Essentials 1332
22.4.3 Property Path Guide 1336
22.4.4 Mouse Events 1342
22.4.5 Shared Memory (SHM) 1344
22.4.6 Example: SHM → Text Update 1346
22.4.7 Example: SHM → Visibility Toggle 1347
22.4.8 Example: SHM → Animation Dispatch 1347
22.4.9 Example: Alpha Fade Over Time 1350
22.4.10 Example: Data-Driven Team Colors 1351
22.4.11 Example: Calculate the Days of a Week and Month 1352
22.4.12 Update a Scene Using a Text File 1353
22.4.13 Scale and Transform Animation 1354
22.4.14 Shader Integration 1357
22.4.15 Common Mistakes 1358
22.4.16 API Quick Reference 1359
22.4.17 Event Pool 1362
23 Shared Memory 1366
23.1 See also: 1366
23.2 Data Sharing 1367
23.2.1 VizCommunication.Map 1369
23.2.2 Map Access through a TCP or UDP Interface 1369
23.2.3 Map Access through the Command Interface 1369
23.2.4 Map Access from within Plug-ins 1370
23.2.5 Script Examples 1370
23.2.6 Set and Retrieve Data from the SHM through Scripting 1371
23.3 Shared Memory Synchronization 1374
23.3.1 TCP and UDP Synchronization 1374
23.3.2 External Control Synchronization 1375
23.3.3 Command Synchronization 1376
23.4 Shared Memory Snapshots 1378
23.5 Shared Memory External Data Input 1379
23.5.1 TCP Protocol 1380

23.5.2 UDP Protocol 1382
23.5.3 Plug-in API 1382
23.5.4 Command Interface 1383
23.5.5 REST Interface 1383
24 Third Party Applications and Files 1385
24.1 Integration with Unreal Engine 1386
24.1.1 Known Limitations 1386
24.1.2 Prerequisites 1387
24.1.3 To Install the Media Plug-in 1388
24.1.4 Viz Engine Preparation 1390
24.1.5 Unreal Engine Preparation 1393
24.1.6 Adding Media Assets from Viz Engine in Unreal Scenes 1406
24.1.7 Quality Settings (Unreal Engine Scalability Settings) 1410
24.1.8 Keep Old Frame vs Black Frame 1412
24.1.9 Keying and Compositing in Unreal 1412
24.1.10 Gamma Correction 1412
24.1.11 Syncing Unreal Project with Graphic Hub 1414
24.1.12 Using Viz Engine Control Objects 1414
24.1.13 Legacy Versions 1414
24.1.14 Troubleshooting 1415
24.1.15 Shared Memory Map 1419
24.1.16 Use Cases for Unreal Integration 1422
24.1.17 Control Objects on Unreal Engine 1433
24.1.18 AR Media Compositing with No Keyer 1450
24.1.19 Compositing in Unreal Engine 5 (Compositor Workflow) 1459
24.1.20 Compositing with an External Keyer 1473
24.1.21 Mixing Depth Buffers 1476
24.1.22 Precision Keyer Compositing on UE5 1481
24.1.23 Setting up a Multi-GPU Workflow 1495
24.1.24 Synchronization with Graphic Hub 1497
24.1.25 Unreal Engine Vizrt Pawn 1499
24.1.26 Lens Distortion on Unreal Engine 1503
24.1.27 Outputting Directly on a DisplayPort 1505
24.1.28 Compositing in Unreal Engine 5 (Composure Workflow) 1507
24.1.29 Using OpenColorIO with the Unreal Integration 1515
24.2 FBX Files 1520
24.2.1 Scene Graph Traversal 1521

24.2.2 Transformations 1522
24.2.3 Export FBX Files from 3ds Max and Maya 1523
24.2.4 Reimporting Maya FBX Files 1529
24.3 CINEMA 4D 1531
24.3.1 System Requirements and Configuration 1531
24.3.2 Cinema4D Substance FBX Workflow 1534
24.3.3 Cinema4D LiveLink 1538
24.3.4 CINEMA 4D Integration Limitations 1543
24.4 Adobe After Effects 1545
24.4.1 System Requirements 1545
24.4.2 Preparation of Your After Effects Project 1547
24.4.3 To Import After Effect Projects 1549
24.4.4 Frame Accurate Output 1550
24.5 NDI and TriCaster 1551
24.5.1 System Requirements 1551
24.5.2 NDI Protocol 1551
24.5.3 To Connect a Viz Engine to TriCaster 1552
24.5.4 To Enable Audio Passthrough from Viz Engine to TriCaster 1552
24.5.5 To Set TriCaster Macros 1552
24.5.6 Viz Trio and Viz Engine for a TriCaster Production on the Same Laptop 1552
24.6 Photoshop Import 1553
24.6.1 What is Supported 1554
24.6.2 Known Limitations 1554
24.7 Alembic Import 1556
24.7.1 Introduction 1556
24.7.2 Supported Features 1556
24.7.3 How Alembic Operates in Viz Engine 1556

1 Introduction to Viz Artist

This User Guide gives the information required to understand the Viz Artist User interface, and how to do certain tasks.

Throughout this guide, the term Viz refers to the complete software suite installed, and as a general reference for the following modes:

- Viz Artist
- Viz Engine (see the Viz Engine Administrator Guide)
- Viz Configuration (see the **Configuring Viz** section of the Viz Engine Administrator Guide)
  The available features and modes of the software depend on the license on the connected license hardware dongle. The different modes and the hardware dongle are detailed in this User Guide.

1.1 Related Documents

- **Viz Engine Administrator Guide:** Contains information on how to install and configure Viz Artist and Engine, and supported hardware.
- **Viz Configuration** (see the **Configuring Viz** section of the Viz Engine Administrator Guide)**:** Contains detailed information on how to configure Viz Artist and Engine.
- **Viz Artist Script Reference:** Contains information for Viz Artist scene script developers. This is installed locally at *C:\Program Files\Vizrt\VizEngine\Documentation\viz-script-doc-webhelp*.
- **Viz Engine Plug-in SDK Reference:** Contains information for developers of customized plug-ins for Viz. This is installed locally at *C:\Program Files\Vizrt\VizEngine\Documentation\viz-pluginsdk-doc-webhelp*.
- **Viz Engine Command Reference:** Contains information about Viz Engine Commands. This is installed locally at *C:\Program Files\Vizrt\VizEngine\Documentation\viz-cmd-doc-webhelp.*

1.2 Keyboard Shortcuts

Keyboard Shortcuts mentioned in this documentation always refer to the default mapping. Keyboard shortcuts can be modified since Viz Artist version 5.3.0.

1.3 Feedback and Suggestions

We encourage suggestions and feedback about our products and documentation. To give feedback and/or suggestions, please contact your local Vizrt customer support team at www.vizrt.com.

Copyright © 2026 Vizrt

2 New in Viz Artist

These are some of the highlights, alongside a vast number of smaller improvements listed in the release notes.

2.1 Gaussian Splatting - Reality, Rendered in Real Time



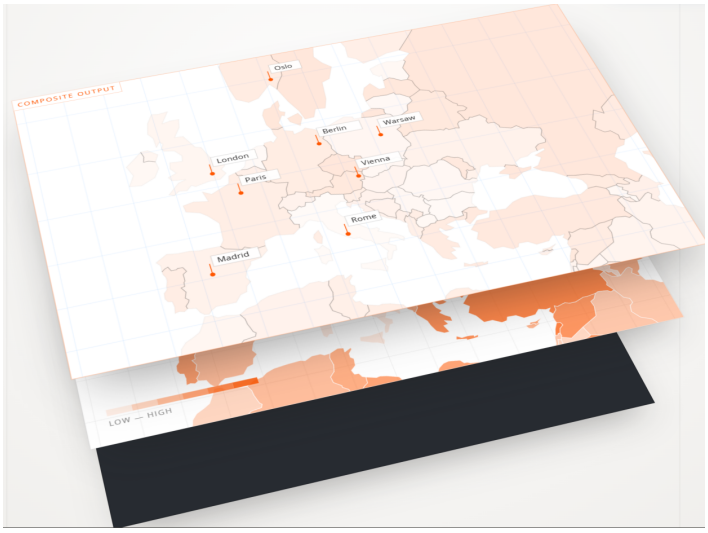
Photorealism used to be a trade-off: weeks of modelling, or "close enough." Gaussian Splatting in Viz Engine removes it. Instead of rebuilding a location, you capture it.

In **virtual studios**, set extensions and backdrops stop looking rendered and start looking photographed — holding their perspective and parallax under a moving tracked camera. In **augmented reality**, the payoff is the reverse: AR elements finally share the visual language of the real footage beside them, and the illusion holds.

Capture instead of model. Photoreal by origin. Real time, on air. Capture once, reuse everywhere.

Copyright © 2026 Vizrt

2.2 Introducing Maps - Every story happens somewhere



Maps in Viz Engine put the whole planet inside your graphics — live, navigable, and styled like everything else you put to air. Not a still someone ordered this morning. Not a browser bolted onto a texture. A real map, rendered in the scene, that you can fly.

Pick a place and go. The camera dives from orbit to a rooftop in one unbroken move, your graphics stay pinned to the ground beneath it, and the whole thing lights and composites with the rest of your design because it *is* part of your design.

2.3 Get started - Introducing Template Library



Viz Artist now includes a **Template Library**, available from the Templates tab of the Welcome Screen.

Browse a growing collection of scenes built by Vizrt, download what you need, and use it as a starting point for your own graphics or as a reference for how specific features work.

Copyright © 2026 Vizrt

2.4 New Logicmaker



Build, preview and check complex Transition Logic pages right inside Viz Artist, in a fast native panel connected directly to the engine.

Layers and states at your fingertips. Click a state to take it animated,
A live state graph. Takes run along the arrows for the real transition duration.
Scenes at a glance. Thumbnails, layer filters and search, with warnings on scenes whose Control Object doesn't match.
Guided setup. A wizard builds the master and leaf scenes from scratch for you.
Health checks and safer edits. Checks explain each problem in plain language and assist in fixing issues

Viz Engine	**Viz Artist**	Integrations
• Realtime Maps rendering • Gaussian Splatting Rendering • Gradient Editor • Security Upgrades	• More modern User Interface • Welcome Screen and Template Library • New Logicmaker • Pools Inspector • Workspace Manager • Quick Look • New Treeview Search	• Unreal Engine 5.7 and 5.8 support • NDI HX support

Copyright © 2026 Vizrt

3 Getting Started with Viz Artist

This section describes the location of the installation files, how to start Viz Artist and logging on to a Graphic Hub, and how to close Viz Artist.

This section contains information on the following topics:

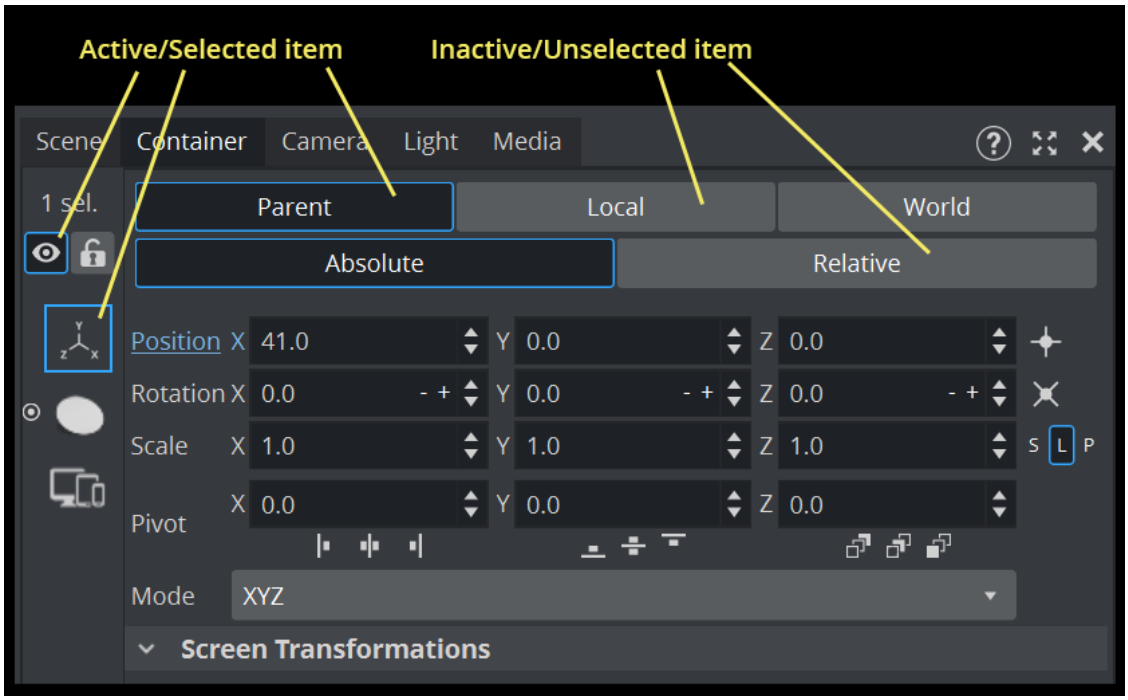
- Viz Artist Startup and Close
- Viz Artist Folders
- Viz Artist Command Line Options
- Welcome Screen

3.1 General Concepts of Viz Artist

The User Interface of Viz Artist has been designed to achieve the best experience while creating and designing stunning graphics. A modern dark themed User Interface helps you to focus on your work while offering a flexible docking system to customize your workspace.

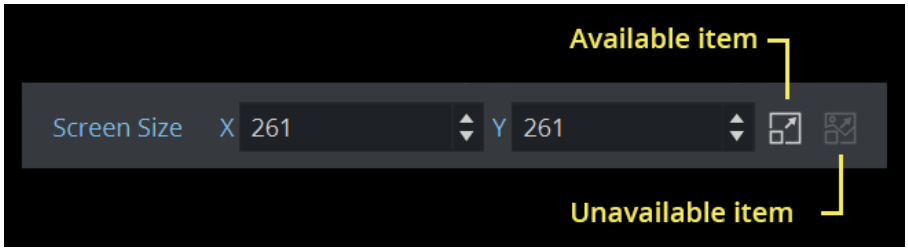
Working in Viz Artist is based on Drag and Drop concept, almost every item can be dragged to different locations, for example to the trash bin to delete items.

Active Items are marked with a blue border and a darker background:



Items that are not available (for example, not supported in the current render pipeline) are greyed out and cannot be used.

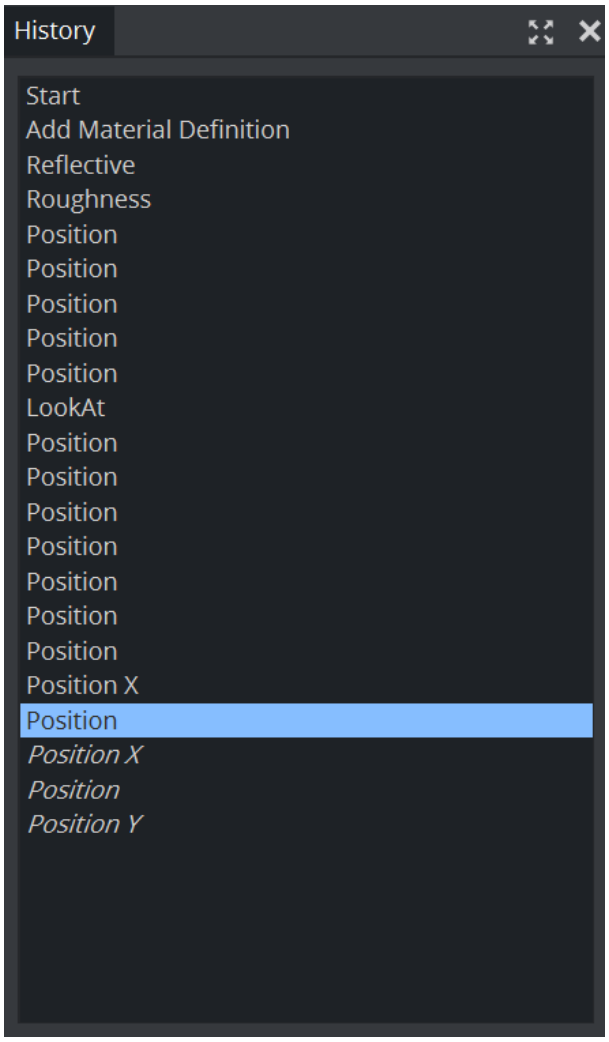
Copyright © 2026 Vizrt



3.1.1 Reset and Undo/Redo

Any action can be reverted and redone by using the **Undo** and **Redo** ( ) buttons on the bottom of the Scene Tree dock.

All steps are shown in the Undo/Redo stack dock. Click on any entry to roll back or forward to the selected action:



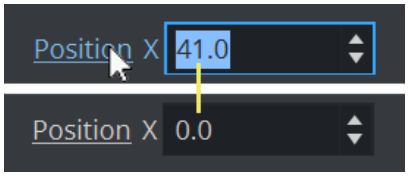
Copyright © 2026 Vizrt

3.1.2 Default Values

Any value that differs from its default setting is marked with a blue label. To revert back to the default value, double click on this blue label. This also works on groups of parameters.

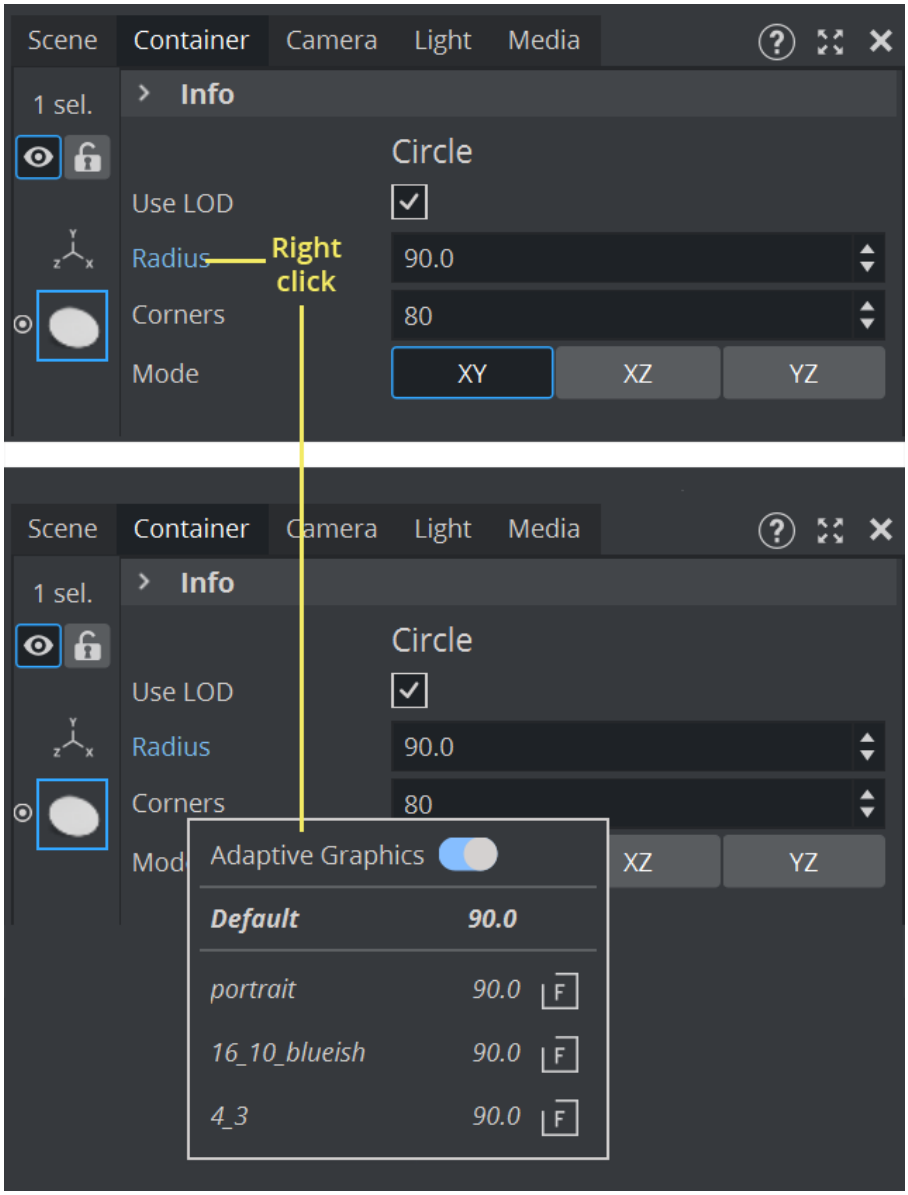
Example

The value of 141.0 of the X position is set to the default value of 0.0 by double clicking on the blue X. By clicking on **Position**, all three values of X, Y and Z are reset to their default values.



Values used in different formats, mainly used for adaptive scene designs are marked with an underline:

Copyright © 2026 Vizrt



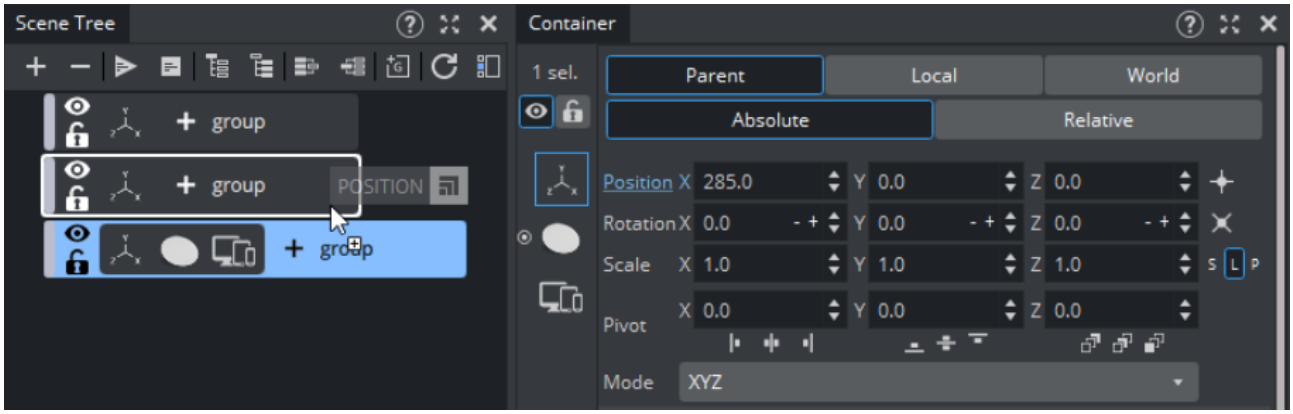
3.1.3 Drag and Drop

Items can be assigned by dragging them from various pools (for example, plug-ins) to their respective destinations.

Example

The position property can be taken and dragged to another container. This means that all X, Y and Z position properties of the source container are applied to the target container.

Copyright © 2026 Vizrt

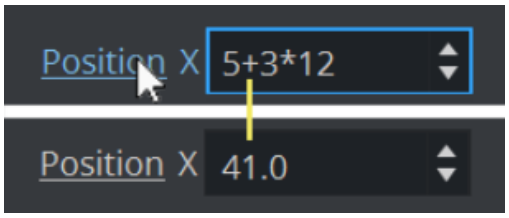


When dragging a property to the Control Channels panel, this property can be access from control applications.

Some icons like images, containers can be deleted by dragging them to trash bin: .

3.1.4 To Change Values

Values can be changed in various ways. They can be entered manually and you can even perform calculations in there:



The mouse can be used to drag values. There are some modifiers to change the precision of your value:

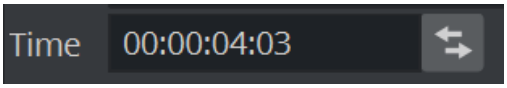
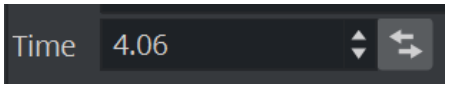
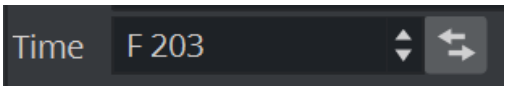
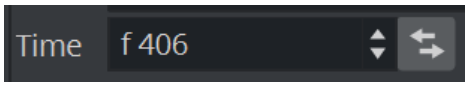
Function	Key and/or Mouse
Increase/Decrease very large steps (100)	**SHIFT + ALT +** Hold left mouse button and move
Increase/Decrease large steps (10)	**ALT +** Hold left mouse button and move
Increase/Decrease normal steps (1)	Hold left mouse button and move
Increase/Decrease small steps (0.1)	**CTRL +** Hold left mouse button and move
Increase/Decrease very small steps (0.01)	**SHIFT + CTRL +** Hold left mouse button and move

Copyright © 2026 Vizrt

3.1.5 Timecodes

When dealing with time based values, Viz Artist version 5.1 and later offer the possibility to change between timecode view and the classic absolute view. The classic absolute view depends on what the user has chosen to work with (Seconds, Frames or Fields). Depending on the output format (interlaced or progressive) the default timecode and step size are displayed in fields or frames.

To change, click the format switcher: 

Timecode view	**Seconds**
	
Frames (F x)	**Fields (f x)**
	

3.1.6 Conflicting Values

When using multi-selection, Viz Artist indicates conflicting values in spinboxes with a hyphen (**-**) character. A value is conflicting if the value is different in the individual selected elements and no common value can be displayed.



Copyright © 2026 Vizrt

3.2 Viz Artist Startup and Close

To start and run Viz Artist, a Graphic Hub must also be available. Viz Artist can also be started with certain options enabled or disabled, by passing selected Viz Command Line Options on start.

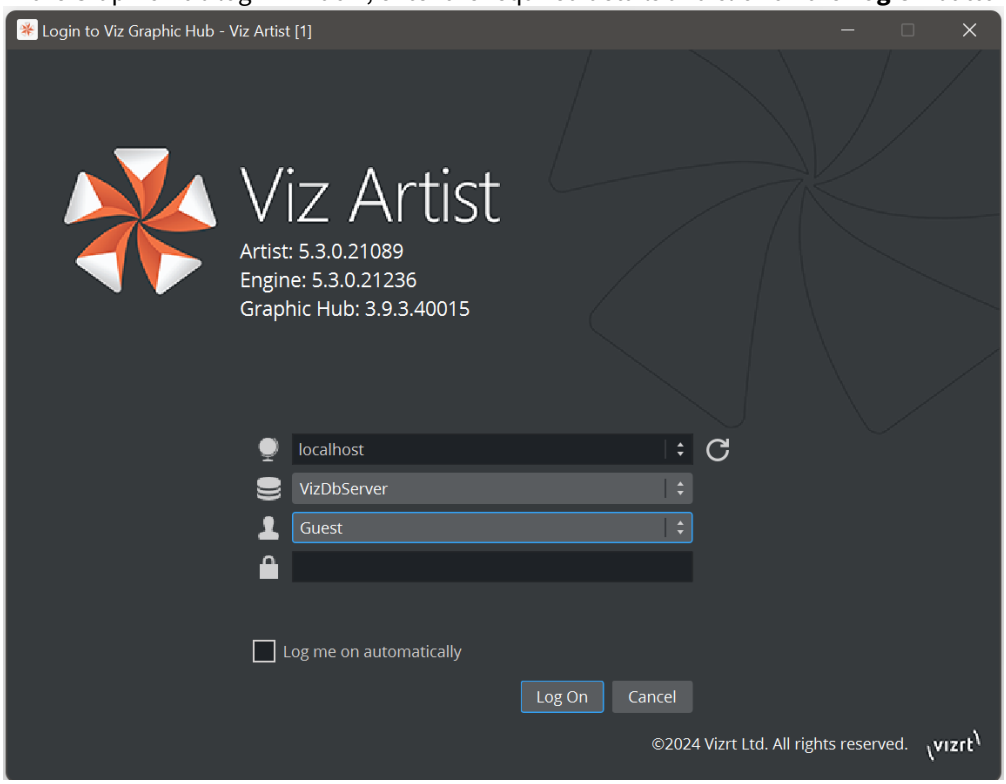
This section contains information on the following topics:

- Starting Viz Artist
- Viz Configuration
- To Add a Viz Artist Startup Option
- Close and Restart

3.2.1 Starting Viz Artist

To use Viz Artist, a connection to a Graphic Hub needs to be established. See Working with Graphic Hub for details.

1. Start Viz Artist.
2. In the Graphic Hub login window, enter the required details and click on the **Log On** button.



3.2.2 Viz Configuration

Viz is mainly configured by the Viz Configuration panel (see the **Configuring Viz** section of the Viz Engine Administrator Guide).

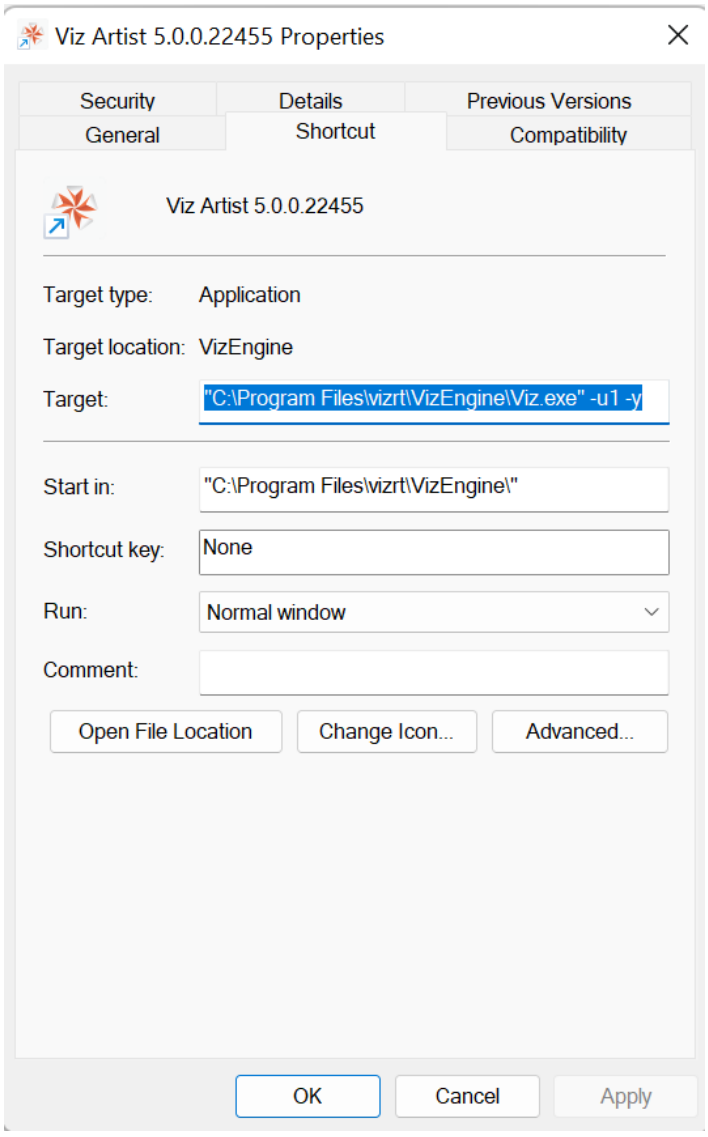
All configuration settings are stored in a configuration file, found in the *<viz data folder>*. The name of the Viz configuration is called: *VizEngine-{instance }.cfg*, where instance is a zero-based index of the instance being used.

IMPORTANT! Folder, File and Path Naming Convention: Do not use the Hash tag (**#**) in any folder, file or path name. If used in Folder, File and Path Names, text after a Hash tag is ignored.

3.2.3 To Add a Viz Artist Startup Option

1. Open the Viz Artist Properties panel:
   - Right click the desktop icon and select **Properties** in the context menu UNKNOWN_ATTACHMENT, or
   - Right click the **Viz Artist** shortcut under the Windows **Start** menu and select **Properties** in the context menu.
2. In the Properties window, apply a command in the **Target** field (see Viz Command Line Options).

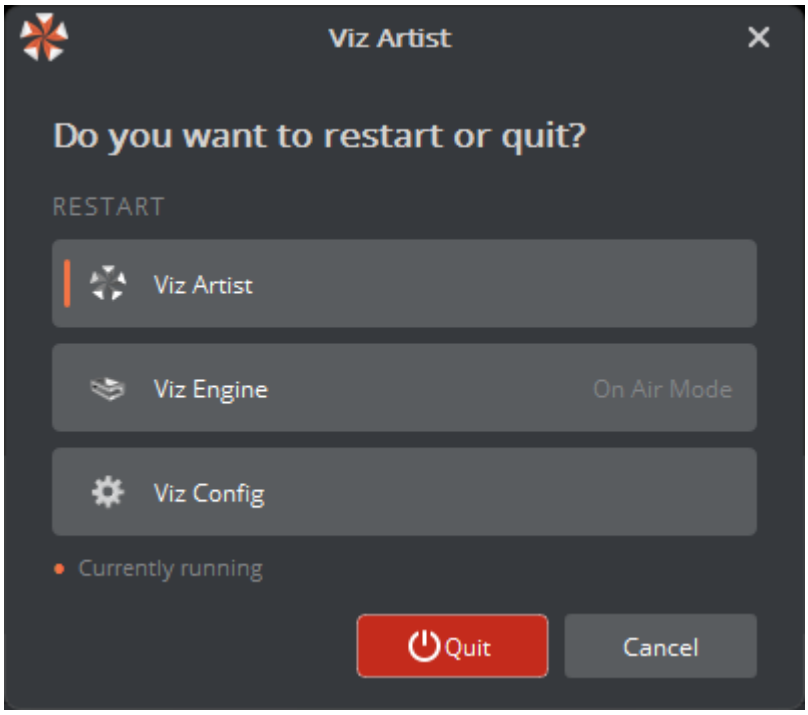
Example: `“<viz install folder> viz.exe” -n -u1 -y`



Copyright © 2026 Vizrt

3. Click **OK**.
4. Start Viz Artist.

3.2.4 Close and Restart



Sometimes a user wants to restart the application in a certain mode, e.g. after reconfiguring.
Beside simply closing the application (Quit), one can choose to restart either into

Viz Artist mode (Viz Engine + Viz Artist Gui)

Viz Engine (Viz Engine with OnAir GUI)

Viz Config (Viz Engine in Configuration mode)

The current mode is indicated with a small orange line.

> If you want to restart Viz Engine without any GUI (Console only) , press Shift and “Viz Engine”

See Also

- Viz Command Line Options
- **Configuring Viz** section of the Viz Engine Administrator Guide.

Copyright © 2026 Vizrt

3.3 Viz Artist Folders

This section details the location of the Viz Artist default installation and data folders.

3.3.1 Installation Folders

The default installation folder for Viz Artist is *C:\Program Files\Vizrt\VizArtist*.

In this User Guide, any reference to the Viz Artist installation folder is replaced with the text *<viz install folder>*.

> **Note:** Viz configuration files that already exist from a previous installation are automatically copied on install time if the installation folder did not change. If Viz Artist is installed in a different installation folder then Viz configuration files must be manually copied from the old installation folder to *<viz data folder>.*

3.3.2 Data Folders

Files which are created or modified by Viz Artist are located in *%ProgramData%\Vizrt\VizArtist*. By default, this is *C:\ProgramData\Vizrt\VizArtist*. This folder is referenced as *<viz data folder>* throughout this User Guide. The folder contents include:

- Viz configuration files,
- Script plug-ins,
- Crash dump files.

Script plug-ins are located in the *<viz data folder>\ScriptPlugins* subdirectory.

> **Note:** Script plug-in files that already exist from a previous installation are automatically copied on install time if the installation folder did not change. If Viz Artist is installed in a different installation folder, then Script plug-in files must be manually copied from the old installation folder to *<viz data folder>\ScriptPlugins*.

See Also

- **User Account Control (UAC)** page in the **Software Configuration** section of the Viz Engine Administrator Guide.

Copyright © 2026 Vizrt

3.4 Viz Artist Command Line Options

You can start Viz Artist and Viz Engine with various startup options. These command line options must be passed to the Viz executable file, *viz.exe*, and not to the batch file, *viz.cmd*. The batch file is only there for compatibility reasons. The table below details the most common Viz Command Line Options. You can get a list of all available startup options, by running `viz.cmd -h` or `-?` from the command line.

Option	Description
-B <path>, --temp_data <path>	Specify the path where Viz Engine stores its temporary data (see the **Viz Artist/Engine Folders** page in the **Software Configuration** section of the Viz Engine Administrator Guide).
-c	Start in Viz Configuration mode (see also -u1, -u2, -u3).
-C	Start without a console.
--db user:pw@server/name-server:port	Specify which Graphic Hub to connect to on Viz Engine startup. `password` can be omitted from the argument, but `user`, `server`, `name-server` and `port` must always be provided. **Example:** `viz.exe --db Guest:@VizDbServer/localhost:19396` In the example above, Viz Engine connects to a Graphic Hub server called *VizDbServer* (⚠ case sensitive) on the name-server `localhost` at port `19396`, as the user `Guest` with no password.
-g <config file>	Start with a predefined Viz Config file. This allows a User to have more than one startup option.
-h, -?, --help	Display the available options.
-i	Enable pre-initialization of textures. Textures are generated on the graphics card immediately after loading an image.
-l <title>	Specify a console title to distinguish Viz Engines in a Dual Channel setup (example: `-l "Channel 1"`).
-n	Start Viz Engine in On Air mode (see also -u1, -u2, -u3).

Copyright © 2026 Vizrt

Option	Description

-o <scene>
-o "<layer>
<scene>"
	Start Viz Engine in On Air mode with specified scene.
	Use the following syntax:
	-o SCENE*scene_id ... load scene_id into the main layer.
	-o "1 SCENE*scene_id" ... load scene_id into the main layer.
	-o "0 SCENE*scene_id" ... load scene_id into the back layer.
	-o "2 SCENE*scene_id" ... load scene_id into the front layer.
	
	Information: Layer supports only classic scene and both arguments need to be covered by quotation masks.
	Example:
	`viz.exe -o SCENE*Model/simple`
	`viz.exe -o "SCENE*<A5D05BF0-AE97-AD4B-986F72A607884F10>"`
	`viz.exe -o "0 SCENE*<A5D05BF0-AE97-AD4B-986F72A607884F10>"`

-P	Disable automatic mouse capture.
-t	Enables non-interactive mode for all occurrences except dongle issues and sound driver setup. The non interactive mode was introduced for use cases where there is no user and when there is no GUI one can interact with.
-T	Keep the Viz Engine Console always on top.
-u<number>, -u <number>, --instance <number> (from 1 up to a maximum of 24)	Instance number of Viz Engine. For systems with two or more graphic cards (Viz Trio One Box / Dual Channel). Specify GPU1 (-u1), GPU2 (-u2), etc., to run Viz Artist on. See Systems with Two or More GPUs below.
-v<level>, -v <level>	Enable verbose mode. This enables all possible information to be shown in the Viz Engine Console. The optional argument is a number composed by addition of: **1:** Produces verbose output to console. **2:** Adds a timestamp. **32:** Logs Graphic Hub related debug messages.
-w	Start in Engine mode and show render windows (videowall mode).
-W	Disable restart on crash.

Copyright © 2026 Vizrt

Option	Description
-x	Start without any GUI.
-X	Write extended dump file (full memory dump) in case of a program crash.
-y	Start in Artist mode. Viz Artist is started by Viz Engine after Viz Engine has started.
-Y <path>, --program_data <path>	Specify the path where Viz Engine stores its program data (see the **Viz Artist/ Engine Folders** page in the **Software Configuration** section of the Viz Engine Administrator Guide).

3.4.1 Systems with Two or More GPUs

It is possible to specify which GPU Viz Engine should run on, by providing the flags `-u1` and `-u2` for GPU1 and GPU2, respectively. A maximum of 24 GPUs are supported. This allows the system to run the Engine on one GPU while running Viz Config on another, for example:

```
<viz install folder>\viz.exe -c -u2
<viz install folder>\viz.exe -n -u1
```

More than one Viz Engine can run on one GPU (multiple Viz Engines can not run on the GPU with the same port number), use the formula m%n to calculate which Viz Engine instance to use (Viz Engine defined by -u<m>). Note, however, that Viz Artist can only run on Viz Engine instance 1 (-u1). Only the control GPU should have a display connected and is used for Artist/Preview.

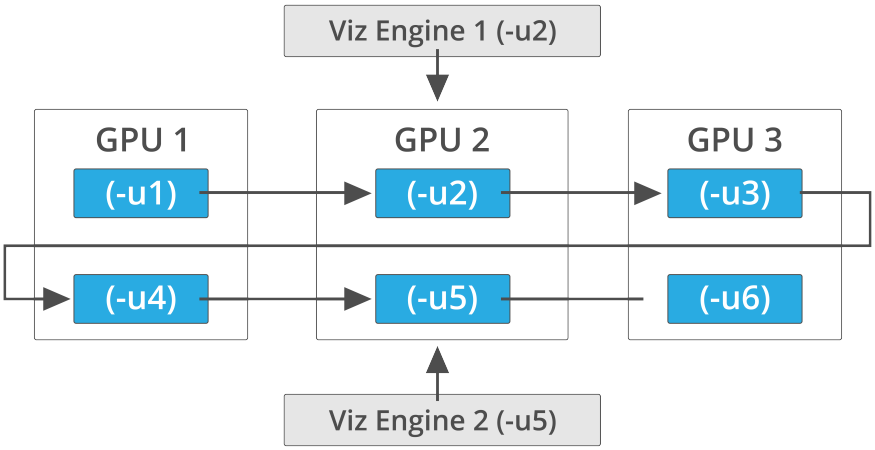
Formula: m%n, where:

- m = -u1/-u2/etc.
- n = number of GPUs

For example, with two Viz Engines:

- System with two GPUs: Engines run on GPU2: Engine 1 (-u2) / Engine 2 (-u4)
- System with three GPUs: Engines run on GPU2: Engine 1 (-u2) / Engine 2 (-u5)

Copyright © 2026 Vizrt

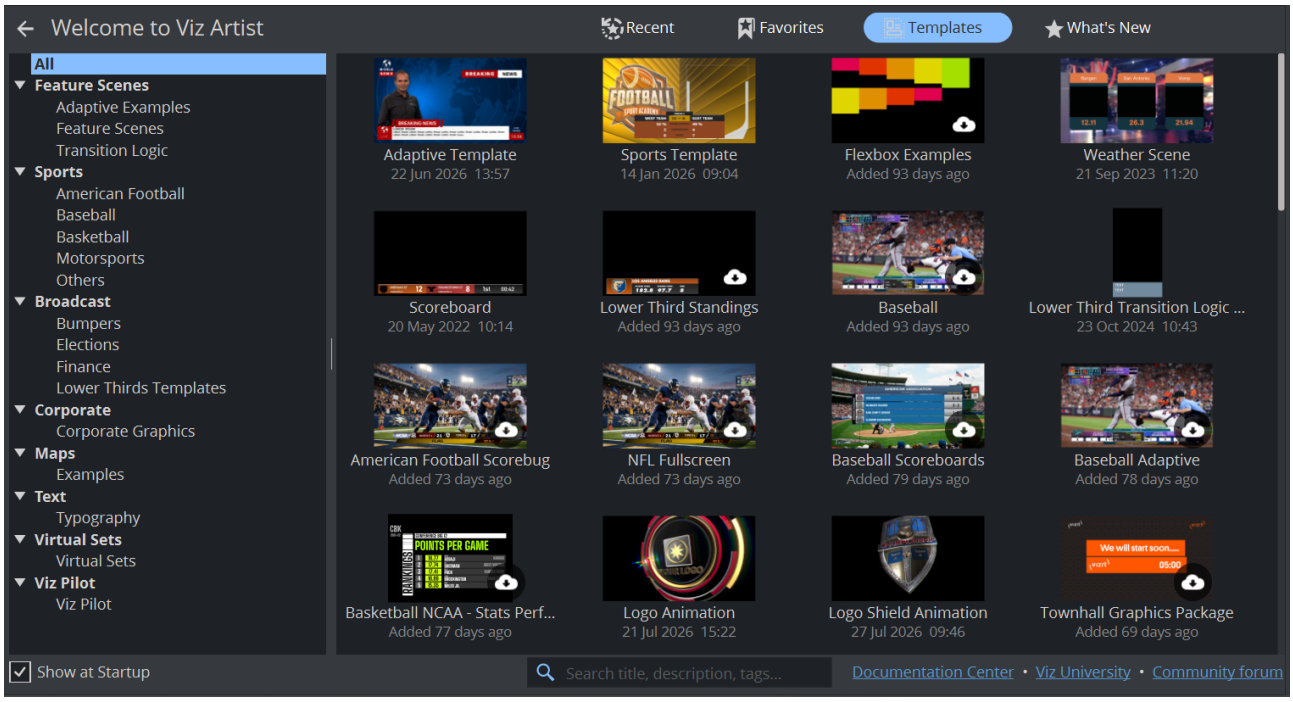


When either Trio One Box or Dual Channel Viz Artist versions are installed, GPUs are selected by default.

Copyright © 2026 Vizrt

3.5 Welcome Screen

The newly introduced Welcome Screen in Viz Artist offers the user to easily continue the latest work or to start something new by using some template.



The Welcome Screen is part of the Asset View and is opened by clicking on the Artist icon.
If will be shown by default. If you do not want to show this every time, uncheck the **“Show at Startup”**

The Screen shows 4 categories.

Recent:

A list of your recently opened scenes. Click on one and continue your work immediate without browsing your Graphic Hub.

Favorites:

If you have scenes marked as favorites, they will show up here

Templates:

See Using Templates

What’s New:

Shows a summary of this versions highlights and new features. Same as New in Viz Artist

On the bottom you’ll find links to the Documentation Center, Viz University and the Communiy Forum.

Copyright © 2026 Vizrt

3.5.1 Using Templates

Starting with Viz Artist 5.6 we are offering Templates, Example scenes and Starter Sets. These files are hosted by vizrt and are continuously updated.
The library offers learning scenes, examples as well as scenes showcasing new features (e.g. the new maps capabilities).

Templates in this library are provided by vizrt as starting points for your productions. You may freely use, modify, and adapt them within your organization and projects. Redistribution or resale of the templates as standalone assets is not permitted.

A single entry does not necessarily mean a single scene, it can also be a whole set of various scenes, for example a whole Transition Logic package.
Once new items are available for download, they will be marked. The library does a version check and disables library items not compatible with your version.

Note: Template library requires an active internet connection

Downloading



If a template shows the download icon, you are able to download it. Items which are already on your Graphic Hub can be opened immediately.
The files are being served by a CDN and stored locally as archive first, then imported into your current Graphic Hub.

After the download has finished, the file(s) will be imported in your Graphic Hub and you are able to load them immediately by clicking on it.

Templates are stored inside a **Template Folder** on your Graphic Hub. However some used fonts will be stored in **Fonts**

Searching

All Templates are linked with various Keywords and Tags. E.g. if you search for “Scoreboard” or “Lower Third” all items with these tags will show up.

Copyright © 2026 Vizrt

4 Viz Artist Interface Overview

This section gives an overview of the Viz Artist user interface and its main panels.

This section contains information on the following topics:

- Main Window
- Main Menu
- Button Bar

4.1 Main Window

When you start Viz Artist, the main window opens after you have logged on. This is the default view:



At the top of the main window is the Main menu bar. Below the Main menu bar is the display area. The content of the display area changes according to selections made in the Main menu.

You can move, dock/undock or resize the panel areas to accommodate your workflow.

At system start (default workspace), the area is divided into four panels:

Fig Item	Panel	Description
1	Asset View	From the Asset View panel, it is possible to browse for items such as Materials, Images, and Fontstyles, as well as search Graphic Hub.

Copyright © 2026 Vizrt

Fig Item	Panel	Description
2	Properties	Properties for Containers, Cameras, Lights and Video Clip Playback Considerations are controlled from the **Property** panel, where size, position and other settings can be modified as required.
3	Scene Editor	A visualization of the Scene is shown in the Scene Editor panel.
4	Scene Tree	Items in the Graphic Hub database can be added to the Scene Tree, which is the logical visualization of a Scene. The Scene Tree is built up of Containers, which can hold items such as Materials or plug-ins.
5	Built-Ins	Shows plug-ins and media assets.
6	Main Menu	Display of views, configuration settings, and import/export.

4.2 Main Menu

Use the buttons in the Main menu to select what is displayed in the display area.



Viz Artist 5.6 introduced a new borderless unified titlebar. If you want to switch to the classic windows title bar, you can do so in the Config settings / User Interface.

4.2.1 Default

This is the interface that shows when Viz Artist is loaded for the first time.

Shows the following panels:

- Asset View
- Properties
- Scene Editor
- Scene Tree
- Plugins

4.2.2 Workspaces

By default the following workspaces are pre-defined. For more information, please see Docking and Workspaces.

Copyright © 2026 Vizrt

Views

The Views area shows three configurable camera views, in addition to the one in the Scene Editor area. The additional camera views make it possible to monitor the scene from up to four different angles simultaneously.

Stage

The Stage is used to control the animations in the scene. The Stage shows the time-line from animation start (left side) to stop (right side).

Shows the following panels:

- Stage
- Scene Tree
- Scene Editor

Scripting

Scripting opens the Script Editor.

4.2.3 Import

Click to open the Import panels. Use these panels to transfer archives and various files into the Graphic Hub database. For more information, see the Import, Post and Archive chapter.

4.2.4 Archive

Click to open the Archive panels. Use these panels to create an archive of multiple database items into a single file, which is saved to the hard drive. For more information, see the Archives chapter.

4.2.5 Post

Opens the Post Rendering panel. From this panel it is possible to export image chains or Scene clips (see Post Render Scenes).

4.2.6 On Air

Opens the On Air panel. This panel switches the application mode from a 3D modeling tool to a plain rendering engine. The application waits for control commands through its communication port, or it can be controlled manually from the control buttons when Viz Artist is set in On Air Mode.

By default, the port number is set to `6100`. The port number and other communication settings can be modified in the **Communication** section of Viz Configuration (see the **Configuring Viz** section of the Viz Engine Administrator Guide).

Copyright © 2026 Vizrt

4.2.7 Config

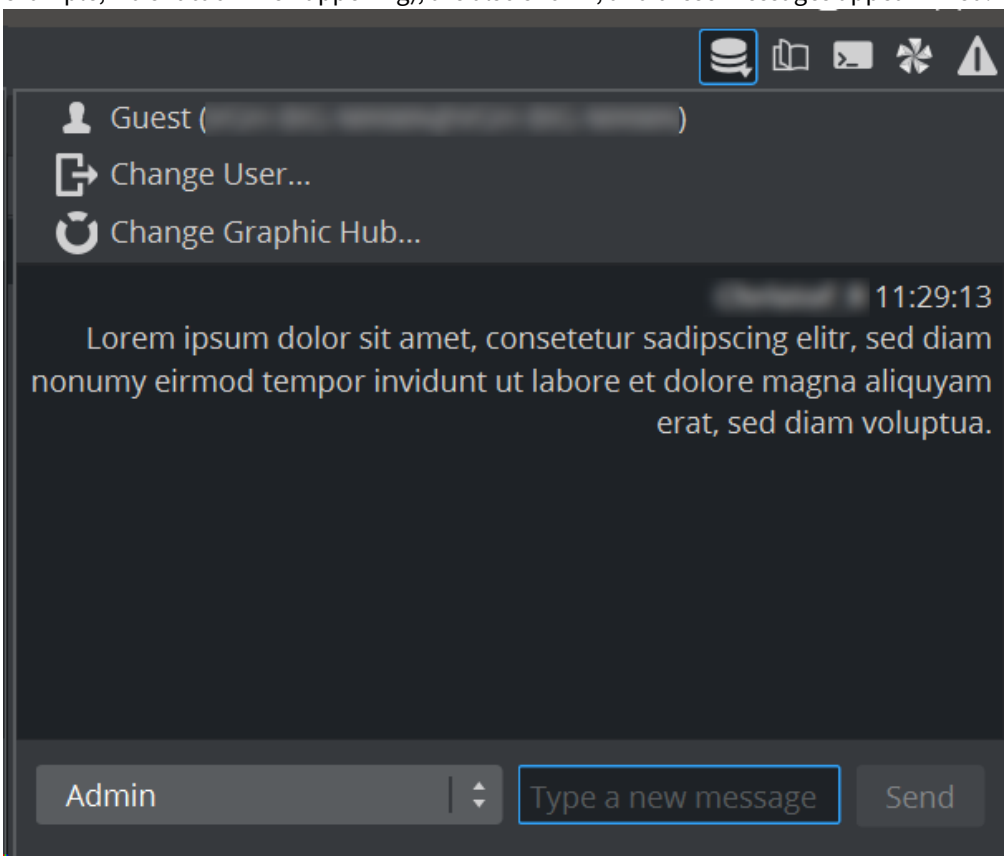
Click to open the Viz Configuration panel. Use this panel to change system options and functionality (see the **Configuring Viz** section of the Viz Engine Administrator Guide). Viz Configuration can also be opened as a separate application from the desktop or Start menu.

> **Note:** After any modifications in the Viz Configuration settings, Viz Artist must be restarted for the changes to take effect.

4.3 Button Bar



- **Server Menu:** Opens the server window. The Server Menu allows communication with other users or other groups as well as changing the current user or the selected database server. System messages (for example, if a shutdown is happening), are also shown, and these messages appear in red.



Select a logged-in user or a group, type your message and click on **Send**. To switch your user, click on **Change user...** and select a new user.

- **Documentation:** Opens the Viz Artist User Guide.
- **Show Commands:** Opens the Viz Engine Console window. This window shows the commands that are used for the communication between the user interface and the rendering Viz Engine. The console window is mainly used for debugging purposes.

Tip: The Viz Engine Console window can be opened by pressing the keyboard shortcut **ALT + C**.

- **License Information:** Opens the License Information and About window. This window shows the status of the current license, with details such as license expiration date, platform type, licensed plug-in packages, Viz Artist version number, and so on. This is also where the license can be renewed. For details on how to renew the license, see the Viz Engine Administrator Guide. Click the **About...** button to view the About statement and see all legal notices.

Copyright © 2026 Vizrt

4.4 Docking and Workspaces

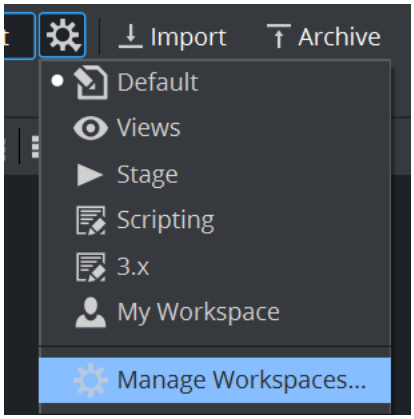
- Workspace Manager
    - List of Available Docks
- Working with Docks
- Documentation
- Render Views
    - Examples

The Viz Artist interface is composed of several windows (docks), which are organized into workspaces. Each workspace is a different collection of docks which are laid out at specific positions and sizes on the screen. Placing docks on multiple monitors is supported, so that screen real estate can be maximized for individual windows like the Scene Editor.

The position, size, and behavior of docks can be customized to create window layouts that work best for individual editing workflows. These custom layouts can be saved as named workspaces and you can then switch between layouts with a single command. For example, a layout for animating and a layout for scripting, and easily switching between them by using the Workspaces menu at the top left of the main user interface window.

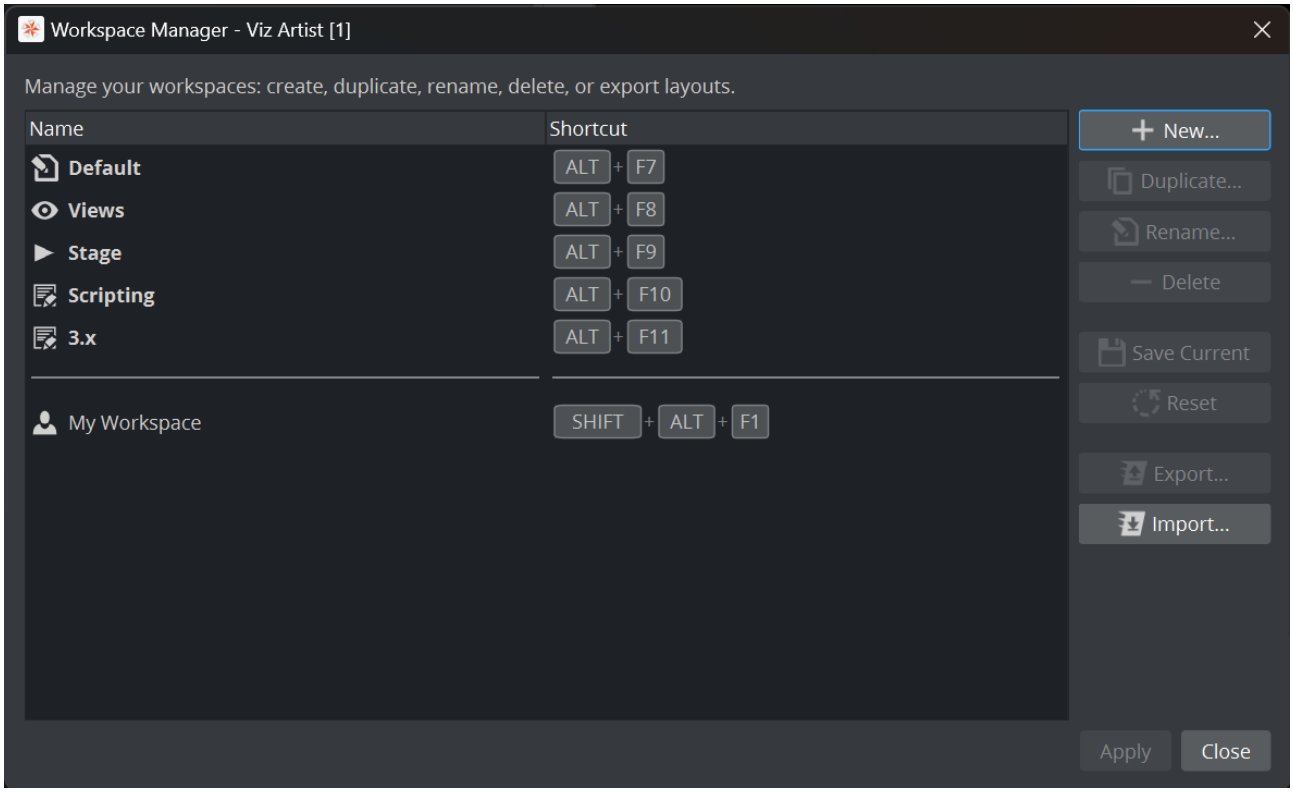
4.4.1 Workspace Manager

The Workspace Manager contains a set of predefined workspaces with saved layouts for the most common workflows in Viz Artist, a list of available docks and an option to manage workspaces. Open the Workspace Manager in the Docks Menu:



The Workspace Manager comes with a preset of layouts:

Copyright © 2026 Vizrt



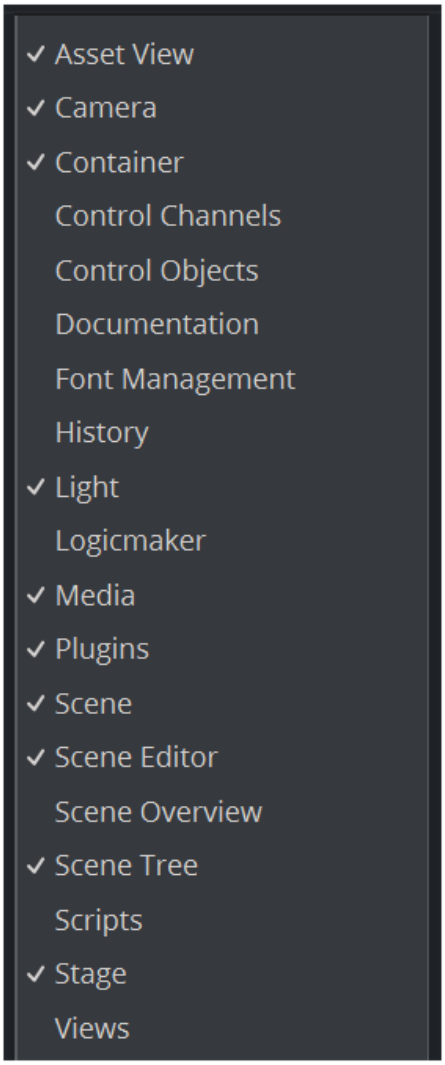
- **Default:** Opens the default workspace of Viz Artist.
- **Views:** Opens a workspace containing all views, the Scene Editor, Stage and Container/Camera/Light/Scene docks.
- **Stage:** Opens a workspace containing the Scene Tree, the Scene Editor, Stage and Container/Camera/Light/Scene docks.
- **Scripting:** Opens a workspace with a large area dedicated to Script editing.
- **3.x:** Opens a workspace resembling the Viz Artist version 3 layout.

You can create your **new workspace** within the Workspace manager. Create a new layout, modify the docks layout to your personal needs and click on **Save Current**.

Layouts can be ex- and imported again.

Copyright © 2026 Vizrt

List of Available Docks



This menu contains all available docks of the Viz Artist user interface, which can be toggled by selecting the corresponding item. Visible and active docks are indicated with a check mark.

4.4.2 Working with Docks

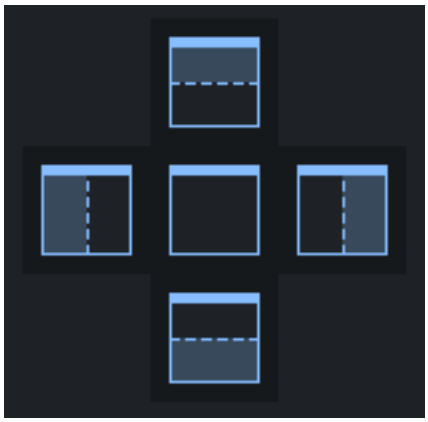
Each dock contains several user interface elements to control its appearance.



Copyright © 2026 Vizrt

- **1:** A tab to detach the dock, allowing it to be dragged around and dropped into a new position.
- **2:** Integrated documentation, click on it to open the documentation window showing the related information.
- **3:** A button to detach the dock and put it into its own floating window.
- **4:** A close button to close the entire group of docks.

Clicking and holding the mouse button on the tab (1) detaches the dock into its own floating window and displays a positioning helper widget.



These controls allow the dock being dragged to be placed to the north, east, south, and west of the dock under the current mouse position. By selecting the center item, the dock is placed as a new tab in the same group. If none of these controls are selected for the dock being dragged, the dock stays in its own floating window, which can be placed anywhere on the screen, even on a separate monitor. Once a dock is in floating window state, these position controls can be brought up again by clicking and dragging the window title bar.

Tip: When a dock in floating window state is closed, it can be reopened using the **List of available Docks** menu item in the **Workspaces menu**.
Help icon is currently only available on docked windows.

4.4.3 Documentation

Starting with Viz Artist 5.1, the offline documentation is integrated into the Artist User interface.

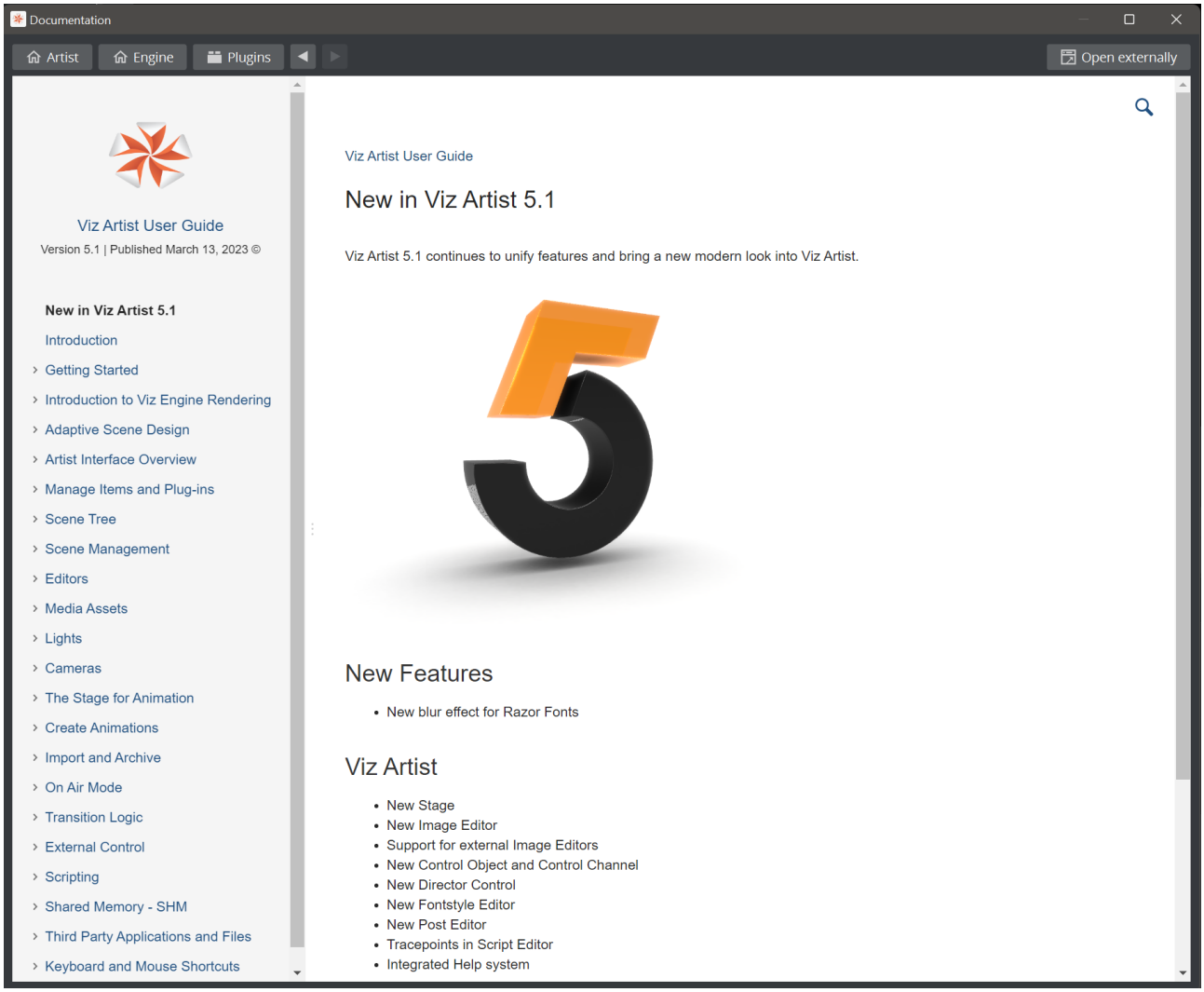
Open the documentation on any page by clicking on the help icon: . The documentation window jumps to the desired location, depending from where the help was opened.

The documentation window allows to browse within following documents (if installed):

- Viz Artist Documentation
- Viz Engine Administrator Guide
- Viz Plug-in Documentation

Information: The web-based Plug-in documentation is installed by Viz Artist.

Copyright © 2026 Vizrt



Use the top bar to switch between the various documents:

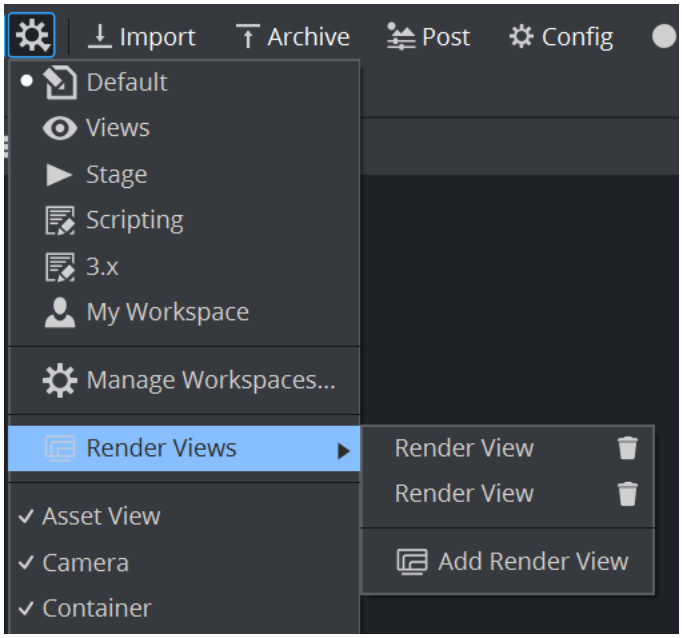


Open externally starts your default browser and opens the current document.

4.4.4 Render Views

Individual Render Windows can be created. They can be used for Adaptive Scene Design to display more than one format at the same time. They can also be used to show different camera shots in traditional three-dimensional scenes. To create a new Render Window, select the **Workspace Menu** and **Render Views**.

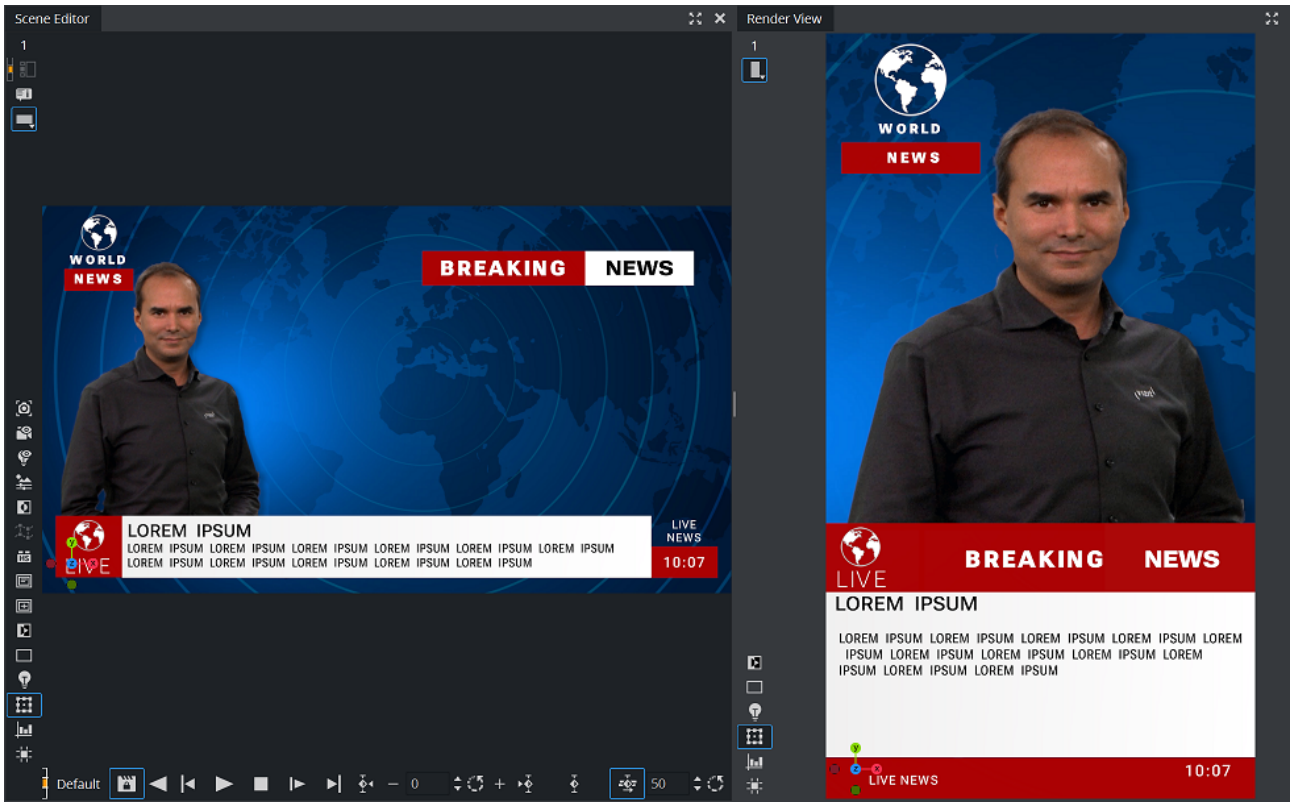
Copyright © 2026 Vizrt

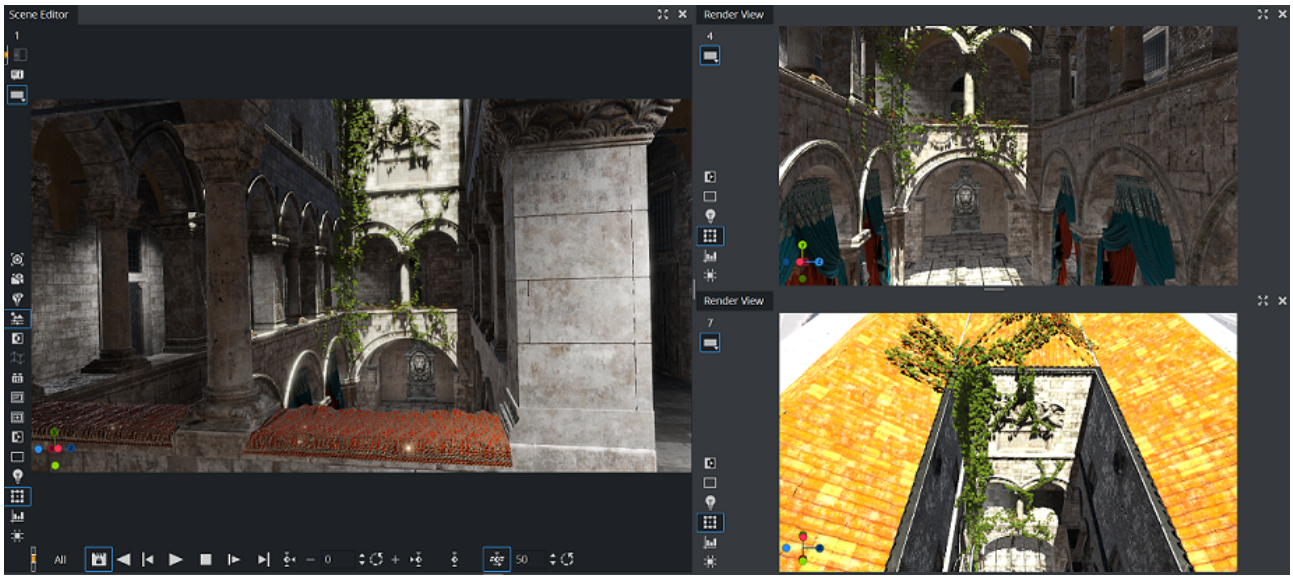


Click **Add Render View** to create a new view, click on an existing one to open it or use the trash bin icon to delete it. Render Views are docking windows, therefore they can either be used as floating windows or be docked into your working environment.

Copyright © 2026 Vizrt

Examples





Note: Picking is not supported in Render Views.

Copyright © 2026 Vizrt

4.5 Keyboard and Mouse Shortcuts

This section lists various keyboard and mouse shortcuts which can be used in Viz Artist. When a keyboard shortcut is used, always make sure that the keyboard **CAPS LOCK** is set to `Off`, unless otherwise stated. Shortcuts can be modified and exported for personalization.

This section contains information on the following topics:

- Application Controls and Shortcuts
- Asset View Shortcuts
- Import Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts
- Logicmaker Shortcuts

4.5.1 Shortcut Help

If you are unsure which shortcut to use in a certain editor, press **CTRL + F1** and all available shortcuts in this editor are shown.

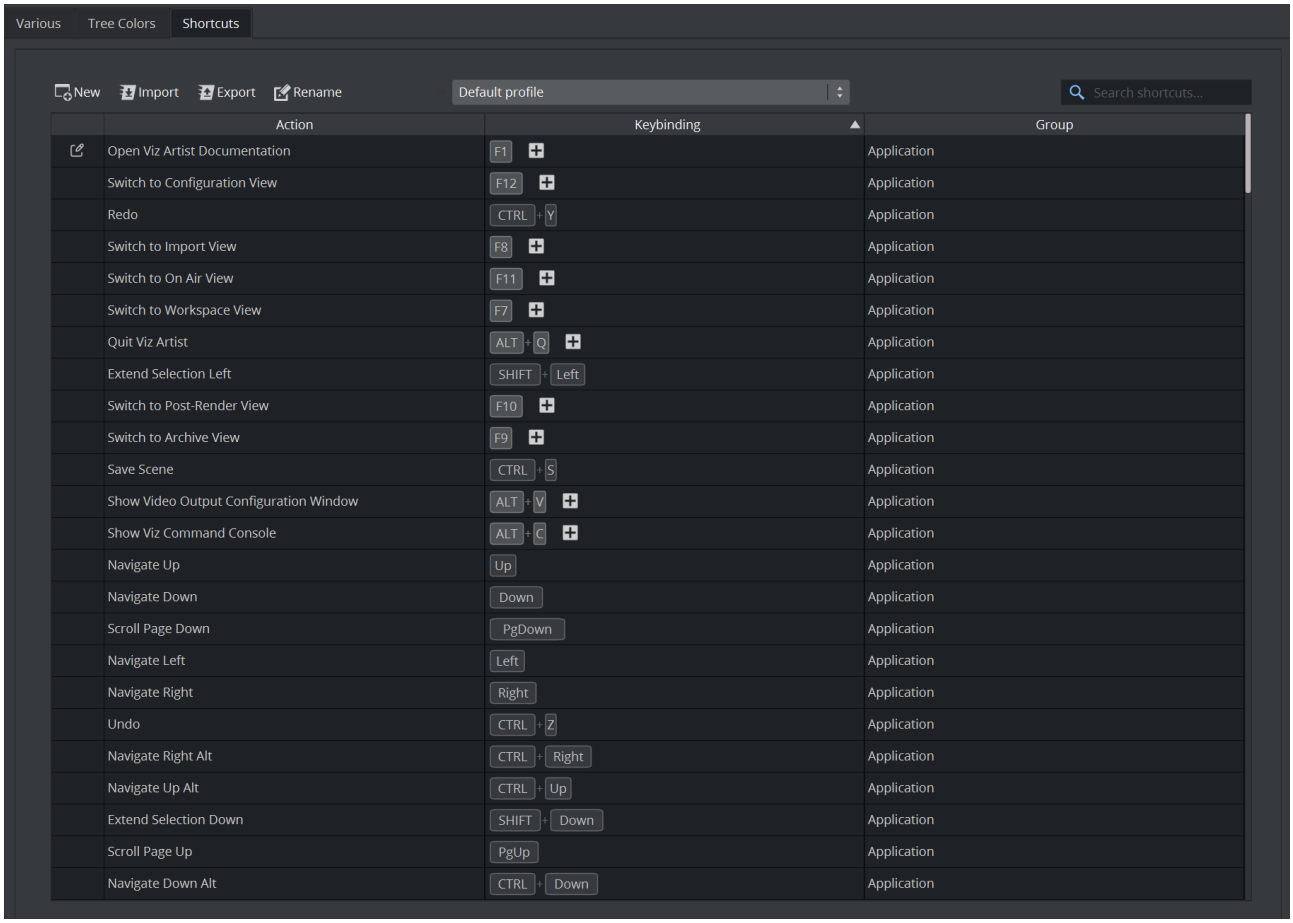
> **Note:**
> - Some shortcuts cannot be modified, they are displayed in a slighter darker grey.
> - Keybindings without modifier (**CTRL**, **ALT** or **SHIFT**) are not allowed (Exceptions: Function keys, Stage and Scene/Render Editor group).

Shortcuts are grouped by their editors:

- General Application
- Archive View
- Asset View
- Import
- On Air + Render Editor
- Plugin List View
- Scene Editor
- Scene Tree
- Script Editor
- Stage
- Stage Tree

Copyright © 2026 Vizrt

Note: On Air (Viz Artist) and Render Editor (Viz Engine) share the same functionality, however Render Editor shortcuts can be mapped different to achieve different behavior (for example, in videowall mode).

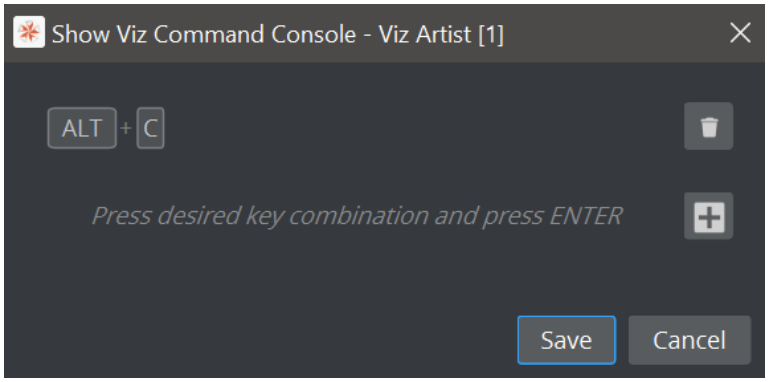


4.5.2 To Edit a Shortcut

Open **Configuration** > **User Interface** to see a list of all available shortcuts and their current mappings.

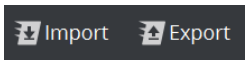
If a shortcut can be modified, the icon is shown in the left row. Select it to show the dialog, or double click on the entry.

Copyright © 2026 Vizrt



If you want to add an additional shortcut, press the desired combination and press the **Add** ( ) button. To remove a shortcut, press the **Trash** ( ) button.

4.5.3 To Export and Import Key Mappings



If you prefer your own key mapping, the current one can be exported as a JSON file and imported on another system.

Note: Key mappings are system based, not user based. They are loaded from *%ProgramData%\vizrt\VizArtist\ShortcutProfiles*.

4.5.4 Application Controls and Shortcuts

Function	Default Key and/or Mouse Shortcut	Configurable
Opens Documentation	F1	✓
Displays Context Menu	Right mouse button click	✗
Quits Viz Artist	ALT + Q	✓
	ALT + F4	✓
Shows/Hides Viz Command Console	ALT + C	✓
Shows Video Output Configuration Window	ALT + V	✓
Shows Workspace View	F7	✓
Shows Import Window	F8	✓
Shows Archive Window	F9	✓
Shows Post Render View	F10	✓
Shows On Air Window	F11	✓
Shows Viz Configuration Panel	F12	✓
Switches to Default Workspace	ALT + F7	✓
Switches to Views Workspace	ALT + F8	✓
Switches to Stage Workspace	ALT + F9	✓
Switches to Scripting Workspace	ALT + F10	✓
Switches to 3.x Workspace	ALT + F11	✓
Quits Viz Engine	ALT + Q	✓
	ALT + F4	✗

Copyright © 2026 Vizrt

Function	Default Key and/or Mouse Shortcut	Configurable
Saves Scene	CTRL +S	✓
Undoes action	CTRL + Z	✗
Redoes action	CTRL + Y	✗
Extends Selection Left	SHIFT + LEFT ARROW (←)	✗
Extends Selection Right	SHIFT + RIGHT ARROW (→)	✗
Extends Selection Up	SHIFT + UP ARROW (↑)	✗
Extends Selection Down	SHIFT + DOWN ARROW (↓)	✗
Navigates Left	LEFT ARROW (←)	✗
	CTRL + LEFT ARROW (←)	✗
Navigates Right	RIGHT ARROW (→)	✗
	CTRL + RIGHT ARROW (→)	✗
Navigates Up	UP ARROW (↑)	✗
	CTRL + UP ARROW (↑)	✗
Navigates Down	DOWN ARROW (↓)	✗
	CTRL + DOWN ARROW (↓)	✗
Scrolls Page Up	PAGE UP	✗
Scrolls Page Down	PAGE DOWN	✗
Selects First	HOME	✗
Selects Last	END	✗
Selects All	CTRL + A	✗

Copyright © 2026 Vizrt

Function	Default Key and/or Mouse Shortcut	Configurable
Extends Selection Up	**SHIFT + UP**	✗
Extends Selection Down	**SHIFT + DOWN**	✗
Extends Selection Right	**SHIFT + RIGHT**	✗
Extends Selection Left	**SHIFT + LEFT**	✗
Toggles Selection	**CTRL + SPACE**	✗

See Also

- Asset View Shortcuts
- Import Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts

Copyright © 2026 Vizrt

4.5.5 Asset View Shortcuts

The following shortcuts are valid for the Asset View:

Action	Default Key and/or Mouse Shortcut	Configurable
Renames an item	F2	✓
Focuses on the search input field	CTRL + F	✓
Copies the UUID of the item to the clipboard	CTRL + U	✓
Copies the path of the item to the clipboard	CTRL + SHIFT + U	✓
Checks in an item	CTRL + I	✓
Checks out an item	CTRL + O	✓
Adds a new scene or classic material (if the type is selected)	CTRL + N	✓
References search on this item	CTRL + R	✓
Exports file (if supported)	CTRL + E	✓
Increases Icon Size	CTRL + + CTRL + Mousewheel	✓
Decreases Icon Size	CTRL + - CTRL+ Mousewheel	✓
Copies	CTRL + C	✓
Cuts	CTRL + X	✓
Pastes	CTRL + V	✓
Shows icon view	CTRL + SHIFT + 1	✓
Shows list view	CTRL + SHIFT + 2	✓
Shows detail view	CTRL + SHIFT + 3	✓

Copyright © 2026 Vizrt

Action	Default Key and/or Mouse Shortcut	Configurable
Selects all items	**CTRL + A**	✓
Shows properties	**CTRL + P**	✓
Duplicates file (needs to be copied first)	**CTRL + SHIFT + V**	✓
Quick Look	**SPACE**	✓

See Also

- Application Controls and Shortcuts
- Import Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts

Copyright © 2026 Vizrt

4.5.6 Import Shortcuts

This section contains the following topics:

- File Shortcuts
- Archive Shortcuts

File Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Switches to Import View	F8	✓
Selects Fonts for Import	CTRL + 1	✓
Selects Images for Import	CTRL + 2	✓
Selects Geometries for Import	CTRL + 3	✓
Selects Scenes for Import	CTRL + 4	✓
Selects Audio for Import	CTRL + 5	✓
Selects Substances for Import	CTRL + 6	✓
Selects Archives for Import	CTRL + 7	✓

Archive Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Switches to Archive View	F9	✓
Shows Archive Scenes	CTRL + 1	✓
Shows Archive Geometries	CTRL + 2	✓
Shows Archive Materials	CTRL + 3	✓
Shows Archive Materials Advanced	CTRL + 4	✓

Copyright © 2026 Vizrt

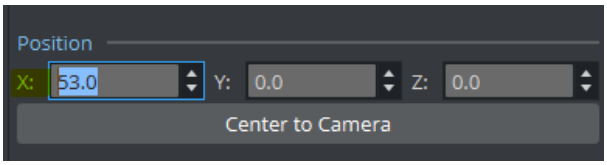
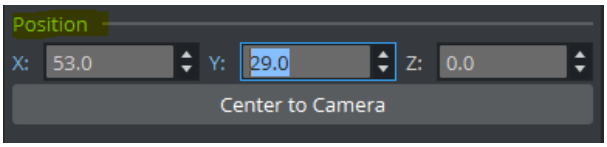
Function	Key and/or Mouse Shortcut	Configurable
Shows Archive Images	**CTRL + 5**	✓
Shows Archive Fonts	**CTRL + 6**	✓
Shows Archive Video Clips	**CTRL + 7**	✓
Shows Archive Substances	**CTRL + 8**	✓
Shows All Archive files	**CTRL + 9**	✓
Displays Entry Point View	**CTRL + E**	✓
Displays Folder View	**CTRL + O**	✓

See Also

- To Import Selected Files
- Application Controls and Shortcuts
- Asset View Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts

Copyright © 2026 Vizrt

4.5.7 Integer Float Controls

Function	Key and/or Mouse Shortcut	Configurable
Increases/Decreases very large steps (100)	**SHIFT + ALT +** Hold left mouse button and move	
Increases/Decreases large steps (10)	**ALT +** Hold left mouse button and move	
Increases/Decreases normal steps (1)	Hold left mouse button and move	
Increases/Decreases small steps (0.1)	**CTRL +** Hold left mouse button and move	
Increases/Decreases very small steps (0.01)	**SHIFT + CTRL +** Hold left mouse button and move	
Resets item	Click the **Blue label** 	
Resets group	Click the **Blue Group label** 	

See Also

- Application Controls and Shortcuts
- Asset View Shortcuts
- Import Shortcuts
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts

Copyright © 2026 Vizrt

4.5.8 On Air Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Switches to On Air	F12	✓
Returns to Workspace Pane	CTRL + Q	✓
	ESC	✓
Toggles VGA Preview	CTRL + P	✓
	SHIFT + BACKSPACE	✓
Toggles Full-Screen Preview	CTRL + F	✓

See Also

- On Air Mode
- Application Controls and Shortcuts
- Asset View Shortcuts
- Import Shortcuts
- Integer Float Controls
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts

Copyright © 2026 Vizrt

4.5.9 Polygon Plug-in Editor Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Opens Shortcut list	**CTRL + F1**	⊗
Sharpens Edge of Selected Point or Points	**SHIFT** + Left click	⊗
Deletes Selected Point(s)	**DELETE**	⊗
Adds New Anchor Point	Left click	⊗
Adds new Bezier point	Hold Left mouse button down and drag	⊗
Creates New Spline	**INSERT** + Left click	⊗
Opens/Closes Active Spline	**C**	⊗
Closes Spline	Click on the first point to toggle open / closed	⊗
Enables/Disables Spline Info	**H**	⊗
Moves Active Spline	**ALT**	⊗
Copies	**CTRL + C**	⊗
Cuts	**CTRL + X**	⊗
Pastes	**CTRL + V**	⊗
Selects Point or Spline	**S**	⊗
Moves Anchor point	**CTRL** + Left mouse button	⊗
Moves handles	**CTRL** + Left mouse button	⊗
Moves single handle	**SHIFT** + Left mouse button	⊗
Removes an Anchor point	Click on existing point with Left mouse button	⊗
Provides Rectangular selection	Hold **S** + Left mouse button down, and drag	⊗

Copyright © 2026 Vizrt

Function	Key and/or Mouse Shortcut	Configurable
Converts straight point to Bezier	**SHIFT +** Left mouse button drag on point	

Tip: Make sure the Handle Event (HS) button is turned off to allow the shortcuts to work.

See Also

- Application Controls and Shortcuts
- Asset View Shortcuts
- Import Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts

Copyright © 2026 Vizrt

4.5.10 Scene Editor Shortcuts

This section contains the following topics:

- General Shortcuts
- Camera Controls and Shortcuts
- Orthogonal View Controls and Shortcuts
- Animation Controls and Shortcuts
- Inline Text Editor

General Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Adds new Key Frame in Stage of selected container	**ENTER**	✓
	,	✓
Copies container(s)	**CTRL** + Left click and drag	✗
Deletes container	**DELETE**	✓
	BACKSPACE	✓
Cancels selection of one out of multiple selected Containers	**CTRL** + Left click	✗
Selects multiple Containers	**CTRL** + Left click	✗
Selects child container	**RIGHT ARROW** (→)	✓
Selects next container on same tree level	**DOWN ARROW** (↓)	✓
Selects parent container	**LEFT ARROW** (←)	✓
Selects previous container on same tree level	**UP ARROW** (↑)	✓
Forces selection lasso (Classic Render Pipeline only)	**S** + draw mouse selection rectangle	✓
Selects all container	**CTRL + A**	✓

Copyright © 2026 Vizrt

Function	Key and/or Mouse Shortcut	Configurable
Unselects all container	**SPACE**	✓
Switches additional performance information	Middle mouse click on (Classic Render Pipeline only)	✗
Toggles bounding box on/off	**B**	✓
Toggles grid visibility on/off	**G**	✓
Adjusts Grid size	**ALT + G + (Left mouse button drag)**	✓
Zooms in	**CTRL +** Plus sign on the numeric keypad	✓
	Z	✓
	Mouse Wheel	✗
Zooms out	**CTRL +** Minus sign on the numeric keypad	✓
	X	✓
	Mouse Wheel	✗
Translate selected object (VER pipeline only)	**ALT + S**	✓
Forces single object selection on (on groups only) (Classic Render Pipeline only)	**F**	✓
Opens Shortcut help	**CTRL + F1**	✓
Switches to On Air	**F12**	✓

Camera Controls and Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Switches Camera from 1 to 10	**1 - 0**	✓

Copyright © 2026 Vizrt

Function	Key and/or Mouse Shortcut	Configurable
Switches to Back Camera	Numeric Keypad **1**	✓
Switches to Bottom Camera	Numeric Keypad **2**	✓
Switches to Front Camera	Numeric Keypad **7**	✓
Switches to Left Camera	Numeric Keypad **4**	✓
Switches to Right Camera	Numeric Keypad **6**	✓
Switches to Top Camera	Numeric Keypad **8**	✓
Camera Position Mode (X, Y, Z)	**P +** Left mouse button click and drag (left-right) **P +** Middle mouse button click and drag (up-down) **P +** Right mouse button click and drag (up-down)	✓
Camera Rotation Mode (Pan, Tilt, Twist)	**I +** Left mouse button click and drag (Pan) **I +** Left mouse button click and drag (Tilt) **I +** Left mouse button click and drag (Twist)	✓
Camera Orbit Mode	**O +** Right mouse button click and drag (Orbit in X) **O +** Left mouse button click and drag (Orbit in Y) **O +** Middle mouse button click and drag (Orbit in Z)	✓
Camera Target Mode (Set selected container for orbit target)	**T**	✓
Resets Camera Transformation	**R**	✓
Zooms Camera	**U +** Left mouse button click and drag (left-right)	✓
Copy current view to camera 'n'	**ALT + 'n'**	✓
Maps Mode. Used to nvagiate within GeoCanvas objects	**M**	✗

Copyright © 2026 Vizrt

Orthogonal View Controls and Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Show whole Scene in Viewport	**V**	✓
Pans	Left mouse button click and drag on background	✗
Rubberband Zoom In	**Q** + Left mouse button click and drag	✓
Rubberband Zoom Out	**W** + Left mouse button click and drag	✓

Animation Controls and Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Manipulates single handle	**CTRL** + Left mouse button click handle and drag	✗
Mirrors Handles	**SHIFT** + Left mouse button click handle and drag	✗
Resets handle of selected Key Frame in animation path	**D**	✓

Inline Text Editor

Function	Key and/or Mouse Shortcut	Configurable
Copies	**CTRL + C**	✗
Cuts	**CTRL + X**	✗
Pastes	**CTRL + V**	✗
Redoes action	**CTRL + Y**	✗
Selects all	**CTRL + A**	✗

Copyright © 2026 Vizrt

Function	Key and/or Mouse Shortcut	Configurable
Undoes action	**CTRL + Z**	✗

See Also

- Application Controls and Shortcuts
- Asset View Shortcuts
- Import Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts

Copyright © 2026 Vizrt

4.5.11 Scene Tree Shortcuts

This section contains the following topics:

- General Shortcuts
- Basic View Controls and Shortcuts
- Plug-in Overlay

General Shortcuts

Function	Default Key and/or Mouse Shortcut	Configurable
Moves to Top Container	**HOME**	✗
Moves to Last Container	**END**	✗
Moves to top of page	**PAGE UP**	✗
Moves to bottom of page	**PAGE DOWN**	✗
Scrolls up the Container list	**UP ARROW (↑)**	✗
Scrolls down the Container list	**DOWN ARROW (↓)**	✗
Opens Container tree	**RIGHT ARROW (→)**	✗
Closes Container tree	**LEFT ARROW (←)**	✗
Opens plug-in panel	**+**	✗

Basic View Controls and Shortcuts

Function	Default Key and/or Mouse Shortcut	Configurable
Inserts New Container	**CTRL + INSERT** (The new Container is created above the first selected Container.)	✓
Rename Container	**F2**	✓
Multiselects Containers	**CTRL +** Left click	✗

Copyright © 2026 Vizrt

Function	Default Key and/or Mouse Shortcut	Configurable
Deletes selected Container(s)	**CTRL + DELETE**	✓
Opens whole Tree	**CTRL + O**	✓
Opens Subtree	**ALT +** +	✓
	ALT + RIGHT	✗
Collapses whole Tree	**CTRL + I**	✓
Collapses Subtree	**ALT +** -	✓
	ALT + LEFT	✗
Locks selected Container(s)	**CTRL + L**	✓
Unlocks selected Container(s)	**CTRL + SHIFT + L**	✓
Groups selected Container(s)	**CTRL + G**	✓
Ungroups selected Container(s)	**CTRL + SHIFT + G**	✓
Hides selected Container(s)	**CTRL + H**	✓
Shows selected Container(s)	**CTRL + SHIFT + H**	✓
Searches Container	**CTRL + F**	✓
Copies selected Container(s)	**CTRL +** Left click on container name and drag	✗
Copies selected Container(s) to Clipboard	**CTRL + C**	✓
Pastes Containers from Clipboard	**CTRL + V** (They are created one hierarchical level underneath selected container.)	✓
Cuts selected Container(s)	**CTRL + X**	✓
Moves selected Container(s)	Left click on container name and drag	✗

Copyright © 2026 Vizrt

Function	Default Key and/or Mouse Shortcut	Configurable
Moves selected Container and retain transformation properties	**ALT** + Left click on container name and drag	✗
Copies Properties of Container	Left click on container Properties and drag	✗
Merges Group	**CTRL + M**	✓
Splits Container	**CTRL + SHIFT + M**	✓
Convert to Dynamic Geometry	**CTRL + D**	✓

Plug-in Overlay

Function	Key and/or Mouse Shortcut	Configurable
Opens plug-in Panel	**+**	✗
Selects plug-in	**Cursor Keys**	✗
Selects plug-in and applies to selected container(s)	**ENTER**	✗
Leaves plug-in Overlay	**ESC**	✗

See Also

- Application Controls and Shortcuts
- Asset View Shortcuts
- Import Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Script Editor Shortcuts
- Stage Shortcuts

Copyright © 2026 Vizrt

4.5.12 Script Editor Shortcuts

Function	Key and/or Mouse Shortcut	Configurable
Copies and Pastes	Click the Middle mouse button to paste the selected text at the Mouse pointer location if enabled in the Viz Engine Configuration's User Interface section.	
Searches for Word	**F3**	✓
Searches for Selection	**F4**	✓
Comments Selection	**F5**	✓
Uncomments Selection	**F6**	✓
Compiles and Runs Script	**CTRL + R**	✓

See Also

- Application Controls and Shortcuts
- Asset View Shortcuts
- Import Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Stage Shortcuts
- The **Configuring Viz** section of the Viz Engine Administrator Guide

Copyright © 2026 Vizrt

4.5.13 Stage Shortcuts

This section contains the following topics:

- General
- Stage Tree
- Timeline
    - Animation
    - Timeline
    - Zooming
    - Searching
    - Keyframes
    - View

General

Function	Key and/or Mouse Shortcut	Configurable
Shows Object Editor	**CTRL + 1**	✓
Shows Dopesheet View	**CTRL + 2**	✓
Minimize Spline Editor Height	**SHIFT + PAGE DOWN** (on stage view focus)	✓
Increase Spline Editor Height	**SHIFT + ↑**	✓
Decrease Spline Editor Height	**SHIFT + ↓**	✓
Shows Spline View	**CTRL + 3**	✓
Maximize Spline Editor Height	**SHIFT + PAGE UP** (on stage view focus)	✓
Shows Dope/Spline View	**CTRL + 4**	✓

Copyright © 2026 Vizrt

Stage Tree

Function	Key and/or Mouse Shortcut	Configurable
Jumps to Top Container	**HOME**	✗
Jumps to Last Container	**END**	✗
Moves a page down in the tree	**PAGE DOWN**	✗
Moves a page up in the tree	**PAGE UP**	✗
Scrolls up the tree	**UP ARROW (↑)**	✗
Scrolls down the tree	**DOWN ARROW (↓)**	✗
Jumps to an item (Director, Actor, or Channel), in the tree	Press the first letter of the item name	✗
Opens or closes the selected tree	+ or **RIGHT ARROW (→)**	✗
	- or **LEFT ARROW (←)**	✗
Group Directors	**SHIFT + D**	✓
Ungroup Directors	**CTRL + SHIFT + D**	✓
Delete Selection	**DEL**	✓

Timeline

Animation

Function	Key and/or Mouse Shortcut	Configurable
Starts Animation	**ENTER**	✓
Starts Animation Backwards	**SHIFT + ENTER**	✓
Start Animation ignoring Stop points	**CTRL + ENTER**	✓

Copyright © 2026 Vizrt

Function	Key and/or Mouse Shortcut	Configurable
Start Animation ignoring Stop points Backwards	**CTRL + SHIFT + ENTER**	✓
Pause/Continues Animation Forward	**SPACE**	✓
Pause/Continues Animation Backwards	**SHIFT + SPACE**	✓
Pause/Continues Animation ignoring Stop points	**CTRL + SPACE**	✓
Pause/Continues Animation ignoring Stop points Backwards	**CTRL + SHIFT + SPACE**	✓

Timeline

Function	Key and/or Mouse Shortcut	Configurable
Sets timeline to 0 (If timeline does not fit)	**HOME**	✓
Sets View to End Director (If timeline does not fit)	**END**	✓
Set View to Time-line Start and move Marker	**CTRL + HOME**	✓
Sets Time-line Marker/Value to End (not when infinity is selected)	**CTRL + END**	✓
Set View to Time-line Start, Zoom Region (If timeline does not fit)	**SHIFT + HOME**	✓
Set View to Time-line End, Zoom Region (If timeline does not fit)	**SHIFT + END**	✓
Set View to Time-line Start, Zoom Region and move Marker	**CTRL + SHIFT + HOME**	✓
Set View to Time-line End, Zoom Region and move Marker (not when infinity is selected)	**CTRL + SHIFT + END**	✓

Copyright © 2026 Vizrt

Function	Key and/or Mouse Shortcut	Configurable
Move Timeline Marker	←/→ In 1080i50 format, moves the time-line marker five fields, left or right, and in 1080i60 the movement is six fields, left or right.	✓
	ALT + ←/→. In 1080i50 format, moves the time-line marker 50 fields, left or right, and in 1080i60 the movement is 60 fields, left or right.	✓
	CTRL + ←/→. Moves the timeline 1 filed left or right.	✓
Moves Time-line Marker to nearest Key Frame	**J**	✓
Brings View to Time-line Marker	**M**	✓
Jumps to next Key Frame	**TAB** (a Key frame must be highlighted first)	✓
Jumps to previous Key Frame	**SHIFT + TAB**	✓
Deletes Selected Key Frame(s)	**DELETE**	✓
Tween Mode (Director Scalemark)	**T** + start/end click on the timeline Allows to stretch a portion of a director	✓

Zooming

Function	Key and/or Mouse Shortcut	Configurable
Pans View	Middle mouse click and drag	✗
Zooms Time Axis In or Out	Mouse wheel	✗
	Z + Middle mouse click and drag	✓
Zooms Time Axis to Selected Channel, keep Spline view	**SHIFT + Z**	✓
	SHIFT + Y	✓

Copyright © 2026 Vizrt

Function	Key and/or Mouse Shortcut	Configurable
Zooms Time Axis to whole Animation, keep Spline view	**SHIFT + X**	✓
Zooms Spline/Dopesheet View In	Mouse wheel	✗
	Z/Y	✓
	CTRL + Plus sign on the numeric keypad	✓
Zooms Spline/Dopesheet View Out	Mouse wheel	✗
	X	✓
	CTRL + Minus sign on the numeric keypad	✓
Zoom Splineview in (Drag middle Mouse button to adjust)	**A** **A +** Middle mouse click and drag	✓
Zoom Splineview out (Drag middle Mouse button to adjust)	**S** **S** + Middle mouse click and drag	✓
Zooms Spline View to whole Spline	**SHIFT + A**	✓
Zooms Time Axis and Spline View to view all Key Frames of selected Channel	**SHIFT + Q**	✓
Zooms Time and Value Axis to View All Key Frames	**SHIFT + W**	✓
Zooms in onto specific region	**X +** Drag mouse rectangle	✗
	Y + Drag mouse rectangle	✗
Zoom Timeline to selected Key Frames	**SHIFT + C**	✓

Copyright © 2026 Vizrt

Searching

Function	Key and/or Mouse Shortcut	Configurable
Open/close Keyframe Search	**CTRL + F**	✓
Jump to first keyframe search result	**CTRL + H**	✓
Jump to last keyframe search result	**CTRL + E**	✓
Jump to next keyframe search result	**CTRL + N**	✓
Jump to previous keyframe search result	**CTRL + P**	✓

Keyframes

Function	Key and/or Mouse Shortcut	Configurable
Provides Rubber band Selection	**R +** Left mouse click and drag	✓
Copies Selected Key Frame	**C +** Left mouse click and drag	✓
Reverses the Order of Selected Key Frames	**CTRL + R**	✓
Jumps to next Key Frame	**TAB** (a Key frame must be highlighted first)	✓
Jumps to previous Key Frame	**SHIFT + TAB**	✓
Insert Stop Point	**P +** Left Mouse button click on Director Bar	✓
Insert Keyframe	**INS +** Left Mouse button click on Spline View	✓
Invert Order of selected Keyframes	**CTRL + R**	✓

Copyright © 2026 Vizrt

View

Function	Key and/or Mouse Shortcut	Configurable
Switch timeline to seconds	CTRL + 8	✓
Switch timeline to frames	CTRL + 9	✓
Switch timeline to fields (on interlaced formats only)	CTRL + 0	✓
Snaps to Grid on/off	CTRL + G	✓
Show spline 1	1	✓
Show spline 2	2	✓
Show spline 3	3	✓
Switch Keyframe info on/off	CTRL + I	✓
Escapes from moving a Key Frame	ESC	✗

See Also

- Application Controls and Shortcuts
- Asset View Shortcuts
- Import Shortcuts
- Integer Float Controls
- On Air Shortcuts
- Polygon Plug-in Editor Shortcuts
- Scene Editor Shortcuts
- Scene Tree Shortcuts
- Script Editor Shortcuts

Copyright © 2026 Vizrt

4.5.14 Logicmaker Shortcuts

Where	Input	Effect	Configurable
Layer row	Click	Select the layer (container and director are selected too)	✗
Layer row	F2 / context menu **Rename**	Rename the layer	✗
Layer row	Delete / context menu **Delete**	Delete the layer	✗
Layer row	Hover, click **+**	Add a state	✗
State badge	Click / Shift+click	Take the state animated / jump	✗
State in graph	Click / Shift+click	Take the state animated / jump	✗
State in graph	Double-click, F2	Rename (Enter confirms, Esc cancels)	✗
State in graph	Delete	Remove the state	✗
Line in graph	Hover	Show the durations	✗
Line in graph	Double-click / context menu **Set duration...**	Change the duration	✗
Empty graph area	Context menu **Add state**	Add a state	✗
Scene row	Click	Take the scene in the master	✗
Scene row	Double-click name, F2	Rename the scene	✗

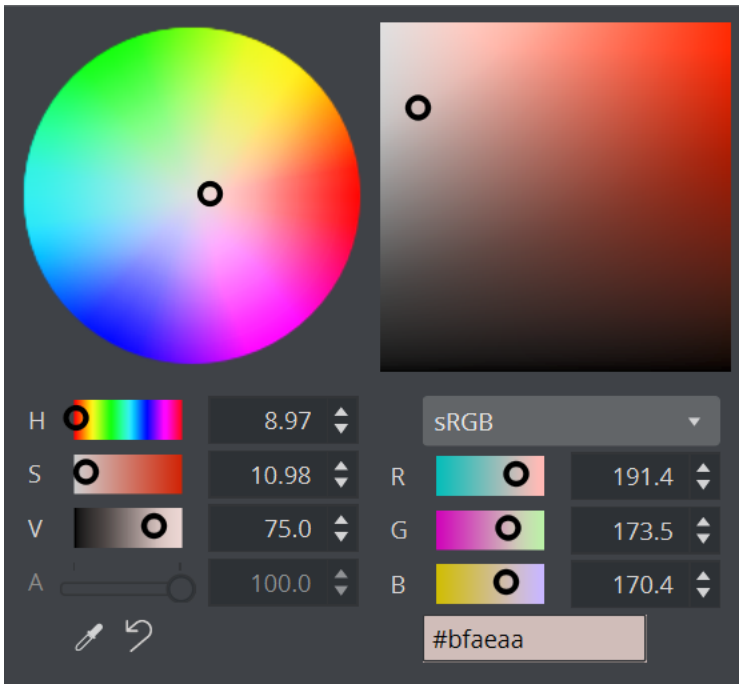
Copyright © 2026 Vizrt

Where	Input	Effect	Configurable
Scene row	Delete / context menu **Delete**	Delete the scene	✗
New Layer / New Scene form	Enter / Esc	Create / cancel	✗

Copyright © 2026 Vizrt

4.6 Color Picker

The Color picker was redesigned in Viz Engine 5.4 to match the needs for an HDR workflow. It automatically converts the selected color into the current color space.



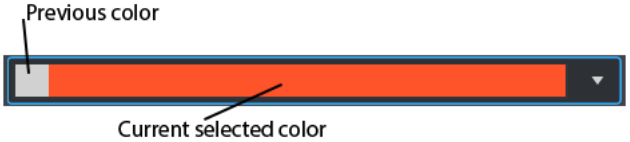
Colors can be represented in different formats:

- Hue, Saturation and Value (HSV).
- Red, Green and Blue Triple (RGB).
- Hexadecimal value (for web colors).

These color models are available in the color picker of Viz Artist. You can either drag the sliders, or enter values manually. If the color requires an additional alpha value, it is enabled.
All values are represented as floating numbers, and not as integers, as they allow for much higher precision (for color space conversions).

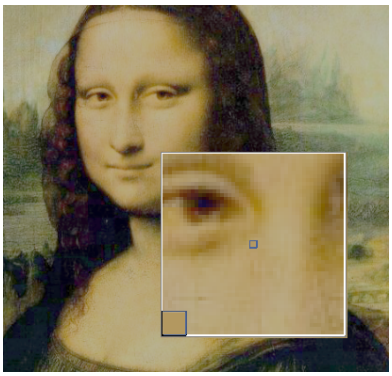
The color picker shows a color spectrum wheel and a Saturation/Value mapping.

4.6.1 General Functions



Copyright © 2026 Vizrt

When opening the color selector, the current selected color is shown in the top bar, as well as the **previous color**. Use the undo button to return to the previous color.



The color picker allows you to grab and apply a color from anywhere of the screen. Click the button to open a magnifier window to better select the color you wish.
The color is shown in the small lower left rectangle.

When picking in HDR or BT709 Linear, a color conversion is performed automatically, to match the selected color space.

Note: You can also pick from outside of the Viz Artist window.

4.6.2 Working with Different Color Spaces

The color selector is aware of the scene color space defined in the scene rendering settings. It is possible to define a color in the scene's encoding, or also in sRGB. The color encoding selector allows you to toggle the color space for editing (this does not modify the scene color space).

4.6.3 Color Picker for BT.709 Linear Mode



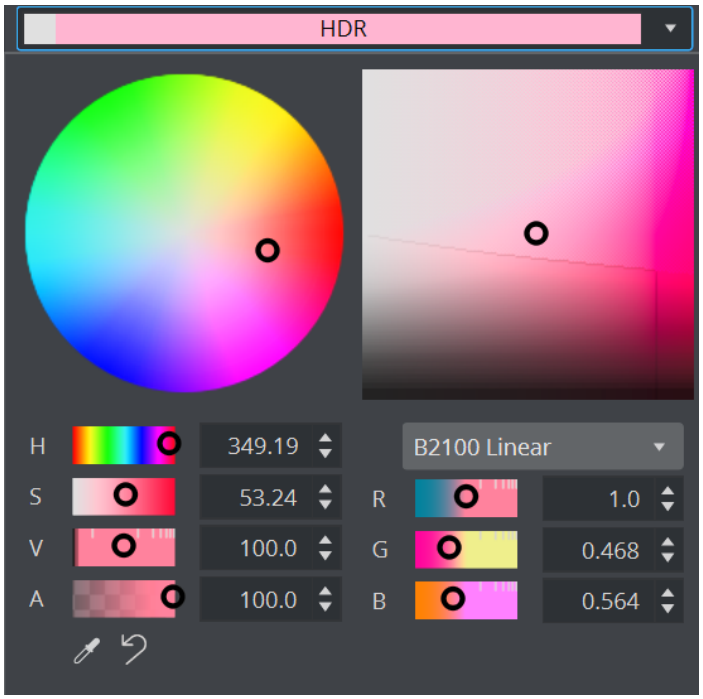
BT709 Linear has the same color range as sRGB, but has a linear mapping instead of the gamma curve. Therefore, the R/G/B channels have a noticeable intensity change in the lower range.

A further difference to the sRGB mode, is that values range from 0 to 1, instead of 0 to 255.

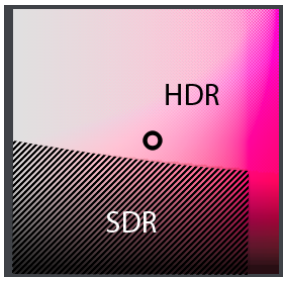
Info: Furthermore, the hexadecimal code also shows the linear representation of the color. When copy-pasting hex colors from the web, this *usually* means that a temporary switch to sRGB is required to get the desired color.

4.6.4 Color Picker for High Definition Color spaces

As HDR color spaces allow a much wider selection of colors than SDR color space, the color picker offers more functionality. If any color outside of the SDR range is selected, the selected color is labelled as **HDR** in the top bar.



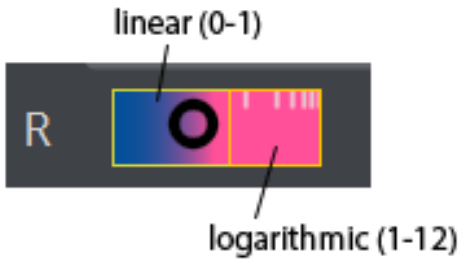
The SDR range is marked inside the spectrum map. As long as your selected color is inside the marked area, you are working in SDR space, a color outside is within the HDR color space.



Copyright © 2026 Vizrt

As color components in high definition color space allow values from 0 to 12, the SDR range is presented inside the sliders as a linear slider, whereas all higher values are shown as logarithmic.

When dragging the value, while in SDR range, your progress is linear, when you enter the HDR range, you see a logarithmic progress.



Info: Hexadecimal color inputs are not available in HDR color space.

Copyright © 2026 Vizrt

5 Introduction to Viz Engine Rendering

The following chapters describe some common workflows using the new render possibilities in Viz Artist and Viz Engine.

- Comparison of Render Pipelines
- Moving from the Classic to Viz Engine Pipeline
- From 2D Logo to AR Object
- Postprocessing Effects
- Color Space
- Working with Phong Materials
- Working with PBR Materials
- Light and Shadows
- Global Illumination
- Gaussian Splatting
- Using Multiple Textures in Viz Engine
- Working with Text and New Font System
- Viz Engine Shaders
- To Improve Render Quality
- Scene Compositing
- Virtual Studio, Augmented Reality and Extended Reality
- Instancing
- Particle System
- Working with Dynamic Geometries
- Stylesheets

Copyright © 2026 Vizrt

5.1 Comparison of Render Pipelines

5.1.1 Rendering Technology

The Classic Render Pipeline was created at a time when shaders were not available yet. It was extended over time using new capabilities based on advancements in rendering technologies, but at its heart it is still using the so called Fixed Function pipeline. In contrast, the Viz Engine Render Pipeline is based on the Programmable Render Pipeline, meaning everything is shader-based without any fixed function parts left. This conceptual difference amongst other things leads to differences between both pipelines, so that they cannot be compared easily.

5.1.2 Framebuffer Format

In the Classic Render Pipeline, the framebuffer format is defined by the configuration flag `bits_per_channel`.

Setting it to `8` gives you an RGBA8 render target, `16` gives you an RGBA16F (16-bit float) render target. The Viz Engine Render Pipeline does not support this flag, so all framebuffers for rendering are always 16 bit half-float to be able to ensure HDR compatibility from the very beginning. This format allows for unclamped calculations within Shaders, especially important for lighting with HDR environment maps or multiple lights.

5.1.3 Color Spaces

In the Classic Render Pipeline, rendering happens in Gamma Space (gamma correction is not removed from textures etc.), but the color values are just taken as is. This is fine as long as no blending between colors happen, but when it does, the result is technically wrong, but could still be fine from a designers point of view. The Viz Engine Render Pipeline uses Linear Space rendering by default, which inevitably leads to differences in rendering, especially when it comes to alpha blending.

See https://en.wikipedia.org/wiki/Alpha_compositing#Gamma_correction

https://www.kinematicsoup.com/news/2016/6/15/gamma-and-linear-space-what-they-are-how-they-differ for additional information on it.

5.1.4 Sorting

The programmable render pipeline has direct impact on the way sorting is done within the Viz Engine Render Pipeline. A change of a shader is one of the most expensive operations for the GPU. For geometries that are considered opaque, our sorting algorithm tries to reduce the amount of shader changes necessary and also tries to reduce the amount of overdraw happening when drawing them. That means, those geometries are first sorted by an internal material ID and then front-to-back. This can lead to confusion, because the drawing order is not exclusively defined by the placement of a geometry within the scene tree.

The Classic Render Pipeline does not need to take shader changes into account, since basically there are none. It is only rendering figuratively speaking with the same shader all the time.

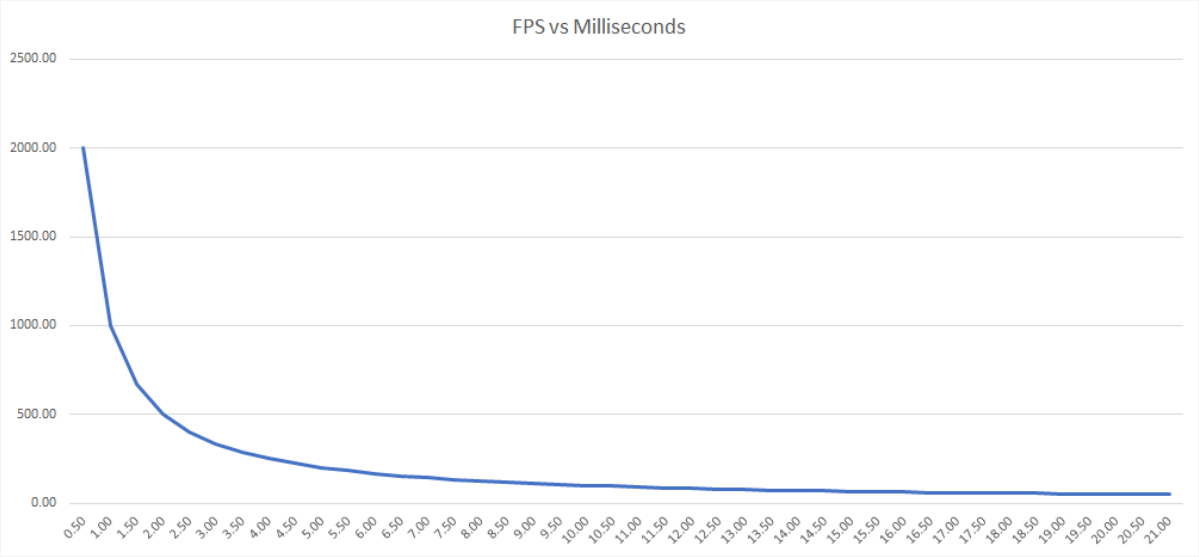
5.1.5 Performance

Comparing performance between The Classic Render Pipeline and the Viz Engine Render Pipeline is kind of difficult, since they are basically two different renderer implementations using two totally different rendering technologies. When comparing the raw numbers (MaxFPS) for an empty scene, someone might think that the Viz Engine Render

Copyright © 2026 Vizrt

Pipeline is slower than the Classic Render Pipeline. There are multiple reasons for this, based on what is being done to get a picture out of the Classic Render Pipeline or Viz Engine Render Pipeline, but this should not be used as a decision-maker on what rendering pipeline to use.

The following graph shows, why someone should not use MaxFPS to compare performance at all. MaxFPS numbers are changing exponentially compared to the time it takes to render a field in milliseconds, which changes linearly.



5.1.6 Compositing of Media Assets

The way the compositor works is slightly different between Classic and the Viz Engine Render Pipelines. Images (with Alpha channel) can only be blended if there is a background available.

Example

To use an Image channel or Super channel with alpha, the ordering has to be changed:

Classic Render Pipeline	Viz Engine Render Pipeline
	

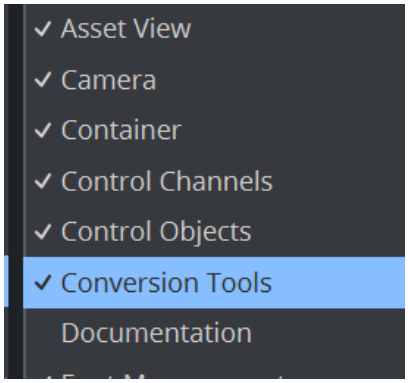
Copyright © 2026 Vizrt

5.2 Moving from the Classic to Viz Engine Pipeline

This document describes the various differences between the Classic Render Pipeline in Viz Engine 3.x and the Viz Engine Render Pipeline introduced in Viz Engine 4. As the Viz Engine Renderer is a new implementation, there are known incompatibilities when switching between Classic and Viz Engine Render blades.

Features and missing functionality will be added in future versions. For a list of known limitations, please refer to Viz Engine Release Notes.

5.2.1 Conversion Tools



Moving existing scenes from the Classic Render Pipeline to the Viz Engine Render Pipeline requires some changes in how to design your scenes or how to use certain plug-ins. Some functions provided by plug-ins are now integrated either directly into the new Viz Engine Render Pipeline or are now obsolete.

Viz Engine version 5.3 and newer provide basic tools to automate these tasks, however, these tools can only assist in basic tasks and do not perform a complete conversion, some manual adjustment is still needed. The tools are located in the **Conversion Tools** dock. The following tools are supported for now, the functionality will be extended in the next versions of Viz Engine.

Info: Conversion Tools are only helpers. Some manual steps are still required.

Open a Classic Scene, change the Render Engine to the Viz Engine Render Pipeline and choose which type of helper you want to use:

Materials

Creates a new Phong material based on the same settings as the classic material. Existing textures are set into the Phong color part.

Lights

Creates compatible lights based on the existing classic lights. All classic lights are reconstructed as new Engine Renderer lights inside the scene tree. Unused lights are not needed, can be deleted afterwards.

Copyright © 2026 Vizrt

Texts

Converts existing Classic text objects to the new font render system in Viz Engine Renderer. The tool convert the original font data into the new format and replace the Classic Text plug-in to Viz Engine Renderer Font.

Classic Text objects can also be converted on a container base level by usign the context menu and choose the *“Convert Classic Text…”* entry.

Information: The original base font needs to be available on the Graphic Hub to allow a correct conversion.

Particles

Helps in transitioning old ParticleFX objects into the new Particle System.

5.2.2 Feature Replacement Table

The following table lists differences between the Classic Render Pipeline and how to achieve the same on the Viz Engine Render Pipeline.

Classic Render Pipeline Feature	Viz Engine Render Pipeline Feature	Comment
Shaders in General	-	Classic Shaders do not work anymore in the Viz Engine Render Pipeline. This affects all fragment as well as vertex Shaders (for example: RTT, PixelFX etc.).
Frame Mask, Soft Mask, Image mask	workaround with Textureslot	To achieve the same effect, use the Textureslot in Screenspace (textGen) mode.
Global Scene lights	obsolete	Scene lights have been replaced with container-based Viz Engine lights.
HDR Lighting	obsolete	HDR environment rendering is an essential part of the Viz Engine Render Pipeline. Viz Engine scenes require a global environment for rendering.
Front- and Back-Layer	supported for compatibility	Front- and Back-Layer in the Viz Engine Render Pipeline use internal GFX channels for compatibility reasons.
Self Layered Scenes	removed	Self layered scenes (Same scene in Front- and/or Backlayer) are not supported in the Viz Engine Render Pipeline.

Copyright © 2026 Vizrt

Classic Render Pipeline Feature	Viz Engine Render Pipeline Feature	Comment
Stencil Shadows	removed	Deprecated - Only Shadowmaps are supported.
Projector and Projector Target	removed	
Bump Map Shader	obsolete	Integrated into PBR Materials as normal/height map.
Per Object Lighting	obsolete	Per object lighting has been replaced with container-based Viz Engine lights.
Multitexturing	removed	Replaced with Texture Slot plug-in.
Image Blending	removed	
Dynamic Scene	obsolete	Dynamic Scenes are not supported anymore, use GFX Channels or Scene Views instead.
Dynamic Texture	obsolete	Dynamic textures are not supported anymore. Try to replace them with Viz Engine Shaders or Texture Renderers if needed.
Interactivity using Viz Scripting Language	partially implemented	Simple mouse object picking is supported already to allow for some base interactivity with Viz Engine Renderer.

5.2.3 Tonemapper, Gamma Correction and Color Spaces

The Viz Engine Render Pipeline is designed to be in linear space whereas the Classic Render Pipeline is sRGB. If pixels are alpha blended, there can be different colors that get blended due to linearization. The Viz Engine Renderer can automatically apply Tonemapping Effects, Color limits etc.

For Media Assets that should not be touched by these effects please use the Masking plug-in.

Copyright © 2026 Vizrt

5.3 From 2D Logo to AR Object

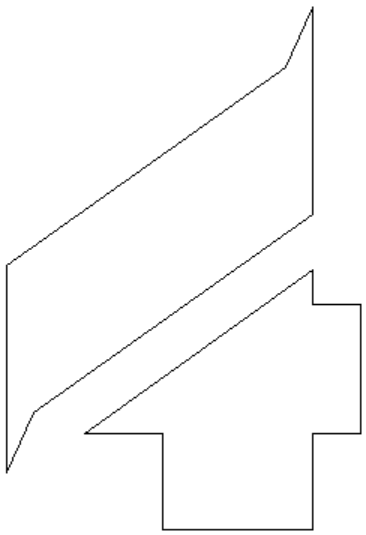
The following workflow describes how to move from a simple 2D logo to a full AR object in Viz Artist:

- Step 1: Preparing Data for Use in a 3D Environment
- Step 2: Bringing the Object into the 3D World
- Step 3: Texturing Using Substance Painter
- Step 4: Creating the AR Object

This chapter covers the following techniques and describes the setup using these products:

Product	Techniques	Links
Adobe Illustrator	Converting to SVG for import into Blender	https://www.adobe.com/products/illustrator.html
Blender	Extruding, triangulating, creating UV Maps and exporting	https://www.blender.org/
Adobe Substance Painter	Texturing	https://www.adobe.com/products/substance3d.html
Viz Engine	Rendering	

5.3.1 Step 1: Preparing Data for Use in a 3D Environment



Copyright © 2026 Vizrt

Lets start with a Logo design as Illustrator File. This file format describes objects as a collection of paths and points. If these objects are going to be rendered in 3D space, all paths need to be closed. Adobe Illustrator is the de facto standard when dealing with vector documents and it can read and export the most common ones. In our use case, as Blender is not able to read the Adobe Fileformat (*.ai) natively, we need to convert it to a more common format. In this case, SVG (Scalable Vector Graphics).

To do so, open your given *.ai* document, select **Save as** and choose *SVG* as the target file format.

5.3.2 Step 2: Bringing the Object into the 3D World

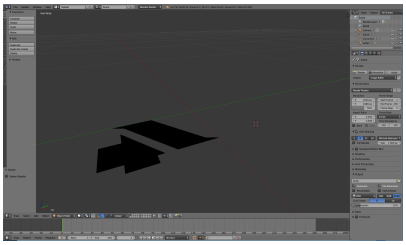
In the next step, we use the open source software Blender to import our SVG Path object, extrude it, convert the paths to 3D vertices and prepare for the next step: Texturing in Substance Painter.

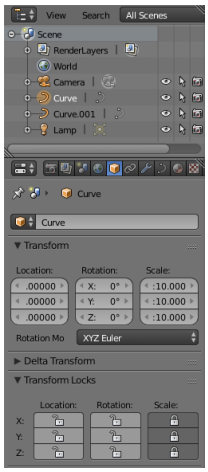
> **Info:** The following steps are related to Blender, they work of course accordingly in any other 3D modelling Software like Cinema 4D, Maya or Autodesk 3D Max.

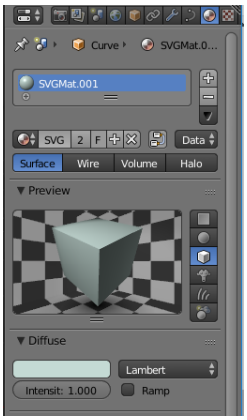
First, we need to import the previously exported logo design into a new Blender scene. We need to create a new empty document, this is done by selecting **File/New...** Delete the default cube by hitting the **Delete** key and confirm.

Next, import our 2D Shape. Select **File > Import > Scalable Vector Graphics (svg)**. Browse to the location where you stored your *.svg* file and import it.

Use the mouse wheel to zoom in until your logo is visible. Scale your logo to the final dimensions. Select each path of your logo by using the right mouse button, and set a proper scaling. Changing the object's color is also a good idea, as it would give you a better impression about the extrusion.

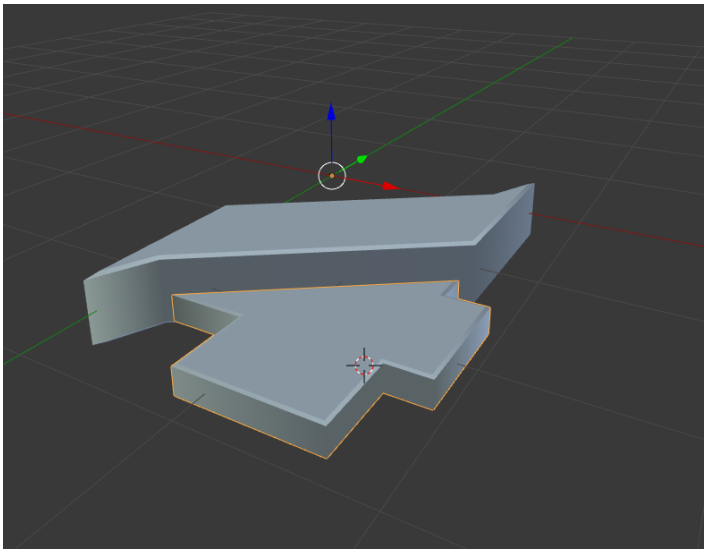


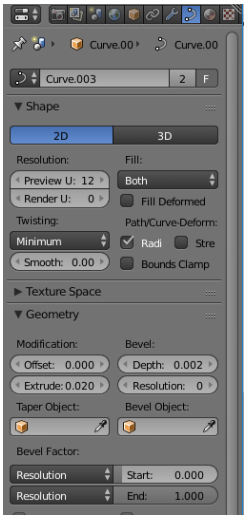




Now we extrude our object and make it a real 3D object. This is done by extruding the paths and transforming them afterwards into triangles. You can of course also convert it to triangles first and the extrude afterwards. This is done a bit differently (by using a path). Select the data tab first, select each face of our logo (right mouse button) and set the desired extrusion in the Geometry section:

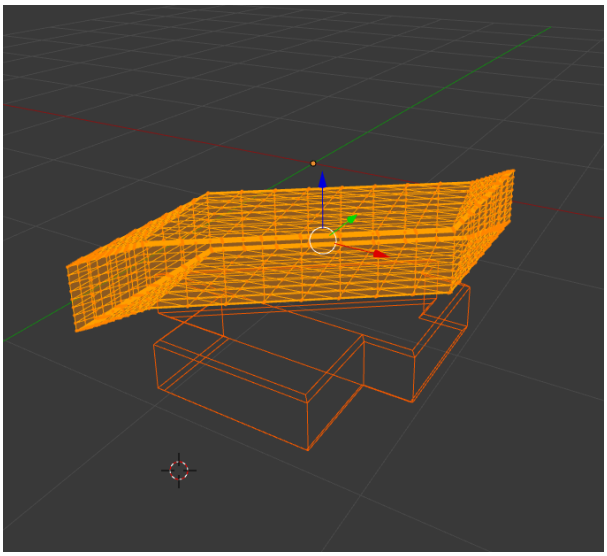
Copyright © 2026 Vizrt





Our object still consists of simple paths (nurbs) and not triangles. Viz Engine requires a mesh of triangles, therefore we now have to transform the object(s) into a triangle mesh. Select again each component and press **ALT + C**, choose **Mesh from Curve/Meta/Surf/Text**. To verify, if you switch to wireframe mode (press **Z**), your object should now show a few triangles.

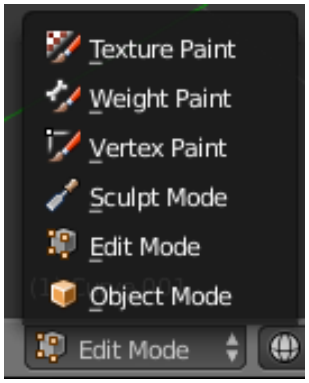
To increase the level of detail, we can now subdivide our geometry. Switch to Edit mode, select your faces (e.g. by using Border select: pressing **CTRL + left mouse down/mode**). From the tools menu, select **Transform/Add/ Subdivide** and enter your number of cuts:



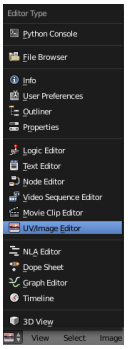


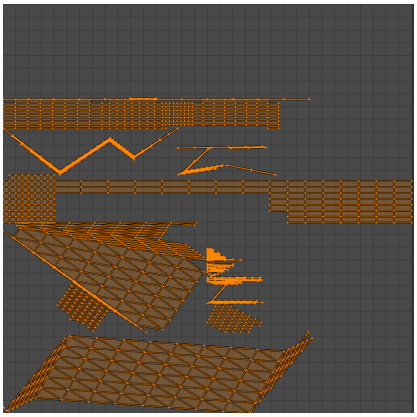
Switch back to object mode, select all your single objects and select **Tools/Join**. We want to have one big UV map for the textures. Although Substance Painter can deal with different set of geometry and multiple UV maps, geometry and UV map need be provided together. One single geometry and multiple UV maps won't work as stitching all UV maps together is currently not possible in Substance Painter. Switch back to Edit mode:

Copyright © 2026 Vizrt



Select all faces (**CTRL + Mouse Bounding Box**), and select **Shading/UVs and Unwrap** from the UV menu. To verify the correct UV map, select **UV/Image Editor**. It should show the unfold object.





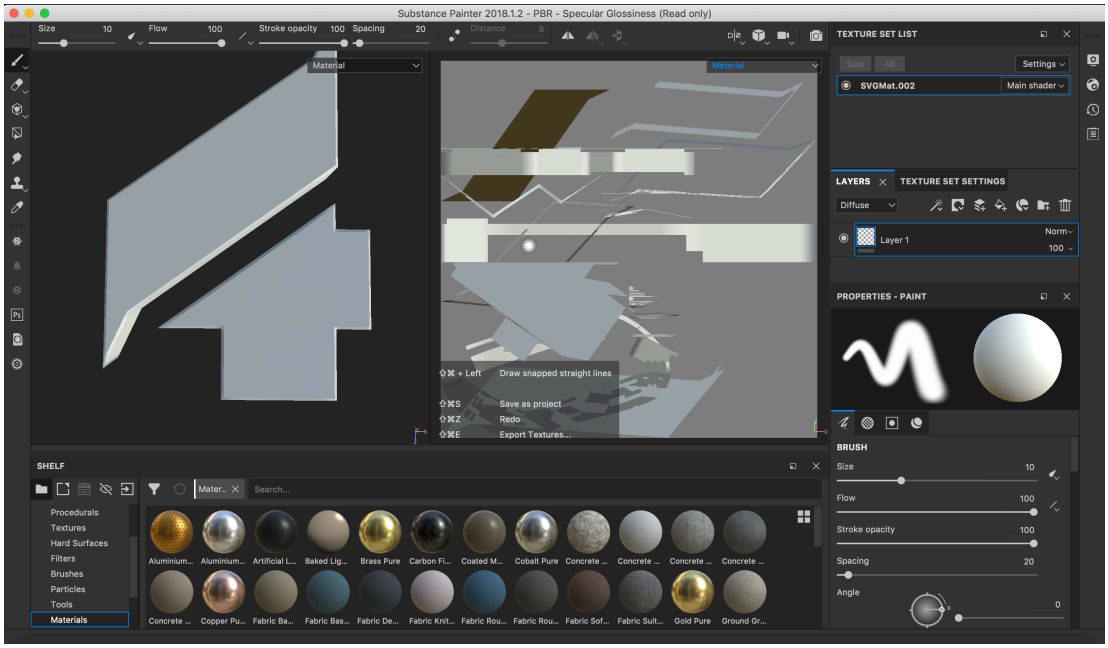


Now we are ready to export our object into a *.fbx* file for texturing it in Substance Painter. Select **File/Export** and choose *FBX* as file format.

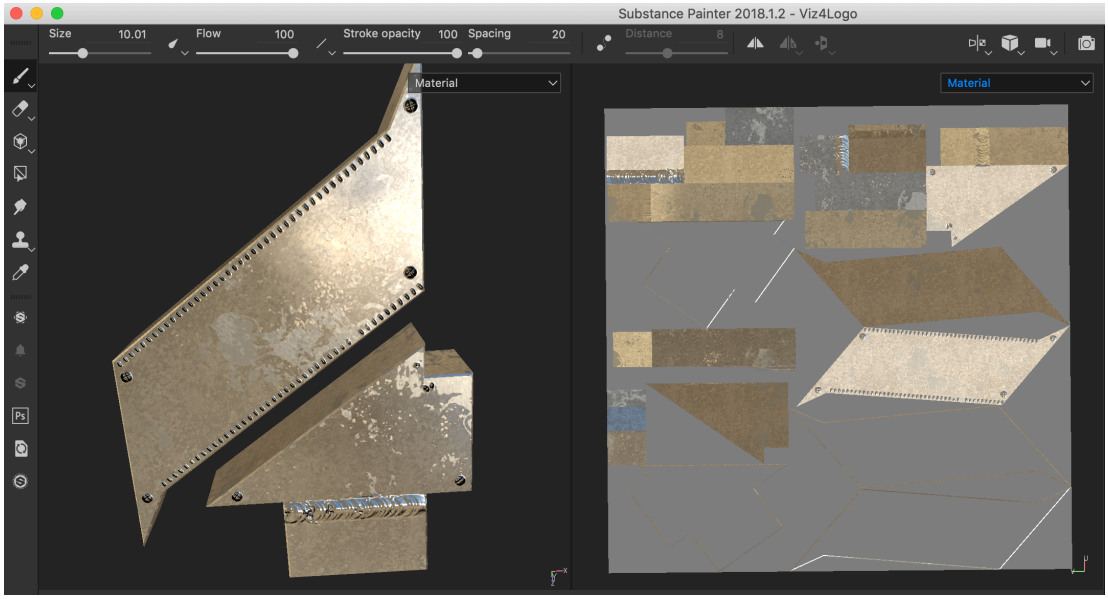
5.3.3 Step 3: Texturing Using Substance Painter

In Substance Painter, create a new project. Select the previously created *.fbx* file as Mesh file and choose **OpenGL** as Normal Map format:

Copyright © 2026 Vizrt

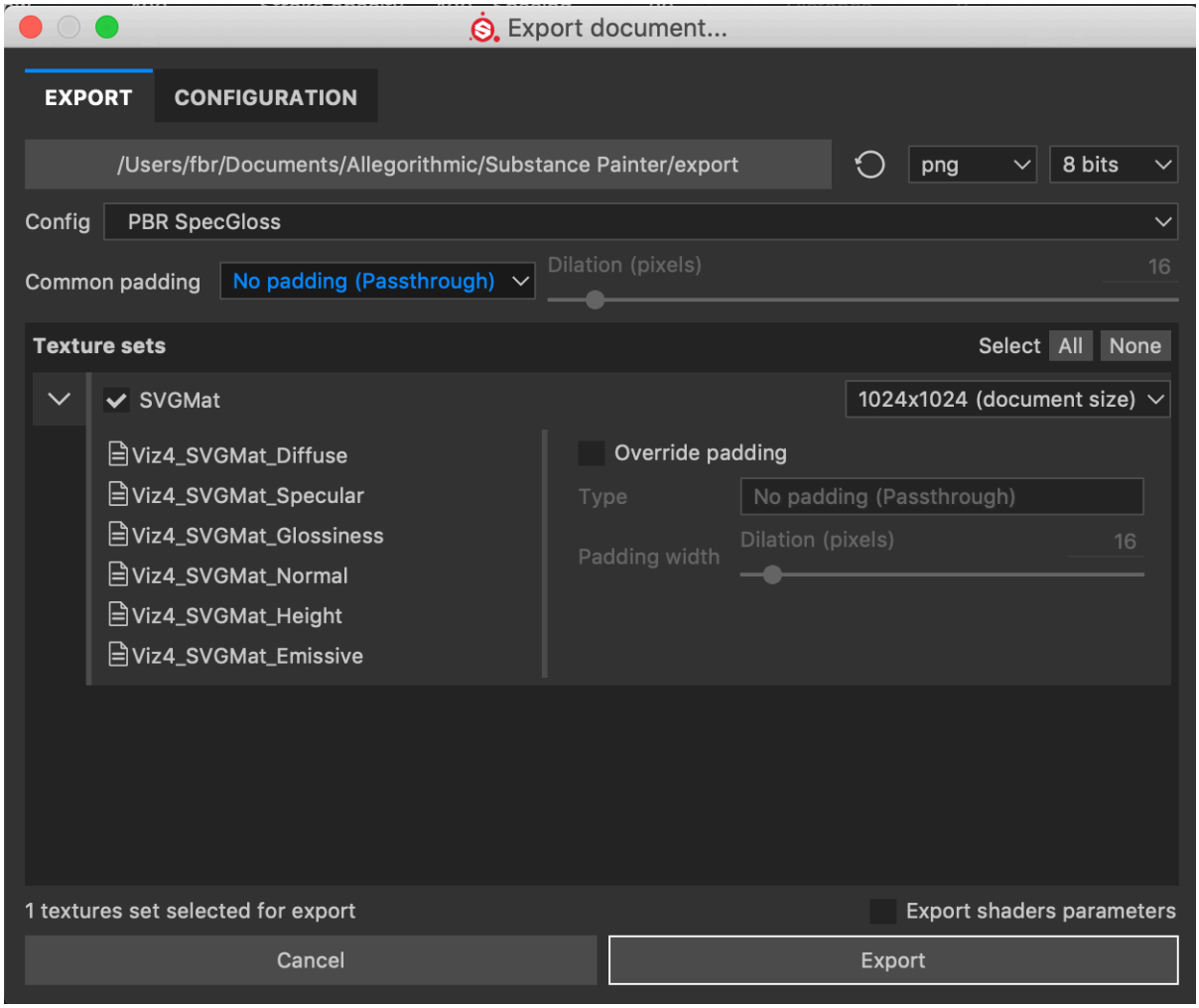


Now start painting your geometry by adding various materials, masks and effects.



Export your textures by selecting **File/Export Textures**. Be sure to select 8-bit color depth, not 16-bit. This generates all needed textures to build our physical-based rendering (PBR) material in Viz Artist.

Copyright © 2026 Vizrt



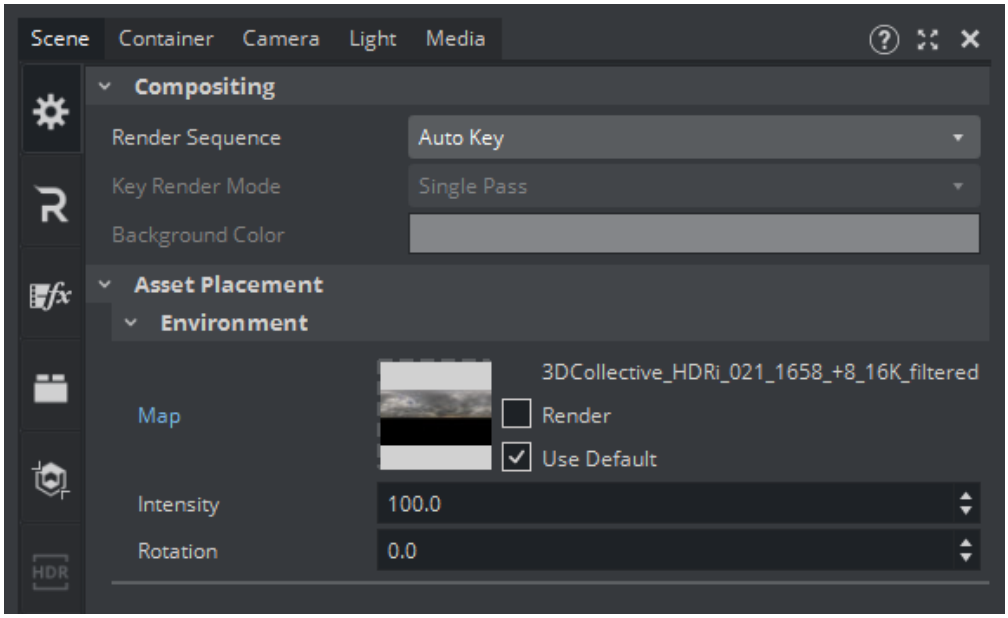
5.3.4 Step 4: Creating the AR Object

Now it is time to bring everything together in Viz Artist. Start Viz Artist and import the *.fbx* file (**Import > Scenes**) as new scene. *.fbx* files come with a light and a camera tracking object, they are currently not needed. Deactivate the Camera Tracking by resetting any Position and Direction Tracking, set all Positions and Rotations to `0`.

Select the object, switch to the transformation editor, reset position and rotation to `0` and bring the object into focus by clicking on **Center to Camera and Face Camera**. The object should now be visible, though it may be necessary to move the camera away a bit. Now switch to Viz 4 render pipeline in the scene settings and apply an environment map.

For realistic AR effects, you may need to build your own environment map that reflects your studio lightning and coloring. A good tutorial on how to create your own HDR environment maps, can be found here: https://sketchfab.com/blogs/community/how-to-create-your-own-hdr-environment-maps/.

Copyright © 2026 Vizrt



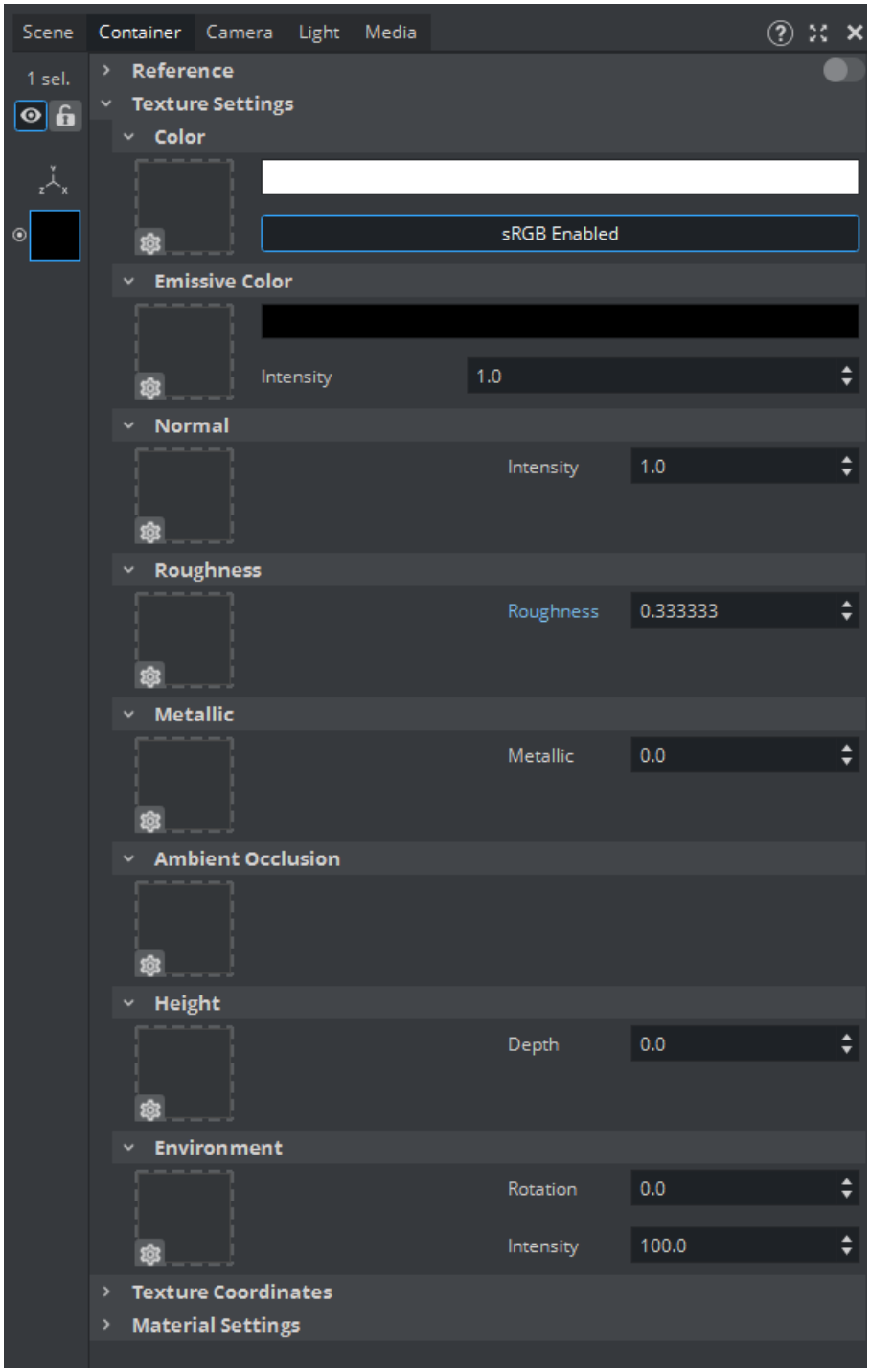
On your objects container, add a Viz Engine PBR Material:



This adds a new PBR material onto your geometry. For more information on how to work with PBR materials, see Working with PBR Materials.

Now, we can import the generated textures and assign the ones we need:

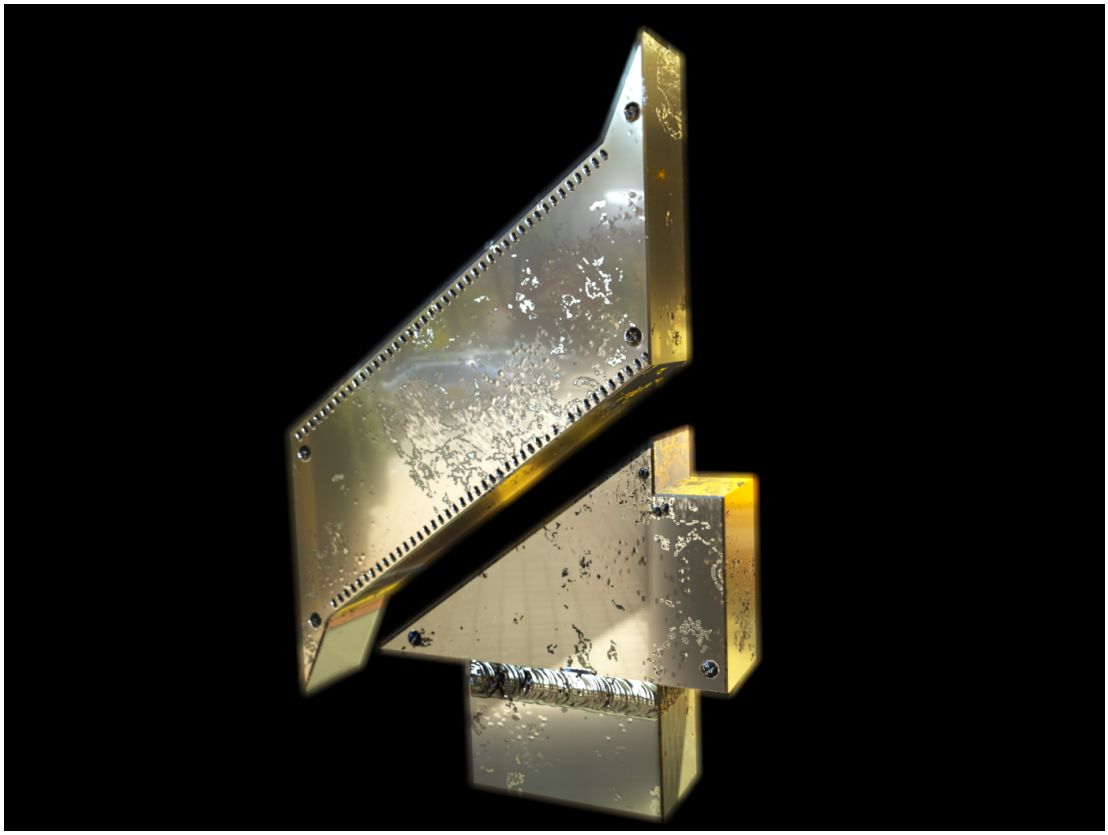
Copyright © 2026 Vizrt

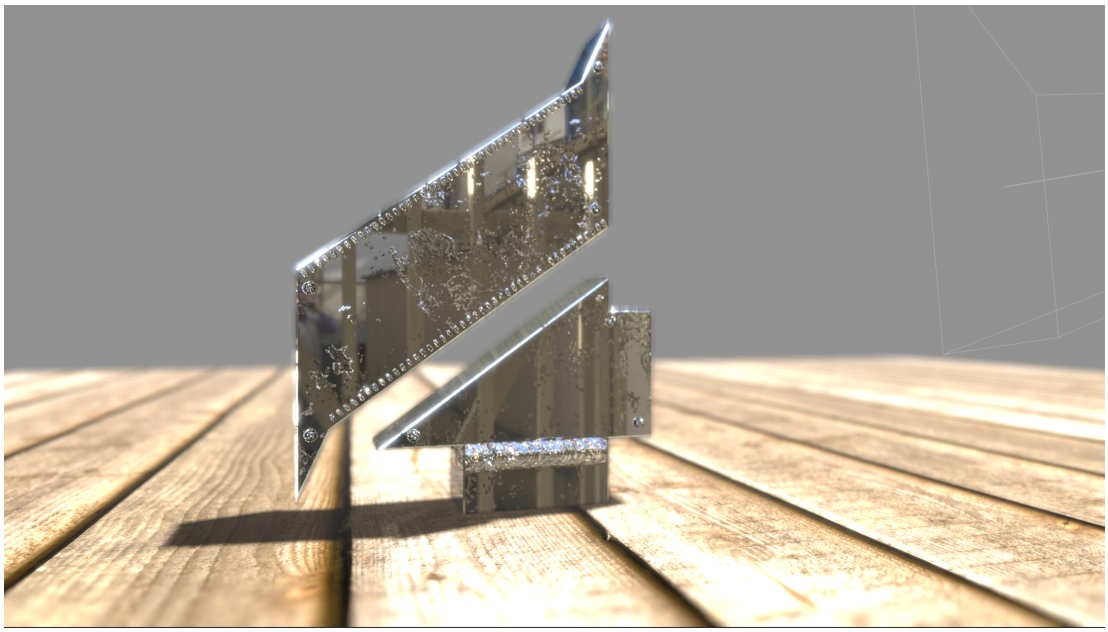


Adjust the parameters for Roughness and Metallic to match your needs. As this object is going to be used as an AR element, a key plug-in is required. As a TV Studio typically has variable lighting conditions, this can also be simulated by adding additional Light like spot lights or direction lights. They should also match your real lighting conditions.

Copyright © 2026 Vizrt

Adding a Caster on the Light sources and Caster/Receiver plug-in to your objects adds photo realistic shadowing. Cinematic Depth of Field also enhances your object. Any fine-tuning for Tonemapping and Depth of Field or should be done in real lighting conditions with a tracked camera connected.





Copyright © 2026 Vizrt

Now as your object is ready, you should think about how you want to bring your object on stage, maybe by opening a virtual floor and lift it up?

Copyright © 2026 Vizrt

5.4 Postprocessing Effects

This section contains information on the following topics:

- Cinematic Depth of Field
- Screen Space Reflection
- Bloom
- Deep Learning Super Sampling
- SSAO
- Fog
- Chromatic Aberration
- Light Glares

5.4.1 Postprocessing Effects

FXAA and Temporal Anti-aliasing

Hardware based anti-aliasing might not always be sufficient enough. Those two methods allows to render a much better image. Please see the chapter about Render Quality for more information.

Copyright © 2026 Vizrt

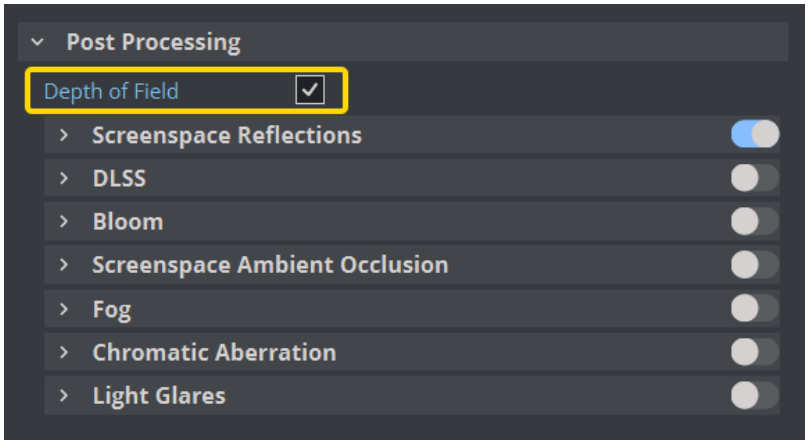
5.4.2 Cinematic Depth of Field

Cinematic depth of field technique aims to provide a look similar to real photography where objects which are out of focus get blurred.



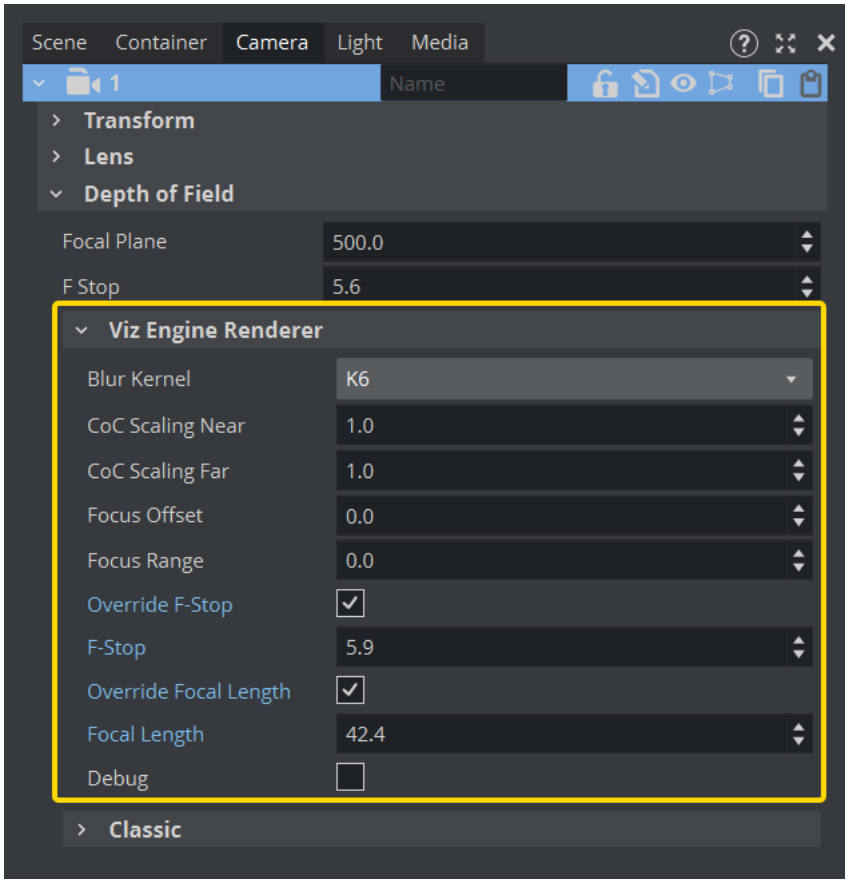
Usage

As Depth of Field is a Camera property, you'll find the relevant settings in the Camera Section, however it needs to be turned on within the Post Processing Settings in **Scene** > **Rendering:**

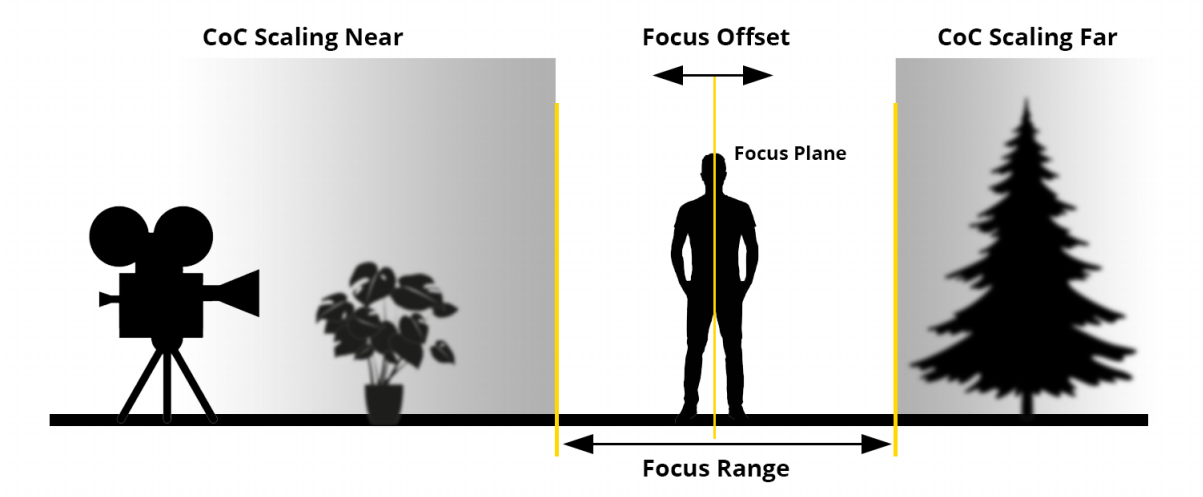


Afterwards head over to the camera section. You'll find all required settings in the **Depth of Field** section:

Copyright © 2026 Vizrt



DOF has common settings for both Classic and Viz Engine Render Pipelines as well as separate ones that only apply to the selected render pipeline. The following explains Viz Engine Render Pipeline settings only.



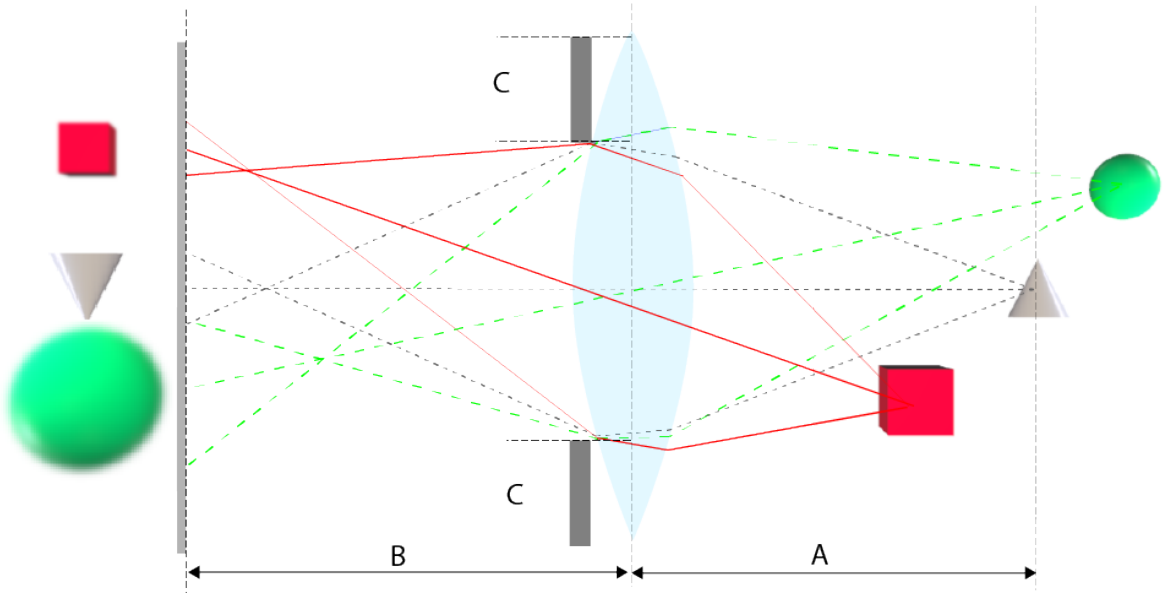
The **Focal Plane** defines the distance to the camera that should appear as "in focus". In addition to this value, a **Focus Offset** can be added or substracted. All objects within this range (=Focu Range) will appear sharp and clear.

Copyright © 2026 Vizrt

Near objects are influenced by the **CoC Scaling Near** and far objects by the **Scaling Far** factor. The **Blur Kernel** defines how precise the is calculated.

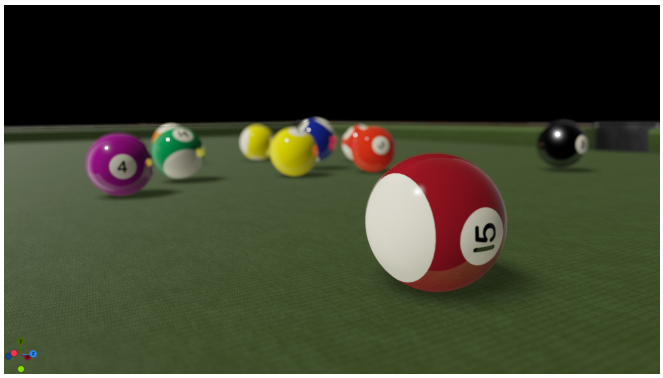
Lens Properties: Focal Length and F-Stop

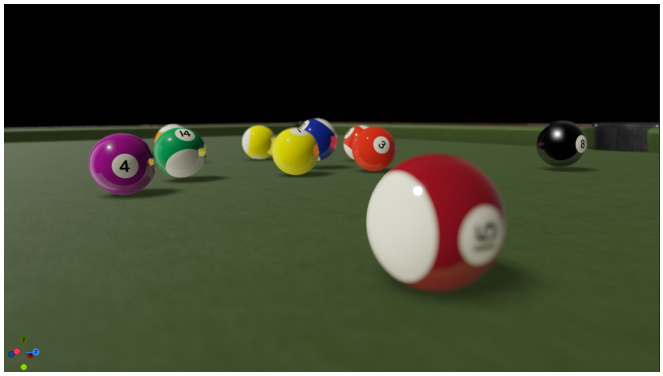
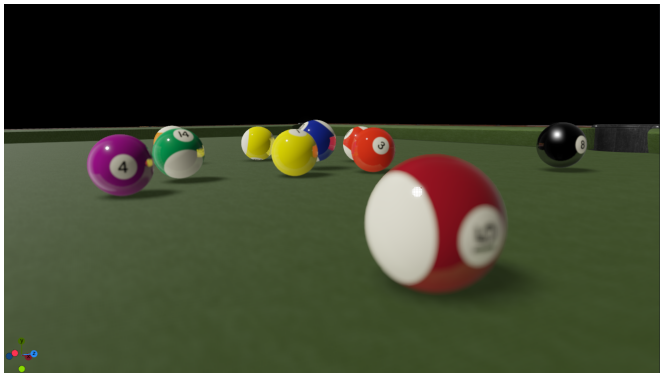
Focal Length and **F-Stop** are usually taken from external cameras signals. The **F-stop** number is determined by the focal length of the lens divided by the diameter of the **aperture**. Focal length refers to a lens' field of view (sometimes called angle of view), which is the width and height of the area that a particular lens can capture. They can be overridden if needed.



Focus Range (A)
Focal length (B)
F-Stop (C)

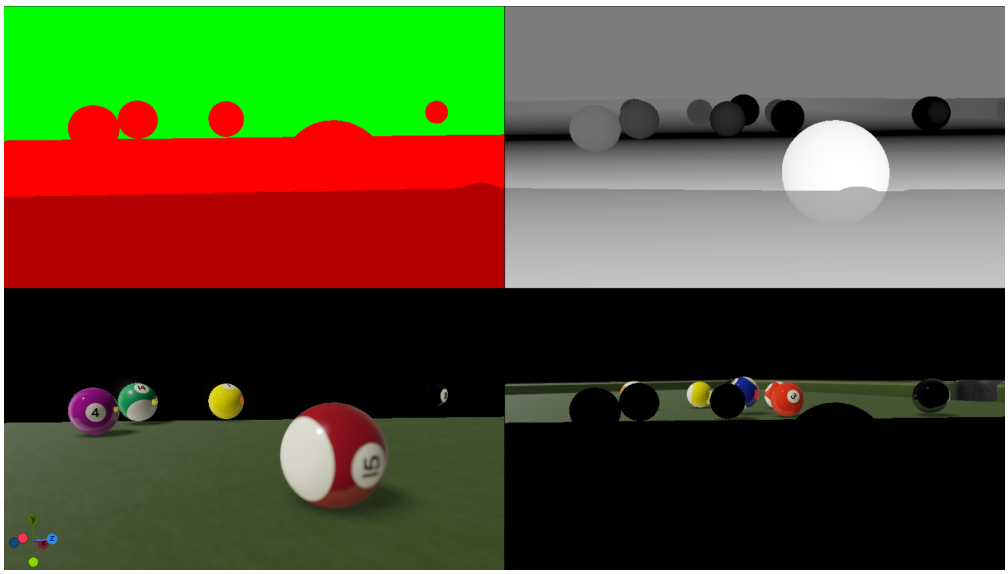
Samples

Focal plane close to camera on Sphere Nr. 15 F Stop: 1.0	

Focal plane on Sphere Nr. 15, Focus Range 100 F Stop: 1.0	
Focal plane on Sphere 3, Focus Range high Small F Stop 0.9	

Debug View

To fine tune your settings, you can activate the Debug View:



Copyright © 2026 Vizrt

Top Left Corner

- The green area shows far field.
- The darker red area shows near field.
- Normal red shows where everything is in focus and nothing blurred.

Top Right Corner

- Shows the blur strength. The brighter the color the more it gets blurred.

Bottom Left Corner

- Shows objects included in the near-field.

Bottom Right Corner

- Shows objects included in the far-field.

Copyright © 2026 Vizrt

5.4.3 Screen Space Reflection



Screen space reflection (SSR) is a technique where image data of the rendered scene is reused to calculate reflections. It is widely used to create subtle reflections for floors or wet surfaces. As it only operates in screen space, it also has the limitation that it can only reflect geometries which are visible. Therefore, it is not possible to create reflections for geometries outside the screen or back-facing surfaces. Further, only geometries which write to the depth buffer are visible in reflections. SSR is computationally heavy, but can lead to good results when the scene design keeps account of its limitations.

Usage

Reflections can be enabled for any container with a PBR or Phong material. Just set the Reflective option to `On`. SSR can be configured directly in the **Scene > Rendering** settings. Enabling post-processing is required to see the effect in the Scene Editor.

Copyright © 2026 Vizrt



SSR Parameters

Effect

- **Intensity:** Controls the reflections strength.
- **Vignette:** Reflections near screen border are faded out.
- **Fade Distance:** Fades out far reflections.
- **Roughness Scaling:** Scales global roughness (used for reflections).
- **BRDF Bias:** Controls how glossy reflections are scattered (higher values give mirror like reflections).
- **Key Intensity:** Controls the resulting key of the reflections.

Copyright © 2026 Vizrt

Ray Marching

- **Minimum Rays:** Determines the number of rays used for smooth surfaces.
- **Maximum Rays:** Determines the number of rays used for rough surfaces.
- **Thickness:** Adds additional thickness for thin geometries which could be missed without it.
- **Iterations:** Determines the number of iterations to find an intersection.
- **Roughness Threshold:** Do not trace rays for surfaces which are very rough.

Temporal

- **Frames:** Determines the number of frames used to accumulate reflections (higher values can cause ghosting for moving objects/camera).
- **Re-projection Strength:** Controls how strong the rays are re projected for moving objects.

Denoising

- **Enable:** Enables reflection denoiser.
- **Strength:** Sets denoise strength.

Debugging

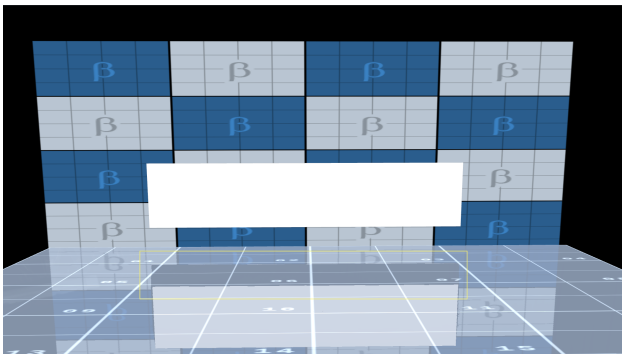
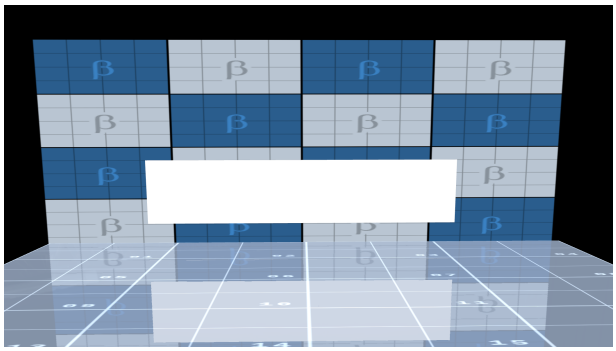
- **Debugging Output**
    - **None:** Disables debug output.
    - **Reflections:** Shows reflections only.
    - **Mask:** Shows which objects are reflective.
    - **Miss:** Shows where reflections with no hit.
    - **Layer:** Shows objects with used for multilayer tracing.
    - **Normals:** Shows scene normals used for tracing.
    - **Depth:** Shows scene depth used for tracing.

Multi-Layer SSR

The main limitation of SSR is that it can only reflect objects which are visible. Objects in the foreground can prevent reflecting objects which are visually blocked.

The **Layer-Plugin** can be used to render specific objects in a way that they do not block reflections behind them. Carefully decide which objects need to be drawn in foreground for SSR as it can impact performance.

Copyright © 2026 Vizrt

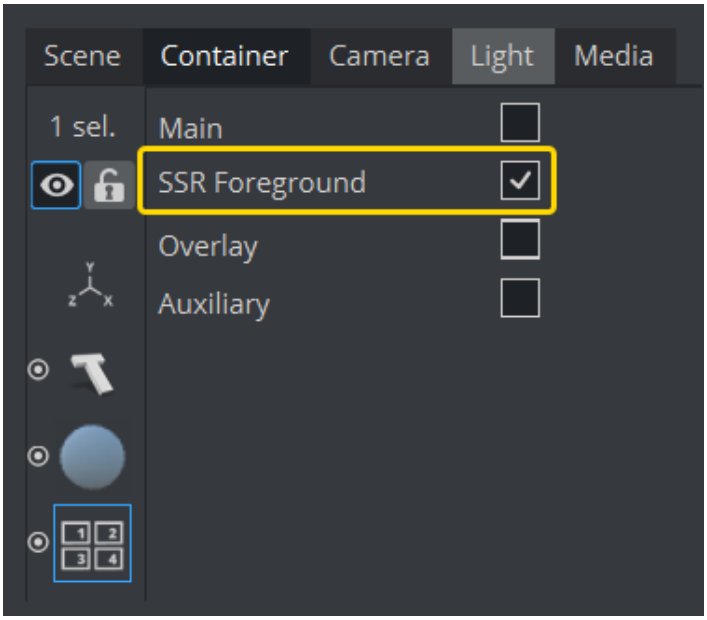
Multi-Layer OFF	Multi-Layer On
	

In the example above, the wall can not be completely reflected on the floor as the Text Geometry is hiding parts of it. To get rid of this limitation, add the Layers plug-in to the text geometry and set it to be **SSR Foreground**. This changes the rendering sequence for the SSR effect.

Usage:

- Add **Layer-Plugin** to the container which should be drawn in foreground for SSR.
- Select **SSR Foreground** as layer.





Performance/Visual Tweaking

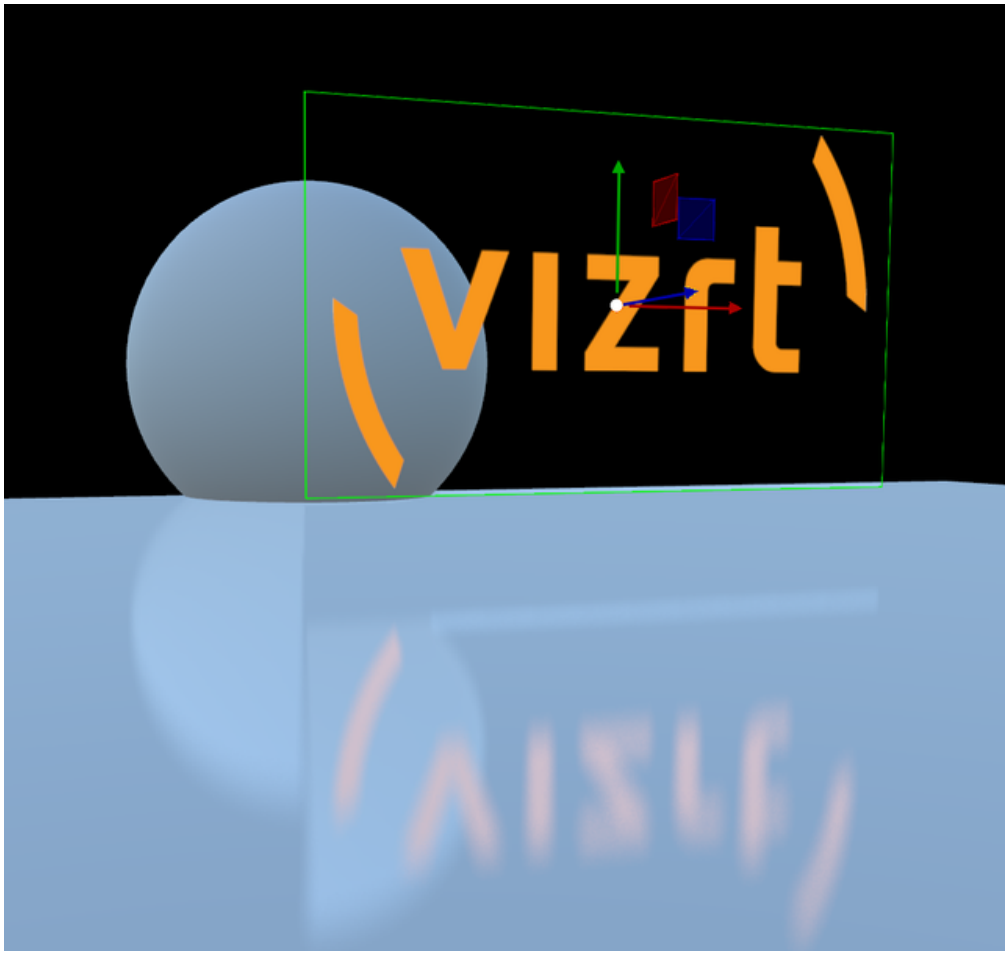
SSR can be quite performance intensive. Make sure to only make objects reflective where reflections are wanted. Reducing **Ray Marching > Iterations** and **Ray Marching** > **Min/Max Rays** can also boost performance.

Copyright © 2026 Vizrt

- Make sure that the Camera Near/Far is set to cover the minimal range. This results in higher precision in the depth buffer which is important for **Ray Marching.**
- Ghosting for moving object and camera flights can be reduced by using lower values for **Temporal > Frames**.
- If the reflections are not strong enough try to increase **Effect > Intensity** or reduce the **Effect > Roughness Scale.**
- If black lines artifacts can be caused by to low **Ray Marching** > **Thickness**.

Troubleshooting

SSR requires a depth buffer to work correctly. This however, has some side effects when transparent images are in use, they produce unwanted artifacts in all transparent areas.

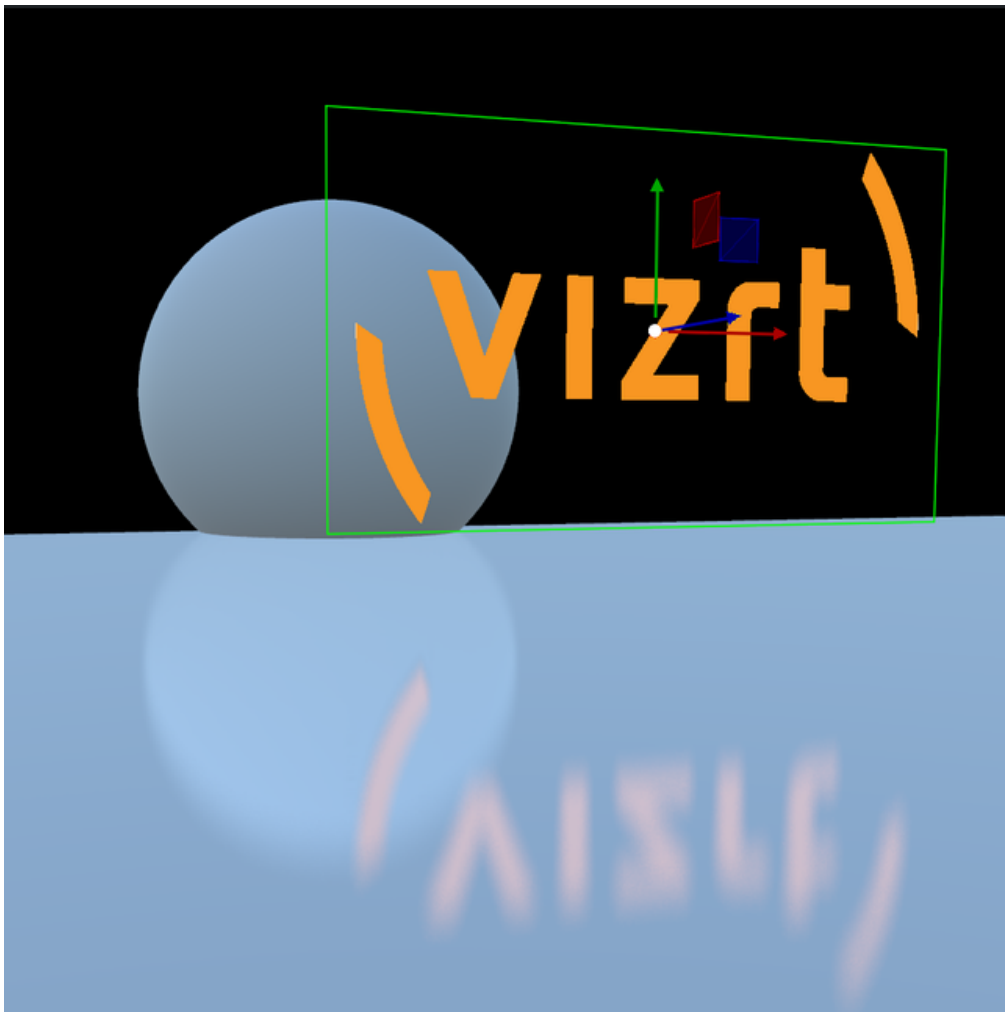


To fix this, we need to use the layers plug-in as described above and get rid of the alpha information to disable the depth write. This can be done by using a custom fragment shader.

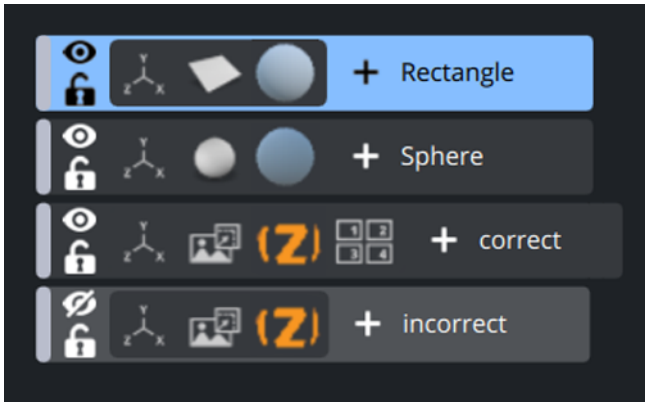
```
@registerParametersBegin
@registerParameterSampler2D(iChannel0, "Texture0")
```

```
@registerParameterFloat(alphaThreshold, "Alpha-Discard", 0.1f, 0.f, 1.f)
@registerParametersEnd

vec4 toLinear(vec4 sRGB)
{
bvec4 cutoff = lessThan(sRGB, vec4(0.04045));
vec4 higher = pow((sRGB + vec4(0.055))/vec4(1.055), vec4(2.4));
vec4 lower = sRGB/vec4(12.92);
return mix(higher, lower, cutoff);
}

void fragmentMain()
{
vec4 fragColor = texture(iChannel0, viz_getTexCoord());
if(fragColor.a < userParametersFS.alphaThreshold)
discard;
viz_setFragment(toLinear(fragColor));
}
```

Using this shader as the material gives the following result:



Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

5.4.4 Bloom

Bloom is a post processing effect which mimics the behavior of real-world cameras, where bright objects (lights, glowing objects) leak into their surroundings. The Bloom shader is directly integrated into Viz Engine Render Pipeline and makes use of the HDR rendering capabilities. It uses multi-pass rendering with downscale and upscale steps to generate the blur that is used for compositing.

The settings for the Bloom Effect are located in the **Scene** > **Rendering** Menu:



It is used to replace the deprecated Glow plug-in.

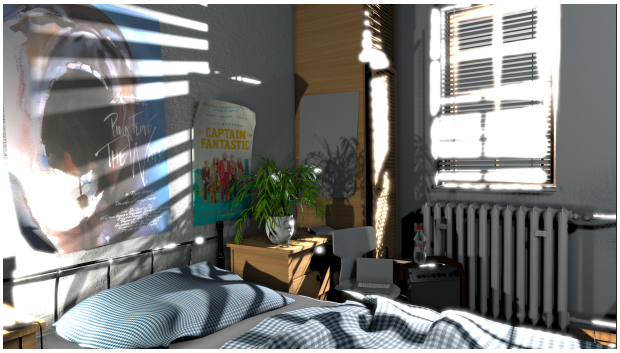
Parameters

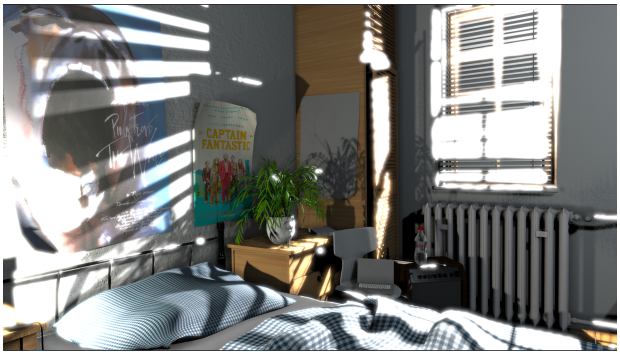
Parameter	Description
`threshold`	Defines the basic brightness threshold. A pixel that is below this threshold is not used in the blur steps.
`softThreshold`	Additional threshold to soften the pixel selection instead of a hard-cut.
`intensity`	Scales the bloom image colors before adding it to the original image.
`iterations`	How many render passes are used to calculate the boom. Higher value means higher quality.
`Key Intensity`	Specifies how much of the effect is used within the key signal.

Copyright © 2026 Vizrt

Examples

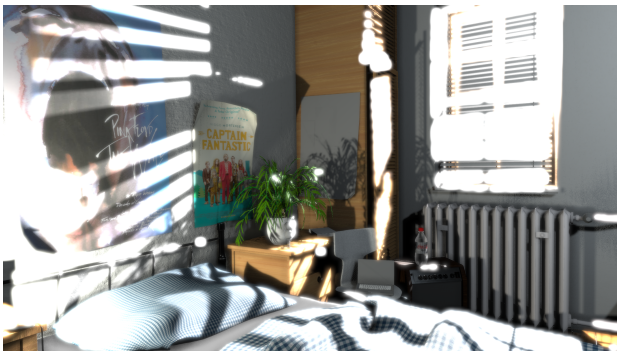
Screenshot	Bloom Settings
	Bloom Disabled

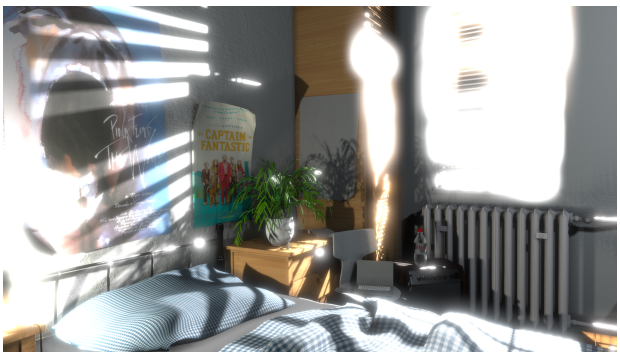
	Bloom Enabled
	`threshold = 1`
	`softThreshold = 0`
	`intensity = 1`

	Bloom Enabled
	`threshold = 0.5`
	`softThreshold = 0`
	`intensity = 1`

Copyright © 2026 Vizrt

Screenshot	Bloom Settings

	Bloom Enabled
	`threshold = 2`
	`softThreshold = 1`
	`intensity = 10`

	Bloom Enabled
	`threshold = 2`
	`softThreshold = 1`
	`intensity = 2`
	`iterations = 4`

Copyright © 2026 Vizrt

5.4.5 Deep Learning Super Sampling

The NVIDIA DLSS technology provides feature enhancement, anti-aliasing and upscaling.

Note: The DLSS version integrated with Viz Engine is DLSS 2.4.2.

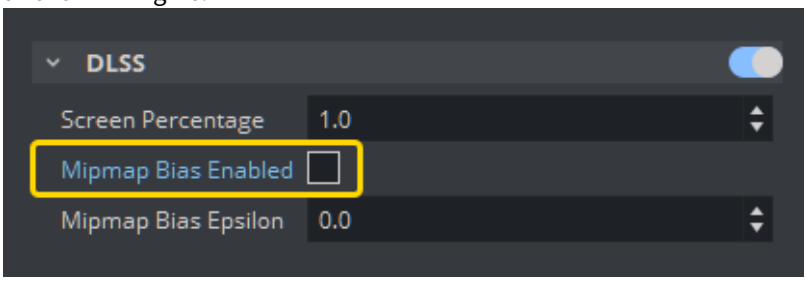
System Requirements

NVIDIA RTX GPU (GeForce, Titan or Quadro)
Minimum driver version 512.15
Windows 10 v1709
(Fall 2017 Creators Update 64-bit) or newer

Using DLSS - Temporal Anti-aliasing Upsampling

DLSS can be used as a means of upscaling when a scene is very performance demanding and it is not possible to render it at the final output resolution in real time.

- Set an output format from the available options in the **Config** > **Output Format tab**.
- Enable DLSS in the **Scene** > **Rendering** > **PostProcessing** > **DLSS** panel.
- Set a screen percentage (less than or equal to 1) based on the output format size, to set the effective render size for Viz Engine:



Using DLSS - Anti-aliasing Only (DLAA)

Follow the same steps as in Upscaling, but set a screen percentage of 1.

Note: Enable the Post-processing effects in the toolbar to visualize the DLSS output in the **Scene Editor**.

Copyright © 2026 Vizrt



Image Artifacts

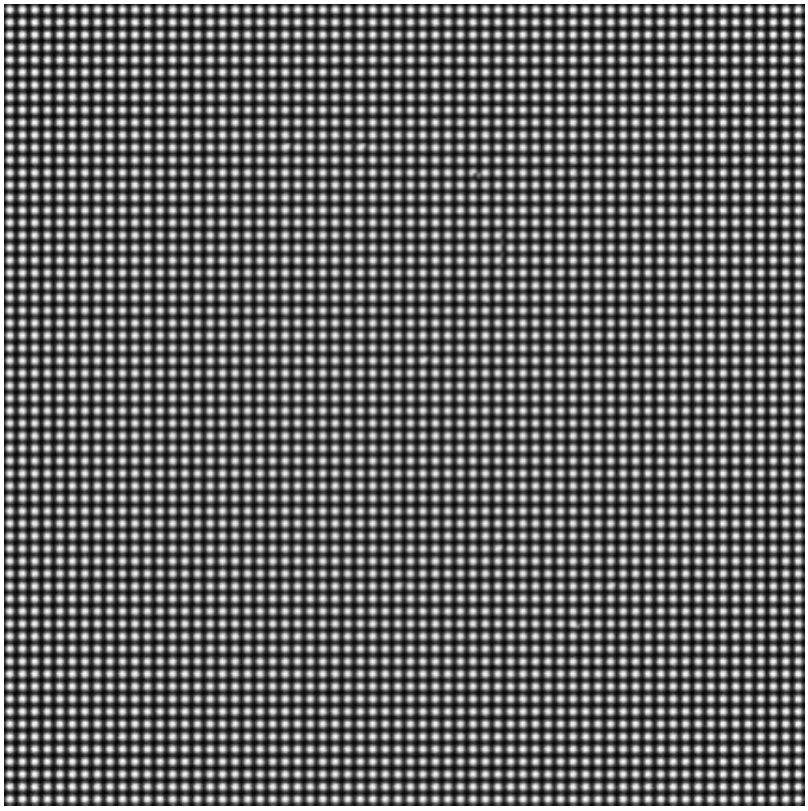
Clips

DLSS needs motion vectors to properly generate an output frame. Currently, Viz Engine only produces motion vectors based on geometry/camera movement and not individual pixel movements, which can cause slight smearing in some clips.

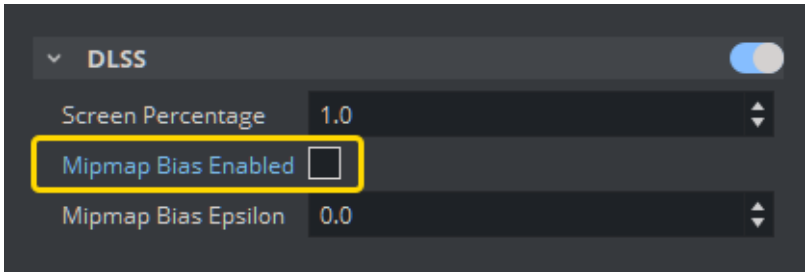
High Frequency Textures

DLSS has difficulty reconstructing a frame for high frequency textures, for example the image below:

Copyright © 2026 Vizrt



In these cases, disabling the **Mipmap Bias** greatly improves the quality of the DLSS output.



Quality Settings

Balanced

It is a middle ground between the render resolution and the output resolution, while trying to maintain the needed frame rate.

Maximum Quality

Favors pixel count and visual quality over frame rate, trying to reconstruct the frame as close as possible to the desired output size before super sampling it.

Copyright © 2026 Vizrt

Performance

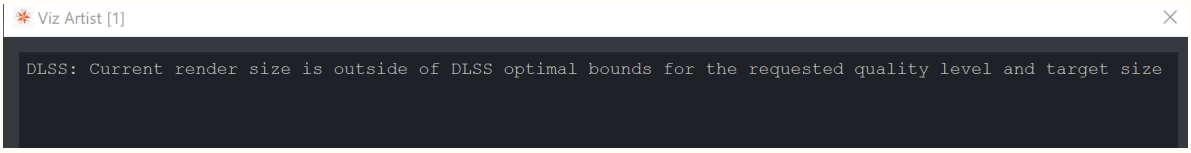
This is probably the most used quality setting when performance is critical but the visual aspect can't be neglected.

Ultra Performance

Favors frame rate over visual quality. Can cause blurriness and other artifacts. The following table shows the optimal screen percentage for the desired output format/quality:

Quality mode	Screen percentage	Optimal Size / Best Used with
Balanced	0.58	All formats.
Maximum Performance	0.5	1080p > UHD
Ultra Performance	0.3334	To render extremely high resolutions (for example, videowalls).
Maximum Quality	0.66	720p > 1080p

Note: If the screen percentage is outside the DLSS optimal range for the given quality mode/output format, Viz Artist displays a log message.



Copyright © 2026 Vizrt

5.4.6 SSAO

Screen Space Ambient Occlusion (SSAO) is used for efficiently approximating the ambient occlusion effect in real-time. Compared to Global Illumination, Ambient Occlusion is less accurate as it only operates in screen space, but it has its advantages. Similar to other post-processing effects like SSR or DOF, it is important that geometries are writing to the Z-Buffer. Otherwise, they won't be affected by this technique. For each pixel, the Screen Space Ambient Occlusion samples the depth values around this pixel and tries to compute the amount of occlusion on this point.

If your graphic only contains static elements, we recommend to use the Ambient Occlusion option within the Global Illumination features. The quality is much better and more accurate. For any dynamic content, you should use the Screen Space Ambient Occlusion feature.

Ambient Occlusion OFF	Real Time Ambient Occlusion	Global Illumination Baked AO
		

Advantages/Disadvantages of Real Time Ambient Occlusion

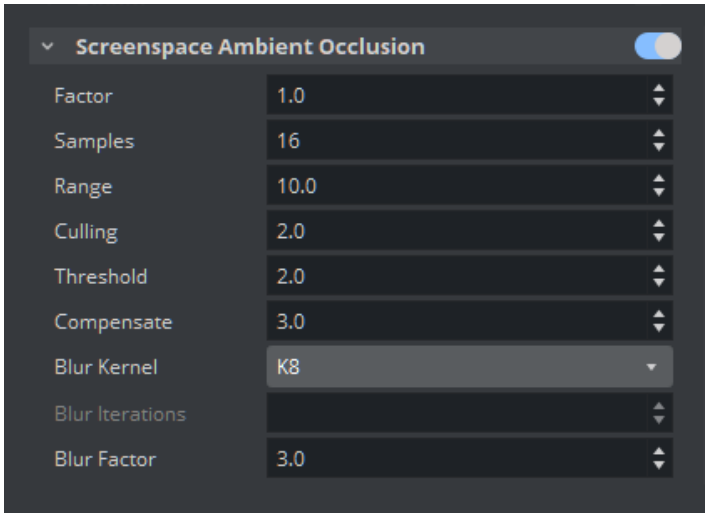
Advantages	Disadvantages
No pre-processing/baking required	Inaccurate
Works with dynamic objects	Performance impact

Copyright © 2026 Vizrt

Advantages	Disadvantages
Easy to use	

How to Use Ambient Occlusion

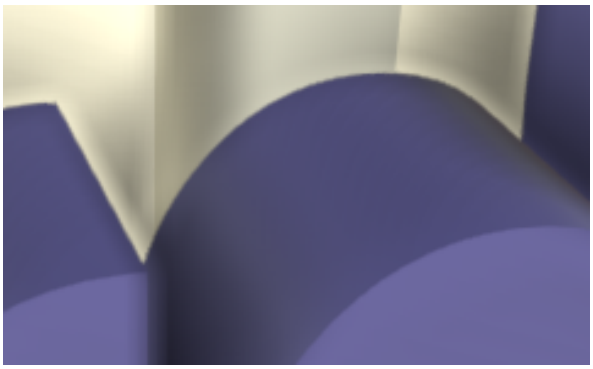
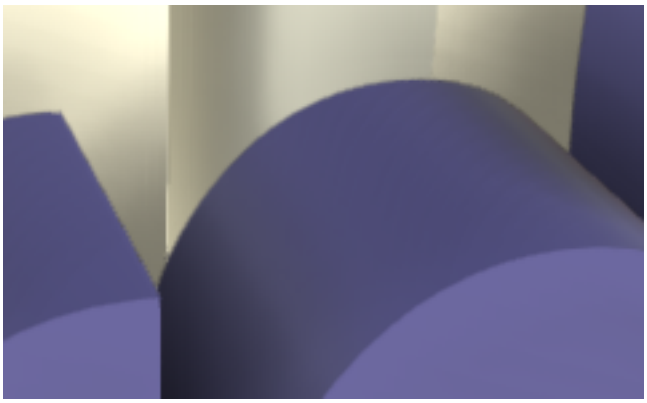
To use Ambient Occlusion, enable it in the Post Processing Effects. This is different from the Ambient Occlusion settings for the Classic Render Pipeline.



SSAO Parameters

- **Factor:** Controls the strength of this effect.
- **Samples:** Sets the number of rays used to gather AO information.
- **Range:** Determines the maximum range of each sample-ray.
- **Culling:** Determines how much irrelevant AO information surrounding geometry surfaces is discarded.
- **Threshold:** Determines the threshold to remove artifacts on geometry edges.
- **Compensate:** Compensates for edge artifacts.
- **Blur Kernel:** Controls the quality of the blur.
    - **K6 - K32:** Normal blur kernels that blur the AO in an entire frame.
    - **K1 - K2 (Edge-Aware):** Bilateral blur kernels that consider geometry edges upon blurring. Performance impact is expected.
- **Blur Iterations:** Controls how strong the AO gets blurred. This applies only to the Bilateral blur kernels.
- **Blur Factor:** Controls how strong the AO gets blurred. This applies only to the Normal blur kernels.

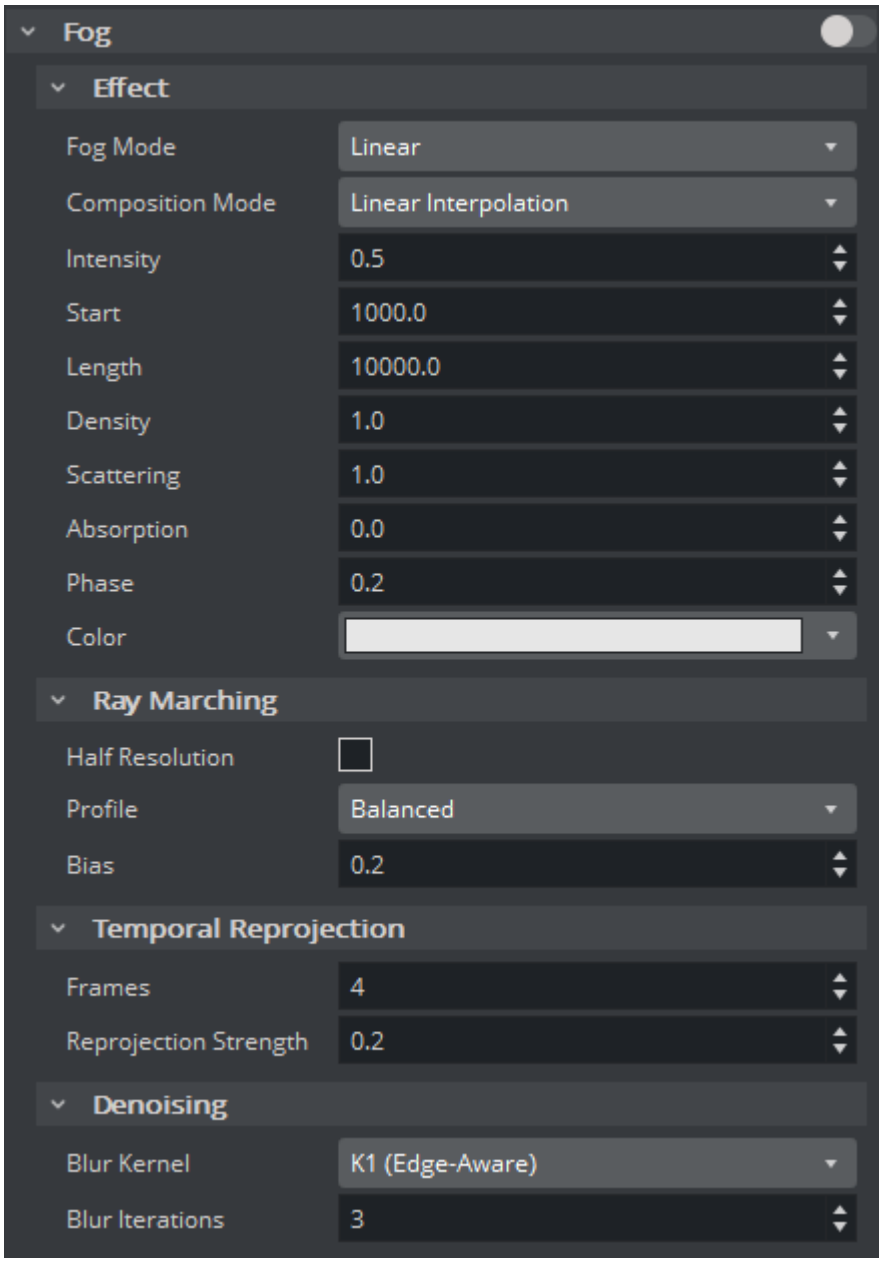
Copyright © 2026 Vizrt

Normal Blur - K12	Bilateral Blur - K2 (Edge-Aware)
	

Copyright © 2026 Vizrt

5.4.7 Fog

A Fog adds depth to the scene. It represents dust and dirt in the air at a given distance. The Fog effect is located in Scene Settings-Rendering inside the Post Processing effects.



Copyright © 2026 Vizrt

Fog Parameters

Effect

- **Fog Mode:** Sets how the transmittance through the fog is calculated.
  - **Linear:** Linearly proportional to the distance from the camera.
  - **Exponential:** Exponentially proportional to the distance from the camera. This makes the decay faster than Linear Mode.
  - **Exponential^2:** Exponentially proportional to the squared distance from the camera. This makes the decay faster than Exponential Mode.
- **Composition Mode:** Sets how the effect is composed to the main scene color. The following options are available:
  - **Linear Interpolation:** Composes by performing linear interpolation between scene the scene color and the fog color.
  - **Multiply:** Composes by dimming the scene according to how dense the fog is, with respect to the fog color.
- **Intensity:** Sets the intensity of the effect applied to the scene.
- **Start:** Sets the distance where the fog starts from the camera.
- **Length:** Sets the length of the fog from the camera. Only available for Linear Fog Mode.

> **Information:** In other Fog Modes, the length is assumed to be the camera's far plane. See Rendering Properties to customize the camera's clipping planes.

- **Density:** Sets the fog particles density.
- **Scattering:** Sets how much light is scattered on fog particles. This is the antecedent of the scattering-to-absorption ratio.
- **Absorption:** Sets how much light is absorbed by fog particles. This is the consequent of the scattering-to-absorption ratio.
- **Phase:** Sets how the light is scattered on fog particles. The negative value means the light is likely to be reflected, while the positive value means the light is likely to be transmitted.
- **Color:** Sets the color of the fog particles.

Ray Marching

- **Half Resolution:** Sets whether ray marching is performed in half resolution. This speeds up the computation at the cost of the quality.
- **Profile:** Sets what ray marching prioritizes. The options are *Performance*, *Balanced*, and *Quality*.
- **Bias:** Sets the bias of ray marching. `0` means to evenly sample along the ray, while `1` means to mostly sample near the camera. This could also reduce noise when the farthest point in the scene is deterministic.

Temporal Reprojection

- **Frames:** Determines the number of frames used to accumulate the effect (higher values can cause ghosting for moving objects/camera).

Copyright © 2026 Vizrt

- **Reprojection Strength:** Controls how strong the rays are reprojected for moving objects.

Denoising

- **Blur Kernel:** Controls the quality of denoising.
- **Blur Iterations:** Controls how strong the effect gets denoised.

Light Volumes

The fog also acts as a participating media for lights by nature, and it leads to a light volume with respect to the light type. This process of simulation requires ray marching and denoising controlled by the parameters in Ray Marching, Temporal Reprojection, and Denoising groups.

The following effect parameters setup is recommended for light volume rendering to achieve the most physically plausible result:

- Fog Mode = `Exponential` (or `Linear` if custom fog length is needed)
- Composition Mode = `Multiply`

To setup volume casting behavior of a particular light when the fog is enabled, see Light and Shadows.

Examples

Screenshot	Fog Settings

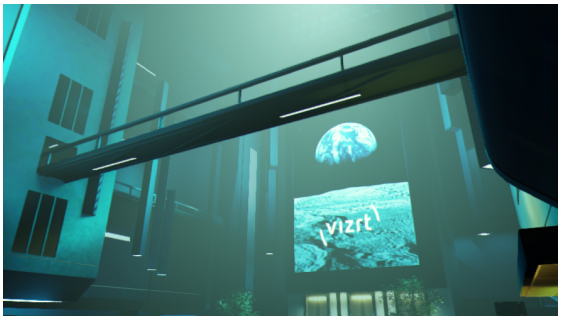
	`Intensity = 1.0`
	`Start = 0`
	`Density = 0.0`

	`Intensity = 0.4`
	`Start = 250`
	`Density = 0.0`

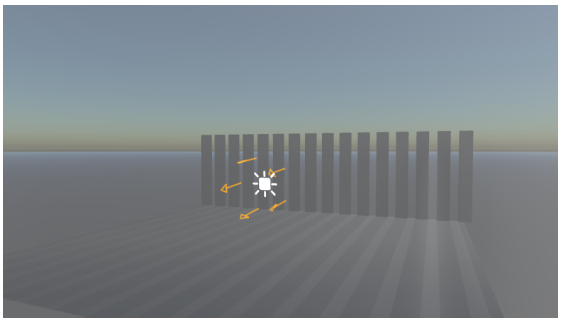
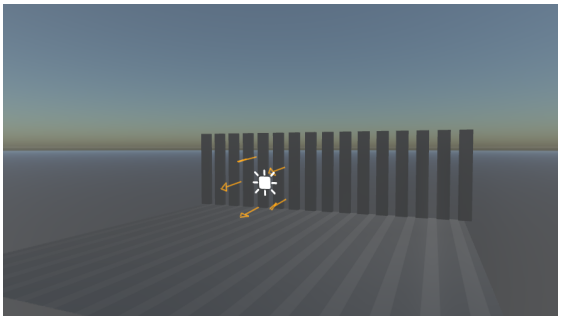
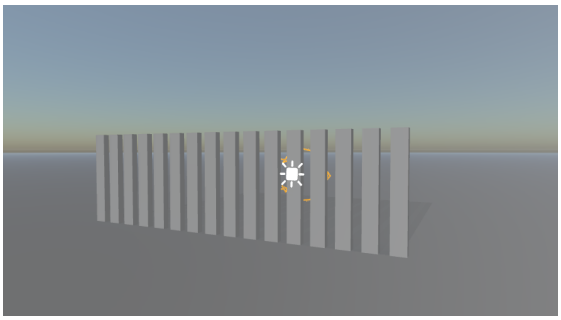
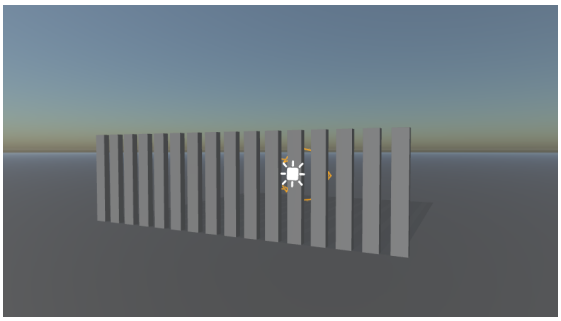
Copyright © 2026 Vizrt

Screenshot	Fog Settings

	`Intensity = 1.0`
	`Start = 170`
	`Density = 100.0`

	`Composition Mode = Linear Interpolation` `Light = Off`
	`Composition Mode = Linear Interpolation` `Light = On`
	`Composition Mode = Multiply` `Light = On`

Copyright © 2026 Vizrt

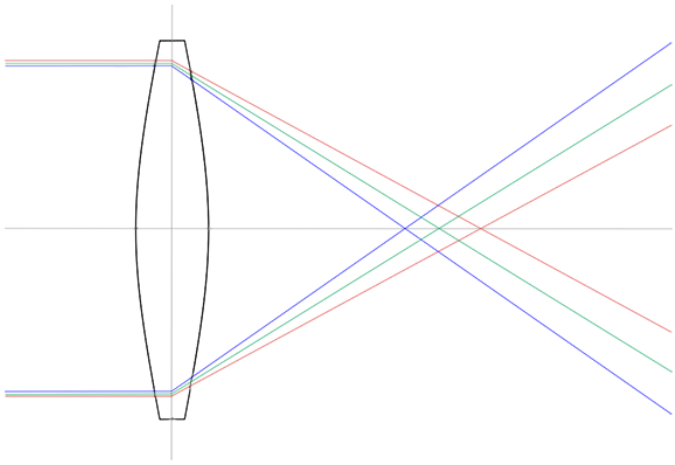
Screenshot	Fog Settings
	`Phase = 0.2` (Notice the light direction)
	`Phase = -0.2` (Notice the light direction)
	`Phase = -0.2` (Notice the light direction)
	`Phase = 0.2` (Notice the light direction)

Copyright © 2026 Vizrt

5.4.8 Chromatic Aberration

Chromatic Aberration is the effect of a defective lens where it fails to focus all colors into a single point. As cameras might show this in real life, in computer graphics this effect needs to be simulated to achieve a photo realistic look.

What happens inside the Lens can be illustrated like this:



In computer graphics, this effect is simulated to achieve an effect as if your graphic is filmed through a real camera.

Without Chromatic Aberration

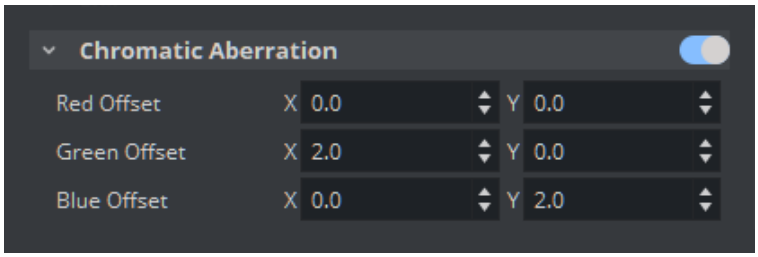
With Chromatic Aberration


Chromatic Aberration is built in as a post process effect in the Viz Engine Render Pipeline.

- Make sure you are using the Viz Engine Render Pipeline and Post Process Effects are turned on in your Scene Editor.

Copyright © 2026 Vizrt

- Switch to Scene settings and turn on **Chromatic Aberration** in **Post Processing**.



X and Y define the offset added to single RGB channels. The higher these values are, the more distorted the lens. Typical values are around `1` - `2` , anything else looks unrealistic.

Copyright © 2026 Vizrt

5.4.9 Light Glares

The Light Glares effect is used to simulate how eyes and camera lenses perceive bright spots (direct, reflected sunlight or artificial light) such as car headlamps at night. Light glares can be enabled in the Lens Effects of the Scene Settings. The more emissive a material is, the more intense the effect is. Light Glares work best in combination with blooming effects.



Light Glares Parameters



Copyright © 2026 Vizrt

- **Threshold:** Controls the basic brightness threshold.
- **Intensity:** Controls the intensity for the streak effect.
- **Soft Threshold:** Softens the pixel selection instead of a hard cut.
- **Samples:** Determines the number of samples to use.
- **Length:** Controls the length of the light glares.
- **Streaks:** Generates a number of streaks.
- **Spread:** Sets distance between different colored glare components.
- **Offset angle:** Determines starting offset.
- **Step angle:** Determines angle between two glares. This is only used if the streak count is `4`. Otherwise, the streaks are evenly distributed.
- **Swirl:** Creates additional rotation effect depending on the screen position.
- **Spectral color 1:** Determines first spectral component color.
- **Spectral color 2:** Determines second spectral component color.
- **Spectral color 3:** Determines third spectral component color.

Copyright © 2026 Vizrt

5.5 Color Space

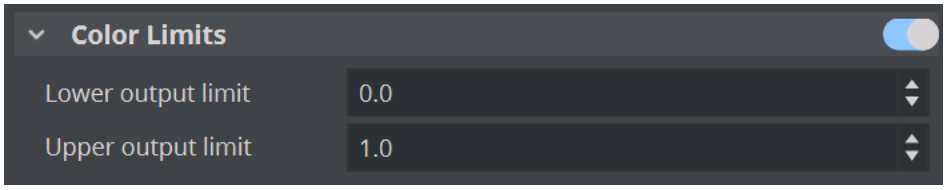
This section contains information on the following topics:

- Color Limits
- Tonemapping

Copyright © 2026 Vizrt

5.5.1 Color Limits

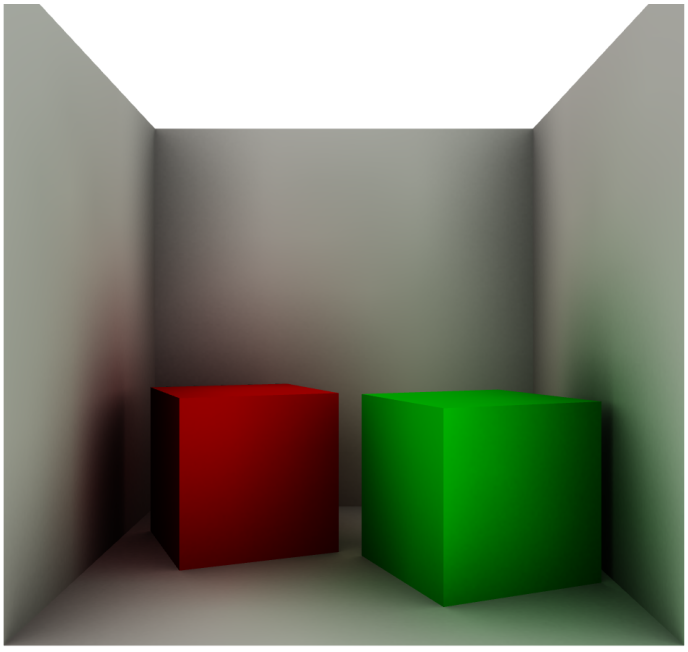
Color Limits (Gamma Correction) can be used to control the overall brightness of your output if it is either bleached out or too dark. Output to SDI usually does not require any further gamma correction. These signals already incorporate a gamma compression matching a typical CRT and these values are fixed and defined by analog video standard.



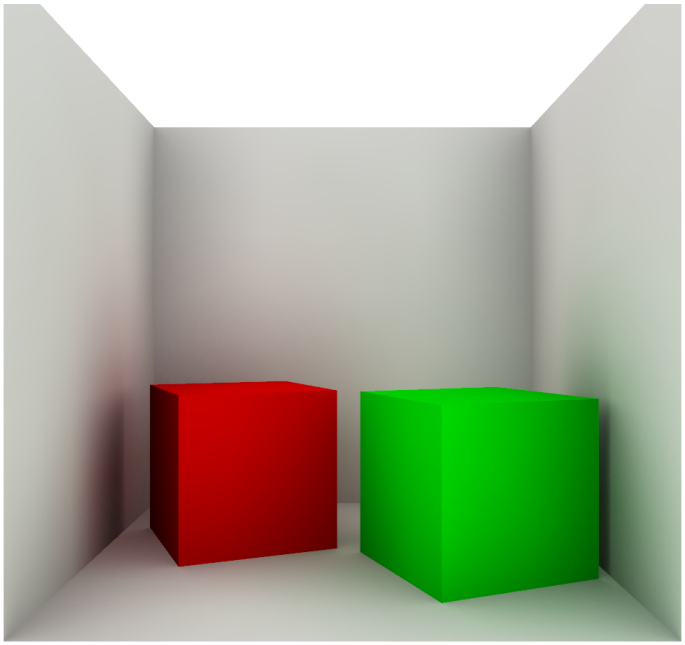
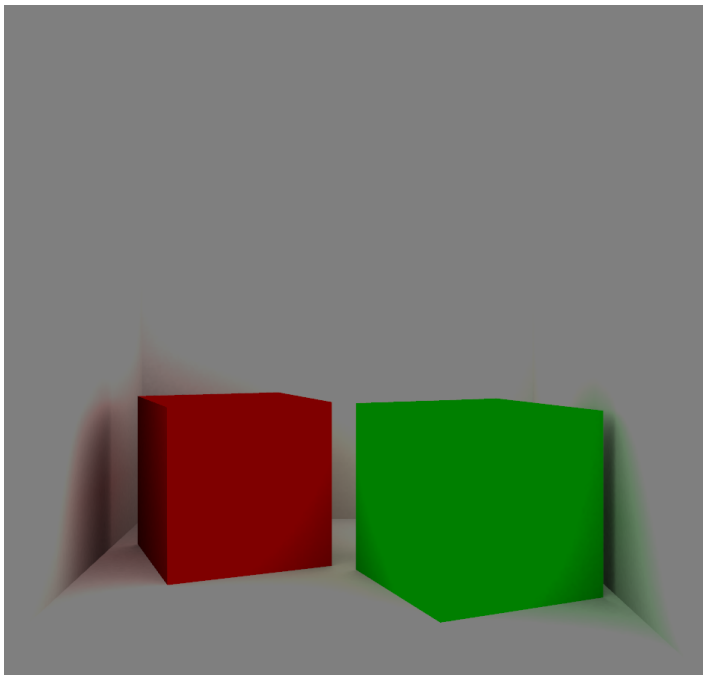
However, output to other devices or broadcast over stream based channels like NDI might require some adaption to the final output signal.

If enabled, your output is normalized in between the lower and the output limit.

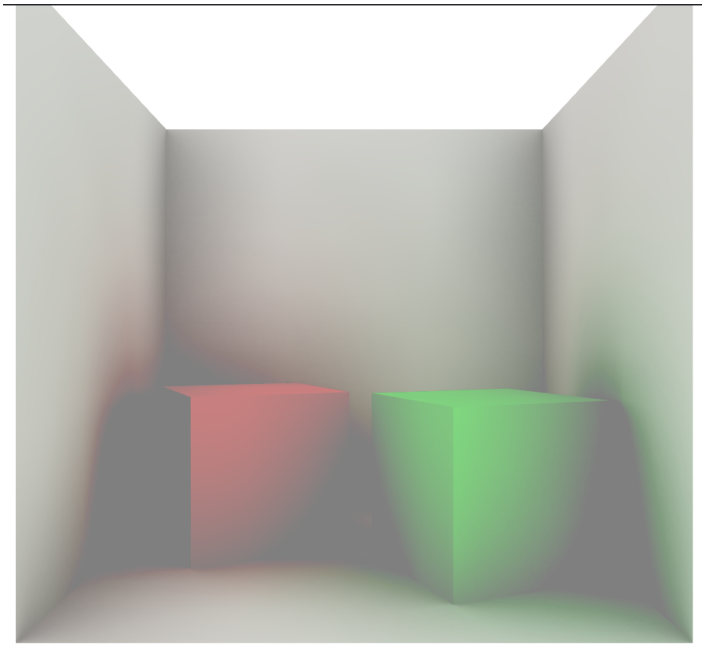
Examples

Upper and Lower limit turned off.	

Copyright © 2026 Vizrt

Lower output limit: **0.0** Upper output limit: **1.0**	
Lower output limit: **-1.0** Upper output limit: **0.5**	

Copyright © 2026 Vizrt

Lower output limit: **0.5** Upper output limit: **1.0**	

Copyright © 2026 Vizrt

5.5.2 Tonemapping

Tonemapping is a technique that converts color values from a high dynamic range (HDR) to a low dynamic range (LDR) a display can output.

The tonemapping shader is directly integrated into the Viz Engine Render Pipeline and is applied as the last step of the main rendering process, just before the image is processed for video output.

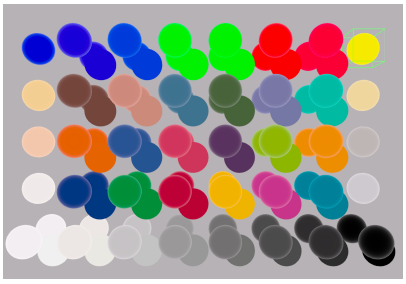
To enable the tonemapping post effect, go to the Scene's **Rendering Settings** and enable **Tonemapping** under **Color Space**. Be sure to enable post-processing effects in the Scene Editor to have a preview of it while editing its properties.

There are two available Tonemapping Operators:

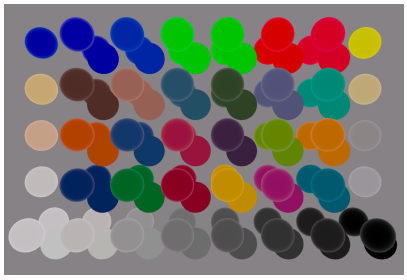
- Neutral
- Filmic

Neutral (Khronos) Tonemapper

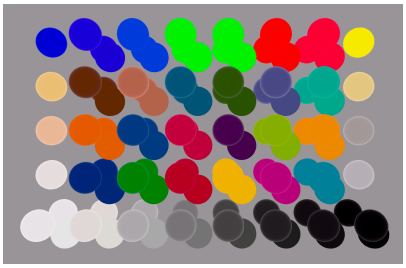
The neutral tonemapper keeps neutral colors up to a certain maximum and only maps highlights into sRGB color space. This allows to keep all neutral colors, but clamps overshooting shadows and highlights.



No tonemapping applied



Filmic Tonemapper
(Note color changes on all spheres)



Neutral Tonemapper
(Note color changes only on outer spheres)

The neutral Tonemapper should be used if your colors match a given Corporate design.

Filmic Tonemapper

The filmic tonemapping operator proposed by John Hable is used.

The order of the adjustments to the input color is:

- Exposure
- Saturation
- Contrast
- Tonemap
- Lift/Gamma/Gain

Copyright © 2026 Vizrt

Exposure / Saturation / Contrast

Parameter	Description
`exposure`	Adjusts the exposure of the input pixels.
`saturation`	Adjusts the saturation of the input pixels.
`contrast`	Adjusts the contrast of the input pixels.

Examples

Screenshot	Information

	Default Settings
	`contrast = 1`
	`exposure = 0`
	`saturation = 1`

	`exposure = 0.5`

Copyright © 2026 Vizrt

Screenshot	Information
	`contrast = 1.2`
	`saturation = 0.2`

Tonemapping Operator

The function that is applied on the HDR input colors can be configured using multiple parameters in the Render Graph node. To see how the operator works check out the interactive graph.

Parameters

Parameter	Description

`shoulderStrength`
`linearStrength`
`linearAngle`
`toeStrength`
`toeNumerator`
`toeDenomiator`
	Defines the shape of the curve.

Copyright © 2026 Vizrt

Parameter	Description
`whitePoint`	Defines the value in the input that translates to `1` in the output. All values below are squeezed into the range from `0` to `1`, everything above results in values larger than `1`.

Examples

Screenshot	Information
	Tonemapping Disabled
	Tonemapping Enabled Default values.

Copyright © 2026 Vizrt

Screenshot	Information
	Tonemapping Enabled Adjusted values compared to the default ones, to have a slightly improved saturation.

Color Correction

Color correction is done after tonemapping and applies the so called Lift, Gamma and Gain input values: `output = (input * gain + lift) ^ gamma.`

The formula being used is mentioned in https://en.wikipedia.org/wiki/ASC_CDL. To see what the separate values do check out the interactive graph.

Parameters

Parameter	Type	Description
`gain`	Multiplier	Adjusts highlights.
`lift`	Offset	Adjusts brightness.
`gamma`	Exponent	Adjusts mid-tones.

Copyright © 2026 Vizrt

Examples

Screenshot	Description

	Default Settings
	`lift = 0`
	`gamma = 1`
	`gain = 1`
	for red, green and blue channels.

	Adjusted `lift` for red and green channels, leaving blue untouched.
	Adjusted `gamma` for red and green channels, leaving blue untouched.

Copyright © 2026 Vizrt

Screenshot	Description
	Adjusted `gain` for blue channel, leaving red and green channels untouched.

Credits

Environment image is taken from https://hdrihaven.com/hdri/?h=leadenhall_market.

Copyright © 2026 Vizrt

5.6 Working with Phong Materials

- Color
- Ambient
- Diffuse
- Specular
- Emissive
- Final Result
- Unlit
- Reflectivity
- Alpha
- Texture Generation
- Clamping

The Viz Engine Render Pipeline focuses on rendering photorealistic scenes. In some cases, the physically-based rendering (PBR) material is too complex. For such scenarios, the Phong material was added. This material type is less complex, has similar properties to the known classic materials, and is usable without a proper light setup, which makes it easy to use for simple 3D geometries and all 2D graphics like lower thirds.

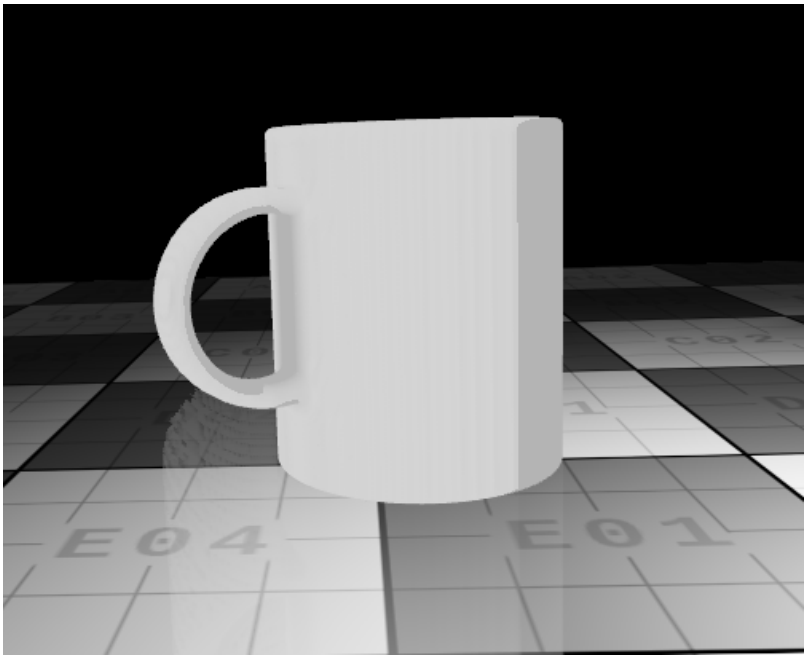
Similar to the PBR material, this material is only usable in the Viz Engine Rendering Pipeline and can be added to a container by right-clicking and select **Add Phong Material (Viz Engine Pipeline)** which adds a default Phong material to your object. The Phong material is based on the material definition and has the same functionality in Graphic Hub as the PBR material.

5.6.1 Color

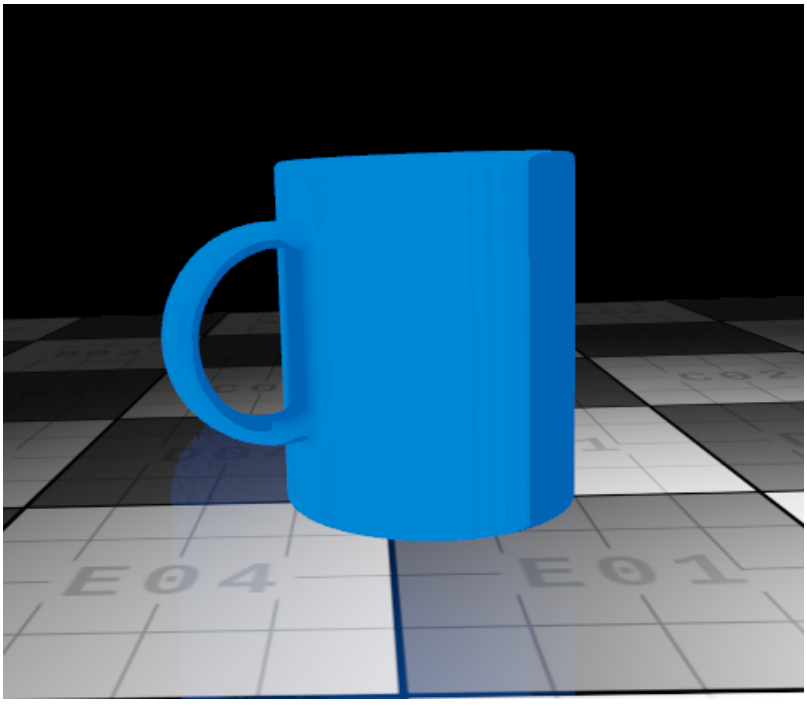
Defines the basic color. Diffuse and specular channels get multiplied to this channel. If a texture is going to be used, the base color can be kept white, unless you want to recolor your image.
Alpha information on textures can be used, but only on the Basecolor component. All other maps discard the alpha information.

White Basecolor

Copyright © 2026 Vizrt



Blue Basecolor



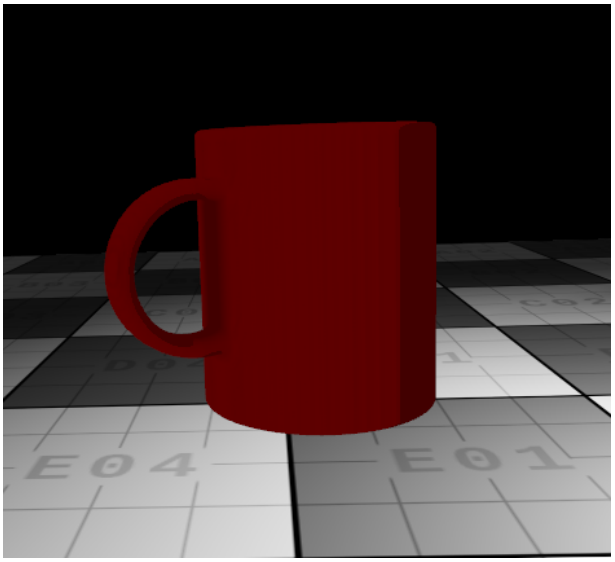
White Basecolor + Texture

Copyright © 2026 Vizrt



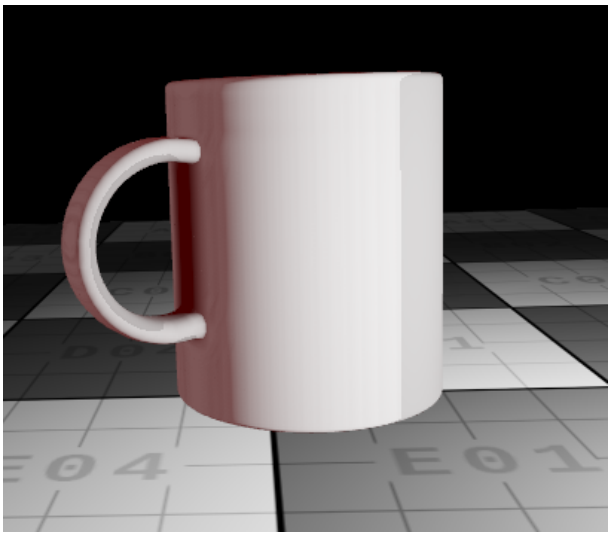
5.6.2 Ambient

The ambient component is used to mimic global illumination by giving it a base color that is present in the scene and unaffected by any other light. As this component is always added to the final color, it should be used with low intensity.

White Basecolor Red Ambient 0.1 intensity	White Basecolor + Texture Grey Ambient 0.5 intensity
	

5.6.3 Diffuse

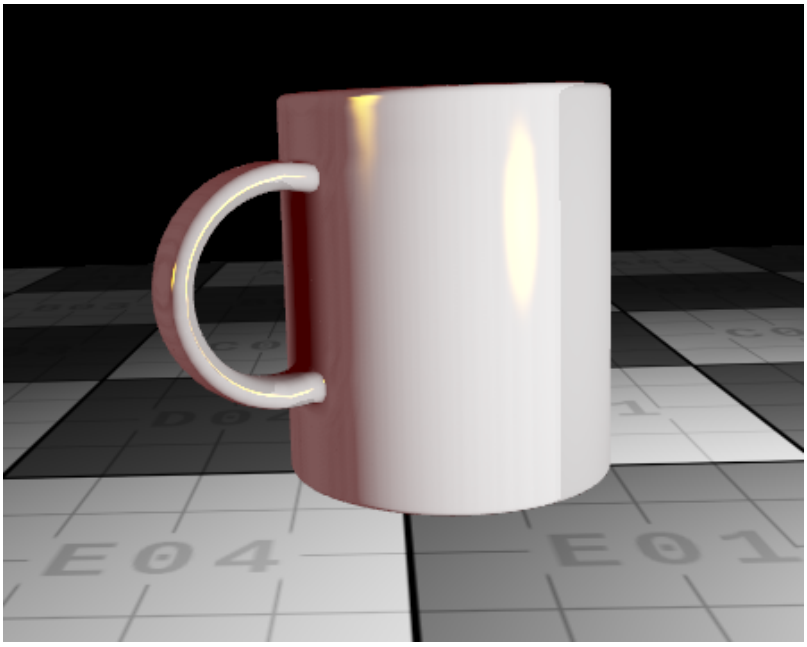
Defines the color of the reflection of incident light in all directions from a surface, so that diffuse illumination of a surface is independent of the camera position.

White Basecolor Red Ambient 0.1 intensity Grey diffuse 1.0 intensity	White Basecolor + Texture Black Ambient 1.0 intensity Grey diffuse 1.0 intensity
	

5.6.4 Specular

White Basecolor Red Ambient 0.1 intensity Grey diffuse 1.0 intensity Orange Specular 1.0 intensity

Copyright © 2026 Vizrt



White Basecolor + Texture
Black Ambient 1.0 intensity
Grey diffuse 1.0 intensity
White Specular 0.8 intensity
Shininess 600



Copyright © 2026 Vizrt

White Basecolor + Texture
Black Ambient 1.0 intensity
Grey diffuse 1.0 intensity
White Specular 0.8 intensity
Shininess 20

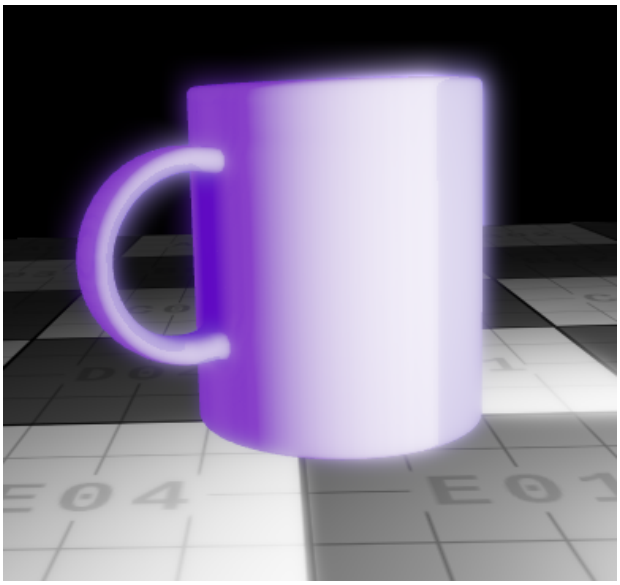


The specular component defines the mirror-like reflection of light rays from a surface. It is only visible if reflections on the surface of the light directly hit the camera. This shininess factor affects the specular channel and controls the sharpness of the specular effect. Small values create flat/smooth highlights and higher values create sharp highlights.

5.6.5 Emissive

The emissive color is used to color parts of the surface which should act like they are emitting light. This can be useful to create glow effects in combination with the bloom post-processing effect.

Copyright © 2026 Vizrt

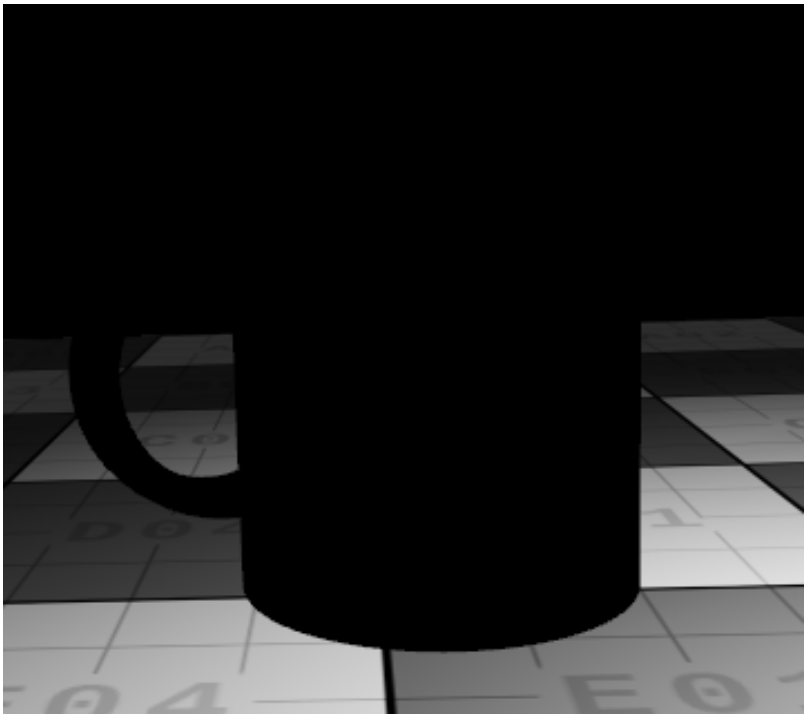
White Basecolor Red Ambient 0.1 intensity Grey diffuse 0.4 intensity Orange Specular 0.0 intensity Blue Emissive 1.0 intensity Shininess 600	White Basecolor + Texture Black Ambient 1.0 intensity Grey diffuse 1.0 intensity White Specular 0.8 intensity Shininess 600 Orange Emissive 1.0 intensity
	 Emissive Map: 

5.6.6 Final Result

Every component added together gives the final material look.

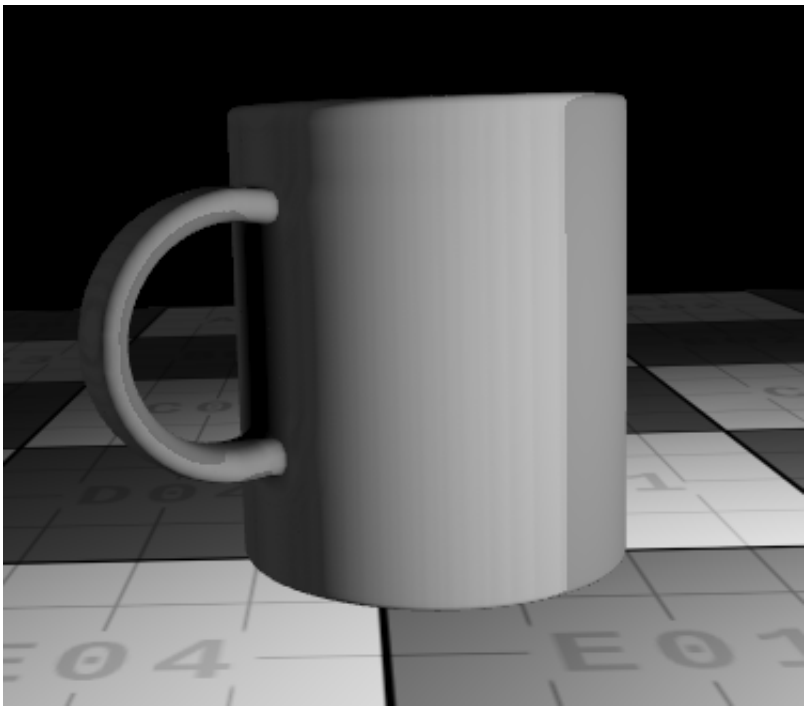
Copyright © 2026 Vizrt

Base Color

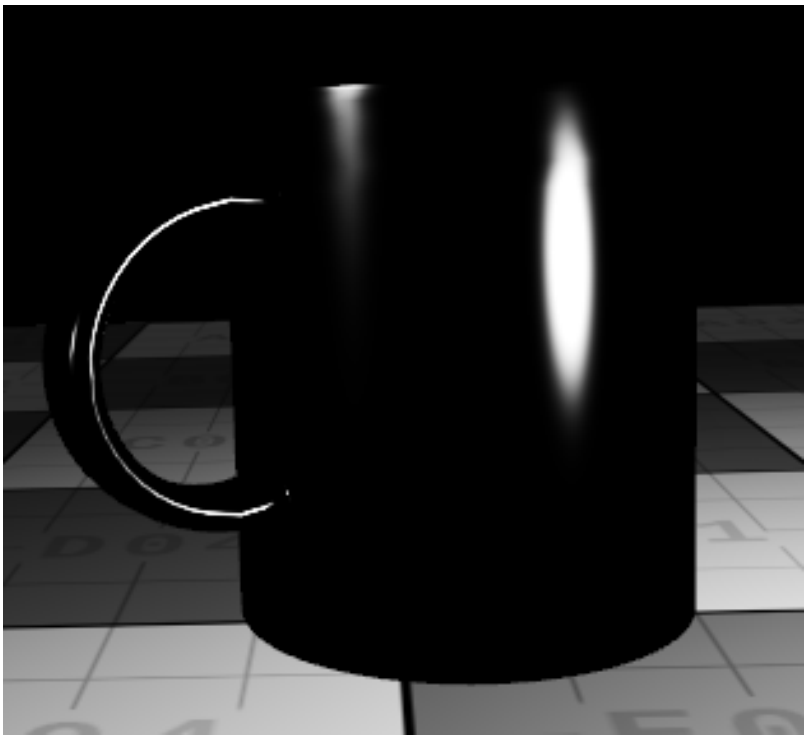
Ambient


Copyright © 2026 Vizrt

Diffuse



Specular



Copyright © 2026 Vizrt

Emissive

Final Result


Copyright © 2026 Vizrt

5.6.7 Unlit

Similar to the classic material, the lighting can be disabled. This means that no scene lights affect the material. This is usually desired for 2D objects and is controlled by the **Lit** checkbox. Unlit materials are defined by the color channel and the emissive channel.

In Unlit mode, it's possible that the geometry also receives shadows. This is controlled by the **Shadow** checkbox. In Lit mode, shadows are always visible.

Unlit w/o Shadows



Unlit with Shadows



Lit



Copyright © 2026 Vizrt

5.6.8 Reflectivity

The Reflective property defines if the material is able to reflect its surroundings. It has the same limitations as the screen space reflection. It can only reflect areas, that are rendered. Parts that are invisible to the camera, can not be reflected.



5.6.9 Alpha

Similar to the PBR material, the alpha can be controlled with the Alpha plug-in. Also, the color channel alpha value is taken into account.

25%

Copyright © 2026 Vizrt



50%



75%

Copyright © 2026 Vizrt

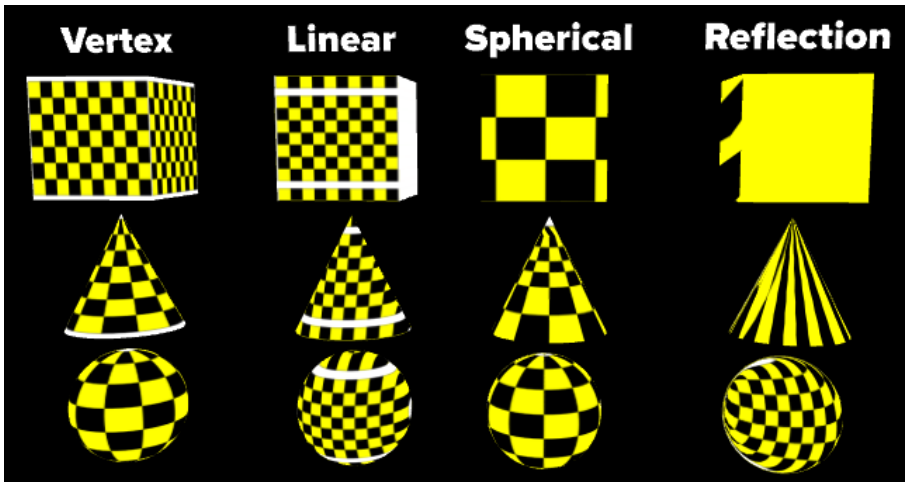


100%



Copyright © 2026 Vizrt

5.6.10 Texture Generation



Besides using the texture coordinates from the geometry, it is also possible to automatically generate them.

There are four modes to describe how the texture coordinates are organized:

- **Vertex:** Uses the texture coordinates from geometry.
- **Linear:** Uses the linear projection of texture coordinates.
- **Spherical:** Uses the spherical projection of texture coordinates.
- **Reflection:** Uses the texture coordinates from the reflection vector of the surface.

Additionally, texture coordinates can be transformed by the following properties:

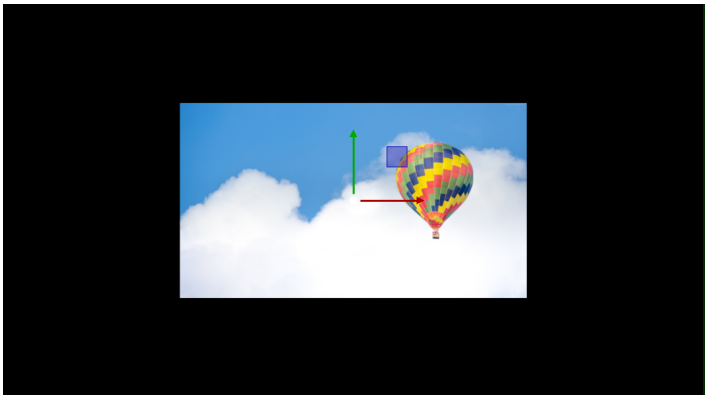
- **Position:** Shifts the texture coordinates along the global axis of the texture.
- **Offset:** Shifts the texture coordinates along the local axis of the texture after the rotation.
- **Rotation:** Rotates texture coordinates.
    - **X (only available in Linear mode):** Stretches or squeezes texture coordinates vertically, as if it is rotated around the global horizontal axis of the texture.
    - **Y (only available in Linear mode):** Stretches or squeezes texture coordinates horizontally, as if it is rotated around the global vertical axis of the texture.
    - **Z:** Rotates on the texture plane.
- **Scaling:** Stretches or squeezes texture coordinates along the local axis of the texture after the rotation.
- **Preserve aspect ratio (only available in Linear and Reflection modes):** Preserves the textures aspect ratio on the 1:1:1-scaled geometry.
- Advanced properties:
    - **Rotation Pivot:** Sets the center where texture coordinates rotate around.
    - **Scaling Pivot:** Sets the center where texture coordinates stretch from or squeeze towards by scaling.
    - **Pre-rotation Scaling Factors:** Stretches or squeezes texture coordinates along the global axis of the texture.
    - **Pre-rotation Scaling Pivot:** Sets the center where texture coordinates stretch away by pre-rotation scaling.

Copyright © 2026 Vizrt

Using Linear mode with Rectangle: To use the texture with Rectangle in Linear mode, given that the Preserve aspect ratio is enabled, the Position needs to be initialized to (0.5, 0.5) first to make the texture appear at the center of the Rectangle.

5.6.11 Clamping

Textures can be set to clamp to a border color. This means instead of repeating the last pixels, they can be overridden by a border color. This can be set by using the Wrapping mode in U and/or V:

Repeat Mode	
Clamp Mode	
Clamp to Border	

Copyright © 2026 Vizrt

Information: The Alpha mode of the border currently only works for RGBA images

Copyright © 2026 Vizrt

5.7 Working with PBR Materials

- Color or Albedo Map
- Emissive Color
    - Specular Anti-aliasing
- Normal Maps
- Roughness
- Metallic
- Ambient Occlusion
- Height Maps
- Environment Maps
- Improved Image Import for PBR Materials
- Using Transparency Effects
- Texture Manipulation
- Clamping
- Working with Material Definitions on Graphic Hub
    - Working with Referenced Materials



The Viz Engine Render Pipeline uses the physically-based rendering (PBR) approach to bring photo realism alive. PBR describes the shading of a geometry using realistic shading/lighting models. PBR materials attempt to render surfaces as accurately as possible using modern shader technologies, ensuring your objects look great under every lighting condition.

To start using PBR materials, make sure you are running in the Viz Engine Render Pipeline, your scene has a light defined and you are using a correct environment map for proper lighting.

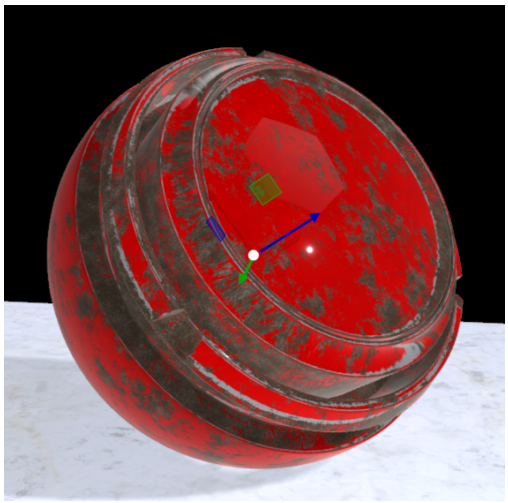
In our example, we use the default Preview Sphere from Allegorithmic Substance Painter samples. Import your geometry (or use a built-in geometry) and add it to your scene. Drag any Material Definition from the server view to a container or right-click and select **Add Material (Viz Engine Pipeline)...**. This adds a default material to your object.

Material Definitions can easily be saved or re-used by dragging them to or from Graphic Hub.

Copyright © 2026 Vizrt

The traditional material in the Classic Render Pipeline has a diffuse setting, describing the the color of diffused light. Working with PBR materials, the same setting is now represented by the color (or sometimes called albedo component). This can be a color only or a texture, typically mapped using UV coordinates.

5.7.1 Color or Albedo Map



Defines the basic color of any diffuse light hitting the geometry. As the Viz Engine Render Pipeline renders in linear space, the sRGB flag defines whether a sRGB texture is in use. If you are not using a UV mapped image, you can change the wrapping method. Additional filter settings for scaling can be applied (linear filtering and nearest filtering algorithms). Usually, those settings do not need to be changed unless you see some jaggy artifacts.

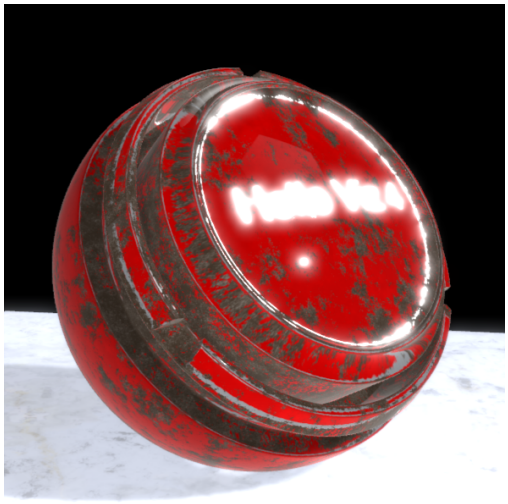
The anisotropic filtering value defines how pixels that need to be stretched, are filtered in z-order. A higher filtering reduces blurriness in the back and adds more detail.

If a texture is going to be used, the base color can be kept white, unless you want to recolor your image. In addition, an alpha value can make the object transparent (without any special transparency effects). Albedo maps should not include any kind of lighting information.

5.7.2 Emissive Color

The emissive component defines the areas and color of any light hit by the object and reflected again. Substance Painter can also export an additional emissive texture. If you want to add some blooming effects afterwards, the emissive map can define these highlight areas.

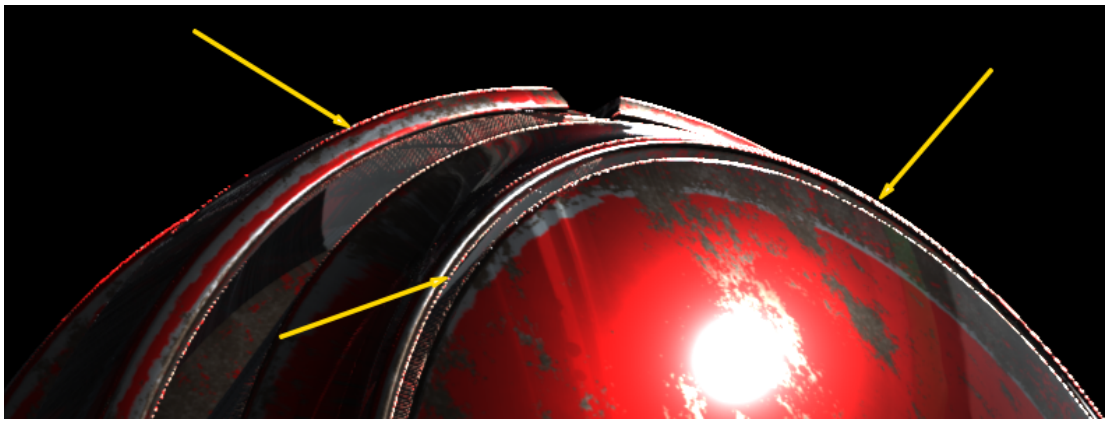
Copyright © 2026 Vizrt



In the given example, we added some highlight effects on the top surface. This is done using an emissive map, the intensity factor and the blooming post-processing step. The base color should be kept black, otherwise your geometry emits all light.

Specular Anti-aliasing

In some conditions, if your light source hits a beveled surface, the lighting effects on the edges can be over sampled, as in the following image:



To correct these unwanted artifacts, use the **Specular Anti-aliasing** settings in the Material Settings.

5.7.3 Normal Maps

A normal map is used to add details to your surface without adding additional polygons. Normal maps do respect any light source, however they do NOT take any global illumination effects into account. Normal maps are exported by any common 3D paint program and can easily identified by theirs blueish/pink color range. These colors define if a certain pixel appears lowered or raised on the surface.

Copyright © 2026 Vizrt

No Normal Map (left) vs Normal Map applied (right)



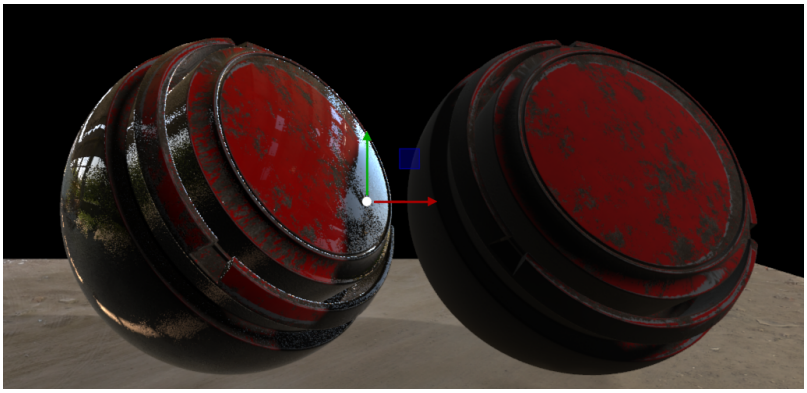
Tip: The Viz Engine Renderer uses the OpenGL normal map format. If only the DirectX format is available, you need to invert the green channel in an external image editing program like Adobe Photoshop, resave it and reimport the normal map into Viz Engine.

5.7.4 Roughness

Roughness maps (or microsurfaces) defines how rough or smooth the surface of a material is. Rougher surfaces show wider but dimmer specular reflections, whereas smoother surfaces show more brighter but sharper reflections.

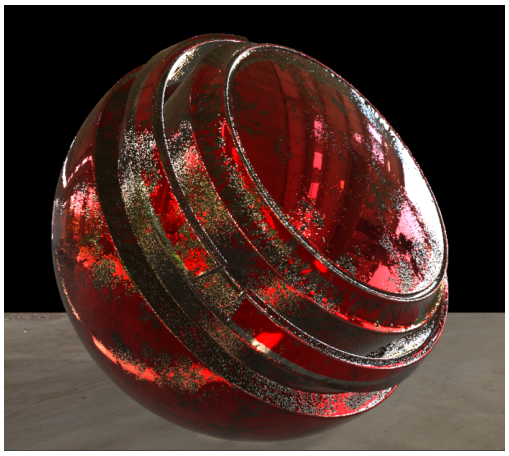
In the Viz Engine Render Pipeline, a roughness factor of `-1` defines a very smooth surface, `+1` the opposite. If you play with surface reflective settings, the roughness factor is very important.

Roughness Factor to `-1` (left) vs Roughness Factor `+1` (right)



Copyright © 2026 Vizrt

5.7.5 Metallic

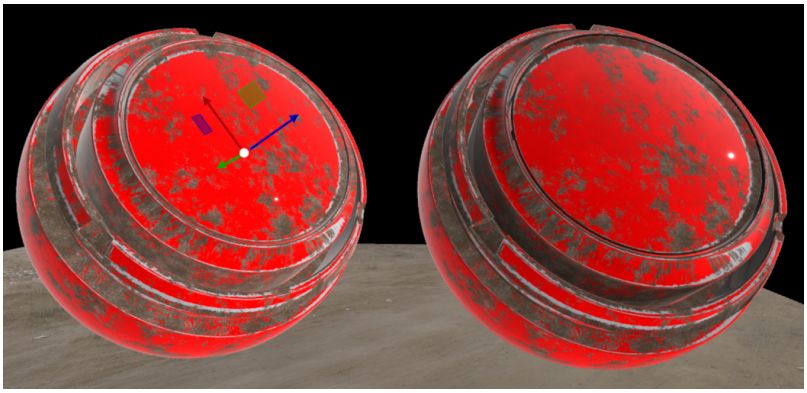


The metallic map defines how reflective the surface of a material is. Insulative surfaces (black pixels) use the albedo map for the diffuse value. Metallic surfaces (white pixels) use the specular color and the intensive from the albedo map, the diffuse part is set to `0`. The roughness factor and the metallic map influence each other. A factor of `-1` means the surface is not reflective at all (can be used to represent rubber or a similar surface), whereas a high positive number (`+1`) is used for polished metals.

5.7.6 Ambient Occlusion

Ambient occlusion is a part of a baking process like the Global Illumination effects. It adds all those tiny little shadow effects in the certain areas of your geometry. When using ambient occlusion as an additional map, the shader can only apply it on ambient diffuse lit areas and not on specular parts.

Without (left) and with ambient occlusion applied (right)



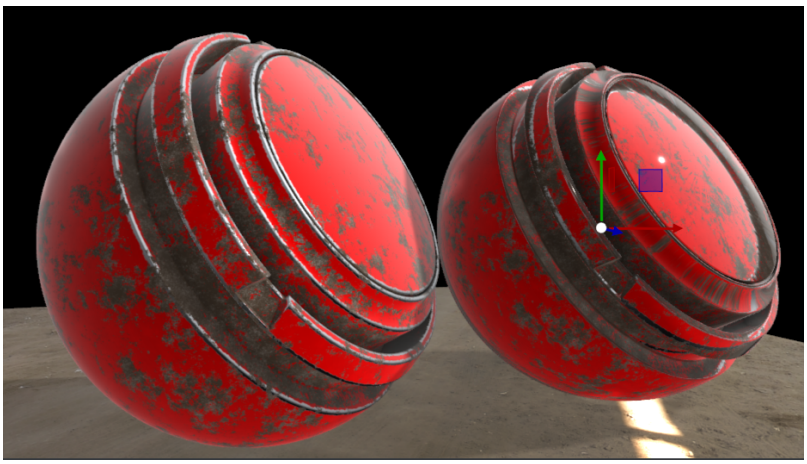
Copyright © 2026 Vizrt

5.7.7 Height Maps

A height map (parallax mapping) is similar to normal mapping, but it requires way more complex calculations. Normal mapping modifies the light across the surface, whereas height mapping shifts the geometry to achieve a surface level occlusion effect. This applies to the surface only, the outline (silhouette) is not touched. This, of course, requires a high resolution polygon.

Warning: Be careful when using height maps, especially on large geometries. They consume a lot of render time.

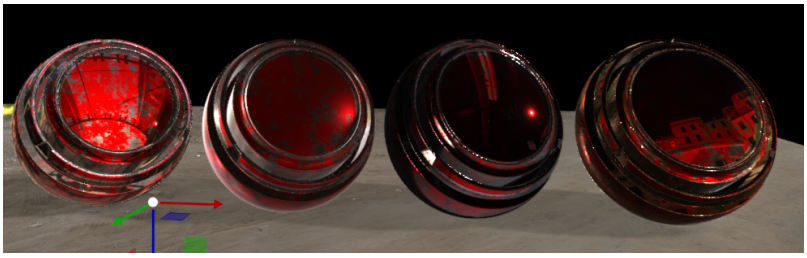
No height map (left) vs height map applied (right)



5.7.8 Environment Maps

By default, each geometry uses the default scene environment map. Under certain circumstances (for example, when simulating glass effects), it might be useful to apply another environment map than the default one.

Various Environment Maps

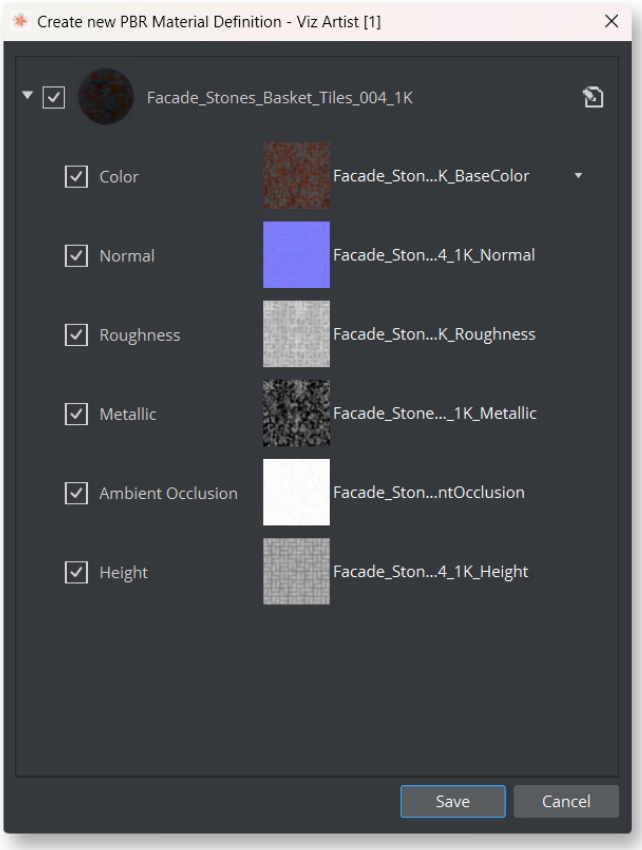


For more information, see Environment Maps.

Copyright © 2026 Vizrt

5.7.9 Improved Image Import for PBR Materials

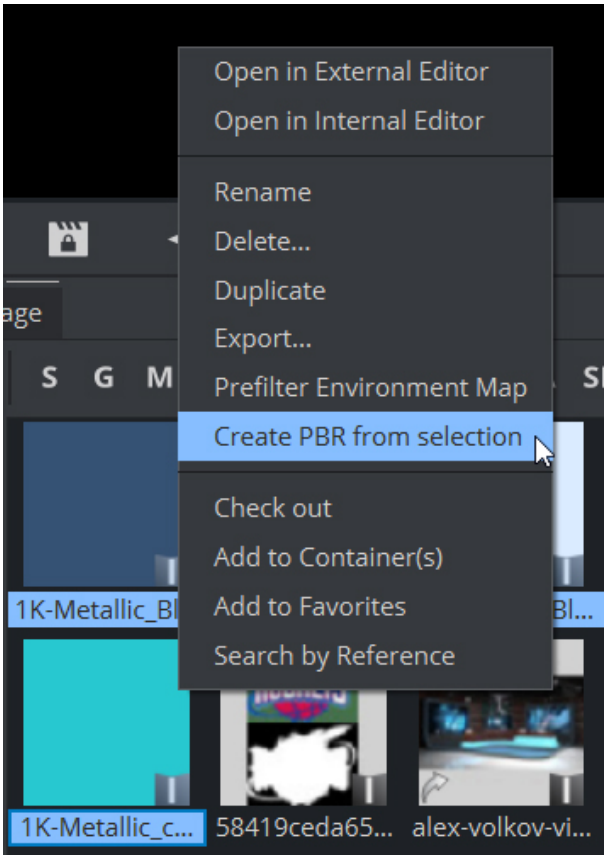
When dragging a collection of images or a *.zip* file into the asset view, the importer automatically recognizes if these files belong to a PBR collection. The identified images are presented in a dialog and the user can choose if the images are correct or, if multiple choices are available, select the right one.



To create a PBR automatically:

- Drag a *.zip* file containing multiple images to the Asset view, OR
- Drag multiple images to the Asset view, OR
- Use the context menu on multiple images in the Asset view:

Copyright © 2026 Vizrt

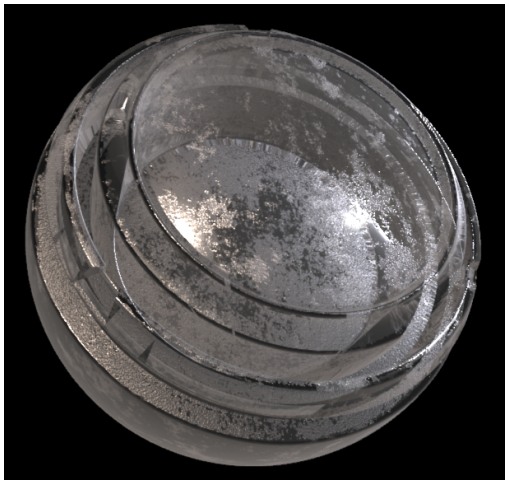


Information: If your images are not identified automatically, you can modify the naming convention list by opening the file *C:\ProgramData\vizrt\VizArtist\MaterialDefinitions.json* and adding your own pattern.

5.7.10 Using Transparency Effects

Creating photo-realistic glass effects have always been a challenge. This is now much more easier using the new PBR material editor. The refractive index in combination with the Alpha Fresnel Power makes your object transparent and adds some refraction effects. The refractive index is the deflection of light when it pass trough your surface to another surface with different density.

Copyright © 2026 Vizrt

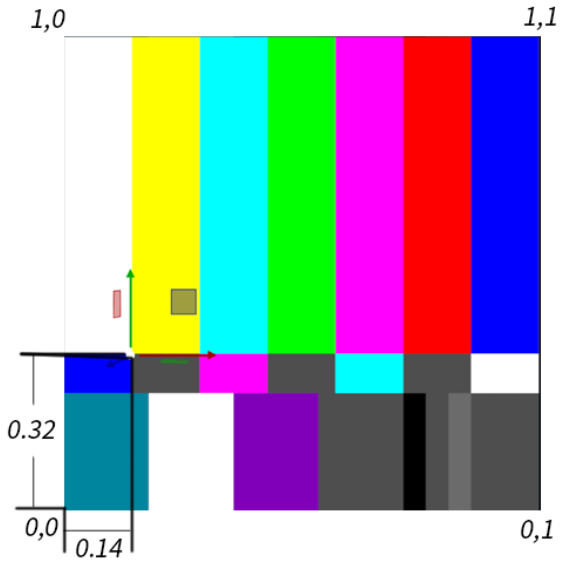


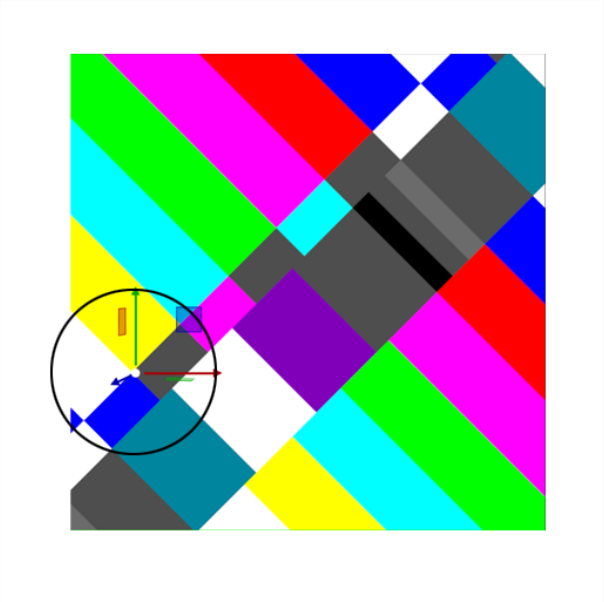
Tip: Turn down the roughness factor and increase the metallic factor a bit to achieve nice reflection effects.

5.7.11 Texture Manipulation

Non-UV mapped textures can be modified on PBR and Phong Materials. This can be **Rotation**, **Offset** and **Scaling**. Rotation and Scaling are based on a Pivot position.

Example: Rotation a texture on a Pivot point: The coordinates of a textures are 0,0 in the lower left (U) and 1,1 in the upper right (V), therefore if the pivot is set (as in the example below) to 0.14 in U and 0.32 in V, the rotation would look like this:



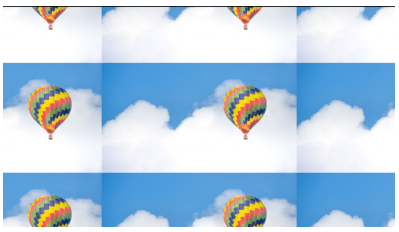


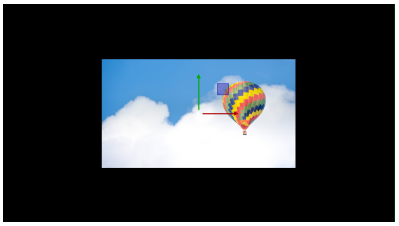
The same applies to the Scaling.

For more information, see Texture Generation.

5.7.12 Clamping

Textures can be set to clamp to a border color. This means instead of repeating the last pixels, they can be overridden by a border color. This can be set by using the Wrapping mode in U and/or V:

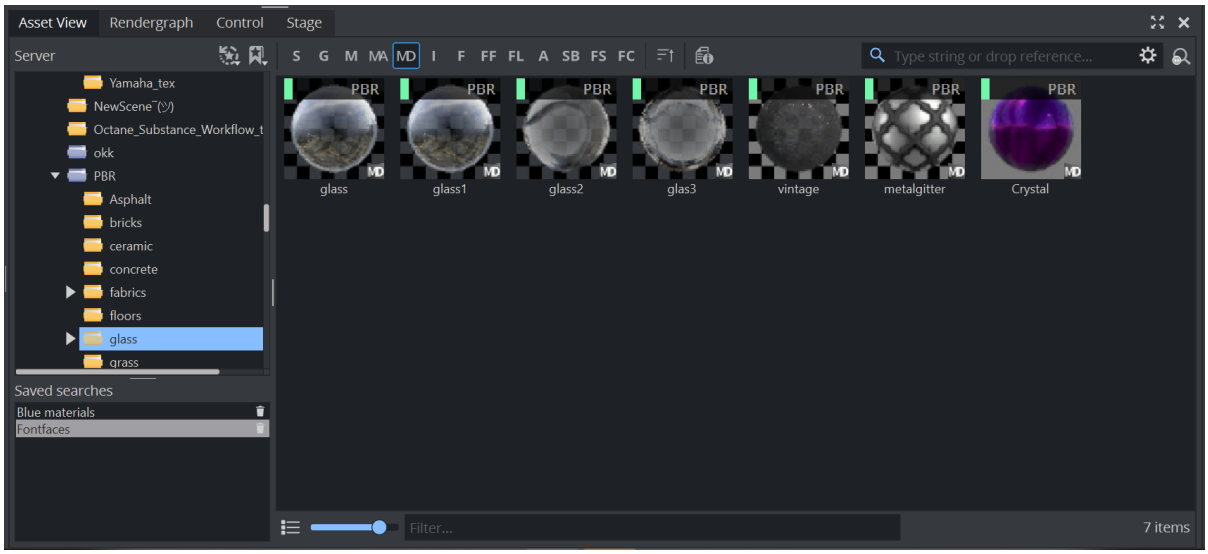
Repeat Mode	
Clamp Mode	

Clamp to Border	

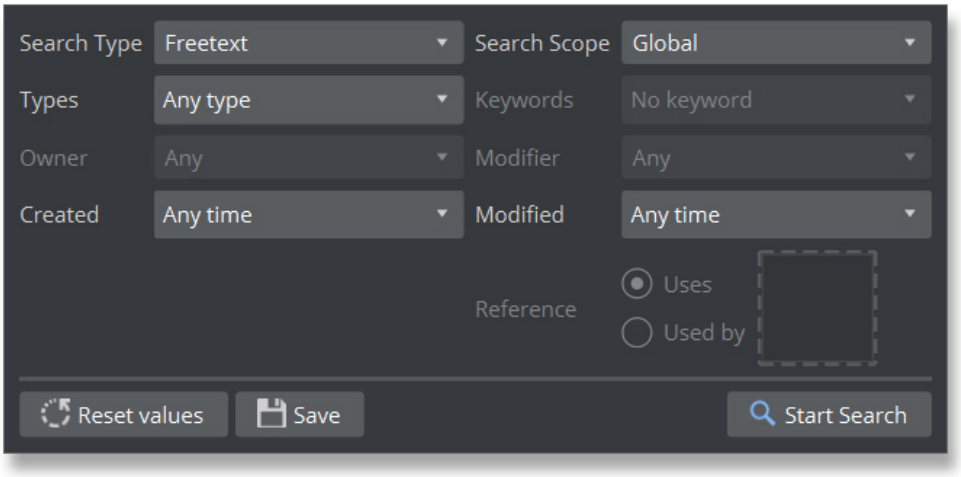
Information: The Alpha mode of the border currently only works for RGBA images.

5.7.13 Working with Material Definitions on Graphic Hub

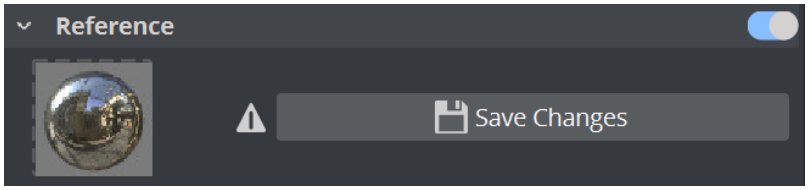
PBR Materials can be stored as *Material Definitions* on Graphic Hub. This means you can build your own library of pre-defined PBR Materials on your Graphic Hub. To store PBR Materials as Material Definitions, drag and drop them from the Scene Tree to your Server view and give them a proper name. PBR materials are labeled with *PBR* on top:



You can also search which texture you have used within this Material by using the search widget in the Asset View. Make sure you select *used by*:



Working with Referenced Materials



When using PBR Materials from Graphic Hub, they are linked into the scene, meaning if you modify them inside the scene, these changes are lost. You can break the connection between PBR materials and Graphic Hub by turning off the reference switch. This means the material becomes part of the scene.

If you modify a referenced material and want to keep these changes, you have to Save them to Graphic Hub (**Save Changes**).

> **Note:** Be careful if you use these Material Definitions in other scenes and modify them, this also modifies the reference and affects all referenced scenes.

See Also

- Environment Maps
- Texture Generation

Copyright © 2026 Vizrt

5.7.14 Environment Maps

Image-based lighting is a technique which illuminates a scene based on lighting information stored in an image. This image is usually a 360 degree capture of an environment, hence the name *Environment Map*. Environment maps are usually provided as HDRIs in different layouts and file formats, but need additional treatment to work properly in combination with a PBR Material.

Image Format

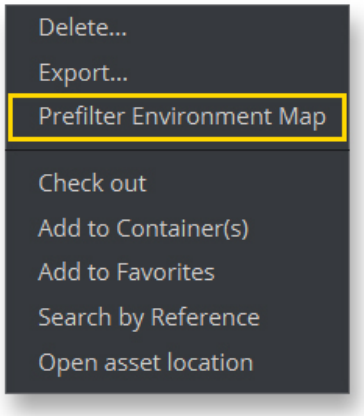
Images to be used as environment maps within Viz Engine must be 2D **panorama** images. The source image format should be **OpenEXR**, because it supports 16-bit or 32-bit float per channel, which is usually needed to properly capture the lighting conditions of a real environment. Do not use *.hdr* images, they are reserved for Classic usage.

Prefiltering

Prefiltering is a necessary step to prepare an imported image for rendering. It generates and extracts data necessary for both diffuse and specular lighting and stores it on the Graphic Hub within the newly generated image. If an environment map is used, that was not prefiltered before, parts of the lighting equation will be wrong and thus the whole environment lighting will behave incorrectly.

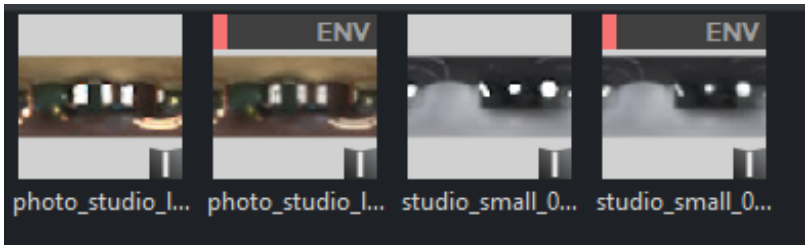
How to Prefilter an Environment Map

Right click a previously imported image in the Asset View, click *Prefilter Environment Map.* Use the suggested name or enter a new one and press **OK**. Depending on the size of the input environment map, this step may take a few seconds to complete.



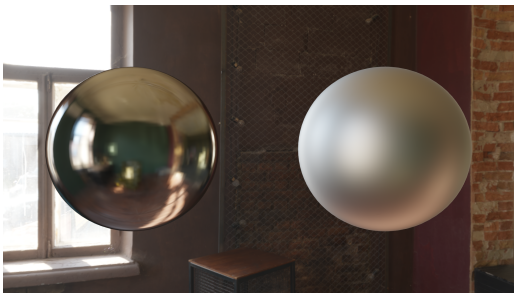
An environment map, which was properly pre-filtered is labelled *ENV*.

Copyright © 2026 Vizrt



Difference between Unfiltered and Filtered

PBR material parameters on the left and right are the same. Left uses a non pre-filtered image, right one uses a filtered one.

Image	Description
	*Metallic* is set to 1.0, *Roughness* is set to 0.5 for both materials. The Left side only shows a blockier representation of the original image (mip-map), whereas the right properly changes the overall look in reaction to roughness.
	*Metallic* is set to 0.0, *Roughness* is set to 0.0 for both materials. The Left side only shows mostly black, whereas the right properly uses the environment map for diffuse lighting.

Acknowledgements

All Environment Maps used are available at *https://hdri-haven.com*.

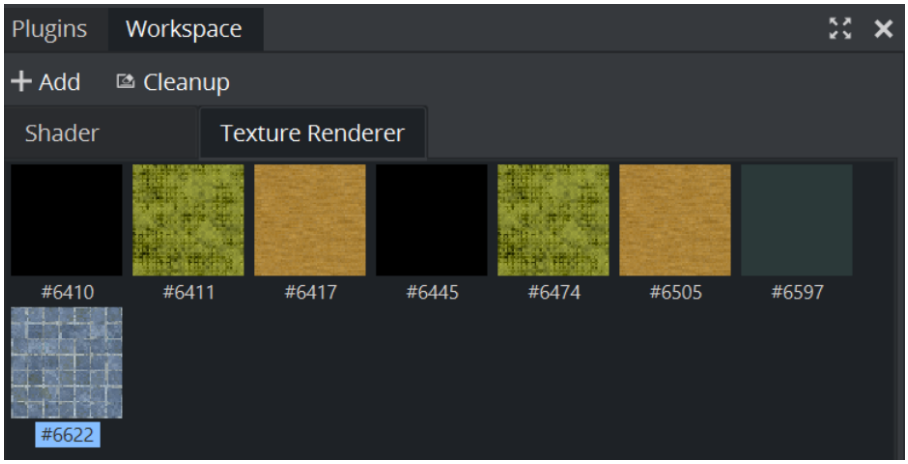
Copyright © 2026 Vizrt

5.7.15 Substance Materials

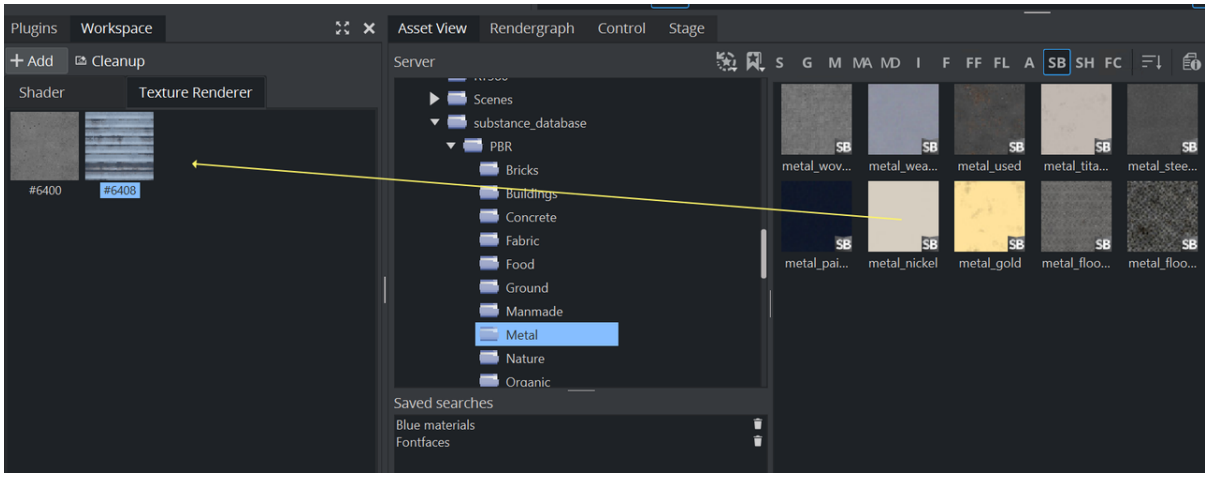
Adobe Substance Materials are the state of the art materials and are supported by using PBR Material Definitions and Texture Renderer. Substance materials can be imported directly into Graphic Hub or as part of the Cinema 4D Substance workflow.

> **Information:** Substance Materials in the Viz Engine Render Pipeline require a valid license (Substance FX Authoring).

To use Substances directly, drag and drop them on a container. These creates a Texture Renderer item, available in the Workspace dock:



To create them manually, drag a Substance Asset from Asset view to the Texture Renderer Workspace.



Copyright © 2026 Vizrt

To Modify Substance Settings

These special materials expose special settings, available if you double click on one of the texture renderers. This opens the substance editor.



The screen above is just an example and varies from Substance to Substance. To go back to the Texture Renderer Overview, use the arrow on the top: ←

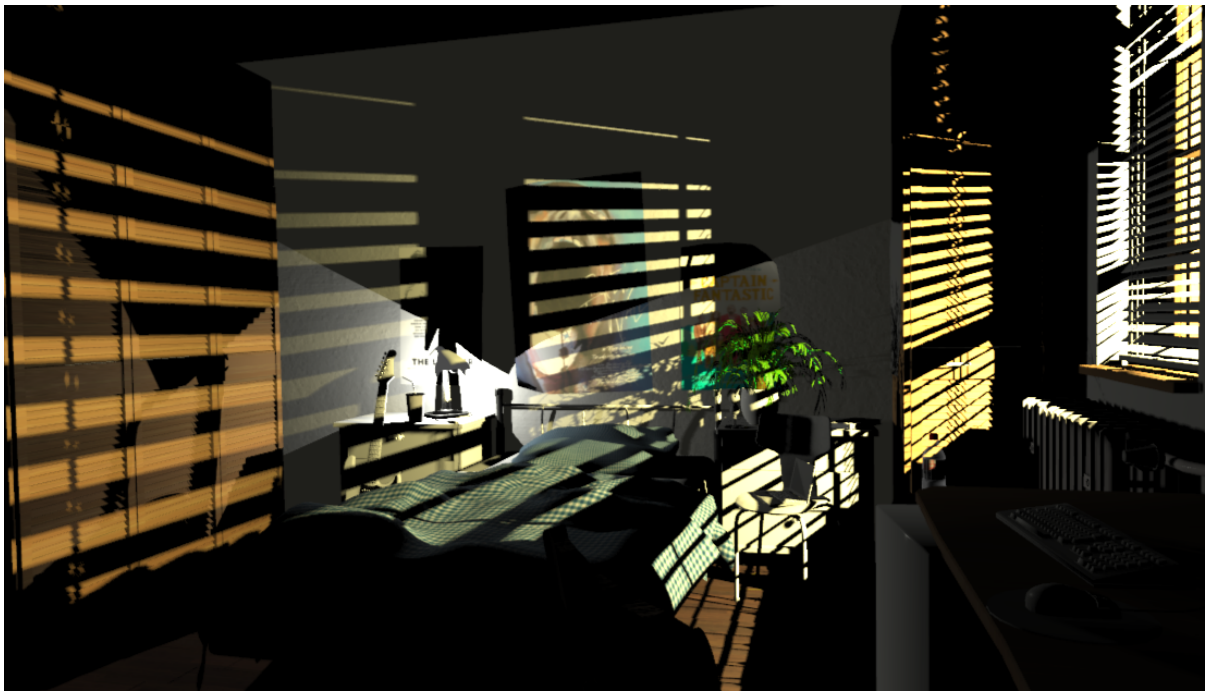
Note: Texture Renderer entries can not be dragged to the Trashbin.

See Also

- Working with Various Materials
- Substance 3D Designer
- Substance 3D Painter

Copyright © 2026 Vizrt

5.8 Light and Shadows



This section contains information on the following topics:

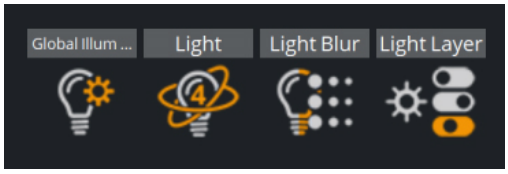
- Lighting in the Viz Engine Render Pipeline
    - Light Layers
- Working with Environment Maps
- Directional Lights
- Area Lights
- Point Lights
- Spot Lights
- Working with IES Files
- Volumes
- Shadows
    - Shadow Lights
- Advanced Light Falloff (Attenuation)
    - Examples
- Troubleshooting Shadows

5.8.1 Lighting in the Viz Engine Render Pipeline

Lights in the Viz Engine Render Pipeline are different than we've been using them in the Classic Render Pipeline. The scene-based eight lights are gone now and no longer supported. To use lights with the Viz Engine Render Pipeline,

Copyright © 2026 Vizrt

you need to place a light object onto a container. Light properties like position, rotation or size can be controlled by simple changing the containers transformation



Lighting is a basic factor when using a physical-based rendering approach. Environment Maps, PBR Materials and Light sources are essential for a photo realistic rendering.

> **Tip:** For a quick overview (for example, if you've freshly imported a new geometry), you can use the default light settings next to the scene editor. This applies default lighting without the need to set everything up first. Be sure to turn it off again when you start setting up your lighting!

The Viz Engine Render Pipeline supports eight lights per type per geometry. If you need more lights, lights can be split into light layers.

Light Layers

Every light has a light layer property, the Light Layer plug-in can be applied to a certain group of containers. The light with its corresponding bit-wise layer number is only applied to the light layer group.

5.8.2 Working with Environment Maps

The Viz Engine Render Pipeline follows the concept of realistic lighting environments for PBR materials by using environment maps. Every Viz Engine scene should have a High Dynamic Range Image (HDRI) spherical environment map applied. Such a map represents a wide-view of our surrounding area. If our graphic is supposed to be used as an AR element inside a TV Studio, the environment map should represent your actual studio to achieve the same lighting conditions as in the real world.

In terms of a proper HDRI environment map, it should be:

- A Map in latitude/longitude format.
- HDR 16-bit color depth.
- Prefiltered.

HDRI refers to the fact that the environment contains light information that extends beyond the usual color range from `0` to `255`. This comes into play when we talk about correct Tonemapping. Details in bright and dark areas can be controlled for realistic light and reflection effects.

Copyright © 2026 Vizrt

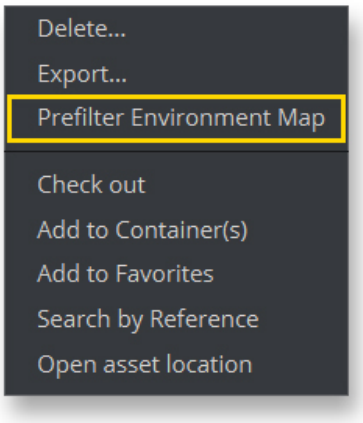
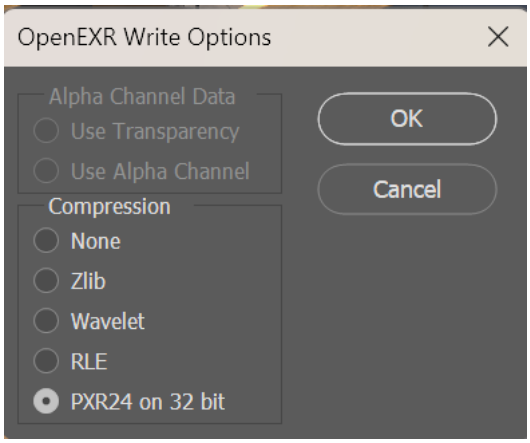
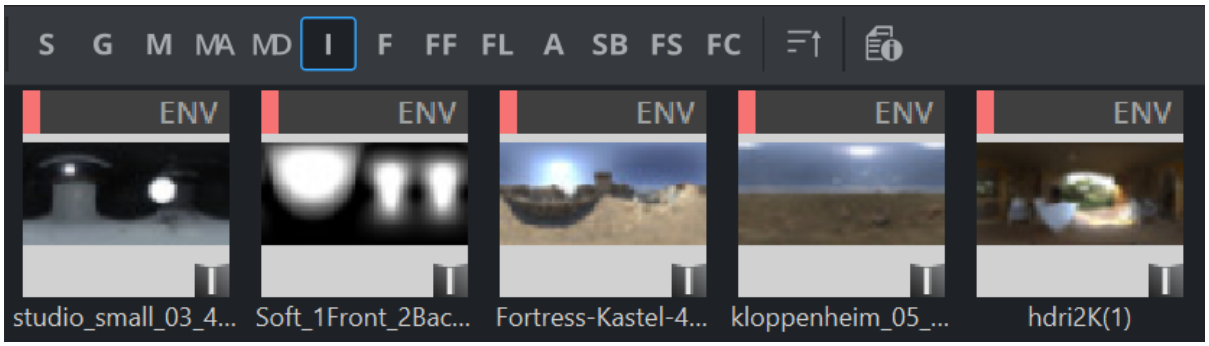


Image maps are imported as *.exr* files. If you are using Adobe Photoshop, make sure the *PXR24 on 32-bit* Compression is used.



Once you have imported these files, right click on them and select **Prefilter Environment Map** and save it as a new map. These maps are required for correct rendering and need to be applied in the Scene Settings **Asset Placement**. All pixels in the map are treated as light source. Environment Maps can consume quite a lot of memory inside your Graphic Hub.

Environment Maps in Graphic Hub are labeled with *ENV*:

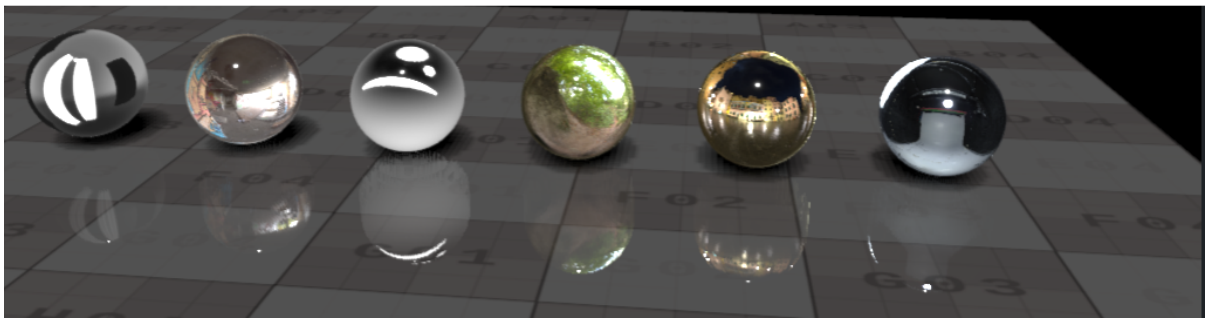


Copyright © 2026 Vizrt

Only pre-filtered Environment maps can be used, any other image will show a warning message.

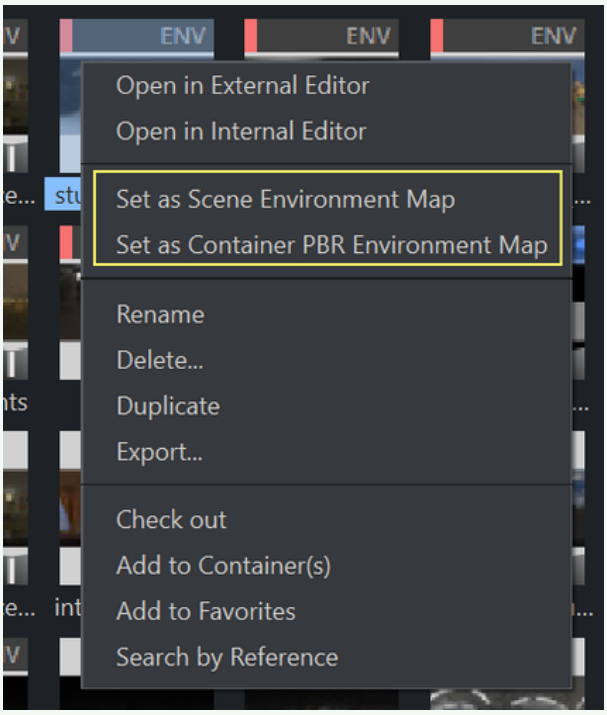
The environment can be rotated to bring lights into the correct position. In addition, the intensity can also be varied. The following illustrates some images with different environment maps applied:





Credits: Environment Maps taken from https://hdrihaven.com/

Tip: Environment Maps can be applied quickly to either the Scene or to Container(s) by using the context menu on Pre-Filtered Environment Maps:

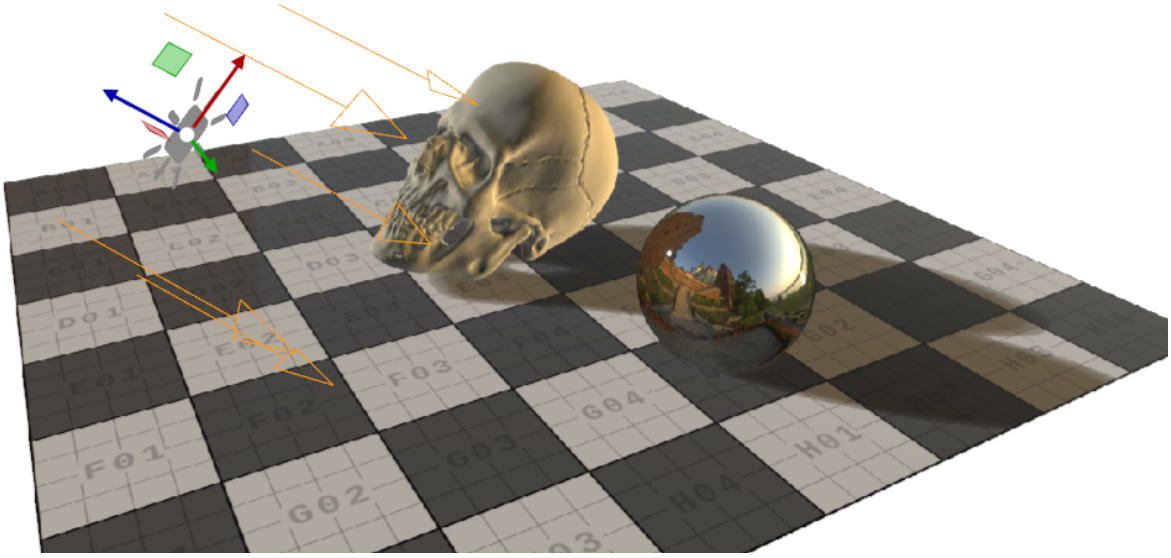


Copyright © 2026 Vizrt

5.8.3 Directional Lights

Directional lights (also known as Distant lights) are lights that shine into a parallel direction. All rays are present everywhere in your scene as there is no falloff: The intensity stays the same. In nature, real directional lights do not exist, in computer graphics they are mainly used to simulate strong light sources far away emitting all objects. A good reference example would be the sun. That's why you would usually only have one distant light in your scene, responsible for a global, scene-wide light.

Directional lights are only controlled by an object's container rotation, position or scaling do not have any influence. They have a specular and a diffuse intensity.



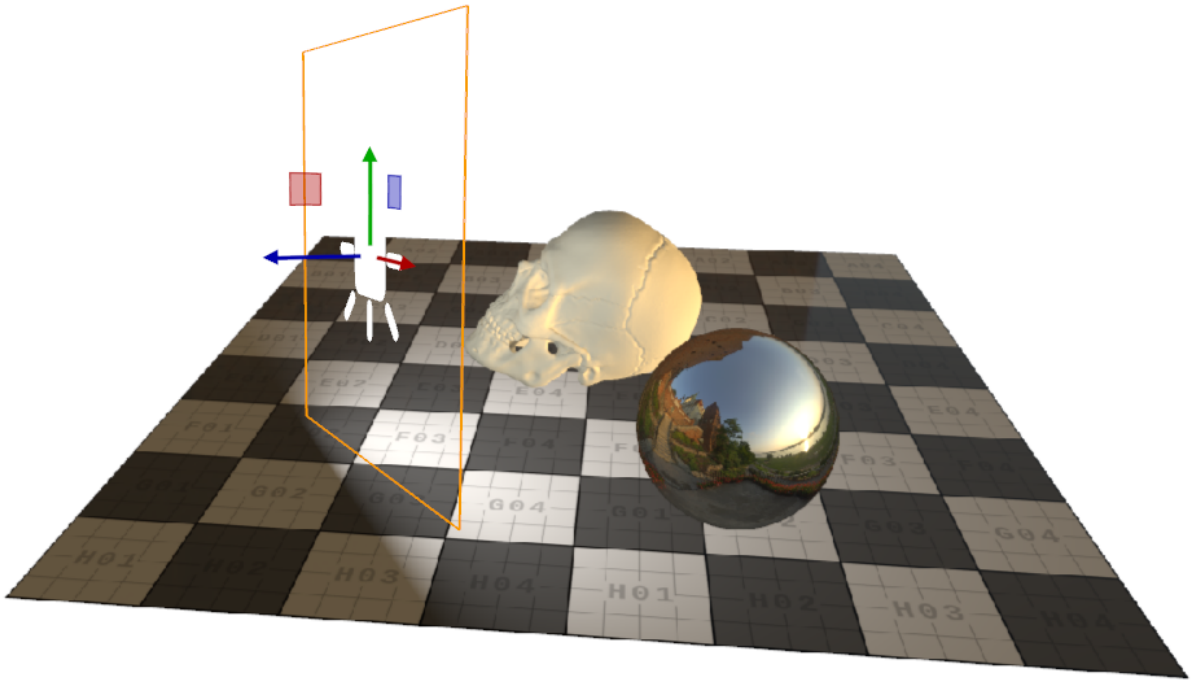
In addition, the number of light cascades used for directional lighting can be set in the light properties. The higher the cascade setting, the better the quality of shadows.

5.8.4 Area Lights

An area light is defined by a square area and sends out light equally into the square's main direction without any falloff. They can be used to simulate flat light panels or rays from a portal light. Area lights are the only lights that do not emit any specular light, nor do they cast shadows!

They are defined by their position, rotation and scaling of the container's transformation.

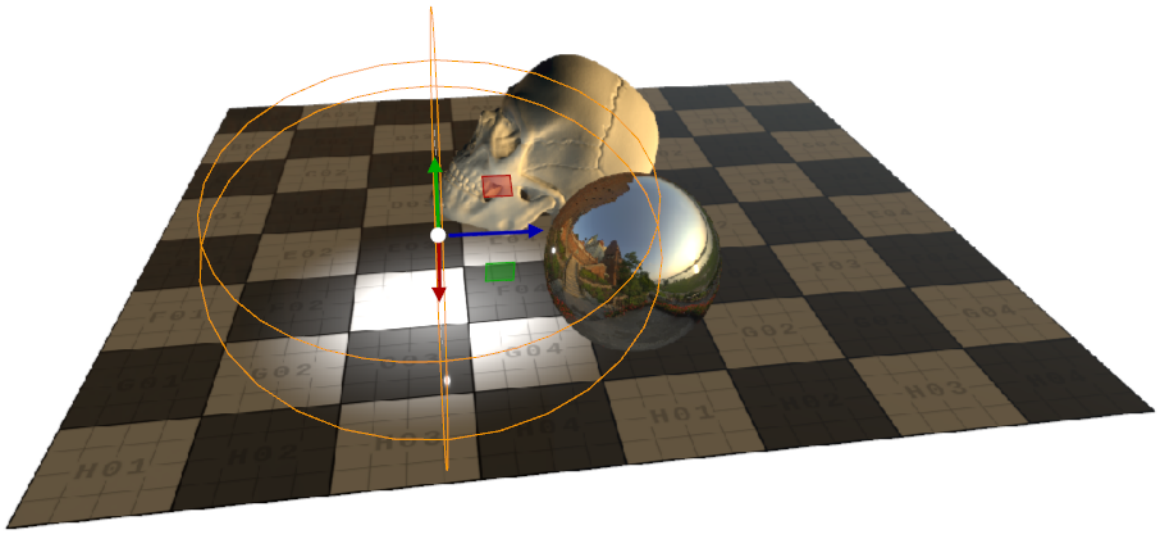
Copyright © 2026 Vizrt



5.8.5 Point Lights

Light emitted from a single point, into all directions. The typical usage for point lights is for example a classical light bulb.

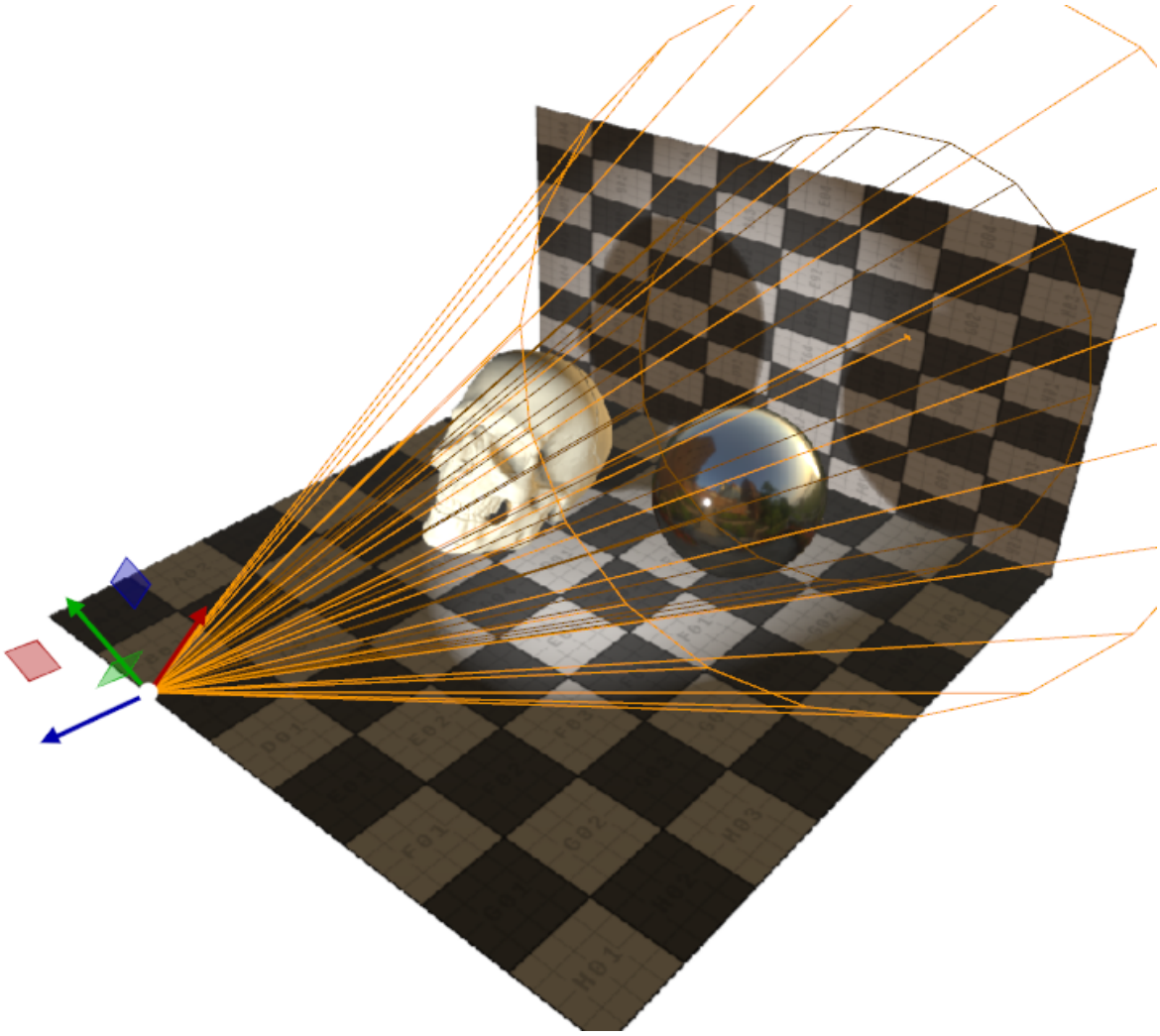
In comparison to the directional light, they have a falloff and are not influenced by any direction. The light casts equally into all directions, but the position is important; while for directional lights, the position doesn't matter.



Copyright © 2026 Vizrt

5.8.6 Spot Lights

Spot lights project light rays onto a geometry using a cone of light. They are very similar to point lights, except that instead of emitting into all directions, they are restricted to a cone. Spot Lights can be used, for example as a flashlight of a lighthouse.

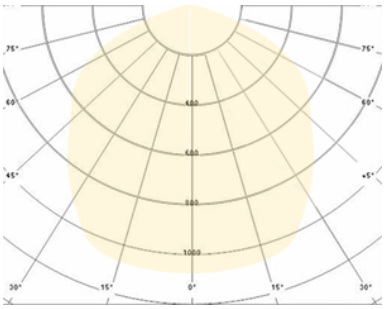
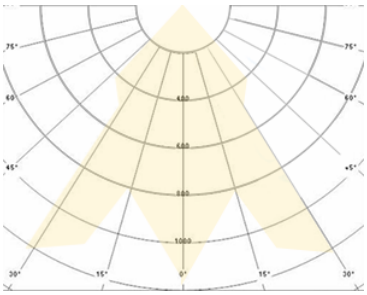


Information: Be careful when using spotlights and shadows! The cascading shadows need to be calculated six times per light, shadows on spotlights decrease your performance.

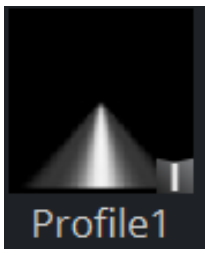
5.8.7 Working with IES Files

Viz Engine supports IES files, which stands for Illuminating Engineering Society. IES files describe how a real light is being distributed from a Light Source. By using an IES profile, the the distribution of light emitted from shapes a more realistic light fixture. Many vendors supply a IES profile for their lights.

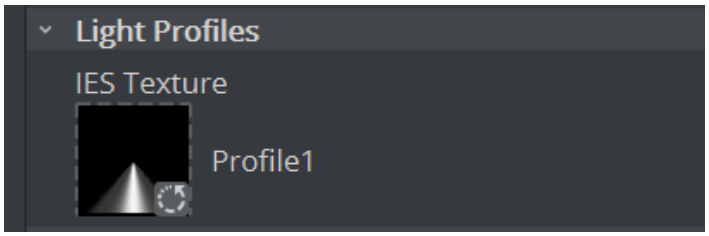
Copyright © 2026 Vizrt

Basic Light	Light with IES profile
	
	

Import *.ies* files as images to Graphic Hub. Viz Engine creates a preview for the image based on the profile information contained in the IES file:

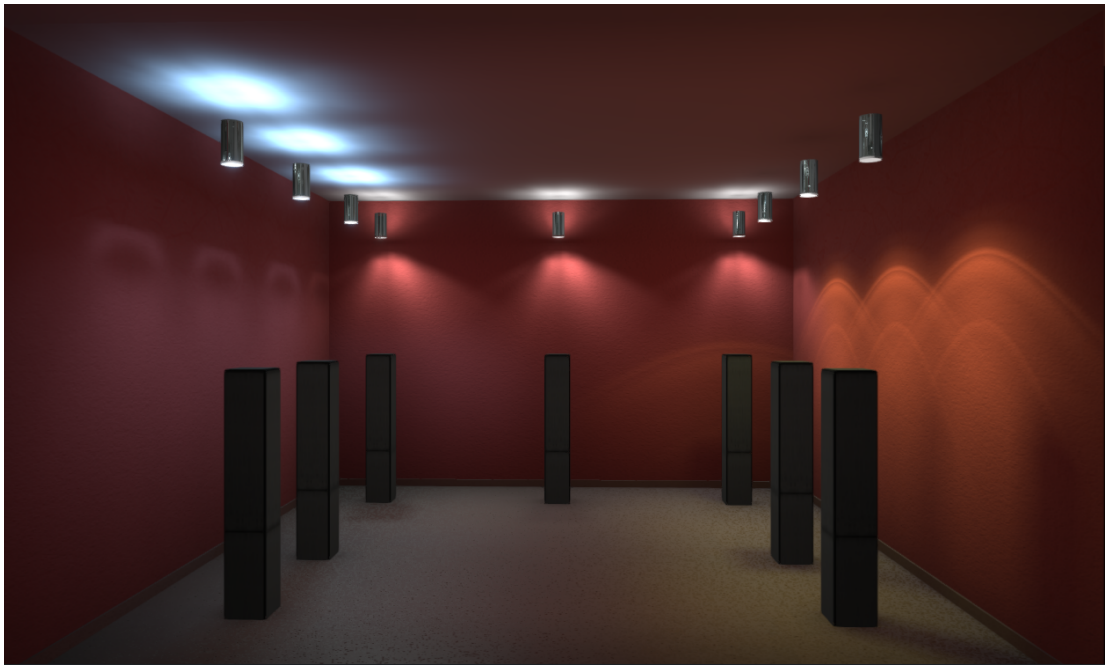


Apply the IES profile to the light by dragging it onto the Light Profile setting in Point or Spot lights:



Using IES light profiles adds more realistic lighting to your scenes:

Copyright © 2026 Vizrt



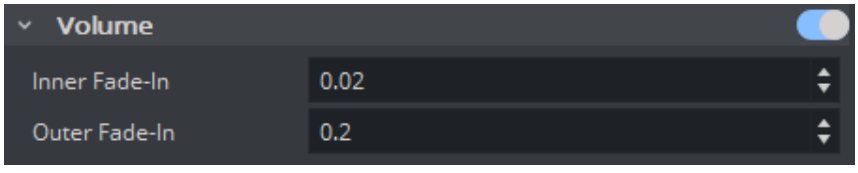
Information: IES Files are supported on Point and Spotlights only.

Information: If an IES profile is applied to a Spotlight, its cones mask the IES profile effect. The maximum outer cone angle for the Spotlight is 180 degrees, so any IES information beyond this point is lost.

5.8.8 Volumes

When the fog is active, the light could cast an illuminated volume due to the light scattering on the fog particles. The occlusion made by an object in the scene could also produce a light shaft if shadow is activated at the same time.

The following properties customize the light volume behavior:

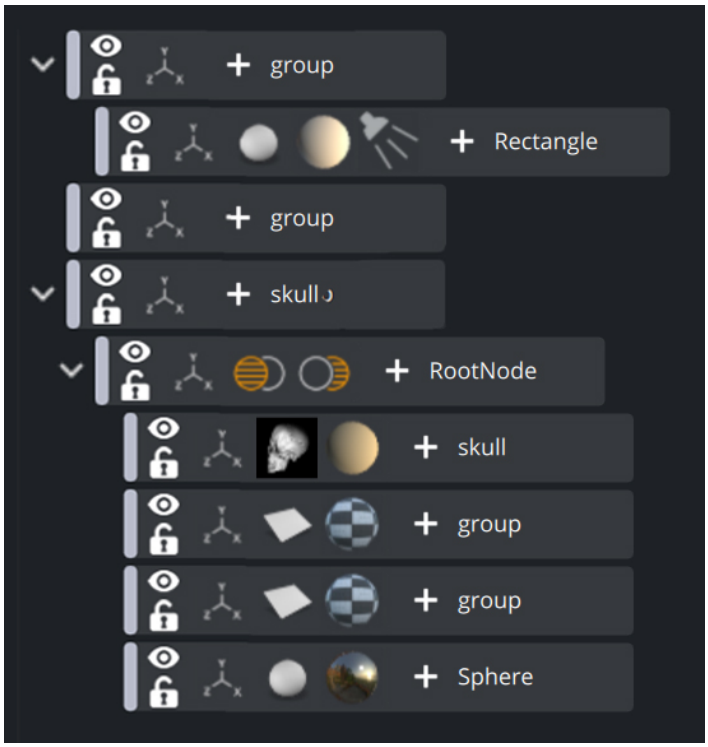


- **Enable:** Allow the light to cast the volume; namely, to allow scattering on the fog particles if such effect and Post-Processing in Scene Editor are enabled.
- **Inner Fade-In:** Set the starting volume fade-in distance from the light source, as a portion of the light radius. Only available for Spot and Area Lights.
- **Outer Fade-In:** Set the ending volume fade-in distance from the light source, as a portion of the light radius. Only available for Spot and Area Lights.

Copyright © 2026 Vizrt

Tip: In contrast to the fade-in, falloff (attenuation) controls fade-out behavior for the selected light types. Increasing Linear (L) and/or Constant (C) falloff factors could emphasize the volume by slowing down the fade-out.

5.8.9 Shadows



Lights cast shadows, the approach how shadows are created has completely changed and been optimized.

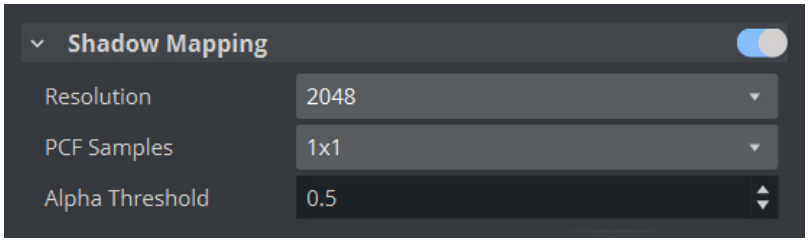
To use shadows, the following preconditions need to be fulfilled:

- Light sources supporting shadows (Directional, Point, Spot Lights) and shadow casting enabled on the light.
- Shadows need to be turned on in **Scene Settings > Render Settings**. This is a global setting for the whole scene.
- Every Geometry that needs to receive and cast shadows must have a caster/receiver plug-in applied.

The shadow rendering algorithm used is Cascaded Shadow Mapping, which splits a light frustum into multiple ones, where each frustum gets its own shadow mapping texture.

The sequences' shadow mapping panel supports multiple configurations, which are scene global options for the shadow rendering:

Copyright © 2026 Vizrt



- **Enable:** Turns shadow mapping on and off for the scene.
- **Resolution:** Sets resolution of a single shadow mapping texture.
- **PCF Samples:** Determines the number of samples used for softer shadows. The higher the softer the shadow, but the higher the performance impact.
- **Alpha Threshold:** Sets the threshold for alpha masked shadows, when the alpha is lower than the given threshold, a pixel is discarded.

Each light has additional options that influence the way the shadow textures are being generated:



- **Enable Shadows:** Enables shadow casting of this light.
- **Offset Factor / Units:** Influences the way depth values of a shadow map are being written. Higher values help reducing shadow acne.
- **Cascades:** Sets number of textures this light uses for shadow mapping.
- **Split Lambda:** Influences the way the frustum splits are calculated. It's a factor to blend between logarithmic and uniform distribution of the splits.

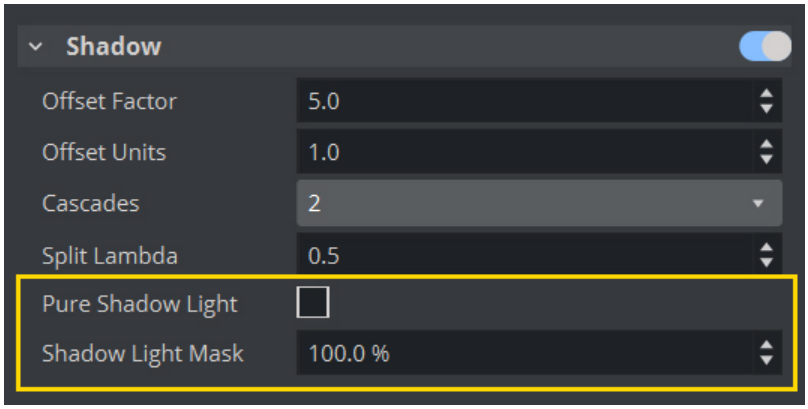
The quality of your shadow is also influenced by the Camera Near and Far value. The larger this distance, the less quality you'll have. Therefore, it is recommended to set the values to a real world environment. The units are in centimeters, a *Far* value of 20 meters does not make any sense if your objects has only a few meters of depth.

Shadow Lights

Although a contradiction, lights can also be turned into shadow lights. This represents a light source that emits no light, but still casts a shadow while the casted shadow is emphasized so there is no illumination on it.

There are two degrees of controlling a shadow light: A pure shadow light or only emphasizing the casted shadow.

Copyright © 2026 Vizrt

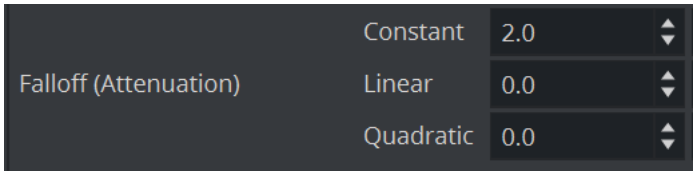


Two new parameters in the light plug-in are introduced.

- **Pure Shadow Light:** If turned on, the light source becomes a pure shadow light source that only casts shadows but does not illuminates (implying shadow light mask = 0%).
- **Shadow Light Mask:** Determines how much light from other sources illuminates the shadow area. The light still illuminates other area as normal. The shadow can still be illuminated by other lights if the shadow itself is transparent (Castor's Alpha < 100%).

5.8.10 Advanced Light Falloff (Attenuation)

Viz Engine 5.4 introduces an alternative falloff option for Point- and Spotlights. It offers the (most common) options:

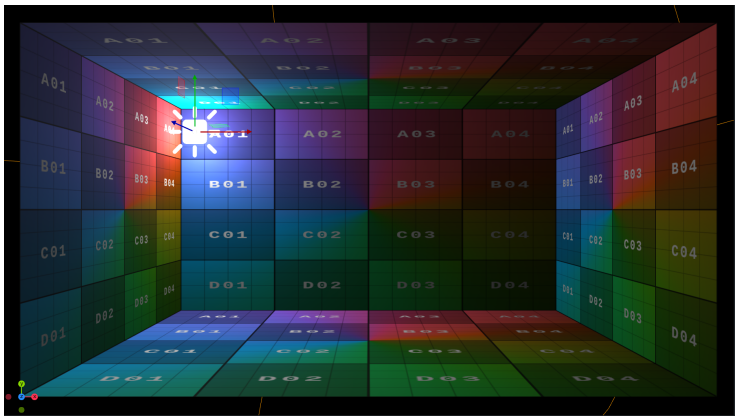
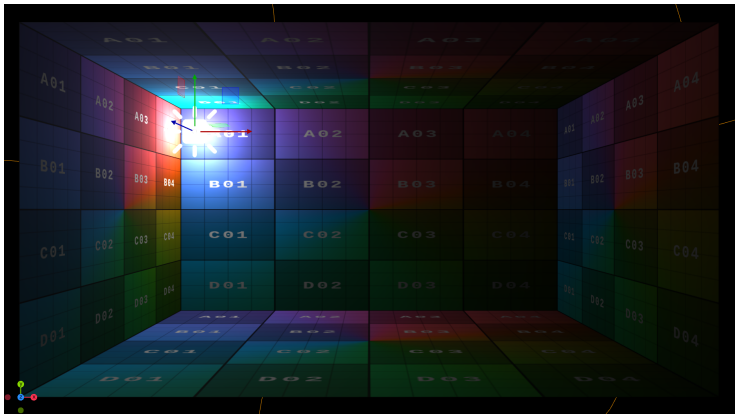


- **Quadratic (Q):** Quadratic light falloff, leaving strength unmodified if smooth is set to `0.0` and corresponds to reality.
- **Linear (L):** Linear light falloff, giving a slower decrease in intensity over distance.
- **Constant (C):** Constant light falloff, where the distance to the light has no influence on its intensity.

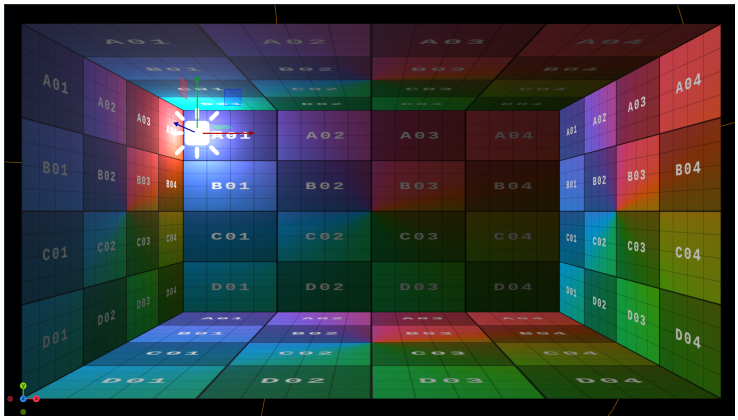
The constant-linear-quadratic falloff model can be mixed together. These three options construct the corresponding ratio used to calculate the total falloff.

Copyright © 2026 Vizrt

Examples

Constant Falloff 100	
Linear Falloff 100	
Quadratic Falloff 100	

Copyright © 2026 Vizrt

Constant Falloff 35 Linear Falloff 35 Quadratic Falloff 35	

5.8.11 Troubleshooting Shadows

What is wrong if I do not see my shadows? It could be:

- **The Environment is too Shiny**

Try to use another Environment Map with less light or reduce the Environment Intensity.

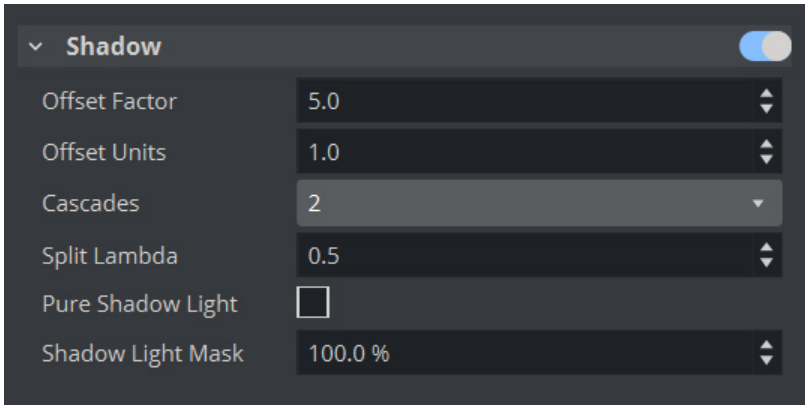
- **Default Lighting is Enabled**

Make sure that you do not have the default lighting button ( ) enabled. This is for a quick check only (for example, if your geometry has been imported correctly). This enables a default light.

- **Rendering is Occurring on Backface**

Viz Engine does not render shadows on the backface, for instance if a rectangle is flipped by -90° or surfaces are flipped inside out. If Rendering is needed, than you have to add an Expert plugin with Backface turned on.

A global backface rendering can be turned on in the scene settings, if your shadow appears, you are rendering on the wrong side of your geometry.



- **Point Light Radius is too Small**

Point Lights require quite a large radius to take effect. Try to increase the value for the Radius.

- **Scene Dimensions are too big**

Copyright © 2026 Vizrt

Shadow Maps are calculated on the dimension of your scene dimensions. If they are too big, shadows do not appear. Try to split up the scene in smaller parts, and only apply the Shadow Mapper/Caster on those areas where shadows are required. The same applies to far/near distance of the camera. If your camera is looking too far, the shadow maps get larger and loose on quality.

- **A Quality Issue**
  - The quality of Shadows depends on various factors. Verify that your camera far and near values match your scene dimensions. A far value of `20000` means that your scene is about 20 meters in depth, usually this can be set closer.
  - Try to use fewer cascades, as more cascades cause less performance.
  - Shadow Map Resolutions. Keep in mind that a 4K shadow map for Point lights needs to be rendered six times. A good compromise between cascades and lightmap size needs to be found.

Note: Keep in mind that Area Lights do NOT cast shadows.

Copyright © 2026 Vizrt

5.8.12 Concepts of Lighting

A perfect lighting setup is crucial for any 3D scenery. A proper light gives an individual mood and atmosphere to your graphics. In real life, natural light already exists, like the sun or artificial lights in a studio. When it comes to integrate a 3D graphics like an AR object, your environment map is not sufficient enough to reflect the real world situation. You should try to simulate the real world lighting situation (like the studio lights).

There are several methods to compose your lighting scenery. One of the basic setups is the so-called *3-Point Light*:

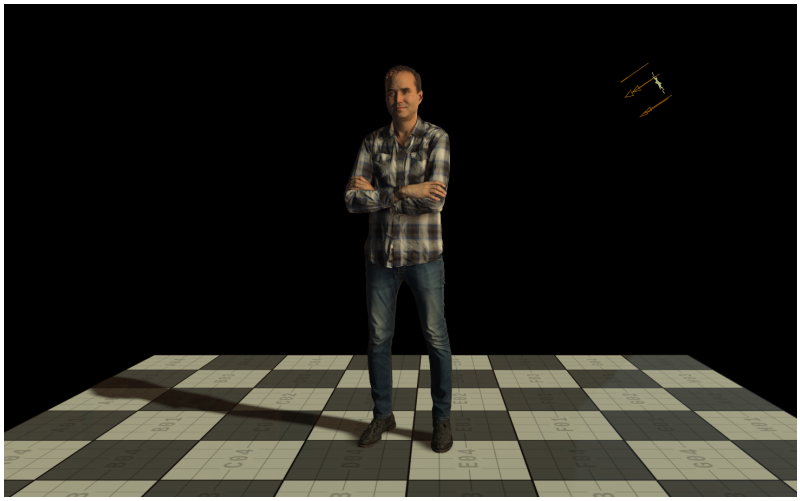
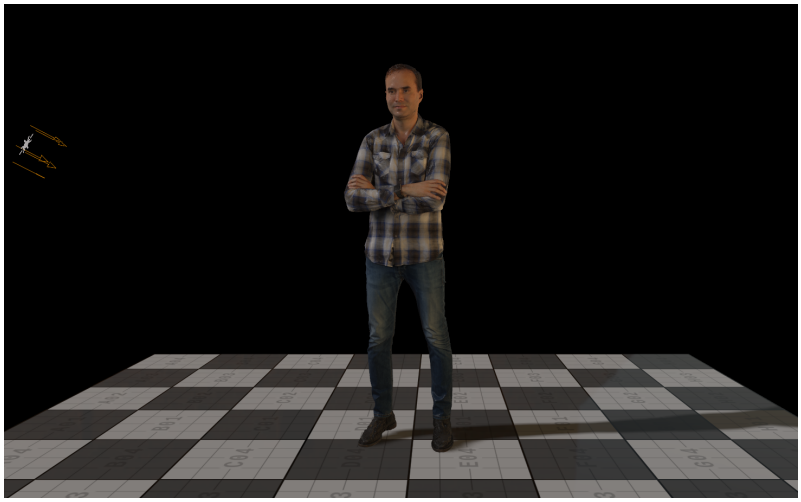
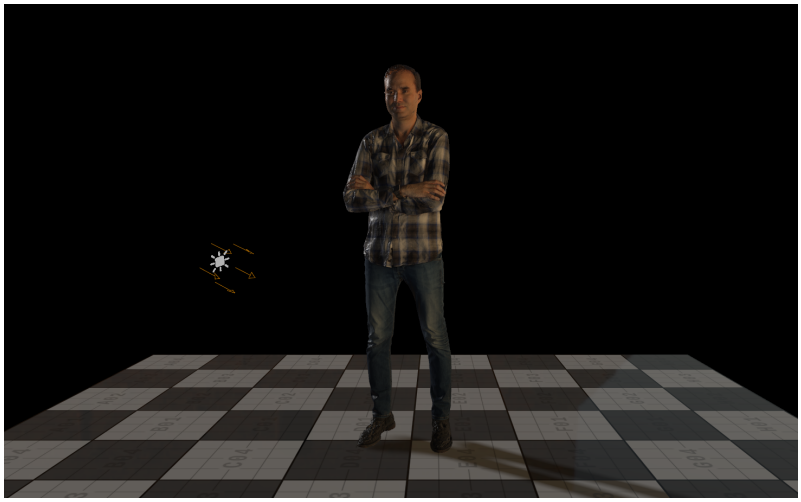
3-Point Light

We use three different light sources positioned as a triangle. All light sources can be chosen; however, a directional light or a spotlight is your best choice:

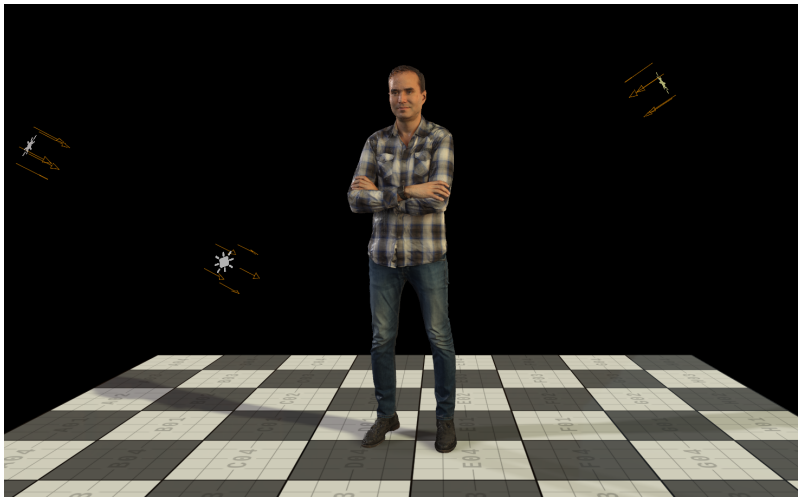
- **Key Light:** Positioned next to your motive and set a bit above your target, this is your main light. The Key Light defines the angle of your lighting. Usually, it is positioned next to the camera at a 45° angle and set a bit above to simulate lighting similar to the sun.
- **Fill Light:** Used to fill areas darkened by the Key Light and positioned opposite the Key Light. A soft light should be used, weaker than the Key Light. Typically it is 50% the intensity of the Key Light.
- **Back Light:** Positioned directly behind your target and opposite of your target. It is positioned above the target and lights the object at an angle between 15° to 45°. It is used to create depth, contrast and exposes the edges of your target.



Copyright © 2026 Vizrt

Key Light	
Fill Light	
Back Light	

Copyright © 2026 Vizrt

3 Point Light Final setup	

Besides the 3-Point light, you can use various other lighting techniques to bring more emotions into your set.

Flat Light

A technique to create basic lighting. The light source is positioned at the same location as your camera as if mounted on the camera itself. There are almost no shadows on your target and no visual depth. The diffuse intensity can be used to create either a hard or a soft light.

Flat light, low intensity (soft light)	Flat light, higher intensity (hard light)
	

Copyright © 2026 Vizrt

Butterfly/Paramount Light

	Mainly used in the 1950s in Hollywood productions. Like the Flat Light setup, the light comes from the front, but positioned slightly higher than the Flat Light and pointed at the target. When lighting people, the nose creates a kind of a butterfly shadow. Be sure not to put the light too high, otherwise it would generate unwanted shadows.

Split Light

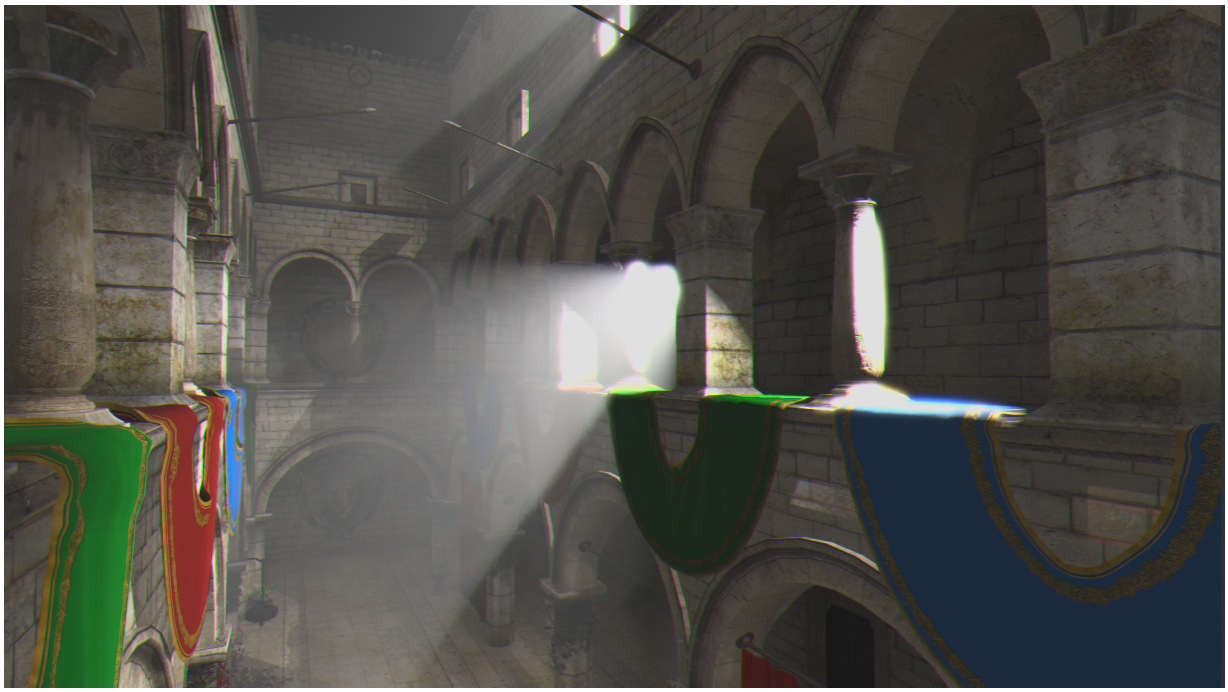
	Used in portrait lighting to split the face into two parts. A hard light is positioned on the left or right side, lighting only one half of the object. When used in photography, typically a reflector would be positioned opposite the light source. When used with the 3D environment of Viz Engine, an additional light source needs to placed instead of the reflector.

Broad Light

	Like the Split Light, the light source is placed on the side of the object, typically the one that is looking towards the camera and represents a softer variant of the Split Light. The smaller side of the object is hidden in the shadow.

Copyright © 2026 Vizrt

5.8.13 Volumetric Lighting



Volumetric Lighting is the new feature introduced in Viz Engine 5.5 onwards. The light is now capable of casting its volume, making such holistic effect and increasing scene realism. This relies on the fog effect, because the rendered volume follows the principle that light scatters on the clouds of particles in the air, and reaches the eye after uncountable bounces. When there is a shadow castor occluding the light volume, it produces the well-known *Light Shaft* or *God Rays* effect.

To have volumetric lighting deployed in the scene, the following components are required:

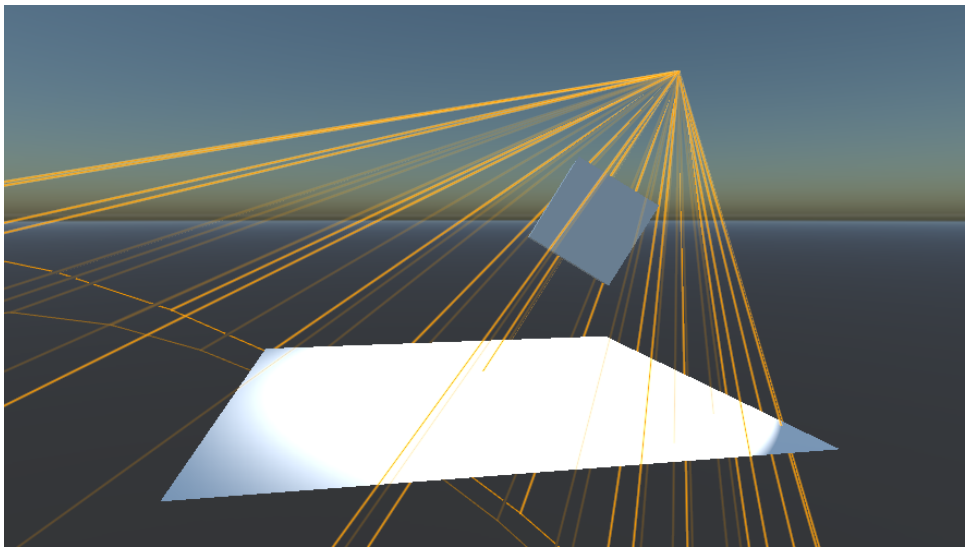
- **Active fog effect:** This could be done by going to **Properties panel > Scene > Rendering > Post Processing > Fog** and enabling it. To examine the result in Scene Editor, make sure that Post Processing effects are activated. See Fog to customize the fog effect.
- **Volume-casting lights:** In Light properties, go to Volume and enable it. See Light and Shadows to customize volume casting behavior.

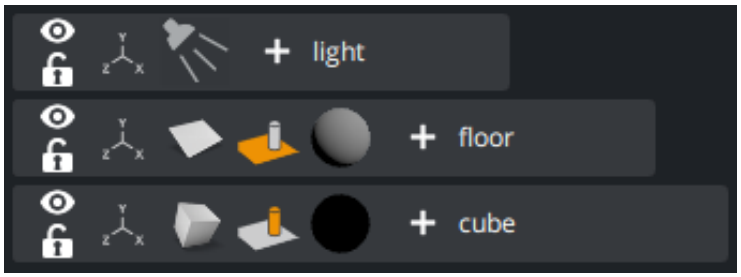
Supported Light Types

- Directional Lights
- Spot Lights
- Point Lights
- Area Lights (non-LTC)

Copyright © 2026 Vizrt

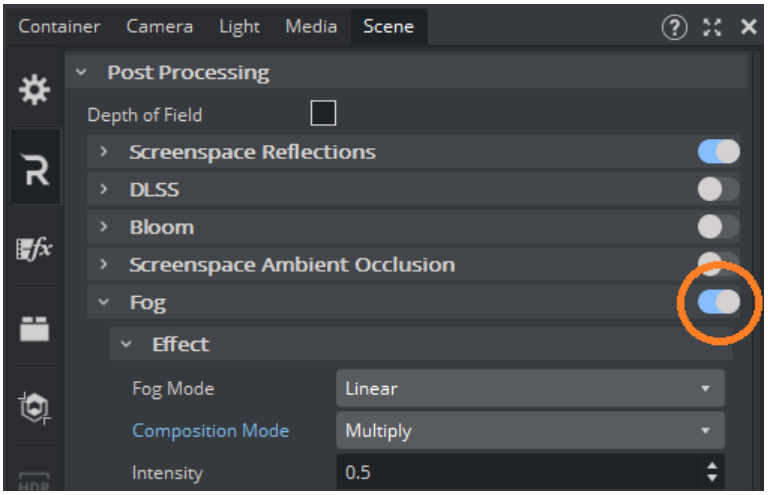
Example





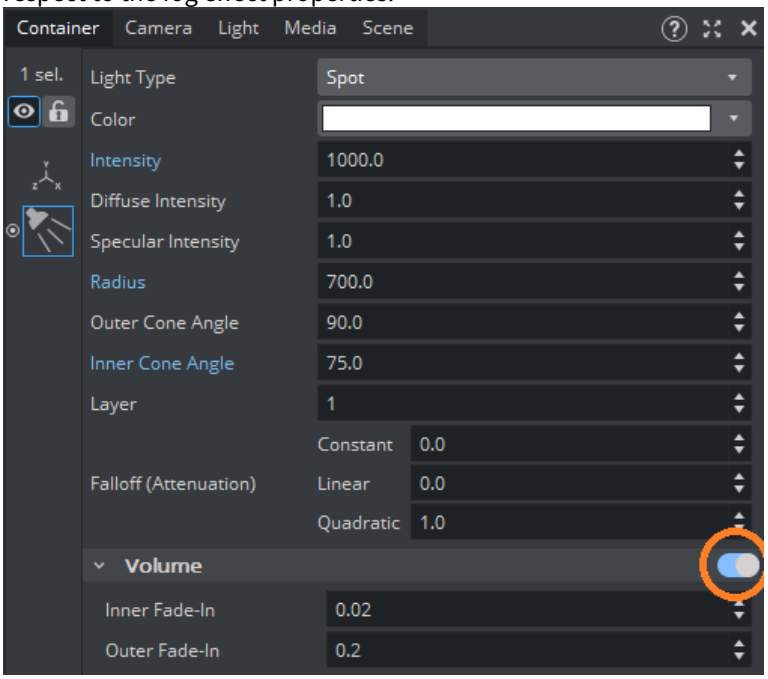
1. Start with the following scene setup:
    a. A light to cast a volume.
    b. A floor to receive shadow with an arbitrary material.
    c. A geometry to cast shadow and occlude some part of the volume, with an arbitrary material as well.
2. Go to **Properties panel > Scene > Rendering > Post Processing > Fog** and enable the fog effect.

Copyright © 2026 Vizrt



Tip: Setting Fog's Composition Mode to *Multiply* is recommended for volumetric lighting.

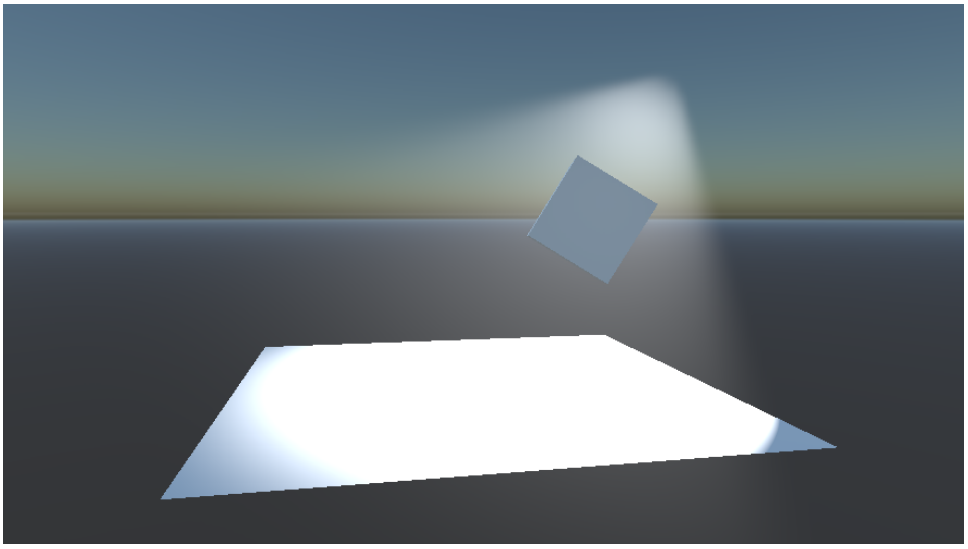
3. In the light properties, find the Volume group and enable it. This allows the light to cast its volume with respect to the fog effect properties.



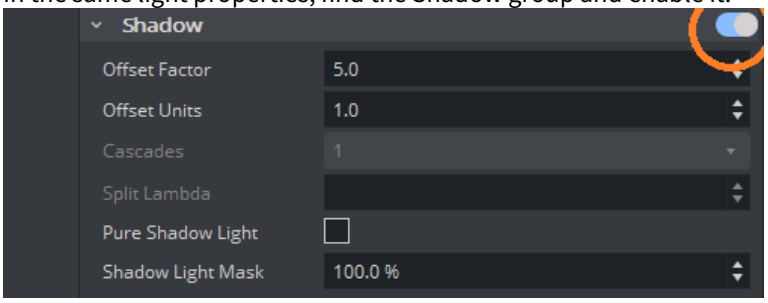
Tip: Apart from customizing Fog effect, the volume intensity of a particular light mainly depends on Light's Intensity. In addition, Light's Falloff (Attenuation) could slow down the volume fade-out, giving the volume a more solid appearance.

4. Now the light volume appears in the scene when Post Processing effects are activated for Scene Editor.

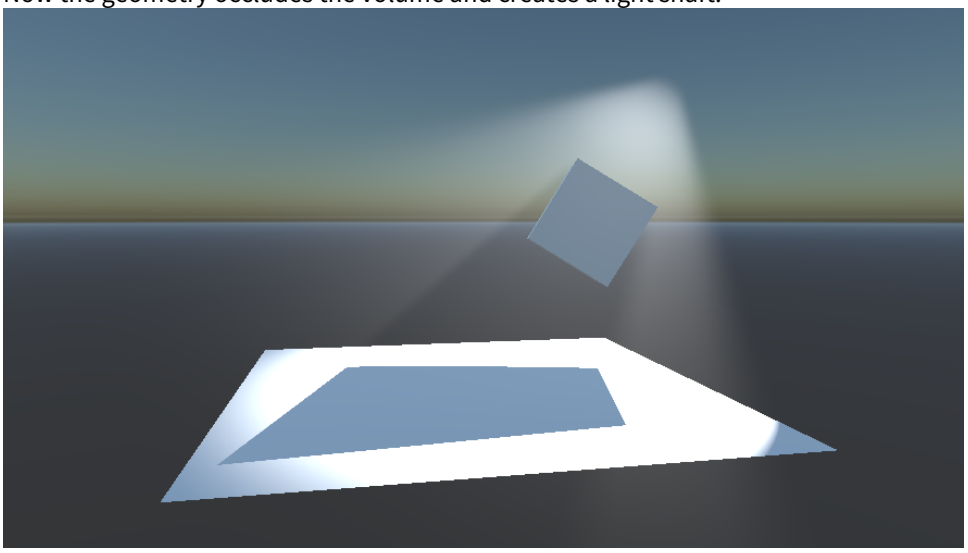
Copyright © 2026 Vizrt



5. In the same light properties, find the Shadow group and enable it.



6. Now the geometry occludes the volume and creates a light shaft.



Copyright © 2026 Vizrt

5.9 Global Illumination

Global Illumination is a rendering technique that attempts to mimic real-world lighting behavior, taking into account that objects are illuminated by direct light (*direct illumination*) and receive/reflect/bounce light themselves from their surroundings (*indirect illumination*).

The Global Illumination solution used in the Viz Engine Render Pipeline is an improved version of the one introduced with Viz Artist 3.9. Major enhancements have been made to increase the realism of the generated output. It relies on precomputed lightmaps to generate indirect lighting. Those lightmaps are updated at runtime when light conditions change.

Important: Global Illumination requires DirectX 9 runtime to be installed.

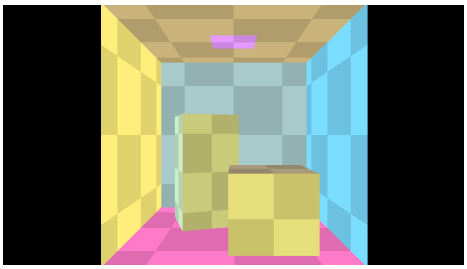
5.9.1 Precomputation

This is the most important part for Global Illumination. Without a successful precomputation, no GI is possible. While setting up a scene, it's always good to keep an eye on scene structuring, because it makes life easier when running into problems. The success of a precomputation is influenced by a few settings, which can be controlled either via the scenes Global Illumination Settings panel or the GI Settings plug-in.

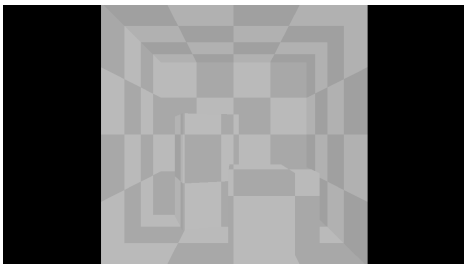
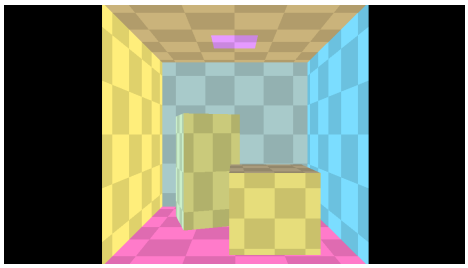
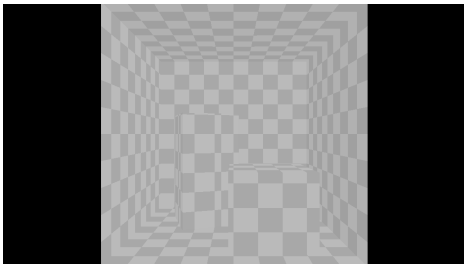
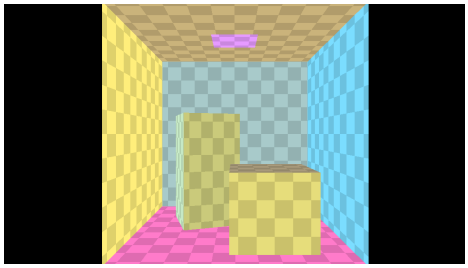
Lightmap Pixelsize

The value of *Lightmap Pixelsize* determines the resolution of precomputed lightmaps. A higher value decreases the resolution and improves performance, but reduces lighting quality. A lower value increases the resolution and improves lighting quality, but reduces performance. Setting the value too low can cause the precompute to fail. The preconfigured value of `50` is a good starting point and indicates that the area of 2x2 pixels of the lightmap are used for a 1m x 1m area of geometry.

Examples

Lightmap Pixelsize	Output Pixel Size Debug	Charts debug
`50`		

Copyright © 2026 Vizrt

Lightmap Pixelsize	Output Pixel Size Debug	Charts debug
`25`		
`10`		

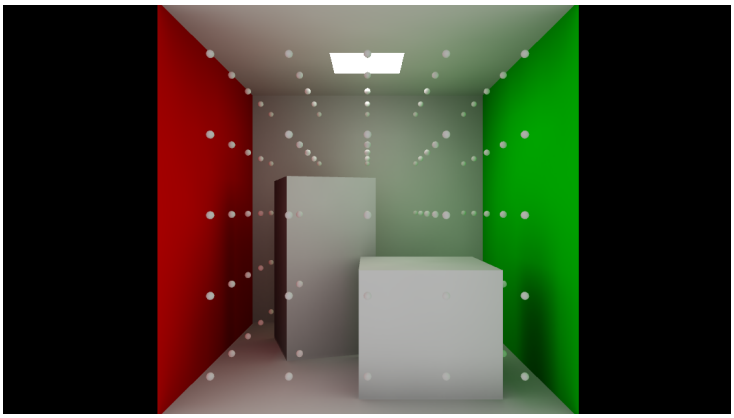
Probe Volume Resolution

Indirect lighting for dynamic objects is calculated using Spherical Harmonics (SHs). To be able to evaluate the SHs at runtime at any position, a grid of light probes can be defined using the *Probe Volume Resolution*.

If no dynamic objects are used in the scene, this value can be kept low. Use the Light Probes Debugging option to show the placement and lighting values of the probes at their predefined grid positions.

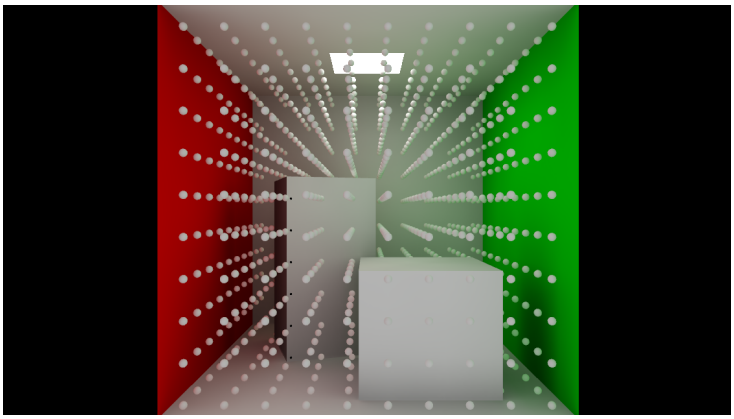
Examples

Probe Volume Resolution	Screenshot

`x = 5`
`y = 5`
`z = 5`
	

Copyright © 2026 Vizrt

Probe Volume Resolution	Screenshot

`x = 10`
`y = 10`
`z = 10`
	

GI Settings Plug-in



One part of the precomputation step generates separate lightmap UVs for each geometry. A precomputed real-time light map has a maximum size of 262,144 pixels (512x512, 1024x256, ...) and corresponds to a so called *System*. While Viz Engine 3.x only supports a single System per scene, multiple systems can be configured for the Viz Engine Render Pipeline using this plug-in.

If precomputation fails because not all geometries fit into a single lightmap, either the *Lightmap Pixelsize* can be increased or a new *System* can be added.

Parameters

Plug-in Parameter	Description
`Visibility`	*Render* means that the container itself and the hierarchy beyond is visible to the real-time renderer. *Global Illumination* means the container itself and the hierarchy beyond is visible to the Global Illumination system. This is for advanced scene design and if not sure, just leave both *Render* and *Global Illumination* enabled.
`System ID`	Name of a System. All geometries in with the same *System ID* are placed into the same lightmap. The same *System ID* can be used in multiple instances of the plug-in.
`Instance Type`	*Radiosity* marks the geometries on the container itself and the hierarchy beyond as immovable and lit by a lightmap. *Fully Dynamic* marks **geometries** on the container itself and the hierarchy beyond as dynamic, they are not included in the lightmap, can be moved at runtime and are lit by the SH probes.

Copyright © 2026 Vizrt

Tip: There is no absolute rule for how large to make your systems, that is, how much geometry to assign to one system. However, it is recommended that you keep your systems small. The precompute time per system should be less than ten minutes, as a general rule.

Note: The GI Settings plug-in supersedes the automatic mesh classification of Viz Artist 3. Automatic mesh classification is not supported anymore in the Viz Engine Render Pipeline.

Precompute

The **Precompute** button starts all GI computation steps. The current scene is analyzed and all necessary files are created so the GI backend can do its work. In the Viz Engine Render Pipeline implementation, incremental changes to the scene and incremental computation steps are supported, as long as the scene does not get not closed in between.

5.9.2 Baking

Baking in general defines the process of pre-computing values so that at a later point those values can be re-used to speed up another process. The Viz Engine Render Pipeline allows you to bake out high-resolution lightmaps which can replace lighting calculations during runtime with a single texture lookup. Baking can be mixed seamlessly with real-time lighting and/or real-time radiosity calculations. Baking is useful for scenes with many static lights.

The size of the final textures is set by the `Map Size` parameter. Increasing the map size increases the quality of the output, but also increased baking time.

Ambient Occlusion

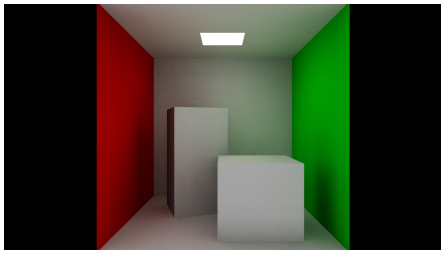
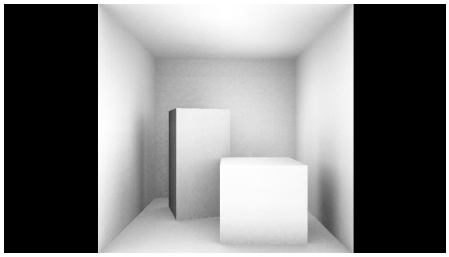
Parameters

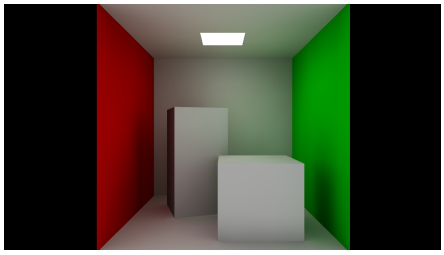
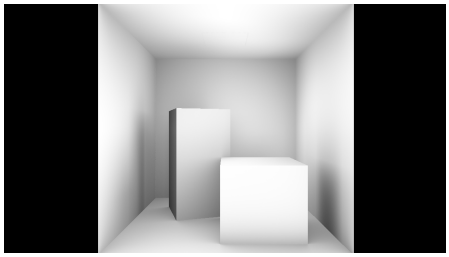
Parameter	Description
`Bake AO`	Enables or disables usage of Ambient Occlusion texture.
`Quality`	Influences the amount of rays being used during AO. A lower value produces noisier result.
`Trace Distance`	Sets the distance to search for occluders. This value sets a distance below which a point is considered occluded. You want to tune this parameter based on the size of your scene.

Copyright © 2026 Vizrt

Examples

Settings	Screenshot	AO debug

`MapSize = 512`
`Quality = 6`
`Trace Distance = 50`
		

`MapSize = 1024`
`Quality = 10`
`Trace Distance = 50`
		

Baked Lighting

Lightmap baking is automatically enabled and used when you have lights in the scene, which have the **Bake** setting enabled.

Parameters

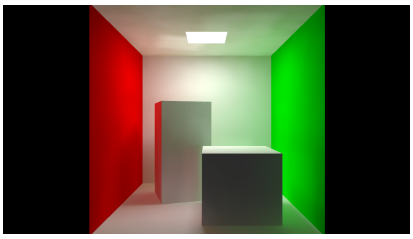
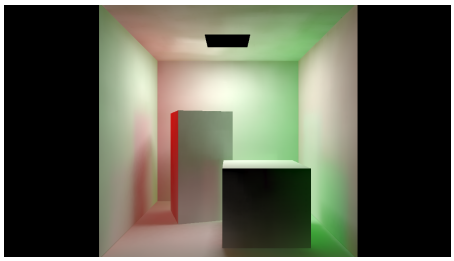
Parameter	Lightmap	Description
`Final Gather Quality`	indirect	The number of rays to cast for each output pixel during final gather.
`Final Gather Samples`	indirect	The number of samples to cast rays from for each output pixel during final gather.
`AA Samples`	direct	The number of samples to take per-pixel in order to reduce aliasing. This is the maximum number of times to supersample a lightmap pixel. For preview bakes, a value of `1` is sufficient, but you may notice jagged shadow edges in the output. The default value of `8` removes the majority of aliasing issues.

Copyright © 2026 Vizrt

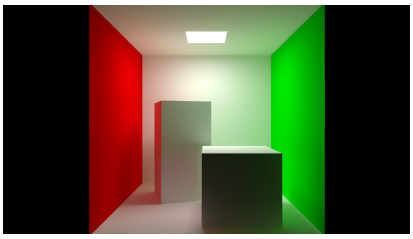
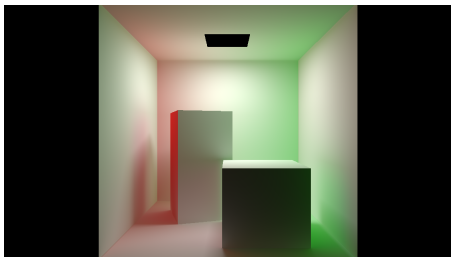
Parameter	Lightmap	Description
`Blur Radius`	direct	The size of the radius of the post-processing filter in pixels that blurs the direct light baking. The filter is aware of geometry and visibility between pixels and thus no light leaking occurs. The filter allows otherwise sharp shadows to be smoothed without the need to use area light sources. In general, a value in the range of `2` and up leads to good results.
`Area Light Quality`	direct	The number of samples to take per-area light in order to capture soft shadowing effects. This is the maximum number of rays to trace to any light with an area. Larger area lights require more rays to produce noise-free output. The default value works with the majority of scene setups, but you should increase this value if you observe noise.

Examples

Settings	Screenshot	Baked Light debug

`MapSize = 512`
`Final Gather Quality = 8`
`Final Gather Samples = 4`
`AA Samples = 2`
`Blur Radius = 2`
`Area Light Quality = 16`
		

Copyright © 2026 Vizrt

Settings	Screenshot	Baked Light debug
`MapSize = 1024` `Final Gather Quality = 512` `Final Gather Samples = 16` `AA Samples = 8` `Blur Radius = 2` `Area Light Quality = 64`		

Bake All

The **Bake All** button initiates lightmap baking only. If **Bake AO** is enabled, ambient occlusion gets baked. If lights with `Bake` flag enabled are found in the scene tree, direct and indirect lightmaps get baked. **Bake All** only performs steps the GI backend considers necessary (e.g. if **Final Gather Quality** is changed from one bake to another, only the indirect lightmap is rebaked).

Tip: When setting up a scene with lights, baked lights, baking options, it can happen that baked textures are out-of-date or are incorrectly computed. If this happens, use the Utilities **Rebuild Bake Textures** feature. This clears all previously baked textures and starts a fresh bake step.

5.9.3 Rendering

When both precomputation and baking achieve the desired result, some parameters can still be tweaked to adjust the composition process of real-time direct lighting, baked lighting and ambient occlusion. In addition, environment color and intensity can be defined to simulate a simple environment light.

Parameters

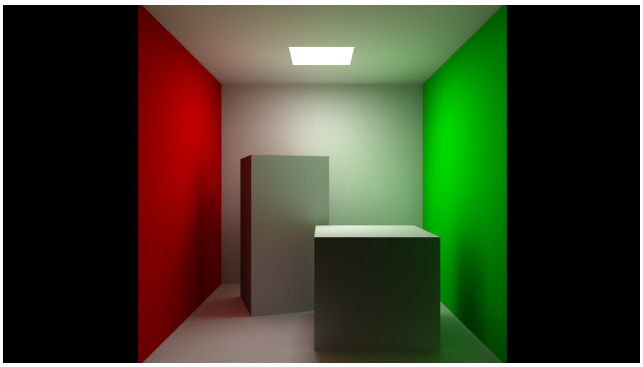
Setting	Description
`Ambient Occlusion Factor`	Modifies the intensity of the baked ambient occlusion texture.
`Environment Color` `Environment Intensity`	Both values combined set the environment color being passed to the GI backend.

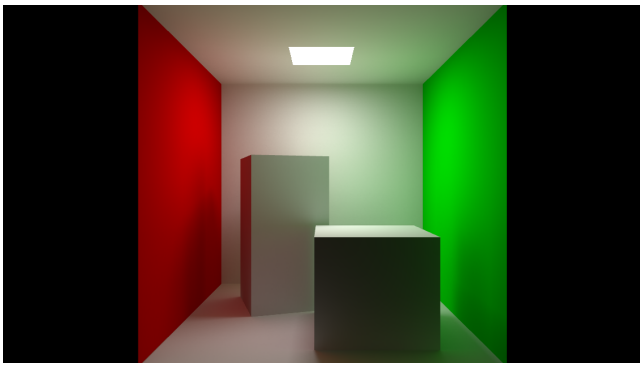
Copyright © 2026 Vizrt

Setting	Description
`Baked Light Direct Factor`	Modifies the intensity of the baked direct light texture.
`Baked Light Indirect Factor`	Modifies the intensity of the baked indirect light texture.

Examples

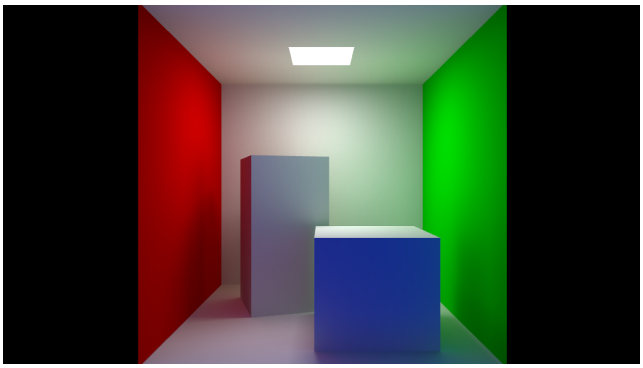
Settings	Screenshot

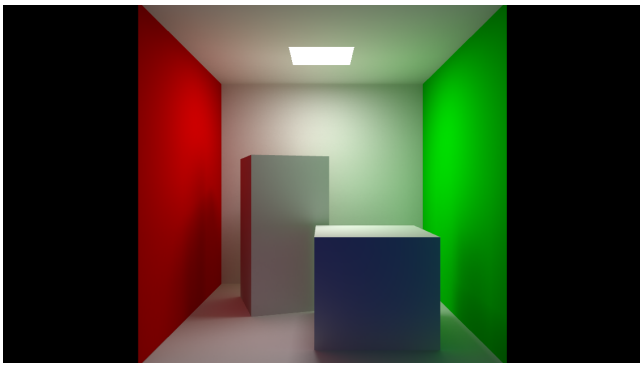
`Ambient Occlusion Factor = 100`
`Environment Color = Black`
`Environment Intensity = 100`
`Baked Light Direct Factor = 100`
`Baked Light Indirect Factor = 100`
	

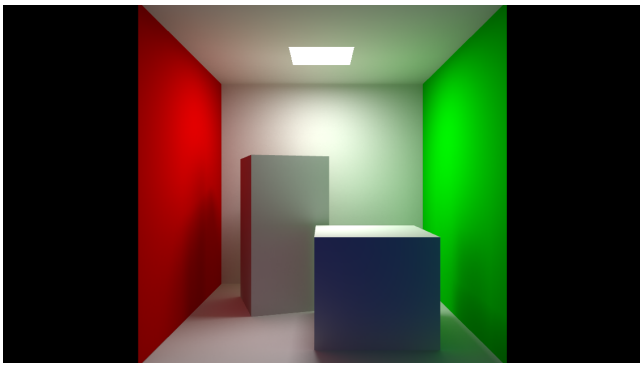
`Ambient Occlusion Factor = 75`
`Environment Color = Black`
`Environment Intensity = 100`
`Baked Light Direct Factor = 100`
`Baked Light Indirect Factor = 100`
	

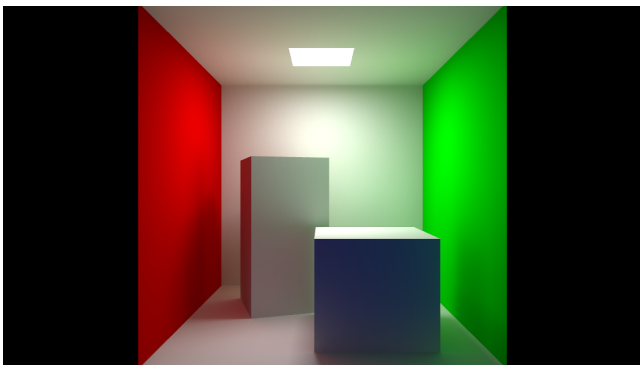
Copyright © 2026 Vizrt

Settings	Screenshot

`Ambient Occlusion Factor = 75`
`Environment Color = Blue Tint`
`Environment Intensity = 100`
`Baked Light Direct Factor = 100`
`Baked Light Indirect Factor = 100`
	

`Ambient Occlusion Factor = 75`
`Environment Color = Blue Tint`
`Environment Intensity = 20`
`Baked Light Direct Factor = 100`
`Baked Light Indirect Factor = 100`
	

`Ambient Occlusion Factor = 75`
`Environment Color = Blue Tint`
`Environment Intensity = 20`
`Baked Light Direct Factor = 150`
`Baked Light Indirect Factor = 100`
	

`Ambient Occlusion Factor = 75`
`Environment Color = Blue Tint`
`Environment Intensity = 20`
`Baked Light Direct Factor = 150`
`Baked Light Indirect Factor = 125`
	

Copyright © 2026 Vizrt

5.9.4 Geometry Requirements

Geometries that are considered during GI precomputation need to be properly triangulated, have normals and one UV set.

In contrast to Viz Artist 3, the Viz Engine Render Pipeline does neither require nor support a second UV sets for precomputation.

> When a geometry is detected that does not follow the requirements `WARNING: Global Illumination FX: Geometry 'containername' not supported` is printed to the console.

5.9.5 Material Texture

Both real-time GI and baked lighting require material properties to be able to properly calculate lighting and bounced lighting. Material Texture is introduced in the Viz Engine Render Pipeline, that supports material colors and textures being used. Albedo and Emissive material textures are generated when the real-time GI processing is started or prior to the light baking process.

The material texture is generated with the UVs being generated during precomputation and directly correlates with real-time GI lightmap textures.

Mapping

PBR Material Property	Material Texture
`Color Texture` `Color Tint`	`Albedo`

`Emissive Texture`
`Emissive Color`
`Emissive Intensity`
	`Emissive`

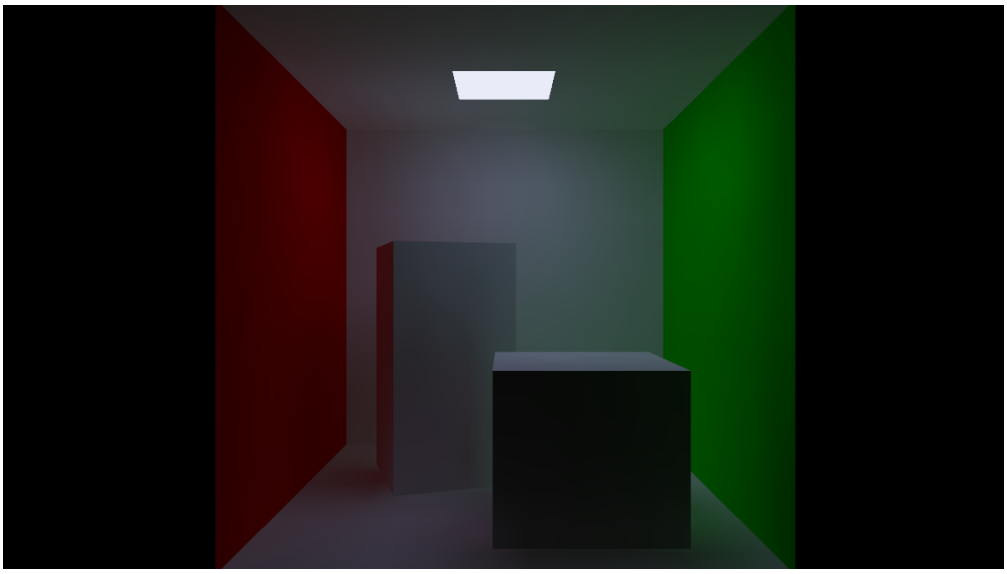
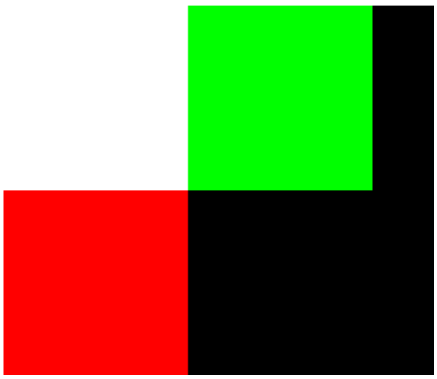
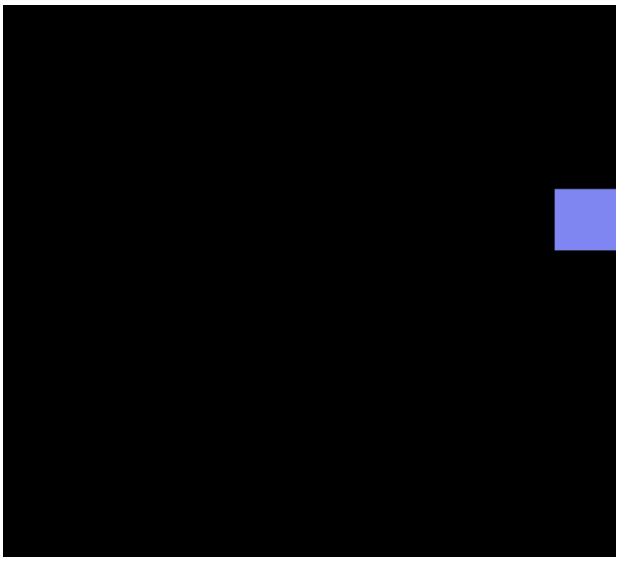
> **Information:** When changing a material color or texture while authoring, the material texture must be updated manually by clicking the **Update Material Texture** button.

> **Tip:** When using multiple GI systems, make sure you do not use referenced materials (from Viz Graphic Hub) in more than one system as the lighting might show up incorrect. Please de-reference all used materials

Examples

The scene used for the screenshot does **not** include any light in the scene, the whole lighting is calculated by the GI using the emissive property of the rectangle on the top of the box and ambient occlusion texture:

Copyright © 2026 Vizrt

Screenshot	
	
Albedo Material Texture	**Emissive Material Texture**
	

Copyright © 2026 Vizrt

5.10 Gaussian Splatting



Viz Engine supports native rendering of Gaussian Splat files, adding ultra photorealism to virtual Studios and AR objects.

Rendering Gaussian Splats requires a license

5.10.1 How Gaussian Splatting works

Traditionally, getting a real location into a Viz scene means someone rebuilds it: geometry is modelled, materials are authored, lights are placed, and an artist spends days chasing the way light actually behaves in that room. The result is a model of the place.

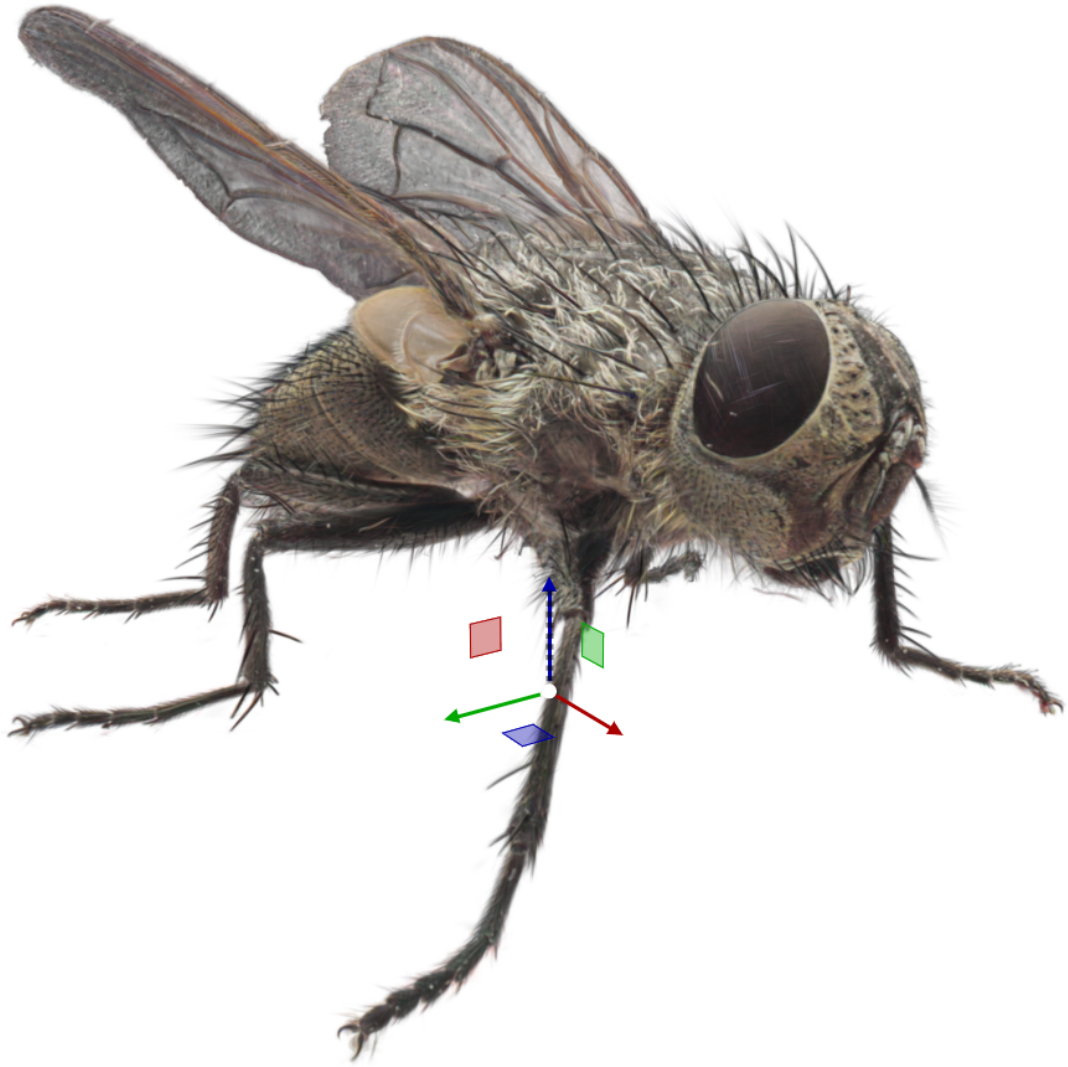
Gaussian Splatting takes a different route. You photograph or film the location, and software works out a representation that *reproduces those photographs* and not by building surfaces, but by filling the space with millions of tiny, soft, coloured blobs. Each blob is a stretched, semi-transparent 3D ellipse. Individually they are meaningless smudges. Collectively, seen from any angle, they resolve into the room.

The technical name for one of those blobs is a 3D Gaussian, hence "Gaussian Splatting". The captured location is a splat scene or splat cloud.

The important thing for a designer is what it replaces. This is not a new way to build geometry. It is an alternative to building geometry at all — for the parts of a scene that already exist somewhere in the real world.

Copyright © 2026 Vizrt

5.10.2 Why it looks real



Two reasons, and neither is resolution.

Nothing was interpreted. In a modelled set, every material is somebody's estimate of how a surface behaves, and every light is somebody's guess at where the real light was. In a capture, the scuffed floor, the bounce off a white wall, the grime in a corner and the slight unevenness of the ceiling lights are all simply present, because they were present when the shutter opened.

The highlights move. Because each splat's color depends on viewing angle, reflections and specular sheen shift as the camera travels — the single strongest cue the eye uses to decide whether it is looking at a real space or a

Copyright © 2026 Vizrt

rendered one. A tracked camera move through a splat scene behaves the way a camera move through a real room behaves.

5.10.3 Benefits and Limitations

Plays to its strengths	Fights it
• Real locations you cannot get a camera into on the day — stadium interiors, venues, streets, historic buildings • Set extensions and backdrops behind a tracked camera • Messy organic detail: foliage, gravel, worn concrete, fabric, clutter • Anything where "looks modelled" is the failure mode • AR objects that must sit convincingly next to real footage in the same frame	• Mirrors and clear glass — these break the solve, producing smeared artefacts • Anything that moved during capture: people, traffic, blowing leaves • Scenes that must be relit, or lit to match a different time of day • Objects that need to be animated, deformed, or taken apart • Surfaces that need real collision or physics • Angles nobody captured — the camera cannot go where the photographer didn't

5.10.4 To use Splat files for Viz Engine

Capture images
Photos from a camera or rendered frames



Create a point cloud
Turn the images into a cloud of 3D points

Copyright © 2026 Vizrt



Turn them into "splats"
Each point becomes a soft,colored, see-through blob



Render
Overlapping blobs merge into a smooth, real view

Copyright © 2026 Vizrt



Viz Engine is only involved in the Render part, however we will give some advice on the preparation steps to create the data needed:

- **Capture images**
  Someone walks the location taking heavily overlapping stills or video, covering every angle the final shot might use. This is the stage where quality is won or lost. Additionally you can also render a sequence of images or clips from within a modelling software.

  There are many ways how to capture these images. From professional automated camera setups, drone flights, mobile phone applications and many more. Also traditional 3D Render Software can be used to create these images.

> The splat scene is only solid from directions that were photographed. Plan the capture around the camera moves you intend to shoot — and if a move is added later, it may need a return visit rather than a scene fix.

- **Create a point cloud and turn them into splats**.
  Software first works out where each photo was taken from, then places and refines the splats — rendering its own attempt, comparing it against the real photo, and adjusting. This runs once, offline, on a workstation. It takes minutes to hours depending on the size of the capture.

Public Sources

There are also many web sites and communities offering splat files for commercial and free usage. Some examples:
https://www.useblurry.com/

https://superspl.at/

Software for processing a capture

> Vizrt does not recommend any particular workflow, but we want to give some advice.

Copyright © 2026 Vizrt

The tool landscape moves quickly, and pricing and platform support change from release to release — treat the table as a starting point and check current details before standardizing on anything.

Tool	Platform	Splat output	Where it fits
Capture on a phone — shoot and solve in one app			
Polycam	iOS · Android · web	.ply	The general-purpose scanning app. Choose **splat PLY** on export — the other export options give you meshes, which is a different thing entirely. Free tier plus subscription.
Scaniverse	iOS · Android	.ply · .spz	Processes on the device, so nothing leaves the phone. Export PLY unless Viz turns out to read SPZ. Free.
Luma AI	Cloud / web	.ply (inside a .zip)	Choose **Gaussian Splat** on export, then extract the PLY from the downloaded archive. Fastest look-see, least control.
Training — turn images into splats, with control			
Postshot	Windows	.ply	The common desktop default. Trains locally on your own GPU, exposes splat count and training steps, crops and cleans in the same tool.
nerfstudio / gsplat	Python · CUDA	.ply	Open source and scriptable — the route for a repeatable in-house pipeline. Needs camera poses supplied to it.
LichtFeld Studio	Windows · Linux	.ply	Newer trainer, exports PLY for interchange.
Brush	Cross-platform	.ply	Cross-platform trainer. Expects camera poses; does not solve them itself.
Camera alignment — necessary upstream, but produces no splat file			
RealityCapture · RealityScan	Windows	none — poses only	Industry-standard alignment. Outputs camera positions and a sparse point cloud that a trainer consumes.
Agisoft Metashape	Cross-platform	none — poses only	Handles 360° source imagery, useful for interiors.

Copyright © 2026 Vizrt

Tool	Platform	Splat output	Where it fits
COLMAP	Cross-platform	none — poses only	The reference structure-from-motion tool most trainers expect input from. Slower, free, predictable.
Clean-up and conversion — the only source of SOG			
SuperSplat	Browser	.ply · compressed .ply · .sog	PlayCanvas' editor. Crop, select and delete splats, then export. The only mainstream interactive tool that writes SOG. Free, open source.
SplatTransform	Command line (npm)	.ply · .sog	`splat-transform in.ply out.sog`. The scriptable version of the step above — what you would automate a watchfolder around.
Large and aerial captures			
DJI Terra	Windows	.ply · 3D Tiles	Drone captures of buildings, venues and sites, with georeferencing and very high image counts per task. Licensed, enterprise.

Before you pick one

- **Where does the data go?** Cloud-processed apps upload your footage to a third party. For an unannounced set, a client location or anything under embargo, that is a policy question, not a technical one — and it is the main argument for on-device or local processing.

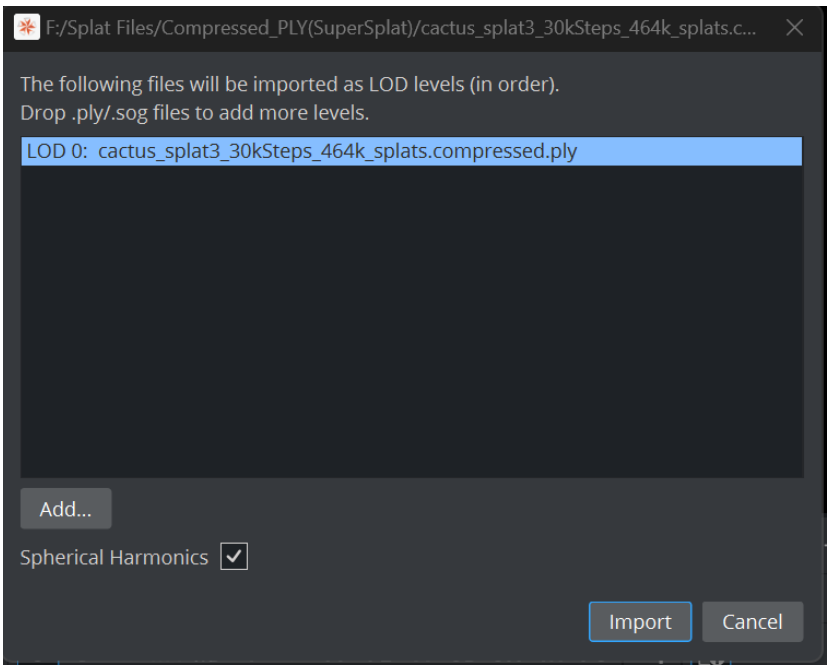
Copyright © 2026 Vizrt

5.10.5 Importing Splat Files

Viz Engine supports importing splat files in the following formats:

sog (Spatially Ordered Gaussians)	A highly compressed file format used for 3D Gaussian Splatting data
ply (Polygon File Format)	A 3D data file that stores the positions, colors, shapes, and transparency values of thousands of tiny overlapping light points

To import these file types, follow the normal import workflow or drag & drop them into Viz Artist Asset view.



Spherical Harmonics: See Spherical Harmonics

The import dialog lets you add multiple Level Of Details LOD files.
Make sure they are in correct order.

> Importing can take a while and as this is a background task, there is no visual feedback until the imported file shows up as new geometry inside the asset view.

Once the import is finished, you'll see a new Geometry asset in your Asset view. Simply drag it to your scene tree to start rendering.

Copyright © 2026 Vizrt

5.10.6 Rendering

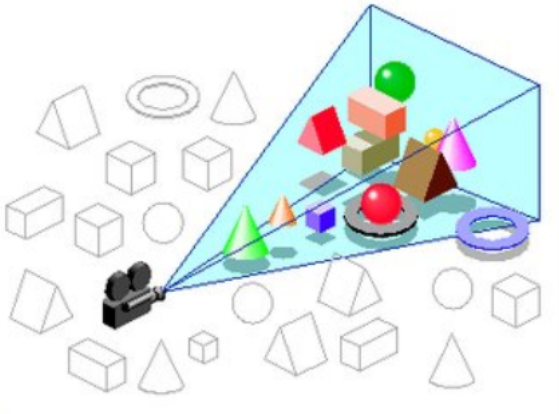


“For something that looks this convincing, the rendering is unusually simple.“

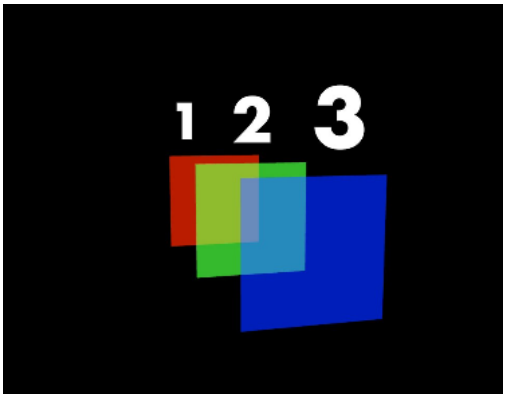
- High level overview
- Coordinate Systems
    - Presets
    - Order and Flip
    - Size threshold
    - Spherical harmonics
    - Off, then on again
- Rendering
- Re-Lighting of Gaussian Splats
- Splat
- Proxy
    - Setting up a proxy
- Known Limitations

High level overview

Culling
Check what needs to be rendered and not



Sort
Splats are transparent - therefore, they must be sorted back to front



Render
Each splat is rendered as an Ellipsoid (in reality, a quad)

Copyright © 2026 Vizrt

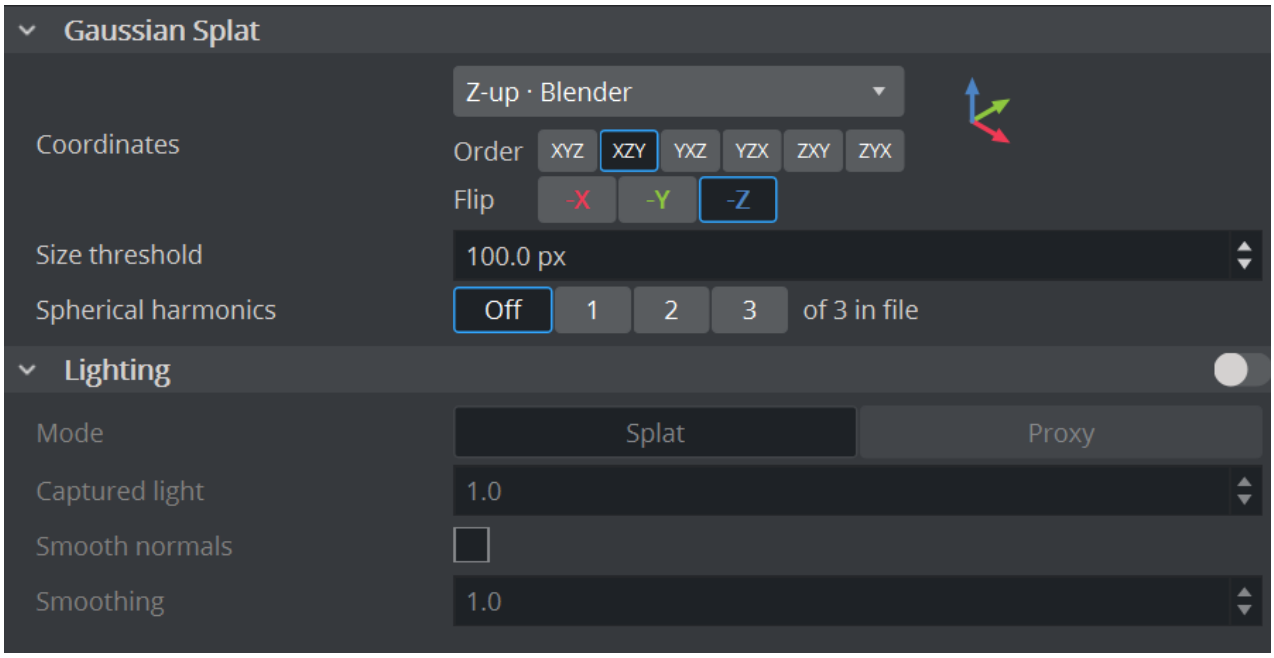


As all splat files come in as geometry, you can simply drag them to the scene tree as new object. These can be used as Virtual Studios, as AR objects and many more.

Coordinate Systems

It might be necessary to choose the correct coordinate system and flip the axis if your model is rendered incorrectly and does not match Viz Engine coordinate system. To do so

- split the geometry
- click on the sub container's geometry



Copyright © 2026 Vizrt

Splat files come from different capture and editing tools, and they don't agree on which way is up. The Coordinates control converts the file's axes into Viz space, where **X** points right, **Y** points up and **Z** points towards the viewer.

Presets

Most files match one of three named conventions. Pick one from the list, and **Order** and **Flip** are set for you.

Preset	Typical source	Order	Flip
Y-up · OpenGL / Viz	Already in Viz space	XYZ	none
Y-down · COLMAP / 3DGS	Y points down and Z points forward, as in most 3DGS training output	XYZ	**−Y −Z**
Z-up · Blender	Z points up and Y points forward	XZY	**−Z**

When the current Order and Flip don't match any preset, the list shows **Custom**. This includes a freshly imported model: the importer's default is `XZY` with **-X -Y**, which isn't one of the presets.

Order and Flip

1. **Order** picks which file axis goes on each Viz axis. `XZY` means Viz X takes the file's X, Viz Y takes the file's Z, and Viz Z takes the file's Y.
2. **Flip** then negates individual Viz axes. It is applied after the order, so **−Z** always negates the axis that ends up on Viz Z, whichever file axis that is.

The small axis preview next to the list shows the result. Each arrow is a file axis in its own colour (**X**, **Y**, **Z**), drawn where it ends up in Viz space.

> **Changing either value rebuilds the model.** The splats are decoded again from the stored data and the LOD octree is rebuilt, so a large model may take a moment to reappear. To line a model up with the rest of the scene, prefer the container's transformation. Order and Flip are for correcting the file's own convention.

Size threshold

The splats are stored in an octree with coarser levels of detail near the root. While rendering, the engine walks down the octree and stops at a node once its projected size on screen falls below the threshold, then draws that coarser level instead of refining further.

- **Lower values** draw more detail and more splats, and cost more GPU time.
- **Higher values** stop earlier and draw fewer, larger splats. Use this for distant or background models.

The value is in pixels. The default is `100.0 px` and the accepted range is `0` to `100000`. Changing it takes effect on the next frame and doesn't rebuild the model.

Copyright © 2026 Vizrt

Spherical harmonics

Spherical harmonics (SH) store the view-dependent part of a splat's colour, such as reflections and sheen that change as the camera moves. Degree 0 is the flat base colour, and each higher degree adds a finer band of view dependence, up to degree 3.

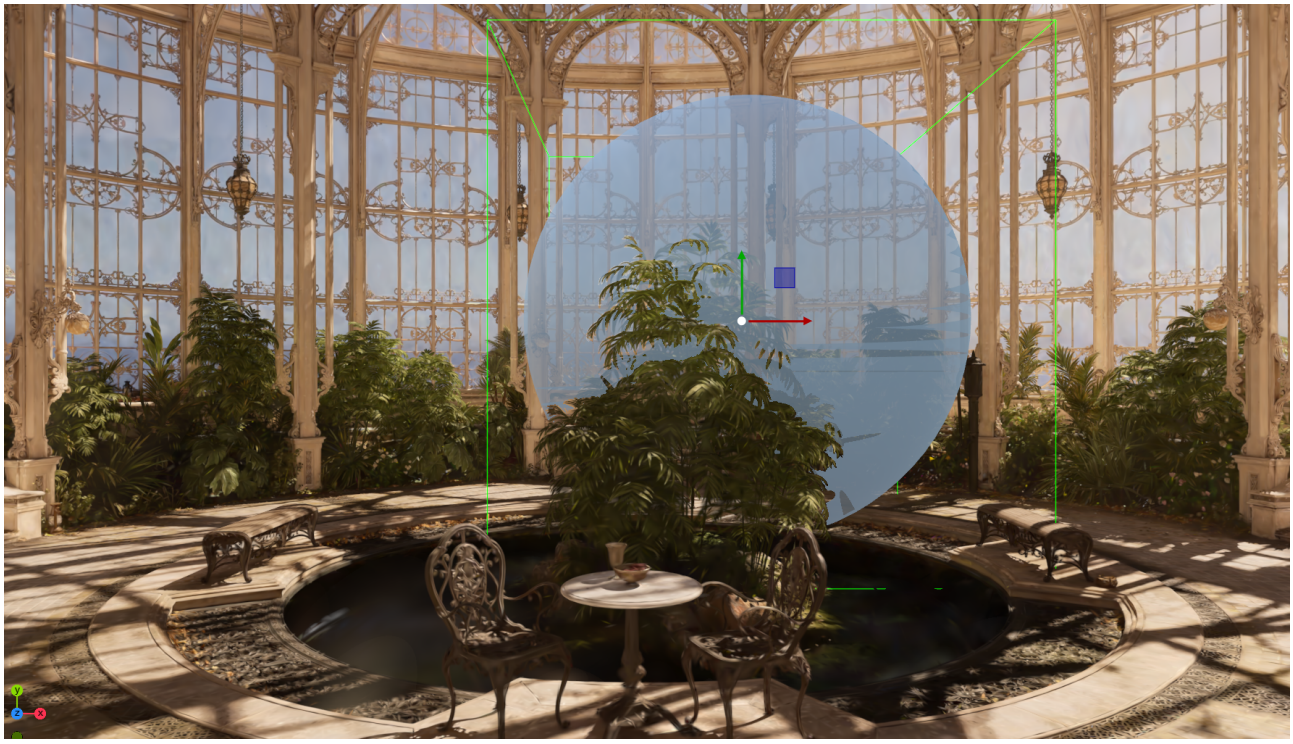
- **Off** renders flat colour and removes the harmonics from the GPU buffer. They are by far the largest part of each splat: 192 bytes, against 16 bytes for everything else.
- **1 / 2 / 3** switches the harmonics on and evaluates bands up to that degree.
- **of N in file** is the degree the imported model actually carries. Buttons above N are disabled.

“of 0 in file” means the model was imported without harmonics. It can only render flat colour. The buttons for degrees 1 to 3 are disabled, and a degree set earlier may still look selected, but it has no effect.

Off, then on again

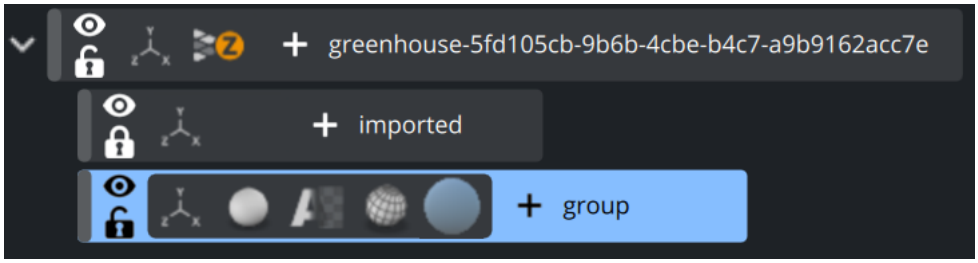
Switching to Off and back rebuilds the model from its stored data, so the harmonics come back and the model reports its original degree again. They are only lost for good when the model was imported with SH disabled. That option is worth using for very large models: it keeps the stored geometry small and within the size one stored level can hold.

Rendering



Objects can fully integrate within the imported Splat Files, but might require some additional Z-Sort information to be added.

Copyright © 2026 Vizrt



Gaussian Splat objects support

- Alpha
- Clipping Planes
- Expert Settings
- MaskSource/Target
- Window Mask

Re-Lighting of Gaussian Splats

A splat's colors carry the lighting from its capture. Viz Engine can add the scene's virtual lights on top of that lighting, so splats fit in with the other objects in the scene. Lighting needs to be manually enabled on the header of the lighting section.

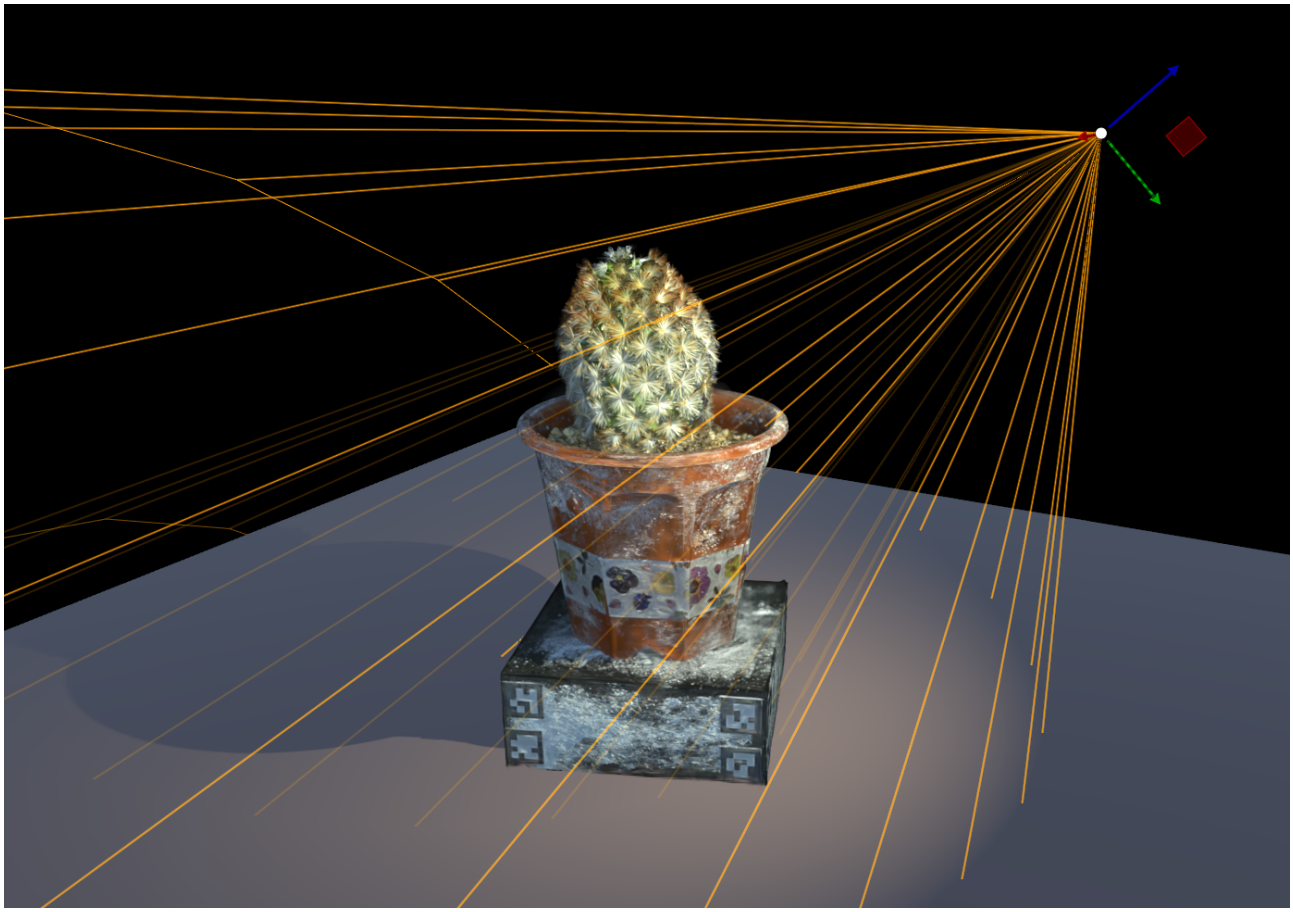
Splats usually have no normals, so we approximate it in one of two modes:

Splat

Splat (default) lights every splat individually. It gives the best quality, but it costs much more than proxy mode, especially on large captures. By default, each splat's normal is its shortest axis. That axis varies between neighbouring splats, so flat surfaces can shade unevenly. **Smooth normals** fixes this by estimating each normal from the surrounding splats, and **Smoothing** controls how many it uses: higher values shade more evenly, lower values keep more detail. Smooth normals are computed when the splat data loads, so they add to load time but not to frame time.

Control	What it does
Smooth normals	Uses neighboring splats to compute a smoother normal. Enabling it removes details, but on flat surfaces it might present a more realistic result.
Smoothing	Defines how many splats are used for computing the smooth normal. Higher values result in a smoother result with less high frequency details.

Copyright © 2026 Vizrt



Proxy

Proxy lights a rough polygon mesh of the same object and passes the result to the splats in screen space, so the mesh only has to cover roughly the same pixels. It costs little more than rendering the mesh, but the lighting is only as good as the mesh's fit. The mesh stays hidden. Align it by transforming its container. Splats it doesn't cover keep their captured lighting.

Control	What it does
Proxy container	Drop a container here. That container and its whole subtree become the proxy. The proxy is used in the splat lighting pass and in shadow rendering, and is hidden in the normal image.
Show proxy mesh	Also draws the proxy in the normal image (`PROXY_SHOW` , off by default), so you can line it up with the splats. Available only while the proxy is active and a container is set.

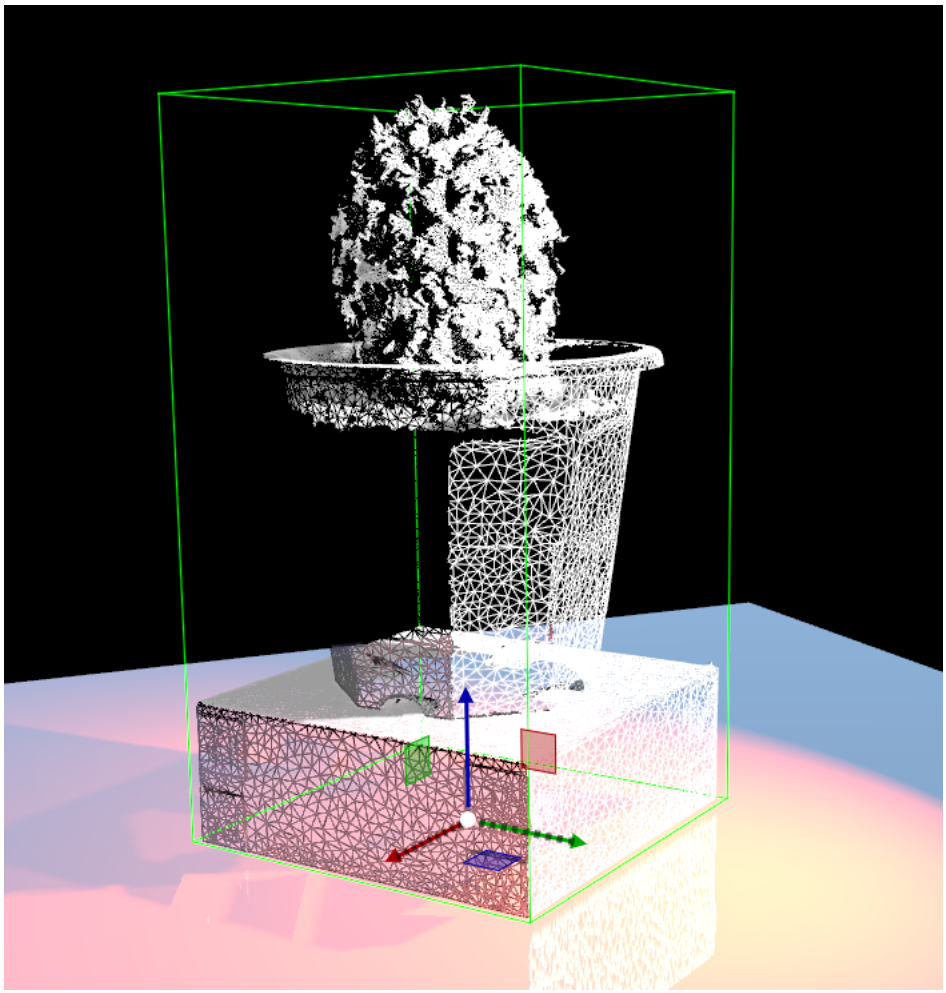
Setting up a proxy

1. Add the mesh on a container of its own and drop that container on **Proxy container**.

Copyright © 2026 Vizrt

2. Switch on **Show proxy mesh** and align the mesh with the splats using the proxy container's own position, rotation and scaling. A separately exported mesh rarely lines up out of the box. Adjust the proxy, not the splat's Coordinates, because those also change how the splats render.
3. Switch **Show proxy mesh** off again once it lines up.
4. Light the scene with containers that carry a light plugin. The V4 renderer lights the proxy, so the scene's built-in lights don't reach it.

A splat that sits inside its own proxy hierarchy isn't treated as part of the proxy. A scene can hold up to 64 proxy hierarchies.



Captured light (0–1) sets how much of the lighting baked into the capture is kept. At 1, the default, the scene lights are added on top. At 0, the splats are lit by the scene alone, which suits captures lit evenly from every side. The scene’s environment map fills in for whatever captured light is removed.

Some more examples:

Copyright © 2026 Vizrt



Virtual Studio at the Hochschule Düsseldorf - University of Applied Sciences







Known Limitations

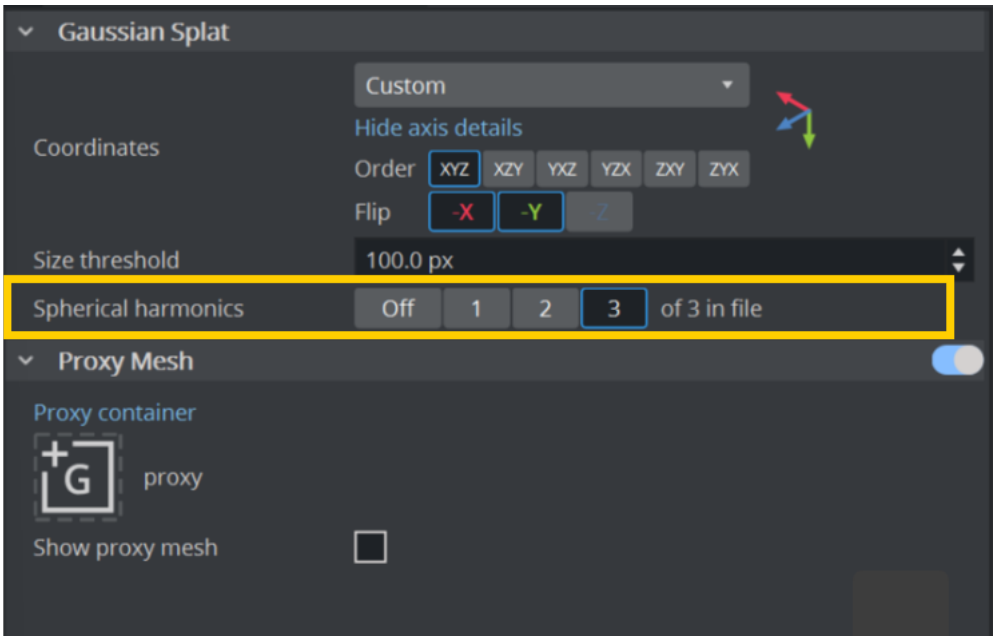
Gaussian Splats fit directly into your existing scenes, therefore they support many of Viz Engine Renderer features including SSR and more. However there are some limitations (some of them are listed here):

- Shader Effects are not supported
- Instancing
- Masking (e.g. for Tonemapper)
- Texture Slot effects like Screenspace UV masking

Copyright © 2026 Vizrt

Spherical Harmonics

A Gaussian splat does not store one color per point. It stores a color that *changes with the viewing angle*, and spherical harmonics are how that change is encoded. They are what makes a captured surface look wet, glossy or metallic as the camera moves past it — and they are by far the most expensive thing a splat carries.



What the harmonics do

Every splat has a base color that looks the same from everywhere. On top of that, a splat can carry a set of spherical harmonic coefficients: a compact description of how its colour varies over the sphere of directions you could look at it from. At render time the engine evaluates that description for the current view direction and adds the result to the base colour.

Without them, a captured scene renders flat and matte — correct in shape, but with every highlight frozen wherever the capture happened to put it. With them, highlights travel across the surface as the camera moves, which is usually the difference between a splat that reads as a photograph and one that reads as a point cloud.

The harmonics are organized in **bands**, and the band count is the **degree**. Each further degree adds finer angular detail, and costs more coefficients than the one before it.

Copyright © 2026 Vizrt

Degree 0	Degree 1	Degree 2	Degree 3
			
0 coef. Base color only. No view dependence at all — the splat looks identical from every angle.	**9 coef.** Broad directional shading. Which side of a surface is brighter, at the coarsest level.	**24coef.** Soft sheen and wide highlights. Enough for most matte and semi-gloss material.	**45coef.** Tight speculars and mirror-like reflections. The highest degree the engine evaluates.

The three properties

Spherical harmonics are controlled on the **splat geometry**, not on the container. The geometry has to be local to the scene before its properties can be reached — in Viz Artist, split the geometry off the shared Graphic Hub object first.

			Splat geometry — spherical harmonic properties
Shown as	**Range**	**Default**	**What it does**
Model SH order (Degree) `read only`	0 – 3	—	The degree the loaded model actually carries. This is the ceiling: a requested degree above it has no effect.
Spherical harmonics	on / off	on	Whether the harmonics are carried through to the octree and uploaded to the GPU at all. Switching this off is the only way to stop paying for them in memory.
SH degree	0 – 3	3	How many bands the renderer evaluates each frame. Clamped to *Model SH order*. Lowering it costs no memory and saves no upload — it only changes what is evaluated.

There is a fourth control, and it sits somewhere else: the **import task** has its own `SH` option, on by default. Turning it off drops the harmonics *before* the model is encoded, so they never reach Graphic Hub at all.

Switching *Spherical harmonics* off on a geometry is reversible. Importing without them is not.

Copyright © 2026 Vizrt

The geometry keeps the imported levels in their stored, compressed form, harmonics included. Turning the property back on rebuilds from those, so nothing is lost. Turning the option off *at import time* means the harmonics were never written — the only way back is to import the file again.

What they cost

The harmonics are not a little extra on top of a splat. At degree 3 they are the overwhelming majority of it: 45 coefficients plus padding, stored as twelve four-component vectors per splat. Every other attribute a splat carries — its position, its scale, its rotation, its base colour, its opacity — fits in a single one.

Because the difference is this large, the engine keeps two separate splat buffers per scene — one for splats that carry harmonics and one for splats that do not — so a model with the harmonics switched off never reserves space in the expensive buffer. The plain buffer also starts four times larger (one million splats against 256 thousand); both grow on demand as the scene fills them.

Choosing what to switch off

- **Need the memory back?** Switch *Spherical harmonics* off, or re-import with the `SH` option off for a permanent saving on disk and in Graphic Hub as well.
- **Need render time back?** Lower *SH degree*. Degree 1 or 2 keeps the sense of a surface reacting to the camera at a fraction of the per-frame evaluation, and the memory is untouched either way.
- **Working with a very large capture?** Import without harmonics from the start. A single harmonic channel that would need more than 4 GiB cannot be written in one block, and the import reports the level it had to skip.

If something looks wrong

- **The model renders flat, but *SH degree* says 3**
  Read *Model SH order*. A model exported without harmonics reports 0, and the requested degree is clamped down to it. The fix is a new export of the source capture, not a setting.
- **Turning *Spherical harmonics* back on changed nothing**
  The model was imported with the task's `SH` option off, so there is nothing stored to restore. Import the file again with the option on.
- **Lowering *SH degree* freed no memory**
  Expected. The degree only controls what is evaluated per frame; the coefficients stay resident. Switch *Spherical harmonics* off instead.
- **The import logged a level it could not encode**
  A harmonic channel exceeded the 4 GiB a single stored block can describe. Re-import with `SH` off, or split the capture into more, smaller levels.
- **The scene loads, the splats are there, the gloss is gone**
  Check *Spherical harmonics* on the reloaded geometry — the property is saved with the scene, so an off state travels with it.

Copyright © 2026 Vizrt

5.11 Using Multiple Textures in Viz Engine

- To Setup Multiple Textures
- To Use Multiple Textures on Objects
- Simply Add some colors
- To Use Screenspace Mapping

In Viz Engine 5, the Viz Engine Render Pipeline now also supports blending multiple textures like the multi-texturing feature in the Classic Render Pipeline. The new implementation has several advantages compared to the old classic one.

Blending textures is done by using the Texture Slot plug-in. For details on this plug-in, please refer to the Viz Plug-ins User Guide.

Texture Slot allows to use up to eight textures to be blended directly into PBR or Phong materials available in the Viz Engine Render Pipeline.
This can be used for several use cases, such as:

- Traditional blending and masking.
- Adding imperfections to 3D objects.
- Masking of geometries.
- Special Transitions.
- Glares and Flares.
- Much more.

5.11.1 To Setup Multiple Textures

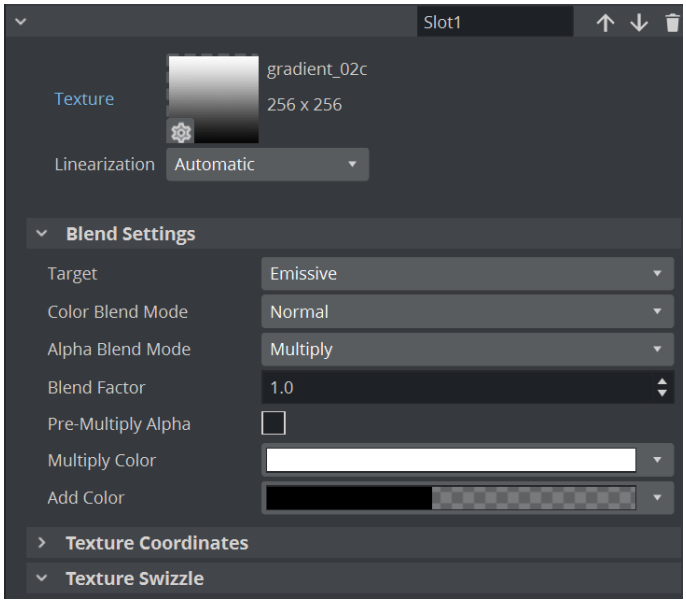
The blend multiple textures in Viz Engine Render Pipeline, a PBR or Phong material needs to be added to act as a base. As these materials already utilize various different textures (for example, normal or height maps), the Texture Slot plug-in is capable to blend into the Basecolor or Emissive part.

In our example, we start with a simple rectangle and add a few textures to illustrate how the plug-in works.

Lets start with a simple rectangle and a Phong Material. Add a rectangle, add a Phong Material and assign a base color (red).

Now we want to add some gradient effect to our base color. We add a Texture Slot plug-in and assign a gradient texture into Slot 1. It blends into the **Base Color**. Find the best blending method to apply the gradient. As an example we use *Multiply*. To reduce the effect, we can lower the blend value to something around 80%.

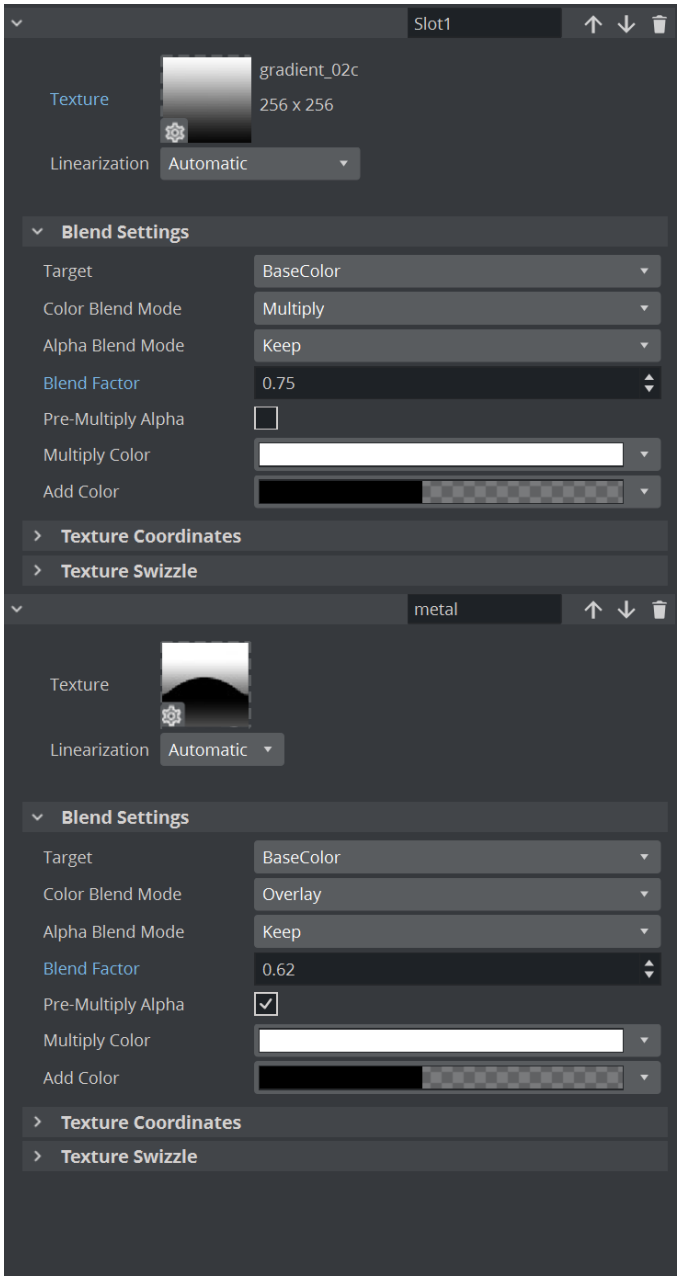
Copyright © 2026 Vizrt





We use another texture, applying some metallic effect. Create another slot named *metal* and apply the texture to the slot. In this case, we use blending into the Base Color again, und using Overlay with a factor of ~60% or higher.

Copyright © 2026 Vizrt

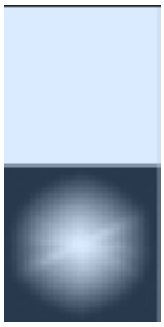


Pre-Multiply Alpha defines whether the Alpha Channel is taken into the RGB channel (enabled).

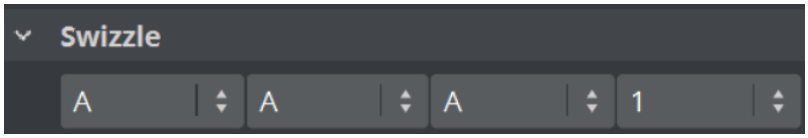
Copyright © 2026 Vizrt



Next we could add some flare effect to our rectangle. We use a flare with an alpha channel embedded.

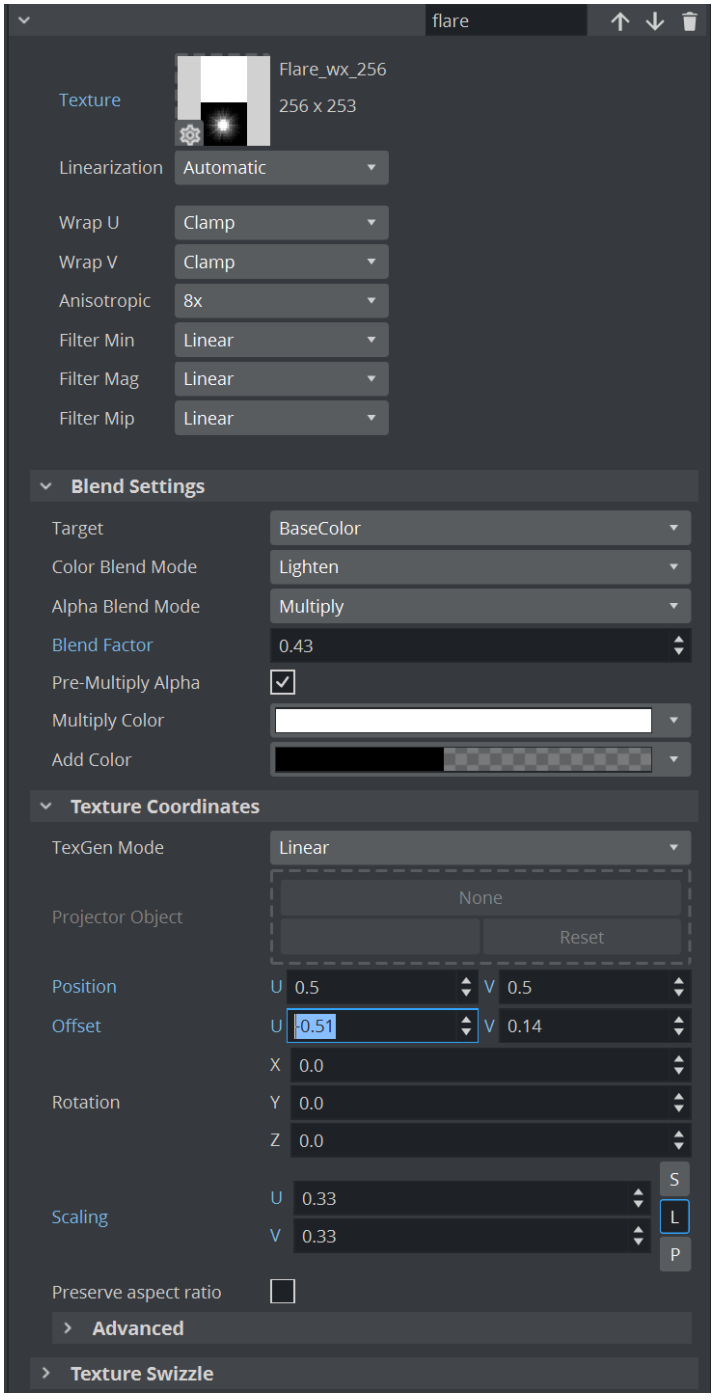


Create another Slot and drag the image to it. As this texture has its own alpha channel, we need to change the alpha blend mode to Multiply or Add to keep the alpha. As an alternative, we could also replace the RGB channels with the Alpha information by using the Swizzle function.

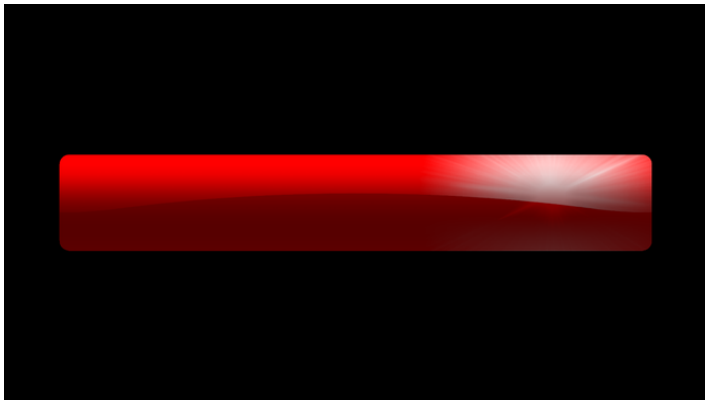


This uses the Alpha Channel as RGB and fill the existing alpha with pure white. The image we use scales up and fills the whole rectangle, therefore we change the mapping mode to be linear. We do not want to tile it, change the image properties to Warp U and V to *Clamp*. A bit of offset and Scaling and play around with the blend modes ends up with something like this:

Copyright © 2026 Vizrt



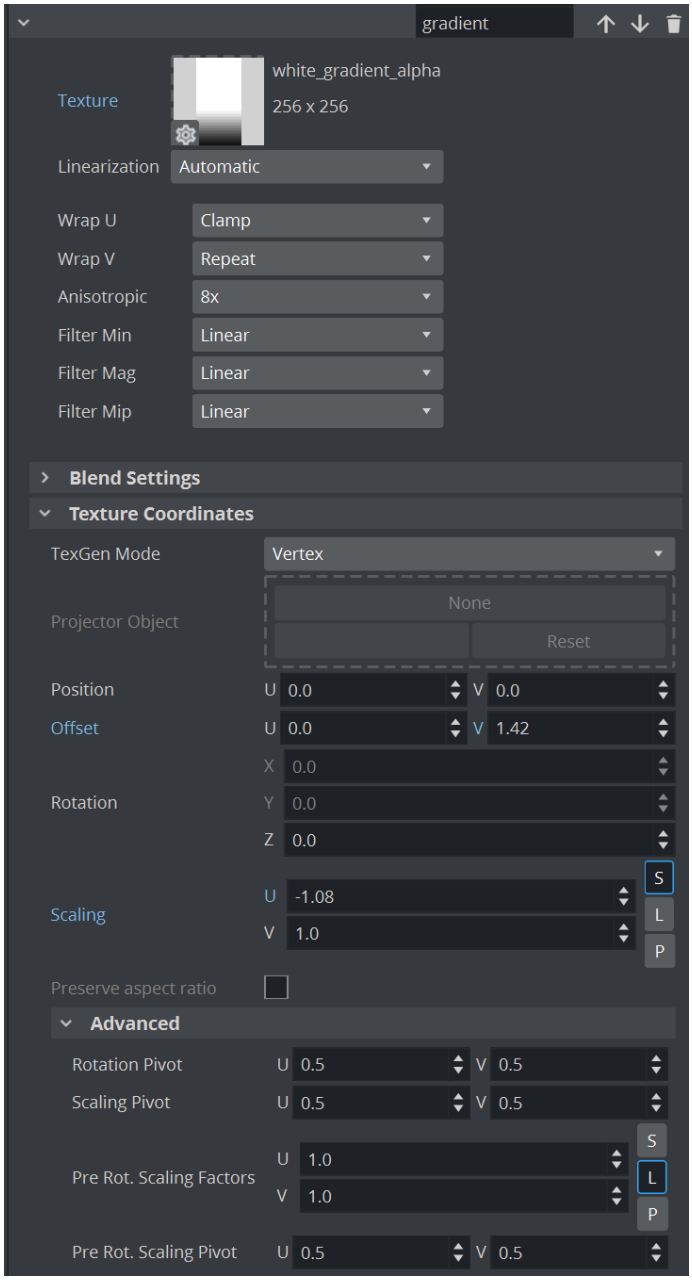
Copyright © 2026 Vizrt



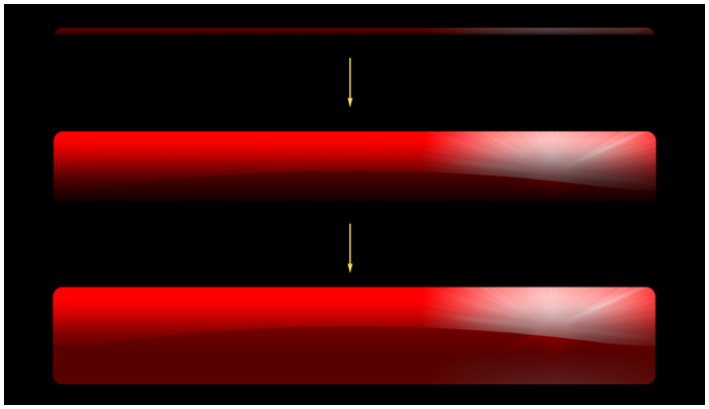
Finally, let's add a gradient to act as an IN-Effect. We replace the alpha of your image with this new one and animate it to achieve a soft wipe effect. Add a new slot, name it *Gradient*, and apply the image.

The Alpha Blend mode needs to be *replace*. Setup you mapping and offsets and animate it from out to in for example by moving the V offset.

Copyright © 2026 Vizrt

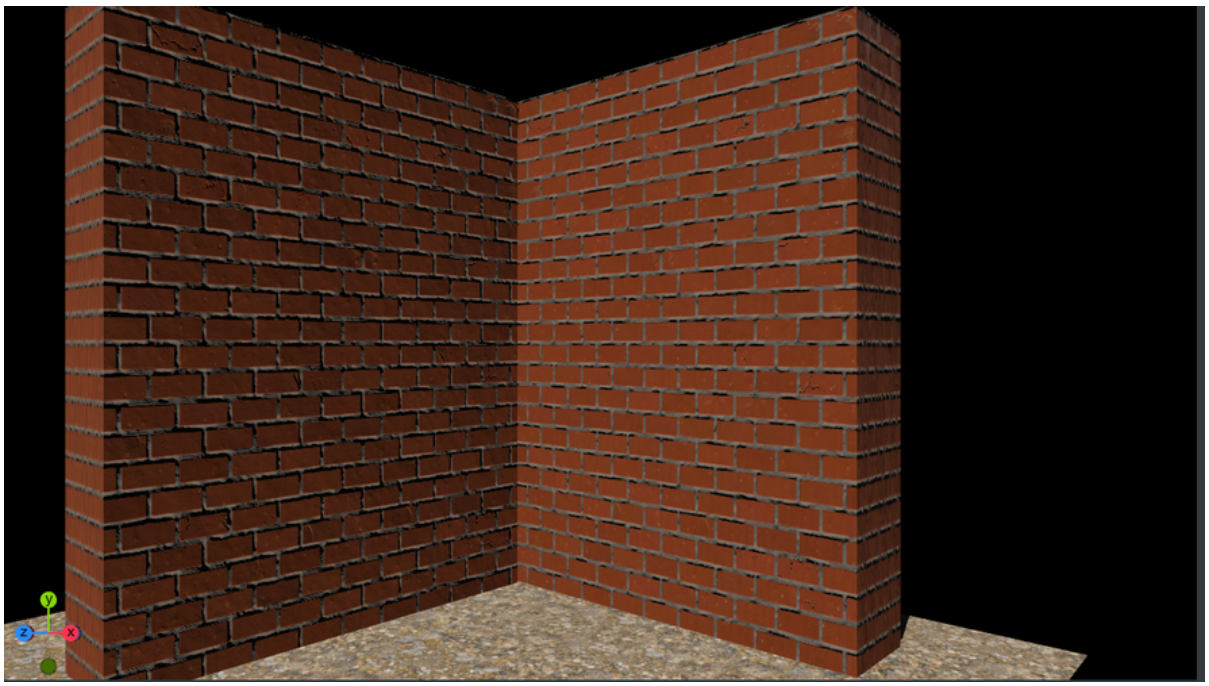


Copyright © 2026 Vizrt



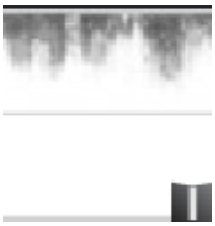
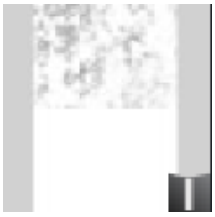
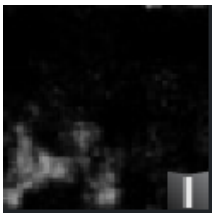
5.11.2 To Use Multiple Textures on Objects

Multi-texturing can also be used to add imperfections or decals to textures to give them a more realistic look. If we take for example this wall, it looks way too perfect. No wall looks like this.



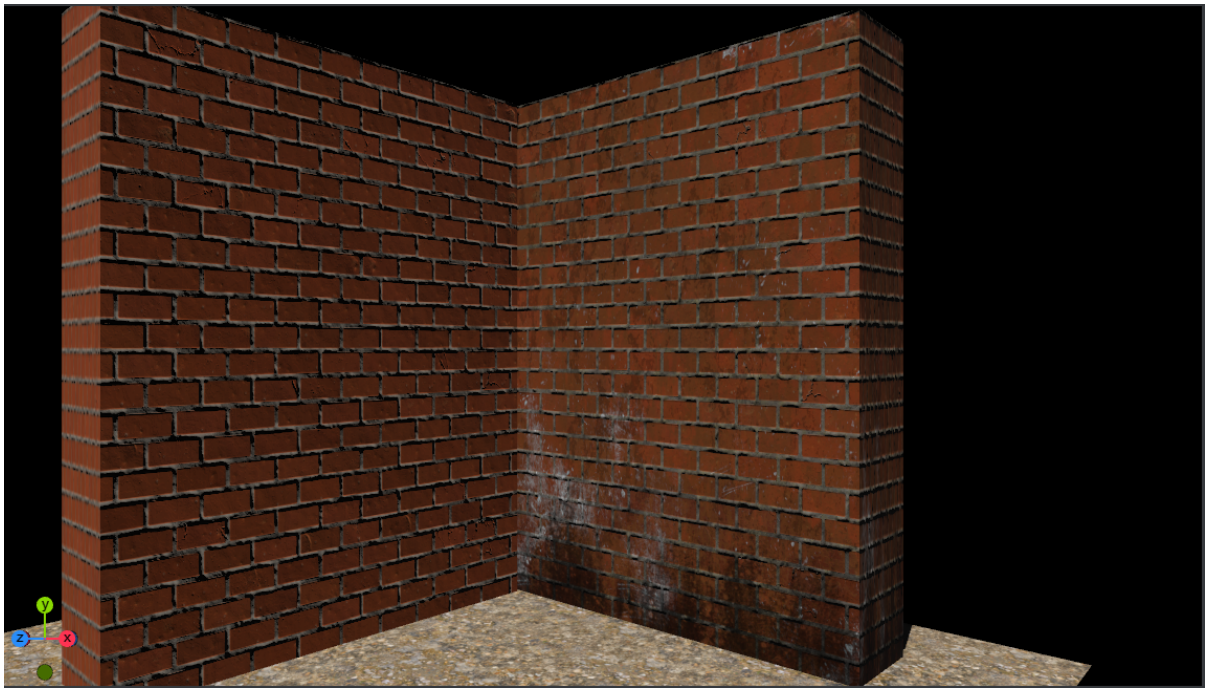
We can now add various textures to add some dirt to it, these are the textures used:

Copyright © 2026 Vizrt

Texture	How it is used
	Rotated by 180 degrees and placed on the bottom. **Multiply** blended into the Base Color with Factor 1.0. Mapped as clamped image on the bottom.
	**Multiplied** onto the whole wall to add some dirt effect with a factor of 0.5.
	Also mapped into the Base color, but as **exclusion**. This adds the white dirt on the final wall.
	Finally, more dirt is added by blending onto the wall using **Pin-Light** and a factor of 0.5

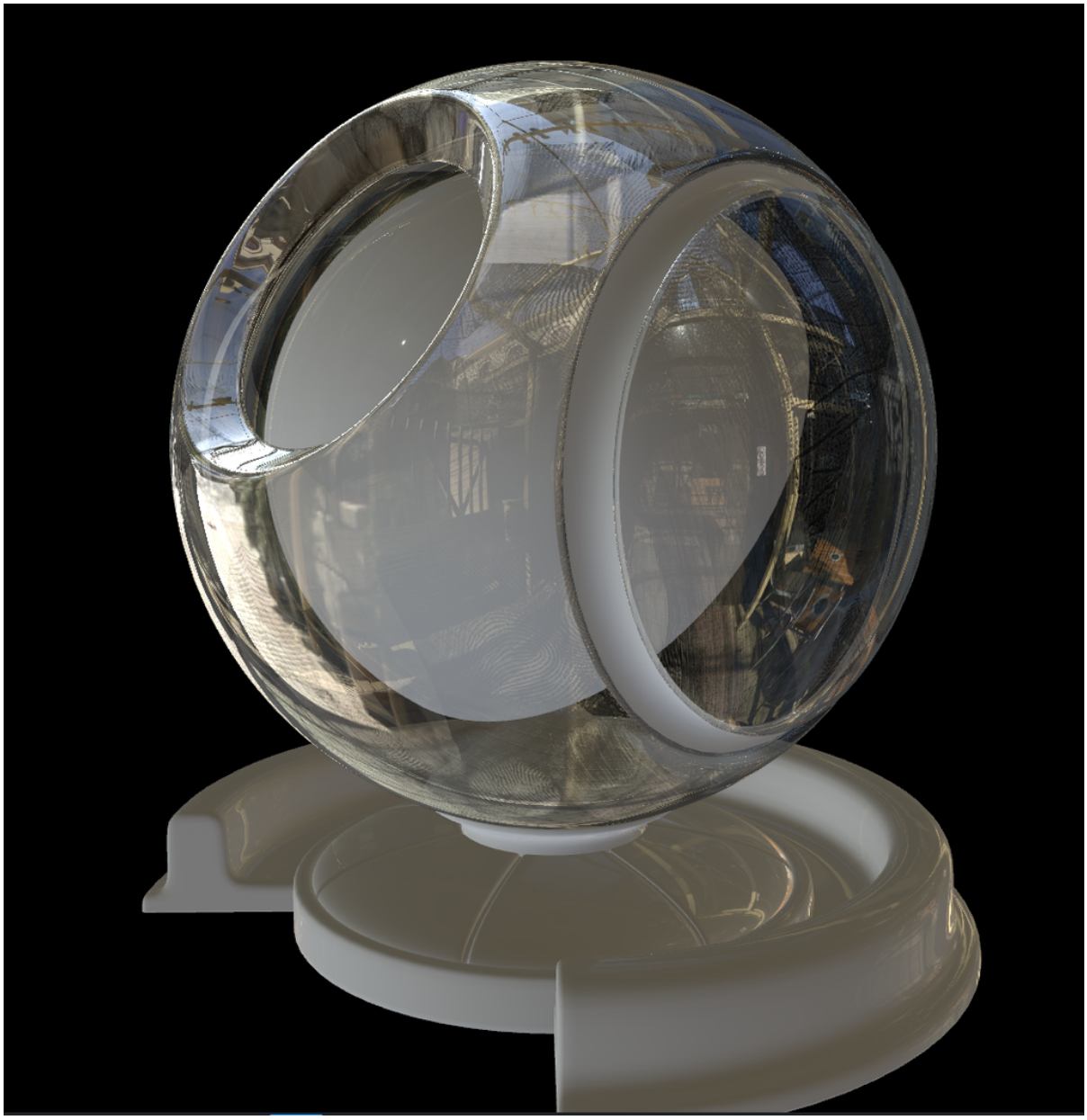
This renders into a much more realistic brick wall (right side).

Copyright © 2026 Vizrt

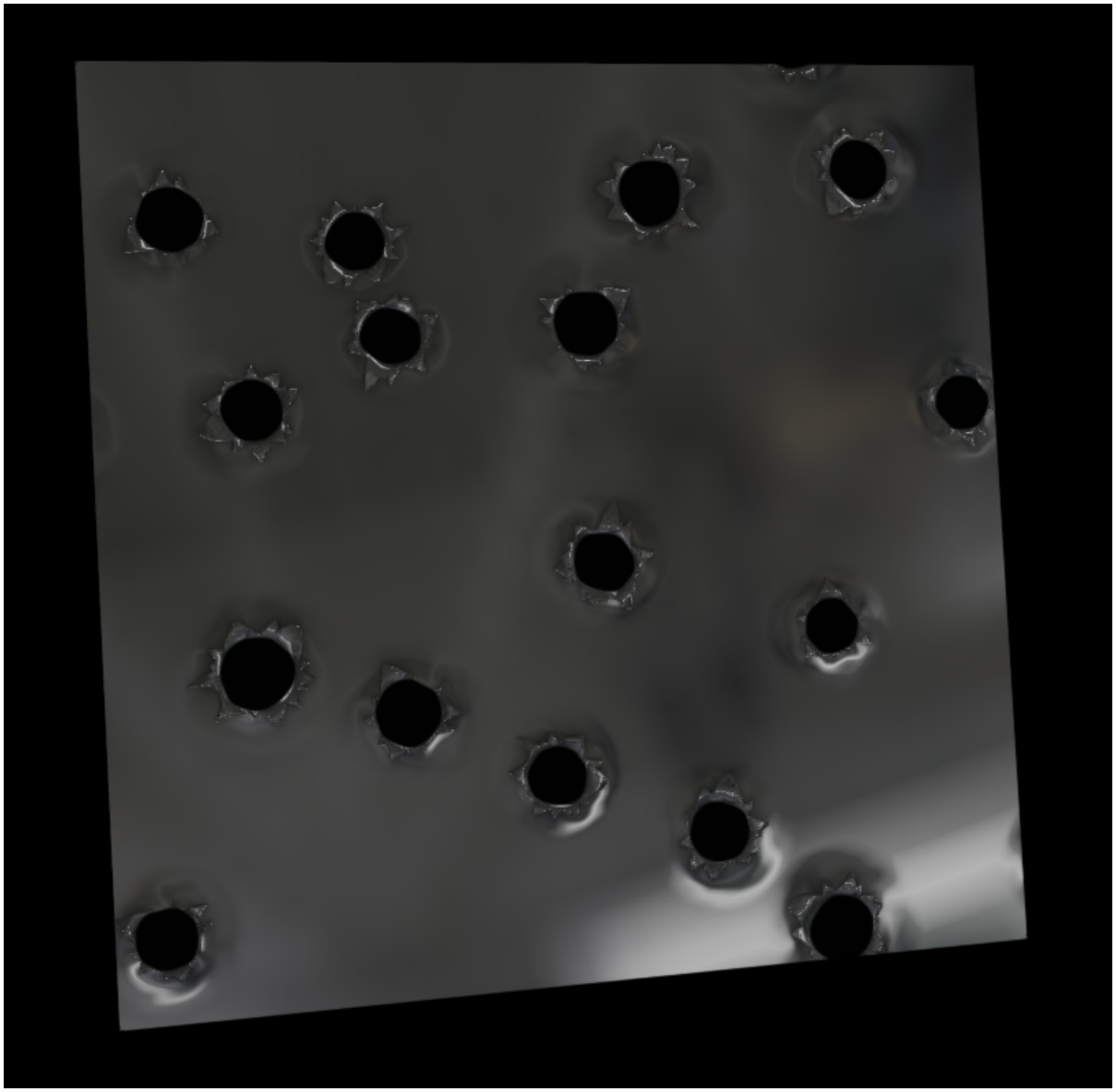


Another example could be to add footsteps on a surface, fingerprints on a glass ball, bullet holes, etc.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt



Another quite often used method is to use gradient or mask and animate them to create some transition effects. This even works with 3D objects.

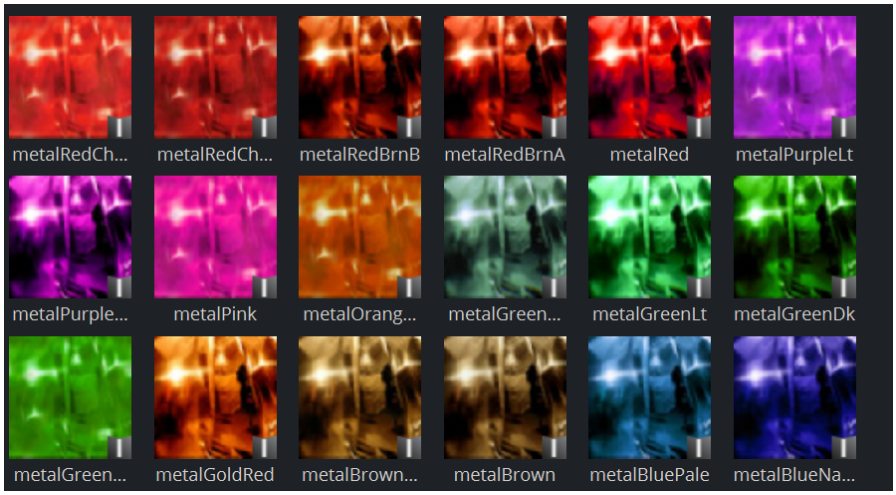
Copyright © 2026 Vizrt



5.11.3 Simply Add some colors

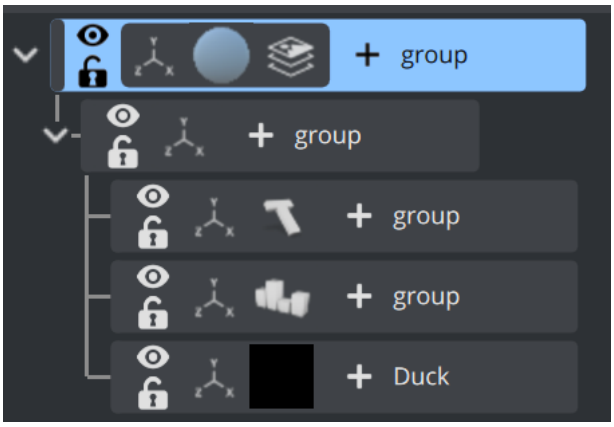
Sometimes you only want to add some colors, without the need to use a texture too.
The Texture slot plugin allows to multiply and add a color.
Instead of creating multiple variant of a texture, use a greyscale and multiply the needed color into it:

Copyright © 2026 Vizrt

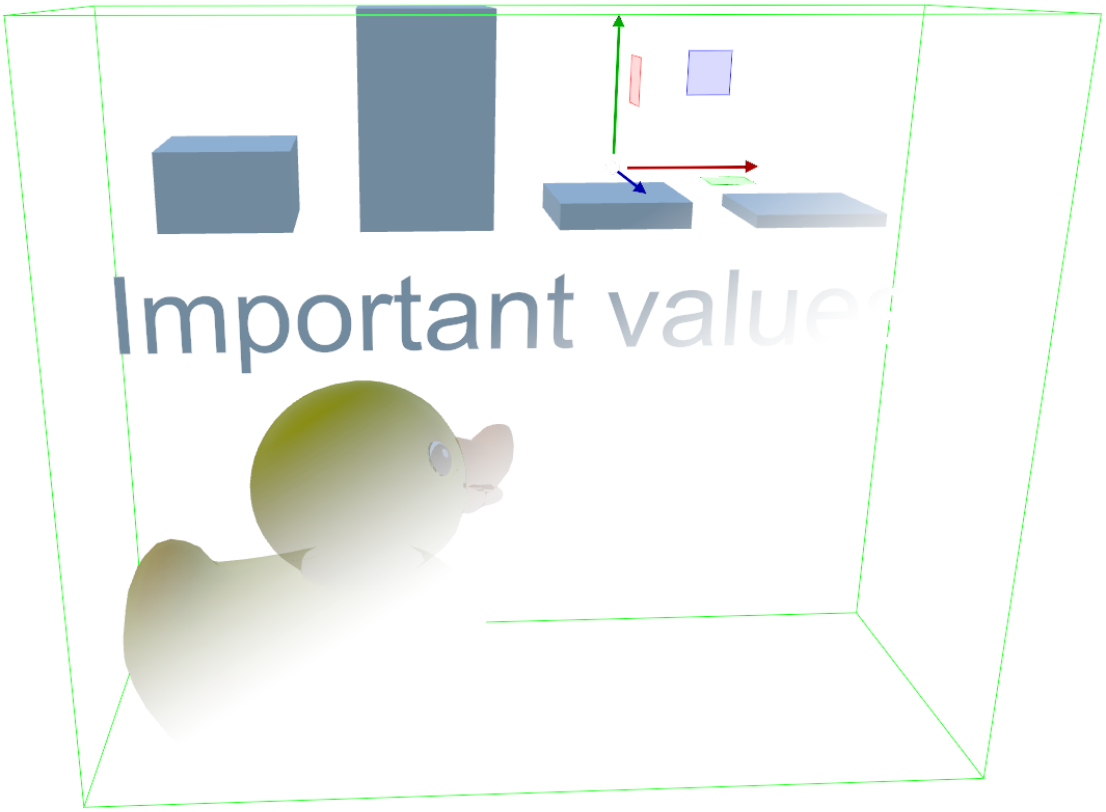


5.11.4 To Use Screenspace Mapping

When setting the Texture Generate (TexGen) mode to use *ScreenSpace*, the TextureSlot plug-in uses the underneath containers as the base to apply the effect. This allows use of, for example, Soft Masks on multiple objects.



Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

5.12 Working with Text and New Font System

Text and Font handling in Viz Engine (Viz Engine Render Pipeline) has been completely overhauled. This is to improve usability and reduce complexity. The concept of how to start working with text, which is completely different from the Classic Render Pipeline, is to *create a text object first and, then, choose a font face from a font library*.

First, a new geometry text plug-in named **Text** is introduced. The Text plug-in comes with a new text shaping and layout algorithm that works directly with new font objects stored in Graphic Hub to ensure that your scene looks and feels the same on every machine. Also, it comes with full support for per-character properties and ligature handling natively. And of course, Unicode-based text in processing.

Lightweight, possible to detect duplication, possible to build font library, and possible to support new font features (Open Type font features).

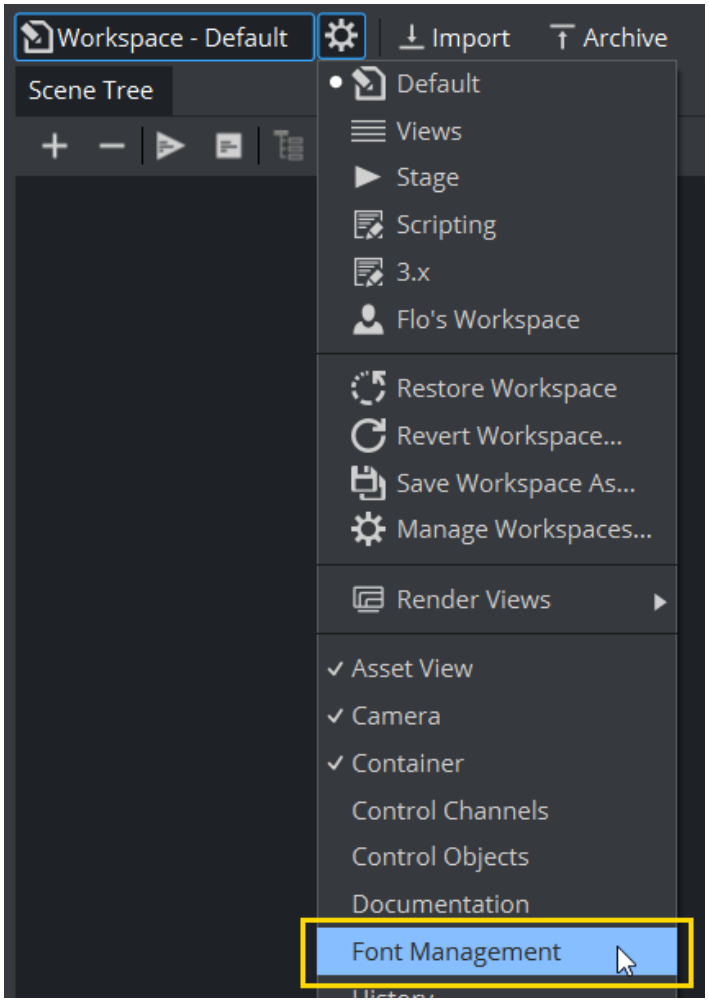
5.12.1 Getting Started

The best way to get started is to have some text rendered with **Text** in the Viz Engine Render Pipeline. First, Open the **Font Management** panel to import new fonts into Graphic Hub so that new font objects are available to be used.

Import New Fonts

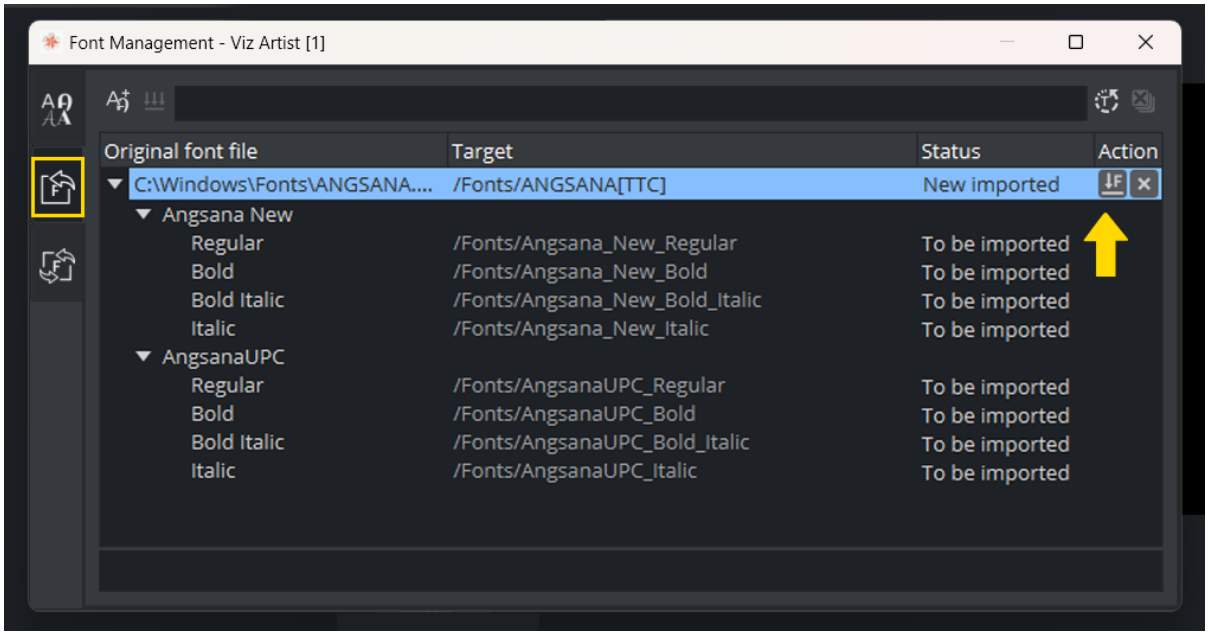
To open the **Font Management** panel, click on the cogwheel button on the top left right next to the workspace. Navigate to **List of available Docks > Font Management**.

Copyright © 2026 Vizrt

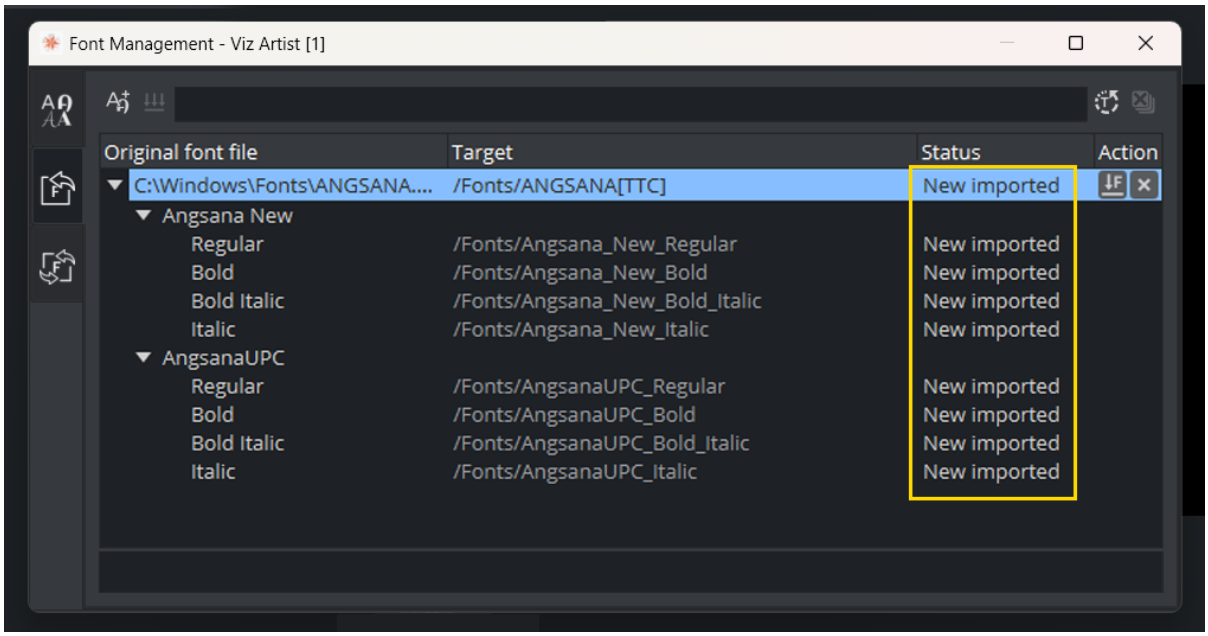


Then, go to the **Import Fonts** tab and drop some fonts into it. And press the **Import** button.

Copyright © 2026 Vizrt



Here, a font is imported to Graphic Hub and is ready to be used.



Create a Text Object



Next, drag the **Text** plug-in from the Plugins panel to either a Scene tree or a Scene Editor. The **Text** plug-in can be found in the Default folder.

Click on the **Text** icon in a container to open **Text Editor**.

Copyright © 2026 Vizrt

Then, in the **Text Editor**, click the database icon and click **Update Font Library** to fetch all available fonts.

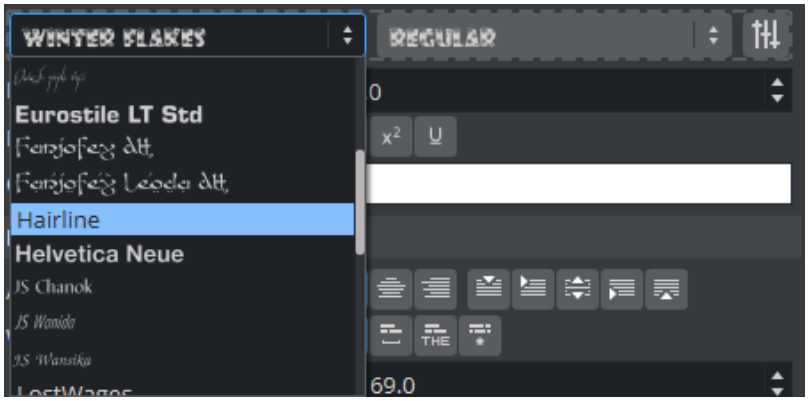
> **Note:** The first time fonts are fetched could take some time if there are many fonts in Graphic Hub.



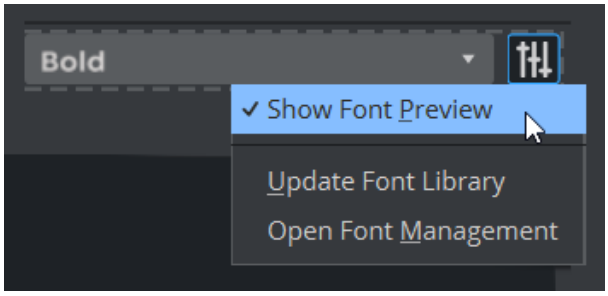
Choose a font and add some text. Then, congratulations. Your ever first text with **Text** in the Viz Engine Render Pipeline is shown.

Tips

Sometimes it can be hard to identify the font when using the font preview.



To identify the font, you can either wait until the tooltip shows up, or you can turn off the preview in the Settings menu:

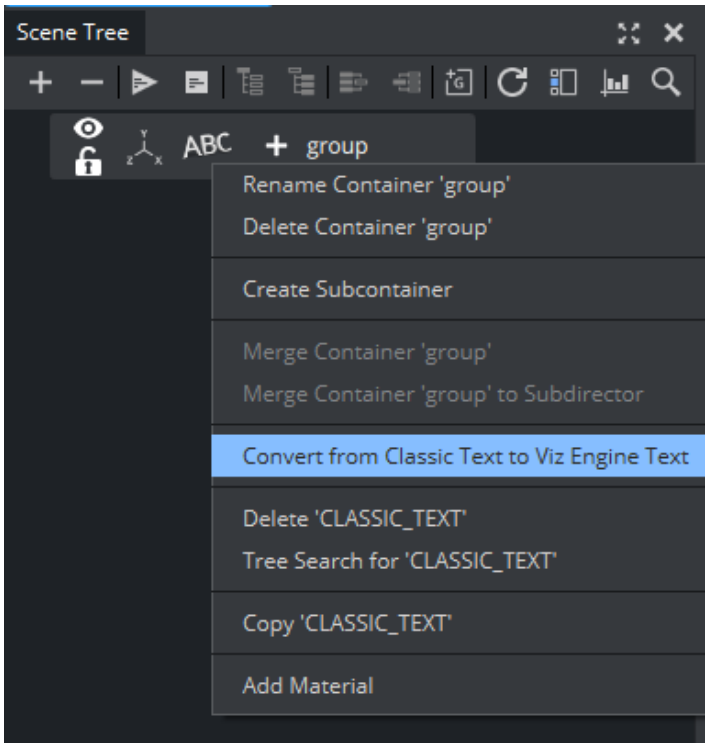


In this case, the preview is not shown anymore, but the tooltip shows a preview of the font instead.

Copyright © 2026 Vizrt



To convert an existing scene from the Classic Render Pipeline to the Viz Engine Render Pipeline, you can switch the Render Pipeline to Viz Engine and then select your classic font, right click and choose **Convert to (new) Text**. This creates a new Viz Engine Font object, replaces your existing classic font with a Text plug-in and rebuilds the Font library.



Examples

Viz Engine Fonts can be used in combination with all features of the Viz Engine Render Pipeline.

Copyright © 2026 Vizrt





Copyright © 2026 Vizrt



See Also

- Supported OpenType Features
- Working with Fontstyle Items

Copyright © 2026 Vizrt

5.12.2 Supported OpenType Features

Overview of the Supported OpenType Features in Viz Engine:

Features	Viz Engine Render Pipeline	Classic Render Pipeline

Text Shaping	The shaping is done by HarfBuzz library.
	• **GSUB** and **GPOS** and various other tables maybe used.
	• HarfBuzz default supported shaping features:
	  • **abvm**, **blwm**, **ccmp**, **locl**, **mark**, **mkmk** and **rlig**
	• Additionally supported shaping features for horizontal text:
	  • **calt**, **clig**, **curs**, **dist**, **kern**, **liga** and **rclt**
		The result depends on multiple factors.
		• For non-complex text, shaping is done naively.
		• For complex Script text, Microsoft Windows Uniscribe API is used, which may produce different results on different versions of the operating system.

Bold, Italic and other styles	All font styles available in Graphic Hub are listed separately in the style selector in the Viz Artist UI. • Optionally, the custom Embolden and Shear effect can be applied as the outline effect, although the result may not be as satisfied to look as how the font designer intended. • The effect is limited to one per font per text, but users can save the preferred effect as a new Font Face object in the Graphic Hub and reuse it separately from the original design.	Each font style is imported as a separate font object in the Graphic Hub. The classic text UI has an additional buttons for Bold and Italic, which: • Searches for the selected styles in the Graphic Hub and uses it (if any). There is no guarantee which one is chosen if multiple font objects of the same style are found. • If the search returns nothing, the selected styles are artificially mimicked.
Subscript and Superscript	Custom implementation. Font-provided subscript and superscript information is currently not supported.	Custom implementation.

Copyright © 2026 Vizrt

Features	Viz Engine Render Pipeline	Classic Render Pipeline
Color Font (Emoji)	Support OpenType **CPAL** and **COLR version 0** with Razor render method. • COLR version 1 which has gradient color is currently not supported. • Other formats (SVG, sbix, CBDT/ CBLC) are currently not supported.	No support.
Underline (UnderlinePosition & UnderlineThickness)	Custom implementation. Font-provided underline information is currently not supported.	Underline information may be pulled from the post or CFF2 table.
Variable Font and Instances	Tables possibly used: • **fvar**, **avar**, **cvar**, **gvar**, **MVAR**, **HVAR** and **VVAR** • Instances (named presets) are regarded as different font styles and are created as separate objects in Graphic Hub when imported.	No support.
Line spacing (Ascender, Descender & Line gap)	Only line spacing from **OS/2** table is used. • If **USE_TYPO_METRICS** flag is set, **sTypoAscender**, **sTypoDescender** and **sTypoLineGap** are used to determine line spacing. • Otherwise **usWinAscent** and **usWinDescent** are used.	Line spacing information from **hhea** table being preferred over line spacing from any other table.

Copyright © 2026 Vizrt

5.13 Viz Engine Shaders

This page contains information on the following topics:

- Introduction to Viz Engine Shaders
- Properties for Viz Engine Shaders
- Viz Engine Fragment Shader
    - Fragment Shader API
        - Geometry API Functions
        - Environment Map API Functions
        - Rendering Globals API Functions
        - Container Lights API Functions
        - Global Illumination FX API Functions
- Viz Engine Vertex and Fragment Shader
    - Vertex Shader API
        - Geometry API Functions
        - Rendering Globals API Functions
        - Environment Map API
        - Passing Values from Vertex to Fragment Shader
        - Viz GLSL Extension
        - Example
- User Parameters
    - Viz GLSL Extensions
        - Examples
        - Environment Map
- Transition Shaders
    - Example
        - Important Parts of the Code
        - To Use the Transition Shader
    - Transition Shader API
        - Geometry API Functions
        - Transition API Functions
- Shader Samples Library
- Built-in Features
    - Clipping
    - Global Illumination FX
        - Parameters and Samplers Naming Convention
        - Example
    - Razor Font Rendering
        - Example
    - Skinning
    - Vertex Colors
    - HDR
        - Requirement
        - Linearization

Copyright © 2026 Vizrt

- Texture Slot
    - Usage
- ControlImage

Viz Engine Shaders are a novel way to create custom materials for the Viz Engine Renderer. Only basic knowledge of how to write OpenGL Shaders is necessary to be able to integrate the material of your needs. Viz Engine provides multiple template Shaders, so that only the most important parts of a material need to be implemented. The common and repetitive steps are taken by the template itself.

With the use of Viz Engine Shaders, designers can achieve outstanding effects. Viz Engine Shaders are executed purely on the GPU. They can be used to create photo realistic effects or you can use them as a backdrop on Videowall instead of utilizing Ultra HD clips.

Viz Engine Shaders can be developed either by using the Viz plug-in or within Viz Artist using the Shader Design Environment.

The following chapters shall give you a starting point on how to work with Viz Engine Shaders and inspire your creativity.

5.13.1 Introduction to Viz Engine Shaders

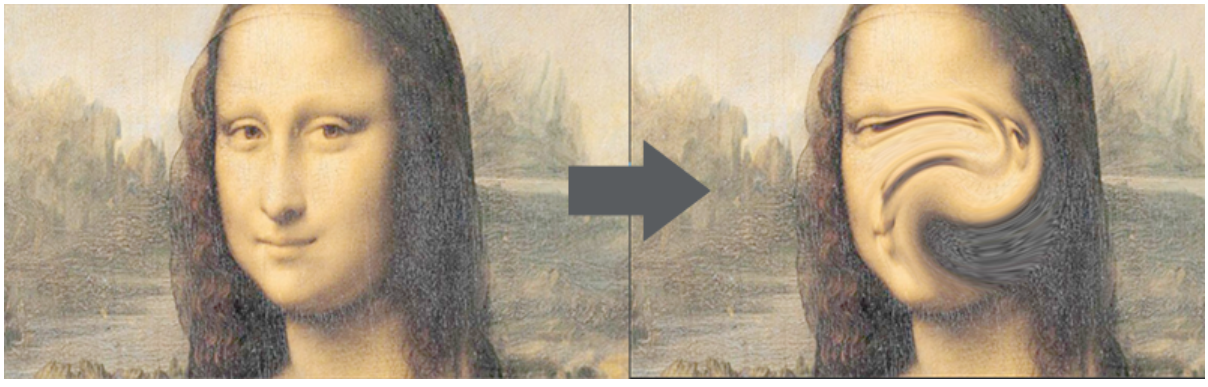
There are three types of Shaders available:

- Fragment Shaders
- Fragment and Vertex Shaders
- Transition Shaders

While Fragment Shaders and transition shader modify pixel data only, Vertex Shaders can additionally modify vertex (geometry) data. A typical vertex shader is for example a bump mapping effect.

The following example explains how to create a simple fragment shader with some exposed parameters.

Let's say you've found some nice twirl shader on sites like shadertoy.com. We will now explain step by step how to create a Viz Engine Shader out of it.



The original code for this shader looks like this:

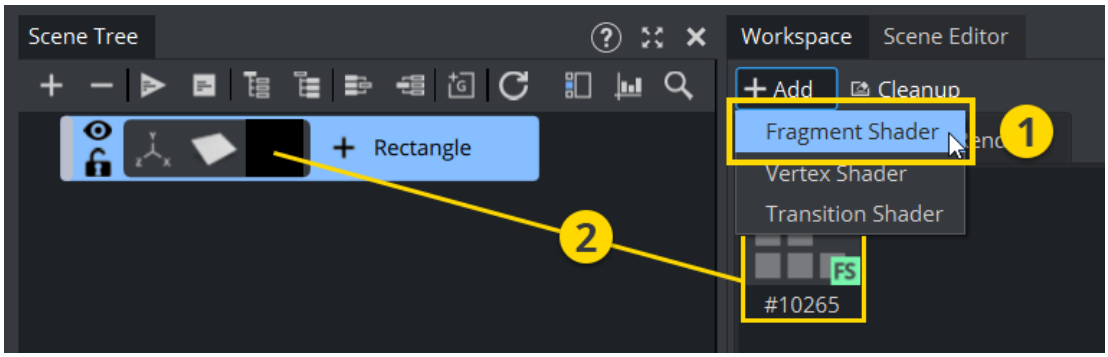
Copyright © 2026 Vizrt

Minimal Fragment Shader

```
#define PI 3.14159

void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    float effectRadius = .5;
    float effectAngle = 1. * PI;

    vec2 center = iMouse.xy / iResolution.xy;
    center = center == vec2(0., 0.) ? vec2(.5, .5) : center;

    vec2 uv = fragCoord.xy / iResolution.xy - center;

    float len = length(uv * vec2(iResolution.x / iResolution.y, 1.));
    float angle = atan(uv.y, uv.x) + effectAngle * smoothstep(effectRadius, 0., len);
    float radius = length(uv);

    fragColor = texture(iChannel0, vec2(radius * cos(angle), radius * sin(angle)) +
center);
}
```

Create a new scene and add a rectangle to it.

Next step is to switch to the Scripting Workspace. In the workspace, select **+Add** and **Fragment Viz Engine Shader**. A new Shader appears as Fragment Shader.



Drag the Shader to the container. the Shader Design Environment is activated. As we are creating a Fragment Shader only, only this tab gets activated. A Vertex Fragment Viz Engine Shader would activate both tabs, Fragment and Vertex Shader.

Now copy and paste the shader code into the Script Editor. As we need to link some properties between the Shader and Viz Engine, some adjustments are needed for our code. Viz Engine calls `fragmentMain()` on a Fragment shader, therefore we call the existing mainImage code from within this function.

Copyright © 2026 Vizrt

Fragment Shader

```
#define iResolution vec2(1920,1080)
#define PI 3.14159265359

void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    float effectRadius = .5;
    float effectAngle = 1. * PI;

    vec2 center = iMouse.xy / iResolution.xy;
    center = center == vec2(0., 0.) ? vec2(.5, .5) : center;

    vec2 uv = fragCoord.xy / iResolution.xy - center;

    float len = length(uv * vec2(iResolution.x / iResolution.y, 1.));
    float angle = atan(uv.y, uv.x) + effectAngle * smoothstep(effectRadius, 0., len);
    float radius = length(uv);

    fragColor = texture(iChannel0, vec2(radius * cos(angle), radius * sin(angle)) +
center);
}

void fragmentMain()
{
    vec4 fragColor = vec4(0);
    vec2 fragCoord = viz_getTexCoord() * iResolution;
    mainImage(fragColor, fragCoord);
    vec3 col = fragColor.xyz;
    viz_setFragment(vec4(col, 1.0));
}
```

Hitting the **Compile and Run** function still gives some errors. We need to replace the *iMouse* variable with a static variable. vec2(0.0,0.0) as we do not want to interact with the user.

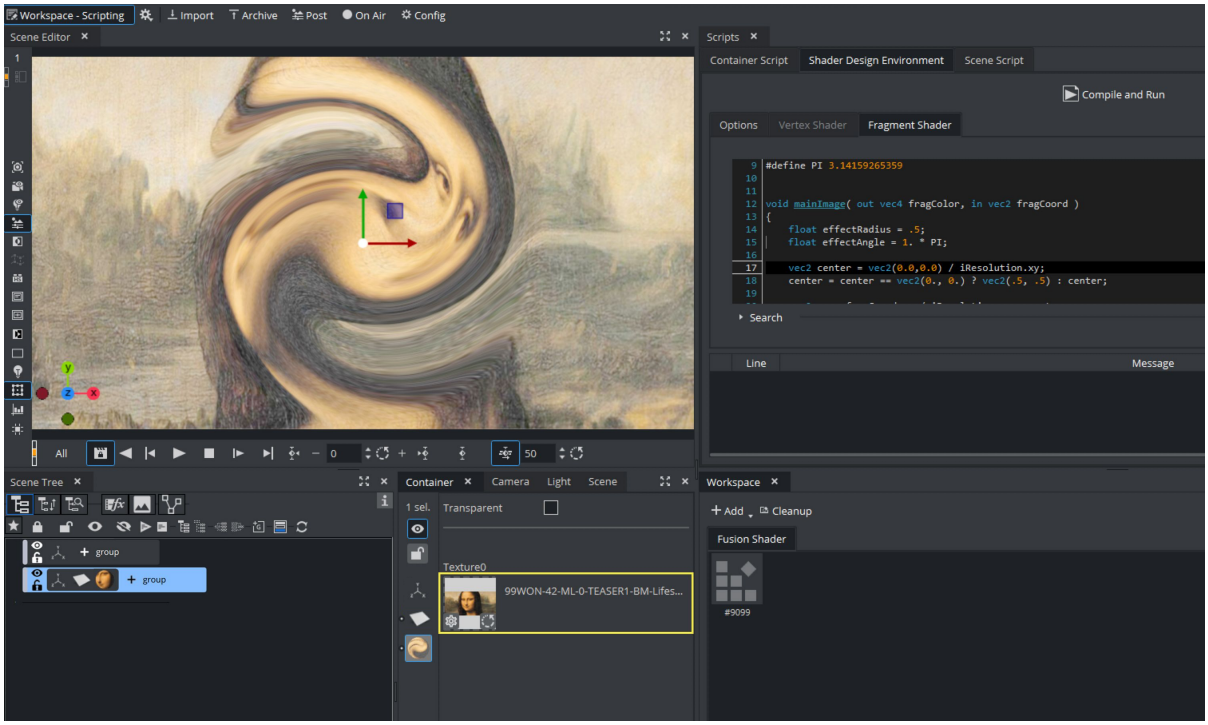
Finally the texture, defined in the original code *iChannel0*, needs to be replaced by a function that provides a Viz Engine texture. Expose this as a property the user can change:

Fragment Shader

```
@registerParametersBegin
@registerParameterSampler2D(iChannel0, "Texture0")
@registerParametersEnd
```

This overrides the iChannel0 with a userParameter. After a **Compile and Run**, we now have a property available to the user. Drag and drop a texture on it and you'll already see some effects:

Copyright © 2026 Vizrt



Now if you change some parameters like the *effectRadius* or *effectAngle* and click on the **Compile and Run** button, the shader behaves differently.

```
float effectRadius = .5;
float effectAngle = 1. * PI;
```

It would be nice, if we can also expose those settings to the user. We extend our code so that we use more parameters to set our variables. Finally, to access the values of this parameters, the lines where those variables are used need to be slightly changed to read the actual values.

Our final code looks like this:

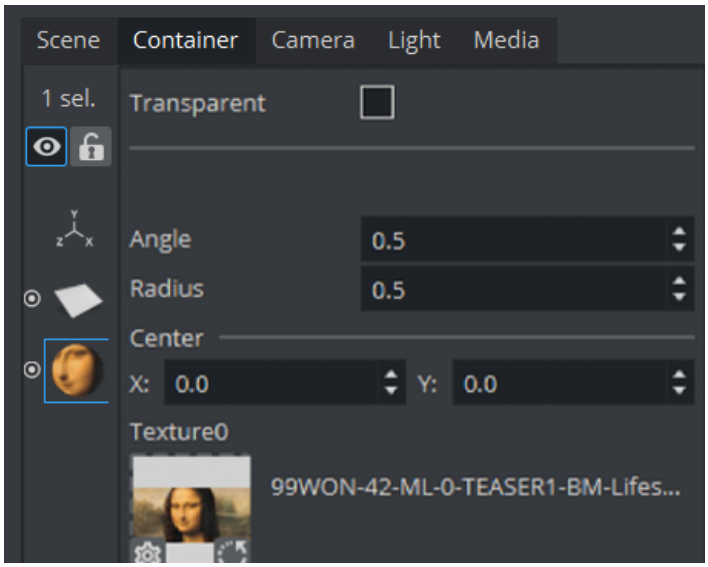
Fragment Shader

```
@registerParametersBegin
@registerParameterSampler2D(iChannel0, "Texture0")
@registerParameterFloat(effectAngle, "Angle", 0.5f, -5.f, 15.f)
@registerParameterFloat(effectRadius, "Radius", 0.5f, -5.f, 15.f)
@registerParameterVec2(myCenter, "Center", 0.0, 0.0)
@registerParametersEnd

#define iResolution vec2(1920,1080)
#define PI 3.14159265359

void mainImage( out vec4 fragColor, in vec2 fragCoord )
{

    vec2 center = vec2(0.0,0.0) / iResolution.xy;
```

Copyright © 2026 Vizrt

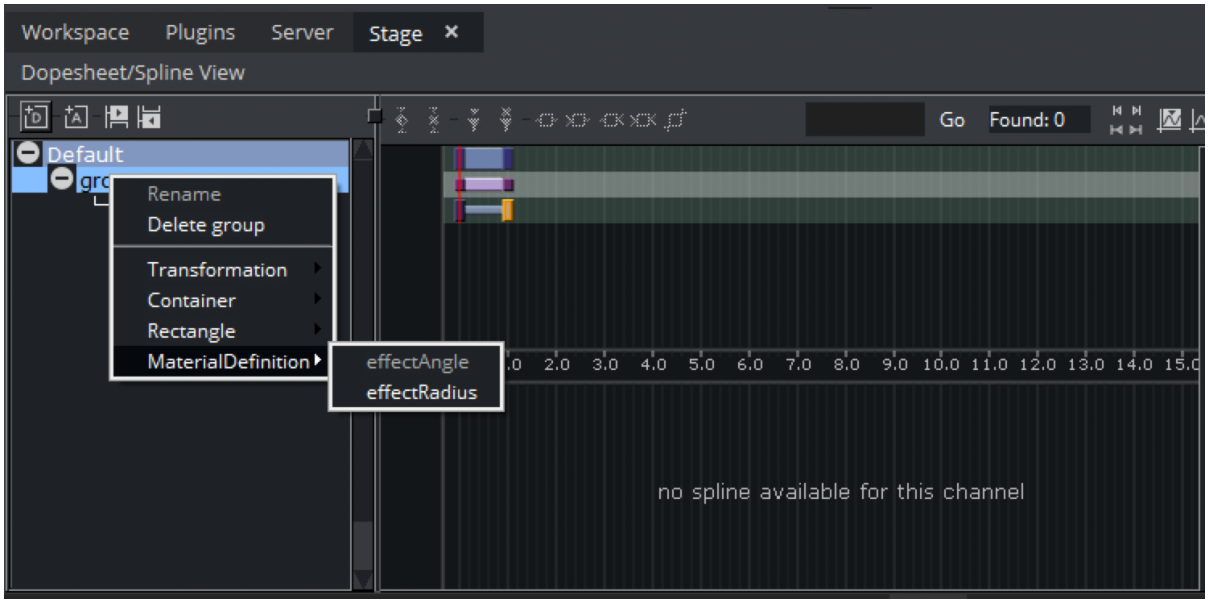
```
    center = center == vec2(0., 0.) ? vec2(.5, .5) : center;

    vec2 uv = fragCoord.xy / iResolution.xy - center;

    float len = length(uv * vec2(iResolution.x / iResolution.y, 1.));
    float angle = atan(uv.y, uv.x) + userParametersFS.effectAngle* PI *
smoothstep(userParametersFS.effectRadius, 0., len);
    float radius = length(uv);

    fragColor = texture(iChannel0, vec2(radius * cos(angle), radius * sin(angle)) +
center);
}

void fragmentMain()
{
    vec4 fragColor = vec4(0);
    vec2 fragCoord = viz_getTexCoord() * iResolution;
    mainImage(fragColor, fragCoord);
    vec3 col = fragColor.xyz;
    viz_setFragment(vec4(col, 1.0));
}
```

The UI now has been extended to show these values:



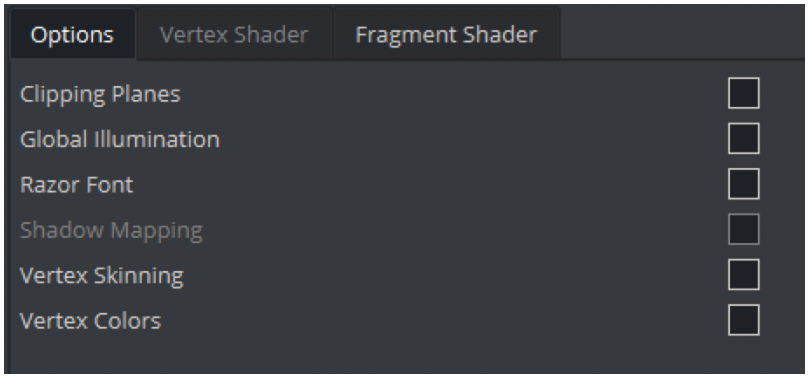
Changing these values immediately changes the way how the shader affects our image. Instead of an image, you can also apply this effect to any Media Asset like Live Sources or Clip Surfaces. These values are also fully animatable.

Copyright © 2026 Vizrt



Tip: Viz Engine Shaders can be saved into Graphic Hub. Drag and drop them from the Workspace window into the Graphic Hub Server view.

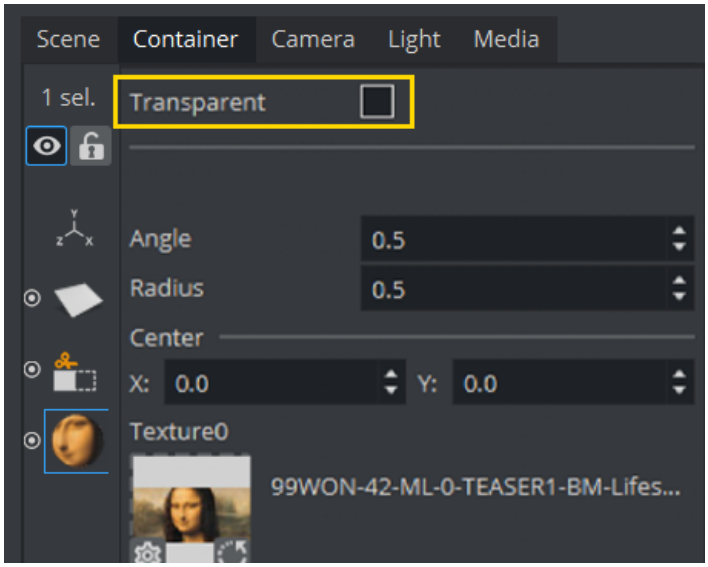
5.13.2 Properties for Viz Engine Shaders



Viz Engine Shaders can integrate into Viz Engine design workflows by exposing some parameters to the user. This enables certain built-in render features inside the Viz Engine Shader. These properties can be found in the Options tab.

Transparency is part of the Viz Engine Shaders User Interface:

Copyright © 2026 Vizrt



5.13.3 Viz Engine Fragment Shader



The Viz Engine Fragment Shader allows it to write a fragment shader only, without the need to write a vertex shader. The vertex shader is provided by Viz Engine and provides the fragment shader API with all standard information available.

The main function for the Viz Engine Fragment Shader needs to be called `fragmentMain`. To set the color of a fragment, `viz_setFragment` needs to be called.

Minimal Fragment Shader

```
void fragmentMain()
{
   viz_setFragment(vec4(1,0,0,1));
}
```

Fragment Shader API

The template shader provides an additional API to get access to the properties of Viz Engine, lights, geometry, etc..

Geometry API Functions

Function Name	Description
viz_getPosition	Returns the current worldspace position as `vec3`.

Copyright © 2026 Vizrt

Function Name	Description
viz_getPositionCS	Returns the current clipspace position as `vec4`.
viz_getNormal	Returns the current worldspace normal as `vec3`.
viz_getTangent	Returns the current worldspace tangent as `vec3`.
viz_getBinormal	Returns the current worldspace binormal as `vec3`.
viz_getTexCoord	Returns the texture coordinates as `vec2`.
viz_getColor	Returns the color as `vec4`.
viz_getGlyphAlpha	Returns the glyph alpha for razor text as `float`.

Environment Map API Functions

Function Name	Description
viz_getEnvironmentMapRotation	Returns the scenes environment map rotation as `float`.
viz_getEnvironmentMapIntensity	Returns the scenes environment map intensity as `float`.

Rendering Globals API Functions

Function Name	Description
viz_getAlpha	Returns the current alpha value as `float`.
viz_getRetraceCounter	Returns the current retrace counter value as `uint`.
viz_getTime	Returns the current time in seconds as `double`. ⚠ **Note:** The time resets to 0 after 12 hours (43200 seconds) as floating point operations are unreliable after such a long period.

Copyright © 2026 Vizrt

Function Name	Description
viz_getCameraPosition	Returns the worldspace position of the current camera as `vec3`.
viz_getViewDirection	Returns the normalized view vector as `vec3`.

Container Lights API Functions

Function Name	Description	Supported Lights
viz_getPointLightCount	Returns the number of pointlights as `int`.	
viz_getPointLightIndex	Returns the global light index of a pointlight as `int`.	
viz_getDirectionalLightCount	Returns the number of directional lights as `int`.	
viz_getDirectionalLightIndex	Returns the global light index of a directional light as `int`.	
viz_getSpotLightCount	Returns the number of spot lights as `int`.	
viz_getSpotLightIndex	Returns the global light index of a spotlight as `int`.	
viz_getAreaLightCount	Returns the number of area lights as `int`.	
viz_getAreaLightIndex	Returns the global light index of an area light `as int`.	
viz_getLightPosition	Returns the worldspace position of a light as `vec3`.	Point, Spot, Area
viz_getLightDirection	Returns the normalized worldspace direction of a light as `vec3`.	Directional, Spot, Area

Copyright © 2026 Vizrt

Function Name	Description	Supported Lights
viz_getLightUp	Returns the normalized worlspace up-vector of a light as `vec3`.	Area
viz_getLightRight	Returns the normalized right vector of a light as `vec3`.	Area
viz_getLightColor	Returns the color of a light as `vec3`.	All
viz_getLightIntensity	Returns the intensity scale factor of a light as `float`.	All
viz_getLightDiffuseIntensity	Returns the diffuse intensity scale factor of a light as `float`.	All
viz_getLightSpecularIntensity	Returns the specular intensity scale factor of a light as `float`.	All
viz_getLightRadius	Returns the radius of a light as `float`.	Point, Spot
viz_getLightInnerConeAngle	Returns the inner cone angle in radians of a light as `float`.	Spot
viz_getLightOuterConeAngle	Returns the outer cone angle in radians of a light as `float`.	Spot
viz_getLightWidth	Returns the width of a light as `float`.	Area
viz_getLightHeight	Returns the height of a light as `float`.	Area

To access light properties, a global light index is needed. So a loop over all available point lights would look like:

Light Iteration Example

```
int pointLightCount = viz_getPointLightCount();
for (int i=0; i<pointLightCount; ++i)
{
    int lightIndex = viz_getPointLightIndex(i);
    vec3 lightColor = viz_getLightColor(lightIndex);
}
```

Copyright © 2026 Vizrt

Global Illumination FX API Functions

Function Name	Description
viz_getGlobalIlluminationIrradiance	Returns the irradiance color as `vec3`.
viz_getGlobalIlluminationAmbientOcclusion	Returns the ambient occlusion factor as `float`.
viz_getGlobalIlluminationBakeLight	Returns the color of the baked lights as `vec3`.

5.13.4 Viz Engine Vertex and Fragment Shader



The Viz Engine Vertex/Fragment Shader allows it to write both the vertex and the fragment shader. In addition to the fragment shader API from above, the vertex shader template gives access to geometry vertex data and allows to override vertex positions, normals, etc..

The main function for the Vertex Shader needs to be called `vertexMain`. Multiple vertex attributes are available to be set by the vertex shader via API functions, the most important one is `viz_setVertex.`

Minimal Vertex Shader

```
void vertexMain()
{
   vec3 pos = viz_getVertex();
   viz_setVertex(pos);
}
```

Vertex Shader API

Geometry API Functions

Function Name	Description
viz_getVertex	Returns the current vertex as `vec3.`
viz_getNormal	Returns the current normal as `vec3.`
viz_getTangent	Returns the current tangent as `vec3.`

Copyright © 2026 Vizrt

Function Name	Description
viz_getBinormal	Returns the current binormal as `vec3.`
viz_getTexCoord	Returns the current texture coordinate as `vec2.`
viz_getColor	Returns the current color as `vec4.`
viz_setVertex	Sets the current vertex as `vec3.`
viz_setNormal	Sets the current normal as `vec3`.
viz_setNormals	Sets the three normals (normal, tangent, binormal) as `vec3`'s.
viz_setNormals	Sets the normal and tangent, binormal is calculated automatically as `vec3`'s.
viz_setTexCoord	Sets the texture coordinates as `vec2.`
viz_setColor	Sets the color as `vec4`.

All `viz_get` functions return their values as they are provided by the geometry, they are not altered.

All `viz_set` functions assume the values are passed in the same space, too. Afterwards, the values are multiplied by the geometry's world space matrix and passed to the fragment shader. The vertex position itself gets multiplied by the model-view-projection matrix and passed as `gl_Position`.

In the case of vertex skinning, the vertex and normals get the skinning transformation applied before being passed to the fragment shader and `gl_Position`.

Rendering Globals API Functions

Function Name	Description
viz_getAlpha	Returns the current alpha value as `float.`
viz_getRetraceCounter	Returns the current retrace counter value as `uint`.
viz_getTime	Returns the current time in seconds as `double`.

Copyright © 2026 Vizrt

Environment Map API

Function Name	Description
viz_getEnvironmentMapRotation	Returns the scenes environment map rotation as `float`.
viz_getEnvironmentMapIntensity	Returns the scenes environment map intensity as `float`

Passing Values from Vertex to Fragment Shader

Passing data between different shader stages is also supported. In GLSL, `out` and `in` need to be used, but to ensure that the data passed by the user does not conflict with the data passed by the template shader, new keywords were added: `varying_out` and `varying_in.`

Viz GLSL Extension

Viz Keyword	Description
`@varying_out(location)`	Declares an `out` value at the given location. `location` must be a number starting from `0`.
`@varying_in(location)`	Declares an `in` value at the given location. `location` must be a number starting from `0`.

Example

Vertex Shader

```
@varying_out(0) vec4 vColor;

void vertexMain()
{
    vColor = vec4(1,1,0,0);
}
```

Fragment Shader

```
@varying_in(0) vec4 vColor;

void fragmentMain()
```

Copyright © 2026 Vizrt

```
{
    viz_setFragment(vColor);
}
```

5.13.5 User Parameters

In GLSL, shader parameters are usually passed using uniforms or uniform buffers, textures most of the time as `sampler2d`. For a better Command Map and Artist integration and forward compatibility, new keywords were added, though. Each shader can have its own, but only one set of user parameters. All parameters need to be declared within a block that starts with `@registerParametersBegin` and end with `@registerParametersEnd`.

Viz GLSL Extensions

Viz Keyword	Description	Arguments
`@registerParametersBegin`	Marks the start of the user parameters block.	
`@registerParametersEnd`	Marks the end of the user parameters block.	
`@registerParameterInt(paramInt, "Int Label", 1, -1, 1)`	Register an `int` parameter.	`name, label, default, min value, max value`
`@registerParameterFloat(paramFloat, "Float Label", 1.0, -1.0, 1.0)`	Register a `float` parameter.	`name, label, default, min value, max value`
`@registerParameterVec2(paramVec2, "Vec2 Label", -1.2, 2.2)`	Register a `vec2` parameter.	`name, label, default x, default y`
`@registerParameterVec3(paramVec3, "Vec3 Label", 1.3, -2.3, 3.3)`	Register a `vec3` parameter.	`name, label, default x, default y, default z`
`@registerParameterVec4(paramVec4, "Vec4 Label", 1.4, 2.4, -2.4, 3.4)`	Register a `vec4` parameter.	`name, label, default x, default y, default z, default w`

Copyright © 2026 Vizrt

Viz Keyword	Description	Arguments
`@registerParameterColor(paramColor, "Color Label", 0, 0, 0, 255)`	Register a `vec4` parameter exposed as a color.	`name, label, defaultr, default g, default b, default a`
`@registerParameterSampler2D(textureA, "Color Texture")`	Register a `sampler2d` parameter.	`name, label`

All parameters except `sampler2d` are managed by a single uniform buffer for each shader stage. To access them, the uniform buffer name inside the vertex shader is `userParametersVS` and in the fragment shader `userParametersFS`.

Examples

Vertex Shader with Parameter Block

```
@registerParametersBegin
@registerParameterInt(paramInt, "Int Label", 1, -1, 1)
@registerParameterFloat(paramFloat, "Float Label", 1.0, -1.0, 1.0)
@registerParameterVec2(paramVec2, "Vec2 Label", -1.2, 2.2)
@registerParameterVec3(paramVec3, "Vec3 Label", 1.3, -2.3, 3.3)
@registerParameterVec4(paramVec4, "Vec4 Label", 1.4, 2.4, -2.4, 3.4)
@registerParameterColor(paramColor, "Color Label", 0, 0, 0, 255)
@registerParametersEnd

void vertexMain()
{
   int intValue = userParametersVS.paramInt;
   float floatValue = userParametersVS.paramFloat;
   vec2 vec2Value = userParametersVS.paramVec2;
   vec3 vec3Value = userParametersVS.paramVec3;
   vec4 vec4Value = userParametersVS.paramVec4;
   vec4 colorValue = userParametersVS.paramColor;

   viz_setColor(colorValue);
}
```

Fragment Shader with Parameter Block

```
@registerParametersBegin
@registerParameterFloat(intensity, "Intensity", 1.0, 0.0, 1.0)
@registerParameterSampler2D(inputTexture,"Texture Label")
@registerParametersEnd
```

Copyright © 2026 Vizrt

```
void fragmentMain()
{
    vec4 textureColor = texture(inputTexture, viz_getTexCoord());
    vec4 vertexColor = viz_getColor();
    vec4 finalColor = textureColor * vertexColor * userParametersFS.intensity;
    viz_setFragment(finalColor);
}
```

Environment Map

It's possible to gain automatic access to a scenes global environment map by declaring a two dimensional texture sampler named `EnvironmentMap` or `environmentMap`.

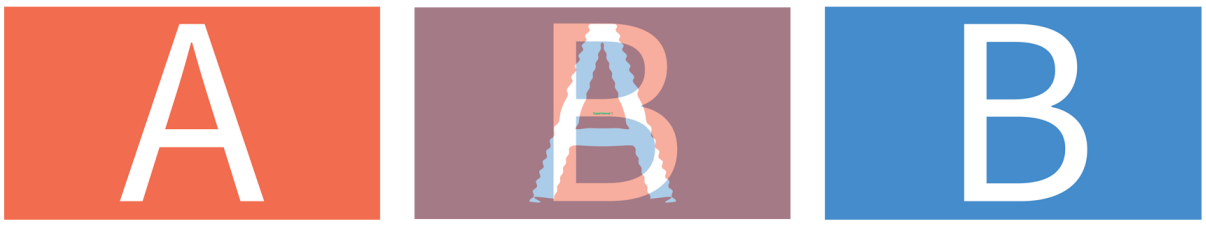
```
@registerParameterSampler2D(EnvironmentMap, "Environment Map")
```

5.13.6 Transition Shaders



Transition Shaders are utilized by Super Channels to allow special transitions from Asset A to B. Whereas classical Super Channel transitions could only animate a subchannels position, scale and alpha value, Transition Shaders can be used to create very complex pixel based transitions from a source image (A) to a destination image (B). Transition Shaders can also be parameterized.

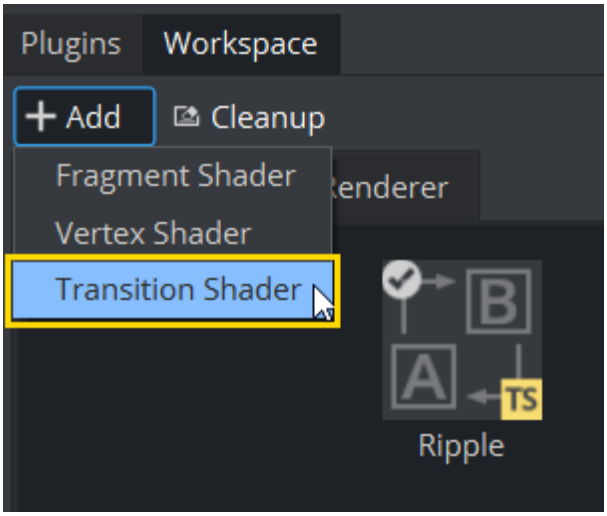
Example



As a transition example we will create a Ripple effect with parameters:

1. First add a Super Channel to your scene and add two images as a source into the Sub Channel A and Sub Channel B slot (these can be also other scenes, live sources, etc..). These textures are available in the Transition Shader as `viz_textureA()` and `viz_textureB()` functions.
2. Open the workspace panel and create a new Transition Shader, give it a unique name:

Copyright © 2026 Vizrt



3. Double click on the Shader to open the scripting window (if the window does not open, make sure it is selected in the Available Docks menu).

The Transition Shader code:

```
@registerParametersBegin
@registerParameterFloat(amplitude, "Amplitude", 100.f, 0.f, 500.f)
@registerParameterFloat(speed, "Speed", 50.f, 0.f, 1000.f)
@registerParametersEnd

float amplitude = userParametersFS.amplitude;
float speed = userParametersFS.speed;

const float PI = 3.141592653589793;
float progress = viz_getTransitionTime();

vec4 getFromColor(vec2 op) {
return viz_textureA(op);
}

vec4 getToColor(vec2 op) {
return viz_textureB(op);
}

vec4 transition (vec2 uv) {
vec2 dir = uv - vec2(.5);
float dist = length(dir);
vec2 offset = dir * (sin(progress * dist * amplitude - progress * speed) + .5) / 30.;
return mix(
getFromColor(uv + offset),
getToColor(uv),
smoothstep(0.2, 1.0, progress)
);
}

void fragmentMain() {
```

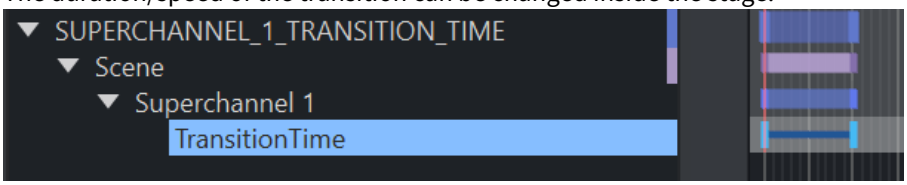
Copyright © 2026 Vizrt

```
vec2 op = viz_getTexCoord();
vec4 result = transition(op);
viz_setFragment(result);
}
```

4. Copy paste the code above into the editor and hit **Compile and Run**.

Important Parts of the Code

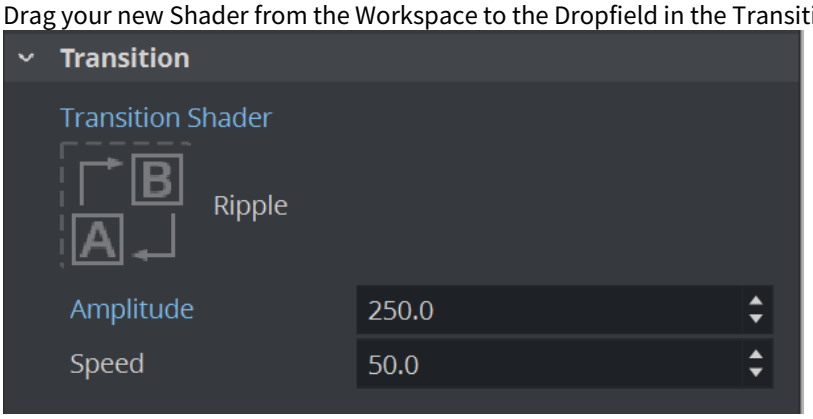
- The first 4 lines define parameters exposed into the User Interface to manipulate the look of our transition.
- The time is used by calling `viz_getTransitionTime();`
  The duration/speed of the transition can be changed inside the stage:



- The calculation is done in the `transition()` function.
- `getFrom` and `getToColor` are wrappers used to read the pixels from the source and destination image, using `viz_textureA` and `viz_textureB`

To Use the Transition Shader

1. Click on **Media**, your **Super Channel** and choose **Super Channel** > **Content.**
2. Drag your new Shader from the Workspace to the Dropfield in the Transition Area.



> **Note:** The two additional fields **Amplitude** and **Speed**, are the exposed parameters defined in the beginning of our script.

3. Once the shader is assigned to the Super Channel as a transition shader, a new director with the name *SUPERCHANNEL_i_TRANSITION_TIME* (where *i* is the Super Channel index) is automatically created to animate the transition time.
4. Now Click on **A** or **B** to see the Transition Shader in Action.

Copyright © 2026 Vizrt

Note: The shader can always be updated in the scripting window. **Compile and Run** should be clicked afterwards to update the transition as well.

Transition Shader API

Like Fragment Shaders, the main function for the Viz Engine Transition Shader is `fragmentMain`, and the fragment color is set by `viz_setFragment.`

Transition Shaders inherit some of the Fragment Shaders geometry functions. These are:

Geometry API Functions

Function Name	Description
viz_getPosition	Returns the current worldspace position as `vec3`.
viz_getPositionCS	Returns the current clipspace position as `vec4`.
viz_getTexCoord	Returns the texture coordinates as `vec2`.
viz_getColor	Returns the color as `vec4`.

In addition, Transition Shaders have three more functions for transition.

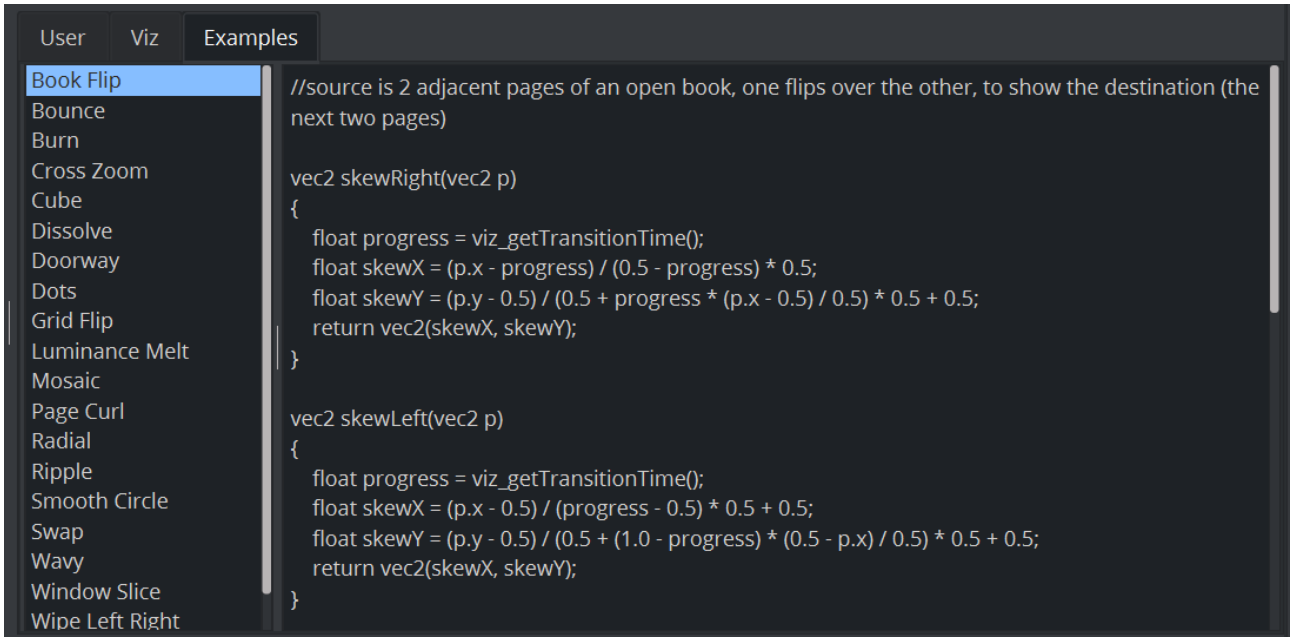
Transition API Functions

Function Name	Description
viz_textureA	Returns the color sample in texture A (at specific texture coordinates) as `vec4`.
viz_textureB	Returns the color sample in texture B (at specific texture coordinates) as `vec4`.
viz_getTransitionTime	Returns the current transition time as `float`.

5.13.7 Shader Samples Library

As a starting point, Viz Engine provides some examples for beginners. You can explore them in the **Shader Design Environment > Examples**, or **Transition Shader > Examples** tab, depending on the shader type. The following figure shows the transition shader samples:

Copyright © 2026 Vizrt



Copy and paste the sample code into the editor window.

5.13.8 Built-in Features

All Viz Engine Shader templates have built-in features that can be enabled separately. Enabling a feature generates multiple variants of the shader and the best fitting one is selected to render. Keep in mind that the more features enabled, the more combinations of Shaders need to be generated and compiled. This can drastically increase the load time of a scene and the performance impact of a material. Only the features that are really necessary should be enabled.

Clipping

Enables support for the scenes `Clipper` objects. The template shader takes care of discarding the elements that are clipped.

> ⚠ **Note:** This feature makes use of the `discard` keyword, which can have a negative impact on performance.

Global Illumination FX

Enables support for Global Illumination FX. The Global Illumination FX API functions should be used to retrieve the computed GI values.

Parameters and Samplers Naming Convention

The Global Illumination system needs material colors and textures to be able to generate realistic indirect lighting. Therefor parameters should be named according the following convention:

Copyright © 2026 Vizrt

Name	Example	Description
AlbedoMap	`@registerParameterSampler2D(AlbedoMap, "Albedo Texture")`	GI takes the albedo color from this texture. Also called base color sometimes. If no AlbedoMap is defined, it's assumed to be full white as in vec4(1,1,1,1)
AlbedoColor	`@registerParameterColor(AlbedoColor, "Albedo Color", 255, 255, 255, 255)`	GI takes the albedo color as a plain color. It's multiplied with the color from the texture. If no AlbedoColor is defined, it's assumed to be full white as in `vec4(1,1,1,1)`
Emissive Map	`@registerParameterSampler2D(EmissiveMap, "Emissive Texture")`	GI takes the emissive color from this texture, alpha value is ignored. If no EmissiveMap is defined, it's assumed to be full black as in `vec4(0,0,0,0)`
Emissive Color	`@registerParameterColor(EmissiveColor, "Emissive Color", 0, 0, 0, 0)`	GI takes the emissive color as a plain color and adds it to the color from the texture, alpha value is ignored. If no EmissiveColor is defined, it's assumed to be full black as in `vec4(0,0,0,0)`
Emissive Intensity	`@registerParameterFloat(EmissiveIntensity, "Emissive Intensity", 1.f, 0.f, 100.f)`	Scale factor on the sum of both emissive texture and color values. If no EmissiveIntensity is defined, it's assumed to be `1.0`

Example

Fragment Shader with GI Support

```
@registerParametersBegin
@registerParameterColor(AlbedoColor, "Albedo Color", 255, 255, 255, 255)
@registerParameterSampler2D(AlbedoMap, "Albedo Texture")
```

```
@registerParameterColor(EmissiveColor, "Emissive Color", 0, 0, 0, 0)
@registerParameterFloat(EmissiveIntensity, "Emissive Intensity", 1.f, 1.f, 100.f)
@registerParameterSampler2D(EmissiveMap, "Emissive Texture")
@registerParametersEnd

void fragmentMain()
{
    vec4 albedoColor = texture(AlbedoMap, viz_getTexCoord()) *
userParametersFS.AlbedoColor;
    vec3 emissiveColor = (texture(EmissiveMap, viz_getTexCoord()).rgb +
userParametersFS.EmissiveColor.rgb) * userParametersFS.EmissiveIntensity;

    vec3 color = albedoColor.rgb + emissiveColor;
    float alpha = albedoColor.a * viz_getAlpha();

    color += viz_getGlobalIlluminationIrradiance(viz_getNormal(), albedoColor.rgb);
    color *= viz_getGlobalIlluminationAmbientOcclusion();

    viz_setFragment(vec4(color, alpha));
}
```

Razor Font Rendering

Enables support for Razor Font rendering. Use `viz_getGlyphAlpha` to get the alpha channel value for the font.

Example

Fragment Shader with Razor Font Support

```
@registerParametersBegin
@registerParameterColor(textColor,"Text Color", 255, 255, 255, 255)
@registerParametersEnd

void fragmentMain()
{
    float glyphAlpha = viz_getGlyphAlpha();
    float alpha = viz_getAlpha();

    vec4 textColor = userParametersFS.textColor;
    textColor.a *= alpha;
    textColor.a *= glyphAlpha;

    viz_setFragment(textColor);
}
```

Copyright © 2026 Vizrt

Skinning

Enables support for Vertex Skinning. There's nothing that needs to be done, the vertex shader template takes care of transforming the necessary vertex attributes.

Vertex Colors

Enables support for Vertex Colors. If Vertex Colors are enabled `viz_getColor` returns the vertex color, full white otherwise.

HDR

Both Fragment and Vertex/Fragment shaders support color linearization and automatic color space conversion.

Requirement

A **sampler** named `BaseColor` must be defined and accessible in the shader.

```
@registerParameterSampler2D(BaseColor, "BaseColor")
```

Linearization

`viz_linearizeBaseColor` is a shader api function that converts gamma-encoded values into linear space based on the texture being used. The function expects a `vec4` input and returns a `vec4` with the RGB channels linearized. The alpha channel is preserved.

Usage Example

```
vec4 baseColor = texture(BaseColor, texCoord);
vec4 linearColor = viz_linearizeBaseColor(baseColor);
```

Texture Slot

Both Fragment and Vertex/Fragment shaders have access to texture slot blending.

Usage

`viz_applyTextureSlotBaseColor` is a shader plug-in function that applies the texture slots with the target “BaseColor” to a given color. The function expects a `vec4` input and returns a `vec4` with the texture slot blending operation result.

Copyright © 2026 Vizrt

Example

```
vec4 baseColor = texture(BaseColor, texCoord);
vec4 linearColor = viz_linearizeBaseColor(baseColor);
vec4 blendedColor = viz_applyTextureSlotBaseColor(linearColor);
```

ControlImage

ControlImage enables dynamic sampler changes on runtime materials. To utilize this feature:

- The sampler must be named **BaseColor**.
- The **Set MaterialDefinition BaseColor** parameter in ControlImage must be enabled.

Copyright © 2026 Vizrt

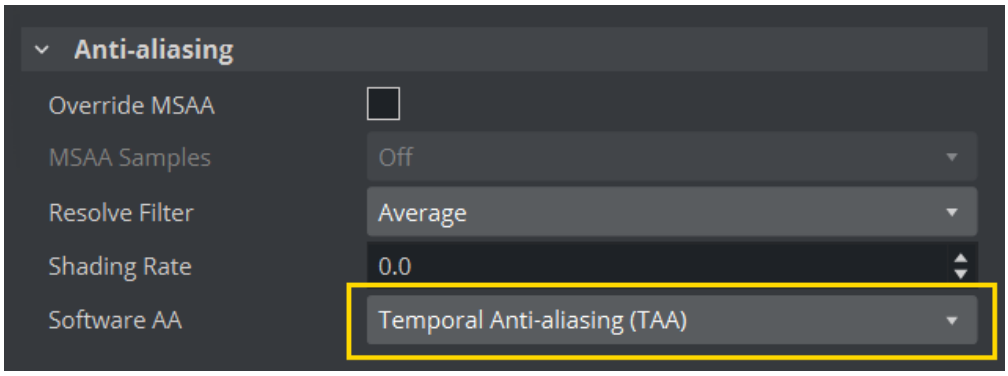
5.14 To Improve Render Quality

The following guides you on how to improve the quality of your rendering by tweaking different settings.

5.14.1 Anti-aliasing

Anti-aliasing is a method to smoothen the edges of geometries. This is usually done by the GPU driver by analyzing the 3D model (spatial anti-aliasing). 4x anti-aliasing as a default setting (4xMS, four times multi-sampling) is recommended. This hardware only anti-aliasing might not be sufficient for some graphics. The Viz Engine Render Pipeline in Viz Engine allows more efficient methods to get rid of the unwanted artifacts.

To modify your anti-aliasing settings, adapt your settings in the **Scene** > **Rendering** Menu:



Resolve Filter

This option requires the engine to be configured with Multi-Sampled rendering. The **Resolve Filter** defines the way multiple color samples are combined into a single one after scene rendering. The available options are:

Screenshot	Resolve Filter	Description
	Average	The *Average* is a standard box filter, that combines all samples and weights them equally. The samples that are included are the ones that were rendered for the current pixel.

Copyright © 2026 Vizrt

Screenshot	Resolve Filter	Description
	Triangle	The *Triangle* filter weights the samples using a triangle function. It includes not only samples from its own pixel, but all samples that are within a radius of 1. While it helps reducing aliasing artifacts, the resulting image looks a bit blurrier than using the *Average* option.

Shading Rate

This option requires Viz Engine to be configured with Multi-Sampled rendering. It specifies the percentage of samples which get shaded for each covered pixel. A value of `0` means only one sample is shaded per pixel, `100` means that all samples are shaded. Any value between 0 and 100 results in a subset of samples being shaded.

Increasing the shading rate can greatly reduce specular aliasing artifacts, but depends on the scene complexity and may have an impact on performance.

Screenshot	Shading Rate
	*Shading Rate* set to 0.
	*Shading Rate* set to 50.

Copyright © 2026 Vizrt

Screenshot	Shading Rate
	*Shading Rate* set to 100.

Software Anti-Aliasing

FXAA - Fast Approximate Anti-aliasing

This very efficient method does not require a lot of computing power. It removes unwanted jagged edges as a post process effect on the final rendered images. As spatial anti-aliasing usually only affects edges, FXAA also tries to smooth pixel artifacts within rendered surfaces.

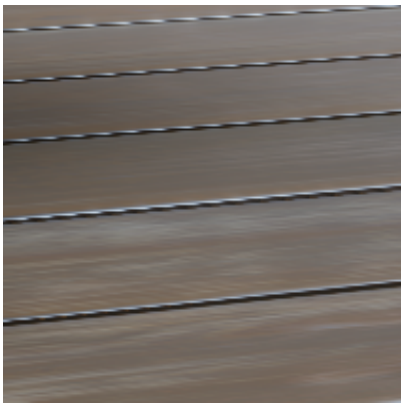
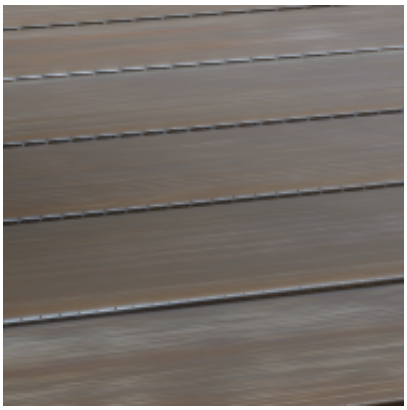
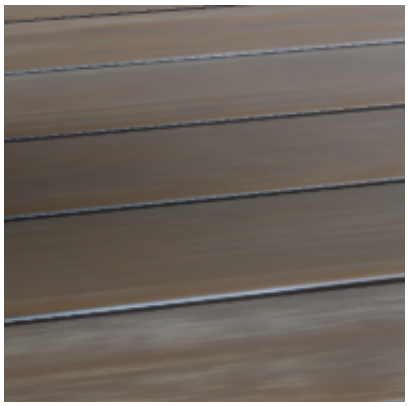
Temporal Anti-aliasing

Temporal anti-aliasing (TAA) is a SAA technique that used multiple frames to generate a very high quality images. Pixels sampled in past frames are blended with pixels sampled in the current frame to generate the final image. With this approach, it might look that the effect jumps in.

The number of frames, configured via **Frame Contribution**, used to calculate the effect can also be modified. A number too high results in a flickering effect. It is recommended to use a value between 3-10.

If the flickering is still shown, **ClipBox Scaling** value can be further adjusted in the range from 50% to 150%. The higher the value is, the more the flicking is suppressed. However, increasing this leads to more-apparent ghosting effect when the camera or the geometry is moved.

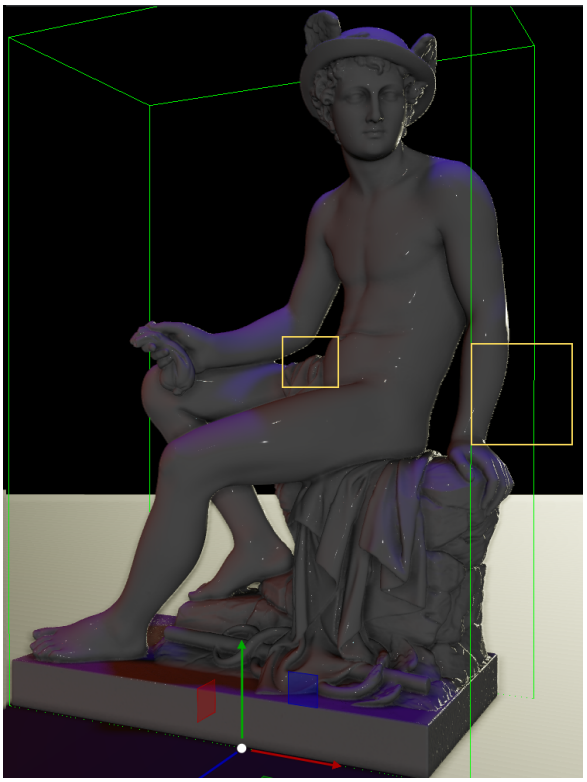
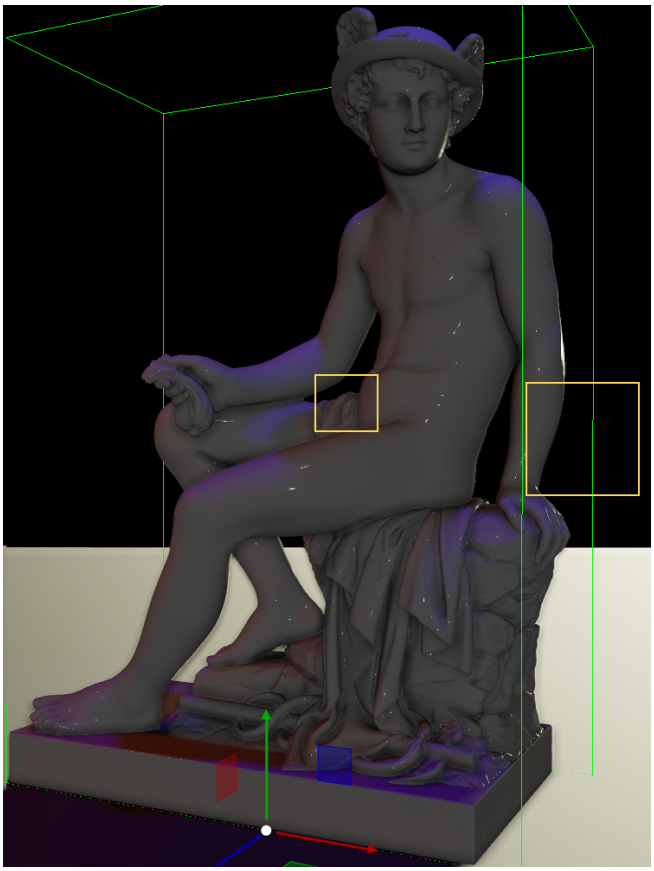
Copyright © 2026 Vizrt

Hardware only (MSAA)	MSAA +Fast Approximate Anti-Aliasing (FXAA)	MSAA + Temporal Anti-Aliasing
		

Specular Anti-aliasing (PBR Material)

Unwanted artifacts on PBR materials can be avoided by using the Specular Anti-aliasing Settings within the PBR Material Editor. This smooths artifacts especially on very detailed curved meshes.

Copyright © 2026 Vizrt

Without Specular AA	With Optimized Specular AA
	

Acknowledgements

All Environment Maps used are available at *https://hdri-haven.com*.

Copyright © 2026 Vizrt

5.15 Scene Compositing

5.15.1 Render Sequence

Render Sequences in the Viz Engine Render Pipeline only distinguish themselves from each other on how the scenes key signal gets generated by default, when the lens distortion gets applied and on which assets (Background and Foreground). In general, compositing is done using an alpha blend operation using the key signal between all stages at the very end of the rendering pipeline. The only exceptions are the Precision Keyer and the External AR sequence, which implement their own blending on the Foreground asset.

Overview

Sequence	Background Key	Geometry Key	Background Support	Background Lens Distortion	Foreground Support	Foreground Lens Distortion	Masking Plug-in Support
Virtual Studio	⛔	⛔	✅	⛔	✅	⛔	✅
Fullscreen	✅	✅	⛔	⛔	✅	⛔	✅
Overlay	⛔	⛔	✅	⛔	✅	⛔	✅
Auto Key	⛔	✅	✅	⛔	✅	⛔	✅
External VS	⛔	⛔	✅	✅	✅	⛔	✅
External AR	⛔	⛔	✅	⛔	✅	✅	✅
Passthrough	not applicable	not applicable	⛔	⛔	✅	⛔	✅
Multi-Layer XR Set	⛔	⛔	✅	⛔	✅	⛔	⛔
Multi-Layer External VS	⛔	⛔	✅	⛔	✅	⛔	⛔
Multi-Layer Virtual Studio	⛔	⛔	✅	⛔	✅	⛔	⛔

Copyright © 2026 Vizrt

Virtual Studio, Multi-Layer Virtual Studio

Scene does not generate a Key signal by default. The Key plug-in needs to be used for scene parts that should have one. Both Background and Foreground are supported, No Lens Distortion is applied on Background and Foreground assets.

Fullscreen

Scene generates a fullscreen key. The Key plug-in has no effect. Only Foreground asset is supported, no Lens Distortion is applied.

Overlay

Scene generates no Key signal by default, the Key plug-in needs to be used. Both Background and Foreground are supported, No Lens Distortion is applied on Background and Foreground assets.

Auto Key

Scene generates Key signal for geometries by default, the Key plug-in can be used. Both Background and Foreground are supported, No Lens Distortion is applied on Background and Foreground assets.

External VS

Special sequence for the Unreal Engine Integration. No Key is generated by default, the Key plug-in needs to be used for scene parts. Both Background and Foreground assets are supported. Lens Distortion is applied on the Background, no Lens Distortion is applied on the Foreground.

External AR

Special sequence for the Unreal Engine Integration. No Key is generated by default, the Key plug-in needs to be used for scene parts. Both Background and Foreground assets are supported. Lens Distortion is applied on the Foreground, no Lens Distortion is applied on the Background.

Passthrough

Special sequence for the Unreal Engine Integration, but can be used for other uses cases as well. The Foreground is sent to both the editor previews and the video output. The Viz Engine Render Pipeline rendering is skipped completely, no rendering, no compositing is done, only output processing.

Multi-Layer XR Set

Sequence to render a Set Extension and AR elements from a single scene with the possibilities to cut-out elements via the Matte Scene.

Copyright © 2026 Vizrt

Multi-Layer External VS

Special sequence for the Unreal Engine Integration to render Viz Engine Graphics on top of an Unreal Engine Virtual Studio.

5.15.2 Precision Keyer

When an asset is being put in the Foreground and the Precision Keyer is activated on it, the keying is executed. The chosen Render Sequence does not matter. The Precision Keyer compositing is different than the linear keying. Graphics that have a key are rendered on top, graphics without are rendered behind the Foreground asset.

5.15.3 Matte Scene

The Matte Scene is supported in all sequences except the Passthrough one. When a Matte Scene is used, the key signal that gets generated is a combination of the scene key, the trash matte and holdout matte values.

See Also

- Working with Holdout and Trash Mattes

Copyright © 2026 Vizrt

5.15.4 Multi-Layer External VS

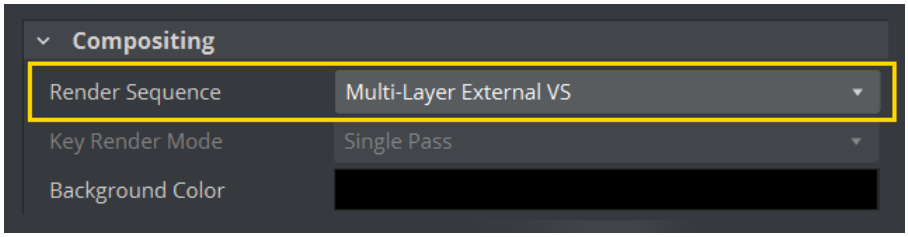
The Multi-Layer External VS sequence is an enhanced Render Sequence that makes it possible to combine an Unreal Virtual Set with Viz Engine Graphics, a keyed input and AR Graphics on top. All from a single scene with lens distortion.

- Viz Engine
  - Scene Configuration
  - Scene Tree Configuration
- Unreal Configuration
  - AUX1
  - AUX2
- Example

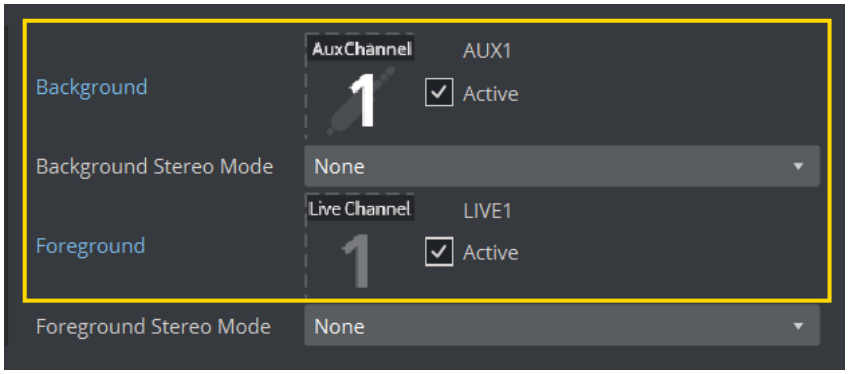
Viz Engine

Scene Configuration

The Render Sequence needs to be set to **Multi-Layer External VS**:



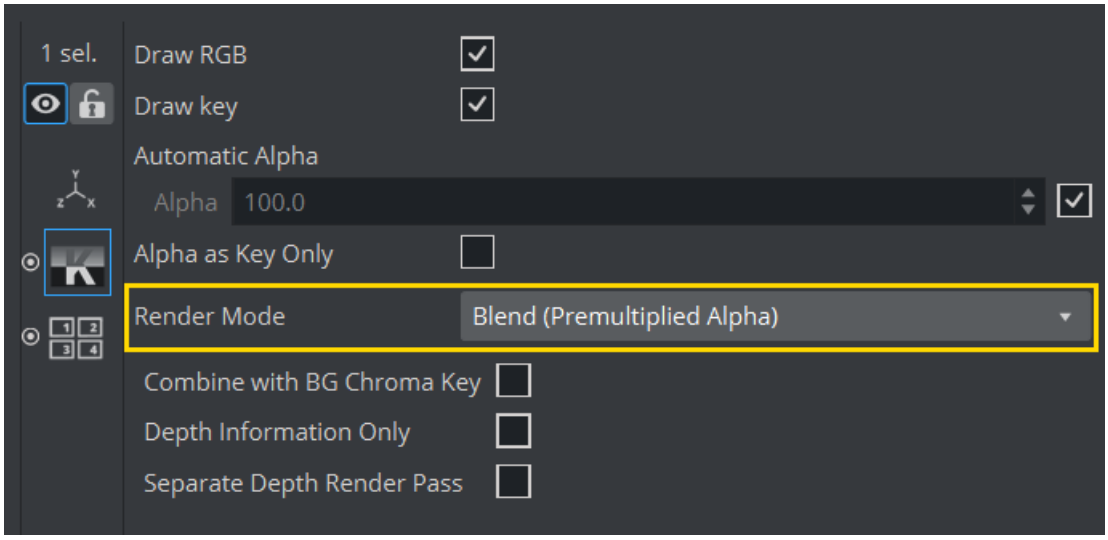
The AUX channel containing the Unreal Engine Virtual Set needs to be put in the scenes **Background** slot, the keyed input in the **Foreground** slot:



Copyright © 2026 Vizrt

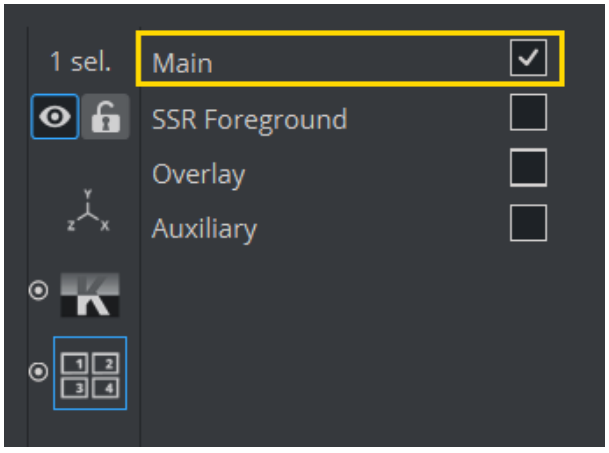
Scene Tree Configuration

All elements need to have a Key for compositing to work properly. To be able to separate the rendering and get proper transparencies back when doing the compositing, the *Key* plug-in's Render Mode needs to be set to **Blend (Premultiplied Alpha)**:



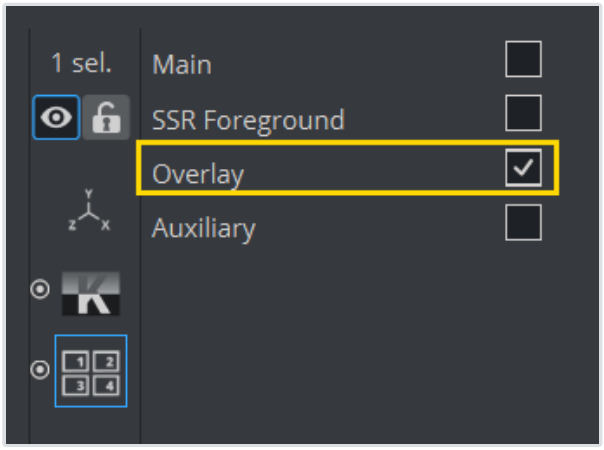
To be able to decide if the Graphics are rendered between the Unreal Virtual Set and the Talent or in front of the Talent, the *Layer* plug-in needs to be used.

Elements rendered in between need to have **Main** checked:



Elements rendered on top need to have the **Overlay** checked:

Copyright © 2026 Vizrt

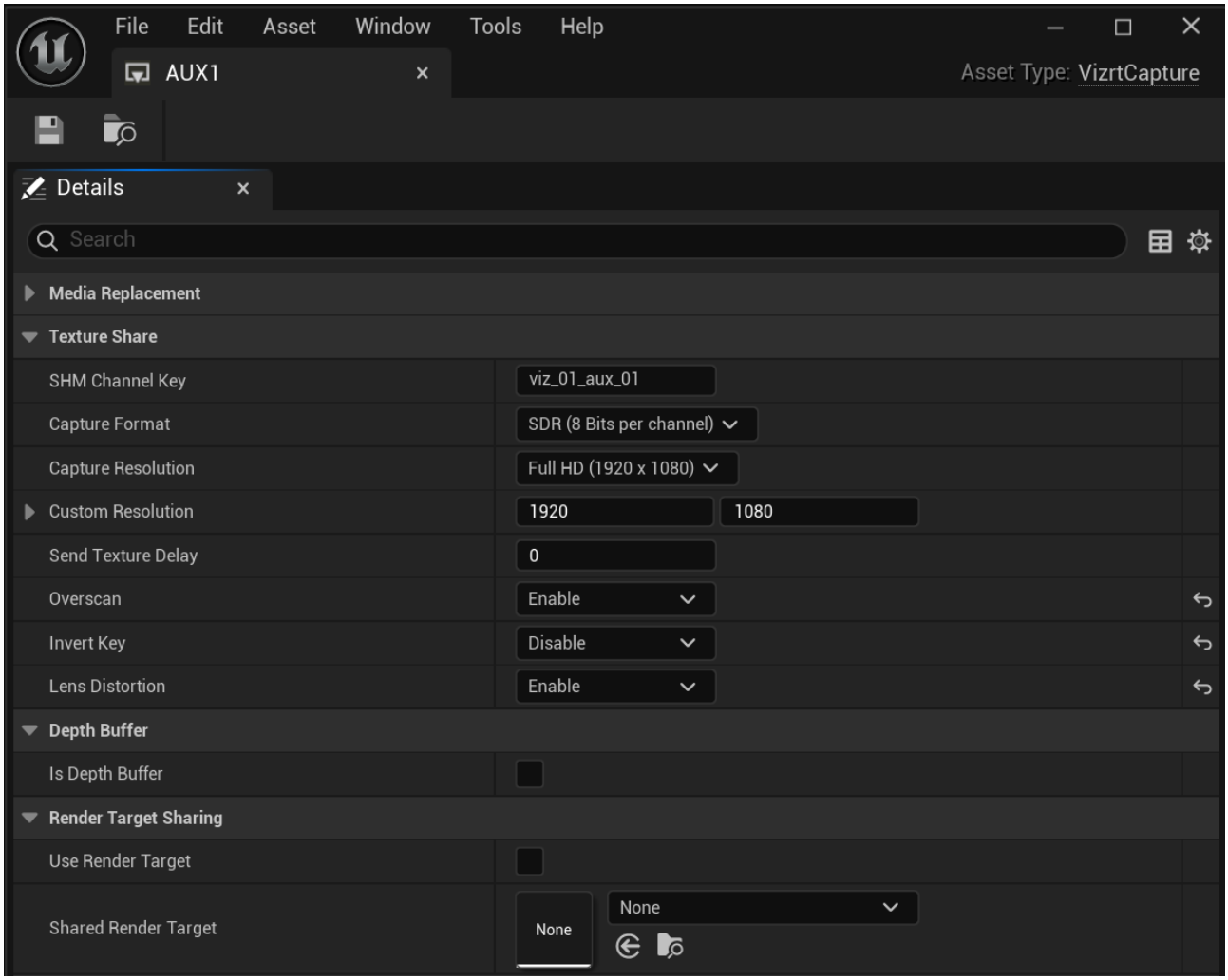


Unreal Configuration

AUX1

Lens Distortion needs to be enabled working and enabled on AUX1.

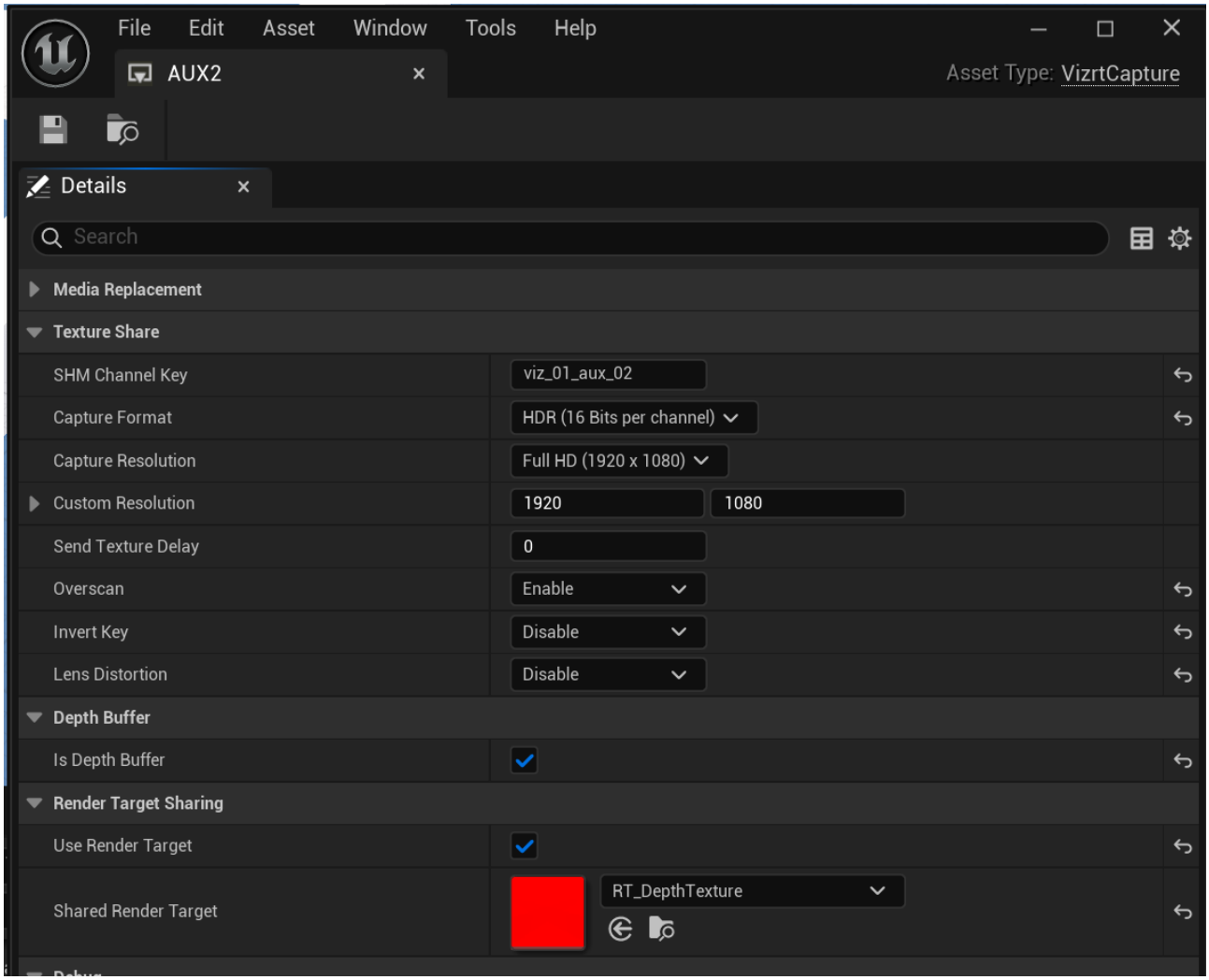
Copyright © 2026 Vizrt



AUX2

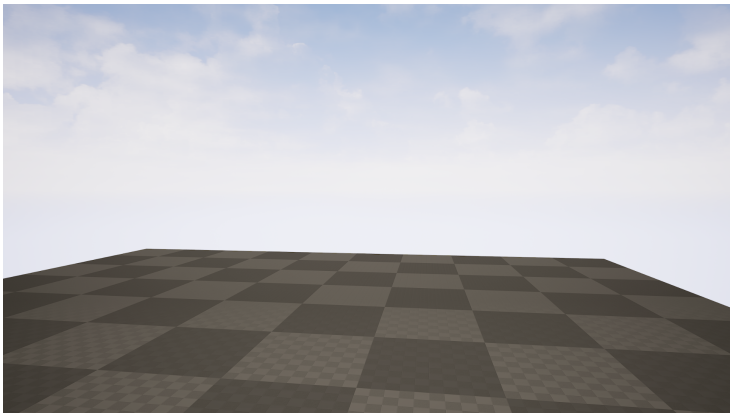
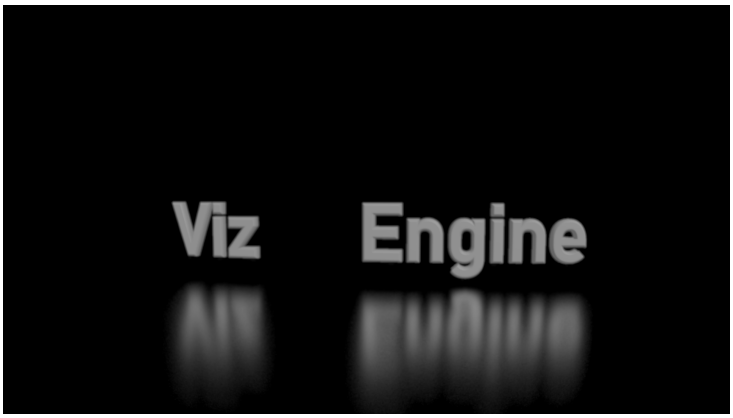
If you use depth sharing, Lens Distortion needs to be Disabled and Overscan Enabled on AUX2

Copyright © 2026 Vizrt

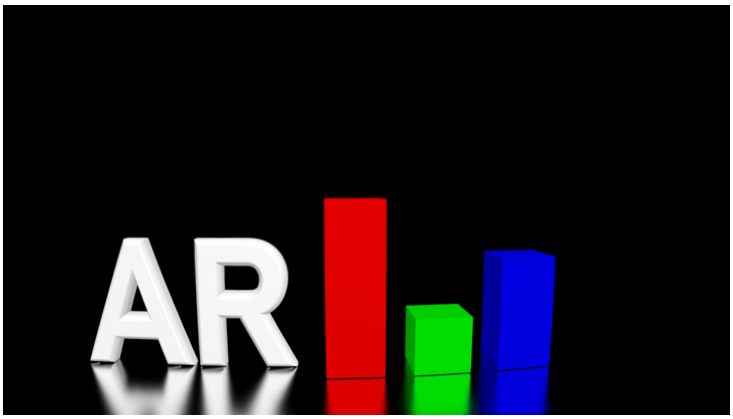
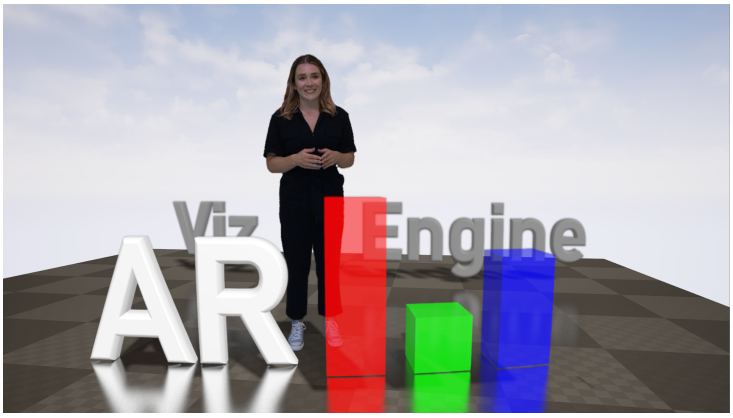


Copyright © 2026 Vizrt

Example

Layer	Rendering Result
Unreal Virtual Set	
Viz Engine Main Layer	
Keyed Talent	

Copyright © 2026 Vizrt

Layer	Rendering Result
Viz Engine Overlay Layer	
Final Composition	

Copyright © 2026 Vizrt

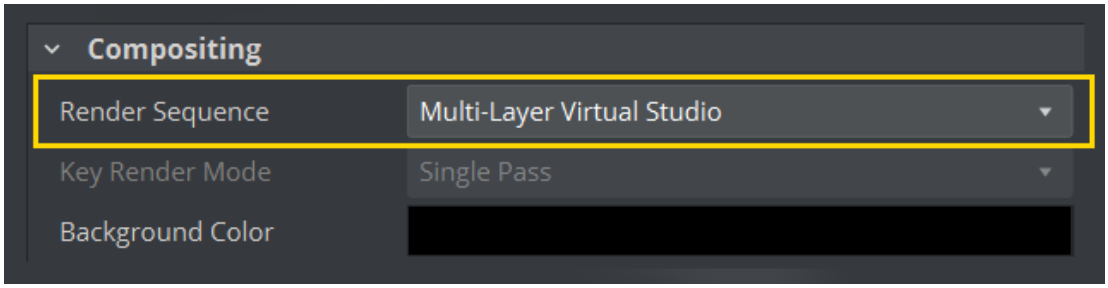
5.15.5 Multi-Layer Virtual Studio

The Multi-Layer Virtual Studio sequence is an enhanced Render Sequence that makes it possible to render a Viz Virtual Studio background, a keyed input and AR elements on top, from a single scene with lens distortion.

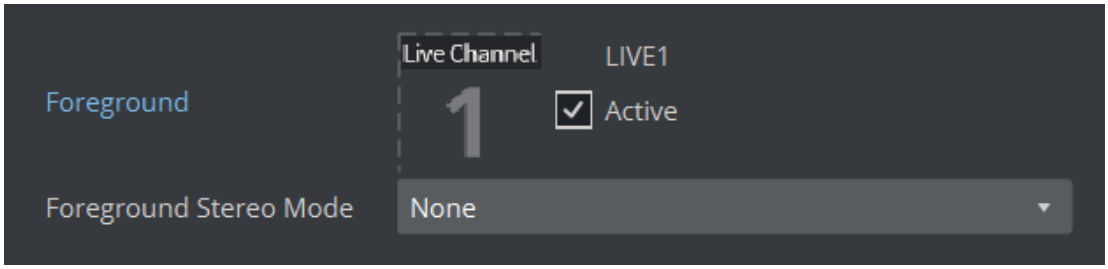
- Scene Configuration
- Scene Tree Configuration
  - Augmented Reality Key
- Post-Processing effects
  - Depth of Field
  - Screenspace Reflections
  - Bloom
- Example

Scene Configuration

The Render Sequence needs to be set to **Multi-Layer Virtual Studio**:



The keyed input needs to be put into the scenes **Foreground** slot:

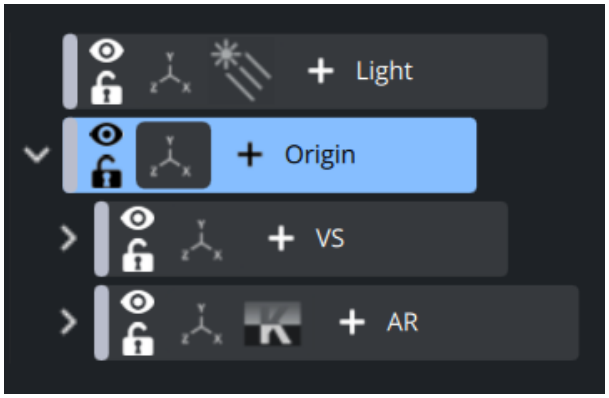


Scene Tree Configuration

Elements without a Key plug-in are part of the background (the Viz Virtual Studio graphics) and elements with a key plugin are part of the foreground (the AR graphics). In contrast to the Viz Virtual Studio sequence, which does the rendering in one pass, the Multi-Layer Virtual Studio sequence separates the scene tree into two parts, which are drawn completely separate and re-combined at compositing time. This needs to be kept in mind when designing

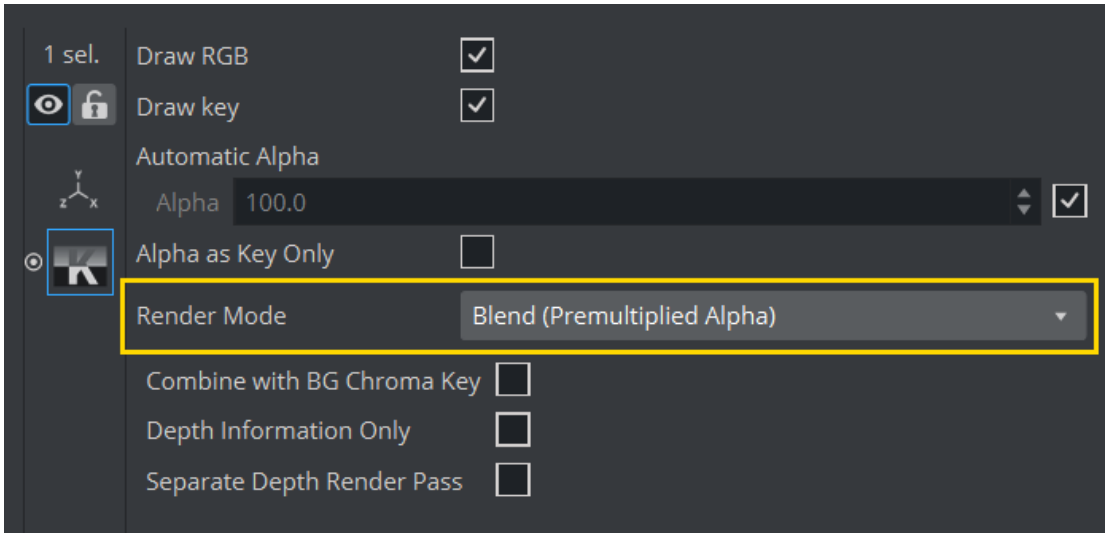
Copyright © 2026 Vizrt

the scene, because rendering *including* post process effects is done twice, one time for Viz Virtual Studio elements and another time for the AR elements.



Augmented Reality Key

To be able to separate the rendering and get proper transparencies back when doing the compositing, the key plugin's Render Mode needs to be set to **Blend (Premultiplied Alpha)**:



Post-Processing effects

All post processing effects are applied on both graphic layers separately. If they can be properly combined during composition depends on the availability of a key signal. If there is no key value, it is advised to not use them.

Depth of Field

Depth of field is working, both layers are affected separately and combined during composition. The AR Key signal is integrated into the depth of field generation.

Copyright © 2026 Vizrt

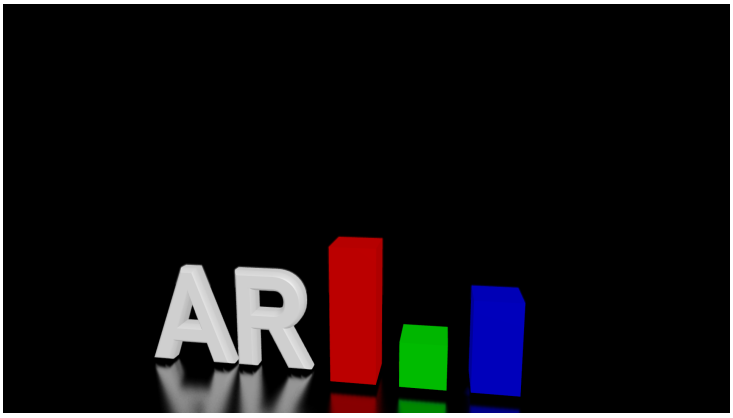
Screenspace Reflections

To make Screenspace Reflections working on the AR graphics, the **Key Intensity** setting needs to be adjusted. The key signal generation is just an approximation, a simplified approach, to be able to combine reflections from AR and the Virtual Studio graphics. Differences between *normal* and *AR* are to be expected.

Bloom

Same as with Screenspace Reflections, to have Bloom on AR elements, the **Key Intensity** setting needs to be adjusted. The key signal generation is just an approximation, a simplified approach, to be able to combine bloom from AR and Virtual Studio graphics. Differences between *normal* and *AR* are to be expected.

Example

Layer	Rendering Result
Viz Virtual Studio Graphics	
Augmented Reality Graphics	

Copyright © 2026 Vizrt

Layer	Rendering Result
Keyed Talent	
Final Composition	

Copyright © 2026 Vizrt

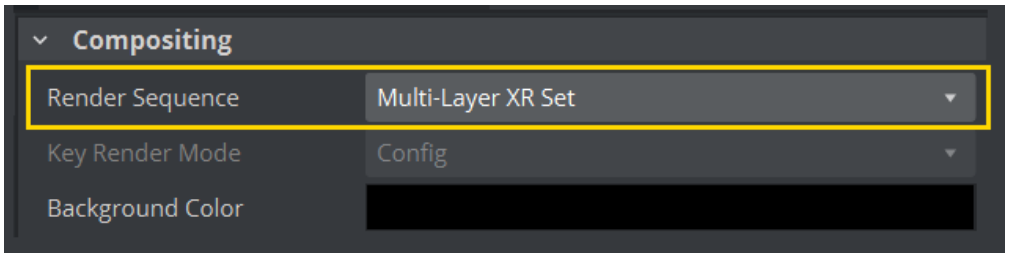
5.15.6 Multi-Layer XR Set

The Multi-Layer XR Set sequence is an enhanced Render Sequence that makes it possible to combine a Live Input, Set Extension and AR graphics using a Matte Scene.

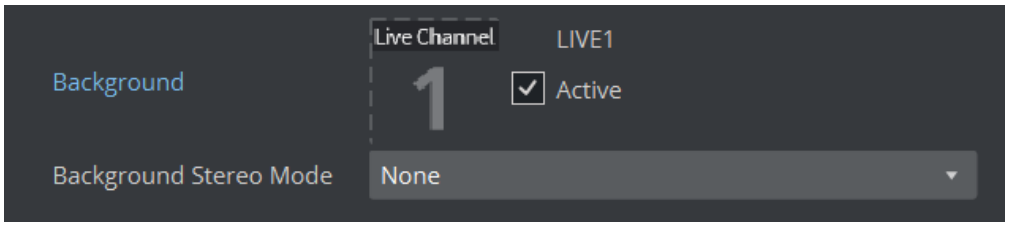
- Scene Configuration
- Scene Tree Configuration
- Matte Scene
    - Example

Scene Configuration

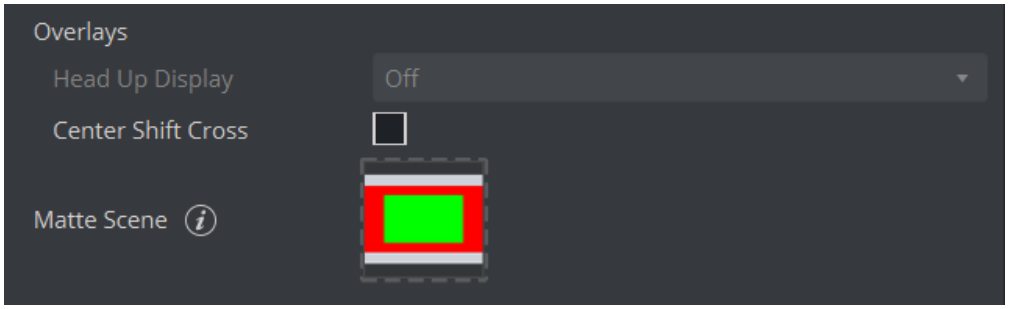
The Render Sequence needs to be set to **Multi-Layer XR Set**:



The Live Input needs to be put in the scenes **Background** slot:



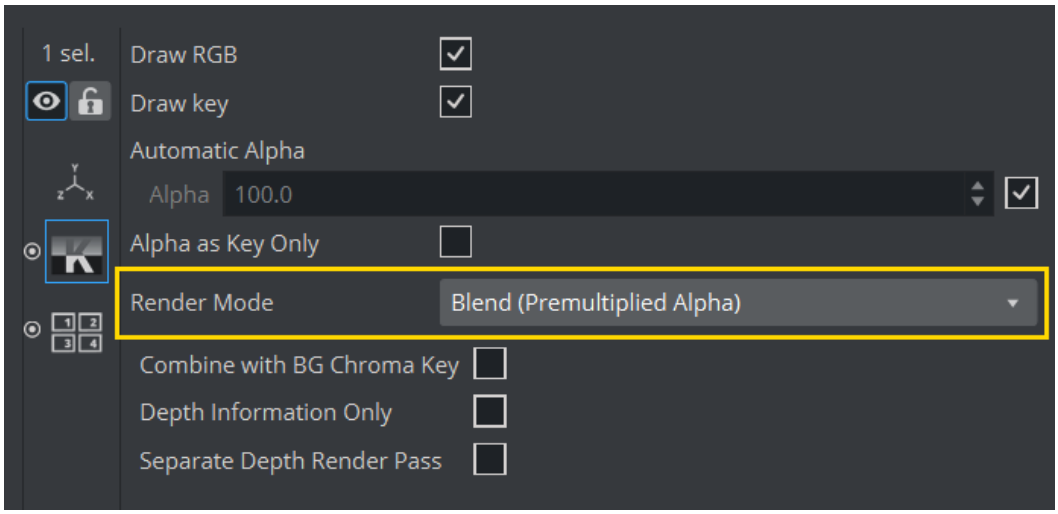
The Matte Scene needs to be referenced within the scenes **Matte Scene** slot:



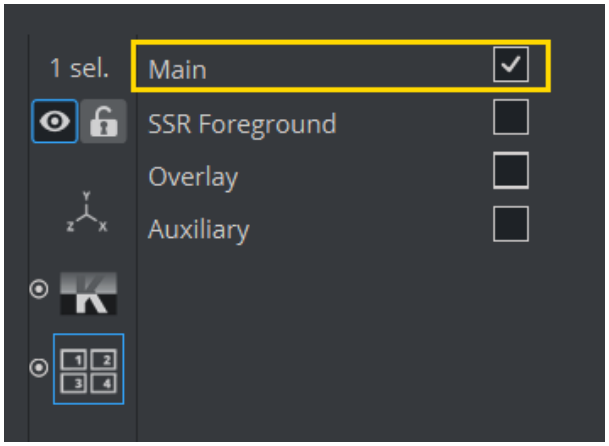
Copyright © 2026 Vizrt

Scene Tree Configuration

All elements that belong to the Set Extension do not need to have a Key plug-in, but all elements that belong to the AR graphics need one. To be able to separate the rendering, and get proper transparencies back when doing the compositing, the *Key* plug-in's Render Mode needs to be set to **Blend (Premultiplied Alpha)**:

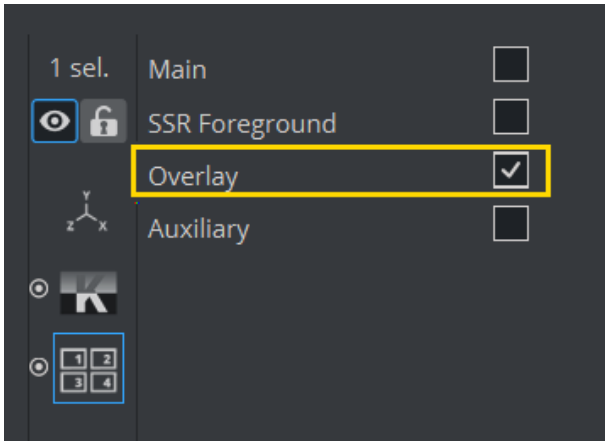


Elements rendered as **Set Extension** need a **Layer plug-in** with **Main** checked:



Elements rendered as **Augmented Reality** graphics need a **Layer plug-in** with **Overlay** checked:

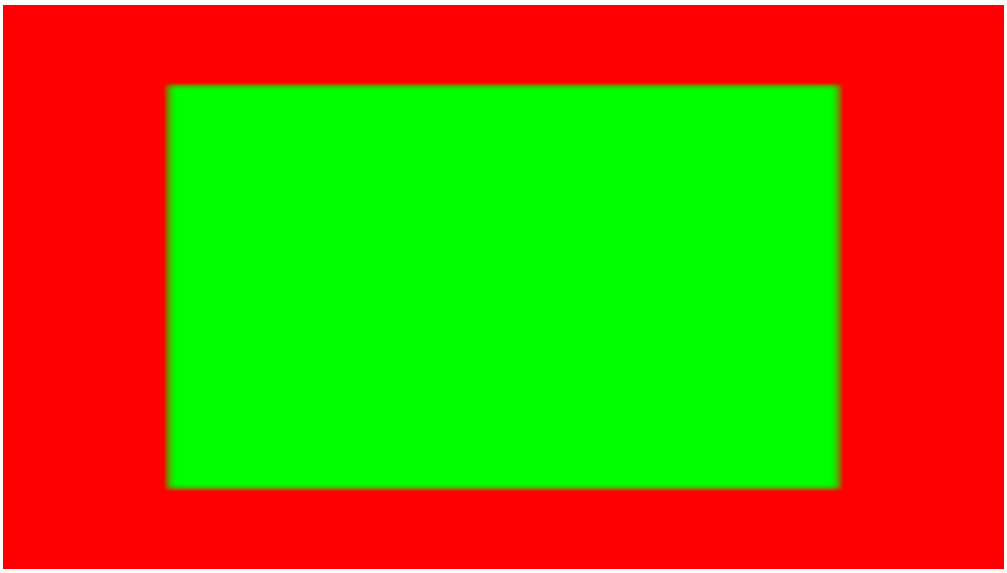
Copyright © 2026 Vizrt



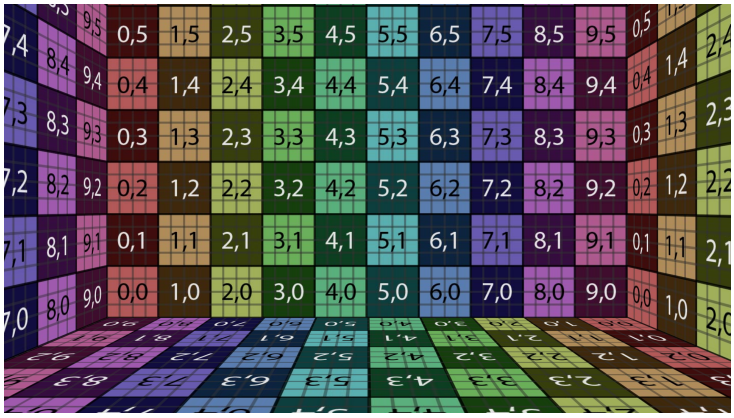
Matte Scene

The trash matte channel of the matte scene, is used to blend between the Live Input and the Set Extension layer. *Red=1* means Set Extension is 100% visible and *Red=0* means 100% Live input. Values in between allow for a smooth blend.

Matte Scene



Example

Layer	Rendering Result
Viz Engine Set Extension	
Live Input	
Viz Engine AR Graphics	

Copyright © 2026 Vizrt

Layer	Rendering Result
Final Composition	

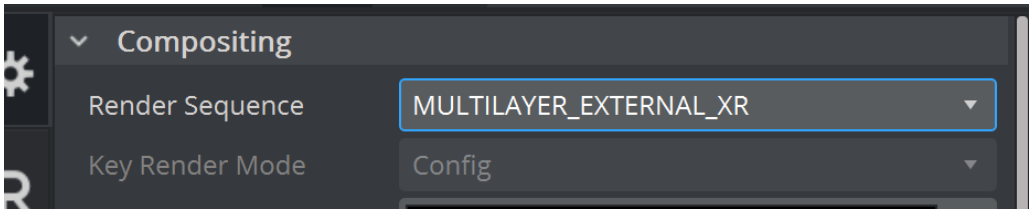
Copyright © 2026 Vizrt

5.15.7 Multi-Layer External XR

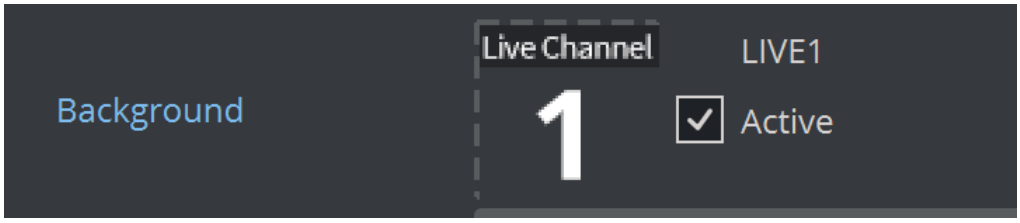
The Multi-Layer External XR sequence is an enhanced Render Sequence that makes it possible to combine a Clip/Live Input, with an Aux Channel and AR graphics using a Matte Scene.
Camera color correction can be applied everywhere except for the Clip/Live input.

Scene Configuration

The Render Sequence needs to be set to **Multi-Layer External XR.**



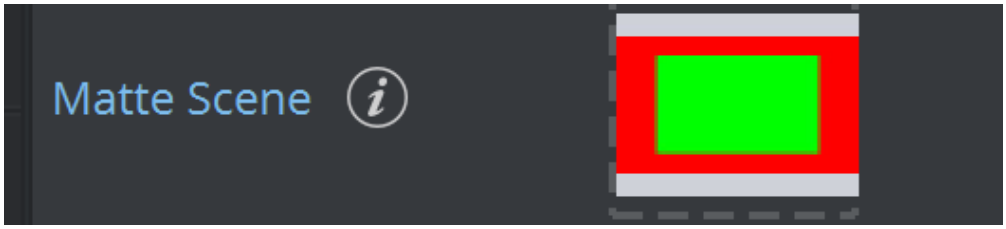
The Clip/Live Input needs to be placed in the scene **Background** slot.



The Aux channel needs to be placed in the scene **Foreground** slot.



The Matte Scene needs to be referenced within the scenes **Matte Scene** slot:

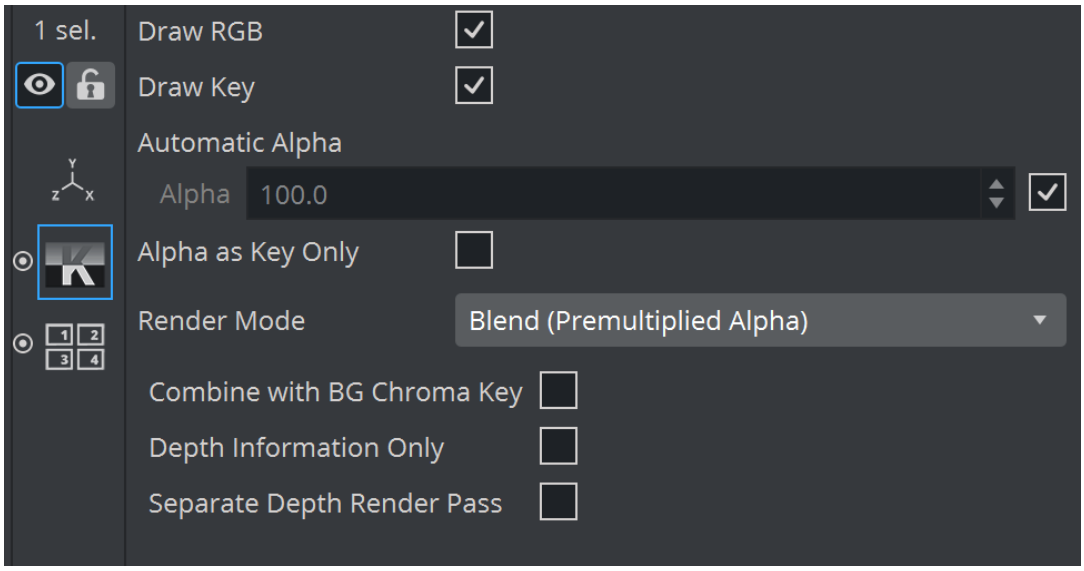


Copyright © 2026 Vizrt

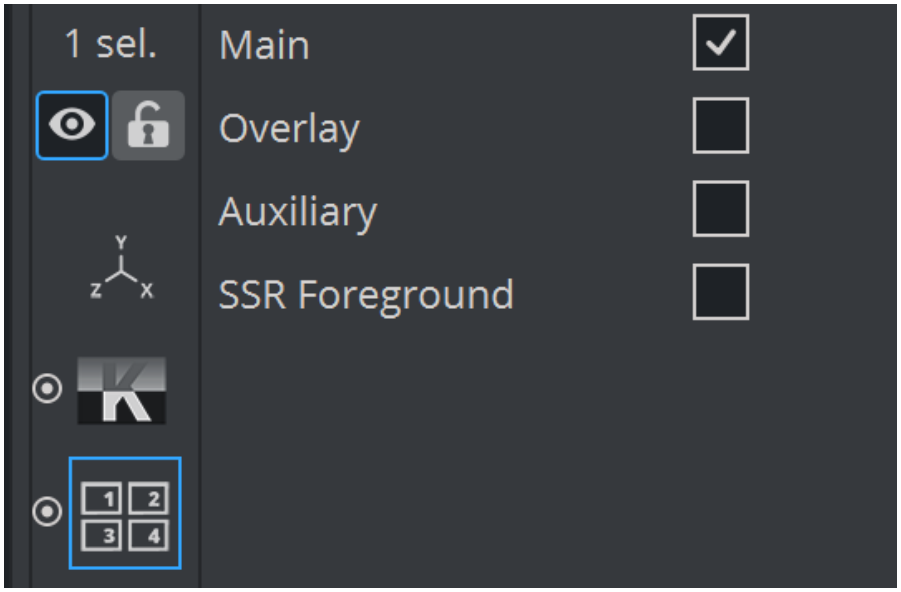
Scene Tree Configuration

Additional elements can be rendered and properly blended on top of the background/foreground composition via a **Key** plugin and a **Layer** plugin.

The **Key** plugin **Render Mode** needs to be set as **Blend (Premultiplied Alpha)**.



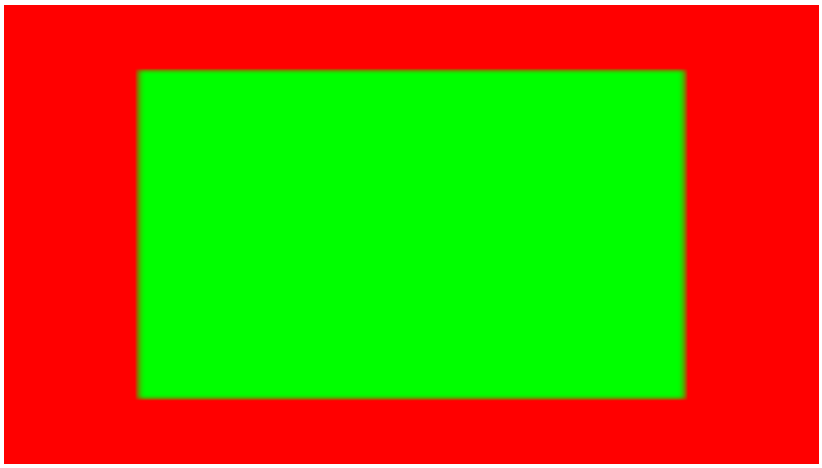
The **Layer** plugin needs to be set as **Main**.



Copyright © 2026 Vizrt

Matte Scene

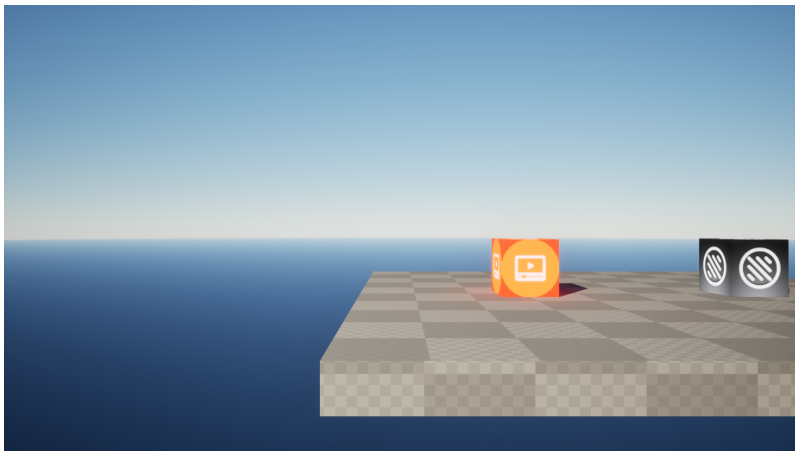
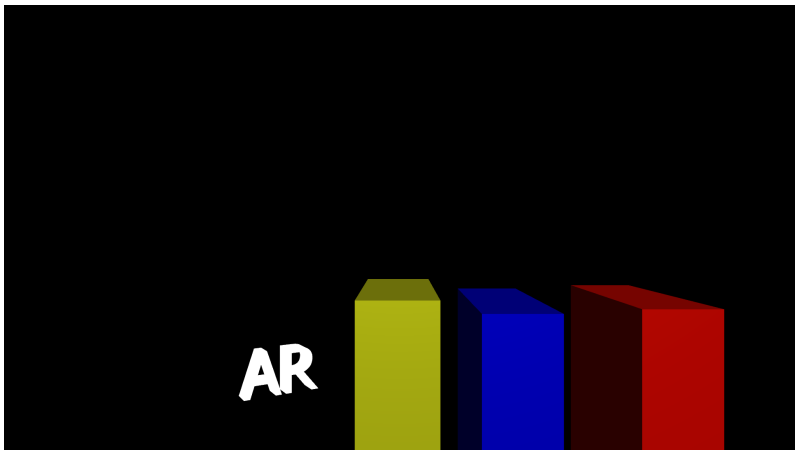
The trash matte channel blends the Clip/Live Input with the Aux Channel. A Red value of 1 makes the Aux Channel 100% visible, while a Red value of 0 displays 100% of the Clip/Live Input. Values between 0 and 1 create a smooth blend between the two sources.



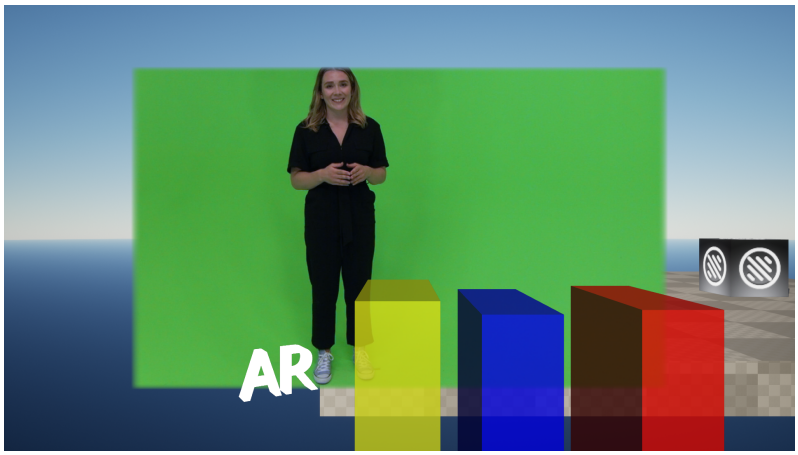
Render sequence example

Layer	Rendering Result
Clip/Live Input (**Background** slot)	

Copyright © 2026 Vizrt

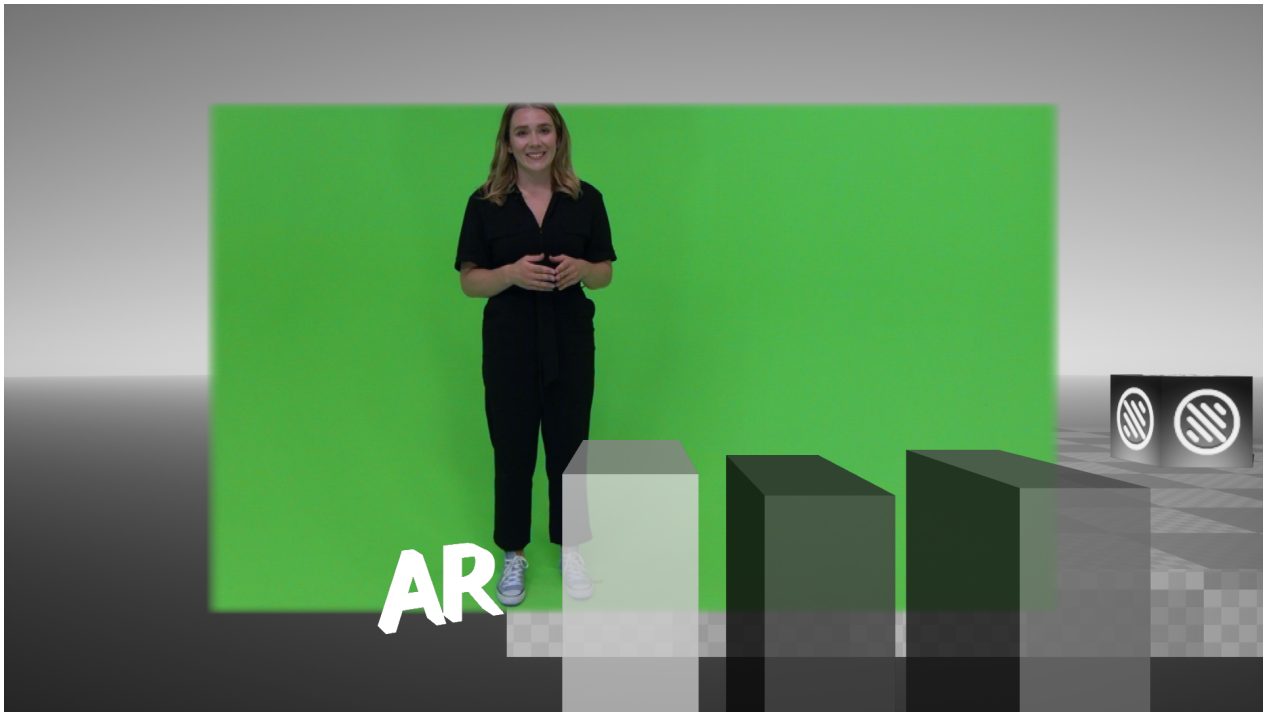
Layer	Rendering Result
Aux Channel (**Foreground** slot)	
Viz Engine AR Graphics (**Main** layer)	

Copyright © 2026 Vizrt

Layer	Rendering Result
Final Composition	

Camera color correction

All layers except for the **Background** slot will be color corrected according to the configured camera color correction LUT. As an example, a grayscale LUT is applied to the scene.



5.16 Virtual Studio, Augmented Reality and Extended Reality

The following chapters explain how to work with Virtual Studios, how to setup and create AR and XR graphics:

- To Setup Talent Shadow in a Virtual Studio
- Setup Precision Keyer
- To Setup a Reflection for Augmented Reality
- To Setup a Talent Reflection in a Virtual Studio
- Working with Holdout and Trash Mattes
- Projected Virtual Window

Copyright © 2026 Vizrt

5.16.1 To Setup Talent Shadow in a Virtual Studio

- Casting Shadows by Doing Pose Estimation
- Casting Shadows by Using Keyed Texture from the Precision Keyer
    - Scene Preparations
    - Scene Tree Preparations

For a more realistic rendering and to prevent the effect of floating talent, shadow can be added in the virtual scene. To cast a shadow into the virtual scene, either a geometry or a texture with an alpha mask is needed.



Copyright © 2026 Vizrt



Casting Shadows by Doing Pose Estimation

To use a geometry, please see our Viz Reality Connect solution.

Casting Shadows by Using Keyed Texture from the Precision Keyer

This section explains how to setup a shadow caster using the keyed texture from the Precision Keyer, which is placed in the scene tree (only) to cast the shadow.

Scene Preparations

- Add a live input with your camera signal (or a clip as a placeholder).
- Enable the Precision Keyer, setup accordingly.
- Enable Talent Tracking on the keyed asset.

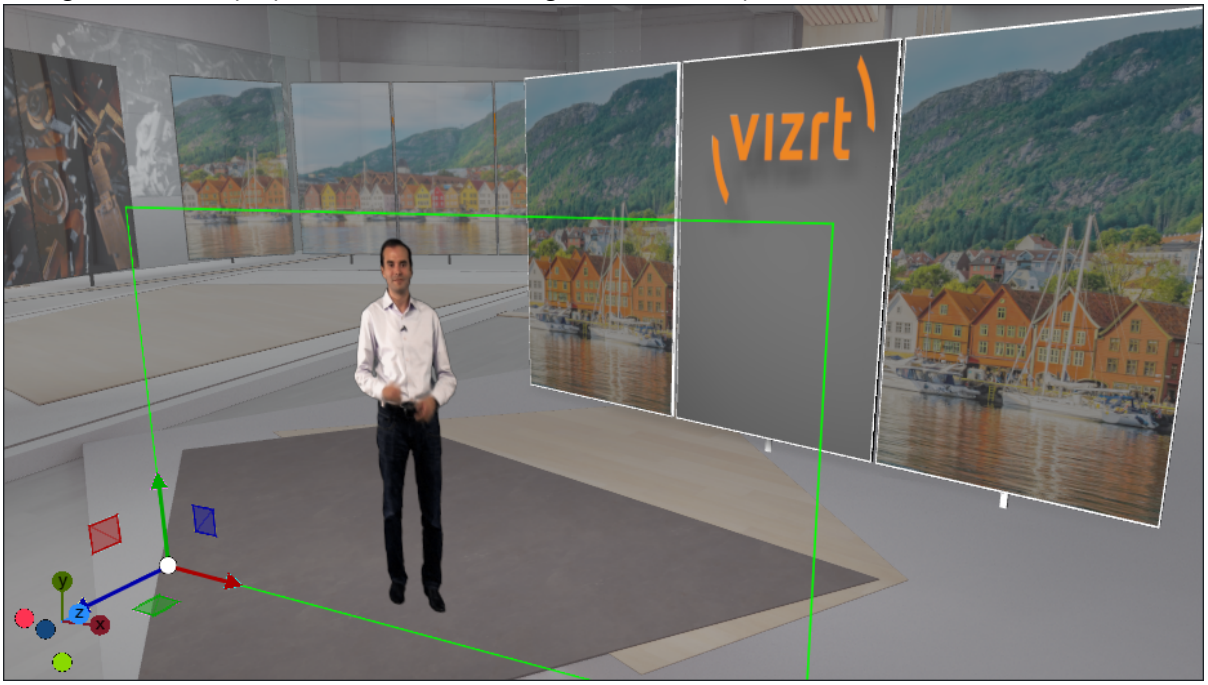
Scene Tree Preparations

- Add a top-level container called *tracker*.
- Add another container *textureAnchor* as a child node, and add a quad as geometry. Add a PhongMaterial and assign your media asset as color input.

Copyright © 2026 Vizrt

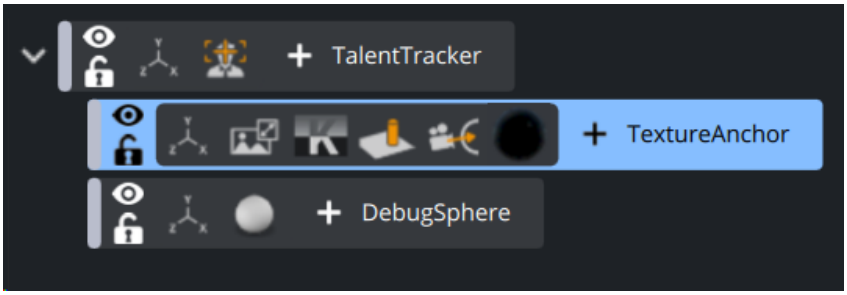


- Change the material properties, set the Rendering Method to *Transparent.*



- Add TalentTracker plug-in on *tracker*, select *TalentShadow* as mode, and select PersonID. Tracking is needed to have the correct contact point of the talent and the floor (Floor must be at y=0 in a Virtual Studio).
- Transform the container *textureAnchor* as needed to be fullscreen. Add a LookAt plug-in and enable *KeepSize*. The texture is now fullscreen and billboards the screen.
- Add a ShadowCaster plug-in to enable shadow casting, and add a Key plug-in and disable *Draw RGB* and *Draw Key* to exclude the texture from normal rendering. It only affects the shadow.
- The scene tree looks like this:

Copyright © 2026 Vizrt



- Now setup your Virtual Studio scene as needed, and add lights that cast shadows.
- Your scene now casts shadows from the talent.



- Add the used media asset as Foreground in the scene settings to get the final composition:

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

5.16.2 Setup Precision Keyer

- Sample
  - Key FOREGROUND Asset
  - Key Textures in Scene Tree
- Prerequisites
  - NDI
- Automatic Adjustment of Base Parameters
- Manual Setup of Base Parameters
  - Hue Adjust
  - White Point (Transparency) and Black Point (Opacity)
  - Processed Foreground / RGB
  - Cyan / Magenta / Yellow Gain (CMY Gain)
  - Final Result
- Other Parameters
  - Background Lightwrap and Color Lightwrap
- Aux Output
- AI Keyer Color Mode
  - AI Keyer and Keying Licensing in Viz Engine

Sample

Key FOREGROUND Asset

The Precision Keyer can work on foreground assets (**SCENE > Foreground**) or on texture which are used within the scene.



Virtual Studio

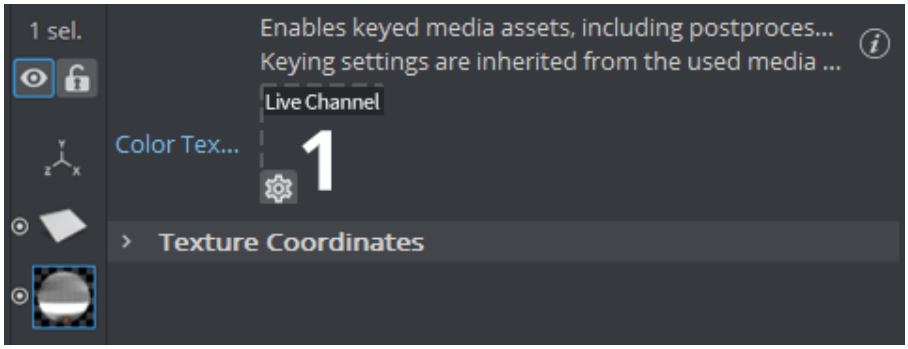
Copyright © 2026 Vizrt

	Live input as FOREGROUND asset
	Result with configured the Precision Keyer

Key Textures in Scene Tree

If used within the scene, use a container with a geometry and add a RenderSystemMaterialPrecisionKeyer. Drag the used media asset on the color widget of the material. The Keyer must be enabled on the asset (for example, Live1).

Additionally, the texture coordinates can be manipulated. See Texture Generation for more information.



Copyright © 2026 Vizrt

	Virtual Studio
	Live input added as texture in the scene
	Result with configured the Precision Keyer

Prerequisites

The Precision Keyer works on live inputs (including streams and NDI) and clips. The asset must be used as a texture in the scene (either in the scene tree, or as "FOREGROUND" in the scene), and it requires a solid background color which is either green or blue.

Copyright © 2026 Vizrt



As a first step, select the color mode which should be used by the Keyer, which is either Greenscreen (default) or Bluescreen.



Then, you need to adjust the base parameters of the Keyer. This can be done either by using the *Auto Adjust* functionality or manually.

NDI

If you want to use the Precision Keyer with active NDI output in a VS setup, please be aware that Viz Engine puts the matte of the Keyer on the key signal. Depending on the device which is used to read the NDI signal, this key might be already applied and thus Virtual Studio elements are not visible on the NDI output. To fix this, you need a fullscreen key or a device that shows fill (RGB) independently from the key (Alpha). To enforce Viz Engine to send a fullscreen key, set `NdiOut1.ContainsAlpha = 0` in the configuration file.

Automatic Adjustment of Base Parameters

Use the auto adjust functionality of Viz Engine. The result looks like this:

Copyright © 2026 Vizrt



The auto adjustment uses the current input signal to calculate:

- Hue Adjust
- White Point (Transparency)
- Processed FG / RGB

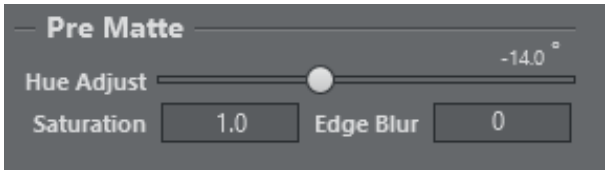
Furthermore, it sets the following values based on some well-proven defaults:

- Black Point (Opacity)
- CMY Gain
- BG Lightwrap
- Radius/Sharpen of Matte Denoise

However, there still might be the need to slightly adapt the parameters, depending on your input signal.

Manual Setup of Base Parameters

Hue Adjust



The Keyer works best if the background is perfect green (RGB(0, 255, 0)) or perfect blue (RGB(0, 0, 255)). As this is not what we have in most real-world setups, you need to color-shift the input signal until you get perfect green or perfect blue.

Enable the *Matte* view and adjust the hue shift until you have the maximum of contrast you can get. In perfect scenarios, this would mean that you get a full white background, and a opaque black foreground. Depending on your lighting situation, you will not get perfect white and black on the matte view, this is handled in the next steps.

Copyright © 2026 Vizrt

Try to find the hue shift which is the best you can get now, and use 100% for the white point (Transparency) and 0% for the black point (Opacity) to see the unscaled values while adjusting the hue shift.

Input Signal



Matte Before Hue Adjust

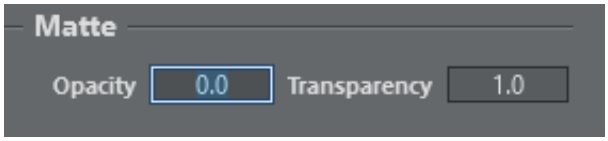


Matte After Hue Adjust

Copyright © 2026 Vizrt



White Point (Transparency) and Black Point (Opacity)



After the hue adjustment, we should adjust the white point. This scales the input values so that the background is fully keyed (full white on the Matte view). Start from 100% and reduce it until the background is fully white, Do not reduce it more as necessary, as it would start remove details at the border of the foreground.

Now we need to find the black point, which scales and shifts the input values so that the foreground is fully opaque. Start from 0% and increase it, until your foreground is fully opaque. Have a look on details and borders and do not increase it more as necessary.

Matte Before White Point Adjustment

Copyright © 2026 Vizrt



Matte After White Point Adjustment

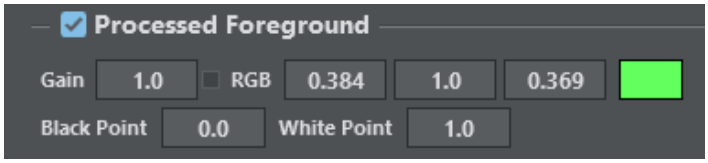


Matte After Black Point Adjustment

Copyright © 2026 Vizrt



Processed Foreground / RGB



After the base adjustment of the hue and the white point and black point, you should adjust the RGB values of the processed foreground. These values are **subtracted** from the input signal and then are used in a linear keying mechanism, so the goal would be that these RGB values match the color of the background. Try to adapt the values so that they match the dominant color of the unprocessed input, but make sure that you do not use higher RGB values as needed. Incorrect or too high values decrease the quality of the linear keying and get noticeable with false colors around the edge between foreground and background.

- **BlackPoint:** Scales the key value used when processing the foreground. Only affects areas where matte is > 0.0 and < 1.0. Can be used to enlarge the talent in semi-transparent regions, increases lightness around edges.
- **WhitePoint:** Scales the key value used when processing the foreground. Only affects areas where matte is > 0.0 and < 1.0. Can be used to shrink the talent in semi-transparent regions, decreases lightness around edges.

If "Processed Foreground" is disabled, the keyer uses linear alpha blending to compose foreground and background.

Input Signal

Copyright © 2026 Vizrt



Processed FG before RGB Adjustment



Processed FG after RGB Adjustment (0.345, 0.713, 0.078)

Copyright © 2026 Vizrt



Cyan / Magenta / Yellow Gain (CMY Gain)

Due to the despilling process, which removes greenish/blueish parts of the skin, it can happen that the color tone is modified too much. Use the CMY gain adjustment to bring Cyan, Magenta or Yellow back as necessary.

There are two versions of lightness restoration available. *DARK* slightly darkens values in despilled areas, while *LIGHT* preserves the lightness and leads to more bright edges.

Before CMY Adjustment



After CMY Adjustment

Copyright © 2026 Vizrt



Final Result

Now add a background or your Virtual Studio and check the result:



Other Parameters

Background Lightwrap and Color Lightwrap

Color Lightwrap

Adds a specific color either around the edge (Edge Gain) or in the spill areas (Spill Gain).

Copyright © 2026 Vizrt

Background Lightwrap

Blends with the background, either around the edge (Edge Gain) or in the spill areas (Spill Gain). This can be used to improve edge quality and to apply the color tone of the background onto despilled areas.

No Lightwrapping



Color Lightwrap, Edge Gain = 0.5, Color = RGB(255, 0, 0)



Copyright © 2026 Vizrt

Color Lightwrap, Spill Gain = 0.4, Color = RGB(0, 0, 255)



Background wrapping, Edge Gain = 0.3



Copyright © 2026 Vizrt

Background wrapping, Spill Gain = 1.0



The background used for lightwrapping can be blurred, and there are multiple parameters available:

- **Edge Blur:** If this value is greater than zero, edge detection is performed and edges found are blurred by the amount set in this field. The blurred edges serve as input to brighten up the edges to overcome overshooting when going from a dark color to the Cyclorama. These edges are used for lightwrapping.
- **Edge Blur Factor:** Determines how much of the background blur is used for the edge lightwrapping. 1.0 = same blur as for spill lightwrapping, 0.0 = use unblurred background.
- **Kernel:** Choose between LAPLACE and SOBEL. LAPLACE is more fine-grained and keeps more details.
- **Blur:** The blurred background is used for lightwrapping onto the foreground together with the spill matte or the edge matte. See also Edge Blur Factor.

The background which is used to do the lightwrapping around edges is blurred by using *Blur * EdgeBlurFactor*, the background used to do the lightwrapping for spill areas is blurred using *Blur* only. This allows to reduce the used blur on the edge lightwrapping, while using more blur on the spill lightwrapping areas, which should be used to prevent transparency effects.

Aux Output

Important: The Parallel Outputs license is necessary to use this feature.

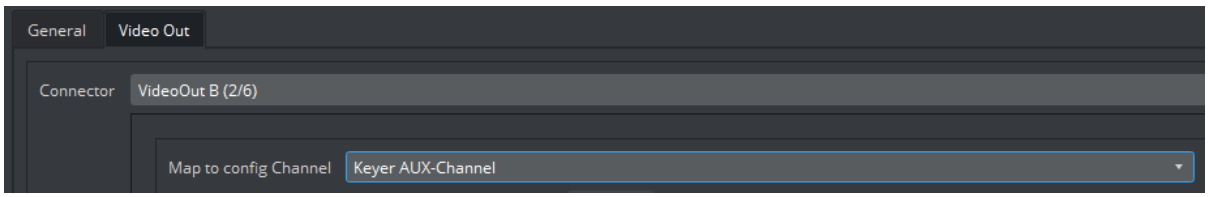
The Precision Keyer generates and consumes multiple internal textures, which can be viewed for monitoring reasons:

- **Primary Matte (Normal, Inverted, Advanced):** Shows the calculated primary matte (`0.0` is black, `1.0` is white). Can be shown *Inverted* or *Advanced*, where `0.0` is green and `1.0` is red, but the range in between still goes from black to white. This helps to distinguish black/white from near-to-black and near-to-white on monitors.

Copyright © 2026 Vizrt

- **Processed Foreground (Normal, Advanced):** Shows the processed foreground. Advanced mode shows RGB (0.0, 0.0, 0.0) as red, while all other values are shown as normal. This helps to distinguish real black (fully removed foreground) from near-to-black (not fully removed) on monitors.
- **Spill Matte:** Shows the spill that has been detected and removed. Used while despilling and to apply Lightwrapping for Spill areas.
- **Edge Matte:** Shows the edges that have been detected.
- **Highlights:** Shows the highlights which are added by the Keyer.

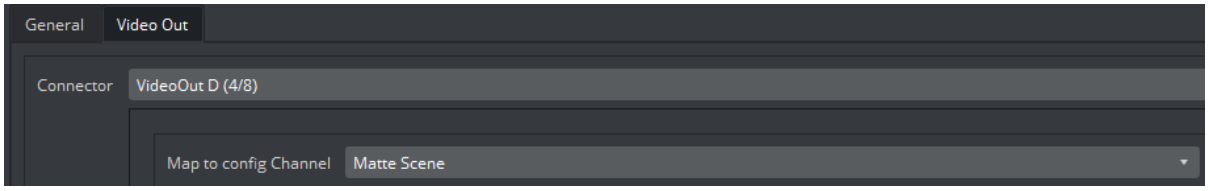
Since Viz Engine version 5.3, these textures can be shown in parallel on a separate SDI outputs. To enable this feature, map one of the video connectors to the config channel *Keyer AUX-Channel*.



If one of the connectors is mapped to *Keyer Aux-Channel*, the Aux Output of the Keyer is sent to this output by default (instead of the Program output). This can be overwritten with commands:

```
# Enable AUX output
MAIN_SCENE*VIDEO*VIDEOIN*1*AUX_OUTPUT SET OFF|KEY|KEY_ADVANCED|INVERSE_KEY|...

# Overwrite AUX output target (AUTO: Use dedicated connector if configured, INTERNAL:
use Program output always, AUXCHANNEL: use dedicated output always, BOTH: use both)
MAIN_SCENE*VIDEO*VIDEOIN*1*AUX_OUTPUT*TARGET SET AUTO|INTERNAL|AUXCHANNEL|BOTH
```

In addition, the matte scene can be sent to a second dedicated output by mapping another connector:



AI Keyer Color Mode

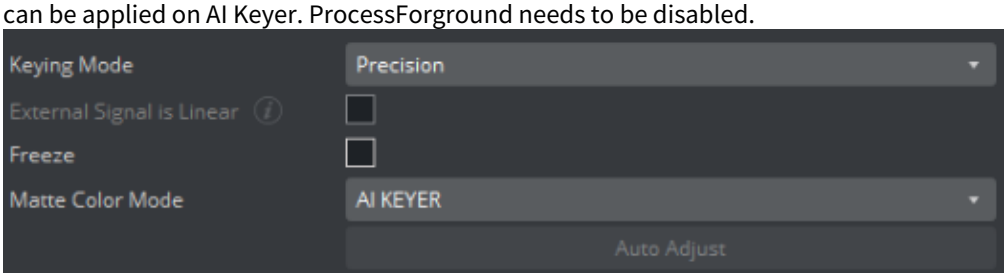
Important: The AI Keyer Greenscreen-less license is necessary to use this feature.

The Precision Keyer settings now support AI Keyer Matte color mode.

AI Keyer and Keying Licensing in Viz Engine

1. No chroma Keying license, no AI Keying license.
    a. Keying mode is inactive.

   b. AI Keyer color mode is not possible to set and returns a missing license answer. We still get key signal (as its embedded to RGBA texture), but extra control of Precision Keyer is not possible.
2. Chroma Keying license available, no AI Keying license.
   a. Keying mode Precision is possible. Precision Keyer color mode of GreenScreen, BlueScreen and External Matte works, other keying modes also work.
   b. AI Keyer color mode is not possible to set, and returns a missing license answer.
3. No chroma Keying license, AI Keying license is available.
   a. Keying mode Precision is possible. Precision Keyer color mode of GreenScreen, BlueScreen and External Matte do not work, other keying modes do not work.
   b. Precision Keyer Color Mode of AI Keyer works, and edge/blur/light wrap properties of Precision Keyer can be applied on AI Keyer. ProcessForground needs to be disabled.
4. Chroma Keying license available, AI Keying license is available.
   a. Keying mode Precision is possible. Precision Keyer Color Mode of GreenScreen, BlueScreen and External Matte works, other keying modes also work.
   b. Precision Keyer Color Mode of AI Keyer works, and edge/blur/light wrap properties of Precision Keyer can be applied on AI Keyer. ProcessForground needs to be disabled.



Commands to get and set **Keying Mode**, and **Matte Color Mode**:

```
send MAIN_SCENE*VIDEO*VIDEOIN*3*KEYING_MODE SET/GET
send MAIN_SCENE*VIDEO*VIDEOIN*3*KEYER_COLOR_MODE SET/GET
```

See Also

- Texture Generation

5.16.3 To Setup a Reflection for Augmented Reality

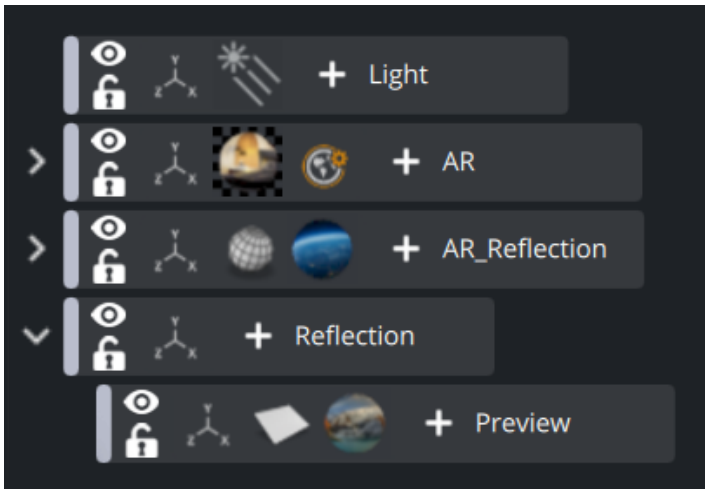
Planar Reflection Material simulates reflections on arbitrary geometries defined either via a virtual rectangle in either 3D space or a fullscreen rectangle. This makes it possible to create reflections on AR objects cast virtually from live or clip input or create talent reflections in a virtual studio setup. In contrast to screen space reflections (SSR), those reflections are always visible even if the virtual rectangle is not visible on the screen.

Common

There are at least two containers necessary to setup the planar reflection, one is needed to setup the material and the second one is responsible to control the location of the reflection plane in space. Use **Add Planar Reflection (Viz Engine)...** to create the Planar Reflection Material. Then drag and drop your reflective_plane on *Reflection Container*.

Augmented Reality Setup

The scene tree below shows a minimal setup of setting up the reflection.



Container Name	Description
ar	The subtree contains everything needed to render the AR object with the default Viz Engine Renderer `PBR Material`.
ar-reflection	This is a copy of the `ar` tree with everything removed that should not be reflective, which has the `Planar Reflection`.
reflection	A simple container which is linked to the `Planar Reflection`'s Reflection Container.

Copyright © 2026 Vizrt

Container Name	Description
preview	For scene editing purposes only. Has a Rectangle plug-in with default settings and a PBR Material with the reflection texture on the Emissive channel. This container should be hidden before going On Air.

Closeup of an AR Object without Planar Reflection



Closeup of an AR Object with Planar Reflection Added



Copyright © 2026 Vizrt

5.16.4 To Setup a Talent Reflection in a Virtual Studio

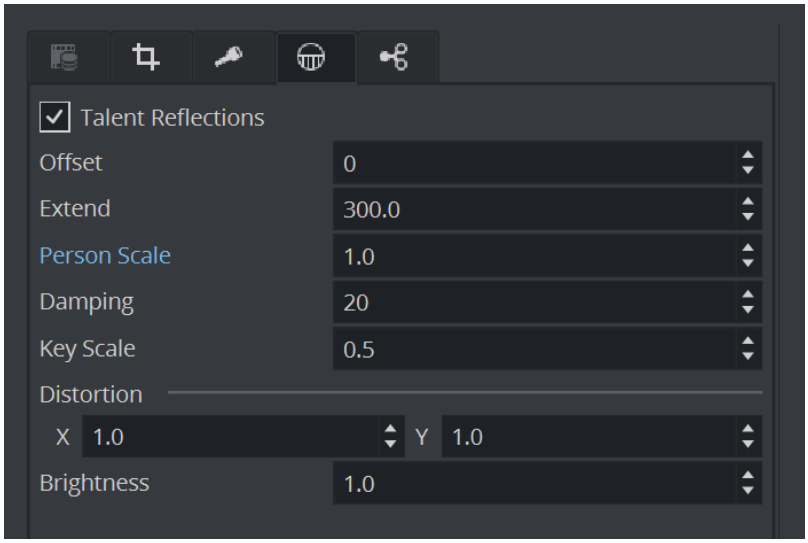
For a more photo-realistic rendering and a better mix between real and virtual elements, a real talent can reflect its shape onto a virtual floor. This can be done using the Viz Engine Render Pipeline in combination with the Precision Keyer and its possible to detect people.

This section explains step-by-step how to setup and modify a talent reflection in a Virtual Studio environment:

- Setting up Talent Reflection on the Precision Keyer
- Prepare the Scene Tree
- Modify the Plane Reflection Material
- Samples

Setting up Talent Reflection on the Precision Keyer

Make sure the Talent Reflections are turned on in the Talent Reflection Tab of the Media Asset property.



Note: In the Viz Engine Render Pipeline, only the Damping value is used.

As the Live signal is processed to detect the person(s), make sure that the Chroma signal is clean and does not contain any dust, noise or similar.

Example:

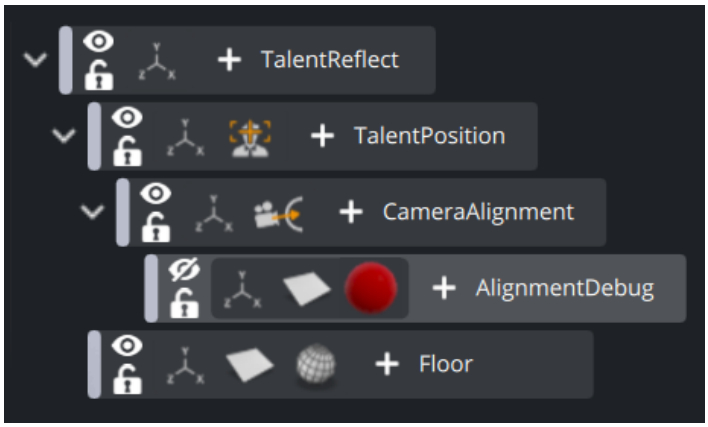
Copyright © 2026 Vizrt



Important: Make sure your Holdout Matte is set up correctly. Otherwise, objects other than the person in the studio can be recognized as a talent and results in jumping reflections.

Prepare the Scene Tree

The scene tree below shows a minimal setup of talent reflection in a virtual studio environment. The Talent Tracker plug-in is used to read the person's position from the Precision Keyer.



Copyright © 2026 Vizrt

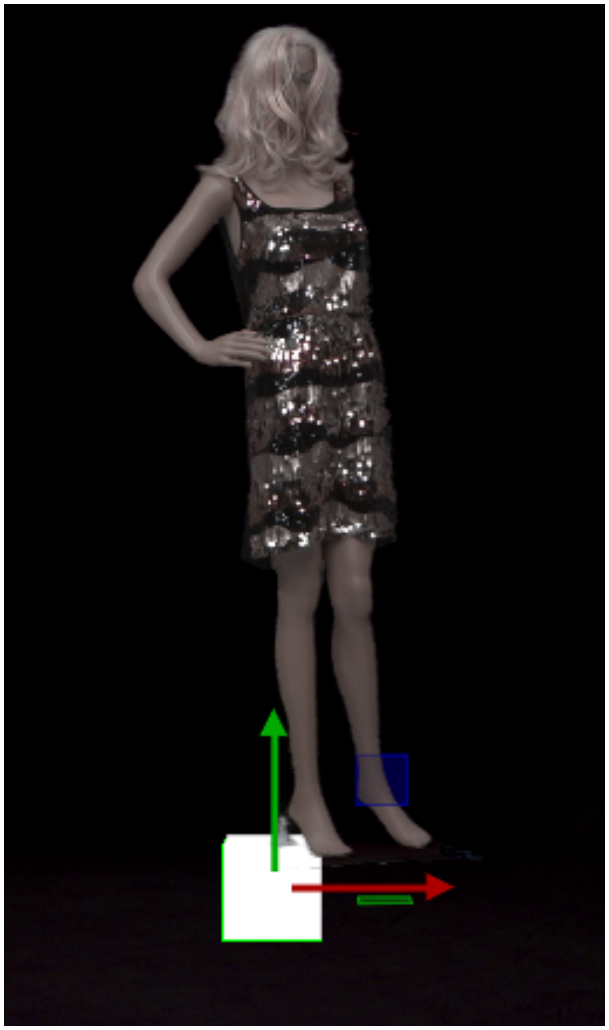
The plug-in on the *talentPosition* container is placed according to the recognized talent position. Each person has their own ID, the current implementation only allows to track one person, therefore set the Person ID in the plug-in to `1`.

> **Note:** Currently only one person can be detected and reflected.

Underneath the *talentPosition*, the *cameraAlignment* container is used to perform a billboarding towards our camera, meaning the container rotation is always looking towards the camera. The Look At plug-in performs this task for you. Set it to AutoRotation = Billboard and the Axis should be set to XY Screen.

Now this container's position represents the detected talent position, aligned to look straight to the camera.

For debugging purposes, you can place a geometry underneath the *cameraAlignment* container to visualize the correct position. In our example above, this is the debug cube.



The floor plane is used to render our reflection texture. Usually, it sits on top of our virtual floor, so make sure a z-sort is used to avoid z-fighting between the floor geometry and the plane for the talent reflection.

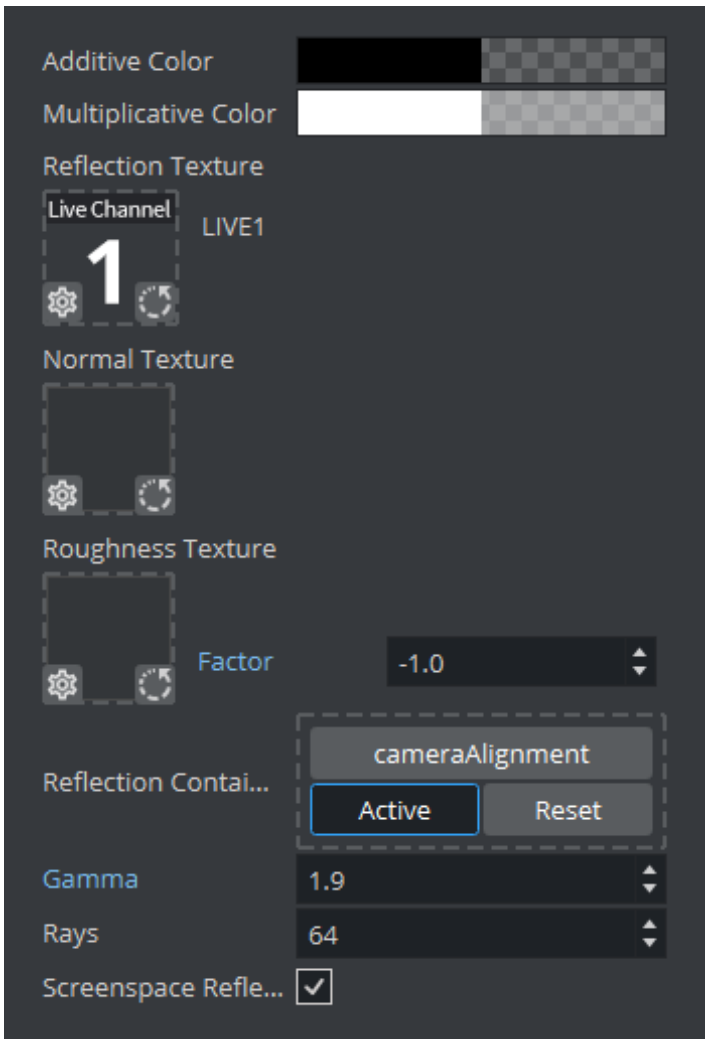
Copyright © 2026 Vizrt

Modify the Plane Reflection Material

Next, we need a plane with a Planar Reflection material. Do this by clicking on the plus to open the Plugin Overlay Panel , type `plane` and add the Plane Material Definition:



This adds a new material to your container allowing you to fine tune the appearance of your reflection.



This material renders your talent into the reflection container, so this container needs to be assigned to the **Reflection Container**. In our setup this is the aligned, billboarded container named *cameraAlignment*. The source also needs to be assigned, drag the *Live1* (respective the source used) onto **Reflective Texture.**

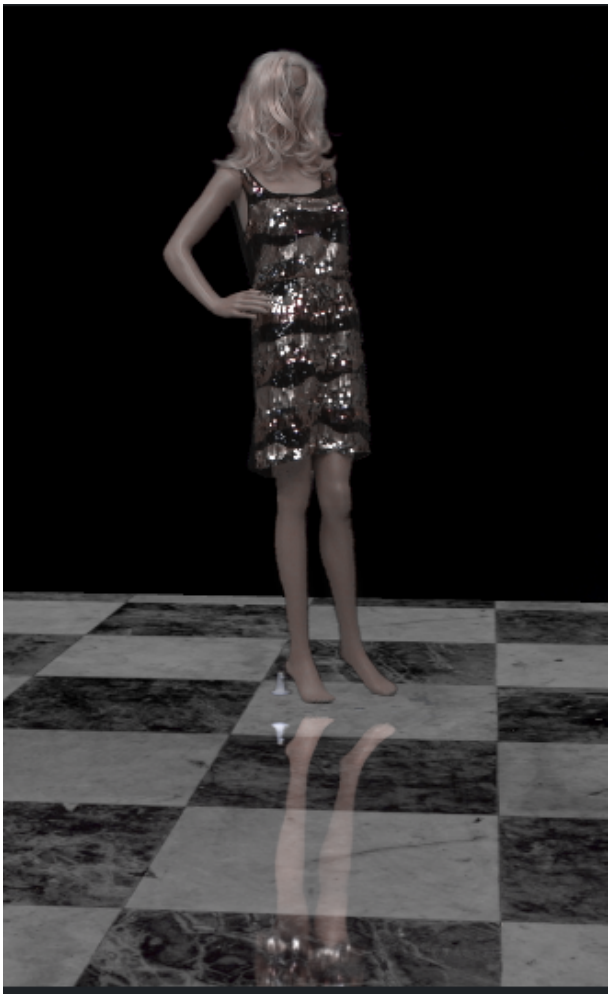
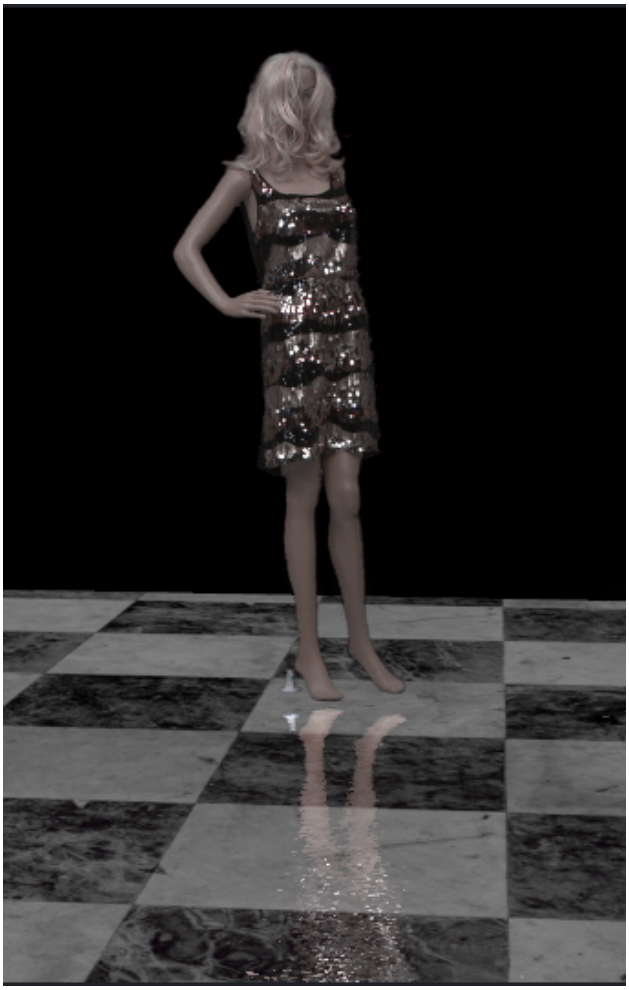
The next parameters are used to fine tune your setup:

- **Additive Color** and **Multiplicative Color:** Used as a base color respective multiplied to the color of your reflection is multiplied with.
- **Normal** and **Roughness Texture:** Define the properties of your floor (same as the textures used in the PBR Editor) (see the example screens below).

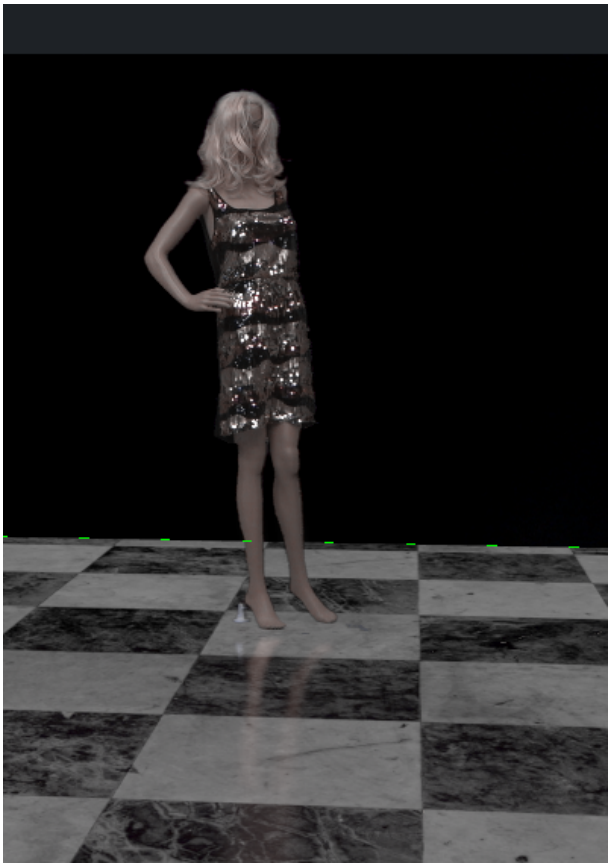
Copyright © 2026 Vizrt

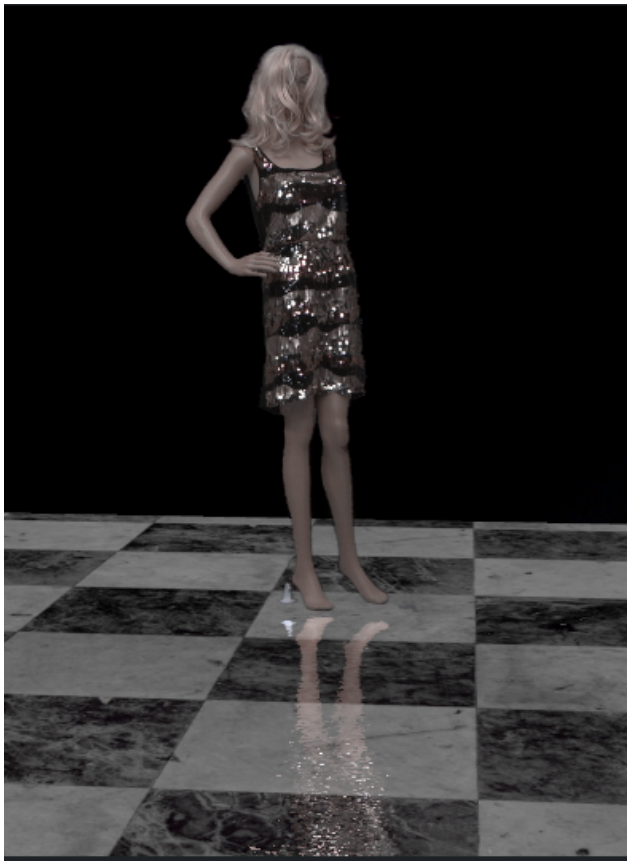
- **Gamma:** Adjusts the brightness.
- **Rays:** Fine tunes the quality of your talent reflection.
- **Screenspace Reflection:** Needs to be turned on for virtual sets.

Samples

Reflection Only	Reflection + Normal
	
Reflection + Roughness	**Reflection + Roughness + Normal**

Copyright © 2026 Vizrt





See Also

- Talent Reflection plug-in in the Global section of the Container plug-ins in the Viz Plug-ins User Guide.
- Talent Tracker plug-in in the Container section of the Container plug-ins in the Viz Plug-ins User Guide.

Copyright © 2026 Vizrt

5.16.5 Working with Holdout and Trash Mattes

The *Trash Matte* and *Holdout Matte* have changed between Viz Artist 3.x and Viz Artist 4.x.

Info: This feature requires Vizrt Virtual Studio/Tracking Hub Version 1.3.1 or later.

In previous Viz Artist versions, the trash-matte was created in VIZIO or Tracking Hub and then transferred to Viz Engine via command interface.

In Viz Artist, the trash-matte and holdout-matte are defined in a scene. This scene must be a scene using the Classic Render Pipeline, it should have active tracking (set necessary cameras to *REMOTE*) and lens distortion activated (if needed). It is rendered like the main scene, except that the result is not visible but only used to define trash matte, Cyc and holdout matte.

The color channels are interpreted as follows:

- **Red:** Trash matte.
- **Green:** Cyc (unused).
- **Blue:** Holdout matte.
- **Cyan:** Unprocessed holdout area.

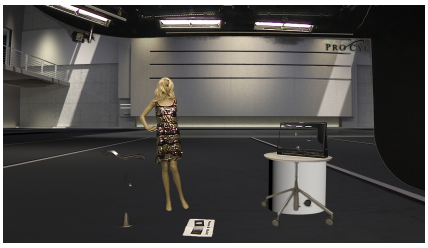
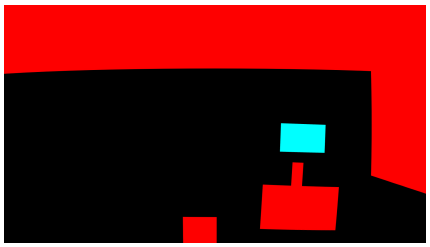
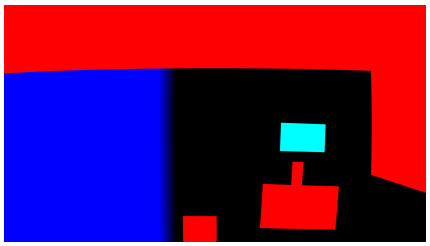
If a holdout matte is set, the green channel is interpreted to select unprocessed (original LIVE input) vs processed (for example, despilled) input for the holdout area. `Green=0` results in fully processed holdout area, `Green=255` results in a unprocessed holdout area.

You can either manually define your scene in Viz Artist (verify correct camera settings) or use Tracking Hub, which provides a plug-in and a GUI to control the mattes.

See the following example with different matte scene scenarios:

	Matte Scene	Result
Original live input.		
Virtual set only, no keying.		

Copyright © 2026 Vizrt

	Matte Scene	Result
No matte scene used, studio lights and obstacles visible, monitor keyed.		
Add trashmatte to mask out lights on top and black area on the right. trashmatte = RGB(255, 0, 0)		
Add trashmatte to mask out obstacles on bottom, add holdout matte to prevent keying of image shown on monitor. unprocessed holdout area = RGB(0, 255, 255)		
Add holdout matte to create unkeyed but color processed area, so that the talent can move around and skin color does not change between keyed and unkeyed area. color-processed holdout area = RGB(0, 0, 255)		

Tracking Hub



Tracking Hub can directly modify this scene via the command interface.

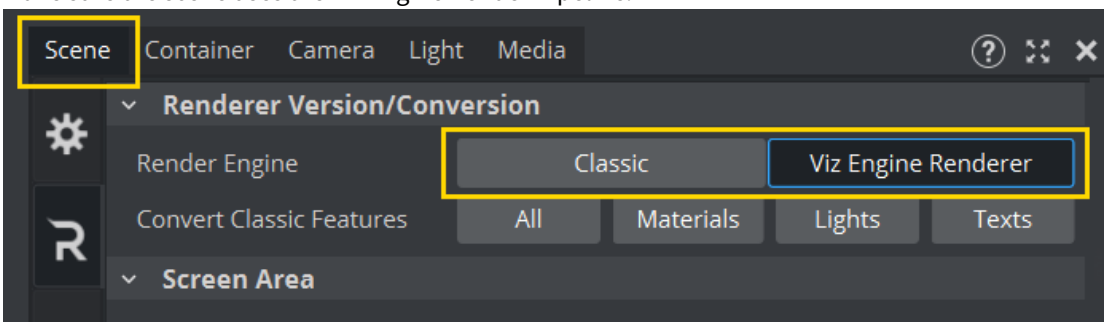
The new geometry plug-in *Cyc* is responsible for rendering the special geometry needed to represent a back wall, one corner and two corner cyclotron. It is now possible to define an overlap size, which blends the hard edges softly into the key.

Creating a Trash Matte Scene

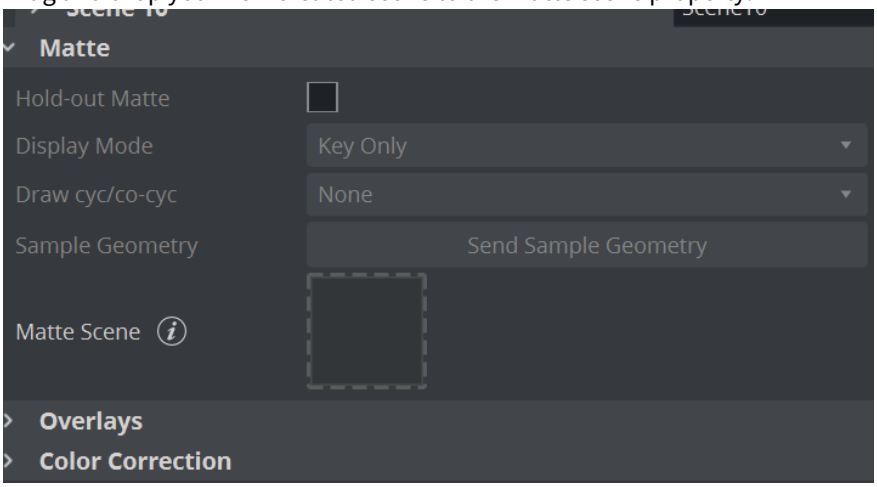
- Create an empty scene in Viz Artist. This scene MUST use the Classic Render Pipeline.

Copyright © 2026 Vizrt

- Drag and drop the Cyc plug-in into the scene.
- Make sure that the *Use Lens Distortion* flag is checked and the camera is on remote.
- Save the scene under a name of your choice and remember the location in the server.
- Open your virtual studio scene.
- Make sure the scene uses the Viz Engine Render Pipeline.



- Go to the scene settings for Virtual Studio.
- Drag and drop your new created scene to the matte scene property.

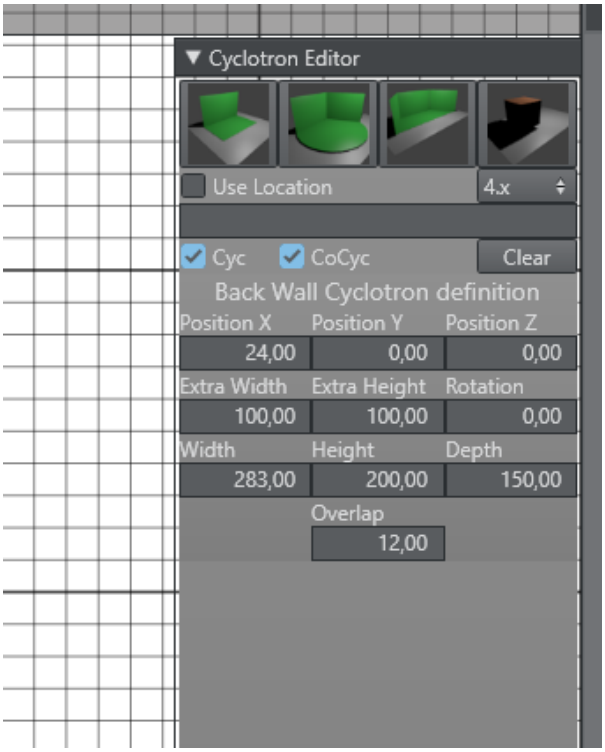


- Set the cameras to **Remote**.
- **Save** your virtual studio scene.
- All position and size changes can then be done in the Studio Manager.

Adjusting the Trash Matte from Studio Manager

- Open the Studio Manager.
- Open the new Cyc editor.
- Create a new Cyc model.

Copyright © 2026 Vizrt



- Set the Render Pipeline to Viz Engine.
- Leave the **Use Location** check box empty! This will explained later.
- Modify the parameters according to your Cyc model.
- The meaning of the parameters is explained in the Tracking Hub User Guide.
- When finished, save the studio. The definitions are then saved into the configuration files.

Copyright © 2026 Vizrt

Selecting the Workflow



Viz Artist 4.0 introduced a new workflow with the Cyc scene as scene property in the scene settings. The dimension and position of the Cyc was sent direct to the Cyc plug-in in the scene. In Viz Artist 4.1, this workflow was enhanced by using the shared memory implementation. From Viz Artist version 4.1 and later, the settings of the Cyc are stored in the shared memory. Therefore, it does not matter where the Cyc scene is located. To adjust the Cyc visible in the editor, the user can load the scene into the main layer and watch the geometry change. Later on, this scene can be moved into the scene property of the Virtual Studio scene and used as in Viz Artist 4.0.

In the Cyc editor of the studio the user can now choose between three modes. The old 3.x workflow, the 4.0 workflow and 4.1 workflow with shared memory. Don't forget to adjust the workflow according to the Viz Artist version used.

Copyright © 2026 Vizrt

5.16.6 Projected Virtual Window

- Prerequisites
  - Tracking
  - Getting Started
  - Content Scene
  - Playout Scene
- Setup Guide
  - Viz Engine
  - Content Scene
  - Playout Scene
  - Setup Scenarios
  - Unreal Engine
- Color Correction

Prerequisites

Tracking

Make sure Tracking Hub is set up properly and the Engine receives the correct tracking data, since tracking data is mandatory for a proper virtual window representation.

Getting Started

To create a projected Virtual Window, at least two scenes are necessary. The first one contains the actual content of the scenery outside the Virtual Window, the second one contains a virtual representation of the video walls position and its dimensions in the actual studio.

Content Scene

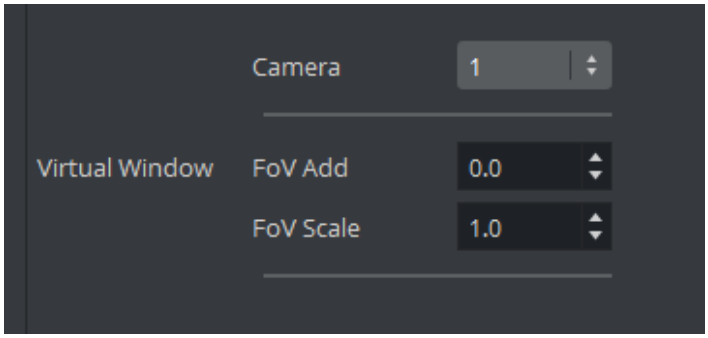
The content scene can be anything, best practices creating a set for a Virtual Studio apply. The content scene is used within a GFX Channel, so keep that in mind as well when designing the scene.

Camera Settings

Configuration

When the content scene is being rendered with the tracking data's Field of View, it happens that the virtual window gets cut off too early when moving the real camera. To mitigate that problem, **FoV Add** and **FoV Scale** can be adjusted for each remote camera in the Engine's **Camera** configuration:

Copyright © 2026 Vizrt

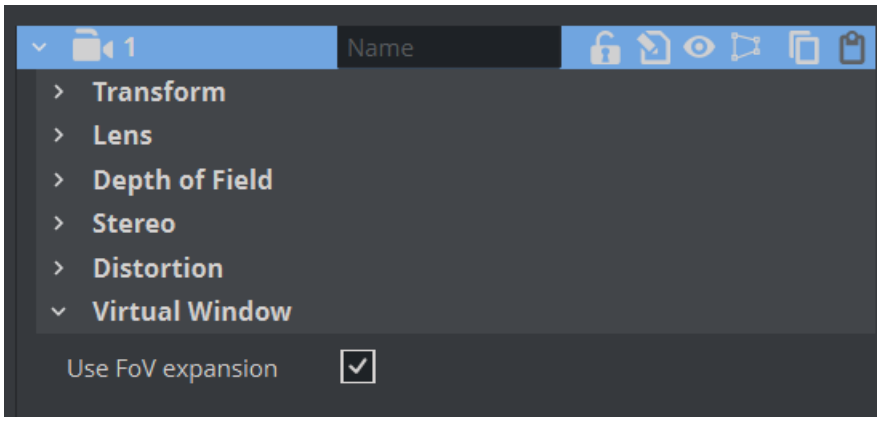


- **FoV Add:** Degrees added to the incoming Field of View.
- **FoV Scale:** Factor the incoming Field of View is multiplied with.

Both values can be changed on the fly, no restart of the Engine is needed. Make sure to save the configuration after changing the values.

Scene

Additionally to configuring those values, it is necessary to enable their usage within the scene's camera by ticking **Use FoV expansion**.



FoV Calculation

The final FoV that is being used for rendering is: *FoV(final) = FoV(Tracking) * FoV_Scale + FoV_Add*.

Playout Scene

The playout scene needs a virtual representation of your Video Wall in the form of a geometry and the Virtual Window plug-in. It is crucial to know the exact placement and dimensions of the video walls relative to the origin of your virtual studio (tracking data).

Copyright © 2026 Vizrt

Virtual Window Plug-in

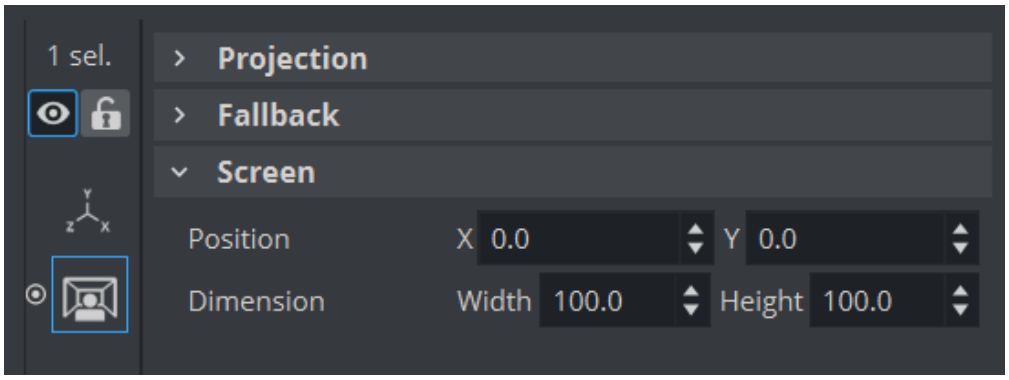


The Virtual Window plug-in is responsible to render your virtual window geometry and generate the output.

It is supported by both the Viz Engine Render Pipeline and the Classic Render Pipeline. The usage is a bit different depending on the selected pipeline.

Settings

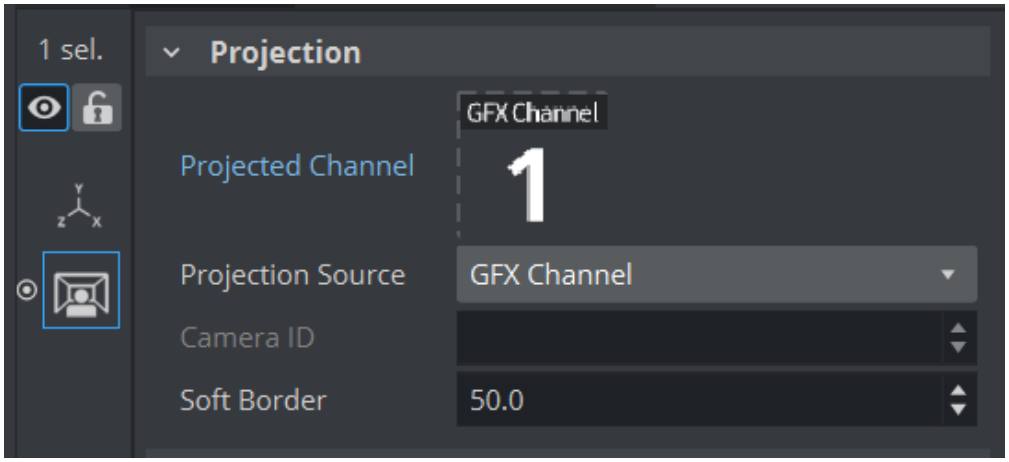
Screen



The screen settings define the area where your virtual window is rendered on the final Viz Engine output. This allows to define the region of the screen that is used to render the virtual window content. It defines the flat 2D region which represents either a whole video wall (X/Y = 0.0 Width/Height = 100.0) or just a part of it when multiple windows are needed within a single Viz Engine output (for example when a curved or tiled wall is used in the studio). It ultimately makes it possible to use more than one Virtual Window plug-in from within a single scene and drive multiple virtual windows with a single setup.

- **Position:** The position of the rectangle in percentages of the output resolution.
- **Dimension:** The dimension of the rectangle in percentages of the output resolution.

Projection



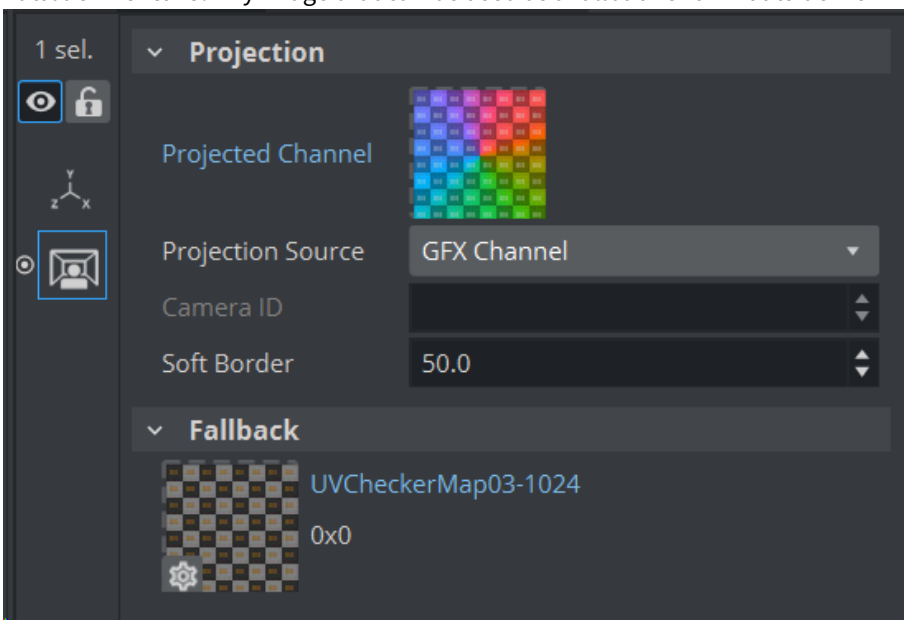
Copyright © 2026 Vizrt

The projection settings are responsible to link the content to the output scene.

- **Projected Channel (Viz Engine Renderer only):** Slot where the GFX Channel that renders the content scene must be added to.
- **Projection Source:** The virtual window plug-in needs to know camera settings the content was rendered with. **GFX Channel** automatically reads the data from the assigned GFX Channel, **Camera** allows to specify a camera via Camera ID.
- **Camera ID:** The camera number from 1-16 where the plug-in reads camera the data from. Only valid when Projection Source is set to **Camera**.
- **Soft Border:** Adds a fade effect on the borders of the projected window.

Fallback

- **Fallback Texture:** Any image that can be used as a fallback shown outside view frustum:



Copyright © 2026 Vizrt



Multi Panel Plug-in

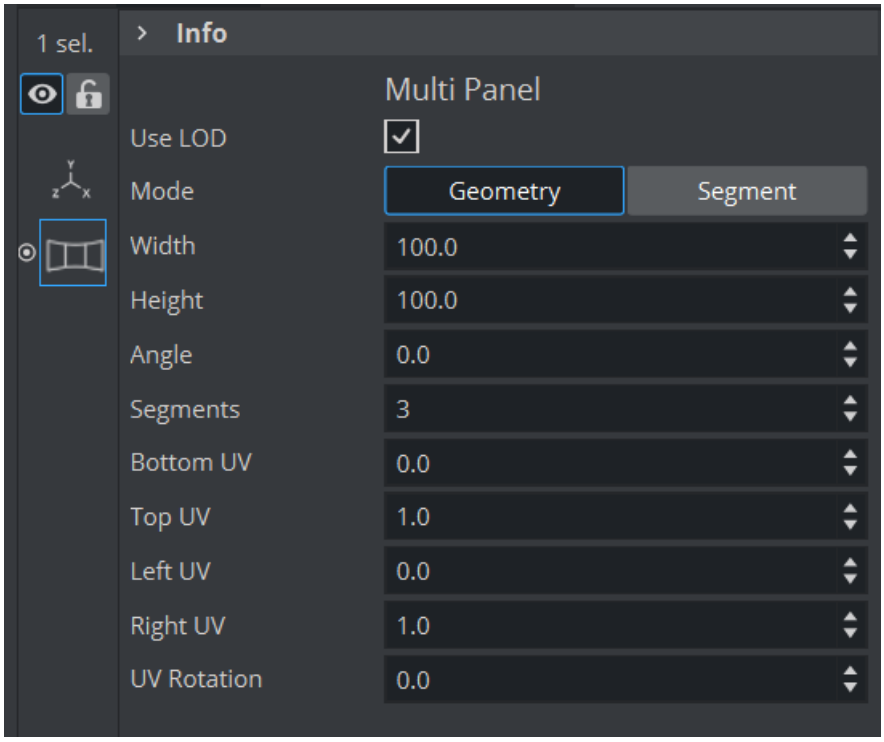


The Multi Panel plug-in is a geometry plug-in that helps create the virtual representation of the video wall setup. It allows to create multiple different shapes from rectangular to curved ones. The plug-in contains the actual video walls dimensions and its alignment in the real studio.

This plug-in can create multiple segments within the same geometry, easily allows it to have rotated tiles seamlessly merged together if the dimension of the tiles are the same (symmetrical video wall setup).

Settings

Copyright © 2026 Vizrt



Mode: Defines how the properties are being used to generate the video wall geometry:

	Geometry	Segment
Width	Width of the geometry in cm.	Width of a single segment in cm.
Height	Height of the geometry in cm.	Height of a single segment in cm.
Angle	Rotation angle around Y-axis in degrees.	Angle in degrees between 2 consecutive shapes.
Segments	Number of subdivisions.	Number of consecutive segments.
Bottom UV	UV coordinate for the lower edge where y = 0.	UV coordinate for the lower edge where y = 0.
Top UV	UV coordinate for the upper edge where y = Height.	UV coordinate for the upper edge where y = Height.
Left UV	UV coordinate for the left edge where x = 0.	UV coordinate for the left edge where x = 0.

Copyright © 2026 Vizrt

Right UV	UV coordinate for the right edge where x = Width.	UV coordinate for the rightmost edge where x = Width * Segments.
UV Rotation	Rotation Angle around the UV center.	Rotation Angle around the UV center.

Setup Guide

Viz Engine

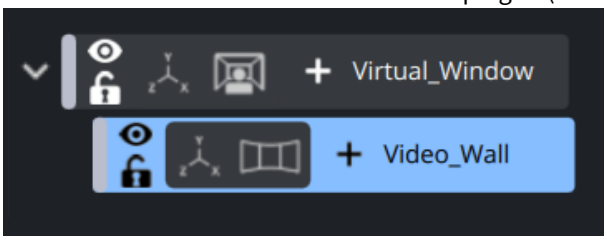
- Start with the Video Wall configuration.
- Enable Tracking Hub and make sure you are receiving the tracking data.
- Configure the Virtual Window FoV Add, set it to 5 for the tracked cameras as a start.
- Render scale must be 1.0.
- Distortion and Parameter mode needs to be setup correctly for the cameras as well. Most likely it is Internal / Tracking Hub.

Content Scene

- Start with the set you want to use as content.
- Cameras need to be set to Remote.
- Enable the **Use FoV Expansion** Flag on the remote cameras.
- Save the scene and close it.

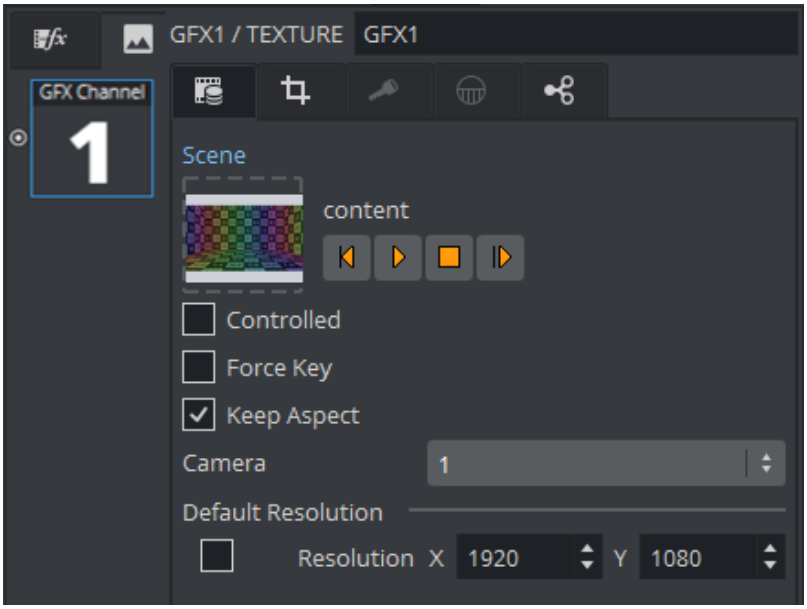
Playout Scene

- Start with an empty scene (or any scene you use for playout and want to modify).
- Add a container with Virtual Window plug-in (**virtual_window**).
- Add a sub-container with the Multi-Panel plug-in (**video_wall**).

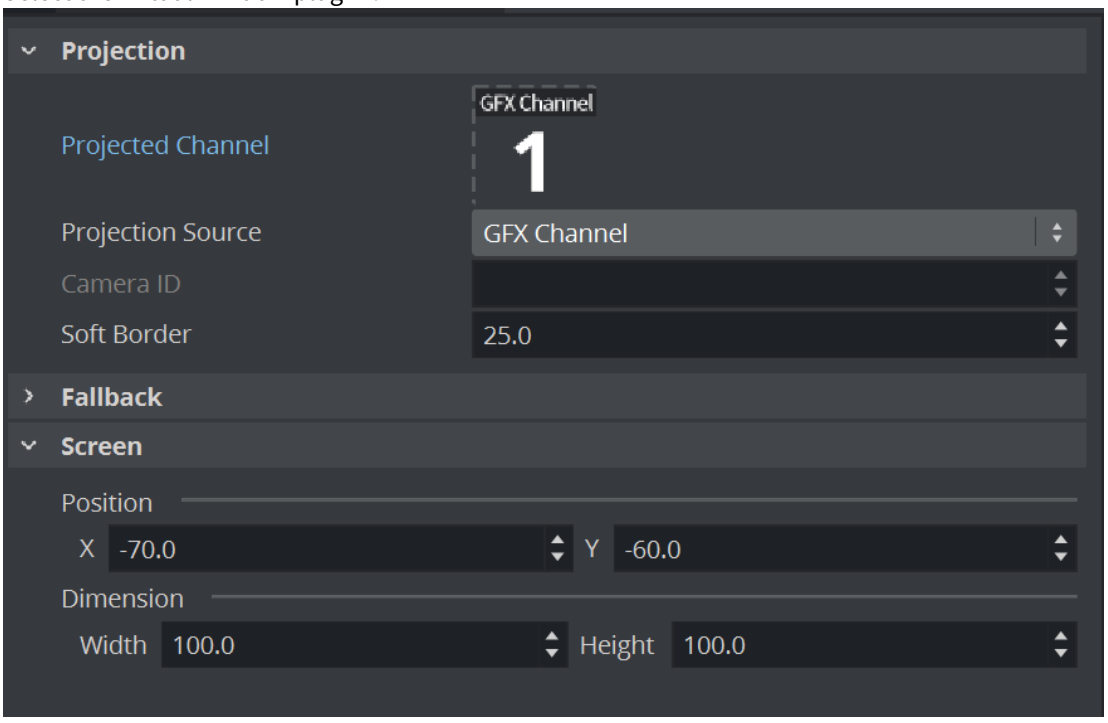


- Add a GFX Channel as Texture, set the Scene.

Copyright © 2026 Vizrt



- Make sure the GFX Channel's Camera is set to the camera that is tracked. Don't use *Auto* or *Main Scene,* because the Playout scene most likely uses a different camera.
- Manually set the Resolution of the GFX Channel. This should match the resolution of your real camera, most likely HD.
- Select the Virtual Window plug-in.



- Set the content scene's GFX Channel to Projected Channel.
- Set the Projection Source to GFX Channel.

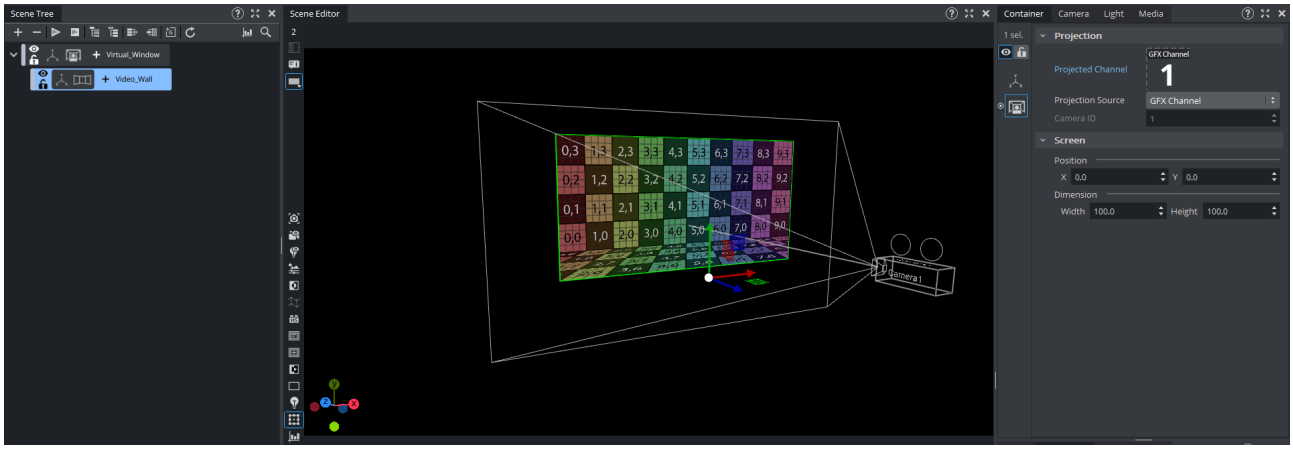
Copyright © 2026 Vizrt

- Select the Container with the Multi Panel plug-in and configure the video wall geometry (position, translation) according the real worlds videowall.

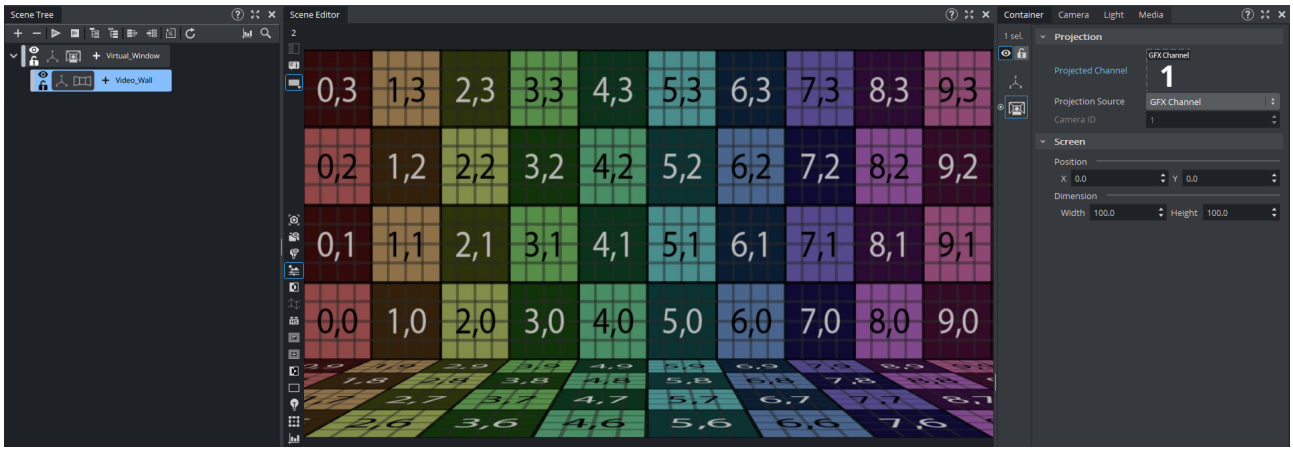
Scene Editor

The final rendering behavior is only done when the **Post Processing Effects** are enabled.

Post Processing Off



Post Processing On



Setup Scenarios

Scenario 1: Single Tile Video Wall

The single tile video wall is showing the virtual window on the whole output from, meaning the Virtual Window plug-in is set to Position X/Y = 0.0 on full scale (Width/Height = 100.0)

The multi panel plug-in contains only one segment, having the physicals video wall dimensions for height and width set.

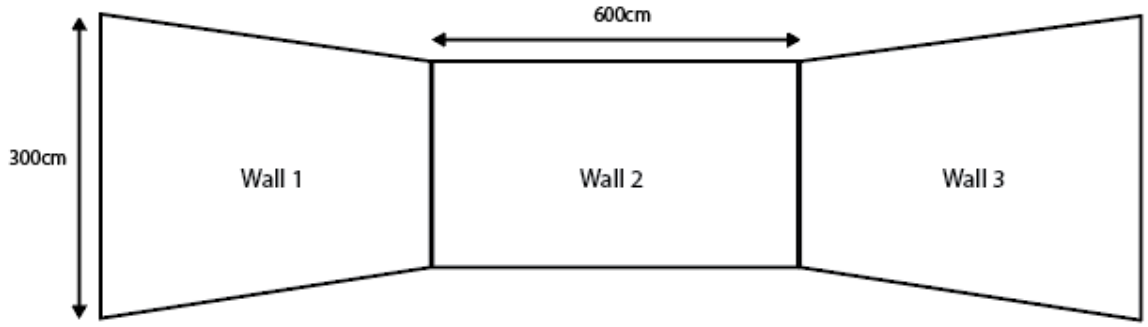
Copyright © 2026 Vizrt

In case a rotation angle is needed, the actual containers translation parameters can be adjusted to fit the geometry in 3D space.

Scenario 2: Multiple Symmetrical Tiles Video Wall or Curved Video Walls

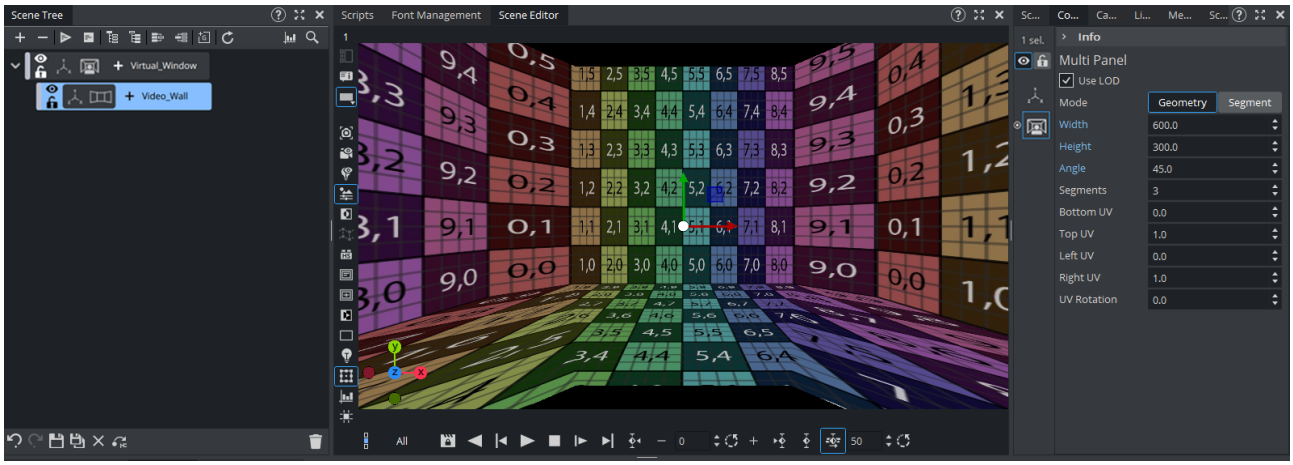
If the real videowall consists of multiple tiles which share the same dimensions a single Multi Panel plug-in, it can handle the whole wall setup. This also can be used when working on curved videowalls where the curved gets interpreted by creating multiple rotated tiles stitched together.

Example A: Three Tiled Wall



This example has three physical video wall segments in the studio which have the same size of 600cm x 300cm.

The setup within the playout scene can be done by using one Videowall plug-in in combination with the Multi Panel plug-in containing three segments, rotated 45 degrees to each other.



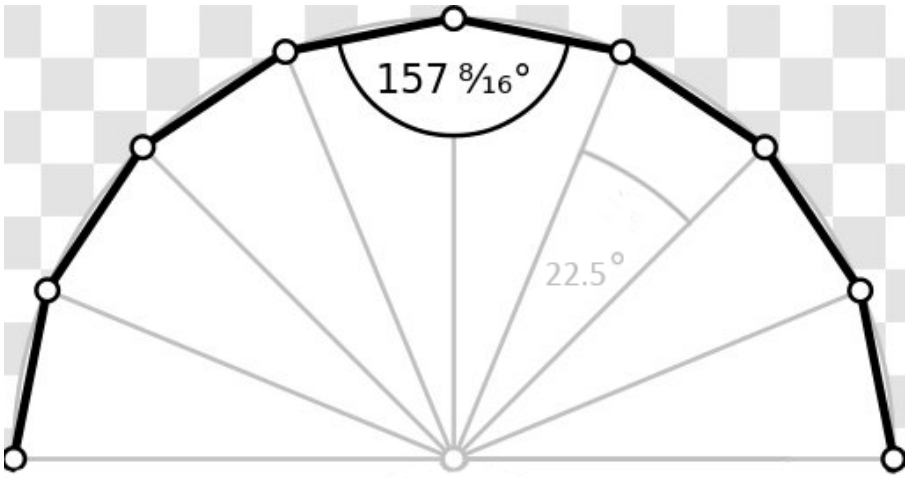
In this scenario, the Multi Panel plug-in is set to geometry mode, containing of three subdivisions having the same dimensions.

Example B: 180 Degree Curved Wall

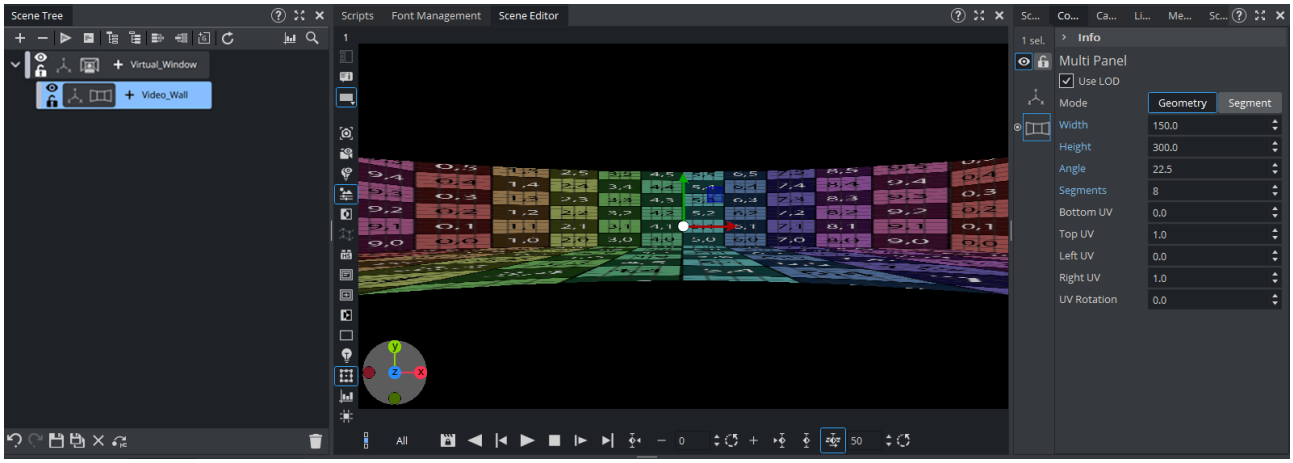
Copyright © 2026 Vizrt

In order to simulate a curved wall, the amount of divisions within the Multi Panel plug-in needs to be increased to have a more realistic looking output on the real wall.

For a 180 degree curved wall using eight subdivisions should be sufficient.



Depending on the size of the real videowall the division dimensions need to be calculated accordingly. This example has each division set to a width of 150cm by keeping the height of 300cm.

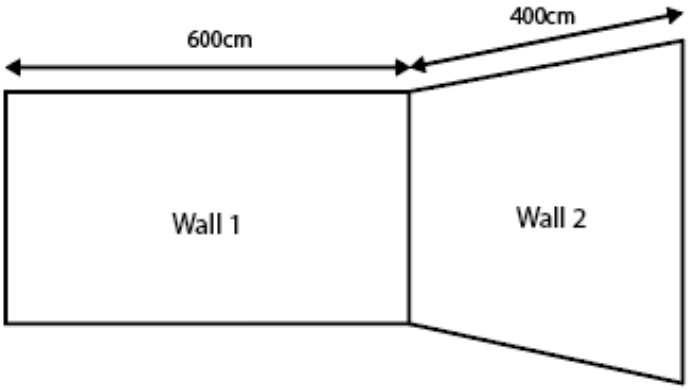


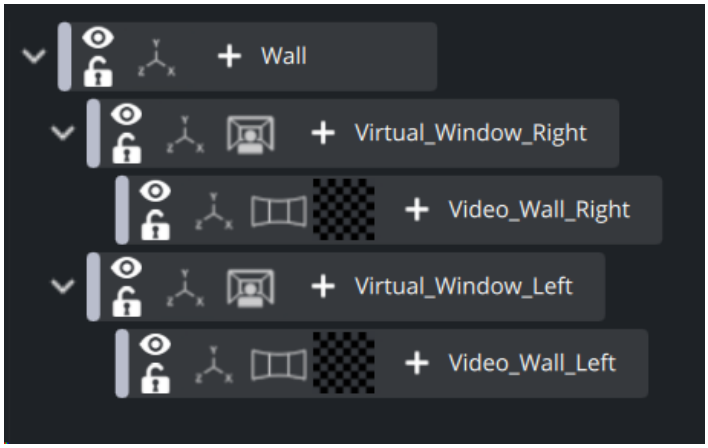
Note: These screenshots are taken by disabling the remote camera feature for explanation purposes. The correct output can only be properly seen when using a remote camera.

Scenario 3 - Non-symmetrical Video Wall

A typical video wall setup can contain two walls having different dimensions. In this case, the setup needs to have two virtual windows set up in the scene tree.

Copyright © 2026 Vizrt

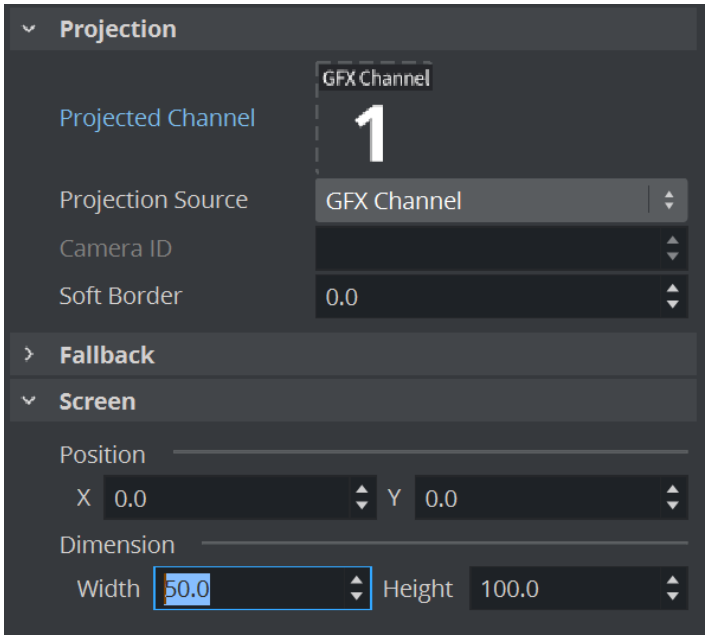


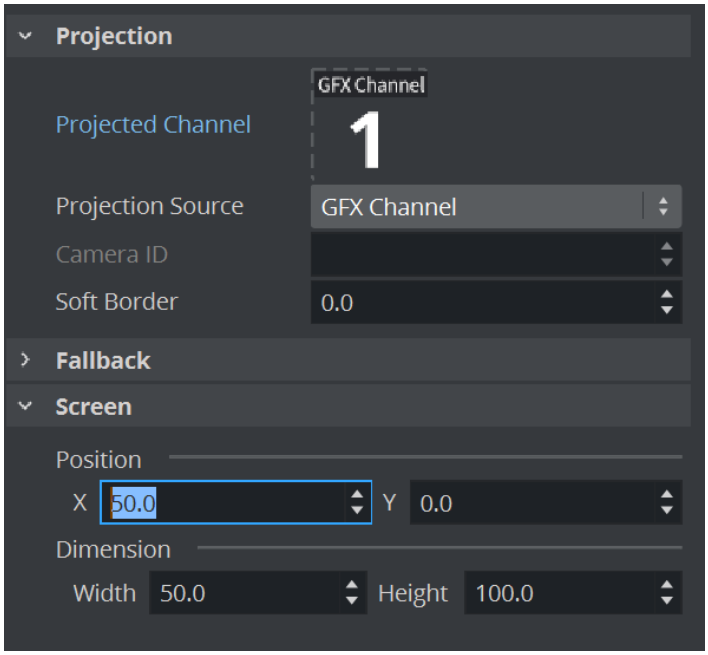


In this case, each of the Virtual Window plug-ins need to be resized to only match 50% of the Viz Engine output. This depends on how the overall NVIDIA Mosaic or the videowalls image processor is configured. Typically having this is set up by assuming this is a split 50-50 screen setup.

Setup of the plug-ins (left / right):

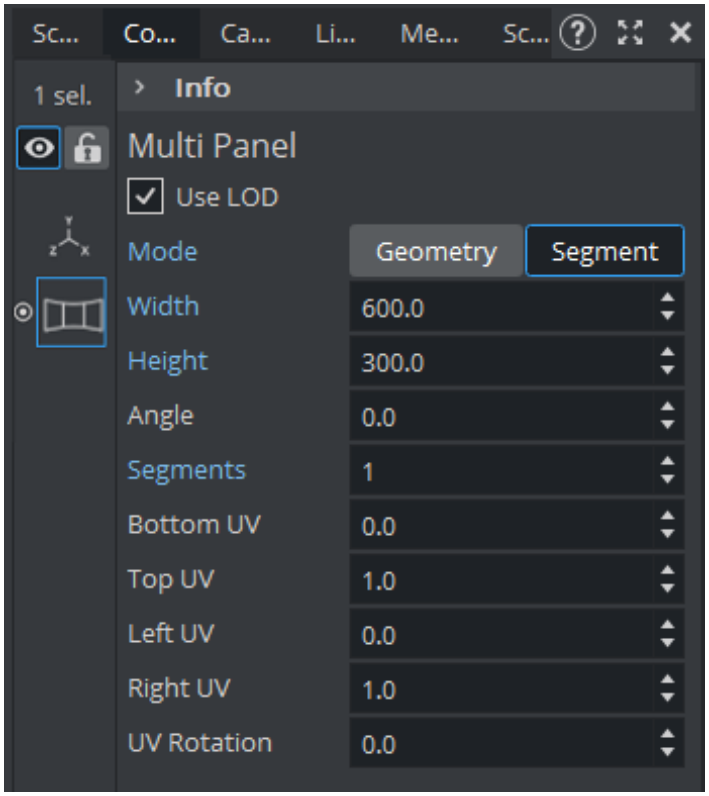
Copyright © 2026 Vizrt

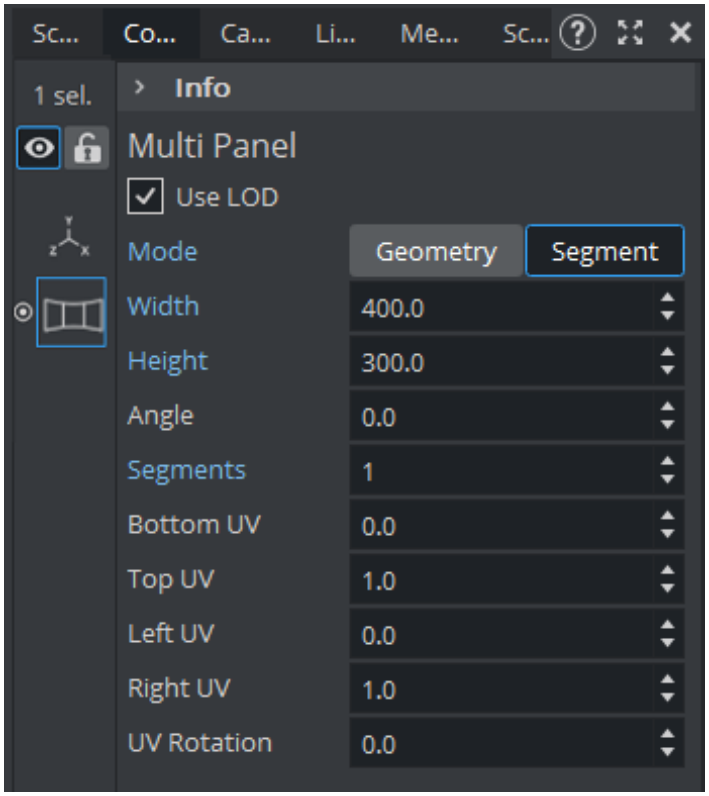




Now each virtual window gets a Multi Panel plug-in assigned, containing the real dimension of the corresponding physical wall in the studio. Using a non-symmetrical setup is done by choosing the **Segment** mode inside the Multi Panel plug-in.

Copyright © 2026 Vizrt

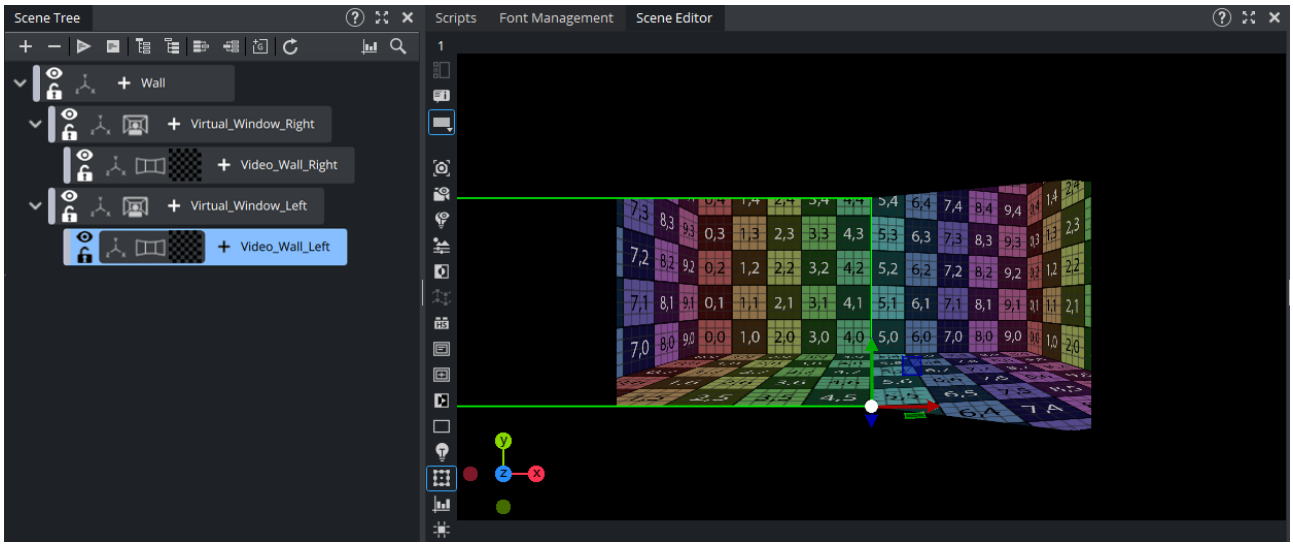




Copyright © 2026 Vizrt

As soon as the dimension of the walls are set, the containers/geometries can be placed correctly in the 3D world. This of course has to be done according to its real position in the studio. All measurements have to be done related to the real studios Zero Point (point of reference).

Turning off the Post Effect view in Artist, helps to easily adjust the containers position.

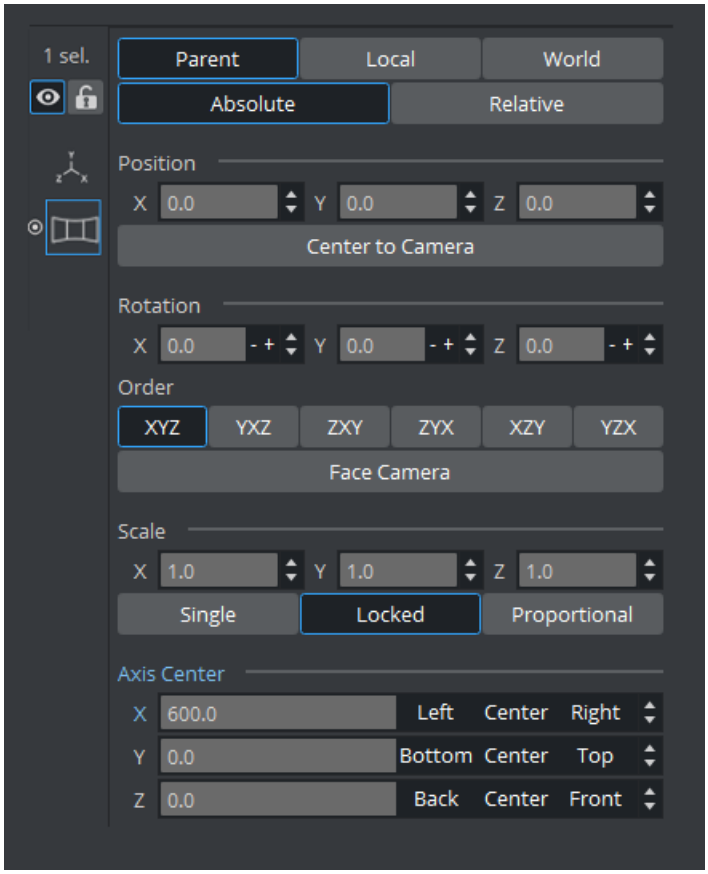


Rearranging the containers Axis center helps it to stitch both geometries together seamlessly. In this example the left wall has its axis center set to right, the right wall is set to left while both container have set their position to X = 0, which makes them touch each other directly on the studios zero point.

Since the right wall also has an angle, this can easily be added within the containers translation properties. The angle within the Multi Panel plug-in does not have to be adjusted.

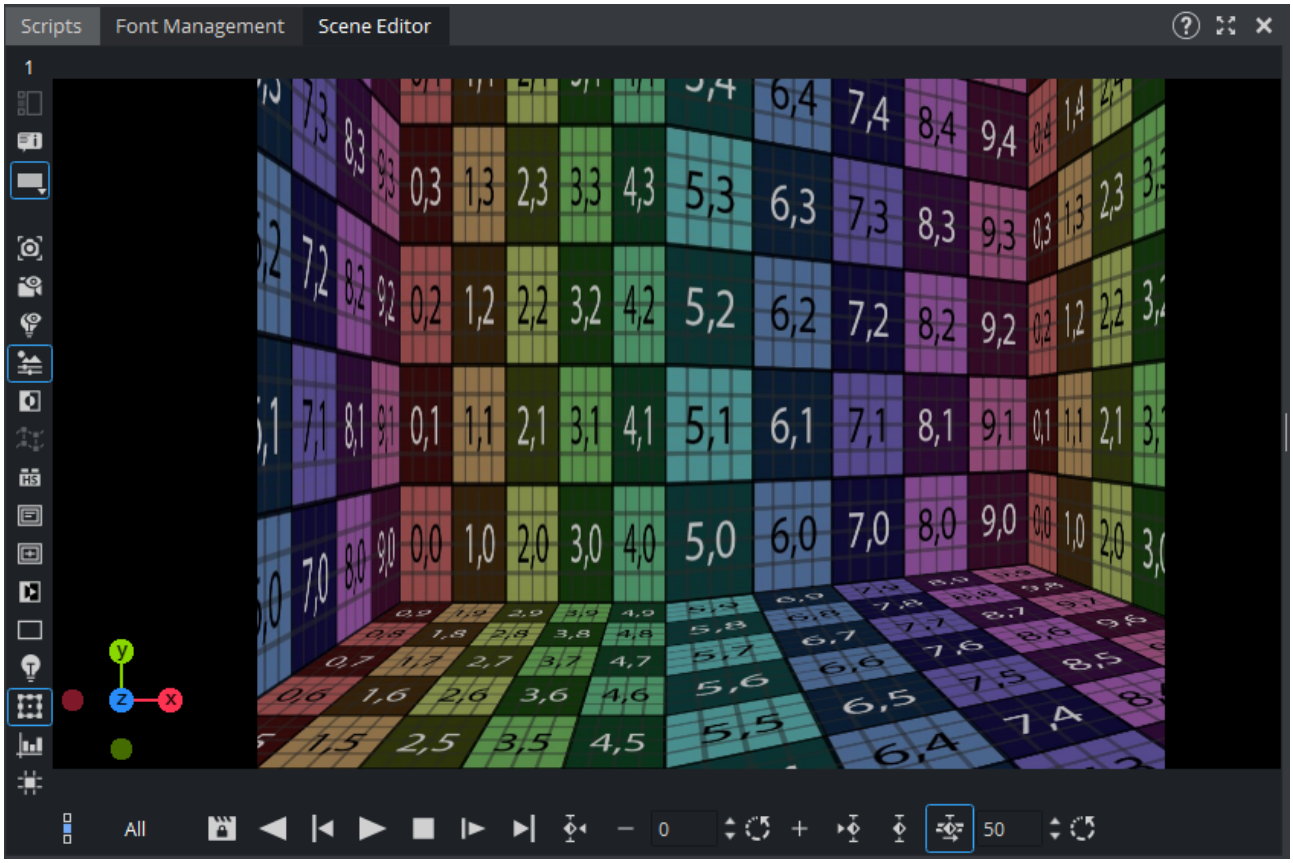
This example has the left wall rotated for 30 degrees:

Copyright © 2026 Vizrt



The actual Viz Engine output (show the post effects) now projects two walls each using 50% of the screen. Depending on the cameras angle within the tracked content scene, the two walls with adapt while still being stitched together seamlessly in the center.

Copyright © 2026 Vizrt

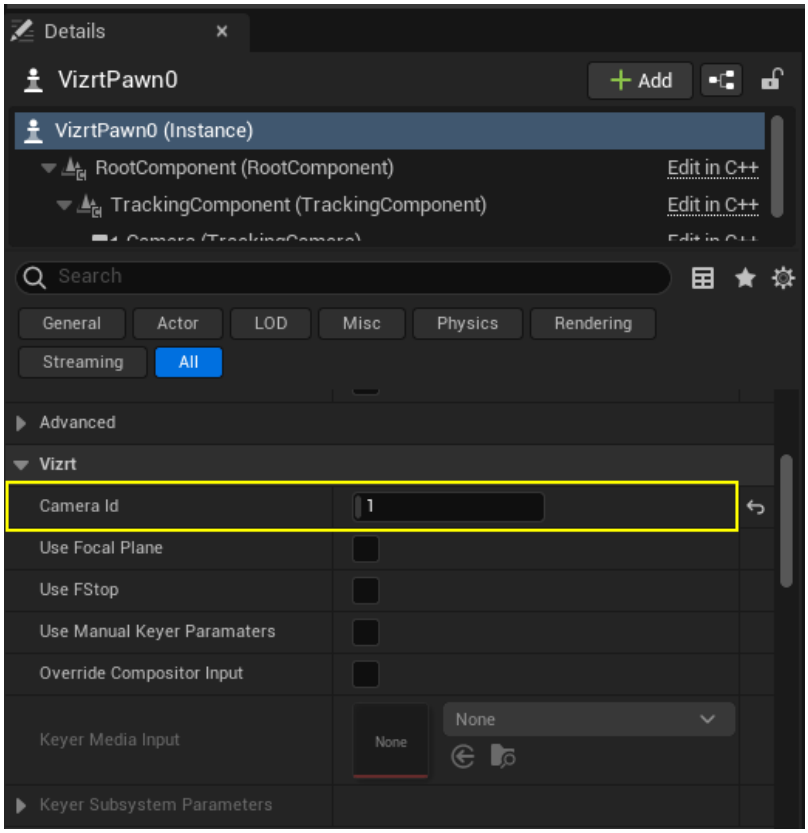


Unreal Engine

Content Scene

Follow the Unreal Engine guide to setup a Viz Virtual Studio scene without the compositor. The VizrtPawn's camera number needs to be set to the one receiving the tracking data:

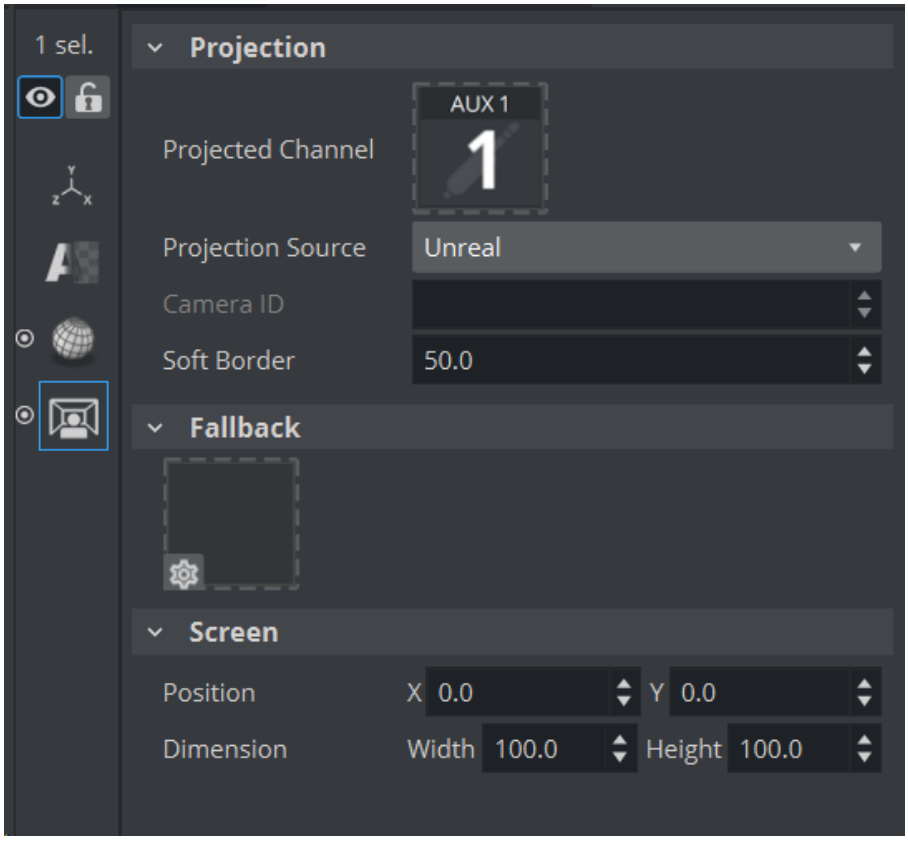
Copyright © 2026 Vizrt



Playout Scene

The setup of the playout scene is the same as using Viz Engine native content, except that the Virtual Window plug-in needs to use the AuxChannel instead of GfxChannel. By choosing the UNREAL option inside the Projection Source menu, Unreal Engine then sends back the correct matrices and parameters for the tracking to work properly.

Copyright © 2026 Vizrt



Warning: It is crucial to have the camera rendering delay configured correctly, otherwise there will be jitter on the videowall when moving the camera.

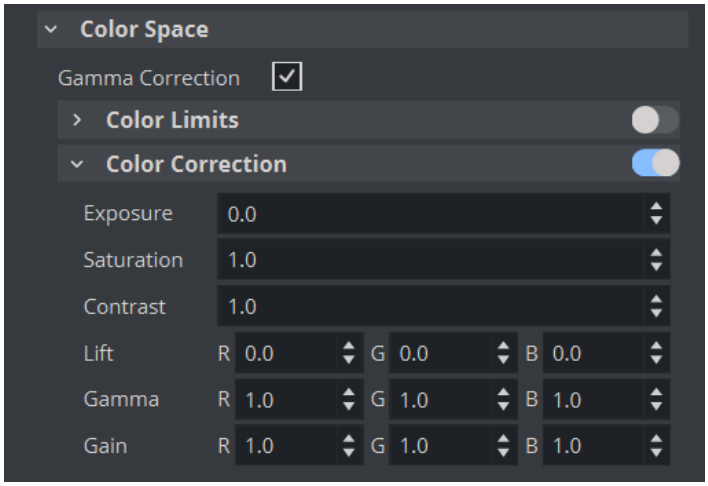
Color Correction

For Virtual Window workflows where multiple inputs and scenes may be needed, it's possible to independently control their color correction settings.

Content Scene

The color correction parameters can be changed by opening the content scene, and going to the **Scene** tab, inside **Color Space > Color Correction**.

Copyright © 2026 Vizrt



Playout Scene

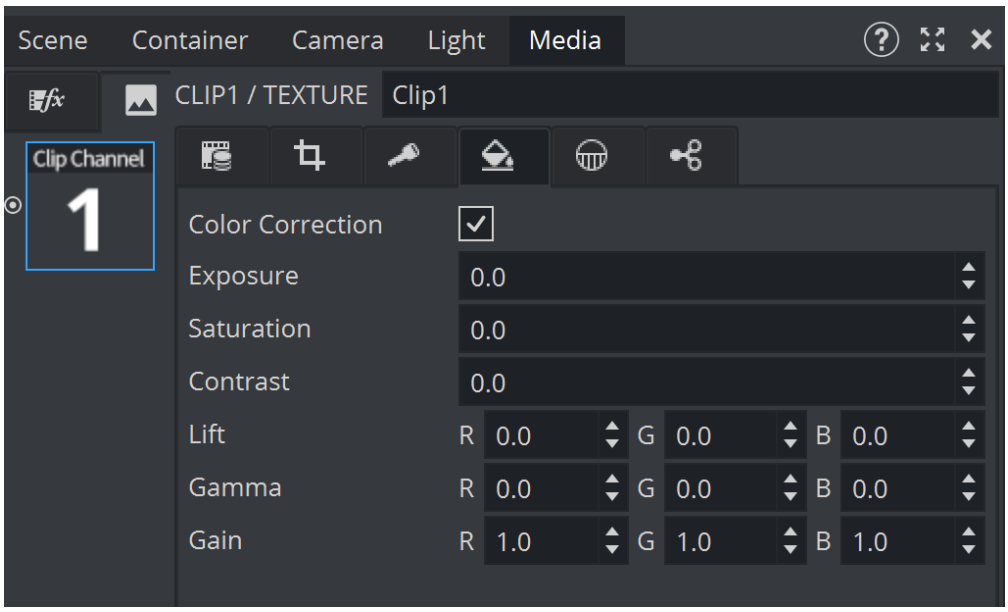
The color correction/tonemapping parameters can be changed by opening the playout scene and going to the **Scene** tab inside **Color Space > Color Correction** and **Color Space > Tonemapping.**



Copyright © 2026 Vizrt

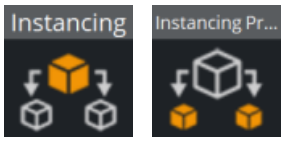
Video/Clip Input

The color correction parameters for clips/inputs can be changed in the **Media** tab, by selecting the corresponding channel and checking the **Color Correction** checkbox.

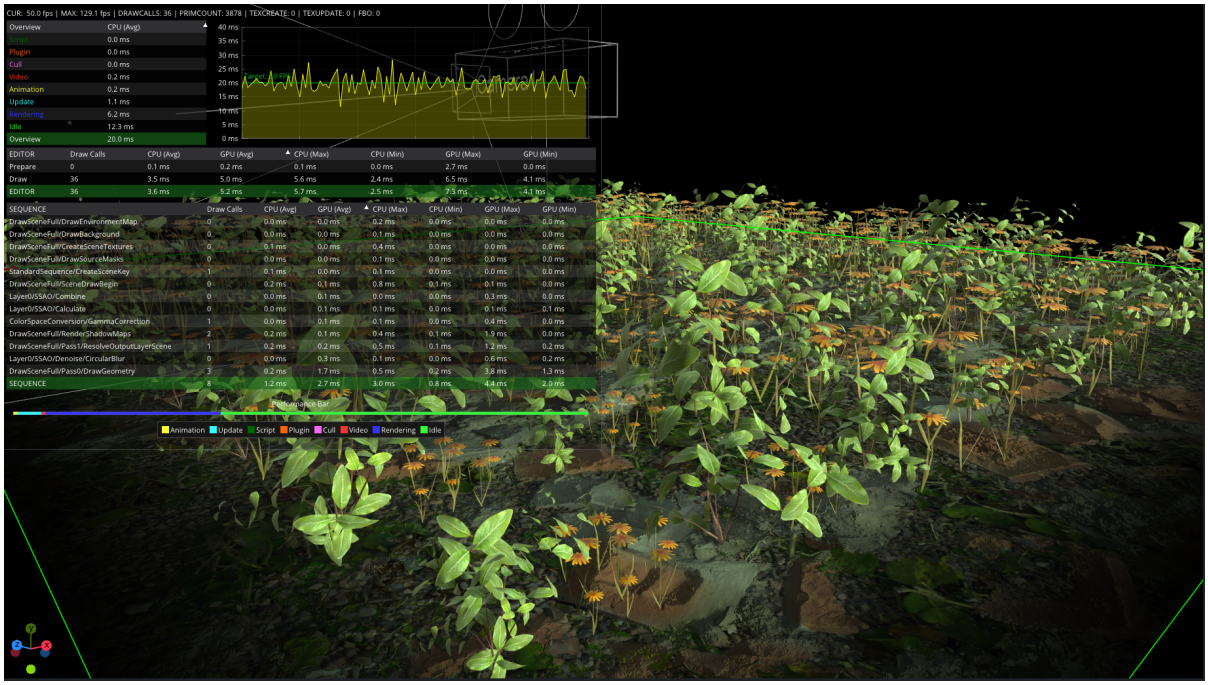


Copyright © 2026 Vizrt

5.17 Instancing



Viz Engine version 5.0 introduced an efficient way to render a single object multiple times. It requires a minimum amount of drawcalls to the GPU and reduces the time to render these objects dramatically. The required information is uploaded to the GPU only once, which allows to draw a lot of copies very efficiently.



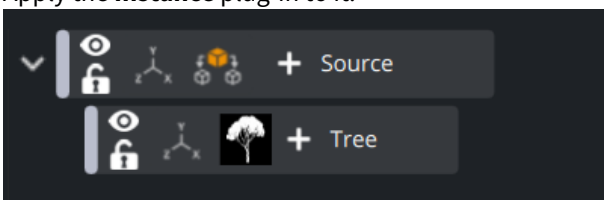
To setup this type of rendering, two new plug-ins are required, an **Instancer** and a **Producer** plug-in.

Icon	Description
	Defines the geometry and look for our setup. Put this plug-in on your geometry to be instanced. You need to define the container holding the corresponding Producer plug-in in the User Interface.
	Uses the defined geometry and distributes them among its subcontainers.

Copyright © 2026 Vizrt

5.17.1 Example

- Create a sample geometry, like in our example a Tree and put it as a Subcontainer.
- Apply the **Instance** plug-in to it.





Our example tree has about 3.2 million vertices and our scene requires 17 drawcalls:

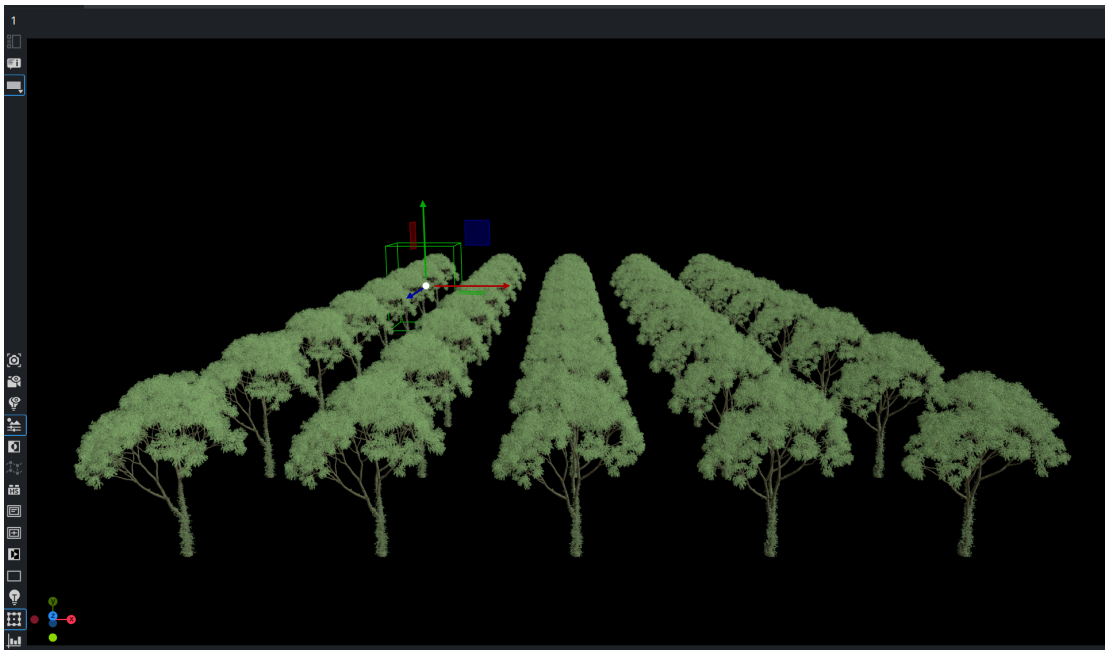
- Create a target container.
- Use Cloner or a similar plug-in to create a certain amount of sub containers. In our example we want to have 40 clones, divided into eight rows and five columns.

> **Note:** Remember to put one container as a source container underneath the target if you are using Cloner.

- Apply the Instancing plug-in on it.
- In the **Producer** plug-in, defines our target container as producer container.



We now have 40 trees.



Giving some rotation and scaling differences as well as some offsets to the XZ position would look make it more realistic.

Note: Not all features are available for instanced objects (for example global illumination).

See Also

- Particle System

5.18 Particle System



The Viz Engine Render Pipeline supports GPU-accelerated particle systems.

A particle system is used to simulate and render complex visual effects and natural phenomena (for example, fire, smoke, rain, snow, fog, dust, and explosions).

Note: The Classic Render Pipeline particle modifiers like Vortex or Turbulence are not supported.

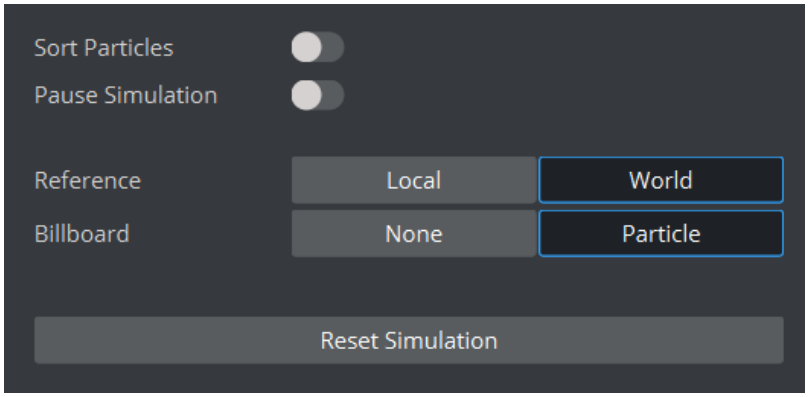
This page contains information on the following topics:

- Creating a Particle System
  - General Properties and Control
  - Common Parameters
- Particles and Instancing
  - To Setup Instancing
- Properties
  - Emission
  - Force
  - Texture
  - Size
  - Color
  - Alpha

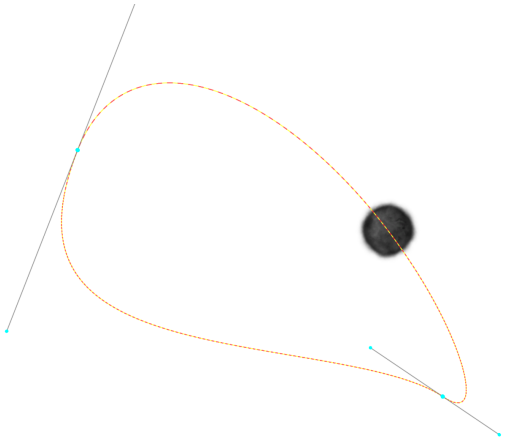
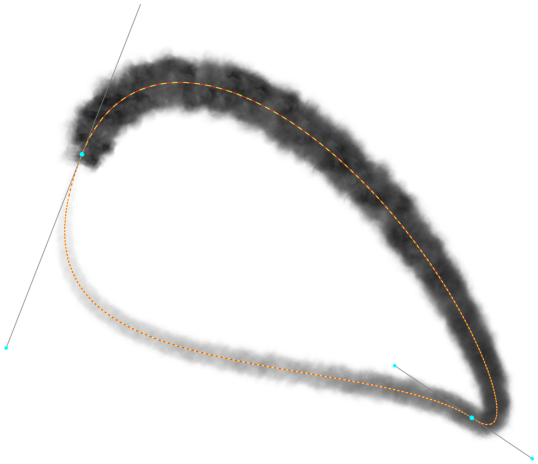
5.18.1 Creating a Particle System

- Drag a container to the scene tree (for instance a group).
- Search for the *Particle System* plug-in and drag it over the container.
- The Particle System plug-in starts emitting particles using the default values.

General Properties and Control



The reference defines if your particle system is part of your scene (World) or if it is running locally. Let's use an animated trail as an example:

Local	World
	

The sorting of particles should always be turned on when alpha images are in use to guarantee a correct blending.

Common Parameters

Properties	Description
Sort Particles	Sorts the particles from back to front to produce correct alpha blending.
Pause Simulation	Pauses/resumes the simulation.
Reference	Changes the way the container transformation is applied to the particle system. • **Local:** Applies the transformation to all particles. • **World:** Applies the transformation to the emitter.
Billboard	Controls the rotation of each particle in relation to the camera. • **None:** No change is done to each particle rotation. • **Particle:** Rotates each particle to face the camera.
Reset Simulation	Resets the particles and restarts the simulation.

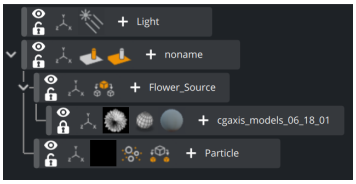
Copyright © 2026 Vizrt

5.18.2 Particles and Instancing

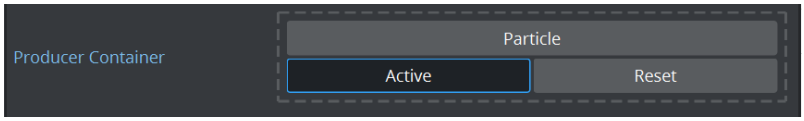
Introduced in Viz Engine 5.4 the Particle System is capable of emitting any instanced geometry. This allows for a very effective rendering as the draw calls to the GPU are reduced to an absolute minimum.

To Setup Instancing

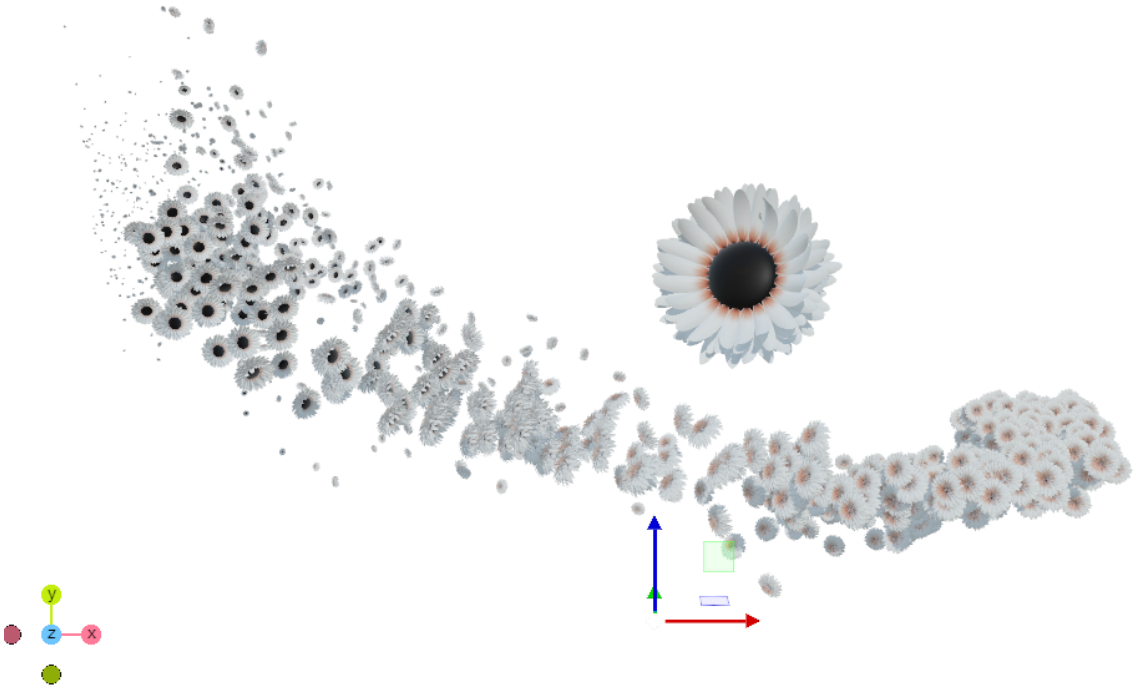
To emit particles, you need to setup a reference geometry using the instancing plugins.



- Create a geometry and put the **Instancing** plug-in on it.
- Create another container, add the **Particle** plug-in on it and the **Instancing Producer** on it.
- Make sure the reference to the **Producer Container** is correct.



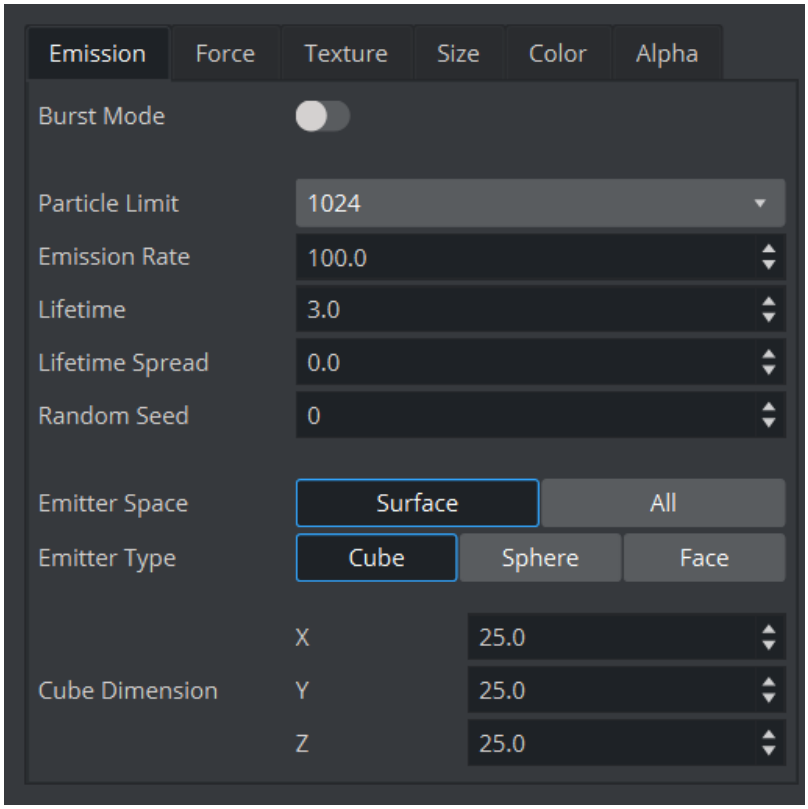
- In the Particle plug-in Tab Emission, set the **Rendering mode** to *Instancing*.



Copyright © 2026 Vizrt

5.18.3 Properties

Emission



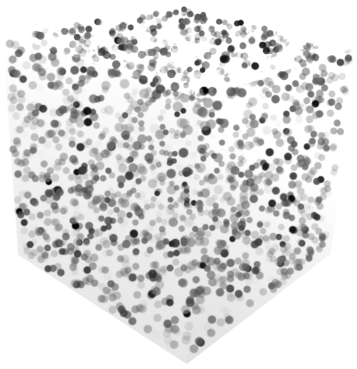
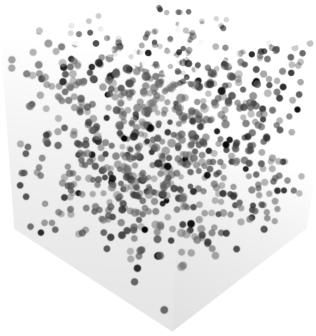
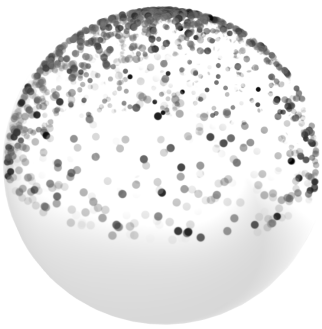
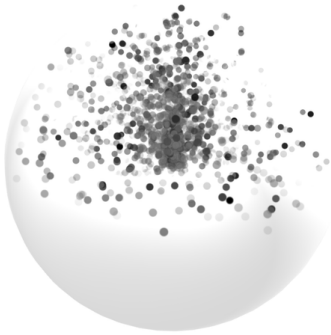
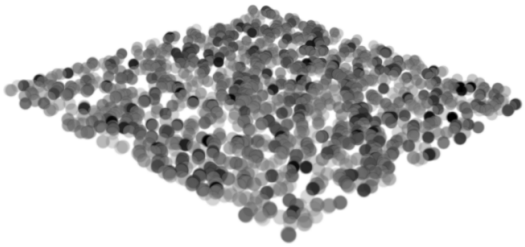
The way particles are created is based on their *Emission Rate* (how many particles are created per second) and their *Lifetime* (how long they are active, in seconds). This is only limited by the *Particle Limit* setting. For example, if you have an *Emission Rate* of 1000 particles and the *Lifetime* is three seconds, you'll need a minimum *Particle Limit* of 3000 to guarantee a constant emission of particles. If the combination of *Emission Rate* and *Lifetime* produces more particles than the limit set by this property, the excess particles won't be simulated/rendered.

In Burst Mode, the particle system generates as many particles as needed and then emits them at once each second.

The location where particles are born and emitted is defined by the *Emitter Type* and the *Emitter Space*.

Type / Space	Surface only	All (whole volume)

Copyright © 2026 Vizrt

Cube	 Particles are created within a cube, aligned to the faces	 Particles are created within a cube and aligned inside the volume
Sphere	 Particles are generated inside a sphere (polar range 90), aligned to the faces	 Particles are generated inside a sphere (polar range 90), aligned to the faces
Face	 Particles are generated based on a plane, the space is not relevant in this mode.	

Copyright © 2026 Vizrt

Emission Parameters

Properties	Description
Burst Mode	Emits particles in the same instant (instead of being continuously emitted) when toggled on.
Particle Limit	Limits the number of particles being simulated.
Emission Rate	Controls the amount of particles emitted per second. Fractional values can be used.
Lifetime	Determines the time (in seconds) for the particles to be simulated.
Lifetime Spread	Controls the randomness for the life time of each particle. A random value between *-Lifetime spread* and *Lifetime spread*, is added to the life time of each particle.
Random Seed	Sets the seed for the random number generator used by the random properties.
Emitter Space	Controls from where in the emitter the particles are emitted. • **Surface:** Emits the particles from the surface of the emitter volume. • **All:** Emits the particles from all the emitter volume.

Emitter Type	Selects the volume from which the particles are emitted. Dimensions for each type available can be changed using:
	• *Sphere* emitter
	  • **Radius** is the sphere radius.
	  • **Azimuth range** is the opening angle on the XZ plane, in degrees.
	  • **Polar range** is the opening angle from the Y axis, in degrees.
	• *Cube* emitter
	  • **Cube Dimension X, Y, Z** is the cube size on each axis.
	• *Face* emitter
	  • **Face Dimension X, Z** is the face (plane) size on each axis.

Copyright © 2026 Vizrt

Force



How particles move is defined in the Forces tab. Depending on their emitter type, particles behave differently:

- Particles born in a cube move in six different directions.
- Sphere particles spread from a single point or sphere surface.
- Particles on a plane travel in only one direction.

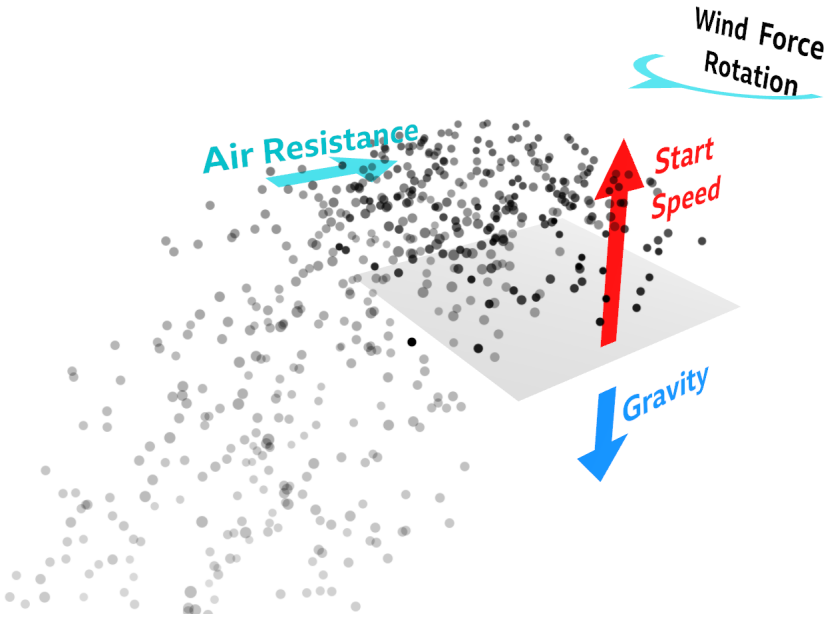
Start speed and the spread modifier (**+** / **-**) defines the initial speed. This force is controlled by *Gravity*, *Air Resistance* and the *Wind Force*.

Example: Particles Emitted from a Face

With their initial starting speed, particles are emitted from the face pointing upwards.

- *Wind Force* and the direction in x and z move the particles.
- *Air Resistance* slows down the speed.
- *Gravity* forces the particles to drop to the ground.

Copyright © 2026 Vizrt



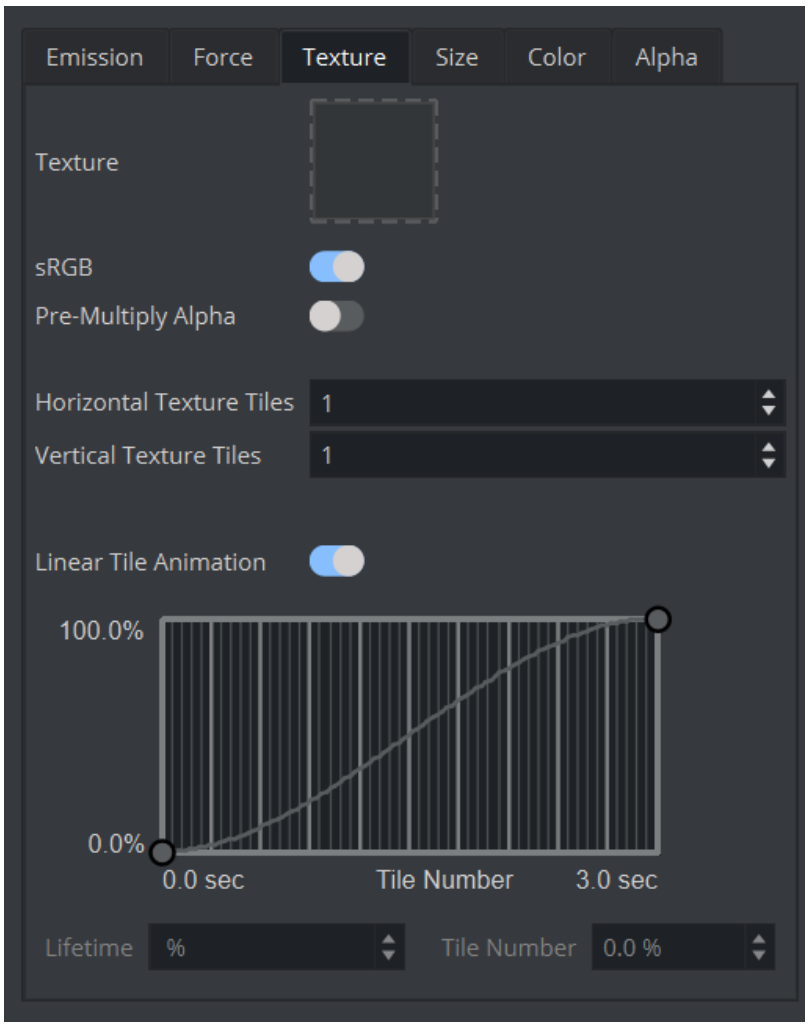
Force Parameters

Properties	Description
Start Speed	Sets the initial speed for each particle.
Start Speed Spread	Controls the randomness for the start speed of each particle. A random value between *-Start speed spread* and *Start speed spread*, is added to the start speed of each particle.
Start Rotation X	Sets the initial speed rotation around the X axis.
Start Rotation Z	Sets the initial speed rotation around the Z axis.
Gravity	Controls the acceleration for the particles in the Y direction.
Air Resistance	Sets the scale of a drag force proportional to the particle speed.
Wind Force	Sets the magnitude of a force applied to each particle.
Wind Rotation X, Wind Rotation Z	Determines the direction of the wind force on the X and Z axis.
Turbulence X,Y,Z	Controls the scale of a random perturbation, of the speed of each particle on each axis.

Copyright © 2026 Vizrt

Properties	Description
Rotation Speed X,Y,Z	Sets particle angular velocity around the given axis.
Rotation Speed Spread X,Y,Z	Controls the randomness for the particle rotation speed of each particle. A random value between *-Particle rotation speed spread* and *+Particle rotation speed spread* is added to the particle rotation speed.

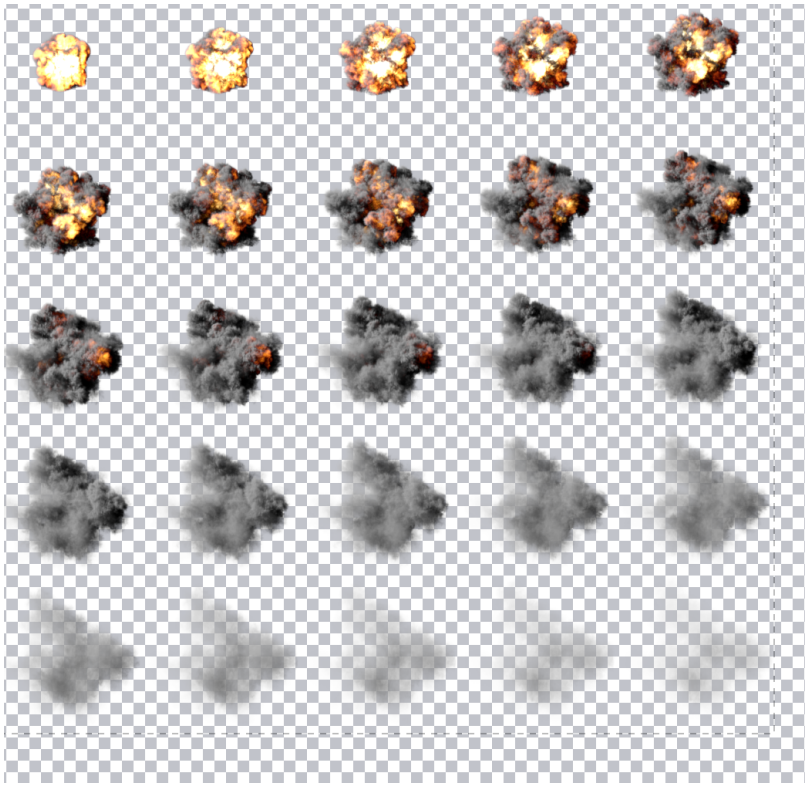
Texture



Particles require a texture to look good. These textures can be static RGBA images or tile sequences. (for example, explosions or smoke). Sequences should be in RGBA format and tiled. The sample below shows a 5x5 tile explosion

Copyright © 2026 Vizrt

animation. The number of particles on such an animation can be kept quite low (for example, about 20 particles), just a bit of turbulence and x rotation is needed.





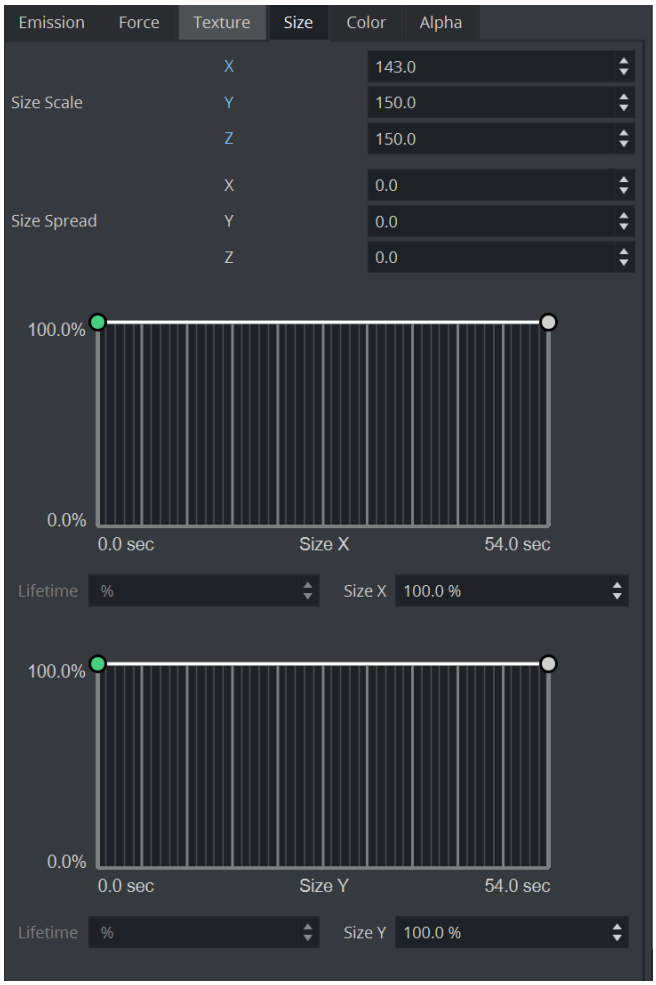
Copyright © 2026 Vizrt

Texture Parameters

Properties	Description
Texture	Selects a texture to apply to each particle.
sRGB	Treats the texture as an sRGB encoded texture (checked by default) when checked.
Pre-Multiply Alpha	Multiplies the final color of the rendered particles with the alpha from the texture.
Horizontal Texture Tiles	Sets the number of horizontal tiles in the texture. Used to control the sprite sheet animation.
Vertical Texture Tiles	Sets the number of vertical tiles in the texture. Used to control the sprite sheet animation.
Linear Tile Animation	Calculates the texture tile selected for rendering by linearly interpolating within the horizontal and vertical tiles using the current life time of the particle when checked. Uncheck to control the tile selection using an animation curve.
Tile Number	Controls the tile number during its lifetime. Horizontal axis is the lifetime in seconds, vertical axis controls the tile number based on the percentage of total tiles.

Copyright © 2026 Vizrt

Size



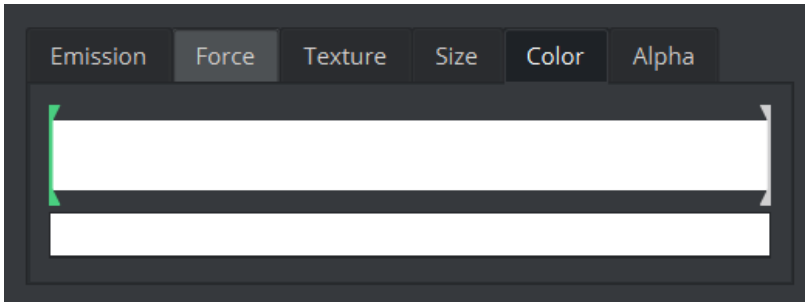
Size Parameters

Properties	Description
Size Scale X, Y, Z	Scales for the size of the particle, where a particle is represented by a rectangle.
Size Spread X, Y, Z	Controls the randomness for the size scale of each particle. A random value between *-Size Spread X,Y* and *Size Spread X,Y* is added to the size scale.

Copyright © 2026 Vizrt

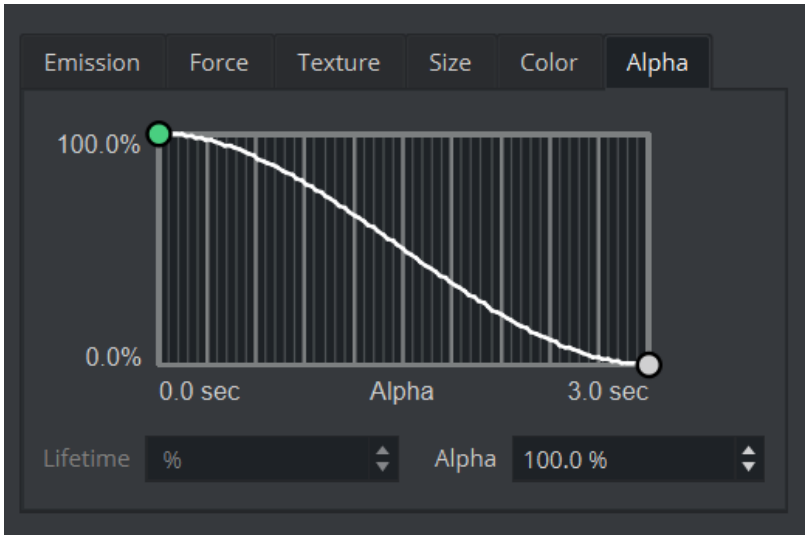
Properties	Description
Size X, Y, Z	Controls the particle size during its lifetime. Horizontal axis is the lifetime in seconds, vertical axis controls the particle size based on a percentage of the size scale.

Color



Use the color ramp box to control the particle color during its lifetime. The leftmost part of the ramp is the initial color and the rightmost is the final color. Multiple color handles can be created and moved. Change the color of each color handle in the smaller box below the color ramp.

Alpha



Copyright © 2026 Vizrt

Alpha Parameters

Properties	Description
Alpha	Controls the particle alpha during its lifetime. Horizontal axis is the lifetime in seconds, vertical axis controls the alpha value

See Also

- Instancing

Copyright © 2026 Vizrt

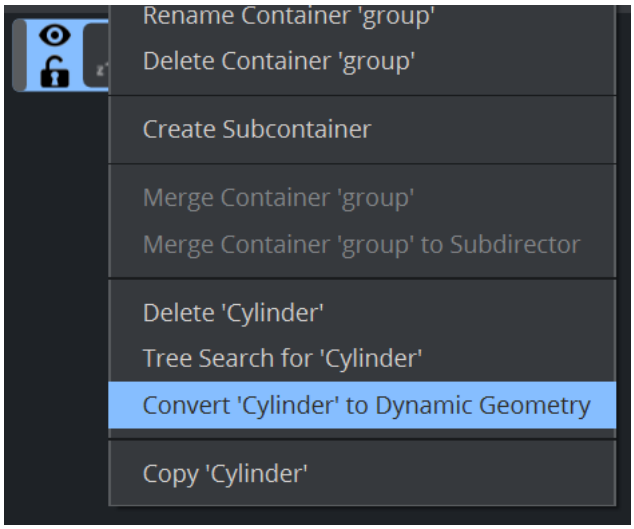
5.19 Working with Dynamic Geometries

Geometries in the Viz Engine Render Pipeline can be either static (for example, from imported 3D models) or dynamic. Dynamic geometries offer a very flexible and efficient way in handling these data.

While static geometries can not be modified after the data has been uploaded to the GPU, dynamic geometries can be transformed by one or multiple *Node Modifiers*. Almost every Classic Render Pipeline geometry plug-in can be converted into a Dynamic Geometry.

5.19.1 Convert a Geometry into a Dynamic Geometry

Right click on a container holding a basic geometry and choose **Convert *n* to Dynamic Geometry**.





Once the geometry has been converted, the Icon of the geometry changes to indicate a Dynamic Geometry. Clicking on this Geometry Icon shows you a list of a Generator and one or more Modifiers.

A generator is an instance of the original geometry plug-in. A special type of generator is a Blendshape model. Operations on Dynamic Geometries should usually be performed on the GPU as this performs much faster than on the CPU. The option **Prefer GPU** calculates only the required ones on the CPU and proceeds to perform all other calculations on the faster GPU. If enabled, it searches for the earliest possible node to upload its data (based on the requirements of all nodes). If an invalid node combination is found, later nodes that do not meet the requirements are skipped.

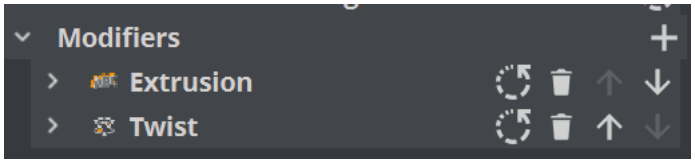
You can replace the existing Generator by clicking on the icon and selecting a new geometry, or by dragging a new geometry on to the Dynamic Geometry icon on the scene tree.

5.19.2 Node Modifiers in Detail

A node modifier takes the node data of the generator or the previous node and transforms it into a new one. Therefore, the order of the applied node modifiers matters. For example, if you first twist a geometry and extrude it afterwards, the result looks different.

Copyright © 2026 Vizrt

Add a new modifier by clicking on the + icon inside the modifier accordion and the list of available modifiers are shown.



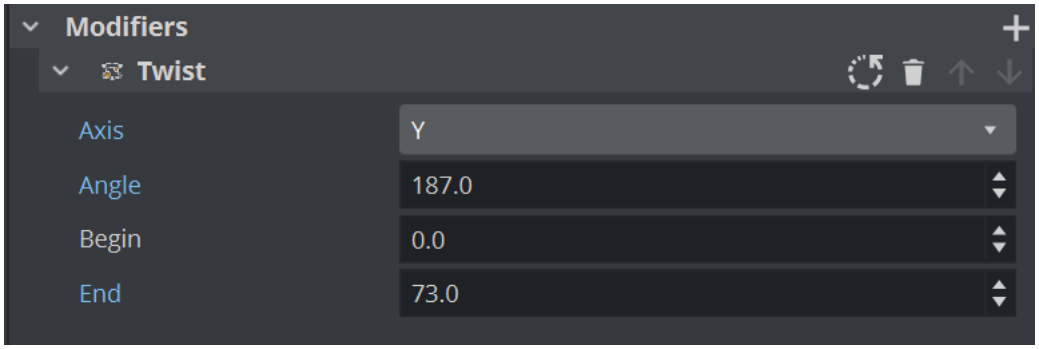
- **:** Replace a modifier.
- **:** Remove a modifier by clicking on the trashcan icon.
- **:** Change the sequence by clicking on the arrow icons, as the order matters.

Twist Node Modifier



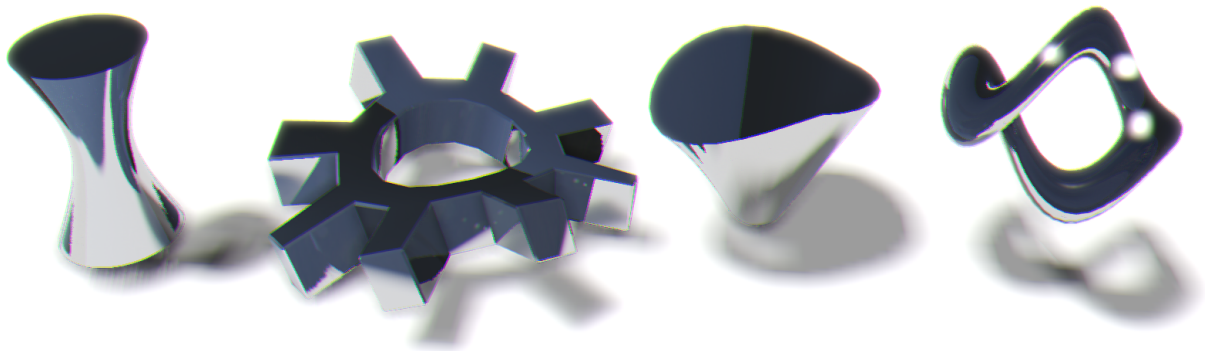
The Twist modifier allows to add a twist effect to any geometry. As more vertices are available on the geometry, the better the result is.

You should always try to increase the number of vertices on the base geometry by increasing the tessellation value.



The Twist modifier allows to rotate on **X,Y or Z-Axis**.

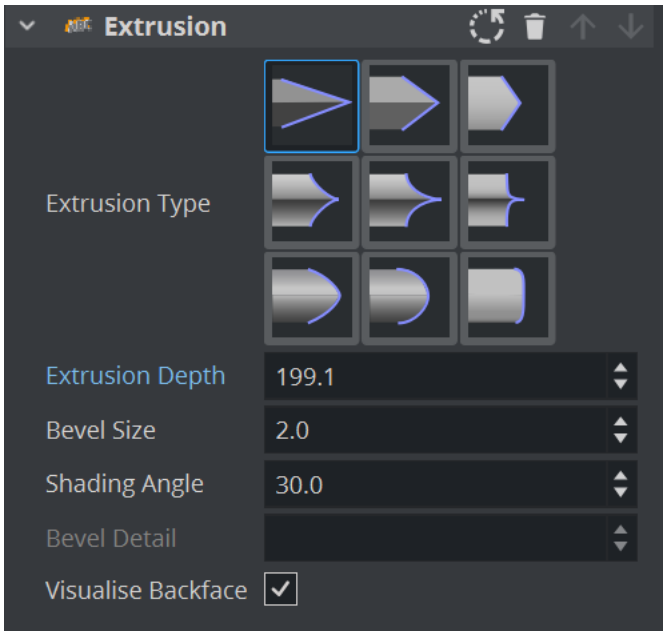
The **Angle** is applied between the **Begin** and **End** property.



Extrusion Modifier



The Extrusion modifier can be used to turn any 2D object (like a rectangle or circle) into a 3D object.



Modify the length of your extrusion using the **Extrusion Depth** parameter.

Start at the end of the extrusion effect can be changed by using the **Bevel Size** option. Depending which Extrusion Type you have chosen, the effect of the bevel looks different.

Some Extrusion types also allow to modify the **Detail** of the bevel effect or the **Shading Angle.**

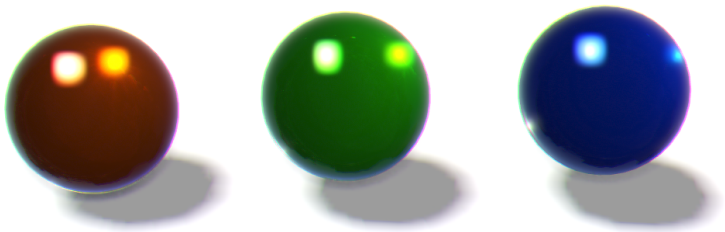
Visualize Backface turns off the additional vertices; however, as all these calculations are performed on the GPU, this might only be needed in some rare use cases.

Information: Text can not be converted into a Dynamic Geometry, but the Built-In Extrusion on the text geometry is based on the same algorithms.

Color Modifier



The color modifier is a very basic one, adding a color to the geometry.



Copyright © 2026 Vizrt

Generate Normals Modifier



The Generate Normals modifer is usually only applied to recalculate normals on a geometry if they are missing.

The normal defines how a face is facing the camera.

UV Modifier



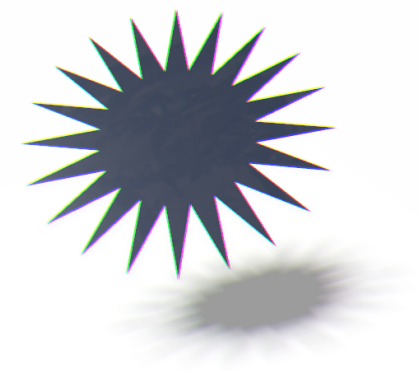
The UV modifier creates a UV map for texturing.
Choose the mode that fits bests, e.g. Box and modify Offset, Scaling, Rotation and Pivot.

5.19.3 Combining Multiple Modifiers

The power of Dynamic geometries and the modifiers is that all of them can be combined. Lets take a simple example:

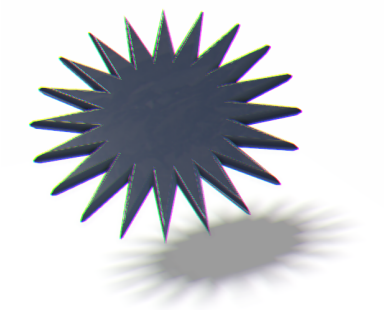
We start with a basic star geometry with about 20 corners.

- Convert it to a Dynamic Geometry by right clicking on the geometry icon in the scene tree and choose **Convert Star to Dynamic Geometry**.

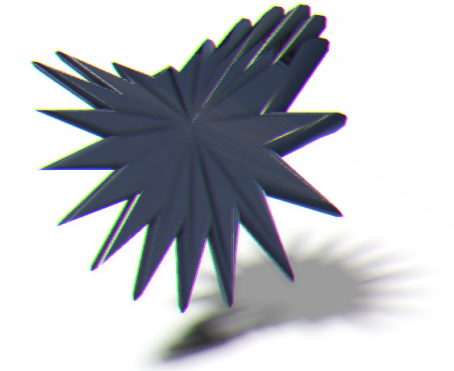


- Now lets add the **Extrusion Modifier** to it.

Copyright © 2026 Vizrt



- Add a **Twist Node Modifier** to it. This twists our geometry along the X -Axis.



- Add a second **Twist Node Modifier** to it and set the axis to the Z-axis. Increase the number of Corners to about 100.



5.19.4 Use Blendshapes



Blendshapes (in conjunction with techniques such as “bone and skin”) are used to animate an imported static geometry (for example, from an *.FBX* file).

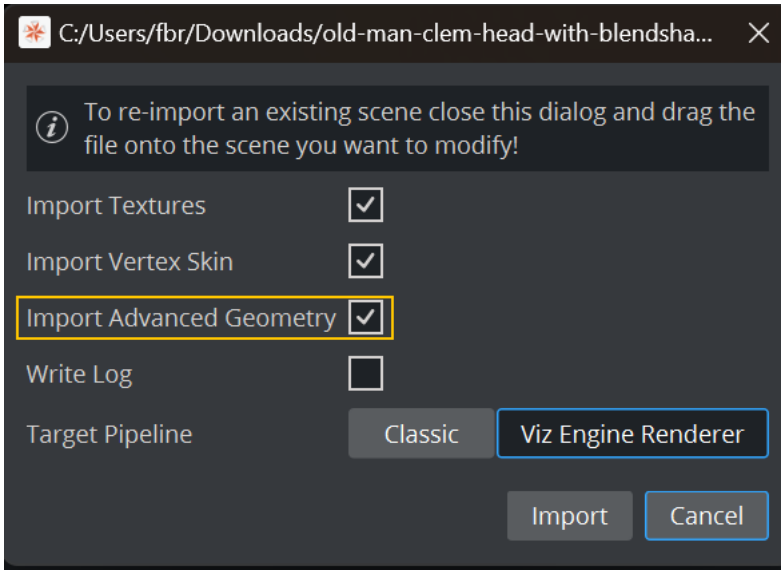
Multiple versions of a deformed mesh are stored within a Blendshape. These expressions are then interpolated and mixed together to create a new impression and is often used for character facial expression, simulate the bending of muscles or other deformations of a 3D model.

Blendshapes are very flexible and efficient as they provide very realistic animations and lightweight data to process. They offer precise control over the character's expression and movement.

Copyright © 2026 Vizrt

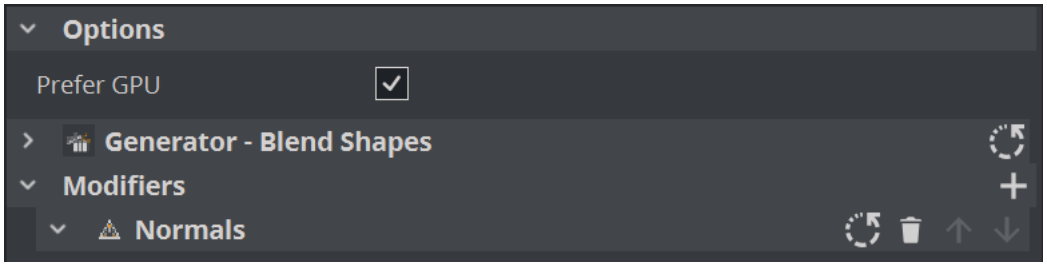
Blendshapes are usually created in modeler tools like Maya or Houdini, Blender (Shape Keys), 3DS Max (Morph Targets) or Cinema 4d (PoseMorph) and then imported in Viz Engine for Realtime animation. Apples ARKit also supports Blendshapes for their real time avatars.

To import an *.FBX* file with blendshapes, make sure to check the option *Import Advanced Geometry* in the import dialog box:



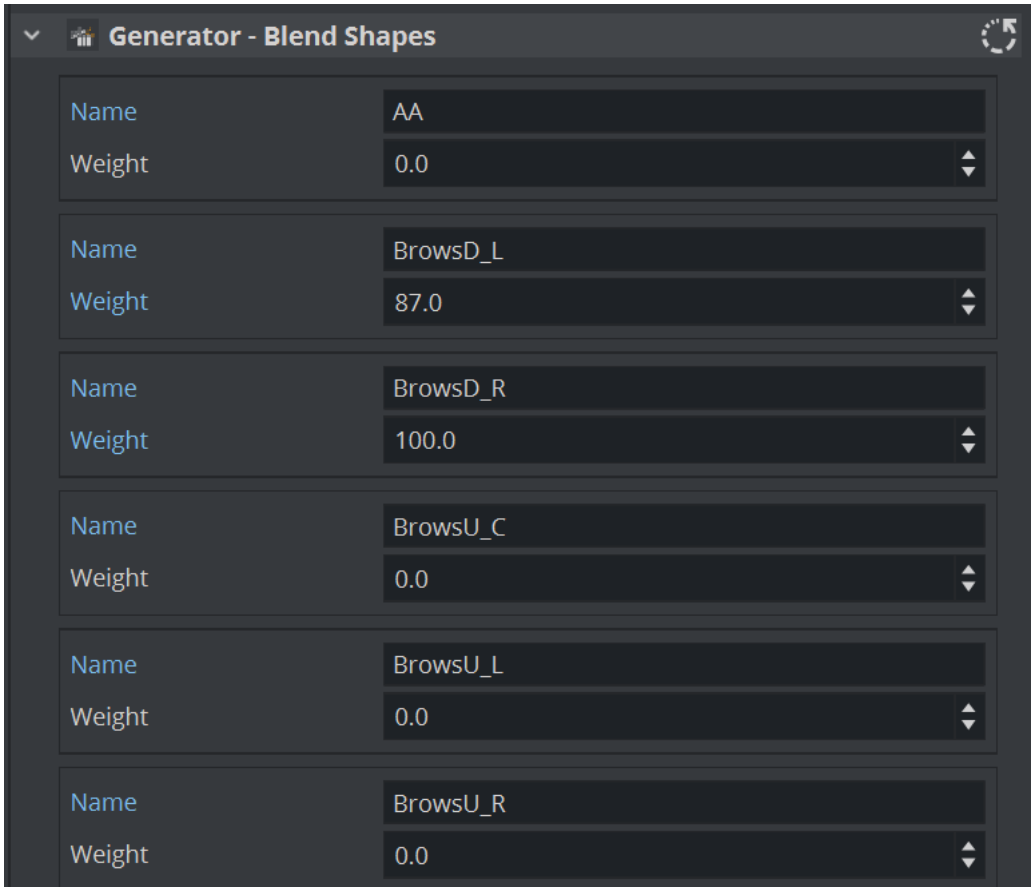
Information: Only *.FBX* files can be imported as Blendshapes.

Once the model is imported, it is shown as Dynamic Geometry and the Blend Shapes Generator as well as the Normals Modifier is applied. The *Prefer GPU* option should always be kept unless there is a reason to move the calculation to the other (slower) CPU.



The Generator node now displays a list of all Blendmodes within the model as well as a modifier to each expression to control the strength of it:

Copyright © 2026 Vizrt



The names of the expressions varies from model to model, however Apples AR Kit defines 52 sets of expressions (for faces).

The weight can be set from a value between 0 and 100 and mixes the shapes into a new expression.



Credits: "Old Man Clem Head with Blendshapes" by skullvez, licensed under Creative Commons Attribution.

See Also

- ARKit 52 Facial Blendshapes: The Ultimate Guide with Anatomy References for Artists

Copyright © 2026 Vizrt

5.20 Stylesheets

- Introduction
- Enabling and Using Stylesheets
    - Defining Styles
        - Adding Styles
        - Naming Conventions and Syntax
        - Defining Style Values
            - Built-in Style Properties
            - Plugin Property Support
        - Style Authoring
            - To add a specific property to a Style
    - Selecting Styles
        - Using the StyleSelect Plug-in
    - Applying Styles
        - When Styles Are Applied
        - Performance Consideration: Explicit Enablement
- Subscene Workflow
    - Style Application in Subscenes
    - Reusing Prototypes with Different Styles
- Scripting
    - Generic Style Functions
    - prop Constants
    - Convenience Functions

5.20.1 Introduction

Stylesheets introduce a powerful, CSS-like styling mechanism to Viz Engine. It enables designers and developers to define and apply styles systematically, particularly when working with a large number of similar objects. It is ideal for scenarios where many objects of the same type require unified visual treatment, where stylesheets allow for easy and maintainable bulk styling.

The use of Stylesheets in Viz Engine is entirely *optional*. Users can enable stylesheet functionality based on the demands of their specific workflow or project. This allows teams to adopt it incrementally or only where it offers clear advantages.

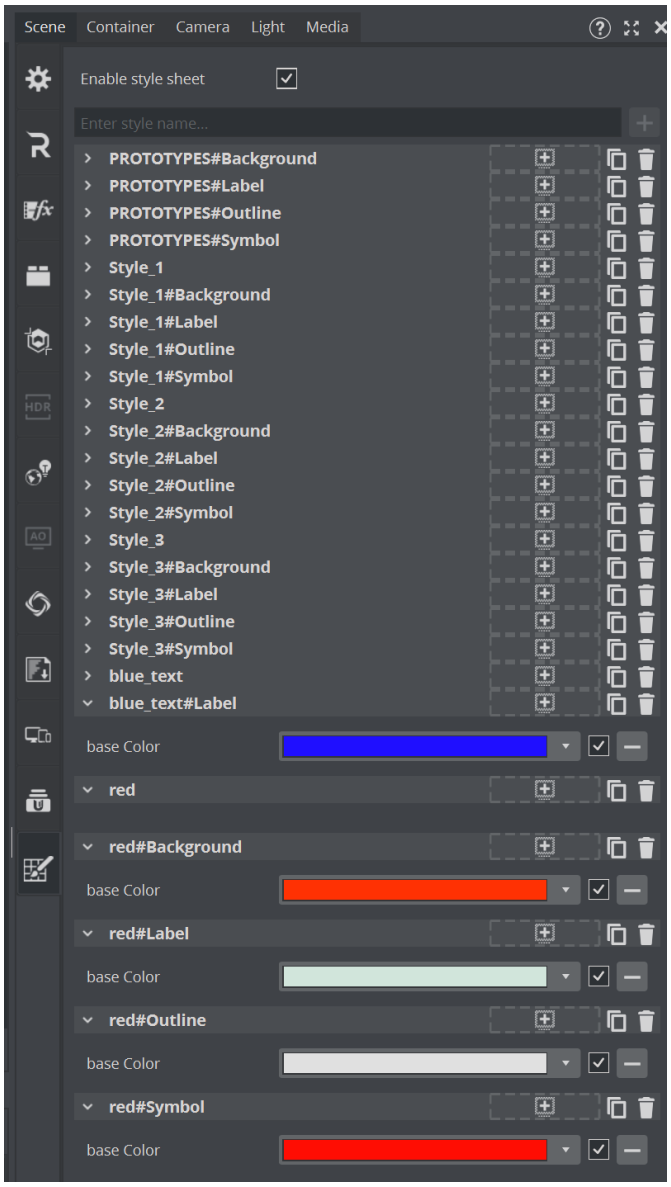
Stylesheet-based styling in Viz Engine is structured around *three distinct phases*:

1. **Style Definition:** This is where visual attributes are established. You define how things *should look*—including colors, shapes, sizes, and other visual rules that describe the design language of your scene.
2. **Style Selection:** Here, logic determines which defined style is appropriate for each object. Styles are chosen dynamically based on **StyleSelect** plug-in, container name, plug-in type and the current format.
3. **Style Application:** The selected style is applied to the object, ensuring consistent visual treatment across similar elements throughout the scene.

Copyright © 2026 Vizrt

5.20.2 Enabling and Using Stylesheets

Stylesheet functionality is disabled by default and must be explicitly enabled in the **Scene Settings** under the **Stylesheet** section:



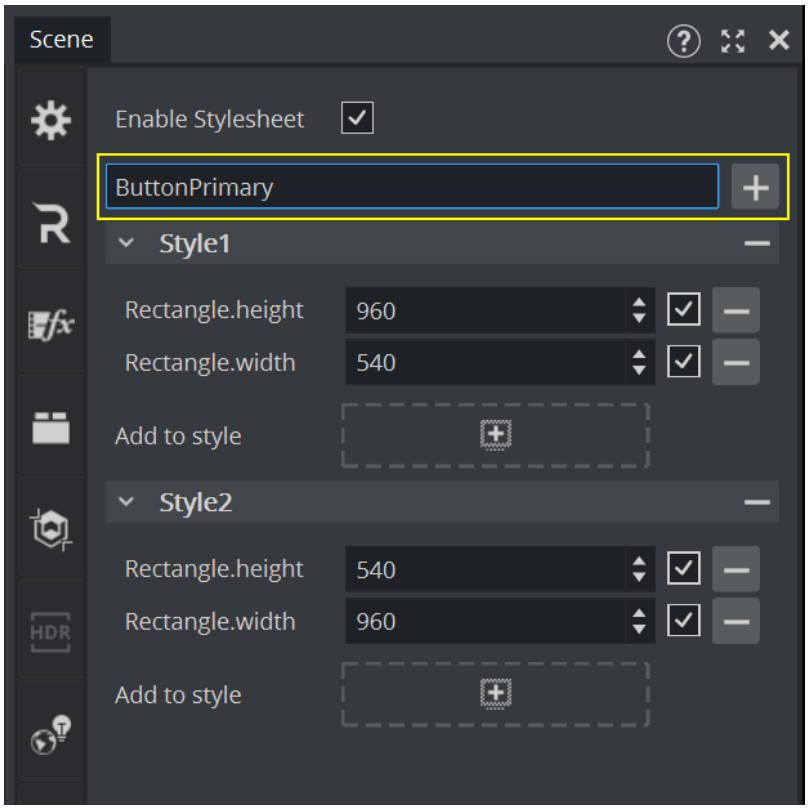
Defining Styles

Adding Styles

Styles can be added directly within the **Stylesheet** section of the **Scene Settings** in Viz Artist. In the input field, type the name of the style you want to create. This can include format prefixes, container name suffixes, or plug-in type

Copyright © 2026 Vizrt

suffixes (for example, `ButtonPrimary` or `Portrait:ButtonPrimary` or `Portrait:ButtonPrimary#MainButton`).



Naming Conventions and Syntax

To ensure clarity and consistency across your stylesheet design, style names need to follow these naming conventions:

- **Alphanumeric Identifiers:** Style names must begin with an alphanumeric string (for example, `ButtonPrimary`, `TitleStyle1`). This serves as the base name for the selector.
- **Format Prefix:** Prefix the style name with a format identifier when necessary to indicate conditional styling based on content format:
  - Syntax: `format:BaseStyle`
  - Example: `Portrait:ButtonPrimary`
- **Container Name Suffix:** Append the container name using a hash symbol `#` to target container with a specific name:
  - Syntax: `BaseStyle#ContainerName`
  - Example: `ButtonPrimary#MainButton`
- **Plugin Type Suffix:** Append the plug-in type using an at symbol `@` to apply styles based on the object's plug-in type:

Copyright © 2026 Vizrt

- Syntax: `BaseStyle@PluginType`
- Example: `ButtonPrimary@Rectangle`

These naming rules can be combined to create flexible and specific styling conditions. For example, `Portrait:ButtonPrimary#MainButton` is used only when the format is `Portrait`, the container name is `MainButton`, and the style selection has `ButtonPrimary` included.

Defining Style Values

A range of style values that can be applied to control both built-in visual attributes and plugin-specific properties. These values define how objects appear and behave when a style is applied.

Built-in Style Properties

The following predefined `StyleId`s represent core visual attributes supported by the stylesheet system:

StyleId	Description
`baseColor`	Material base color of Phong and PBR Materials
`fontSize`	Font size of Text
`textColor`	Text color of Text
`opacity`	Alpha value of Alpha plug-in
`position`	Position (X, Y, Z) of 3D Transformation
`rotation`	Rotation (X, Y, Z) of 3D Transformation
`scaling`	Scaling (X, Y, Z) of 3D Transformation
`active`	Active state of Container
`lineWidth`	Line Width of Expert

These properties are commonly used to define the visual language of a scene and ensure consistent styling across similar elements.

Plugin Property Support

In addition to built-in properties, stylesheets can also target properties of **Function** and **Geometry Plug-ins**, provided those properties are of the following types:

Copyright © 2026 Vizrt

- **Integer (** `int` **)**
- **Floating-point (** `float` **)**
- **Color**

This allows for dynamic and flexible styling of plugin-driven attributes, such as:

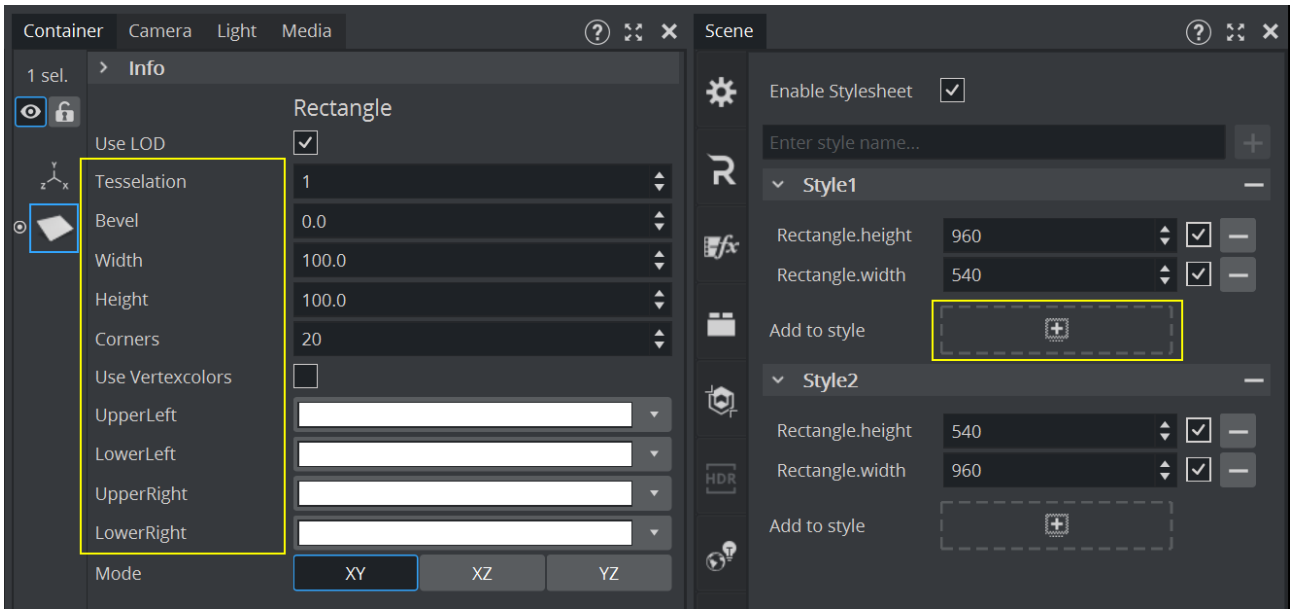
- Adjusting a **radius** or **thickness** value via a float or int.
- Controlling plug-in-specific numeric parameters in a format-aware and maintainable way.

Style Authoring

Viz Artist offers a visual and efficient way to author stylesheets using *drag-and-drop interactions*. This approach allows designers to build styles directly from scene content, reducing manual effort and ensuring consistency. Instead of manually typing out style definitions, you can *drag properties* into the stylesheet editor. Viz Artist automatically captures the corresponding `StyleId` and the current value of each supported property, then inserts them into the selected style block.

To add a specific property to a Style

1. Select a *container plug-in* in the scene.
2. In the **Property Editor**, locate the property you want to include in the style.
3. Drag the *name* of the property (not the value) to the drop area next to **Add to style** in the Stylesheet editor.
4. The property and its current value is added to the selected style.



Selecting Styles

To apply styles defined in a stylesheet, Viz Engine uses the `StyleSelect` plug-in. This plugin is responsible for determining which styles should be applied to a container and its children.

Copyright © 2026 Vizrt

Using the StyleSelect Plug-in



To enable style selection, the StyleSelect plug-in must be added to a container. This container becomes the starting point for style resolution.

- **Inheritance to Child Containers**
  All containers **below** the one with StyleSelect inherit and use the resolved style(s), unless overridden by another StyleSelect further down the hierarchy.
- **Combining Multiple Styles**
  The StyleSelect plug-in allows you to specify *multiple style names*. These styles are combined in the order they are listed.
  - If a property is defined in more than one style, *the last style wins* (the last defined value for a given `StyleId` takes precedence).
- **Multiple Plugins in Hierarchy**
  You can place multiple StyleSelect plug-ins at different levels in the scene tree. When this occurs:
  - The style values from all active StyleSelect plug-ins in the hierarchy are *combined*.
  - Again, if the same property is defined in multiple places, the *lowest (deepest) plug-in in the hierarchy takes precedence*.

Applying Styles

To ensure that styles are applied correctly and consistently, Viz Engine performs an *additional scene tree traversal* during the scene update cycle. This traversal is responsible for resolving and applying styles based on the current `StyleSelect` configuration and the active stylesheet.

When Styles Are Applied

The style application phase occurs:

- *After* the *Stage / Animation Update*
- *Before* the *Transformation Update* and *Plug-in Execution Traverse*.

This timing ensures that:

- Any changes made by animations or stage updates are taken into account before styles are applied.
- Styles can influence transformations and plug-in behavior in a predictable and consistent way.

Performance Consideration: Explicit Enablement

Because style application introduces an *extra traversal* of the scene tree, it is *disabled* by default. This is a deliberate design choice to avoid unnecessary performance overhead in scenes that do not use stylesheets.

This opt-in model ensures that:

Copyright © 2026 Vizrt

- You *don't pay for what you don't use*— scenes without stylesheets incur no additional processing cost.
- Teams can adopt stylesheets incrementally without impacting existing performance-sensitive workflows.

5.20.3 Subscene Workflow

Viz Engine supports the use of stylesheets in scenes that include *subscenes*, enabling flexible reuse of visual components with different styling contexts. However, there are important rules to understand regarding how styles are resolved across scene boundaries.

Style Application in Subscenes

- *Styles are applied within the subscene*, but *only styles defined in the main scene are considered*.
- *Stylesheets defined inside subscenes are ignored* during style resolution.
- This ensures that subscenes behave like reusable prototypes, styled externally by the parent scene.

Reusing Prototypes with Different Styles

This design allows you to:

- Create a *single subscene* (for example, a button, label, or graphic element).
- *Reuse it multiple times* in the main scene.
- Apply *different styles* to each instance by placing a `StyleSelect` plug-in above each subscene reference and assigning different style names.

5.20.4 Scripting

Styles are scriptable and can be changed at runtime via the `Scene` object's Style API. Each style is identified by a **string name**, and individual properties can be accessed via **string keys** or **integer property IDs**. The functions return a `boolean` indicating whether the operation was successful or not.

Generic Style Functions

`Scene.SetStyleInt`	Sets an integer value for a style property.	`(string style, string key, integer value)` `(string style, integer prop, integer value)`
`Scene.SetStyleFloat`	Sets a float (double) value for a style property.	`(string style, string key, double value)` `(string style, integer prop, double value)`

Copyright © 2026 Vizrt

`Scene.SetStyleVec3`	Sets a 3D vector value (x, y, z).	`(string style, string key, double x, double y, double z)` `(string style, integer prop, double x, double y, double z)`
`Scene.SetStyleVertex`	Sets a vertex value.	`(string style, string key, vertex vtx)` `(string style, integer prop, vertex vtx)`
`Scene.SetStyleVec4`	Sets a 4D vector value (x, y, z, w).	`(string style, string key, double x, double y, double z, double w)` `(string style, integer prop, double x, double y, double z, double w)`
`Scene.SetStyleColor`	Sets a color value.	`(string style, string key, color clr)` `(string style, integer prop, color clr)`
`Scene.SetStyleValueEnabled`	Enables or disables a style property.	`(string style, string key, boolean enabled)` `(string style, integer prop, boolean enabled)`
`Scene.RemoveStyleValue`	Removes a style property.	`(string style, string key)` `(string style, integer prop)`

prop Constants

When using the `prop` variant of the functions, the following constants can be used to refer to common style properties:

Copyright © 2026 Vizrt

- `STYLE_PROPERTY_BASECOLOR`
- `STYLE_PROPERTY_FONTSIZE`
- `STYLE_PROPERTY_TEXTCOLOR`
- `STYLE_PROPERTY_OPACITY`
- `STYLE_PROPERTY_POSITION`
- `STYLE_PROPERTY_ROTATION`
- `STYLE_PROPERTY_SCALING`
- `STYLE_PROPERTY_ACTIVE`
- `STYLE_PROPERTY_LINEWIDTH`

These constants provide a more efficient and type-safe way to reference style properties compared to string keys.

Convenience Functions

For frequently used style properties, the `Scene` object provides dedicated convenience functions. These are simpler alternatives to using `SetStyle*` with property constants.

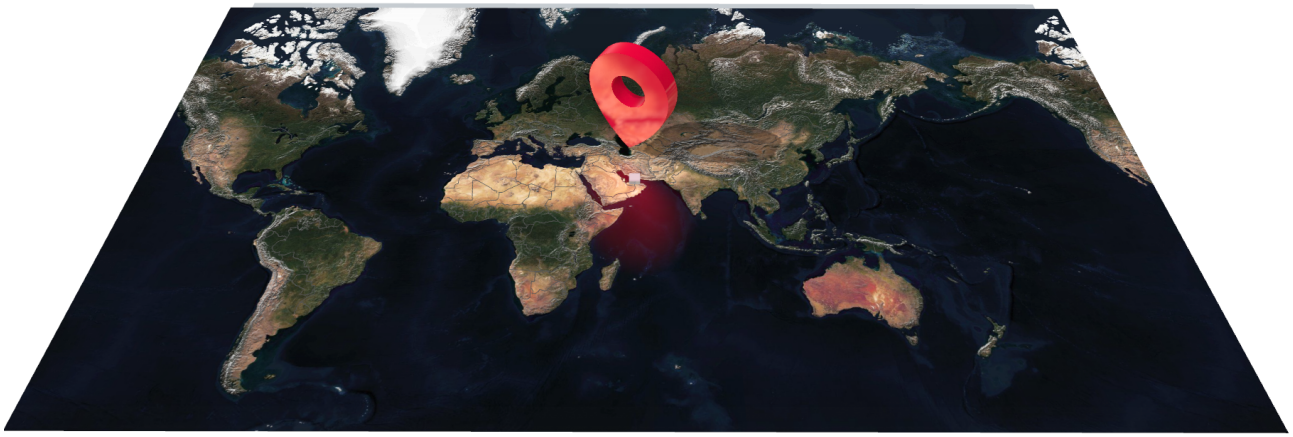
Function	Description
`Scene.SetStyleBaseColor(string style, color clr)`	Sets the base color of the style.
`Scene.SetStyleFontSize(string style, double fontSize)`	Sets the font size.
`Scene.SetStyleTextColor(string style, color clr)`	Sets the text color.
`Scene.SetStyleOpacity(string style, double opacity)`	Sets the opacity (transparency).
`Scene.SetStylePosition(string style, double x, double y, double z)`	Sets the 3D position.
`Scene.SetStyleRotation(string style, double x, double y, double z)`	Sets the 3D rotation.

Copyright © 2026 Vizrt

Function	Description
`Scene.SetStyleScaling(string style, double x, double y, double z)`	Sets the 3D scaling.
`Scene.SetStyleActive(string style, boolean active)`	Activates or deactivates the style.

Copyright © 2026 Vizrt

6 Working with Maps



A map in Viz Engine is not a picture you import. Viz Engine comes with a built-in native rendering of detailed Map Tiles and 3d Geometry onto a canvas, integrates tightly in your existing 3D environment, so it lights, composites and animates alongside the rest of your graphics rather than sitting on top of them. Viz Engine 5.6 comes with a new powerful native Maps rendering solution built in. It allows designers to create easy to use, broadcast quality maps scenes.

- Maps Prerequisites
- Start Using Maps
- Working with GeoJSON
    - Importing GeoJSON
    - Rendering GeoJSON
    - Textures on GeoJSON
    - Adding GeoJSON data
- Working with Animations
    - Using GeoExpression
- Template based workflow
    - Adding Labeldesigns
    - Adding Shapedesigns
    - Adding a Locator
    - Controlling the flight

Anywhere, in one scene - Build the graphic once. A street in Dubai, a coastline in Norway, a stadium in São Paulo — same scene, different coordinates.

Late change? No rebuilding. - The story moves at ten to air. You type a new place name. Nobody rebuilds anything.

It stays where you put it - Pins, labels and shapes belong to a latitude and longitude, not to a spot on the screen. Fly across the world and they hold their ground.

Looks like your channel - A style controls the entire design — imagery, roads, water, labels, type. Change the look of every map you own without opening a single scene.

Copyright © 2026 Vizrt

Rehearse once, hit it every time - Imagery is cached on the machine that plays it out. Fly the move in rehearsal and it goes to air instantly.

Hand it to the operator - A search box instead of a folder of pre-built variants. They type “Reykjavik”; the map is already there.

The Maps Render Engine does a lot more than simply fetching tiles. It also renders borders, streets, rivers, corrects font rendering and many more.
Therefore a map built in the designer will look exactly the same in Viz Engine, in Viz Flowics and other applications. It is built for performance and offers a lot of “behind the scenes” optimizations.

6.1 Note about searching Locations

When searching for certain locations, the results will be the most closest one. However if e.g. a county and a city have the same name, the city will be returned, the county will not show up.
If you are not able to find your location, we recommend to use Nominatim. This lists all entries for a certain name.

Example:
Schwaz is a city in Tyrol, but also the name of one of the counties in Tyrol. Searching for “Schwaz” will only return the city location, not the county.
Nominatim returns all locations found and also lists “Bezirk Schwaz” as the correct search term to get the County and not the City.

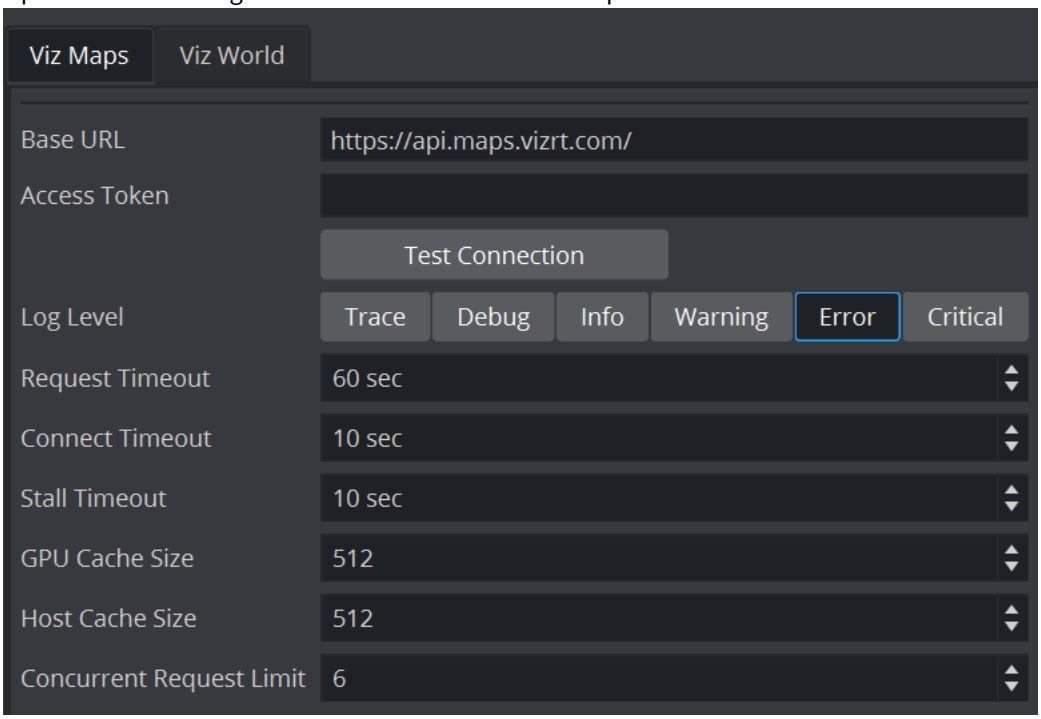
Copyright © 2026 Vizrt

6.2 Maps Prerequisites

6.2.1 Configuration

To start working with Maps within Viz Artist, an account on Vizrt Maps Service is required. If you do not already have an account, please contact Vizrt Support.
They will provide you with an API endpoint and a unique Access token.

- Open Viz Artist Configuration and browse to the Viz Maps section:



- Put the data you've been provided by vizrt support into **Base URL** and **Access Token**.
- You most probably won't need to change the other settings.
- Log level simply defines how detailed log messages of the Maps Subsystem will be logged.
- GPU and Host Cache define limits of how much data is being stored on the Graphic Board and on internal RAM. Setting the to 0 disables any limit.
- You can limit the maximum amount of parallel requests to limit the bandwidth request to the maps backend server.

6.2.2 Creating Styles

To start working with maps within Viz Artist, Styles need to be created within the Viz Maps Designer.



See Viz Maps Designer User Guide how to create map styles.

Copyright © 2026 Vizrt

6.2.3 Attribution Settings

Attribution is displayed as part of the customer's licensing agreement. Customers licensed under the Attribution tier are required to display attribution, while Attribution-Free licenses remove this requirement.

Attributions can be set on a scene level.



You can choose your font and style as well as the position to match your corporate identity.

> There are certain limits, for example the font size must not be smaller than 16.
> If the alpha is being changed to be a very low number, a fallback to the default font will happen.

Copyright © 2026 Vizrt

6.3 Start Using Maps

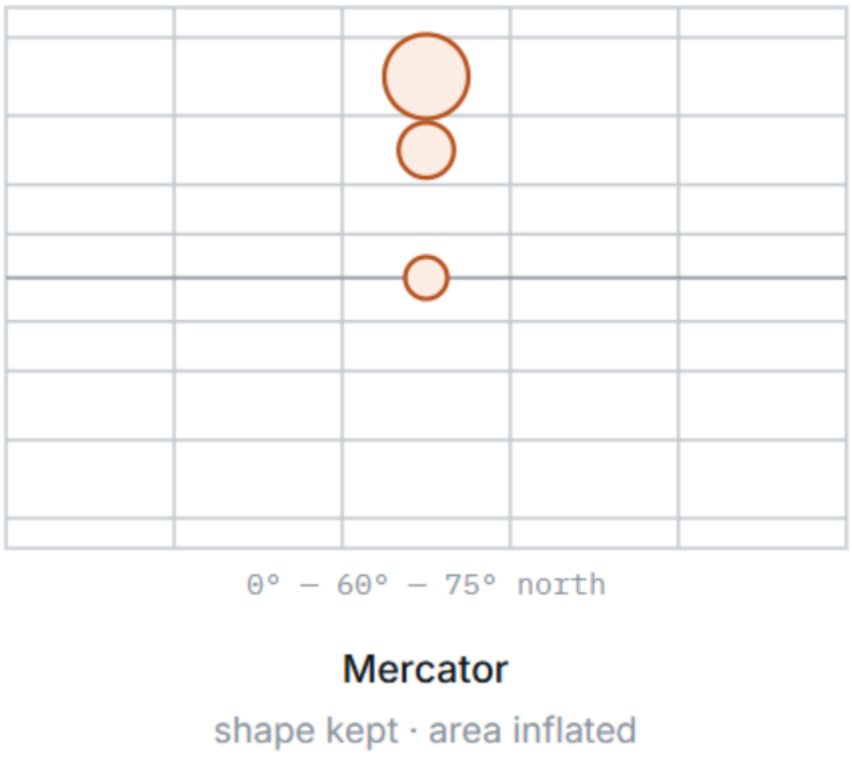
6.3.1 Projections

The Earth is round and your screen is flat. A **projection** is a method that turns the location on Earth, given by latitude and longitude, into positions on a flat map.

Every projection distorts. That is not a flaw of a particular projection; it is a mathematical assurance. Because a sphere cannot be flattened without stretching something. A projection lets you choose *which* property survives: shape, area, distance or direction. You get to keep some and you must give up others.

Viz Engine offers the most common ones and the choice is usually obvious once you know what each retains.

Mercator



The projection every web map uses, and the default here. Meridians are straight, evenly spaced vertical lines; Parallels are straight horizontal lines that spread further apart the closer you get to the poles.

That spreading is deliberate. It is exactly the amount needed to keep **shapes correct**: a bay, a lake or a city block has the right form and angles, and north is straight up everywhere on the map. This is why it feels natural to navigate — you can turn, zoom and read a street layout without anything looking bent.

The price is **area**. Distance from the equator inflates things dramatically: Greenland appears roughly the size of Africa, when it is about a fourteenth of it. The poles themselves are infinitely far away in the mathematics, so the map is cut off a little beyond 85° — the Arctic and Antarctic simply are not there.

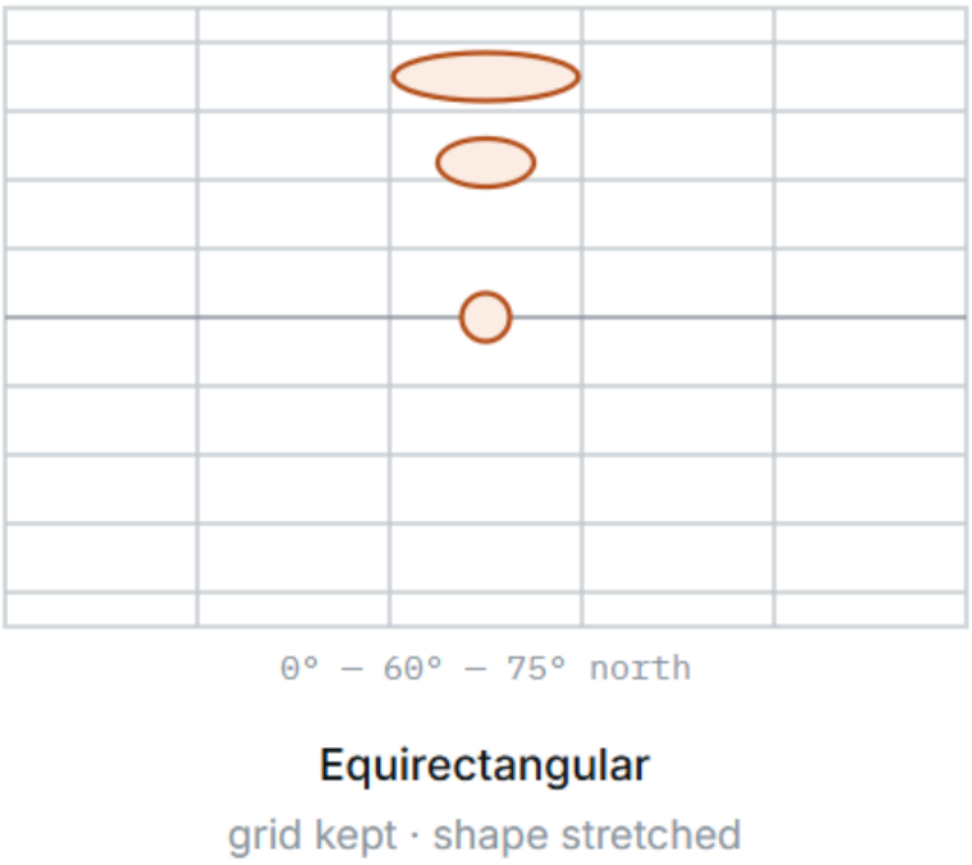
There is a practical reason it dominates. Mercator makes the world a square that divides neatly and repeatedly into four, which is what makes tiling and zooming work at all. That is the subject of the next-but-one section, and it is why imagery on the backend is organized this way.

Copyright © 2026 Vizrt

Choose Mercator for

- Anything the audience navigates: cities, streets, a push into a location.
- Anything where shape carries the meaning — a coastline, a stadium, a border.
- Any map where the viewer's instinct is a phone map, which is most of them.

Equirectangular



The simplest projection is: longitude becomes the horizontal position and latitude becomes the vertical position, with nothing else applied. One degree is the same number of pixels everywhere, so the grid is a plain, even grid of squares.

That evenness is its strength. The whole world fits in a clean two-to-one rectangle, the poles are included rather than cut off, and any data expressed in degrees maps onto it without arithmetic.

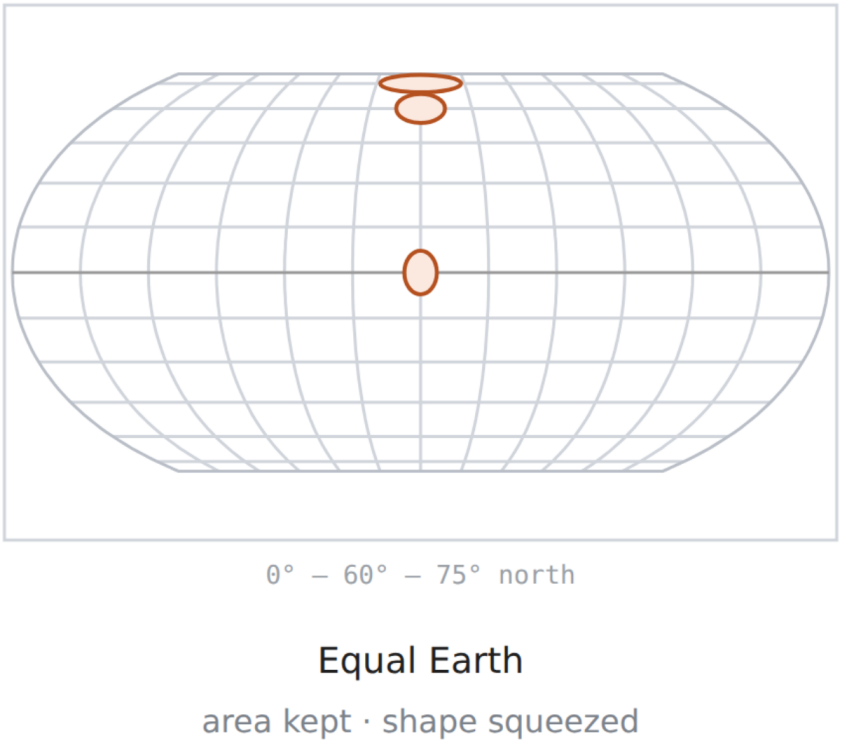
The cost is **shape**. Because the meridians stay parallel instead of converging as they do on the globe, everything is stretched horizontally as you move away from the equator — increasingly so toward the poles, where a small area becomes a wide smear. Near the equator the two projections look almost identical; at high latitude they look nothing alike.

Choose equirectangular for

- Whole-world graphics where you want the entire globe in one clean rectangle.
- Anything that has to include the polar regions.
- Overlays and datasets already expressed as a plain latitude/longitude grid.

Copyright © 2026 Vizrt

Equal Earth



Equal Earth is an equal-area projection. Every region keeps its true size relative to the others, so Greenland appears at its real size next to Africa, not inflated as on Mercator. Parallels are straight horizontal lines, which keeps latitude easy to read. Meridians curve gently toward flat poles, giving the map its rounded outline. To keep area constant, shapes get squeezed toward the poles: features are flattened vertically and stretched horizontally, and this grows with latitude. Near the equator the distortion is small.
Use it for world maps where comparing sizes matters, such as data-driven thematic maps, global overviews and weather or climate graphics.

Choose equal earth for

- **Comparing sizes matters.** Showing country areas, population density or regional coverage without Mercator's inflation near the poles.
- **You're mapping data on a world map.** Choropleths, heat maps, climate or weather overviews, election results or anything shaded by region. Equal area keeps the shading honest.
- **You need a full-world overview.** It shows the whole globe including the poles and looks natural to viewers.
- **You want an alternative to Robinson.** It looks similar, but it is truly equal-area and simple to compute.
- **Latitude should be easy to read.** Parallels stay straight and horizontal, so north/south comparisons remain clear.

If you are unsure, start with Mercator. It is the default, it matches what the imagery is built for, and it is what viewers expect.
Use Equal Earth for world maps where sizes get compared, Mercator for zoomed-in maps where shapes and angles count, and Equirectangular for maps built directly on imagery.

6.3.2 GeoCanvas and GeoCamera

A map is two things placed on one container. They are easy to confuse and they do quite different jobs.

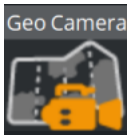
Copyright © 2026 Vizrt



GeoCanvas — the map surface

This is the map itself: the object that works out which imagery the current view needs, requests it, and draws it. It holds the **style**, and the settings that decide how eagerly imagery is loaded and how long the map will wait for it.

Source Mask Layer: The GeoCanvas can automatically create a Source mask for you. This is the same Mask as you would use it with MaskSource/MaskTarget plugins. Use it to cut off anything that goes outside your map.

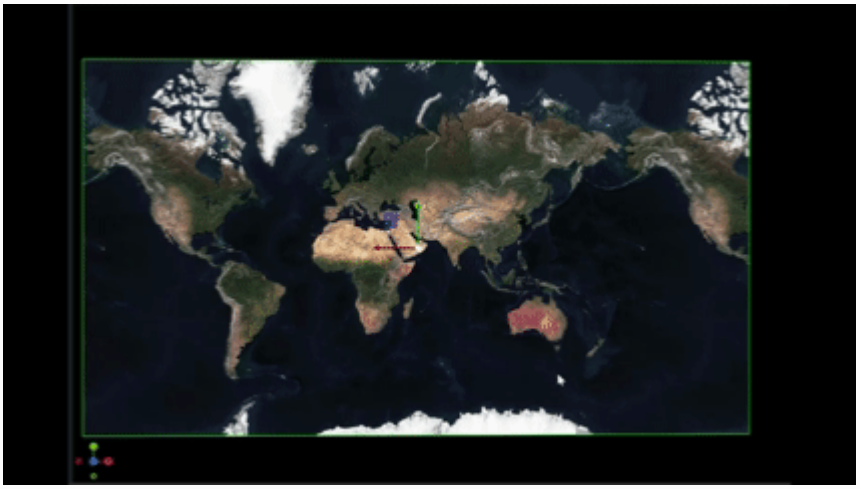


GeoCamera — the view

This decides which part of the world you are looking at and from what angle: latitude, longitude, zoom, bearing, pitch and projection.

The **scene camera** frames the map inside your shot — how big it is, where it sits on screen. The **GeoCamera** flies over the Earth. If you are moving the scene camera to reach another country, you are using the wrong one. If your map is the wrong size in frame, that is the scene camera and not a maps problem at all.

Tip:
press the “M” key (for Map) to directly navigate inside a Map canvas. Use your mouse to change the latitude and longitude, the wheel to change the zoom level.
Notice the camera widget on the lower left will change to a Navigation Arrow instead of the xyz widget.

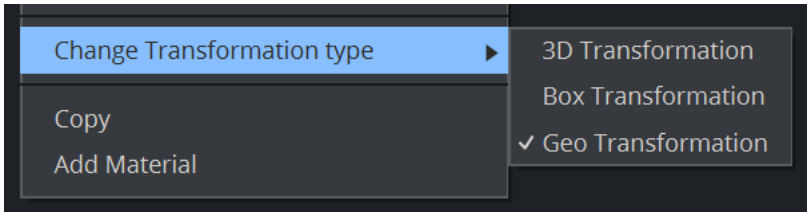


6.3.3 Geo Transformation

To allow for easier navigation and placing objects based on coordinates, there is a new transformation mode introduced, **Geo Transformation**.

Click on a container's transformation icon to switch to **GeoTransformation**:

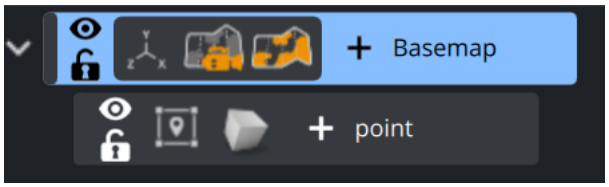
Copyright © 2026 Vizrt



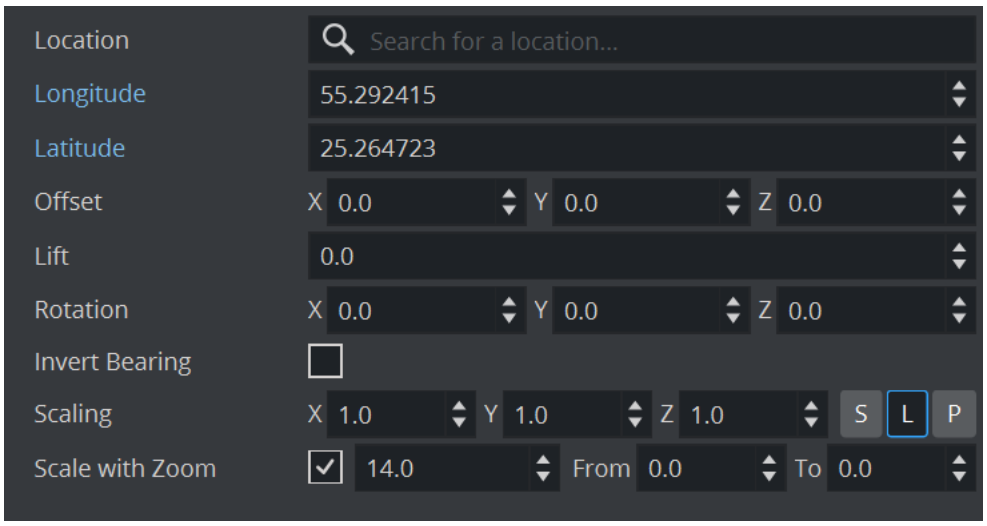
Every container underneath a **GeoCanvas** will use this container as a Basemap. Anything you put on the map - a pin, a label, a shape - will be positioned *geographically* rather than in scene space. A container placed the ordinary way sits at a fixed point in the scene while the map slides underneath it. A container given a geographic position belongs to a place on Earth and travels with the map.

Example:

Let's do a simple example: Create a new container and add the **GeoCanvas** to it, the **GeoCam** is added automatically. Choose a style, the map appears.
Create a sub container and switch to **Geo Transformation**, add a simple geometry to it:



Now clicking on the **Geo Transformation** of the point container will show you several options:



Inside Location, you can search for almost any point of interest, let's choose Dubai. Once selected, Longitude and Latitude will be filled in automatically and our point now sits at the Dubai location in the UAE.

Copyright © 2026 Vizrt



To check if everything works correctly, fly the GeoCamera to another continent. Everything anchored properly leaves the frame along with the ground it belongs to. Anything still sitting in the middle of the screen was never given a geographic position.

There are several more options inside the GeoTransformation:

Row	What it does
Location	A search box. Type a place name and the coordinates below are filled in for you — the quickest way to pin something without looking numbers up.
Longitude	East–west position, −180 to 180. Note that it comes *first* in this panel.
Latitude	North–south position, −90 to 90.
Offset	A nudge in X, Y and Z away from the anchor point once it has been placed. For fine adjustment — moving a label clear of the pin it belongs to, for instance.
Lift	Raises the object off the map surface, so it does not sink into the map or flicker against it.
Rotation	Turns the object on X, Y and Z, on top of whatever the map is doing.
Invert Bearing	Keeps the object upright while the map rotates beneath it. Switch it on for anything carrying text.

Copyright © 2026 Vizrt

Row	What it does
Scaling	The object's size in X, Y and Z.

Lift and Offset are not the same thing

Both shift an object from the position given by its coordinates, but they do so for different reasons.
Lift adds a small amount of height, making the object sit on top of the map rather than inside it.
Offset is a visual adjustment that moves an element slightly to make the overall arrangement clearer

Reach for Lift when something is disappearing into the ground, and Offset when something is in the wrong place relative to its neighbors.

Scale with Zoom

This single row decides how an object behaves as you fly in and out, and it carries three values.

- **The checkbox.** Off, the object stays the same size on screen at every zoom. On, it grows and shrinks with the map, like something drawn on the ground.
- **The zoom value** beside it is the zoom level at which the object appears at exactly the size set under Scaling. Above that level it is larger, below it smaller.
- **From and To** set the limits for scaling. Inside the range the object scales with the map; outside it, the size holds. This avoids markers becoming oversized at high zoom levels or disappearing at low ones.

> A city pin or a label has to stay readable at any zoom — leave **Scale with Zoom** off. Something that represents real ground, like a stadium footprint or a flood area, should grow with the map — switch it on, set the zoom value to the level you designed it at, and give it a From/To range so it stays sensible at both extremes.

Quick Map Guide:

1. **Make a container for the map -** One container at the top of the scene tree. This container *is* the map — everything you place on the map later goes underneath it, which is what lets those things move with it.
2. **Add the GeoCanvas and the GeoCamera -** Both go on the container. Nothing appears yet: the map has no style, so there is nothing for it to draw. That blank state is expected.
3. **Choose a style -** Set the style to one your backend publishes — *Satellite* is a good first choice because it is unmistakable when it works. Imagery should begin arriving.
4. **Frame it with the scene camera -** The map renders in the scene like everything else, so the scene camera decides how much of the shot it fills. Adjust it until the map sits in frame the way you want. You should not need to touch this again.
5. **Fly it with the GeoCamera -** Set latitude, longitude and zoom for your first location — try `40.7128`, `-74.0060` at zoom 12 for New York. Get the position right before adding bearing and pitch, so you always know which control caused what.

Copyright © 2026 Vizrt

Animating GeoTransformation



GeoTransformation parameters can be animated.
If you need to quickly animate a flight from Vienna to Brazil to New York, simply type in the locations, hit the **Set Key frame** and you're done.

6.3.4 Caching and Optimizations

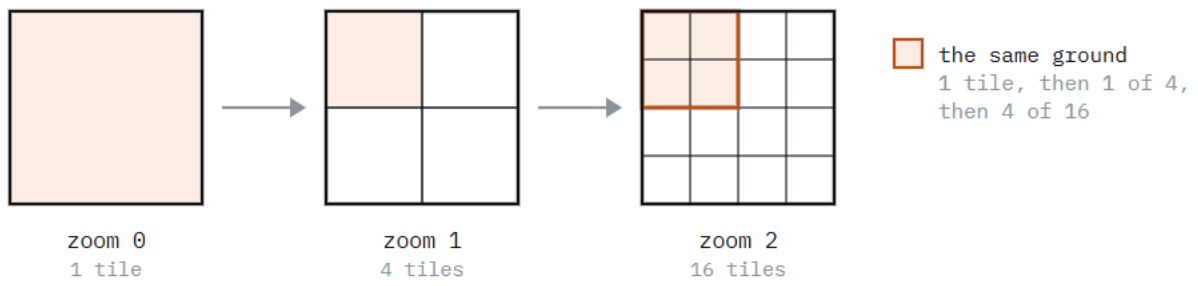
How tiling works

The map is not one enormous image that gets scaled. If it were, the whole world at street detail would be far larger than any machine could hold. Instead the imagery is pre-cut into small squares called **tiles**, at many levels of detail, and the engine fetches only the handful that cover what you are actually looking at.

The pyramid

Level 0 is the entire world in a single tile. Each level down splits every tile into four, so there is twice the detail in each direction. Level 1 is 4 tiles, level 2 is 16, and so on — by level 12 the world is more than sixteen million tiles, which is why nothing but the visible ones is ever requested.

Copyright © 2026 Vizrt



Two kinds of tile

Some tiles arrive as finished pictures — satellite photography, for instance, which can only be an image. Others arrive as compact descriptions of what is there: this road, that river, this place name. The engine draws those itself using the style.

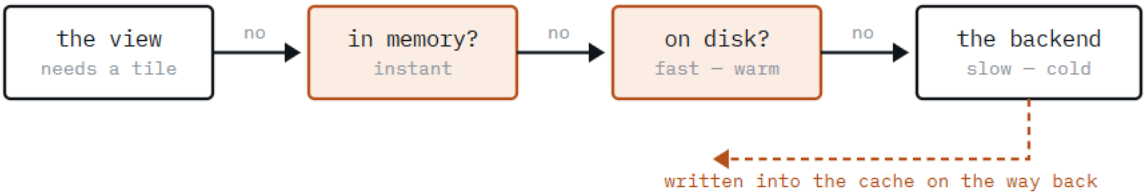
That difference explains two things you will notice. Labels and roads stay crisp when you zoom, because they are being drawn at your screen resolution rather than enlarged from a picture. And the same underlying data can look completely different under two styles, because the style is doing the drawing.

Where detail runs out

Each source of imagery covers a range of zoom levels, and they are not all the same. Photographic imagery over a city may go very deep while a specialist layer starts only at a middling zoom. Push past what exists for a remote area and the map stops getting sharper — that is the data ending, not a fault in your scene.

How caching works

Every tile the engine fetches is kept in a local cache on the machine. The next time you look at the same place at the same zoom, it comes off the disk instead of the network — which is why a map you have already viewed appears instantly, and one you have never viewed takes a moment.



This is the single most important thing to understand before you go on air: **a map you have rehearsed is not in the same state as a map nobody has opened yet.**

Copyright © 2026 Vizrt

Cold and warm

A *cold* view is one whose tiles are not cached: a location nobody has visited on this machine, a style just switched to, or a zoom level not yet used. It needs the network. A *warm* view is already on disk and needs nothing.

> If your map design is part of a playlist element, this is handled by the control application during initialization, the essential step before playing out elements.
> Otherwise, fly the map through the places and zoom levels you will use on air. This fills the cache, so the on-air move is a warm one. Treat it like preloading clips.

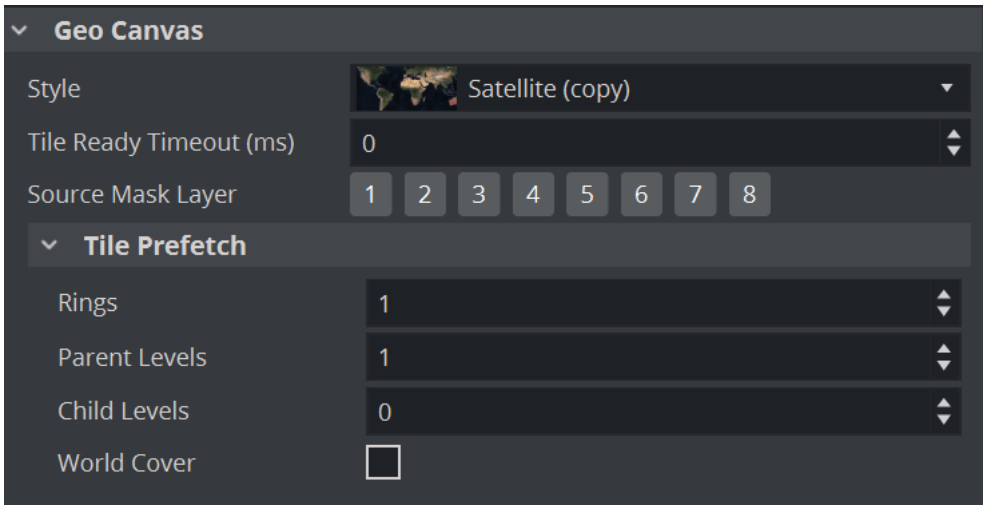
Ready

The map reports whether it has everything it needs for the current view. Anything that captures or hands over the picture — a snapshot, a transition, a take — should wait for that rather than a fixed pause, because how long it takes depends on the network and on what is already cached.

There is also a **tile ready timeout**: the longest the map will wait for imagery before drawing what it has. Raise it for slow connections or deep cold zooms; a map that goes on air half-drawn usually needs a longer timeout, not a longer sleep.

Prefetching

The map can fetch more than the view strictly needs, so the next move is already warm.



Three settings control how much:

		Prefetch settings
Setting	**Fetches ahead**	**Helps with**
Rings	Tiles around the edge of the view	Panning — moving sideways across the map.

Copyright © 2026 Vizrt

Child levels	Levels below the current zoom	Zooming in.
Parent levels	Levels above the current zoom	Zooming out.

Be aware, every child level counts times 4, keep this number low otherwise it will exponentially increase your cache size.

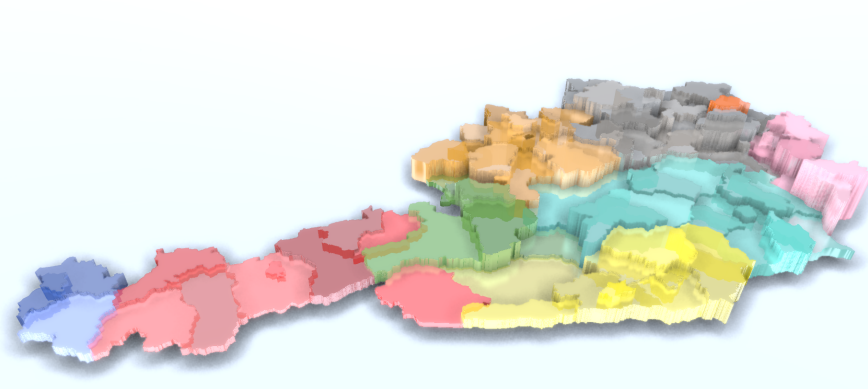
All three cost network traffic and cache space, so they are off by default. Turn on the one that matches the move you are about to make: rings for a pan, child levels for a push in. Prefetching is an optimisation — it makes a move more likely to be smooth, it does not guarantee it.

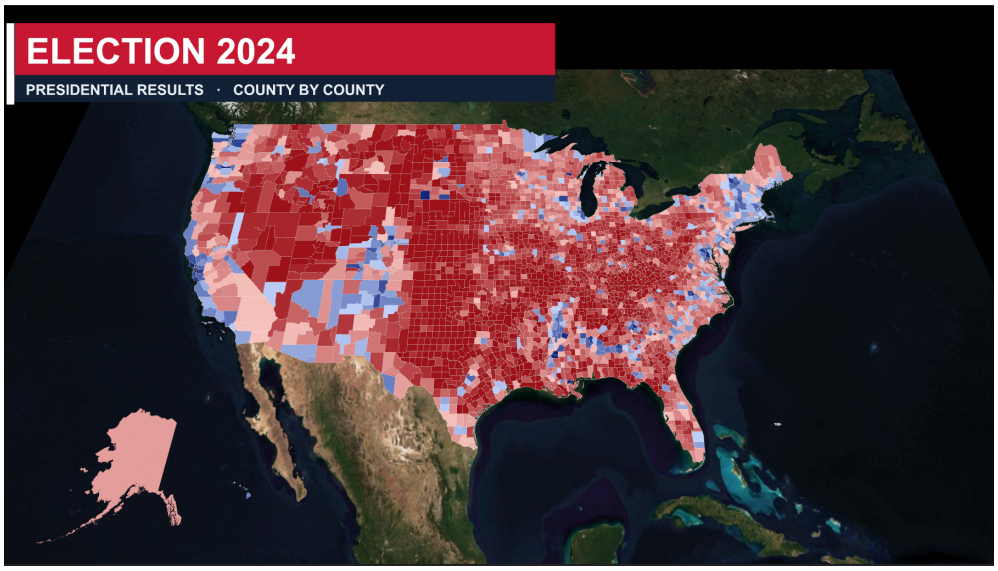
What prefetching is not
It improves the odds that a move is smooth; it does not guarantee it. On a cold machine with a slow link, a fast move can still outrun the imagery. Prefetching plus a rehearsed cache is reliable — prefetching alone is not.

Copyright © 2026 Vizrt

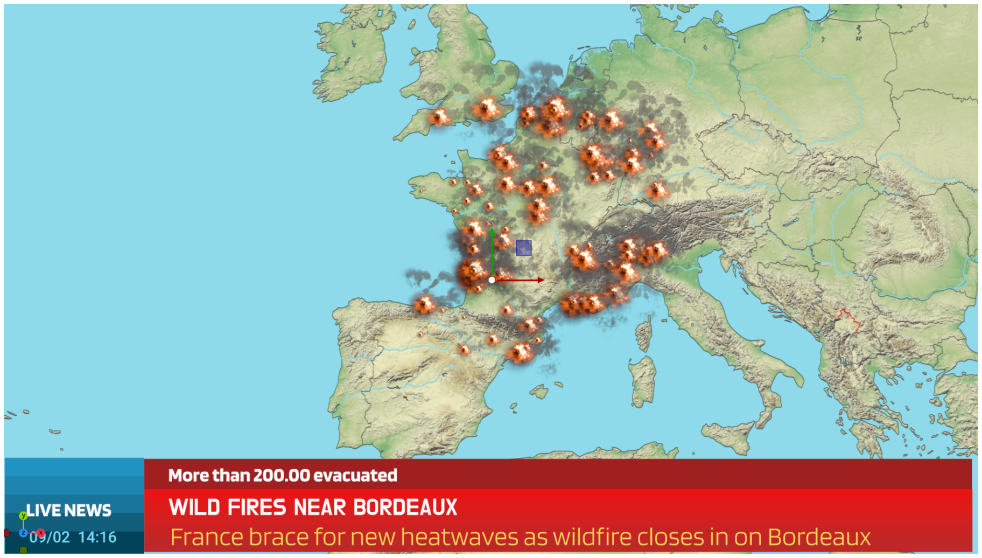
6.4 Working with GeoJSON

Borders, regions, routes and points of interest, brought into a scene as real geometry — where the data comes from, how it draws, and how to hand the control of it to an operator.





Copyright © 2026 Vizrt



6.4.1 What is a GeoJSON

GeoJSON is the standard, plain-text format for geographic shapes. Every open-data portal publishes it, most GIS tools export it, and a file is readable enough that you can open it and see what is inside. It carries three kinds of thing:

- **Points** — a single position. Stations, incidents, venues.
- **Lines** — a run of positions. Routes, streets, rivers, borders, a journey.
- **Polygons** — a closed outline, optionally with holes. Countries, districts, zones.

Viz Engine turns those shapes into geometry in your scene. It is not an image laid over the map — it is geometry, so it sits in the world at the right coordinates, holds its place as you fly, and can be lit, colored, animated and composited like anything else you build.

The following chapters explain how to search and use GeoJSON, import and prepare your data, how to use it in combination with Viz Engine's Maps capabilities.

- Importing GeoJSON
- Rendering GeoJSON
- Textures on GeoJSON
- Adding GeoJSON data

Copyright © 2026 Vizrt

6.4.2 Importing GeoJSON

What a GeoJSON can contain

A GeoJSON file is normally a **collection of features**. Each feature is one thing like a station, a district, a route and carries two parts: its *geometry*, the shape itself, and its *properties*, whatever the publisher chose to attach, such as a name, a population or a status.

Viz Engine only processes the geometry data, any labels or similar are ignored.

One GeoJSON can hold a mixture. A district dataset might have a polygon per district and a point per town hall in the same collection, and it will draw as one shape.

The geometry types

All seven types from the GeoJSON standard are supported. They fall into three families, and the family decides how the shape behaves when you switch between fill and line.

		Geometry types and how they draw
Type	**Is**	**How it draws**
Point	A single position.	Marked at its coordinates. Nothing to outline, so fill and line look much the same.
MultiPoint	Several positions as one feature.	As above, all of them together — every fire station in a district as a single shape.
LineString	A run of positions.	A line. This is the family the reveal animation is for: routes, rivers, borders, journeys.
MultiLineString	Several separate runs.	Several lines as one shape — a river with its tributaries, a road that is interrupted.
Polygon	A closed outline, optionally with holes.	Fill draws the area, line draws the boundary. Holes are cut out of the fill, so a lake inside a region stays a lake.
MultiPolygon	Several closed outlines.	One shape made of separate areas — a country with islands, a constituency in two parts.
GeometryCollection	A mixture in a single feature.	Each part drawn as its own kind. Useful when one thing genuinely is a point and an area at once.

Copyright © 2026 Vizrt

Two forms of geometry

The same GeoJSON file can become two different things, and which you pick decides what you can do with it afterwards.

GeoJSON geometry

A geometry object in its own right, loaded from a file or from the database and dropped on a container like any other geometry. It is the straightforward choice for a shape that is simply *there* — a country outline that never changes, a set of regions that is part of your template's furniture.

It also has preloading controls, which matter for large datasets: you can ask for extra detail levels to be prepared up front, and for parent levels to be kept, so the shape does not visibly rebuild as the camera moves.

A GeoJSON node in a Dynamic Geometry

The flexible form. Here the shape is a *node* inside a dynamic geometry, which means it is generated on demand and can be changed while the scene is running — new data, a different draw mode, a different thickness, a reveal animation.

This is the form to choose when the data changes, when an operator needs control of it, or when you want the shape to animate. Everything from the fill-and-line settings onward in this chapter applies to it.

> **GeoJSON geometry** - Fixed shape that is part of the design.
> **GeoJSON node in a dynamic geometry** - Anything an operator loads, changes or animates. If you are not sure which one to use, take the dynamic geometry node. it can do everything the plain geometry can and more.

Where the data comes from

Usually a GeoJSON is served by the Maps backend server. However the data can also be provided by an external file, as part of a merged geometry on the Grapchi Hub or as external URL.

The source is set through the **ControlGeoJsonGeometry** plugin, on its `input` field. That field accepts three kinds of value (x,y,z), and the plugin works out which it has received.

> GeoJson are often saved as .json instead of .geojson.
> Please make sure to rename it to .geojson to allow Viz Engine to pick it up.

Static GeoJSON on Graphic Hub:

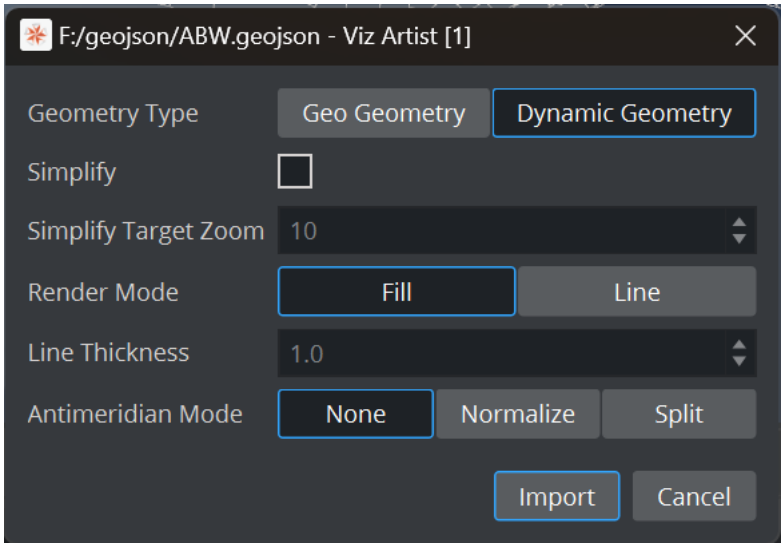
Pointing a scene at a file on disk is fine while you are designing. For anything you intend to keep, you rather bring the data into Graphic Hub, where it becomes a geometrical object like any other. You can just simply drag the file into Artist.

> Importing is done as a background task. It might take some time until your file appears in the Asset view.

Copyright © 2026 Vizrt

Two formats can be dropped in

- **GeoJSON** (`.geojson`) — the plain-text standard that every open-data portal publishes.
- **FlatGeobuf** (`.fgb`) — the same shapes in a compact binary form. Much smaller and quicker to read, and increasingly offered alongside GeoJSON. For a large dataset, take the FlatGeobuf if there is one.

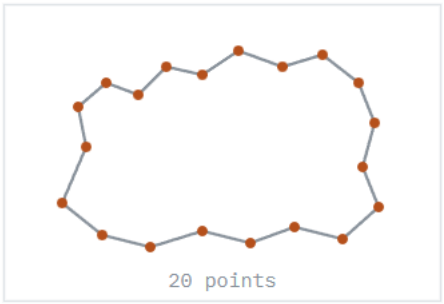


Choose your Geometry type (see above).

Simplification

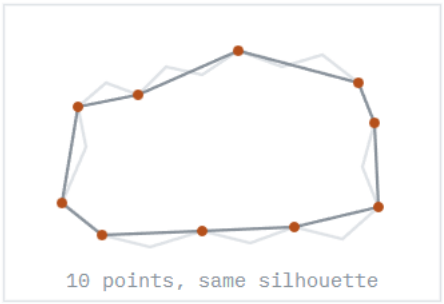
Open data is often far more detailed than a broadcast graphic needs. A coastline surveyed to the metre can carry tens of thousands of points; on air, at the zoom you are actually using, most of them land inside the same pixel.

Simplify thins those points out, and **target zoom** tells it how much detail to keep: the level you intend the shape to be seen at. Set it to the zoom you design for and the shape keeps the detail that shows, and drops the detail that does not.



As published

every surveyed vertex



Simplified

for the zoom you will use

Copyright © 2026 Vizrt

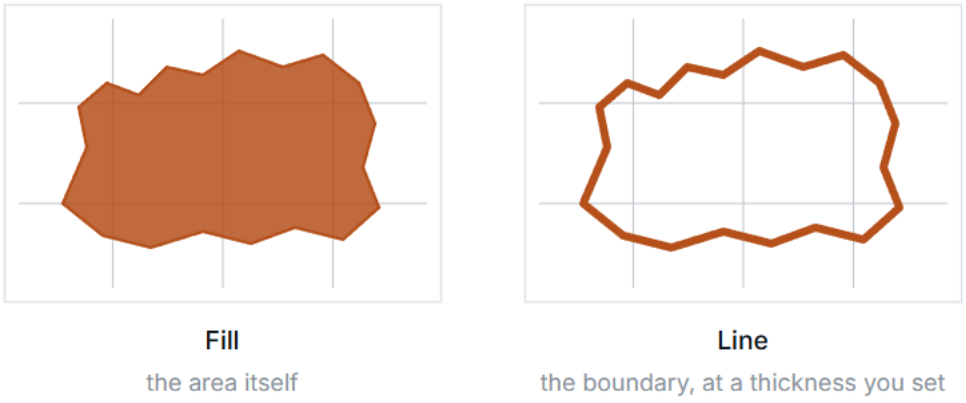
Dynamic Data

There are three ways to get the data into the **ControlGeoJsonGeom**:

A file path	Data that ships with the scene or lives on the machine. You need to take care the data is available on every single engine playing this scene.
A URL	Data published elsewhere and fetched at run time. Live feeds, an open-data endpoint, a newsroom service. This is mainly being used when operating in combination with Viz Maps backend.
Inline text	Directly copy/paste the content of a GeoJSON. Very small shapes only. This is limited to 64K. A command carries at most about 64 KB of text, and real GeoJSON datasets are usually far larger — a district boundary at full precision runs to hundreds of kilobytes. Inline text works for a handful of points or a simple polygon; for anything real, use a file path or a URL. If a shape silently fails to appear after pasting text, this is almost certainly why.

Fill and Line Mode

A shape is drawn in one of two ways depending on the chosen render mode: **filled** as a solid area, or drawn as an **outline**. It is a single setting but it changes the whole character of the graphic.



Line options:

Line thickness

In line mode the thickness is yours to set, and the control plugin also carries a **minimum** and a **maximum**. Those two are not decoration: they become the bounds of the control an operator is given, so they are how you stop a

Copyright © 2026 Vizrt

hairline that disappears on air or a slab that swamps the map. Set them to the range you are happy to see broadcast.

Revealing a line

A line does not have to appear all at once. Two settings turn a route into an animation: **progress**, how much of the line is drawn, and **start offset**, where the drawing begins.



Antimeridian Settings

The line at 180 degrees, where longitude wraps from east to west, is where shape data traditionally breaks. A route from Tokyo to Los Angeles crosses it; so does anything covering the Pacific, Fiji, New Zealand's outer islands or Russia.

The difficulty is that a shape spanning the line has coordinates at both extremes — some near `179`, some near `-179`. Read naively, the shortest path between those two numbers goes the wrong way round the planet, and what should be a short hop across the Pacific becomes a band stretched across the entire map.



How data expresses a crossing

Publishers solve this in three different ways, and which one you have decides what you should do about it. At import time, we do not know how the data is structured, it can be one of the following:

Copyright © 2026 Vizrt

In the data	Looks like
Natural wrap	Coordinates stay within −180 to 180 and simply jump: `179.5` then `-179.5`. The commonest and the most ambiguous — nothing in the file says whether that jump is a short hop or a trip round the world.
Extended coordinates	Longitude is allowed past the limit: `179` continuing to `181`. Unambiguous, and technically outside the standard, but widely produced by GIS tools.
Split at the meridian	The shape is already cut in two, as the standard prescribes: an eastern piece from `179` to `180` and a western piece from `-180` to `-179`, held together as a MultiPolygon. This is what a well-formed dataset does — Fiji is usually published this way.

The GeoJSON node has an **antimeridian mode** with three settings, which decide what the engine does with the coordinates it was handed.

None	Takes the coordinates exactly as given, without interpretation.	The data is already correct for how you want it drawn — typically a properly split dataset. Start here.
Normalize	Brings coordinates back into the standard −180 to 180 range.	The file uses extended coordinates that run past the limit, and you want them folded onto the map normally.
Split	Cuts shapes at the meridian into separate pieces either side.	A shape spans the line as one piece and is drawing as a band across the world. This is the repair.

Leave it on **None** and look. Most datasets are fine, and a Pacific one that draws correctly needs nothing. If a shape distorts across the map, try **Split**. If a shape is missing or is drawn far off the edge of the world, its coordinates run past 180 — try **Normalize**. It takes seconds to try all three, and the right one is obvious on screen.

The setting has no effect on data that never approaches 180 degrees - Therefore an European or American dataset will look identical in all three modes. Judge it on the Pacific, not on the shape you happen to have open.

Copyright © 2026 Vizrt

6.4.3 Rendering GeoJSON

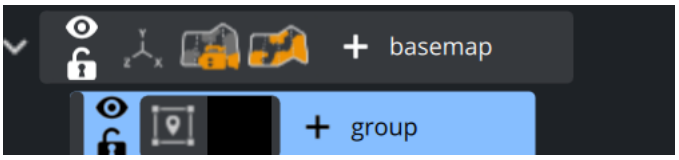
As GeoJSONs hold geographical information, we can also use those geo referencing inside our maps workflow:

Example

In our example we will use a GeoJSON of Scotland to visualize distilleries on the “Isle of Islay”. We could also use the data we get out of the Maps Server however as we need a hihger resolution of the shape data, we will use external sources.

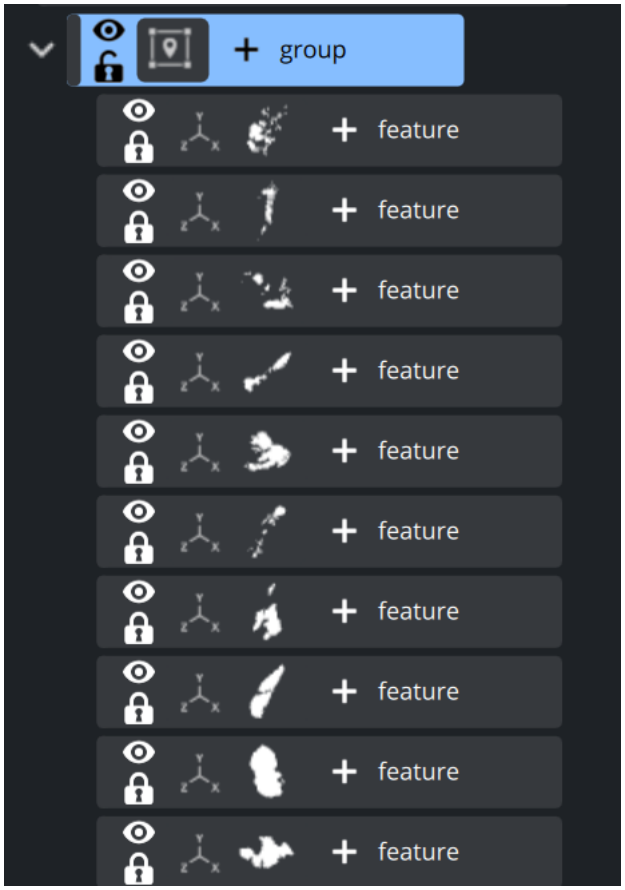
Let's do this step by step:

1. Create a Basic Map Container by adding **GeoCanvas** or **GeoCam** to a base container.
2. Choose your style and use the GeoCam or the keyboard “M” to navigate to your desired location.
3. Download a static GeoJSON file and drag and drop it into Artist as a dynamic geometry.
4. Create a sub container and switch it to be **GeoTransformation** Mode.
5. Drag the imported GeoJSON to that container. Most likely the rendering will show its BoundingBox, but no geometry.
6. Click on the **GeoTransformation** and change the z Offset to be 1:

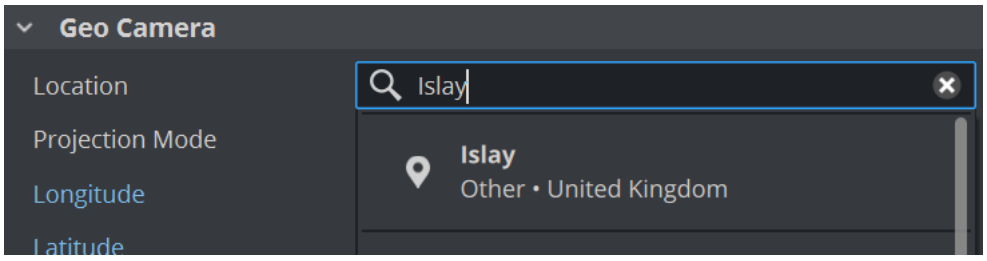




If we split the imported geometry, you’ll see the geometry holds a set of districts or counties:



As we only need to render the Isle of Islay, we now use the **GeoCamera** plugin to get closer to this Island.



Copyright © 2026 Vizrt



Click on the centered Island and the related GeoJSON will be highlighted in the Scene Tree.
We can now delete or hide the others as they will not be used.

Change the zoom so that we get closer, and also modify the Pitch inside the **GeoCamera** to have a more tilted view.
We will now extrude the geometry and add some bevel to it:

If you click on the Geometry Icon, you'll see it has been imported as a Dynamic Geometry, so we can now add a modifier to it. But first add a basic PBR or Phong material to it.
Next add an Extrusion Modifier to it.

> The extrusion currently needs to go negative, so therefore we need to lift the whole GeoJSON group inside the BoxTransformation a bit higher.

Copyright © 2026 Vizrt



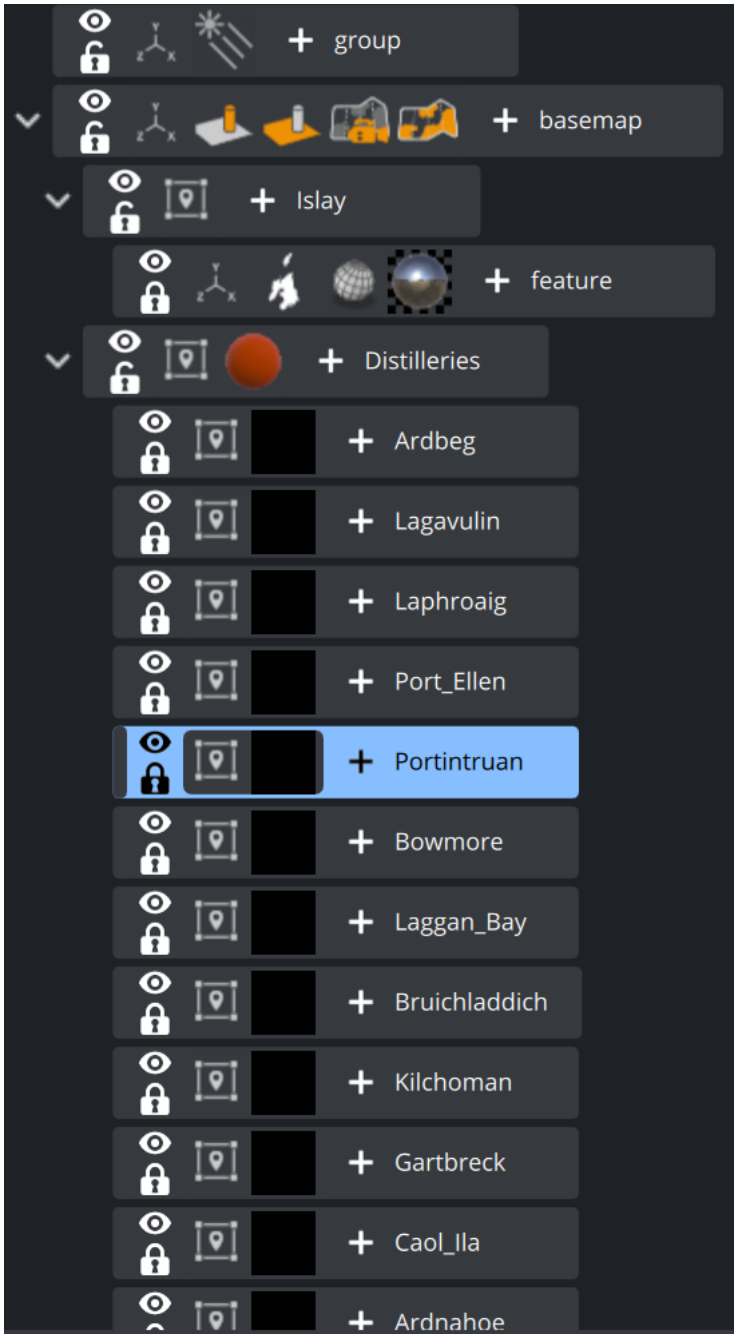
As we want to show the location of the most famous distilleries there, we will now import a GeoJSON with point data of these locations.

> For this example, we asked Anthropic Claude to generate a GeoJSON with all distillery locations.

Import the file as in the previous step. Create a Subcontainer, set it to GeoTransformation and apply the imported file.
If you split the file, the locations will show up. As this is pure point data only, there will be no geometry created, but as we only need to know the locations, we can simply drop any geometry, like a pin or marker to it.

It might be required to adjust the scaling and the offset as we have already have an extrusion on our map.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt



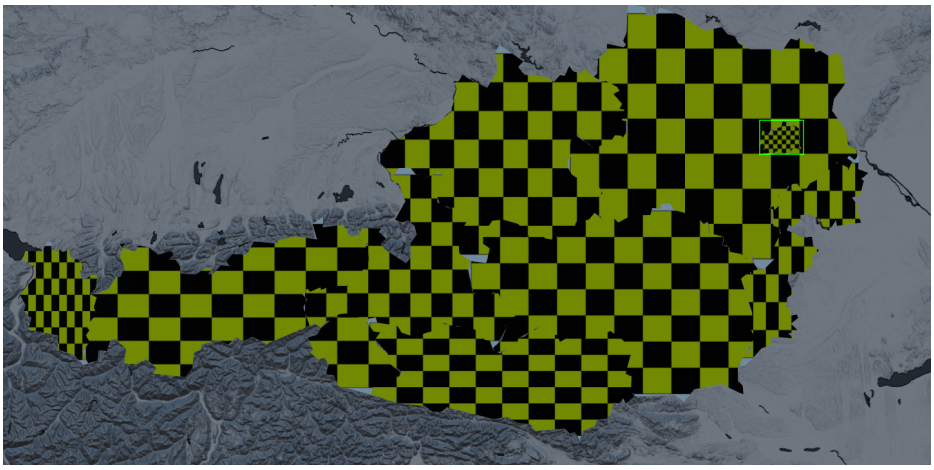
Copyright © 2026 Vizrt

6.4.4 Textures on GeoJSON

Mapping Textures in Fill Mode

A GeoJSON geometry builds its own texture coordinates for the filled shape, so any material texture you assign has something to map onto. **UV Options › Mode** decides how those coordinates are worked out — either fitted to the shape, or anchored to the world.
Both modes produce the same kind of result — a texture across your filled geometry. The difference is what the texture is measured against, and that changes everything else about how it behaves.

Bounding Box:



The texture is stretched to fill the shape's extent exactly once, corner to corner. Whatever you assign shows up whole, whether the shape is a city block or a continent, and it looks the same however far you zoom the map in or out.

Reach for it when

- You want one picture to fill one shape — a flag inside a country, a photo inside a region, a team colour ramp inside a district.
- You are using a gradient and want it to run the full width or height of the shape.
- The shape needs to look the same in a wide establishing shot and a tight close-up.

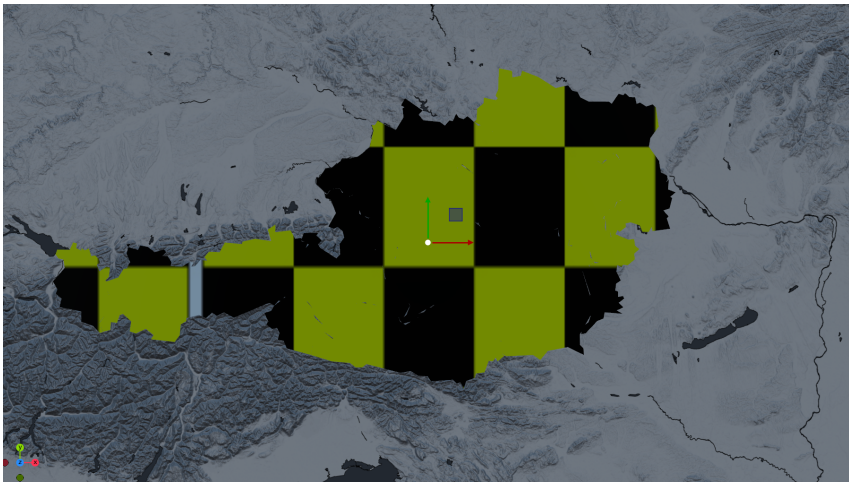
Worth knowing

- Width and height are fitted independently, so the texture takes on the proportions of the shape. A circle in your image becomes an oval on a wide, flat shape.
- One box covers *everything* in that geometry. Load a whole country's districts into a single geometry and the texture spans the entire set once, not once per district.
- Because the fit follows the data, the texture shifts and rescales whenever the extent changes — loading a different file, or adding a feature further out.

Copyright © 2026 Vizrt

- Two separate geometries get two unrelated fits, so a pattern will not carry across the border between them.
- The **Zoom** value has no effect here.

World:



The texture is pinned to the map itself and repeats every so many meters on the ground. Each shape shows whatever part of that endless pattern it happens to cover, so the pattern has a real physical size and a fixed position — and it runs continuously from one shape into the next.

Reach for it when

- You want a repeating fill — hatching, cross-hatch, dot screen, noise, water ripple, terrain — at a believable size rather than stretched to fit.
- Several shapes must read as one surface: neighboring regions, a country split at the antimeridian, or an imported dataset where every feature becomes its own geometry.
- You want the pattern to stay on the map, even when the data behind changes.
- Small and large shapes appear together and should share the same visual texture weight.

Worth knowing

- Set the material's wrap to **Repeat**. The coordinates run well past 0–1, and on Clamp you will see one smeared edge pixel instead of a pattern.
- It never auto-fits. A small shape at a low **Zoom** may only show one flat corner of the texture — raise Zoom until the pattern reads.
- The grid is anchored at the map origin, so the pattern lands in the same place on the same ground every time you build the scene.
- At high **Zoom** values the pattern repeats very densely and can shimmer. Mip-mapped textures handle this far better than sharp line art.

Copyright © 2026 Vizrt

Zoom Levels (World Mode only)

Zoom works like a map zoom level: each step up halves the tile, so the pattern repeats twice as often. It ranges from 0 to 20, and only matters in World Tile mode.

Zoom	One tile spans	Reads as
0	40 075 km	The whole world in one tile
2	10 019 km	Continental
4	2 505 km	Several countries
5	1 252 km	A large country
8	157 km	A region or state
10	39 km	A metro area
12	9.8 km	A city
15	1.2 km	A neighbourhood
18	153 m	A few city blocks
20	38 m	A building footprint

Distances are measured at the equator. Because the map is projected, the same tile covers less real ground the further you go towards the poles — roughly half as much at 60° latitude as at the equator.

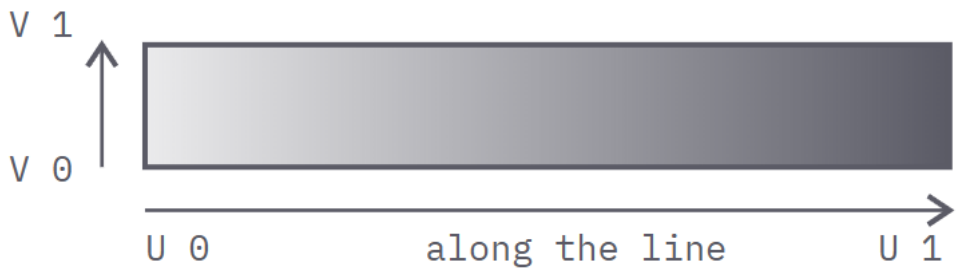
Import defaults. When you import a `.geojson` or `.fgb` file as dynamic geometry, the import options carry the same two settings and stamp them onto every geometry the import creates. The importer makes one geometry per feature — which means on Bounding Box each feature is fitted separately, and on World Tile they all share one grid.

Mapping the outline

Switch **Render Mode** to **Line** and the geometry stops building a filled shape and strokes the borders instead. That stroke is its own mesh with its own texture coordinates, so it gets its own setting — **Line Options › Tex Coord Mode**. UV Options play no part here; changing UV Mode while rendering lines won't make any visible changes for you.

The two axes are fixed, and worth knowing before you make artwork for it: **U runs along the line**, from the start of the stroke to its end, and **V runs across the thickness** — 0 on one edge, 1 on the other. So a texture for an outline wants to be drawn as a strip: put the motif that should travel along the border across the width of your image, and whatever should vary from edge to edge up its height.

Copyright © 2026 Vizrt



Contour Normalized

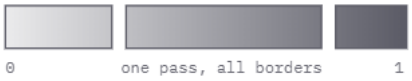


Every separate border runs U from 0 to 1 on its own. Each one shows the complete texture exactly once, however long or short it is — so a short border compresses the image and a long one stretches it.

Reach for it when

- Each border should read as its own object: one arrow per border, one fade per border, a ribbon that starts and ends at the edge of each shape.
- You are wiping borders on with **Progress** and want the leading edge of the texture to travel with the wipe.

Normalized



U runs 0 to 1 once across *all* the borders in that geometry, each picking up where the previous one left off. Every border shows only its share of the texture, in proportion to its length.

Reach for it when

- You want a single color ramp or gradient to run across an entire dataset rather than restart on every border.
- The borders should read as one continuous run of line, not as separate pieces.

Copyright © 2026 Vizrt

Unnormalized



U is the distance travelled along the line, not a fraction of it. The texture repeats every fixed step of length, so the pattern is the same size on every border and long borders simply carry more repeats.

Reach for it when

- You need dashes, ticks, chevrons or a rope pattern at even spacing, regardless of how long each border happens to be.
- You are animating **Progress** and the pattern must hold its size instead of stretching as the visible length grows.

Unnormalized needs two things set up:
Set the material wrap to **Repeat**, and turn the texture scaling *right* down. U here is a raw map distance, so one repeat of the texture covers one unit of map distance - across a country-sized dataset that is thousands of repeats and reads as noise. Scale the material texture until the pattern is the size you want; for roughly one repeat per kilometre you are scaling by about a thousandth.

	Contour Normalized	Normalized	Unnormalized
U is measured against	the length of each border	the total length of all borders	distance travelled
Texture shows	once per border	once across the set	repeats by distance
Pattern size, short vs long border	Different	Different	Same
While Progress animates	Re-fits to the visible run	Re-fits to the visible run	Holds its size
Wrap mode needed	Any	Any	**Repeat**

Tex Coord Mode sits with Thickness, Progress and Progress Mode under Line Options, and is set per geometry.

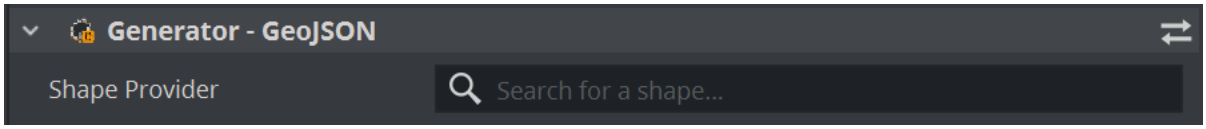
Copyright © 2026 Vizrt

6.4.5 Adding GeoJSON data



The quickest and easiest way to add GeoJSON shapes to your design is to ask the Maps Backend for it. However there are more possibilities how to add other sources. See Importing GeoJSON for details.

To start we will use the built-in search:

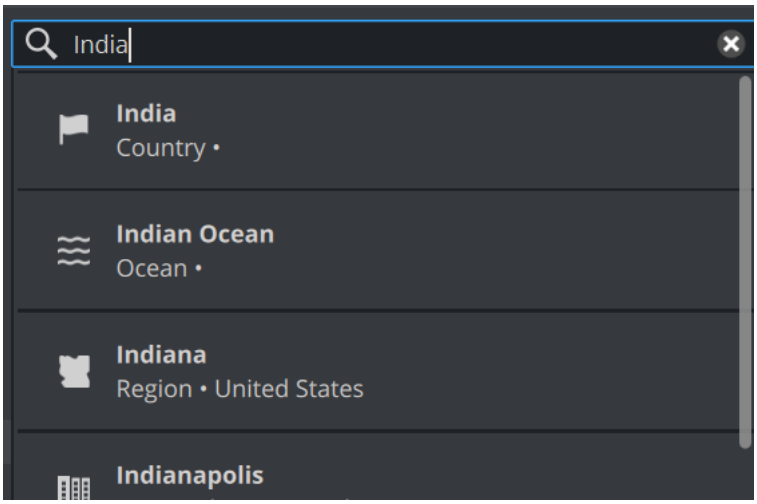


- Add a Container with **GeoCanvas** and **GeoCamera** on it.
- Create a sub container and add the GeoJson plugin on it. This will automatically change the transformation mode to geo transformation so we are able to work in latitude and longitude mode.

The GeoJSON is a plugin used by the Dynamic geometry rendering method. Clicking on the icon will show the Generator options and the Shape Provider input field.

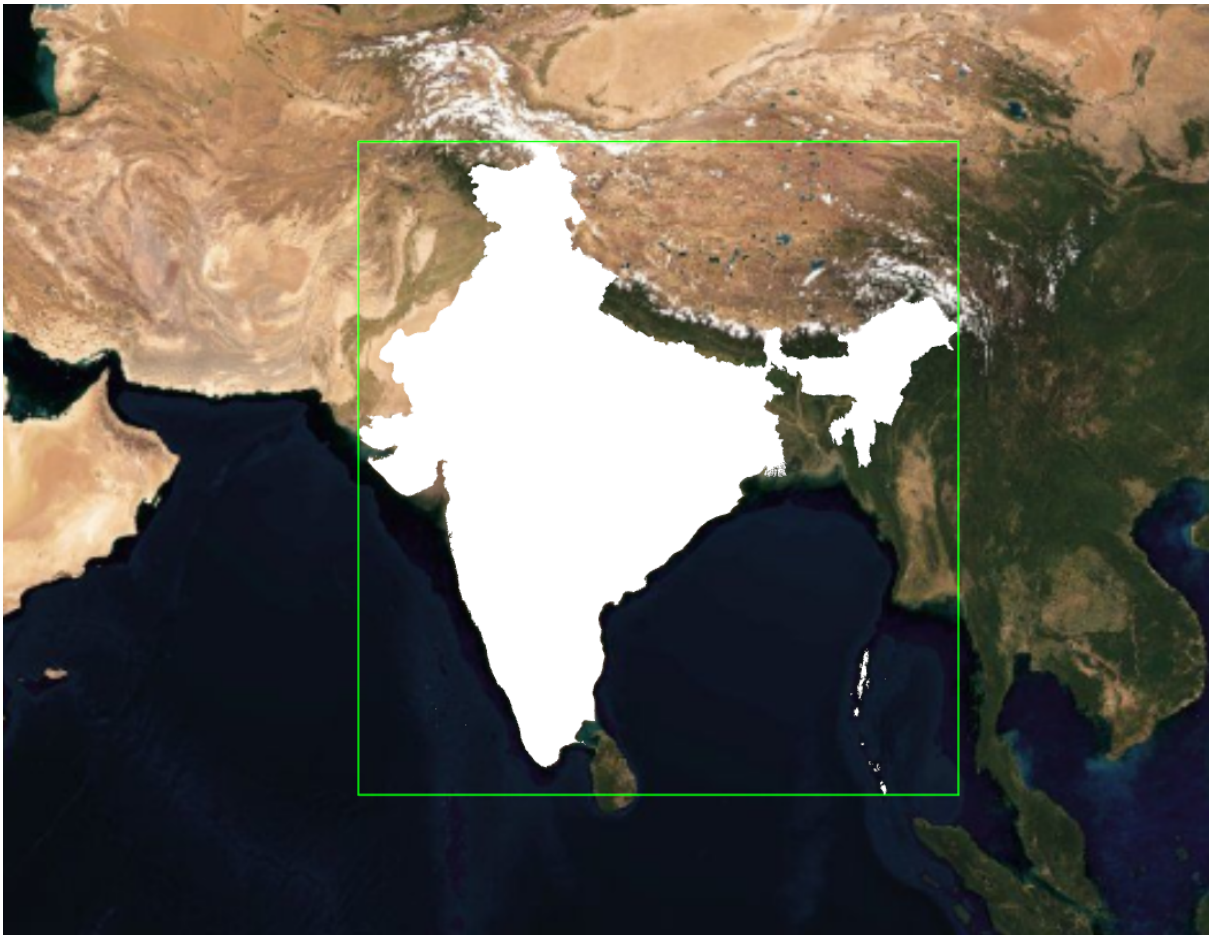
Type in what you are searching for. This can be e.g. a country outline, a location and even more.

- Lets search for “India”:



The search result now adds Countries, Oceans, Regions etc…
We will choose **India** as a country.

Copyright © 2026 Vizrt



The shape data is being fetched from the Maps Backend and rendered directly on our map.
Now we can style the shape. See Rendering GeoJSON and Textures on GeoJSON.

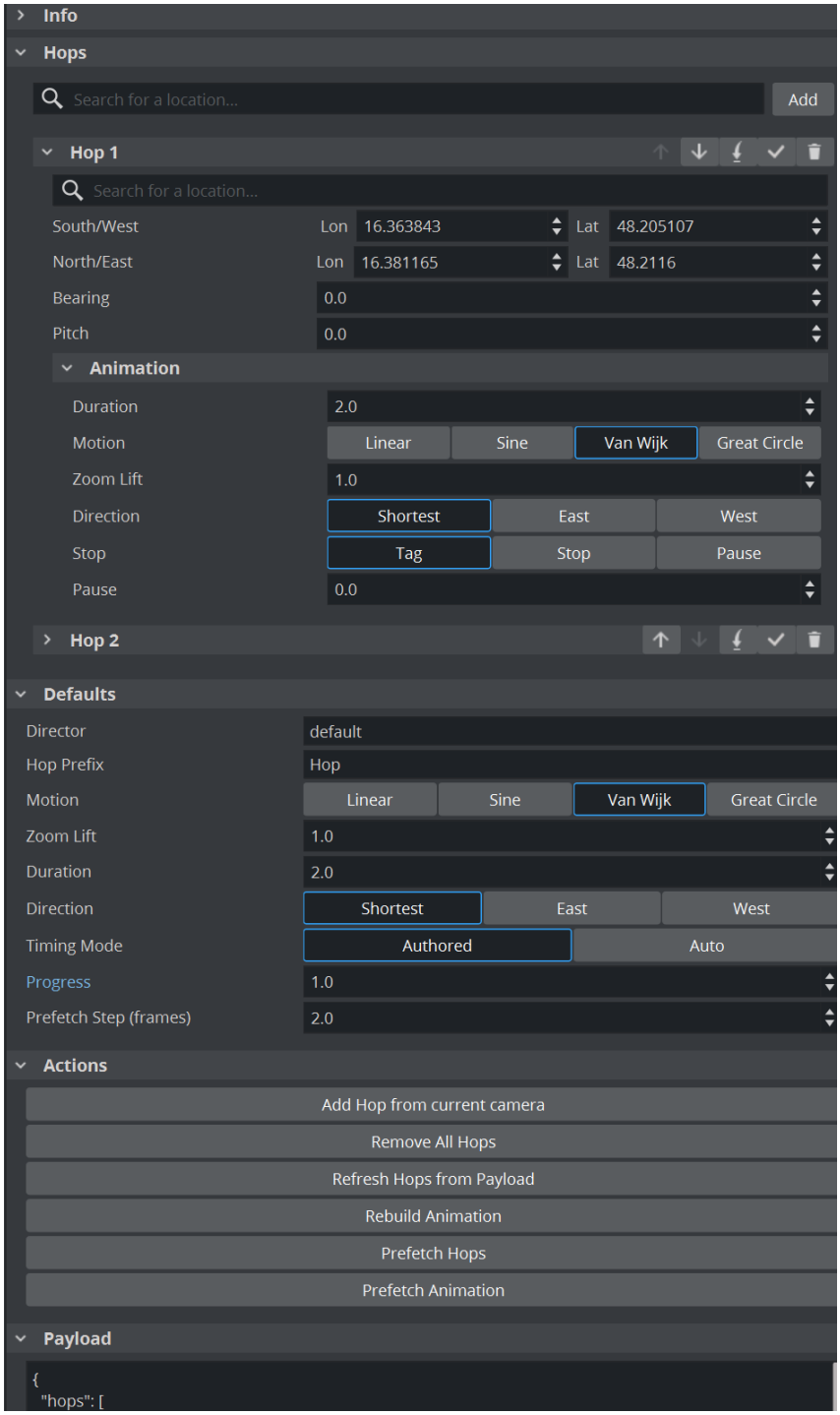
Copyright © 2026 Vizrt

6.5 Working with Animations



GeoNavigator turns a list of places into a camera flight over your map. You mark the stops you want to visit, choose how the camera should travel between them, and it builds a real animation on a director — one you can play, scrub and trigger like any other.

Copyright © 2026 Vizrt



Moving a map camera by hand is tricky. You can animate longitude, latitude, zoom, bearing and pitch separately, but getting them to peak together in the middle of a move — so the camera pulls back, travels, and settles again — means hand-tuning five curves that all have to match.

GeoNavigator does that for you. You give it an ordered list of places, called **hops**, and it figures out the whole camera move between them: the path across the globe, the pull-back in the middle, the easing at each end. Then it writes the result onto a director as a normal animation.

Copyright © 2026 Vizrt

6.5.1 How a flight is built

Three things make up a flight, and it is worth keeping them apart in your head, because each one is edited in a different place.

The hops

Where you stop

An ordered list of places. Each hop is a box on the map, not a pin — a minimum and maximum longitude and latitude. The camera frames that box when it arrives, so the size of the box is what decides how close in you end up.

The flight settings

How you travel

How long the move takes, how the camera eases, how high it climbs on the way, and which way round the globe it goes. Set once for the whole flight — and overridable on any individual hop when one leg needs to behave differently.

The director

Where it ends up

The stage director the finished animation is written to. GeoNavigator creates the keyframes on it. Nothing is animated until you ask for the build, and building again replaces what was there.

6.5.2 Example

Let's create a quick flight from Vienna to Rome.



Create a basic Map using GeoCanvas and GeoCamera,

Copyright © 2026 Vizrt

Put the plugin on the map

Drag the **GeoNavigator** plugin onto the container that carries your map — the same one holding the GeoCanvas and the geographic camera.

Look at the first place

Type *Vienna* into **Search Location** and select it. The Navigator will navigate our view to Austria, Vienna. (Do not press the Add button yet).
The map flies straight there, so you can see what you are about to capture. Nothing has joined the flight yet — searching only moves the camera. Think of it as scouting.

Frame the shot, then capture it

Use the GeoCamera or press the “M” and use the Geo Controls to zoom, pan and rotate until the map looks the way you want it to look when the flight arrives. Then press **Add Hop from current camera**.
That adds a row to the hop list holding exactly what you are looking at. This is the heart of the workflow: you compose the shot by eye and hand it over, rather than describing it as four numbers.

Do the same for Rome

Search for Rome, then frame it and press the **Add Hop from current camera** button. You now have two rows — the flight will run from the first to the second, in list order.

Say how long it should take

Set **Duration** to *4* seconds, and put a name in **Director** — *GeoNav* will do. Name a director of your own rather than accepting whatever is already there. It keeps the flight separate from the rest of your scene's animation, and makes it easy to find on the stage afterwards.

Build it

Press **Rebuild Animation**.

> This is the step people forget. Until you press it there is no animation, and **every change you make afterwards needs another press** — a new row, a different duration, a different motion. Nothing you change takes effect on its own.

Fly

Inside the Stage, there is now a GeoNavigator Channel with a progress animation. Start the animation and the camera flies from Hop 1 to Hop 2, using the flight you just built.

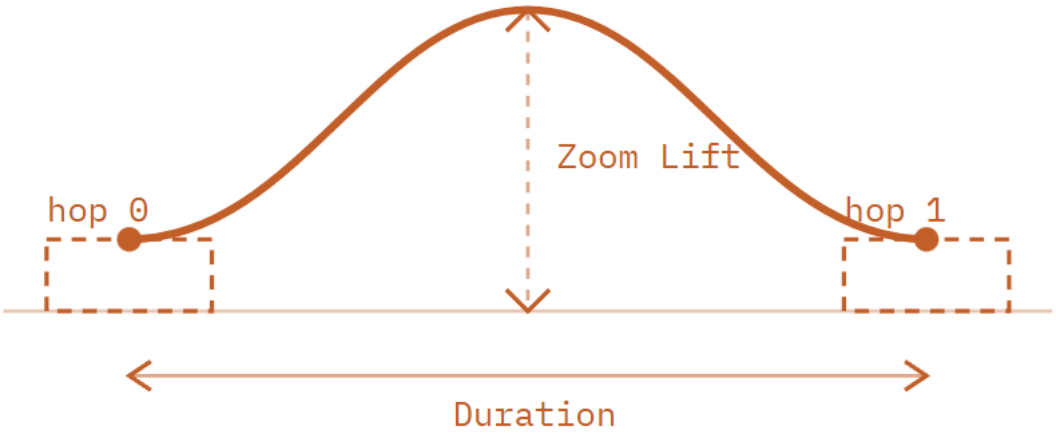
6.5.3 The settings in detail

These live in the plugin editor and apply to the whole flight. Any of them can be overridden on a single row — see Working with Animations — but start here.

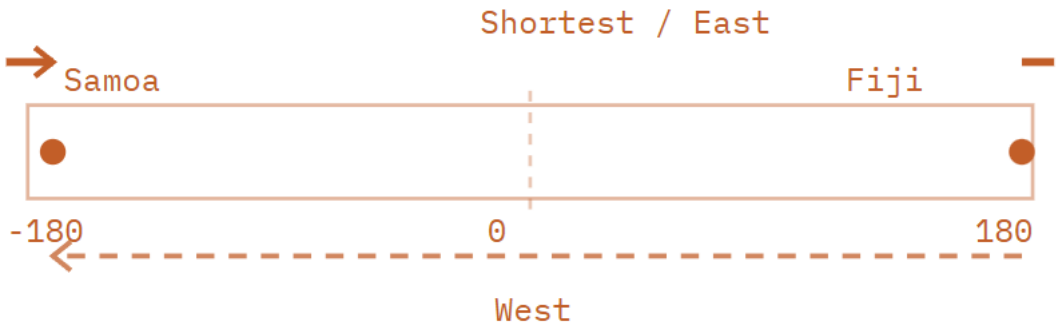
Copyright © 2026 Vizrt

Defaults	
Director	The name of the stage director the flight is built on. Name your own — a flight built onto a director you already use for something else will overwrite it the next time you build.
Hop Prefix	The naming stem for new rows, so they come out as *Hop1*, *Hop2* and so on. Worth changing when a scene carries more than one flight and you need to tell their stops apart at a glance.
Motion	How the camera travels between two stops. This is the single setting that most changes how a flight feels, so it is worth trying all four on a real leg. • **Linear** moves at a constant rate from start to finish. Mechanical, but predictable — and it is the one to pick when you want to shape the arc yourself with **Zoom Lift**. • **Sine** is Linear with the ends eased, so the camera accelerates away and settles gently. A safe choice for short, local moves. • **Van Wijk** works out its own pull-back so the ground appears to move at a comfortable rate the whole way. On a long flight it looks like a camera operator who knows what they are doing. Because it calculates its own climb, it ignores **Zoom Lift**. • **Great Circle** follows the true shortest path over the surface of the globe. On an intercontinental leg this is the one that reads as a flight rather than a slide — the path bows towards the pole the way an aircraft route does.

Copyright © 2026 Vizrt

Defaults	
Zoom Lift	How far the camera pulls back as it passes the midpoint — the height of the arc in the diagram above. *0* keeps the flight flat, at the zoom level of the two stops; larger values climb higher and show more of the world on the way.  **It does nothing while Motion is set to Van Wijk**, which is how the plugin arrives. If you drag Zoom Lift around and the flight does not change, this is why — switch Motion to Linear or Sine first.
Duration	How long the flight lasts. This sets the length of the animation on the director, so if the move has to fit a specific slot, set it here rather than time-stretching afterwards. Under **Auto** timing this becomes a starting point rather than a promise — see **Timing Mode** below.

Copyright © 2026 Vizrt

Defaults	
Direction	Which way round the globe the camera travels. It only makes a visible difference near the date line in the Pacific — but there it makes an enormous one. Fiji and Samoa are about ten degrees apart, but one sits just east of the date line and the other just west of it. Read as plain numbers that is a 350-degree gap, and a naive move drags the camera the whole way round the planet through Africa. **Shortest** recognizes the wrap and takes the ten-degree hop. **East** and **West** force a direction, which is what you want when the long way round is the point — a globe-spanning establishing move, or matching the direction of a graphic. 
Timing Mode	**Authored** means the durations you set are the durations you get. Use it whenever the flight has to fit a known slot — a voiceover, a fixed break, a hand-off. **Auto** lets GeoNavigator share the time out itself, so legs are paced against how far they actually travel instead of every stop getting an equal slice. On a tour that mixes a hop across a city with a hop across an ocean, Auto stops the short one crawling and the long one racing.
Progress	This is the base for the animation. travels from 0 to 1.
Prefetch Step (frames)	Defines how many tiles will be preloaded ahead.

6.5.4 Per-hop overrides

Every shaping setting above can also be set on a single row, where it applies to that stop alone. A row you leave alone simply follows the flight-level setting. This is how you build a tour that mostly behaves one way but slows down, turns, or pauses at one particular stop.

Within the Hops Accordions, some buttons help you to quickly apply your actions:

Copyright © 2026 Vizrt



The **arrow buttons** allow you to reorder your animation. Either move a hop up or down to change the order. The **trashcan** button removes the hop from the hop list.
Replace Hop with current camera: Frame your destination using the GeoCamera or the M shortcut to quickly replace the Hops data with your new view.

> Remember to press the “Rebuild Animation” Button to update the flight.

Set camera to hop: Replaces your current view to the stored Hop data. Use this to quickly jump to a hop without the need to play the animation.

Hops Settings	
Bounds South/ West North East	The box the camera frames on arrival. A tight box arrives close in, a wide box arrives pulled back. There is no zoom setting on a stop — the framing comes from the box. Typically a journalist will define an area and not a single location point.
Bearing	Which way is up when the camera arrives. Rotating the map between two stops is a cheap way to make an arrival feel deliberate
Pitch	How far the camera tilts towards the horizon on arrival. Flat overhead at 0, increasingly oblique above that
Animation	
Duration	See Default Settings
Motion	See Default Settings
Zoom Lift	See Default Settings
Direction	See Default Settings
Stop	What happens on arrival — carry straight on into the next leg, or come to a stop here
Pause	How long to wait on a hop, if Pause mode is used.

Copyright © 2026 Vizrt

Actions	
Add Hop from current Camera	Creates a new Hop using the current GeoCamera settings
Remove All Hops	Removes all current Hops
Refresh Hops from Payload	Reads the Payload Json again, All current settings will be overwritten by these data.
Prefetch Hops	Loads all Tiles required on the certain hop points.
Prefetch Animation	Loads all tiles required for the whole animation

The Payload

This is actually the storage with all information needed for the GeoNavigator. The JSON data is exposed, so it can be used by a control application to build a whole animation with one single command instead of creating it step by step.

6.5.5 Driving it from a template



A second plugin, **ControlGeoNavigator**, hands the flight to a control application so an operator can change it without ever opening the scene. Drop it onto the same container as GeoNavigator.

Give it a **Field Id** — the name the control application uses to find this flight — and then tick which settings an operator is allowed to touch. Each *Expose* checkbox adds or removes one field from what the template shows:

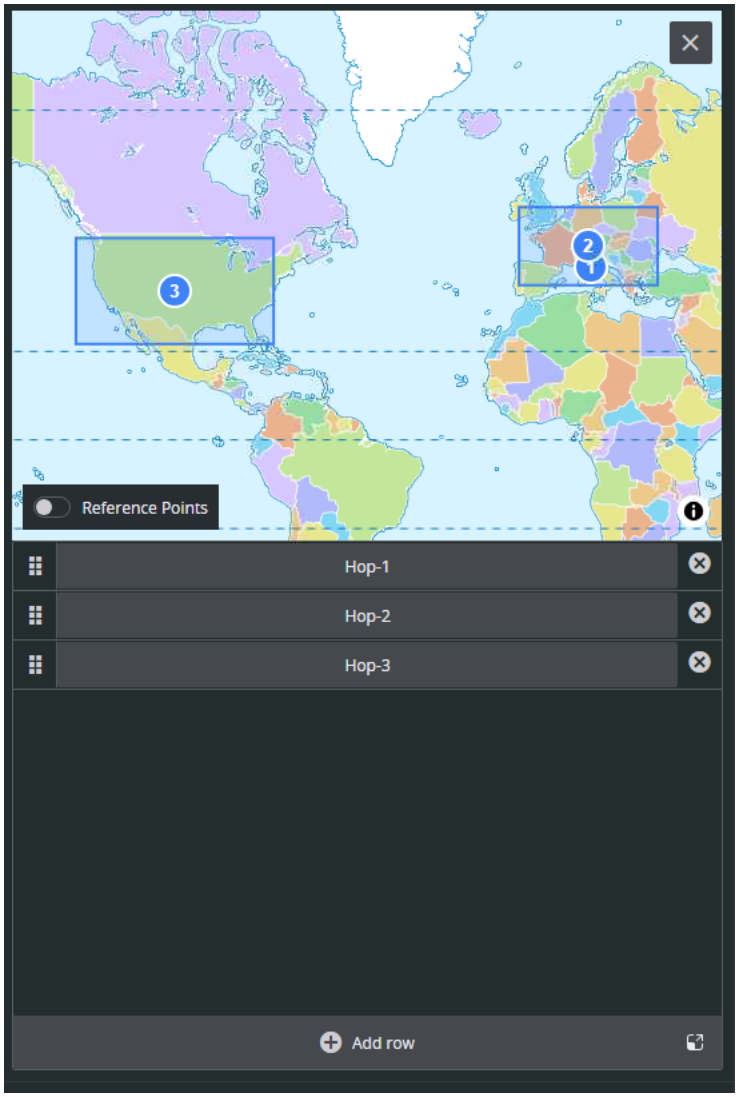
Expose…

Bearing, Duration, Flight Direction, Motion, Pause Duration, Pitch, Stop Type, Timing Mode, Zoom Lift and User Data are ticked to start with. Bounds As Text is not.

Untick everything the operator has no business changing. An exposed field is one more thing to get wrong on air, and a template offering three meaningful choices is easier to use than one offering eleven.

The operator then sees one set of these fields per stop, and edits them row by row exactly as you would in the scene — the values land straight on the flight. They still need the flight rebuilt afterwards, so if your template lets an operator change timing or motion, give them a button that rebuilds too.

Copyright © 2026 Vizrt



6.5.6 Tips and troubleshooting

Nothing moved - Rebuild after every change.
Hops, duration, motion — none of it reaches the animation until `update_animation` runs again. This is the most common reason a flight looks stale.

zoomLift does nothing - You are on Van Wijk.
The default motion computes its own zoom curve and ignores the manual value. Switch `motion` to Linear or Sine to take control of the arc yourself.

The flight will not play - Check which director it landed on. If the director you named is empty, look at the scene's default director before assuming the build failed — the flight may have been written somewhere you were not looking.

It went the wrong way round the world - That is the antimeridian.
Put `flightDirection` back to Shortest. Any leg between the eastern and western Pacific will do this if a direction is forced.

Copyright © 2026 Vizrt

Arriving too close or too wide - Resize the hop's box.
There is no zoom on a hop — the framing comes from the size of `bounds`. Grow the box to arrive further out, shrink it to arrive closer in.

Editing a captured hop - Write a fresh box.
A hop captured from the camera may not expose a `bounds` object you can edit in place. Building the box yourself and writing it back through `payload` always works.

Legs feel unevenly paced - Try Auto timing.
On a tour mixing short and long legs, `timingMode` 1 shares the time out by distance instead of giving every hop an equal slice.

Authoring by eye - Fly first, capture second.
Position the camera by hand until the shot is right, then use the `add_current`. Composing a shot as four numbers is much harder than recognizing a good one.

Copyright © 2026 Vizrt

6.5.7 Using GeoExpression

Making the map react to how far you are zoomed in



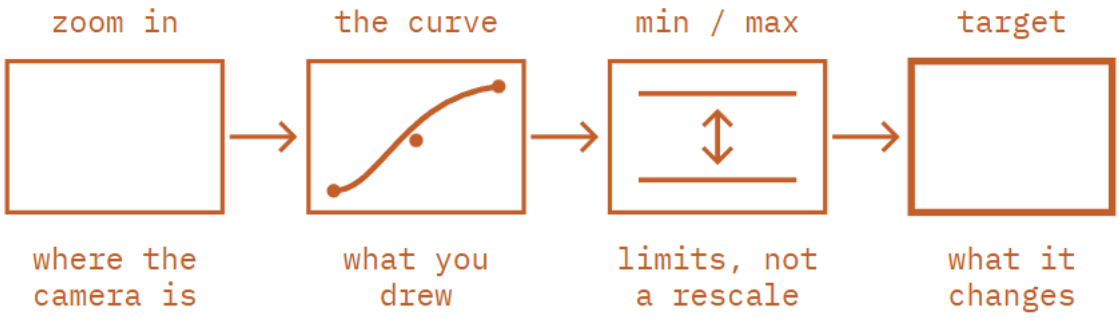
GeoExpression watches the map's zoom level and turns it into a number, following a curve you draw. That number then drives something else in the scene — a line that thins as you close in, a layer that fades up only over a city, a label that counts.

What it does

A map that looks right at country level rarely looks right at street level. Borders drawn thick enough to read across a continent turn into slabs over a city. Labels sized for a close-up disappear when you pull out. Detail that belongs on a regional map becomes clutter on a world one.

The fix is to make those values depend on the zoom rather than fixing them once. That is what GeoExpression is for: **zoom goes in, a number comes out**, and you decide the shape of the relationship between them by placing points on a curve.

If you have used zoom-dependent styling in a web mapping library, this will feel familiar. The difference is that the number does not have to style the map — it can drive a parameter on almost anything in your scene.

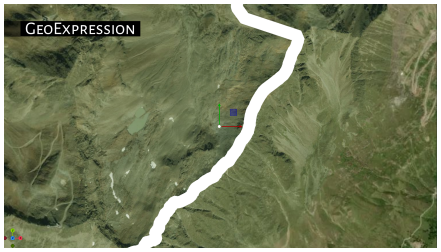


How to use the Plugin

As an example we are using a country outline (from a GeoJSON). The problem is the following:
If we have a very far zoom, the outlines linewidth is hardly visible, on a very close zoom, it will cover mostly too much of our shot.

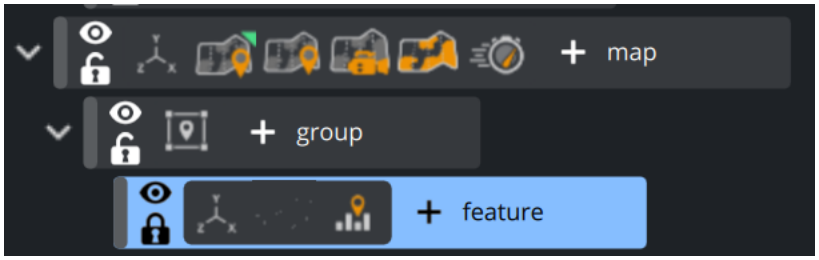


Copyright © 2026 Vizrt



Put GeoExpression on the geometry

Drag the **GeoExpression** plugin onto the container carrying the GeoJSON geometry — the same container that has ControlGeoJsonGeometry on it. Not the map container: the expression belongs with the thing it changes.



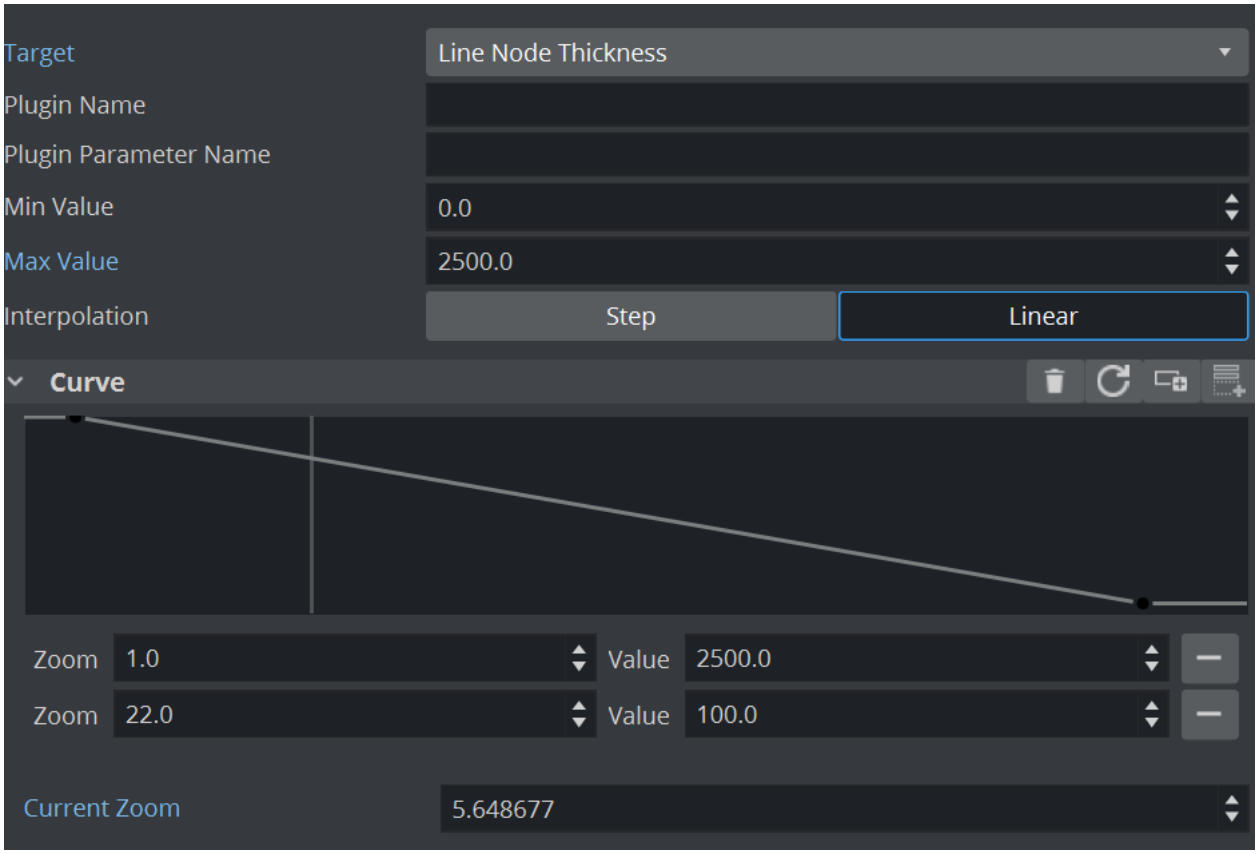
Tell it what to drive

Set **Target** to **Line Thickness**. The plugin now knows its output is a line width, and will write it into the geometry and also all Child containers underneath it.

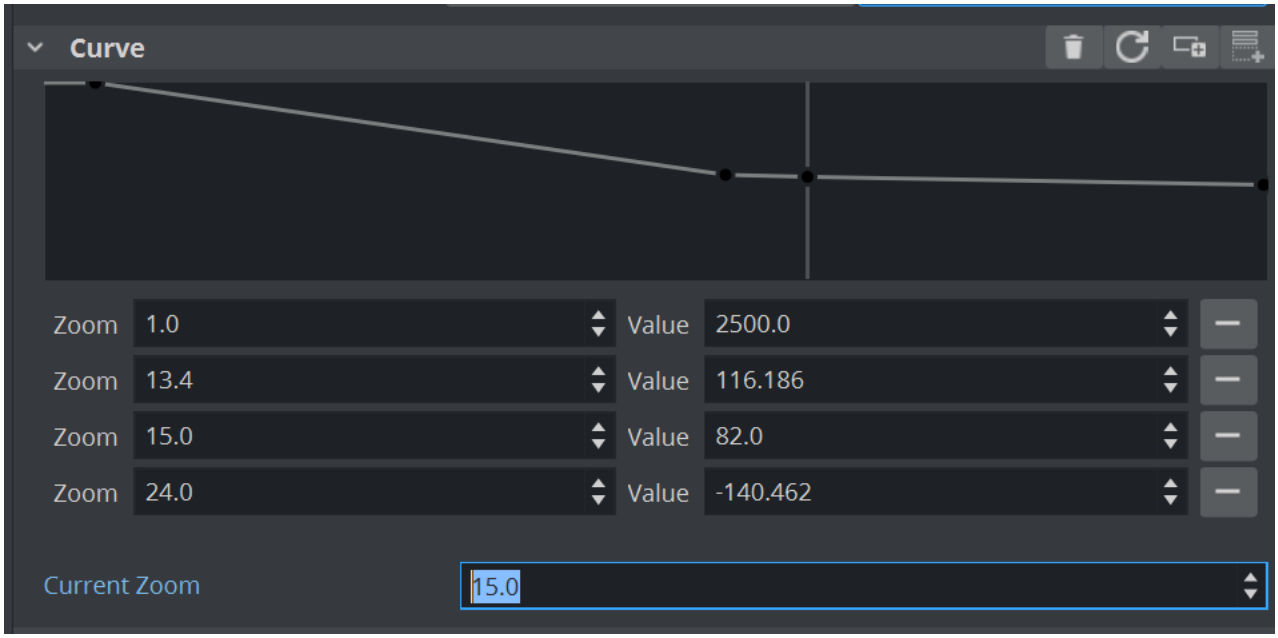
Adjust the values

Now zoom out. Our country border disappears. Put **Max Value** at *2500*. Line thickness here is a real distance on the ground, so the numbers are large — a border metres wide is invisible on a map of Europe.
This is a limit, not a target. It stops the output going higher than 4000; it does not stretch your curve up to it.
The plugin shows you the current zoom level. With a maximum of 2500 the border should be visible now, even on a zoom level equal to Europe.

Copyright © 2026 Vizrt



Now we have to adjust the width if we are on a very close zoom level. Set a Min Value, e.g. *-2500* and adjust the values to match the width at a zoom level like e.g. 15.
You'll notice that this is not a linear curve, it is more a logarithmic one. Therefore, it might be necessary to use more points to adjust the best looking value.



Copyright © 2026 Vizrt

Place points relating zoom to thickness. This set works well for regional borders:

At zoom	Thickness	Reads as
0	3620	Whole world
6	725	A large country
8	238	A region
11.4	39.8	A city
15.5	6	A few streets

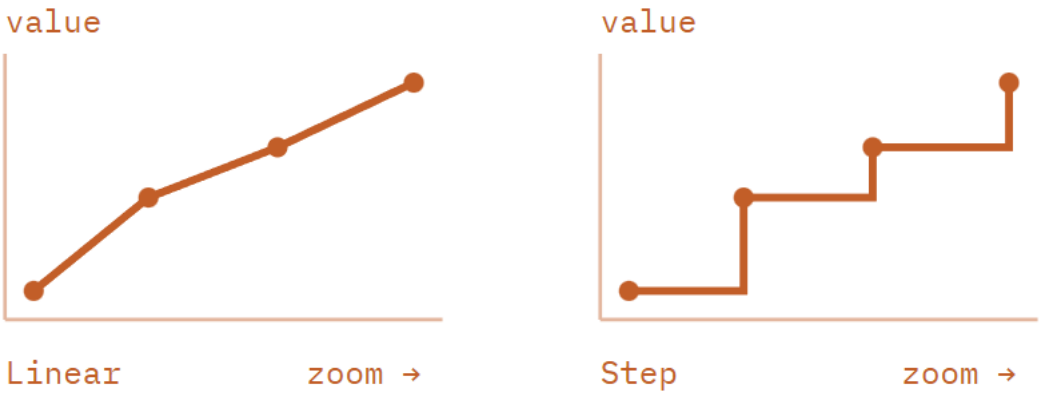
Notice how steeply it falls — roughly a third of the thickness for every two zoom levels. That is not arbitrary: as you zoom in, the same piece of ground covers far more of the screen, so the world-space width has to shrink hard just to keep the line looking constant.

Check it by zooming

Move the map camera in and out. The borders should stay visually similar the whole way rather than bloating or vanishing. To check a specific level without moving the camera, type it into **Current Zoom Level** and watch **Current Output Value** — that pair is the fastest way to see what your curve actually produces.

Drawing the curve

The curve is a list of points. Each one pins an **input** — a zoom level — to an **output** — the value you want there. Between two points the plugin fills in the gap, and how it fills it is the one choice you have.



- **Linear** blends smoothly between neighbouring points. Anything continuous — a width, an opacity, a size — wants this, because the change happens gradually as the operator zooms rather than snapping.

Copyright © 2026 Vizrt

- **Step** holds each point's value until the next point is reached, then jumps. Use it for decisions rather than quantities: a layer that is either on or off, a label set that appears wholesale past a threshold. Also worth a try when a linear fade looks mushy and a clean switch reads better on air.

Points are held in zoom order, and you need at least two for the plugin to have anything to interpolate between. Beyond your first and last point the value simply holds — a curve ending at zoom 15 keeps producing its zoom-15 value at zoom 20.

What it can drive

Alpha

The output drives the container's own colour and opacity, so the object it sits on can fade up and down with the zoom.

Reach for it when

- A layer should only appear once the operator is close enough for it to mean something — district shading over a city, but not over a continent.
- You want overlapping layers to hand over to each other rather than fight, one fading out as the other fades in across the same zoom range.

Line Thickness

The output becomes the line width of a GeoJSON geometry rendering in Line mode - see our example above.

Reach for it when

- Borders, roads, rivers or routes have to stay readable across a wide zoom range.
- Remember the width is a distance on the ground, not pixels. That is exactly why it needs an expression: a fixed ground width cannot look right at two different scales.

Plugin Parameter

The output is written into a named parameter of another plugin on the same container. Name the plugin and the parameter, and the value lands there every frame.

Reach for it when

- This is the general case, and much the most powerful of the three.
- A worked example: point it at a *Counter* plugin's *number* and the rendered text becomes a live readout that changes as the operator zooms.
- The same route drives text size, a shader input, a geometry parameter — anything a plugin exposes as a number.

Copyright © 2026 Vizrt

Script Parameter

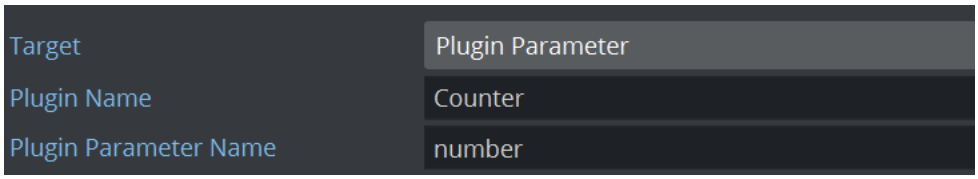
As with the Plugin Parameter, the value is passed over to a script. It needs to expose a parameter of `double` type.
`OnParameterChanged` will get notified once the value changes.

Reach for it when

- Whenever you are limited by Plugin Parameter option.
- To implement custom logic when needed

Using Plugin Parameter mode

If Alpha and LineThickness do not work, the GeoExpression Plugin can talk to any other plugins. For example, we write the normalized value into a Counter Plugin.
Create a subcontainer under your GeoCamera/GeoCanvas container. Add a Font, a Counter Plugin and the GeoExpression on it.



Click on initialize on the Counter plugin and the value should already be taken.

Using Script Parameter mode

If you need to implement your own logic or if you are limited by the other options, go for a simple script. For example you can have the zoom level driving the rotation of an “Elevator” indicator.

Create a script like

```
Sub OnInitParameters()
RegisterParameterDouble(“zoom”,”Zoom”,0,0,360)
end sub
sub OnParameterChanged(parameterName as string)
this.rotation.z = GetParameterDouble(“zoom”)
end sub
```

Setup the GeoExpression plugin to use script mode, pass 'zoom' as the parameter name.

> You have to set values for the Zoom level. If there is no “translation curve”, it can’t send any value

In our case make Zoom level 0 → 0, Level 24 → 360.
This will now apply a rotation to a container, based on the zoom level of our map.

Copyright © 2026 Vizrt

Tips and troubleshooting

Values look capped - Min and Max are limits.
They clip the curve, they do not scale it. Raising Max Value will not make a curve that tops out at 100 produce more — edit the curve instead.

Nothing changes as you zoom - Check the two live fields.
Type a zoom into Current Zoom Level and read Current Output Value. If the output moves, the curve is fine and the problem is the target; if it does not, the problem is the curve.

Plugin target does nothing - Check the parameter's internal name.
Plugin Parameter wants the name the plugin uses internally, which often differs from the label in the editor. A wrong name fails quietly.

The change is too abrupt - Add a point, or switch to Linear.
Step jumps by design. On Linear, a sharp corner usually means two points sit close together in zoom but far apart in value.

Getting the numbers right - Sample, do not guess.
Set the camera where the shot will be, note the zoom, tune the value until it looks right, and add that as a point. Three or four sampled points beat a curve derived on paper.

Beyond the ends - The value holds flat.
Past your last point the output stops changing. If a shot beyond that range looks wrong, add a point out there rather than reshaping the middle.

Copyright © 2026 Vizrt

6.6 Template based workflow

For a final template-based workflow, we need to add more options to our scene as one might be able to place labels in there, highlight certain areas, add symbols and many more. Therefore, a designer needs to prepare for all features to be added to a map.
This chapter explains a few of these scenarios.

Copyright © 2026 Vizrt

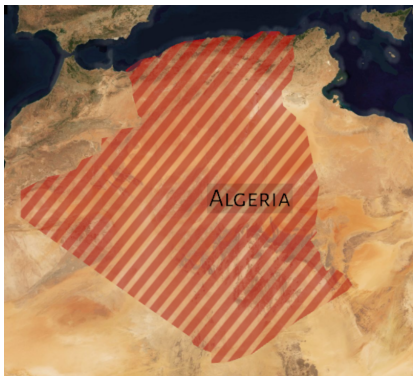
6.6.1 Adding Labeldesigns

Labels are one of the most essential elements on a map. They can either be geographically bound, e.g. to label a city capital, a country etc, or if you want to label a certain position, the user is free to change latitude and longitude, e.g. to mark the location of an earthquake.

- Geo-bound Labels
  - Example:
- Animated Labels

Geo-bound Labels

A geo-bound label has its latitude and longitude automatically assigned by the control application, like Viz Pilot edge. The journalist searches for a new point on the chosen map, everything else is done automatically.
A designer usually would like to create different styles for different kind of labels. This would be for example:

Type	Usage	Example
Country label	Usually a larger label used to indicate countries or county level.	
Major City label	Capital cities with a higher population than minor cities	

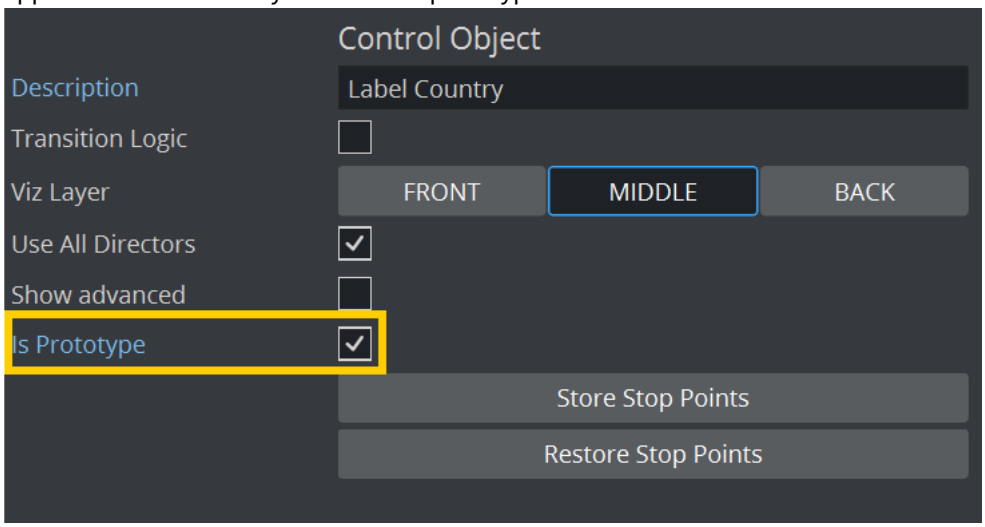
Copyright © 2026 Vizrt

Type	Usage	Example
Minor City label	Smaller cities with less population rather than a Major City.	
POI	Landmarks, Mountains, etc…	

As all of these are not bound to a fixed number of items on a map, we will use the ControlProtolist Plugin. This is an extension to the existing ControlList plugin.
This plugin allows to add an (un)limited amount of elements to be added from a given set of prototypes.

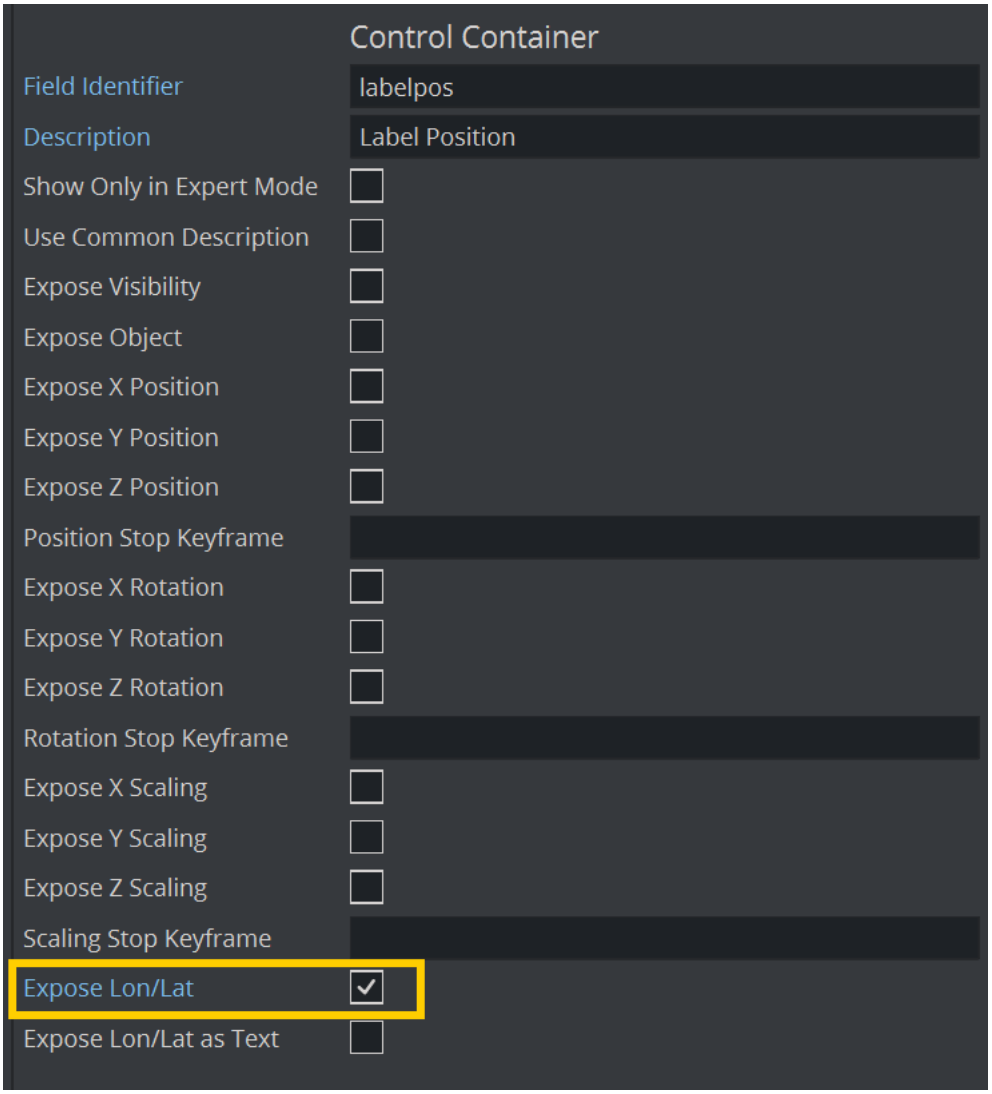
Example:

- Create a basic map based with GeoCamera and GeoCanvas
- Create a container called *prototypeslabels* inside your scene tree. The location does not matter, but as the labels will be georeferenced it would make sense to place it under your map.
- Next we create a sub container for “Country” style labels. Add a **ControlObject** plugin on it and check the “Is Prototype”. This will make sure it will not show up as an additional control object inside the control application as this is only be used as a prototype.

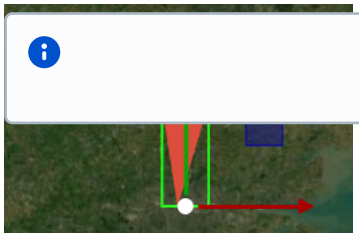


- Create your design. For example add a larger text, a TextBG for a backplate. This container must be in **GeoTransformation** Mode, otherwise it can not be placed based on latitude/longitude information coming from the control application.
  Add a **ControlContainer** plugin to your design, this will be the responsible container for placing the label correctly. make sure the Expose Lon/Lat is switched on, otherwise your label will not be located on the correct position.

Copyright © 2026 Vizrt



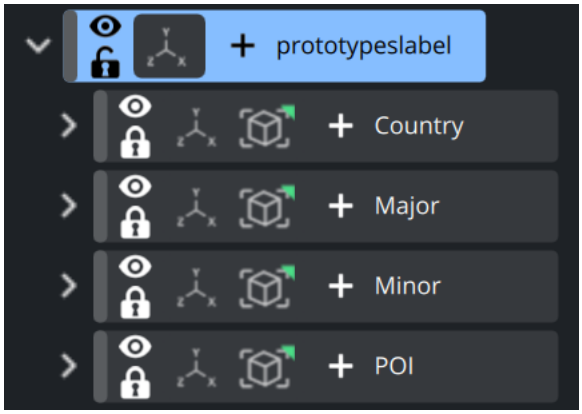
This will be your anchor point. If your style has some location pointer, make sure this one matches the containers origin.



Right now there is no automatic label layout. Prepare some styles to allow labels to set on top, bottom, left or right side.

- Continue to create more design you'd like to be added to your map.

Copyright © 2026 Vizrt

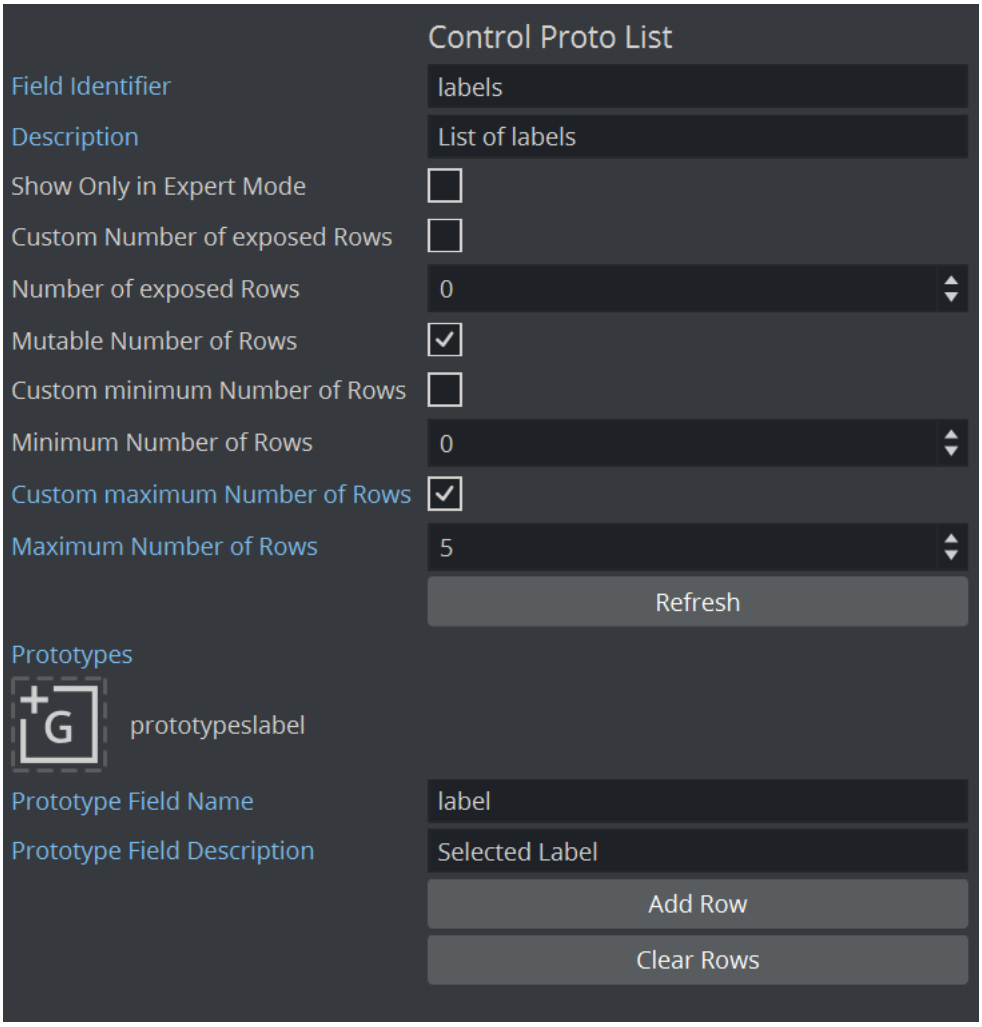


While designing your labels, make sure they fit on all zoom levels. See Start Using Maps#Scale-with-Zoom

Once your labels are in place, we need to expose them to the client application as prototypes. This will allow a journalist to add as many labels in one of our 4 designs to the map:

- Add a new container under your base map, name it e.g. "*listlabels*"
- Put the **Control Proto List** plugin on it:

Copyright © 2026 Vizrt



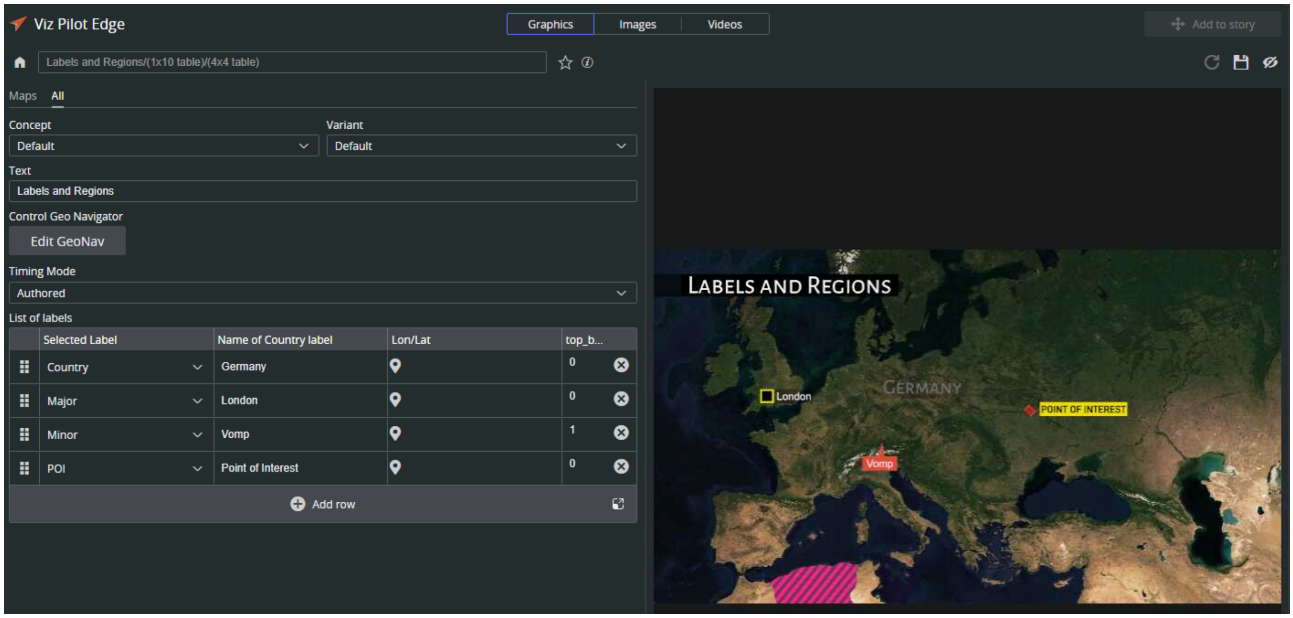
Drag your container holding our designs to the Prototypes field, give it a field identifier and hit the Refresh button.

> Your existing prototypes container will turn invisible, this is on purpose.

You can limit the maximum amount of labels to be added by enabling the “Control maximum number of Rows” and give it a number (see example on the left).

A control application will then read the scenes VDF model and allows a user to add these labels to the scene and give the journalist the option to choose between our 5 styles.

Copyright © 2026 Vizrt



Animated Labels

If you add labels to your maps design, you'll usually want them to animate in and out. This should depend on which location you are flying, for example you want to have "Sicily" labeled permanently while you are zooming in, the Capital City Catania should fade in and while you are continue your flight toward e.g. the volcano Mount Ethna, the city label should animate out again and Mount Etna label should appear.

As all labels are handled by **ControlProtoList** plugin we need to add additional plugins to also control the animation behavior.

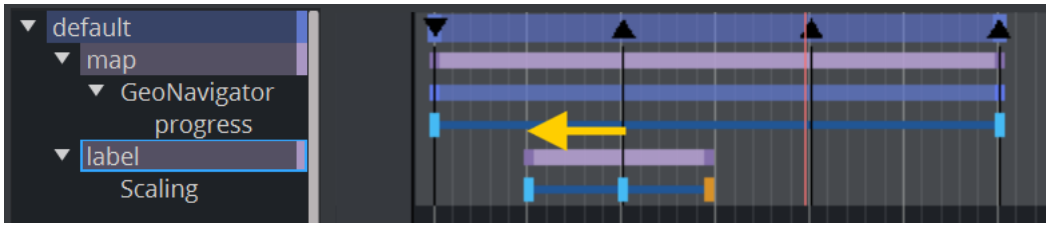


GeoHop and **ControlGeohop** are used to control the animation of labels based on a certain hop.
The GeoHop plugin copies the animation of our prototypes container and matches it to the stage animation.

GeoHop offers two options:

- **Hop number** is the hop the animation is bound to.
  For example if your first flight destination is "Catania", the label for "Catania" is bound to Hop 1. This is exposed to the client control application.
- The second one is the **offset** to the animation:
  If your label design has an in and an out animation, 1 second to build up and one second to build down, you will add an offset of -1 second.
  This guarantees that within the stage, our label animation starts 1 second earlier to build up, synchronized hop animation and then, as soon as our flight continues, animates one second out.

Copyright © 2026 Vizrt



GeoHop works in combination with the GeoNavigator plugin.
If you change anything on your label designs, make sure you click on the Refresh button on the ControlProtoList button and you rebuild the animation in the GeoNavigator plugin



GeoHop simply drives the animation you build for your label. If you want to have a label to stay on the map and not to fade out once the animation continues, you'll need to create a prototype label with no out animation.

See also:

ControlProtolist Plugin

Copyright © 2026 Vizrt

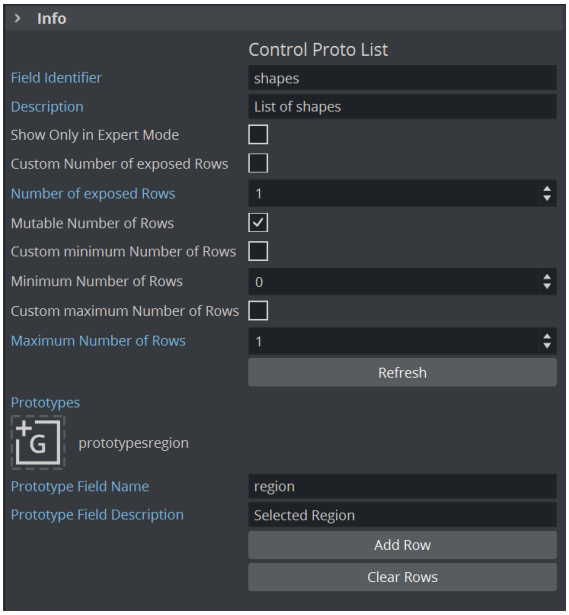
6.6.2 Adding Shapedesigns

Highlighting countries, regions, areas, streets, places, disputed regions - this is the main task when telling a story and using maps.
Therefore we will need to prepare our scene to allow a journalist to use GeoJSON data in various styles.

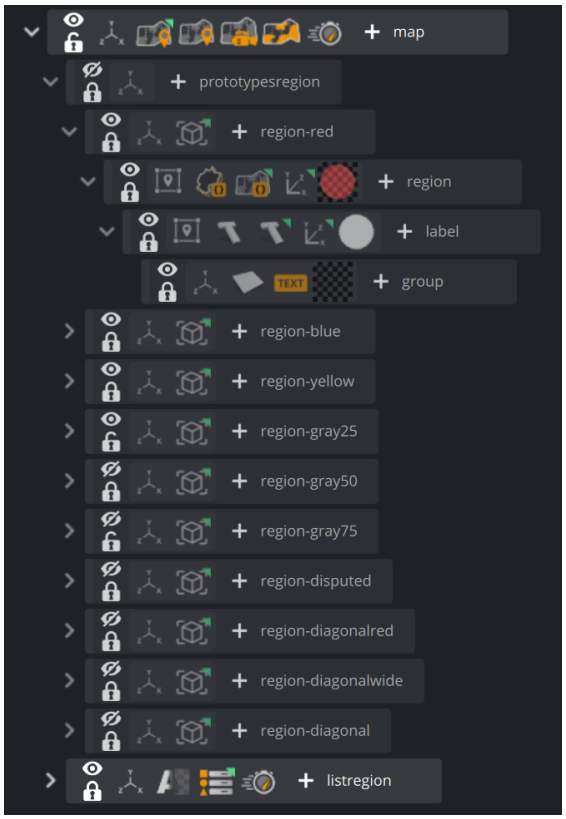
We will mainly use the same concept as we did for adding label designs - using Prototypes and ControlProtoList plugin.

Use your previous scene created in the Label design chapter or start from scratch.

- Create a basic scene with GeoCanvas and GeoCamera.
- Add a prototypes container.
- Now we need to create designs for the shapes:
  - For each design, create a new container, e.g. region-red, region-blue, region-yellow
  - Add **ControlObject** plugins to the containers and check the “Is Prototype”.
    This will make sure it will not show up as an additional control object inside the control application as this is only be used as a prototype.
  - Drag the GeoJSON plugin on it and the ControlGeoJson plugin
  - 
    This allows the GeoJSON shape data to be controlled from a control application.
  - If you also want a label for your selected region, create a Subcontainer with Text, and **ControlText** plugin. Also add a **ControlContainer** to it and expose **Lon/Lat.**
  - Make sure all containers holding a GeoJson or Label are on GeoTransformation mode.
  - As a last step, we need to create a container, responsible to hold all regions on our map. This one needs to have the **ControlProtoList** plugin on it. Point it to your shapes prototypes and hit the Refresh button.



Copyright © 2026 Vizrt



Save your scene and import it into your control application, e.g. Pilot Edge with Template Builder. The control application now exposes a list of configurable shape data, usually filled from the maps backend.

Selected Region	GeoJSON Source	Render Mode	Line ...	Label for shape	Lon/Lat
region-yellow		Fill	1	Ukraine	
region-red		Fill	1	Russia	
region-disputed		Fill	1		

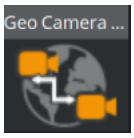
Copyright © 2026 Vizrt



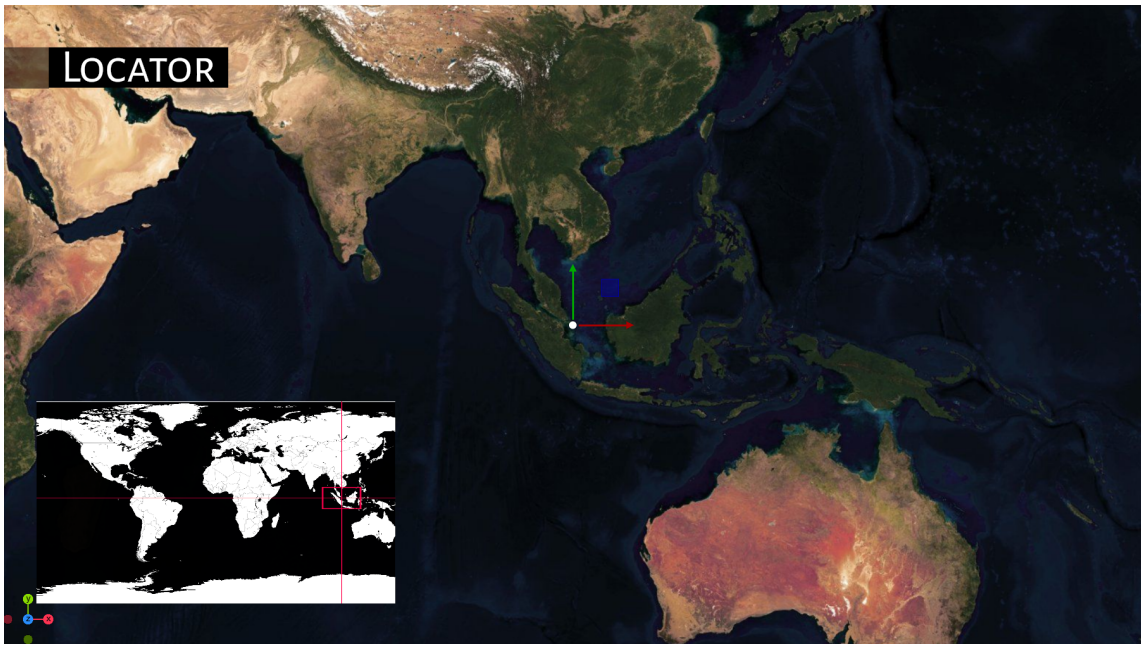
Shapedata can also be pasted manually and must not represent a country shape. It can also be a track of e.g. Tour de France or as in the sample above disputed regions.

Copyright © 2026 Vizrt

6.6.3 Adding a Locator



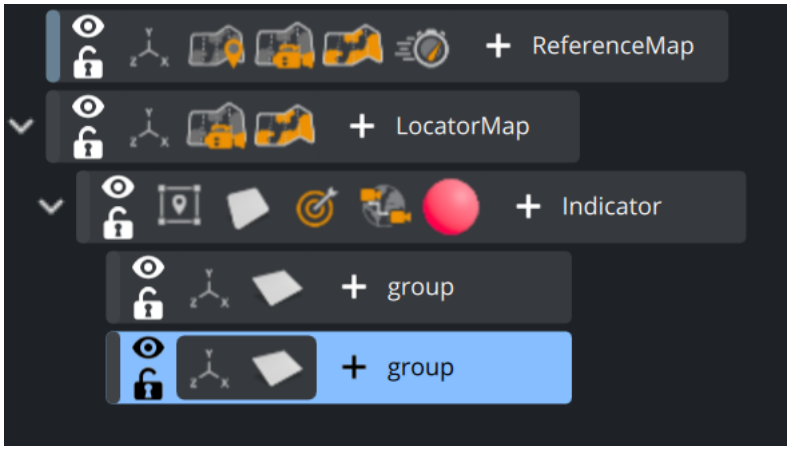
Many map designs come with a small locator to indicate where the story happens.
To transfer the geographical information from one Map system to another container in GeoTransformation mode, we will use the **GeoCameraLink**



- We need to build a scene with two map containers, both with GeoCanvas and GeoCamera on it. One as Reference, one for the Locator Map.
- Resize the Locator Map
- Add a sub container in GeoTransformation mode to the Locator map.
- Create some graphics indicating this position. e.g. a circle or a Noggi as outline.
- Now add the **CameraLink** plugin to it.
- Drag the Reference map container as the new Source Container inside the CameraLink plugin.

Any change on the GeoCamera will now be applied correctly to the smaller locator container.

Copyright © 2026 Vizrt



As a map can automatically act as a MaskSource, we can easily cut off the indicator to render outside the map by applying the MaskTarget to it. This prevents the two lines in our example to be “off”-map.

The plugin takes care of different projections. You’ll notice that when moving towards the poles. This is because of the fact, that you cannot project a globe correctly on a flat surface.

Copyright © 2026 Vizrt

6.6.4 Controlling the flight

How to turn a map flight into a template — so a journalist can change where the camera goes, minutes before air, without opening the scene.



A GeoNavigator flight is a list of places and the movement between them. On its own that list lives in the scene, and changing it means opening Artist. **ControlGeoNavigator** is what lifts that list out into the template, so the destinations become something an operator fills in from Trio or Pilot like any other field.

The plugin describes itself as exposing *a virtual list model for GeoNavigator camera pose fields*. In plainer terms: the operator gets a list of **hops**. Each hop is one stop on the journey — where the camera goes, and optionally how it travels there.

> ***The division of labour***
> *GeoNavigator does the flying and holds the defaults. ControlGeoNavigator decides which parts of that an operator is allowed to change. You design the movement once; they supply the geography on the day.*

What one hop contains

Every row in the operator's list is a camera pose plus the flight settings for getting to it. Only the fields you switch on appear — the rest keep the defaults you set on the GeoNavigator.

One hop

`bounds` Where the camera ends up. Held as a geographic area rather than a point, so the hop frames a region rather than balancing on a coordinate. This is the only field the operator always needs.

`bearing` Which way is up on arrival. Rotate to follow a coastline or a border.

`pitch` How far the camera tilts from straight-down into perspective.

`zoomLift` How far the camera pulls back mid-flight before descending again — the arc of the journey.

`duration` How long the flight to this hop takes.

`timingMode` Whether that duration is used as written, or worked out from the distance.

`motion` The shape of the movement — steady, eased, or a true map flight.

`flightDirection` Which way round the globe to travel.

`stopType` Whether the flight halts here and waits for the next take, or pauses and carries on by itself.

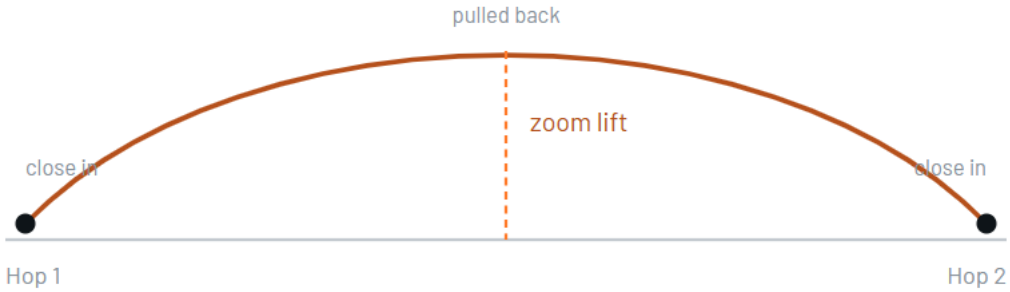
`pauseDuration` How long that automatic pause lasts.

`userdata` Free text carried on the hop. Useful for a caption, a lower-third cue, or the name of the place in your own words.

Copyright © 2026 Vizrt

The shape of a flight

Two settings do most of the work in making a flight look deliberate rather than mechanical: how high the camera rises between stops, and how it accelerates.



Motion

Motion	Use it when
`Linear`	You want a constant, mechanical rate — usually because something else in the graphic is setting the rhythm.
`Sine`	Short hops. It eases in and out, which stops the move looking like it was cut from a longer one.
`Van Wijk`	Long flights. It pulls back and pushes in automatically so the journey stays readable instead of skimming the ground at impossible speed. The default choice for anything continental.
`Great Circle`	The route itself is the story — a flight path, a shipping lane, an evacuation corridor. Follows the true shortest path over the globe.

Direction, and why it has three options

`Shortest` picks the shorter way round and is right nearly always. `East` and `West` exist for flights near the date line, where the shortest path may cross the Pacific when your story is going the other way. If a flight between two Pacific locations swings the wrong way across the map, this is the setting that fixes it.

Copyright © 2026 Vizrt

Stopping and pausing

A hop set to `Stop` halts the flight and waits — the operator takes the next stop when the presenter is ready. A hop set to `Pause` waits for the pause duration and then continues on its own. Stop is for live discussion; Pause is for a self-running sequence.

Choosing what the operator can change

This is the real design work. Each switch adds a column to the operator's list — and each column is one more thing that can be got wrong at ten to air.

Switch	Gives the operator
`Bearing`	Control of the compass rotation at each stop.
`Pitch`	Control of the camera tilt at each stop.
`Zoom Lift`	The height of the arc into each stop.
`Duration`	How long each flight takes. Often worth keeping — timings change with the running order.
`Timing Mode`	The choice between an authored duration and one derived from distance.
`Motion`	The movement type per hop.
`Flight Direction`	Which way round the globe. Worth exposing only for stories that go near the date line.
`Stop Type`	Whether each hop halts or pauses. Genuinely useful — it is the difference between a live-driven and a self-running sequence, and that changes per story.
`Pause Duration`	How long an automatic pause lasts.
`User Data`	The free-text field on each hop.
`Bounds as Text`	The destination as an editable text value rather than a picked area. The one switch that starts off — see below.

Bounds as text

Off by default, and that is usually right: an operator picking an area is harder to get wrong than one typing coordinates. Turn it on when the destinations arrive *from another system* — a newsroom feed, a results database, a script that writes the hop list — because then the value needs to be settable as text rather than chosen by hand.

> **A good template is a narrow one**
> Everything is exposed by default, which is convenient for testing and wrong for air. Start by switching most of them *off*. An operator with destinations, durations and stop types — three columns — works faster and makes fewer mistakes than one facing eleven. The movement style is a design decision you already made; it does not need re-deciding under pressure.

Building the template

1. Get the map working first
   A GeoCanvas for the basemap and a GeoCamera on the same container. Add the camera yourself — creating the canvas does not create one. Fly around by hand and confirm the map looks right before any of the flight machinery goes in.
2. Add GeoNavigator and set the defaults
   On the map container. Set motion, zoom lift, duration and timing mode to what a typical flight in this template should look like. These are the values every hop inherits, so spend time here — it is what stops the operator having to make design decisions.
3. Build a real flight and watch it
   Capture two or three destinations with the camera, press **Update Animation**, and play it. You are judging the defaults, not the destinations. If a flight only looks right at one particular pair of locations, the defaults are wrong.
4. Add ControlGeoNavigator to the same container
   It finds the GeoNavigator itself. Give it a **field identifier** and a **description**: the identifier is how the template addresses the list, the description is the wording the operator reads. Write the description as an instruction — “Places to fly to, in order” beats “GeoNavigator control”.
5. Switch off what they do not need
   Go through the expose list and turn off everything the story does not require. This is the step that decides whether the template is pleasant or overwhelming.
6. Drive it from the client, not from Artist
   Open it in Trio or Pilot as the operator will. Add a hop, reorder the list, delete one from the middle. Anything awkward here is worth fixing now, because on the day nobody will have time to work around it.
7. Prefetch, then save and reload
   Run the tile prefetch so the first playback is not also the first tile fetch. Then save the scene, reload it, and check the controls come back with the same fields and values — that is the state the newsroom actually meets.

Things that catch people out

- **The animation does not rebuild itself.** Changing hops or settings leaves the previously generated flight in place until **Update Animation** runs again. If a change seems to have done nothing, this is almost always why.
- **A ragged first run is tiles, not animation.** If the flight looks perfect on the second playback and stuttery on the first, the map was still loading. Prefetch before air.
- **Expose flags are on by default.** A template shipped without going through them gives the operator every field there is. Treat the defaults as a starting point for testing, not for delivery.

Copyright © 2026 Vizrt

- **Exposing a field does not change the flight.** It only decides who can edit it. A hop with nothing exposed still flies — using the GeoNavigator's defaults.
- **Expert mode hides, it does not disable.** Fields marked expert stay functional; they just stop cluttering the everyday view.

A worked example

An election night template: the map opens on the whole country, flies to whichever state the desk wants to talk about, then drops into a single county.

Hop	Destination	Flight in	Ends
1	National	—	Stop
2	State — operator picks	Van Wijk, 3.0s, moderate lift	Stop
3	County — operator picks	Sine, 2.0s, little lift	Stop

Exposed: **destination**, **duration**, **stop type**. Everything else — motion, lift, bearing, pitch, direction — stays with the designer. The desk can go anywhere in the country and re-cut the timing live, and cannot accidentally tilt the camera into the horizon while doing it.

Note the third hop uses Sine rather than Van Wijk: the state-to-county move is short, and Van Wijk's pull-back would put air between two places that are meant to feel adjacent.

Copyright © 2026 Vizrt

7 Manage Items and Plug-ins

To start Viz Artist, the user must log on to a Graphic Hub. The Graphic Hub is the database solution where all Viz Artist items are stored. Items can be either:

- Asset View
- Working with Items
- Background Loading
- Plug-ins Tab

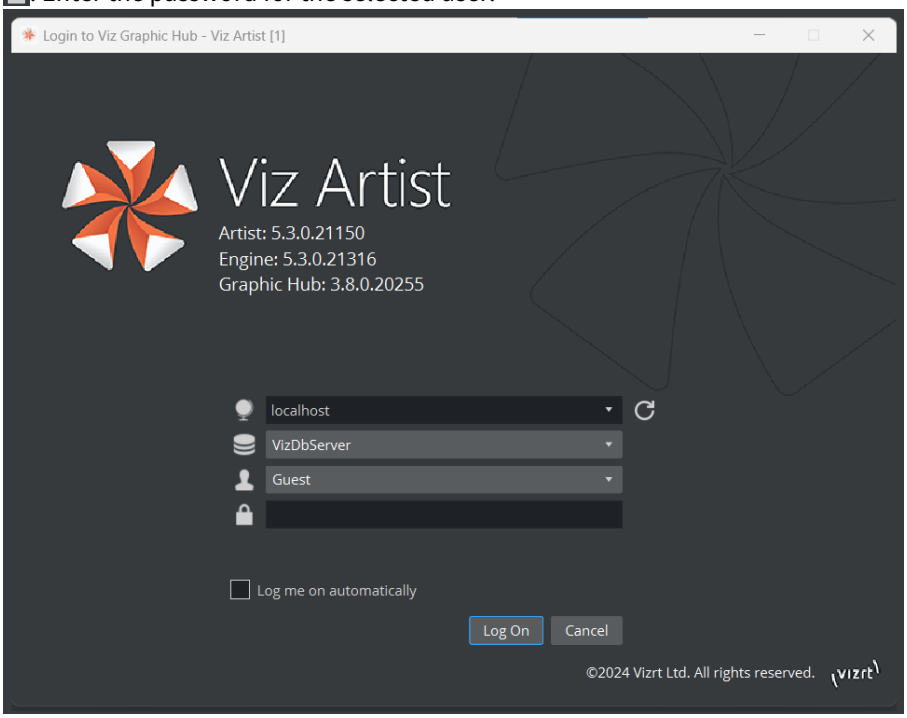
The Graphic Hub database can be installed on the local machine for a single-user environment, and it can also be installed on a centralized server for a multi-user environment, which provides shared access to multiple users simultaneously.

Individual items are stored in the Graphic Hub database as file objects and are managed in terms of properties and *Universally Unique Identifiers*, commonly referred to as *UUIDs*.

Clients can work with and organize items through Viz Artist, in a logical folder structure provided by Graphic Hub. Although each item can be listed and linked in more than one folder, it is in Graphic Hub only once.

7.1 Working with Graphic Hub

1. Start Viz Artist.
2. In the Graphic Hub log in window, enter the required details:
   - : Select a Host from the drop-down list.
   - : Select a Graphic Hub from the drop-down list. Choose the server you want to log in from the list of the auto discovered hostnames.
   - : Select a User from the drop-down list.
   - : Enter the password for the selected user.



- : Press the refresh button to rescan your network for available Graphic Hubs. If a server is started after Viz Artist has launched, click on the refresh button to start the auto discovery again. When working with a single server or if you always use the same server, you can enable the **Autologin** feature in the database settings inside the configuration settings.
- **Log me on automatically:** If automatic log on is enabled in Viz Configuration, checking this check box activates it. When active, Viz Artist logs on to the selected Graphic Hub using the provided credentials on start.

Note: In some circumstances (for example, if the server is located on a different subnet) the host is not in the list. The host name must then be entered manually. In such cases, you can whitelist the server. See the Viz Engine Administrator Guide for details.

Info: Failover servers are shown in green, they should only be used as fallback if your main working server is not available any longer. Whitelisted servers appear in cyan.

Tip: If a Graphic Hub cannot be discovered automatically (if for example, located in another subnet), it can be whitelisted in the Graphic Hub configuration section (additional hostnames).

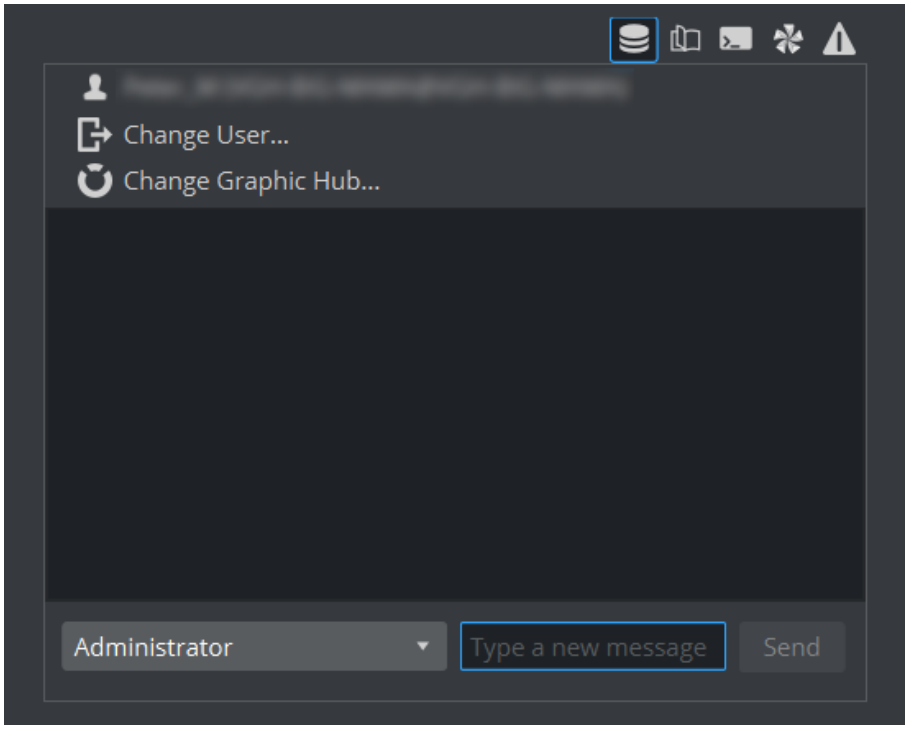
3. Click:
   - **Log on:** Log on to the selected Graphic Hub and start Viz Artist.
   - **Cancel:** Cancel log on and close the application. After logging on to a Graphic Hub, Viz Artist starts and opens the main screen.

The log on credentials are saved in the **Database** section of the Viz Configuration (see the **Configuring Viz** section of the Viz Engine Administrator Guide) where it can be modified as required.

7.1.1 To Switch Active Graphic Hub

Since Viz Artist version 5.3, you can switch the current Graphic Hub without restarting Viz Artist. Select **Change Graphic Hub...** from the top database icon.

Copyright © 2026 Vizrt



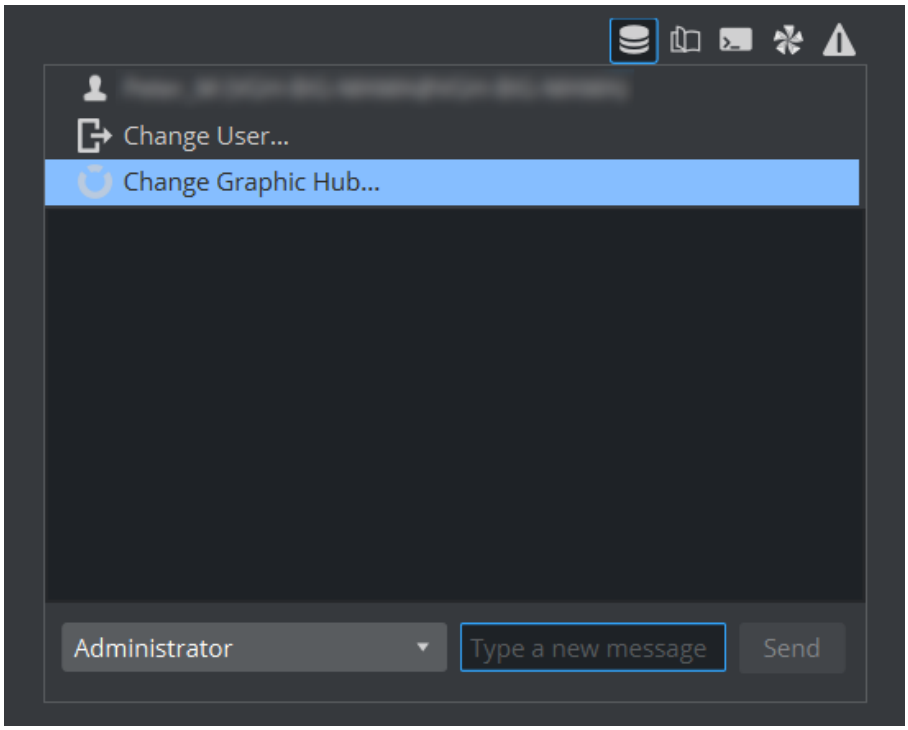
Any open asset on the current Graphic Hub needs to be closed first. If you want the new Graphic Hub to be your default one, click on **Set as default database**.

Switching Graphic Hub also switches the following:

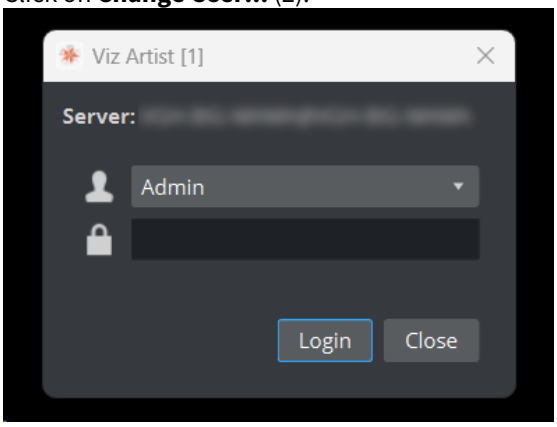
- Asset View, including Bookmarks, History and Saved Searches.
- Font Library Items.
- Post Render Templates.

Copyright © 2026 Vizrt

7.1.2 To Log on a New User after Viz Artist has Started



1. Click on the **Server Icon** on the upper right.
2. Click on **Change User...** (2).



3. In the **Change User** window, select a User from the drop down menu and enter the password in the **Password** field.
4. Click:
   - **Login:** To change User.
   - **Cancel:** To cancel the procedure.

Copyright © 2026 Vizrt

7.2 Asset View

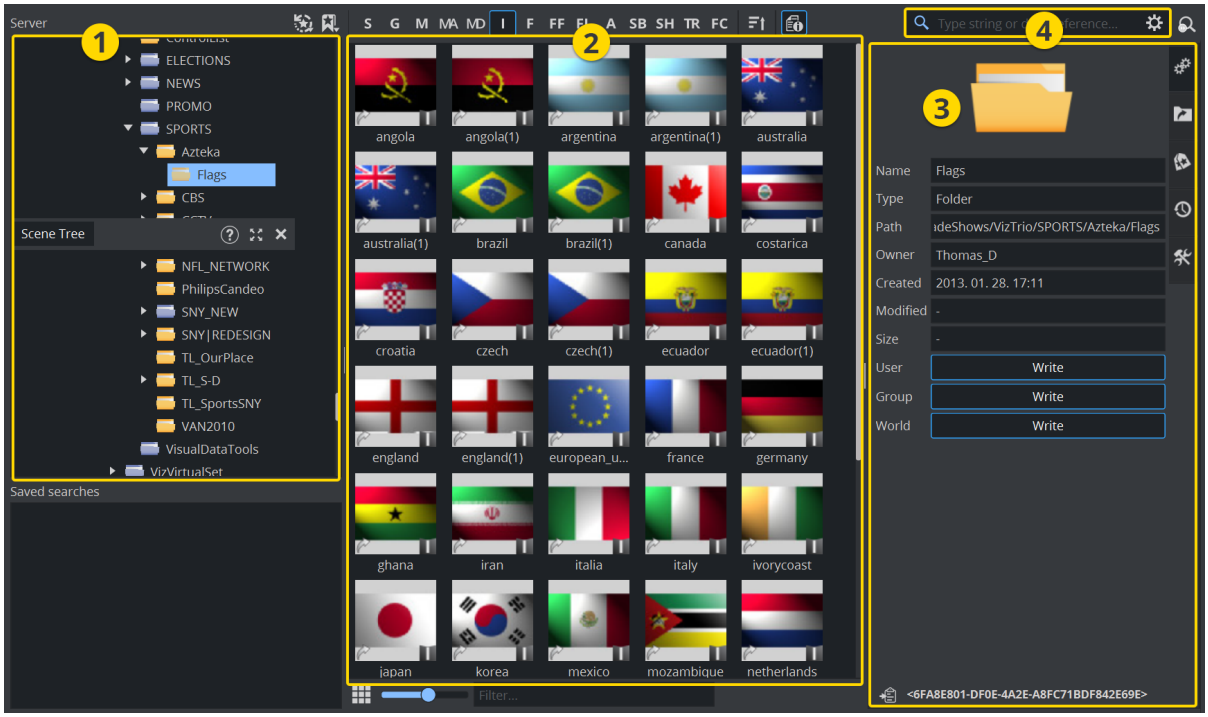
Viz Artist comes with a new Asset view replacing the old server and items view. It also merges the search functionality and freetext search option of previous versions into a unified search tool.

- Asset View Window
- Server Tree
    - Working with Project/Folders
    - Working with Bookmarks
    - History
- Items View
    - Sorting and Filtering Items
        - Natural Sorting vs Legacy Sorting
    - Content Switcher
    - Item Types
        - File Collections
    - Item States
        - File Operations on Items
            - To Rename an Item
            - To Lock an Item
            - Images
            - Fonts
            - Audio
        - To Select Multiple Items
        - Location History
        - Hot Folders
- Quick Look
    - To Quick Look Images
    - To Quick Look Material Definition types
- Properties
    - General
    - Links
    - Keywords
    - History
    - Advanced
- Search
    - Simple Search
    - Advanced Search
    - Reference Search
        - Reference Search on an Item
        - Advanced Reference Search
        - Troubleshooting Search
- Shortcuts

Copyright © 2026 Vizrt

7.2.1 Asset View Window

The Asset View window is split into the following parts:



- **Server Tree (1):** Manage files and projects on your Graphic Hub.
- **Items View (2):** Browse and view your available items.
- **Properties (3):** An even more detailed view on your assets and a set of tools.
- **Search (4):** A powerful search tool to find the items on your Graphic Hub.

7.2.2 Server Tree

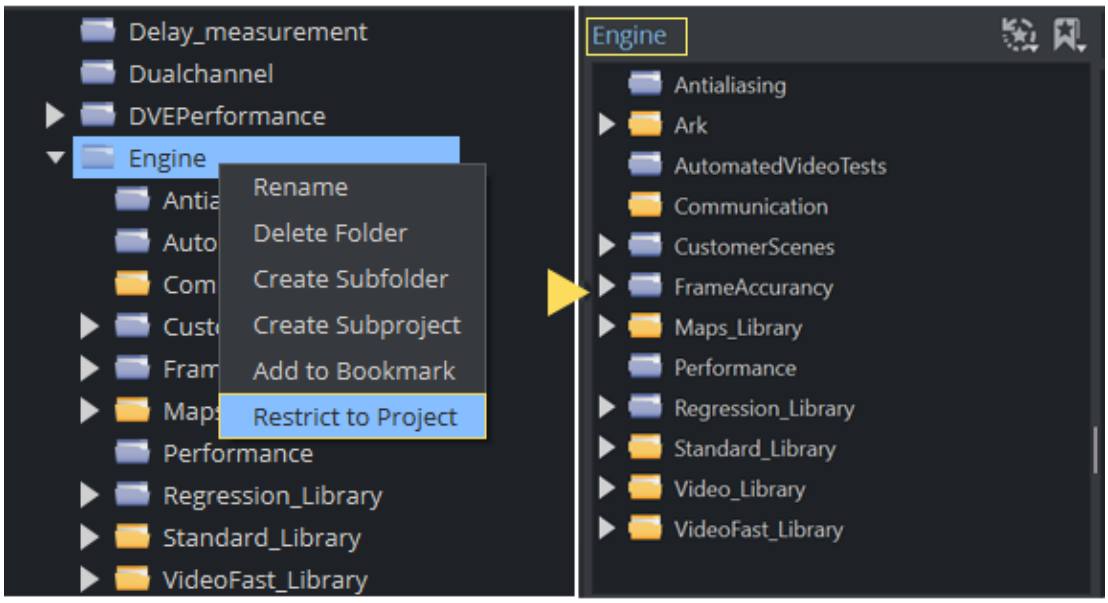
The Server Tree shows a structured representation of your Graphic Hub. It has the possibility to create projects and folders, so anyone can organize assets matching various needs.

There are two types of folders:

- Regular folders
- Project folders

The difference between a regular folder and a project folder is that by defining a working project, the tree can be collapsed so that only the currently selected project and its sub-folders and sub-projects folders are shown. A project can contain various folders, but a folder cannot contain projects. To restrict the view to your current project only, right click on your project folder and select **Restrict to Project.**

Copyright © 2026 Vizrt

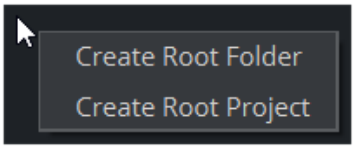


This allows to see only the relevant structure of your project, while search functionality and others are still available. The current select project is shown on the top of the server tree. To switch back to the original view, right click in the scene tree and select **Remove restriction** or click on the light blue colored project name on top of the tree (**Engine** in the sample above).

Working with Project/Folders

New folders or projects can be created by right clicking on an existing folder or project in the tree view and selecting **Create Subfolder / Create Subproject**. The new item is created underneath the selected folder/project. Sub-folders can be added to both regular folders and project folders.

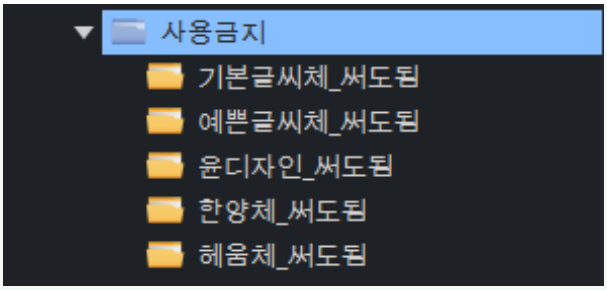
If you click on an empty area, you can choose **Create Root Folder** or **Create Root Project** to create new Folders/ Project on the root level.



Information: Only Project folders can contain Sub-Project folders.

To rename a folder or project, right click and select **Rename**. Be aware that not all characters are allowed as names, some reserved ones like *** { [**, spaces, etc... are not accepted.

Folders or projects are not restricted to Latin characters:



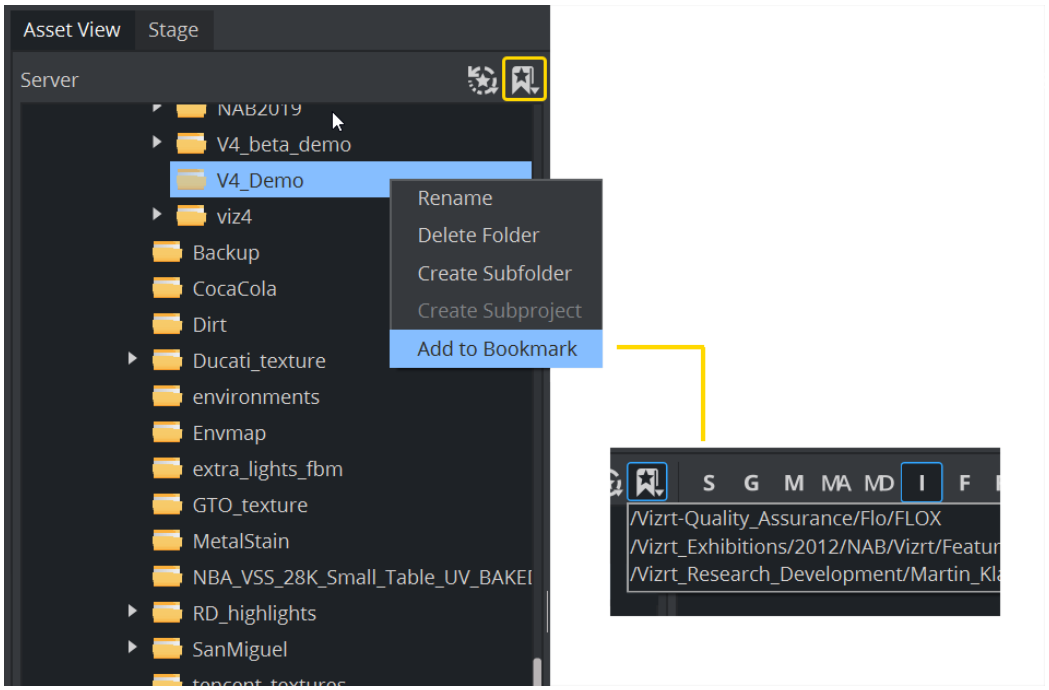
To reorder your structure, both types of folders can be dragged around after being added to the Server Tree. Only Project folders can contain Sub-Project folders; however, sub-folders can be added to both regular folders and project folders. To prevent folders/project being moved by an unwanted drag operation, this operation can be limited in the User Interface option in the Viz Config section:



This can be either unrestricted (**Normal**) or only by using the **SHIFT** key in combination with the drag operation (with **SHIFT**) or can be completely forbidden (**No**).

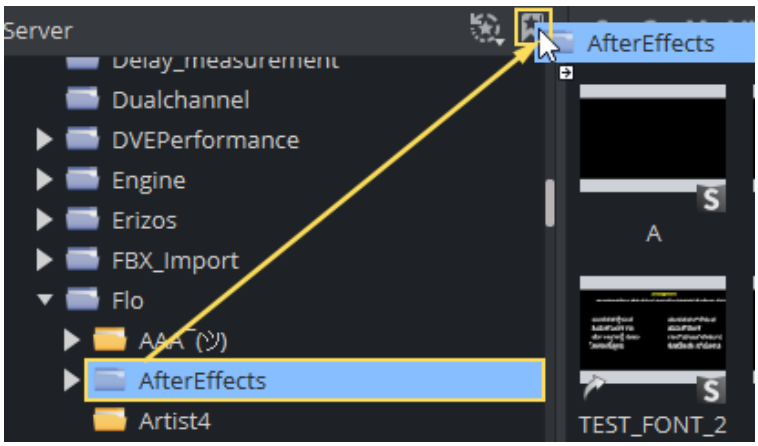
Working with Bookmarks

To provide quick access to the most important folders or projects, they can be bookmarked to speed up navigation. Bookmarked folders are added to the Server Tree Bookmark context menu. Right click on a folder or project and select **Add to bookmark**. This adds your favorite into the bookmark drop menu on top of the server view:



Copyright © 2026 Vizrt

Bookmarks can also be created by dragging a folder to the Bookmarks Icon.

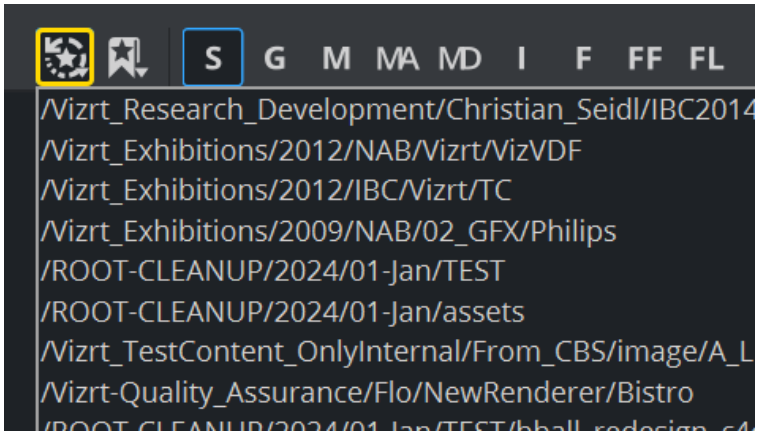


Bookmarks can be deleted by opening the bookmarks menu and clicking on the small trashcan next to the item. Bookmarks are sorted alphabetically by their full path.

Information: Bookmarks are stored on a per user level on Graphic Hub. If multiple users use the same login (for example, Guest) Bookmarks will be mixed up.

History

The asset view automatically adds up to 20 folders and project locations into a history once an asset has been used from within that folder. This applies to actions (like an item has been deleted or double-clicked, renamed, dragged into the scene tree and so on). Whenever an action happened within a folder or project, it is added to the history list. This history is available at the top of the server tree.

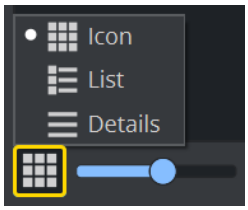


Information: History is stored on a per user level on Graphic Hub. If multiple users use the same login (for example, *Guest*) history entries are mixed up.

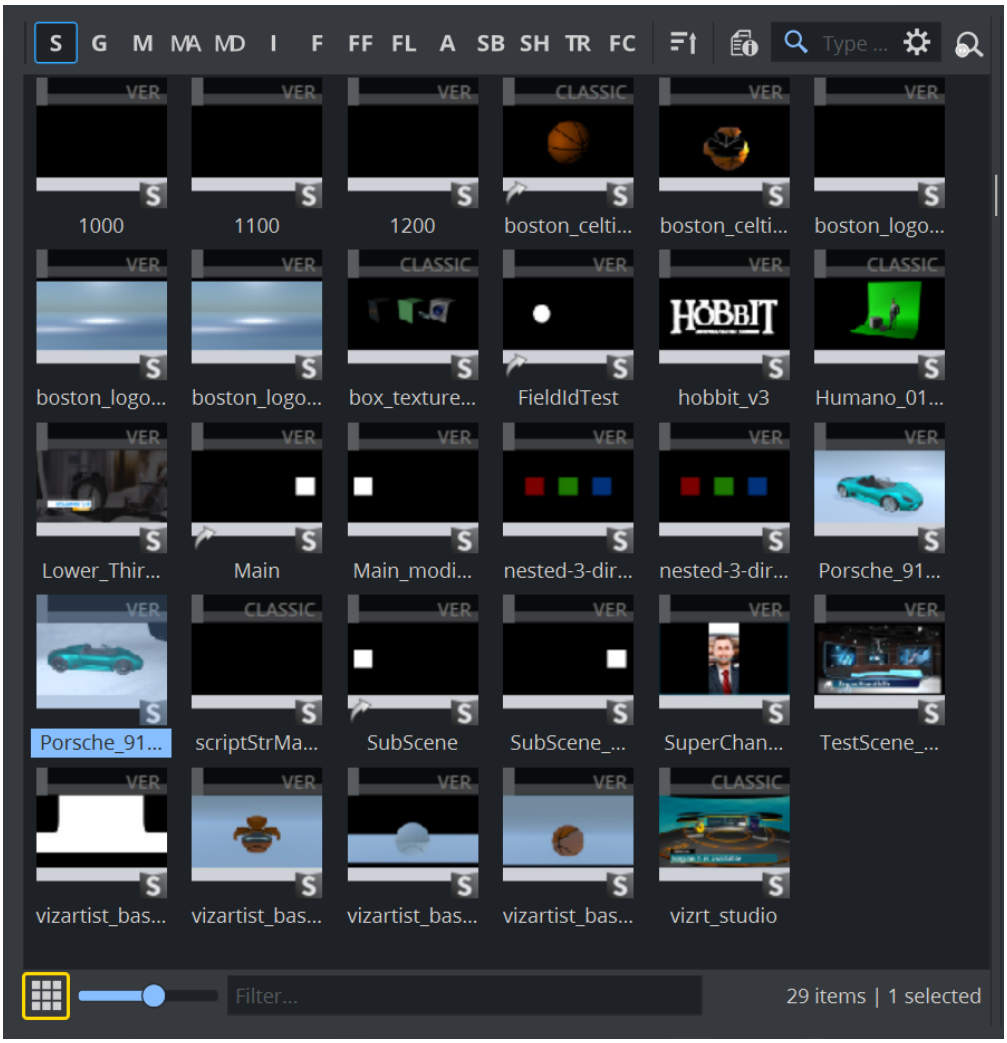
Copyright © 2026 Vizrt

7.2.3 Items View

The content of a selected folder or project is shown in the Items view. Items can be represented as icons, list or details view.



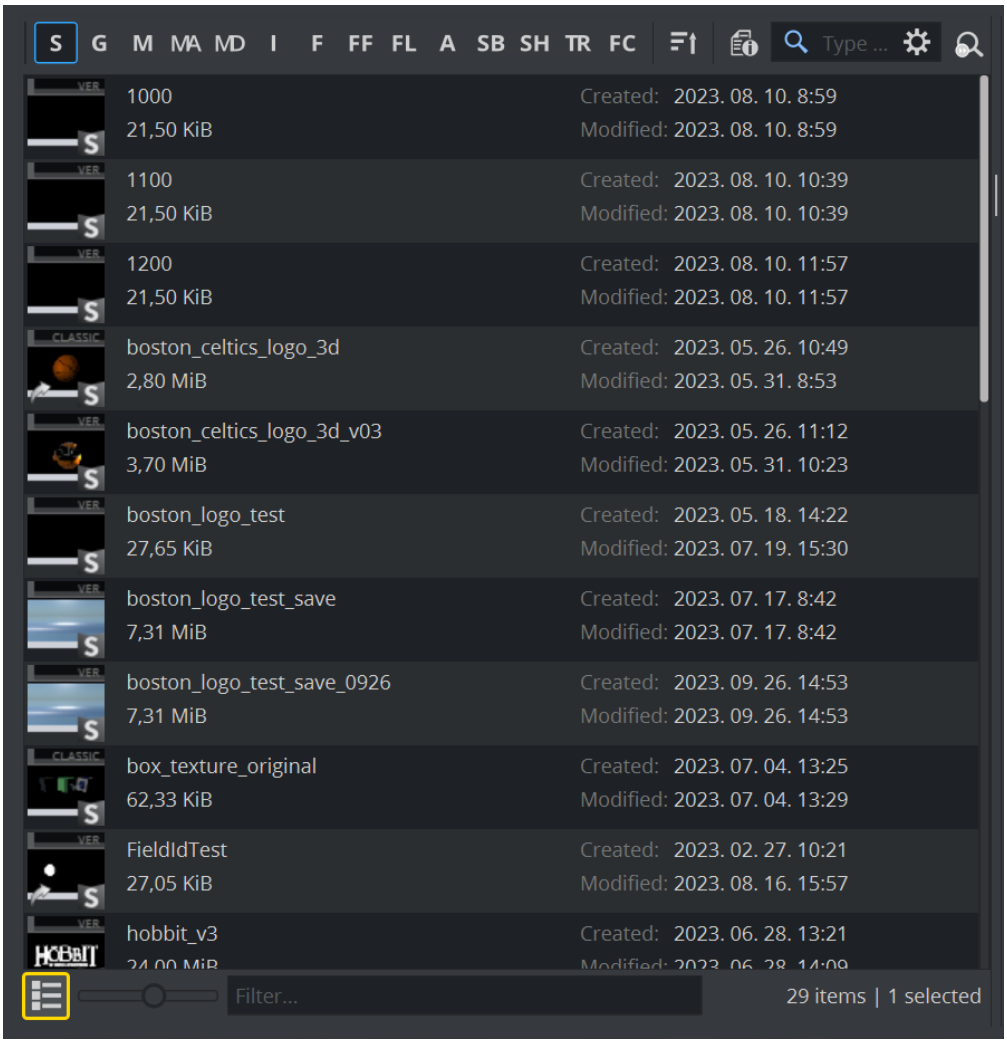
Icon View



Note: The size of the icons can be changed by using the slider, the configured keyboard shortcut (default **CTRL** + + and **CTRL** + -) or by using **CTRL** + Mousewheel.

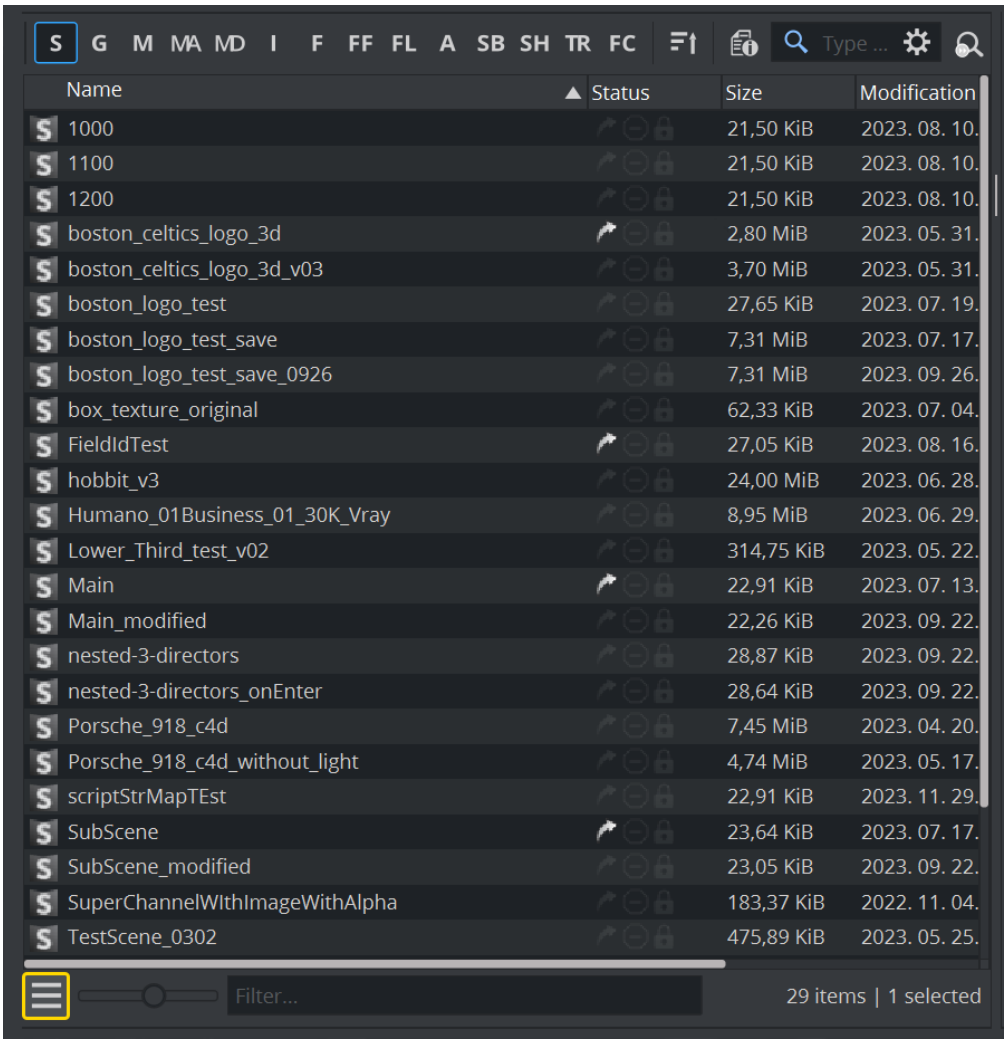
Copyright © 2026 Vizrt

List View



Copyright © 2026 Vizrt

List View with Details



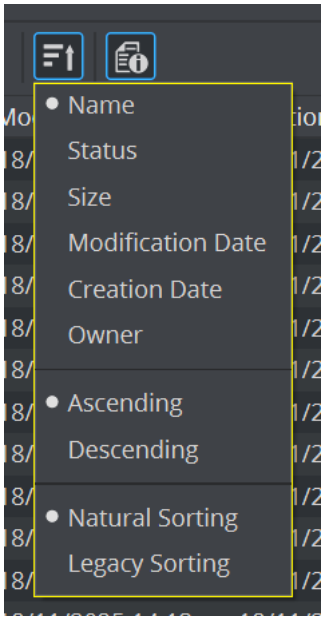
Information: Columns can be shown/hidden by right clicking on the column bar. Columns can also be resized and reordered. Clicking a column header sorts the list according to the selected column.

By using the slider on the bottom, you can choose the size of your icons in Icon view. Once you switch to list view, this option is disabled. Since Viz Artist version 4.4, the creation time is also shown.

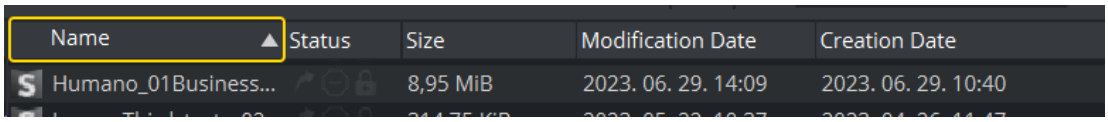
Sorting and Filtering Items

You can sort how items are being displayed by using the sort icon next to the Content switcher. Then you can sort by *Name*, *Modification Date*, *Creation Date*, *Size* and in both *Ascending* and *Descending* order.

Copyright © 2026 Vizrt



In the *Details* view the same can be achieved by clicking the corresponding column header. Clicking the header again changes the sorting order.

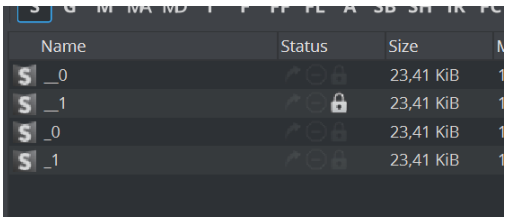
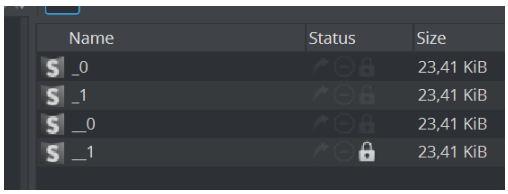


Underneath the Items view, there is an input field for doing a quick filtering on your items. The filter applies to any part of the filename. As an example, if the content switcher is limited to images and given that your items follow a certain naming convention, you set a filter to "normal", it only shows the normal maps within a folder:

The name and the UUID of any item are taken into account.

Natural Sorting vs Legacy Sorting

Viz Artist uses Unicode collation algorithm for sorting items within a folder. If you prefer the old behavior (like Graphic Hub Manager does), switch to legacy sorting. Legacy Sorting means that, for example, items starting with an underscore (_) are listed first.

Natural Sorting	Legacy Sorting
	

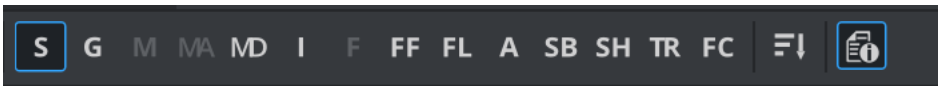
Copyright © 2026 Vizrt

Content Switcher

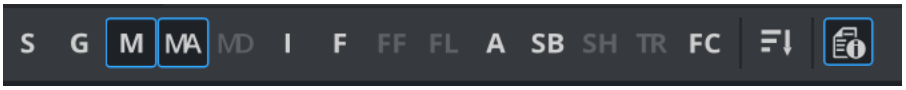
Folders can contain various types of items. To view only a selected type of items, contained in a folder, click an item type tab from the Content Switcher. This also allows multi-selection while holding the **CTRL** key while clicking on it. If you want to see all of your items, click on any previous selected item, so no one is selected. This replaces the old *All* tab.

Media Types which are currently not available in the selected Render Pipeline will be disabled.

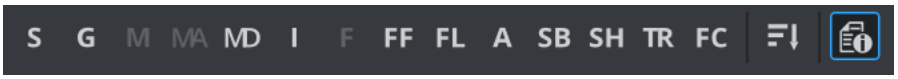
Content Switch to Show only Scenes in the Viz Engine Render Pipeline

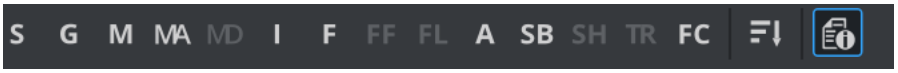


Content Switch to Show All Material Types in Classic Render Pipeline



Content Switch to Show All Types in Classic and Viz Engine Render Pipelines





Item Types

The various types are represented by a symbol in the content switcher as tabs. Currently all types are shown, independent from the selected render pipeline. This is expected to be improved in future versions.

Symbol	Meaning on Graphic Hub
S	Scenes
G	Geometries
I	Images
F	Fontstyles *(Classic Render Pipeline only)*
FF	Fontface *(Viz Engine Render Pipeline only)*

Copyright © 2026 Vizrt

Symbol	Meaning on Graphic Hub
FL	Fontfile *(Viz Engine Render Pipeline only)*
A	Audio
M	Materials *(Classic Render Pipeline only)*
MA	Materials Advanced *(Classic Render Pipeline only)*
MD	Material Definitions *(Viz Engine Render Pipeline only)*
SB	Substances
SH	Shader *(Classic Render Pipeline only)*
TR	Texture Renderer *(Viz Engine Render Pipeline only)*
FC	File Collections

File Collections

File Collections have been introduced in Viz Engine to allow a user to store data from external applications (like Unreal projects) in Graphic Hub. The File Collection holds the information about this data and is usually linked to amount of binary data. This binary data (`BINARY_BLOB`) is invisible to the Viz Artist user. Deleting File Collections from within Viz Artist always deletes all references too.

Item States

Items on Graphic Hub can have different states, while they are being worked on. Items that are currently in work are active items. Active items are locked or checked out.

Overlay Icon	Meaning
	Linked Item This item appears one more than one place on Graphic Hub, but is physically stored only once. The UUID of the one saved item is the same for all linked items, while information such as path and name may vary. If modifying the item in one of the folders, for example cropping an image, all the linked items reflect these changes. If this item is being deleted, only the reference counter is being decreased. To delete all references of linked items, choose "With all References" in the confirmation dialog. To create a linked item, press the **ALT** key and drag it to the destination location.

Copyright © 2026 Vizrt

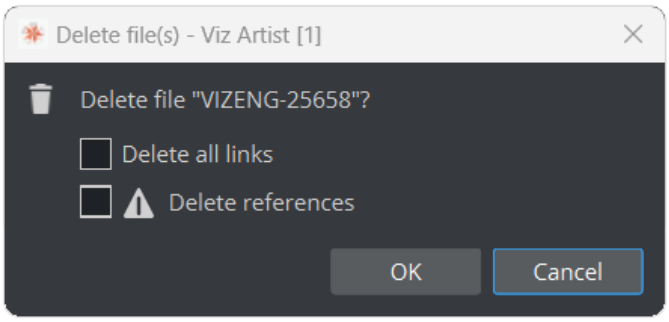
Overlay Icon	Meaning
	Locked Item If a user connected to the database opens an item, the item is locked for all other users and marked with the locker icon. A locked item means that the item can be opened and viewed by all users, but only modified or saved by the user that has locked it. The item is unlocked when the user closes the item or disconnects from the database.
	Checked out Item If a user connected to the database checks out an item, the item is locked for all other users and marked with the stop icon. A checked out item means that the item can be opened and viewed by all users, but only modified and saved by the user that has checked it out. To cancel the check out, the item must be checked in. Any item can be checked out within Viz Artist by using the context menu and selecting **Check out** or **Check in**. The item remains checked out (locked) even if the user closes the item or disconnects from the database.
	Locked and Checked out Item For a locked and checked out item, the icon represents a combined keyhole and stop sign. It can be opened read-only but not modified unless unlocked and checked in again.
N/A	**Expiring Item** Some items, for example images staged by the IMEX Agent can have an expiration date set. This is, however, not yet covered by Viz Artist Asset view.
	**Learners Edition** Items created with the Viz Artist for Learners Edition are shown with an additional overlay icon. These are shown on Scenes and on Geometries created with this special license.

File Operations on Items

Items can be copied, moved, renamed, deleted or linked.

- To copy an item, you simply drag the object to its new location. The copy operation is shown within the mouse cursor by a small **+** sign. As an alternative you can use **CTRL + C** and to paste **CTRL + V** within the new location.
- To move an item, keep the **SHIFT** key pressed while moving to a new location. As an alternative you can Cut (**CTRL + X**) and paste (**CTRL + V**) the item.
- To create a link, keep the **ALT** key pressed while moving to the new location.
- Items can be deleted by either selecting **Delete...** on the context menu or by hitting the **DELETE** key. In addition, you can drag any item to the trash bin to delete it.

Copyright © 2026 Vizrt



If you perform a delete operation on an referenced item, an additional option **With all references** is available. This removes all references of this file from Graphic Hub. A regular delete operation only deletes the reference within the current folder.

The **Delete all links** option also removes all Folder links of this object.

Note: Referenced items are deleted if they are referenced to ONLY the selected item. If they are also referenced to others, then they are not deleted. File Collections are **ALWAYS** deleted with all references. As the operation to resolve a lot of references takes some time, please be aware that deleting File Collections can require a lot of time!

Tip: To delete multiple items, select the items to be deleted and drag to the Trash Can.

- Creating a copy of an item within the same location is currently only possible by selecting **Duplicate** in the context menu.
- To **create a new scene**, open the context menu on either a scene or somewhere in the free space area.

To search for references item, please see Searching for References.

Tip: Files can be imported by dragging them from the filesystem (Windows Explorer) into the Asset View. For details, please see Import of Files and Archives.

To Rename an Item

1. Select the item to be renamed.
2. Open the rename text field:
   - Right-click the item and select **Rename** from the context menu, or
   - Press **F2**.
3. In the text field that appears, type a new descriptive name.
4. Press **ENTER**.

Note: Special characters, such as space, are not allowed in item names.

Note: If renaming a linked item, only the name of the selected item changes, not the linked one.

Copyright © 2026 Vizrt

To Lock an Item

An item is automatically locked when a user opens the item. It is not possible to perform this operation manually. Some types offer additional options on the context menu: **Add to Container** is available for the following types: Geometries, Materials (Classic), Material Advanced, Material Definition, Images, Fonts (Classic), FontFaces and Substances. This requires a container to be selected in the Scene Tree.

Images

There are several options available for images:

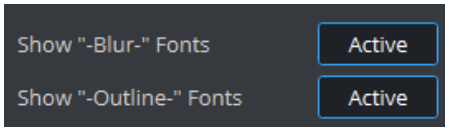
- **Open in External Editor:** Opens the image in the External Image Editor.
- **Open in Internal Editor:** Opens the image in the integrated Internal Image Editor.
- **Export...:** Exports the image from Graphic Hub to your hard drive. several formats are available.

Information: The name of the image is taken from the existing name in Graphic Hub, The file dialog is for selection the destination folder only.

- **Prefilter Environment Map:** Prepares Cylindric HDR images to be used as Environment maps. This applies to the Viz Engine Render Pipeline only.
- **Convert Image:** Converts an image from one color space to another.
- **Create PBR from selection:** Attempts to identify parts of a PBR Material set according to a naming convention, and creates a new PBR material. Only available if more than one image is selected.

Fonts

Hide or Show *Blur-* and *Outline-* variants of imported fonts. This cleans up the view by hiding those variants from the asset view. Those variants are used for example as a drop shadow effect for your fonts and are not necessarily visible. This can also be set as default in the configuration section:



Audio

- **Export...:** Exports the audio as *.wav* file from Graphic Hub to your hard drive.

Information: The name of the image is taken from the existing name in Graphic Hub, The file dialog is for selection the destination folder only.

To Select Multiple Items

You can select multiple items in the Asset view. To select single items, keep the **CTRL** Key pressed while choosing the items you want to select. If you want to select a range of items, select the first item and then use the **SHIFT** key to select the last one. An alternative is to use your mouse to draw your selection.

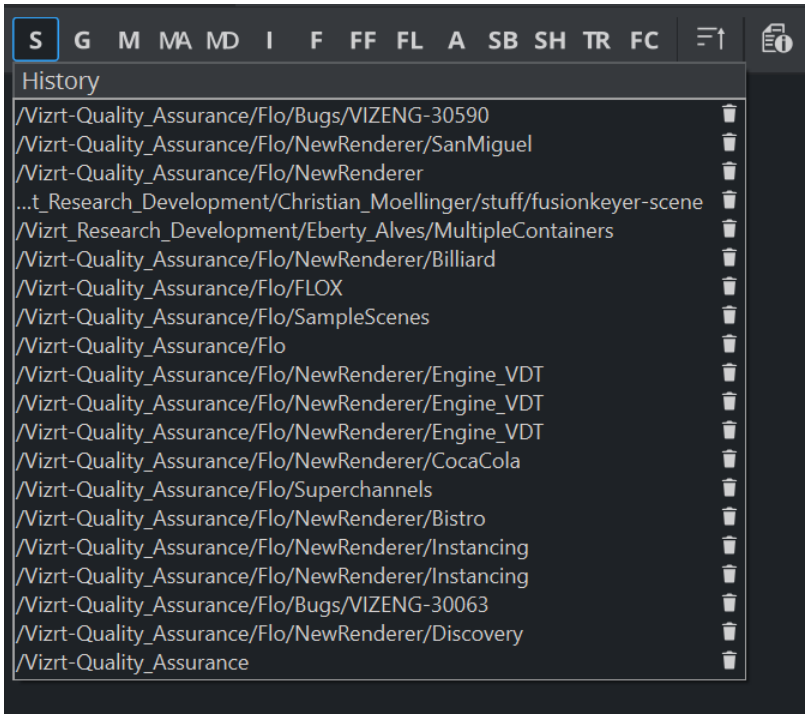
Copyright © 2026 Vizrt

Tip: Multiple items of the same type can be selected and dragged into the scene tree or scene editor. This is for example useful if you want to animate an image sequence using the OMO plug-in.

The Properties Panel is disabled if multiple items are selected.

Location History

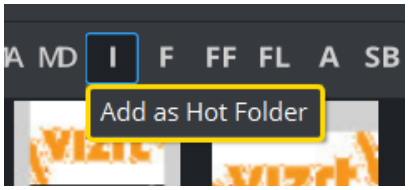
Asset view keeps a history of each location where a certain type has been used. This can be shown by **CTRL+ right clicking** on one of the tabs. For example, if you **CTRL + right-click** on the **S** tab, a history of the last 20 locations where an item of type Scene has been modified are shown.



To remove this browsing history, click on the small trash can next to the entry.

Hot Folders

Hot folders are a kind of history items, but bound to a certain type of item. For example a folder with all your environment maps (images) can be set as Hot Folder. This allows to quickly jump to this folder and set the view to *Image*. To do so, right click on the folder you want to remember and the type of your item and choose **Add as Hot Folder**.



Copyright © 2026 Vizrt

You can then quickly jump to your Environment maps by right clicking on **I** and select your Hot Folder location.

To remove Hot Folders, click on the small trash can next to the entry.

Information: Hot Folders and Location History are stored on a per user level on Graphic Hub. If multiple users use the same login (for example, Guest) they will be mixed up.

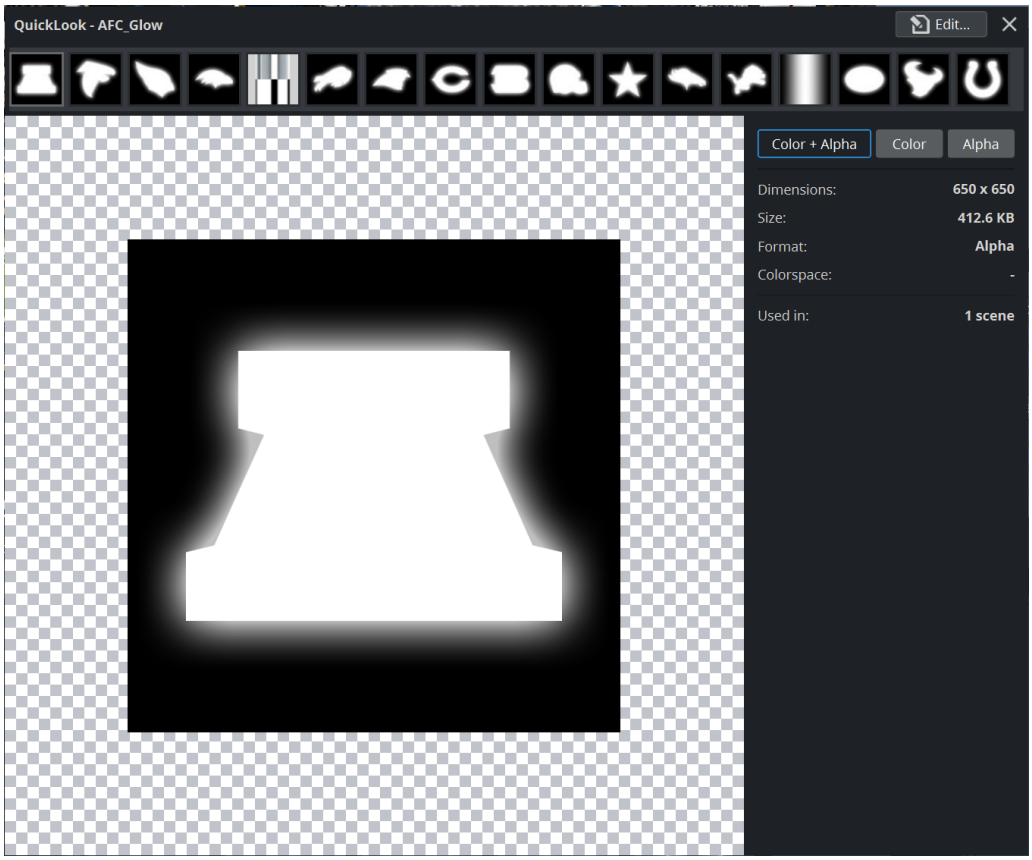
7.2.4 Quick Look

If you select one or more items and press the **SPACE** (default) key, a window previewing the item(s) will be shown without the need to add them to the current project.
The Quick look feature works on the following items.

- Images
- Material Definitions like
  - Phong Materials
  - PBR Materials
  - Substance
  - Texture Renderers
- Audio Files
  As an example, select a series of images and hit the space key.

Copyright © 2026 Vizrt

To Quick Look Images

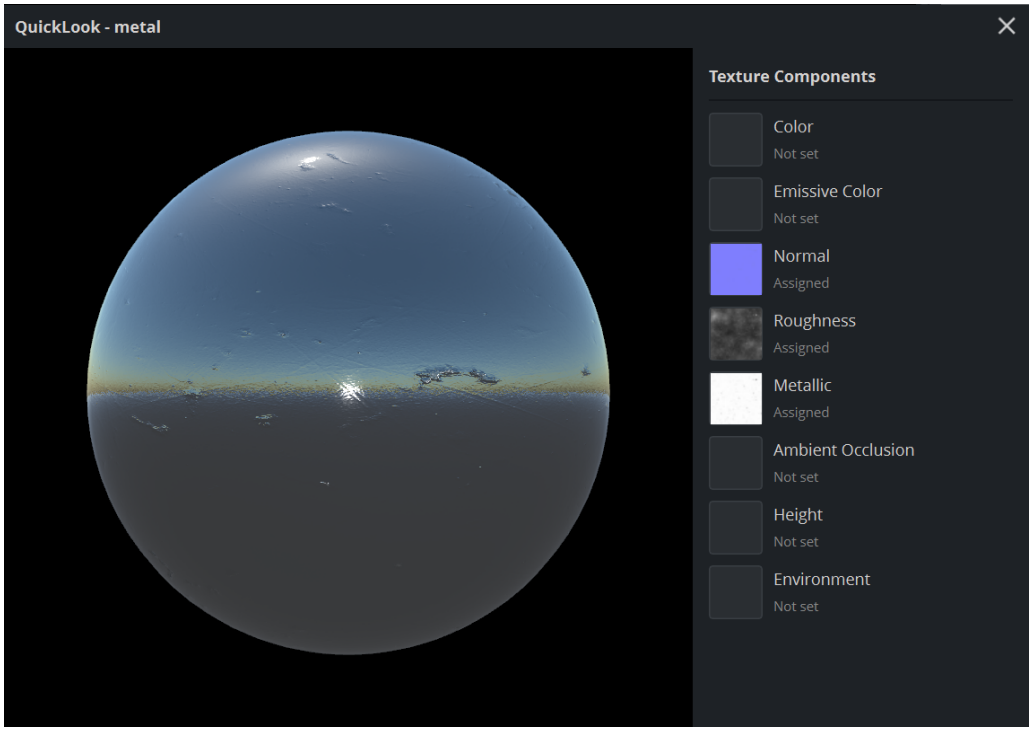


This will shown a new window with a preview of the selected image, on the right side, you'll see additional information like format, dimension, color depth etc.

To Quick Look Material Definition types

Material Definitions like PBRs, Phong or Substance renders can also be previewed.

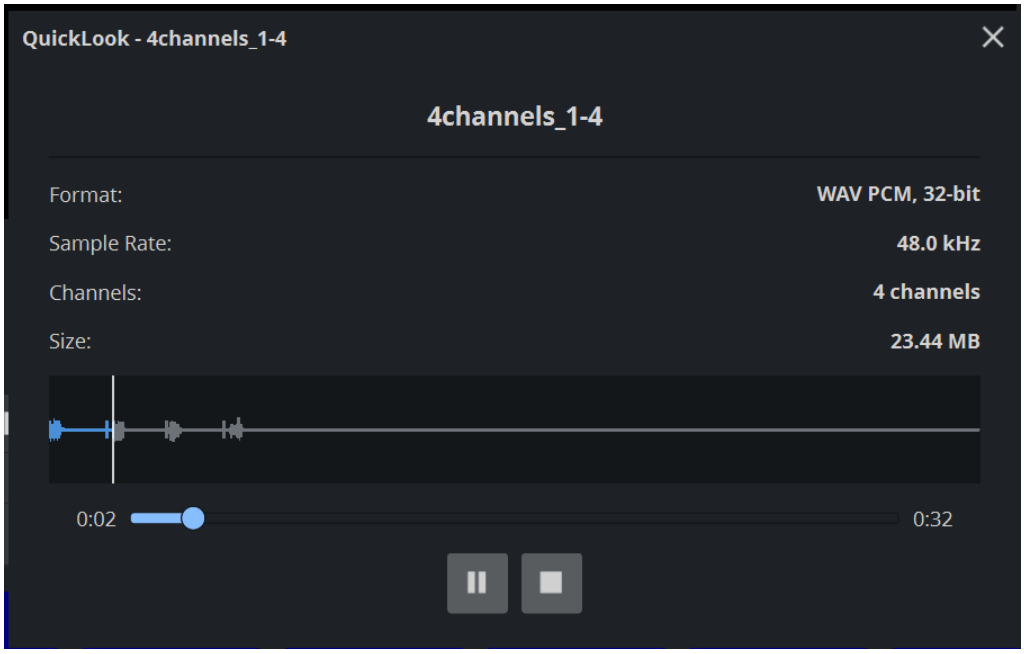
Copyright © 2026 Vizrt



The Preview Sphere can be modified by using the normal Scene Editor shortcuts.

To Quick Look Audio Files

For audio files, the Quick Look feature allows to play the file first without adding it to the project:

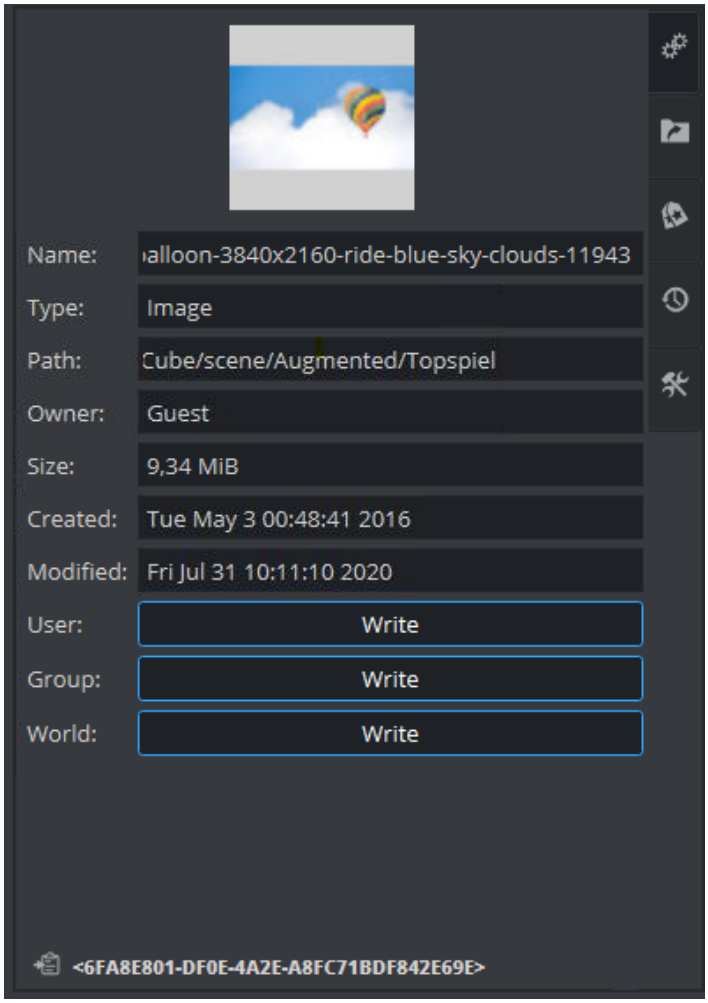


Copyright © 2026 Vizrt

The Waveform preview is only supported on some formats.

7.2.5 Properties

The Asset view offers a side panel which shows detailed information about an item. Click on the button to show or hide the panel.



General

The general tab shows basic item properties. On top a large preview icon is shown.

- **Name:** Shown underneath it, you can copy the name, but you can not modify it here. To rename an item, see Renaming an Item.

- **Type:** Shows what kind of asset you selected.
- **Path:** Shown underneath. The location of the item. Please keep in mind, that a linked item can appear in multiple locations. The path shown here is the currently selected one on Graphic Hub. If an asset from the *Search result* is selected, then it is the first *folder link*, otherwise the exact path of it.
- **Owner:** Shows the active user that owns this item. For more information about user and group management, see **User and Group Management** in the **Database Management** section of the Graphic Hub Administrator Guide.
- **Size:** Shows the actual size of the selected item in Graphic Hub. Items on Graphic Hub have different sizes than for example the same file on your hard drive (images on Graphic Hub contain additional data like mipmap levels, preview icons, etc.).

The two date items shows when an item has been **created** and when it was last **modified.** Since Viz Artist version 4.4, the time is also shown. **Permissions** can be changed on User, Group and World level.

- **User:** User may overwrite their files and create files.
- **Group rights:** User can overwrite files which belong to the group they are a member of.
- **World rights:** User may overwrite files which belong to anyone.

On the bottom, the **UUID** of the item is shown and be copied to the clipboard with the small clipboard icon.

Tip: The UUID of any selected item can also be copied to the Clipboard by using the **CTRL + U** shortcut.

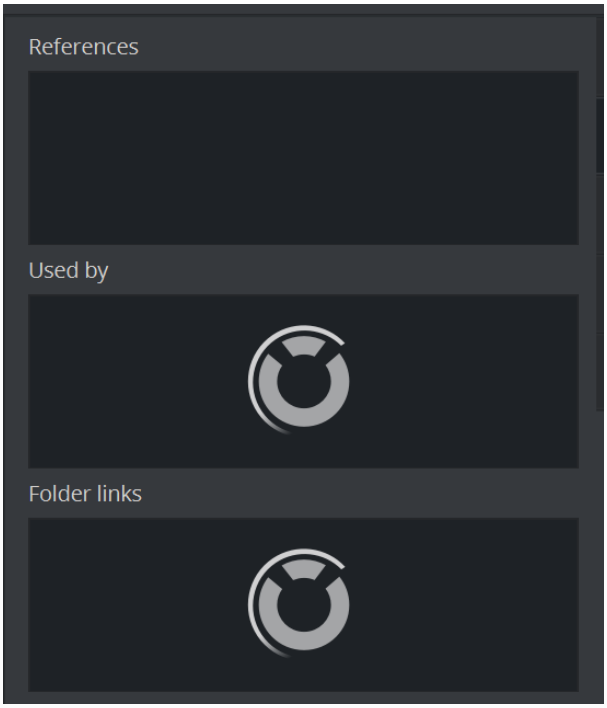
Links

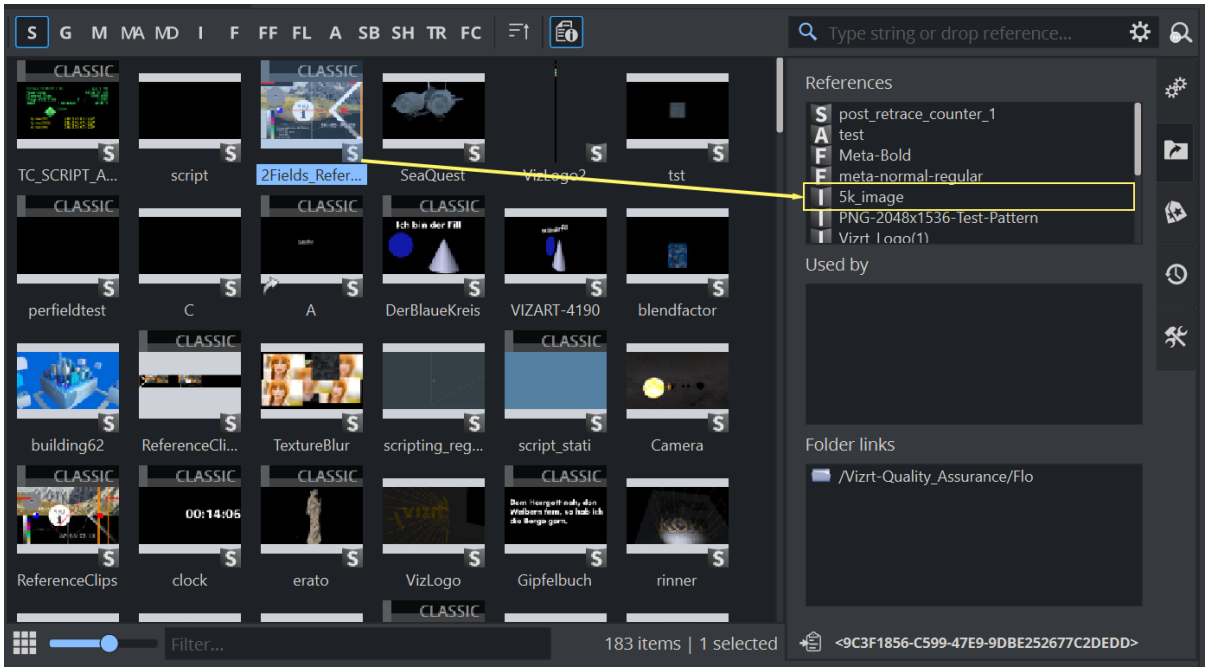
The Links tab shows the links that are assigned to the selected item. There are two different types of links. *References* and *File links*.

- **References:** Contains all items an item consists of. For example, a scene contains of various Images, Fonts and Geometries. All of these belong to that scene are references.
- **Used by:** Represent the location the item is being used. This can be for example a scene, a merged geometry etc.
- **Folder Links:** Represent all locations this item is placed on the Graphic Hub.

Information: Some files can have quite a lot of References or File/Folder links. While retrieving this information, a spinner is shown:

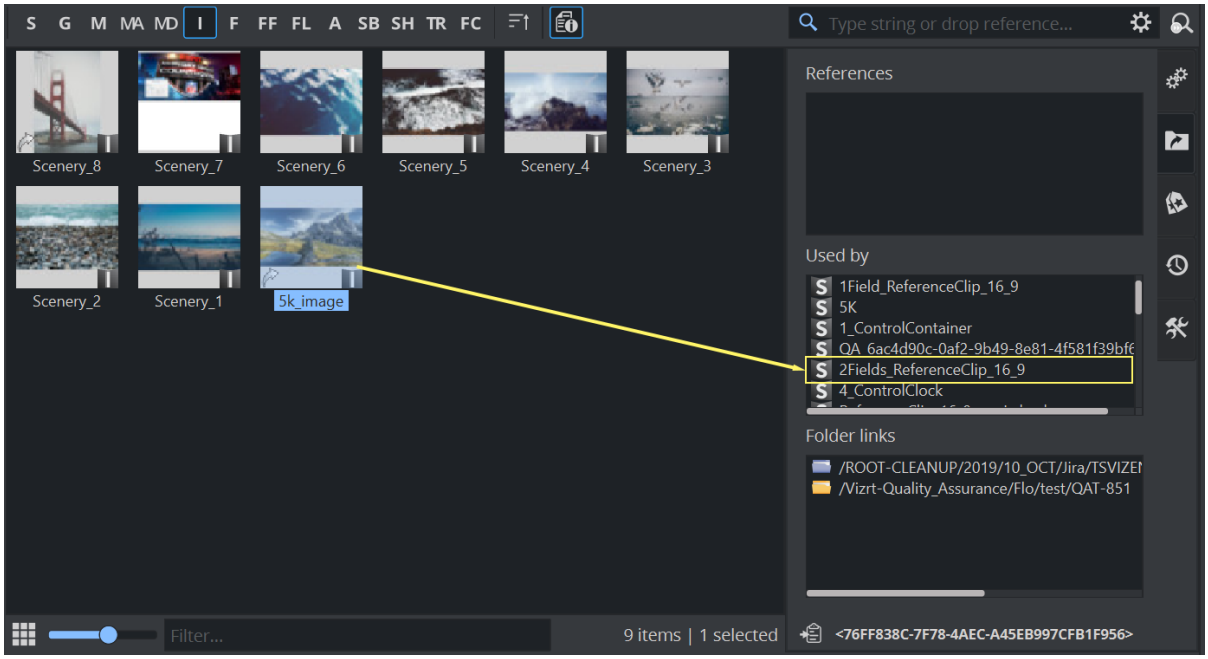
Copyright © 2026 Vizrt





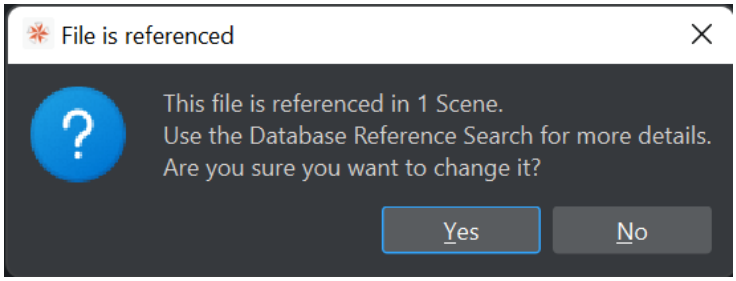
Example: The selected *2Fields_ReferenceClip_16_9* scene has various references. One of them is an image called *5K_image*. If you select the *5K_image* entry in the reference list, the asset view jumps to first location (if the image is linked) and shows the file links this image has also be used.

Copyright © 2026 Vizrt



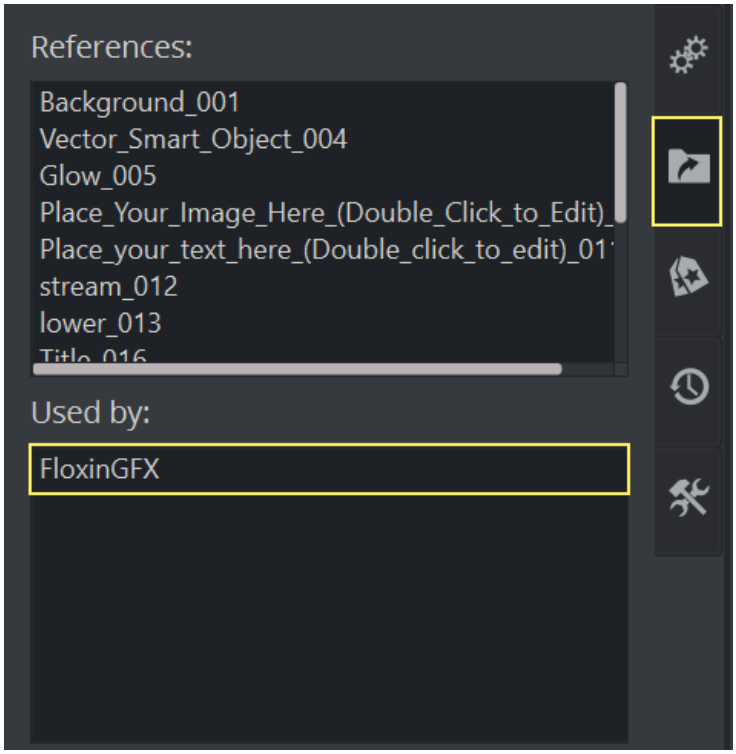
This shows shows all other objects that use this image. One of these File links is the scene *ReferenceClip16_9*. If you click on *ReferenceClip16_9* in the File Links list, the asset view would jump back to the location this scene is stored in or as in this example back to the first explanation image. A detailed explanation on how to search for reference can be found chapter Searching for References.

If you get an error message when opening a scene



the scene is most probably being used inside a GFX channel or Superchannel within another scene. Check the **File Link** properties of this item to check which other files is using this:

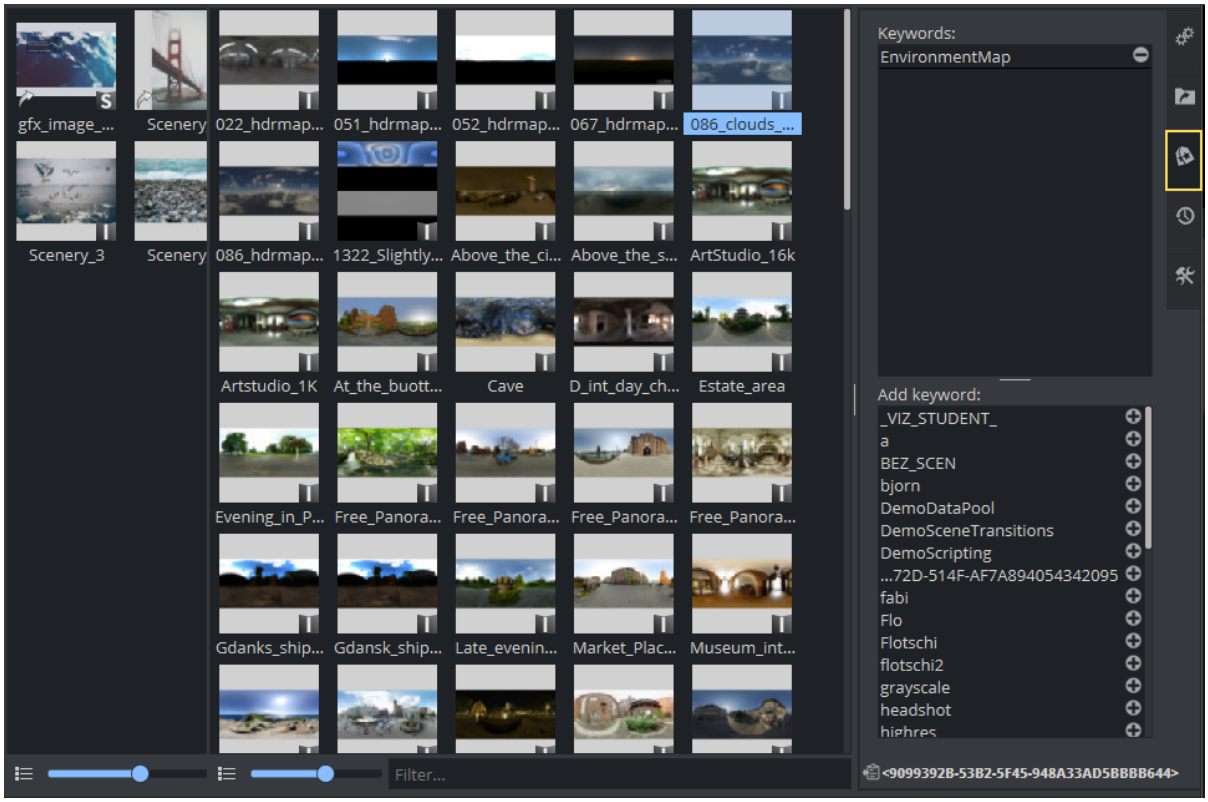
Copyright © 2026 Vizrt



Keywords

Keywords can be assigned to all types of items. They can be useful for example to categorize or label your items. Keywords can also be used as a search criteria.

Copyright © 2026 Vizrt



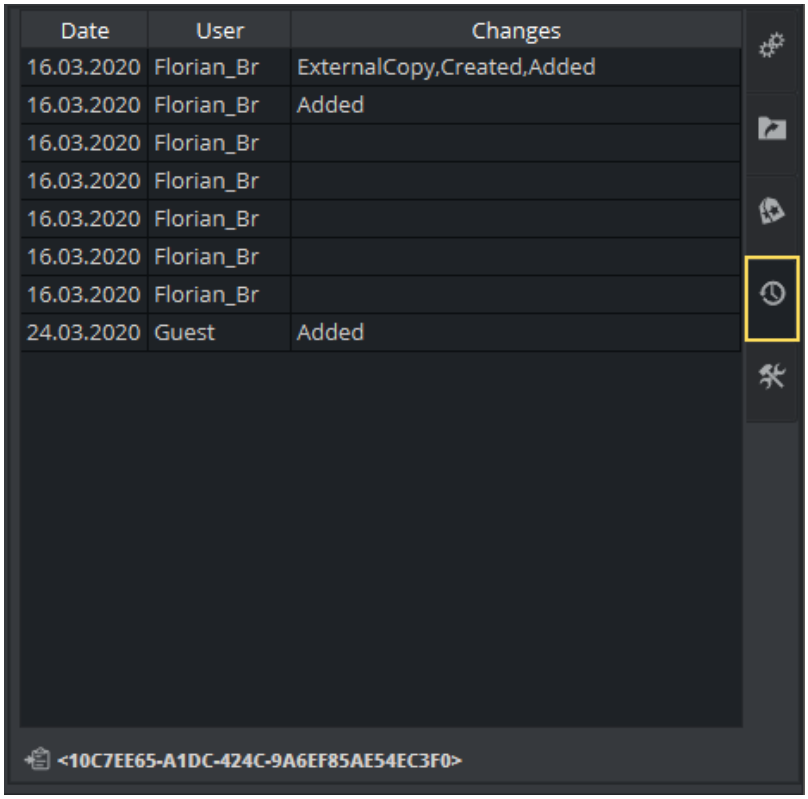
To assign a keyword to your selected item, either double-click on the keyword you want to assign or click the small **+** sign. To remove it, do the opposite.

Tip: Use Graphic Hub Manager to create and manage your keywords.

History

Graphic Hub stores a simple history of what has happened with your items. When a assets has been created or imported, modified etc. This is useful if you need to know who modified or imported something.

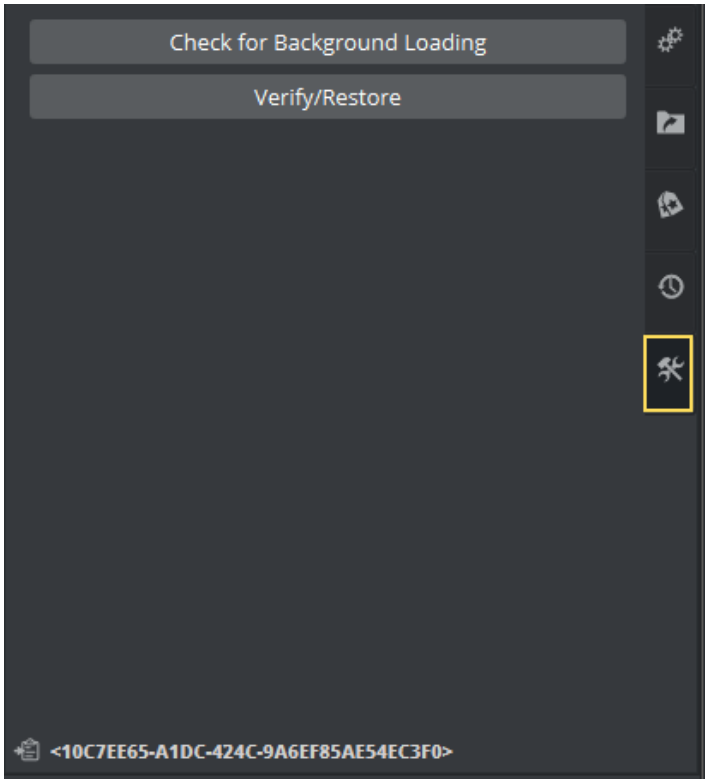
Copyright © 2026 Vizrt



Advanced

Special tools can be found in the Advanced section.

Copyright © 2026 Vizrt



- **Check for Background Loading:** Checks if the selected item is suitable for Background Loading. A message box opens and shows if the item is suitable, or shows why the item is not suitable.
- **Verify/Restore:** Locates, verifies and attempts to correct a corrupted item.

7.2.6 Search

Searching for items has been unified in Version 4.3. There is no longer a split between normal search and the freetext search and the options have been reduced to a minimum.

Copyright © 2026 Vizrt



The following widgets are important for finding items on Graphic Hub

- **Search Field (1):** Defines the search term.
- **Result Bar (2):** Shows what query has been executed and the number of search results.
- **Advanced Properties (3):** Displays additional options for more precise queries.
- **Show last search results (4):** Allows to quickly jump to the result so your last search.
- **Saved searches (5**): Shows a list of saved search queries.

Search results are shown inside the Items View. On top of the view is the results Bar. This shows what search result is being displayed. For example *Freetext:* or *Name:* followed by the search term. On the right side, the number of search results is shown.

	To switch back to the basic Items view, click the button on the left side.
	To switch back to the search result.

Search results can be saved, so it is easy to quickly recall them. They are listed in the **Saved searches** Panel. If not needed, the panel can be moved to the bottom.

To save your search, open the advanced panel and click on the  button. Items can be removed again by clicking on the trash bin icons.

Copyright © 2026 Vizrt

Information: Search queries are stored on a per user level on Graphic Hub. If multiple users use the same login (for example, Guest) they will be mixed up.

Simple Search

To locate an item in Graphic Hub, type any property into the search field on top of the assets view. This can be either part of the name, the owner, the type, or keywords. In addition, Graphic Hub add-ons are also included by default.

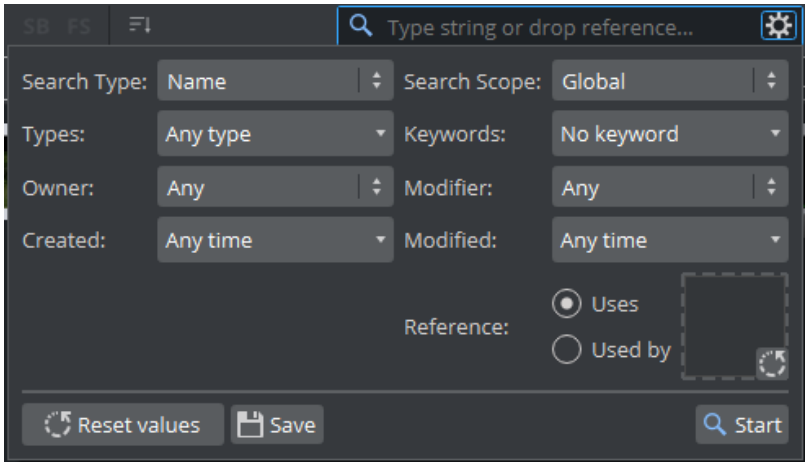
Some examples:

Search Term	Description
vizrt	Searches for all items, that has *vizrt* as: • Part of the name • A keyword assigned to *vizrt* • Owned by user *vizrt*
scene	• Shows all scenes found on your Graphic Hub. • All items with *scene* in the name. • All items with a keyword *scene* assigned.
project folder -geometry -scene -image	Lists all folders and projects (excludes geometries, scenes and images).
image Mousey_Mouse	Lists all images owned by user *Mousey_Mouse.*
blue material -folder -project	Lists all materials with part of the name like *blue*, but excludes folder and projects
Flo?an	Lists all elements, that have the term *Flo* followed by **any character** followed by *an* **.**
*Flo*an*	Lists all elements, that have the term *Flo* followed by **any text** followed by *an*.
Elem[b-g]nt *Elem[shje]nt*	Matches a range or a list of characters.
{AB3584	Lists all elements where *AB3584* is part of the UUID.

Copyright © 2026 Vizrt

Advanced Search

If the simple search is not fulfilling your needs, you can use the Advanced Search. It allows you to specify more search criteria in detail.



The default Search Type is **Freetext**, which means, your search criteria matches on:

- The name
- The owner
- The type
- The keywords.

Graphic Hub Add-ons are also included by default. Search by **Name** uses only the item's name as matching criteria. To limit the search to a single folder or product including its subprojects/folders, switch the **Search Scope** from Global to Selected folder. This is for the currently selected folder only. **Types** specifics which item types to search for. This is more or less the same as for example doing a *blue material* to find all blue materials, but lets define your type more gradually. By specifying **Keywords**, your search also includes these in the search query. If you need to find an item by a specific **owner**, the dropdown list lets you to choose a certain user.

To search for a certain time an item has been modified, for example all scenes that have been changes last week, the modified property can be used. It can be:

- **Any time:** Uses default value with no range specified.
- **From:** Uses a start date until now.
- **Until:** Shows any item modified before the specified date.
- **Between:** Shows if an item has been modified between two dates.

Copyright © 2026 Vizrt

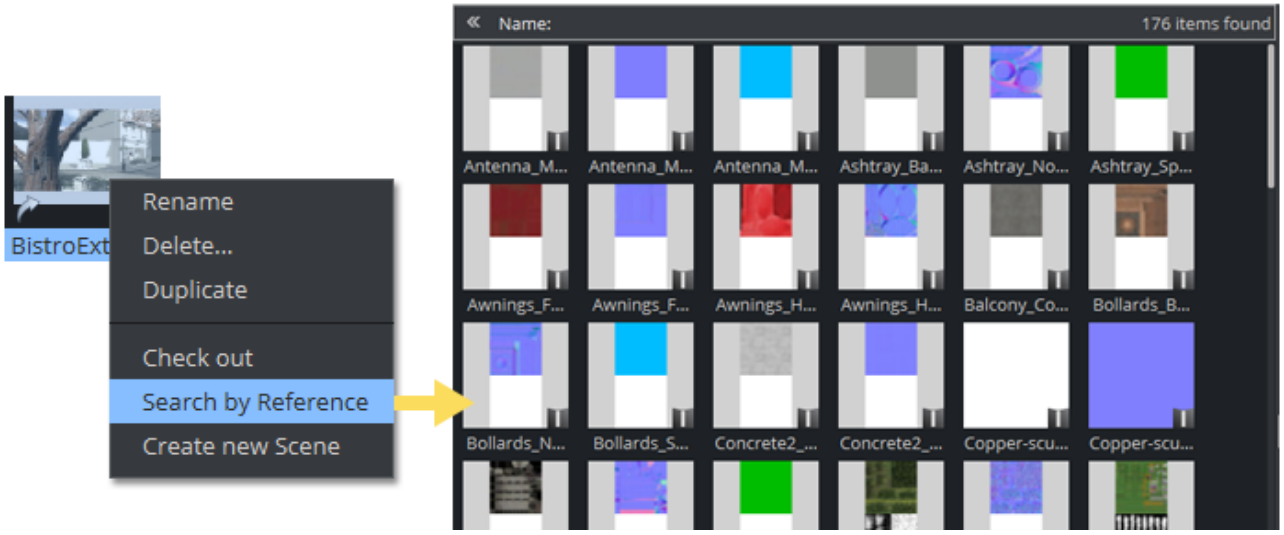


Reference Search

There are two different ways to search for references:

Reference Search on an Item

Right click on an item (or **CTRL + R**) gives one the possibility to search for its references.



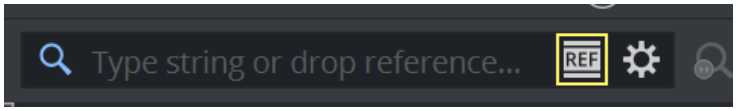
- Scenes automatically show the items that refer to this scene.
- Fontfiles show their related Fontface.
- Other items (like images, etc.) show the File links instead.

Material Definitions for example can have textures as file links, but also be referenced by scenes. By default, the Reference search on a Material Definition shows the scenes it has been used by. If you need to know the textures a Material Definition is using, you need to use the Advanced Settings.

Note: It is not possible to perform a reference search for classic materials, as materials are not referenced. Material information is held inside the Scene.

Copyright © 2026 Vizrt

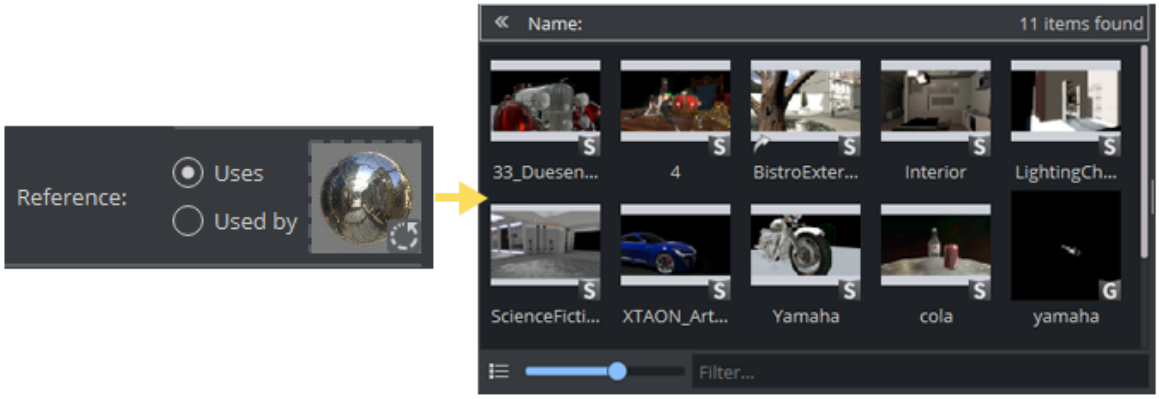
A Reference Search is indicated by a small icon in the search bar:



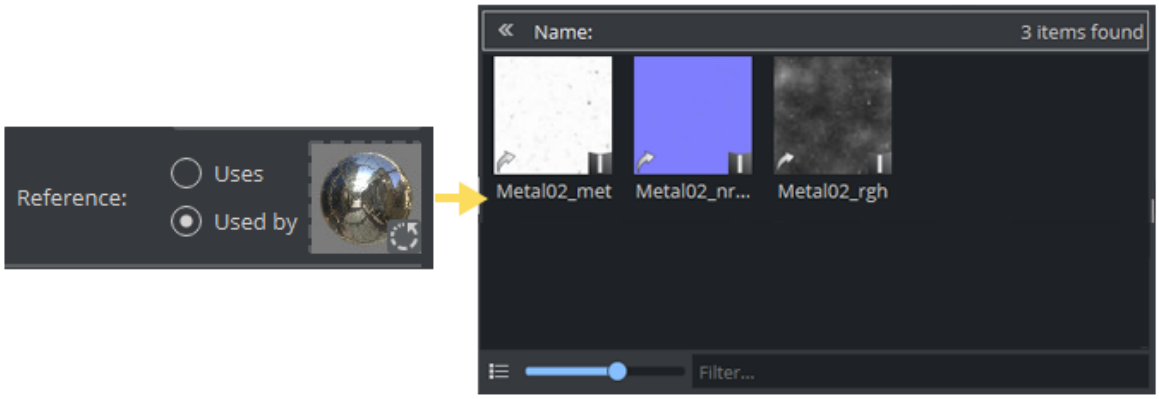
Advanced Reference Search

The advanced search panel allows you to specify your request even further. Drag an item to the drop area or directly into the search field. You can then specify if you want to search for References (**Used by**) or for File links (**Uses**).

- **Uses:** Searches for all objects that use this object, in our example all scenes that use the Material Definition:



- **Used by:** Searches for all items, that are being used by this object. In our example, the textures for our PBR Material Definition.



Troubleshooting Search

If your search result is not the expected one, please verify your advanced settings. If the search type is switched to *Name*, the search input field delivers fewer results than running in freetext mode. Always verify the mode in the search bar. Use the **Reset values** to bring back the advanced properties to its default value.

Copyright © 2026 Vizrt

7.2.7 Shortcuts

Please see Asset View Shortcuts.

See Also

- Working with File Collections
- Asset View Shortcuts

Copyright © 2026 Vizrt

7.3 Working with Items

This section describes how to work with various items stored on Graphic Hub.

- Working with Fontstyle Items
- Working with Various Materials
- Working with Geometry Items
- Working with Scene Items
- Working with Image Items
- Working with Audio Items
- Working with File Collections

There are more types of items stored on Graphic Hub like Shared Memory Maps, Video Clips or Viz Pilot Templates. They are not part of this chapter.

Copyright © 2026 Vizrt

7.3.1 Working with Fontstyle Items

The way how fonts are handled depends on the selected Render Pipeline. While the Classic Render Pipeline scenes use traditional fonts and fontstyles linked to basefonts, the Viz Engine Render Pipeline scenes follow a more modern approach and use Font faces and Font files.

- Fonts in the Viz Engine Render Pipeline
  - Font File
  - Font Face
  - Font Library
  - Font Management
  - Text
- Fonts in the Classic Render Pipeline
  - To Create or Edit a Fontstyle
  - Fontstyle Editor
  - Modifiers
  - Fontstyle Viewer
  - Text Properties

Fonts in the Viz Engine Render Pipeline

To understand the concept of the **New Font System**, some Font terminologies need to be clarified first.

- **Font family:** A group of font faces that share basic design characteristics and may vary in weight, width, or style (for example, Arial, Cordia New, or Helvetica).
- **Font face:** A specific weight, width, or style of a particular font family (for example, Arial Regular, Arial Bold, or Helvetica 12 Roman Medium Normal).
- **Font:** A software file container that contains a single font face or multiple font faces whether from a single font family or multiple font families. For example, *angsana.ttc* font, a font containing eight font faces from two font families: Angsana New (Regular, Bold, Italic, and Bold Italic) and AngsanaUPC (Regular, Bold, Italic, and Bold Italic).

By mimicking font terminology, two new database types are introduced. One is a **Font File(FL)**, and another one is a **Font Face(FF)**. Also, The **New Font System** comes with a **Font Library** which serves font discovery that has never been before.

Font File

Font File is designed to mimic Font in Font terminology with **FONT_FILE(FL)** as its corresponding database type**. Font File** stores the binary data of a font regardless of its container type(*.ttc*, *.ttf*, *.otf*, *.otc*, *.woff*, etc.). Also, **Font File** stores the checksum calculated from the binary data. This allows the **New Font System** to use the checksum in duplication detection.

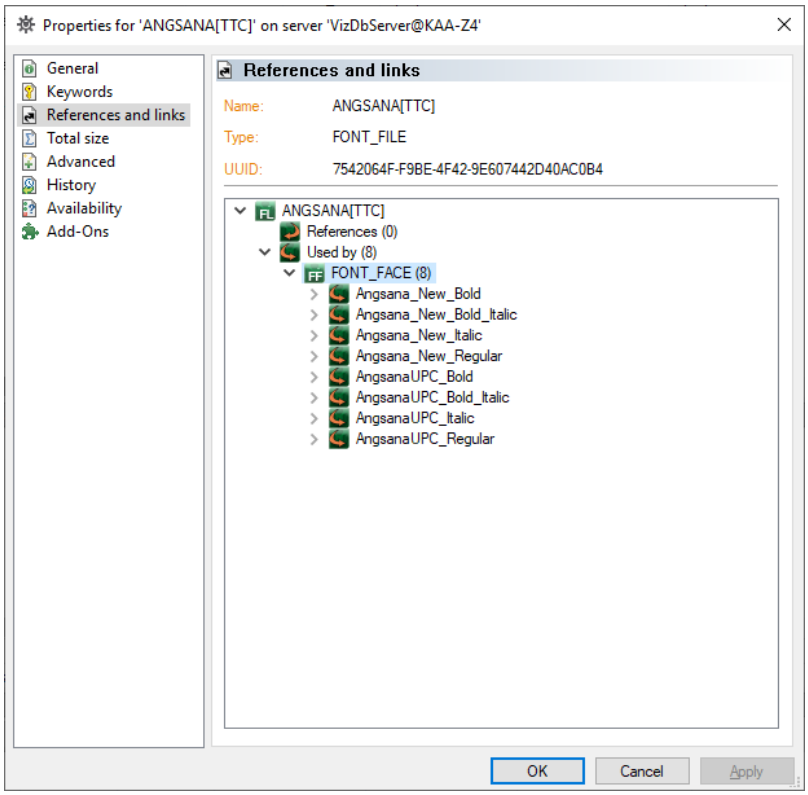
> **Information:** Different versions of fonts have different binary data, hence the checksum is different.

Copyright © 2026 Vizrt

Font Face

Font Face is designed to mimic Font face in Font terminology with **FONT_FACE(FF)** as its corresponding database type. This is a database object that is used to assign to **Text** to gain the looks and feels of a particular font face. Unlike fonts in the Viz Engine Classic Render Pipeline, the Viz Engine Render Pipeline comes with an advancement in new text rendering techniques that requires no pre-generated data to be stored in a **Font Face** object; whether glyph geometries or vector atlas texture for a new Razor text rendering. This reduces disk consumption significantly and speeds up scene loading time.

With references and links in Graphic Hub Manager, it visually illustrates the relation between **Font File** and **Font Face** objects.



Font Library

A component in Viz Engine is capable of discovery of all Font Faces in Graphic Hub. It is designed to be used during Scene design time. It could take some time to update the Font Library for the first time since it needs to discover and open all Font Face objects from Graphic Hub. It is faster later on because of the caching mechanism. **Font Library** stores caches at *%programdata%/vizrt/VizEngine/FontLibraryCache*.

Information: When dragging a Classic Font into a Viz Engine Scene, a Text placeholder is created.

Copyright © 2026 Vizrt

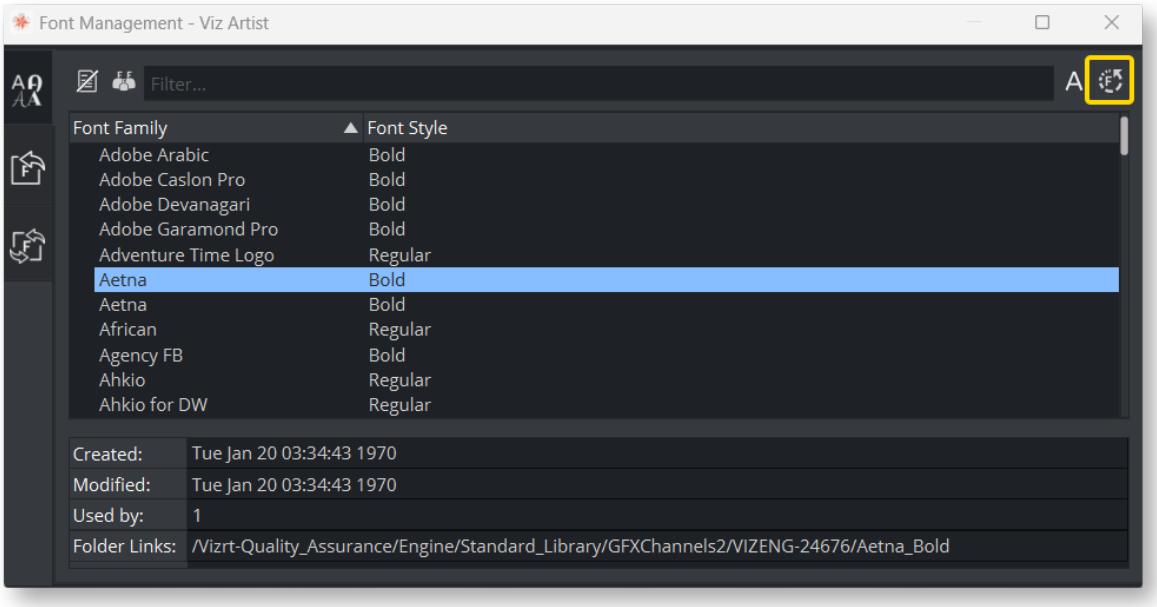
Font Management

It is a Viz Artist UI component that aims to simplify the way to work with the New Font System. Three main functionalities are:

- Browse Fonts
- Import Fonts, and
- Convert (Classic) Fonts

Browse Fonts

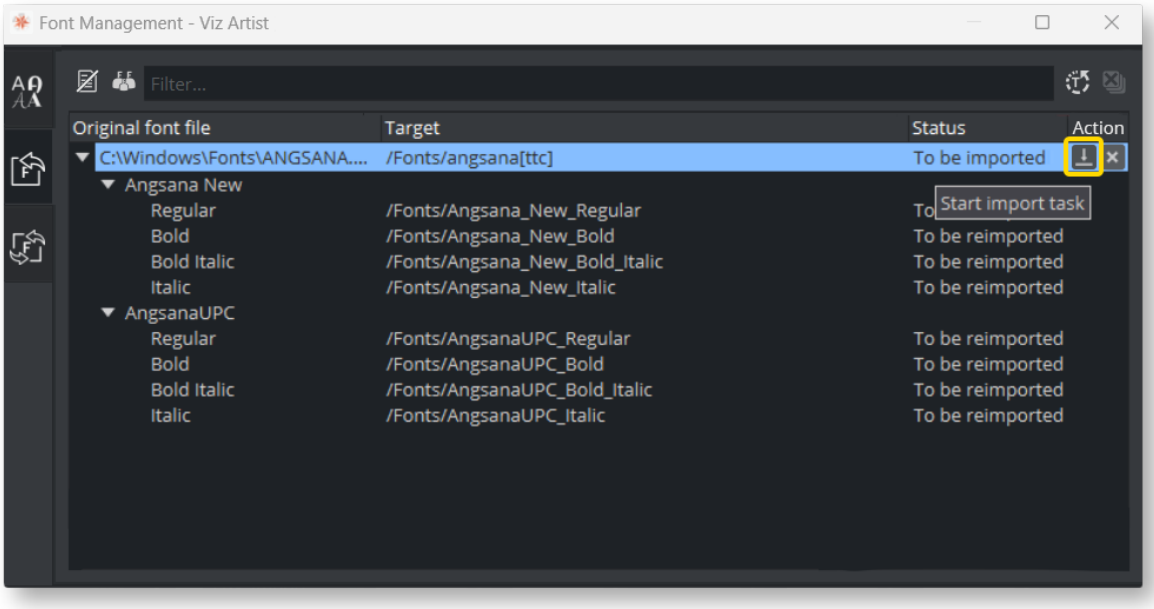
The *Browse Fonts* tab shows all font faces available in the current Graphic Hub. If nothing shown, click **refresh** to build a font system library.



Import Fonts

The *Import Fonts* tab allows users to import fonts from disk to Graphic Hub. You can drag a font from a local filesystem to the panel, then click **import**.

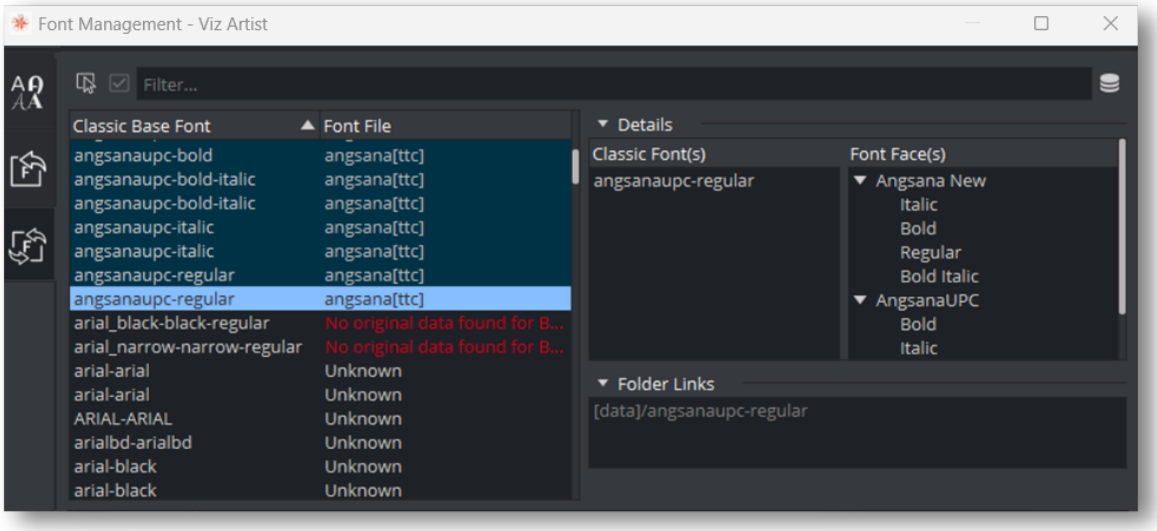
Copyright © 2026 Vizrt



Convert (Classic) Fonts

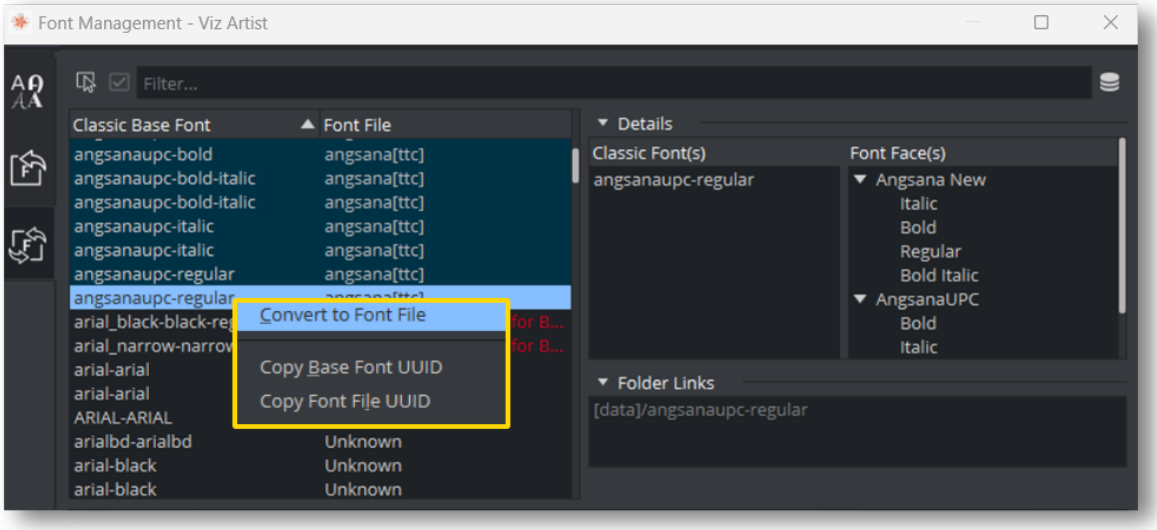
The *Convert Fonts* tab allows user to convert the existing Classic fonts to be available in the New Font System. The panel shows a list of Classic Base Font in one column. And for Viz Engine Font File column, here are possible values:

- **Viz Engine Font File name:** Means the Classic Base Font is converted (for example, angsana[ttc]).
- **No original data found for Base Font:** Means the Classic Base font has no original data, so Viz Engine cannot convert it.
- **Unknown:** Means the status is not yet determined. Users can click on it to force updating its status.

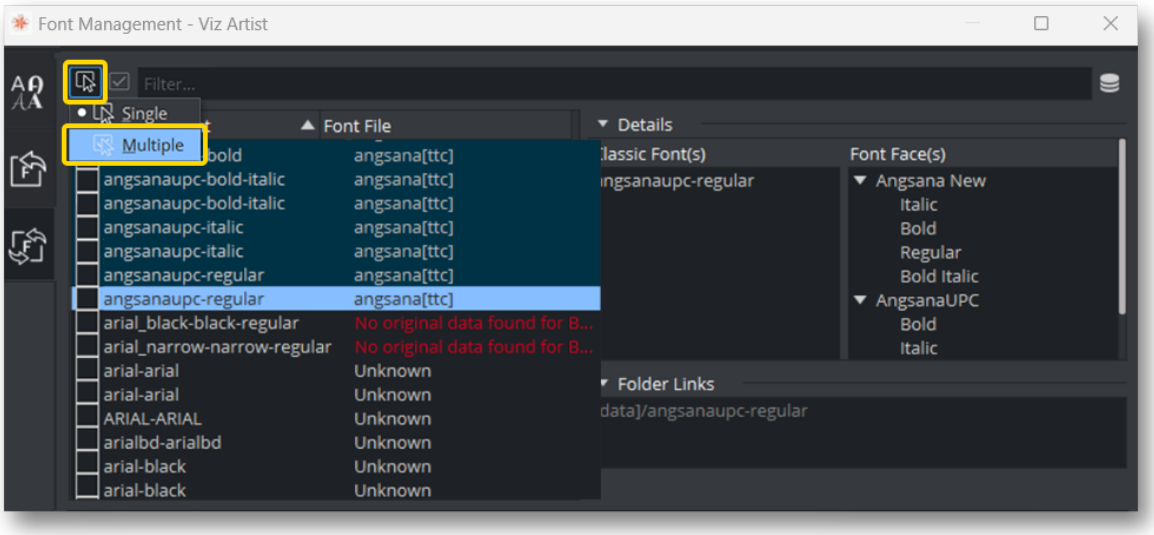


To convert a single Classic Base Font, right click on the Classic Base Font to show the context menu. Then, click **Convert to Viz Engine Font File**.

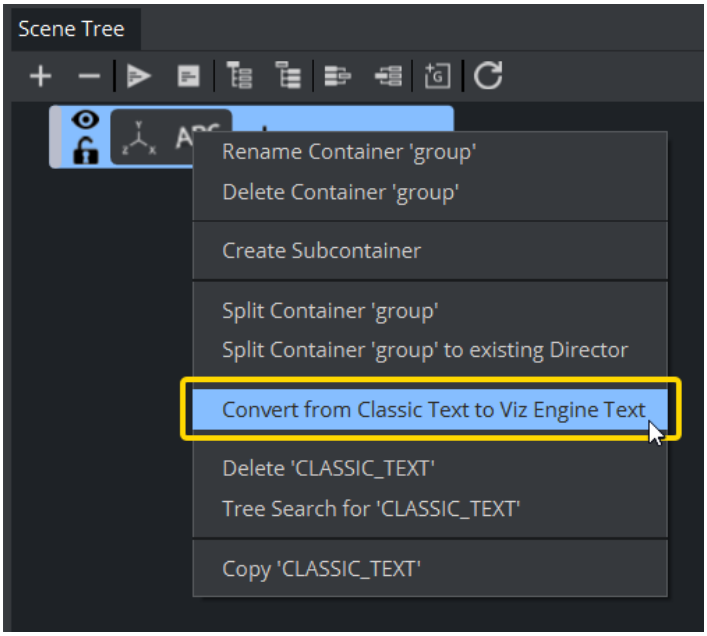
Copyright © 2026 Vizrt



To convert multiple Classic Base Fonts, click the button on the top left of the tab. Click **Multiple**. Then, select Classic Base Fonts to be converted. Then, click **Convert N Base Font(s)**.



Fonts can also be converted by using the context menu within the scene tree:

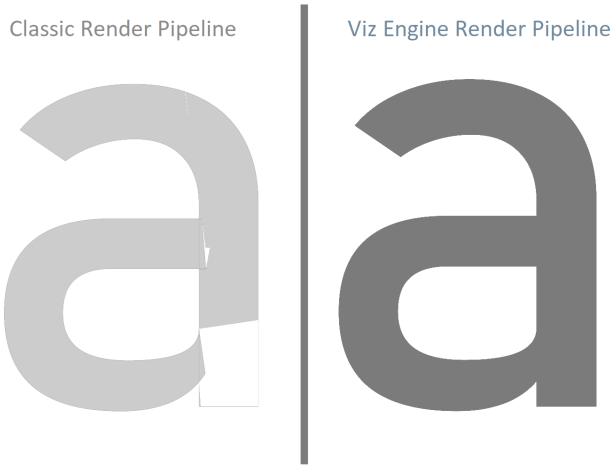


Text

The new text rendering system only supports two ways of rendering: **Geometry** and **Razor** These two methods aim for different purposes.

Razor

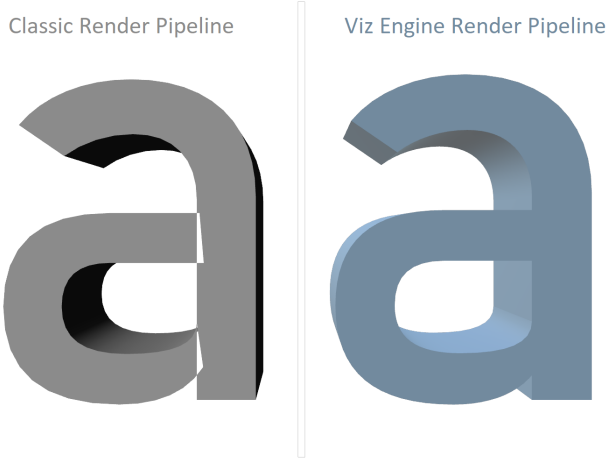
Sharp, Fast, and High performant suitable for 2D use cases such as Lower Thirds, Scoreboard, etc. This rendering method is also overhauled completely. It serves faster, sharper, and better results than before. This should be your preferred render method for Text.



Information: Razor fonts can also be rendered into a texture. This can be used to improve rendering of overlapping transparent glyphs. This feature will also enable more effects in future versions.

Geometry

Flexible and suitable for 3D use cases such as Augment Reality, etc. With an improved tessellation algorithm, **Text** serves a better look compared to the classic geometry text in the Classic Render Pipeline. Also, **Text** is equipped with Extrusion functionality that could be turned on or off when the rendering method is Geometry. This is to make the workflow simpler than ever.

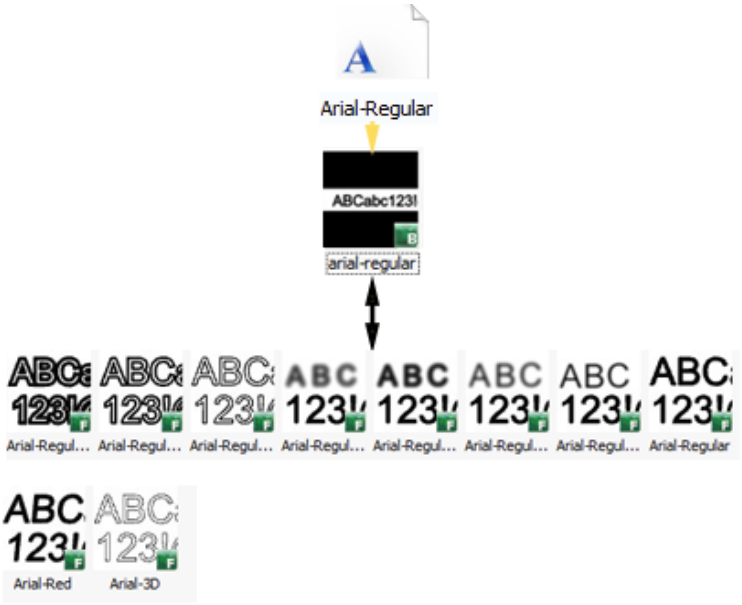


For additional information about Font Rendering in Viz Engine, please see Working with Text and New Font System.

Fonts in the Classic Render Pipeline

Fonts in the Classic Render Pipeline are based on so-called basefonts, They are located in the *data* folder of Graphic Hub and store the original font. When importing a font (for example, *.ttf*, *.otf*, etc), the basefont is imported and a Font object is created linked to this basefont. In addition, Blur and Outline variants are created.

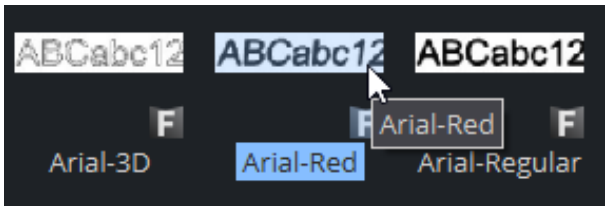
Copyright © 2026 Vizrt



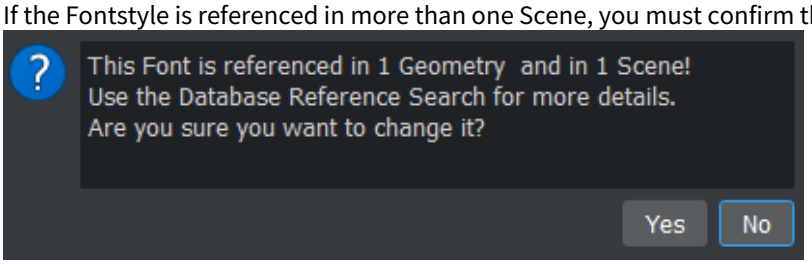
In the above example, the font Arial-Regular is imported as a basefont into Graphic Hub. Based on this basefont, several Font variants are created. Each of them is referring to the basefont. These variants and a set of Fontstyle Modifiers can be used to create new styles of his font. In our example, a designer also created two variants using the Fontstyle Editor (Arial-Red and Arial-3D).

Each Font and its variants can be used inside a scene using the Classic Text Editor.

To Create or Edit a Fontstyle



1. Double-click the Font in Asset View.
2. If the Fontstyle is referenced in more than one Scene, you must confirm that you want to change it.

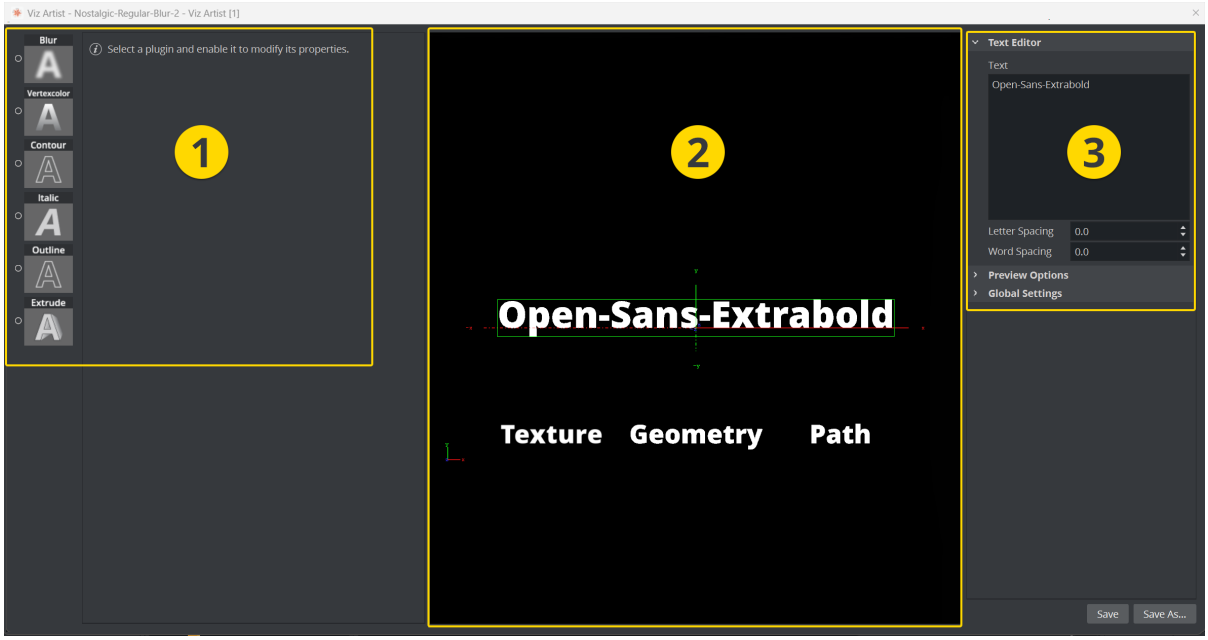


3. In the Fontstyle Editor define the new settings.
4. Click the **Save** button.

Fontstyle Editor

The Fontstyle Editor makes it possible to modify fontstyle items in the Graphic Hub database to match your needs and save it as a new font, based on the original font and with additional styles provided by the font style modifiers.

As Viz Artist uses an open interface for the plug-ins used as modifiers, almost any modifications can be applied to a fontstyle. New fontstyles can be created based on existing fontstyles through the **Save As...** operation.



The Fontstyle Editor consists of:

- Modifiers (1)
- Fontstyle Viewer: Preview of the modifications (2)
- Text Properties
    - Text Editor
    - Preview Options
    - Global Settings

Note: The properties help to illustrate how the modifications look in a scene. Changes made on the right side of the editor inside the Preview Options are kerning and material, are not saved with the fontstyle. However, there is one exception to this rule. If the kerning between two fixed characters are modified, this is saved with the fontstyle. To create a kerning between two fixed characters, use the keyboard shortcuts **ALT + Left Arrow**/**ALT + Right Arrow**.

- **Global Settings:** Shows various properties for the fontstyle.
    - **Font Default Style:** Defines the font default style. Options are *Geometry* or *Texture*.
    - **Shadow Default Style:** Defines the shadow default style. Options are *Geometry* or *Texture*.
    - **Texture Quality:** Defines the quality of the texture. Options are *Low*, *Normal*, or *High*.

Copyright © 2026 Vizrt

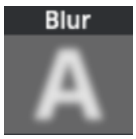
Note: The texture quality cannot be changed in a Text Editor.

Modifiers

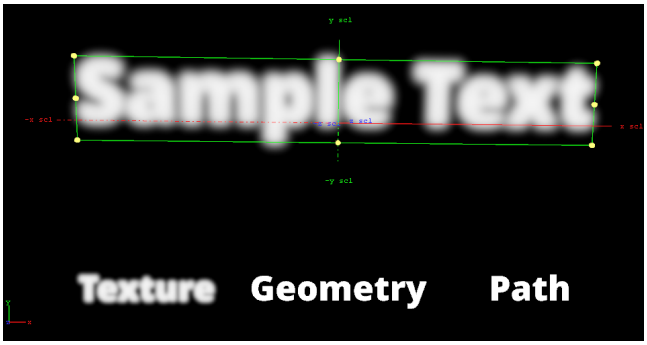
Modifiers are used to change either the texture, geometry, path or all components of a fontstyle. For example, the blur function can only be applied to the texture part of a fontstyle, whereas the extrusion only works on geometry. A fixed set of modifiers come with a standard Viz Engine installation, but the system provides an open connection so that additional modifiers can be obtained or created and added as plug-ins. A modifier that is applied to a fontstyle is automatically enabled. An enabled modifier implies that modifications are reflected in the preview area. When adding multiple modifiers, the overview in the preview area can get confusing, and it may be necessary to disable one or more modifiers.

To enable or disable a modifier, click on the checkbox next to the modifier.

Blur



Use the Blur modifier to blur a fontstyle in texture mode.

Options	Sample
• **Level:** Sets the blur level. • **White Level:** Sets the white level in the blur effect.	
Supported Render modes: • Texture	

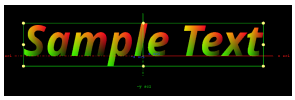
Color Per Vertex



Use the Color Per Vertex modifier to change the color of fontstyles by applying it to the vertices. The circles in the Color Per Vertex modifier represent the four corners of the fontstyle text:

Copyright © 2026 Vizrt

Options	Sample

• **Color Level:** Defines how much color should be applied.
• **Color Box**
  • **Max BB:** Applies the color to the vertices of the bounding box.
  • **Local:** Applies the color to the vertices of every single character.
• **Use Alpha:** Shows where the color should include. For example if Alpha is enabled and the bottom right alpha is set to `0`, the characters fade out the bottom right.
• **Z Active:** If the Extrude modifier is used in addition to Color Per Vertex, the colors can be changed both in the front and back. If no such modifier is applied, only the colors in the front can be edited.
	

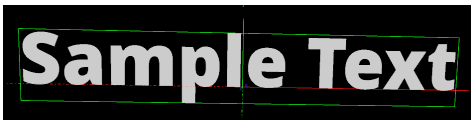
Supported Render modes: • Texture • Geometry	

- Click a circle to edit its color values.
- Click on a line that connects two circles to select both circles.

Contour



Use the Contour modifier to change the bold level of fontstyles, which makes it show thicker or thinner.

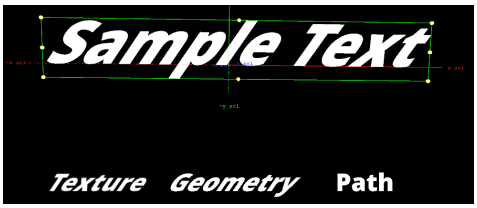
Options	Sample
• **Bold Level:** Defines the bold level of a fontstyle. The higher the value, the bolder the fontstyle.	Bold Level:1  Bold Level:100 
Supported Render modes: • Texture • Geometry	

Copyright © 2026 Vizrt

Italic



Use the Italic modifier to create a cursive appearance of a fontstyle.

Options	Sample
• **Level:** Defines the cursive level of a fontstyle. The higher the value, the more the text is tilted.	

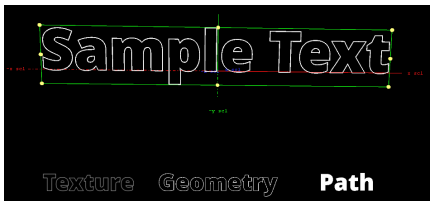
Supported Render modes:

- Texture
- Geometry

Outline



Use the Outline modifier to create an outline of a fontstyle.

Options	Sample
• **Outline Level:** Defines the width of the outline level. The higher the value, the bolder the outline of a fontstyle. • **Debug:** Draws the actual color points to see and understand how color changes effect the characters.	

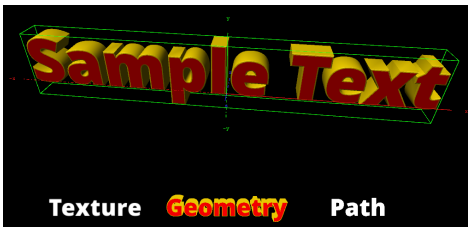
Supported Render modes:

- Texture
- Geometry

Extrude



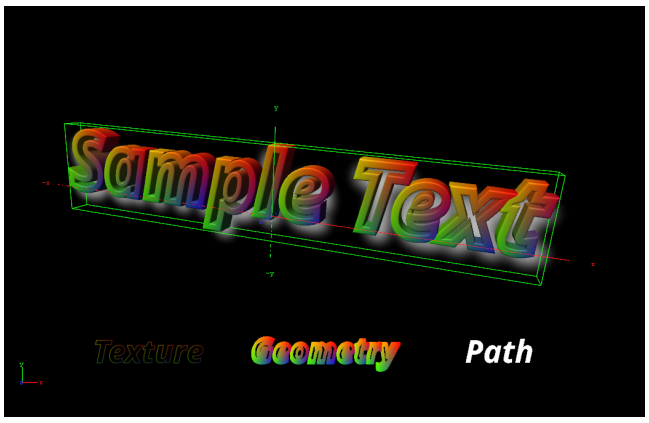
Copyright © 2026 Vizrt

Options	Sample
• **Extrusion Type:** Defines the bevel style of the extrusion. • **Extrusion Depth:** How strong the extrusion depth is applied. • **Bevel Size:** A multiplier to the bevel effect. • **Shading Angle:** Defines the angle the bevel is being drawn. • **Bevel Detail:** How precise the bevel effect is calculated (only available on rounded bevel types)	
Supported Render modes: • Geometry	

Information: To add a Classic Material to your font for a better visualization, right click on the font in the Font Style viewer, choose **Properties** and **Material.**

Fontstyle Viewer

Use the Fontstyle Viewer to visualize how the fontstyle looks when being used in scenes, the Fontstyle Viewer shows a preview of the fontstyle.



Additional to the customizable text in the Fontstyle Viewer, the words Texture, Geometry and Path are shown. These words show the modifications performed to Texture, Geometry and/or Path, applied by the modifiers, without reflecting the changes in the properties. For example, if kerning is applied to the customizable text, this is not reflected in the Texture, Geometry or Path texts.

The customizable text reflects either the Texture, Geometry or Path text type, which is selected in the Global Settings, available from the Modifier Parameters.

Right-click the Fontstyle Viewer to open a menu, similar to the menu in the Scene Editor (see Working with the Scene Editor).

Copyright © 2026 Vizrt

Text Properties

- **Text Editor:** To define a sample text and change spacing between words and letters. These settings together with the modifiers are saved into the new font.
- **Preview Options:** These settings help a designer to visualize the font, but they are not saved into the font.
- **Global Settings:** Shows various global properties for the fontstyle rendering.
    - **Font Default Style:** Defines the font default style. Options are *Geometry* or *Texture*.
    - **Shadow Default Style:** Defines the shadow default style. Options are *Geometry* or *Texture*.
    - **Texture Quality:** Defines the quality of the texture. Options are *Low*, *Normal*, or *High*.

See Also

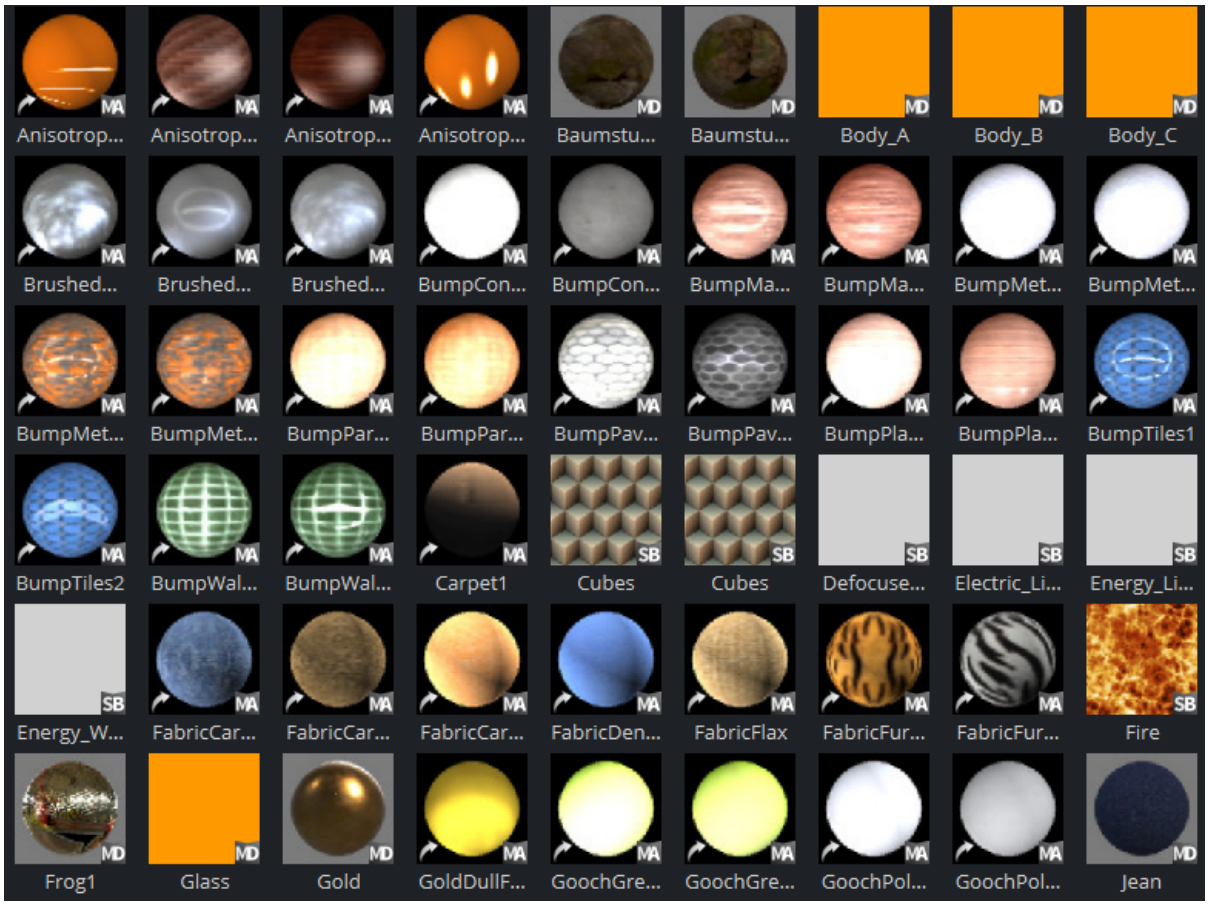
- Working with the Scene Editor
- Modifier Parameters
- Text Editor
- Working with Text and New Font System
- Classic Text Editor

Copyright © 2026 Vizrt

7.3.2 Working with Various Materials

A designer can work with various materials depending on the selected Render Pipeline. This section contains information on the following:

- Viz Engine Renderer Materials
  - Material Definitions
    - To Create a Material Definition
  - Working with PBR, Substance and Phong Materials
  - Viz Engine Shaders
- Classic Materials
  - Basic Materials
    - To Create a New Material
    - To Modify a Material
    - To Change a Material Color
  - Material Advanced
    - To Create a Material Advanced
    - To Edit a Material Advanced
  - Substances



Copyright © 2026 Vizrt

Viz Engine Renderer Materials

Material Definitions

Material Definitions represent the new materials for the Viz Engine Render Pipeline. These can be either PBR-based materials, Phong shaded materials, Reflections, Invisibles or Material Definitions created by plug-ins. The PBR approach defines the surface of an object by a physically correct description. PBR materials usually consist of various maps like a base color, normal map, roughness etc. Phong materials provide a modern approach as a replacement of the materials in the Classic Render Pipeline.

To Create a Material Definition

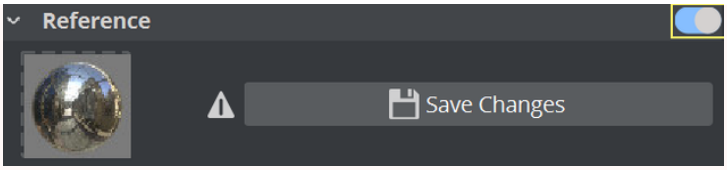
Once you've assigned textures and adjusted parameters for your Material Definitions, you can save it to Graphic Hub, by dragging and dropping it from the Texture Settings panel to the Server view:



Material Definitions stored on Graphic Hub can be used the same way as other items. Drag and drop them from the Server View to your geometry and the Material Definition is applied.

Important: If a material from the Database is used in multiple scenes and this materials is modifies, it affects **ALL** scenes using this material. To allow modification of references materials without affecting any other instance, you first have to break the reference and un-link it from the database. Do this by clicking the **Deactivate Reference** Button in the Material Settings:

Copyright © 2026 Vizrt



This option is only available on referenced Material Definitions.

Working with PBR, Substance and Phong Materials

For a more detailed description of various Material types in Viz Engine Renderer, see:

- Working with PBR Materials
- Working with Phong Materials

Viz Engine Shaders

Viz Engine Shaders are special types of material definitions. They can be either Vertex Shaders or a combined Vertex- and Fragment Shader.



To create a new Viz Engine Shader,

1. Open the **Workspace Panel** (1).
2. Click + Add.
3. Choose **Fragment Viz Engine Shader** or **Vertex Fragment Viz Engine Shader.** A new shader is added to your workspace.
4. Assign this shader to a container as Material Definition. (2).
5. Add Shader code using the **Shader Design Environment** (3).
6. Save your Viz Engine Shader to Graphic Hub by dragging it from the container (2) into the Asset view (4).

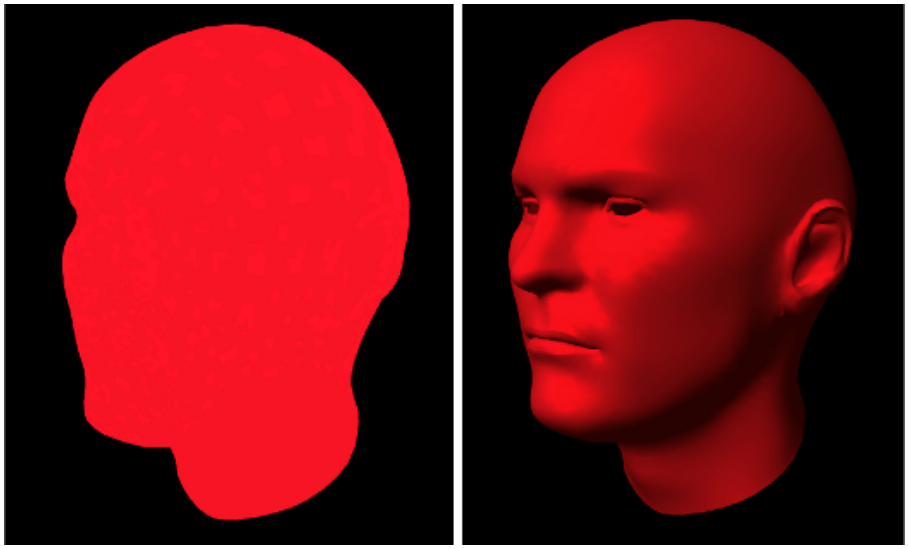
For detailed information on how to work with Viz Engine Shaders, please see Viz Engine Shader Techniques

Copyright © 2026 Vizrt

Classic Materials

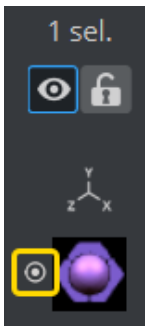
Basic Materials

Basic Materials are needed to a three-dimensional impression to any geometry. Without a material, each geometry appears flat:



- **Left:** Face shape without any material, appears flat.
- **Right:** Same geometry with material applied.

If the Material icon in the Property area is disabled it causes the object's base color to be shown (left picture). This is a color without any shading and it is not influenced by light, which means that it is visible even if there are no active light sources. Its initial color is created through a conversion of the material RGB components, but the values can be modified afterwards.

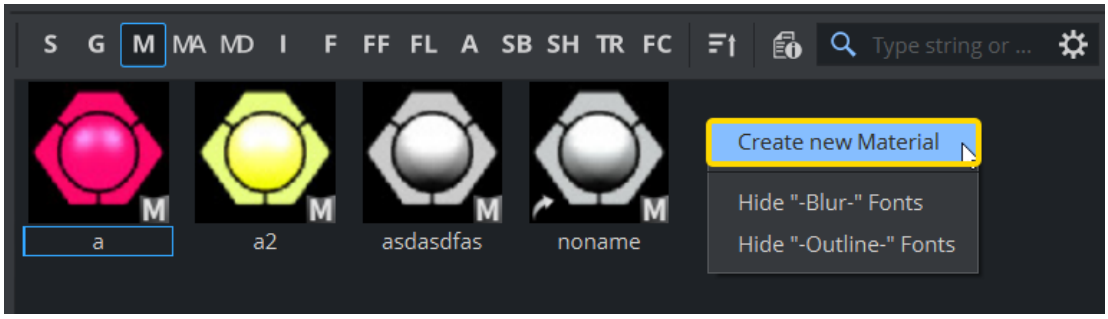


To Create a New Material

1. Click Asset View.
2. Click the **M** (Material) tab.

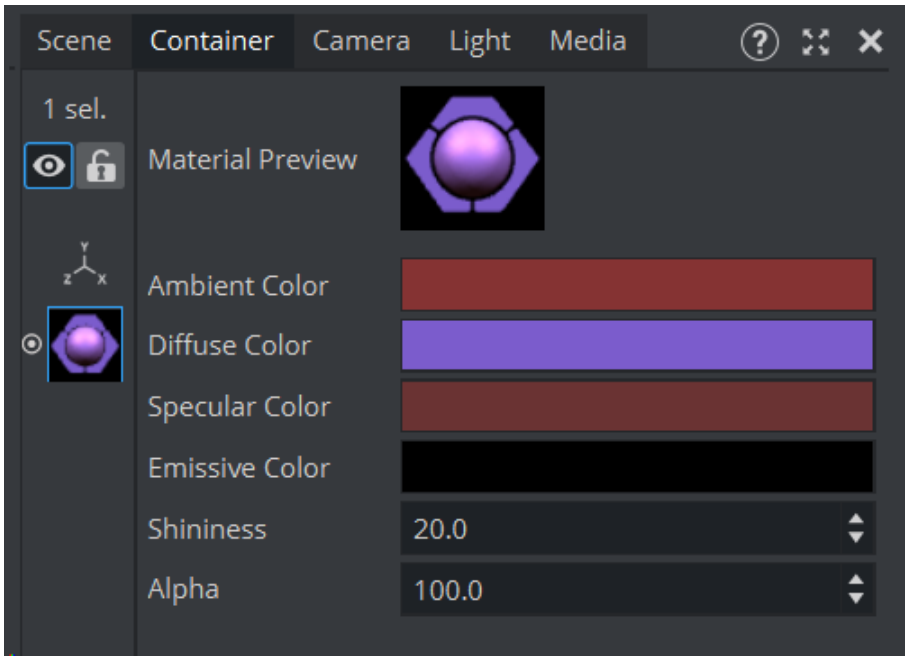
Copyright © 2026 Vizrt

3. Right-click and select **Create new Material**.
4. Enter a name for the new Material.
5. Click **OK**.



To Modify a Material

Materials define how a geometry appears in 3D space. It allows light to be added as well as information about the base color, specularity and more. When a Material has been added to a Container, the Container specific Material can be modified in a Material Editor that opens in the Properties Panel. The Material Editor is used to create and modify materials with customized colors and Lights effects.



The icon in the upper right corner of the Material Editor shows the current state of the Material. The color for a lit object is shown on the sphere within the icon, while the three parts around the sphere show the plain color for an object that is not lit.

- **Ambient:** The area which is not directly lit by any light source. Ambient light surrounds the object and the color looks the same on every side. Since the light does not come from any direction, but instead from *all* directions, the color/light spreads evenly over the object.

Copyright © 2026 Vizrt

- **Diffuse:** The object's *core color* observed under normal lighting conditions. Shows the color of an object when illuminated by a directional light source. It spreads evenly over the object. Light normally has some level of diffusion.
- **Specular:** Shows the color that shows when light comes from a particular direction and bounces back from the surface in a mirrored direction. The specular color is in fact the direct reflection of a directional light source. This is the typical effect of shiny metal and plastic. Specular color can be understood as a kind of shininess of color.
- **Emission:** Emissive color is produced by an object's natural glow. The object itself is the light source illuminating the color. The emissive color is unaffected by any other light sources and does not produce any light for other objects in the scene.
- **Shininess:** The shininess defines how spread out the highlighted area is. The lower the value, the more spread out the shine; the higher the value, the more focused the shine on a single point. However, if specular is black (all values `0`) then shininess does not have any visible effect.
- **Alpha:** The alpha value can be set directly on a material to adjust the level of transparency. Viz Artist uses schemes to describe the colors of a material.

To Change a Material Color



1. In the color selection area (1) select RGB or HSV.
2. In the color selection area, click a color bar, or the color circle, and drag to change the color.

Copyright © 2026 Vizrt

Tip: For RGB only: Right-click on the color sliders and drag to lock all three sliders, so that all values can be changed at once.

or

1. Click on the eye dropper icon in the color field.
2. Move the cursor to pick a color from anywhere on the screen.
3. Click to accept the new color.
   or
4. Click a numeric input box and enter a color numeric value.

Important: Materials are always stored within the scenes and are NOT referenced by Graphic Hub. Therefore, you are not able to search for referenced materials.

Material Advanced

Material Advanced consists of at least one basic material, a texture and usually some shader plug-in.

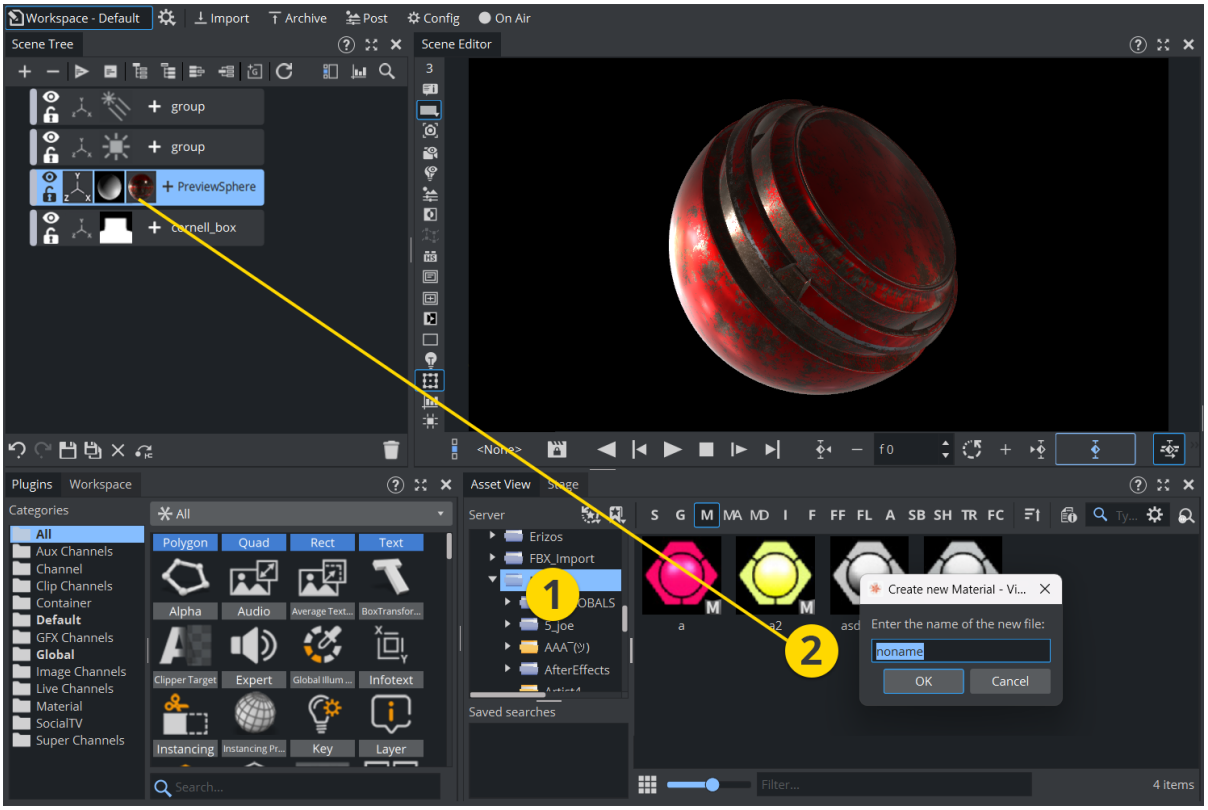
To Create a Material Advanced

1. From the Item List, drag a Material and/or an Image to a Container in the Scene Tree.
2. From the Plugins, drag a Shader to the same Container in the Scene Tree (if required).

Note: A combination of at least two items (Material, Shader or Image) must be applied to create a Material Advanced.

3. Right-click the Container in the Scene Tree.
4. Click **Merge to Material Advanced**. The Material, Shader, and Image icons in the Container are then merged into one combined Material Advanced icon.
5. Select the project or folder where the Material Advanced should be placed (1).

Copyright © 2026 Vizrt



6. Drag the Material Advanced icon from the Container in the Scene Tree to the Item List (2).
7. In the dialog box that opens, type a descriptive name for the Material Advanced and click the **Ok** button.

To Edit a Material Advanced

1. Select the project or folder where the existing Material Advanced is placed.
2. Drag the Material Advanced to a Container in the Scene Tree.
3. Right-click the Material Advanced icon on the Container.
4. In the context menu click **Split Material Advanced**. The Material Advanced icon on the Container is then replaced by separate icons for any Material, Shader or Image that the Material Advanced consists of.
5. Apply new items or modify the Material, Shader or Image settings.
6. Right-click the Container in the Scene Tree.
7. In the context menu, click **Merge to Material Advanced**. The Material, Shader and Image icons in the Container are merged into one combined Material Advanced icon.
8. Select the project or folder where the Material Advanced should be placed (1).
9. Drag the Material Advanced icon from the Container in the Scene Tree to the Item List (2).
10. In the dialog box that opens, type a descriptive name for the Material Advanced and click the **Ok** button.

Note: Material Advanced are used in the Viz Classic Render Pipeline only!

Substances

Substances are usually created in Substance Designer and imported to be used within Viz Engine.

Substances use two different file extensions, *.sbs* and *.sbsar*:

- *.sbs* files are editable substance source files. Use Substance Designer with these files.

Copyright © 2026 Vizrt

- *.sbsar* files are performance optimized, published substances. You can import these files to Viz Artist. You cannot edit these files. To create a *.sbsar* file, first create a shader with Substance Designer, and then publish this shader.

After importing a substance to Viz Artist, they can be used within the Substance Shader. Drag and drop the Substance Shader (**Plugins > Shader > Texture**) onto a geometry. Then locate your imported substance assets in Graphic Hub and drag it to the Substance Field.

For details about the various options, please refer to the **Substance plug-in** Description.

Tip: The import and use of Substances in Viz requires a separate license. You can obtain this license from your local Vizrt representative.

Copyright © 2026 Vizrt

7.3.3 Working with Geometry Items

- To Merge Scene Trees as Geometries
  - Limitations
- Referenced and InScene geometries
  - Referenced (active link to Graphic Hub)
  - InScene (copy inside the scene)
- Geometry Editor Properties
  - Working with Referenced Geometries

Geometries are stored in the Graphic Hub database and are available through the Asset View.

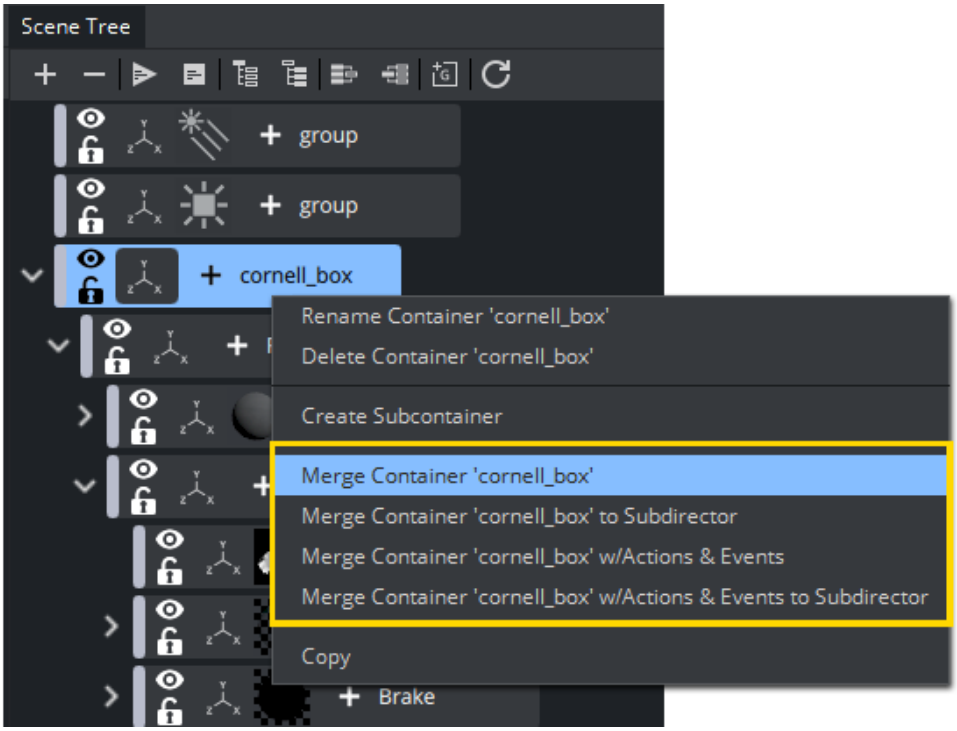


There are two ways to generate geometries:

- A geometry can either be imported from an external modeling program (See Import of Files and Archives).
- A part of the scene can be merged as geometry and saved to Graphic Hub.

To Merge Scene Trees as Geometries

A part of the scene tree can be merged and saved as geometry. This can be used for example to copy parts of a scene to a new scene.

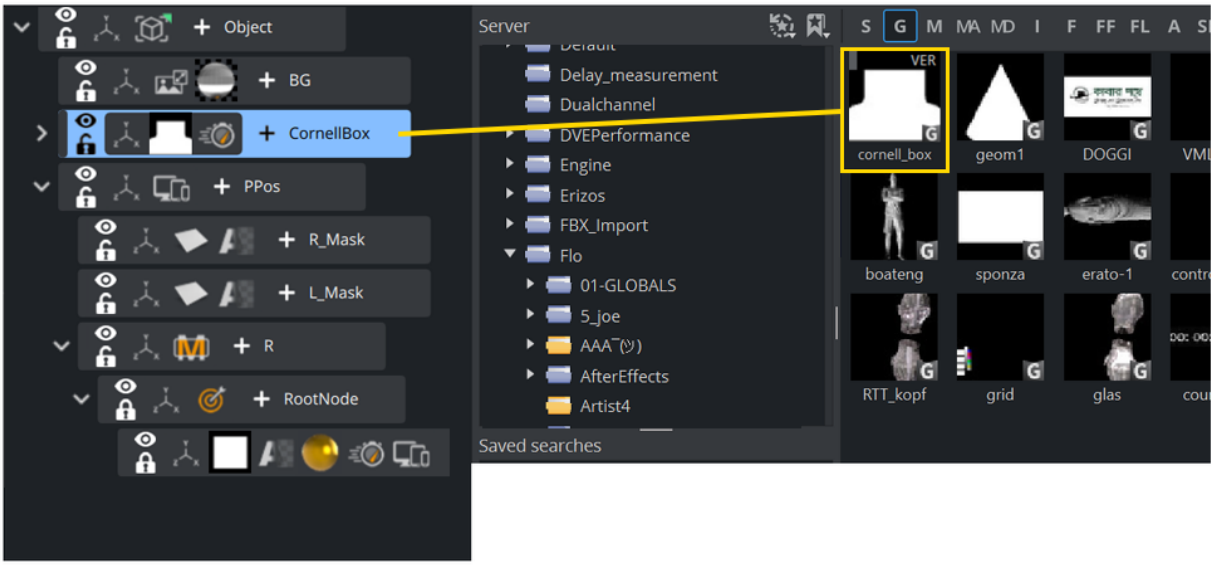


Right click on the part of the scene tree that needs to be merged. From the context menu, select **Merge Container...** which type you need:

- **Merge Container:** Merges the selected Container and all its Sub-Containers into one compound object, or extracts an already merged object.
- **Merge Container to Sub Container:** Merges the selected Container(s) and all the respective Sub-Containers into one compound object and creates a new sub director for the merged animation.
- **Merge Container w/ Actions & Events to Sub Director:** Merges the selected Container(s) and all the respective Sub-Containers into one compound object, including actions and events and creates a new sub director for the merged animation.

This merges anything underneath the selected container into a single geometry. This geometry can be dragged and saved to Graphic Hub.

Copyright © 2026 Vizrt



The geometry can be used in another scene by dragging into the scene tree and splitting it up again.

Info: To differentiate if a geometry has been created from a Classic Render Pipeline Scene (CLASSIC) or the modern Viz Engine Render Pipeline (VER), a label indicates the Render Pipeline used.

A note on TextureRenderer and Runtime Shaders:
TextureRenderes and Runtime Shaders need to be exported alongside merged geometries / subscenes to be able to restore their state properly when importing again. The location where they will be placed along the merged geometry can be set in the configuration file.

```
DependencyLocationMode = SUBFOLDER/SAMEFOLDER/RELATIVEPATH
```

This will place TextureRenderers and Runtime Shaders in the chosen default location.

Limitations

Merged geometries can be generated automatically for Transition Logic scenes if the flag `AutoExportTransitionLogicGeometries = 1` is enabled. The new subscene workflow is recommended instead.

Referenced and InScene geometries

An imported or merged geometry in Viz can either stay **referenced**, linked to a single geometry object in Graphic Hub, or be **deactivated (local)**, which turns it into its own copy stored inside the scene. This only applies to Graphic Hub geometries such as imports and merges. A built-in plugin like Rectangle or Cube has no reference to manage.

Copyright © 2026 Vizrt

Referenced (active link to Graphic Hub)

The scene stores only a pointer, by UUID, to the geometry object. Changes to that object reach every scene that uses it (source).

Pro	Con
One source for many scenes: fix a mesh once (for example the NFL helmet, a logo or a stadium) and every template picks up the change.	**Changes spread everywhere:** an edit to the Graphic Hub object hits every scene that uses it, including on-air ones you didn't mean to touch.
Lighter scene files: the mesh data isn't duplicated in each scene.	**Depends on Graphic Hub:** the scene needs that geometry UUID to be present and resolvable. If you move scenes between Graphic Hubs without their dependencies, you get missing geometry.
Reusable library: a merged compound can be dragged into Graphic Hub and reused anywhere (source).	**Limited editing: Face Orientation** and **Shading** can only be edited once the geometry is deactivated (source).

See also: Working with Geometry Items

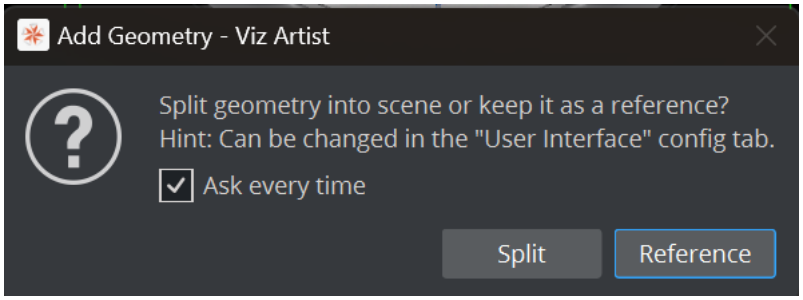
InScene (copy inside the scene)

Deactivating creates a unique copy for that scene and breaks the link to the database (source).

Pro	Con
Isolated: later changes in Graphic Hub can't break or change this scene, which is safer for a locked show package.	**Duplicated data:** each scene carries its own copy, so files are bigger. That matters with heavy meshes like a scan of about 280k faces.
Fully editable: you can flip normals and adjust shading. This is often needed on imported FBX or glTF meshes.	**Drift:** copies in different scenes slowly diverge, and a fix has to be repeated in every scene.
Self-contained: fewer external geometry dependencies to carry along when deploying.	**Splitting deactivates for you:** splitting a merged geometry automatically breaks the reference. You have to merge it again before you can use Re-establish or Update (source).

Viz Artist now will ask if one drags a Geometry from Graphic Hub into the scene tree or scene editor.

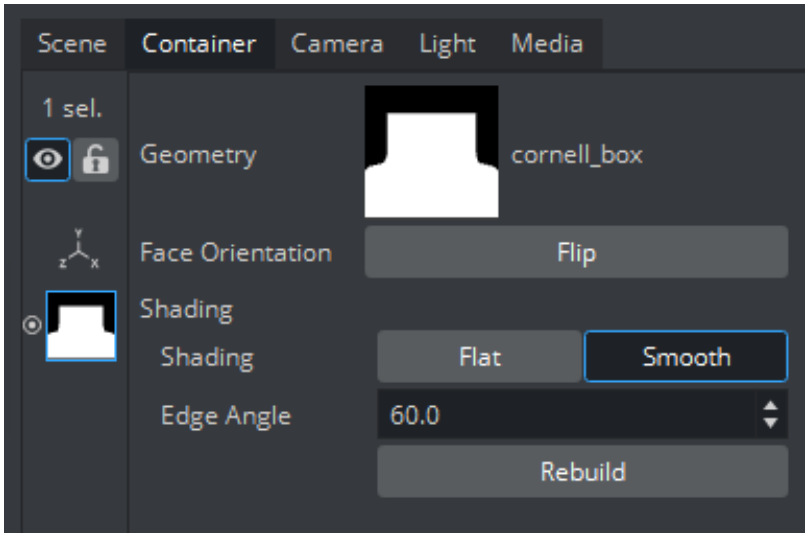
Copyright © 2026 Vizrt



This behavior can be changed in the Configuration/user Interface/Geometry Drop Mode

Geometry Editor Properties

Geometries can be modified by clicking on the Geometry editor:

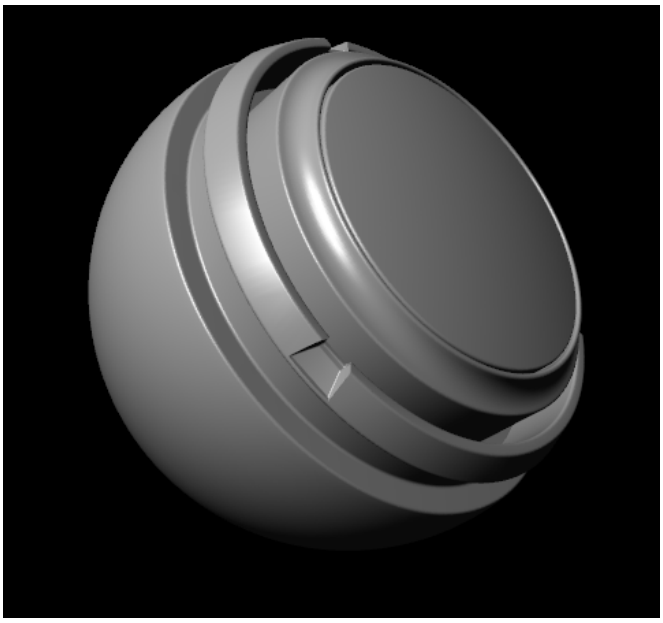
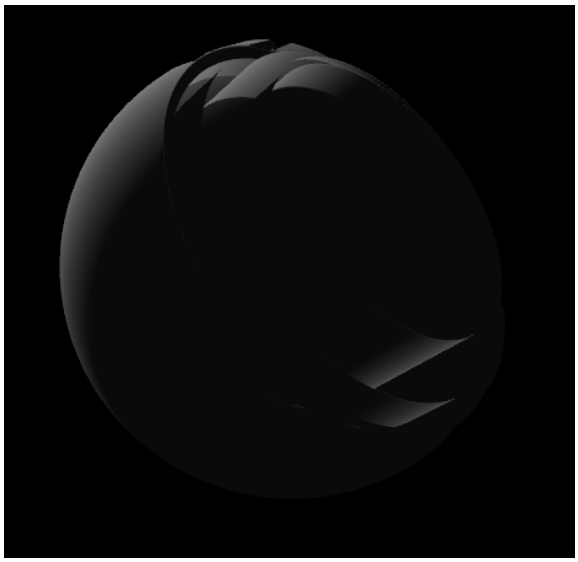


- **Geometry:** Shows a thumbnail of the Geometry, and also its name. Click the thumbnail to select the Geometry in the database.

Note: This can only be done when the reference to the database is active.

- **Face Orientation:** After importing Geometries that have been created by external 3D programs, it might happen that the polygons face orientation is wrong. By clicking the **Flip** button, the polygons are set correctly.

Copyright © 2026 Vizrt

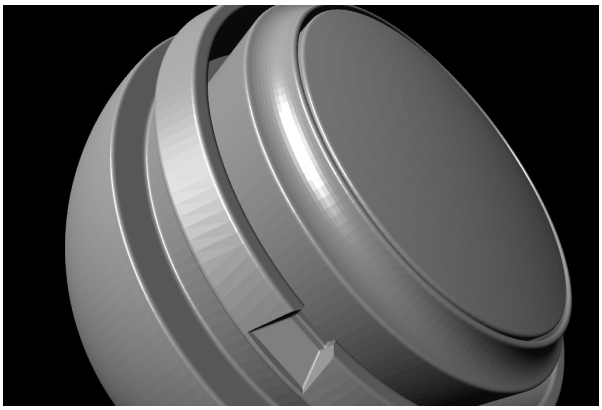
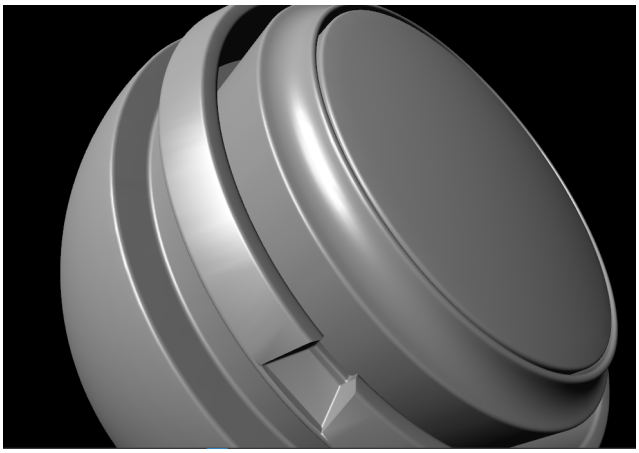
Correct Face Orientation	Incorrect Faces
	

Try to use the Expert plug-in and **Back Face** and **Twosided Lighting** if there are still rendering issues.

Tip: In the Classic Render Pipeline, you can show the vertices orientation to visualize the orientation.

- **Shading:** Imported polygon Geometries normally use shading for creating a smooth surface on the object, which is set together by a planar polygon mesh. The shading value in the Geometry editor sets a threshold between which edges are to be smoothed by shading and which are to be kept as sharp edges. All edges with an angle smaller than the defined value are smoothed by shading. Shading in Viz Artist is performed in the following way: A shade value is produced at each vertex of the polygonal mesh representing the surface of the object. This is done by creating a surface normal on each polygonal facet. The surface normal at each vertex is created by averaging the surface normals for the surrounding facets. The shade at each vertex can then be calculated. Once the shade at the vertices of the polygonal mesh are known, the shade at points interior to the polygonal facets are interpolated from the values at the vertices. This technique makes curved surfaces look shaded smoothly even though based on a representation of planar facets. The interpolation of shade values is incorporated into the polygon scan conversion routine. Hence, an increase in realism is obtained at far less performance cost than carrying out a pixel-by-pixel shading calculation over the whole original surface. Shading may be set to *Flat*, which means that all shading is disabled, or *Smooth*, which enables the *Edge Angle* option. After adjusting the *Edge Angle*, make sure to click the **Rebuild** button for the changes to take effect.

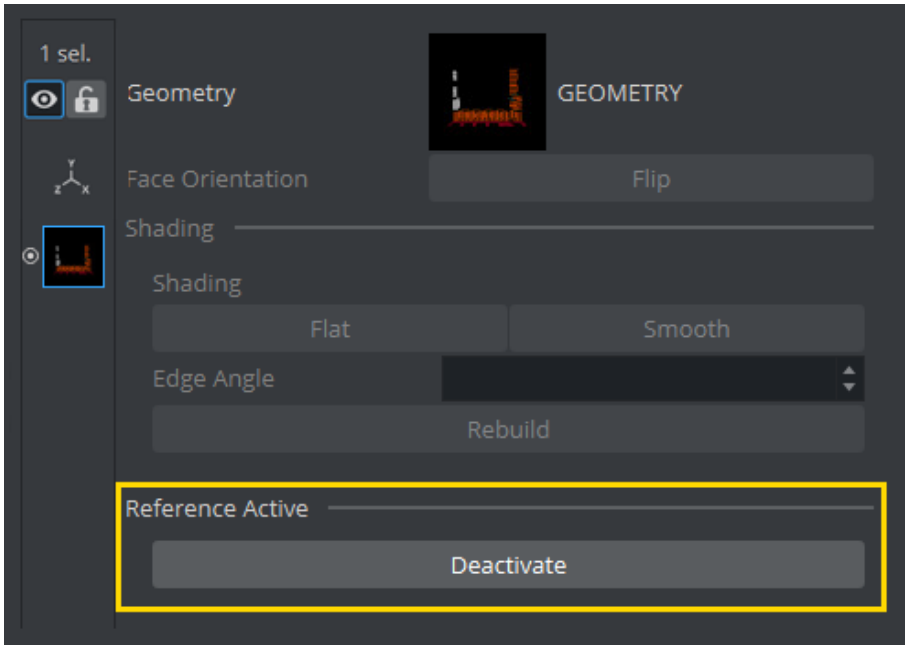
Copyright © 2026 Vizrt

Flat Shading	Smooth Shading
	

Note: Face Orientation and Shading are only applicable if the reference is deactivated.

Working with Referenced Geometries

If Geometries stored in Graphic Hub are used in other Scenes, they still keep their reference. This means, if the Geometry on Graphic Hub is modified, it also affects all scenes using this Geometry. To break the reference, click on the Geometry and choose **Deactivate.** The geometry is then part of the scene and is stored within the scene.



Reference: If using an imported or compound object in a Scene, there is a reference between the object in the Scene and the object in the database. If changes are applied to the object in the Scene, the object in the database

Copyright © 2026 Vizrt

including any Scenes where this object is used with an established reference, it reflects those changes, as it is the same object.

- **Deactivate:** Creates a unique copy of the object for the Scene. The reference link must be deactivated. Changes that are applied to a deactivated object do not lead to changes in the original object in the database.
- **Re-establish:** Reactivates the reference and overwrites the object in the Scene with the object in the database. An active reference is practical when creating a set of Scenes with the same object. If for instance, all the Scenes should have the same background and the background item is created as an object with active references, changes can easily be applied to all the Scenes that use that object.
- **Update:** Reactivates the reference and overwrites the object in the database with the object in the Scene.
- **Status:** Shows the current status of the reference.



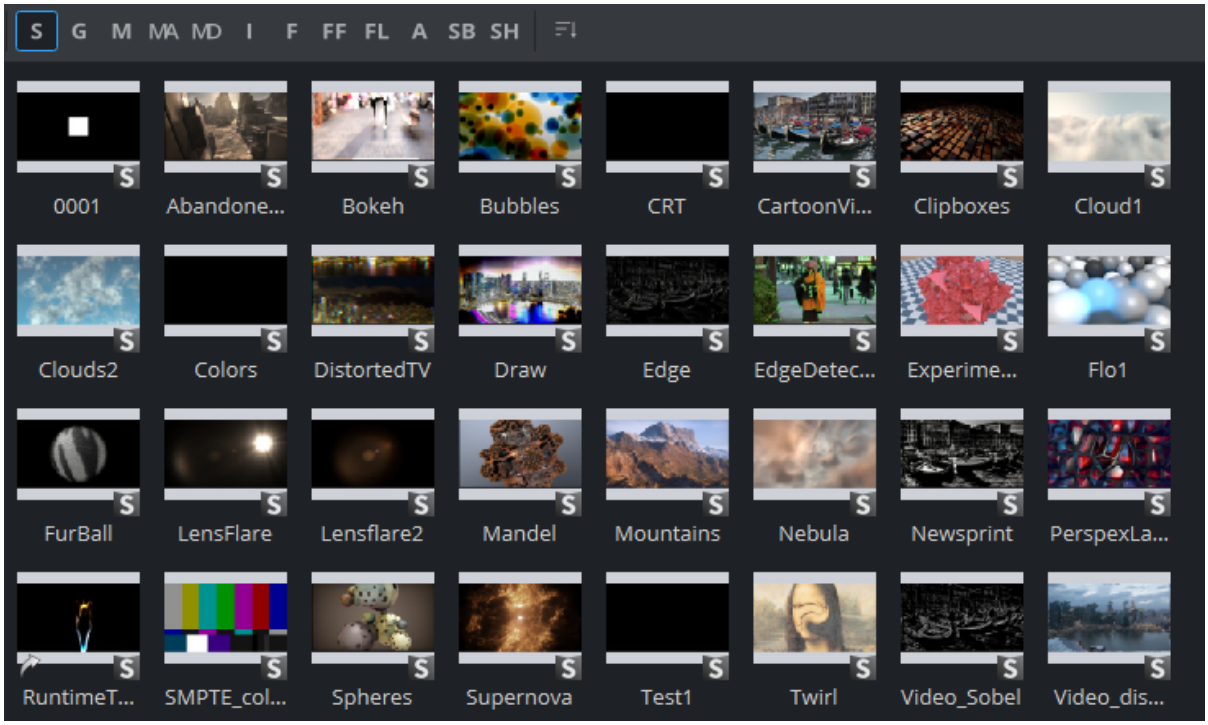
Note: The reference is automatically deactivated if splitting the object. If the object is split, it must first be merged before the Re-establish and Update operations can be performed.

Copyright © 2026 Vizrt

7.3.4 Working with Scene Items

Scenes

Scenes are root items that hold other items. Scenes are stored in the Graphic Hub database and are available through the Asset View. Scenes are the base for further use as templates within control applications like Viz Trio or Viz Pilot.



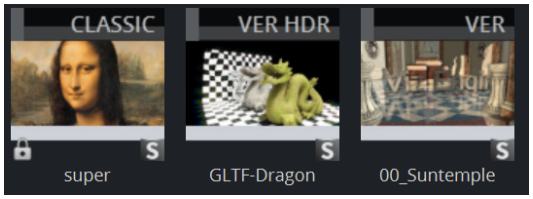
If a Scene is opened, the Scene Tree area shows the logical visualization of all the objects within the Scene. The Scene Tree consists of Containers that hold items and properties, organized in a hierarchy and grouped in logical divisions. The items and properties can be Geometries, Images, Materials, Fonts and many more (see Asset View and Plug-ins).

Scene can be rendered in two different render pipelines:

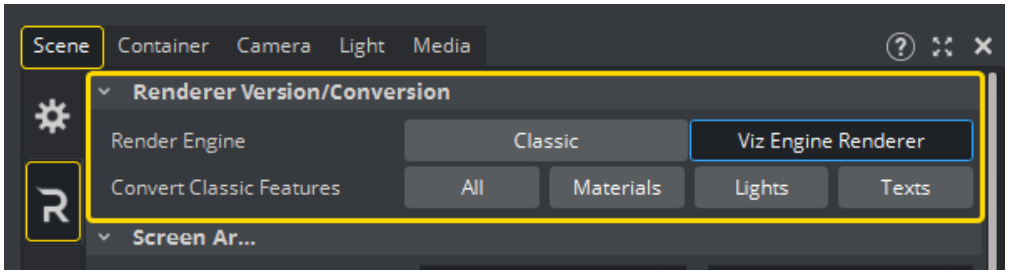
- **Classic Render Pipeline:** For backward compatibility of Viz 3.x scenes.
- **Viz Engine Render Pipeline:** To take advantage of new rendering possibilities.

To differentiate in the Asset views, a label indicates if the scene has been created either in the Classic Render Pipeline or the Viz Engine Render Pipeline. In the Viz Engine Render Pipeline, an additional *HDR* marks HDR enabled scenes:

Copyright © 2026 Vizrt



The Render Pipeline used can be selected in the scene settings:



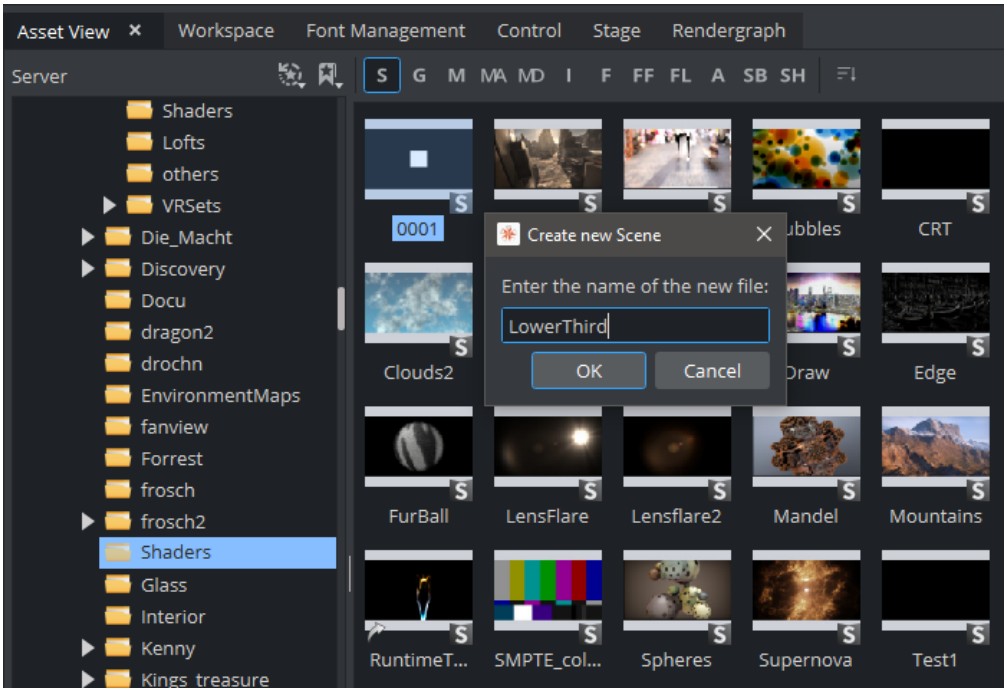
Information: It is recommended to create a copy of a scene in one pipeline and adapt them to other pipeline rather than try to use the same scene for both render pipelines.

To Create a New Scene

This creates an empty new Scene. Add containers as required in the scene tree.

1. Switch to Scenes in the Content switcher.
2. Select the folder/project where the new Scene should go.
3. Right-click in the Item List. and select **Create new Scene.**
4. In the Create New Scene panel enter a valid name for the new Scene.
5. Click on **OK**.

Copyright © 2026 Vizrt



To Import a Scene

For information about how to import a Scene (or an Archive) from a hard disk or network share to the database see Import, Post and Archive.

To Save a Scene

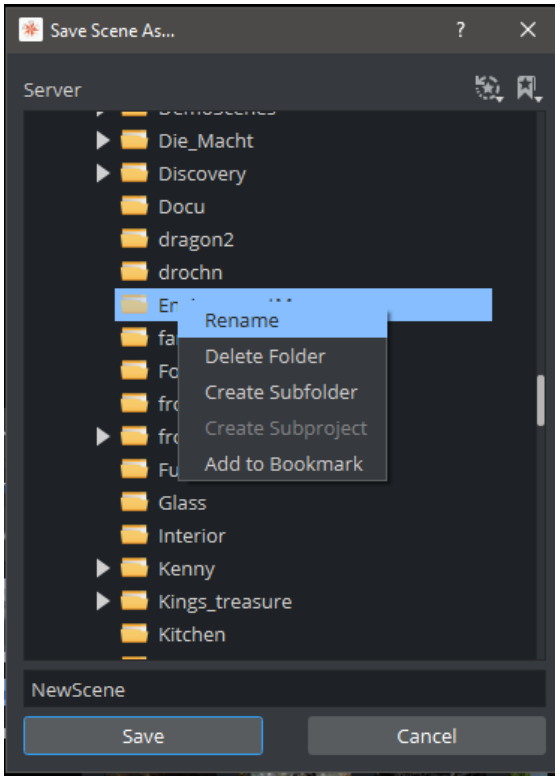
Click the **Save** button in the lower left corner of the window.

To Save a Scene with a New Name

1. Click the **Save As** button in the lower left corner of the window.
2. In the **Scene Save As...** dialog box that opens, type a descriptive name in the **Scene Name** text box.
3. Click the **Ok** button to confirm.

Copyright © 2026 Vizrt

Scene Save as... Panel Context Menu



Right-click in the **Scene Save as** panel to open its context menu. Select from:

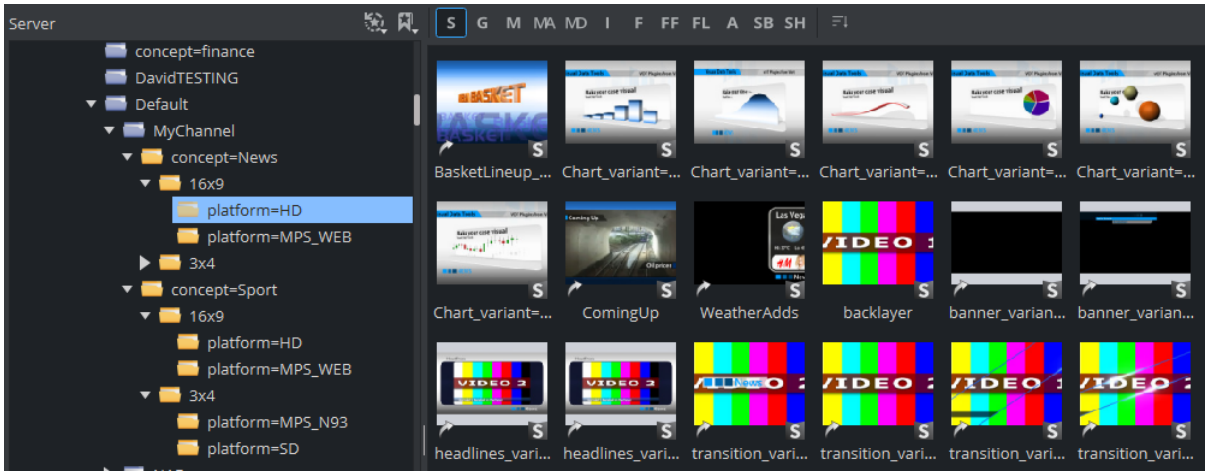
- **Rename:** Renames the selected folder.
- **Delete:** Deletes the selected folder.
- **Create Sub Project:** Creates a new sub Project folder in the selected Project folder.
- **Create Sub Folder:** Creates a new sub folder in the selected folder.
- **Add to Bookmark:** Adds selected folder to Bookmarks.

Scenes for Context-enabled Shows

A context-enabled show is a Viz Trio specific feature that allows the Viz Trio operator to switch between concepts and variants within a show. For example, a News concept could be switched to a Sports concept, depending on the content being taken to air.

When designing context-enabled shows in Viz Artist, each scene must follow a set of naming conventions. All variants of a Scene within a specific concept appear as one *template* in the Viz Trio **Template List**. Please refer to the Viz Trio User Guide for further information on Viz Trio templates.

Copyright © 2026 Vizrt



For a show to successfully use context-enabled Scenes in Viz Trio, the folders and Scene names must follow a naming convention that gives them properties and values for Viz Trio to recognize and use:

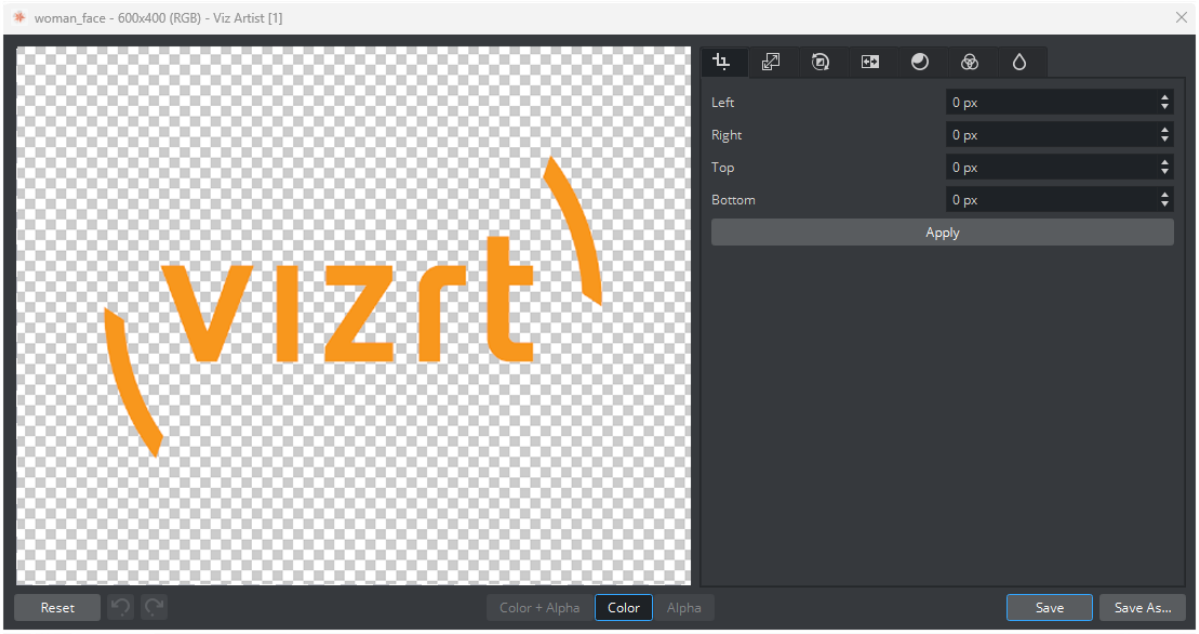
Type	Property		Name	Example
Concept (folder)	concept	=	`<conceptname>`	concept= `<conceptname>` concept=News concept=Sport
Context (folder) User-defined	<context>	=	`<contextname>`	`<context>` = `<contextname>` platform=HD platform=SD
Variant (file)	_variant	=	`<variantname>`	`<scene>` _variant= `<variantname>` chart_variant=Stock headlines_variant=rotation headlines_variant=simple weather_variant=3Days

In addition, to successfully create context-enabled show-designs, make sure that each scene variant is using the same containers, container names and plug-ins.

IMPORTANT! For the variants it is important to add an underscore (_) to separate the Scene name from the variant keyword.

Copyright © 2026 Vizrt

7.3.5 Working with Image Items



Images are essential for any designer. They can be used either as textures on 3D objects, part of a PBR material, as regular image, environment maps and a lot more. Images from Graphic Hub already have mipmap levels included and can be directly uploaded to the GPU. Using external images from the file system, via HTTP(S) or other locations should be avoided.

This section contains information on the following procedures:

- Terminology
- To Add an Image to the Graphic Hub Database
- To Use Images from Graphic Hub Image Library Extension (ImEx)
- To Export Images
- Converting and Assigning Color Spaces
- Editing Images
- Internal Image Editor
    - Crop and Expand
    - Scale/Stretch
    - Flip and Rotate
    - Color Inversion
    - Brightness, Hue, and Saturation
    - Format
    - Blur
    - To Apply an Effect to an Image
    - Image Preview Modes
- External Image Editor

Copyright © 2026 Vizrt

Terminology

- An **image** is the file stored on Graphic Hub. They can be imported, exported and modified using the Image Editor.
- A **texture** is an image uploaded into the GPU memory. It can be mapped as a texture, used by a PBR Material and is being modified using the Texture Editor.

To Add an Image to the Graphic Hub Database

- **Import Images:** Images can be imported to the GH database (see Import of Files and Archives) by using the Import View or by simply dragging them to the Asset view.
- **Create a Snapshot:** Images can be created by taking a Scene Editor Snapshot of the Scene Editor.
- **Image Editor:** In the Asset View, open an Image in the Image Editor (double-click the Image), modify (if required) and select **Save as...**.

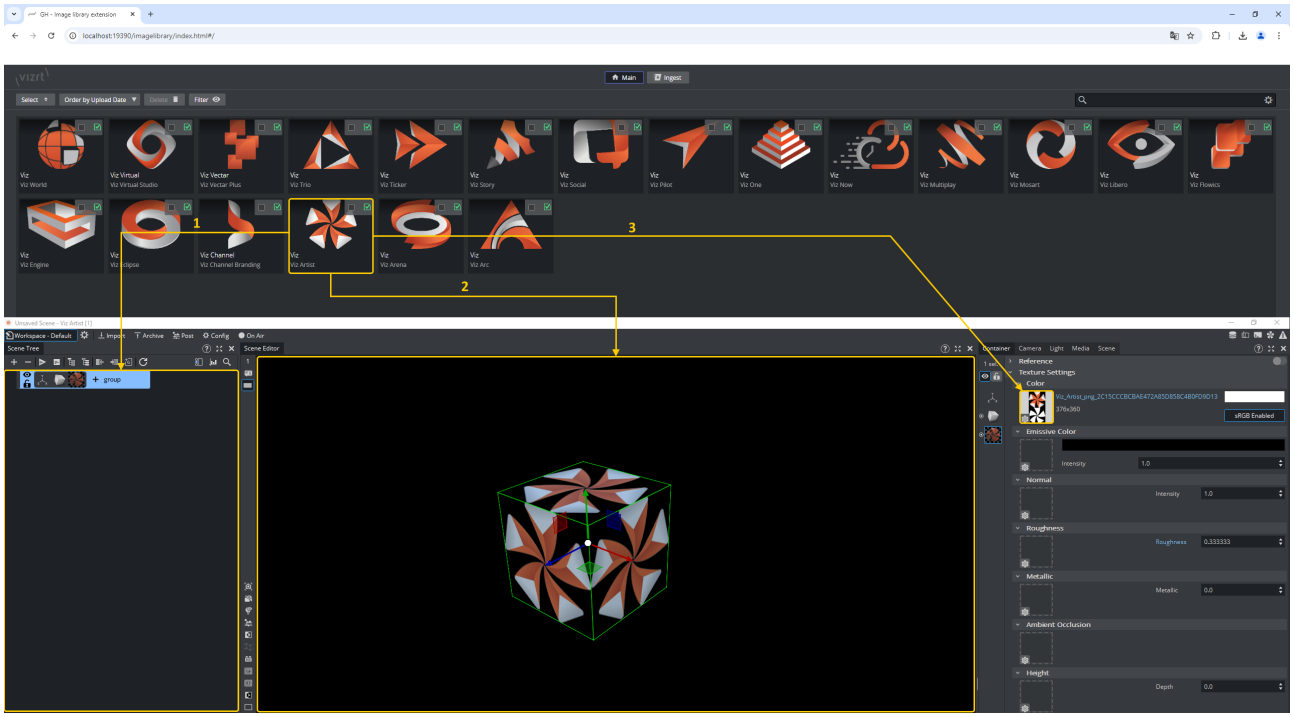
Information: Even if it is possible to use images from other locations like hard drives or even network locations, it is recommended to use images stored on the Graphic Hub. These assets are already prepared to be used in a real time environment.

To Use Images from Graphic Hub Image Library Extension (ImEx)

Images stored in Graphic Hub ImEx can be directly used in Artist. For this drag and drop the image from ImEx into Artist.
The following locations can be used with images from ImEx:

1. **Scene Tree**
2. **Scene Editor**
3. **Material Channels (PBR / Phong)**

Copyright © 2026 Vizrt



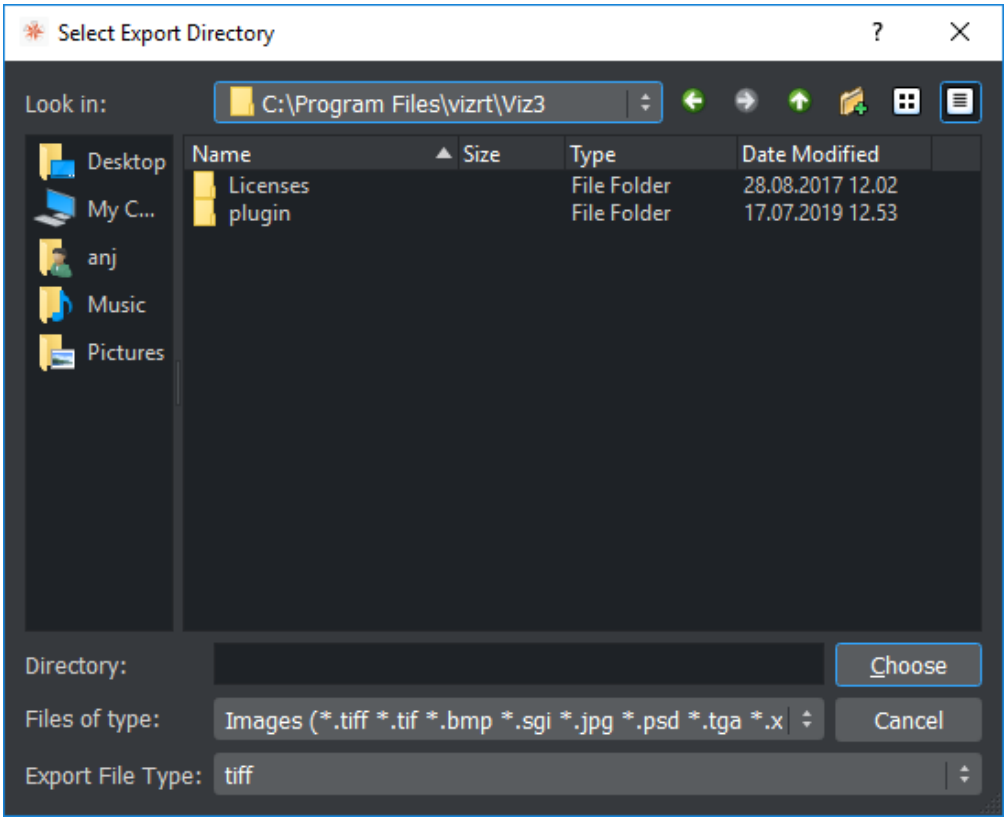
Note: Drag and drop from Graphic Hub ImEx is only supported if Viz Artist was configured to run on the same Graphic Hub database as the ImEx agent.

To Export Images

You can export one or multiple images. Exporting RGB images with alpha creates RGBA image files.

1. Select the image(s) to be exported.
2. Open the **Select Export Directory** panel and select the folder where the image should be exported to.
    - Select **Export...** from the Asset View, or (when only one image is selected).
    - Right-click the image to be exported. In the menu that opens, select **Export**...

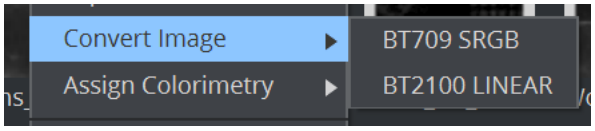
Copyright © 2026 Vizrt



3. Select an image format for export.

Note: The exported image retains its original name.

Converting and Assigning Color Spaces

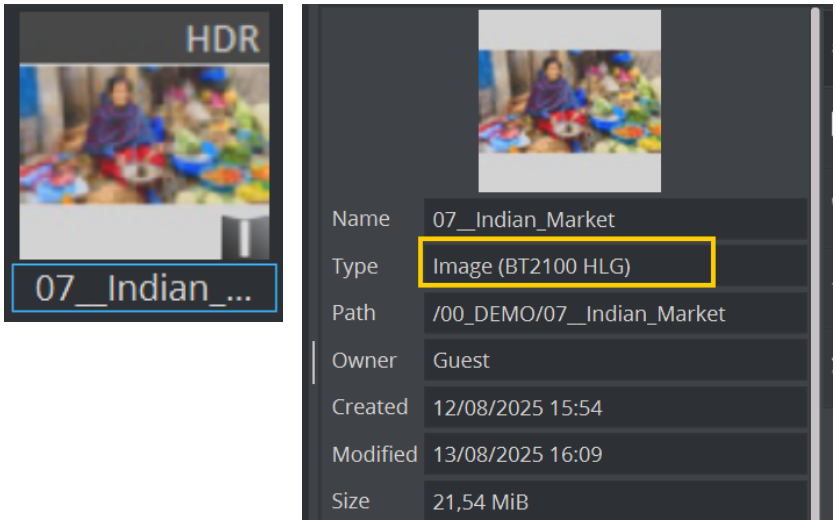


To use images in a certain color space (as BT709 SRGB or BT2100), they need to be mapped into that color space.

You can convert them from one color colorimetry to another, by right clicking and choosing **Convert Image** (just like Photoshop`s Convert to Profile). This maps the existing colors to the new profile.

Assign a new colorimetry with **Assign Colorimetry** (this does not change any colors).

Copyright © 2026 Vizrt



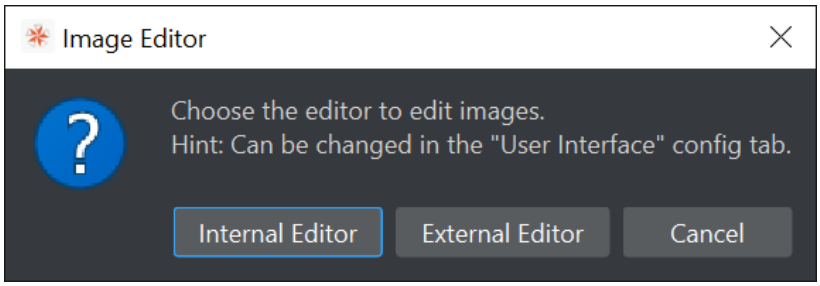
Images for HDR color spaces are labeled in the asset view, and details are shown in the file properties.

Editing Images

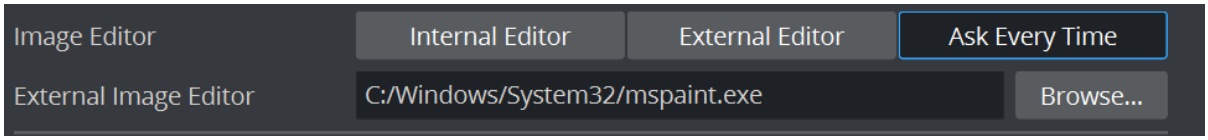
Double-click the image in the Asset View to open the image in the internal or external Image editor. If the image is referenced in a Scene, a dialog box shows to confirm the action.

> **Note:** Items that are opened in an editor, are locked in the database.

If this is done the first time, a Dialog pops up to ask if images should generally be opened in the external or internal Image Editor. This is saved as the default double click action for images.



This can be changed in the User Interface config page.



- **Image Editor**
  - **Internal Editor:** Opens the image in the integrated image editor when double clicked.

Copyright © 2026 Vizrt

    - **External Editor:** Opens the image in the external image editor (either the selected program in the *External Image Editor* setting or not set, the default Windows image viewer) when double clicked.
    - **Ask Every Time:** Opens a dialog asking for the editor to use when an image is double clicked.
- **External Image Editor:** Determines the filepath to the external image editor.

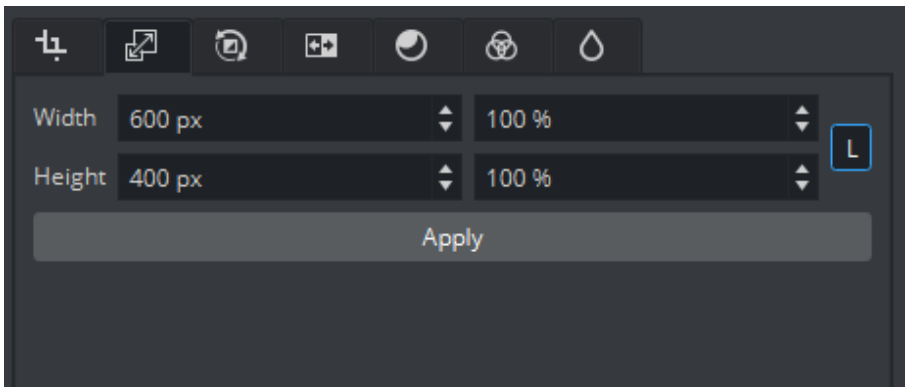
Internal Image Editor

The Image Editor makes it possible to modify image items in the Graphic Hub database. The modifications that can be done in the image editor are similar to the ones in other image editing software.

The Image Editor consists of:

- Main Image Area
- Effects
- Effect Modifications

Crop and Expand

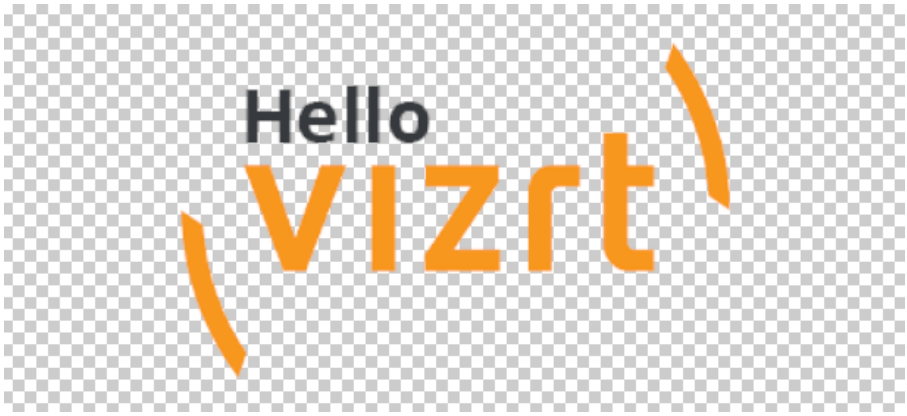


- **Crop and Expand:** Crops the image according to the values in the input boxes. The values can be entered manually.
    - **Left:** Crops the image the specified number of pixels from the left side of the image border.
    - **Right:** Crops the image the specified number of pixels from the right side of the image border.
    - **Top:** Crops the image the specified number of pixels from the top of the image border.
    - **Bottom:** Crops the image the specified number of pixels from the bottom of the image border.

Tip: Use negative values to expand the image beyond the original size.

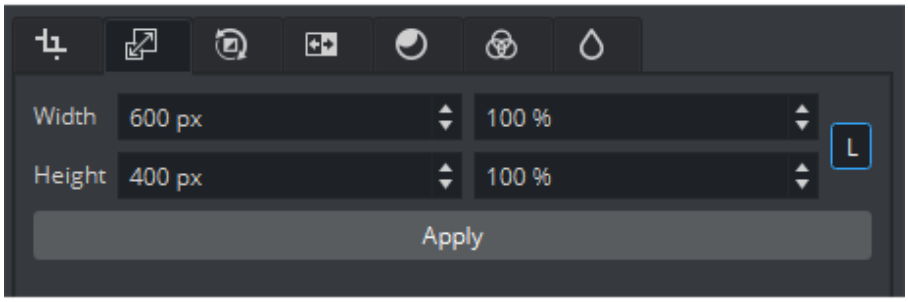
This can also be done be dragging the dotted lines.

Copyright © 2026 Vizrt



Scale/Stretch

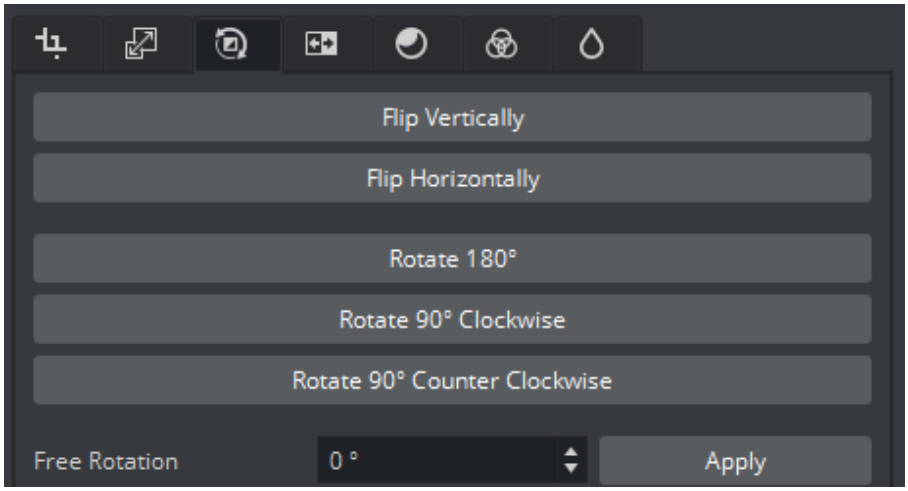
The image can be scaled or stretched depending on the state of the Lock button.



Flip and Rotate

- **Flip**
  - **Flip Vertically:** Flips the image upside down. To flip it back to its original settings, click the button again.
  - **Flip Horizontally:** Flips the image horizontally as if seen in reverse/through a mirror. To flip it back to its original settings, click the button again.
- **Rotate**
  - **Rotate 180°:** Rotates the image by 180 degrees.
  - **Rotate 90° Clockwise:** Rotates the image by 90 degrees in clockwise direction.
  - **Rotate 90° Counter Clockwise:** Rotates the image by 90 degrees in counter clockwise direction.
  - **Free Rotation:** Rotates the image by the entered number of degrees.

Copyright © 2026 Vizrt



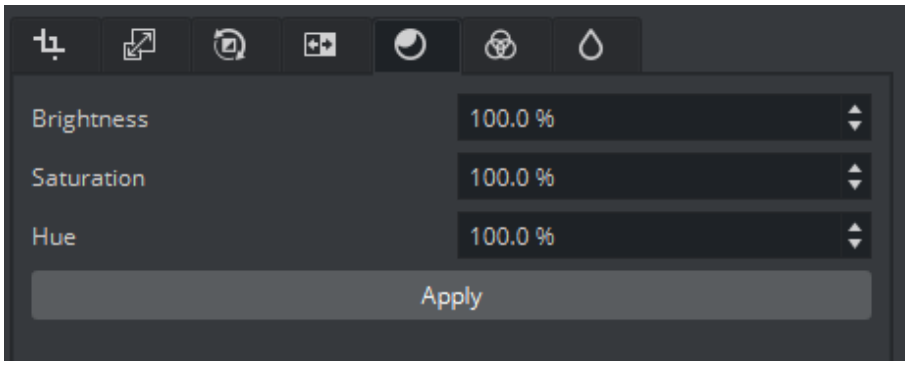
Color Inversion

Inverts the colors of an image. When inverting an image, the brightness value of each pixel in the channels is converted to the inverse value on the 256-step color values scale.

- **Invert RGB:** Inverts the RGB values.
- **Invert Alpha:** Inverts the Alpha values.
- **Invert RGB and Alpha:** Inverts the RGB and Alpha values.

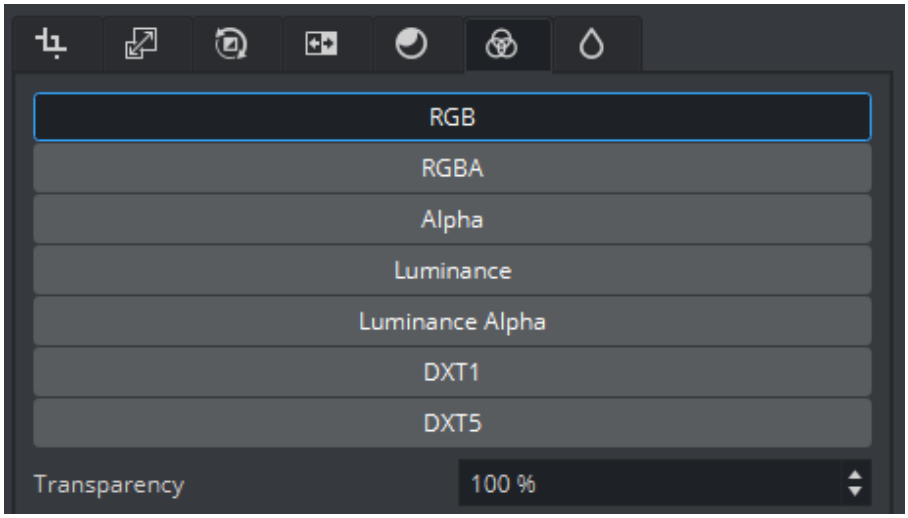
Brightness, Hue, and Saturation

Adjusts the brightness, saturation, and hue settings for the image.



Copyright © 2026 Vizrt

Format

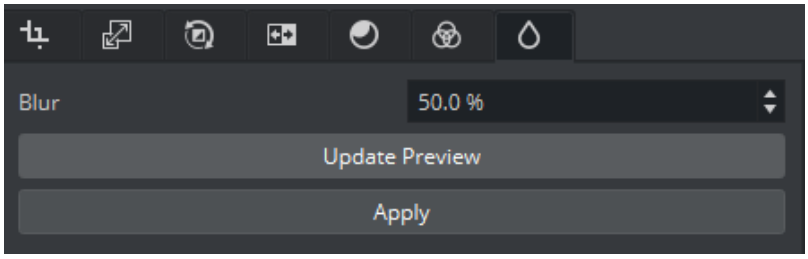


- **Convert Format:** Changes the format of the image to one of the following:
  - **RGB**
  - **RGBA**
  - **Alpha**
  - **Luminance**
  - **Luminance Alpha**

Note: All images are treated as RGB or alpha format. An imported image that has alpha can be viewed as either RGB or alpha. Modifications to an image are automatically applied to the RGB or alpha information.

Transparency allows to change the level of transparency if the image has an alpha channel (for example, RGBA).

Blur



- Sets the image blur intensity.

Blur and Sharpen can also be applied in Realtime. Open the image settings e.g. in the PBR Editor and set the Blur and/or Sharpness.

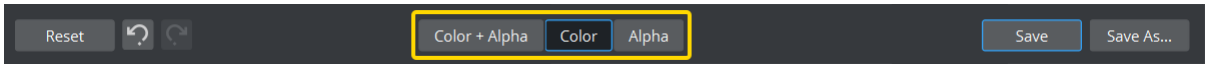
Copyright © 2026 Vizrt

To Apply an Effect to an Image

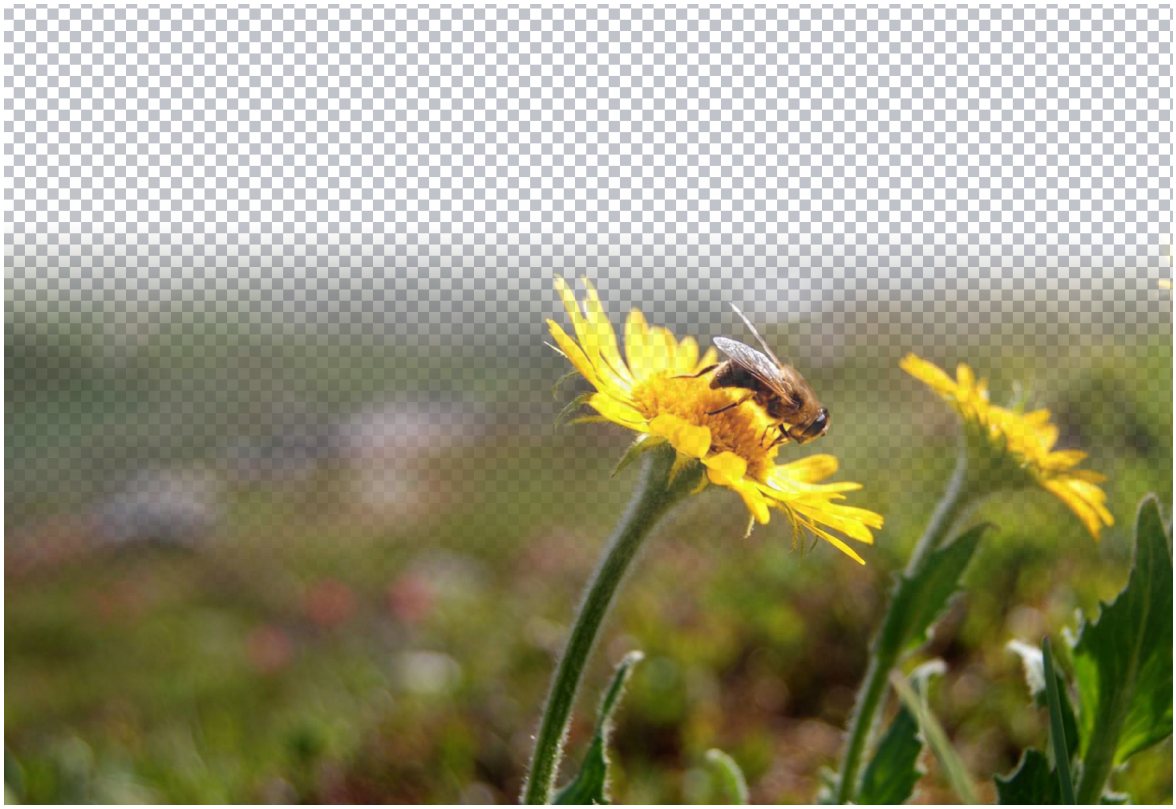
1. Select an effect from the **Image Effects** list.
2. Make modifications to the effect as required in the Effect Modification area.
3. Add other effects as required.
4. If the image needs to be reset to the version on the Graphics Hub click the reset button on the bottom left.
5. Click **Save** or **Save as...**

Image Preview Modes

The button group in the center bottom of the editor allows to select how the image is displayed in the main preview window.



Color + Alpha: Shows the image composition with its alpha map.



Color: Shows only the RGB channels of the image.

Copyright © 2026 Vizrt



Alpha: Shows the alpha map of the image. Black represents full transparency, white indicates fully opaque.

Copyright © 2026 Vizrt



Note: The **Color + Alpha** and **Alpha** modes are only available if the loaded image has an alpha channel assigned to it.

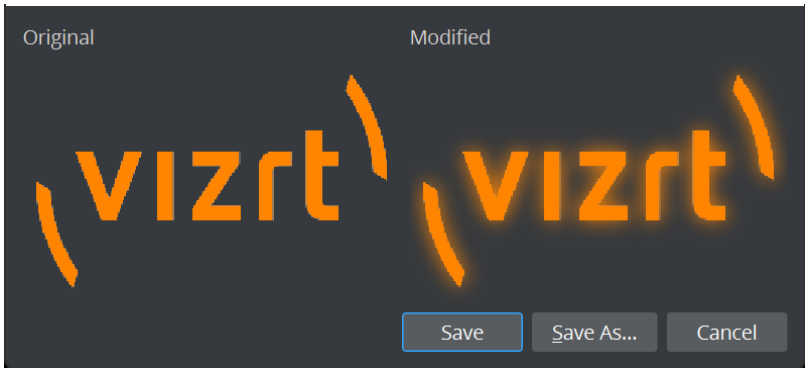
External Image Editor

The image is exported from the Graphics Hub to a temporary folder and opened in the specified program (see Editing Images). All saved changes made in the external editor are reflected in a preview window with the original image on the left side and the edited image on the right side.

When the image editing is done, use the **Save** or **Save as...** button to reimport the image or press the **Cancel** button to cancel the operation.

Note: Make sure you save the modified image with the same name as Viz Artist exported it to the temporary folder, otherwise the re-import can not be done.

Copyright © 2026 Vizrt



Note: Do not reformat the image with the external editor, just save it.

Copyright © 2026 Vizrt

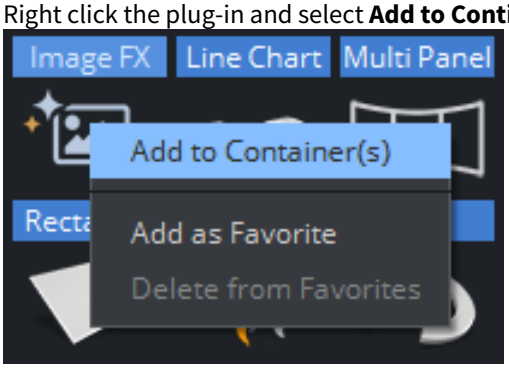
7.3.6 Working with Audio Items

This section contains information on the following procedures:

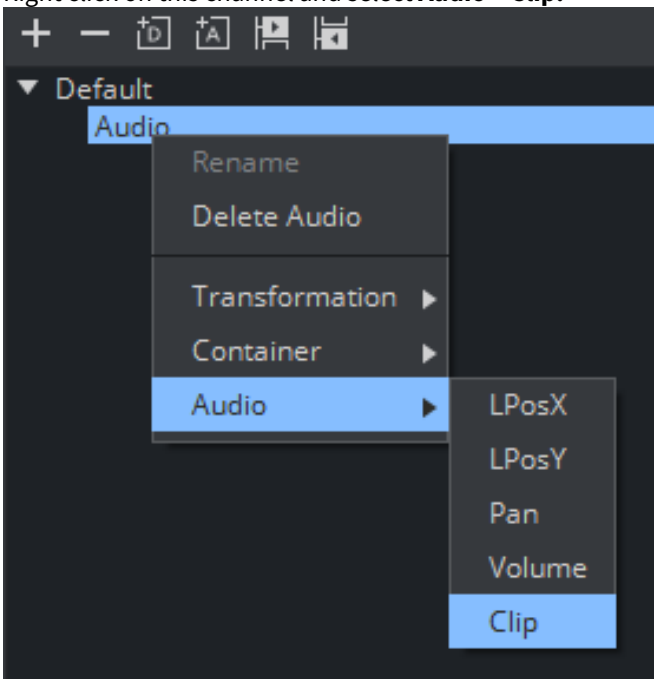
- To Add an Audio Clip to a Scene (Context Menu)
- To Import or Export an Audio Clip

To Add an Audio Clip to a Scene (Context Menu)

1. Select the target Container.
2. Go to **Plugins** and find the **Audio** plug-in.
3. Right click the plug-in and select **Add to Contianer**. The Audio plug-in is added to the selected Container.

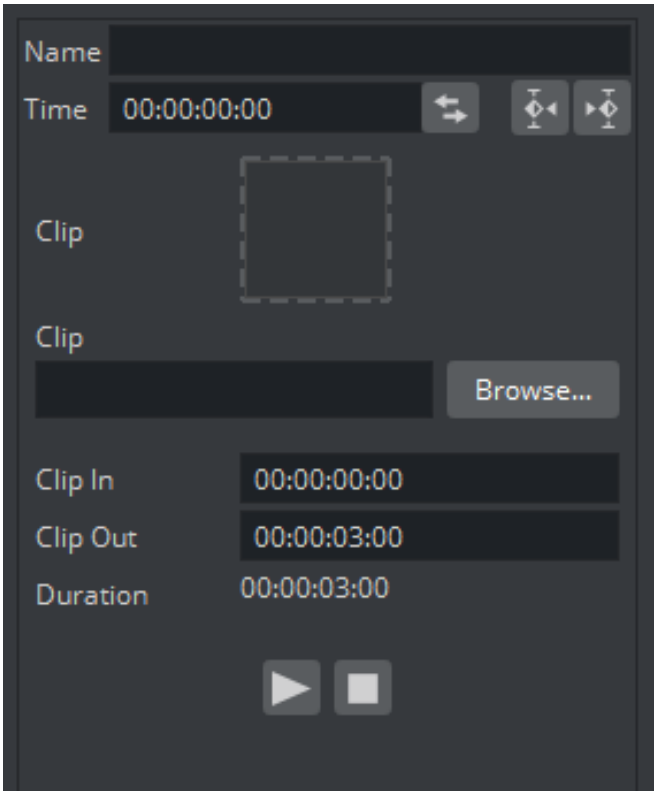


4. The audio needs to be added as animation into the stage, Click on the **Stage** Button.
5. Create a new Director or choose an existing Director.
6. Drag the container to the Director.
7. Right click on this channel and select **Audio** > **Clip.**



8. There are two ways to add an Audio Clip:
   a. **DB Clip:** Select and drag an Audio Clip from the Asset view panel to the **Clip drop box**.
   b. **File:** Browse for the audio clip in the local filesystem.

Copyright © 2026 Vizrt



To Import or Export an Audio Clip

For information about how to import or export an audio clip from a hard disk or a network share to the database see Import, Post and Archive.

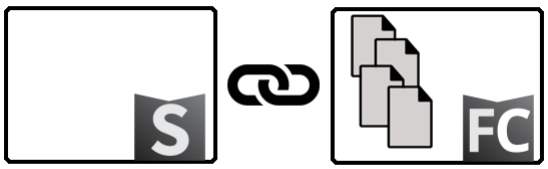
Copyright © 2026 Vizrt

7.3.7 Working with File Collections

File Collections (FC) are additional packages that can be referenced by a scene. A file collection usually consists of a larger number of Binary Files, for example an Unreal project file. These BLOBs (Binary Large Objects) are hidden from the Viz Artist User interface, but are referenced towards the File Collection. A scene can have a reference to an existing File Collection. This implies that:

- When writing an archive, all associated data and project files are written into the archive file.
- Scenes can be deployed together with all associated data.
- When a scene is loaded with a file collection associated, the data can be restored on the playout machine (for example an unreal project can be restored, as soon as the linked scene is loaded).

⚠ **Note:** When a a file Collection File (FC) is deleted, ALL referenced Binary Files are also deleted. This is intended behavior.

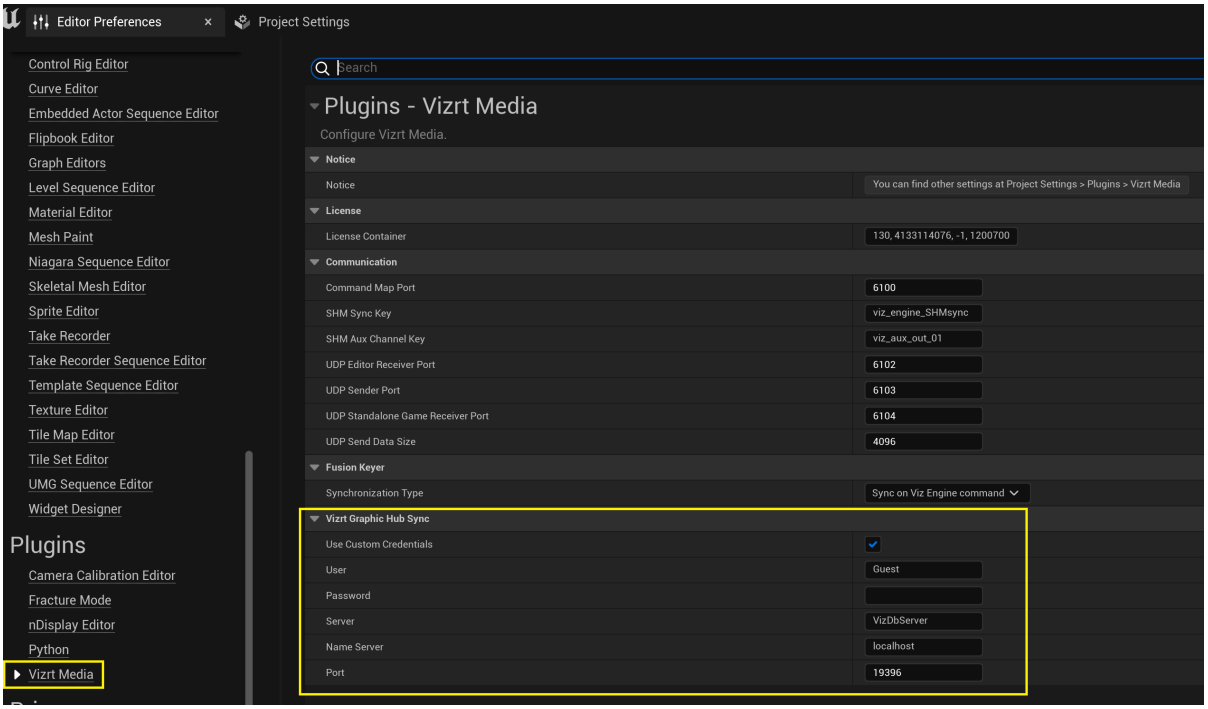


A scene is referenced to a File Collection. The File Collection has references to Binary Data files. This data is stored on Graphic Hub, not on the local drive unless Viz Engine is configured to transfer and restore this data to the play out machine.

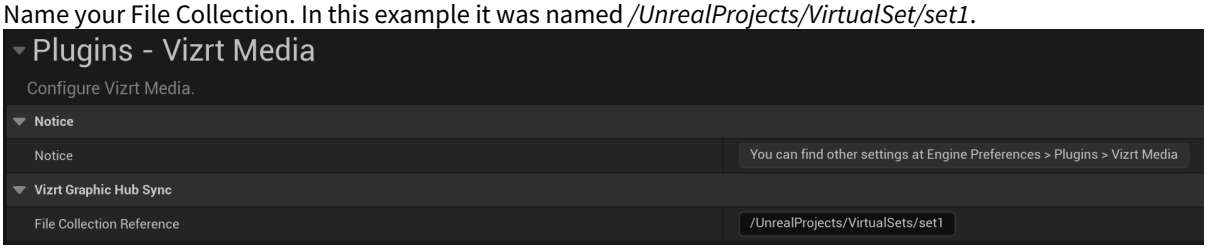
To Sync an Unreal Project to Graphic Hub Using a File Collection

1. Go to the menu **Edit > Editor Preferences**, navigate to the **Vizrt Media** section.
2. Configure the credentials to login to the Graphic Hub server.

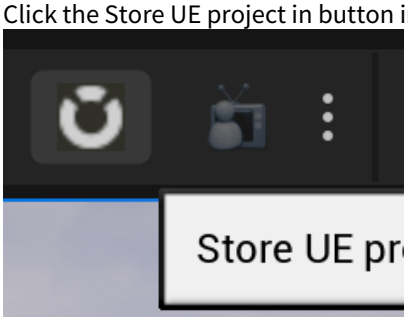
Copyright © 2026 Vizrt



3. Go to **Settings > Project Settings**. Navigate to the **Plugins > Vizrt Media** section.
4. Name your File Collection. In this example it was named */UnrealProjects/VirtualSet/set1*.

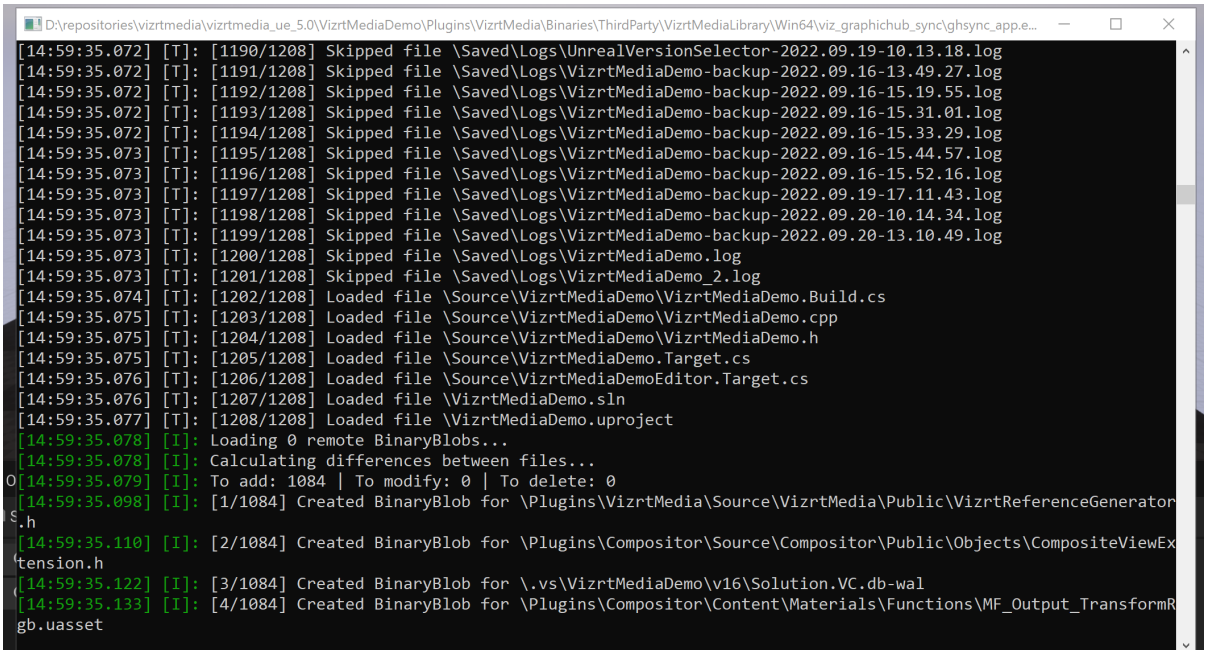


5. Click the Store UE project in button in the toolbar.

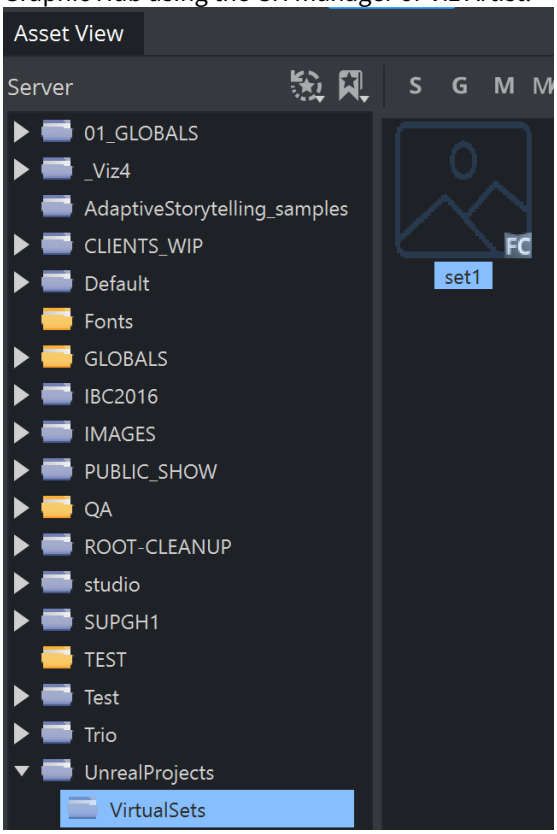


6. A window opens logging the state of the store action. When the window closes the action is completed.

Copyright © 2026 Vizrt



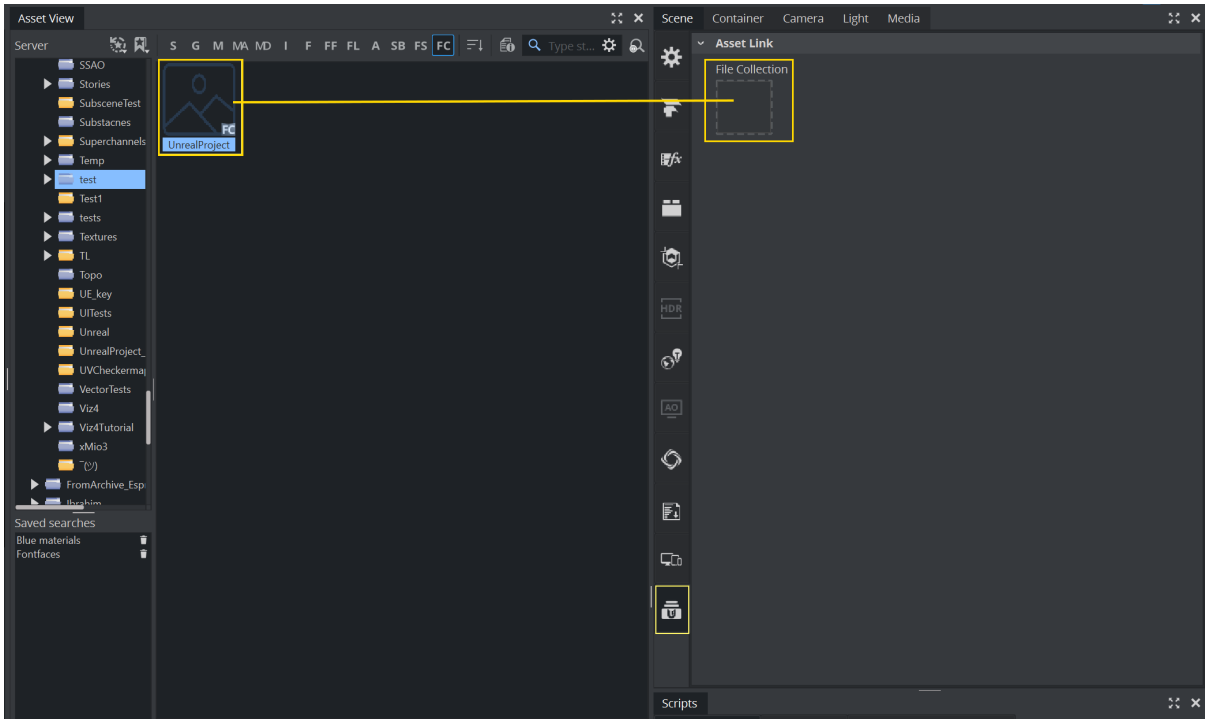
After the process is completed it's possible to confirm that the Unreal project was successfully synced with Graphic Hub using the GH manager or Viz Artist.



Copyright © 2026 Vizrt

To Link a File Collection to a Scene

Drag a File Collection to the Scene Settings of any open scene.



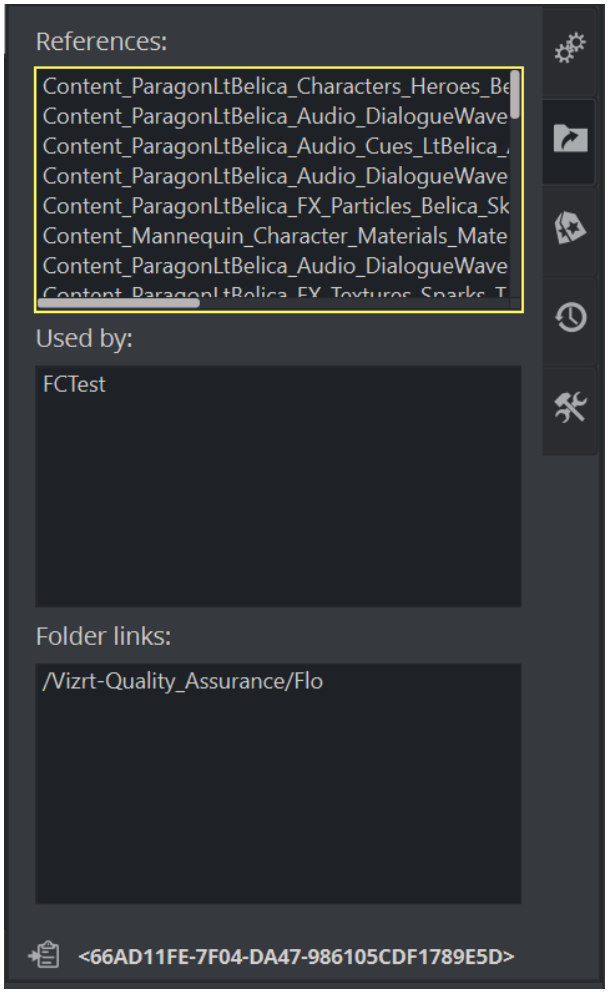
To remove the file collection, double click on the blue **File Collection** label.

> **Note:** Refer to **Configuring Viz > Unreal Engine** in the Viz Engine Administrator Guide to set the restore directory used by Viz Engine when restoring a file collection linked to a scene.

The Content of a File Collection

What is inside of a File Collection is hidden to the Viz Artist User to keep data consistent and protect the data from accidently being deleted by the user. However, to see which files are referenced by a File Collection, use either Graphic Hub Manager or use the Properties Panel of Asset view:

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

7.4 Background Loading

The background loading of Geometries, Materials and Images, without disturbing the rendering and video output, is supported in Viz Engine Classic Renderer with the limitations listed below. Background Loading can either be an automatic or a manual operation. When background loading Geometries, the render method (See the **Render Options** page in the **Configuring Viz** section of the Viz Engine Administrator Guide) must be set to **VBO** (Vertex Buffer Objects).

This section contains the following topics:

- Automatic Background Loading
  - Automatic Background Loading Configuration
- Manual Background Loading
  - Manual Background Loading Configuration
- Background Loading Limitations
  - Scenes
  - Compressed and Uncompressed Images
  - Objects
  - Function Plug-ins
  - Image Combining
  - Black/Transparent Flashing Image Textures
  - Textures Not Visible on SDI Output with Background Loading Enabled
  - Toggle Plug-in

7.4.1 Automatic Background Loading

When set to perform automatic background loading, Viz Engine automatically uses background loading whenever an image, and/or an object, is loaded. There are some limitations to this behavior, which are detailed in Background Loading Limitations.

Automatic Background Loading Configuration

In the Viz Config file, the option **Background_Loading_Level** is available to set up automatic background loading.

Available values for Background_Loading_Level are:

Value	Description
`BGL_INACTIVE`	Disables automatic background loading.
`BGL_IMAGES`	Enables automatic background loading for images.
`BGL_OBJECTS_AND_IMAGES`	Enables automatic background loading for images, geometries and materials.

Copyright © 2026 Vizrt

7.4.2 Manual Background Loading

Manual background loading can be done on an on demand basis. To do this, commands are available to use Container objects:

Command	Description
`GEOM SETBGL`	Loads (and sets) a geometry in the background. The usage of the command is exactly the same as the existing `GEOM SET` command.
`IMAGE SETBGL`	Loads (and sets) an image in the background. The usage of the command is exactly the same as the existing `IMAGE SET` command.
`MATERIAL SETBGL`	Loads (and sets) a material in the background. The usage of the command is exactly the same as the existing `MATERIAL SET` command.

Manual Background Loading Configuration

To use manual background loading, the **Background_Loading_Level** configuration flag in the configuration file must be set to `BGL_INACTIVE`.

7.4.3 Background Loading Limitations

This section details the limitations of background loading.

Scenes

Background loading of whole Viz Artist Scenes is not supported.

Compressed and Uncompressed Images

Only compressed images (DXT) are supported for background loading. Their mipmap-levels have been pre-generated and stored in the Graphic Hub, which is required for background loading to work properly. The generation of mipmap levels on-demand (as required for uncompressed images) can interrupt the renderer. This primarily depends on the size of the uncompressed images. The larger the image, the more impact on the renderer during background loading.

Objects

Only native geometry and imported third-party geometry objects containing vertex-data, materials and textures are supported targets for background loading. Containers hosting any sort of other plug-in may fail to load in the background. Containers hosting Function plug-ins are NOT loaded in the background, but rather fall back to a normal foreground loading. Containers hosting RTT shaders interrupts the renderer if the particular RTT shader is

Copyright © 2026 Vizrt

used for the first time (shader is being compiled), but no longer interrupts the renderer when loading the same shader in the background again.

Function Plug-ins

If a geometry, which is to be loaded in the background, contains one or more Function Plug-ins, the console shows a warning and the geometry is not loaded in the background but in the foreground instead, this can interrupt the renderer.

Image Combining

If a geometry uses the second image (image2) in a texture, this might have an impact on the performance of the renderer. Mostly if one of the images are too large and/or uncompressed.

Black/Transparent Flashing Image Textures

When setting an image with background loading, for technical reasons, the renderer gets the handle to a black (or transparent) texture until the first mipmap level of the image is loaded. This can result in a short flash of black (or transparent) texture while background loading textures on a visible object. To fully prepare textures of invisible objects and avoid texture flashes if there is enough preparation time, activate Preload Textures in the Memory Management Configuration.

Textures Not Visible on SDI Output with Background Loading Enabled

When Viz Trio prepares and takes a scene, it momentarily stops the Viz Engine renderer. If background loading is enabled, this can result in unexpected behavior, such as textures becoming visible only in the preview and not the video output. To avoid this issue when background loading and controlling the engine with UPDATE commands, such as sent by Trio, Viz Engine by default ignores the command `RENDERER *UPDATE SET 0`. This can be changed by locating the following configuration option in the Viz Configuration file, in the section `RENDERER_OPTIONS`, and changing the value from `0` to `1`:

```
RendererUpdatesForBackgroundloading = 1
```

This fixes potential background loading texture issues on the second channel of dual channel setups as well.

Toggle Plug-in

The toggle plug-in can not handle background loading of geometries or scenes.

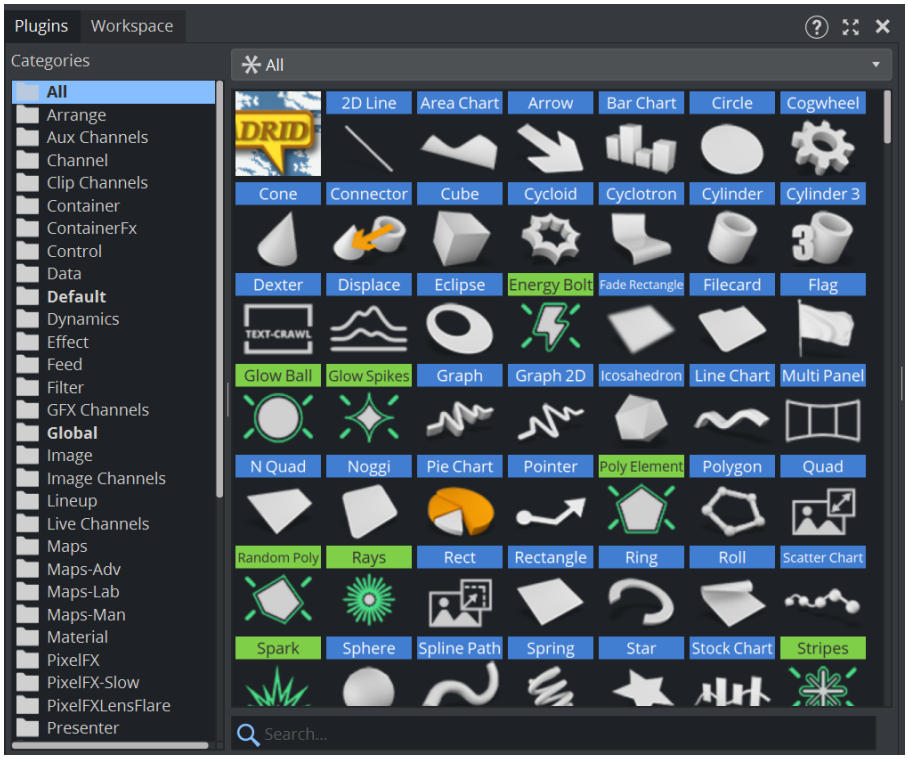
Copyright © 2026 Vizrt

7.5 Plug-ins Tab

In Plug-ins, there are a standard set of Viz Artist installation plug-ins. Most plug-ins can be added to Containers, except, for example, Scene plug-ins. Because Viz Artist uses an open interface for the plug-ins, new geometries can either be obtained or created and then installed with Viz Artist. Also, most plug-ins have a properties editor. When a plug-in is applied to a Container in the Scene Tree, a plug-in icon is added to the target Container.

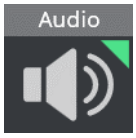
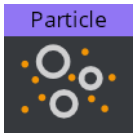
The drop-down button gives access to the various plug-ins:

- Geometry plug-ins
- Container plug-ins
- Classic Shader plug-ins (Classic Render Pipeline only)
- Viz Engine Shader plug-ins (Viz Engine Render Pipeline only)
- Scene plug-ins
- Media Assets



7.5.1 Plug-in Color Coding

Plug-ins have different label colors for easier and faster work.

Icon	Plug-in	Style
	Geometry plug-ins	Blue label
	Container plug-ins	Gray label
	DataPool plug-ins	Gray label with blue triangle
	Control plug-ins	Gray label with green triangle
	Scene plug-ins	Pink label
	Shader plug-ins	Green label
	Viz Engine Shader plug-ins	Purple label
	Media Asset plug-ins	Black label

The search field on the bottom of the Plugin panel allows to do a free text search for all kind of plug-ins. For example, if you want to add a Rectangle plug-in, simply start typing `Rect` and only plug-ins with *rect* in their name are displayed.

This section contains information on the following procedures:

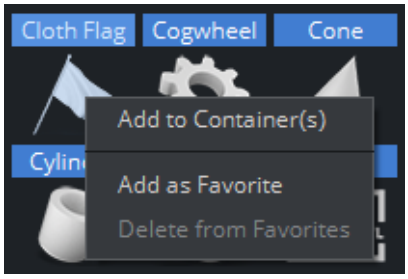
- Plug-in Context Menu
- View and Search for Plug-ins
- Add Plug-ins to Containers

Copyright © 2026 Vizrt

7.5.2 Plug-in Context Menu

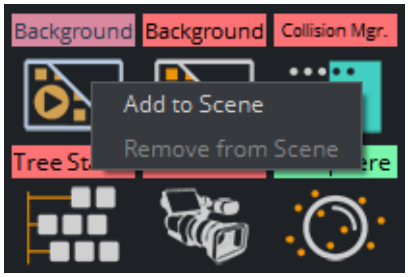
A right-click on a plug-in icon to open the plug-in context menu.

Geometry, Container, Classic Shader and Viz Engine Shader Menu



- **Add to Container(s):** Adds the selected plug-in to a Container(s).
- **Add as Favorite:** Adds the selected Texture to the Plug-in Overlay.
- **Delete from Favorites:** Deletes the selected Texture from the Plug-in Overlay.

Scene Menu



- **Add to Scene:** Adds the selected plug-in to a Scene.
- **Remove from Scene:** Removes the selected plug-in from a Scene.

Copyright © 2026 Vizrt

Media Asset Menu



- **Add <Media Asset> as Texture:** Adds the selected Media Asset to a Scene (automatically adds a new Container).
- **Add <Media Asset> as DVE:** Adds the selected source as DVE.
- **Add as Hybrid:** Some Media Assets (like Live inputs) can be used as DVE and Texture at the same time.
- **Show settings of <Media Asset>:** Shows the settings of the Media Asset, if it has been added to the scene and the Scene Tree is in Texture or DVE view.
- **Add as Favorite:** Adds the selected source to the Plug-in Overlay.
- **Delete from Favorites:** Deletes the selected source from the Plug-in Overlay.

For the **Background** and **Foreground** media assets, the context menu shows the Favorite options only.

7.5.3 View and Search for Plug-ins

All plug-ins can be dragged straight into the Scene Tree (except Scene plug-ins). Double-click a folder tab to go to the **Default** folder for that plug-in category.

To View Plug-ins

1. Select **Plugins**.
2. Select the appropriate plug-in category, either:
    - From the drop-down menu that shows above the plug-ins view, or
    - Click on of the tabs.

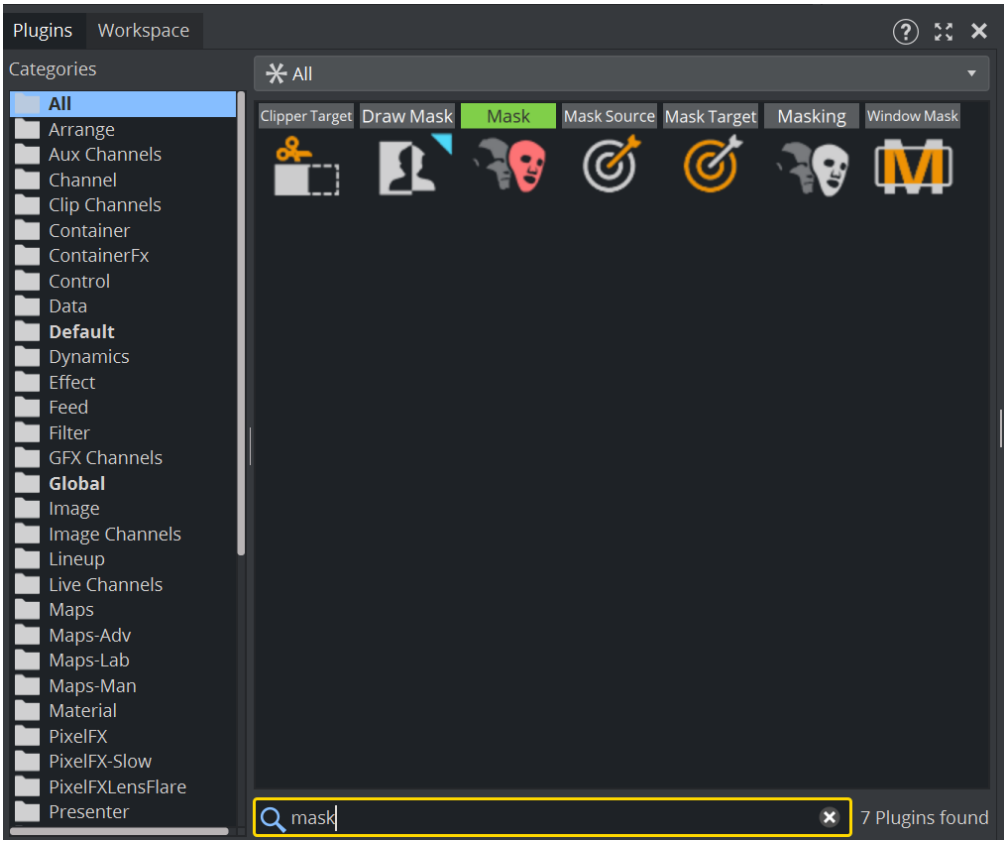
The All folder in the left panel, offers an uncategorized view of all plug-ins. Pressing a letter on the keyboard to jump to a plug-in which starts with that letter.

To Search for a Plug-in

You can drag plug-ins from the search results to the desired destination.

1. Select **Plugins**.
2. Enter a search string. Results narrow as you type.

Copyright © 2026 Vizrt



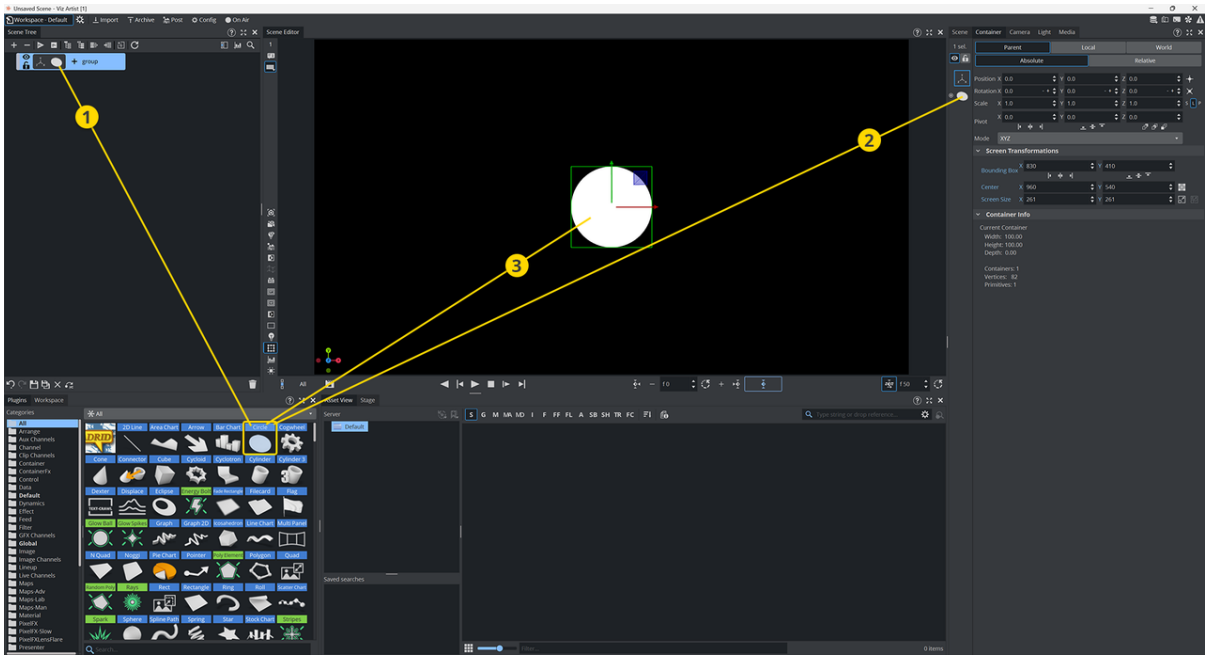
7.5.4 Add Plug-ins to Containers

The quickest way to add a plug-in to a container is to use the Plugin Overlay Panel.



For more details please see Plug-in Overlay.

Copyright © 2026 Vizrt



Most plug-ins can be dragged to a target Container(s) in the scene tree (1), the Properties Panel (2) or the Scene Editor (3). Each plug-in also has a context menu where a plug-in can also be added to a Scene.

To Add a Geometry Plug-in

A Container can only contain one Geometry plug-in at a time. If a new Geometry plug-in is added, the fist one is removed from the Container.

1. Drag the Geometry plug-in icon to either:
   - A Container or multi selected Containers.
   - The scene tree area (creates its own container).
   - The Scene Editor area.
   - The Container Editor area.

   or

1. Highlight the target Container(s).
2. Right-click the plug-in icon and select **Add to Container(s)** from the plug-in context menu.

To Add a Container Plug-in

1. Drag the Container plug-in icon to either:
   - A Container or multi selected Containers.
   - An object in the Scene Editor (only some container plug-ins do this, when they do, the object becomes active when the icon is moved over the object).
   - The Container Editor area.

   or

1. Highlight the target Container(s).
2. Right-click the plug-in icon and select **Add to Container(s)** from the plug-in context menu.

Copyright © 2026 Vizrt

To Add a Classic Shader Plug-in

A Container can only contain one Classic Shader at a time. If a new Classic Shader is added, the fist one is removed from the Container.

1. Drag the icon to either:
   - A Container or multi selected Containers.
   - An object in the Scene Editor.
   - The Container Editor area.

   or

1. Highlight the target Container(s).
2. Right-click the plug-in icon and select **Add to Container(s)** from the plug-in context menu.

> **Information:** Classic Shaders are only supported in the Classic Render Pipeline.

To Add a Viz Engine Shader Plug-in

Viz Engine Shaders work the same way as old Classic Shaders, but are only available in the Viz Engine Render Pipeline.

> **Information:** Viz Engine Shaders are only supported in Viz Engine Render Pipeline.

To Add a Scene plug-in

1. Right-click the Scene plug-in icon to open the plug-in context menu.
2. Click on **Add to Scene**...

To Add a Plug-in to Multiple Containers

1. Highlight the required Containers.
2. Drag a plug-in icon to a highlighted Container.
3. Release the plug-in icon.

   or

1. Drag a plug-in icon.
2. Hold the plug-in icon over a Container.
3. Continue to hold the left-click and do a right-click to drop the plug-in icon on the Container.
4. Continue to hold the left-click and repeat for each required Container.

To Add a Plug-in to Favorites

1. Right-click the icon to open the plug-in context menu.
2. Click **Add as favorite...**

You can also add plug-ins as favorites from the search panel.

Copyright © 2026 Vizrt

To Open a Plug-in Properties Editor

Click on the **Media Asset** icon, in the Container or the in the Properties Editor.

See Also

- Plug-in Overlay
- Media Assets
- Containers
- Scene Editor

Copyright © 2026 Vizrt

8 Working in HDR Color Space



The most brilliant spectrum of highlights and shadows in modern HDR products, also demands beautiful graphics and a perfect composition of these assets, along with enhanced realism in color and detail. The improved contrast ratio of HDR, closely mirrors real-life conditions, facilitating image reproduction that aligns closely with reality.

For broadcasters and content creators, adopting an HDR workflow, introduces various questions and challenges. A primary concern is the necessity for a parallel SDR/HDR production workflow, which allows for the adaptation of signals through up/down mapping, as needed. This adaptability is crucial for mixing and compositing incoming content formats, and output signals without compromising quality.

The need to produce real-time enabled graphics in addition to HDR enabled formats for the broadcast industry as well for streaming services, requires new workflows. Viz Artist and Viz Engine come with new capabilities to help designers fulfill these new challenges.

The following chapters explain the processes and workflows in Viz Engine associated with HDR production, and outlines designing solutions in native HDR:

- Introduction to HDR
- Prerequisites and Configuration Details
- Usecases for HDR
- Mediatypes in HDR Workflows

Copyright © 2026 Vizrt

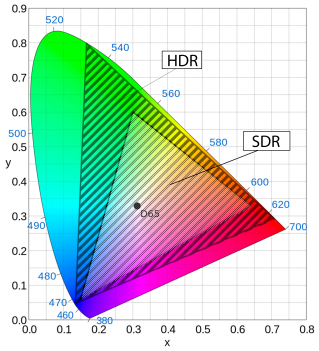
8.1 Introduction to HDR

High Dynamic Range (HDR) is a technology utilized in video production that enables a wider contrast ratio, increased brightness, and enhanced color depth, compared to Standard Dynamic Range (SDR) video. HDR improves the visual experience by providing bright highlights, deep shadows, and a broader color spectrum, making videos appear more realistic and lifelike.

In HDR video, the range of light intensity surpasses that of traditional video, allowing scenes with extreme lighting conditions (such as a sunset or a dimly lit room) to retain greater detail. This results in richer, more immersive experiences, particularly on HDR-compatible displays including modern televisions, computer monitors, and mobile devices.

Info: Viz Engine supports Hybrid Log Gamma (HLG-HDR).

8.1.1 Color Spaces, Brightness and Contrast



Color values within the inner triangle, are within the SDR color space. Other colors are represented in the HDR color space.

A color space defines the range of colors that can be displayed.

The human eye is capable of recognizing many more colors than common displays or TV screen can produce. Traditionally a television monitor is capable of displaying a certain, limited range of colors, also known as **Standard Definition Range SDR (BT 709)** color space. The SDR color space is represented as the smaller rectangle in the image on the left.

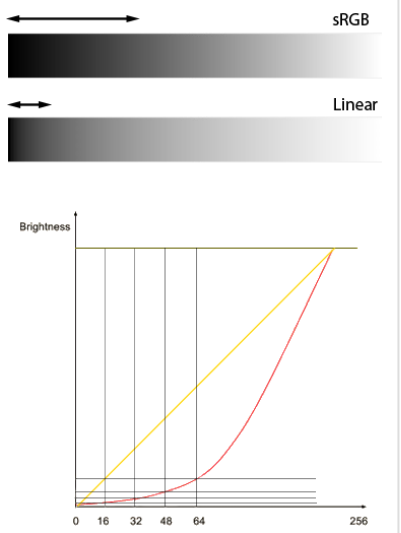
In comparison to SDR, the **High Definition Range** or **HDR**, is capable of representing a much higher range of details, colors and brightness.
The image on the left represents the HDR color space as the larger triangle. HDR captures a greater range of brightness levels, therefore, it is able to represent more details in shadows and highlights, on an image. The most commonly used color space for HDR is **Rec. 2020**, also used in Viz Engine 5.4.

As of the larger color spaces, HDR is able to display a much higher contrast. With more distinct differences between darker and lighter areas, resulting in a more vibrant and realistic output. It comes with much finer details, and subtler gradations of brightness levels. High Definition requires more color depth, therefore, representing a color requires 10 bit instead of 8 bit per channel, to avoid any banding in gradients.

HDR capable devices are able to display brighter pixels than standard monitors. A typical HDR capable monitor can display a peak brightness of 1000 nits and more, whereas SDR is not able to show more than 100 nits. These higher brightness levels allow a greater detail in highlights and shadows. The maximum level of brightness shown in HDR ranges from 0 (complete black) to 100 (brightest white).

Copyright © 2026 Vizrt

Linear and sRGB Color Spaces



The human eye is more sensitive to light in lower intensities, then to higher ones. If we use a linear gradient (bottom) from black to white, the eye perceives more dark shades than light ones. In mathematical terms, human eyes have a roughly logarithmic response to light.

This means that representing colors linearly encodes more different bright colors that we can actually distinguish, but not enough dark colors to prevent banding. The simplest solution, using more bits to increase the number of distinct color levels, is inefficient in terms of data size. Therefore, most common color encoding systems use non-linear brightness levels like gamma curves. They redistribute available code points to match human perception. Smaller differences in the dark areas are encoded at the cost of larger gaps in the bright areas.

Hybrid Log-Gamma (HLG)

Hybrid log-gamma is the most common HDR format in broadcast. It was developed by the BBC and NHK. It is standardized in ITU recommendation BT.2100 and uses the BT.2020 color space. Similar to established SDR standards, HLG encodes relative scene light in a non-linear curve. The main difference to SDR, is that HLG switches from a gamma curve to a logarithmic curve in brighter areas. This system supports practical luminance levels from 1,000 up to 2,000 cd/m2.

HLG is very common in broadcast, and Viz Engine supports it far better than anything else. At this point, HLG is the only recommended HDR format to use with Viz Engine.

Other HDR Formats

This section provides a short overview of other HDR, and HDR-adjacent video formats.

S-Log3

S-Log3 was developed by Sony, and Viz Engine only offers basic support for this format.

Perceptual Quantizer (PQ)

PQ was developed by Dolby, and unlike all other formats described, it is an absolute display-referred system. Meaning that a PQ signal directly encodes the brightness of a pixel on the display in nits. Common consumer HDR formats like HDR10 are based on PQ. As HLG, PQ is standardized in ITU recommendation BT.2100, and uses the BT.2020 color space. Conversions from HLG to PQ are relatively straight forward.

Note: Viz Engine does not support PQ.

Copyright © 2026 Vizrt

Wide Color Gamut SDR Video

ITU recommendation BT.2020 defines a wide color gamut SDR video format that still uses a traditional gamma curve. This approach has not found wide adoption in broadcast, and is not supported by Viz Engine.

Reference White - White Level

The white level (or reference white) is the brightness of a non-reflective, white object. Colors that exceed this brightness, are called specular highlights. The established white level of Hybrid Log Gamma (HLG) is at 75% of the signal, which corresponds to 203 nits on a Monitor with a peak brightness of 1000 nits.

In Viz Engine, white level is configurable and serves two purposes:

1. 100% of an SDR color is mapped to the HDR white level.
2. The HDR white level is scaled to a linear light value of 1.0 (or 100% white) in HDR scenes.

It is recommended to use brightness values around graphics (white for overlays, graphics, fonts, etc.).

Look-up Tables - LUTs

Color spaces need to be transformed from one to another. For example, translating an SDR live input to an HDR output in Viz Engine, or if a calculation like blending needs to be performed. There are multiple ways to perform these conversions and map the colors of an input image to a target color space. They serve different purposes and make different tradeoffs in terms of color accuracy, brightness preservation, and more.

Look Up Tables (LUTs) are a common approach to encode a given conversion. Viz Engine comes with built-in NBCU LUTs to support color conversions between SDR and HLG, but also offers support for external 3D Look Up Tables from *.cube* files. The BBC established a nomenclature of different HDR conversion LUT types, depending on the processing range. Viz Engine supports Type I LUTs (which are limited to nominal range), and Type III LUTs (which support extended range).

Copyright © 2026 Vizrt

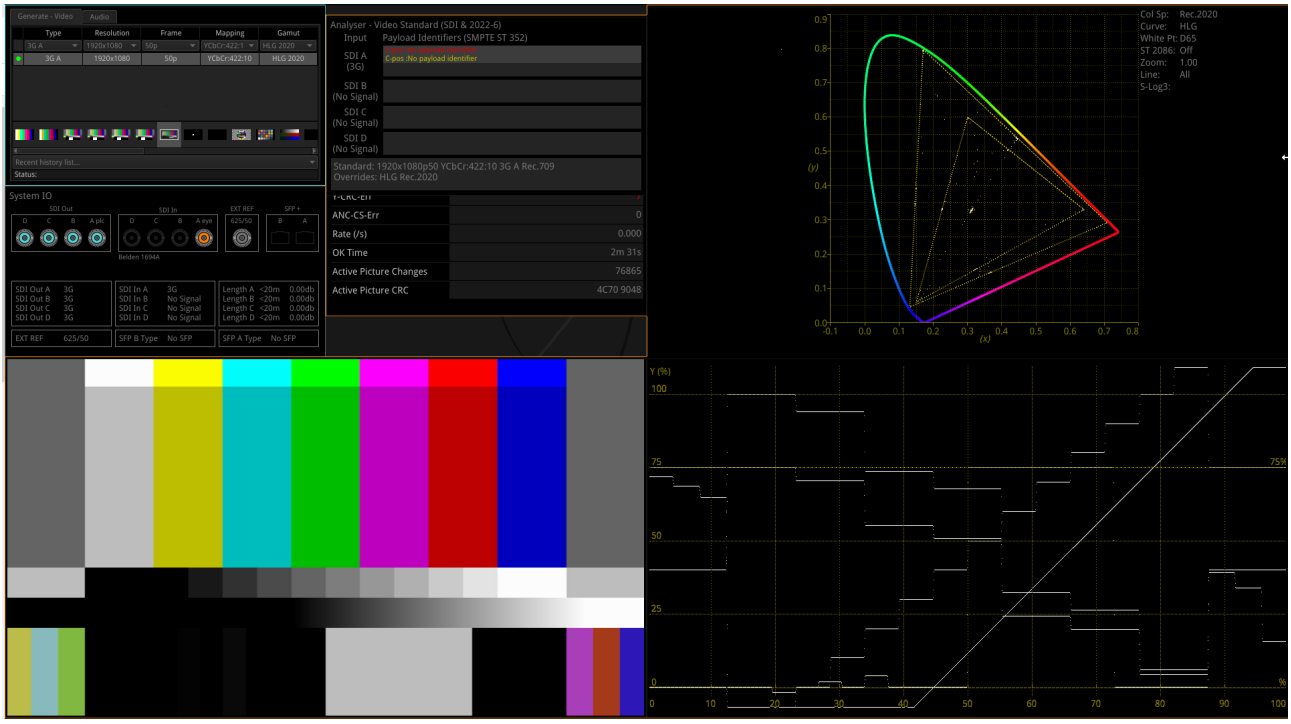
8.2 Prerequisites and Configuration Details

8.2.1 Prerequisites

Supported Video Signals and Engine IO Modes

HDR is supported with Matrox SDI or IP boards, and with software I/O mode NDI. The HDR preview (see below) on desktop monitors, can be used with VGA engine versions as well.

To verify the correct output of your HDR design, we recommend using a calibrated SDI reference monitor. A spectrum and waveform analyzer helps measure and validate the correct output.



Requirements

HDR requires resolutions of 1080p or higher (HDR in 1080i is not supported). HDR rendering is highly performance and memory intensive, with the use of higher bit depth, and the additional conversion steps required.

8.2.2 Setup HDR Preview

For designing purposes, an HDR capable monitor can be used. However, to verify the final output, an HDR capable UHD SDI reference monitor is required.

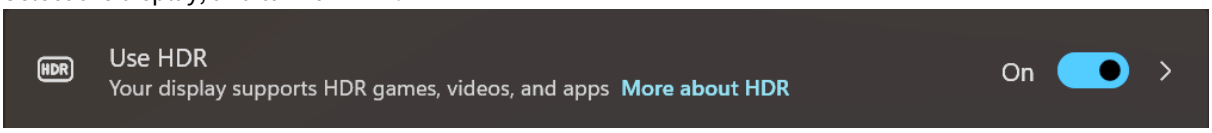
Copyright © 2026 Vizrt

Viz Engine can convert to scRGB colorspace to create an HDR window, provided by the Operating System (Windows 10 and 11). This requires an HDR capable Desktop Monitor.

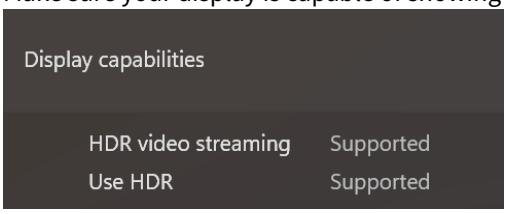
> **Note:** The HDR preview is for design purposes only. The final output needs to be verified on a profession HDR SDI/IP environment, with proper analysis tools.

To setup an HDR window preview, you need to enable **Use HDR** in the Windows Display settings:

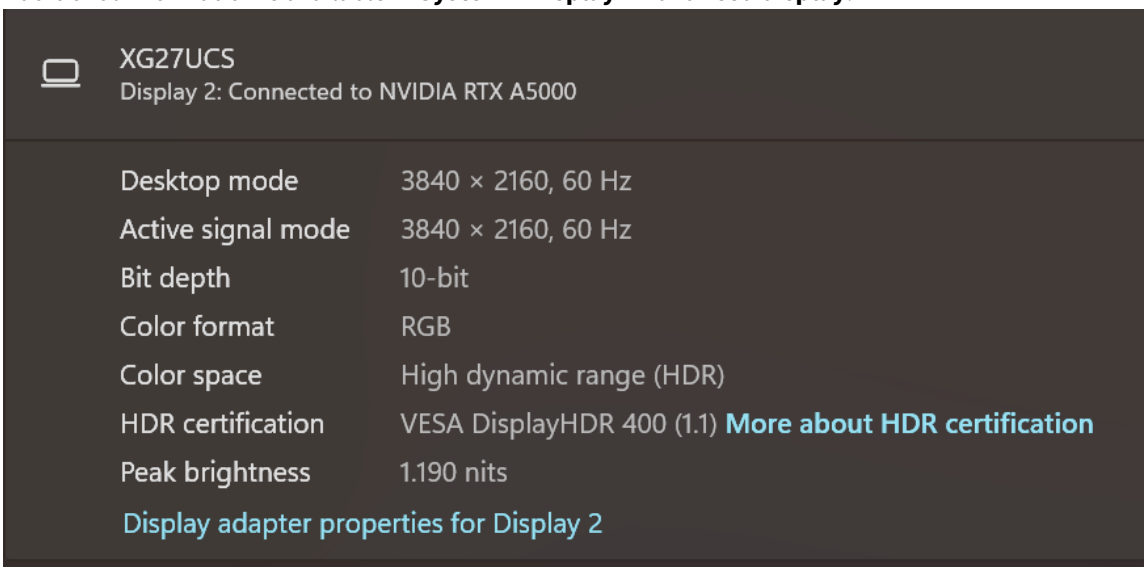
- In Windows Control Panel, click on **System** > **Display**.
- Select the display, and turn on HDR:



- Make sure your display is capable of showing HDR content:



Additional information is available in **System** > **Display** > **Advanced display**:



We recommend installing a color profile provided by your display vendor, or do a manual calibration using the Microsoft HDR Calibration tool:



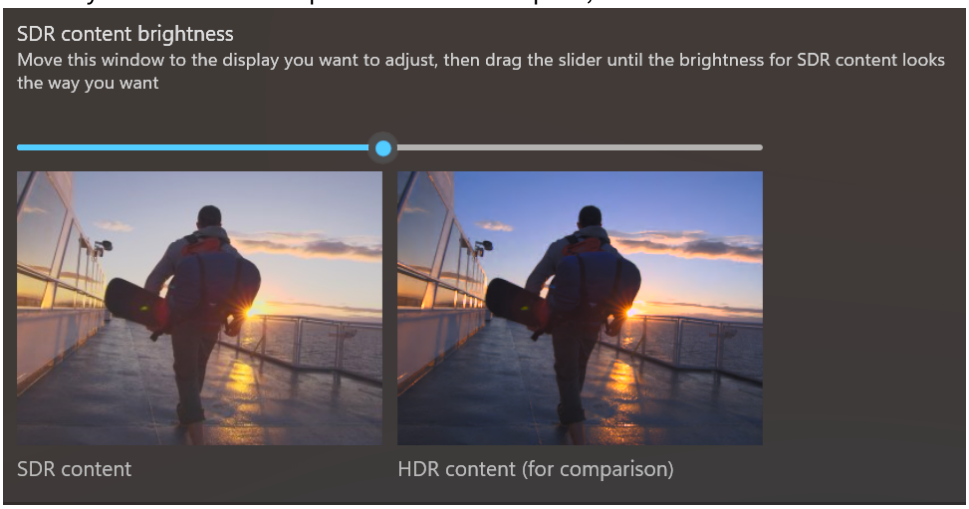
- In Viz Config Settings, enable HDR preview:



The Scene Editor now shows the content in scRGB colorspace, providing an HDR preview of your scenes.

Copyright © 2026 Vizrt

- To verify if the window is capable of HDR color space, move the slider in the SDR content brightness:



If the scene editor window is **not being affected** by the slider setting, the HDR output is working correctly.

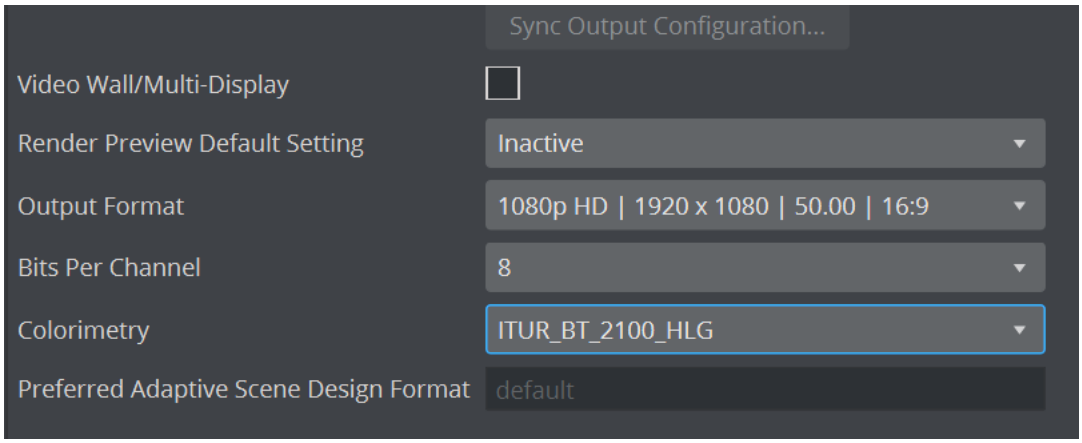
Note: Even if the operating system handles changes in color space, and window positions of multiple monitors, we recommend using one single HDR monitor.

Copyright © 2026 Vizrt

8.3 Usecases for HDR

High Dynamic Range (HDR) introduces new workflows and requirements in video production. This document outlines the main use cases involving HDR, the interaction between environment and scene configurations, and the necessary conversions between SDR (Standard Dynamic Range) and HDR.

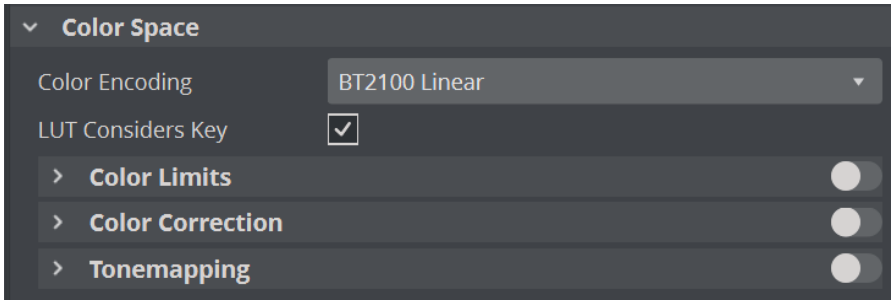
The **environment** refers to the format of video signals and clips in use. It is generally static, with the engine supporting both SDR and HLG signals (the configuration must match the environment). Configuration details are described in the Viz Engine Administrator Guide.



Note: The incoming signal needs to match the configuration.

The **scene** can be either SDR or HDR, but Classic Render Pipeline scenes are always SDR. Starting with Viz Engine 5.4, it is possible to create Viz Engine Render Pipeline (VER) scenes in either SDR, or HDR. VER scenes built in earlier versions are SDR.

HDR scenes define as scenes that use the linear BT.2100 color encoding. The color encoding of a scene should be determined in advance and remain unchanged. The scene color limits are also relevant for HDR, the default limits of 0 and 1 clamp the output to nominal signal range. Using superblack and superwhite requires relaxed or disabled color limits.



The use case is defined when combining the environment with the scene type. The traditional use case, where both are SDR, is not covered in this document. Any difference between environment and scene, requires conversion.

Copyright © 2026 Vizrt

Input and output must always match the environment, and the required configuration for HDR conversions is similar across different scenarios.

- SDR Scenes in an HDR Environment
- HDR Scenes in an HDR Environment
- HDR Scenes in an SDR Environment
- Limitations and Known Issues

8.3.1 SDR Scenes in an HDR Environment

This is the most common scenario when introducing HDR workflows, as the majority of the existing scenes, are built in SDR. It allows for compatibility with existing SDR content, and requires minimal redesign. A major advantage of SDR scenes, is that they are built in the color space that is most familiar to the majority of designers.

Existing SDR scenes can be used as overlays or graphics on HDR video signals, or integrated into HDR environments. In this setup, graphics are limited to SDR, and video within the scene is converted to SDR, while the video used in the scene background remains HDR. The recommended scene design is an input as scene background, or DVE with an SDR overlay. Depending on the render sequence, the conversion is applied to keyed elements of a scene.

This use case requires an SDR to HDR conversion, and LUT to embed SDR scenes into HDR signals. If HDR inputs should be used within a scene, an HDR to SDR conversion is required as well.

8.3.2 HDR Scenes in an HDR Environment

This is the native approach for full HDR workflows, and the sole method to use HDR's full capabilities.

Scenes must be designed from the start in HDR, using the linear BT.2100 color space. If all assets are HDR, no conversions are required. However, SDR assets require conversion using an SDR to HDR LUT. HDR to SDR conversion is not necessary, but should be enabled in the configuration, if it is possible that SDR scenes are used as well.

8.3.3 HDR Scenes in an SDR Environment

This scenario is typically used as a backup, or a transitional solution when existing HDR scenes need to be used in an SDR-only environment. The final output is in SDR. There is generally no benefit in designing new HDR scenes for SDR environments, but existing HDR scenes can be reused this way. Conversions from and to HDR are needed.

8.3.4 Limitations and Known Issues

- The Precision Keyer is only supported in pure SDR environments.
- Postrendering, ClipOut as well as NLE workflows, are not supported.
- Image Export and Snapshots are not supported.
- Image Editor is not able to handle HDR images. We recommend using an external image editor instead.

Copyright © 2026 Vizrt

8.4 Mediatypes in HDR Workflows

To match the scene's color space, assets need to be converted from their given space, to the target/scene color space. Depending on the type of media, a few actions might be required to work correctly.

The input color space conversion needs to be active, preferably to **Both**:



- Using Live Video
  - SDI/IP and NDI Sources
  - AUX Channels
- Clip Playback in HDR Workflows
  - Matrox Clip Player
  - VML Clip Player
  - Texture Clip Player
  - Media Assets
    - Nested Channels - GFX Channels and Super Channels
    - Image Channels
  - How to Work with Images in HDR
    - JPEG XL Image Support

8.4.1 Using Live Video

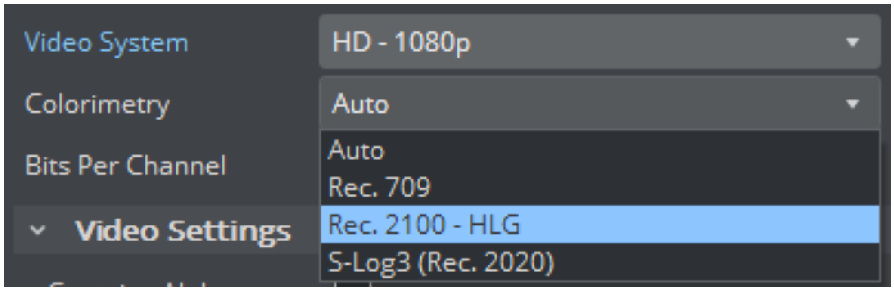
SDI/IP and NDI Sources

SDI/IP inputs with Matrox boards and NDI inputs are fully supported in HDR workflows. They are converted between SDR and HLG, and the renderer selects the appropriate version of the input texture for the given use case.

The colorimetry of NDI inputs is determined based on NDI signal’s metadata. Signals with missing metadata are treated as SDR. The only requirement to use (and send) NDI stream, is to set the in and output color depth to 16-bit. NDI streams deliver the required metadata to automatically set the correct color space.

Other live input types (for example, SDI inputs with AJA or Bluefish, SHM SMURF, WebRTC, etc.) are only supported in SDR scenes. To use SDI or IP based sources, make sure the correct colorimetry is set in the config settings for your input source.

Copyright © 2026 Vizrt



Mismatches between the configured colorimetry lead to incorrect conversions.

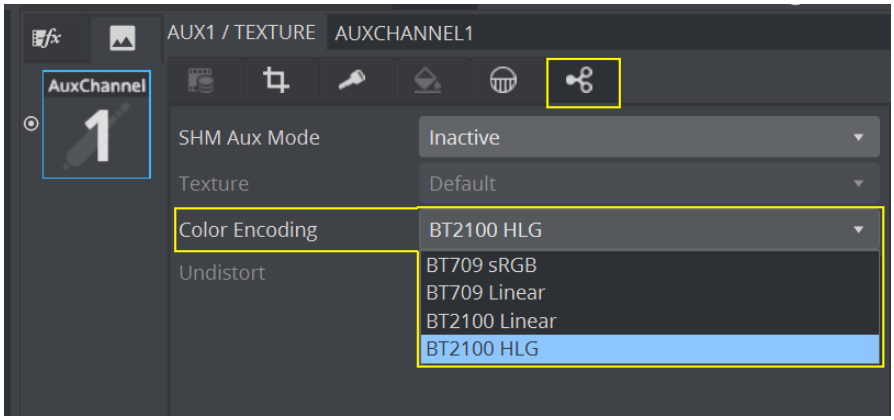
Note: Input channels using hardware DVE on Matrox X.mio boards (SDI, IP, and Matrox Clip player) only work if the colorimetry of the input channel matches the output colorimetry.

Limitations:

- S-Log3 (Rec 2020) is for backward compatibility only.
- SDI/IP sources in DVE mode need to match the output colorimetry.
- SDI or IP based HDR workflows are supported on Matrox boards only.

AUX Channels

AUX channels (used to exchange data with third party renderers, such as Unreal Engine) need to have the correct color space set in the in the Aux tab.



8.4.2 Clip Playback in HDR Workflows

HDR is supported by the Matrox clip player, VML clip player, and, for Viz Engine Render Pipeline (VER) scenes, clip player texture renderer. As with supported live inputs, the textures are automatically converted, and the correct variant is chosen.

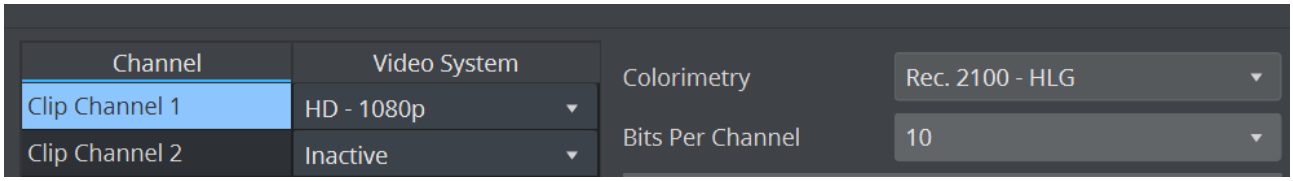
The Matrox clip player treats all opened clips as if they had the configured colorimetry. Loading a clip with a different colorimetry leads to incorrect results.

Copyright © 2026 Vizrt

VML and clip player texture renderer use the metadata of the clip file instead. Clips with incorrect metadata are used incorrectly.

Matrox Clip Player

To use the Matrox clip player for HDR clip playback, the colorimetry needs to be set explicitly in the Viz Configuration:



Note: If you are using DVE inputs, the colorimetry needs to match the output colorimetry.

VML Clip Player

The VML based clip player can read the metadata of a clip and set the color space accordingly. Therefore, the metadata within the clip needs to be set correctly.

Texture Clip Player

As the texture clip player is also using the VML clip player, metadata is being used to convert to the scenes color space.

Media Assets

Nested Channels - GFX Channels and Super Channels

GFX channels are converted into the host scene's color space if the config flag `gfxchannel_dynamicrange_conversion = 1` is active.

If the output format is set to be HLG and the scene is SDR, a conversion of GFX channels from HDR → SDR → HDR might happen, resulting in severe losses.

Image Channels

Image channels do not perform conversions between SDR and HDR. They only work correctly if the image has the same colorimetry as the Viz Engine output, independent from the scene.

How to Work with Images in HDR

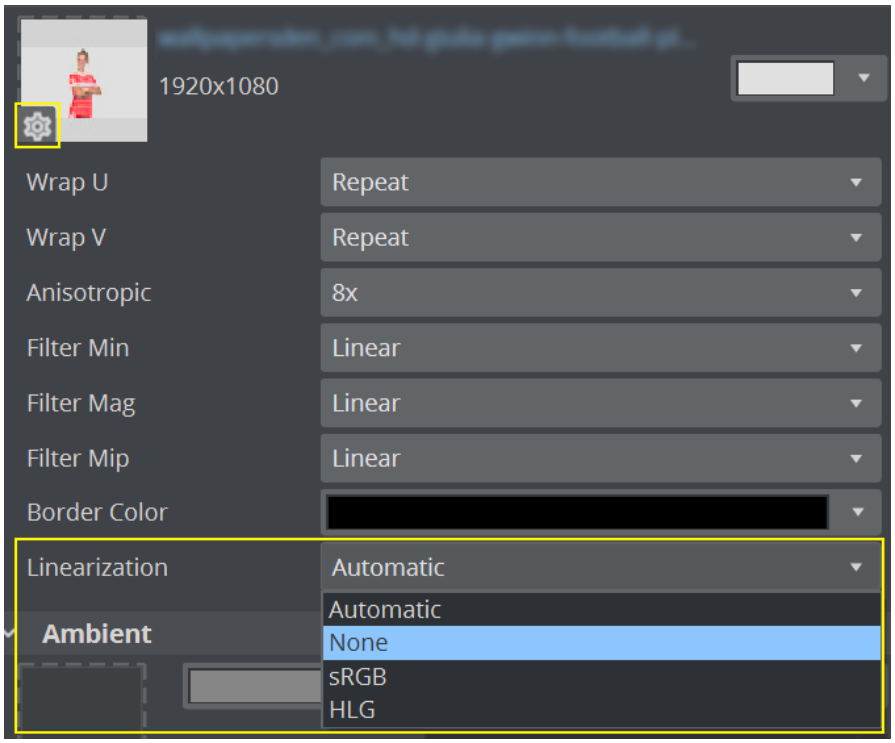
Images for HDR require **at least a 10-bits per channel**, often 12-bit or 16-bit floating-point, 8-bit formats (JPG, PNG) cannot represent HDR properly (banding and clipping occurs).

Copyright © 2026 Vizrt

Images (or Textures) within any scene are automatically linearized into the scene's color space if they are located on Graphic Hub. This happens automatically while loading a scene if the option **Automatic Image Conversation** in the Viz Engine Configuration is enabled: Automatic Image Conversion.

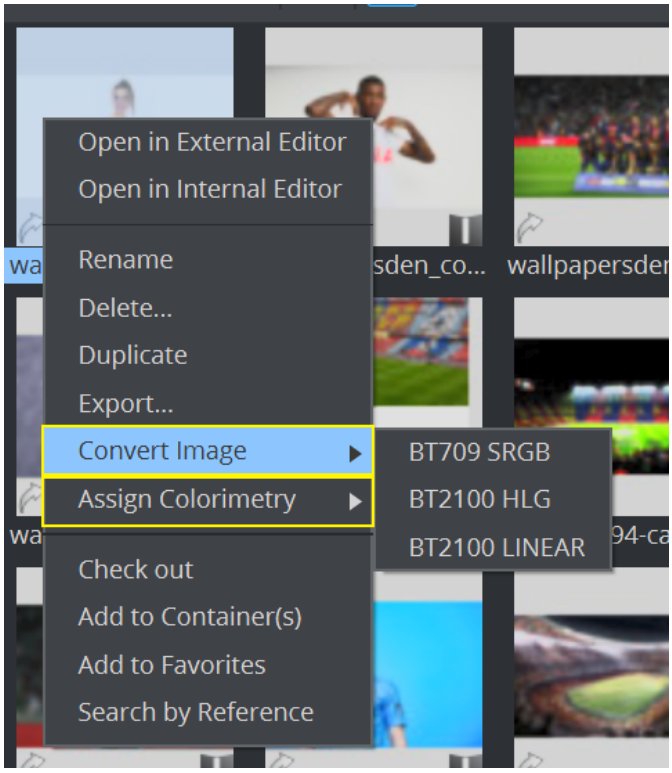
As images require embedding metadata to interpret content correctly, images from other sources (a local file storage) might require a manual setting of the linearization method.

The linearization method can be set to **sRGB** or **HLG** within the texture options of several plug-ins (PBR, Phong, Texture Slot, etc.).

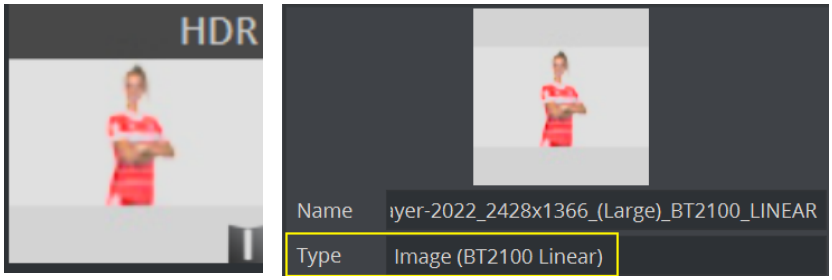


You can also reassign the colorimetry on existing images, or convert images from one color space to another. In Asset view, right click on an image to either re-assign, or convert to a different color space.

Copyright © 2026 Vizrt



Images with correct metadata for HDR are labelled in Asset view, and more details are shown in the properties panel:



Info: To provide a fallback to SDR versions, you need to tone-map HDR images (to avoid clipping or washed-out colors).

JPEG XL Image Support

To utilize the full color space, we recommend using JPEG XL images. JPEG XL is a modern image format supporting wide color gamut, HDR color space, and also supports alpha transparency. Viz Engine 5.4 and later allow to natively import JPEG XL files (*.jxl* files). These files can be created (or converted) for instance on Photoshop v26.8 and later.

Info: *.exr* and *.hdr* images, are used for environment maps rendering only.

Copyright © 2026 Vizrt

References: JPEG XL: Superior Image Compression

Copyright © 2026 Vizrt

9 Adaptive Scene Design



- Introduction to Adaptive Scene Design
- Box Transformation and 3D Transformation
- Working with Different Formats
- Introduction to Flexbox
- Flexbox in Detail
- Designing for Adaptive Storytelling
- Transition Logic and Adaptive Scenes
- Parallel Outputs and Adaptive Scenes
- Adaptive Scene Design Tips and Tricks
- Adaptive Scene Design Troubleshooting

Copyright © 2026 Vizrt

9.1 Introduction to Adaptive Scene Design



Graphics matter. As television fragments and mobile screens become as important as living room sets, 69% of consumers now watch video without sound in public places. Graphic content needs to work harder than ever to convey the story as a result. With audience attention and advertising revenue now split across different platforms, getting those graphics right is more important than ever.

This is becoming increasingly difficult. We've all seen poor quality pictures on LinkedIn or Facebook — a blurry torso, a cut-off headshot — where the resolution is all wrong for the target platform. Broadcasters face that problem x 10. One viewer can be watching a program on a giant OLED screen while another watches the same content on a smartphone on a train. Both demand the same high-quality audio-visual experience, and both also expect to be able to switch at will, to transition from one screen to another seamlessly.

Broadcasters, from linear to OTT, therefore, need to deliver graphics that work and are readable on any and all devices. Even more challenging, they need to do this while maintaining brand consistency across this new multiplicity of formats and without using additional in-house or outsourced resources. While graphics matter, so do budgets.

The solution is to take an Adaptive Storytelling approach, allowing user-generated rules-based automation to adapt content to target platforms without swallowing additional resources. Which is why with the launch of Viz Engine 5 we are introducing Adaptive Graphics; single workflow-multi-platform content delivery that automatically adjusts resolution and format to support specific display devices and effectively enables graphic artists to create once and publish multiple times.

Producing broadcast graphics can be likened to a factory production line. In the current broadcast environment, broadcasters typically find themselves running several production lines in parallel. One serves linear, one serves social media, one is pointing at online, and so on; each producing content but with variations to accommodate different target outputs. There are several problems with this scenario. First, it wastes resources; artists end up repeating work when they could be moving on to other creative projects. Second, everything is multiplied from

Copyright © 2026 Vizrt

equipment cost to carbon footprint. Third, it encourages drift away from a consistent Corporate Identity, as individual users meet different challenges in their own workflows in a variety of ways.

With Adaptive Graphics everything is produced from a single production line and the final output is only locked in as the very last process. Everything up to then is fluid, editable, and — crucially — consistent.

By giving artists the tools to create flexible, adaptive content that automatically adjusts to pre-defined output formats on the fly, broadcasters can collapse multiple, parallel workflows into a single pipeline that automatically optimizes the content for the delivery platform.

Adaptive Graphics in Viz Engine 5 means that graphics will render perfectly for every intended device – in the layout you choose and in the way that just works best. It does not matter whether your target platform is 16:9, portrait, landscape, square, or even the custom resolutions and aspect ratios of modern video walls, your content adapts each and every time.

9.1.1 So, How Does It Work?

In essence, we've added some new tools and capabilities to Viz Artist and Viz Engine that allow elements to be positioned relative in space; either relative to a given viewport, a given container volume or size, or to other objects. The same adaptability can also apply to their size or scaling factors. All this allows your graphic elements to effectively flow into the aspect ratio and screen size of the desired format.

In many ways this is similar to the tools that web developers have had access to for years that allow them to create responsive websites accommodating a variety of target screen sizes. We have taken the concepts behind those and applied them to the production of high-quality broadcast graphics, giving graphic artists the assurance that their work will display as they intended on their target screens.

New features include:

- **Intelligent scaling:** With Viz Engine 5, additional attributes to control graphic element behavior are added via a new Flexbox plugin for Viz Artist. This operates seamlessly within the familiar easy-to-use Viz Artist user interface, and allows behaviors to adapt to pre-defined criteria, vertically stacking score box elements in a mobile display for instance. Minimum and maximum boundary values can also be set for scaling operations, for example min scaling at 20% and max scaling at 120%, to ensure an image stays within an accepted range of quality and clarity.
- **Variable fonts:** Critically, given the importance of text-based graphics in muted viewing environments, there is also full control over all text and font aspects. This goes a long way beyond the current simple tools of text wrapping and scaling depending on a bounding box and introduces Variable Fonts. These give designers precise control over strength, kerning, and more attributes of a font
- **Container nesting:** Containers are also used to structure and group the different behaviors together, with Adaptive Graphics fully supporting container nesting.
- **Texture Fit:** A plug-in to show your images best fit to your aspect ratio.
- **Real-time feedback:** Lastly, Viz Artist allows designers to set up multiple preview windows, giving them real-time feedback on the final design on different target platforms during build and edit stages. With each output governed by a different Viz Engine instance, this gives them compete confidence in the final look across their target platforms.

Adaptive Graphics in Viz Engine 5 allows broadcasters to run a single graphics production line rather than multiple ones in parallel. It enables users to genuinely create once and then publish many times, using an easy-to-use new toolset that allows them complete control over how their graphics will look on everything from a mobile phone in stadia to a giant city center video wall. With Adaptive Graphics, consistency is guaranteed, flexibility is innate, and enhanced fan engagement is inevitable.

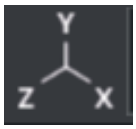
Copyright © 2026 Vizrt

9.2 Box Transformation and 3D Transformation

A new way of designing scenes was introduced in Viz Engine version 5: **Box Transformation.**

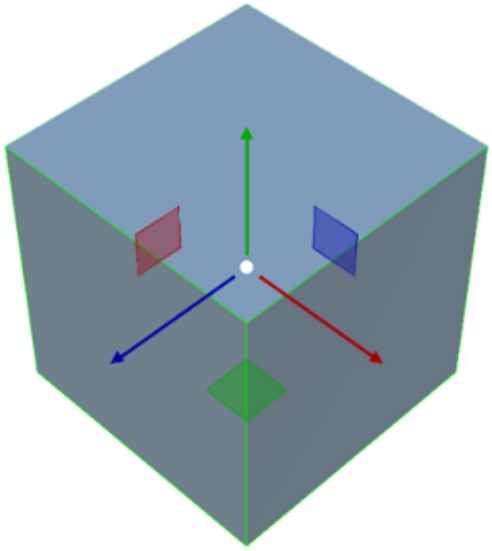
Box transformation and the Flexbox plug-in are the base for adaptive scene design. Viz Engine is now capable of handling screen elements in a similar way to how modern web designers are using it. While classic layout focuses on three dimensions, box transformation focuses on content.

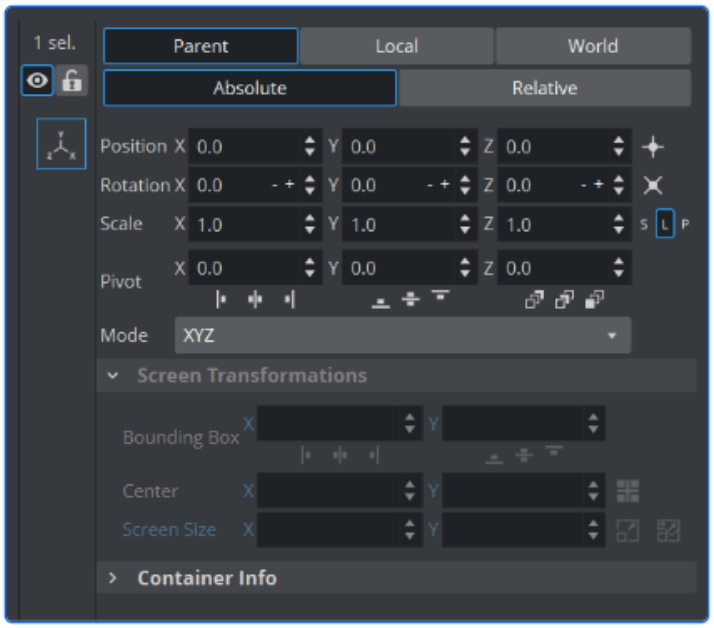
9.2.1 3D Transformation



Three dimensional space defines its objects by XYZ **position**, **scaling** and **rotation**. This space is well known by any 3D artist.

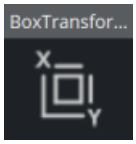
Three dimensional designs are used for virtual sets, AR graphics, simulations etc.





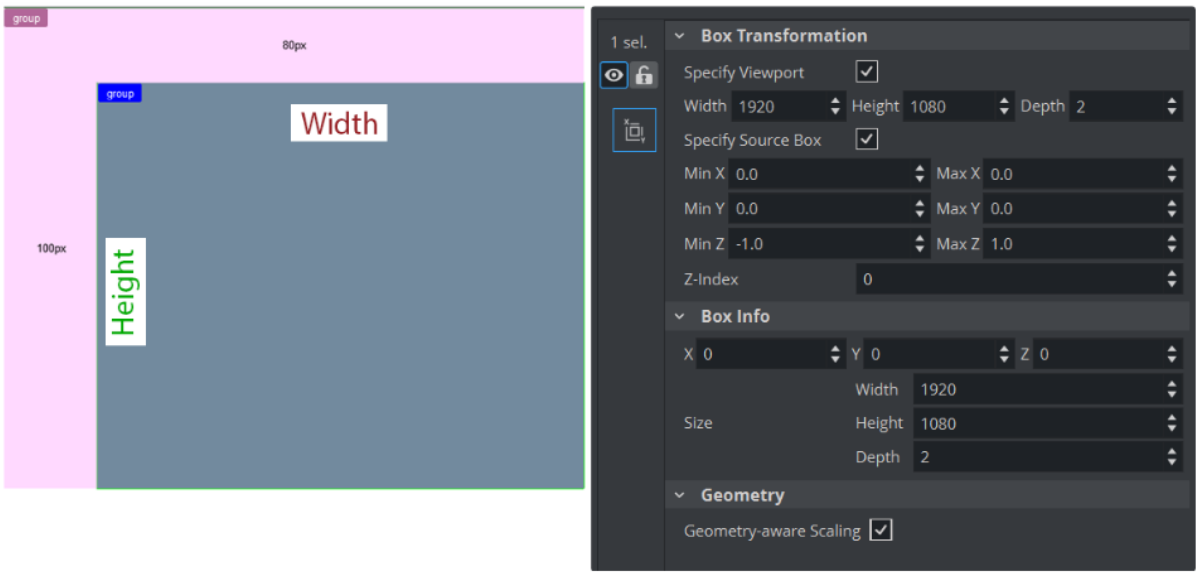
Objects in 3D space can intersect each other and there is no automatic layout. The positioning is absolute and can usually only be influenced by plug-ins or script logic. Measurements are done in units (one vizunit is comparable to one centimeter). Objects in 3D space can be modified in local or in world coordinates.

9.2.2 Box Transformation



The Box Transformation defines its objects mainly by **width** and **height**, the **aspect ratio** and the behavior of the box or container it is placed within, much like modern web design. It allows a flexible layout, independent of the output dimensions and is therefore used mainly for 2D graphics and flexible layouts. Since Viz Artist version 5.1, it can also handle 3D objects.

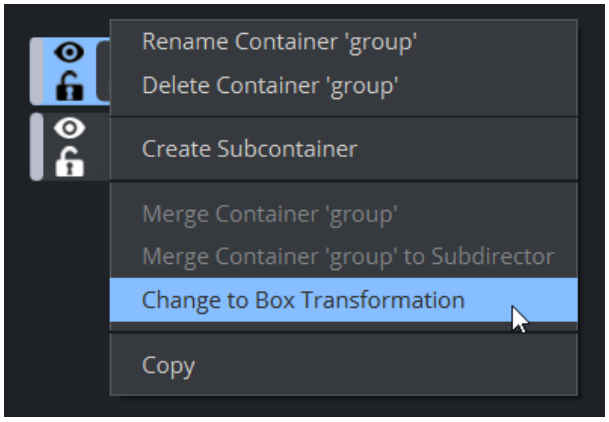
Copyright © 2026 Vizrt



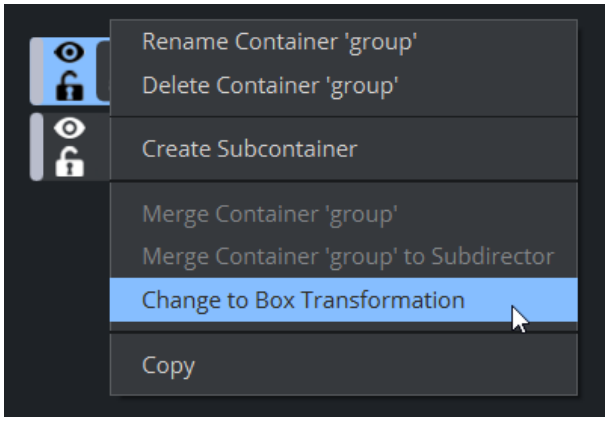
Elements usually do not overlap, rotation is only available in the z-axis and there is almost no depth in the design. The z-axis is only rarely needed for technical purposes. In combination with the Flexbox plug-in, positioning is done automatically and varies from layout to layout. The measurement within a box transformation is done in pixels, or in combination with the Flexbox plug-in in percent too.

9.2.3 To Switch between Box- and 3D Transformation

The type of transformation is automatically switched if a Flexbox plug-in is assigned. You can quickly switch between the different types of transformation modes by right clicking and selecting **Change to Box Transformation** or **Change to 3D Transformation**.



Copyright © 2026 Vizrt



Content-aware Scaling

Box Transformation is able to perform a geometry-aware scaling. Some geometries like Noggi are recognized and scaled correctly:

Noggi scaled without geometry-aware scaling	Noggi scaled with geometry-aware scaling
	

Content-aware scaling is currently supported for:

- Text
- Noggi
- Rectangle
- Filecard

9.2.4 3D Objects Support for Box Transformation

Since Viz Artist version 5.1, Box Transformation can handle 3D objects using the new parameters added for depth and z coordinate control.

We added depth values and z values for Box Transformation and FlexBox plug-ins, in the FlexBox plug-in we have a flag to control if we want FlexBox to control Box Transformation's depth/z values or not, the flag also controls auto modes of Box Transformation.

Box Transformation

1. **min/max z for source-box:** Specifies the min/max values for z in the source box.
2. **Depth for viewport:** Specifies the depth value of the viewport.

Copyright © 2026 Vizrt

3. **Z and Depth for box info:** Specifies the z position and depth value of Box Transformation.

3D Content Section in FlexBox Plug-in

1. **Update BoxTransformation flag:** This flag is enabled by default, if enabled, the flex box controls the z, depth, and auto mode for source-box and viewport of Box Transformation and x/y and width/height values of box info in Box Transformation. If it is disabled, FlexBox just controls the x/y and width/height values of box info in Box Transformation.
2. **Box Transformation Z:** Control the z value of box info in Box Transformation.
3. **Box Transformation Depth:** Control depth value box info in Box Transformation.

Copyright © 2026 Vizrt

9.3 Working with Different Formats

Starting with Viz Engine 5, Viz Engine is able to handle different formats within one scene. A scene holds the logical structure of your graphics as Scene Tree, but can render it in different resolutions and aspect ratios. This, in combination with the layout capabilities of the flexbox system and format independent properties allows the user to create flexible and adaptive scenes for visual storytelling.

Viz Engine hosts different formats for a single scene, properties of containers, assets and plug-ins and much more are bound to certain formats. This gives anyone the flexibility to create different looks for different formats.



- To Create a New Format
- To Preview Formats
  - Working with Multiple Previews
  - Format Dependent Properties
    - Example
    - Information Popup
  - Format Dependent Animation
  - Setting the Default Format
- Format Overview
  - Plugin Properties:
  - Director Properties:

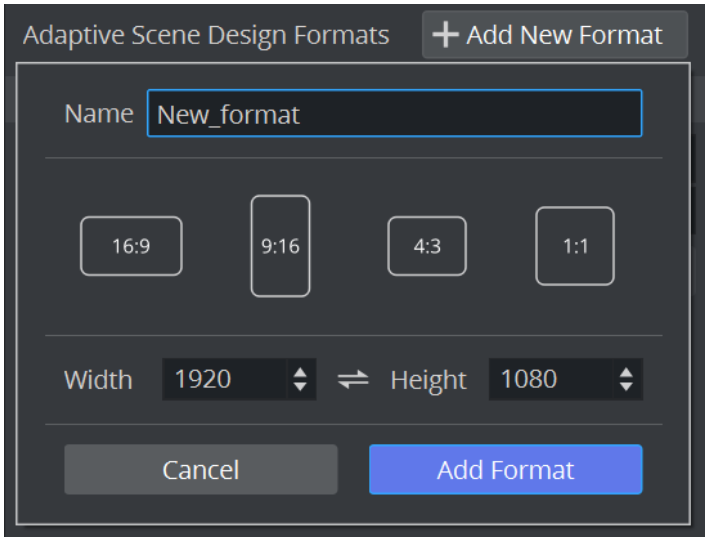
9.3.1 To Create a New Format

Each scene comes with a default format. This is based on the configured output size (for example, 1920x1080 with an aspect ratio of 1.77) for HD resolution. This is the base and default for all properties. In addition to this format, many additional formats can be added to the scene. These formats can vary in dimension, but do not necessarily. For example, a scene can hold three different HD formats, all in the 1920x1080 dimension, but with different logic and looks.

Copyright © 2026 Vizrt

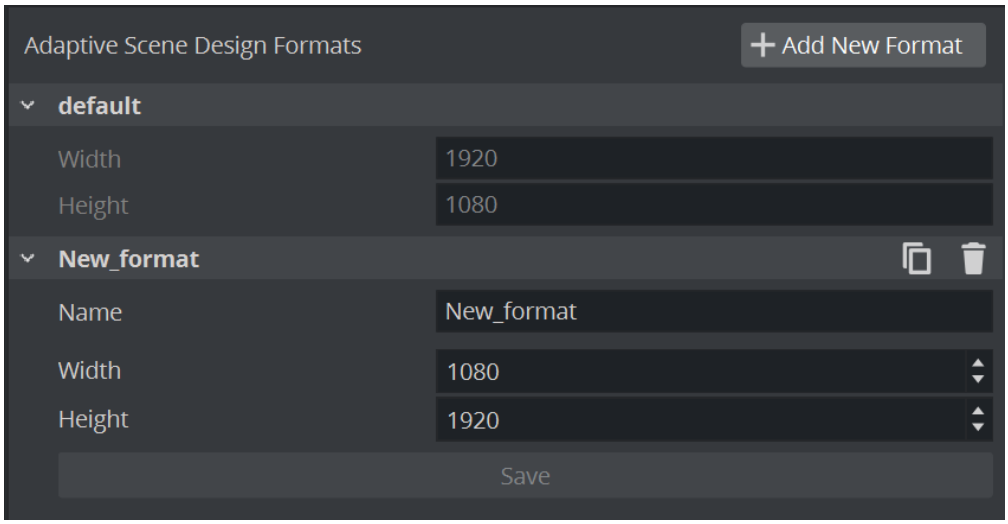
To add a new format, select **Scene** and click on the **Formats** ( ) icon.

The default format is already listed and can not be deleted. To add a new format, click on the button:



Give your format a unique name. Viz Artist versions 5.0 and newer come with presets for the most common aspect ratios. This feature will be extended in future versions to serve more presets and let the user create own profiles. The *Width* and *Height* values are taken automatically from the output system, but can be changed to any other values or switched by clicking on the desired button. The most important factor is the aspect ratio, that is calculated automatically internally and used afterwards by all required plug-ins like the Flexbox plug-in.

Once set, click on the **Add Format** button. Your new format is now listed as available. The values are inherited from the default format.



Any format name, width or height can be changed afterwards. Change the property and press the **Save** button. Each format can also be cloned or deleted afterwards. Once your scene has an additional format defined, it can be used to create and preview your adaptive graphics.

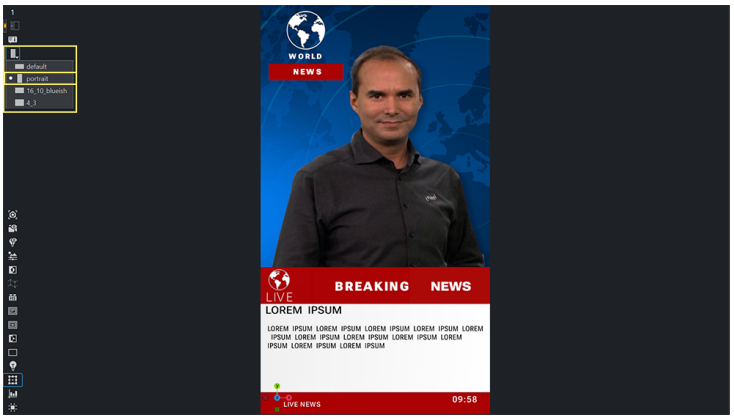
Copyright © 2026 Vizrt

- : Clones an existing format.
- : Deletes an existing format.

The default format cannot be changed.

9.3.2 To Preview Formats

All formats within a scene can be switched in the scene editor. Click on the small format editor next to the scene editor to switch the active format:

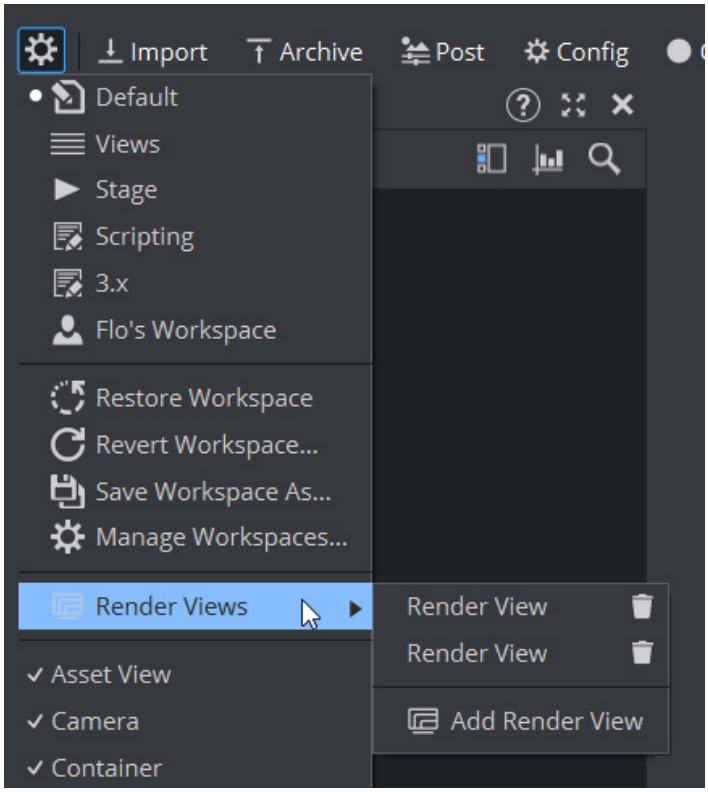
Format	Sample
Default 16:9	
Portrait 9:16	

Once a different format has been chosen, the scene editor changes its size and display the scene with this new format applied. The stage also shows all animated properties for this format. The chosen format in the scene editor is the active working format, any parameter changed is tied to this format. This also affects animations.

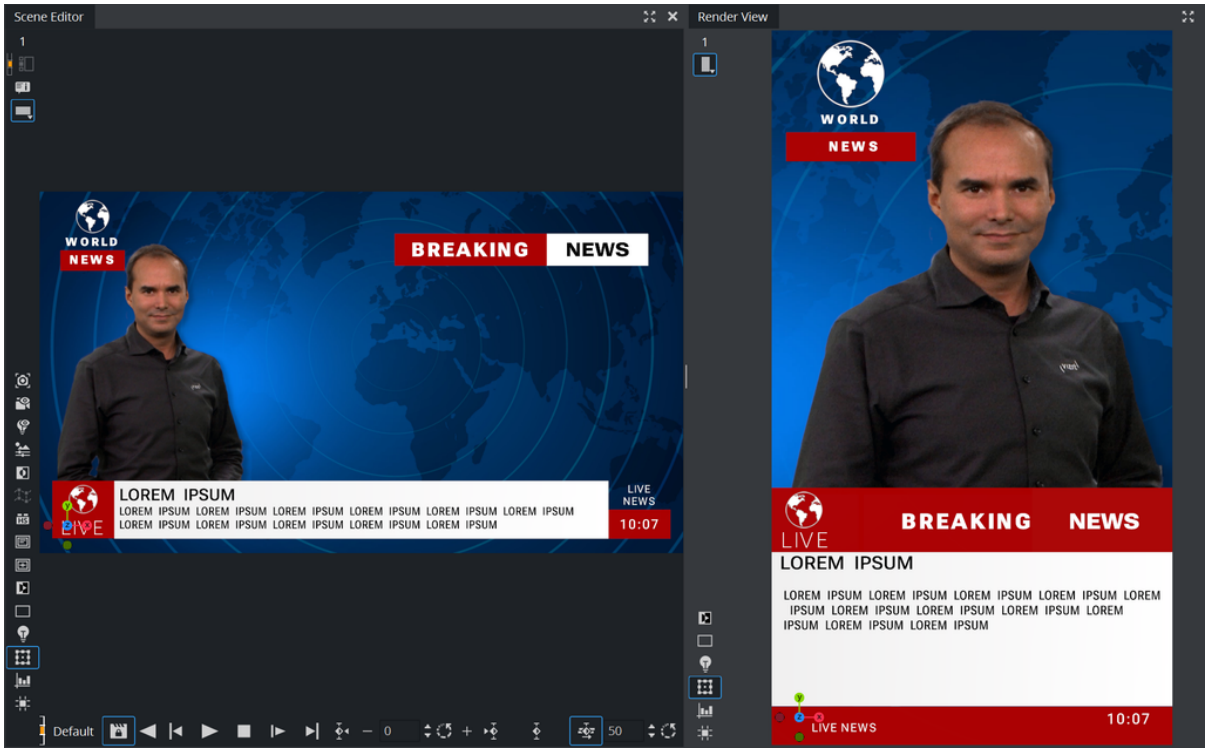
Working with Multiple Previews

To see the effect of your changes while working on the format within the scene editor, additional preview windows can be opened. To do this, choose **Render Views** > **Add Render View.**

Copyright © 2026 Vizrt

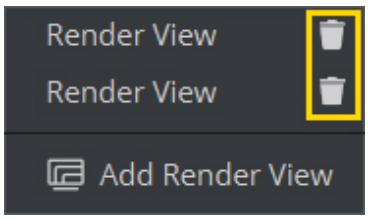


This adds a new render view. To launch it, click on it and you'll get a new floating window hosting a preview editor. This window can be also docked:



Copyright © 2026 Vizrt

If the preview windows are not used anymore, delete them by using the small trash bin icon.



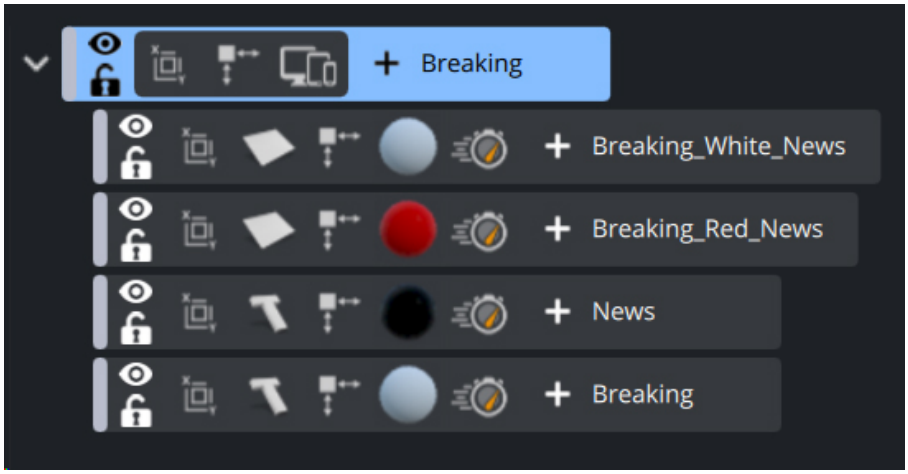
Information: The Render Views are not only available during designing for adaptive graphics. They can also be used for example in Virtual Sets to verify different camera positions and shots.

Format Dependent Properties

To benefit from these different formats, Viz Engine is capable to handle different properties within different formats. This means in other words, any property has its own look within different formats.

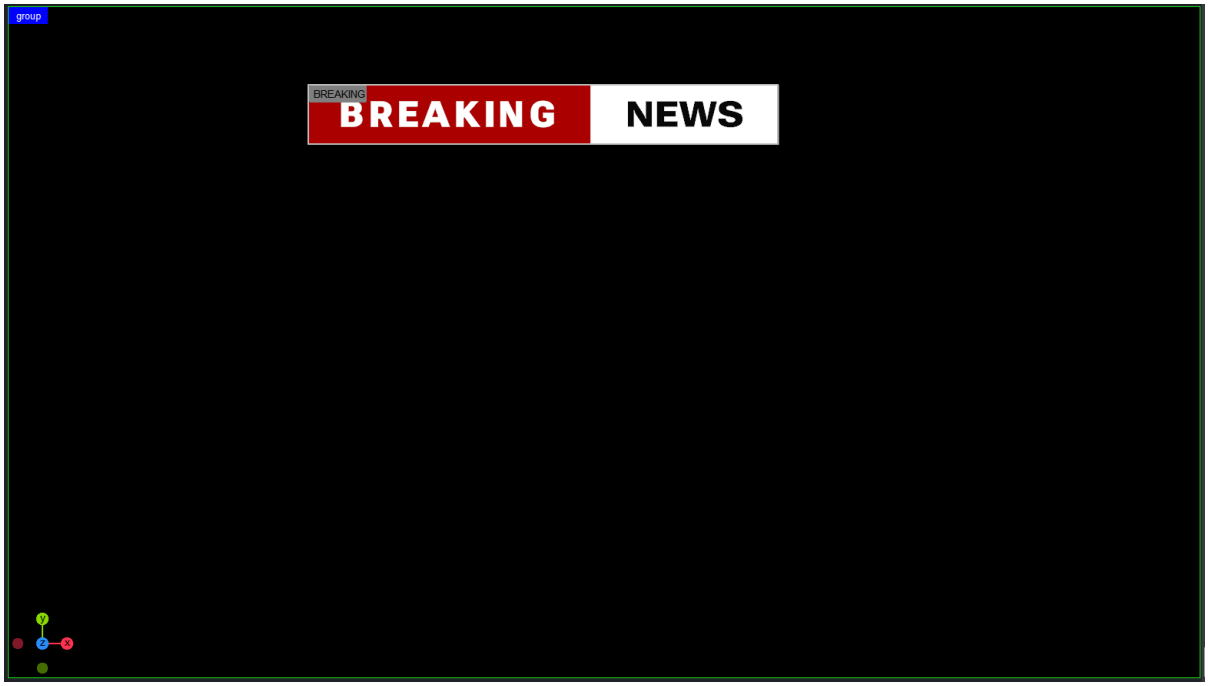
Example

A basic scene, with a rectangle and material and some text on it:



Looks like this in the default format:

Copyright © 2026 Vizrt



Now create a new format and call it *Version 2* by either cloning the default format or create a new one. The dimensions can be different, but this is not required.

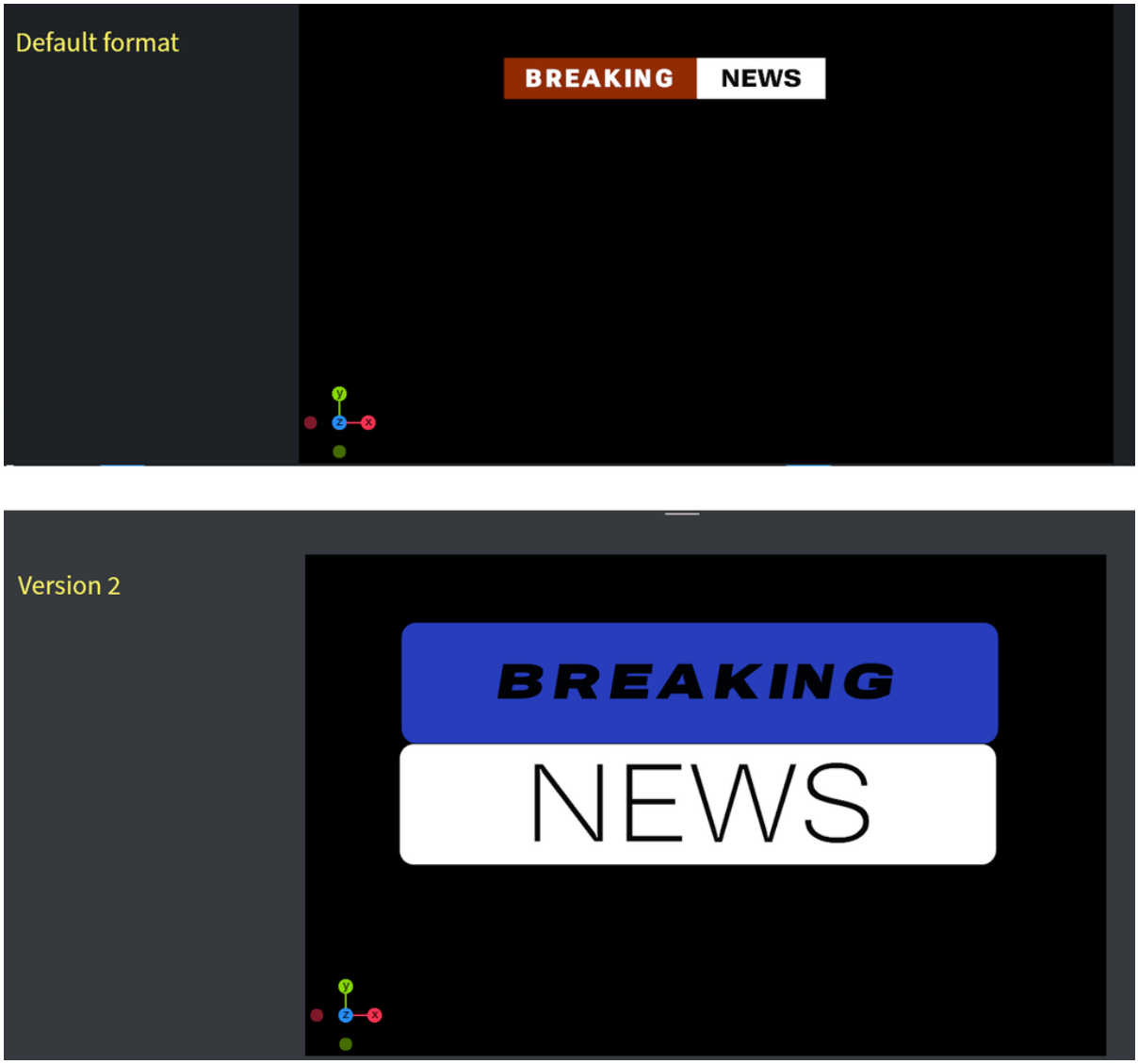
Switch to the *Version 2* format. It still looks the same as the default one. Click on the red material we defined for the "Breaking" background and change it to any other color, open a preview window and have the default and *version 2* side by side:

Copyright © 2026 Vizrt

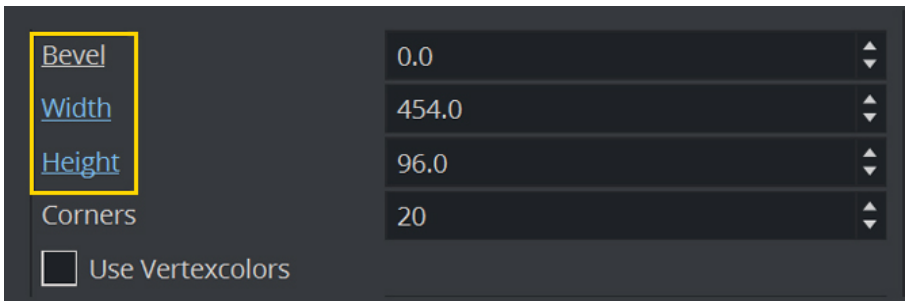


Any parameter that is able to be animated can also be format independent. This also includes properties for variable fonts, geometry properties or layouting options of the Flexbox plug-in:

Copyright © 2026 Vizrt



The same scene structure can look completely different in different formats. To get an overview of what is different in the various formats, an information panel is available. This is shown in the User Interface as underlined property:



This indicates that:

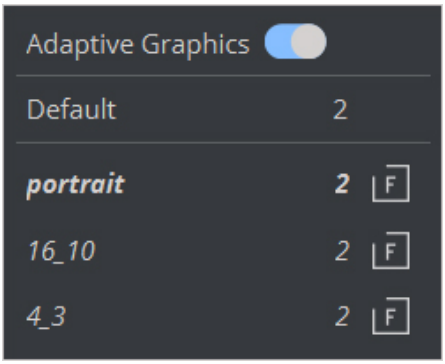
Copyright © 2026 Vizrt

- Bevel has different values in other formats.
- Width and Height have different values in other formats, and do not have their default values.

In default format, a property label is shown as underlined if it differs from default in any other format. In other formats, a property label is shown as underlined only if it differs from default in the current format.

Information Popup

Right clicking on the label opens the information popup:



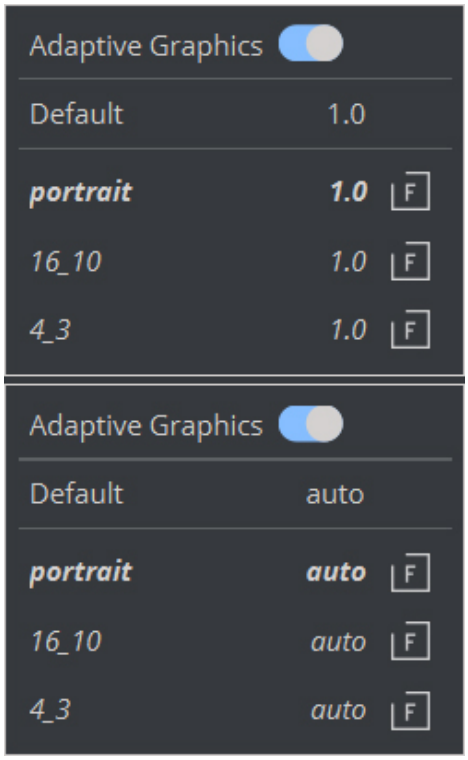
The following information is shown:

- Format dependency can be turned on or off.
- The default value is always shown and all other values are listed.
- If the value is the same as the default, it is shown in italic.

To make a value independent from the default format, click the F button. Clicking on the **Reset** button reverts the value back to the default value.

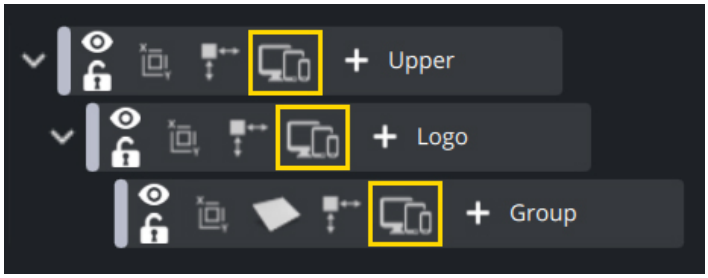
As the flexbox plug-in offers the possibility to change a value and also the mode for it, the popup menu has been extended to also support this:

Copyright © 2026 Vizrt



The value in default format is **20%**, whereas in Portrait mode it has a value of **857 pixels**.

Containers with different properties for different formats are marked with an icon:



An overview of all properties will be implemented in a future version.

Information: Media Assets like clips are not format dependent. If you want to use different clips for different formats, you need to use different clip channels and set the visibility per format.

Format Dependent Animation

Along with the possibility to store different values, also the animation can be different from format to format. The same property animated in Format A can be animated in a complete different way on format B.

Copyright © 2026 Vizrt

Warning: There is only one timeline (Default) available, no matter how many formats are defined. Therefore, if a value is being animated beginning with 0.0 to 1.0 and the same value is animated in another format from 0.0 to 2.0, playing the animation still only runs it from 0.0 to 1.0. Please consider this information when starting to animate in different formats.

Setting the Default Format

The default format can be overwritten by setting its name in the **Output Section** of the Configuration dialog. This forces the renderer to show a certain format if it can be found within the scene, if not, it falls back to the default format. This can be used to have a couple of Engines/instances rendering different output formats.

9.3.3 Format Overview

When clicking on the format icon , an overview of all format-dependent properties is shown:

The overview panel has two categories: **Plugin Properties** and **Directors**

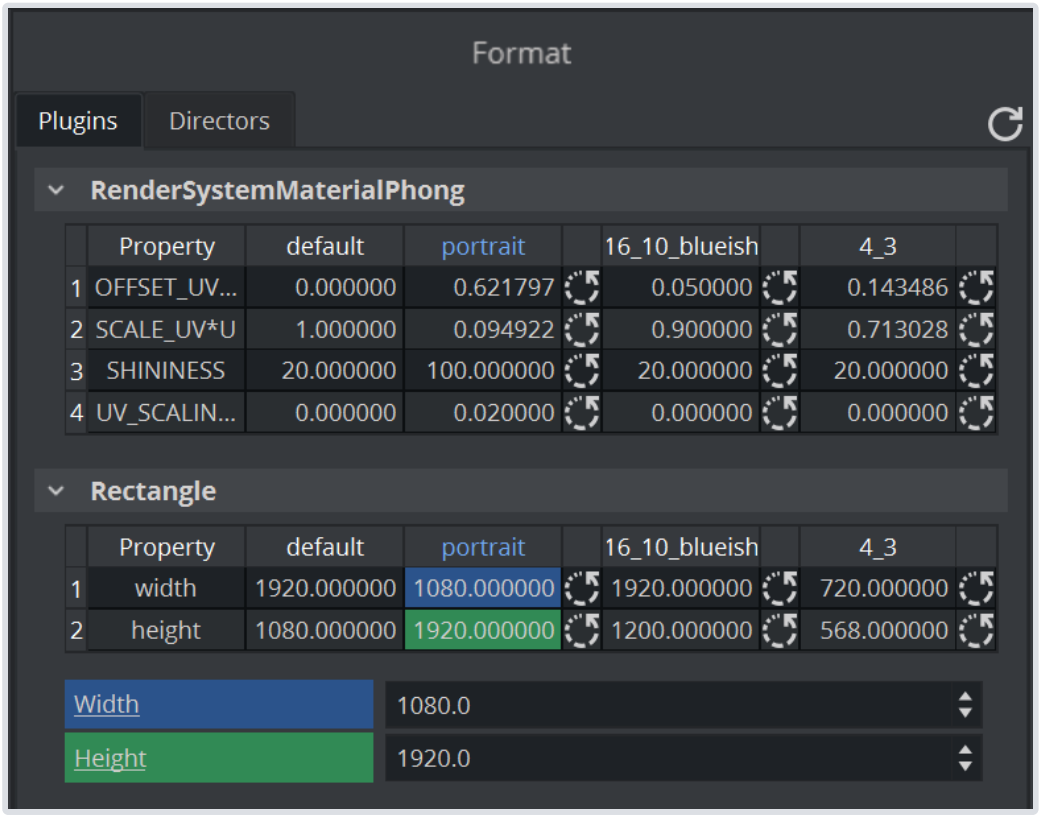
Plugin Properties:

The Plugins panel lists all format dependent properties:

The following sample has 4 different formats and shows differences on the Phong Material and the Rectangle Geometry.
The current selected format is shown as blue (portrait) and the Rectangle geometry differs from the default format in width and height. They are marked in blue, green, red, … labels for better readability. To change e.g. the width on the rectangle geometry, simply change the value.

Editing inside the table is not supported

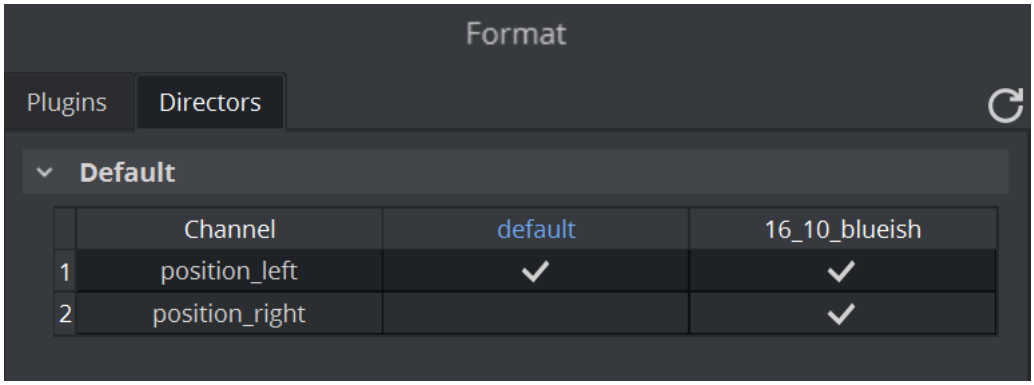
Copyright © 2026 Vizrt



The reset buttons will reset the value to the default one.

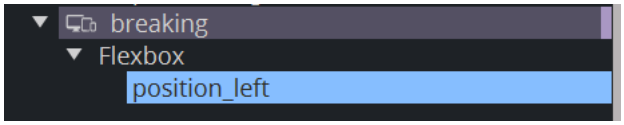
Director Properties:

The director panel shows an overview of what properties are animated in different channels.



Format dependent animations are also marked inside the stage:

Copyright © 2026 Vizrt



The *position_left* is animated in the “default” and in “16_10_blueish” format. the right position is only animated in the “16_10_blueish” format.

Due technical limitations this view is only available in the default format

9.4 Introduction to Flexbox

To adapt you graphics to different aspect ratios, resolutions and layouts, the classic 3D transformation is not flexible enough. The layout has to be done by using plugins or customized scripts. Absolute pixels values used for position or similar do not work as soon as the output format changes.

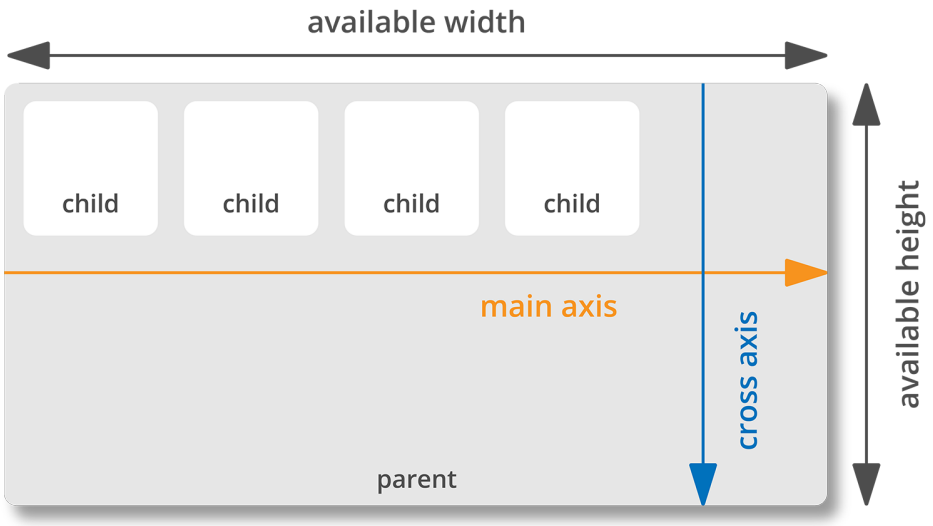
Modern web page design uses a very flexible approach by using nested boxes to define a layout. These layouts are very responsive, can automatically adapt theirs sizes and the way how to show its content. Positioning is no longer based on absolute pixel values, instead a flexible layouting algorithm uses aspect ratio, percentages and pre-defined behavior to render the content, even without knowing the final resolution. Scenes based on Flexboxes form-fit, based on the output format(s).

Building scenes based on boxes is a complete new approach and might be a bit tricky to start with. In the other hand, it solves problems where one typically had to use different plugins or even needed to write custom logic.

Information: The order of your tree structure has an influence on how your boxes are aligned. This does not apply to the classic XYZ-layout.

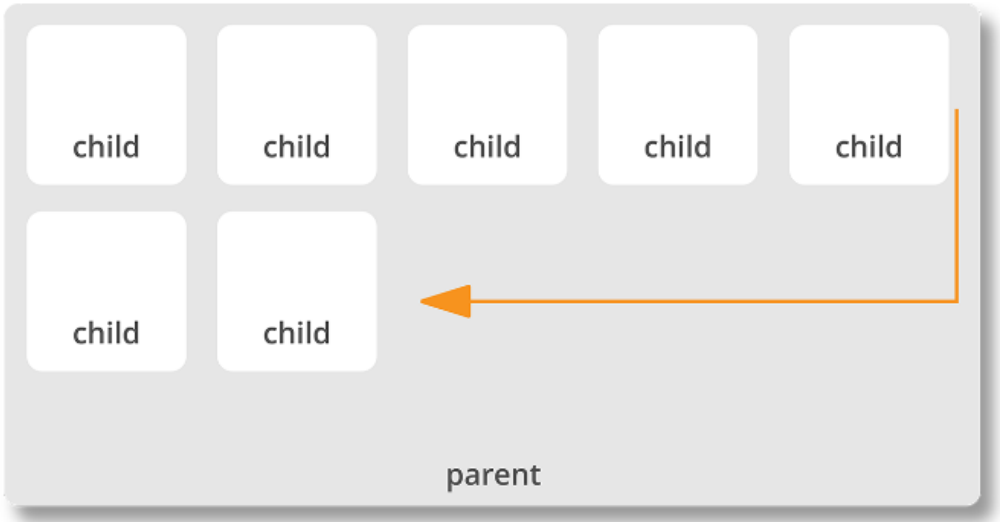
The flexbox integration is based on state-of-the-art web technologies, therefore a lot of documentation is available as well as simulators to test a concept.

Using boxes on the container allows to alter its content width and height to best fill the available screen space. Flexbox layout is direction-agnostic. Lets have a look onto the following picture:

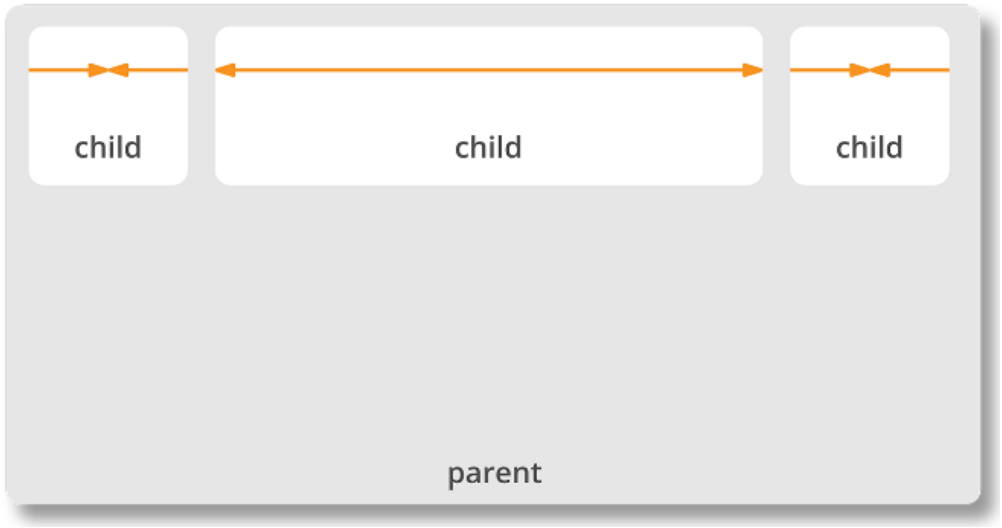


The alignment of any children is based on the direction of the main axis. The flex directional flow can be specified as left to right, right to left, top to bottom or bottom to top or in other words in columns or rows. Moreover, the flex layout is less restrictive in terms of content flow. The layout can adapt to the available width and height, boxes can adjust if their content won't fit.

Copyright © 2026 Vizrt



Boxes can adjust in size by either decreasing, to avoid unnecessarily monopolizing space, or increasing to make room for contents to be constrained within its boundaries. Individual items within a flex container may also be automatically reordered and rearranged to suit the available layout space



The Flexbox follows a two dimensional alignment, however it is possible to still use it within a 3D environment. Looking at a typical broadcast graphic, the layout can be split into the following boxes:

Copyright © 2026 Vizrt



With the flexible layout and the possibility to work with aspect ratio, the graphics can easily adapt to different formats:





Information: The Flexbox plug-in requires a valid license and is available for the Viz Engine Render Pipeline only!

Copyright © 2026 Vizrt

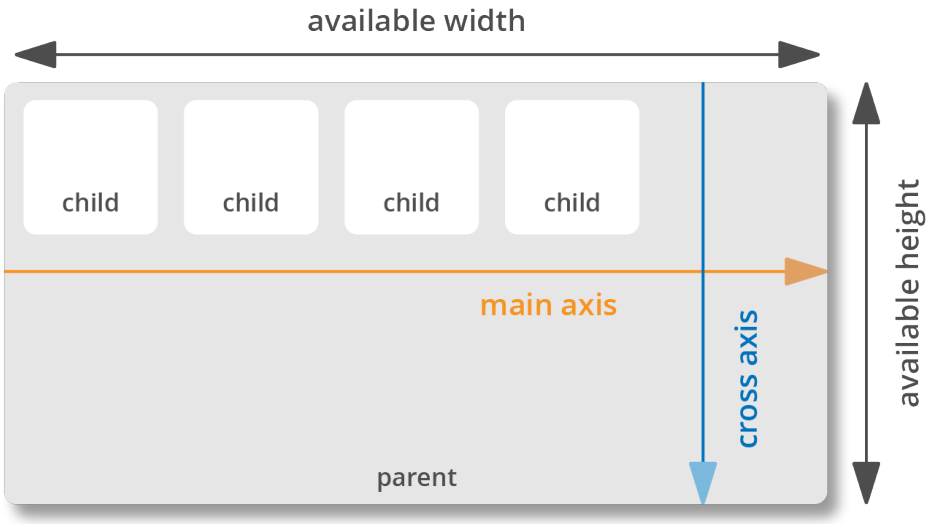
9.5 Flexbox in Detail

This chapter explains all settings of the Flexbox plug-in in detail.

- Flex Wrap
  - No Wrap
  - Wrap and Wrap Reverse
- Flex Direction
- Direction
- Flex Basis
- Growing and Shrinking
- Handling of 3D Objects
- Aspect Ratio
- Width, Height and Limits
- Margins, Padding and Borders
- Positioning
  - Tracking
- Justify Content
- Align Items
- Align Self
- Align Content

The flexbox plug-in takes advantage of the Box Transformation and allows to build graphics for adaptive storytelling. It is based on the flex implementation of CSS and web development. Therefore you can find a lot of documentation, tips and tricks online.

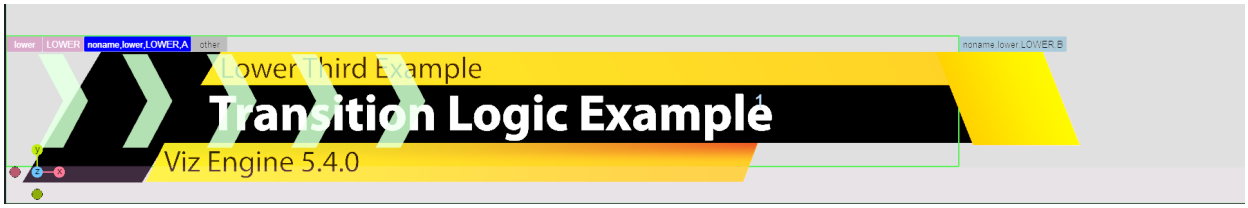
Layout in Flexbox uses two axes, the main axis and the cross axis. The alignment of any child items along those two axes depends on the available height and width



The Main Axis is defined by the direction property. If the direction is set to row, the Main Axis is horizontal. If set to column, the Main Axis is vertical.

Justification defines how items are aligned along the Main axis. Alignment defines how items are aligned along the Cross axis.

The plug-in shows an indicator overlay to visualize the flow direction:



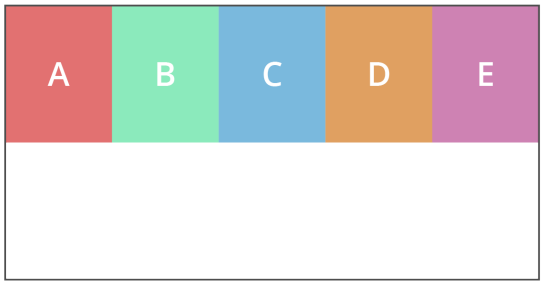
9.5.1 Flex Wrap

This defines the Cross Axis direction. By default children are scaled down to fit into a box. By changing the wrapping mode, children are automatically aligned based on the setting. The following examples use boxes A-E are 300px in width and a minimum width of 150px, but they can grow and shrink.

No Wrap

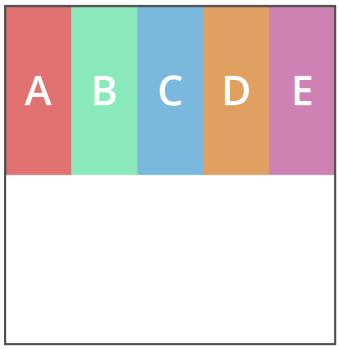
If no wrapping is used, the boxes are scaled down until they reach their minimum width (140 pixels).

Parent 100%

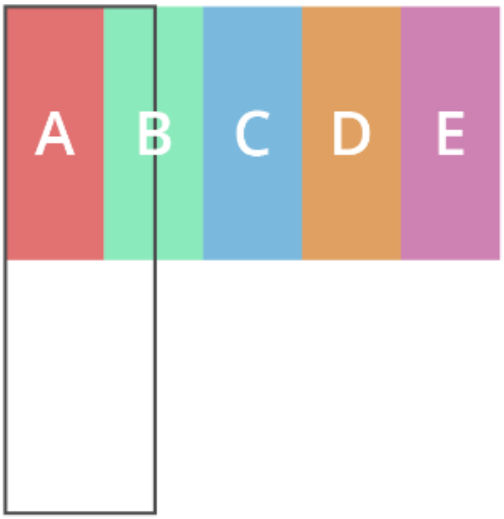


Parent 50%

Copyright © 2026 Vizrt



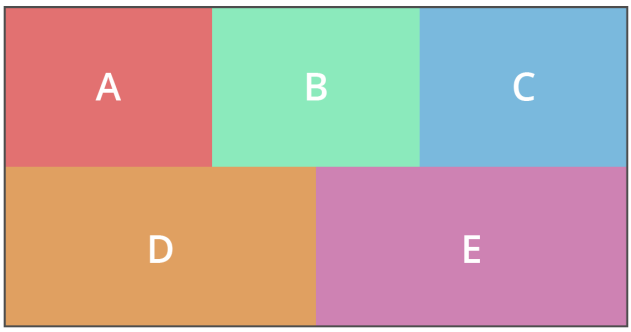
Parent 20%



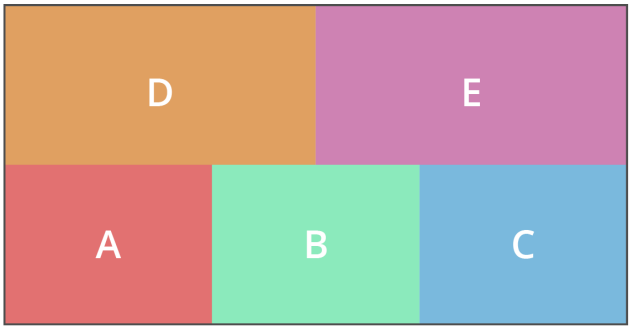
Wrap and Wrap Reverse

If wrapping is turned on, the boxes are scaled down until they reach their minimum size and are then wrapped to fill the space.

Wrapping enabled - Parent 50%



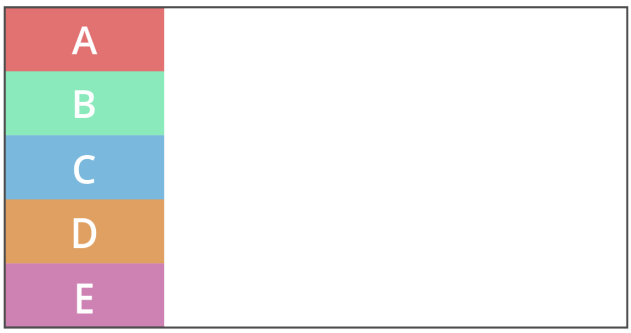
Wrapping Reverse enabled - Parent 50%



9.5.2 Flex Direction

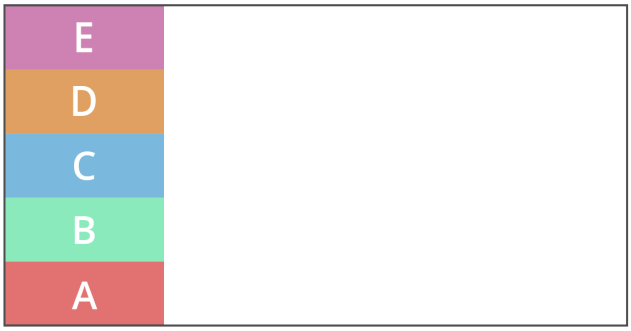
Defines the Main Axis direction. Layouts can be based on **Rows** or **Columns** for all child boxes. This means that all children are placed within a row or within a column. The Flex Direction defines the main axis and defines the direction of all children placed underneath the flexbox container.

Column

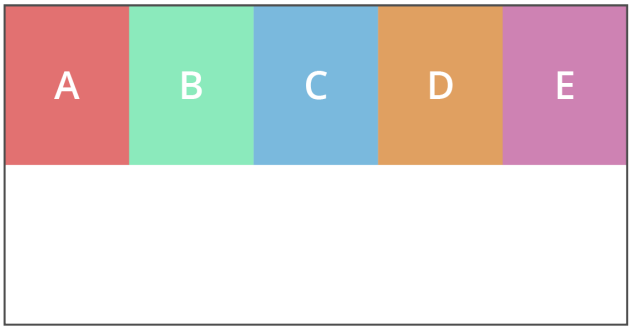


Copyright © 2026 Vizrt

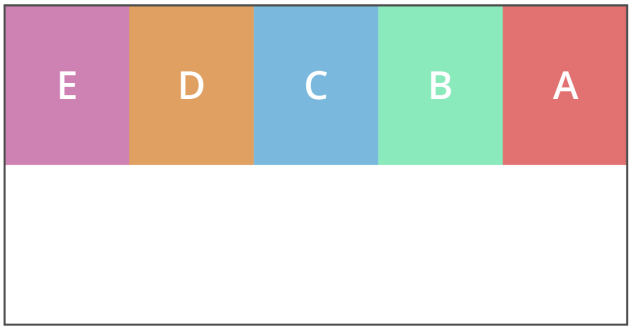
Column Reverse



Row



Row Reverse



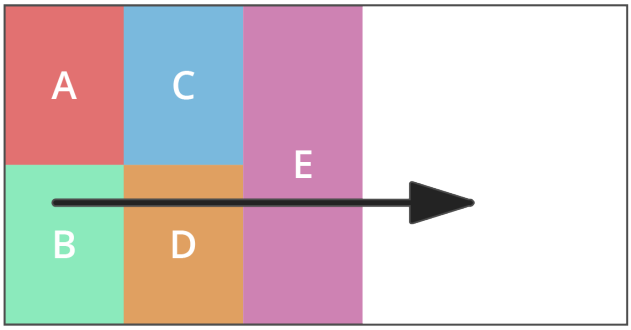
Copyright © 2026 Vizrt

9.5.3 Direction

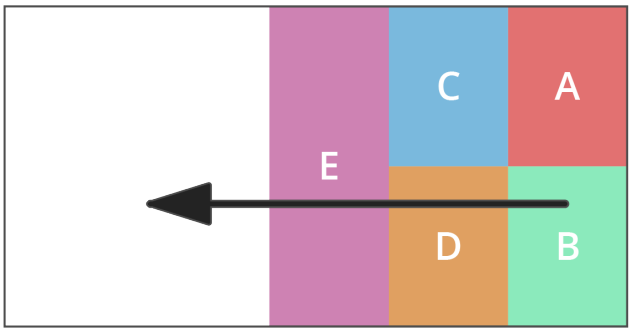
The direction property defines the main axis. Your children can be aligned starting from the **left** or from the **right** side. By default it **inherits** the setting of the parent flex container.

In **Column mode** this is being flipped and the main axis is going **top** to **down** or reverse.

Left to Right



Right to Left



9.5.4 Flex Basis

The basis value defines the default size of a container along the main axis before any growing or shrink is performed. It can be either *auto* or a given value. This value is like setting the width if flex direction row is used or height or column layout is in use.

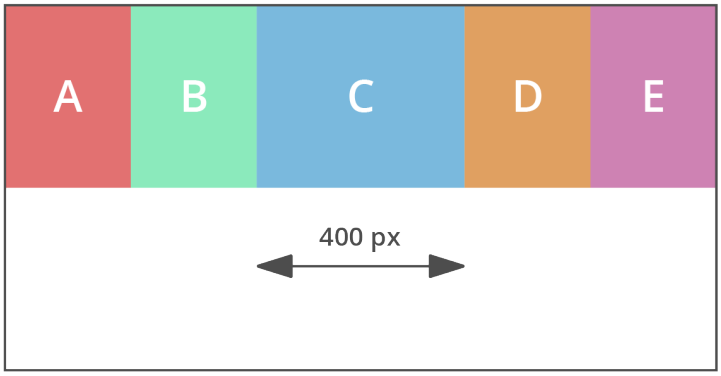
9.5.5 Growing and Shrinking

Defines if and how strong a container can grow or shrink to fit into a parent container. The minimum and maximum width/height values are taken into account. If growing is enabled for all child containers, the remaining space is distributed equally to all children.

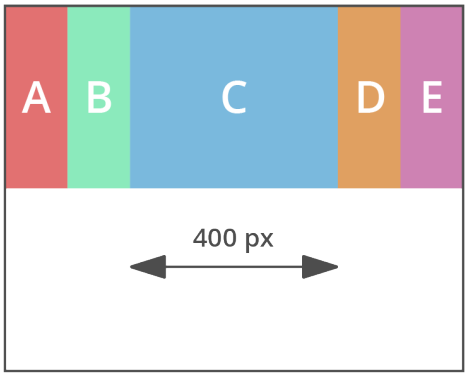
Copyright © 2026 Vizrt

The value is used as a proportional factor. For example if one container has a shrink value of `2`, it shrinks twice as much as all other containers (see below).

Parent container on 100 %
No wrapping
Container C has growing = 0 and Shrinking = 0, size is set to 400 px



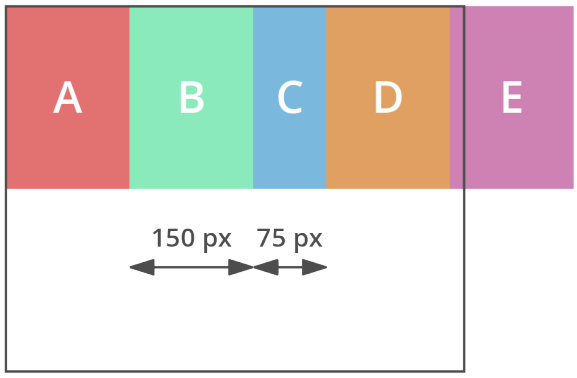
Parent container on 60 %
No wrapping
Container C has growing = 0 and Shrinking = 0, size is set to 400 px



Scaling of container C stays, whereas all other containers shrink to fit into parent dimensions.

Parent container on 60 %
No wrapping
Container C has growing = 0 and Shrinking = 2.0, size is set to 400 px

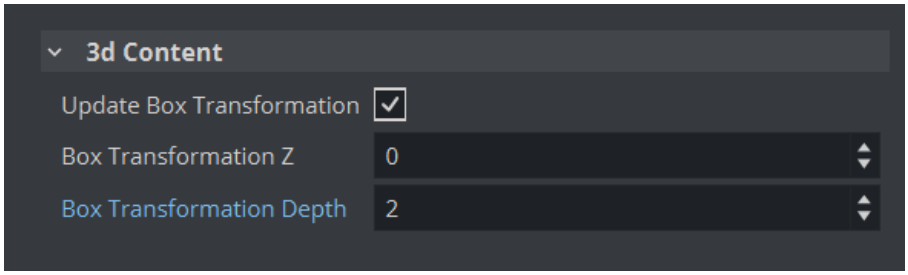
Copyright © 2026 Vizrt



Container C is scaled down twice as all other containers (shrink = 2.0)

9.5.6 Handling of 3D Objects

Since Viz Artist version 5.1, FlexBox can control the Box Transformation depth and z values. If the **Update Box Transformation** flag is enabled, the values of Box Transformation are overridden by FlexBox in each field. This flag does not affect the width/height and x/y values of Box Transformation being overridden by the FlexBox plug-in.

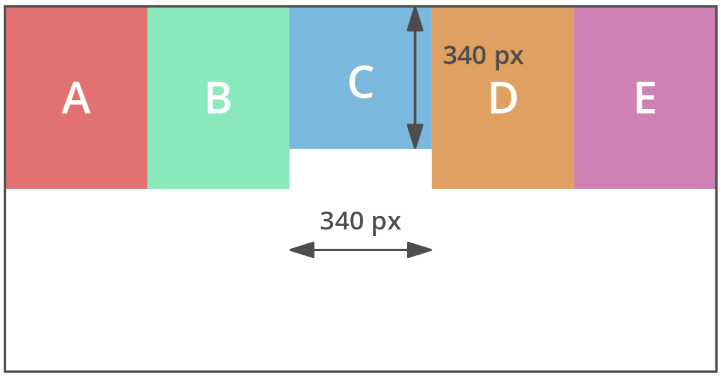


9.5.7 Aspect Ratio

Defines how a flexbox is being scaled. If it is on auto, the width and height taken as it is. If a value is used, it defines the ratio between width and height. For example: An aspect of `2.0` means the width is twice the value of the height.

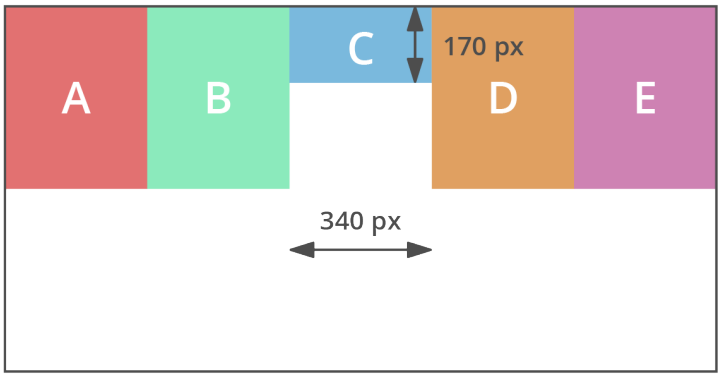
Aspect ratio = 1

Copyright © 2026 Vizrt

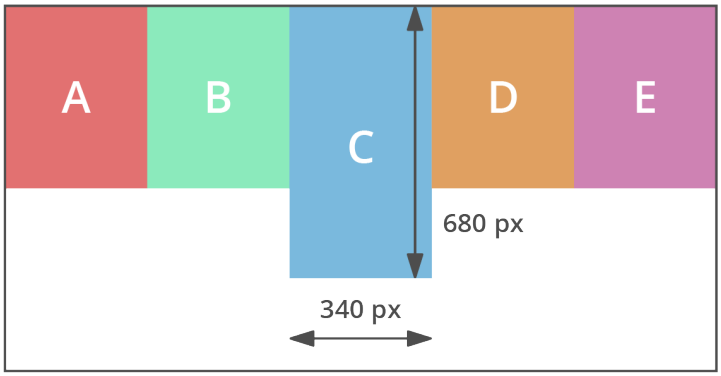


Width and height are the same.

Aspect Ratio = 2

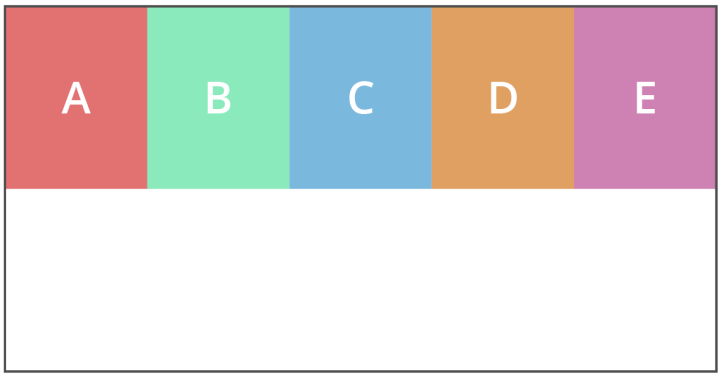


Aspect Radio = 0.5



Copyright © 2026 Vizrt

Auto



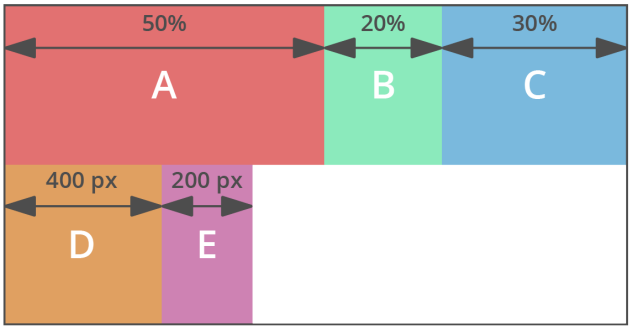
9.5.8 Width, Height and Limits

The sizes for width, height, minimum width and height as well as the maximum width and height can be set to be either auto, in percentage values or in pixel sizes.

- **Auto:** The size for the box is being calculated automatically based on grow and shrink values, the content as well as the aspect ratio.
- **Pixels:** Defines the width/height in pixels.

Note: This dimension can be influenced by other properties.

- **Percentage:** Same as pixels, but based on percentage values.



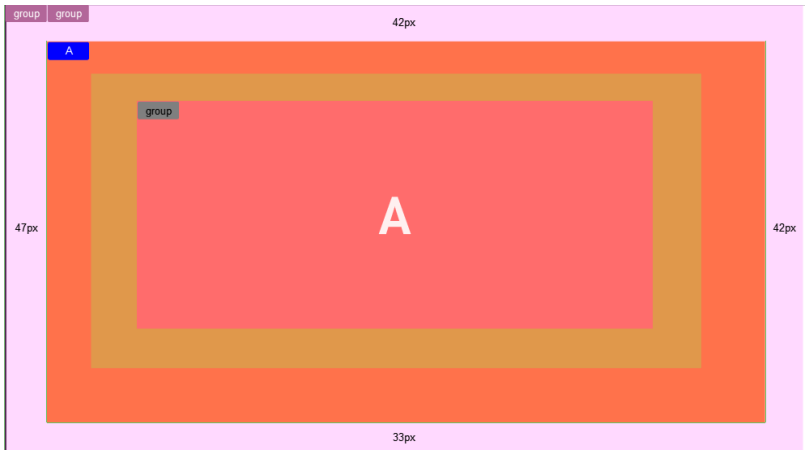
The minimum and maximum values size set the upper and lower size constraints of a container. These properties have higher priority than all other properties and are always respected.

Copyright © 2026 Vizrt

9.5.9 Margins, Padding and Borders

Each container can have some offsets by adding either a margin, padding or border.

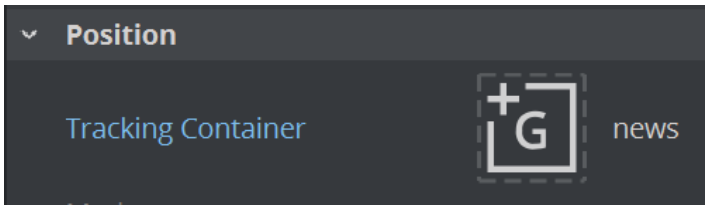
- **Margin**: Adds an offset to the outside of the element. It offsets itself and all children from the calculated position. It adds to the total size for the element. it can be set to **auto**, **pixels** or **percentage** values. Negative numbers are allowed too.
- **Padding**: Affects the size of the container it is applied to. It does not add to the total size of an element if it has an explicit size set. It can be set in **pixels** or **percentage**, but can not be negative.
- **Border**: Does more or less the same as padding. It can only be set in pixels. The main purpose of having an additional border is to keep compatibility with the original flexbox implementation.



9.5.10 Positioning

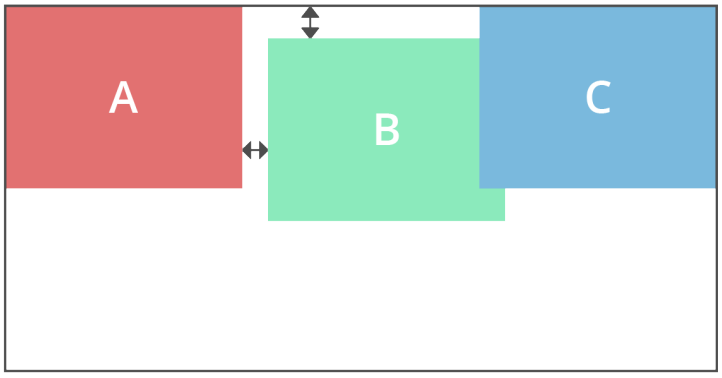
Tracking

To bind one container to another, Flexbox supports tracking. You can specify a reference container and your container follows this position. All other properties are disabled, but offsets can still be applied.



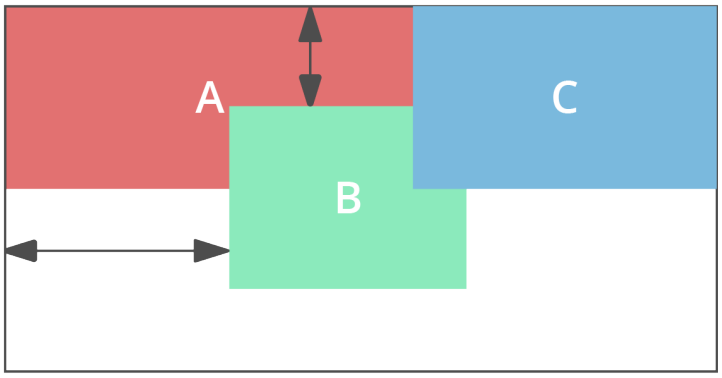
Adding an offset to a container shifts the whole container and its content based on the calculated position by the given values. This offset can be either relative or absolute.

Relative offset



Offsets can be set as **auto** (calculated automatically when using certain content alignments) or as value in **pixels** or **percentage**.

Absolute offset



Takes out the container from the calculation and sets the position to the given values. It can be set as **pixels** or **percentage** (*auto* is available, but is ignored).

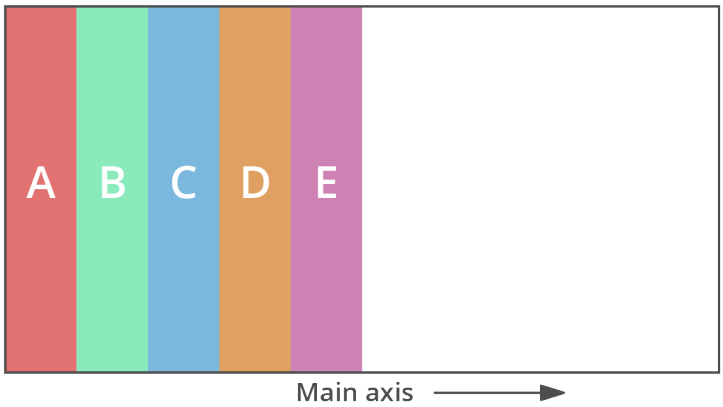
When using absolute positioning, all other properties that influence the position are no longer effective.

9.5.11 Justify Content

This setting describes how child containers align along the **Main Axis**. This can be used to automatically center items for example. The samples below show five child containers with a width of 10% and a height of 100%. The following settings are available:

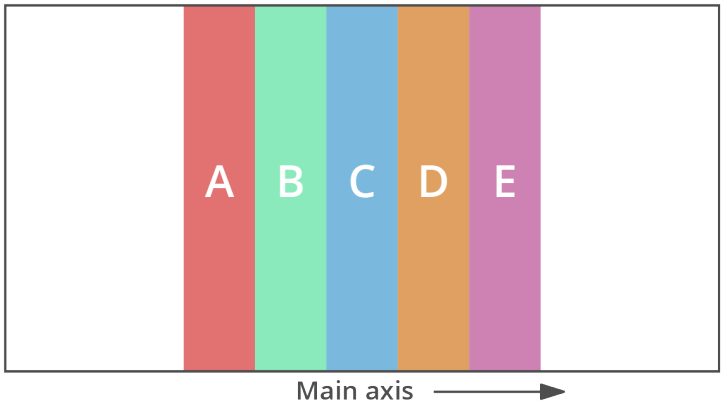
Flex Start

Copyright © 2026 Vizrt



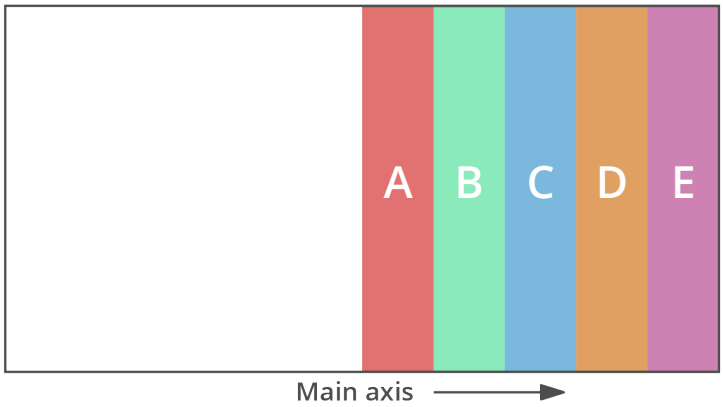
Aligns child containers to the **start** of the main axis.

Center



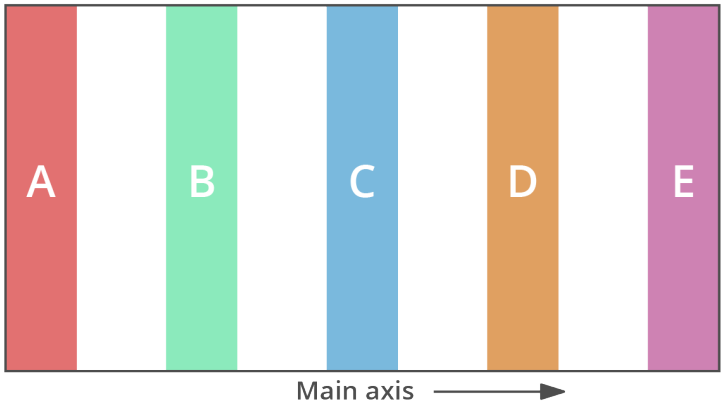
Aligns child containers to the **center** of the main axis.

Flex End



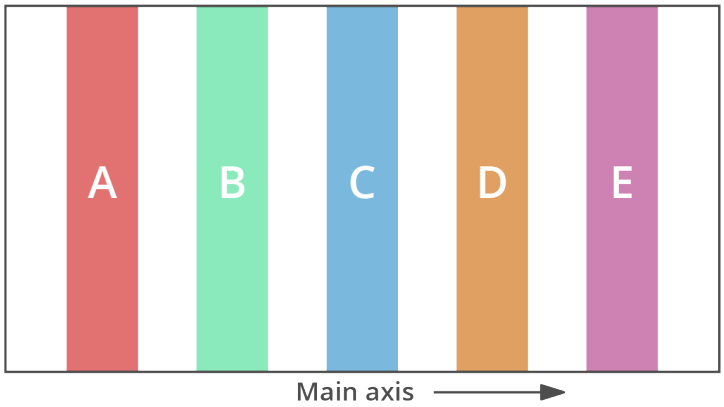
Aligns child containers to the **end** of the main axis.

Space between



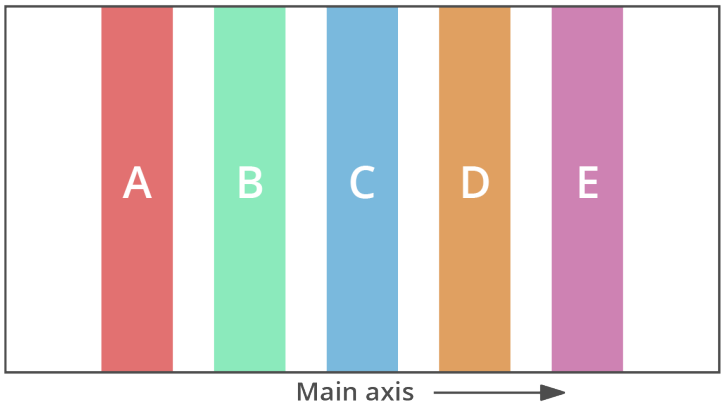
Distributes the remaining space between all child containers.

Space around



Distributes the remaining space between all child containers, including start and end as one element

Space Evenly



Distributes the remaining space between all child containers,
space between all elements including start and end are exactly the same.

9.5.12 Align Items

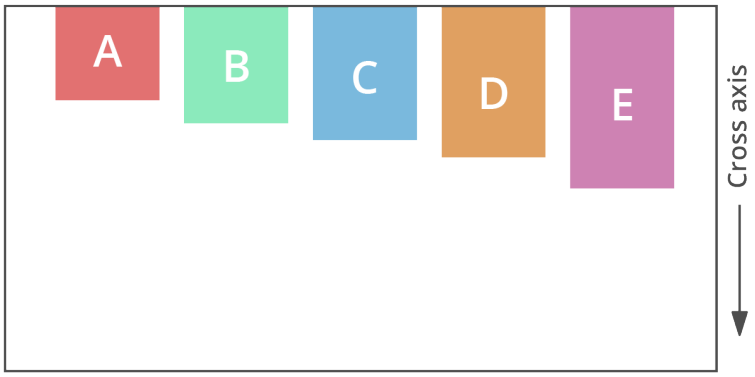
This setting describes how child containers align along the **cross axis**. It sets the default alignment for all child containers, however it can be overwritten by a children using the Align-Self property. The samples below show five child container with a width of 10% and a height of 25%, 30%, 35%, 40% and 45%, Justify content is set to *space around*. The following settings are available:

Auto

Uses the align item of the parent container.

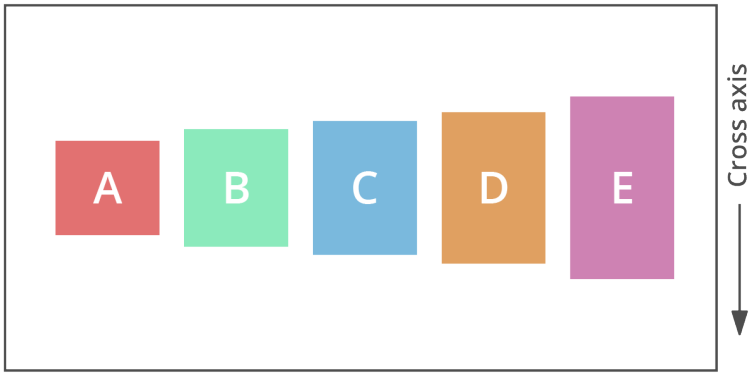
Copyright © 2026 Vizrt

Flex Start



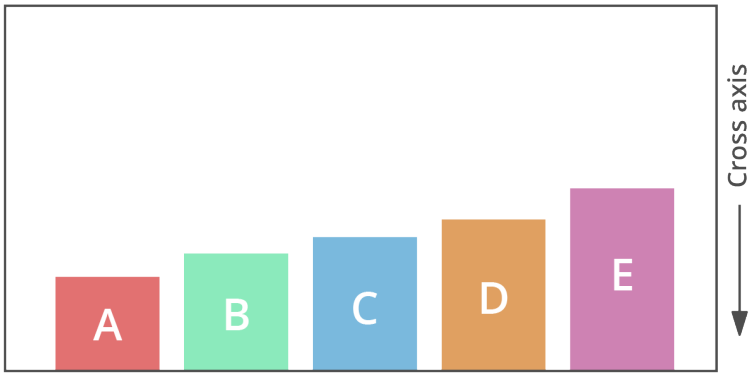
Aligns child containers to the **start** of the cross axis.

Center



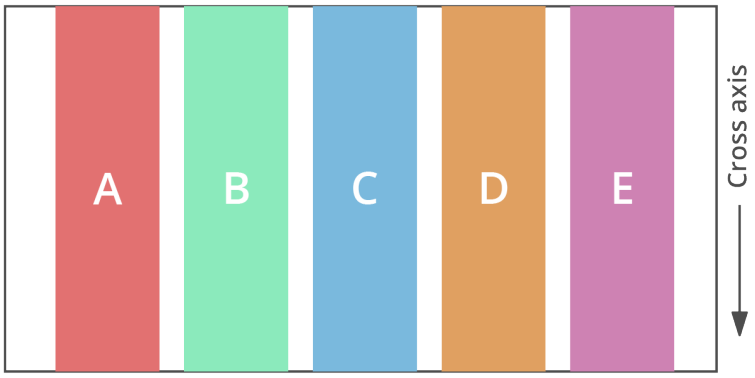
Aligns child containers to the **center** of the cross axis.

Flex End



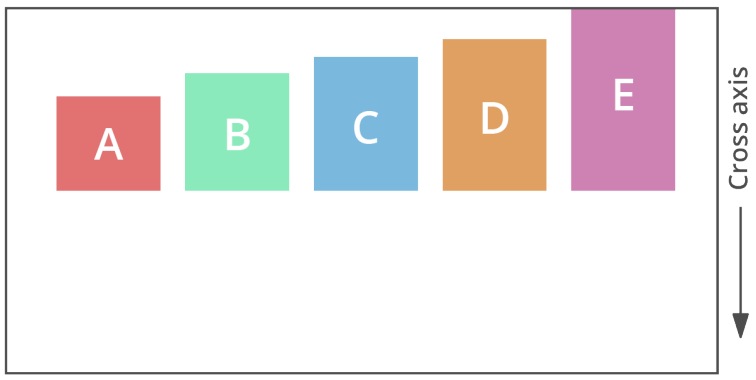
Aligns child containers to the **end** of the cross axis.

Stretch



The items are stretched to fill the cross axis. Therefore the cross axis setting (in our case the height) must be set to be **auto** to allow the parent to overwrite the value

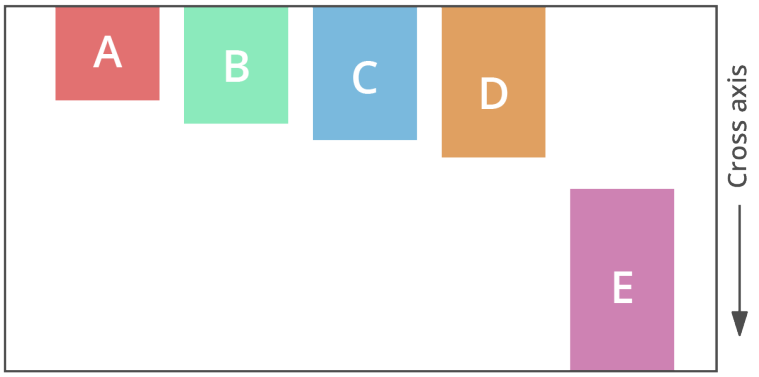
Baseline



Aligns all containers in a way such that their baselines align. The highest one starts on the top.

9.5.13 Align Self

Align self is used to overwrite the Align Items property given by the parent container to is own. In the example below, the parent container is set to be Align Items: Start, but container E has an Align-Self set to be Flex-End. This takes it out of the computed position of the top (parent is set to Flex Start) and overwrite it with its own property Flex-End:



9.5.14 Align Content

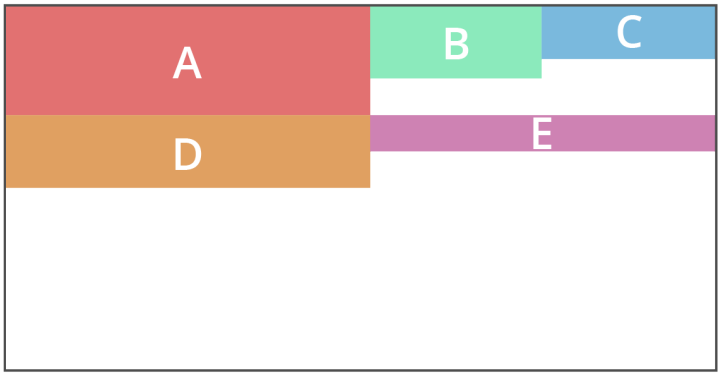
Defines how containers are aligned, if the wrap mode forces elements to be distributed in more than on row or column. It has no effect if no wrapping happens.

The following settings are available:

Auto

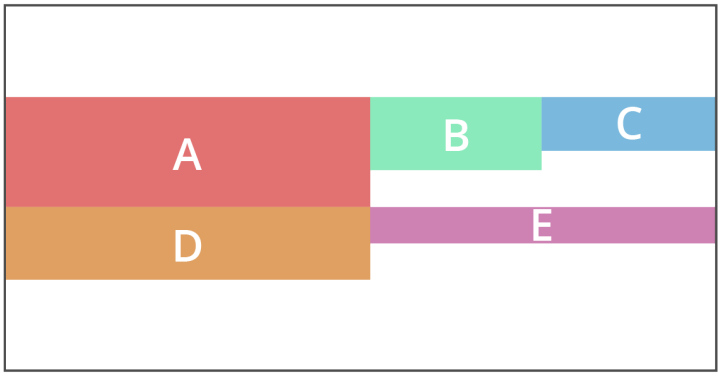
Uses the align item of the parent container.

Flex Start



Aligns wrapping child containers to the **start** of the cross axis.

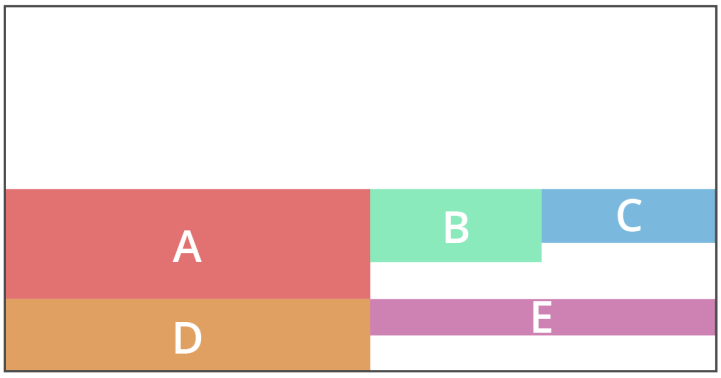
Center



Aligns wrapping child containers to the **center** of the cross axis.

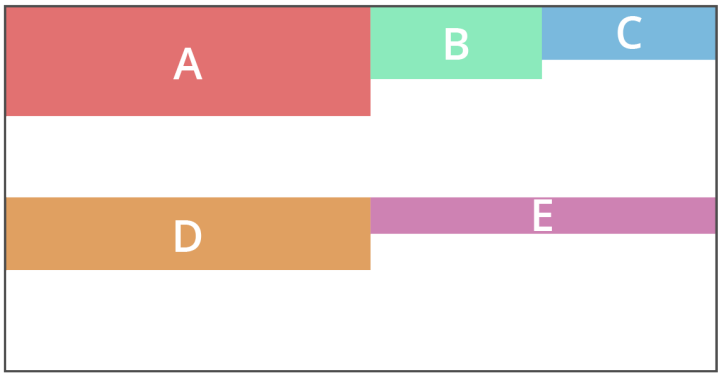
Flex End

Copyright © 2026 Vizrt



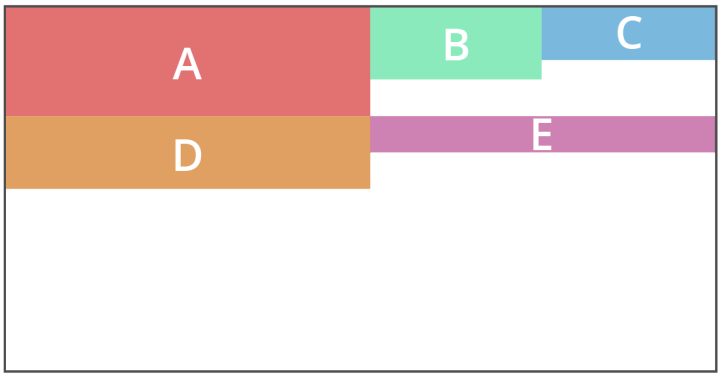
Aligns child containers to the **end** of the cross axis.

Stretch



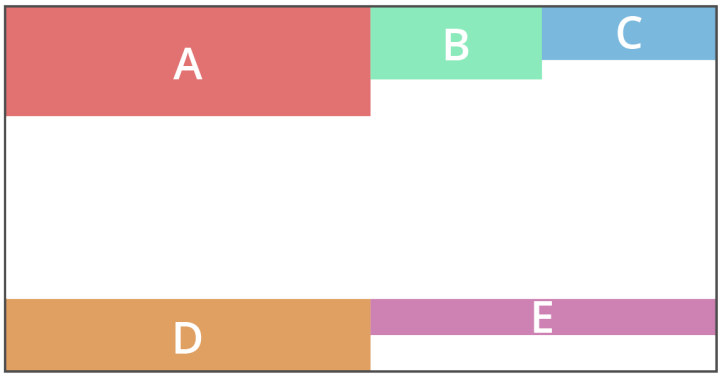
Stretches the space between the elements to reach the parent containers cross axis.

Baseline



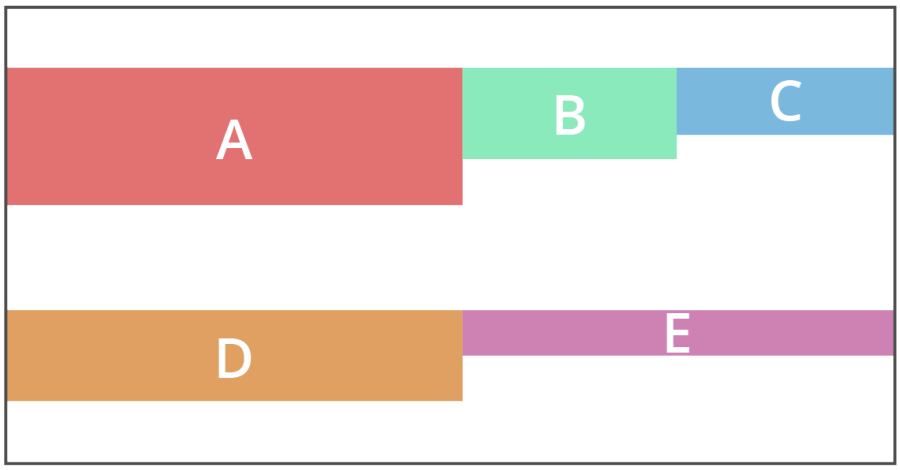
Aligns all containers in a way such that their baselines align.

Space between



Distributes the remaining space between all lines to match the parent containers cross axis.

Space around



Like space between, but also distributes space including top and bottom.

Copyright © 2026 Vizrt

9.6 Designing for Adaptive Storytelling

The following tutorials outlines how to start building a typical information graphic and adapt it for multiple output formats.

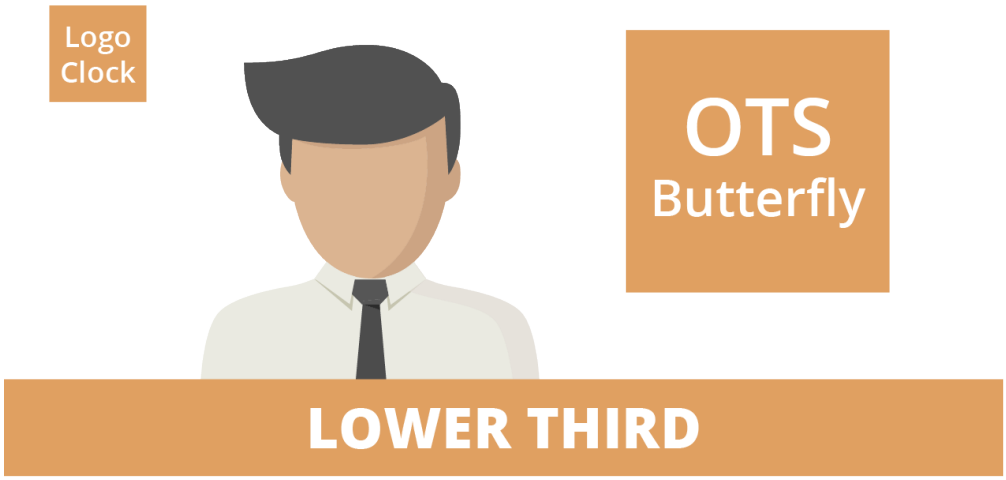
- Planning Phase
    - Typography
- Layout
- Tutorial
    - Step 1: Creating the Fundamental Layout
    - Step 2: Creating the Logo
    - Step 3: Creating the Breaking News OTS
    - Step 4: Creating the Banner Part
        - The Live Badge
        - The Clock
        - The Information Part
        - Adding the Top Header
        - The Final Scene Tree
    - Step 5: Animating the Over The Shoulder
    - Step 6: Adding a Portrait Mode
        - Adjusting the Logo for Portrait mode
        - Hide the Headline
        - Adjusting the Breaking News for Portrait mode
        - Finalizing the Header by Adjusting the Badge Part

9.6.1 Planning Phase

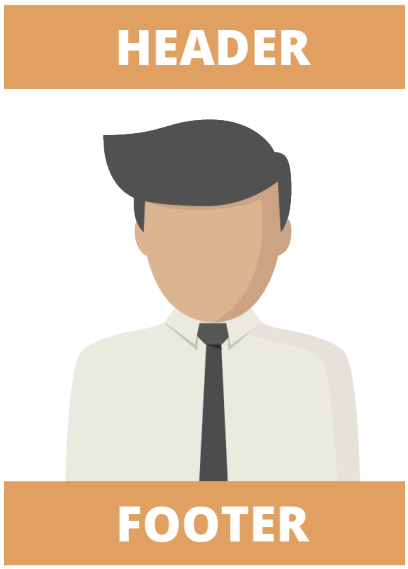
Creating scenes for adaptive storytelling requires some planning phase(s). As every format has a different layout, it needs to be decided which elements are needed in which format. Some format like a portrait mode for smartphones might not necessarily have all elements visible. In addition, font sizes might need to be different between different formats. While a regular broadcast resolution works well with smaller and thin fonts, a mobile format might requires more bold fonts. The viewers distance to the device also plays an important role to be able to recognize all screen elements.

Traditional broadcast layouts offers lower thirds, butterflies and over-the-shoulder elements. The content (for example, an anchor) is placed in a golden shot alignment.

Copyright © 2026 Vizrt



Mobile layouts use a complete different layout. Graphic elements are placed either as headers and/or footers and the main content is located in the middle of your screen.



A table layout might be completely different than a portrait smartphone layout. Notice that Safe or Title areas are not really existent on Mobile Phones, but notches for cameras may need to be respected.

Typography

The viewing distance between traditional broadcast graphics like TV screens is much larger (two meters for a 55" screen) than other devices like smart phones or tablets. A TV screen is usually located a few meters away from the viewer, whereas a smartphone or tablet are very close.

Copyright © 2026 Vizrt

Thin fonts like a *Helvetica Neue* work better on smartphones/tablets. On a TV screen these fonts may not be used as the viewers distances and eventually the interpolation of the TV screen would make such thin lines unreadable.

9.6.2 Layout

Creating scenes for Adaptive Storytelling requires a new way on how to place your containers. Elements are now positioned relative in space and in relation to other objects. The arrangement needs to be split into different boxes. Given the following layout, the layout can be split into the following boxes:





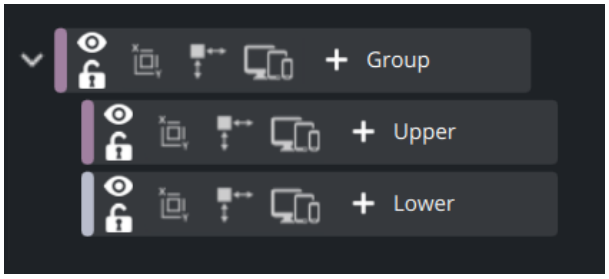
9.6.3 Tutorial

Step 1: Creating the Fundamental Layout

Lets start by building the basic layout for our design. The design starts with a layout of two boxes located underneath the main box:

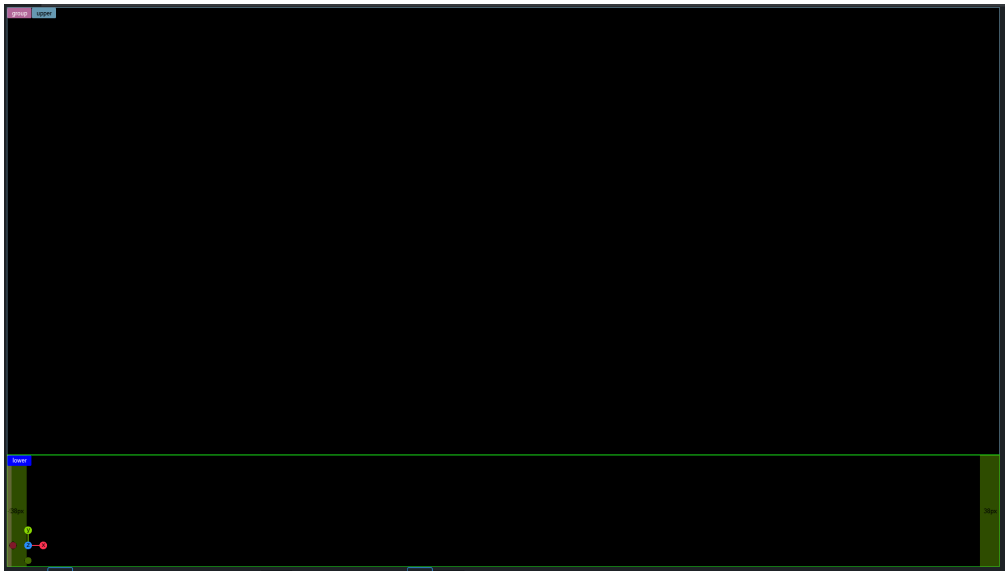


Create the main container and two Subcontainer representing the Upper and the Lower part. In our default format (16:9 HD resolution), the lower takes 20% of the whole height. The top container (*group*) has a width and height of 100% to fill the whole screen and the Flex direction is set to be column. This aligns the upper on top of the lower. The *lower* container is set to **height of 20%**, and **Grow** and **Shrink** set to 0. This ensures that the lower always has a fixed size of 20% of the whole screen. It also has a **padding** on **left** and **right** set to **2%.** The *upper* container gets a height to 100% (or auto) and Shrink set to 1. This allows to shrink to the remaining space of 80% in height.



Your layout should now look like this.

Copyright © 2026 Vizrt



Step 2: Creating the Logo

The logo has the same layout as in the previous step. A basic container holding two subcontainer, layouted as **columns** to make sure the are positioned from top to bottom. The alignment (Align Items) is set to be **Center,** the **position** is done on a **absolute** base, the **width** is set to **10%** of the whole screen, **height** is **20%.** The second container holds two subcontainers, one is the container for *World*, the other one is the red backplate. The *news* container is placed outside the structure, but tracked to the backplate container.

The *World_icon* container has a **width** and **height** of 50%. As the parent box has its items aligned on center. It is placed in the middle of the parent box. **Grow** is set to **0**, this makes sure it does not get bigger, but **Shrink** is set to **1.0** to allow it to be smaller in other formats that will be added later on. The world Icon itself is a rectangle with a unlit Phong material on it with the world as a texture.

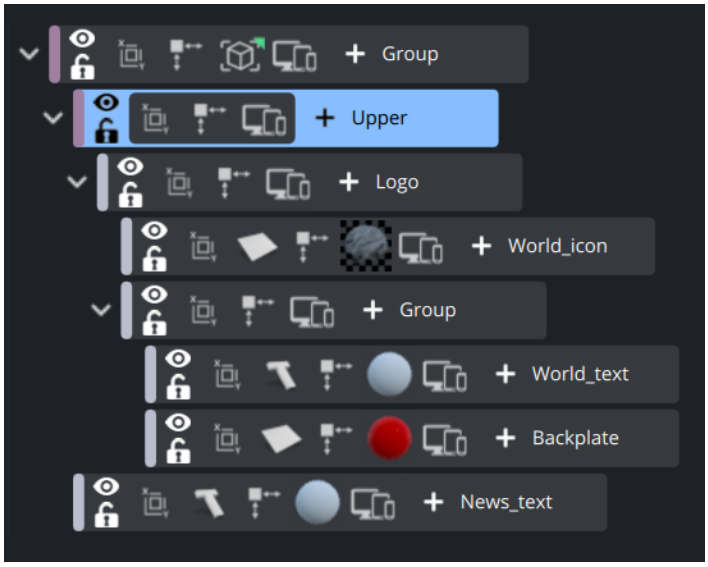
The second container also has a **height** of **50%**, it holds 2 subcontainers *WORLD_text* and *Backplate,* aligned also as columns. *WORLD_text* represents the *WORLD*" term of our logo. It has a **height** of **50%**, (effectively 25% of the whole logo), **no shrink**, **no grow**. There is a Text on it with a variable fontstyle, fontsize is set to 37.0 to fit into the height of the box.

Underneath the *WORLD_text* container is the red *Backplate* container. Settings are exactly the same, but it has a red rectangle assigned to it.

As the *News* part of our logo is located at the same position as the backplate in our design, create a new font container outside and underneath of our logo structure. In theory, it could also be within the structure, but it destroys the current design we have. It is important that it is located underneath, otherwise the backplate is rendered after the text.

Put a Text "NEWS" onto the *News_text* container. In the Flexbox container, we set the **Tracking** to the ***Backplate***. This makes sure the position is exactly on the backplate. Adjust the position by settings the **Offsets** correctly

Copyright © 2026 Vizrt



As the whole *logo* container is set to be on **Absolute Position**, we can position it by modifying the offsets according to our design. Your layout should now look like this.



Step 3: Creating the Breaking News OTS

The Over the Shoulder element contains of an additional box, split into two rows. The first one shows a red backplate and the text *BREAKING*, the second one a white backplate and the text *NEWS*. As these words are not intended to be changed afterwards, the dimension of the container must not be dynamic. This box is also positioned in **absolute** mode.

- Create a new container named *OTS* with the Flexbox plug-in.
- **Width** = 40%, **Height** = 10%.
- Set the **offsets** to be **15%** from the top, and **5%** from the **right side**.

This container holds two subcontainers: The first one with a width of 60%, second one with 40%. Add rectangles and a red and a white Phong material.

Copyright © 2026 Vizrt

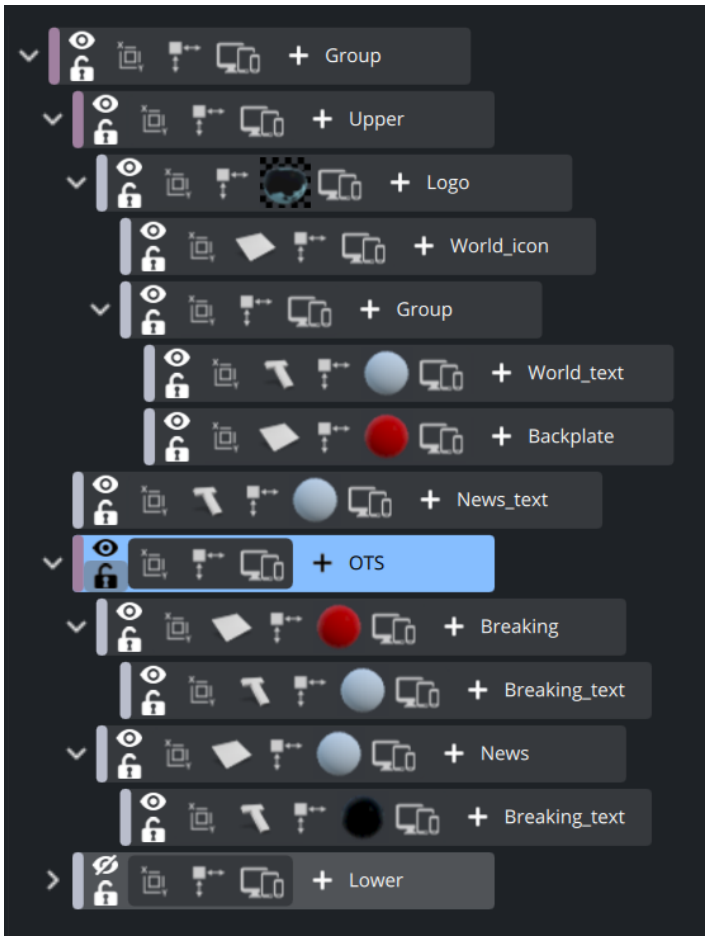
The two text containers *BREAKING* and *NEWS* are subcontainers of these containers as their position is already given. The Text is centered and a white and a black material is needed here. It might be required to add some offset of about 20% to have the text also centered vertically.

> **Tip:** If the texts are not showing up correctly, try to increase the **z-value** in the Box Transformation Editor to make sure it is placed in front of the backplate rectangles.

This leads to a layout like this:



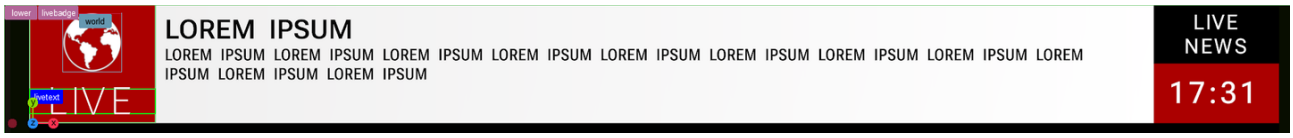
And the Scene Tree:



Copyright © 2026 Vizrt

Step 4: Creating the Banner Part

Lets continue with the more complex lower third banner. This element can be split into three sections. A live badge on the left side, a centered part that shows a headline and some more information text and on the right side, we will have again a tile part and a live clock.



Using our previously added *lower* container, we need to add three subcontainers and the lower container needs to have a Flex Direction of **Row**.

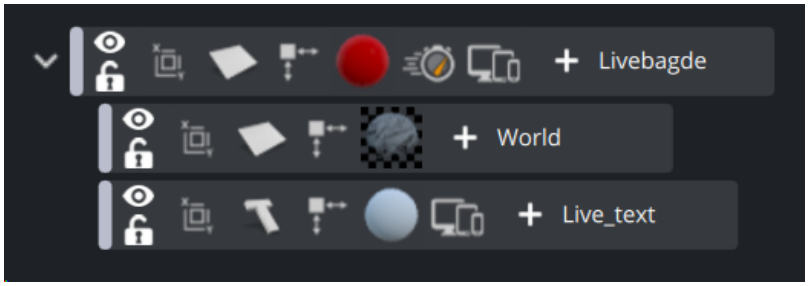
- Container 1 (the world and live) occupy 10% of the whole width and 100% of the height. Set it to **not shrink** and **not grow**. This ensures this part is always 10%. It is named *livebadge.*
- Container 2 represents the middle area. It takes 80% of the width and 100% of the height. Set the grow and shrink to `1`, so the width of this part and be flexible. It is named *mainbanner.*
- Container 3 shows the name of the show and a clock. It is built in the same way as container 1. Width of 10%, no shrink, no grow. It is named *clock*.

The Live Badge

Let's have a look at *livebadge*. Its structure is almost the same as in our previous section Designing for Adaptive Storytelling#Creating the Logo.

The container first gets a rectangle as backplate with our red material assigned to it. The Align Items setting should be **Center**. Underneath, two more containers are placed and aligned in Flex Direction **Column.** The bottom container holding the *LIVE* text has a fixed height, which is set at 20%. The *World* container is flexible in height, set it to **Grow** and **Shrink** to 0 and adjust the dimension by setting a **height** (for example, 50%).

This results in a tree structure as this:



The Clock

Lets continue on the right part, the title and a clock. Create a new container named *clock* as next container within the *lower* container. Its **width** is set to 10%, **no shrink**, **no grow**. The structure is like in the previous step. The

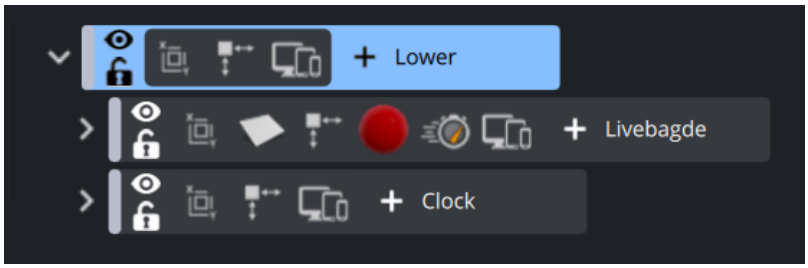
Copyright © 2026 Vizrt

container is divided into **two rows**, both of them a **height of 50%**. The first container called *title* holds text *LIVE NEWS* as two center aligned lines. The height of the container is fixed to 50%.

The next and final container of this part of the graphic has also a fixed **height** of 50% and **width** of 100%, it holds a red rectangle as backplate. Create a subcontainer for the final clock. Give it a **fixed width** of 100%, place Text inside, and align it to the center. The clock is done by adding a JaCK plug-in (any similar clock plug-in might work too). Adjust the font size to match your design. It might be required to center your clock vertically by giving an offset to it.

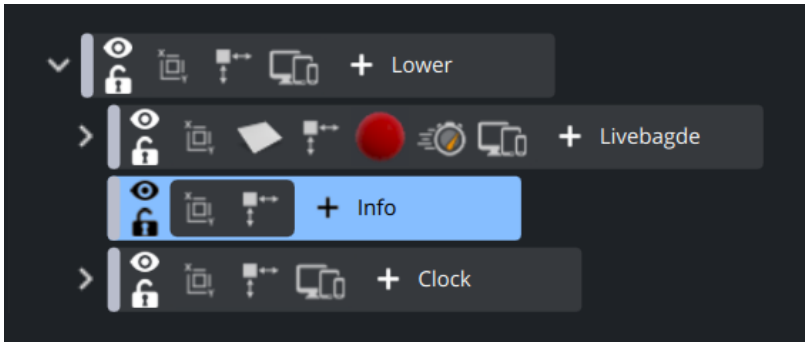
Don't be worried if your clock is stuck to the first container:





The Information Part

The information container goes in between the *live* and the *clock* container:



Copyright © 2026 Vizrt

If the width of the info container is changed in the flexbox plug-in, the clock container automatically aligns to the right side.



Set **Grow** to 1 and the container automatically scales up to the available width. Add a white rectangle as backplate. **Width** should be set to auto, the **Height** to about 80%. The content is the info container is (as previously) split into to headline container and an additional text container, therefore the **Alignment** of children must be **Column**.

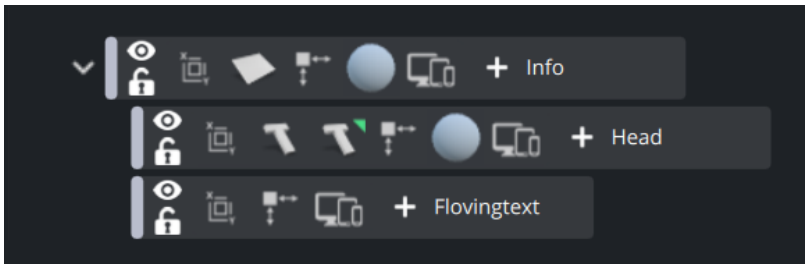
Create two subcontainers, name them *head* and *flowingtext*. They should already be stacked top-bottom inside the info container. Both containers should be set to grow/shrink on `1`. The head container gets a text on it, with a black Phong material. Enter some sample text and adjust the size correctly. Turn off the **Geometry-aware scaling** in the Box Transformation properties and then set the Bounding Box mode of the Text to **Typographic**.

In the flexbox properties of the *head* container, set **Grow** and **Shrink** to `0`, **Width** and **Height** to `Auto.` In addition, set all margins (expect left) to `Auto`. The left margin now controls the distance/offset of our headline text to the left *Live Badge*. You'll notice that the textbox now grows and shrinks correctly according to the amount of text being entered. If we would have kept the geometry aware scaling, the text can be sized down if it gets too large.

Put a **ControlText** plug-in on it, so it can be controlled from the outside.

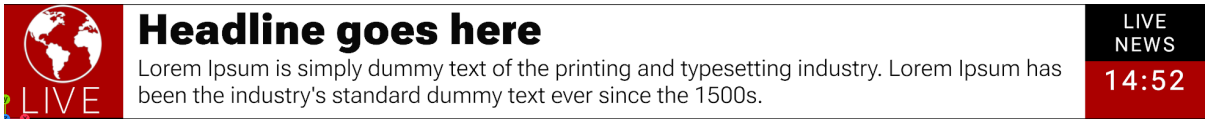
> **Information:** if you add a **ControlText** plug-in to Text, the format-dependent properties will be disabled.





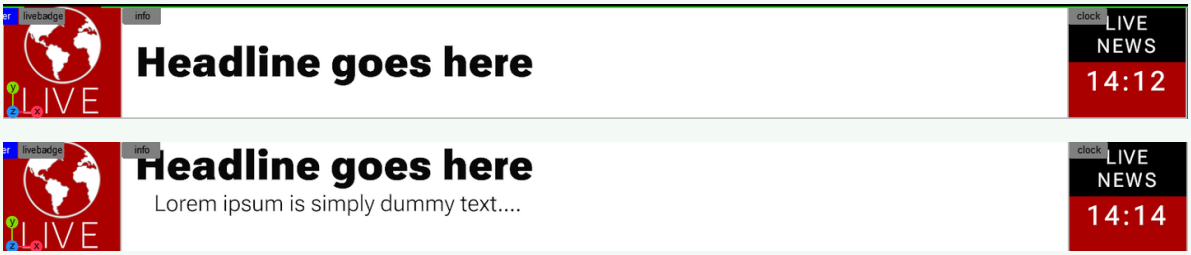
The flowing text part is built in the same way, except that we change the behavior of the box. Here, our box stays at a fixed width, to make sure the text breaks into the next line if it gets longer:

Copyright © 2026 Vizrt



Make sure the wrapping is correctly set in the Text properties, otherwise the text does not wrap into the second line.

Expert Tip: It would be nice if the Head is automatically centered if there is no flowing text set. If the *info* container is set to align in center, our headline is placed in the center. However this does not work as long as our *flowingtext* container is still being calculated by the flexbox container on the *info* container. We have to disable the flexbox plug-in if there is no text set. As this functionality is not yet implemented we need to do a little trick using a script.
What this script does, is that it registers for a callback whenever the text is being changed. the flexbox function is turned off, if the text is empty, otherwise it is turned on. So now we are able to show a centered Headline, if there is no text or we show the regular headline plus flowing text:



Adding the Top Header

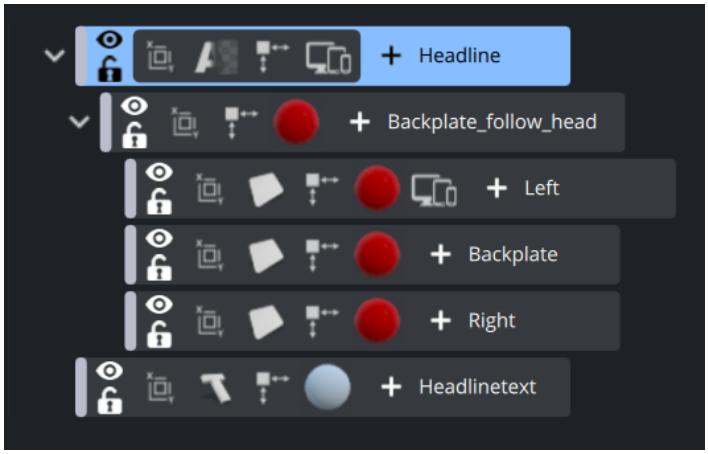
The last and final part of our graphics is a bit more advanced, it combines the functionality of a TextBG (align the Background to the Text) plug-in as well as a Autofollow functionality. The graphics should automatically align to a certain text length and be decorated let and right.

Examples:



Our structure consists of two parts, the backplates with our red Noggi geometries (split into three subcontainers) and the reference text.

Copyright © 2026 Vizrt



Create *Headline* container underneath *lower*. The **position** is set to **absolute**, so it does not influence our layout and we can position it wherever needed.

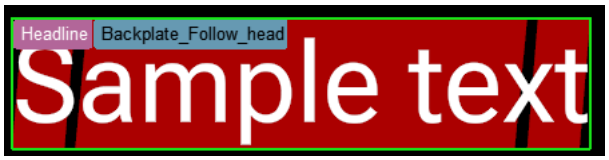
Create a "head" container and name it *Backplate_Follow_head*. Create a Textcontainer named *HeadlineText*.

The Textcontainer gets a Text on it, the **Geometry aware Scaling** in the Box Transformation needs to be turned **off**, as we want the size of the box to grow and shrink according to the width of our text. The **Bounding Box** needs to be set to **Typographic**. The Flexbox container needs to be set to **Grow** and **Shrink** enabled (=**1**), **Width** and **Height** should be **auto**.

It is important to make changes to the *Headline* container too. The **width** and **height** need to be set to **auto**, otherwise a fixed value (like a certain amount of pixels or percentage) forces our text container to scale up or down again.

This should already allow the box to react to the size of the text. The next step is to bind the *Backplate_Follow_head* container to the size of our text. Drag the *HeadlineText* container as **Tracking Container target** in the Flexbox plug-in of container *Backplate_Follow_head.* Create the three subcontainers *left*, *backplate* and *right*. Add a Noggi and add our red material to it. As *left* and *right* stay at their sizes, they need to have **Grow** and **Shrink** turned off (`=0`).

The **width** is a **fixed value** in pixels like 45px. The middle container *backplate* shrinks and grows (`=1`). Now it looks like this:



Add `-50px` Offset on left and right side to the *Backplate_Follow_head.* The *Headline* container is in absolute position mode, place it on top of our lower third banner. It might be required to increase the Z-Offset a bit.

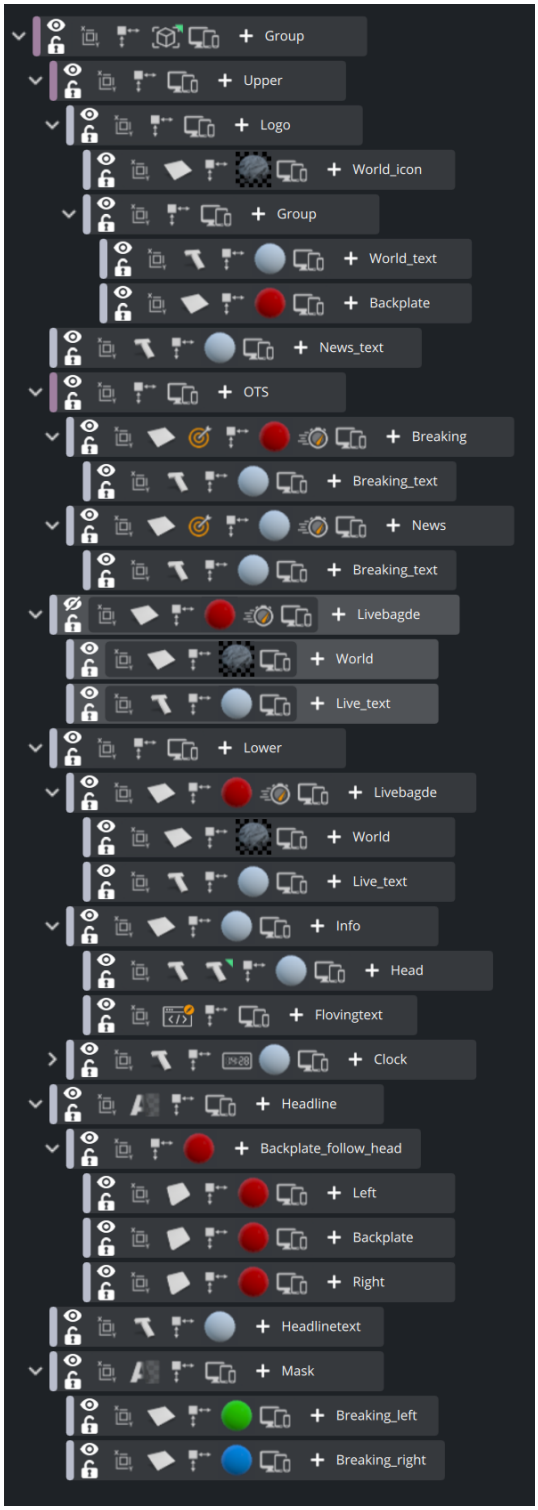
This completes our design:

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

The Final Scene Tree



Copyright © 2026 Vizrt

Step 5: Animating the Over The Shoulder

We'll create a simple animation as an example in our tutorial. Animation is limited compared to the traditional three dimensional transformation mode as we are working in two dimensional space only. For example, there is no rotation around the x axis. In our tutorial, we animate the Breaking News OTS by using two masks and let the two boxes fly in from the center:

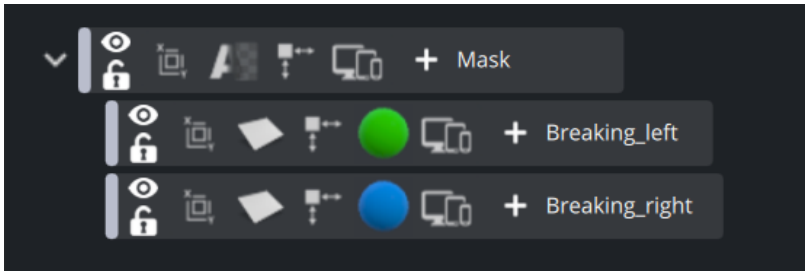
In our example, we create a set of container to act as masks. This container uses our OTS as tracking container, the masks do not require and re-positioning in other formats. Create a new *Mask* container, with two sub containers. One masks out the *Breaking* part, the second one masks the *News* part.

As the mask is on the same position of the *OTS* container, we can adjust our two masks by in/decreasing the **Width** and the **Offset** on the **Left** or **Right** side. Add a **Mask Source 1** on the left (=green) and a **Mask Source 2** and the right (=blue) container.

To illustrate it, we've put a green and blue color on the containers and added some alpha on it:



Tree:



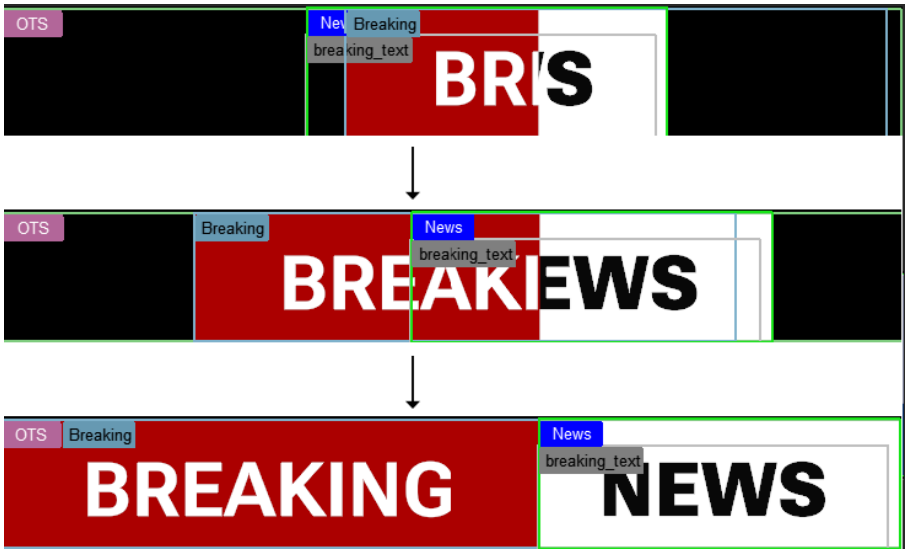
Now we need to put the Mask Target plug-ins on the *OTS* sub containers, but in opposite order. Our *Breaking* container needs to be masked by **Mask 2**, the *News* container is affected by **Mask 1**.

To build our unveil animation, we put the *Breaking* into the Mask 2 area and the *News* container into the Mask 1 area. Do this by by modifying the **Offset** values of both containers, in this case it would be better to use absolute pixel values instead of a percentage value.

- *Breaking*: Offset left = 460px -> 0px
- *News*: Offset right = 310px -> 0px

The values above are sample values only. Now we can animate those values to go from the start offset back to 0 px. Our unveil animation should finally look like this:

Copyright © 2026 Vizrt



Animating the other parts of our sample works accordingly. This is not part of this tutorial.

> **Information:** There is no need to target link the masks to the source containers, they can also be positioned in absolute mode or even without flexbox control. There are no limitations how to place them.

Step 6: Adding a Portrait Mode

As our design is now done and animated for a typical 16:9 layout, we are now going to add a portrait format (for example, to create a layout for smartphones). We are going to adapt our 16:9 broadcast layout into a portrait format suitable for mobile phones:



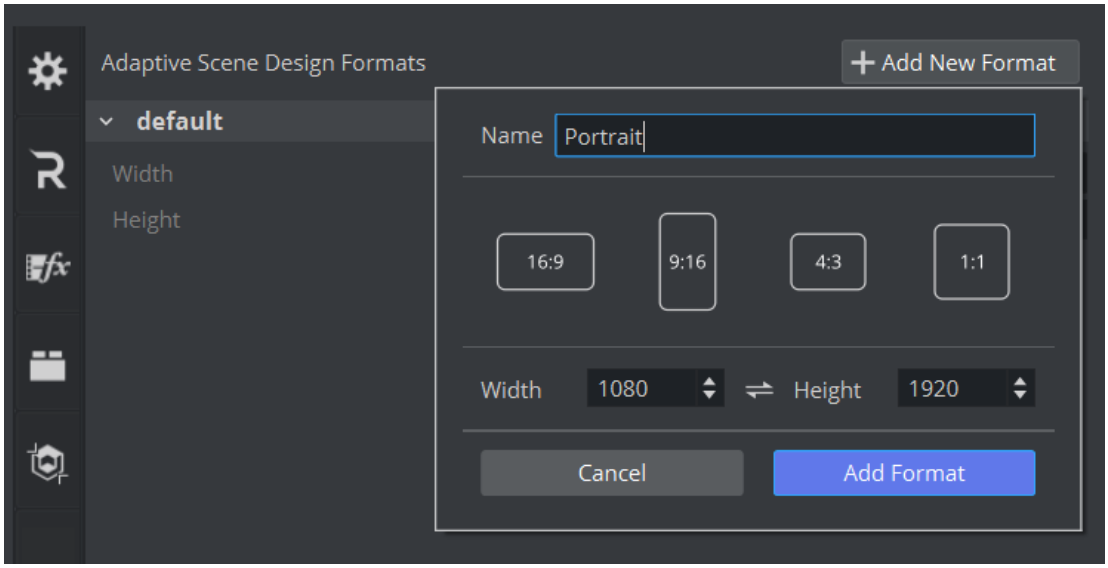
Most of the parts from our original design are also present in the portrait mode:

Copyright © 2026 Vizrt

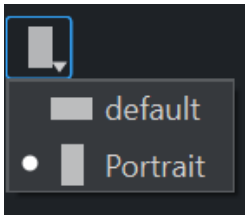
The **Logo (1)** is more or less the same, the typography is a bit smaller, the logo appears bigger. The **Breaking News (2)** becomes part of our Footer graphics block, the *news* part gets a red background instead of white. The Live **Badge (3)** becomes part of the Footer graphics block too. The **information text (4)** gets a smaller headline font, the infotext has more lines to fit in its information. Finally the **Clock and Title (5)** goes into the last part of the footer graphics.

The **additional headline (6)** only appears in the broadcast version, not in the mobile version.

We start by adding a new format in the scene settings. Go to Scene Settings **Adaptive Scene Design Formats** and create a new **Portrait** format in **9:16 aspect ratio**:



This adds a new entry in the format switcher.

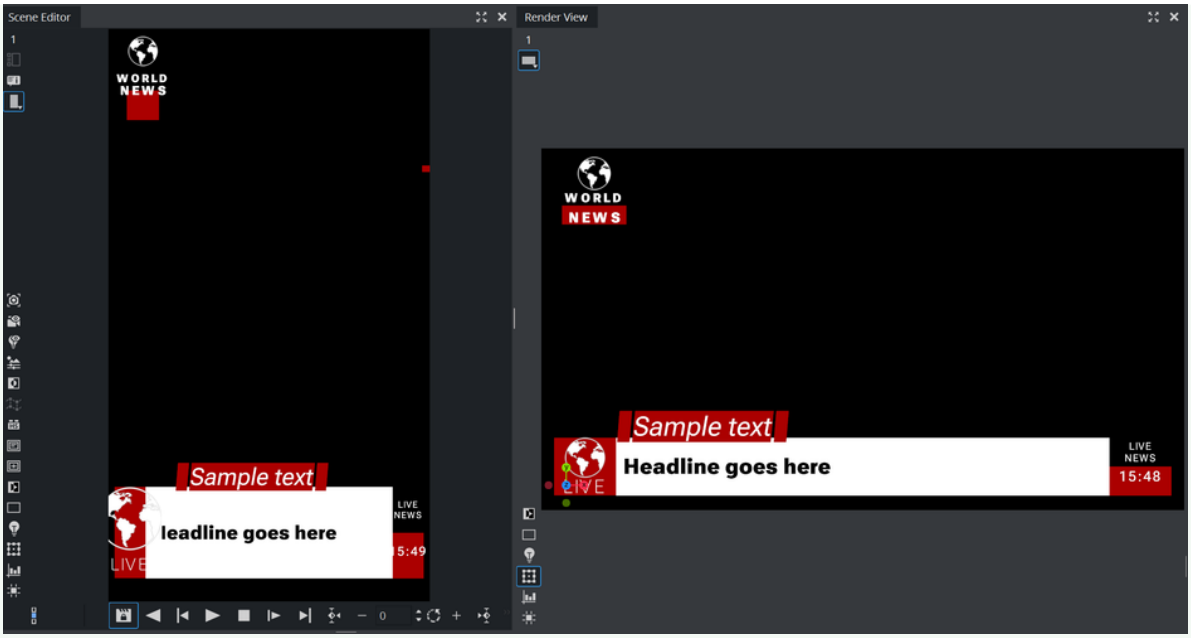


Switch your format to Portrait. This is important as all our changes are then applied to the Portrait, the default is kept untouched.

Tip: In the case of having a **Text** plug-in with format-dependent properties, you should select the entire text in the text area before switching formats.

Add a Render Preview:

Copyright © 2026 Vizrt

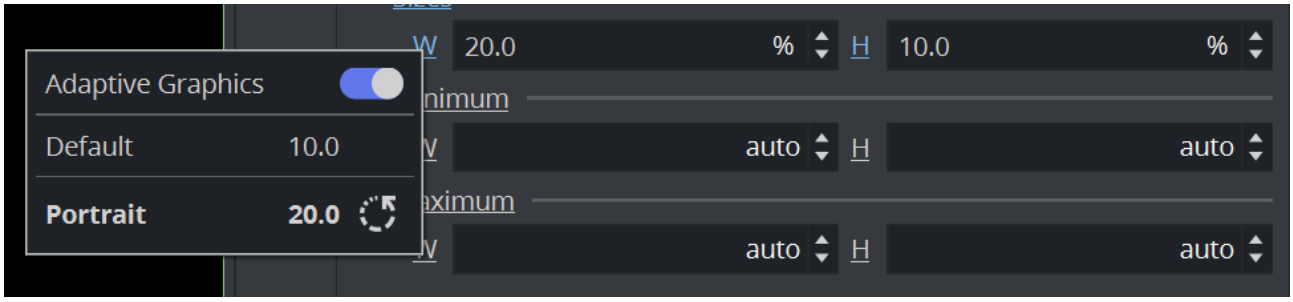


It is a good idea to add a new Render view from within the Workspace Menu. This is set to our Default format. This view is used to monitor our changes and make sure that the original design stays as it should.

Our new Portrait format looks already a bit how it should, however there are a few things that we need to adjust:

Adjusting the Logo for Portrait mode

The logo needs to be adjusted in width and height. Click on the logo container and change the **Width** from 10 to 20%, the **Height** from 20% to 10% on the Flexbox instance. Less height, more width. Please notice, that while you change the values in the portrait mode, the default mode should stay untouched. The **Width** and **Height** values now are underlined. Right click on the **Width** or **Height** entry of your Flexbox plug-in:



The information window illustrates, that the default width value is 10, but in portrait it is set to be 20.

Change the **width** of the *World_icon* to get bigger (100%). The *NEWS_text* container can be adjusted by simply modifying the Left and Top Offsets. You can use pixel values or %.

This should make our Logo already look correct in portrait and in default format.

Copyright © 2026 Vizrt

Default	Portrait
	

Hide the Headline

The additional headline (6) is invisible in portrait mode. As it is anyways positioned in absolute mode, it does not influence our layout if it is not present anymore. Click on the **Hide** ( ) icon.

> **Note:** The Default view it is still visible, but hidden in Portrait mode.

Adjusting the Breaking News for Portrait mode

The *Breaking News* OTS is transformed the most between Default and Portrait. It changes its position, background color and color of our text. Lets start by adjusting the position. First, make the masks invisible. This makes it easier to adjust the position.

The *OTS* container is in **absolute position** mode, so adjust the position offset. Set the **top** back to **auto**, but the bottom gets a value of 20%. This moves the original Over the Shoulder container as a headline into our footer. The width can be increased to be 80%, the border on the right to 0%. This aligns it to the right. The height can be set to 5%. Adjust the container to match your design by either giving more offsets or shifting using padding on the box or the text underneath.

Next, give the white background of the *News* part a red color, the text (*News*) itself should have a white material.

> **Note:** Do not copy the materials, this also changes the original design. Click on the material itself and change the Color part only.

If you want to animate this part also in portrait mode, you have to make sure, your masks are on the correct place too. As animations are also format dependent, you need to recreate the unveil animation for portrait again. This is needed as the position values have changed in this format.

Copyright © 2026 Vizrt

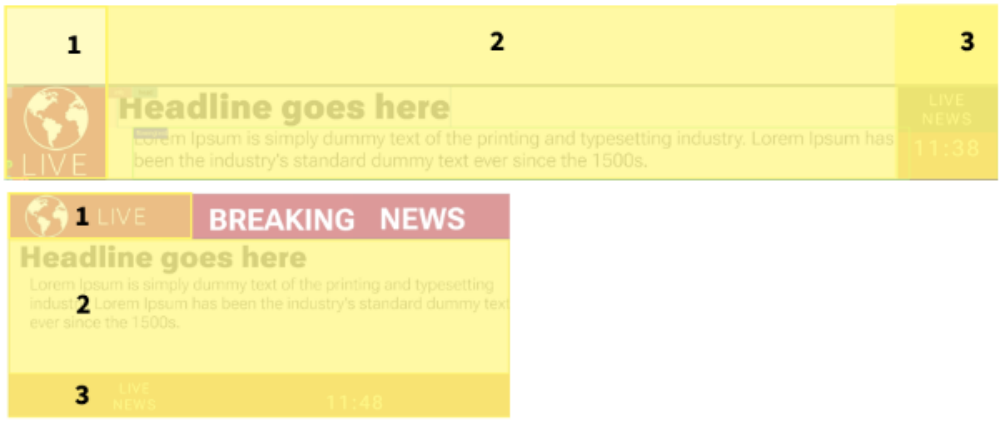


This should now make our original OTS graphic like this:

Default	Portrait
BREAKING NEWS	BREAKING NEWS

Finalizing the Header by Adjusting the Badge Part

We need to complete our Header in portrait mode by moving the *Live* badge to the top. In the Original design, the lower container is arranged as as rows, in portrait it would be better to change this to be columns:



Part 1 becomes part of the Header and changes its layout to a **row**, Part 2 mainly keeps its layout and **Part 3** goes from top-bottom to left-right layout in a **row** and finalizes our footer part.

Change the **Flex Direction** of the *lower* container to **Column**. This arranges the *livebadge* on top, the *info* next and finally the *clock* container top to bottom. Remove the paddings of the lower, they are not needed in Portrait mode. The next step is to adjust the live badge: The **Flex Direction** should also be changed to **Row** to arrange the world next to the *LIVE* text. Adjust the **width** of *livebadge* to fill the remaining space in our header. The Height should match the Height of our previous *Breaking News* part.

The world icon needs to be scaled to match into our header. Make sure that Grow/Shrink is turned off, otherwise it scales down. Maybe some **Offset** on the left side is needed.

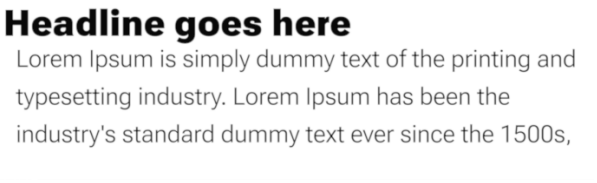
The information text is the next step. By default, we have a lot of width to fill the text, in portrait this is limited. We would need to **decrease** the **font size** and use more than two lines of text. Change the width of the *info* to be 100%,

Copyright © 2026 Vizrt

so we can use the area. The height can also be extended, leave some space for the final footer part. The layout of our two text container should already match, our info text might need some more line height as we are using more lines now. In the Text editor, adjust the **line height**, for example **+ 0.4em**.

Let's finish the Portrait design by switching the layout of our *clock* container also to be **row**. Increase the **Width** to be 100% and adjust the **bottom offset**. The red backplate of the clock needs to fill up the footer space. As the *title_landscape* ("="Live") is influencing our width, change the position mode from to **relative** to **absolute**.

The text in portrait will not be two lines, change it to be only one line (note that the text has not changed in our default layout). Finally, align the clock container by adjusting the offsets on **Left** and **Top**.

Default	Portrait
	
	
	

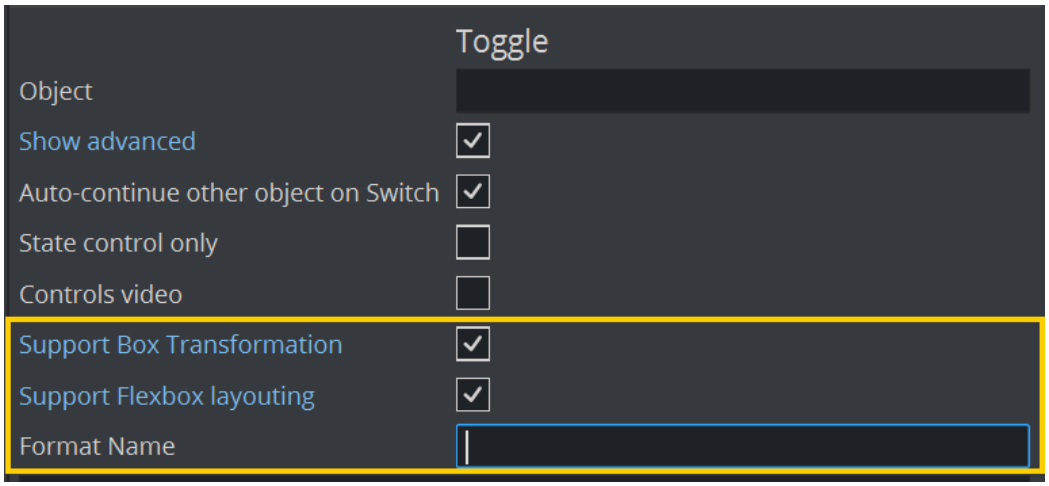
Now we have finished our tutorial. We've created a flexible layout with animations. It can be played out in classic 16:9 aspect ratio and also looks great in portrait mode. This tutorial highlights some of the use cases on how to create scenes for adaptive playout. There might be smart ways to achieve different looks, but it least it gives an introduction into adaptive scene design.

Copyright © 2026 Vizrt

9.7 Transition Logic and Adaptive Scenes

Support for Adaptive Scenes (Box transformation) was added into the Toggle plug-in from Viz Engine 5.4. It requires the use of *Subscenes*, merged geometries are not supported to be used with box transformation scenes.

The following options have been added to the Toggle plug-in:



- **Support Box Transformation:** Enables the use of adaptive scenes within the Toggle plug-in. *Enable* if you have Box Transformation used in leaf scenes.
- **Support Flexbox layouting:** Forces the Toggle plug-in to handle the layouting capabilities of Flexbox plug-in to match the required toggle functionality. *Enable* if you have the Flexbox plug-in used in leaf scenes.
- **Format Name:** Forces the Toggle plug-in to render the used subscene in a certain format. If none is provided, the format of the *master scene* is used.

Note: After setting these values, the scene needs to be saved and reloaded to add proper Toggle structure.

Copyright © 2026 Vizrt

9.8 Parallel Outputs and Adaptive Scenes

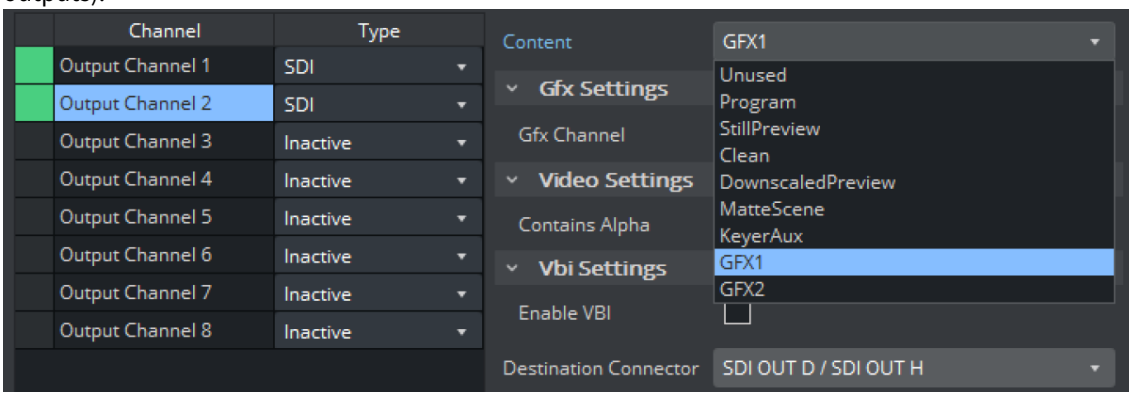
Utilizing Viz Engine's Parallel output feature and Adaptive Scenes, one single Viz Engine is able to deliver multiple NDI streams or SDI/IP signals in different formats.

9.8.1 To Setup Parallel Outputs

Prerequisites

Please configure your engine to use the following licenses:

- Enable Parallel Outputs license.
- For NDI workflows:
  - Enable either NDI input/out single or NDI Out Channel license.
  - If no additional SDI/IP output is required, the Software I/O mode should be set to NDI.
- For SDI/IP workflows:
  - Map one or more SDI/IP outputs to a connector.
  - Choose your type of output (Program, StillPreview, Matte, etc, or one of two available graphics outputs):



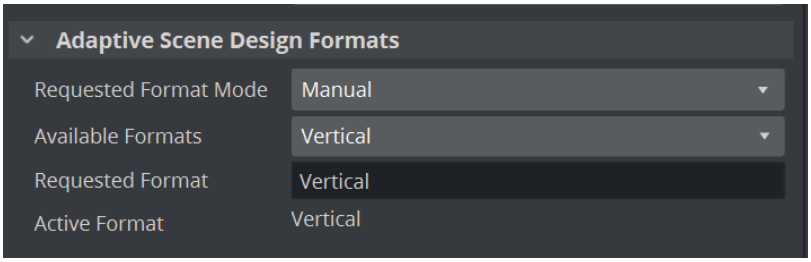
- Configure GFX channels to default (`port 55000`).

To Create a GFX Channel Master Scene

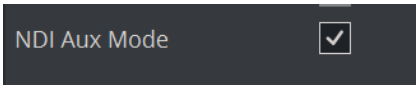
Viz Engine needs to have one main scene serving multiple GFX channels for each format. These GFX channels can be controlled individual by (for example, a Viz Trio playlist through the assigned ports `50000 – 5000(n)`).

- Create scene with two GFX channels as texture
- On the first GFX channel, choose the first format to output. There are several options available:
  - **Manual:** Renders a given format.
  - **Inherit from Main Scene:** Uses the format of the GFX Channel Master scene.
  - **Use Aspect Ratio:** Selects the format automatically based on the width/height and therefore the aspect ratio of the scene). The first matching format is applied. Only works for DVE.
- (for example, Vertical)

Copyright © 2026 Vizrt



- Enable the NDI Aux Mode in the Aux tab.



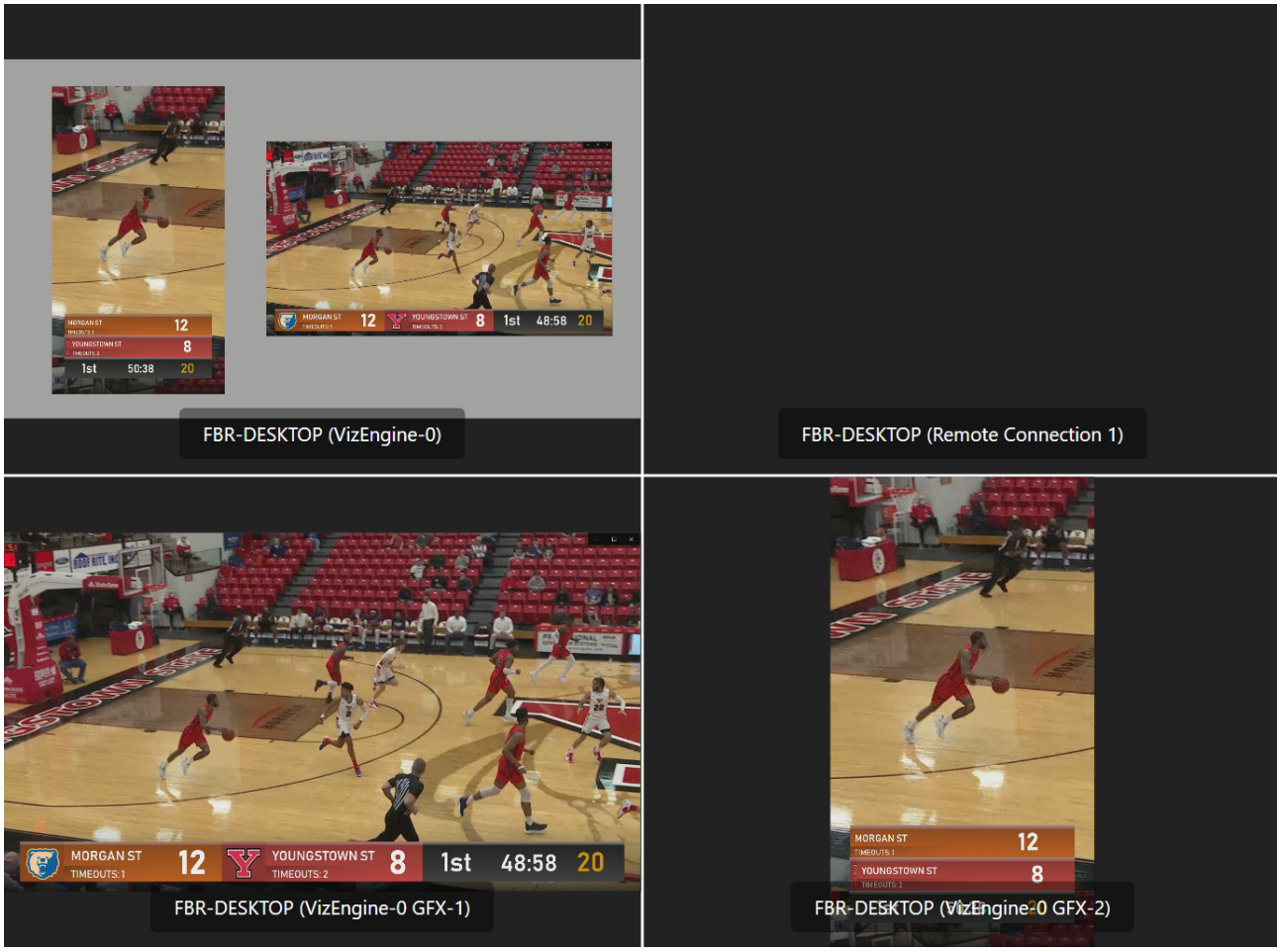
  Do the same for GFX Channel 2, choose for example, horizontal as format.

Viz Engine now serves three different NDI streams:

- The master GFX scene, named *VizEngine-0*.
- The content of GFX channel 1 in vertical format, named *VizEngine-0 GFX-1*.
- The content of GFX channel 2 in horizontal format, named *VizEngine-0 GFX-2*.

In Viz Trio, create a profile to talk to two Viz Engines, one on port `55000`, and the second one on Port `55001`.
Any commands being send to this profile are rendered in the related GFX channel, one in horizontal and the other in vertical format.

Copyright © 2026 Vizrt



Information: Adaptive Scene Design Formats section is also available for each Subchannel in Superchannels settings.

Copyright © 2026 Vizrt

9.9 Adaptive Scene Design Tips and Tricks

The following lists some typical use cases and most common mistakes:

- Building TextBG Functionality
  - Example of Building TextBG Functionality with Three Containers
- Building AutoFollow Functionality
  - Example of Building AutoFollow Functionality with Three Containers
  - Building TextBG and AutoFollow Capability with Six Containers

9.9.1 Building TextBG Functionality

The TextBG plug-in in the Classic Render Pipeline allows any background graphics (like a Noggi or Rectangle) to grow or shrink according to a variable text string. This is a often required functionality that can also be built using Flexbox and Box Transformation:



To have the Box scale accordingly to the text dimension, the following need to be set:

- Disable the **Geometry aware Scaling** in the Box Transformation.
- **Width** and **Height** in Flexbox should be set to **auto**.
- The Bounding Box property of the Text needs to be set to **Typographic**.

This should make sure that box dimensions are already following the textbox dimensions. To have a background, add a Rectangle with Flexbox plug-in. As **Tracking Container** use the container holding the text geometry.

- Set the margins of your Rectangle to be `0` (or according to design).

Example of Building TextBG Functionality with Three Containers

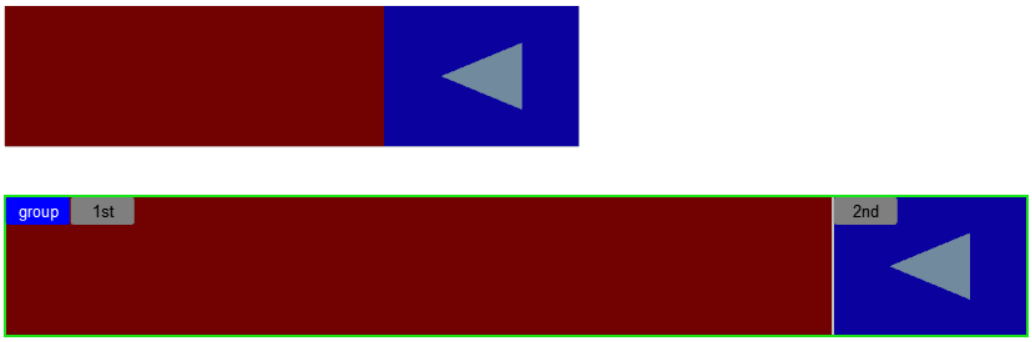
1. Add three containers with one as a parent and two as children.
2. Add the FlexBox plug-in on all three.
3. Set Flex Grow value on the parent FlexBox to `0`.
4. In FlexBox for all three containers, set the Sizes W/H from the Dimensions section to *auto*.
5. For child containers, disable the Geometry aware scaling.
6. Add the Text plug-in into one of the child containers with some text, and add Rectangle/Noggi to the other one.
7. Add material to child containers for visibility.
8. Check the BoundingBox property of the Text plug-in and make sure it's set to Typographic.
9. Select the FlexBox plug-in of the child container with geometry Rectangle/Noggi attached to it, then drag/drop the container with Text plug-in as Tracking container to it.
10. Change the text and it should automatically resize the Geometry based on Text size.

Copyright © 2026 Vizrt

11. Use the z-index of the text container's BoxTransformation to move text a bit ahead of the geometry (Z-index=1).

9.9.2 Building AutoFollow Functionality

AutoFollow allowed a container to follow another containers position based on its bounding box. This is also possible in the Box transformation world, but needs some different way of organizing.



Create a container and split it into the two required containers, one for the master and the second one for the container following the master. Depending on how the second container should stick on the the master, the FlexDirection should be either Column or Row, normal or reversed. The second container has a fixed width/height, therefore **Grow** and **Shrink** should be set to `0`. Set the width and height according to the dimension you want to achieve. The second container now follows the first container based on the layout chosen as Flex Direction.

Example of Building AutoFollow Functionality with Three Containers

1. Add three containers with one as a parent and two as children.
2. Add the FlexBox plug-in on all three.
3. Add Rectangle/Noggi geometry and material to child containers for visibility.
4. Name child containers properly, one as lead, and another as a follower (for the purpose of having the same name here).
5. For the follower container, set the Grow and Shrink values to `0` in FlexBox.
6. Now the following can be done in a row or column directions.
7. For auto-follow in row direction:
    a. First, we need to set the parent container's FlexBox plug-in Flex Direction to row.
    b. Then follower container we need to have a specific value for Width (Sizes/W) in pixels (fixed size) or percent (will always be a percentage of the screen).
    c. For the lead container, we also need to set the Width to 0%.
    d. Now if we resize the parent containers FlexBox Width, we will see that the follower follows the lead container in the row direction.
8. To do the same in the column direction.
9. First, we need to set the parent container's FlexBox plug-in Flex Direction to Column.
10. Then follower container needs to have a specific value for Height (Sizes/H) in pixels (fixed size) or percent(will always be a percentage of the screen).
11. For the lead container, we also need to set the Height to 0%.

Copyright © 2026 Vizrt

12. Now if we resize the parent containers FlexBox Height, we see that the follower follows the lead container in the Column direction.

Building TextBG and AutoFollow Capability with Six Containers

1. Build the TextBG Functionality.
2. Build AutoFollow Functionality.
3. Add the parent container in auto-follow as a child to the container with text in TextBG Functionality.
4. Set width/height for parent auto-follow part to be 100%.
5. Then, set the Offset for the second container in auto-follow to be the same as W/H (for row direction set Offset/L to be the same as Width, for column direction set the Offset/R to be the same as Height).
6. Remove geometry and material from first container of auto-follow.



Copyright © 2026 Vizrt

9.10 Adaptive Scene Design Troubleshooting

Note: The Flexbox plug-in requires a license. Make sure the license is available and the Flexbox plug-in loads as expected.

9.10.1 Wrong Aspect Ratio

If your design appears in a wrong aspect ratio, make sure you have set the Camera Aspect Ratio in the scene settings to **Format**.

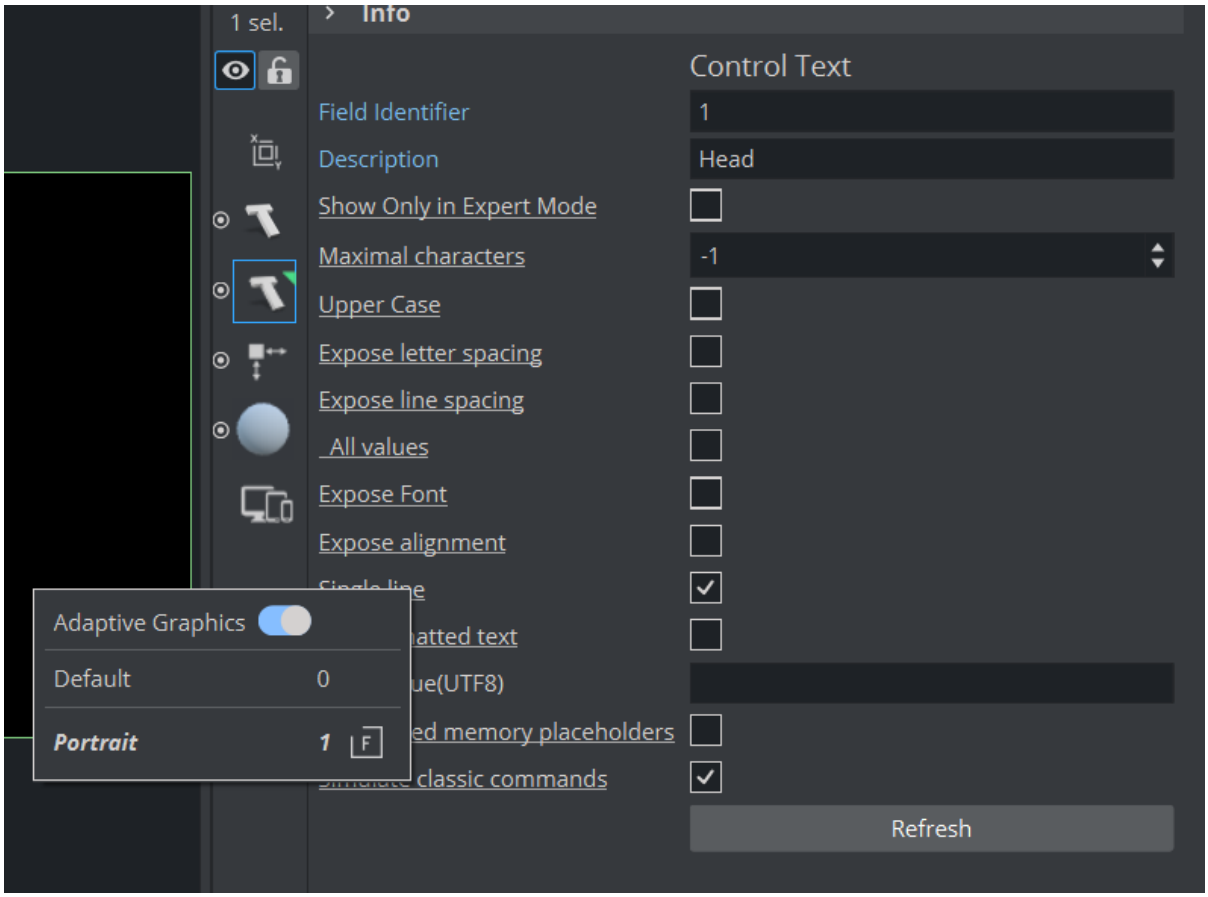
Viz Engine 5.4.1 added support for custom aspect ratios inside the Scene Editor; therefore, the aspect might be wrong if the scene is not set to **Format**.

9.10.2 Verify Plug-in Settings for Different Formats

Format dependent properties can have a negative side effect. As almost every parameter can be different for various formats, it can happen that essentials settings, like the name of a control field or the way how a control text is intended to be received might vary between formats.

Copyright © 2026 Vizrt

Example



This screenshot of a Control Text plug-in illustrates what can happen. In the first format, the text is expected to be in plain format, but in the second format, it is expected to be sent as formatted text. Please make sure that essential settings like these are unique on all formats.

9.10.3 Text Limitations

Text offers some very powerful options on how to format your text. It supports a on character based formatting. However, when using text for multiple formats, only the whole text itself is format dependent.

If there is more than one format is available, a per character styling is not supported due to technical limitations. The following text does *NOT WORK* on multiple formats:

Copyright © 2026 Vizrt

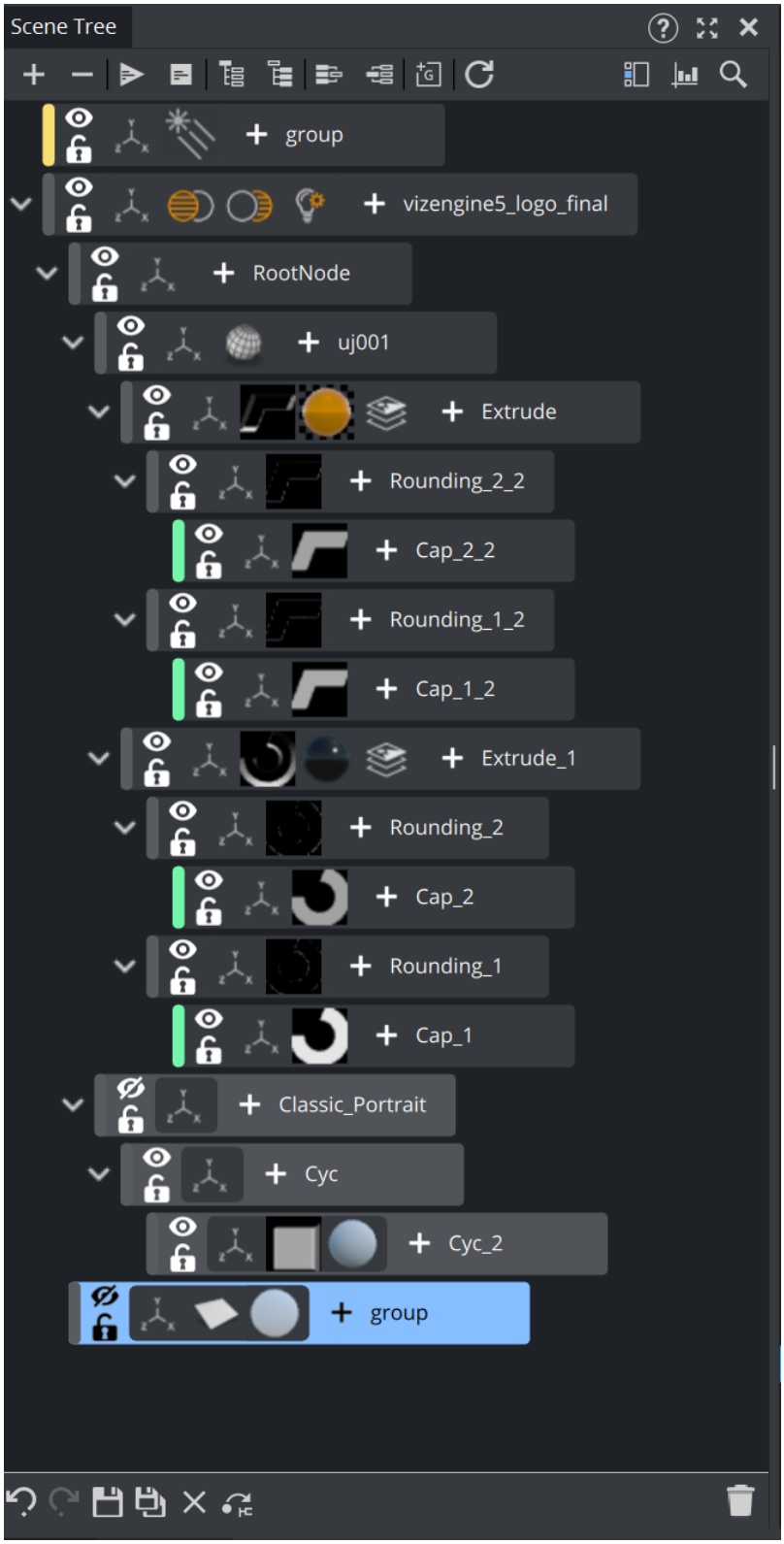


Copyright © 2026 Vizrt

10 Scene Tree

The Scene Tree is the logical visualization of all the properties in a Scene. The Scene Tree consists of Containers that hold properties. The properties can be Geometries, Images, Materials, Texts, Transparency functions, Key functions, and many more. Scenes are root items that hold other items. Created Scenes are stored in the Graphic Hub and are available through the Asset View. For general information about Scenes and procedures on how to create, import, rename, and delete Scenes, see Working with Scene Items.

Copyright © 2026 Vizrt



This section contains information on the following topics:

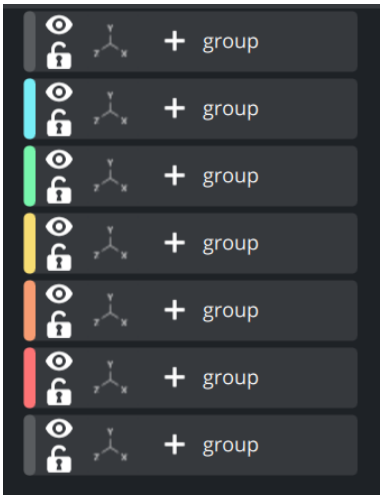
- Containers
- Plug-in Overlay
- Searching
- Sorting
- Subscenes
- Layer Manager

10.1 Scene Tree in General

The new Scene Tree in Viz Artist 5 has been streamlined and optimized for a better workflow. It offers new ways on adding properties to containers, navigating through large scene trees, as well as searching and sorting improvements. A few of the improvements:

10.1.1 Color Style

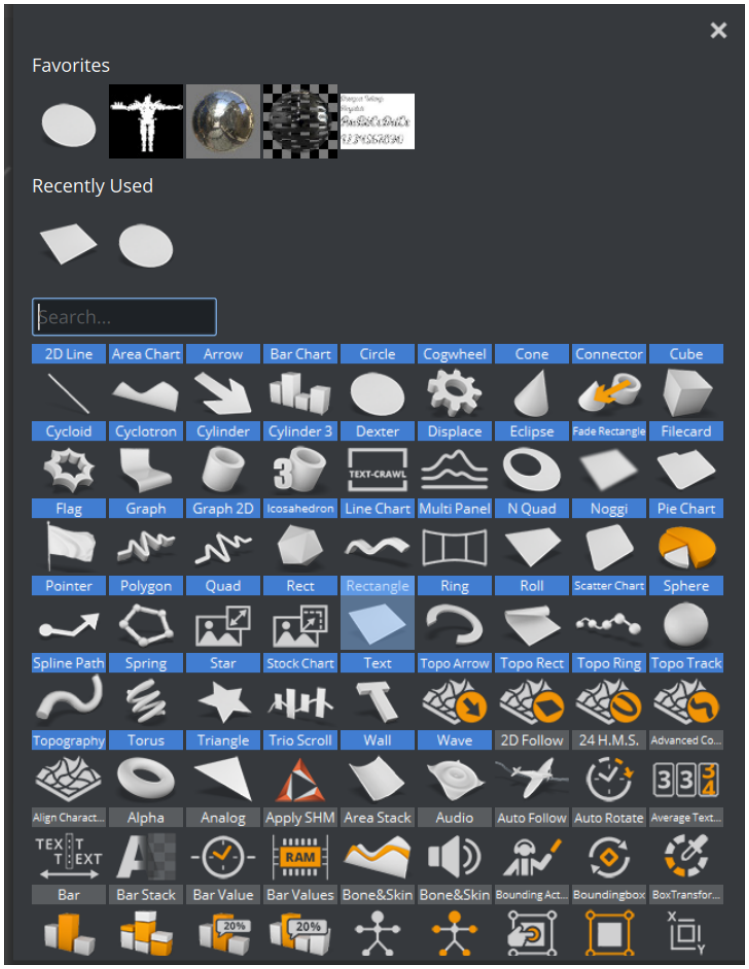
How Viz Artist represents a colored container can be set in **Configuration > User Interface**:

Highlight Container Setting	
Highlight Complete Container `ON`	Highlight Complete Container `OFF`
	

10.1.2 Plug-in Overlay

The Plug-in Overlay provides a quick way to add functionality, access your favorites and shows a history of the recently used assets.

Copyright © 2026 Vizrt



See Plug-in Overlay for more information.

10.1.3 Smart Scrollbar

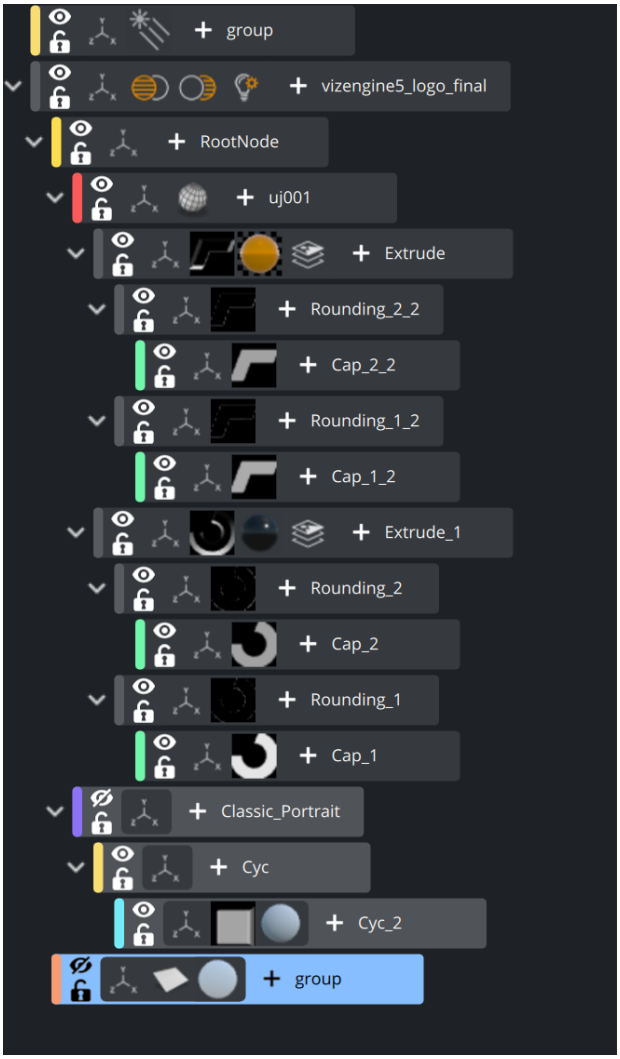
The Scrollbar shows various information:

- Currently selected containers are marked in the scrollbar to give a quick sense of where they are.
- Search results are shown in the scrollbar to immediately show the hits.
- Containers tagged with Color Labels are shown for easy navigation. Searching for color labels is also integrated into the new search panel.
- A heatmap during performance analysis.

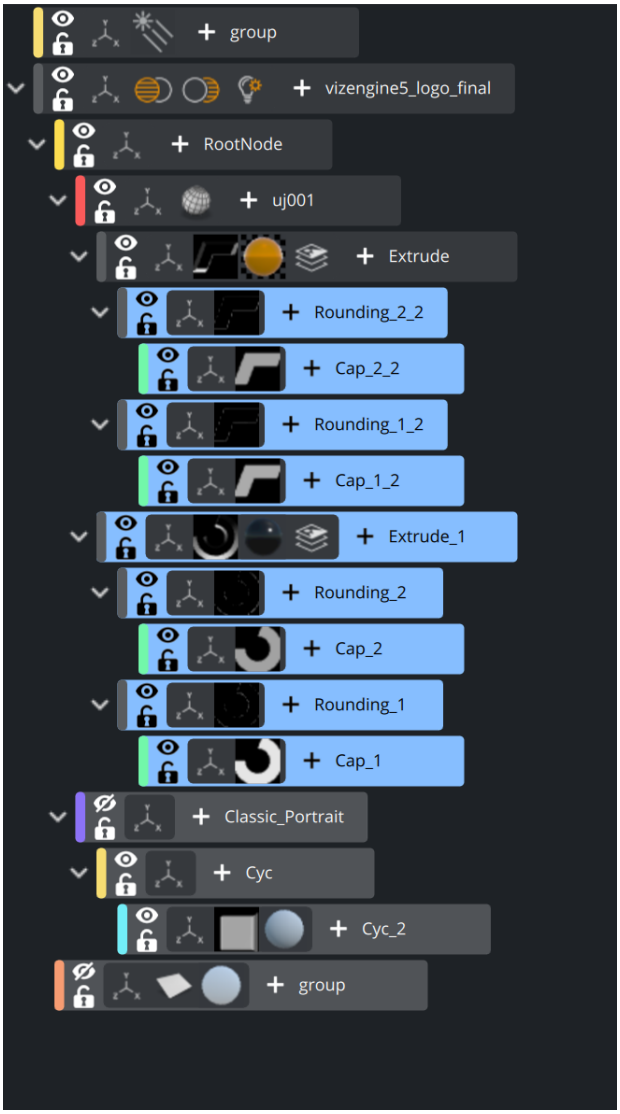
Examples

Scene Tree with Color Labels	Scene Tree with Selection Marked

Copyright © 2026 Vizrt

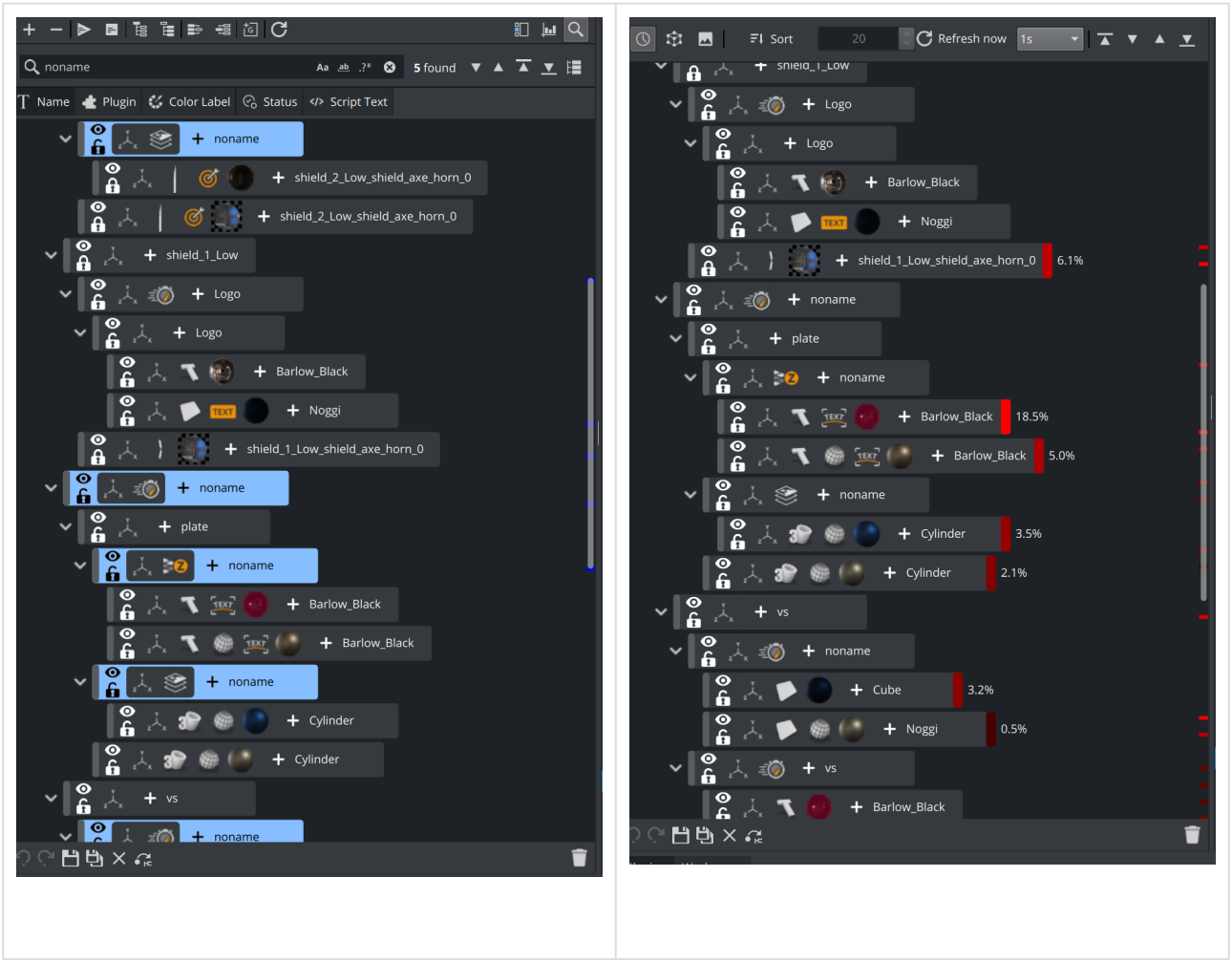


Search Result Shown in Smart Scrollbar



Heatmap Shown in Smart Scrollbar

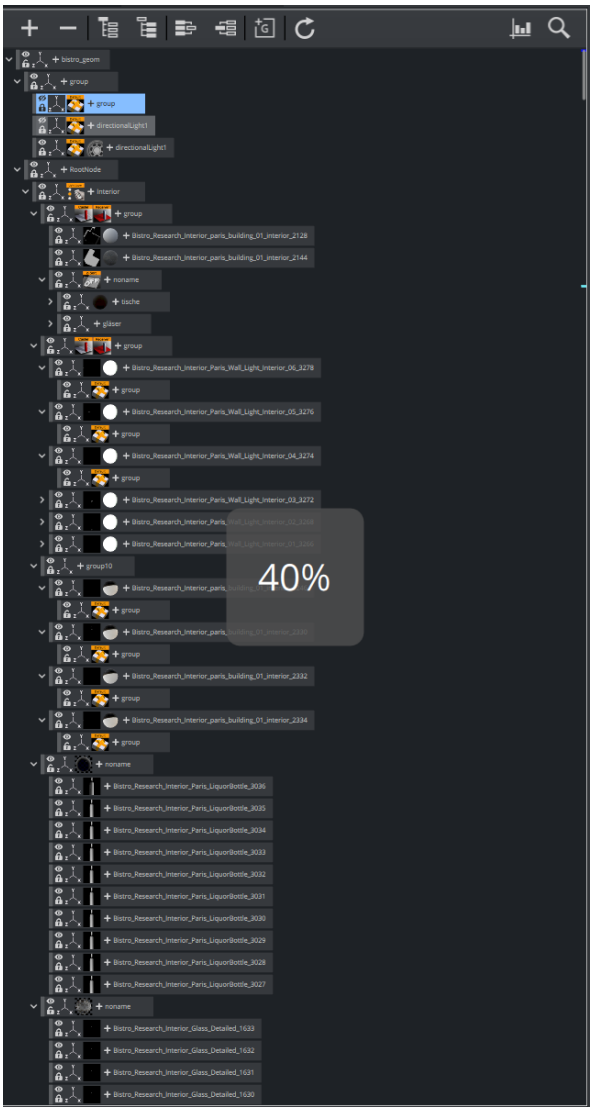
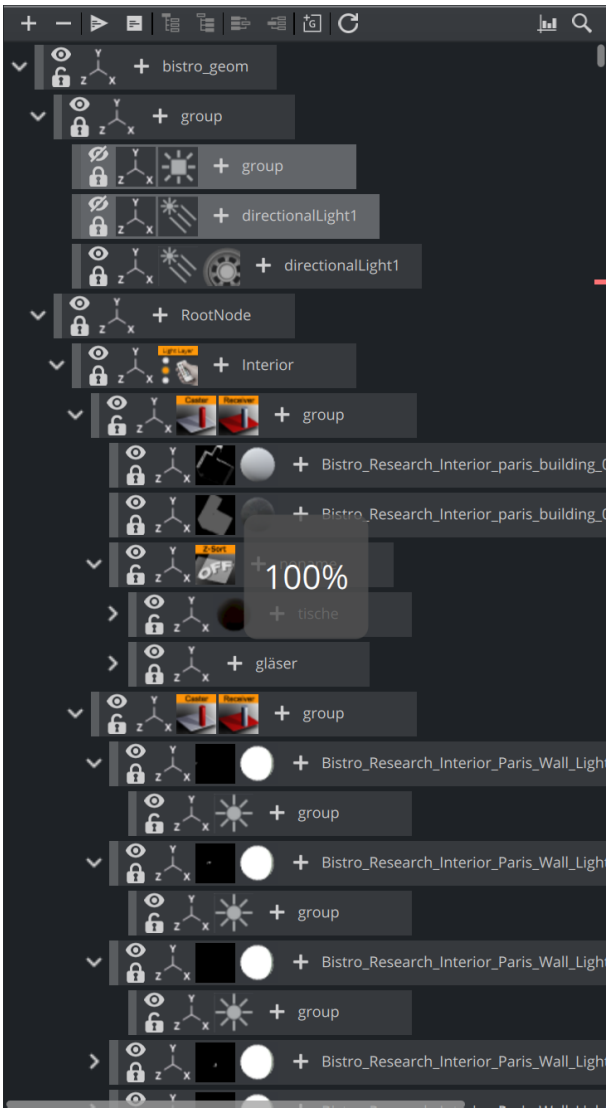
Copyright © 2026 Vizrt



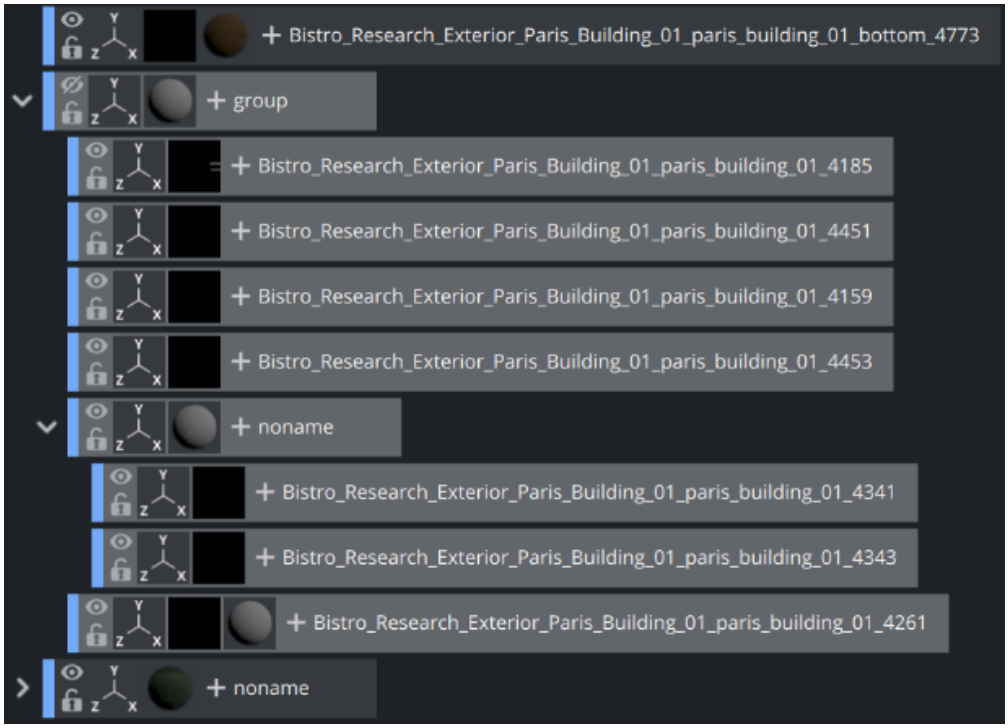
10.1.4 Zooming

By holding down the **CTRL + Mousewheel** the Scene Tree can now be scaled from small (40%) to large (100%):

Copyright © 2026 Vizrt

Scene Tree Zoomed out	Scene Tree Zoomed in
	

Active/Inactive status is now reflected on the container and its sub containers:



10.2 Toolbars

The top toolbar offers the following functions:

Icon	Description
	Expands the whole scene tree.
	Contracts the whole scene tree.
	Compiles and runs all scripts.
	Stops all scripts.
	Groups a selection of containers.
	Ungroups containers.
	Merges containers.
	Splits merged geometry.
	Creates new container.

Copyright © 2026 Vizrt

Icon	Description
	Refreshes the tree.
	Opens performance analyzer.
	Opens search.

The bottom bar:

Icon	Description
	Undoes last operation.
	Redoes last operation.
	Saves current scene.
	Saves current scene as specified filename.
	Closes current scene.
	Shows current scene in Asset View.
	Trashes items dragged here.

See Also

- Plug-in Overlay
- Working with Scene Items
- Asset View

Copyright © 2026 Vizrt

10.3 Containers

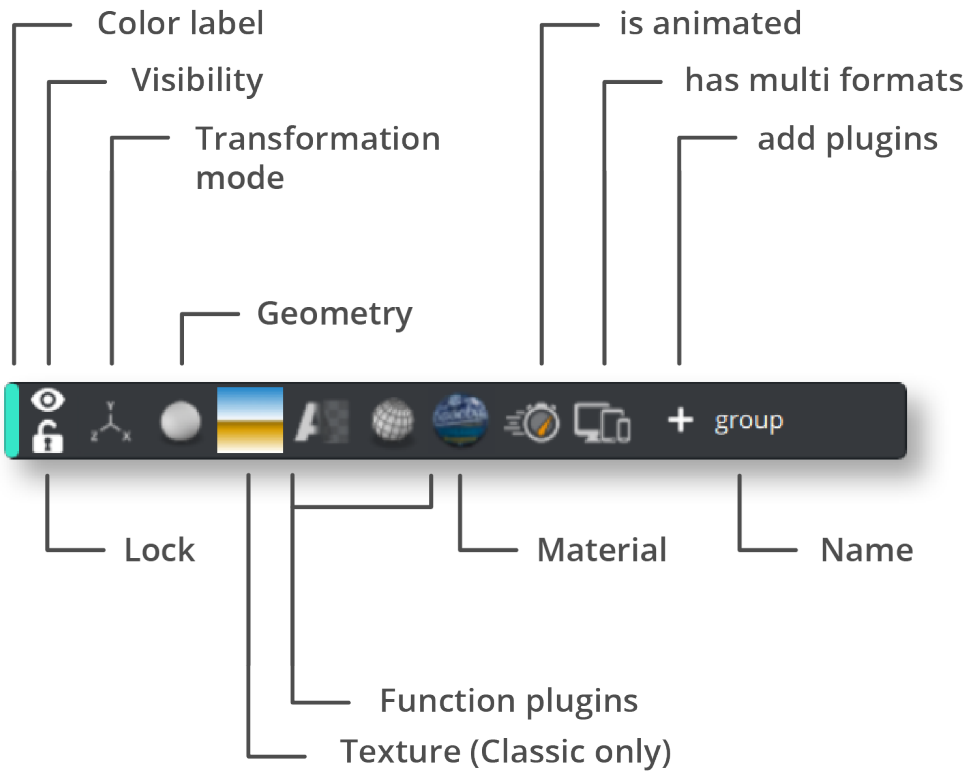
This section contains information on these topics:

- Container Components
- Container Names
- Basic Concept of Using a Container
- Selecting Multiple Containers
- Copying Containers
- Deleting Containers
- Grouping Containers
- Merging and Splitting Containers
- Container Context Menu
- Container Properties and Editors
- Plug-ins and their order

A scene is build by using a set of containers. Each container can hold multiple items and defines the layout and hierarchy of a scene.

Copyright © 2026 Vizrt

10.3.1 Container Components



A container consists of one of the following components:

- **Color Label:** Is always available and helps to quickly identify a container. See Color Labels for details
- **Visibility:** Turns rendering of the container and its children on or off. This is always available.
- **Transformation Mode:** Indicates if the container is used as a regular 3D container or if it is used in Box Transformation mode. Always available. The transformation is inherited to its siblings. See Box Transformation and 3D Transformation for details.
- **Lock:** Prevents unwanted modification in the scene editor. Always available. The container can still be modified by the Transformation editor.
- **Geometry:** Shows assigned geometry. A container can, but must not have a geometry assigned. This can be a merged or imported geometry or one of the Built-in geometry primitives. The thumbnail icon reflects the geometry of imported assets.
- **Texture:** This is available in Classic Scenes only. It represents a thumbnail of the texture this container is assigned to. Click the icon to open the Image Editor.
- **Function Plugins:** Function plug-ins enhance the functionality of this container and may also influence its siblings. There can be multiple function plug-ins assigned to a container.

Copyright © 2026 Vizrt

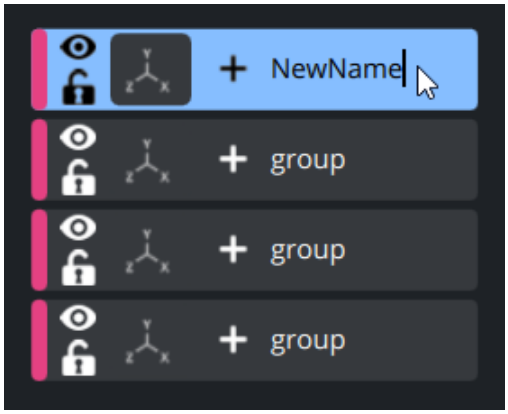
- **Material:** Represents a Classic Material for classic scenes or a Phong, PBR or other Material Definition based materials in Viz Engine pipeline. The material is inherited.
- **Animation**: Indicates that the container is animated.
- **Multi Format:** Indicates that the container has different properties for different formats. See for Working with Different Formats details.
- **+ Button:** Allows to quickly and easily add functionality to the container. See Plugin Overlay Panel for details.
- **Name:** Displays the container name. See Container Names for details.

10.3.2 Container Names

You can rename containers to ease identification of their role(s) in your scene design. Container names can be any combination of uppercase and lowercase letters, or numbers. However, they cannot contain any of the following special characters:

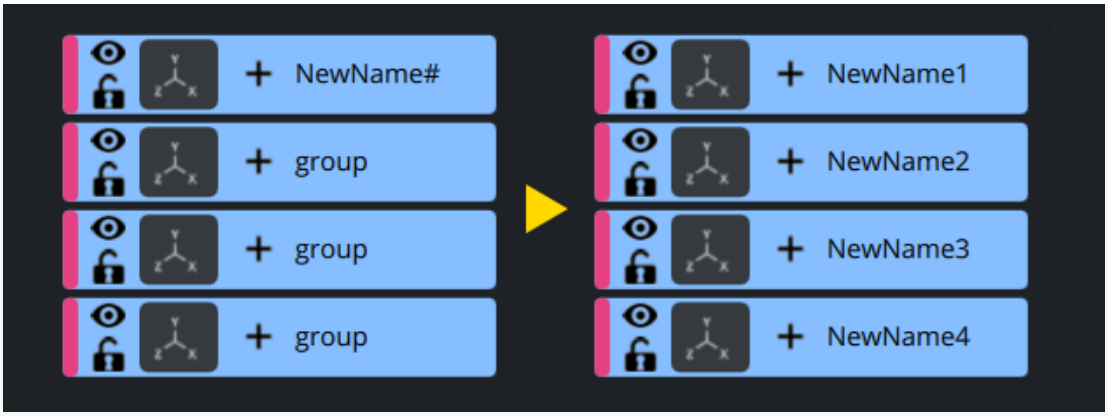
- `$` (dollar sign)
- `/` (slash mark)
- `\` (backslash)
- `@` (at sign)
- `*` (asterisk)
- `?` (question mark)
- `%` (percent)
- `"` (quotation mark)
- **{ }** (curly brackets)

To rename a container, click on its name:



To rename a selection of containers, you need to use the context menu.

Tip: If you use a # in the name of a multi selection, all your containers are automatically numbered:



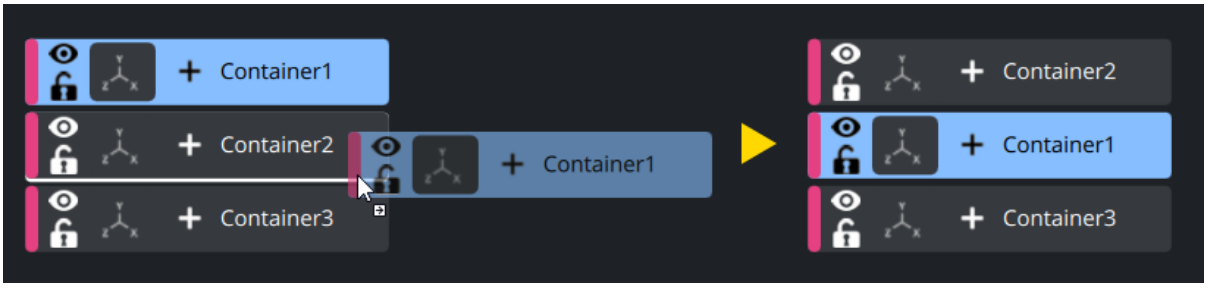
10.3.3 Basic Concept of Using a Container

A scenes structure is based on container and the way how they are organized. Containers can be added to the Scene by clicking on the icon or by dragging the icon to its final position within the Scene Tree.

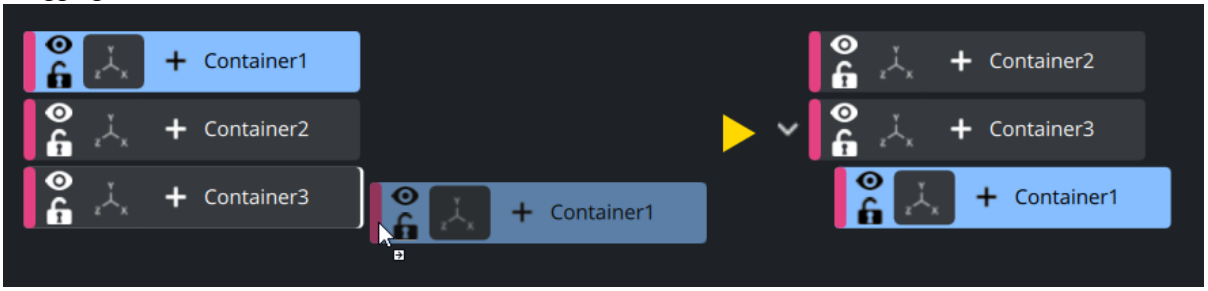
The position of the target destination is marked with a white line:

Examples

- Dragging Container1 on top of Container3 rearranges the order. It would be the same operation as dragging Container1 under Container2:



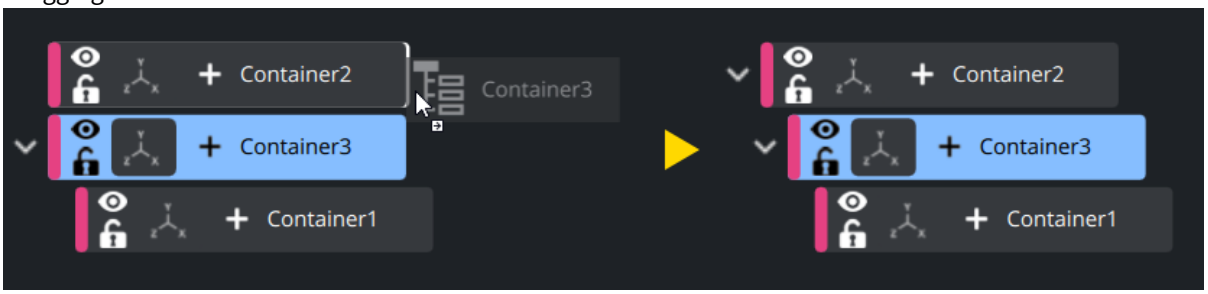
- Dragging Container1 next to Container3 moves Container1 as child of Container3.



Copyright © 2026 Vizrt

Note: On this operation, the transformation of Container3 is inherited onto Container1. Pressing the **ALT** button while moving a Container (or group of Containers) within the Scene retains the transformation properties. This lets a Container, which is part of a Group, to be moved to another group while the absolute transformation of the Container does not change, although the parent Containers (Groups) have different transformations.

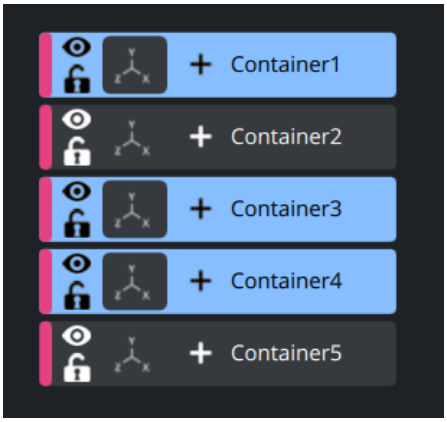
- Dragging Container3 and its Sub Container1 underneath Container2.



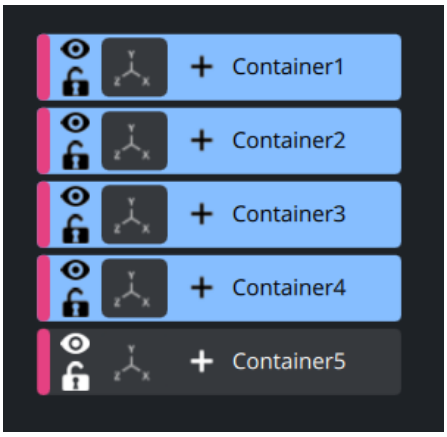
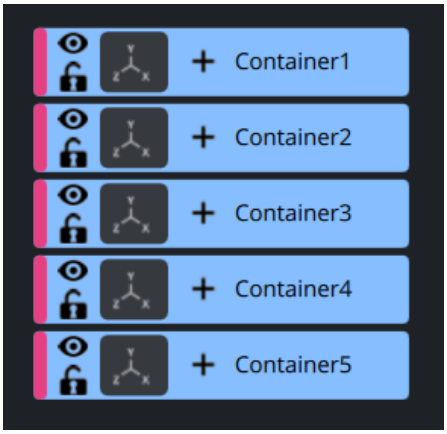
10.3.4 Selecting Multiple Containers

More than one Container can be selected in the Scene Tree. This can be done by using the **CTRL** key to select single containers or **SHIFT** to select a range. To select All containers use **CTRL + A.**

When multiple Containers are selected in the Scene tree, properties can be applied to all the selected items. This multiple selection feature is global.

Multi Selection	Result
CTRL Key 1. Click on a Container. 2. Press **CTRL** and click on all Containers to be selected.	

Copyright © 2026 Vizrt

Multi Selection	Result
SHIFT 1. Click on a Container. 2. Press **SHIFT** and click the last required Container. All Containers in between are selected.	
CTRL + A	

10.3.5 Copying Containers

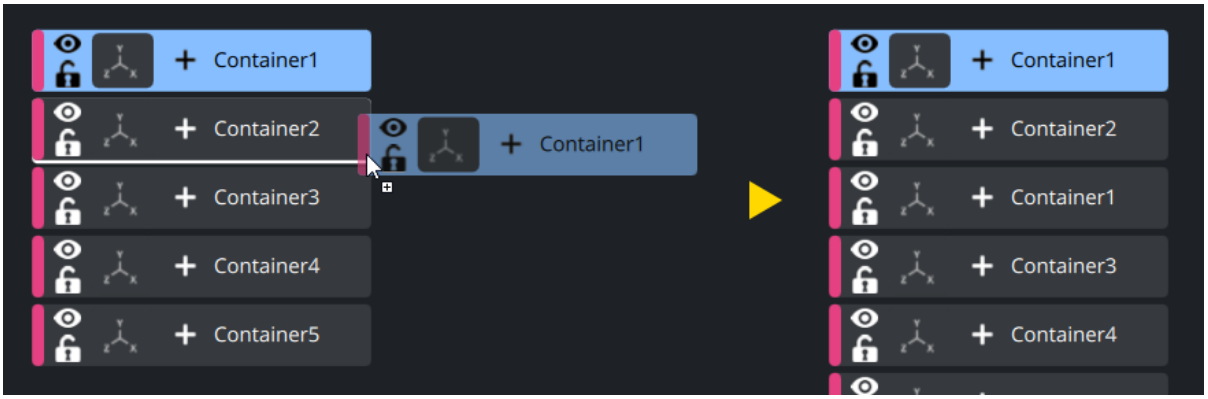
- Hold **CTRL** and drag the Container to its new position, or
- Use **CTRL + C** and **CTRL + V** to copy and paste the Container. Copied Containers are placed as Sub-Containers of the selected Container.

Note: When copying a Container with Sub-Containers, all Sub-Containers are copied as part of the same operation.

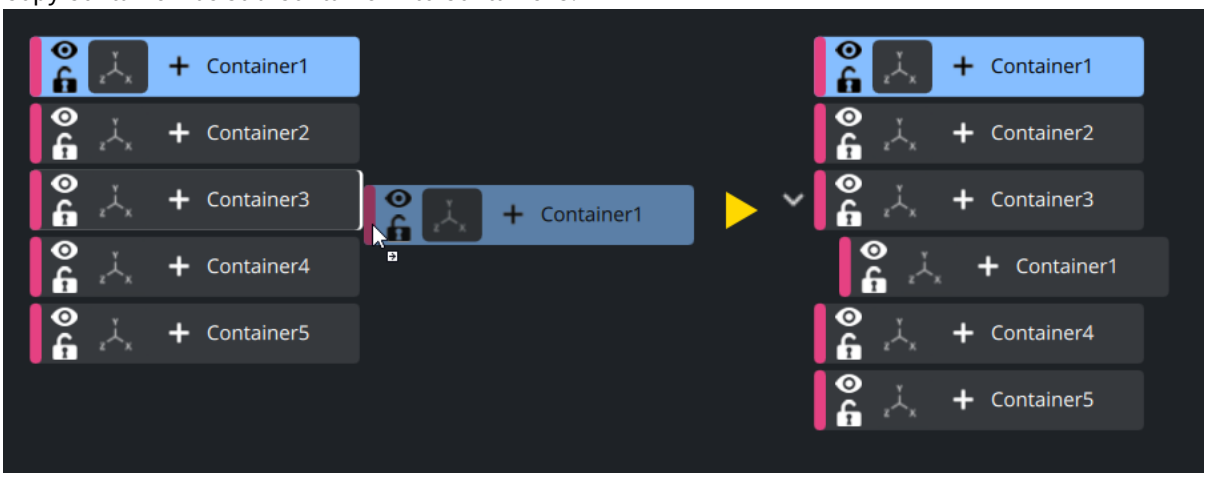
Examples

- Copy Container1 below Container2.

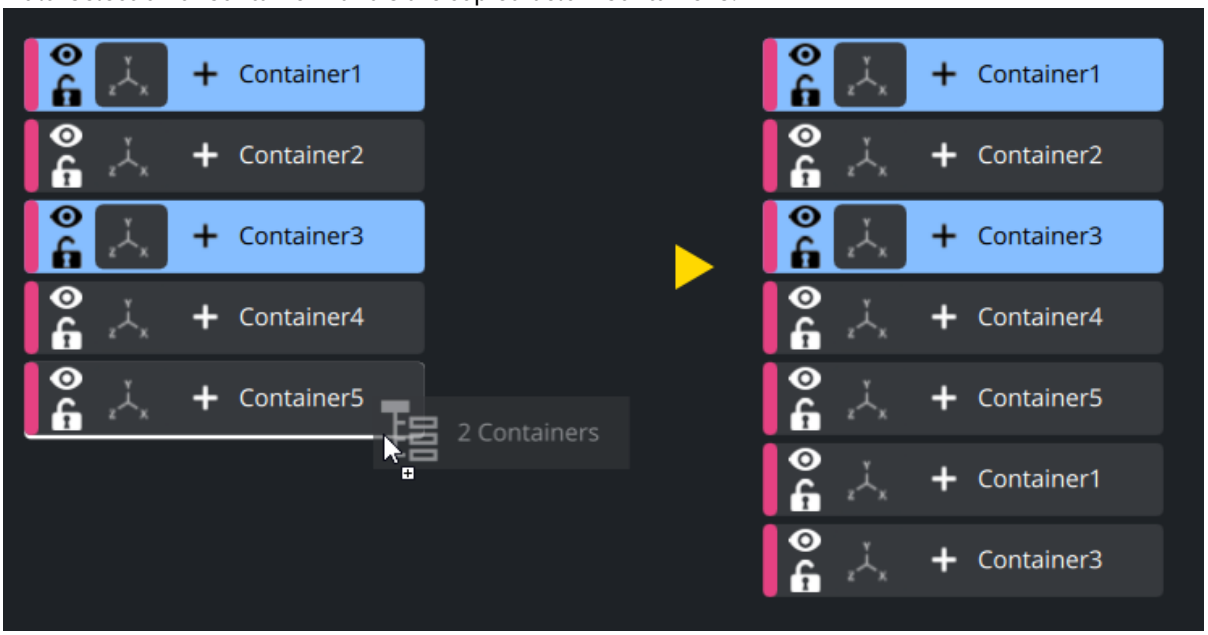
Copyright © 2026 Vizrt



- Copy Container1 as Sub-Container into Container 3.



- Multi-Selection of Container 1 and 3 are copied below Container 3.



Copyright © 2026 Vizrt

10.3.6 Deleting Containers

To Delete a Container from a Scene

1. Drag a Container to the trash can, or
2. Right-click the Container, and in the context menu select Delete Container [Name], or
3. Use the shortcut keys **CTRL + DELETE**.

> **Note:** If deleting a parent Container, all Sub-Containers are deleted too.

10.3.7 Grouping Containers

Containers in the Scene Tree can be organized in a hierarchy and grouped in logical divisions. The hierarchy depth is practically unlimited. Grouping Containers can be compared to creating a parent/child hierarchy. Such a hierarchy provides a better overview, especially in complex Scenes, and is very practical since changing the properties of the group (for example adding a new material) also applies the changes to all the Sub-Containers.

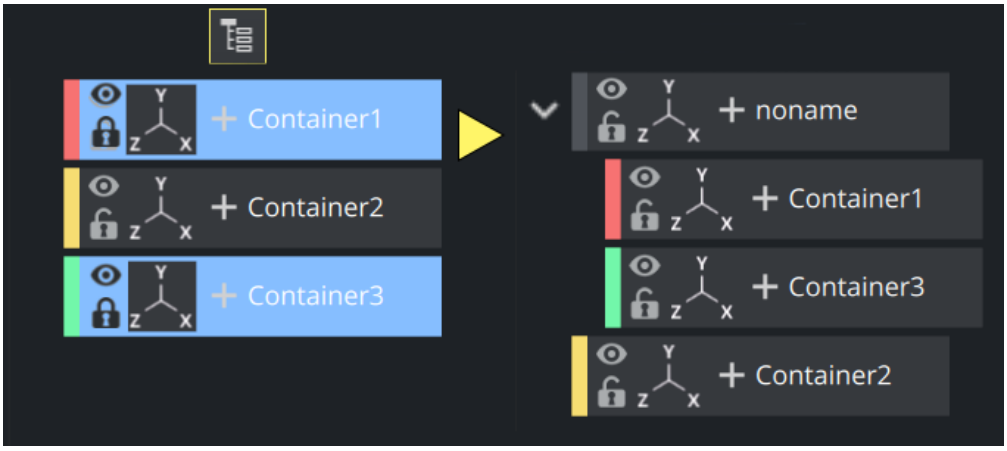
> **Note:** Changing the properties of one of the Sub-Containers (for example adding a material) overrides the settings for this particular Sub-Container.

Moving one group is also much easier than moving a lot of single Containers.

A parent Container, that has Sub-Containers, is marked with a white arrow to the left of the Container. If the arrow points to the right (), the Container is collapsed and the Sub-Containers are not shown in the Scene Tree. If the arrow points downwards (), the Container is expanded and the next level of Sub-Containers are shown. Any Container can basically be used as parent, although group Containers (that hold transformation only) are best suited.

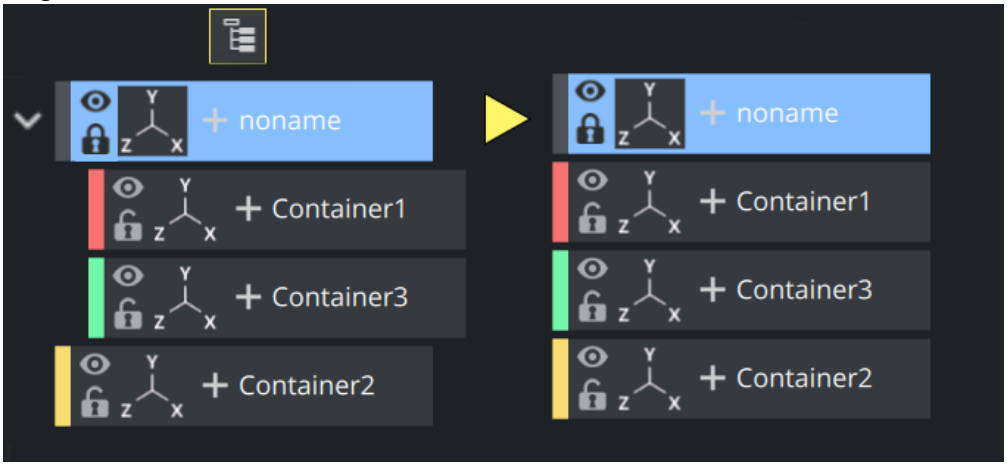
Examples

- **Group** (**CTRL + G**)
  Container 1 and Container 2 are grouped together into a new "*noname*" container, placed on top of the first selected container. Any transformation, operation or plug-in applied to the group also affects Container1 and Container3.



Copyright © 2026 Vizrt

- **Ungroup** (**CTRL + SHIFT + G**)
  The group "*noname*" is being ungrouped. Container 1 and Container 2 are taken out of the group, the group container "*noname*" remains. Any transformation, operation or plug-in applied to the "*noname*" group no longer affects Container1 and Container3!



10.3.8 Merging and Splitting Containers

Containers can be merged into compound objects and saved as reusable components to the Graphic Hub database. Any animations are also merged and added to the new compound object's Container.

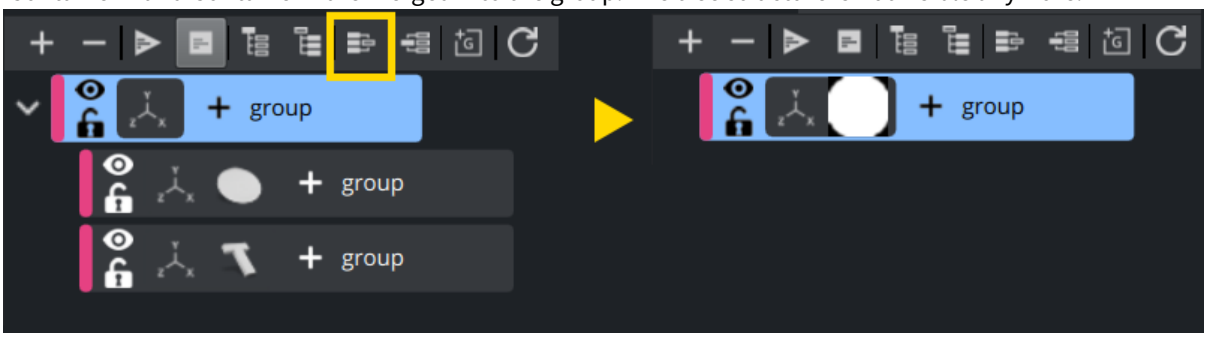
A saved compound object can be used in other Scenes have an active reference between the stored object and all Scenes where it has been used by default. Through use of stored objects with active references, changes can be applied to many Scenes at once, which is typically a meaningful way to work when designing a set of Scenes that share many design items. The geometry of a compound object cannot be edited. A compound object must be split for the geometry to be edited again. Splitting a referenced compound object disables the reference.

> **Note:** Some older scenes are converted to use a newer merge method, a message "*Old merge style, convert to new merge style*" informs the user.

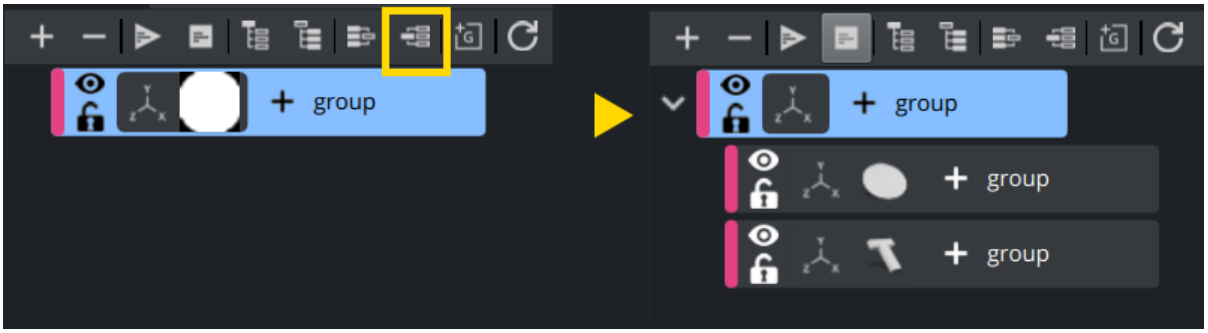
An alternative to merged scenes is the new Scene-in-Scene functionality.

Examples

- **Merge** (**CTRL + M**)
  Container 1 and Container 2 are merged into the group. The tree structure is not visible anymore.



- Split **(CTRL + SHIFT + M)**
  The merge group is split up again into its single components. You need to expand the resulting container to see its content.



To Save a Compound Object to a Graphic Hub

1. Right-click a Container (with or without Sub-Containers),
2. In the context menu select **Merge Container**.
3. Drag the compound object's icon from the property editor or Scene Tree and drop it onto a folder/project in the **Server Panel**.
4. Type a descriptive name for the object in the dialog box
5. Click **Ok**. The object is now visible in under the **Server Objects** tab.

Tip: Any animations or scripts used within the object are kept when saving.

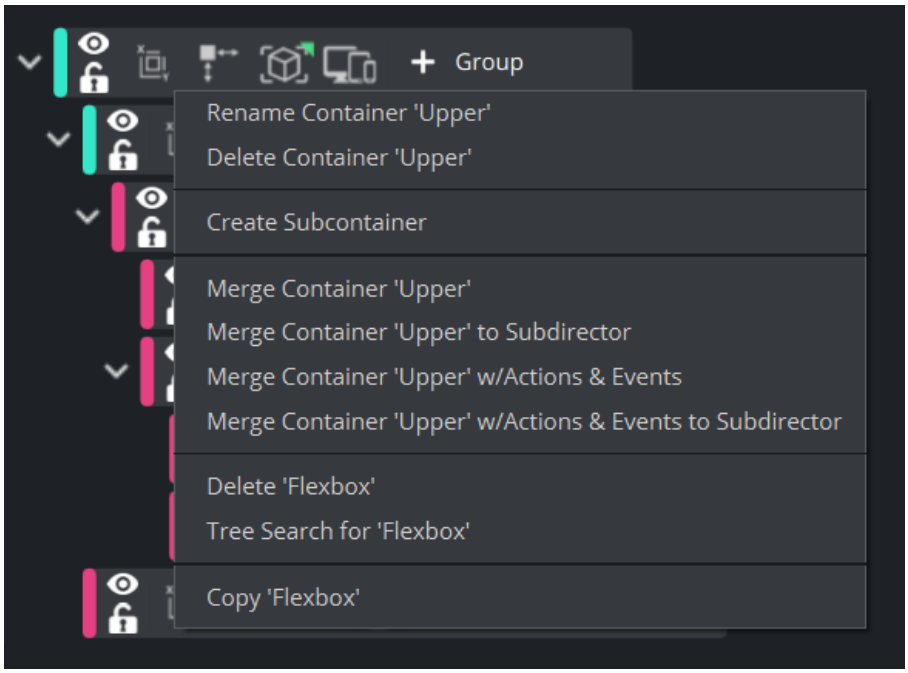
Note: Make sure to drag only the object icon, not the whole Container.

10.3.9 Container Context Menu

A right-click on a Container in the Scene Tree opens a Container context menu.

Copyright © 2026 Vizrt

Right-click on a Container Name



Right-click on the Container name to show these context menu options. Depending on which component you are clicking on, the content of the menu might vary.

Global Container Functions

- **Delete Container <name>:** Deletes the selected container.
- **Create Sub Container:** Creates a new empty subcontainer for the selected container.
- **Copy/Cut/Paste:** Allows to copy and paste various Container Components or the whole container.

> **Information:** Please be aware that now it is possible to copy components too, in previous versions this was copying the whole container only.

- **Merge options:**
  - **Merge Container:** Merges the selected Container and all its Sub-Containers into one compound object, or extracts an already merged object.
  - **Merge Container to Sub Container:** Merges the selected Container(s) and all the respective Sub-Containers into one compound object and creates a new sub director for the merged animation.
  - **Merge Container w/ Actions & Events to Sub Director:** Merges the selected Container(s) and all the respective Sub-Containers into one compound object, including actions and events and creates a new sub director for the merged animation.
  - **Merge Container w/ Actions & Events to Sub Container:** Merges the selected Container(s) and all the respective Sub-Containers into one compound object, including actions and events.

Copyright © 2026 Vizrt

- **Change to Box Transformation / Change to 3D Transformation:** Changes the coordinate system (Viz Engine Render Pipeline only). Box Transformation is 2D mainly and works with Flexbox Plugin. 3D is traditional xyz-space.

Content Menu for Images/Materials/Geometries

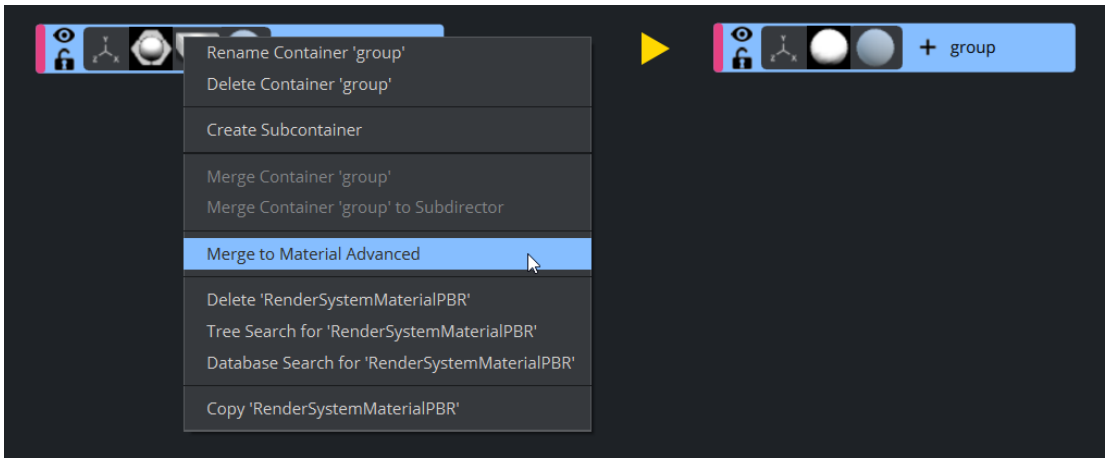
- **Delete property:** Deletes the property from the Container.
- **Tree Search for property:** Opens the **Search** and searches the Scene Tree for instances of this property.
- **Database Search for property:** Searches for the property in the database, and shows the result in the Server Panel.

Content Menu for Media Assets

- **Show settings of <Media Asset>:** Opens the Texture View (Media Asset Manager) and the Media Assets tab of the Scene Settings in the Property panel (see Media Assets).

To Create and Split Material Advanced

A combination of Image, Material and as Shader can be combined and stored as Material Advanced.



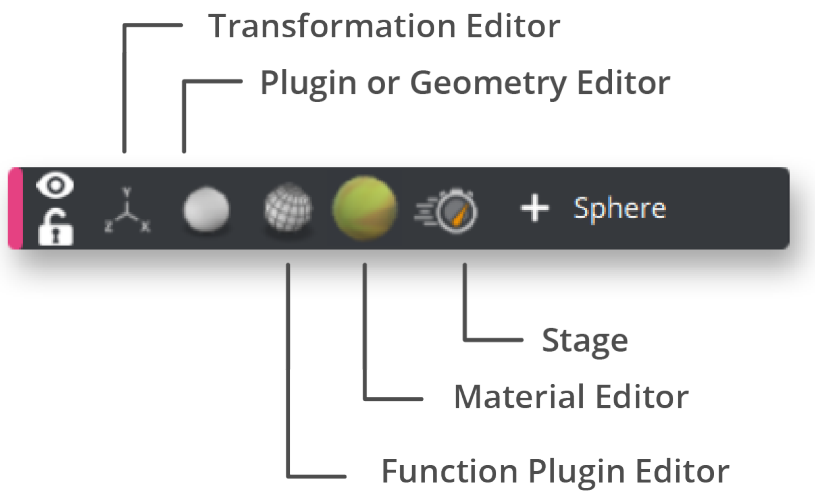
This Material Advanced can then be dragged into the Asset View to store it. Once a Material Advanced has been created, it can be split again to access the original components.

Note: This functionality is for the Classic Render Pipeline only!

10.3.10 Container Properties and Editors

Container Editors

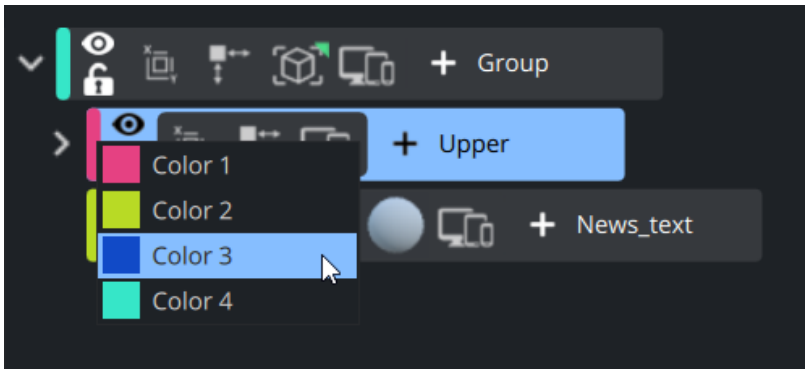
Clicking on any of a container activates different editors:



- In Classic mode, clicking on a texture opens the Texture Editor.
- If you are using a built-in geometry (like a Sphere), the Plug-in Geometry Editor is opened, if a imported or merged geometry from Graphic Hub is used, the geometry editor is shown.
- A font (Classic or Viz Engine) is also handled as a geometry, therefore the respective Font Editor (Classic or Viz Engine) is opened.
- Multiple plug-ins can be assigned to a container, each of them has its own configuration interface.
- Viz Engine scenes can hold material definitions like Phong, or PBR materials. Each of them has its own editor.
- If an animation is applied to a container, a click on the gear icon jumps to the stage.

To Add Color Labels

A label can be assigned to a container to categorize them or to easily locate them on larger scene trees. For instance, every container that is going to be controlled by an external application can get a yellow label. The number of labels as well as the color and the text can be configured in the User Interface section of the configuration. Labels are also used as search criteria.



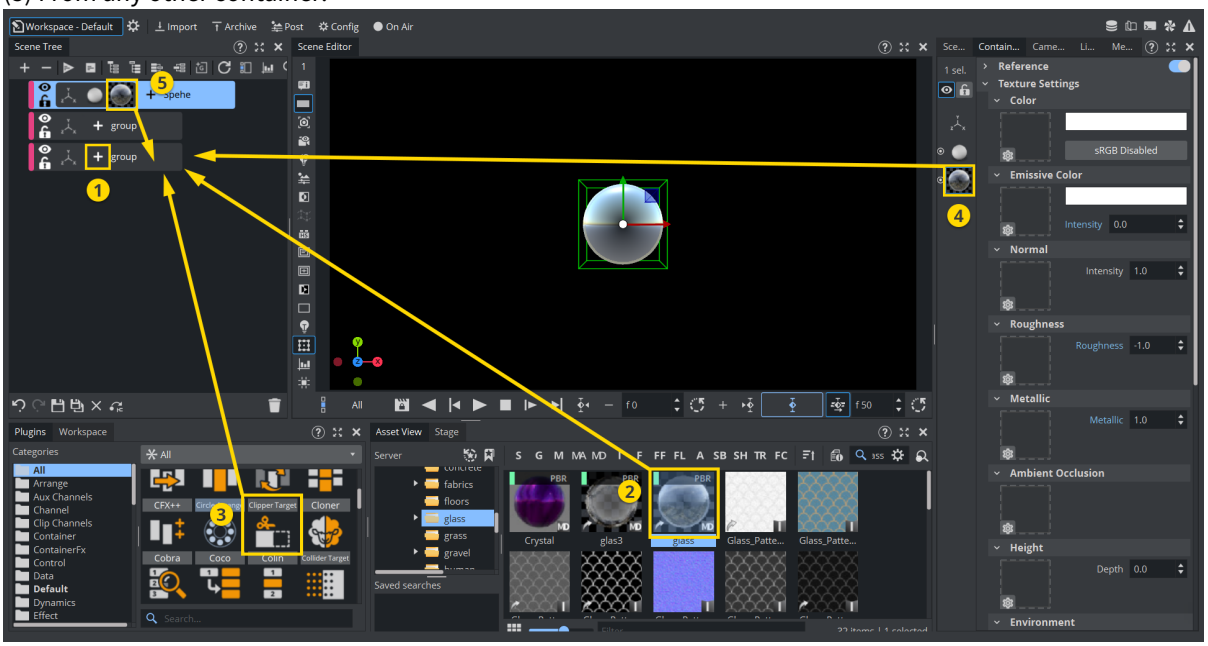
Copyright © 2026 Vizrt

You can choose if you want your labels to colorize the whole container or only the left side. See Scene Tree.

To Add Properties

To add new properties, several sources can be used:

- (1) Use the Plug-in Overlay Panel to quickly assign any function to a container.
- (2) Fonts, Images, Geometries can be dragged from the Asset View.
- (3) Built in Geometries, Media Assets and Function Plug-ins can be dragged from the Plug-ins Panel to a container.
- (4) From the sidebar of an selected container to the destination container.
- (5) From any other container.



These can be dragged either to the container or to the sidebar (4) of any selected container.

When multiple Containers are selected in the Scene tree, properties can be applied to all the selected items. This multiple selection feature is global. Select the required Containers then apply the appropriate property.

These properties can be added to a Container:

Items (Classic)	Classic Render Pipeline	Viz Engine Render Pipeline

Copyright © 2026 Vizrt

• Geometries • Materials • Material Advanced • Images • Fontstyles • Phong Materials • PBR Materials • Fontfaces • Lights • Planar Reflections • GI Settings	✓ ✓ ✓ ✓ ✓ - - - - - -	✓ ✓ ✓ ✓ (1) - ✓ ✓ ✓ ✓ ✓ ✓
Built Ins		
• Geometry plug-ins • Container plug-ins • Classic Shaders • Viz Engine Shaders • Media Assets	✓ ✓ ✓ - ✓	✓ ✓ - ✓ ✓
(1) When dragging an image in Viz Engine Render Pipeline, a quad geometry and correct Phong material are created.		

Only one Geometry plug-in, one Shader or one Media Asset can be applied to a single Container. If a new Geometry plug-in, Shader or Media Asset is added the first one is removed from the Container.

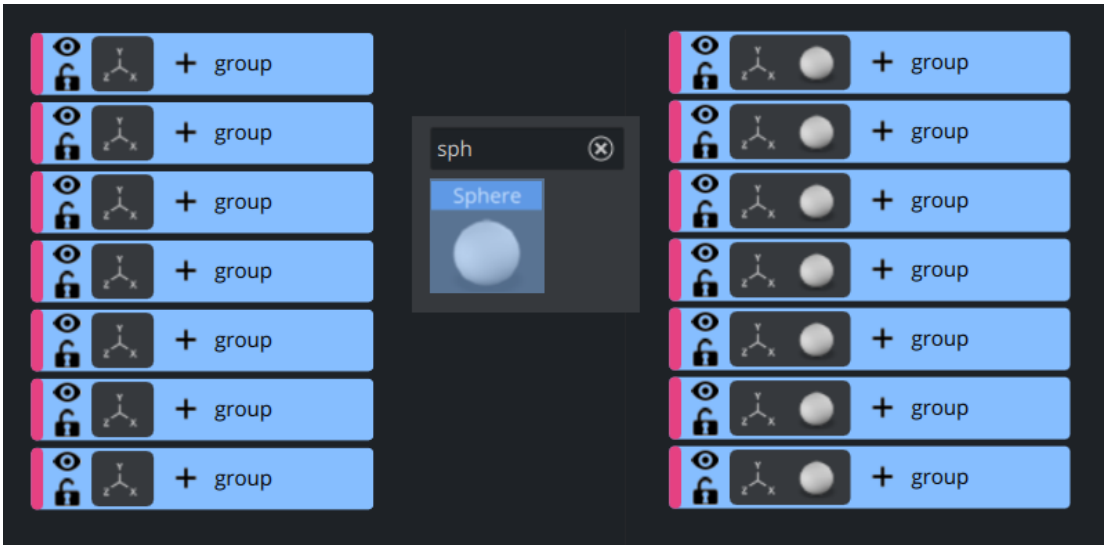
Tip: If a **Shader**, **Media Asset** or a few **Container plug-ins** are dragged to an object in the Scene Editor, they are added to the Container which contains the Object. The **Media Asset** is added as a Texture. If a Container plug-in can be added this way, the object becomes active when the Container plug-in is moved over the object.

Tip: If multiple Geometry or Image items are dragged from the **Assets View** to the **Scene Tree**, a Container for each selected item is created.

Example

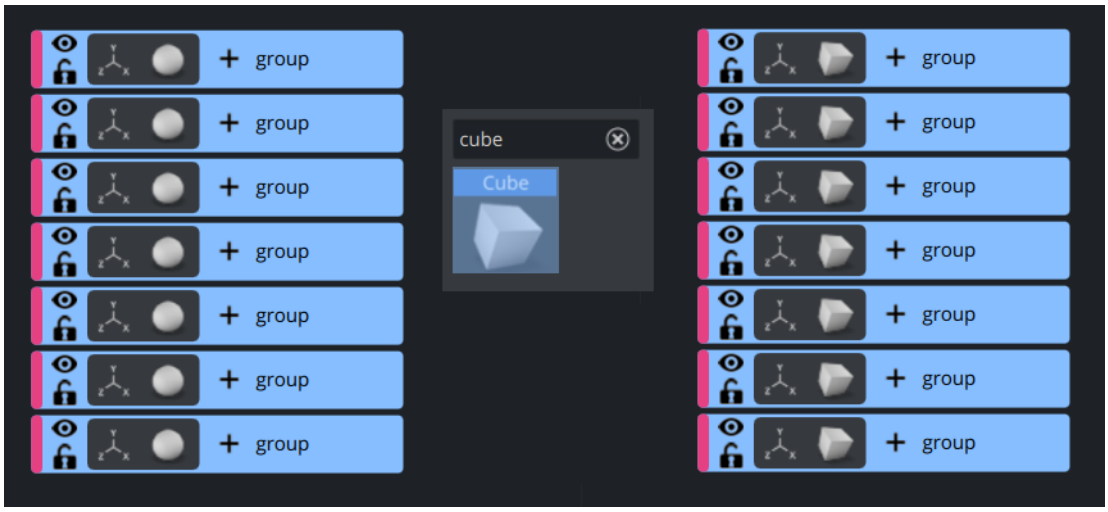
- Create a few containers.
- Select all of them by clicking on the first and last one.
- Open the Plug-in Overlay Panel, type `sphere` and add it to the selection:

Copyright © 2026 Vizrt



To Change Specific Properties on Containers

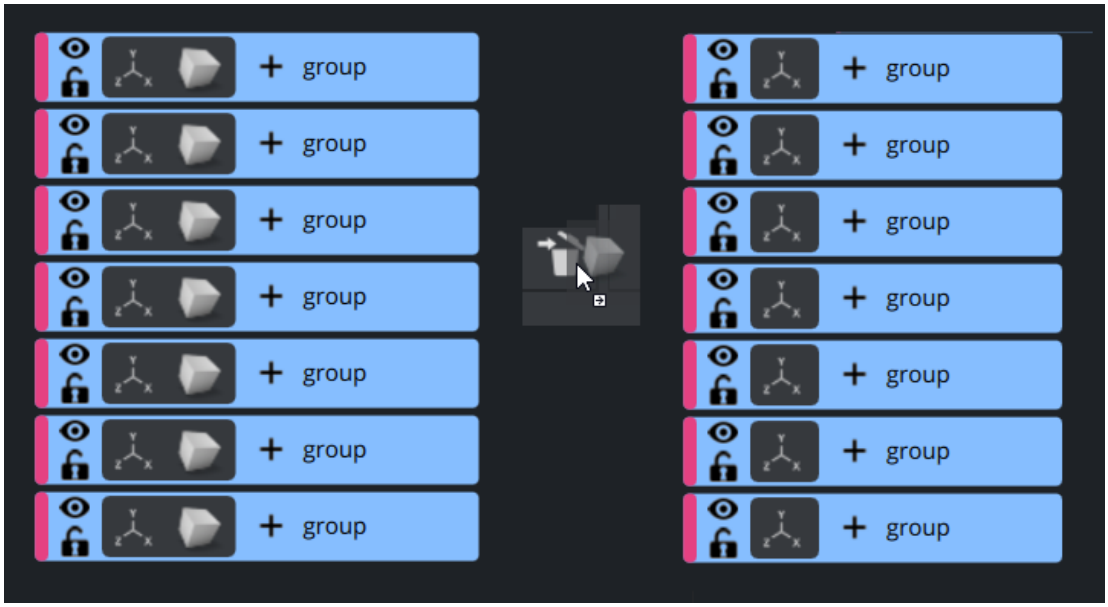
To replace our Sphere with any other geometry, grab any geometry and drag it to the destination container. This works also on multiple containers.



To Delete Properties

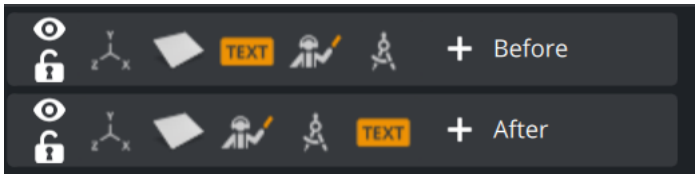
Grab your property and move it to the trash bin. Make sure you grab the Cube geometry and not the container itself.

Copyright © 2026 Vizrt



10.3.11 Plug-ins and their order

In some rare cases, the order of certain function plug-ins can be important as they rely on each other (for example, Justifier after TextBG). This is especially important for some layout plugins like Autofollow or TextBG. Sometimes it is required to change the order of plugins, therefore you can drag a function plugin to a new position inside the container.
Click on a function plug-in, and drag it to a new position:



Info: There is no visual indication for this operation.

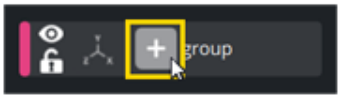
See Also

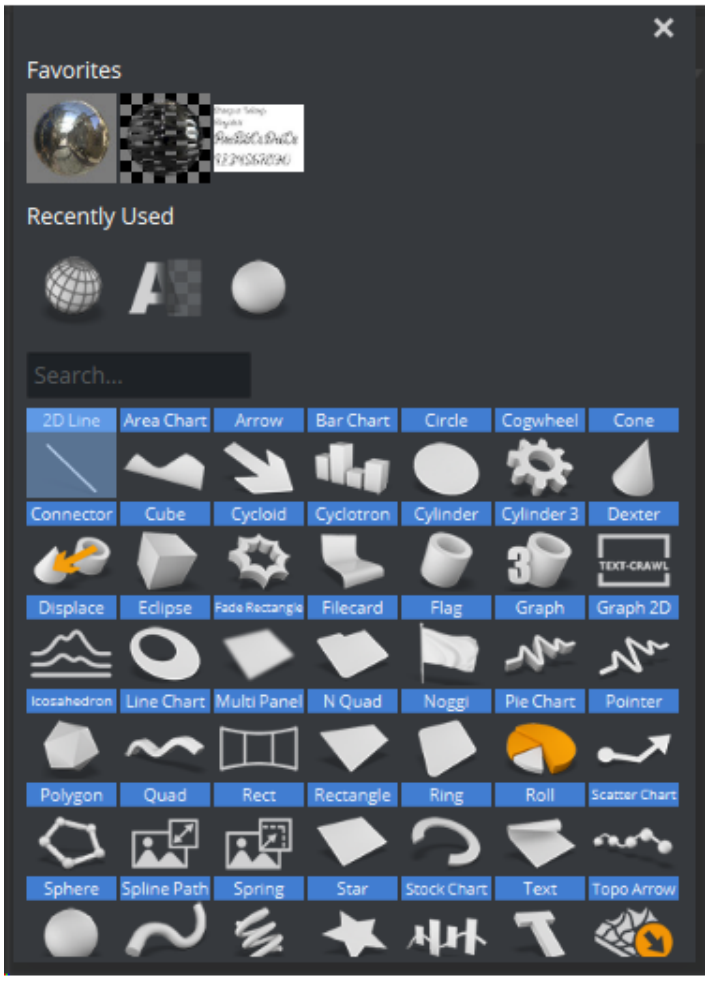
- Working with PBR Materials
- Working with Planar Reflections
- Working with Various Materials
- Scene Tree
- For more information about setting default merge and split behaviors, see the **User Interface** page in the **Configuring Viz** section of the Viz Engine Administrator Guide.
- For more information about how to configure colors, see the **Colors** page in the **Configuring Viz** section of the Viz Engine Administrator Guide.

Copyright © 2026 Vizrt

10.4 Plug-in Overlay

To quickly add new functions to your container (like plug-ins, materials, fonts or similar) the Scene Tree offers a very efficient and quick way. Each container holds a small button . Clicking on this opens the Plug-in Overlay window. Pressing the **+** Button on your keyboard can also quickly bring up the overlay.





The overlay panel combines several functions into one:

- Quick access to all your plug-ins.
- Favorites.
- History of last used items.

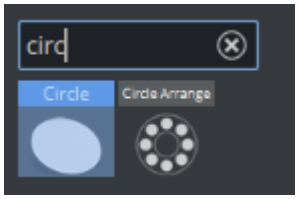
Copyright © 2026 Vizrt

10.4.1 To Add Plug-ins

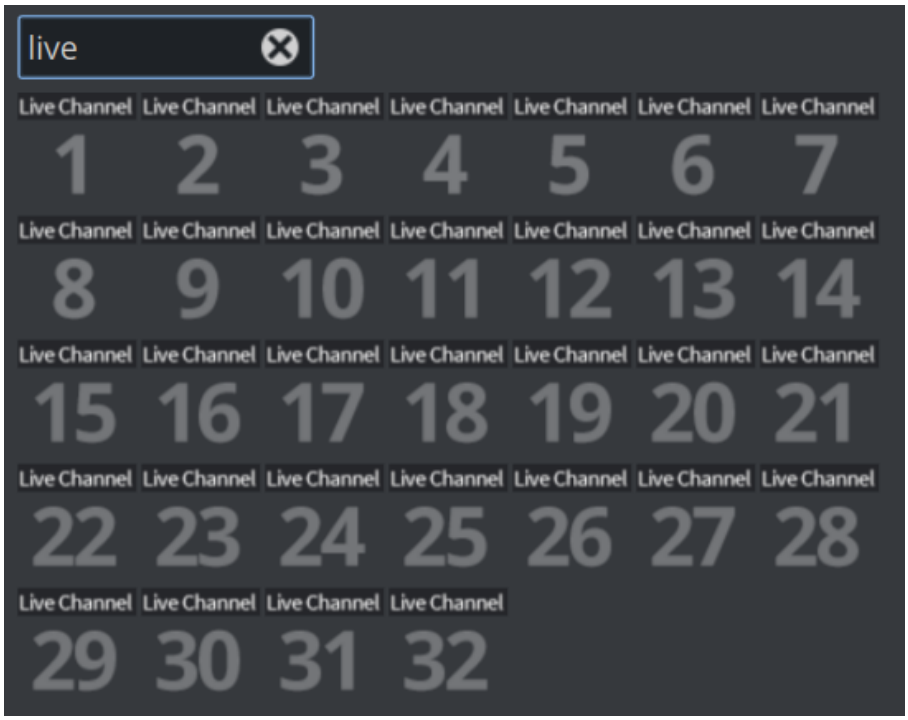
The overlay panel lists all installed plug-ins and media assets as well as internal Viz Engine functions (like Viz Engine lights, material definitions and similar). Click on one of them and it is added to your selected container(s). To filter your result, type a few characters of your plug-in name and the selection shrinks to match the search criteria. A plug-in can be added either by clicking on it or by pressing **ENTER** on the selected plug-in. The selected plug-in can be controlled by using the Arrow Keys.

Example

Open the panel and type `circ` : to show plug-ins with *circ* in the name.



Type `live` to show all available live sources.



The list can be scrolled by using the mouse wheel.

Tips:

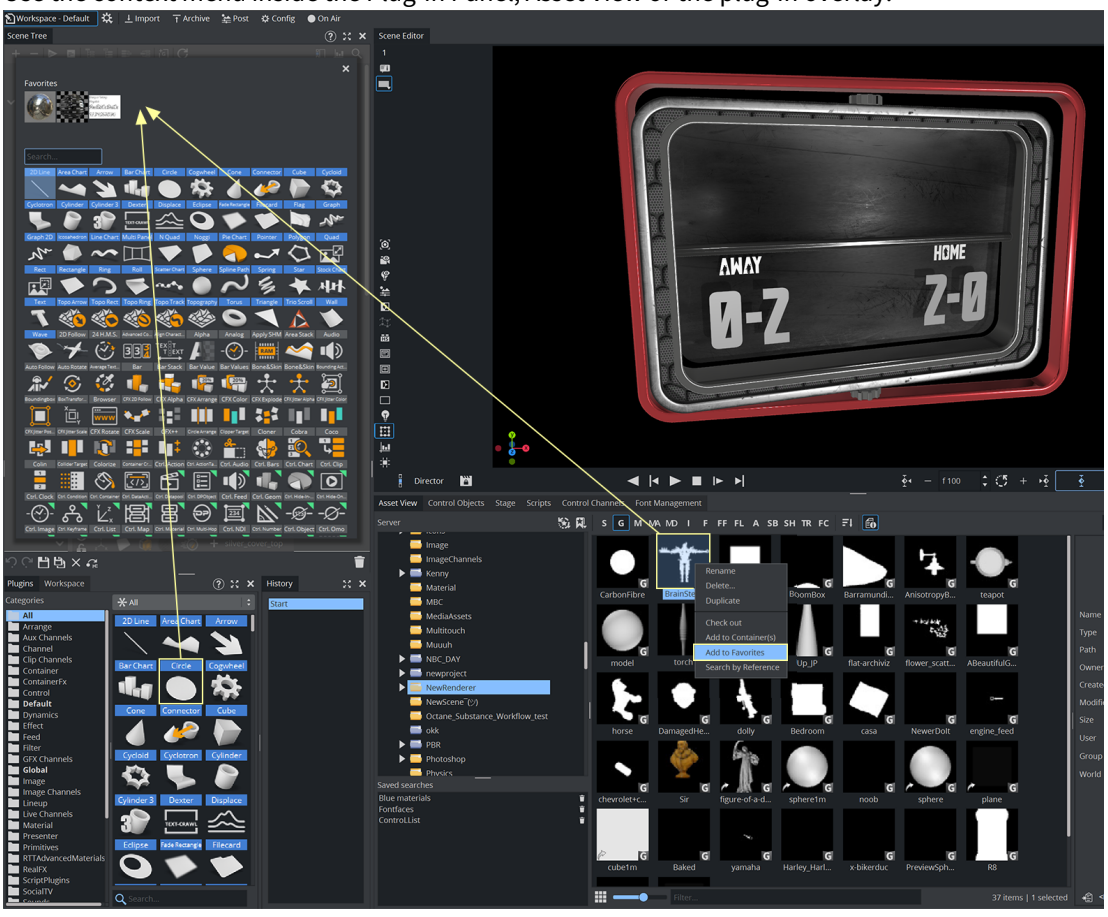
- You can select multiple containers and use the plug-in panel to apply a plug-in to a larger selection or a search result of containers.
- Phong, PBR, Reflection, Invisible and Classic Materials can be added too.
- Hover over the icon and wait to see a tooltip with the full name of the plug-in if you are not familiar with the icon.

10.4.2 To Manage Favorites

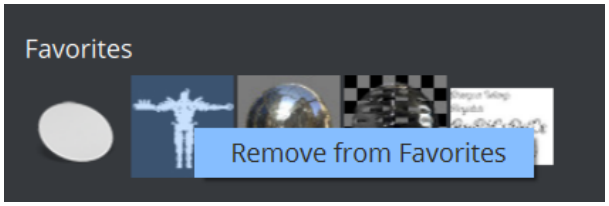
For quicker access to items that are used often, you can add items to the Favorites section within the Plug-in Panel. This section is located on the top and allows quick access to your most common used assets.

To add plug-ins to the favorites, open the Plug-in Panel by hitting the icon and

- Drag them from the Plug-in Panel or Asset View.
- Use the context menu inside the Plug-in Panel, Asset View or the plug-in overlay.



To remove favorites, right click on them in the Plug-in Panel or Plug-ins Dock and use the context menu:



The favorites list can be scrolled by using the mouse wheel.

10.4.3 Recently Used Items

The Plug-in Overlay also provides a collection of the recently used items. This list is per session, the list is blank the next time you start Viz Artist.

Copyright © 2026 Vizrt

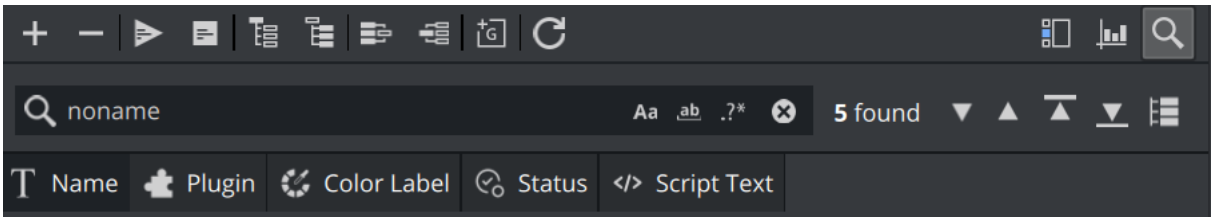
10.5 Searching

- To Search for Container Names
  - Match Case
  - Exact Match
  - Regular Expressions
- To Search for Plug-ins
  - Known Limitations
- To Search for Color Labels
- To Search for Containers Status
- To Search for Scripts Fragments
  - Known Limitations

Especially on larger scenes, the scene tree can get really large and confusing. A tool to find assets inside the scene tree is available on the top menu of the scene tree.

Click on the icon or use **CTRL+F** to open the search tool. The input field allows to switch between **Match Case** (UPPER/lower case is taken into account), **Exact Match** or **Regular Expressions**.

You can search for various types. To change the type, click on the buttons on the bottom:



This allows to change the type of search, the following are available:

Type	Description
Name	Searches for container names.
Plugin	Searches for plug-ins assigned to containers.
Color Label	Searches for labels assigned to containers.
Status	Searches for a certain container status.
	Searches for locked containers.
Script Text	Searches for script fragments inside scene and container scripts.

Each various search method shares the same common buttons and information:

Copyright © 2026 Vizrt

- : Jumps to **the next container** that matches the search criteria.
- : Jumps to **the previous container** that matches the search criteria.
- : Jumps to **the first container** that matches the search criteria.
- : Jumps to **the last container** that matches the search criteria.
- : Selects **all container** that matches the search criteria.

The number of matches is shown next to the search fieldand the respective locations are marked inside the scrollbar as blue markers, if the affected containers are selected. This allows to use either the navigation buttons described above to or to quickly scroll to the target location.



Copyright © 2026 Vizrt

10.5.1 To Search for Container Names

Your search can be customized in various ways. By default, the search does a string matching. If any part of your string matches the container name it finds, asterisks can be used as placeholders.

Default Search - Examples	
oup	Finds all containers having "*oup*" somewhere in the name, this is the same as "**oup**".
*o*o*	Any container with an o inside followed by another o.
**up*	Lists containers like "*group*", but also containers named, for example "*upper*".
*lower*third*	Container contains '*lower*' and '*third*' in this order. This also matches with names like "*lower_third_1*".

Aa Match Case

Turns on or off case sensitive search.

ab Exact Match

Narrows down the default search mode by using a word search. Asterisks can be used as placeholders.

ab Exact Match - Examples	
oup	Finds all containers **named exactly** "*oup*", a container named "group" will not be listed.
*gr**	Any container **starting** with "*gr*".
**up*	Any container **ending** with "*up*".
*lower*third*	Container **starts with** 'lower' and **ends with** 'third'.

.?* Regular Expressions

Provides the most precise results, however you have to follow the Regular Expression Syntax.

.?* Regular Expression - Examples	
.\d*	Finds all containers that end with a (single) digit.

Copyright © 2026 Vizrt

Regular Expression - Examples	
.\d.**	Searches for all containers containing at least one digit.
^[1-9]	All containers starting with a number.
[1-9]$	All containers ending with a number.
*^A.*B$*	Searches for all containers starting with 'A' and ending with 'B'.
^[bch]at$	All containers named 'bat', 'cat', or, 'hat'.
\d\d\d	Searches for all containers containing a three digit number.
gr(a|e)y	All containers named either "*gray*" or "*grey*".

Example: If you want to get rid of all the *group* containers and have a unique container names, search for *group*, press the select all button, then right-click on the first container and rename it to *container#*. All previous *group* container are then renamed to *Container1*, *Container2*, etc...

10.5.2 To Search for Plug-ins



The plug-in search allows to search for container properties like geometry and function plug-ins, classic materials and textures.
You can drag a texture to the plugin panel and it will also show their search locations within Material Definitions like PBR materials or shaders.

To use the plug-in search, switch the search mode to **plugin** and drag the property to search from the plug-ins panel, the asset view or from another container to the drop target:

If you drag any plugin, the search panel will automatically switch to plugin search mode.

Example: To replace all occurrences of a cube to a sphere, drag the cube to the search panel, select all and drag a sphere to one of the selected container.

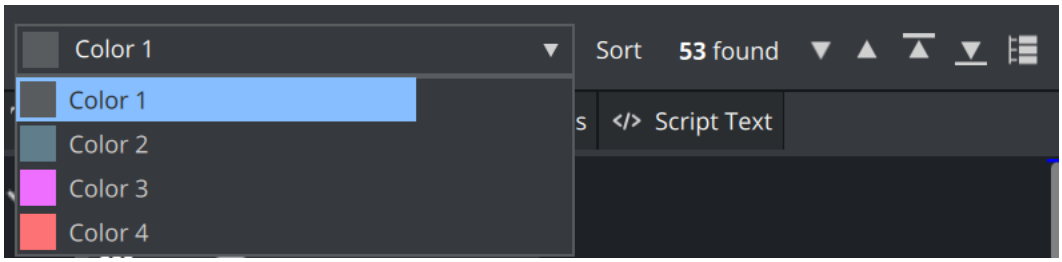
Known Limitations

- Searching for Viz Engine Fonts is currently not supported.

Copyright © 2026 Vizrt

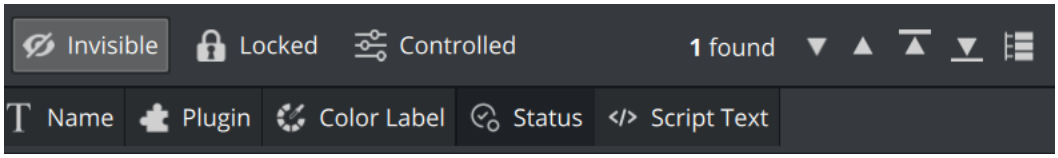
- Searching for Transformation types is not supported.

10.5.3 To Search for Color Labels



When switching to this search mode, you can select one of your label colors and the respective containers are highlighted.

10.5.4 To Search for Containers Status



This mode allows to search for a certain container status:

Status	Description
Invisible	Searches for all containers that are currently invisible. For example, if you want to set all containers as visible, search for all invisible ones, select all and enable them with a single click.
Locked	Searches for all containers having the lock state set.
Controlled	Will search for all containers with any kind of ControlPlugin assigned

10.5.5 To Search for Scripts Fragments

The search result contains all container scripts that contain the search term within the script code. The number of containers is shown as a search results, but not the number of occurrences within the script.

Copyright © 2026 Vizrt

Example: If you need to know which script writes into the console and disable the output, open the search window and search for "println". Open the Scripts dock, jump to the found containers and search/replace the code from "println" to "' println".

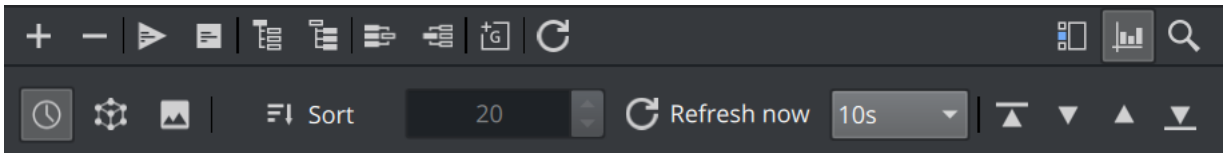
Known Limitations

- Viz Engine Shader code is not included in the search.
- The scene script is not included in the script!
- Regular Expressions are not supported.

Copyright © 2026 Vizrt

10.6 Sorting

If you are running into performance issues, it is required to strip down your scene and to optimize it. The new Scene tree implementation offers a profiling tool exactly for these tasks. You can find the profiler on the toolbar of the scene tree:

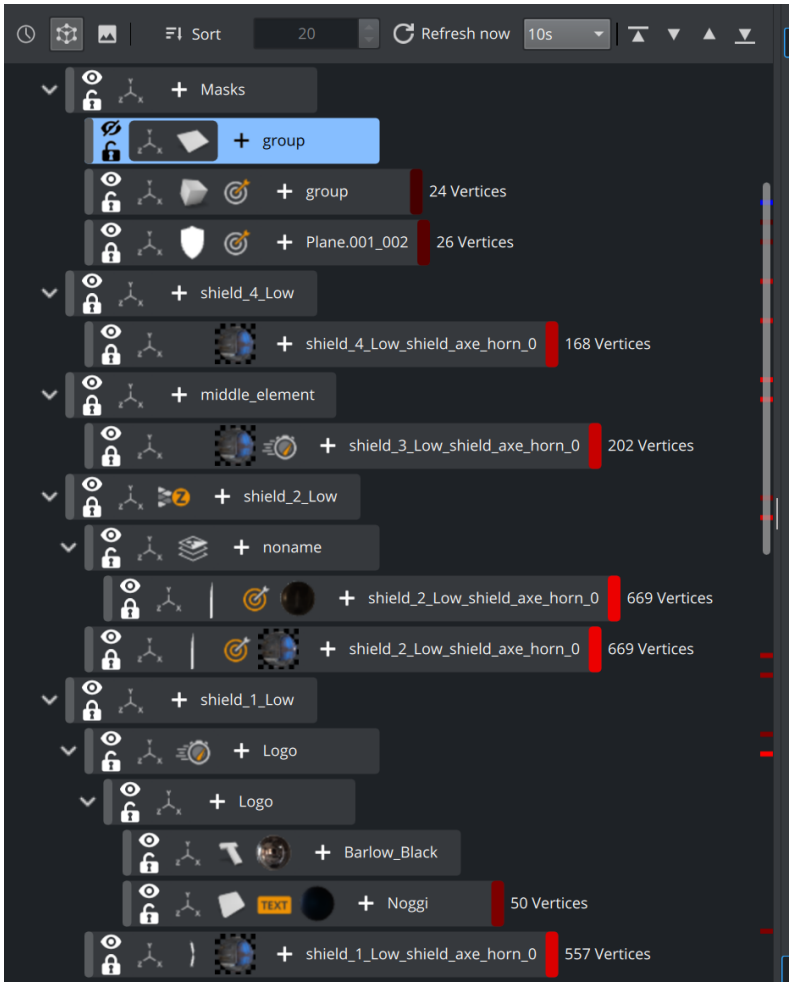


This opens the profiler tools, similar to the search tools. There are three different options available:

- Render Time
- Vertices
- Texture Size

The profiler acts like a heatmap on your scene. More red means heavier impact. The results are shown next to the affected containers. In addition, the hits are marked inside the scrollbar and the navigation buttons can be used to jump from the topmost affected container downwards.

Copyright © 2026 Vizrt



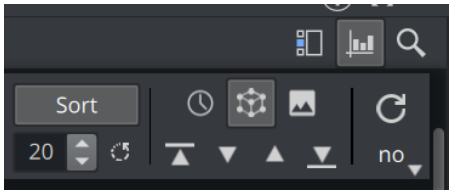
Tip: Make sure your containers are all expanded to make sure the scrollbar is able to show the entries.

Use the navigation buttons to jump from the top most to the lowest affected director:

Icon	Description
	Shows the most affected container.
	Jumps to next less affected container.
	Jumps to previous more affected container.
	Jumps to the less affected container.

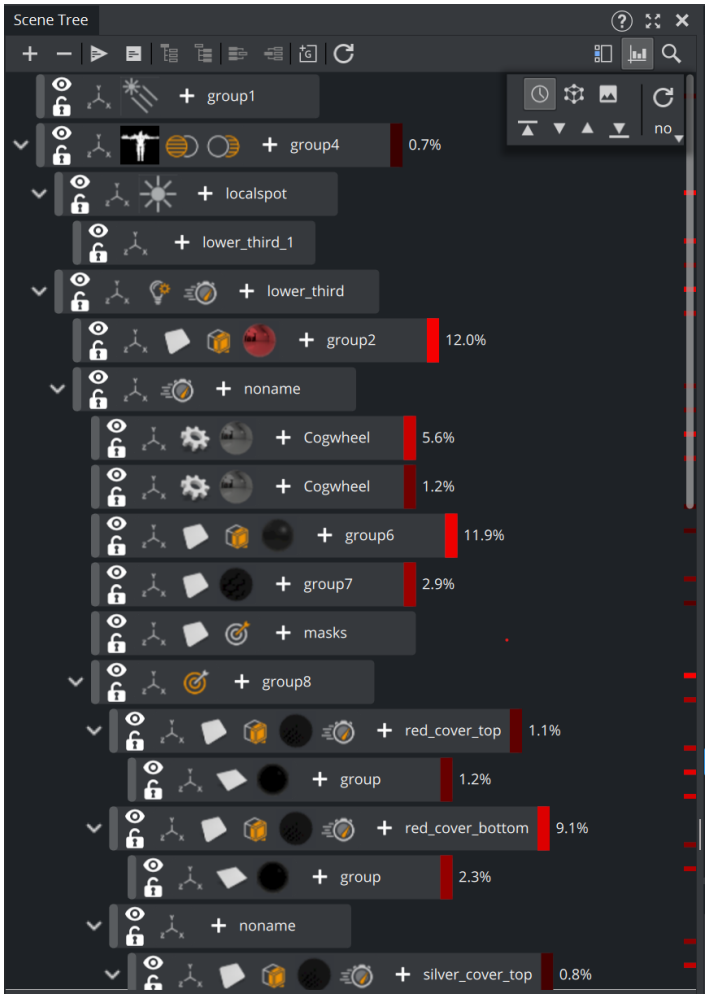
In addition to the normal Scene Tree view, you can also switch to a sorted list view by clicking on the **Sort** button.

Copyright © 2026 Vizrt



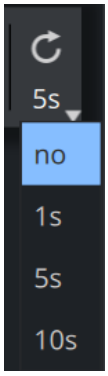
10.6.1 To Optimize Render Time

The first option is most probably the most important one. It analyzes the scene and returns those containers that require the most time to render.



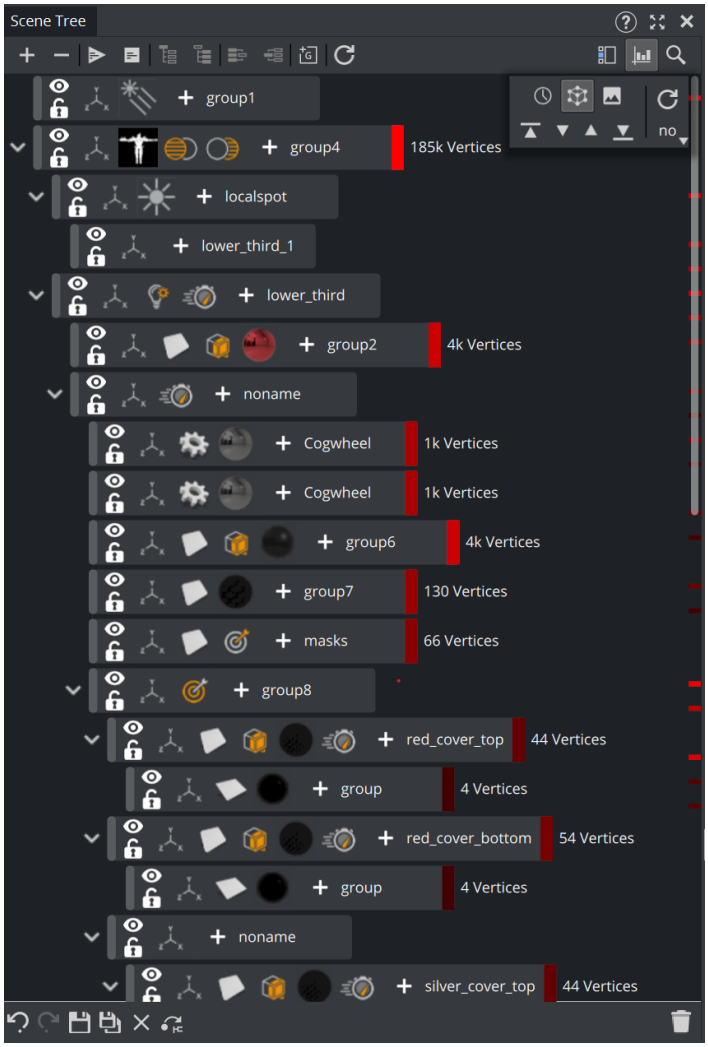
Please note that this is a snapshot of the current state at a given time. When showing a different view or during an animation these values can be completely different. Therefore, a time function has been added to choose the interval your measurements shall be taken. This can be manually (refresh icon) or on a given interval of one, five or ten seconds.

Copyright © 2026 Vizrt



10.6.2 To Optimize Your Geometries

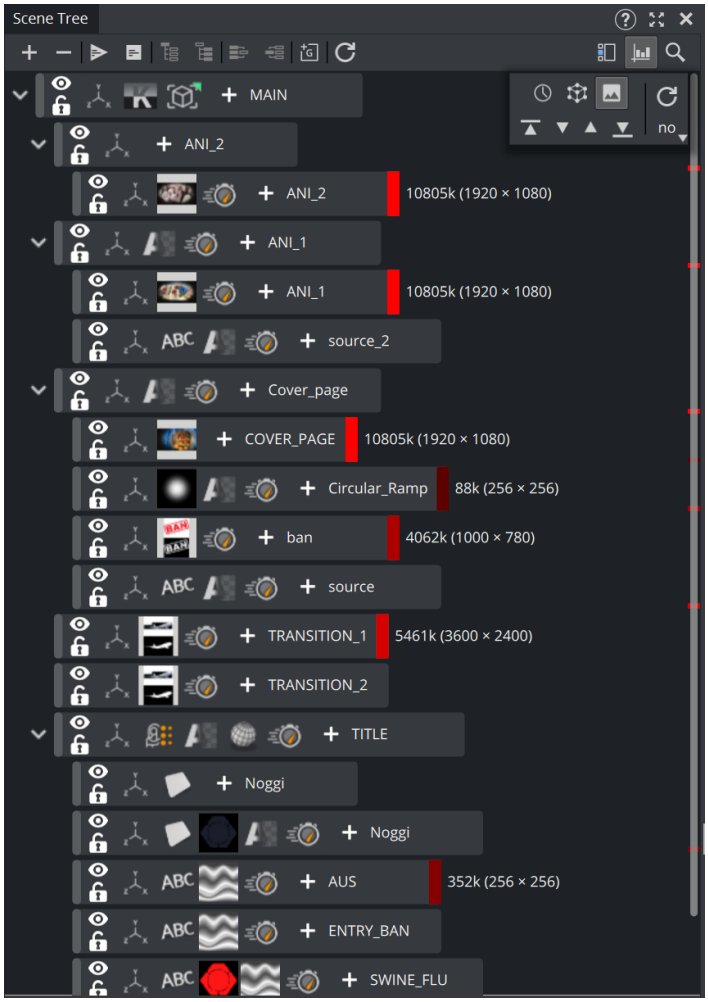
Even though modern graphics boards can process millions of triangles, it might be necessary to reduce the complexity of your models. To identify which geometry has a high vertices count, use this tool. Only containers with geometries applied are included. The number of vertices is shown next to the container:



10.6.3 To Reduce Texture Size

The available texture memory on GPUs is limited, even if they have Gigabytes of memory. Heavier textures also increase the loading time of your scene as they need to be transferred onto the graphics board. And keep in mind that they also need to be loaded into regular memory. Huge textures can have an performance impact if the driver of the GPU needs to start optimizing or freeing up available memory. To optimize your scene, search for the largest textures and check if they can be replaced with a smaller one.

Copyright © 2026 Vizrt



Information: This feature currently works only for scenes in the Classic Render Pipeline. Profiling Material Definitions will be included in one of the next versions of Viz Engine.

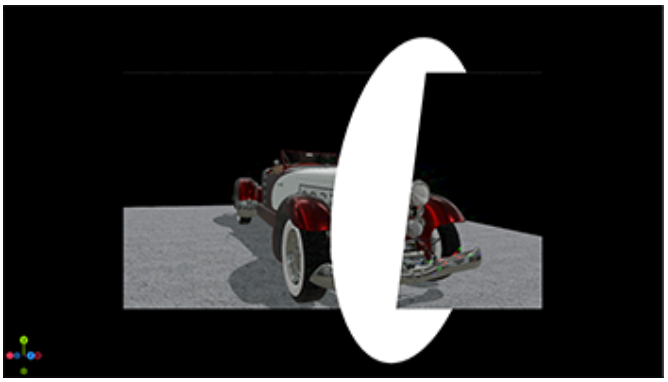
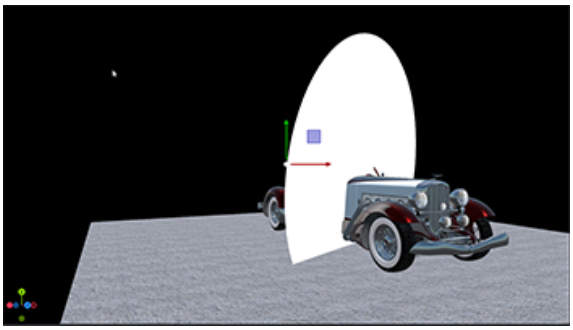
See Also

- Performance Bar

Copyright © 2026 Vizrt

10.7 Subscenes

Like GFX channels, you can attach a scene directly into your Scene Tree and its Scene Tree is automatically traversed and rendered like it is part of the main scene. Subscenes are integrated as part of the scene tree, similar to merged geometries, whereas GFX Channels or Viewport Tiles are rendered as flat 2D objects.

Scene as GFX Channel	Subscene
 Rendered as flat 2D texture.	 Rendered as part of the scene.

The attached scene inherits all global settings from the main scene, only its Scene Tree is merged into the main scene. There is no separate instance of the subscene for each container.

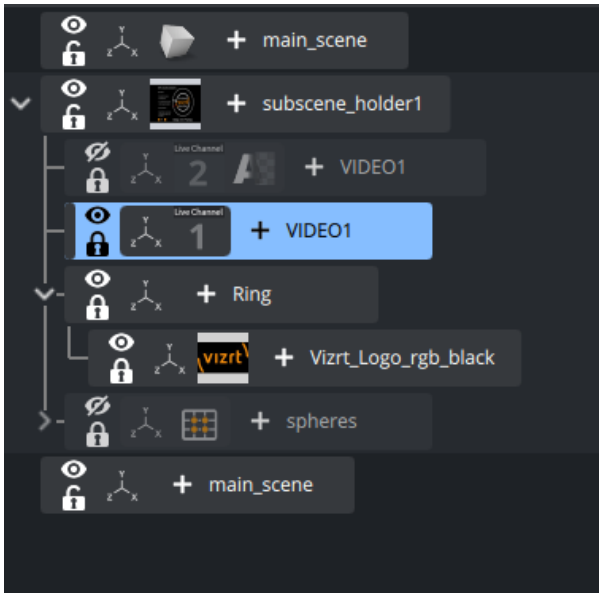
Subscenes can replace merged geometries used in Transition Logic scene. The Toggle plug-in automatically loads the scene as subscene if supported, otherwise the merged geometry is used.

Tip: The old behavior can be enforced by using `TransitionLogicUseMergedGeometries = 1`. This is only needed in some rare cases to keep compatibility with scenes that used for instance Scripting within the Transition Logic Tree structure.

Important: While GFX Channels render their content in the scene's pipeline, Subscenes are always rendered in the Main Render Pipeline (Classic or Viz Engine).

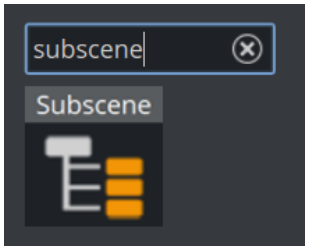
The containers belonging to the subscene are colored in the scene tree with a different background color.

Copyright © 2026 Vizrt



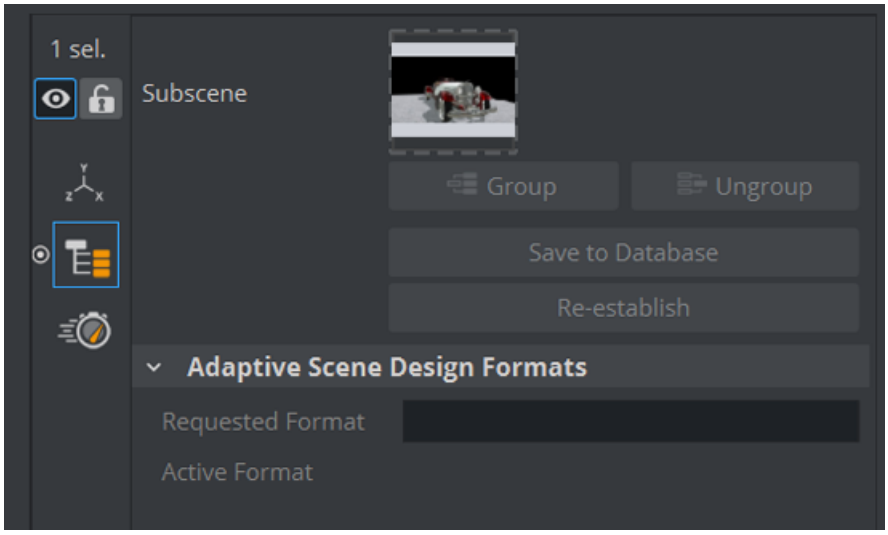
10.7.1 To Add a Subscene

Create new container, click on the Plus (**+**) icon and select **Subscene**.



This adds a subscene icon to your container. Click on the subscene icon and drag a scene from Asset View to the placeholder.

Copyright © 2026 Vizrt



The scene is now rendered as part of the Master Scene. The animation has been added to the stage.

Information: If you hover over the scene icon in the Scene Tree, the tooltip shows the *name* and *uuid* of the assigned scene.

Subscenes are the successor of merged geometries and are optimized for workflows (used in, for example, Transition Logic scenes). Instead of splitting up merged geometries, modify, merge and save again. Any existing scene can be brought into the current scene, modified and saved again with a single click.

Subscenes are indicated with a grey background within the scene tree for better identification.

10.7.2 Grouping and Ungrouping

Like merge geometries, the structure of a subscene can be expanded into the active scene by ungrouping the subscene. This makes the subscene's Scene Tree as well as the Stage Directors available inside the host scene and allows for modification on its structure. Click on the **Group** button to collapse the Scene Tree and the Stage Animation again.

Information: If you save an ungrouped subscene, all of its data becomes part of the original scene!

10.7.3 Save to Database

If you have modified your subscene and regrouped it, you can save all the changes in the subscene by clicking on the **Save to Database** button. This updates the original scene on Graphic Hub and reloads the data into the scene pool.

10.7.4 Re-Establish a Subscene

The **Re-Establish** button reloads the object from the scene pool. This reverts any modifications and replaces the current part of the Scene Tree with the original loaded from Graphic Hub.

Copyright © 2026 Vizrt

10.7.5 To Remove a Subscene

Drag the subscene to the trash bin.

10.7.6 To Select a Container of Subscene

By default, the whole Scene tree of the subscene is merged into the main scene. However, you can also specify a specific container which should be merged into the main scene:

```
' select container by level
MAIN_SCENE*TREE*/1*SUBSCENE SET <uuid> /1/1

' select container by name
MAIN_SCENE*TREE*/1*SUBSCENE SET <uuid> $ContainerName

' select container by property, search for container with ControlObject and use it as
root for the subscene handling
MAIN_SCENE*TREE*/1*SUBSCENE SET <uuid> %BUILT_IN*FUNCTION%ControlObject
```

Note: Subscenes added to the Scene Overview to have a better overview of the assigned scene.

10.7.7 Accessing "Main" Containers from within a Subscene

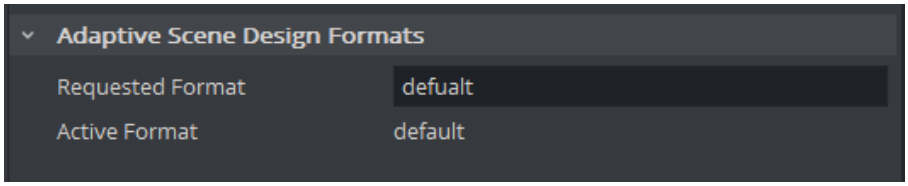
By default, subscenes keep their tree structure inside their scene from a scripting perspective. To access other scenes outside your subscene, you need to call the *MergedParentContainer* function.

```
dim topcontainer = this.MergedParentContainer
dim bContinue = True

Do
if topcontainer.MergedParentContainer<>Null then
   topcontainer = topcontainer.MergedParentContainer
Else
   bContinue = false
end If

loop while bContinue
```

10.7.8 Adaptive Scene Design Formats Support for Subscene

Subscenes can be set to render a certain format using Viz Artist or the command `MAIN_SCENE*TREE*/1*SUBSCENE*REQUESTED_FORMAT SET <name of format>`.

Copyright © 2026 Vizrt



Note 1: The requested format needs to be set before assigning the subscene.
Note 2: By default, the requested format is empty, which means the Subscene follows the main-scene format.
Note 3: If the requested format is set and it is not supported by the Subscene, then we fall back to the default format as active format, and the active format and requested format differ.
Note 4: BoxTransformation and Flexbox layouts Supports Subscene boundaries and considers Subscene containers in FlexBox calculations.

10.7.9 Shared Memory Support in Subscenes

Registering a *Scene.map* within a subscene also reflects in the main scene's shared memory map.

10.7.10 Workspace Items Support in Subscenes

The workspace items (Texture Renderers and Runtime Shaders) are cloned with subscene and accessible from workspace tab of Viz Artist.

10.7.11 Known Limitations

- Global Illumination is not supported in subscenes.
- Assigning a container within a subscene inside a script needs to be called after the Scene tree has been processed like after the OnInit() function.

Copyright © 2026 Vizrt

10.8 Layer Manager

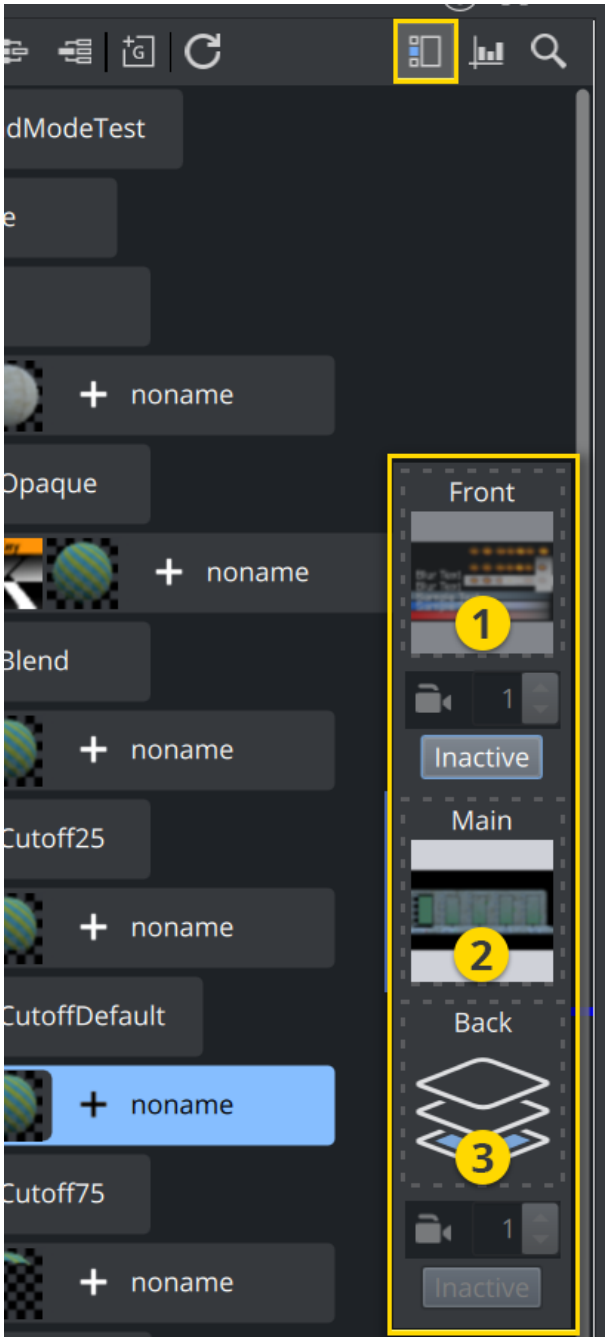
This section contains the following topics and procedures:

- Layer Support in the Viz Engine Render Pipeline
- To Add Scenes to the Front or Back Layer
- To Edit a Scene Used as a Front or Back Layer
- Self Layers

All scenes are designed to run in the Main layer (2) by default, but can be played out in the Front (1) and Back (3) layer as well. The Layer Manager can be used to test combinations of Scenes in the three layers.
A Scene set in the Main Layer can be edited in the Scene Tree and the Scene Editor panel. Scenes set in the Front or Back Layer can be edited by a double-click on the relevant Scene.

The Main Layer is accessible within the Scene Tree editor and indicates which layer is currently active: .

Copyright © 2026 Vizrt



The current active layer is also indicated with a blue line next to the drop fields for the different layers.

Picking in the Scene Editor is done on the loaded layer for Classic Render Pipeline Scenes. For Viz Engine Render Pipeline Scenes, picking only happens on the Main Layer. The Grid Tool-bar also works for the Main layer only in Viz Engine Render Pipeline.

Note: DVE Assets and Placeholders in Artist are only shown for the current layer for better visibility (for example, if a front layer shows a Fullscreen DVE asset, it is hidden when working in the main layer).

Copyright © 2026 Vizrt

10.8.1 Layer Support in the Viz Engine Render Pipeline

Support for layers has been introduced in Viz Engine Renderer Pipeline for compatibility reasons and to avoid the need to add GFX channels to the Main scene. However, there are some limitations:

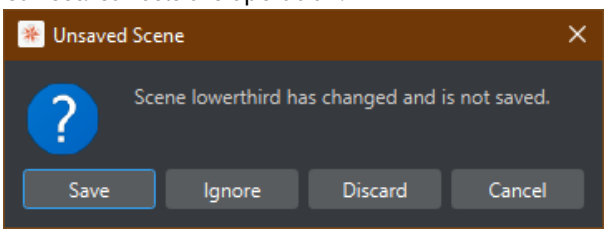
- Layers in the Viz Engine Render Pipeline are handled internally as GFX channels, therefore picking for Viz Engine Render Pipeline scenes is not supported.
- Grid functions are not supported in the Viz Engine Render Pipeline.
- If the Main Scene is a Classic Render Pipeline scene, Front and Back layers are also forced to be rendered in the Classic Render Pipeline.
- For best composition, the Render Sequence should be set to **Auto Key**.

10.8.2 To Add Scenes to the Front or Back Layer

Drag and drop scenes from the asset view onto the layer placeholder. To remove them, drop them to the trash bin. The visibility can be set by clicking ion the Active/Inactive button.

10.8.3 To Edit a Scene Used as a Front or Back Layer

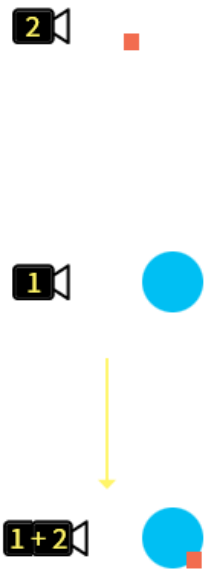
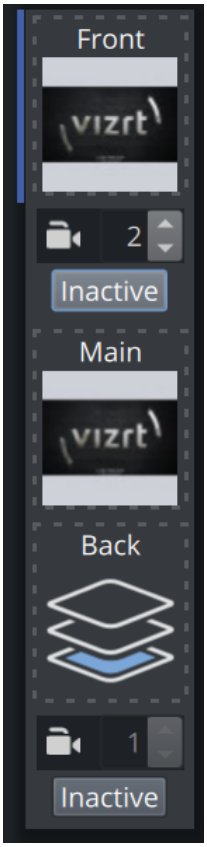
1. Double-click the Scene to be edited.
2. If another Scene in one of the three layers was modified, select an option from the message which opens:
   - **Save:** Saves any changes made in the last selected Scene.
   - **Ignore:** Switches to the layer without saving the last selected Scene.
   - **Discard:** Discards any changes made in the previously selected Scene and reloads the Scene when switching layers.
   - **Cancel:** Cancels the operation.



10.8.4 Self Layers

The same Scene can be set to the Front and Back layer, which creates what is known as self layer(s). This allows the Scenes to be linked, and the adjustment of the camera settings for the Front layer. When saving a Scene, any self layer settings are preserved. For example, some part of a scene should be rendered on top of the other parts. You can place your camera 2 in y.-position of 10.000 and put the part that should be rendered on top in view of this camera.

Copyright © 2026 Vizrt

		Camera 2 is placed in a height of y = 10.000 and rendering only the orange square The Main Camera1 is located at height y = 0 and rendering the blue circle. Loading the same scene again as self layered scene and having the front layer rendering camera 2, will results into a composition of Camera 1 and Camera 2 drawn on top of it.

Information: Self Layers are deprecated and only used for some old weather and maps scenes. They can be replaced with GFX channels.

See Also

- Scene Editor Buttons
- Working with the Scene Editor
- Time-line Editor

Copyright © 2026 Vizrt

11 Scene Management

This section details the management of Scenes, which includes the parameters of Scene Settings and the Scene Editor.

The Scene Editor has a set of useful functions that let the designer see how graphics look when keyed, seen from a different angle in Viz Virtual Studio (virtual sets), positioned on-screen, performance of the scene and more.

This section contains information on the following topics:

- Working with the Scene Editor
- Scene Overview
- Scene Settings
- Scene Editor
- Scene Editor Views

See Also:

- Working with Subscenes

Copyright © 2026 Vizrt

11.1 Working with the Scene Editor

Some transformation parameters of a Container or Containers in a Scene can be edited in the Scene Editor. Furthermore, you can switch some properties of a container On and Off.



This section contains information of the following topics:

- Scene Editor Functions
- Scene Editor Context Menu

11.1.1 Scene Editor Functions

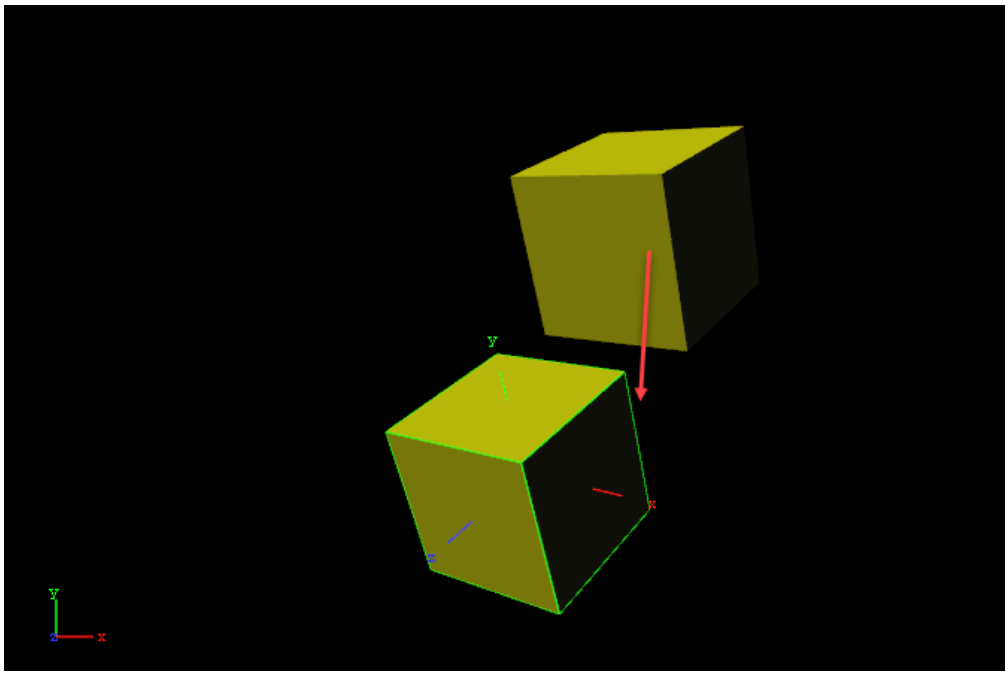
To Select/Deselect an Object

1. Left-click on an object, to select that object.
   - A bounding box shows which object has been selected (if the Bounding Box option is enabled, see Scene Editor Buttons).
   - The selected object's Container is highlighted in the Scene Tree
   - Click on the Transformation editor in the Properties panel to show to the transformation properties.
2. Press **SPACE** or click in an empty space to deselect an object.

> **Note:** This is done on a per pixel basis.

Copyright © 2026 Vizrt

To Copy an Object



1. Hold **CTRL** down and click and hold the **left mouse button.**
2. Drag and drop the object to the new location.

To Move an Object

1. Left-click on an object, to select that object.
2. Keep the left button pressed and move the object within the Scene Editor.
3. Release the left button.

To Move an Object on the Z Axis

1. Left-click on an object, to select that object.
2. Keep the left button pressed and press the right button. Hold both buttons down.
3. Move the mouse to the left or right to move selected object in the Z axis.

Multiple Objects Selected

When multiple objects are selected in the Scene Editor, the Transformation editor (in the Properties Panel) shows the editable properties for all selected objects.

All selected objects are included in one bounding box (if the Bounding Box option is enabled, see Scene Editor Buttons). The selected object's Containers are also highlighted in the Scene Tree.

Copyright © 2026 Vizrt

Note: When more than one object is selected, the Group coordinate system shows and a bounding box covers all selected objects. When Local and World is selected a bounding box for each selected object shows.

To Select Multiple Objects

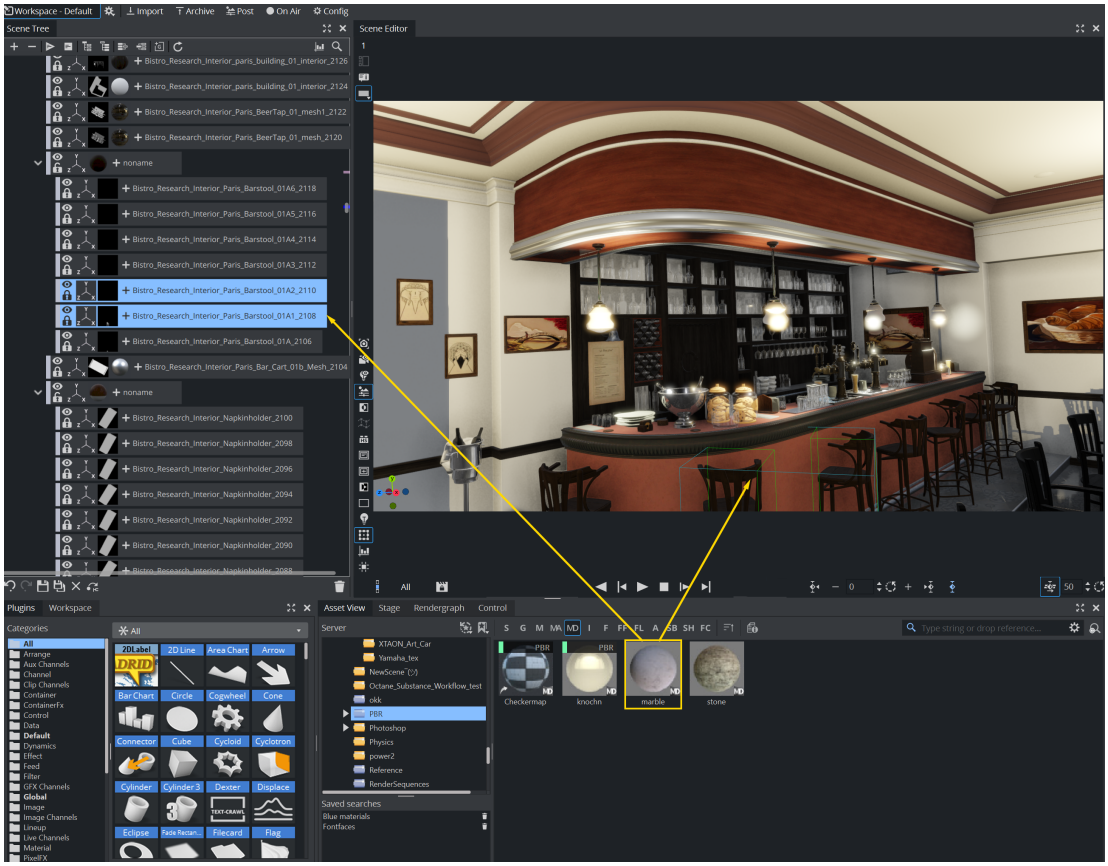
1. Hold the **CTRL** key and click on each object to select, as required, or
2. Click in an empty space in the Scene Editor and drag around the required objects to select.

To Deselect Multiple Objects

1. Press the **SPACE** bar, or
2. Click in an empty space on the Scene Editor.

Note: Toggle the **SPACE** bar to select or deselect multiple objects.

To Add Properties to Multi Selected Objects



1. Multi select the required objects.

Copyright © 2026 Vizrt

2. From the Server Panel select the required item.
3. Drag the item to one of the selected objects.
4. All selected objects take on the property.

To Move Multi Selected Objects

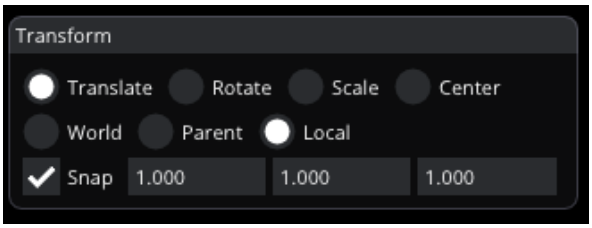
1. Select the required objects.
2. Left-click one of the objects to be moved and move the multi selected objects within the Scene Editor.
3. Release the left button.

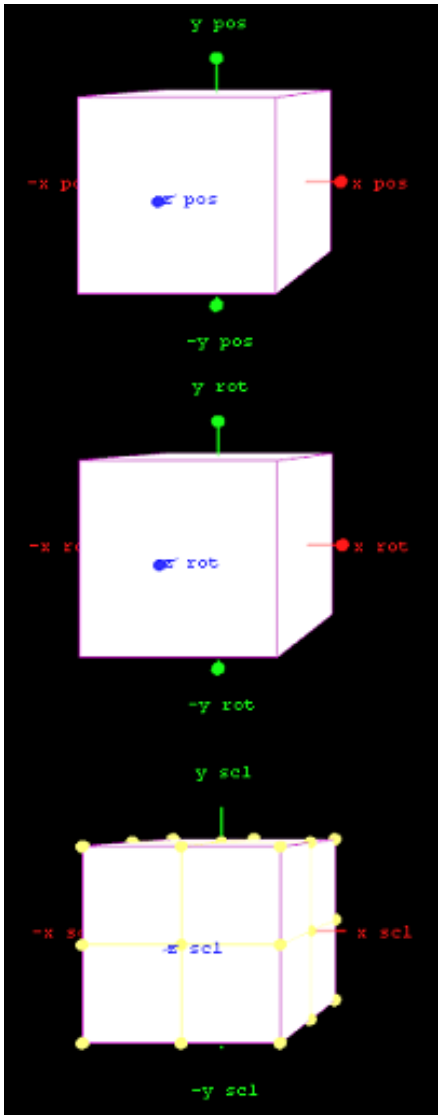
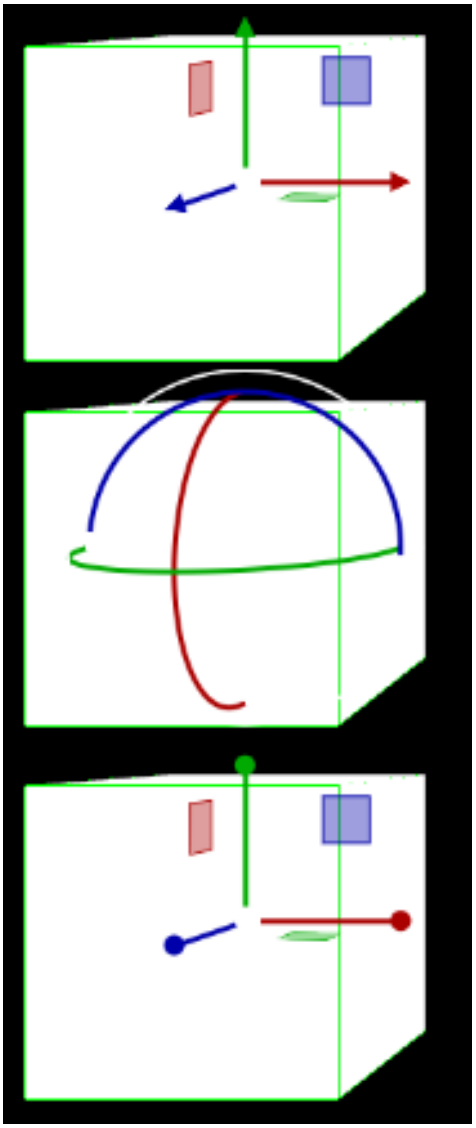
To Move Multi Selected Objects on the Z Axis

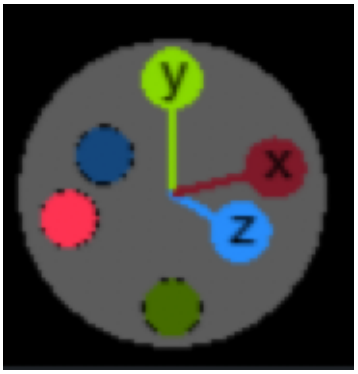
1. Select the required objects.
2. Left-click on one of the multi selected objects.
3. Keep the left button pressed and press the right button. Hold both buttons down.
4. Move the mouse to the left or right to move selected objects in the Z axis.

11.1.2 Scene Editor Context Menu

Using the Scene Editors Context menu differs between the Viz Engine Render Pipeline and the Classic Render Pipeline. Open the Context menu by right clicking the Scene editor.

	Classic Render Pipeline	Viz Engine Render Pipeline
Context Menu		 With the Viz Engine Render Pipeline, it is possible to also configure a snap size for each transform operation. This snap value controls the step size for editing.

Gizmos/ Handles		 Translation and scale gizmos provide plane-handles (rectangle handles) to lock one axis for editing. For scaling with the plane-handles, holding **CTRL** proportionally scales both axes.

Camera Control		 In the Viz Engine Render Pipeline, an additional camera control is available to quickly orbit around a selected object. To do so, select a geometry in the scene editor, hit **T** for target mode and use the *x*, *y* and *z* axes to orbit around the selected object.

Selected containers can be transformed inside the Scene Editor window with manipulation gizmos. The manipulation gizmos are visible when at least one container is selected.

You can switch between the different transform operations (Translate, Rotate, Scale, Center) by using the context menu. Switching the coordination space (Local, Parent, World) can also be done inside the context menu.

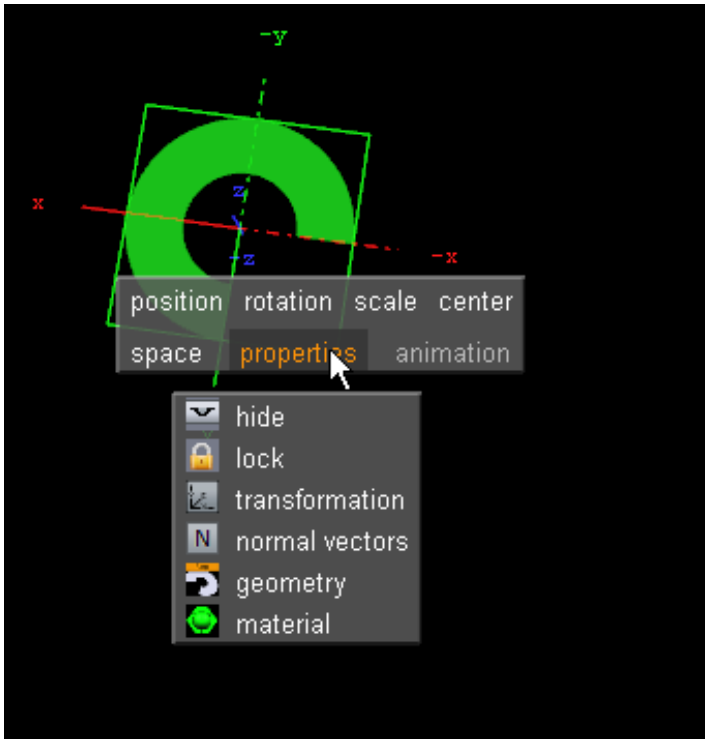
The context menu and the corresponding gizmos are modernized for the Viz Engine Renderer, but the concept is still the same. With the Viz Engine Renderer, it is possible to also configure a snap size for each transform operation. This snap value controls the step size for editing.

> ⚠ **Note:** In Viz Engine Renderer, transform editing is disabled in Scene Editor if any axis of the edited container has a scaling of `0`.

Scene Editor Menu in the Classic Render Pipeline

In the Scene Editor right-click and hold to open a context menu:

Copyright © 2026 Vizrt



Note: The context menu selection is available on a single Container or multiple Containers.

- **Position:** Switches into the position mode.
- **Rotation:** Switches into the rotation mode.
- **Scale:** Switches into scale mode.
- **Center:** Switches into center editing mode.
- **Space:** Switches between coordinate systems:
  - **Parent/Group:** Refers to changes that can be made to the Container or Containers in its Parent or Group coordinate system.
  - **Local:** Refers to changes that can be made to the container in its local coordinate system.
  - **World:** Refers to changes that can be made to containers in their world coordinate system.
- **Properties** (only applicable if a Container is selected):
  - **Hide/Show:** Hides or shows the Container in the Scene Editor.
  - **Lock/Unlock:** Locks or unlocks the container. When locked the Container cannot be modified in the Scene Editor.
  - **Transformation:** Opens the transformation editor for the Container.
  - **Normal vectors:** Toggles the display of the normal vectors at the bottom of the properties you find a list of all properties applied to the container with the possibility to switch them `On` and `Off`.
  - **Material:** Toggles diffuse color `on` (enlighted) or `off` (colored).
  - Any other properties of the container can set to on and off (except materials).
- **Animation:** Shows the animation path.

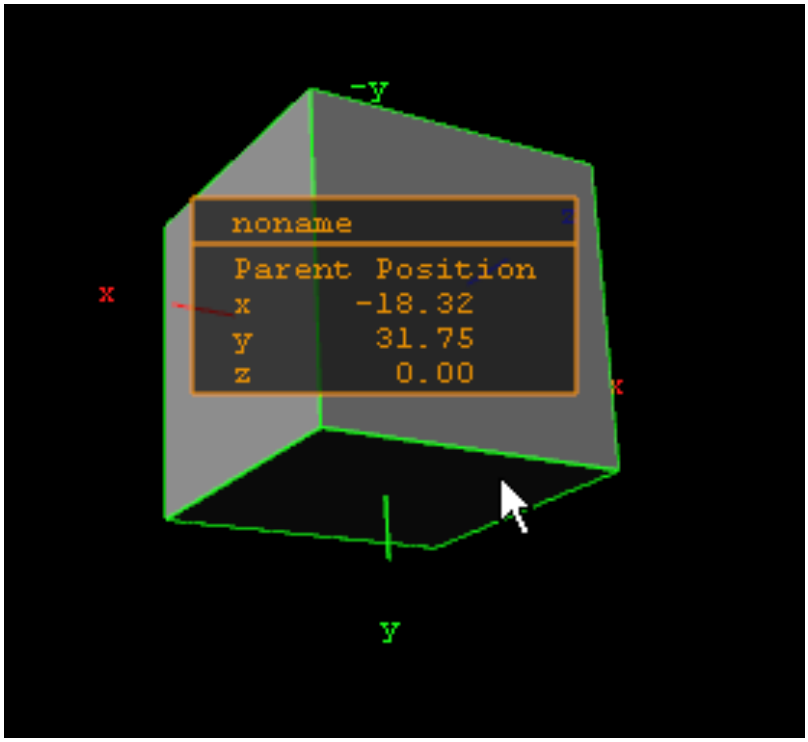
Copyright © 2026 Vizrt

Note: Multi selected objects: When more than one object is selected, Group shows and a bounding box covers all selected objects. When Local and World is selected a bounding box for each selected object shows.

To Select an Option from the Context Menu

1. Right-click and hold.
2. Move the cursor to the required option.
3. Release to select.

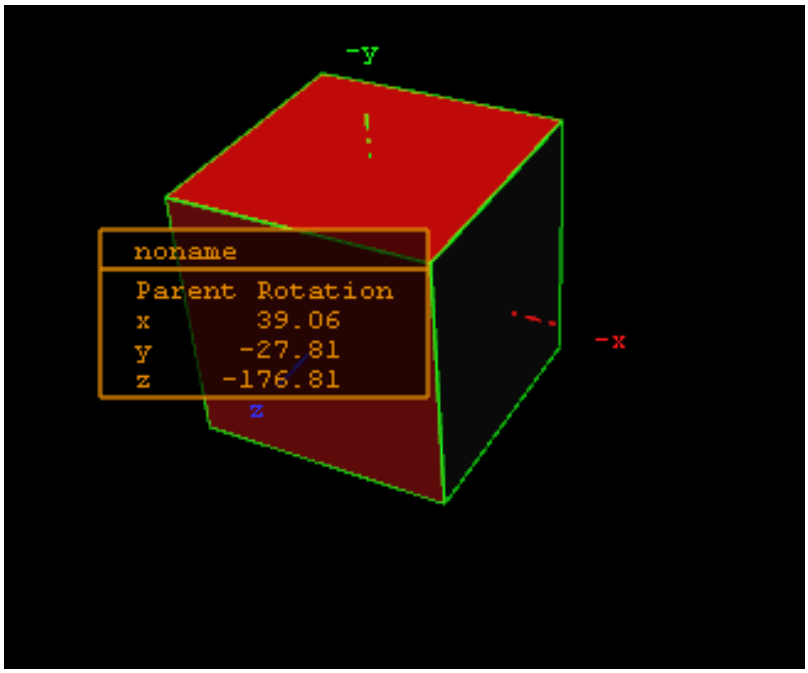
To Change the Position of an Object



1. Select an object.
2. Right-click the object and select **position** from the context menu.
3. Grab and drag a handle. The container follows the movement along the axes shown in the selected camera view. During this operation a status window is shown, giving you information about the position.

Copyright © 2026 Vizrt

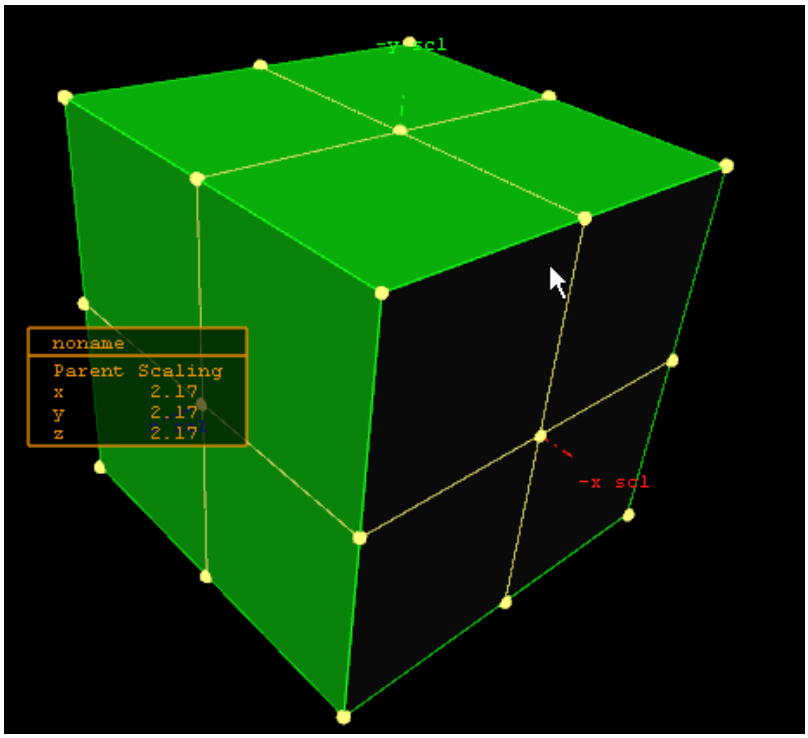
To Rotate an Object



1. Right-click to open the context menu.
2. Select **rotation**.
3. Drag the handles side ways to rotate on all the axes.

Copyright © 2026 Vizrt

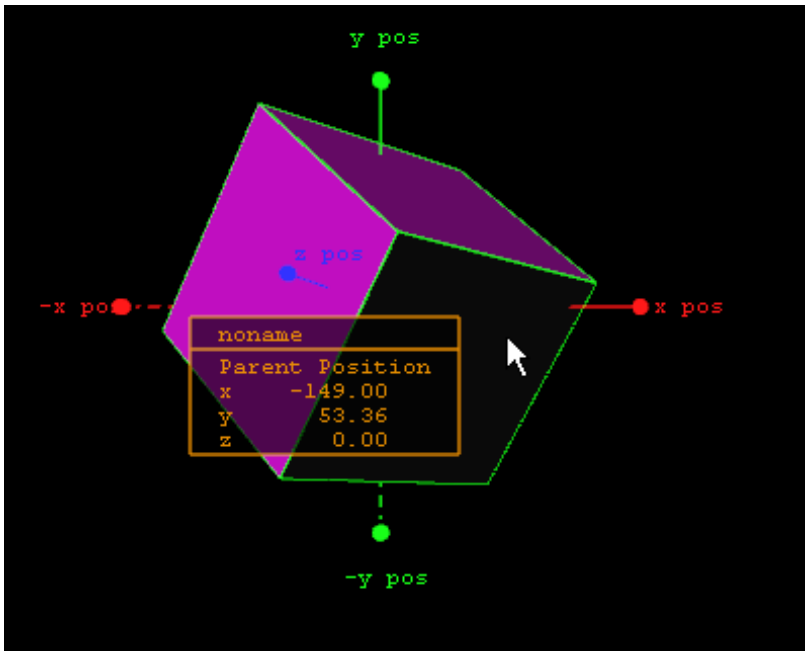
To Scale an Object



1. Right-click to open the context menu
2. Select **scale**.
3. Drag the handles to scale the object.

Copyright © 2026 Vizrt

To Move the Axis Center of an Object



1. Right-click to open the context menu.
2. Select **Center**. This shows the axis of the selected container with handles on their ends.
3. Grab one of the handles and change the center position by dragging them around the Scene Editor.

See Also

- Layer Manager
- Time-line Editor

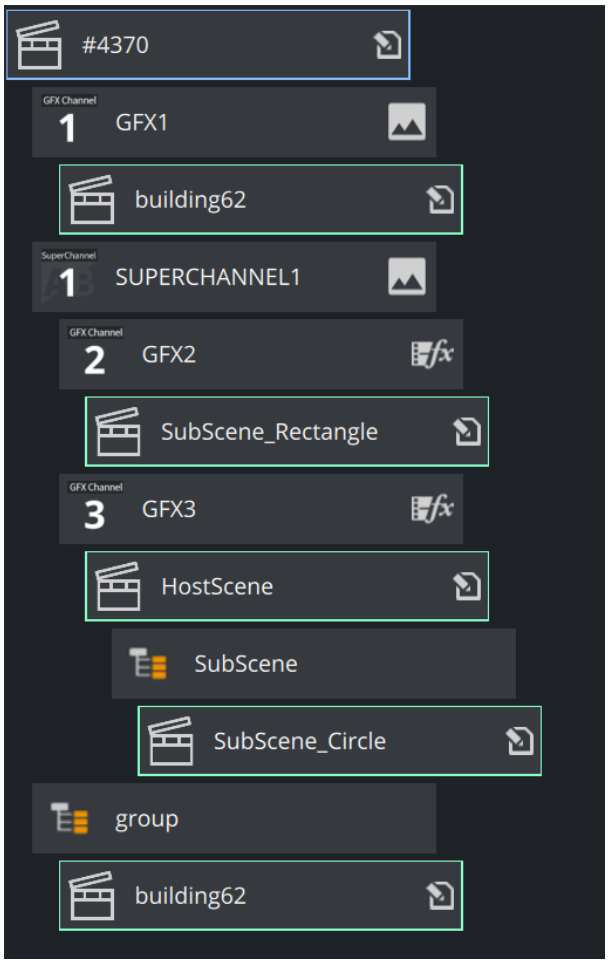
Copyright © 2026 Vizrt

11.2 Scene Overview

Scenes can get quite complex as they can utilize any type of media assets, GFX channels, Super Channels or Subscenes. The amount of nested media types sometimes makes it difficult to keep track of all required assets. Along with the new Scene Tree, a complete new overview view has been implemented. This dock gives you an overview about used assets and scenes and gives you the possibility to load and modify them on the fly without the need of leaving the main scene.

The view is available in your available docks. You can align them (for example, next to your scene tree).

11.2.1 Sample View



The image above shows a Main scene containing:

- GFX channel 1, with subscene *building62* assigned.
- A Super Channel with:
  - Scene A: With scene *Subscene_Rectangle* loaded.
  - Scene B: *Hostscene* with *SubScene_Circle* loaded as a subscene.

Copyright © 2026 Vizrt

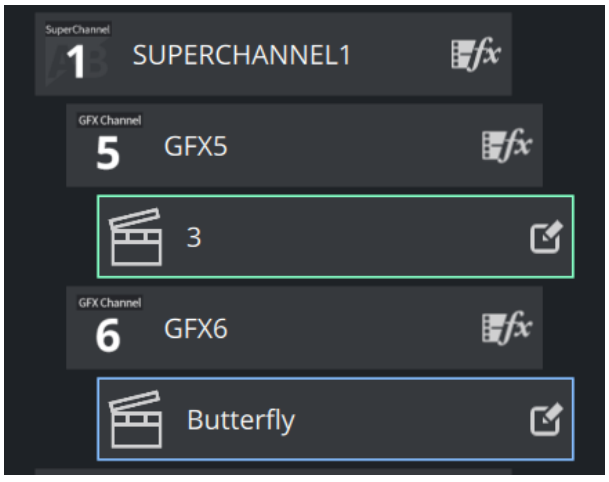
- *buildingsg62* loaded again inside a subscene.

This panel shows the currently loaded scenes and all media items used inside. It shows one of the following icons:

Icon	Description
	Represents a Scene.
	A GFX channel and its loaded content. If the GFX is shown as DVE or Texture, these are also displayed as icons: • Texture • DVE
	A Clip Channel.
	A Subscene.
	An Image Channel (always as DVE).
	An AUX Channel.

11.2.2 In-Scene Editing

The **Active Scene** is shown with a blue outline, whereas any other scene is shown with a green outline.



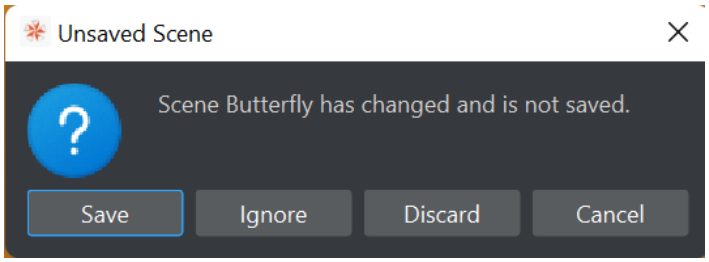
Items labeled with an edit icon and a green outline can be loaded and edited. **Double click** on one of these items and the scene is loaded into the editor (except for subscenes).

- The Scene Tree shows the content of the child asset.
- Stage shows the content of the child asset.

Copyright © 2026 Vizrt

- Scene Settings are disabled until you switch back to the Main scene again.
- The Render Editor still shows the content of the Main scene.

When switching scenes, the following dialog pops up:



- **Save:** Saves the current scene with all its changes.
- **Ignore:** Keeps the scene in memory, but doesn't save it.
- **Discard:** Does not save any changes, closes the current scene and switches to the new one.
- **Cancel:** Stops the current operation.

11.2.3 To Switch Back to the Main Scene

To switch back to the original main scene, the Scene Tree shows an additional button on top:



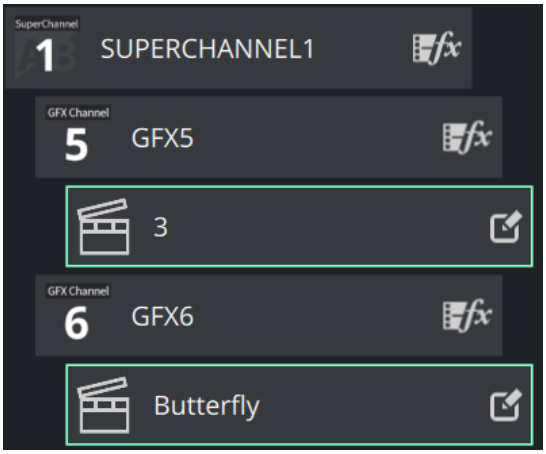
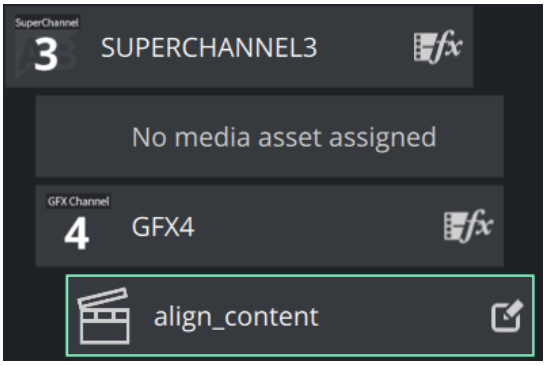
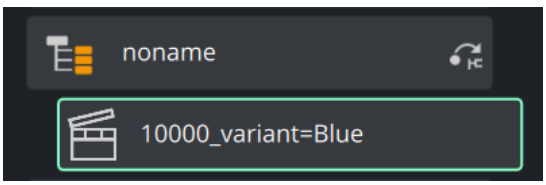
This takes you back to the original main scene. The top entry in the scene overview is selected again. A single click on any other entry in the scene overview jumps the linked media asset.

Note: Be sure that you are back in the main scene, when clicking on media assets in the scene overview. Otherwise, jumping to the entries in the Media Assets Manager will fail.

Tip: Sometimes it might be necessary to manually refresh the content of the Scene Overview. Use the refresh ( ) button on top to refresh.

Copyright © 2026 Vizrt

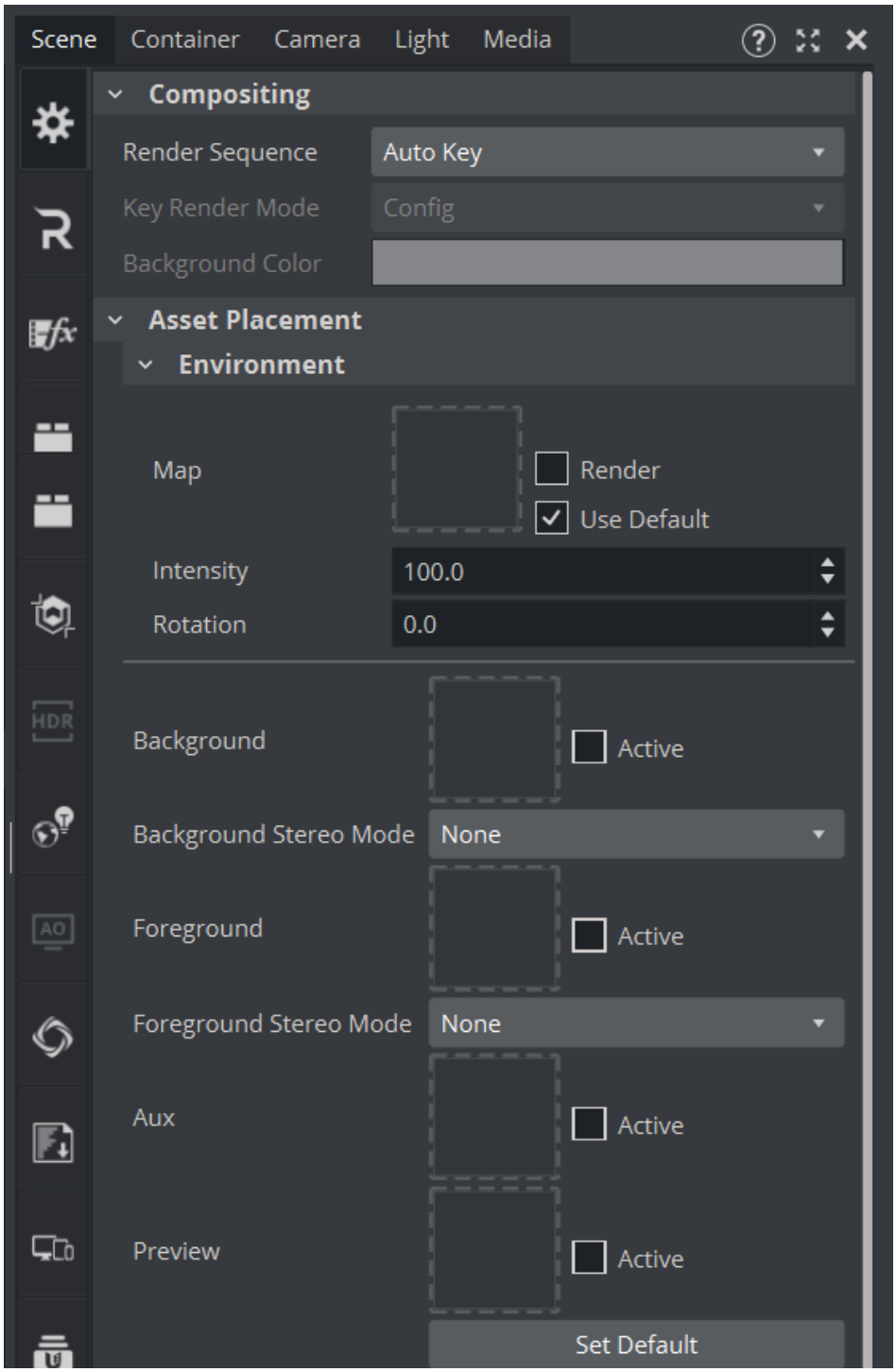
Examples

Sample	Content Shown
	Super Channel1 as DVE loaded with two scenes (named *3* and *butterfly*). The Super Channel utilizes internally GFX channels 5 and 6. As the two subchannels are indicated with a green border, they can be edited.
	Super Channel 4 as DVE effect. Subchannel A is loaded with Imagechannel2 and a Clip Channel on Subchannel B.
	Finally, Super Channel 3 as DVE, only Subchannel B is filled with a scene.
	Scene hosting a subscene. Click on the icon to jump to the container holding the subscene.

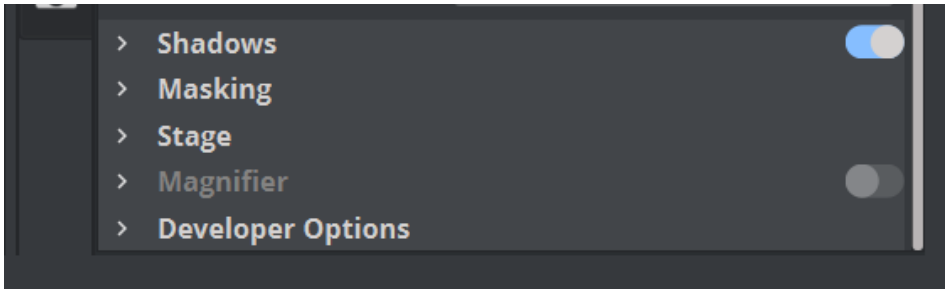
Copyright © 2026 Vizrt

11.3 Scene Settings

To open the Scene Settings editor, select the Scene Settings tab in the Property panel. The Scene Settings editor makes it possible to modify the global parameters for the scene. Some of the settings may be disabled as they depend on the chosen Render Pipeline.



Copyright © 2026 Vizrt



There are various types of scene settings. Click the tabs at the side of the Scene Settings editor to open the respective panels:

- Rendering Panel
- Global Settings Panel
- Global Media Assets Settings
- Plug-in Panel
- Clipper Panel
- High Dynamic Range Panel
- Real Time Global Illumination
- Classic SSAO
- Virtual Studio Panel
- Script Panel
- Adaptive Scene Design Formats
- Unreal Scene Settings

See Also

- Formats Settings
- Working with File Collections

Copyright © 2026 Vizrt

11.3.1 Rendering Panel

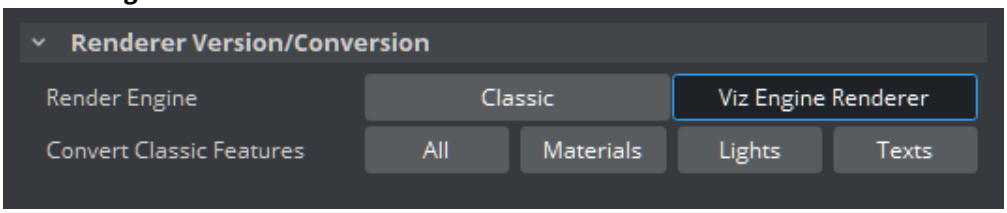
In the Rendering panel, basic rendering parameters can be modified.

This section contains information on the following topics and procedures:

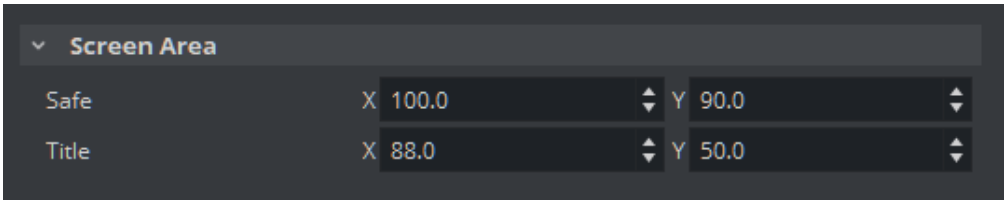
- Rendering Properties
- To Design Graphics with a User Defined Aspect Ratio

Rendering Properties

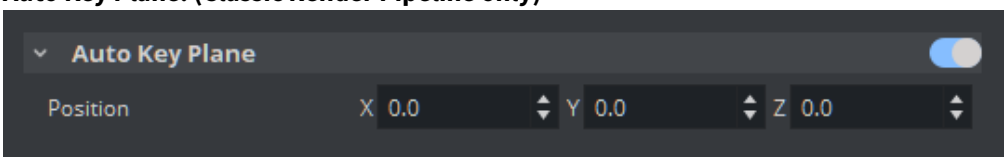
- **Render Engine and Scene Conversion**



  - **Render Engine**
    - **Classic:** Uses the Classic Render Pipeline (from Viz Engine 3).
    - **Viz Engine:** Uses the Viz Engine Render Pipeline (introduced in Viz Engine 4).
  - **Convert Classic Features:** Migration helper functionality that converts Classic Render Pipeline scenes to Viz Engine Render Pipeline scenes (with limitations in quality of conversion).
    - **All:** Converts all materials, lights and texts, from a classic scene.
    - **Materials:** Converts all Classic materials to Viz Engine Renderer material definitions.
    - **Lights:** Converts all Classic lights to Viz Engine Renderer (V4) lights.
    - **Texts:** Converts all Classic texts to Viz Engine Renderer texts.
- **Screen Area**



  - **Safe X/Y:** Sets the size of the Safe area X and Y values in percent of the screen resolution.
  - **Title X/Y:** Sets the size of the Title area X and Y values in percent of the screen resolution.
- **Auto Key Plane: (Classic Render Pipeline only)**

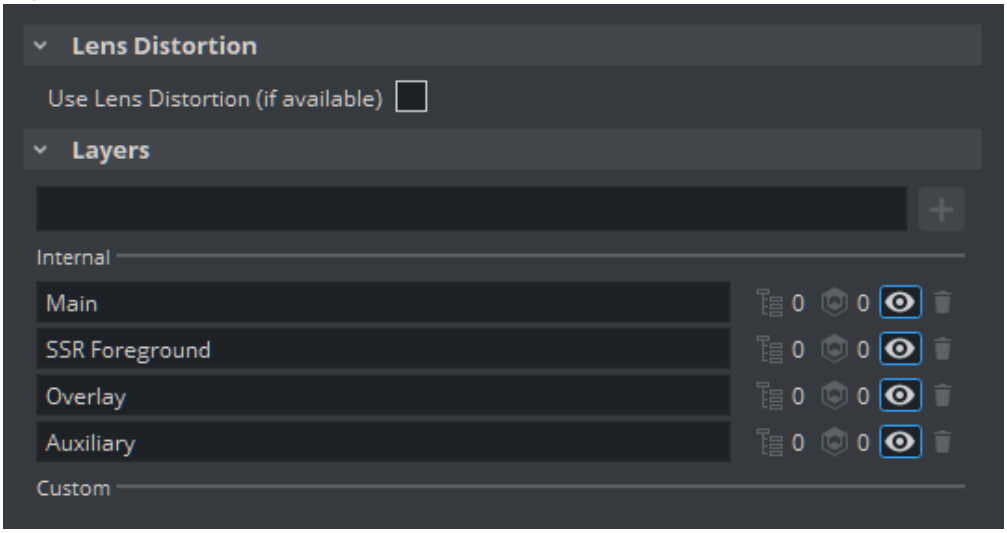


  Makes it possible to define the X, Y and Z Position, that serves as a threshold between objects with and without key. The position can be set according to a tracking device carried by a person on the set, or it can be controlled by an operator. The auto key position decides which objects or parts of objects in the Scene that are to be part of the key signal. Only objects positioned between the auto key position and the camera are rendered with the key signal. A typical setup for a Virtual Studio would be that the scene background is

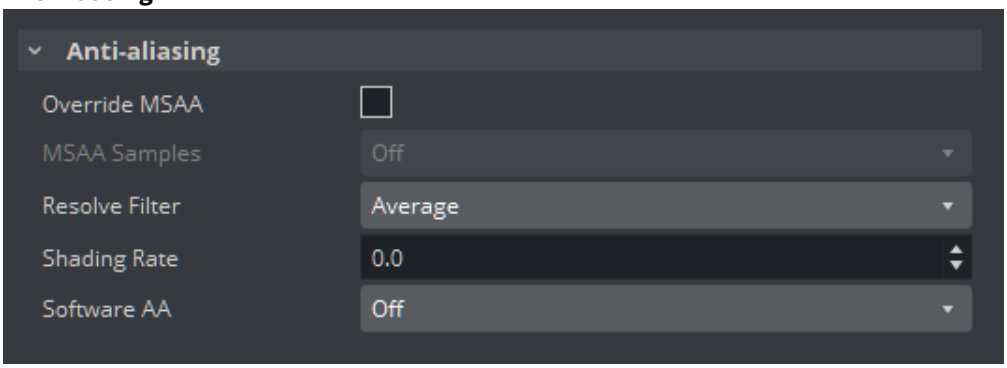
Copyright © 2026 Vizrt

put into a separate channel of the external Keyer. The objects of the background do not have a key function attached. The objects that are meant to be in front of the person on the set are placed in another channel/layer of the external Keyer and have a key function attached. If the person on the set moves against the camera, the auto key function disables the key function of those objects that now are placed behind the person, thereby moving them into the background layer of the external Keyer. To learn more about this, see the separate Viz Virtual Studio documentation.

- **Lens Distortion:** Enables the lens distortion for this scene, see Advanced Lens Distortion.
- **Layers**



  - **Internal:** Shows information about the two fixed layers **Main** and **SSR Foreground** and allows to set the visibility to be toggled.
  - **Custom:** Allows to create and modify new Render Layers. This is for future features. For more information about Render Layers see Screen Space Reflection.
- **Anti-Aliasing**



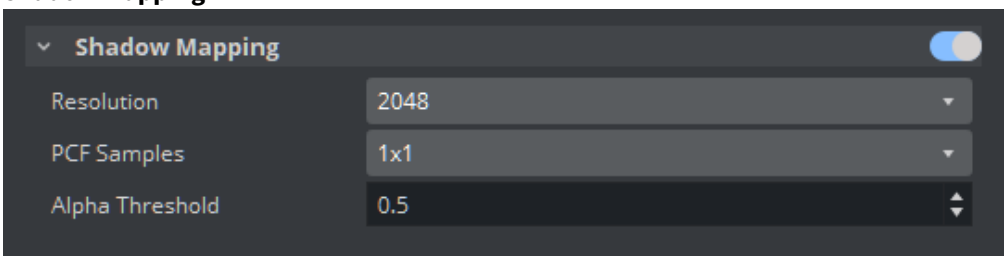
  - **Override MSAA:**
    - Disabled: Use MSAA settings from **Config** > **Render Options** > **Full Scene AA**
    - Enabled: Override MSAA settings from Config with **MSAA Samples**
  - **MSAA Samples:**
    - Number of samples used for MSAA.
    - **Off** to disable MSAA completely.
  - **Software AA**:
    - **Off:** Uses hardware antialiasing only.

Copyright © 2026 Vizrt

- **FXAA:** Removes pixel jiggering through an additional post processing effect based on the final rendering.
- **Temporal AA:** Uses multiple frames to calculate smooth edges (best quality).

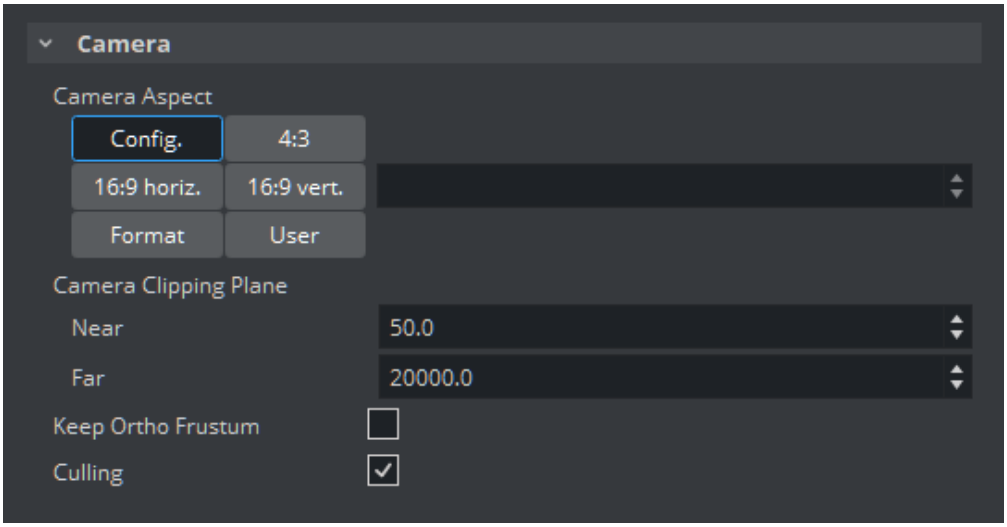
> **Note:** For more information about anti aliasing methods, please see To Improve Render Quality.

- **Shadow Mapping**



  - Sets the **resolution** of the shadow map (higher means more compute time).
  - **PCF Samples** defines the quality of the shadow map.
  - The **Alpha Threshold** defines at which opacity an object starts to cast a shadow.
- **Post Processing**
  - **Depth of Field:** Emulates the behavior of a real camera where objects further away from the focal plane show blurred and objects close to the focal plane show sharp. The respective parameters can be set in the camera settings under the Focus settings (see Camera Editor).
  - **Screen Space Reflections:** See Screen Space Reflection.
  - **DLSS:** See Deep Learning Super Sampling.
  - **Bloom:** See Blooming Effects.
  - **Screenspace Ambient Occlusion:** See Ambient Occlusion.
  - **Fog:** See Fog.
  - **Chromatic Aberration:** See Chromatic Aberration.
  - **Light Glares:** See Light Glares.
- **Color Space**
  - **Color Limits:** Defines the lower and upper limits.
  - **Gamma Correction:** Modifies the luminance of the final rendered image.
  - **Color Correction / Tonemapping:** See Tonemapping.
- **Camera**

Copyright © 2026 Vizrt

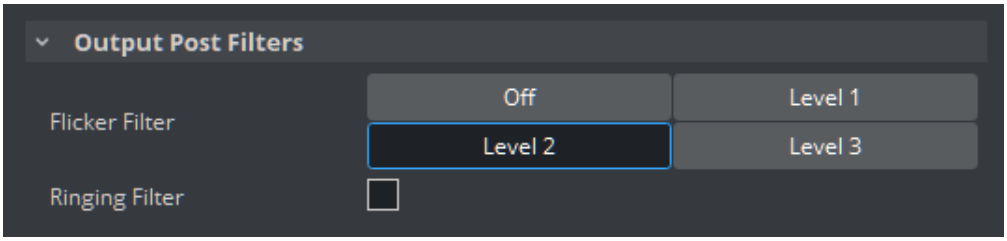


- **Camera Aspect:** Sets the camera aspect.
  - **Config:** Makes the camera aspect equal to the settings in the **Output Format** panel of the Viz Configuration (see the **Configuring Viz** section of the Viz Engine Administrator Guide).
  - **4:3, 16:9 Horizontal** and **16:9 Vertical:** Makes the camera aspect independent of the aspect for the rest of the scene.
  - **Format:** Uses the aspect ratio of the selected format (See Adaptive Scene Format)
  - **User:** Enables the user to set a user defined camera aspect ratio. When building a Scene with a customized aspect ratio for a **Video Wall**, set the user-defined camera aspect to the same value as the **Aspect** shown in the Viz Configuration (**Output Format > User Defined > Aspect** (which is set dynamically based on the other values). When building a Scene with a customized aspect ratio for other (for example, hand-held/mobile devices or online publishing), set the **Keep Editing Aspect** (**Render Options** in Viz Configuration) parameter to **Editor** as this does not affect on the On Air or Viz Engine output (see also To Design Graphics with a User Defined Aspect Ratio).
- **Camera Clipping Plane:** Sets the range of the virtual camera. `Near` sets the close range while `Far` defines the far range. Only containers within this range are rendered. This setting also affects the shadow rendering.
- **Keep Ortho Frustum:** Takes the original frustum stored and does not allow its re-computation after changes have been made to the camera parameters in the orthogonal mode when active.
- **Culling:** Saves performance for a scene when set to **Active**. Objects whose bounding box is outside the camera view are not rendered. This is of great importance to performance in large scenes and virtual studio. Normally Culling should be **Active**.

⚠ **Note:** The camera range is where the Zbuffer is within. So if Zbuffer problems arise, they may be solved by editing the camera clipping plane settings.

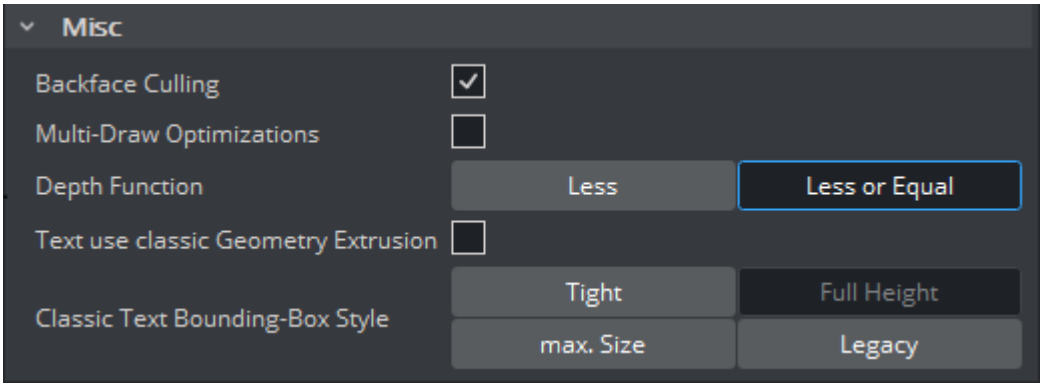
- **Output Post Filters**

Copyright © 2026 Vizrt



- **Flicker Filter:** Makes it possible to change the level of the flicker filter from `Off` up to `Level 3`. The flicker filter is used to remove/reduce flickering from a scene in case of interlaced video output. If a flicker filter is applied and the scene still flickers, try using the next level. Flicker filtering is off by default in HD resolution. Set `allow_flickerfilter_hd = 1` in the config file to enable it for HD resolutions, too.
- **Ringing Filter:** Reduces the ringing artifacts on the SDI output and applies only to SD formats. The ringing filter can be set to Off (see **Render Options** (set **Force Ringing Filter Off** to **Active**) under Viz Configuration in the Viz Engine Administrator Guide). The default Ringing Filter is located in **Scene Default Values** in Viz Configuration).

- **Misc**



- **Backface Culling:** Draws geometry usually not visible to the camera. This can be enabled for certain objects using the Expert plug-in.
- **Multi-Draw Optimizations** (Viz Engine Render Pipeline only): Allows to render multiple geometries with only one call to the GPU (for example, for Text rendering)
- **Depth Function:** Defines the depth function calculation.
- **Text use classic Geometry Extrusion:** Sets whether to use the modern text (instead of the now obsolete classic style of extrusion for text). This option is for compatibility reasons, and is not checked by default for scenes created from Viz Engine 5.4.0 or later. The newer style generally has better performance, especially with animation.
- **Classic Text Bounding-Box Style:** Defines how the bounding boxes for classic texts are calculated. This is for compatibility reasons only and usually does not need to be changed.
  - **Tight:** Draws the bounding box tightly around the text.
  - **Full Height:** Default and recommended setting.
  - **max. Size:** Only required for imported old scenes from 2.x.
  - **Legacy:** Uses the same algorithm as in 2.x versions (not in use anymore).

Copyright © 2026 Vizrt

To Design Graphics with a User Defined Aspect Ratio

1. Start Viz Artist.
2. Click **Config** (or press **F11**).
3. If you plan to render graphics in many different aspects (for example, still images for online publishing) you should select the **Render Options** section and set **Keep Editing Aspect** to **Editor**. This makes sure your graphics are rendered according to the scene's configured *camera aspect* (and not be affected by the Output Format aspect).
4. In the **Output Format** section, set the format that best fits the screen resolution and aspect (for example, 16:9).
    - Viz Artist is resolution independent and its workspace is optimized for designing in 16:9 and 4:3 aspects.
    - When designing graphics with a user defined aspect, only set the *Output Format* to size the aspect of your Scene Editor and adjust the *camera aspect* further to define it according to your scene's required and final aspect.
5. Restart Viz Artist.
6. From the Properties Panel click the Scene Settings tab.
7. Under the Rendering Panel tab set **Camera Aspect** to **User**.
8. Enter your aspect ratio in the field beside the User field. To calculate your aspect ratio simply divide width with height (for example, 1600:900=1.778).

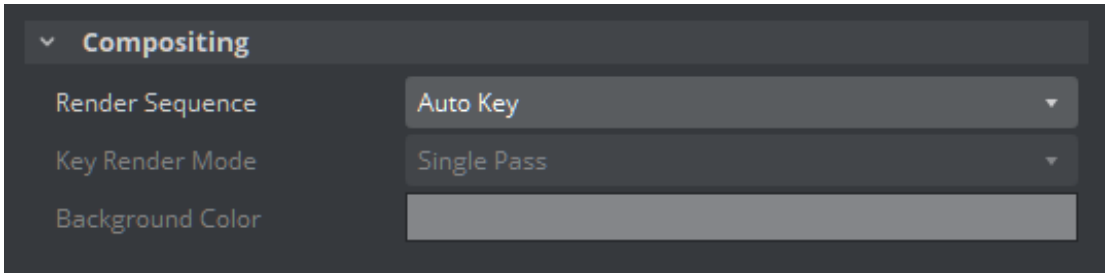
See Also

- Designing for Adaptive Storytelling

Copyright © 2026 Vizrt

Viz Libero and Viz Arena Render Sequences

This section details the render sequences for the Viz Libero, Viz Arena and Viz Eclipse (see Global Settings Panel).



This section contains the following topics:

- Libero Render Sequence
  - Classic Render Pipeline
  - Viz Engine Render Pipeline
- Libero Fill and Key Render Sequence
- Arena Render Sequence (Mechanical Based Tracking)
- Arena Image-Based Render Sequence (Image Based Tracking)
  - Classic Render Pipeline
  - Viz Engine Render Pipeline
- Arena Image-Based Fill & Key Render Sequence
- Rendering Support for Dirty Video Feeds with Viz Arena
  - To Add Image Masks for Dirty Feeds for Viz Arena
- Eclipse Render Sequence (based on the Viz Arena Image-based Render Sequence)

Libero Render Sequence

The mechanisms in Classic Render Pipeline and Viz Engine Render Pipeline are different:

Classic Render Pipeline

1. **Back Layer:** If a scene is present, the background image (live video) is rendered, and only containers with a linear key are rendered on top, with optional lens distortion.
2. **Main Layer:** The following sequence is performed with optional lens distortion (unless specified).
   a. Background image (live video) is rendered without lens distortion.
   b. Containers with the Key plug-in, with **Combine with BG chroma key** set to `On` and **Render Mode** set to `Add`, are keyed with the background video. The final key of these items results from the key information of the background video, provided by Viz Libero, and the key information of the Container.
   c. Containers with standard keying are rendered and linearly keyed over.
3. **Front Layer:** If a scene is present, only containers with a linear key are rendered on top, without lens distortion.

Copyright © 2026 Vizrt

Viz Engine Render Pipeline

1. **Back Layer:** If a scene is present, the background image (live video) is rendered, and only containers with a linear key are rendered on top, with optional lens distortion.
2. **Main Layer:** The following sequence is performed with optional lens distortion (unless specified).
    a. Background image (live video) is rendered without lens distortion.
    b. Containers with Key and Layer plug-ins, such that **Main** layer is checked, are keyed with the background video. Chroma keying information is used to superimpose these Containers.
    c. Containers with Key and Layer plug-ins, such that **Auxiliary** layer is checked, are rendered and tested with depth buffer provided by Viz Libero, and also linearly keyed over.
    d. Containers with Key and Layer plug-ins, such that **Overlay** layer is checked, are rendered and linearly keyed over.
3. **Front Layer:** If a scene is present, only containers with a linear key are rendered on top, without lens distortion.

Note that only **Render Mode** set to `Blend (Premultiplied Alpha)` is supported in Key plug-in.

See Layer for more details on configuring containers with Key and Layer plug-ins.

> In the case of Viz Engine Render Pipeline, the depth buffer shared by Viz Libero through the same live channel as the scene background must be explicitly set up in Unreal Scene Settings. This is done by assigning such live channel to the **Depth** slot and unchecking **Use Custom Clipping** to receive the clipping values from Viz Libero.

Libero Fill and Key Render Sequence

Same as Viz Libero, but with a separate Fill and Key output.

> To use a linear keyer, set Fill Mode to `Unshaped` in Viz Configuration. See Render Options - Viz Artist and Viz Engine - Vizrt Confluence.

Arena Render Sequence (Mechanical Based Tracking)

1. **Back Layer:** The keying information for Containers in the Main Scene is created. Typically objects with the Key plug-in on them are used for this:
    a. Background image without alpha and without lens distortion.
    b. Containers which contain standard key information either as linear key or generated through chroma keying.
2. **Main Layer**
    a. Containers with the Key plug-in, with **Depth information only** set to `On`, are rendered. Only the depth information is rendered and used to cutout "real" objects from chroma keyed virtual objects.
    b. Containers with the Key plug-in, with **Combine with BG chroma key** set to `On`, are rendered. Depth and chroma keying information are used to superimpose these Containers.
    c. Containers with the Key plug-in, with **Depth information only** set to `On`, are rendered.
    The **depth** information is used for Containers which should be keyed linearly.
    d. Containers with a linear key are rendered with lens distortion and linearly keyed over.
3. **Front Layer**

Copyright © 2026 Vizrt

a. Background image is rendered. This only makes sense if the last visible layer for SDI output is set to Main Layer. If this is so, a different video source can be shown on the Viz Engine preview to prepare the next item(s), which goes On Air.
b. Containers with a linear key are rendered with lens distortion and keyed over.

Arena Image-Based Render Sequence (Image Based Tracking)

The mechanisms in Classic Render Pipeline and Viz Engine Render Pipeline are different:

Classic Render Pipeline

1. **Back Layer:** Nothing is rendered from the Back Layer Scene.
2. **Main Layer:** The following sequence is performed with lens distortion (unless specified).
    a. Background images are rendered without lens distortion.
    b. If a foreground image (Stencil Mask) is present, but not active, only the alpha information is used to mask out parts of the containers in the Main Scene. Alpha is rendered without lens distortion.
    c. Containers with the Key plug-in, with **Combine with BG chroma key** set to `On` and **Render Mode** set to `Add`, are keyed with the background video. Chroma keying information is used to superimpose these Containers.
    d. Containers with a linear key are rendered and linearly keyed over.
3. **Front Layer:** If a scene is present, only containers with a linear key are rendered on top, without lens distortion.

The final image is combined using linear keying and a final combined Fill output is generated.

Viz Engine Render Pipeline

1. **Back Layer:** Nothing is rendered from the Back Layer Scene.
2. **Main Layer:** The following sequence is performed with lens distortion (unless specified).
    a. Background images are rendered without lens distortion.
    b. If a foreground image (Stencil Mask) is present, but not active, only the alpha information is used to mask out parts of the containers in the Main Scene. Alpha is rendered without lens distortion.
    c. Containers with Key and Layer plug-ins, such that **Main** layer is checked, are keyed with the background video. Chroma keying information is used to superimpose these Containers.
    d. Containers with Key and Layer plug-ins, such that **Overlay** layer is checked, are rendered with linearly keyed over.
3. **Front Layer:** If a scene is present, only containers with a linear key are rendered on top, without lens distortion.

Note that only **Render Mode** set to `Blend (Premultiplied Alpha)` is supported in Key plug-in, and the Classic Render Pipeline's **Arena Image-Based** and **Arena Image-Based Fill & Key** sequences are combined, resulting in this single sequence.

- Fill output operates as created by the Classic Render Pipeline's Arena Image-Based.
- Key output operates as created by the Classic Render Pipeline's Arena Image-Based Fill & Key.

See Layer for more details on configuring containers with Key and Layer plug-ins.



To use a linear keyer, set Fill Mode to `Unshaped` in Viz Configuration. See Render Options - Viz Artist and Viz Engine - Vizrt Confluence.

Copyright © 2026 Vizrt

Arena Image-Based Fill & Key Render Sequence

Same as Arena Image-Based (Classic), but with separate Fill and Key output.

Rendering Support for Dirty Video Feeds with Viz Arena

A dirty video input signal is a video containing graphical items such as logos, banners, clocks, etc., that are overlaid on top of the video signal before being sent to the Viz Engine. To avoid disturbances in the viewing experience, Viz Engine has to render its graphical items *below* the preexisting overlays. If supplied with a dirty video input signal, Viz Engine can suppress rendering of graphical items within certain image regions. To do this, alpha image masks need to be sent to Viz Engine, defining in which regions of the image graphical items must be rendered, and in which rendering shall be suppressed.

In other words, areas with existing graphics from the broadcast need to be excluded to avoid rendering new graphics on top of them. To achieve this, an alpha image needs to be loaded as a foreground image in the scene. The area with alpha information is then excluded from rendering, working as an inverse mask. Viz Engine then incorporates the currently activated image mask(s) for rendering of graphical items. In image regions where the value of the image mask is below 255, the alpha of all rendered graphics is adjusted accordingly. Applying the image mask also works when the render scale is different than 1. Independent of the currently selected render scale, the size of the image mask always refers to the original video resolution.

IMPORTANT! The foreground image needs to be set to inactive to ensure only the key is applied.

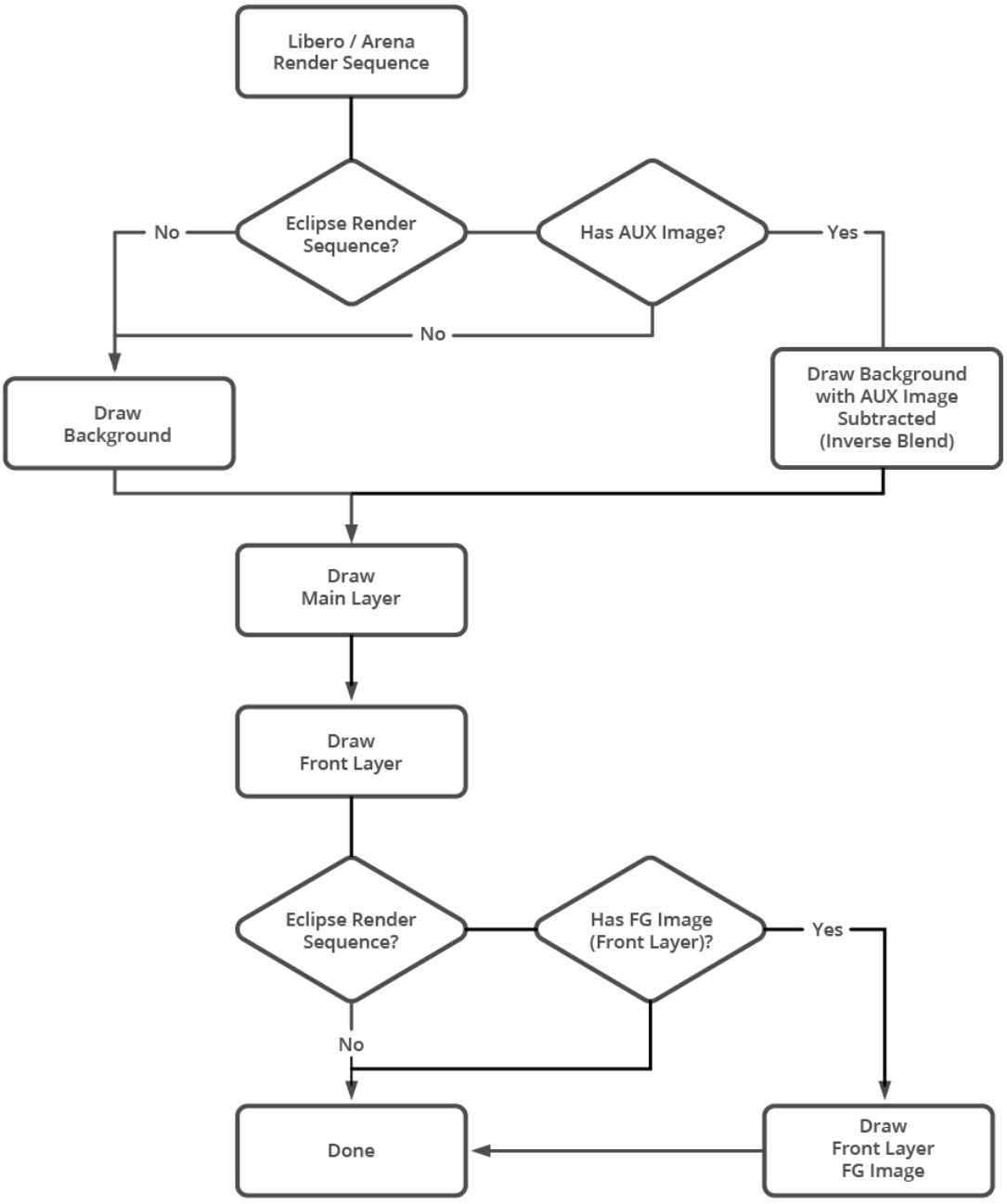
To Add Image Masks for Dirty Feeds for Viz Arena

1. An alpha image must be created, with the exact width and height of the currently used resolution. For example, for 1080 resolutions, the size is 1920x1080 pixels. As images with a lower resolution are stretched, possibly giving unexpected or unwanted results, make sure to always create the alpha image with the required resolution.
2. Load the image with the alpha information as a Foreground image in the scene. The area with alpha information is excluded from rendering.
3. Select *Arena Image-Based* as **Render Sequence** in the Global Settings Panel.
4. During scene design, the Foreground Image can be set to *Active*, however, when the scene is completed and ready to be taken On Air, the Foreground image *must* be set to *Inactive* to allow only the key information to be applied. This procedure works with both Fast Texture Mode and Normal Texture Mode, and requires **Arena/Libero** to be set as **Render Sequence** in the Global Settings Panel. Viz Arena can be used to deactivate a currently activate image mask, transfer new image masks, or activate one of the previously transferred image mask using the corresponding mask ID.

Eclipse Render Sequence (based on the Viz Arena Image-based Render Sequence)

This render sequence's logic is equal to that of Arena Image-Based (formerly known as Arena/Libero) Render sequence, except for the changes depicted below.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

11.3.2 Global Settings Panel

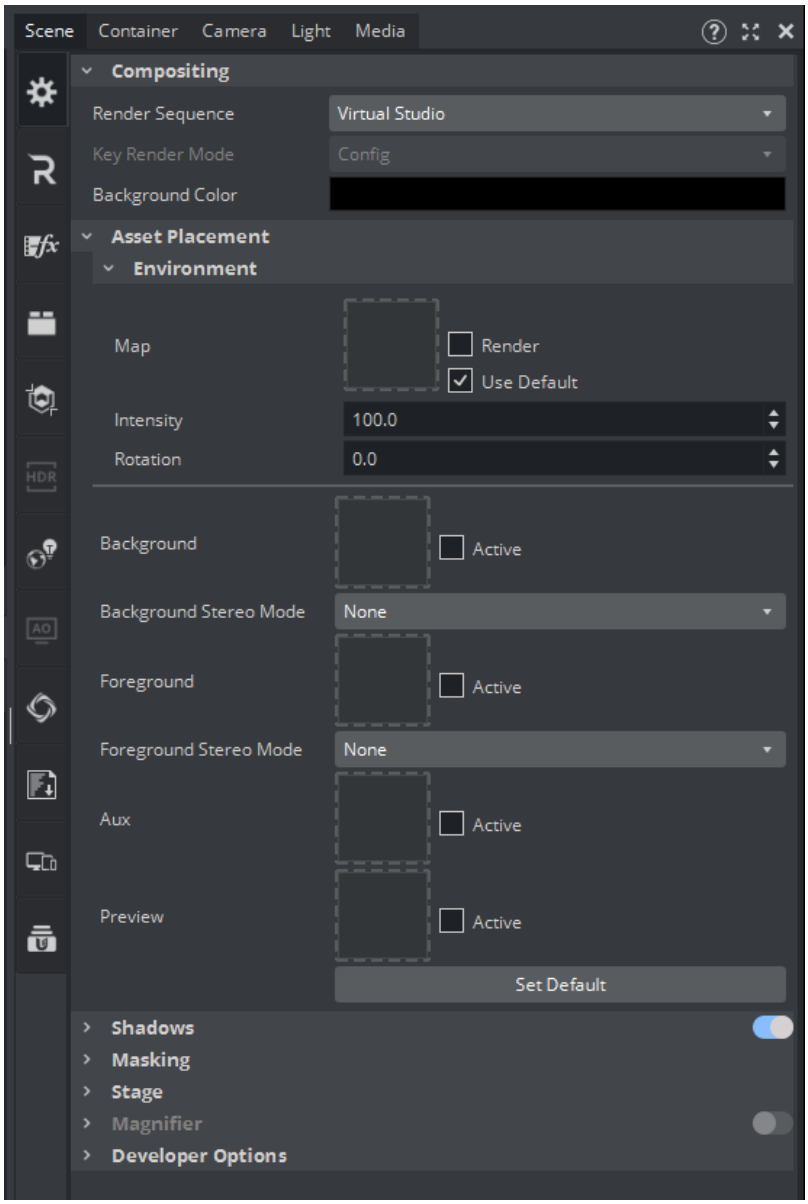


This section contains information on the following topics and procedures:

- Global Settings Properties
  - Compositing
  - Asset Placement
    - Environment Settings
    - Image and Media Asset Placement
  - Shadows
  - Masking
  - Stage
  - Magnifier
  - AI Keyer
  - Attribution
  - Developer Settings
- Images as Background or Foreground
  - To Remove an Image
  - To View an Image Source Project or Folder
  - To Show/Hide an Image in the Scene Editor

Copyright © 2026 Vizrt

Global Settings Properties



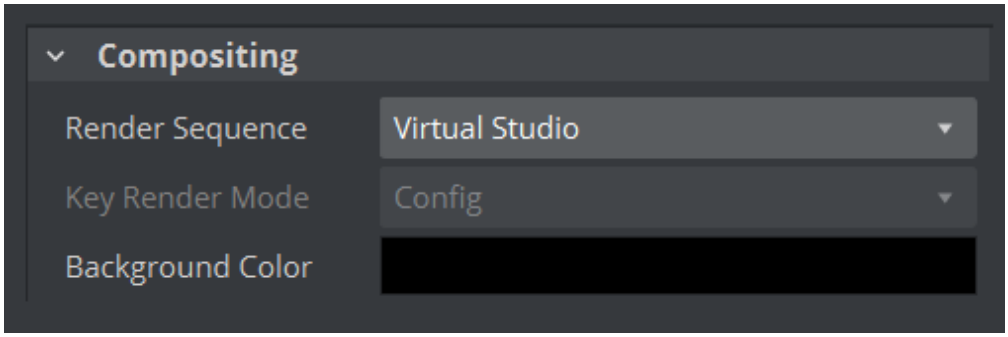
In the **Global Settings** panel, the basic parameters for a Scene can be modified:

Compositing

- **Render Sequence:** Defines how the alpha key signal must be generated (default is **Overlay**). These settings define how the compositor of Viz Engine generates the final fill and key signal.

Warning: Caution is advised when changing these settings.

Copyright © 2026 Vizrt



The following render sequences are available:

Sequence	Classic	Viz Engine	Description
Virtual Studio	✓	✓	Switches from **Overlay** to **Full Screen** when the control mode of the current camera switches from **Remote** to **Editor** (see Camera Editor). In a Virtual Studio set, to perform a camera flight with the virtual camera when the real camera is stationary, the graphics must be set to **Full Screen** to avoid a situation where the virtual and the real camera have different position values. When the camera is switched from **Remote** to **Editor** (to do the flight) the keyed graphics automatically switch from **Overlay** to **Full Screen**.
Fullscreen	✓	✓	Enables downstream compositing and generates a full screen key that outputs a non-transparent key signal when the Render Sequence is set to Full Screen. This means everything without a Key plug-in is treated as background graphics and appear behind DVEs on the output. Everything with a Key plug-in is rendered in front of DVEs. For setups using a Matrox X.mio video board, the key preview is all black, instead of all white, while show key is enabled in the Scene Editor. This allows the designer to work with containers with Key plug-ins during scene design.
Overlay	✓	✓	Sets as key signal only those Containers that have a Key function applied. Only Containers in the Scene Tree with a Key function applied are rendered into the key signal. Use this mode to correctly blend graphics and DVEs.
Background Image	✓	⊖	Uses the alpha values of the background image as a key signal and Containers with a Key function applied and rendered on top. This option is only applicable if a background image with an alpha layer is added.

Copyright © 2026 Vizrt

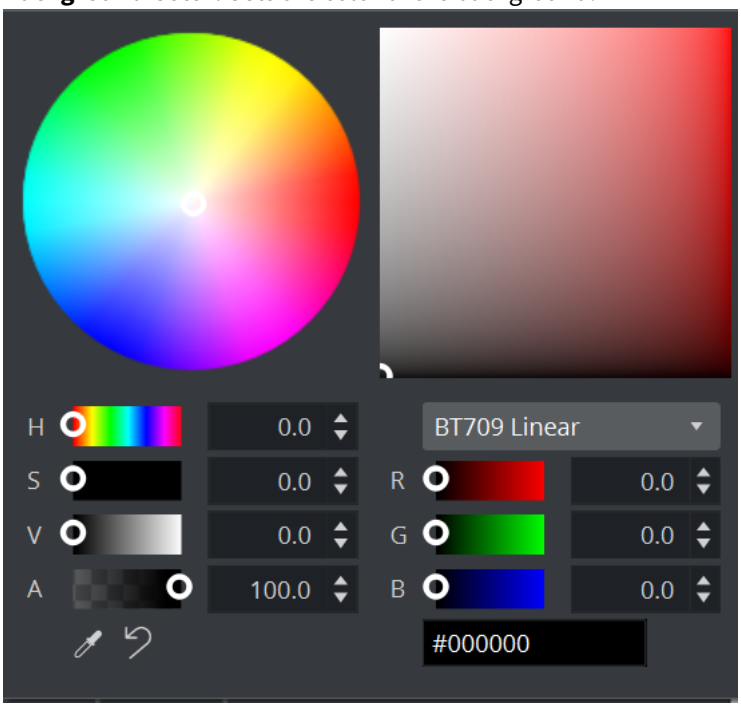
Sequence	Classic	Viz Engine	Description
Auto Key	✓	✓	Generates the key signal automatically from the Containers in the Scene Tree, it does not require the application of Key functions to the Container. This function provides a worry-free overlay. This is the preferred mode for overlay graphics.
Libero	✓	✓	The render sequence for a Viz Engine used in with Viz Libero (see Viz Libero and Viz Arena Render Sequences).
Arena	✓	⊖	The render sequence for a Viz Arena/multi-zone keying setup (see Viz Libero and Viz Arena Render Sequences).
Arena Image-Based	✓	✓	The render sequence if the tracked camera is a Viz Arena image tracker (see Viz Libero and Viz Arena Render Sequences).
Eclipse	✓	⊖	The Viz Eclipse render sequence, based on the Arena Image-Based render sequence with additional functionality for dirty PGM feeds. (see Viz Libero and Viz Arena Render Sequences).
External VS	✓	✓	The render sequence for the Unreal Engine Integration (see Integration with Unreal Engine).
External AR	✓	✓	The render sequence for the Unreal Engine Integration (see Integration with Unreal Engine).
Arena Image Based Fill & Key	✓	⊖	The render sequence if the tracked camera is a Viz Arena image tracker and needs separate fill and key output (see Viz Libero and Viz Arena Render Sequences).
Libero Fill & Key	⊖	✓	The render sequence if the separate fill and key output for Viz Libero is needed (see Viz Libero and Viz Arena Render Sequences).
Passthrough	⊖	✓	Foreground Media Assets are passed directly to the output without any rendering. Needed by third party integrations like Unreal.
Multi-Layer Virtual Studio	⊖	✓	Multipass rendering where geometries without key are rendered first, then the talent and geometries with key are properly composited on top of it.

Copyright © 2026 Vizrt

Sequence	Classic	Viz Engine	Description
Multi Layer XR Set	⛔	✅	Multipass render sequence for advanced Set Extension and AR rendering.
Multi Layer External VS	⛔	✅	Multipass render sequence for Unreal Integration combining Unreal Virtual Studio with Viz Engine Renderer graphics and Precision Keyer.

Info: For any multi-layer render sequence, Layer plug-in must be used to control which container is rendered to which layer. See Layer for more details.

- **Key Render Mode:** (Classic Render Pipeline only)
  - **Config:** Renders the key according to the settings in the Viz Configuration (see the **Configuring Viz** section of the Viz Engine Administrator Guide).
  - **Single Pass:** Uses one rendering step. Use this for all new scenes and for scenes imported from version 2.8, as it is much faster.
  - **Double Pass:** Uses two rendering steps. Use for backward compatibility in scenes created in Viz Artist versions 3.0 to 3.3.
- **Background Color:** Sets the color of the background.

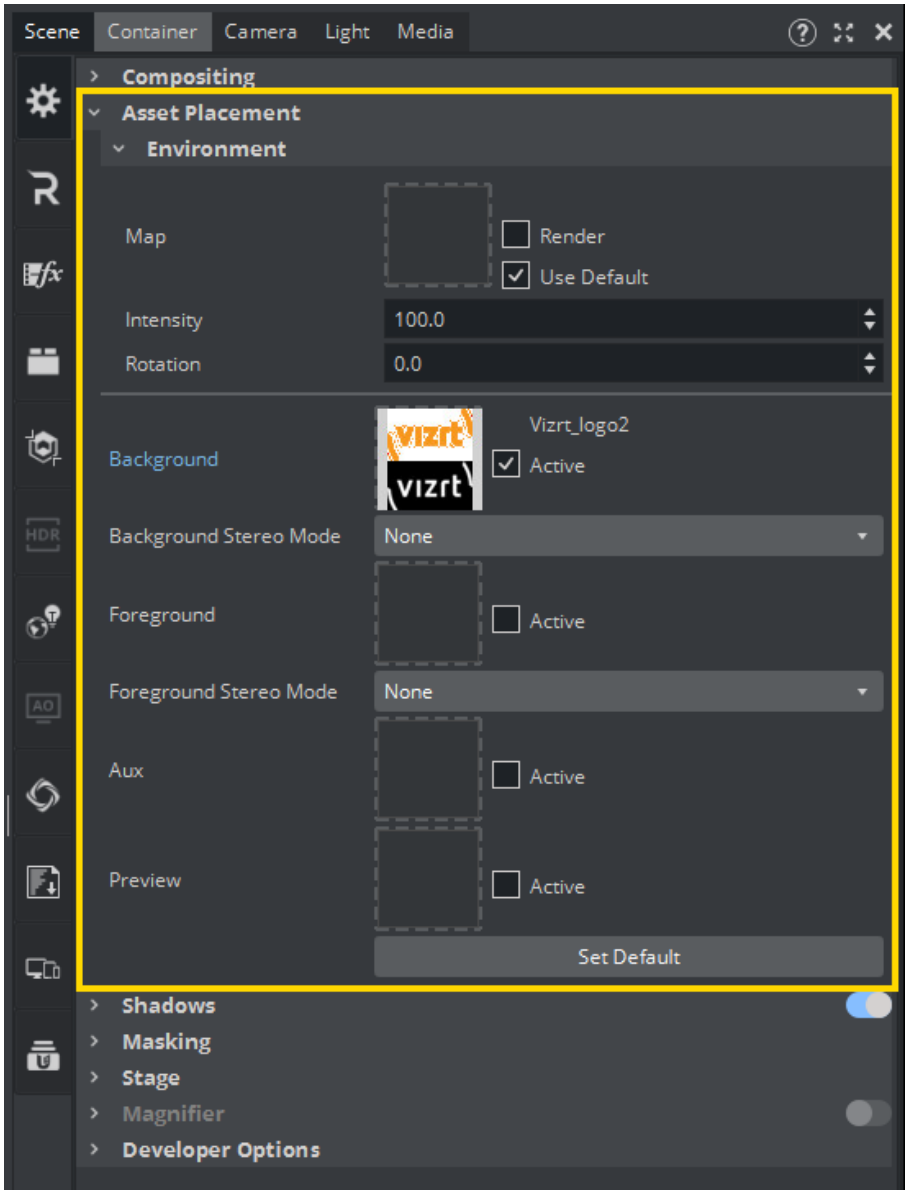


Asset Placement

Environment Settings

The Environment Map for Viz Engine Rendering is defined in this section. It requires a prefiltered HDR Environment Map to be placed here. If no suitable map is available, **Use default** renders a blue sky environment map. This map defines how much environment map lighting is used as well as how "metallic" your objects are, the **intensity** may vary the output of your design. your environment can also be **rotated**, for example to put more specular light on your object.

Image and Media Asset Placement



Copyright © 2026 Vizrt

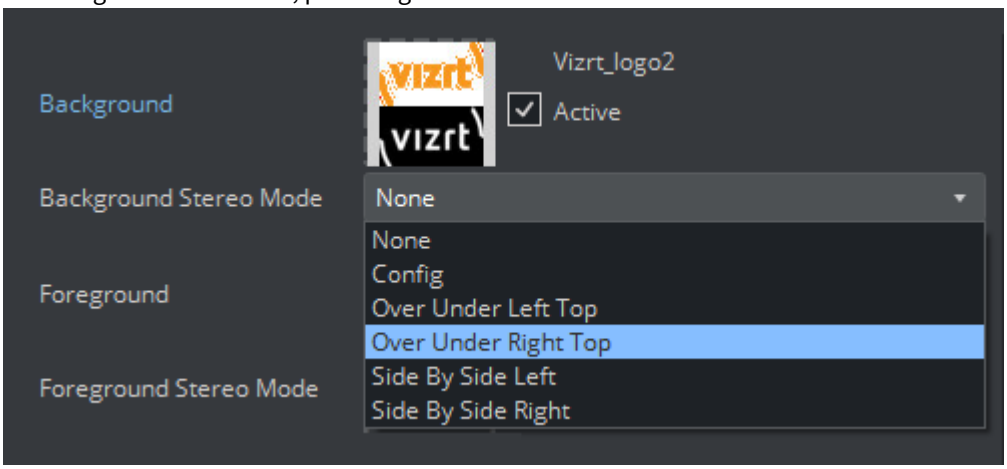
- **Background:** Adds an image or Media Asset as the Scene background.
- **Foreground:** Adds an image or Media Asset as the Scene foreground.

Note: In the Viz Engine Render Pipeline, Post Processing Effects need to be turned on to see the Foreground in the Scene editor.

- **Preview:** Adds a fixed layout to the Scene. This visualizes how the total preview (camera picture and/or graphics/Media Asset) looks. To set the image as global default, click the **Set Default** button.

Tip: It is possible to drag results from a database search onto the Background, Foreground, and Preview drop zones.

- **Stereo Mode:** (license required) Selects the correct stereo mode for the inserted image (not available if **Stereo Mode** is set to `Off` in the Viz Configuration (see the **Camera** page in the **Configuring Viz** section of the Viz Engine Administrator Guide):
  - **None:** No Stereo mode.
  - **Config:** Images are drawn as set in the Viz Configuration (see the **Camera** page in the **Configuring Viz** section of the Viz Engine Administrator Guide).
  - **Over Under Left Top** or **Over Under Right Top:** Two images are stored, one over (top) and the other under (bottom), left or right eye (top) first. The image height is halved, providing half resolution.
  - **Side by Side Left** or **Side by Side Right:** Two images are stored, side by side, left or right eye first. The image width is halved, providing half resolution.

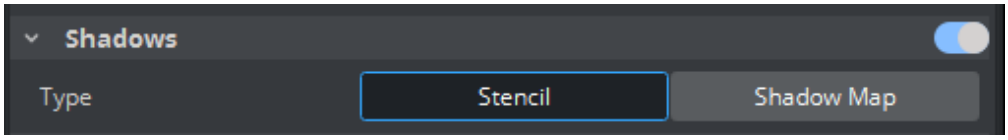


Shadows

Enables or disables the shadow function in a Scene.

For the Classic Render Pipeline only: The shadow function uses two layers from the mask function. With shadows enabled, there are only six layers available for the mask function (see Mask Source and Mask Target). In a scene where more than six mask layers are needed, shadows can be disabled (only very complex mask configurations need more than six layers). The **Stencil** shadow type is the least resource-demanding way to render shadows in scenes. For more advanced and better-looking shadows, cascaded shadow maps can be enabled. For more in-depth information on shadows in Viz Artist, please refer to Shadow Maps and Shadow Caster and Shadow Receiver.

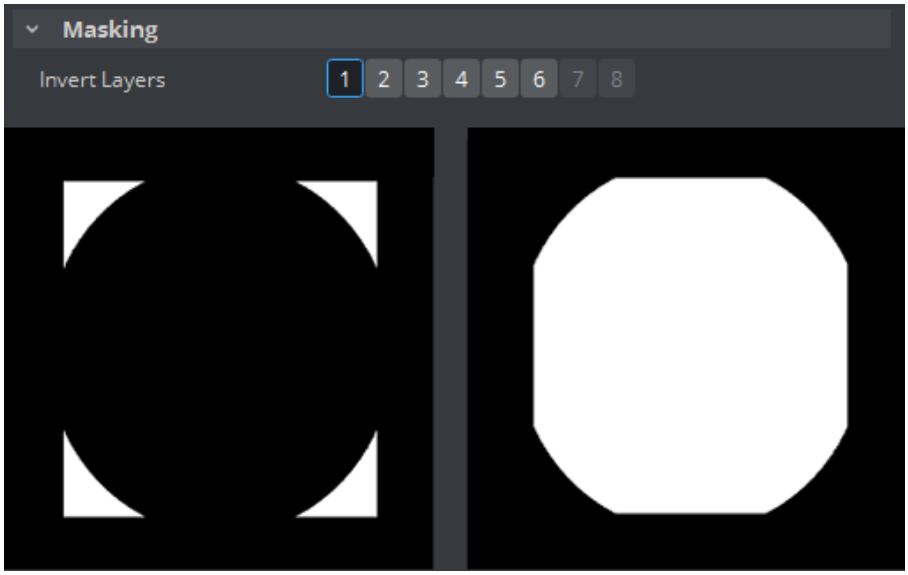
Copyright © 2026 Vizrt



Classic Render Pipeline: The Shadows On/Off button is only available when shadow type is set to Stencil, as Shadow Mapping is not affected by this setting.

Masking

Invert Layers: Makes it possible to set one or more of the mask layers in a Scene to be inverted. With no mask inverted (1), the mask cuts out a hole where it covers the mask target object. With mask invert enabled (2), the mask target is only visible where the mask covers it:



Stage

These settings define, how the timeline is positioned:

- **Selected:** Sets the timeline of all selected directors to the first found tag selection.
- **First:** Sets the timeline of the first found director containing the searched tag to the tag position.
- **Separate:** Sets the timeline for each director, containing the searched tag, to the tag position.

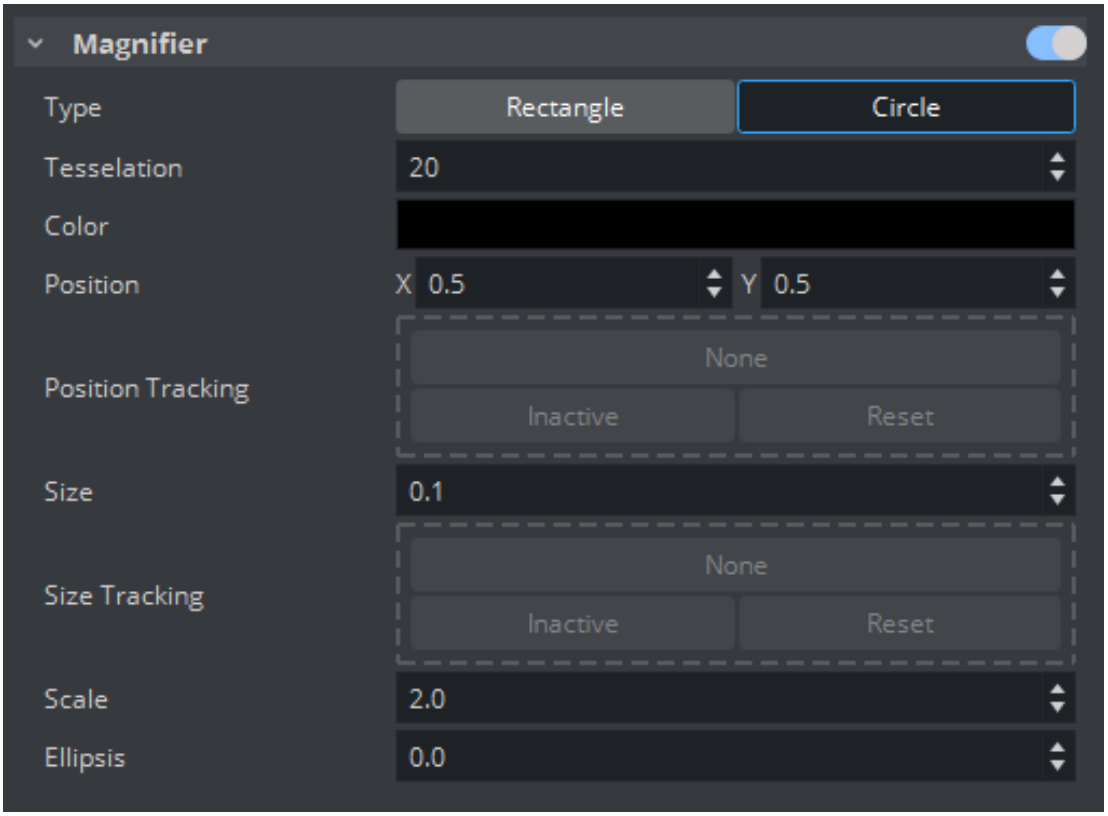
Magnifier

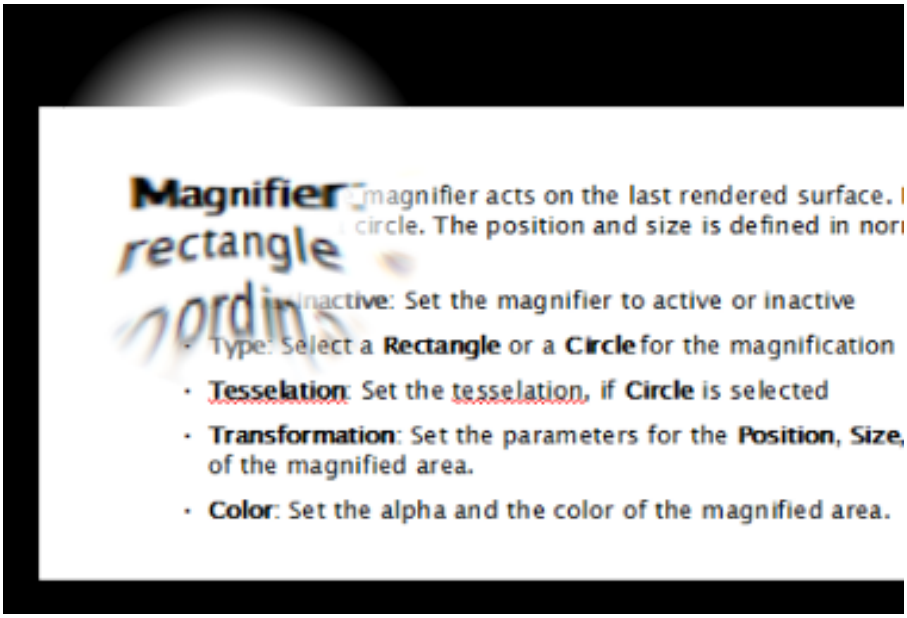
The magnifier is mainly used by Viz Sports products like Viz Libero or Arena. It acts on the last rendered surface. It can be either a rectangle or a circle. The position and size is defined in normalized view-port coordinates.

- **Active/Inactive:** Sets the magnifier to active or inactive.
- **Type:** Selects a **Rectangle** or a **Circle** for the magnification area.
- **Tesselation:** Sets the tessellation, if **Circle** is selected.

Copyright © 2026 Vizrt

- **Transformation:** Sets the parameters for the **Position**, **Size**, **Scale** and **Ellipsis** of the magnified area.
- **Color:** Sets the alpha and the color of the magnified area.
- **Container:** Selects a Container to control the position and, or, the size of the magnifier. Drag the Container to be tracked in the desired box.

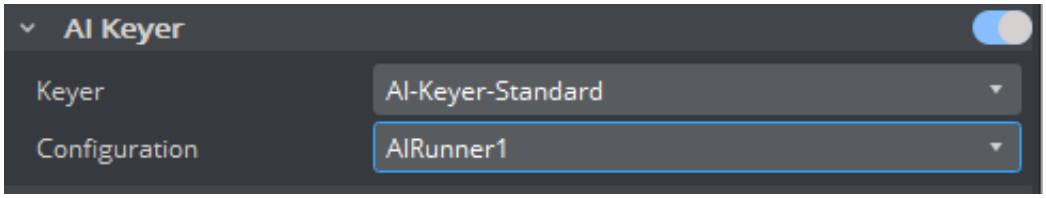




Copyright © 2026 Vizrt

AI Keyer

This feature allows to automatically launch an AI Keyer process if the scene is set to the Main layer.
The available options represent pre-existing configurations in the local AI framework.
Configurations for the AI Keyer must be created from within AI Terminal.



- **AI Keyer**: Enable / Disable automatic launch of AI Keyer process for this scene
- **Keyer:** Type of keyer to be used.
- **Configuration:** available configurations according to the chosen keyer type.

As soon as a scene configured with AI Keyer option is started, pre-existing AI Keyer processes will be terminated and the required one is started up.
Since the AI Keyer is started up on scene load, it might take a few seconds until the image compositing is final.

Attribution

Sets the Attribution for Viz Maps rendering.
See Maps Prerequisites for details

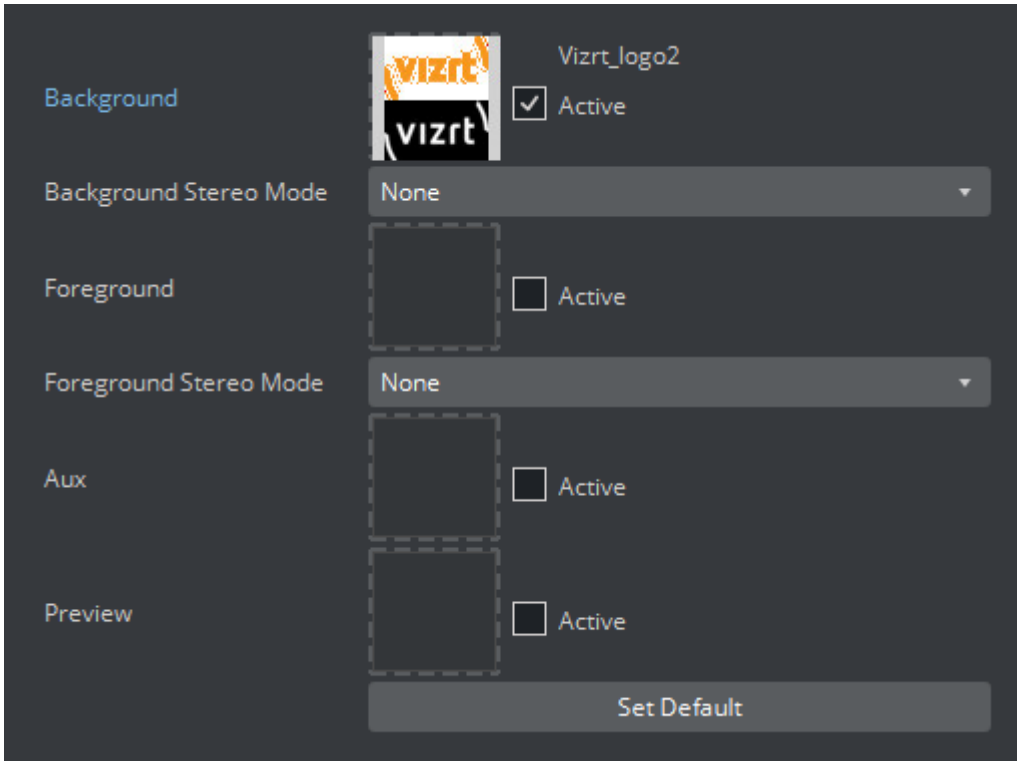
Developer Settings

Reserved for special settings. These settings should only be used when told so.

Images as Background or Foreground

This section details procedures with images use as background, foreground and preview.

Copyright © 2026 Vizrt



To Add an Image

1. Click **Asset View**.
2. Click the **I** tab for Images (1).
3. Drag the required image onto the drop zone (2). A thumbnail of the image is shown in the drop zone button. Dragging a new image onto the drop zone automatically replaces the old image.

To Remove an Image

Drag the image from the drop zone to the trash can or double click on the blue label.

To View an Image Source Project or Folder

Click the image on the drop zone. The project or folder it is placed in is visible. If the image is linked, the first project or folder is selected.

To Show/Hide an Image in the Scene Editor

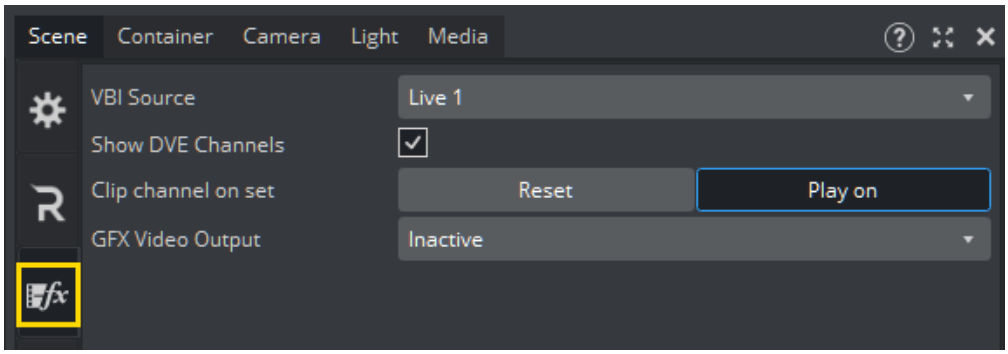
Click the **Active/Inactive** button next to the drop zone (3). If the image has a different aspect ratio than the screen resolution, the image is stretched to fit.

Copyright © 2026 Vizrt

11.3.3 Global Media Assets Settings

These are the common Media Asset properties that apply on a global level.





- **VBI Source:** Selects which channel to use for output. Types of data sent include closed captions, teletext, time codes and other digital data. The setting **Clip Timecode Type** can enable Viz Engine to provide additional time code data in the VBI-signal, if required. The setting can be changed by editing the Viz Configuration file and setting the variable `clip_timecode_type = <VALUE>`. Please refer to the table Clip Time Code Type Values below for valid values. It can also be set on the fly using the `Send` application to issue the following command: `CONFIGURATION*CLIP_TIMECODE_TYPE SET <VALUE>`

Note: This setting only applies for installations with Matrox cards.

- **Show DVE Channels:** Enables or disables DVE placeholder box drawing.
- **Clip Channel on set**
  - **Play on:** Continues to play the clip even if the Scene does not contain that specific clip channel. This is the default behavior. When setting a Scene with this channel active, the clip is shown from its current position onwards, when no clip or the same clip is set in the Scene.
  - **Reset:** Starts playout from the beginning of the clip when it has the same clip name, or show black when no clip is set in the channel at all. Please observe that the commands and values are case sensitive. Any on-the-fly changes are saved to the Viz Configuration file upon proper shutdown of Viz Engine.
- **GFX Video Output:** Allows sending the content of the selected GFX Channel to the Program Channel Output.

Note: This setting only applies for installations with Matrox cards.

Copyright © 2026 Vizrt

Clip Time Code Type Values

Value	Description
`DEFAULT`	Mimics the behavior of earlier versions, and provides a time code if available. If not, no time code information is provided in the VBI-signal.
`TIMECODE`	Provides a time code if available. If an explicit time code is not embedded in the clip, no time code information is provided in the VBI-signal.
`ZEROBASED`	The embedded time code is `0` at the clip start and counts up to the last frame based on the number of frames in the clip.
`COUNTDOWN`	The embedded time code is the duration of the clip at the beginning and counts down to `0` based on the number of frames in the clip.

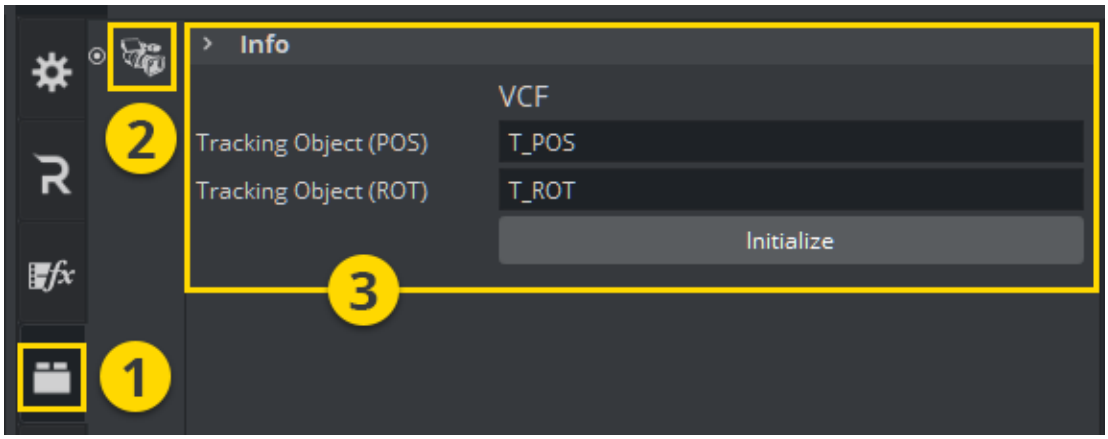
Copyright © 2026 Vizrt

11.3.4 Plug-in Panel



Scene plug-ins can be applied to a Scene in the plug-in panel. Scene plug-ins are global functions that have influence on the whole Scene. They are mainly used when external programs are interfaced with Viz Artist. Viz Artist uses an open interface for plug-ins. New plug-ins can either be obtained or written, and then installed into Viz Artist. Installed Scene plug-ins can be found under Plugins in the Main Menu. To view the Scene plug-ins, click on the **Plugins** button and select **Scene plug-ins**.

For a detailed list and description of Scene plug-ins, see **Scene plug-ins** in the Viz Plug-ins User Guide.



The Scene plug-in panel (1) is divided in two sections:

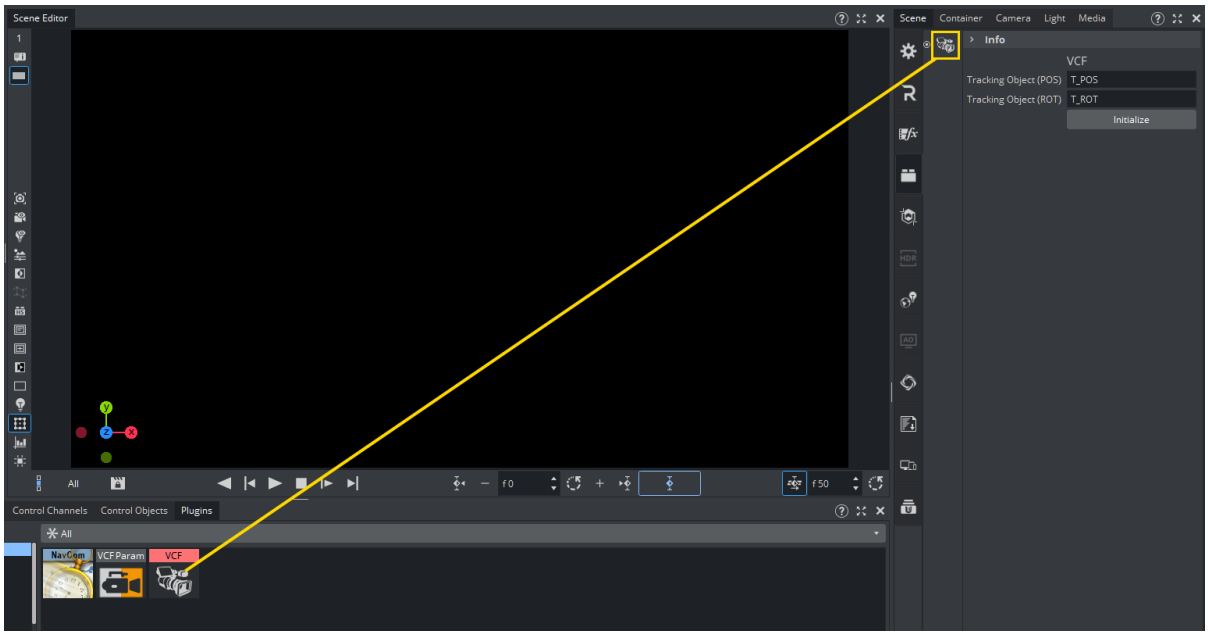
- Scene plug-ins applied to a Scene, as icons (2)
- Scene plug-in editor (3) opens when the Scene plug-in is highlighted).

This section contains information on the following topics and procedures:

- To Add a Scene Plug-in
- To Edit a Scene Plug-in Properties
- To Show Scene Plug-in Information
- To Enable/Disable a Scene Plug-in
- To Remove a Scene Plug-in

Copyright © 2026 Vizrt

To Add a Scene Plug-in



1. Click **Plugins**.
2. Select **Scene plug-ins** (SP). Browse for the Scene plug-in to apply to a Scene.
3. Drag the Scene plug-in to the Scene plug-in panel:
    - Drag the selected Scene plug-in to the drop zone, or
    - Right-click on the Scene plug-in icon and select **Add to Scene...**.

To Edit a Scene Plug-in Properties

1. In the Scene Settings plug-in area, select the plug-in.
2. Edit the plug-ins properties in the editor to the right.

To Show Scene Plug-in Information

1. In the Scene Settings plug-in area, select the plug-in.
2. In the plug-in editor, click the info (Info) icon.

To Enable/Disable a Scene Plug-in

In the Scene Settings plug-in area, click on the enable/disable switch in the icon.

Copyright © 2026 Vizrt



For greater convenience, Viz Artist allows you to enable or disable the plug-ins already applied to a Scene. This gives overview when adjusting the plug-in properties to the values you need.

> **Note:** After you add a plug-in to the scene, it is automatically enabled.

To Remove a Scene Plug-in

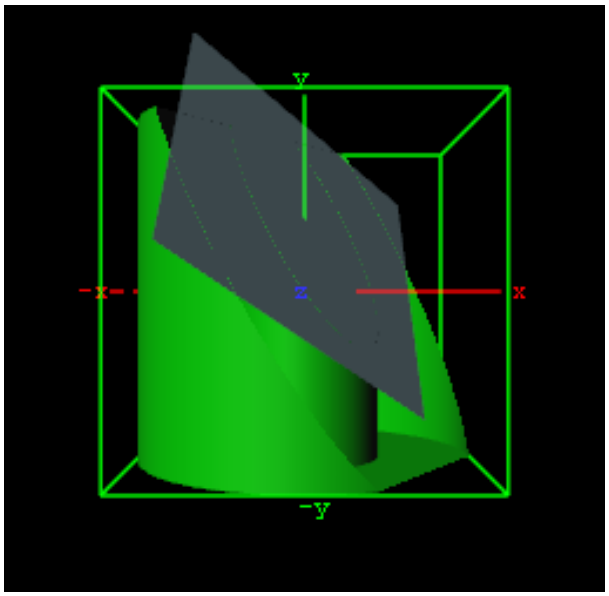
- Drag the plug-in to the trash can (), or
- In the Scene Plug-in tab, right-click the plug-in and select **Remove from Scene**.

Copyright © 2026 Vizrt

11.3.5 Clipper Panel



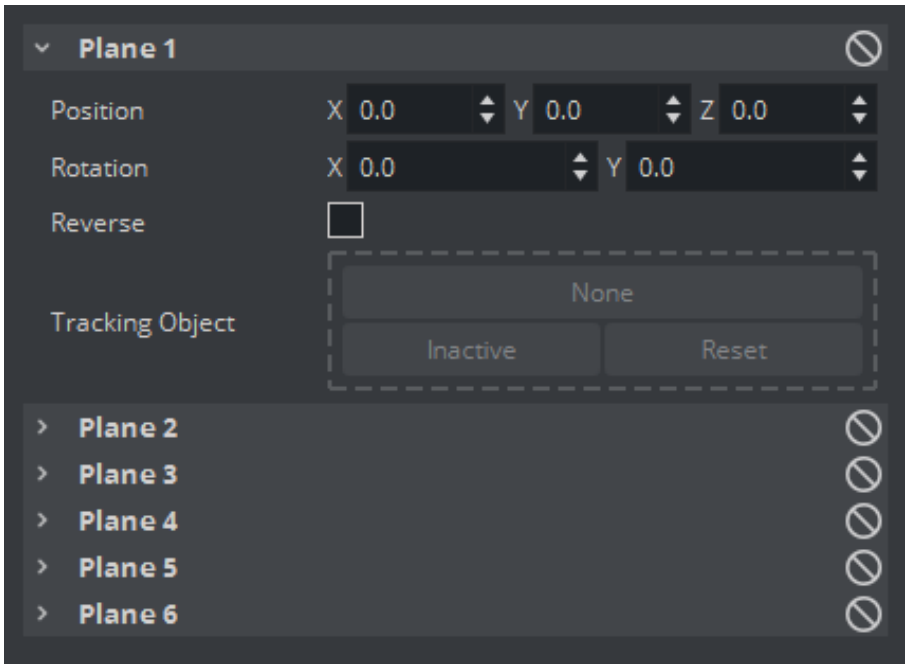
In the Clipper panel, settings for scenes that have the Clipper plug-in attached can be modified.



The Clipper plug-in is an alternative to the mask function; the clipper works in 3D and the mask in 2D. Objects using the clipper plug-in, when placed behind the clipper plane, are masked/clipped. If an object is only partially behind a clipper plane, only the part of the object that is behind the plane is clipped out since the clipping is done in true 3D space. The plug-in uses the OpenGL clipping planes that supports up to six planes.

Copyright © 2026 Vizrt

Clipper Properties



- **Plane Number:** Shows the clipper plane number, from `1` to `6`.
- **State:** Sets the selected clipper plane `On` or `Off`.
- **Position:** Modifies the position of the clipper plane along the X, Y, and Z axis.
- **Rotation:** Modifies the rotation along the X and Y axis.
- **Reverse:** Shows everything in front of the clipper plane when set to `Off`. Shows everything behind the clipper plane when set to `On`. Tracking Objects with Clipper Plane.
- **Tracking Object:** A clipper plane can be set up to track containers regarding position and rotation. Tracking position means that the position of the clipper plane is in the center of the container, and tracking rotation means that the clipper plane rotates if the object rotates. This is especially useful when animating a scene.

See Also

- Create Animations

Copyright © 2026 Vizrt

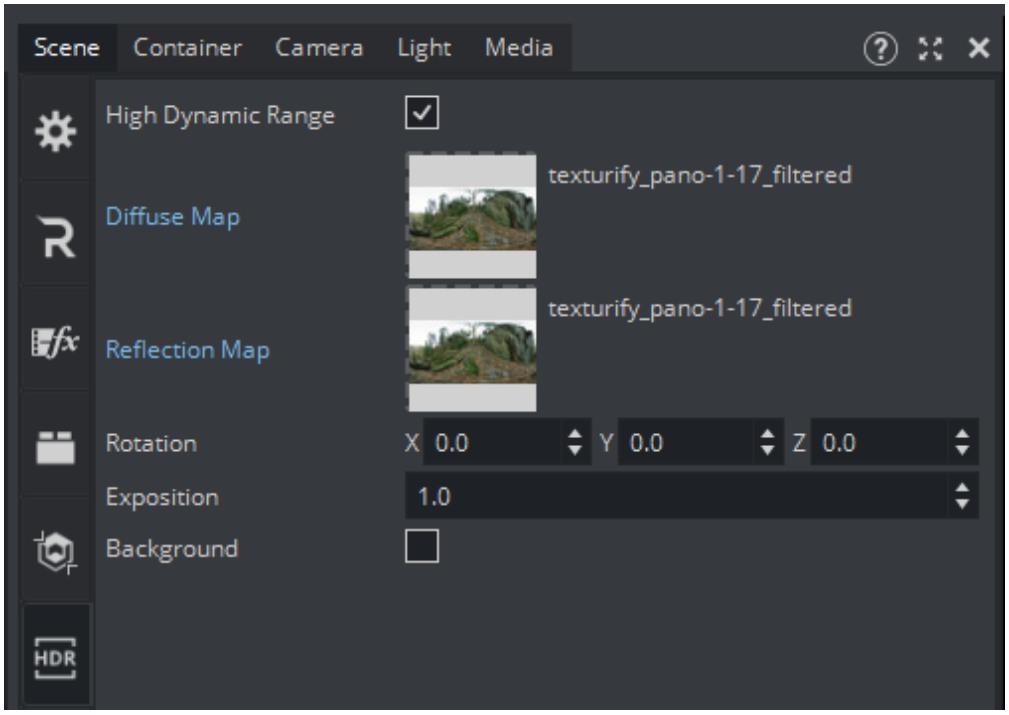
11.3.6 High Dynamic Range Panel



Enables High Dynamic Range Rendering/Imaging for the selected container. This improves the contrast of the generated scene in a much more realistic way.

> **Note:** This applies to the Classic Render Pipeline only.

HDR Properties



- **High Dynamic Range:** Enables HDR lighting instead of the default OpenGL lighting on Containers tagged for HDR lighting when set to `On`. To tag Containers for HDR lighting, the HDR plug-in must be placed in a Container to enable HDR lighting on the respective container and its sub-tree.
- **Diffuse Map:** Defines the diffuse light map. This is a HDR Cube map (must be a vertical cross and should have been imported from an *.exr* file).
- **Reflection Map:** Defines the diffuse light map for the reflection part of the lighting.

Copyright © 2026 Vizrt

⚠ **Note:** Diffuse and Reflection map require the texture mapping method to be `Cube`. Do not reuse the same texture, where the texture mapping method needs to be something different than `Cube`, but use a duplicate instead.

- **Rotation:** Rotates the environment map with the set parameters.

⚠ **Note:** Diffuse map and reflection map together form the 'environment map'.

- **Exposition:** Controls the brightness of the maps (similar to the exposure time of an ordinary camera).
- **Background:** Draws the Reflection map as background when set to `On`.

Copyright © 2026 Vizrt

11.3.7 Real Time Global Illumination



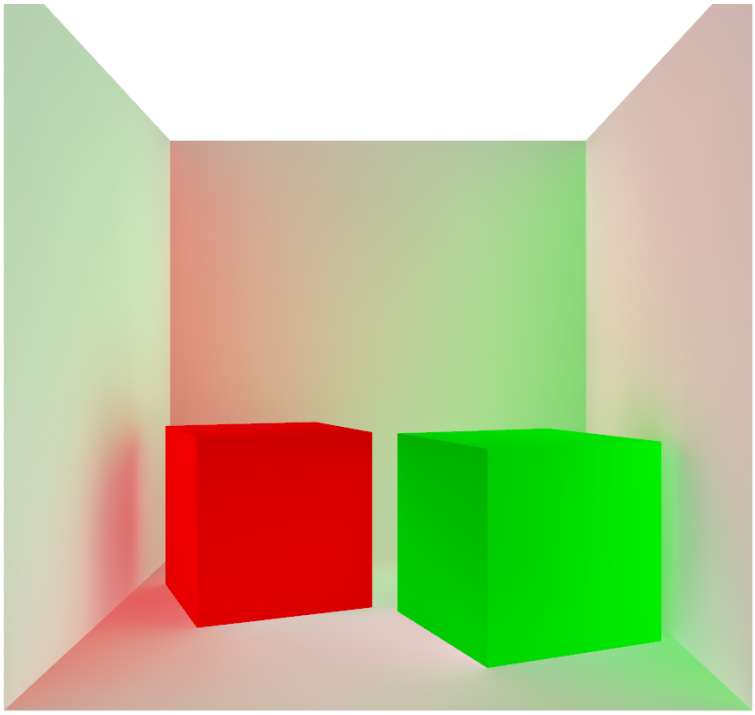
Global Illumination (GI), or indirect illumination, is a general name for a group of algorithms used in 3D computer graphics, that are meant to add more realistic lighting to 3D scenes. Such algorithms take into account not only the light that comes directly from a light source, *direct illumination*, but also subsequent cases in which light rays from the same source are reflected by other surfaces in the scene (*indirect illumination)*, whether reflective or not.

> **Note:** If you do not have a license for the Real Time Global Illumination, contact your local Vizrt Support or send an e-mail to license@vizrt.com to obtain a time-limited demo license or a full license.

This section contains the following information:

- Introduction to the Global Illumination Function of Viz Artist
- Working with Global Illumination
- Global Illumination Settings
    - Global Illumination
- Precomputation
    - Precomputed Realtime GI
        - Precompute Computation Examples
    - Mesh Classification
    - Ambient Occlusion (AO)
    - Render Mode
        - Render Mode Examples for Global Illumination
        - Debug Modes for Global Illumination
    - Show Probes
- Creating a Scene with Global Illumination
- Limitations

Copyright © 2026 Vizrt



Introduction to the Global Illumination Function of Viz Artist

Introduced in Viz Artist 3.9, Real Time Global Illumination allows you to create Lightmaps and Ambient Occlusion maps that are modifiable in real time. This technique is based on the lightmap functionality widely used in leading game engines. When activating GI, you need to specify the computation settings, and precompute the Global Illumination and Ambient Occlusion maps. In Viz Artist, you find this feature section Scene Settings **> Global Illumination**. The details of the settings are described below.

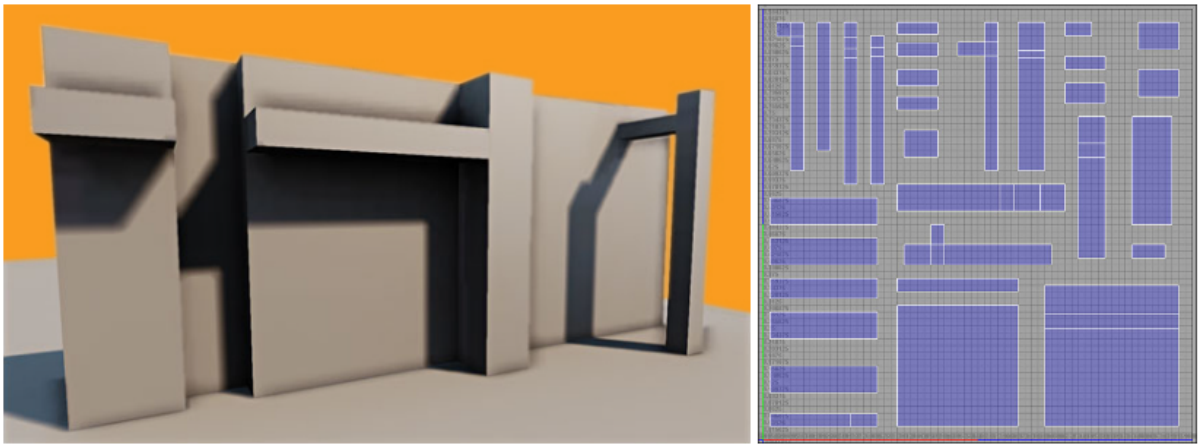
Working with Global Illumination

In general, Global Illumination requires two UV sets on any geometry that should be computed with GI: one UV set used for texture mapping, and one UV set used to create Lightmaps. There are two ways to create the Lightmap UVs to get GI working properly. You can either import an *.fbx*-file with two UV sets that you have created by yourself. The first UV set must be used for texture mapping, and the second UV set is only used for Lightmaps. Or, you can use the first UV set (UV for texturing), and let the GI system calculate the according UVs for Lightmaps.

Global Illumination has an advanced algorithm that can extract the UV coordinates of your first UV set, and automatically create UVs for the Lightmaps. The better the layout of your unwrapped UVs for texture mapping are, the better the GI system can generate the Lightmaps based on this information. The most efficient way is to create your own UVs for Lightmaps. Here, it is important to have your UV layout in the range from `0` to 1. Also, all UVs for Lightmaps must be unique and non-overlapping, otherwise GI cannot compute all Lightmaps. In general, the layout of the UVs for Lightmaps look different than the layout of UVs for texture mapping. There are several tutorials available online concerning how to create UVs for Lightmaps.

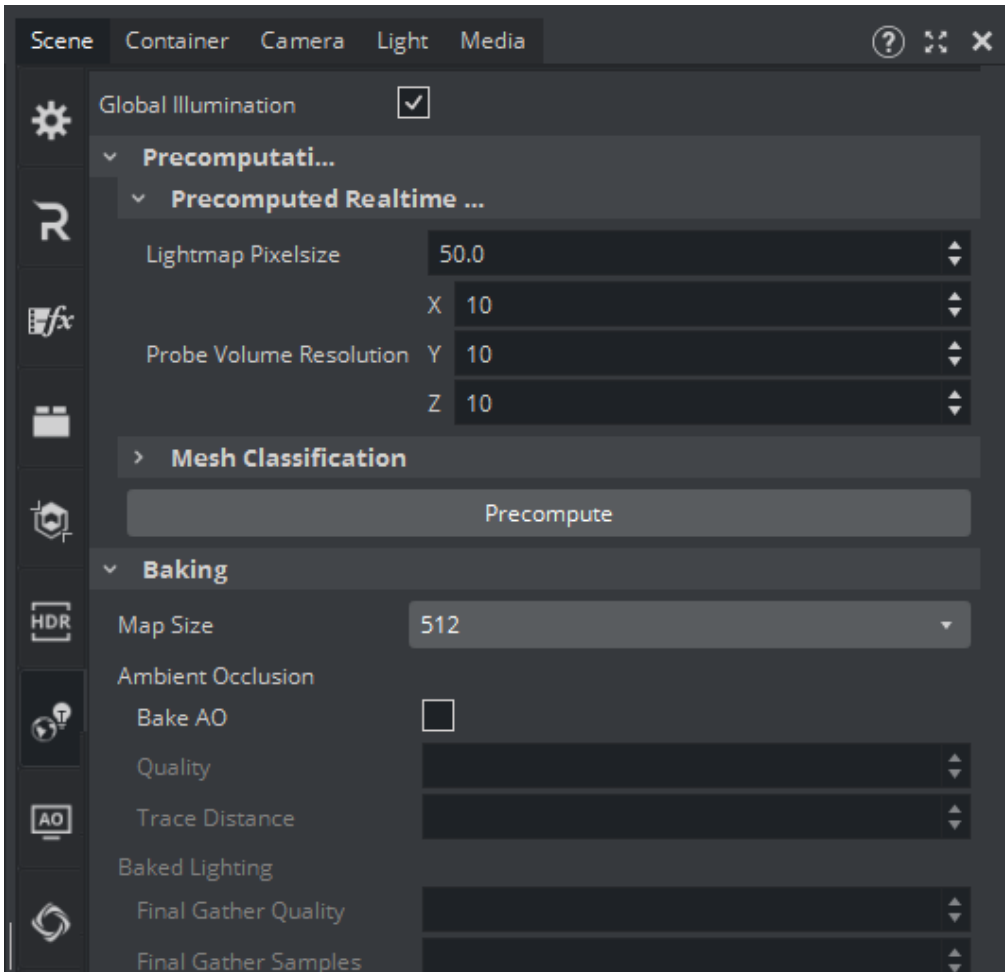
The following is an example of correct UV Layout for Lightmaps. In the image below, you see the UV layout of the second UV channel, which is used for lightmaps only.

Copyright © 2026 Vizrt

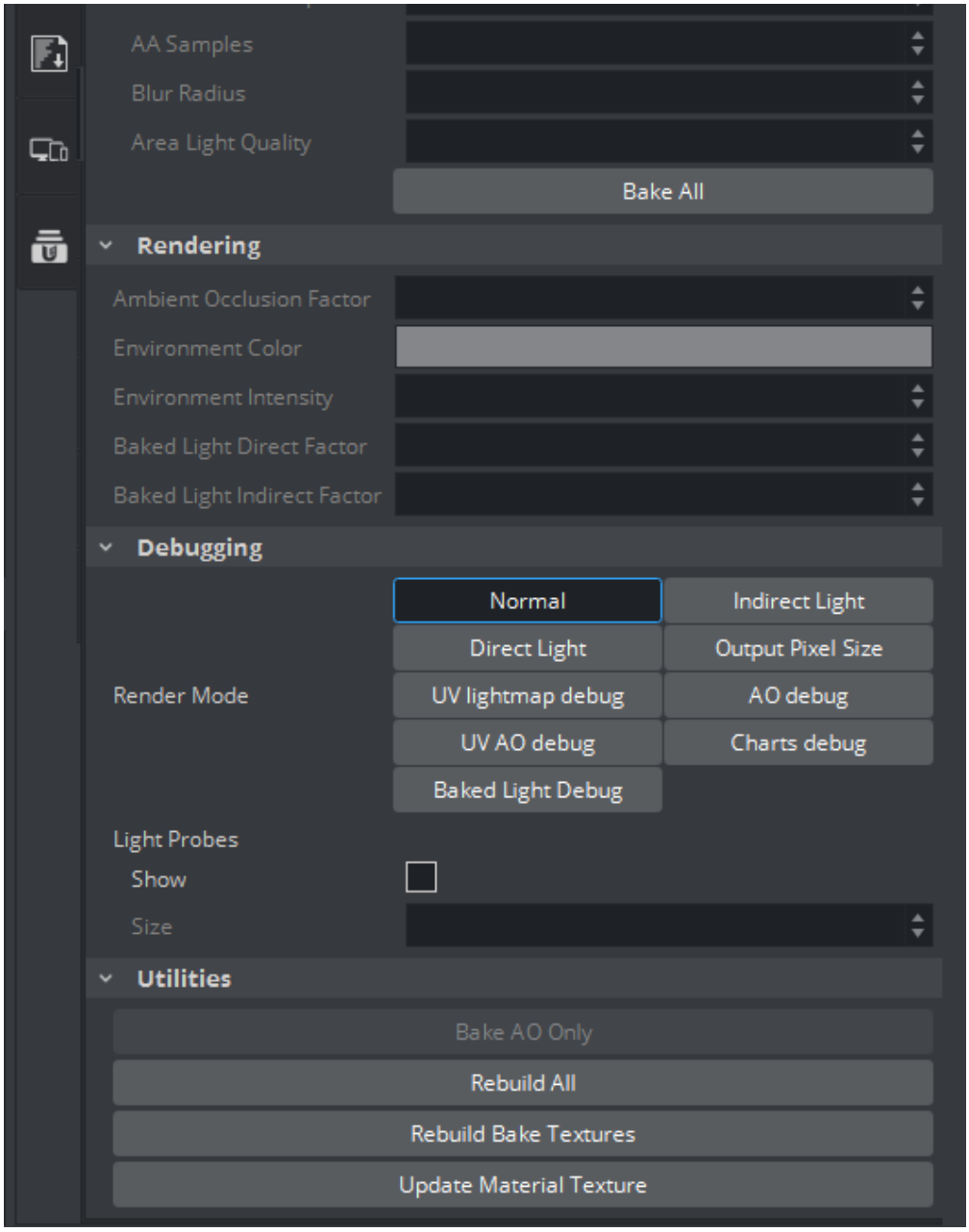


Precomputed Lightmaps do not use any memory on your GPU, so the texture memory of Viz is not reduced when having Lightmaps computed. The Lightmaps are not visible on your objects/geometries in the scene tree, as they are applied to the rendering pipeline in the render window.

Global Illumination Settings



Copyright © 2026 Vizrt



Global Illumination

Available settings: *Active* / *Inactive*

This function enables or disables Global Illumination for your scene. For the final result, Lightmaps must be precomputed. If no Lightmaps are precomputed, it enables a general pixel shading for your scene.

Lightmaps are stored inside the Viz Graphics Hub. When saving as a new scene (**Save as**), the Lightmaps is precomputed again as they are stored and linked to the UUID of your scene. If you have already calculated the Lightmaps, switching between *Active* and *Inactive* loads or unloads them. The time required to load and unload lightmaps depends on the size of your scene and Lightmaps. While loading the scene, the lightmaps are copied

Copyright © 2026 Vizrt

from the Graphic Hub to a local folder, for faster loading times. When the scene is open, the lightmaps files can be found in the Windows temporary folder:

Example: *C:\Users\[user name]\AppData\Local\Temp\vizrt\VizArtist*

The folder name the Lightmaps are stored in corresponds to the UUID of the scene they belong to.

Precomputation

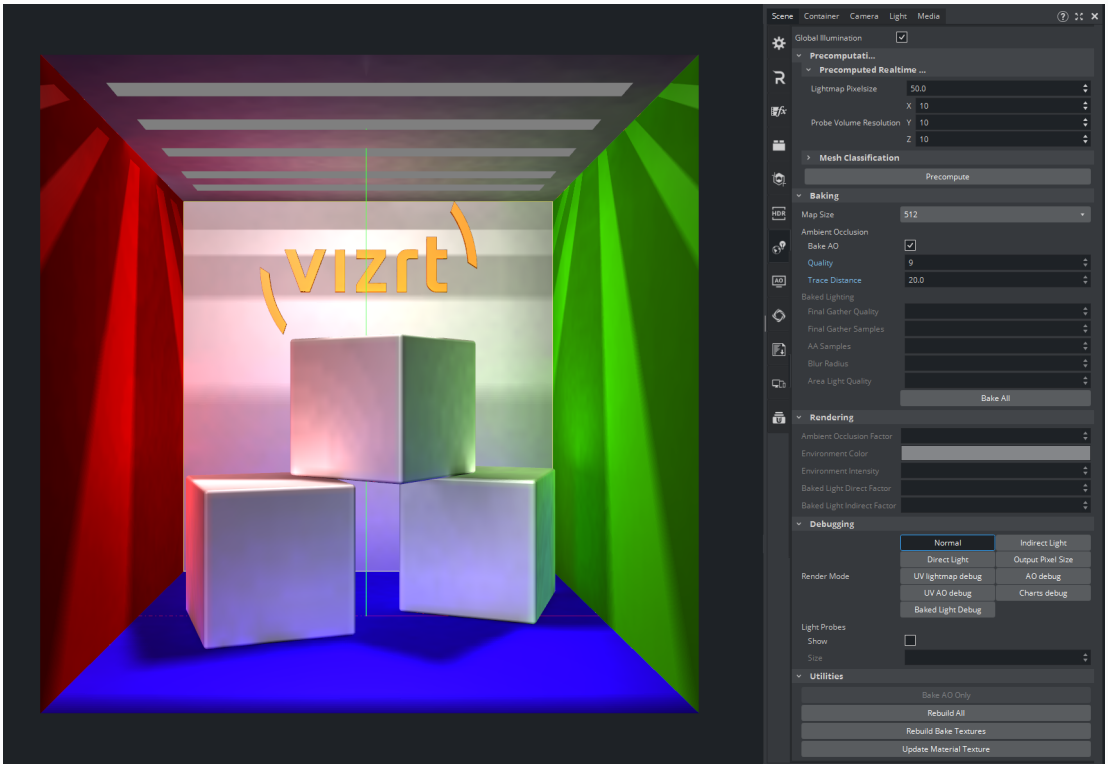
Precomputed Realtime GI

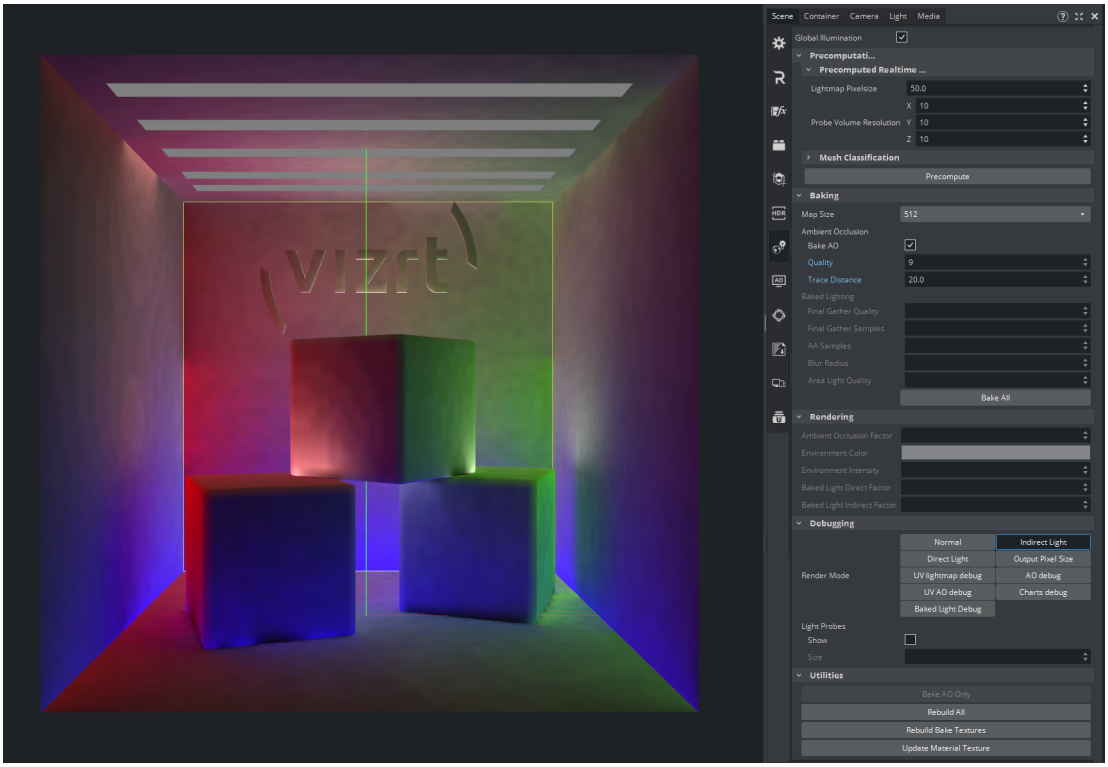
Defines the Output pixel size for lightmaps. If the pixel size is set too low, depending on your scene complexity, you are not able to generate all Lightmaps. This is because there is only a certain amount of memory available for all Lightmaps. Using a smaller pixel size value increases quality, and causes an error message when the maximum amount of pixels is exceeded. When adjusting the pixel size, make sure you are working with real world coordinates and transformations/scaling on your meshes/scene.

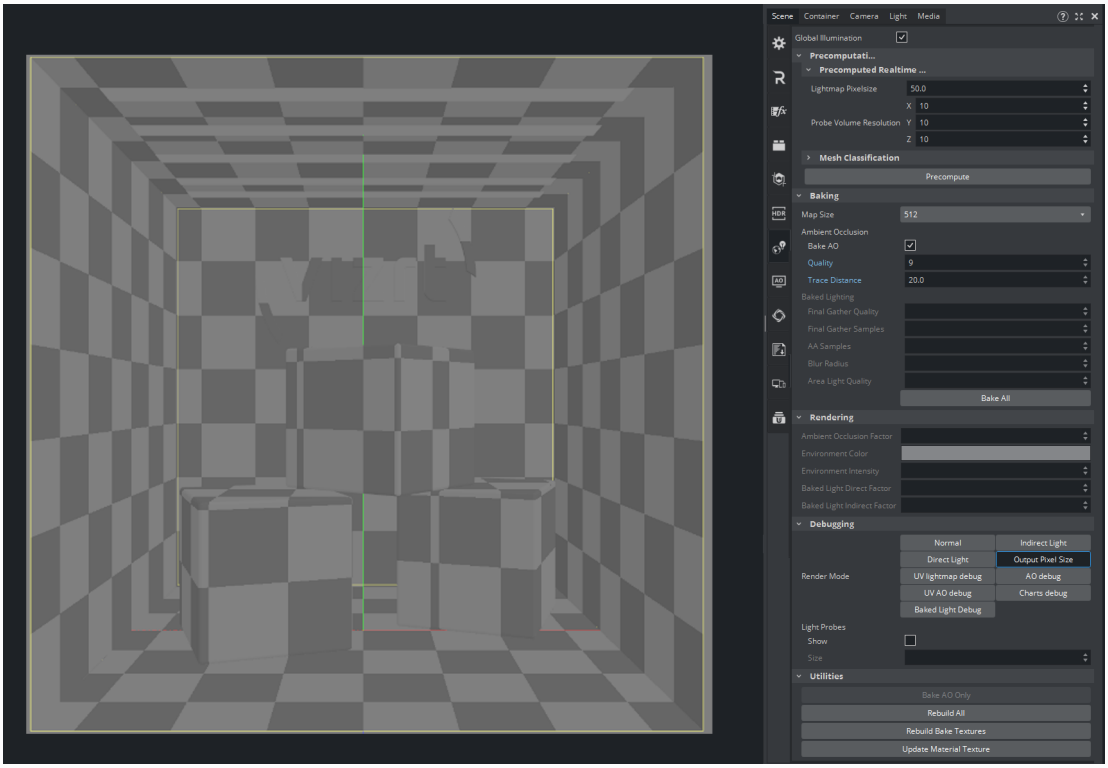
Tip: For a good start with low computation time, set the pixel size to `50` on a `1` x `1` meter cube (two pixels in one cube).

The following two examples shows the difference when using a lightmap pixel size of 50, and a pixel size of 10. Lower pixel sizes increase quality, but also increases computation time.

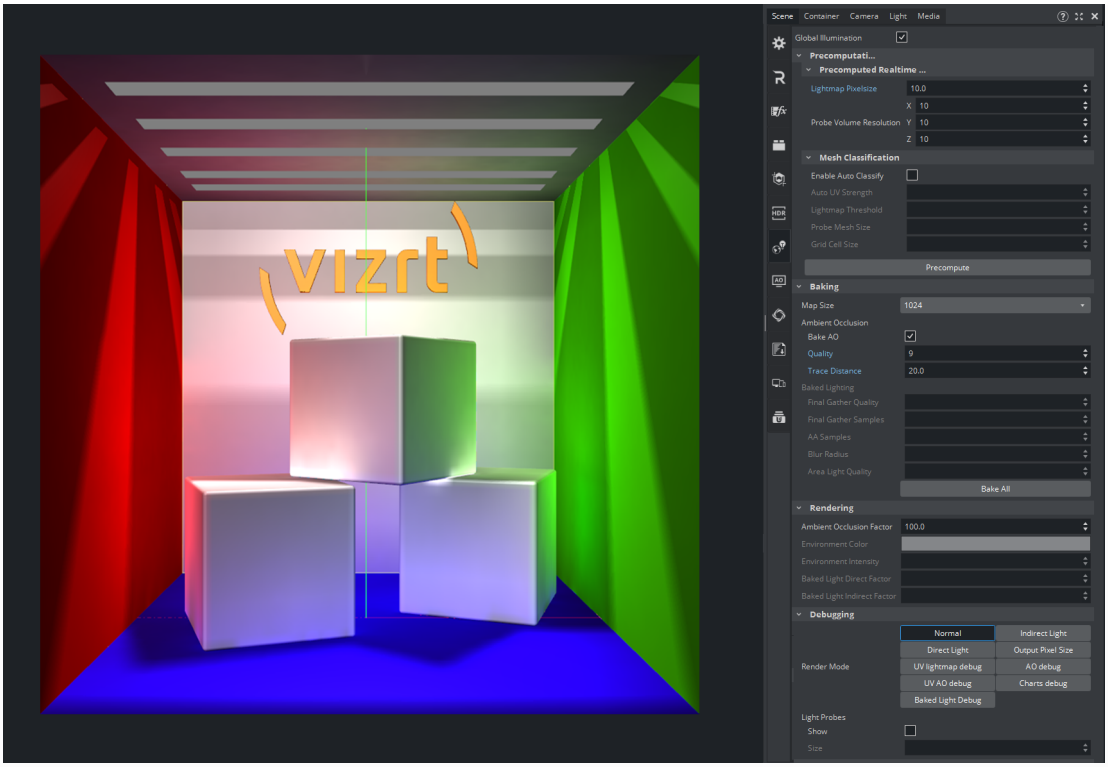
Example: In this example there is visible noise in **Normal Mode**, and also in the **Indirect Light** render pass. The pixel size used in this scene is `50`.

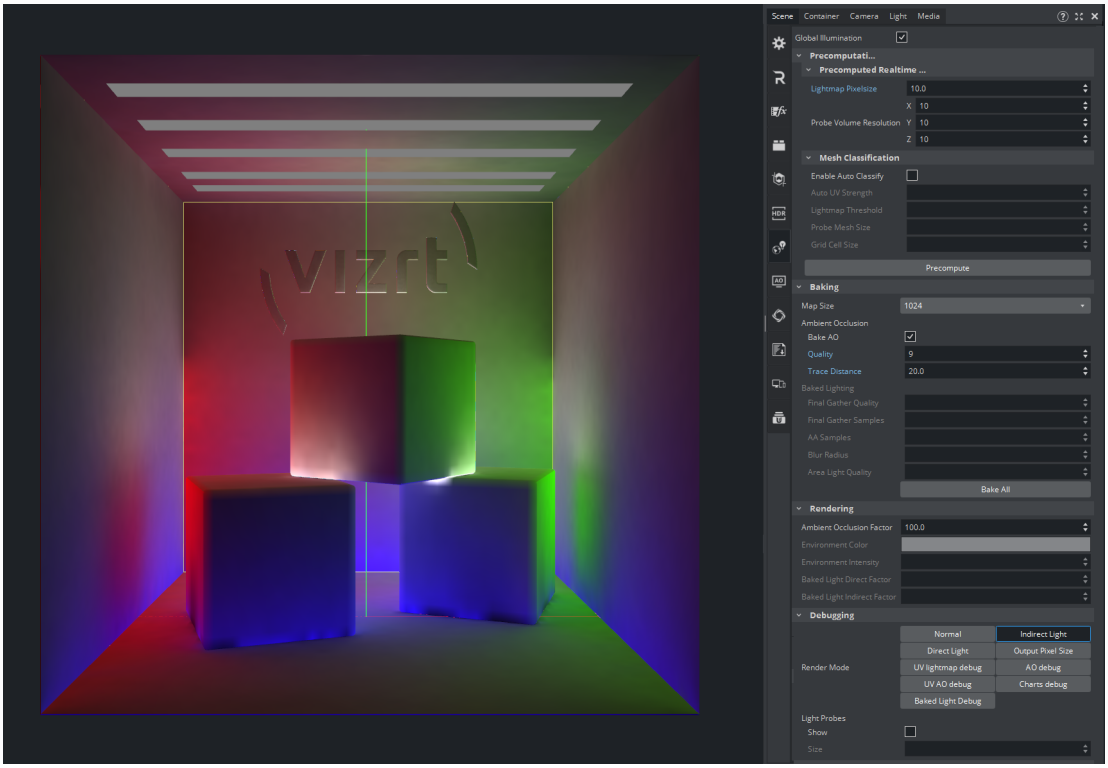




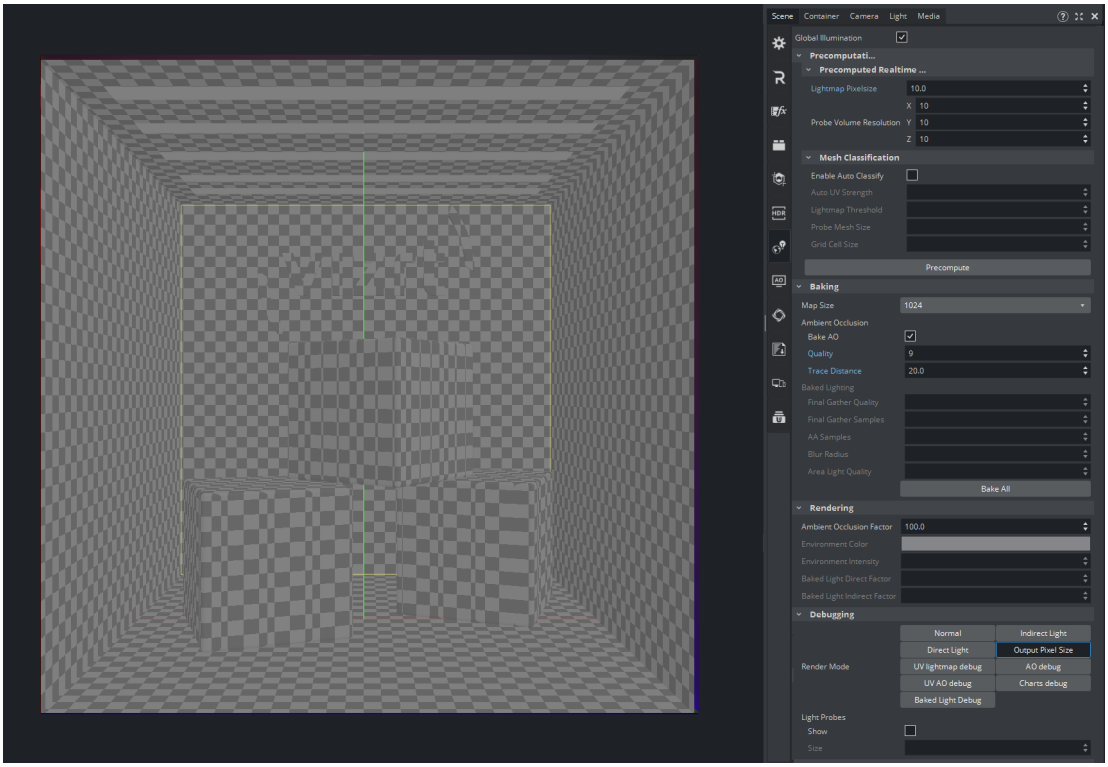


Example: In this example, there is no visible noise. The pixel size used in this scene is `10`.





Copyright © 2026 Vizrt



Probe Volume Resolution: Defines the number of probe samples in the scene in x-, y- and z axis. When activating the ***Show Probes*** feature you can see the amount of probes in your scene. When changing the amount of probes you need to recompute the Lightmaps again to see the changes.

Precompute: Pressing the **Precompute** button to generate all lightmaps. An overlay window appears, showing you detailed debug info. The first overlay window displays the scene analysis procedure. When this task is done, a second overlay window appears, displaying the baking procedures.

Note: You must precompute all maps again each time you add additional objects to your scene.

As individual objects cannot be computed on their own, precompute always recalculates the whole scene. If you have wrong UVs, or any other issues, for example with the pixel size, a window with error messages appears. You can read the detailed error messages and computation information in the log-file. The error messages are available in the scene info menu, which turns red in the case of any computation issues.

Example Scene Info Error Messages:

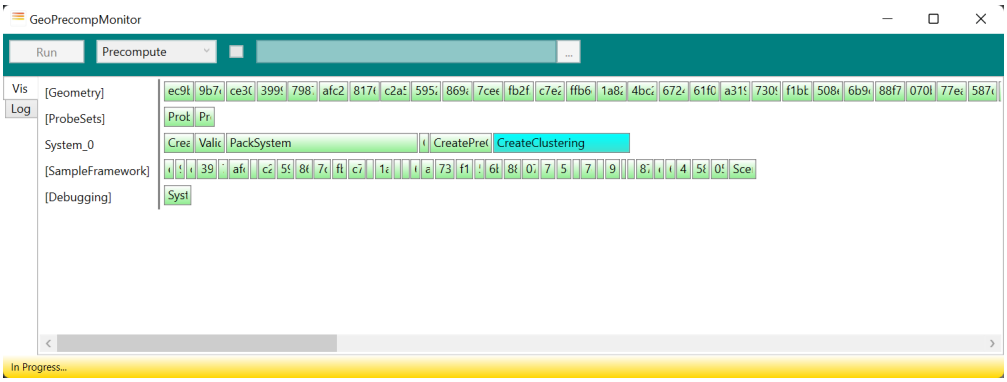
```
Global Illumination Save As: Warning: Save As does not include Global Illumination Precompute data. Please rerun Global Illumination Precompute or use Scene duplicate.
Global Illumination: The mesh on container /CornellBox/RootNode/cornell_box/VizrtLogo/Group1/bevelPolygon16 has non-conforming UVs preventing a successful AO bake
Global Illumination: The mesh on container /TV_Studio/RootNode/table/Table_1/Table/leg has non-conforming UVs preventing a successful AO bake
```

Copyright © 2026 Vizrt

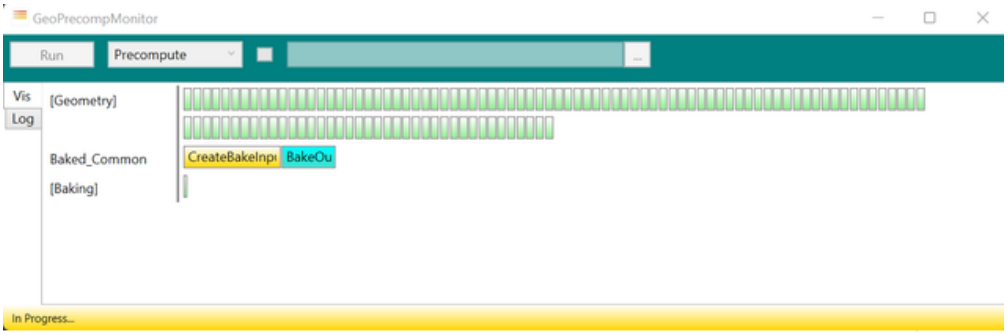
```
Warning: Input mesh '3999a322e55a4199a645027d441f14b2_AutoUv' in geometry
00000000000000000000000060adaa3f has 1 chart(s) that self-overlap; this
is most likely due to a reflection in the UV mapping.
```

Precompute Computation Examples

1: The Geo Precompute Monitor during scene analysis and UV preparation.



2: The Geo Precompute Monitor during the lightmap baking process.



Mesh Classification

Information: These settings are for the Classic Render Pipeline only!

This is the most important setting of Global Illumination. Mesh classification automatically classifies meshes for Lightmaps and light probes, and creates the necessary UVs for them. If Mesh Classification is turned `Off`, only the UVs from the second UV channel is used to generate the Lightmaps. If you set it to *Auto*, Mesh Classification generates the UVs for the lightmaps from your first UV set of your geometries/meshes. The computation of Global Illumination is faster when a second UV set is available, as analysis of the geometries and generation of additional UVs is not required.

Mesh Classification can be adjusted with the following parameters:

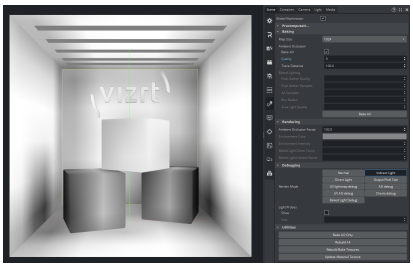
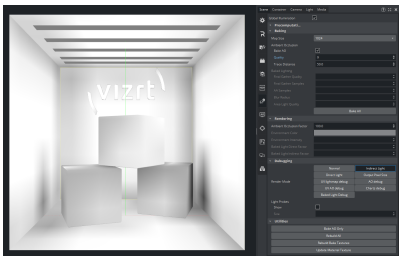
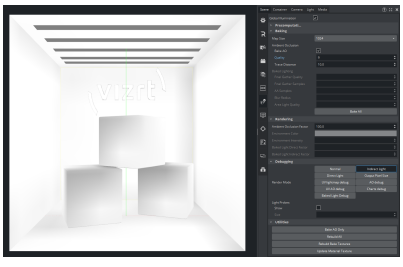
Copyright © 2026 Vizrt

- **Auto UV Strength:** Controls how aggressively the Auto UV process flattens the meshes onto planes. Lower values lead to better UVs with less overlap. Thus, more meshes are classified as lightmap-lit, at the expense of larger UV space requirement. Higher values result in worse UVs, with more meshes classified as probe lit, and lower UV space usage.
- **Lightmap Threshold:** Controls the threshold where Auto UVs are considered bad enough that a mesh should be probe-lit instead of lightmap-lit. Lower values increase the number of meshes probe-lit, and higher values increase the number of meshes that are lightmap-lit.
- **Probe Mesh Size:** Controls how big a probe-lit mesh and its surroundings have to be before it becomes a Contributing Probe.
- **Grid Cell Size:** Splits up the scene into systems of cubical cells of this dimension in World units.

Ambient Occlusion (AO)

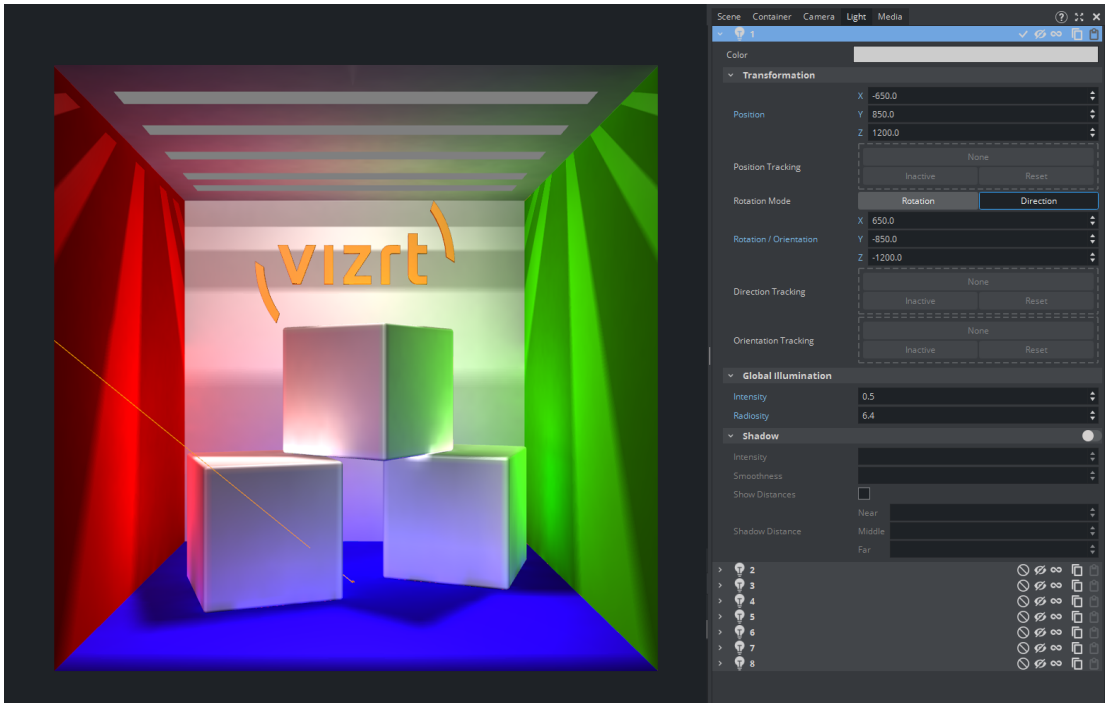
- **Bake AO:** Recalculates the AO maps for every bake process (precomputation) when set to **active**. When Bake AO is set to `Off`, only the indirect light maps are calculated when you press the **Precompute** button. Enable **Bake AO only** to recalculate the AO maps only. This is useful when you only to play with the AO without adjusting your Global Illumination parameters, as it reduces the computation time by only calculating the AO maps. For best results, always run a full computation process, as the GI and AO maps do not work properly together if computed with differences. This can happen if you change or delete objects in the scene.
- **Map Size*:*** Specifies the size of your AO maps for the scene, ranging from `512` pixels up to `16K`. The default setting is `512` pixels.
- **Quality:** Increases or decreases the quality of your AO maps. Increasing quality also increases calculation time. The quality setting has a range from `4` to `17`. The default setting is `8`.
- **Factor:** Changes the factor to influence the strength/blending of the ambient occlusion. You can change the factor at any time without recomputing the meshes/scene.
- **Trace Distance*:*** Sets the distance where a point shall be considered occluded. When you change this value, you cannot see the results in real time, as you must recompute the AO maps for the changes to take effect. Usually, the double value of the Lightmaps pixel size is used here; if your lightmap pixel size is set to `50`, you should have a value of `100` in the Ambient Occlusion Trace Distance field. Since this comes down to the individual designer's style, use any value you like here.

The following images show some examples:

Trace Distance: 100	Trace distance: 50	Trace Distance: 10
		

Copyright © 2026 Vizrt

Finally, you can now play with the **Radiosity** setting in the Light Editor:



The input controls for Global Illumination are only visible in scenes where Global Illumination is set to `Active`.

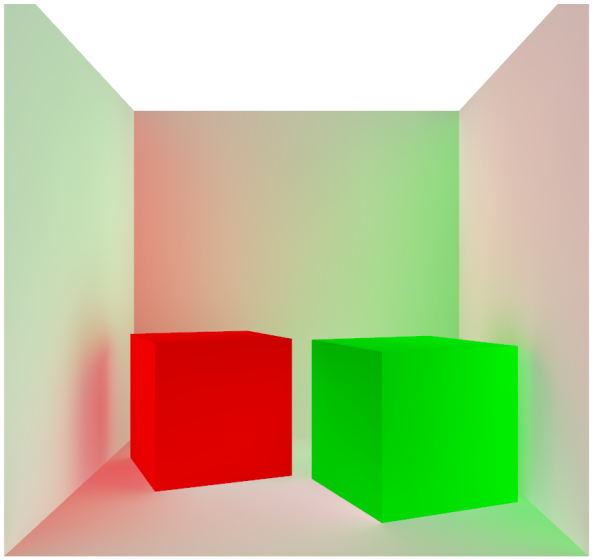
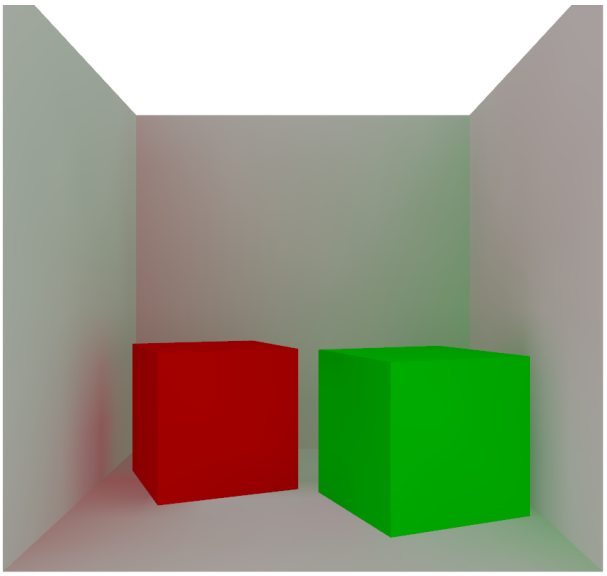
Render Mode

Available settings: *Normal Mode* / *Indirect light* / *Direct Light* / *Output Pixel size* / *UV lightmap debug* / *AO debug* / *UV AO debug* / *Charts debug*

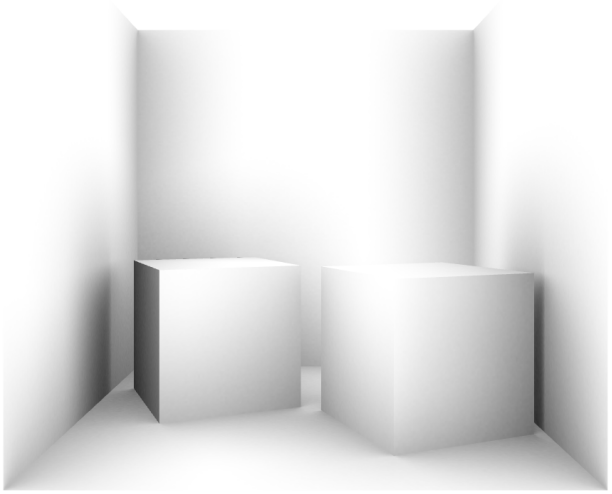
These are different visual modes for debugging the different render passes: the **Ambient Occlusion** render pass, the **Indirect Light** render pass, and the **Direct Light** render pass. You can also debug the **Pixel Size**, which controls the quality of the Lightmaps, or view the UVs for the Lightmaps, Ambient Occlusion, or the UV Charts. *Normal Mode* presents all maps together as a final result.

Copyright © 2026 Vizrt

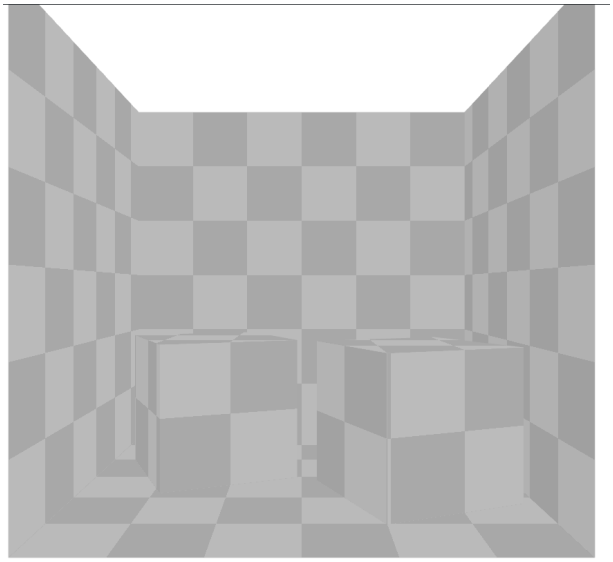
Render Mode Examples for Global Illumination

Render Mode	Visual Example
Global Illumination Normal Mode This view represents the final output, with direct light, indirect light and ambient occlusion.	
Global Illumination Indirect Light This view represents the indirect light (Global Illumination) only.	

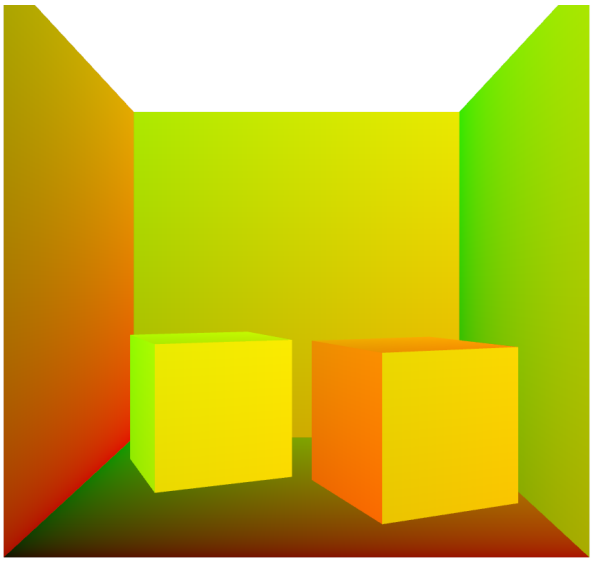
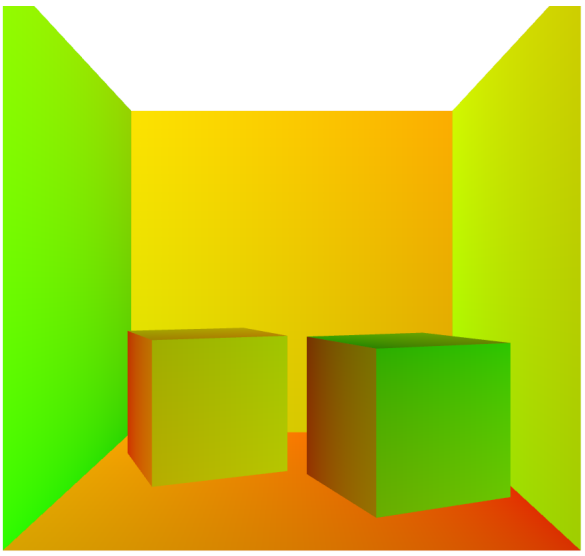
Copyright © 2026 Vizrt

Render Mode	Visual Example
Global Illumination Ambient Occlusion This view represents baked Ambient Occlusion map only.	

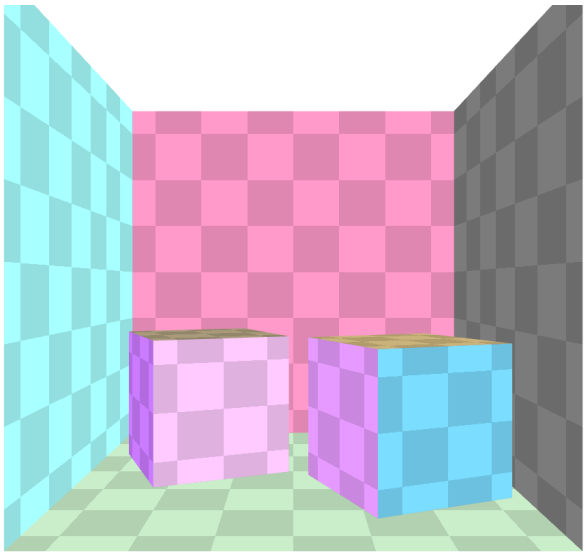
Debug Modes for Global Illumination

Debug Mode	Visual Example
Global Illumination Pixel Size This view shows the output of the Lightmaps in pixel size.	

Copyright © 2026 Vizrt

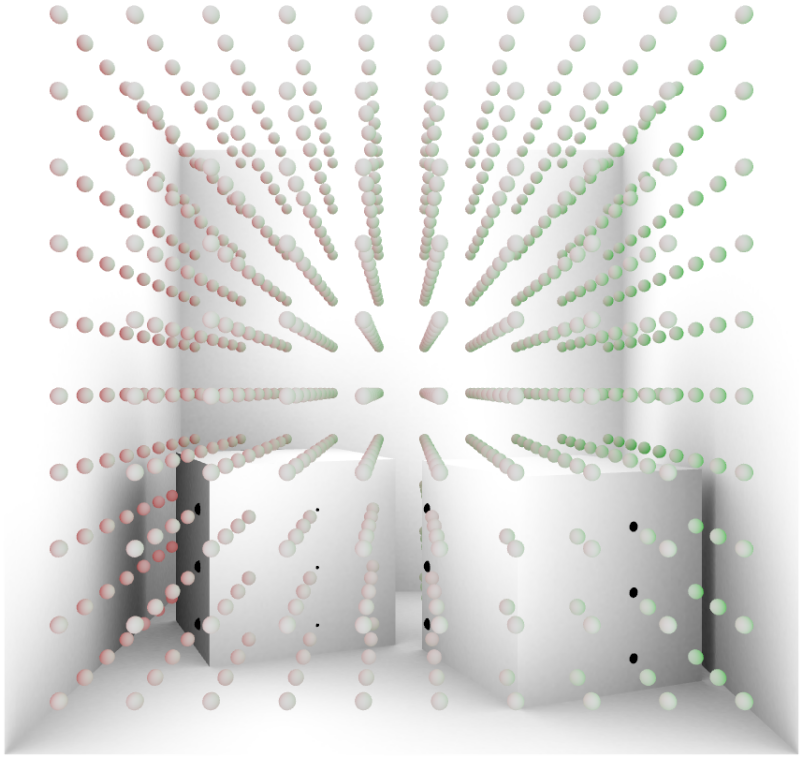
Debug Mode	Visual Example
Global Illumination UV Lightmap Debug Mode This view shows the lightmap UVs only.	
Global Illumination UV AO Debug This view shows the ambient occlusion UVs only.	

Copyright © 2026 Vizrt

Debug Mode	Visual Example
Global Illumination Charts Debug This view shows the individual UV charts.	

Show Probes

Enable or disable debug view for the light probes inside your set. The light probes provides you with a visual representation of how light is distributed within your scene. The **Size** parameter adjusts the size of the light probes in the viewport, but does not influence computation of the Lightmaps or light probes themselves.



Copyright © 2026 Vizrt

Be aware that not every object can be lit by lightmaps, due to the complexity of the UVs. This applies especially to organic shapes, for example, a statue. Objects that cannot be lit by lightmaps are probe-lit.

Creating a Scene with Global Illumination

First, you need to import a geometry or scene with proper UVs and normals. This can be either just one UV set with unwrapped UV coordinates to be able to do a proper texture mapping and extract the lightmaps from this UV set. You can also import an *.fbx*-file containing the second UV set, with a layout for lightmaps created in your preferred 3D application. On your imported geometries make sure the normal orientation is facing outwards, flipping the normals with the Expert plug-in leads to wrong computation results. Do not forget to turn **Auto Mesh Classification** on or off, depending on your existing UV sets.

Next, if your scene is not already modeled or imported with the proper size, you need to set up the real world size of your scene before you do the baking. Then place some lights in your scene. After this, adjust all the parameters described above and run a test bake, to see if you get any error messages concerning UVs or baking. If your render passes look good from the debug views, no error messages appeared, or at least no visual errors are visible, you can start working on the detailed light setup.

All the light setups, color, type, transformation, can be done at any time now, without a need to recompute the whole scene. This is the benefit of working with lightmaps. It is very important to increase the radiosity setting of the lights that light up the whole scenery, using the Light Editor. Be aware that Lightmaps are static maps that react in real time to the GI parameters. They are not dynamically calculated with animations. When you have an object, like the Cornell box example above, and animate the rotation of one of the cubes inside the Cornell box, you see the wrong colors reflected during the rotation. This is because the lightmaps are statically projected onto geometries. When rotating the middle cube of the Cornell Box example, which is receiving a green color from the green wall next to it, it does not turn red when rotating to the other side, because lightmaps are static. The same applies to Ambient Occlusion maps. Whenever you want to animate an object, you should not compute the lightmaps or Ambient Occlusion maps for it, as otherwise the result does not look correct. If an object is hidden in the Scene Tree, it is not calculated with GI.

After setting up all the parameters and running a successful precomputation process, switch to the **Indirect Light** debug view from the **Global Illumination** pane, to see how the radiosity intensity of the light affect your scene. Experiment with different lights and radiosity settings to get your desired result. During this process, you should switch between the debug view for **Indirect Light** and **Normal Mode**, to see how much GI affects in your scene. Now, you can start refining your scene with adding Substance Shaders to your scene to increase realism. Compared to using regular plain textures, substance's light reflection is aiming a physical correct behavior depending on the shader properties. This is something a regular texture does not do.

If you want to work with plain textures only, this is possible. Be aware that no other material and effect Shaders than Substance Shaders are supported in combination with Global Illumination.

To get your light color correctly spread into your scene, you need to apply a material to every geometry and use the eye dropper tool of the material editor to get the correct color from the texture or shader used on the selected geometry. Otherwise, radiosity cannot blend the surface color into the set, since it is not possible to extract the color directly from textures or Shaders. Applying a material with color does not influence the Substance Shader, as this material color is only used for the color bleeding of the radiosity.

Limitations

Global Illumination is built to work in conjunction with Physically Based Rendering (PBR) materials, to get the most realistic results. In general, it is common to use Global Illumination in conjunction with Substance Shaders. Substance Shaders are the only available PBR Shaders for Viz Artist. Therefore, any other material shader, such as RTT and others, does not work with Global Illumination. When using a feature which is not supported by Global

Copyright © 2026 Vizrt

Illumination, you get an error message in the scene info. Error messages in the scene info are indicated by a red info icon.

The following list are examples of features and/or plug-ins that are not supported in combination with Global Illumination:

- Instancing
- RTT Shaders
- Softmask
- Framemask
- ImageMask
- Fluid Effect
- Water Shader
- Drop Shadow
- Emboss
- Anisotropic Light
- Cartoon Shader
- Gooch Shader
- Lighting Shader
- Normal Map
- Filter Blend
- Filter Blur
- Filter Color Balance
- Filter Radial Blur
- Filter Sepia
- Filter Sharpening
- HDR

Copyright © 2026 Vizrt

11.3.8 Classic SSAO

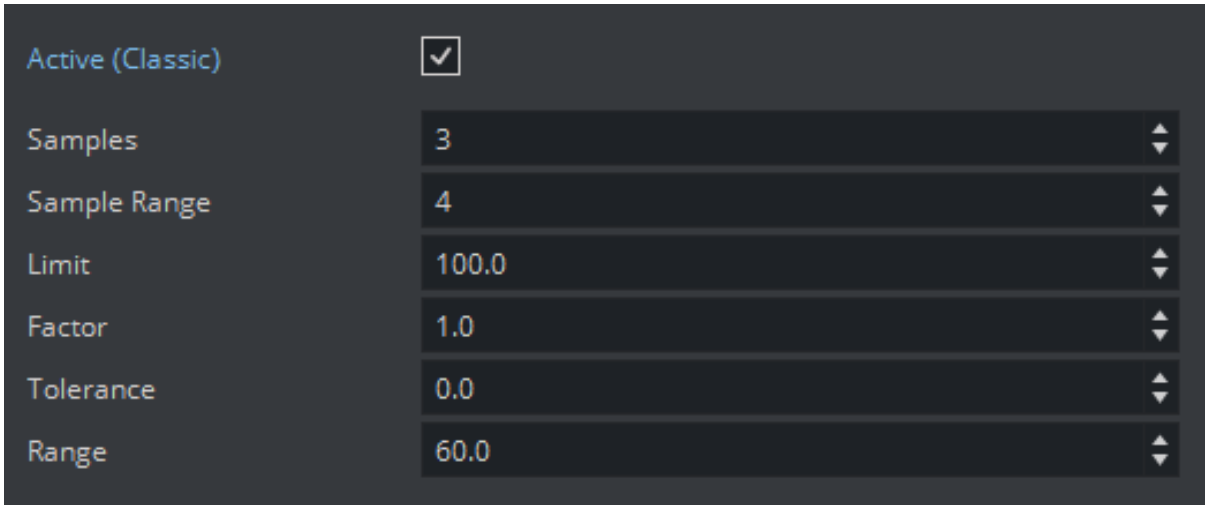


Note: This information is for the Classic Render Pipeline only!

Screen space ambient occlusion (SSAO) is a computer graphics technique for efficiently approximating the ambient occlusion effect in real time. SSAO is mainly used for virtual sets, and for augmented reality objects and scenes. In regular broadcast graphics, for example a lower third, SSAO is hard to notice. This is because the details in these types of graphics are too small for the effect to be visible.

To activate SSAO for your scene, or adjust the settings, open Scene Settings and select the **Screen Space AO** tab. SSAO is applied to the final rendering image. This means that when SSAO is active, all objects in your scene are rendered with SSAO.

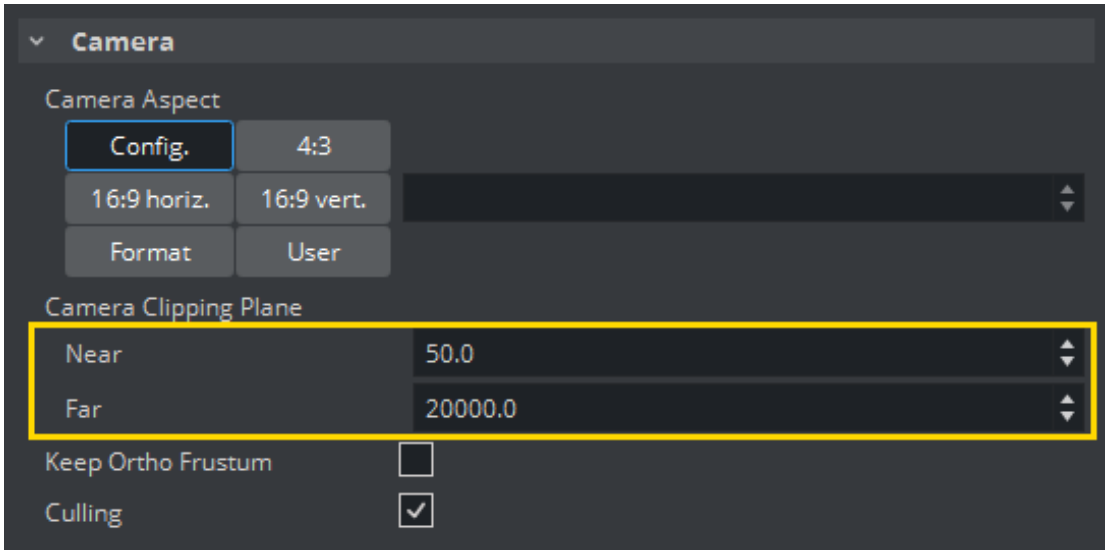
Adjusting the settings may impact performance, depending on the parameters changed: Increasing the **Samples** and **Sample Range** values increases the load on the system GPU for SSAO computation, while improving the result. Changes to the **Limit**, **Factor**, **Tolerance** and **Range** settings affect the visual look of the SSAO, without impact on performance. **Limit**, **Factor**, **Tolerance** and **Range** influence the SSAO grayscale.



To see the result in your Render view, press the button to activate the Post Processing effects. If you are loading scenes into Viz Engine via an external control application, the SSAO is visible both on the final output and the VGA preview.

Tip: To get the best result with SSAO, decrease the **Far** value for **Camera Clipping Plane** value to the maximum depth of your scene. By default, Camera Clipping Plane far is set to `20000.0`.

Copyright © 2026 Vizrt



The following examples show the difference between scenes rendered with and without SSAO. SSAO gives the object a more realistic look. By changing the parameters of the settings, you can control the quality, detail and render performance of the SSAO. The objects in the examples provided below have the same light setting, and cast no shadows. The only difference is the Screen Space Ambient Occlusion.

Copyright © 2026 Vizrt

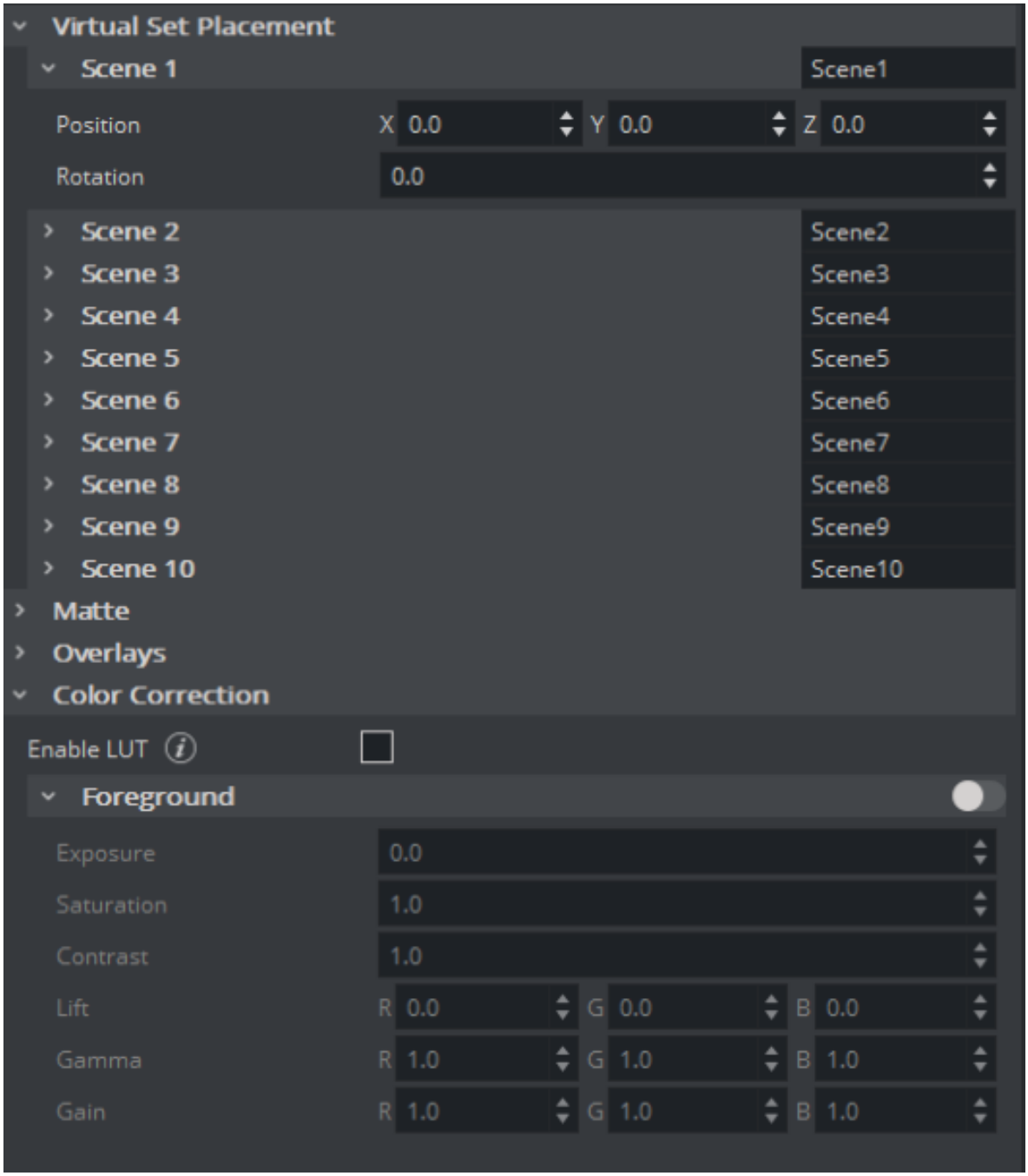
11.3.9 Virtual Studio Panel



The Virtual Set panel is only available if the Virtual Set option is installed. This panel makes it possible to set up and save configurations for virtual sets.

For information about the parameters, see the Viz Virtual Studio User Guide.

Copyright © 2026 Vizrt



- **Scene:** Sets the camera view offset in the selected Scene.
- **Virtual Blue Screen:** Check and view settings:
  - **Hold-out Matte:** Allows chroma keying and color correction only in the cyc area (as set in Viz Virtual Studio) if set to `On`. The trash matte is not applied.
  - **Display Mode:** Sets the Trash Matte visibility.
  - **Draw cyc/co-cyc:** Selects the Trash Matte.
  - **Sample Geometry:** Sends sample cyc/co-cyc geometry to the Viz Engine.
- **Overlays**

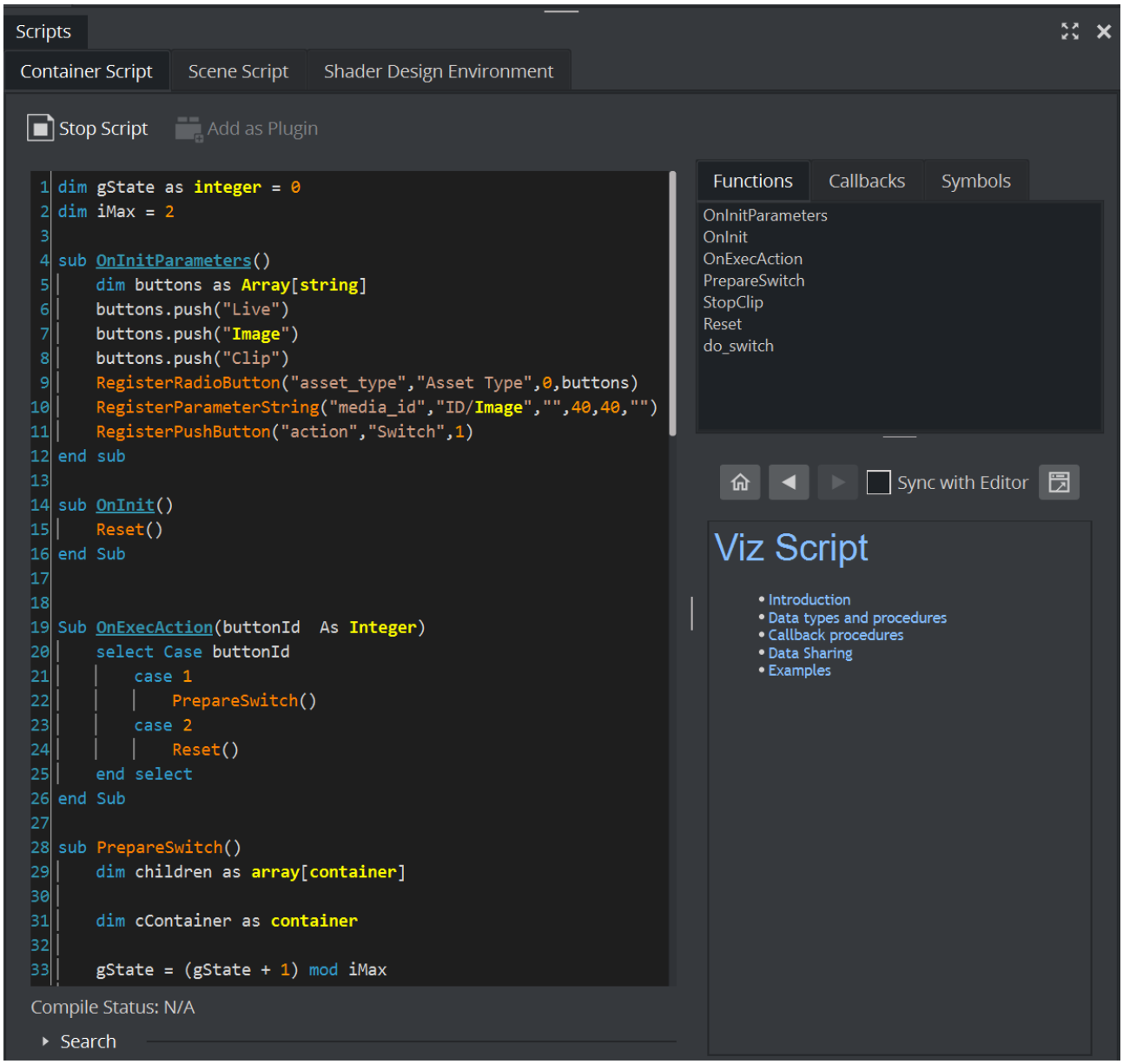
Copyright © 2026 Vizrt

    - **Head Up Display:** Shows either the **Camera** parameters or the **Chroma Keyer** head up display in the Scene Editor when set to `On`. Removes the Head Up Display when set to `Off`.
    - **Center Shift Cross:** Shows or hides the Center Shift Cross in the Scene Editor.
- **Color Correction:** Adding color correction options for XR use cases.
    - **Enable LUT:** activates the support of external Cube files for color correction, sent from Viz Tracking Hub. This is mainly used to tint an AR engine to match a Videowall.
    - **Foreground**: When using the Multi-Layer External XR this feature allows manual color adjustements of the Foregound Image. This only works in combination with an active LUT file.

Copyright © 2026 Vizrt

11.3.10 Script Panel

In the Scene Script panel, you can add and edit scripts which are part of a Scene (see Script Editor). Those scripts act as a global script responding for the whole scene whereas Container Script usually only act on Container Level.



To Enable the Scene Script Editor

1. Click on the Workspace ( ) button.
2. Scroll down to **List of available Docks**.
3. Select the **Scripts** option.

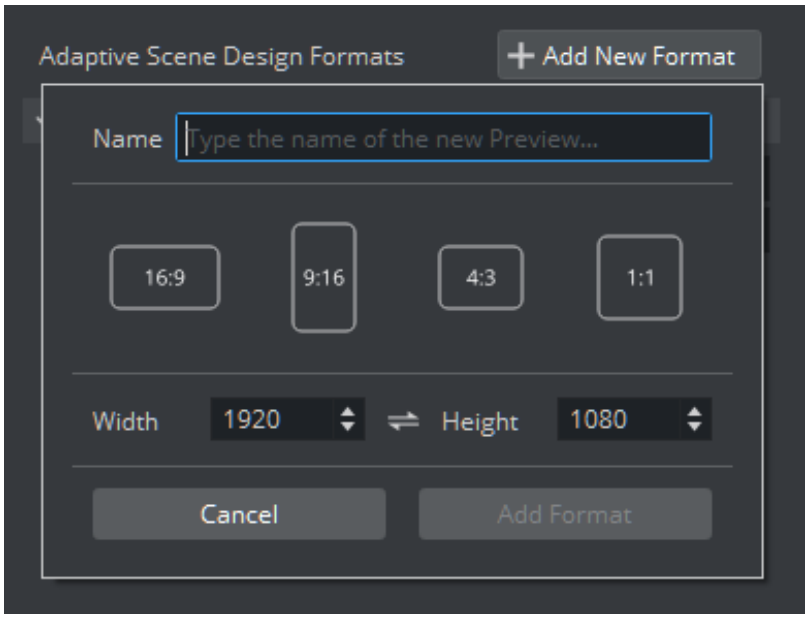
Copyright © 2026 Vizrt

11.3.11 Adaptive Scene Design Formats

Each scene comes with a default format. This is based on the configured output size (for example, 1920x1080 with an aspect ratio of 1.77) for HD resolution. This is the base and default for all properties. In addition to this format, many additional formats can be added to the scene. These formats can vary in dimension, but do not necessarily. For example, a scene can hold three different HD formats, all in the 1920x1080 dimension, but with different logic and looks.

To add a new format, select **Scene** and click on the **Formats** icon .

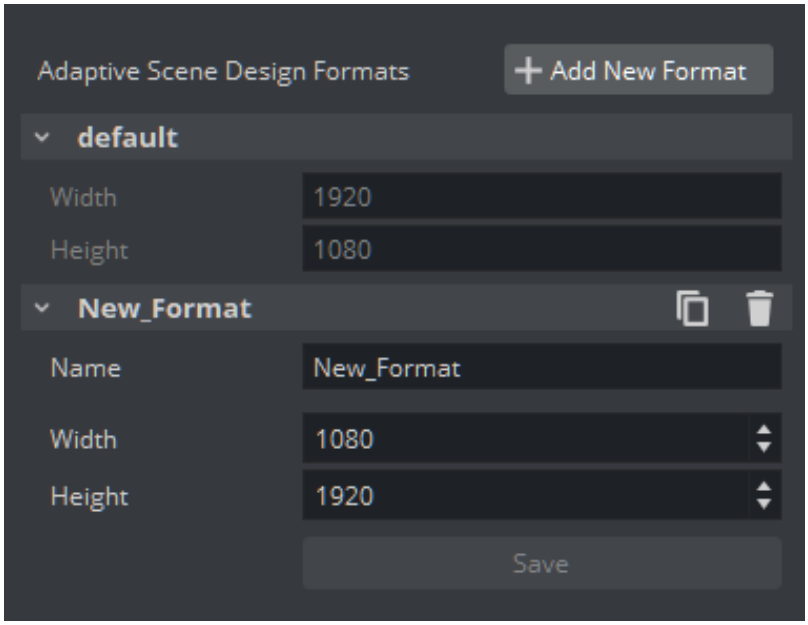
The default format is already listed and can not be deleted. To add a new format, click on the button:



Give your format a unique name. Viz Artist versions 5.0 and newer come with presets for the most common aspect ratios. This feature will be extended in future versions to serve more presets and let the user create own profiles. The *Width* and *Height* values are taken automatically from the output system, but can be changed to any other values or switched by clicking on the desired button. The most important factor is the aspect ratio, that is calculated automatically internally and used afterwards by all required plug-ins like the Flexbox plug-in.

Once set, click on the **Add Format** button. Your new format is now listed as available. The values are inherited from the default format.

Copyright © 2026 Vizrt



Any format name, width or height can be changed afterwards. Change the property and press the **Save** button. Each format can also be cloned or deleted afterwards. Once your scene has an additional format defined, it can be used to create and preview your adaptive graphics.

- **:** Clones an existing format.
- **:** Deletes an existing format.

The default format cannot be changed.

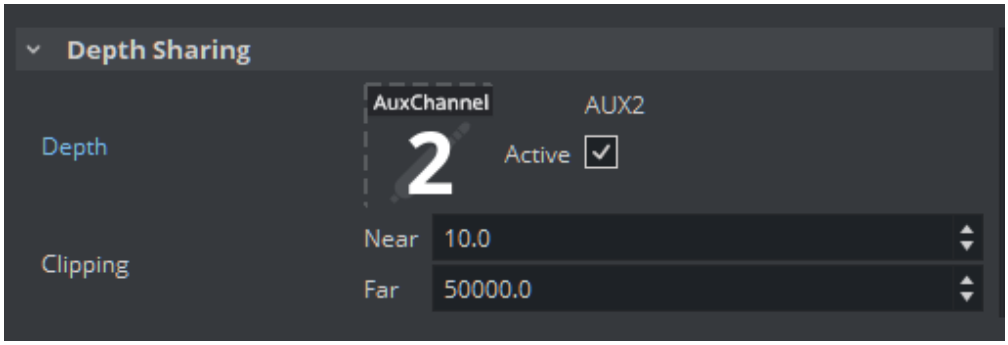
Copyright © 2026 Vizrt

11.3.12 Unreal Scene Settings

Depth Sharing

To receive depth information from the Unreal output, drag the corresponding shared asset to the **Depth** slot. By default, the VizrtMedia plug-in stores the Unreal depth information in the AUX2 asset.

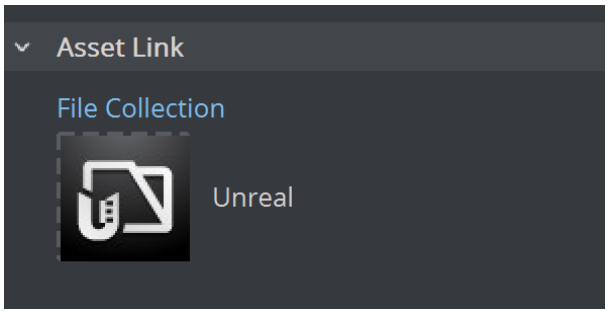
It is also possible to control the Unreal clipping planes position by changing the distance from the camera for the **Near** and **Far** planes:



Info: Please check the Mixing Depth Buffers page for more information about the depth sharing between Unreal and Viz Engine.

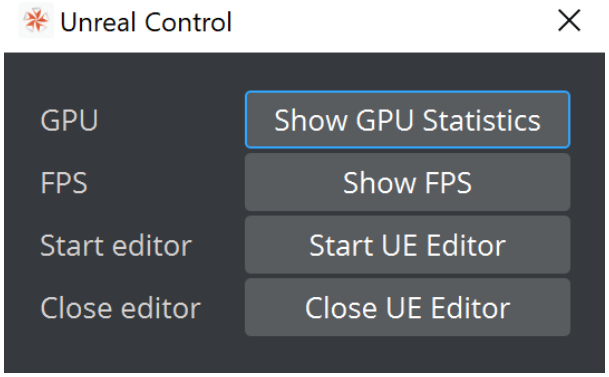
Asset Link

We can associate an Unreal project file collection with a Viz scene by dragging it to the **File Collection** slot. In the image below, a file collection named **Unreal** was dragged to the slot:

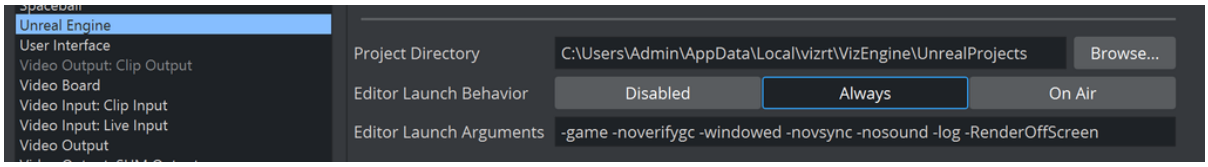


The file collection (Unreal project) is automatically restored from Graphic Hub when the scene is opened and then loaded by Unreal when Unreal is launched from Viz Engine. To launch Unreal manually from Viz, use the **Unreal Control** window on **On Air** mode:

Copyright © 2026 Vizrt



To launch Unreal automatically when loading the scene or going On Air, set the desired **Editor Launch Behavior** option in **Config**:



Info: Please check the Working with File Collections page for more information about file collections in general, and Synchronization with Graphic Hub for information about how to use file collections with Unreal.

Copyright © 2026 Vizrt

11.4 Scene Editor

The Scene Editor has a set of useful functions that lets the designer see how the graphics look when keyed, seen from a different angle in Viz Virtual Studio (virtual sets), positioned on-screen, performance of the scene and more.



On the left side of the Scene Editor are the Scene Editor Buttons. These buttons make it possible to show additional information in the Scene Editor, as well as providing more options while working.

For a better indication of one interacting with the Scene Editor, a small light blue border is shown while interacting with the Scene Editor.

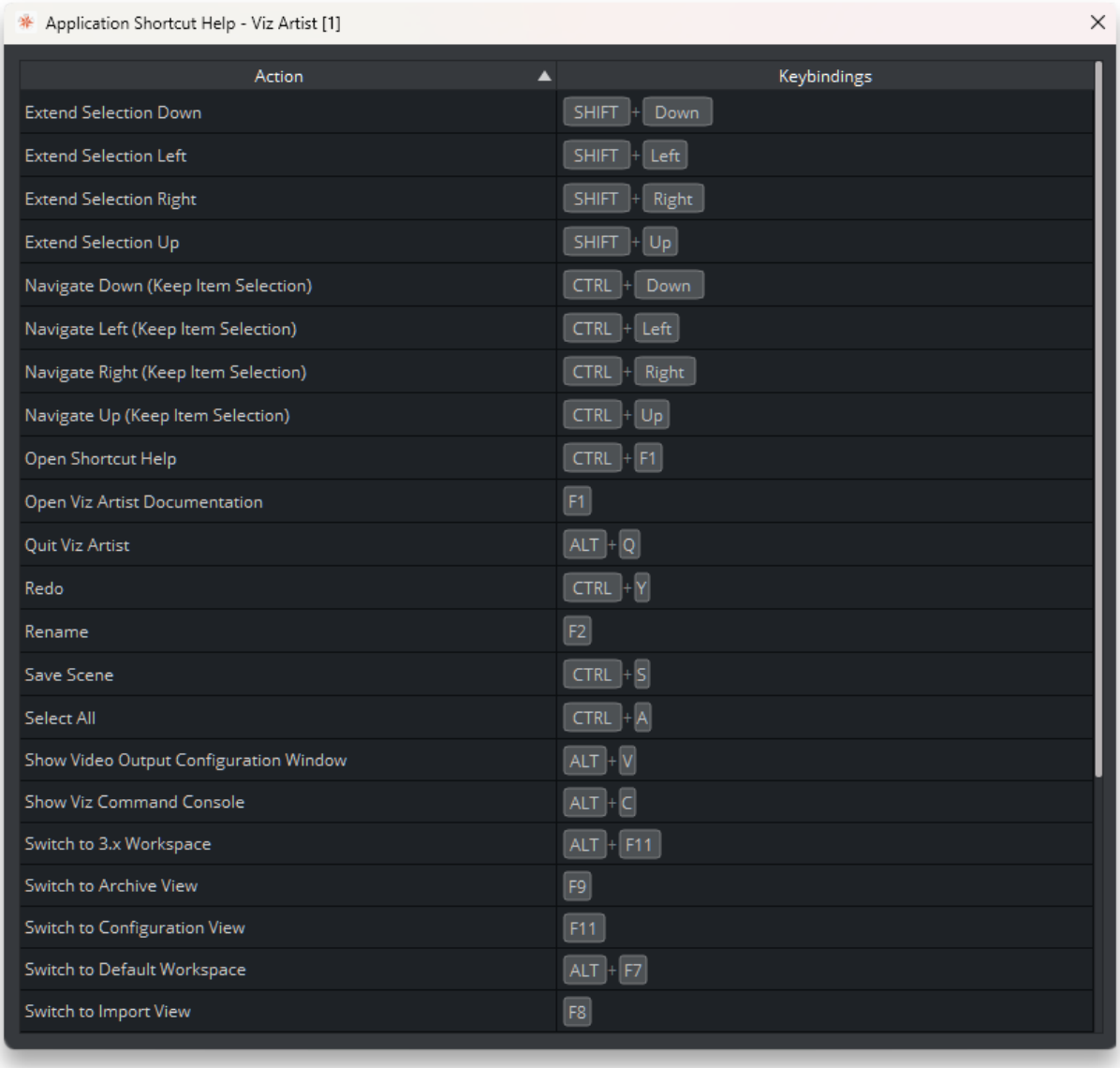
This section contains information on the following topics:

- Grid Tool-bar
- Performance Bar
- Scene Editor Buttons
- Scene Editor Snapshot

11.4.1 Shortcuts

To show a list of available shortcuts, press **CTRL + F1**.

Copyright © 2026 Vizrt



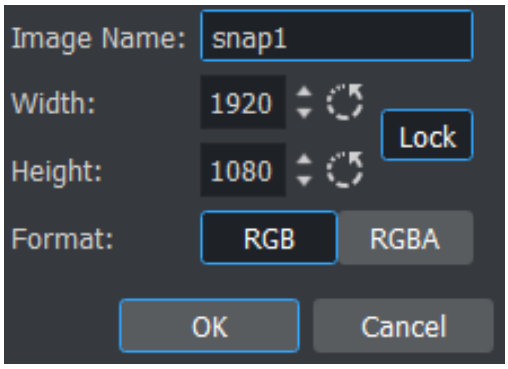
Note: Starting with Viz Artist version 5.1, Shortcuts only work if the Scene Editor has focus. This is indicted by a small blue rectangle.

Copyright © 2026 Vizrt

11.4.2 Scene Editor Snapshot

A snapshot can be taken of the current content in the Scene Editor.

When clicking the Snap button, the Server Panel is automatically selected from the main menu, and the database content shows. The image is placed in the folder/project that was last visited.



To Take a Snapshot

1. Click the **Snap** button.
2. In the Snapshot dialog box that shows, enter the **name** to save the snapshot as.
3. Select the **size** and **format** for the image.
4. Click **OK** to save the image to the database.

> **Note:** An RGBA image uses the key information as alpha.

> **Note:** To save the image in another folder/project: First select the destination you want to use, then click the **Snap** button again.

Taking snapshot from a running video is allowed. The procedure above is interactive. If required, you can alternatively instruct Viz to take a snapshot using the commands below. The examples below, calling *renderer* directly, assumes the Engine is in On Air mode:

```
RENDERER*VIDEO*SNAPSHOT CREATE file:///c:/images/snapshot.png
RENDERER*VIDEO*SNAPSHOT CREATE vizgh:///my/images/snapshot
RENDERER*VIDEO*SNAPSHOT CREATE vizgh:///{159FC7E7-B724-4CC3-B474B7EE46CDA043}/
snapshot
```

- Example 1 writes a *.png* image to disk. The *.png* file format is particularly useful when working with RGBA images or four channel data in general.
- Example 2 saves the snapshot to the Graphic Hub (GH) using a path.
- Example 3 saves the snapshot to the GH using a GH ID (unique identifier, the UUID must already exist).

See Also

Copyright © 2026 Vizrt

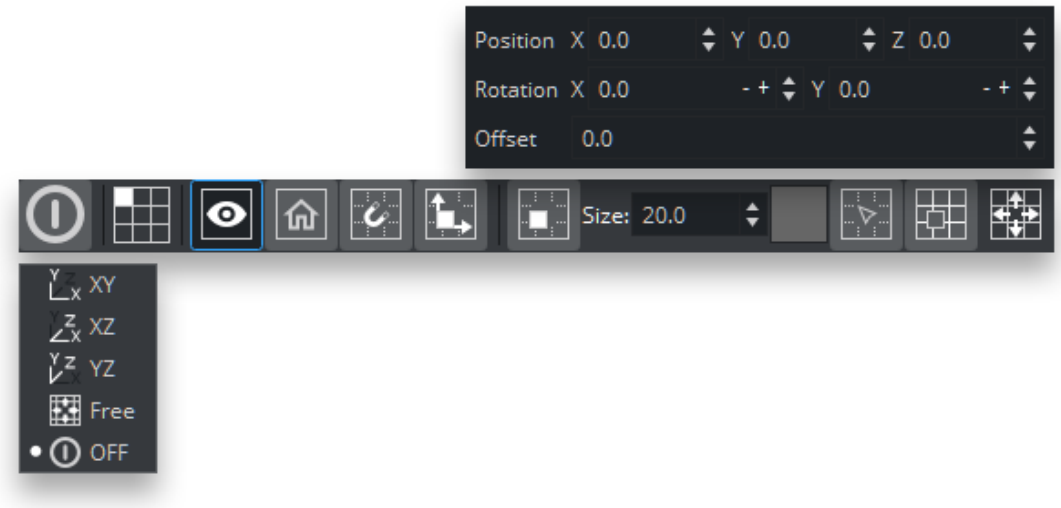
- Scene Editor Buttons
- Performance Bar
- Grid Tool-bar

Copyright © 2026 Vizrt

11.4.3 Grid Tool-bar

The Grid tool-bar can be shown in the lower part of the Scene Editor. Grids are used to align containers. If the **Grid Editor** is closed while a grid is still active, the Grid button turns orange. This is especially useful when the grid is active, but hidden in the Scene Editor using Show/Hide Grid. All grid settings are saved with the scene; thus any such settings are reloaded the next time the scene is loaded.

Click the Scene Editor's **Grid** button to open the Grid Toolbar:



- **Plane Type:** Shows which plane the grid should lie in:
  - **Align X/Y:** Aligns the grid to the X/Y-plane.
  - **Align X/Z:** Aligns the grid to the X/Z-plane.
  - **Align Y/Z:** Aligns the grid to the Y/Z-plane.
  - **Free Grid:** Makes it possible to use a free grid, aligned for example to an object.
  - **Off:** Shows that the grid is set to `OFF`.
- **Select Grid:** Sets a predefined grid. Nine grids can be selected and manually configured as presets for the scene.
- **Show/Hide Grid:** Shows or hides the grid in the Scene Editor.
- **Home:** Sets the grid back to its default position.
- **Snap to Grid:** If this button is enabled and the position of a container is modified by dragging it around in the Scene Editor, the position of the object snaps to the grid based on the **Axis Center** defined for the object's container.
- **Align Object on Plane:** To align the axes of a container to the corresponding axes of the grid, first enable this option and next left-click the container you want to align and drag it around. As long as this option is switched on, the container moves only along the plane if you drag it around in the Scene Editor.
- **Move Object to Plane:** If you select a container and next click this icon, the container is moved so its center is on the plane.
- **Size:** Defines the raster size of the grid.

Copyright © 2026 Vizrt

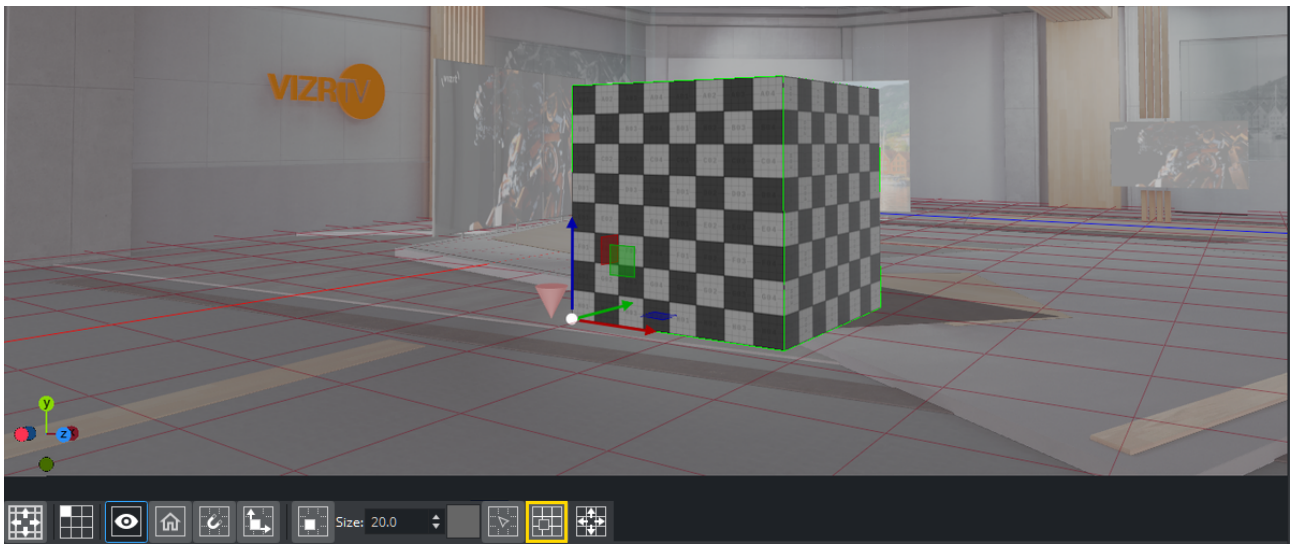
- **Change Color of Grid:** Changes the color of the grid. The three buttons detailed below are only available if the grid type is set to Free Grid:
- **Picking On/Off:** Aligns the grid to the object. Enable picking and click the object to align the grid. The grid is aligned to the vertex you click on.
- **Move Grid to Object:** Moves the grid so the center of the selected container is on it.
- **Free Plane Value Editor:** Switches the free plane value editor on and off. In this editor you can adjust the grid's plane as illustrated in the image of the Free Plane Editor options below:

  Adjusting the **Position** or **Rotation** in the Free Plane Editor changes the grid's own Position (x, y, z) or Rotation (x, y). Changing the Offset adjusts the grid's location along the normal vector of the grid XY plane (Z position).

Tip: The offset of the grid can be adjusted by pressing **ALT + G + Left Mouse**.

The grid can be used to align objects in a virtual set to the floor. Select your floor in your scene, open the grid and click on **Move Grid to Object.**

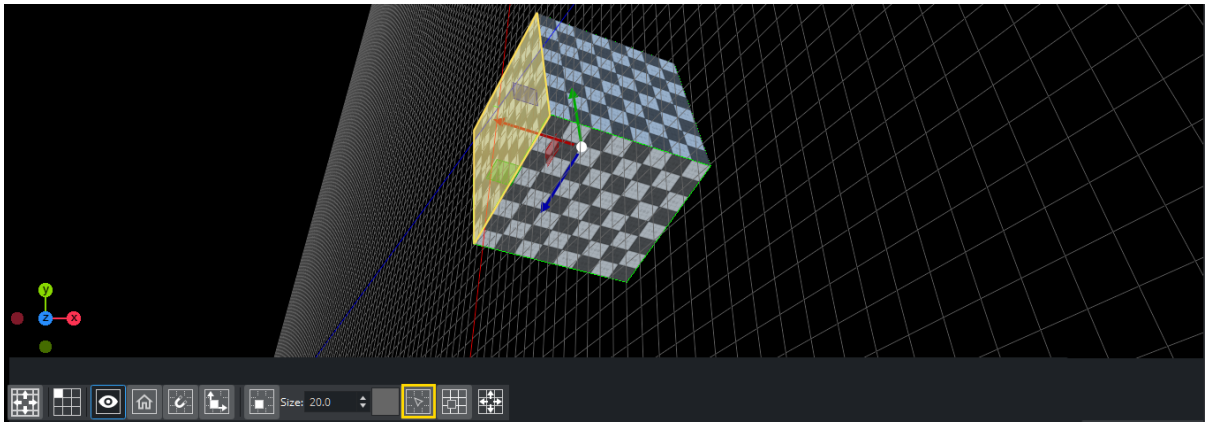
This aligns grid 1 to your floor object. If you now add new objects to your scene, you can click on **Align Object on Plane** and your object is aligned to the floor. If you also set a proper grid size and turn on **Snap to Grid** your object can be aligned very precisely.



You can add up to nine grids, one can be floor, the second one the wall, etc.

You can also align your grid to the side of an object (for example, a Cube). Create a new Free Grid, Turn on Picking and click on one of the side of your Cube.

Copyright © 2026 Vizrt



Grids can also be very useful when creating tables. Set your grid size accordingly to you row height and all your objects are aligned correctly.



See Also

- Scene Editor Buttons
- Snapshot
- Performance Bar
- Layer Manager
- Working with the Scene Editor
- Time-line Editor

Copyright © 2026 Vizrt

11.4.4 Performance Bar

This section also contains information on the following topics and procedures:

- Viz Engine Render Pipeline
- Classic Render Pipeline
- Performance Bar Properties
- To View All Performance Bar Indicators

The performance bars are different between the Viz Engine Render Pipeline and the Classic Render Pipeline. To show the Performance Bar, click in the Scene Editor.

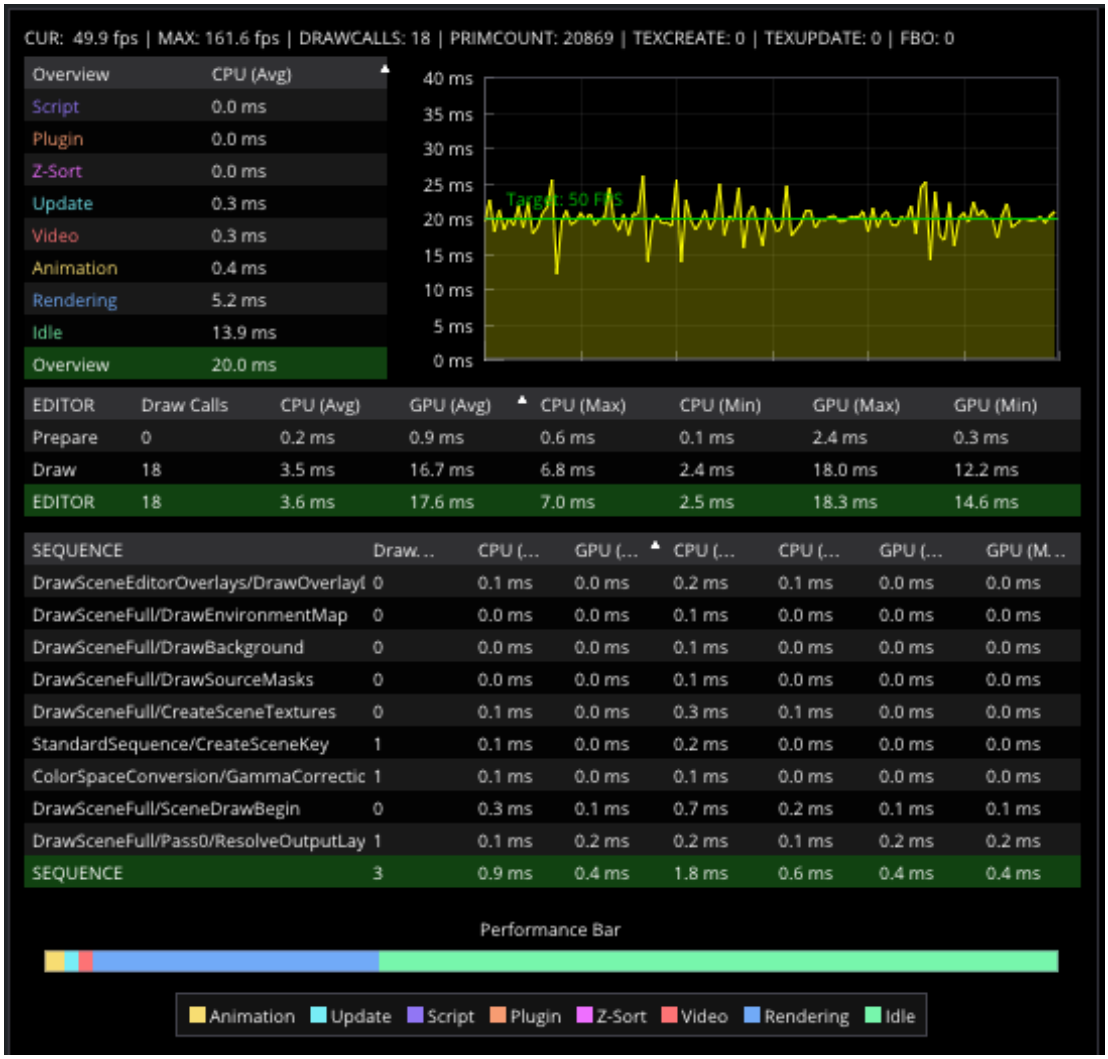
Do not use the maxfps as an indicator of your performance. The important factor is the current fps, this value needs to be stable on your output refresh rate (50 fps or 60fps). Rendering on 50Hz means that the whole operation of downloading live sources from Media Assets, rendering, composing and uploading to the video IO board must not take more than 25ms. The Performance Bar lists all those operations and gives you an indication on where to optimize your scene to run in real time.

When rendering on Viz Engine Render Pipeline, the Render sequence also allows an even more in depth analysis of your post processing effects.

Viz Engine Render Pipeline

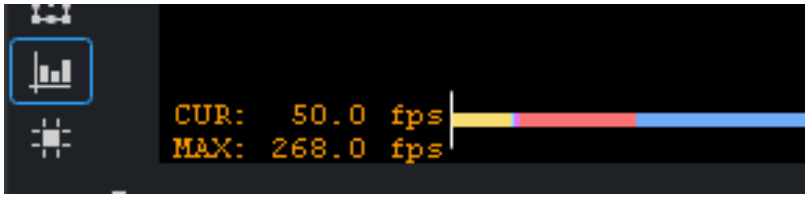
In the Viz Engine Render Pipeline, the Performance Bar shows a timeline for the current frames per second. This allows a user to identify performance issues (for example, when playing an animation). When rendering in real time, the line is green, anything else turns the line to red. The information shown on top is identical to the values in the Classic Render Pipeline.

Copyright © 2026 Vizrt



Classic Render Pipeline

A Performance Bar can be shown at the bottom of the Scene Editor. The Performance Bar gives an indication of the performance (frames per second) the Scene is rendering in.



Note: A Performance Bar can also be shown in On Air Mode (see Viz Engine Performance).

Copyright © 2026 Vizrt

Performance Bar Properties

- **VER (Vertices):** Shows the number of vectors in the scene.
- **TET (AllocTexSize):** Shows the total allocated size of Texture memory.
- **TEC (TexSize):** Shows the size of the current in use Texture memory.

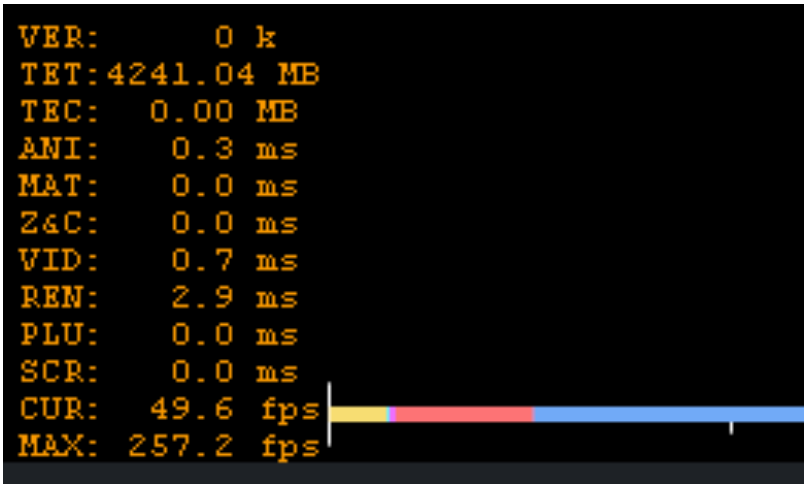
> **Note:** The actual drawing of Textures has an impact on the performance, more visible textures cause more render time. If texture memory is low it is possible that Textures are not visible in the Scene. The Texture memory usage is an indication for this.

- **ANI (Animation):** Shows how many milliseconds all active directors and animation channels take. This indicator is linked to the yellow bar.
- **MAT (Matrix):** Transforms each container in the scene into world coordinate space. This indicator is linked to the cyan bar.
- **Z&C (Z-Sort):** Refers to Z-sort and Culling, and sorts all containers for correct transparency drawing and determines if containers are visible in the current camera view. This indicator is linked to the pink bar.
- **VID (Video):** Shows how many milliseconds video input (live video Media Asset) and video output take. De-interlaced video inputs take longer time than progressive and interlaced. The only way to improve this value is to use a faster system. This indicator is linked to the red bar.
- **REN (Rendering):** Shows how many milliseconds it takes to render all objects on the screen. A faster graphics card improves this value. This indicator is linked to the blue bar.
- **PLU (plug-in):** Indicates how much time in milliseconds all active plug-ins spend in each render cycle. This indicator is linked to the orange bar.
- **SCR (Script):** Shows the consumed time in milliseconds from all active scripts. This indicator is linked to the dark green bar.
- **CUR (Current):** Shows how many frames per second the scene renders at in On Air mode. The number should be above 50 (PAL) or 60 (NTSC), according to the rate that has been specified in the Output Format section.
- **MAX (Maximum):** Shows how many frames per second the scene can render at without waiting for vertical retrace. The higher the maximum value, the more performance is left. If the maximum value is reduced to below 50 or 60, the scene is not rendering in real-time.
- **Blue bar:** Shows the number of frames used.
- **Green bar:** Shows the number of frames available. The Performance Bar refresh rate can be modified in **User Interface** (see the **Configuring Viz** section of the Viz Engine Administrator Guide).

To View All Performance Bar Indicators

- Middle-click .

Copyright © 2026 Vizrt



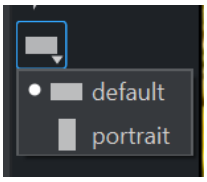
See Also

- Scene Editor Buttons
- Snapshot
- Grid Tool-bar
- Viz Engine Performance

Copyright © 2026 Vizrt

11.4.5 Scene Editor Buttons

Click the Scene Editor buttons once to enable the features, click them again to disable.

- **Camera Selection:** See Camera Selection.
- **Scene Info:** if there is some additional information added to the currently loaded scene, this information is available here.
- 

  Format Selector: To switch between different format in Adaptive Scene Design.
- **Snap:** Takes a snapshot of the current content in the Scene Editor (see Snapshot).

Preview/View Options

- **Post-Processing:** Enables rendering of advanced post-processing effects, such as Depth of Field or Lens Distortion, in the scene editor window. Under normal operation, these effects are only rendered in On Air Mode.
- **Key Preview:** Shows a preview of the keyed output.
- **Normal Vertices:** Shows the normal vertices. For example, it is practical to use this feature when the lighting is not reflected in the required manner. Enabling this feature shows the vertices as purple handles. These handles makes it easier to predict how the light is reflected.
- **Key:** Shows what the key output looks like.
- **Events:** Enables the handling of interactive script/plug-in events. Interactive scripts and plug-ins are those related to mouse or keyboard actions.

Guidelines and Helpers

- **Show Cameras:** Turns on/off camera placeholders.
- **Show Lights:** Turns on/off light placeholders.
- **Wire Frame Mode:** Renders the scene in wire frame mode.
  Default Lighting: Sets the default lighting (Viz Engine Render Pipeline only).
- **Safe and Title Area:** Shows the safe and/or title area as the outline in the Scene Editor.

> **Tip:** The size of the areas can be defined under the Rendering Panel scene settings.

- **Bounding Box:** Enables the bounding box visualization for the selected Container.
- **Performance Bar:** Shows the Performance Bar.
- **Grid:** Opens the Grid Tool-bar.

See Also

- Layer Manager
- Snapshot

Copyright © 2026 Vizrt

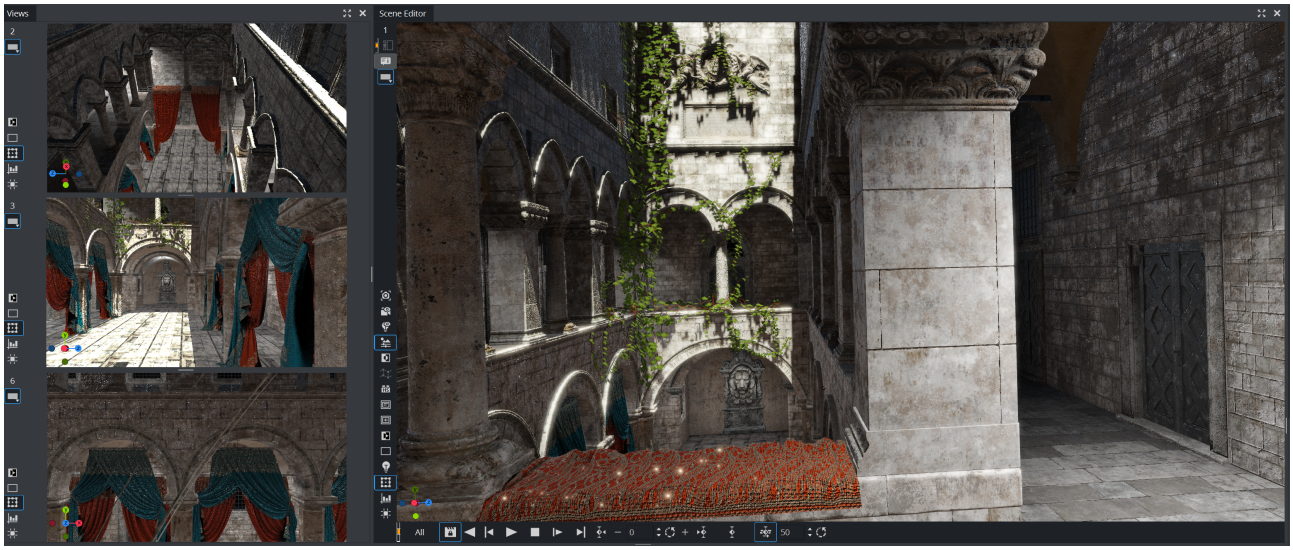
- Performance Bar
- Grid Tool-bar

11.5 Scene Editor Views

To get a better overview of an object, or objects, in a Scene, and to visualize their position in relation to lights and cameras, open the Scene Editor Views function. To open the views panes, click on the **Views** button in the main menu.

As an alternative, you can open a single new Render Window.

> **Note:** The triple view window might be replaced with Render Windows only in a future version.



In principle, each of the three views has the same functionality, and lets you do the same operations on objects as is possible in the Scene Editor.

As well as the Scene Editor, the three side views can be set to view through one of the 16 editable cameras or through one of the six predefined orthogonal views (see Camera Selection). When set to one of the predefined orthogonal views this allows the position and direction of lights and cameras, to be modified.

If the Scene Editor is set to show one of the predefined orthogonal side views, there are some special shortcuts that may be used to pan and zoom:

- **Pan:** Left-click, hold and drag in the background (not the objects).
- **Zoom:** Hold down **Z** (zoom in) or **X** (zoom out) and draw a rectangle with the left button pressed, or position the cursor over a panel and press the **Z** or **X** keys multiple times to zoom in or out in a stationary position.

In the six predefined orthogonal views, the currently selected camera, and in addition all cameras that have been given the status **Vis** or **inf**, show as symbols.

Click on a camera in the side views and that cameras focus is set in the Scene Editor, and handles to edit the position and direction show (drag the handles to change position and direction).

The same functionality is available for lights (see Lights). The light currently selected and those lights which have their **Vis** status set to `On`, are visible, as symbols. Their position and direction can be edited with the cursor.

See Also

Copyright © 2026 Vizrt

- Light Editor
- Cameras

Copyright © 2026 Vizrt

12 Editors

Find information about the following editors here:

- Transformation Editor
- Texture Editor
- Classic Text Editor
- Viz Engine Text Editor
- Workspace Editor

See Also

- Camera Editor
- Light Editor
- Scene Editor

Copyright © 2026 Vizrt

12.1 Transformation Editor

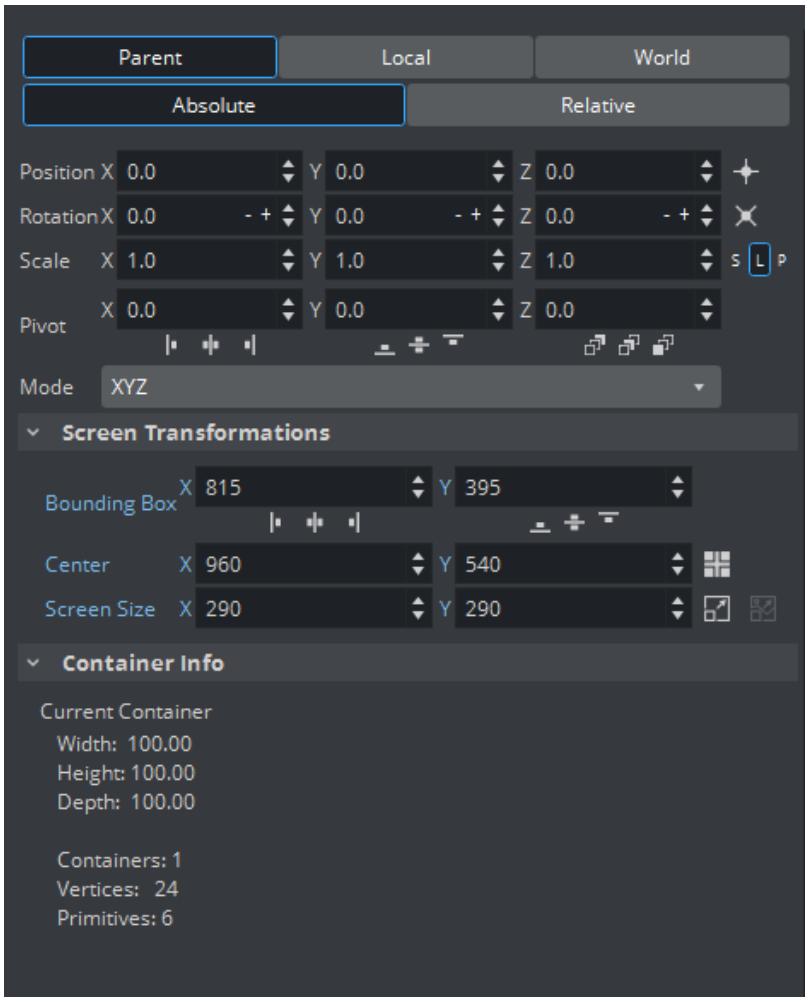
Every Container in a Scene contains transformation properties. The transformation editor holds information about the most basic properties of a Container, such as position and rotation. The transformation editor is also able to stick transformation properties to the container view. This feature allows Position, Rotation, Scaling and Axis Center parameters to stick to the Container view while opening other editors. This enables designers to adjust transformation properties while working on other objects (for example, text or geometry).

This section contains information on the following topics and procedures:

- Transformation Editor Properties
  - Transformation Settings
  - Container Information
  - To Open the Transformation Editor
- Transformation Principles
- Multiple Select Transformation
- Copy Transformation Properties to Another Container
  - To Copy Transformation Properties to Another Container

Copyright © 2026 Vizrt

12.1.1 Transformation Editor Properties

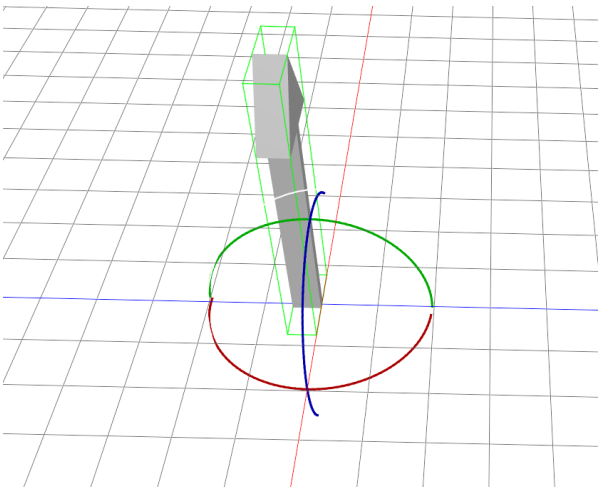
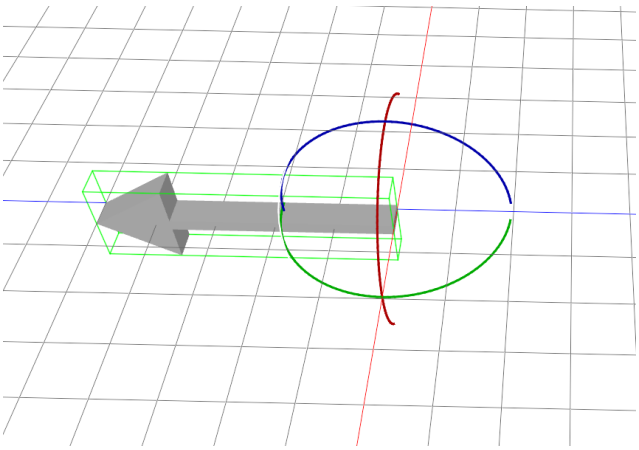


- **Parent / Local / World:** See Transformation Principles.
- **Absolute / Relative:** See Transformation Principles.
- **Position:** Sets the position of the object along the X, Y, and Z axis.
  - **Center Camera:** Clicking this button centers the Container to the camera.
- **Rotation:** Rotates the object according to the X, Y, and Z axes. With the bounding box option enabled, the rotation axes are visible as three lines; X is red, Y is green, and Z is blue. To rotate in steps of 45 degrees, use the buttons labeled **45** and **+45.**
  - **Face Camera:** Changes the rotation in such a way that the object faces the camera.

Rotation Order: Sets the order of rotation the three angles **X**, **Y**, and **Z**, are applied to the Container. If multiple Containers are selected, the order buttons are disabled.
When rotating around each of the three axes, the order in which the rotations are executed matters. Performing a rotation around the axes in the order x, then y, then z will produce different results than performing the same rotations in any of the other five possible orders.

Copyright © 2026 Vizrt

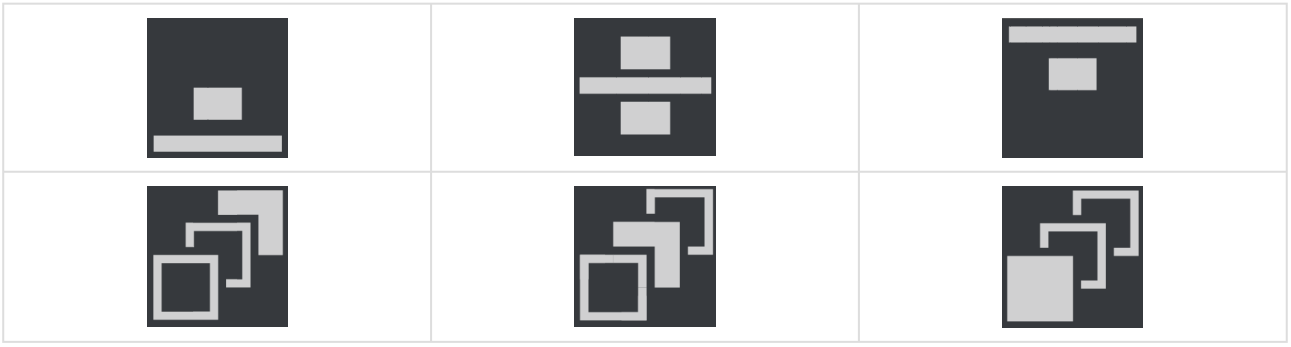
Given a cone with rotation X=90°, Y=90°, Z=90°

Rotation Order XYZ	Rotation Order ZXY
	
The object is rotated in 90° in X, then 90° in Y and 90° in Z direction	The object is rotated in 90° in Z, then 90° in X and 90° in Y direction
Note the colors of the handles in the sceneeditor: Red = X, Green =Y, Blue = Z	

- **Scale:** Before changing the scale values of the Container, make sure to set the preferred kind of scaling:
  - **(S) Single:** Scales each axis by itself.
  - **(L) Locked:** Scales all axes at once, so that X, Y, and Z have the same value.
  - **(P) Proportional:** Scales all axes at once, so that the values remain proportional.
- **Pivot:** Sets the origin of the object. The rotation axis of the object is visible in the Scene Editor if the Bounding Box option is enabled.
  - The origin can be set to left, center, or right on the object's X axis by clicking on the related icons.
  - The origin can be set to bottom, center, or top on the object's Y axis by clicking on the related icons.
  - The origin can be set to back, center, or front on the object's Z axis by clicking on the related icons.

Copyright © 2026 Vizrt



- When a group is selected but certain Containers in it are not visible, clicking Pivot with the middle mouse button defines the axis center only for those Containers in the group which are visible.

Transformation Settings

Sets the screen position along the X and Y axis for images and some objects by specifying pixels instead of the regular positioning which do not use pixels as input values. The Screen Position option is only available if the image or object faces the camera. To achieve this, click the Face Camera button. When changing the Screen Position values, the values for the regular positioning also changes. Set the screen position to left, center, or right by clicking on the related icons.

In Viz Artist, position calculations are always from the upper left corner and based on the current Z value for the Container and camera views. For design purposes, it may sometimes be desirable to position an objects *center*. Therefore, screen positioning of objects can be defined by either:

- **Bounding Box:** Defines the position of the object upper left corner, or
- **Pivot:** Defines the position of the object center
  Both are calculated from the upper left corner of the current view.

Example: Setting an object's screen position Bounding Box values to X=0 and Y=0 places the object in the upper left corner. Note however, that if the Z position changes the object no longer aligns to the viewpoint corner. The pixel values must be set to zero again to reposition the image or object.

- **Align to Pixel:** When using the normal positioning editor to position an image/object that is facing the camera head on, it can occur that the image/object is off by fractions of a pixel. This can happen because of rounding errors in the screen size and face camera functions. This can produce visual artifacts in some cases on some graphics cards. When the image/object is not aligned to the screen pixels, the **Align to Pixel** button turns red. Clicking the button adjusts the image/object to be moved to the nearest matching whole pixel.
- **Screen Size:** Makes it possible to scale an object/image based on pixels. The Screen Size option is only available if the object/image faces the camera. To achieve this, click the Face Camera button.
  - **Scale to Screen:** Scales the object/image to the size of the output format.
  - **Scale to Image:** Scales the image to the original pixel value.

Container Information

This information includes parent Containers plus all visible child Containers.

- **Width, Height, and Depth:** Shows the real size of the object in units (not pixels).

Copyright © 2026 Vizrt

- **Containers:** Shows the number of Containers that are selected. If the selected Container holds Sub-Containers, all Sub-Containers are counted in addition to the main Container. A Container that holds geometry is counted as two.
- **Vertices:** Shows the number of vertices that are rendered within the selected Container, including the vertices in all Sub-Containers.
- **Primitives:** Shows the number of primitives in the Container including the primitives in all Sub-Containers.

To Open the Transformation Editor

1. Click the Transformation icon on the Container, or
2. Select the Container from the Scene Tree, and then click the icon located on the left side of the Property panel.

12.1.2 Transformation Principles

Containers can be manipulated in several contexts:

- **Parent/Group:** Refers to changes that can be made to the Container or Containers in its Parent or Group coordinate system.

> **Note:** Multi selected objects: When more than one object is selected Group shows and a bounding box covers all selected objects.

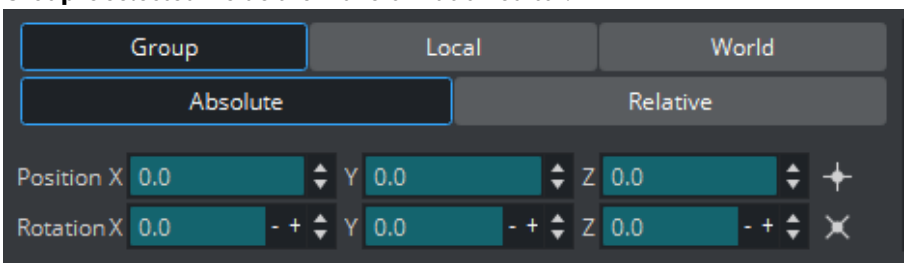
- **Local:** Refers to changes that can be made to the Container in its local coordinate system.
- **World:** Refers to changes that can be made to Containers in their world coordinate system.
- **Absolute:** Refers to changes that can be made to Containers in absolute values.
- **Relative:** Refers to changes that can be made to Containers relative to current values.

12.1.3 Multiple Select Transformation

With multiple select transformation, a group of Containers can be moved to a new position, which is determined by parameters of one selected Container in the selection. All Containers keep their relative position to the selected Container.

All changes made in the Transformation Editor are also applied on each selected Container:

- **Group** is selected inside the Transformation editor.



Copyright © 2026 Vizrt

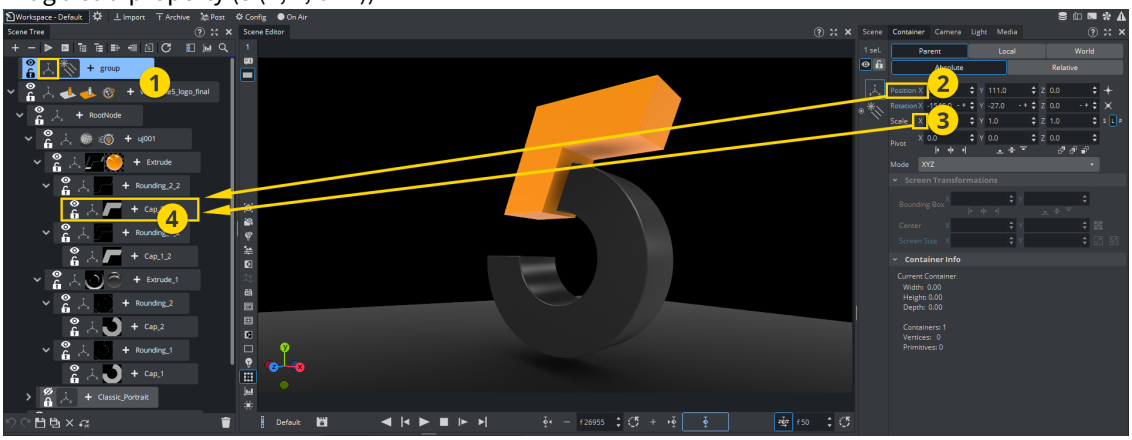
12.1.4 Copy Transformation Properties to Another Container

If a Container has certain settings, for example position or rotation, and another Container in the Scene should have identical settings, it is possible to drag transformation properties from one Container to the other.

All transformation properties can be copied to another Container, either the whole property (for example, **Screen Size**), or just one axis of a property (for example, **X** of **Screen Position**).

To Copy Transformation Properties to Another Container

1. Open the Transformation Editor by clicking on the Transformation icon of the **source** Container (1).
2. Drag the desired property to the **target Container** (4):
    - Drag the whole property (2), or
    - Drag a sub-property (3 (X, Y, or Z))



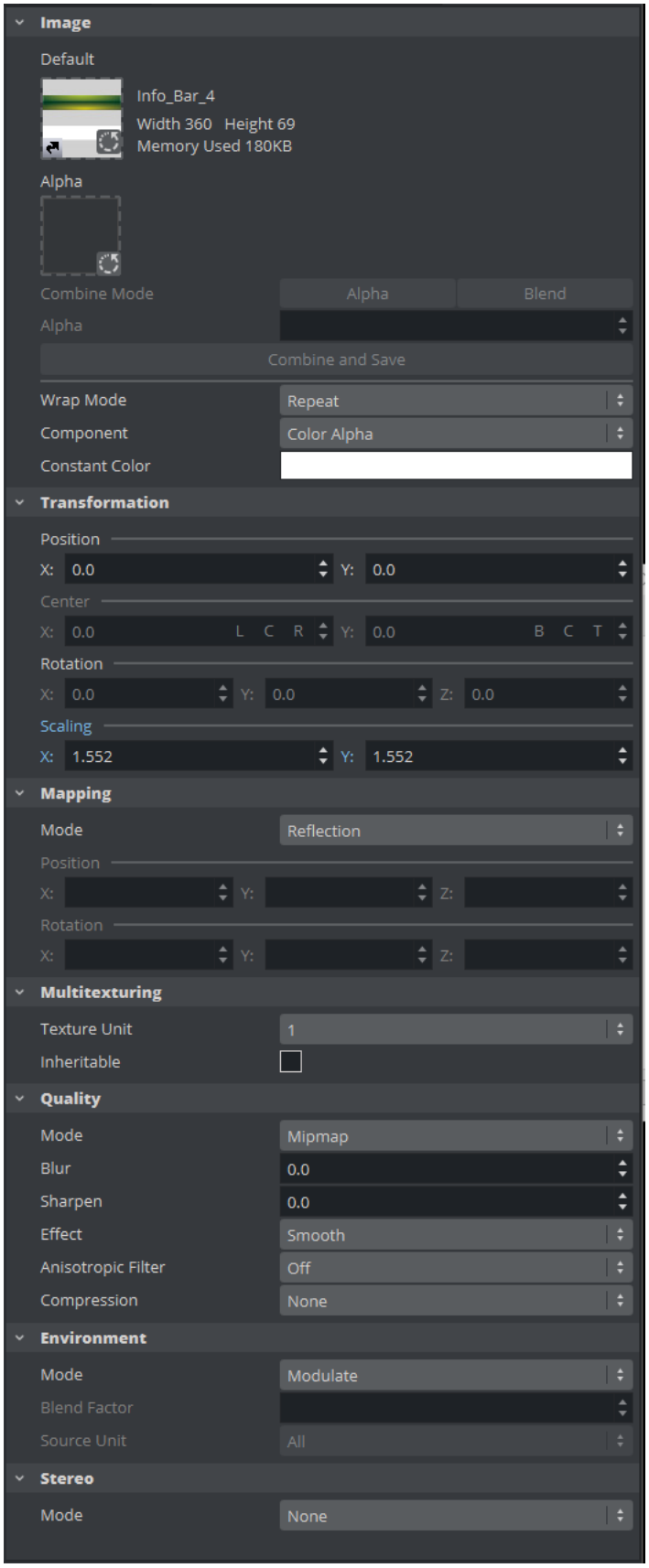
Tip: Transformation properties can also be dropped on multiple Containers in a single mouse drag. Drag a selected property to the Scene Tree as usual, but keep the left mouse button pressed. Tap the **right** mouse button on the desired target containers to apply the same property to multiple containers.

Copyright © 2026 Vizrt

12.2 Texture Editor

To apply a texture to a Container drag a texture (for example, an image) from the Server Panel to a Container. In the Container, an icon (a thumbnail of the texture) shows. If a texture is dragged to the Scene Tree area it creates its own Container, Geometries can be added later on, or the texture can be left as it is in the Scene. Open the Texture Editor by clicking the icon representing the texture, either on the Container or in the Properties editor.

Copyright © 2026 Vizrt



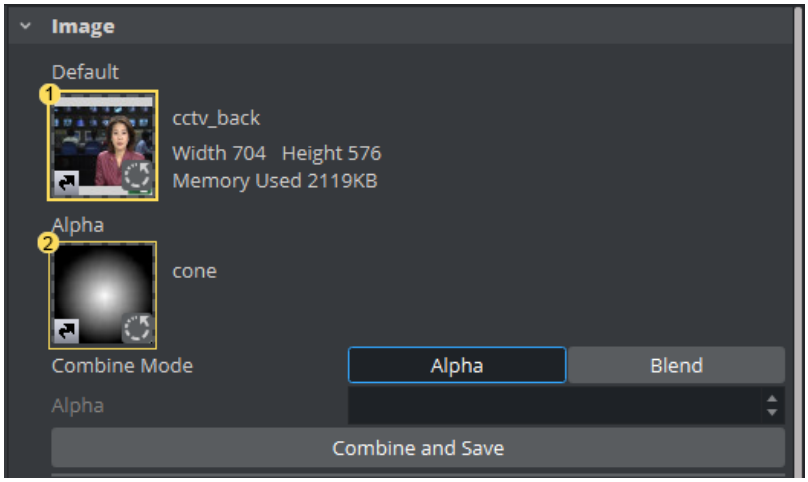
Copyright © 2026 Vizrt

Tip: Modifications to the texture in the Texture editor only apply to the texture in the selected Container.



12.2.1 Image Properties

Shows the added texture:

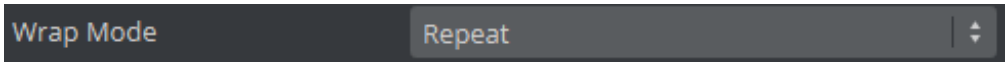


Copyright © 2026 Vizrt

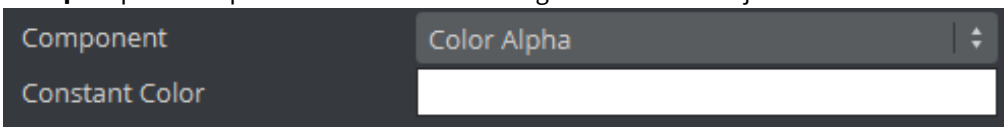
Two image textures can be used in one Container, however, Media Asset textures are limited to one. Click on a Media Asset image to open the editor for the Media Asset.

- **Default Texture** (1): Shows a thumbnail of the primary texture (the same thumbnail that is shown on the Container). To change the texture, drag another texture to the drop zone. If only one texture is used the other functions in the Image panel are inactive.
- **Alpha Texture** (2): Shows the thumbnail of a secondary (alpha) texture, if any. This texture is used as an alpha channel for the Default texture. When two textures are used the Image panel Alpha and Combine functions are active.
- **Combine Mode:** The second image can be used either as an Alpha channel or it can be blended into Texture (1).
- **Combine and Save:** When a second image is added, click this button to create a new single image from the two images. The new image is created in the currently open folder in the Server Panel.

Tip: To save the new image in another folder, move to a different folder in the Server Panel before the Combine button is clicked.



- **Wrap Mode:** Wraps the object by repeating or clamping (no repeat – the texture's edge color is used to fill the rest of the object) the texture onto the object's surface. This setting comes into operation if the texture mapping cannot fill the whole object (for example, the texture was scaled below 0.0):
  - **Repeat:** Repeats the whole texture over again to fill out the object.
  - **Clamp:** Repeats the pixels on the end of the image to fill out the object.



- **Component:** Select from:
  - **Color:** Uses only the RGB color of the texture. Set the alpha to a constant value in item one.
  - **Alpha:** Uses only the alpha of the texture. Set the RGB color of the texture to a constant color specified in item one.
  - **Color Alpha:** Uses both the alpha and the RGB color of the texture (default setting).
- **Constant Color:** Set the constant RGB color of the texture when **Alpha** is selected.

Copyright © 2026 Vizrt

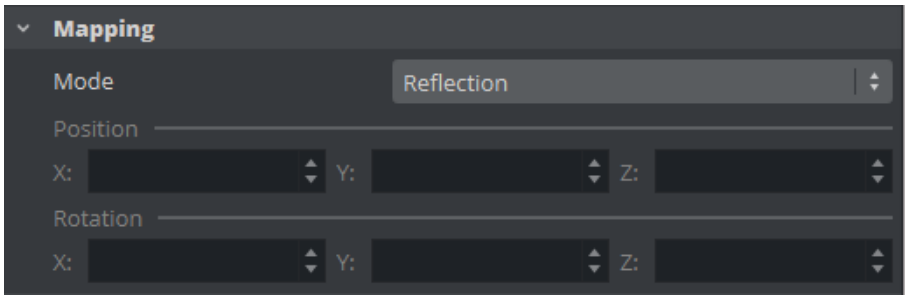
12.2.2 Transformation



- **Position:** Defines the position of the texture relative to the one stored in the picture.
- **Center:** Shifts the center of the image. The center can be set to **L**eft, **C**enter **R**ight respective **B**ottom, **C**enter, **T**op.
- **Rotation:** Defines the rotation of the texture relative to the one stored in the picture.
- **Scaling:** Scales the image that is used as texture.

12.2.3 Mapping

The mapping option adjusts the way texture mapping is done:



- **Vertex:** Means that the texture mapping is done using the object texture, or UV, coordinates. These coordinates index a texture image, and are interpolated across the object to determine, at each of the object pixels, a texture image value. In most cases on 3D objects, this gives the best looking result. The built-in geometry plug-ins in Viz Artist have defined their texture coordinates. Imported objects can have texture coordinates if they have been generated by an external 3D modeling program. If for example adding a picture to a cube, all sides look the same, and the whole image is shown on each side.
- **Linear:** Is a 'straight forward' mapping where the image so to say is laid over the object directly on the frontal plane. The object is then moved through the image and every pixel on the object inherits the pixel it

Copyright © 2026 Vizrt

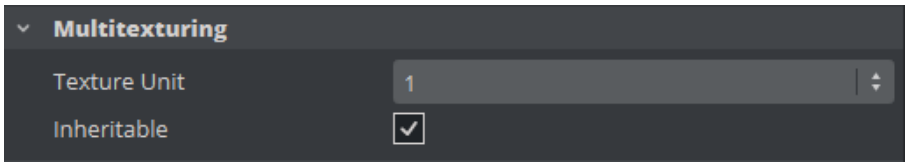
"touches" on the image. On 3D objects, this causes a repetition of the pixels on the surfaces in the Z axis. If for example adding a picture to a cube, the front and back side look good, but the remaining sides are a repetition of the pixels laying on the edge of the front and the back side.

- **Reflection:** Maps the texture as if it were a reflection of a surrounding object. The reflection simulates that the texture image is an all-embracing object that casts its reflection on the object. The object acts in a way like a mirror.
- **Spherical:** Spherically texture projection onto a geometry. The mapping position and rotation can be adjusted with the corresponding parameters.
- **Cylindrical:** Cylindrical texture projection onto a geometry.
- **Cube:** Cube texture projection onto a geometry. This option needs a vertical cross cubemap which specifies how each of the six side projections should look like. A second image cannot be used with **Cube** Mapping (see Image Properties). When using Cube mapping method for textures, it is not possible to use the same texture on another object with a different mapping method.
- **Mapping Position:** Sets the texture projection origin for Spherical and Cylindrical mapping.
- **Mapping Rotation:** Sets the texture projection rotation for Spherical and Cylindrical mapping.

Note: Mapping position and rotation are only available for Spherical and Cylindrical mapping types.

12.2.4 Multitexturing

Texture units take a texture and map them to a piece of geometry:

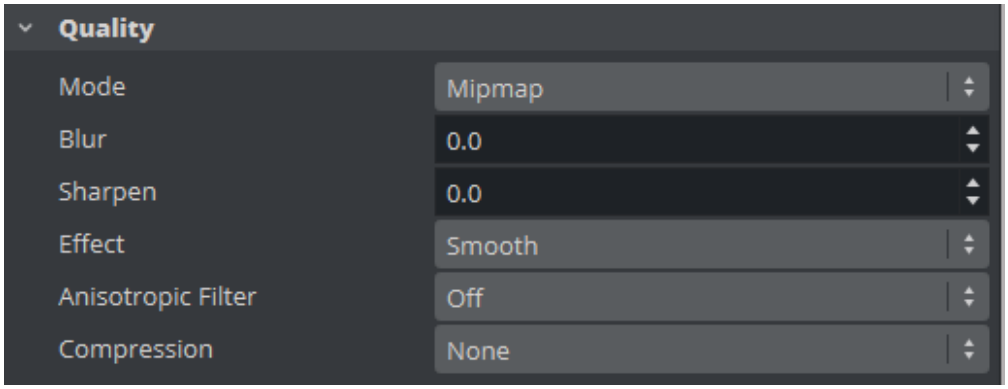


- **1-4:** If the graphics card supports it, it is possible to have up to four available texture units which can be used in parallel to do multi-texturing.
- **Inheritable:** Units can be inherited onto the textures of child Containers. Multiple inherit is possible, but be sure to use different texture units for each texture that should be inherited to not overwrite a unit.

12.2.5 Quality Properties

The quality parameters specify a trade between rendering speed and texture quality:

Copyright © 2026 Vizrt



- **Mode:**
  - **Pixel:** Shows the texture 'as it is' with no smoothing and anti aliasing. It takes little resources to render, but it can often create a poor visualization. This can be visible as 'toothed' lines and transitions, especially when close to the texture. A motion of either object or camera, especially in the Z axis, creates a 'blurring motion' on the texture, as the texture scales to fit the object.
  - **Linear:** Performs a linear interpolation to smooth the texture when being magnified or shrunk. The texture looks good, but some distortions can be visible when the textured object is animated further away on the Z axis. As the object then gets smaller and smaller, the shrinking and interpolation of the texture creates a lot of "noise" on the texture. As a consequence, the linear quality is appropriate when the objects that have the texture do not change their size much.
  - **Mipmap:** Performs a linear interpolation to smooth the texture. In addition, it offers a solution to the problem that shows on the two other qualities when the object is being moved away along the Z axis or shrunk. To avoid the "noise" that we see when a texture constantly scales to try to fit onto a shrinking/magnifying, the mipmap quality creates a set of prefiltered images in a decreasing scale down to almost on pixel. These are used as the object scales up and down and a more static look of the texture is thereby achieved. Mipmap quality absorbs more memory performance than the two other qualities.
- The following quality options are only available for the Mipmap quality.
  - **Blur:** Makes the image blurry. The blur filter can be set to either `Smooth`, which blurs the image smoothly, or `Mosaic`, which makes the blur look like tiles.
  - **Sharpen:** This option is useful when the rendering window is scaled to half height mode.
  - **Anisotropic Filter:** The Anisotropic Filter can be set to either `Off` or `2-16x`. Anisotropic filtering is used, when textured objects need to be rendered in z-direction, where textures usually get blurred.

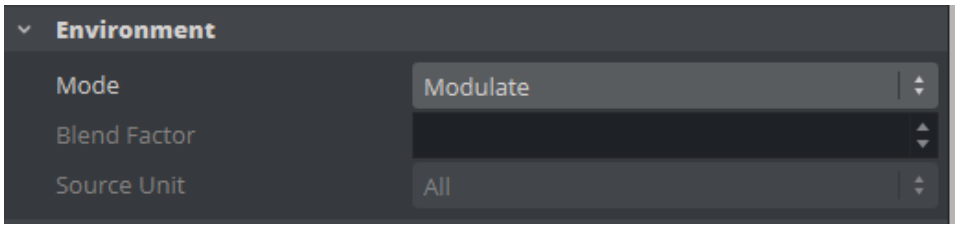
Texture compression can be used to save texture memory on the graphics card:

- **None:** No compression.
- **DXT1:** Compresses to an RGB image. This format has a 6:1 compression ratio (DXT1 does not store alpha).
- **DXT3:** Compresses to an RGBA image with a 4:1 compression ratio.
- **DXT5:** Compresses to an alternative RGBA image that treats the alpha channel different than DXT3 (alpha looks smoother). DXT5 has the same compression ratio as DXT.

Copyright © 2026 Vizrt

12.2.6 Environment Properties

The environment parameters specify the way the texture RGBA information is combined with the object RGBA information:

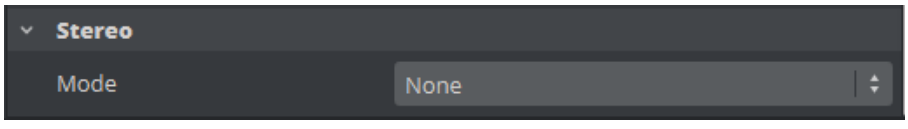


- **Environment:** Select from:
  - **Blend:** Blends the RGBA parameters of texture and object.
  - **Decal:** Uses only the RGBA information of the texture and disregards the RGBA of the object.
  - **Modulate:** Uses the RGBA information of the texture and enables in addition use of lighting on the textured object.
  - **Add:** Adds source unit one and source unit two together.
  - **Add Signed:** Adds source unit one and source unit two then subtracts 0.5.
  - **Interpol.:** Linear interpolation between source unit one and source unit two.
- **Blend Factor:** Defines in interpolation mode how much of source unit one and unit two is used (0.0%: shows current unit, 100%: shows source unit).
- **Source Unit:** Sets the second source unit for interpolation mode (can be all other units together or just a specific unit).

Note: Blend Factor and Source Unit environment options are only available for the Interpolated environment.

12.2.7 Stereo Mode Properties

Select the correct stereo mode for the inserted image:



- **None:** No Stereo mode.
- **Config:** Images are drawn as set in the Viz Configuration (see the **Configuring Viz** section of the Viz Engine Administrator Guide).
- **Over Under Left Top** or **Over Under Right Top:** Two images are stored, one over (top) and the other under (bottom), left or right eye (top) first. The image height is halved, providing half resolution.

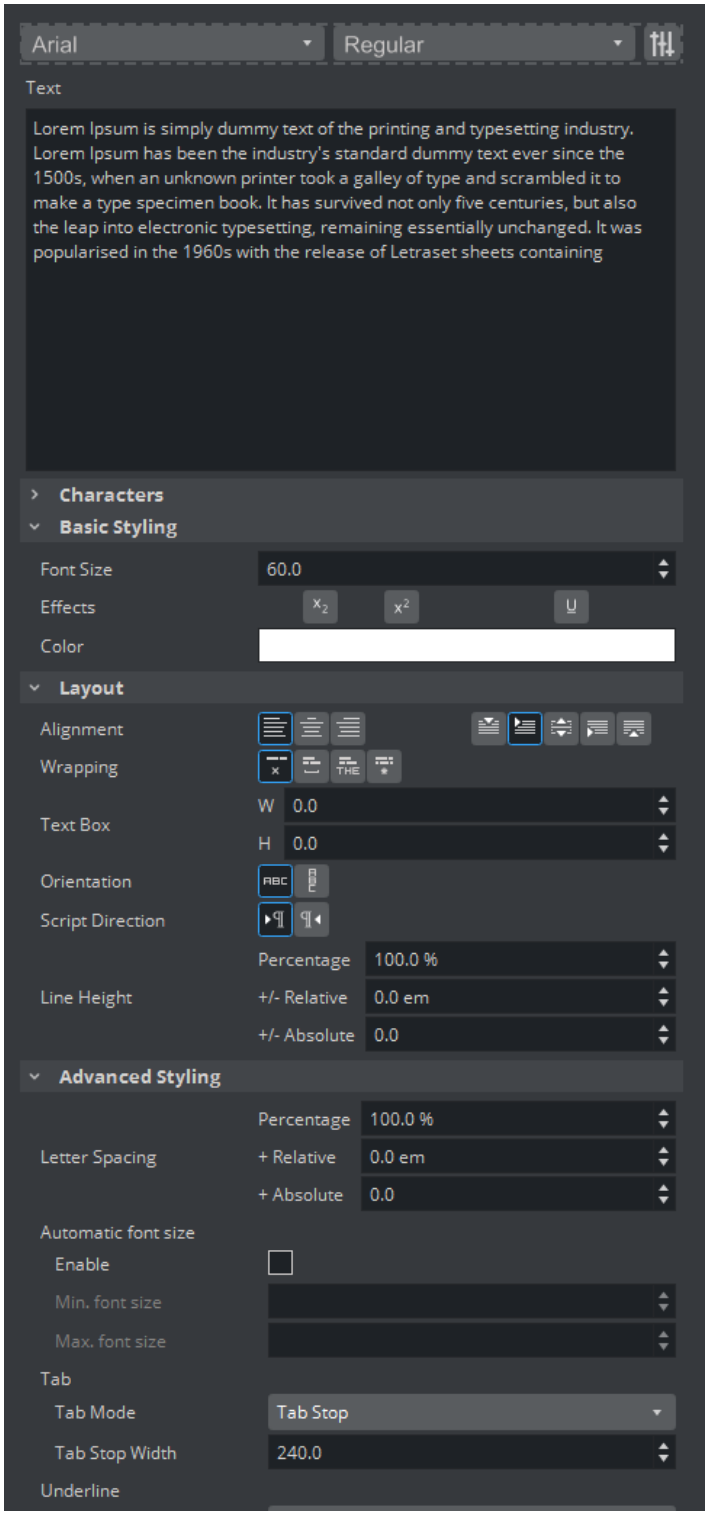
Copyright © 2026 Vizrt

- **Side by Side Left** or **Side by Side Right:** Two images are stored, side by side, left or right eye first. The image width is halved, providing half resolution.

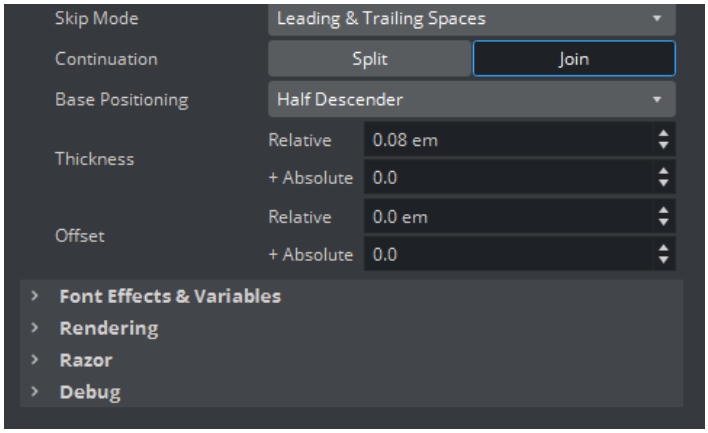
Copyright © 2026 Vizrt

12.3 Classic Text Editor

Use the Text Editor to make changes to a text in a scene.



Copyright © 2026 Vizrt



This section contains information on the following procedures and topics:

- To Open the Classic Text Editor
- To Create a Classic Text Object
- Classic Text Editor
  - Font
  - Characters Tab
  - Basic Tab
    - Effects
    - Spacing
    - Underline Options
    - Coloring Options
    - Rendering Options
  - Orientation Tab
    - Alignment
    - Text Box
  - Shadow Tab
  - Clock Tab
    - Clock Format Examples
  - Special Tab
- Inline Text Editor
- Special Fonts

Note: In the Viz Engine Render Pipeline, text rendering is implemented completely differently than in the Classic Render Pipeline. Some features are not yet implemented. Only Geometry and Razor type text are supported by the Viz Engine Render Pipeline.

12.3.1 To Open the Classic Text Editor

There are two ways to open the text editor.

- Click the icon on the container of the text you want to edit.
- Double click on the text in the scene editor. This also activates the inline text editor inside the scene editor.

Copyright © 2026 Vizrt

12.3.2 To Create a Classic Text Object

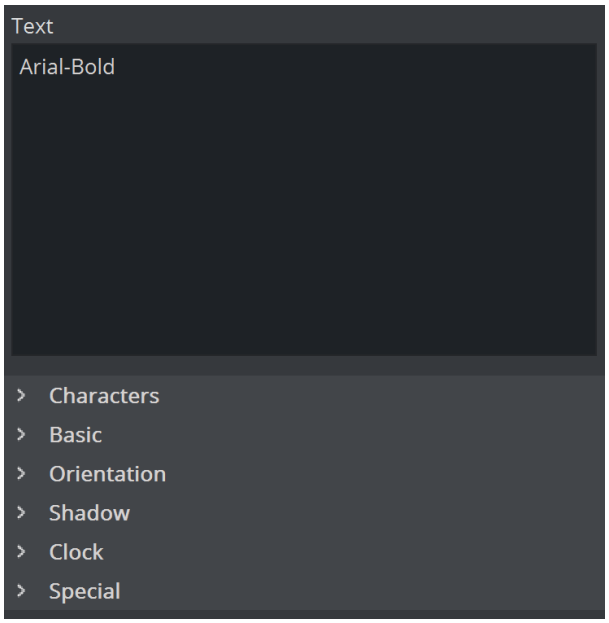
There are two ways to create a new classic text object.

- Drag a font from the Server tab into the scene tree. A new classic text object is created in or under the container where the font is dropped.
- Drag a font into the scene editor. A new classic text object is created at the dropped location.

By default, the content of the new classic text is set to the name of the font. You can modify the text by opening the classic text editor.

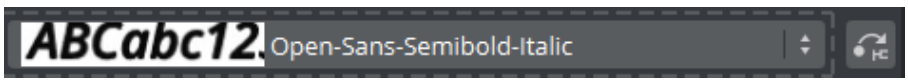
> **Note:** Classic Fonts in the Viz Engine Render Pipeline are blocked. Please be sure to use the Classic Render Pipeline.

12.3.3 Classic Text Editor



At the top of the Classic Text Editor is an input box. Characters in this input box are rendered in the Scene Editor. The Classic Text Editor contains six tabs allowing detailed modification. More advanced modifications can be done in the Fontstyle Editor. Modifications made with the Fontstyle Editor apply to *all* instances of the font, not just the one used in this Scene.

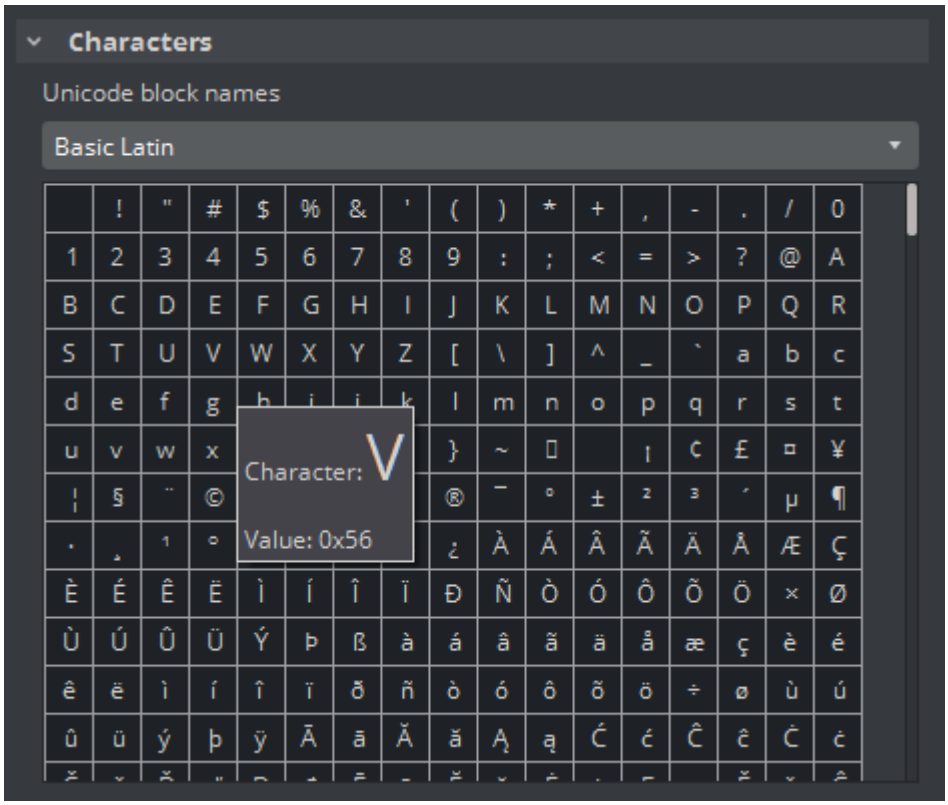
Font



Copyright © 2026 Vizrt

Lists all available fonts created from the same base font. To use another font, select it from the list. To use a totally different font in the scene, drag the new font from the **Asset View** panel onto the Font Areas of the text.
To locate the font in Asset view, click on the button.

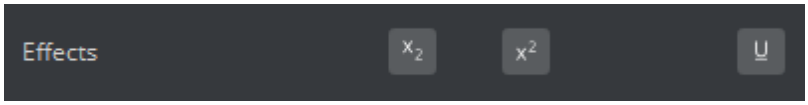
Characters Tab



The Characters tab shows the character table in the fonts library. As an alternative to the keyboard, the symbols from the table can be dragged onto the input box at the top of the Classic Text Editor. This is a helpful tool when using special characters that do not exist on the keyboard, or are hard to find/use.

Basic Tab

Effects

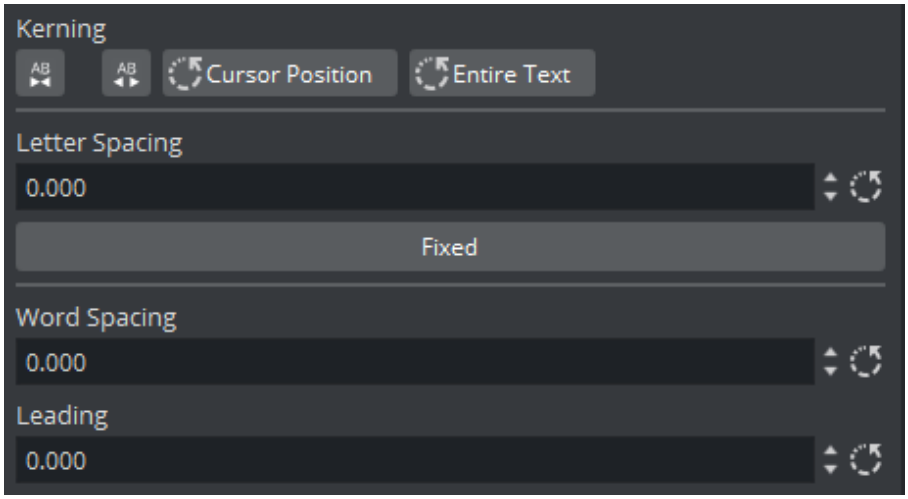


- **Bold:** Toggles bold effect.
- **Italic:** Toggles italic effect.
- **Underline:** Toggles underline effect. Additional options can be found in the following section.

Copyright © 2026 Vizrt

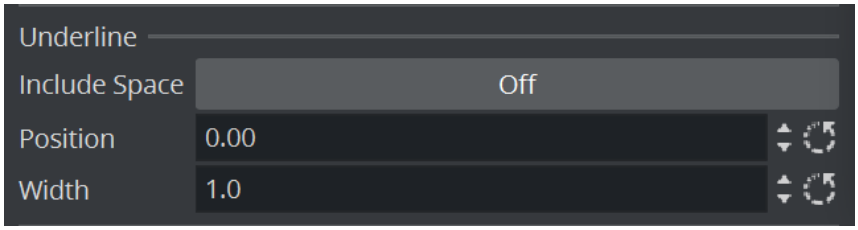
- **Subscript:** Toggles subscript effect.
- **Superscript:** Toggles superscript effect.

Spacing



- **Kerning:** Set custom kerning between the characters at the current cursor position. This can also be done by using the **ALT** key and cursor left and right. This property can be reset for the current cursor position or the entire text using the corresponding reset button.
- **Letter Spacing:** Changes the spacing between all characters by the same amount. Clicking the **Fixed** button changes the text from true type to one where each character uses the same space.
- **Word Spacing:** Changes the size of the spaces between the words in the text.
- **Leading:** Modifies the distance between the text lines.

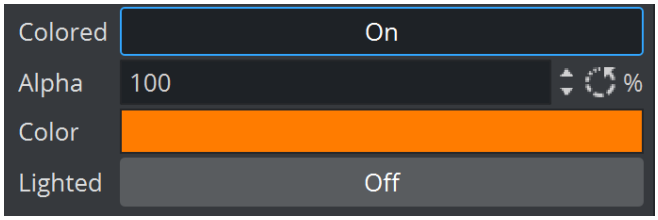
Underline Options



- **Include Space:** Sets whether to underline the whitespace characters.
- **Position:** Sets the underline offset.
- **Width:** Sets the thickness of the underline.

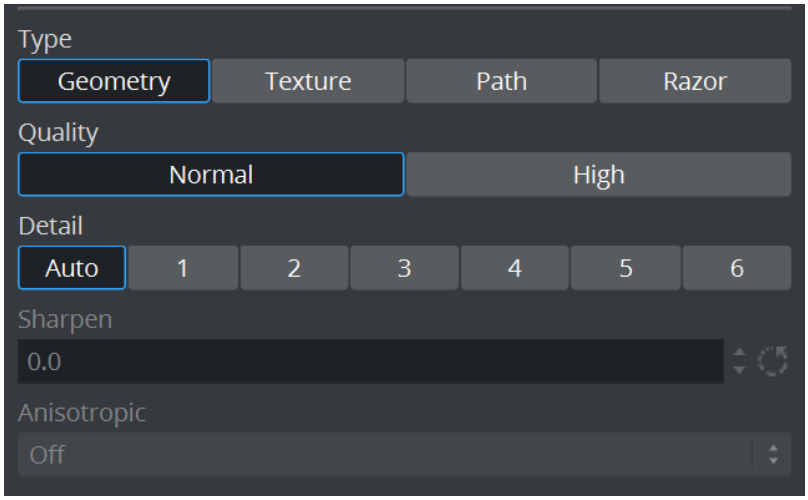
Copyright © 2026 Vizrt

Coloring Options



- **Colored:** Activates the selected color for the text as a solid color, overriding the color of any material on the text container in the scene tree.
- **Lighted:** Applies the selected color to the text, adjusted for the scene lighting. Transforming the object creates the visual impression of changes to the color. When inactive, the selected color is applied to the text as a solid color.

Rendering Options



Type: Sets the rendering method for the text.

- **Geometry:** Renders geometry type fonts as polygons.
- **Texture:** Renders the font as texture instead of polygons. This can, to some degree, create a poorer look, but it reduces the number of polygons in the Scene. The texture resources get an increased demand, so to render a font as texture is a trade-off between polygon and texture performance. For half height mode rendering, texture fonts are normally the better choice. When selected, the anisotropic value is set to `4X` by default.
- **Path:** Renders the font as vector graphics.

Copyright © 2026 Vizrt

Note: To use the **Path** type, the machine must be equipped with NVIDIA GPU and have a driver version 275.33 or higher installed.

- **Razor:** Renders the font as texture using graphics shader. This method of rendering produce higher quality of anti-aliasing. To use the **Razor** type, the font must have been imported with Viz Artist version 3.11.0 or higher.

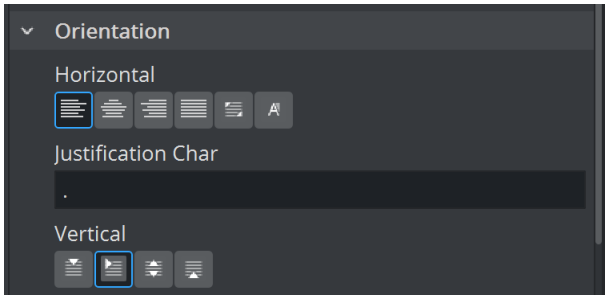
Note: A **Razor** type text may produce distorted result when rendering glyphs with high-complexity. Artifacts may appear when two glyphs overlap each other. Lighting is currently not support when using this type.

Note: Some effects, like Extrusion, are not supported by **Texture**, **Path**, and **Razor** rendering.

- **Quality:** Sets the quality of the font to either `Normal` or `High`. The quality of a texture font can be changed in the Fontstyle Editor. This option only applicable if the font type is set to **Geometry**.
- **Detail:** Sets the level of detail, or tessellation of the font, to either automatic level or one out of six fixed settings. The fonts are rendered as polygons, so by switching to wireframe mode, the difference becomes obvious as the detail level changes. One is the highest level of detail, six is the lowest. If setting a fixed level of detail, the automatic level is overwritten and rendered with the same level of detail independent of the font size. This option is only applicable if the font type is set to **Geometry**.
- **Sharpen:** Makes it possible to sharpen the font. This option is only available if font type is set to **Texture**.
- **Anisotropic:** Improves the texture sampling for anisotropic (in this case non-square) pixel filter areas in mipmapping mode, for example a rotated text texture. Choose either to use the standard mipmap level selection, or select the anisotropic filter rate from a number of available settings. Higher values give better image quality at the expense of some performance. The default value for anisotropic filtering is `4X`. This option is only available if font type is set to **Texture**.

Orientation Tab

Alignment



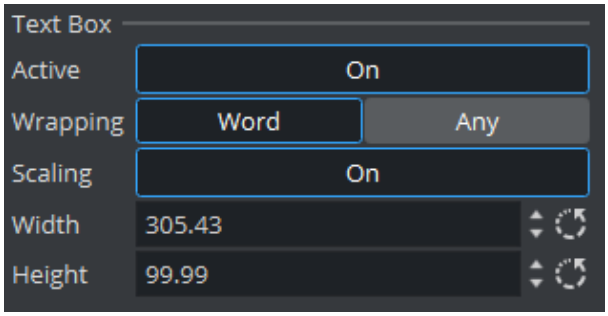
- **Horizontal:** Sets the text alignment or justification in horizontal axis.
  - **Left**, **Right** or **Center**.
  - **Justify** to fill the text box (must be enabled).

    - **Scaled** to fill the text box (must be enabled).
    - **Justify to Character**.
- **Justification Char:** Sets character to use for Justify to Character option.

> **Note:** Justify to Character is not supported in the Viz Engine Render Pipeline.

- **Vertical:** Sets the text alignment in vertical axis.
    - **Top:** Align text to the top of the container position.
    - **First line:** Align the first baseline of the text to the container position.
    - **Middle:** Align text to the middle of the container position.
    - **Center:** Align text to the bottom of the container position.

Text Box



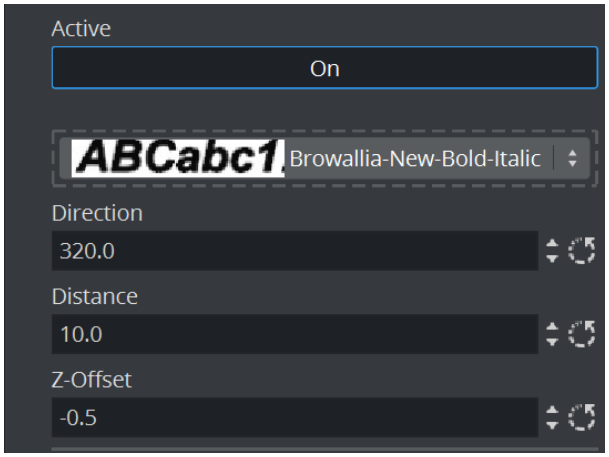
- **Active:** Wraps the text within a specified area of the text box when enabled. The text box is shown as an orange outline in the Scene Editor when the text editor is active (if Bounding Box is enabled in the Scene Editor).
- **Word:** Each line of text contains whole words.
- **Any:** Each line of text contains all possible letters. Words are split at the Text Box right border.
- **Scaling:** When enabled, only the defined textbox is responsible for breaking the text. If turned off, the scaling of the geometry is also taken into account.
- **Width:** Sets the width of the Text Box.
- **Height:** Sets the height of the Text Box.

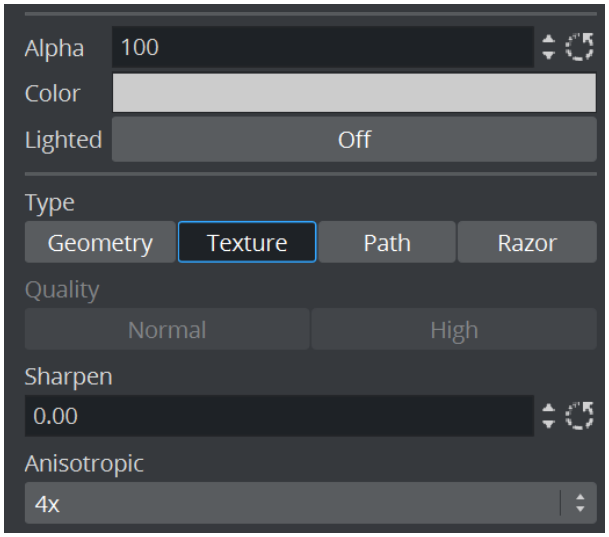
Shadow Tab

From the Shadow panel, shadow can be added to the text. Click the small switch left of the font visualization to enable text shadowing.

> **Note:** Shadow is currently not supported in the Viz Engine Render Pipeline.

Copyright © 2026 Vizrt





Note: All the modifiers that are applied to the text in the Fontstyle Editor are applied to the shadow too.

- **Font:** Lists all available fonts created from the same base font as selected. To use another font, click the arrow and select it from the list. To use a different font in the scene, drag a new font from the Server Panel onto the visualization of the text.
- **Direction:** Sets the direction of the shadow from 0 to 360°.
- **Distance:** Sets the distance between text and shadow.
- **Z-Offset:** Sets the displacement of the shadow on the Z axis.
- **Alpha:** Changes the alpha value of the shadow.
- **Color:** Sets the color of the shadow.
- **Lighted:** Lights the font with the current light settings (see Light Editor).

Note: The Color option is only applicable if the color modifier has not been applied to the font.

- **Type:** Sets the rendering method for the shadow (see Basic Tab).
- **Quality:** Sets the quality of the font to either Normal or High (see Basic Tab).

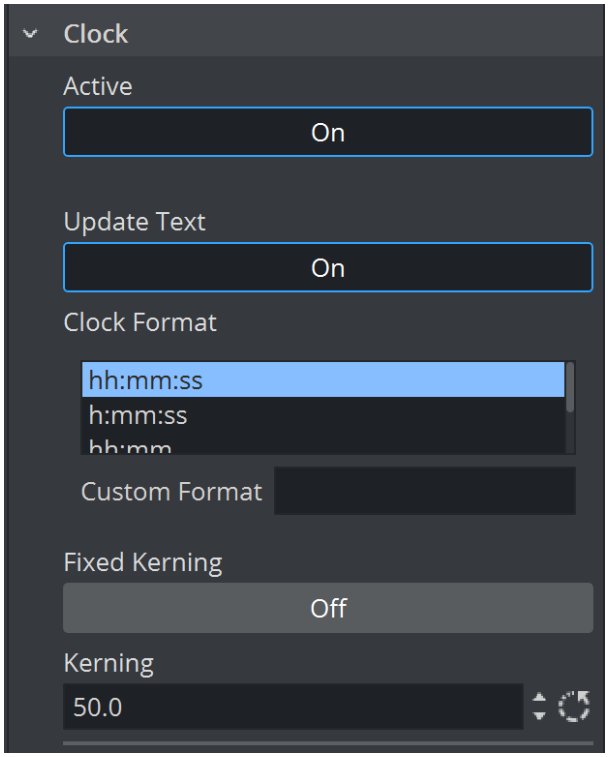
Copyright © 2026 Vizrt

- **Sharpen:** Makes it possible to sharpen the shadow font (see Basic Tab).
- **Anisotropic:** Sets the quality of shadow textures (see Basic Tab).

Clock Tab

Click the **Clock** tab in the Text Editor to open the **Clock** panel. In the **Clock** panel, the display of a clock can be inserted and edited. Click the small switch in the upper left corner of the panel to enable the clock. The clock is added to the text string, if any, at the cursor position.

> **Note:** Clock is currently not supported in the Viz Engine Render Pipeline.



- **Update Text:** When enabled, any running clock in the scene is updated in real time in the **Scene Editor** and **On Air** outputs. If disabled, the output to the Scene Editor and On Air outputs are not updated, even if the clock is running. An external control application, such as for instance *Viz Trio*, could be used to enable or disable Update Text when running in On Air mode, as required.
- **Clock Format:** Sets the format of the clock display.
  - **H:** Hours.
  - **M:** Minutes.
  - **S:** Seconds.
  - **D:** Digits of a second. The first digit shows a tenth of a second, the second digit shows a hundredth of a second, the third digit shows a millisecond, and so on.
- **Custom Format:** Uses a user defined (non-standardized) clock format in the box below. Make sure to add correct separation characters.

Copyright © 2026 Vizrt

Note: Viz Engine distinguishes between uppercase and lowercase clock formats, allowing the designer even more customization options. When defining the clock format by lowercase letters, for instance `hh:mm`, the digit is shown even if the value is `0`. However, if the clock format is defined by uppercase letters, Viz Engine does not display the value if it is zero. This allows the designer to omit unused digits while still planning for their use when designing the scene.

- **Fixed Kerning:** Sets a fixed width for each letter. The kerning of non-proportional fonts vary for the various letters. For example, 1 has less kerning than 6. To avoid the letters from “jumping” around while counting, the Fixed option can be enabled. The Fixed value refers to the size of the text box around each letter.

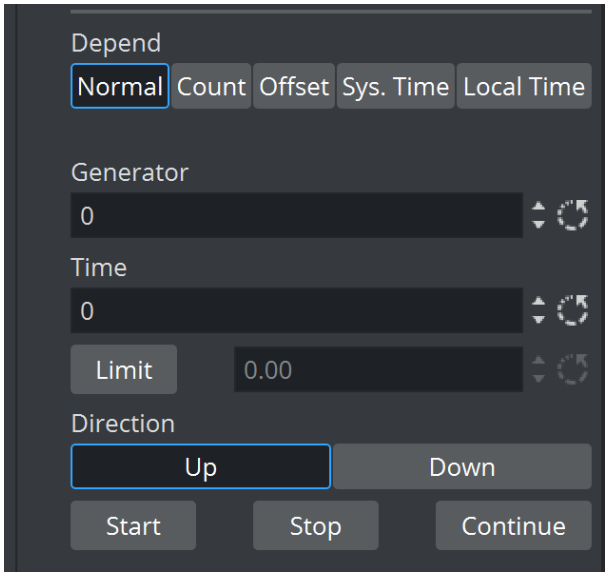
Note: The problem with changeable kerning can be avoided by using a proportional font.

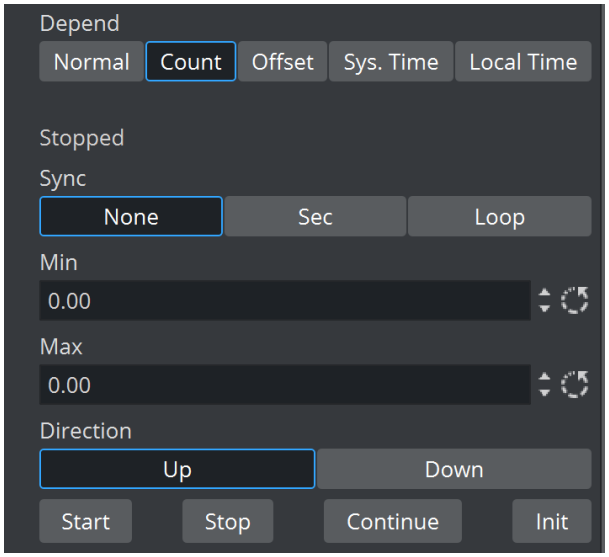
Clock Format Examples

Clock Format	Displayed Result
`HH:mm:ss`	`03:05`
`Hh:mm:ss`	`0:03:05`
`HH:Mm:ss`	`3:05`
`HH:Mm:s`	`3:5`
`hh:mm:ss`	`00:03:05`
`hh:M:ss`	`00:3:05`

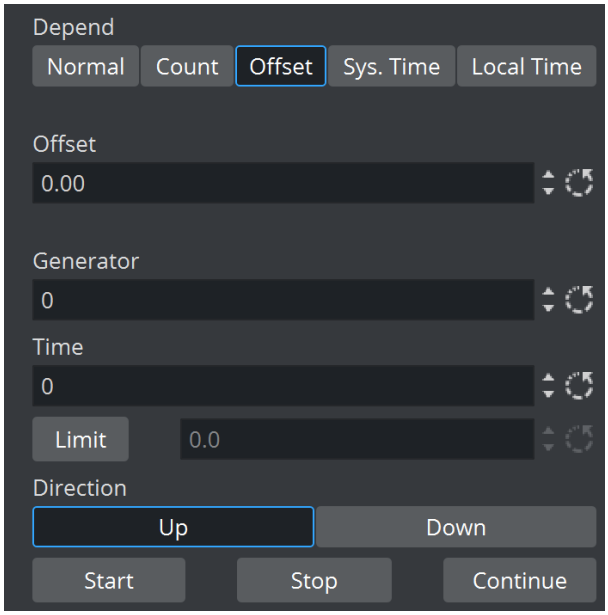
Tip: Up to 20 clock formats can be defined in **Clock Formats** (see the **Configuring Viz** section of the Viz Engine Administrator Guide).

Copyright © 2026 Vizrt





Copyright © 2026 Vizrt



- **Depend:** Selects the type of clock to use.
  - **Normal:** Shows the normal main clock view.
  - **Count:** Creates a counter, which counts a defined time lap in relation to one of the running clocks. For example, if a clock is running, another Container with the same clock can be created. The state of the second clock should be set to Count, and a time range defined. When clicking the **Start** button under the **Count** section, the count clock starts counting for the defined time. It is related to the main clock, so if stopping the main clock, the count clock also stops. A typical use for this function is for counting penalty times in sport graphics. In a sport game running on efficient time, a penalty count down should stop when the main game time stops.
  - **Offset:** Displays an input box where an offset value in seconds can be entered. For example, if the value is set to 20, the clock starts running after 20 seconds.
  - **Sys. Time:** Sets the clock to the system time.
  - **Local Time:** Sets the clock to the local time. This is useful if running on a system which automatically adjusts to daylight savings time.
- **Remember time properties:** Saves the state of the clock (time, limit and direction) in the scenedata.

Each kind of clock comes with different additional options.

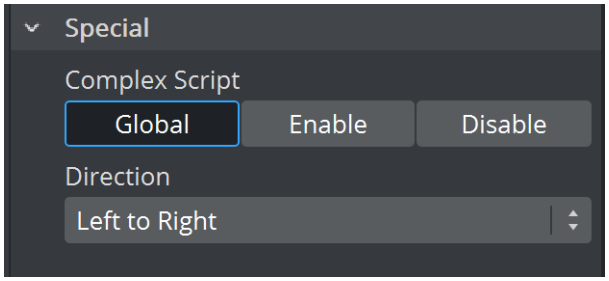
- **Generator:** (Normal/Offset) Sets the ID of the clock. Up to ten different clock IDs (0-9) can be used in a Scene. Clocks with the same ID are linked together, which means they are synchronized and all start when one clock is started. This can be used to show different time zones, for example
- **Time:** (Normal/Offset) Defines the start time in seconds. For example, if the clock should start to count at two minutes, set Time to `120`.
- **Limit:** (Normal/Offset) Defines the count limit in seconds. For example, if the clock should stop counting after two minutes, set Limit to `120`.

Copyright © 2026 Vizrt

Note: If counting down to 0, Limit must be set to `0` seconds, or the clock continues to count below 0. If counting upwards starting at 0, make sure that Limit is not `0`, as the clock does not count at all.

- **Sync:** (Count)
  - **None:** Does not synchronize the count clock to the main clock when enabled. It is almost impossible to start the count clock exactly at a whole second, so the count clock most likely runs unevenly compared to the main clock. In many cases, this does not create the desired output, but may be useful if an offset output is desired.
  - **Sec:** Synchronizes the count clock with the main clock.
  - **Loop:** Loops the count clock within the defined count range.
- **Min and Max:** (Count) Sets the count clock minimum and maximum parameters.
- **Direction:** (Normal/Count/Offset) Specifies if the clock should count up or count down.
- **Up and Down:** (Normal/Count/Offset) Specifies if the clock should count upwards or perform a countdown.
- **Start, Stop and Continue:** (Normal/Count/Offset) Makes the clock start, stop and continue counting.
- **Init:** (Count) Resets the clock's counter.

Special Tab



- **Complex Script:** Enabling Complex Script allows usage of advanced typographical features including, but not limited to, font ligature (important for many Asian scripts and handwriting fonts), right to left and bi-directional writing, special symbols and Unicode emojis. You can **Enable** or **Disable** this feature on a per-text basis using this option, or select **Global** to use the globally set configuration. **Global** is selected by default.

Note: In the Viz Engine Render Pipeline, text is rendered with these advanced features enabled by default.

- **Direction:** Switches the text direction from left to right, right to left, downwards, and upwards.

Note: When complex script is disabled, all characters are rendered directly to glyphs according to the **Direction** option. For example, the text *ABC* is rendered as *CBA* when the direction is right to left. When complex script is enabled, ligature and bi-directional text algorithms are also performed, and the **Direction** option dictates the primary writing direction. Thus, the rendered glyphs are ordered in the direction of its native writing system.

Copyright © 2026 Vizrt

12.3.4 Inline Text Editor

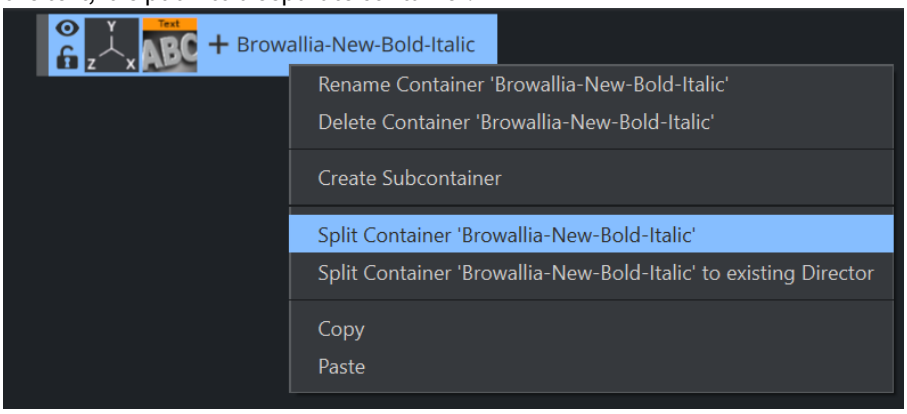


Inline text editor allows you to perform simple editing to the text directly inside the Scene Editor. It can be activated by double clicking on the rendered text inside the preview window. While the inline editor is active, the text's bounding box turns brown.

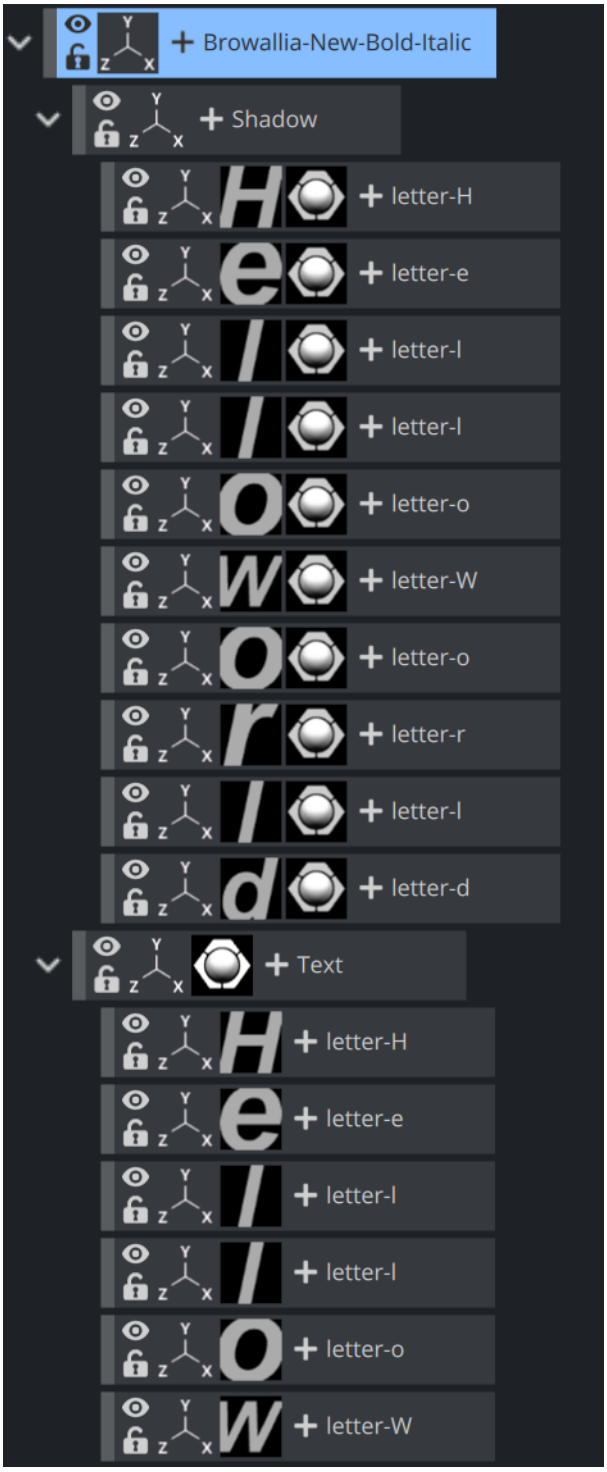
To exit from the inline text editor, click outside the text, or press **ESC** key.

12.3.5 Special Fonts

- **Geometry:** When adding a geometry font, the text is treated like a compound object created from characters. The text can be split to become a group of single character objects (not text!). Splitting of text makes it possible to edit the properties of single character objects. If a Drop shadow had been assigned ot the text, it is put into a separate container.



Copyright © 2026 Vizrt



Note: Be aware that it is a one way operation. It is not possible to merge characters into text again, so by performing the splitting operation, the text cannot be edited anymore.

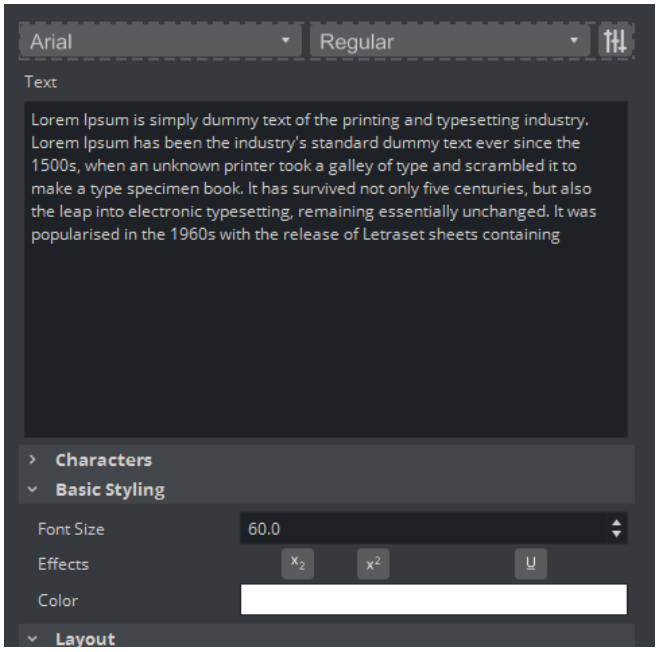
See Also

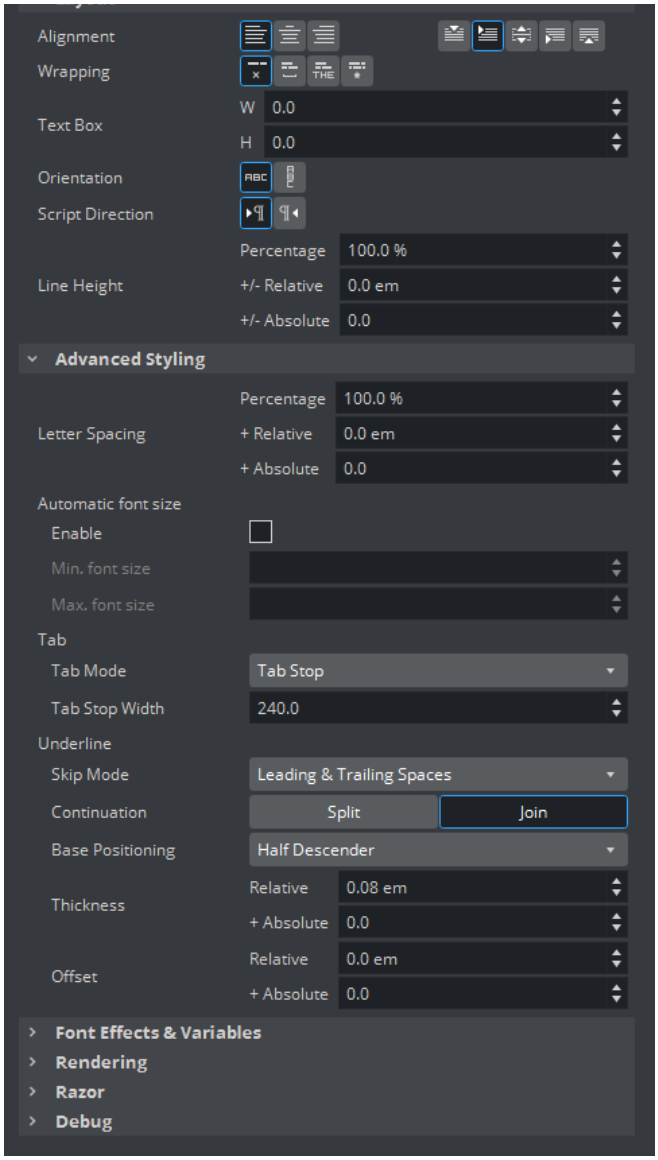
Copyright © 2026 Vizrt

- Fontstyle Editor

12.4 Viz Engine Text Editor

- Text Input
- Font Favorites
- Font Family / Font Style
- Characters
- Basic Styling
    - Font Size
    - Effects
    - Color
- Layout
    - Alignment (Horizontal Orientation)
    - Alignment (Vertical Orientation)
    - Wrapping
    - Text Box
    - Orientation
- Advanced Styling
    - Letter Spacing
    - Automatic Font Size
    - Tab
    - Underline
- Font Effects & Variables
- Rendering
- Razor
- Geometry
- Debug





12.4.1 Text Input

Enter the content of the text here.



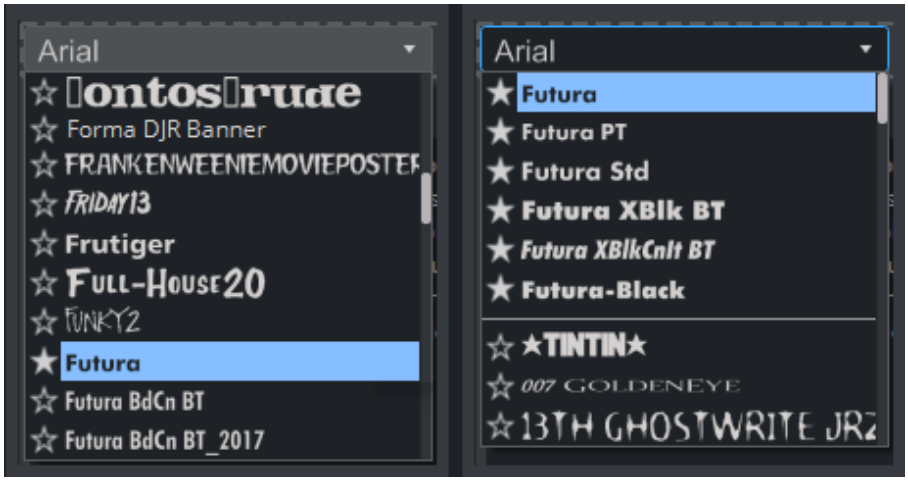
Click on the RichText/Plain Text icon to switch between the mode how text is applied.

In PlainText mode, every change applies directly to the whole text, whereas in Rich Text ediot mode, one needs to select the characters the styling should be applied to.

12.4.2 Font Favorites

You can mark your most used fonts by clicking on the star left to it, adding this font as a favorite. All favorites are listed on the top of the dropdown list.

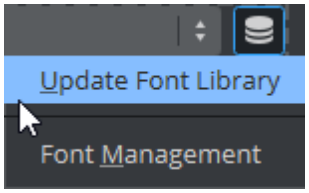
Copyright © 2026 Vizrt



You can also use the dropdown to search for a specific font by typing. The font is highlighted.

12.4.3 Font Family / Font Style

Selects the font family and style to apply to the selected text. The list of the font shows all of the font faces currently stored in the connected Graphic Hub.

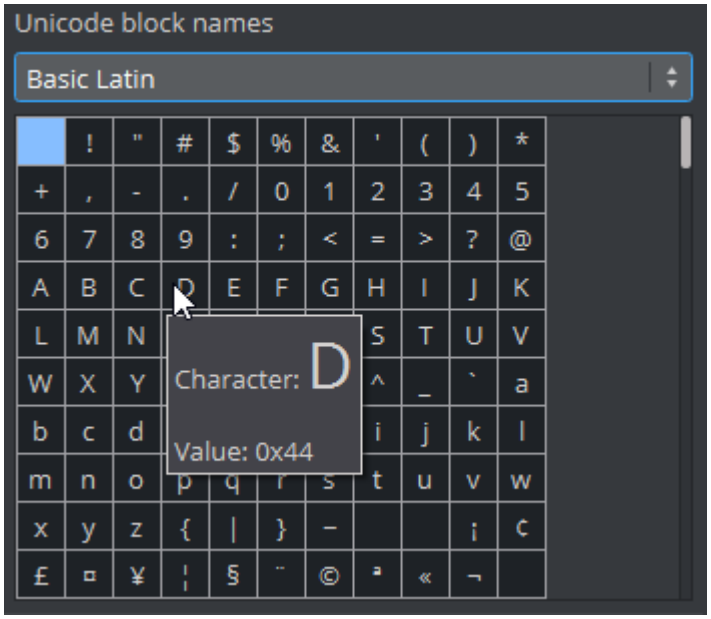


The list may not be up to date if another user on your network imported a new font from a different machine to the same Graphic Hub. In which case, click on the database icon next to the combo box and choose *Update Font Library.*

12.4.4 Characters

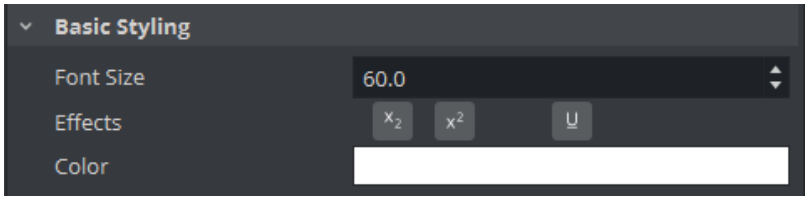
Shows a grouped list of all available glyphs in the selected font.

Copyright © 2026 Vizrt



This can be useful if, for example, the Euro symbol (€) is not mapped correctly in the font map.

12.4.5 Basic Styling



Font Size

Adjusts the size of the selected text. The font size determine how *1EM* unit font is mapped in to the size of the text in the scene.

> **Note:** The resulting size may not exactly match this parameter, however, due to additional interference from the font's internal parameters.

Effects

Applies additional effects or decorations to the selected text.

- **Subscript:** Applies a subscript effect to the selected text.
- **Superscript:** Applies a superscript effect to the selected text.
- **Underline:** Decorates the selected text with underline.

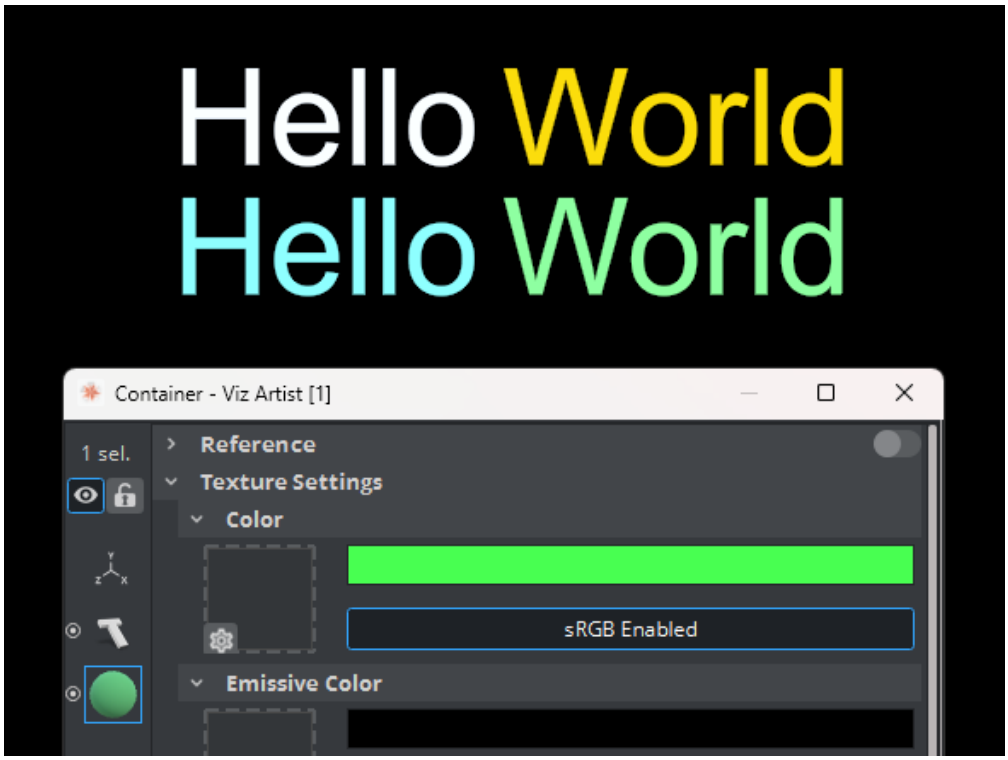
Copyright © 2026 Vizrt

Color

Applies a color and alpha to the selected text. Without any material, the text is rendered with solid color chosen from this option. When using text with a PBR material or a Phong material, the base color of the material and the text color are *multiplied* by their RGBA values.

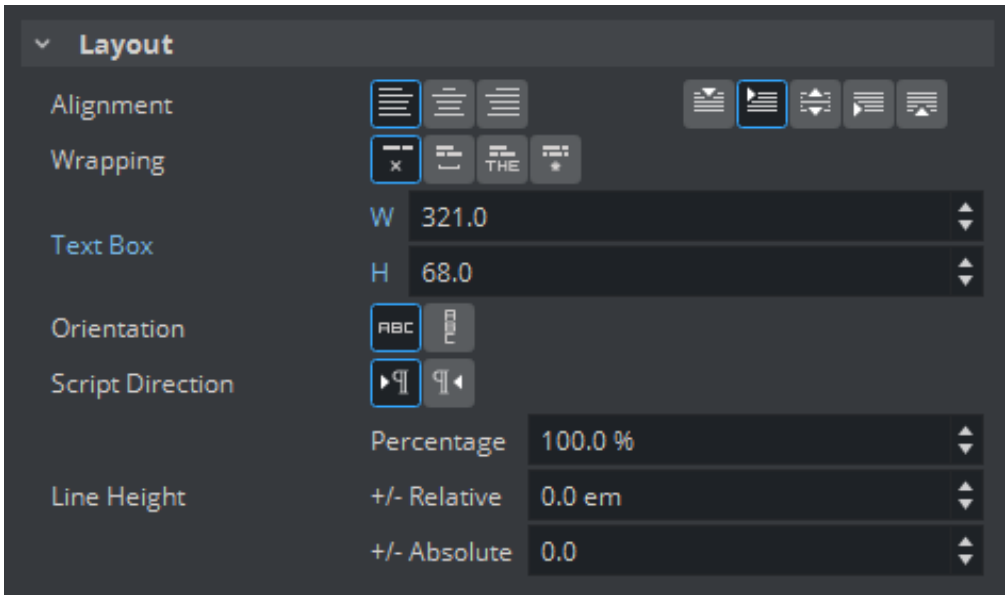
Tip: It is recommended to set either the text color or material color to white, such that the multiplied base color always match the other. This does not determine the final color resulting from lighting and other material's properties, however.

Example of text color interacting with material color, both texts are the exact copies, but the upper one does not have material assigned to its container:



Copyright © 2026 Vizrt

12.4.6 Layout



The alignment depends on how you set your orientation:

Alignment (Horizontal Orientation)

 Sets how the text align in relative to the text box defined in the **Layout** section.

Horizontal Alignment

- **Left:** Aligns the text to the left of the text box.
- **Center:** Centers the text in text box horizontally.
- **Right:** Aligns the text to the right of the text box.

Vertical Alignment

- **Top:** Aligns the text to the tallest element.
- **Middle:** Centers the text vertically.
- **Baseline:** Sets the text to the baseline of the last entry in the textbox.
- **Bottom:** Aligns the text to the lowest element.

Alignment (Vertical Orientation)



Copyright © 2026 Vizrt

Vertical Alignment

- **Top:** Aligns the top of the tallest element to the top of the textbox.
- **Middle:** Centers the text vertically
- **Bottom:** Aligns the bottom of the lowest element to the bottom of the textbox.

Horizontal Alignment

- **Left:** Aligns the text to the left of the text box
- **Center:** Centers the text in text box horizontally
- **Right:** Aligns the text to the right of the text box

Wrapping

 Sets how to wrap text content in the text box.

- **No wrapping:** Disables wrapping.
- **Wrap by space:** Wraps text using only spaces. Works best with languages that separate words by spaces.
- **Wrap by word:** Wraps text based on lexical words. Works best with languages that does not separate words by spaces.
- **Wrap any:** Wraps text at any given point that overflow the text box.

Text Box

Sets size of text box for wrapping and alignment boundary.

> **Tip:** By default, with the text box size of zero, and wrapping disabled, text alignment is working based on the position of the container.

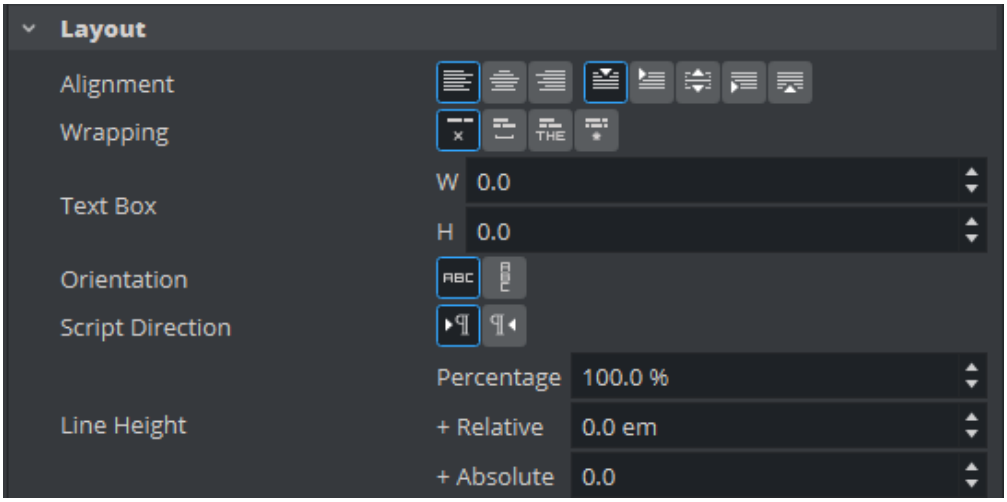
Orientation

 Sets which orientation the text is written in.

Switching the orientation also changes the behavior for the Alignment, Script direction/Column settings and distance options:

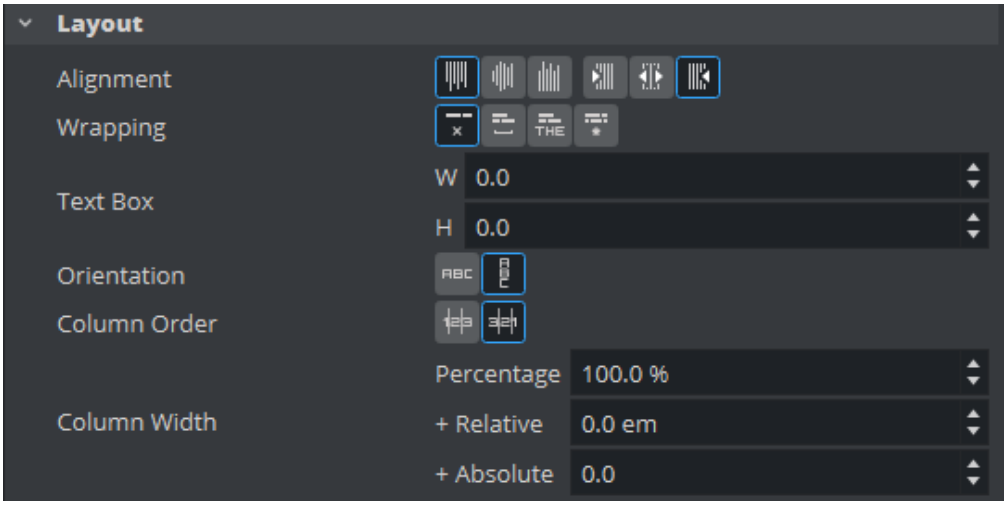
Horizontal

Copyright © 2026 Vizrt



Writes text horizontally, with each new line started below the previous one.

Vertical



Writes text vertically from top to bottom.

Script Direction

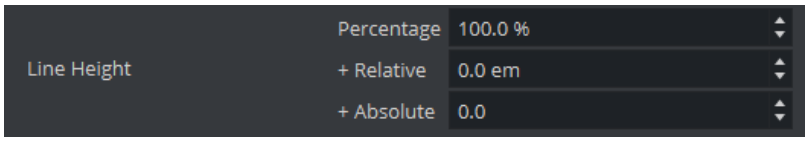
Sets the primary writing direction of the script used in the text so that the words are ordered linguistically correct. This option only available when **Horizontal Orientation** is selected.

- **Left to right:** The primary direction of the script is written from left to right. Language using this script direction includes, but not limited to, English, German, Chinese, Thai, etc.
- **Right to left:** The primary direction of the script is written from right to left. Languages using this script direction includes, but not limited to, Arabic, Persian, Hebrew, Yiddish, etc.

Copyright © 2026 Vizrt

Note: Entering text of different script direction from the setting does not automatically make the text written in reverse direction, this only make the text become *bi-directional*.

Line Height



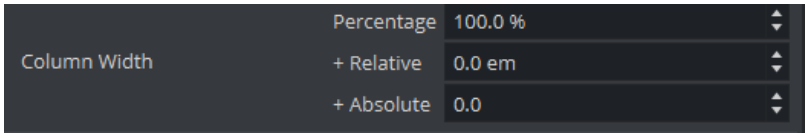
Defines the distances between lines in **percentage**, relative to the font size (**em**) or in absolute unit. The values are added.
Example: 50% of the default line height + 5em + 10 absolute units.

Column Order

Sets where each new line is appended compared to the previous line. This option only available when **Vertical Orientation** is selected.

- **Left to right:** Appends each new line to the right of previous line.
- **Right to left:** Appends each new line to the left of previous line.

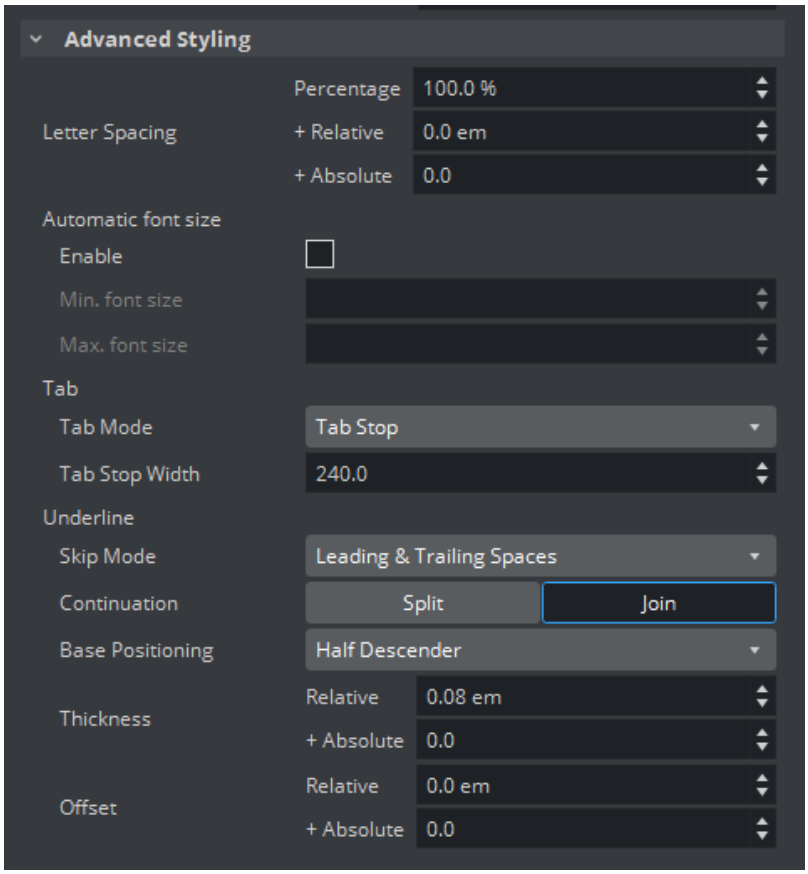
Column Width



Defines the distances between columns in **percentage**, relative to the font size (**em**) or in absolute units. The values are added.

Copyright © 2026 Vizrt

12.4.7 Advanced Styling



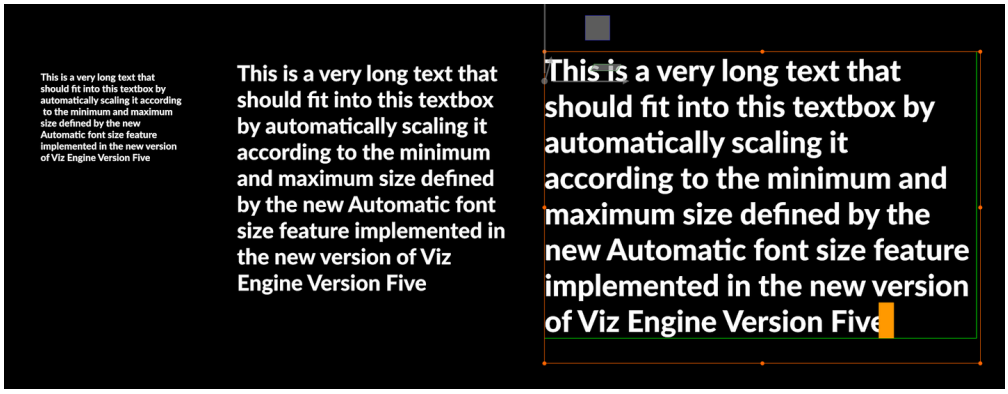
Letter Spacing

Allows to add additional spacing between characters by either **percentage** relative in (**units**) or in **absolute values**.

Automatic Font Size

Allows to scale a text into the Textbox, similar to the TextAutoScale in the Classic Render Pipeline.

Copyright © 2026 Vizrt



Minimum and **Maximum** size can be defined.

Tab

Tab Mode

- **Tab Stop:** Makes use of tab stops, similar to how word processing software works. The separation of the tab stops can be adjusted through the **Tab Stop Width** option.
- **Space:** Treats any tab character like a space character, it behaves the same as a normal space character (including spacing and word wrapping).
- **Glyph:** Lets the font decide how to interpret the tab character. Result depends on the font.

Underline

Skip Mode

Sets whether underline should be skipped for spaces, even if the space is marked to be underlined.

- **None:** Underlines everything.
- **Leading and Trailing Spaces:** Spaces at the start and at the end of a line are not underlined.
- **All Spaces:** All space characters are not underlined.

Continuation

- **Split:** Generates separated rectangular geometries for every glyph that is marked to have the underline.
- **Join:** Generates a single rectangular geometry, regardless of the glyph for any consecutive underline with the same width, offset and color (default).

Base Positioning

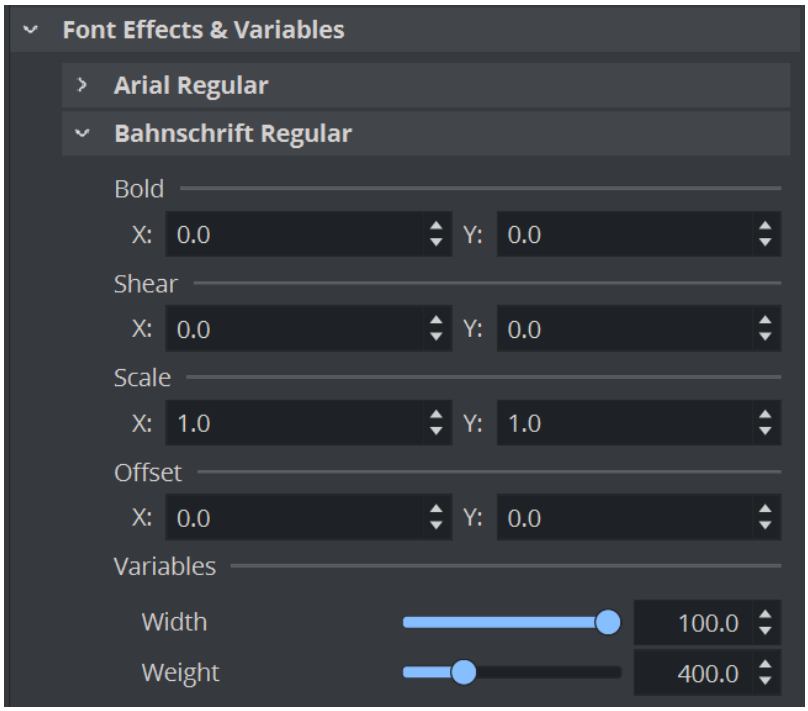
The underline is drawn based on the following settings:

Copyright © 2026 Vizrt

Setting	Sample
Baseline	A sample text
Half Descender	A sample text
Descender	A sample text

The position of the underline can also be changed by using the **Offset** setting. Thickness allows to set the strength of the underline in either **relative** (em) or **absolute** units.

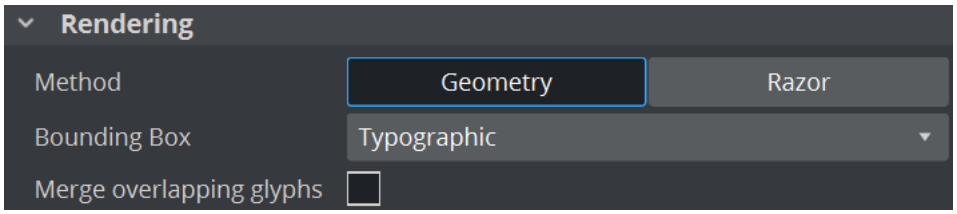
12.4.8 Font Effects & Variables



Allows to modify fonts if they for example missing a bold or italic variant. When using variable fonts, some additional properties are shown here. These settings are unique for every single font.

Copyright © 2026 Vizrt

12.4.9 Rendering

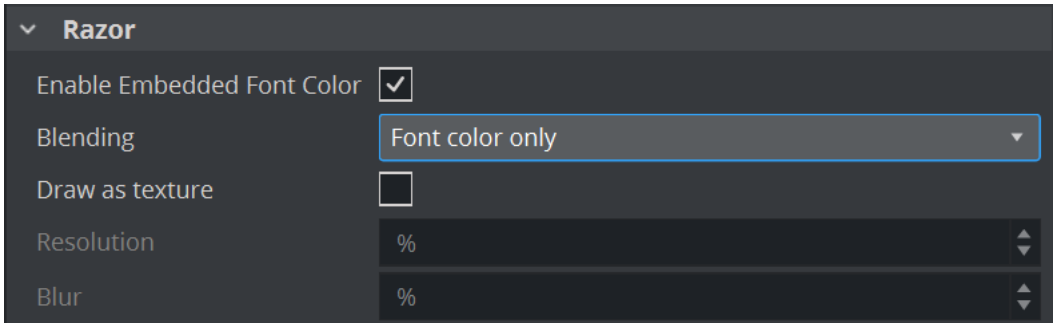


- **Method:** Selects how to render the text.
  - **Geometry:** Renders the text as standard geometry with polygons. Selecting this option enables Geometry tab.
  - **Razor:** Renders the text as vector graphics on flat plane. Selecting this option enables Razor tab.
- **Bounding Box:** Selects how bounding box should be calculated.
  - **Typographic:** Calculates bounding box based on logical metrics of the fonts used in the text.
  - **Geometric:** Calculates bounding box based on visually visible geometry of the glyphs.
  - **Text Box:** Uses the text box as the bounding box.
- **Merge overlapping glyphs**
  - When adding a transparency to some languages, you'll see a visual overlap on some glyphs:

Glyphs overlapping	Merged overlapping glyphs
	

This feature is currently enabled for Razor fonts only

12.4.10 Razor



Copyright © 2026 Vizrt

This tab is accessible only when the rendering method is set to **Razor**.

Enable Embedded Font Color

This option allows the usage of font color if the font being used in the text contain the OpenType CPAL and COLR table. The **Blending** option can be used to control how the font color interacts with the text color (set in the Basic Styling tab).

- **Font color only:** Uses the color from the font only and ignores the text color.
- **Multiply:** Uses the result of multiplication of RGBA value between the font color and the text color.
- **Text color only:** Ignores the font color and use only the text color.

Tip: The Text color only option may not sound useful. However, when enabling font color, the shape of the glyph may change. You can use it when you want to the text as a shadow of another text with font color enabled, for example.

Blending

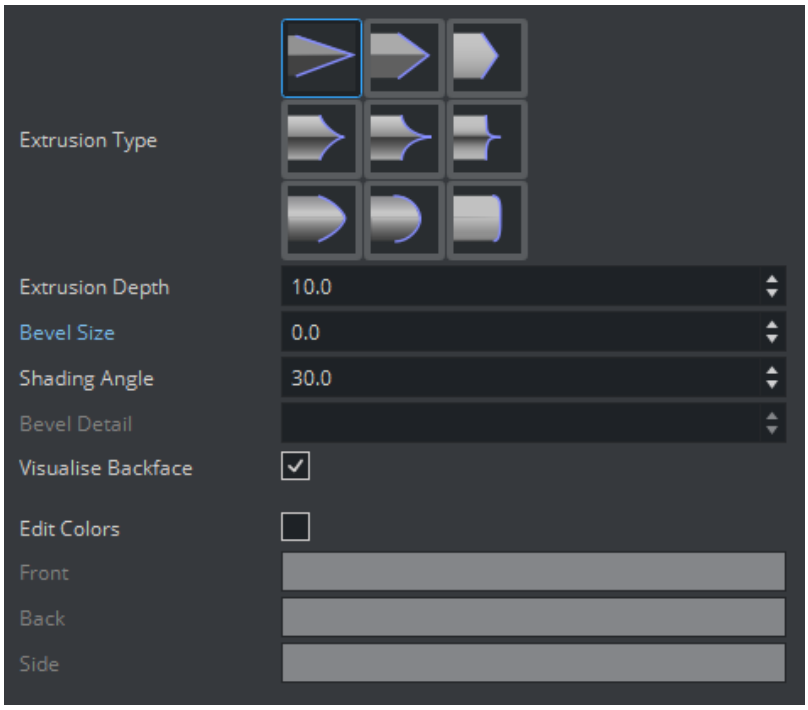
Changes how the various colors of a text are blended.

Draw as Texture

Draws the text with Razor method on a temporary offscreen texture before drawing it back into the scene. This option helps increase performance, in exchange for lower rendering quality of the text. The **Level of Details** option can be used to control how detailed the texture should be. Change **Resolution** and **Blur** level in percent.

Copyright © 2026 Vizrt

12.4.11 Geometry



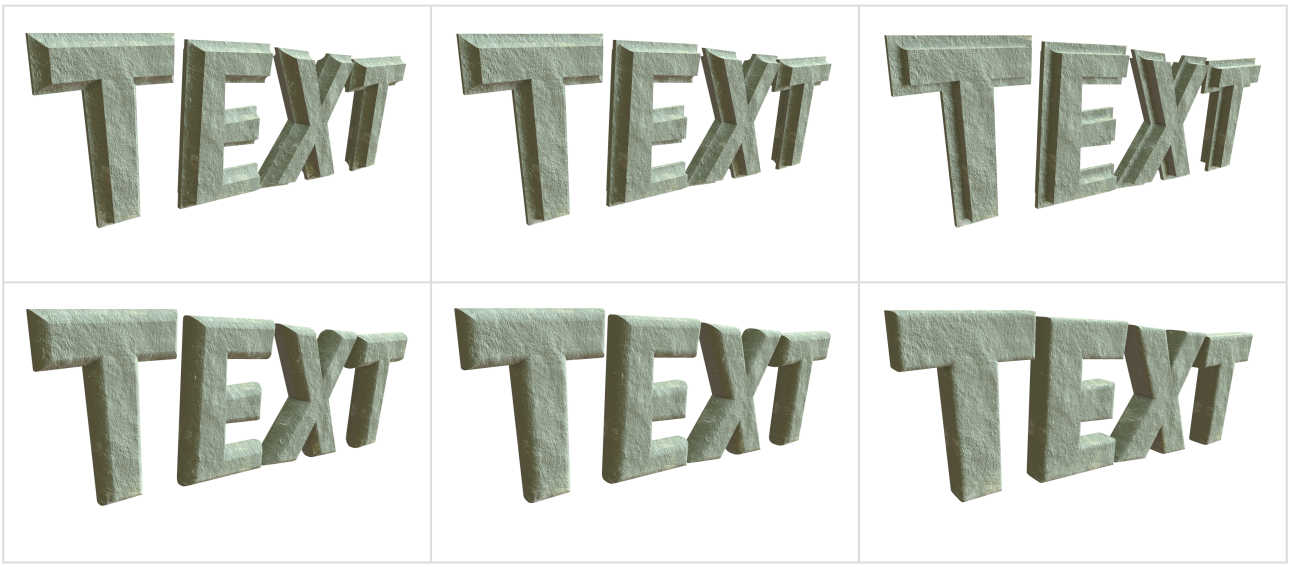
This tab is accessible only when the rendering method is set to **Geometry**.

Extrusion

To use Extrusion with the Geometry text, **Enable Extrusion**. Choose how your extrusion should look and tweak it by setting the **Depth**, **Size** and the **Shading Angle**. **Visualise Backface** also applies the bevel on the usually hidden backface of your text.



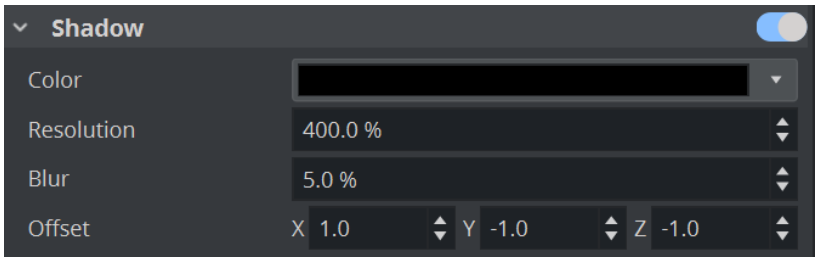
Copyright © 2026 Vizrt



Information: Extrusion was updated from classic style to a newer style in Viz Engine 5.4.0. The result in a new scene may appear slightly different from a scene created prior to the mentioned version. To switch between styles, use the option **Text use classic Geometry Extrusion** from the Rendering Panel.

Shadow





Introduced in Viz Engine 5.5, this options allows for a much more easier creation of a drop shadow effect. The text is rendered again as a texture and can be placed as part of the original text, no matter if it is Geometry or Razor.

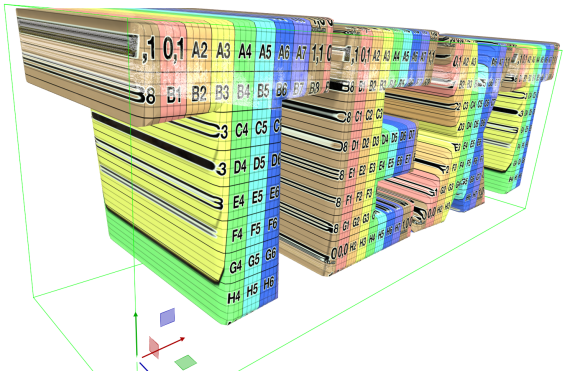
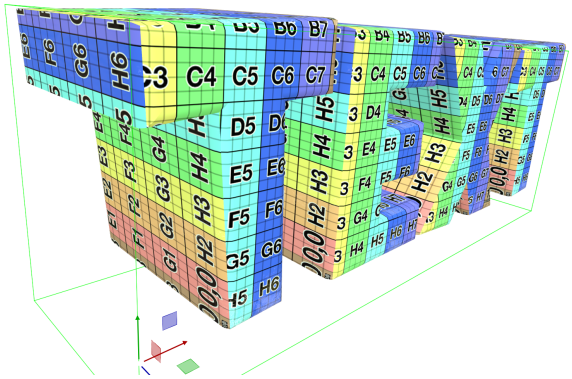
The quality of the text is defined by the *Resolution* factor, the sharpness of the drop shadow effect can be modified using the *Blur* value. Change the placement using the offset coordinates.

Copyright © 2026 Vizrt

Note: The Drop Shadow feature renders a second texture (uninfluenced by any other lights within the scene) as part of the original text.

Texture Coordinates

If text geometries are going to be extruded and textured, it is recommended to turn on Cubic Mode. This generates the correct UV coordinates to allow textures to be correctly placed on extruded text objects.

Cubic Mode disabled	Cubic Mode enabled
	

- **Enable Cubic Mode:** Enables generation of texture coordinates based on six projection planes forming. The face normal of a triangle is used to determine which projection plane is used.
- **Cubic**
  - **Scaling:** Changes the scaling of the projection planes, applied while calculating the projected UVs.
  - **Translation:** Changes the position of the projection planes, applied while calculating the projected UVs.
  - **UV Rotation:** Applies rotation on the generated UVs, with each axis representing a major axis of the projection plane.

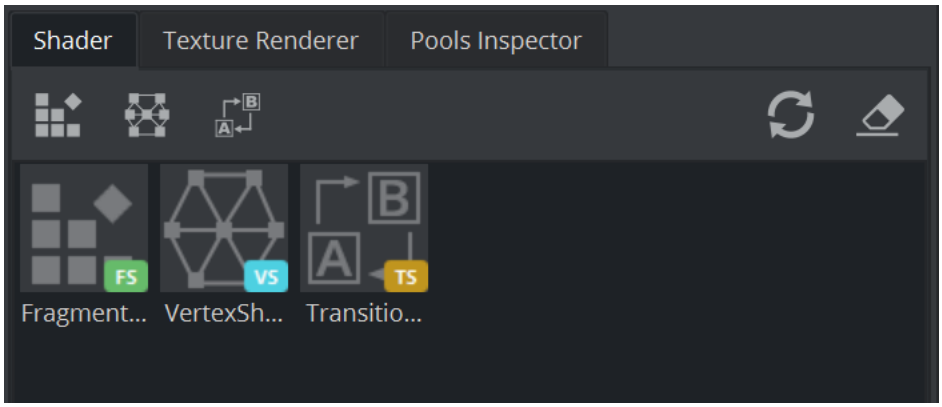
12.4.12 Debug

- **Preview Font:** Allows changing the font used to display the input field.
- **Preview Size:** Allows changing the font size used in the input field.
- **Text Data:** Helps track down issues with Text objects. Only used internally.

Copyright © 2026 Vizrt

12.5 Workspace Editor

The Workspace editor is a central place to create and modify any kind of material definitions like special textures and shaders.



These include:

12.5.1 Shaders

- Fragment and Vertex Shaders
- Transition Shader

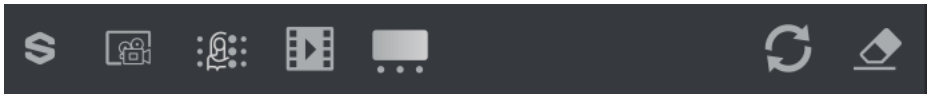
12.5.2 Textures

- Substance Materials
- SceneView Texture
- Texture Renderer
- ClipPlayer Texture

Copyright © 2026 Vizrt

12.5.3 Texture Renderer

- Substance Texture Renderer
- SceneView Texture Renderer
    - Example
- Preprocessing Texture Renderer
    - Using Effects
    - Animating Effects
- Clipplayer
- Gradient Editor



Substance Texture Renderer

Please see the details at Substance Materials.

SceneView Texture Renderer

Renders the view of a certain camera into the texture. This can be used for creating effects like mirrors.

Example



Copyright © 2026 Vizrt

The image in the window shows the rendered output of camera 2, which is looking out of the window. The content of the outside is mirrored inside the window. It uses the Texture Slot plug-in to blend into the existing window texture.

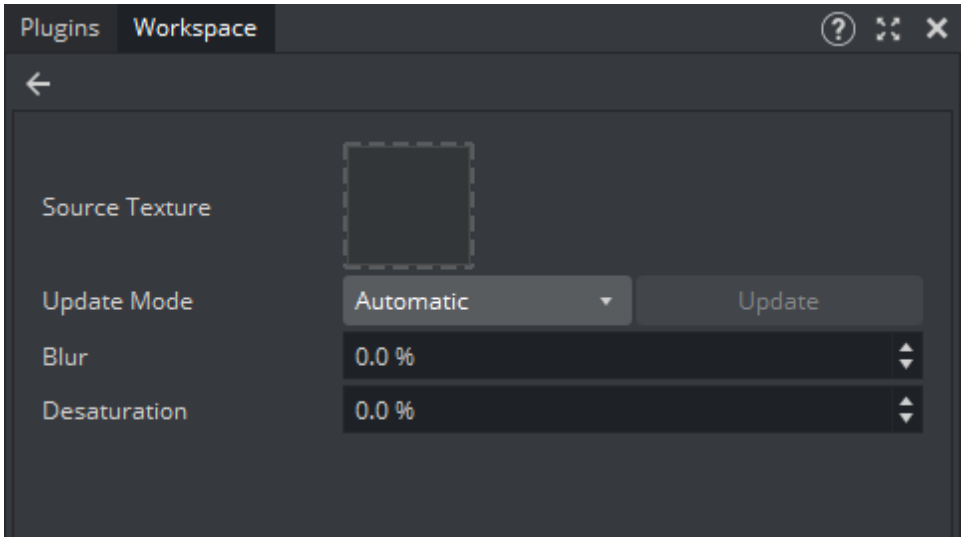
Preprocessing Texture Renderer

Adds a blur and desaturation effect to a texture.

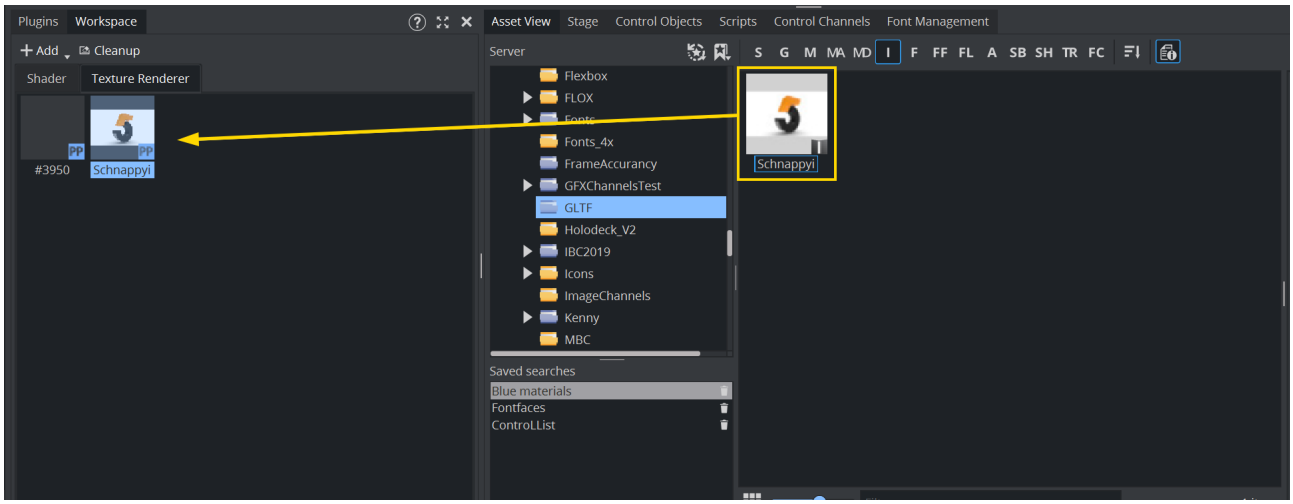


To use this type of texture renderer, click on the icon. This creates a workspace object with the blue *PP* tag at the right bottom.

Double click on it, and the Workspace dock shows the properties of the texture renderer. After that, drag and drop a texture in the Source Texture box.

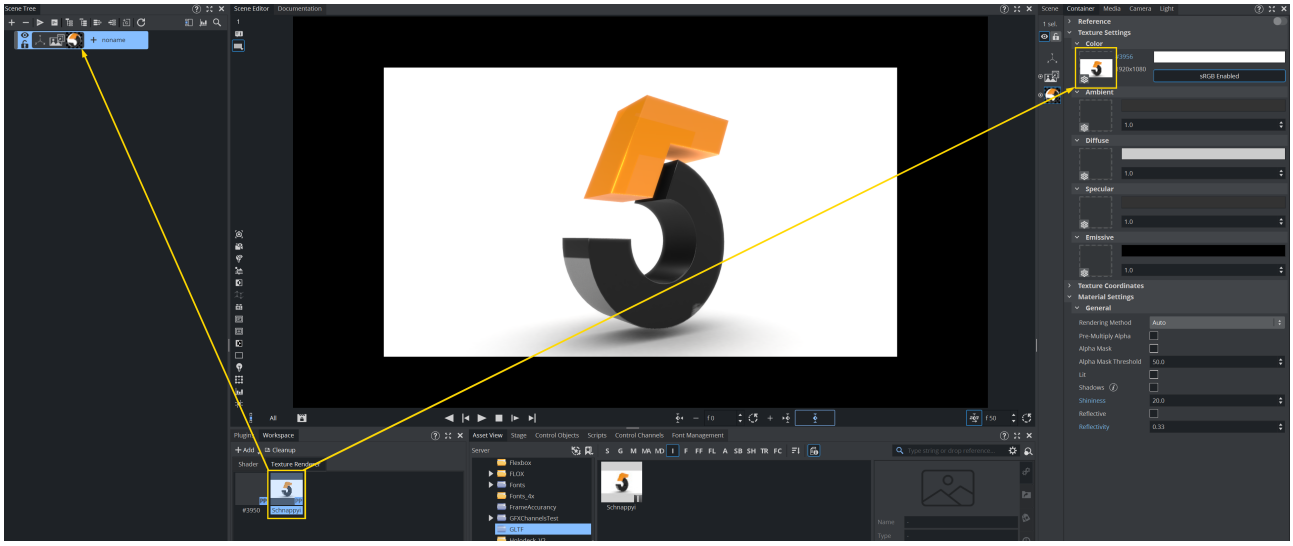


Alternatively, drag an image asset from Asset View to the Texture Renderer Workspace.



Copyright © 2026 Vizrt

To apply the texture renderer to the container, either drag the texture renderer to any texture input box in a material or Texture Slot, or drag it to the container in Scene Tree dock. The latter automatically creates an unlit Phong material using the texture as a Base Color for that container.



Info: A texture renderer can also be dragged into Scene Editor to make a rectangle geometry with the texture renderer applied.

- In case of a Substance texture renderer, a PBR material is created for the rectangle.
- In case of a Preprocessing texture renderer, an unlit Phong material is created for the rectangle.

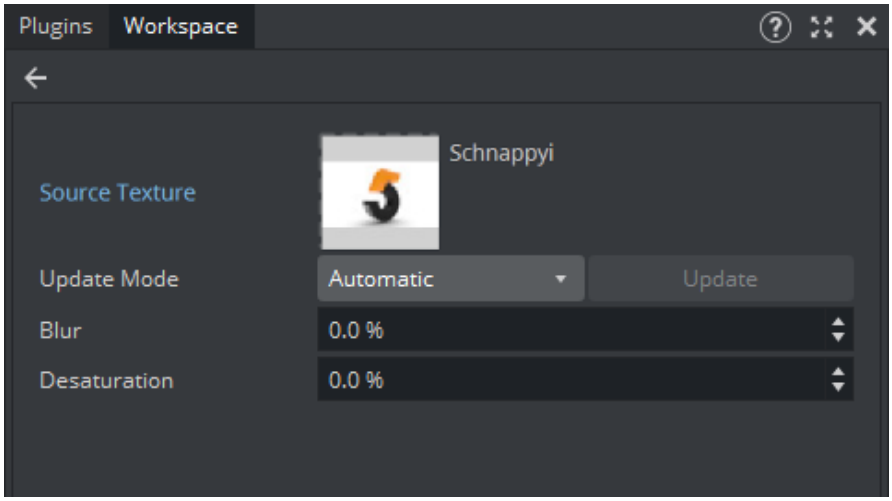
Using Effects

These are the available effects in Preprocessing Texture Renderer:

- **Blur:** Blurs the source texture until its content is barely visible and recognizable.
- **Desaturation:** Reduces the color saturation of the source texture until it becomes monochrome.

To adjust the effect, double click in the texture renderer to examine its properties. The effect can be adjusted from 0% to 100%.

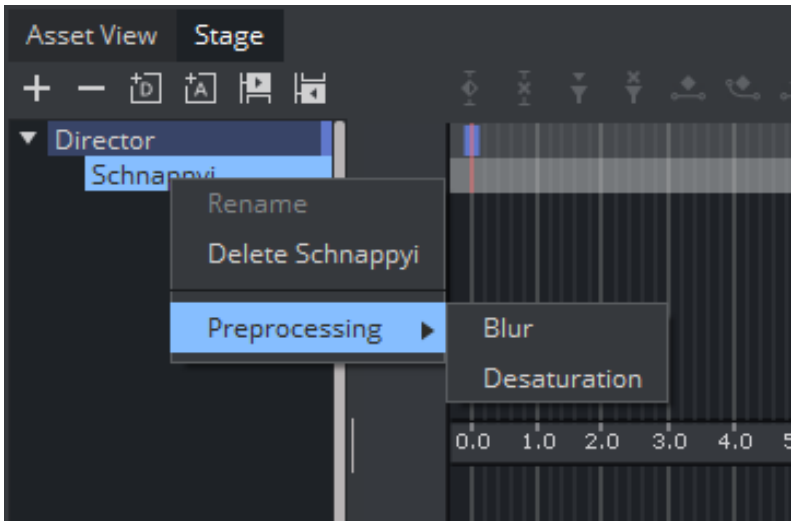
Copyright © 2026 Vizrt



Note: Changing the values affects the container while using it in real-time, unless the **Update Mode** is not set to **Manual**.

Animating Effects

To animate the effect, open the Stage dock and find a director where the texture renderer is going to be manipulated. After that, drag the texture renderer to the director to create the actor for the texture renderer.



Right click on the actor, select *Preprocessing* and choose the effect to be animated. There are a set of keyframes initialized. See further details about how to manipulate an animation at The Stage for Animation.

Clipplayer

See ClipPlayer Texture Renderer

Copyright © 2026 Vizrt

Gradient Editor

See Gradient Editor

Copyright © 2026 Vizrt

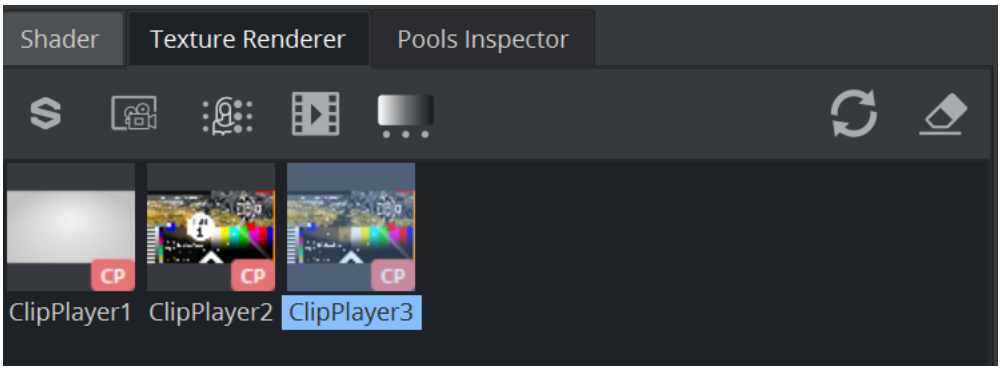
ClipPlayer Texture Renderer

- ClipPlayer Texture Renderer Parameters
- Getting Started
- Stage Integration
- NVDEC Support for Clip Texture Renderer
    - Technical Requirements
    - Configuration
- HAP Support for Clip Texture Renderer
- Known Differences with SoftClip

The ClipPlayer Texture Renderer is a container-based clip playback solution like SoftClip, MoViz, or VLC plug-ins. Unlike the mentioned plug-ins that can only be used in the Classic Render Pipeline, the ClipPlayer Texture Renderer can only be played in the Viz Engine Render Pipeline. The clips played are saved as part of the scene and do not form part of the global clip channel pool. The clips contained in the scene do not interfere with other scenes and can be combined with TL (Transition Logic). The clip playback is controlled externally via commands or Stage.

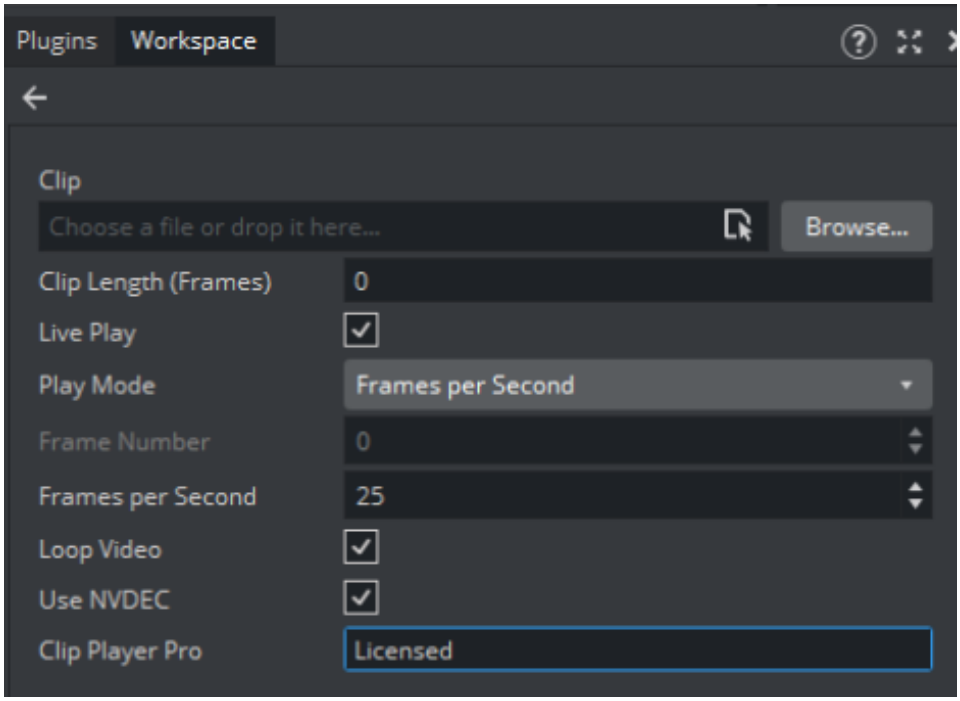


Click on the icon to create a new instance.



After adding the ClipPlayer Texture Renderer component, it is possible to load a clip and perform other operations after ***double-clicking*** the added component.

Copyright © 2026 Vizrt

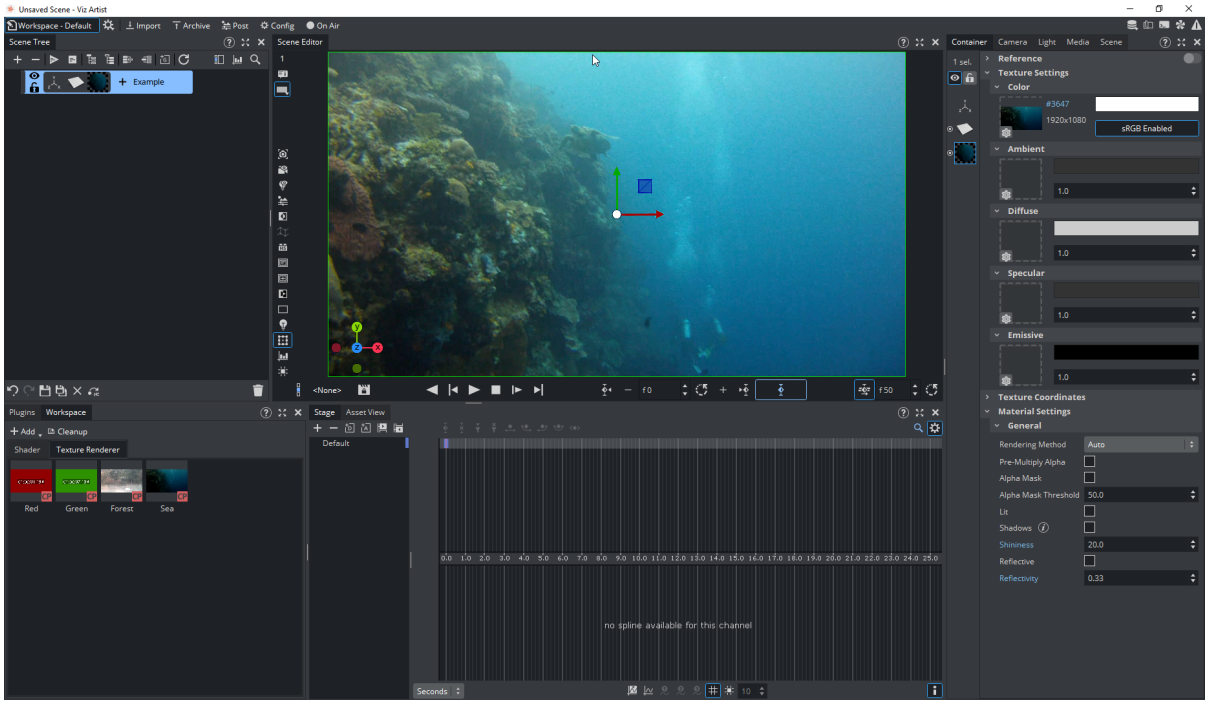


ClipPlayer Texture Renderer Parameters

- **Clip:** Path of the clip to be opened and played.
- **Clip Length (Frames):** Length of the loaded clip.
- **Live Play:** Guarantees the clip is played and synchronized to the frame rate of the Viz Engine. This is set in the **Output Format** section of Viz Configuration (see the **Configuring Viz** section of the Viz Engine Administrator Guide). Disable the Live Play to allow the clip playback to be Key Frame animated.
- **Play mode**
  - **Synchronized:** Runs the clip synchronized with the scene playback. This is set in the **Output Format** section of Viz Configuration.
  - **Frames per Second:** Adjusts for different speeds, set in the next parameter.
- **Frame Number:** Enables the frame numbers to be manually animated. This is an alternative to playing the whole clip (available when **Live Play** is set to `Off`).
- **Frames per Second:** Sets the frame rate (available when **Play mode** is **Frames per second**).
- **Loop Video:** Loops the video automatically when enabled.
- **Use NVDEC:** Uses NVIDIA extensions to perform any decoding processes on the GPU (faster) rather than in CPU.
- **Clip Player Pro:** Shows license options for the Clip Player Pro licenses. This allows additional formats to be playable.

Copyright © 2026 Vizrt

Getting Started

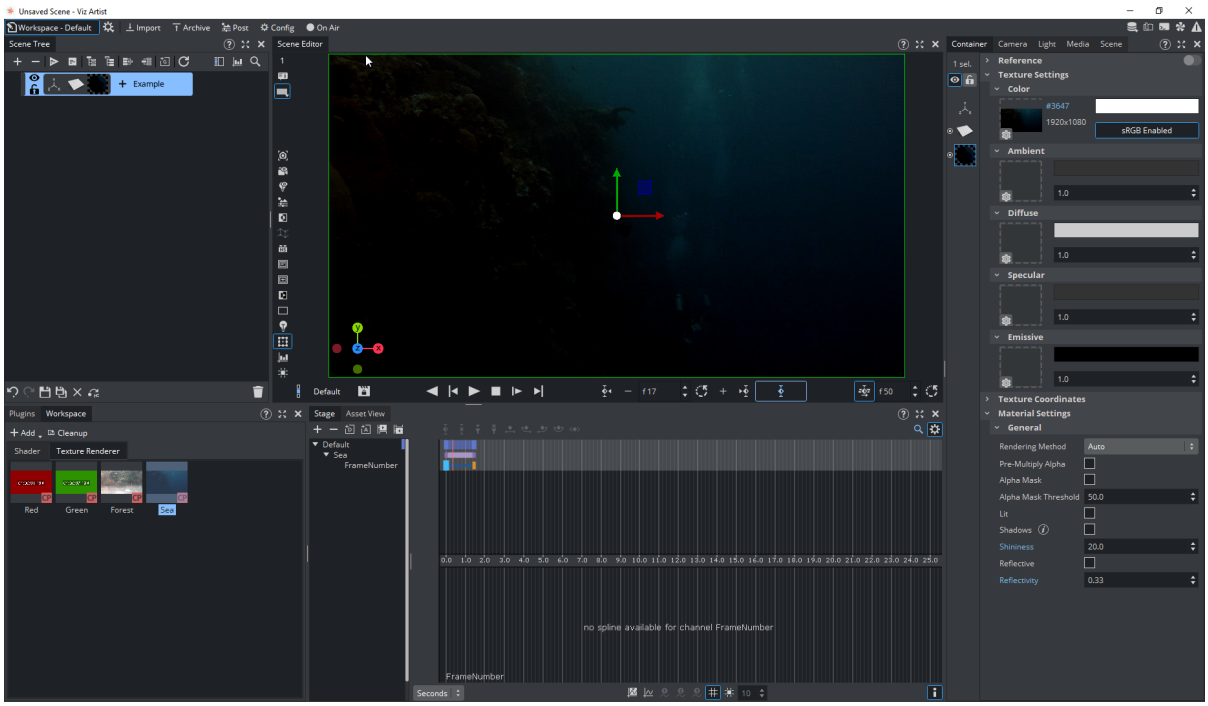


To be able to see the clip playback in Viz Artist preview, a Phong Material is needed. The following is the workflow:

- Add a new ClipPlayer Texture Renderer.
- Double-click the component in Workspace to access the ClipPlayer Texture Renderer parameters.
- Load a clip using the **Browse...** button.
- Drag a Rectangle to the Scene Tree (any other shape can be chosen).
- Drag a Phong Material to the Rectangle.
- Now drag ClipPlayer Texture Renderer from the Workspace tab to the Phong Material and Rectangle.

Copyright © 2026 Vizrt

Stage Integration



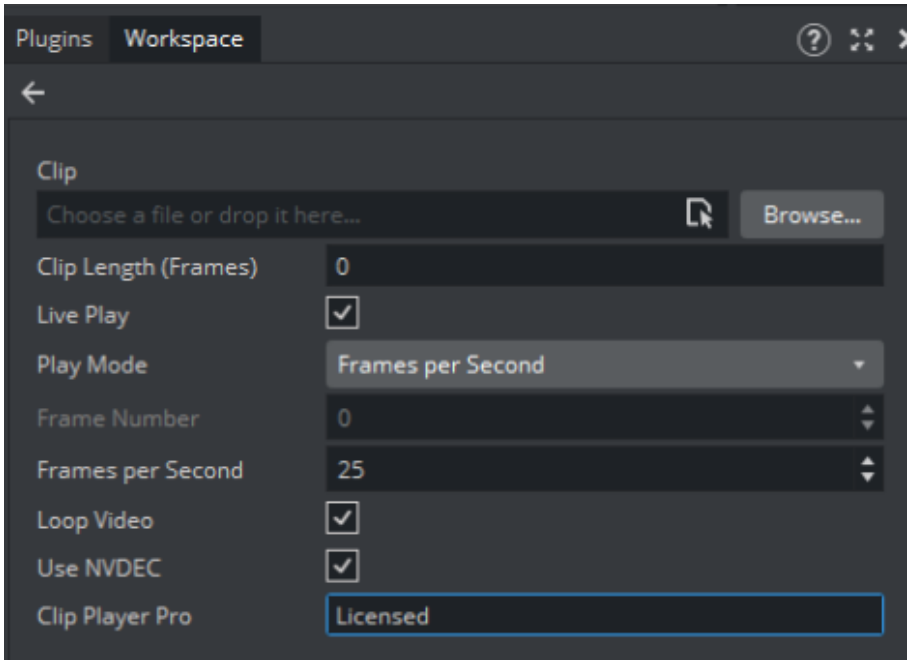
It is possible to animate the clip just like in the SoftClip plug-in, via the Stage:

- Create a new director in the Stage.
- Drag the ClipPlayer Texture Renderer to the Director.
- Right-click the newly created element below the Director.
- Add FrameNumber.

NVDEC Support for Clip Texture Renderer

With enabled hardware acceleration, the Clip Texture Renderer now makes use of VML clip player's NVDEC support. This implementation allows for hardware-accelerated video decoding, improving performance and efficiency.

Copyright © 2026 Vizrt



Technical Requirements

- Compatible NVIDIA GPU
- Latest graphics drivers
- Hardware acceleration enabled in application settings
- Clip Player Pro license

Configuration

- Open the Workspace panel.
- Navigate to the **Clip** section by double clicking on the **ClipPlayer Texture Renderer** component.
- Check the *Use NVDEC* checkbox to enable hardware acceleration.
- Ensure *Clip Player Pro* shows *Licensed* status.

HAP Support for Clip Texture Renderer

A Clip Player Pro license must be available in order to play HAP-based clips. The supported formats are:

Texture Format(s)	Human-Readable Name	Four-Character Code
RGB DXT1/BC1	Hap	Hap1
RGBA DXT5/BC3	Hap Alpha	Hap5
Scaled YCoCg DXT5/BC3	Hap Q	HapY

Copyright © 2026 Vizrt

Known Differences with SoftClip

ClipPlayer Texture Renderer has been implemented as an alternative to the SoftClip plug-in to be used in Viz Engine Renderer, however, not all features of the SoftClip plug-in have been implemented. Most functionalities related to performance found in SoftClip, are deemed unnecessary and have been removed (such as *preload frame count*, *use decode thread*, *preferred CPU*, *priority*, etc.). Other parameters like *force opaque* are now controlled via the Phong Material, associated under the **Material Settings** > **General** tab.

The backend implementation that allows the playback of the clip is different than the one used in the SoftClip plug-in. This means that not all clips that are playable in SoftClip are playable in ClipPlayer Texture Renderer, and vice-versa. One important difference is that the SoftClip plug-in supports *Matrox IFrame with alpha*, while opening this format in ClipPlayer Texture Renderer results in the alpha channel being discarded.

Another difference is how the colorimetry of the clip is dealt with. In the SoftClip plug-in, if the clip does not have metadata regarding colorimetry, then it defaults to Rec.601, but in ClipPlayer Texture Renderer it uses Rec.709 instead. This results in a different color if the same clip is opened and the output between the SoftClip plug-in and ClipPlayer Texture Renderer is compared.

To streamline operation and design, all clips are deinterlaced and set to match the output frame rate of the configured Viz Engine. Consequently, if a clip is interlaced (i25), one second is equivalent to 50 frames instead of 25. The same applies to progressive clips (p25), where one second also equals 50 frames rather than 25. This should be carefully considered when designing the stage.

ClipPlayer Texture Renderer supports frame-accurate playback with timed commands.

Copyright © 2026 Vizrt

Gradient Editor

The Gradient Texture Renderer draws a color gradient as a texture, so any material in a scene can use a linear, radial, angular, reflected or diamond gradient without an image file. You set the colors once in a color ramp, pick a style, and every setting can be changed live or animated.



Click on the icon to add a new gradient. Then double click it to change its settings. Drag the gradient to any texture drop area to use it.



A new Gradient starts as a black-to-white **Linear** gradient at 45°, centered, 512 × 512 px. For a flat, poster-like look, switch off lighting on the material (unlit), otherwise scene lights shade the gradient.

Gradient types

The Type decides the shape the color ramp is laid along; the same ramp gives five very different looks.



Copyright © 2026 Vizrt

Type	Shape	Settings that matter
Linear	Straight bands, ramp runs from the Origin along the Angle	Angle, Origin
Radial	Circles around the Origin	Origin, Radius
Angular	A sweep around the Origin, like a clock hand; starts at the Angle and runs counterclockwise once around	Angle, Origin
Reflected	Linear, mirrored on both sides of a line through the Origin	Angle, Origin
Diamond	Squares turned 45° around the Origin	Origin, Radius, Angle rotates it

Radial and Diamond reach the last stop at the Radius. Linear and Reflected ignore Radius and reach it one texture width from the Origin, which is why a centered Reflected gradient only shows the first half (0–50 %) of the ramp.

Settings

Every setting can be changed live; Angle and Radius can also be keyframed.

Setting (editor)	What it does	Used by	Default	Range
Type	Gradient style	all	Linear	Linear, Radial, Angular, Reflected, Diamond
Angle	Direction in degrees: 0° runs left to right, 90° bottom to top	Linear, Reflected, Angular (start line), Diamond (rotation)	45°	any value, animatable
Origin X / Y	Start point or center, in texture space: 0 0 = bottom left, 1 1 = top right. Buttons L / C / R and B / C / T snap to edges and center	all	0.5 0.5	any value, outside 0–1 moves it off the texture
Radius	Distance at which the last color is reached, in texture widths	Radial, Diamond	1	0 or more, animatable

Copyright © 2026 Vizrt

Setting (editor)	What it does	Used by	Default	Range
Resolution	Size of the generated texture	all	512 × 512 px	1–16384 px per side
Interpolation	How colors blend between stops: RGB or OKLab	all	RGB	RGB, OKLab
Update mode	Auto redraws on every change, Manual only on request	all	Auto	Auto, Manual
Color Ramp	The colors and where they sit	all	black at 0 % → white at 100 %	any number of stops

Linear and Reflected start the ramp **at the Origin** and reach the last stop one texture width further along the Angle; everything behind the Origin shows the first color. For an edge-to-edge linear gradient, put the Origin on the edge the gradient starts from (for example Angle 0° with Origin X = 0).

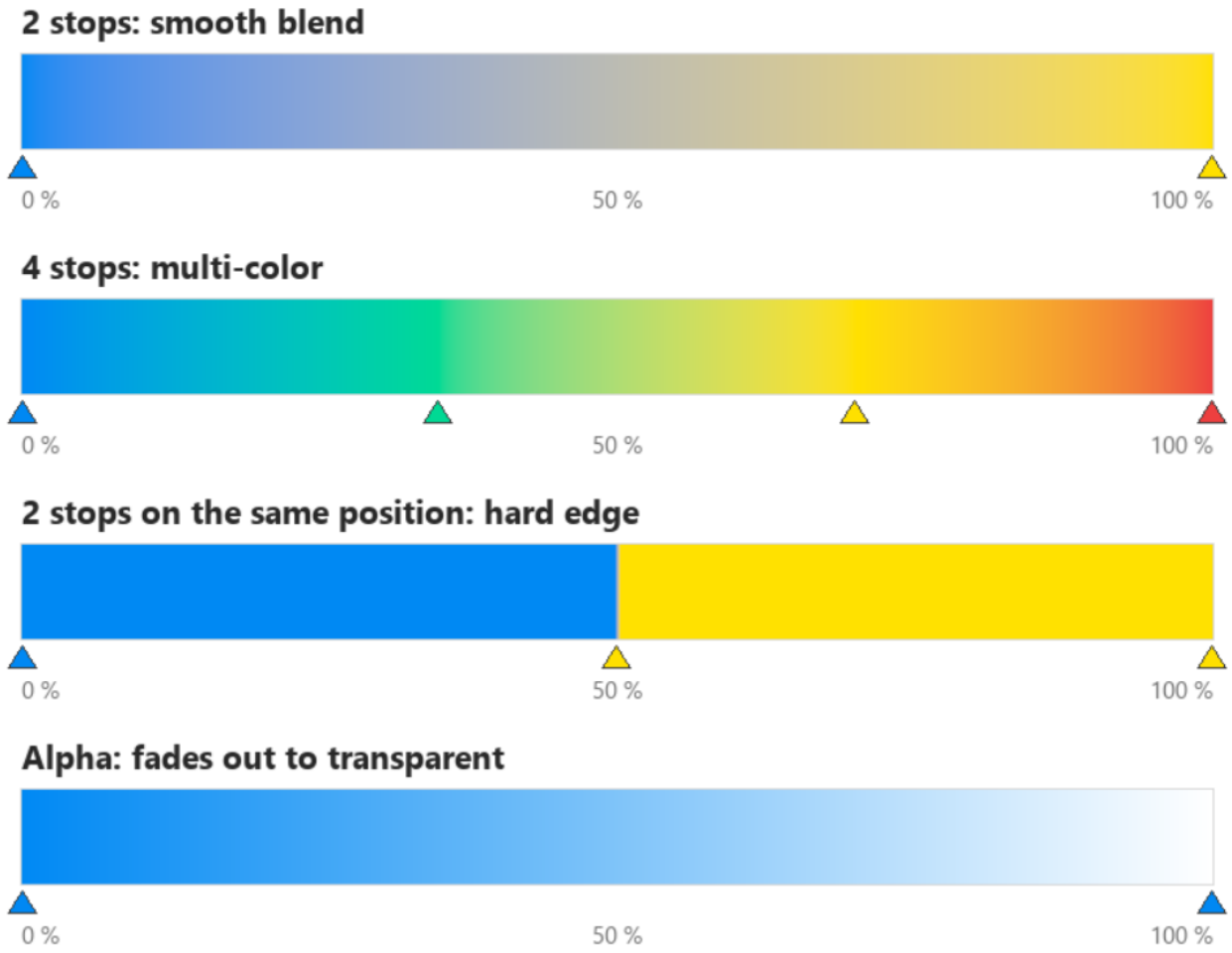
Colors: the color ramp

The color ramp is a bar of **stops**; each stop is a color (with alpha) at a position from 0 % to 100 %, and the gradient blends between neighbouring stops.

In the Color Ramp editor:

- **Add a stop**: double-click an empty spot on the bar, or use the add button (adds one at 50 %).
- **Select and move**: click a marker and drag it, or type its **Position** in %.
- **Change its color**: pick it in the color selector next to the bar; alpha is part of the color.
- **Remove a stop**: select it and press Delete (the last remaining stop cannot be removed).

Copyright © 2026 Vizrt

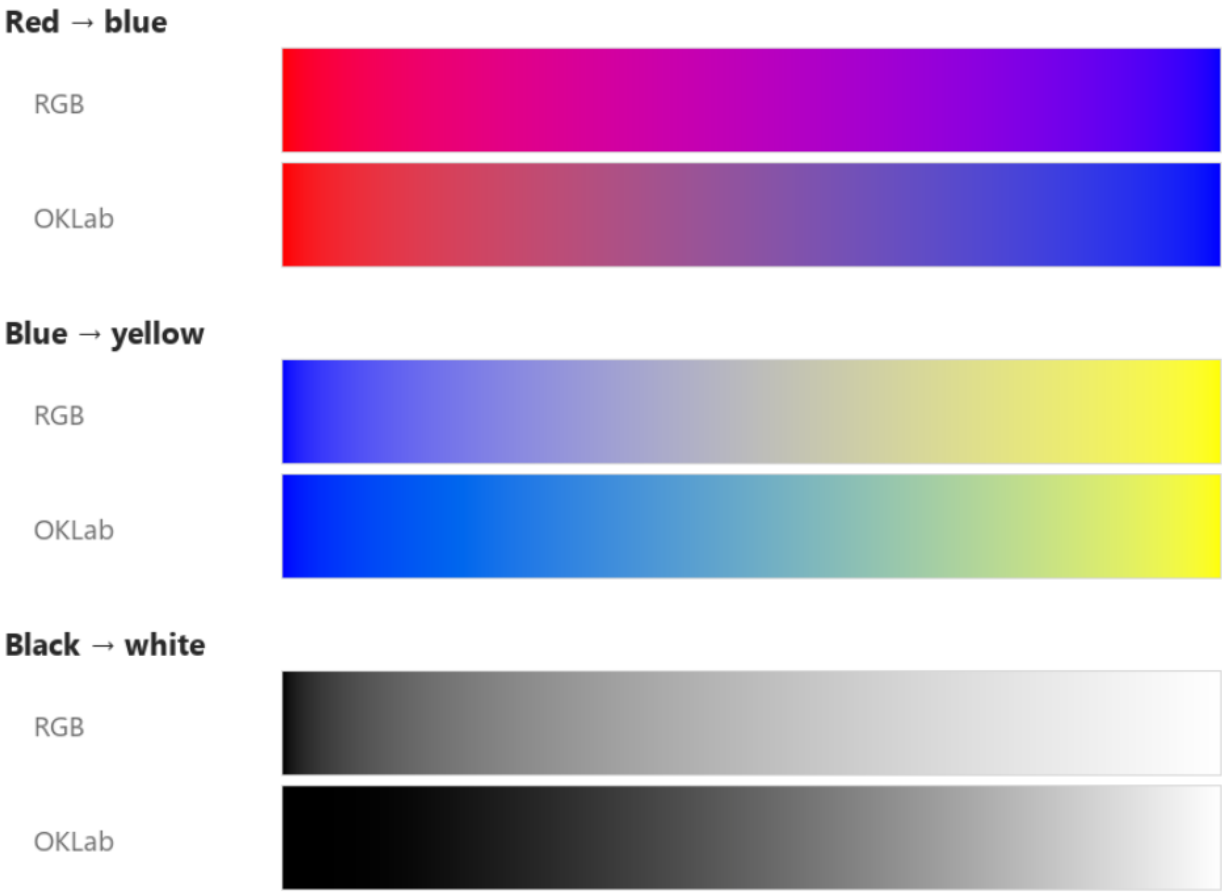


Before the first stop and after the last one, the end colors simply continue, so a ramp with stops only between 40 % and 60 % gives solid color at both ends. Two stops on the same position make a hard edge, which is how pie charts, progress rings and stripes are built.

RGB or OKLab blending

Use **OKLab** for blends between two different hues; use **RGB** for classic, technical looks and when you need to match an older design.

Copyright © 2026 Vizrt



RGB mixes the color channels directly, which can pass through greyish or overly bright middle tones (blue to yellow goes through grey). OKLab mixes in a perceptual color space, so hue changes evenly and the brightness steps look even to the eye.

Update mode

Keep **Auto** unless you need several changes to appear at the same moment.

- **Auto**: the gradient is redrawn as soon as any setting or stop changes. Best for design work and animation.
- **Manual**: changes are stored but not shown until an update is triggered (`UPDATE` command). Use it when a control system changes type, colors and position one after another and the viewer must never see the half-changed state; send all changes, then one update.

A Gradient in Manual mode is still drawn once when it first appears, so it never shows up black.

Use cases

Eight recipes cover most broadcast needs; copy the settings, then swap in your brand colors.

Copyright © 2026 Vizrt



Panel background
Linear, Angle 0, Origin 0 / 0.5



Vignette overlay
Radial, Radius 0.75, clear → black

Glow / spotlight
Radial, Radius 0.5, color → clear

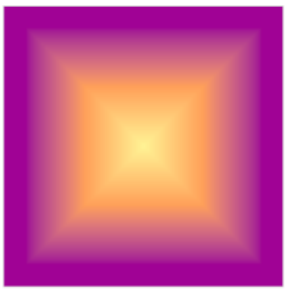


Metallic bar
Reflected, Angle 90, stops 0-50 %



Progress / pie (65 %)
Angular, Angle 90, hard stop at 65 %

Radar sweep
Angular, keyframe Angle 0 → 360



Badge / sunburst
Diamond, Angle 45, Radius 0.6



Text readability fade
Linear, Angle 90, Origin 0.5 / 0

Use case	Type and settings	Color ramp	Where to use it
Panel background	Linear, Angle 0, Origin 0 / 0.5, OKLab	0 %: dark brand color → 100 %: lighter brand color	Lower thirds, tickers, full-screen info panels
Vignette overlay	Radial, Origin center, Radius 0.75	45 %: black, alpha 0 → 100 %: black, alpha 85 %	Full-screen rectangle above the scene to darken the edges and focus the center
Glow / spotlight	Radial, Origin where the glow sits, Radius 0.5, OKLab	0 %: warm light color → 100 %: same hue, alpha 0	Behind player cut-outs, logos, product shots
Metallic bar	Reflected, Angle 90, Origin center	0 %: light grey, 12 %: mid grey, 50 %: near black	Score bugs, name straps, buttons
Progress / pie	Angular, Angle 90 (starts at the top), Origin center	Fill color 0 % and X %, then grey X % and 100 % (hard edge at X)	Polls, timers, progress rings (on a ring-shaped geometry); move the two X stops to change the value

Copyright © 2026 Vizrt

Use case	Type and settings	Color ramp	Where to use it
Radar sweep	Angular, keyframe Angle 0 → 360	0 %: transparent → 85 %: color, alpha 35 % → 100 %: bright color	Rotating light sweeps, scanners, "live" indicators
Badge / sunburst	Diamond, Angle 45 (square look), Radius 0.6, OKLab	0 %: yellow → 50 %: orange → 100 %: purple	Badges, retro sports graphics, tiles
Text readability fade	Linear, Angle 90, Origin 0.5 / 0	60 %: transparent → 100 %: near black, alpha 90 %	Over video or photos so white titles stay readable at the top (use Angle 270 and Origin 0.5 / 1 for the bottom)

Angle and Radius can be keyframed, so a breathing glow is a Radial gradient with Radius animated between 0.4 and 0.6, and a rotating sweep is an Angular gradient with Angle animated over 360°.

Tips and limits

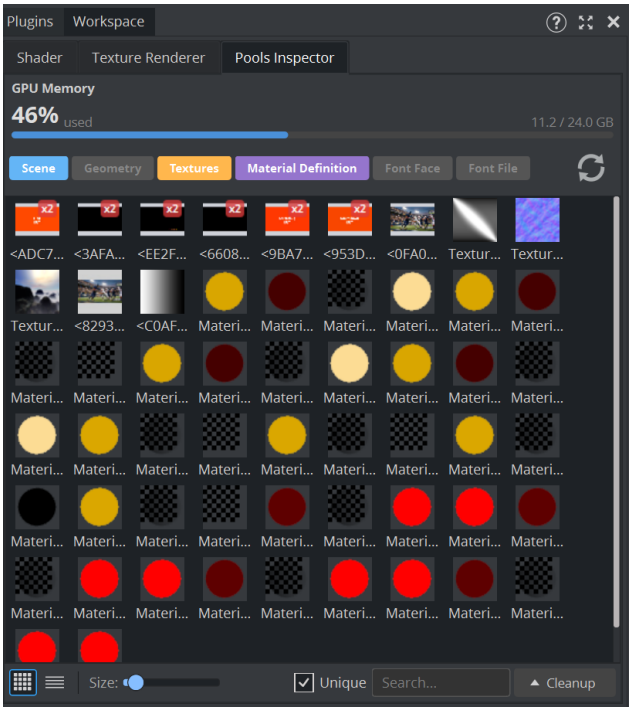
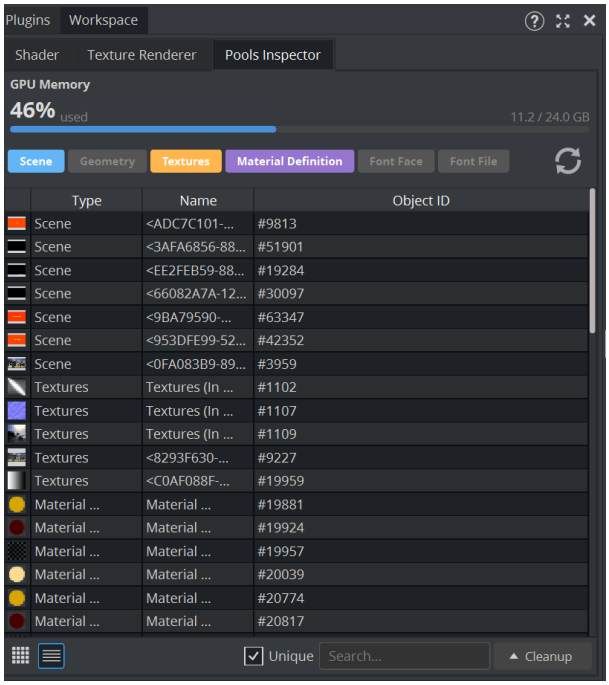
- **Resolution**: match it roughly to the size the gradient covers on screen. 512 × 512 px is plenty for soft blends; use 1024–2048 px for hard edges (pies, stripes) that fill a large area. Very large textures cost GPU memory: 4096 × 4096 px uses about 128 MB, 16384 × 16384 px about 2 GB and may fail to create, leaving the gradient black.
- **Aspect**: the texture stretches to the object's mapping. A square texture on a 16:9 rectangle turns circles into ellipses; set the Resolution to the object's aspect (for example 1920 × 1080) to keep circles round.
- **Detail limit**: the ramp is sampled in 256 steps across 0–100 %. Stops closer than about 0.4 % apart, or more than about 256 stops, cannot be told apart.
- **Transparency**: alpha in the stops is kept, so a gradient can fade objects out; the material must be set up to use the texture's alpha.
- **Lighting**: on a lit material the gradient is shaded by scene lights; switch lighting off for exact brand colors.
- **Values to avoid**: keep colors between 0 and 1 and positions between 0 % and 100 %. Extreme or invalid numbers sent from scripts (for example 1e39 or nan) can give black or broken output.
- **Manual update**: in Manual mode nothing changes on screen until an update is sent, which can look like a frozen gradient during design work; switch back to Auto while designing.

Copyright © 2026 Vizrt

12.5.4 Pools Inspector

The Pools Inspector, introduced with Artist 5.6; is an overview of which resources are currently allocated. This includes assets like Textures, Fonts, Geometries, scenes etc.

> The content might not be in real-time, therefore it is recommended to press the refresh button on the top to get an updated view.

Icon View	Table View
	

On top of the Pools inspector a progress bar shows the current usage of the GPU memory. This is the available memory on your current Graphics board.

The type of asset can be chosen by clicking on the colored labels, either click on single items or if you ALT+click only this selected type will be shown.
Choose between an **icon based overview** (size can be changed) or a more detailed view as **list**.
If a resource is allocated more than once (e.g. in multiple scenes) it will show up in the pool more than once, but is allocated only once. This will be indicated with a small red badge on top (like the first row in the sample above). Use the **Unique button** to merge them into a collection.

If resources are listed but not needed anymore, you can use the **Cleanup** function to clear certain pools.

Right clicking on an entry allows you to located the resource in the scene tree or (if available) in the asset view.

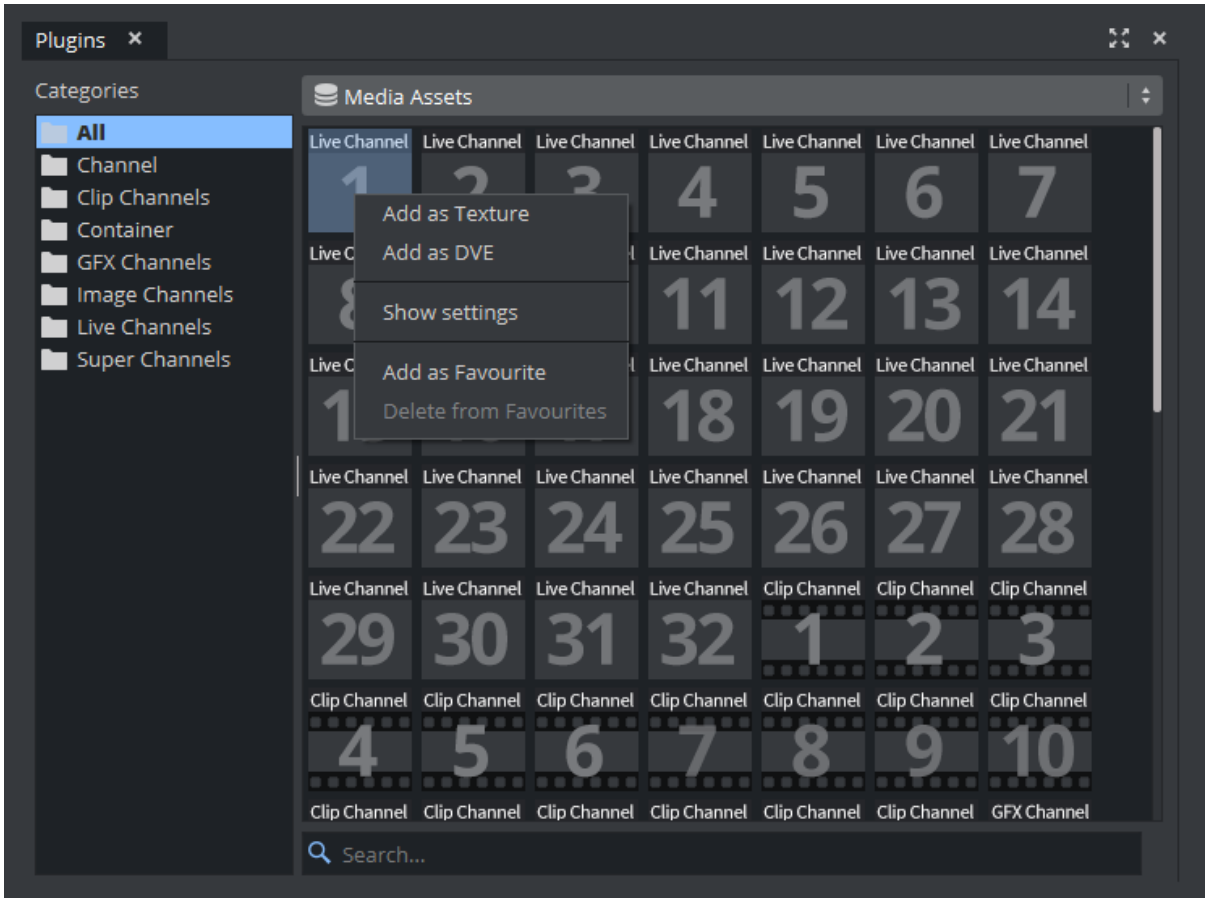
Copyright © 2026 Vizrt

13 Media Assets

Use Media Assets to add input sources to scenes as video clips, live video, video stream, graphics channels, image channels or Super Channels.

This section describes the workflow and each type of Media Asset, and details Media Asset requirements and considerations for video playback, which includes video clips, live video feeds and live feeds through video streaming.

The required and supported video codecs are also detailed.



The Media Assets are located in the Built Ins panel (see Main Menu), and are organized in folders corresponding to the Media Asset Channel Types. You can view all available media assets by selecting the **ALL** folder.

This section contains the following topics:

- Video Clips
- Media Asset Manager
- Media Asset Panel
- Media Asset Channel Types
- Playback of Media Assets
- Keying Mode

Copyright © 2026 Vizrt

- Viz Artist Supported Codecs

Copyright © 2026 Vizrt

13.1 Video Clips

This section details where video clips are stored, the transfer and download from Viz One and video clip playback.

This section contains the following topics:

- Video Clip Storage
  - Hard Drive Video Clips
  - Viz One Video Clips
  - Video Items
  - Dual Channel and Trio One Box Configuration
  - Variables Used in File-name Expansion
- Matrox Clip Player information
  - Playback of Clips with VBI
    - To Playback Clips with VBI in a Separate File
- VML Clip Player information
- Video Clip Playback
  - Video Clip Playback General Guidelines and Considerations
  - To Play a Video Clip as Clipchannel (Texture)
  - To Play a Video Clip as Texture ClipPlayer
  - To Play a Video Clip as Clipchannel (DVE, Matrox only)
  - Video Clip Animation in the Stage
  - To Animate a Video Clip
  - To Play a Single Clip in the Stage with Stage Loop
  - To Play Multiple Clips in One Channel
  - To Play Multiple Clips in One Channel with Stage Loop
  - To Play Single and/or Multiple Clips in Multiple Channels
  - Video Clip Queue
  - Video Clip Queue Commands
  - Example Clip Queue Commands

13.1.1 Video Clip Storage

This section details where video clip can be stored and accessed for use in Viz Artist.

Hard Drive Video Clips

Video clips should be placed on a fast SSD drive. Machines assembled by Vizrt are usually equipped with a *D:* drive.

IMPORTANT! It is not recommended to store video clips on the *C:* drive, as this is where the operating system is installed.

A second or third video clip directory can be added in **Local Settings** of Viz Configuration (see the Viz Engine Administrator Guide), which can be used for:

- A Dual Channel and Trio One Box Configuration, or

Copyright © 2026 Vizrt

- With any Viz Artist/Viz Engine configuration for the selection of a secondary video clip directory if a directory fails. If the file path for a video clip is set to *<clip_root>*, and the first directory falls over the same video file is searched for and used, from the next directory in the Clip directory list. If, when multiple Clip directories are selected, the **Clip Name** box is used to search for a video, the search defaults to the last used directory.

Viz One Video Clips

Video clips can also be transferred and downloaded from a Viz One. To do this Viz One must be integrated with a Viz Engine. See the **Viz One** page in the **Configuring Viz** section of the Viz Engine Administrator Guide and the **Viz One Deployment Guide**.

Video Items

Video clips are also stored in the Graphic Hub (see Working with Items). These are only used in combination with the After Effects import feature.

Dual Channel and Trio One Box Configuration

When more than one directory path is selected, for example, *D:\AEClips*; *D:\DecoClips*, etc., in the **Clip Name** box change the file path to *<clip_root>*. The file name is then searched for, in the selected directories (selected in the Viz Configuration), one after the other, for stored video clips with the same name.

Example: Search for a video clip named *next_tuesday.avi* in *D:\AEClips\AFL\Promos*. In the **Clip Name** box change *D:\AEClips\AFL\Promos* to *<clip_root>.* The filename shown in Viz Artist should be *<clip_root>\next_tuesday.avi*.

Variables Used in File-name Expansion

Variable	Definition
`<base_scene_name>`	Expands to the scene name.
`<absolute_scene_name>`	Expands to the complete path of the loaded scene.
`<absolute_scene2_name>`	Expands to the complete path of the second loaded scene.
`<clip_root>`	Expands to the clip_root.
`<hostname>`	Expands to the hostname.
`<if0>`	Expands to the IPV4 network address of the first network interface.

Copyright © 2026 Vizrt

Variable	Definition
`<if1>`	Expands to the IPV4 network address of the second network interface.
`<if2>`	Expands to the IPV4 network address of the third network interface.

13.1.2 Matrox Clip Player information

Playback of Clips with VBI

VBI (Vertical blanking interval) is the interval of time between the last line of a given field/frame and the beginning of the next field/frame, during which the incoming data stream is not shown on a monitor.

Video clips which contain VBI information, can be played back in two ways:

- One: Use a video clip which contains the embedded VBI information.
- Two: Use a separate video clip which contains the VBI data. This separate video clip must be named *[clipname]_vbi.avi* and conform to the size limitations detailed in the procedure below.

To Playback Clips with VBI in a Separate File

> **Note:** The links in this procedure all refer to the Viz Engine Administrator Guide.

1. Make sure that VBI is set to On for both the Clip Channel (Video Input: Clip Input) and the Video Output (Matrox).
2. For each clip channel, make sure that the Video Input and Output Format resolutions are the same.
   The VBI clip contains only VBI information. The size of the clip must match the maximal total lines for VBI in this resolution:
   - **NTSC:** 32
   - **PAL:** 34
   - **720p:** 19
   - **1080i:** 30
   - **1080p:** 35

Using VBI embedded in a full size clip is not supported as separate clip. This is only supported when played as the main clip.

13.1.3 VML Clip Player information

The VML player is a replacement of the Directshow Clip Player (Softclip). It is meant for all systems that require clip playback without the need for a Matrox Videoboard. This can be the preview of videoclips on a Trio Installation or the playback of decoration clips (formerly handled by Softclip plug-in). VML player does not require the additional installation of various codecs like Softclip or Directshow Player.

Copyright © 2026 Vizrt

The VML player is automatically being used if no Matrox board is present. Therefore the VML player is active on all systems without a Matrox board. To force a Clip player to use VML (or Matrox), you can specify the player in the Viz Engine configuration file:

```
Section CHANNELS_CONFIG
ClipInX.Player = 0 (Matrox → VML)
ClipInX.Player = 1 (VML only)
ClipInX.Player = 2 (Matrox only)
```

For more information, see **Software Configuration > VML player** in the Viz Engine Administrator Guide.

The VML Player can be used either as a Clipchannel or as a container-based player.

13.1.4 Video Clip Playback

Video clip playback should be used sparingly. The size and formats of some video clips can affect performance of the Viz Engine Renderer. Beside the Matrox Clip player and the new introduced VML player, plug-ins like MoViz, Softclip, VLC can be used. However, we recommend to use either Matrox or VML as all other plug-in related clip players depend on system extensions like Directshow or VfW which are already deprecated.

Video Clip Playback General Guidelines and Considerations

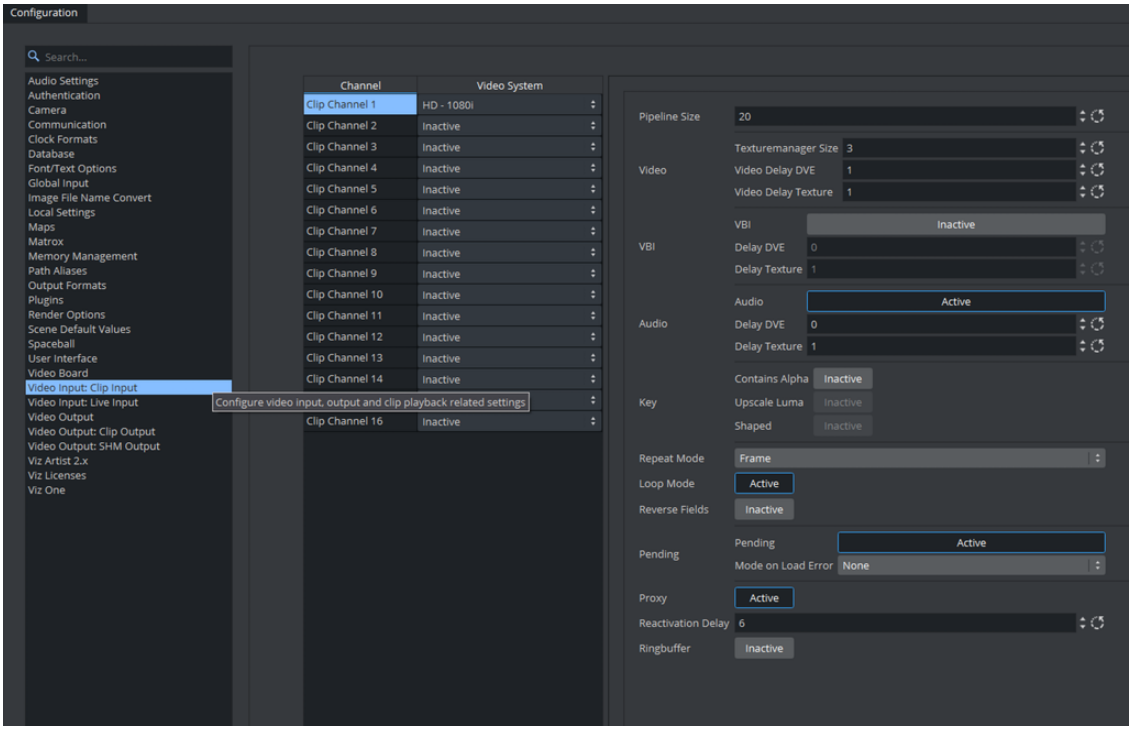
For video clip playback methods, consider these general guidelines:

Use-case	Playback Method
If no 3D effects or animation control, and only limited 2D effects, are needed on the clips.	To Play a Video Clip as DVE.
For full screen (for example, background) clips.	To Play a Video Clip as DVE.
If video must be mapped onto a texture in the scene or requires minor adjustments.	Video Clip Animation in the Stage.
For non-standard image format/resolution/codec.	Use VML player.

IMPORTANT! Always test the performance of the different solutions and work with the solution that gives the best quality versus performance ratio. A performance test should always take place on a machine with the same specifications as those used for production playback.

- The selected video clip must have the same frame rate as the clip channel. You can mix different **Output Formats** and **Video Input Formats** with different resolutions, but you must make sure to use either progressive or interlaced for Input and Output. You can play, for example, a 720p clip with an Output format of 1080p, but you cannot mix and 720p with a 1080i format:
  - **Video Input: Clip Channel 1:** HD 1080i
  - **Output Format:** 1080i

Copyright © 2026 Vizrt



Each available Clip Channel can be configured for independent resolutions, providing the designer quick access to a range of options. These settings are configured in the Video Input section in Viz Config. Please refer to the **Video Input** page in the **Configuring Viz** section of the Viz Engine Administrator Guide.

- Make sure that the correct container and codec is selected (see Supported Codecs).
- Video clip playback can be applied through either DVE or Texture.
- To be able to use a Live input channel, a Clip channel or a Stream channel, they first have to be set to active in **Video Input** (see the **Configuring Viz** section of the Viz Engine Administrator Guide).
- If alpha is required for a video clip, click **Config** and configure the alpha in **Video Input** (see the **Configuring Viz** section of the Viz Engine Administrator Guide). Enable **w/Alpha** when alpha is required in video clips. Either two clips, or one clip that contains both the fill and key, are required. If there is only one clip, that contains the fill (without a key), the key is created dynamically and is fully opaque. For the naming convention of two clips, the alpha/key clip must have the same name with an attached `_key` before the file extension, this means if the main/fill clip is named `example.mxf`, the alpha/key clip must be named as follows: `example_key.mxf`.
- The Matrox Clip Player supports TDIR (Time Delay Instant Replay), which enables Viz Engine to play back clips while they are still being written by Viz One. When a clip is played, with TDIR support enabled, looping is not enabled for this clip. Viz Engine determines automatically if it needs to set TDIR support **on**, or if the clip is already fully loaded.
- BG Graphics are always the lowest in the stack, even below zero, and FG graphics are assigned a default of `100`. Please note that priority can not be assigned to BG Graphics.

Tip: The Media Asset Manager provides a useful tool-tip when hovering the mouse pointer over **Clip Channel** media assets, displaying the resolution configured for the Clip Channel in use.

Copyright © 2026 Vizrt

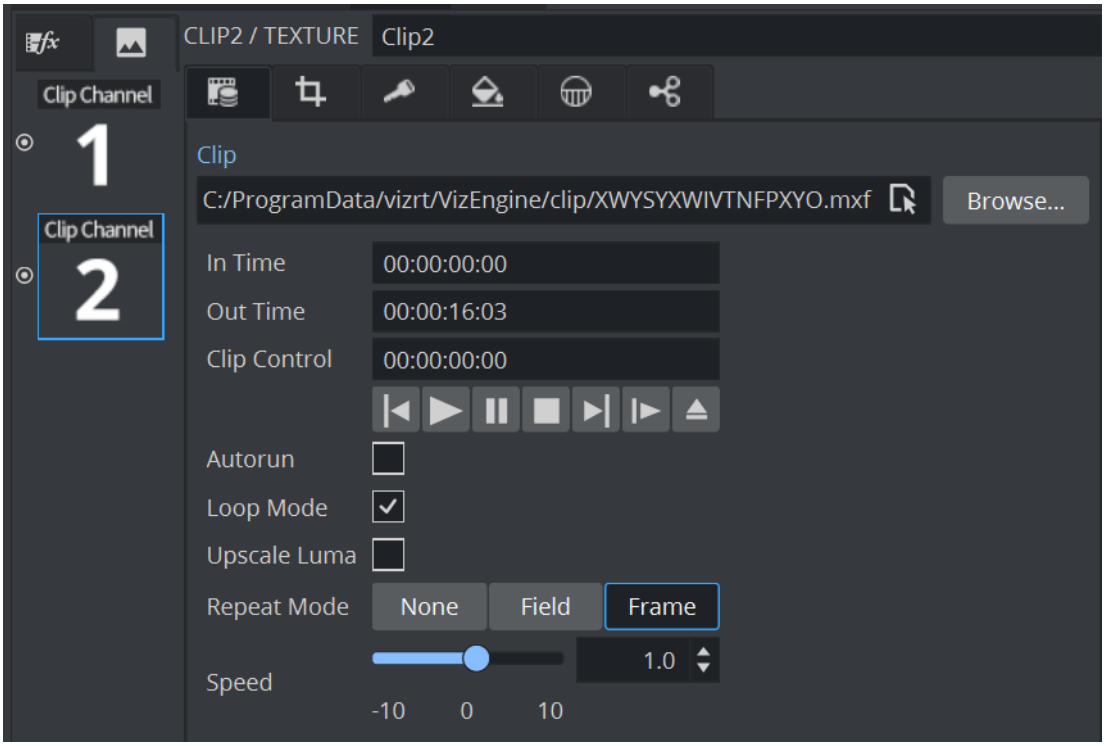
To Play a Video Clip as Clipchannel (Texture)

1. Click on **Plugins** / **Media Assets** / **Clip Channels**.
2. Drag one of the available Clip channels to the scene tree or **Right-click** on a Clip Channel, and select **Add Clip<1> as Texture**.
    - The **Texture** button in the Media Asset Manage) is automatically selected.
    - The selected Clip Channel is added to the Scene Tree and a new Container is created (click **Tree** to view).
    - 

Tip: The **Clip<1>** Channel can be made active or inactive. Toggle the active/inactive button.

4. Click on **Clip<1>** Channel, in the Scene Tree panel (Media Asset Manager) Texture view, to open its properties panel.

Copyright © 2026 Vizrt



5. Select the required **Keying Mode** (see Keying Mode).
6. Add a video clip:
    - Drag a clip from **Windows Explorer** or from **Viz One** browser
    - Click **Browse...**, and select a video clip to import
7. Configure the video clip playback as required.
8. Click **Play** to test the video clip.

Note: To transfer and download videos from Viz One, Viz One must be integrated and configured with Viz Artist/Viz Engine.

To Play a Video Clip as Texture ClipPlayer

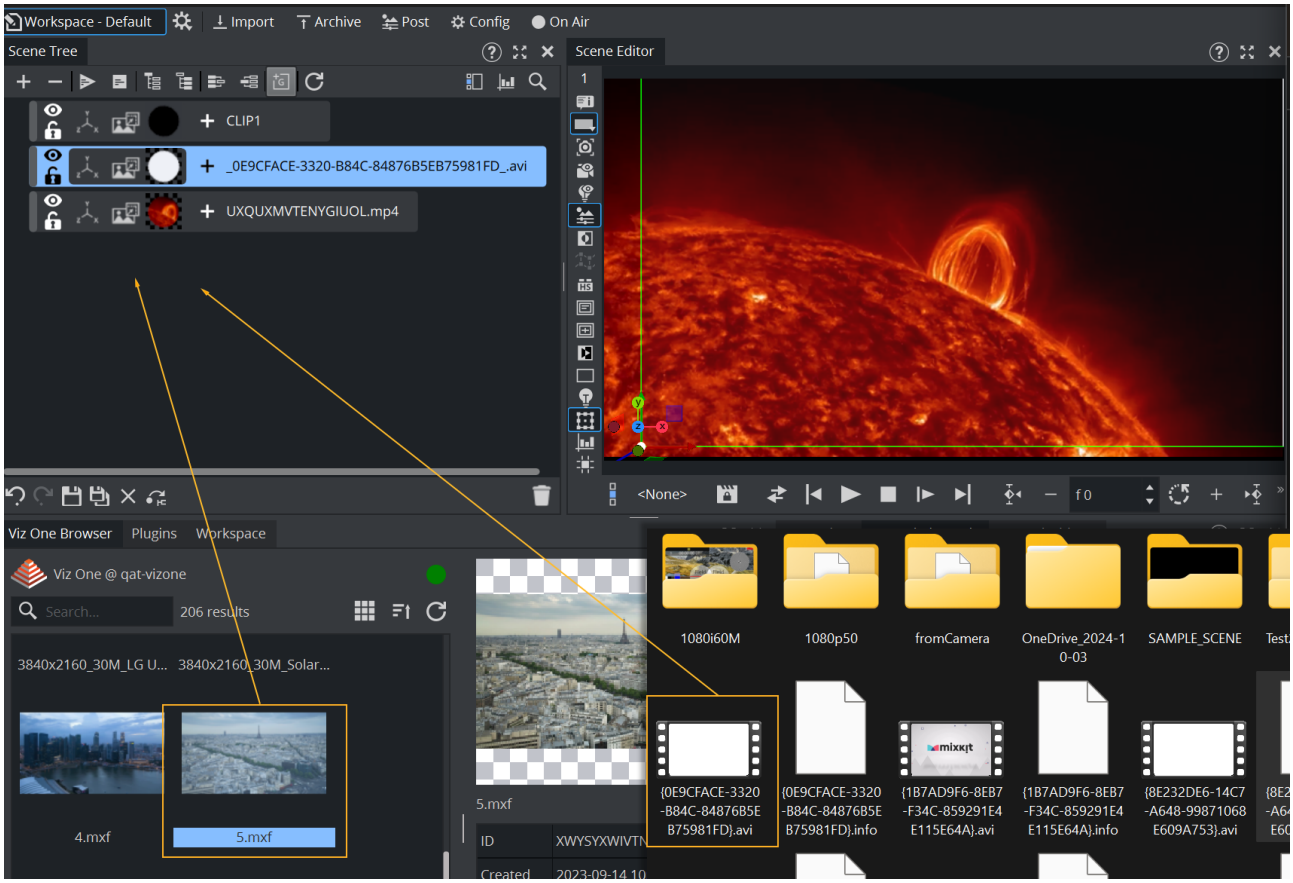
Container-based Clip Playback is mainly used for decoration clips, promos or if the clip dimensions are non broadcast formats. The texture clip player is based on the VML player and supports various formats as well as transparency.

Note: This feature is only available in the Viz Engine Render Pipeline.

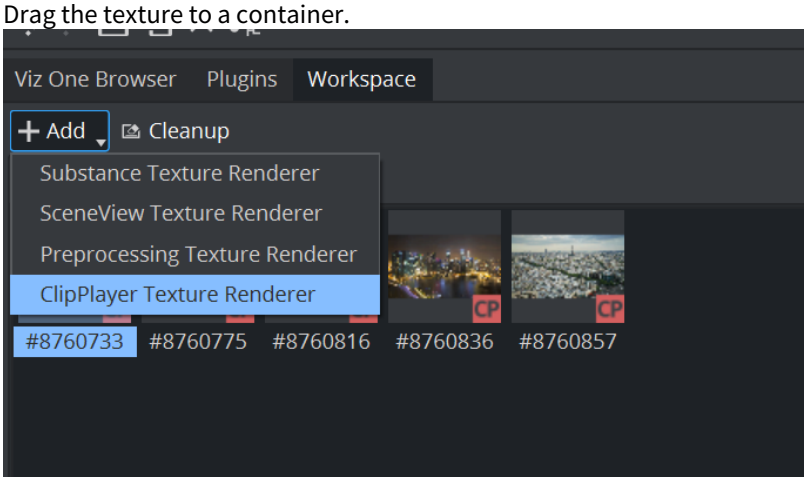
When using the new Viz One browser or if you drag any clip from the filesystem inside the scene editor or scene tree, the ClipPlayer Texture Renderer is used automatically.

Drag a clip from **Viz One browser** or **filesystem**

Copyright © 2026 Vizrt



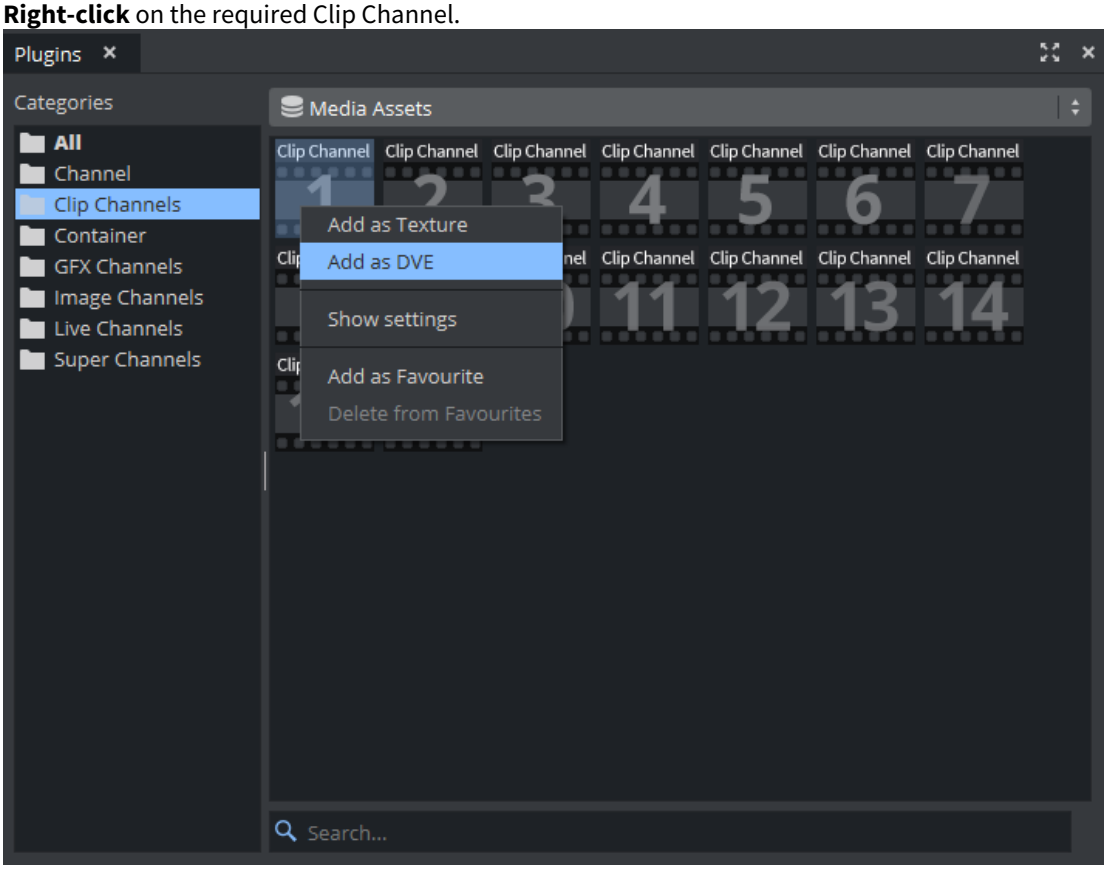
or create it manually:

1. Open the **Workspace** Dock.
2. Click on **Texture Renderer.**
3. **Add** > **ClipPlayer Texture Renderer.**
4. **Double Click** on the new generated Texture ClipPlayer.
5. Browse or assign a clip file.
6. Drag the texture to a container.
7. 

Copyright © 2026 Vizrt

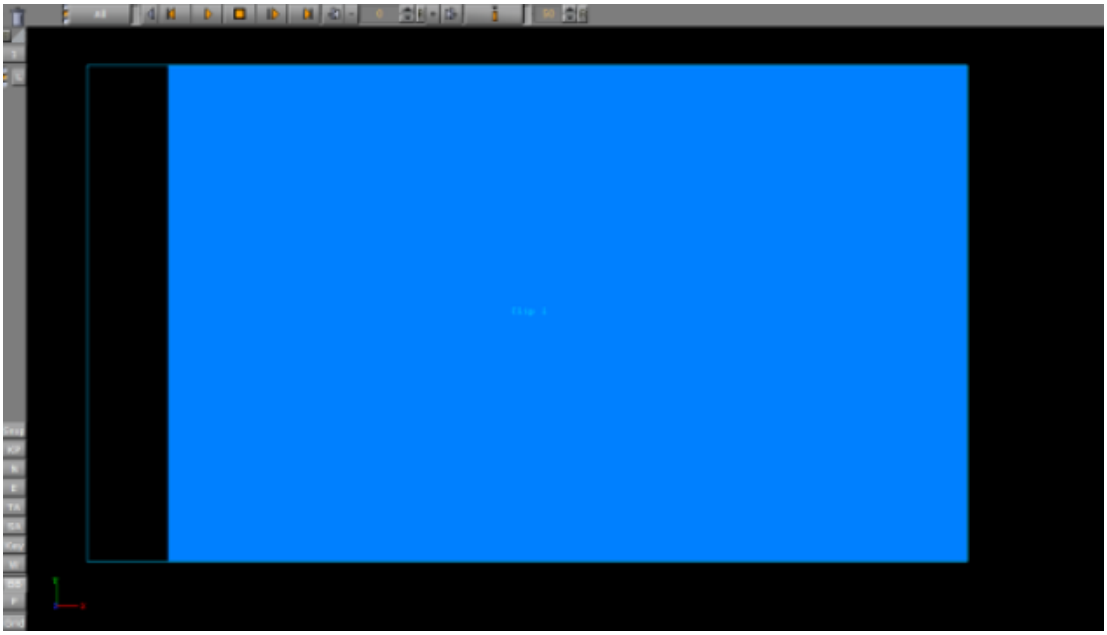
To Play a Video Clip as Clipchannel (DVE, Matrox only)

1. Click on **Built Ins**.
2. Click on **Media Assets** (MA).
3. **Right-click** on the required Clip Channel.



4. Select **Add Clip<1> as DVE**.
    - The **DVE** button in the Scene Tree panel (Media Asset Manager) is automatically selected. The Scene Editor shows a blue rectangle, which shows that the live input channel is correctly configured for live video.

Copyright © 2026 Vizrt



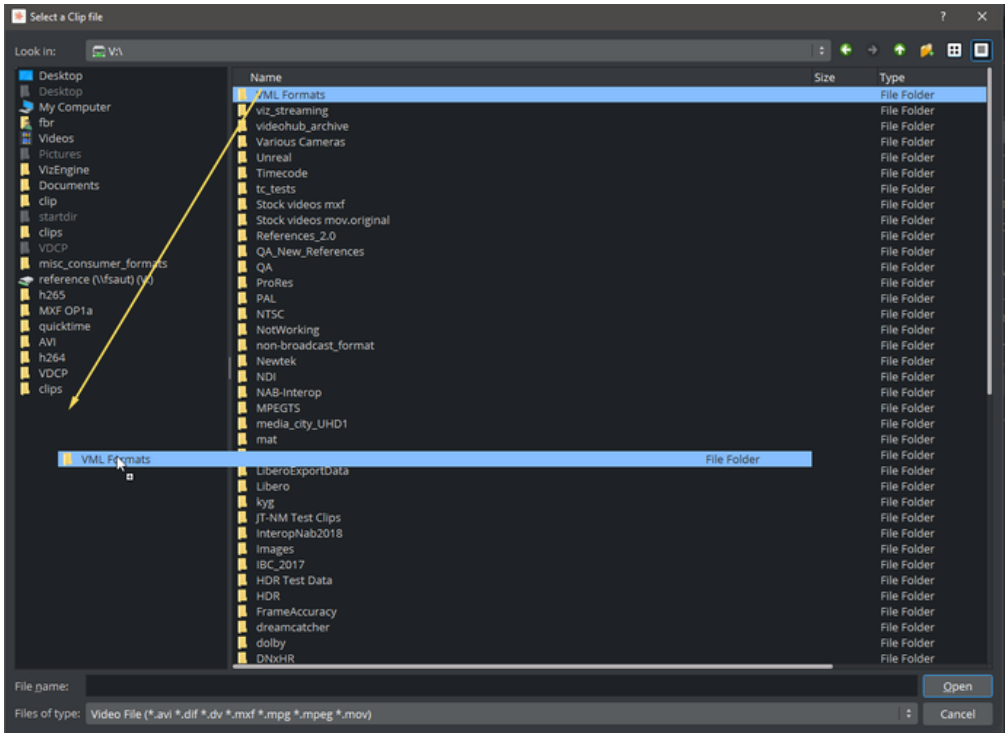
5. Click on the **Clip<1>** Channel to open its properties panel.
6. Adjust the parameters in the DVE control panel, as required.
7. Add a video clip:
    - In the **Clip Name** line, browser for a File, or
    - Drag a Video Clip item from the Asset View, or
    - Click **Viz One...**, and select a video clip to import.
8. Configure the video clip playback as required.
9. Click **Play** to test the video clip.

Note: To transfer and download videos from Viz One, Viz One must be integrated and configured with Viz Artist/Viz Engine.

Tip: The **Clip<1>** Channel can be made active or inactive. Toggle the active/ inactive button.

Browsing for Clips always shows the default Clip Root location. If you need to access other locations, you can create an alias inside the file browser dialog. To do so, drag an folder from the right side into the locations area on the left side:

Copyright © 2026 Vizrt



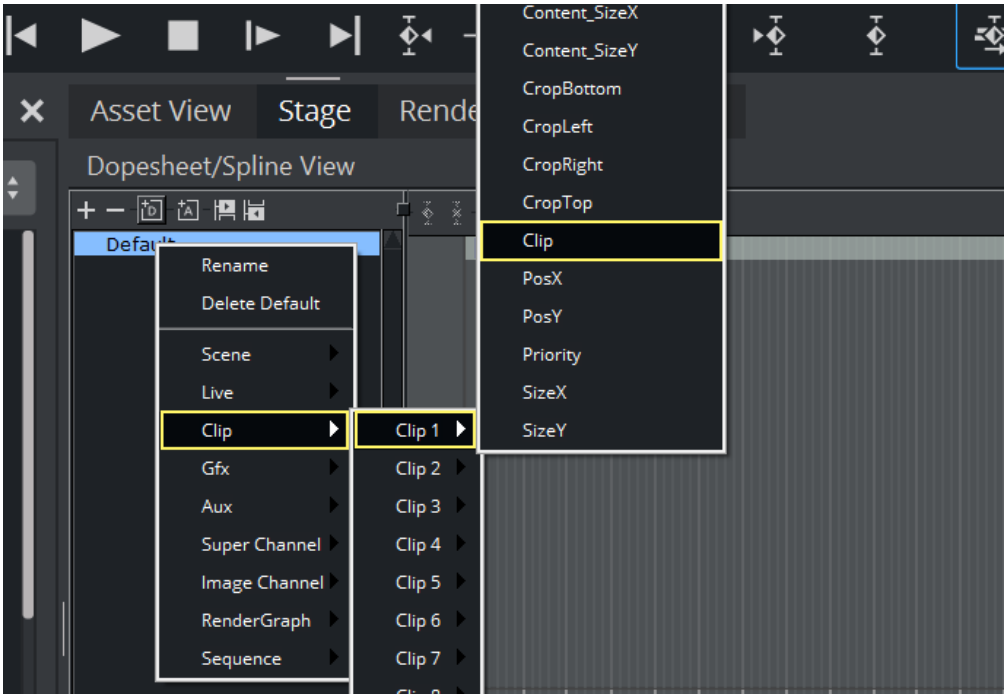
Video Clip Animation in the Stage

Video can also be played out as part of an animation. This is useful when video clips are required to interact with animation.

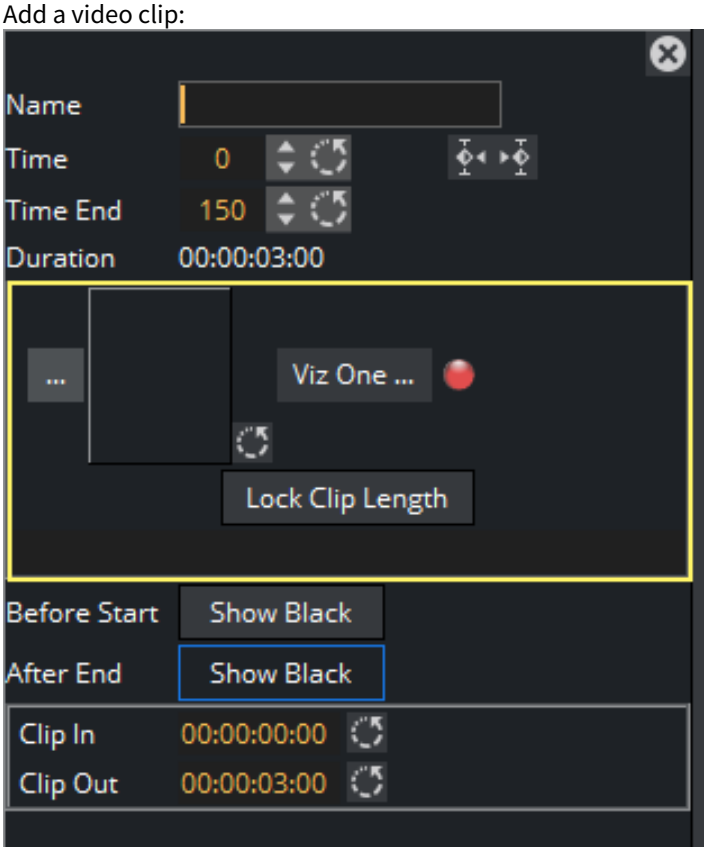
To Animate a Video Clip

1. Before a video clip can be animated, it must first be applied as a Texture or DVE.
2. Click **Stage**.
3. Click the New Director button.
4. Right-click on the Director.
5. Select **Clip > Clip1 > Clip**

Copyright © 2026 Vizrt



6. In the Clip1 channel, click in the dark blue area. The Clip Key Frame Editor opens on the far right.
7. Add a video clip:



To load a clip,

- Click ... to browse for a local clip, or drag a Video Clip item from the Asset View.

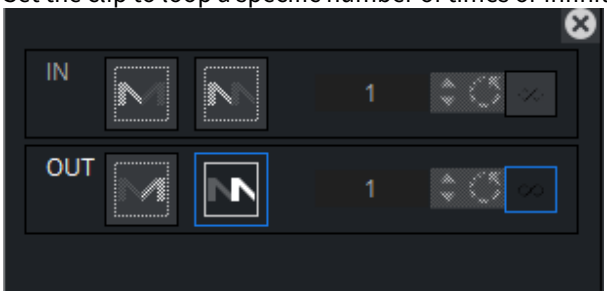
Copyright © 2026 Vizrt

- Click **Viz One...**, and select a video clip to import.
- Click **Lock Clip Length** to specify a certain time/length for your clip. This is useful when loading clips from an external control application via Viz One. In this case, the lock clip length acts as a placeholder, but does not change the clip length automatically after loading.

8. Configure the Clip Key Frame Editor as required.

To Play a Single Clip in the Stage with Stage Loop

1. Set a clip in the stage as described in To Animate a Video Clip.
2. Click **Clip**, in the Stage Tree.
3. Set the clip to loop a specific number of times or infinite.



To Play Multiple Clips in One Channel

1. Add a clip to the stage as described in To Play a Single Clip in the Stage with Stage Loop.
2. Set the time indicator to a position after the first clip Key Frame.
3. Click to add a Key Frame.
4. Select a clip for the new Key Frame (see To Animate a Video Clip). Show black rules for pauses between clips:

Black Rules	Frames Shown
Both clips NS	Frame 1 of Clip 2 is shown.
Clip 1 AE, Clip 2 NS	Frame 1 of Clip 2 is shown.
Clip 1 NS, Clip 2 BS	Last frame of Clip 1 is shown.
Clip 1 AE, Clip 2 BS	Black is shown.
Key:NS	Nothing is shown.
setBS	Shows black before start.
setAE	Shows black after end set.

If **Pending** is Active (which is recommended) subsequent clips are loaded into the pending clip player and are swapped at the correct time. This feature enables the clips to concatenate seamlessly. However, when pending

Copyright © 2026 Vizrt

support is set to Inactive, black frames are shown until the clip is ready to be played (see the **Configuring Viz** section of the Viz Engine Administrator Guide).

To Play Multiple Clips in One Channel with Stage Loop

1. Set multiple clips in the stage as described in To Play Multiple Clips in One Channel.
2. Set the clips to loop. Click the corresponding entry in the Stage Tree.
   All Key Frames including the pauses in between and before are taken into account during play back, for example, pause - clip 1- pause - clip 2 loops as such.

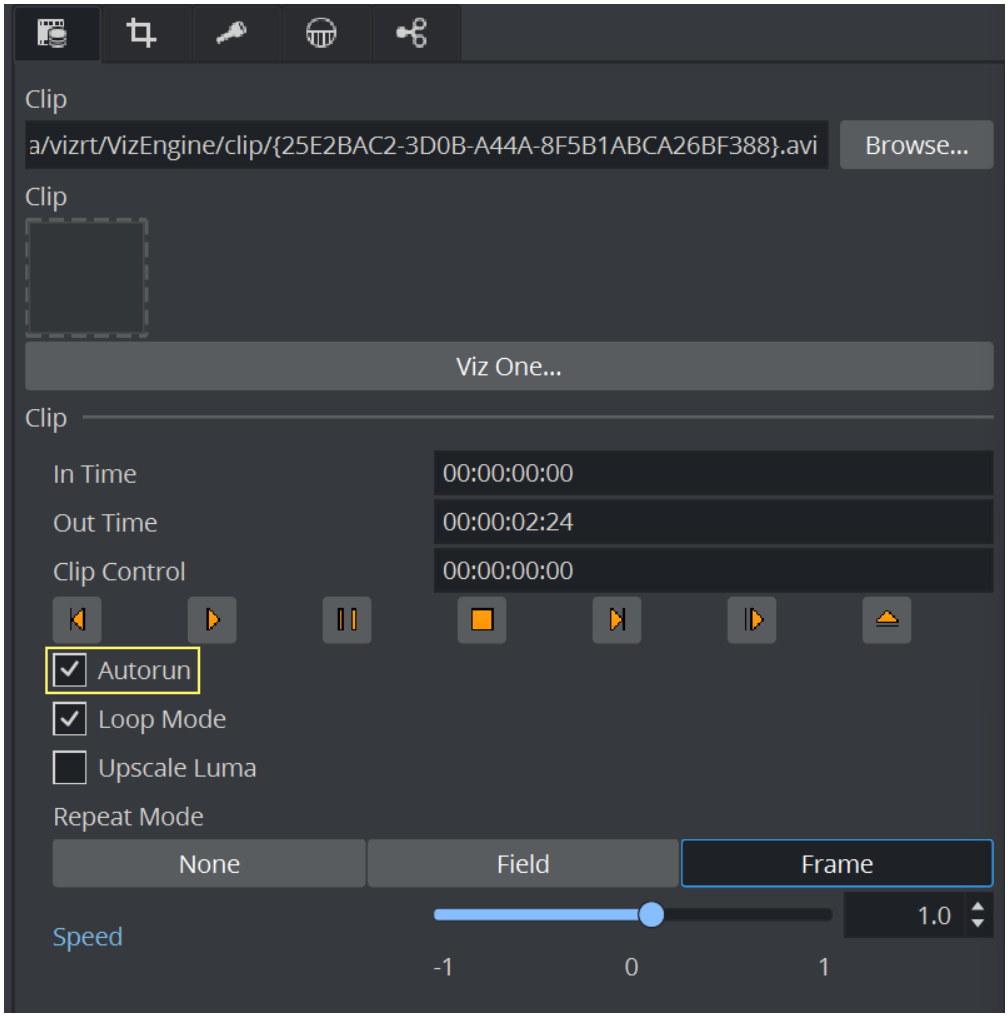
To Play Single and/or Multiple Clips in Multiple Channels

This is a combination of the use cases described above, with the clips used on several channels.

Video Clip Queue

Video clips can also be queued, providing automatic playback of a series of clips. If no clip is playing at the moment of clip queue creation, playback of the first clip in the queue starts immediately. All other clips are played in the sequence they were added to the list. By default, every clip needs to be played out manually, although it is not required to set their names again. This behavior can be overridden by clicking the **Autorun** button, setting it to `On`.

Copyright © 2026 Vizrt



The current state of the auto-play flag is stored with every queued clip, and can thus be used on demand.

Video Clip Queue Commands

Commands (`n` is the clip channel):

Command	Description
`[...]*CLIPIN*n*NAME CUE name in out`	Adds clip with `name` to the queue, using `in` and `out` as trim in and out points. The parameters `in` and `out` are optional and can be omitted.
`[...]*CLIPIN*n*QUEUE LIST`	Lists all the clips currently queued.

Copyright © 2026 Vizrt

Command	Description
`[...]*CLIPIN*n*QUEUE INSERT name in out position`	Inserts clip name using `in` and `out` at `position` in the queue.
`[...]*CLIPIN*n*QUEUE REPLACE name in out position`	Replaces the clip at `position` with the new one using `name`, `in` and `out`.
`[...]*CLIPIN*n*QUEUE DELETE position`	Deletes the clip at `position` in the queue.
`[...]*CLIPIN*n*QUEUE MOVE source target`	Moves the clip from position `source` to position `target` in the queue.
`[...]*CLIPIN*n*QUEUE FLUSH`	Removes any and all clips from the play-list.

Example Clip Queue Commands

```
RENDERER*VIDEO*CLIPIN*n*NAME CUE
```

```
MAIN_SCENE*VIDEO*CLIPIN*n*NAME CUE
```

See Also

- For Dual Channel and Trio OneBox setup, see Dual Channel and Trio One Box Configuration.
- Supported Codecs
- To Play a Video Clip as Texture
- To Play a Video Clip as DVE
- ClipPlayer Texture Renderer

Copyright © 2026 Vizrt

13.1.5 Video Clip Playback Considerations

There are two methods to playback Video Clips:

- Playback directly in a Clip Channel (see Video Clip Playback).
- Playback as a Key Frame in the Stage (see Video Clip Animation in the Stage).

In both methods, the Media Assets all share the same available Clip Channels (1 to 32).

A Clip Channel can only have one video clip applied to it. If Clip Channel1 is applied to a Container, and is also applied to a Director in the Stage, the same video clip is shown in the Container and Stage, and the whichever is set last has priority. When the same Clip Channel is used in both the Scene Settings and the Stage, the main problem is that, whatever is changed in one place is reflected in the other. For example, when the out time of the clip is changed in the Scene Settings (which means the content of the player itself are changed) these changes affect the clip playback in the Stage as well, because it is the same player. In most if not all cases you do not want this to happen. To have Clip Channels in both the Scene Tree and the Stage use two different Clip channels (Clip1 and Clip2), or do not use both methods in one Scene.

The following points are important if one wants to playback clips without any jittering or stuttering:

- Each clip channel requires several resources like buffers, codec initialization, file handles etc. Therefore please activate only the required amount of clip players going to be used.
- Clips do have a pending player in background, meaning a clip is loaded into the pending player and when ready, it becomes the active player. This is not required anymore when using Super Channels, please deactivate the pending clip player.
- To avoid jittering when going On Air, please use a warmup scene to utilize the clip channels. This warmup scene should have all required clip channels available as textures.
- The warmup scene should load a clip into its clip channels. This ensures all buffers are prepared and filled up. The codec of the clip should match the codec used later on (switching to a different codec should not affect the clip player).

Information: You can use the flag *Matrox.Clip.DelayedInitialization = 0*, to make sure all clip channels are initialized on engine startup. The loading of clips is still required by the warmup scene. This increases your startup and shutdown time!

Layers

All Scenes, in all layers, also share the same Clip Channels.

If a Scene is loaded in the Main Layer with a video clip on Clip Channel 3, and a Scene is also loaded to the Back Layer with a different video clip, but also on Clip3, both Scenes/Layers plays the last loaded video clip. The one from the Back Layer in this example.

Clip Players

Each Clip Channel holds at least one Clip Player. If pending Clip Player support is enabled for the Clip Channel, an additional Clip Player is used for play back. Every Clip Player uses up additional system memory. But seamless transition from one video clip to another Video clip is only possible with pending Clip Player support enabled.

This section contains the following topics:

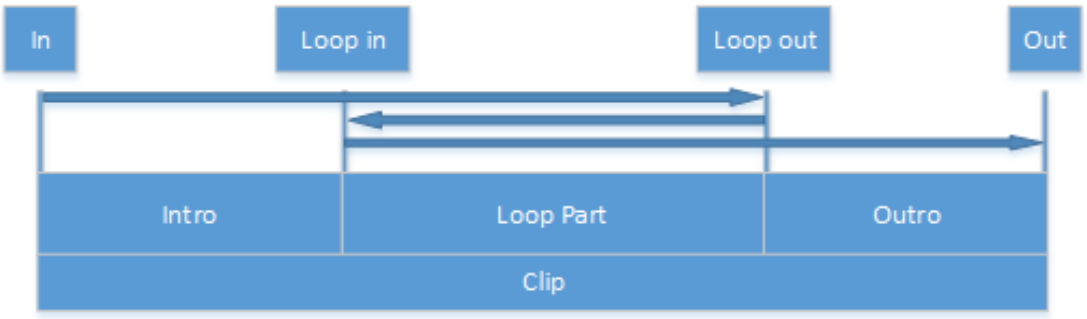
- Layers

Copyright © 2026 Vizrt

- Clip Players
    - 3 and 4-Point Loops in Clip Player
    - Example: 4-Point Loop Use
    - Parameters for 4-Point Loops in the Stage Editor
- General, Mixed and Special Modes
    - Play Clips with Resolutions Different from Channel
    - Frame Accurate Clip Playback
- Video Clip Playback Use Cases
    - Playing Overlay Clips (Not Full Screen)
    - Playing UHD Clips
- Clip Handling
    - Slow Motion Playback
    - Fast Forward and Fast Backward
    - Flushing Clip Channels
- Settings
    - Repeat Mode
    - Reverse Fields
    - Loop Mode
    - Auto Run
    - Auto Format
- Scripting and Plug-in Interface
    - Scripting
    - Plug-ins
- IO Matrix
    - Source PAL
    - Source 720p
    - Source 1080i

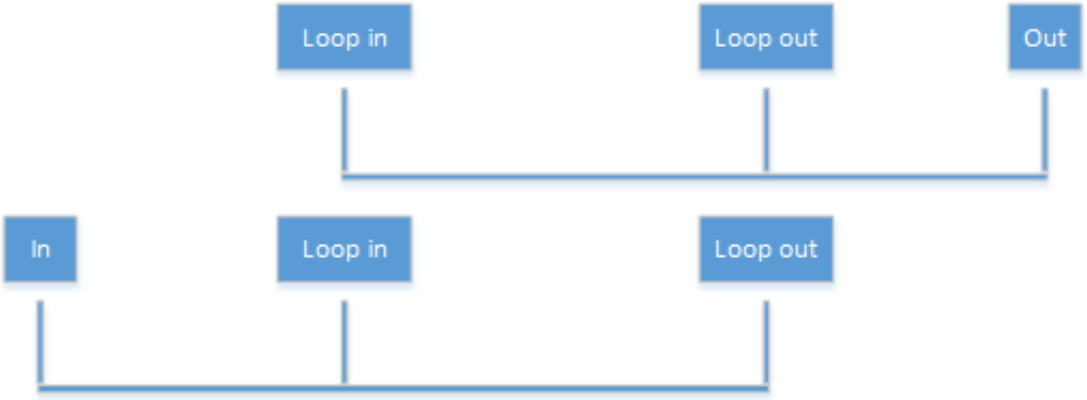
3 and 4-Point Loops in Clip Player

A 4-Point Loop describes a clip that has at least a loop part and an intro or an outro or both. To define a 4-Point Loop, the loop in and out time code and the number of times the loop part should be repeated, possibly infinite, are required.



Copyright © 2026 Vizrt

Note: A 3-Point Loop is a clip which has a loop part and only an intro or an outro (see below).



IMPORTANT! The two clip players need to be configured correctly for the 4-Point Loop to work correctly (the animation in the Stage must be set to the beginning). Click on the **Jump to beginning of animation** button to set the two clip players, to the beginning of the animation.



Example: 4-Point Loop Use

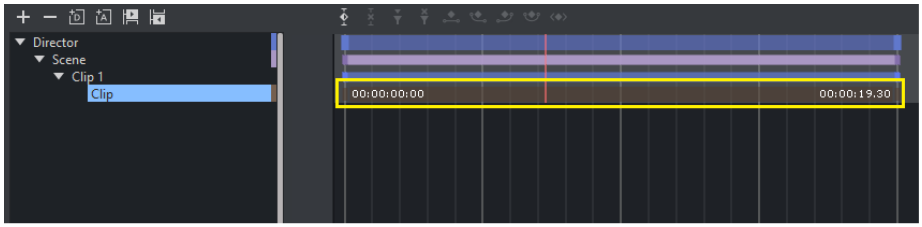
4-Point Loops can be useful in a wide variety of cases where you need to loop a video segment. Perhaps you do not know in advance exactly how many times you want to loop the video. For example, let us assume the video clip is a spinning globe animation. You know you have an intro, but how long the globe should spin, is dependent of the time another scene item is shown, say a video of the anchor speaking. When the anchor-video is done, you want the spinning globe to exit the scene.

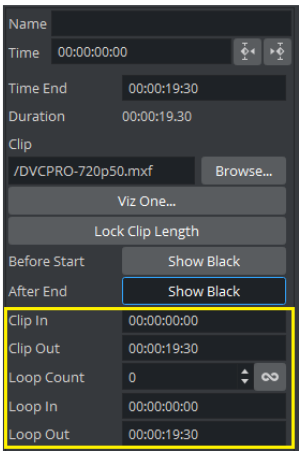
Parameters for 4-Point Loops in the Stage Editor

Additional fields have been added to the clip setting pane where loop in, loop out and loop count can be set as shown in the image below. No additional command is needed to start or stop the loop.

Click on the Clip channel to view the 4-Point Loop parameters.

Copyright © 2026 Vizrt





During scene-design the values of the 4-Point Loop are recalculated automatically when the Key Frames are changed by dragging them in the director tree. Any changes made in the entry fields are reflected in the time line pane to show the real-time length of the clip. When the start or the end frame of the clip is dragged, it snaps to the nearest possible time code to have an integer loop count.

General, Mixed and Special Modes

Play Clips with Resolutions Different from Channel

It is possible to play clips even if the resolution is different from the one set for the channel. However, there are some restrictions to that:

- Can only play clips from the same genlock family.
- The clip's resolution must be of a regular broadcast format. For detailed information on which combinations are possible check the appendix.

Frame Accurate Clip Playback

If the clip, in the stage, needs to be played frame accurately, these features need to be set to active:

- First, the pending clip playback support has to active. If not, the clips are not ready when needed (see the **Video Input: Clip Input** page in the **Configuring Viz** section of the Viz Engine Administrator Guide).
- Additionally the Frame Accurate Viz Commands (FAVC) need to be set to active in **Global Properties** (see the **Communication** page in the **Configuring Viz** section of the Viz Engine Administrator Guide).
- And the Command Execution Field Dominance should be set to **Odd Retrace Counter** (see **Communication > Global Properties** in the **Configuring Viz** section of the Viz Engine Administrator Guide). When using external control applications, it may be necessary to tweak the FAVC Bias.

For a scene containing a clip in the stage to play frame accurately it is necessary to use the `SCENE*CUE` command, which preloads the first clip Key Frame.

Video Clip Playback Use Cases

With the Clip Channel Media Asset it is possible to play images as DVE or Texture with the Matrox clip player or VML player. Supported image containers are *.jpg* and *.png*. The images have to be available on a physical disk for this

Copyright © 2026 Vizrt

feature to work. One advantage of an image played as a clip, is that the same features available for a video clip can be used on the image.

Playing Overlay Clips (Not Full Screen)

Playing a clip which is not full screen in size, like a spinning logo, an insert snipe or a special effect without audio. These are known as overlay clips. To play overlay clips it is recommended to use the VML clip player. VML supports clips with alpha and plays a lot of codecs.

SoftClip is able to play all installed VFW (Video For Windows/AVI) codecs on your computer. Please note that VfW is officially deprecated. There are dozens of codecs available, each with it's own advantages and drawbacks. There is no general rule which codec to use (see Supported Codecs), but it is recommended to use the different types of Matrox codecs (depending on whether the alpha channel is necessary or not).

Playing UHD Clips

Clips in Ultra HD resolution require a lot of CPU resources for decoding. A strong CPU is required. The performance depends on the used codec, bitrate, color depth etc. It is recommended to use a special decoder board from Matrox (Matrox M264) in combination with a Matrox topology board.

Clip Handling

Slow Motion Playback

A shuttle rate between `-1.0` and `1.0` results in the clip playing slower than normal, where negative values result in a backwards playback. The actual frame to play is determined by the internal frame counter which is increased on every grabbed frame. For a frame rate of `0.3` this would be:

InternalCounter	ActualFrame	InternalCounter	ActualFrame
Internal 0	Frame 0	Internal 0.3	Frame 0
Internal 0.6	Frame 0	Internal 0.9	Frame 0
Internal 1.2	Frame 1	Internal 1.5	Frame 1
Internal 1.8	Frame 1	Internal 2.1	Frame 2

Fast Forward and Fast Backward

A shuttle rate <-1.0 or >1.0 results in the clip playing faster than normal, where negative values result in a backwards playback. The calculation of the next Frame to be delivered is done the same way as described above using the set shuttle rate.

Copyright © 2026 Vizrt

Flushing Clip Channels

The `FLUSH` command can be used to clear the active clip player and reset all parameters to their default value. All valid clip frames still in the pipeline are cleared as well and black is sent. If the pending clip player also needs to be cleared, send `FLUSH` to the PENDING command channel.

Settings

Repeat Mode

Determines how clip playback should behave when paused:

- **None:** Black is played out.
- **Frame:** The last frame is shown, which shows both fields.
- **Field:** Only the last field is shown, copied to first and second field. This setting is only relevant if the clip's scan mode is interlaced.

Reverse Fields

This setting is used for smooth playback when playing interlaced clips in reverse. Enabling reverse fields swaps the playback of first and second field during reverse playback. This setting applies only to interlaced clips.

Loop Mode

If on, the clip loops infinitely between clip in and out times.

Auto Run

If on, the clips start immediately after being loaded and delivering the first frame.

Auto Format

This setting was introduced in Viz Engine version 5.1 to streamline the configuration of clip channels.

When enabled, the clip channel decides the internal buffer characteristics to match the loaded clip format. The configurations *ClipInX. ContainsAlpha*, *ClipInX. UseV210,* and *clip_bpcX* are ignored.

The Auto Format setting has some limitations in some workflows, such as in stage when there is b2b playback and the two clips involved have different formats (for example, first clip has alpha and the second does not). For these cases, it is advisable to disable Auto Format so the clip channel has the same behavior as prior Viz Engine version 5.1.0, or to use VML Clip Player as the default clip player (*ClipInX.Player* = 1).

Scripting and Plug-in Interface

Scripting

The scripting interface refers to the plug-in interface and has therefore the same limitations as the plug-ins have. To use a clip channel in scripting, first retrieve a channel from the system:

Copyright © 2026 Vizrt

```
oClipChannel as ClipChannel = system.GetClipChannel( iChannel )
```

This object can now be used to control the clip channel:

```
oClipChannel.SetClipName( "clip.avi" )
oClipChannel.LoopMode = true
oClipChannel.Play(0)
```

The scripting interface to clip channel allows for the same use cases as the plug-in interface does.

Plug-ins

The plug-in interface provides almost all functions available in the Command Interface. The only limitation here is, all functions refer to the current scene. The use cases are the same as described in Video Clip Playback Considerations#3 and 4-Point Loops in Clip Player. Stage use cases cannot be addressed with the plug-in interface.

IO Matrix

Source PAL

Output DVE	Video In				Output Texture	Video In		
	PAL	720p	1080i			PAL	720p	1080i
PAL	✓	-	-		PAL	✓	-	-
720p	-	-	-		720p	-	-	-
1080i	✓	-	-		1080i	✓	-	

Output DVE	Clip In				Output Texture	Clip In		
	PAL	720p	1080i			PAL	720p	1080i
PAL	✓	-	-		PAL	✓	-	-
720p	-	✓	-		720p	✓	✓	-
1080i	✓	✓	✓		1080i	✓	✓	✓

Copyright © 2026 Vizrt

Source 720p

Output DVE	Video In				Output Texture	Video In		
	PAL	720p	1080i			PAL	720p	1080i
PAL	-	-	-		PAL	-	✓	-
720p	-	✓	-		720p	-	✓	-
1080i	-	✓	-		1080i	-	✓	-

Output DVE	Clip In				Output Texture	Clip In		
	PAL	720p	1080i			PAL	720p	1080i
PAL	✓	-	-		PAL	✓	✓	-
720p	-	✓	-		720p	✓	✓	-
1080i	✓	✓	✓		1080i	✓	✓	✓

Source 1080i

Output DVE	Video In				Output Texture	Video In		
	PAL	720p	1080i			PAL	720p	1080i
PAL	-	-	-		PAL	-	-	✓
720p	-	-	-		720p	-	-	✓
1080i	-	-	✓		1080i	-	-	✓

Copyright © 2026 Vizrt

Output DVE	Clip In				Output Texture	Clip In		
	PAL	720p	1080i			PAL	720p	1080i
PAL	✓	-	-		PAL	✓	-	✓
720p	-	✓	-		720p	✓	✓	✓
1080i	✓	✓	✓		1080i	✓	✓	✓

See Also

- The Stage for Animation
- Key Frame Editors
- Channel Editor

Copyright © 2026 Vizrt

13.1.6 Transfer Clips From Viz One

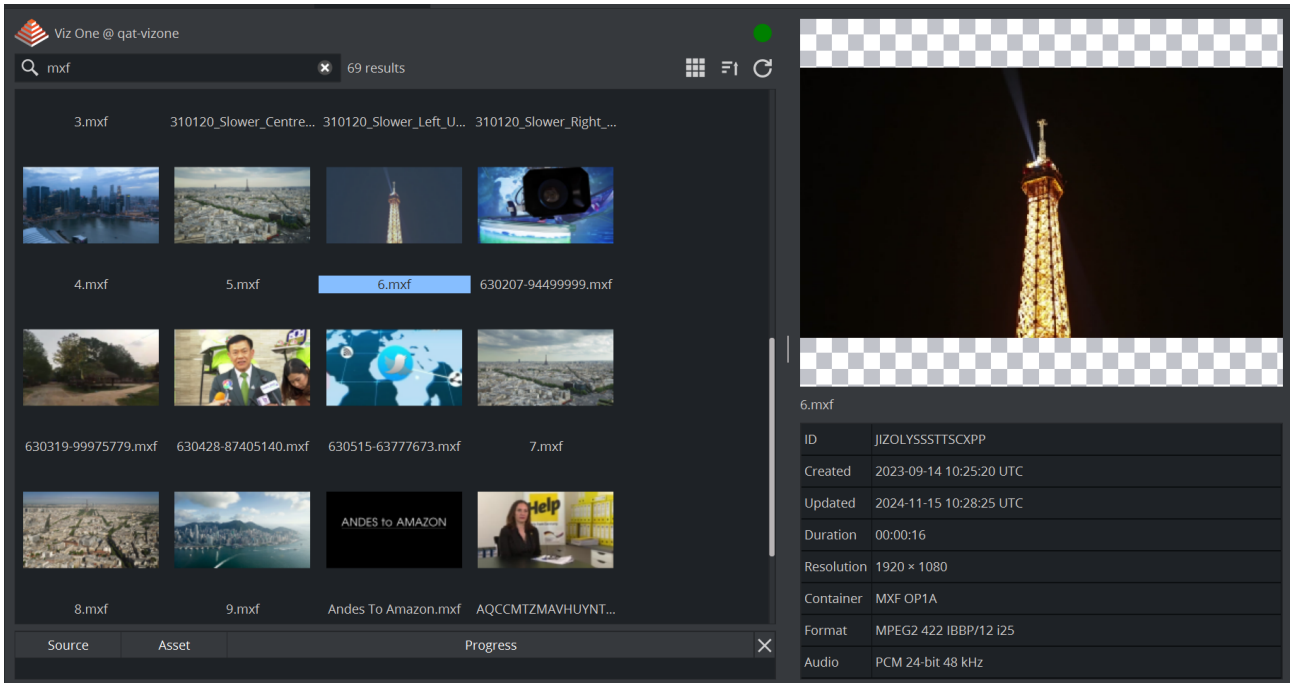
This section details how to transfer and use video clips from a configured Viz One.

To transfer and download a video clip or image from Viz One, Viz One must be integrated and configured with Viz Artist/Engine (see the **Viz One** and **Authentication** pages in the **Configuring Viz** section of the Viz Engine Administrator Guide and the **Viz One Deployment Guide**).

> **Information:** Viz One transfer requires **Mediaftp (File Transfer)** and **Fsmon (File System Monitor)** services are installed and running, and **Viz One (MessageQueue Server)** is correctly configured. Please refer to the Viz Engine Administrator Guide for detailed information on how to configure and install these services.

Using the Viz One Browser

The Viz One Browser is available as a dock inside Viz Artist. Once the configuration is set, the status icon will turn into green, otherwise it will indicate an error.



The Viz One browser allows to search for **Video Clips** and **Still images** on a Viz One Storage. Simply type you search term in the search field and the browser will display the result. Clicking on any item will show a larger preview and some more metadata.

> **Note:** Clip preview is currently not yet supported.

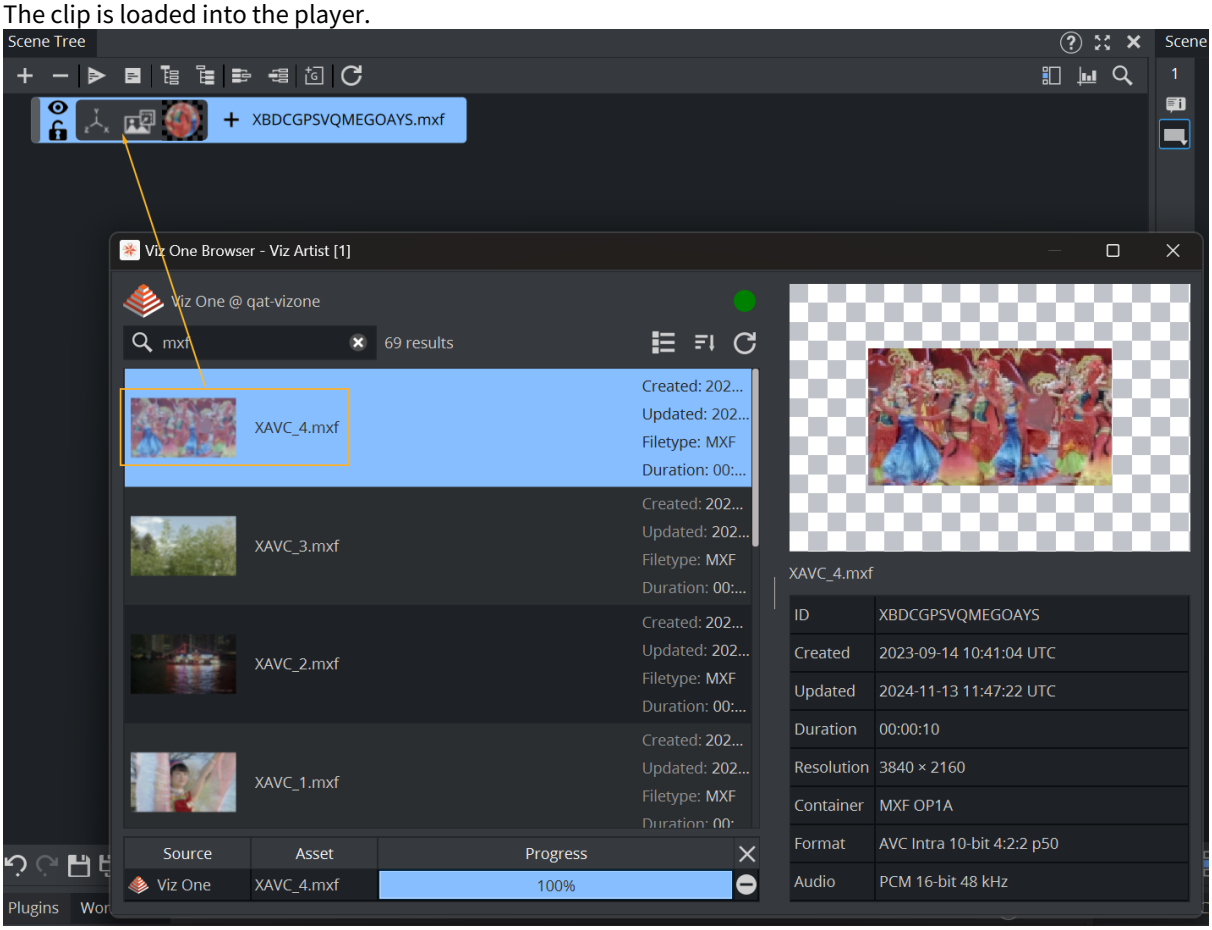
Copyright © 2026 Vizrt

The view can be switched between an **icon view** or a **list view**. Ordering allows to sort by modification date, name etc.
Once your item has been found it can easily be applied by simply dragging it to your desired location.

To Use Clips on a Container

The easiest way is to directly drag a clip from within the Viz One browser to your scene tree. This performs the following actions:

1. A new container is created.
2. A Clip Player as Texture Renderer is assigned.
3. The clip is automatically transferred to your configured Clip Location.
4. The clip is loaded into the player.

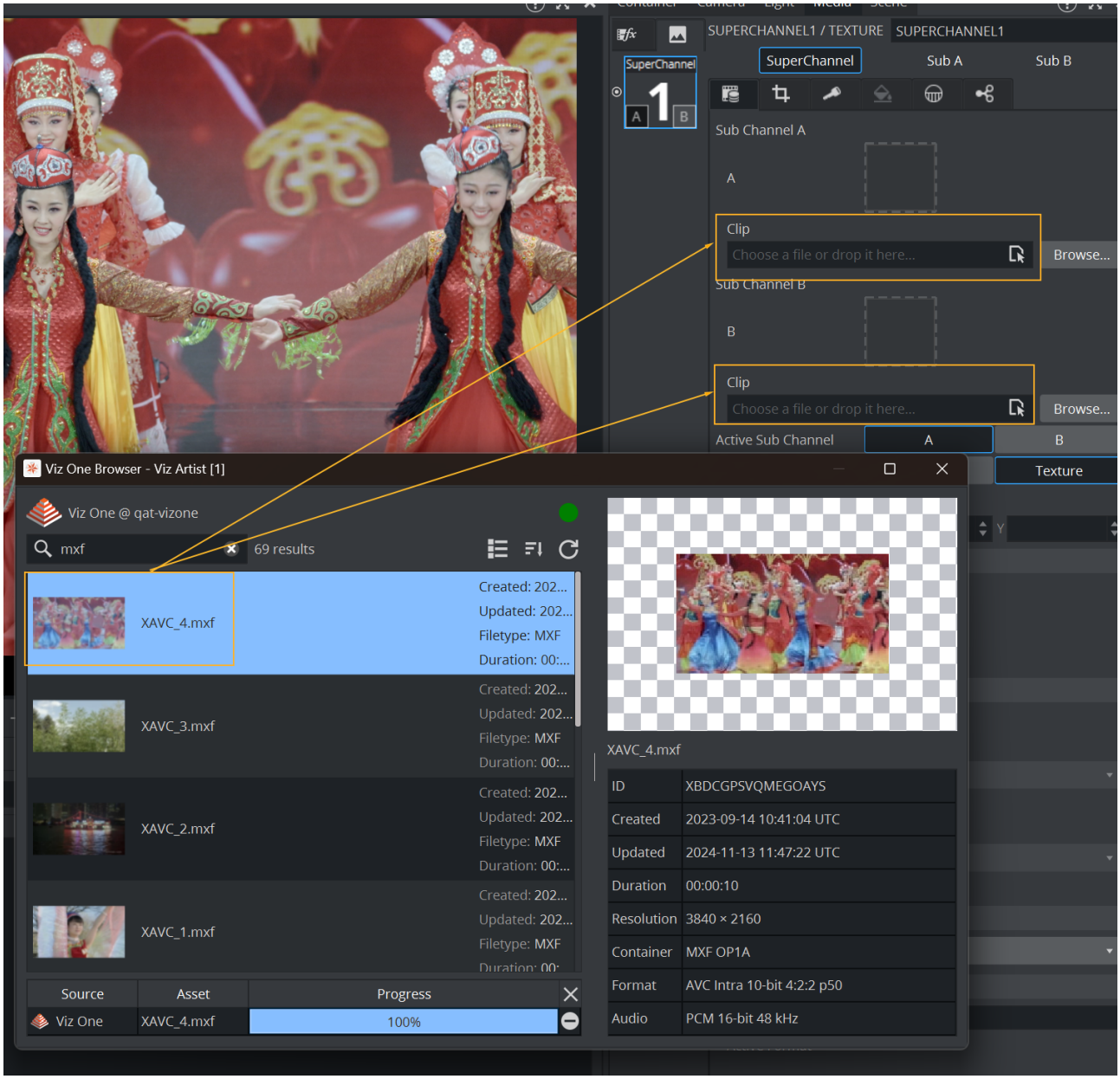


To Use Clips as Media Channel

By default, Clips from Viz One can be used by dragging them onto a container. However, if you want to assign them to a Media Asset (Like Clip Channel or Super Channel), do the following:

Create the wanted Media Channel (either a Clip Channel as DVD or Texture or a Superchannel)

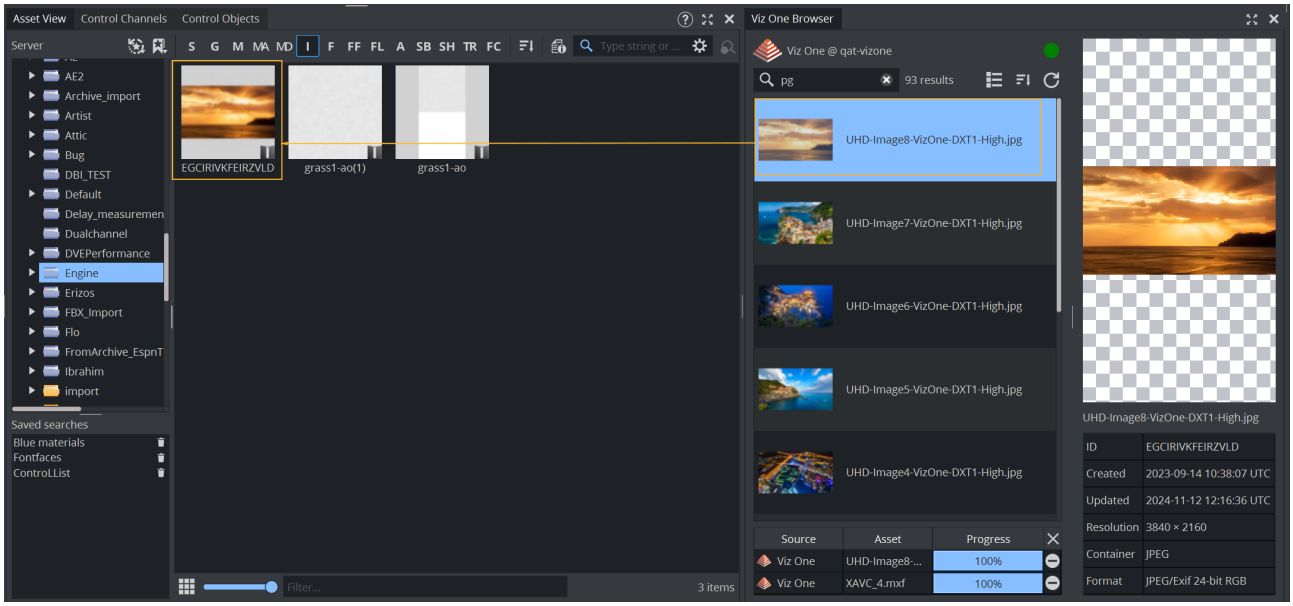
Copyright © 2026 Vizrt



To Use an Image

Images can (but should not) be used directly from a Viz One system, they need to be staged to Graphic Hub first. Therefore, you need to drag them from the Viz One Window (or dock) to the Asset view to start a transfer and import the image into Graphic Hub.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

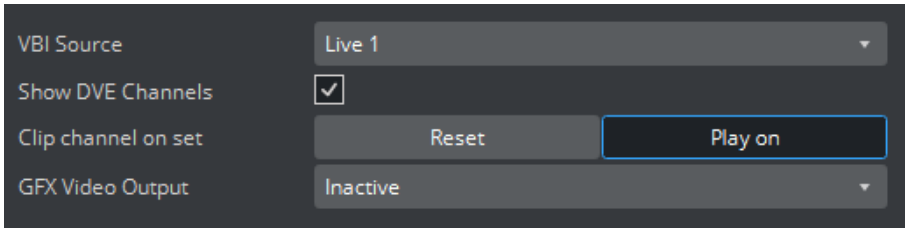
13.2 Media Asset Manager

This section contains the following topics:

- Global Settings for Media Assets
- Working with the Media Asset Overview
  - To Add Media Types
    - Media Assets as DVE
  - To Remove Media Types
  - To Switch Media Types
  - To Switch Priority
  - To Activate Media Assets
  - To Name Media Assets

13.2.1 Global Settings for Media Assets

The Global Settings for Media Assets are available in **Scene Media Assets:** See Global Media Assets Settings.

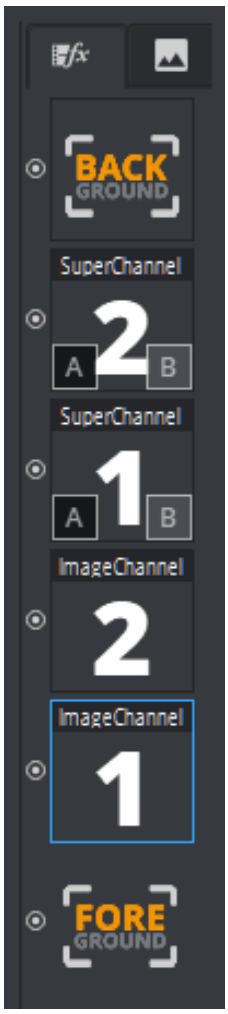
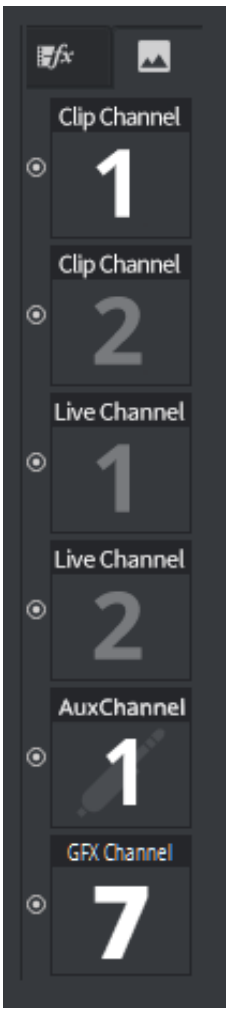


The Media Asset Manager is used to add and manipulate Media Assets. Media Assets can easily be added to a scene as either DVE or Texture asset types, depending on the scene's requirements and hardware capabilities.

DVE Assets are processed on the video board, therefore only a placeholder is usually shown. Texture media Assets need to be downloaded first from the videoboard, then modified and uploaded back to the videoboard. Working with DVE is faster, working with Textures gives more flexibility.

The Media Assets panel has been re-designed in Viz Engine 5. It requires less screen space and brings the manager closer to the settings. In addition, a preview icon is shown if possible. The Media Asset panel is now a sidebar within the Media Dock.

Copyright © 2026 Vizrt

DVE Mode	Texture Mode
	

13.2.2 Working with the Media Asset Overview

The Media Asset overview lists all Media Assets being used either as DVE, Texture or as Hybrid Surface.
The following Media Assets can be used:

Type	Description	Texture	DVE
FOREGROUND	Represents the Foreground Media rendered foreground graphics on the DVE Compositor.		✓
BACKGROUND	Represents the Background Media Assets rendered background graphics on the DVE Compositor.		✓

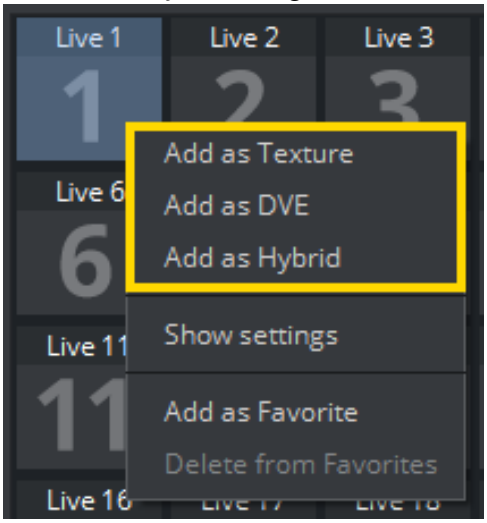
Copyright © 2026 Vizrt

Type	Description	Texture	DVE
Live Channel	Uses A SDI or IP live video feed. Can be used as Hybrid.	✓	✓
Clip Channels	Plays a Clip by the Matrox- or VML clip player.	✓	✓ (simulated in VML)
GFX Channels	Renders a Scene within another scene.	✓	✓
Image Channel	Represents 2D image with automatic adjustment features.		✓
Superchannel	Allows a combination of Live/Clip/GFX or Image Channels with A/B transitions.	✓ Viz Engine Pipeline only	✓
AUX Channels	Exchanges data with third party applications.	✓	
Web Channels	Displays Web content.	✓	✓

To Add Media Types

Media Types used as Texture automatically show up once assigned as a Texture. To add Media Assets either as Texture or DVE (or even as Hybrid),

- Select the entry in the Plugins Panel, section **Media Assets** and right click on it:

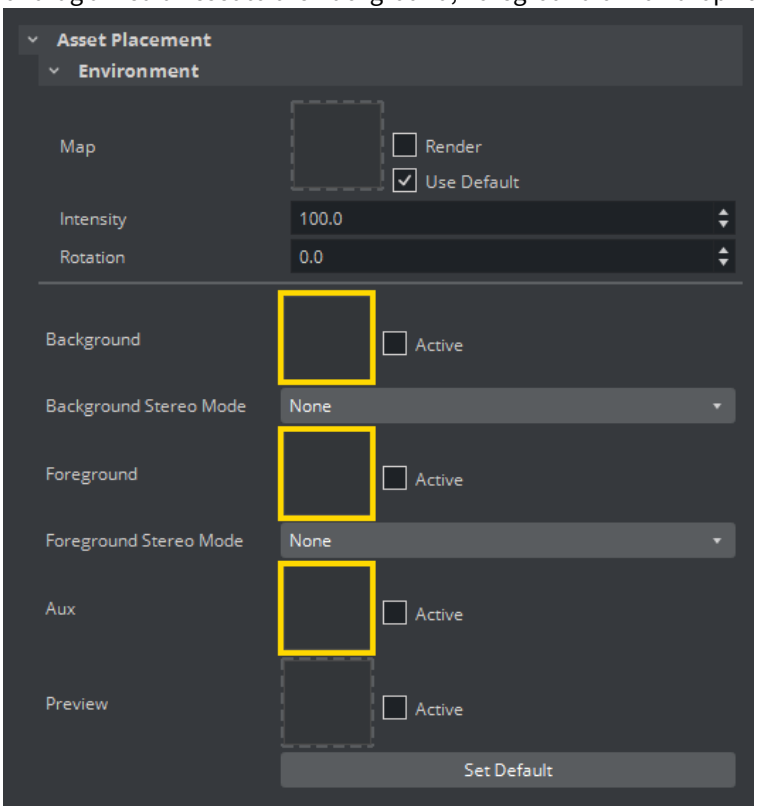


Copyright © 2026 Vizrt

- or activate the Media Dock (usually grouped together in the Container/Scene dock) and drag them from the Plugins panel to the side bar.

Note: Live Inputs can be used in Hybrid mode, which means, both Texture and DVE are available at the same time in versions 3.12 and later.

- or drag a Media Asset to the Background, Foreground or Aux drop zone in **Scene > Asset Placement**.



If a Media Asset is dragged to the Background or Foreground drop zones, it is applied in the Scene as a full screen texture.

- GFX channels can be added by dragging a scene onto the Media Asset Manager. The GFX channel needs to be dragged to the scene if it is going to be used as texture.

Note: If the **Video Input Layer Targets and Priority** settings are made **Active** in **Scene Default Values**, each active Media Asset is added to either the DVE or Texture section of the Media Asset Manager. They are set in order of priority as set in **Scene Default Values** (see the **Configuring Viz** section of the Viz Engine Administrator Guide)

Media Assets as DVE

Important: Hardware surfaces in Viz Engine (such as a Live Video Media Asset or Clip Channel Media Asset) are rendered with the video hardware. Software based DVEs (such as Image Channels and GFX Channels) are rendered with the Graphics card and then composited on the video hardware as one surface. The software renderer ensures the correct ordering within this surface, but as there is only one

Copyright © 2026 Vizrt

final surface, multiple hardware-based surfaces and software-based surfaces cannot be blended interchangeably.

If your DVE surfaces do not show up correctly, check the console output for a line like:
`[13:51:47.182] [E]: Error: (Clip9) No DVE supported in fast texture mode and/or DSX LE boards.`

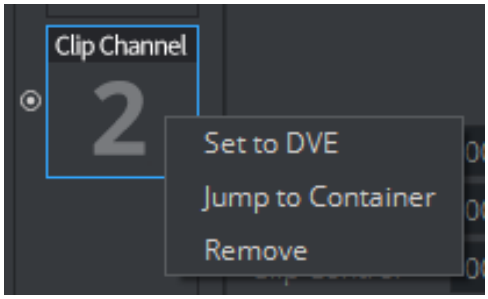
As DVEs are composed on video hardware, the graphics can be placed either in the front or in the back of the DVE layer. The main layer in Viz Artist defines this behavior. If the same scene is set into the background layer, this means the DVE effect is always drawn in the back of graphics.

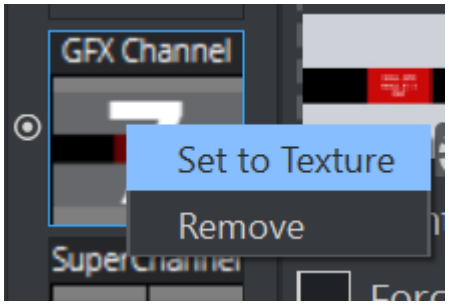
To Remove Media Types

- Right click in the Media Asset Manager and select **Remove**.
- Drag and drop it to the Trash bin icon.

To Switch Media Types

Once a Media Asset has been added to the Media Asset Manager, certain Media Asset can be switch from DVE to Texture and vice versa. To do so, also right click on the icon:



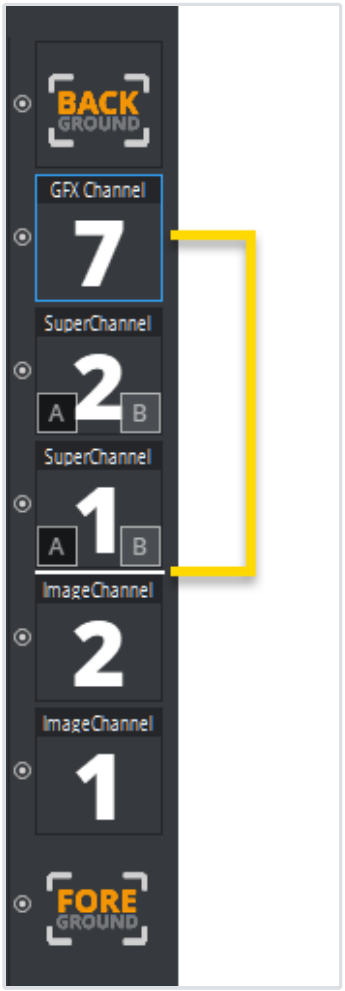


To Switch Priority

The Media Manager lets you set the rendering priority of Media Assets, by the way they are ordered. To change an asset's priority, drag it to a new position, a white line indicates the final destination. The asset with the highest **Order** value is rendered on top, with declining **Order** values below it.

Copyright © 2026 Vizrt

Information: Reordering is available for Media Assets as DVE only.

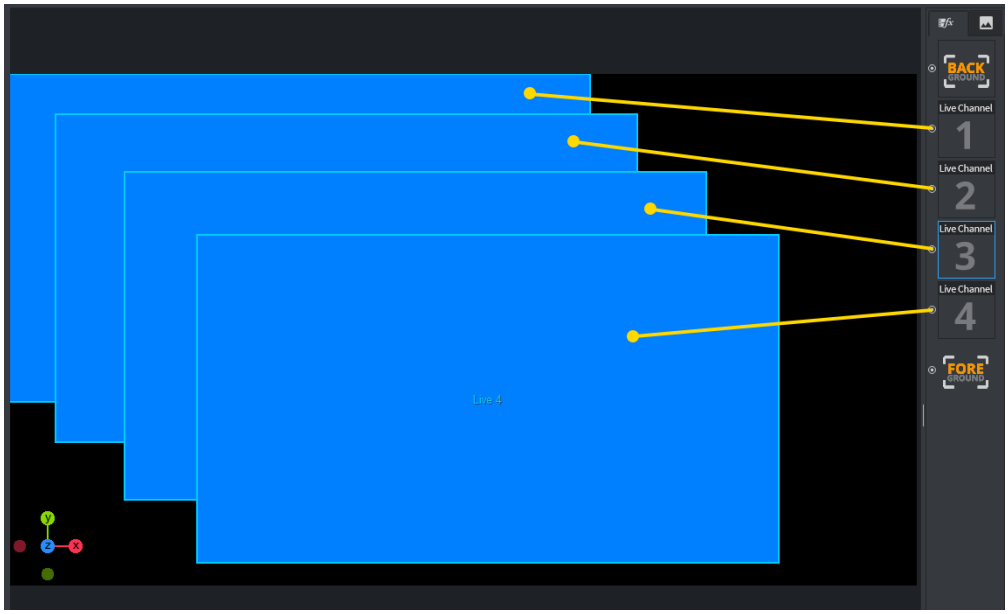


The **Background** and **Foreground** Media Assets are, by default, added as DVE layers. The Background Media Asset is always the first Media Asset to be rendered. The Foreground Media Asset can be ordered the same as other Media Assets.

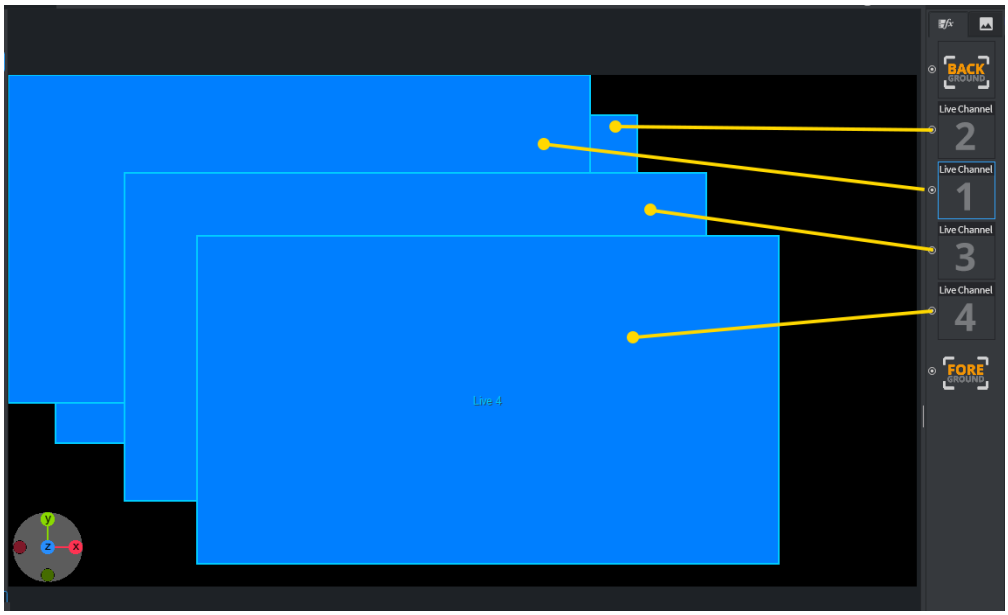
The order of each DVE Media Asset determine the order how each Media Asset is rendered. The first Media Asset in the list is rendered first, the second is rendered next, and so on. See the image below:

Example 1: The Live Input order is 1, 2, 3 and 4. So Live Input 1 is rendered first and Live Input 4 is rendered last.

Copyright © 2026 Vizrt



Example 2: The Live Input order is 2,1,3 and 4. So Live Input 2 is rendered first and Live Input 4 is rendered last.



To Activate Media Assets

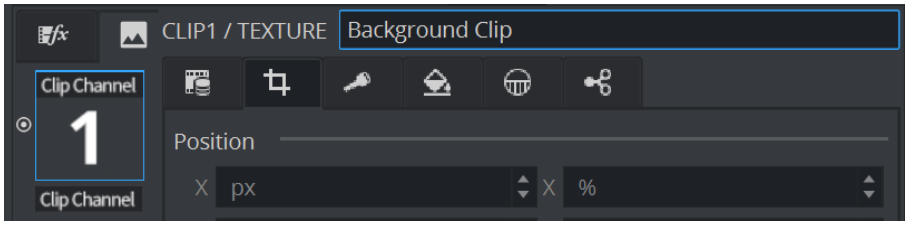
Media Assets can be turned on or off, by using the small checkbox next to the Media Icon.

Copyright © 2026 Vizrt



To Name Media Assets

Media Assets can be named.



Copyright © 2026 Vizrt

13.3 Media Asset Panel

This panel contains parameters and settings for Media Assets. There are different settings for all kinds of various Media assets like Clip Channels, Graphic Channels, Image Channels, Live Input Channels and Super Channels.

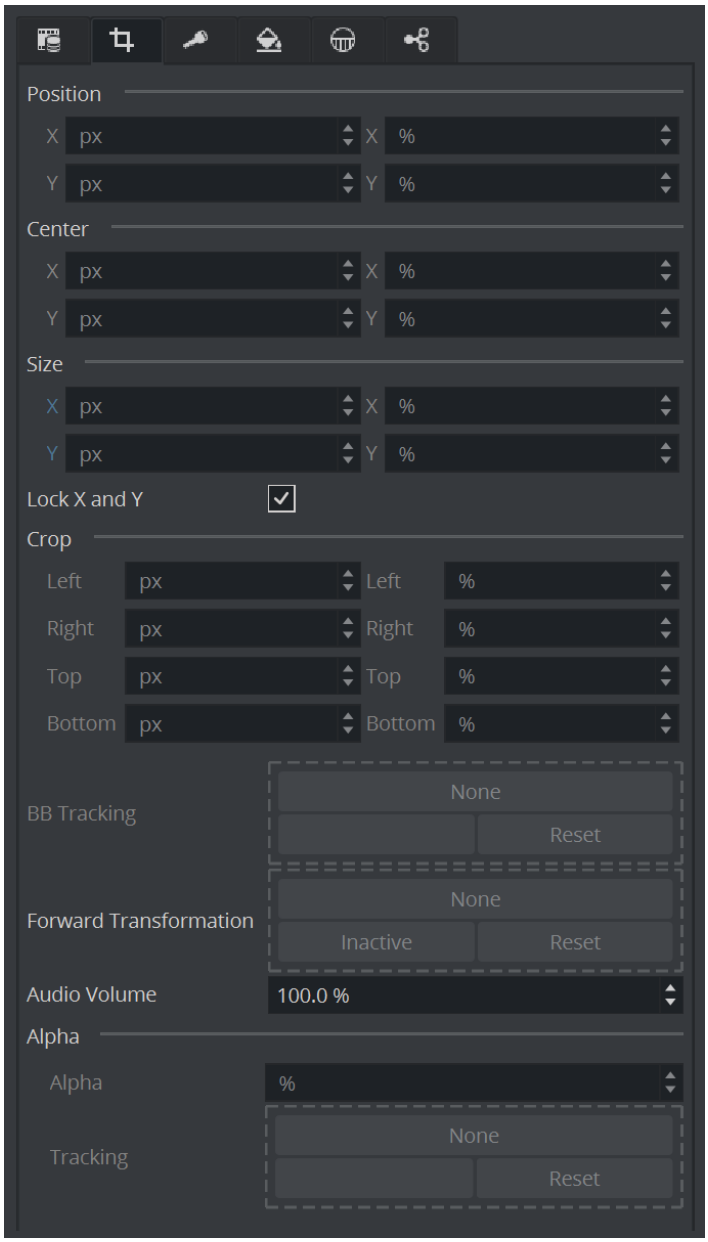
- Transformation Settings
- Media Asset Keying Settings
- Color Correction
- Talent Reflection
- Shared Memory (SHM) Aux

The individual settings are described in the following sub chapters:

- Clip Channel Media Assets
- Graphic Channels
- Image Channels
- Live Media Assets
- Super Channels
- AUX Channels

Copyright © 2026 Vizrt

13.3.1 Transformation Settings



- **Position:** The position controls are used to set the X and Y coordinates for the Media Asset. The coordinates are relative to the configured output format (see the **Configuring Viz** section of the Viz Engine Administrator Guide).
- **Center:** Uses the Center X and Y values to define the center position of the Media Asset. As for the **Size** control, the **Center** position is relative to the configured channel resolution.

Copyright © 2026 Vizrt

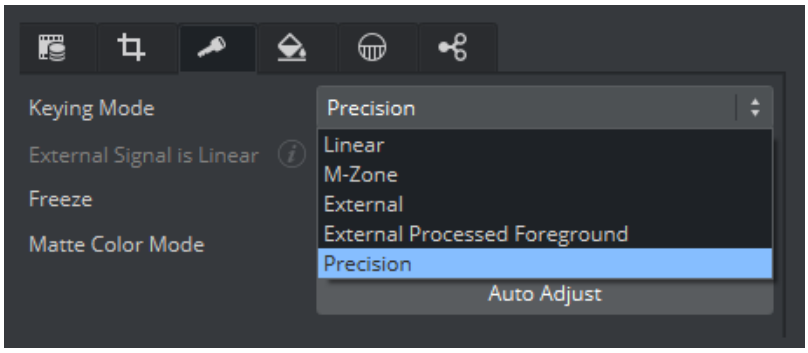
- **Size:** Sets the X and Y values to define the size of the Media Asset. The size is based on the channel resolution as defined in the **Video Input** section of **Configuring Viz** (see the Viz Engine Administrator Guide), which may be different than the configured output resolution.
- **Crop:** Media Assets can also be cropped. The crop values are relative to the size as defined using this editor.

Tracking options: There are several options available to either follow your Media Asset or to position your Media Asset. according other objects. This can be done by selecting either "**DVE follows object**" (The position and size of the DVE follows any other object) or by "**Object follows DVE**" to align any other to this Media Assets size an position. In addition the transparency of your Media Asset can also be bound to the alpha of any other object. (**Alpha Tracking**)

Absolute / Relative Position Controls: Controls to adjust the **Position**, **Size**, **Center** position, and **Crop**. The values can be provided either as **Absolute** pixel values, or as **Relative** percentage values. The corresponding value is displayed next to the input value field. If provided with values that exceed the logical limit, the last value is adjusted automatically to the maximum value. For example, if setting the Left crop value to 51 %, and then the **Right** crop value to 75 %, the Right crop value is set to 49 %. Applies to DVE only.

> **Note:** By default, the center position for DVE transformations is defined as the top left corner of the DVE placeholder. Setting the DVE Center `X` and `Y` positions to `50 %` defines the center for the transformations as in the middle of the DVE placeholder. This behavior deviates from that of transforming Containers, as they have their transformation center in the middle.

13.3.2 Media Asset Keying Settings



In addition to the common Media Asset properties, Keying has these properties.

> **Note:** A Chroma Keyer License is required!

- **Keying Mode** (Texture only: see Keying Mode):
  - **Linear:** No chroma keying.
  - **M-Zone:** Uses the multi-zone Keyer.
  - **External:** To be used with Arena/Libero.
  - **External Processed Foreground:** Reads processed foreground and matte from an external signal and applies the postprocessing steps of the Precision Keyer.
  - **Precision:** Switches to the Precision Keyer.

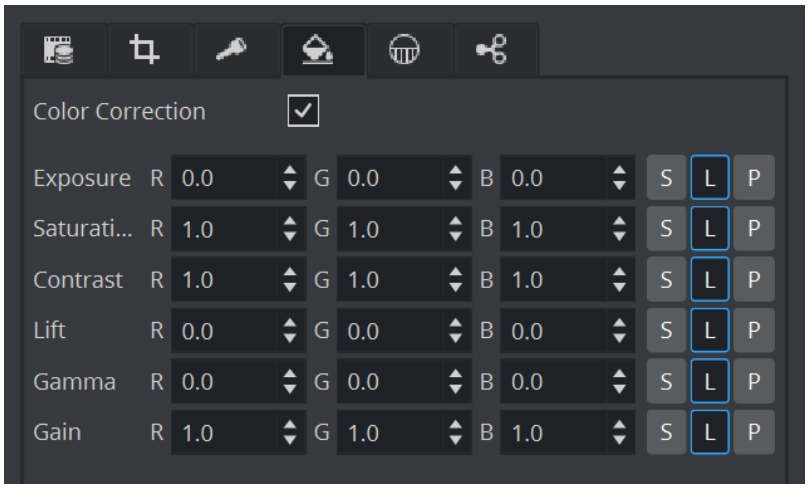
Copyright © 2026 Vizrt

- **External Signal is Linear:** Whether Viz Engine should skip color space conversion as the external signal is already linear. Only available with **External Processed Foreground** keying mode.
- **Freeze:** Stops the incoming live signal from any movement (freezing). This is used during finetuning your chroma key.
- **Matte Color Mode:** Selects which kind of matte is used. It can be a **Greenscreen**, a **Bluescreen** or the matte can be sent from **external.**

Auto Adjust: Automatically adjusts the main parameters of the Precision Keyer.

Tip: It is recommended to use an external application like Viz Arc to set your Keyer properties.

13.3.3 Color Correction



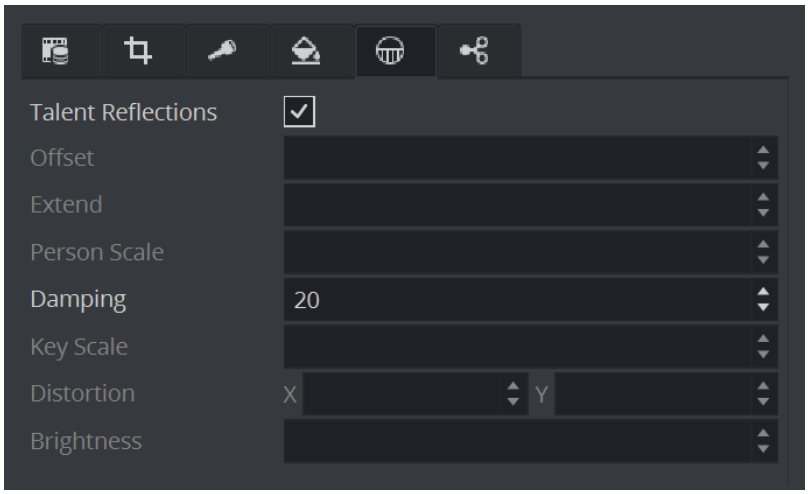
This panel allows modification of the rendered output of any Media Asset. It can be used to modify (for example, clips) or to finetune the content on videowalls.

The following options are available:

- **Exposure**
- **Saturation**
- **Contrast**
- **Lift**
- **Gamma**
- **Gain**

Copyright © 2026 Vizrt

13.3.4 Talent Reflection



Talent Reflection: Sets Talent Reflection `On` or `Off` (use with Viz Virtual Studio):

- **Damping:** Sets how many previous fields are to be used to calculate the current **Y** position. The larger the number the higher the damping, but the **Y** coordinate is less responsive to quick movements. The setting is a compromise between a stable and a non-responsive **Y** coordinate for the reflection (Default: `2` , Minimum: `2` , Maximum: `100`).

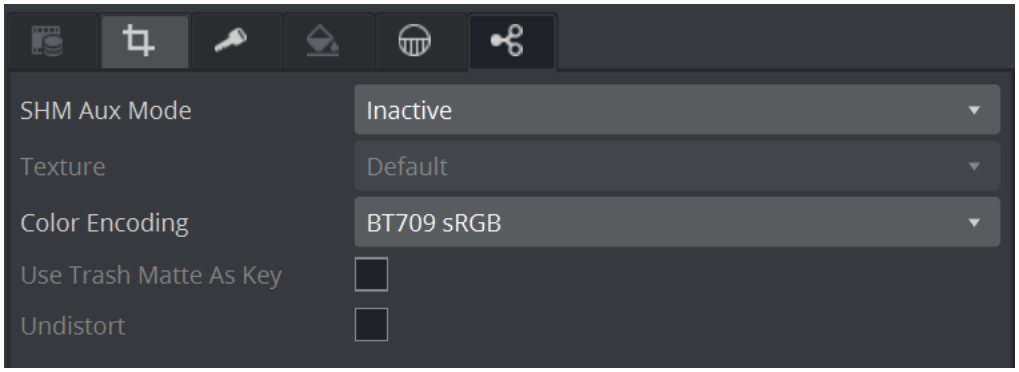
The following options are available in the Classic Render Pipeline only.

- **Offset:** Sets the offset of the talents lowest body part from where the reflection starts (pixels (Default: `0` , Minimum: `0` , Maximum: `1000`)).
- **Extend:** Sets the extend of the total reflection (pixels). The reflection eases out with distances (Default: `300` , Minimum: `0` , Maximum: `1000`).
- **Person Scale:** Sets a global scale which applies to the talent (Default: `1` , Minimum: `0.2` , Maximum: `10`).
- **Key Scale:** Sets the scale applied to the alpha of the reflection, on top of the parameters set in **Offset** (Default: `0.5` , Minimum: `0.1` , Maximum: `10`).
- **Distortion X / Y:** Parameter **X** defines by how much x axis is affected by the sinus distortion which goes vertically down the screen (Default: `1` , Minimum: `1` , Maximum: `10`). Parameter **Y** defines how much the sinus waveform is stretched vertically down the reflection (Default: `0` , Minimum: `0.001` , Maximum: `1`).
- **Brightness:** Sets the brightness for video going through the Chroma Keyer (Default: `0` , Minimum: `0` , Maximum: `10`).

13.3.5 Shared Memory (SHM) Aux

Defines if the content of this Media Asset should be send as Shared Memory. This is needed for Integration with third party render engines. For integration with Unreal Engine, please select the respective setting.

Copyright © 2026 Vizrt



- **Texture:** Defines what kind of texture should be send through the Shared Memory output.
- **Color Encoding:** Selects the colorspace for the shared texture to use (not available on all types).
- **Use Trash Matte as Key:** Includes the Trashmatte into the signal (for the AI Keyer).
- **Undistort:** Performs an undistortion action before the texture is being sent.

Copyright © 2026 Vizrt

13.4 Media Asset Channel Types

The following media asset channel types are available:

- AUX Channels
- Channel Folder Media Assets
- Clip Channel Media Asset
- Container Folder Media Assets
- GFX Channels
- Image Channels
- Live Video Media Asset
- Super Channels
- Web Channels

Note: DVE Assets and Placeholders in Artist are only shown for the current layer for better visibility (for example, if a front layer shows a Fullscreen DVE asset, it is hidden when working in the main layer).

Copyright © 2026 Vizrt

13.4.1 AUX Channels

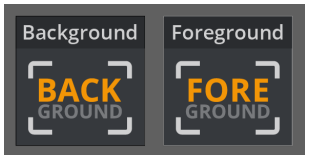


AUX channels are used to exchange data between Viz Engine and other Third Party Render Blades (for example, Unreal Engine). The settings are the same as for other Media Settings.

Copyright © 2026 Vizrt

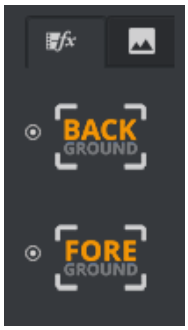
13.4.2 Channel Folder Media Assets

Background and Foreground Media Assets are located in the Channel folder of the Built Ins panel.



The Background and Foreground Media Assets represent the rendered background and foreground graphics on the DVE Compositor.

Note: By default, a Background and Foreground Media Asset is applied to each new Scene.

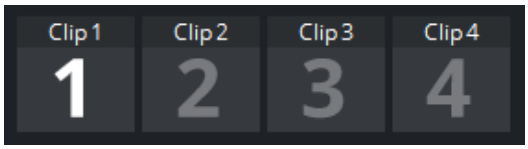


The Foreground Media Asset can be ordered as required, but the Background Media Asset is always rendered first. Both the Background and Foreground Media Assets can be deleted. If they have been deleted, drag the Background and, or the Foreground Media Asset icon, from the Server Panel, to the Media Asset manager, if required.

Note: Only one Background and one Foreground Media Asset can be used in the DVE Media Asset Manager at a time.

Copyright © 2026 Vizrt

13.4.3 Clip Channel Media Asset

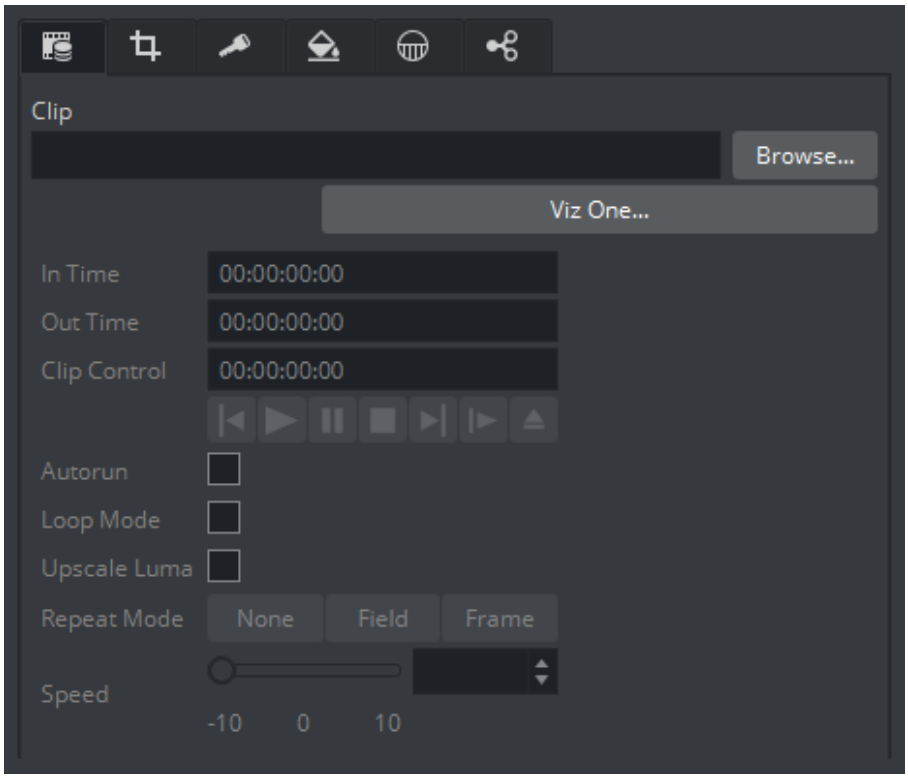


Clip Channels represent clip players. The clip players are either hardware bound native players (Matrox) or software based player like VML. Clips can be used either as DVE or Texture effect. Clips set to be inactive are shown as grey, active clip players are shown as white number.

With the Clip Channel Media Asset it is possible to play images as **DVE** or **Texture** with the Matrox clip player. Supported image containers are *.jpg* and *.png*. The images have to be available on a physical disk for this feature to work. See a detailed description of the functionality in the Media Asset Panel.

Note: DVE is simulated for software players like VML Player.

In addition to the common Media Asset properties, Video Clip also has these properties:



- **Clip:** Selects the video clip to be shown:
  - In the **Clip** line, click **Browse**, or
  - Drag a Video Clip item from the Server Panel, or

Copyright © 2026 Vizrt

    - Click the Viz One button, and select a video clip to import.
- **In Time:** Sets the video clip 'in time'.
- **Out Time:** Sets the video 'out time'.
- **Clip control:** Shows player controls for the selected clip.
- **Autorun**
    - **Off:** Presents the first frame as soon as the video clip is loaded.
    - **On:** Begins playback as soon as the video clip is loaded.
- **Loop Mode**
    - **Off:** Plays the video clip once.
    - **On:** Plays the video clip in a loop.
- **Upscale Luma**
    - **Off:** Uses the key as is.
    - **On:** Upscales the key information from `16-235` to `0-255`.
- **Repeat Mode**
    - **None:** Plays back black frames when the clip is in an erroneous state.
    - **Field:** Plays out the first field of the last frame when the clip is paused.
    - **Frame:** Plays out the last frame when the clip is paused.
- **Speed:** Controls playback speed. `1.0` is the normal speed. Values between `0` and `1.0` are forward playback in slow motion. Values above `1.0` are forward playback in fast motion. Likewise, values between `0` and `-1.0` are reverse playback in slow motion. Values less than `-1.0` are reverse playback in fast motion.

Note: See Transfer Clips From Viz One. To transfer and download videos from Viz One, Viz One must first be integrated and configured with Viz Artist/Engine.

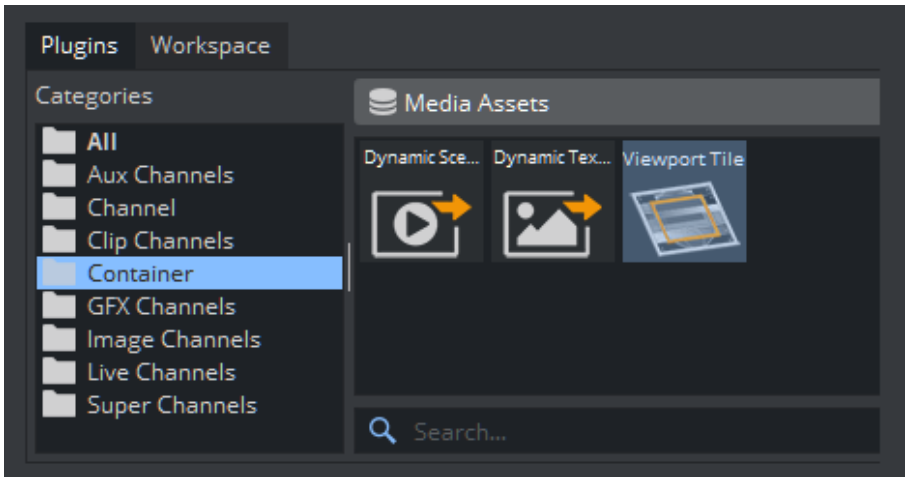
Note: With the Clip Channel Media Asset it is possible to play images as DVE or Texture with the Matrox clip player. Supported image containers are *.jpg* and *.png*. The images have to be available on a physical disk.

See Also

- Video Clips

Copyright © 2026 Vizrt

13.4.4 Container Folder Media Assets



The following Media Assets are located in the Container folder of the Built Ins panel:

- Dynamic Scene Media Asset
  - Dynamic Scene Texture Properties
  - Dynamic Scene Dynamic Properties
- Dynamic Texture Media Asset
- Viewport Tile Media Asset
  - Viewport Tile Properties
  - To Render a Scene in a Viewport Tile

Dynamic Scene Media Asset



The Dynamic Scene is a Media Asset that is able to render a different scene, local scene or parts of a local scene into a texture and show the texture in the current scene being rendered.

Showing a different scene as 'a window' in the current scene can be used in a virtual studio scene where you would like to 'open a window' (as a texture) and render a different scene (for example, a weather forecast).

You can also use it to show parts of a local scene if you want to animate the camera and at the same time show static items in the scene. The animating camera can be a different camera (two) and be used to show dynamic images in camera one's view. This way you can show more than one camera at the same time.

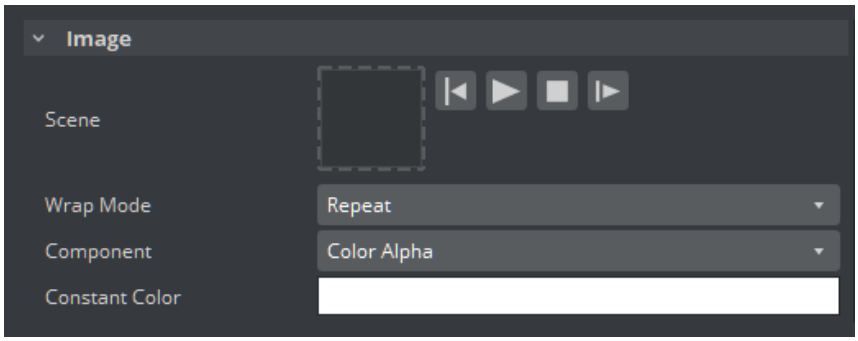
For visual effects, the whole scene or parts of it can be pulled into an image, and have Shaders applied to the whole scene or parts of it.

Copyright © 2026 Vizrt

To transition from one scene (A) to another (B), both scenes are rendered into dynamic images and a third scene is used to hold both dynamic images to create a transition between them.

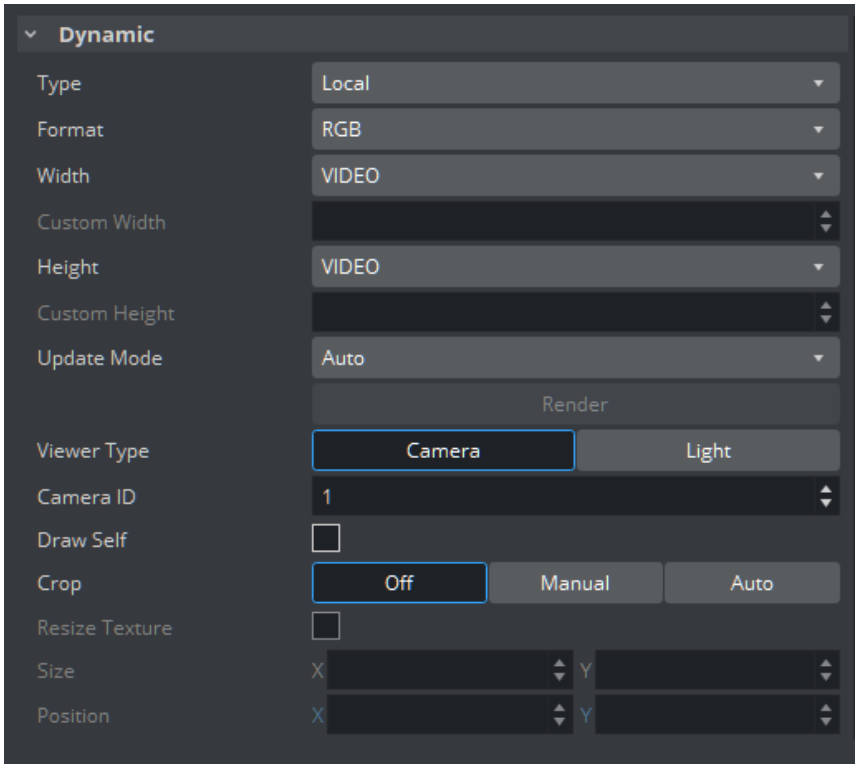
Note: This Media Asset is located in: Built Ins -> Media Assets -> Container

Dynamic Scene Texture Properties



- **Scene:** Placeholder for the rendered scene. If it is empty, the current scene is rendered.

Dynamic Scene Dynamic Properties



Copyright © 2026 Vizrt

- **Type:** Determines what is used as dynamic scene. This could be the same scene (local), another scene or another layer.
- **Format:** Sets the format to be used for the dynamic image. If the format has alpha key it must be used on the render scenes or it shows as black.
- **Width / Height:** Sets the width and height of the image. The Video option takes the current output size. When selecting Custom you may set any size; however, it is always better (when possible) to use power of two images.
- **Update Mode:** Sets when the scene in the dynamic image is rendered.
    - **Change:** Renders the scenes when anything changes.
    - **Always:** Renders each frame regardless if there was any change.
    - **Command:** Renders the scene when the render command is sent.
    - **Grab:** Renders the scene on command and can then be copied to another image.
- **View Type:** Sets the view port to Camera or Light. If Camera is selected and Camera ID is set to zero (`0`), it takes the current camera. If Light is selected, it uses the light according to the Light ID.
- **Draw Self:** Creates an endless mirror effect.
- **Crop:** Enables different crop modes.
    - **Off:** Renders the entire view port.
    - **Manual:** Enables you to set the size and position of the rendered scenes.
    - **Auto:** Sets the size and position of the rendered scene by the bounding box holding the dynamic scene (acts as a mask).
- **Size / Position:** Sets the size and position of the dynamic image.
- **Resize Texture:** Should the image be resized automatically to reflect the size of the scene being rendered.
- **Back/Play/Stop/Forward:** Controls the playback of the scene.

Dynamic Texture Media Asset



Dynamic Texture is a feature that generates dynamic textures on-the-fly.

It can be used as an alternative to importing textures to save texture memory usage.

Another advantage of using a dynamic texture is that a texture is not pixelated as you zoom in or out of it.

> ⚠ **Note:** This Media Asset works for the Classic Render Pipeline only.

Copyright © 2026 Vizrt

Viewport Tile Media Asset



The Viewport Tile renders a Scene into a part of the current viewport with the need for a texture, like a GFX Channel Media Asset.

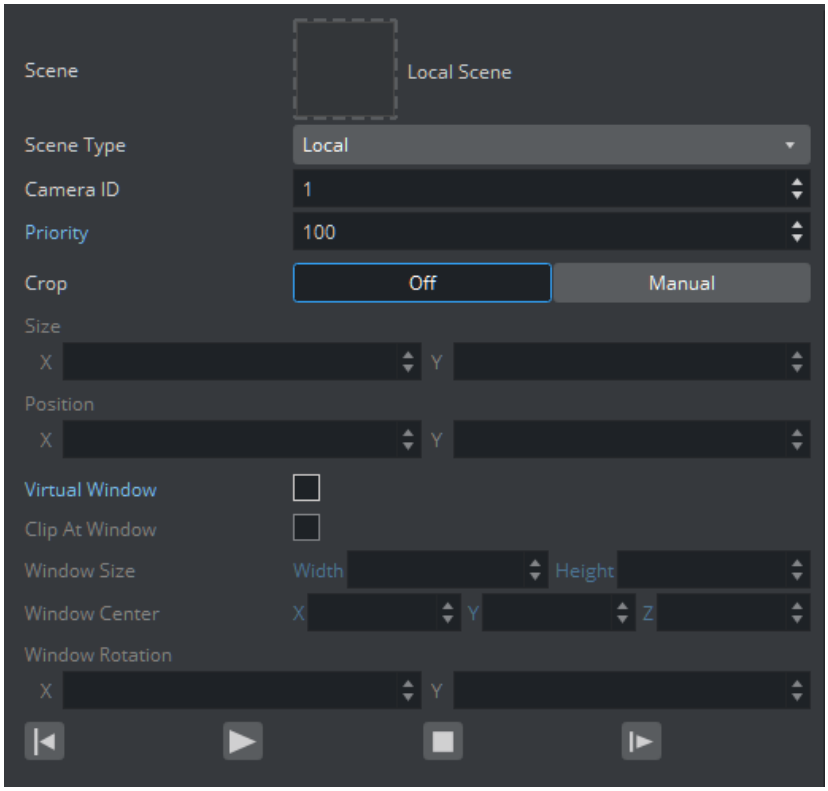
IMPORTANT! This Media Asset is considered legacy. Please use GFX Channel Media Assets when designing new scenes in Viz Artist.

- **Advantage:** The rendering can be slightly faster in specific use-cases.
- **Disadvantages:**
  - Viewport tile does not support post processing effects such as lens distortion and defocus.
  - The Viewport Tile is always orthogonal to the camera. Viewport tiles can also be interactive and make use of the regular mouse and multi-touch events. However, Viewport tiles can not be nested. A Scene with a Viewport tile, which contains a Scene, can not be used as a Scene in another Viewport tile.

Note: This Media Asset is located in: Built Ins -> Media Assets -> Container

Copyright © 2026 Vizrt

Viewport Tile Properties



- **Scene:** Placeholder for the rendered scene.
- **Scene Type**
  - **Local:** Renders the current scene.
  - **Other:** Renders the Scene put in the Scene drop zone (drag and drop a scene).
  - **Main Layer:** Renders the scene of the main layer.
- **Camera ID:** Select a camera for the Viewport Tile scene.
- **Crop**
  - **Off:** No cropping.
  - **Manual:** Define the new viewport **Size** and **Position** with the parameters X and Y.
- **Virtual Window:** Allows rendering as a virtual window. Width, Height, Center and Rotation are used to tweak the appearance.
- **Play controls:** Used to play, stop, etc. the animations of the Viewport Tile scene.

To Render a Scene in a Viewport Tile

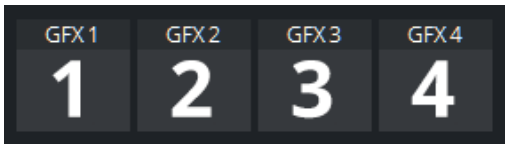
1. Add the **Viewport Tile** to the Scene Tree.
2. Open the Viewport Tile editor and add a Scene to the **Scene** placeholder.
3. Open the **Transformation Editor.**
4. Set the **Screen Size** (**width** and **height**) of the Viewport Tile.
5. Open the Viewport Tile editor.

Copyright © 2026 Vizrt

6. Click the **Play** button.

Copyright © 2026 Vizrt

13.4.5 GFX Channels



Graphics (GFX) Channels allow a designer to compose several independent scenes into a new scene. Utilizing a GFX Channel, a previously created scene can be added to the current scene as a DVE or Texture.

GFX Channel Media Assets can be nested. This means a scene with a GFX Channel Media Asset containing a scene, can be used as a scene placed within another GFX Channel Media Asset. However, nested scenes cannot use the same GFX Channels as the scene they are nested into. For example, if *Scene 1* uses GFX channel 1, it cannot be nested within GFX Channel 1 of *Scene 2*. It can however be nested within GFX channel 2.

Also, note that cropping GFX channels have an impact on the render performance. You can see a detailed description of the functionality in the Media Asset Panel.

- Scene Instances
- GFX Properties
    - Software Composition
    - Adaptive Scene Design Formats
- Performance Considerations for GFX Channels
- Shared Memory Output for GFX Channels
- NDI Output for GFX Channels
- Adaptive Scene Design Formats Support for GFX Channels

Scene Instances

When loading scenes into GFX Channels, multiple instances of a scene may be loaded in the scene pool. For each unique scene assigned to a GFX channel, Viz Engine loads the scene once from the Graphic Hub to a so-called "template instance". Then, this scene is cloned locally into a separate instance for this GFX channel. When the same scene gets assigned again to another GFX channel, the template scene is cloned again to a new separate instance for this GFX channel. The same scene can be assigned to multiple GFX channels without interfering with each other and additional multiple lookups to the Graphic Hub are prevented.

Thus, when accessing scenes in GFX channels using the normal communication port instead of dedicated GFX ports, its not enough to specify the UUID like:

```
SCENE*<45263836-8A7F-47AC-969C7BA08E7AADC1>*TREE DO_SOMETHING
// will access wrong scene instance!
```

Instead, the scene should be accessed by querying the object ID first:

```
// query object id first
MAIN_SCENE*GFX*1 GET_OBJECT
returns <#998>
```

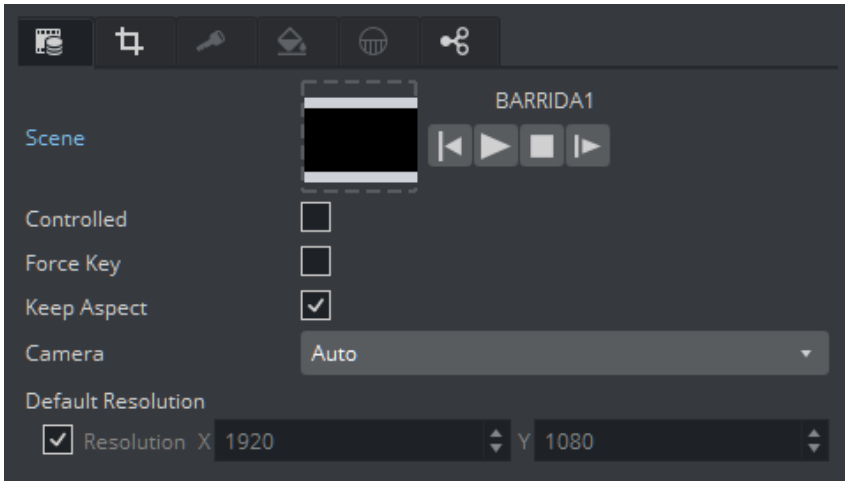
Copyright © 2026 Vizrt

```
SCENE*#998 DO_SOMETHING
```

Note: This means that there may be additional instances of the scene, even if a scene is only loaded into one GFX channel. Do not rely on the assumption that there is only 1 instance of a scene when working on scripts or plugins.

GFX Properties

By default, a GFX Channel automatically inherits the camera settings of the Scene it is applied to. In addition to the common Media Asset properties, GFX Channels also have the following properties:



- **Scene:** Drag a scene, which should be rendered in the GFX channel.
- **Play Controls:** Controls the animation of the set scene.
- **Controlled:** Controls the GFX Channel animation using the main or parent Scene animation controls if active. For example, if the animation in the main Scene starts, it triggers the start of the GFX Channel Scene.
- **Force Key:** Renders the entire Scene as keyed if the Scene does not have any keying information. If a GFX channel is rendered as Texture, it is in RGBA. Therefore, the Scene needs to have keying information for an alpha channel (the key scene setting is set to auto key or a key plug-in is used in the Scene Tree).
- **Keep Aspect:** Renders the GFX Scene with its own aspect ratio (undistorted) if set to `On` in DVE mode. The Scene is scaled to fit the GFX channel size if set to `Off`.
- **Camera:** Selects the camera view of the GFX Channel Scene. *Auto* means that the current camera setting of the loaded scene is used (may also change due to animations within the loaded scene).
  - **MAIN_SCENE:** Inherits the camera from the main scene.
  - **AUTO:** Uses the camera from the local scene and supports animations in the local (nested) scene.
  - **1-16:** Overrides camera view (fixed).
- **Resolution:** Defaults the GFX Channels to the resolution configured for the current *Output Format* in the Viz Engine configuration. To change the current resolution, click the **Default Resolution** button to make the **Resolution X** and **Y** fields available for input, and enter the desired resolution setting. Clicking the Default resolution again resets the resolution for the GFX Channel, as well as disable the **Resolution**

Copyright © 2026 Vizrt

X and **Y** input fields. If the Default Resolution button is in a pressed state, and the Resolution X and Y fields are dimmed, the default resolution is in use.

Software Composition

Content is composed with either **Alpha Blending** or a **premultiplied with the alpha** component. This can be set for the RGB (color) and the Alpha part (Key) of the Media Asset. Changing the composition may be required for correct blending with third party applications.

Adaptive Scene Design Formats

When using various formats within one scene, the format can be set explicitly. If multiple outputs are required, a scene hosting multiple GFX channels running multiple formats of one scene can be used. See Adaptive Scene Design.

Note: The resolution setting only applies to GFX Channels in Texture mode.

Performance Considerations for GFX Channels

Rendering a GFX channel means typically that a second scene has to be rendered in multiple render passes. To get the best performance out of this, please consider a few hints:

- GFX channels can be rendered in three internal modes (from fastest to slowest):
  - The scene is directly rendered on top of the main scene, using the correct size as needed.
  - The scene is rendered in a temporary texture (using the correct size) and this texture is blended into the main scene (needed when render sequence of nested scene is *Overlay*, or resulting texture is used with alpha < 1.0 in main scene).
  - The scene is rendered in a temporary texture (using **full render size**) and this texture is blended and scaled down into the main scene (needed when inner scene contains Defocus, Glow, or a Magnifier).
- This leads to two main performance tips:
  - If your nested scene is a fullscreen scene, make sure that you are using render sequence **Fullscreen** instead of using **Overlay**.
  - Consider if you really need postprocessing effects Defocus, Magnifier or Glow in your nested scene, especially if the GFX channel is scaled down in your main scene.

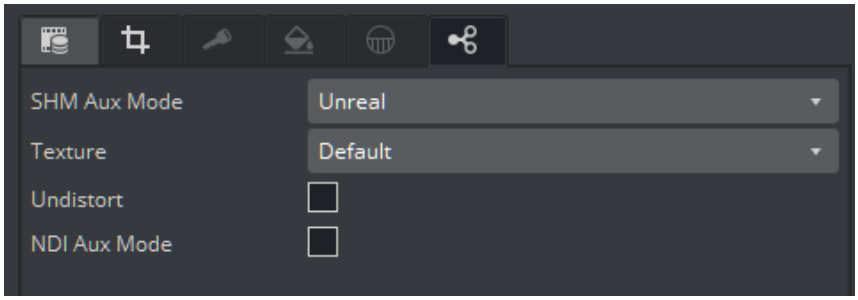
Info: Some post effects in the Viz Classic Render Pipeline are only available if the GFX channel has the same size as the render output, otherwise they are disabled automatically.

Shared Memory Output for GFX Channels

Shared Memory (SHM) output is now supported from one or multiple GFX channels with configurable output resolutions (different than overall output format), Alpha, and 16-bit textures.

It is possible to send the output of the GFX channels to Shared Memory to exchange data with third party applications using Viz Artist.

Copyright © 2026 Vizrt



- **SHM Aux Mode:** Which mode to be used to exchange data with third party applications.
  - **Send:** General mode for any third party application.
  - **Unreal:** Defines the behavior when used with Unreal Engine. The texture (either the default resolution or the original size) can be **distorted** or **undistorted**.

You can get the related output using Aux Key. The default value for aux key is: `viz_<instance_id>_gfx<channel_id>_aux` (for example, *viz_01_gfx1_aux*).

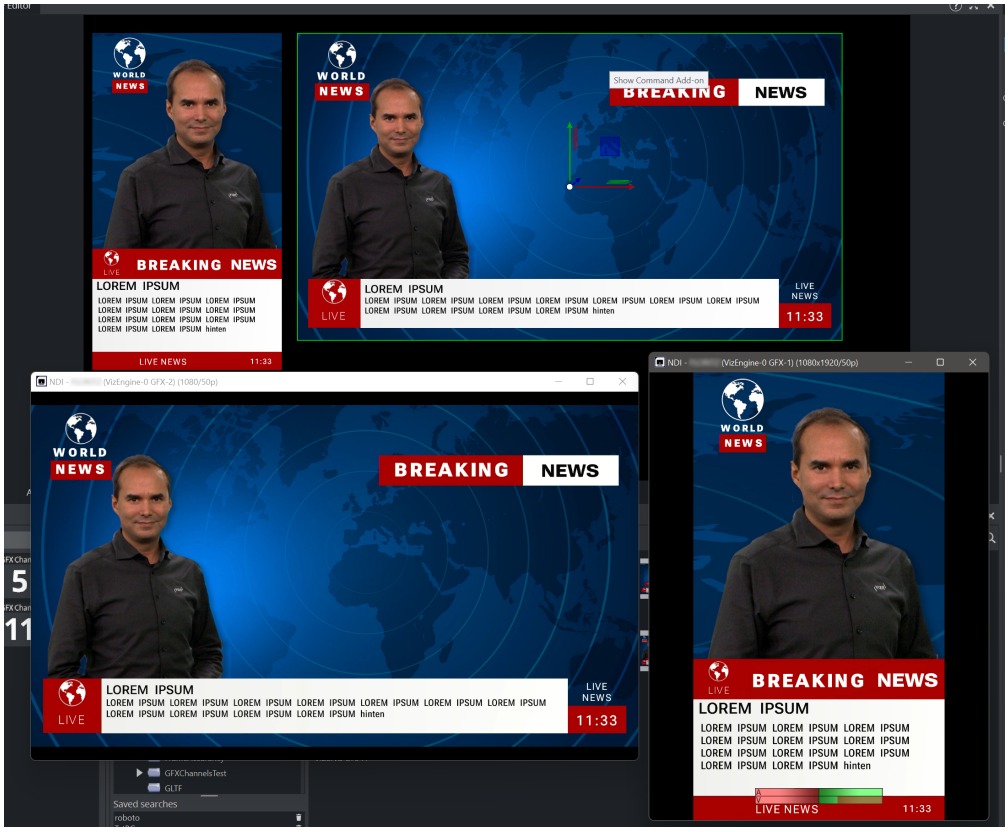
> **Note**: DVE mode is not supported.

NDI Output for GFX Channels

To support multiple outputs, the content of any GFX channel can also be broadcasted into an NDI network. The name of the NDI stream is then *<engine-instancename> GFX-n* (for example, *VizEngine-0 GFX-1*).

For more details see **NDI Output from GFX Channel** in the Viz Engine Administrator Guide.

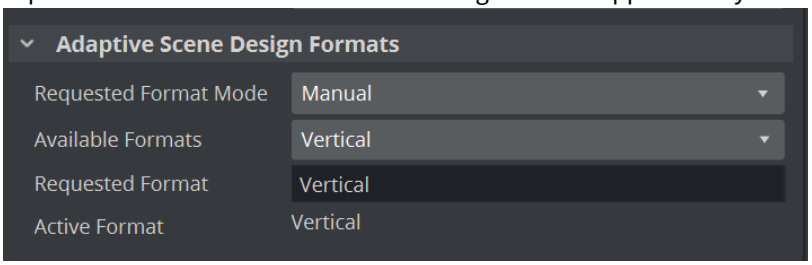
Copyright © 2026 Vizrt



Adaptive Scene Design Formats Support for GFX Channels

GFX channels can be set to render a certain format using the command or Viz Artist:

- There are several options available for Requested Format Mode:
  - **Manual:** Renders a given format. Can be set using **Available Formats**.
  - **Inherit from Main Scene:** Uses the format of the GFX Channel Master scene.
  - **Use Aspect Ratio:** Selects the format automatically based on the width/height, and therefore, the aspect ratio of the scene. The first matching format is applied. Only works for DVE.




```
#<sceneID>*GFX*1*REQUESTED_FORMAT SET <name of format>

#<sceneID>*GFX*1*REQUESTED_FORMAT_MODE SET <mode> // options are MANUAL,
INHERIT_FROM_MAIN_SCENE or USE_ASPECT_RATIO
```

Copyright © 2026 Vizrt

You can have one scene with two GFX Channels, one rendering in 16:9 and one in 9:16 portrait mode, and two NDI outputs, one in portrait mode and one in 16:9 mode.

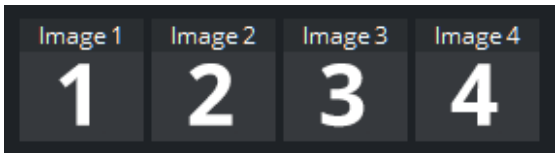
Note: If the requested format is not supported by the GFX channel scene, then it falls back to the default format as active format, and the active format and requested format differ.

See Also

- **NDI Output from GFX Channel** in the Viz Engine Administrator Guide
- Event Pool

Copyright © 2026 Vizrt

13.4.6 Image Channels



This section details how to create, assign and work with Image Channel media assets.

Image Channels are mainly used in conjunction with Viz Multiplay for creating video wall layouts. However, you can also use Image Channels to add images to a 2D environment in DVE mode. You can still add additional 3D objects to your scene tree, but the Image Channel always resides in the background of your scene unless you change the layer order via the DVE menu.

Note: Image Channels are designed to work only in DVE mode, not Texture mode.

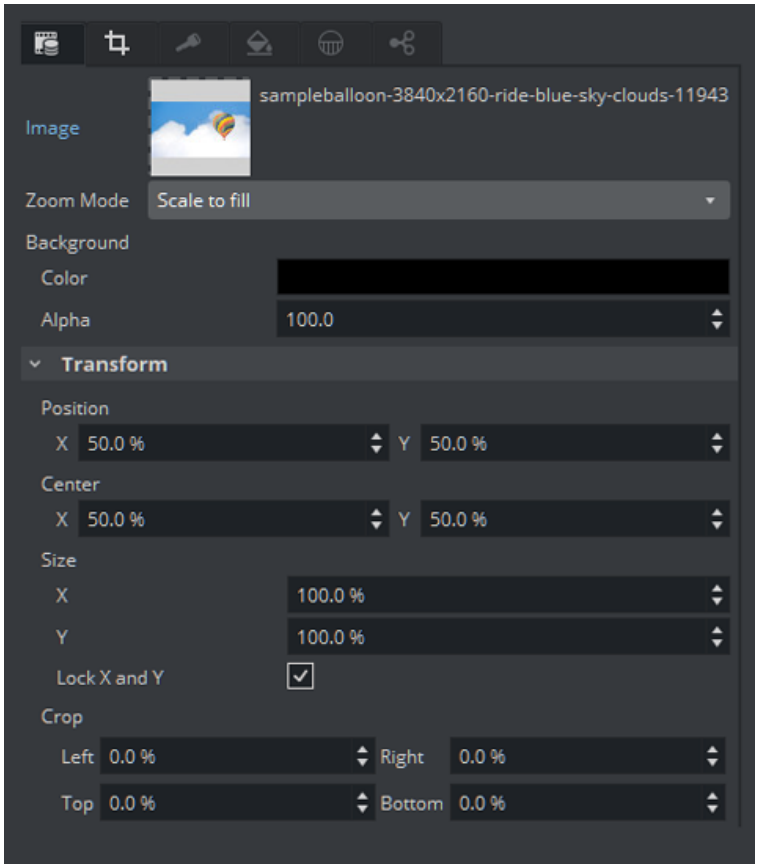
Some of the benefits of using Image Channels are:

- Eliminates the need to use special plugins to adjust image proportions.
- Shows assets only when they are fully loaded.
- Provides a notification as soon as assets are available.
- Allows you to design your own transition to use to switch between elements.

Image Channel Specific Settings

The following settings lets you to manipulate the content inside the Image Channel:

Copyright © 2026 Vizrt



- **Image:** Determines the image to use.
- **Zoom Modes**
  - **Original:** Displays the image in a 1:1 ratio (without any scaling transformation).
  - **Scale to fill:** Stretches the image to the image channel dimensions.
  - **Scale to fit max:** Stretches the image to fit the dimensions of the image channel keeping the maximum aspect ratio.
  - **Scale to fit min:** Stretches the image to fit the dimensions of the image channel keeping the minimum aspect ratio.
- **Border Color:** Fills any other pixel not taken by the image with a given color and transparency.
- **Alpha:** Sets the transparency factor of the border.

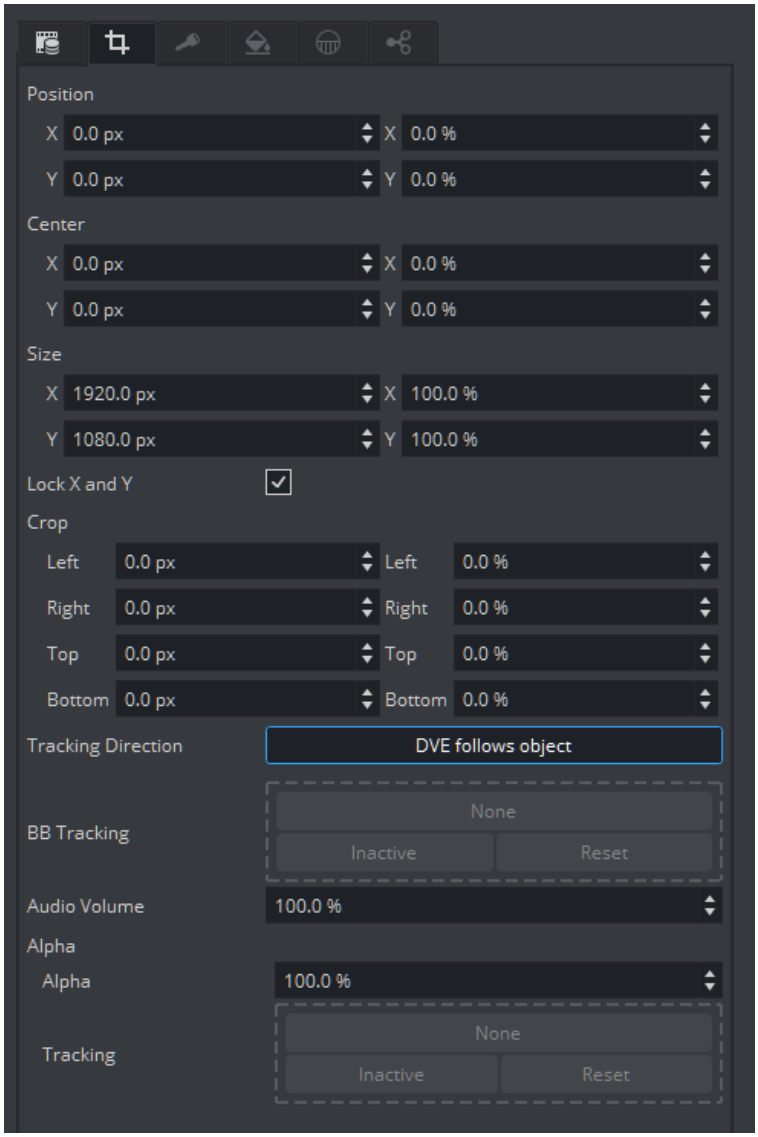
Software Composition

Content is composed with either **Alpha Blending** or a **premultiplied with the alpha** component. This can be set for the RGB (color) and the Alpha part (Key) of the Media Asset. Changing the composition may be required for correct blending with third party applications.

Image Channel Transformation Settings

The Image Channel media asset settings are available in the Media Asset Panel. Here, you can manipulate transformation of the currently selected channel.

Copyright © 2026 Vizrt



- **Tracking Direction**
  - **Object follows DVE:** Applies any change to the bounding box to the dimensions of the image channel immediately for any container assigned this property. This property is only available if **Basic** mode is selected.
  - **DVE follows object:** Allows creation of a videowall in a virtual set which can still be controlled by Viz Multiplay. In this mode, the 2D DVE effects are forwarded to a container.
- **Alpha:** Defines the alpha value of the image channel. Optionally, you can drag a reference container to this property, every change in the alpha value of the reference container is applied to the image channel.
- **Tracking:** Tracks the Alpha value of a container.

Copyright © 2026 Vizrt

Enforced Background Loading

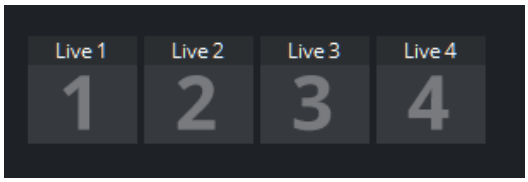
Similar to generic images in the scene tree, images for image channels can be loaded using background loading even if background loading is disabled in the config. To enforce background loading, use the following command when loading an image:

Viz Engine Commands

```
$(scene)*IMAGECHANNEL*$(id)*IMAGE SETBGL $(image_path_or_uuid_or_url)
example:
MAIN_SCENE*IMAGECHANNEL*2*IMAGE SETBGL <31E74EAF-E4B1-254C-B50987E7DF19182F>
```

Copyright © 2026 Vizrt

13.4.7 Live Video Media Asset



Live Media Assets integrate live video sources like SDI, IP or NDI sources either as DVE or Texture Effect. You can see a detailed description of the functionality in the Media Asset Panel.

To better differentiate which type of Input has been configured, Live Input types have a color schema:

Type
SDI Live Input
2110 Live Input
NDI Live input
SMURF Shared Memory Input
Stream Live Input

A live feed can be can be applied through either DVE, Texture or Hybrid mode. If alpha is required for the external video, open **Config** and enable alpha in **Video Input** (see the **Configuring Viz** section of the Viz Engine Administrator Guide).

- Specifically for Matrox SDI cards: Enable **w/Alpha** when you require external video with alpha. The Matrox board combines two of the video inputs in pairs, to serve both fill and key. Inputs A and B, and input C and D are combined, respectively.

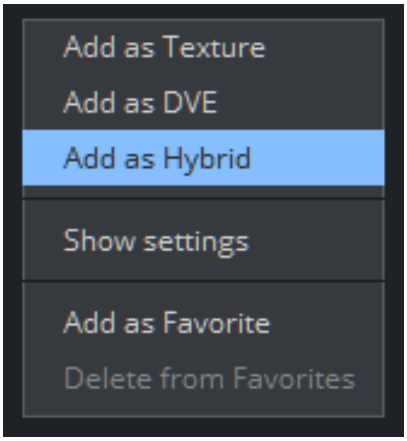
Live Feeds as Hybrid Mode

Live Inputs can be used as DVE and Texture at the same time for Viz Engine versions 3.12 and later.

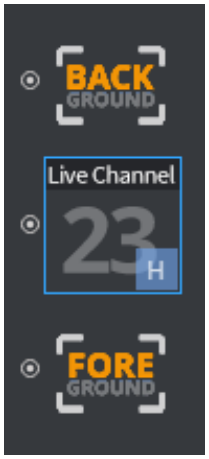
To Add a Live Feed in Hybrid Mode

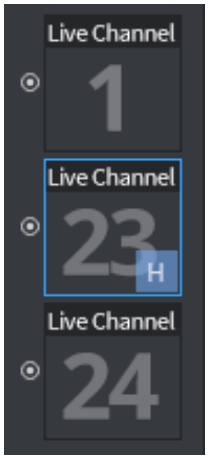
1. Click **Built Ins > Live**.
2. Right click on any of the live inputs.
3. Select **Add as Hybrid**.

Copyright © 2026 Vizrt



Hybrid Assets are indicated with a small capital H:





Copyright © 2026 Vizrt

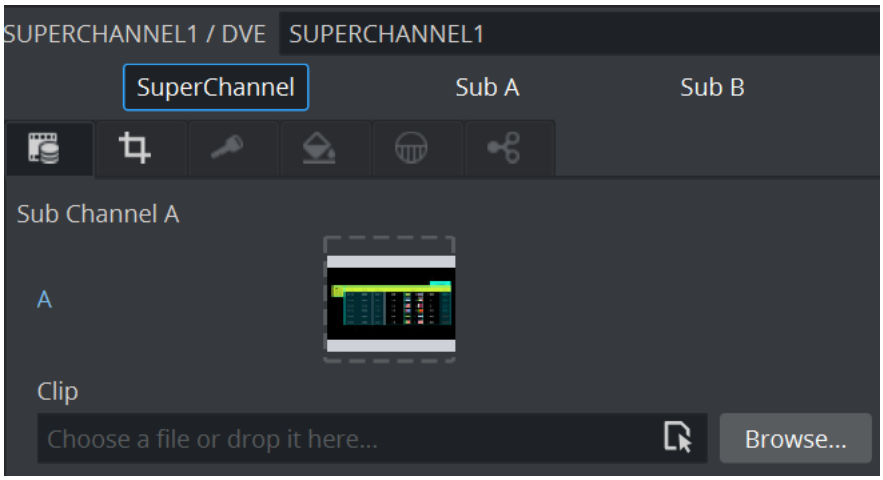
13.4.8 Super Channels

- Super Channel and Subchannels
- Super Channel Settings
- Super Channel Transformation
- Sub Channels
    - Transformation Settings
    - Start Mode Settings
    - Scene Settings
    - Image Settings
    - Clip Settings
- Clip Channels Used
- Transition Shaders
- Super Channel Events
- Enforced Background Loading
    - Viz Engine Commands
- Adaptive Scene Design Formats Support for Super Channels
    - Set the Requested Format for SubChannel A or B

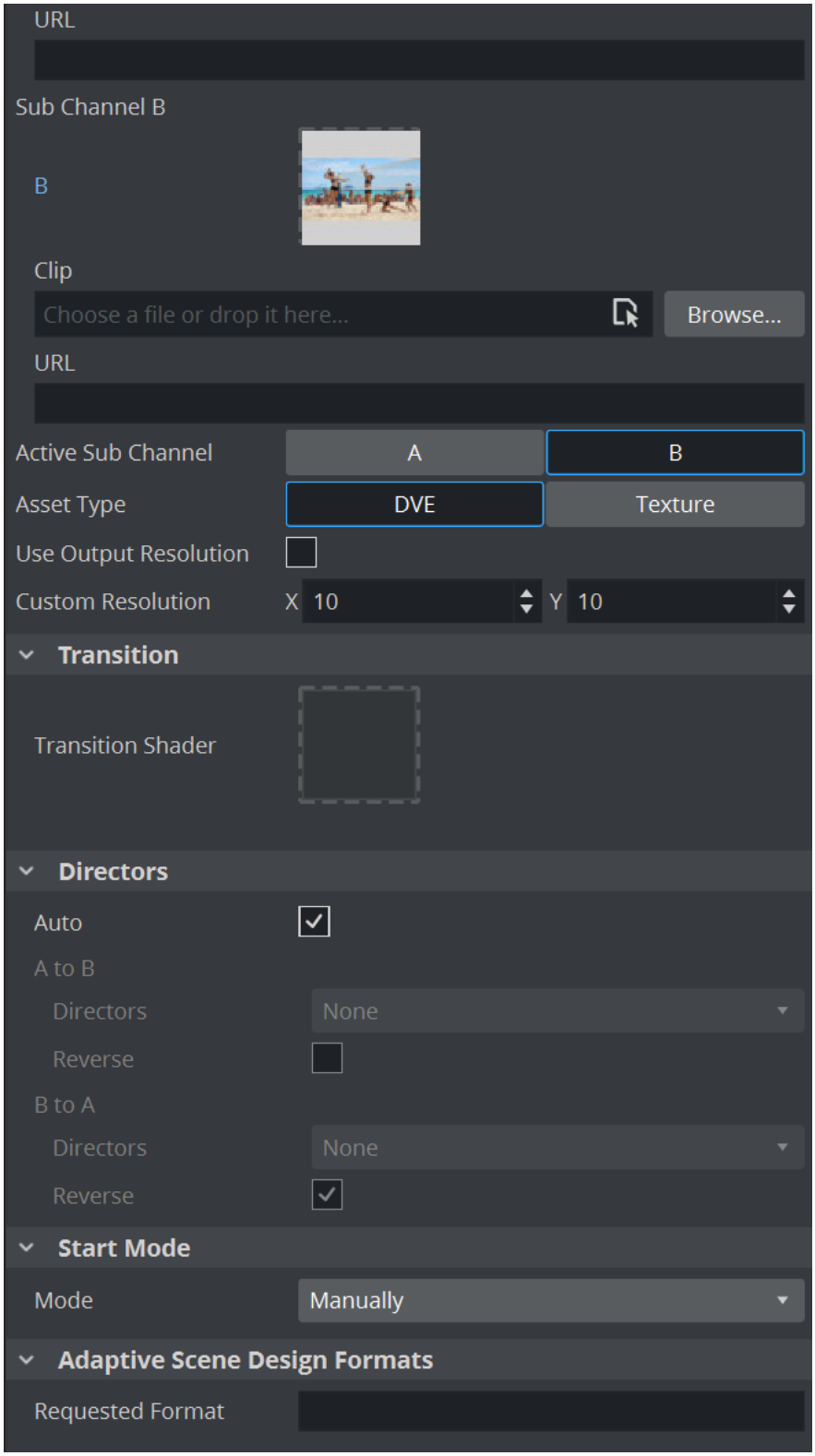


Super Channels provide a generic tool to create and group transitions between various types of assets in Viz Artist. They are an abstraction mechanism to represent, access and apply transitions between the various types of media assets.

Within the Super Channel, you assign two media assets, A and B, to provide event based transitions between them. Internally, the Super Channel then utilizes the channel type required of the corresponding media asset to represent the underlying content.



Copyright © 2026 Vizrt



Some of the benefits of using Super Channels are:

- Replaces Transition Logic.

Copyright © 2026 Vizrt

- Eliminates the need to use special plugins to adjust image proportions.
- Shows assets only when they are fully loaded.
- Provides a notification as soon as assets are available.
- Allows you to design your own transition to use to switch between elements.

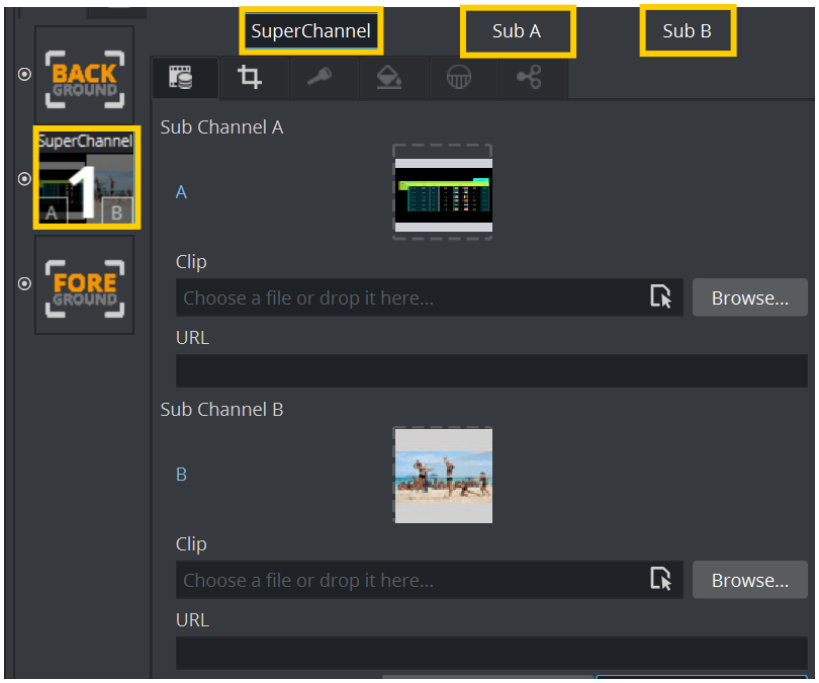
Here, you find information on the following:

- Super Channel and Subchannels
- Super Channel Settings
- Super Channel Transformation
- Sub Channels
    - Transformation Settings
    - Start Mode Settings
    - Scene Settings
    - Image Settings
    - Clip Settings
- Clip Channels Used
- Transition Shaders
- Super Channel Events
- Enforced Background Loading
    - Viz Engine Commands
- Adaptive Scene Design Formats Support for Super Channels
    - Set the Requested Format for SubChannel A or B

Super Channel and Subchannels

A Super Channel utilizes some generic settings and settings for A and B channel. The icon in the side panel shows a preview of the two Subchannels. To switch between the different properties you can use the tabs on the top or you can click the icons to switch:

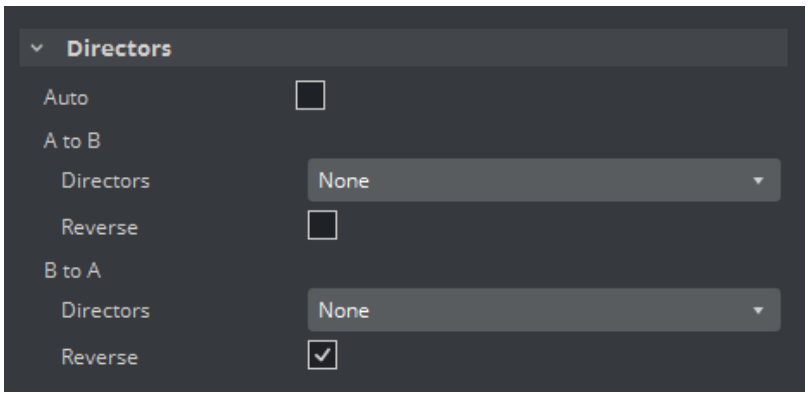
Copyright © 2026 Vizrt



Super Channel Settings

Sub Channel (A, B): Drag scenes, images, live feeds or clips to the **Sub Channel** icon.

To assign a clip from Viz One, drag it from the Viz Onebrowser to the clip entry.



Active Sub Channel

Forces the Super Channel to switch between A and B.

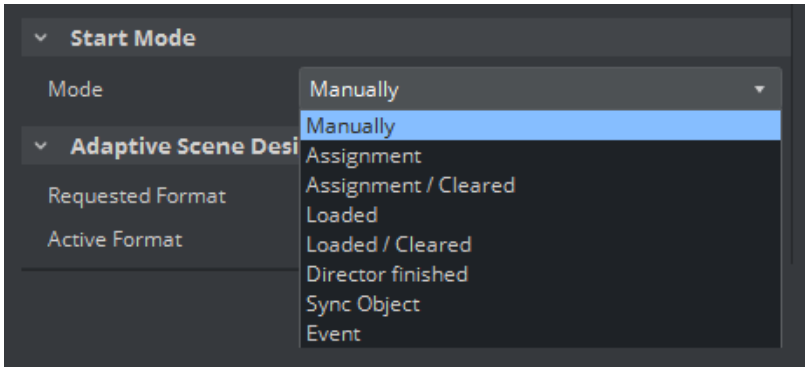
- **Asset Type:** Switches between DVE and Texture mode. DVE requires a videoboard with hardware compositor capabilities.

Copyright © 2026 Vizrt

- You can override the default output resolution (*Use Output Resolution*) with a custom resolution if needed (useful in certain scenarios like videowalls setups or if the surface of the texture channel is needed in third party applications).

> This will affect scenes only!

- **Directors:** Sets the animation of the transition, from sub channel **A to B**, or from **B to A**. Disable `Auto` to select an existing Director to run the transition with, or `None` to disable the transition.



Start Mode: Selects what triggers the Super Channel transition:

- **Manually:** Sets user manual switching from sub channel A to B, or vice versa, by selecting the sub channel in Viz Artist or through external commands.
- **Assignment:** Triggers transition when you assign a new asset to inactive sub channel.
- **Assignment / Cleared:** Triggers transition asset assignment, or when you clear the sub channel.
- **Loaded:** Triggers transition when an asset is loaded. This is useful for Super Channels that display images when image background loading is active in the Viz Engine configuration.
- **Loaded / Cleared:** Triggers transition when an asset is loaded, or when you clear the sub channel.
- **Director finished:** Triggers transition once a given director has finished execution.
- **Sync Object:** Select a Super Channel from the scene in the **Sync objects** area. Sync objects provide a way to synchronize execution of Super Channel transitions between multiple Super Channels. Sync objects monitor all linked Super Channels until all their inactive sub channels have been loaded. The sync object then calls `onAllChannelsArmed` to trigger the transition for all Super Channels managed by the sync object. You can add as many Super Channels to a sync object as are required.
- **Event:** Allows you specify the Scene, the sender (Animation from Stage or from Super Channel) and the Event (on Started, on Stopped, onDirectorStarted, onDirectorStopped).
- **Mode:** Selects the mode for using internal channels. This only affects hardware based channels (for example, Live1 or Clip1). In **DVE** mode, the internally used channel for a subchannel is used in DVE mode, all properties of the subchannel are copied to the DVE layer. In **Texture** mode, the internally used channel is used in texture mode and an ImageChannel is used to show the texture.

Copyright © 2026 Vizrt

Sync Object Example: Add two Super Channels to the scene. Then set **Start Mode** of Super Channel 1 to **Sync Object** and drag Super Channel 2 into the list of linked channels. Notice how this links Super Channel 1 and Super Channel 2 to each other. The transition from the active to the inactive **Sub Channel** starts as soon as both inactive Sub Channels have fired **onAssigned** and **onLoaded**. Assign an image to sub channel **B** of both Super Channels to achieve this. You can do this manually in Viz Artist, or via external commands:

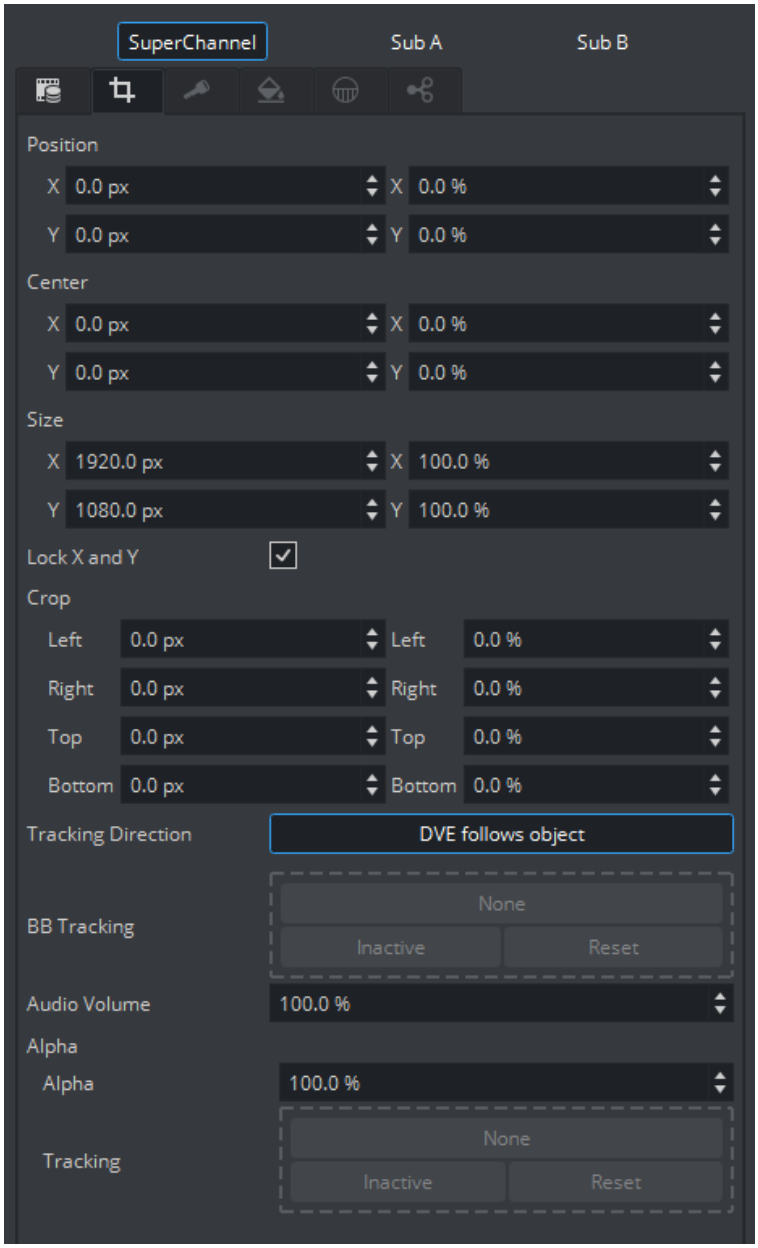
```
MAIN_SCENE*Super Channel*<CHANNEL_ID>*<A|B> SET IMAGE <UUID>
```

In case a sync object is active, but you do not want to change the assets on **allsyncedSuper Channels**, you still have to call the assign command on all Sub Channels. Otherwise, the sync object does not fire, and thus the transition does not trigger. However, re-assigning the already assigned Media Asset is completely free in terms of performance, as no internal reloading takes place. As such, this represents a viable option to trigger the `requiredonAssignedevent`.

Super Channel Transformation

This section is used to animate the whole Super Channel, including both sub channels. Super Channel animations are independent of the sub channel animations. This means that you can have a transition animation between multiple Super Channels, which is often required for content on Videowalls.

Copyright © 2026 Vizrt



You can animate certain Super Channel properties, such as **Position**, **Size**, or **Alpha**, to define custom transitions between one or more Super Channels. A detailed description of the parameters can be found in the Media Asset Panel.

Note: Super Channels deal with rendering content assigned to the two sub channels, A and B, in the current main scene. Thus, they represent an intuitive and easy to use mechanism for transitioning between the sub-channels A and B.

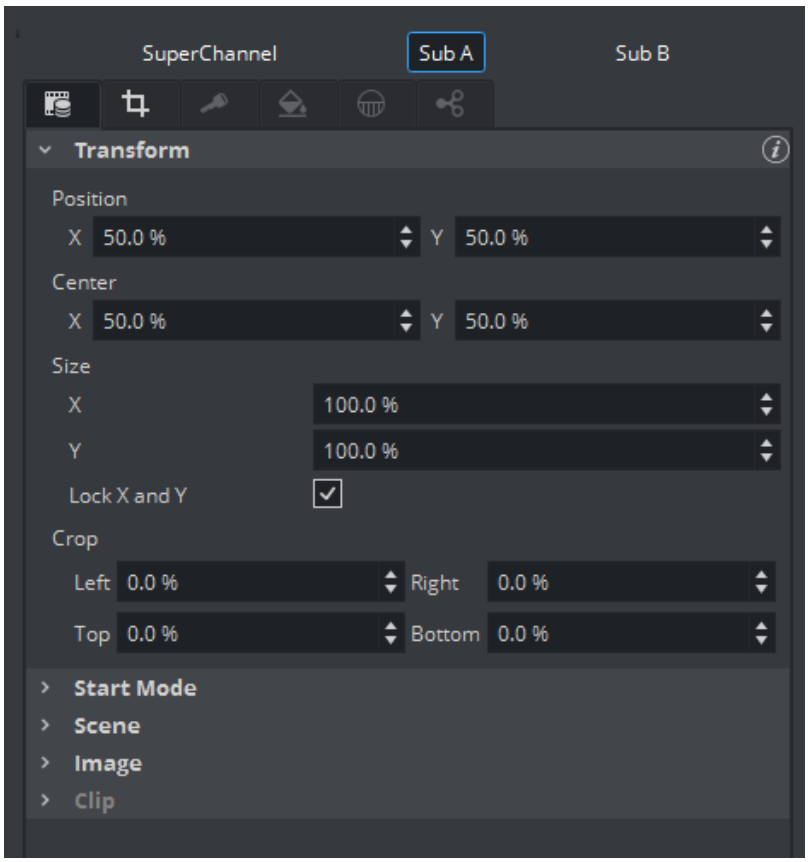
Sub Channels

A Super Channel is composed of two Sub Channels, A and B.

Copyright © 2026 Vizrt

You can animate certain Sub Channel properties, such as **Position**, **Size**, or **Alpha**, to define custom transitions between the two Sub Channels.

Transformation Settings

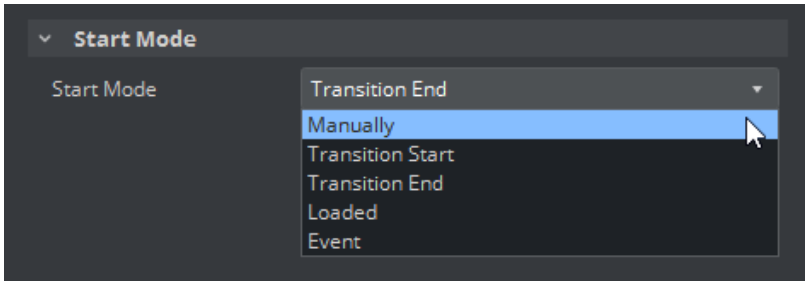


A detailed description of the parameters can be found in the Media Asset Panel.

Note: To utilize background loading in combination with Super Channels, enable it globally in Viz Engine. The dedicated `SETBGL` commands used with containers, `<ObjectID>*TEXTURE*IMAGE SETBGL <UUID>`, do not work with Super Channels.

Copyright © 2026 Vizrt

Start Mode Settings



Here you can select how to trigger the Sub Channel animation. The **Start Mode** options are:

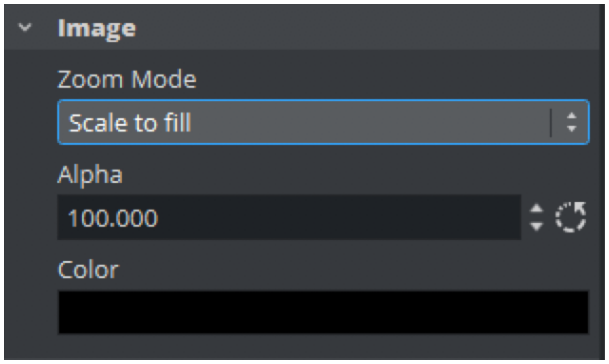
- **Manually:** Sets user manual switching from subchannel A to B or vice versa by clicking the corresponding subchannel in Artist or through external commands.
- **Transition Start:** Triggers on transition start.
- **Transition End:** Triggers on transition end.
- **Loaded:** Triggers transition once a newly assigned asset has been loaded. This is of special interest for Super Channels displaying images, with image background loading activated in the Viz Engine configuration.
- **Event:** Allows you specify the Scene, the sender (Animation from Stage or from Super Channel) and the Event (on Started, on Stopped, onDirectorStarted, onDirectorStopped). An event is fully defined by its event name, the identifier/name of the sender, and its arguments. A new internal event system was introduced in Viz Engine 3.9. This allows certain objects to fire and listen to internal events. Amongst those objects are:
  - **Super Channels or their sub channels**
  - **Stage**
  - **Scripts**
- For details, please see Super Channel Events.

Scene Settings

- **Scene Force Key:** Allows you to force a full-screen key of your loaded scene, even there is no key function applied to it.
- **Keep Aspect:** When using orthogonal camera projection, this flag ensures that the output is not going to be distorted if the main scene has a different aspect.

Copyright © 2026 Vizrt

Image Settings



- **Image:** Allows you to control the Zoom mode, Alpha value and the border color of a loaded Image.

Clip Settings

Provides control of a loaded clip.

Clip Channels Used

Subchannels use image/clip/live/gfx channels internally to display their contents. When a clip is selected for a subchannel, the first un-used clip channel is chosen by default to display the clip. However, if the subchannel property *ASSIGNED_CLIPCHANNEL* is set (using command `SET`), the subchannel is enforced to use this specifically assigned clip channel.

Also, the property *RESERVED_CLIPCHANNELS* of the scene saves a list of clip channels, which the subchannels of the scene should avoid. Clip channels can be added to (command `ADD`) or removed from (command `REMOVE`) the list. The whole list can also be set (command `SET`).

Assigning/Restricting Clip Channels (Command Line):

```
MAIN_SCENE*SUPERCHANNEL*1*A*ASSIGNED_CLIPCHANNEL SET Clip1
MAIN_SCENE*SUPERCHANNEL*1*A*ASSIGNED_CLIPCHANNEL GET
MAIN_SCENE*RESERVED_CLIPCHANNELS ADD Clip1
MAIN_SCENE*RESERVED_CLIPCHANNELS REMOVE Clip1
MAIN_SCENE*RESERVED_CLIPCHANNELS SET Clip1 Clip1 Clip3
MAIN_SCENE*RESERVED_CLIPCHANNELS GET
```

Viz script also has commands for the same functionality, as in the following examples:

Assigning/Restricting Clip Channels (Viz Script):

```
scene.getSuperchannel(0).Pending.AssignedClipChannel = "CLIP1"
println(scene.getSuperchannel(0).Pending.AssignedClipChannel)
```

Copyright © 2026 Vizrt

```
scene.AddReservedClipChannel("CLIP1")
scene.RemoveReservedClipChannel("CLIP1")
scene.SetReservedClipChannels("CLIP1 CLIP2 CLIP3")
println(scene.GetReservedClipChannels())
```

Transition Shaders

How to create and work with Super Channel Transition Shaders is explained in chapter Viz Engine Shaders.

Super Channel Events

Event Example:
An event is fully defined by its Eventname, the identifier/name of the Sender and its Arguments.
Eventname: `onDirectorFinished`
Sender: `#1004*STAGE`
Arguments: `director1`

Query available/registered events using the following commands:
`EVENT_POOL*REGISTERED_OBJECTS GET`
`EVENT_POOL*REGISTERED_EVENTS GET <$objectname>`
Example
`EVENT_POOL*REGISTERED_EVENTS GET #427*STAGE`

With Viz Engine 4.2, the behavior once a subchannel gets cleared has been changed:

Viz Engine versions before 4.2:
If B is the active subchannel and A gets cleared, it switches to A.
If B is the active subchannel and B gets cleared, nothing happens.

Viz Engine versions 4.2 and later:

If B is the active subchannel and A gets cleared, it switches to A.
If B is the active subchannel and B gets cleared, it switches to A.

Note: To obtain detailed information about status changes on the various Super Channels and subchannels enter command: `EVENT_POOL*DEBUGGING SET 1`. Additional status information related to the Super Channels and their Sub Channels can be found in the Viz Engine's console output.

Enforced Background Loading

Similar to generic images in the scene tree, images for Super Channels can be loaded using background loading even if background loading is disabled in the config. To enforce background loading, use the following command when loading an image:

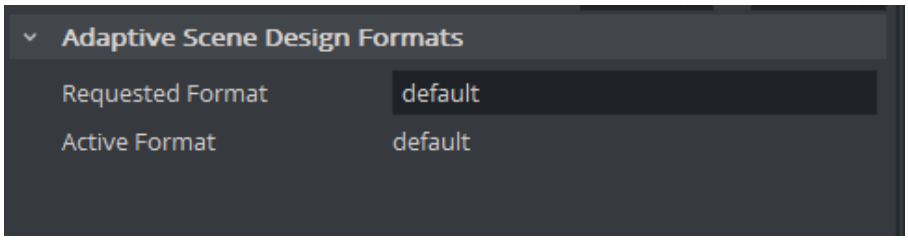
Copyright © 2026 Vizrt

Viz Engine Commands

```
$(scene)*SUPERCHANNEL*$(id)*$(SUBCHANNEL) SETBGL IMAGE $(image_path_or_uuid_or_url)
example:
MAIN_SCENE*SUPERCHANNEL*2*A SETBGL IMAGE <31E74EAF-E4B1-254C-B50987E7DF19182F>
```

Adaptive Scene Design Formats Support for Super Channels

Super Channels can be set to render a certain format using the command or Viz Artist:




```
#<sceneID>*SUPERCHANNEL*1*REQUESTED_FORMAT SET <name of format>
```

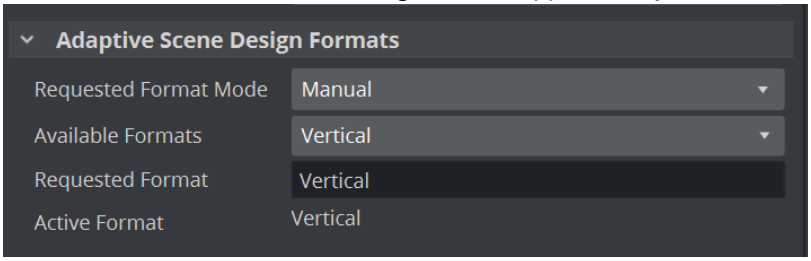
Note: If the requested format is not supported by the Super Channel scene, then it falls back to the default format to render content.

Note: Formats support is possible if we are using a scene in SubChannel A or B.

Set the Requested Format for SubChannel A or B

There are several options available for Requested Format Mode in SubChannels:

- **Manual:** Renders a given format. Can be set using **Available Formats**.
- **Inherit from Main Scene:** Uses the format of the GFX Channel Master scene.
- **Use Aspect Ratio:** Selects the format automatically based on the width/height, and therefore, the aspect ratio of the scene. The first matching format is applied. Only works for DVE.




```
#<sceneID>*SUPERCHANNEL*1*A*SCENE*REQUESTED_FORMAT SET <name of format>
```

Copyright © 2026 Vizrt

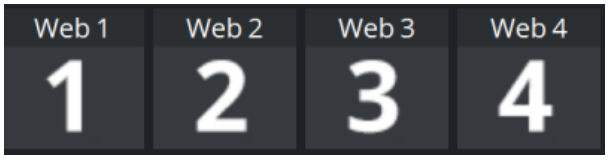
```
#<sceneID>*SUPERCHANNEL*1*A*SCENE*REQUESTED_FORMAT_MODE SET <mode> // options
are MANUAL, INHERIT_FROM_MAIN_SCENE or USE_ASPECT_RATI
```

See Also

- Event Pool

13.4.9 Web Channels

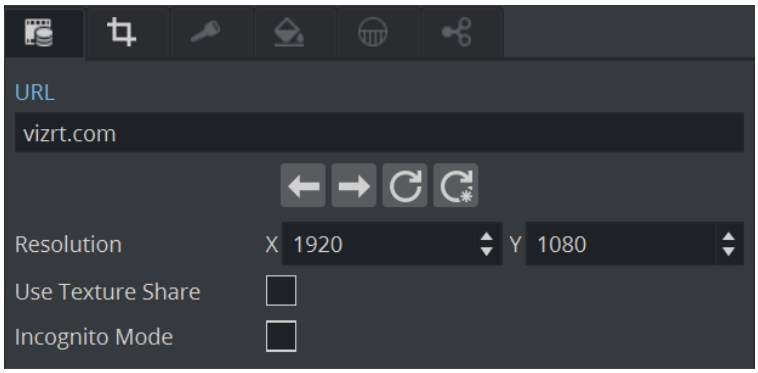
This section details how to create, assign and work with Web Channel media assets.



Render Layers

Web Channels allow a designer to use up to eight HTML-based render layers. They can be used either as *Texture* or as *DVE* effect.

> **Note:** Web Channels are only available in Viz Engine Render Pipeline.



The navigation keys can be used to navigate within the Web Channel history. The shown Web Channel page can be refreshed by commands or buttons in the User Interface, If you want to do a “hard” reload, use the *Ignore Cache* reload function.

On single GPU instances, you can turn on *Use Texture Share* to directly access the texture of the Web Channel. If you want to start your browsing session in a clean environment, use *Incognito Mode*.

Requirements

Web Channels only require a valid URL. Once entered, the webpage is rendered and can be interactively used in DVE mode.

> **Note:** The rendering of web content is synchronized with Viz Engine output refresh rate.

> **Information:** Keyboard/mouse interaction is supported only in *DVE* mode.

Copyright © 2026 Vizrt

Blending

If you want to blend your webpage against any background in DVE mode, make sure your are using the following order:



Audio

Audio from Webpages is supported and mixed into Viz Engine Audio system. The Volume of Web Channels can be controlled in the Media Asset Transformation Tab.

Copyright © 2026 Vizrt

13.5 Playback of Media Assets

The playback of Media Assets depends on the machine and video board configuration. There are two Media Asset playback methods:

- Video Clip Playback
- Live Media Assets

Media Assets are applied to a Scene either as DVE or as a Texture.

Video Input settings must be configured in Viz Configuration (see the Viz Engine Administrator Guide). These settings determine the frame rate (PAL/NTSC) of the whole video system.

Video clips transferred from Viz One can be played if the clip's container and codec combination is supported by the Matrox board. Any other format should be played by using the VML player.

Note: Please refer to the Viz Engine Administrator Guide for supported video boards.

IMPORTANT! The most basic rule for working with Viz Engine is to always have a valid genlock attached to the video board, no matter which board is installed. Without the correct genlock setup there is no guarantee that the Engine functions correctly (see **Viz Configuration > Video Output** in the Viz Engine Administrator Guide).

Media Asset parameters are modified in the Media Asset Panel Tab in the Scene Settings. To control Media Assets from an external control application, see the **Viz Command Interface Documentation** located in the <*Viz Installation Folder*>.

Media Assets are applied in Viz Artist either as Digital Video Effect (DVE) or as a Texture:

13.5.1 DVE

The playback of external video through **DVE** does not have any impact on rendering performance, since it is handled solely on the board. When DVE is used, the graphics scene and the channel are composited on the video card (Matrox X.mio) and an SDI monitor is needed to show the video channel. There is also a limited set of 2D control options available in the design and animation of the Viz Artist scene (position, size, crop and alpha).

Chroma keying is not supported for channels set to DVE. The **DVE Alpha** setting for **GFX Channels** allows Media Assets as DVE to be blended, however, setting alpha for nested DVEs is currently not supported.

13.5.2 Texture

The playback of external video through **Texture** has an impact on rendering performance, since the live video needs to be transferred to the graphics board. When a video is applied as a Texture, the video is part of the graphics.

When more control of a video is needed, for example, to map the video onto a 3D object in a Scene, then Texture mode can be used. The video would then have the same control parameters as an image in Viz Artist. However, the video may now affect rendering performance, since it needs to composite the video into the render window.

Copyright © 2026 Vizrt

13.6 Keying Mode

There are different types of Keying modes built into Viz Artist, which can be used with Media Assets applied as Texture.

13.6.1 Linear Key

Linear Key is the standard keying technique, and can be used for most applications. This is a logical key (enabled by the Key plug-in):

- If a video or clip channel is applied to an object, it is treated the same way as if it would be a static image.
- If it is used as a background image, it is the first item to be drawn and all objects in the scene are drawn on top of it regardless of the key settings in the global section of scene settings.
- If it is set as the foreground image, it is the last item to be drawn and all other objects are only visible in the parts of the screen where the key/alpha of the video is not 100% (opaque)

13.6.2 Multi-Zone Key

Multi-Zone Key is often used in outdoor sporting events broadcasts.

The Multi-Zone Keyer prepares the input video to be keyable by the Chroma Keyer plug-ins in the scene tree. It is typically used in sports productions where more than one color needs to be keyed out. The video source needs to be dragged into the background image in global settings. An unlimited number of Chroma Keyer plug-ins can be created in the scene, each keying out a different color.

13.6.3 External Key

Choose this option if an external keying interface (for example, for Viz Eclipse) is going to be used.

The External Keying interface aims to connect the Viz Engine to the external keyer process via a SHM interface using AVVCdata packages. Additionally, Viz Engine should not be blocked or hindered in any way from doing its work if the external process is either not yet existing, not performing fast enough, or in the event of a crash (or even intended shutdown or restart) of the external process.

13.6.4 External Processed Foreground

Reads processed foreground and matte from an external signal and applies the postprocessing steps of the Precision Keyer. This is supported in the Virtual Studio and Multilayer Virtual Studio sequences.

Depending on the input signal, **External Signal is Linear** needs to be enabled to get a proper blend result around details.

Linear external signal OFF

Copyright © 2026 Vizrt



Linear external signal ON



13.6.5 Precision Keyer

The Precision Keyer. This option provides the best quality. We recommend using an application like Viz Arc to control the Precision Keyer.

13.6.6 Keying Best Practices

This section contains information on some best practices when Keying.

- **Take advantage of the self-configuration:** Before you start to play with the different parameters, let the system configure by itself. Select a section of the image, which contains in best case all color tones, you want to key. If you have no foreground objects, you can set the whole input video as section for the self-configuration.
- **Modify the keying color:** Put your real foreground objects into the Scene. Mostly your objects still have a blue edge after the self-configuration. To eliminate it, increase the Outer and Inner UV-Radius. When increasing the Inner UV-Radius have a look at opaque foreground objects in the video. It is perfectly configured if it cuts off also a little bit of the object. Have a look at tiny things, like hair or transparent objects and modify the Outer UV-Radius to get a smooth fade out at the edges of these objects.
- **Desaturate the blue/green edges:** Up till now you have probably had tiny edges around people or objects which are close to the background color. If you enable color correction by setting it to blue or green, the color is replaced.
- **Add highlights and shadows:** By modifying the add highlights and add shadows, borders of objects show, that you have keyed before by setting the inner and outer radius.
- **Play around with it:** You can either try to improve the keyer by increasing or decreasing the values, or you just play around with it.

Copyright © 2026 Vizrt

13.6.7 To Apply Keying Parameters via SHM and Scripting

Parameters for the Chroma Keyer (including the Precision Keyer) can be applied by using shared memory and Viz Engines internal scripting language. No need to send a lot of commands to the Viz Engine. And in additional, shared memory values can be stored on Graphic Hub.

Sample Script

The following sample explains how to set and apply Chroma Keyer parameters:

```
sub OnInit()
    dim key as string = "preset1"

    'fill shm map with keyer test parameters
    system.map[key] = 1 'define base key
    system.map[key & ".KeyHueAdjust"] = 0.2
    system.map[key & ".KeySaturationAdjust"] = 0.21
    system.map[key & ".WhitePoint"] = 0.22
    system.map[key & ".BlackPoint"] = 0.23
    system.map[key & ".GrayPoint"] = 0.24
    system.map[key & ".ProcessFGGain"] = 0.25
    system.map[key & ".DenoiseRadius"] = 2
    system.map[key & ".DenoiseSharpen"] = 0.26
    system.map[key & ".BackingPlateRGB"] = cvertex(0.27, 0.28, 0.29)
    system.map[key & ".Shrink"] = 3
    system.map[key & ".EdgeWidth"] = 4
    system.map[key & ".EdgeBlur"] = 0.3
    system.map[key & ".EdgeLightWrapGain"] = 0.31
    system.map[key & ".SpillLightWrapGain"] = 0.32
    system.map[key & ".LightWrapRGB"] = cvertex(0.33, 0.34, 0.35)
    system.map[key & ".EdgeLightWrapRGBGain"] = 0.36
    system.map[key & ".SpillLightWrapRGBGain"] = 0.37
    system.map[key & ".LightWrapBGBlur"] = 0.38
    system.map[key & ".CyanGain"] = 0.39
    system.map[key & ".YellowGain"] = 0.40
    system.map[key & ".MagentaGain"] = 0.41
    system.map[key & ".AddShadows"] = false
    system.map[key & ".InnerShadows"] = 0.42
    system.map[key & ".OuterShadows"] = 0.43
    system.map[key & ".ShadowsGain"] = 0.44
    system.map[key & ".ShadowsOffset"] = 0.45
    system.map[key & ".AddHighlights"] = true
    system.map[key & ".InnerHighlights"] = 0.46
    system.map[key & ".OuterHighlights"] = 0.47
    system.map[key & ".HighlightsGain"] = 0.48
    system.map[key & ".HighlightsOffset"] = 0.49

    'apply keyer preset to video in 1
    System.SendCommand("#" & scene.vizid & "*VIDEO*VIDEOIN*1*KEYER_PARAMS_SHM SET " &
key)
```

Copyright © 2026 Vizrt

```
end sub
```

To Save and Load Shared Memory Keys from Graphic Hub

Shared Memory Maps can be saved and loaded from Graphic Hub.

- **Function LoadFromDb(path As String) As Boolean:** Loads the contents of the map from a database file. The file is specified by path, which must be an absolute path, such as */User/mem_map*.
- **Function LoadFromDbAsync(path As String) As Boolean:** Loads the contents of the map from a database file. The file is specified by path, which must be an absolute path, such as */User/mem_map*. This function is called asynchronous, the value does not have to be loaded when the function returns.

Copyright © 2026 Vizrt

13.7 Viz Artist Supported Codecs

This section details the supported Codecs for all supported Video Cards used by Vizrt.

13.7.1 Bluefish Codecs

Bluefish Video cards support only the **Microsoft DirectShow Filter Graph** framework to render video clips. Therefore, it depends on what codecs/DirectShow filters are installed on the machine.

See Also

- Video Clips

13.7.2 Supported Codecs for VML Clip Player

The following lists all container and codecs that have been tested and verified to work correctly using the VML player introduced in Viz Engine 4.3.

There is no perfect codec, therefore we can not recommend a particular one. Some codecs (for example AVC) are very CPU intense, whereas others result in large files and require a higher I/O bandwidth. Additional factors like framerate, bitrates, resolution also have a huge influence on playback performance. The following lists all containers and codecs that have been verified so far to be readable by the VML clip player.

> ⚠ **Note:** There is no guarantee that this list is complete. All Clips have been verified on common framerates and orientations.

- Supported Containers
  - MPEG-4 Part 14 (MP4)
  - MPEG 4 Video (M4V)
  - MPEG 2 Video (M2V, MPG, VOB) & Transport Stream (M2TS, TS)
  - Moving Picture Experts Group Phase 1 (MPEG)
  - Audio Video Interleave (AVI)
  - Apple QuickTime Movie (MOV)
  - Material Exchange Format (MXF)
  - Windows Media Video (WMV)
  - Advanced Systems Format (ASF)
  - Third Generation Partnership Project (3GP)
  - WEB Media (WEBM)
  - Matroska Multimedia Container (MKV)
- Clipplayer Pro License
- Known Limitations
  - Unsupported Containers
  - Unsupported Codecs and Known Incompatibilities

Supported Containers

MPEG-4 Part 14 (MP4)

Video Codec	Audio	Formats
H.265 (5)	AAC/AC3/MP3	

Copyright © 2026 Vizrt

Video Codec	Audio	Formats
H.264	AAC/AC3/MP3	**Profiles:** *Baseline@L1, L1.1, L1.2, L1.3* *Baseline@L2, L2.1, L2.2* (3) Baseline@L3, L3.1, L3.2 Baseline@L4, L4.1, L4.2 Baseline@L5, L5.1, L5.2 Baseline@L6, L6.1, L6.2 (4) *Main@L1, L1.1, L1.2, L1.3* *Main@L2, L2.1, L2.2* (3) Main@L3, L3.1, L3.2 Main@L4, L4.1, L4.2 Main@L5, L5.1, L5.2 Main@L6, L6.1, L6.2 (4) *High@L1, L1.1, L1.2, L1.3* *High@L2, L2.1, L2.2* (3) High@L3, L3.1, L3.2 High@L4, L4.1, L4.2 High@L5, L5.1, L5.2 High@L6, L6.1, L6.2 (4) *High10(1)@L1, L1.1, L1.2, L1.3* *High10(1)@L2, L2.1, L2.2* (3) High10(1)@L3, L3.1, L3.2 High10(1)@L4, L4.1, L4.2 High10(1)@L5, L5.1, L5.2 High10(1)@L6, L6.1, L6.2 (4)
MPEG-2	AAC/AC3/MP3	**Profiles:** Main@High
MPEG-4	AAC/AC3/MP3	**Profiles:** Visual Simple@L0, L5, L6
XDCAM EX	AAC/AC3/MP3	

(1) HDR playback is not supported yet

(2) Shows incorrect colors

(3) The level defines the maximum resolution and bitrate. Anything less than level 3 is not relevant for broadcast environments

(4) Known as CPU intensive

(5) Requires ClipPlayer Pro license

Copyright © 2026 Vizrt

MPEG 4 Video (M4V)

Video Codec	Audio	Formats
AVC	AAC LC	**Profiles:** High@L1, L1.2 High@L2.1 High@L3, L3.1 High@L4, L4.2 High@L5

MPEG 2 Video (M2V, MPG, VOB) & Transport Stream (M2TS, TS)

Video Codec	Audio	Formats
MPEG Video 2	Mpeg Audio , Layer2	**Profiles:** Main@Main Main@High, High 1440
MPEG Video 2 Transport Stream	Mpeg Audio , Layer2	**Profiles:** Main@Main Main@High 1440

Moving Picture Experts Group Phase 1 (MPEG)

Video Codec	Audio	Formats
MPEG Video 1 (1)	Mpeg Audio, Layer1	
(1) known as very low quality		

Audio Video Interleave (AVI)

Video Codec	Audio	Formats
Lagarith	PCM	supports Alpha
uncompressed	PCM	RGB +Alpha UYVY 422

Copyright © 2026 Vizrt

Video Codec	Audio	Formats
Newtek Codecs		Speed HQ 420 Speed HQ 4224 Alpha Speed HQ 4224 DCT Alpha Speed HQ 4444 (2) Speed HQ 4444 DCT (2)
DVCAM DV50 DVPAL DVCPro DVCPro100	PCM	**Profiles:** 420, 422 411
JPEG	PCM	
Intel IYUV	PCM	
Matrox Mpeg2 i-Frame Matrox Mpeg2 i-Frame HD	PCM	alpha not supported alpha not supported
Mpeg-4 Visual	MPEG Audio Layer 3	Simple@L1
(2) Shows incorrect colors		

Apple QuickTime Movie (MOV)

Video Codec	Audio	Formats
Animation (3)	PCM	
DV25 DV50 DVC Pro	PCM	
ProRes	PCM	422: HQ/LT/Proxy 4444:
DNxHD	PCM	SQ 8bit HQ 8bit HQX 10bit

Copyright © 2026 Vizrt

Video Codec	Audio	Formats
DNxHR	PCM	SQ 8bit HQ 8bit LB 8 bit HQX 10bit (1) 444 10bit (1)(2)
AVC	AAC/PCM	**Profiles:** Baseline@L4.1 High@L3, L3.2 High@L4, L4.1, L4.2
uncompressed (3)	PCM	RGB YUV 422 8bit
Apple JPEG	PCM	
HAP (4)	AAC/PCM (5)	HAP, HAP Alpha, HAP Q, HAP Q Alpha, HAP R

(1) HDR playback is not supported yet
(2) Shows incorrect colors
(3) Known as extremely I/O intensive
(4) Requires ClipPlayer Pro license
(5) HAP is a pure image compression codec. Audio is based on the stream inside the container.

IMPORTANT! The DNxHD and DNxHR codecs require a separate license, issued by Vizrt.

Material Exchange Format (MXF)

Video Codec	Audio	Formats
DV, DVCPro, DVCPro HD	PCM	
IMX	PCM	Class 30/40/50
DNxHD	PCM	
Matrox MPEG iFrame	PCM	alpha not supported
MPEG 2	PCM	**Profiles:** Main@Main Main@High, High 1440

Copyright © 2026 Vizrt

Video Codec	Audio	Formats
XDCam/XDCAM HD	PCM	Class 18/25/35 4:2:2 50
XDCam EX	PCM	Class 18/35
AVC Intra	PCM	Class 50/100/200
AVC Long GOP	PCM	**4:2:0** Class 6/Class 12/Class 25 **4:2:2** Class 25
XAVC Intra *(4)*	PCM	Class 100/Class 200
XAVC Intra 2K *(4)*	PCM	Class 100
XAVC Intra QFHD (4)	PCM	*Class 300/480*
XAVC Intra 4K (4)	PCM	*Class 300/480*
XAVC Long GOP *(4)*	PCM	25MBit/35MBit/50MBit
XAVC QFHD Long GOP (4)	PCM	*4:2:0* *4:2:2: 100MBit/140MBit/ 200Mbit*
(4) Known as extremely CPU intensive		

Windows Media Video (WMV)

Video Codec	Audio	Formats
Windows Media V7	WMA Pro	**Profiles:** V7 V8 V9, V9 Advanced, V9 Screen
MPEG-4 Visual (Microsoft)	n/a	

Copyright © 2026 Vizrt

Advanced Systems Format (ASF)

Video Codec	Audio	Formats
MPEG-4 Visual (Microsoft)	n/a	

Third Generation Partnership Project (3GP)

Video Codec	Audio	Formats
MPEG-4	AAC	**Profiles:** Visual Simple@L2, L6
AVC	n/a	**Profiles:** High@L4, L5.1

WEB Media (WEBM)

Video Codec	Audio	Formats
VP9	opus	
VP8	opus	
H.263		**Profiles:** Baseline@1.0
AVC	opus/aac	**Profiles:** Baseline@L1

Matroska Multimedia Container (MKV)

Video Codec	Audio	Formats
AVC	Vorbis, opus, aac	**Profiles:** High@L1.2 High@L2.1 High@L3, L3.1 High@L4 High@L5
AVC (H264)	AAC, opus, aac, eac3	High@L4

Copyright © 2026 Vizrt

Video Codec	Audio	Formats
MPEG-2	AAC	
MPEG-4 Visual	AAC	Simple@L1
Theora	AAC	
VP8	AAC	
VP9	AAC	

Clipplayer Pro License

The following codecs need to be licensed to be played:

- H.265
- HAP

Known Limitations

Unsupported Containers

The following containers are known to be not compatible with VML Clip Player:

Container	Additional Information
R3D	RED camera container
OGV	Ogg Video File
F4V	Flash Video
MJPEG	Motion JPEG
RM	Real Media Video
SWF	Small Web Format / Shockwave
WTV	Windows Recorded TV Show
DV	Raw DV stream (.dv file, no container). DV video plays in AVI, MOV or MXF.

Copyright © 2026 Vizrt

Container	Additional Information
OGG	Ogg Audio File (.ogg). Vorbis audio can't be seeked accurately, so audio can be missing or scratchy after a seek. Files with zero padding at the end don't open. Use WAV, or AAC in MP4.

Unsupported Codecs and Known Incompatibilities

The following codecs are known to be not compatible with VML Clip Player:

Container	Video Codec	Additional Information
AVI	Ross Expression	
AVI	huffyuv	
AVI	Cinepak	
AVI	FMP4	
AVI	M701	
AVI	xVid	
AVI	Microsoft RLE	
AVI	Microsoft Video1	
AVI	YUV 10bit V210	
MOV	AIVM	
MOV	ProRes 4444 XQ	
MOV	GoPro Cineform	
MOV	uncompressed YUV 422 10bit	V210
MP4	AV1	

Copyright © 2026 Vizrt

13.7.3 Advanced Issues with Video Codecs

Many codecs have specific limitations and only work on certain types of input. It is not possible to list all things to be aware of, but here are some:

- 24-bit uncompressed video is valid, but rare. Some programs may have difficulty in reading such files. Try creating an uncompressed video using RGBA to create a more common 32-bit uncompressed video.
- Not all codecs support both 24-bit and 32-bit video. You may need to change between RGB and RGBA to get a specific codec to work.
- Some codecs work with RGBA, but nevertheless store the video as 24-bit.
- If you plan to use an RGBA video in a program that can handle alpha, you need to make sure that you have KEYFunctions on containers you want to have visible. Otherwise the alpha channel is completely blank, thus making the whole video totally transparent.
- Some codecs require a handle to the top level application window. The Viz Artist plug-in interface is unable to supply such a handle. As a result the configuration, windows of some of the codecs fail to show. Also, some codecs crash unless they get a handle.
- Files created in raw uncompressed mode can display artifacts when played back in Windows Media Player. This is some oddity of the Media Player. The AVI video file contains pure RGB data, but the Media Player dithers it (often using only 16 colors) for some unknown reason. As said, the video file is correct and you should be able to import it into other video editing applications without any problems.

Copyright © 2026 Vizrt

13.7.4 Supported Codecs for Matrox Clip Playback

The Matrox X.mio series cards all support a selection of different codecs for both SD and HD. Certain codecs require certain card classes. Depending on the card class, a license upgrade may be required to extend the range of codecs supported by the installed video card. The Matrox video cards have a built-in license dongle used by the Matrox codecs to determine licensing rights.

- For the Matrox topology based video cards, the following classes apply:
  - **100 class:** No clip playback.
  - **500 class:** SD and HD clip playback.
  - **550 class:** SD and HD clip playback, including support for Apple ProRes codecs.
- For UHD playback a M264 board is highly recommended or mandatory for some codecs.

All codecs are implemented in the software.

QuickTime Codecs

Viz Engine uses Matrox hardware to utilize QuickTime codecs for clip playback and post-rendering with MatroxFileWriter and ClipOut channel.

The following QuickTime codecs are supported:

- RLE (animation): (Playback only)
- H264
- Apple ProRes
- HDV
- XDCam
- DVCPro
- DNxHD (requires Matrox Topology Video board)

IMPORTANT! The Codec DNxHD requires its own license, issued by Vizrt.

Matrox cards support the following codec types and formats:

- QuickTime Codecs
- PAL Codecs
  - MPEG2 IFrame
  - MPEG2 IBP
  - MJPEG Lossless
  - DVCAM
  - DVCPRO
  - XDCAM
  - H264
  - ProRes
  - Uncompressed
- NTSC Codecs
  - MPEG2 IFrame
  - MPEG2 IBP
  - MJPEG Lossless

Copyright © 2026 Vizrt

    - DVCAM
    - DVCPRO
    - XDCAM
    - H264
    - ProRes
    - Uncompressed
- 720p50 and 720p59.94 (60M) Codecs
    - MPEG2 IFrame
    - MPEG2 IBP
    - DVCPRO HD
    - HDV
    - XDCAM HD
    - DNxHD
    - AVC-Intra
    - H264
    - ProRes
    - Uncompressed
- 1080i25 and 1080i29.97 (30M) Codecs
    - MPEG2 IFrame
    - MPEG2 IBP
    - DVCPRO HD
    - HDV
    - XDCAM HD
    - DNxHD
    - AVC-Intra
    - H264
    - ProRes
    - Uncompressed
- 1080p50 and 1080p59.94 (60M) Codecs
    - MPEG2 IFrame
    - Matrox Uncompressed
    - H.264
    - DNxHD
    - ProRes
- 2160p UHDTV Codecs
    - MPEG2 IFrame
    - ProRes
    - Matrox Uncompressed
    - XAVC

Copyright © 2026 Vizrt

PAL Codecs

MPEG2 IFrame

Container	Codec	Features	Audio
.avi	MPEG2-IFrame422 720x576	Alpha: no Bitrate: 10-50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	MPEG2-IFrame4224 720x576	Alpha: yes Bitrate: 10-50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

MPEG2 IBP

Container	Codec	Features	Audio
.avi	MPEG2 IBP SD 4:2:0, itemary, Program, Transport720x576	Alpha: no Bitrate: 1-15 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	MPEG2 IBP SD 4:2:2, itemary, Program, Transport720x576	Alpha: no Bitrate: 5-50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mxf	MPEG2 IBP SD 4:2:0, itemary, Program, Transport720x576	Alpha: no Bitrate: 1-15 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf	MPEG2 IBP SD 4:2:2, itemary, Program, Transport720x576	Alpha: no Bitrate: 5-50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

MJPEG Lossless

Container	Codec	Features	Audio
.avi	MJPEG Lossless 720x576	Alpha: no Bitrate: 160 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

DVCAM

Container	Codec	Features	Audio
.avi	DV/DVCAM 4:2:0 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.dv	DVCAM 4:2:0 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz
.mov	DVCAM 4:2:0 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	DVCAM 4:2:0 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 4 ch: 16 in 16-bit; 48kHz

DVCPRO

Container	Codec	Features	Audio
.avi	DVCPRO 4:1:1 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	DVCPRO 50 4:2:2 720x576	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.dv	DVCPRO 4:1:1 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz
.dv	DVCPRO 50 4:2:2 720x576	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz
.mov	DVCPRO 4:1:1 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	DVCPRO 50 4:2:2 720x576	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	DVCPRO 4:1:1 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	DVCPRO 50 4:2:2 720x576	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mxf Panasonic P2 (OP-Atom)	DVCPRO 4:1:1 720x576	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio
.mxf Panasonic P2 (OP-Atom)	DVCPRO 50 4:2:2 720x576	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio

XDCAM

Container	Codec	Features	Audio
.mxf XDCAM (OP1a)	D10 (IMX) 4:2:2 720x608	Alpha: no Bitrate: 50 Bitdepth: 10 VBI: required Timecode: required TDIR: no	AES3 • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz

H264

Container	Codec	Features	Audio
.mov	H.264/AVC 720x576	Alpha: no Bitrate: VBR Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

ProRes

Container	Codec	Features	Audio
.mov	ProRes 422 720x576	Alpha: no Bitrate: 42 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	ProRes 422 HQ 720x576	Alpha: no Bitrate: 42 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Uncompressed

Container	Codec	Features	Audio
.avi	Uncompressed YUVU 4:2:2 720x576	Alpha: no Bitrate: 160 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	Uncompressed YUAVUA 4:2:2:4 720x576	Alpha: yes Bitrate: 240 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

NTSC Codecs

MPEG2 IFrame

Container	Codec	Features	Audio
.avi	MPEG2-IFrame422 720x480	Alpha: no Bitrate: 10-50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	MPEG2-IFrame4224 720x480	Alpha: yes Bitrate: 10-50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

MPEG2 IBP

Container	Codec	Features	Audio
.avi	MPEG2 IBP SD 4:2:0, itemary, Program, Transport 720x480	Alpha: no Bitrate: 1-15 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.avi	MPEG2 IBP SD 4:2:2, itemary, Program, Transport 720x480	Alpha: no Bitrate: 5-50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	MPEG2 IBP SD 4:2:0, itemary, Program, Transport 720x480	Alpha: no Bitrate: 1-15 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	MPEG2 IBP SD 4:2:2, itemary, Program, Transport 720x480	Alpha: no Bitrate: 5-50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

MJPEG Lossless

Container	Codec	Features	Audio
.avi	MJPEG Lossless 720x480	Alpha: no Bitrate: VBR Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

DVCAM

Container	Codec	Features	Audio
.avi	DV/DVCAM 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.dv	DV/DVCAM 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz
.mov	DVCAM 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kH • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	DVCAM 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 4 ch: 16 in 16-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mxf Panasonic P2 (OP-Atom)	DV/DVCAM 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio

DVCPRO

Container	Codec	Features	Audio
.avi	DVCPRO 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	DVCPRO 50 4:2:2 720x480	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.dv	DVCPRO 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz
.dv	DVCPRO 50 4:2:2 720x480	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mov	DVCPRO 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	DVCPRO 50 4:2:2 720x480	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf	DVCPRO 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf	DVCPRO 50 4:2:2 720x480	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf Panasonic P2 (OP-Atom)	DVCPRO 4:1:1 720x480	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio
.mxf Panasonic P2 (OP-Atom)	DVCPRO 50 4:2:2 720x480	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio

Copyright © 2026 Vizrt

XDCAM

Container	Codec	Features	Audio
.mxf XDCAM (OP1a)	D10 (IMX) 4:2:2 720x512	Alpha: no Bitrate: 30, 40, 50 Bitdepth: 10 VBI: required Timecode: required TDIR: no	AES3 • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz

H264

Container	Codec	Features	Audio
.mov	H.264/AVC720x480	Alpha: no Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

ProRes

Container	Codec	Features	Audio
.mov	ProRes 422 720x486	Alpha: no Bitrate: 42 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	ProRes 422 HQ720x486	Alpha: no Bitrate: 63 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Uncompressed

Container	Codec	Features	Audio
.avi	Uncompressed YUVU 4:2:2 720x480	Alpha: no Bitrate: 160 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	Uncompressed YUAVUA 4:2:2:4 720x480	Alpha: yes Bitrate: 240 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

720p50 and 720p59.94 (60M) Codecs

MPEG2 IFrame

Container	Codec	Features	Audio
.avi	MPEG2-IFrame422 1280x720	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.avi	MPEG2-IFrame4224 1280x720	Alpha: yes Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

MPEG2 IBP

Container	Codec	Features	Audio
.avi	MPEG2 IBP HD 4:2:0, itemary, Program, Transport1280x720	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	MPEG2 IBP HD 4:2:2, itemary, Program, Transport1280x720	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mxf (OP1a)	MPEG2, IBP HD 4:2:0, itemary, Program, Transport1280x720	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	MPEG2, IBP HD 4:2:2, itemary, Program, Transport1280x720	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

DVCPRO HD

Container	Codec	Features	Audio
.avi	DVCPRO HD1280x720	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.dv	DVCPRO HD1280x720	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mov	DVCPRO HD1280x720	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	DVCPRO HD1280x720	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf Panasonic P2 (OP-Atom)	DVCPRO HD1280x720	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio

HDV

Container	Codec	Features	Audio
.mov	HDV1280x720	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

XDCAM HD

Container	Codec	Features	Audio
.mov	XDCam EX 1280x720	Alpha: no Bitrate: 25 CBR, 35 VBR Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	XDCam HD 4221280x720	Alpha: no Bitrate: 18-35 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mp4	XDCam EX 1280x720	Alpha: no Bitrate: 25 CBR, 35 VBR Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf XDCAM (OP1a)	XDCAM HD, IBP HD 4:2:0,itemary, Program, Transport 1280x720	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz • 4 ch: 16 in 16-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mxf XDCAM (OP1a)	XDCAM HD422, IBP HD 4:2:2, 1280x720	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	PCM: • 8 ch: 24 in 32-bit; 48kHz

DNxHD

Container	Codec	Features	Audio
.mxf (OP-Atom)	DNxHD 145, 220, 220x 1280x720	Alpha: yes (except 220x) Bitrate: 100 Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio
.mxf (OP1a)	DNxHD 145, 220, 220x 1280x720	Alpha: yes (except 220x) Bitrate: 100 Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	DNxHD 4:2:2 1280x720	Alpha: no Bitrate: 36, 145, 220 Bitdepth: 8; 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

AVC-Intra

Container	Codec	Features	Audio
.mxf (OP1a)	AVCIntra Class 50 1280x720	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	AVCIntra Class 100 1280x720	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf Panasonic P2 (OP-Atom)	AVCIntra Class 50 1280x720	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio
.mxf Panasonic P2 (OP-Atom)	AVCIntra Class 100 1280x720	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio

Copyright © 2026 Vizrt

H264

Container	Codec	Features	Audio
.mov	H.264/AVC 1280x720	Alpha: no Bitrate: VBR Bitdepth: 8; 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mp4	XDCAM EX 1280x720	Alpha: no Bitrate: VBR Bitdepth: 8; 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

ProRes

Container	Codec	Features	Audio
.mov	I-Frame 4:2:2 1280x720	Alpha: no Bitrate: 100 (LT), 147, 220 (HQ) Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mov	I-Frame 4:4:4:4 1280x720	Alpha: yes Bitrate: VBR Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Uncompressed

Container	Codec	Features	Audio
.avi	Uncompressed YUVU 4:2:2 1280x720	Alpha: no Bitrate: 700 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	Uncompressed YUAVUA 4:2:2:4 1280x720	Alpha: yes Bitrate: 1050 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

1080i25 and 1080i29.97 (30M) Codecs

MPEG2 IFrame

Container	Codec	Features	Audio
.avi	MPEG2-IFrame422 1920x1080	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	MPEG2-IFrame4224 1920x1080	Alpha: yes Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

MPEG2 IBP

Container	Codec	Features	Audio
.avi	MPEG2 IBP HD 4:2:0, itemary, Program, Transport1920x1080	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.avi	MPEG2 IBP HD 4:2:2, itemary, Program, Transport1920x1080	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	MPEG2, IBP HD 4:2:0, itemary, Program, Transport1920x1080	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	MPEG2, IBP HD 4:2:2, itemary, Program, Transport1920x1080	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

DVCPRO HD

Container	Codec	Features	Audio
.avi	DVCPRO HD1920x1080	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.dv	DVCPRO HD1920x1080	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz
.mov	DVCPRO HD1440x1080	Alpha: no Bitrate: 100 Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP-Atom)	DVCPRO HD1440x1080	Alpha: no Bitrate: 100 Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: no	No Audio
.mxf (OP1a)	DVCPRO HD1440x1080	Alpha: no Bitrate: 100 Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

HDV

Container	Codec	Features	Audio
.mov	HDV1440x1080	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

XDCAM HD

Container	Codec	Features	Audio
.mov	XDCam EX1920x1080	Alpha: no Bitrate: 25 CBR, 35 VBR Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	XDCam HD1440x1080	Alpha: no Bitrate: 35 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mov	XDCam HD 4221920x1080	Alpha: no Bitrate: 18-35 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mp4	XDCam EX 1920x1080	Alpha: no Bitrate: 25CBR, 35VBR Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mp4	XDCAM EX1920x1080	Alpha: no Bitrate: VBR Bitdepth: 8; 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	XDCAM HD,IBP HD 4:2:0,itemary,Program,Transport 1920x1080	Alpha: no Bitrate: 25 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 16 in 16-bit; 48kHz • 4 ch: 16 in 16-bit; 48kHz

Copyright © 2026 Vizrt

DNxHD

Container	Codec	Features	Audio
.mxf (OP-Atom)	DNxHD 145, 220, 220x, 1920x1080	Alpha: yes (for 220) Bitrate: 100 Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio
.mxf (OP1a)	DNxHD 145, 220, 220x, 1920x1080	Alpha: yes (for 220) Bitrate: 100 Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

AVC-Intra

Container	Codec	Features	Audio
.mxf (OP-Atom)	AVCIntra Class 50 1440x1080	Alpha: no Bitrate: 50 Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: no	No Audio
.mxf (OP-Atom)	AVCIntra Class 100 1920x1080	Alpha: no Bitrate: 100 Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: no	No Audio

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.mxf (OP1a)	AVCIntra Class 50 1440x1080	Alpha: no Bitrate: 50 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mxf (OP1a)	AVCIntra Class 100 1920x1080	Alpha: no Bitrate: 100 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

H264

Container	Codec	Features	Audio
.mov	H.264/AVC 1920x1080	Alpha: no Bitrate: VBR Bitdepth: 8; 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

ProRes

Container	Codec	Features	Audio
.mov	I-Frame 4:2:2 1920x1080	Alpha: no Bitrate: 100 (LT), 147, 220 (HQ) Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	I-Frame 4:4:4:4 1920x1080	Alpha: yes Bitrate: VBR Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Uncompressed

Container	Codec	Features	Audio
.avi	Uncompressed YUVU 4:2:2 1920x1080	Alpha: no Bitrate: 800 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Container	Codec	Features	Audio
.avi	Uncompressed YUAVUA 4:2:2:4 1920x1080	Alpha: yes Bitrate: 1200 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

1080p50 and 1080p59.94 (60M) Codecs

MPEG2 IFrame

Container	Codec	Features	Audio
.avi	MPEG2-IFrame422 1920x1080	Alpha: no Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	MPEG2-IFrame4224 1920x1080	Alpha: yes Bitrate: 50-300 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Matrox Uncompressed

Container	Codec	Features	Audio
.avi	Uncompressed YUVU 4:2:2 1920x1080	Alpha: no Bitrate: 800 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	Uncompressed YUAVUA 4:2:2:4 1920x1080	Alpha: yes Bitrate: 1200 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

H.264

Container	Codec	Features	Audio
.mov	H.264/AVC 1920x1080	Alpha: no Bitrate: 50/100 Bitdepth: 8; 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

DNxHD

Container	Codec	Features	Audio
.mxf (OP-Atom)	DNxHD 145, 220, 220x, 1920x1080	Alpha: yes (for 220) Bitrate: 100 Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio
.mxf (OP1a)	DNxHD 145, 220, 220x, 1920x1080	Alpha: yes (for 220) Bitrate: 100 Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

ProRes

Container	Codec	Features	Audio
.mov	I-Frame 4:2:2 1920x1080	Alpha: no Bitrate: 147, 220 Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	I-Frame 4:4:4:4 1920x1080	Alpha: yes Bitrate: VBR Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

2160p UHDTV Codecs

MPEG2 IFrame

Container	Codec	Features	Audio
.avi	MPEG2-IFrame422 3840x2160	Alpha: no Bitrate: 900-1200 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.avi	MPEG2-IFrame4224 3840x2160	Alpha: yes Bitrate: 900-1200 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

ProRes

Container	Codec	Features	Audio
.mov	I-Frame 4:2:2 3840x2160	Alpha: no Bitrate: VBR Bitdepth: 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz
.mov	I-Frame 4:4:4:4 3840x2160	Alpha: yes Bitrate: VBR Bitdepth: 8, 10 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

Copyright © 2026 Vizrt

Matrox Uncompressed

Container	Codec	Features	Audio
.avi	Uncompressed YUVU 4:2:2 1920x1080	Alpha: no Bitrate: 800 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16,24 bit; 48kHz • 4 ch: 16,24 bit; 48kHz • 8 ch: 16,24bit; 48kHz • 16 ch: 16,24; 48kHz
.avi	Uncompressed YUAVUA 4:2:2:4 1920x1080	Alpha: yes Bitrate: 1200 Bitdepth: 8 VBI: N/A Timecode: N/A TDIR: no	No Audio PCM: • 2 ch: 16,24 bit; 48kHz • 4 ch: 16,24 bit; 48kHz • 8 ch: 16,24bit; 48kHz • 16 ch: 16,24; 48kHz

XAVC

Container	Codec	Features	Audio
.mxf (OP1a)	H.264 level 5.2, 3840x2160	Alpha: no Bitrate: 960 Bitdepth: 8, 10, 12 VBI: yes Timecode: yes TDIR: yes	No Audio PCM: • 2 ch: 24 in 32-bit; 48kHz • 4 ch: 24 in 32-bit; 48kHz • 8 ch: 24 in 32-bit; 48kHz • 16 ch: 24 in 32-bit; 48kHz

IMPORTANT! Due to its huge amount of required CPU power, it is not recommended to use this codec. If is to be used, only use the V210 surface format. XAVC codecs should only be used with special hardware like a M264 card.

Copyright © 2026 Vizrt

14 Lights

Lights are used to illuminate objects in a Scene. How lights and materials are used in a Scene makes a big difference in the appearance of a rendered Scene. Materials define how light reflects off a surface, while direct light and ambient light levels define the light that is reflected.

When materials are used in a scene, lights are also required. A Material needs to reflect light to be visible. Without light, just the ambient part of the material is visible, which means that the object is rendered almost black with default settings. Shadows are closely related to lights. Objects that have a light-source applied on them casts a shadow in real life. Shadow mapping is a process by which shadows are added to 3D scenes in Viz Artist for more realistic images, see Shadow Maps.

The following chapters apply to the Classic Render Pipeline only. To get information about how lights are used in the Viz Engine Render Pipeline, please see Lights and Shadows in Viz.

This section contains information on the following topics:

- Types Of Light
- Light Source Animation
- Light Editor
- Light Visualization
- Shadow Maps

Copyright © 2026 Vizrt

14.1 Types Of Light

To understand what effect light has on the surface of rendered 3D objects, three types of light can be defined. When using lights, it is important to understand the effect each type of light creates on the surface of a rendered 3D object.

Compatibility Note: This section applies to the Viz Classic Render Pipeline only. Lights in the Viz Engine Render Pipeline are explained here.

Light Type	Description
Local Light	Local lights have color and position within a Scene, but no single direction. They give off light equally in all directions (isotropic light source). A light bulb is a good example of a local light.
Spot Light	Spotlights have color, position, and direction in which they emit light. Light emitted from a spot light is made up of a bright inner cone and a larger outer cone, with the light intensity diminishing between the two.
Infinite Light	Infinite lights have color and direction. They emit parallel light. This means that all light generated by infinite lights travels through a Scene in the same direction. Imagine a directional light as a light source at near infinite distance, such as the sun.

Copyright © 2026 Vizrt

14.2 Light Source Animation

All light source parameters can be animated in the same way as all other animations:

1. Set a value in the Light editor.
2. Add a Key Frame.
3. Change the value.
4. Add a Key Frame again.

Animations can also easily be created in one of the orthogonal views. There, the light sources show as symbols, and by repositioning them and adding Key Frames, animation sequences are created. For more information, see Light Visualization.

More advanced light source animations are normally created with the light source tracking feature, see To Set a Light to Track Containers.

For more general animation information, see Create Animations.

Copyright © 2026 Vizrt

14.3 Light Editor

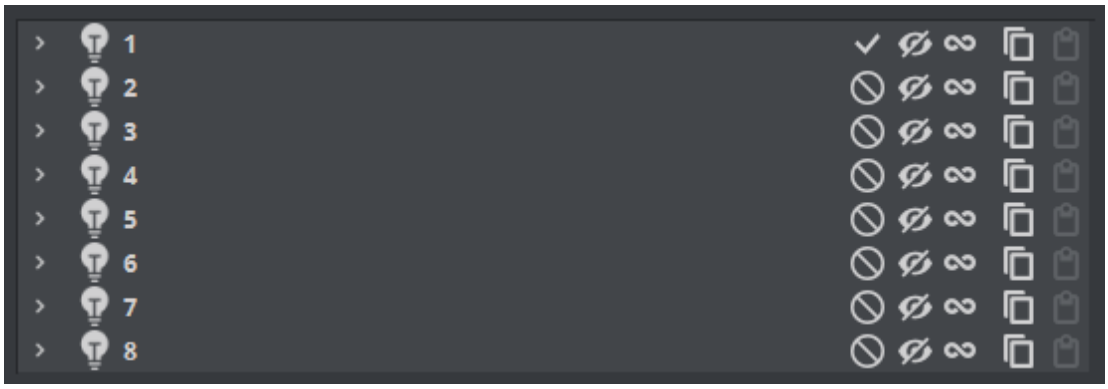
The Light editor modifies the light settings in a Scene. A Scene can have up to eight active, simultaneous, lights.

This page contains information on the following topics:

- Light Editor
- Light Editor Properties
- Light Editor Procedures
    - To Apply Light in a Scene
    - To Set a Light to Track Containers
    - To Copy the Properties of One Light to Another Light

14.3.1 Light Editor

The Light Editor panel shows a list of all eight editable lights, and their current status.

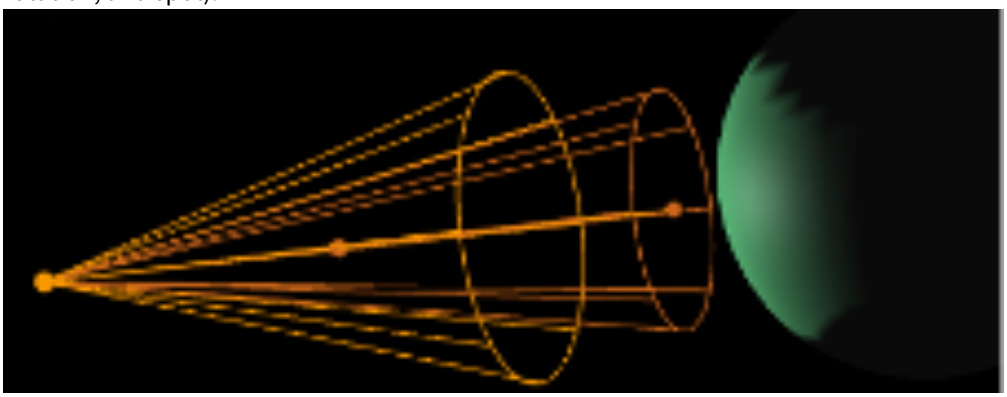


- **Number:** The number of each available light, from `1` to `8`.
- **State:** Sets the light `On` or `Off`.
- **Vis:** Sets the visualization of a light `On` or `Off` (see Light Visualization):
    - **On:** The visualization of a light can be seen in the Scene Editor (and the Scene Editor Views).
    - **Off:** The visualization of a light cannot be seen in the output window.
- **Type:** The type of light used, which is selected in the Light Editor Right Panel.
    - **Local:** Gives off light equally in all directions (color and position).

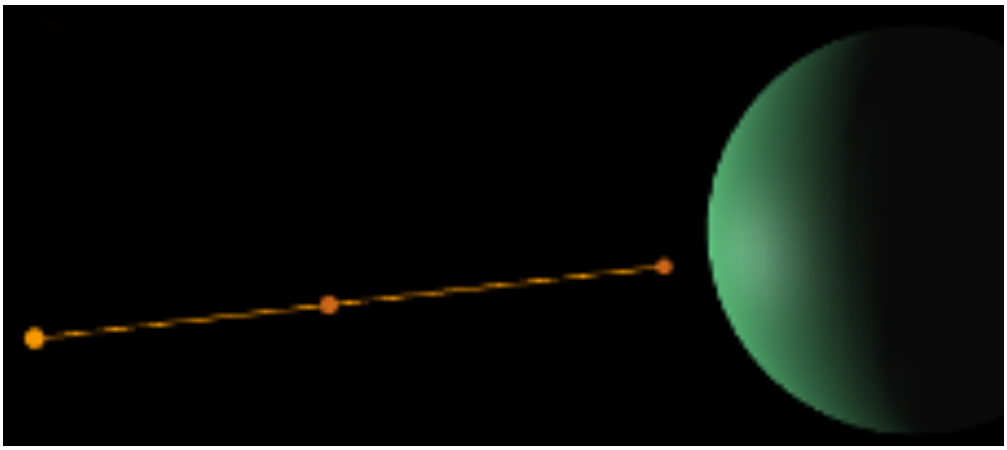
Copyright © 2026 Vizrt



- **Spot:** Gives off light in a bright inner cone and a larger outer cone (color, position, direction/rotation, and spot).



- **Infinite:** Gives off parallel light (color, position and direction/rotation). Infinite is selected by default.



- **Copy:** Copies light information from selected light.
- **Paste:** Pastes copied light information to selected light.

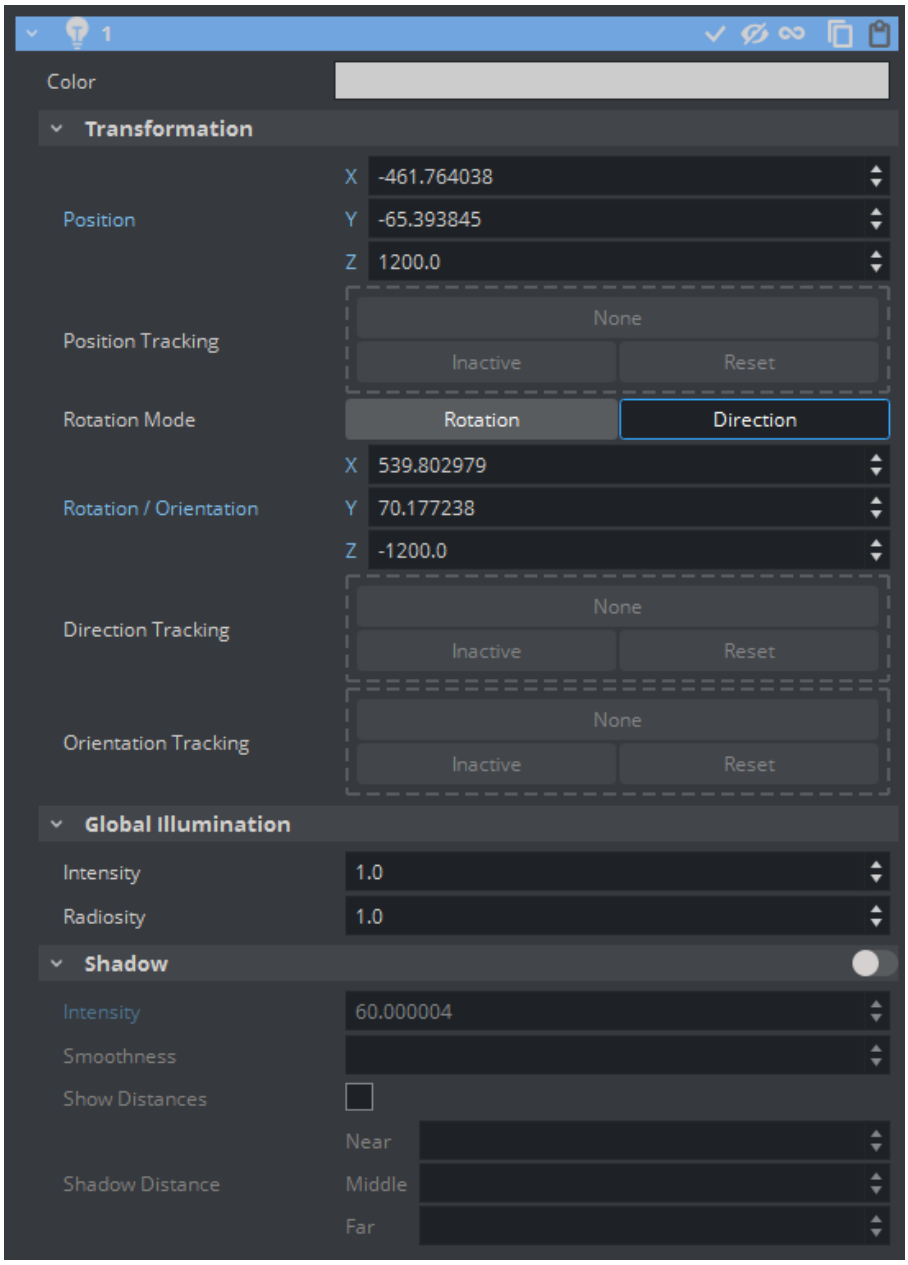
Note: The Bounding Box (BB (see Scene Editor Buttons)) must be set to `On`. If it is set to `Off`, no light visualizations can be seen in the Scene Editor or Scene Editor Views.

Copyright © 2026 Vizrt

14.3.2 Light Editor Properties

Selecting a camera from the Light Editor Panel expands to show the editable parameters for the currently selected light.

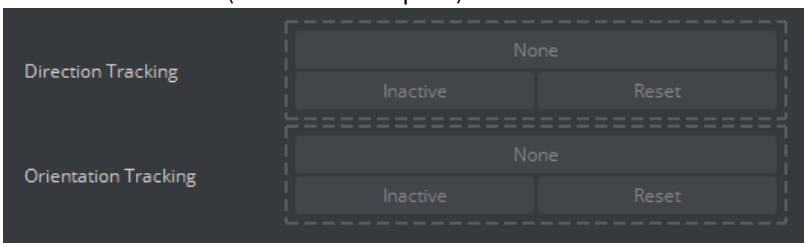
The Light Editor right panel shows the editable parameters for the currently selected light:



- **Color:** Sets the color of the light.
- **Position:** Sets the position of the light along the X, Y, and Z axis.
- **Direction:** Defines the direction vector for **Spot** or **Infinite** light.

Copyright © 2026 Vizrt

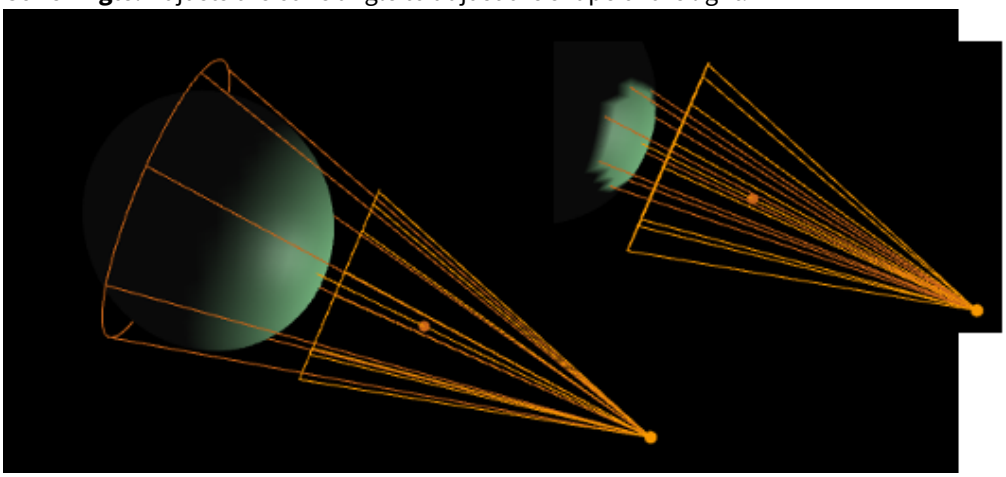
- **Rotation:** Sets the values to rotate the **Spot** or **Infinite** light sources.
- **Tracking:** A light can be configured to track the Position, and Direction or Orientation of a Container (see To Set a Light to Track Containers):
  - **Position Tracking:** The position of the light is the same as the target Containers position.
  - **Direction Tracking:** The direction of the light is updated to always point towards the current position of the target container.
  - **Orientation Tracking:** The rotation (orientation in space) of the light is the same as the target Containers rotation (orientation in space).



  The following parameters can also be adjusted for tracking with light:
  - **Active/Inactive:** Drag a Container to the drop zone, to automatically activate light tracking. To stop tracking, click the **Active/Inactive** button.
  - **Reset:** Disables light tracking.

Note: Direction and Orientation tracking cannot be used at the same time. The results would contradict each other (both of them result in a potentially different rotation value for the container).

- **Spot** (available when **Spot** is selected):
  - **Penumbra:** Sets the concentration of the light (penumbra) within the light cone of the Spot light. When set to zero, the whole light cone has the same intensity. If set greater than zero, the intensity decreases away from the center
  - **Cone Angle:** Adjusts the cone angle to adjust the shape of the light.



Note: A Material item must be used to show light effects.

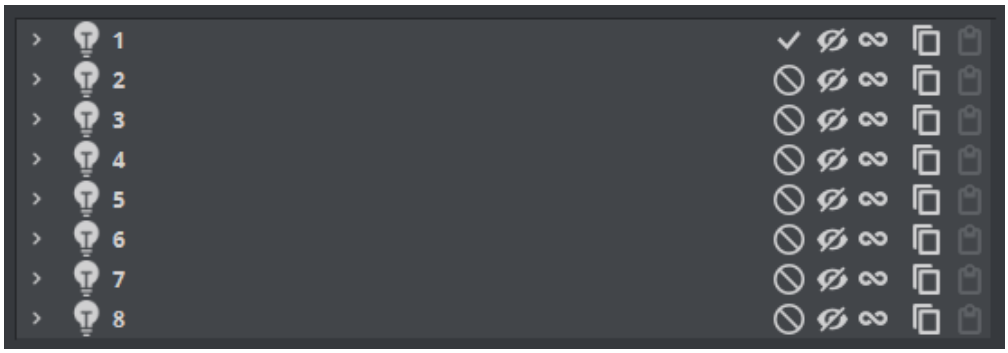
Copyright © 2026 Vizrt

14.3.3 Light Editor Procedures

To Apply Light in a Scene

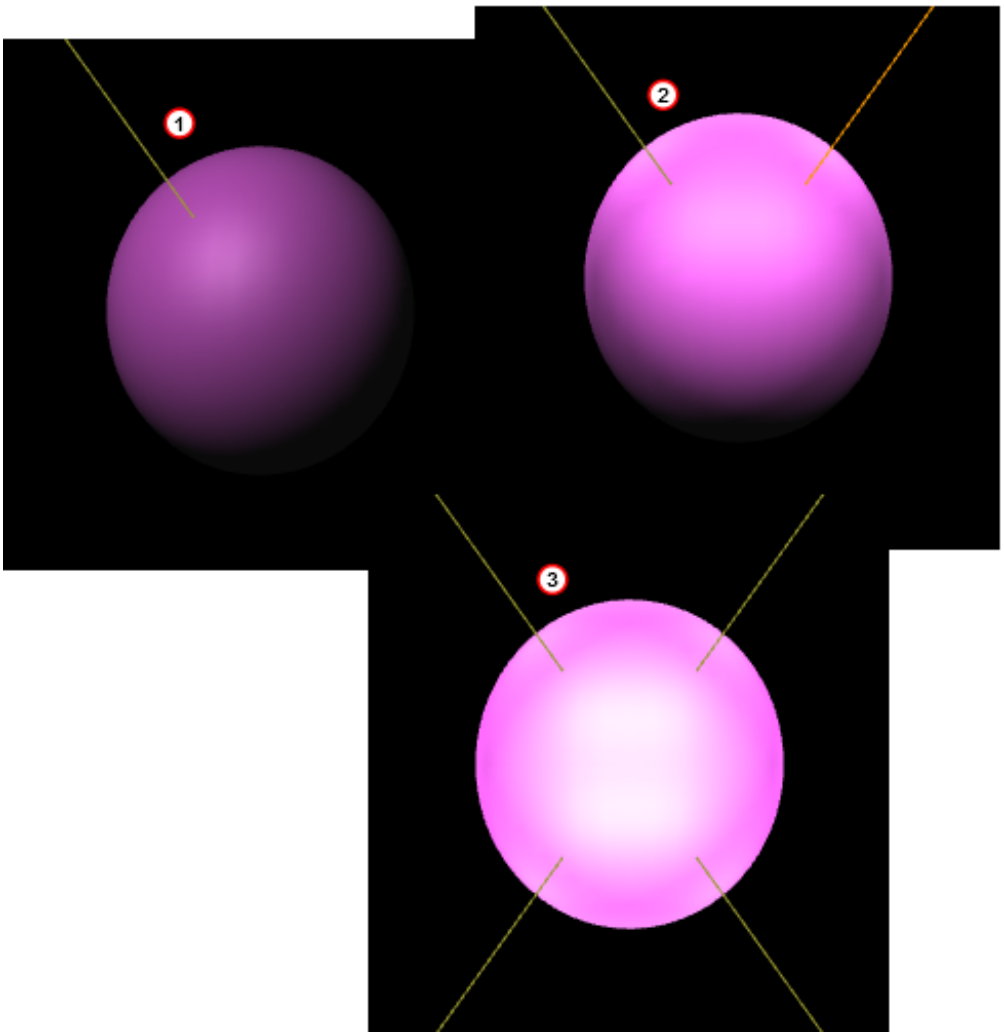
1. Click **Built Ins**.
2. Click the Geom plug-ins tab.
3. Drag an object into the Scene Tree.
4. Add a material to the same Container.
5. Click the **Light** tab to view the Light Editor.

> **Note:** Light number 1 is `On` by default.



6. In the left panel, click any or all of the lights in the **State** column to show different light perspectives:
   - One light (1)
   - First four lights (2)
   - All lights (3)

Copyright © 2026 Vizrt

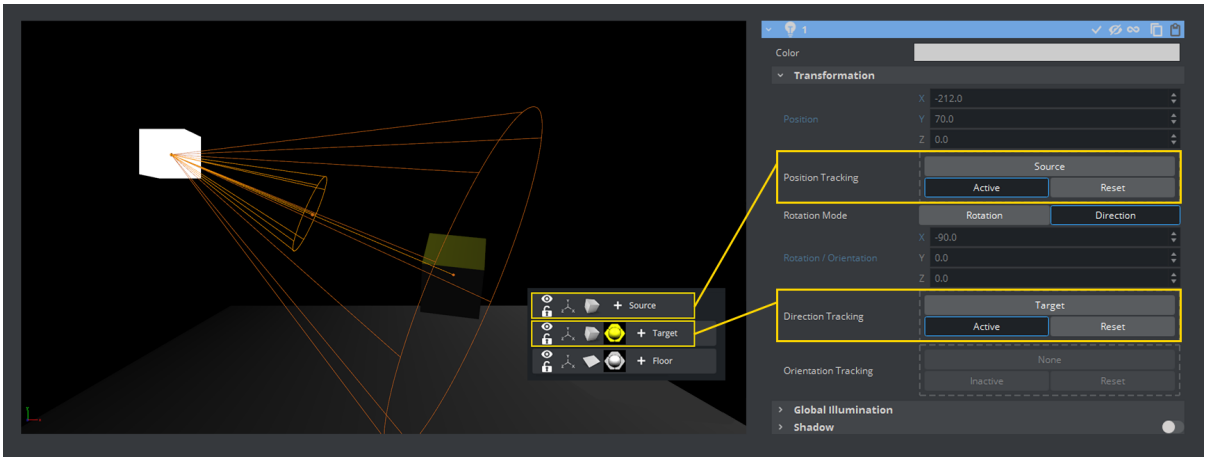


To Set a Light to Track Containers

A light can be configured to track containers in both position and direction (see **Tracking** in the Light Editor Right Panel)

1. In the Scene Tree create two Objects (1), one to track position (*source*) and one to track rotation (*target*).
2. In the right panel select the **Light** tab.
3. Drag *source* to the Position Tracking drop zone (2).
4. Drag *target* to the Direction Tracking drop zone (3).

Copyright © 2026 Vizrt



5. In the Properties panel click the **Container** tab.
6. Adjust the Position settings to track the objects.

To Copy the Properties of One Light to Another Light

1. Select the light for the source properties. Click the **Copy** button for that light.
2. Select the light for the target properties. Click the **Paste** button for that light.

The target light inherits all properties belonging to the source light.

See Also

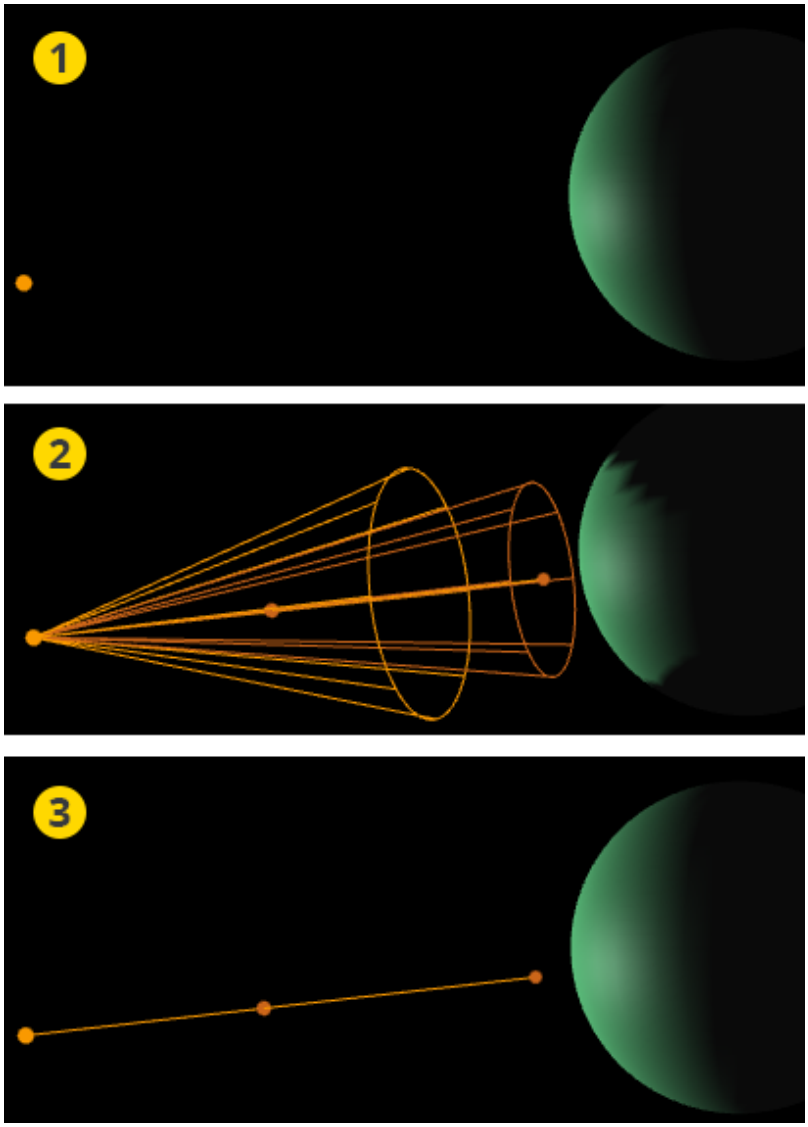
- Types Of Light
- Light Visualization
- Light Source Animation

14.4 Light Visualization

The position, direction or rotation of a light source can be set and modified in the Light Editor or directly in the Scene Editor or the Scene Editor Views, by selection of handles, which are part of the light visualization.

In the Light Editor, a light can be set to `On` or `Off` (its State), and its visualization (editing parameters) can also be set `On` or `Off`.

A Local Light is visualized as a single point (1), and the Spot Light (2) and Infinite Light (3) are visualized as a line with three handles. The **Spot** light also has a wire-frame box that shows the penumbra and light cone angle (the scope of the light).



Copyright © 2026 Vizrt

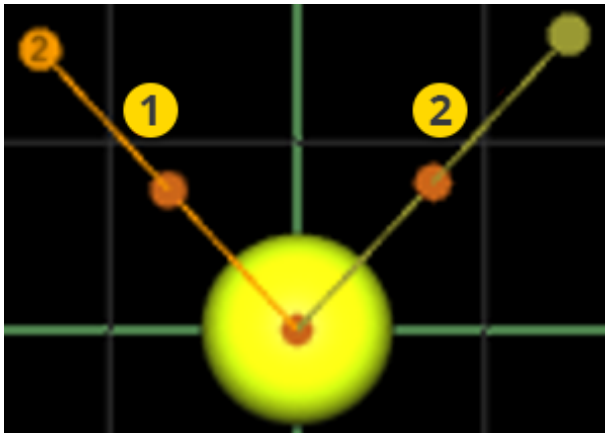
If the **State** of a light is set to `On`, and it is also highlighted in the Light Editor, its visualization is shown in the Scene Editor (if **BB** is clicked (see Scene Editor Buttons)) and the Scene Editor Views.

To view the visualization of all lights, whose **State** is set to `On`, set **Vis** for each light to `On`.



If all available light visualizations are visible, click on a light in the Light Editor to select its visualization in the Scene Editor and the Scene Editor Views.

A selected light visualization shows orange (with its light source number) (1). An unselected light visualization shows green (2), with two handles shown in orange.



Each light visualization has handles, which are used to modify the light direction in the Scene Editor and Scene Editor Views.

14.4.1 To Modify the Light Direction with Light Handles

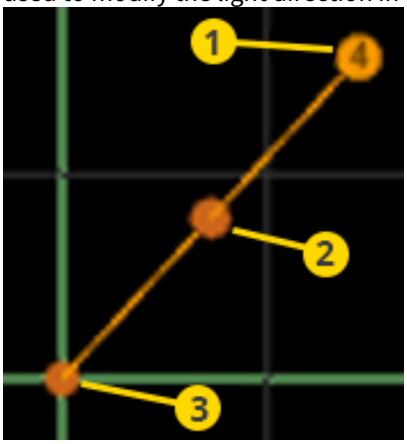
1. Open the Light Editor.
2. Select a light, which has its **State** set to `On`. The visualization of the selected light shows in the Scene Editor.

> **Note:** To view the visualization of all active lights set **Vis** to `On` for each light.

Copyright © 2026 Vizrt



3. Click on a light visualization handle and drag where required. Each light visualization has handles, which are used to modify the light direction in the Scene Editor and Scene Editor Views.



- **Handle** (1): The light source. Drag the handle to set its position.
- **Handle** (2): Drag the whole light without altering the proportionality of position and direction.
- **Handle** (3): The end point. Drag the handle to set the direction and distance of the light.

Copyright © 2026 Vizrt

14.5 Shadow Maps

Shadow maps in Viz Artist produce more realistic images, but shadows can be resource intensive to render in real-time so care should be taken to consider the effects of using shadows on various graphics hardware.

The concept of shadows in a scene is closely related to how you see shadows in real life: When lights are applied to an object, the object casts a shadow, and the intensity and direction of the light source as well as the distance from the light source to the object that the lights hit affect the shadow. The darkness, intensity and smoothness of the shadow affect the quality of the rendered scene.

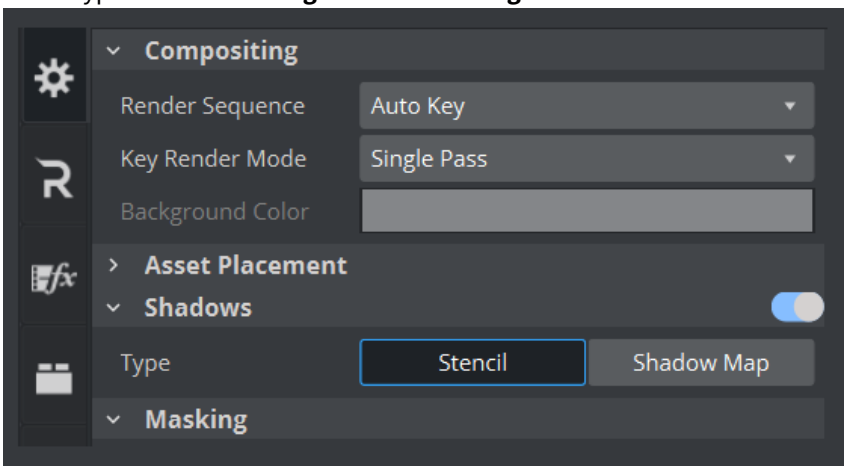
This section contains information on the following topics:

- About Shadow Maps
- To Use Shadows in a Scene

14.5.1 About Shadow Maps

There are two principal methods to show shadows in Viz Artist:

- **Shadow Maps:** More advanced and better looking shadows, but uses more graphic resources. Often called soft shadows. Uses cascaded shadow maps.
- **Stencils:** Simpler, not so photo-realistic, but uses fewer graphic resources. You enable shadow maps and select type in **Scene Settings > Global Settings:**

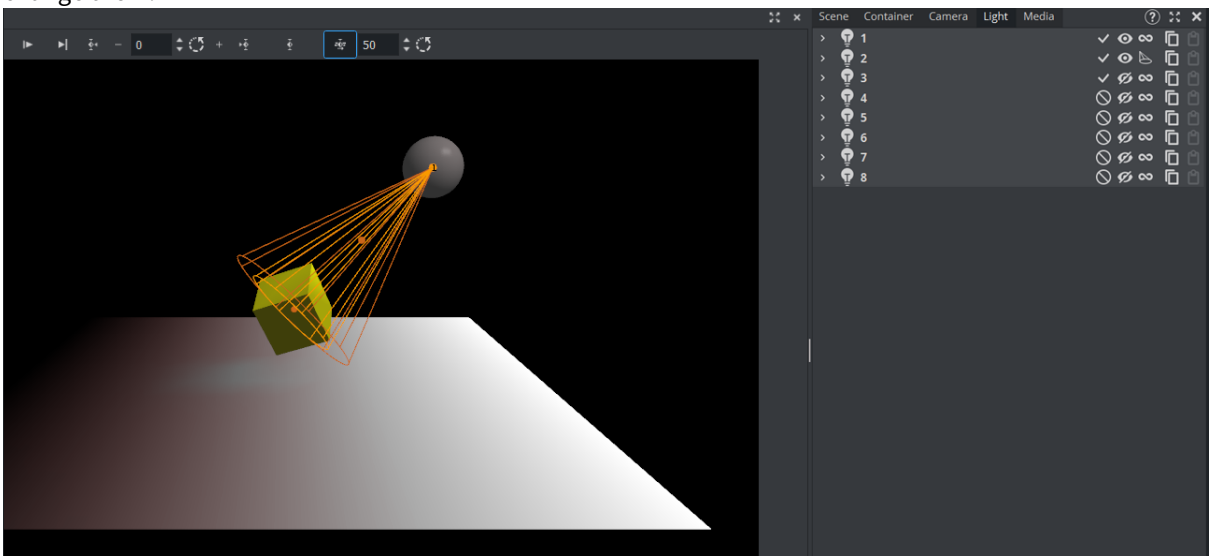


Select Light from the menu to show and change the shadow map properties:

Copyright © 2026 Vizrt



- **Intensity:** Sets the intensity of the shadow effect, in percent. Controls the alpha value (makes the shadows darker).
- **Smoothness:** Determines how smooth the edges and the shadow itself blend with the surroundings given a light-source. Makes shadows smoother the farther away a receiver is from the shadow caster. Possible values: `1` to `10`.
- **Show Distances:** Shows a color-map of the "slices" (areas) affected by the shadow maps. When designing a scene, use the on/off switch to see a color-visualization of how the shadow map affects each slice of the shadow map. Areas affected by *Near* are shown in green, *Middle* in red and Far in magenta as indicated in the image below. The different layers of the shadow map are grouped into *Near*, *Middle* and *Far* slices or areas. The default values for near, middle and far would normally be fine to use as is, but if required you can change them:



  - **Near Distance:** Determines the distance, in viz units, of the near layer of the shadow map.
  - **Middle Distance:** Determines the distance, in viz units, of the middle layer of the shadow map.
  - **Far Distance:** Determines the distance, in viz units, of the far layer of the shadow map.

Copyright © 2026 Vizrt

14.5.2 To Use Shadows in a Scene

1. Go to **Light** and select the light source that should cast a shadow. More than one caster is possible but this comes with a noticeable performance hit for every additional shadow casting light. After selecting the light, on the property panel click the button that says Shadow to change property values for shadows, if required.
2. From **Plugins > Container Plugins** select the **Caster** and **Receiver** plug-ins and drop them on containers to show shadows. The layer parameters correspond to light sources, so you can have two lights cast shadows but containers can only cast or receive shadows from one of the light sources.

Note: The shadow mapping technique can be used together with lens-deformation as well, see Rendering Panel.

Copyright © 2026 Vizrt

15 Cameras

The camera which shows the Viz Artist Scene can be controlled much the same way that a real camera is controlled. Viz Artist cameras are used by assigning two tracking objects. The first is the camera itself, and the second is the object being tracked.

This section contains information on the following topics:

- Track Objects with a Camera
- Camera Editor
- Stereo Settings
- Change Camera Parameters in Orthogonal Views
- Receive Tracking Data from a Real Camera
- Copy Properties from One Camera to Another
- Camera Selection
- Camera Animation
- Advanced Lens Distortion

Copyright © 2026 Vizrt

15.1 Track Objects with a Camera

A virtual camera can be set up to track Containers both regarding position and direction. Position Tracking means that the position of the camera is in the center of the container. Direction Tracking direction means that the camera looks at the center of the container, for example throughout an animation. There are three options to configure the parameters view, The options depend on the number of tracking objects:

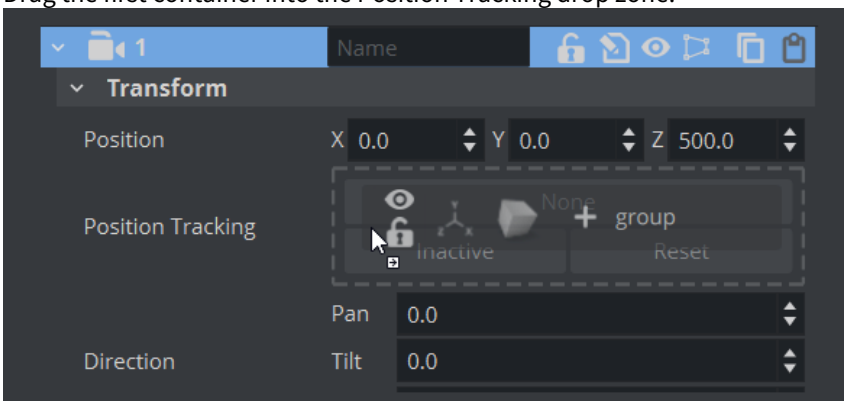
- **Zero Tracking Objects:** Only Position X and Y can be modified.
- **One Tracking Object:** When a Container is dragged to the Position Tracking drop zone, all the parameters are grayed out. In the Properties Panel, switch from **Camera** to **Container** to modify Positions X, Y and Z.
- **Two Tracking Objects:** Only the Lens and Relative Position parameters can be modified.

15.1.1 To Track an Object

1. In the Scene Tree create two Objects, one to track position (Sphere_1) and one to track rotation (Sphere_2).

Tip: It is better to use objects with a low polygon count, like a rectangle, as a tracking object. High polygon tracking objects, such as a sphere, make the scene heavy and slow.

2. Drag the first container into the Position Tracking drop zone.



3. Drag the second container into the Direction or Orientation Tracking drop zone.

Note: Direction and Orientation tracking cannot be used at the same time. The results would contradict each other (both of them result in a potentially different rotation value for the container).

4. Modify the camera parameters: Select the Position and, or, Direction tracking object in the Scene Tree, and change the position and/or direction in the Container properties (select **Container** from the Properties Panel).

See Also

- Camera Editor
- Stereo Settings
- Stereoscopy Best Practices
- Stereoscopic Output Using Shutter Glasses
- Change Camera Parameters in Orthogonal Views
- Receive Tracking Data from a Real Camera

Copyright © 2026 Vizrt

- Copy Properties from One Camera to Another
- Camera Selection
- Camera Animation
- **Tracking** in Parameters for Perspective View

Copyright © 2026 Vizrt

15.2 Camera Editor



The Camera editor makes it possible to modify the properties for the 16 editable cameras that can be used in Viz Artist. To open the Camera Editor, select the **Camera** tab in the Properties panel.

The Camera editor shows a list of all 16 editable cameras and the current view of each camera. Clicking on a camera shows the parameters for the currently selected camera.

- **Camera Number:** Shows the camera number, from 1 to 16. Selecting a camera automatically sets the viewing perspective in the Scene Editor to that of the selected camera.
- **Name:** Shows the camera name. Click inside the **Name...** box to assign a new camera name.
- **Lock/Unlock:** Locks and unlocks a camera when the Keyhole is clicked. Locked cameras are indicated by an orange icon. Except for shortcuts that switch to other cameras, the Scene Editor keyboard shortcuts are disabled for locked cameras. The properties of a locked camera can still be edited through manipulating it in the Scene Editor Views, or by changing the camera parameters in the Camera Editor.
- **Control:** Toggles between:
  - **Remote:** Receives tracking data from a real camera when enabled.
  - **Editor:** Configures camera settings when enabled.
- **Vis:** Shows the status of the camera:
  - **Off:** The camera is not shown in the six predefined orthogonal views. To see a camera in a predefined orthogonal view can help to change the properties of a camera, by dragging the parameters in those views.
  - **On:** Shows the camera.
  - **Inf:** The camera is shown with infinite lines. This is a good aid for visualizing the scope of a camera that is located far away.

Copyright © 2026 Vizrt

- **View:** Shows the current Camera Editor view of each camera. Options are:
  - **Perspective:** Perspective view.
  - **Orthogonal:** Orthogonal view.
  - **Window:** Window view.
- **Copy:** Copies camera information from selected camera.
- **Paste:** Pastes copied camera information to selected camera.

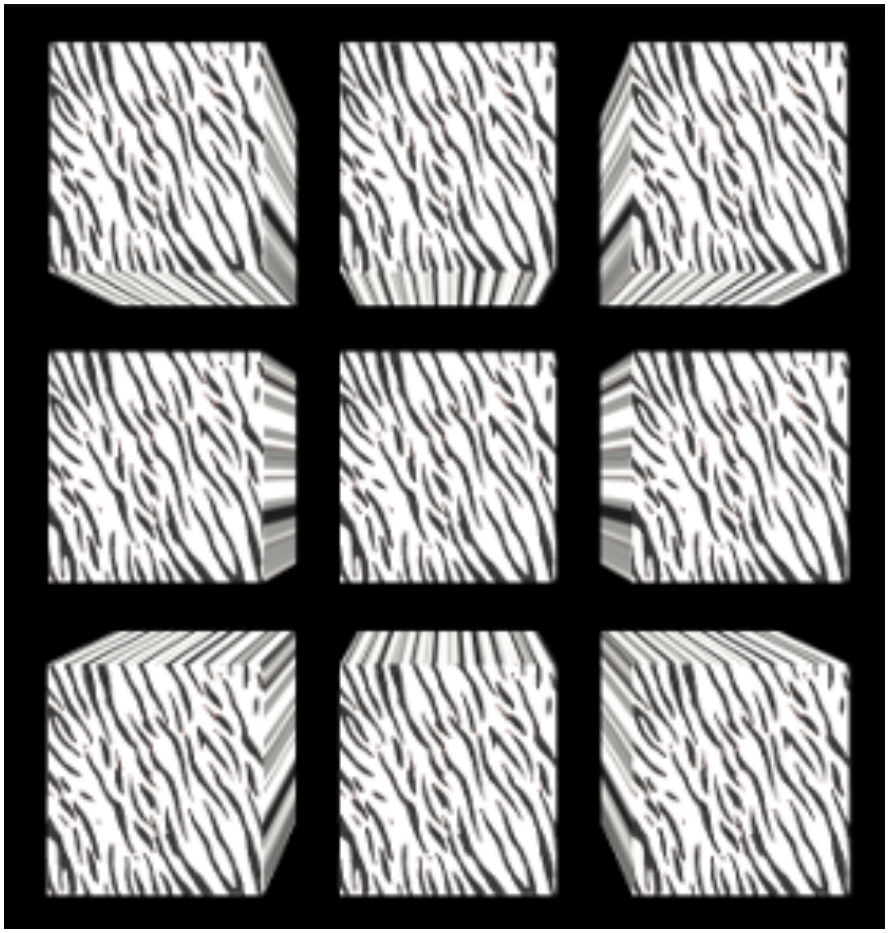
There are three camera views available, and the corresponding parameter options depend on the selected view. The views are described below, and their parameters are explained in the following pages:

- Parameters for Perspective View
- Parameters for Orthogonal View
- Parameters for Window View

Copyright © 2026 Vizrt

15.2.1 Parameters for Perspective View

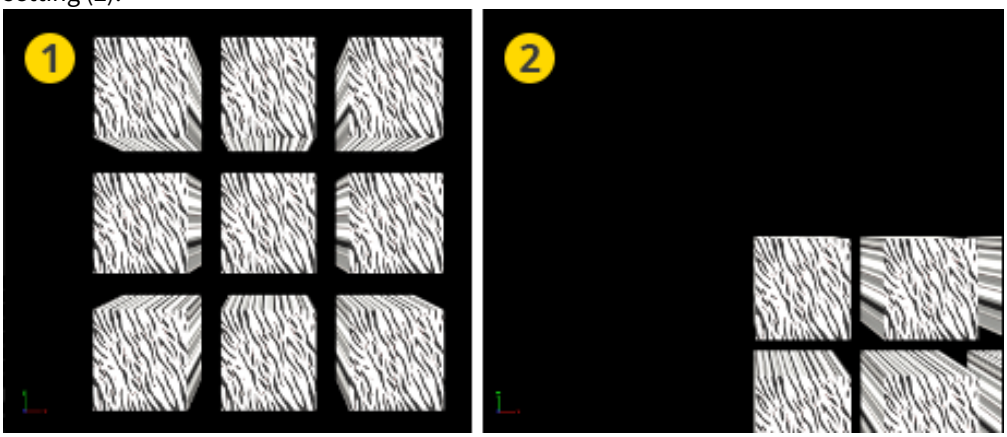
Perspective view shows the basic view of a virtual camera is a normal depth perspective, showing three dimensions.



Copyright © 2026 Vizrt

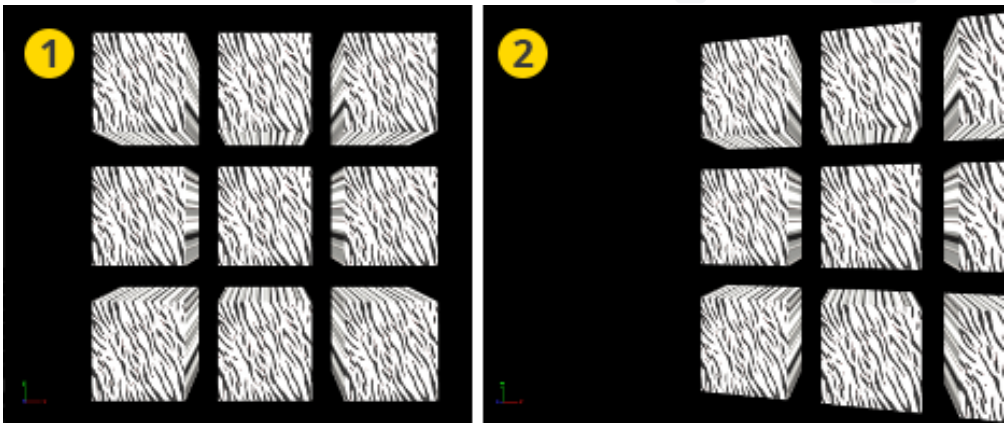


- **Transform**
  - **Position:** Sets the position of the camera along the X, Y, and Z axis. Default setting (1) and modified setting (2).



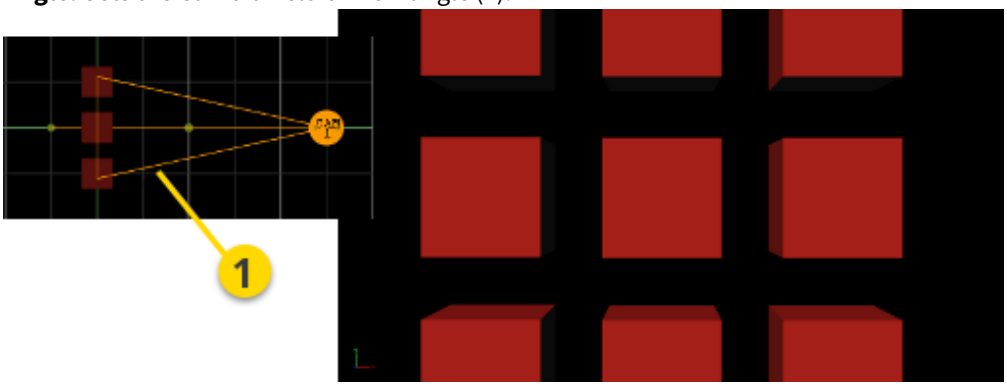
Copyright © 2026 Vizrt

- **Direction:** Sets the values for Pan, Tilt, and Twist. Pan set to `0` (1). Pan set to `15` (2).



- **Relative Position:** Sets the position of the camera along the X, Y, and Z axis, relative to a real camera. Entering a value in for X, Y and Z axis for the Relative Position changes the actual camera position values, but relative to its orientation.

- **Lens**
  - **Angle:** Sets the Camera field of view angle (1).



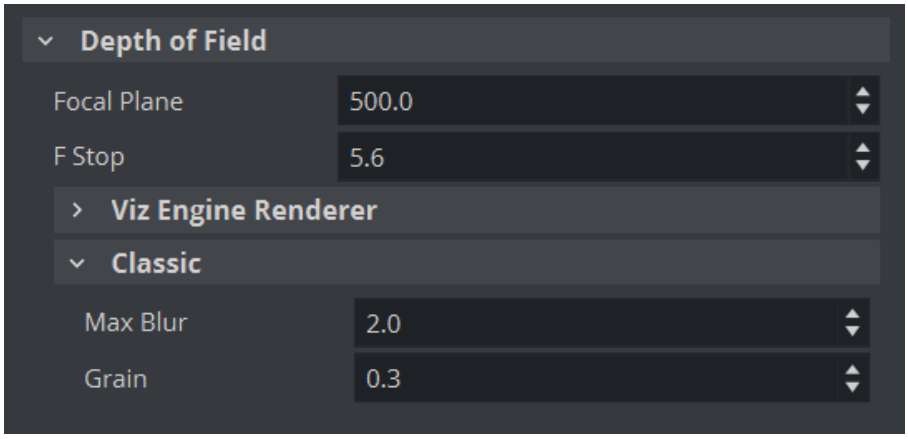
  - **Center Shift X:** Shifts the center of the lens on the X axis.
  - **Center Shift Y:** Shifts the center of the lens on the Y axis.
- **Depth of Field**
  - **Focal Plane:** Defines the distance between the camera and objects which are in focus.
  - **Max Blur:** Sets the size of maximum blur furthest from the camera. For example, a value of `20` signifies that a pixel is enlarged to a size of 20 pixels.
  - **F Stop:** Sets the F-Stop value (relative aperture): The lens model used by the depth of field algorithm mimics a real lens behavior
  - **Grain:** Defines the blur level.
- **Stereo:** Available if set in **Camera Properties** (see the **Configuring Viz** section of the Viz Engine Administrator Guide).
  - **Eye Dist.:** Fine-tunes for individual users. A higher value results in more depth whereas a lower value flattens the image.
  - **Conv. Plane:** A higher value for the zero parallax distance means that objects appear to come closer to the viewer, a lower one pushes the graphics to the back.
  - **Method**
    - **Off Axis:** Creates a distortion-free stereo pair. This is the preferred option.

Copyright © 2026 Vizrt

- **Toe-In:** Points the cameras slightly inwards towards each other so that the lens axes converge at a single point (“vergence point”). Objects in the same plane as the vergence point show to be closer to the plane of the image, with other objects seeming to be behind or in front of this point. These values must be set during production in accordance with the convergence values used for the cameras.
- **Verbose:** Linked to cameras in Virtual Studio. For more information about the Virtual Studio option see the Virtual Studio documentation.

Depth of Field (Classic)

Depth of Field (DOF) can be set in the **Camera** (see the **Configuring Viz** section of the Viz Engine Administrator Guide) Configuration.



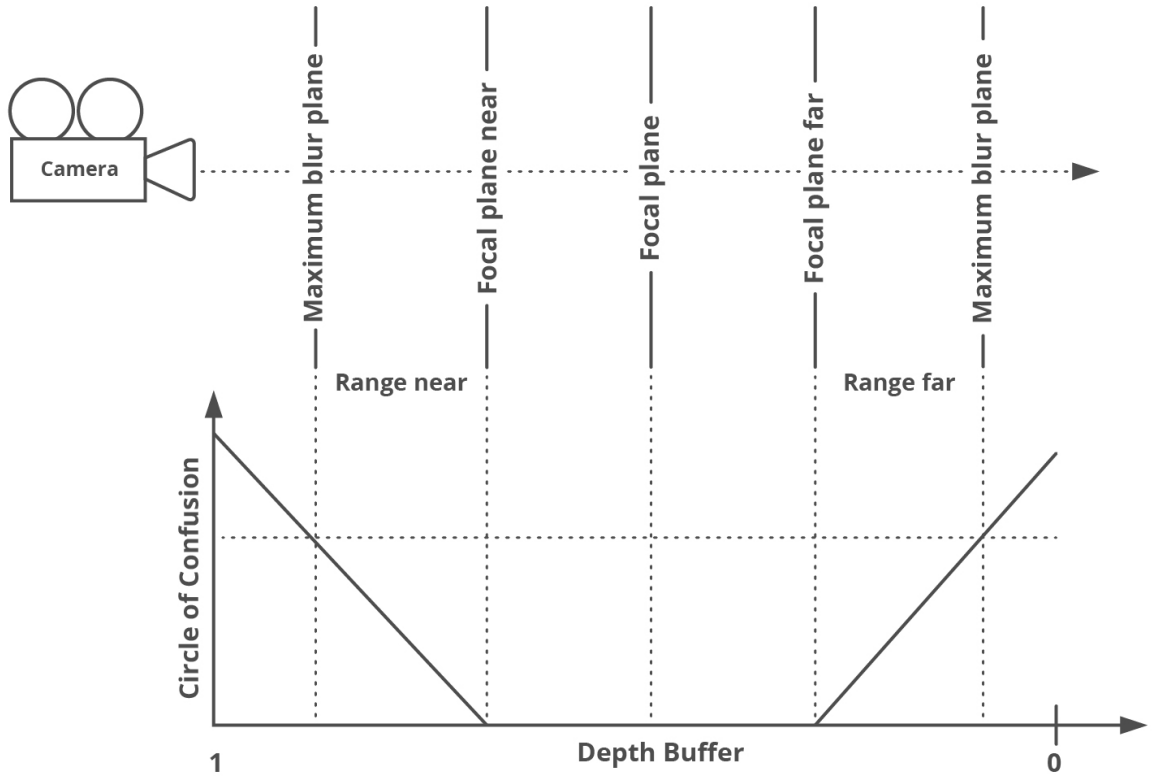
To Set DOF to Follow the Camera Focus

1. Adjust the Max Blur. This value doesn't need to be higher than `12` to give a good result.
2. Adjust the F-Stop. Changing the iris also changes the range of depth. Try a value around `10`.
3. Adjust the Grain. This value sets the noise and influences the DOF. Try a value between `0.001` and `0.003`.

To Avoid Artifacts

To avoid artifacts, avoid using maximum blur levels which are higher than 30 pixels. A blur radius of 30 pixels creates blurry results which should be sufficient. It is possible to increase this maximum radius, but this also increases the rendering time.

Copyright © 2026 Vizrt



Set the clipping planes in the range you really need for the scene. So if your scene has only a depth of 200, set the far clipping plane to `250` or a similar value. Since the depth of field shader takes the depth from the depth buffer, this gives more accurate results. Try to avoid high blur levels for objects before the focal plane, as this can cause unwanted effects. You can increase performance, but artifacts would show for blur levels lower than 30px.

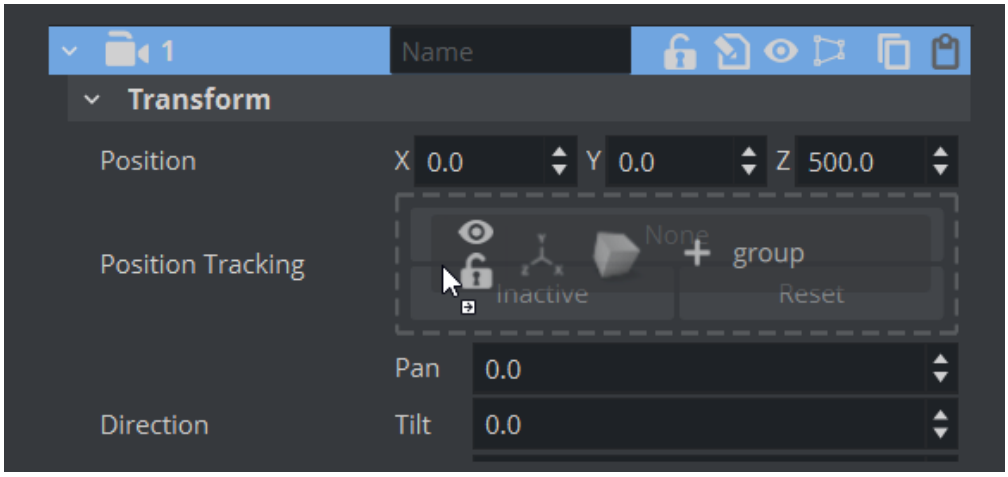
Depth of Field (Viz Engine Render Pipeline)

For adjusting the DOF parameters with Viz Engine Render Pipeline, please see the section Postprocessing Effects - Cinematic Depth of Field.

Tracking

A camera can be set up to track the Position, Direction or Orientation of a Container.

Copyright © 2026 Vizrt



- **Position Tracking:** The position of the camera is the same as the target containers position.
- **Direction Tracking:** The rotation of the camera is updated to always point towards the current position of the target container.
- **Orientation Tracking:** The rotation (orientation in space) of the camera is the same as the target containers rotation (orientation in space).

Note: Direction and Orientation tracking cannot be used at the same time. The results would contradict each other (both of them result in a potentially different rotation value for the container).

- **Active/Inactive:** Drag a container to the drop zone to automatically enable camera tracking. To disable tracking, click the **Active**/**Inactive** button.
- **Reset:** Disables camera tracking.

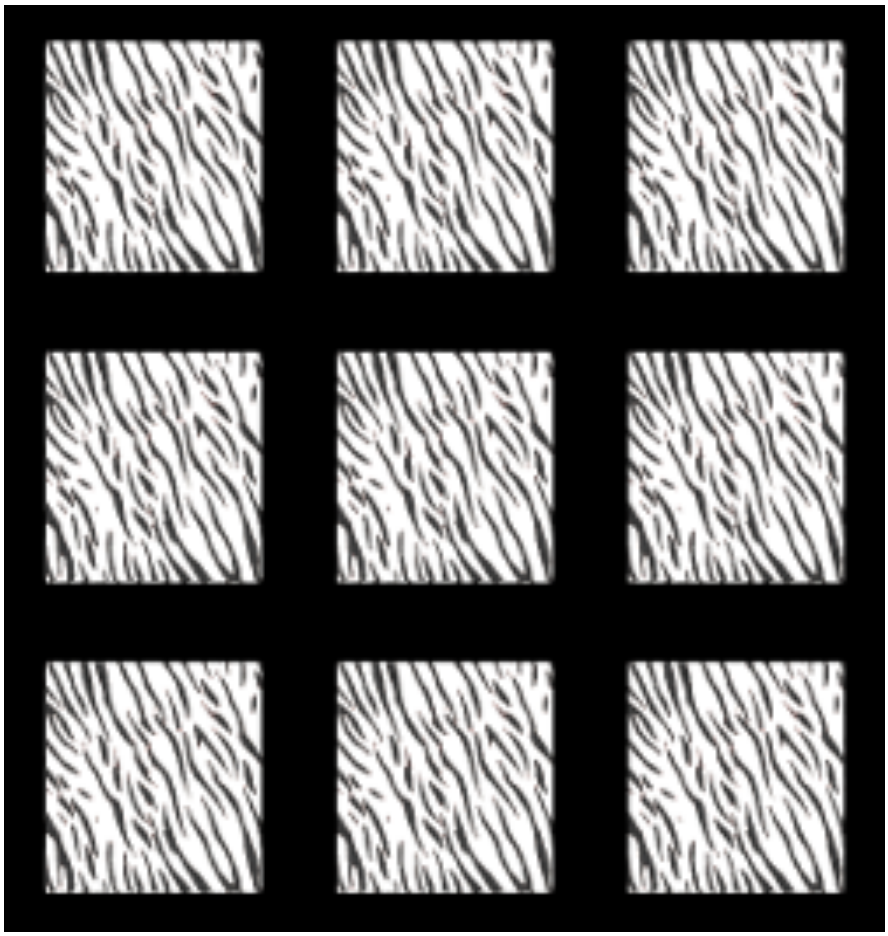
See Also

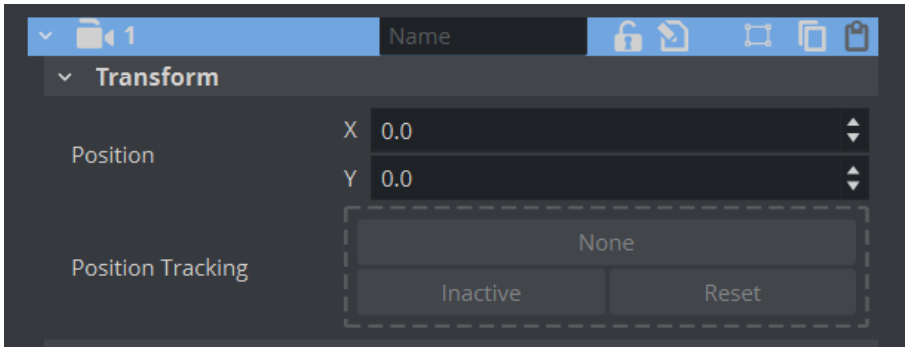
- Track Objects with a Camera
- Parameters for Orthogonal View
- Parameters for Window View

Copyright © 2026 Vizrt

15.2.2 Parameters for Orthogonal View

Orthogonal View shows a camera view without perspective, so that objects do not change their size when being moved towards or away from the camera.





The parameters for Orthogonal view are:

- **Transform**

- **Position:** Sets the position of the camera along the X and Y axis.
- **Tracking:** Apply a container so that the position is based on this container.

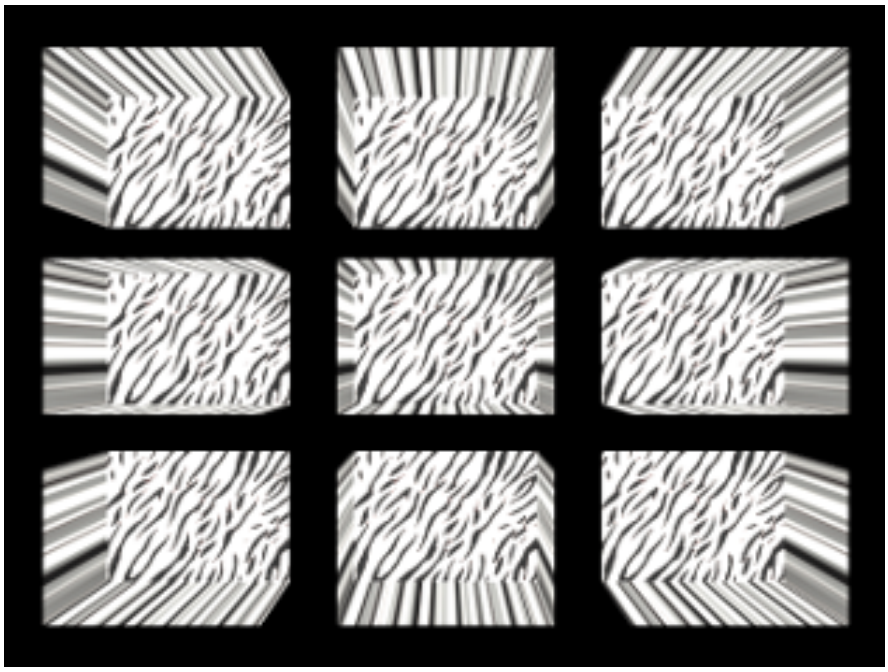
See Also

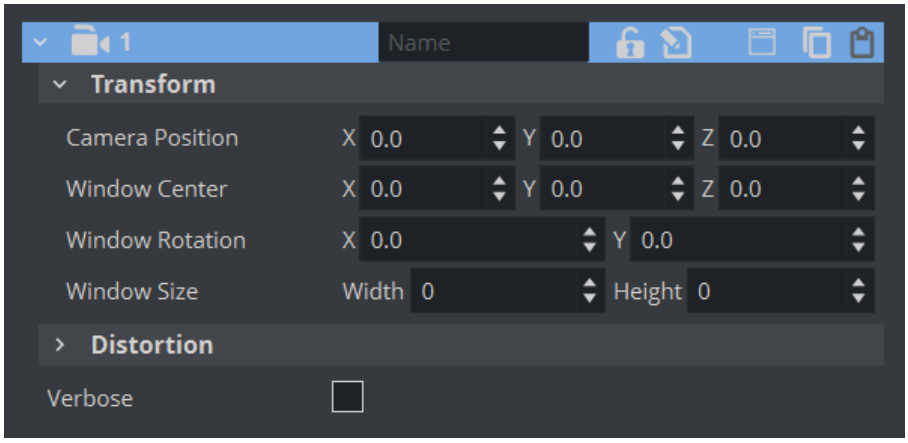
- Parameters for Perspective View
- Parameters for Window View

Copyright © 2026 Vizrt

15.2.3 Parameters for Window View

Window View shows the camera view like looking through a real window. The graphics, through a virtual camera, are adapted according to the real camera movement to produce this behavior. Set parameters for the physical position and rotation of the display unit.





- **Camera Position:** Sets the position of the camera along the X and Y axis.

Note: When Control is set to Remote, Camera Position is not available and the Tracking system controls the Camera Position.

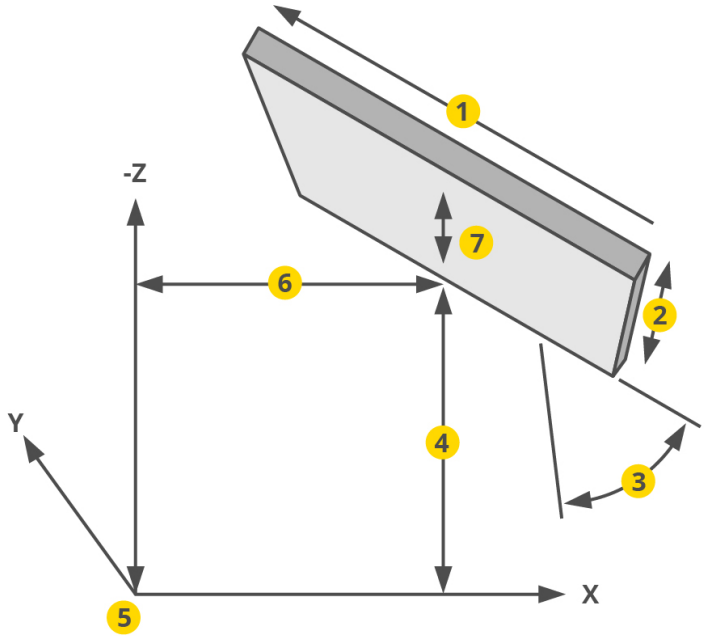
- **Window Size:** Sets the window size (set to the size of the display unit).
- **Window Center:** Sets the position of the window from the zero position of the Tracking system.

Copyright © 2026 Vizrt

- **Window Rotation:** Rotates the window in the Y axis.

Configure a Virtual Window

1. Set the **View** to **Window**.
2. Set the **Control** to **Remote**.
3. Set **Window Size** (1 and 2). The width and height of the video wall, screen or monitor.
4. Set **Window Center** (4, 6 and 7). The distance of the screens center point to the zero point of the studio, defined in the studio setup.
5. Set **Window Rotation** (3).The rotation (y axis) angle of the video wall, screen or monitor, in relation to the coordinate system of the studio.



Item	Description
1	Window Width
2	Window Height
3	Window Rotation
4	Window Center
5	Zero Point of VSS
6	Window Center X

Copyright © 2026 Vizrt

Item	Description
7	Window Center Y

See Also

- Parameters for Perspective View
- Parameters for Orthogonal View

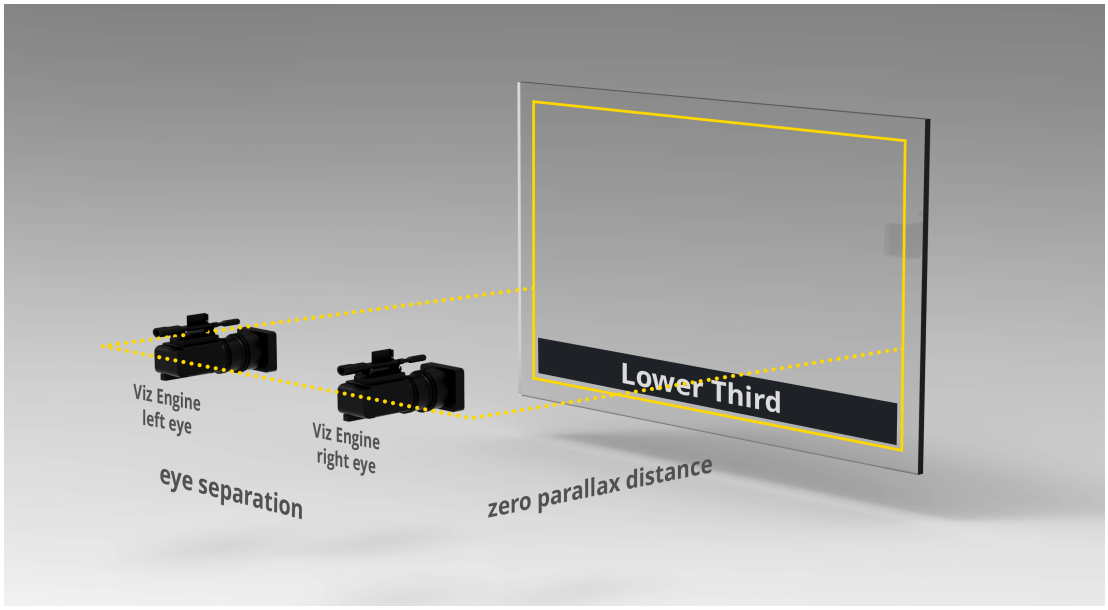
Copyright © 2026 Vizrt

15.3 Stereo Settings

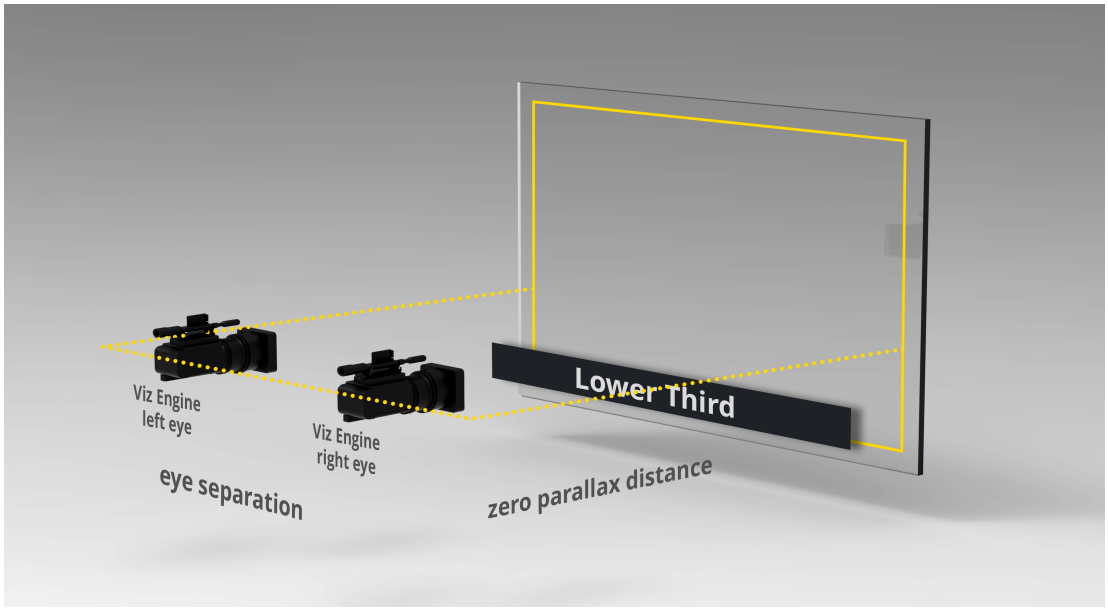
Any Viz Artist animation that was created for 2D broadcast can also be rendered as a stereo pair. There are two parameters which define how each image is rendered. Both parameters are part of the camera configuration, but they can be changed remotely on the fly as well.

Zero parallax means that the pixels for the left and the right eye are identical. An object which would be positioned at zero parallax distance from the camera would be identical for the left and right eye. The eye separation defines the distance of the cameras for the left and right eye. When looking at the drawing the cameras are separated by the same amount as the eyes of the viewer. A different scaling might be used in the scene and the eye separation must be adjusted accordingly.

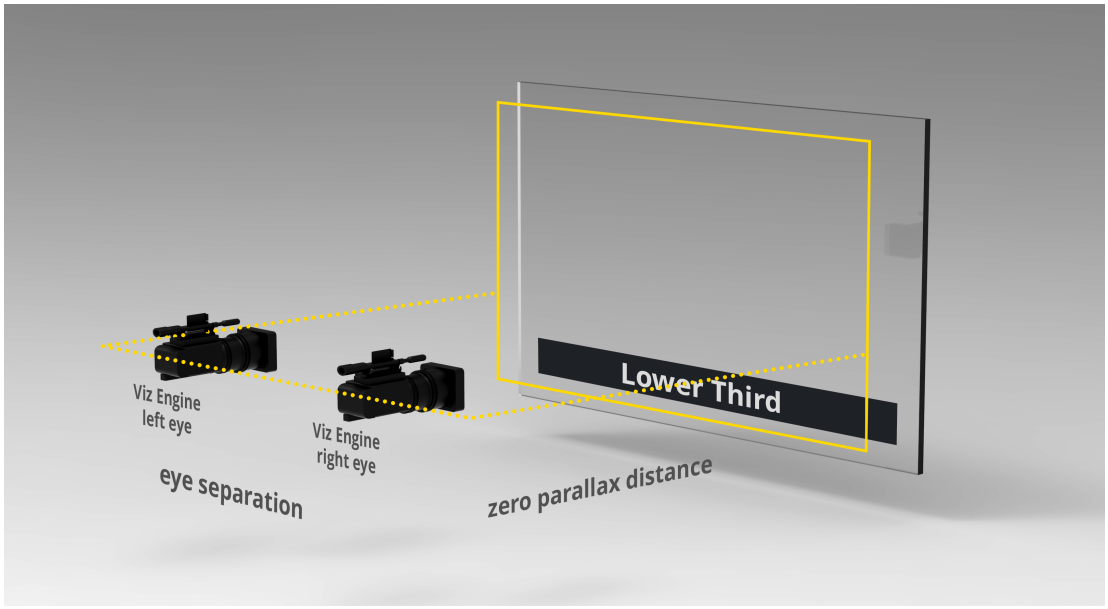
If the scene is built in the same scale as the monitor, with the camera at the same distance from the object as the viewer, and the eyes separated at the same amount an object positioned at the zero parallax, the distance shows to be flat with the screen's surface (1):



If the zero parallax distance is increased the object moves to the front (2):

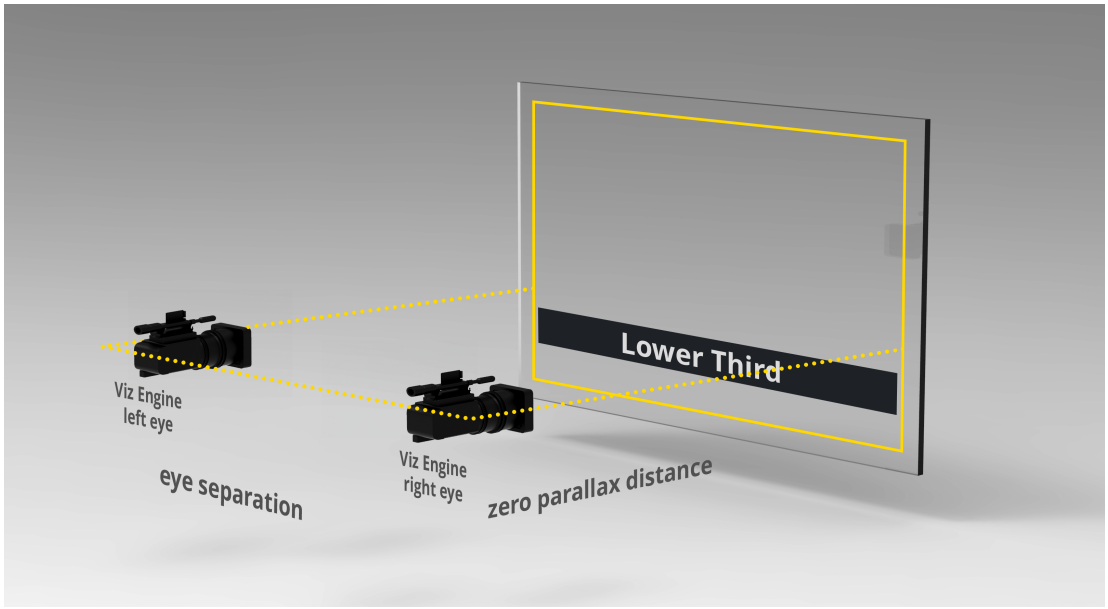


If the distance is decreased the object moves away (3):



The eye separation defines how flat or deep an object shows. If the eye separation is increased, the scene shows deeper (4):

Copyright © 2026 Vizrt



A very large eye separation parameter results in an unreadable 3D representation, as the brain cannot mix the images for the left and right eye anymore. Zero eye separation results in a flat representation, like a 2D image.

See Also

- Stereo Settings in Parameters for Perspective View
- Stereoscopy Best Practices

Copyright © 2026 Vizrt

15.3.1 Stereoscopy Best Practices

The following is a list of what to do and not to do to get the best results.

What to Do

- Think of it as a window you are looking through.
- Make sure that single items which form a new item, like rows of a table, are all at the same distance and not scaled to show the same size in 2D.
- Make sure that all graphics which belong to one show are positioned roughly the same distance away from the camera. This way you do not need to adjust the convergence parameters all the time.
- With alioscopy, keep the floating objects near the 0 plane, otherwise they have artifacts.
- For best effect have objects animate slowly forward and then fade to 0 alpha before hitting the screen edge.
- With alioscopy, for text that is supposed to be readable, keep it big. Dropshadow is a very good effect to enhance the sense of depth. 3DS productions use HD signals.
- For a 24" screen the best viewing distance is about 2.8 meters. For a 40" it is about 4.4 meters.
- With alioscopy, look at the scene from the top view and space things out like in a theatre. Depth and various objects in different Z positions make it more believable: foreground, middle ground and background.
- Wobble effects are fine as long as they do not exaggerate the effect, otherwise you get the paper bag effect. This again depends on how far or close to the camera the object is positioned.

What Not to Do

- Do not use transparency where it goes over video. It's OK if a transparent graphics object is over another graphical item if the distance is not too large.
- Do not use too much depth. Try to keep the items separated not too far from each other. The Z space available for graphics is limited.
- Do not create animations which come in from the side, especially when the final result floats in front of the screen (this is very disturbing).
- Do not use moving flairs or effects.
- Do not have any objects pulsating forth and back in Z space as a part of a loop as this leads to the paper bag effect.

See Also

- Stereo Settings
- Stereoscopic Output Using Shutter Glasses

Copyright © 2026 Vizrt

15.3.2 Stereoscopic Output Using Shutter Glasses

It is very easy to set up a working 3D environment for a designer to start creating stereoscopic scenes. The demands on the hardware are moderate.

Hardware Requirements

- Any workstation
- Any NVIDIA Quadro graphics card with a 3-pin DIN connector
- NVIDIA 3D vision or equivalent
- Monitor with a refresh rate >= 100Hz

Software Requirements

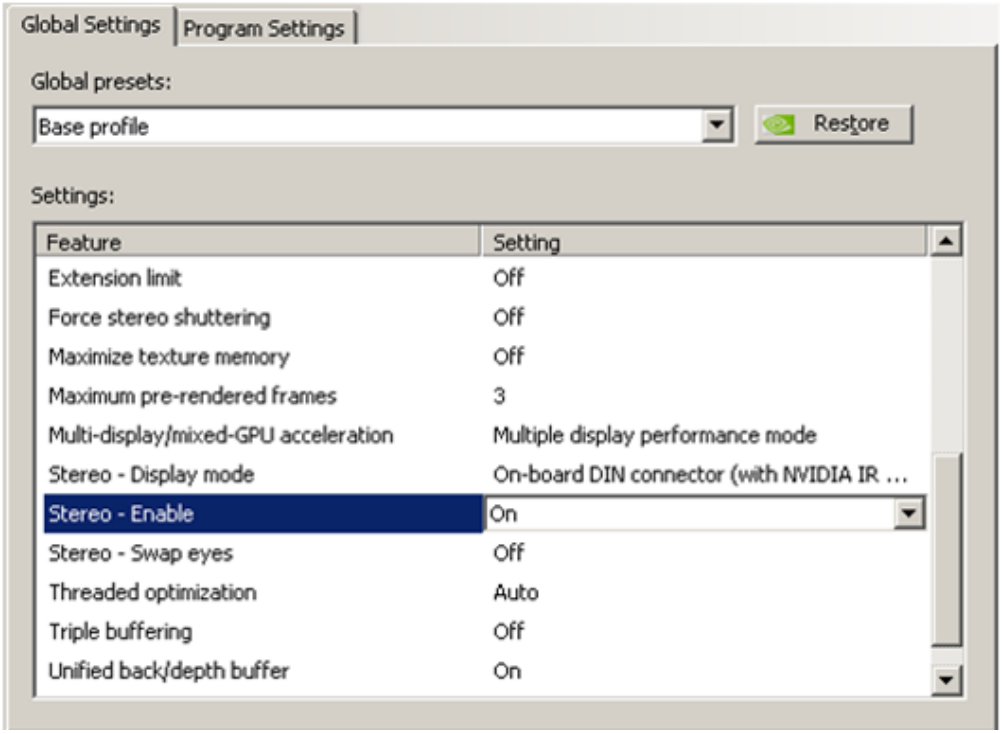
- Viz Artist/Engine 3.3 or later
- The display must be set to a refresh rate >= 100Hz
- **Classic Render Pipeline only.**

To Set Up a 3D Working Environment

1. In the Windows video board 3D settings, Set **Stereo - Enable** to `On`.
2. Select the appropriate **Stereo - Display mode** for each stereo device.

> **Note:** For NVIDIA's 3D Vision, this is 'On-board DIN connector'.

Copyright © 2026 Vizrt



3. In **Camera** settings (see the **Configuring Viz** section of the Viz Engine Administrator Guide) set **Stereo Mode** to **Quad Buffered**.
   Also it is possible to use an autostereoscopic monitor that supports sub-fields or side-fields, such as Miracube's c190 series. The drawback with these displays is that you need to go On Air to get a full-screen image, which is halved in resolution.

See Also

- Camera Editor
- Stereo Settings
- Stereoscopy Best Practices
- Change Camera Parameters in Orthogonal Views
- Track Objects with a Camera
- Receive Tracking Data from a Real Camera
- Copy Properties from One Camera to Another
- Camera Selection
- Camera Animation

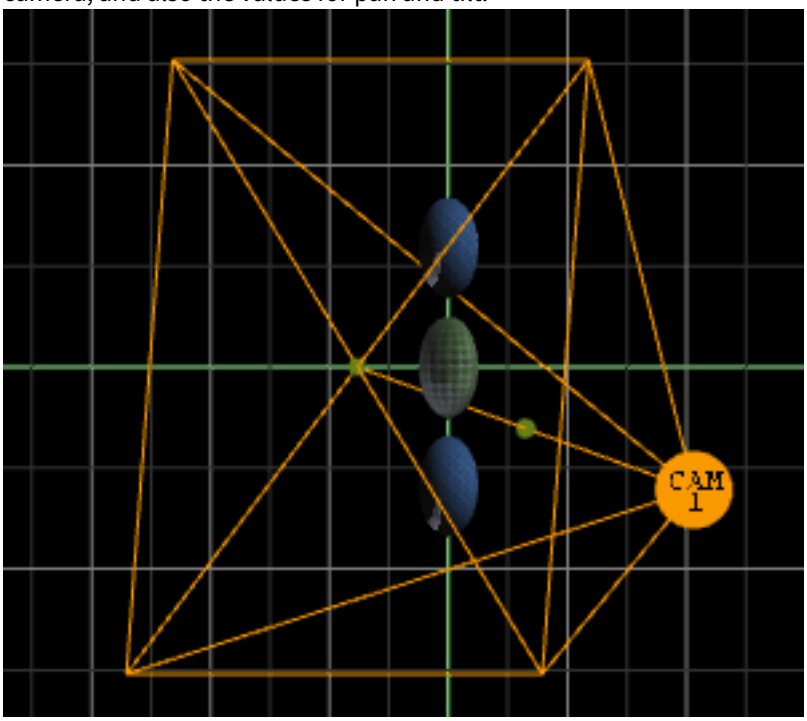
Copyright © 2026 Vizrt

15.4 Change Camera Parameters in Orthogonal Views

Camera parameters can be modified in the predefined orthogonal views.

15.4.1 To View the Predefined Orthogonal Views

- Click **Views**. The camera is shown as an orange circle (3) in the predefined orthogonal views. Two small, green, circular handles (1 and 2) make it possible to modify the position along the X, Y and Z axis of the camera, and also the values for pan and tilt.



15.4.2 To Change the Camera Views

- **Drag handle** (1)**:** Change the pan and tilt while the position of the camera stays fixed.
- **Drag handle** (2)**:** The camera (3) and handle (1) are dragged along, so that pan and tilt are not altered.
- **Drag the camera** (3)**:** Change the position of the camera while handle (1) stays fixed. The values for pan and tilt also change.

See Also

- Camera Editor
- Stereo Settings
- Stereoscopy Best Practices
- Stereoscopic Output Using Shutter Glasses
- Track Objects with a Camera
- Receive Tracking Data from a Real Camera
- Copy Properties from One Camera to Another

Copyright © 2026 Vizrt

- Camera Selection
- Camera Animation

15.5 Receive Tracking Data from a Real Camera

In virtual studio scenarios, as the real cameras moves, the virtual camera should move in the same manner.

15.5.1 To Receive Tracking Data from a Real Camera

In the Camera Editor, in the Control column, click the **Editor** button to set it to **Remote**.



For more information about the Virtual Studio option, see the Virtual Studio documentation.

See Also

- Camera Editor
- Stereo Settings
- Stereoscopy Best Practices
- Stereoscopic Output Using Shutter Glasses
- Change Camera Parameters in Orthogonal Views
- Track Objects with a Camera
- Copy Properties from One Camera to Another
- Camera Selection
- Camera Animation

Copyright © 2026 Vizrt

15.6 Copy Properties from One Camera to Another

The properties applied to one camera can be copied and applied to another camera.

15.6.1 To Copy Camera Properties

In the Camera Editor, click the copy button for the source camera and click the paste button for the target camera. The properties of the source camera are copied onto the target camera.

See Also

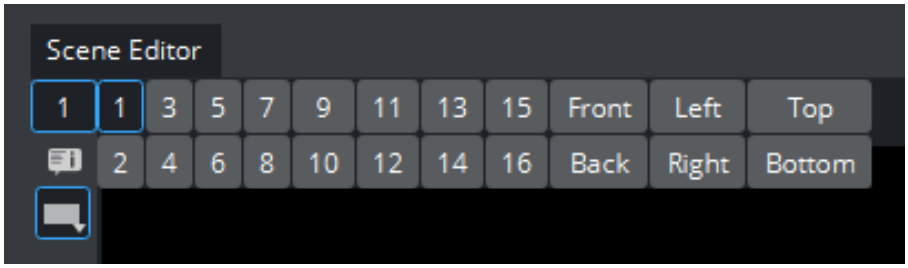
- Camera Editor
- Stereo Settings
- Stereoscopy Best Practices
- Stereoscopic Output Using Shutter Glasses
- Change Camera Parameters in Orthogonal Views
- Track Objects with a Camera
- Receive Tracking Data from a Real Camera
- Camera Selection
- Camera Animation

Copyright © 2026 Vizrt

15.7 Camera Selection

A Scene can be viewed from various camera angles and views, with 16 editable cameras and six predefined orthogonal views available.

In the Scene Editor, Render Views and Scene Editor Views the Camera Selection button shows which camera or view is currently selected and used. Click the button to open the Camera Selection panel.



Each Camera can also be selected or deselected in the Camera Editor.

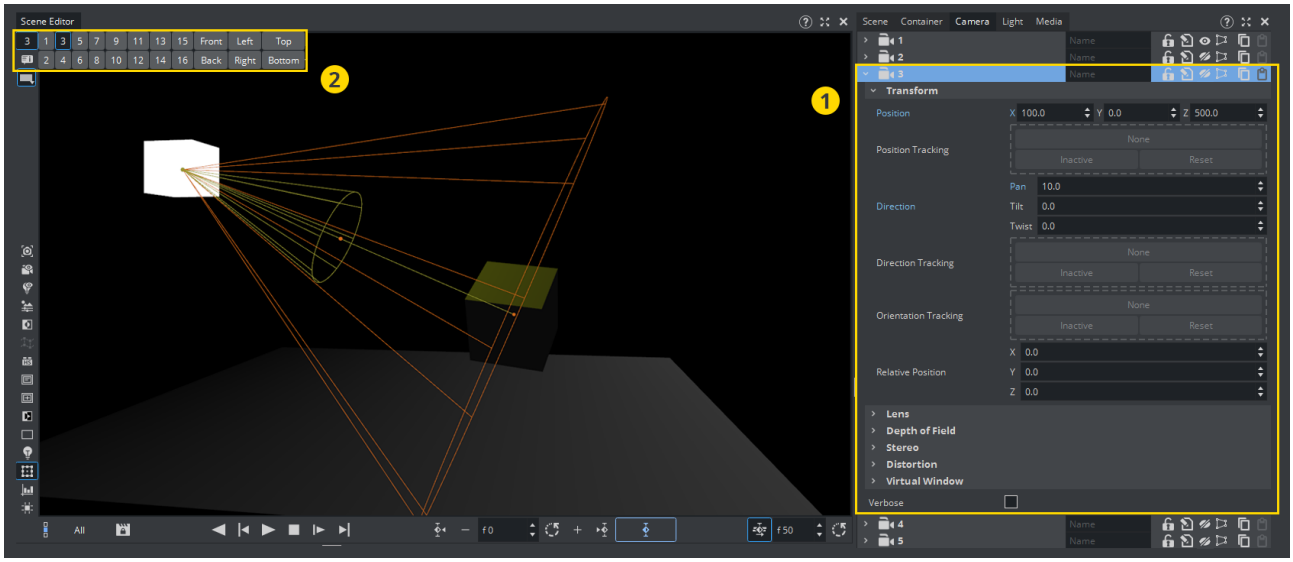
15.7.1 Camera Selection Panel Properties

Note: The position and direction of the 16 editable cameras can be defined in the Camera Editor Panel.

- **1-16:** Selects one of the 16 different cameras.
- **Front:** Shows output through orthogonal camera in front of the scene.
- **Back:** Shows output through orthogonal camera behind the scene.
- **Left:** Shows output through orthogonal camera at the left side.
- **Right:** Shows output through orthogonal camera at the right side.
- **Top:** Shows output through orthogonal camera above the scene.
- **Bottom:** Shows output through orthogonal camera below the scene.

A Scene can be viewed from different camera angles and views in the Scene Editor. You can select different angles or views using the Camera Editor (1) or the Camera Selection panel (2). Camera Selection is displayed in the Scene Editor.

Copyright © 2026 Vizrt



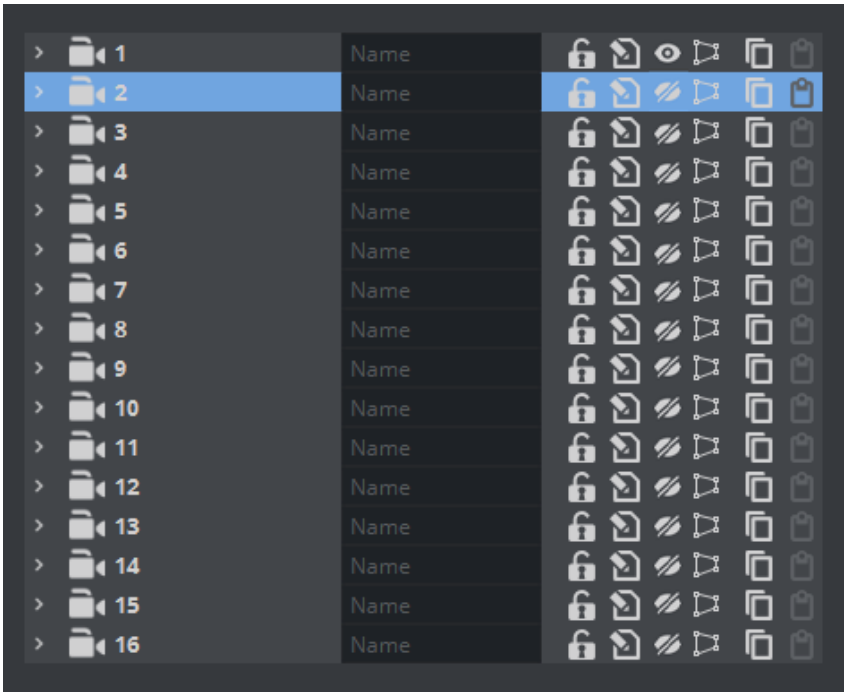
To Select a Different Camera Angle or View

1. Click the **Camera Selection** button (1), or open the Camera Editor.
2. Select a Camera angle or view:
   - In the Camera Selection panel, select a camera (1 to 16) or a camera view. To close the Camera Selection panel, click the **Camera Selection** button again.



   or
   - Select a camera in the Camera Editor,

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

15.8 Camera Animation

All camera parameters can be animated in the same way as Create Animations in general. The process is roughly as follows:

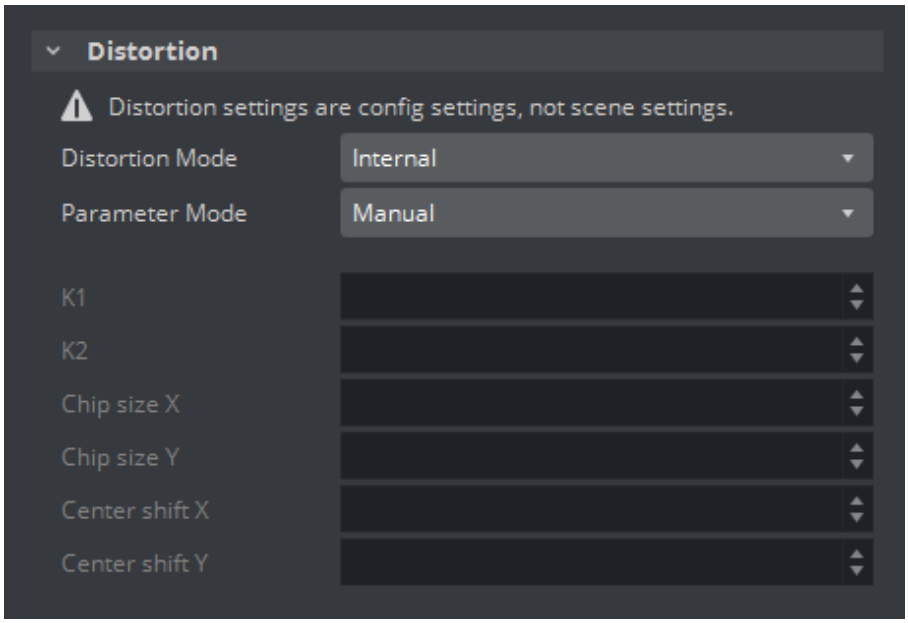
- Set a value in the Camera editor.
- Add a Key Frame.
- Change the value.
- Add another Key Frame.

Animations can also easily be created in one of the orthogonal views. There the cameras show as symbols and by repositioning them and adding Key Frames, animation sequences are created. For more information, see Change Camera Parameters in Orthogonal Views.

More advanced camera animations are normally created with the camera tracking feature, as in Track Objects with a Camera.

Copyright © 2026 Vizrt

15.9 Advanced Lens Distortion



Here, you find the following information:

- Important Changes Compared to Previous Versions
- Distortion Algorithms
- Parameter Providers
- Configuration File Flags
    - Lens Compatibility Mode

Lens distortion replaces the lens deformation, and was introduced with Viz Artist and Viz Engine 3.9. You can now select the distortion algorithm and distortion parameters provider for each individual camera. There is no longer a need for multiple configuration flags, such as `render_scale` and Virtual Studio.

The distortion algorithm defines how the rendered image is distorted, and the parameters are the input to the distortion algorithm. Typical algorithms expect *k1*, *k2*, *centershift_x*, *centershift_y*, *chipsize_x*, *chipsize_y*, etc.

15.9.1 Important Changes Compared to Previous Versions

- Activation of lens distortion does not depend on other settings, such as `render_scale` or entering Virtual Studio mode.
- You can explicitly select the algorithm and parameter source per camera, also in Viz Artist. There is an **auto** mode, which attempts to determinate the correct algorithm and parameter mode based on the available data.
- Now called **Lens Distortion** instead of **Lens Deformation**.
- `render_scale` is now correctly applied to FieldOfView. There is no longer a jump when enabling or disabling post processing (**PP**) in the Scene Editor.

Copyright © 2026 Vizrt

Technical note: In previous versions, FOV was adapted when data was received from the external camera. The received value was already modified to compensate the `render_scale` setting. This could cause wrong results when the setting was not considered for rendering, such as when disabling post processing effects with the **PP** button. In 3.9.0, the FOV is untouched and modified on the fly while rendering if necessary.

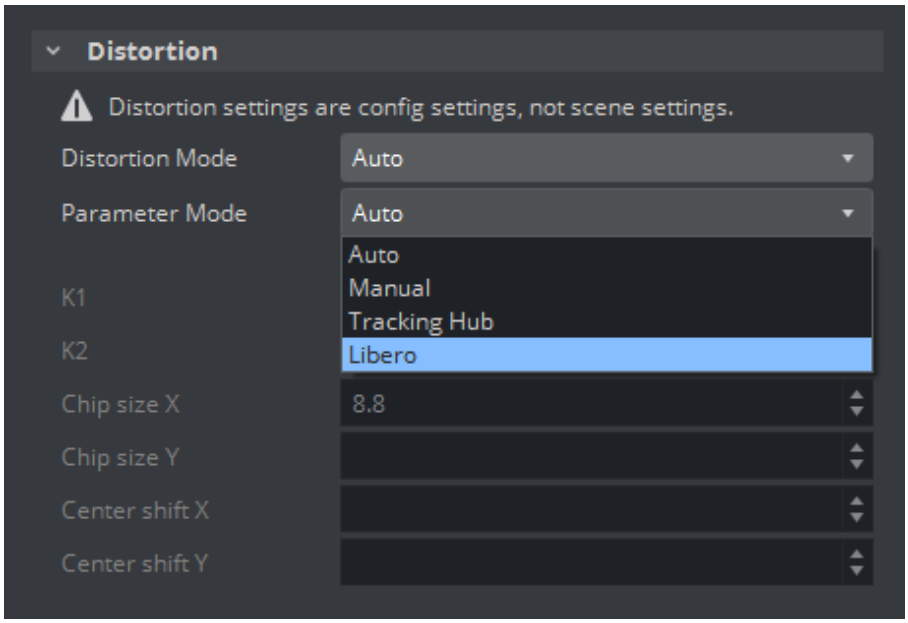
- Previous versions rejected distortion values if the distortion was too strong, if pixels from the outside of the rendered area would have been moved inside the visible area. The `render_scale` defines how much of the border around the visible area should be rendered, to be able to “move” parts from outside to the inside - think of the barrel distortion! If distortion values were too high, and non-available pixels from the outside would have been distorted into the screen, it simply used the last valid distortion parameters instead. This could lead to confusion, as the distortion did not use the real values. Now, the distortion uses the values provided, even if some invisible areas show up on the borders. If you see unexpected black or transparent areas around the border, consider increasing the `render_scale.` You can do this by editing the entry for **render_scale** in the configuration file. The default value is `0` , but you can change it as an example to `1.2` or even higher.
- New distortion algorithms:
  - **Radial:** A classic algorithm.
  - **Stype** and **Trackmen:** Implemented according to specifications from tracking data providers.

15.9.2 Distortion Algorithms

- **Distortion Mode**
  - **Inactive:** Does not use a distortion algorithm.
  - **Auto:** Automatically selects one of the algorithms below, depending on available data.
  - **Internal:** The internal distortion algorithm used when reading data from lens file. It uses a forward calculation, using a mesh with the undistorted render result applied as texture. The result then gets distorted by moving the vertices of the mesh. To calculate the distorted position of each vertex, the inverse formula as proposed in the Brown-Conrady model is used in an iterative approach. The distortion is normalized to the resolution.
  - **Libero:** The distortion algorithm used when connected to Viz Libero or Viz Arena.
  - **Xpecto:** The distortion algorithm used with external data, similar to the Internal approach but normalized to the chipsize.
  - **Radial:** Radial lens distortion, based on the Brown-Conrady model. See Software Correction.
  - **Stype:** Lens distortion as expected by Stype, based on the Brown-Conrady model. This method uses a backward calculation, rendering a full screen rectangle with modified texture coordinates, so that we look up in the undistorted texture for each pixel in the final output (distorted space).
  - **Trackmen:** Lens distortion as expected by Trackmen, based on the Brown-Conrady model. This method uses a similar approach like Stype.
  - **ImageBased:** The distortion algorithm used for image-based lens distortion calculations in Viz Sports products.

Copyright © 2026 Vizrt

15.9.3 Parameter Providers



- **Parameter Mode**
  - **Auto:** Automatically selects one of the providers below, depending on the received data.
  - **Libero:** Reads **K1/K2** from Viz Arena or Viz Libero.
  - **Manual:** Uses parameter from the configuration settings.
  - **Tracking Hub:** Reads parameters from Tracking Hub.

Note: Please refer to the Tracking Hub Administrator Guide for migrating old lens files (previously Parameter Mode: Lens File) to Tracking Hub.

15.9.4 Configuration File Flags

```
## Possible distortion modes: 0=INACTIVE, 1=AUTO, 2=INTERNAL, 3=XPECTO, 4=LIBERO,
5=RADIAL, 6=STYPE, 7=TRACKMEN
## Possible distortion param modes: 0=AUTO, 1=MANUAL, 2=LENSFILE, 3=Tracking Hub,
4=LIBERO
camera1_distortion_mode = 0
camera1_distortion_param_mode = 0
camera1_distortion_param_manual = {"centershift_x": 0.000000, "centershift_y":
0.000000, "chipsize_x": 8.800000, "chipsize_y": 5.200000, "k1": 0.000000, "k2":
0.000000}
camera2_distortion_mode = 0
camera2_...
# lens compatibility mode, see note
```

Copyright © 2026 Vizrt

```
use_lens_compatibility_mode=0
```

Important! For new lens calibrations, always use `use_lens_compatibility_mode=0`.

Lens Compatibility Mode

Until Viz Artist and Engine 3.8.2, the result of the lens deformation was affected by the **hd_halfheight** setting. This was corrected in 3.8.3, and the lens deformation now generates the same result for `hd_halfheight=1` and `hd_halfheight=0`. However, if you have lens files that were calibrated with `hd_halfheight=1` using an older version, the result now differs. In this case, enable `use_lens_compatibility_mode`.

Copyright © 2026 Vizrt

16 The Stage for Animation

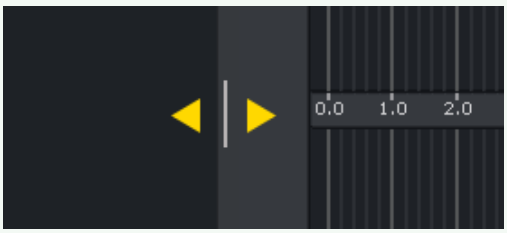
The Stage is the interactive user interface to create animations.



The Stage is divided in two parts:

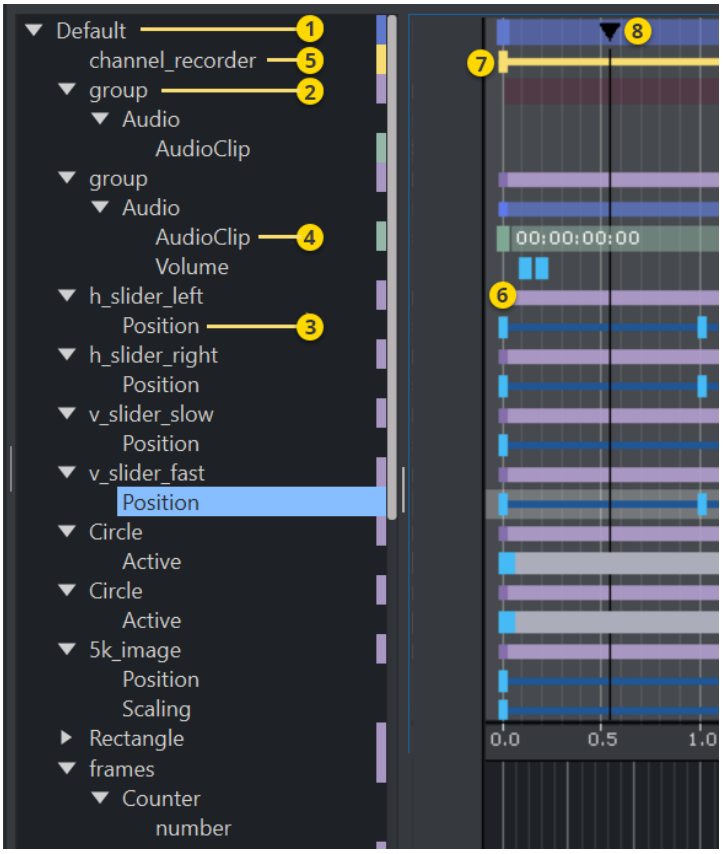
- Stage Tree Area **(1):** Shows a representation of all animated items in the current scene.
- Stage Editor **(2):** Shows the animated items as bars along the time-line.

Tip: To resize the two areas, drag the vertical separator.



16.1 Stage Elements

The Stage consists of the following elements:



1. Directors
2. Actors
3. Channels
4. Audio/Video Channels
5. Action Channels
6. Keyframes
7. Action Keyframes
8. Events

Elements are marked with unique colors (for example, yellow for action keyframes). Currently they can not be customized.

This section provides a detailed description of the following Stage for Animation topics:

- Channel Editor
- Time-line Marker
- Key Frame Editors
- Event Editor
- Stage Tree Area
- Stage Editor
- Time-line Editor
- Director Editor
- Director Control Panel
- Master Clip
- Actor Editor
- Dopesheet Editor
- Spline Editor
- Stage Object Editor

Copyright © 2026 Vizrt

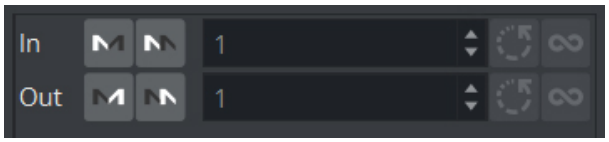
16.2 Channel Editor

The Channel editor shows details about the selected Channel.

This section contains information on the following topics:

- Common Channel Editor Properties
- Position Channel Editor Properties
- Action Channel Editor Properties

16.2.1 Common Channel Editor Properties



Note: Not available for four Point Loop clips.

IN:

- **Swing In:** Enables swing in.
- **Loop In:** Enables loop in.
- **Number of Swings/Loops:** Defines the number of swings/loops in.
- **Infinite:** Sets number of swings/loops in to infinite (from the start of the animation).

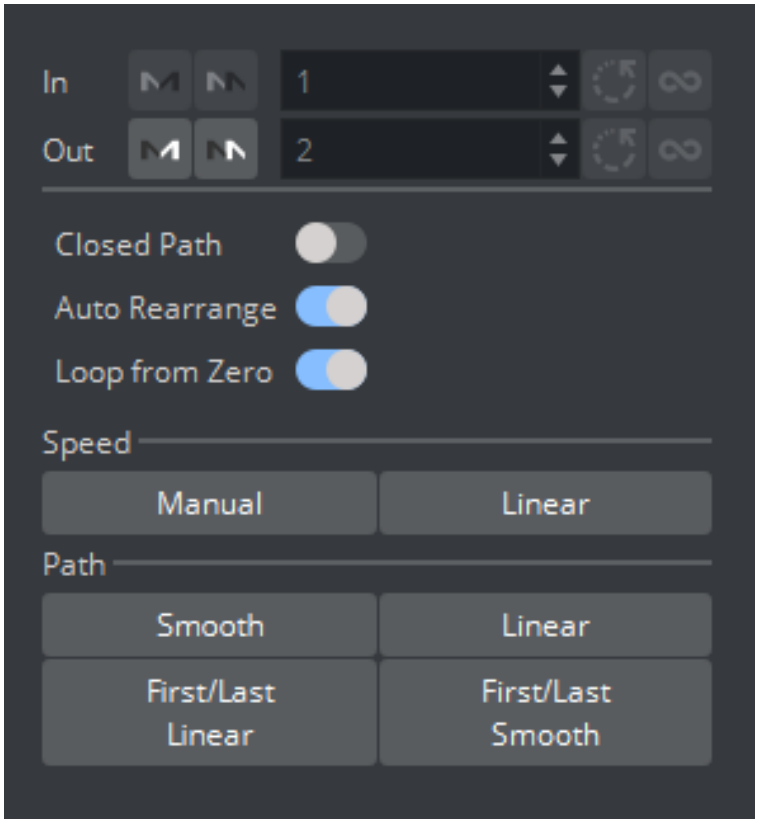
OUT:

- **Swing Out:** Enables swing out.
- **Loop Out:** Enables loop out.
- **Number of Swings/Loops:** Defines the number of swings/loops out.
- **Infinite:** Sets the number of swings/loops out to infinite (animation never ends).

16.2.2 Position Channel Editor Properties

In addition, The **Position Channel** editor has these extra parameters that can be edited:

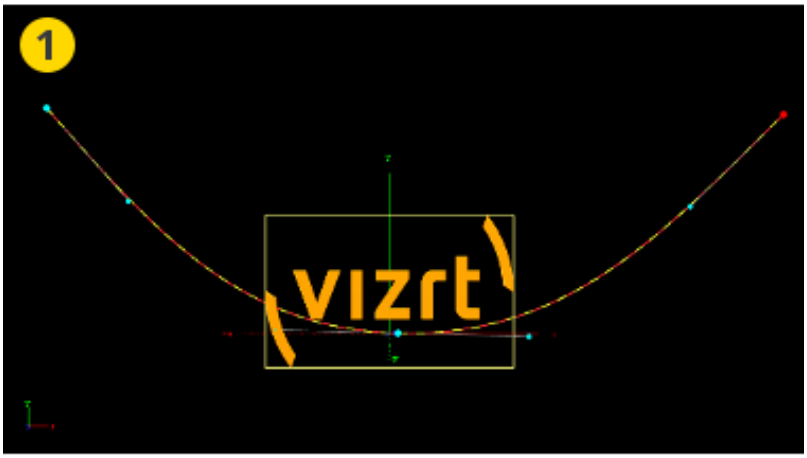
Copyright © 2026 Vizrt

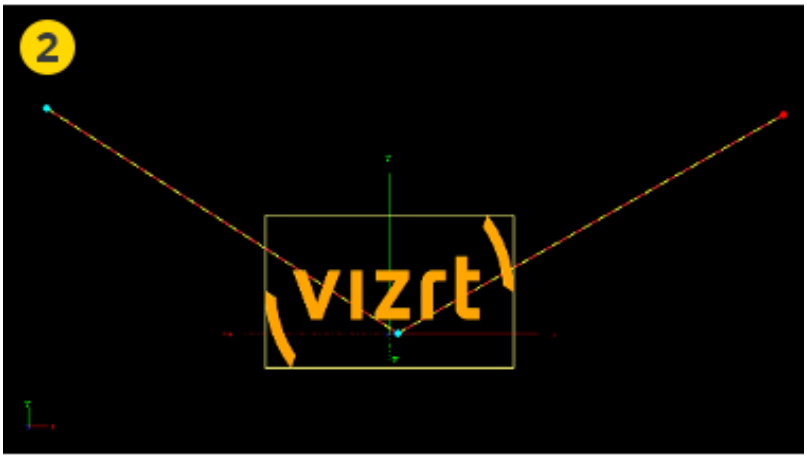


- **Closed Path:** Forces the animation path to go from the last Key Frame to the first when activated. When deactivated, the animation path remains open.
- **Auto Rearrange:** Ensures that the order of the Key Frames in the Dopesheet editor is the same as in the Scene Editor so that the animation only ever continues forward. When disabled, if you change the order of the Key Frames in the Dopesheet editor, the spline remains the same, but the animation may return from the previously last Key Frame to the newly added Key Frame now in last position. By default, this setting is enabled.
- **Loop from Zero:** Includes a gap in the loop (like 2.x behavior), if there is a gap before an animation loop, when enabled.
- **Speed**
  - **Manual:** Locks all Key Frames. There is no linear speed with this option. The duration between each Key Frame can be different, depending on the timing value and the transformation values between the different Key Frames. The result is a non-continuous animation speed.
  - **Linear:** Unlocks all Key Frames. The Key Frames between the first and last Key Frames are locked. If you change the value of either the first or last Key Frame, or any of the Key Frames in between, the timing changes automatically for each Key Frame. The result is an animation with a continuous speed between all Key Frames.
- **Path**
  - **Smooth** (1): Sets the animation path to be smooth.
  - **Linear** (2): Sets the animation path to be linear.
  - **First/Last Linear:** Sets the animation to be linear for the first and last Key Frame, and smooth for the rest.

Copyright © 2026 Vizrt

- **First/Last Smooth:** Sets the animation to be smooth for the first and last Key Frame, and linear for the rest.

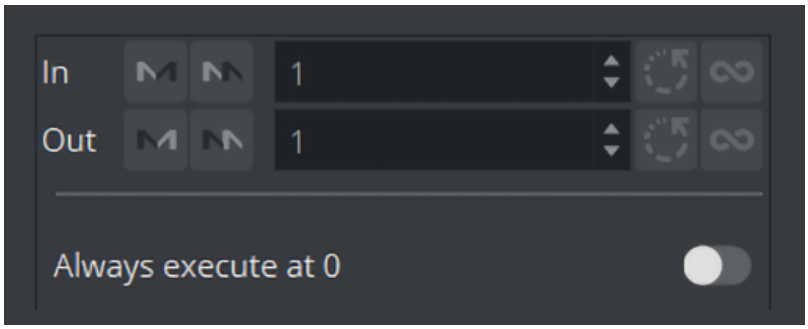




Note: These settings are global for the whole animation in question.

16.2.3 Action Channel Editor Properties

The Action Channel editor has, in addition, these extra parameters that can be edited:



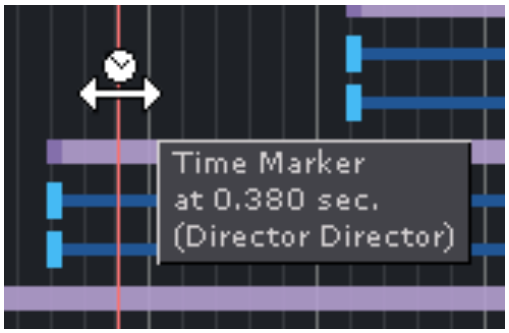
Copyright © 2026 Vizrt

- **Always Execute Action at 0:** If an action is set to execute at `0` on the time-line, press this button to make sure that the action is executed at `0`. If this button is not pressed (inactive), the action may not work.

Copyright © 2026 Vizrt

16.3 Time-line Marker

The time-line marker defines the current position along the time axis (X) in the Dopesheet and Spline Editor. The time-line marker is shown as a thin red vertical line.



A separate time-line marker is created for each Director in the Stage Tree. Dragging a time-line marker shows the animation in the Scene Editor accordingly.

If the Show Info option is enabled in the Stage Editor Bar, a tool tip with information about the current time and the belonging Director shows when holding the pointer over the time-line marker. If the Animate Time-line Marker option is enabled in the Stage Editor Bar, the time-line marker is animated in the Dopesheet while playing the animation.

This section contains information on the following procedures:

- To Manually Move the Time-line Marker
- To Move the Time-line Marker a Set Amount of Fields
- To Move the Time-line Marker a Set Amount of Fields with Shortcuts

16.3.1 To Manually Move the Time-line Marker

- Drag the time-line marker along the time-line in either the Dopesheet Editor or Spline Editor
  or
- Click an empty area of the Dopesheet Editor or Spline Editor.

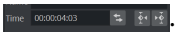
> **Note:** In complex animations with many directors, channels, and actors, there might not be an empty area to click. If this is the case, press **ALT** while clicking.

16.3.2 To Move the Time-line Marker a Set Amount of Fields

- In the Dopesheet Editor, enter the time (in fields) in the **Time-line** input box.
  OR

1. Enable the Stage Object Editor.
2. Select the Director in the Stage Tree or Dopesheet Editor.
3. In the Director Editor that shows, enter the time (in fields) in the **Time-line** text box.
   OR

Copyright © 2026 Vizrt

- Drag the Time-line in the Scene Editor Time-line editor .

16.3.3 To Move the Time-line Marker a Set Amount of Fields with Shortcuts

- **Move One Field at a Time:**
  - In the Dopesheet Editor or Director Editor click the **Time-line** up/down buttons, or
  - Press **CTRL** (or the configured shortcut) and the **LEFT** or **RIGHT ARROW** key on the keyboard.
- **Move More Than One Field at a Time:**
  - Press the **LEFT** or **RIGHT ARROW** key (or the configured ones) on the keyboard. This, for example, in 1080i50 format, moves the time-line marker five fields, left or right, and in 1080i60 the movement is six fields, left or right.
  - Press **ALT** and the **LEFT** or **RIGHT ARROW** key (or the configured ones) on the keyboard. This, for example, in 1080i50 format, moves the time-line marker 50 fields, left or right, and in 1080i60 the movement is 60 fields, left or right.

Note: The movement interval in fields, is format dependent.

Copyright © 2026 Vizrt

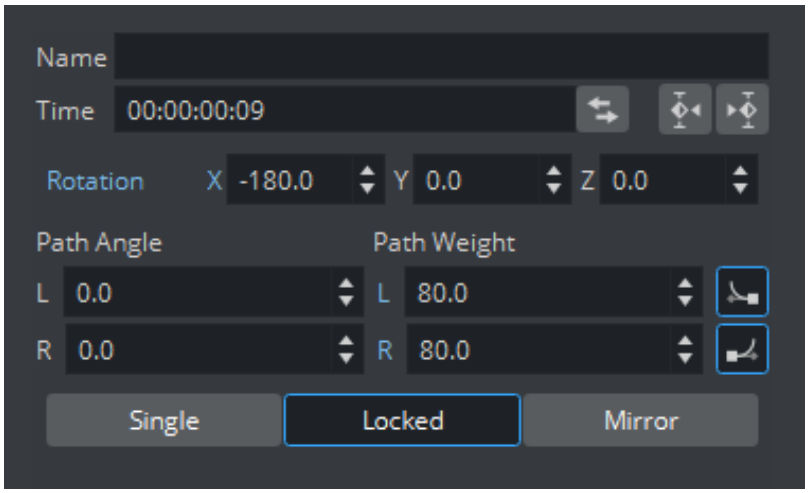
16.4 Key Frame Editors

The Key Frame editors show details about the selected Key Frame. To view the Key Frame editor click on a Key Frame. Make sure that the Stage Object Editor is enabled.

This section contains information on the following topics:

- Common Key Frame Editor Properties
- Locked and Unlocked Key Frames
- Action Key Frame Editor Properties
- Position Key Frame Editor Properties
- Rotation Key Frame Editor Properties
- Scale Key Frame Editor Properties
- Material Key Frame Editor Properties
- Clip Key Frame Editor Properties
- Adding Camera Events Using Key Frames

16.4.1 Common Key Frame Editor Properties



- **Name:** Defines the name of the Key Frame. This name is used for external commands or scripts.
- **Time:** Defines where along the time-line (in fields) the Key Frame is placed.
  - To switch the format between timecode and absolute values, press the button. The absolute values depends on which format you are working with (Seconds, Frames, Fields).
  - **Previous/Next buttons:** Selects the previous or next Key Frame along the time-line if the property has more than one Key Frame.
- **Value:** The rest of the Key Frame editor options vary according to which property is being animated.

Example: For scaling, it is possible to define the size of an object in direction X, Y, and Z.

Copyright © 2026 Vizrt

16.4.2 Locked and Unlocked Key Frames

By default the Key Frames in the position animation are locked. They store information about the position of the container along the X-, Y- and Z-axis, as well as the time the Key Frame passed by in the animation.

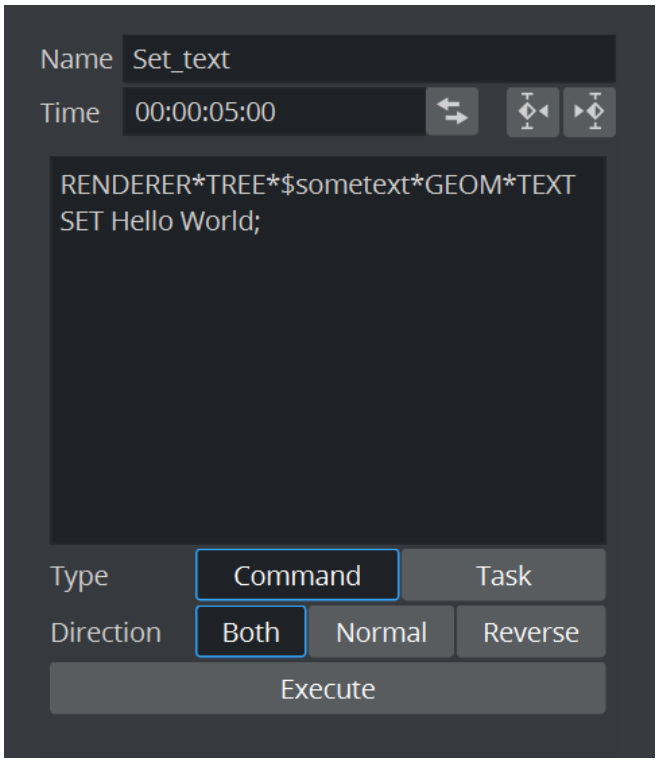
An unlocked Key Frame stores only information of the position along X-, Y- and Z-axis, without the information at which time the Key Frame should be passed. To reflect this, in the object editor for a locked Key Frame the **Time** is grayed out. Furthermore, as the spline editor shows time and path-percentage, the unlocked Key Frames are not shown there. In the animation editor, unlocked Key Frames are shown at half height of locked Key Frames. Viz Artist calculates the time the Key Frame is passed to fit with the handles set in the next locked Key Frame before and after the unlocked Key Frame.

To better understand the difference between locked and unlocked Key Frames, create a simple position animation. You see the container moving with different speeds between the Key Frames. Next unlock all Key Frames and play the animation once more. The container moves evenly during the whole animation.

Use an unlocked Key Frames to define the path without editing the time for every Key Frame. This can save time.

16.4.3 Action Key Frame Editor Properties

In addition to the common key frame editor properties, the Action Key Frame Editor defines the Action Channel. The action works when the Director time-line equals the Action Channel Key Frame.



- **Command:** Executes the command or OS task entered here.
- **Type:** Initializes action Key Frames for two main types of actions:

Copyright © 2026 Vizrt

    - **Command:** Use Viz Artist internal commands to execute a number of automated tasks in the program (for Viz Artist internal commands documentation go to: *<viz install folder>\Documentation\viz-cmd-doc-webhelp\index.html*)
    - **Task:** Enter a task command. Any relevant command which can be started from a command line window (*cmd.exe*) can be executed as a task. If the path to the executable file contains spaces, set the complete path in quotation marks, for example, "*C:\program files\microsoft office\office\winword.exe*".
- **Dir:** Sets which way, forwards or backwards, the animation should go to trigger the Action Key Frame:
    - **Both:** The Key Frame is triggered when the animation goes forward and in reverse.
    - **Normal:** The Key Frame is triggered when the animation goes forwards.
    - **Reverse:** The Key Frame is triggered when the animation goes backwards.
- **Execute:** Starts the action.

Tip: Multiple commands can be executed in the Action Key Frame Editor. The commands must be terminated by a `;` (semi-colon) and on a new line. For example:

```
RENDERER*TREE*$sometext*GEOM*TEXT SET Hello World;
MAIN_SCENE*STAGE*DIRECTOR*$Default GOTO_TRIO $B $A;
```

Examples of Commands for Action Key Frames

Here are some examples of Viz Artist internal commands, which can be used in the Action Key Frame Editor to create actions:

- **THIS_SCENE:** The created action acts on this Scene only.
- **THIS_EDITOR:** The created action acts on this Editor only.
- **THIS_DIRECTOR:** The created action acts on this Director only.
- **THIS_PARENT_DIRECTOR:** The created action acts on this Parent Director only.

Example of Invoking a Script Function:

```
THIS_SCENETREE$[name of the container]*SCRIPT INVOKE [your function name/
sub routine name in the script]
```

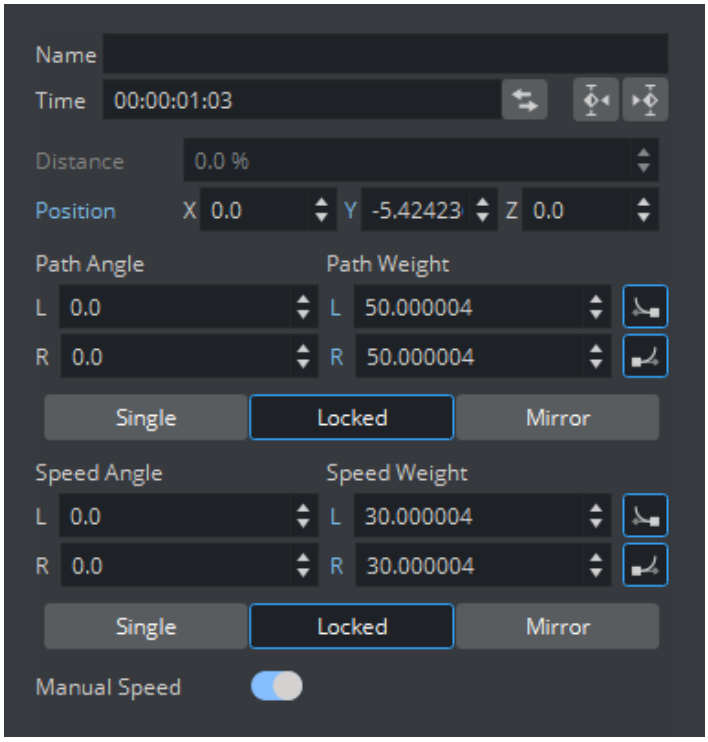
Example of Setting a Plug-in Value:

```
THIS_SCENETREE$[name of the container where the plug-in
resides]FUNCTION[name of plug-in]*[name of UI item of the plug-in, (the
button or the drop down list)] SET [value]
```

16.4.4 Position Key Frame Editor Properties

In addition to the Common Key Frame Editor Properties, a Position Key Frame contains information about the position of a Container at a specific time. The parameters of Position Key Frames are:

Copyright © 2026 Vizrt



- **Distance:** Defines the distance of a selected Key Frame (value in percent) between the first and last Key Frame of the animation (information only).
- **X:** Defines the position along the X axis.
- **Y:** Defines the position along the Y axis.
- **Z:** Defines the position along the Z axis.

Path Control

- **Angle:** Defines the angle of the handles for the path.
- **Weight:** Defines the weight of the handles for the path.
- **Smooth Left/Right:** Allows the handles for the path to be smooth or linear (see Channel Editor).
- **S / L / M:** Defines the way handles for the path should be modified:
  - **S (Single):** Only the left or right parameter can be adjusted.
  - **L (Locked):** Left and right parameters are locked in their current position and adjustment changes both left and right parameters together.
  - **M (Mirror):** The left and right parameters are mirrored when adjusted

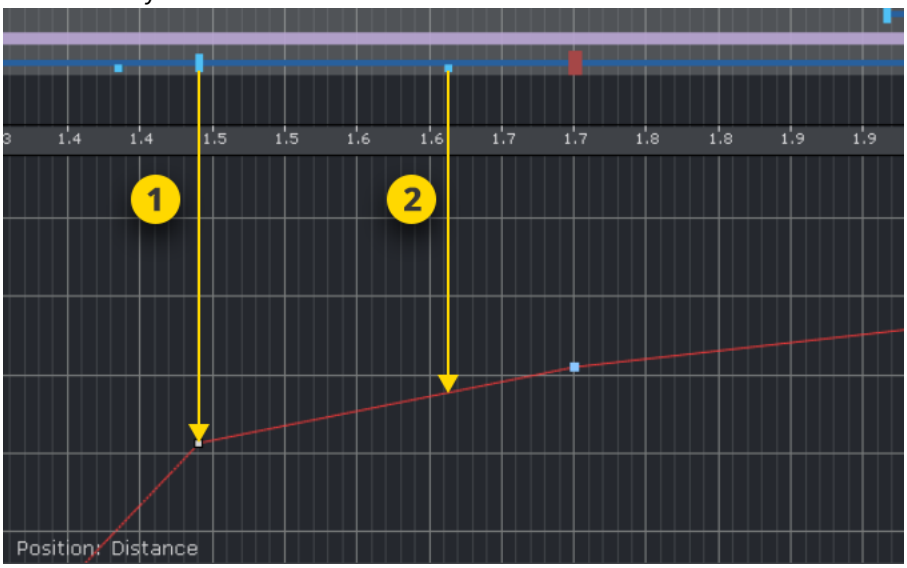
Speed Control

- **Angle:** Defines the angle of the handles for the speed.
- **Weight:** Defines the weight of the handles for the speed.
- **Smooth Left/Right:** Allows the handles for the speed to be smooth or linear.
- **S / L / M:** Defines the way handles for the speed should be modified:
  - **S (Single):** Only the left or right parameter can be adjusted.

Copyright © 2026 Vizrt

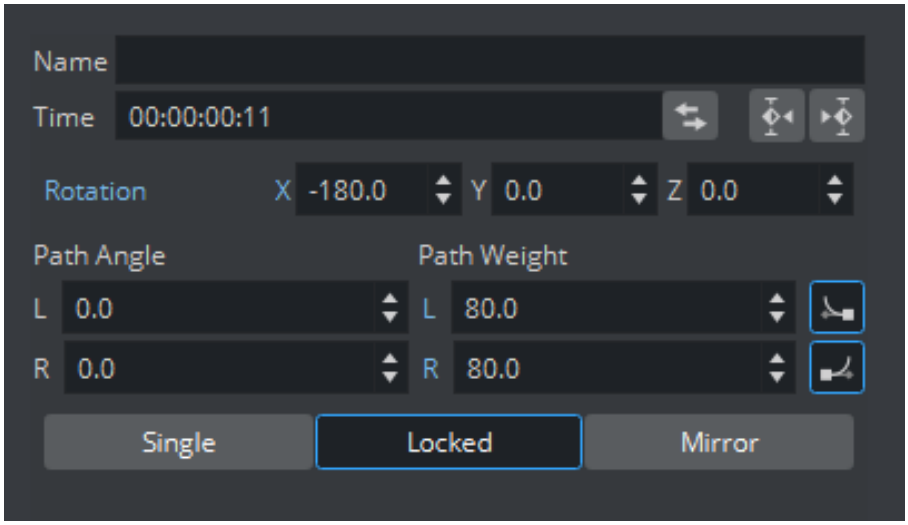
- **L (Locked):** Left and right parameters are locked in there current position and adjustment changes both left and right parameters together.
- **M (Mirror):** The left and right parameters are mirrored when adjusted

- **Manual Speed:** Toggles the Key Frame Active or Inactive:
  - **Active:** Unlocks Speed Control (1). The speed between the previous, selected, and the next Key Frame is calculated by the position and the speed control of this Key Frame in the time-line.
  - **Inactive:** Locks Speed Control (2). The speed between the previous and the next Key Frame is calculated linear between those two and cannot be modified. If the value of either the previous or next Position Key Frame is changed, or any of the Position Key Frames, the timing is adjusted automatically for each locked Key Frame. The result is an animation with a constant speed between all locked Key Frames.



16.4.5 Rotation Key Frame Editor Properties

A rotation Key Frame contains information about the container rotation. In addition to the default options for a rotation Key Frame it is possible to define specific rotation options.

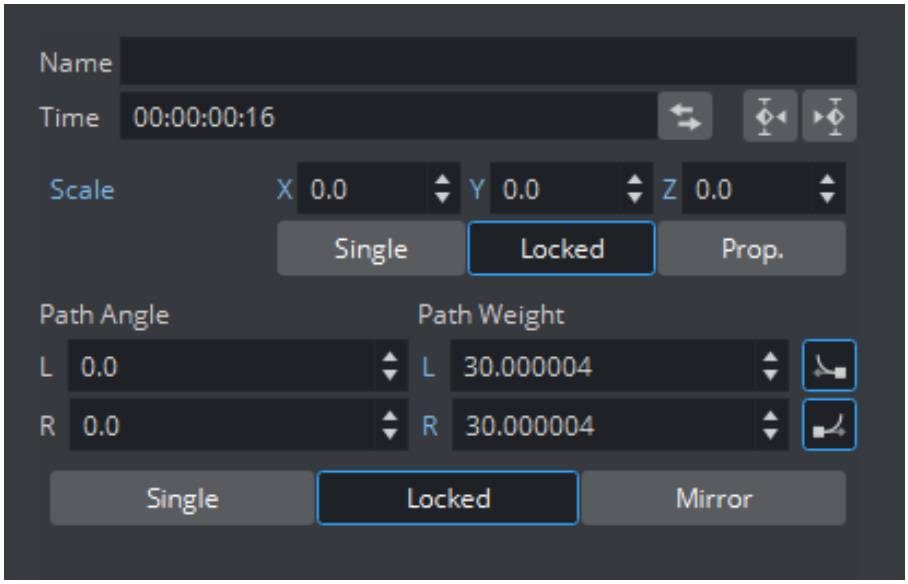


- **X:** Defines the rotation along the X axis.
- **Y:** Defines the rotation along the Y axis.
- **Z:** Defines the rotation along the Z axis.
- **Angle:** Defines the angle of the handles for the path or speed.
- **Weight:** Defines the weight of the handles for the path or speed.
- **Smooth Left/Right:** Allows the handles for the path to be smooth.
- **Single (S)**, **Locked (L)** and **Mirror (M):** Defines the way handles for the path or speed should be modified. In the spline editor, three different splines are shown, one for every axis. The red one is for the X-axis, the green one for the Y-axis and the blue one for the Z-axis. As you can only edit the currently selected spline, you must use three spline buttons at the top of the Dopesheet Editor (default shortcuts: **1**, **2** or **3**) to select the spline you want to edit. For ease of use, in the lower left corner you find an indication of which axis is currently selected.

16.4.6 Scale Key Frame Editor Properties

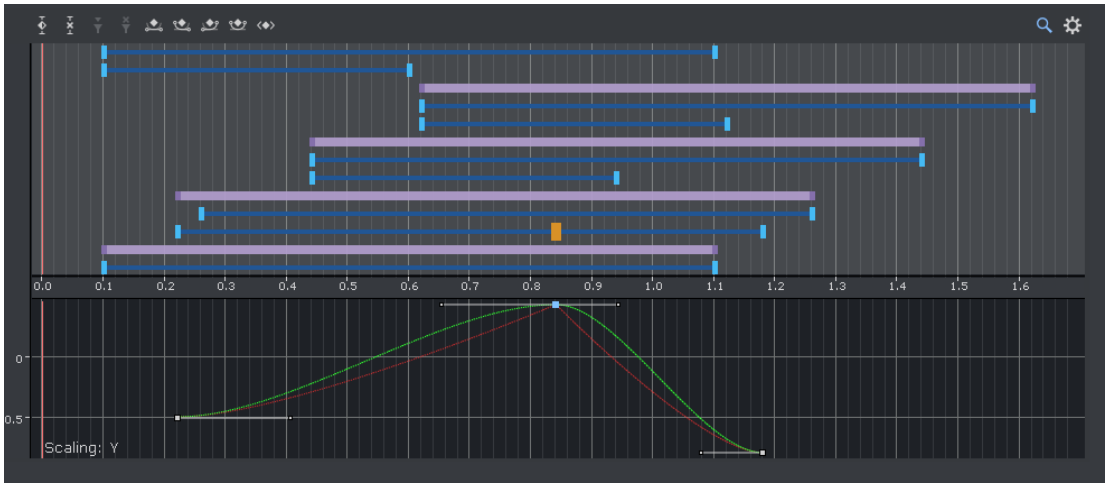
Scale Key Frames have to do with the scaling of the object in the animation. For a scale Key Frame, in addition to the default options, it is possible to define the scaling along the X, Y, and Z axes in the three input boxes.

Copyright © 2026 Vizrt



If you have animated the scale, you can scale along up to all three axes. To change all three scale values, in the object editor you find an input box for **X**, one for **Y** and one for **Z**. There you can change if the values should be **Locked** or if you change them proportional (**Prop**) or each **Single** value. In the spline editor, three different splines are shown (when **Prop** or **Single** is selected), one for every axis. The red one is for the X-axis, the green one for the Y-axis and the blue one for the Z-axis. As you can only edit the currently selected spline, you must use three spline buttons at the top of the Dopesheet Editor (default shortcuts: **1**, **2** or **3**) to select the spline you want to edit. For ease of use in the lower left corner you find an indication of which axis is currently selected.

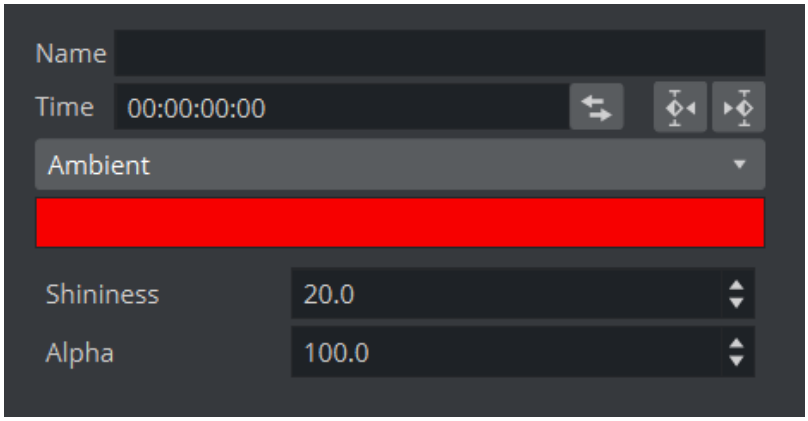
Scale Axes, X, Y and Z



16.4.7 Material Key Frame Editor Properties

With a Material Key Frame, in addition to the default options, it is possible to define color.

Copyright © 2026 Vizrt

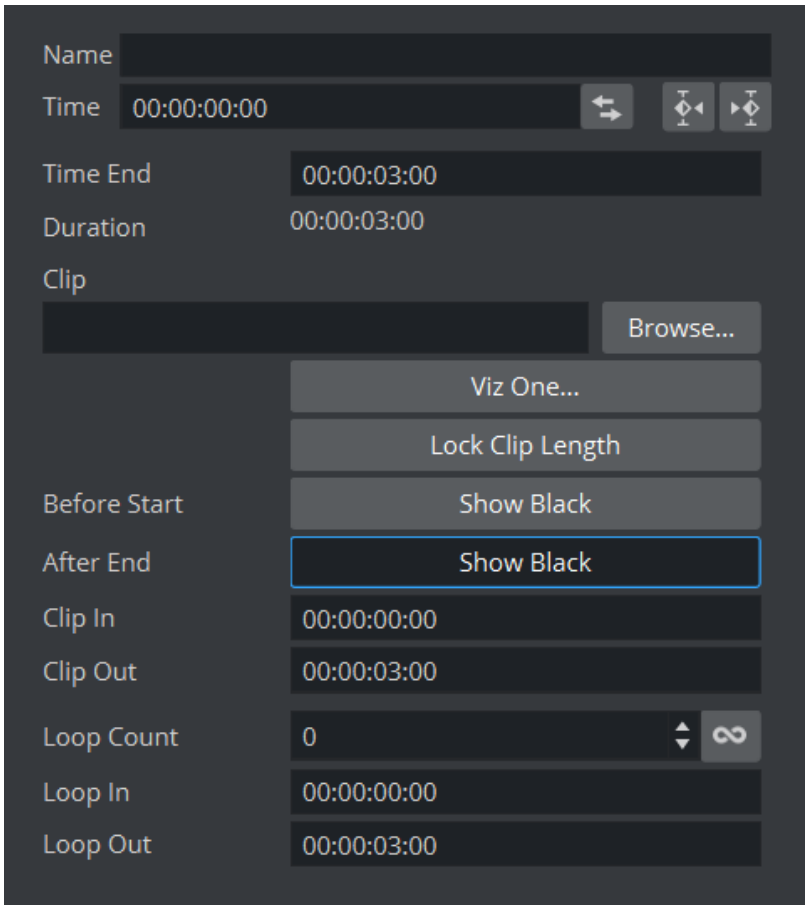


If a material has been animated, the object editor for the Key Frames shows a Material Editor. This allows you to conveniently the change the material. In the spline editor, two different splines are shown. The red one is for the alpha value (1), the green one for the shininess (2). As you can only edit the Key Frames/handles of one spline, you must use the number keys in the Stage Editor Bar to select the required spline. For ease of use in the lower left corner you find an indication of which parameter is currently selected.



Copyright © 2026 Vizrt

16.4.8 Clip Key Frame Editor Properties



- **Name:** Sets a name for the Clip Key Frame.
- **Time/Time End:** Sets the start and end time, in fields, of the Key Frame.
- **Duration:** Sets the time duration of selected clip Key Frame as defined by **Time/TimeEnd.**
- **Clip:** Selects a clip to use when "Browse" is clicked.
- **Viz One:** Selects a clip from Viz One (see Transfer Clips From Viz One).
- **Lock Clip Length:** Locks automatic length adjustment. Keyframe maintains length for set clips. Useful when exporting the scene after making adjustments to the clip length in the stage, as this keeps the clip length as designed, and not revert to the length of the clip file.
- **Before Start:** Sets what should happen before the clip starts to play within the animation. If the button is **disabled**, it shows the in-frame. If the button is **enabled**, it shows black until the animation gets to the clip.
- **After End:** Sets what should happen after the animation leaves the clip Key Frame. If the button is **disabled**, it shows the out-frame. If the button is **enabled**, it shows black.
- **Clip In/Out:** Sets the time the clip comes in and goes out.

Copyright © 2026 Vizrt

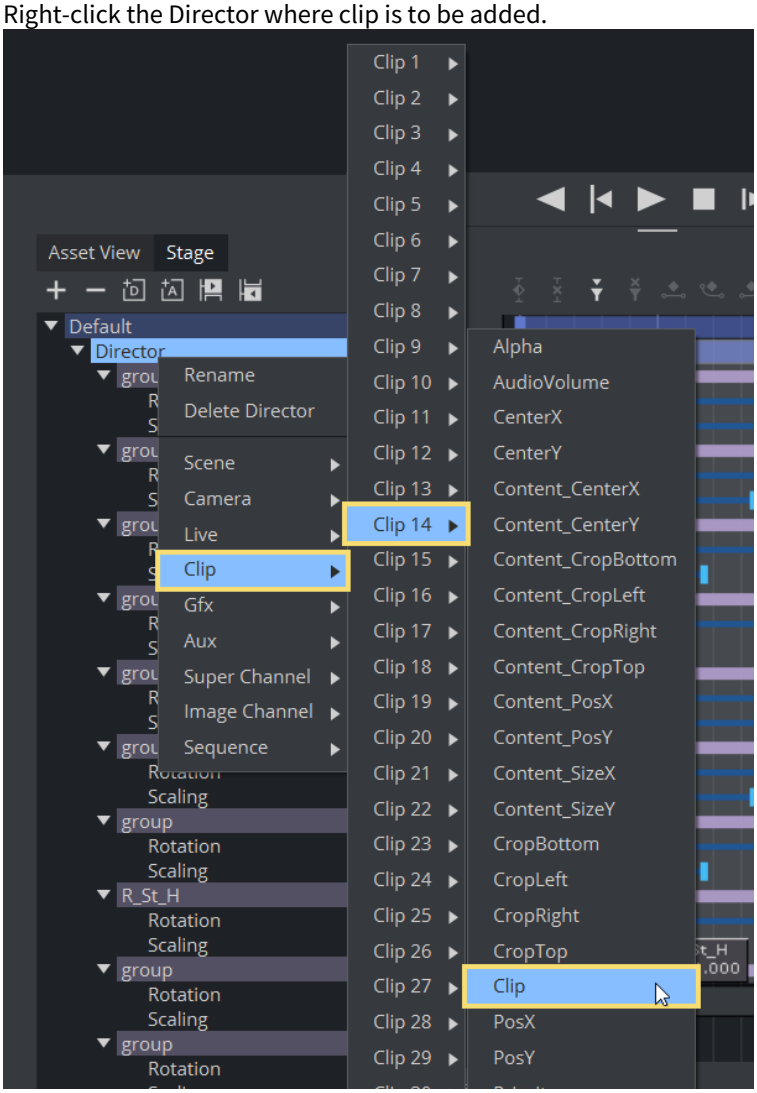
Example:
Time = 50
Time End = 250
Clip Length = 4 sec (50i clip)
Clip In = 00:00:01:00
Clip Out = 00:00:03:00

The clip playback starts one second after the Director starts. The clip starts at frame 25 and ends at frame 75. Because the Key Frame duration is twice as long, the clip part is played twice.

- **Loop Count**: To repeat the clip *n* times.
- **Loop In and Loop out:** The in and out times on 3- and 4-point loops.

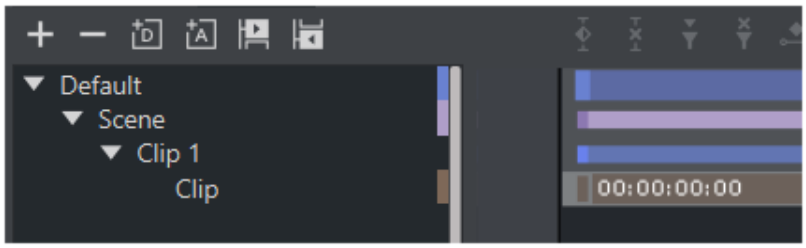
To Add a Clip to the Stage

1. Right-click the Director where clip is to be added.

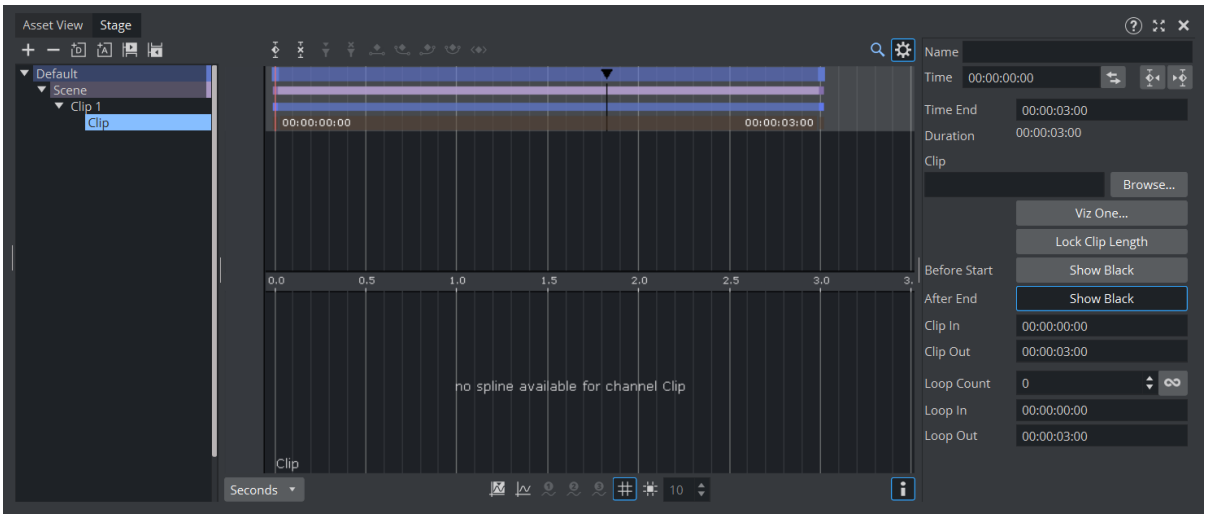


2. Select **Clip <number>** from the **Clip** menu, then choose **Clip**. To open the clip Key Frame editor, click the Clip Key Frame in the Dopesheet.

Copyright © 2026 Vizrt



3. To define the clip, you can either browse for a local clip, drag a clip from the Graphic Hub or select a clip from Viz One.

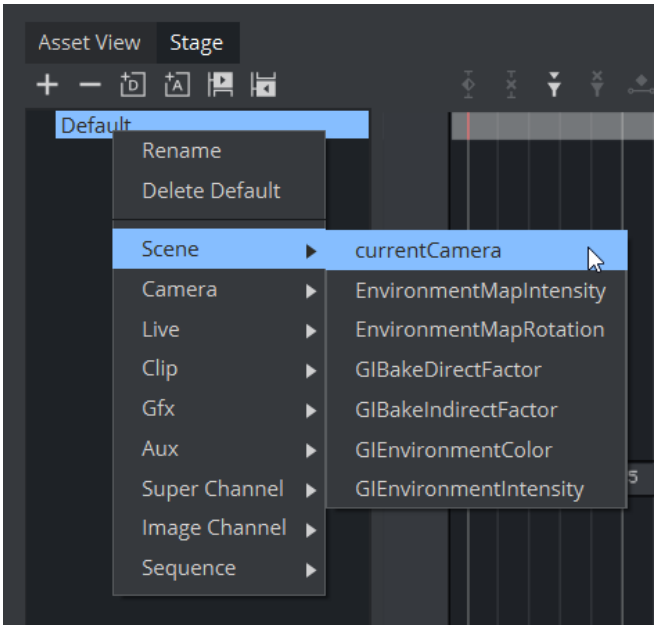


16.4.9 Adding Camera Events Using Key Frames

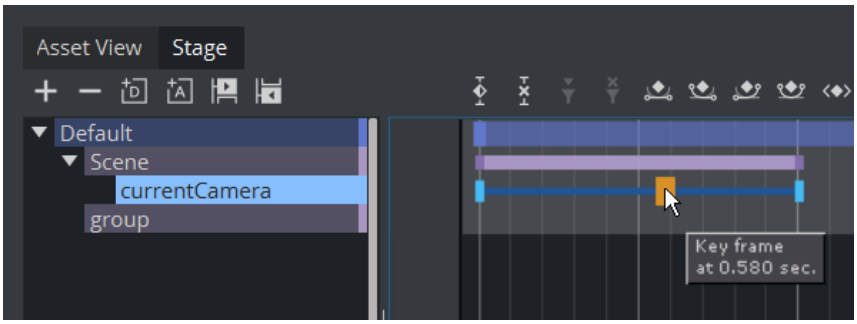
The Stage Editor also allows the designer to change the active camera in a scene by using key-frames.

To Add a Basic Camera Switcher

1. In the **Stage Tree**, right-click a Director and select **currentCamera** from the context menu. This adds a channel called **Scene** with **currentCamera** as an actor, with two key-frames.



- Select the first key-frame. In the Actor Editor, enter a descriptive **Name** and a **Value** for the initial camera position. The default value is `1`, which translates into Camera 1.
- Select the second key-frame. Enter a descriptive **Name** and a **Value** for the second camera position in the Actor Editor.

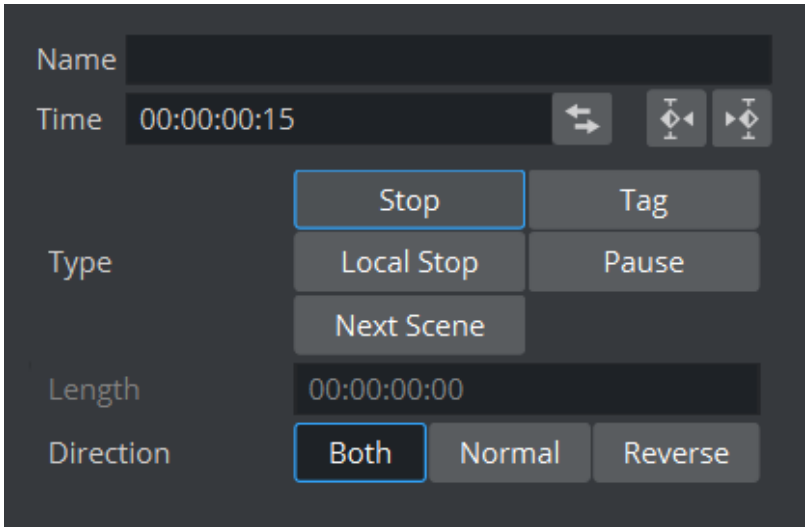


If you play this animation, Frames 0 up to 20.600 shows camera 1, then switches to camera 2 etc....

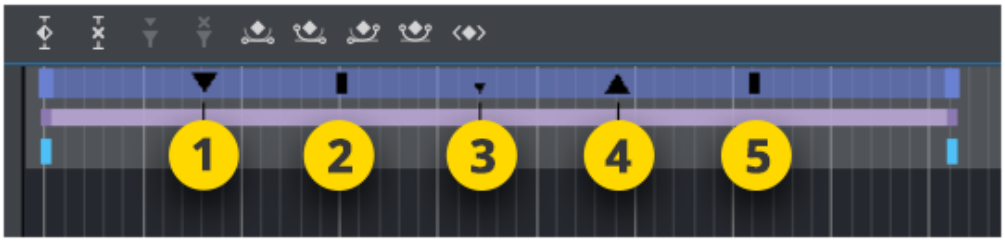
Copyright © 2026 Vizrt

16.5 Event Editor

The Event Editor shows details about the selected Stop, Tag, Pause, Local Stop or Next Scene Events.



- **Name:** Sets a name for the Event. This name is used for external commands or scripts.
- **Time:** Sets where in the time-line (in frames) the Event is placed.
- **Type:** Defines the type of Event used:
  - **Stop (1):** Stops the animation until continue animation is clicked. A stop applies for the Director and all subdirectors.
  - **Tag (2):** Shows a position along the time-line. This Event type does not stop the animation.
  - **Local Stop (3):** Stops the animation until continue animation is clicked. A local stop only applies for the Director it is placed on, not its subdirectors.
  - **Pause (4):** Stops the animation for a specified amount of frames (set in **Length**).
  - **Next Scene (5):** Marks a position where a Scene ends. The next Scene starts automatically.



- **Length:** Sets the length (frames) of a Pause.
- **Dir:** Sets the direction where the Event is valid:
  - **Both:** The Event is valid to both normal (forwards) and reverse directions.
  - **Normal:** The Event is only valid when the animation direction is normal (forward).
  - **Reverse:** The Event is only valid when the animation direction is reversed.

Copyright © 2026 Vizrt

16.6 Stage Tree Area

The Stage Tree area shows a representation of all animated items in the current scene.

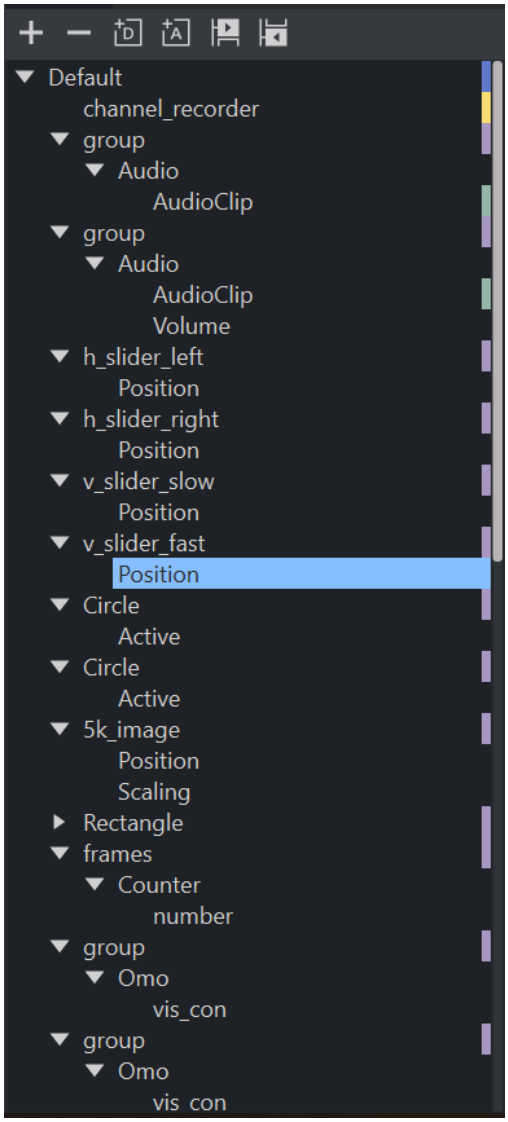
This section contains information on the following topics:

- Stage Tree
  - Expand or Collapse Stage Tree Branch
- Stage Tree Bar
- Stage Tree Menu
  - Directors
  - Actors
  - Channels
  - Action Channels

16.6.1 Stage Tree

The Stage Tree shows an overview of all Directors, Actors, and Channels that are used to create the animations in the current scene.

Copyright © 2026 Vizrt



Expand or Collapse Stage Tree Branch

Click on the arrow to the left of the item to expand or collapse a branch.

- **Expand a Stage Tree Branch:** Click the **Expand** icon.
- **Expand all Stage Tree Branches:** Click the top **Expand** button or press **ALT** and click the **Expand** icon to the left of the branch name.
- **Collapse a Stage Tree Branch:** Click the **Collapse** icon to the left of the branch name.
- **Collapse all Stage Tree Branches:** Click the top **Collapse** button.

16.6.2 Stage Tree Bar

The Stage Tree bar is located above the Stage Tree.

Copyright © 2026 Vizrt



The Stage Tree bar contains of these options:

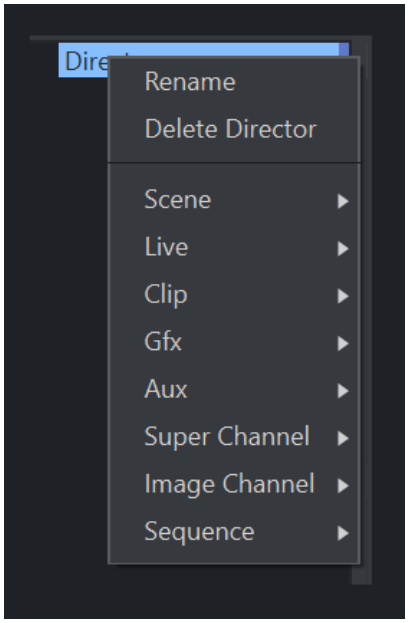
- **Expand / Collapse:** Expands or collapses all branches in the Stage Tree.
- **Add New Director:** Adds a new Director to the Stage Tree. The icon can also be dragged to add a new Director on a specific position.
- **Add New Action:** Adds a new action to a Director.
- **Jump to Next / Previous Animation:** Selects the next or previous channel in the Stage Tree if an Actor (Container) has multiple animation channels placed on various Directors.

16.6.3 Stage Tree Menu

This section contains information on these topics:

Directors

Right-click a Director to open a context menu. For more information about Directors, see Directors.



- **Rename:** Renames the Director.
- **Delete "DirectorName":** Deletes the Director.
- **Scene:** Adds a camera channel for camera cuts.
- **Live:** Adds any of the eight Live Video Inputs.
- **Clip:** Adds a Clip Channel.
- **Gfx:** Adds a GFX Channel.
- **Super Channel:** Adds a Super Channel.
- **Image Channel:** Adds an Image Channel.

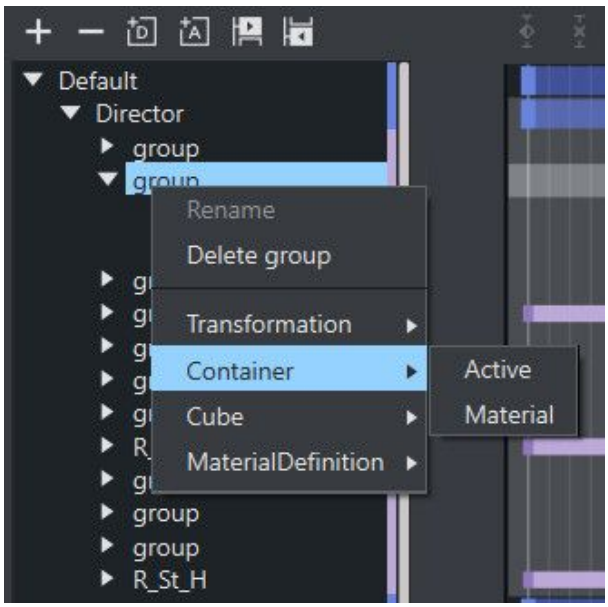
Copyright © 2026 Vizrt

- **Sequence:** Adds a Sequence.

Note: Directors can be grouped and ungrouped by using **CTRL + D** and **CTRL + SHIFT + D**.

Actors

Right-click an Actor in the Stage Tree to open its context menu (see also Actors).



- **Delete Actor:** Deletes the actor.
- **Transformation:**
  - **Position:** Creates a position channel for the actor, and automatically adds two default Key Frames.
  - **Rotation:** Creates a rotation channel for the actor, and automatically adds two default Key Frames.
  - **Scaling:** Creates a scaling channel for the actor, and automatically adds two default Key Frames.
- **Container**
  - **Active:** Creates an active channel for the actor, and automatically adds two default Key Frames. The active status can be modified in the Key Frame Editors. If an active Key Frame is set to Value= `Off`, the whole actor (container) including all animations are hidden in the Scene Editor. The action remains hidden until another Key Frame shows in the time-line with Value= `On`.
  - **Material:** Creates a material channel for the actor, and automatically adds two default Key Frames. The material can be modified in the Key Frame Editors.
- **Texture:** This option varies according to the item that is available on the actor.

Note: If a transformation channel is disabled (grayed out), the channel is already available for the actor in this Director.

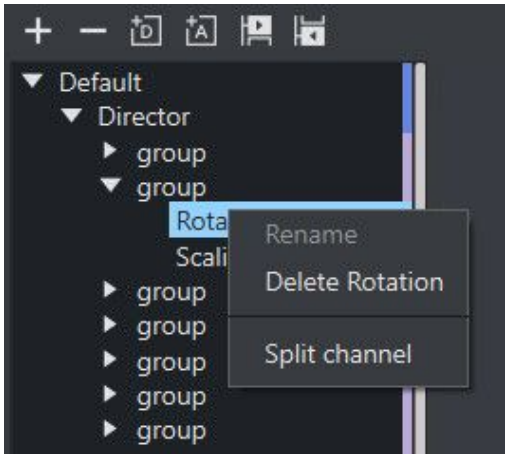
Note: The material option is only available if a material is added to the container.

Copyright © 2026 Vizrt

Example: If an image is added to the container, various texture options become available. If a built-in geometry is added, for example a cube, various cube options become available. If a font is added to the container, various text options become available. For details about options for the various items, see the respective editor sections, for example Image Editor and Fontstyle Editor.

Channels

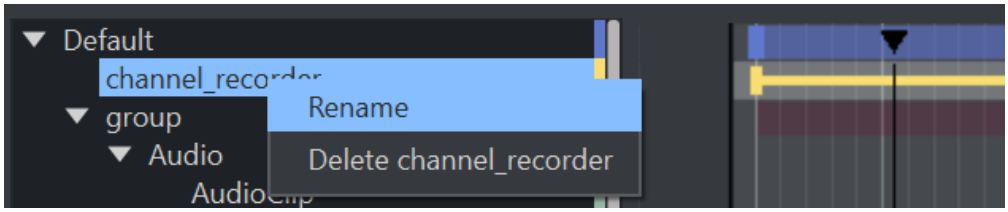
Right-click a channel, in the Stage Tree, to open the channel context menu (see also Channels).



- **Rename:** Not available.
- **Delete Channel:** Deletes a channel.
- **Split/Unsplit Channel:** Shows or closes Transformation Channel parameters.

Action Channels

Right-click an Action Channel, in the Stage Tree, to open the Action Channel context menu (see also Action Channels).

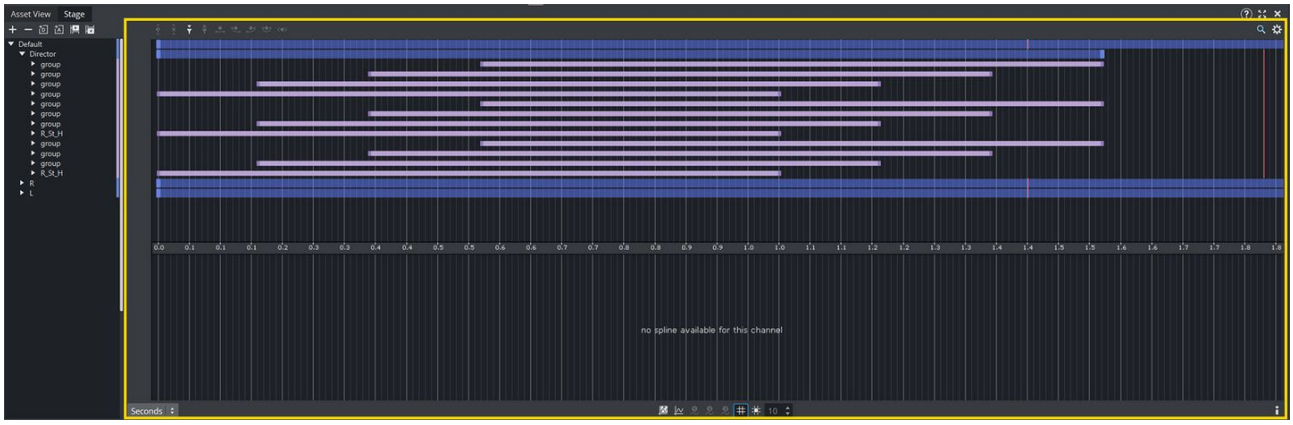


- **Rename:** Makes it possible to rename an Action Channel.
- **Delete Action Channel:** Deletes an action channel.

Copyright © 2026 Vizrt

16.7 Stage Editor

The Stage Editor area shows the animated items as bars and splines along the time-line.

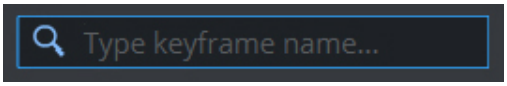


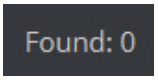
16.7.1 Stage Editor Bar

The Stage Editor bars are located above and below the Dopesheet Editor/Spline Editor. All modifiers and search options are located on the top:

- **Add Key Frame:** Adds a Key Frame at the current time/frame/field (see Key Frames).
- **Delete Key Frame:** Deletes the selected Key Frame.
- **Add Stop/Tag:** Adds a stop/tag Event to the selected Director at the current time/frame/field.
- **Delete Stop/Tag:** Deletes the selected stop/tag Event.
- **Linear Left and Right:** Sets the left and right spline handles in the selected Key Frame to linear.
- **Smooth Left and Linear Right:** Sets the spline handles in the selected Key Frame to smooth left and linear right.
- **Linear Left and Smooth Right:** Sets the spline handles in the selected Key Frame to linear left and smooth right.
- **Smooth Left and Right:** Sets the left and right spline handles in the selected Key Frame to smooth.
- **Handles Tangential:** Sets the left and right spline handles in the selected Key Frame to tangential.
- **Search:** Opens the Search popup window.

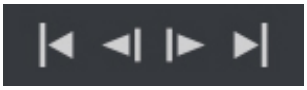
Search Options

- 

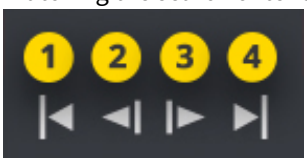
  Search: Searches the name of a Key Frame.
- 

  Items Found: Shows how many items were found.

Copyright © 2026 Vizrt

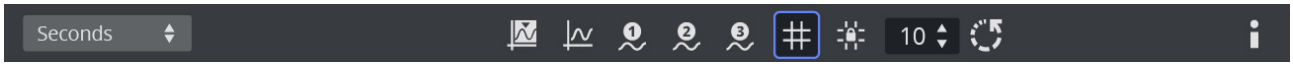
- 

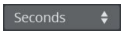
  Search Results: The arrow buttons can be used to search for the first, next, previous and last Key Frame matching the search criteria:

- 

  1: **Previous** Key Frame
  2: **Next** Key Frame
  3: **First** Key Frame
  4: **Last** Key Frame

The look and behavior of the Stage Editor is located on the bottom:



- 

Seconds/Frames/(Fields): Switches the time scaling between seconds, frames, and fields.

Note: *Fields* is only available in interlaced formats. It is not available in progressive formats.

- **Zoom to Standard:** Zooms the Dopesheet Editor and Spline Editor to standard view.
- **Zoom to Whole Spline:** Zooms the Dopesheet Editor and Spline Editor to fit the animation time-line of the current Channel.
- **Show Info:** Shows time-line information as a tool tip when holding the pointer over an item in the Dopesheet or Spline Editor, for example a Key Frame, when enabled.
- **Show Grid:** Shows a grid of vertical lines in the Dopesheet and Spline Editor, making it easier to position items accurately along the time-line when enabled.
- **Snap to Grid:** Shows the snap to grid feature in the Dopesheet and Spline Editor when enabled. This feature shows vertical lines at the field interval defined in the grid width option. By using this feature, Key Frames and other items can be dragged precisely along the time-line, as it is not possible to drop items between the grid lines.

Example: If the snap to grid feature is enabled and the grid width is set to `10`, it is only possible to drag Key Frames to field 10, 20, 30, and so on. If a Key Frame should be placed at field 23 while the snap to grid feature is enabled, this must be manually entered in the Key Frame Editors

- 
Grid Width: Defines the width of the snap to grid in fields.
- 
Show Spline 1, 2, or 3: Shows spline 1, 2, or 3 in the Spline Editor. The splines refer to animations performed on various axes. If only one axis is animated, only the spline 1 option is available.

Example: When animating a rotation, each spline describes the rotation around one axis. Spline 1 shows the rotation around axis X, spline 2 shows axis Y, and spline 3 shows axis Z.

Copyright © 2026 Vizrt

16.8 Time-line Editor

The Time-line editor is located above the Scene Editor. It gives various options to control animations.



16.8.1 Layer/ Director Controls

- **Layer Indicator:** Shows the current layer, as one of three vertical indicators, in the Director Control Panel. The three layers are Front, Main and Back and only available for Classic Pipeline.
- Default **Director:** Defines the selected (active) Director and opens the Director Control Panel.

> **Note:** The settings in the Director Control Panel are not saved with the scene. If only one Director is selected, the button is labelled with the Director name. If multiple directors are selected, the button is labelled *User Defined*. If all directors are selected, the button is labelled *All*.

- **Lock Selection:** Activates the Director(s) of the selected container in the scene tree to be active in the stage when enabled. When disabled, locks the selected Director(s) so that no matter where else you click in the scene tree, only the selected Director is controlled by the buttons in the Time-line Editor.

16.8.2 Play Controls

- **Reverse:** Plays the animation in reverse when enabled.
- **Go to Start:** Jumps the time-line to the start of the current Director(s).
- **Start Animation:** Plays the current Director(s) in the current layer from the beginning.
- **Stop Animation:** Stops the currently playing animation at the current point in the time-line. Right-clicking stops the currently selected Director or directors.
- **Continue Animation:** Continues the animation from where it was stopped.
- **Go to End:** Jumps to the end of the animation.

> **Note:** The animation is automatically stopped when it reaches the end of the time-line, or a stop point along the time-line.

16.8.3 Keyframe and Timeline Controls

- **Key Frame Next:** Jumps to the next Key Frame of the selected container in the scene tree.
- **Key Frame Previous:** Jumps to the previous Key Frame of the selected container in the scene tree.
- **Create Keyframe:** Creates a Key Frame at the defined time-line. This can also done by hitting **ENTER** or**,** if the Scene editor is active (Blue border).
- **Jump to Previous Time-line Position:** Jumps to the previous position in the Time-line.
- F100 **Time-line:** Shows the current position on the time-line in fields or frames depending on the currently set output format.

Copyright © 2026 Vizrt

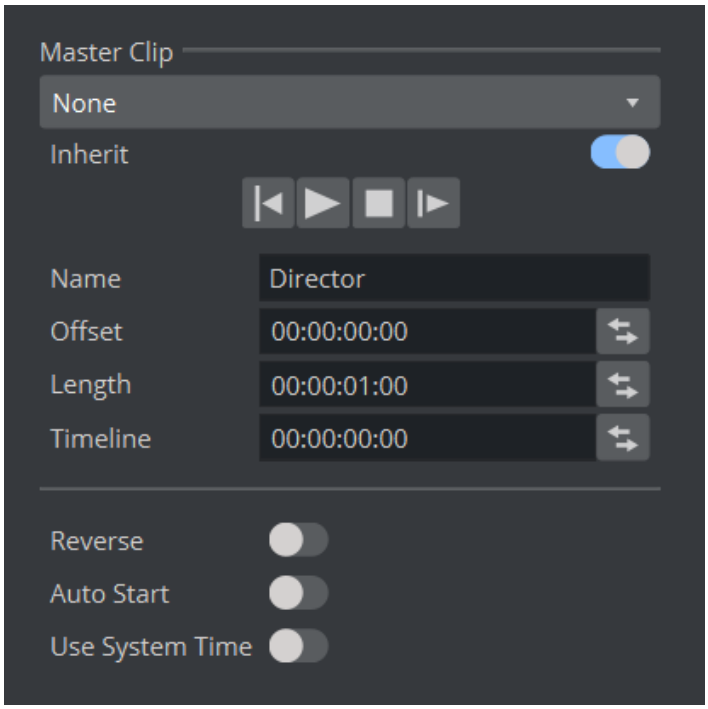
- **Jump to Next Time-line Position:** Jumps to the next position in the Time-line.
- **Move/Advance Time-line:** Moves time-line after a new keyframe is set. This setting is also used by the − and + buttons. By default, it automatically moves one second.
- F60 **Set Time-line Jump:** Sets the number of fields or frames the time-line is moved by a position jump or after a new keyframe is set.

Copyright © 2026 Vizrt

16.9 Director Editor

The Director editor shows details about the selected Director. Renders an animation, defined in the Stage, based on the actual frame of a played clip. This clip can either be set in Video Clip Playback Considerations or within a separate Director in the Stage.

16.9.1 Director Editor Properties



- **Drop-down box:** Select `None`, Clip `1` to `16`. Select the Clip Channel that contains the clip, which controls the animation.
- **Inherit:** When set to `On`, the set Master Clip Channel is passed to all sub directors
- **Play Controls:**
  - **Go to Start:** Jumps to the start of the time-line.
  - **Start Animation:** Starts the animation from the beginning.
  - **Stop Animation:** Stops the animation at the current time-line. (**Note:** The animation is automatically stopped when it reaches the end of the time-line, or a stop point along the time-line.)
  - **Continue Animation:** Continues the animation after it has been stopped.
- **Name:** Defines the name of the Director.
- **Offset:** Sets the offset in fields for the whole Director from the start of the time-line. This is the easiest way to shift all animations within the Director.
- **Length:** The total length of the Director.

Copyright © 2026 Vizrt

- **Timeline:** The current position within the time line is shown as timecode (each Director has a separate time-line). Changing the value in this box moves the Time-line Marker in the Time-line Editor, and the other way around.



Note: Use the format switcher  to switch between timecode based view and fields (for interlaced output formats) or frames (for progressive output formats) view.

- **Reverse:** Plays animations in this Director backwards when enabled.
- **Auto Start:** Starts the animation automatically when it is loaded from the database when enabled.
- **Use System Time:** Plays seamless looping directors, if you have multiple scenes with the same background animation. As an example, if you have multiple scenes for elections but each of the scenes has the same background animation/Director and you set the Use System Time to **Active** for this Director, it plays the background without dropouts or breaking the loop while loading the other scenes, as long as they have the same Director where the system time is activated.

See Also

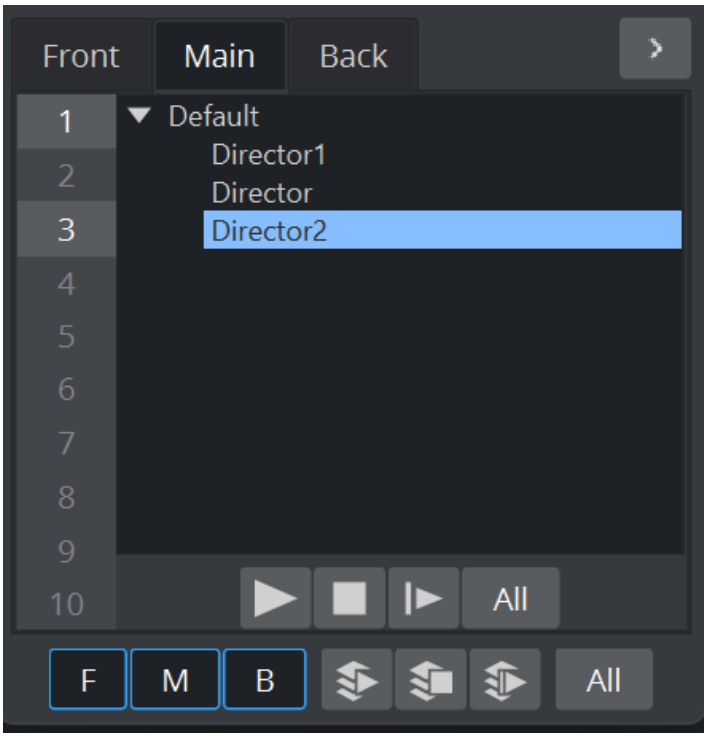
- Master Clip

16.10 Director Control Panel

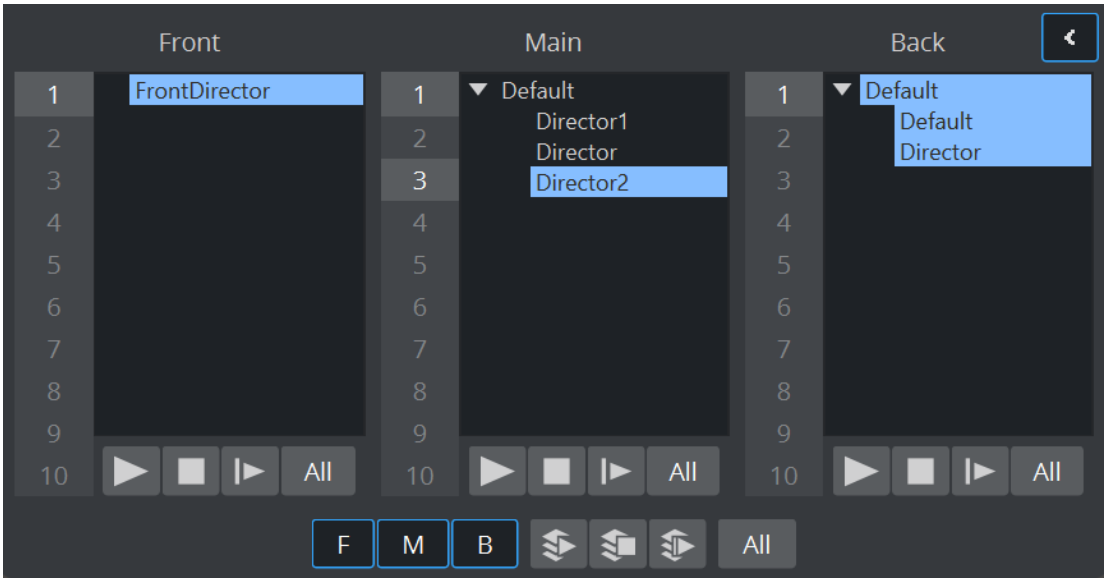
Click the **Director** option in the Time-line Editor to open the **Director Control** panel.

The **Director Control** panel lists all available Directors of the selected layer. The Director Control window allows to select one or more directors in the list and control them individually.

Collapsed View



Expanded View



To switch between compact and expanded view, use the **Expand** (›) button.

Note: There are two sets of control buttons; one set for controlling individual directors in the selected layer and one set for controlling the all directors in all layers.

- **Front:** Lists the directors belonging to the front layer scene.
- **Main:** Lists the directors belonging to the main scene.
- **Back:** Lists the directors belonging to the back layer scene. This button is only enabled if a back layer has been defined in the Layer Manager.
- **All:** Selects all directors in the list when activated.
- **Start Animation/Play Layer:** Plays the current Director(s) of all layers or the selected layer from the beginning (see Basic Animation Functions).
- **Stop Animation/Stop Layer:** Stops the currently playing animation at the current point on the time-line for all layers or the selected layer. Right-clicking stops only the selected directors (see Basic Animation Functions).

Note: The animation is automatically stopped when it reaches the end of the time-line, or a stop point along the time-line.

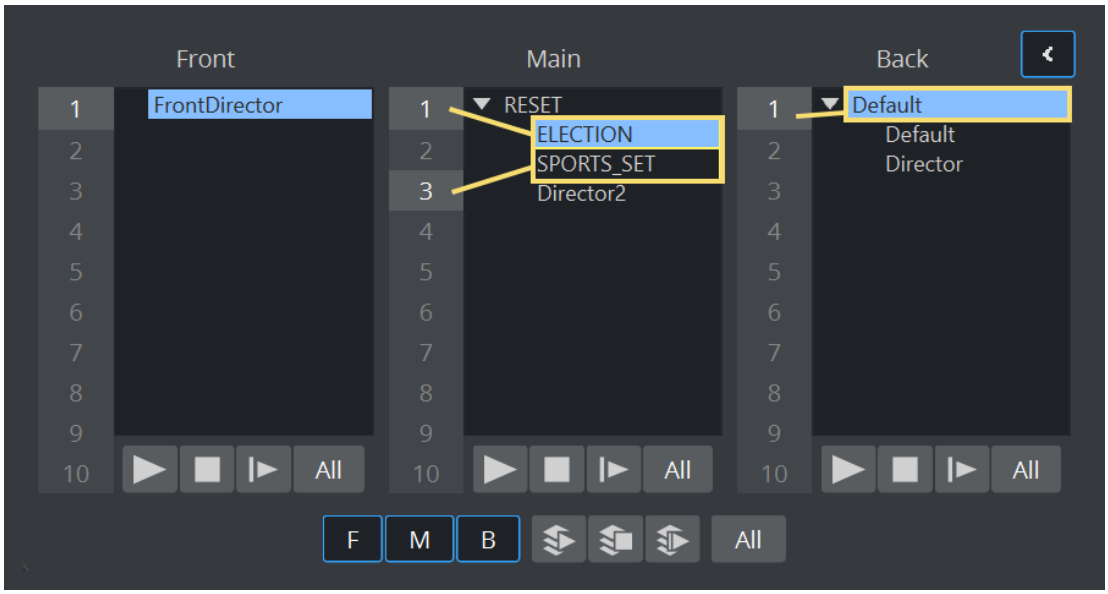
- **Continue Animation/Continue Layer:** Continues the animation of all layers or the selected layer to the next point on the time-line.
- **[1-10]:** You can save up to ten combinations of selected directors per layer by selecting them and dragging them onto one of the numbers. Afterward, when you click a number, the combination of saved directors is then selected.

Note: These selections are on a per-layer basis and also cannot be saved.

- **F, M, B:** Enables the layers that should be controlled using the Play Stop and Continue Layer buttons.

Copyright © 2026 Vizrt

16.10.1 Working with the Director Control Panel



When multiple Directors in the selected layer are available, you can select and drag them onto the preset/numbers bar in the **Director Control** panel to call up different presets at any time.

To do so, you need to:

1. Select at least one Director (or multiple directors with **SHIFT** or **CTRL + click**).
2. Drag the selection on to one of the numbers in the **Director Control** panel.

For example, in the image above, *ELECTION* can be dragged onto slot **1**, *ELECTION* + *SPORTS_SET* is assigned to 3.

To then control other combinations you need only drag the selected Container on to a number and a new preset is defined. In this example, you can select *ELECTION* and *SPORTS_SET* and drag them on to a *new number*. Then, whenever this number is clicked, the predefined selection is highlighted/selected and can be triggered via the **Director Control** buttons.

> **Note:** Director selections are not stored within the scene. They are supposed to be used for designing purposes only.

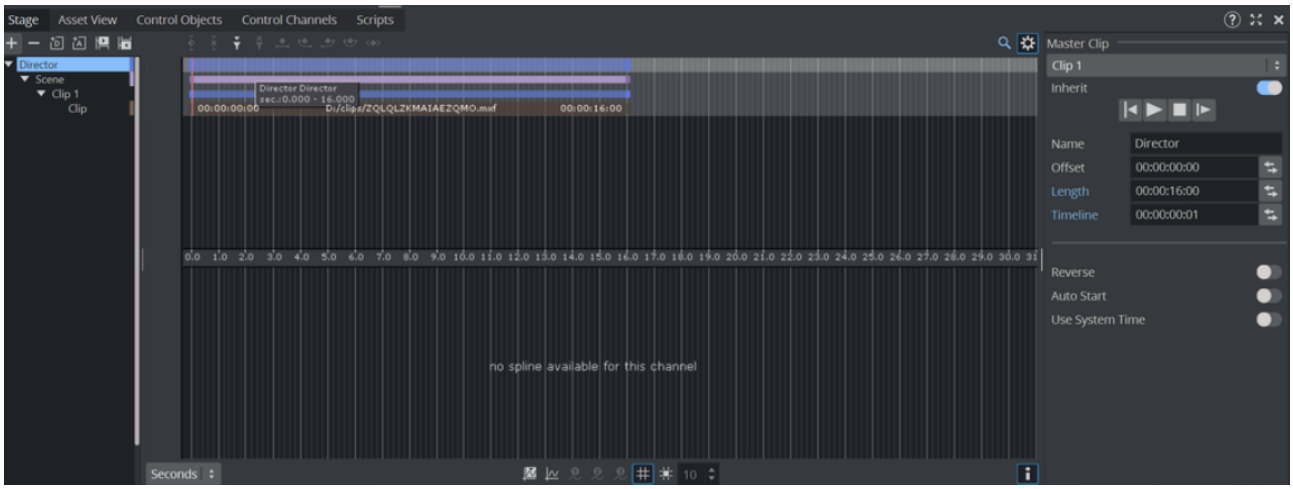
Copyright © 2026 Vizrt

16.11 Master Clip

The Master Clip feature is particularly useful when you need to tie graphic animations to a specific location in a video clip. Using the director properties to tie a master clip to a director in the Director Editor makes sure that the stage and the clip are in sync. This is to make sure, that when the clip plays the frame at time 3:12 for example, you also get the animation of the graphic rendered at 3:12.

It is essential that you have at least two top-level directors in a scene using the Master Clip feature. One contains the clip only, and the second all other scene items. The second director can also contain several sub-directors. For the animations in the second director to be bound to the clip it is necessary to set the Master Clip to the clip channel containing the clip for this director. If there are sub-directors, which should also respect the clip as the animation controller, the **Inherit** button must be checked in the Director Editor.

A simple scene using the Master Clip feature could be constructed like this:

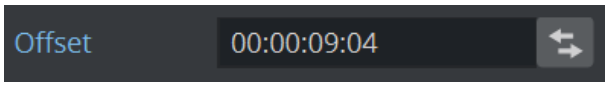


Note: Scrubbing in the controlled director is not possible when using the Master Clip feature as the timing is set by the clip. Scrubbing is only supported in the director containing the clip.

Copyright © 2026 Vizrt

16.12 Actor Editor

The Actor Editor shows details about the selected Actor. The available actions may vary, depending on the actor selected.



- **Offset:** Sets the offset for the whole actor from the start of the time-line. This is the easiest way to shift all animations within the actor. The format of the offset can be switched from timecode to the one currently selected (Seconds, Fields, Frames).

16.12.1 Actor Editor for Integer Channels

Integer channels are channels for objects that have numeric values written without a fractional component, as opposed to floating point values.

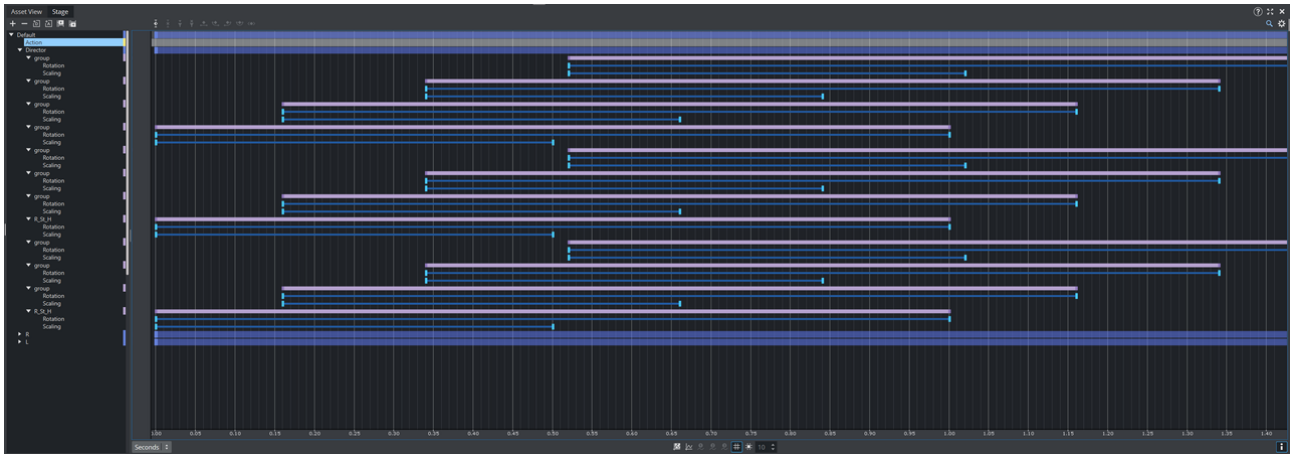


- **Interpolation:** General setting for all integer animation channels. If interpolation is switched off, no interpolation of the integer values happens when animating from one key-frame to another. The value changes only when a key-frame is passed when running the animation. The default setting is `Off`.

Copyright © 2026 Vizrt

16.13 Dopesheet Editor

The Dopesheet editor shows the items in the Stage Tree as bars along the time-line.



The items are presented vertically in the Dopesheet editor according to the structure in the Stage Tree. The horizontal scaling represents the time value.

Tip: The time value can be set to either seconds, frames, or fields from the Stage Editor bar.

The Time-line Marker defines the current time, and the Scene Editor shows the animation accordingly. The Grid and Snap to Grid options that can be enabled in the Stage Editor make it possible to position items in the Dopesheet editor more accurately.

The stage always works on fields, the X-axis can be set to show **seconds**, **frames** or **fields**. The scaling can be set on the toolbar.

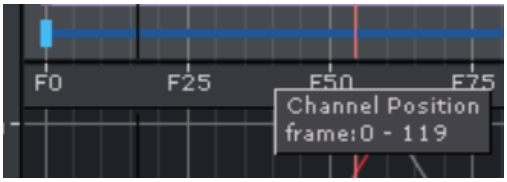
Seconds

The timeline markers are shown as seconds.



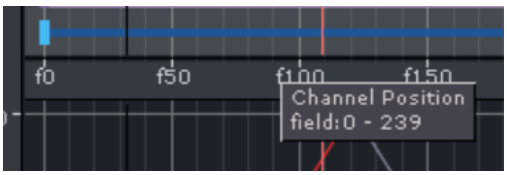
Frames

The timeline markers are shown as frames (**F**).



Fields

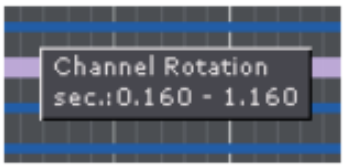
The timeline markers are shown as fields (**f**).



To make sure your keyframes are aligned (for example, on a ten field base), a **Snap to Grid** () option can be set.

The marker then snaps on every tenth field.

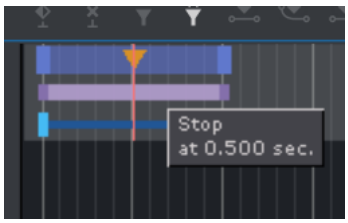
If the Show Info option is enabled in the Stage Editor, a tool tip with time information shows when the pointer is held over an item in the Dopesheet editor:



The Dopesheet editor can contain bars for various stage items: Directors, Actors, Channels, etc. Similar to the Stage Tree, Directors are shown with a gray blue color, Container Actors with light purple, while channels and actions are shown with a gray color. To build up an animation, Key Frames and stops/tags can be added to the Scene. A Key Frame defines a property at a given time. Multiple Key Frames belonging to the same channel are connected in the Dopesheet editor with one or more bars.

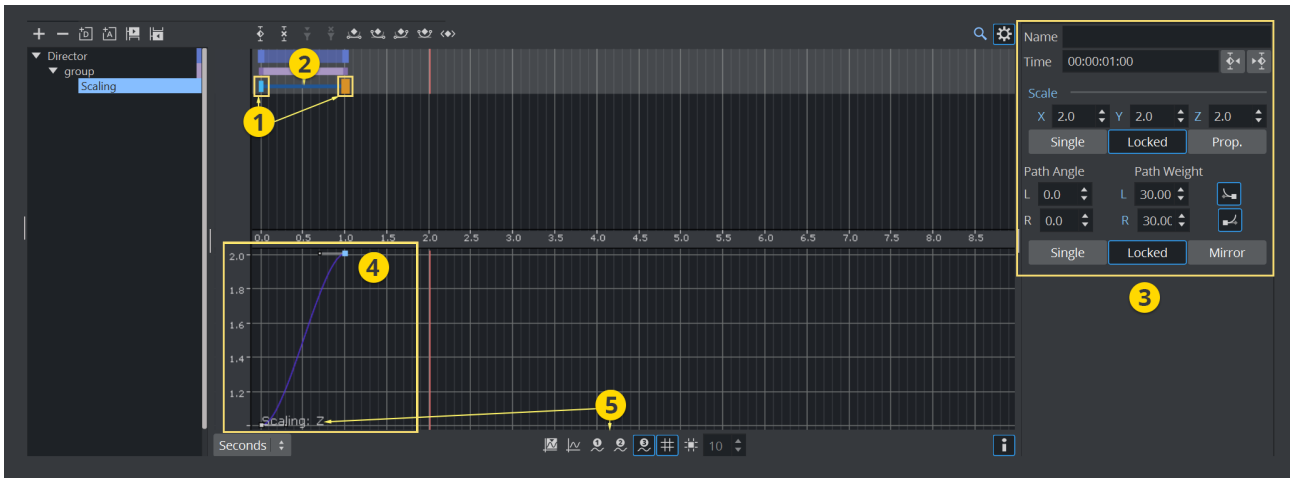
Copyright © 2026 Vizrt

The image below shows an animation where a stop/tag has been added between two Key Frames:



A stop/tag can be added to a Director to end or pause an animation at a given time.

> **Example:** If the scaling of a cube is 1.0 at field 0 and 2.0 at field 50, one Key Frame is created at field 0 and another at field 50. A bar connects the two Key Frames in the Dopesheet editor.

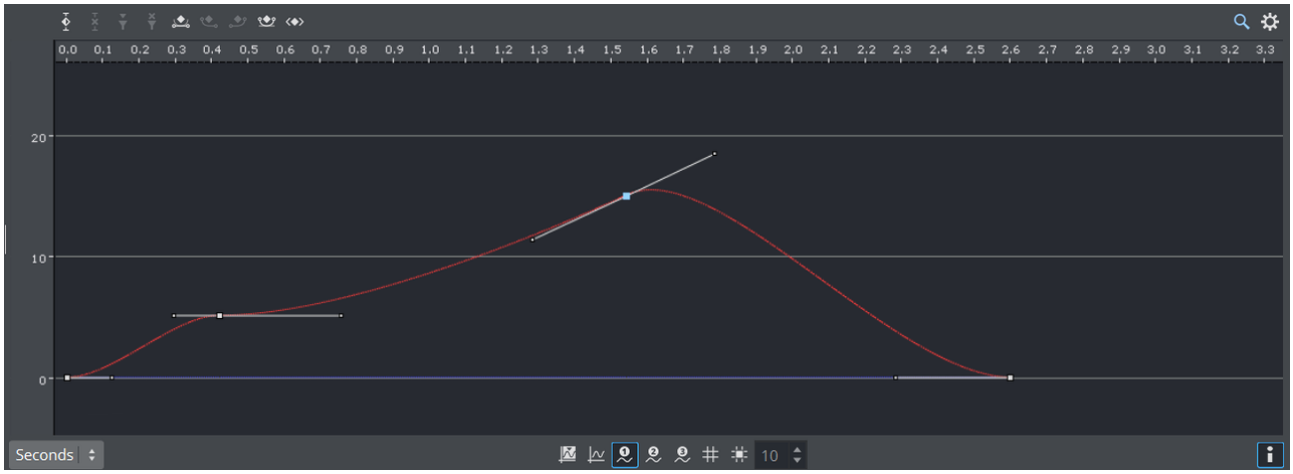


- (1) Keyframes
- (2) Connecting bar between two keyframes
- (3) Corresponding Keyframe editor
- (4) Spline Editor
- (5) Selected Spline

Copyright © 2026 Vizrt

16.14 Spline Editor

The Spline editor shows the Key Frames on a Channel as splines along the time-line.



The Key Frames are presented vertically in the Spline editor according to the properties they possess. The horizontal scaling represents the time value.

> **Tip:** The time value can be set to either seconds, frames, or fields from the Stage Editor Bar.

The Time-line Marker defines the current time, and the Scene Editor shows the animation accordingly. It is possible to drag Key Frames in the Spline Editor both horizontally (changing the time) and vertically (changing the value) at once.

This section contains information on the following topics:

- To Resize the Height or Width of the Spline Editor
- Zooming
- To Switch Between the Various Splines
- Example of Axes X, Y, and Z in the Spline Editor
- To Move Key Frames in the Spline Editor
- Editing Handles
- Position in the Spline Editor
- Rotation in the Spline Editor
- Scale in the Spline Editor
- Material in the Spline Editor
- Additional Best Practices for the Spline Editor
    - Example of Key Frame and Handle Tool Tips

16.14.1 To Resize the Height or Width of the Spline Editor

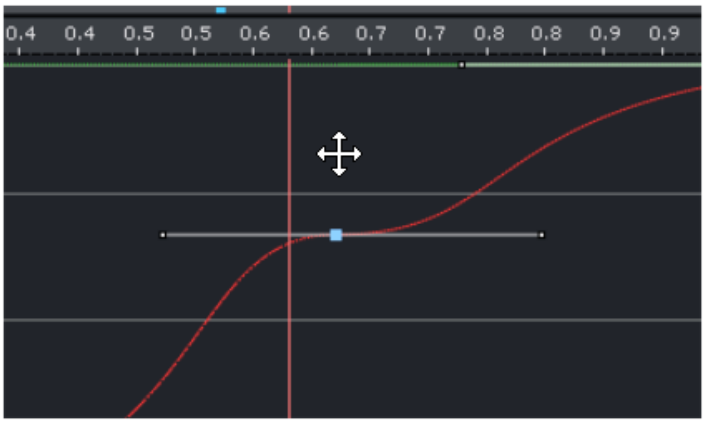
- **Height:** Drag the horizontal separator between the Dopesheet editor and Time-line Marker.

Copyright © 2026 Vizrt

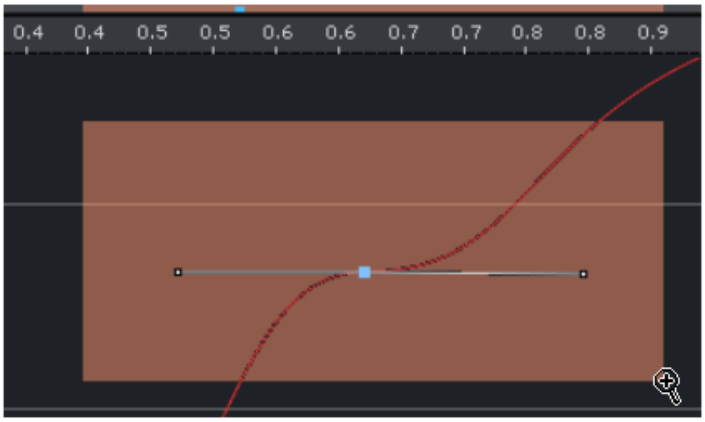
- **Width:** Drag the vertical separator between the Stage Tree Area and Stage Editor.

16.14.2 Zooming

You can zoom in and zoom out on your spline. The **Z** or **Y** keys zoom in, the **X** key zooms out. Alternatively, the mouse wheel allows to zoom in, where a click and drag on the middle mouse button allows to pan your view.



You can also **Zoom In** when pressing the **Z** Key and drag the zoom area with your mouse:



Use the **A** and **S** keys to change the scaling in the Y-Axis. To reset your view you can either click on the reset to standard view or to reset to the selected spline.

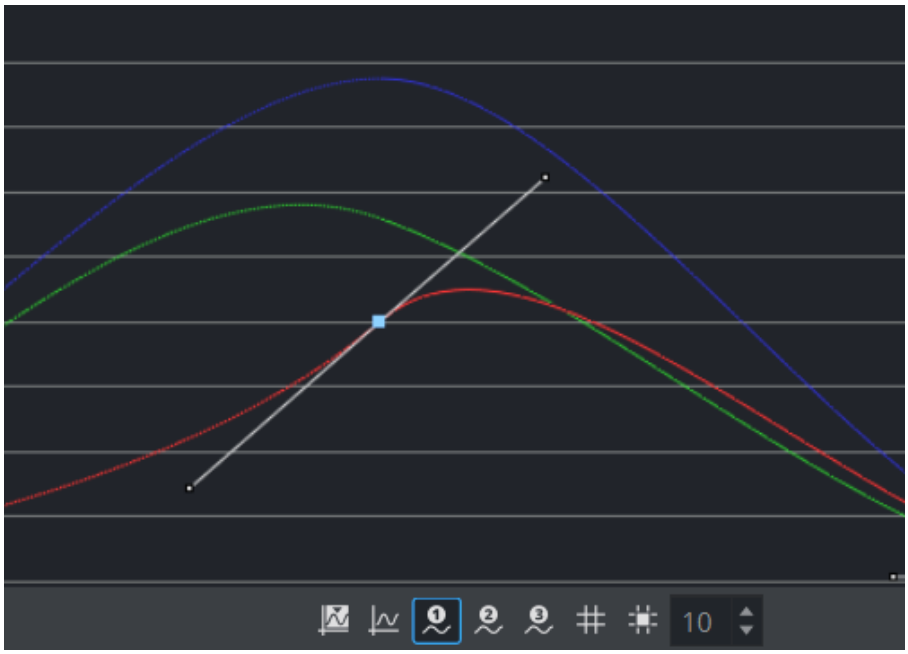
16.14.3 To Switch Between the Various Splines

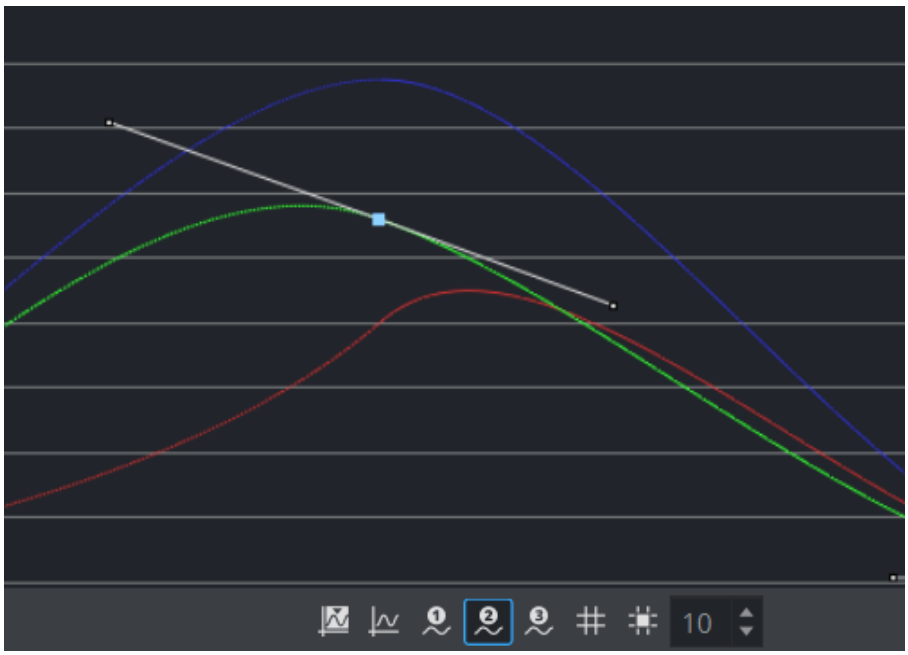
Some properties like Scaling, Rotation require three splines (X,Y,Z) on the same timeline. All of the three splines are showed at the some time, but you can only modify the selected one.

- Use the Show Spline 1,2, and 3 buttons from the Stage Editor Bar

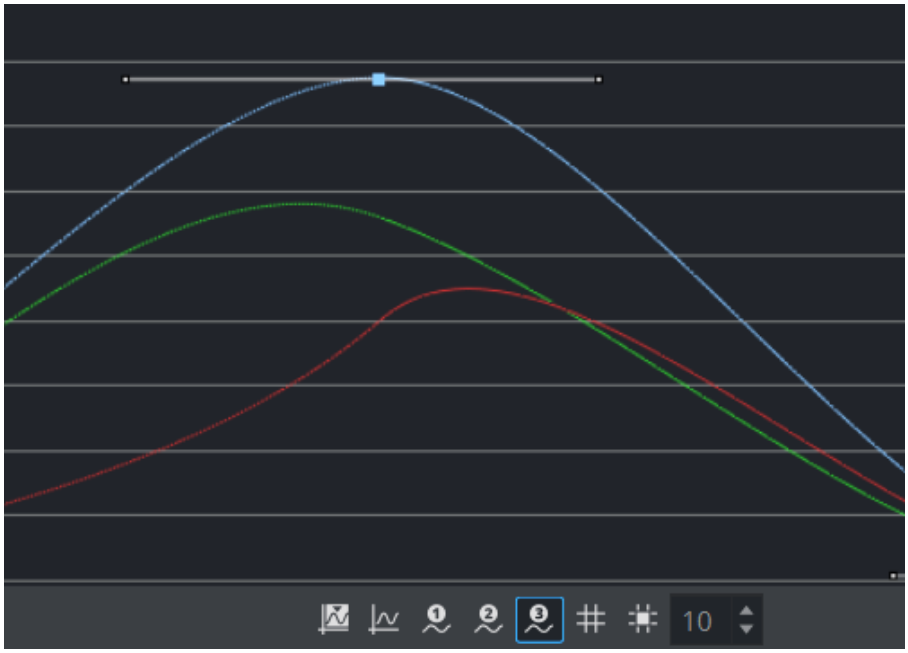


16.14.4 Example of Axes X, Y, and Z in the Spline Editor





Copyright © 2026 Vizrt



16.14.5 To Move Key Frames in the Spline Editor

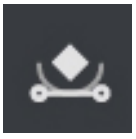
- Select the required Key Frame and drag it around.
- To change only the position on the time-line or the value, press **SHIFT** to lock the axes. Keep **CTRL** pressed while you drag the Key Frame if you want to snap to grid along the time-line.
  You can move a Key Frame in the spline editor, so either the position on the time-line (left and right) or the value stored (up and down) in the Key Frame is changed.

16.14.6 Editing Handles

As each Key Frame on the spline has two handles, moving them is the most easy way to edit the curve. To edit a handle left-click it and drag it around. You see that the spline reflects your movement.

By default you move the handles proportional this way. This means if you change the handle on one side, the one on the other side moves too. If you want to only edit the handle on one side, you need to left-click the required handle. Keep the left button pressed and press **CTRL**. If you drag the handle, only the one you drag right now moves and the other one stays. This “single handle mode” is enabled for the Key Frame as long as you press **CTRL**.

Furthermore you can use the buttons in the main menu to edit the handles at a Key Frame. To do so select the Key Frame you want to change the handles in and select the action in the main menu. For ease of use multiple Key Frames can be selected.

Icon	Description
	Sets left and right spline handles in selected Key Frame to linear.

Copyright © 2026 Vizrt

Icon	Description
	Sets spline handles in selected Key Frame to smooth left and linear right.
	Sets spline handles in selected Key Frame to linear left and smooth right.
	Sets left and right spline handles in selected Key Frame to smooth.
	Sets left and right spline handles in selected Key Frame to tangential.

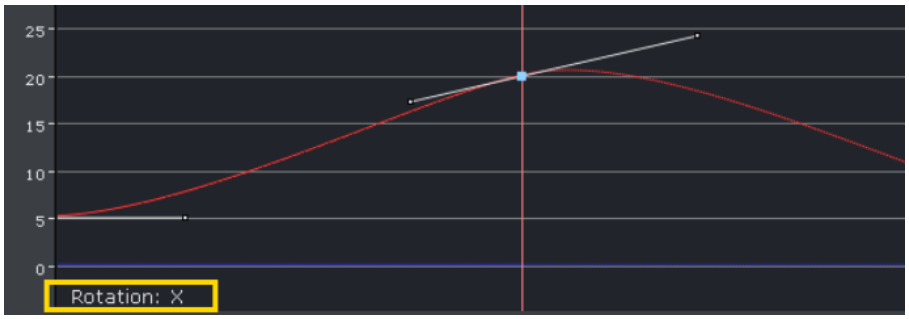
16.14.7 Position in the Spline Editor

If you animate a position, there are much more considerations to make.

16.14.8 Rotation in the Spline Editor

If you have animated the rotation, you can rotate about up to all three axes at the same time. In the spline editor, three different splines are shown, one for every axis. The red one is for the X-axis, the green one for the Y-axis and the blue one for the Z-axis. For ease of use in the lower left corner, you find indicated which axis is currently selected.

To switch between the three axes, click on one of these icons: 

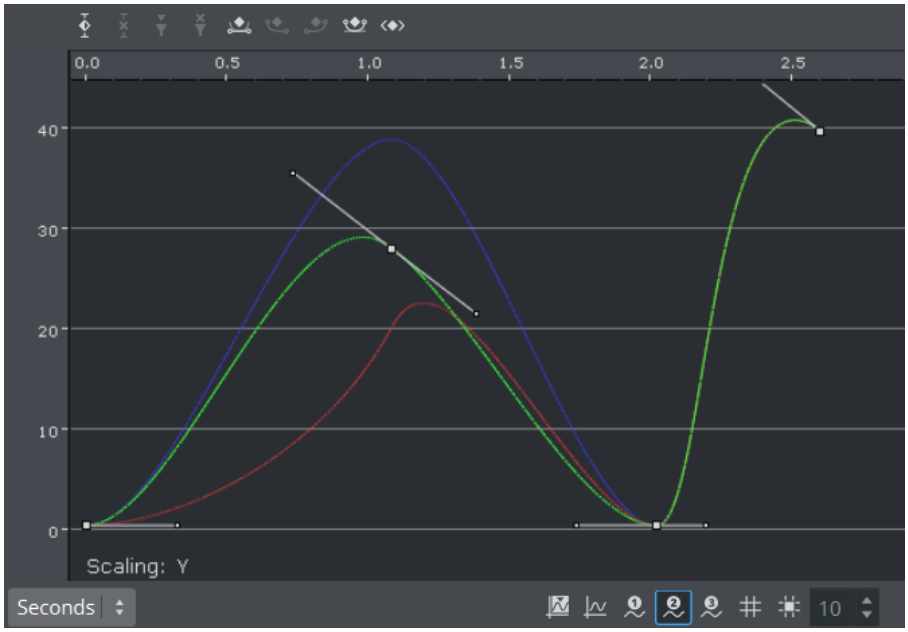


The selected property (in the above sample Rotation: X) is shown in the lower left corner.

Copyright © 2026 Vizrt

16.14.9 Scale in the Spline Editor

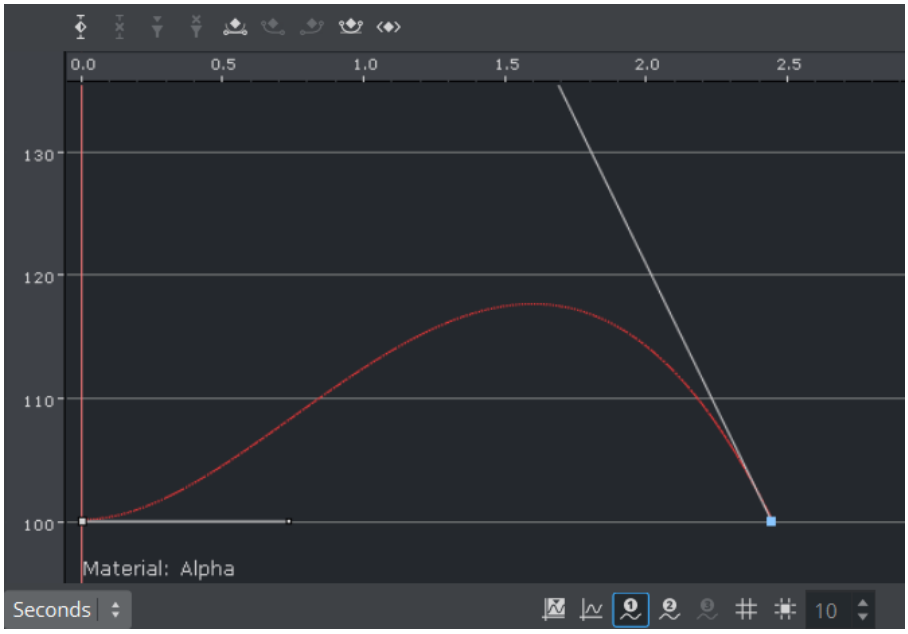
If you have animated the scale, you can scale along up to all three axes. In the spline editor, three different splines are shown (when **Prop** or **Single** is selected), one for every axis. The red one is for the X-axis, the green one for the Y-axis and the blue one for the Z-axis. For ease of use in the lower left corner, you find indicated which axis is currently selected.



16.14.10 Material in the Spline Editor

If the material has been animated, two different splines are shown in the spline editor. The red one is for the alpha value, the green one for the shininess. For ease of use in the lower left corner, you find which parameter is currently selected.

Copyright © 2026 Vizrt



The color values are edited by selecting the keyframe.

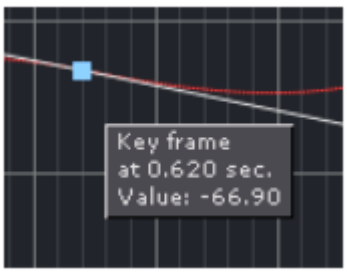
16.14.11 Additional Best Practices for the Spline Editor

Normally for each channel only one spline is shown in the spline editor. But there are animation channels where more splines are available for editing. As you can only edit the Key Frames/handles of one spline, you must use the number keys in the Stage Editor Bar to select the required spline.

Pressing **SHIFT** while moving a Key Frame in the Spline Editor enables you to drag it only horizontally or vertically, but not both directions at the same time. Pressing **CTRL** while moving a Key Frame enables the Snap to Grid function, allowing the Key Frame to be placed only at certain time-line intervals. For more information (see Stage Editor Bar).

When animating certain channels, such as scaling or rotation, up to three splines are shown in the Spline editor, each describing the state along one axis, either X, Y, or Z. For a better overview, different colors are used; red for X-axis, green for Y-axis, and blue for Z-axis. Every spline contains various Key Frames with handles.

Example of Key Frame and Handle Tool Tips

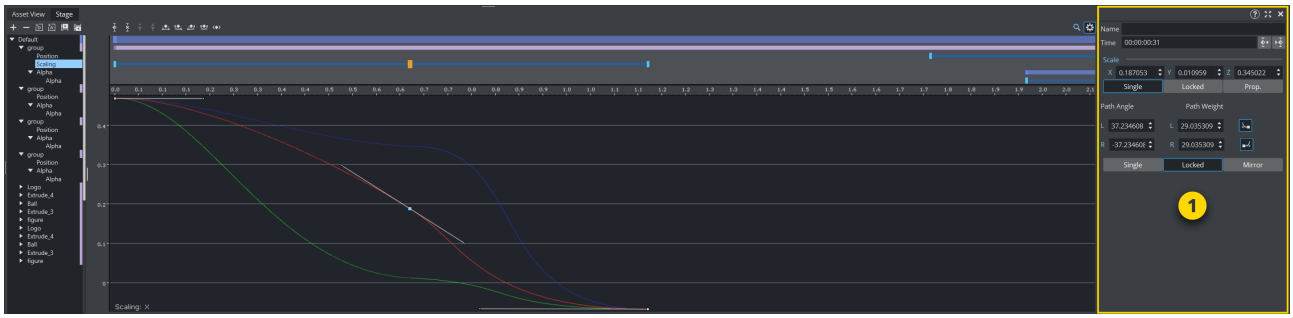


If the Show Info option is enabled in the Stage Editor Bar, placing the pointer over a Key Frame or handle shows a tool tip with status information about time-line and value.

Copyright © 2026 Vizrt

16.15 Stage Object Editor

The Stage Object editor (1) is positioned at the right side of the Stage Editor. The editor has different views depending on the particular editor shown.



Information: The Director Editor shows information about directors, the Actor Editor shows information about actors, the Channel Editor shows information about channels, the Actor Editor shows information about actions, the Key Frame Editors shows information about Key Frames, the Event Editor shows information about stops/tags, and so on.

Clicking an object in the Dopesheet Editor or selecting an item from the Stage Tree opens the corresponding editor in the Stage Object Editor.

Copyright © 2026 Vizrt

17 Create Animations

When an animation is created, the state of an object (for example, its position, rotation, or scaling) is defined at various points in time. Key Frames are used to define these states, and the animation system interpolates between them to create a continuous animation.

This section contains information on the following topics:

- Actors
- Channels
- Basic Animation Functions
- Create a Basic Animation
- Key Frames
- Directors
- Create a Stand-alone Scene
- Create an Over the Shoulder Scene
- Action Channels
- Create an Advanced Animation
- Advanced Animation Functions

Copyright © 2026 Vizrt

17.1 Actors

A Container is represented as an Actor in the Stage Tree. The Actor carries the same name in the Stage Tree as the corresponding Container in the Scene Tree. If the Container is renamed, the Actor reflects the change.

If an Actor or Actor channel is clicked on it also shows the relevant Container (in the Scene tree) and object (in the Scene Editor).

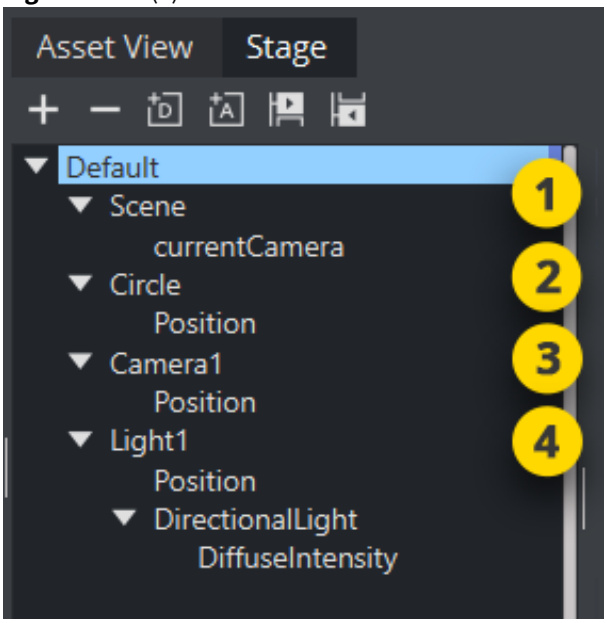
> **Note:** If multiple Actors or Actor Channels are selected this is also shown in the Scene Tree and Scene Editor.

When adding an animation to a Container, the corresponding Actor and channel automatically shows in the Stage Tree inside a Director. The Actor for a Container holds the information of all animated channels: Scene, Camera, or Light.

As it is possible to place different animation channels of an Actor in separate Directors, the same Actor can be available in several places in the Stage Tree.

Each Actor type is represented by a different color:

- **Scene Actors** (1)
- **Container Actors** (2)
- **Camera Actors** (3)
- **Light Actors** (4)



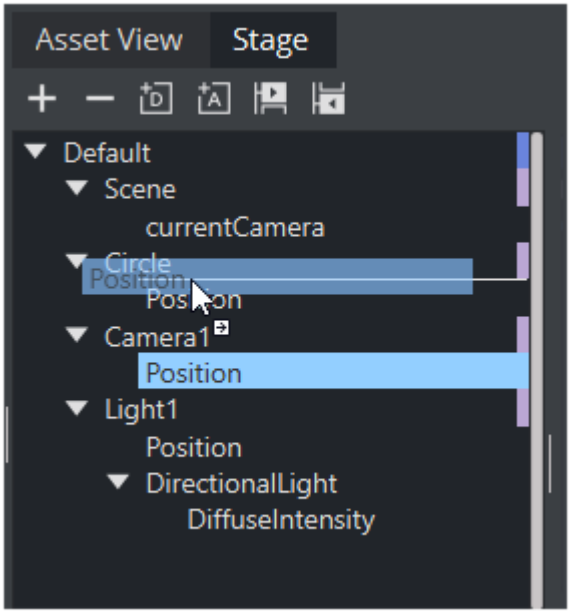
Most actions on Actors are done in the Stage Tree. Actions which affect the time-line are done directly in the Dopesheet editor. Working with Actors in the Dopesheet editor enables the Actor Editor.

Copyright © 2026 Vizrt

17.1.1 Actors in the Stage Tree

To Move an Actor

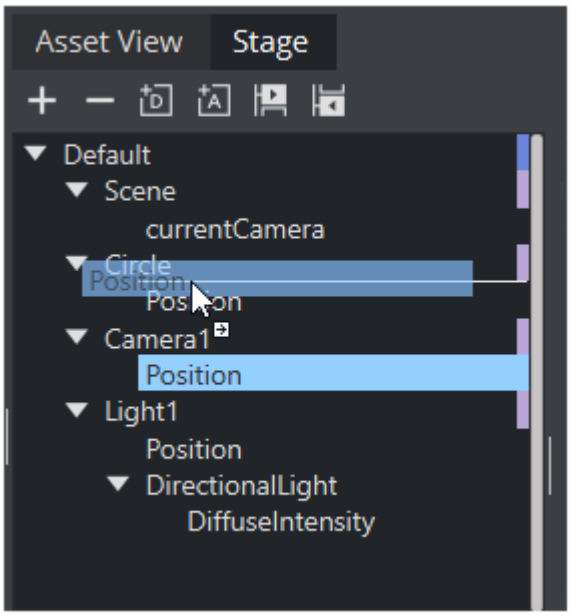
Drag one or more actors from a Director in the Stage Tree to the target Director.



To Copy an Actor

Press **CTRL** while dragging one or more actors from a Director in the Stage Tree to the target Director.

Copyright © 2026 Vizrt



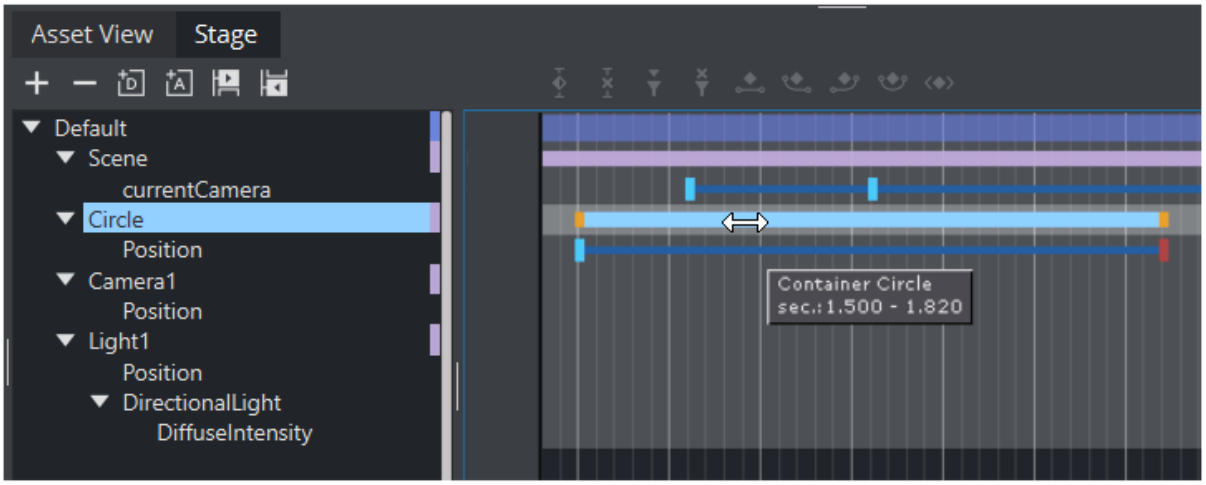
To Delete an Actor

- Right-click an actor in the Stage Tree and select **Delete Actor**.
  or
- Drag the actor to the trash can.

17.1.2 Actors in the Dopesheet Editor

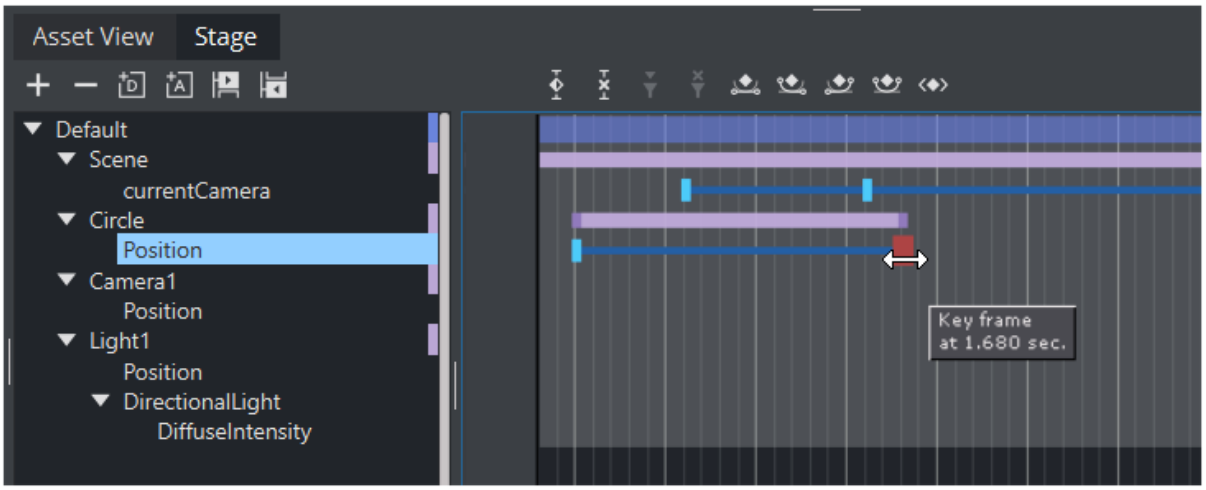
To Move an Actor in the Time-line

Drag the actor to a different time.



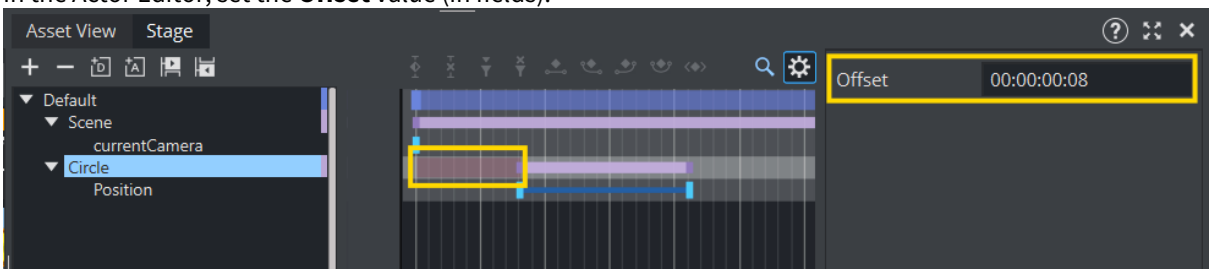
To Speed Up or Slow Down an Actor

- Click on the start or the end of the Actor.
- Drag the start or end point to change the Actor speed:



To Delay all the Animations in an Actor

1. Enable the Actor Editor.
2. Select an actor.
3. In the Actor Editor, set the **Offset** value (in fields).



Also, it is possible to move the actor (horizontally along the time-line) including all its animations directly in the Dopesheet Editor, although this does not actually add the offset value.

Copyright © 2026 Vizrt

17.2 Channels

A Channel stores the animated properties of an Actor.

Separate Channels are created for Position, Rotation, Scaling, and so on. The Channel is named automatically according to the property being animated. It is not possible to rename a Channel.

Key Frames are placed along the Channel. Key Frames hold information about the value of the property at the time they are placed on the time-line. If Key Frames are not connected by a Channel in the Dopesheet Editor, the information about the property is identical, and there is no animation of this property between these Key Frames.

Most actions on Channels are done in the Stage Tree. Actions which affect the time-line are done directly in the Dopesheet editor. Working with channels in the Dopesheet editor enables the Channel Editor.

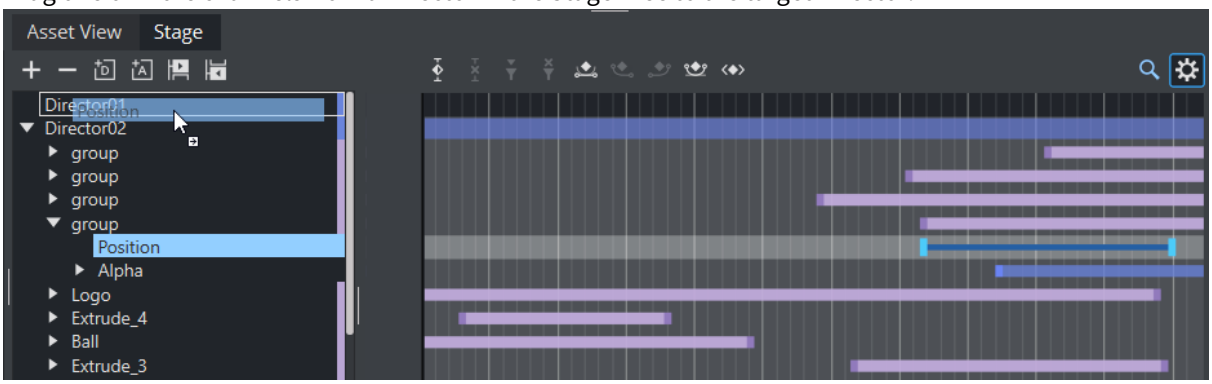
This section contains the following information:

- Channels in the Stage Tree
- Channels in the Dopesheet Editor

17.2.1 Channels in the Stage Tree

To Move a Channel

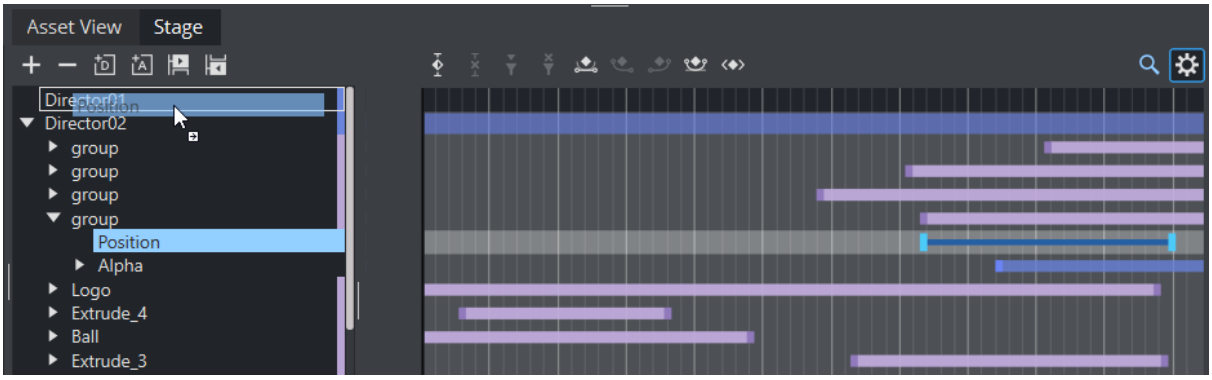
- Drag one or more channels from a Director in the Stage Tree to the target Director.



To Copy a Channel

- Press **CTRL** while dragging one or more channels from a Director in the Stage Tree to the target Director.

Copyright © 2026 Vizrt



To Select a Channel

- Click a channel in the Dopesheet Editor, or
- Click one or more channels in the Stage Tree.

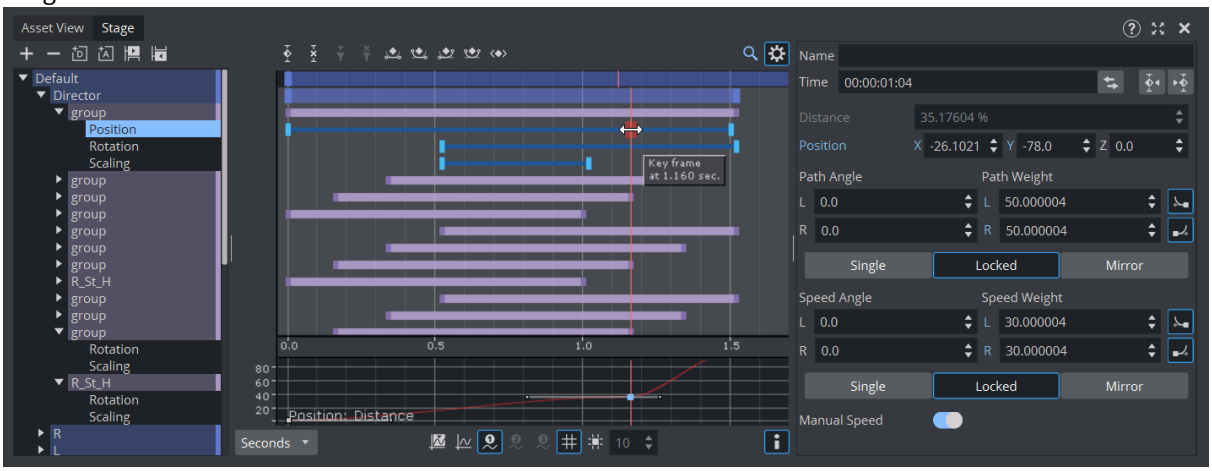
To Delete a Channel

- Right-click a channel in the Stage Tree and select **Delete Channel**
  or
- Drag the channel to the trash can.

17.2.2 Channels in the Dopesheet Editor

To Move a Channel in the Time-line

- Drag the channel to a different time.



To Speed Up or Slow Down a Channel

- Click on the start or the end Key Frame of the Channel.

Copyright © 2026 Vizrt

- Drag the start or end Key Frame to change the Channel speed:
    - **Speed up:** Lengthens the Channel.
    - **Slow Down:** Shortens the Channel.

Copyright © 2026 Vizrt

17.3 Basic Animation Functions

This section contains information on the following procedures:

- To Play All Animations in a Director from the Start of the Time-line
- To Play Multiple or All Animations in a Scene
- To Play an Animation in Reverse
- To Stop a Director
- To Stop an Animation
- To Stop All Animations
- To Continue an Animation
- To Jump to Start of an Animation
- To Jump to the End of an Animation
- To Delay Animations in a Director with Offsets
- Scale Animations
- To Scale Animations in a Director

17.3.1 To Play All Animations in a Director from the Start of the Time-line

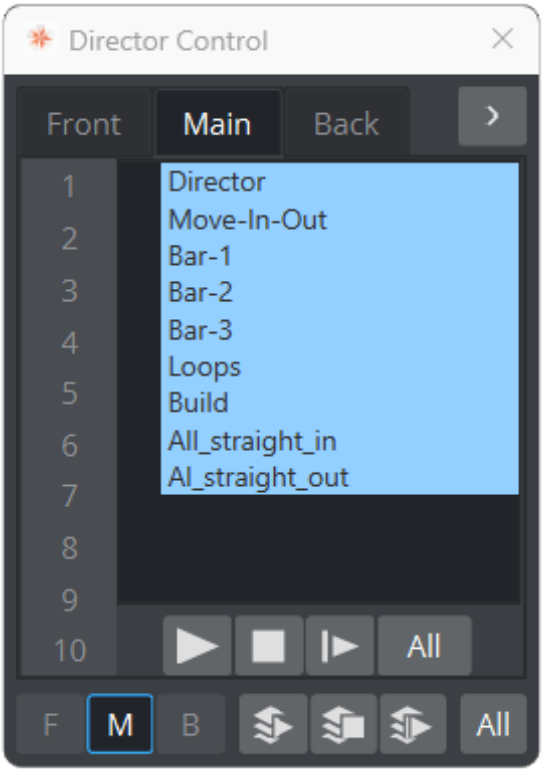
1. Select the required Director.
2. Either
    - Click the **Start Animation** button in the Time-line Editor.
    - When the **Stage** is active, pressing **ENTER** starts any animations on the selected director(s) from the first frame. Pressing **SPACE** pauses or resumes play-out.
    - Click the **Start Animation of Selected Layers** button from the Director Control Panel.

17.3.2 To Play Multiple or All Animations in a Scene

When working with a scene that contains more than one director, the current director is always selected by default in the scene editor preview window. To preview all animations in a multi-director scene, all directors can be started simultaneously:

1. Open the Director Control Panel.
2. Click the **All** button, or hold **SHIFT +** click to select the range of directors. Holding **CTRL** and clicking allows for selecting individual directors instead of a range.

Copyright © 2026 Vizrt



3. To start the animation(s), either
   - Click the **Start** button in the Time-line Editor.
   - Click the **Play layer** button in the Director Control Panel.
   - Click the **Start animation of selected layers** button in the Director Control Panel. This also starts the selected animations in the other layers, if they are available.
   - When the **Stage** is active, pressing **ENTER** starts any animations on the selected director(s) from the first frame. Pressing **SPACE** pauses or resumes play-out.

17.3.3 To Play an Animation in Reverse

1. In the Time-line Editor, click the ◀ (Reverse Play) button.
2. Either:
   - Click the **Start** button in the Time-line Editor.
   - Click the **Play layer** button in the Director Control Panel.
   - Click the **Start animation of selected layers** button in the Director Control Panel. This also starts the selected animations in the other layers, if they are available.
   - Press **SHIFT + ENTER** when the Stage is active.

17.3.4 To Stop a Director

1. Select one or more Directors.
2. Either:

- In the Time-line Editor, right-click the **Stop** button.

Copyright © 2026 Vizrt

- Click the **Stop** button from the Director Editor.
- Press **SPACE** while the Stage is active.

17.3.5 To Stop an Animation

In the Time-line Editor, click the **Stop** button. It is also possible to add one or more stop points to a Director, so that each time the time-line reaches a certain frame, the animations stop automatically.

17.3.6 To Stop All Animations

1. Enable the Director Control Panel
2. Select all directors and all layers.
3. Click the **Stop animation of selected layers** button. This stops all animations in all layers.

17.3.7 To Continue an Animation

1. Select the required Director.
2. Either:

- Click the **Continue Animation** button in the Time-line Editor
- While in the Stage, press **SPACE**.
- Click the **Continue Animation** button in the Director Editor.

17.3.8 To Jump to Start of an Animation

1. Select the required Director.
2. Click the **Go to Start** button in the Time-line Editor, or press **CTRL + HOME**.

17.3.9 To Jump to the End of an Animation

1. Select the required Director.
2. Click the **Go to End** button in the Time-line Editor, or press **CTRL + END**.

17.3.10 To Delay Animations in a Director with Offsets

1. Enable the Director Editor and select the required Director.
2. In the Director editor, set the **Offset** value (in fields). It is also possible to move the Director, including all its animations, horizontally along the time-line in the Time-line Editor. This does not add the offset value.

17.3.11 Scale Animations

Scaling a Director is the easiest way to change the total running time of the animations in the Director. Scaling is done proportionally and includes all Sub-Directors.

Copyright © 2026 Vizrt

17.3.12 To Scale Animations in a Director

Drag the start or the end point of a Director in the Dopesheet Editor. It is not possible to drag the start point beyond the offset value.

Copyright © 2026 Vizrt

17.4 Create a Basic Animation

This section details how to create a basic animation in a Scene.

17.4.1 Animation Task

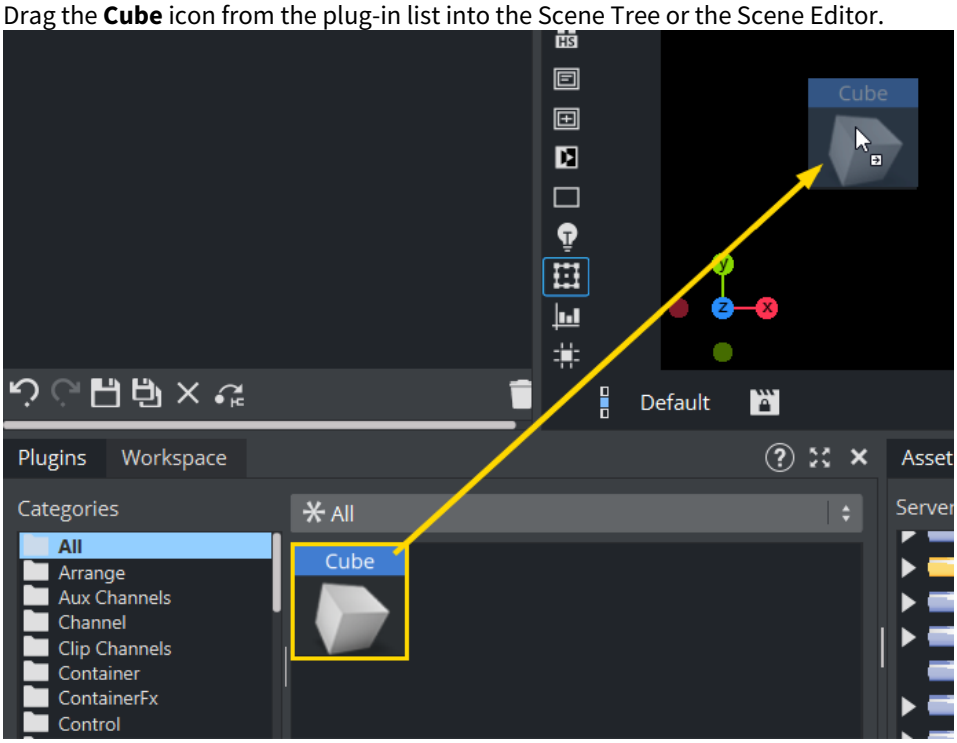
Create a Scene with single a Cube Geometry. Animate the Cube to move right to left.

This use case contains the following procedures:

- Animation Task
- To Create the Basic Animation Scene
- To Create a Basic Animation
- To Create Key Frames with the Transformation Editor

17.4.2 To Create the Basic Animation Scene

1. Create a new scene, and save it as Basic Animation.
2. Open the Basic Animation Scene.
3. In the Plugins Panel, type `Cube` to select the Cube Geometry.
4. Drag the **Cube** icon from the plug-in list into the Scene Tree or the Scene Editor.



17.4.3 To Create a Basic Animation

1. Set the time-line, in the Scene Editor (1), to `0` (fields).
2. Create the first Key Frame:

Copyright © 2026 Vizrt

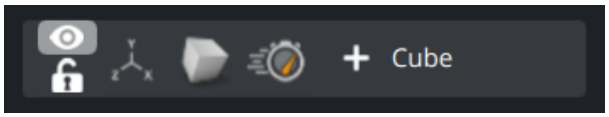


a. Click on the Cube in the Scene Editor.
b. Click the **Set Key Frame** button (2) or press **ENTER**. This sets the first Key Frame.

3. Create the second Key Frame:
   a. In the Scene Editor, drag the Cube to the right, so that it shifts to a different position.
   b. Click the **Set Key Frame** button or press **ENTER**. This sets the second Key Frame.

Tip: Use the Transformation editor to move the Cube to precise position.

Note: On the Cube Container, in the scene tree, an Anim icon is automatically added, which shows that one or more properties on this Container has been animated.

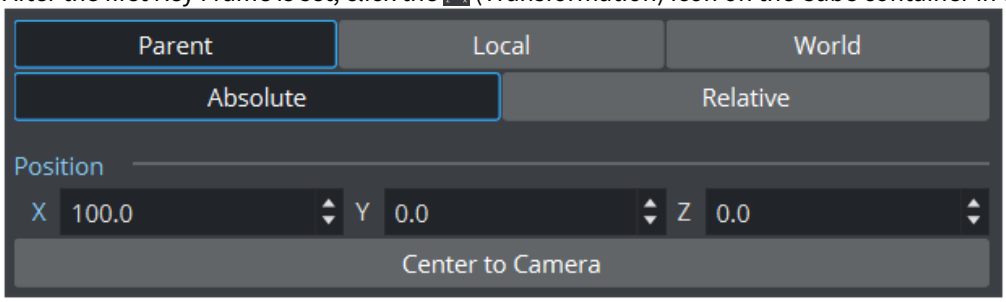


5. Repeat from step 1 to create more movement, if required.
6. Click the **Start Animation** button (3) to see the created animation.

17.4.4 To Create Key Frames with the Transformation Editor

Use the Transformation editor to move the Cube to precise locations then set a Key Frame.

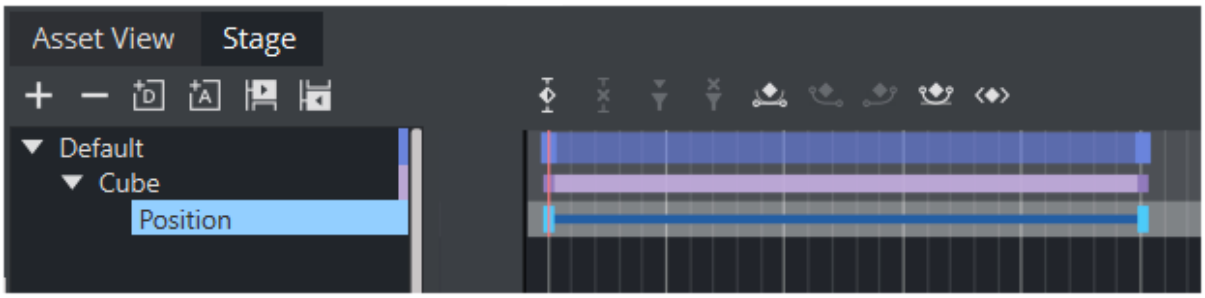
1. After the first Key Frame is set, click the (Transformation) icon on the Cube container in the scene tree.



2. In the Transformation Editor, set **Position X** to `100`.
3. Click the **Set Key Frame** button or press **ENTER**. This sets the second Key Frame.
4. Move the Cube as required with the **Position** parameters to a new position. In the new position click the **Set Key Frame** button, or press **ENTER**, to set a Key Frame.

A representation of the animation can be seen in the Stage Tree Area.

Copyright © 2026 Vizrt



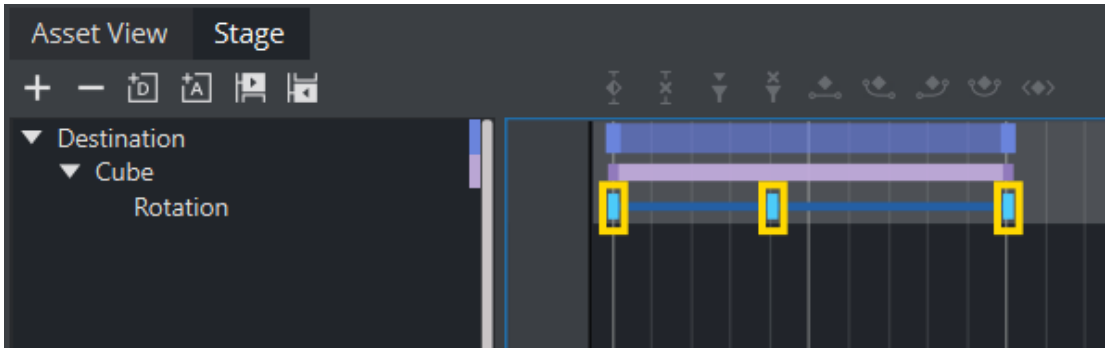
Copyright © 2026 Vizrt

17.5 Key Frames

A Key Frame stores information about a property at a given time. Key Frames can be added to Channels and Action channels. If two Key Frames next to each other hold different values, the Key Frames are connected with a bar in the Dopesheet Editor.

When selecting a Key Frame in the Dopesheet editor or the Spline Editor, the corresponding Key Frame in the other editor is selected too. Working with Key Frames in the Dopesheet enables the Key Frame Editors.

Example: The position of a cube can be at X=0 at field 0 and X=100 at field 50. In such a scenario, two Key Frames are created, one at field 0 and one at field 50. Since the position of the cube is different in the two Key Frames, a bar is added between them, indicating that an animation takes place when the time-line moves from Key Frame A to Key Frame B:



This section contains information on the following procedures:

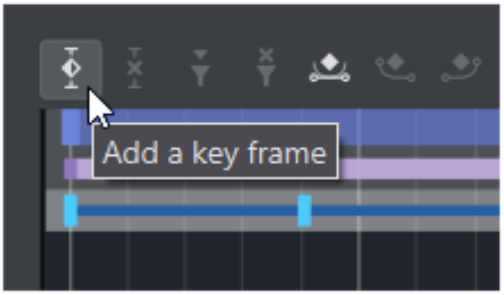
- To Add a Key Frame
- To Select One or More Key Frames
- To Move One or More Key Frames
- To Move a Key Frame to a Specific Time
- To Copy a Key Frame
- To Rename a Key Frame
- To Delete a Key Frame
- Reverse the Order of Selected Key Frames

17.5.1 To Add a Key Frame

There are four ways to create a Key Frame:

1. Select the required **Channel** or **Action Channel** in the Stage Tree.
2. Set the Time-line Marker to when the Key Frame should be added.
3. Click (Add a Key Frame).

Copyright © 2026 Vizrt



Or:

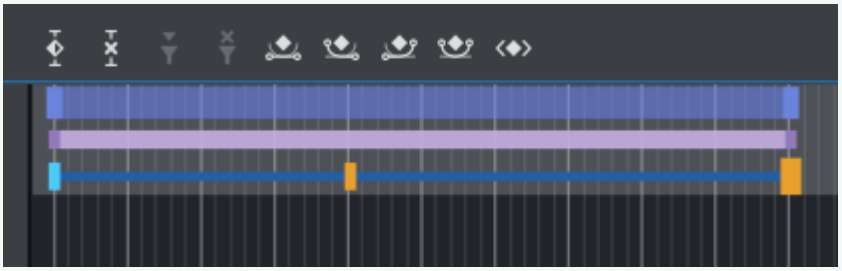
4. Select the required **Channel** or **Action Channel** in the Stage Tree.
5. Set the Time-line Marker to when the Key Frame should be added.
6. Set the Key Frame:
   - Click in the Time-line Editor, or
   - With the mouse pointer over the Scene Editor, press **ENTER**.

   Or:
7. Select the required **Channel** or **Action Channel** in the Stage Tree.
8. Set the Time-line Marker to when the Key Frame should be added.
9. In the Spline Editor, make sure to hold the pointer above the spline at the required time-line location (the spline should turn orange if doing this correctly).
10. Click the spline.

17.5.2 To Select One or More Key Frames

- Click on a Key Frame to select it. This clears the selection of any previously selected Key Frames.
- **CTRL +** click on a previously selected Key Frame to clear the selection of it.
- **CTRL +** click on Key Frames to have more than one selected.
- **SHIFT +** click on a segment between two Key Frames to select both.

Tip: You can also hold down the **R** key and use the rubber band to select multiple Key Frames.



Note: When several Key Frames are selected, actions like move or delete affect all selected Key Frames.

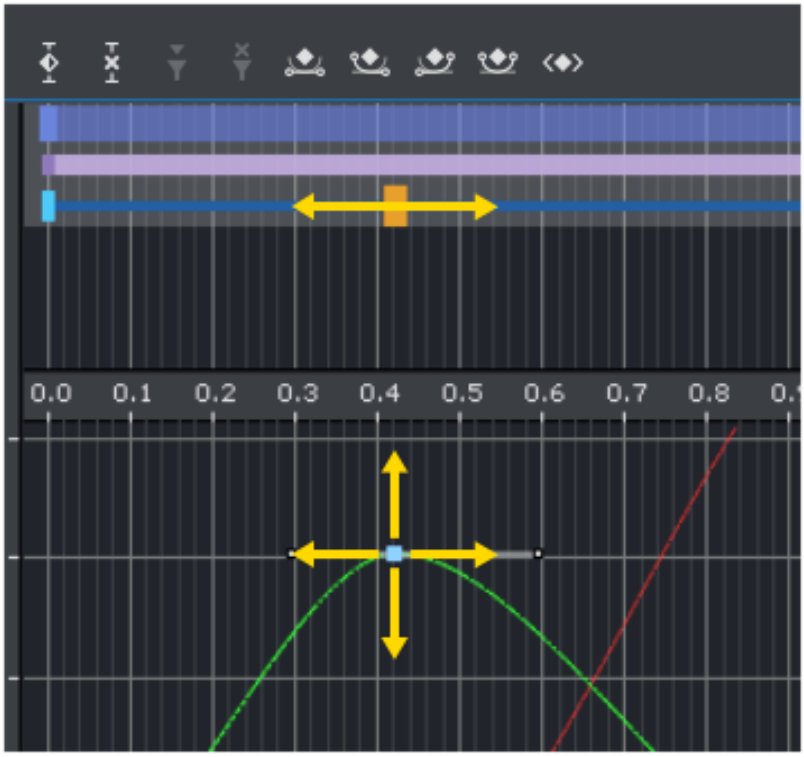
17.5.3 To Move One or More Key Frames

1. Select one or more Key Frames.
2. Drag the Key Frames along the time-line in the Dopesheet Editor.
   Select one or more Key Frames.

Copyright © 2026 Vizrt

3. Drag the Key Frames along the time-line (horizontally) in the Spline Editor.

Note: Dragging a Key Frame vertically in the Spline Editor changes the value of the Key Frame, not the time-line value.



17.5.4 To Move a Key Frame to a Specific Time

1. Enable the Key Frame Editors.
2. Select a Key Frame.
3. In the Key Frame editor, enter the time (in fields) in the **Time** input box.

17.5.5 To Copy a Key Frame

1. Select one or more Key Frames.
2. Press **C** while dragging the Key Frames to the new position in the Dopesheet Editor.

17.5.6 To Rename a Key Frame

1. Enable the Key Frame Editors.
2. Select a Key Frame.
3. In the Key Frame editor, enter a descriptive Key Frame name in the **Name** text box, and press **ENTER**.

Copyright © 2026 Vizrt

Note: Special characters, such as space, are not allowed in Key Frame names.

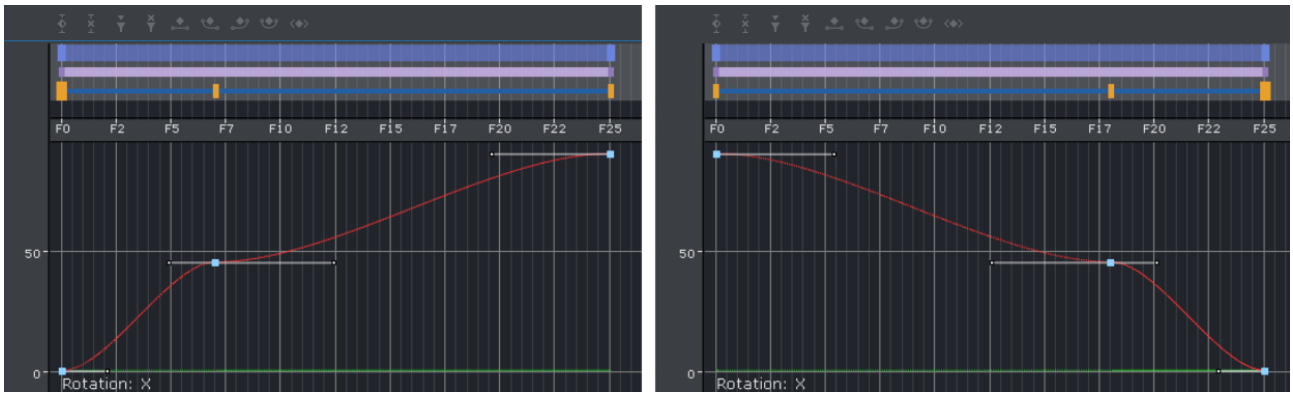
17.5.7 To Delete a Key Frame

1. Select one or more Key Frames.
2. Click the (Delete Key Frame) button.
   Or:
3. Select one or more Key Frames.
4. Press **DELETE**.

17.5.8 Reverse the Order of Selected Key Frames

Reversing the order of Key Frames is like mirroring the Key Frames.

Example: Image 1 (below) illustrates a rotation animation, with Key Frames placed at fields 0, 7, and 25. The rotation value at field 0 is X=0, at field 7 X=45, and at field 25 X=90. After the Key Frames have been rotated, image 2 illustrates the animation with Key Frames placed at fields 0, 18, and 25. The rotation value at field 0 is X=90, at field 18 X=45, and at field 25 X=0.



1. Select two or more Key Frames in the Dopesheet Editor.
2. While holding the mouse pointer over the Stage, Press **CTRL + R**.

Copyright © 2026 Vizrt

17.6 Directors

To create an animation, there must be at least one Director in the Stage Tree. Directors are used to group a certain set of Stage items, such as Actors and Channels. It is possible to create Directors at the root level of the Stage Tree, or as Sub-Directors.

A Director can hold as many items as required, but to improve usability it is recommended to group certain items in different Directors, especially when working with complex animations. In addition to improved overview in the Stage Tree, multiple Directors make it possible to show only the animations contained within each Director. This opens the possibility to, for example, have multiple animation channels on one object, driven by multiple Directors.

Each Director can also hold controlling items such as stop points and actions that only affect the animations within the Director (see Basic Animation Functions).

> **Note:** The first Director created in the Stage is always called **Default** Director. Any subsequent Directors are called **Director** (rename as required).

Most actions on Actors are done using Directors. Actions affecting the time-line are performed directly in the Dopesheet Editor. Working with Directors in the Dopesheet editor enables the Director Editor.

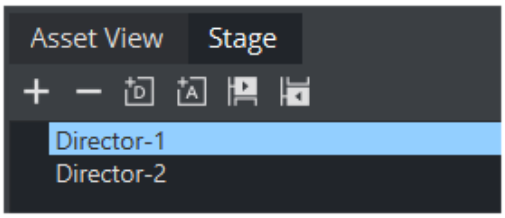
This page contains information on the following topics:

- Directors in the Stage Tree
  - To Create a Director
  - To Move a Director
  - To Copy a Director
  - To Rename a Director
  - To Delete a Director
- Directors in the Dopesheet Editor
  - To Move a Director in the Time-line
  - To Speed Up or Slow Down a Director
  - To Stretch a Portion of a Director
- Events in a Director
  - To Create a New Event
  - To Change an Event Type
  - To Move an Event along the Time-line
  - To Name an Event
  - To Delete an Event
  - To Create a Tag (Preview Point) for External Applications
  - To Create a Tag (Preview Point) for Viz Trio
  - Preview Callback Events

17.6.1 Directors in the Stage Tree

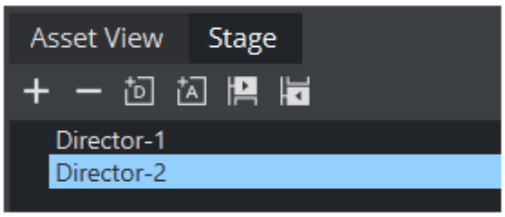
Directors are shown in the Stage Tree with a gray blue background color.

Copyright © 2026 Vizrt



To Create a Director

Note: The first Director created in the Stage is always called **Default** Director. Any subsequent Directors are called **Director** (rename as required).



- Click the (Add Director) button
  Or
- Drag the (Add Director) icon to the Stage Tree:
  Directors that are dragged to the Stage Tree can be positioned in various ways, and create either root or Sub-Directors.

Note: The new Director is added under the currently selected Director. If no Director is selected then its added at the end of the director tree.

To Move a Director

Drag one or more Directors to a different place in the Stage Tree hierarchy

To Copy a Director

Press **CTRL** and drag one or more Directors to a different place in the Stage Tree hierarchy

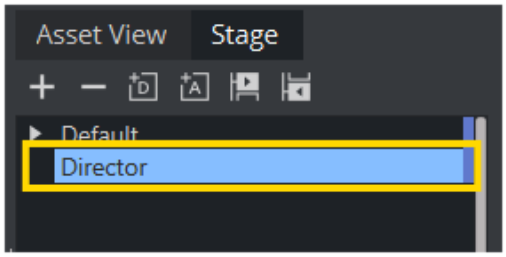
To Rename a Director

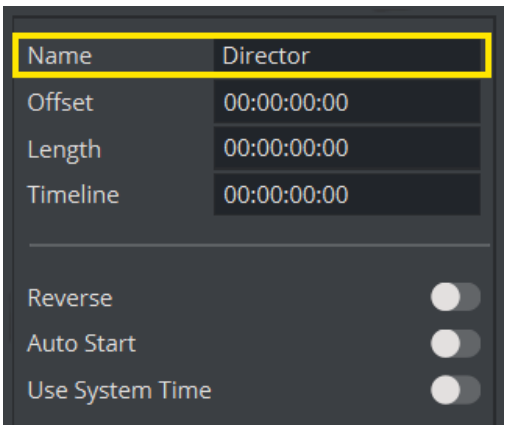
Note: Special characters, such as space, are not allowed in Director names.

1. Right-click a Director.
2. In the context menu, select **Rename**.
3. Type a new name.

Copyright © 2026 Vizrt

4. Press **ENTER**.
   Or:
5. Select the required Director in the Stage Tree.
6. Open the Director Editor.
7. Type a descriptive name in the **Name** text box.
8. Press **ENTER**.





To Delete a Director

1. Right-click a Director in the Stage Tree.
2. In the context menu, select **Delete Director**.
   Or:

- Drag the Director, or multiple Directors, to the trash can
  Or:
- Select a Director, or multiple Directors, and press **DELETE**.

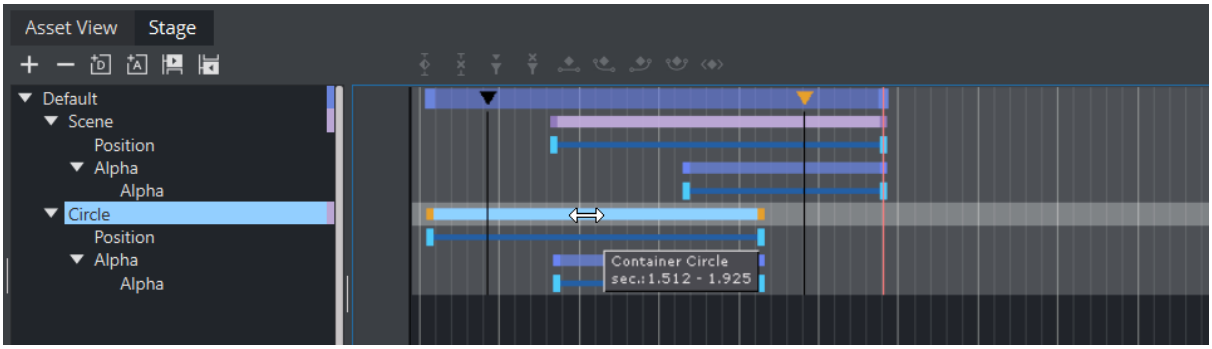
17.6.2 Directors in the Dopesheet Editor

Directors are shown in the Dopesheet Editor with a gray-blue color.

To Move a Director in the Time-line

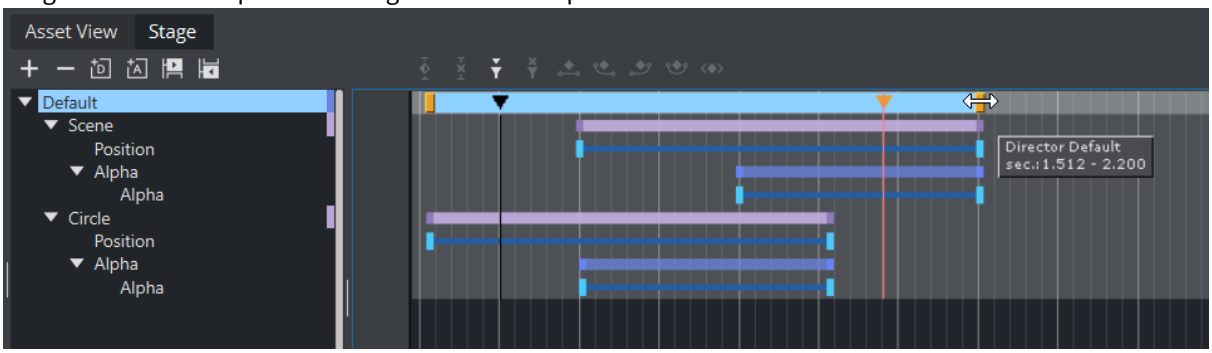
- Click on the Director and drag to a different time.

Copyright © 2026 Vizrt



To Speed Up or Slow Down a Director

- Click on the start or the end of the Director.
- Drag the start or end point to change the Director speed:

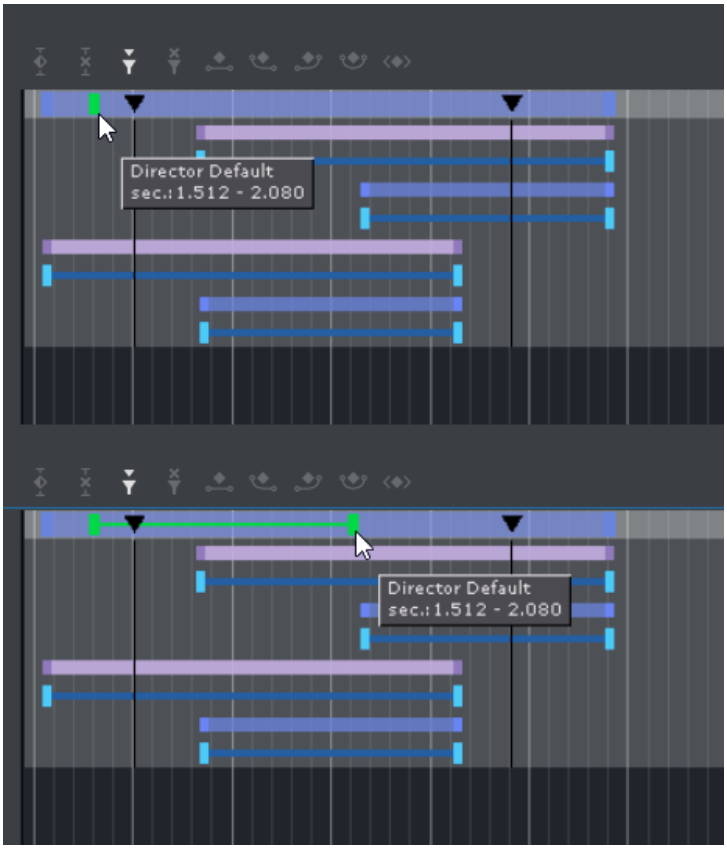


To Stretch a Portion of a Director

If only a part of a director needs to be lengthen or shortened, you can use the rubberband function.

- Keep the **T** key pressed click on the Start of your Selection. A green marker appears.
- Keep **T** key pressed again and select your end point.
- Stretch or shrink your selection by using the two green handles.

Copyright © 2026 Vizrt



17.6.3 Events in a Director

Events can be added to a Director to add control to an animation. Events that can be added to the Director are:

- **Stop:** Stops the animation until continue animation is clicked. A Stop applies for the Director and all Sub-Directors.
- **Tag:** Shows a position along the time-line. This Event type does not stop the animation.
- **Local Stop:** Stops the animation until continue animation is clicked. A local stop only applies for the Director it is placed on, not its Sub-Directors.
- **Pause:** Stops the animation for a specified amount of frames.
- **Next Scene:** Marks a position where a Scene ends. The next Scene starts automatically.

Note: Viz Weather has an automatic play list option where the user does not need to start the next Scene. The Next Scene tag starts the next Scene automatically when the current Scene reaches the Next Scene Tag.

To Create a New Event

1. Click on a Director in the Stage Tree.
2. Set the Time-line Marker to the point where the new Event is to be added.
3. Add an Event:
    - Click the (Add Stop/Tag) button, or

Copyright © 2026 Vizrt

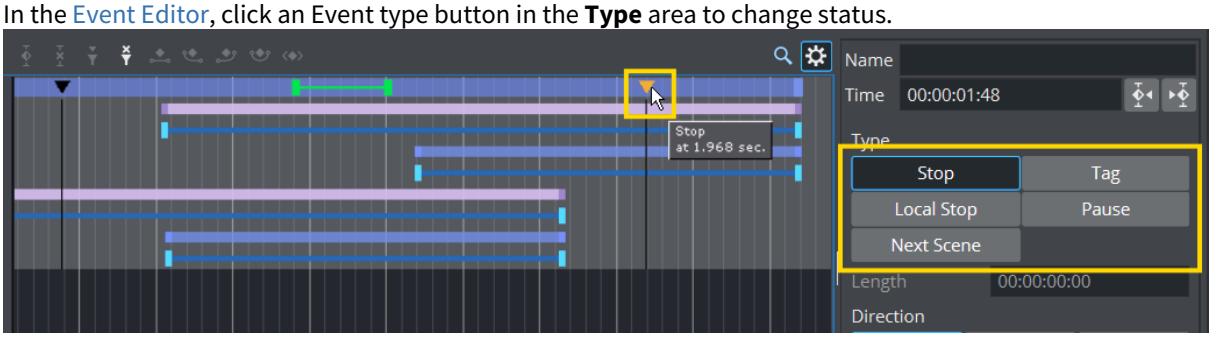
- Press **S** and click in a Director where the new Event is to be added.

4. If required, change the Event type.

> **Note:** The default new Event is a **Stop** Event.

To Change an Event Type

1. Enable the Event Editor in the Stage.
2. In the Director, select the Event to be change.
3. In the Event Editor, click an Event type button in the **Type** area to change status.



4. If **Pause** is selected: In the Value box, define the number of frames that the animation should pause for, before it continues.

To Move an Event along the Time-line

1. Select one or more Events.
2. Drag the selected Events along the Director Time-line.

> **Note:** To select more than one Event press and hold **CTRL** and click on each required Event.

or

3. Enable the Event Editor in the Stage.
4. Select an Event in the Director.
5. Move the selected Event:
   - In the Event Editor, enter the new position (in frames) in the **Time** input box, or
   - Click in the **Time** input box and drag left or right.

> **Note:** This moves only the last selected Event in the Director.

To Name an Event

1. Enable the Event Editor in the Stage.
2. Select an Event in the Director Time-line.
3. In the Event Editor, enter a descriptive name in the **Name** text box.
4. Press **ENTER**.

> **Note:** Special characters, such as space, are not allowed in Event names.

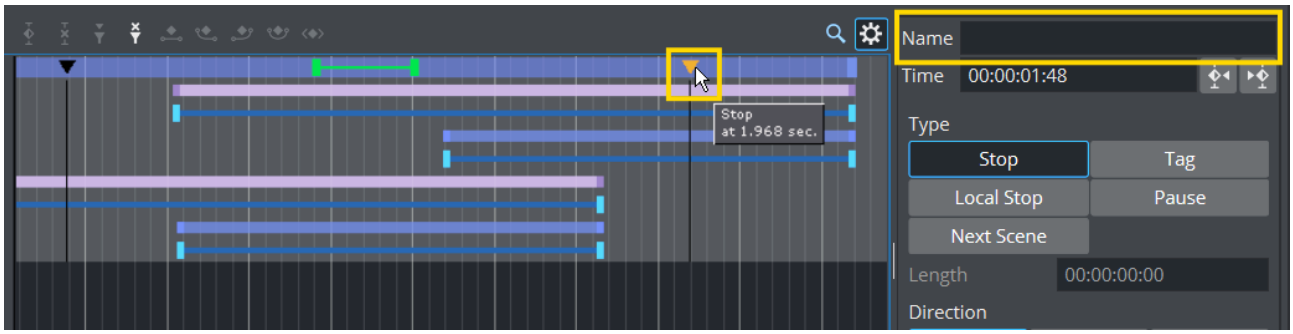
Copyright © 2026 Vizrt

To Delete an Event

1. Select the Event to be deleted.
2. Click the (Delete Stop/Tag) button.
   or
3. Select one or more Events.
4. Press **DELETE**.

To Create a Tag (Preview Point) for External Applications

For example, in Viz Trio and Viz Pilot:



1. Open the Stage.
2. Select the default Director.
3. Position the Time Marker at the point where the Scene is to be previewed.
4. Click the (Add Stop/Tag) button to add an Event.
5. In the Event Editor, enter `pilot1` (case sensitive) in the **Name** text box. The name *pilot1* is the Tag name requested by control application, and is also the default preview point for an Object scene in Transition Logic-enabled scenes. For labeling more Tags, or preview points, continue with *pilot2*, *pilot3*, etc. Make sure *Tag* is selected in the **Type** area.

> **Note:** Custom Tag Events can be used as well. However, those Tags Events are only selectable within the Viz Pilot Newsroom client.

To Create a Tag (Preview Point) for Viz Trio

1. Open the Stage, and select the default Director.
2. Position the Time Marker at the point where the Scene is to be previewed.
3. Click the (Add Stop/Tag) button to add an Event.
4. In the Event Editor enter the **Name** as **01**, **02**, **03**, etc. according to the value of the corresponding Field identifiers (01, 02, 03, etc.). Setting these points previews the Scene in the local Viz Trio preview window when a tab field in Viz Trio is selected (01, 02, 03, etc.).

Copyright © 2026 Vizrt

Preview Callback Events

IMPORTANT! This event must only be triggered for the local preview in the External Control Application, and not for any external preview or program engines.

Viz Engine supports a script event called **OnPreview** upon loading a scene in an External Control Application, such as Viz Trio or Viz Pilot. This gives the designer additional capabilities for when the scene is loaded in the local preview, as it can be used to show or hide parts of the scene to make it clearer for the operator what is shown later in the animation, or to write some instructions for the operator, while the scene should be taken On Air without changes.

It may also be used to compensate for unavailable capabilities on a preview Engine. For example, if a scene plays a video in the background, which cannot be previewed in Engine, a back plate can be shown in the preview with a snapshot of the video. This gives the operator an idea of how the scene looks when taken On Air. It could also be used as an optimization, to save the local preview engine from loading large video files.

Note: Preview Callback Event support requires Viz version 3.8.1 or higher.

The **OnPreview** script event can be used in a scene or container script. If adding the script as a container script, make sure it is added to the same container as the Control Object plug-in. The script can also be loaded as both a scene and a container script, in which case both scripts are invoked when the scene is loaded.

In Transition Logic scenes, the function can be placed in the scene script of the Master scene, or as a container script on the Control Object container in one or more of the Object scenes. In Transition Logic scenes, the scene script of the Master scene, *and* the container script on the root Control Object container of all Object scenes, is invoked.

A simple **OnPreview** script event handler could be like:

```
sub OnPreview(doPreview as Integer)
if doPreview then
println("4015 Container OnPreview")
end if
end sub
```

See Also

- Transition Logic
- Scripting

Copyright © 2026 Vizrt

17.7 Create a Stand-alone Scene

This section details how to create a basic lower third as a stand-alone Scene. This scene can then be imported to a Viz Trio to create a template for pages which are used for playback.

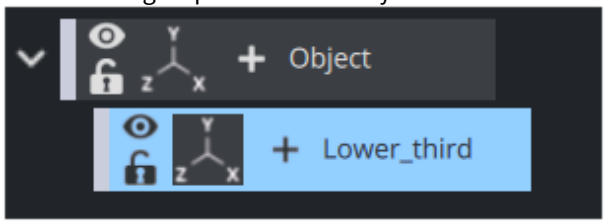
Note: For ease of use, the Scene example has been split up in several procedures. For the example to work, it is recommended to follow each procedure in the order given.

This section contains information on the following topics:

- To Create a Scene
- To Add a Background
- To Add Text
- To Create an In and Out Animation
- To Add Stop Tags
- To Add Key Functions to the Container
- To Add Expose Properties
- To Edit Multiple Items with a Single Value

17.7.1 To Create a Scene

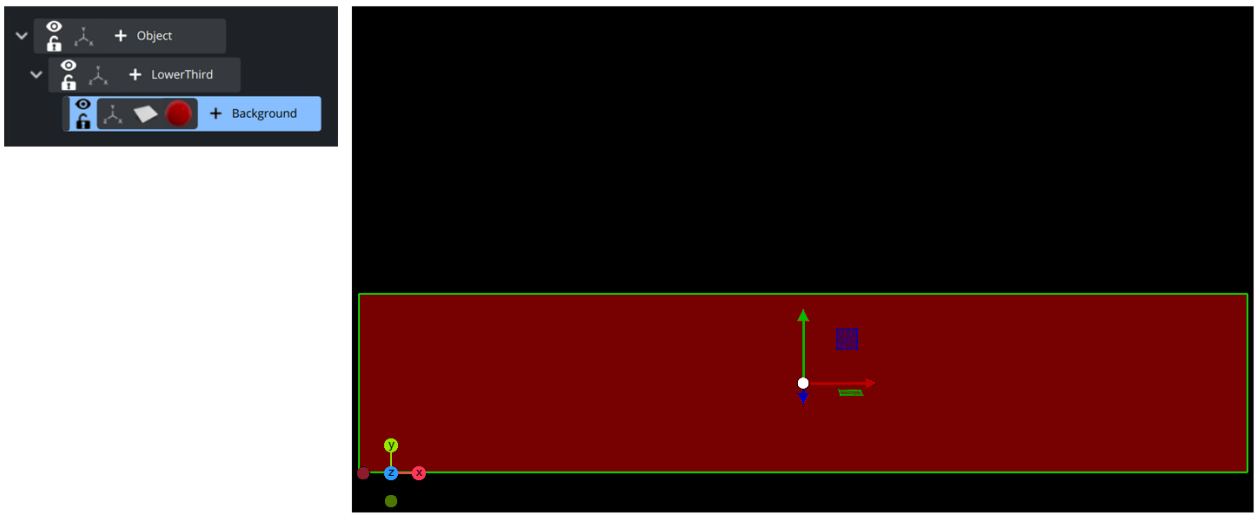
1. Add a group container to the Scene Tree.
2. Rename the group container to *object*.



3. Add second group container as a sub-container to the *object* container.
4. Name the second group container *lower_third*.
5. Save the new scene as *stand_alone_scene* (in this example).

Tip: For ease of use set the Scene Editor Title (TA) and Safe (SA) areas to `On` .

17.7.2 To Add a Background



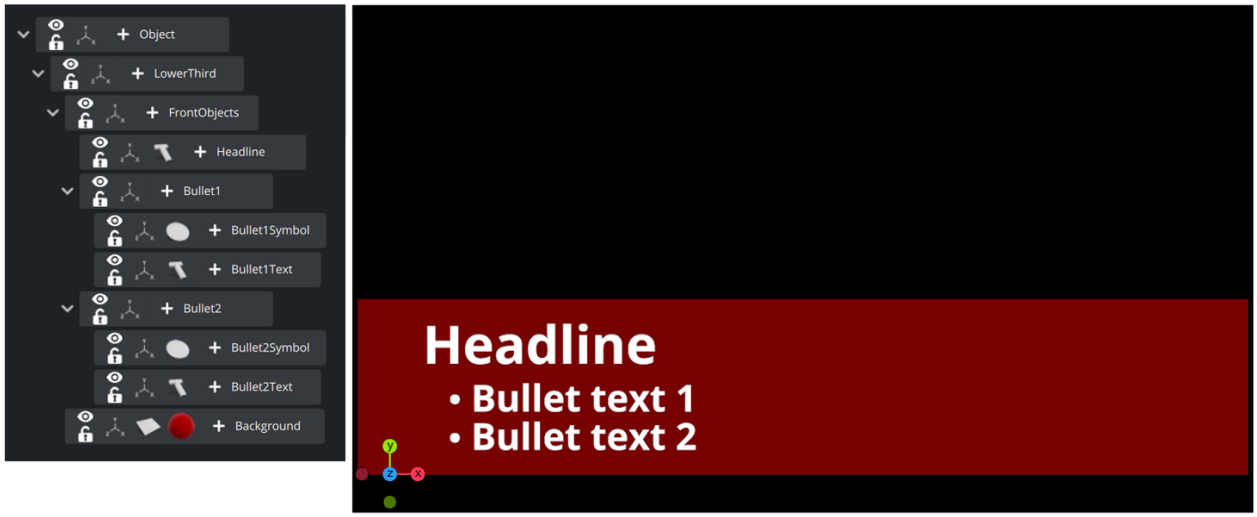
1. Add a Rectangle geometry as a sub-container of the *lower_third* container,
2. Rename it to *background*.
3. Position the rectangle object so that it covers the lower third of the screen:
    - Rectangle editor:
        - **Width:** `800`
        - **Height:** `125`.
    - Transformation editor:
        - **Position Y:** `-160`
        - **Position Z:** `-100`.
4. Add a Phong material object to give the Rectangle a color.

> **Note:** Position Z: Since this item is a background item it should be a little bit behind the other objects on the Z-axis.

17.7.3 To Add Text

Add a headline item and two text lines, lead by a bullet symbol.

Copyright © 2026 Vizrt

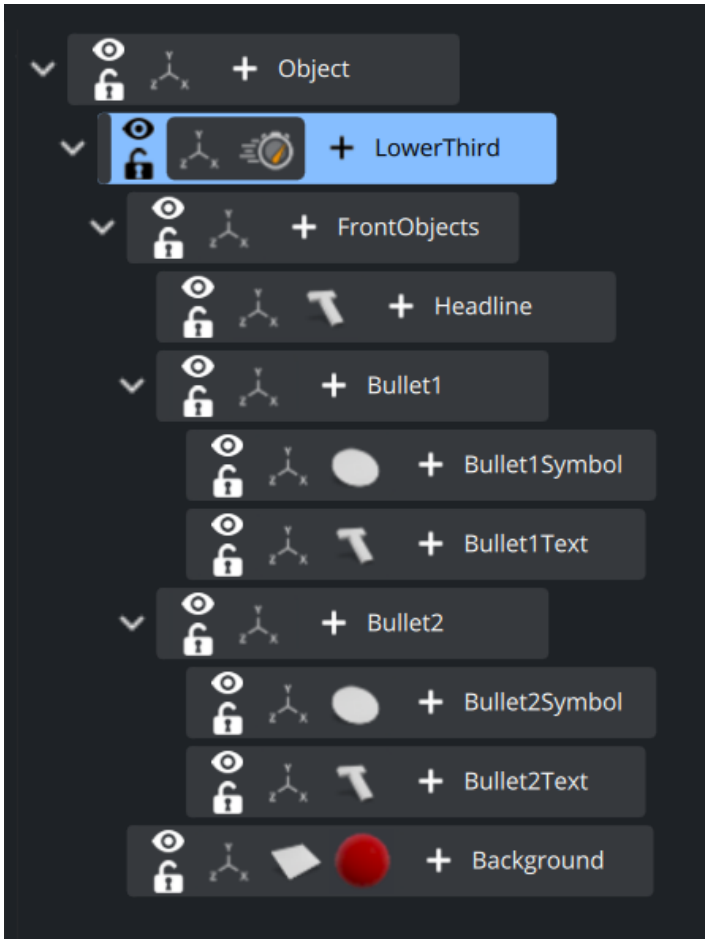


1. Add a container as a sub-container to the *lower_third* container.
2. Name the container *front_objects*.
3. Add a Fontstyle as a sub-container of the *front_objects* container.
4. Name the Fontstyle container *headline*.
5. Open the Fontstyle text editor and:
    - Enter the word `Headline` in the **text input** field.
    - Set the Horizontal orientation to **Left**.
6. Add a material to the *headline* container.
7. Open the transformation editor of the *headline* container.
8. Scale and Position the Fontstyle so that it is placed in the upper left corner of the lower third background.
9. Add a group container as a sub-container of the *front_objects* group.
10. Name the container *bullet_1*.
11. As sub-containers of *bullet_1*, add:
    - A Fontstyle: Name the container *bullet_text*.
    - Any Image or Object that can function as a bullet symbol: Name the container *bullet_symbol*.
12. In *bullet_text* text editor:
    - Enter the text: Bullet 1 text here
    - Set the horizontal orientation to **Left**.
13. Add a material to the *bullet_text* container.
14. Scale and Position the *bullet_text* and *bullet_symbol* containers.
15. Position the *bullet_1* container under the *Headline* text.
16. Make a copy of the *bullet_1* container as a sub-container of the *front_objects* group.
17. Rename the copy to *bullet_2*.
18. Adjust the **Position Y** of *bullet_2*.
19. In *bullet_text* text editor enter the text: Bullet 2 text here.

17.7.4 To Create an In and Out Animation

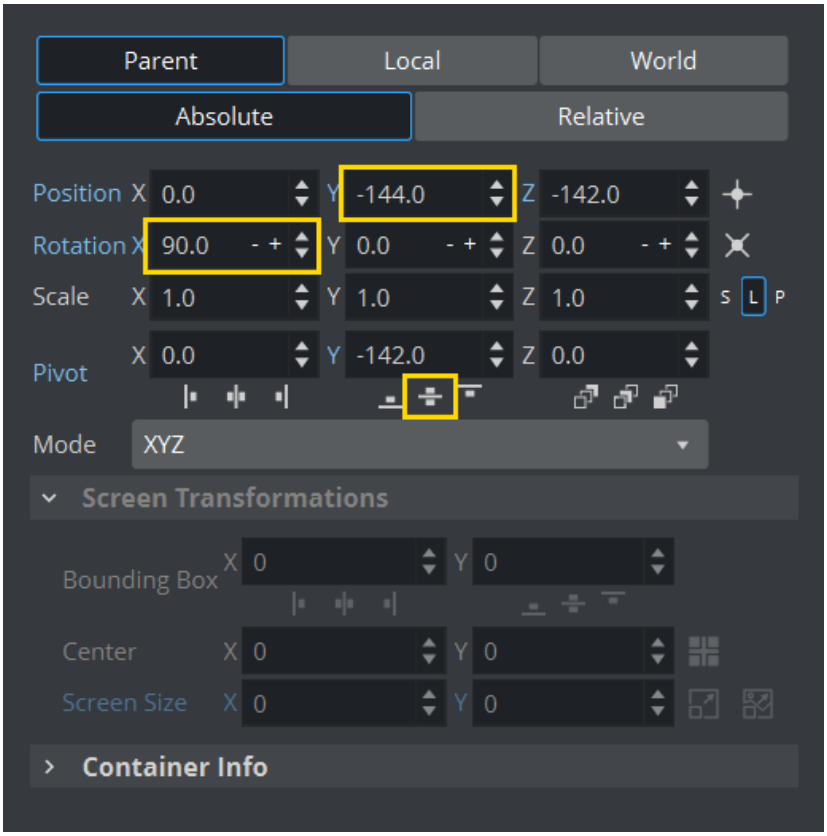
To make the whole lower third rotate in from the left side of the screen and then fade out, the *lower_third* container must be animated.

Copyright © 2026 Vizrt

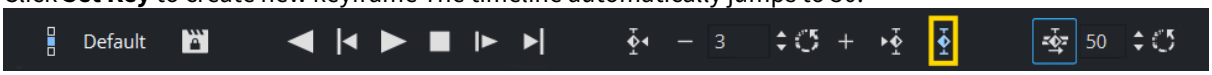


1. Open the Transformation Editor of the *lower_third* container.

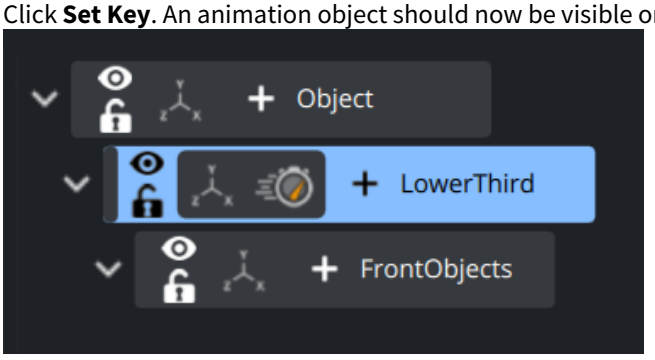
Copyright © 2026 Vizrt



2. Move the **Y-Position** of the container to the lower of the screen.
3. Click on **Center** of **Axis Center Y** to make sure the center point is in the middle of our graphics.
4. Change the **Rotation X** value so the container rotates and is not visible anymore.
5. Click **Set Key** to create new keyframe The timeline automatically jumps to 50.



6. Click on the blue **Rotation X** label to reset it back to zero. This will change the **Rotation** value to `0`.
7. Click **Set Key**. An animation object should now be visible on the **object** container.

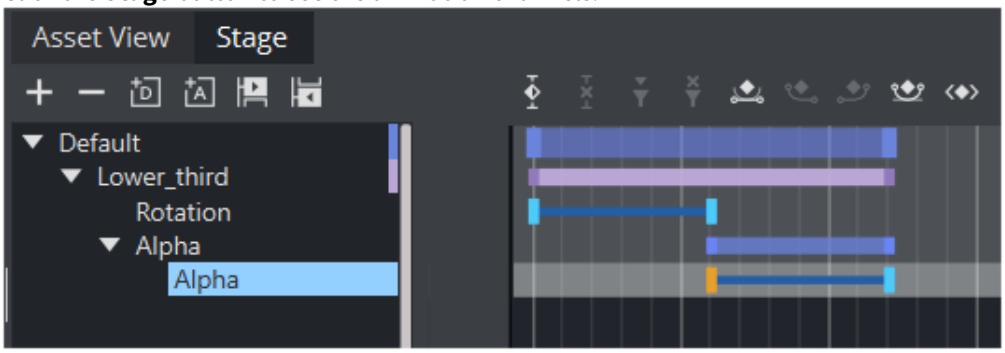


8. Play the animation in the Scene Editor.
9. To create the out animation - add the Alpha plug-in to the *lower_third* container.
    - Click on the + icon on the container, type "alpha" or find the Alpha plug-in and click on the icon.

Copyright © 2026 Vizrt



10. Open the Alpha editor and:
    a. Set the Alpha value to `100.0%`.
    b. Press the **Set Key** button.
    c. Set the alpha value to `0.0%`.
    d. Press the **Set Key** button.
    This has created an alpha animation in addition to the rotation animation.
11. Click the **Stage** button to see the animation channels.



Note: It is possible to move the Key Frames to achieve a different timing. Either drag with the mouse or click on them and alter the time settings in the Key Frame Editors.

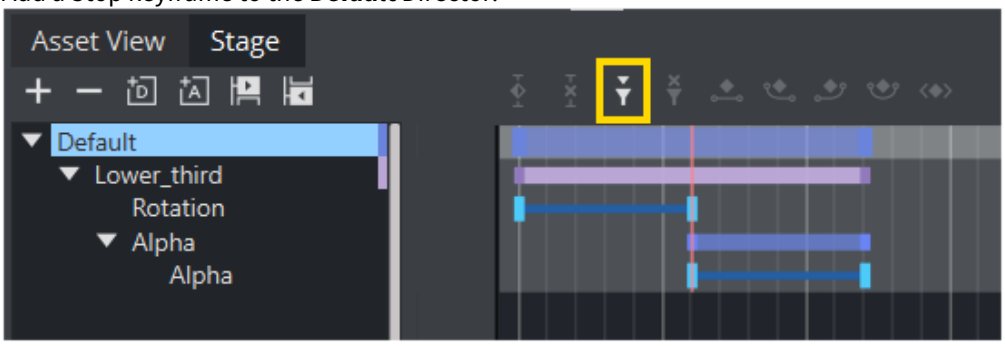
17.7.5 To Add Stop Tags

When the Scene is played out, the objects should rotate in from the left and then fade out. To stop the animation, after the rotate in, a stop point must be added before the animation can continue and fade out.

1. Move the time-line (the thin red vertical line) to the second key frame of the Rotation channel.

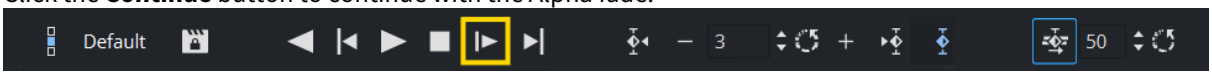
Note: An exact spot is not needed as it can be adjusted afterwards.

2. Add a Stop keyframe to the **Default** Director.



Copyright © 2026 Vizrt

3. Drag the Stop point (or use the editor that opens when it is selected) so that it has the same time settings as the second keyframe on the Rotation channel.
4. Play the animation to check that the animation stops when the rotate in has finished.
5. Click the **Continue** button to continue with the Alpha fade.

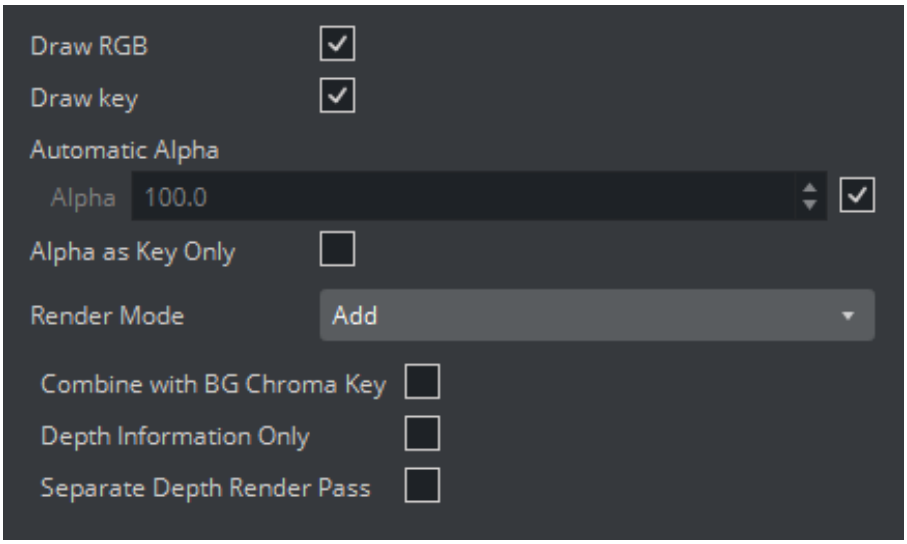


17.7.6 To Add Key Functions to the Container

1. Add the Key plug-in to the *background* container.
   - **Built Ins > Container plug-ins > Global**



2. Click the Key plug-in icon to open its editor:
   - Set **Alpha as Key only** to `On`. This is to avoid a dirty key if the background has some level of transparency.
   - Set **Render Mode** to **Blend**.



3. Add a Key plug-in to the *front_objects* container.
4. Click the Key plug-in icon to open its editor:
   - Set **Alpha as Key only** to `Off`.
   - Set **Render Mode** to **Add**.
     This is to avoid the foreground objects to cut a hole in the Key signal of the background object.

17.7.7 To Add Expose Properties

To make a Scene ready for import into Viz Trio, Control plug-ins must be added. The Control plug-ins enable the designer to expose scene properties to a Viz Trio user.

1. Add the **Control Text** plug-in to the *headline* and both *bullet_text* containers.

Copyright © 2026 Vizrt

- **Built Ins > Container plug-ins > Control**

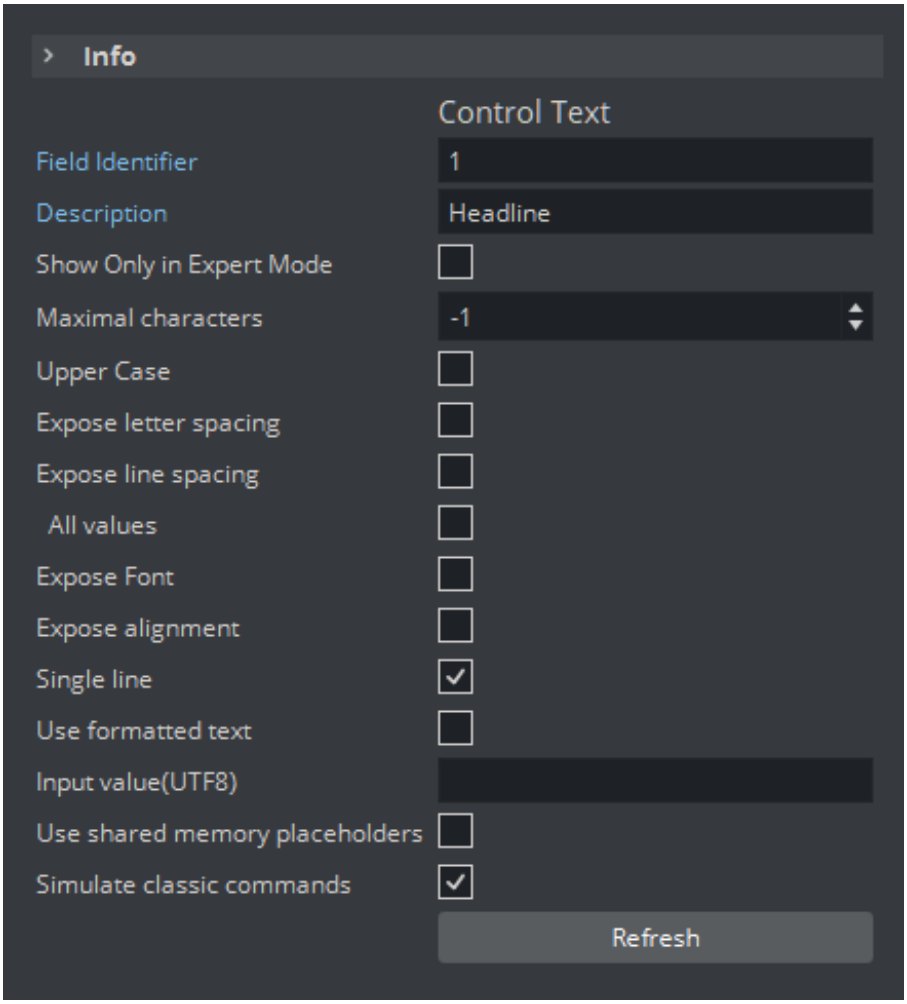




Note: The **Control Object** plug-in is automatically added to the top most container in the Scene Tree, which should always be named *object*.

2. Click on a **Control Text** plug-in icon to open its editor:
    a. Set the **Field identifier**. The Field identifier must be a numeric value and the value is used to give the Viz Trio page a tab order.
    b. Add a **Description**.
    c. Repeat for the other two **Control Text** plug-ins.

Copyright © 2026 Vizrt

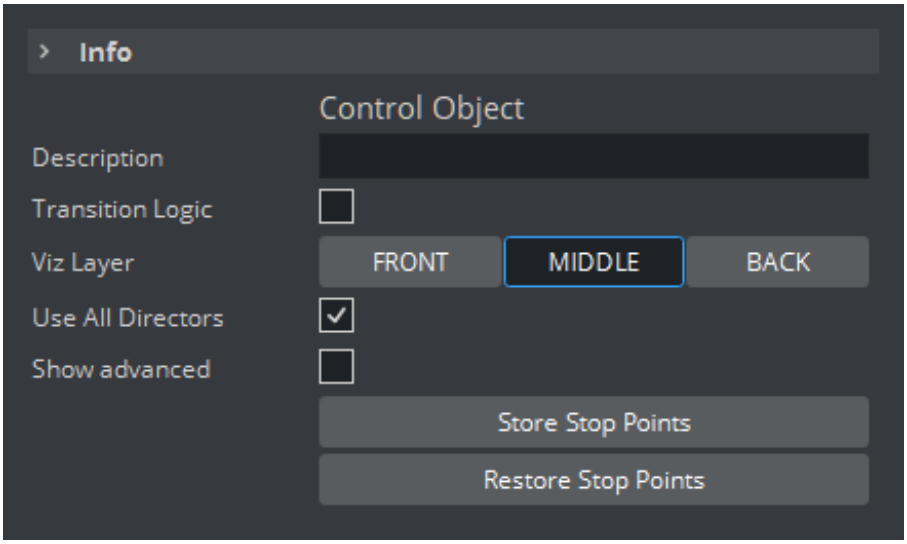


Note: In this scene, the *headline* container typically gets identifier 1, and the two *bullet_text* containers get identifiers 2 and 3.

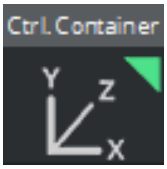
Note: More properties can be exposed to the Viz Trio operator. If the **Position**, **Rotation**, and **Scale** properties are exposed the operator can hide or show an object and add material.

3. Click on the **Control Object** icon to open its editor.
4. Type a **Description** for the Viz Trio template (for example, lower_third).

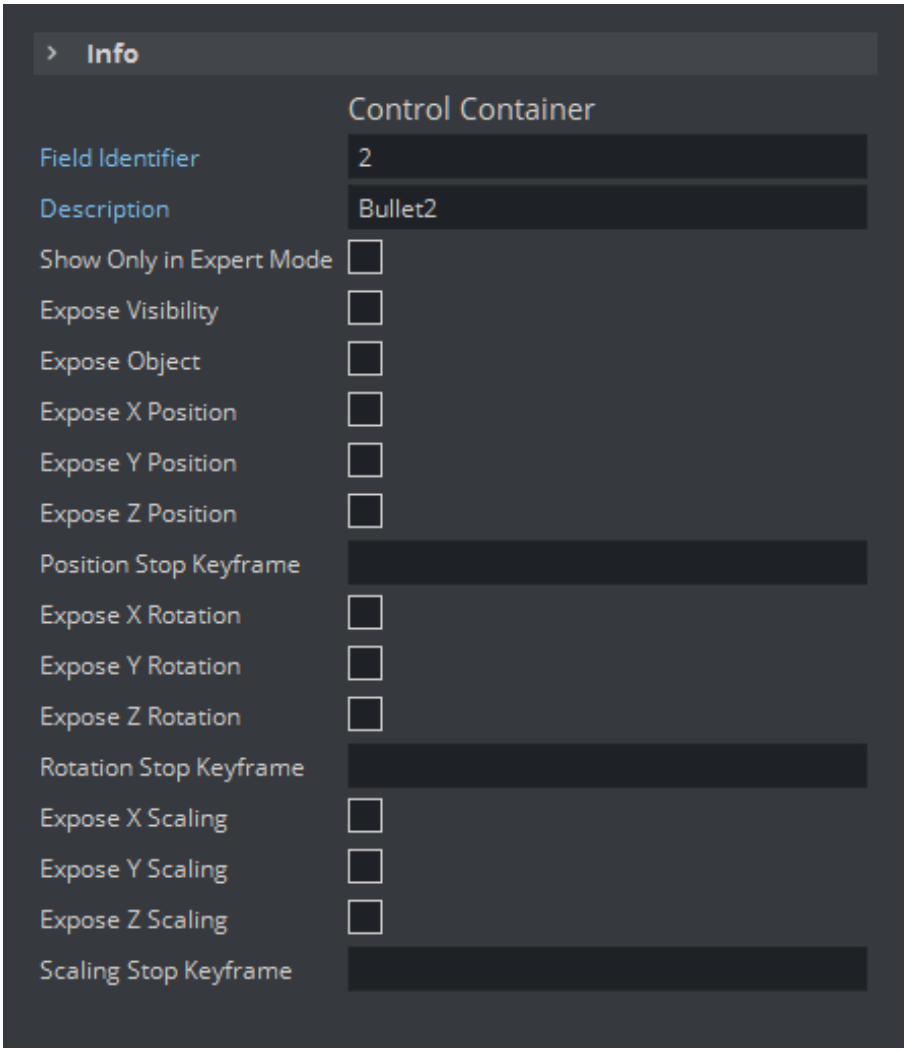
Copyright © 2026 Vizrt



5. Add the **Control Container** plug-in to a container to be exposed (for example, *headline*).



6. Click the **Control Container** plug-in icon to open its editor.



7. Type a **Field Identifier** and expose the y position. This allows an operator to position the lower-third in y position.
    - X/Y/Z properties for a keyframe in an animation must be specified by a Stop Key Frame name.

17.7.8 To Edit Multiple Items with a Single Value

Use the same **Field Identifier** for multiple control plug-ins to make two or more fields as a single item that receives the same value. This can, for example, be used with a bar that scales and a text label that shows its value. The scale value and the text can then be set in a single operation, which makes sure that there is no mismatch between the bar size and the written text value.

Copyright © 2026 Vizrt

17.8 Create an Over the Shoulder Scene

An Over the Shoulder (OTS) graphic is a graphic which is placed over the shoulder of, for example, the news anchor, and can show on the left or right side of the screen. An OTS graphic can be used to highlight a story with a large image and a headline, and sometimes a sub-headline.

This section contains information on the following procedures, which when completed in sequence create an OTS Scene:

- To Create a Scene
- To Add an Image and Basic Geometries
- To Add Materials
- To Scale Geometries and Place Them in Z Space
- To Add Text
- To Add Background Items
- To Animate
- To Animate Sub Containers
- To Animate Single Containers
- To Animate Single Objects
- To Animate out
- To Fade out the Graphic with the Alpha Plug-in
- To Add Controls

17.8.1 To Create a Scene

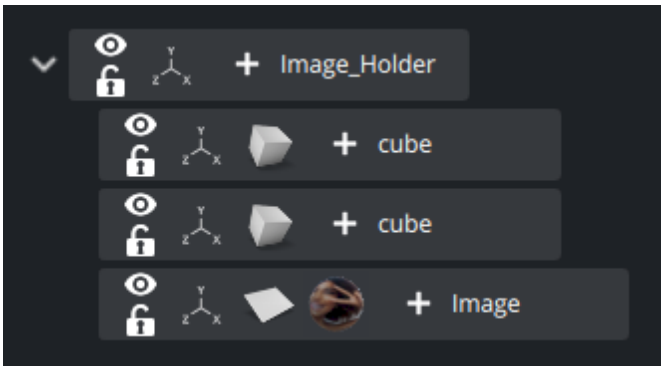
Create a new scene and save it as OTS.

17.8.2 To Add an Image and Basic Geometries

1. Add a Rectangle geometry to the Scene Tree.
2. Add an image (for example, a headshot) to the Rectangle container.
    a. Open the Image Editor, set the mapping to vertex.
    b. Open the Geometry Editor to set the Rectangle **Width** and **Height** to match the image proportions.
3. Rename the Rectangle container to Image.
4. Add a Cube geometry to the scene. Place it above the Image container.
5. Open the Cube plug-in editor and do the following:
    a. Set **Center Y** to `Bottom`.
    b. Adjust the **Size X** value to fit the width of the image.
    c. Adjust the **Size Y** value to `10.0`.
    d. Adjust the **Size Z** value to `1.0`.
6. Copy the Cube container. Place it below the original Cube container.
    - Press **CTRL + C** then **CTRL + V** to place a copy below the original container.
    - Press **CTRL + Left mouse button** to drag and drop a copy to a new location.
7. In the transformation editor of the Cube container, set **Center Y** to `Top`.
8. Rename the Cube containers to *Top_Cube* and *Bottom_Cube*.
9. Open the transformation editor of each Cube container and move the position of the Cubes. Place one above and the other below the image.

Copyright © 2026 Vizrt

10. Group all containers under a new root container:
    - Select all containers and click the **Group** button, or
    - Select all containers and press **CTRL + G**
11. Rename the new group container to *Image_Holder*.

Scene Tree	Output
	

Note: Geometries can also be scaled with the container's transformation editor. But this has an effect on the its proportions when, for example, beveled edges are added.

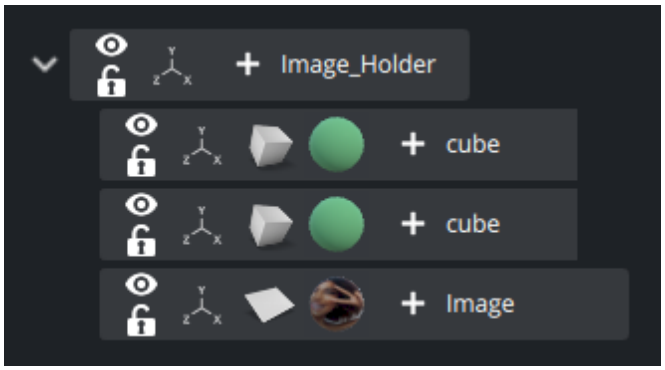
17.8.3 To Add Materials

1. Open the OTS scene.
2. Expand the *Image_Holder* container.
3. Click the **+** button and type `material`.



4. Open the Material Editor and define a color (for example, blue or red).
5. Copy the Material onto the *Bottom_Cube* container:
    - Drag the Material from one container to the other, or
    - Right-click the Material and from the context menu select **Copy MATERIAL** and then **Paste MATERIAL** onto the receiving container.

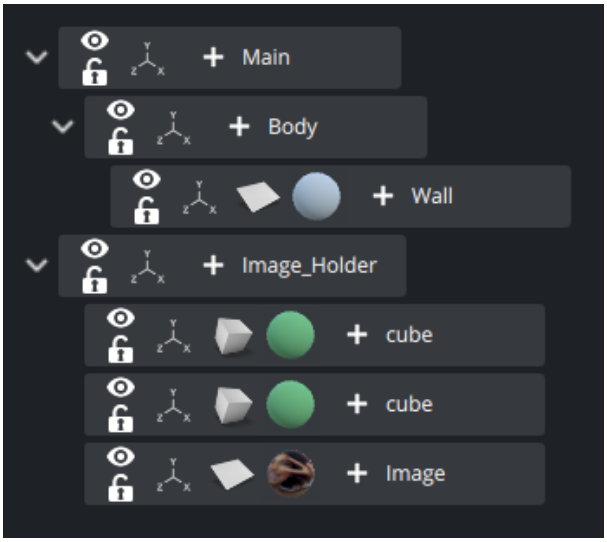
Copyright © 2026 Vizrt

Scene Tree	Output
	

17.8.4 To Scale Geometries and Place Them in Z Space

1. Add a new group container. Place it above the Image_Holder container:
   - Select the *Image_Holder* container and press **CTRL + INSERT**, or
   - Drag the **New Group** button to the Image_Holder container.
2. Rename the New Group container to Body.
3. Add a Rectangle geometry to the scene. Place it as a sub-container of the *Body* container.
4. Rename the new container to Wall.
5. Open the Rectangle editor (not the transformation editor) and do the following:
   a. Set the **Center** option to Center.
   b. Adjust the **Height** value to `450` (for example).
   c. Adjust the **Width** value to `800` (for example).
6. Group the Body and Image_Holder containers under a new container:
   - Select the *Body* and *Image_Holder* containers and click the **New Group** button, or
   - Select the *Body* and *Image_Holder* containers and press **CTRL + G** (when Caps Lock is disabled).
7. Rename the new group container to Main.
8. Copy the Material from one of the Cube containers to the Wall container.
9. Open the *Body* container transformation editor.
10. Adjust the **Position Z** value to `-30.0` (This avoids potential conflicts with the geometries in the *Image_Holder* container).

Copyright © 2026 Vizrt

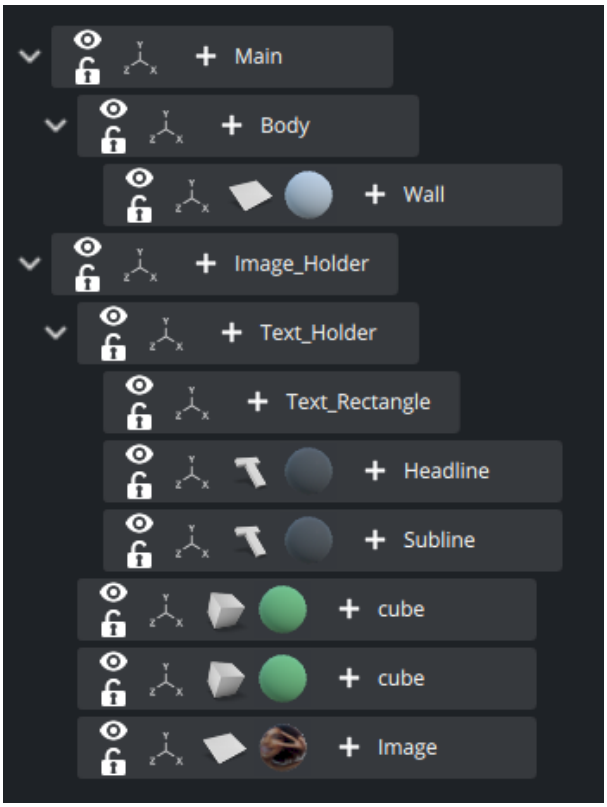


Note: If the position and/or scaling of the Wall geometry need to be adjusted, make sure to adjust those values on the *Body* container, as this keeps future additions to the container aligned.

17.8.5 To Add Text

1. Add a New Group container. Place it below the *Image_Holder* container.
2. Rename the New Group container to *Text_Holder*.
3. Add a Font to the scene. Place it as a sub-container of the *Text_Holder* container.
4. Copy the Font container. Place it below the first Font container.
5. Rename the Font containers to headline and sub-headline.
6. Open the Classic Text Editor of both containers and do the following:
    a. Enter some text (for example, headline, sub-headline).
    b. Set the **Orientation** option to Center.
7. Open the transformation editor of the headline and sub-headline containers and scale and position the containers to fit each other.
8. Add a Rectangle geometry to the scene. Place it above the Font containers and rename it to *Text_Rectangle*.
9. Open the transformation editor of the *Text_Rectangle* container and do the following:
    a. Adjust the **Position Y** value. Center the rectangle between the text objects.
    b. Enable single (**S**) Scaling.
    c. Adjust the **Scaling X** value to equal the width of the image.
    d. Adjust the **Scaling Y** value to `0.1` (for example).
10. Copy the Material from one of the Cube containers to the *Text_Rectangle* container and modify if necessary.
11. Open the transformation editor of the Text_Holder container and do the following:
    a. Adjust the **Position Y** value to equal the *Image_Holder* objects.
    b. Adjust the **Scaling** (Locked) if needed.

Copyright © 2026 Vizrt

Scene Tree	Output
	

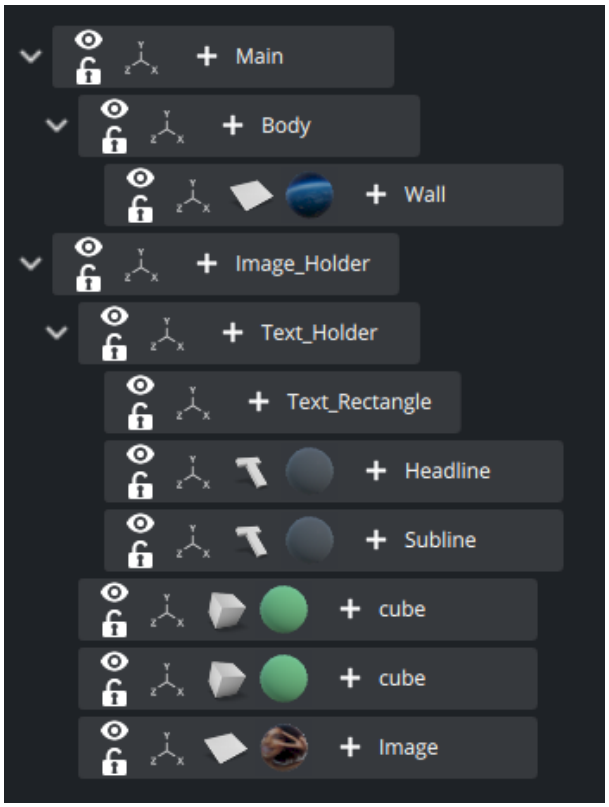
Note: If the position and/or scaling of the font containers need to be adjusted, make sure to adjust those values on the *Text_Holder* container, as that keeps future additions to the container aligned.

17.8.6 To Add Background Items

This procedure details how to add a background image with moving effect.

1. Apply a suitable image to the Wall object. This can have either an Alpha channel assigned or it can be shown as Fullscreen. Don't forget to set the mapping to vertex.
2. Open the transformation editor and do the following:
    a. Adjust the **Scaling** values as required.
    b. Adjust the **Position Z** value (for example, `2.0`) so that it shows in back of the foreground objects.

Copyright © 2026 Vizrt

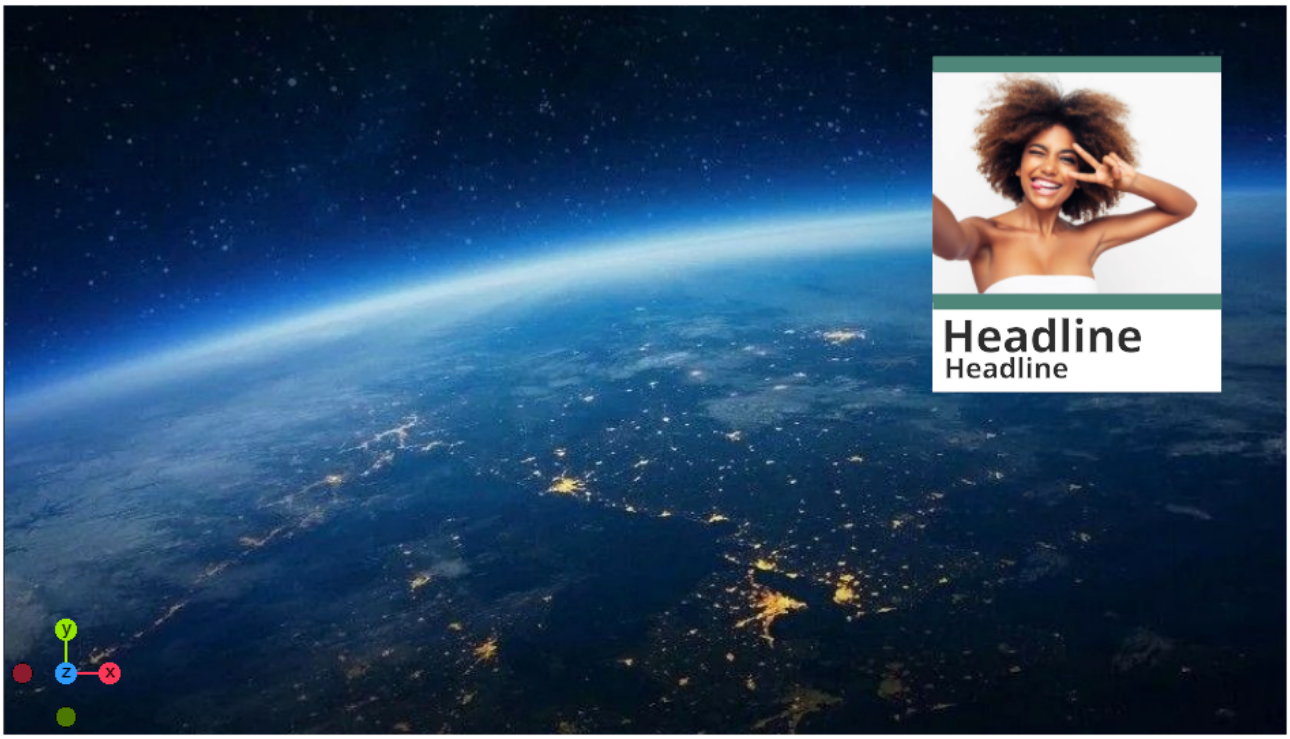
Scene Tree	Output
	

Note: If the position and/or scaling of the background items as a group need to be adjusted, make sure to adjust those values on the *Body* container, as that keeps future additions to the container aligned.

17.8.7 To Animate

Before animations are added, move the OTS graphic to its final on-screen position (normally to the top left or right) where it should be seen by the viewer.

Copyright © 2026 Vizrt

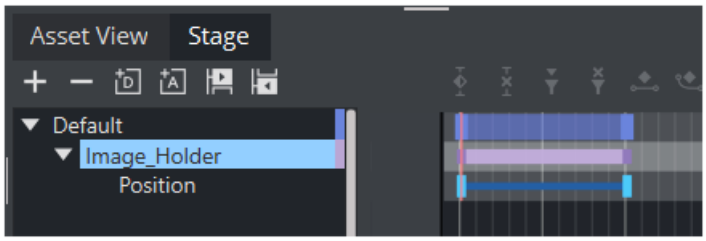


1. Open the OTS scene.
2. Open the transformation editor of the Main container.
3. Adjust the **Scaling**, **Rotation**, and **Position** values. Make sure any conflicts are resolved now. Notice that when the Scene is rotated, objects may drift apart as they are placed at different depths (Z space).
4. Save the scene. This procedure focuses mainly on the animation of a root container, mainly to start with the simplest factor (how do I want my graphics to animate into view?).
5. Open the OTS scene.
6. Set Key Frames to `60/50` (for example) fields.
7. Click the **Set Key Frame** button.
8. Open the transformation editor of the *Image_holder* container and adjust the **Position X** value so that the OTS element is off screen.
9. Set Key Frames to `0` fields.
10. Click the **Set Key Frame** button.

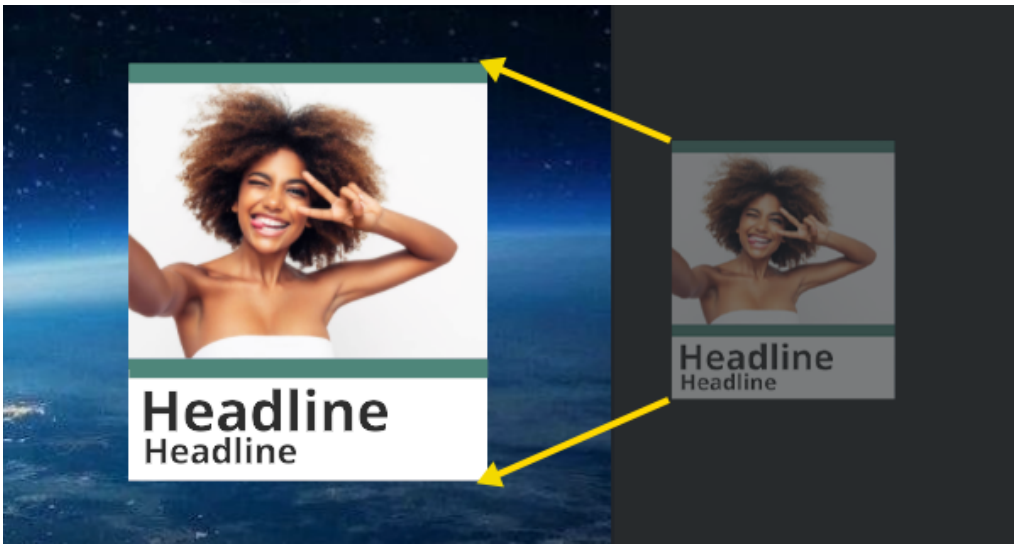
> ⚠ **Note:** Notice that an Animation object has been added to the *Main* container.

11. Change the **Position X** back to the final destination, hit the **Set Key Frame** button.
12. Click the **Start animation** button on the Time-line Editor. The graphic now moves from Position X at `0` fields to its final on-screen position.
13. Open the Stage Editor. The *Image_Holder* container and the animated Position property are located under the Default Director.

Copyright © 2026 Vizrt



14. Right-click the *Image_Holder* container and from the context menu select **Transformation > Scaling**. This adds a time-line for the **Scaling** property of the Main container.
15. Click the first Key Frame on the Scaling animation's time-line on the Dopesheet Editor. This opens the Scaling animation Stage Object Editor.
16. Set the **Scaling** values to `0.5`.
17. Click the **Start animation** button on the Time-line Editor. The graphic now animates from Position X at `0` fields with Scaling at `0.5` to its final size on-screen.



Note: If the Viz Configuration is set to 50 hertz the default animation time-line is 50 fields. If set to 60 hertz, it is 60 fields. It is always important to configure the required output format before work is started on the designs.

17.8.8 To Animate Sub Containers

This procedure focuses on the animation of a sub-containers. In this case the image and text holder containers.

1. Open the Stage Editor.
2. Drag the *Text_Holder* container onto the Default Director.
3. Right-click the *Text_Holder* container and select **Transformation > Scaling** from the context menu. This adds a time-line for the *Text_Holder* container's scaling property.
4. Set values for the Start and End keyframe
5. Click the Key Frame for the Scaling animation. In the Key Frame Editors set it to `30` (for example) fields, to delay the introduction of the Head and Subline.

Copyright © 2026 Vizrt

Note: The *Image_Holder* and *Text_Holder* containers are set to grow from the Center. To grow the containers from other angles, open the transformation editor, for each container, and adjust the **Axis Center** properties. If the OTS is placed to the right, try setting the *Text_Holder* container's Axis Center X to R (Right), and conversely.

Note: Always scrub the time-line in the time-line editor to check for potential conflicts (caused by rotation of objects).

17.8.9 To Animate Single Containers

This procedure focuses on the animation of a single container, and how to add additional Key Frames to the animation. In this case the background image is animated and an additional Key Frame is added to grow the size of the background image before it shrinks back to its final size.

1. Open the OTS scene.
2. Select the *BackgroundImage_Holder* container.
3. Set the number of fields to `30` in the Time-line Editor.
4. Open the Transformation Editor of the *BackgroundImage_Holder* container.
5. Set **Scaling** to `0.0`.
6. Click the **Set Key Frame** button.
7. Set **Scaling** to the original value.
8. Set the number of fields to `60` in the Time-line Editor.
9. Click the **Set Key Frame** button.
10. Open the Stage Editor.
11. Select the **Scaling animation** time-line for the *BackgroundImage_Holder*.
12. Move the Time-line Marker to a position closer to the end of the animation.
13. Click the **Add a Key Frame** button.
14. Click the new Key Frame and set the **Scaling** value to be larger than the final size.
15. Save the scene.

Note: Always scrub the time-line in the time-line editor to check for potential conflicts (caused by rotation of objects).

17.8.10 To Animate Single Objects

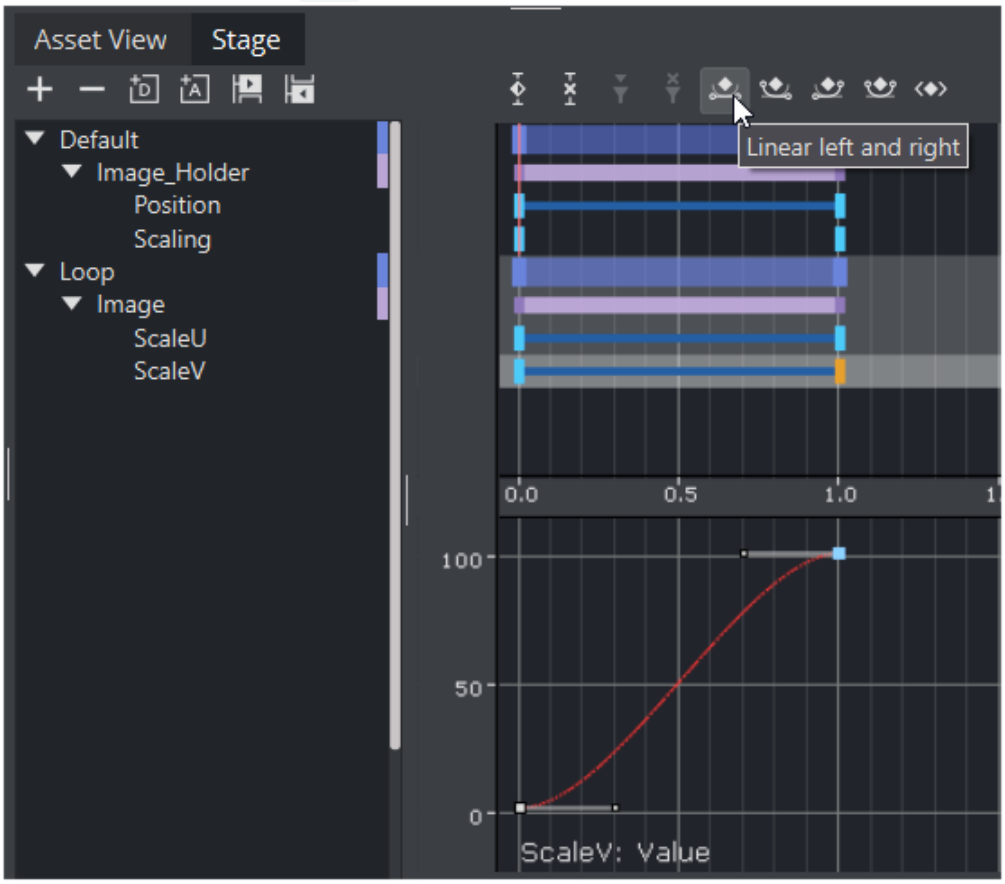
This procedure focuses on the animation of single objects. Animating single objects increases the dynamics of the scene, especially when the scene has reached its final position on-screen and all other animations have stopped.

This procedure describes how to make a flare move, continuously, along another object.

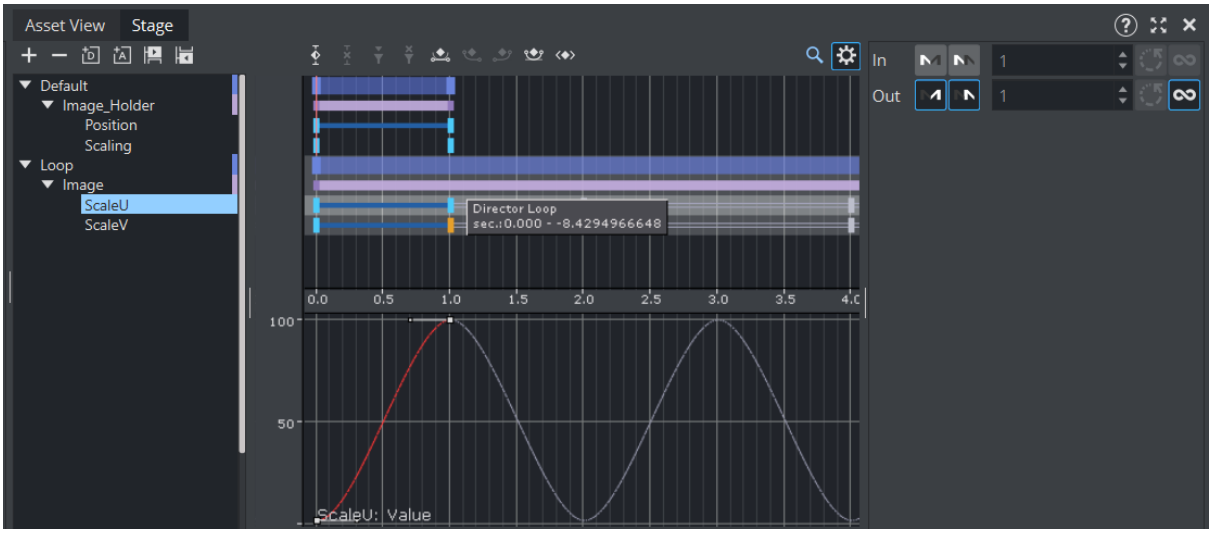
1. Open the OTS scene.
2. Open the Stage Editor.
3. Create a new Director and name it Loop:
   - Click the **+D** button, or
   - Right-click the Stage Tree and select **Add Director**.
4. Select the Loop Director.
5. Set the Time-line Editor to `0` fields.
6. Open the Texture Editor of the Image:

Copyright © 2026 Vizrt

- Set **Scale X** and **Scale Y** to `1` and click the **Set Key Frame** button.
- Set **Scale X** and **Scale Y** to `1.2` and click the **Set Key Frame** button.



7. Select both of the **ScalingX** and **ScalingY** Key Frames and click the **Linear left and right** button.
8. Click the **ScalingX** and **ScalingY** channel to open its Channel Editor. Click the **Loop out** and **Infinite** buttons.



9. The time-line should now look similar to the image above. **Save** the scene.

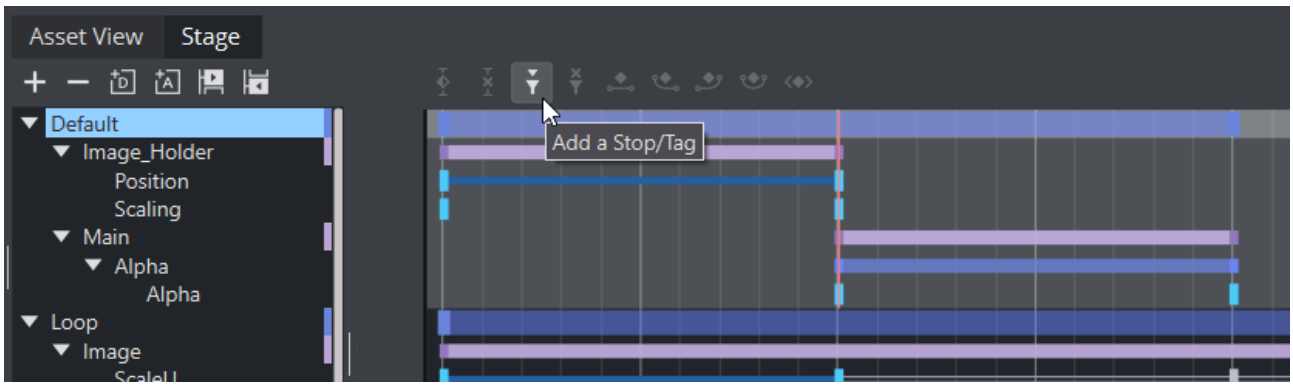
17.8.11 To Animate out

Animations to the scene so far has brought the graphic on-screen for the viewer to see. However, the scene should also be able to animate out instead of simply removing it (hard cut). This can be achieved in three ways:

> **Note:** With one or a combination of the following three methods an out animation can be created. The scene now works in terms of animation, however, it does not work well if someone needs to control, for example, when or how the out animation should start. For this we need to add controls (using control plug-ins) that allows an operator to control the scene On Air.

17.8.12 To Fade out the Graphic with the Alpha Plug-in

1. Add the Alpha plug-in to the Main container.
2. Set the time-line to the end of the in animation (at 60 fields for example). Be sure to use the Default director
3. Open the Alpha plug-in editor.
4. Set the **Alpha** value to `100%`.
5. Click the **Set Key Frame** button.
6. Set the **Alpha** value to `0%`.
7. Click the **Set Key Frame** button.
8. Now we need to add a stop event. This ensures one can take the graphics in, then it is paused until the operator continues the animation which then plays the out animation.
9. Place the timeline cursor to time 1.0 (where all is animated in) and add a stop event.



When played, the animation animates in, stops at position 1.0 and after clicking on the continue button fades out to Alpha at 0%.

> **Note:** The placed *pilot1* stop/tag point on the Default Director's time-line, at a given field, enables Vizrt control applications to create a thumbnail and show it in, for instance, the playlist.

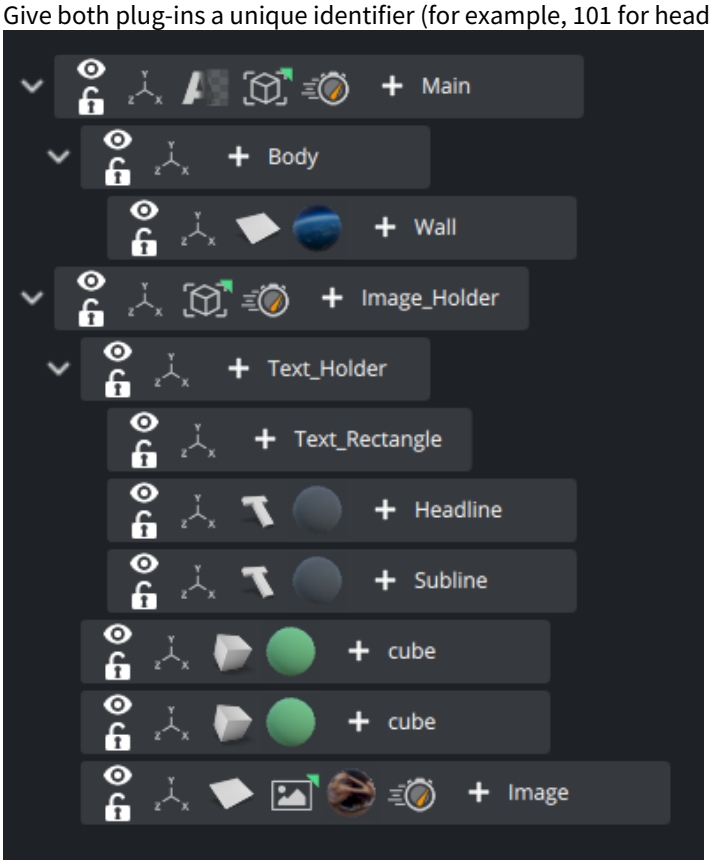
17.8.13 To Add Controls

Stop/tag points are added to the Director to allow the operator to control the animation. However the operator, when required, should be able to control the content of the Scene. For this purpose Control plug-ins are used.

Copyright © 2026 Vizrt

In our example we would like to control the image and the Head and the Subline text. Therefore, we need to add control plug-ins to the respective containers.

1. Open the Plugins Panel and search for Control Image.
2. Drag it to the Image container.
3. Click on the plug-in and give it a unique identifier (for example, 100).
4. Search for Control Text.
5. Drag the plug-in to Headline container.
6. Drag the plug-in to Subline container.
7. Give both plug-ins a unique identifier (for example, 101 for headline, 102 for Subline).

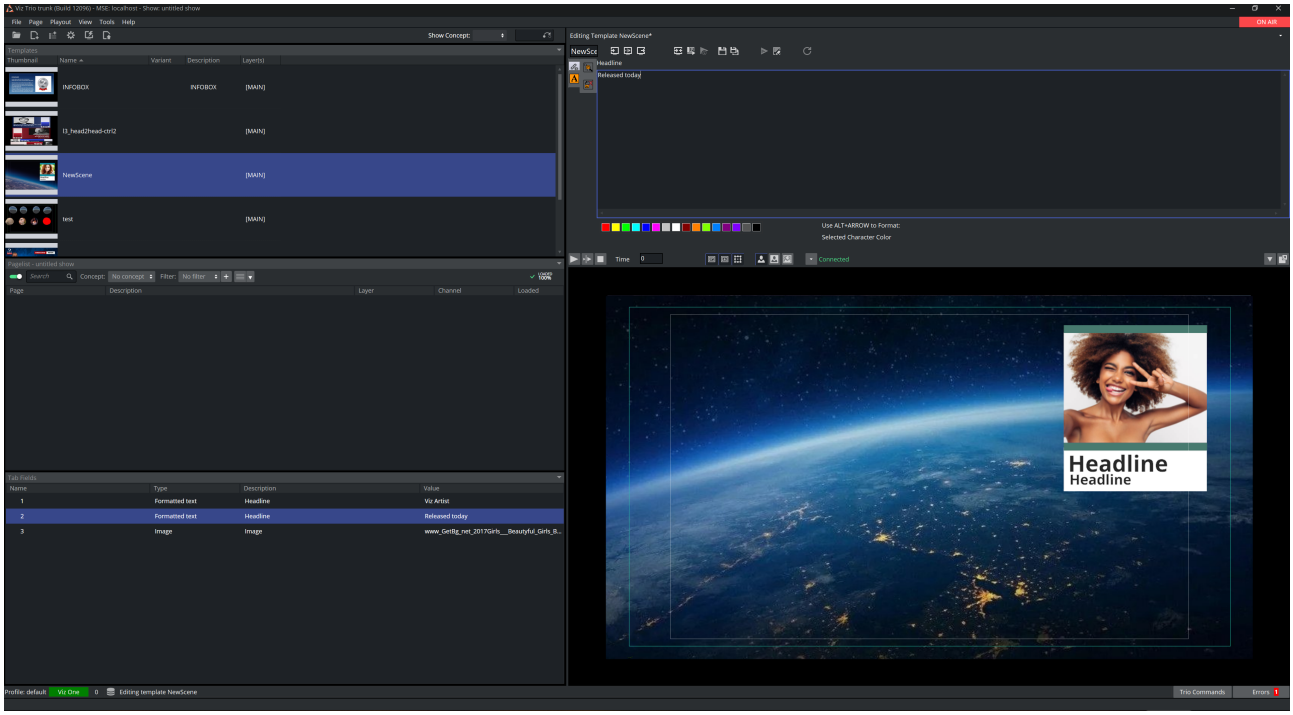


If this scene is now imported in, for example, Viz Trio, an operator is able to fill in the control values:

Tip: Number the Field Identifiers to help order them in the control application interface.

Note: To check that an operator can use the scene test it with a control application (for example, Viz Pilot with the Template Wizard, Viz Trio, Viz Channel Branding).

Copyright © 2026 Vizrt

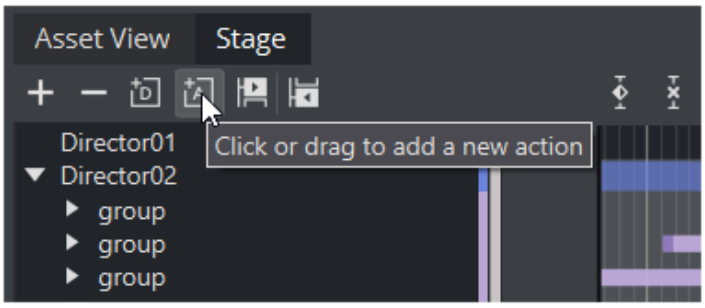


Copyright © 2026 Vizrt

17.9 Action Channels

Actions Channels are control items which, when an Action Key Frame is added, do special tasks when the time line passes them. Regular Channels are typically basic actions that can be created through the use of the Viz Artist GUI. For example, the rotation of a cube can be defined in the Transformation Editor of the cube container.

For more special scenarios, such as audio animations, it may not be possible to create the animations based on the standard options. In such cases it is necessary to create actions with custom scripts that suit the situation.



Most procedures on Action channels are done in the Stage Tree. Procedures which affect the time-line are done directly in the Dopesheet Editor. Working with Action channels in the Dopesheet Editor enables the Action Channel Editor.

17.9.1 Action Channels in the Stage Tree

To Create an Action Channel

The created Action Channel also requires an Action Key Frame.

1. Select a Director.
2. Click (Add New Action) in the **Stage Tree** Bar, or drag the icon in the Stage Tree Bar to a Director.

To Move an Action Channel

Drag an action channel from a Director to the target Director.

To Copy an Action Channel

Press **CTRL** while dragging an action channel from a Director to the target Director.

To Rename an Action Channel

1. Right-click an Action channel and select **Rename** or press **F2**.
2. In the text box that shows, type a new descriptive name.
3. Press **ENTER**.

Copyright © 2026 Vizrt

Note: Special characters, such as space, are not allowed in action channel names.

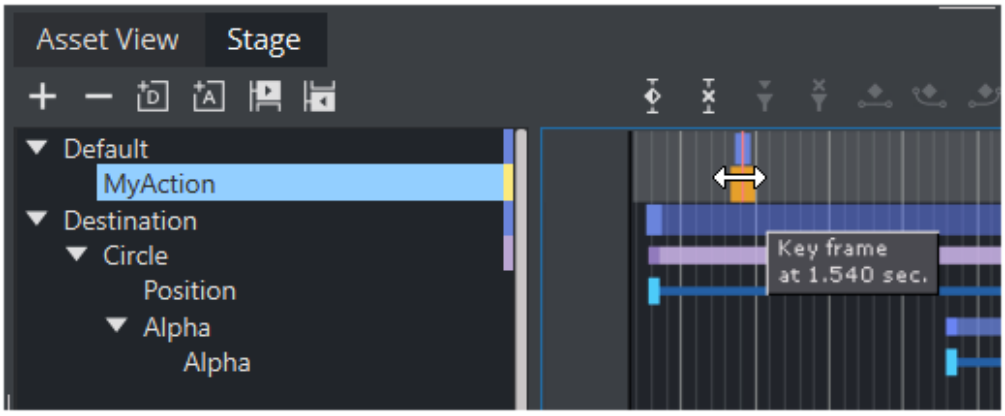
To Delete an Action Channel

- Right-click an action channel in the Stage Tree and select **Delete Action Channel**.
  or
- Drag the action channel to the trash can.

17.9.2 Action Channels in the Dopesheet Editor

To Move an Action Channel in the Time-line

Drag the action to a different time.



To Speed Up or Slow Down an Action Channel

- Click on the start or the end Key Frame of the Section Channel.
- Drag the start or end Key Frame to change the Action Channel speed:

17.9.3 To Use Action Channels

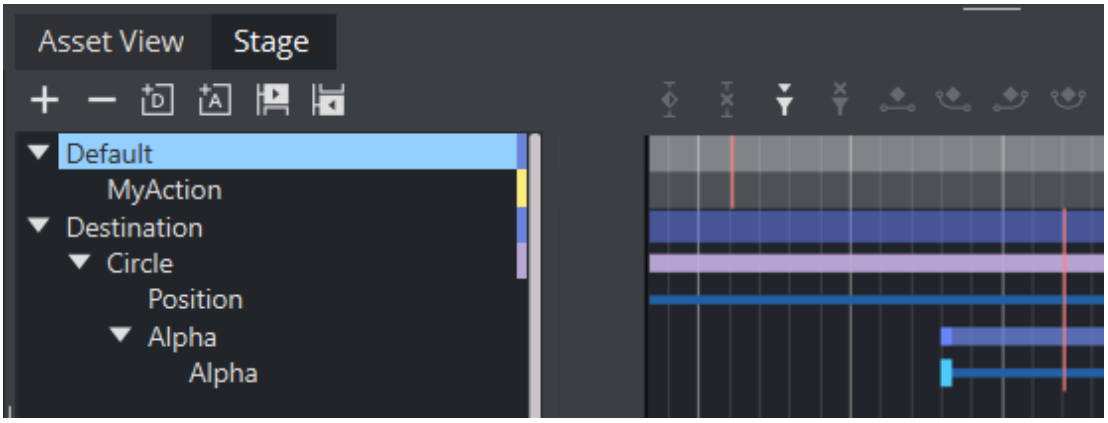
Action Channels can be used to trigger Actions as soon as the timeline hits this event.

Example:

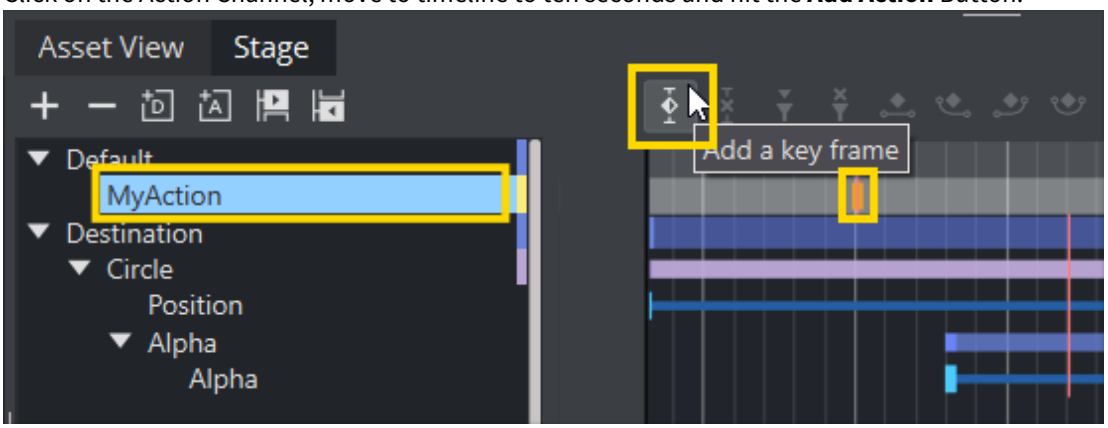
At the end of our animation, we want to start another director. This should be restricted to work in forward direction only.

- Step 1: Create an action channel.
  Drag the to the Channel Editor and give it a proper name (for example, *MyAction)*.

Copyright © 2026 Vizrt



- Step 2: Create a Action Keyframe at position 40.
  Click on the Action Channel, move to timeline to ten seconds and hit the **Add Action** Button.

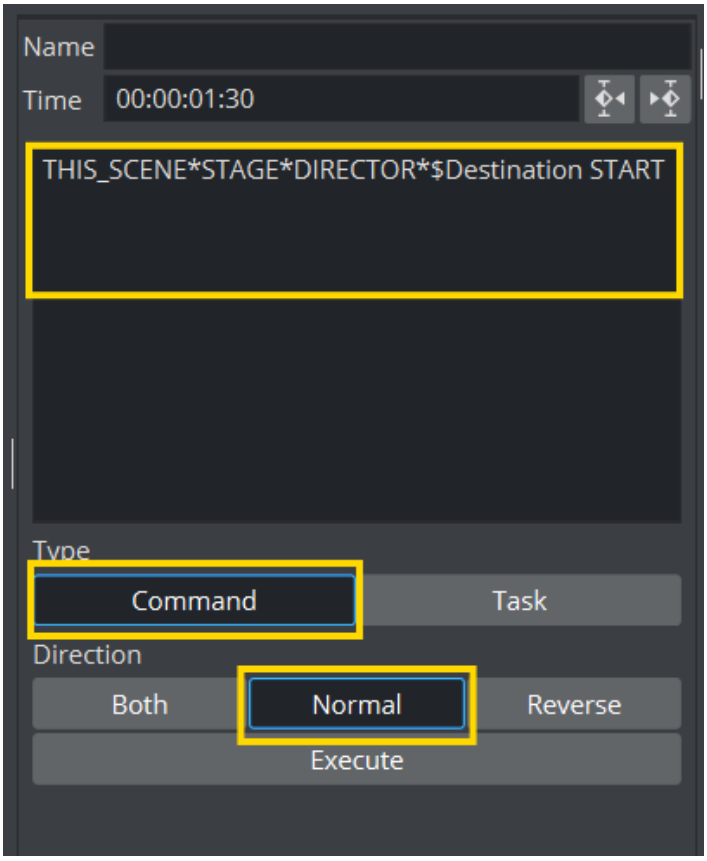


- Step 3: Define your action.
  Select the new Keyframe, switch tot he Property Editor and add the commands to be executed. This commands are Viz commands being sent to the local Engine. In our case, we want to start director *Destination*. The command is `THIS_SCENE*STAGE*DIRECTOR*$Destination START`.

Be sure to set the type to *Command*, this ensures that it is executed as viz command. *Task* means an operation command is issued, for example *Notepad.exe* launches notepad as soon as our animation hits the keyframe.

- Step 4: Define the direction.
  Select the keyframe and set the correct direction, in our case *Normal*.

Copyright © 2026 Vizrt



Start the animation *Default*. As soon as it reaches the ActionKeyframe at Position 10, it executes it. Based on our command, it starts the Director *Destination*.

Copyright © 2026 Vizrt

17.10 Create an Advanced Animation

Once you have mastered Create a Basic Animation, where the position of a cube is moved, you can move on to more advanced tasks. This section provides some additional procedures, so that a cube can continue its journey rightward, rotate while moving, and gain a nicer appearance by adding color to it.

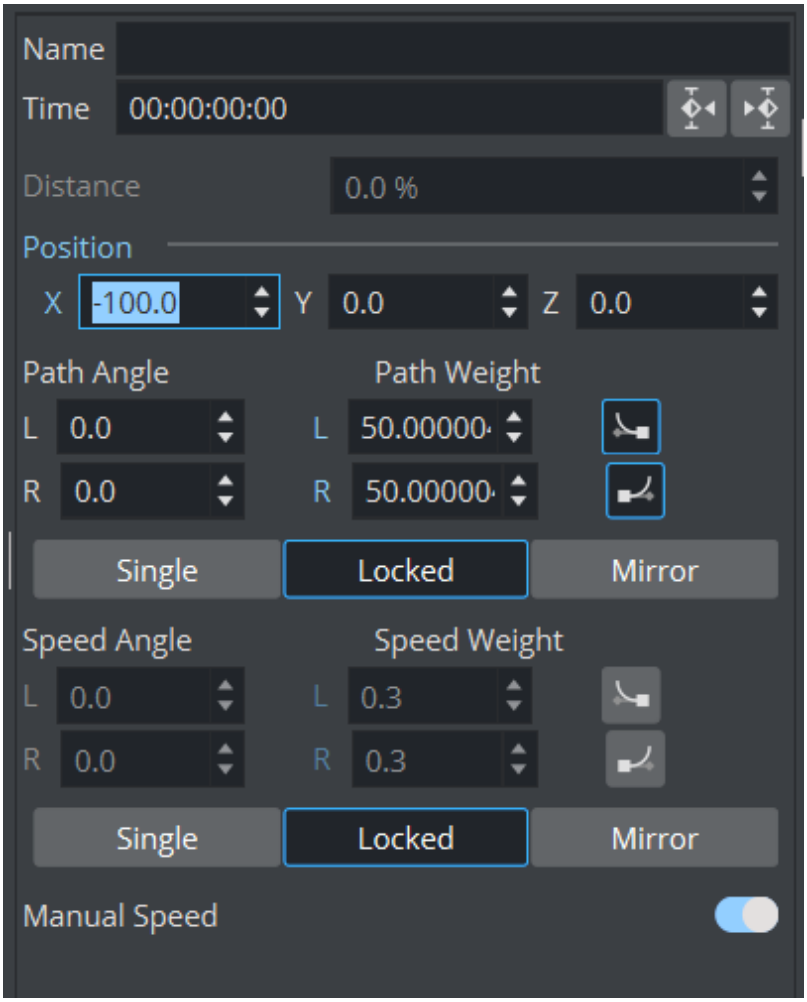
This section contains the following procedures:

- To Edit the First Key Frame
- To Edit the Second Key Frame
- To Create an Additional Key Frame
- To Add a Stop Point
- To Create Key Frames for Rotation
- To Modify the Rotation Settings
- To Run the Animations Independently
- To Add and Animate a Material

17.10.1 To Edit the First Key Frame

1. Enable the Key Frame Editors.
2. Select the Key Frame at the far left in the Dopesheet Editor.

Copyright © 2026 Vizrt



3. In the Key Frame Editors, set the **Position** to X = `-100`.

17.10.2 To Edit the Second Key Frame

1. Select the Key Frame at the right in the Dopesheet Editor.
2. In the Key Frame Editors, set the **Position** to X = `0`.

> **Tip:** Set the position either by typing in the input box, using the up/down buttons, or clicking the **R** (reset/default setting) button.

3. When playing the animation now, the cube still slides rightward, but from a point at the left part of the Scene Editor to the center.
4. Click the **Play** button in the Time-line Editor.

17.10.3 To Create an Additional Key Frame

1. In the Time-line Editor, set the time-line field to `100`.
2. In the Scene Editor, drag the cube to the right, so that it shifts to a different position.

Copyright © 2026 Vizrt

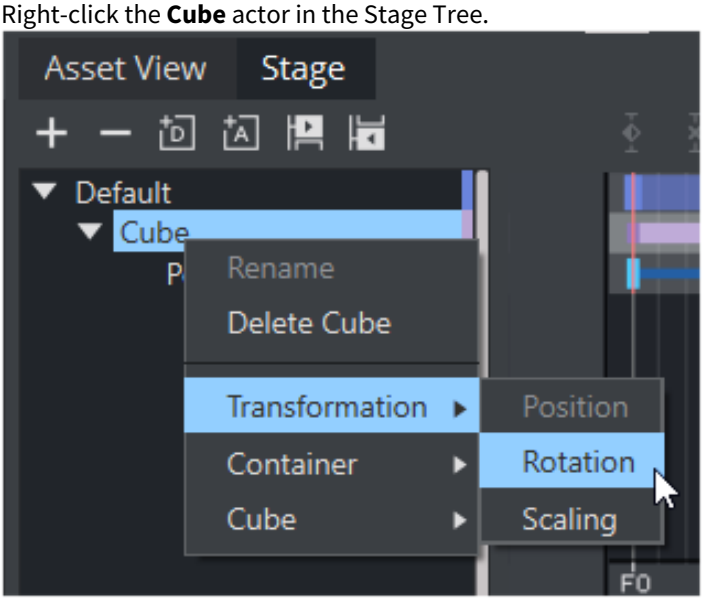
3. Set an additional Key Frame.
4. Click the **Play** button in the Time-line Editor. When playing the animation now, the cube slides rightward from a point at the left part of the Scene Editor, through the center, and all the way to the right part.

17.10.4 To Add a Stop Point

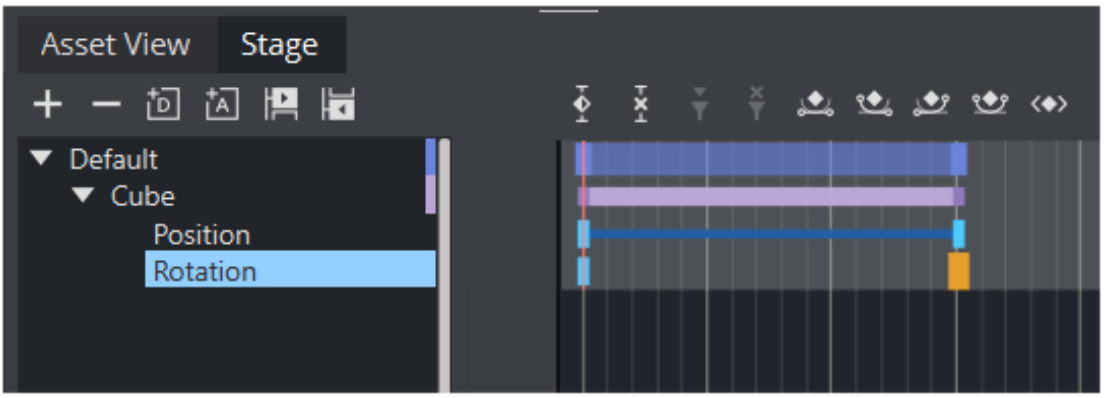
1. Select the required Director.
2. In the Time-line Editor, set the time-line field to `50`.
3. Create a stop event.
4. Click the **Play** button in the Time-line Editor. When playing the animation now, the cube slides rightward until it reaches the center of the Scene Editor, and then come to a halt.

17.10.5 To Create Key Frames for Rotation

1. Right-click the **Cube** actor in the Stage Tree.



2. In the menu that shows, click **Transformation > Rotation**. A rotation channel and two Key Frames are added.



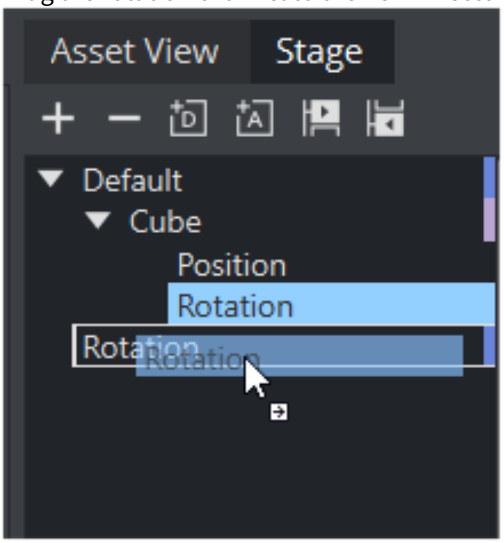
Copyright © 2026 Vizrt

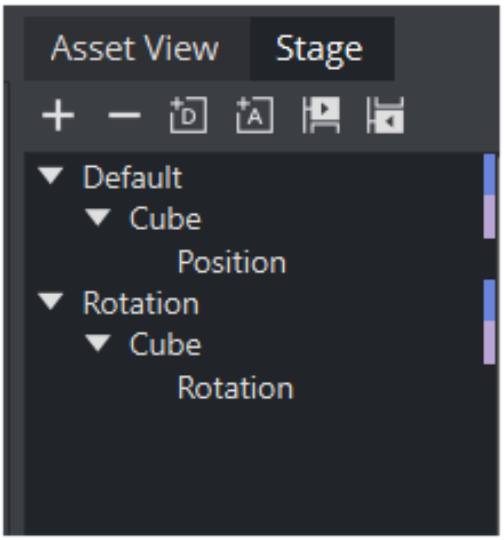
17.10.6 To Modify the Rotation Settings

1. Select the Key Frame at the right in the Dopesheet Editor.
2. In the Key Frame Editors, set the **Rotation** to X = `360` .
3. Click the **Play** button in the Time-line Editor. When playing the animation now, the cube slides rightward until it reaches the center of the Scene Editor, while rotating all the while before coming to a halt.

17.10.7 To Run the Animations Independently

1. Create a Director at the root level of the Stage Tree.
2. Rename the new Director to **Rotation**.
3. Drag the rotation channel to the new Director. The Stage Tree and should now look like this:



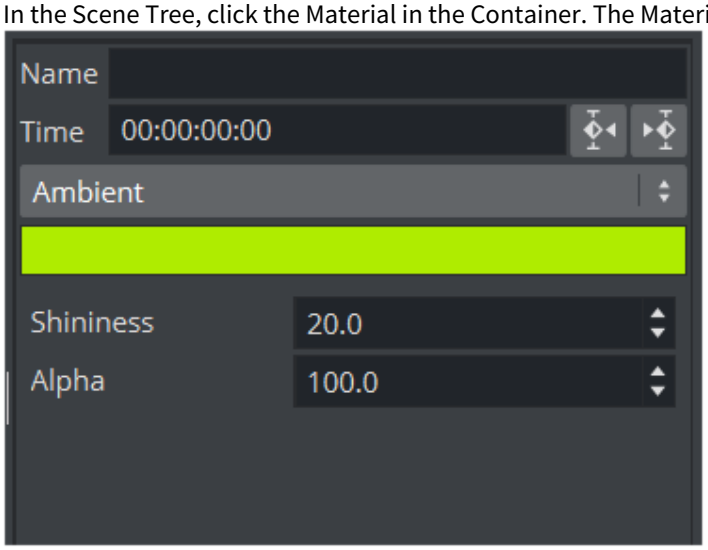


When playing the animations now, the output depends on which Director is selected. If the **Default** Director is selected, only the position animation is played. If the **Rotation** Director is selected, only the rotation animation is played. Selecting both/all Directors plays both the position and rotation animations.

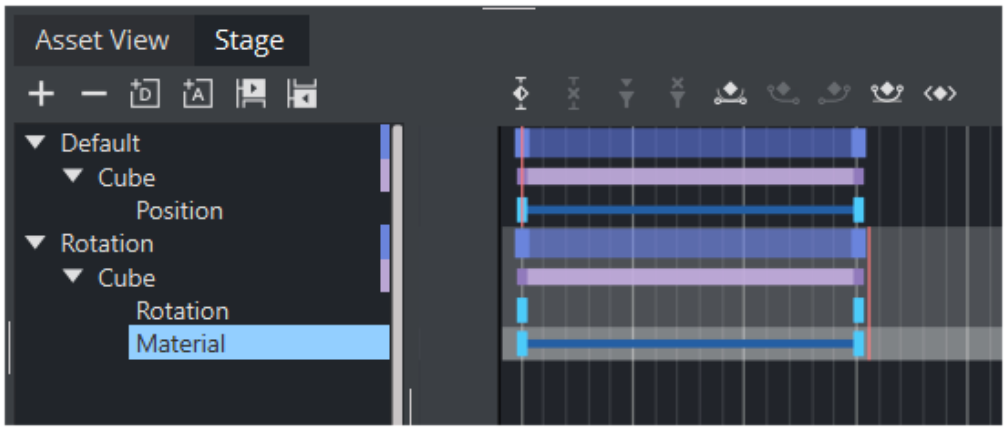
Copyright © 2026 Vizrt

17.10.8 To Add and Animate a Material

1. Right Click on the Container and choose **Add material...**
2. In the Scene Tree, click the Material in the Container. The Material Editor is shown.



3. In the Time-line Editor, set the time when you want the Key Frame to be created.
4. In the Time-line Editor, click the **Set Key** button.
5. Change the material's color.
6. To create the next Key Frame, in the Time-line Editor, click the **Set Key** button.
7. Click the **Play** button in the Time-line Editor. The cube now has a color that changes as part of the animation.



Copyright © 2026 Vizrt

17.11 Advanced Animation Functions

This section details the creation of a Scene with the more advanced animation features in Viz Artist. As the basis, take the Scene created in Create a Basic Animation and improved on in Create an Advanced Animation.

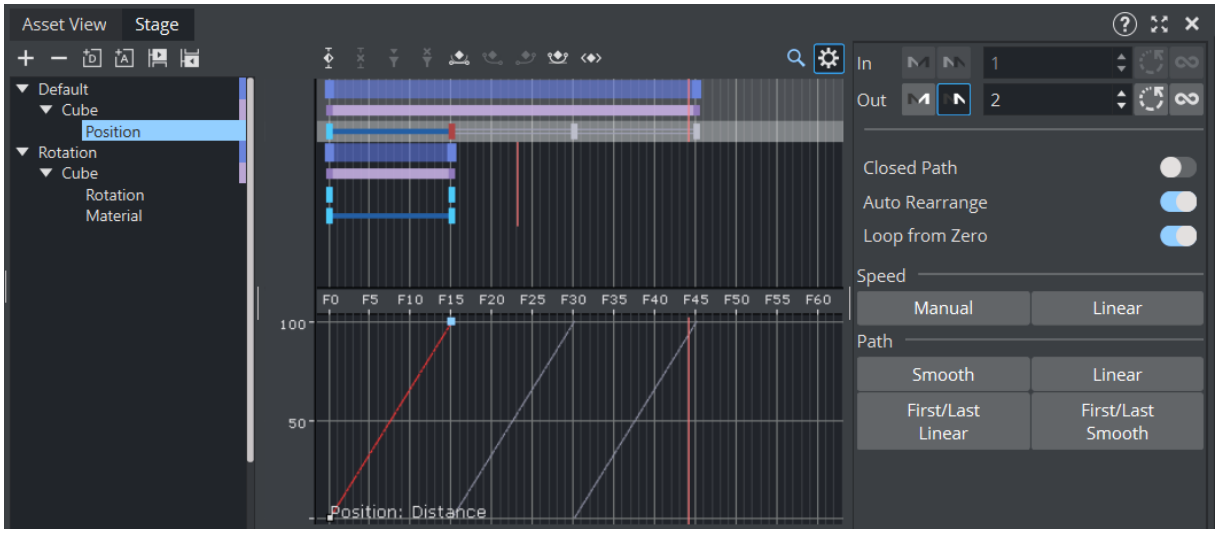
This section contains the following topics:

- Animating Channels
- Path Control
- Spline Control
- Animate Audio
- Animate Cameras
- Animate Light
- Merging Animations
- Multi Select Animation

17.11.1 Animating Channels

Repeat Animation Channels (Loop)

The loop operation makes it possible to repeat the animation of a channel either at the beginning or end of the channel time-line.

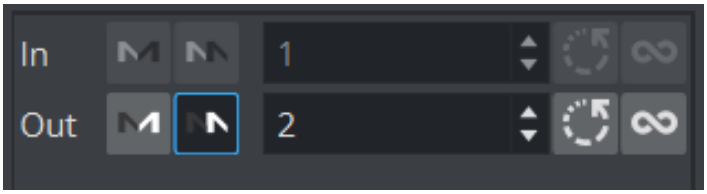


To Repeat the Animation of a Channel

1. Select a channel.
2. In the Channel Editor, enable the **Loop In/Out** buttons.

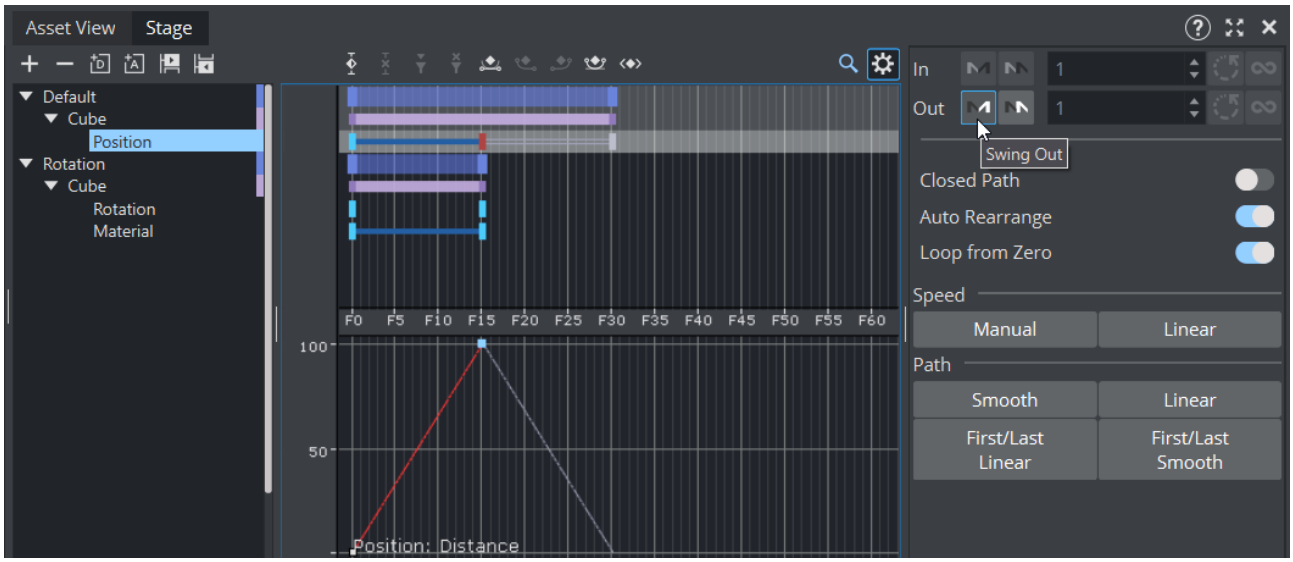
Note: Loop at the beginning is only available if the belonging actor has a defined offset value.

3. Define how many times the loop should run before coming to a halt. Either enter a number in the field or click the **Infinite** button. When enabling loop, the looped animation shows as gray in the in the Dopesheet Editor.



Repeat Animation Channels in Reverse (Swing)

The swing operation makes it possible to repeat the animation of a channel in reverse play either at the beginning or end of the channel time-line.



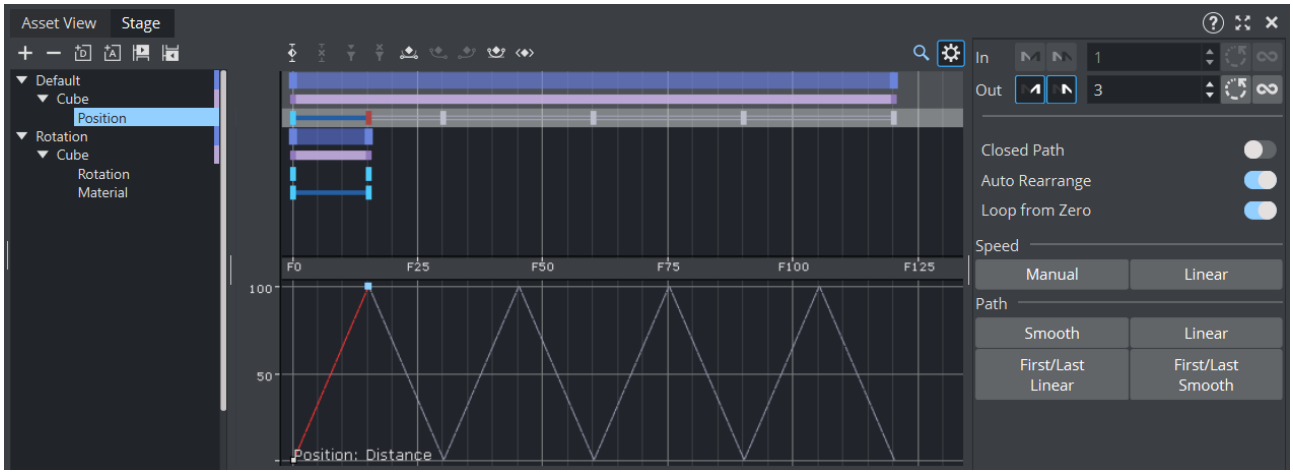
To Create a Swinging Animation

1. Select a channel
2. In the Channel Editor, enable the **Swing In/Out** buttons.
   When enabling swing, the animation shows as gray in the in the Dopesheet Editor.

Combine Swing and Loop

It is possible to combine the swing and loop operations on a channel. The animation is first added in reverse play (swing) and then started once again from the beginning to the end (loop). This saves a great amount of time if in need of a recurring animation, for example a bouncing ball. Animations with swings and loops show as gray in the in the Dopesheet Editor.

Copyright © 2026 Vizrt



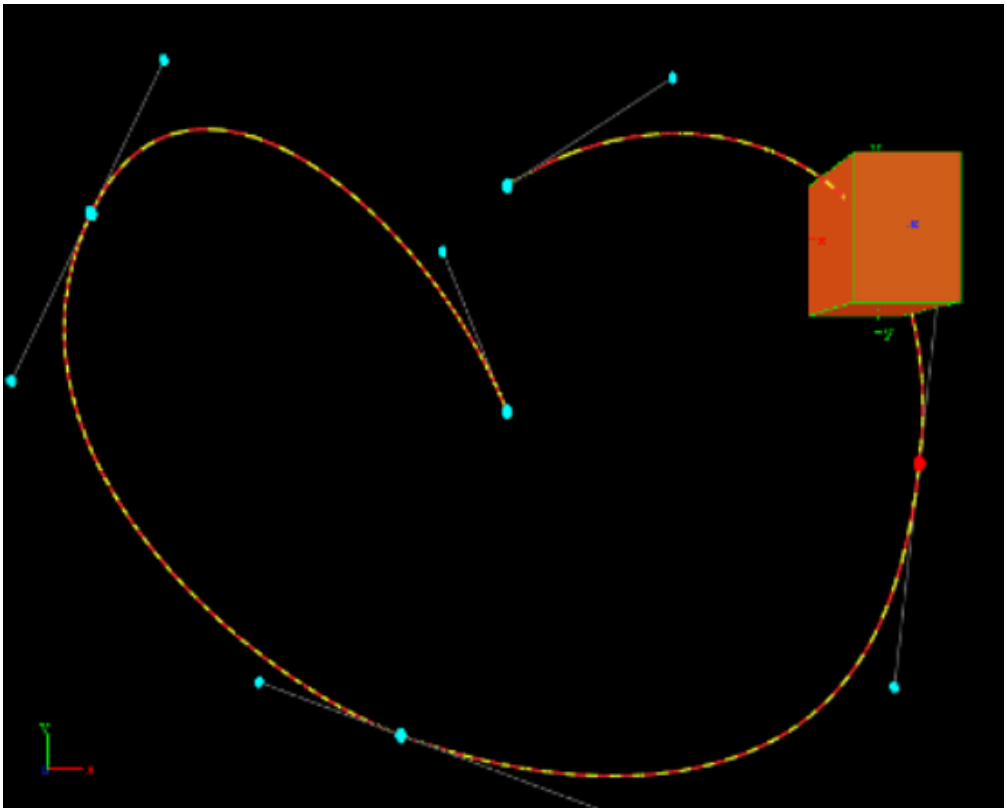
17.11.2 Path Control

The position animation follows a path you define by the Key Frames. A spline is interpolated from the Key Frames. You can view and edit this spline in the Scene Editor.

To Edit the Spline in the Scene Editor

1. Enable the bounding box:
   - Click the **BB** button to the left of the Scene Editor, or
   - With the mouse pointer over the Scene Editor, press **B**.
2. In the scene tree, select the animated container.
3. Right-click in the Scene Editor and select **Animation**. The path is shown as a spline with the Key Frames marked on it.

Copyright © 2026 Vizrt



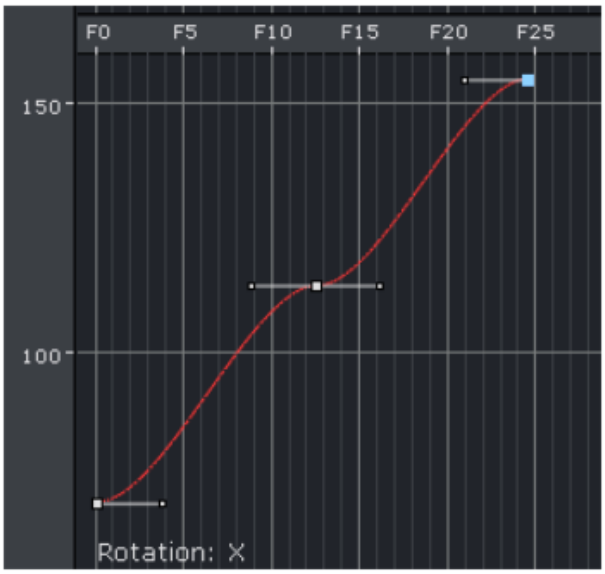
4. Drag the Key Frames around to modify the animation trajectory. The red and yellow markings on the spline provide information about the speed the object is moving along its trajectory. One segment represents a field, the longer the segments are, the faster the object travels.

As every Key Frame has handles (seen in blue) you can use them to modify the Key Frame's trajectory. To do so select a handle and drag it around. By default the handles are set to be locked and proportional. If you want to switch to a mirrored mode, press **SHIFT** while you drag the handle around. Pressing **CTRL** allows you to edit the single handle only. An other option to set the handles is the object editor. Select the Key Frame you want to change the handles in and set either **Single**, **Locked** or **Mirrored** and change the values in **Weight L** and **R**.

17.11.3 Spline Control

In the spline editor the distance spline is shown, which sets the current position of the container along the path. The y-axis in the spline editor is scaled to use the percentage of the trajectory, while the x-axis still uses the time. In the spline editor you can only change the position of a Key Frame along the time-line, but not its percentage of the total trajectory.

Copyright © 2026 Vizrt



By default the handles at every Key Frame on the spline are set to be linear.

To Edit the Spline

- Drag the handles around to change their positions. By default the handles are set to move proportionally. So as you move the handle of a Key Frame the other one moves too.
- Press **CTRL** while you drag a handle to modify only the selected one. This “single handle mode” is enabled for the Key Frame as long as you do not reset it by a click the Key Frame icons in the Stage Editor Bar.

Example: If you, for example, modify the spline in such a way that the value decreases, the container moves backwards.

To Move Key Frames in the Spline

Select the required Key Frame and drag it. Multiple Key Frames can be selected to move at once.

To Insert Key Frames from the Spline

1. Move the cursor on the spline exactly where you would like to add the Key Frame.
2. Click on the required location to set the Key Frame.

To Delete a Key Frame from the Spline

1. Select the required Key Frame.
2. Press **DELETE** or click **Delete Key Frame** button in the Stage Editor Bar. Multiple Key Frames can be selected to be deleted at once.

Copyright © 2026 Vizrt

17.11.4 Animate Audio

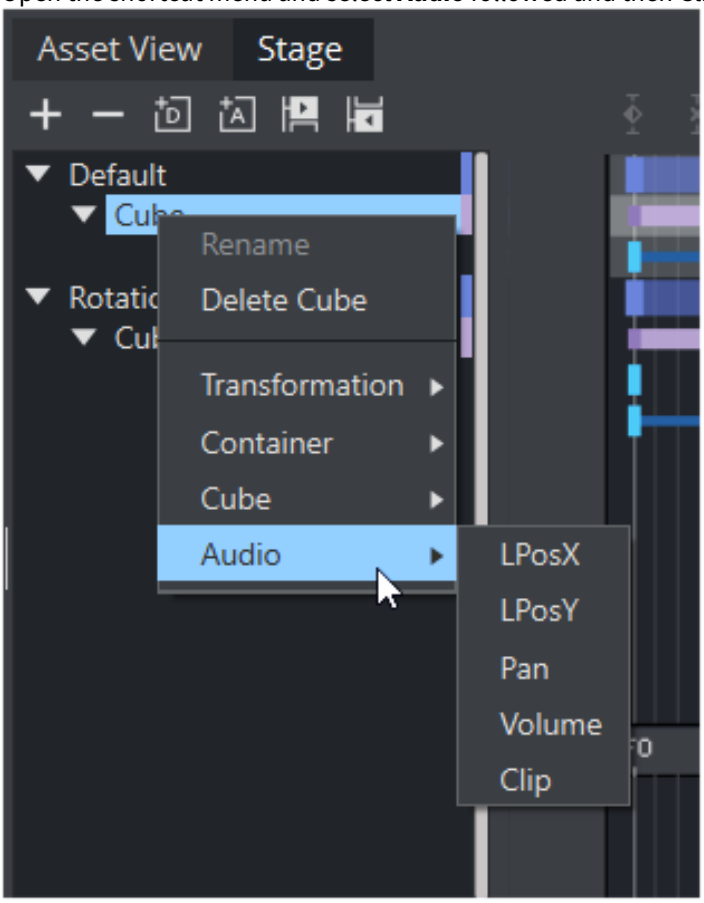
An audio animation plays an audio clip in an animation.

To Enable Audio in an Animation

1. Apply the audio container function to a container in the Scene.
2. Add an actor for this container to the Stage Tree, if it is not present yet.

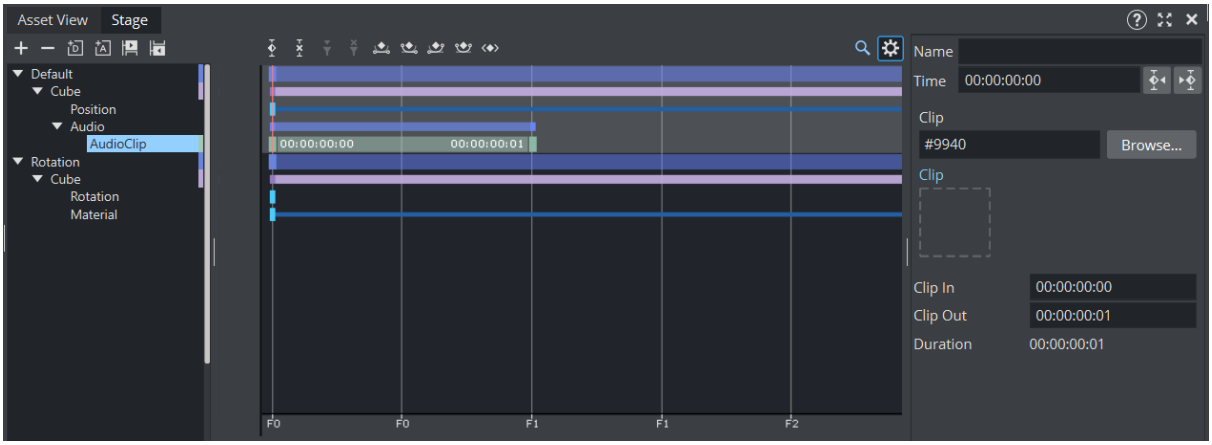
> **Tip:** The easy way to do is to drag the container from the scene tree onto the Stage Tree.

3. Open the shortcut menu and select **Audio** followed and then **Clip**.



4. Select a Key Frame in this channel to open the object editor.
5. Switch to the **Server/Stage** view and drag an audio clip from the Server Panel onto the **Clip** drop zone in the object editor.
6. In the object editor for the Key Frames of the AudioClip channel, configure the following parameters, as necessary.

Copyright © 2026 Vizrt



- **Name:** Sets a name for the Key Frame. This name is for use only for external commands or scripts.
- **Time:** Shows the time value in fields the Key Frame is placed. Editing this value changes to the position of the Key Frame.
- **Clip:** Drag an audio clip from the Server Panel onto this drop zone.
- **Stop:** Stops playback.
- **Play:** Plays the clip.
- **Delete:** Removes the clip.
- **Begin Time:** Sets the begin time within the clip. Everything before this time is not played in the animation. Another way to achieve this is to drag the Key Frame at the beginning of the clip.
- **End Time:** Sets the end time within the clip. Everything after this time is not played in the animation. Another way to achieve this is to drag the Key Frame at the end of the clip.
- **Duration:** Shows the duration to be played.

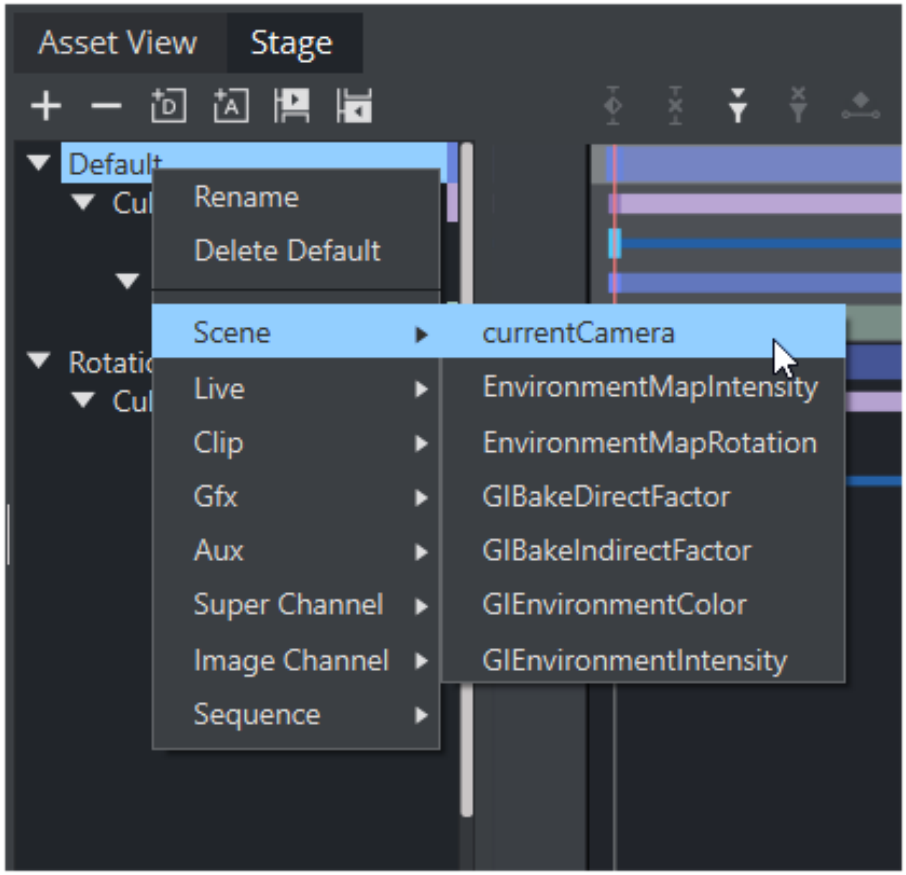
17.11.5 Animate Cameras

You can animate cameras by adding an actor for a camera to the Stage Tree is by setting Key Frames in the Scene Editor.

To Animate a Camera

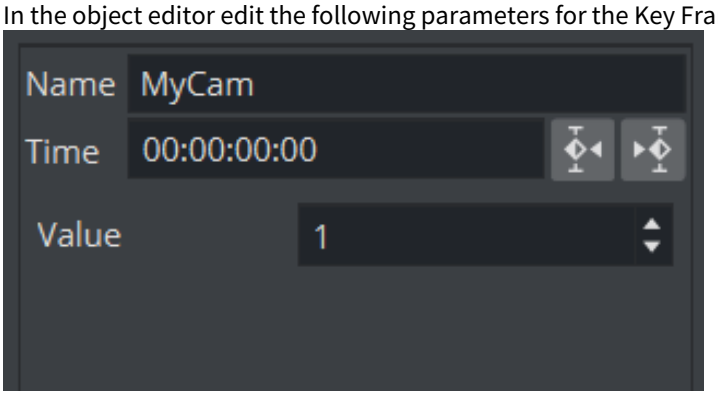
1. Set the Time-line Marker to when the Key Frame should be added to start with the animation of the camera.
2. Click the **Add Key Frame** button in the Stage Editor Bar.
3. Change one of the selected camera properties and set the next Key Frame. This adds an actor in the Stage Tree for the camera whose properties you have changed. Another way to create camera animations is to enable position and/or direction tracking in the camera editor to animate the camera through a container transformation animation.

Copyright © 2026 Vizrt



To Animate the Current Camera

1. Set a Key Frame in the current Camera channel.
2. In the object editor edit the following parameters for the Key Frame as necessary.



- **Name:** Sets a name for the Key Frame. This name is for use only for external commands or scripts.
- **Time:** Shows the time value in fields the Key Frame is placed. Editing this value changes to the position of the Key Frame. If you click arrow buttons in the Stage Editor Bar the previous or next Key Frame is selected.

Copyright © 2026 Vizrt

- **Value:** Sets the camera you want to switch to.
  The animation of the current Camera allows you to switch the camera through which to view at a given time.

To Create a Zoom Animation

If you click Spline button Stage Editor Bar or press **SHIFT + Q**, the whole spline which is currently shown is zoomed to. If you have selected an actor or a Director, the zoom is adjusted to fit this entry. When zooming the whole spline, both the time-axis for animation editor and the spline editor is zoomed, as well as the value-axis in the spline editor.

Another way to zoom the time-axis is by pressing **X** to zoom in and **Y** or **Z** to zoom out. As an alternative you can use the mouse wheel. If you press **SHIFT + Y**, you set the zoom along the time-axis to zoom the whole spline while **SHIFT + X** sets the zoom along the time-axis to show the whole animation.

In the spline editor you can zoom the value axis too. This is done by pressing **A** to zoom in or **S** to zoom out. If you press **SHIFT + *A*** you zoom the whole spline along the value axis. You can zoom also if you press one of the “zoom keys” and move the cursor around with the middle button pressed. If you have changed the zoom level, you can switch back to standard zoom at any time by a click designated button in the Stage Editor Bar or if you press **SHIFT + W**. As you maybe zoomed the view in such away you are no longer able to edit the spline comfortably, you can pan by moving around the cursor while the middle button is pressed. Zooming along the time axis zooms both the animation editor and the spline editor.

17.11.6 Animate Light

The only way to add an actor for a light to the Stage Tree is by setting Key Frames in the Scene Editor. To do so set the time-line value to the time you want to start with the animation of the light and set the first Key Frame. Now change one of the selected lights properties and set the next Key Frame. This adds an actor for the light which property you have changed to the Stage Tree.

An easier way the animate a light is to enable position and/or direction tracking in the light editor. To learn more about the light editor, see Light Editor.

17.11.7 Merging Animations

In Viz Artist, it is possible to merge container groups into one single compound object. If an animation has been defined for at least one of the containers, on merging a new Director is created, given the name of the merged object in the scene tree. There is a Sub-Director by the name of the original Director the animations were created in. The animations created for the single containers are kept, but cannot be changed anymore unless you split the object into the containers it has been created from.

The animations are kept also if you save the newly created object in the database. Please be aware that you cannot add Sub-Directors to the Directors created automatically by merging.

17.11.8 Multi Select Animation

If two or more objects are selected multiple animations can be created in one step. This adds independent animations channels for each container within the selection.

1. Select the objects to be animated.
2. Create the animation, as required (see Create a Basic Animation).

Copyright © 2026 Vizrt

An Actor is created for each selected animated object. Once the animation has been created each Actor can be edited separately.

Copyright © 2026 Vizrt

18 Import, Post and Archive

This section details how to import files and archives, and how to archive (export) graphical resources such as one or several Scenes including fonts, images and so on to a file (archive) on disk. Scenes can be exported as video or image files. This process is called Post Rendering.

Import of Scenes from third party file formats is also supported.

This section contains information on the following topics:

- Post Render Scenes
- Archives
- Import of Files and Archives
- Deploy Items

Copyright © 2026 Vizrt

18.1 Post Render Scenes

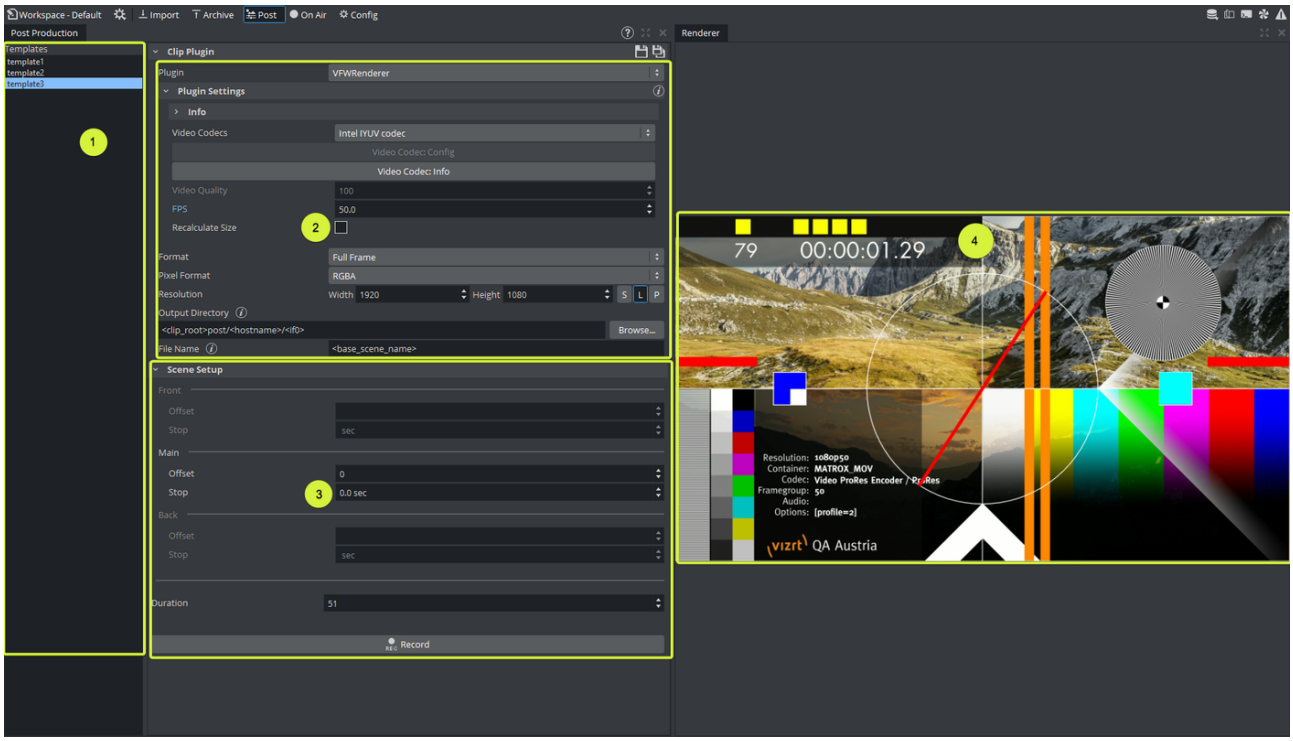
Scenes can be exported as video or image files. This process is called post rendering.

> **Note:** Post rendering is not possible with clips or live video feeds. However, these items can be post rendered if the clips are place within the stage.

This section contains information on the following topics:

- Post Render Screen
- To Post Render a Scene
- Post Render Templates
  - To Save a Post Render Template
  - To Change a Post Render Template
  - To Load a Post Render Template
  - To Rename a Post Render Template
  - To Delete a Post Render Template
- Post Rendering Plug-ins
  - VfW Plug-in
  - VideoRenderer Plug-in
  - Matrox File Writer

18.1.1 Post Render Screen



The post render panel consists of:

- Templates (**1**) (see Post Render Templates).
- Output Settings (**2**) (see Post Rendering Plug-ins): Used to configure the output format and the destination of your post project
- Post rendering setup (**3**): The configuration for your scenes to be rendered (In the Classic Render Pipeline you can render Front, Main and Backlayer) and set various offsets and stop points. Offsets can be used for the Main, Front and Back Layer, so they start at different times.
  Stop points in post are usually lost, therefore the Stop points can be changed to a pause duration.

> **Note:** Layers need to be set before switching to the post renderer.

- Output preview (**4**)

18.1.2 To Post Render a Scene

- Open the Scene to be post rendered.
- Assign (if needed) **Front** and **Backlayer**.
- Click **Post**. The Scene is loaded **Main Layer** target area.
- Select a Template to load all settings or choose your output module. Output modules are integrated as plug-ins. These can be one of the following:

Format	Supports Alpha	Description
JPG	(1)	JPEG uses a lossy form of compression and therefore is not well suited for line drawings and other textual or iconic graphics. Sharp contrast results in noticeable artifacts. The JPEG plug-in allows to set a quality level. It is recommended to use a value higher than 90.
PNG		PNG is a raster-graphics file-format that supports lossless data compression. PNG images also support transparency. The PNG plug-in allows to set a quality level.
TGA		TGA files are based on a lossless RLE compression . This type of compression works acceptably well for simpler images, such as icons, cartoons and line drawings, but performs poorly on photographs. TGA files support transparency.
TIFF		TIFF files are widely used and use a lossless compression. TIFF files support transparency.

Copyright © 2026 Vizrt

Format	Supports Alpha	Description
VBN	✓	Vizrt Binary are already compressed mipmapped images. It is the same format as already used when storing images on Graphic Hub. When used as an image sequence for OMO or similar plug-in, it is recommended to load them directly from Graphic Hub and not from the file system.

(1) If native Alpha format is not supported, a separate file can be generated (RGB AAA).

Clip Format	Comments
VFWRenderer	Based on *Microsoft Video For Windows* technology. Requires codecs in 64bit to be installed on the Windows System.
Video Renderer	The plugin is based on Microsoft's DirectShow filters. Therefore it also requires 64bit codecs to be installed on the Windows System.
Matrox File Writer	Based on the Matrox SDK. Requires installation of Matrox Topology driver. Capabilities depend on the Matrox board and/or license. Please see Chapter Matrox File Writer for details.

Information: Additional settings like options, information, quality, FPS are exposed by the installed codecs.

- Select a frame **Format**. Viz Artist/Engine always renders 50 images per second for PAL or 60 for NTSC. Dependent on the post render format, not all of them or not the full frames are stored. Options are:
  - **Full Frame:** Stores 50 or 60 full height frames for each second of video. The number depends on the video settings, PAL/50 or NTSC/60 frames, and the size also changes with the video format (720x576 for PAL digital and 720x486 for NTSC digital). Normally TV is 25 or 30 interleaved frames, but in this case the post rendering produces twice the amount of pixels needed. An editing system (or wherever you put the images you render) can throw away 50% of the pixels and still produce a “normal” video stream.
  - **Fields Top/Fields Bottom:** Stores 50 fields per second of video. It is the same amount of data as interlaced top/bottom, but instead of shuffling the data the post rendering process writes different files for each field.
  - **Interl. Top, Interl. Bottom:** Stores 25 interlaced frames for each second of video. Each frame consists of two fields (half size of PAL or NTSC for example 720x288). These two fields are combined line by line. Interlaced top means that the first line of the first field is the first line in the interlaced frame. In interlaced bottom the first line of the second field is the first line in the interlaced frame. This determines field dominance.

Copyright © 2026 Vizrt

    - **Full Frame Skip:** Ignores every second field (i.e., skips odd-numbered fields if indexing starts from 0), resulting in 25 (or 30 for NTSC) frames per second.
- Select the color settings. These can be
    - **RGB:** Creates RGB images. Here a pixel is represented by three bytes containing a red, green and blue sample.
    - **RGBA:** Creates a red, green and blue sample, along with an additional color sample, all in a single image. When shown on top of other images, RGBA pixels are blended with the background pixel according to the value of this alpha component.
    - **RGB AAA:** Creates an RGB and an Alpha image for each image rendered.
- **Select resolution:** By default, the output resolution is set. You can scale down your output format, but you can not set it higher.
- **Output Destination:** The destination is specified by the Output directory, the filename defines the naming convention. You can use the following placeholders:

Note: First Field top is only used for NTSC format, Standard is First field bottom.

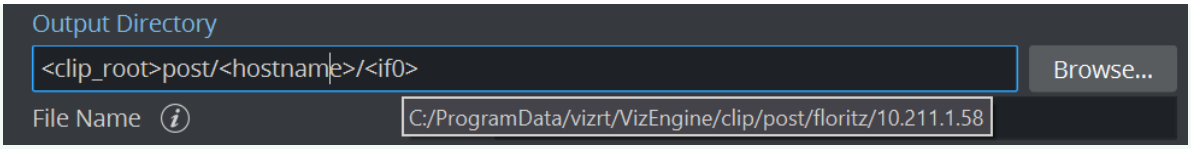
Variable	Definition
`<absolute_scene_name>`	Expands to the complete path of the loaded scene.
`<base_scene_name>`	Expands to the scene name.
`<clip_root>`	Expands to the *clip_root*.
`<hostname>`	Expands to the host name.
`<if0>`	Expands to the IPV4 network address of the first network interface.
`<if1>`	Expands to the IPV4 network address of the second network interface.
`<if2>`	Expands to the IPV4 network address of the third network interface.

Example Usage

```
<clip_root>post/<hostname>/<if0>/<base_scene_name>
```

Tip: Hover over the setting to see how those placeholders will be applied on your system:

Copyright © 2026 Vizrt



- Set an **Offset**, if required. How the offset affects the Scene can be seen in the Post Rendering area.
- If your scene(s) has Stop points in the timeline, these can be converted into Pause intervals (in seconds).
- Finally specify the **Duration** of your recording.
- Click the **Record** Button to start your export.

18.1.3 Post Render Templates

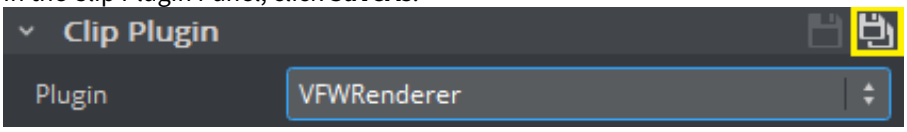
If Scenes are post rendered often, with the same settings, create a Post Render Template. To use a saved template double-click it in the Template panel.

Note: Post rendering templates are saved in the Graphic Hub *and not* on the local machine.

Important: If pools are cleaned up using POOLS CLEANUP or the Graphic Hub is changed, the list of post render templates is deleted and reloaded when opening the post render screen again.

To Save a Post Render Template

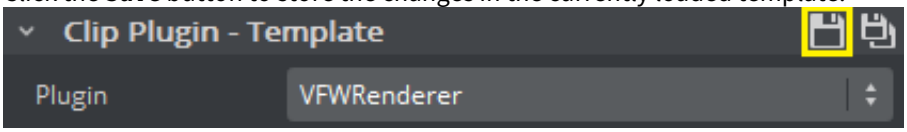
1. Configure all the settings in the procedure To Post Render a Scene.
2. In the Clip Plugin Panel, click **SaveAs**.



3. Enter a name for the template.
4. Click **OK**. Saved templates show in the Templates panel.

To Change a Post Render Template

1. Double-click the post rendering template in the Templates panel.
2. The template name appears in the Clip Plugin header to indicate that it is loaded.
3. Change the required settings.
4. Click the **Save** button to store the changes in the currently loaded template.



To Load a Post Render Template

Double-click the post rendering template in the Templates panel.

Copyright © 2026 Vizrt

To Rename a Post Render Template

1. In the Templates panel, right-click the template and select **Rename** or press **F2**.
2. Enter a new name and click **OK**.

To Delete a Post Render Template

In the Templates panel, right-click the template and select **Delete <name>**.

18.1.4 Post Rendering Plug-ins

Post rendering plug-ins define the format of the post rendered files. Viz Artist offers a set of image and video rendering plug-ins:

VfW Plug-in

Based on *Microsoft Video For Windows* technology. Requires codecs in 64-bit to be installed on the Windows System. Some clips may randomly be marked as truncated resulting in rejection by certain ingest systems. To resolve the issue, version 5.2 of Viz Engine introduced a new flag called *Recalculate Size*. Enabling this option prevents the recordings from being marked as truncated. *Recalculation Size* is used to correct the AVI file header size.

VideoRenderer Plug-in

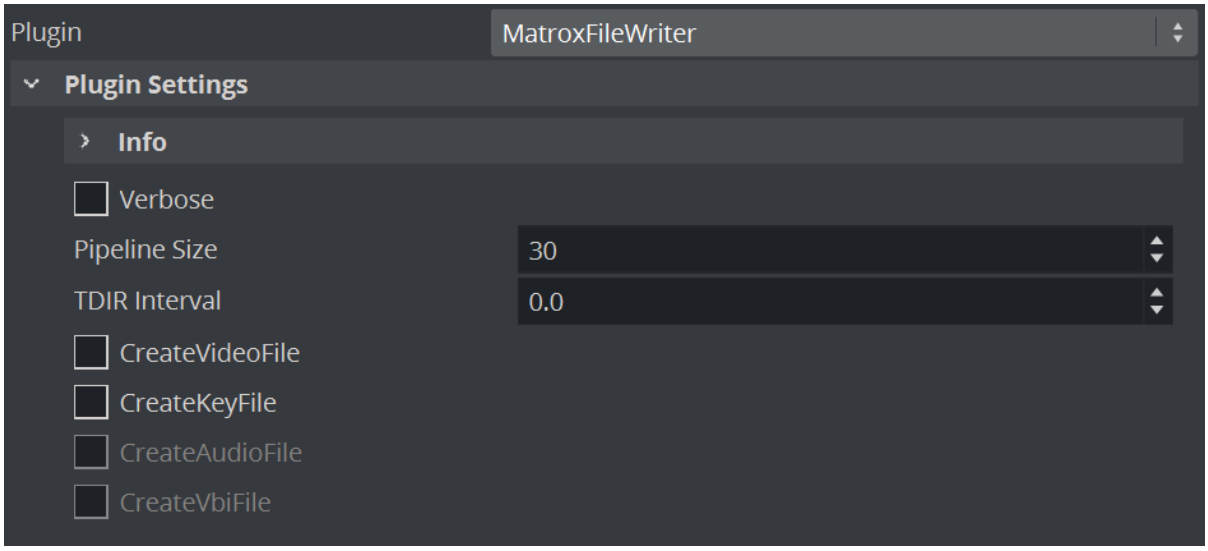
The video rendering plug-in render a Scene to a Video clip using DirectShow. It requires 64bit DirectShow plugins to be installed and the Codec must also include an Encoder. As the number of such plugins is limited, we recommend to use either an image sequence or the Matrox File Writer.

Matrox File Writer

The Matrox File writer is only available, if a Matrox board is present. This can be a native board with Codec upgrade or a DSX.Core virtual license.

These additional settings for the **Matrox File Writer** plug-in:

Copyright © 2026 Vizrt



- **Pipeline Size:** Defines the number of internal buffers in frames for the Matrox framework to operate. For SD+720p+1080i should be around one second (~25-30). It is recommended that you leave the default values.
- **TDIR Interval:** Sets the interval time, in seconds, to update the video file while it is recorded. The value must be `1` or more (less than one does not activate the TDIR feature).

Note: TDIR is only supported on *.avi*, *.p2.mxf*, *xdcam.mxf*, *xdcam_422.mxf*, *.mov*, and *.wav* containers. The MATROX_MOV .mov containers write a separate file (*<basename>.Ref<.extension>*) that is updated to read the *.mov* file. This reference file needs to be used while the file is growing. The QT_MOV containers update the .mov file directly.

- **CreateVideoFile:** Creates a fill clip when set to `On`.
- **CreateKeyFile:** Creates a key file when set to `On`.
- **CreateAudioFile:** Not in use.
- **CreateVbiFile:** Not in use.
- **CreateProxyFile:** Specifies a custom resolution copy of the clip when set to `On`. These settings behave in a similar manner to the video file options and show when this is set to `On`.
- **Video Resolution:** Shows clip resolution information.

Important: Only broadcast resolutions are supported. If a non-broadcast resolution is configured, the plug-in initialization fails with a *Hardware not supported* message.

See Also

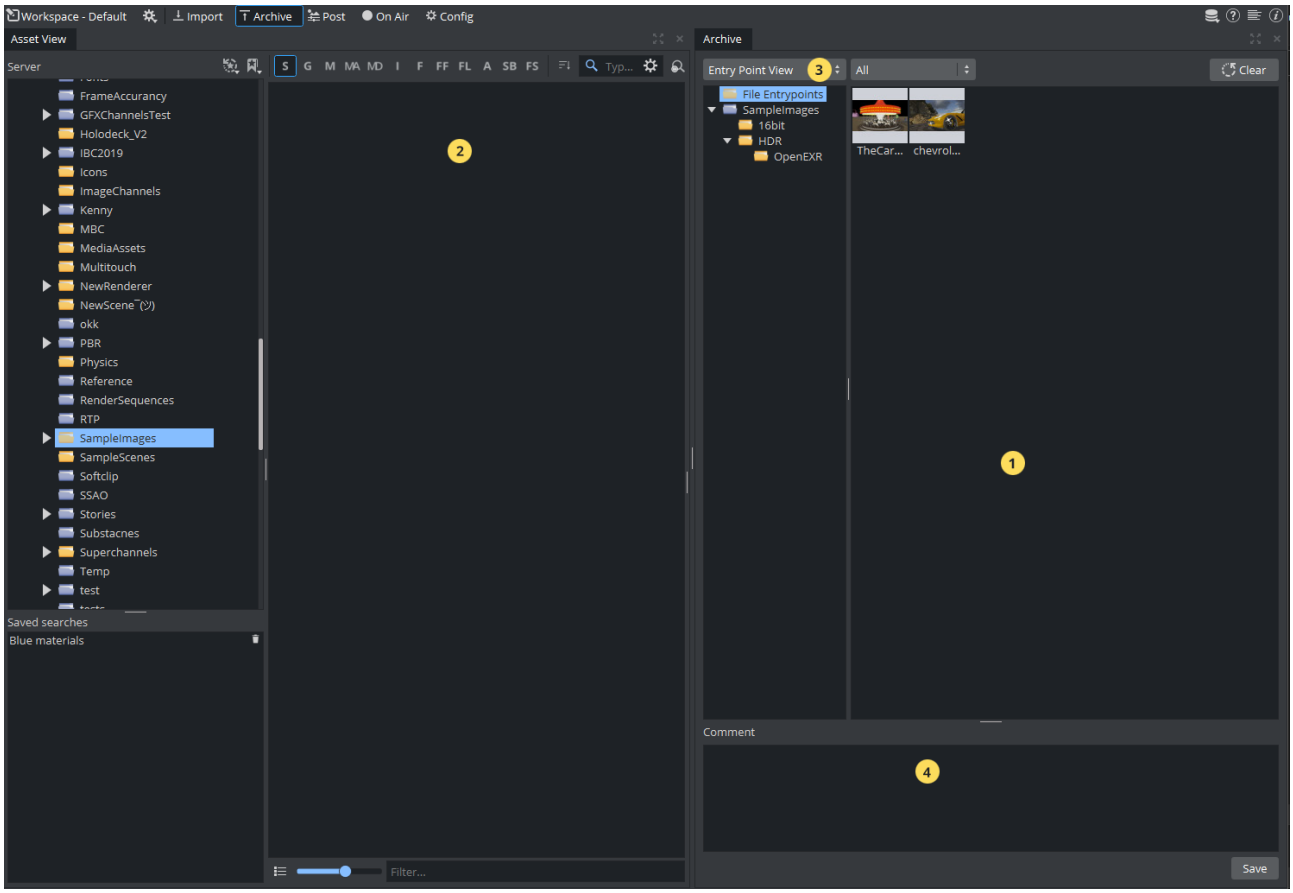
- **Video Output Clip Output** section of the Viz Engine Administrator Guide

Copyright © 2026 Vizrt

18.2 Archives

The archive function exports graphical resources, such as one or several Scenes including fonts, images and so on to a file (archive) on disk. The archive has, by convention, a *.via* filename extension. The archive can be used for the safekeeping and/or for transferring of data to another Graphics Hub.

18.2.1 Archive Workspace

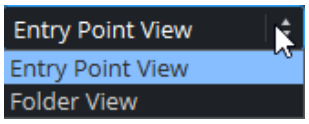


The archive workspace is divided into the following areas

- **Archive View (1):** Shows which items are going to be included in your archive.
- **Asset View (2):** Displays the content of your Graphic Hub. Drag items from (2) into (1) to add content to your archive.
- **Entry Points Menu (3):** Switches between folder view and Entry point view.
- **Comments (4):** Provides an area to add descriptive comments about the archive.

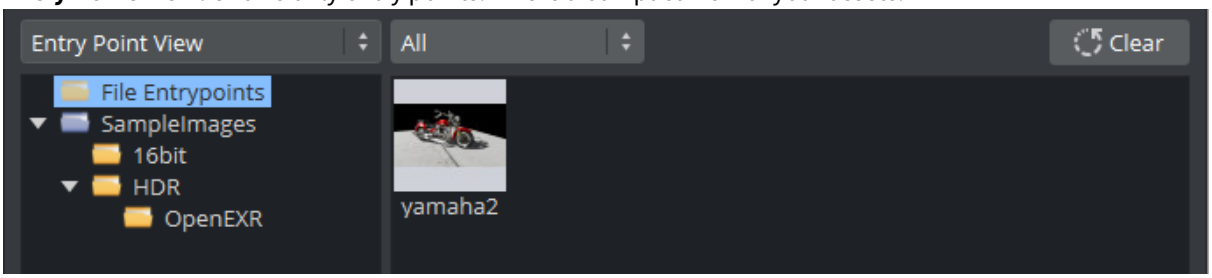
Copyright © 2026 Vizrt

Entry Point vs Folder View



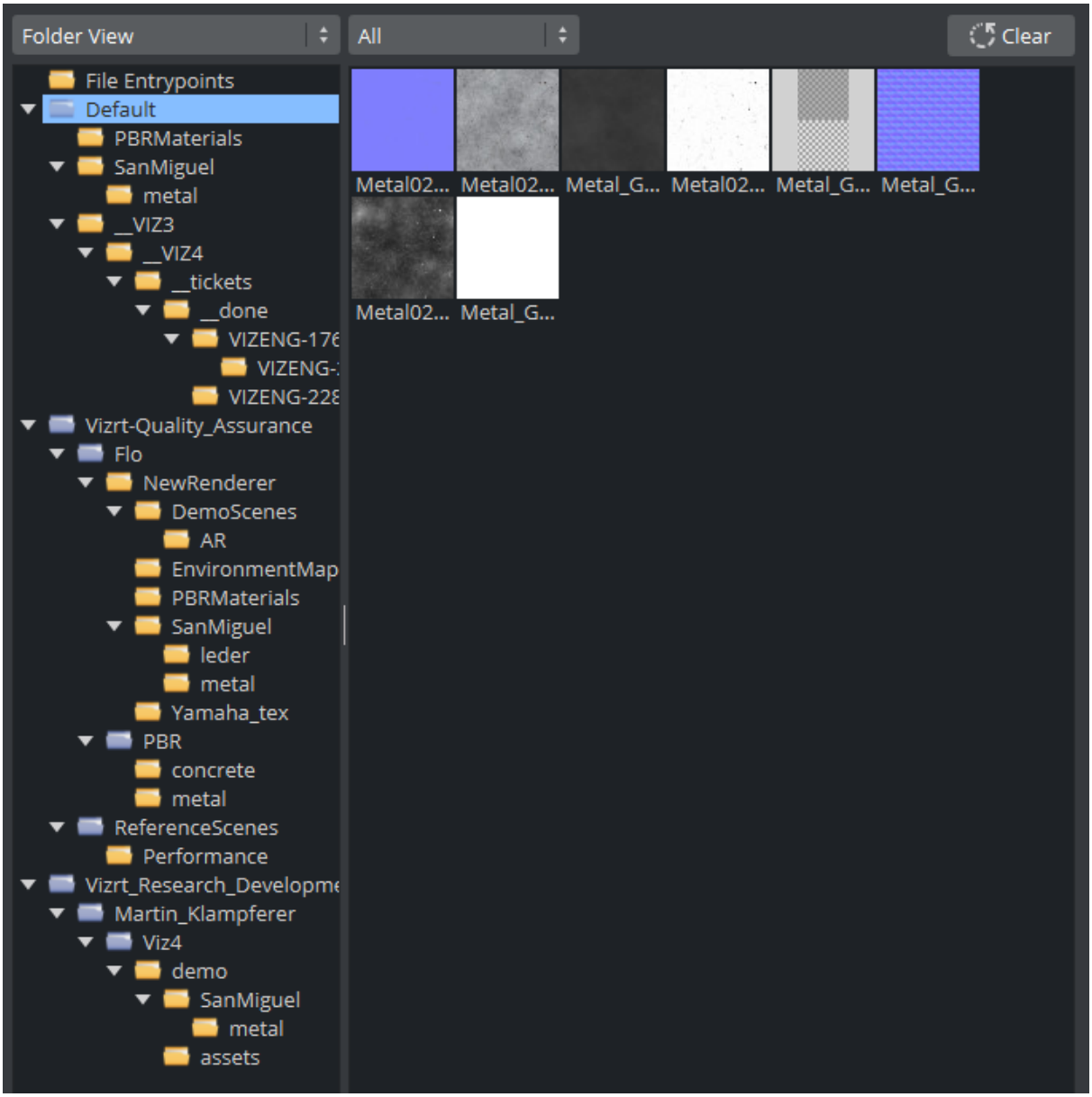
At the top of the archive view there is a pull-down menu enabling you to switch the way the content of your archive is displayed. Entry points represents what was added to the archive. This can be folders, project scene or single items. A scene, for example consists of many different types, like textures, fonts, etc. A entry point represents this scene only (without any file links and references), whereas the Folder View shows all files that will be added to the archive.

- **Entry Point View:** Shows only entry points. This is a compact view of your assets.



- **Folder View:** Shows the whole folder hierarchy, including all folder links.

Copyright © 2026 Vizrt



You can choose which items are displayed on the item dropdown. to filter the content being displayed.

Information: If a scene contains an item with folder links, all folder links are added in your archive.

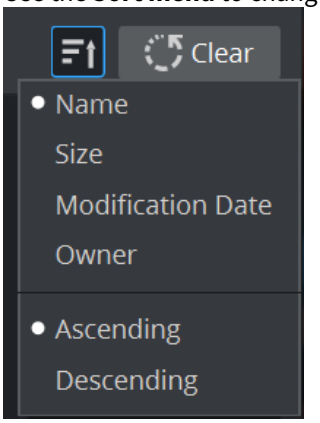
18.2.2 To Create an Archive

1. In the default workspace, click Archive (Archive). The Archive view is shown.
2. From the **Asset View (2)**, drag folders or projects the the Tree view (left) of the Archive View (1). Any other item (scenes, images, etc.) need to be dragged to right part of the Archive View (1).
3. If required, enter a comment in the comments area (4).

Copyright © 2026 Vizrt

Note: Comments can be a maximum of 255 characters.

5. The Archive is updated as soon as entry points are added.
6. Use the **Sort menu** to change how assets are displayed.



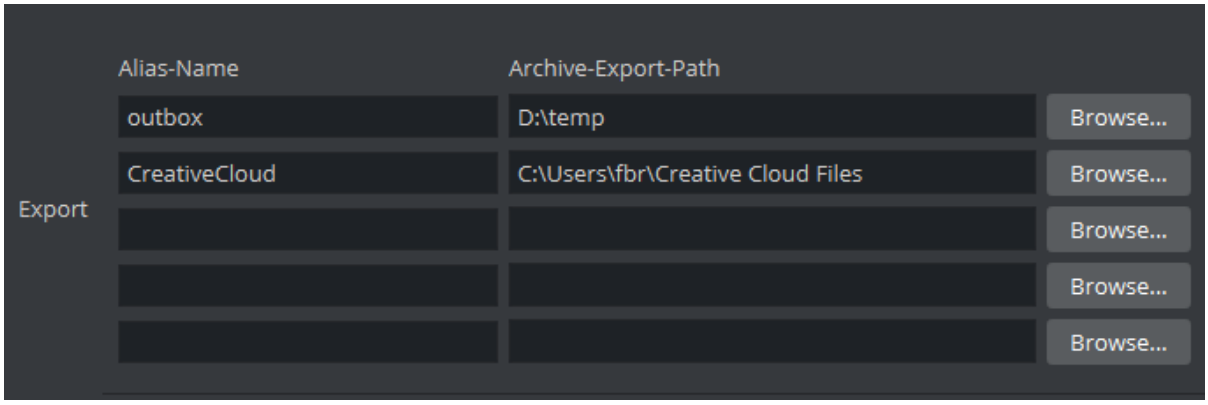
7. Click the **Save** button, and choose location and filename.
8. To reset the content of your archive, use the Clear button. This removes any item previously added to your archive.

A *.via* file is written. This file can be shared and re-imported again on another Graphic Hub by either using Viz Artist, Viz Engine in On-Air Mode or by using Viz Graphic Hub Manager.

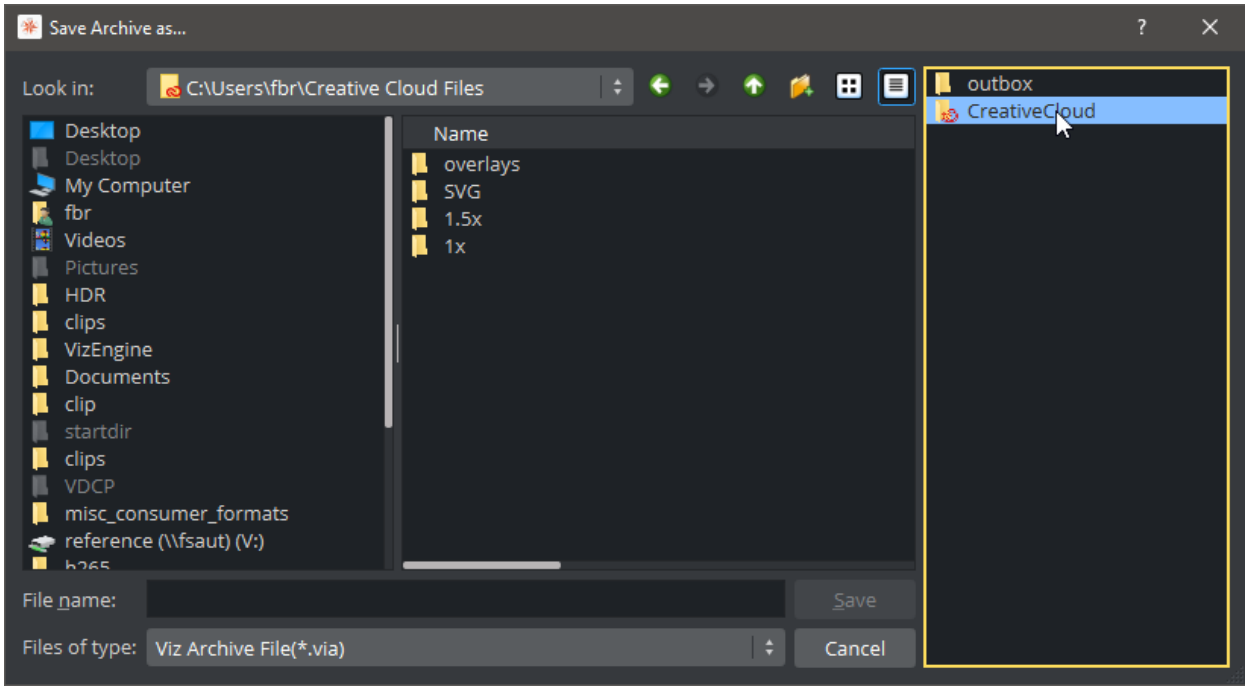
Note: Content can not be added to your archive after it has been written.

18.2.3 To Use Path Aliases

Like Path Aliases for importing Files, a list of given export locations can be defined in **Config Path Aliases**.



Set a Name for your Path Alias and provide a destination. They can be used within the export dialog when creating an archive.



Copyright © 2026 Vizrt

18.3 Import of Files and Archives

The following chapter explains how to get assets into Viz Artist.

- Import Using Asset View
- Import Window
- To Import Various File Types
  - Import of Images
    - SVG Images
  - Images for PBR Material Definitions
  - Supported Image Formats
  - Import for Scenes
    - Autodesk FBX
    - After Effects
      - Import Limitations
    - Cinema 4D
    - Photoshop
  - Import of Fonts
  - Import of Geometries
  - Import of Audio
  - Import of Substances
  - Path Aliases
- To Import Archives
  - View Archive Content
    - Sorting Options
  - To Import an Archive "as is"
  - To Import an Archive to a Specific Location in a Graphic Hub
  - To Import Part of an Archive
  - To Search for Imported Files
  - To Import an Archive in Viz Engine Mode

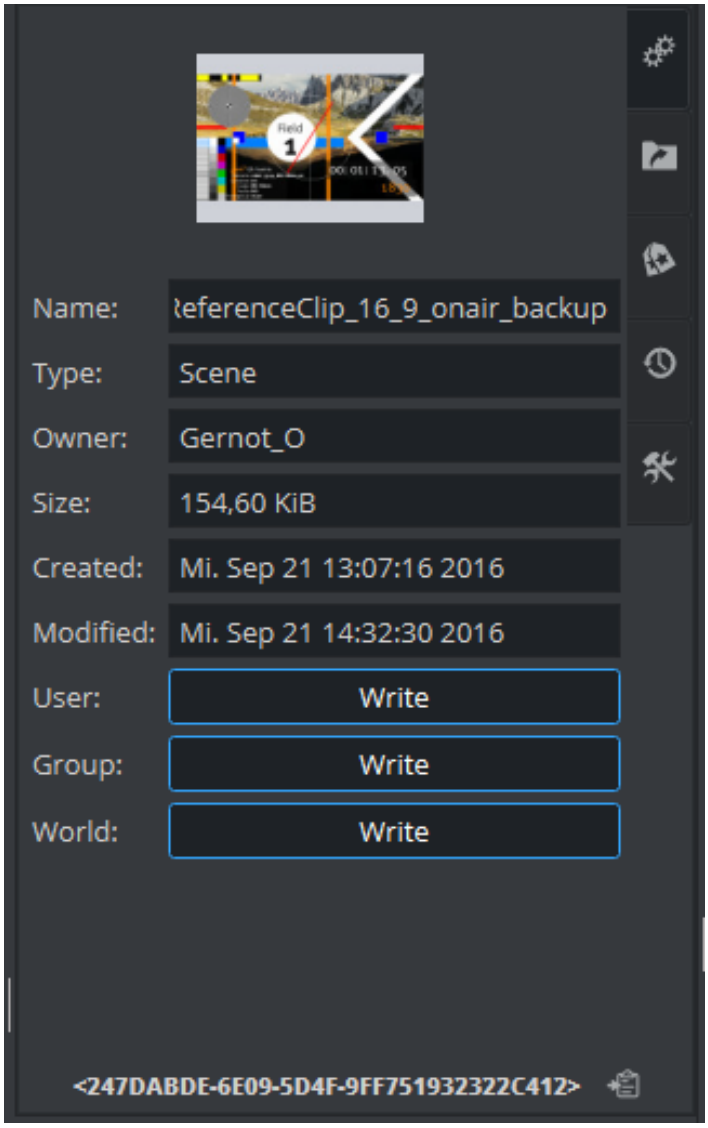
Viz Artist can import Files (Fonts, Images, Geometries, Scenes, Audio clips) and Archives (archives have the filename extension *.via*). Archives are, if possible, self contained and complete. This means that if a user archives one or more Scenes, then **all** referenced items in the scene, including links, fonts, images and so on, are archived. A user can choose to archive only individual items such as some fonts or some images.

All items used in Viz Artist are stored in Graphic Hub (GH). Viz Artist connects to the GH through a network connection, even if the GH is local. Graphic Hub is a **database**, and every file in Graphic Hub is uniquely identified by its **UUID** (a unique identifier number). Files in Graphic Hub can have links. This means that possibly several Paths/ Names can link (or point) to the same file in the GH database.

Note: Special characters like **%** and ***** are automatically renamed to _ as these characters are used internally.

Details of an item in Graphic Hub are shown in the properties panel:

Copyright © 2026 Vizrt



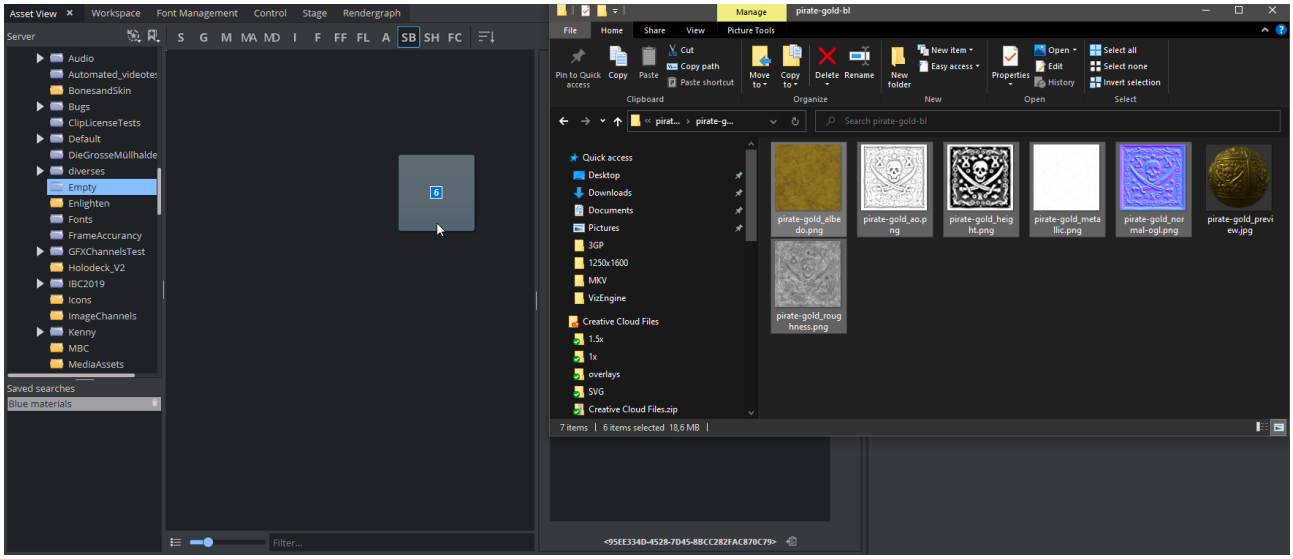
Tip: It is strongly recommended to have a clear and well communicated plan for organizing the paths and naming conventions for all resources in Graphic Hub. This makes life easier for users and import/export operations are easier to do without the creation of clutter.

Note: If Viz Artist is running in Viz Engine mode, the import is limited to only import archives.

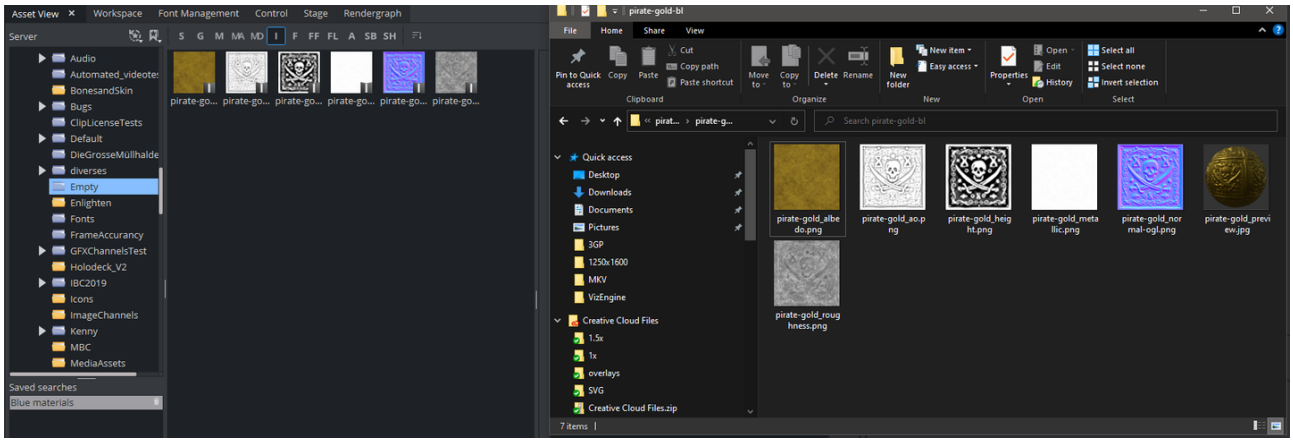
18.3.1 Import Using Asset View

Asset view can be used to import files via drag and drop from Windows Explorer. To do so, select your files in Windows Explorer, drag them into Viz Artist and drop them onto the Asset View.

Copyright © 2026 Vizrt



Depending on the type of file you dragged, a dialog pops up asking how to import your files. For images you need to select compression and quality. Scenes (*.fbx*and *.c4d*) are imported in the background, so there is no visual feedback. After the import has finished, items show up in the Asset view.



You can drag multiple different types at once to the asset view. For each file extension, the import settings dialog is shown.

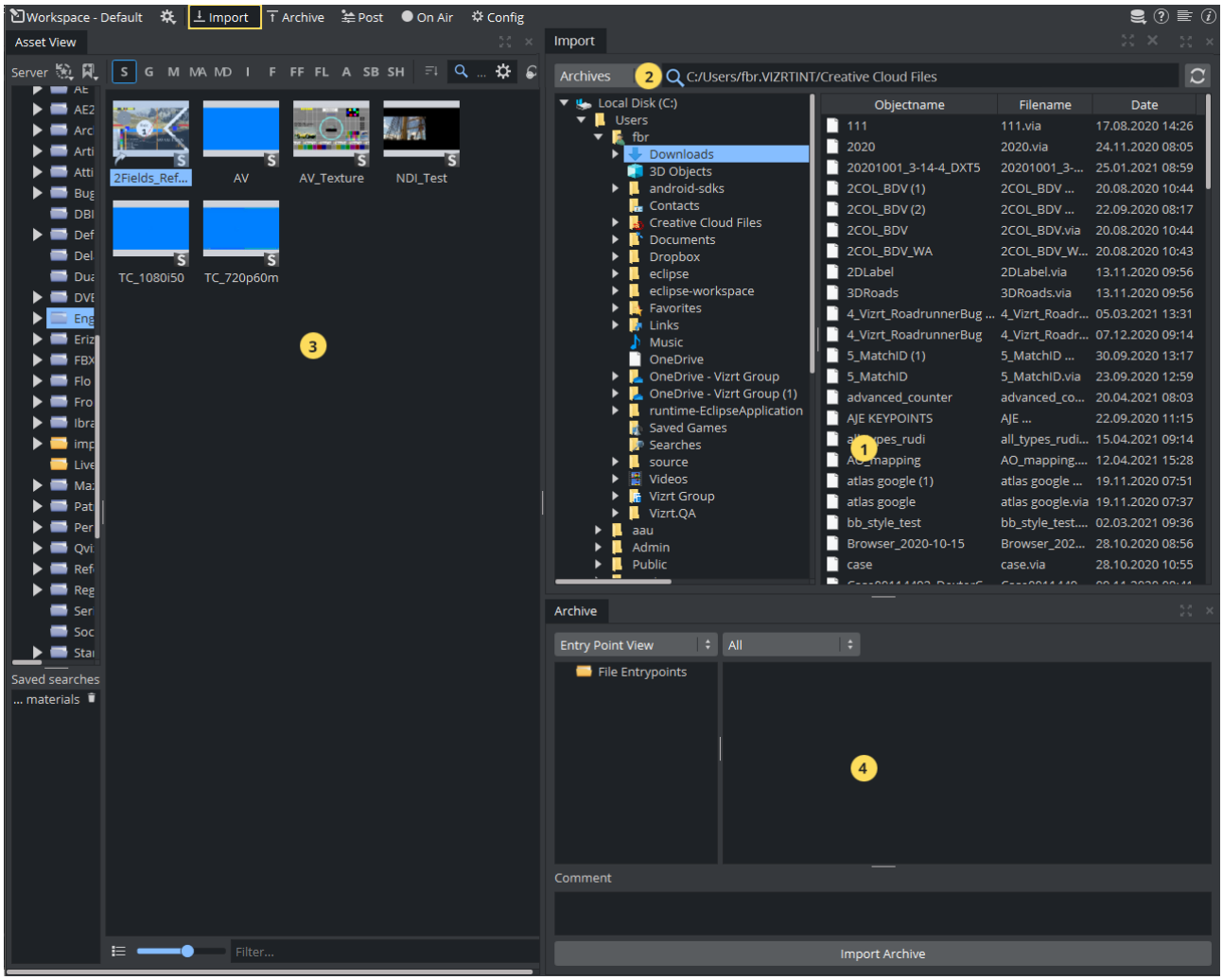
Information: *.zip* files can also be dragged into the asset view. The content is automatically extracted and imported.

Information: Adobe After Effect files are limited to one item only. Multiple After Effect scenes are not supported for import.

Copyright © 2026 Vizrt

18.3.2 Import Window

Another way to import files, is to use the Import Window. To Open the Import Window, click on the **Import** entry in the top menu.



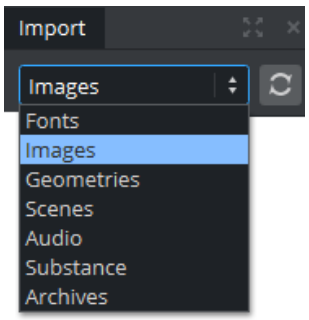
The Import Window consists of the following sections:

- **Import Panel (1):** Gives access to the source drives, where assets can be imported from.
- **Import Menu (2):** Selects what kind of assets are listed in the Import Panel.
- **Server Panel (3):** Gives access to all folders and projects in Graphic Hub.
- **Archive View (4):** Shows the content of Archives.

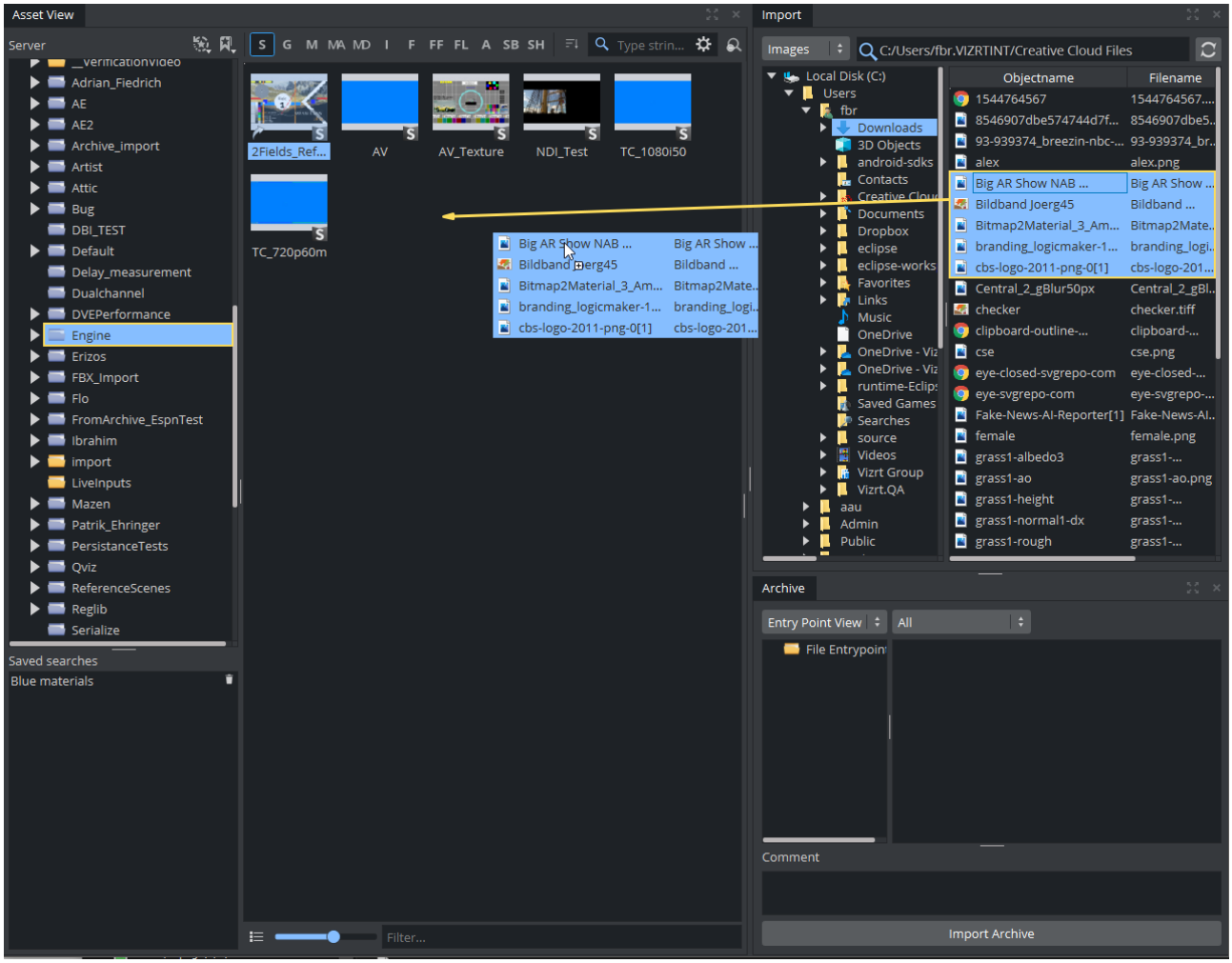
Note: Folder Shortcuts in the file system are displayed with a - *Shortcut* suffix. The Shortcut does not display the subfolders of the target folder. Double-click on the shortcut to jump to the target folder of the shortcut.

Copyright © 2026 Vizrt

Make sure you correct type has been chosen in the Import Menu:

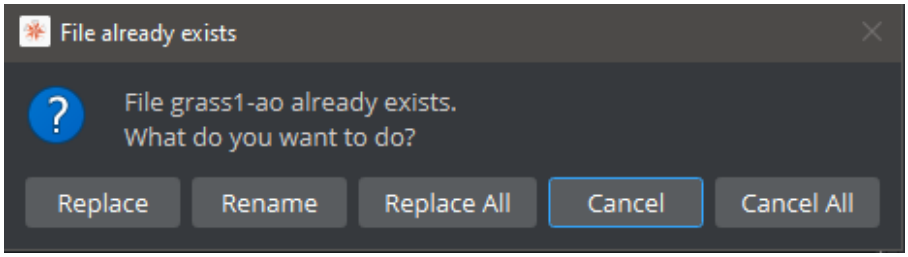


To import any asset, make sure the correct destination folder has been chosen in the Server Panel. If your destination folder does not yet exist, right click on one of the folder/project entries in the server panel and create a new folder or project as your new destination.



Grab the files to be imported from the Import Panel and drop them over to the Server Panel.

If a reference to a file, which is to be imported, already exists in Graphic Hub, this dialog shows:



Select from:

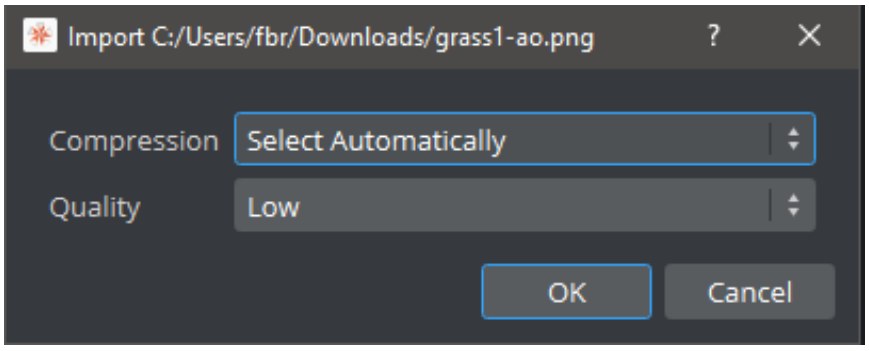
- **Replace:** Overwrites the file on Graphic Hub.
- **Rename:** Renames the file and import (new file). Opens a new dialog, type a new name and press **OK.**
- **Replace All:** Stops the file import of the file shown in dialog window. The import procedure continues with the next file.
- **Cancel:** Overwrites all files if there are multiple files.
- **Cancel All:** Stops the import of all files.

If further options are required, an option window prompts for those. This window shows up for every single file extension. If you want to import several *.jpg* images with different options, you need to import them file by file to specify the options.

18.3.3 To Import Various File Types

Import of Images

Images on Graphic Hub are prepared to be used as textures. This means a compression can be applied and mipmap levels are being generated. Compressed images are smaller in size.



The compression can be set to be:

- **None:** No compression is applied.
- **Select Automatically:** Either DXT1 or DXT5 is used, depending if your image has alpha information.

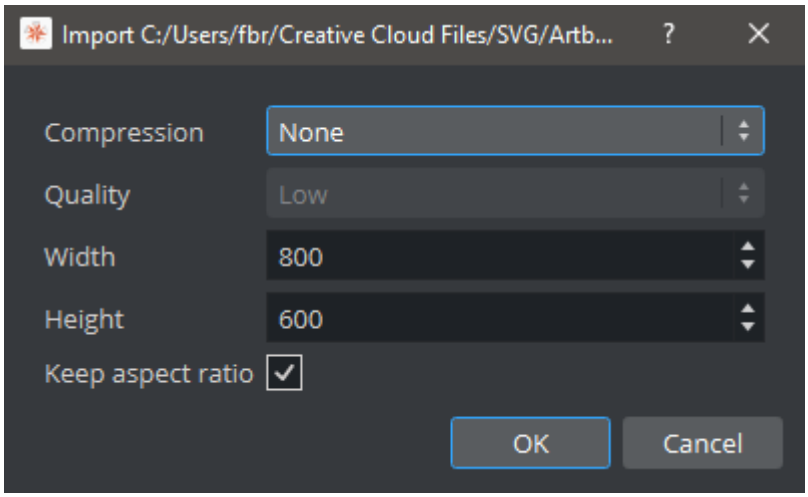
Copyright © 2026 Vizrt

- **DXT1:** Compression for images without alpha.
- **DXT5:** Compression for images with alpha.

The quality defines the compression for *.png* and *.jpg* images. it can be chosen between **Low**, **Medium** and **High**.

> **Note:** The maximum dimensions of an image that can be imported and used as a texture in Viz Artist is 16384 pixels in width or height.

SVG Images



SVG files are rendered into images, therefore a few more options need to be provided as the final **width** and **height**, combined within the keep aspect ratio settings.

> **Note:** Since Viz Engine version 5.3, the DPI value on importing is fixed to 96.

Images for PBR Material Definitions

Please see Improved Image Import for PBR Materials.

Supported Image Formats

Format	Importable (with/without compression)	Exportable	Supported by the Classic Render Pipeline	Supported by the Viz Engine Render Pipeline
BMP	Yes	Yes	Yes	Yes
PNG(1)	Yes	Yes	Yes	Yes

Copyright © 2026 Vizrt

Format	Importable (with/without compression)	Exportable	Supported by the Classic Render Pipeline	Supported by the Viz Engine Render Pipeline
TGA	Yes	Yes	Yes	Yes
JPEG	Yes	Yes	Yes	Yes
HDR(2)	Yes	Yes	Yes	Yes
EXR	Yes	No	Yes	Yes
VBN(3)	Yes	No	Yes	Yes
TIF	Yes	Yes	Yes	Yes
PSD(4)	Yes	Yes	Yes	Yes
GIF	Yes	No	Yes	Yes
SVG	Yes	No	Yes	Yes

(1) No support for 16-bit per channel PNG images.

(2) Requires use of the HDR plug-in.

(3) Compressed images are not supported.

(4) The import of Photoshop files has been replaced by Photoshop Scene import. It is still possible to import image-only photoshop files with the command: `IMAGE IMPORT Destination Name PathToFile DXTSMART`

EXIF information is read when importing JPEG information. Images are automatically rotated based on the rotation information stored within the image.

Import for Scenes

Scenes created in certain other 3D applications can be imported to Viz Artist. However, any geometry that is to be imported must be a polygon geometry. Pay special attention to this when importing geometries from CINEMA 4D, as these must have been saved as a *.c4d* file with the **Save Polygons for Melange** option enabled (Located under **Edit > Preferences > Files** in CINEMA 4D), to make sure that all items saved are stored with their corresponding polygon data.

Format	Comments
Autodesk (*.fbx*)	FBX is a 3D asset exchange format compatible with many 3D tools (see FBX Files).

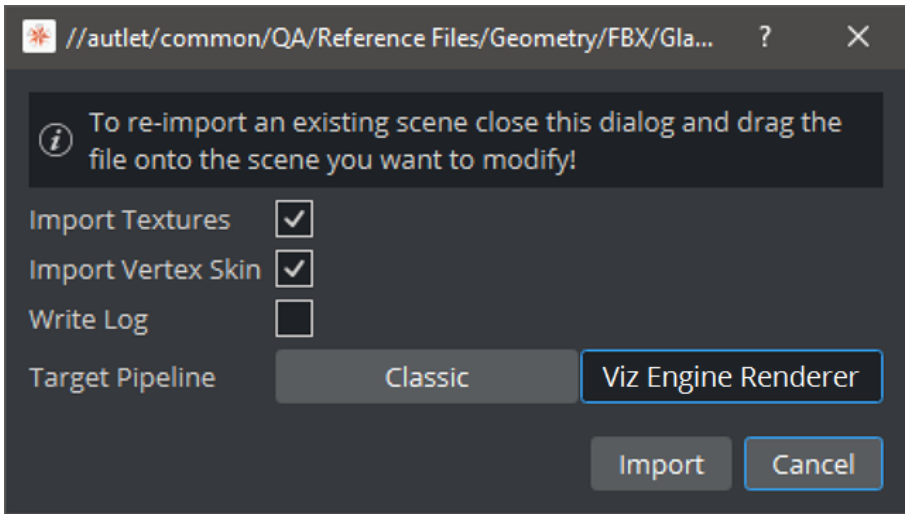
Copyright © 2026 Vizrt

Format	Comments
After Effects (*.aep*)	See Adobe After Effects.
CINEMA 4D (*.c4d*)	See CINEMA 4D. **Note:** A license is required for the import of CINEMA 4D Scene, issued by Vizrt.
VRML (*.wrl*)	Deprecated format, Should not be used anymore.

Autodesk FBX

FBX is a very common file format that most 3D applications support. It is the recommended format to bring 3D scenes into Viz Engine.

FBX supports Lights, Camera, Textures, Materials and Bones. When importing an FBX file, the following dialog shows up:



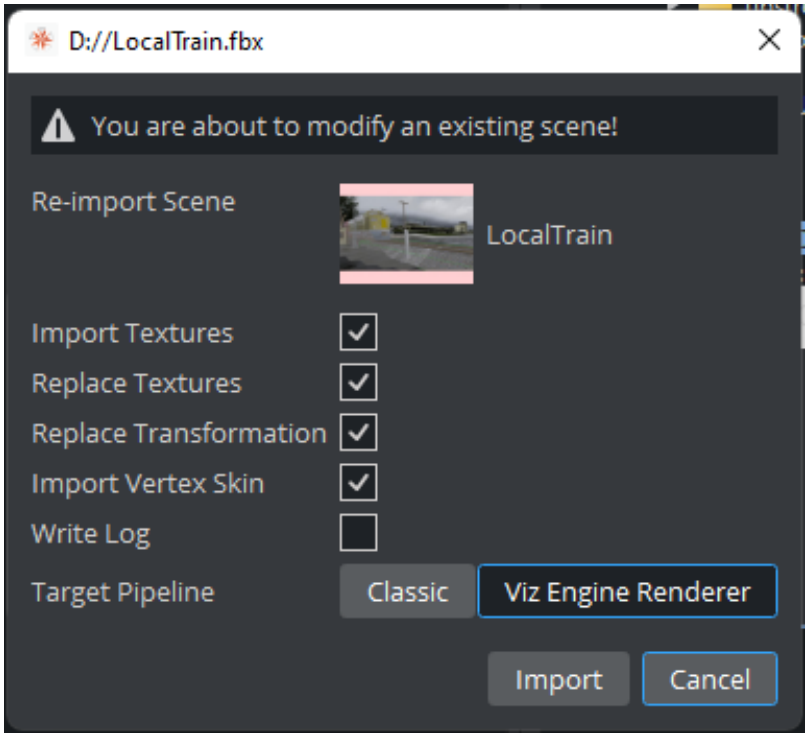
Then, choosing **Viz Engine** as Target pipeline, the importer creates PBR material definitions. The importer either imports "Stingray PBS" materials from Maya or tries to derive some basic values from any other Material available in the *.fbx* file. In **Classic**, basic Materials are generated.

If your FBX file contains bones/vertex skinning animations, these options needs to be enabled explicitly. The maximum levels of bones is currently restricted to 256.

The **Log File** is located at *C:\ProgramData\vizrt\VizEngine\Engine_00_FBXImport_20210604Z115647.log*

When you change a FBX scene already imported in Viz Artist, the scene can be easily reimported by dragging the changed FBX file onto the Viz Artist scene.

Copyright © 2026 Vizrt



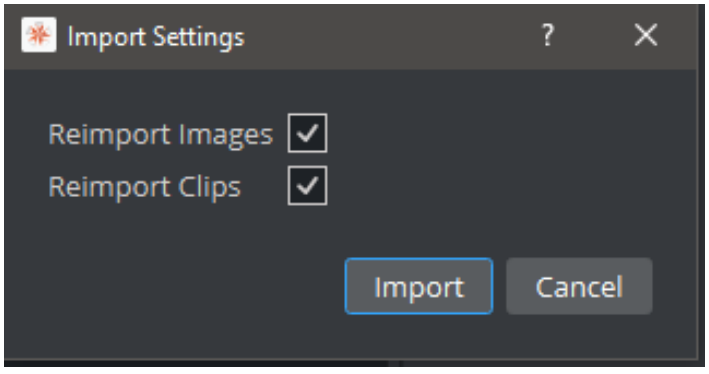
Note: To allow correct re-importing, the VizUUID plug-in needs to be used. This helps identify existing objects.

Detailed information about how to handle FBX files can be found in the Third Party Integrations section.

After Effects

When you import an Adobe After Effects project to Viz Artist, it automatically creates containers for text and image items, and then adds the Control Text and Control Image plug-ins for each text or image container. The import process creates the sub-folder *import* within the target folder, and then imports images, fonts, and sub-scenes, to this sub-folder. It does not import fonts that already exist on Graphic Hub, but references the already existing fonts in instead.

When re-importing a scene, you must select if Viz Artist also re-imports images and clips:



Copyright © 2026 Vizrt

- Import and overwrite existing files (**Enabled**) or,
- Do not import files which already exist (**Disabled**).

Set the respective **Import Settings** to disabled if there are no changes to the image or clip files. The speeds up the import process, as it bypasses the Adobe After Effects renderer.

Items that Viz Artist cannot interpret, or that cannot be rendered in real-time (such as special effects), are rendered as a clip by the After Effects renderer. Because of this, Open **Viz Configuration** and make sure that the **Output Format** setting in Viz Artist matches the **Composition** setting in Adobe After Effects. Also, activate the number of clip channels you require in the **Video Input** section, and enable the **w/Alpha** option. Otherwise, Viz cannot play out these clips.

Viz Artist automatically imports these clips to the Graphic Hub as a Video Clip item, and composes them in the imported Scene. Like any other file in the Graphic Hub, the Scene references these clips by UUID.

Import Limitations

Import of Adobe After Effects compositions is limited to the following Adobe After Effects items:

- Cameras
- Texts
- Solids
- Masks (no animated points and no feathering)
- Shapes (no animated points and no feathering)
- Images

Items included in nested compositions are imported/exported in the same way as items that belong to a single (root) composition.

IMPORTANT! Adobe After Effects needs to be installed on the same machine as Viz Engine.

These items are imported with their animations (Key Frames are not baked, they are converted directly to Viz Artist Key Frames).

All items and/or effects from Adobe After Effects, which cannot be converted to Viz Artist objects, are rendered as **video clips** and imported into the Graphic Hub. The import tries to combine multiple objects, which cannot be imported, and render them into one video (MPEG2 iFrame HD + Alpha). Currently, the importer is configured to use *no more* than two *Clip-* and *16 GFX* channels for one scene (including sub-scenes). If more channels are necessary to import the composition, the importer renders layers which are actually supported as native items, but create an importable/not-importable alternation, as clip.

For example, if an imported After Effects scene would need more than two clip/16 GFX channels in Viz Artist, then the importer collapses those layers step by step until the resulting import does **not** need more than 2/16 channels. However, this *may* lead to the situation that layers/items which are actually supported as native items in Viz, are still imported as pre-rendered clips. You may want to consider this when working in After Effects before importing into Viz.

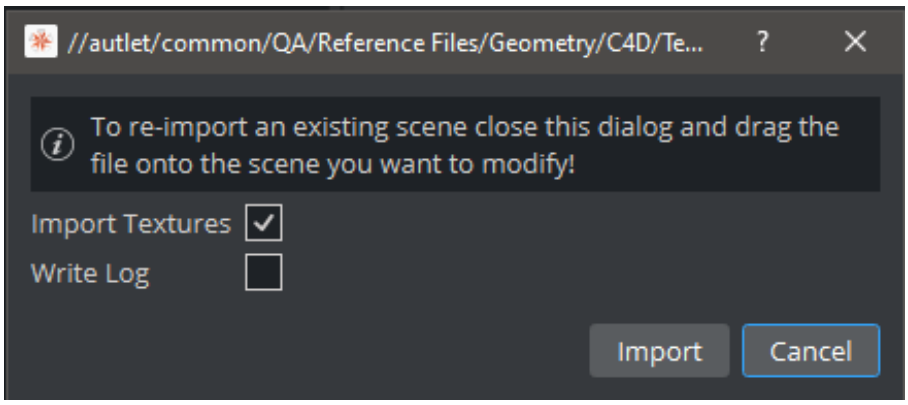
Note: This can lead to several video clips if there are many alternations of importable and not importable layers.

Adobe After Effects offers a 2D/3D layer option. In Viz Artist, there is one camera per Scene. This means that each time 2D and 3D layers are mixed in the composition the import needs to create a new sub-scene (GFX Channel) to set a camera accordingly and still maintain the correct layer Z-sorting.

Copyright © 2026 Vizrt

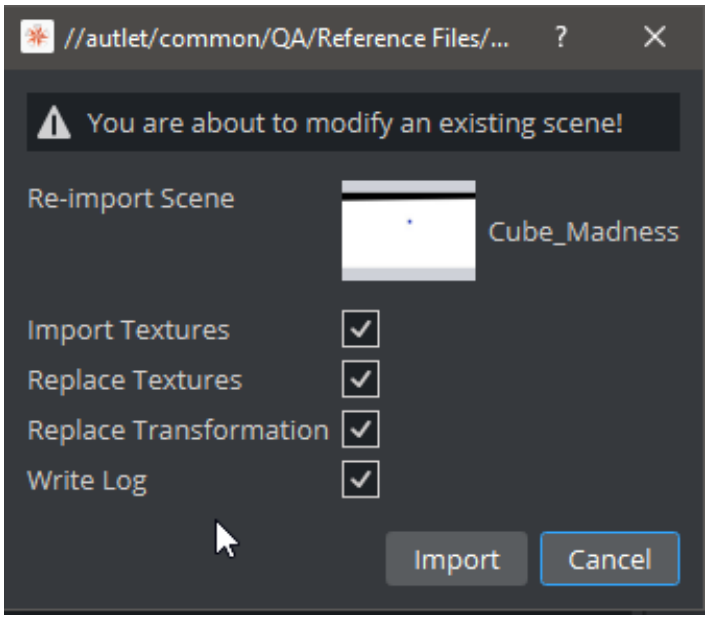
Cinema 4D

Cinema 4D files are always imported as Classic scenes. Only basic texture mapping can be chosen, therefore you can only specify if textures are imported or not.



The **Log File** is located at *C:\ProgramData\vizrt\VizEngine\Engine_00_C4DImport_20210604Z114310.log*

When you change a CINEMA 4D scene already imported in Viz Artist, the scene can be be easily reimported by dragging the changed *.c4d* file onto the Viz Artist scene.



Information: Cinema 4D import needs to be licensed!

Photoshop

For information about importing Photoshop files as scenes, please see Photoshop Import

Copyright © 2026 Vizrt

Import of Fonts

The most common file formats for fonts are supported. Please observe however, that path rendering of fonts is not supported for fonts imported with **Font Import** set to *Windows (Advanced)* in **Font/Text Options** in Viz Configuration.

When importing Classic fonts, Blur and Outline variants are created automatically.

Type	Comments
Adobe Type1 fonts	Postscript Type 1 fonts (*.pfb* files).
Adobe Type2 fonts	Postscript Type 2 fonts.
True type fonts	Recommended, most common format.

Tip: To import Fonts for Text, please use the **Font Management**

Import of Geometries

The following formats can be imported for geometries. It is recommend to use FBX if possible.

Format	Comments
Autocad (*.dxf*)	Drawing Interchange Format.
Wavefront (*.obj*)	Geometry definition file format
3D studio (*.3ds*)	3D-Studio
(.glb) and (.glTF)	Graphics Language Transmission Format
GeoJson (.geoJson) FlatGeoBuffers (.fgb)	Geographic Shape data for Maps rendering.

IMPORTANT! Autocad (*.dxf*) files to be imported need to have 3DFACE entity.

Import of Audio

Audio files need to be 48kHz -16 bit. Maximum 16 channels, minimum two (stereo) as mono is not supported. The following file formats can be imported:

Copyright © 2026 Vizrt

Format	Comments
WAV	Waveform Audio File Format
OGG	Ogg Vorbis
MP3	Mpeg Layer 3 Audio

Import of Substances

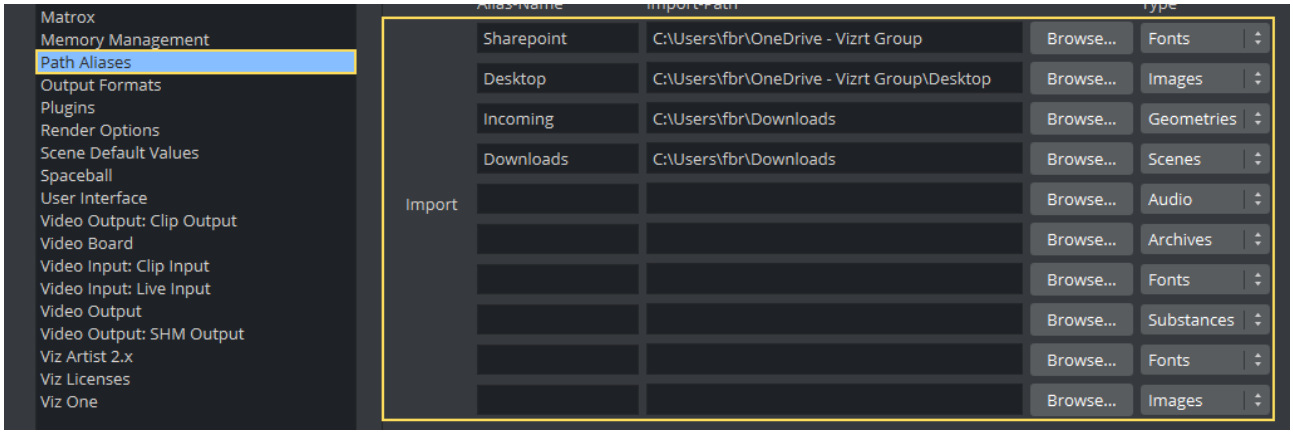


Substance materials are supported by the Substance Shader. There are no options for the Substance Import.

Format	Comment
.sbsar	Substance Designer Archive

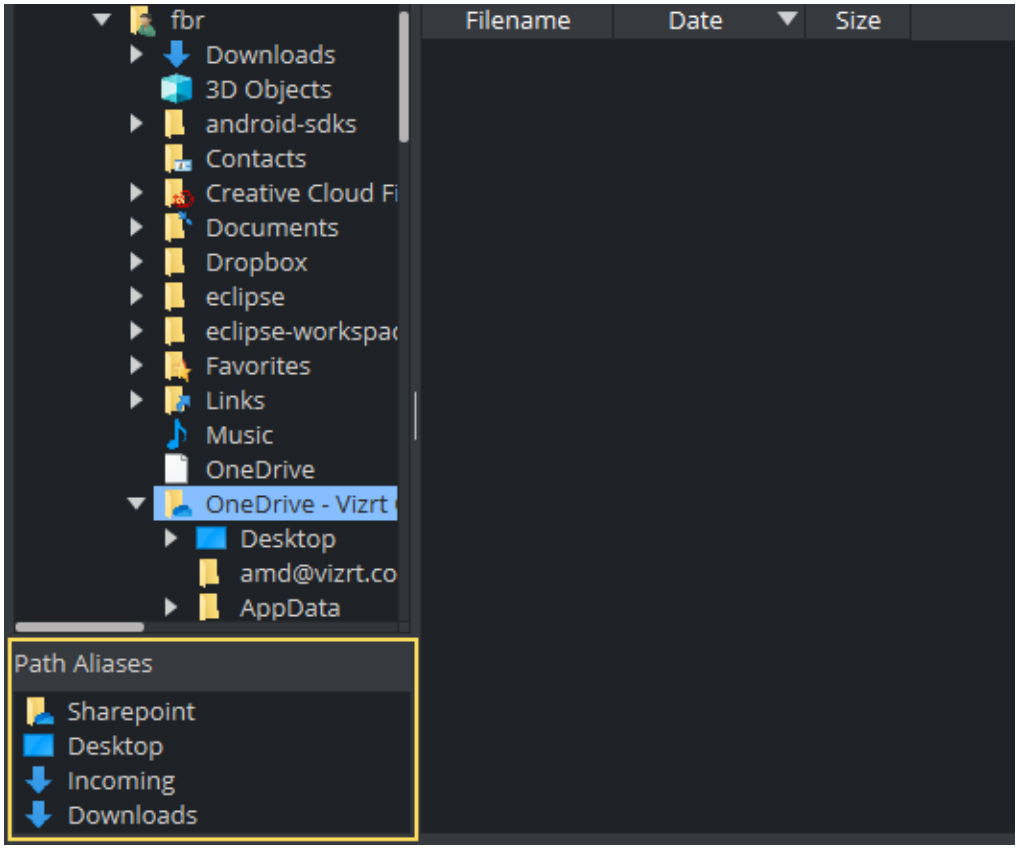
Path Aliases

Use a path alias to help streamline the import process. Path Aliases, defined in Viz Configuration, create shortcuts to frequently used file paths for import operations. This switches the default path (for example, images to be your Camera or Teams-synchronized Sharepoint folder). To define them, set your Aliases in **Viz Configuration Path Aliases.**



Once they are set, the import shows them as a list. Clicking on them directly navigates to the selected folder.

Copyright © 2026 Vizrt



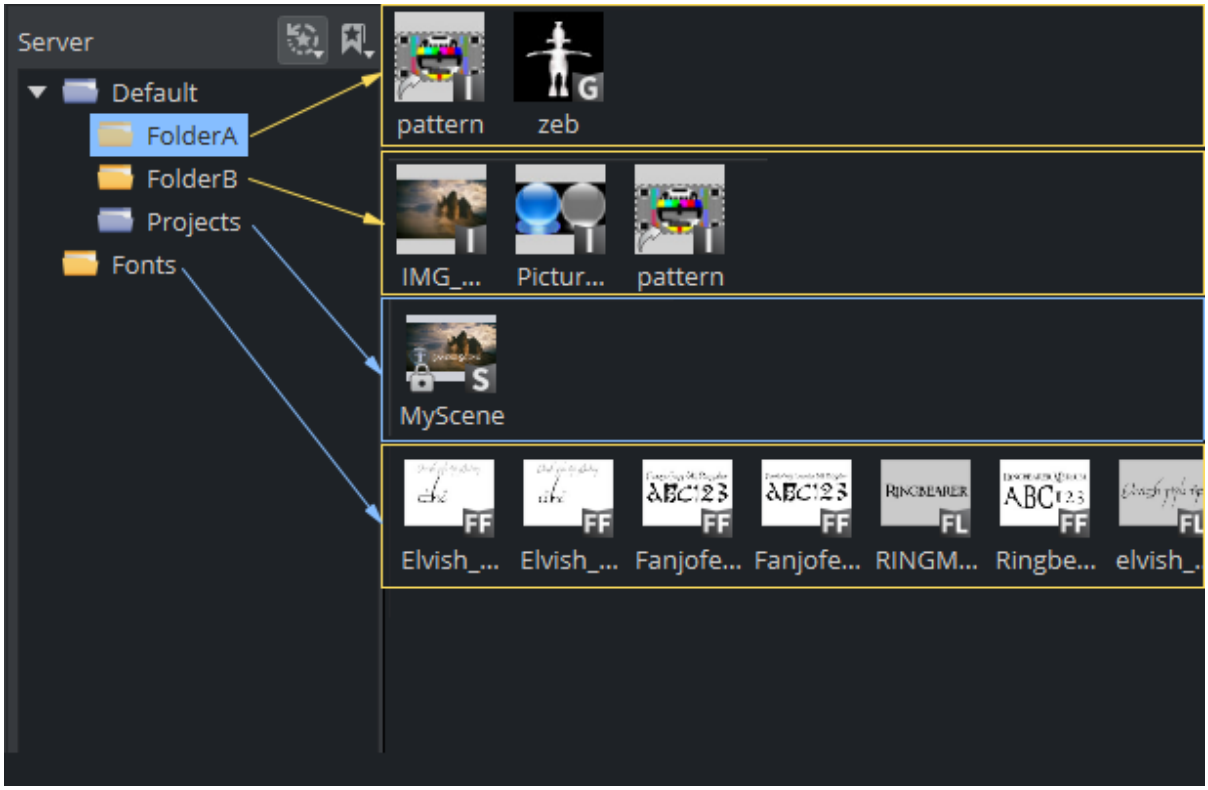
18.3.4 To Import Archives

An Archive (filename extension *.via*) is a single file that holds various types of items and remembers the project or folder they were stored. Archives are typically used for backup and for transferring Scenes from one Graphic Hub to another.

IMPORTANT! Archives created with Viz Artist 2.x must not be imported to Viz Artist 3.x in this way. They need to be imported using the Graphic Hub Manager (see the Graphic Hub User Guide).

For the following examples, lets assume one has the following basic structure defined on Graphic Hub:

Copyright © 2026 Vizrt



- **Folder A** consists of an image and a geometry, the image is also linked into Folder B.
- **Folder B** has three images.
- **Projects** holds a scene.
- **Fonts** has some Fonts imported.

A scene is being built (*MyScene*):

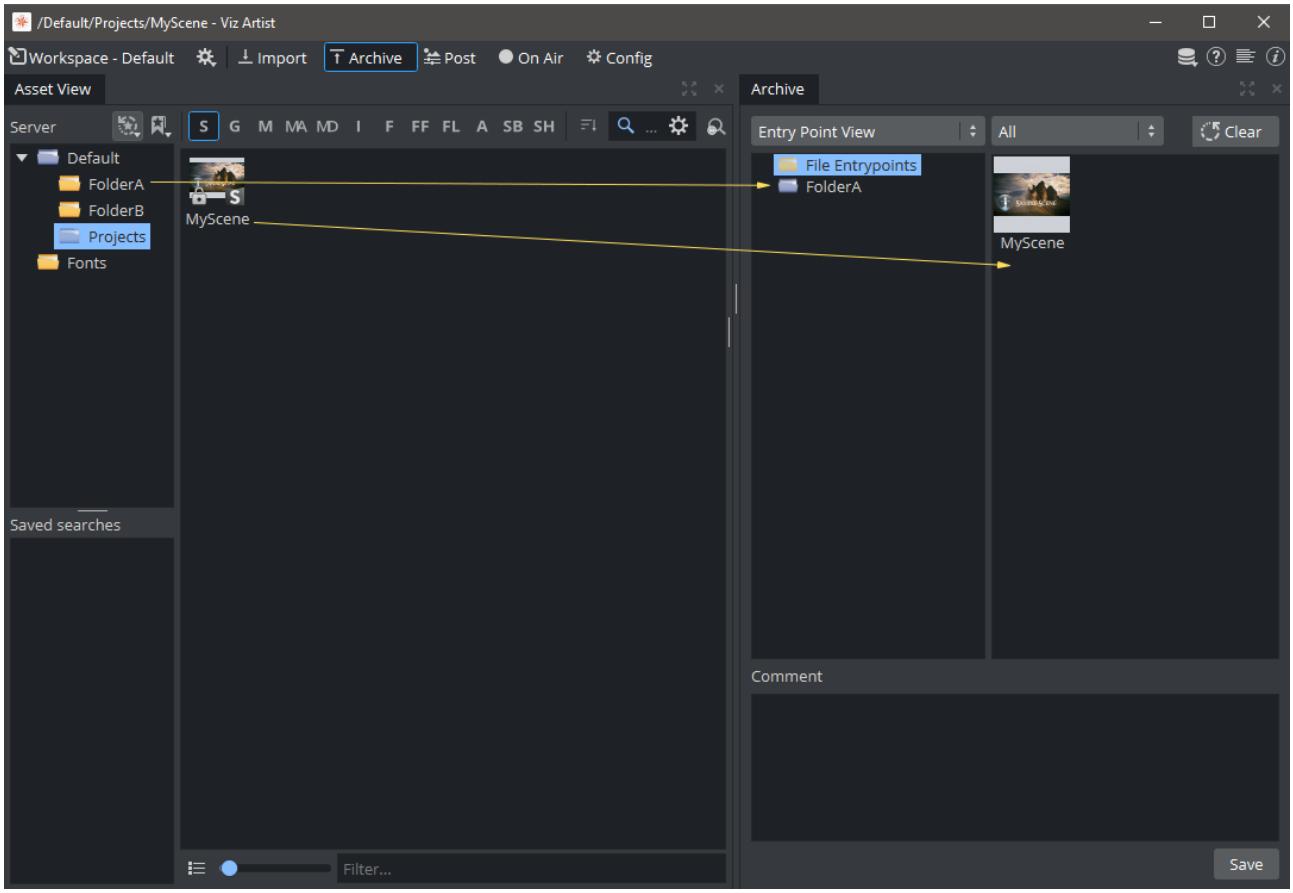
Copyright © 2026 Vizrt



It consists of an background image, a font, another image and a simple geometry (Note, this is not using the linked image pattern).

The user now creates an archive of his work to import this content later one to another machine. He drops the Scene into the Archive view and also decides to add the image pattern (which is not being used by the scene) by adding the FolderA to the archive and creates a *sample.via* **.**

Copyright © 2026 Vizrt



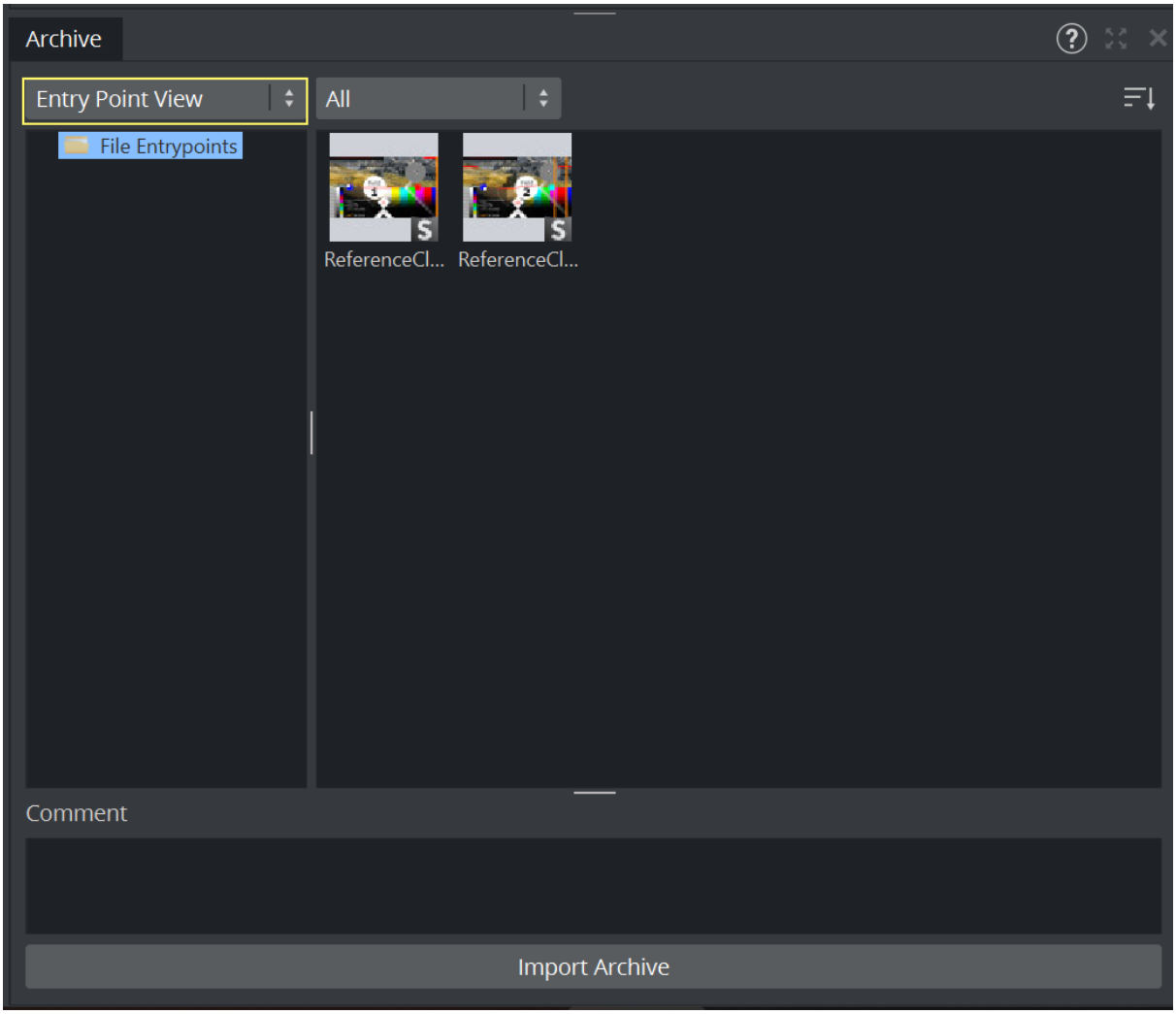
View Archive Content

An archive can be compared with a *.zip* file. To view the content of an archive, select the *.via* file in the import panel. If the *sample.via* is now opened on another machine, the content of the archive is displayed in the Archive pane on the bottom of the Import panel.

Use the drop-down menu in the preview panel to see the different aspects of an item's composition.

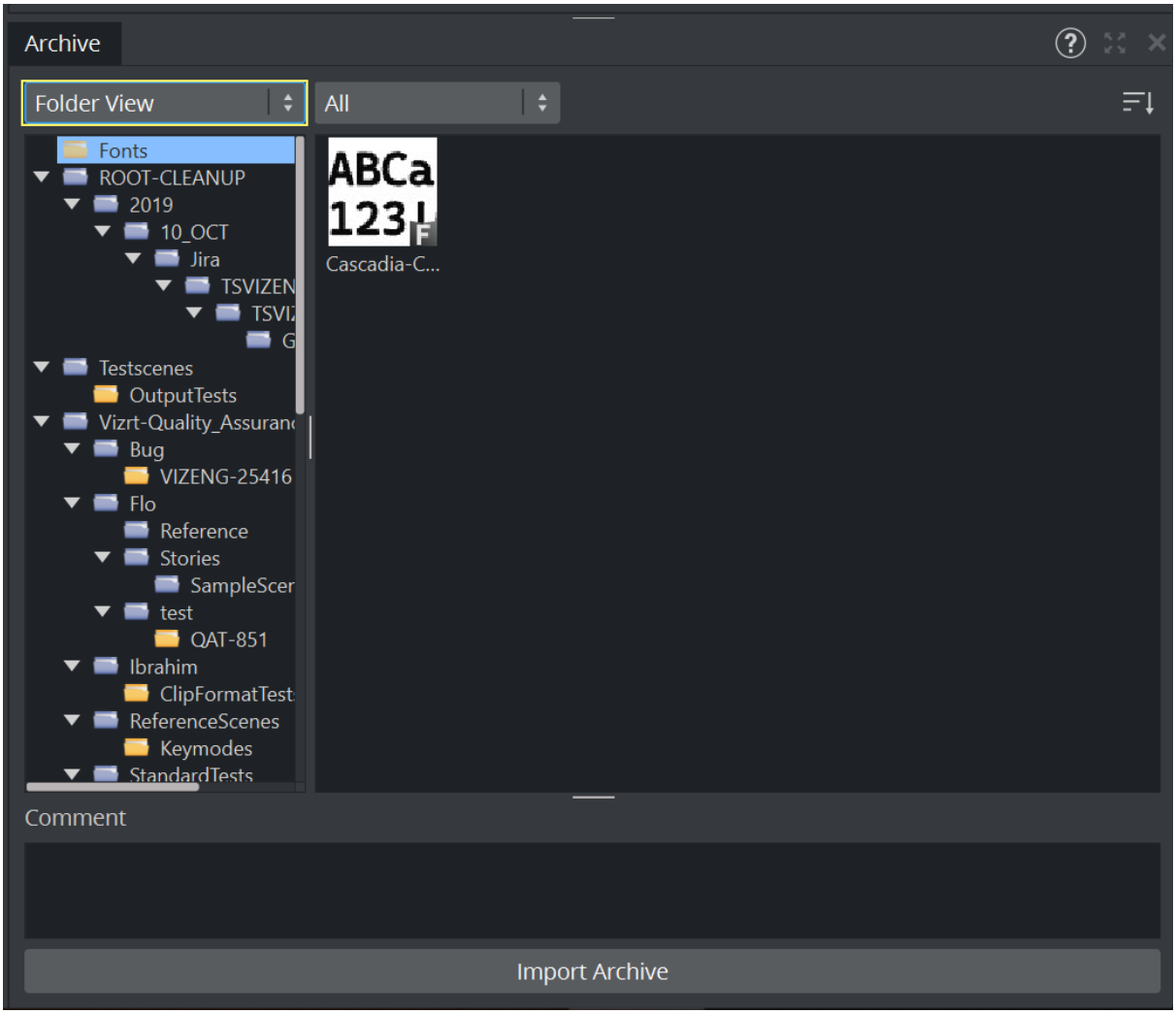
- **Entry Point View:** Shows all Entry Points. An entry point is created for each item or project or folder added to an archive by drag and drop. Note that the scene is displayed as a single Entry Points, the references and the related folders are not shown.

Copyright © 2026 Vizrt



- **Folder View:** The folder view shows what is included in the archive and is more close to the example of a *.zip* file. Note that the archive took all items linked to the scene and their folders. Fonts only shows the fonts being used by the scene.

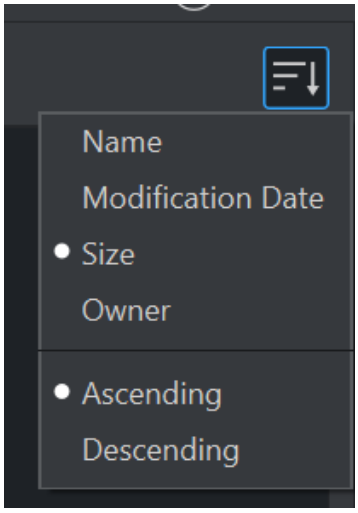
Copyright © 2026 Vizrt



Sorting Options

You can sort the content of an archive by several options.

Copyright © 2026 Vizrt



There are several options how this archive can be restored now.

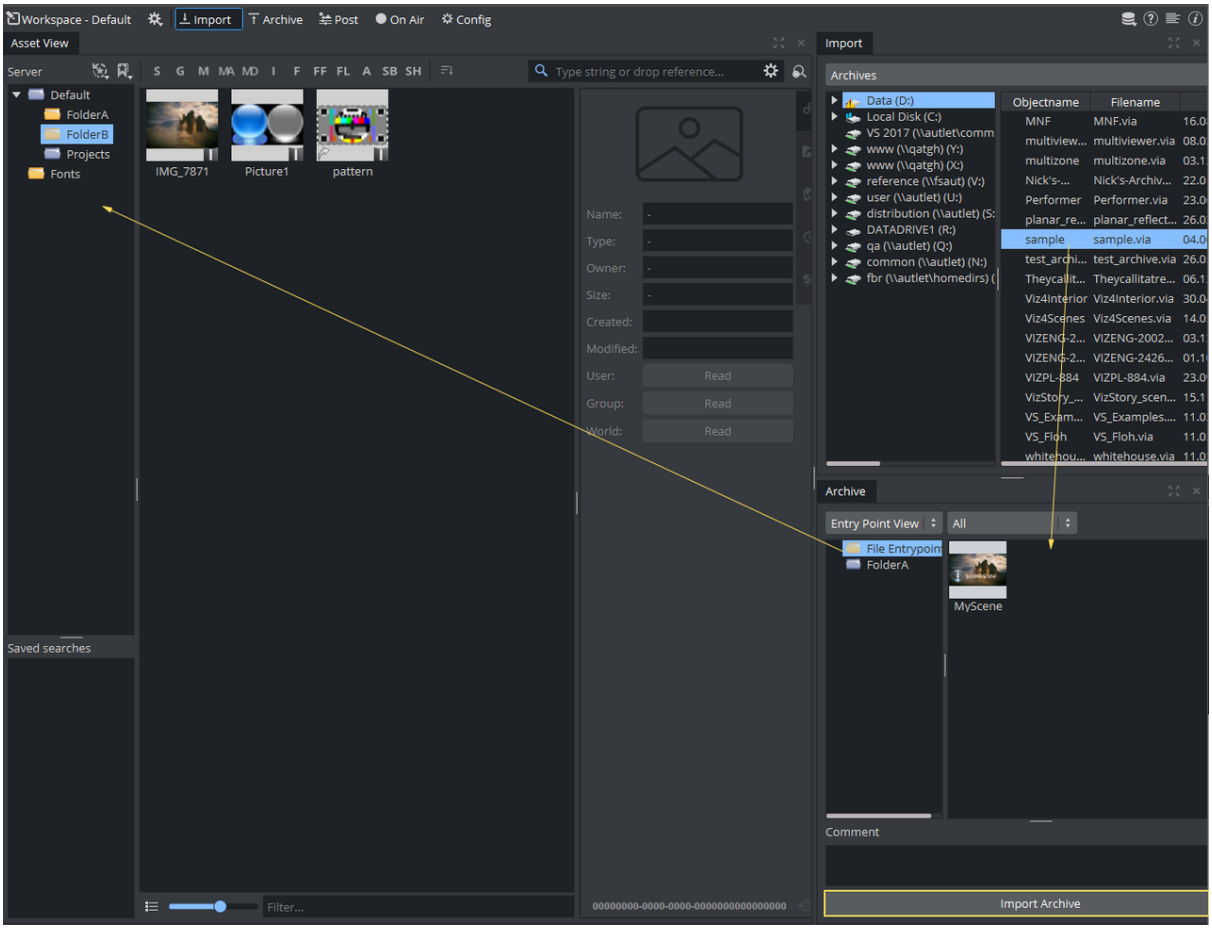
- Import the Archive "as is".
- Import an Archive to a Specific Location.
- Import Part of an Archive.

To Import an Archive "as is"

IMPORTANT! Using this method of import, the archive is imported to the root (/) of the GH while maintaining the path and names of the items in the archive.

1. In the Import panel, search for the Archive to import.
2. Select the required Archive.
3. Click on **Import Archive.**

Copyright © 2026 Vizrt



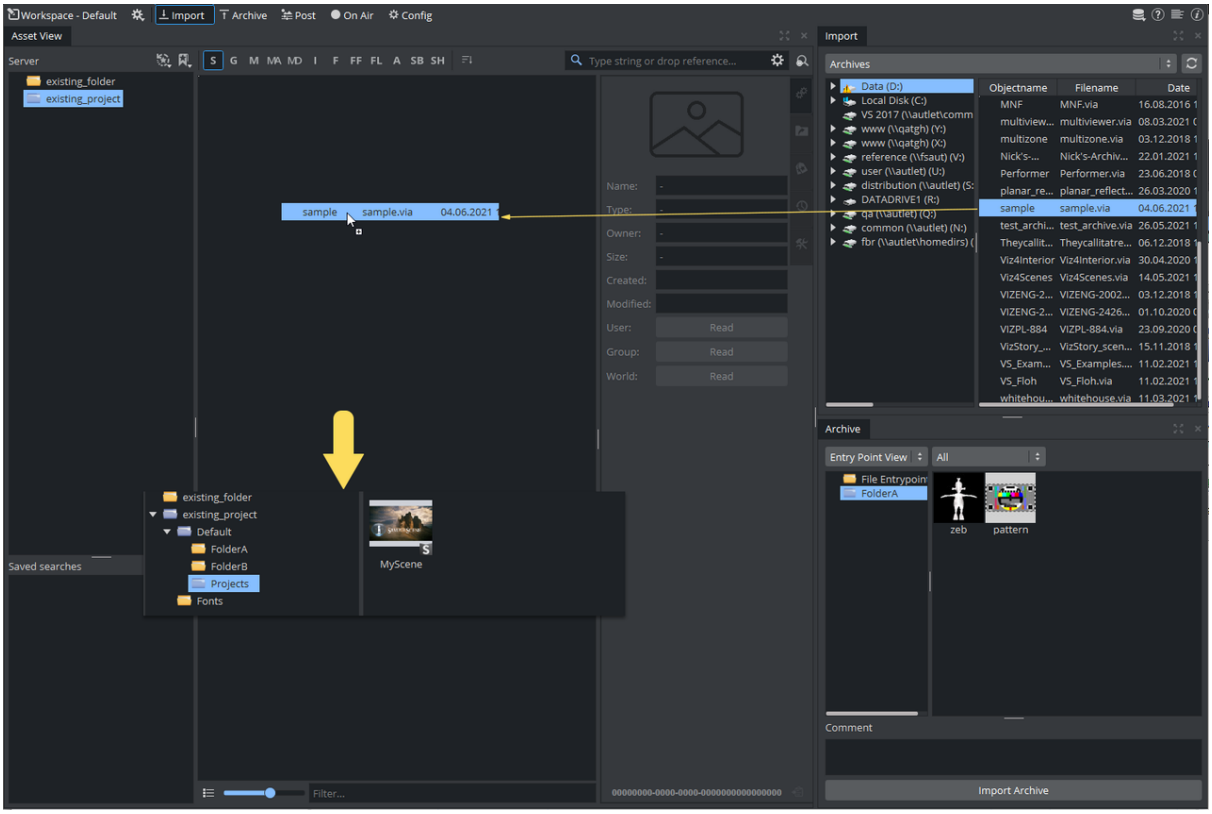
The archive has been restored exactly as it has been written. Folder A, Folder B, Projects and the Fonts have been restored as well as the items being used and the content of Folder A (added as entry point). Projects created for Playout usually follow a certain folder structure, therefore importing an archive as is is mainly used.

To Import an Archive to a Specific Location in a Graphic Hub

Instead of restoring the archive 1:1 and keep the original structure, you may want to add this structure into an existing folder structure. To do so:

1. Be sure the correct destination project has been selected (existing_project).
2. In the Import panel, search for the Archive to import.
3. Select the required Archive.
4. Drag the archives into their new location on the new Graphic Hub.

Copyright © 2026 Vizrt



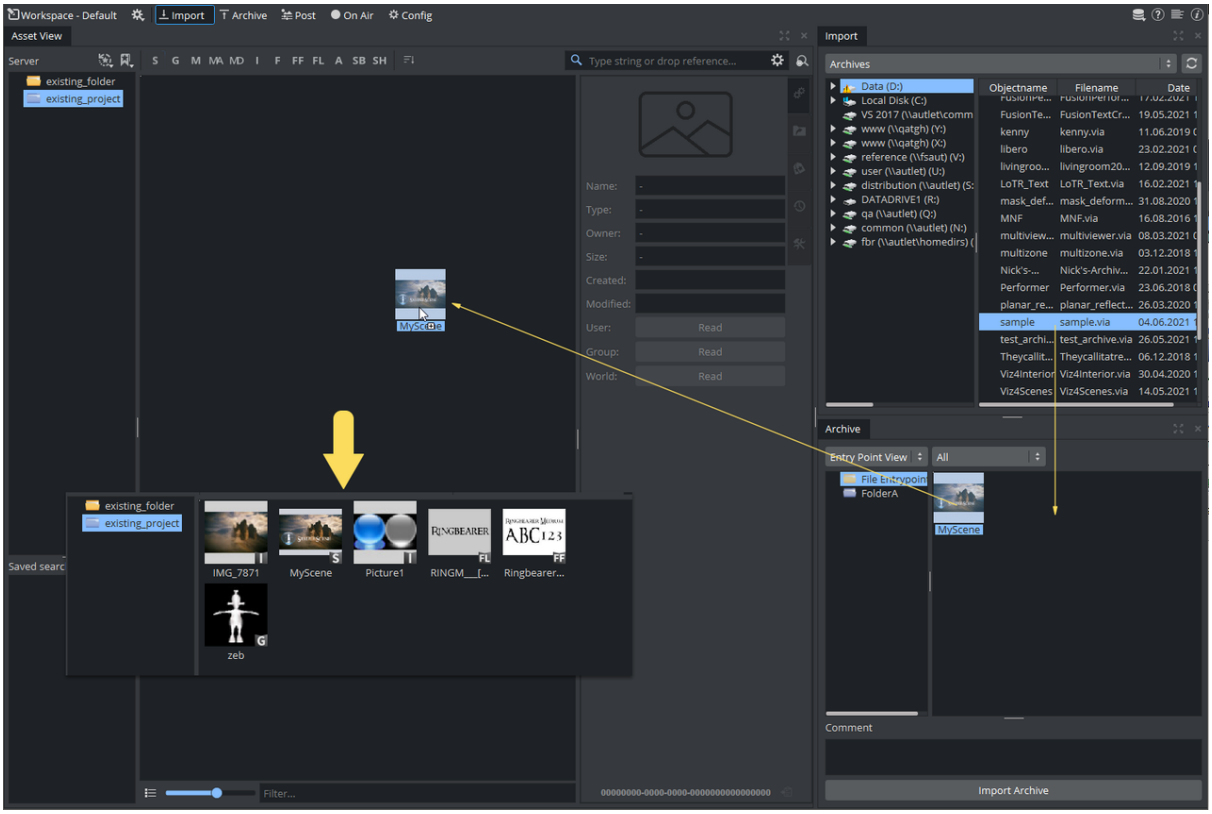
The content of the archive has been restored underneath the selected project. The original structure of the archive is now available as subfolders and subprojects within the selected destination folder. This operation can fail if the destination is a folder and not a project folder and the archive has projects inside. Projects can not be subitems of folders. Be sure that your destination points are always projects.

To Import Part of an Archive

If only single items of an archive are needed, they can be imported without importing the whole content.

1. In the Import directory tree, search for the Archive to import.
2. Select the required Archive in the right side panel.
3. From the top menu, select **Entry Points View**.
4. Select an import option:
   - To import all entry point files from the archive, drag the **File Entry Points** file folder and drop it in the left panel.
   - To import a folder and its contents from the archive, drag the required folder from the folder entry points and drop it in the left panel.
5. To import one or more files, drag and drop them into the left panel.

Copyright © 2026 Vizrt



In this example, the user decided to restore the scene only. He clicks on the scene and restores it into the project *existing_project*. The scene and all related references are imported into the destination project without and structure. All files end up in an flat representation.

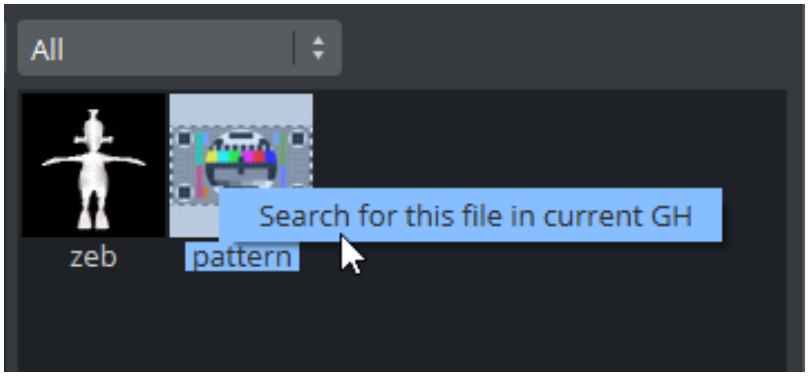
IMPORTANT! The archive is imported into the folder selected in the Graphic Hub tree. All reference files are included to guarantee full functionality. This method of import flattens the imported items. This means that the path or directory structure of the items in the imported archive are not kept. Use this method of import for importing parts of, or items of an archive.

Note: To import projects or folders, the same procedure applies, with the exception that you select **Folder View** from the pull-down menu, and drag and drop the required folder.

To Search for Imported Files

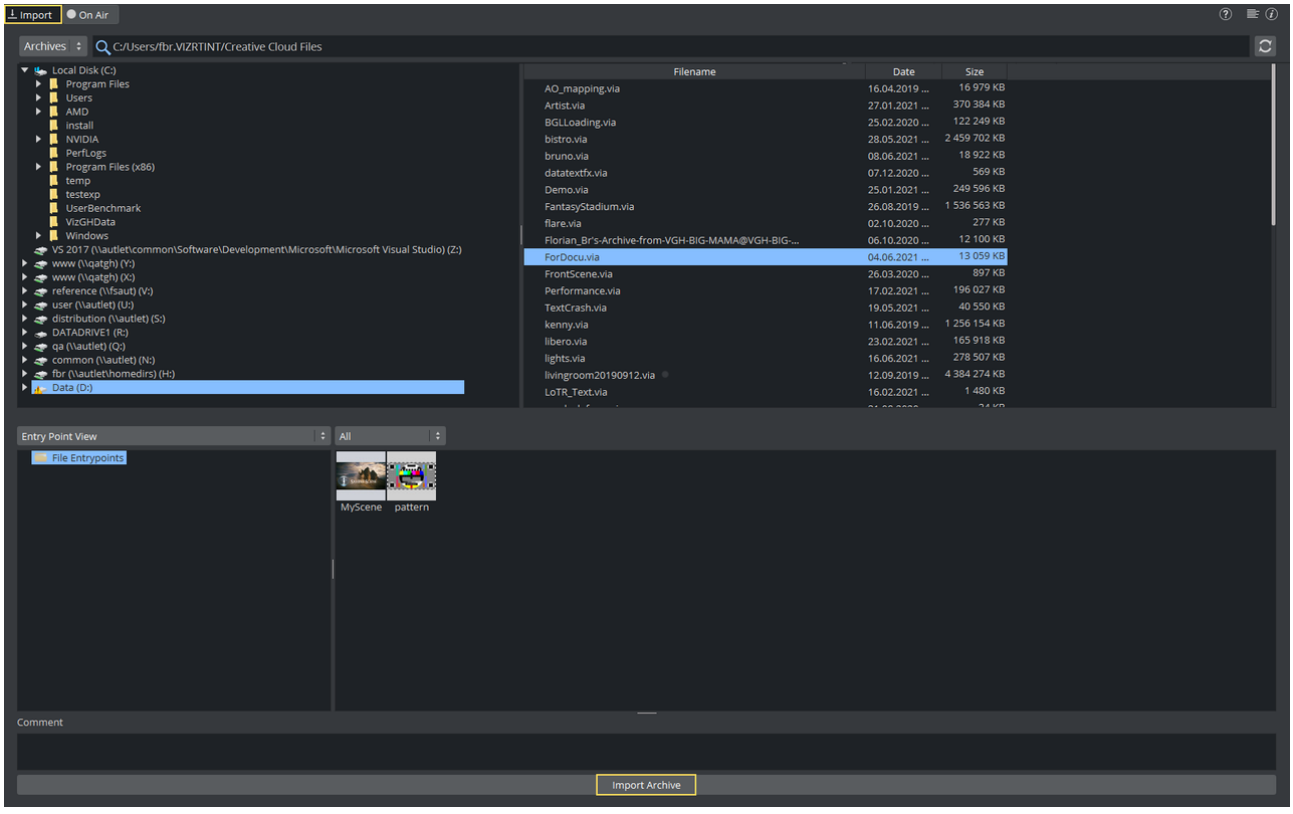
Search for files before or after the import for an Archive. Use this feature to see if similar or the same files are present in the system and to find the file's location after import.

Copyright © 2026 Vizrt



To Import an Archive in Viz Engine Mode

An archive can be imported in the On Air User Interface if Viz Artist is not installed, Viz Artist is not licensed or Graphic Hub Manager is unavailable. The functionality is limited to import an Archive only "as is".



- Click on the **Import** button in the On Air User Interface.
- Browse for the Archive to import.

Note: There is no drag and drop import function if Viz Engine is in On Air mode.

Copyright © 2026 Vizrt

- You can browse for the content of your Archive, but there is no functionality to only import parts of the Archive.
- To import the Archive to its original location, click the **Import Archive** button.

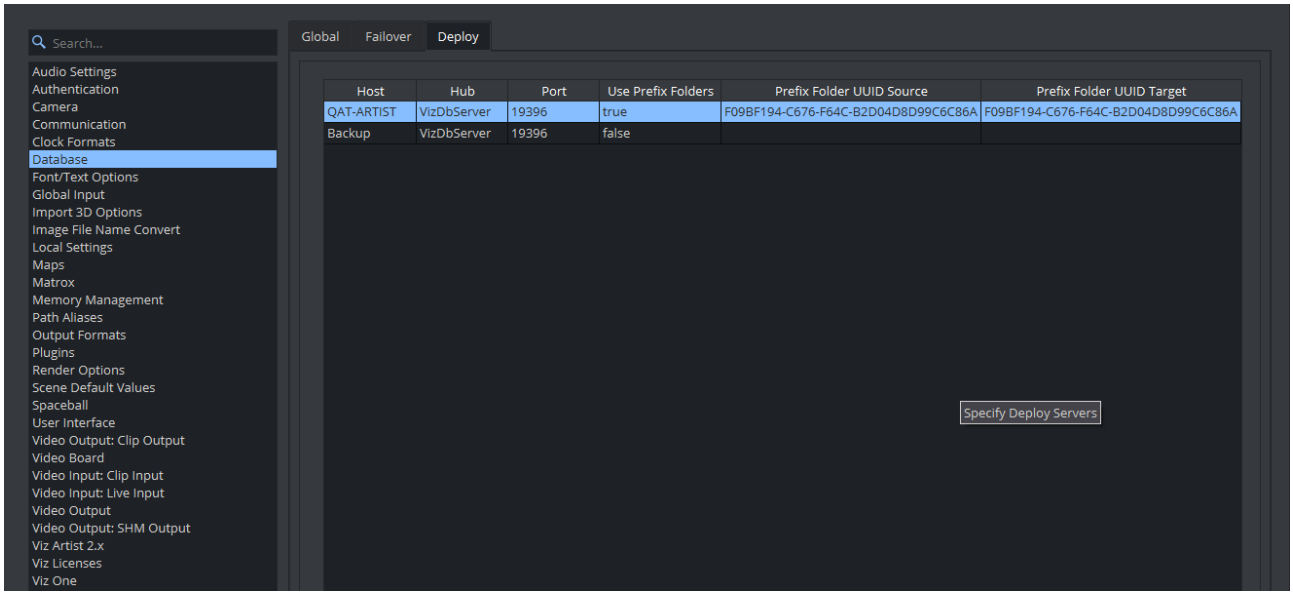
Once imported, the Scenes can be used for playout (scenes need to be preloaded first).

Copyright © 2026 Vizrt

18.4 Deploy Items

The Asset view makes it possible to copy items, with all their necessary data, and deploy them from one server to another. For example, if a Scene is deployed, all items referenced by the Scene (Images, Fonts, etc.), are also deployed.
To do so, you need to configure your destination(s) servers in the **Deploy** section page of the **Database** - **Configuring Viz** section:

18.4.1 To Configure Deploy Destinations



You can add one or more Graphic Hubs to deploy our items to. The following information is needed.

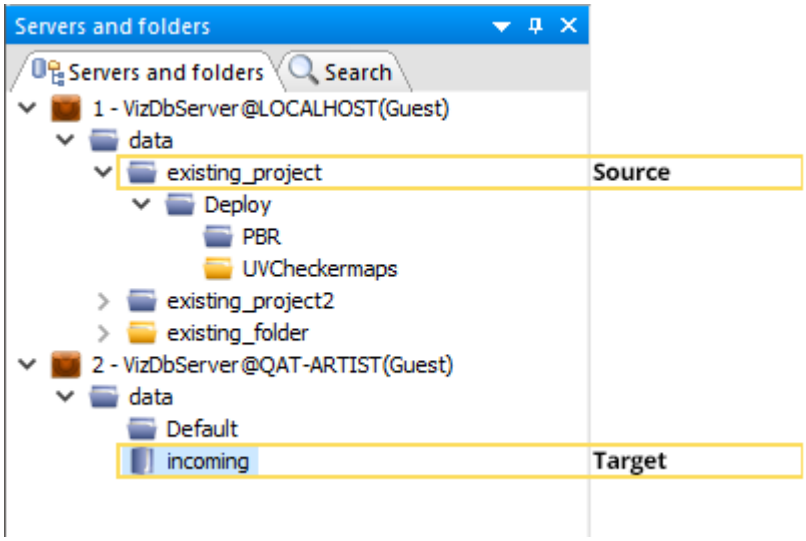
- **Host:** The naming service / hostname your Graphic Hub is running on.
- **Hub:** The Graphic Hub instance this naming service is pointing to, Default: *VizDbServer.*
- **Port:** The network port the naming service is listening to.
- **Use Prefix Folders:** If this option is set to *true*, the structure tree starting with the selected item is being rebuilt underneath the target folder, if the item is underneath the source UUID, otherwise the original structure is rebuilt. If use Prefix Folders is set to be *false,* the same structure as on the source server is kept.

Important: The destination server needs to have the same user account available than the source server.

Copyright © 2026 Vizrt

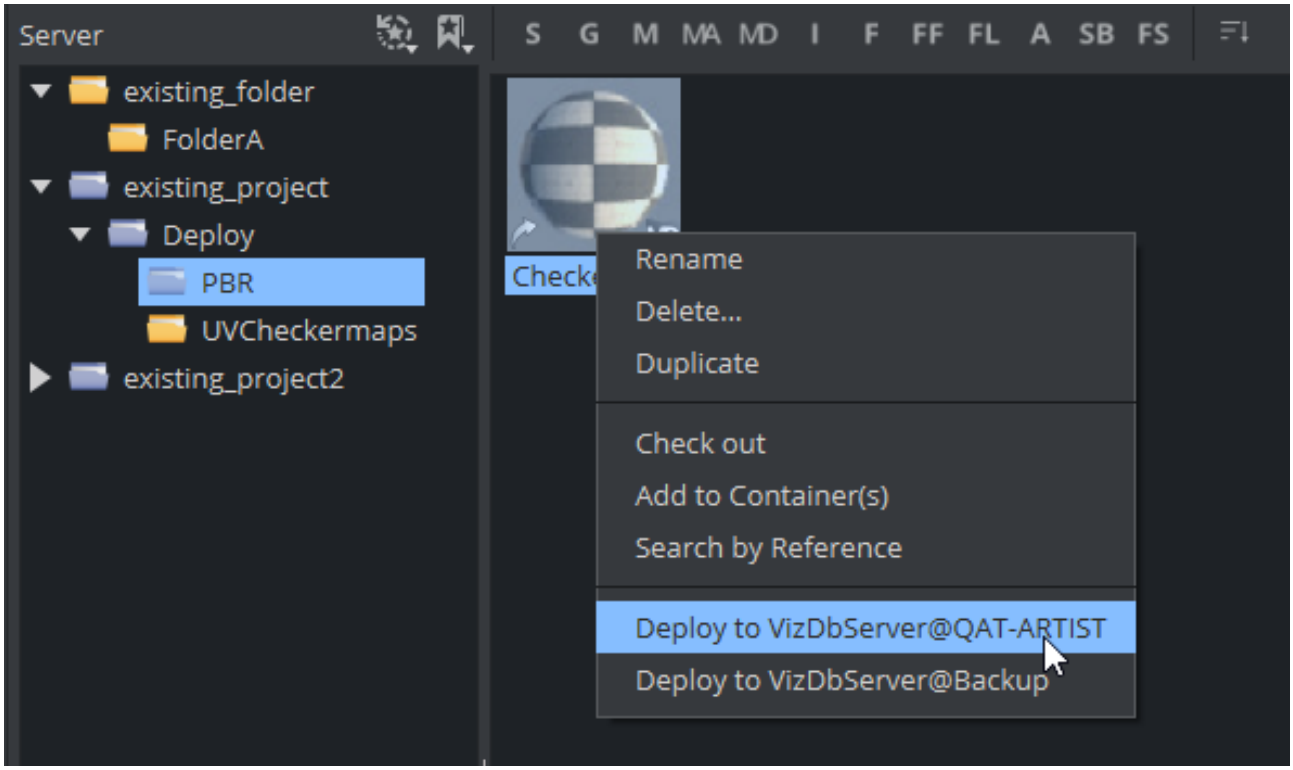
18.4.2 To Deploy Items

Example 1: Deploy with Source and Target UUID

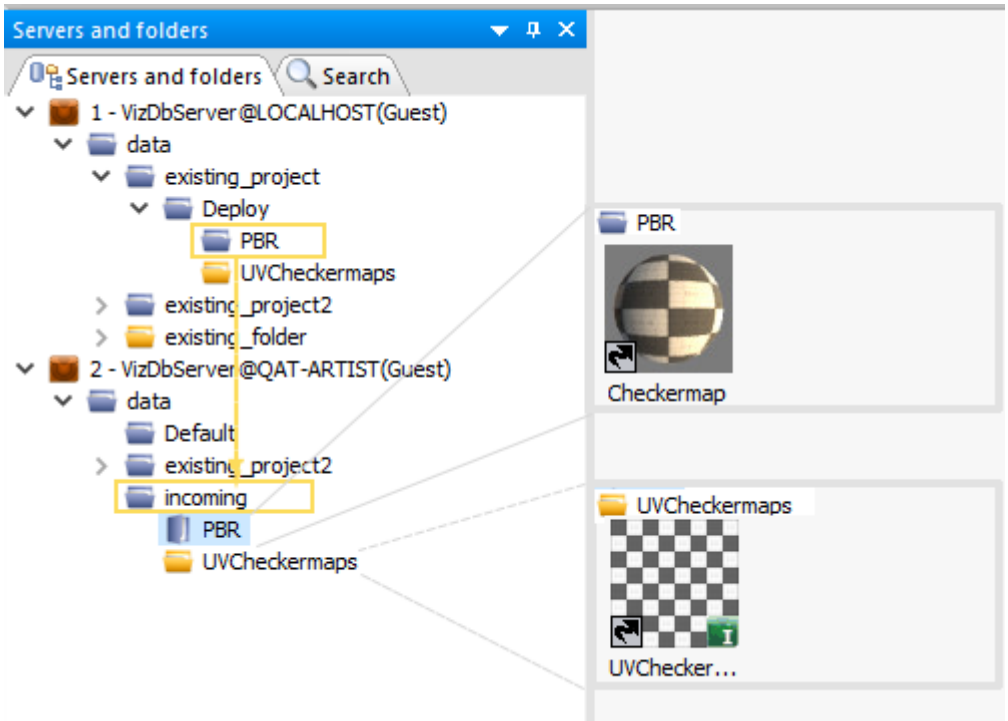


A PBR Material (from *data\existing_project\Deploy\PBR*) is deployed from Server A (**localhost**) to Server B (**QAT-ARTIST**). Project *existing_project* is set as **Source UUID** and *incoming* is set as **Target UUID** in deploy configuration. The Material itself has a reference to a texture in folder *UVCheckermaps* and has itself a folderlink to *UVCheckermaps.* From within Asset view, right click on the item you need to deploy and select **Deploy to Target**.

Copyright © 2026 Vizrt



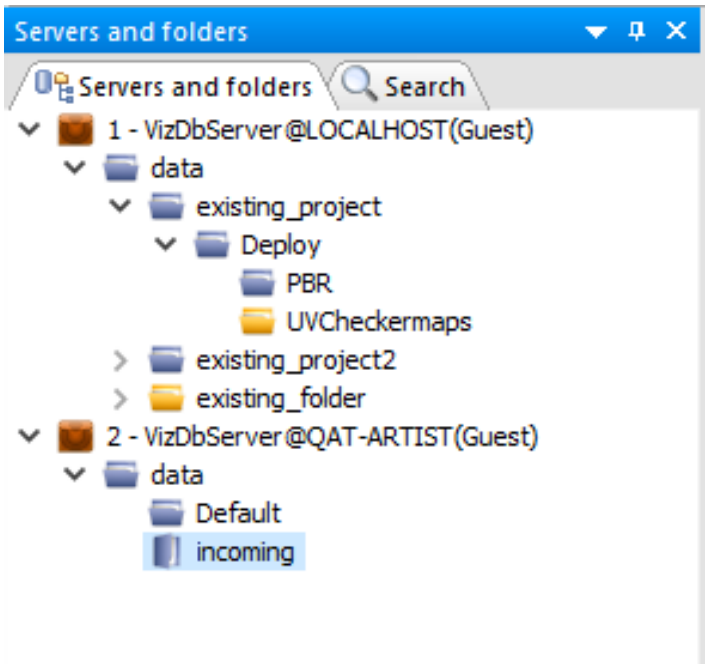
The PBR Material *Checkermap* is within the structure of the source UUID, therefore the folder structure will be copied to the target server:



Copyright © 2026 Vizrt

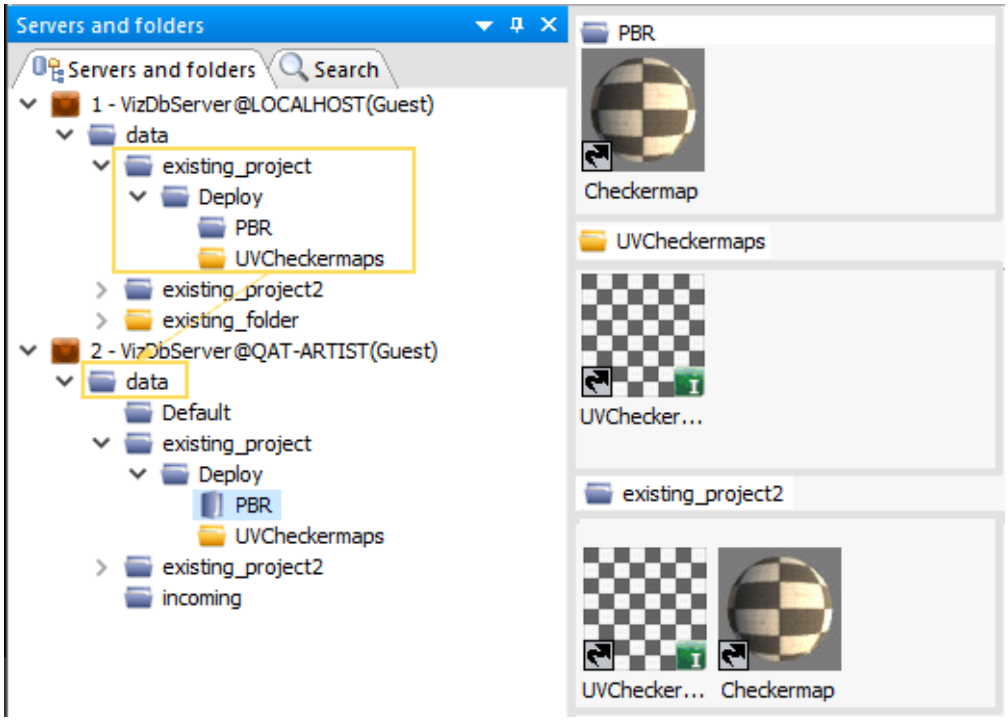
- The Material is deployed to the destination's folder "incoming" into the PBR folder.
- The folder "UVCheckermaps" with the reference to the Texture has been put underneath the target folder.
- The project with the folderlink "*existing_project2*" has been created outside the Target folder.

Example 2: Basic Deploy



A PBR Material is deployed from Server A (**localhost**) to Server B (**QAT-ARTIST**).

Copyright © 2026 Vizrt



The original structure is kept as it was in the source server. All relevant folders have been re-created underneath the *data* root project.

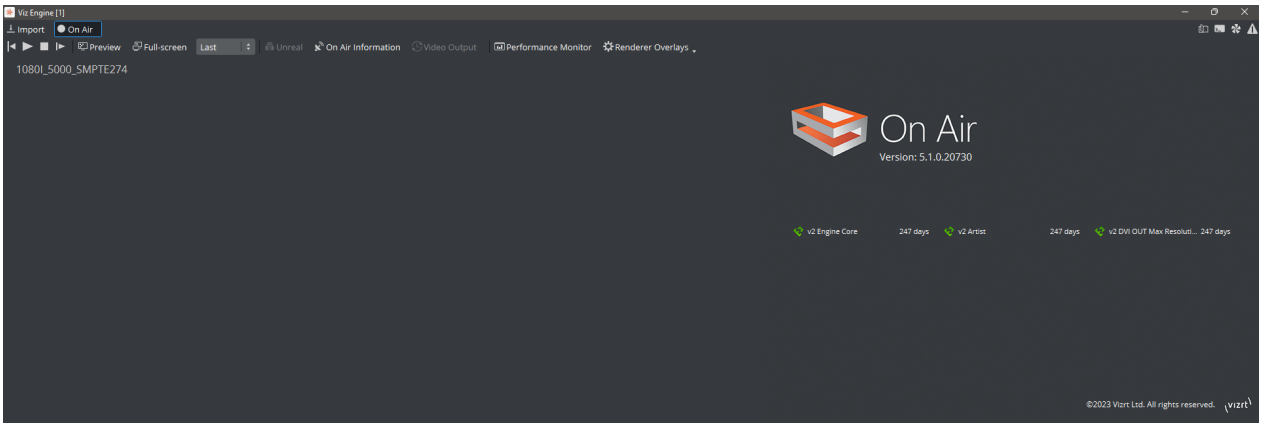
Tip: It is also possible to deploy multiple files by using multi-selection within the Asset view.

Copyright © 2026 Vizrt

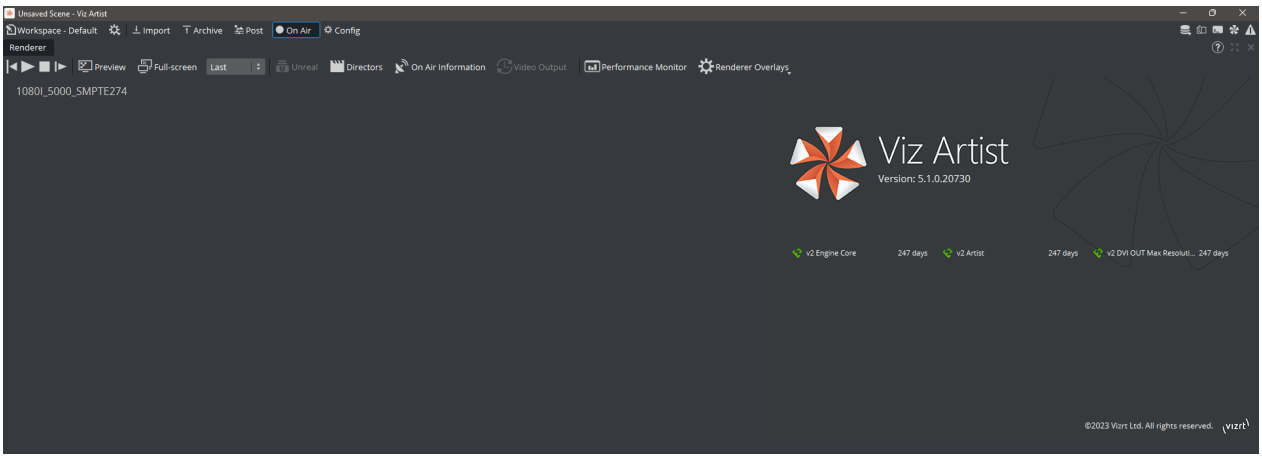
19 Viz Artist-On Air Mode

The On Air interface can vary, depending on the software and hardware configuration used. In Viz Artist, designers can click the On Air button on the main menu to switch Viz Artist from a modeling tool to a render engine. The application then waits for control commands. However, scene animations can also be rendered by the use of Control Buttons (top-left corner).

Viz Engine Only On Air Mode



Viz Artist On Air Mode



The top left of the On Air screen shows a set of Control Buttons, and the Director Control Panel and Performance Editor buttons. All license information is shown at the top right of the screen. Depending on the software and hardware settings, additional buttons and information is available. For example in design mode, the scene shows in an output window (lower right). All script events that are added to a scene or to single containers are executed in On Air mode. To learn more about scripting see Scripting and Program Examples.

Tip: To prevent unexpected data loss when switching to **On Air** mode, Viz Artist automatically creates a backup of the current On Air scene with the extension *_onair_backup*. If for example the scene is called *ABC*, the created backup is called *ABC_onair_backup.*

This section contains information on the following topics:

- Viz Artist On Air Control Buttons
- Viz Artist On Air Director Control Panel
- Viz Engine Performance
- On Air Information
- Unreal Control Panel
- Viz Artist Information
- Tracking Timing Monitor

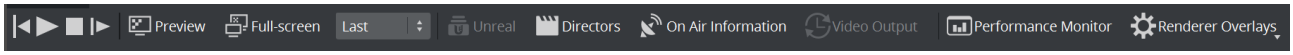
See Also

- On Air Shortcuts

Copyright © 2026 Vizrt

19.1 Viz Artist On Air Control Buttons

This section contains information on the Controls Buttons, which include the Play and the On Air buttons.



19.1.1 Animation Controls

- **Back:** Jumps to the beginning of the animation in the scene.
- **Play:** Starts the animation of the scene.
- **Stop:** Stops the animation of the scene.
- **Continue:** Continues the animation after it stopped at a stop point.
- **Clapper Board:** Shows or hides the Director Control Panel window.

19.1.2 Preview Controls

Shows or hides the VGA Preview window (only available on machines with video cards). The keyboard shortcut for activating or deactivating this window is **SHIFT + BACKSPACE**. The state of the VGA Preview window depends on how the **Render preview default setting** in the **Video Output** section of Viz Config is configured:

- **Show Preview:** Switches the preview render window on/off.
- **Full-Screen Preview:** Uses a full-screen window for preview.
- **Full-Screen Output:** Switches the display to use for full-screen previews.

Note: Preview needs to be licensed.

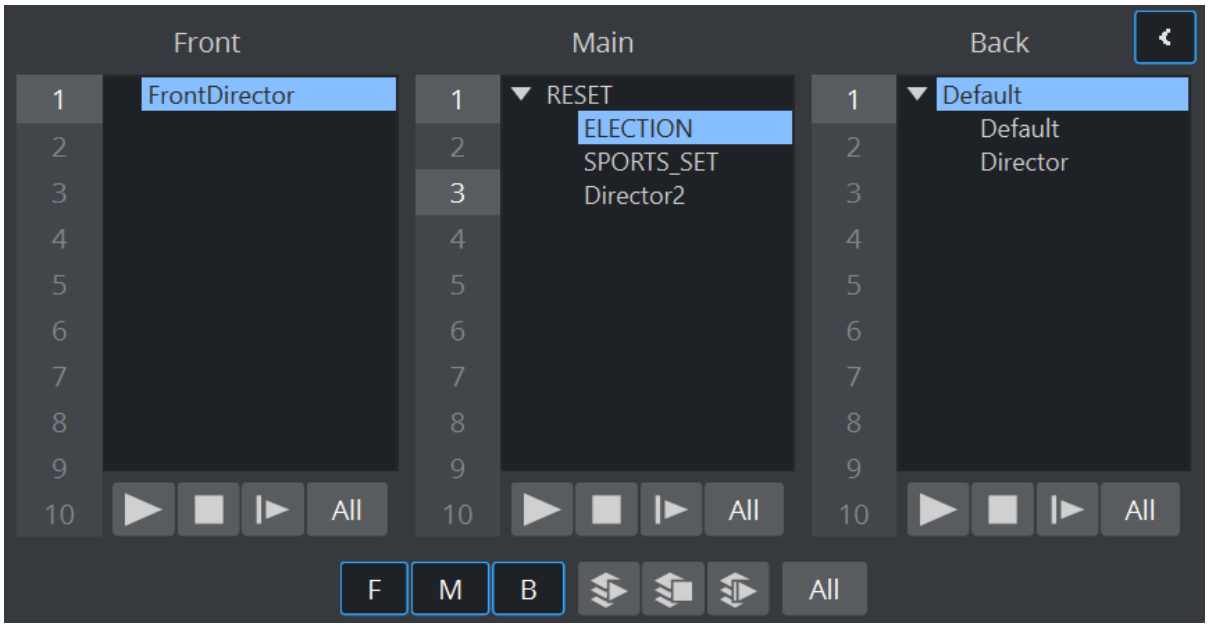
19.1.3 Tools

- **Unreal Engine Controls:** Shows or hides the Unreal Control Buttons.
- **Performance Editor:** Shows or hides the Performance Editor.
- **System Information:** Shows or hides the On Air Information window.
- **Video Output:** Shows the editor for Sync options on SDI and IP boards.

Copyright © 2026 Vizrt

19.2 Viz Artist On Air Director Control Panel

The Director Control Panel window can be used to select and animate one, multiple or all directors in the front, main or back layer. In addition, it can be used to set slots and to animate a combination of director(s).



Click the Directors button, when in On Air mode, to open the Director Control Panel window. See Director Control Panel for a detailed description of the Director Control Panel.

Copyright © 2026 Vizrt

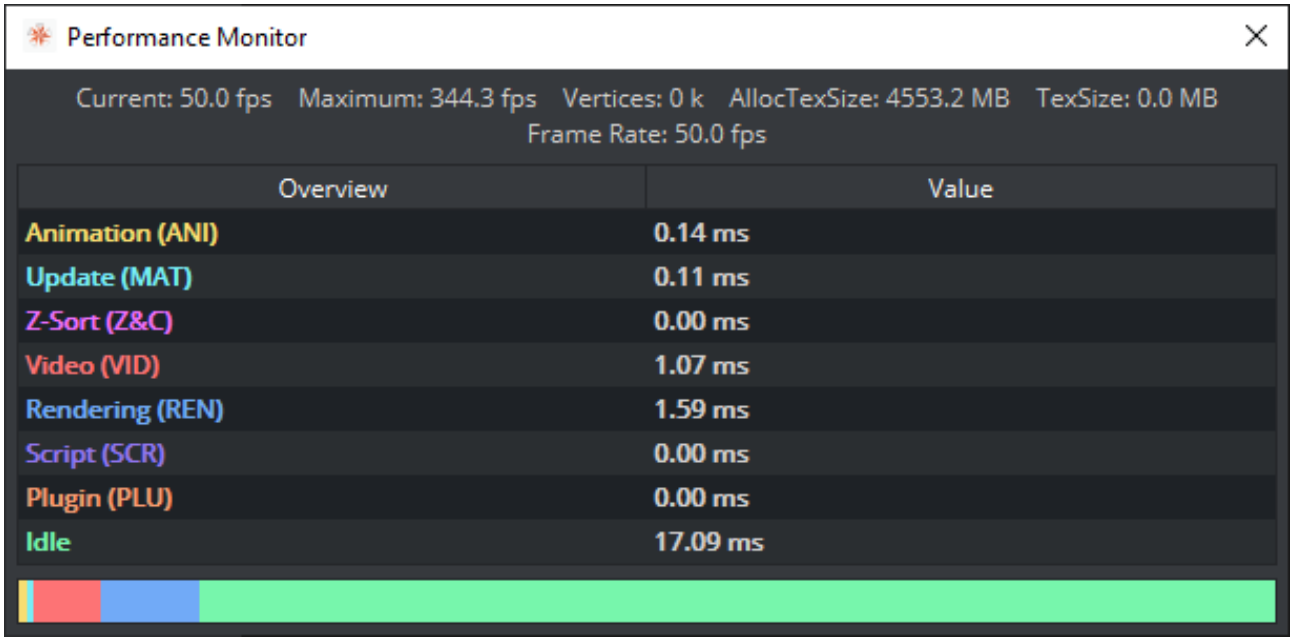
19.3 Viz Engine Performance

Analyzing the performance of Viz Artist/ Viz Engine can be done with:

- **Performance Monitor:** Closely monitors a range of parameters for analyzing real-time performance.
- **Performance Analyzer:** Monitors key performance and camera parameters, as a head-up display in the renderer window.
- **Performance Tracing:** Captures a detailed trace report over multiple frames.

19.3.1 Performance Monitor

To open the Performance monitor click on the Performance Monitor toolbar button.



The performance bar gives an idea of the current scene rendering performance (frames per second).

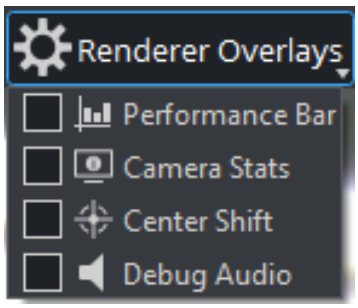
- **Current (CUR):** Shows how many frames per second the scene renders in On Air mode. The number should be above 50 (PAL) or 60 (NTSC), according to the rate that has been specified in the **Output Format** section of the Viz Engine Administrator Guide.
- **Maximum (MAX):** Shows how many frames per second the scene can render at without waiting for vertical retrace. The higher the maximum value, the more performance is left. If the maximum value is reduced to below `50` or `60`, the scene is not rendering in real-time.
- **Vertices (VER):** Shows the number of vectors in the scene.
- **AllocTexSize (TET):** Shows the total allocated size of texture memory.
- **TexSize (TEC):** Shows the size of the currently used texture memory.
- **Animation (ANI):** Shows how many microseconds all active directors and animation channels take. This indicator is linked to the yellow bar.

Copyright © 2026 Vizrt

- **Update (MAT):** Transforms each container in the scene into world coordinate space. This indicator is linked to the cyan bar.
- **Z-Sort (Z&C):** Refers to Z-sort and Culling, and sorts all containers for correct transparency drawing and determines if containers are visible in the current camera view. This indicator is linked to the pink bar.
- **Video (VID):** Shows how many microseconds video input (live video texture) and video output take. De-interlaced video inputs take longer time than progressive and interlaced. The only way to improve this value is to use a faster system. This indicator is linked to the red bar.
- **Rendering (REN):** Shows how many microseconds it takes to render all objects on the screen. A faster graphics card improves this value. This indicator is linked to the blue bar.
- **Script (SCR):** Shows the consumed time in microseconds from all active scripts. This indicator is linked to the dark green bar.
- **Plugin (PLU):** Indicates how much time in microseconds all active plug-ins spend in each render cycle. This indicator is linked to the orange bar.
- **Idle:** Shows available resources in microseconds the renderer has available. This indicator is linked to the light green bar.

19.3.2 Renderer Overlays

To select which performance data to show in the renderer window, click on the **Render Overlays** tool button then select the options to show:



- **Performance Bar:** Shows the a detailed Performance analyzer.
- **Camera Stats:** Enables the head-up display (HUD) showing the following parameters in the renderer view:
  - **Camera 1-n:** Shows the currently selected camera.
  - **Position:** Shows the camera's X, Y and Z position.
  - **Pan/Tilt/Twist:** Shows the camera's pan, tilt and twist parameters.
  - **FovX/FovY:** Shows the camera's field of view (FOV) for the horizontal (X) and vertical (Y) plane.
  - **Center Shift:** Shows the X and Y position of the camera's center shift.
- **Center Shift:** Shows the center shift marker as a cross hair in the renderer.
- **Debug Audio:** Turns on/off a visual representation of the audio mixing.

Performance Bar (Classic)

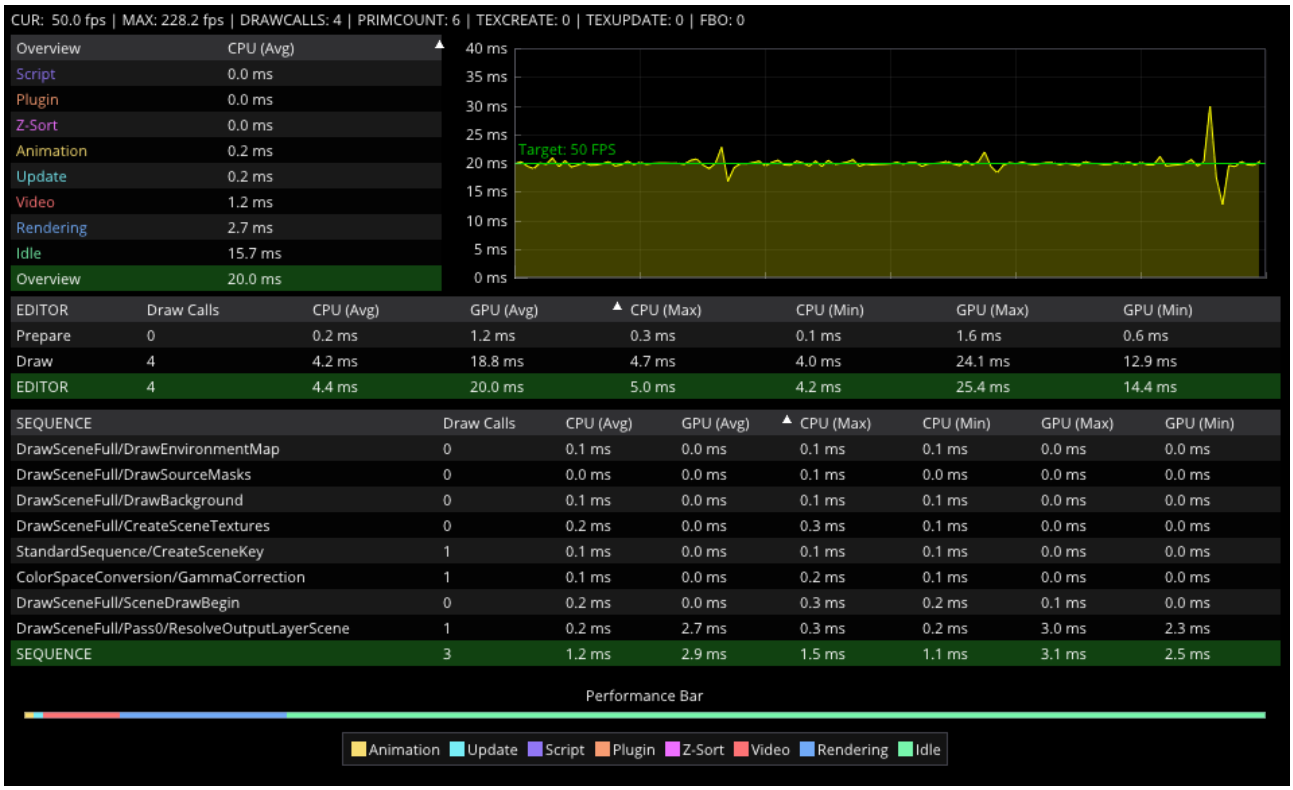


Copyright © 2026 Vizrt

The performance bar gives an idea of the current scene rendering performance (frames per second).

The values shown are the same as described in the Performance Monitor section above.

Performance Bar and Analyzer (Viz Engine Render Pipeline)



The Viz Engine Render Pipeline shows more detailed information about:

- Current and Maximum Framerate.
- Segmented render time and historic data.
- Timing information for all used Sequence pipelines.
- Performance Bar.

Performance Tracing

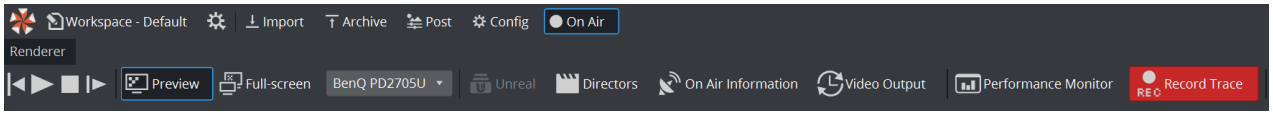
Performance Tracing records exactly what Viz Engine does in every frame, for example rendering, scripts, plug-ins and video. The recording is saved as a trace file (`.pftrace inside C:\ProgramData\vizrt\VizEngine\trace`) in the open Perfetto trace format.

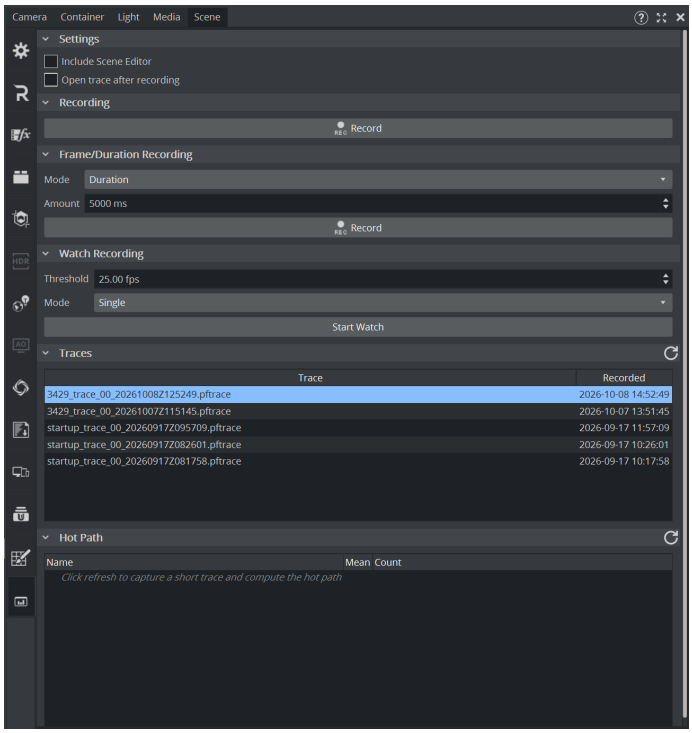
If you see performance problems such as dropped frames, stutter or a low frame rate, record a trace while the problem happens and attach the file to your support ticket.

Copyright © 2026 Vizrt

Where to Find Tracing

- **Record Trace toolbar button:** The record button in the On Air toolbar starts and stops a recording with one click. The button turns red while recording.
- **Performance Monitor > Tracing tab:** Open the Performance Monitor and select the **Tracing** tab. All tracing options are here.
- **Scene Settings > Performance:** The same tracing options are also in the Scene Settings panel (not available in Engine mode).





Recording a Trace

1. Prepare the system so the problem can happen, for example by loading the affected scene and taking it On Air.
2. Click **Record Trace** in the On Air toolbar, or **Record** in the Tracing tab.
3. Reproduce the problem. Keep the recording short: a few seconds that cover the problem are enough.
4. Click the button again to stop the recording. The trace file is saved automatically.
5. Open **Performance Monitor > Tracing** and find the new trace at the top of the **Traces** list.
6. Drag the trace from the list straight into your support ticket, an email or a folder.

Tip: *If the problem is hard to reproduce on demand, use* ***Watch Recording*** *(see below). It waits for the frame rate to drop and saves a trace for you automatically.*

Copyright © 2026 Vizrt

Recording Options

Settings

- **Include Scene Editor:** Also records what the scene editor does, not only the renderer. Turn this on if the problem happens while you work in Viz Artist rather than On Air.
- **Open trace after recording:** Opens each finished recording in the trace viewer straight away. Off by default: new traces only appear in the Traces list.

Recording

- **Record / Stop:** Starts a recording and stops it when you click again. This is the same as the Record Trace toolbar button.

Frame/Duration Recording

- **Mode:** Records either for a set **Duration** (milliseconds) or for a set number of **Frames**.
- **Amount:** The duration (default 5000 ms) or the number of frames (default 300).
- **Record:** Starts the recording. It stops by itself when the duration or frame count is reached.

Watch Recording

Watch Recording runs in the background and saves a trace only when the frame rate falls below a threshold. Use it for problems that happen only now and then.

- **Threshold:** The frame rate (fps) that triggers a trace. Set it to your output rate (for example 50 or 59.94), or slightly below it.
- **Mode:**
    - **Single:** Saves one trace at the first frame-rate drop, then stops watching.
    - **Continuous:** Keeps watching and saves a trace at every drop.
- **Start Watch / Stop Watch:** Starts or stops watching.

Note: *Only one kind of recording can run at a time. While a recording or watch is running, the other record buttons are unavailable.*

Traces

The Traces list shows all recorded traces, newest first.

- **Trace:** The file name. Hover over it to see the full path on disk.
- **Recorded:** When the trace was recorded, in local time. The time in the file name is UTC.
- **Double-click:** Opens the trace in the trace viewer.
- **Drag and drop:** Drag a trace into a ticket, an email or a folder to copy the file.
- **Refresh:** Reloads the list.

Hot Path

Hot Path gives a quick view of where time is spent right now, without opening a trace file.

- **Refresh:** Records a short 3-second trace and shows it as a call tree. The time of that capture is shown next to the button.
- The most expensive chain of calls (the hot path) is expanded and highlighted.
- **Name:** The track or function.
- **Mean:** Average time per call.

Copyright © 2026 Vizrt

- **Count:** Number of calls during the capture.
- Hover over an entry to see its total time and self time, as a time value and as a share of the track.

Copyright © 2026 Vizrt

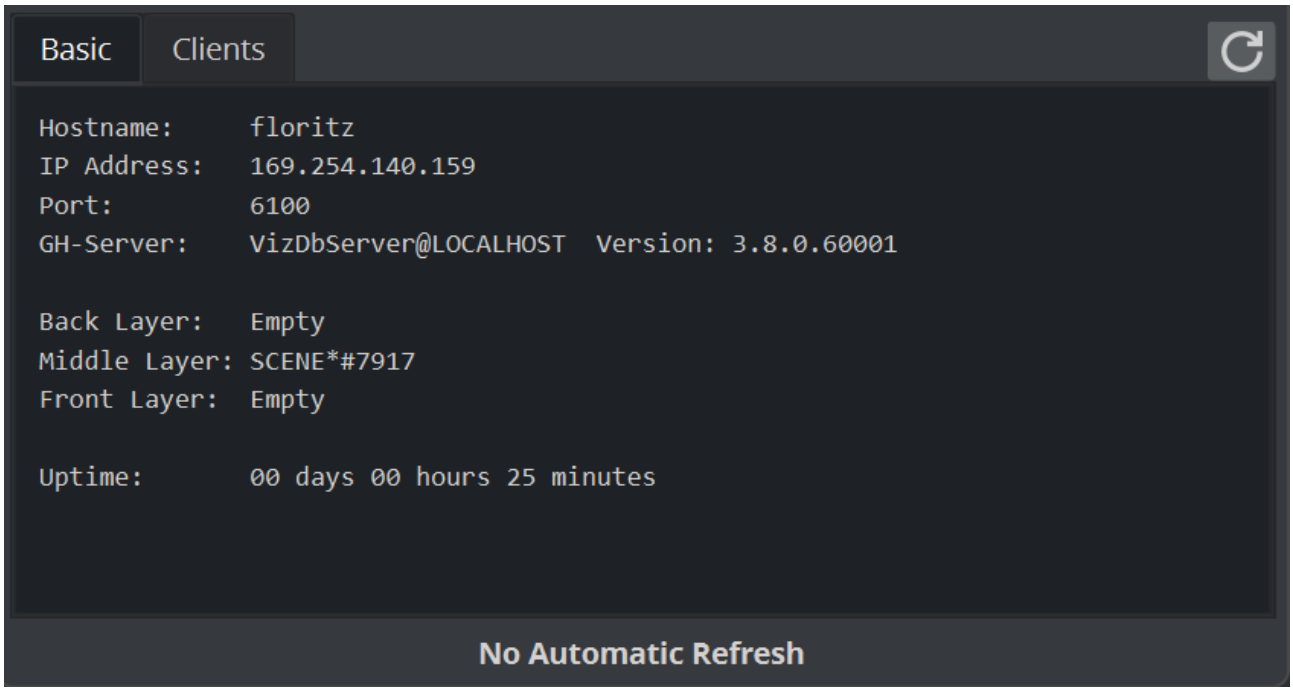
19.4 On Air Information

The On Air Information shows details about the currently running Viz Engine, and shows all connected clients with the IP address, host name and Viz Port. To show or hide the On Air Information panel, click On Air Information.

> **Note:** Polling for On Air information can decrease the performance. For information on how to adjust the On Air Update Interval see User Interface in the Viz Engine Administrator Guide.

19.4.1 Basic Tab

The **Basic** tab displays some basic information, such as name and IP address of the current host, database connection, and the current scene.

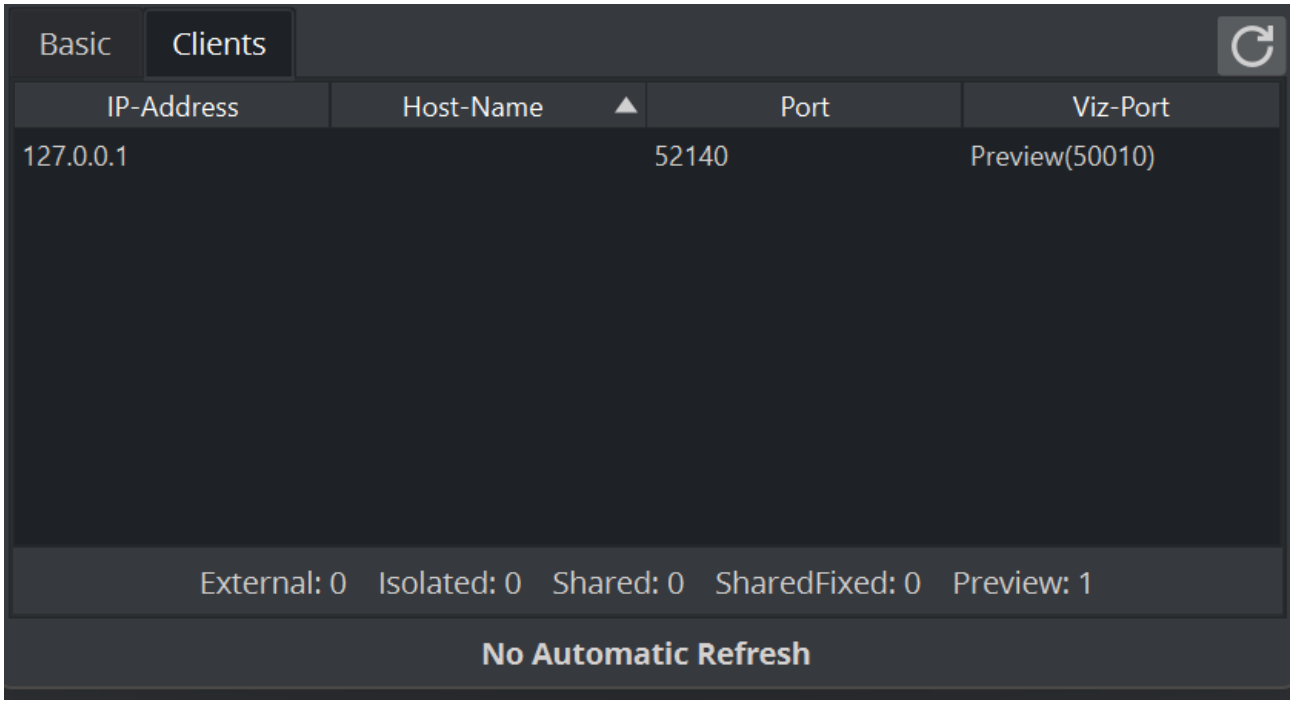


- **Refresh button:** Refreshes the status information.
- **Hostname:** Shows the name External Control Applications can use to communicate with Viz Artist.
- **IP Address:** Shows the IP address External Control Applications can use to communicate with Viz Artist.
- **Port:** Shows the port Viz Artist is using. Default port is `6100` (see the Viz Engine Administrator Guide).
- **GH-Server:** Shows the Graphic Hub server Viz Artist is connected to.
- **Back Layer:** Shows the name of the scene that is defined to run in the background layer.
- **Middle Layer:** Shows the name of the scene that is defined to run in the middle layer.
- **Front Layer:** Shows the name of the scene that is defined to run in the foreground layer.
- **Uptime:** Shows the time elapsed since Viz Artist/Engine was started.

Copyright © 2026 Vizrt

19.4.2 Clients Tab

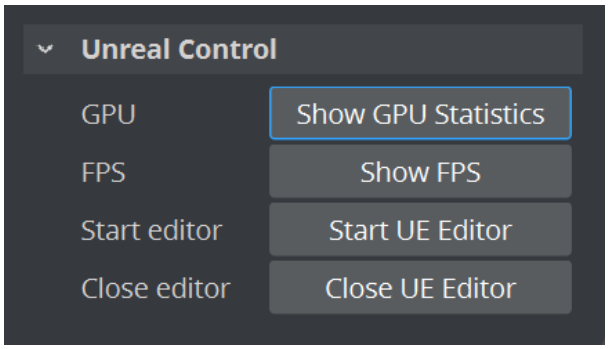
In the **Clients** Tab, all connected clients are shown, with the IP address, Hostname and Viz Port.



19.5 Unreal Control Panel

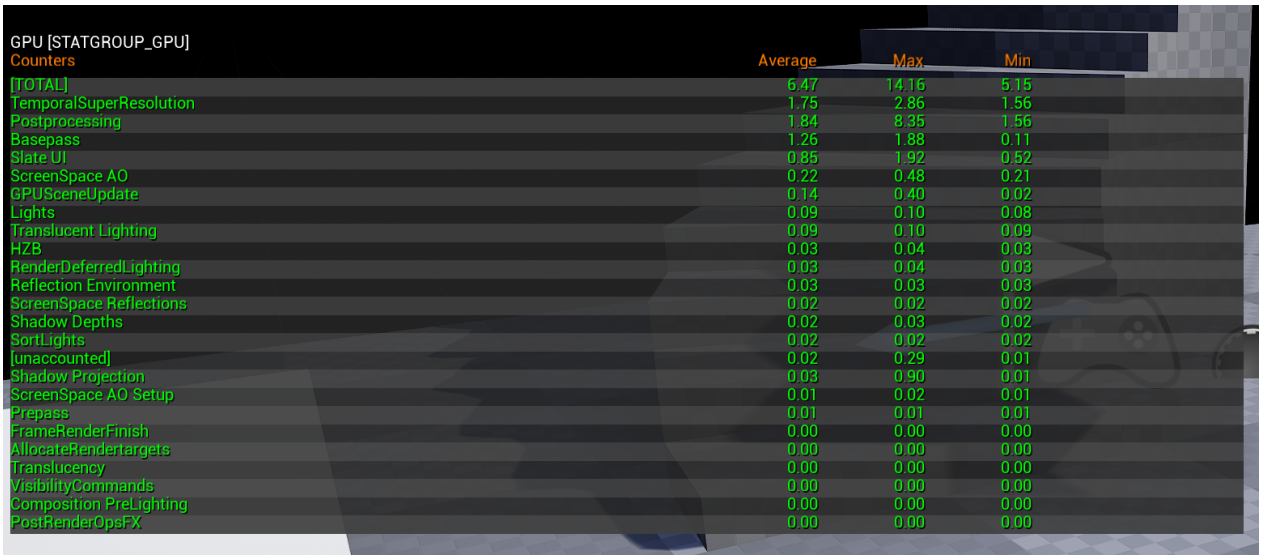
The On Air window offers a panel to control Unreal Engine. This allows for a better control of both Engines from within a single place.

Open the panel using the button . This button is only active, if AUX rendering is turned on.



- **Show GPU Statistics:** Enables a performance counter overlay on the Unreal output. This is for profiling purposes.
- **Show FPS:** Displays a frame counter.
- **Start UE Editor:** Starts the Unreal Editor, if a *.uproject* file is available.
- **Close UE Editor:** Closes the Unreal Editor.

Unreal Editor Showing GPU Statistics



Copyright © 2026 Vizrt

19.6 Viz Artist Information



The Viz Artist/Viz Engine Information panel opens from the **Artist Info** button in the upper right corner .

The panel show the current installed versions, the build date and License and Legal notices.

20 Transition Logic

Transition Logic is first and foremost a method of designing a graphics package that allows the designer to maintain the look and feel of the graphics while also allowing anyone else, such as journalists or producers, to add graphics items to a rundown without the need for any kind of technical knowledge of how the system works.

Transition Logic allows the independent control of any number of graphics layers, providing a code-free and design-based method for building graphics that gracefully animates in and out, and transitions from one to another automatically. This is accomplished by using a *Master Scene* which coordinates the animation of independently controlled objects which make up the whole. The master scene commonly contains the background items of the graphics package. Such items can be looping backgrounds or the design items of the lower third, over the shoulders, and full-screen graphics. The *variable* or changing content, such as the text in a lower third, is stored separately in *Object Scenes.*

A Transition Logic package consists of:

Part	What it is
Master scene	The background scene that holds the logic. It stays loaded for the whole show.
Layer	An independent slot of the master, for example Fullscreen, Lower or Bug. Technically a top-level container with a Toggle plugin and its own transition director.
State	A named position of a layer. Every layer has the state O (Out: nothing on screen) and at least one on state, called IN by default. You can add more, for example TWO_LINES.
Transition	The animation between two states. It is the stretch of the layer's director timeline between the stop markers of the two states. Logicmaker creates the transitions; designers animate them in the Stage.
Object scene or Leaf scene (page)	A small scene holding the content of one layer in one state. This is what a control application takes.
Out page	A leaf scene that takes a layer to O. Operators use it to clear the layer.

When a lower third is played On Air, the object scene for the lower third is triggered. This tells Viz Engine to load the master scene, place the object scene inside the master, and animate the appropriate timeline(s). Thankfully, all of that is handled automatically, which is what makes Transition Logic so valuable.

Copyright © 2026 Vizrt

20.1 How a take works.

When a control application (for example Viz Trio, Viz Pilot or Viz Mosart) takes a leaf scene, the scene's content is loaded into its layer and the layer animates to the leaf's state. Whatever was on that layer before animates out through its own transition. Layers that are not addressed stay untouched - that is the point of Transition Logic.

Under the hood: the Toggle plugin gives every layer two placeholder subcontainers named <Layer>,A and <Layer>,B. Takes alternate between them so the outgoing and incoming content can cross-fade, and four small helper directors (<Layer>,AX, ,AO, ,BX, ,BO) animate the placeholders during those cross-fades. Logicmaker creates and maintains all of this; you normally only meet it in the Stage tree and in the health check

Transition Logic scenes are designed with a built in logic that allows for morphing background graphics, handles multiple layers of graphics, and manages the transition from the graphic to graphic of each layer. The built in logic also minimizes the operator's need to know how scenes affect each other, as the considerations of these various options are handled in the design phase. This ultimately results in a simplified work-flow and a look that is consistent and glitch-free.

To understand what Transition Logic is, one needs to understand a few basic concepts:

- A Transition Logic scene is not a single scene, but a set of Viz graphics scenes that consist of a *master scene* that may have multiple layers of graphics that can be On Air at the same time and independently controlled.
- Each layer in the master scene may have multiple referring *object scenes*. However, only one object scene per layer can be active at any given time.
- Transition Logic layers are conceptual, not spatial.

Note: With Transition Logic scene design, *take in* and *take out* commands are still used as with standalone scene design. Where standalone scene design demands that only a single scene can be On Air at a time, however, Transition Logic allows for more than one scene to be On Air simultaneously. This means using Transition Logic allows a graphic covering the lower third of the screen, and another covering the left and/or the right side of the screen for over the shoulder graphics, to be On Air at the same time.

Transition Logic can be played out by most of Vizrt's control applications such as Viz Trio, Viz Pilot and Viz Multichannel.

20.2 Architecture Overview

```
RENDERER
 └──── MAIN_LAYER (master scene – always on-air)
     |
     ├──── Container "Banner"                    ← design elements (backplate)
     |   ├──── Container "BannerLyr"             ← Toggle plugin here
     |   |   ├──── $current                      ← visible object scene
     |   |   |   └──── $object
     |   |   └──── $other                        ← next object (loading buffer)
     |   |       └──── $object
     |   └──── Container "BannerBackplate"       ← animated rectangle
     |
     ├──── Container "OTS"
     |   ├──── Container "OTSLyr"                ← Toggle plugin here
```

```
     |   |   ├─── $current
     |   |   └─── $other
     |   └─── Container "OTSBackplate"
     |
     └─── Container "Ticker"
         └─── Container "TickerLyr"            ← Toggle plugin here
             ├─── $current
             └─── $other

STAGE (Directors)
├─── Director "BannerLyr"    stops: $0 ──── $LOWERTHIRD ──── $0
|   ├─── BannerLyr,AX        (A-side cross-fade – 1s alpha dissolve)
|   ├─── BannerLyr,BX        (B-side cross-fade – must mirror AX)
|   ├─── BannerLyr,AO        (A-side object animation – auto-managed)
|   └─── BannerLyr,BO        (B-side object animation – auto-managed)
|
├─── Director "OTSLyr"       stops: $0 ──── $OTS_LEFT ──── $OTS_RIGHT ──── $0
|   ├─── OTSLyr,AX
|   ├─── OTSLyr,BX
|   ├─── OTSLyr,AO
|   └─── OTSLyr,BO
|
└─── Director "TickerLyr"    stops: $0 ──── $TICKER ──── $0
    └─── ...
```

Each layer operates independently, taking Banner Off Air does not affect OTS or Viz Ticker.

$current and $other: Content is always loaded onto `$other` (invisible). Data is pushed via ControlObject. Then `switch` triggers the cross-fade: `$current` fades out, `$other` becomes `$current` and fades in. The viewer never sees a loading state.

20.3 The Take Lifecycle (TCP)

This is the complete sequence that newsroom applications (Viz Trio, Viz Pilot, Media Sequencer Engine) execute when an operator presses **Take**. Understanding this sequence is essential for automation integrators.

20.3.1 First Take: Layer Goes from $O to $IN

```
# 1. Ensure renderer is ready
IS_RENDERER_READY

# 2. Load master scene (only needed once per session)
RENDERER*MAIN_LAYER SET_OBJECT SCENE*default

# 3. Reset the layer's current placeholder
RENDERER*MAIN_LAYER*TREE*$BannerLyr*FUNCTION*Toggle*reset_current INVOKE
```

Copyright © 2026 Vizrt

```
# 4. Load object scene into the invisible buffer ($other)
RENDERER*MAIN_LAYER*TREE*$BannerLyr*FUNCTION*Toggle*object SET GEOM*News/
LowerThird_Breaking

# 5. Push data to the not-yet-visible scene
RENDERER*MAIN_LAYER*TREE*$BannerLyr$other$object*FUNCTION*ControlObject*in SET "ON 01
SET BREAKING NEWS"
RENDERER*MAIN_LAYER*TREE*$BannerLyr$other$object*FUNCTION*ControlObject*in SET "ON 02
SET Details to follow"

# 6. Animate in — GOTO_TRIO triggers the state transition + Toggle cross-fade
RENDERER*MAIN_LAYER*STAGE*DIRECTOR*BannerLyr GOTO_TRIO $O $LOWERTHIRD
```

20.3.2 Update On Air (No Animation)

```
# Push new data to the currently visible scene
RENDERER*MAIN_LAYER*TREE*$BannerLyr$other$object*FUNCTION*ControlObject*in SET "ON 01
SET UPDATED HEADLINE"
```

20.3.3 Swap: A/B Switch to a Different Template

```
# 1. Load a different object scene onto $other
RENDERER*MAIN_LAYER*TREE*$BannerLyr*FUNCTION*Toggle*object SET GEOM*News/
LowerThird_TwoLine

# 2. Push data
RENDERER*MAIN_LAYER*TREE*$BannerLyr$other$object*FUNCTION*ControlObject*in SET "ON 01
SET New Story"
RENDERER*MAIN_LAYER*TREE*$BannerLyr$other$object*FUNCTION*ControlObject*in SET "ON 02
SET More details here"

# 3. Cross-fade switch (old fades out, new fades in — same state)
RENDERER*MAIN_LAYER*TREE*$BannerLyr*FUNCTION*Toggle*switch INVOKE
```

20.3.4 Take Out: Layer Goes from $IN to $O

```
# Animate out — backplate reverses, Toggle cross-fades object out
RENDERER*MAIN_LAYER*STAGE*DIRECTOR*BannerLyr GOTO_TRIO $LOWERTHIRD $O
```

20.3.5 Multi-Layer Concurrent Operation

All layers operate independently. You can take Banner in, then take OTS in, then take Banner out while OTS remains unaffected:

Copyright © 2026 Vizrt

```
# Take Banner in
RENDERER*MAIN_LAYER*TREE*$BannerLyr*FUNCTION*Toggle*reset_current INVOKE
RENDERER*MAIN_LAYER*TREE*$BannerLyr*FUNCTION*Toggle*object SET GEOM*News/Breaking
RENDERER*MAIN_LAYER*TREE*$BannerLyr$other$object*FUNCTION*ControlObject*in SET "ON 01
SET Headline"
RENDERER*MAIN_LAYER*STAGE*DIRECTOR*BannerLyr GOTO_TRIO $O $LOWERTHIRD

# Take OTS in (while Banner is on-air)
RENDERER*MAIN_LAYER*TREE*$OTSLyr*FUNCTION*Toggle*reset_current INVOKE
RENDERER*MAIN_LAYER*TREE*$OTSLyr*FUNCTION*Toggle*object SET GEOM*News/OTS_Graphic
RENDERER*MAIN_LAYER*TREE*$OTSLyr$other$object*FUNCTION*ControlObject*in SET "ON 01
SET Expert Name"
RENDERER*MAIN_LAYER*STAGE*DIRECTOR*OTSLyr GOTO_TRIO $O $OTS_LEFT

# Take Banner out (OTS stays on-air)
RENDERER*MAIN_LAYER*STAGE*DIRECTOR*BannerLyr GOTO_TRIO $LOWERTHIRD $O
```

20.3.6 Session Cleanup

```
# Full cleanup sequence – frees all cached resources
SCENE CLEANUP
GEOM CLEANUP
IMAGE CLEANUP
FONT CLEANUP
MATERIAL CLEANUP
MAPS CACHE CLEANUP
REND SET_OBJECT
POOLS CLEANUP
```

This section contains information on the following topics:

- Standalone versus Transition Logic Scene Design
- Adaptive Scenes and Subscene Mode
- Logicmaker
- Toggle Layer
- State Transition Animation
- Cross Animation
- Master Scene
- Object/Leaf Scene
- Troubleshooting

Copyright © 2026 Vizrt

20.4 Standalone versus Transition Logic Scene Design

20.4.1 What Transition Logic Solves

In traditional **standalone scene design**, taking one graphic On Air replaces whatever was previously shown with a hard cut. If you need a lower third and an over-the-shoulder on screen simultaneously, you're stuck.

Transition Logic (TL) solves this by separating a broadcast graphics package into:

- A **master scene** (always On Air) that contains animated backplates and layer placeholders.
- Multiple **object scenes** (templates) that carry the variable content (text, images, data tables) and swap in and out of layers independently.

Aspect	Standalone	Transition Logic
Scenes On Air simultaneously	One at a time	Multiple layers, independently controlled
Taking a new scene On Air	Hard-cuts previous scene	Other layers remain unaffected
Background animation	Reloads each time (may jitter)	Master scene stays resident (smooth)
State transitions (for example, small → large)	Must rebuild entire scene	Director animation between stop points
Operator complexity	Must manage scene order	Press **Take** and Transition Logic handles everything

TL is driven by three plug-ins working together:

1. **Toggle:** Creates A/B placeholders and cross-fade directors per layer on the master scene.
2. **ControlObject:** Declares which layer and state to target on each object scene.
3. **ControlList:** Exposes tabular/repeating data within an object scene (optional).

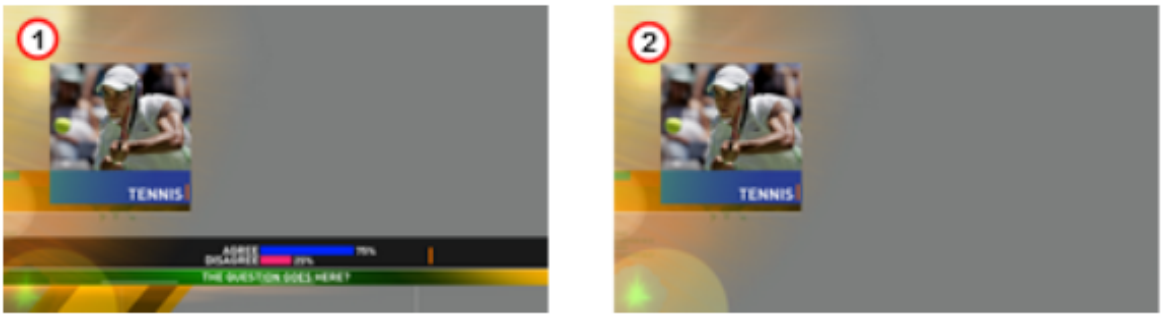
Unlike standalone scenes, a Transition Logic scene can have multiple defined layers that control the transitioning of multiple object scenes per layer (for example, transitioning between variants of lower and top thirds). The following images show the difference when executing commands towards Transition Logic scenes compared to standalone scenes. The images to the left (1) show a Transition Logic scene with two layers and two object scenes, an Over the Shoulder and a Lower Third. The images to the right (2) show two standalone scenes that are identical to the Transition Logic Object scenes.

Copyright © 2026 Vizrt

20.4.2 Take Lower Third



20.4.3 Take OTS



20.4.4 Take Out OTS



Copyright © 2026 Vizrt

20.4.5 Take Out Lower Third



The first action, *take* lower third, results in a lower third graphics being placed on-screen. The second action, *take* over the shoulder (OTS), places the OTS on-screen; however, with standalone scene design this also causes the lower third to be taken out, that is, a hard cut.

The third action, *take out* OTS, takes out the OTS; however, with Transition Logic scene design the lower third remains on-screen until it is taken out (last action).

Copyright © 2026 Vizrt

20.5 Adaptive Scenes and Subscene Mode

20.5.1 Merged Geometry vs Subscene Mode

By default Object scenes are handled as subscenes. This means the scene tree of the object scene is being added as subtree into the master scene. This is more efficient and does not require to create merged geometries, and it supports adaptive format handling as well.

In Merged Geometry mode, the object scene is brought as a merged geometry into the master scene.

> **Information:** We recommend to use Subscene Mode as it is much more efficient than the Merged Geometry mode. For backward compatibility, you can set `TransitionLogicUseMergedGeometries = 0` in the Viz Configuration file.

	Merged Geometry (GEOM*)	Subscene Mode (SCENE*)
How it loads	Object scene tree is merged directly into the master scene tree	Object scene is loaded as a separate scene reference (scene-in-scene)
Performance	Faster (single draw call tree)	Slight overhead (separate scene context)
Box Transformation	❌ Not supported	✅ Supported
Flexbox	❌ Not supported	✅ Supported
Object-local directors	Merged into master stage	Preserved in subscene scope
After switch	Works automatically	Requires explicit `continue INVOKE`
TCP prefix	`GEOM*path/to/scene`	`SCENE*path/to/scene`

20.5.2 Toggle Adaptive Properties

When using subscene mode with adaptive/responsive layouts:

Toggle Property	When to Enable
Support Box Transformation	Object scenes use Box Transformation for responsive positioning.

Copyright © 2026 Vizrt

Toggle Property	When to Enable
Support Flexbox layout	Object scenes use Flexbox for layout. Toggle uses `active` toggling instead of position-based animations.
Format Name	Forces subscenes to render in a specific format (for example, `16:9`, `9:16`). If empty, master scene format is used.

Save and reload after changing adaptive properties: After enabling *Support Box Transformation* or *Support Flexbox*, the scene must be saved and reloaded to regenerate the Toggle's internal structure.

20.5.3 Subscene Mode TCP Workflow

```
# Load via SCENE reference (not GEOM)
MAIN_SCENE*TREE*$BannerLyr*FUNCTION*Toggle*object SET SCENE*News/LowerThird

# Animate in
MAIN_SCENE*STAGE*DIRECTOR*BannerLyr SHOW $LOWERTHIRD
MAIN_SCENE*TREE*$BannerLyr*FUNCTION*Toggle*switch INVOKE

# IMPORTANT: In subscene mode, continue is required after switch
MAIN_SCENE*TREE*$BannerLyr*FUNCTION*Toggle*continue INVOKE
```

Copyright © 2026 Vizrt

20.6 Logicmaker

The Logicmaker is the Viz Artist editor for Transition Logic. It builds and maintains the master scene of a graphics package - its layers, their states and the transitions between them - and the leaf scenes (pages) that a control application takes on air. It works on the scene that is open in the Scene Editor and is available as the Logicmaker dock.

This guide explains the concepts, walks you through your first package, describes every part of the editor, shows complete workflows with examples, and lists every check the Logicmaker performs on a package together with the ways to resolve what it reports.

- Layers pane
- The State Graph
- Scenes Pane
- Control Objects Pane
- Health Check

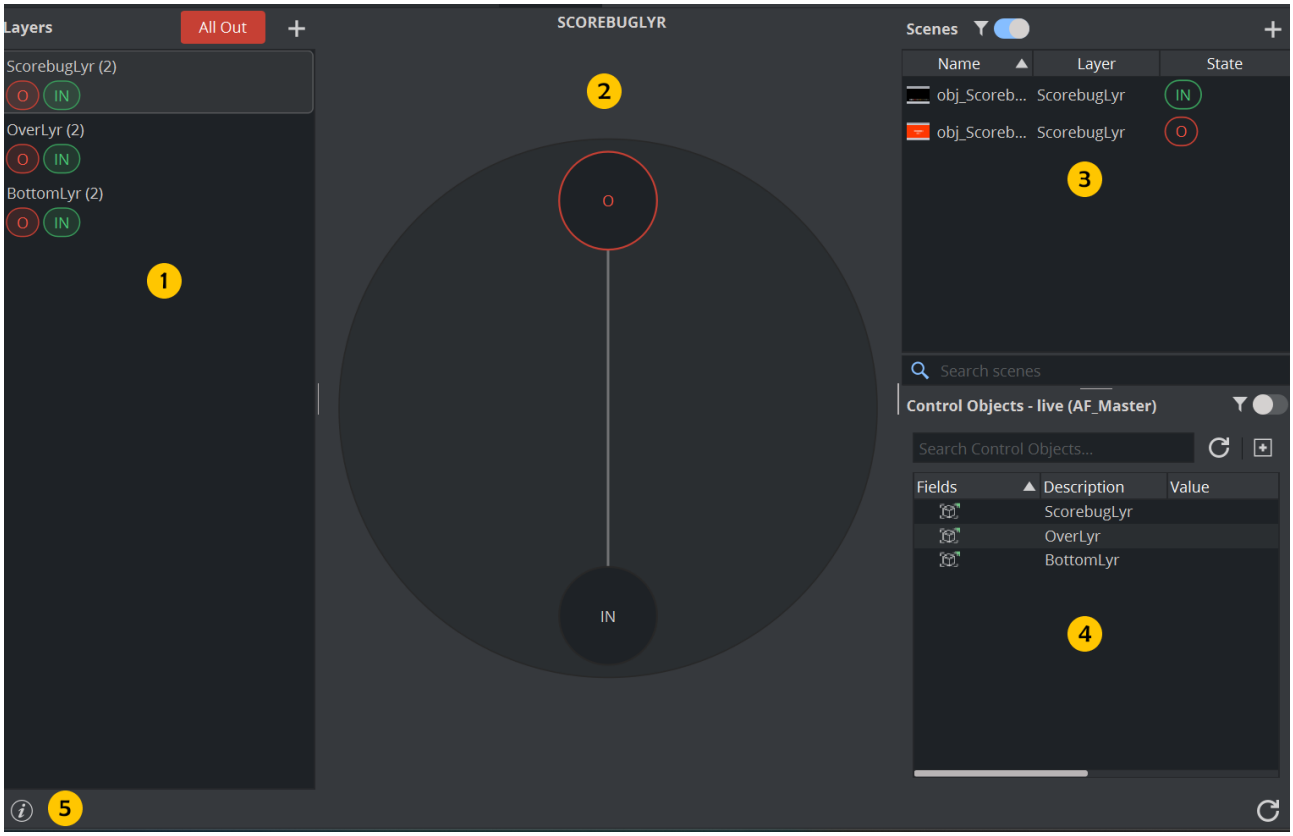
- Quick start
- User Interface
- Creating a package
    - The empty state
    - Start from scratch
        - Step 1 - Master and Layers
        - Step 2 - Leaf scenes
        - What Finish does
    - Generate from Directors
        - Step 1 - Directors
        - Example
- Messages and troubleshooting
- Reference
    - Naming rules
    - Good to know

20.6.1 Quick start

1. In the Asset View, select the folder that should hold the package, for example *Default/News*.
2. Open a new scene in the Scene Editor and open the Logicmaker dock. A scene without Transition Logic shows the empty state with two buttons.
3. Click Start from scratch. The Create Transition Logic wizard opens.
4. On Master and Layers, name the master, rename the first layer, add more layers with Add Layer, and add states with the + next to a layer's state badges. Click Next.
5. On Leaf scenes, add one scene per page and choose its layer and state. Click Finish.
6. In the Save transition logic master as... dialog confirm the folder and the name of the master.
7. Result: the master and all leaf scenes are created in that folder, the Asset View jumps to it, the Scene Editor shows the saved master and the Logicmaker lists its layers.
8. Select a layer and click its states in the graph - the transitions play in the renderer. Design them in the Stage, build the content of the leaf scenes, run the health check (the i button in the bottom bar), click All Out and save the master.

Copyright © 2026 Vizrt

20.6.2 User Interface



Area	Purpose
Layers (1)	All layers of the master with a badge for every state. Add, rename and delete layers; take states; All Out.
State graph (2)	The states of the selected layer on a circle. Take states, add/rename/ remove states, change transition durations.
Scenes (3)	The leaf scenes of the master. Create, preview, rename and delete pages.
Control Objects (4)	The fields exposed by the objects loaded into the master.
Bottom bar	The health check button, the status line and the reload button

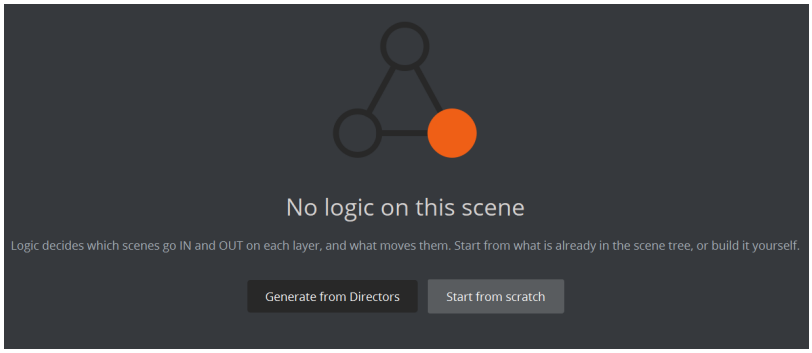
When you select a layer, the Logicmaker also selects the layer's container in Viz Artist and its transition director in the Stage dock, so you can jump straight from logic to animation work.

Copyright © 2026 Vizrt

20.6.3 Creating a package

The empty state

When the loaded scene has no transition layers, the dock shows an empty state instead of the editor.



There are three variants:

You see	When	What you can do
No scene loaded	No scene is open in the Scene Editor.	Open a scene.
No logic on this scene with the buttons **Generate from Directors** and **Start from scratch**	The scene has no Toggle layers and is not a page of a master.	Build the logic by Starting from scratch or use existing Directors
Scene belongs to "Master" with the button **Open master scene**	The scene is a leaf scene (it has a ControlObject naming a master).	Open the master to edit the logic this page is driven by.

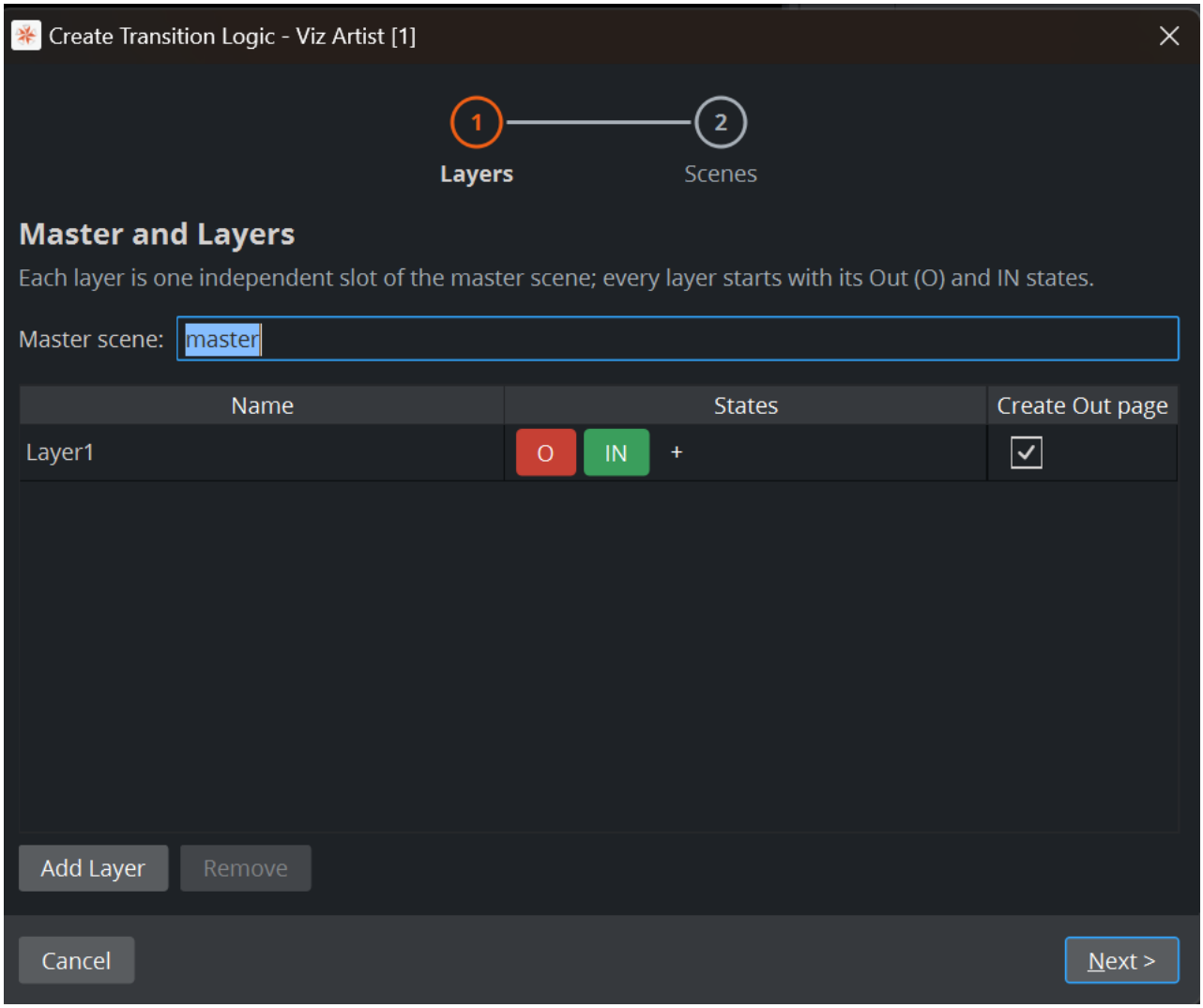
Generate from Directors is enabled only when the scene tree contains named top-level containers to offer for conversion.
Open master scene loads the master (a sibling of the page in the same folder) into the Scene Editor.

While a scene is being analyzed, the Layers section shows a spinner with the text **Gathering information**.

Start from scratch

Use this when you build a package from nothing. The wizard has two steps.

Copyright © 2026 Vizrt



Step 1 - Master and Layers

- **Master scene** - the name of the master. For an unsaved scene the field is editable (default `master`)
- **Layer table** - one row per layer with the columns **Name**, **States** and **Create Out page**:
  - **Name** - editable. Names must be unique and valid **(see chapter 12).**
  - **States** - every layer shows a red **O** and a green **IN** badge, then one green badge per extra state, then a **+**.
    - Click **+** and enter a name to add a state.
    - **Right-click IN** and choose *Rename...* to give the on state another name, for example `ON`.
    - **Click** an added state badge to remove it, or **right-click** it for *Rename...* and *Remove*.
  - **Create Out page** - ticked by default. Creates a scene `<Layer>_OUT` that takes the layer to **O**.
- **Add Layer** adds a row with the next free name (`Layer1`, `Layer2`, ...). **Remove** deletes the selected row and is enabled only while a row is selected.

Copyright © 2026 Vizrt

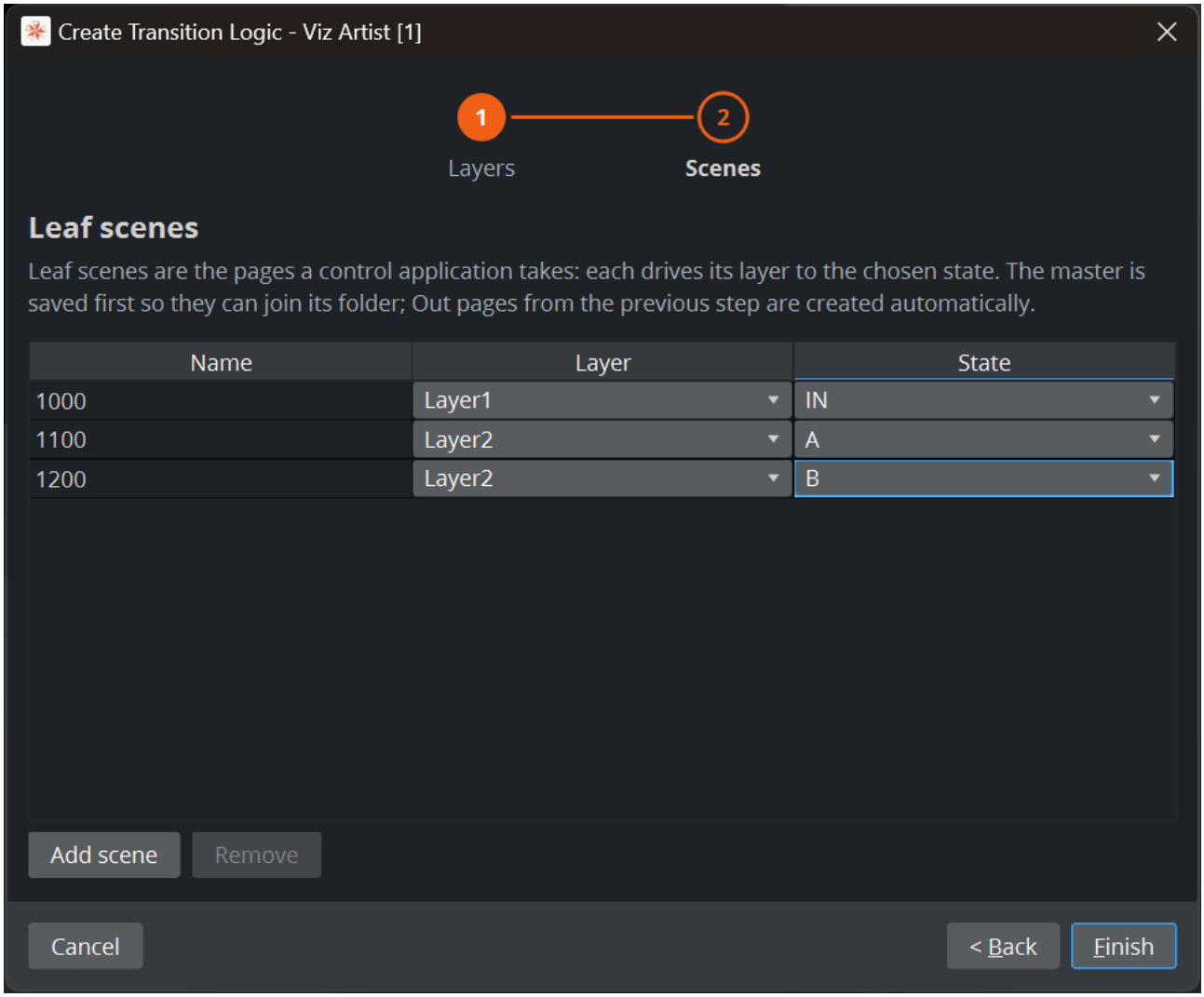
Next to the buttons a yellow hint explains why **Next** is disabled:

Hint	Meaning
The master scene needs a usable name.	The master name is empty or contains forbidden characters.
Add at least one layer.	The table is empty.
A layer is missing its name.	A name cell is empty.
Layer name "X" is not usable.	The name contains a space or a forbidden character.
Two layers are named "X".	Layer names must be unique.

Step 2 - Leaf scenes

Leaf scenes are the pages a control application takes. Each one drives its layer to the chosen state.

Copyright © 2026 Vizrt



- **Add scene** adds a row. **Name** defaults to the next free page number (`1000` , `1100` , `1200` , ...). **Layer** lists the layers from step 1. **State** offers the on state, the extra states and **O**.
- **Remove** deletes the selected row (enabled only while a row is selected).
- The **Out pages** from step 1 are created automatically; do not add them here.
- You may finish without any scene and create the pages later **(section 7.2).**

Hints on this step: *A scene is missing its name*, *Scene name "X" is not usable*, *Two scenes are named "X"*, and *"X" collides with a layer's automatic Out page* (for example a scene called `Lower_OUT` while **Create Out page** is ticked for `Lower`).

When you go **Back** and change layers, the choices you made for the scenes are kept as far as they are still valid.

Clicking on Finsih lets you choose the location where Master Scene, Leaf Scenes and Out Scenes are created.

What Finish does

1. The layers are created with their states. New layers are inserted at the **top of the scene tree**.
2. The master is saved to the chosen location. (A master that is already saved is saved in place.)

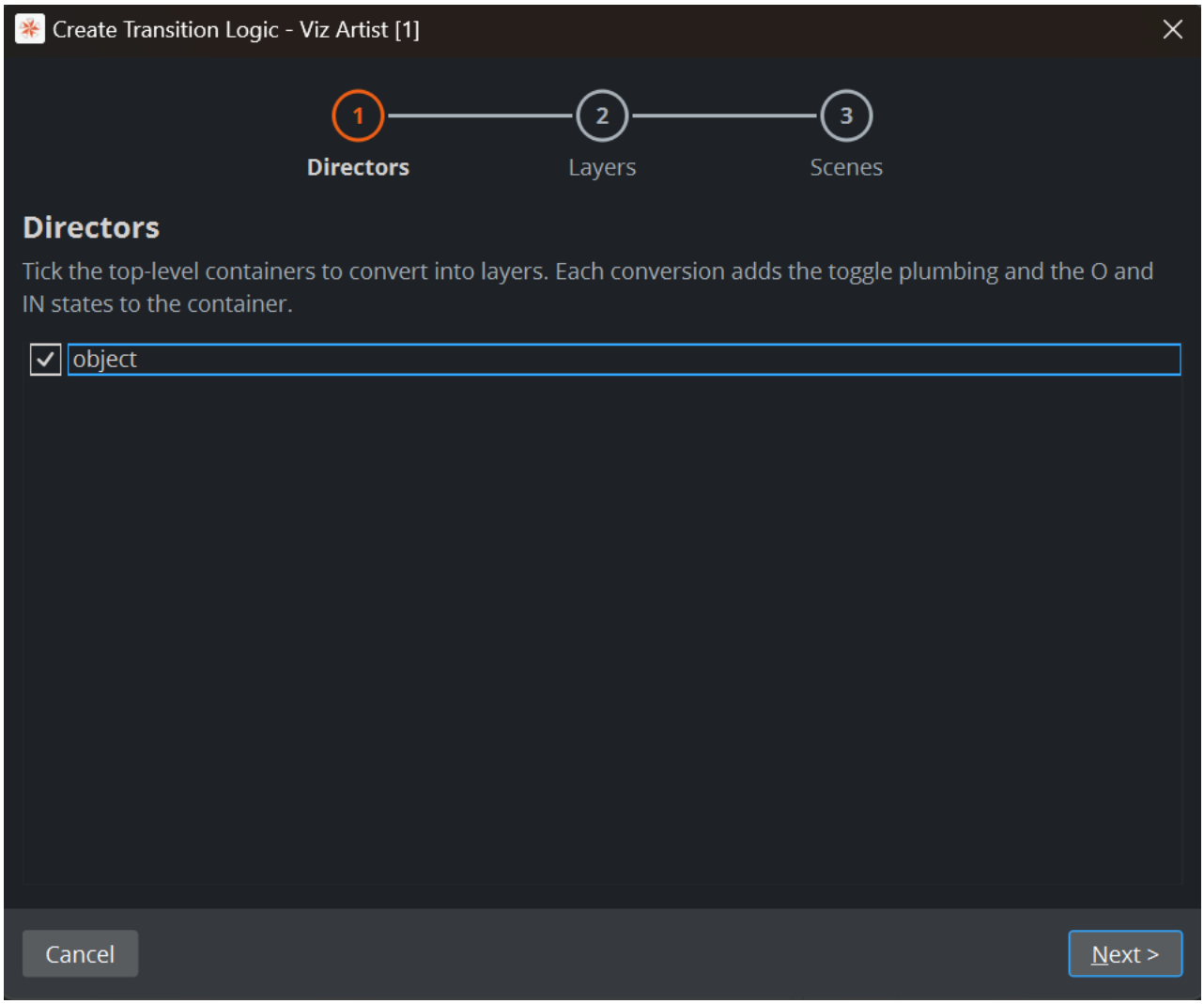
Copyright © 2026 Vizrt

3. The Out pages and then the leaf scenes are created in the same folder.
4. The Scene Editor switches to the **saved master**, the Asset View jumps to its folder with the master selected, the Logicmaker shows the layers, and the status bar says *Transition logic created.*

A new leaf scene contains one container named `object` carrying the **ControlObject** plugin. Its fields `layer`, `state` and `background` tell the master where the page belongs; `description` is filled with `<layer> <state>` (for example *Lower IN*) so the page is recognizable in a control application. You build the page's graphics inside `object`.

Generate from Directors

Use this when the scene already has top-level containers that should become layers. The wizard has three steps: **Directors**, **Master and Layers**, **Leaf scenes**.



Step 1 - Directors

The detected top-level containers of the scene tree are listed with check boxes, all ticked. Tick the ones to convert. Each conversion adds the Toggle plumbing and the **O** and **IN** states to the container; the container keeps its name and its content.

Two kinds of entries cannot be ticked:

- *" - N states (designed animation; wire up manually for now)"* - the container already has a director of the same name that carries designed stop markers. Converting it automatically would collide with that animation, so wire such containers up by hand in the Stage.
- *" - (name not usable as a layer)"* - the name contains forbidden characters. Rename the container in the scene tree and open the wizard again.

If nothing is detected the page says *Nothing detected in the scene tree - continue to build layers by hand.*

Steps 2 and 3 work as described in Create from Scratch

Example

A scene has the top-level containers `Lower`, `Bug` and `Clock`. `Clock` already has a director named `Clock` with five stop markers. In the wizard `Lower` and `Bug` are ticked and `Clock` is disabled as *designed animation*. After **Finish**, `Lower` and `Bug` are layers with the states O and IN and a one-second transition; `Clock` is untouched and has to be wired up manually.

20.6.4 Messages and troubleshooting

Messages appear in the **status bar** at the bottom. A warning has a warning sign and a yellow color; messages disappear after about eight seconds.

Message or symptom	Cause	What to do
No scene loaded	No scene is open in the Scene Editor.	Open a scene.
Generate from Directors is disabled	No convertible top-level containers were found.	Use **Start from scratch** or add named top-level containers.
A container is listed but cannot be ticked	It carries a designed animation, or its name is not usable.	Wire it up by hand in the Stage, or rename it and reopen the wizard.

Copyright © 2026 Vizrt

Message or symptom	Cause	What to do
Could not read the scene's transition logic.	The scene could not be analyzed (connection problem, scene changed while loading). The status bar shows the engine's message.	Click **reload**; check the connection to Viz Engine and the Graphic Hub.
Skipped: Container X (#id) carries a Toggle plugin but no transition logic: the Toggle has no out channel, and no stage director is named after the container.	A container has a Toggle plugin that was never set up as a layer. The rest of the scene loads normally.	Remove the stray Toggle plugin from that container, or delete the container and create a proper layer with **+**.
Background scene has not been saved.	The master has no Graphic Hub location, so pages cannot be created beside it.	Save the master in the Scene Editor.
Scene already contains a layer with the same name. Please choose a different name.	Layer names are unique within the master.	Choose another name.
Folder already contains a scene with the same name. Please choose a different name.	Scene names are unique within the folder (automatic Out pages count).	Choose another name.
Invalid layer name / Invalid scene name / Invalid state name: X	The name is empty or contains a space or a forbidden character.	See chapter 12.
State already exists: X	The layer already has a state of that name.	Choose another name.
Can't edit Out state. / Can't remove Out state.	`0` is the anchor of every layer.	Leave `0` alone.
Can't have less than 2 states.	A layer needs `0` and at least one more state.	Delete the layer if that is the intent.
There is no direct transition between A and B on the timeline.	The two states have no transition.	Run the health check, see 10.1.
Duration must be at least 0.05 seconds.	The minimum transition length is 0.05 s.	Enter a larger value.

Copyright © 2026 Vizrt

Message or symptom	Cause	What to do
Shortening would cross a designer keyframe at T s ...	A keyframe lies inside the transition beyond the new end.	Move or delete that keyframe in the Stage first.
"X" is a master scene (contains toggle layers) and cannot be loaded into a layer.	You clicked a master scene in the Scenes pane.	Pages are the other rows; the master is shown as `[master]`.
Cannot rename director to "X": an ancestor container name contains characters the engine rejects.	The helper director names are derived from the names of the container's ancestors, one of which has a forbidden character.	Rename that ancestor container in the Scene Tree, then rename the layer again.
The transition director of layer X was not found on the stage.	The layer's director was deleted or renamed outside the Logicmaker.	See 10.4; recreate the director's name or the layer.
A scene shows a warning sign	Its ControlObject names a layer, state or master that does not exist.	See 10.7.
The highlighted state differs from what is on air	The Logicmaker shows the state **it** last took. It does not follow takes made by a control application, and after **reload** no state is highlighted.	Take a state, or click **All Out**, to bring the display back in line.
The Logicmaker shows old data after you edited the Stage or Scene Tree	Changes made elsewhere are not tracked live.	Click **reload**. The dock also reloads when you undo or redo.

20.6.5 Reference

Naming rules

Layer, state, master and scene names must not be empty, must not contain spaces and must not contain any of these characters:

```
$  {  }  #  \  *  ?  [  ]  "
```

State names must be unique within the layer; **O** is reserved for the Out state. Layer names must be unique within the master, scene names within the folder. The input fields already refuse the forbidden characters.

Copyright © 2026 Vizrt

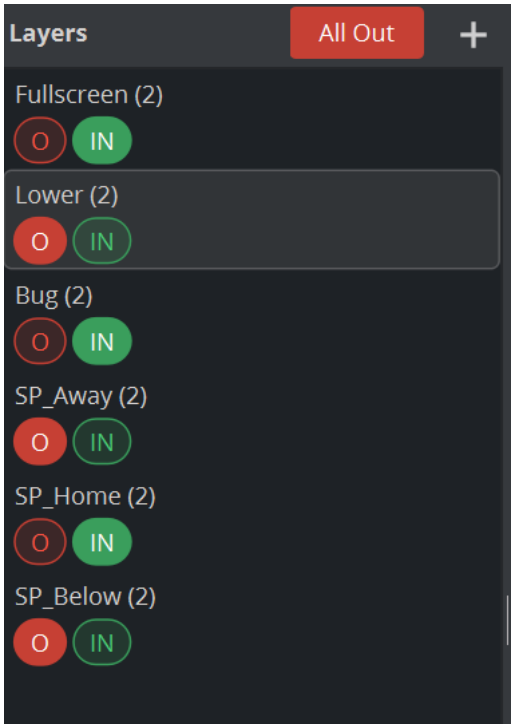
Good to know

- The Logicmaker works on the scene that is open in the Scene Editor and reloads when that scene changes. While the dock is hidden it reloads when it is shown.
- Changes to **layers, states and durations** modify the loaded master: **save it** to keep them. **Leaf-scene** operations (create, rename, delete, retarget) are written to the Graphic Hub immediately.
- Removing or recreating structure outside the Logicmaker (Scene Tree, Stage) is possible but bypasses its consistency work; the health check finds what drifted.
- Containers that already carry a designed director animation cannot be converted automatically yet.

Copyright © 2026 Vizrt

20.6.6 Layers pane

Each layer row shows the layer name with its **number of states** in brackets, for example `Lower (3)`, and a **badge for every state**: red for **O**, green for all others. The state the Logicmaker last took on that layer is **filled solid**.



Action	How
Select a layer	Click the row. The graph shows its states, its container is selected in Viz Artist and its director in the Stage.
Take a state	Click a state badge. This also works on layers that are not selected.
Jump to a state without animation	**Shift**+click the badge.
Add a state	Hover the row and click the **+** at its right end.
Rename the layer	Select it and press **F2**, or right-click and choose **Rename**.
Delete the layer	Right-click and choose **Delete**, or press **Delete**.

Copyright © 2026 Vizrt

Action	How
Add a layer	The **+** in the header, or right-click and choose **Insert new Layer**.
Take all layers to Out	The red **All Out** button.

Adding a layer



The **+** in the header slides in the **New Layer** form: a **Name** field (a free name such as `NewLayer` is proposed) and the check box **Create Out page** (ticked). **Create** is enabled when the name is valid and unique. **Enter** creates, **Esc** cancels. The new layer starts with **O** and **IN** and a one-second transition. With **Create Out page** ticked, a scene `<Layer>_OUT` is created as well.

Changes to layers and states modify the loaded master, so **save the master** in the Scene Editor.

Renaming a layer

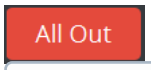
Press **F2**, type the new name and press **Enter**. The Logicmaker renames the layer's container and its directors together and refreshes the Stage.

> If leaf scenes refer to the layer, a dialog asks first if these scenes should also be changed or not. By default Layers and scenes will be changed.

Deleting a layer

After the confirmation *Do you want to delete Layer: Lower?* the layer's **container, including everything inside it, and its transition directors** are removed from the master.
Leaf scenes of that layer are **not** deleted; they show a warning sign until you delete or retarget them.

All Out



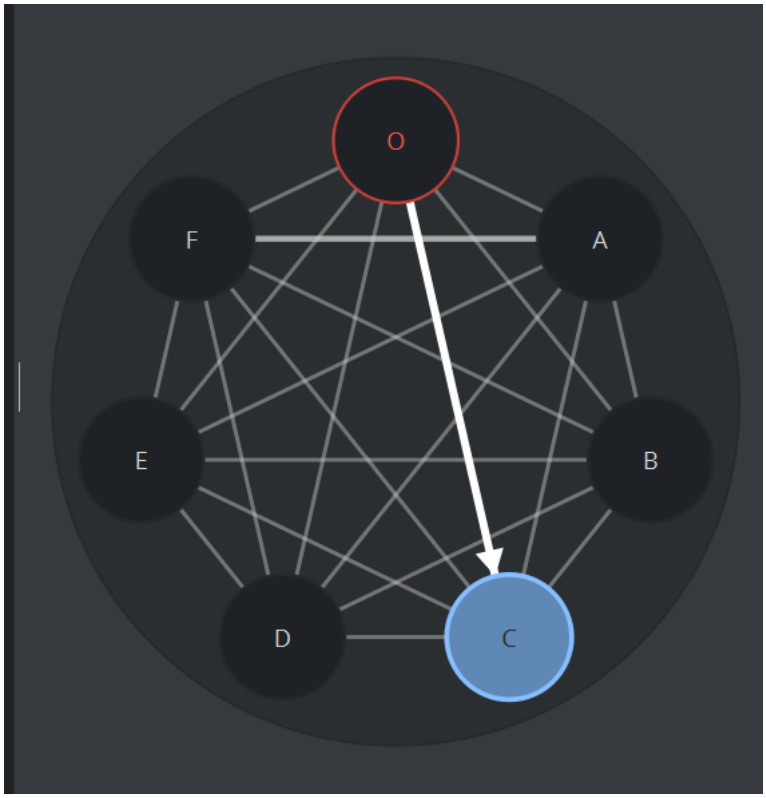
All Out takes every layer to its Out state.

> Do this before you save the master: a master that is saved mid-animation shows content the moment it is loaded.

Copyright © 2026 Vizrt

20.6.7 The State Graph

States and transitions - the state graph



The graph shows the states of the selected layer on a circle: **O** at the top in red, all other states in green. A line connects each pair of states; each line stands for the transition between them. The **current state** is filled. The header above the graph shows the layer name; below it a line reads *State: X* or *Animation from A to B* for the current selection. Hover a line to see its duration, for example *O → IN: 1 s* (or *no direct transition*).

Action	How
Take a state (animated)	Click the state. A white arrow travels along the line for the real duration of the transition.
Jump to a state (cut)	**Shift**+click the state.
Rename a state	Double-click it, or select it and press **F2**, or right-click and choose **Rename**. **Enter** confirms, **Esc** cancels.
Remove a state	Select it and press **Delete**, or right-click and choose **Remove**.

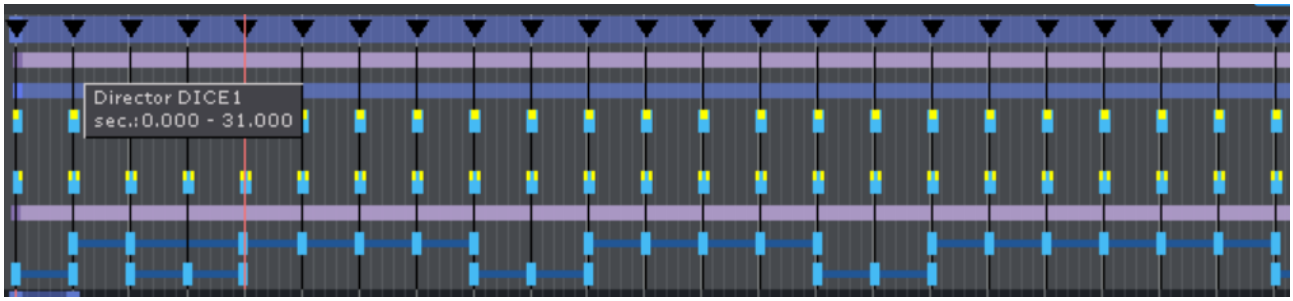
Copyright © 2026 Vizrt

Action	How
Add a state	Right-click empty space and choose **Add state**, or use the **+** on the layer row.
Change a transition's duration	Double-click the line, or right-click it and choose **Set duration...**.

Adding a state

A new state named `State`, `State2`, ... appears with its name editor already open. Type the name and press **Enter**; press **Esc** to keep the generated name.

The Logicmaker will do the work for you:
it appends the transitions the new state needs so that **every state can reach every other state**; each new transition is one second long.
Also see The State Graph .



Renaming a state

The name must be valid and not used by another state of the layer; **O** cannot be renamed (*Can't edit Out state.*). If leaf scenes still target the old name, a dialog asks:

> *State "a" was renamed to "A". 1 scene(s) still target "a": 1400. Update them to target "A"?*

Answer **Yes** to retarget those scenes (the status bar confirms *Updated 1 scene(s).*) or **No** to leave them; scenes left behind show a warning sign and are reported by the health check.

Removing a state

After the confirmation *t*he Logicmaker removes the state **together with its whole footprint on the timeline**:

- its stop markers, including any stop that only existed as a transition target of that state;
- tags inside the removed stretches and the **designer keyframes** of the removed transitions;
- the gaps are closed: later transitions move earlier;
- helper directors that become empty are removed, and the Toggle's keyframes are rebuilt from the remaining stops.

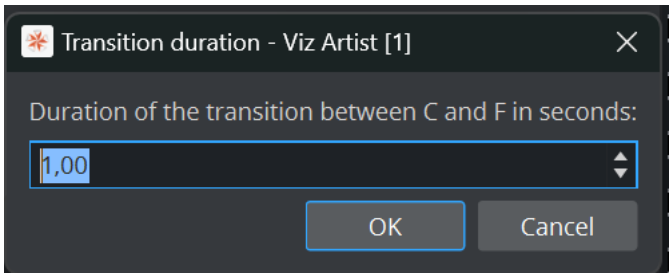
Copyright © 2026 Vizrt

Two guard rails always apply: **O cannot be removed** (*Can't remove Out state.*) and **a layer never drops below two states** (*Can't have less than 2 states.*). If the removed state was the layer's **current state**, the layer is taken to **O** so that later takes start from a valid state.

> ⚠ Removing a state discards the animation you designed on its transitions.
> Leaf scenes that target it are not deleted; they show a warning sign.

Changing a transition's duration

Double-click a line (or right-click it and choose **Set duration...**).



- The new duration is applied to **every direction that exists** on the timeline. A pair can have separate transitions A to B and B to A, and both are changed so the line stays symmetric.
- Everything after the changed stretch **shifts** by the difference, **including designer keyframes** on the director; the Toggle's keyframes are rebuilt to match.
- Designer keyframes **inside** the changed transition keep their position.
- **Shortening** is refused if the new end would fall before a designer keyframe inside the transition: *Shortening would cross a designer keyframe at T s inside the transition; retime or remove those keyframes first.*

Example: setting `O → IN` from 1 s to 2 s moves every later stop and keyframe by one second; the keyframes you placed inside the first second stay where they are.

Designing the animation in the Stage

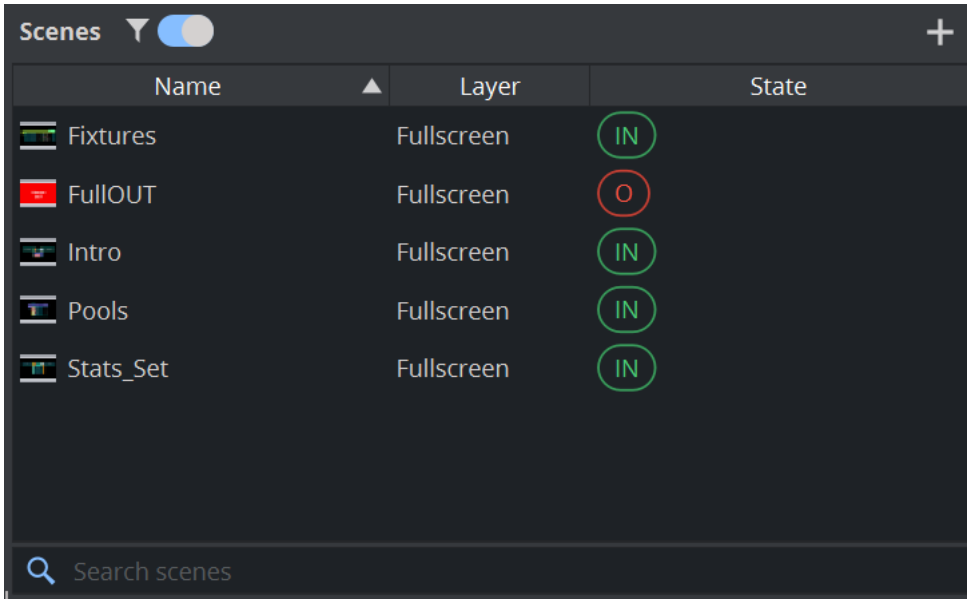
The Logicmaker creates the structure; the look of a transition is designed in the **Stage**.

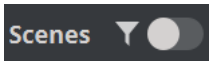
1. Select the layer. Its transition director is selected and revealed in the Stage dock.
2. The director's timeline carries one **stop marker per state**, named like the state. The stretch between two markers is the transition between those states.
3. Add keyframes for the layer container's properties (for example Position or Alpha) **inside** the stretch. For a lower third, slide it up between the O marker and the IN marker.
4. Test in the Logicmaker: click the states in the graph. The arrow shows the real duration.
5. Adjust the length with **Set duration...** if needed. Run the health check: *Transitions without animation* lists stretches you forgot to design.

Copyright © 2026 Vizrt

20.6.8 Scenes Pane

The table lists the scenes that live in the **master's folder**, with a thumbnail and the columns **Name**, **Layer** and **State**. Rows are sorted by name with numbers in numeric order.
By default only those scens which belong to a certain layer will be shown. See image below, it only shows the Scenes attached to the “Fullscreen” layer.





Use the switch on the top to toggle between the filtered view (only selected layer) or to see all scenes inside the current Graphic Hub folder.

Also see Scenes Pane

Logicmaker needs to query all scenes in a folder first to query its information.

Action	How
Take a scene in the master	**Click** the row.
Create a scene	The **+** in the header, or right-click and choose **Insert new Scene**.
Rename a scene	Double-click the name, or press **F2**.
Delete a scene	Right-click and choose **Delete**, or press **Delete**.
Search	Type in the **Search scenes** field at the bottom.

Copyright © 2026 Vizrt

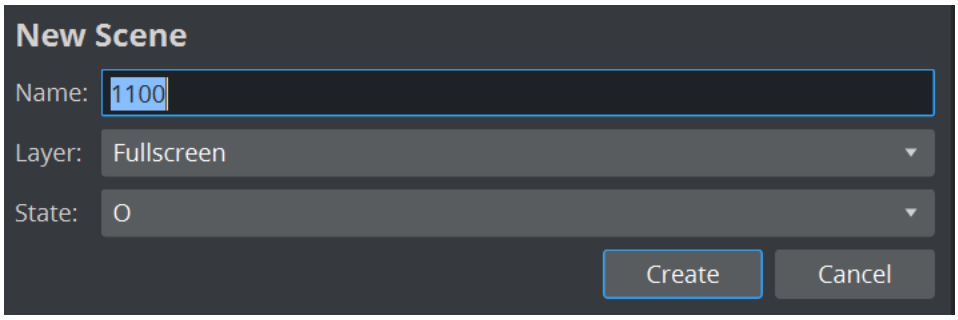
Action	How
Filter by layer	The funnel switch in the header.

Taking a scene

Clicking a row does what a control application does: it loads the scene's content into its layer in the master and animates the layer to the scene's state. The layer is selected, its state highlighted and the transition arrow plays; with the filter switch of the Control Objects panel on, the panel switches to the scene's fields. Use it to preview a page in the context of the master.

> A master scene cannot be loaded into a layer.

Creating a scene



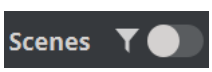
The **New Scene** form has **Name** (the next free page number such as `1000`, `1100`, ... is proposed), **Layer** and **State**. **Create** is enabled when the name is valid, unique in the folder and a state is selected.
The scene is created and saved in the master's folder, appears in the list and is selected. It needs a **saved master** (*Background scene has not been saved.* otherwise).

To create an Out page by hand, choose the layer, the state **O** and a name such as `Lower_OUT`.

Renaming and deleting

Renaming a scene renames it in the Graphic Hub; a name that already exists in the folder is rejected.
Be careful, when **deleting the scene from the Graphic Hub**; this cannot be undone from the Logicmaker.

The layer filter



The funnel switch **starts on**: only the scenes of the selected layer are shown. Switch it off to see **all** scenes of the folder. With the filter off the table also shows entries that are not ordinary pages:

Copyright © 2026 Vizrt

Shown as	Meaning
`1400` with a warning sign	The scene's ControlObject names a layer, state or master that does not exist. See health check *Scenes pointing at a wrong master, layer or state*.
`[name]` in square brackets	The scene has no ControlObject; it is not a page of Transition Logic.
Layer column `[master]`	A master scene (it contains Toggle layers), possibly this one.
Layer column `[legacy]`, state `[combo]`	A scene of the legacy combined format.
Layer column `...`	Still being examined.

What a page needs

A page is an ordinary scene with the container `object` and its ControlObject. Build the graphics inside `object` and add Control plugins (for example ControlText for text, ControlImage for logos) for the fields operators fill in. The ControlObject's `layer`, `state` and `background` entries link the page to the master; keep them intact.

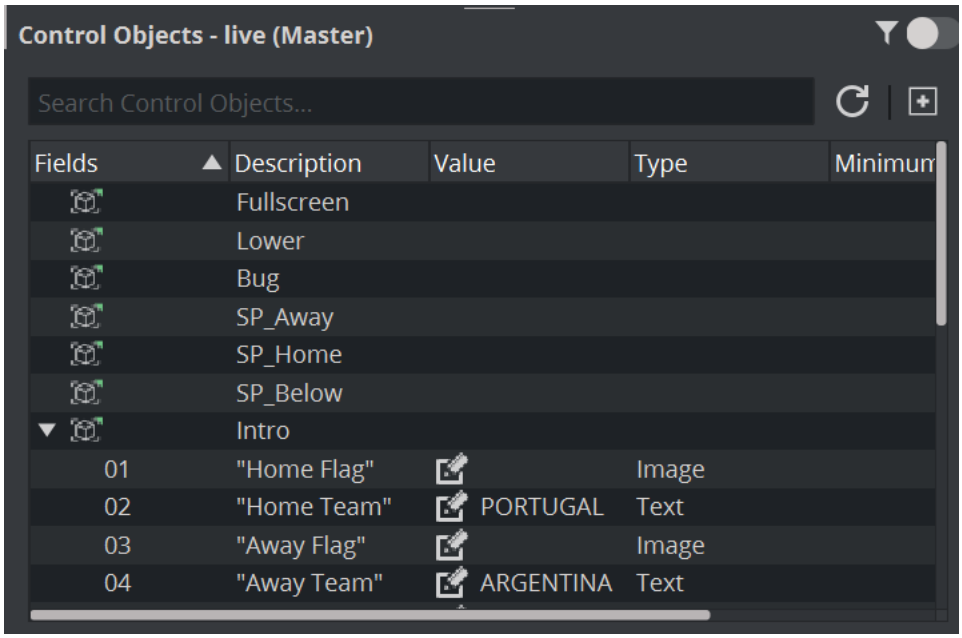
Subscenes and merged geometries

By default the page is referenced as a **subscene**, which is the standard workflow. If the legacy configuration flag `TransitionLogicUseMergedGeometries` is active, the Logicmaker shows once per master the notice *Legacy merged-geometries workflow is active (TransitionLogicUseMergedGeometries).* and adds the check *Merged geometries for all scenes* to the health check.

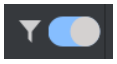
Copyright © 2026 Vizrt

20.6.9 Control Objects Pane

The panel lists the controllable fields of the objects currently loaded into the master - what a control application would show to the operator. Values changed here are applied to the rendered master immediately, the same way a control application sends them.



The **funnel switch** in the panel header decides how much is listed:

Title		Shows
Control Objects - live (NewsMaster)		The fields of everything loaded into the master. This is what you see while the switch is **off**.
Control Objects - 1000		Only the fields of the scene you selected or took last in the Scenes pane. This needs the switch to be **on**.

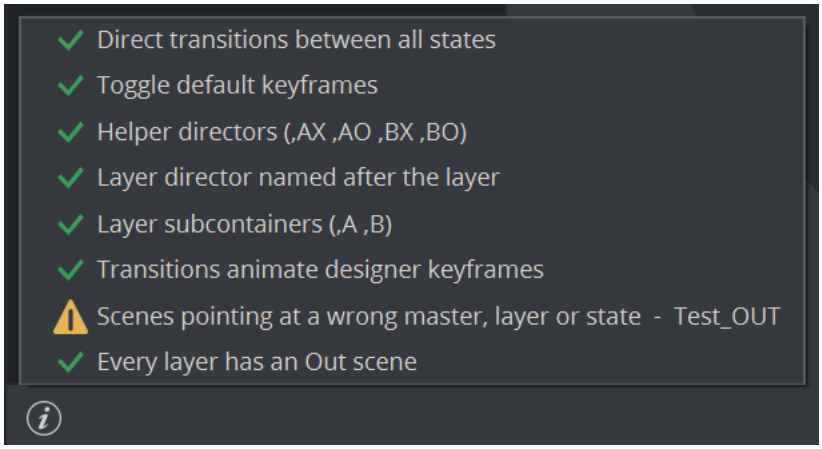
With the switch on, the panel follows your selection in the Scenes pane and moves to the scene you take; selecting the master row returns it to the whole master.

Copyright © 2026 Vizrt

20.6.10 Health Check

As debugging Transition Logic scenes is complex, Logicmaker comes with a health check feature. Click the **i** button in the bottom bar. The button spins while the checks run, then a checklist opens.
A passing check reads as a statement of what holds, with a green check mark; a failing check names the problem, has a warning sign and lists the affected layers or scenes.
Hover any line for an explanation of the check and the suggested fix. A scene without transition layers reports *No transition logic loaded.*

Example report:



The report is a snapshot for the master that was loaded when you clicked the button; switching scenes while the checks run discards it.

Direct transitions between all states

- **Checks:** on each layer's director the stop markers form transitions that connect **every pair of states** (for n states, n(n-1)/2 distinct pairs).
- **Why it matters:** between two states without a direct transition the layer cannot animate; it can only cut.
- **Typical causes:** the layer was built or edited by hand in the Stage (stop markers moved or deleted), or a state was added or removed outside the Logicmaker.
- **How to resolve:**
  a. Select the layer; its director appears in the Stage; inspect the stop markers.
  b. For states you have not designed yet: remove the state in the Logicmaker and add it again; the Logicmaker creates all missing transitions (one second each).
  c. For designed states: add the missing stretch in the Stage by placing the stop markers of the two states next to each other, then rebuild the Toggle keyframes (see 10.2).
  d. Run the check again.

Toggle default keyframes

- **Checks:** the Toggle's `out` and `switch_in` button channels exist and each holds exactly **two keyframes per transition** (2 x (number of stops - 1)).

Copyright © 2026 Vizrt

- **Why it matters:** the Toggle uses these keyframes to switch between the two placeholders and to cross-fade during a take. Out of step with the stops, a take can show the wrong content or the wrong timing.
- **Typical causes:** stops moved, added or deleted by hand in the Stage; the Toggle's keyframes edited by hand; a layer copied or imported from a package with a different structure.
- **How to resolve:**
  - Use the Toggle plugin's **Delete Keyframes** and then **Default Keyframes** functions; they regenerate the standard pattern from the current stops. Or
  - change a transition's duration in the Logicmaker (for example 1.00 to 1.10 and back); the Logicmaker rebuilds the channels as part of the edit.
- **Good to know:** the check measures deviation from the *standard pattern*. A layer with hand-made hold keyframes can take correctly and still be reported. The Logicmaker replaces the Toggle's keyframes with the standard pattern when you change a duration or remove a state, so hand-made ones would be lost then.

Helper directors (,AX ,AO ,BX ,BO)

- **Checks:** a layer has either **none** of the four crossfade helper directors or **all four**. The report lists what is missing per layer, for example `Lower: missing ,AO ,BO`.
- **Not a problem:** a layer without any helper directors is valid - fresh layers start that way, and removing a state clears helpers that become empty. Only a **partial** set is reported.
- **Other lines:** `<layer>: missing transition director ...` means the layer's director could not be found; `<layer>: stage unavailable` means the Stage could not be read (run the check again).
- **Typical causes of a partial set:** a helper deleted or renamed by hand, or a rename done outside the Logicmaker.
- **How to resolve:**
  a. Rename the layer in the Logicmaker to another name and back (choose **Layer only** in the dialog if you do not want the pages touched). All existing helpers are renamed consistently with the layer.
  b. If helpers are still missing, delete the remaining helper directors of that layer in the Stage - a layer with none is valid - or recreate the layer.

Layer director named after the layer

- **Checks:** the director that the Toggle's `out` channel animates is named **exactly like the layer**.
- **Why it matters:** control applications and the Logicmaker address the director by the layer's name; a different name breaks takes by name.
- **Typical causes:** a layer container was copied (for example `LeftBar` to `LeftBarExt`) and renamed in the Scene Tree, while its director kept the old name; or the container was renamed outside the Logicmaker.
- **How to resolve:**
  - In the **Stage**, rename the director to exactly the layer's name (and the helpers to `<name>,AX` and so on), or
  - rename the layer in the Logicmaker to another name and back; the container and all directors follow.

Copyright © 2026 Vizrt

Layer subcontainers (,A ,B)

- **Checks:** the layer container has the two placeholder subcontainers whose names end in `,A` and `,B` directly below it (`Lower,A` and `Lower,B`).
- **Why it matters:** the Toggle loads the content of a take into these containers; without them nothing can be shown.
- **Typical causes:** a placeholder was deleted, renamed or moved in the Scene Tree.
- **How to resolve:** recreate the missing placeholder as an empty child container of the layer container and name it `<Layer>,A` or `<Layer>,B` , then run the check again.

Transitions animate designer keyframes

- **Checks:** for layers whose director animates designer properties (for example Position or Alpha), **every transition contains at least one keyframe**. A keyframe at the start of a transition counts for the previous one; one at its end counts for it. Layers whose director has no property channels at all (pure toggles) are skipped.
- **Why it matters:** a transition without keyframes does not animate the layer: the content switches with no movement (a hard cut).
- **The report** names the affected transitions as `Layer (from→to)` , for example `Lower (O→IN, IN→O)` .
- **How to resolve:** select the layer, find the listed transition between the two stop markers in the Stage and add keyframes. If the cut is intended, you can leave the finding.

Scene identifiers (background, layer, state)

- **Checks:** every leaf scene in the folder with a ControlObject names **this master** in `background` , an **existing layer** in `layer` and one of that layer's **states** in `state` . Scenes that fail show a warning sign in the Scenes pane and are listed here.
- **Why it matters:** a page with wrong identifiers cannot be taken: the control application does not find its layer or state.
- **Typical causes:**
  - a layer or state was renamed or removed without letting the scenes follow (answering **Layer only** or **No**);
  - the master was saved under another name, so every page still names the old master;
  - a page was copied from another package or edited by hand.
- **How to resolve:**
  - Open the scene, select its `object` container and correct the ControlObject entries `background` , `layer` and `state` , then save the scene. Or
  - delete the page and create it again with **+** in the Scenes pane.
  - Next time answer **Yes** / **Layer and scenes** in the rename dialogs so the pages follow.

Copyright © 2026 Vizrt

Every layer has an Out scene

- **Checks:** each layer has at least one leaf scene whose state is **O**.
- **Why it matters:** without such a page operators cannot clear the layer from a control application.
- **How to resolve:** click **+** in the Scenes pane, choose the layer and the state **O** and name it `<Layer>_OUT`. Layers created with **Create Out page** ticked have one already. A layer that is meant to stay on air permanently can be left without one.

Merged geometries for all scenes (legacy workflow only)

- **Checks:** only shown when `TransitionLogicUseMergedGeometries` is active: every leaf scene carries a merged geometry.
- **How to resolve:** click the listed scene once in the Scenes pane - taking it merges its geometry on demand - and run the check again, or move to the subscene workflow.

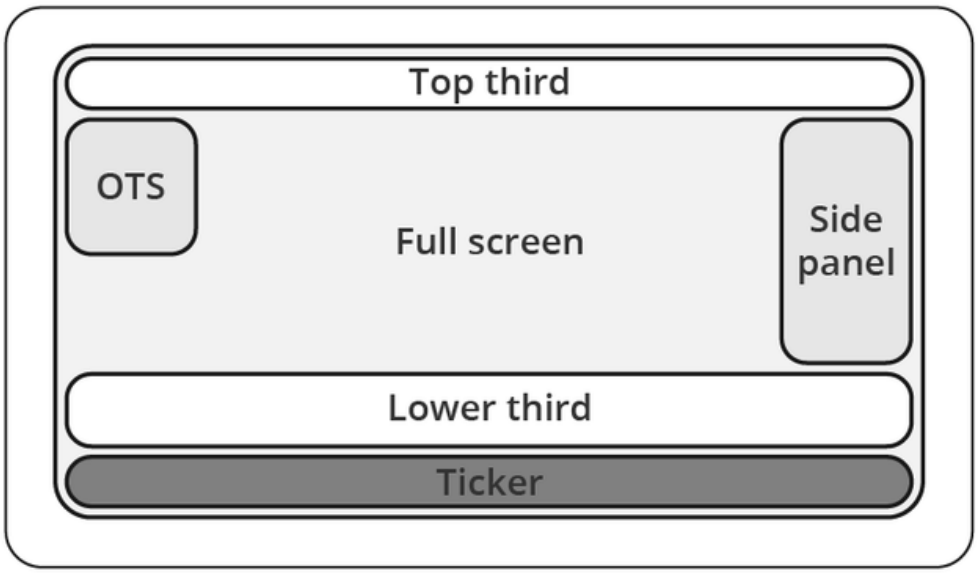
Which findings can I leave?

Finding	Can it be advisory?
Helper directors: none present	Not reported at all; valid.
Transitions without animation	Yes, if the hard cut is intended.
Layers without an Out scene	Yes, for layers that stay on air permanently.
Toggle keyframes out of step	Yes, if the layer takes correctly and you do not mind that Logicmaker edits normalize it.
Directors not named after their layer	**No: fix it, takes by name depend on it.**
Direct transitions between all states	**No: some state changes cannot animate.**
Subcontainers (,A ,B)	**No: nothing can be shown.**
Scene identifiers	**No: those pages cannot be taken.**

Copyright © 2026 Vizrt

20.7 Toggle Layer

The Toggle plug-in is used to establish a Toggle-layer in the master scene and acts as a placeholder for the contents of one or more graphical items. This toggle is used to establish the location of a placeholder for the insertion point of the object scene into the master scene. In other words, a Toggle-layer is actually a combination of this toggle placeholder and a linked **Director** on the **Stage**. The toggle container, together with a director of the same name, is what defines a layer.



Implementing multiple Toggle-layers creates the ability to place graphical items into several logical placeholders simultaneously, allowing multiple "layers" of graphics to be shown on-screen at the same time.

Note: Layer, in Transition Logic terms, does not lend itself to how the term layer is understood with most 2D applications. In Viz Artist, Toggle-layers, or more commonly just layers, refer to any segment of a graphics package which requires independent control.

A toggle-layer needs to be created for anything in the graphics package which requires independent control. Layers may or may not be allowed to be on-screen simultaneously. All toggle-layers reside in a common scene known as the master scene. This master scene requires at least as many toggle-layers as there are independent items to be shown on screen at the same time, for example lower-third, over-the-shoulder, current time and temperature, etc. It may even be as complete as containing the additional toggle-layers needed to display the entire graphics package. Additional graphics often include full-screens, a breaking news banner, and a locator, to name a few.

In the example above, it would be possible to place the top-third, lower third, and a news ticker in the same toggle-layer. However, only one of those items could have been on-screen at any given time. If the news ticker and the lower-third need to be on-screen at the same time they have to be in different toggle-layers. Creating placeholders for logical layers of graphics is done using the Toggle plug-in, which lends its name to the term toggle-layer. Incidentally, since it usually would not be useful to have both the top-third and lower-third on screen at the same time, as their use would normally be one or the other, it could be beneficial to place both the top-third and lower-third in the same toggle-layer, depending on the design.

Copyright © 2026 Vizrt

Residing in different logical layers also means that their animations may affect each other. For example, having the lower third and news ticker On Air at the same time may for instance push the lower third further up the screen to make room for the news ticker.

See Also

- Master Scene

Copyright © 2026 Vizrt

20.8 State Transition Animation

As previously mentioned, the make-up of a toggle-layer in a Transition Logic scene is a relationship between the **Toggle** plug-in and a **Director** in the stage. Through the use of the Director's timelines, it is possible to animate between different variations in the design, called states. In order to make use of these states, each object scene refers to a specific toggle-layer and desired state.

20.8.1 Toggle Layer States



Animating between different toggle layer states is called state transition animation. The example above shows a lower third's backplate being animated from a narrow version to a wide version (two states).

20.8.2 Defining States

A toggle layer is required to have a default state named O. The O state is often referred to as the Out state as it is the state the toggle layer is taken to when the operator issues a Take Out command. This means that in most cases, this is also the state where the toggle layer's referring foreground scene is no longer visible on-screen.

In addition to the O state, at least one additional state is required to run animations when taking items to air. Any state that is not the O state is often referred to as an *in state*, or *visible* state, because it most often refers to the state when a referring object scene is shown on-screen.

Tip: The master scene should be saved with the play head at the beginning of the timeline, which is equal to the O state. This makes sure that the scene is properly initialized in the O state.

Use the following conventions when naming states:

- O states must be named in upper case only. This is a reserved state that is tied to the take-out command.
- It is recommended to name states according to the referring control scene's appearance or other distinguishing characteristic. For example, for a toggle-layer responsible for content displayed in the lower third of the screen that varies in size from large to small typically has two states named *LARGE* and *SMALL*. A graphic placed over the shoulder of a presenter, where the graphics should switch between the left and right side of the screen, would be named *LEFT* and *RIGHT*, or *OTS_LEFT* and *OTS_RIGHT*, with OTS being short for Over The Shoulder.
- To maintain consistency between designers, it is recommended to write all state names using upper case characters only. This also makes it simpler to implement the state change in later steps.

Copyright © 2026 Vizrt

O → SMALL

When taking an item which requires the SMALL state, the master scene animates from the O state to the SMALL state, once that item is taken On Air. When a take-out command is issued, the master scene then animates in reverse to the O state.

O → SMALL → O

Transitioning from a SMALL to an O state is achieved by using the animation from the O to the SMALL state. In such a case, the animation plays in reverse. If this is not required, you simply add another O state after the SMALL state, and create a new animation.

O → SMALL → LARGE → O

If the layer should transition from the small sized version to the large version, you simply add the state to the toggle-layer director, and create a transition animation for the layer going from `SMALL` to `LARGE`. This also requires that the `LARGE` state has an `O` state such that the graphics can transition and animate out from the `LARGE` state.

How GOTO_TRIO Works

When `GOTO_TRIO $FROM $TO` is issued, Viz Engine searches the layer director:

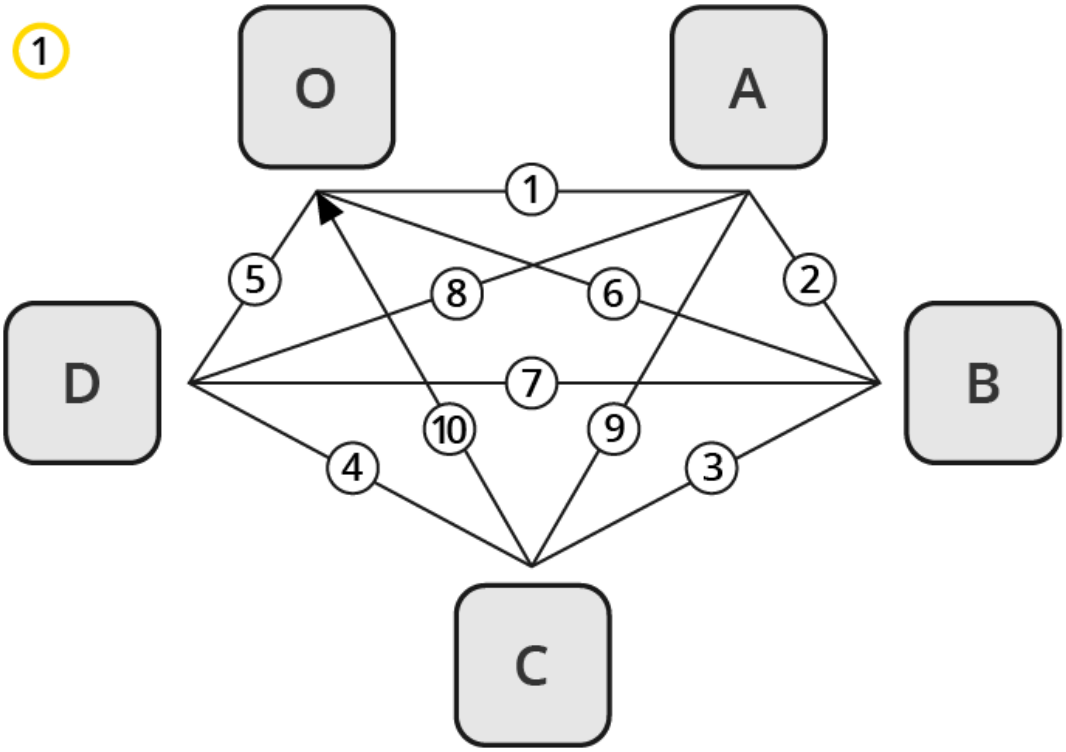
1. **Left to right:** Find `$FROM`, then look right for `$TO`. If found, play the animation forward between them.
2. **Backtrack:** If not found left-to-right, find `$TO` and look right for `$FROM`. If found, play that segment in **reverse**.

This means a single animation segment can serve both directions, `$O → $IN` plays forward, `$IN → $O` plays the same segment in reverse.

20.8.3 Identifying States

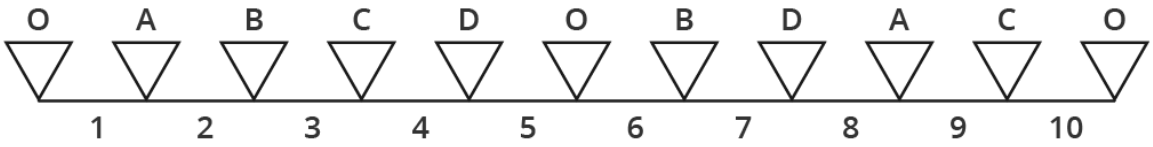
There are no hard limits to the number of states. However, increasing the number of states also increases the number of animations, and consequently the complexity of the scene. To decide the minimum number of animations required it is possible to do a Euler Walk (1).

Copyright © 2026 Vizrt



The Euler Walk, although not always possible, is defined as the optimal route taken to visit each edge exactly once. However, the theory can be applied to Transition Logic to identify the minimum required states needed to connect each state to the others exactly once.

Straightening the Euler walk out onto the layer's director shows the required transition animations:



Minimum Animation Count

The Euler Walk determines the minimum number of animation segments needed to connect all states:

States	Min Animations	Example Director Layout
2	1	`$O ——— $IN ——— $O`

Copyright © 2026 Vizrt

States	Min Animations	Example Director Layout
3	3	`$0 ——— $SMALL ——— $LARGE ——— $0 ——— $LARGE ——— $SMALL ——— $0`
4	6	(complexity increases significantly)
5	10	(consider splitting into separate layers)

Practical guideline: Keep layers to a maximum of two to three states. If a layer needs four or more states, consider whether the design can be split into multiple layers, each with fewer states.

Note: When performing transition animations from one state to the other Viz Artist/Engine searches from left to right on the layer's director for the correct combination of states. If the operator wishes to go from state A to B, Viz Artist/Engine traverses the layer's director for state A and looks to the right and see if it finds state B. If a match is found, Viz Artist/Engine performs the transition animation.

Looking at the above example director, it becomes clear that not all combinations can be found by traversing from left to right. If a change from state A to D is not found, Viz Engine first traverses from left to right, then starts to backtrack and traverse from right to left looking for a D from A animation.

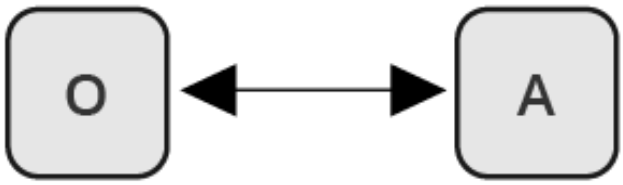
When the minimum required animations have been discovered, it is time to add extra animations required by the overall design. For example, it might be desirable to add a transition from state B to O, to animate the layer out from state B in a different way than reversing the in animation. As an example, think of a lower third graphics animating in from the left, stopping, then animating out to the bottom. Adding new out states quickly increases the complexity of a layer. As all states are required to be connected to each other, it would not be advisable to simply make an O state in place of the B to C animation.

Detailed below are the minimum required number of transition animations for a graphics layer with a given number of states:

- **One State:** Animations are not possible.

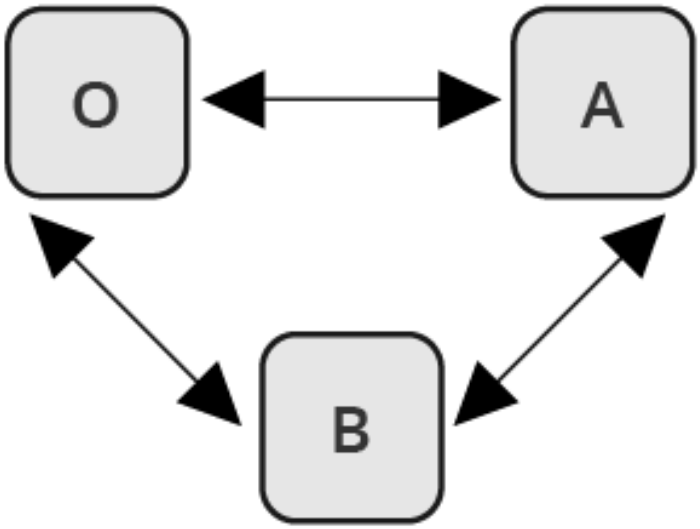


- **Two States:** Requires a minimum of one animation.

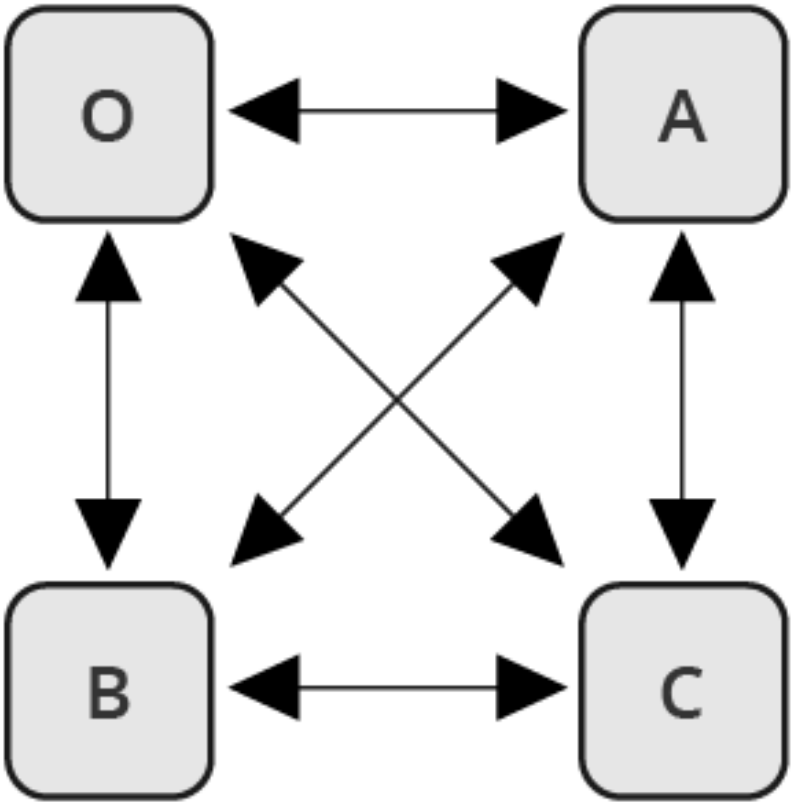


- **Three States:** Requires a minimum of three animations.

Copyright © 2026 Vizrt

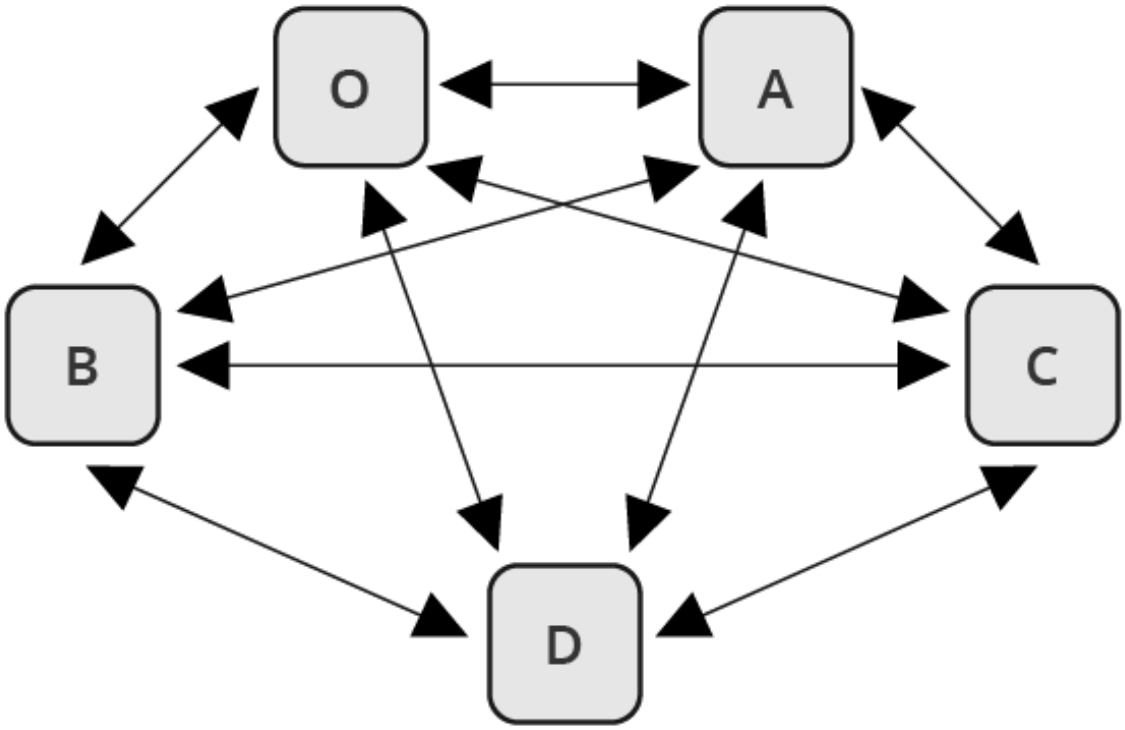


- **Four States:** Requires a minimum of six animations.

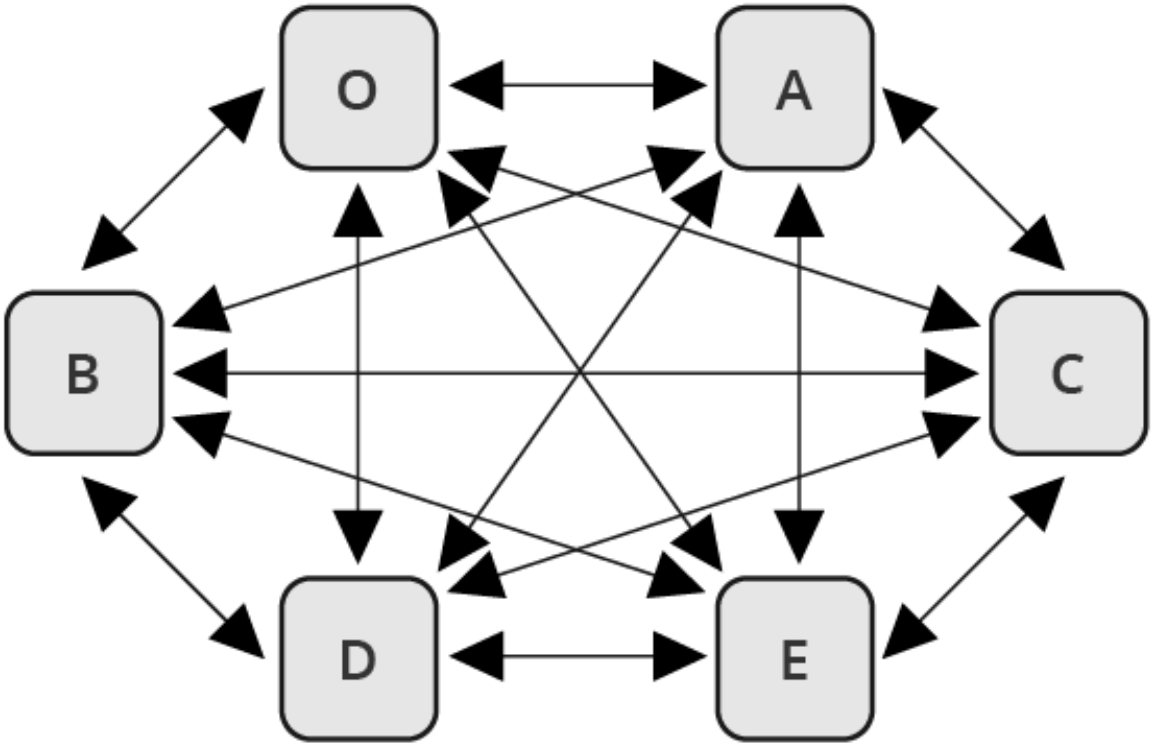


- **Five States:** Requires a minimum of ten animations.

Copyright © 2026 Vizrt



- **Six States:** Requires a minimum of 15 animations.



Copyright © 2026 Vizrt

Adding Custom Out Animations

By default, going from `$IN` to `$O` simply reverses the in-animation. To create a different out animation (for example, slide in from left, drop out to bottom), add a second `$O` stop point after the visible state:

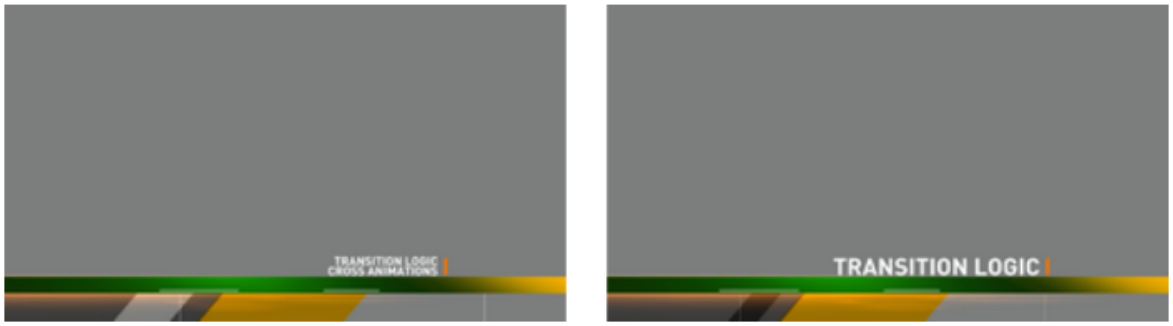
```
Director "BannerLyr"
──$O──────────────$LOWERTHIRD──────────────$O──
  | ← in animation → | ← out animation → |
  (slide from left)  (drop to bottom)
```

After adding new stop points, always re-run **Default Keyframes** on the Toggle plug-in (delete the old ones first).

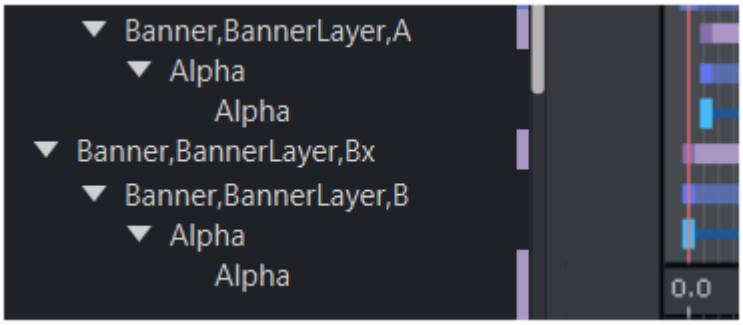
Copyright © 2026 Vizrt

20.9 Cross Animation

Cross animations are used in Transition Logic scene design to animate one object in and another out. The example below shows a lower third that has an alpha cross fade fading from a two lines of text to a one line of text.



Cross animation directors are added by the Toggle plug-in. The directors are always tagged with the letters *AX* and *BX*. *AX* and *BX* holds the cross animation and effect used to animate from one object to another. The default cross animation is a one second cross fade of the alpha value.



Animation and effect length can be changed by adjusting the director's animation and effect(s) on the stage. In addition it is also possible to remove or add effects (for example, material, position and scaling).

20.9.1 Custom Transitions

The default AX/BX cross-fade is a one second alpha dissolve. You can replace it with any animation:

Transition Style	What to Animate on AX/BX
Slide	Position X or Y (e.g. slide from off-screen left to center)
Scale	Scale from 0 to 1 (grow in) or 1 to 0 (shrink out)
3D Flip	Rotate on Y-axis from -90˚ to 0˚ (flip in)

Copyright © 2026 Vizrt

Transition Style	What to Animate on AX/BX
Wipe	Animate a crop mask or stencil
Dissolve + Blur	Combine alpha with a Blur plugin (blur out, sharpen in)

Important: AX and BX must ALWAYS be identical! Toggle may load content on either the A or B side. If AX and BX have different animations, the transition looks inconsistent depending on which placeholder is active. Always duplicate changes from AX to BX (or vice versa).

Editing the Cross-Fade

1. Open the Stage and expand the layer director (for example, `BannerLyr`).
2. Select `BannerLyr,AX`. This is the A-side cross-fade director.
3. Delete the default alpha keyframes.
4. Add your custom keyframes (position, scale, rotation, blur, etc.).
5. Copy the exact same keyframes to `BannerLyr,BX`.
6. **Do not edit** `BannerLyr,A0` or `BannerLyr,B0`, these are auto-managed by the Toggle plug-in for object scene animations.

Copyright © 2026 Vizrt

20.10 Master Scene

The master scene serves several important functions. Most importantly it is the controlling part of a Transition Logic scene, and is essentially what makes standalone scene design different from Transition Logic scene design.

A Transition Logic master scene is:

- Constantly On Air, and is used for controlling state transition animations on backplate graphics in combination with the cross animations of object scenes, such as toggling from one line to two lines of text.
- Able to replace a layer's referring object scene without taking other layers off air.
- Able to have graphical items such as backplates, which separates the background for object scenes.

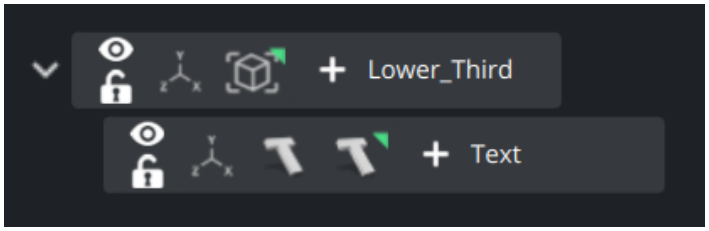
Having the master scene constantly On Air means that when controlling the toggle layers, it can smoothly animate the backplate graphics from one size to another, providing an artifact-free backplate that fits the changing requirements of the object scenes. With a geometry holding one to two lines of text, the text as a bare minimum has a built-in cross-fade animation. Additionally, as the background scene is already On Air, the transition animations on the background items have no artifacts, or jittering, caused by graphics being loaded.

Use Logicmaker to setup your master scene, see Logicmaker#Creating-a-package

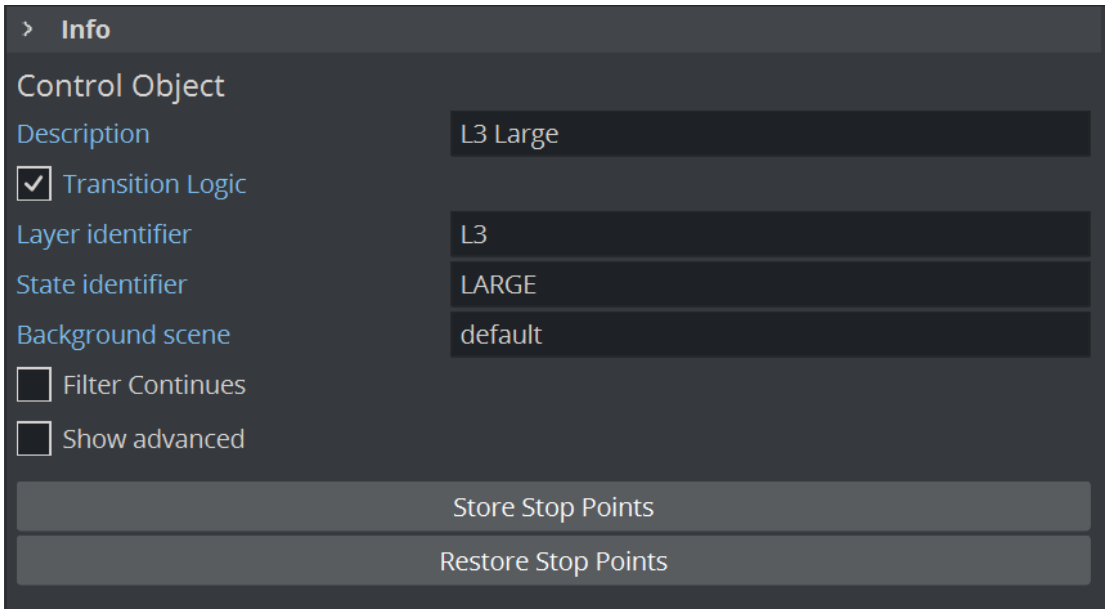
Copyright © 2026 Vizrt

20.11 Object/Leaf Scene

Transition Logic object scenes are similar to traditional standalone scenes that contain graphical items such as geometrical shapes and textures (for example, images or video clips), but are different in that they are capable of affecting the state of the master scene's toggle layer, and typically only contain the variable data of a graphic such as the changeable text, while the design items are located in the master scene.



For object scenes to have an effect on the master scene's toggle layer, it has to refer to the master scene, the layer and the layer's state. This is obtained by adding the Control Object plug-in to the object scene's scene tree at root level.



When Control Object is added to an object scene, it must be Transition Logic enabled. This enables the object scene to set layer, state, and master scene parameters. When an object scene is taken On Air, these parameters are sent as commands to Viz requesting the master scene to change the toggle layer's current state to the new state the object scene is referring to. If the object scene's Control Object parameters are incorrect, Viz is not able to trigger the correct state transition animation, and the object scene is not able to be animated in.

Control Object also keeps track of other control plug-ins' properties that, when exposed, allow the operator to change their parameters (for example, text size, color, geometry, texture, video clip and so on).

Copyright © 2026 Vizrt

IMPORTANT! The ControlObject fields are case sensitive.

Use Logicmaker to setup your Leaf scenes, see Logicmaker#Creating-a-package

Copyright © 2026 Vizrt

20.12 Troubleshooting

Debugging Transition Logic scene is a complex task, we recommend to use the Health Check in Logicmaker to perform most common basic checks, however additional issues are listed here:

Symptom	Cause	Fix
Object scene doesn't animate in	ControlObject's *Layer identifier* or *State identifier* doesn't match the master scene (case-sensitive mismatch)	Verify names match exactly. Check capitalization: `BannerLyr` ≠ `bannerlyr` ≠ `Bannerlyr`
Backplate visible on first take	Master scene was saved with a director not in $O state	Seek all directors to $O / frame 0, then re-save the master scene
Cross-fade looks different on alternate takes	AX and BX directors have different animations	Ensure AX and BX are identical. Delete both, re-create, and copy keyframes
Object scene loads but data doesn't appear	Data pushed to `$current` instead of `$other`, or wrong field_id	Always push data to `$other$object` before switch. Verify field_id values
Toggle generates no sub-directors	Default Keyframes wasn't clicked, or the director has fewer than two stop points	Add at least $O + one visible state, then click Default Keyframes
Scene frozen after switch (subscene mode)	Missing `continue INVOKE` after switch in subscene mode	Add `Toggle*continue INVOKE` after `Toggle*switch INVOKE`
ControlList shows fewer rows than expected	Not enough row containers pre-created in the scene tree	Pre-create sub-ControlObject containers up to your maximum row count
Stop points lost after object-pool merge	Stop points were not stored before saving	Click *Store Stop Points* on ControlObject before saving the object scene
Master scene named "background" causes issues	The name `background` triggers legacy mode in MSE	Rename to `default` or any other name

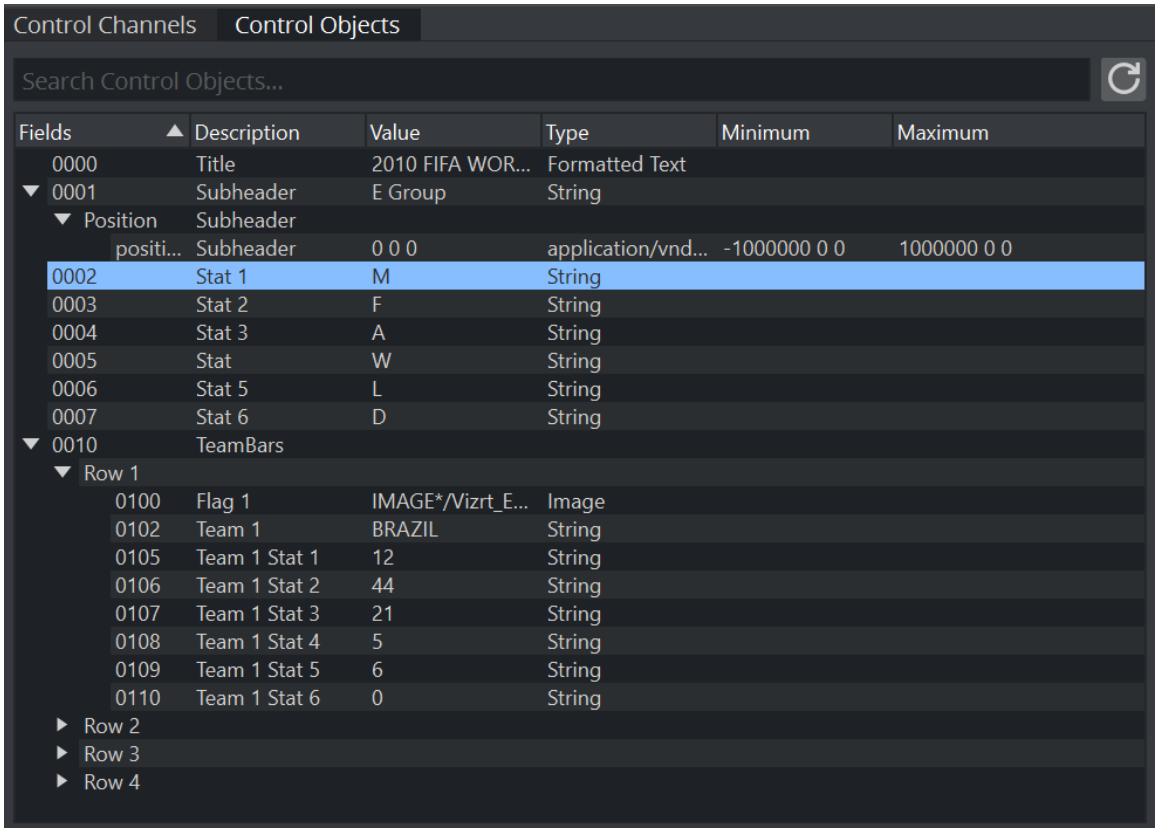
Copyright © 2026 Vizrt

Symptom	Cause	Fix
GOTO_TRIO doesn't find state pair	Stop points not connected on the timeline (missing Euler walk segment)	Verify the director has a path from $FROM to $TO — either left-to-right or by backtracking

Copyright © 2026 Vizrt

21 External Control

Parameters in a Scene can be exposed for use with external controls, meaning these exposed parameters can be modified by an external control application, such as Viz Libero, Viz Trio or Viz Pilot.



The Viz Artist GUI provides two options to expose parameters and controls:

- **Control Channels:** Provides a convenient way to expose parameters from the Properties panel in a Scene.
- **Control Objects:** Shows exposed parameters when a Control plug-in is used (for use with Viz Trio only).

This section contains information on the following topics:

- Control Channels
- Control Objects

Copyright © 2026 Vizrt

21.1 Control Channels

Parameters in a Scene can be exposed for use with external controls, meaning these exposed parameters can be modified by an external control application, such as Viz Libero, Viz Trio or Viz Pilot. To expose parameters, the parameters can be dragged to the Control Channel panel and configured.

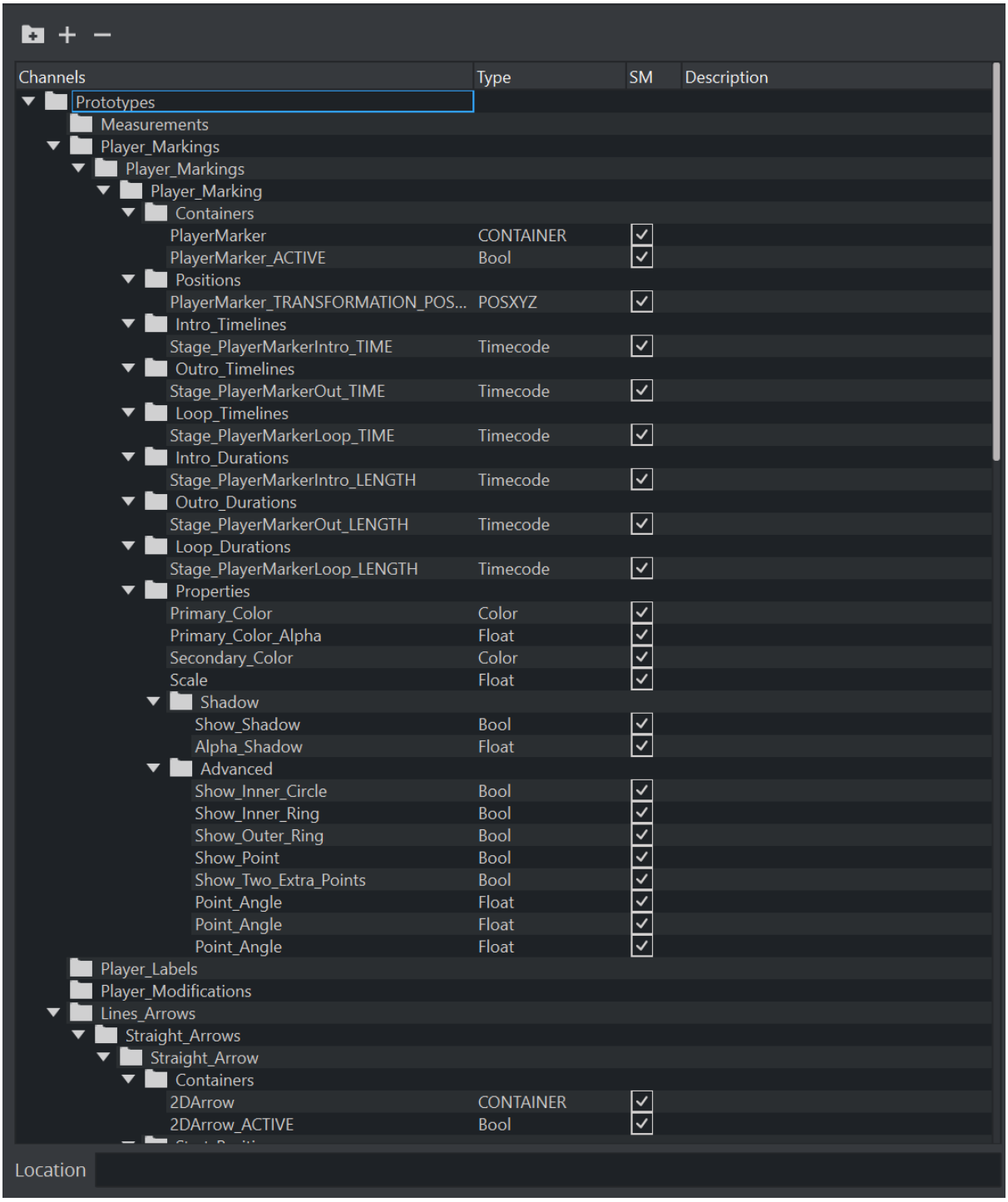
Although parameters can be set through a Command Interface, the Control Channels offer three advantages:

- Only parameters which should or can be modified, are exposed in certain external control applications (which use control channels).
- Instead of a long command string (as used in a Command Interface), a control channel name points to the parameter location, where **@** in the command location is the indicator for a control channel name.
- Multiple Control Channels can have the same name, allowing them all to be updated in a single operation.

Most parameters can be added to the Control Channels. Properties that can be added to the Controls Channels are:

- **Properties Panel:** Container, Camera, Light and Scene Settings tabs.
- **Scene Tree:** Containers, Geometries, Materials.
- **Stage:** Editors and parts of Editors (Director parameters, Channel Editor, Key Frame Editor, Stops and Tags, Action Key Frames).
  Navigating scenes with extensive use of Control Channels is made easy, as Viz Artist automatically selects all Control Channels belonging to a container when the container is selected. Correspondingly, selecting a single Control Channel also selects the container it belongs to.

Copyright © 2026 Vizrt



Control Channels can also be grouped. This provides a convenient way to group a series of parameters which can be controlled from an external source.

Example: Click on the **Container tab > Transformation Editor**. Drag the parameter Position X of an Object to the Control Channels panel. This exposes Position X of that object.

Copyright © 2026 Vizrt

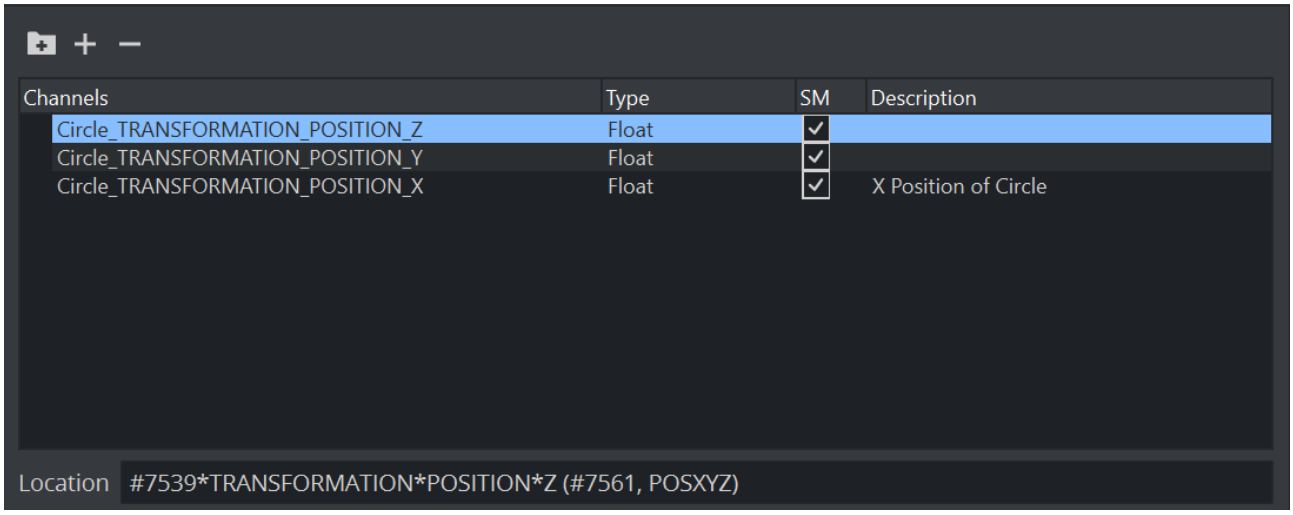
Example: In the **Properties** panel, click on the **Camera** tab and drag a camera's **Position Y** parameter to the Control Channels panel. This exposes Position Y of that camera.

Example: In the **Stage**, click on the header of a Key Frame editor (in-line with the) and drag it to the Control Channels panel. This gives access to all parameters contained in the Key Frame editor. To add control channels from the Stage, Control Channels need to be floated.

This section contains the following procedures:

- Control Channels Properties
- Control Channels Procedures
  - To Create a Control Channel for x Position
  - To Control Channels via Shared Memory
  - To Remove Exposed Parameters

21.1.1 Control Channels Properties



- **:** Click to add a new group
- **:** Opens or collapses the **Control Channels tree**
- **Channels, Type, SM, Description:** Exposed parameter details.
  If required, click in the **SM** box to synchronize the Channel values with the corresponding key-value pair in the Scene Shared Memory Map.
- **Description:** Adds a description for the exposed parameter.
- **Location:** Shows command location of the command.

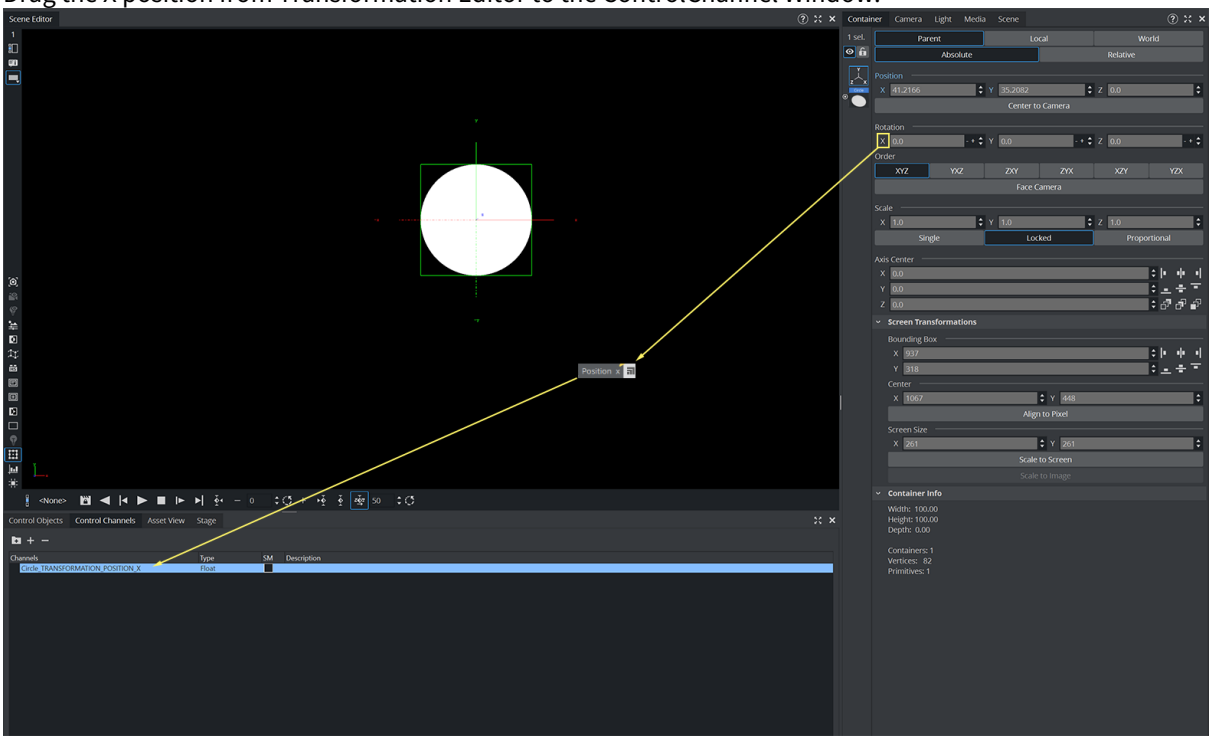
Info: A selection of multiple entries from within the Control Channel window is currently not possible as it is synched with the selection of the Scene tree. Vice versa, if you select multiple containers in the scene tree, the corresponding Control Channels are selected.

Copyright © 2026 Vizrt

21.1.2 Control Channels Procedures

To Create a Control Channel for x Position

1. Open **Control Channel** window.
2. Drag the X position from Transformation Editor to the ControlChannel Window:



3. Enter a unique name for this channel (for example, *Circle_TRANSFORMATION_POSITION_X*).
4. The Control Channel creates a new Channel of type float.
5. In addition, you can double click in the Description area to give your channel a meaningful name.

Instead of providing the full tree in a command, you can now access this channel with:

```
MAIN_SCENE*TREE*@Circle_TRANSFORMATION_POSITION_X SET -20.0
```

To Control Channels via Shared Memory

Each control channel can also be accessed via the Shared Memory Interface. This can be, for example, by a external command or when using the REST interface by sending a POST request.

In the example above, we created a control channel for the x Position of our circle named *Circle_TRANSFORMATION_POSITION_X.* Therefore, if we now modify the local scene shared memory variable *Circle_TRANSFORMATION_POSITION_X,* we can control the x position of our circle.
Example:

Copyright © 2026 Vizrt

```
MAIN_SCENE*MAP SET_DOUBLE_ELEMENT Circle_TRANSFORMATION_POSITION_X 40.0
```

This command changes the x position of our circle and set it to 40.0.

Note: The Shared Memory key syntax for grouped control channels is *<groupname>.<channelname>*, for example, *MyGroup.Circle_TRANSFORMATION_POSITION_X*.

To Remove Exposed Parameters

Drag an exposed parameter, or multi selected exposed parameters, to the Trash Can.

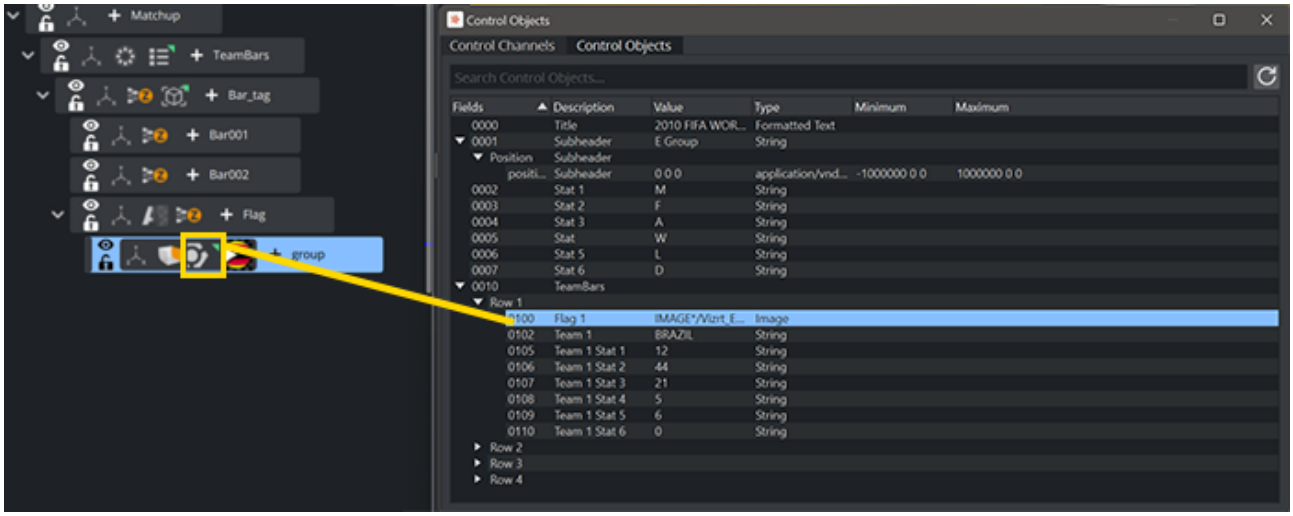
See Also

- Control Objects
- External Control

Copyright © 2026 Vizrt

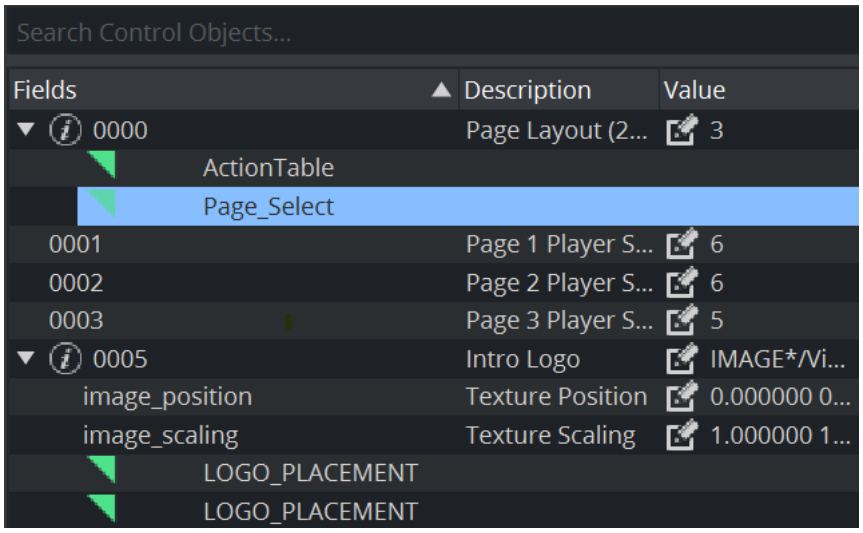
21.2 Control Objects

The Control Objects panel is used to show the control parameters for control applications like Viz Trio or Viz Pilot.



21.2.1 Control Object Parameters

Control Objects are defined by adding certain plug-ins, so called Control Objects to containers. These plug-ins expose certain functionality to Control Applications like Viz Trio or Viz Pilot. Every Control plug-in must have a Control Object plug-in in the upper hierarchy of the scene tree.

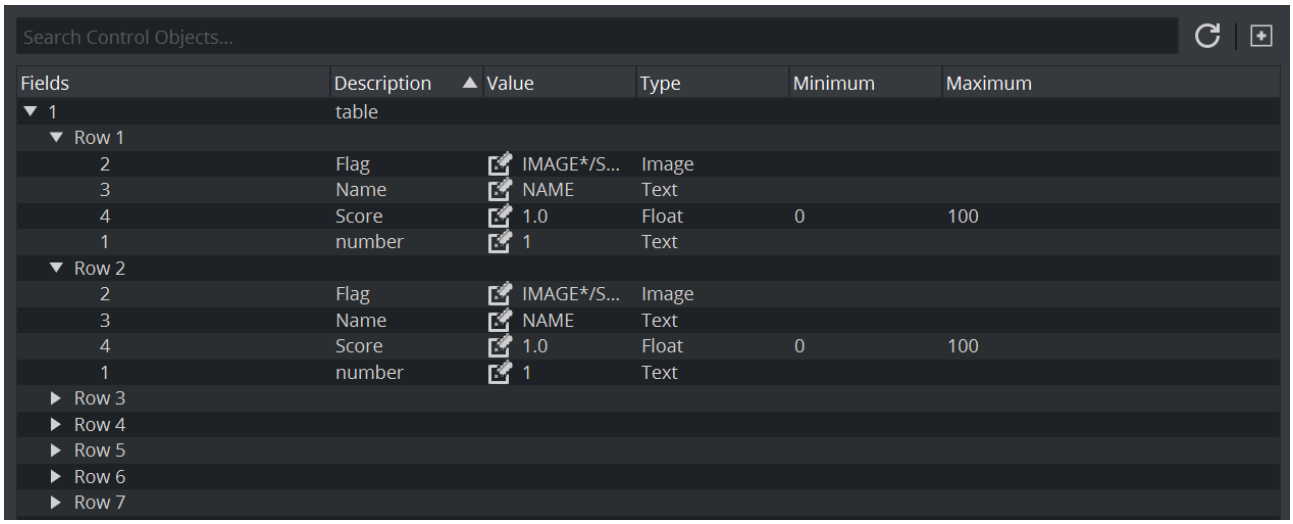


Information: The green triangle indicates that this property has the same field identifier used on more plug-ins. Clicking on the entry automatically jumps to the related container.

Copyright © 2026 Vizrt

- **Fields:** A unique field identifier of this property. This is shown in the control applications. The following characters are allowed to guarantee compatibility with other control applications:
  - - , _ , .
  - 0 - 9
  - a - z
  - A - Z
- **Description:** Enter a description in the plug-in Properties panel.
- **Value:** The editable plug-in value. This value is then overwritten by the operator in the control room.
- **Type:** The type of Control Object.
- **Minimum and Maximum values:** Used for Integer and Float values.

ControlList structures are shown as expandable groups within the editor:



21.2.2 Search

The new Control Object Editor allows to search for any property shown within the editor. This makes it easier to quickly identify and find control objects within your scene.

Use the Search option to quickly lookup Field Identifiers, Names or Types.

Examples

Search Term	Search Result
013	Shows any Field identifier like "%013%"
Image	Any Control Channel of type "image"
triplet	Any Control Channel of type "application/vnd.vizrt.triplet"

Copyright © 2026 Vizrt

Info: The ControlObject Editor currently does not list exposed objects from GFX Channels, Super Channels and Scene-in-Scene.

See Also

- Control Channels
- External Control

Copyright © 2026 Vizrt

22 Scripting

This chapter provides a brief introduction to creating and running Viz scripts; how to create reusable plug-ins from scripts; and lastly provides a few example scripts that help develop an understanding of the Viz script functionality. For more in-depth information, see the built-in API and Scripting documentation located in the Viz installation folder.

Note: Multitouch and 6DOF events are not yet implemented in Viz Engine Render Pipeline.

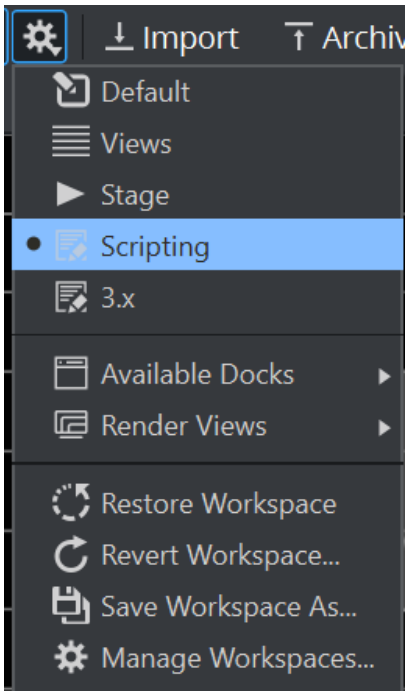
This section contains information on the following topics:

- Script Editor
- Create and Run Scripts
- Create Script-based Plug-ins
- Integration Guide

Copyright © 2026 Vizrt

22.1 Script Editor

This section explains how to work with the Script Editor. To switch to the scripting perspective, select **Scripting** from the workspace menu.



Viz Artist can be scripted on two levels - using Scene Scripts or Container Scripts. In addition, you can write Viz Engine Shaders within the scripting environment. The following chapters mainly refer to Viz Scripting. For information on how to write Viz Engine Shaders, see Viz Engine Shaders.

Before adding scripts and code, the designer needs to think about where the logic should reside.

This section contains the following topics:

- Scene Scripts
- Script Editor Properties
    - Editing Area
    - Function, Callbacks and Symbols
    - Search and Replace
        - Example
    - Scripting Reference
    - Debugging

22.1.1 Scene Scripts

- If there are functions that act like an interface to the scene, it's a good idea to add this logic as a scene script.

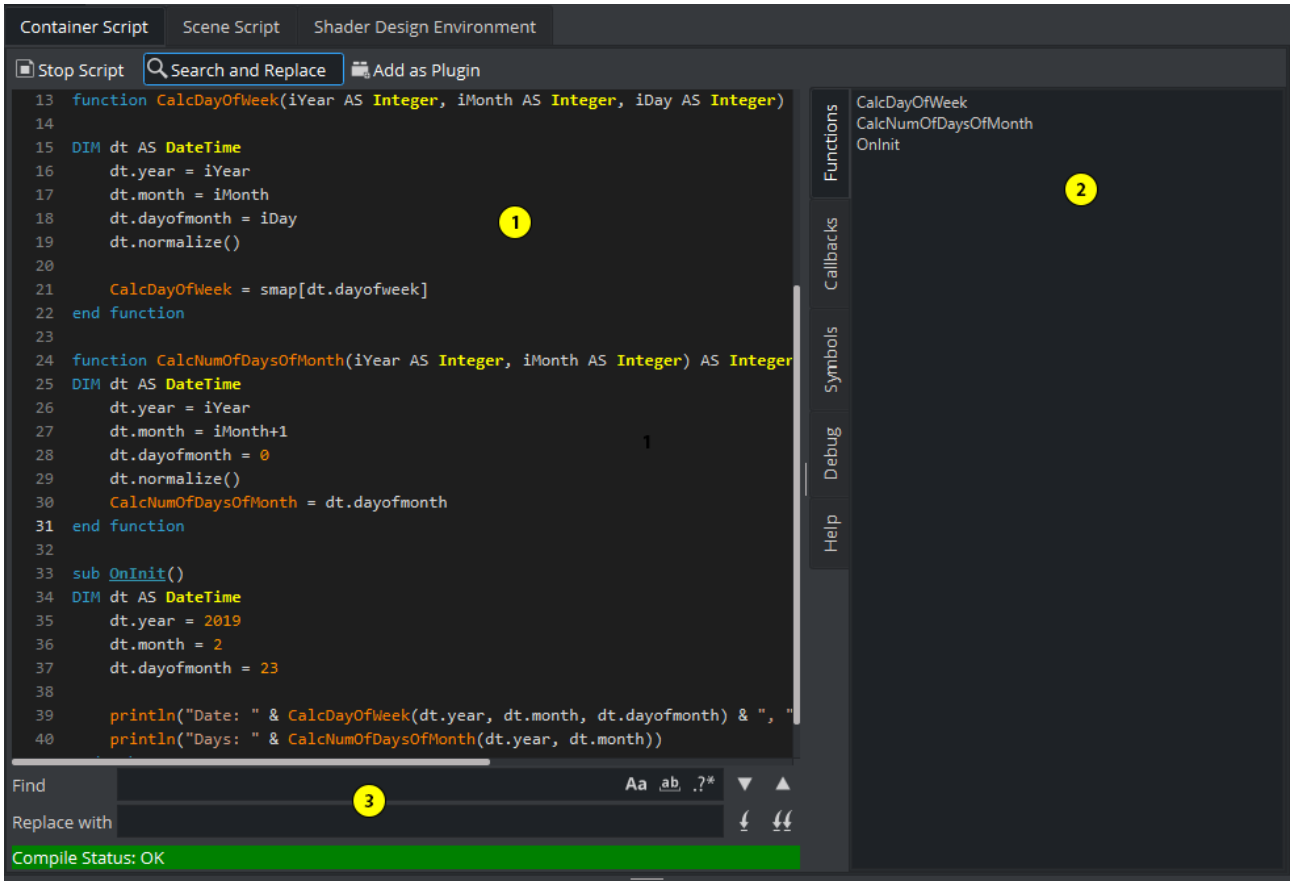
Copyright © 2026 Vizrt

- If functions or variables are used throughout several other scripts within the scene, the scene script is the proper location for those.

On the other hand, if the script code is container-specific or planned for a container prototype-instance(s) approach, the script code should be hosted in a container script.

22.1.2 Script Editor Properties

Although the **Scene Script Editor** and the **Container Script Editor** have the same properties and functionality, a script created in one editor can not be placed in the other editor's folder. This section provides details on the Script Editor's properties and context menu.



The Script editor is divided into the following sections:

1. Editing Area
2. Function, Callbacks, Symbols, Debugging and the Scripting Reference
3. Search and Replace

Editing Area

Select which type of script you wish to edit by clicking the tabs at the top of the menu:

Container Script: Container scripts reside on containers and are mainly responsible to add logic onto single container(s).
Scene Script: All global functionality or logic that apply to the whole scene can be defined as a Scene script.
Shader Design Environment: Introduced in Viz Engine 4. See Viz Engine Shaders for more information.

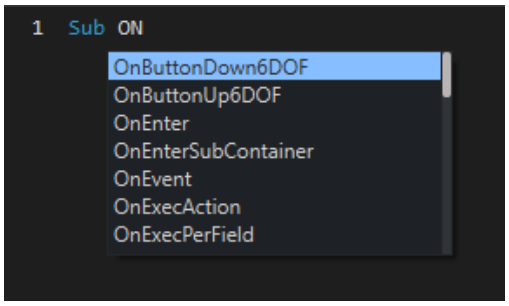
A script can have three states: running, editing or error mode:



To toggle between run mode and edit mode, click on the Compile and Run respective the Stop Script button. A script that has been successfully compiled and runs without any errors, can be saved as a Script plug-in. This is done by clicking on the Add as Plugin button. For more information, please see chapter Create Script-based Plug-ins. The status of the compile status is shown under the editor area: Compile Status: OK

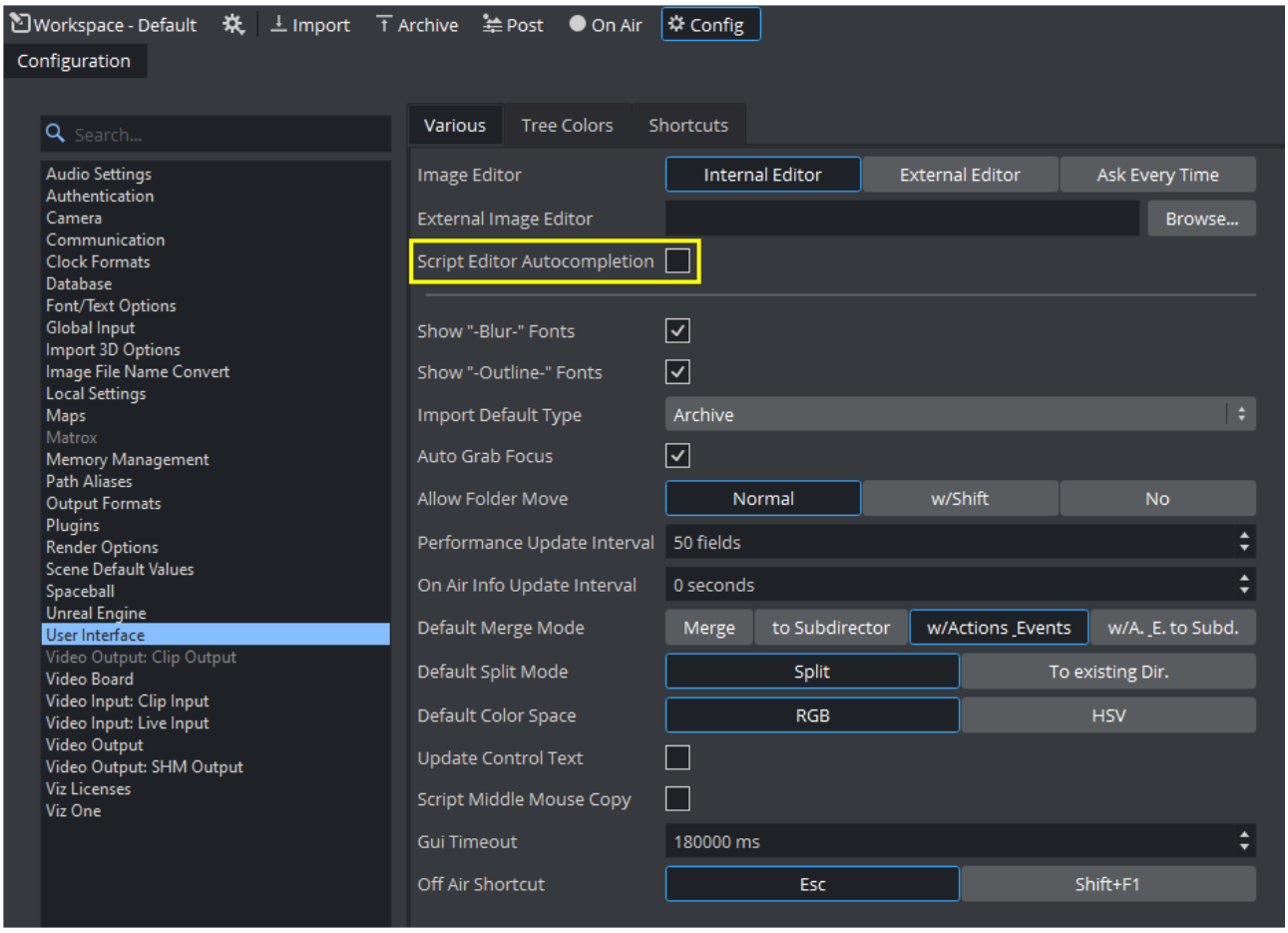
To toggle the search and replace window click on the Search and Replace button. Within the editing area, **F3** repeats the last search and **F4** searches for the current word under the cursor. Continue pressing **F3**/**F4** to search for the next occurrence.

To start editing your script, start by typing one of the keywords, for instance *dim* or *Sub*. The Editor automatically offers you a list of possibilities and auto-completes your selection. Example: Start adding a new sub by typing *Sub O*. A list of valid options immediately shows up:



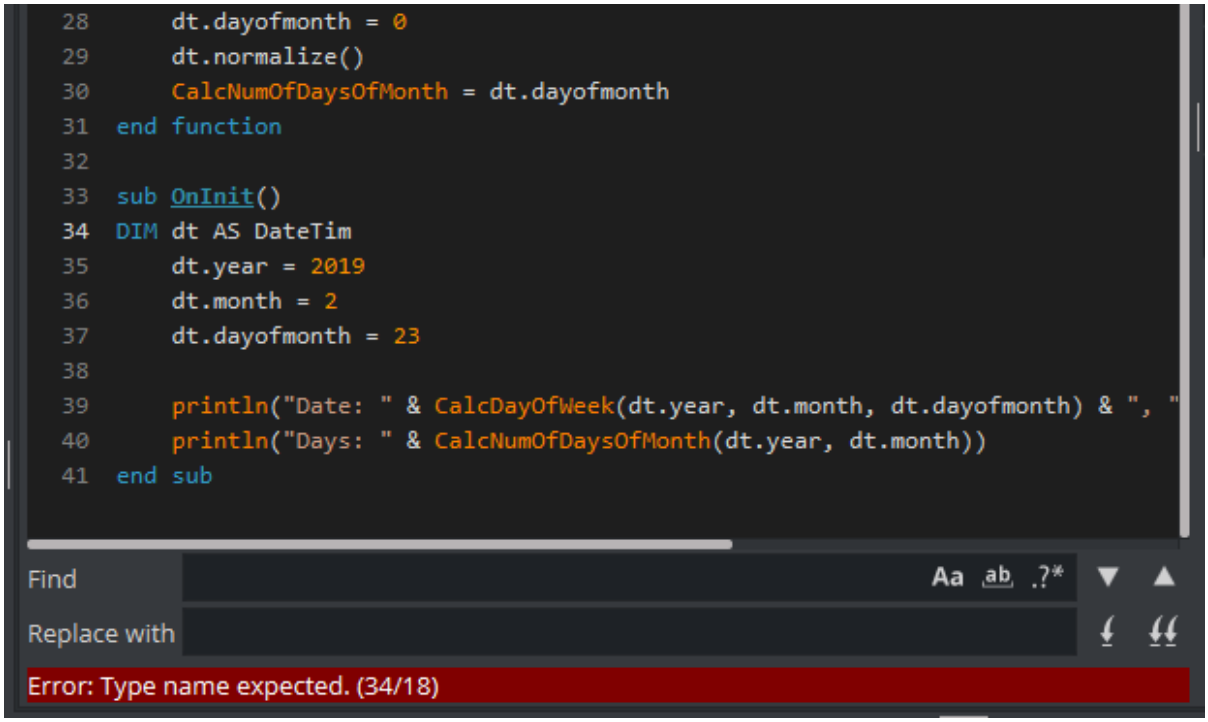
Autocompletion can be enabled/disabled under **Config > User Interface > Script Editor Autocompletion**.

Copyright © 2026 Vizrt

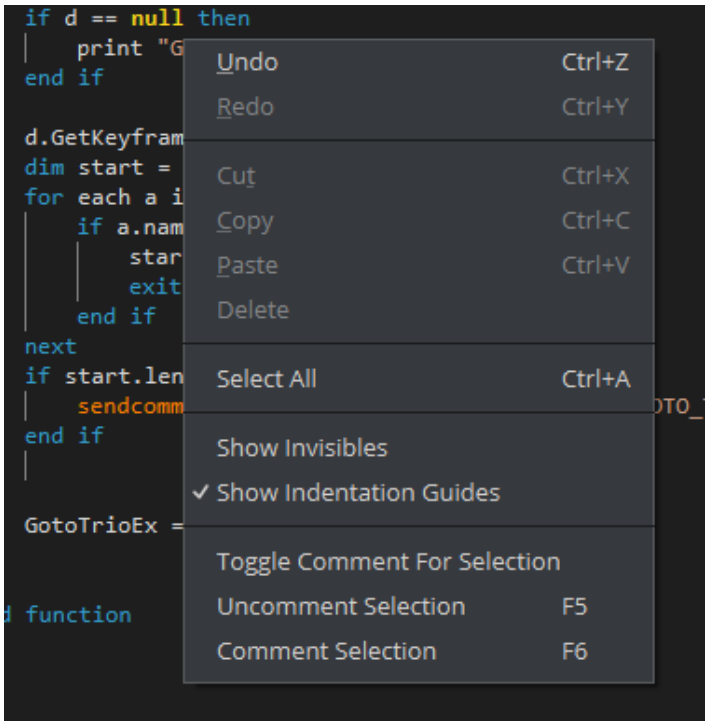


The script editor automatically colorizes keywords for a better reading. Once an error occurs (when pressing the **Compile & Run**), the editor shows an error and automatically jumps to the line the (first) error occurs.

Copyright © 2026 Vizrt



Right clicking inside the Editor opens the Context menu:

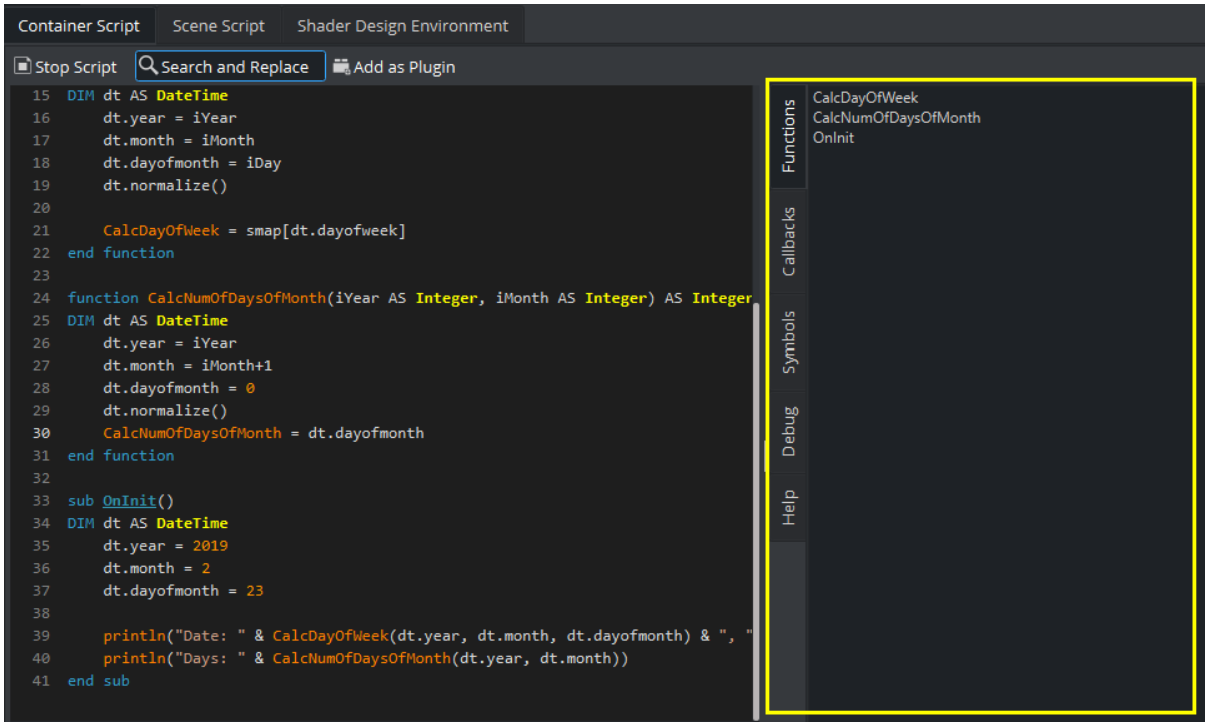


Changing the font size: The font size of the script can be changed using **CTRL +** mouse wheel.

Copyright © 2026 Vizrt

Function, Callbacks and Symbols

The function tab shows a list of pre-defined functions within your script, including the built-in Callbacks and self-defined functions. Clicking on a function automatically moves the view of your editor to this selected function:



The callback tab shows a list of all built-in callbacks Viz is using. For example the *OnInit* is called whenever the script get initialized. Callbacks written in Bold are used by the current script. Click on one of the entries to directly jump to the location within your script.

> **Note:** Callbacks differ between Scene and Container Scripts.

The **Symbols** tab lists the names the scripting language provides. The letter in front of each name shows its kind:

Icon	Kind	Examples
T	Data type	Container, String, Array
F	Global function	Println, Random
V	Global constant	TX_QUAL_MIPMAP, True
S	Procedure of a data type	Container.FindSubContainer

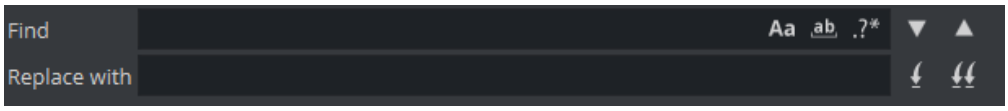
Properties of data types, for example *Container.Name*, are not in the list. They are listed on the page of the data type in the Scripting Reference. All global constants with their values are on the *Constants and Enumerations* page.

Copyright © 2026 Vizrt

If **Sync with Editor** is checked in the Scripting Reference, clicking a name shows its page there.

Select a name and press *Ctrl+C* to copy it, then paste it into the script.

Search and Replace



The search functionality allows you to search for terms or even replace them automatically.

Add the term you want to search for and press either:

- **:** Shows the next or the previous occurrence of your search term.
- **:** Replaces a single or all occurrences of your search term with your replace term when a replace term has been entered.

To better define your search criteria, specify your options on the right:

- **Aa:** Switches between upper and lowercase search.
- **ab:** Searches for whole words (separated by a white space character) or by a sequence.
- **.?*:** Allows to search by a Regular Expression, based on Perl's regexp language.

Example

To match strings containing 'test' or 'my' or 'search' but only match whole words (not 'searches'), enter `\\b(test|my|search)\\b` in the Find field.

> **Tip:** You can also search for the occurrence of a word within scripts from the Scene Tree. For example, to show all scripts that contain an "onExecPerField" procedure.

Scripting Reference

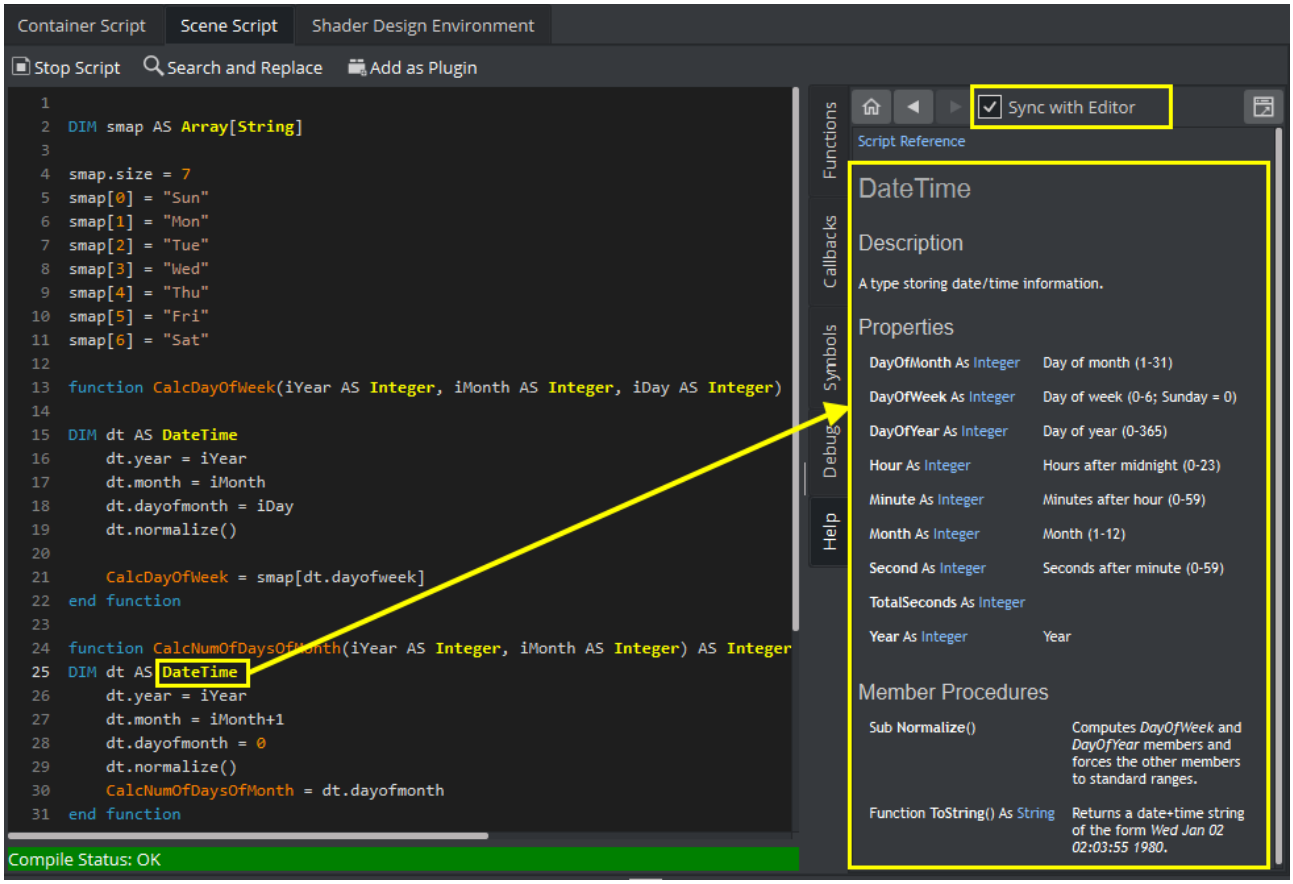
The Scripting documentation is usually located on your hard drive in *C:\Program Files\Vizrt\VizEngine\Documentation\viz-script-doc-webhelp.* This view allows you to browse the documentation without opening a browser window and switch between Viz Artist and the documentation.
Lets you open the scripting reference in a browser window.

Navigation buttons:

Navigate through the documentation by clicking on the home button to return to the main page or use the back and forward buttons to browse through your navigation history.

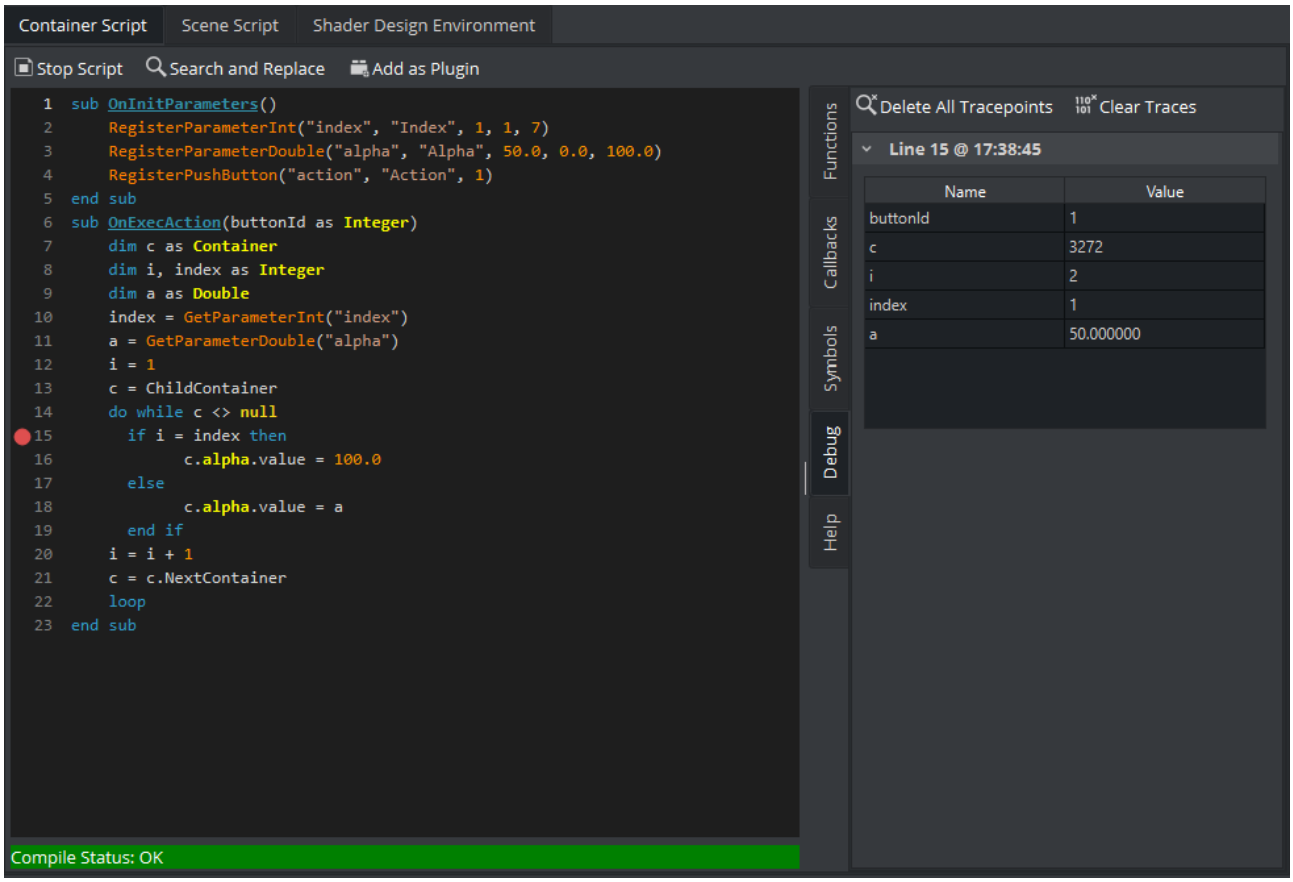
> **Tip:** Sync your documentation view with the script editor. Click on a keyword within the editor and the Scripting Reference automatically shows you the correct chapter within the scripting documentation. For example, if you click on *"scaling"*, the Script reference automatically shows you the chapter about scaling, as in the screenshot below:

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

Debugging



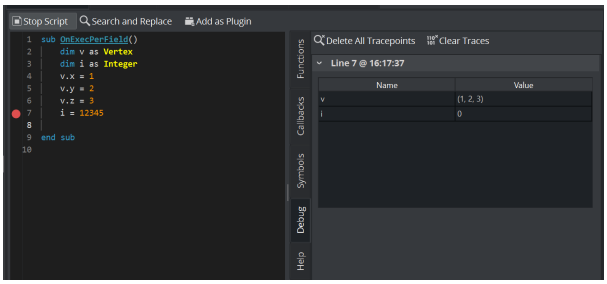
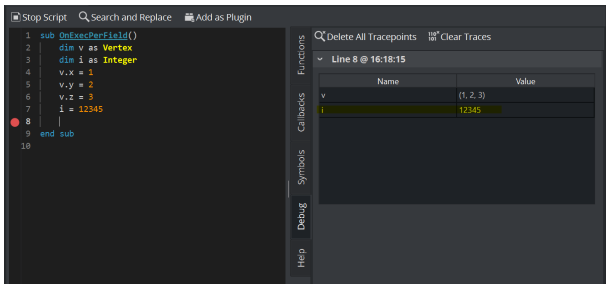
The script editor supports setting tracepoints which provide insights into the state of the variables at a certain point of the execution.

Whenever the program execution hits a line with a tracepoint set, a snapshot of all variables calculated up to this point and can be inspected.

> **Note:** In contrast to breakpoints, the execution does not stop when reaching a tracepoint.

Please be aware that only variables that have been calculated **up to this line** are shown correctly. Put your tracepoint onto the next line to get a correct result:

Copyright © 2026 Vizrt

Incorrect	Correct
	

A tracepoint can be toggled by clicking in the column left of the line numbers. The variables snapshots can be seen in the Debug tab.

There are entries for each tracepoint that was hit with its line number and a timestamp.

To stop tracing your variables, press **Delete all Tracepoints**. This keeps the list of your traces. Press **Clear Traces** to also remove the list of your traces.

Copyright © 2026 Vizrt

22.2 Create and Run Scripts

This section contains two script examples:

- Example I: This example shows how to write a simple script that controls the alpha value of an object.
- Example II: This example shows how to write a simple script that scales the x-scaling value of a container, based on x-scaling value of another container.

22.2.1 Example I

Start with a simple event-driven script.

1. Add a Sphere to a Container.
2. Add a Material and some light.
3. Create a script object.
    a. Drag the built-in Script plug-in from the Container plug-ins' Global folder onto a Container.
    b. Click on the Script plug-in icon in a Container
    c. Copy and paste the code, below, into the Script Editor.

```
Sub OnEnter()
     Alpha.Value = 100.0
End Sub
Sub OnLeave()
     Alpha.Value = 50.0
End Sub
```

This script consists of two event procedures, *OnEnter* and *OnLeave*, which are recognized by the system and invoked whenever the cursor enters or leaves the area occupied by the container in the output window. Take a closer look at what happens inside *OnEnter*:

```
Alpha.Value = 100.0
```

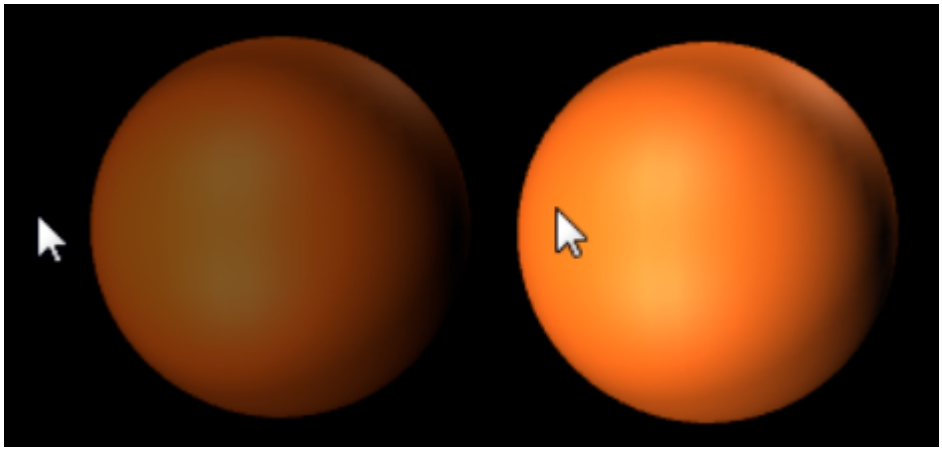
This code sets the alpha value of the container to `100` (the maximum value). If there is no Alpha object, no action occurs, so make sure that the container has an alpha object by adding the Alpha plug-in. *Alpha* refers to the container's alpha object, *Value* is a member (in this case: the only member) of the alpha object, representing its value. Changes made to *Alpha.Value* are immediately made visible in the output window.

Similarly, the code in the body of *OnLeave* sets the alpha value back to `50` .

4. Add the Alpha plug-in to the container.
5. Execute the script, click Compile and Run.
6. Click **Edit** to return to editing mode if required. To interact with the scene click the button (Scene Editor Buttons) in the Scene Editor.

Note: The button offers a good way for designers to test touch-based interaction points added to a scene. The script is then the logic which triggers what happens next (events).

Copyright © 2026 Vizrt



22.2.2 Example II

This example defines the event procedure **OnExecPerField**, which is called for each field. The functions and amount of code used in this procedure needs to be decided upon with great consideration. Include only the code that requires execution on a per-field basis, to keep the resulting performance impact as low as possible.



```
Dim cube As Container = Scene.FindContainer(“Circle”)
Sub OnExecPerField()
     If scaling.x > 1.5 Then
         cube.Scaling.x = 2.0
     Else
         cube.Scaling.x = 1.0
     End If
End Sub
```

In the first line, a variable named **cube**, of type Container, is defined:

Copyright © 2026 Vizrt

```
Dim cube As Container = Scene.FindContainer(“Cube”)
```

Variables of type Container can be initialized to point to a container in the scene tree. If there is no such container, cube is set to Null, the consequence being that any further operations on cube have no effect.
The reason why this is done on a global space is that if you called it inside the OnExecPerField, this code would be executed on every single field, impacting performance.

Now, check if the x-scaling of the current container (that is, the container holding the script) is greater than 1.5, and, depending on the result, set the x-scaling of the container named cube to 2.0 or 1.0:

```
If Scaling.x > 1.5 Then
     cube.Scaling.x = 2.0
Else
     cube.Scaling.x = 1.0
End If
```

Note: The expression `Scaling.x` refers to the current container, while `cube.Scaling.x` refers to the container referenced by the variable cube.

Copyright © 2026 Vizrt

22.3 Create Script-based Plug-ins



Scene and Container scripts can be created, saved and used again in other Scenes or on other Containers. To do this scripts can be saved as a plug-in.

Script-based plug-ins that you create are saved as *.vsl* files at *<viz data folder>\ScriptPlugins*. Usually this will be *%ProgramData%\vizrt\VizEngine\ScriptPlugins* on Windows machines.

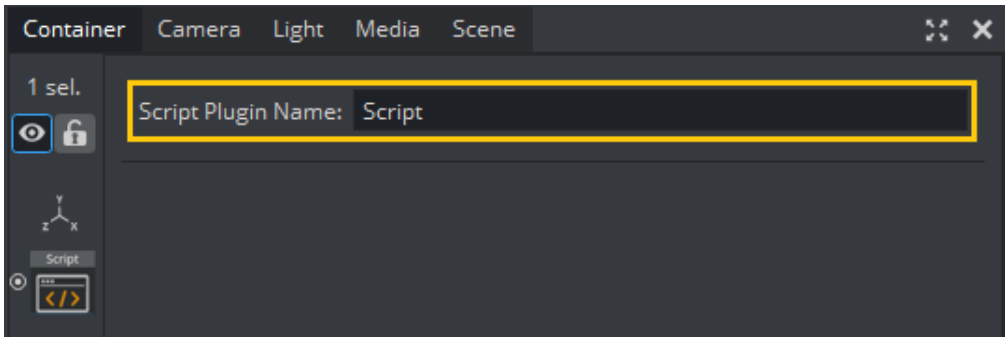
Scripts plug-ins can be used in the same way as Scene or Container plug-ins.

22.3.1 To Name Your Script

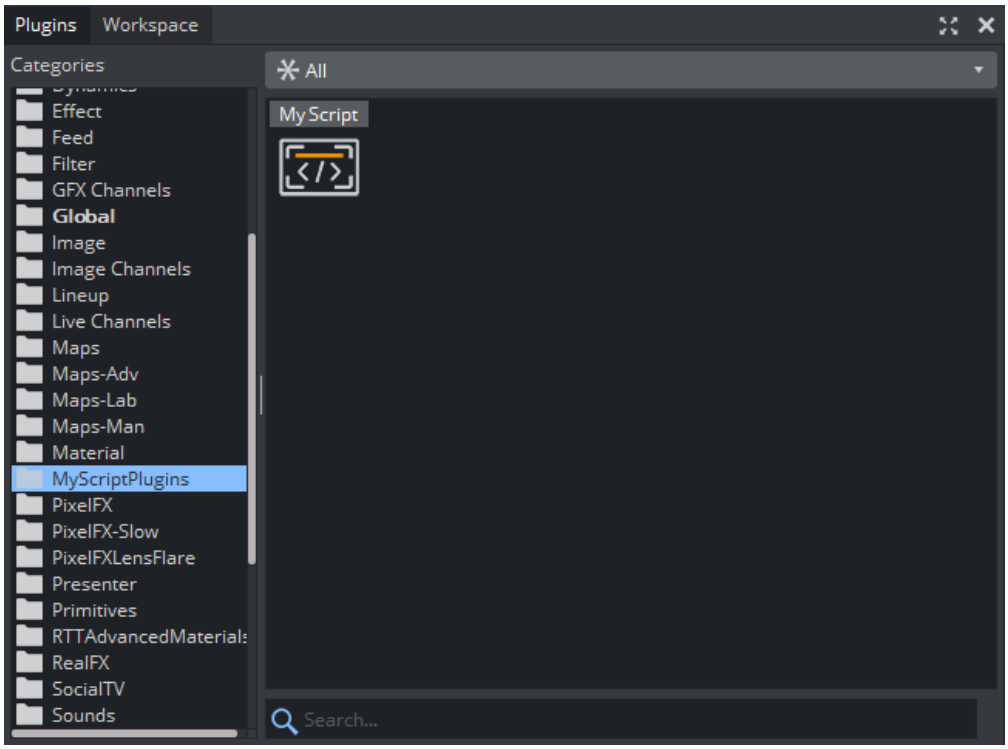
To identify your plug-in for commands or for example as an actor in the stage, you can give your plug-in a unique name.

For container scripts this can be done by entering the name at the top of the script user interface.

For scene scripts the same setting can be found under 'Scene' → 'Scene Script' from the settings selection bar.



22.3.2 To Create a Script-based Plug-in



1. Create a script.
2. Click **Compile and Run**.
3. If the script compiles successfully the **Add as Plugin** button gets enabled.
4. Click the **Add as Plugin** button.
5. Enter a name for the Script plug-in.
6. Click **OK**.

As an alternative, you can drag the script and drop it onto the Plugins Panel. The script is stored automatically in the *<viz data folder>\ScriptPlugins* folder.

This also adds some description code at the beginning of the *.vsl* file.

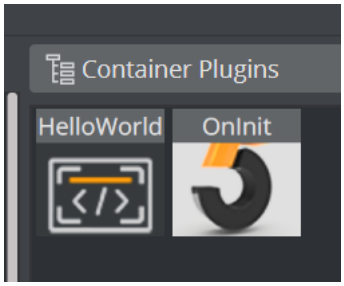
```
sub OnInitPlugin()
    RegisterPlugin("Script") 'Name of plugin
    RegisterPluginDisplayName("Script") 'Name used for icon label and tooltip
    RegisterPluginFolder("ScriptPlugins") 'Name of plugin folder
    RegisterPluginType(PLUGIN_TYPE_FUNCTION) 'Type of plugin, will be
PLUGIN_TYPE_FUNCTION or PLUGIN_TYPE_SCENE for container or scene script
end sub
```

Copyright © 2026 Vizrt

Tip: Script plug-ins are ordinary text files. If you want to change the folder they appear in, edit the *.vsl* file and change the foldername in RegisterPluginFolder("ScriptPlugins"). If you want to change the icon label and tooltip, edit the *.vsl* file and change the name in RegisterPluginDisplayName("Script").

To Create a Custom Icon for Your Script Plug-in

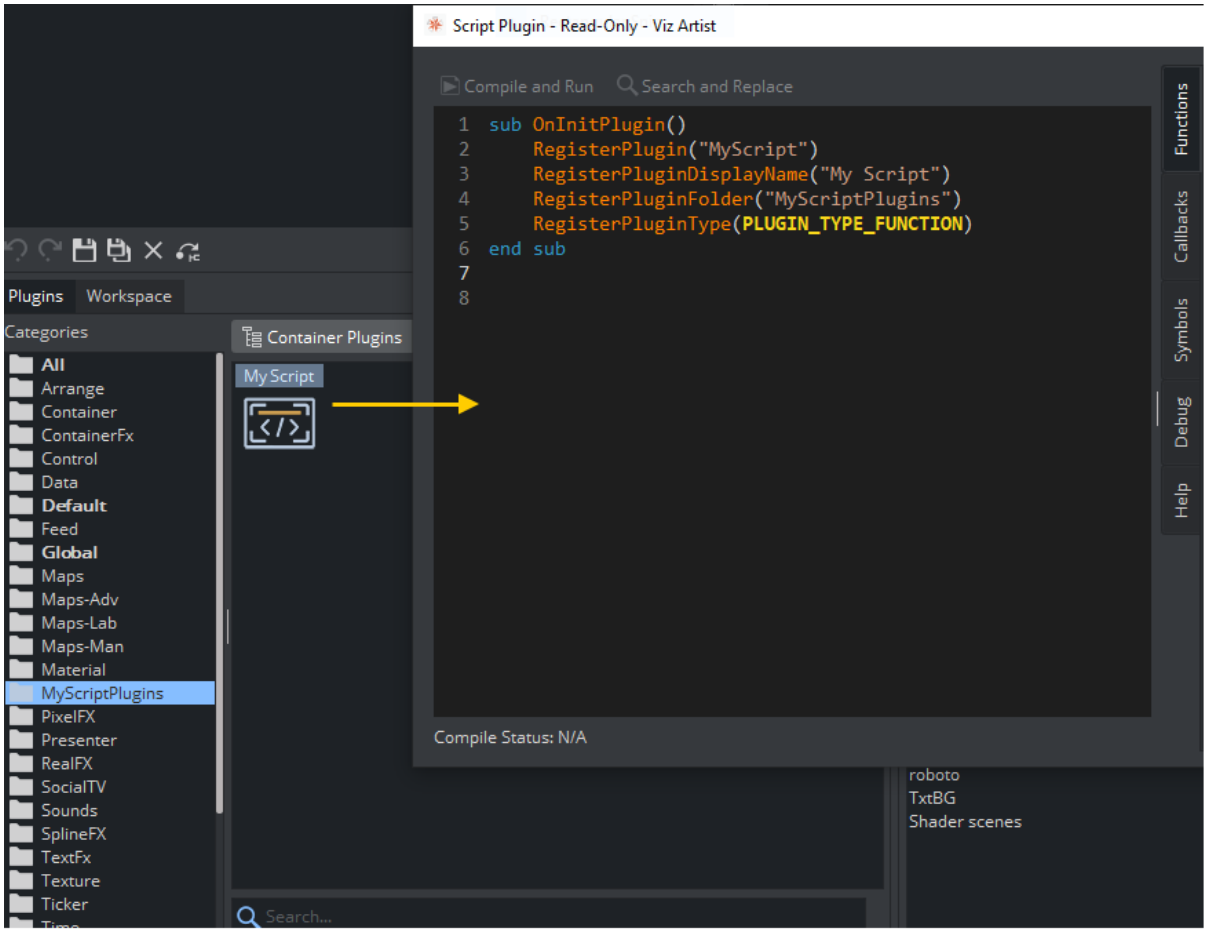
Your script plug-in is stored in *%Programdata%\vizrt\VizEngine\ScriptPlugins*.
If you create a custom folder named "**ICON**" and put a 64x64 *.bmp*, *.png*, *.tga*, *.jpg* or *.jpeg* file in there, it is used as a custom icon instead the default one.



22.3.3 To Retrieve the Original Code

To get the original code of a Script-based plug-in, **double click** it in the Plugins view. The script editor opens in read-only mode. You can not edit and save the plug-in again.

Copyright © 2026 Vizrt



22.3.4 To Edit a Script-based Plug-in

Once your script has been saved as a plug-in, it can not be modified afterwards. However, you can create a new script and drag and drop the plug-in into the script editor.

The stored plug-in code is loaded into the script editor. If you save it, you need to give it a new name.

See Also

- Script Editor

22.4 Integration Guide

22.4.1 Overview

This document extends the official Scripting documentation with production-ready examples covering Shared Memory, material control, animation dispatch, shader integration, and common pitfalls.

Most of the examples detailed here are written in an easy and understandable form for programing beginners. Some are basic, while others are more advanced and specific for their use case. More efficient code structures and improvements are possible. To explain the described terms and definitions, this section shows some source code examples and explains the important steps and commands.

This section contains information on the following topics:

- Overview
- Language Essentials
  - Key Callbacks
  - Type Casting
  - Container References
  - Initializing Scripts
  - Exposing Parameters
- Property Path Guide
  - Scene Tree Structure
  - TCP Command Path Syntax
    - Common TCP Property Paths
    - Examples
  - VizScript Property Access
    - Finding Containers
    - FindContainer vs FindSubContainer
    - Property Dot-chains
    - Plugin Parameters from VizScript
    - Batch Plug-in Updates
  - DataPool Path Syntax
  - Cross Reference
    - TCP
    - VizScript
    - DataPool
    - More cross-references
  - Naming Rules and Escaping
- Mouse Events
  - Example: Set
  - Example: Start, Stop and Reverse
  - Example: Grow and Shrink
  - Simple Gesture Recognition
- Shared Memory (SHM)
  - Reading and Writing

Copyright © 2026 Vizrt

        - Change Callbacks
        - Multi-value Strings
        - Persistence
    - Example: SHM → Text Update
    - Example: SHM → Visibility Toggle
    - Example: SHM → Animation Dispatch
        - Director API at a Glance
        - Material and Color Control
            - Set Diffuse Color
            - Convert 0-255 RGB to Engine Range
            - Specular, Emissive and Shininess
            - Material Property Map
    - Example: Alpha Fade Over Time
    - Example: Data-Driven Team Colors
    - Example: Calculate the Days of a Week and Month
    - Update a Scene Using a Text File
    - Scale and Transform Animation
        - Pop-in with Ease-out
        - Bounce Effect (Overshoot and Settle)
        - Shrink and Hide
        - Position and Rotation
    - Shader Integration
        - Classic Render Engine Pipeline (V3)
        - Viz Engine Render Pipeline (V4)
        - V3 → V4 Shader Migration
        - Controlling Texture Renderers from Script
    - Common Mistakes
    - API Quick Reference
        - Shared Memory
        - Container
        - Director

22.4.2 Language Essentials

VizScript is a VB-like language compiled and executed inside Viz Engine. Scripts live on **containers** or at **scene level**.

Key Callbacks

Callback	Triggered when
`OnInit()`	Scene load / script compile
`OnExecPerField()`	Every rendered frame (≈ 50/60 Hz)

Copyright © 2026 Vizrt

Callback	Triggered when
`OnSharedMemoryVariableChanged(map, key)`	A registered SHM key changed
`OnSharedMemoryVariableDeleted(map, key)`	A registered SHM key was deleted
`OnEnter() / OnLeave()`	Mouse hover in/out (interactive mode)
`OnButtonDown() / OnButtonUp()`	Mouse click (interactive mode)

Type Casting

VizScript is weakly typed at the SHM boundary. Always cast explicitly:

Cast	Use
`CStr(value)`	To String
`CInt(value)`	To Integer
`CDbl(value)`	To Double
`CBool(value)`	To Boolean

Container References

```
' The container this script lives on "this"

' Find any container by name (scene-wide search)
Scene.FindContainer("Background")

' Navigate the tree
this.ParentContainer
this.ChildContainer
this.NextContainer

' Guard: always check FindSubContainer succeeded
dim c as Container
c = this.FindSubContainer("Label")
```

Copyright © 2026 Vizrt

```
If c.VizId > 0 Then
    ' safe to use c
End If
```

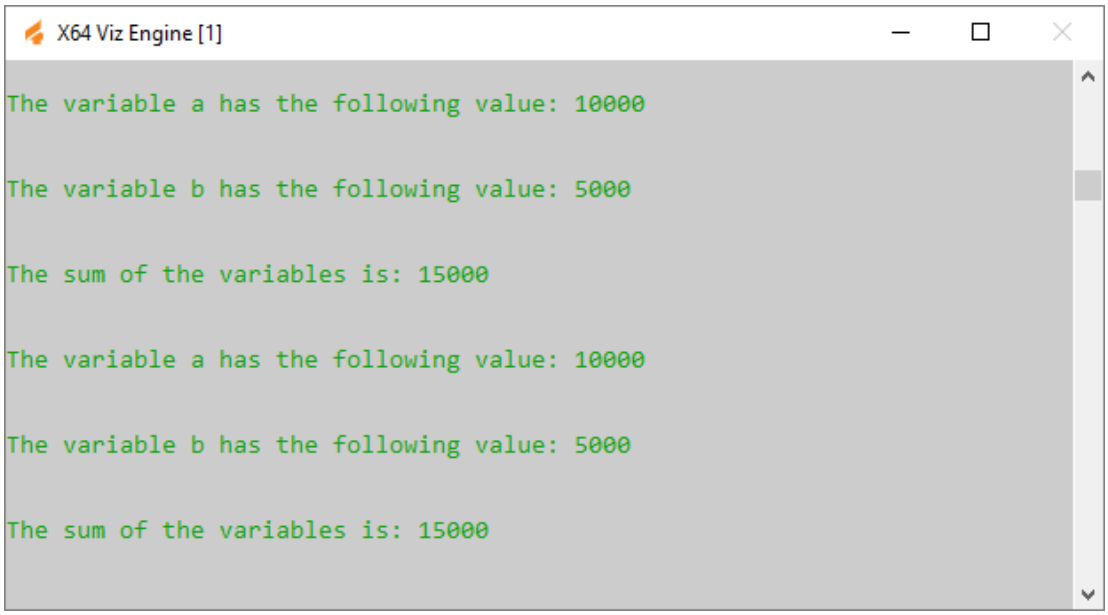
Initializing Scripts

This example declares and defines two variables with a value. In the main procedure `OnInit()` those two variables and their addition are output to the console using Println.

Callback procedures like `OnInit()` are used to initialize scripts.

```
sub OnInit()
'Declaration of the variables
dim a as Integer
a = 10000
dim b as Integer
b = 5000
'Start of the main procedure sub OnInit()
'Executing the commands
println "\n"
    println "The variable a has the following value: " & a
println "\n"
    println "The variable b has the following value: " & b
println "\n"
    println "The sum of the variables is: " & a + b
'End of the main procedure
end sub
```

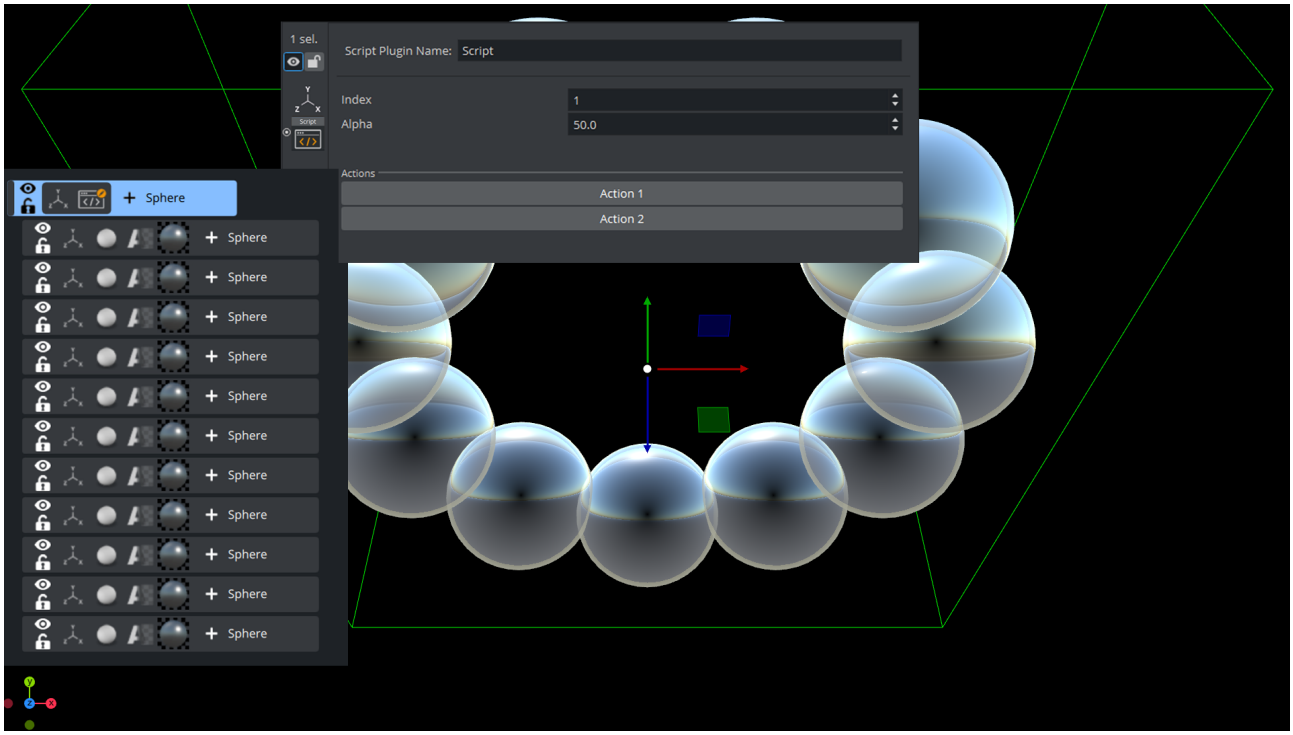
Result output to the Viz Engine console:



Copyright © 2026 Vizrt

Exposing Parameters

This script example creates a custom editor for filling in index and alpha values. The index value decides which container does not receive the alpha value, whereas the alpha value is set for all other containers.




```
sub OnInitParameters()
    RegisterParameterInt("index", "Index", 1, 1, 7)
    RegisterParameterDouble("alpha", "Alpha", 50.0, 0.0, 100.0)
    RegisterSpacer("space", 30)
    RegisterSeparator("sep", "Actions")
    RegisterPushButton("action1", "Action 1", 1)
    RegisterPushButton("action2", "Action 2", 1)
end sub
sub OnExecAction(buttonId as Integer)
    dim c as Container
    dim i, index as Integer
    dim a as Double
    index = GetParameterInt("index")
    a = GetParameterDouble("alpha")
    i = 1
    c = ChildContainer
    do while c <> null
        if i = index then
            c.alpha.value = 100.0
        else
            c.alpha.value = a
```

Copyright © 2026 Vizrt

```
        end if
    i = i + 1
    c = c.NextContainer
    loop
end sub
```

22.4.3 Property Path Guide

Viz Engine exposes scene properties through **three access methods**: TCP commands, VizScript, and DataPool. Each has its own path syntax addressing the same underlying data. This chapter explains how paths are constructed, how to translate between the three systems, and how to resolve plugin parameters.

Scene Tree Structure

```
SCENE (root)
  └─── TREE (scene root container)
       ├─── $Background
       │    ├─── $BG_Image
       │    └─── $BG_Color
       ├─── $Content
       │    ├─── $Header
       │    │    └─── $HeaderText
       │    └─── $Body
       │         ├─── $Row1
       │         └─── $Row2
       └─── $Foreground
            └─── $Logo
```

Containers are addressed by **name** (prefixed with `$` in TCP, plain in VizScript) or by **index** (`#n` , 1-based).

TCP Command Path Syntax

The universal pattern:

```
MAIN_SCENE*TREE*$<ContainerName>*<PROPERTY> <ACTION> [<VALUE>]
```

The `*` character is the path delimiter. Segments can be combined freely:

Component	Syntax	Description
Scene reference	`MAIN_SCENE`	Currently loaded main scene (underscore, not `*`)
Tree root	`TREE`	Root of the container hierarchy

Copyright © 2026 Vizrt

Component	Syntax	Description
Container by name	`$ContainerName`	Name lookup (case-sensitive)
Container by index	`#1`, `#2`	Child index, 1-based
Property	`*POSITION`	Property node access
Sub-property	`*POSITION*X`	Nested property

Common TCP Property Paths

What	TCP Path Suffix	Example Value
Position XYZ	`*POSITION`	`SET 100.0 50.0 0.0`
Position X only	`*POSITION*X`	`SET 100.0`
Rotation	`*ROTATION`	`SET 0.0 0.0 45.0`
Scaling	`*SCALING`	`SET 1.0 2.0 1.0`
Visibility	`*ACTIVE`	`SET ON` / `SET OFF`
Alpha	`*MATERIAL*ALPHA`	`SET 0.5`
Diffuse color	`*MATERIAL*DIFFUSE`	`SET 1.0 0.0 0.0 1.0`
Text content	`*GEOM*TEXT`	`SET "Hello"`
Plugin parameter	`*FUNCTION*<Plugin>*<Param>`	`SET 42`

Examples

```
' By name - nested path through hierarchy
```

Copyright © 2026 Vizrt

```
MAIN_SCENE*TREE*$Content*$Header*$HeaderText*GEOM*TEXT SET "Breaking News"

' By index - 1st child of TREE, then 2nd child, then 1st child
MAIN_SCENE*TREE*#1*#2*#1*GEOM*TEXT SET "Title"

' Mixed - name and index combined
MAIN_SCENE*TREE*$Content*#1*$HeaderText*GEOM*TEXT SET "Mixed path"

' Read a property
MAIN_SCENE*TREE*$Header*POSITION GET
MAIN_SCENE*TREE*$Header*ACTIVE GET
```

MAIN_SCENE uses underscore. Always write `MAIN_SCENE` (with underscore), never `MAIN*SCENE` (with asterisk). The asterisk is the path delimiter and would break the command.

VizScript Property Access

VizScript uses **dot notation** on object references. Container names are *plain strings* without the `$` prefix.

Finding Containers

```
' Scene-wide search by name (depth-first, returns FIRST match)
dim c as Container
c = Scene.FindContainer("HeaderText")

' Relative: search children of a specific container
dim child as Container
child = this.FindSubContainer("Label")

' Tree navigation
dim parent as Container
parent = this.ParentContainer

dim first as Container
first = this.ChildContainer

dim next as Container
next = this.NextContainer

' Scene root
dim root as Container
root = Scene.RootContainer
```

Copyright © 2026 Vizrt

FindContainer vs FindSubContainer

`Scene.FindContainer("X")` searches the *entire* scene tree (depth-first).
`this.FindSubContainer("X")` searches only *descendants* of the current container. Use `FindSubContainer` when you know the relative position. It is faster and avoids name collisions.

Property Dot-chains

```
dim c = this
' Transform properties
c.Position.X = 100.0
c.Position.Y = -50.0
c.Rotation.Z = 45.0
c.Scaling.X = 2.0

' Visibility
c.Active = True

' Material (see chapter 7 for full coverage)
c.Material.Color = cColor(1,1,1)        ' CColor - diffuse
c.Material.Alpha = 0.5         ' Double - transparency
c.Material.Specular.Red = 1.0
c.Material.Emission.Green = 0.1
c.Material.Shininess = 0.8

' Geometry / text
c.Geometry.Text = "Breaking News"
```

Plugin Parameters from VizScript

```
dim c = this
' Access a function plugin by name
dim fn as PluginInstance
fn = c.GetFunctionPluginInstance("ControlText")

' Read / write parameters by name
fn.SetParameterString("value", "Hello World")
dim val as String
val = fn.GetParameterString("value")

' Access geometry plugin
dim gp as PluginInstance
gp = c.GetGeometryPluginInstance()
gp.SetParameterDouble("width", 200.0)

' Access another container's SCRIPT plugin
dim sp as PluginInstance
```

Copyright © 2026 Vizrt

```
sp = this.FindSubContainer("F1").ScriptPluginInstance
sp.SetParameterDouble("topTransition", 0)
```

Batch Plug-in Updates

```
' Collect plugin instances into an array for bulk operations
dim arrows as Array[PluginInstance]
arrows.Push(this.FindSubContainer("ArrowFill").GetGeometryPluginInstance())
arrows.Push(this.FindSubContainer("Cylinder").GetGeometryPluginInstance())

dim i as Integer
for i = 0 to arrows.Size - 1
    arrows[i].SetParameterDouble("Point1X", 10.0)
    arrows[i].SetParameterDouble("Point1Y", 20.0)
next
```

DataPool Path Syntax

DataPool uses **dot-separated** paths. Container names appear *without* the `$` prefix:

```
DataPool.Scene.<ContainerName>.<Property>
```

DataPool Path	Equivalent TCP Path
`Scene.Header.Position.X`	`MAIN_SCENE*TREE*$Header*POSITION*X`
`Scene.Header.Active`	`MAIN_SCENE*TREE*$Header*ACTIVE`
`Scene.HeaderText.Geom.Text`	`MAIN_SCENE*TREE*$HeaderText*GEOM*TEXT`
`Scene.Logo.Material.Alpha`	`MAIN_SCENE*TREE*$Logo*MATERIAL*ALPHA`
`Scene.MyCtrl.Function.Ctrl.Value`	`MAIN_SCENE*TREE*$MyCtrl*FUNCTION*Ctrl*VALUE`

Cross Reference

These three code blocks do *exactly* the same thing: Set the X position of a container named "Header" to 100.

Copyright © 2026 Vizrt

TCP

```
MAIN_SCENE*TREE*$Header*POSITION*X SET 100.0
```

VizScript

```
dim c as Container
c = Scene.FindContainer("Header")
c.Position.X = 100.0
```

DataPool

```
DataPool.Scene.Header.Position.X = 100.0
```

More cross-references

Operation	TCP suffix	VizScript	DataPool
Set text	`*GEOM*TEXT SET "val"`	`c.Geometry.Text = "val"`	`Scene.X.Geom.Text`
Visibility	`*ACTIVE SET ON`	`c.Active = True`	`Scene.X.Active = 1`
Alpha	`*MATERIAL*ALPHA SET 0.5`	`c.Material.Alpha = 0.5`	`Scene.X.Material.Alpha`
Plug-in param	`*FUNCTION*P*key SET val`	`fn.SetParameterString("key","val")`	`Scene.X.Function.P.key`

Naming Rules and Escaping

- Container names should use **alphanumeric characters and underscores** only.
- The `*` character is **reserved** as the TCP path delimiter, never use it in container names.
- The `$` prefix is required **only in TCP** paths. VizScript and DataPool use plain names.
- Spaces in names are supported but may break TCP commands, avoid them.
- `FindContainer` returns the **first match** (depth-first). Use unique names to avoid ambiguity.
- Always guard `FindSubContainer` results: `If c.VizId > 0 Then`.

Copyright © 2026 Vizrt

22.4.4 Mouse Events

Example: Set

This script example shows how to use a 'mouse event'" to set an object's alpha value with OnEnter and OnLeave.

```
sub OnEnter()
    alpha.value = 100.0
end sub
sub OnLeave()
    alpha.value = 50.0
end sub
```

Example: Start, Stop and Reverse

This script, when added to a container, allows an animation to be started with the click of a mouse button over an object (for example, rectangle). Press again and it stops. Press again and it continues from the point it stopped.

```
dim my_dir = stage.FindDirector("Default")
sub OnLButtonDown()
     if my_dir.IsAnimationRunning() then
          my_dir.StopAnimation()
     else
         my_dir.ContinueAnimation()
     end if
end sub
```

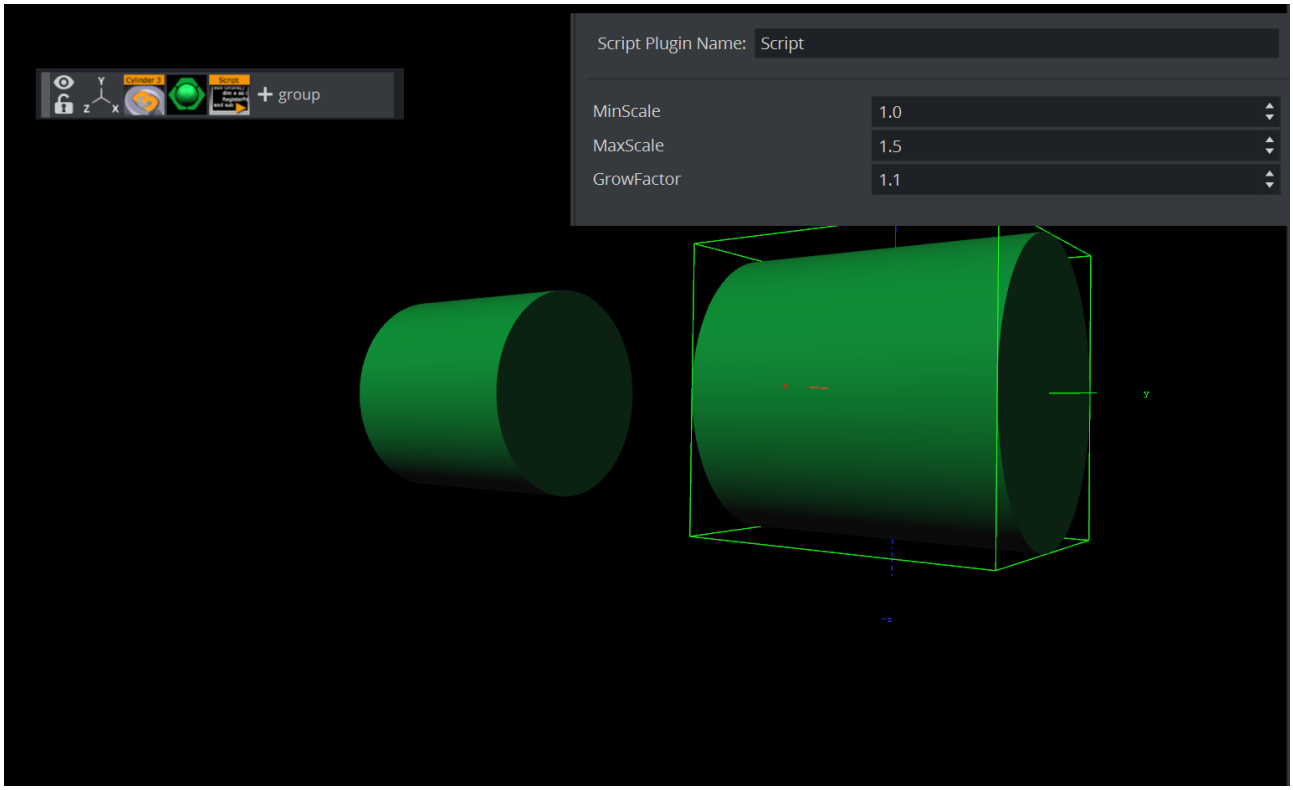
With the script detailed below, start an animation and let it run until a stop point. Click again and it runs in reverse:

```
dim my_dir = stage.FindDirector("Defult")
sub OnLButtonDown()
     menu.ContinueAnimation()
     if my_dir.reverse == true then
         my_dir.reverse = false
     else
         my_dir.reverse = true
     end if
end sub
```

Example: Grow and Shrink

With this script added to a Container an object grows when the mouse is hovering over the object and conversely. In this case, the center of the object is set to the lower right corner to let it grow out from that center.

Copyright © 2026 Vizrt




```
dim growing as Boolean = false
sub OnInitParameters()
     RegisterParameterDouble("min", "MinScale", 1.0, 0.0, 10.0)
     RegisterParameterDouble("max", "MaxScale", 1.5, 0.0, 10.0)
     RegisterParameterDouble("fac", "GrowFactor", 1.1, 1.0, 10.0)
end sub
sub OnEnter()
     growing = true
end sub
sub OnLeave()
     growing = false
end sub
sub OnExecPerField()
     dim min = GetParameterDouble("min")
     dim max = GetParameterDouble("max")
     dim fac = GetParameterDouble("fac")
     if growing then
         scaling.xyz *= fac
         if scaling.x > max then scaling.xyz = max
     else
         scaling.xyz /= fac
         if scaling.x < min then scaling.xyz = min
     end if
end sub
```

Copyright © 2026 Vizrt

Simple Gesture Recognition

This example script can be used to detect a drawn square on the screen:

```
dim gRecognizer as gesturerecognizer
sub OnInit()
    gRecognizer.handlepatterns = true
    gRecognizer.addhandler(scriptplugininstance)
end sub

sub OnGesture(gesture As Gesture)
    if gesture.patternid == PATTERN_SQUARE then
        println("This is a square")
    end if
end sub

sub OnTouchTrace(trace as Trace, touch as Touch)
    gRecognizer.addtrace(trace)
end sub
```

Note: Some gesture patterns need to be checked for their alignment as well, for example *up*, *down*, *left*, and so on. For this you can activate the `CheckPatternOrientation` property in the gesture recognizer object.

22.4.5 Shared Memory (SHM)

Shared Memory is the primary data transport between external systems (Viz Trio, Media Sequencer Engine, automation) and VizScript. Three scopes exist:

Scope	Object	Visibility
Scene	`Scene.Map`	Current scene only, cleared on scene unload
Engine	`System.Map`	All scenes on this Viz Engine instance
Distributed	`VizCommunication.Map`	All Viz Engines in the cluster (network sync)

Reading and Writing

```
' Write
Scene.Map["score_home"] = CStr(3)
```

Copyright © 2026 Vizrt

```
' Read (returns String - cast as needed)
dim val as String
val = Scene.Map["score_home"]
dim score as Integer
score = CInt(val)
```

Change Callbacks

```
Sub OnInit()
    ' Listen to a specific key
    Scene.Map.RegisterChangedCallback("score_home")

    ' Listen to ALL keys (wildcard)
    Scene.Map.RegisterChangedCallback("*")
End Sub

Sub OnSharedMemoryVariableChanged(map as SharedMemory, mapKey as String)
    If mapKey = "score_home" Then
        dim val as String
        val = map[mapKey]
        ' ... update visuals
    End If
End Sub
```

Performance: Wildcard callbacks (`"*"`) fire on *every* SHM write across the registered scope. Use specific keys in production. Always call `UnregisterChangedCallback(key)` if you no longer need notifications.

Multi-value Strings

```
' Store CSV: "Arsenal,3,Chelsea,1"
Scene.Map["match_data"] = "Arsenal,3,Chelsea,1"

' Parse
dim raw as String
raw = Scene.Map["match_data"]
dim parts as Array[String]
raw.Split(",", parts)
' parts[0] = "Arsenal", parts[1] = "3", etc.
```

Copyright © 2026 Vizrt

Persistence

```
' Save SHM map to Graphic Hub (blocks renderer - use sparingly)
Scene.Map.SaveToDb("path/to/snapshot")

' Restore
Scene.Map.LoadFromDb("path/to/snapshot")
```

Warning: `SaveToDb` and `LoadFromDb` block the render thread. Never call them during live On Air output. Use them only during scene setup or offline prep.

22.4.6 Example: SHM → Text Update

The most common VizScript pattern: bind a Shared Memory key to a text geometry so external data appears on screen.

```
dim shmKey as String

Sub OnInit()
    shmKey = "headline"
    Scene.Map.RegisterChangedCallback(shmKey)
    UpdateText()
End Sub

Sub OnSharedMemoryVariableChanged(map as SharedMemory, mapKey as String)
    If mapKey = shmKey Then
        UpdateText()
    End If
End Sub

Sub UpdateText()
    dim val as String
    val = Scene.Map[shmKey]

    If val = "" Then
        Exit Sub
    End If

    dim c as Container
    c = Scene.FindContainer("HeadlineText")
    c.Geometry.Text = val
End Sub
```

Copyright © 2026 Vizrt

Pattern

`OnInit` → register + immediate read; `OnSharedMemoryVariableChanged` → re-read on change. Both delegate to a shared `UpdateText()` so the logic is never duplicated.

22.4.7 Example: SHM → Visibility Toggle

Remote show/hide of a container via distributed Shared Memory. Each container uses `this.Name` as its SHM key, so the same script works on any container without modification.

```
Sub OnInit()
    VizCommunication.Map.RegisterChangedCallback(this.Name)

    ' Set initial value if it doesn't exist
    If VizCommunication.Map[this.Name] = "" Then
        VizCommunication.Map[this.Name] = CStr(True)
    End If
End Sub

Sub OnSharedMemoryVariableChanged(map as SharedMemory, mapKey as String)
    If mapKey = this.Name Then
        this.Active = CBool(map[mapKey])
    End If
End Sub
```

Drop this script on any container. The container's name becomes the SHM key automatically. From any Viz Engine in the cluster, write `VizCommunication.Map["LogoPanel"] = "False"` to hide it.

22.4.8 Example: SHM → Animation Dispatch

Map integer SHM values to different Director animation states. Production pattern from a multi-studio broadcast environment.

```
Sub OnInit()
    VizCommunication.Map.RegisterChangedCallback("vizrt/" & this.Name)
End Sub

Sub OnSharedMemoryVariableChanged(map as SharedMemory, mapKey as String)
    If mapKey = "vizrt/" & this.Name Then
        dim val as Integer
        val = CInt(map[mapKey])

        Select Case val
            Case 0
                ' Hide: jump director to frame 0
                this.GetDirector().Show(0)
            Case 1
                ' Show: jump director to frame 3
```

Copyright © 2026 Vizrt

```
                this.GetDirector().Show(3)
            Case 2
                ' Animate in: jump to start, play forward
                this.GetDirector().Show(0.02)
                this.GetDirector().ContinueAnimation()
            Case 3
                ' Animate out: jump to end, play reverse
                this.GetDirector().Show(3)
                this.GetDirector().ContinueAnimationReverse()
        End Select
    End If
End Sub
```

Director API at a Glance

Method	Effect
`GetDirector().Show(time)`	Jump timeline to *time* (seconds)
`GetDirector().ContinueAnimation()`	Play forward from current position
`GetDirector().ContinueAnimationReverse()`	Play backward from current position
`GetDirector().StartAnimation()`	Play from the beginning
`GetDirector().StopAnimation()`	Pause at current position

Material and Color Control

VizScript controls material appearance through the `CColor` type and the container's `Material` property.

Set Diffuse Color

```
Sub SetDiffuse()
    dim c as Container
    c = Scene.FindContainer("Background")

    dim col as Color
    col.Red   = 0.2
    col.Green = 0.4
    col.Blue  = 0.8
    col.Alpha = 1.0

    c.Material.Color = col
```

Copyright © 2026 Vizrt

```
End Sub
```

Convert 0-255 RGB to Engine Range

```
Sub SetColorFromRGB()
    dim r as Integer  = 255
    dim g as Integer  = 128
    dim b as Integer  = 0

    dim c as Container
    c = Scene.FindContainer("Highlight")

    dim col as Color
    col.Red   = CDbl(r) / 255.0
    col.Green = CDbl(g) / 255.0
    col.Blue  = CDbl(b) / 255.0
    col.Alpha = 1.0

    c.Material.Color = col
End Sub
```

Specular, Emissive and Shininess

```
Sub SetAdvancedMaterial()
    dim c as Container
    c = Scene.FindContainer("GlossPanel")

    ' Diffuse white
    dim col as Color
    col.Red = 1.0
    col.Green = 1.0
    col.Blue = 1.0
    col.Alpha = 1.0
    c.Material.Color = col

    ' Specular highlight
    c.Material.Specular.Red   = 1.0
    c.Material.Specular.Green = 1.0
    c.Material.Specular.Blue  = 1.0

    ' Emissive glow
    c.Material.Emission.Red   = 0.1
    c.Material.Emission.Green = 0.05
    c.Material.Emission.Blue  = 0.0

    ' Shininess: 0.0 (matte) → 1.0 (mirror)
    c.Material.Shininess = 0.8
```

Copyright © 2026 Vizrt

```
End Sub
```

Material Property Map

Property	What it controls	Range
`Material.Color`	Diffuse color (base appearance)	CColor (0.0–1.0 per channel)
`Material.Alpha`	Transparency (classic materials)	0.0 (invisible) – 1.0 (opaque)
`Material.Specular`	Highlight reflection color	CColor
`Material.Emission`	Self-illumination glow	CColor
`Material.Shininess`	Specular tightness	0.0 – 1.0

22.4.9 Example: Alpha Fade Over Time

Smoothly fade a container out by decrementing alpha each frame in `OnExecPerField`.

```
dim fadeAlpha as Double
dim fadeActive as Boolean

Sub OnInit()
    fadeAlpha = 1.0
    fadeActive = False
End Sub

Sub StartFade()
    fadeAlpha = 1.0
    fadeActive = True
End Sub

Sub OnExecPerField()
    If fadeActive = False Then
        Exit Sub
    End If

    fadeAlpha = fadeAlpha - 0.02
    If fadeAlpha < 0.0 Then
        fadeAlpha = 0.0
```

Copyright © 2026 Vizrt

```
        fadeActive = False
    End If

    dim c as Container
    c = Scene.FindContainer("FadePanel")

    dim col as CColor
    col = c.Material.Color
    col.Alpha = fadeAlpha
    c.Material.Color = col
End Sub
```

Frame rate: At 50 fps, decrementing by 0.02 per frame gives a ~1 second fade. Adjust the step value to control speed.

22.4.10 Example: Data-Driven Team Colors

Parse a CSV color string from an external data source and apply it to a container's material. Production pattern from VizArena sports graphics.

```
dim SEPARATOR_COLOR as String

Sub OnInit()
    SEPARATOR_COLOR = ","
End Sub

Sub ApplyColorFromData(container as Container, colorString as String)
    ' Expected format: "R,G,B,A" where each value is 0-255
    dim colorComponents as Array[String]
    colorString.Split(SEPARATOR_COLOR, colorComponents)

    If colorComponents.Size = 4 Then
        dim noggi as Container
        noggi = container.FindSubContainer("Noggi")

        If noggi.VizId > 0 Then
            dim c as Color
            c.Red   = CDbl(colorComponents[0]) / 255.0
            c.Green = CDbl(colorComponents[1]) / 255.0
            c.Blue  = CDbl(colorComponents[2]) / 255.0
            c.Alpha = CDbl(colorComponents[3]) / 255.0

            noggi.Material.Color = c
            noggi.Material.Alpha = CDbl(colorComponents[3]) / 255.0
        End If
    End If
End Sub
```

Copyright © 2026 Vizrt

Material.Alpha
On classic (V3) materials, `Material.Color.Alpha` and `Material.Alpha` are *separate*. Set both for transparency to work correctly.

22.4.11 Example: Calculate the Days of a Week and Month

This script example calculates the day of the week and the number of days of the month:

```
DIM smap AS Array[String]

smap.size = 7
smap[0] = "Sun"
smap[1] = "Mon"
smap[2] = "Tue"
smap[3] = "Wed"
smap[4] = "Thu"
smap[5] = "Fri"
smap[6] = "Sat"

function CalcDayOfWeek(iYear AS Integer, iMonth AS Integer, iDay AS Integer) AS
String

DIM dt AS DateTime
    dt.year = iYear
    dt.month = iMonth
    dt.dayofmonth = iDay
    dt.normalize()

    CalcDayOfWeek = smap[dt.dayofweek]
end function

function CalcNumOfDaysOfMonth(iYear AS Integer, iMonth AS Integer) AS Integer
DIM dt AS DateTime
    dt.year = iYear
    dt.month = iMonth+1
    dt.dayofmonth = 0
    dt.normalize()
    CalcNumOfDaysOfMonth = dt.dayofmonth
end function

sub OnInit()
DIM dt AS DateTime
    dt.year = 2019
    dt.month = 2
    dt.dayofmonth = 23

    println("Date: " & CalcDayOfWeek(dt.year, dt.month, dt.dayofmonth) & ", " &
dt.year & "" & dt.month & "" & dt.dayofmonth)
    println("Days: " & CalcNumOfDaysOfMonth(dt.year, dt.month))
end sub
```

Copyright © 2026 Vizrt

22.4.12 Update a Scene Using a Text File

This script example updates a Scene from a text file.

```
dim timer,limit as integer
timer = 0
sub OnInitParameters()
    RegisterInfoText("Text File Grabber 1.0" & chr(10) & chr(10) & "Grabs text file
and puts optional prefix/suffix around the contents of the text file." & chr(10)
& "Filename supports paths with drive letters and UNC paths." & chr(10) & chr(10)
& "Text in \"Default Text\" box is used if file specified in File box cannot be opend
or accessed." & chr(10) & chr(10)         & "Listens to scene map variable
\"text_update\" -- text files are re-read when this variable changes.")
    RegisterFileSelector("file", "File", "c:\test", "1.txt", "*.*")
    RegisterParameterString("default","Default text","",40,50,"")
    RegisterParameterString("prefix","Prefix (optional)","",40,50,"")
    RegisterParameterBool("prefixIfNull","Include prefix if file not found",FALSE)
    RegisterParameterString("suffix","Suffix (optional)","",40,50,"")
    RegisterParameterBool("suffixIfNull","Include suffix if file not found",FALSE)
    RegisterParameterInt("updateFields","Auto-Update interval
(fields)",300,60,999999999999)
    RegisterParameterBool("debug","Debug output to console",FALSE)
end sub

sub OnParameterChanged(parameterName As String)
   're-populate text when a parameter changes
    if parameterName <> "updateFields" then UpdateText
    if parameterName = "updateFields" then
        limit = GetParameterInt("updateFields")
    end if
end sub

sub OnExecPerField()
    timer = timer + 1
    if timer >= limit then
        if GetParameterBool("debug") then println "Timer limit reached"
            UpdateText
            timer = 0
    end if
end sub

sub OnInit()
    Scene.Map.RegisterChangedCallback("text_update")
    limit = GetParameterInt("updateFields")
    UpdateText
end sub

sub OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)
    if mapKey = "text_update" then
        UpdateText
    end if
```

Copyright © 2026 Vizrt

```
end sub

sub UpdateText()
    geometry.text = GetTextFile()
end sub

function GetTextFile() as string
    dim content, path as string
    path = GetParameterString("file")
    if GetParameterBool("debug") then println "File path: " & path
    if System.LoadTextFile( path, content ) then
        if GetParameterBool("debug") then println "Successfully loaded file"
        content.trim
        if len(GetParameterString("prefix")) > 0 then content =
GetParameterString("prefix") & " " & content
        if len(GetParameterString("suffix")) > 0 then content = content & " " &
GetParameterString("suffix")
    else
        if GetParameterBool("debug") then println "Could not load file; using default
text"
        content = GetParameterString("default")
        if GetParameterBool("prefixIfNull") then content =
GetParameterString("prefix") & " " & content
        if GetParameterBool("suffixIfNull") then content = content & " " &
GetParameterString("suffix")
    end if
    GetTextFile = content
    if GetParameterBool("debug") then println "Returning '" & GetTextFile & "'"
end function
```

22.4.13 Scale and Transform Animation

Pop-in with Ease-out

```
dim scaleT as Double
dim scaleActive as Boolean

Sub OnInit()
    scaleT = 0.0
    scaleActive = False
    dim c as Container
    c = Scene.FindContainer("PopItem")
    c.Scaling.X = 0.0
    c.Scaling.Y = 0.0
    c.Scaling.Z = 0.0
End Sub

Sub PopIn()
    scaleT = 0.0
```

Copyright © 2026 Vizrt

```
    scaleActive = True
End Sub

Sub OnExecPerField()
    If scaleActive = False Then Exit Sub

    scaleT = scaleT + 0.04
    If scaleT > 1.0 Then
        scaleT = 1.0
        scaleActive = False
    End If

    ' Ease-out: 1 - (1-t)^2
    dim inv as Double
    inv = 1.0 - scaleT
    dim eased as Double
    eased = 1.0 - inv * inv

    dim c as Container
    c = Scene.FindContainer("PopItem")
    c.Scaling.X = eased
    c.Scaling.Y = eased
    c.Scaling.Z = eased
End Sub
```

Bounce Effect (Overshoot and Settle)

```
dim bounceT as Double
dim bounceActive as Boolean

Sub StartBounce()
    bounceT = 0.0
    bounceActive = True
End Sub

Sub OnExecPerField()
    If bounceActive = False Then Exit Sub

    bounceT = bounceT + 0.03
    If bounceT > 1.0 Then
        bounceT = 1.0
        bounceActive = False
    End If

    dim s as Double
    If bounceT < 0.6 Then
        s = (bounceT / 0.6) * 1.15
    Else
        dim remainder as Double
        remainder = (bounceT - 0.6) / 0.4
```

Copyright © 2026 Vizrt

```
        s = 1.15 - 0.15 * remainder
    End If

    dim c as Container
    c = Scene.FindContainer("BounceItem")
    c.Scaling.X = s
    c.Scaling.Y = s
End Sub
```

Shrink and Hide

```
dim shrinkT as Double
dim shrinkActive as Boolean

Sub ShrinkAndHide()
    shrinkT = 0.0
    shrinkActive = True
End Sub

Sub OnExecPerField()
    If shrinkActive = False Then Exit Sub

    shrinkT = shrinkT + 0.05
    If shrinkT > 1.0 Then
        shrinkT = 1.0
        shrinkActive = False
        dim c as Container
        c = Scene.FindContainer("DismissItem")
        c.Active = False
        Exit Sub
    End If

    dim s as Double
    s = 1.0 - shrinkT

    dim c as Container
    c = Scene.FindContainer("DismissItem")
    c.Scaling.X = s
    c.Scaling.Y = s
    c.Scaling.Z = s
End Sub
```

Position and Rotation

```
' Move a container
dim c as Container
c = Scene.FindContainer("Slider")
c.Position.X = 120.0
```

Copyright © 2026 Vizrt

```
c.Position.Y = -50.0
c.Position.Z = 0.0

' Rotate
c.Rotation.X = 0.0
c.Rotation.Y = 0.0
c.Rotation.Z = 45.0    ' 45° around Z axis
```

22.4.14 Shader Integration

Viz Engine's shader system changed significantly between V3 and V4. VizScript interacts with shaders differently depending on the pipeline.

Classic Render Engine Pipeline (V3)

In the Classic Render Engine Pipeline, shaders are **plugin DLLs** assigned to containers. VizScript accesses shader parameters through the plugin system, just like any other function plugin:

```
' Access a V3 shader plugin parameter via the container's function stack
' (shader must already be assigned to the container in Viz Artist)
dim c as Container
c = Scene.FindContainer("EffectPanel")

' Shader params are accessed like plugin params -
' use the TCP command interface from external control,
' or set material properties that the shader reads.
```

Viz Engine Render Pipeline (V4)

In the Viz Engine Render Pipeline, shaders are **GLSL programs** managed in the Workspace. They use `@registerParameter` annotations to expose parameters. VizScript can control the visual result through the **Material Definition** properties:

```
' V4: Control material properties that the shader reads
dim c as Container
c = Scene.FindContainer("PBRPanel")

' Diffuse / base color
dim col as Color
col.Red = 0.8
col.Green = 0.2
col.Blue = 0.1
col.Alpha = 1.0
c.Material.Color = col

' For advanced PBR shader parameters (roughness, metallic, etc.),
' use TCP commands from the automation layer:
```

Copyright © 2026 Vizrt

```
'   MAIN_SCENE*TREE*$PBRPanel*MATERIAL*MATERIAL_DEFINITION SET ROUGHNESS_FACTOR -0.3
'   MAIN_SCENE*TREE*$PBRPanel*MATERIAL*MATERIAL_DEFINITION SET METALLIC_FACTOR 0.8
```

V3 → V4 Shader Migration

Aspect	Classic Renderer (V3)	Viz Engine Renderer (V4)
Implementation	Plugin DLLs (C/C++)	GLSL shaders in Workspace
Assignment	`SHADER*<PluginName>`	`MATERIAL*VIZ_SHADER`
Parameters	Plugin-defined	`@registerParameter` GLSL annotations
Script access	Plugin API / TCP	Material properties + TCP
Lighting	Fixed-function + per-plugin	Integrated PBR / Phong

Controlling Texture Renderers from Script

Texture renderers (clip players, scene views, preprocessing) are Workspace objects. VizScript can control playback and texture assignment indirectly via Shared Memory or TCP commands triggered by the automation layer. Direct VizScript API for Workspace objects is limited, use the TCP command interface for advanced shader parameter manipulation.

22.4.15 Common Mistakes

Mistake	Symptom	Fix
Using `+` for string concatenation	Compile error or numeric addition	Use `&` operator: `"hello " & "world"`
Inline variable initialization	`dim x as Integer = 5` → compile error	Declare and assign separately: `dim x as Integer` `x = 5`
Using `Trace()` for debug output	Compile error (does not exist)	Use `Println "message"`
Using `GetVariableString()`	Compile error (does not exist)	Use bracket syntax: `map["key"]`

Copyright © 2026 Vizrt

Mistake	Symptom	Fix
Forgetting to register SHM callback	`OnSharedMemoryVariableChanged` never fires	Call `RegisterChangedCallback("key")` in `OnInit`
Wildcard callback in production	Performance degradation, hundreds of calls per frame	Register specific keys, not `"*"`
Not checking `FindSubContainer` result	Null reference crash	Guard with `If c.VizId > 0 Then`
Setting only `Material.Color.Alpha`	Container still opaque (classic materials)	Also set `Material.Alpha` separately
Calling `SaveToDb` On Air	Renderer freeze / dropped frames	Only use during offline scene setup
Color values 0–255 instead of 0.0–1.0	White or clipped colors	Divide by 255.0: `CDbl(r) / 255.0`

22.4.16 API Quick Reference

Shared Memory

Method / Property	Description
`Scene.Map["key"]`	Read/write scene-scope SHM
`System.Map["key"]`	Read/write Viz Engine-scope SHM
`VizCommunication.Map["key"]`	Read/write distributed SHM (cluster-wide)
`.RegisterChangedCallback(key)`	Subscribe to changes (`"*"` = wildcard)

Copyright © 2026 Vizrt

Method / Property	Description
`.UnregisterChangedCallback(key)`	Unsubscribe from changes
`.SaveToDb(path)`	Persist SHM to Graphic Hub (blocks renderer)
`.LoadFromDb(path)`	Restore SHM from Graphic Hub (blocks renderer)

Container

Property / Method	Description
`this`	Current container
`this.Active`	Show/hide (Boolean)
`this.Name`	Container name (String)
`this.VizId`	Object ID (0 = invalid/not found)
`this.Position.X/Y/Z`	World position
`this.Rotation.X/Y/Z`	Euler rotation (degrees)
`this.Scaling.X/Y/Z`	Scale multiplier (1.0 = default)
`.Geometry.Text`	Text content (text geometry only)
`.Material.Color`	Diffuse color (Color)
`.Material.Alpha`	Transparency (Double, classic materials)
`.Material.Specular`	Specular highlight (Color)
`.Material.Emission`	Self-illumination (Color)

Copyright © 2026 Vizrt

Property / Method	Description
`.Material.Shininess`	Specular tightness (Double)
`.FindSubContainer(name)`	Find child by name
`.ParentContainer`	Parent in tree hierarchy
`.GetDirector()`	Director controlling this container
`Scene.FindContainer(name)`	Scene-wide search by name
`Scene.RootContainer`	Top of the scene tree

Director

Method	Description
`Show(time)`	Seek to time position (seconds)
`StartAnimation()`	Play from beginning
`StopAnimation()`	Pause at current position
`ContinueAnimation()`	Resume forward
`ContinueAnimationReverse()`	Resume backward

Copyright © 2026 Vizrt

22.4.17 Event Pool

A new event notification system allows different objects to emit and/or listen for events. The event pool tracks all registered objects and possible events from the command line. To enable debugging of the new event system, send the following command:

```
// activate / deactivate debugging (prints message on console for each fired event):
send EVENT_POOL*DEBUGGING SET 1
```

Other commands allow the event pool to fetch a list of registered objects which emit events:

Generic Viz Engine Commands

```
// get list of all objects which have registered some events (which may be fired at
some time)
send EVENT_POOL*REGISTERED_OBJECTS GET

// get list of all registered events of a specific object
send EVENT_POOL*REGISTERED_EVENTS GET $objectname
// example:
send EVENT_POOL*REGISTERED_EVENTS GET #727*STAGE

// get list of all registered objects+events and number of registered callbacks:
send EVENT_POOL LIST
```

Built-In Events

For now, the following objects have built-in events:

- **Stage:** (*onStarted*, *onStopped*, *onDirectorStarted*, *onDirectorStopped*)
- **Superchannels:** (*onArmed*, *onDisarmed*, *onTransitionStarted*, *onTransitionStopped*) and its subchannels (*onAssigned*, *onLoaded*, *onCleared*)
- **Gfx channels:** (*onPreSetObject*, *onPostSetObject*)
- **Editors:** (*onPreSetObject*, *onPostSetObject*)
- **Scripts:** (defined by the script itself)

Additionally, Super Channels and Scripts can be configured to listen on events.

Technical Notes

Generally, an event can have an arbitrary name (for example, *onLoaded*) and can be sent by any object. Scripts and objects can also listen for specific events and are notified whenever that event has been triggered.

Additional notes:

- Potential events must be registered by sending objects before they are emitted (because listeners can only register to **known** events).

Copyright © 2026 Vizrt

- The name of an object can be chosen by the object itself on creation (for example, *SUPERCHANNEL1* or *script1*) but it is always preceded by the object id of the scene. Example: *#199*SUPERCHANNEL1*.

> **Note:** Aliases like *MAIN_SCENE* are not allowed.

- When registering for an event, wildcards are allowed. The object name, the event name, or both can be wildcarded:

```
' Register for all events called "onLoaded"
eventpool.registerAsListener("*", "onLoaded")

' Register for all events from a specific sender:
eventpool.registerAsListener("#" & sceneid & "*SUPERCHANNEL1/A", "*")

' Register as listener for all events
eventpool.registerAsListener("*", "*")
```

- A KeyValueMap (Stringmap for scripts) can be attached to an emitted event.

Sample Scripts

Scripts can be used to emit events and to listen for specific events. See the sample scripts below.

Emit Events

Sample Script to Emit Events

```
sub OnInit()
Dim events as Array[String]
events.push("OnTriggered")
Eventpool.registerEvents("script1", events)

' use commands EVENT_POOL*REGISTERED_OBJECTS GET and
' EVENT_POOL*REGISTERED_EVENTS GET sceneID*script1 to
' verify that events are registered when script is active
' and that they are automatically unregistered when script
' is stopped/modified

' use EVENT_POOL*DEBUGGING SET 1 to
' see which events have been fired
end sub

sub OnInitParameters()
RegisterPushButton("test-button", "press here", 1)
end sub

sub OnExecAction(buttonId as Integer)
if (buttonId = 1) then
```

Copyright © 2026 Vizrt

```
Dim params As StringMap

' now emit our event
Eventpool.emitEvent("OnTriggered", params)

' this fails, because we have not registered the event:
'Eventpool.emitEvent("OnBlabla", params)

end if
end sub
```

Listen for Events

Sample Script to React on Events

```
sub OnInit()
dim sceneid as String
sceneid = cstr(scene.vizid)
' Register as listener for three events of Super Channel1 / subchannel A
eventpool.registerAsListener("#" & sceneid & "*SUPERCHANNEL1/A", "onLoaded")
eventpool.registerAsListener("#" & sceneid & "*SUPERCHANNEL1/A", "onAssigned")
eventpool.registerAsListener("#" & sceneid & "*SUPERCHANNEL1/A", "onCleared")

'' Wildcards are also possible
' Register for all events called "onLoaded"
' eventpool.registerAsListener("*", "onLoaded")

' Register for all events from a specific sender:
' eventpool.registerAsListener("#" & sceneid & "*SUPERCHANNEL1/A", "*")

' Register as listener for all events
' eventpool.registerAsListener("*", "*")

end sub

sub OnEvent(objectName As String, event as String, arguments as StringMap)
'println("GOT EVENT!")
'println(objectName)
'println(event)
'println(arguments)

if (event=="onCleared") then
    material.color.red = 255
    material.color.green = 0
elseif (event=="onAssigned") then
    material.color.red = 100
    material.color.green = 100
elseif (event == "onLoaded") then
    material.color.red = 0
    material.color.green = 255
```

Copyright © 2026 Vizrt

```
end if
end sub
```

Copyright © 2026 Vizrt

23 Shared Memory

The information in this section relates to the *VizCommunication.Map* Data Sharing mechanism. A local *VizCommunication.Map* in each Viz Engine (as part of a cluster), collects and stores data. This data can be internal data, like a scene script pushing data to the map, or data from external control applications through TCP or UDP.

This section contains information on the following topics:

- Data Sharing
- Shared Memory Synchronization
- Shared Memory Snapshots
- Shared Memory External Data Input

23.1 See also:

- Viz Engine REST Interface – read and write the shared memory maps over HTTP (`/api/v1/shm/...`).

Copyright © 2026 Vizrt

23.2 Data Sharing

- VizCommunication.Map
- Map Access through a TCP or UDP Interface
- Map Access through the Command Interface
- Map Access from within Plug-ins
- Script Examples
    - Example I
    - Example II
- Set and Retrieve Data from the SHM through Scripting
    - To Set a Value
    - To Retrieve a Value
    - To Split the Values into an Array
    - To Assign a Value to a Text Container
    - To Automatically Update Graphics
    - Receiving Notification on Map Changes
    - See also:

This section describes the data sharing mechanism provided by Shared Memory Maps (SHM). A SHM is a map which holds user defined variables indexed by a string. It is exposed to the script interface via the Shared Memory data type. The variables are stored as Variant objects and thus can hold objects of any other type.

There are three maps, and the only real difference between them is **scope** — who can see the value and how long it lives.

- **Scene.Map:** Referenced in plug-ins as Scene. This is the map local to the current Scene only. Every Scene has one map that can be used to exchange data among the scripts in the Scene.
- **System.Map:** Referenced in plug-ins as Global. The system-wide map allows for data sharing among the scenes currently loaded into memory (e.g. data sharing between scenes in different layers).
- **VizCommunication.Map:** Referenced in plug-ins as Distributed. A distributed map that enables data sharing among the Viz Engines connected to the same Graphic Hub. The Viz Engines must be logged in with users from the same Graphic Hub group.

	Scene	Global / System	Distributed
Script access	`Scene.Map`	`System.Map`	`VizCommunication.Map`
Command prefix	`<scene>*MAP` (e.g. `MAIN_SCENE*MAP`)	`GLOBAL*MAP`	`VIZ_COMMUNICATION*MAP`

Copyright © 2026 Vizrt

	Scene	Global / System	Distributed
Scope	One scene instance	Whole engine, all layers	All engines in the distribution
Lifetime	Cleared on scene unload	Until overwritten or engine restart	Until overwritten or restart
Speed	Fastest — local	Fast — local	Network latency in the path
Setup needed	None	None	Distribution must be configured
Pros	No name collisions; self-cleaning; no overhead	Crosses layers and scene changes; still local and fast	One write updates every engine; keeps multi-engine setups coherent
Cons	Invisible outside the scene; can't carry state across a scene change	Flat namespace shared by everything — collision risk; stale values persist	Latency and no frame-accurate ordering guarantee; bandwidth scales with write rate; facility-wide collision risk; fails silently if misconfigured
Use for	Script-to-script inside one scene — your default	Cross-layer state, handing data to the next scene	State that genuinely must match across engines
Avoid for	Anything another scene needs to read	Per-show data on a shared machine without a naming convention	A general-purpose message bus

The selection rule: use the narrowest scope that does the job — Scene, then Global, then Distributed. Every step outward buys reach and costs isolation, and in the distributed case, determinism.

Three gotchas worth knowing regardless of which you pick:

- Shared memory is untyped key/value. Nothing validates a key exists or holds what you expect, so a typo reads as empty rather than erroring.
- Values persist across scene loads in Global and Distributed. A field showing yesterday's value is almost always this.
- High-frequency writes are the usual performance culprit — updating on every field when every few frames would do, particularly over distribution.

Copyright © 2026 Vizrt

23.2.1 VizCommunication.Map

The map exposed by the VizCommunication type is used to share data among the client computers connected to a Graphic Hub. Whenever a script on one client adds a new value or changes an existing value in the map, the change is propagated to the other clients via database messaging to update the local copy of each client's map. This does not necessarily mean that all local client maps are identical. When a client starts up after other clients have made changes to the *VizCommunication.Map*, it is not informed of those changes.

When data is modified on one Viz Engine through a script or through a plug-in, the data change is reflected on the other Viz Engines automatically. A use case could be a touch screen scene which modifies data, which is also used for HD-SDI Viz Engines or Viz Engines driving a Video Wall. This synchronization uses the Graphic Hub as a relay. Therefore, it is important that all Viz Engines which are to receive the data are connected to the same Graphic Hub, which use the same user or users that belong to the same group of users in Graphic Hub.

See Synchronization in Shared Memory (SHM).

23.2.2 Map Access through a TCP or UDP Interface

Elements in the distributed memory map (*VizCommunication.Map*) can be set (but not retrieved) through a TCP or UDP interface, exposed by Viz Artist. This option is particularly useful when writing external control applications. For more information about map access through a TCP or UDP Interface, see External Data Input in Shared Memory (SHM).

23.2.3 Map Access through the Command Interface

There is limited support for getting and setting map elements through a command interface. The maps are accessible through these locations:

- `<scene>*MAP` for the scene-specific map.
- `GLOBAL*MAP` for system-wide map.
- `VIZ_COMMUNICATION*MAP` for the distributed map.

A full list of supported commands is at: *<viz install folder>\Documentation\viz-cmd-doc-webhelp\index.html*.

From the list of commands, the commands `CLEAR`, `DELETE_ELEMENT` and `PURGE_ELEMENTS` only works when sent through the command interface of Viz Artist. The command `CLEAR` must be run on each Engine where the MAP is to be reset (`VIZ_COMMUNICATION*MAP CLEAR`).

> **Note:** The difference between `DELETE_ELEMENT` and `PURGE_ELEMENTS` is that `PURGE_ELEMENTS` deletes multiple elements that have the same prefix. For example, `PURGE_ELEMENTS Key` deletes keys named *Key1*, *Key2* and *Key3*.

Copyright © 2026 Vizrt

23.2.4 Map Access from within Plug-ins

Functions for getting and setting map elements are provided by the plug-in interface. For more information, refer to the plug-in API documentation in your Viz installation folder.

23.2.5 Script Examples

This section contains two script examples for Shared Memory (SHM):

Example I

This is an example that shows how a Scene map can be used to pass data from one script to another (within the same Scene).
This script defines the *OnEnterSubContainer*, which is called whenever the mouse-cursor enters any of its sub-containers. In it, the name of the subcontainer is stored in the Scene map under the key subcontainer_name.

```
‘ Script 1:
Sub OnEnterSubContainer(subContainer as Container)
Scene.Map["subcontainer_name"] = subContainer.Name
End Sub
```

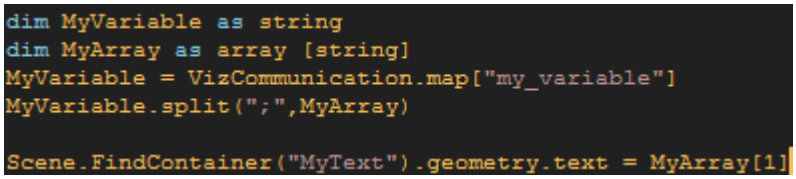
Example II

This example shows how external data can be sent to all Viz Engines connected to a Graphic Hub, and how they can utilize the same data. The example also shows how to group together related external data into an array inside the Viz Engine. This is not the only way of managing variables. They could just as well be sent individually, which does not demand extra variable splitting in a script.
Vizrt do not recommend this as a method for the import of large amounts of data. For more information see External Data Input.

To create and set a value shared by all Viz Engines, that are connected to the same Graphic Hub, send the following command through the Viz Engine Console:

```
send VIZ_COMMUNICATION*MAP SET_STRING_ELEMENT "my_variable" 1.2;10.8;20.3;15.9
CONSOLE: answer <>
send VIZ_COMMUNICATION*MAP GET_STRING_ELEMENT "my_variable"
CONSOLE: answer <1.2;10.8;20.3;15.9>
```

This example shows a command sent to the Viz Engine, that assigns a value to a variable named my_variable. This image shows a script that uses this variable, and assigns one of its values to a text container named *MyText*:



Copyright © 2026 Vizrt

From outside a Viz Engine the shared memory can be accessed by a connection to the Viz Engine listener port `6100`. For more information on ways to access shared memory, see Map Access through the Command Interface.

23.2.6 Set and Retrieve Data from the SHM through Scripting

To Set a Value

Through a Command Console:

```
send VIZ_COMMUNICATION*MAP SET_STRING_ELEMENT "my_variable" 1.2;10.8;20.3;15.9
```

or through Script:

```
VizCommunication.map["my_variable"] = "1.2;10.8;20.3;15.9"
```

To Retrieve a Value

Dim MyVariable as string

```
MyVariable = (string)VizCommunication.map["my_variable"]
```

Here, string is used to format the data from the SHM to a text value.

To Split the Values into an Array

You can then split the data inside MyVariable into a table called MyArray:

```
Dim MyArray as array [string]
MyVariable.split(";",MyArray)
```

The information in MyVariable has now been split into four:

```
‘ MyArray[0] = 1.2
‘ MyArray[1] = 10.8
‘ MyArray[2] = 20.3
‘ MyArray[3] = 15.9
```

To Assign a Value to a Text Container

One of these values can then be assigned to a text container:

Copyright © 2026 Vizrt

```
Scene.FindContainer("MyText").geometry.text = MyArray[1]
```

To Automatically Update Graphics

When a SHM variable is updated you need to register the value for updates in the scene:

```
Scene.Map.RegisterChangedCallback("my_variable")
```

Changes to `my_variable` execute a subroutine called `OnSharedMemoryVariableChanged`. This subroutine can do whatever you decide with the data. For further details, refer to the callback procedures described in the plug-in API documentation.

Receiving Notification on Map Changes

Sometimes you want to be notified when a value in an SHM changes, so you do not have to poll for changes within `OnExecPerField`. This is done by calling a map's `RegisterChangedCallback` procedure, passing it the name of a key that you want to monitor. Now, whenever the variable identified by that key is changed, the `OnSharedMemoryVariableChanged` callback (if present) is invoked. Similarly, the `OnSharedMemoryVariableDeleted` callback is invoked, when the variable is deleted.
Example:

```
Sub OnInit()
    Scene.Map.RegisterChangedCallback("subcontainer_name")
End Sub

Sub OnSharedMemoryVariableChanged(map as SharedMemory, mapKey as String)
    If mapKey = "subcontainer_name" Then
        Geometry.Text = (String)map["subcontainer_name"]
    End If
End Sub
```

`OnSharedMemoryVariableChanged` is passed to the map as well as the key whose variable has been changed.

Passing an empty string (`""`) to `RegisterChangedCallback` makes sure that `OnSharedMemoryVariableChanged` and `OnSharedMemoryVariableDeleted` is invoked whenever any variable in the map is modified or deleted:

```
Sub OnInit()
Scene.Map.RegisterChangedCallback("")
End Sub
Sub OnSharedMemoryVariableChanged(map as SharedMemory, mapKey as String)
...
End Sub
```

Copyright © 2026 Vizrt

See also:

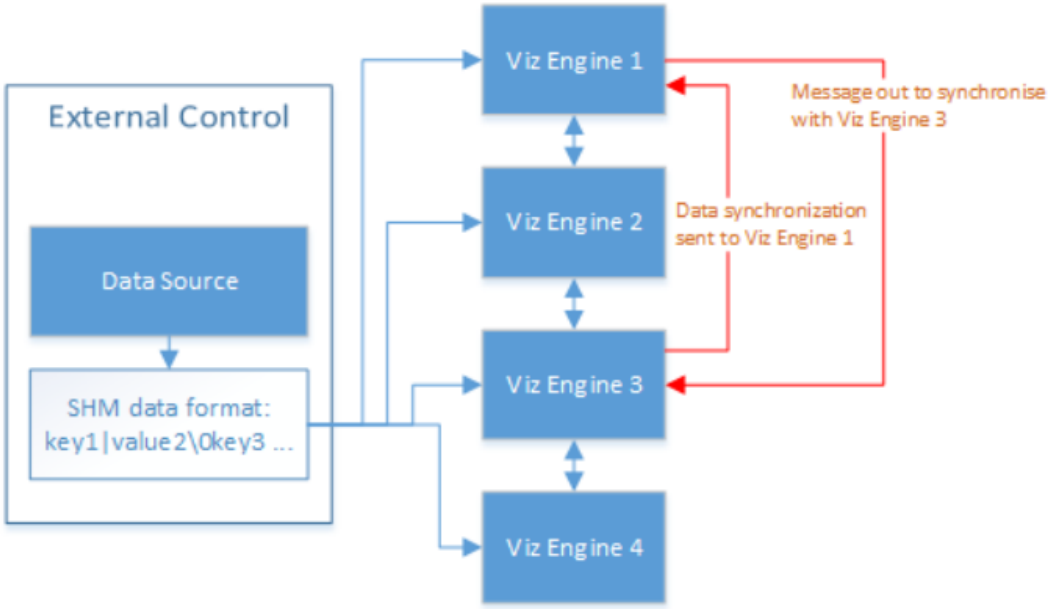
- Viz Engine REST Interface – read and write the shared memory maps over HTTP (`/api/v1/shm/...`).

Copyright © 2026 Vizrt

23.3 Shared Memory Synchronization

If a Viz Engine is restarted or added to the cluster of Viz Engines for playback, the *VizCommunication.Map* data is not available on this Viz Engine. The local map on a new or restarted Viz Engine can be synchronized with the local map on another running Viz Engine in the same cluster.

Example: If Viz Engine 1 restarts it look to, for example, Viz Engine 3 to update its local *VizCommunication.Map.*



A startup Viz Engine can be synchronized through:

- TCP
- UDP
- An External Control Application
- Commands

This section contains information on the following topics:

- TCP and UDP Synchronization
- External Control Synchronization
- Command Synchronization

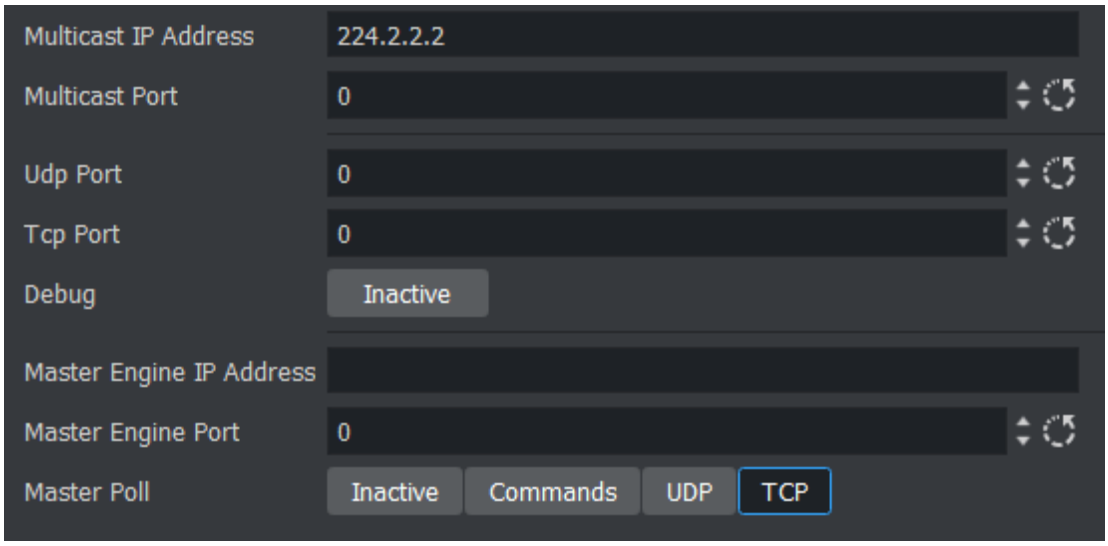
23.3.1 TCP and UDP Synchronization

Use the procedure detailed below to synchronize a restarted or added Viz Engines, in a cluster, with the TCP or UDP protocol:

Copyright © 2026 Vizrt

- **TCP:** The recommended communication protocol to synchronize maps on startup because it is reliable and efficient (see TCP Protocol).
- **UDP:** For fast communication, but has drawbacks. For instance, packets can get lost (see UDP Protocol).

To Synchronize a Viz Engine with TCP or UDP



1. Open Viz Configuration.
2. Click on **Communication**.
3. Click on the **Shared Memory Properties** tab.
4. Enter the port number for either **UDP** or **TCP Port**.
5. In **Master Engine IP Address**, type the IP address of a running Viz Engine to synchronize with. This will be the IP address of a running Viz Engine in the same cluster system.
6. In **Master Engine Port**, type the port number to be used (default `6100`).This port sends the single startup synchronization command to start synchronizing. It should be the same as the general communication port of the Viz Engine set in **Master Engine IP Address**.
7. In **Master Poll**, select either **UDP** or **TCP**, corresponding to the set port.
8. Click **Save**.

Note: No specific port number is recommended. Always make sure that the selected port number is not in use by any other program on the same subnet.

23.3.2 External Control Synchronization

Synchronization can also be done from an External Control Application. The following command has to be sent to the Viz Engine which has the memory map populated:

Copyright © 2026 Vizrt

From a Command Interface

```
VIZ_COMMUNICATION SYNCHRONIZE_TO <engine> <port>
```

where the <engine. is the Viz Engine which receives the data through the Command Interface. The port is usually `6100` (standard command interface port).

> **Note:** This is not recommended as a method for synchronization.

Through TCP Communication

```
VIZ_COMMUNICATION SYNCHRONIZE_SMMTCP <engine> <port>
```

where the <engine> is the Viz Engine which receives the data, and the port is the one that was specified for incoming TCP key|value pairs on the Viz Engine which is to receive the data (see TCP Protocol).

Through UDP Communication

```
VIZ_COMMUNICATION SYNCHRONIZE_UDP <engine> <port>
```

where the <engine> is the Viz Engine which receives the data, and the port is the one that was specified for incoming UDP key|value pairs on the Viz Engine which is to receive the data (see UDP Protocol).

23.3.3 Command Synchronization

Another communication option is by Commands. Here each map entry is synchronized through commands. It is reliable, but very slow and blocks Viz Engine for quite an amount of time if the map is large.

> **Note:** This is not recommended as a method for synchronization.

> **Note:** The Viz Engine versions need to match.

To Synchronize a Startup Viz Engine with Commands

1. Open Viz Configuration.
2. Click on **Communication**.
3. Click on the **Shared Memory Properties** tab.
4. In the Shared Memory panel set these parameters:
    - **Master Engine Port:** Enter the communication port of the primary Viz Engine (default is `6100`).
    - **Master Poll:** Click on **Commands**.

Copyright © 2026 Vizrt

5. Click **Save**.

Copyright © 2026 Vizrt

23.4 Shared Memory Snapshots

You can take a data snapshot (save the whole content of the map) at any time. One Viz Engine can save the whole content of the map by calling the map's `SaveToDb` procedure, and another client can read it by calling `LoadFromDb`.

The downside of this approach is that these functions block the renderer and can cause poor performance of the Graphic Hub database if the map is stored repeatedly.

IMPORTANT! The resulting SHM map objects are replicated as well. So saving the map each field or every minute can result in serious problems for the Graphic Hub database (replication failing, etc.).

Note: For more information see the script function documentation. Go to **Start > All Programs > vizrt > Script Documentation > Data types and procedures > SharedMemory**, or ***<viz install folder>* > ScriptDoc > DataTypeSharedMemory.**

See Also

- External Data Input
- Synchronization

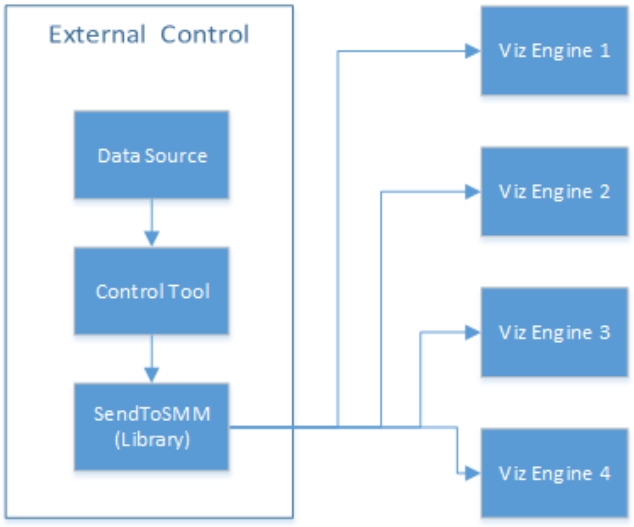
Copyright © 2026 Vizrt

23.5 Shared Memory External Data Input

Data to **Shared Memory** should be fed through the dedicated UDP or TCP IP ports for SHM. Vizrt provides a set of development components called *SendToSMM* to make developing tools for this purpose easier. The component files are included with the standard Viz installation and located at *<viz install folder>\Tools\SendToSMM*.

Data can also be sent to shared memory through the Command Interface. Data sent through the Command Interface may be seen as a good option because the data would need to be sent to one Viz Engine only, and this Viz Engine engine would then distribute the data to the other Viz Engines. But Data sent to shared memory through the Command Interface has problems:

- Data sent through the Command Interface blocks the render queue of the receiving engine causing potential frame drops. Since the data needs to be sent through a command significant more bytes are transferred over the Network.
- This Viz Engine is also a single point of failure.
- The data arrives at this one Viz Engine sooner than on all other Viz Engines.
- The notification method of the Graphic Hub is used to distribute the data and can cause additional load for the Graphic Hub.
  The preferred method to send data is to use the *SendToSMM* library (or an equivalent) to send the data to the individual Viz Engines.



The communication protocol for the import of Shared Memory data depends on the type and final output of the data. There are set protocols to use with large amounts of data, in which all of the data must reach its destination graphic, and also where large amounts of data must be received, quickly, but some loss of data is acceptable.

For use-cases, see:

- TCP Protocol
- UDP Protocol
- Plug-in API
- Command Interface
- REST Interface

Copyright © 2026 Vizrt

Note: It is also possible to import data through Multicast. This method is not recommended as it can pollute the network.

23.5.1 TCP Protocol

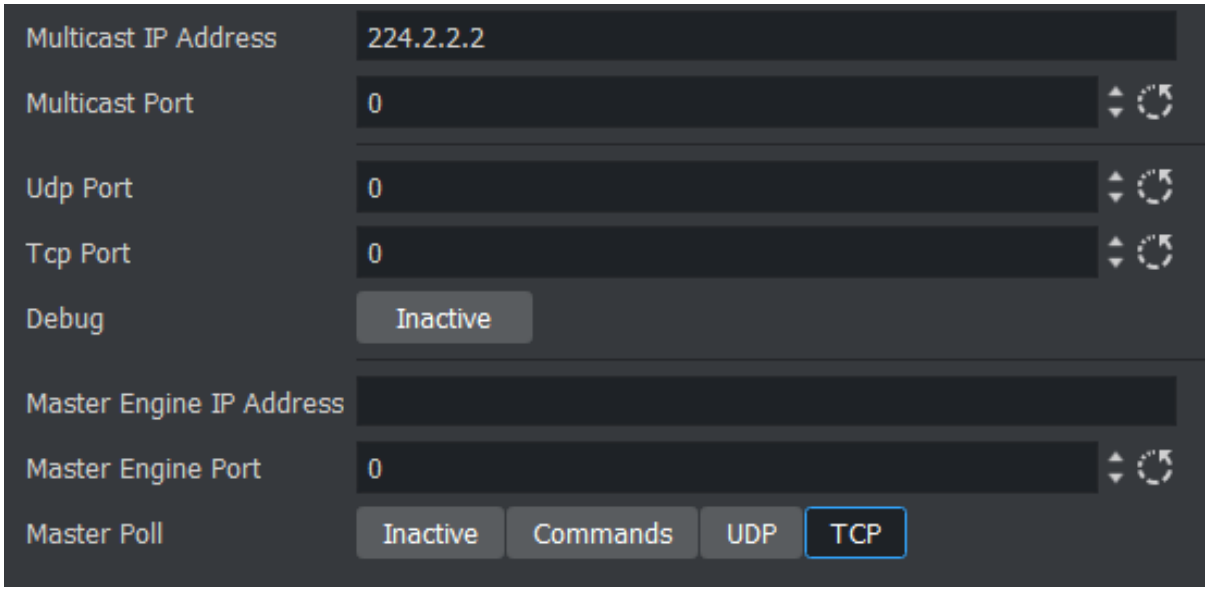
The SHM TCP communication protocol guarantees a reliable delivery of packages. It is a much more efficient than the Command Interface, but not as fast as the UDP Protocol. Use cases for a TCP connection could be finance stocks and currencies, or election result information, where the requirement is to deal with large amounts of information, and all of this data must reach its destination graphic. A single piece of lost data can have economic consequences, falsify charts, show mathematically wrong results, etc.

A TCP connection to a Viz Engine can be held open for a long time (this is recommended), and should not be opened and closed between sending variables.

Note: The default number of connections is `1` and the maximum number of TCP Shared Memory connections is limited to `255`. Within this number of connections, a user-defined limit of maximum connections can be set in the config file with *smm_thread_count* (see **To Limit the Number of TCP Connections** in the **Configuring Viz** section of the Viz Engine Administrator Guide).

IMPORTANT! The external program which provides the data, must connect and send the data to each Viz Engine individually. Vizrt provides a C# library for this purpose, called *SendToSMM*, as part of the Viz installation.

To Use TCP for Shared Memory



1. Open Viz Configuration.
2. Click on **Communication**.

3. Click on the **Shared Memory Properties** tab.
4. In the Shared Memory panel set the **TCP Port** parameter. A specific port is not recommended. Always make sure that the selected port is not in use by any other program on the same subnet.
5. Click **Save**.

   When accessing the *VizCommunication* Memory Map, the syntax for sending the key-value pairs is `<key>|<value>`, followed by a null-termination character: `key|value\0`

Note: `Key` and `value` are divided by the pipe character, also known as vertical bar: **|**

- Multiple key-value pairs can be sent at once. This requires that each pair is null-terminated:
- `key1|value1\0key2|value2\0`

  When accessing a *Scene* or *System* shared memory map, the syntax requires the inclusion of further information.

Example: When sending multiple key-value pairs to system shared memory the syntax should be *SMMSystem_SetValue|key1|value1\0SMMSystem_SetValue|key2|value2\0* and in this case if *SMMSystem_SetValue* is not specified before each key-value pair, only the first one is sent to the system shared memory while the rest of sent pairs are sent to VizCommunication shared memory.

- Syntax for accessing the VizCommunication shared memory map:
  - `SharedMemoryMap_SetValue|<key>|<value>\0`
  - `SharedMemoryMap_SetValueDistribute|<key>|<value>\0`
  - `SharedMemoryMap_CreateElement|<key>\0`
  - `SharedMemoryMap_DeleteElement|<key>\0`
  - `SharedMemoryMap_SetPartialValue|<key>|<part>|<number-parts>|<value>\0`
    - Used to send long values over UDP where one command would exceed the UDP-Package size. With this command, the value can be sent in multiple parts.
    - Current part index <part>.
    - Total number of parts <number-parts>.
    - Example: sending *SharedMemoryMap_SetPartialValue|key1|0|2|123\0* and *SharedMemoryMap_SetPartialValue|key1|1|2|456\0* would set the key 'key1' = 123456.
- Syntax for accessing the Scene shared memory map:
  - `SMM#<scene object id>_SetValue|<key>|<value>\0`
  - `SMM#<scene object id>_CreateElement|<key>\0`
  - `SMM#<scene object id>_DeleteElement|<key>\0`
- Syntax for accessing the System shared memory map:
  - `SMMSystem_SetValue|<key>|<value>\0`
  - `SMMSystem_CreateElement|<key>\0`
  - `SMMSystem_DeleteElement|<key>\0`

Note: A C++ library to write to shared memory maps and an example is included with the local viz installation: *<viz installation location>\Tools\VizControlLib*.

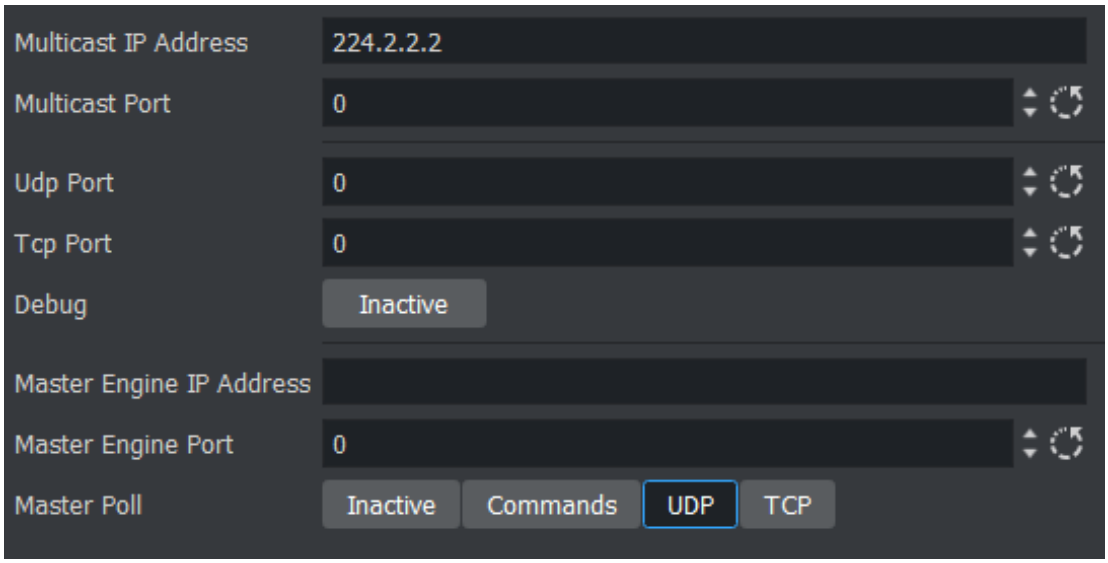
Copyright © 2026 Vizrt

23.5.2 UDP Protocol

The SHM UDP communication protocol should be used for the delivery of volatile data. It is quicker than the TCP Protocol, but less reliable, and is much more efficient than the Command Interface.

A use case for UDP would be Motor Sports, where data like speed, velocity, etc., is required. This is where there is a requirement to deal with large amounts of data, but not all of this data must reach its destination. A single piece of data lost does not affect the constant data update.

To Use UDP for SHM



1. Open Viz Configuration.
2. Click on **Communication**.
3. Click on the **Shared Memory Properties** tab.
4. In the Shared Memory panel set the **UDP Port** parameter. A specific port is not recommended. Always make sure that the selected port is not in use by any other program on the same subnet.
5. Click **Save**.

The syntax for sending key-value pairs is the same as for TCP and UDP Synchronization.

23.5.3 Plug-in API

An option to manipulate data in SHM is by a plug-in interface. A use case would be where a TCP or UDP connection cannot be used, or is not to be used. It is possible to write a plug-in to import data (e.g. from an XML file, another database, etc.) and push it to SHM. Another use case would be an interactive Scene.

Note: The plug-in API is documented in the plug-in API documentation and comes with the Viz installer (go to, **Start > All Programs > vizrt > Viz x.x > Documentation > plug-in SDK Documentation > Classes > Class List > Shared_Memory**).

Copyright © 2026 Vizrt

23.5.4 Command Interface

For small and single value changes, the Command Interface of Viz Artist can be used, for example, to update the headline in a Scene.

IMPORTANT! A command operation can block the renderer. If there are too many commands within a short time-frame, or if commands containing a large amount of data is being sent, this can result in not rendering real-time anymore.

Note: This is not recommended as a method for synchronization.

Any external program should consider the performance of the single or all connected Viz Engines. If there is a burst of thousands of SHM variables this can have implications on the Viz Engine rendering performance (Current (CUR) and Maximum (MAX)). A full list of commands is available at: *<viz install folder>\Documentation\viz-cmd-doc-webhelp\index.html*.

Note: From the list of commands, the commands `CLEAR`, `DELETE_ELEMENT` and `PURGE_ELEMENT` only works when sent through the command interface of Viz Artist.

IMPORTANT! The command `CLEAR` must be run on each Engine where the MAP is to be reset (`VIZ_COMMUNICATION*MAP CLEAR`).

Whenever a new entry is made in the map (a new key-value pair) or values are changed, then the change is propagated to the other Viz Engines through a database messaging service to update the local copy of each Viz Engine's map (this only works when sent over the general communication port of Viz Engine (default port `6100`)).

Command Examples

`VIZ_COMMUNICATION*MAP` can be used to access the map.

`SET_DOUBLE_ELEMENT` and `GET_DOUBLE_ELEMENT`

Example: `VIZ_COMMUNICATION*MAP SET_DOUBLE_ELEMENT "my_double" 1.2`

Example: `VIZ_COMMUNICATION*MAP GET_DOUBLE_ELEMENT "my_double"`

23.5.5 REST Interface

If the Viz Engine REST Interface is enabled, all three shared memory maps can also be read and written over HTTP:

Copyright © 2026 Vizrt

Map	Command location	REST route
System.Map	`GLOBAL*MAP`	`/api/v1/shm/system`
VizCommunication.Map	`VIZ_COMMUNICATION*MAP`	`/api/v1/shm/distributed`
Scene.Map	`<scene>*MAP`	`/api/v1/shm/<scene uuid>` (UUID without brackets)

Append `/<key>` to address a single value, e.g. `http://localhost:61000/api/v1/shm/system/<key>`.

- On a map: `GET` reads it, `POST` adds a key, `PUT` replaces the whole map.
- On a key: `GET` reads it, `PUT` sets it, `DELETE` removes it.

Request bodies and status codes are described in the generated reference at `http://<host>:<port>/api/v1/documentation`. An OpenAPI description is installed at `<viz install folder>\library\REST\openapi.json`.

Copyright © 2026 Vizrt

24 Third Party Applications and Files

This section details the Third Party applications that can be integrated with Viz Artist.

This section contains the following topics:

- Integration with Unreal Engine
- FBX Files
- CINEMA 4D
- Adobe After Effects
- NDI and TriCaster
- Photoshop Import
- Alembic Import

Copyright © 2026 Vizrt

24.1 Integration with Unreal Engine

This section explains how to set up Vizrt's Unreal Integration for Unreal Engine. This integration lets you use the render capabilities of the Epic Games' Unreal Engine and compose directly in Viz Engine for playout.

Surfaces from Unreal Engine can be added inside a Viz Engine scene and Media Assets from Viz Engine can be used inside the Unreal Engine environment. This provides professional Virtual Set productions and Augmented Reality graphics.

This section covers the following topics:

- Known Limitations
- Prerequisites
    - System and Licensing Requirements
    - Hardware Requirements
    - Software Requirements
- To Install the Media Plug-in
- Viz Engine Preparation
- Unreal Engine Preparation
    - Enable Vizrt Media for Unreal Engine
    - Licensing
    - Additional Editor Settings
    - To Establish Correct Syncing with Viz Engine
    - To Use the Vizrt Media plug-in Capture
        - Manual Configuration
        - VizrtAutoCapture
        - VizrtPawn
- Adding Media Assets from Viz Engine in Unreal Scenes
- Quality Settings (Unreal Engine Scalability Settings)
- Keep Old Frame vs Black Frame
- Keying and Compositing in Unreal
- Gamma Correction
- Syncing Unreal Project with Graphic Hub
- Using Viz Engine Control Objects
- Legacy Versions
- Troubleshooting
    - General
    - Hardware-accelerated GPU Scheduling

The Unreal Integration serves and provides surfaces from and to Viz Engine by using shared memory technology. This means that multiple Media Assets can be brought into Unreal Engine and surfaces rendered by Unreal can be captured and brought back into Viz Engine. You can attach a Vizrt Shared Texture to any surface and take advantage of Unreal's Cine Actor camera with all its post-processing effects.

24.1.1 Known Limitations

Starting from version 5.0 of the Unreal integration, it's now possible to have Unreal objects' shadows and reflections on top of Viz Engine input. It's also possible to use Unreal Engine effects which require an alpha channel

Copyright © 2026 Vizrt

(for example, bloom effects, glow, transparency) to be blended properly with a Viz Engine input by doing the composition in Unreal, using the Vizrt Compositor plug-in. However, there's **no guarantee** that all effects from Unreal Engine will be properly blended with Viz Engine.

24.1.2 Prerequisites

System and Licensing Requirements

- Valid WIBU license for Vizrt Media for Unreal Engine (Unreal Engine Render Blade).

Note: Starting with Unreal Engine 5.4, Epic Games requires companies to pay license fees if the annual gross revenue exceeds USD 1 million. Customers are directly responsible for reporting this to EPIC Games and purchasing relevant Unreal Engine licenses. Vizrt is not responsible for determining if the fees apply, nor will it handle the payment to Epic Games. More information on this topic from Epic Games can be found here: https://www.unrealengine.com/en-US/license
Vizrt will continue to provide best-in-class integration with the latest versions of Unreal Engine.

Hardware Requirements

- NVIDIA Quadro RTX6000 or better.
- 64 GB of memory.
- A certified Vizrt Render workstation like Lenovo P620 is recommended.

Software Requirements

- Windows 10 64-bit versions 2004 and 20H2 revision 789 or higher.
- Viz Engine version 5.0 or later.
- Epic Games Launcher (needed to install Unreal Engine).
- Unreal Engine and Viz Engine need to be installed on the same machine.

Information: Please note that Unreal Integration requires a valid WIBU license.

Information: For more details about Unreal Engine hardware/software requirements, please refer to Unreal Engine Requirements. For specific feature requirements refer to the section Unreal Engine Requirements for Rendering Features.

Refer to the following version compatibility table for an overview of supported versions of the different software components:

Viz Engine Version	Supported Unreal Engine Versions
5.5	5.6, 5.5
5.4	5.5, 5.4 (1)

Copyright © 2026 Vizrt

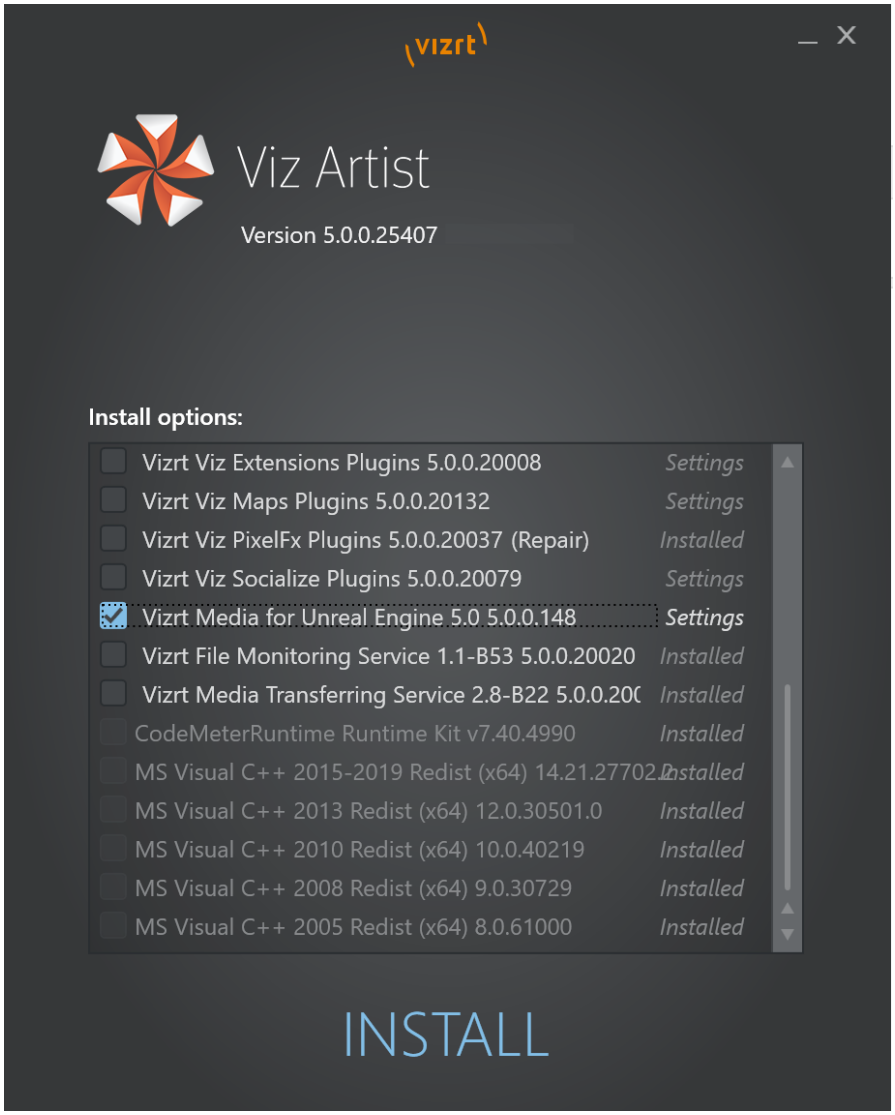
Viz Engine Version	Supported Unreal Engine Versions
5.3	5.4, 5.3
5.2	5.3, 5.2
5.1	5.1, 5.2
[1] Due to a technical limitation in Unreal Engine 5.6, Vizrt Media Plugin is not yet supported in UE 5.6.	

24.1.3 To Install the Media Plug-in

The Unreal Integration is part of the Bundle installer and also available as single *.msi* installer package. We recommend to use the Bundle Installer.

> **Note:** There are several installers for specific Unreal Engine versions. Please make sure the correct one is selected in the Bundle installer.

Copyright © 2026 Vizrt



After the installation is completed, the files are located in *C:\Program Files\Vizrt\VizEngine\Resources\Unreal\5.0.*

For all Vizrt Media plug-ins and assets to be available for:

- **A specific Unreal project:** Copy the contents of the source folder to the *Plugins* folder of the Unreal project.
- **All Unreal projects:** Copy the contents of the source folder to the *Plugins\Runtime* folder of the Unreal Engine installation.

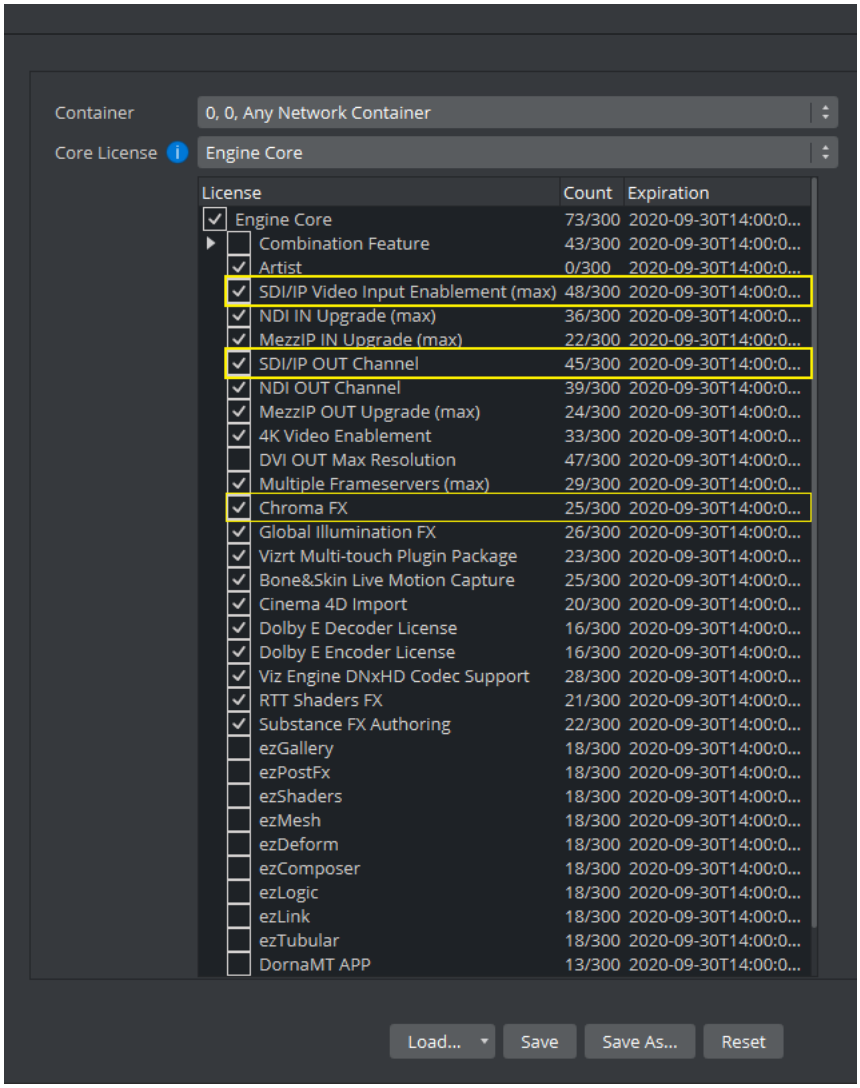
Source	Destination
C:\Program Files\Vizrt\VizEngine\Resources\Unreal\5.0	*C:\Program Files\Epic Games\UE_5.0\Engine\Plugins\Runtime* or *<UnrealProject folder>\Plugins*

Copyright © 2026 Vizrt

Caution: Keep in mind that any changes made to the Vizrt Media assets and sample scenes are shared across all the Unreal projects if the source folder is copied to the *Plugins\Runtime* folder of the Unreal Engine installation.

24.1.4 Viz Engine Preparation

To prepare Viz Engine for dealing with Unreal scenes, start Viz Engine in Configuration Mode and switch to the Licensing section. Make sure the following licenses are selected or available:



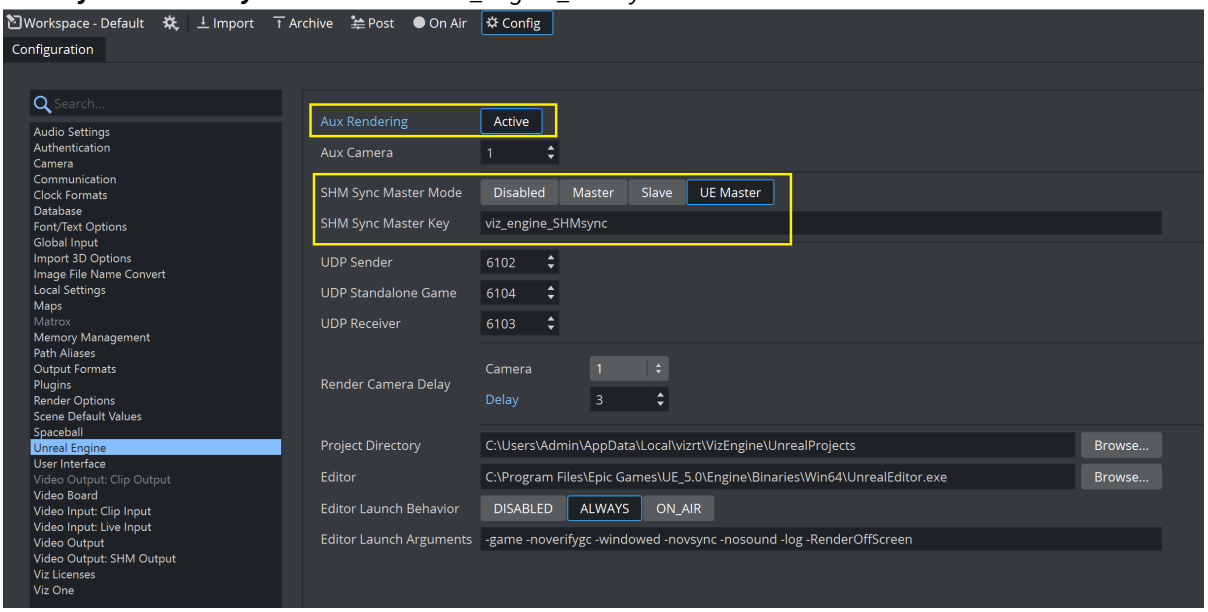
The following licenses are required:

- **SDI/IP Output Channel:** For general SDI/IP output of the final compositing.
- **Precision Keyer:** (Optional) For virtual sets that require the use of Precision Keyer.

Copyright © 2026 Vizrt

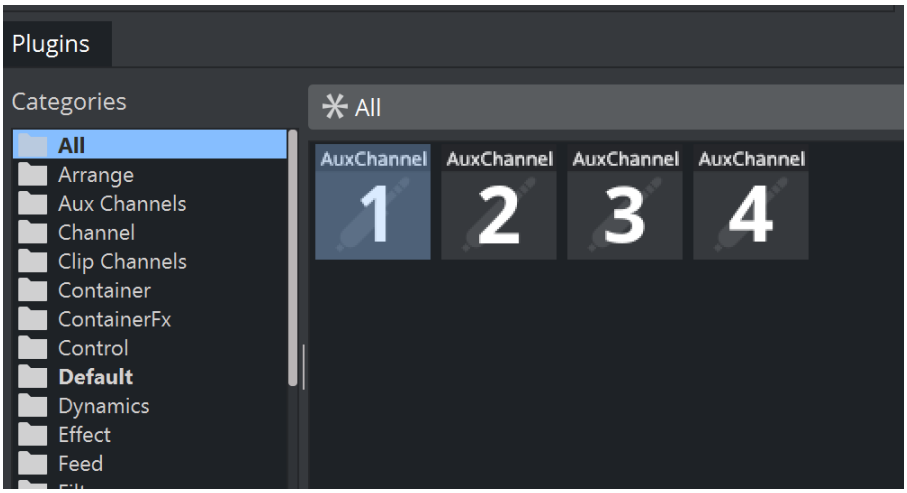
For Unreal output to be sent to Viz Engine, the following must be set in the Unreal Engine section of Viz Engine configuration:

- **Aux Rendering** should be *Active*.
- **SHM Sync Master Mode** should be set to *UE Master*.
- **SHM Sync Master Key** should be set to *viz_engine_SHMsync*.

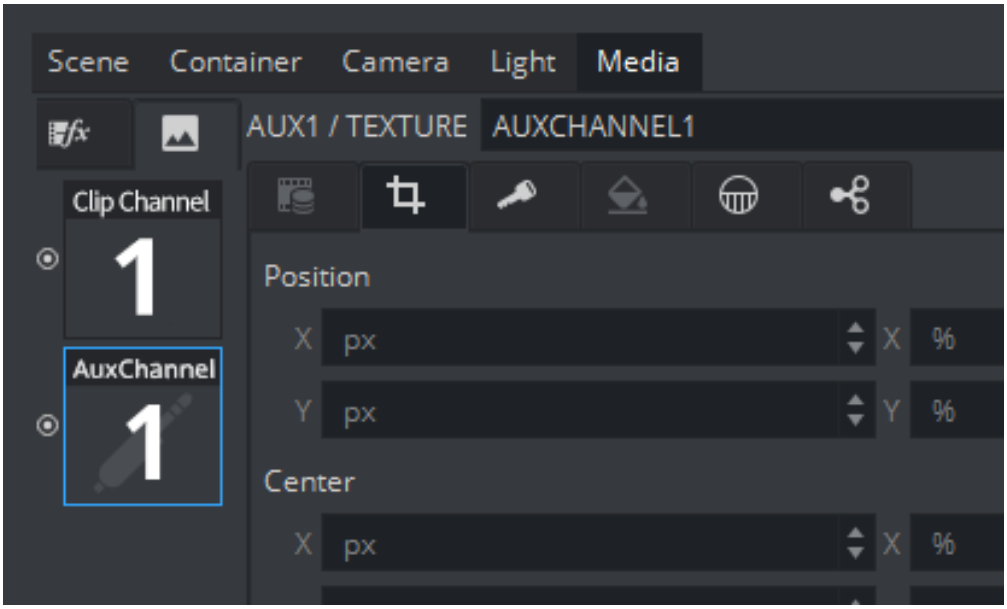


Save the configuration. Restart Viz Engine and make sure it is locked to a proper sync signal. The Unreal Engine sends data to Viz Engine using Aux channels. The following shows AuxChannel1 being used, but any AuxChannel can be used as long as it matches the AuxChannel configured in Unreal.

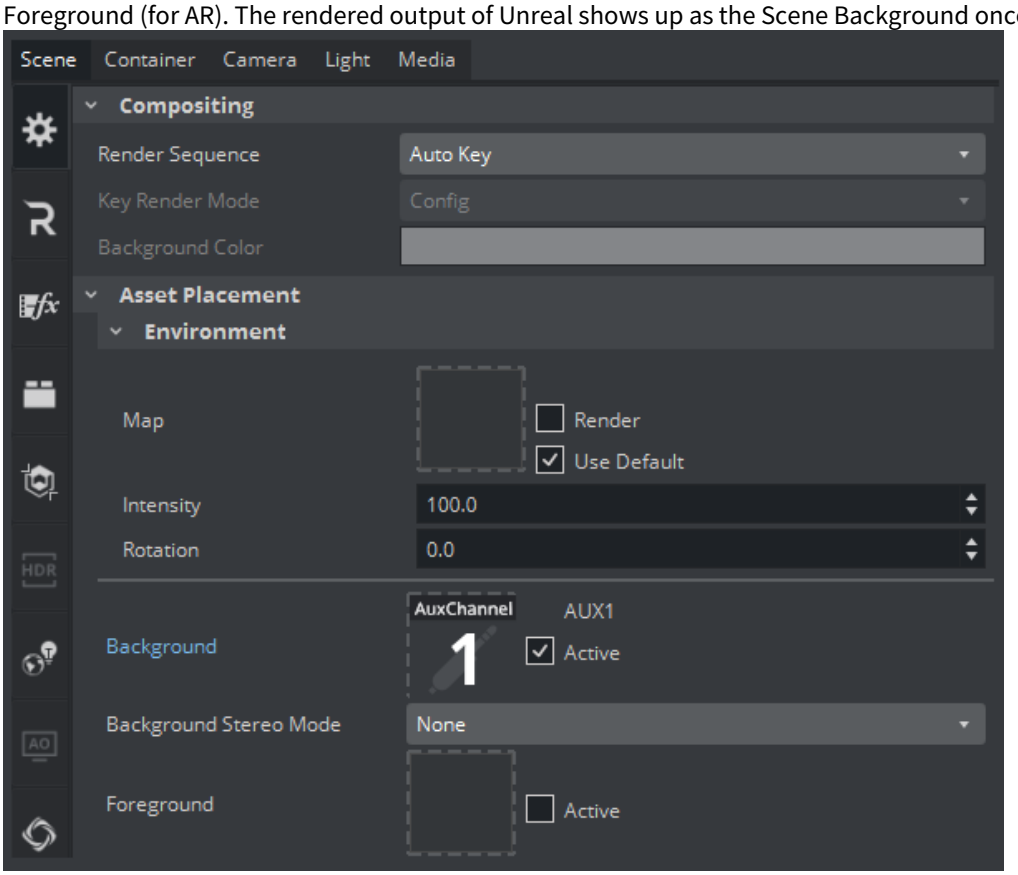
1. In Viz Artist, create an empty scene. Switch to the Render Pipeline you need.
2. The Render Sequence should be set to **External VS** (for Virtual Sets) or **External AR** (for Augmented Reality).
3. Add AuxChannel1 to the Media tab.



Copyright © 2026 Vizrt



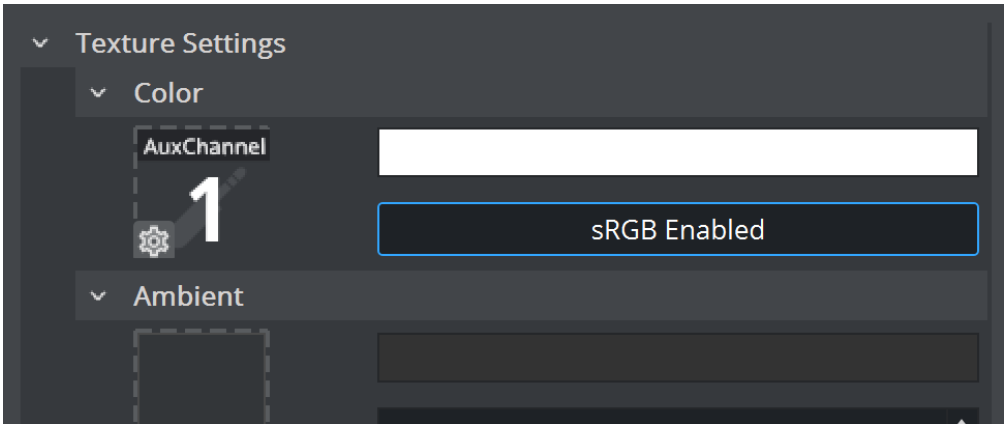
4. Add the AuxChannel1 onto the Scene Background property in Scene Settings. (for Virtual Sets) or Foreground (for AR). The rendered output of Unreal shows up as the Scene Background once Unreal is setup.



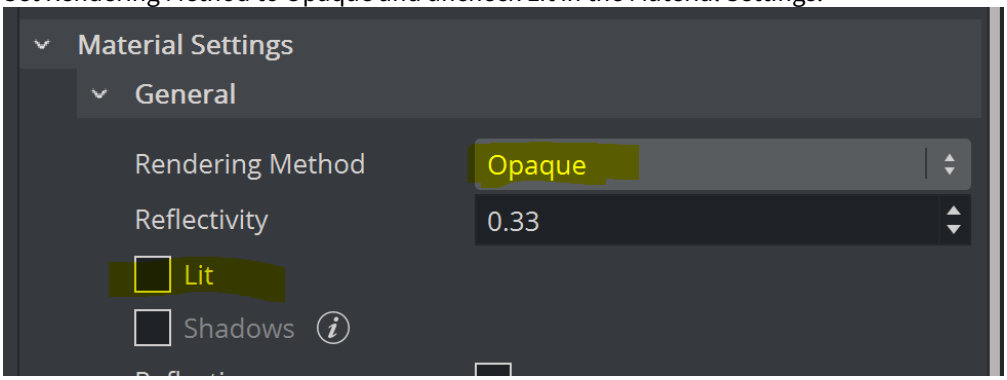
The output from Unreal can also be applied to a Viz Engine material (for example a Phong material).

1. Drag the AuxChannel to the Color property of the Material.

Copyright © 2026 Vizrt



2. Set Rendering Method to Opaque and uncheck Lit in the Material Settings.

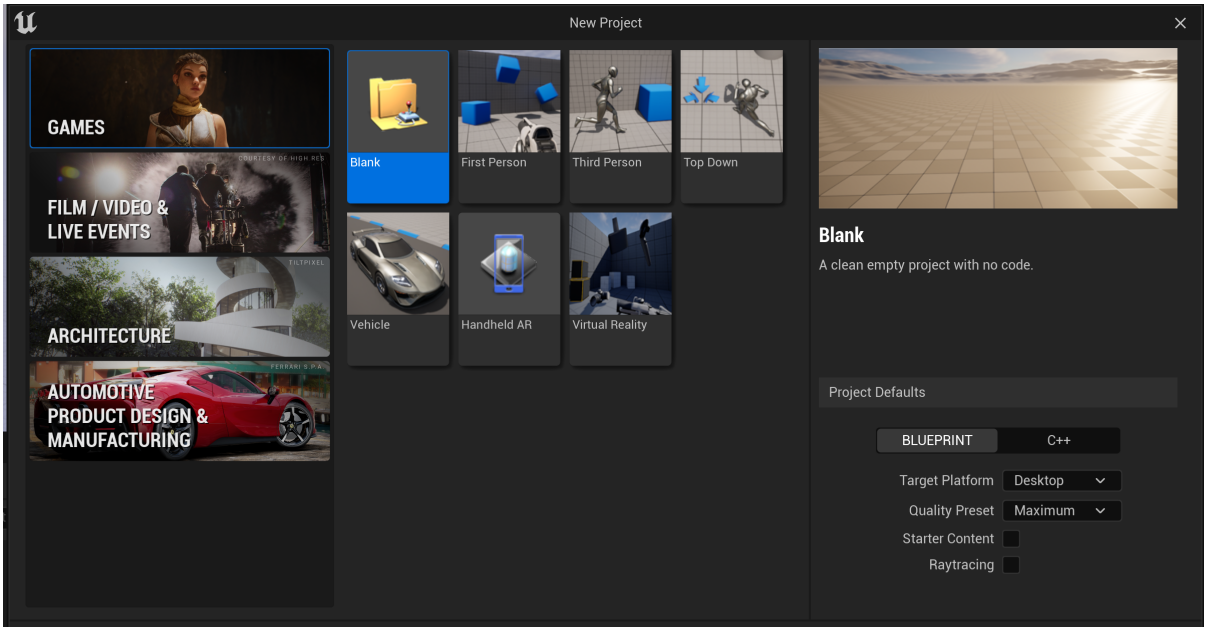


24.1.5 Unreal Engine Preparation

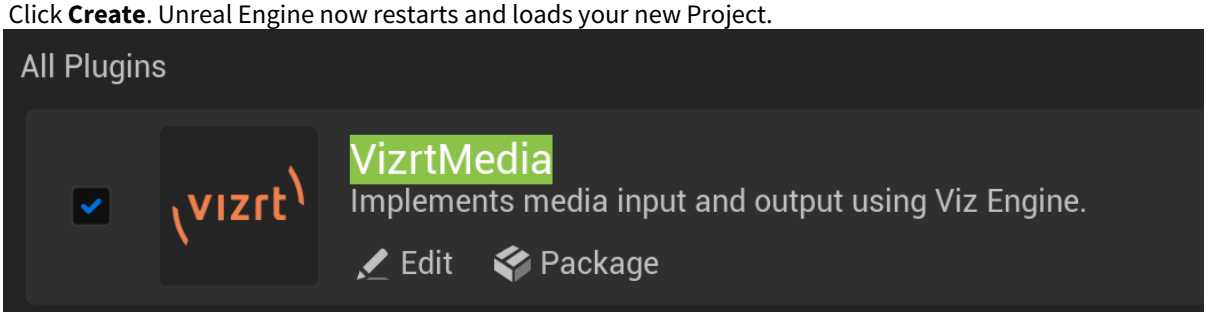
Enable Vizrt Media for Unreal Engine

1. Start Unreal Engine. Now you can either load your existing project or create a new one based on a template.
2. Now choose a **Blank** project and uncheck **Starter Content**.

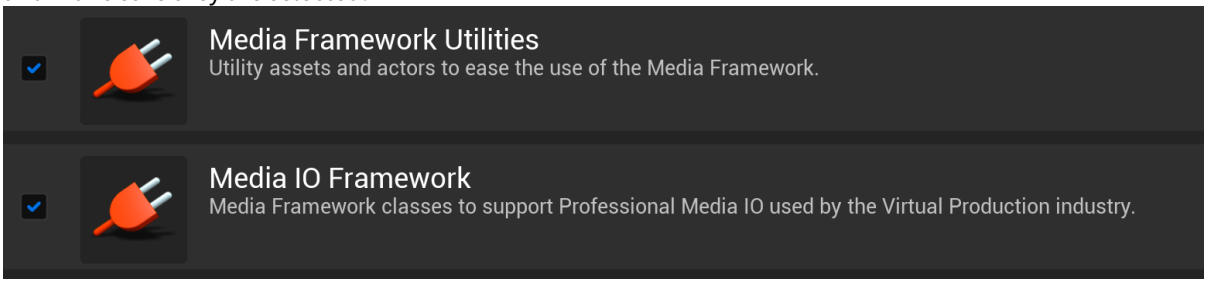
Copyright © 2026 Vizrt



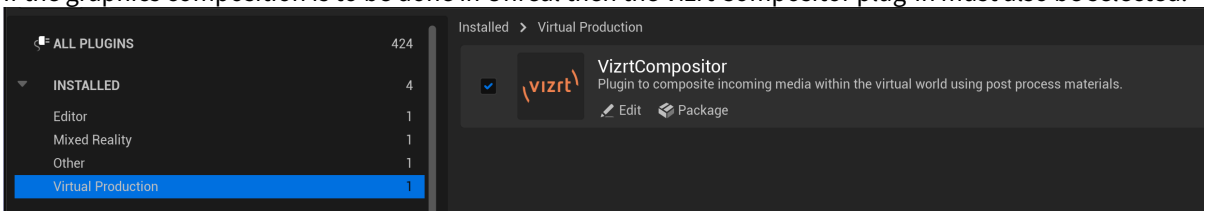
3. Click **Create**. Unreal Engine now restarts and loads your new Project.



4. Click on **Settings/Plugins** and enable **VizrtMedia.**
5. Search for **Media IO Framework** and **Media Framework Utilities** from the list of available plug-in sections and make sure they are selected.

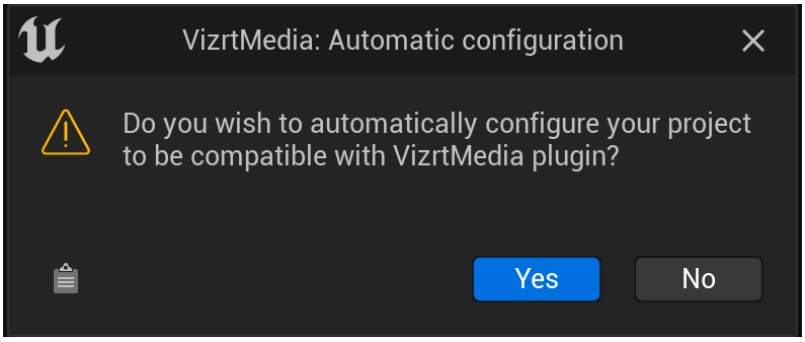


6. If the graphics composition is to be done in Unreal then the Vizrt Compositor plug-in must also be selected.



7. Click on **Restart Now** to apply those plug-ins to our project.
8. After restart press **Yes** to this dialog to auto configure the project for the VizrtMedia plug-in.

Copyright © 2026 Vizrt



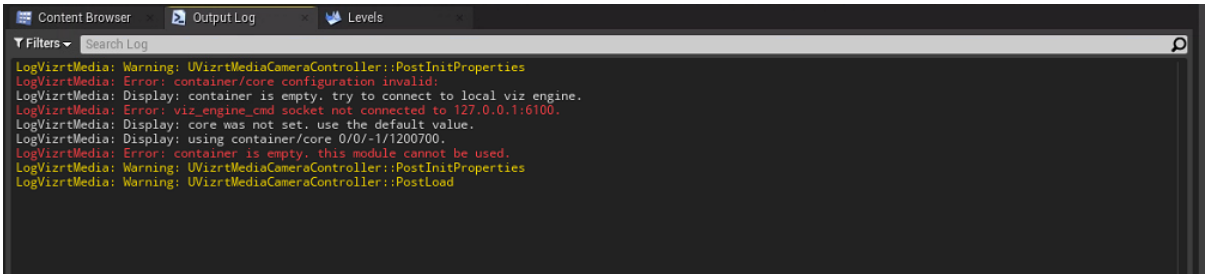
9. If Unreal shows a dialog requesting to switch projects, press **Yes**.

Information: It is highly recommended to use the project auto configuration, but if manual configuration is desired please follow the project settings manual configuration and syncing configuration.

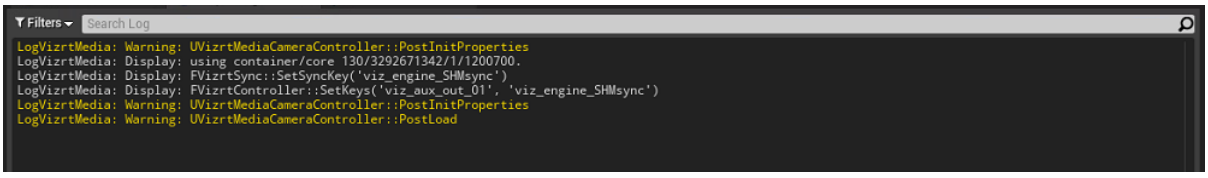
Licensing

Vizrt Media for Unreal Engine requires a valid license. Without any license the plug-in does not operate correctly and no valid timecode is available. To verify if a license is available and acquired by Unreal, simply check the Log entries and Filter for *VizrtMedia*.

Output without Correct License

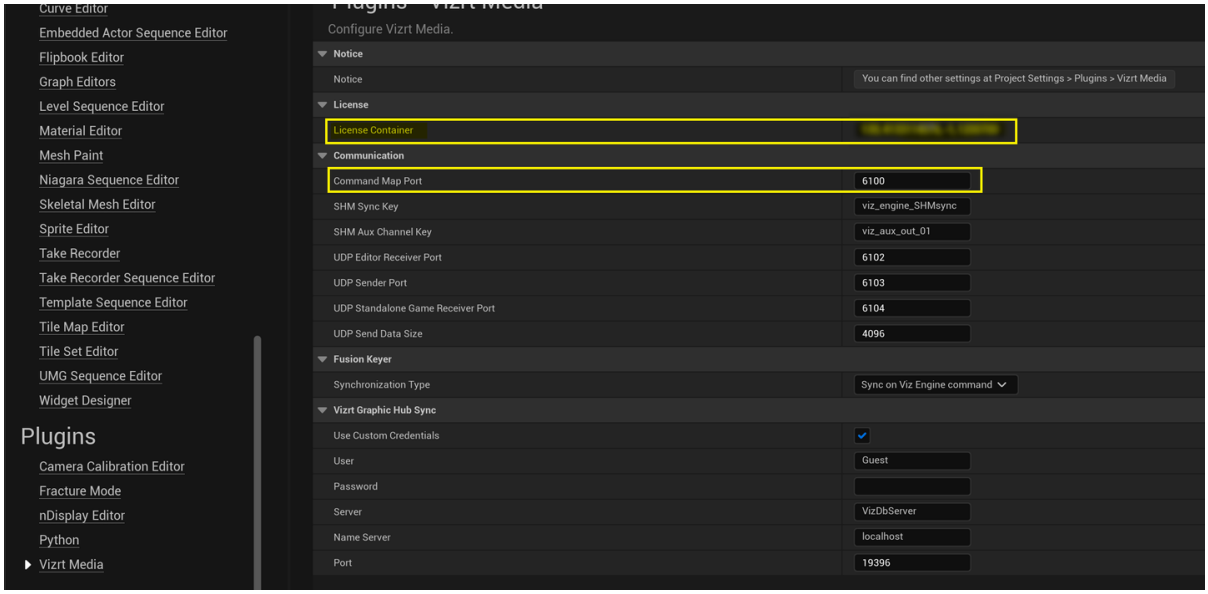


Output with Correct License



To setup a correct license, it is recommended to start Viz Engine and have the Vizrt Media plug-in query Viz Engine for its license configuration. Be sure the correct Command Map Port is set in the Vizrt Media Settings by clicking on **Edit > Editor Preferences** and navigating to **Plugins > VizrtMedia**:

Copyright © 2026 Vizrt



If Viz Engine finds a correct container with a valid *Vizrt Media for Unreal* license, this one is used and added in *License Container*. These settings are stored in an *.ini* file located in *%Appdata%\Local\UnrealEngine\5.0\Saved\Config\WindowsEditor\EditorSettings.ini.*

The setting in the *.ini* file is used. If there is no entry in the *.ini* file, Viz Engine is queried for a valid container. If the setting in the *.ini* file is incorrect, Vizrt Media for Unreal fails to load.

Information: If the Vizrt Media for Unreal license is located on a different location (server) rather than the Viz Engine licenses, you need to manually change the *.ini* file. The entry has the following format: *CONTAINER-MASK,CONTAINER-SERIAL,LICENSE-LOCATION,WIBU-CORELICENSE-ID*.

Additional Editor Settings

To improve UE performance while testing in Editor mode, navigate to **Edit > Editor Preferences > General > Performance** and deactivate *Use Less CPU When in Background*.

Note: For production/On Air use, UE has to run in Game mode and not Editor mode. Editor mode is intended for design and testing only.

To Establish Correct Syncing with Viz Engine

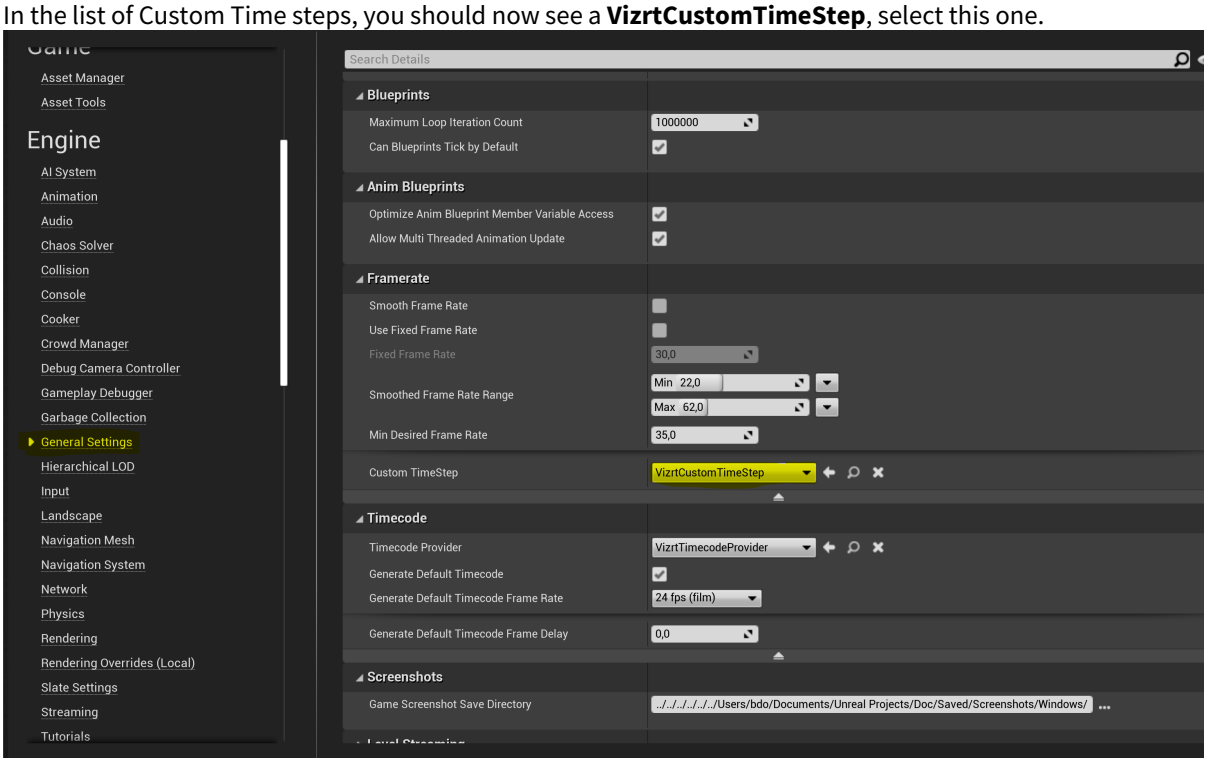
If you skipped the plug-in auto configuration follow the instructions below.
Make sure you have Virtual Production Utilities active:



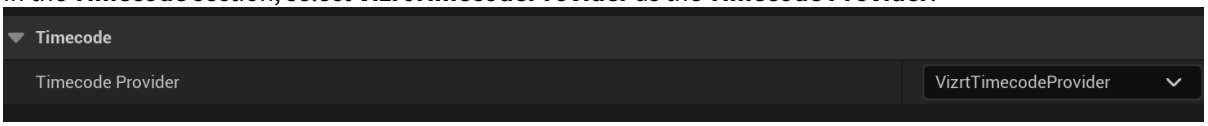
Copyright © 2026 Vizrt

A timecode provider is required to sync the Viz Engine with Unreal Engine, as both of them need to exchange data on a given timestamp.

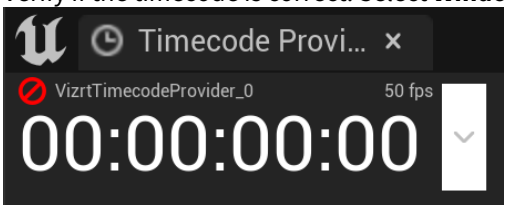
1. Select **Settings > Project Settings** in the list of settings select **Engine General Settings**.
2. Scroll down to **Framerate** settings and click on the **Show Advanced** arrow.
3. In the list of Custom Time steps, you should now see a **VizrtCustomTimeStep**, select this one.



4. In the **Timecode** section, select **VizrtTimecodeProvider** as the **Timecode Provider**.

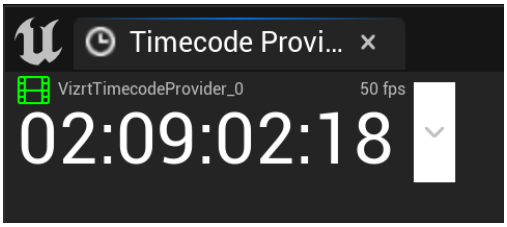


5. Press the **Apply** Button and restart Unreal Engine by clicking on the **Restart Now** button on the lower right.
6. Verify if the timecode is correct. Select **Window > Virtual Production > Timecode provider**:



   The timecode shows an invalid timecode. This is because Viz Engine is not running to provide a valid time code.
7. Start Viz Engine and after a few seconds, the timecode should show an increasing value, based on the uptime of Viz Engine.

Copyright © 2026 Vizrt



A correct timecode is absolutely mandatory.

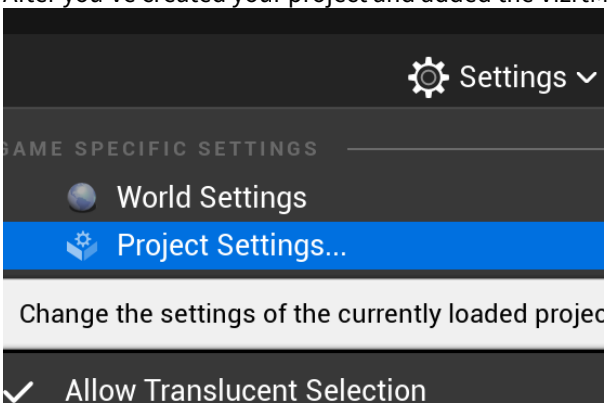
Tip: Unreal Engine can show the current framerate within its render window. Open the console and type `stat FPS` and press **ENTER**. The current frame rate is now shown in the Editor Window. The information shown should be green and shows a value of 50 FPS (depending on your settings).

To Use the Vizrt Media plug-in Capture

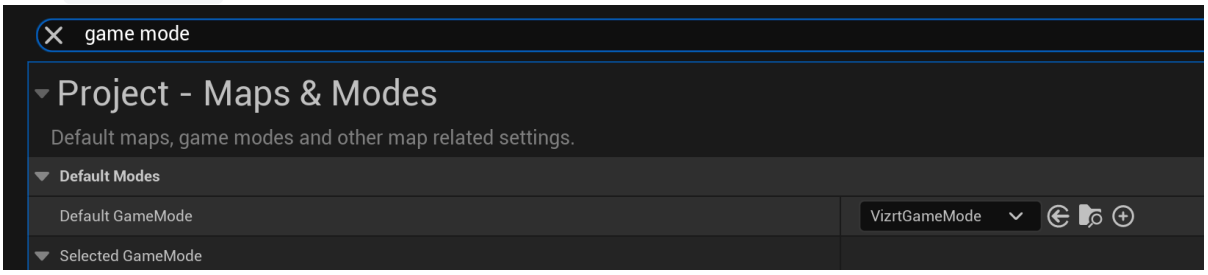
If you skipped the plug-in auto configuration follow the Manual configuration instructions below.

Manual Configuration

1. After you've created your project and added the VizrtMedia plug-in, go to **Settings > Project Settings**:

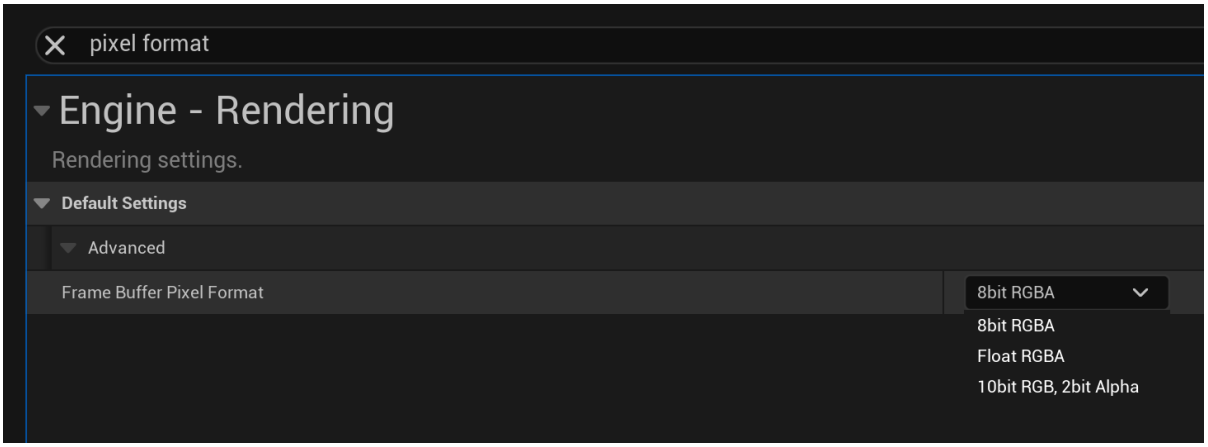


2. Type `Game Mode` in the upper search menu and set the Default GameMode to **VizrtGameMode**:

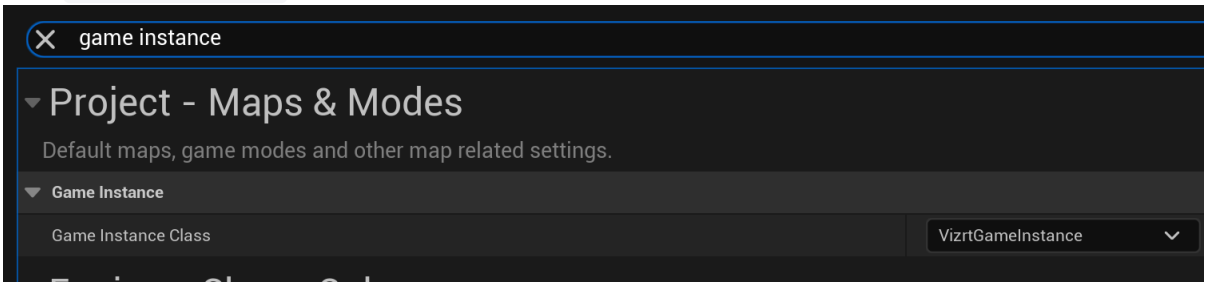


3. Type `Pixel Format` in the upper search menu and switch the Frame Buffer Pixel Format to **8bit RGBA** for SDR rendering or **Float RGBA** for HDR rendering:

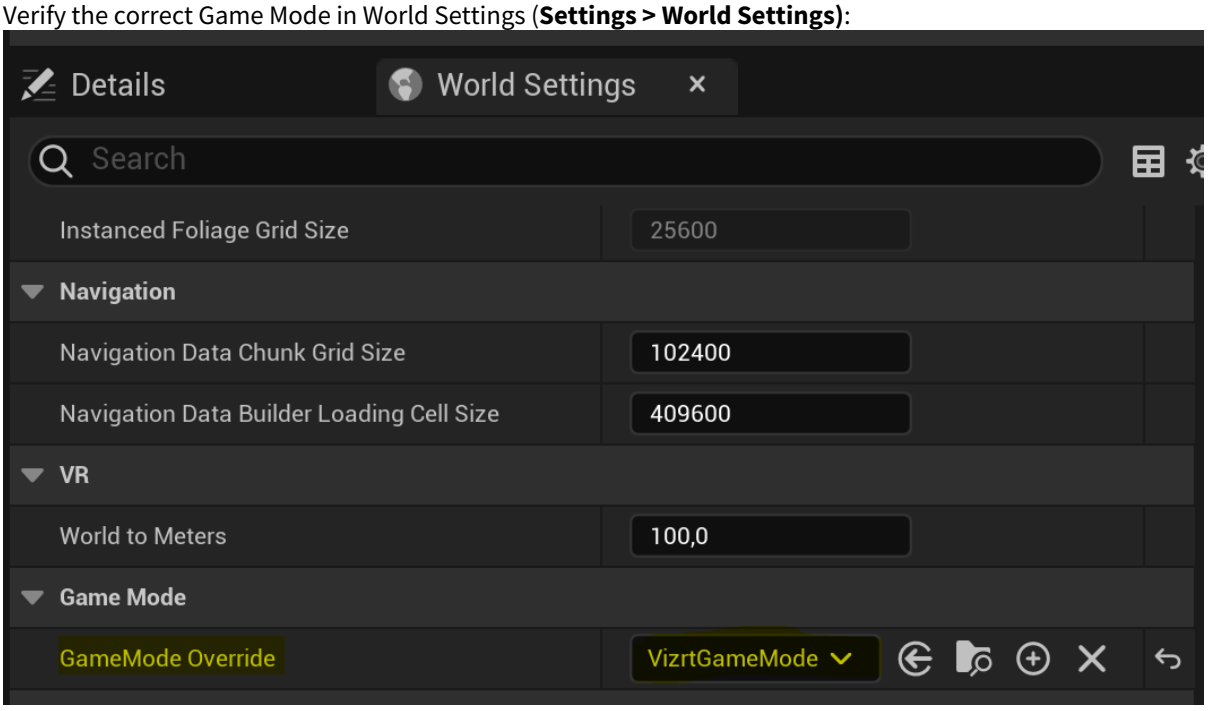
Copyright © 2026 Vizrt



4. Type `Game Instance` in the upper search menu and set **Game Instance Class** to VizrtGameInstance:

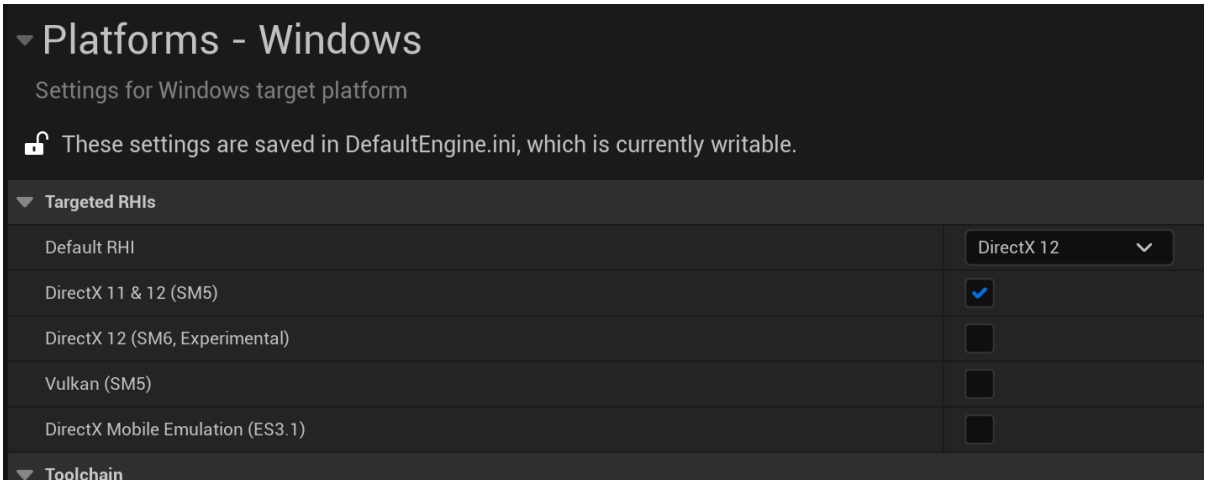


5. Verify the correct Game Mode in World Settings (**Settings > World Settings**):



6. Set the rendering platform to **DirectX 12** in **Platforms > Windows**:

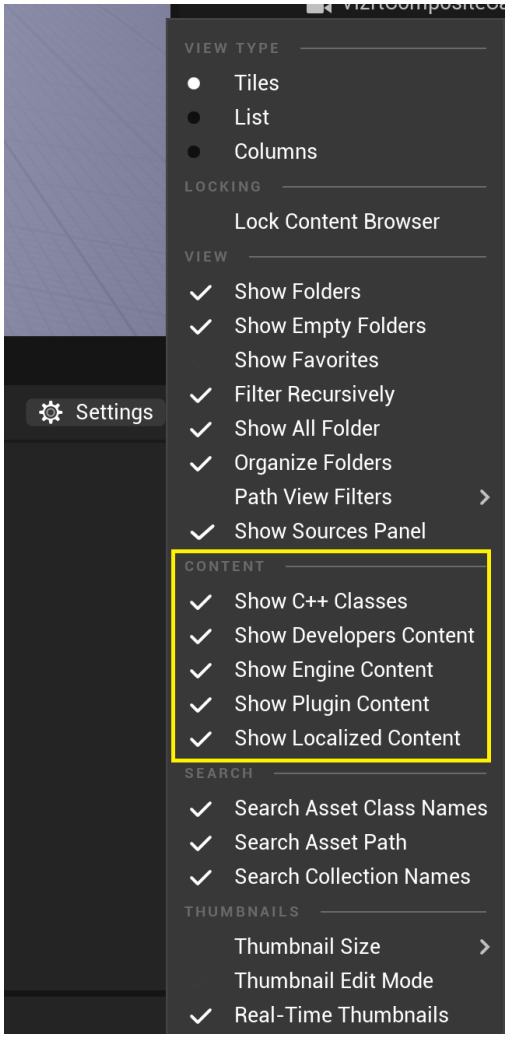
Copyright © 2026 Vizrt



VizrtAutoCapture

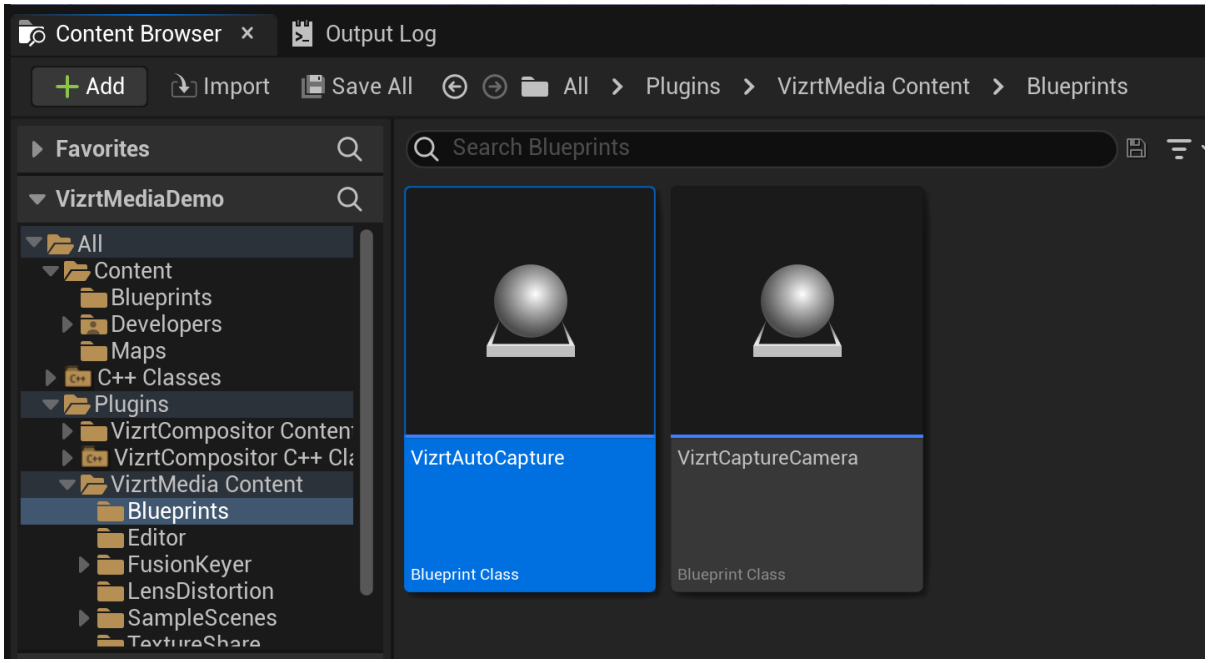
The next step is to set up the correct blueprint to capture the Unreal Engine rendering. Select **All Content** in the **View Options** of the Content Browser:

Copyright © 2026 Vizrt



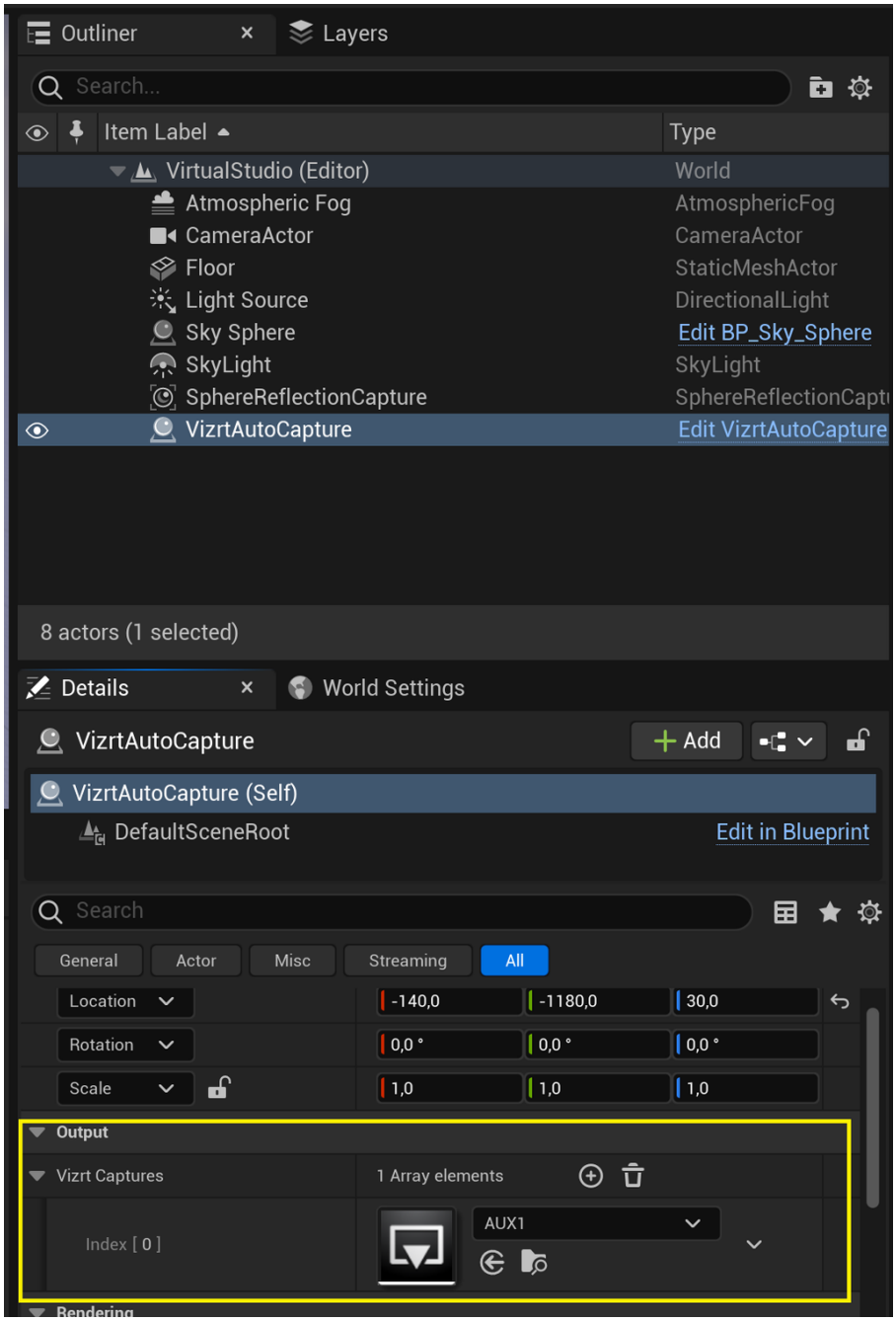
This allows the **VizrtAutoCapture** Blueprint to show up in **Plugins > VizrtMedia Content > Blueprints**. Select it and drag it into your World:

Copyright © 2026 Vizrt



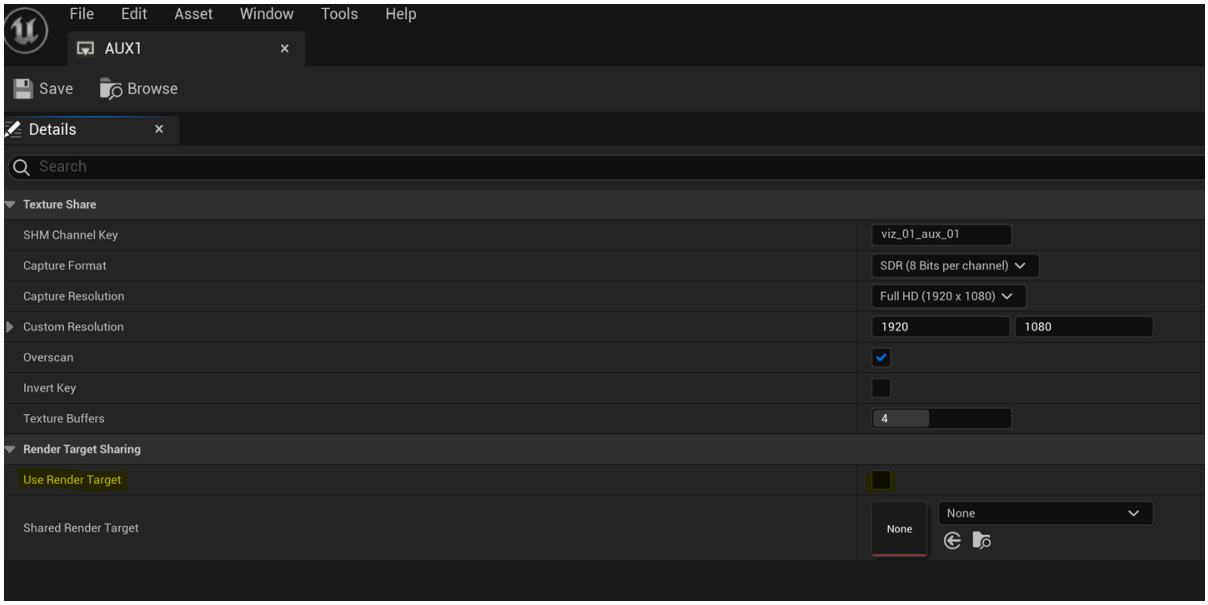
A capture must be added for the VizrtAutoCapture object. Click on **VizrtAutoCapture** in the World Outliner, then **Details** and scroll down to **Vizrt Captures**. Add a capture using one of the **AUX** assets and make sure the **AUX** number matches the **AuxChannel** used in Viz Engine.

Copyright © 2026 Vizrt

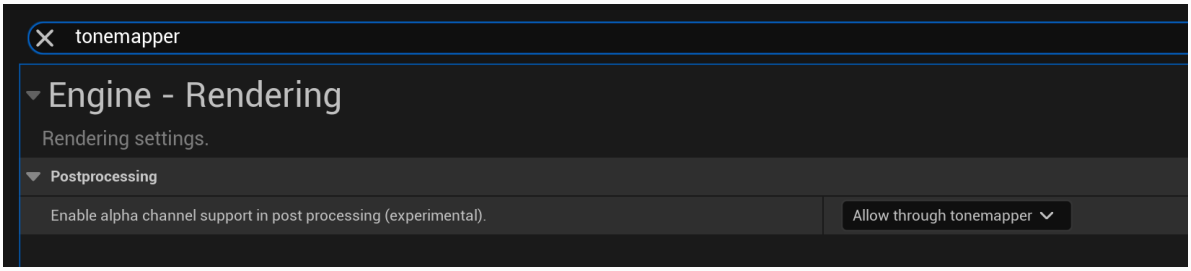


The **AUX** asset can be configured. Make sure its settings match the project requirements and that **Use Render Target** is unchecked.

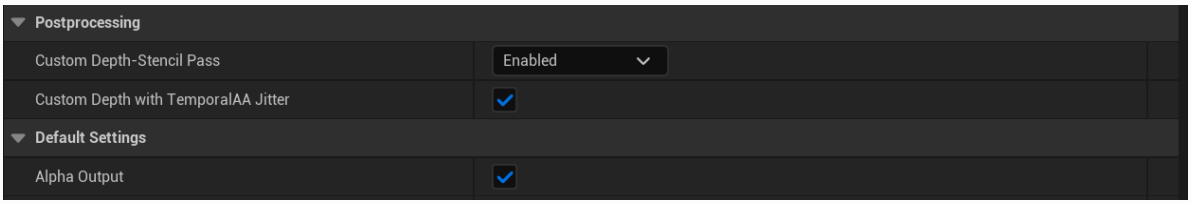
Copyright © 2026 Vizrt



Information: For key information to be sent to Viz Engine, set **Enable alpha channel support in post processing** to **Allow through tonemapper**, in Project Settings. This is only for Unreal Engine 5.5 and older.



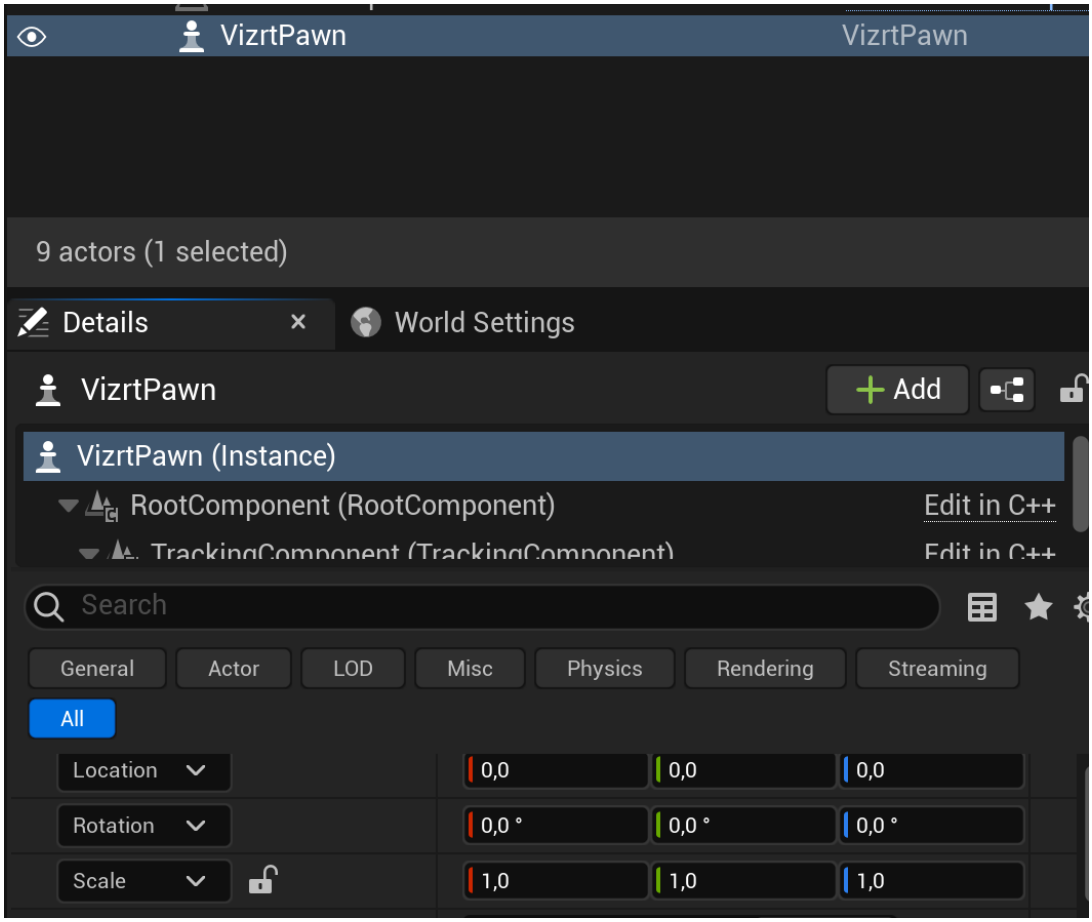
For Unreal Engine 5.6 and newer, this option is replaced by the **Alpha Output** on the Default Settings in Rendering.



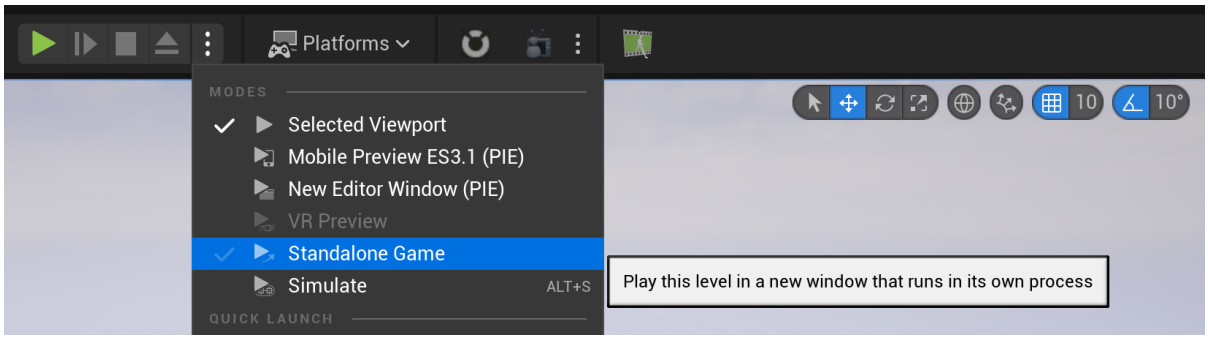
VizrtPawn

Finally you need to add the pawn controlling the camera. In the **Place Actors** tab, select **All Classes**. Find the **Vizrt Pawn** actor and drag it to the world and set *Location to (0, 0, 0)*, *Rotation to (0, 0, 0)* and *Scale to (1, 1, 1).*

Copyright © 2026 Vizrt

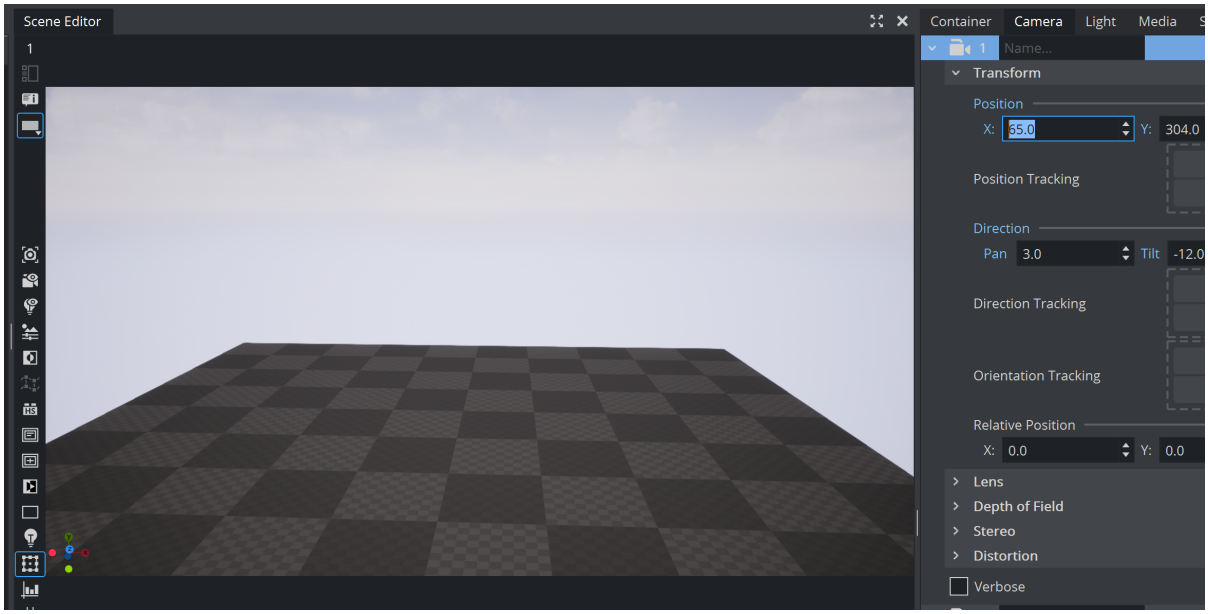


You can now test your setup by playing the game. To do so, select Play **Standalone Game**. This is the recommended mode for productions, but the other play modes can also be used while designing the scene.

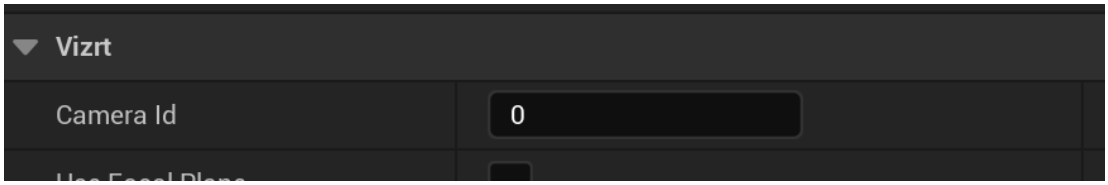


Viz Engine now shows output from Unreal as background (or how the scene is designed). To verify your correct setup, move the camera. The Unreal camera must move accordingly.

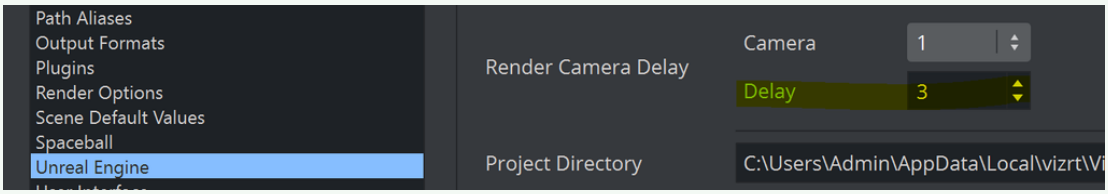
Copyright © 2026 Vizrt



Select which Viz Engine camera should be used in Unreal by setting the pawn **Camera Id** property.
A value of 0 uses the currently selected camera in Viz Engine. A value between 1 and 16 uses the corresponding Viz Engine camera.



Tip: To adjust the delay between Viz Camera and Unreal camera, please adjust the render camera delay in Viz Engine going to **Config > Unreal Engine.**



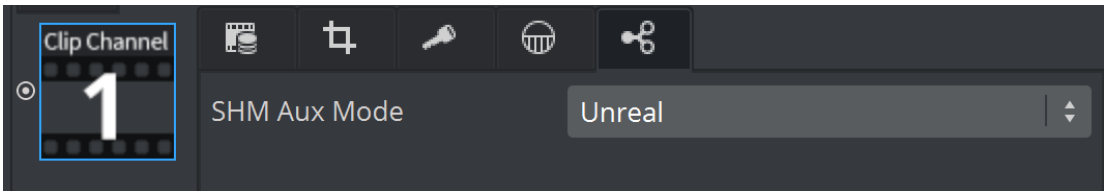
24.1.6 Adding Media Assets from Viz Engine in Unreal Scenes

If your design requires to have input sources from Viz Engine shared with Unreal Engine, this can also be achieved (for example, if a Live Feed needs to reflect on your Virtual Studio floor). It's possible to send to Unreal clip channels, graphics channels and live channels. In our example, we use a clip as texture to be shared with Unreal Engine.

1. Drag *Clip1* as texture to Media Assets in Viz Engine, load a clip and set it to **Autoplay**.
2. Now switch to the *SHM Aux Mode* in Media Asset properties.

Copyright © 2026 Vizrt

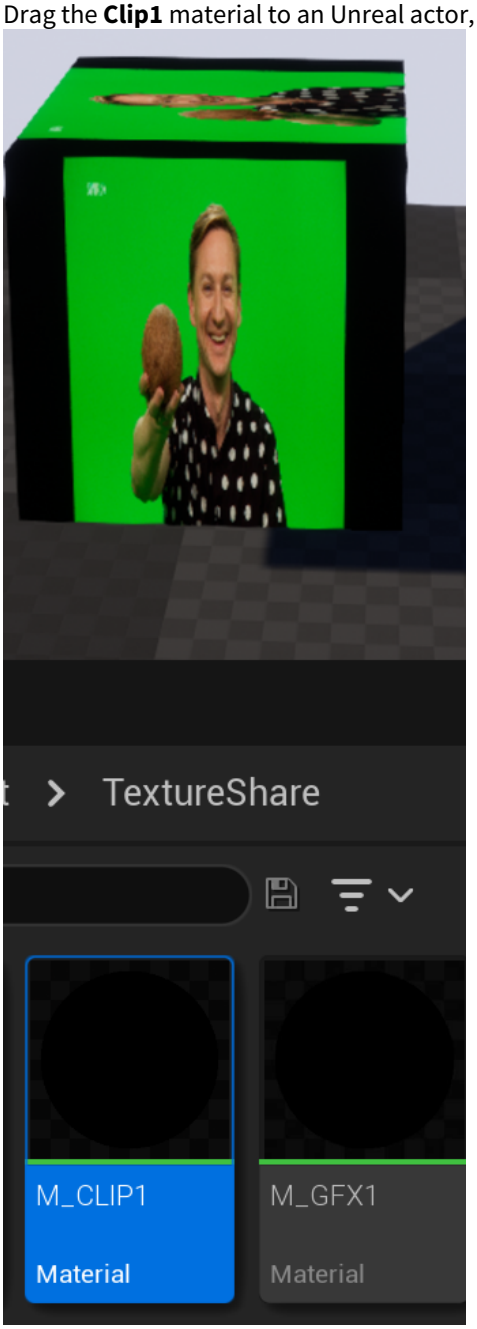
3. Select **Unreal** as *SHM Aux Mode.*



4. Switch to Unreal Engine and stop the current Playmode if necessary.

Now on the Unreal side, we need to drag the CLIP1 material to an object.

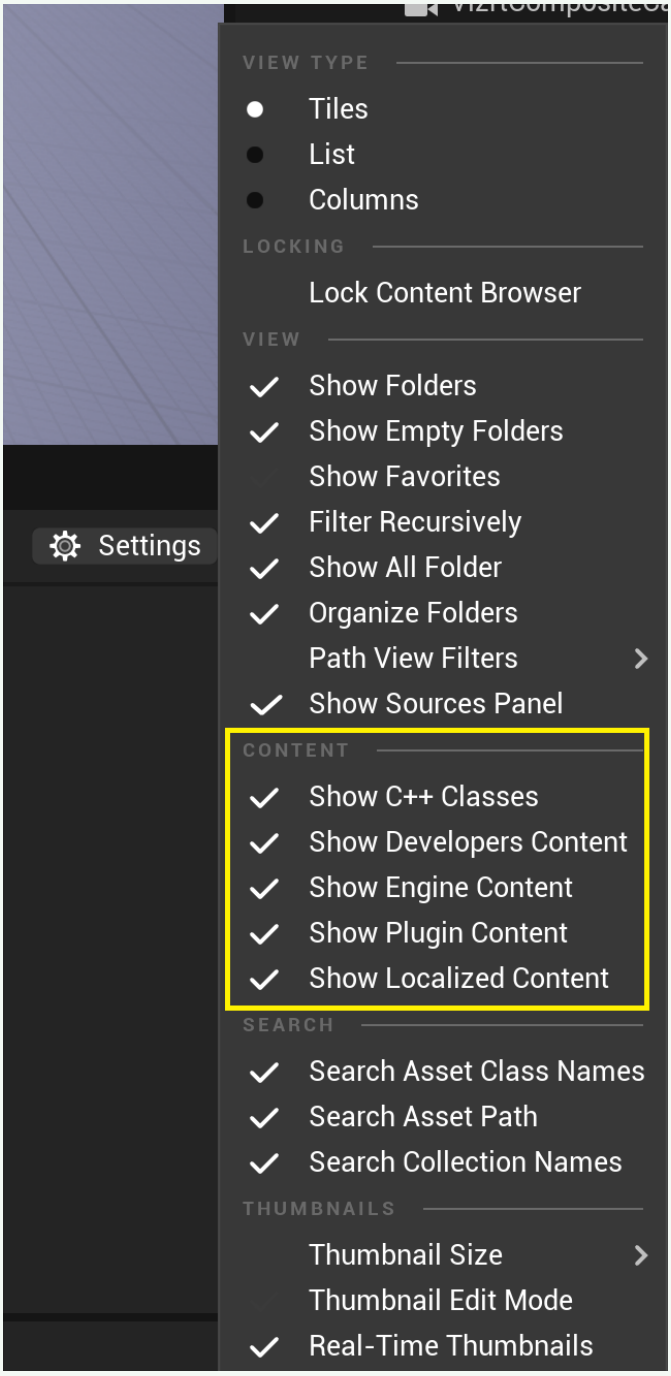
1. Switch to **Content Browser.**
2. Drag the **Clip1** material to an Unreal actor, for instance a cube.



Copyright © 2026 Vizrt

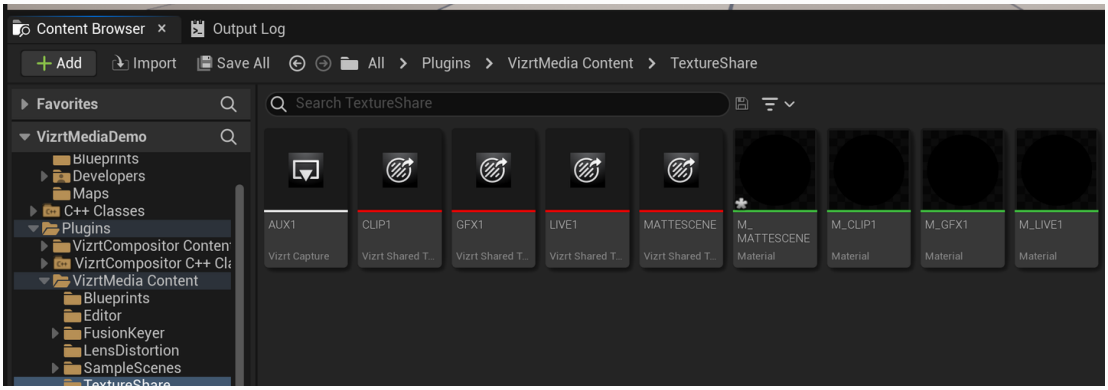
3. Play the game in Unreal.

Tip: If you search for the material but it does not show up, make sure you have both **Show Engine Content** and **Show Plugin Content** enabled.



Copyright © 2026 Vizrt

Information: All the texture sharing assets are located in **Plugins > VizrtMedia Content > Texture Share** in the **Content Browser**.



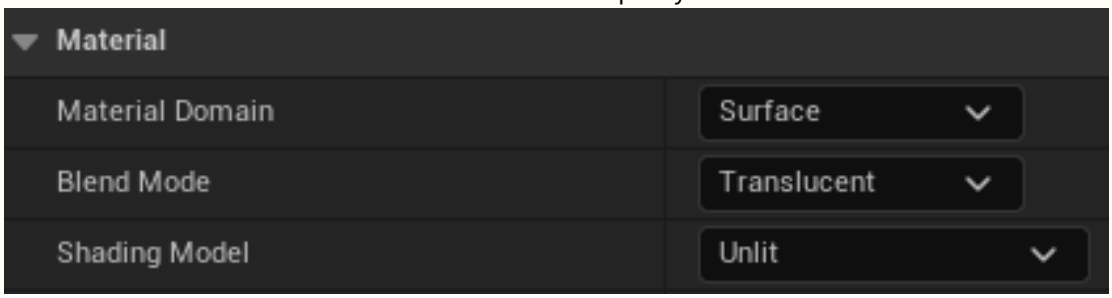
- **Vizrt Capture** assets capture Render Targets or the Viewport and send them to Viz Engine.
- **Vizrt Shared Texture** assets receive clips, GFX, lives and the matte scene from Viz Engine.

The remaining texture share content materials can be applied to an Unreal object to show the corresponding textures.
The **Vizrt Shared Texture** can be configured by double clicking the asset. Check if the SHM channel key clip number matches the clip number used in Viz Engine.



Note: When using Temporal Anti Aliasing (TAA) or Temporal Super Resolution (TSR), some options have to be enabled for live/clip materials to avoid a ghosting/smearing effect.

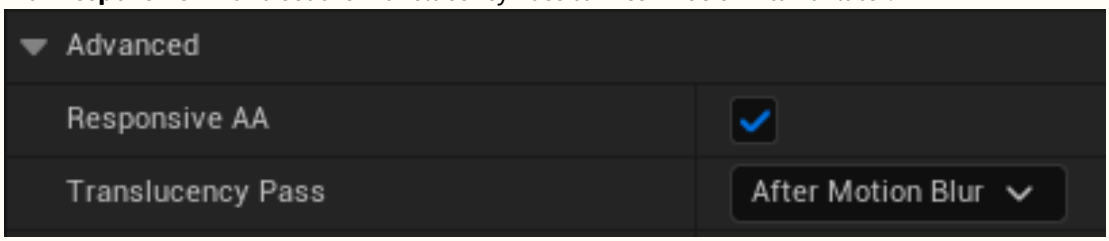
- The material needs to be set as **Translucent** and with opacity = 1.0.



Copyright © 2026 Vizrt

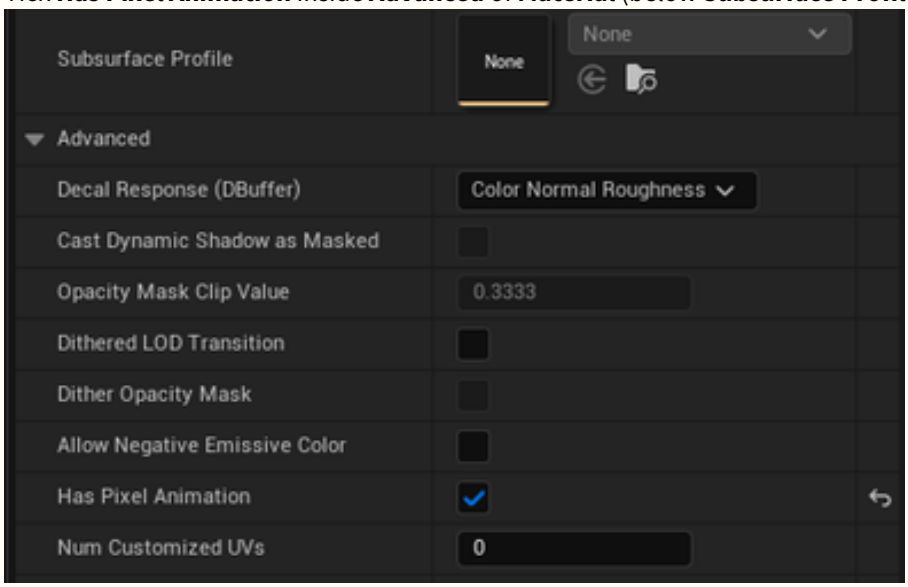


- Tick **Responsive AA** and set the Translucency Pass to **After Motion Blur** or later.



On **Unreal Engine version 5.4 and newer**, there is a new *Has Pixel Animation* flag that can be set on Surface materials. That should be activated when using TAA or TSR as it stabilizes the image on the material. This can also be used with Blend Mode - Opaque materials which was not possible before.

- Tick **Has Pixel Animation** inside **Advanced** of **Material** (below **Subsurface Profile**).



Note: The texture share assets should be seen as a template. Please do not modify them. If one of these assets need to be changed, please create a copy.

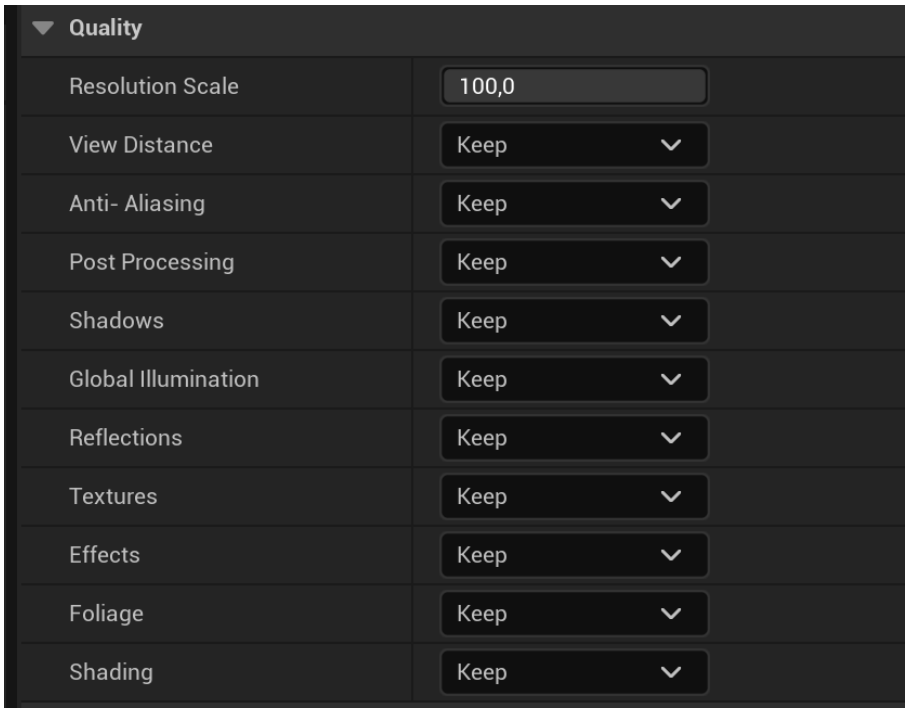
24.1.7 Quality Settings (Unreal Engine Scalability Settings)

The Quality settings allows the quality adjustment of various features in order to maintain the best performance. Here you can change the most common settings that impact the rendering quality and performance of the Unreal

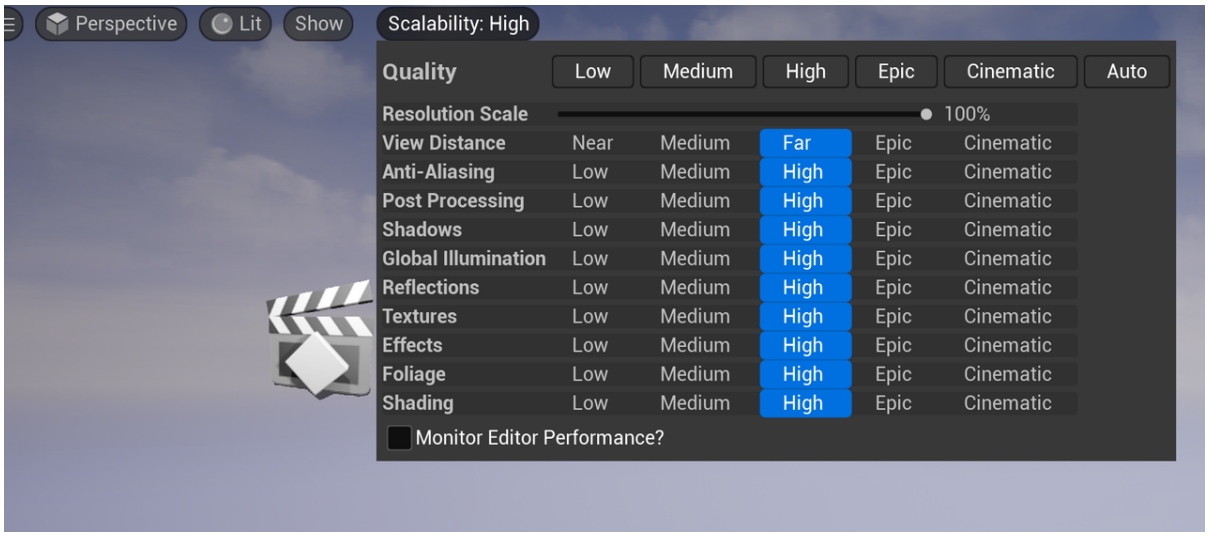
Copyright © 2026 Vizrt

project.
Use these settings to make sure you have the same Quality settings both in Editor and Standalone game.

To access and change these settings select the **VizrtAutoCapture** actor, go to the properties panel and scroll to **Quality**. After changing the settings press **Play** to visualize their impact.



It is possible to check what quality settings are being applied on the **Scalability** widget, inside the render window:

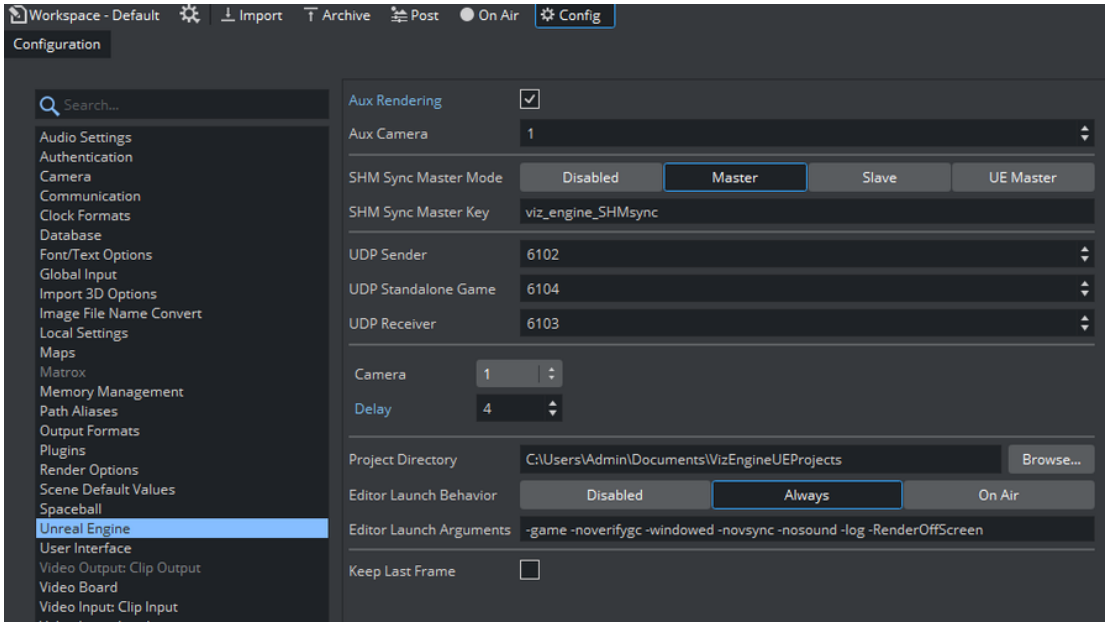


Copyright © 2026 Vizrt

Information: To get in depth information about the impact of these settings in Unreal, please check its documentation about Scalabilty Settings.

24.1.8 Keep Old Frame vs Black Frame

In case of a sudden loss of connection between Viz Engine and Unreal Engine (a crash on Unreal, as an example), it's possible to choose to keep the older frame or to have a black frame while connection is down. That option is in **Configuration > Unreal > Keep Last Frame**.



24.1.9 Keying and Compositing in Unreal

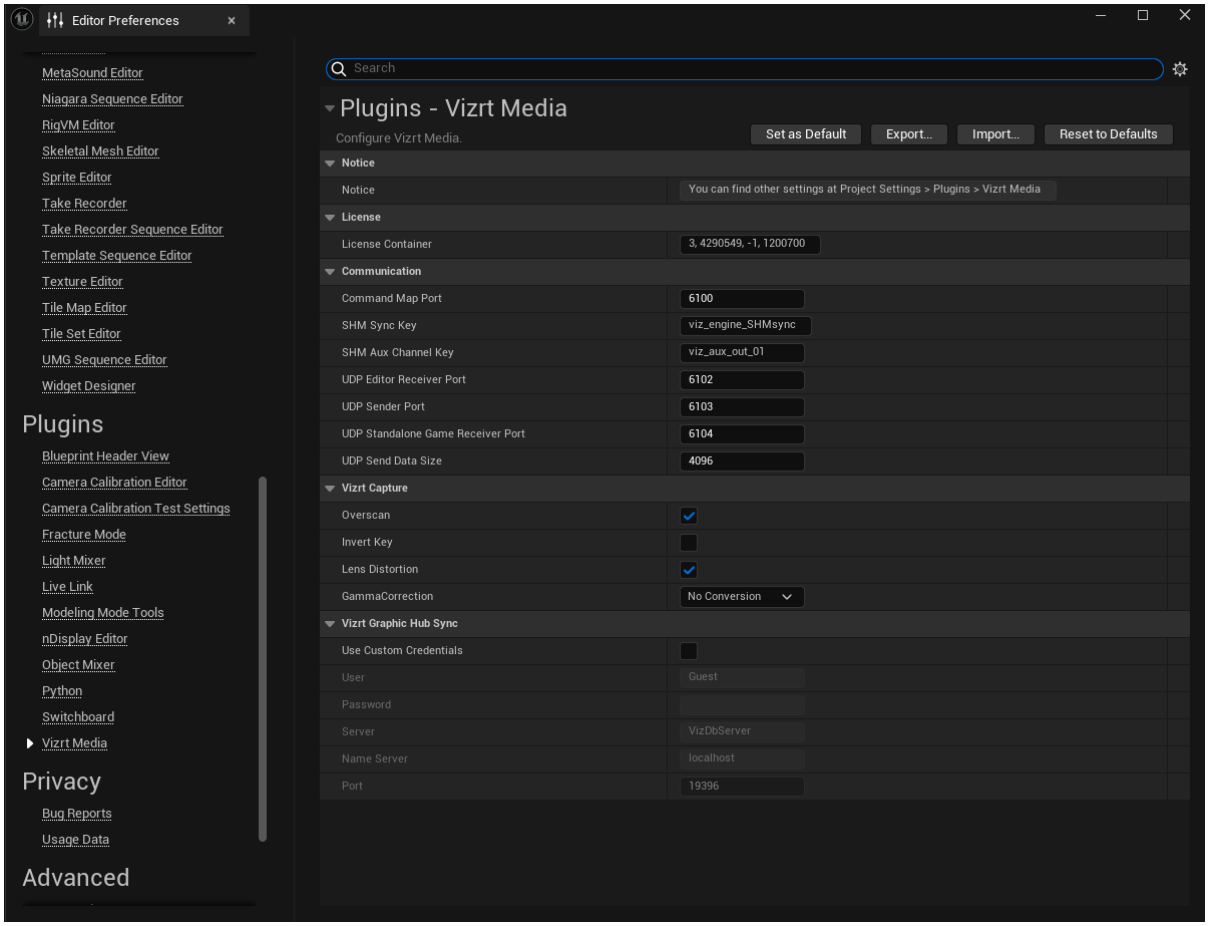
All the previous information assumes that the graphics keying and compositing is done in Viz Engine. Starting from version 5.0 of the Vizrt Media plug-in, chroma keying and compositing can be done in Unreal Engine.

- For a guide on how to setup keying using the implementation of the Precision Keyer for Unreal refer to Precision Keyer Compositing on UE5.
- For a guide on how to setup compositing with reflections and shadows using the Vizrt Compositor plug-in please refer to Compositing in UE5.

24.1.10 Gamma Correction

To allow for different conversions (to support HDR as an example), a Conversion selection menu as been added to the Unreal Engine's Editor Preferences (**Editor Preferences > Vizrt Media > Vizrt Capture > GammaCorrection**):

Copyright © 2026 Vizrt

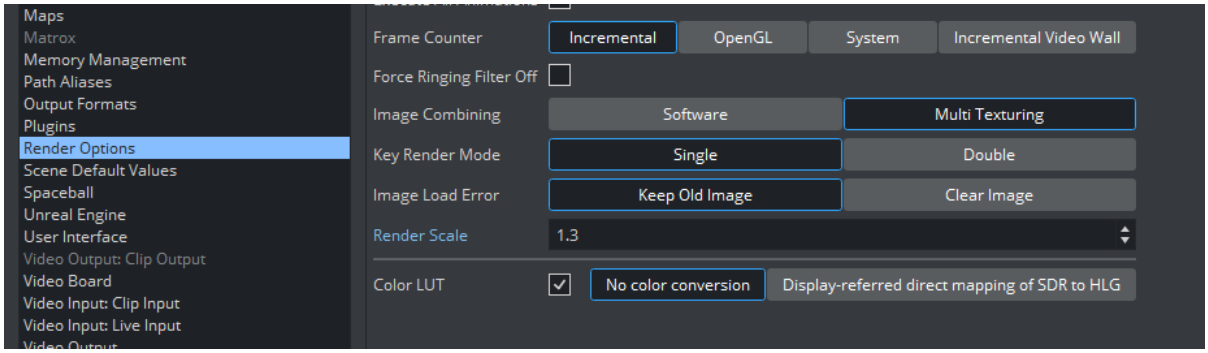


The Unreal Tonemapper can have some effect on the final colors so to properly view the colors without the tonemapping affecting them, a console command can be used (**ShowFlags.Tonemapper 0**) or these commands can be added to the *DefaultEngine.ini* file in the project folder */Config*:

r.TonemapperGamma = 0
r.TonemapperFilm = 0
r.Tonemapper.Quality = 0
r.ToneCurveAmount = 0
r.Mobile.TonemapperFilm = 0
r.MobileTonemapperUpscale = 0
r.EyeAdaptationQuality = 0
r.EyeAdaptation.ExponentialTransitionDistance = 0

On the Viz Engine side, LUT should be activated:

Copyright © 2026 Vizrt



No color conversion should be selected if Unreal has "No conversion" as well.

24.1.11 Syncing Unreal Project with Graphic Hub

Starting from version 5.0 of Viz Engine and the VizrtMedia plug-in, it's possible to sync the Unreal project with Graphic Hub using a file collection, and link that file collection to a Viz scene. Please refer to Working with File Collections for more information.

24.1.12 Using Viz Engine Control Objects

Starting from version 5.0 of Viz Engine and the VizrtMedia plug-in, Viz Engine can update actor parameters in Unreal. Please refer to Control Objects on Unreal Engine.

24.1.13 Legacy Versions

The following shows an overview of older supported versions:

Viz Engine Version	Unreal Engine 5	Unreal Engine 4	Vizrt Media for Unreal Engine	Legacy NcamAR Plug-in
4.4 (Classic & Viz Engine Render Pipelines)		4.27.*, 4.26.*	✓	-
4.3 (Classic & Viz Engine Render Pipelines)		4.26.*, 4.25.*	✓	-
4.2 (Classic & Viz Engine Render Pipelines)		4.25.*, 4.24.*	✓	-
4.1 (Classic & Viz Engine Render Pipelines)		4.24.*, 4.23.*	✓	-
4.0 (Classic Render Pipeline)		4.20.*, 4.21.*	-	3.2.1

Copyright © 2026 Vizrt

Viz Engine Version	Unreal Engine 5	Unreal Engine 4	Vizrt Media for Unreal Engine	Legacy NcamAR Plug-in
3.14		4.20.*, 4.21.*	-	3.2.1
3.13		4.20.*, 4.21.*	-	3.1.2
3.12.0, 3.12.1		4.20.*, 4.21.*	-	3.1.0
3.11.1		4.20.*	-	3.0.0
3.11.0		4.16.1, 4.16.2, 4.16.3	-	2.0.0

24.1.14 Troubleshooting

In case of any trouble, please crosscheck the following:

- Do you have a valid license?
- Is the Engine configuration correct?
- Does the output size match your resolution Viz Engine?
- Are the keys for shared memory set correctly?
- Does Unreal receive a correct timecode if the Engine is running?
- Do you see a preview if you start the capture in Unreal (when using Capture Device)?
- Is Unreal in Play mode?
- Are the Viz assets added to the Unreal scene (GFX, CLIP, LIVE) also added to the Viz Engine scene?

In case the key signal is not correct, you can manually invert the signal in the settings for the used **AUX** asset. Unreal has several profiling tools built in to identify performance bottlenecks. Please refer to the Unreal documentation about details.

General

Keep in mind that both Viz Engine and Unreal Engine require their resources and capacities to deliver a stable and correct output. Make sure both applications have enough resources available. For Example, if the Unreal Scene already occupies 70% of the GPU power, Viz Engine is not able to perform its operations which results in jittering. The whole setup needs to be profiled and optimized on the final hardware with the final setup. If the Unreal project performs correctly on the designers desk, there is no guarantee if the whole project will then run correctly if everything is combined together. This involves a Matrox videoboard, the Unreal running in Game Mode, Viz Engine with its compositing scene and correct Chroma Keyer setup and Tracking Data coming in.

When profiling the unreal scene, check the following:

Copyright © 2026 Vizrt

- Use Lights only where needed. If lights are static, check if they can be baked. Spot Lights are very compute intense, try to minimize as much as possible.
- Unreal Engine needs to run at at least 60-70fps in real time.
- Try to keep the Light map size low, especially when using dynamic lights.
- Try to optimize the Texture sizes. The GPU memory is limited and Viz Engine also needs to allocate memory within this space. If Texture Memory is getting low, the GPU driver needs to free up or swap and this results in jittering.
- Scene Capture 2D and Cube capture requires performance.,
- Make sure the Unreal Engine is running at least in Game mode and the Editor is closed.
- Make sure that Unreal's Garbage Collector does not become active during production. Try the settings below:



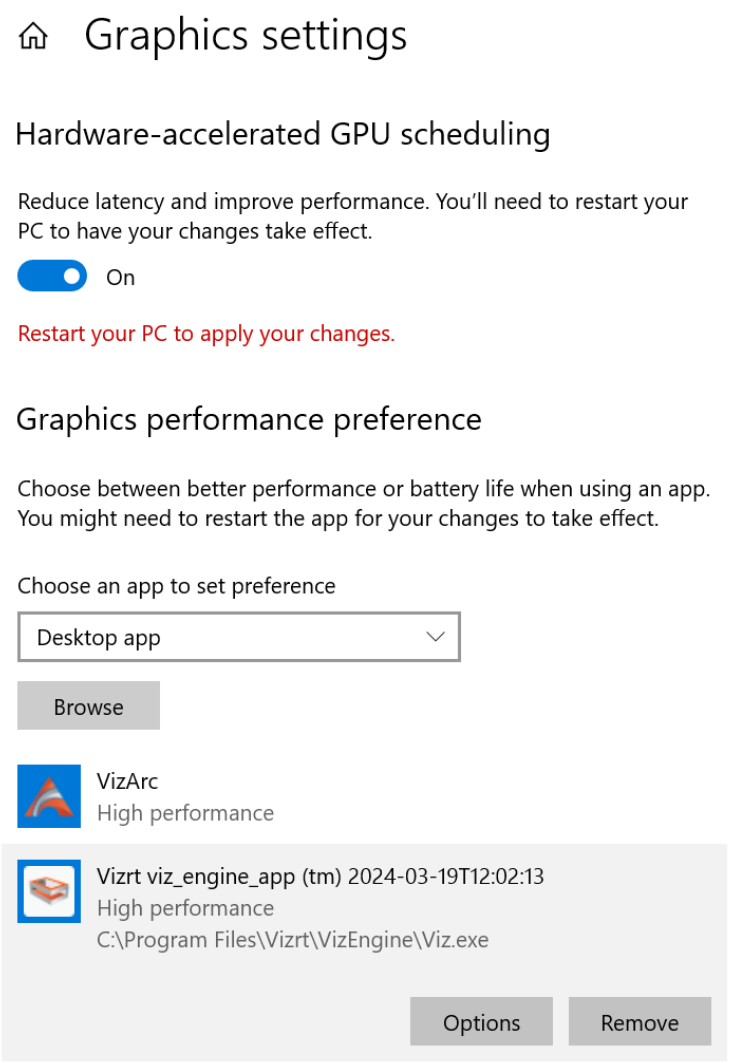
An overview of all available Profiling Tools can be found in the Unreal Engine Documentation.

Hardware-accelerated GPU Scheduling

One possible performance improvement in Windows 10 21H2 settings is in the Windows Graphics Settings.

Go to **Display Settings** > **Graphics** to open this window:

Copyright © 2026 Vizrt



This option can improve performance in some scenarios when using **UnrealEngine** + **VizEngine**. On Viz Engine-only scenarios, performance can degrade, so this should always be tested per project.

There are two options for these settings:

- Only activating Hardware-accelerated GPU scheduling.
- Activating **Hardware-accelerated GPU scheduling** and adding Viz Engine and Unreal Engine to the **Graphics performance preference** section, and selecting the **High-performance** GPU (browsing and selecting both executable engines, adding them to the list).

See Also

- Unreal Engine Documentation
- Viz Arc Documentation
- Working with File Collections
- Control Objects on Unreal Engine
- Shared Memory Map
- Compositing in UE5

Copyright © 2026 Vizrt

- Precision Keyer Compositing on UE5
- Mixing Depth Buffers

Copyright © 2026 Vizrt

24.1.15 Shared Memory Map

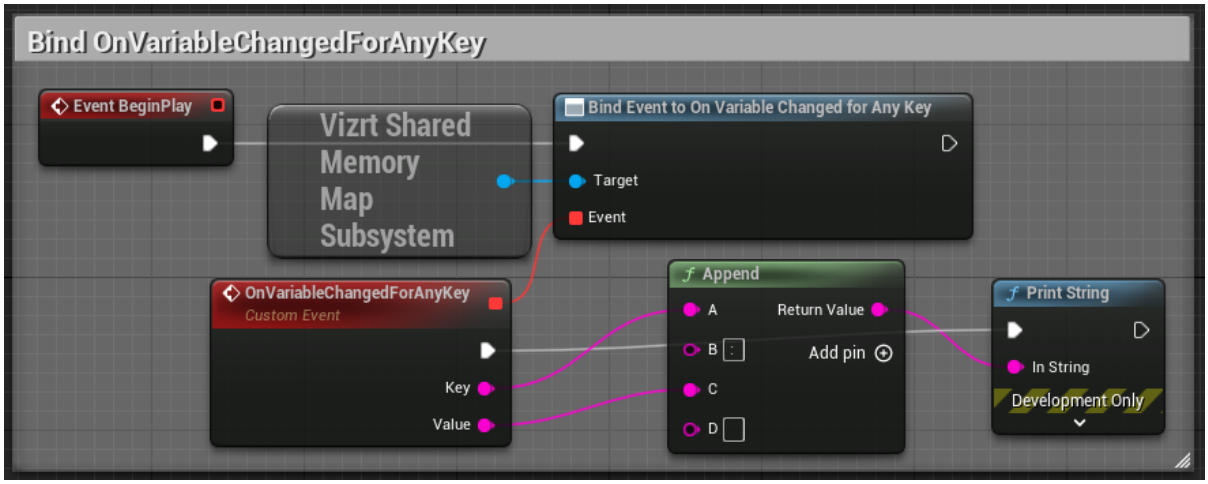
This feature allows Unreal Engine to be notified whenever the **Global** Shared Memory Map is updated on Viz Engine. To enable this feature, you need to install **Vizrt Viz Extensions Plugins**.



1. In your **Unreal Engine** project, you need to setup a Blueprint script, either as a Level Blueprint or as a Blueprint Actor.

Note: All the received *key:value* pairs are logged into the screen. Therefore, you can use a new *key:value* pair to modify a specific property of a specific Actor you control.

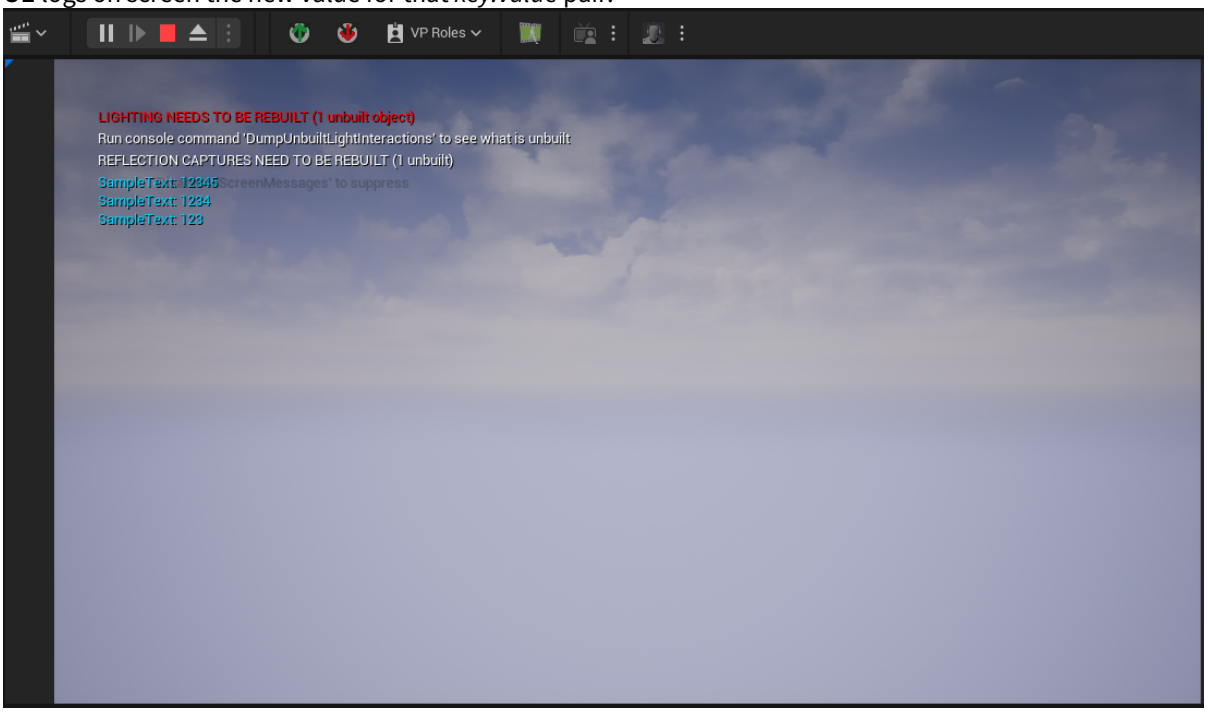
Copyright © 2026 Vizrt



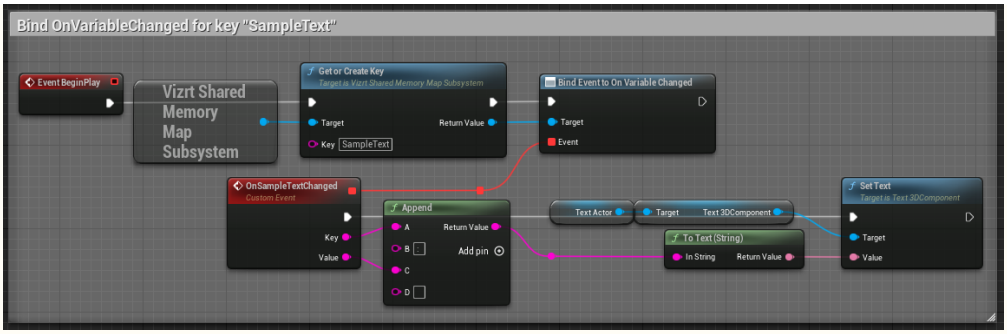
2. In **Viz Engine**, update any parameter through the **Global** Shared Memory Map:

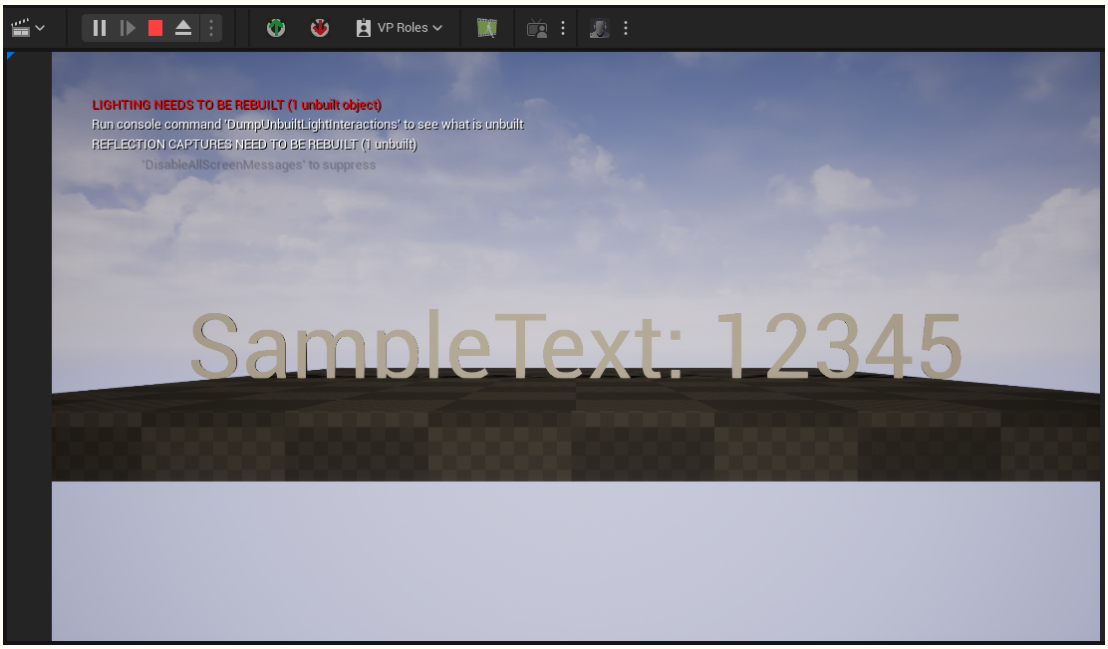
```
send GLOBAL*MAP SET_VALUE SampleText 12345
```

3. **UE** logs on screen the new value for that *key:value* pair.



Note: Instead of listening for all Key:Value updates, you can listen for updates of a specific Key only. The example below shows how this can be done, and use the received Value to set the Text property of an Actor.





Copyright © 2026 Vizrt

24.1.16 Use Cases for Unreal Integration

The following lists typical use cases and describes the correct setup of them. All of these use cases refer to the Viz Engine Render Pipeline.

- Unreal Engine as Backdrop in Virtual Studio
- Unreal Engine as AR Object
- Virtual Camera Flights
- Talent Reflection
- Unreal Engine as the Compositor
- Acknowledgements

Note: Be sure that Postprocessing Preview is turned on when using the Design mode in Viz Artist.

Unreal Engine as Backdrop in Virtual Studio

Unreal Engine is used to render the backdrop within a Virtual Set. Viz Engine is used to compose the backdrop and the keyed camera signal. The camera is tracked by Tracking Hub and the internal Precision Keyer is used.

Render Sequence	Virtual Studio
Background Asset	Backdrop rendered in Unreal 

Copyright © 2026 Vizrt

Foreground Asset	Live tracked Camera Signal 
Scene Graphics	Not necessarily required. Can be added either as Overlay (requires Key plug-in) or behind the anchor. Scene must have a matte scene definitions. Can support Talent reflection.
Notes	Camera must be set to Remote. Live in must be chroma-keyed. Lens deformation and Render scale must be set.

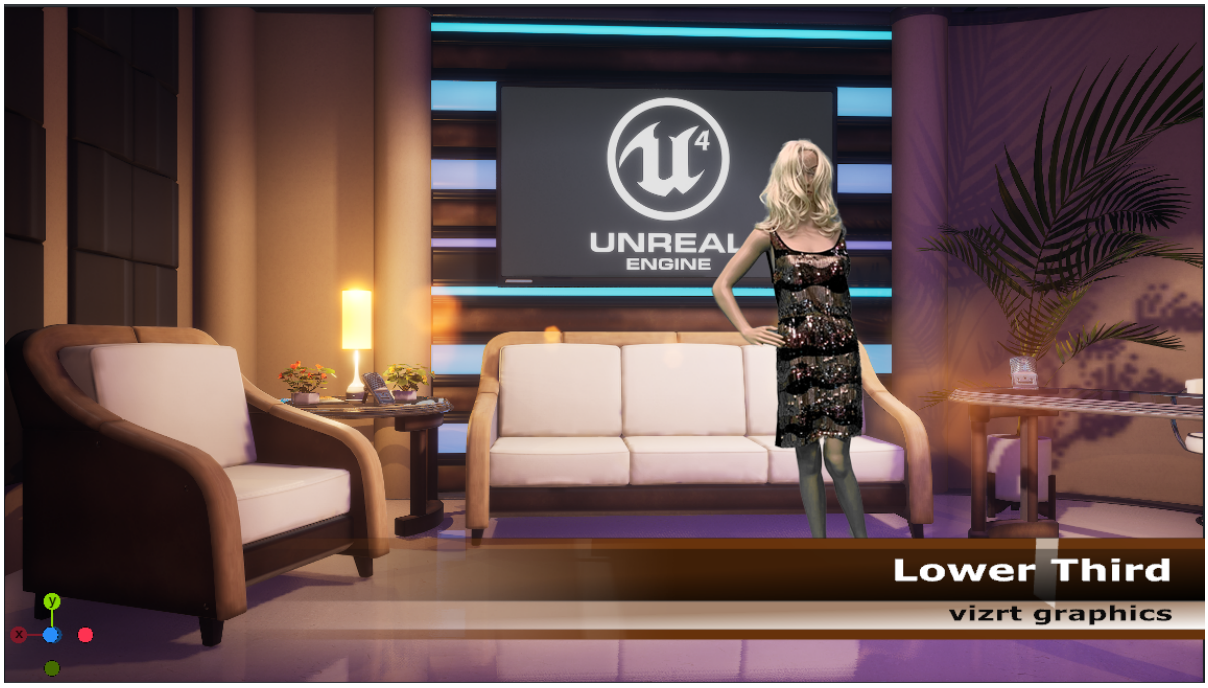
Copyright © 2026 Vizrt

Result	The final result (no Viz Graphics). 

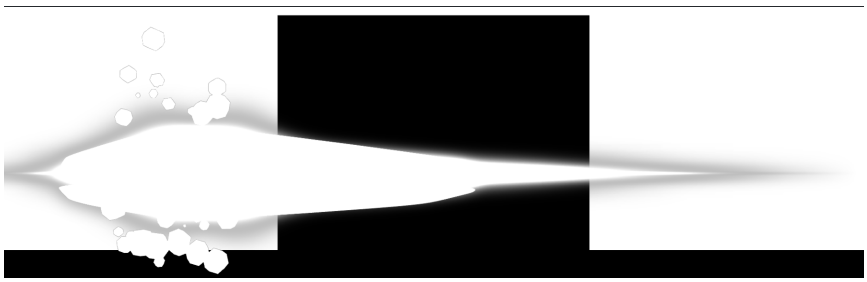
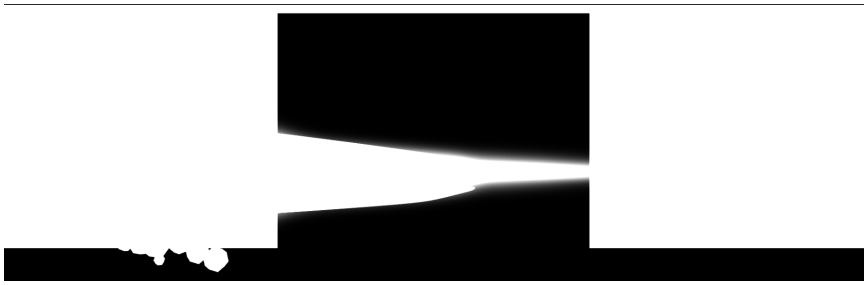
To add an additional layer as a front scene, such as a lower third, you need to assign a GFX channel as Classic Scene and as a full frame DVE effect in front of the foreground asset:



Copyright © 2026 Vizrt



It is important that this DVE GFX channel is in front of the FOREGROUND layer.
The GFX Scene needs to have a correct key applied. When used in combination with a Matte scene (e.g. for external Keying) you might not want the foreground key to be blended with the matte key signal, it should be added to the existing key signal.

Key Signal blended with additional GFX Frontlayer (INCORRECT)	
Key Signal added with additional GFX Frontlayer (CORRECT)	

To change the way how the of DVE layers are blended, you need to manually change the blending mode to ADD. This is currently implemented as command only and will be accessible via GUI in a later version. Type the following on your console window:

Copyright © 2026 Vizrt

```
send MAIN_SCENE*GLOBALS*DVE*KEY_COMBINE_MODE SET ADD
```

This renders the key correctly.

> **Note:** These settings are only available if a Scene has been defined as DVE layer.

Unreal Engine as AR Object

1. Viz Engine is used to compose an AR object rendered by Unreal Engine into a live signal. The Camera is tracked by Viz Tracking Hub. Ensure that Unreal creates valid key information by enabling **Enable alpha channel support** in your **Unreal Project Settings > Rendering**. It should be set to **Allow through tonemapper**.
    a. Verify the key input on Viz Engine.

Render Sequence	External AR
Background Asset	Live tracked Camera Signal 

Foreground Asset	AR graphics rendered in Unreal  Sample Key Signal from Unreal in Viz Engine 
Scene Graphics	Not necessarily required.
Notes	Camera must be set to Remote. Lens deformation and Render scale need to be set. Effects like Glowing, Bloom, Reflections on the Unreal Object require additional alpha information and are **NOT** supported. Casting shadows also requires an "invisible" Floor matching your real floor in Unreal and a material that casts shadows only and renders alpha information.

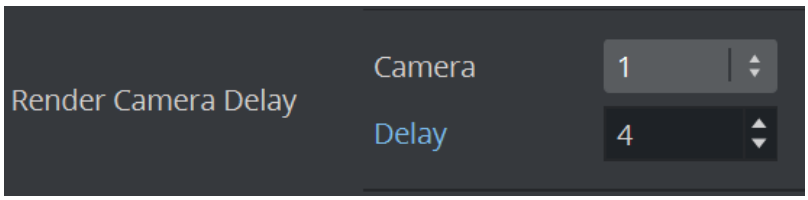
Copyright © 2026 Vizrt

Result	

Virtual Camera Flights

To use a Real camera and transition to a position in your Virtual Set and start an animation, the camera delay can be adjusted.

The number of **fields** the camera data gets delayed can be configured in the configurations settings in section Unreal Engine:



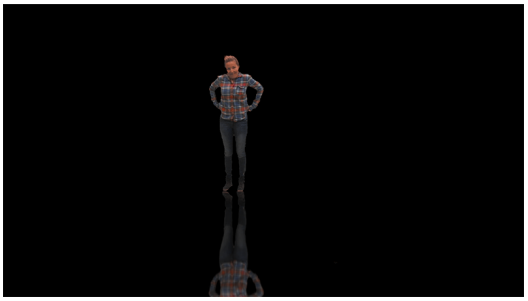
Information: The camera delay is automatically enabled/used when the *aux_rendering* option is selected.

Talent Reflection

Information: This feature requires Viz Engine 4.3.1 or higher.

Talent reflection can be set up to allow an more realistic looking Virtual Studio and tighter integration between Viz Engine components and the Unreal Engine backdrop rendering. Make sure your integration is working as described in the previous sections. The reflection itself is an effect that is rendered on Viz Engine and composed on top of the Unreal rendering.

Copyright © 2026 Vizrt

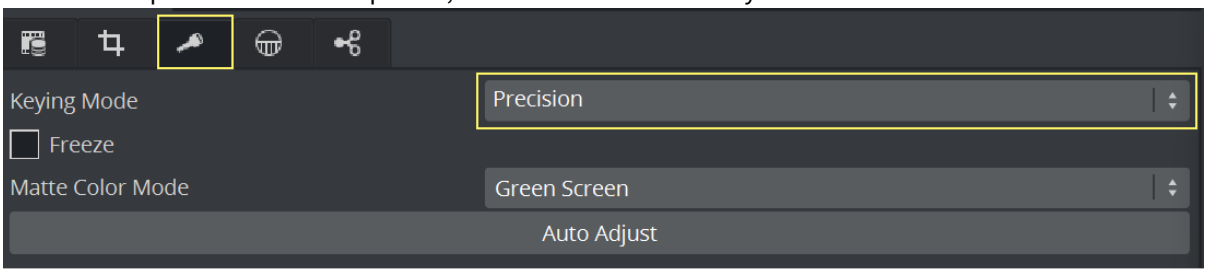
Asset	Description
	Unreal Engine backdrop rendered and sent to Viz Engine via SHM channel.
  	• Live Video Input. • Precision Keyer applied. • Talent Reflection applied.

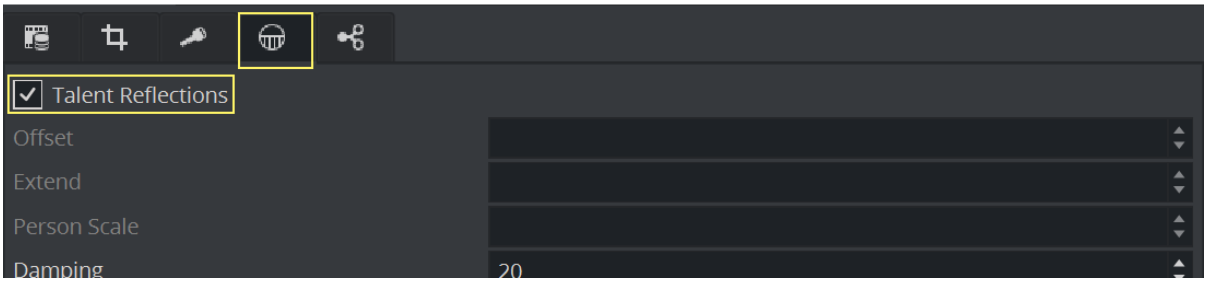
Asset	Description
	Final rendered result with masking applied.

Note: The images are not taken from a production set, it has been built for demonstration purposes only.

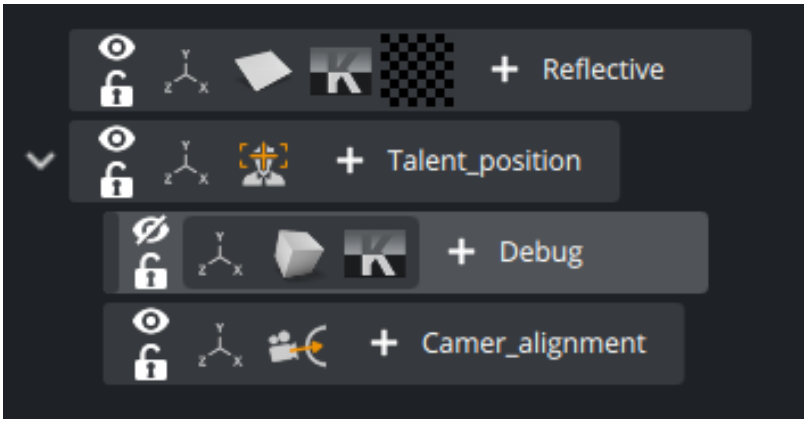
Follow the following steps to setup a Talent reflection with an Unreal Backdrop:

- Add your Unreal SHM Output as Background.
- Rendersequence needs to be *External VS*.
- Put your Live Input as Foreground.
- On the LiveInput Media Asset Properites, make sure the Chroma Keyer is set to *Precision*:

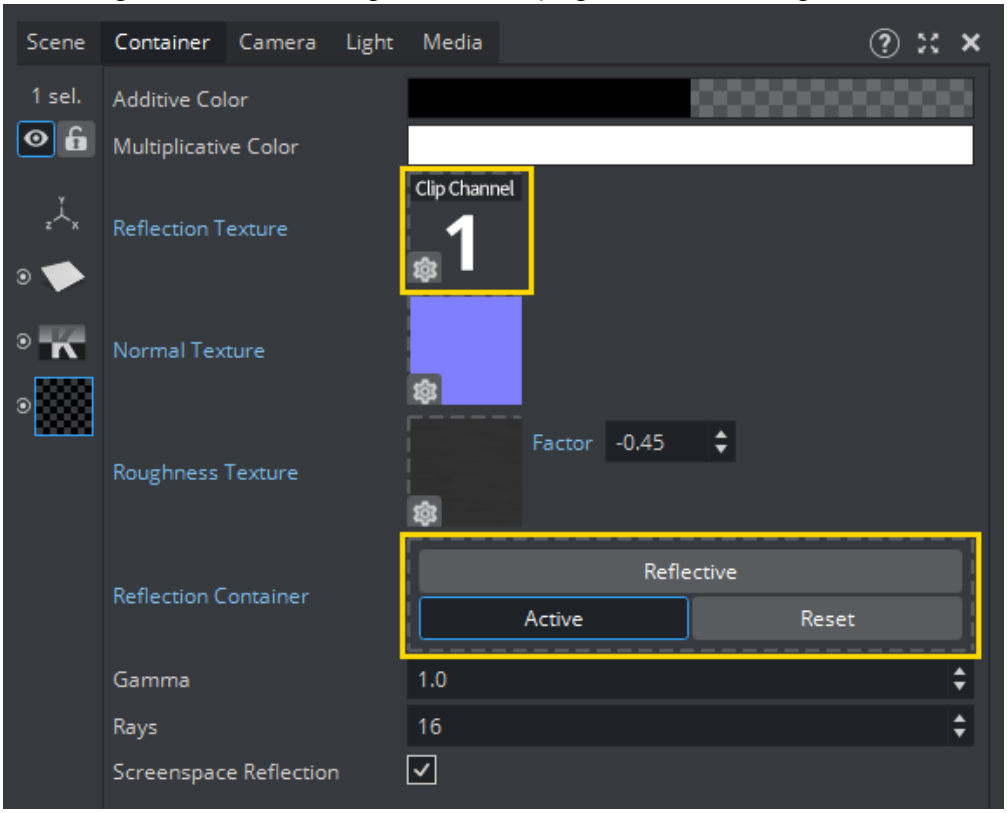




- Setup the following container:



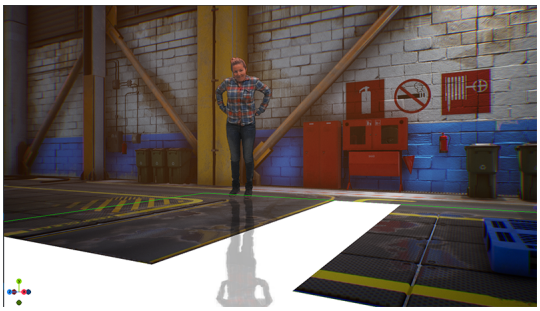
  - **reflection:** The container that finally renders the Talent Reflection. It needs to have a Planar Viz Engine Reflection plug-in assigned.
  - **talentPosition:** The container that reads the detected position of the talent from Precision Keyer. The two sub containers:
    - **debug:** For debugging purposes. If the sphere appears on the shoe of the talent, the talent position tracking works correctly.
    - **cameraAlignment:** The position that is used by the Reflection plug-in. It has a Look At plug-in assigned to have a billboarded orientation to the camera. It should be set to *Billboard - XY Screen*.
- It is important that the **reflection** container holds a Key plug-in. Otherwise it is not visible.
- The rectangle on the **reflection** container should match the floor position in Unreal.
- The settings for the Planar Viz Engine Reflection plug-in are the following:



- The important settings are the Reflection Texture, this is the one the Precision Keyer is using and the Reflection Container, this is the container that holds the detected person position. Please use the *Screenspace Reflection* option (the textures for *Normal* map and *Roughness* are optional).

Once these settings have been applied, the talent should now reflect now on the virtual floor. To finetune your look, you can simulate rough surfaces using a normal and/or roughness map.

Tip: If you have different areas like in our example above, you can use some mask geometry to mask out such surfaces.

Asset	Description
	Masking areas. A basic geometry that matches the floor structure in the Unreal Scene.
	Mask applied. This shows the effect only on the "wet part" of the floor. Mask Source plug-in on the Geometry. Mask Target plug-in on the reflection container.

Unreal Engine as the Compositor

Starting from version 5.0 of the VizrtMedia plug-in, it's possible for the Unreal Engine to act as a Keyer and compositor, receiving video input from the Viz Engine. Please refer to Precision Keyer Compositing on UE5.

Acknowledgements

The Broadcast Studio Set shown in this guide is the property of Warren Marshall. The dragon is part of the Quadruped Fantasy Creatures Pack and is the property of PROTOFACTOR INC. Thanks to Steffi Mair.

Copyright © 2026 Vizrt

24.1.17 Control Objects on Unreal Engine

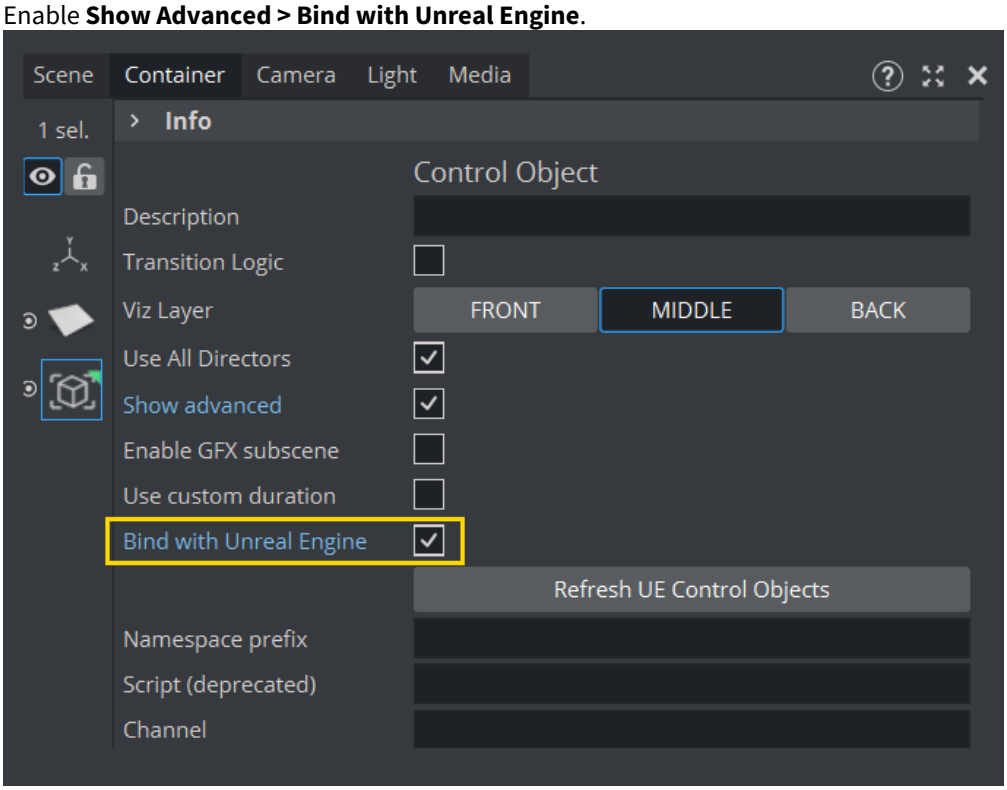
This feature allows exporting several parameters (float, textures, string, etc.) on Unreal Engine through specialized Actors. These parameters can then be accessible through Viz Engine and updated using the ordinary Control Objects functionality. Currently, we support ControlAnimation, ControlImage, ControlText and ControlMaterial.

A sample Unreal Engine level is provided with the Vizrt Media plug-in. You can find it at */Plugins/VizrtMedia/Content/SampleScenes/ControlObjects*.

The following instructions use the Viz Artist user interface, but control objects can also be used by Viz Trio using **Tab Fields**.

Setup Control Objects in Viz Engine

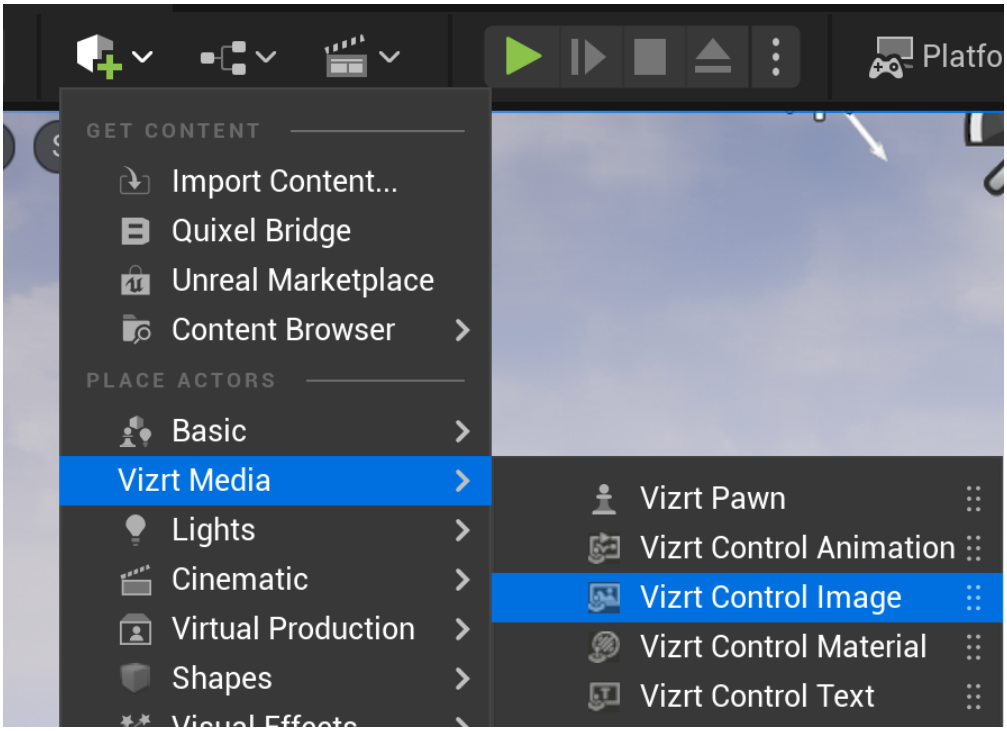
1. Create a scene with a single container.
2. Add a ControlObject plug-in.
3. Enable **Show Advanced > Bind with Unreal Engine**.



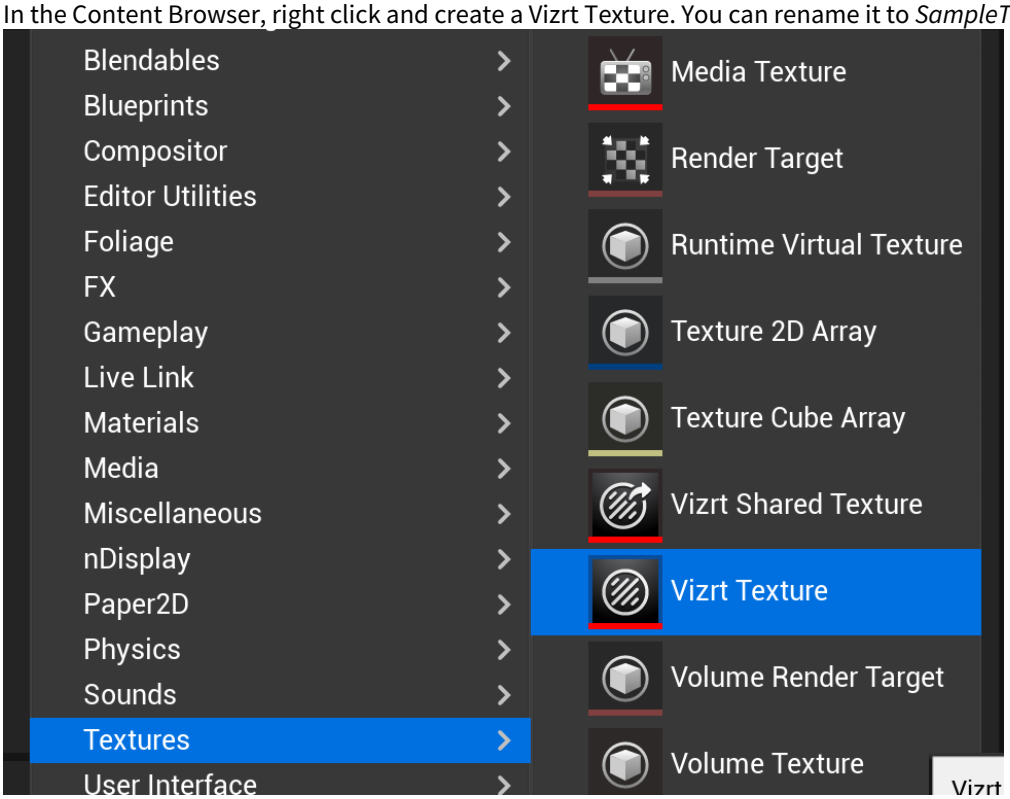
Control Image

1. In Unreal, add a Vizrt Control Image actor to the world.

Copyright © 2026 Vizrt

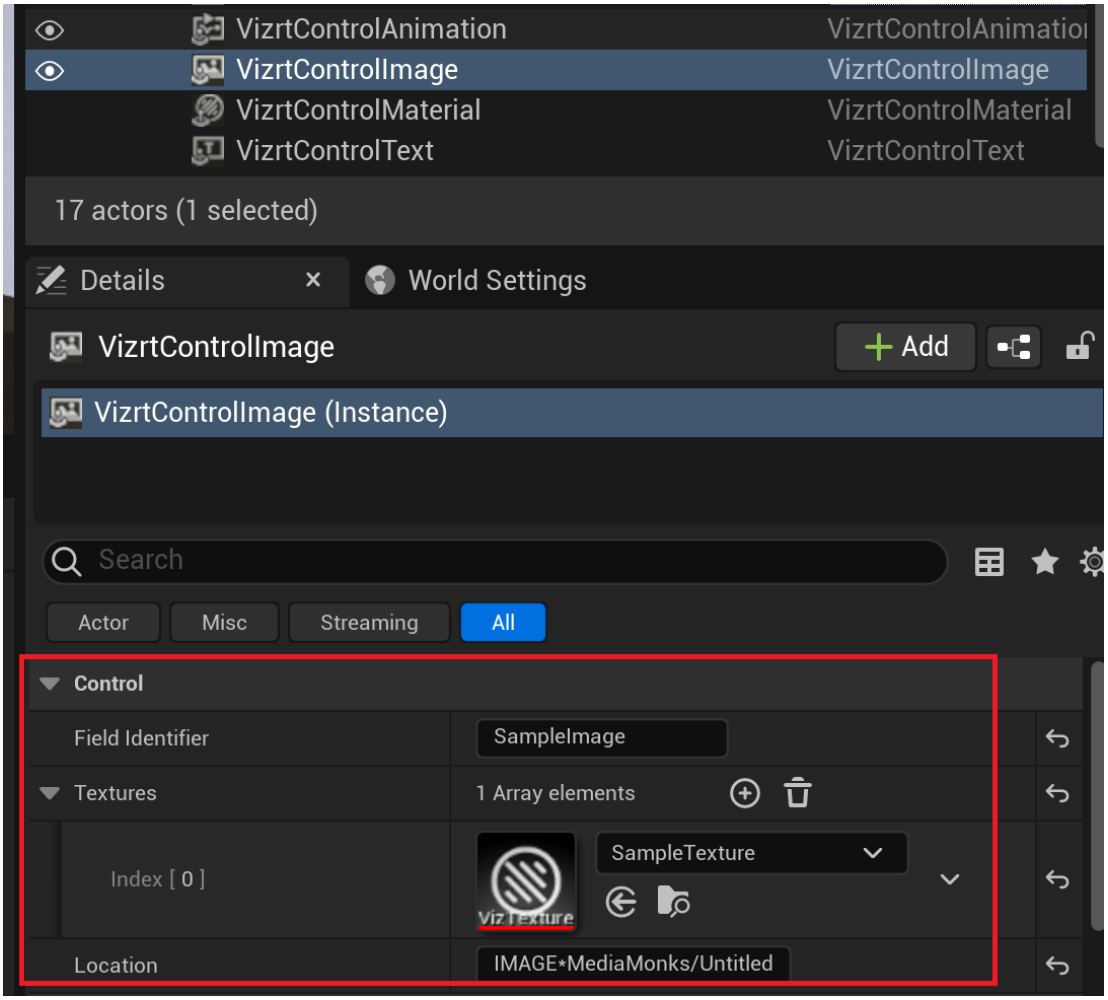


2. In the Content Browser, right click and create a Vizrt Texture. You can rename it to *SampleTexture*.

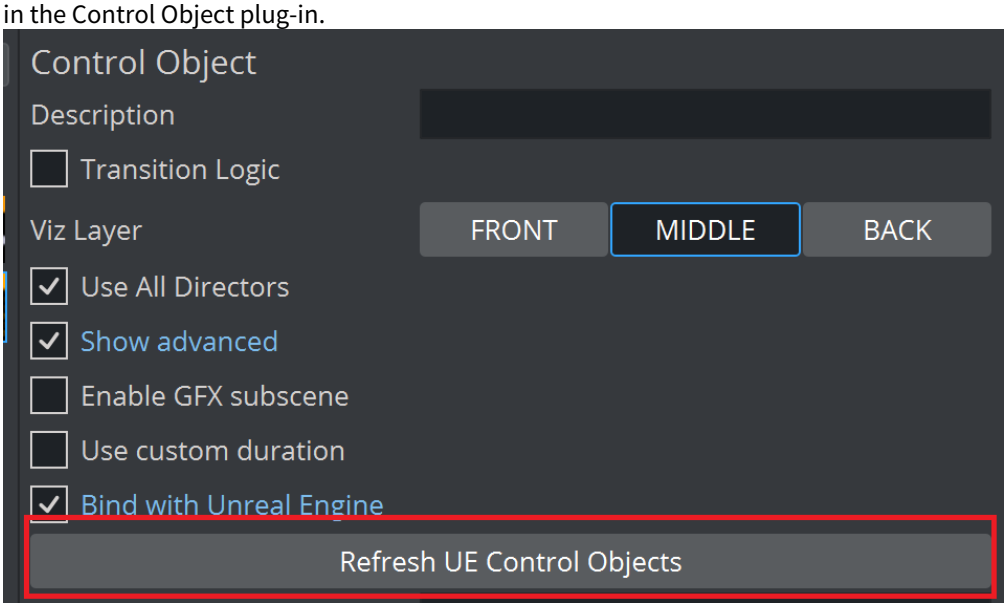


3. Select the newly created VizrtControlImage actor and go to Details.
4. In **Control > Field Identifier** set a name. This identifies the control object Field name in Viz Engine. In **Control > Textures** add the VizrtTexture created before. Its **Control > Location** property receives the image location from the Viz Engine Control object.

Copyright © 2026 Vizrt

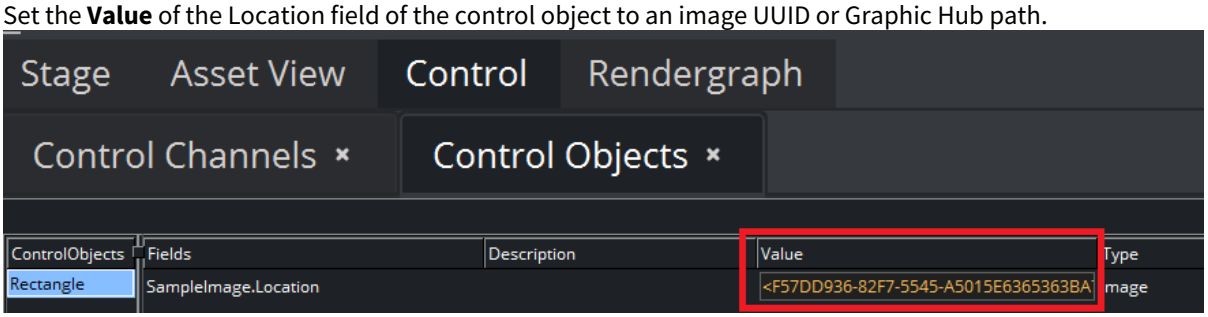


5. In Viz Engine go to the **Control** tab. Select **Control Objects.**
6. The control object for the Unreal ControlImage is shown. If not. click the **Refresh UE Control Objects** button in the Control Object plug-in.

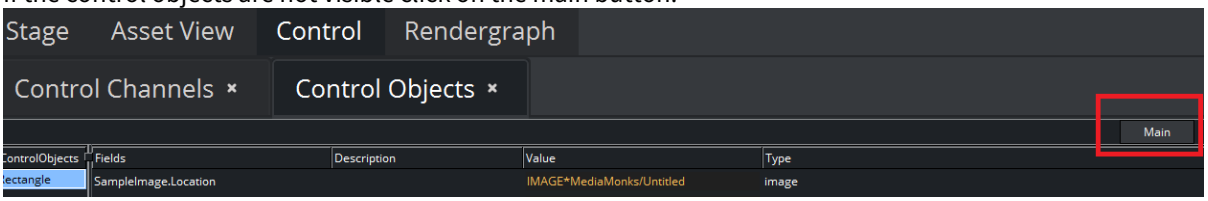


Copyright © 2026 Vizrt

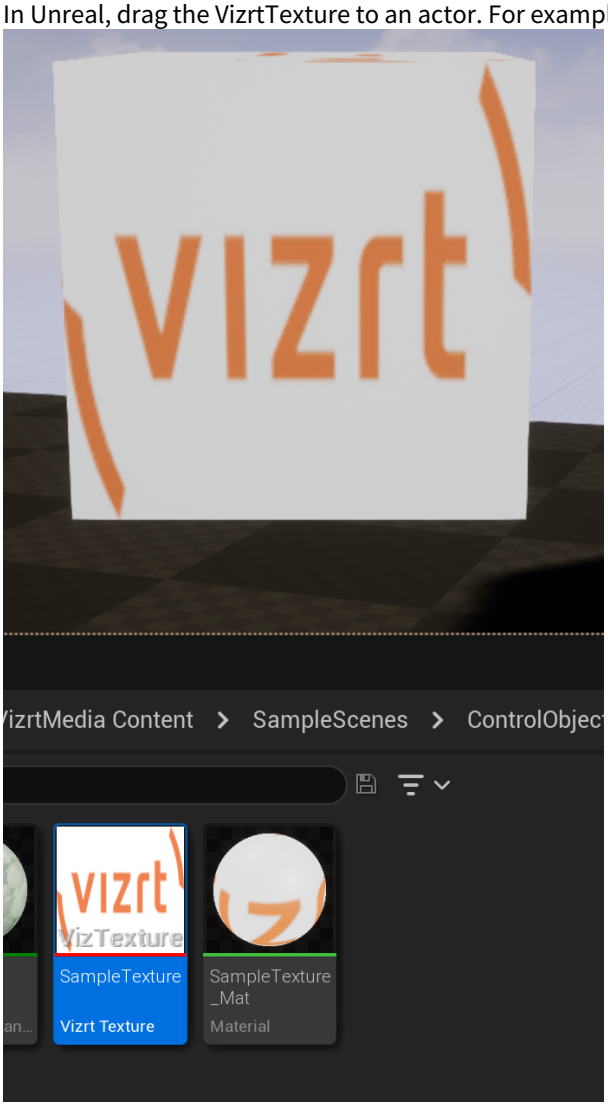
7. Set the **Value** of the Location field of the control object to an image UUID or Graphic Hub path.



If the control objects are not visible click on the main button.



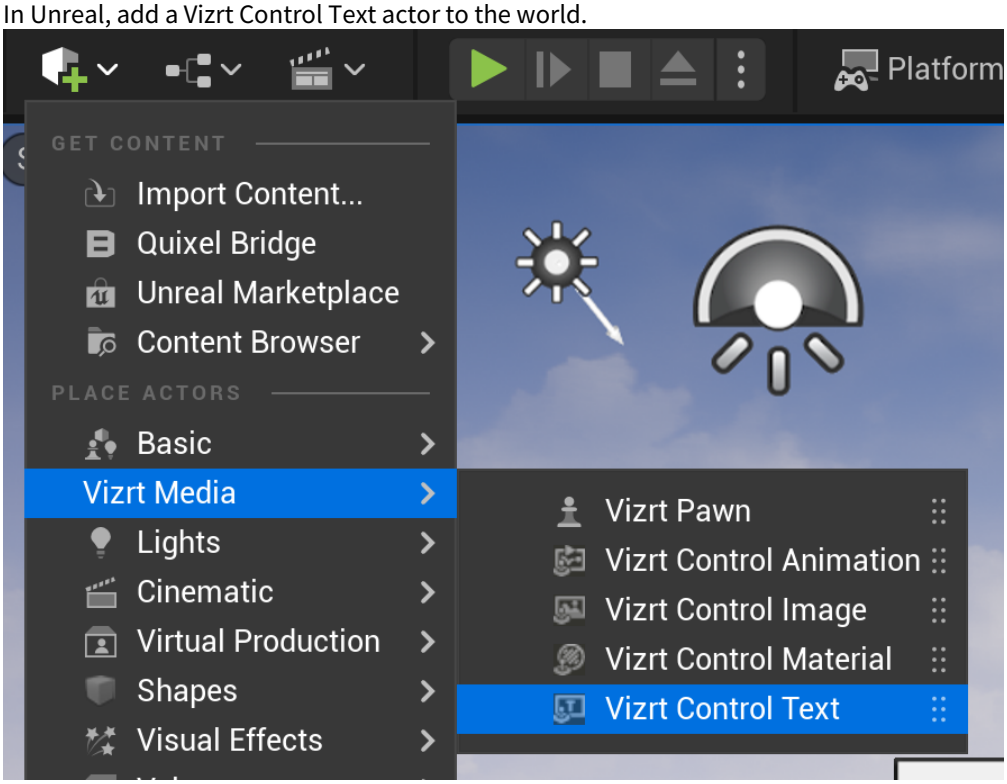
8. In Unreal, drag the VizrtTexture to an actor. For example, a cube.



Copyright © 2026 Vizrt

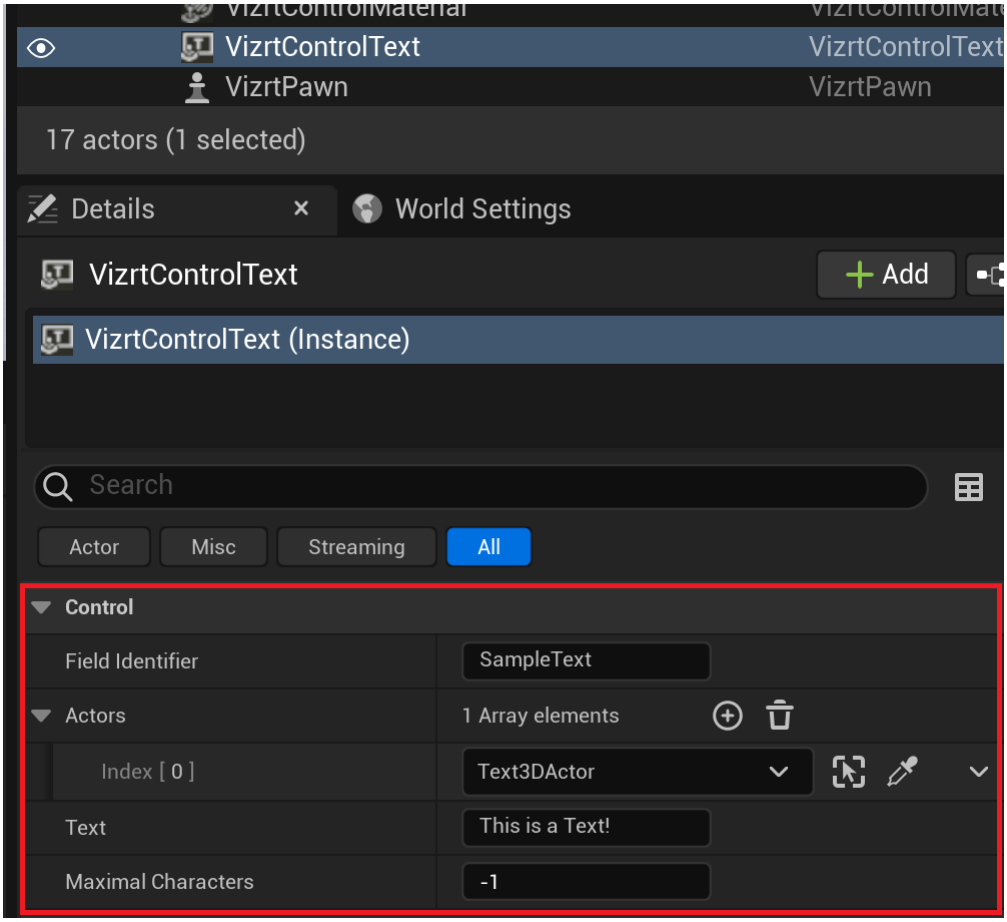
Control Text

1. In Unreal, add a Vizrt Control Text actor to the world.

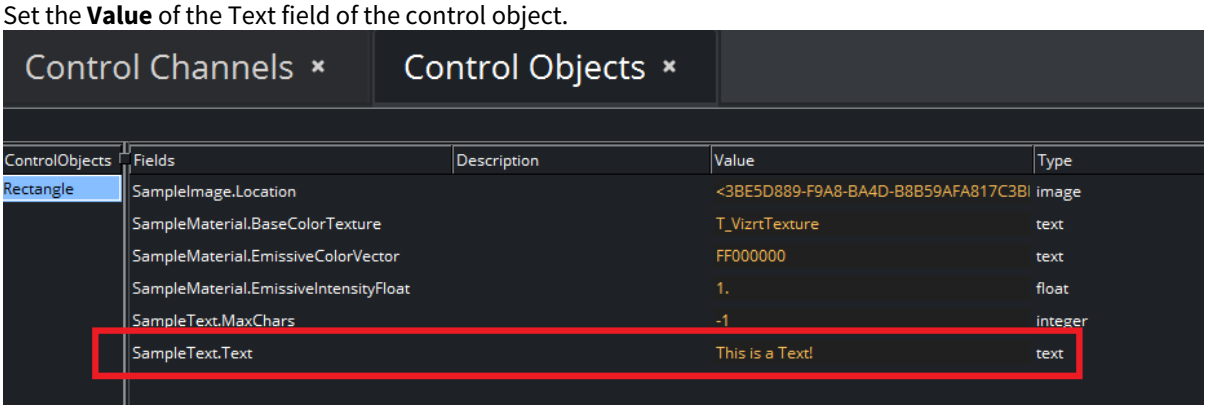


2. Select the newly created Vizrt Control Text actor and go to Details.
3. In **Control > Field Identifier** set a name. This identifies the control object Field name in Viz Engine.
   In **Control > Actors**, specify an Unreal actor which receives the text from the control object. For example a Text3D actor. The **Control > Text** property receives the text from the Viz Engine Control Object.

Copyright © 2026 Vizrt



4. In Viz Engine, go to the **Control** tab. Select **Control Objects.**
5. The control object for the Unreal ControlText is shown. If not, click the **Refresh UE Control Objects** button in the Control Object plug-in.
6. Set the **Value** of the Text field of the control object.

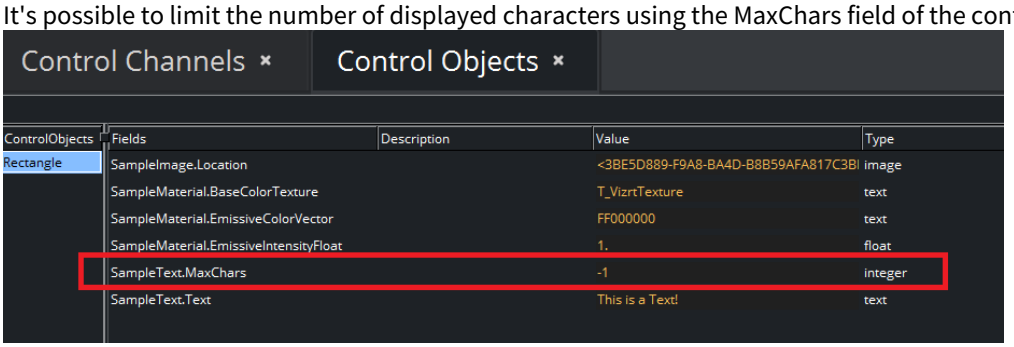


7. The Unreal Text3D actor updates its text.

Copyright © 2026 Vizrt

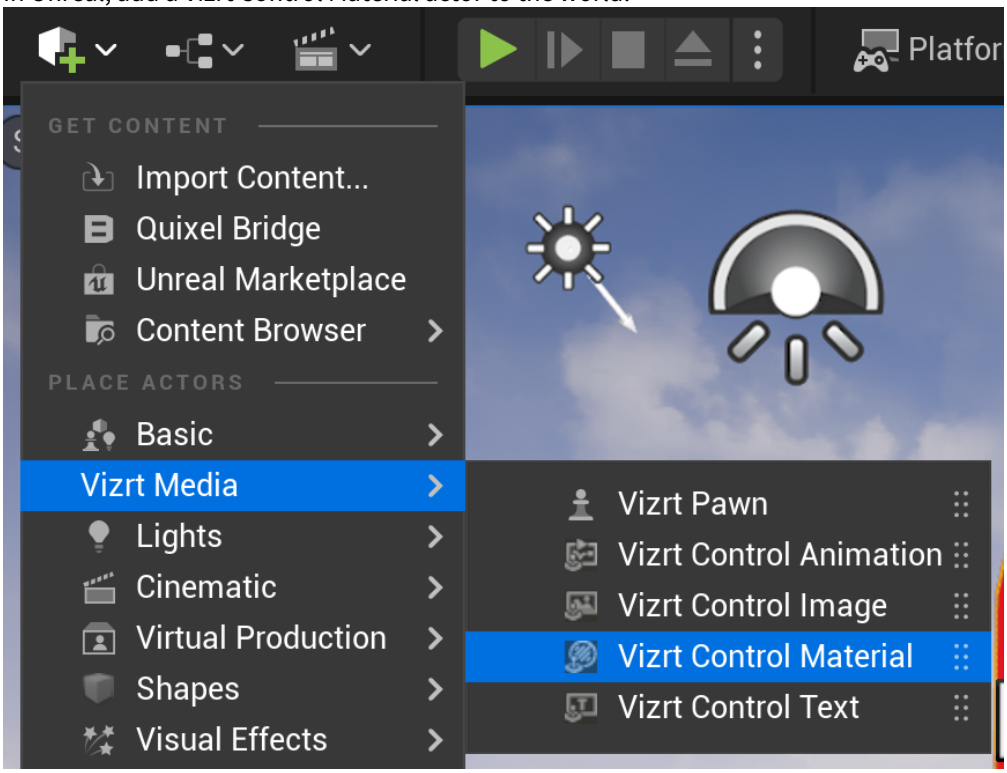


8. It's possible to limit the number of displayed characters using the MaxChars field of the control object.



Control Material

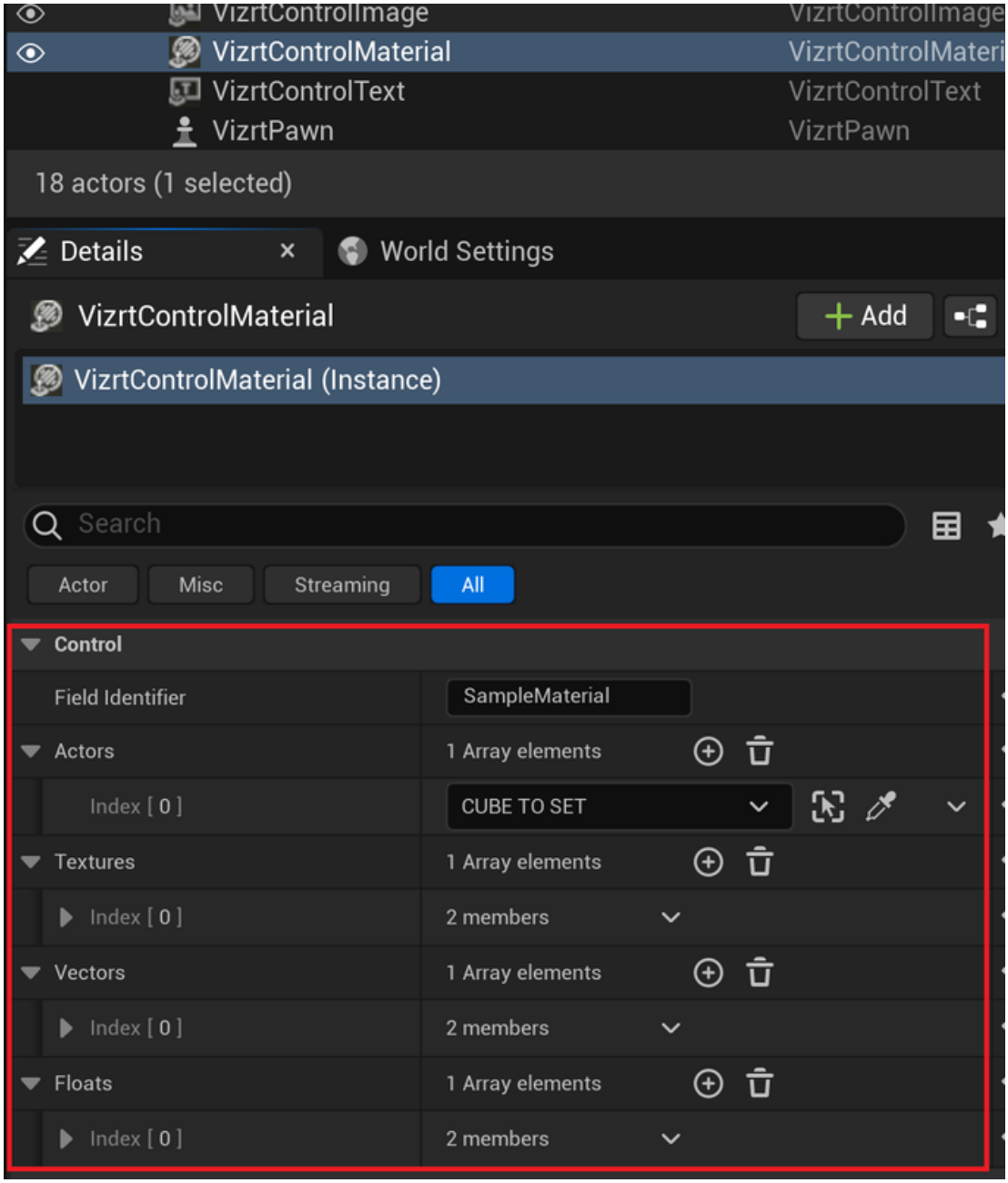
1. In Unreal, add a Vizrt Control Material actor to the world.



2. Select the newly created VizrtControlMaterial actor and go to Details.

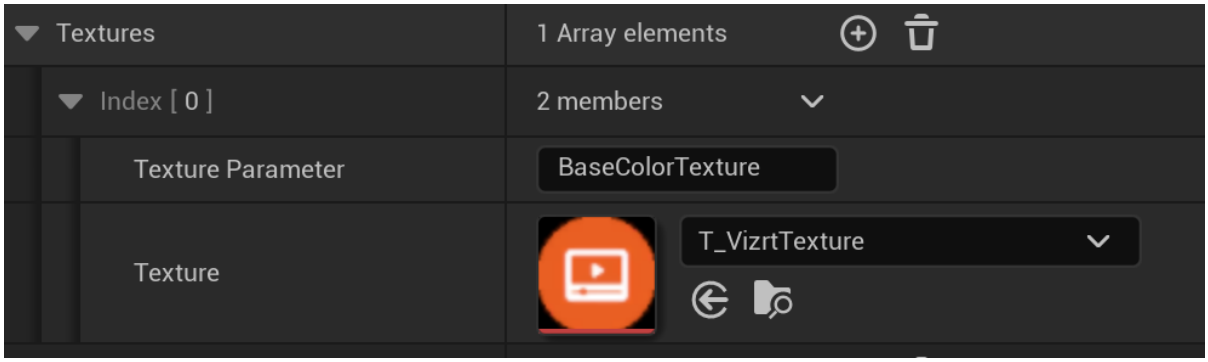
Copyright © 2026 Vizrt

In **Control > Field Identifier** set a name. This identifies the control object Field name in Viz Engine. In **Control > Actors**, specify the Unreal actors which are to have their materials changed by the Control Object. For example, specify a Cube actor (in this case named *CUBE TO SET*).

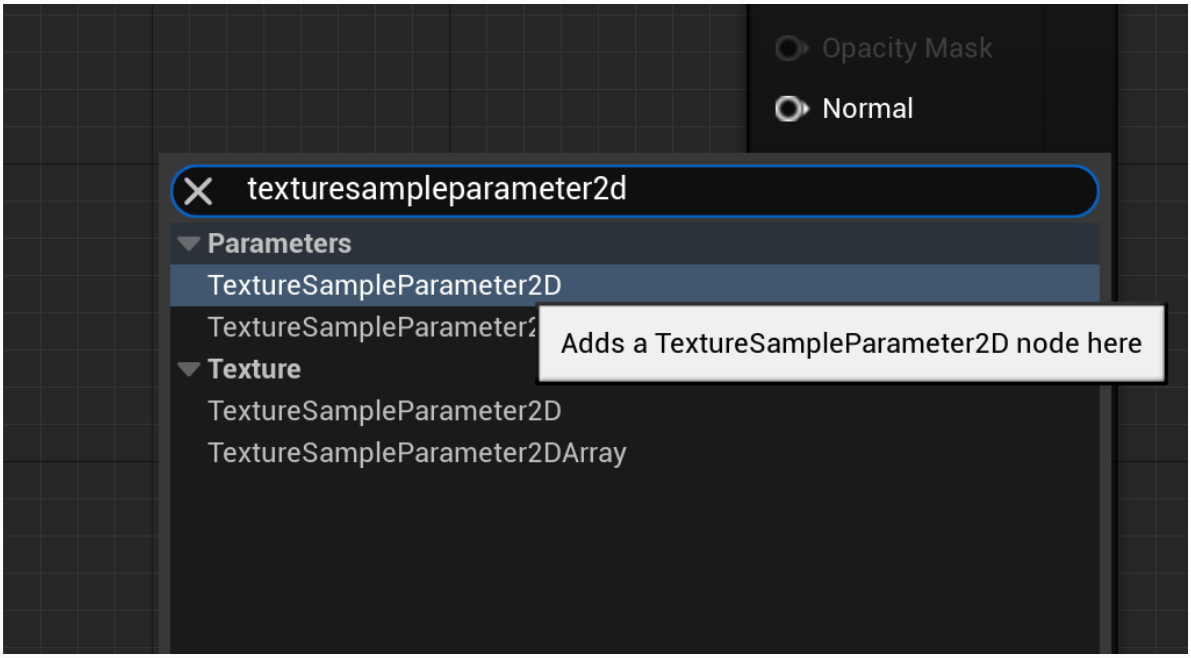


In **Control > Textures**, specify images of type VizrtTexture which links to a parameter of type TextureSampleParameter2D in the actor material. The **Texture Parameter** field must have the same name as the material parameter in the target actor.

Copyright © 2026 Vizrt

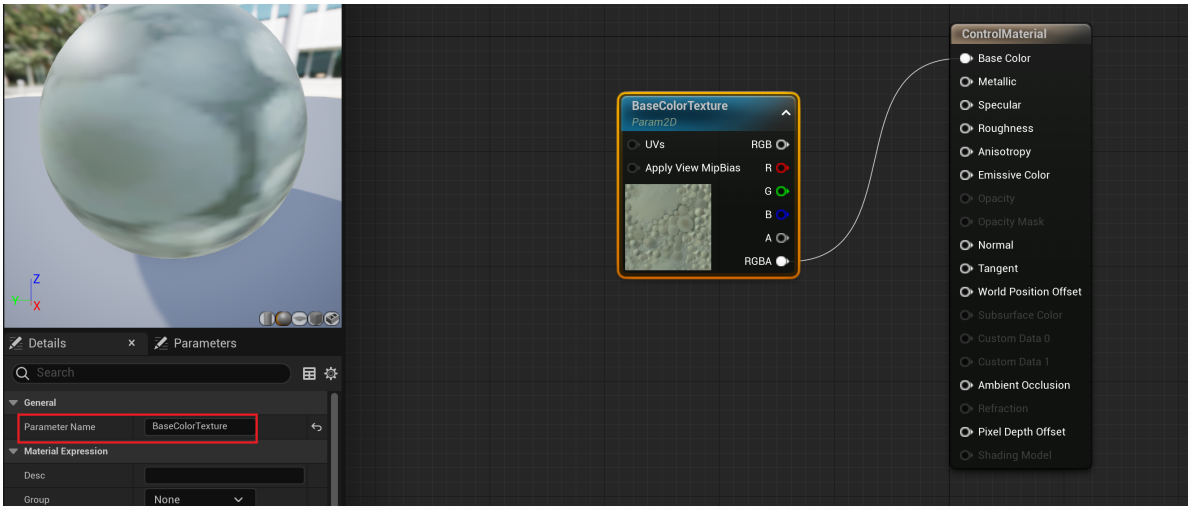


To Setup the Unreal Material Parameter: To create the matching parameter in the material, double click the material applied to the target actor to open the material editor. Create a **TextureSample2D** parameter by double clicking in the empty space of the material editor, searching for **texturesampleparameter2d** and selecting **TextureSampleParameter2D**.

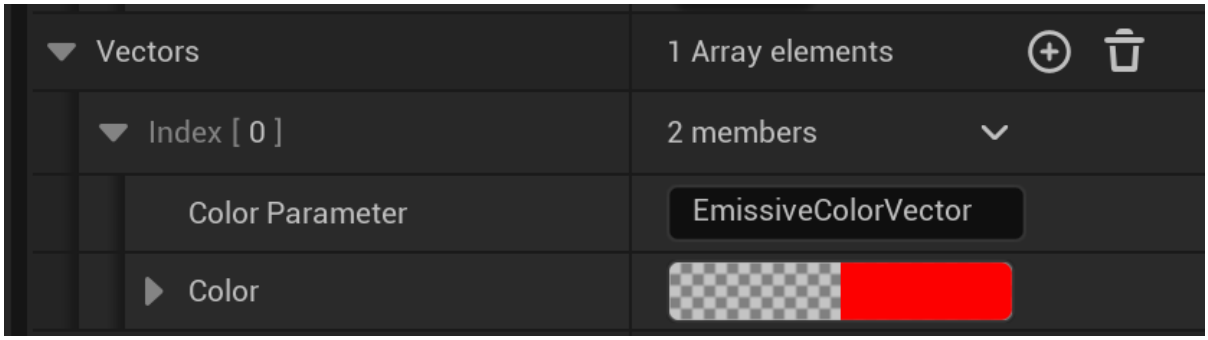


Rename the parameter to match the **Texture Parameter** specified above. In this case it was named **BaseColorTexture**.
Drag the RGBA node of the parameter and connect it to **Base Color** in the material.

Copyright © 2026 Vizrt

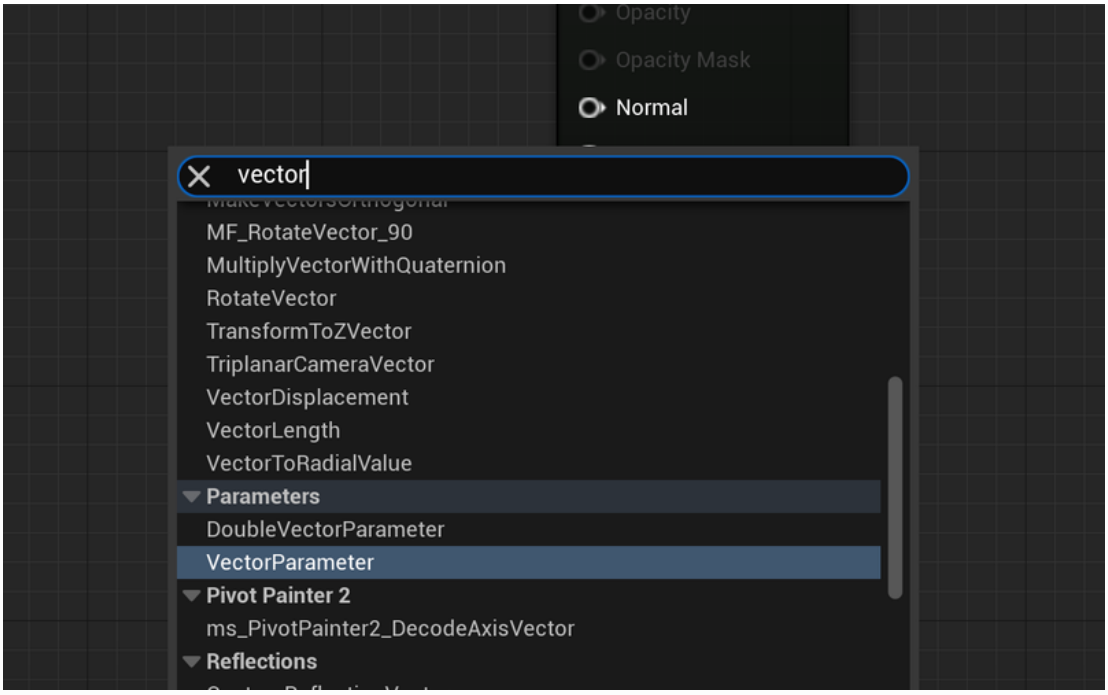


In **Control > Vectors** specify a color/vector which links to a parameter of type Vector in the actor material. The **Color Parameter** field must have the same name as the material parameter in the target actor.

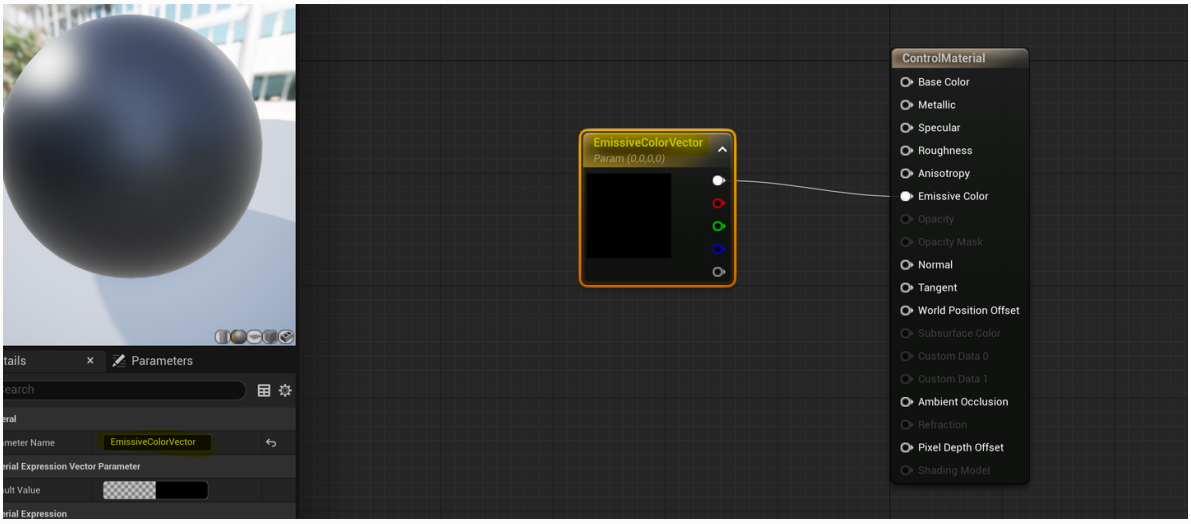


To Setup the Unreal Material Parameter: To create the matching parameter in the material, double click the material applied to the target actor to open the material editor. Create a **Vector** parameter by double clicking in the empty space of the material editor, searching for *vector* and selecting **VectorParameter.**

Copyright © 2026 Vizrt



Rename the parameter to match the **Color Parameter** specified above. In this case, it was named **EmissiveColorVector**. Drag the RGBA node of the parameter and connect it to **Emissive Color** in the material.

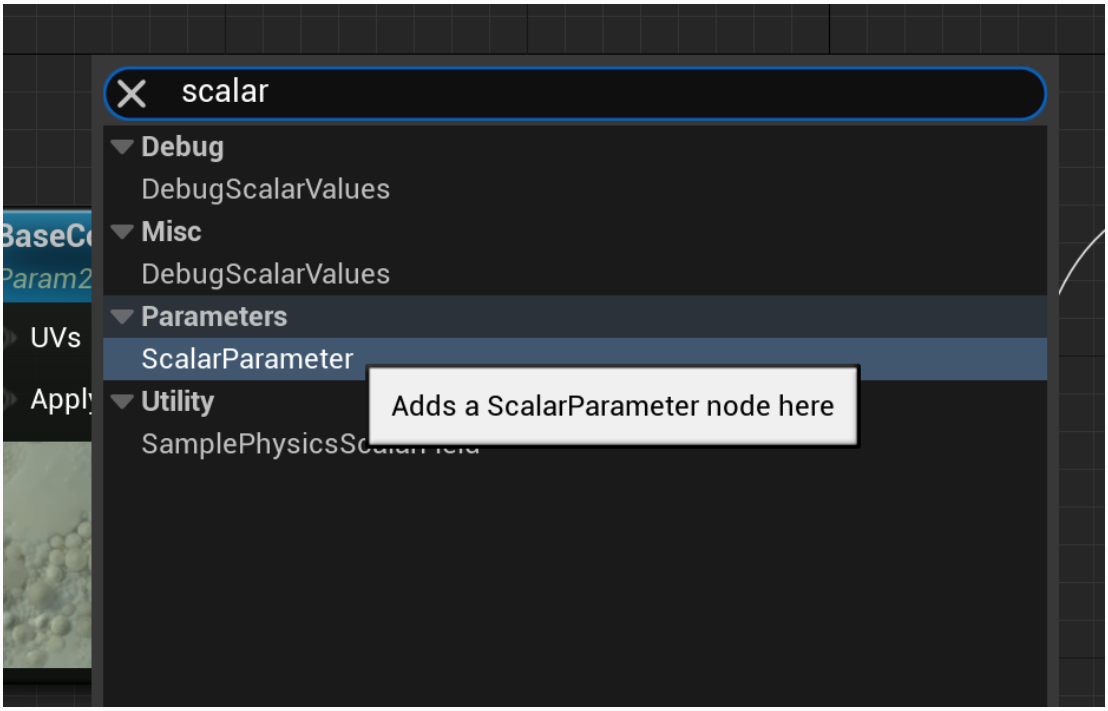


In **Control > Floats**, specify a float number which links to a parameter of type float in the actor material. The **Float Parameter** field must have the same name as the material parameter in the target actor.

Copyright © 2026 Vizrt

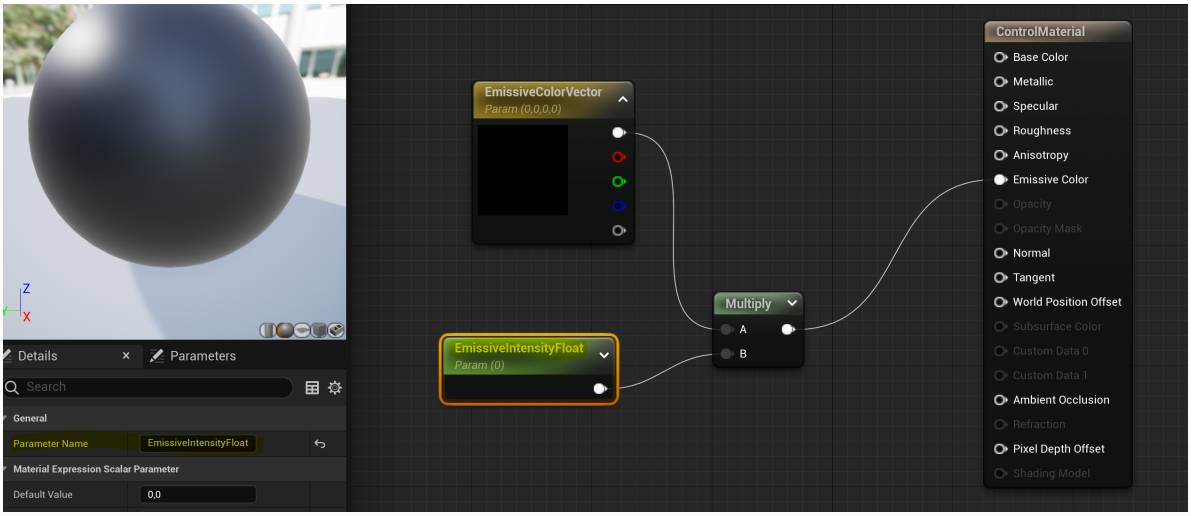


To Setup the Unreal Material Parameter: To create the matching parameter in the material, double click the material applied to the target actor to open the material editor. Create a **Scalar** parameter by double clicking in the empty space of the material editor, searching for *scalar* and selecting **ScalarParameter.**



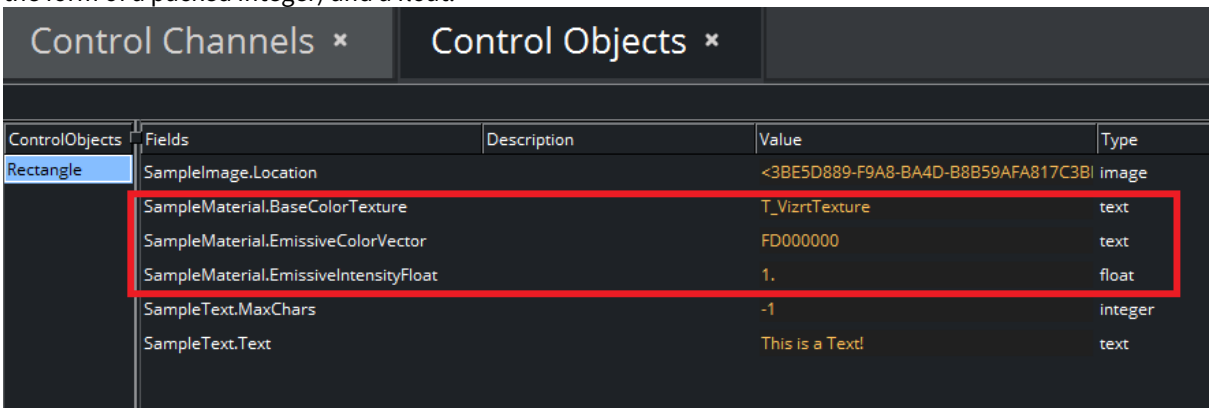
Rename the parameter to match the **Texture Parameter** specified above. In this case, it was named **EmissiveIntensityFloat**. Drag the output node of the parameter and connect it to the rest of the graph. In this case it is connected to a **Multiply** node that scales the intensity of the emissive color parameter.

Copyright © 2026 Vizrt



Note: Inspect the material M_Cube present in the sample scene */Plugins/VizrtMedia/Content/SampleScenes/ControlObjects* for a material with parameters matching the steps above.

1. In Viz Engine, go to the **Control** tab. Select **Control Objects**.
2. The control object for the Unreal ControlMaterial is shown. If not, click the **Refresh UE Control Objects** button in the Control Object plug-in.
3. Set the **Value** of the corresponding fields of the control object to an Unreal VizrtTexture name, a vector (in the form of a packed integer) and a float.



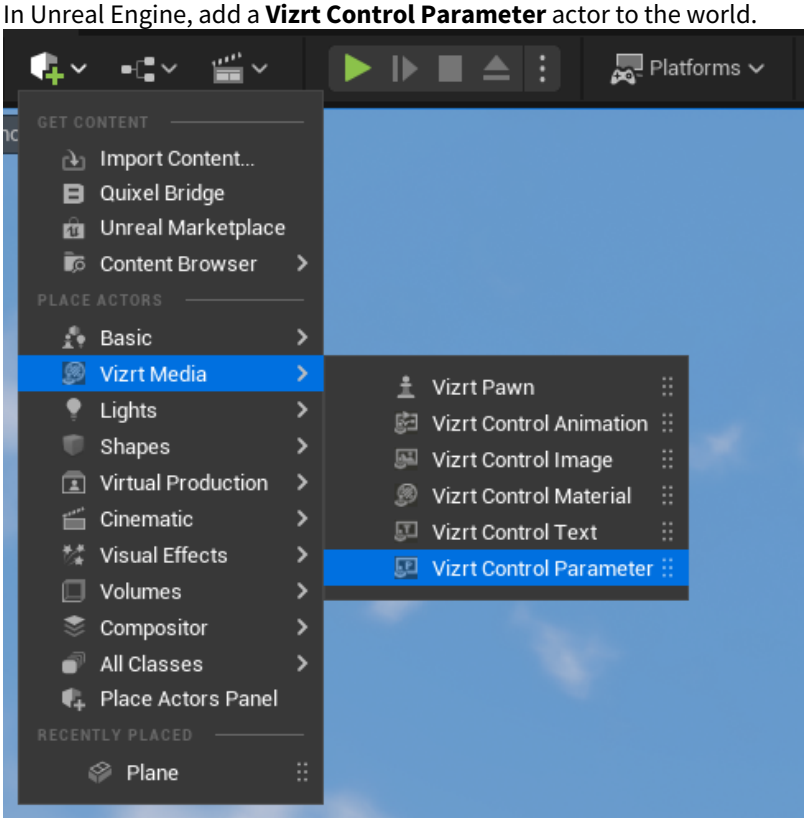
4. In Unreal, the target actor, in this case a cube, is updated.

Copyright © 2026 Vizrt



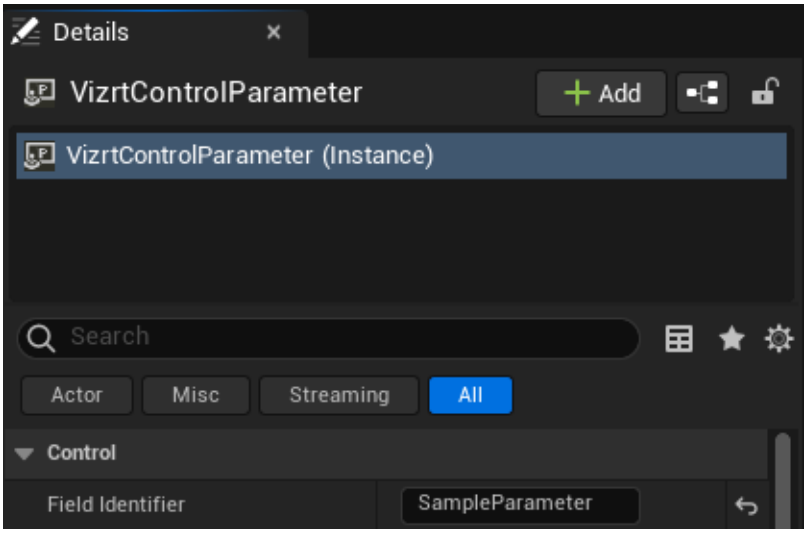
Control Parameter

1. In Unreal Engine, add a **Vizrt Control Parameter** actor to the world.



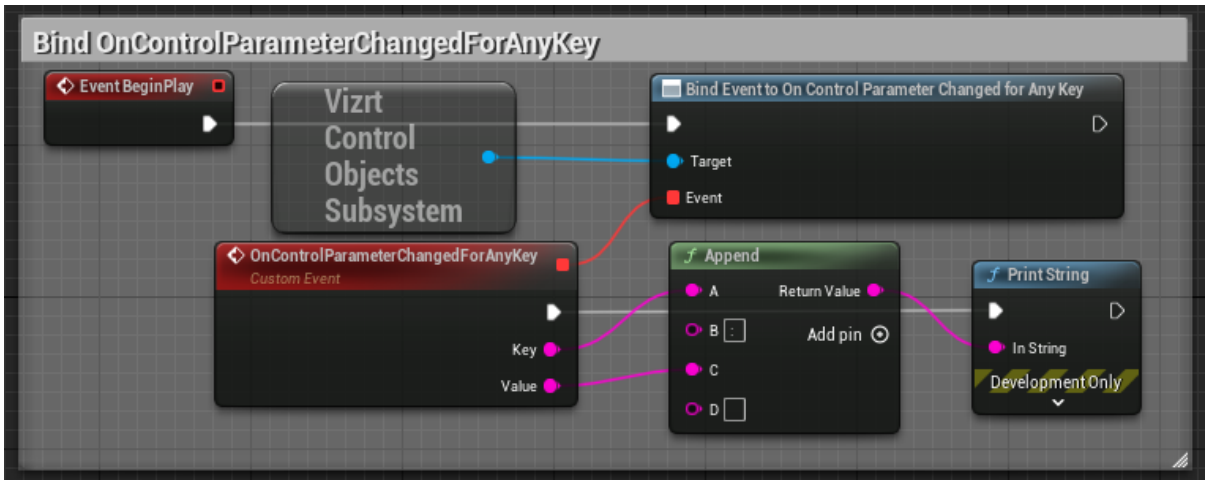
2. Select the newly created **VizrtControlParameter** actor and go to the **Details** panel.
3. In **Control > Field Identifier** set a name. This identifies the control object Field name in Viz Engine.

Copyright © 2026 Vizrt



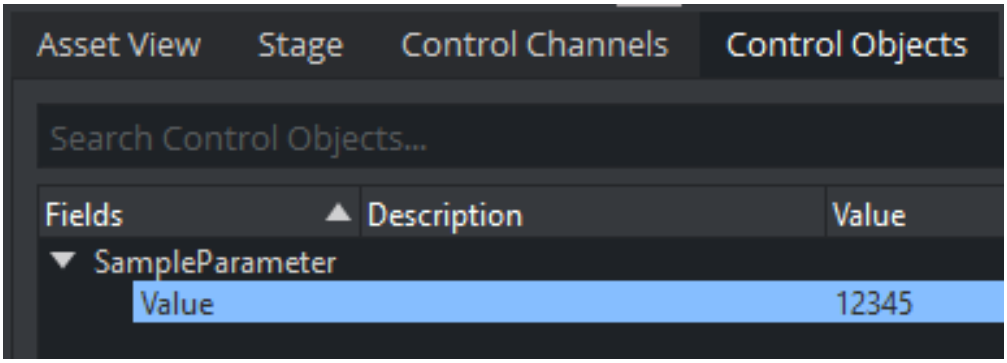
4. In Viz Engine, go to the **Control Objects** panel.
5. The control object for the Unreal ControlParameter is shown. If not, click the **Refresh UE Control Objects** button in the Control Object plug-in.
6. In order to receive updates for ControlParameters on Unreal Engine, we need to setup a Blueprint script, either as a Level Blueprint or as a Blueprint Actor.

> **Note:** All the received key:value are logged into the screen. Therefore, you can use the new key:value to modify a specific property of a specific Actor you control.

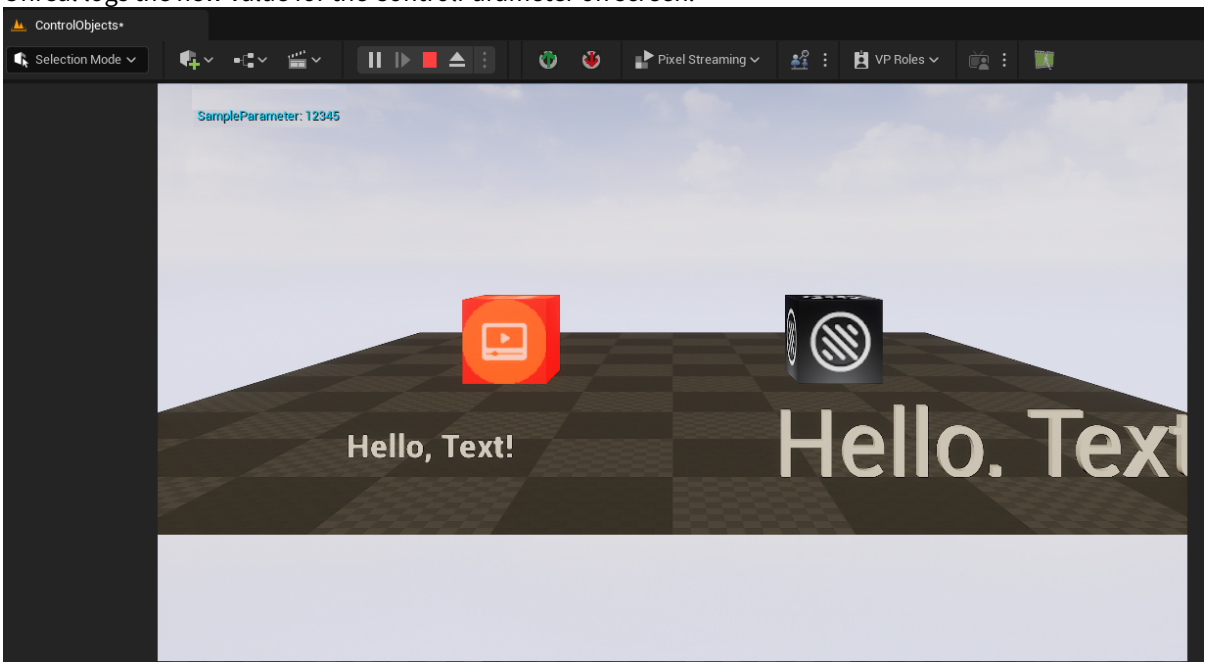


7. Set the **Value** field of the control object.

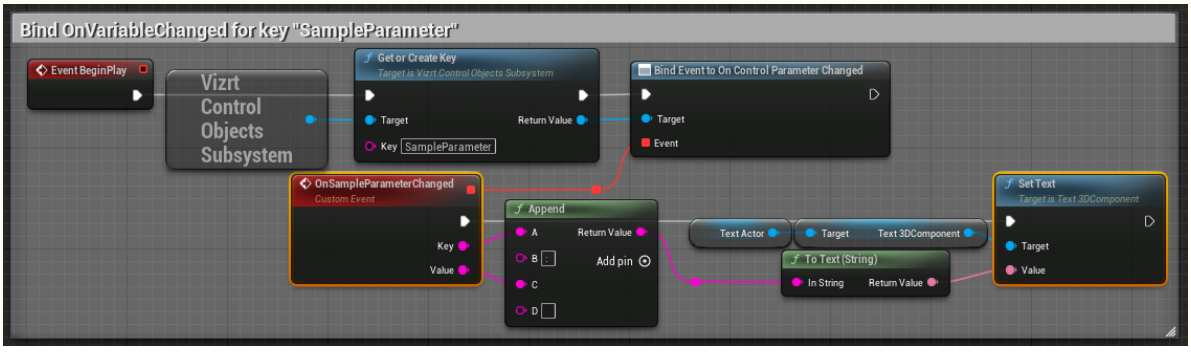
Copyright © 2026 Vizrt



8. Unreal logs the new value for the ControlParameter on screen.



Note: Instead of listening for all ControlParameter updates, you can listen to a specific Field Identifier.The example below shows how this can be done, and use the received Value to set the Text property of an Actor.





Control Animation

The control object for the control animation is not yet available in Viz Engine.

Copyright © 2026 Vizrt

24.1.18 AR Media Compositing with No Keyer

With the release of Viz Engine 5.1, the AR Compositing when not using a Keyer has been given extra functionality which allow for a higher fidelity media output.

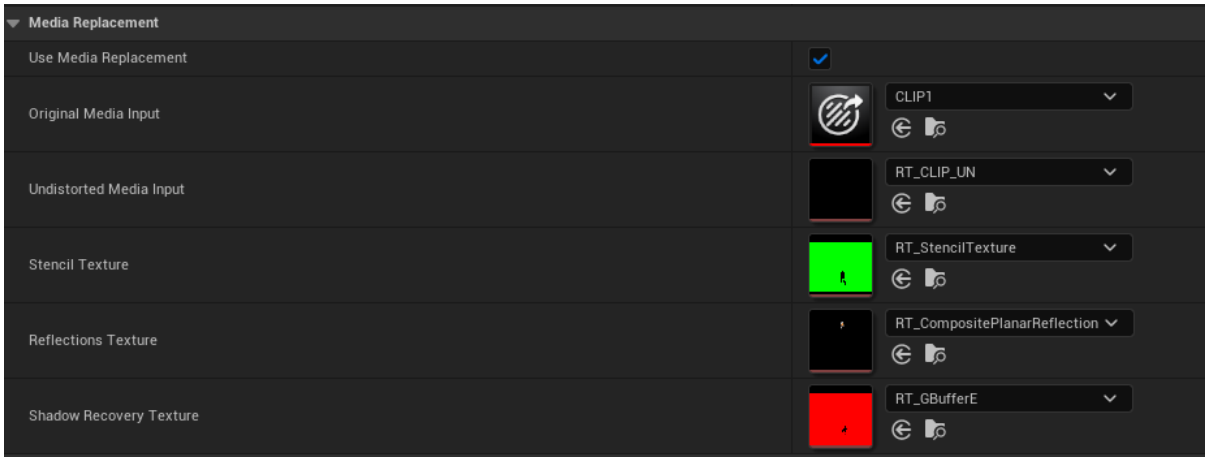
Using the Media Replacement Feature

To use the Media Replacement feature, the Compositor needs to be added into the world and configured with RT_CLIP_UN (or RT_LIVE_UN) as an input. The Keyer should also not be activated.



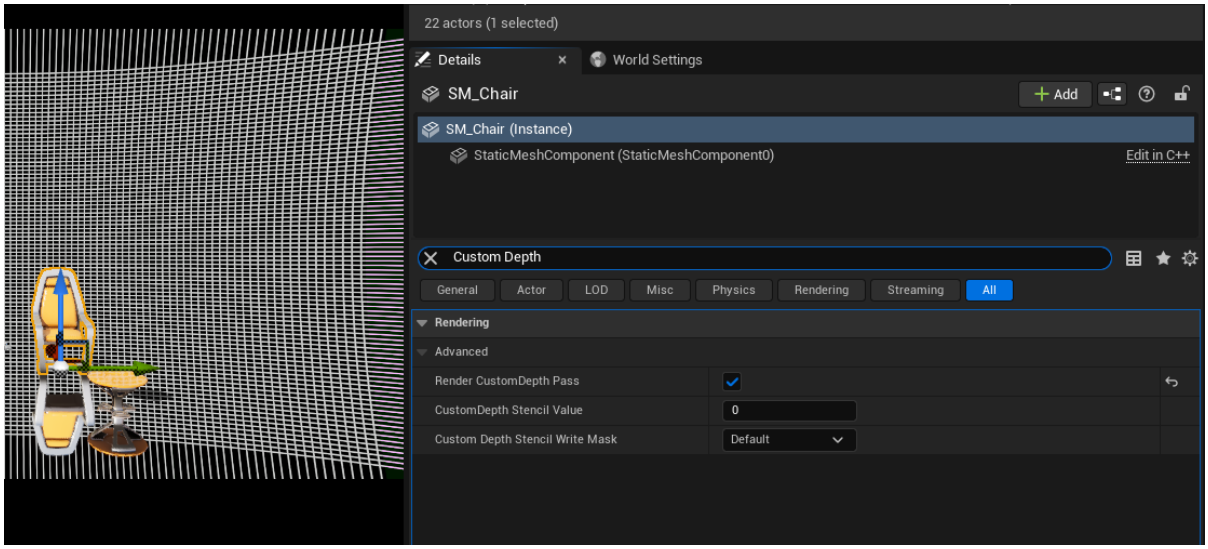


On the AUX1 in the AutoCapture blueprint, activate the **Use Media Replacement** toggle and make sure the proper CLIP/LIVE textures are there as well as the RT_StencilTexture, RT_CompositePlanarReflection and RT_GBufferE.



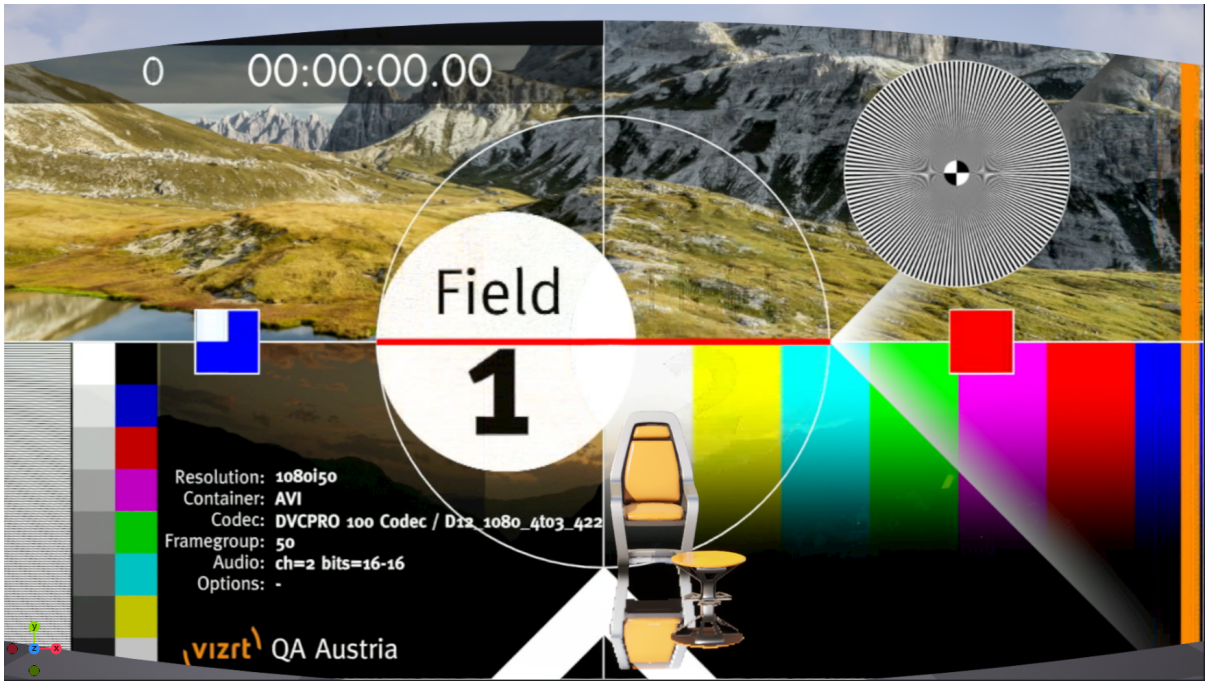
If there are objects in front of the Compositor on your scene, we need to add them to the custom depth and give those objects a value of 0;

Copyright © 2026 Vizrt

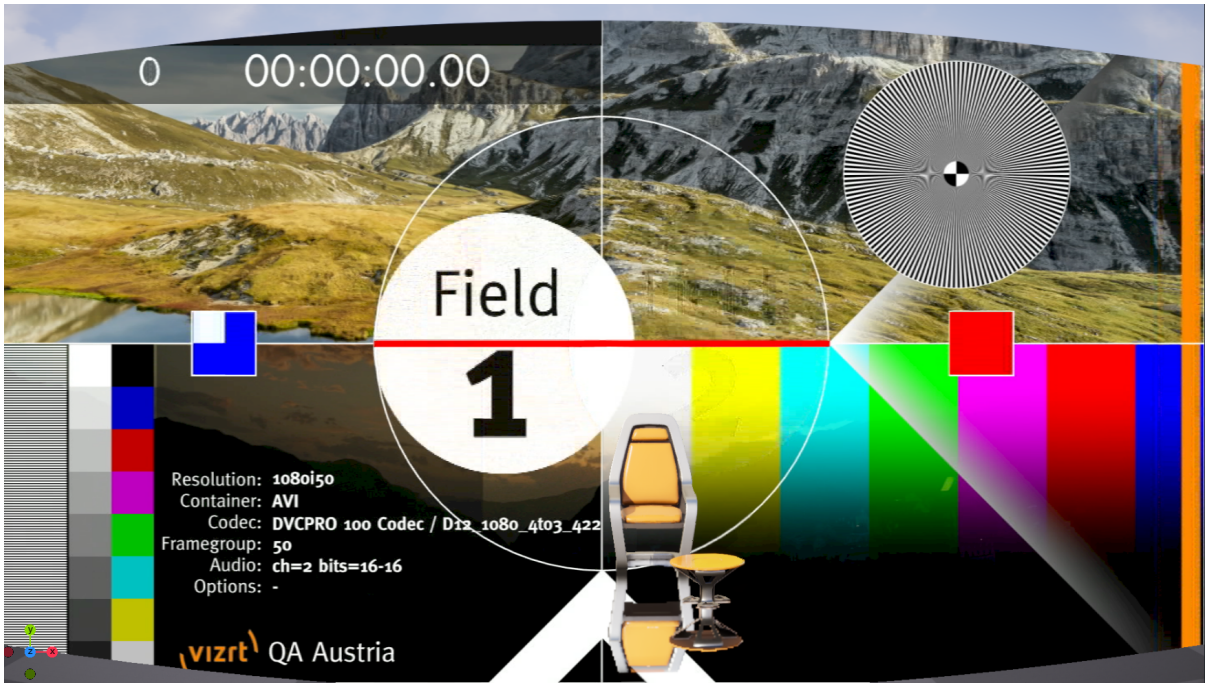


This provides the following result:

Without Media Replacement

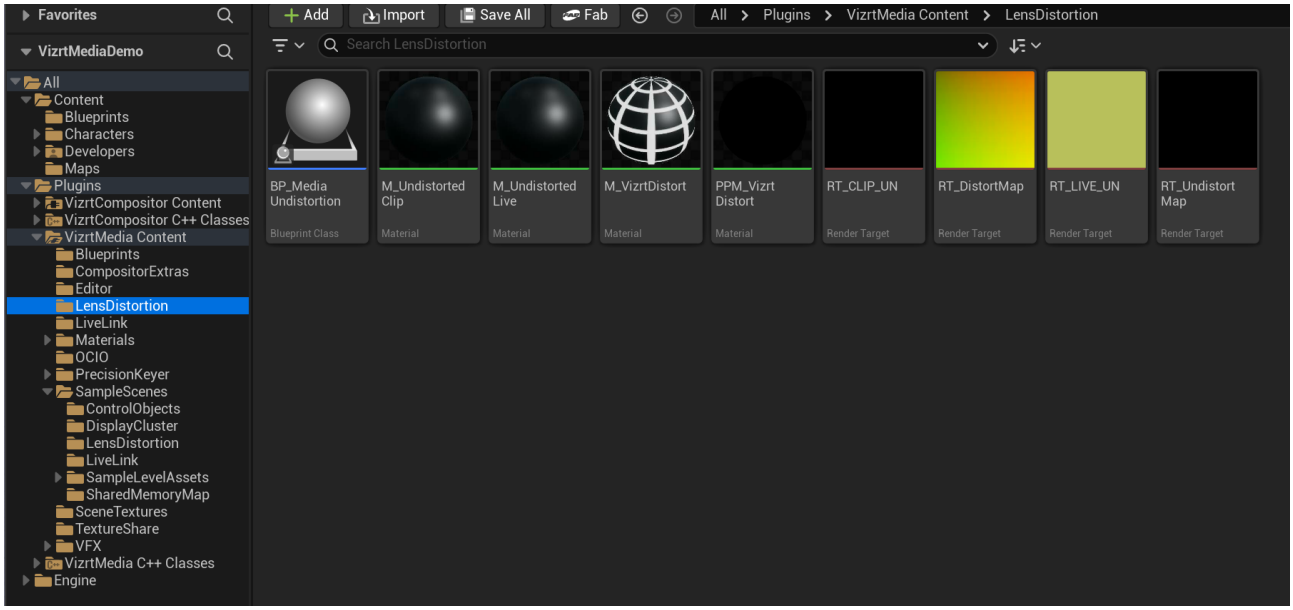


With Media Replacement

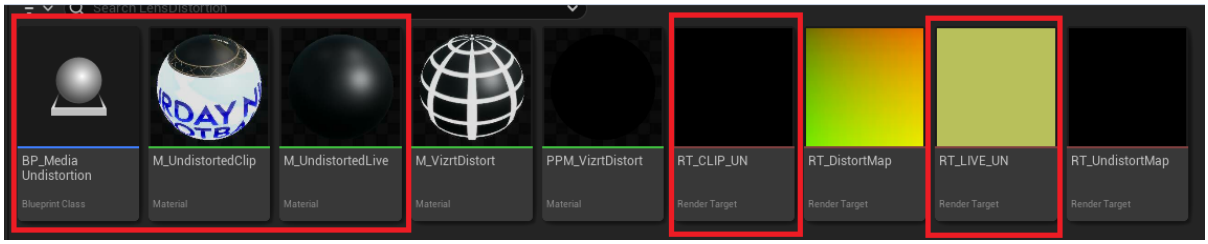


Adding Different Clip/Live Inputs

The current integration already has Materials, Render Targets and Blueprints ready for CLIP1 and LIVE1 but if extra clip or live inputs are needed, the material and render target should be created and added to the BP_MediaUndistortion blueprint. You can find these assets on the VizrtMedia\LensDistortion folder.

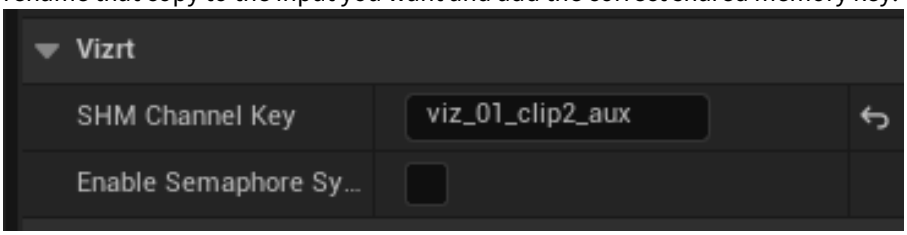


Copyright © 2026 Vizrt

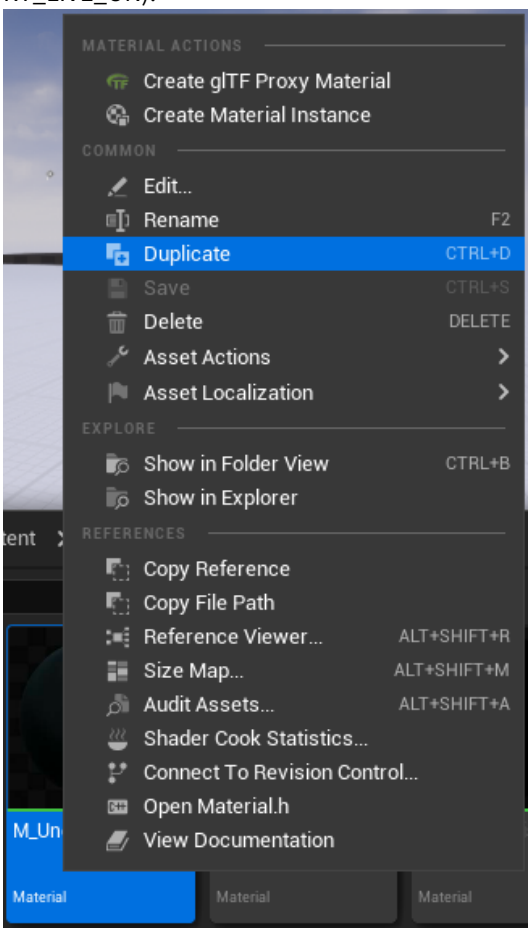


The steps to have a new CLIP/LIVE input added are as follows:

- If the CLIP/LIVE input you want still does not exist in the *TextureShare* folder, duplicate CLIP1 or LIVE1, rename that copy to the input you want and add the correct shared memory key.

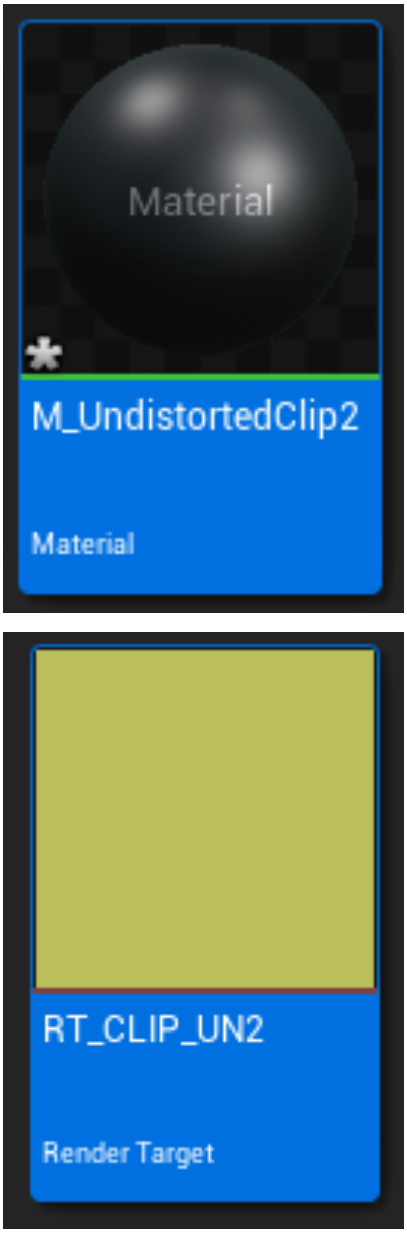


- Duplicate the material (either M_UndistortedLive or M_UndistortedClip) and render target (RT_CLIP_UN or RT_LIVE_UN).

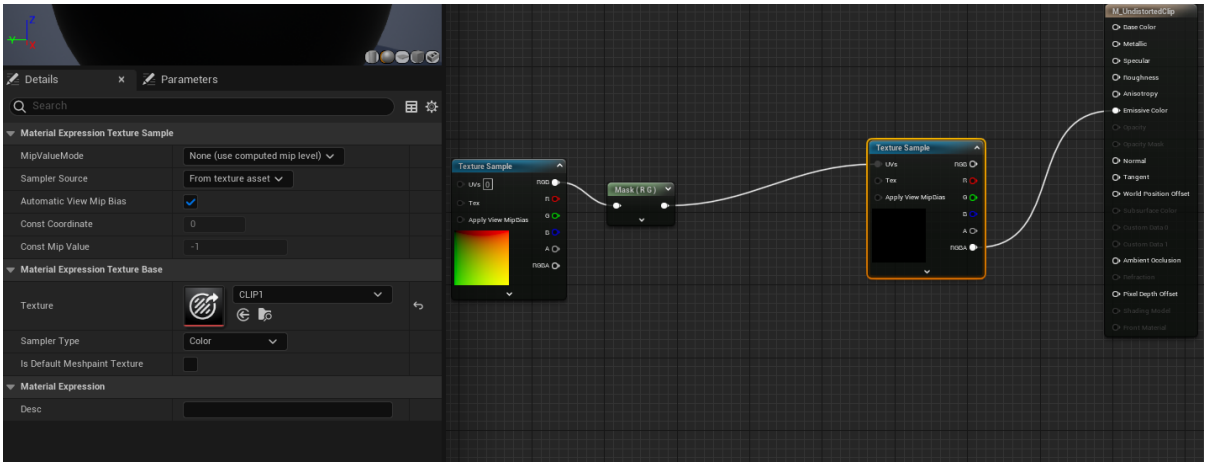


- Rename them to what CLIP/LIVE input number needed.

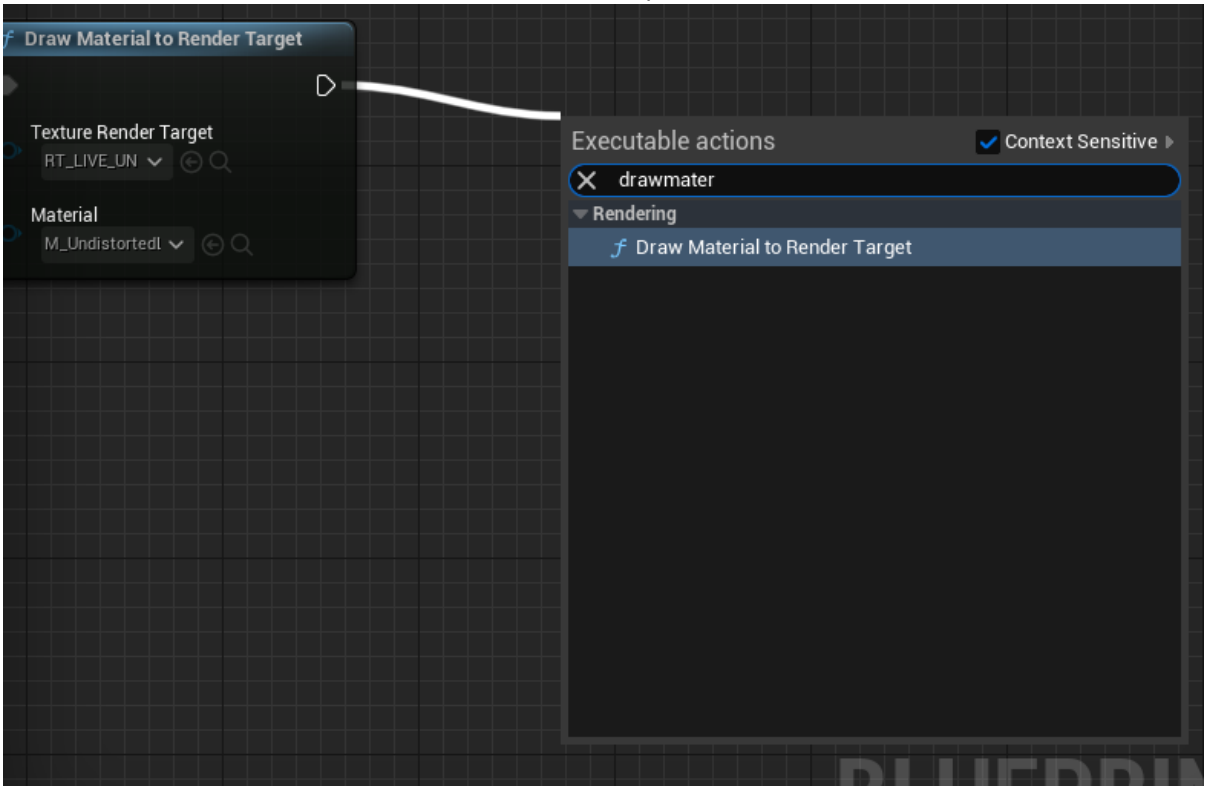
Copyright © 2026 Vizrt



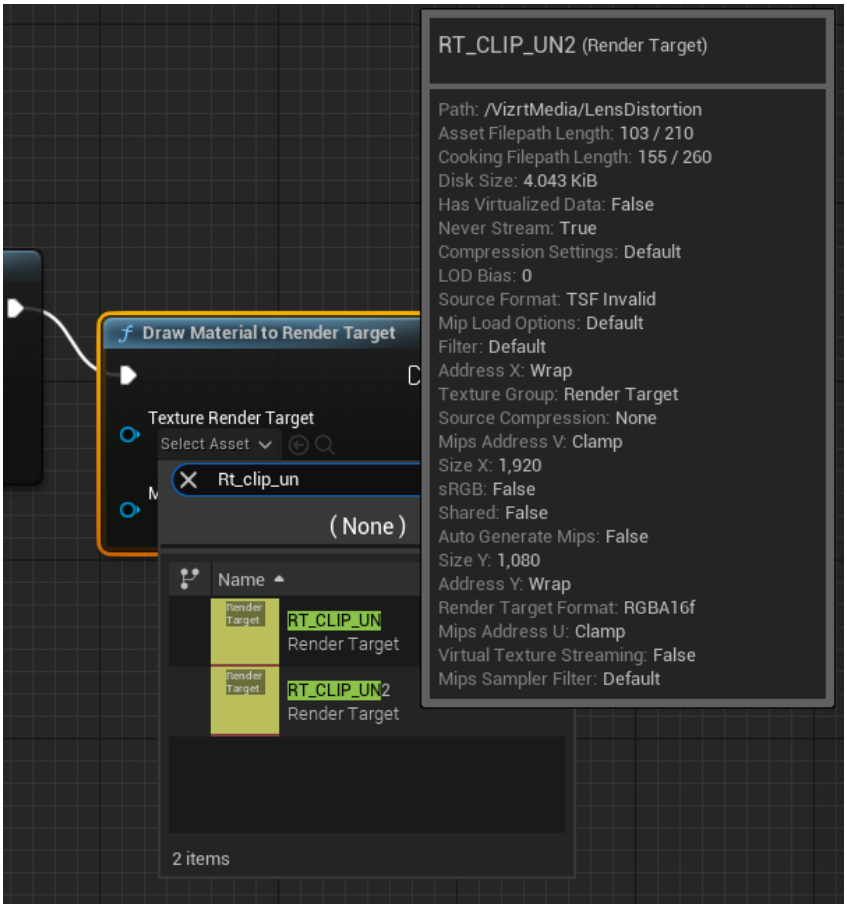
- Inside the material change the following node to CLIPX or LIVEX (where CLIP1 is in this example) and save the material:



- Inside the BP_MediaUndistortion added another DrawMaterialToRenderTarget node and add both CLIPX/LIVEX and RT_CLIP_UNX / RT_LIVE_UNX and save the blueprint.



Copyright © 2026 Vizrt



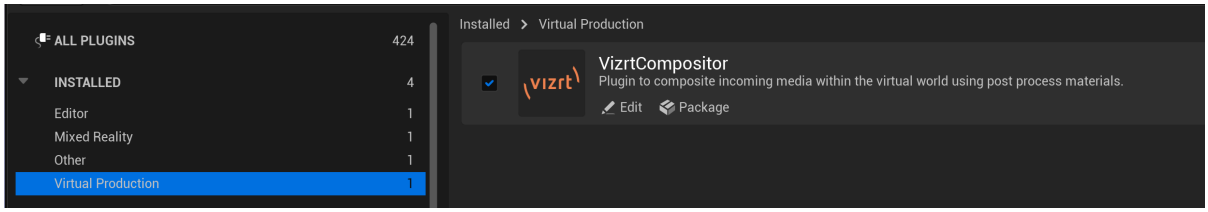
Copyright © 2026 Vizrt

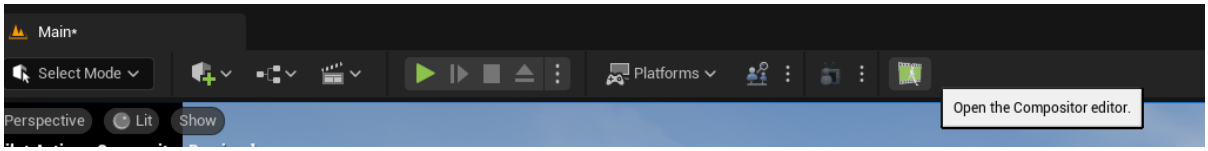


Copyright © 2026 Vizrt

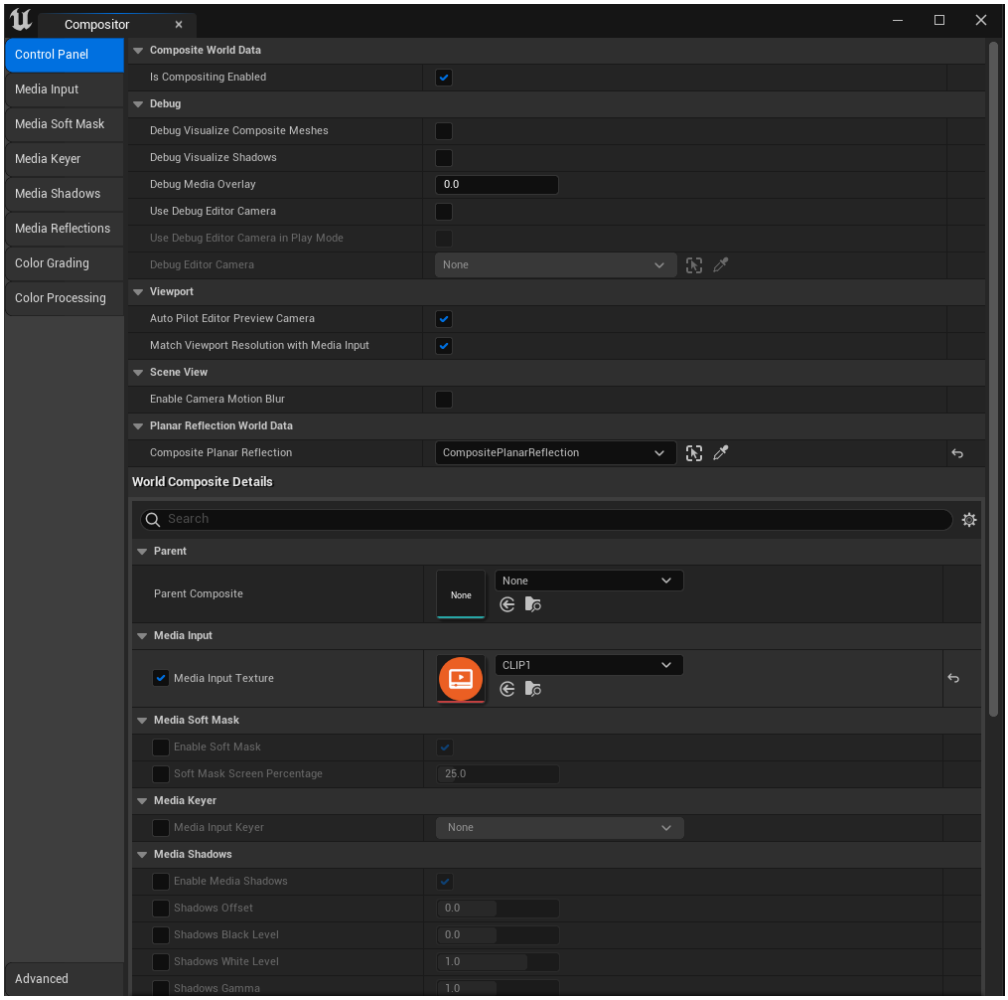
24.1.19 Compositing in Unreal Engine 5 (Compositor Workflow)

Activate the VizrtCompositor plug-in.





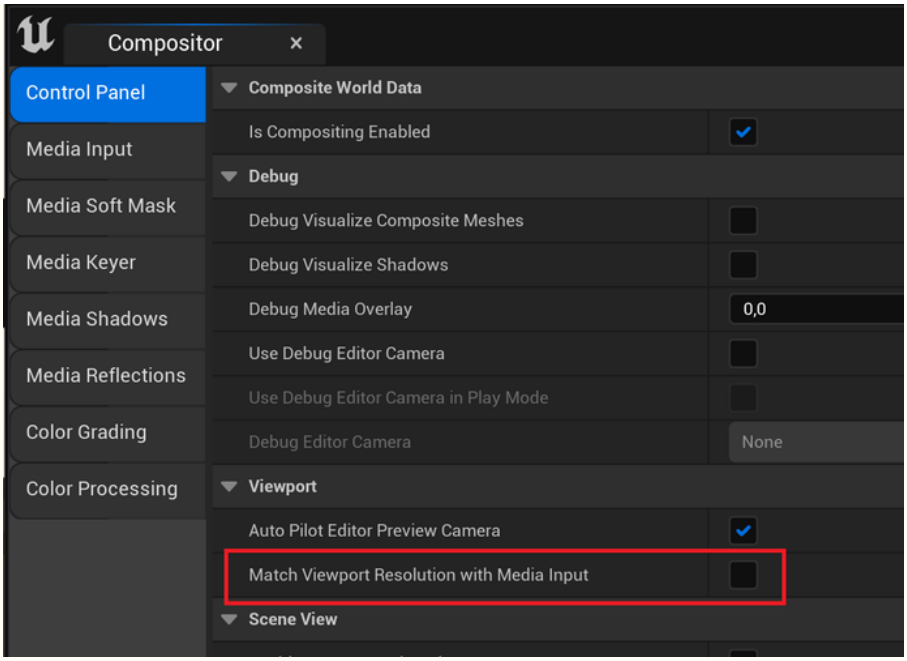
We need to add Compositor to the World and after that we see this window:



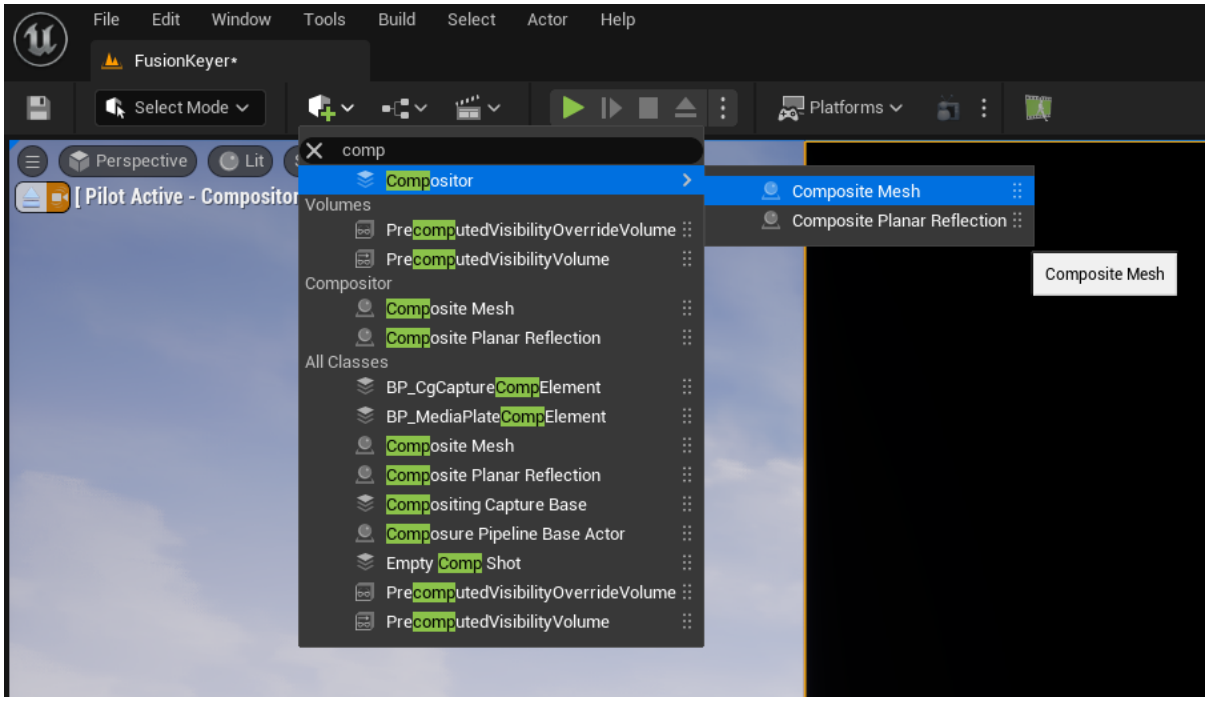
If we want a Clip coming from Viz as a background we should add the CLIP1 Texture in the Media Input as in the picture.

> ⚠ **Note:** If the Unreal output is being sent to Viz Engine through an **AUX** asset with the **Overscan** option set to true (used when Lens Distortion is active), the option **Match Viewport Resolution with Media Input** must be set to *False*.

Copyright © 2026 Vizrt

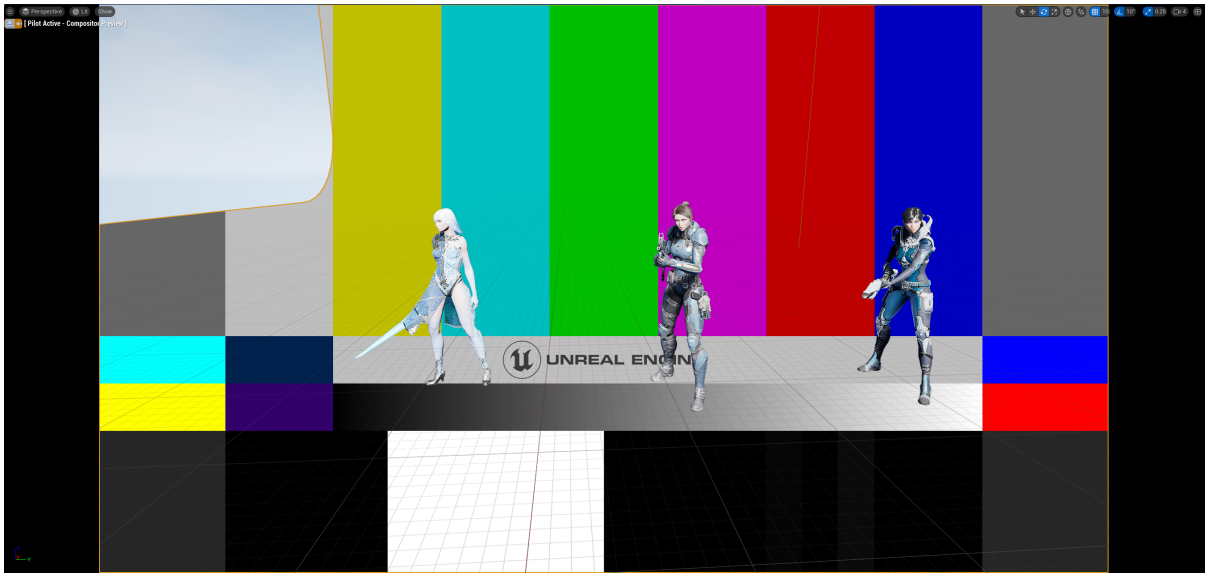


We need to drop the Composite Mesh into the level.



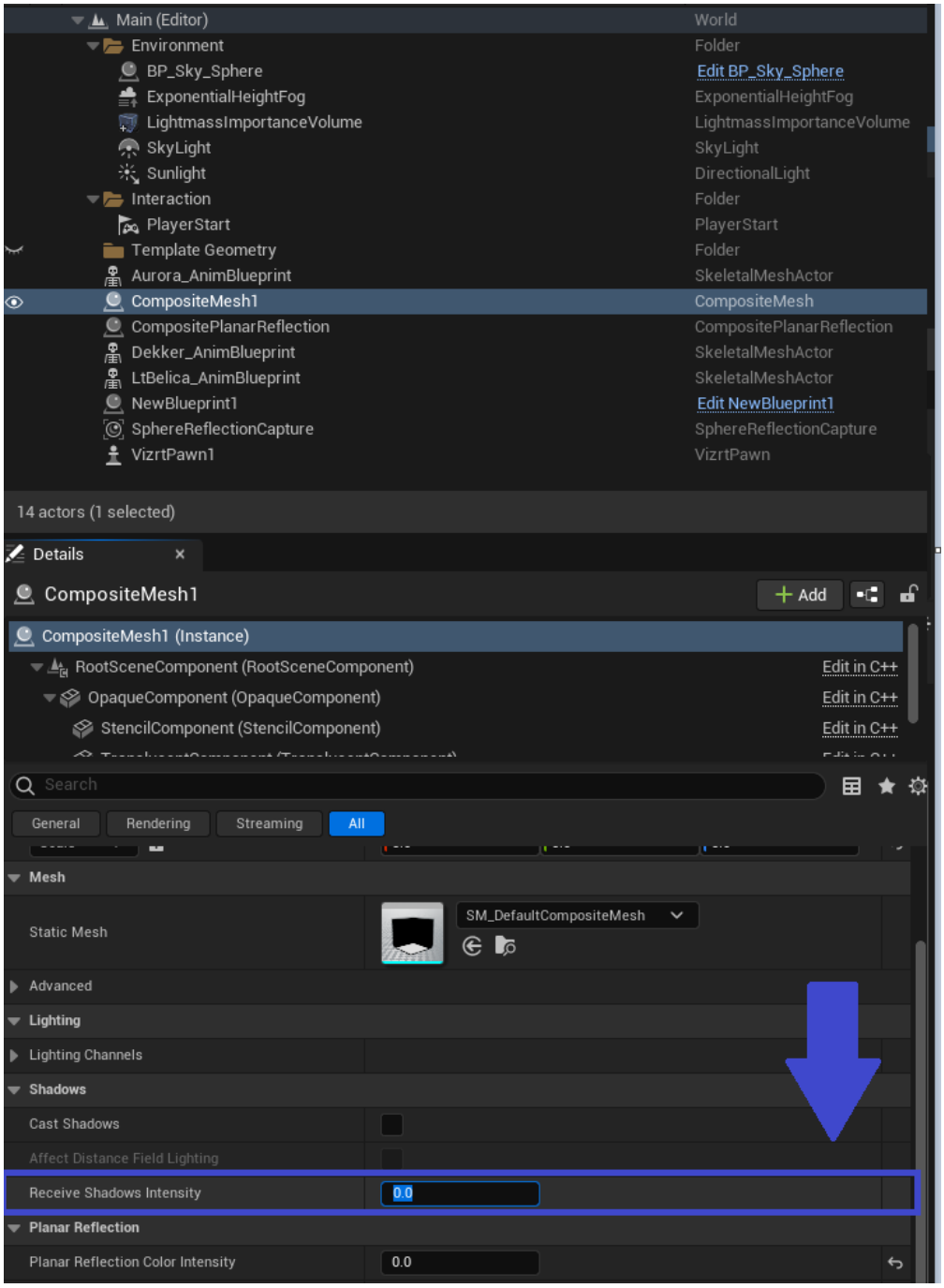
After dropping the Composite Mesh into the level we have the following:

Copyright © 2026 Vizrt



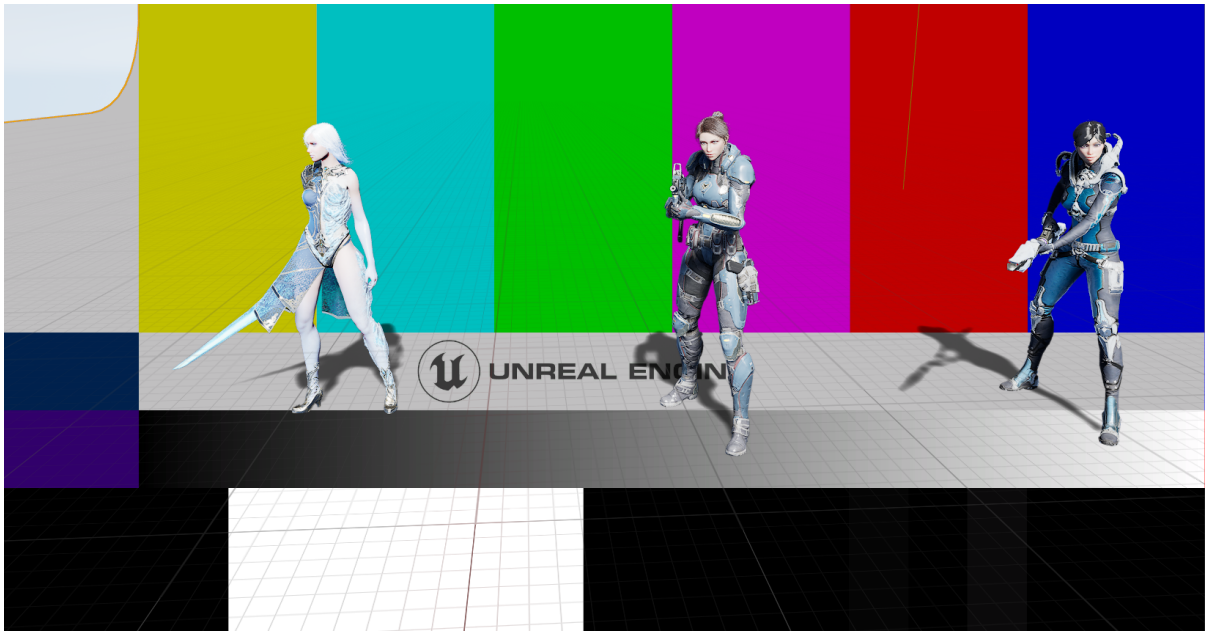
To add the shadows, we have to access the CompositeMesh's Details panel and change the field below to a value greater than `0`.

Copyright © 2026 Vizrt

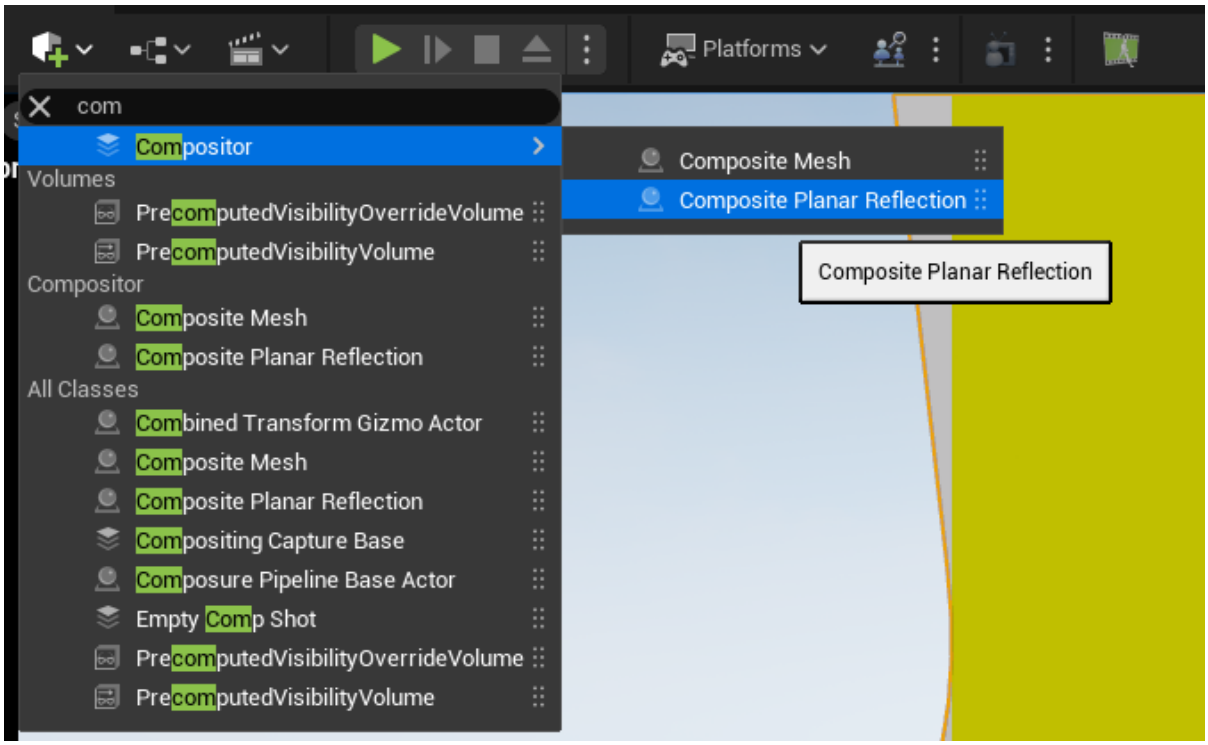


This gives the shadows from the example meshes in the composite mesh.

Copyright © 2026 Vizrt

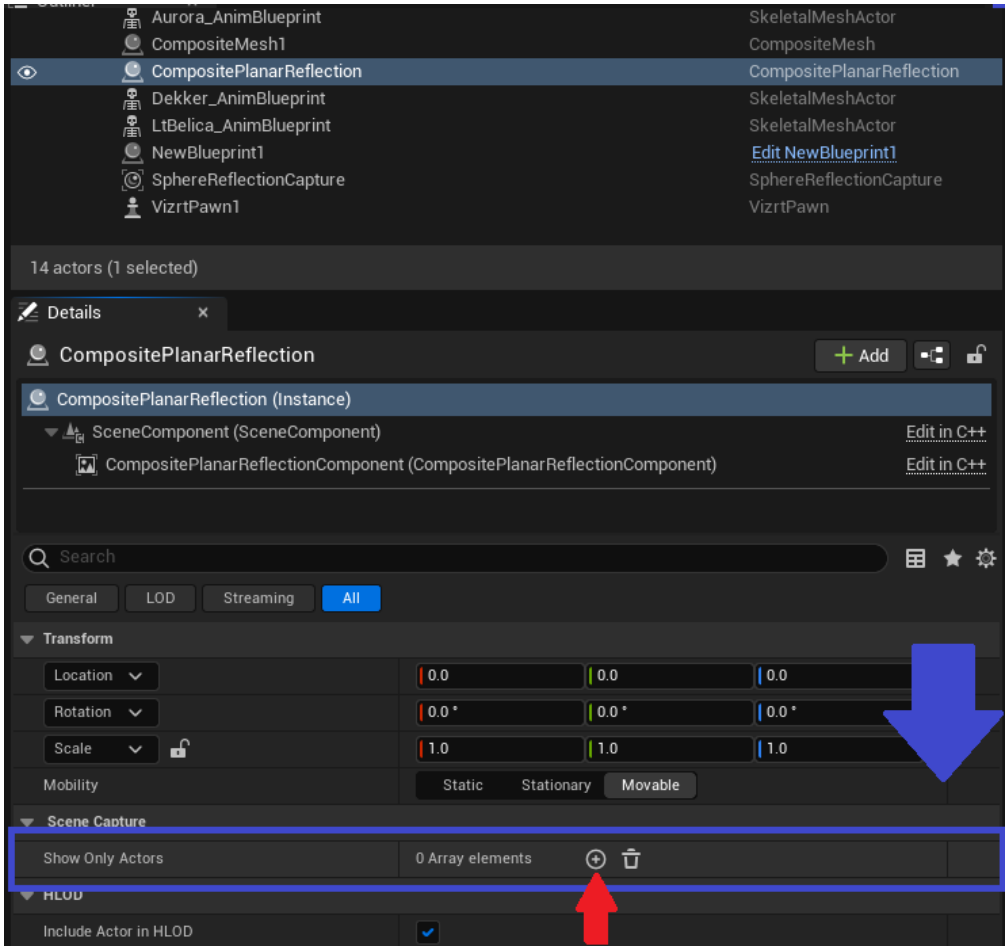


To add the reflections we need to add a Composite Planar Reflection to the level.

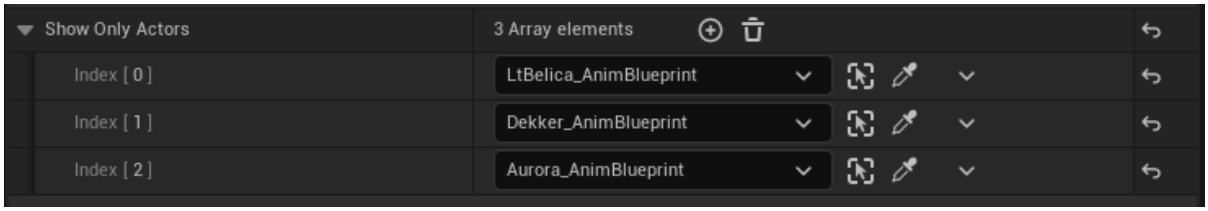


In the Composite Planar Reflection Details panel, we add our actors to the reflection capture:

Copyright © 2026 Vizrt

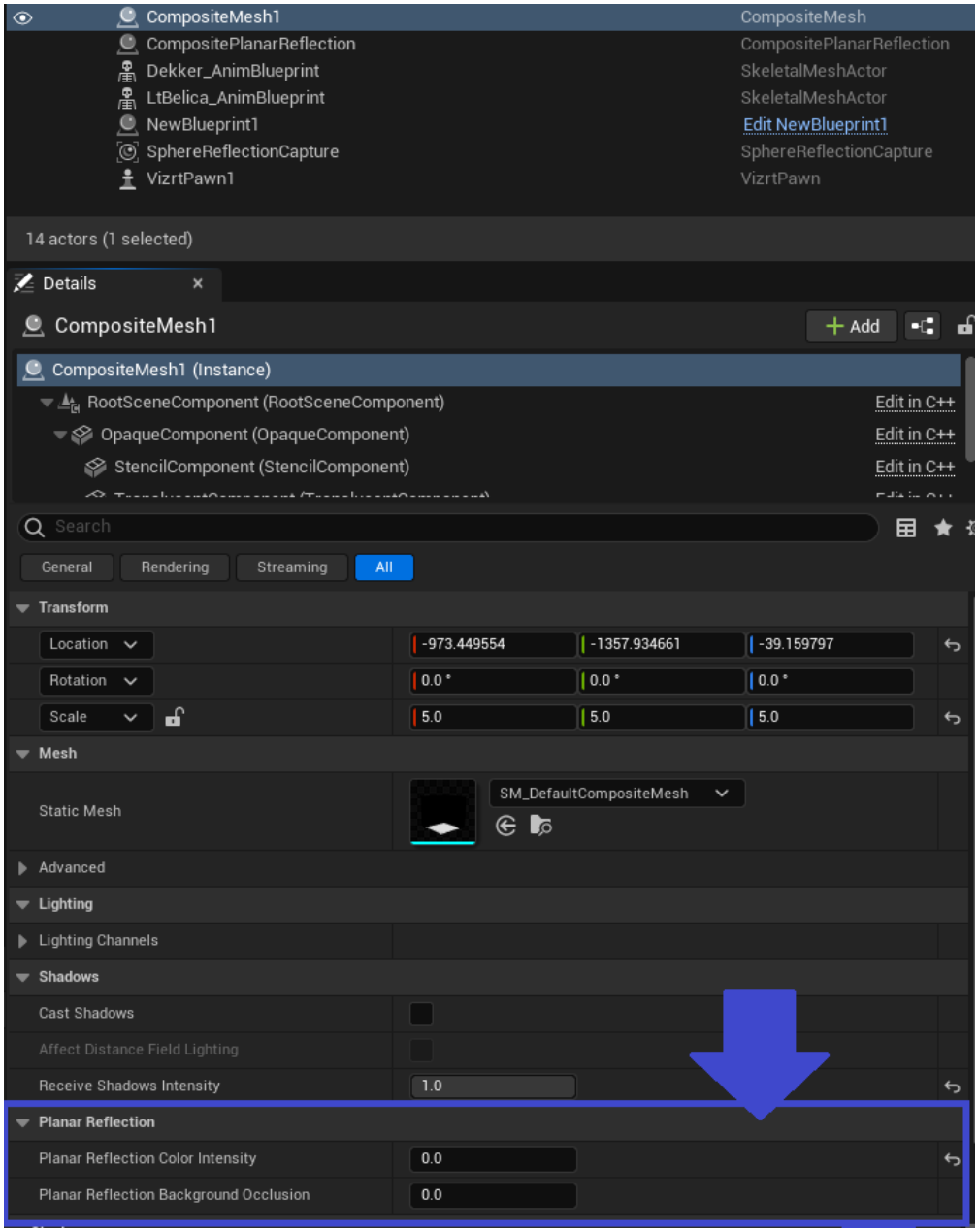


Something like this:

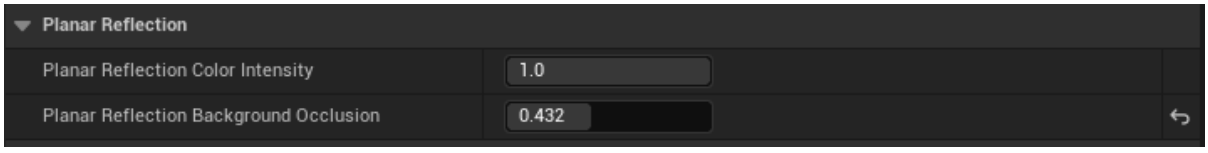


After adding all the actors we want to reflect to the Planar Reflection from the Compositor, we just need to go to the Composite Mesh and change the Planar Reflection fields:

Copyright © 2026 Vizrt

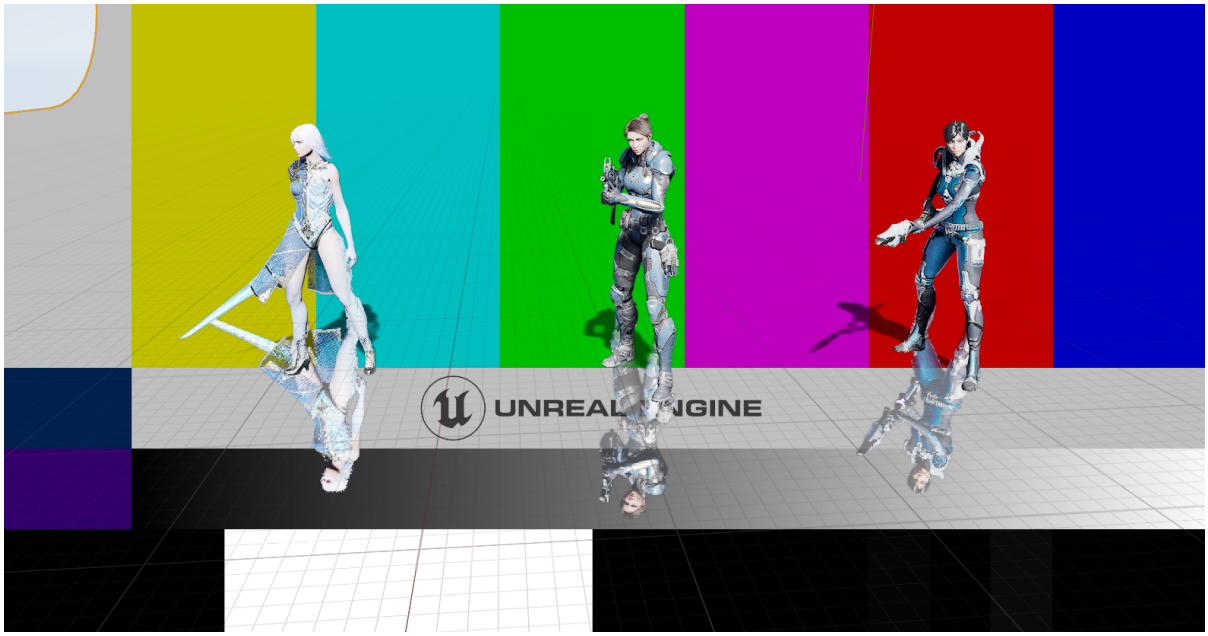


(different values give different results)

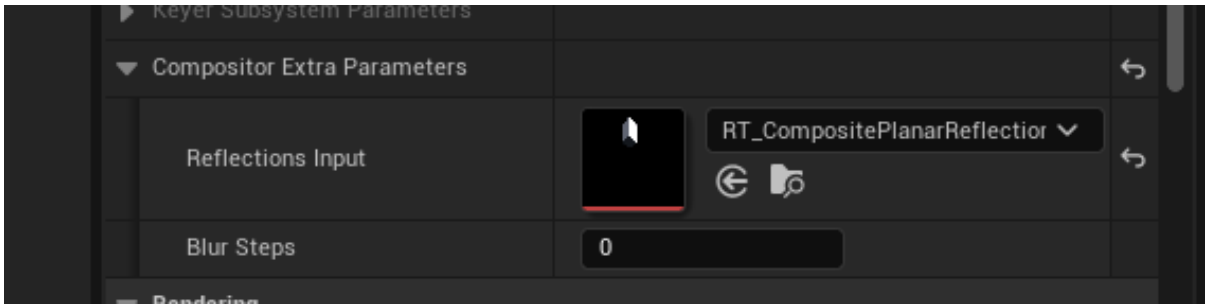


In the end of this process, we have this:

Copyright © 2026 Vizrt



With the added ability to blur the reflections, we now have to also do the following in VizrtPawn (even if you are not blurring the reflections):



Warning: If transparent objects are composited using the Vizrt Compositor, please make sure the **Anti-Aliasing Method** in **Project Settings > Engine - Rendering > Default Settings** is **NOT** set to Temporal Super-Resolution (TSR).



Note: To also have the center shifts affect the Reflections, the new Vizrt Compositor Connect blueprint should be added to the scene.

Copyright © 2026 Vizrt



In newer versions of Unreal Engine, support for VDB (Volumetric Breakdown) was added.

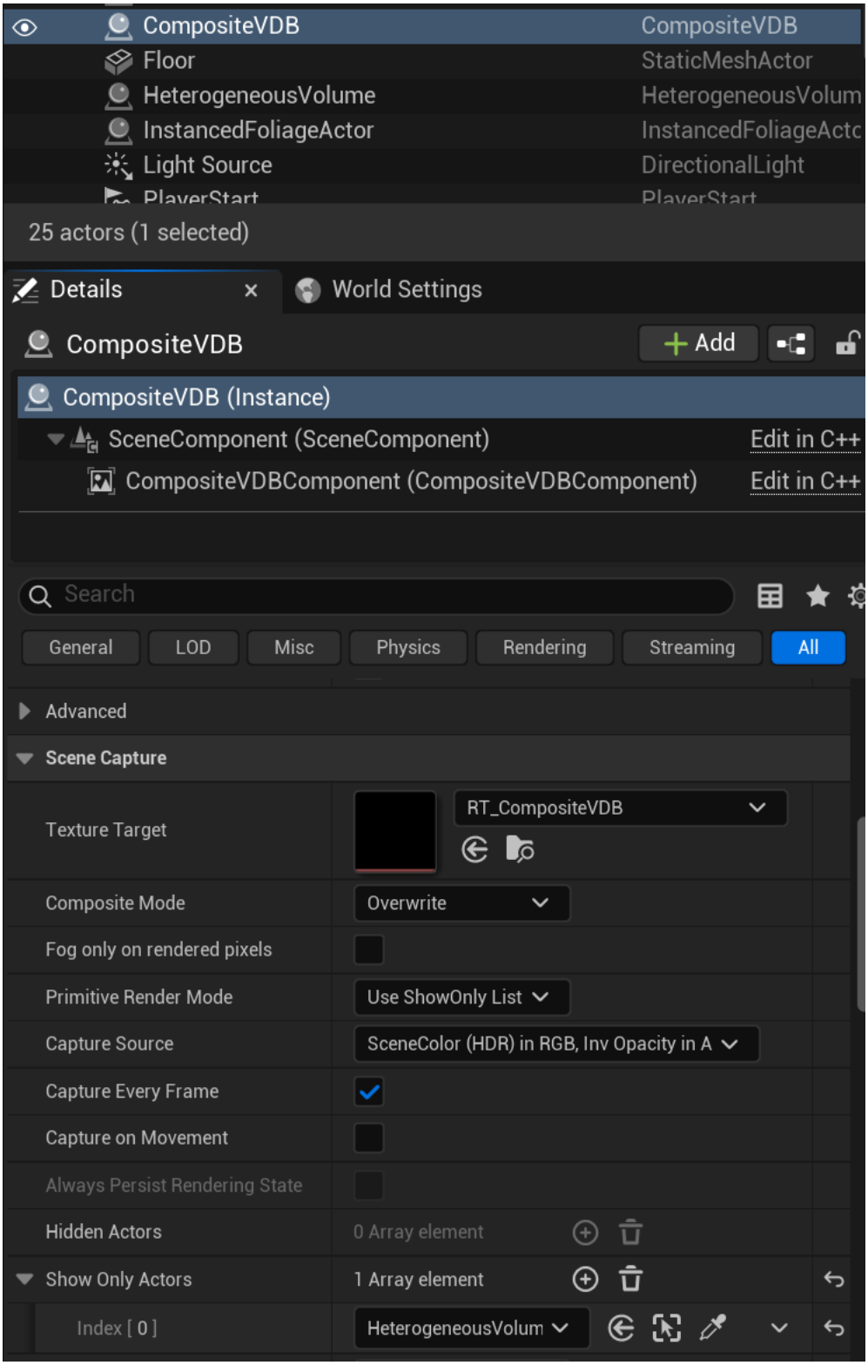


On sets that are using keying inside Unreal, VDB goes through normally, but if media is not keyed, some VDBs may not show up properly. To solve this, a new CompositeVDB actor was added, triggering a scene capture that allows the VDB to be cutout and shown in front of the talent.

To activate this:

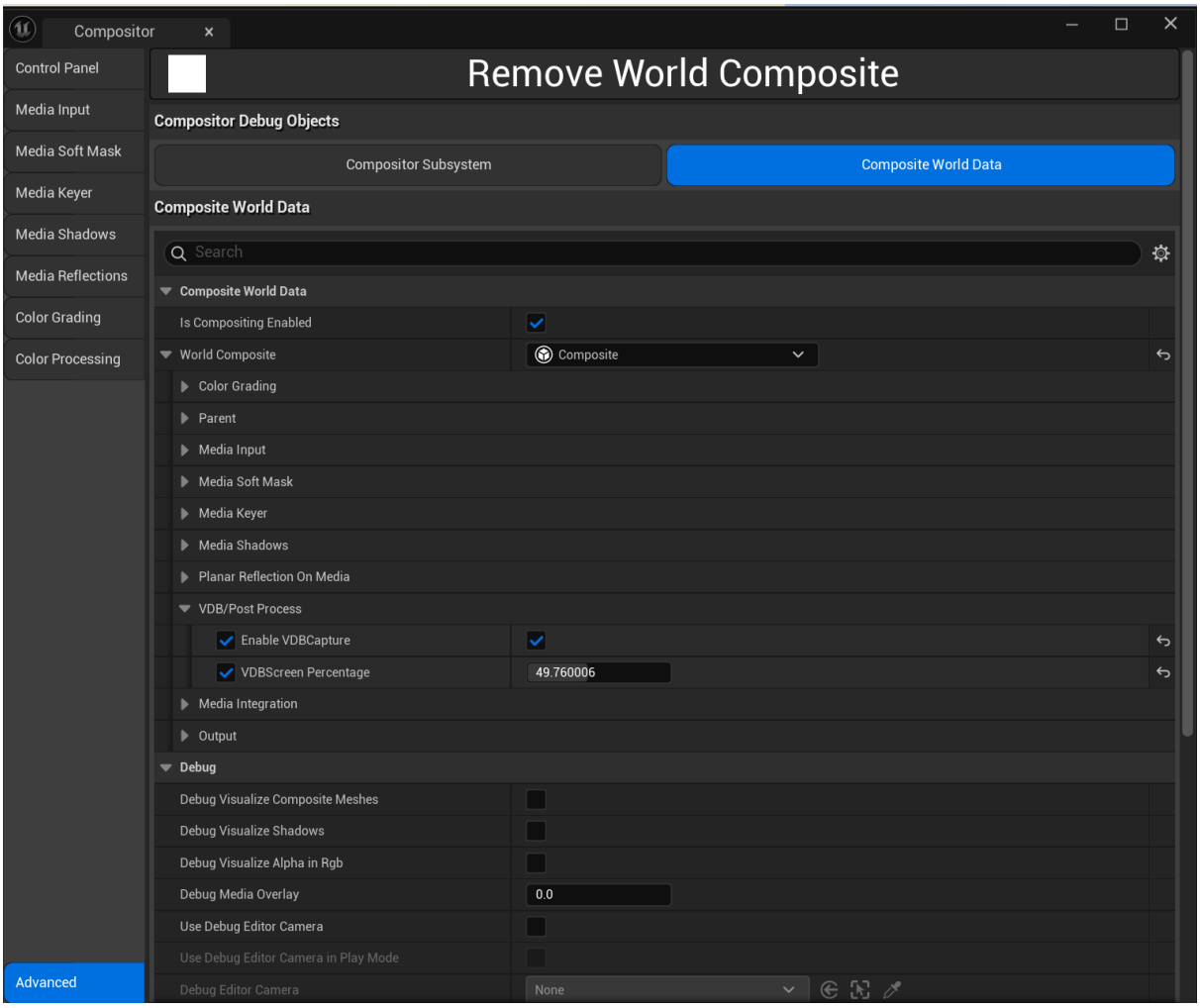
- Add the CompositeVDB to the scene.
- On the Scene Capture settings make sure to select **RT_CompositeVDB**, **Overwrite Composite Mode** and **Use ShowOnly List Render Mode**. Then add the actor with the heterogeneous volume to **Show Only Actors list**.

Copyright © 2026 Vizrt



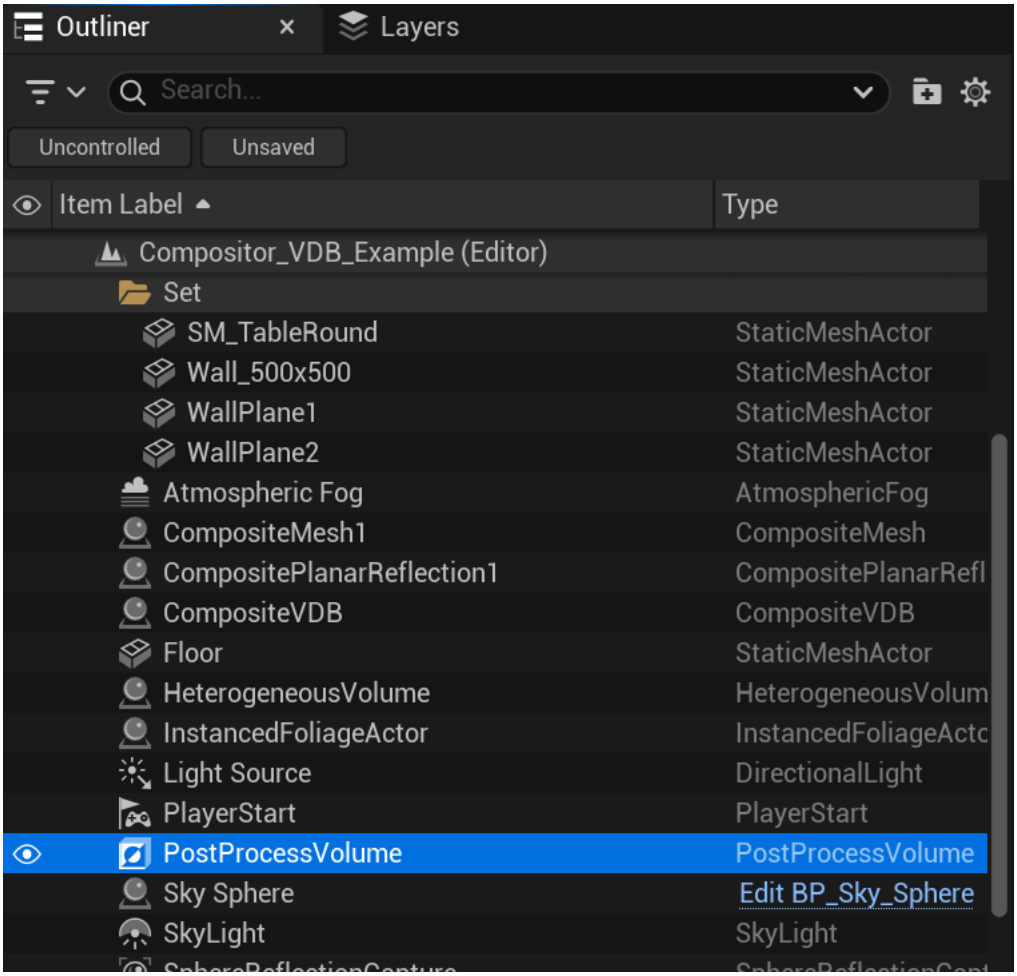
- Go to Compositor Editor in the advanced tab and click the **VDB/Post Process** section. Check mark **Enable VDBCapture** and type the screen percentage desired. The higher the value, the better the quality and the performance impact (50% is a good middle ground).

Copyright © 2026 Vizrt



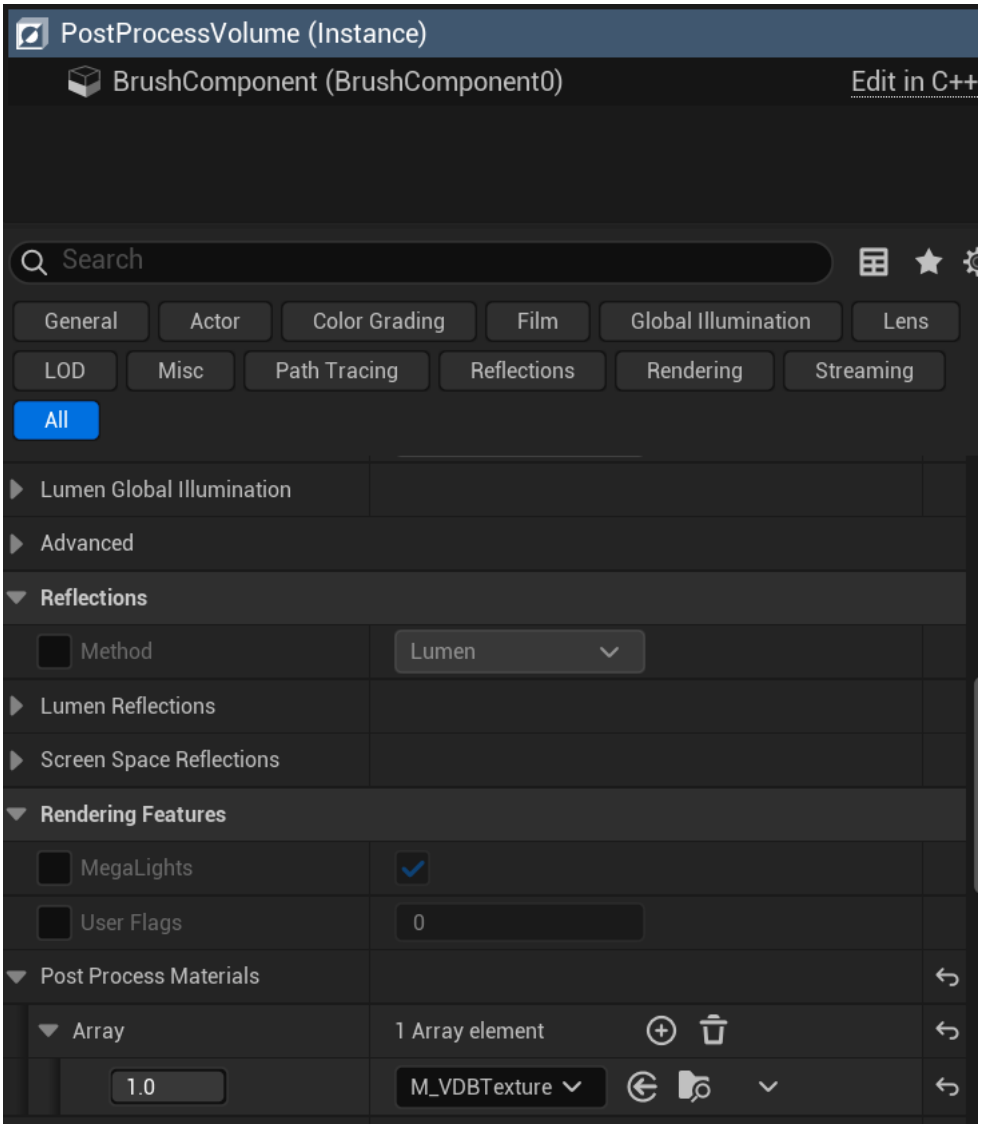
- To finalize, on the World Outliner, select **PostProcessVolume** (or add one).

Copyright © 2026 Vizrt

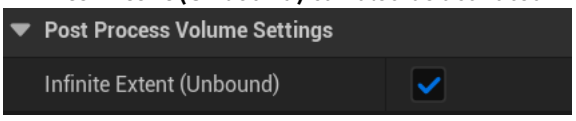


Make sure **M_VDBTexture** is on Post Process Materials.

Copyright © 2026 Vizrt



- If VDB is not going through, make sure **Post Process Volume** is large enough and covering the desired area. **Infinite Extent (Unbound)** can also be activated if needed.



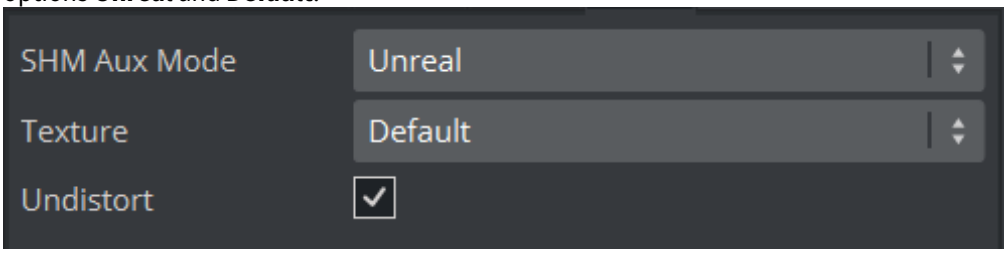
Copyright © 2026 Vizrt

24.1.20 Compositing with an External Keyer

Viz Engine version 5.3.0 introduces the possibility of using an external Keyer (Hardware, Viz-based, etc.) with the Compositor inside Unreal Engine.

Follow these steps for a quick and easy setup:

1. Inside Viz Artist, set up the externally keyed Clip or Live input, to be shared with Unreal Engine, with the options **Unreal** and **Default**.

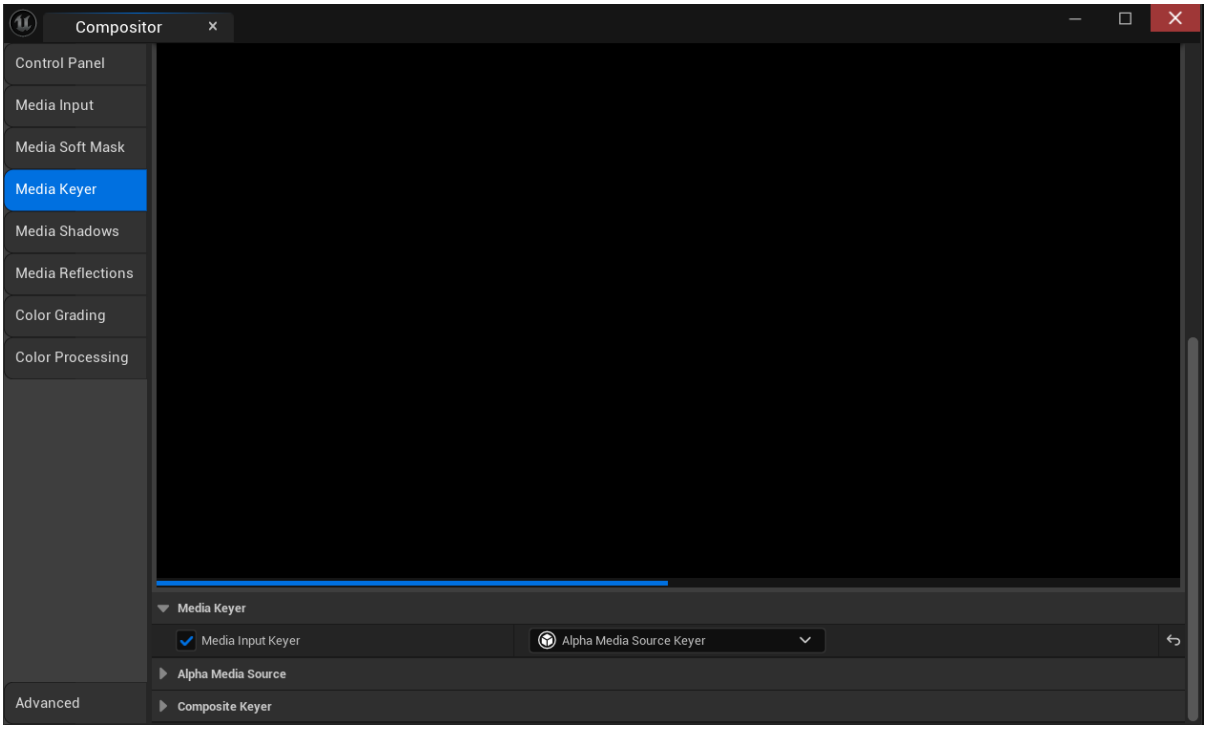


It should look like this:



2. Inside Unreal Engine, set up the Compositor (Precision Keyer Compositing on UE5, section *Using the Compositor Plug-in*) as normal and go to the **Compositor** settings to select the new **Alpha Media Source Keyer**.

Copyright © 2026 Vizrt



You can now grab any externally keyed texture, use it in the Compositor in Unreal and send it back to Viz.



Copyright © 2026 Vizrt



No other setting is required, as the Keyer is controlled externally.

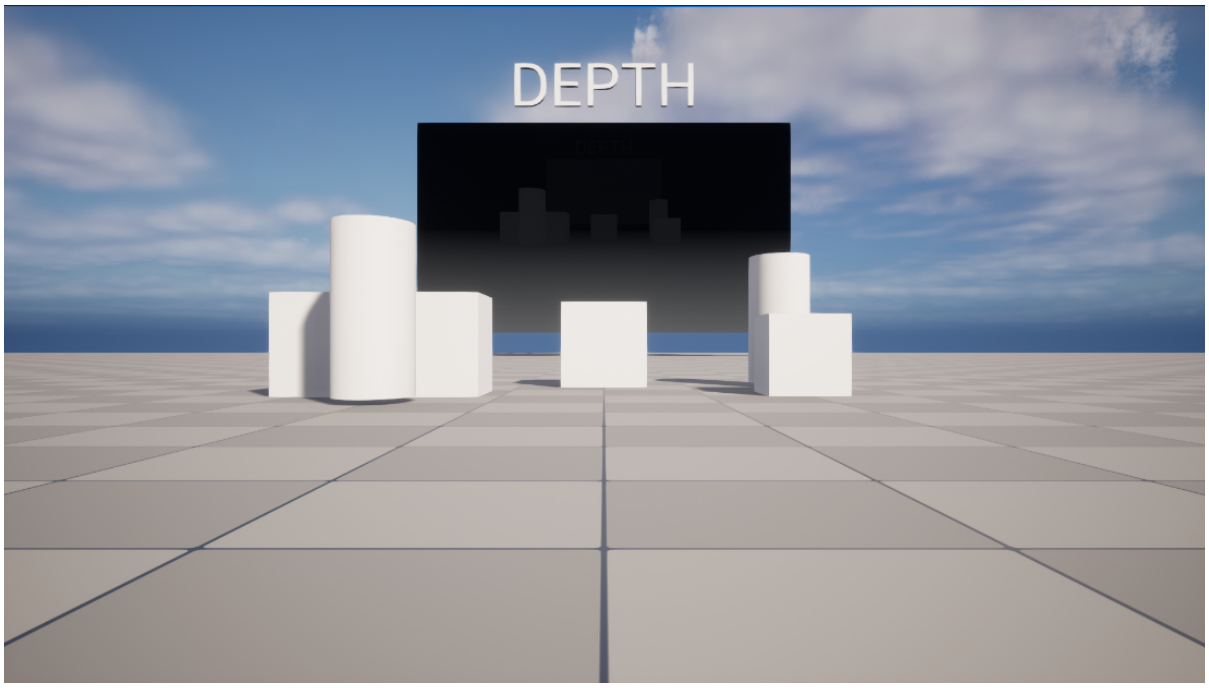
Copyright © 2026 Vizrt

24.1.21 Mixing Depth Buffers

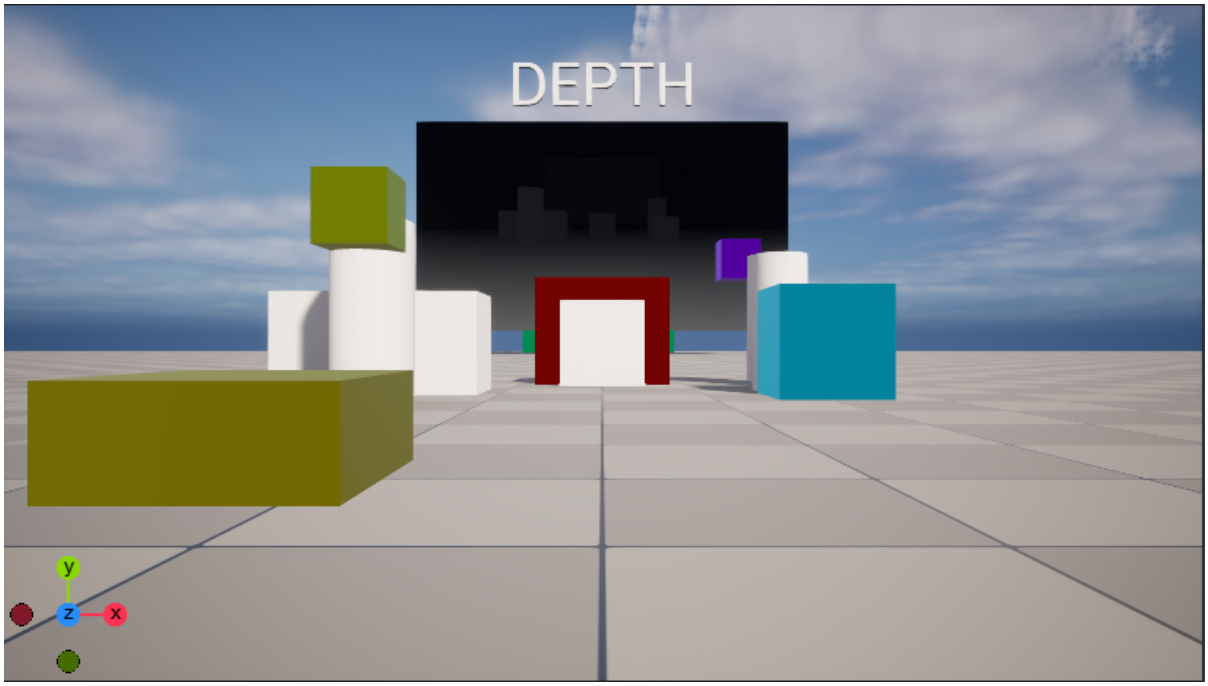
Warning: Do not use Temporal Anti-Aliasing methods (Temporal AA and Temporal Super Resolution) with this feature as it could lead to small artifacts on the final mixed image. We recommend using FXAA or MSAA to use this feature.

Starting with Viz Engine version 5.1, it is possible to send the Unreal depth information to Viz Engine even when the Unreal output is used as background.

This allows us to mix Unreal and Viz objects in more interesting and seamless ways. In the following example, first we have the Unreal viewport and secondly, the mixed result in Viz (where the Viz colored objects are both in front and in the back of the unreal white ones) using this approach.

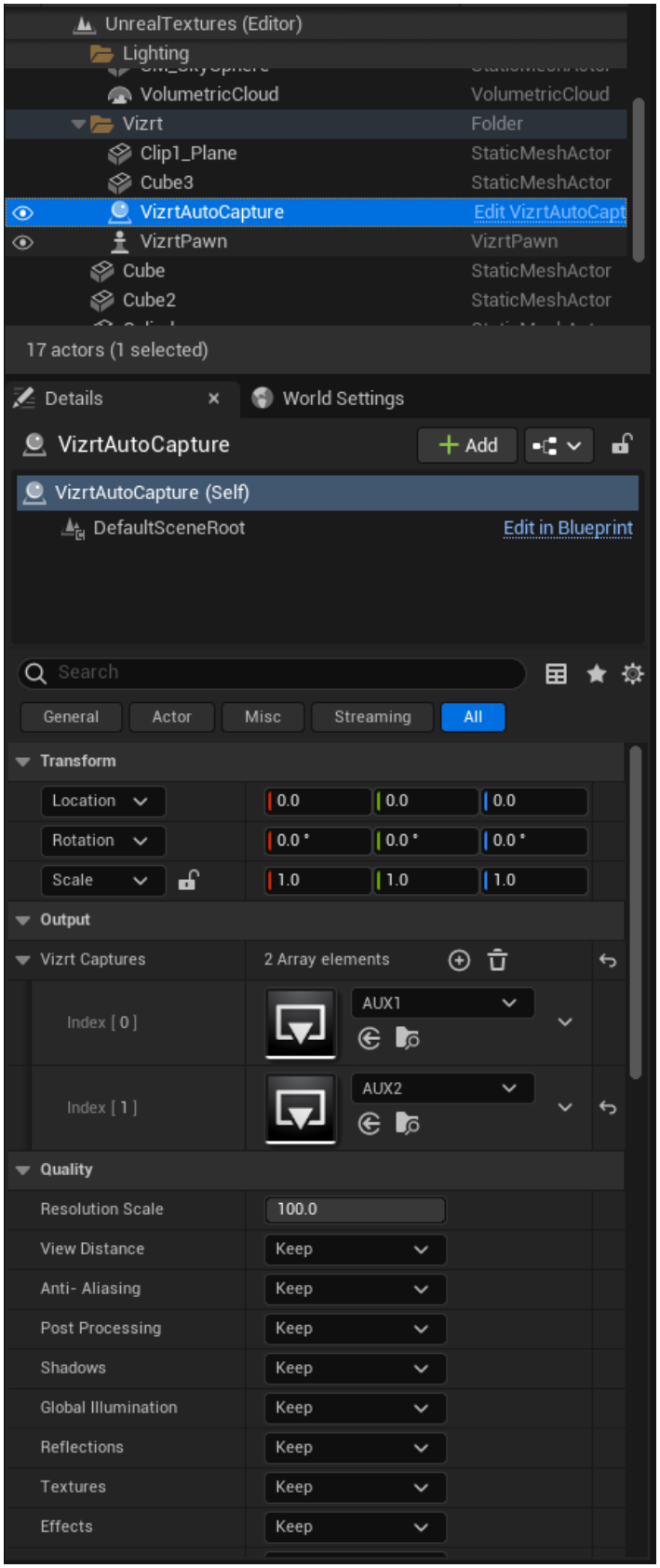


Copyright © 2026 Vizrt

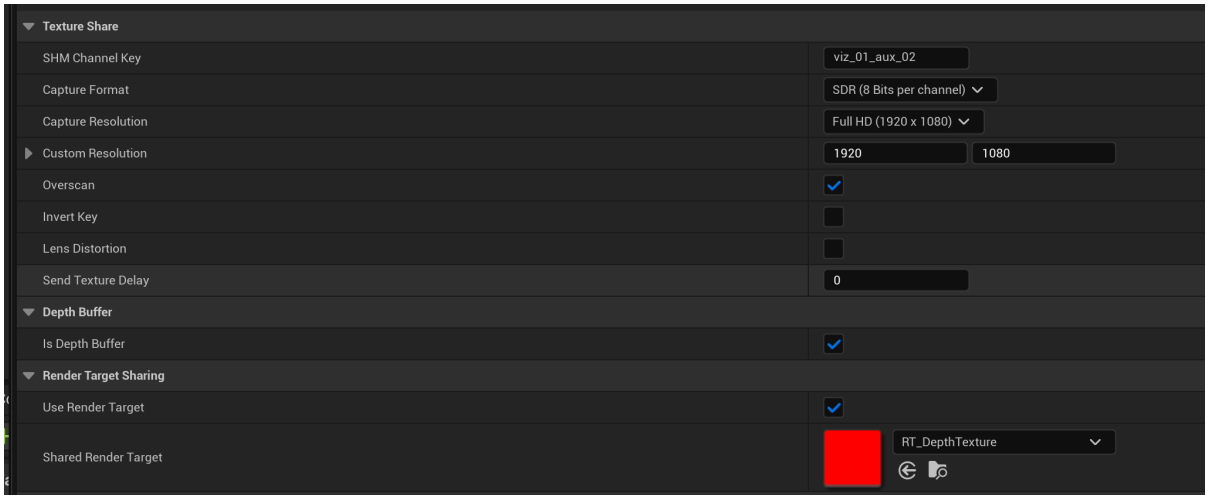


To activate the Depth sharing from Unreal, we set the AUX2 asset as a DepthTexture using the **Is Depth Buffer** checkbox (which should come as default with the plug-in). The AUX2 asset then needs to be added to the **VizrtAutoCapture** actor.

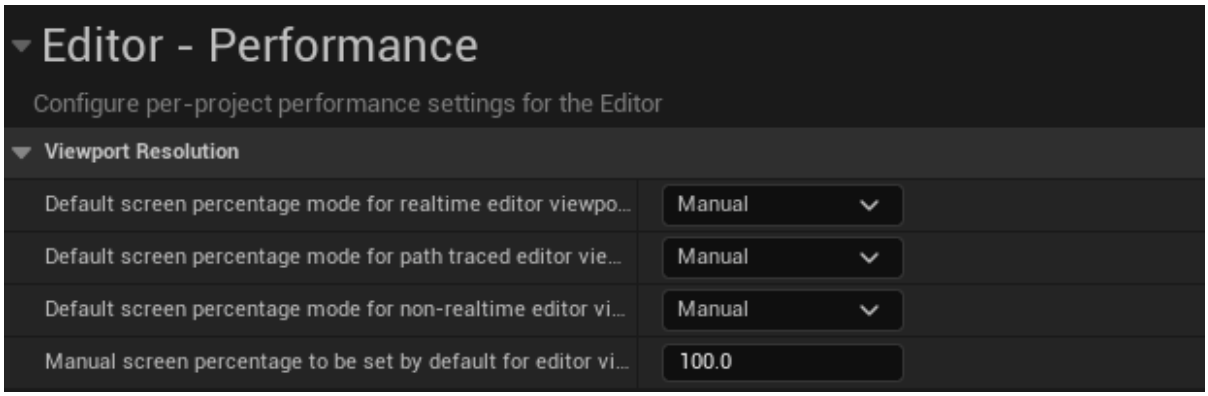
Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

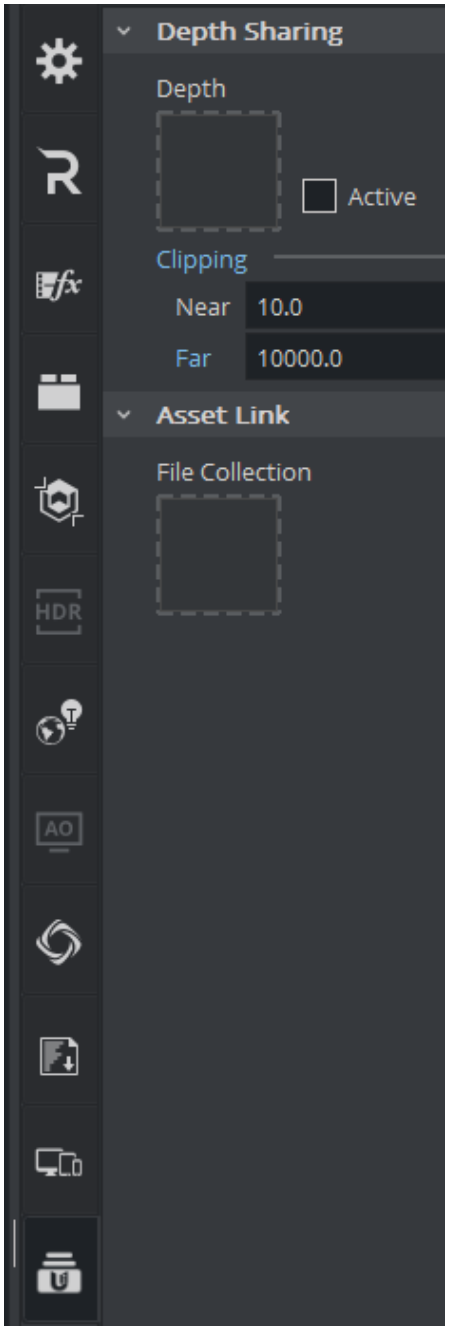


In the Project Settings, we need to have the default screen percentage mode as **Manual** and not based on the operating system's DPI scale.



In Viz Artist, we drag the AUX2 asset to the **Depth** slot.

Copyright © 2026 Vizrt



We can also use the Clipping values (**Near** and **Far**) to control the drawing parameters.

Copyright © 2026 Vizrt

24.1.22 Precision Keyer Compositing on UE5

The Viz Precision Keyer is now available as an Unreal Integration with the Unreal Engine 5. This is a quick guide to help set up the Keyer in the Unreal Engine with the Vizrt Compositor plug-in.

Warning: Do not use Temporal Anti-Aliasing methods (Temporal AA and Temporal Super Resolution) with the Compositor and Keyer as this leads to possible artifacts on the image (similar to key and fill being out of sync). On the project settings set the Anti-Aliasing to *FXAA* or *MSAA*.

Precision Keyer Basics

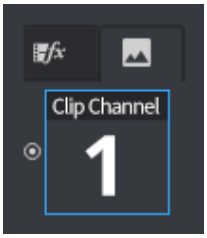
Sending the Clip and Matte Scene

To send the Clip scene in Viz Engine and have it playing in Unreal:

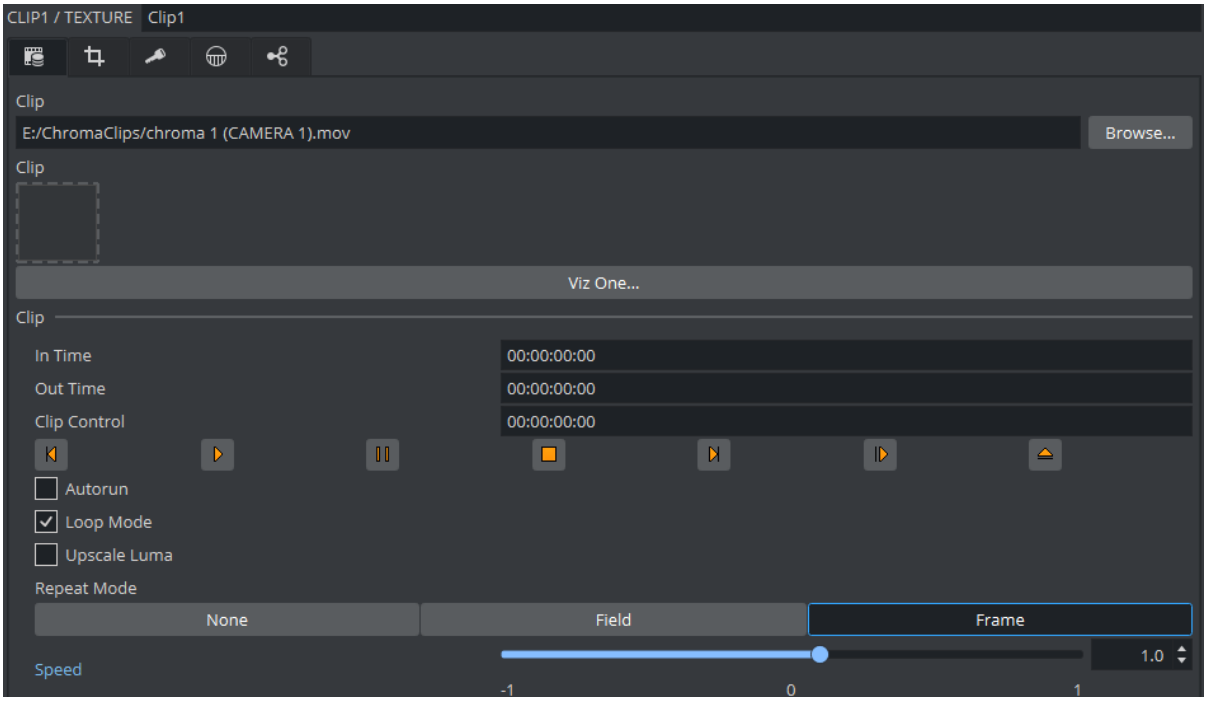
- Grab CLIP1 from the plug-ins.



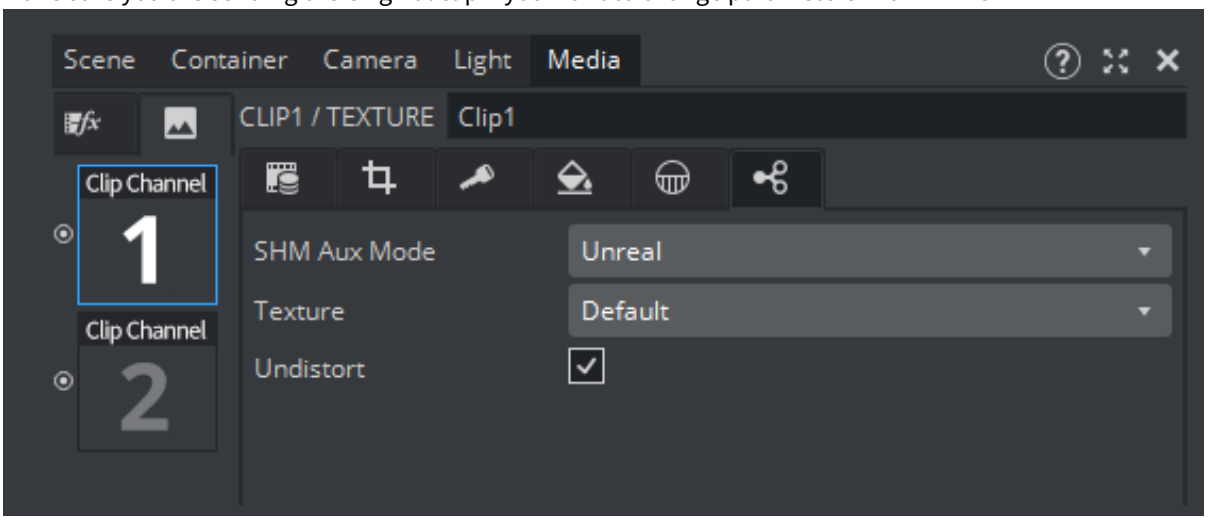
- Add to the Media tab.



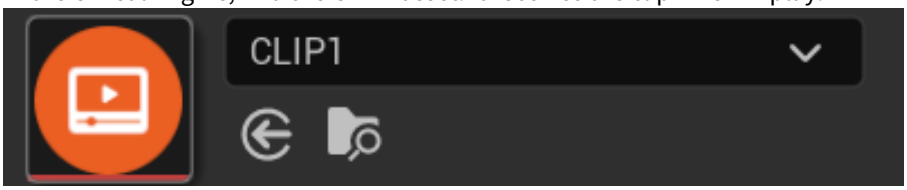
- Add a clip.



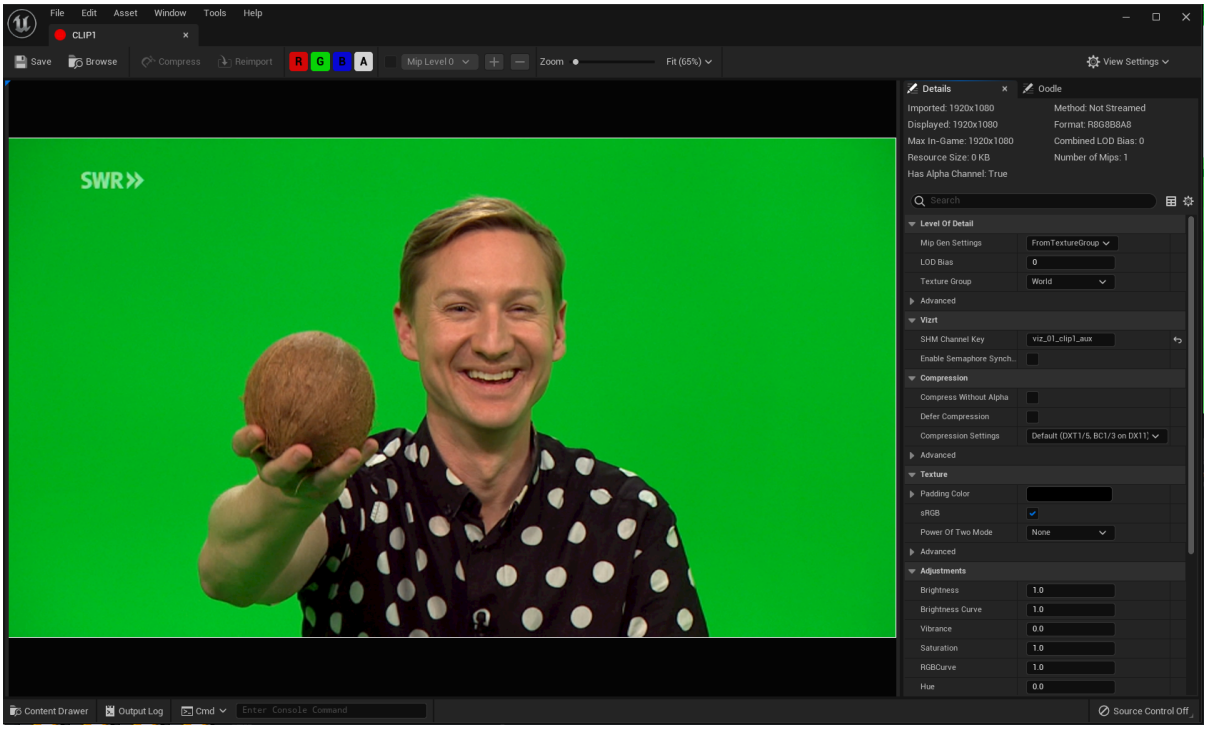
- Make sure you are sending the Original clip if you want to change parameters with Viz Arc



- In the Unreal Engine, find the CLIP1 asset. It receives the clip when in play.

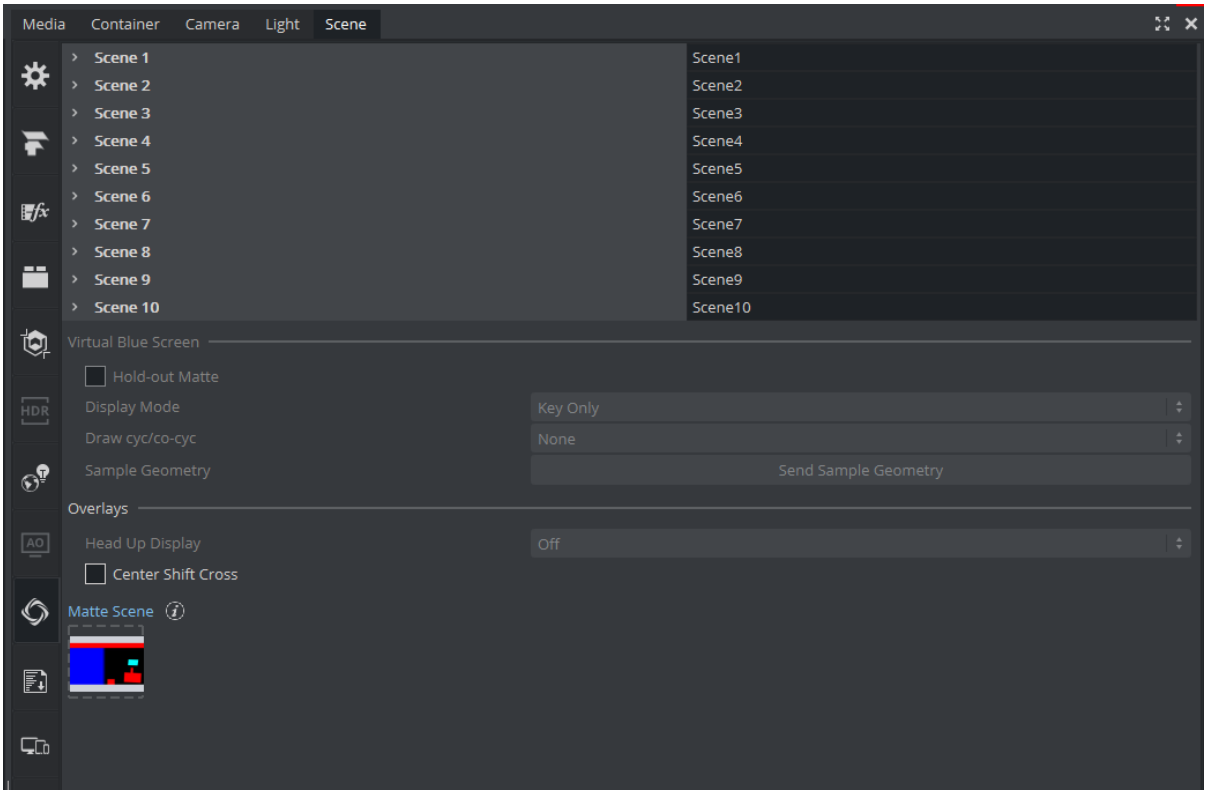


Copyright © 2026 Vizrt



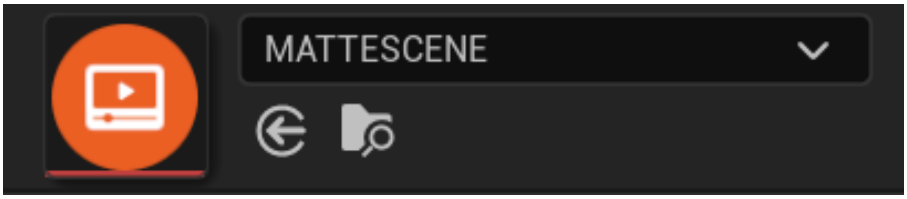
To send the Clip scene in Viz Engine and receive it in Unreal:

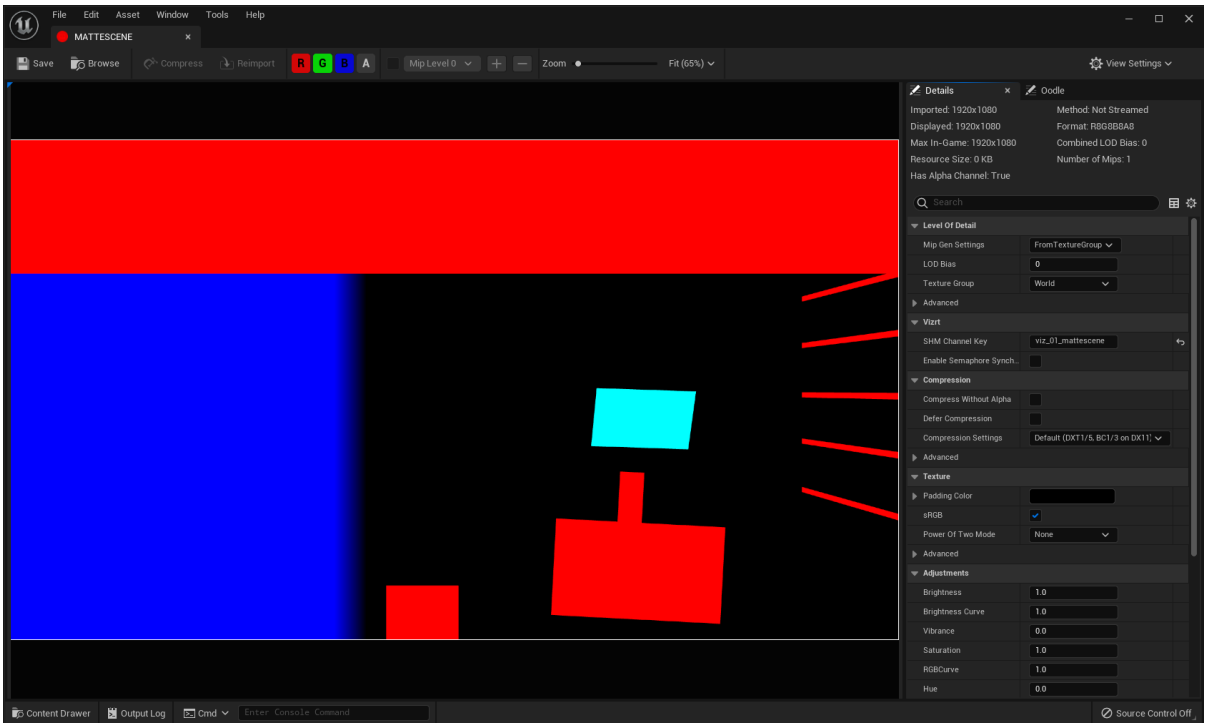
- Add a Matte Scene in Viz.



- In the Unreal Engine find the MATTESCENE asset, it has the matte scene when in play.

Copyright © 2026 Vizrt





The process to send a LIVE input is the same as with the clip, but using LIVE1 as the asset.

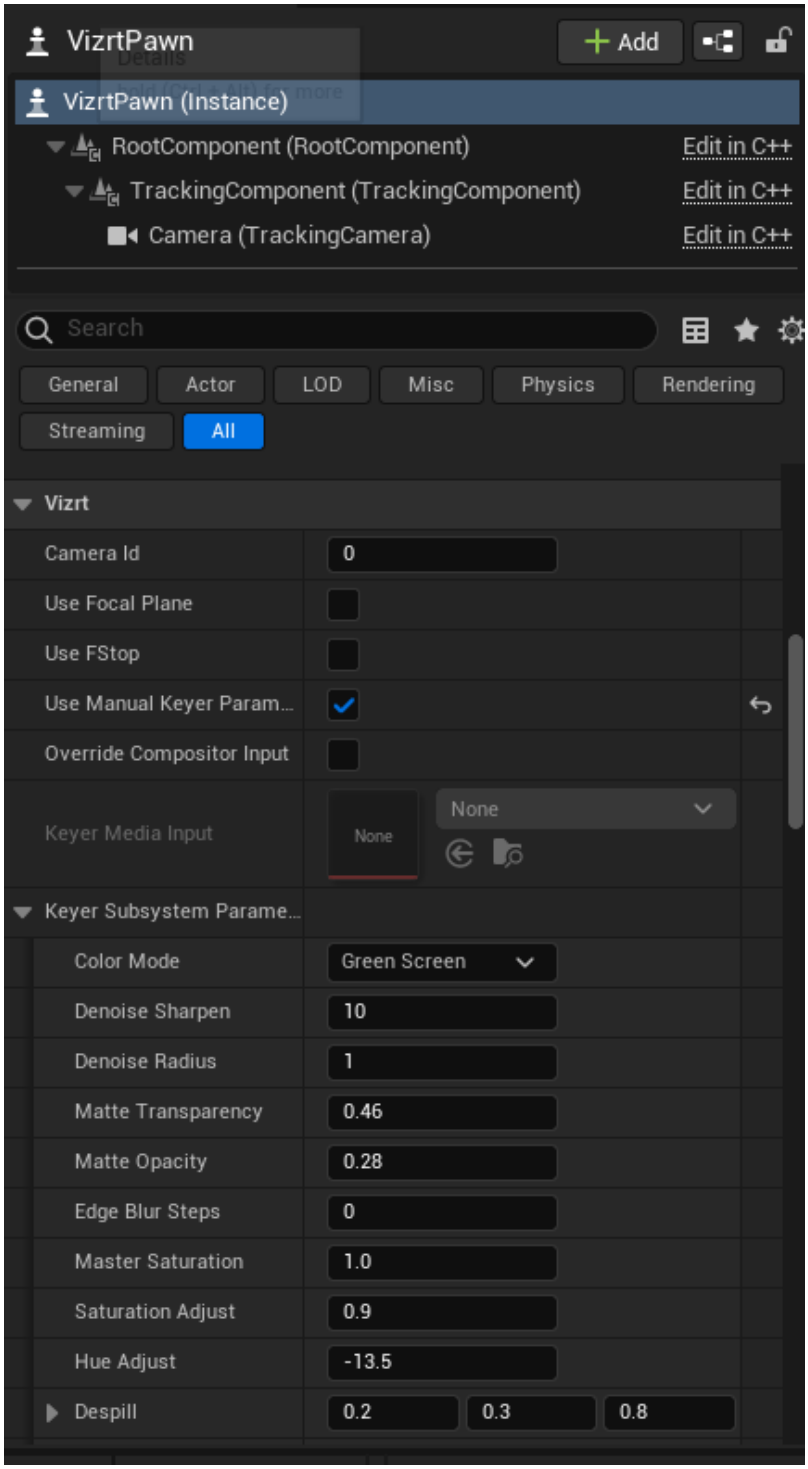
Setting up the Keyer

Unlike previous versions, the Precision Keyer is rendered via a subsystem. The Media Input follows the Compositor (when available) media input so that should happen automatically when using the Compositor plug-in. This simplifies the setup when we are also using the Compositor plug-in.

The UE Precision Keyer can now be synced with Viz Engine and Viz Arc. As soon as the parameters change on Viz Arc, they are synced with Unreal Engine. Some small tweaking can be done via Viz Arc.

If for some reason we want to have Unreal overriding the parameters that come from Viz, the Keyer settings are now on a separated tab in the VizrtPawn object. To do so we need to tick *Use Manual Keyer Parameters*.

Copyright © 2026 Vizrt



If we want to override the Compositor media input (or if we do not have a Compositor in the level) we need to tick *Override Compositor Input*.

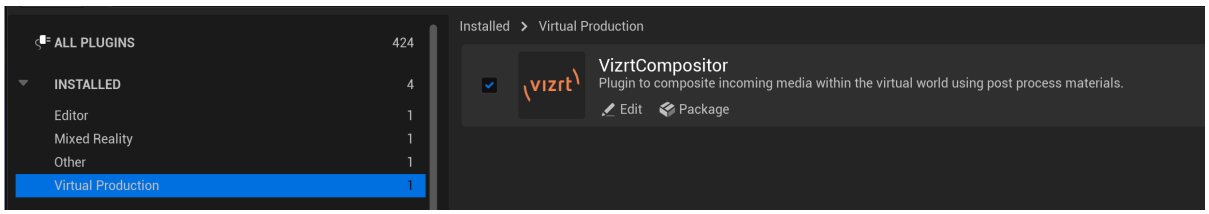
Copyright © 2026 Vizrt



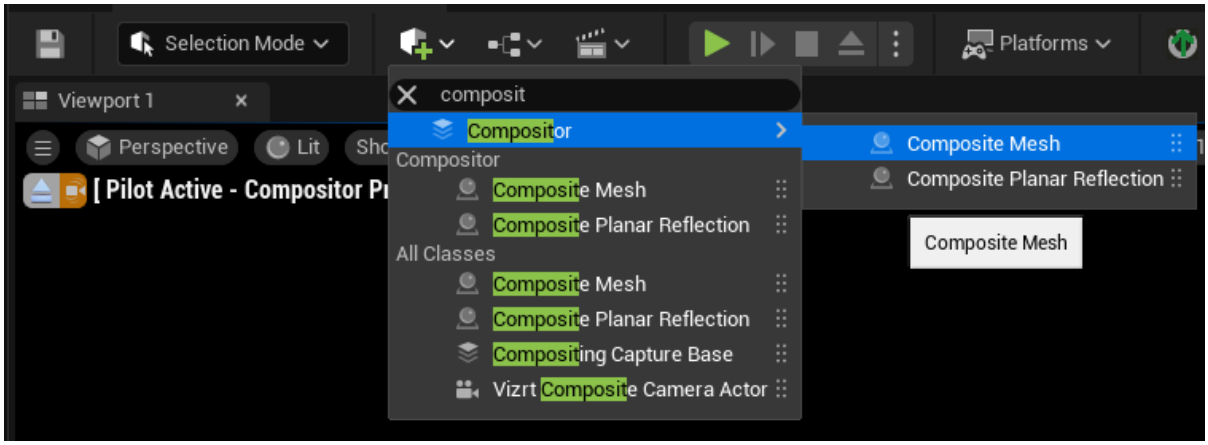
Using the Compositor Plug-in

Setting up the Composite Mesh

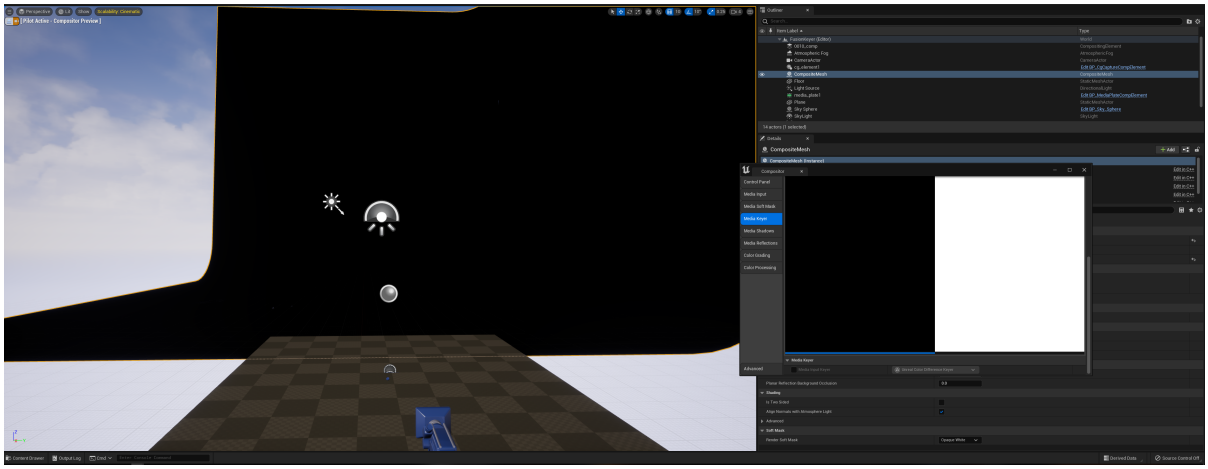
If the plug-in is not activated by default we need to activate it.



We need to drop the Composite Mesh into the level.



After dropping the Composite Mesh into the level we have the following:



Note: When using more than one Composite Mesh, make sure that the *Render CustomDepth Pass* toggle is active on both OpaqueComponent and StencilComponent for each Composite Mesh. Not having active on all the meshes could lead to color artifacts and differences between the meshes.



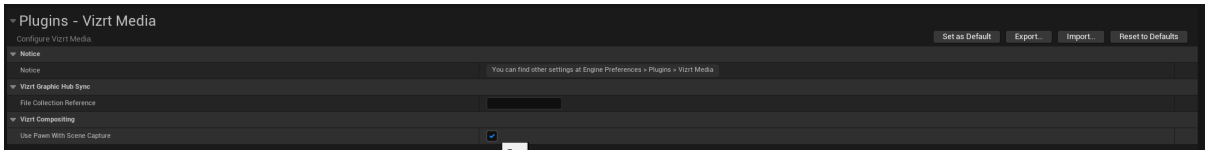
Setting up the Composite Planar Reflections

For the Shadows and Reflections, refer to Compositing in UE5.

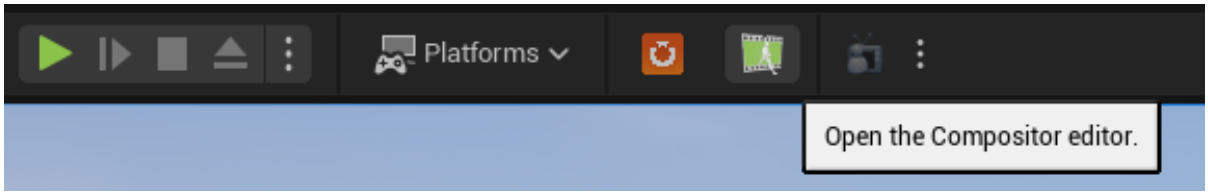
Setting up the Media Input and Keyer

Make sure you have the following activated on your Project Settings.

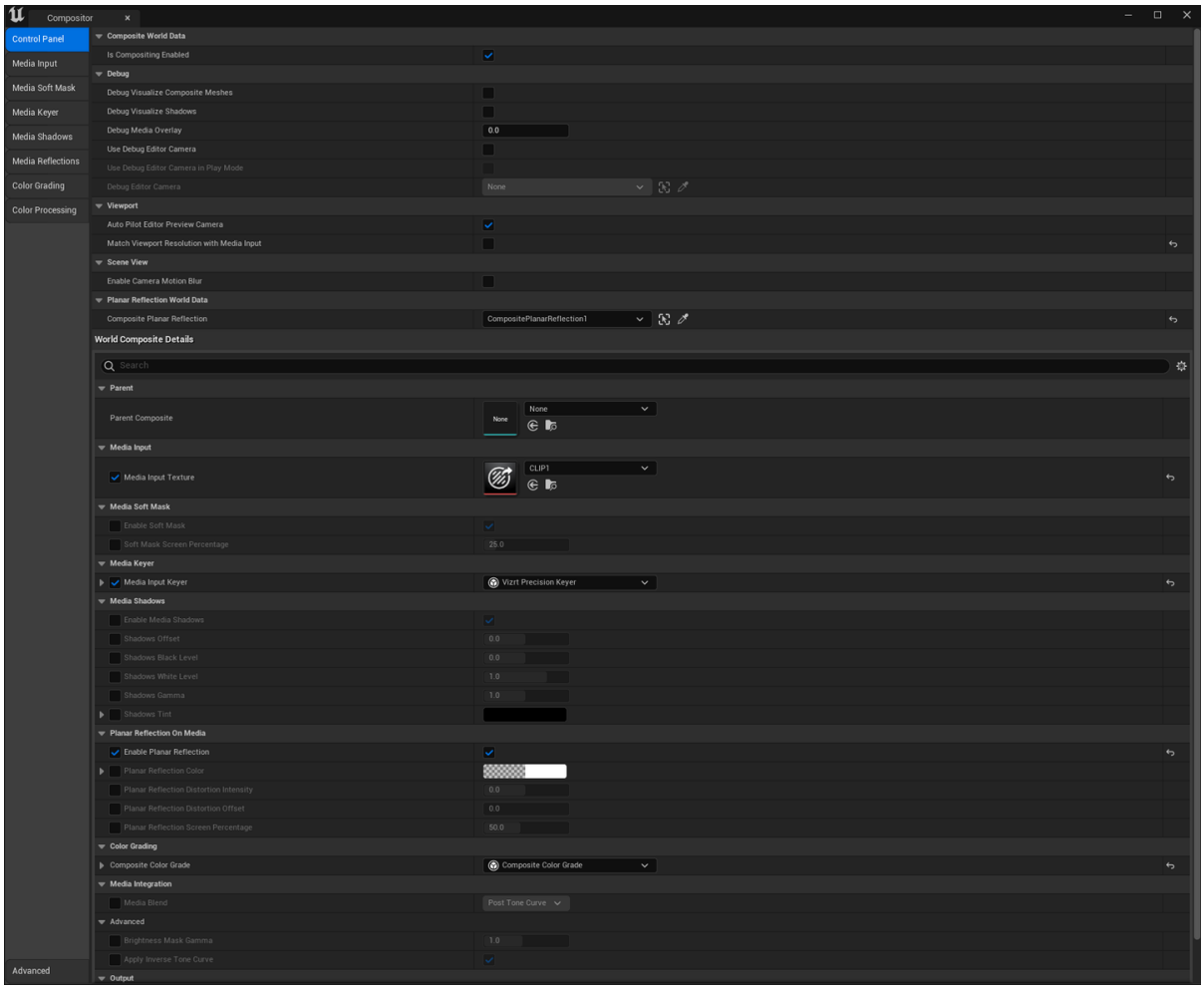
Copyright © 2026 Vizrt



To change Compositor settings we need to open the Compositor Menu.

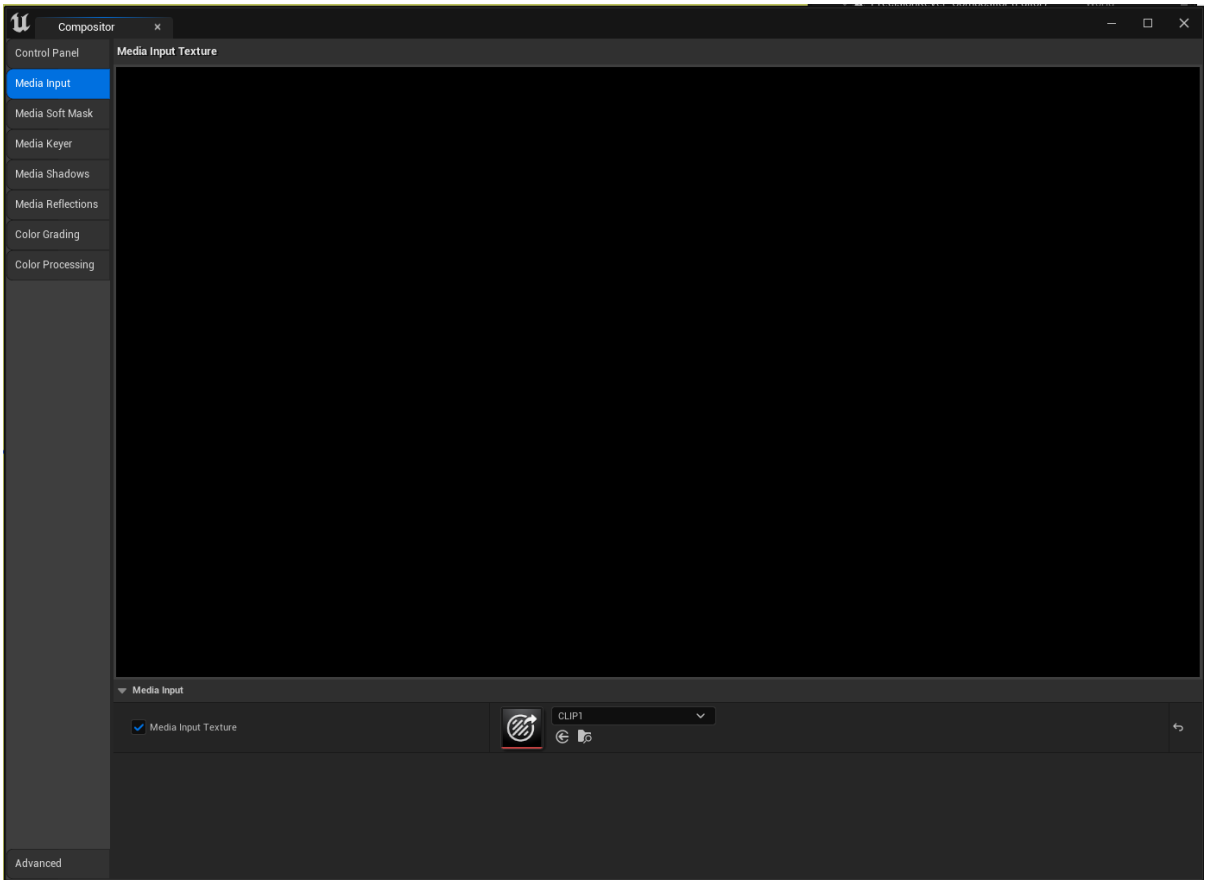


That shows the following menu.



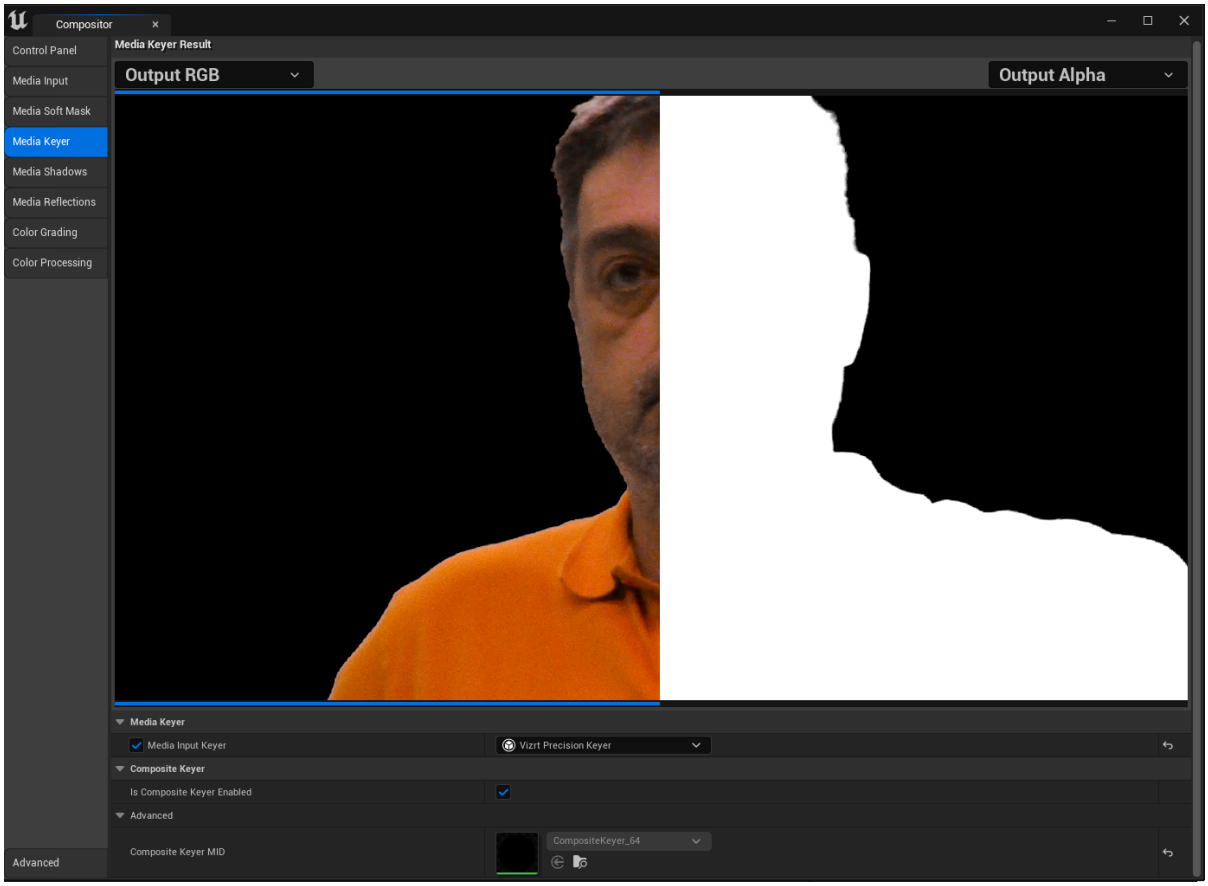
In the media input tab, select the input to use for the compositor.

Copyright © 2026 Vizrt



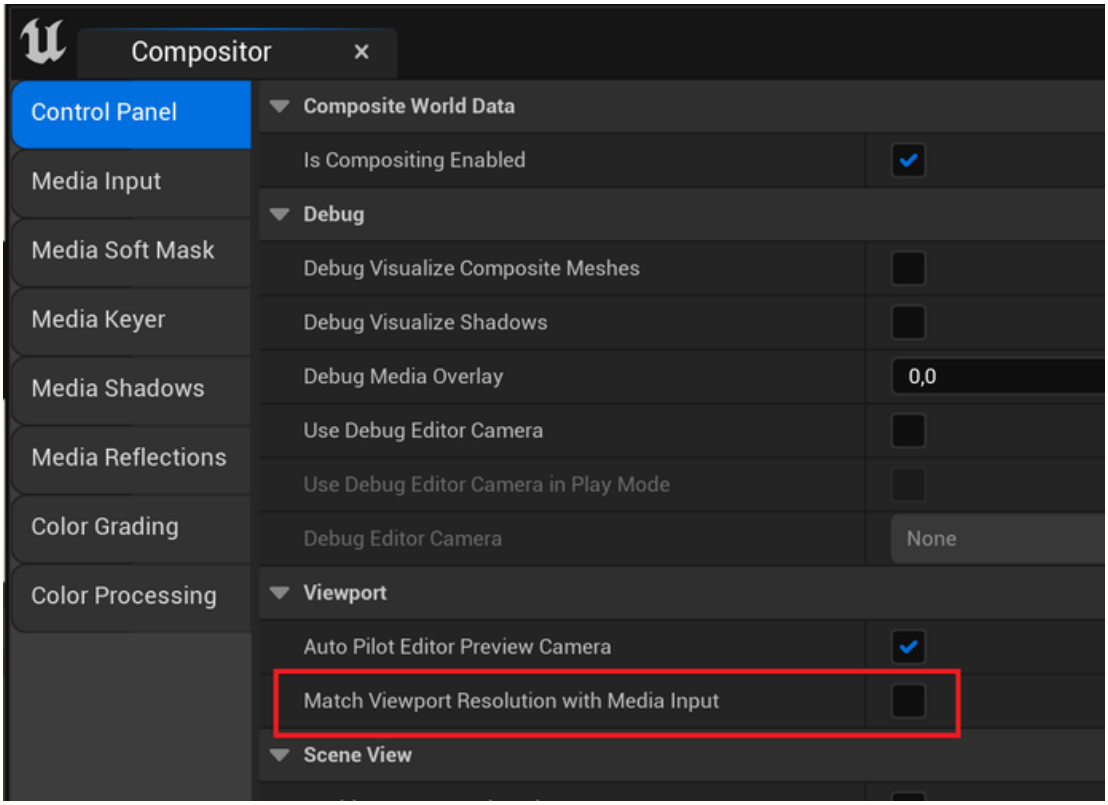
To setup the Keyer go to the Media Keyer tab and select the Vizrt Precision Keyer

Copyright © 2026 Vizrt



Note: If the Unreal output is being sent to Viz Engine through an **AUX** asset with the **Overscan** option set to True (used when Lens Distortion is active), the option **Match Viewport Resolution with Media Input** must be set to False.

Copyright © 2026 Vizrt

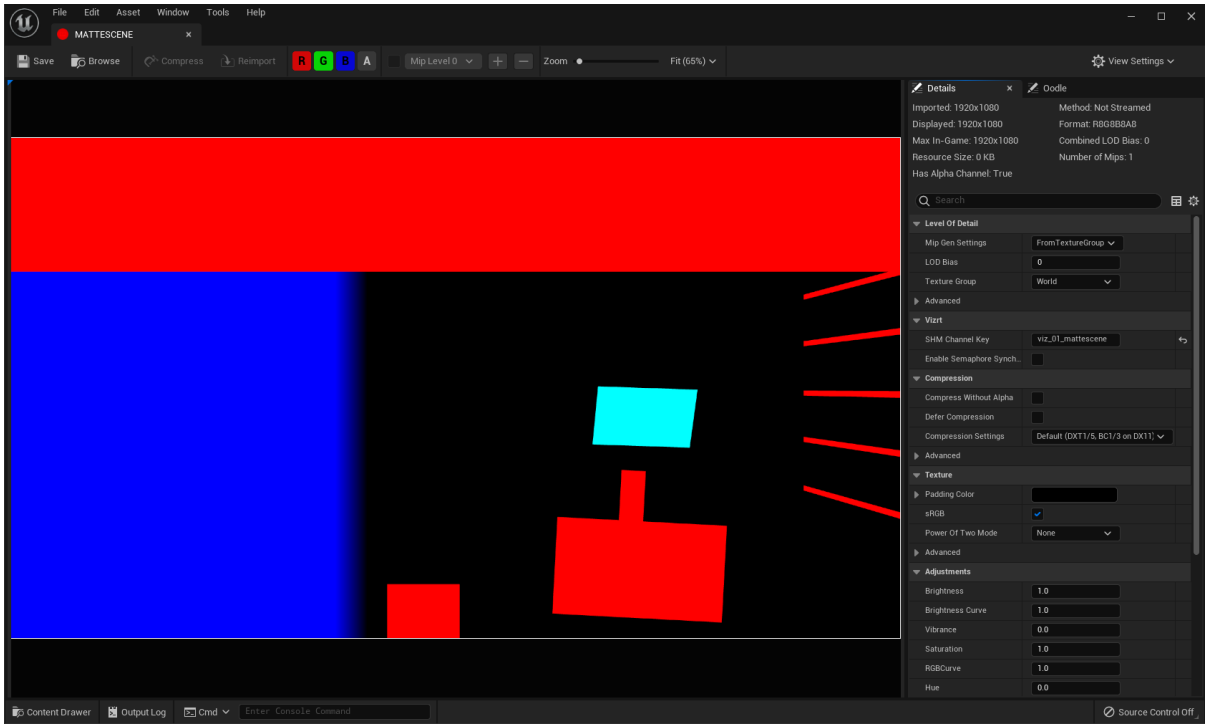


This is the result from the Precision Keyer:



If we have a matte scene the result reflects that.

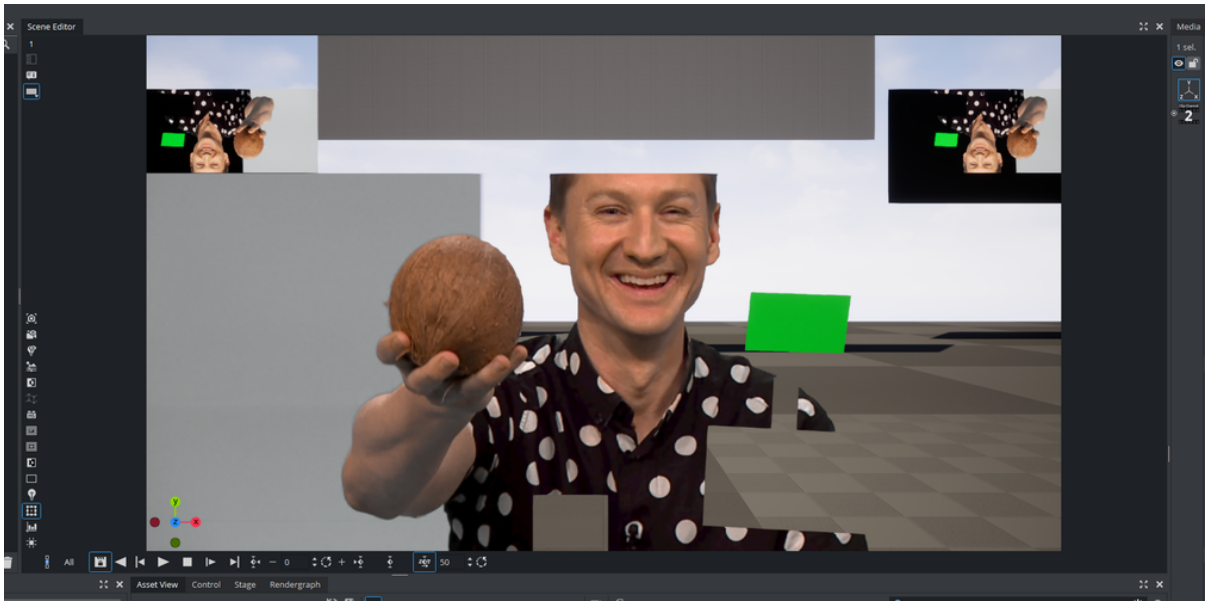
Copyright © 2026 Vizrt





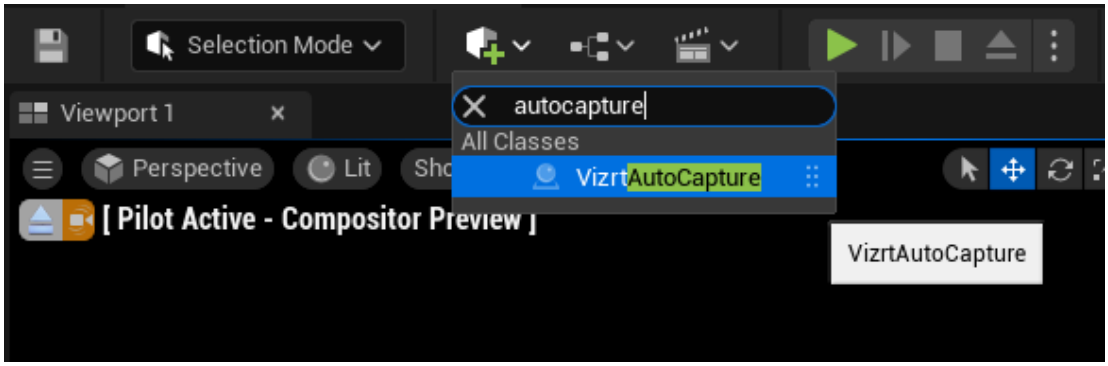
This also shows on the Viz Engine side if the AutoCapture is properly set up.

Copyright © 2026 Vizrt



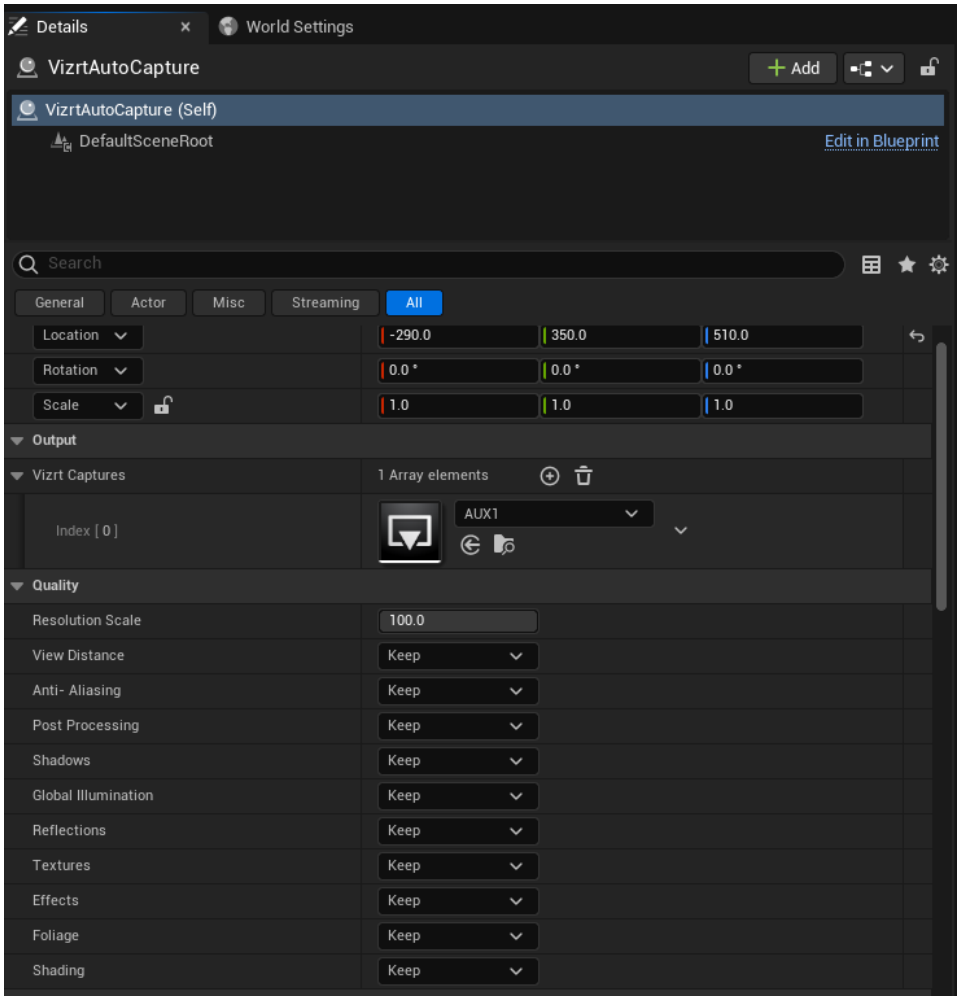
Setting up the AutoCapture to Share the Result with Viz Engine

Drag the VizrtAutoCapture into the level.



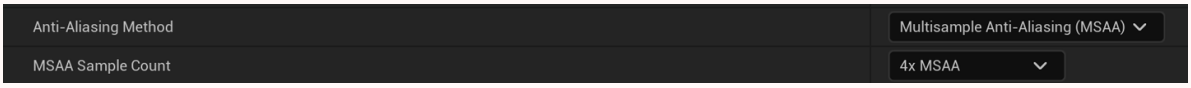
Make sure it has AUX1.

Copyright © 2026 Vizrt



When playing the level (PIE or Game Mode), the output is shared with Viz Engine.

Warning: If transparent objects are composited using the Vizrt Compositor, please make sure the **Anti-Aliasing Method** in **Project Settings > Engine - Rendering > Default Settings** is **NOT** set to Temporal Super-Resolution (TSR).



See Also

- Compositing in UE5

24.1.23 Setting up a Multi-GPU Workflow

This section explains how to set up a multiple GPUs to effectively split both Viz Engines to individual GPUs. With Viz Engine version 5.3.0 and newer, multiple GPU capabilities have been implemented.

Additional Hardware Requirements

In addition to the standard Hardware Requirements for the normal single GPU integration, a second GPU should be added.

- **Minimum Requirements:** 2x NVIDIA RTX A4000 (PCI 4.0x16 or higher)
- **Recommended Requirements:** 2x NVIDIA RTX A5000 (PCI 4.0x16 or higher)

Different GPU models of the same generation can be mixed, too, depending on the use case (for example, NVIDIA RTX 6000 Ada with RTX 4000 Ada).

> ⚠ **Note:** Unreal projects should be run in game mode with the `-game -renderoffscreen` flags for optimal performance.

Additional Software Requirements

- Viz Engine version 5.3.0 or newer.
- Unreal Engine version 5.3.2 or newer.

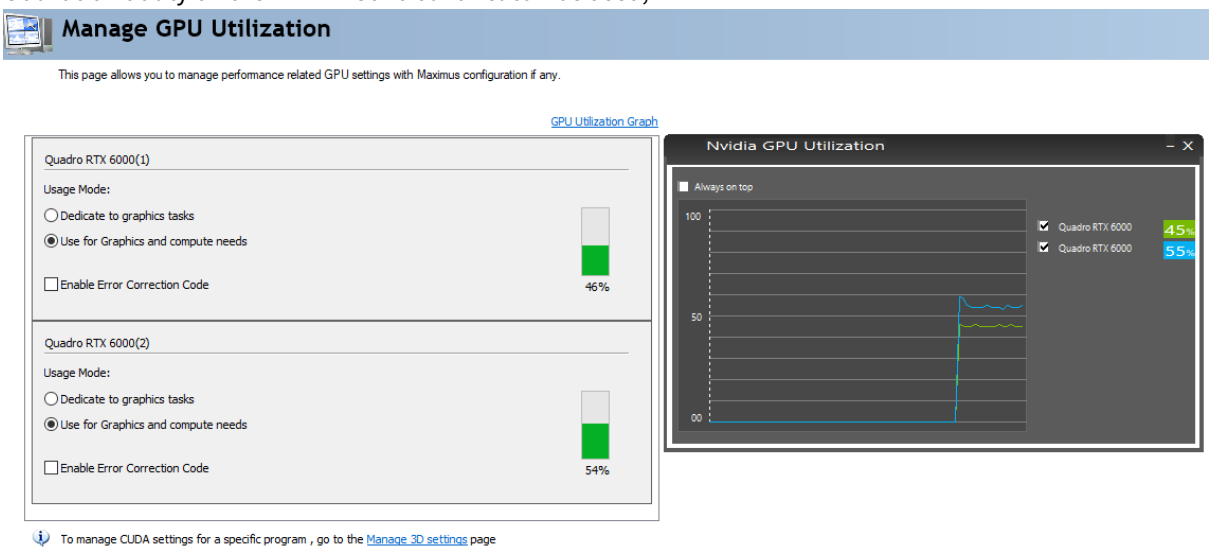
Activating Multi-GPU Support

To activate support for multiple GPUs inside Viz Engine with Unreal there are a few steps:

- Navigate to *X:\ProgramData\vizrt\VizEngine\VizEngine-0.cfg*. If you remove the second GPU from the system and want to return the integration to its single GPU pipeline, set the flag to `0`.
- Search for the `unreal_use_multi_gpu` flag and set it to `1`.
- Navigate to the installed Unreal Engine folder (should be under Program Files\Epic Games\UE_5.X), go to the *\Engine\Config* folder and open *BaseEngine.ini*.
- There, add the following command `r.GraphicsAdapter=X (X being the gpu adapter you want for UnrealEngine)`, to the **[/Script/Engine.RendererSettings]** section.
  If Unreal is on the monitor's GPU, 0 is whichever GPU is connected there. If it is offscreen, 0 is GPU0. To return the integration to the single GPU pipeline, remove the command from *BaseEngine.ini*. Alternatively if the above command is not working for you, the following flag can also be set on the launch options `-graphicsadapter=X`
- Launch Viz Engine with the command `-GX` (where X is the GPU number. This can be checked on NVIDIA inspector and task manager). When in doubt, use NVIDIA inspector to see if the GPU load is correct (as in, if there is movement on both GPUs as that would indicate that Unreal is on one GPU and Viz on another). Typically, the first GPU is `-G1` and the second GPU is `-G2`.
- The simplest possible configuration to use would be having Unreal just use the default GPU it wants and setting the GPU flag only on the Viz Engine side.

Copyright © 2026 Vizrt

- An example of a simple configuration would be: Lenovo P620 with 2x NVIDIA A5000 GPUs, Unreal Engine set to use the first GPU and Viz Engine set to use the second GPU with the `-G2` flag.
- To make sure the multi GPU integration is correctly splitting both Engines to each GPU, the NVIDIA GPU Utilization utility on the NVIDIA Control Panel can be used;



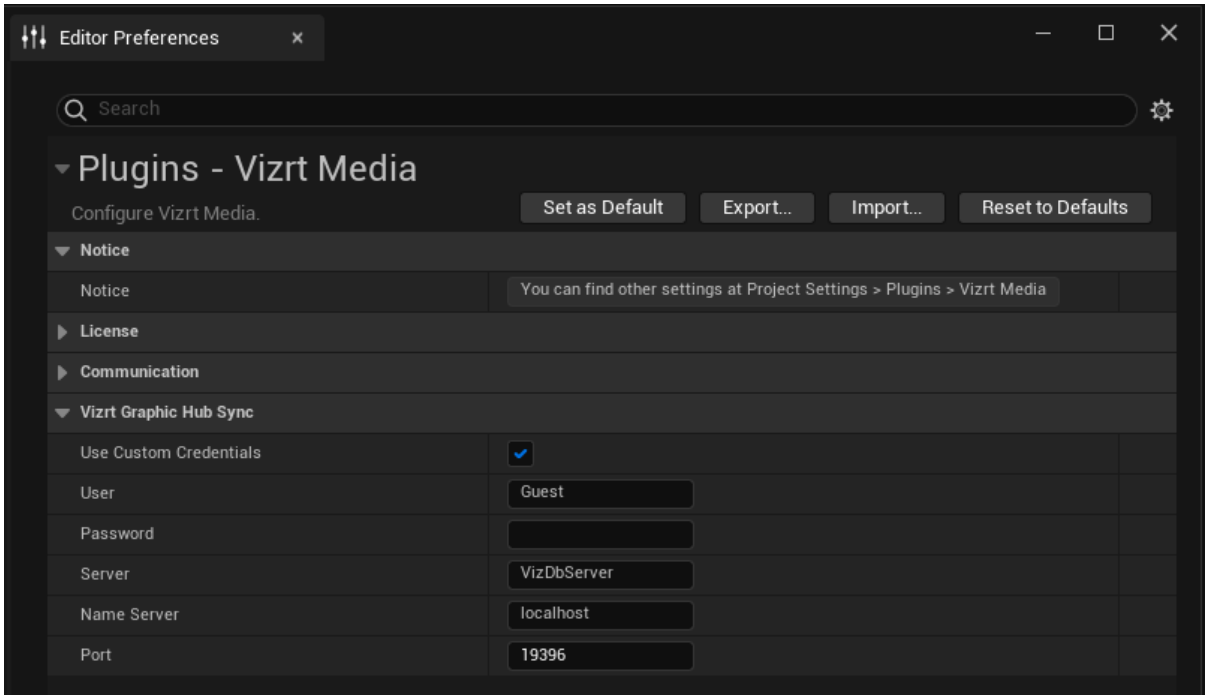
Other Considerations

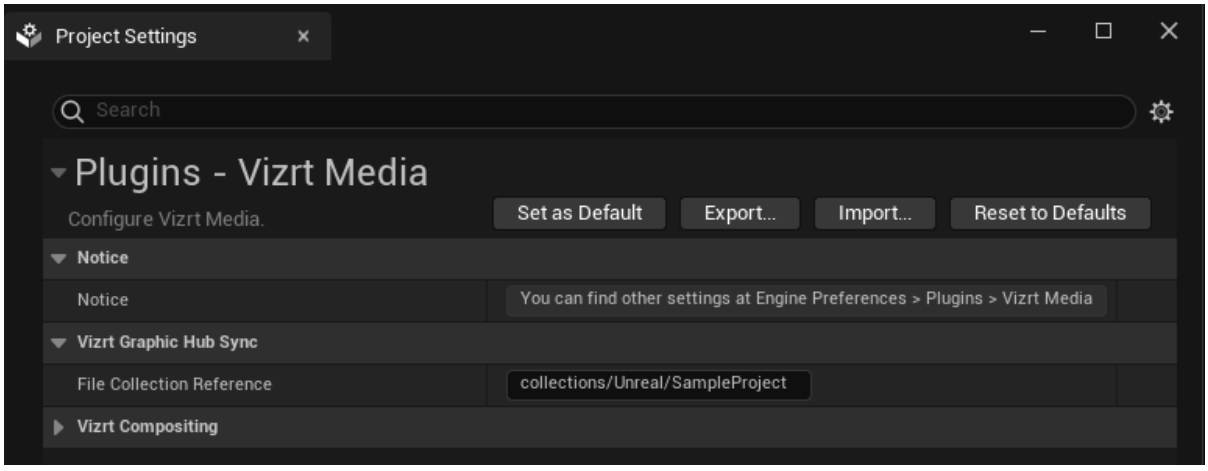
- Delay configuration should be an extra two frames over the normal single GPU delay.

Copyright © 2026 Vizrt

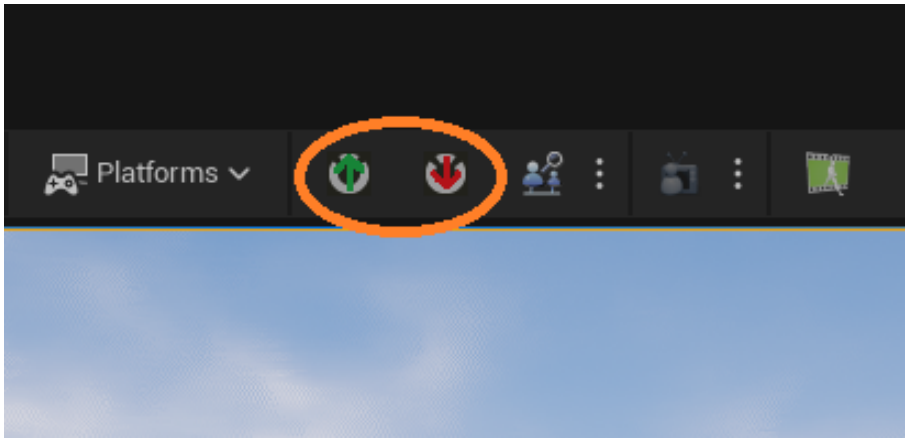
24.1.24 Synchronization with Graphic Hub

Note: Only the folder of the current project is synchronized with Graphic Hub. Any modifications to plug-ins or content installed directly in the Unreal Engine installation folder are not synchronized.





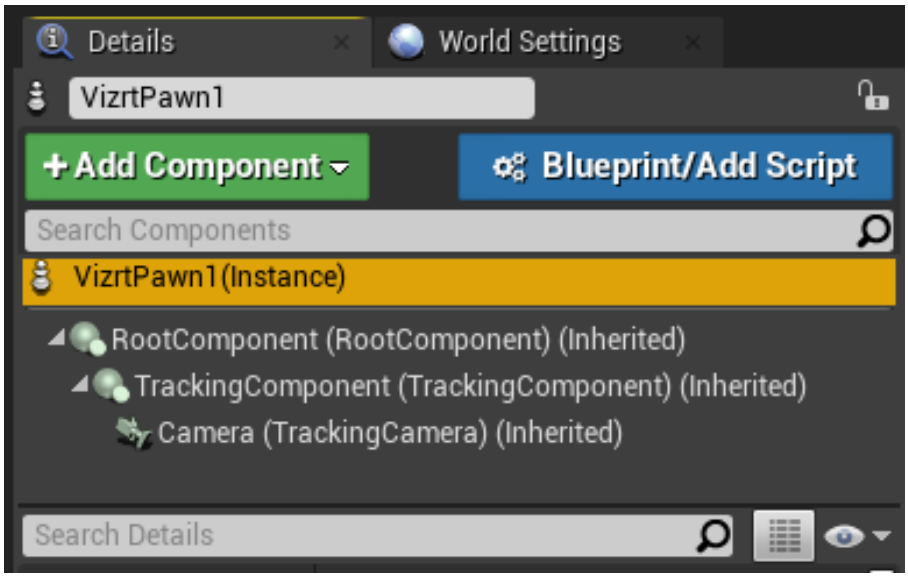
Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

24.1.25 Unreal Engine Vizrt Pawn

The **VizrtPawn** is the Actor that receives the tracking data (location, rotation, field of view, etc.) and controls the values of the Components within its hierarchy:



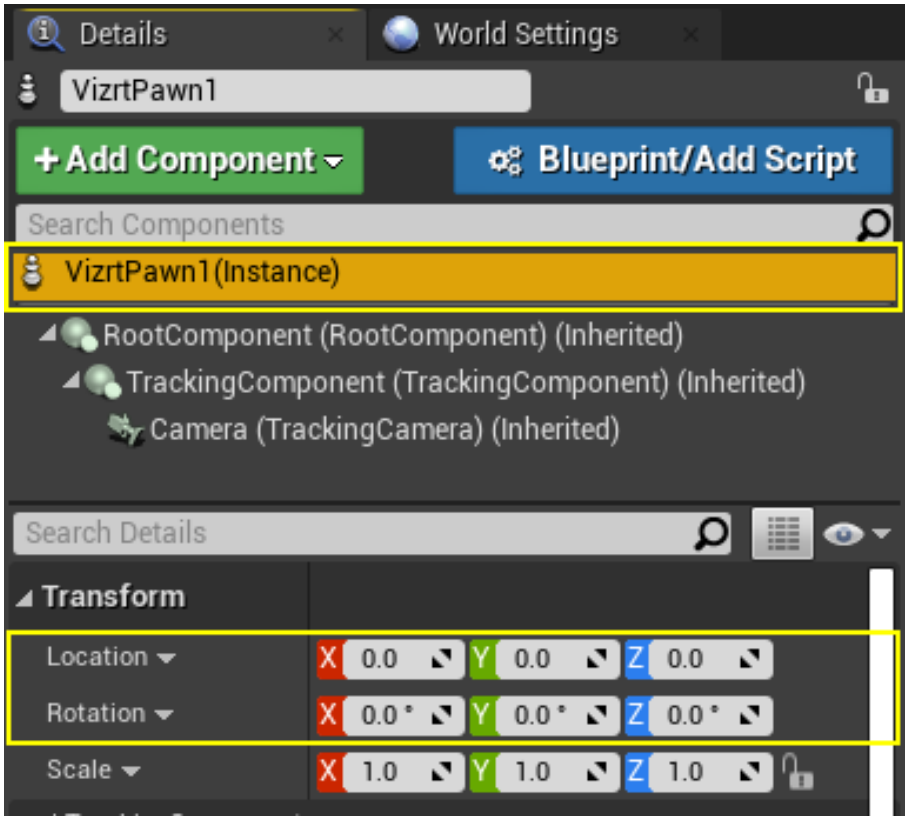
- **VizrtPawn1:** The main Actor component.
- **RootComponent:** An internal component and cannot be changed.
- **Tracking Component:** Receives tracking location and rotation.
- **Camera:** A Cinematic Camera derived Component, which receives tracked Field of View, Focal Plane and FStop.

Camera Tracking Offset

Tracking data sent from Viz Engine to Unreal Engine usually has its origin set at (0, 0, 0), which might not always match the center of a Virtual Set design. To overcome this limitation, it is possible to change the global location of the tracked camera inside Unreal Engine.

The **VizrtPawn** Component defines the global location of the Camera within a set. Since tracking data is being applied on **TrackingComponent**, it is possible to change the origin by just changing the location of the VizrtPawn:

Copyright © 2026 Vizrt



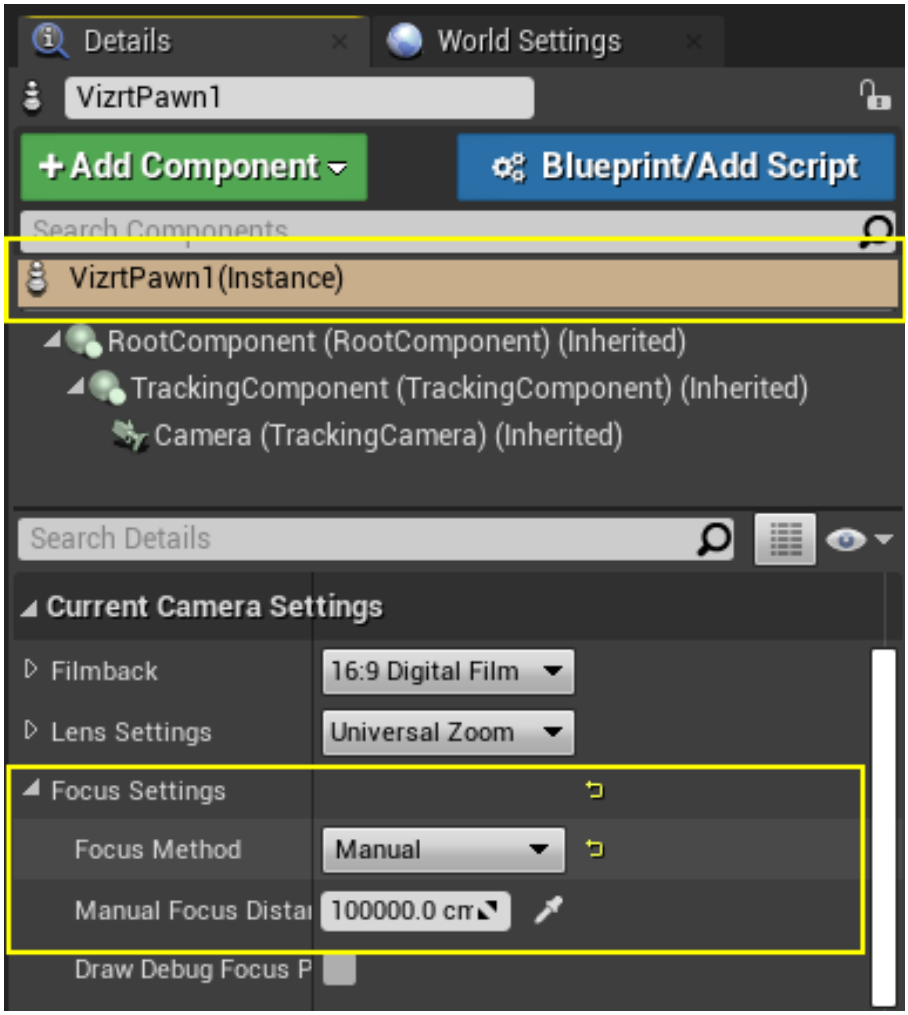
Set *Location* to the origin of the virtual set. Usually only the X and Y components need to be changed and Z should be 0.

If the main view direction of the tracking data does not match with the Virtual Set design the *Rotation* can be changed in addition. Only change Z component, X and Y should be 0.

Depth of Field

The Depth of Field effect is disabled by default, but can be enabled within the **Camera** Component of the VizrtPawn:

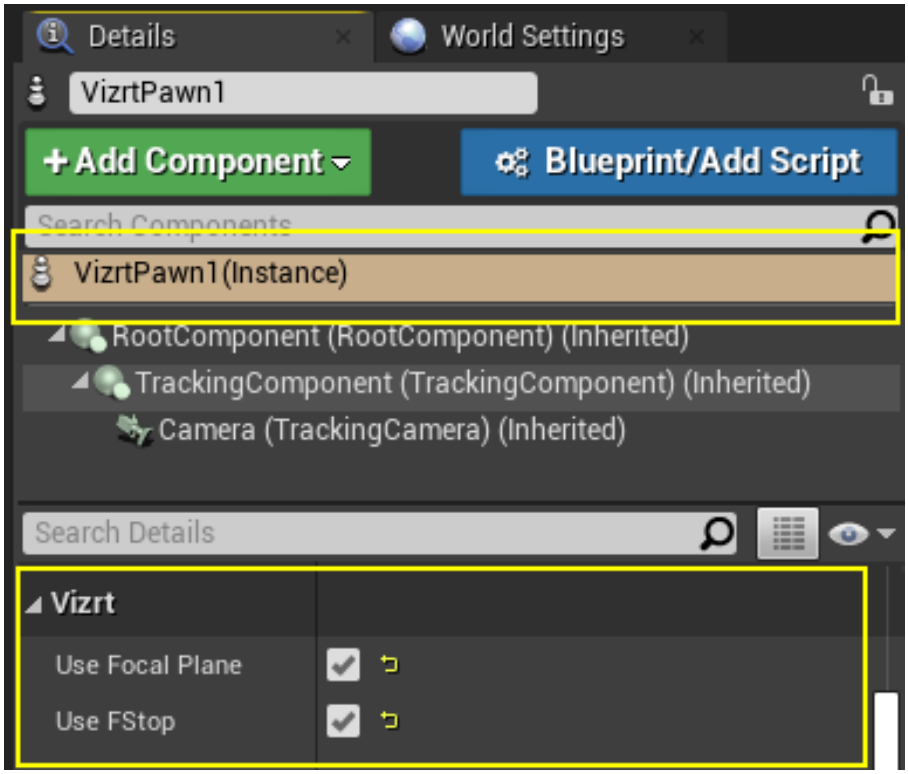
Copyright © 2026 Vizrt



Open the **Focus Settings** section of the VizrtPawn`s Camera and change the `Focus Method` to `Manual`.

The `Manual Focus Distance` can be changed manually to achieve the desired effect or can be linked to Viz Engine tracking data.

Copyright © 2026 Vizrt



Tracking Hub and Viz Engine can be calibrated and configured to send **Focal Plane** and **FStop** to Unreal Engine. To link Unreal's Camera with the tracking data, **Use Focal Plane** and **Use FStop** can be enabled.

Focal Plane changes Camera's the `Manual Focus Distance` value. FStop changes the Camera's `Current Aperture` value.

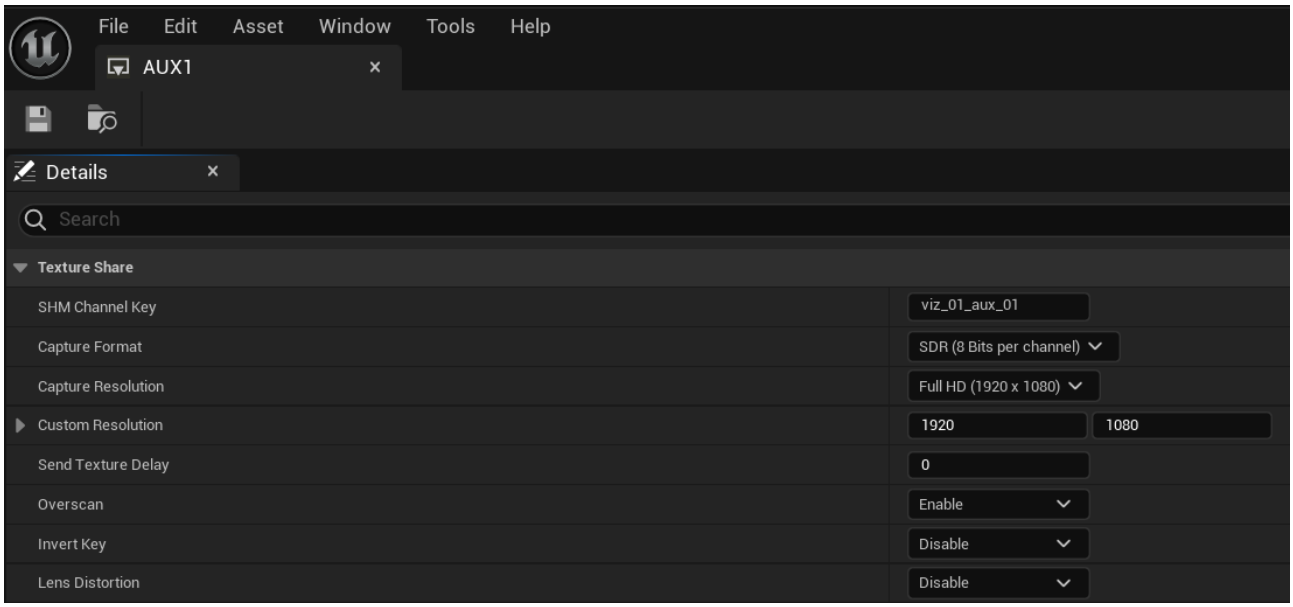
Information: The Depth of Field function is designed to work in the Viz Engine Render Pipeline only!

Copyright © 2026 Vizrt

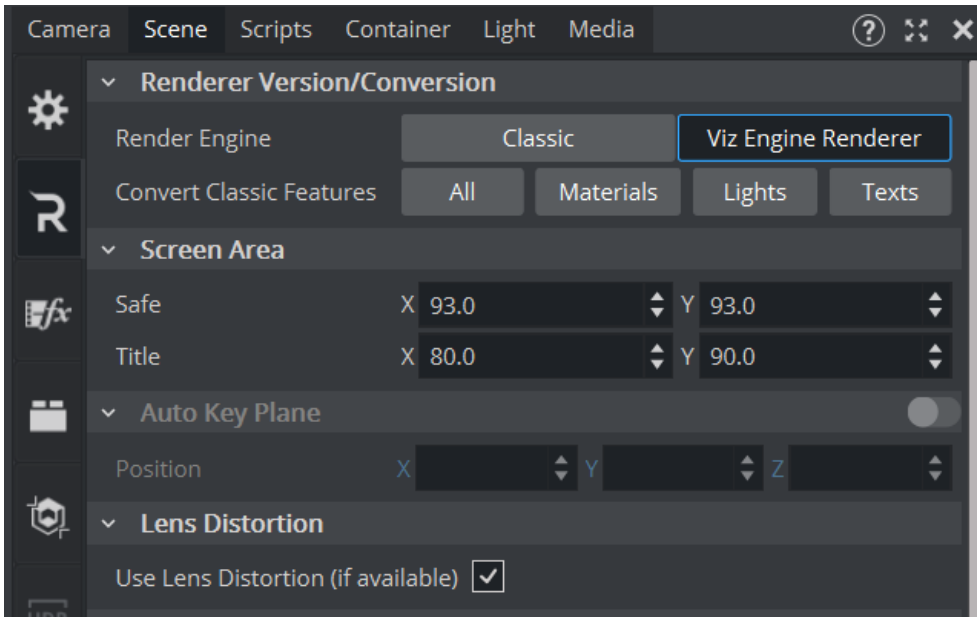
24.1.26 Lens Distortion on Unreal Engine

When using Unreal Engine, we need to have the live/clip undistorted for it to properly blend with Viz Engine.

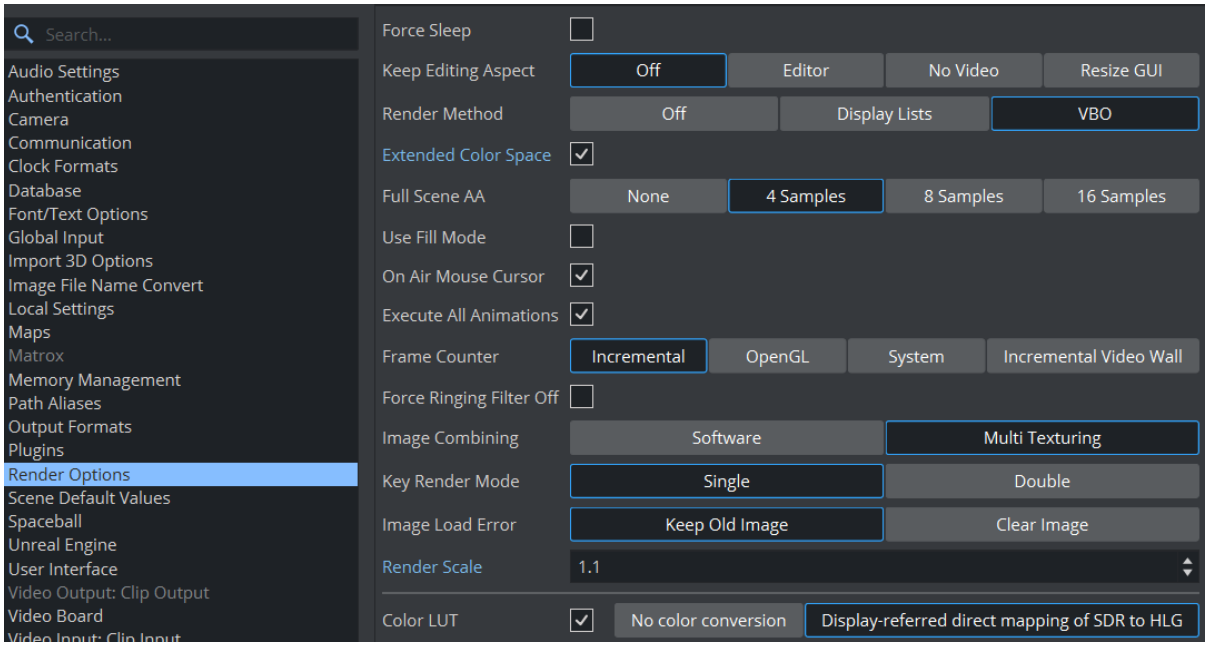
Depending on the Render Sequence, we may need to activate Lens Distortion and Overscan on the AutoCapture in Unreal.



Lens Distortion should be active in Viz Engine and have a greater than 1 render scale.



Copyright © 2026 Vizrt



As such, we currently support the following:

	Virtual Studio (2)	External VS (2)	External AR (2)	Multi-Layer Virtual Studio (2)	Multi-Layer XR Set (2)	Multi-Layer External VS (2)
Internal	✅	✅	✅	✅	✅	✅
Libero	✅	✅	✅	✅	✅	✅
Xpecto	✅	✅	✅	✅	✅	✅
Radial	✅	✅	✅	✅	✅	✅
Stype	⚠ (1)	⚠ (1)	⚠ (1)	⚠ (1)	⚠ (1)	⚠ (1)
Trackmen	⚠ (1)	⚠ (1)	⚠ (1)	⚠ (1)	⚠ (1)	⚠ (1)
Image-Based	✅	✅	✅	✅	✅	✅

(1) Clip/Live won't match due to missing undistortion when compositing in Unreal Engine. Viz Engine should be used for composition.

(2) Auto capture with Overscan On and Lens Distortion On.

Copyright © 2026 Vizrt

24.1.27 Outputting Directly on a DisplayPort

On some specific cases (for example, a videowall with multi GPUs on a system with significant compounded delay) outputting the Unreal Engine directly through the DisplayPort could be desireable.

On a multi-GPU configuration, going directly through the display port has a “saving” of four frames (frame 610 through DP vs frame 606 through Viz Engine on the following screenshot).



There is a limitation when doing this which is the compositing has to be fully done in the Unreal Engine (items from Viz Engine can be sent to Unreal via GFX Channel).

To run Unreal directly on the DisplayPort output the project needs to be run using `-fullscreen` and `-game` flags along with the proper resolution flag. The AutoCapture blueprint should be removed to avoid also sending the texture to Viz (it can be there, but it won’t be useful in this specific case).

To make sure it is an exclusive fullscreen with your desired resolution, edit the *GameUserSettings.ini* (or create one) with the following:

```
[/Script/Engine.GameUserSettings]
FullscreenMode=0
LastUserConfirmedDesktopResolution=1920x1080
ResolutionSizeX=1920
ResolutionSizeY=1080
```

(`FullscreenMode` : `0` for exclusive fullscreen, `1` for borderless windowed, `2` for windowed)

Copyright © 2026 Vizrt

These settings assume Exclusive Fullscreen with a resolution of 1920x1080. To change the resolution, set the values required (3840x2160, maybe more for higher resolution displays / videowalls). This should be enough for the project to run on the DisplayPort output.

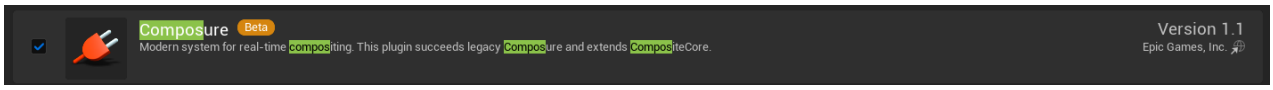
For videowalls that run through NVIDIA Mosaic, the process is similar with the added set-up of the NVIDIA Mosaic part. An official guide and manual can be found here: https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/quadro-product-literature/NVMosaic-UG.pdf

A more streamlined guide can also be found at https://nvidia.custhelp.com/app/answers/detail/a_id/3568/~/how-to-setup-mosaic-using-nvidia-control-panel.

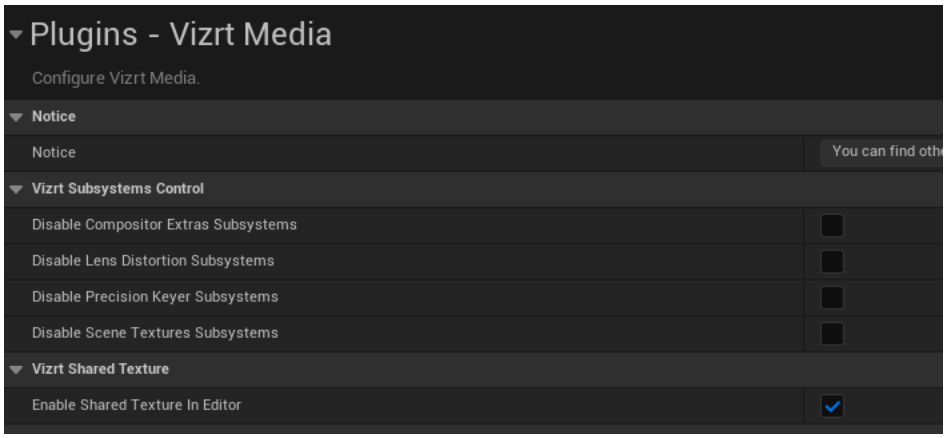
Copyright © 2026 Vizrt

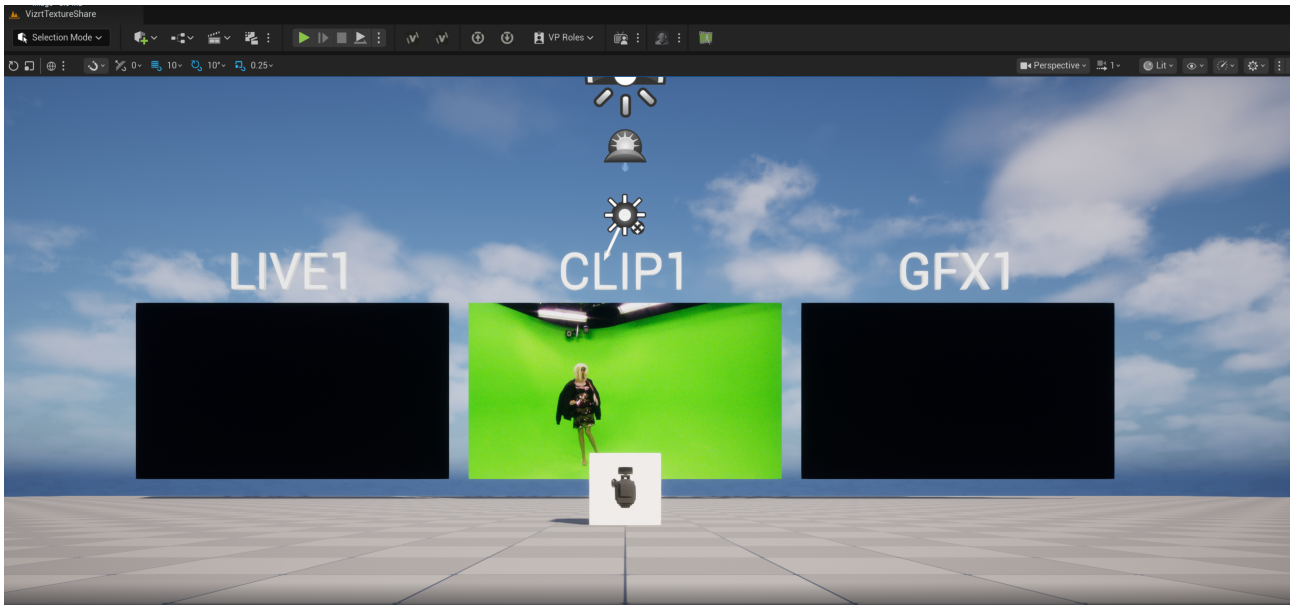
24.1.28 Compositing in Unreal Engine 5 (Composure Workflow)

Activate the Composure plug-in.

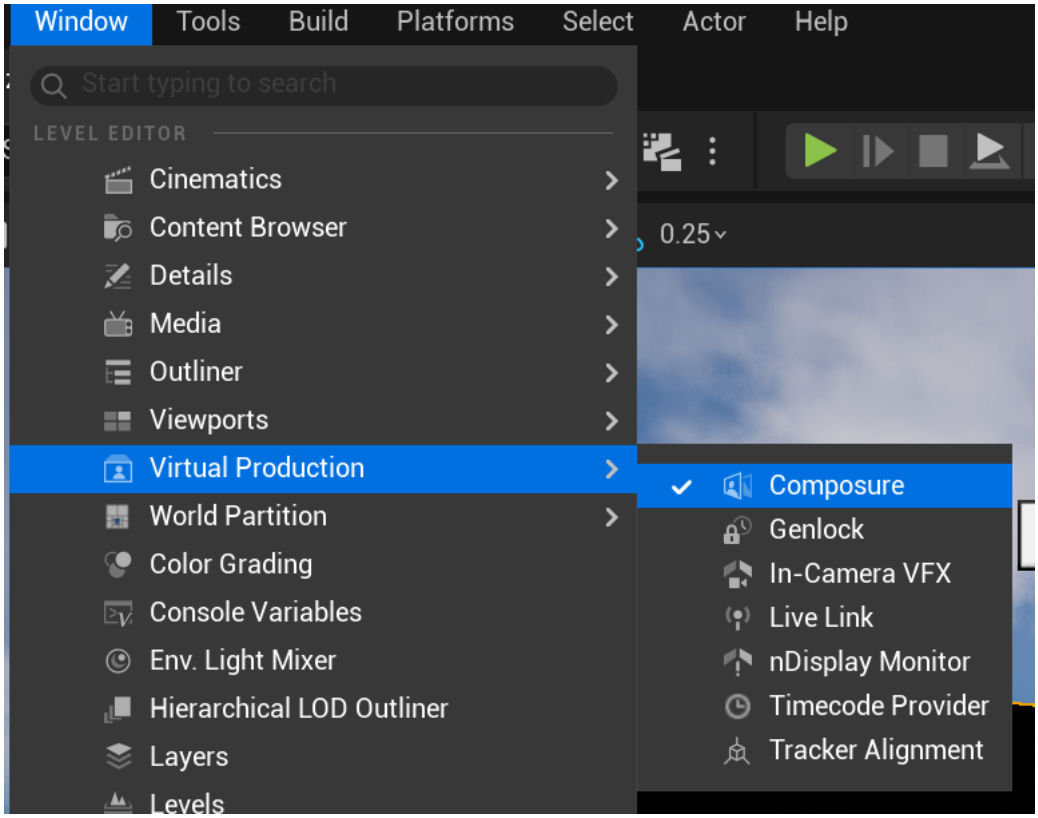


In the settings, activate “Enable Shared Texture in Editor” to allow for shared textures to work without the project running in editor mode:

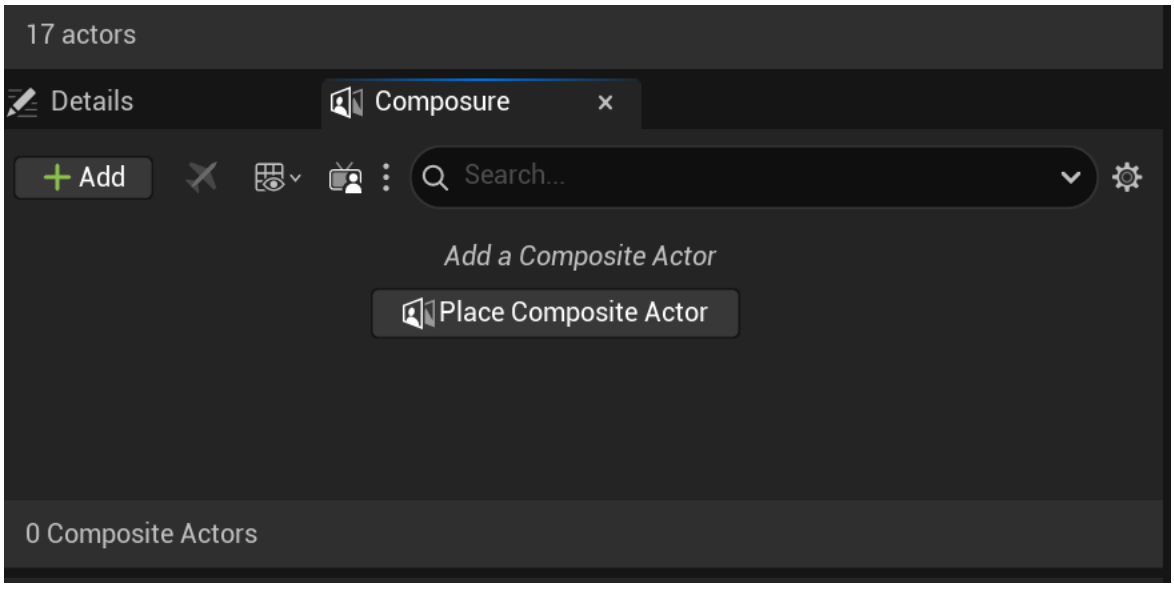




We need to go to Composure window:

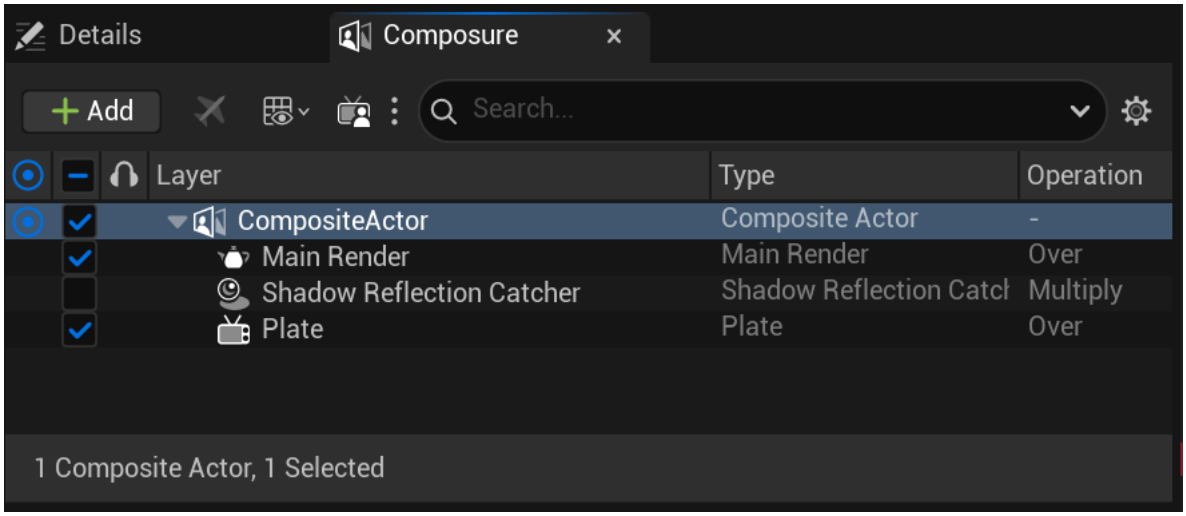


This should open a Composure window:

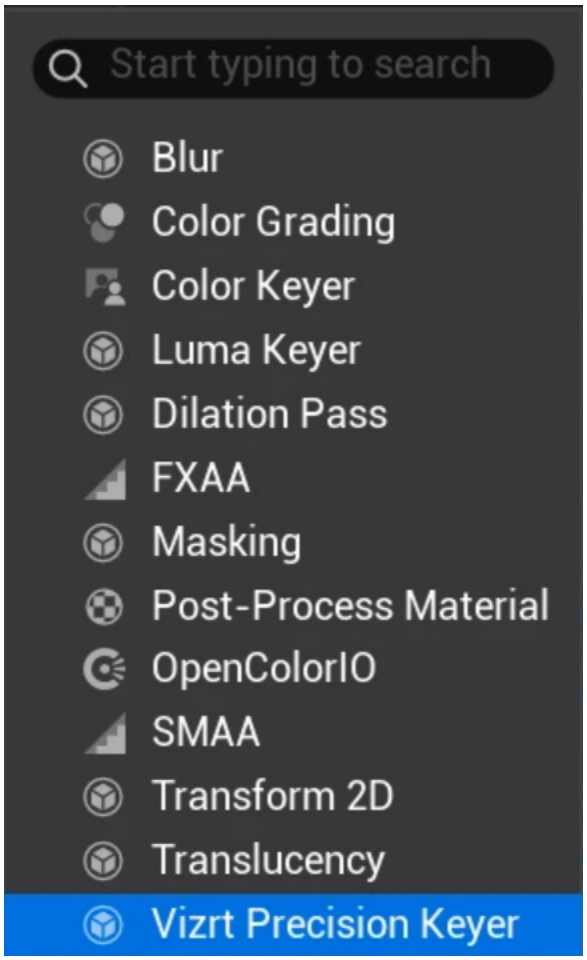


Select Place a Composite Actor:

Copyright © 2026 Vizrt



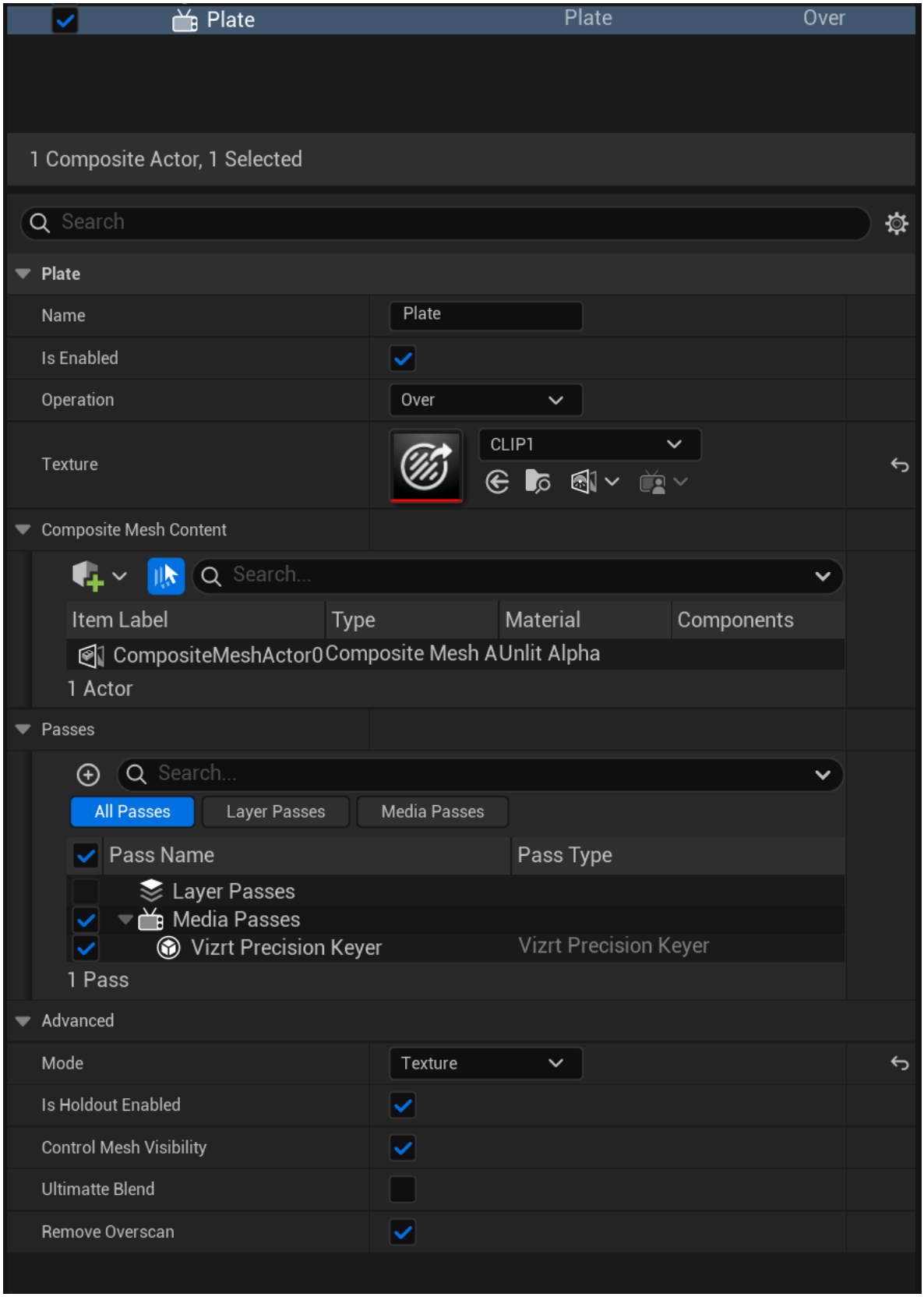
On Plate select:
Texture - CLIP1 (or LIVE1 or another media plate you want);
Add the Media Passes you might want (VizrtPrecisionKeyer or something else);



Copyright © 2026 Vizrt

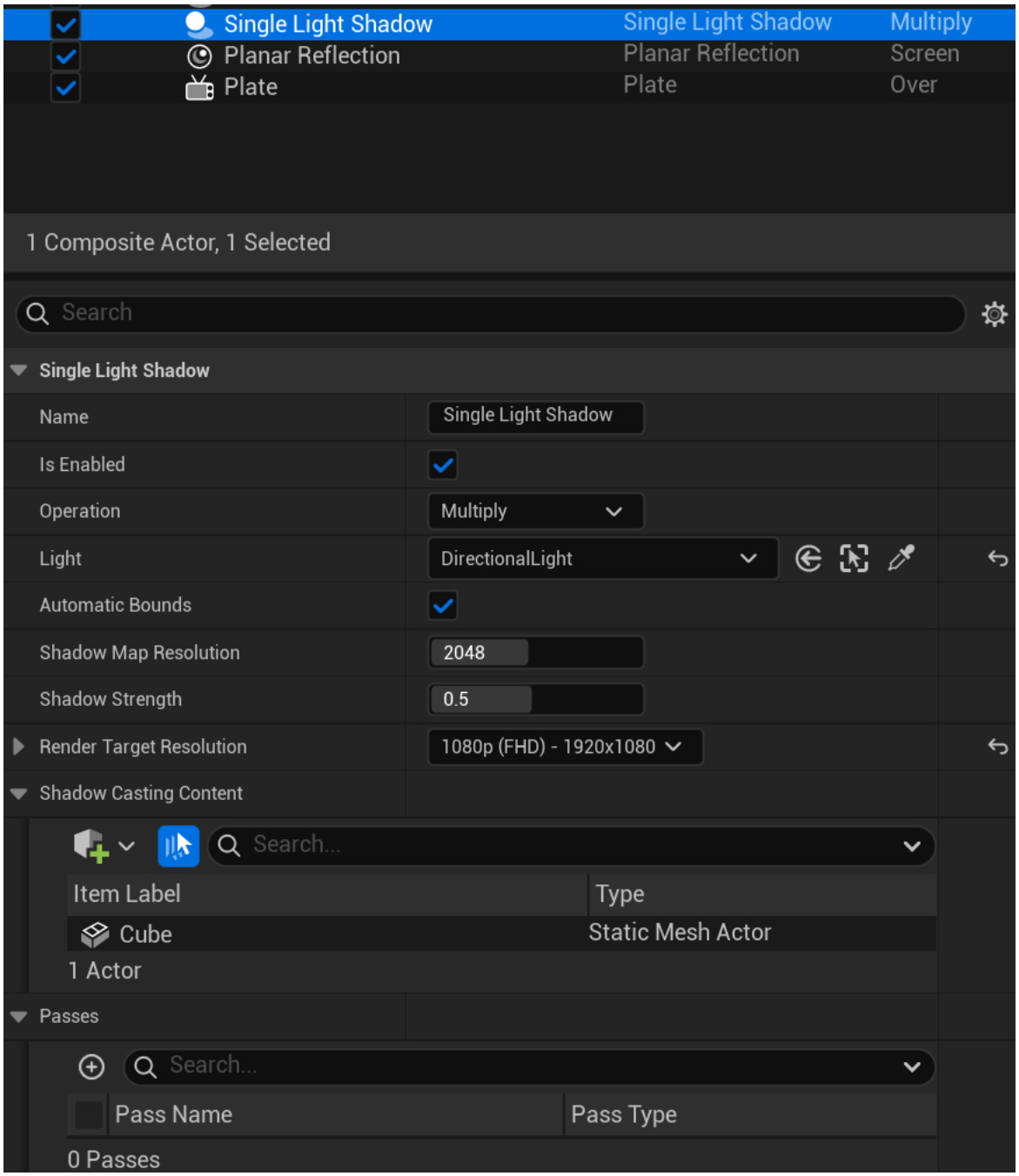
Mode - Texture

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

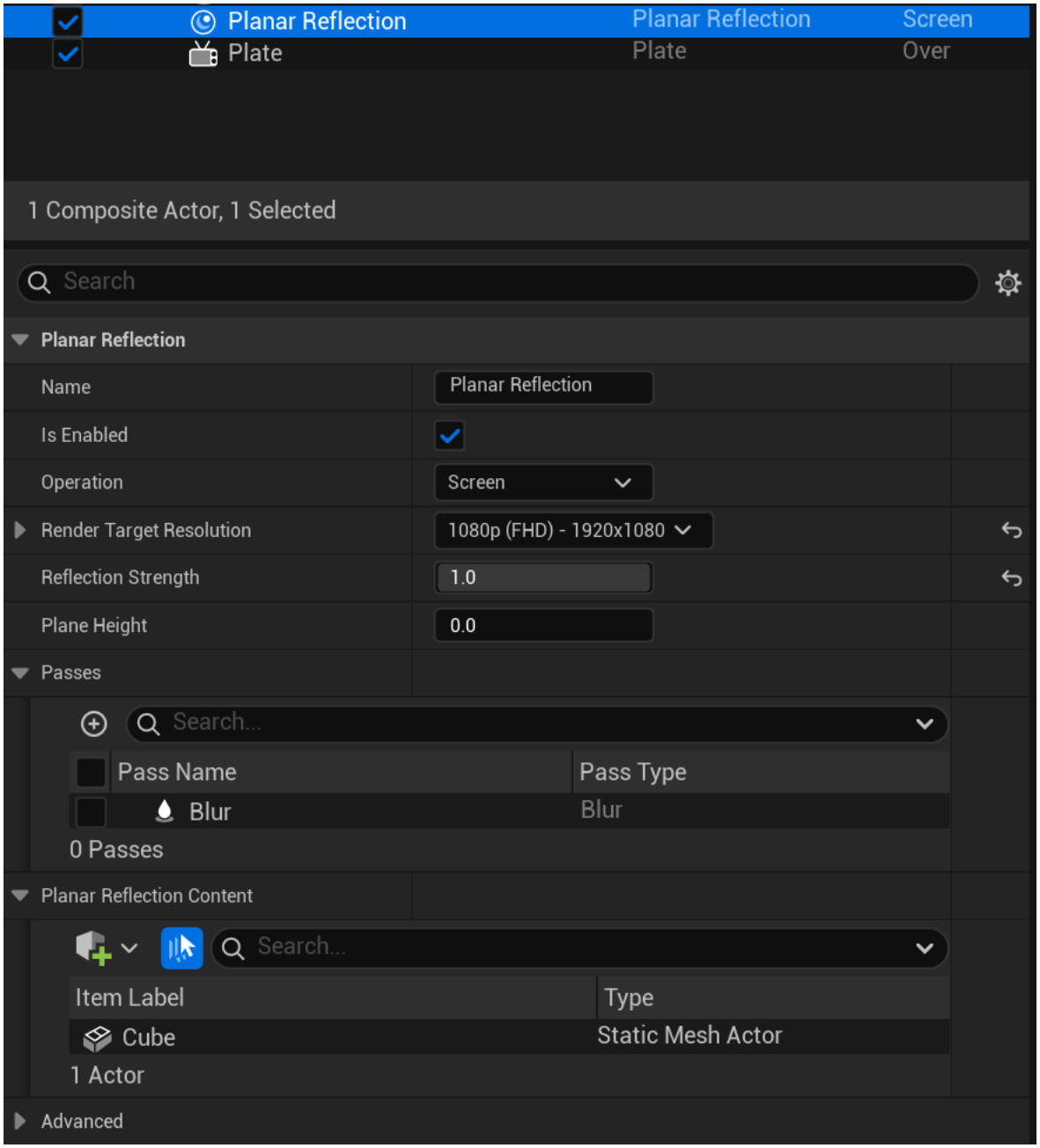
To add shadows add a Single Light Shadow and check its settings - you can add Passes to modify its aspect and you need to add the actors you want the shadows to be represented (Cube in this example):



To add

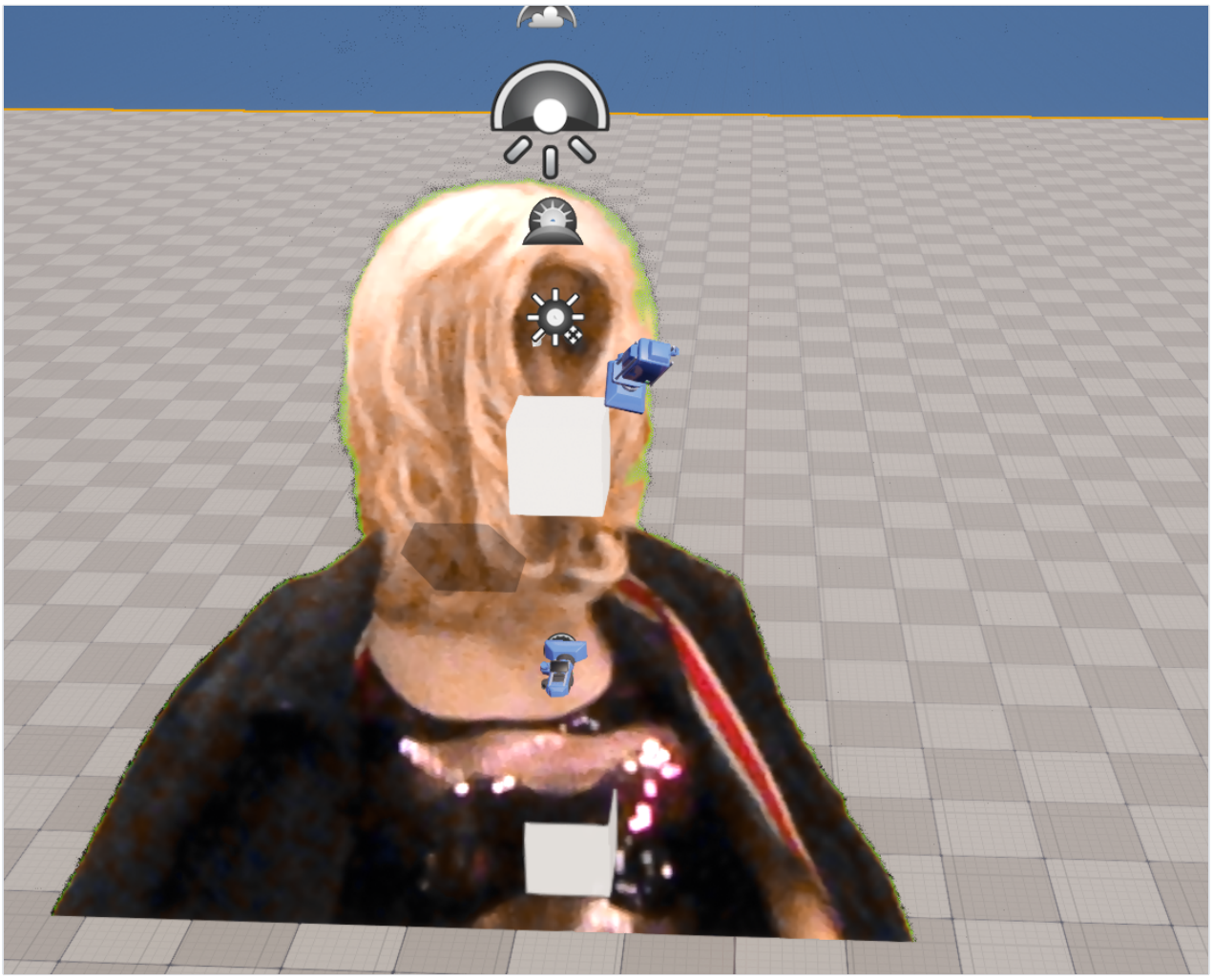
Reflections add Planar Reflection and check its settings - you can add Passes to modify its aspect and you need to add the actors you want to be reflected (Cube in this example):

Copyright © 2026 Vizrt



With these settings you should now be able to have the texture being keyed (the following is a zoomed in example and not a representation of final quality):

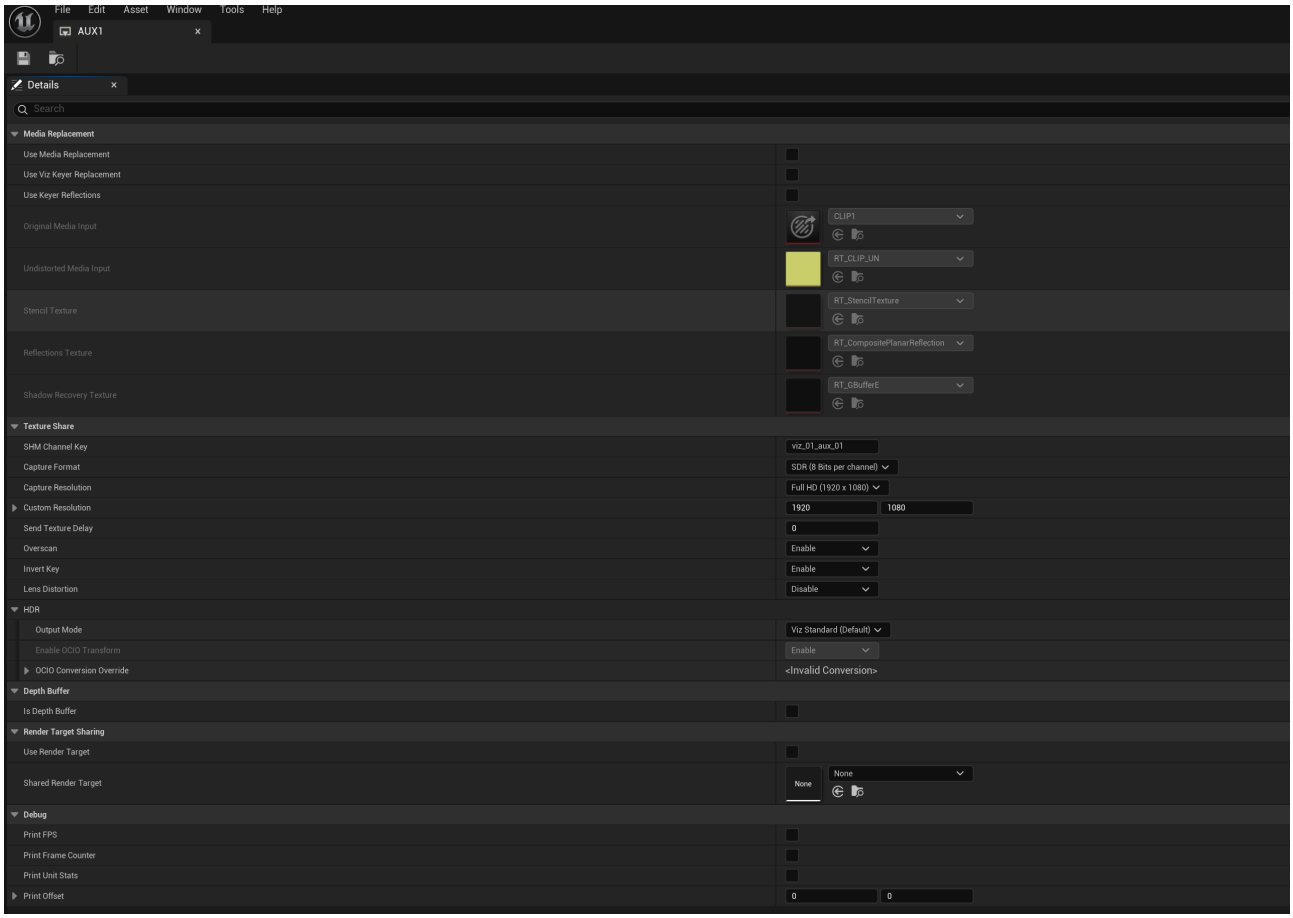
Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

24.1.29 Using OpenColorIO with the Unreal Integration

The Unreal Integration now supports OpenColorIO and its vast array of color spaces.
This allows for a more seamless and diversified SDR and HDR workflows with Viz+Unreal scenes.
The options to change between pipelines are located in the AutoCapture blueprint in the HDR tab:

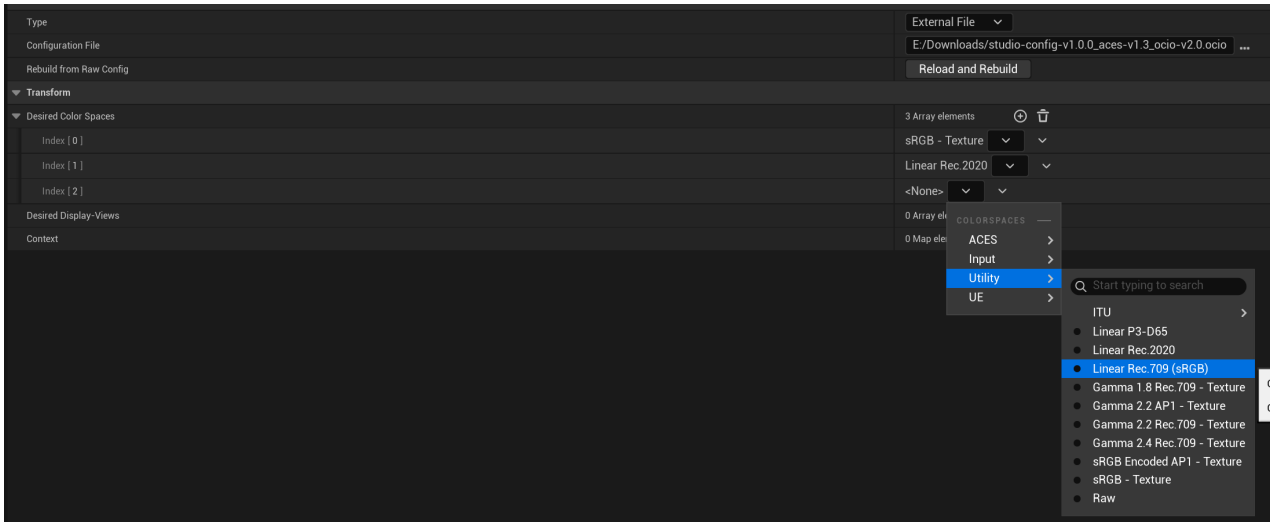


To use the OCIO profiles, a OCIO Config file should be created and a profile added:

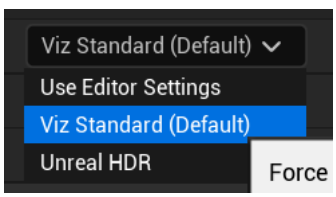


The desired color spaces should be added:

Copyright © 2026 Vizrt

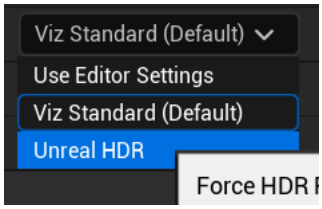


Back to the AutoCapture, to use the previous non-OCIO pipeline, the setting should be kept in Viz Standard (Default):



To use the new pipeline, Unreal HDR should be selected:

Copyright © 2026 Vizrt



From there, the option to add a OCIO profile is now available:



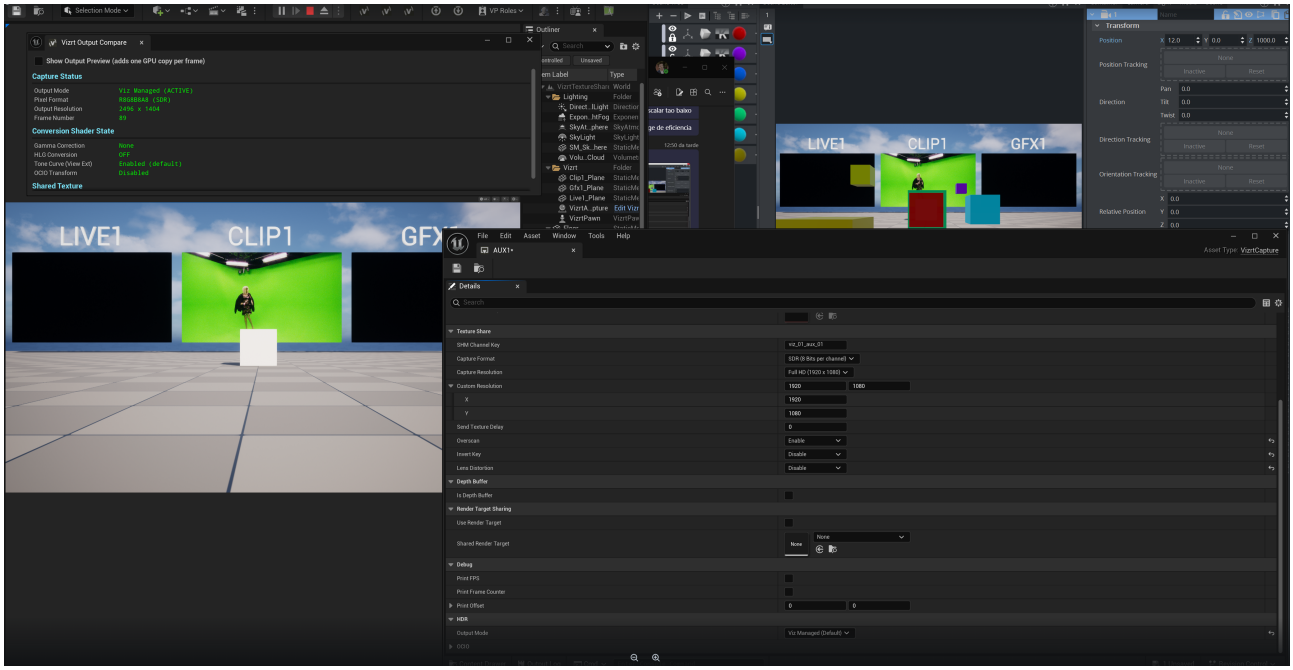
With the correct OCIO config selected, the source and destination color spaces can now be chosen:



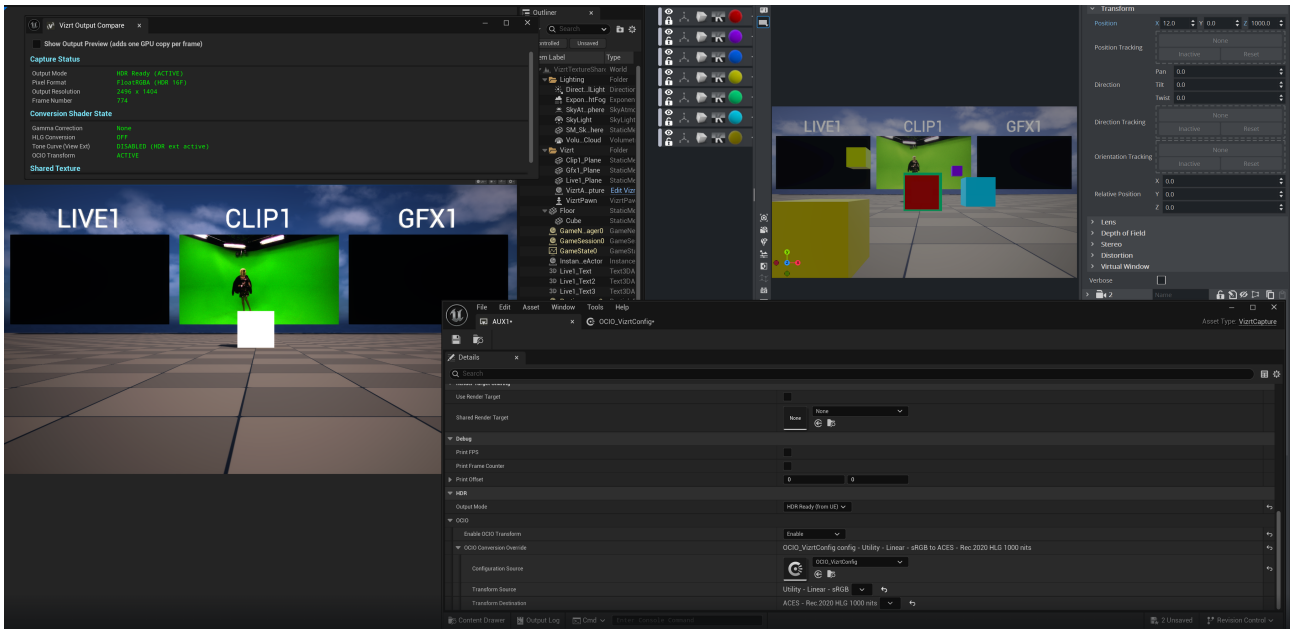
When running the project now, the output color will now fully respect the selected source and destination spaces without the unreal tonemapping affecting the tones. A few examples below:

Standard

Copyright © 2026 Vizrt

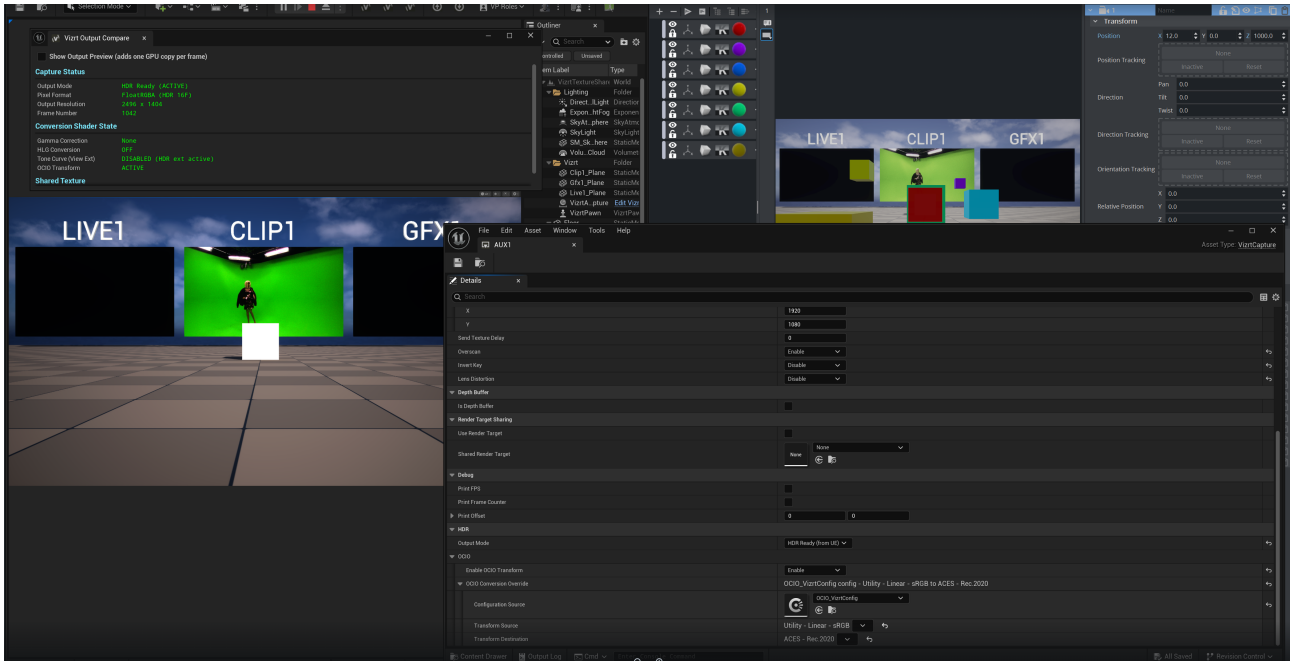


Linear - sRGB to ACES - Rec. 2020 1000 nits



Linear - sRGB to ACES - Rec. 2020

Copyright © 2026 Vizrt


LIVE1
CLIP1
GFX1
Vizrt Output Compare
Capture Status
Conversion Shader State
Shared Texture
Outliner
Transform
AUX1
Details

24.2 FBX Files

Viz Artist's built-in FBX Importer reads Autodesk's exchange file format for 3D assets. FBX files (*.fbx*) are normally saved in a binary (or native) format, but they can also be saved in ASCII format. Binary FBX files and ASCII FBX files both use the *.fbx* filename extension.

Which features of a scene, that can be carried over to a Viz Scene, depends on:

- **FBX support of a 3D modeling application:** The FBX file format supports many common features of a 3D scene. But not every feature offered by the major 3D modeling packages can be covered by an interchange format.
  Refer to the documentation of the 3D modeling application to discover which features can be exported to the FBX format.
- **Availability of the contained features in Viz Engine:** Viz Engine's feature is optimized for real-time. Features which are not available in Viz Engine cannot be imported and are ignored.

Note: Certain effects can be technically possible in Viz Engine but would require an elaborate conversion which may not be available.

To import an FBX file to Viz Artist, see Import of Files and Archives.

This section contains information on the following topics:

- Scene Graph Traversal
  - Scene Graph Traversal, Child Nodes First
  - FBX Node Types and How they are translated into Viz Scene Tree Items
- Transformations
  - Avoid Transformations Being Baked
  - FBX Import: Avoid Separate Scaling Pivot
  - Avoid Transformations Being Combined
- Export FBX Files from 3ds Max and Maya
  - To Export FBX Files from 3ds or Maya
  - Units
  - Maya Units Imported into Viz Artist
  - Materials
  - Materials with Viz Classic Render Pipeline
  - Lambert Shading
  - Phong Shading
  - Other Materials
  - Materials with Viz Engine Render Pipeline
  - Stingray PBS
  - Other Materials
  - Textures with Viz Classic Render Pipeline
  - Camera
  - Lights with Viz Classic Render Pipeline
  - Supported Lights
  - Spotlights
  - OpenGL Spotlight Parameters and Corresponding Light Parameters in Viz

Copyright © 2026 Vizrt

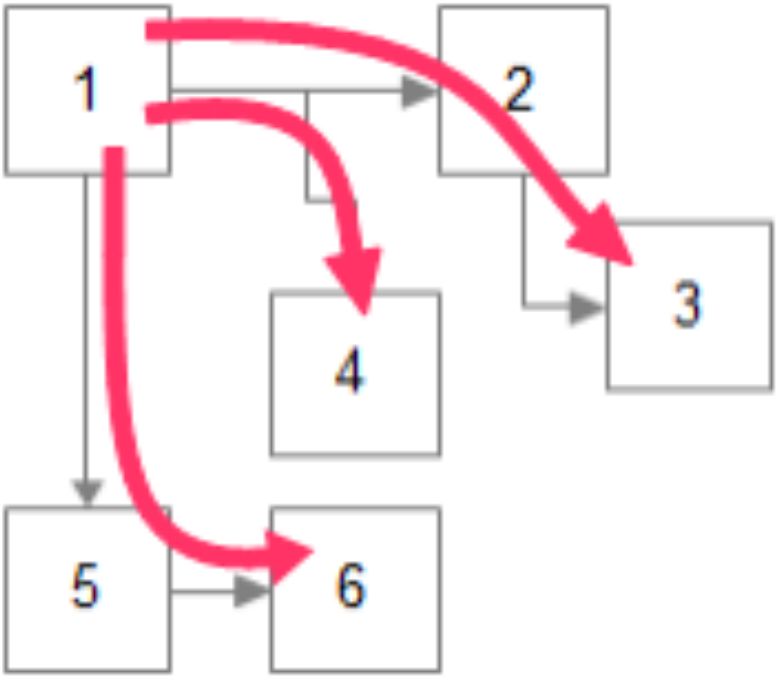
- Exponent in Maya
- Lights with Viz Engine Render Pipeline
- Reimporting Maya FBX Files
  - VizUUID Plug-in Installation

24.2.1 Scene Graph Traversal

The Scene representation in the form of a Scene Graph is common to all 3D applications, the FBX file format is also built this way. When an FBX file is imported to Viz the Scene Graph is traversed node by node. All child nodes are traversed before advancing to the next sibling node.

This is how FBX nodes are transformed into Viz containers:

Scene Graph Traversal, Child Nodes First



FBX Node Types and How they are translated into Viz Scene Tree Items

FBX Node	Child Nodes with Viz Classic Render Pipeline	Viz Tree with Viz Classic Render Pipeline	Child Nodes with Viz Engine Render Pipeline	Viz Tree with Viz Engine Render Pipeline
Empty	Imported	Group	Imported	Group

Copyright © 2026 Vizrt

FBX Node	Child Nodes with Viz Classic Render Pipeline	Viz Tree with Viz Classic Render Pipeline	Child Nodes with Viz Engine Render Pipeline	Viz Tree with Viz Engine Render Pipeline
Camera	Ignored	Imported into global camera	Ignored	Imported into global camera
Light	Ignored	Imported into global light	imported	Imported as Light plug-in
Mesh	Imported	Geometry	Imported	Geometry
Other	Imported	Group	Imported	Group

24.2.2 Transformations

The computation of transformations is more streamlined in Viz Engine than in FBX because the focus is on real-time graphics. To limit computing time spent on transformations the number of parameters incorporated in computing the final transformation matrix has deliberately been kept low in Viz. In contrast, in a 3D modeling application the number of parameters incorporated in computing the final transformation matrix is high, to give maximum flexibility.

Note: If unsupported transformation features are contained in a FBX scene, Viz Engine tries to provide them by combining several transformations into one (maintaining the transformation order). This limits animation and can only work to a certain degree, after which Viz Engine falls back to baking the imported transformation onto the imported vertices.

- **Baking transformations:** This is the fallback which may be chosen for imported nodes as necessary. It implies computing the transformation matrix with the FBX formula and to apply the result to the imported vertices.
  - **The good:** To Bake the imported transformations onto the imported vertices basically means that arbitrarily complex transformations can be imported into Viz Engine.
  - **Drawback:** Any information on how the result of a complex transformation has been obtained is discarded through baking. In particular individual items of the transformation cannot be animated anymore.

Avoid Transformations Being Baked

If unsupported features are detected the transformations of those nodes are Baked, which means that individual transformations are not retained in a Viz Scene. The transformed vertices are in the Viz Scene, but with all transformation parameters reset to their default values.

To avoid this make sure that unsupported features are not used in a Scene and, ideally, that the FBX properties which import into the same transformation in a Viz Scene are not used at the same time:

- **Do not use a scaling offset:** An extra offset applied after rotation and before scaling. It depends on the modeling application if it is available and under which name. To be on the safe side try to limit this to the regular translation and the rotation pivot.

Copyright © 2026 Vizrt

- **Do not use a separate scaling pivot:** This can be tricky, depending on the 3D modeling application. In Maya the two pivots move together by default, which keeps the possibility to map them both to a single axis center in Viz Engine. If the modeling application moves the rotation pivot separately by default, and objects are scaled afterwards, there are two transformations based on different pivots. Viz Engine has to bake the results to reflect them correctly.

FBX Import: Avoid Separate Scaling Pivot

Rotation and scaling settings	State	Result
Rotation and scaling pivot identical	Object rotated and/or scaled	Ok, fully retained in a Viz Scene
Rotation and scaling pivot different	Object not scaled	Ok
	Object scaled	Baked

Note: More complex transformation hierarchies can be built and multiple pivots simulated by placing the objects in groups and apply additional transformations to the group or groups. This is reproduced in the scene tree when importing to Viz Engine.

Avoid Transformations Being Combined

If, during import, two FBX properties have to be combined into a single property in Viz, they cannot be animated separately anymore. Limitations are not fully specified yet, since animation import is still under development, but to keep 100% the same control over transformations as in the original scene it is a good idea to consider these tips:

- Do not use a rotation offset.
- Do not use geometric translation/rotation/scaling in 3ds Max.
- Do not use unit conversion (see Export FBX Files from 3ds Max and Maya).
- Do not use axis conversion.
  - FBX uses a Y-up axis system, this is also the default setting for the FBX export. If the modeling application uses a different axis system it applies pre-rotation of ± 90° to all objects to change the orientation of the Scene to match the FBX axis system. This can interfere with animation.
  - Maya uses a Y-up axis system by default so all pre- and post-rotations are set to zero, which is perfect. If the modeling application uses another axis system (for example, 3ds Max uses Z-up) and problems occur it is possible to select a different option in the 'Axis Conversion' section in the FBX export settings, so that no corrective rotation has to be applied.

24.2.3 Export FBX Files from 3ds Max and Maya

To Export FBX Files from 3ds or Maya

1. From the File menu in Maya 2013, select **Export (All)...** or **Export Selection...**.

Copyright © 2026 Vizrt

2. Select a directory and enter a filename.

Note: Make sure that FBX is selected as the export file type to make export options show.

Units

It depends on each modeling application how units in virtual 3D space correspond to real world units. The default working unit is centimeters. When a Scene is exported choose a scale factor in the FBX export dialog, the option is named **File units converted to** and can be found in section **Advanced Options > Units**.

The table below shows how units import from Maya into Viz Artist when different settings are used:

Maya Units Imported into Viz Artist

Maya Working Units	Object size in Maya	File units converted to	Size in Viz Artist	Scaling factor
Centimeter	6	Centimeters	6	1
Centimeter	6	Millimeters	60	10

Caution: It is recommended that Scenes are built so that a scaling factor of `1`, or, if this is not possible, to scale the Scene up in Viz Artist after it is imported. **The reason:** If scaling is enabled during export Maya automatically alters the Scene to preserve the hierarchical transformation results. In some cases, this causes problems (for example, creating scaling pivot points, which are not supported in Viz).

Materials

Viz Engine supports one material per mesh node. The first material found on a FBX node is imported into a Material object, which is then applied to the Viz Container.

Note: Multiple or layered materials are not supported.

Materials with Viz Classic Render Pipeline

Supported FBX material classes are Lambert Shading and Phong Shading. Both types of shading have there own set of parameters which are described in more detail in the tables below:

Lambert Shading

FBX	3ds	Maya	Viz
Ambient Color	Ambient	Ambient Color	Ambient

Copyright © 2026 Vizrt

FBX	3ds	Maya	Viz
Diffuse Color	Diffuse	Color	Diffuse
Diffuse Factor	-	Diffuse	Diffuse Color Attenuation
-	-	-	Specular =0,1%
Emissive Color	Self-Illumination	Incandescence	Emission
Transparency Factor/Color	Opacity (%)	Transparency (Color)	Alpha

Phong Shading

FBX	3ds	Maya	Viz
Ambient Color	Ambient	Ambient Color	Ambient Color
Diffuse Color	Diffuse	Color	Diffuse Color
Diffuse Factor	-	Diffuse	Diffuse Color Attenuation
Emissive Color	Self-Illumination	Incandescence	Emission Color
Specular Color	Specular	Specular Color	Specular Color
Specular Factor	Specular Level	-	Specular Color Attenuation
Shininess	Glossiness	Cosine Power	Shininess
Transparency Factor/ Color	Opacity (%)	Transparency (Color)	Alpha

Other Materials

Other kinds of material which the modeling application supports may be mapped to a FBX Lambert or Phong material by the exporter, at least the parameters which are compatible to those material types.

Materials with Viz Engine Render Pipeline

The main material used in the Viz Engine Render Pipeline is the PBR Material. The importer either imports "Stingray PBS" materials from Maya or tries to derive some basic values from any other Material available in the *.fbx* file.

Copyright © 2026 Vizrt

Stingray PBS

This is the preferred material to be used when using Maya in combination with Viz Engine, because the channels directly map from Maya to Viz Engine.

Stingray PBS	PBR Material

`Use Color Map`
`Color Map`
`Base Color`
	If `Use Color Map` is enabled, then the `Color Map` is imported as a Texture on the `Base Color` Texture slot.
	If `Use Color Map` is disabled, the `Base Color` is imported as the `Base Color`. `Base Color` animations are also imported when available.

`Use Metallic Map`
`Metallic Map`
`Metalness`
	If `Use Metallic Map` is enabled, then the `Metallic Map` is imported as a Texture on the Metallic Texture slot.
	If `Use Metallic Map` is disabled, `Metalness` is directly imported as Metallic Factor.

`Use Roughness Map`
`Roughness Map`
`Roughness`
	If `Use Roughness Map` is enabled, then the `Roughness Map` is imported as a Texture on the `Roughness` slot.
	If `Use Roughness Map` is disabled, `Roughness` is directly imported as Roughness Factor.

`Use Emissive Map`
`Emissive Map`
`Emissive Color`
	If `Use Emissive Map` is enabled, then the `Emissive Map` is imported as a Texture on the Emissive Texture slot.
	If `Use Emissive Map` is disabled, the `Emissive Color` is imported as the `Emissive Color`. Color animations are also imported when available.

`Emissive Intensity`	Intensity value of the Emissive channel.
`Use Normal Map` `Normal Map`	If `Use Normal Map` is enabled, then the `Normal Map` is imported as a Texture on the Normal Texture slot If `Use Normal Map` is disabled, the slot is left as empty.
`Use Ao Map` `Ao Map`	If `Use Ao Map` is enabled, then the `Ao Map` is imported as a Texture on the Ambient Occlusion Texture slot. If `Use Ao Map` is disabled, the slot is left empty.

Copyright © 2026 Vizrt

Other Materials

For all materials where *.fbx* is able to create a so called Surface Material (usually Lambert and Phong), a PBR Material is created, where a few basic parameters are imported.

Surface Material	PBR Material
`Diffuse` `Diffuse Factor`	If a texture is assigned on the diffuse channel, it is imported onto the Base Color texture slot. The diffuse color is imported as Base Color. If a Diffuse Factor is available, the Diffuse Color gets pre-multiplied with it. Diffuse color animations are also imported when available.
`Emissive` `Emissive Factor`	If a texture is assigned on the emissive channel, it is imported onto the Emissive Color texture slot. The emissive color is imported as is. If an Emissive Factor is available, it is imported as the Intensity of the Emissive slot. Emissive color animations are also imported when available.
`Ambient`	If a texture is assigned on the ambient channel, it is imported onto the Ambient Occlusion texture slot.

Textures with Viz Classic Render Pipeline

FBX can associate a texture with each color channel on a material.

> **Note:** Currently one texture for the diffuse color channel is supported.

Textures are referenced by the FBX file through their path and filename by default. This feature is broken in the current FBX SDK provided by Autodesk (the exported file may contain an absolute path and in some cases this path is not correct, even on the machine where the file was created). Texture images can be embedded into the FBX file which is a lot more robust. Tick the check-box Embed Media in the export dialog to enable this.

> **Note:** Regardless of the export parameters chosen, the FBX file can still contain incorrect texture references. If texturing problems occur use a free FBX viewer (for example, the QuickTime plug-in) to check if the texture import really failed or if there is a problem with the Viz import.

Camera

Currently, up to 16 cameras are imported. The import is limited and can give little more than the correct camera position and direction. Dependent on the transformations that have been applied to a camera, and additional parameters unique to the modeling application, the camera position and viewing parameters can deviate a lot after import.

Copyright © 2026 Vizrt

Lights with Viz Classic Render Pipeline

The lights detailed below are supported, but not all their parameters are imported:

Supported Lights

	Lights
3ds	Target Spot
	Free Spot
	Target Directional (Switch to 'Overshoot' to make them resemble Viz Artist's Infinite lights)
	Directional (Switch to 'Overshoot' to make them resemble Viz Artist's Infinite lights)
	Omni (Switch to 'Overshoot' to make them resemble Viz Artist's Infinite lights)
Maya	Directional Light
	Spot Light
	Point Light

The only animation properties imported for Lights are:

- **Translation**
- **Cutoff / Cone Angle** (because hardware accelerated OpenGL lights are limited, this is unlikely to produce the exact same effect produced in the modeling software)
  Color animation is not imported because it requires separate channels for red, green and blue components.

Spotlights

For performance reasons Viz Engine only supports the standard OpenGL light parameters.

OpenGL Spotlight Parameters and Corresponding Light Parameters in Viz

OpenGL	Viz	Description
Cutoff	Cone Angle	Specifies the angle between the axis of the light cone and a ray along the edge of the cone. Can also be named 'outer cone angle'.

Copyright © 2026 Vizrt

OpenGL	Viz	Description
Exponent	Penumbra	Controls how concentrated the light is. The light's intensity is highest in the center of the cone. It's attenuated toward the edges of the cone by the cosine of the angle between the direction of the light and the direction from the light to the vertex being lit, raised to the power of the spot exponent.

FBX translates a modeling application's spotlight parameters to an inner and outer cone angle. During import into Viz the exponent is approximated based on the inner angle of the spotlight. Results can be different from what is seen in Maya or 3ds.

Exponent in Maya

Maya has an alternative spotlight parameter, **Dropoff**, which corresponds directly to the OpenGL parameter **Exponent**. With the Viz Artist VizFbxExtension plug-in, this parameter can be exported. This allows work with real GL-spotlight parameters in Maya and gives a viewport lighting much more faithful to the actual rendering results in Viz. Without the VizFbxExtension plug-in, Maya does not export that parameter, because no other Autodesk product supports importing it.

Lights with Viz Engine Render Pipeline

In contrast to the Viz Classic Render Pipeline, lights are now directly imported in the Viz Engine Render Pipeline as container plug-in. The following light types are supported:

FBX	Viz	Attributes
Point Light	Point Light	Color Intensity
Spot Light	Spot Light	Color Intensity Inner Cone Angle Outer Cone Angle
Directional Light	Directional Light	Color Intensity
Area Light	Area Light	Color Intensity

24.2.4 Reimporting Maya FBX Files

When you change a Maya scene that is already in use in Viz Artist, you can reimport the Maya scene without losing changes made in your Viz Artist scene. When reimporting scenes, changes in the *.fbx* file and the Viz Artist scene are combined. This requires that you install a separate plug-in for Maya, which is included with the Viz Artist installer. After installing the plug-in, activate the *VizUUID.py* plug-in in Maya. When the plug-in is active, you find a new

Copyright © 2026 Vizrt

section called **Viz UUID** in the **Extra Attributes** tab when exporting from Maya. Based on this UUID, Viz Artist determines which scenes or objects to reimport. Reimport requires that the UUIDs are identical.

1. Create a scene in Maya and export it as *.fbx*.
2. Import the scene into Viz Artist.
3. Add different features to the Viz Artist scene, such as animations, textures, or effects. Then save the scene.
4. Make some changes to the original scene in Maya, then export it again using **Export as** or **Export** and replace the previous file.
5. Open the **Import** pane in Viz Artist. Reimport the changed *.fbx* file by first selecting the scene to import. Then, drag the target scene where the changes are imported to, to the **Re-import Scene** field.
6. In this example, a changed *.fbx* file has been imported to the original Viz scene, where multi-texturing and light changes have been applied in Viz Artist:

VizUUID Plug-in Installation

- When Installing Viz, make sure you have the **Maya FBX Export** component selected.
- This installs all necessary files into the Viz Engine subfolder *export/FBX*. Standard installation path is *C:\Program Files\vizrt\VizEngine\export\FBX*.
- Adjust your `MAYA_PLUGIN_PATH` so that it points to the folder above. Please refer to your Maya documentation on how to do this.

See Also

- Import of Files and Archives
- Export FBX Files from 3ds Max and Maya

Copyright © 2026 Vizrt

24.3 CINEMA 4D

This section details how to work with CINEMA 4D and Viz Artist.

The CINEMA 4D and Viz Artist integration imports complete geometries, lights, and animations, from CINEMA 4D to Viz Artist. Viz Artist imports scenes from CINEMA 4D without the need for a local installation of CINEMA 4D on the Viz Artist workstation. 3D objects are imported either as their direct primitive object counterpart in Viz Artist, or as meshes.

Viz Artist integrates a CINEMA 4D LiveLink tool that lets you make changes to a scene in CINEMA 4D. When Viz Artist is in On Air Mode, it updates live based on your changes. The CINEMA 4D LiveLink tool is able to update the position, rotation and scaling properties of all objects in a Scene.

IMPORTANT! The license required for the import of CINEMA 4D Scenes is issued by Vizrt.

This section contains the following topics:

- System Requirements and Configuration
  - To Enable Correct File Format
    - Cinema 4D R15-R20
    - Cinema 4D R21
    - Cinema 4D R23 or Later
- Cinema4D Substance FBX Workflow
  - Modifying Substance Settings
- Cinema4D LiveLink
  - To Install CINEMA 4D LiveLink During Viz Artist Installation
  - To Install CINEMA 4D LiveLink to an Existing Viz Artist Installation
  - Troubleshooting CINEMA 4D LiveLink Installation
  - To Open a LiveLink
- CINEMA 4D Integration Limitations
  - Cameras
  - Lights
  - Materials and Textures
  - General Limitations
  - Acknowledgements

24.3.1 System Requirements and Configuration

The following are the minimum requirements to import CINEMA 4D Scenes to Viz Artist:

- Viz Artist
- **License:** Maxon CINEMA 4D license
- **To Import:**
  - **Viz Artist:** Version 3.7 or higher
  - **License:** CINEMA 4D import 1 license

Refer to the following version compatibility table:

Copyright © 2026 Vizrt

Viz Artist	CINEMA 4D
5.5	
5.4	
5.3	
5.2	Version R23, R25.120 2023, R26
5.1	Version R23, R25.120 2023, R26
5.0 (1)	Version R23, R25.120 2023
4.4	Version R23
4.0, 4.1, 4.2, 4.3	Version R20
3.12, 3.13, 3.14	Version R19
3.10, 3.11	Version R18
3.9.0, 3.8.3, 3.8.2, 3.8.1	Version R17
3.8.0, 3.7.2, 3.7.1, 3.7.0	Version R15 **IMPORTANT!** If the CINEMA 4D version is R15 <= SP2, do the procedure detailed in *<viz install folder>\r15_sp2_fix* after installation.
(1) use R23 plugin directory for R25 and 2023	

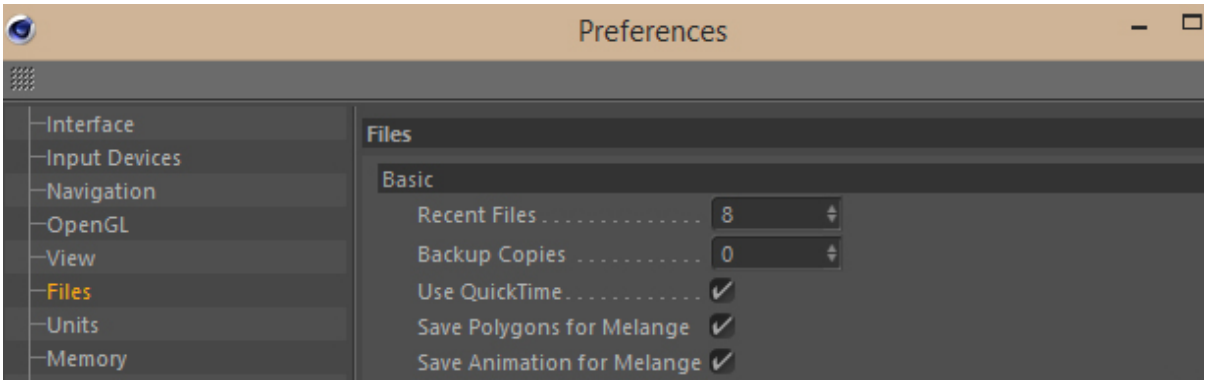
To Enable Correct File Format

To make sure that CINEMA 4D saves all the information Viz Artist requires when importing from *.c4d* files, enable the following settings in CINEMA 4D's preferences.

Cinema 4D R15-R20

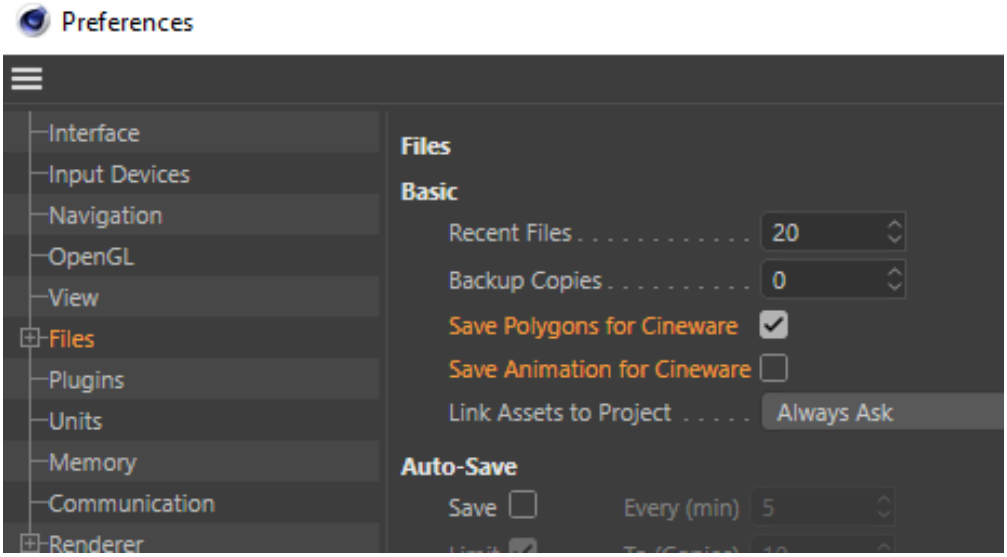
1. Open CINEMA 4D.
2. Choose **Edit > Preferences > Files**.
3. Check the **Save Polygons for Melange** checkbox. If required, check the **Save Animation for Melange** checkbox. This setting generates Key Frames for animations that CINEMA 4D's MoGraph toolset produces, by baking the key frames into the timeline. This can have a performance impact when playing out the scene in Viz Artist.

Copyright © 2026 Vizrt



Cinema 4D R21

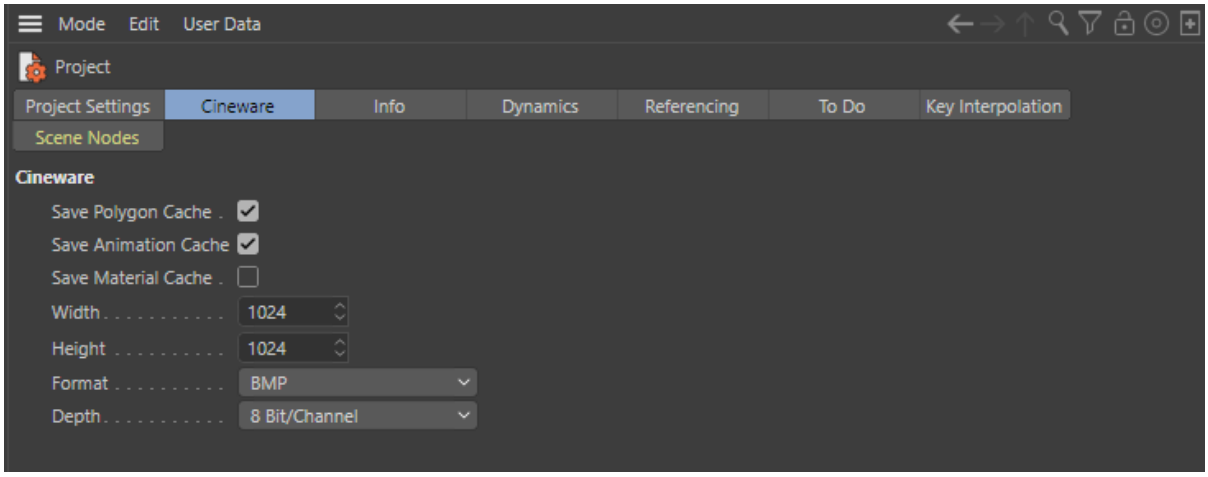
1. Open CINEMA 4D.
2. Choose **Edit > Preferences > Files**.
3. Check the S**ave Polygons for Cineware** checkbox. If required, check the **Save Animation for Cineware** checkbox. This setting generates Key Frames for animations that CINEMA 4D's MoGraph toolset produces, by baking the key frames into the timeline. This can have a performance impact when playing out the scene in Viz Artist.



Cinema 4D R23 or Later

1. Open CINEMA 4D.
2. Choose **Edit > Project Settings.**
3. Switch to the **Cineware** Tab.
4. Check the **Save Polygons Cache** checkbox. If required, check the **Save Animation Cache** checkbox.

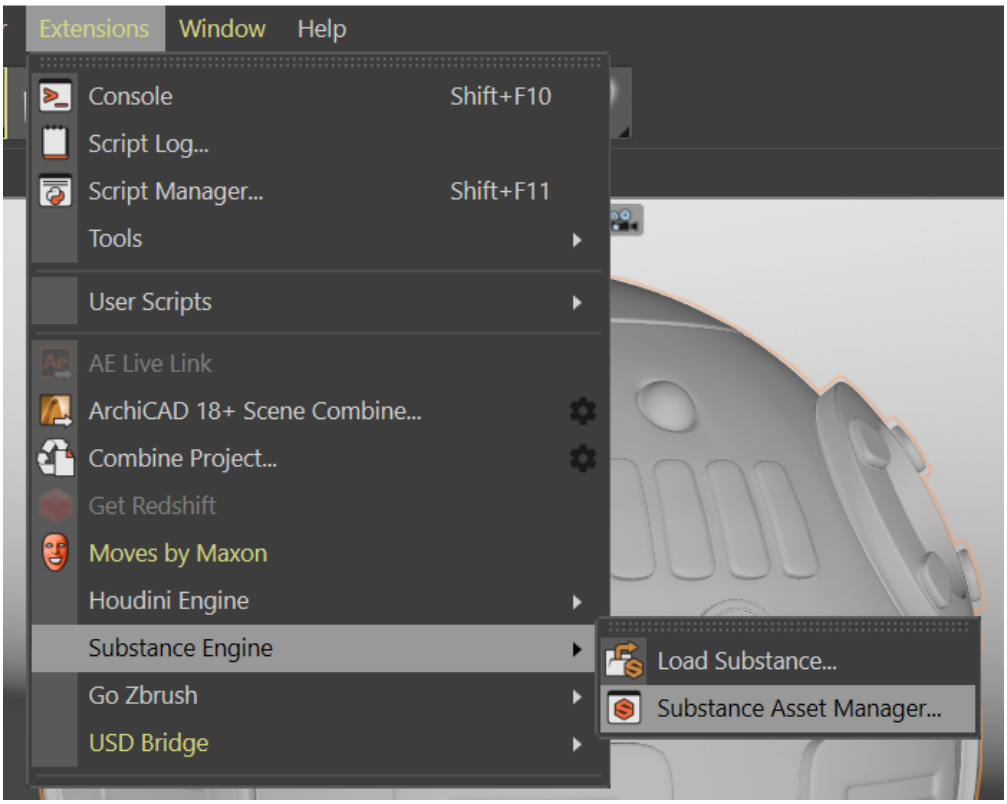
Copyright © 2026 Vizrt



24.3.2 Cinema4D Substance FBX Workflow

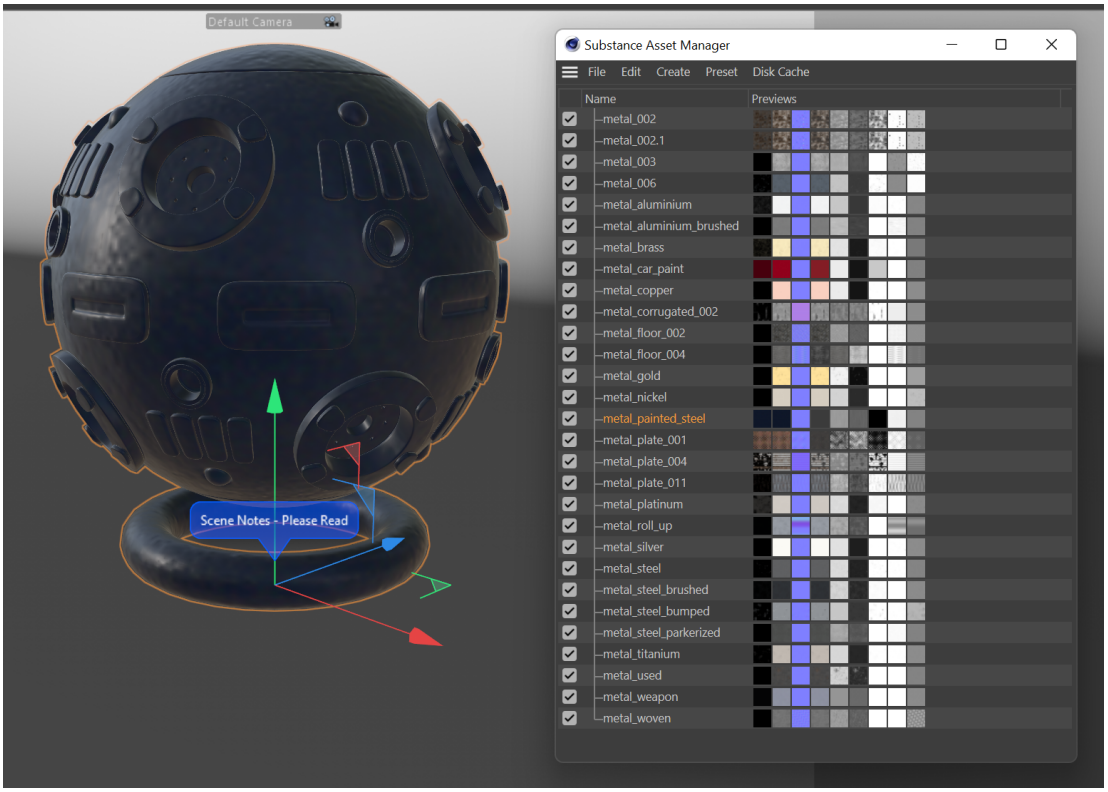
Starting with Viz Engine version 5.0, Viz Engine is capable of importing FBX files with Substance surfaces and translates them to our PBR rendering. You can use all your existing Substances in your Cinema4D, export as an FBX file and import them afterwards as Scene in the Viz Engine Render Pipeline of Viz Engine 5 or later.

- Create or load your Cinema4D project.
- In **Extensions > Substance Engine**, load your SBAR file.

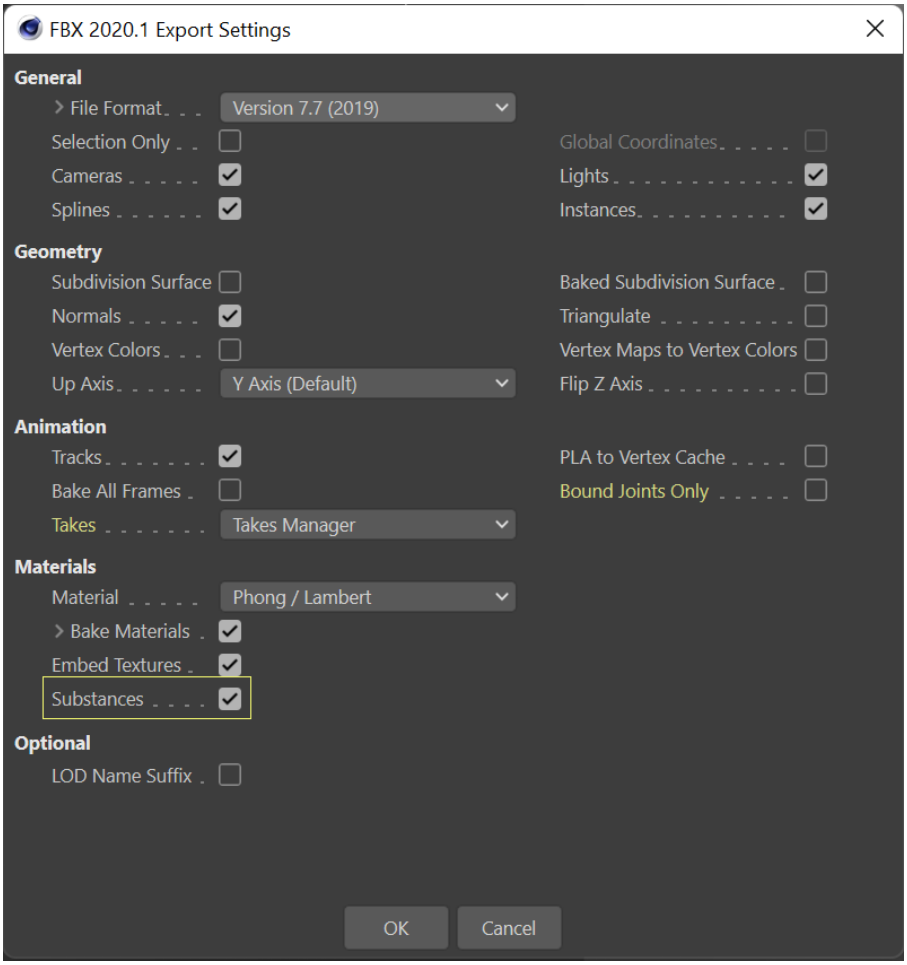


- This imports your SBAR file(s):

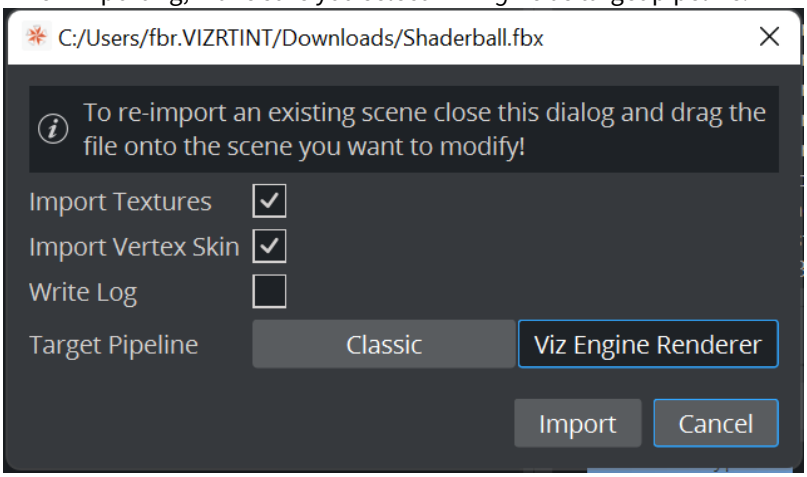
Copyright © 2026 Vizrt



- All of these materials can now be used on your geometries by dragging and dropping them onto your meshes.
- Select **Export > FBX**:

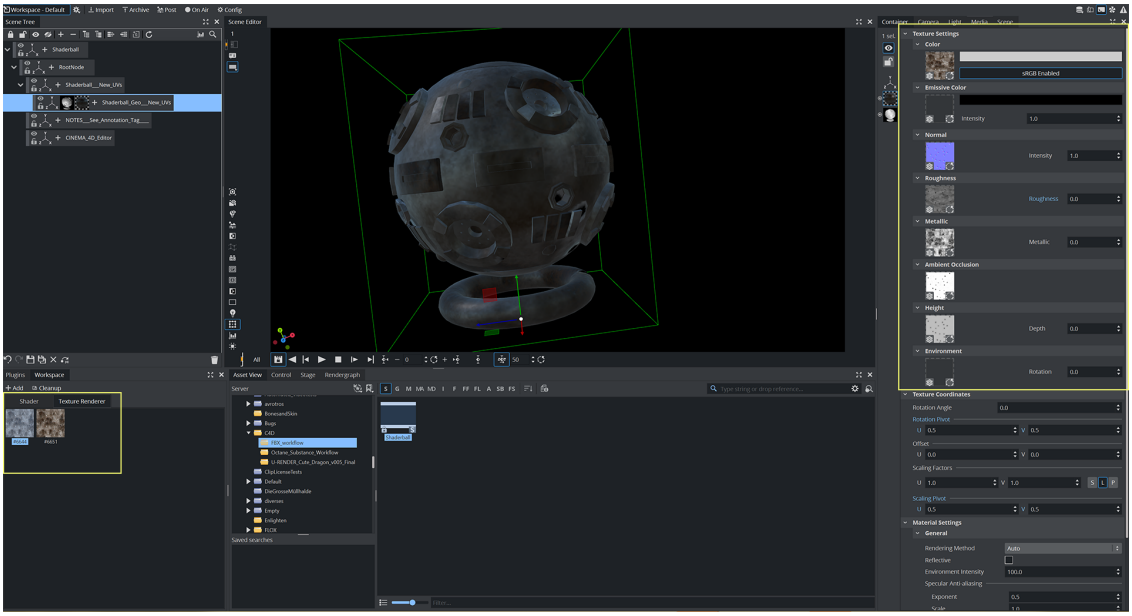


- Make sure, that *Substance* is selected.
- Save the file as FBX as seen in the screenshot above.
- Switch to Viz Artist and go to **Import** and select *Scenes*. Your FBX is imported as a standalone scene.
  - When importing, make sure you select *Viz Engine* as target pipeline.



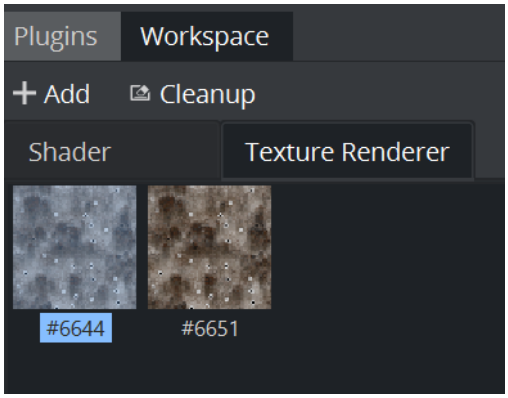
  - Once the import has finished, open the scene. Your substance textures have been mapped correctly to a newly created PBR material in Viz Engine, the textures are also added to Workspace area:

Copyright © 2026 Vizrt



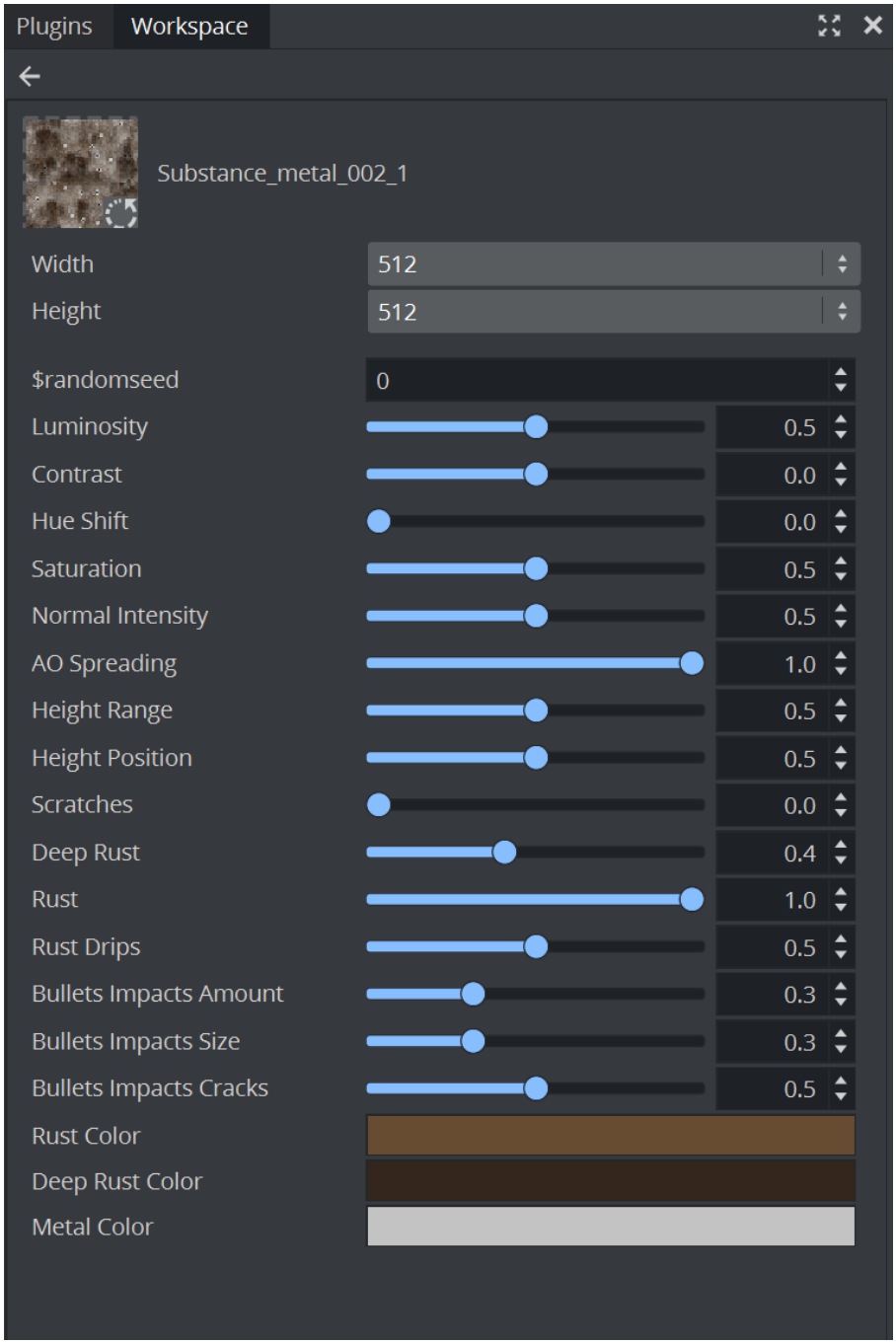
Modifying Substance Settings

Once you imported your Substance files into Viz Engine, they show up in the **Workspace** Area:



Double clicking on one of the Workspace items opens the exposed settings:

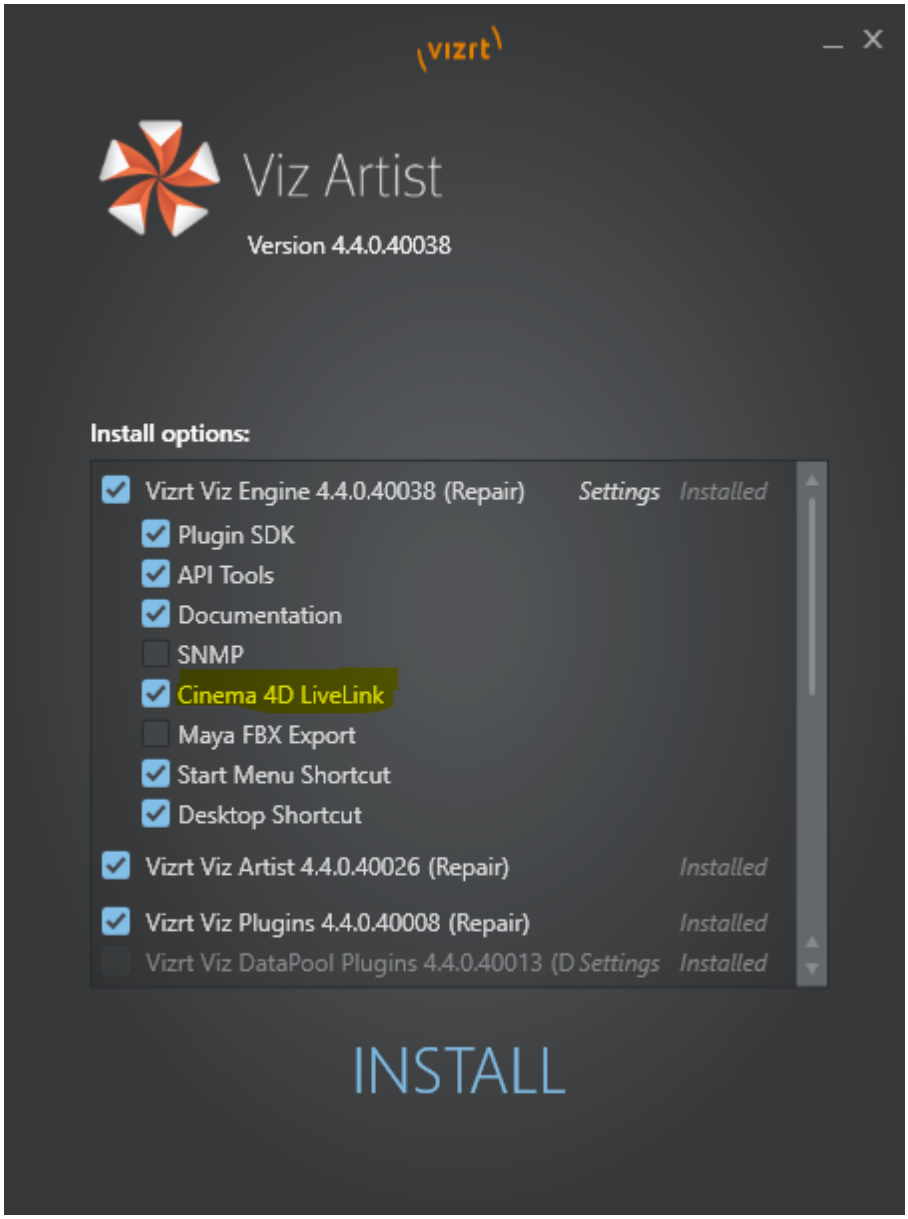
Copyright © 2026 Vizrt



24.3.3 Cinema4D LiveLink

To install CINEMA 4D LiveLink, you must first install CINEMA 4D. The Viz Artist installation files include the CINEMA 4D LiveLink tool, but it is not selected for installation by default.

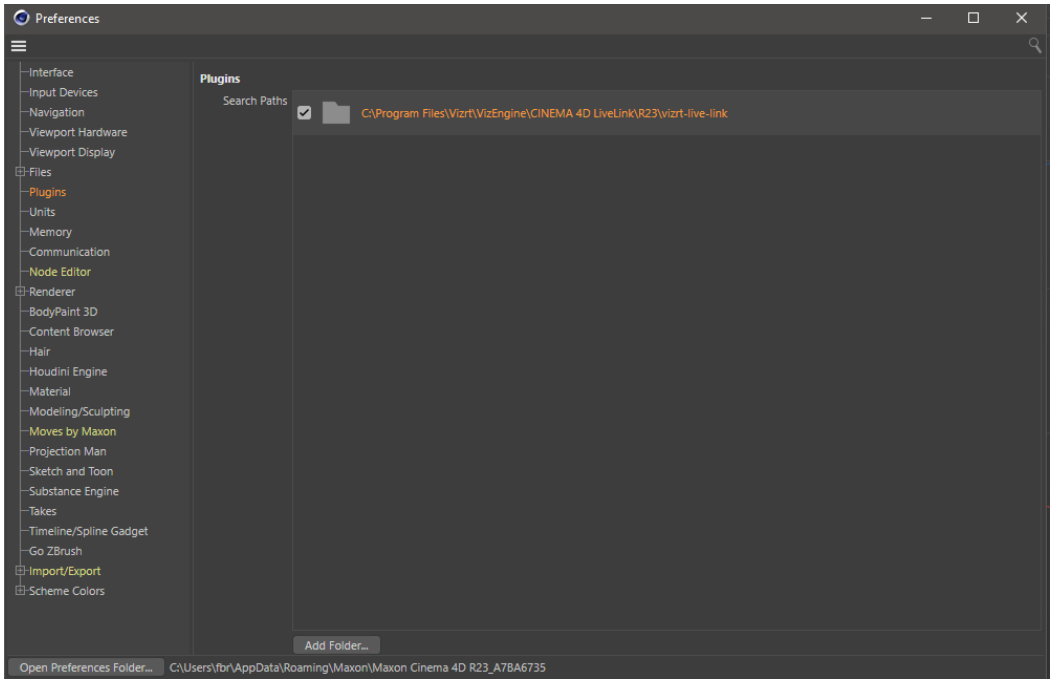
Copyright © 2026 Vizrt



To Install CINEMA 4D LiveLink During Viz Artist Installation

- For Cinema 4D R20 or newer.
  - Complete the Viz Engine Installation.
  - There is a prompt at the end of the installation reminding the user to manually add the plug-ins.
  - Install CIMEMA 4D.
    - For Cinema 4D R20 to R22, add the directory <viz install folder>\CINEMA 4D LiveLink\R17 as plugin directory in Cinema 4D (see https://www.plugins4d.com/installing).
    - For Cinema 4D R23 or newer, add the directory <viz install folder>\CINEMA 4D LiveLink\R23 as plugin directory in Cinema 4D (see https://www.plugins4d.com/installing).

Copyright © 2026 Vizrt



To Install CINEMA 4D LiveLink to an Existing Viz Artist Installation

Compatibility: These steps are only compatible for CINEMA 4D R20 or older. See https://www.plugins4d.com/installing

1. Install CINEMA 4D.
2. Start the Viz Artist installation in **Repair Mode**. The Viz Artist installer looks for the CINEMA 4D installation folder, and copies the LiveLink files to the CINEMA 4D plug-in folder.

Example: *C:\Program Files\MAXON\CINEMA 4D R17\plugins\vizrt-live-link*

After installing the LiveLink files, they remain in the CINEMA 4D installation folder even if you remove the LiveLink feature later on.

Troubleshooting CINEMA 4D LiveLink Installation

If the installation displays the **CINEMA 4D LiveLink will not work** warning, check the following possible solutions:

- **The Cinema 4D installation location is not the default folder.**
  - Manually copy the LiveLink files to the Cinema 4D plug-in folder.
  - The LiveLink files are in the Viz Engine installation folder: *<viz install folder>\CINEMA 4D LiveLink\[R15|R17|R23]\vizrt-live-link.*
- **CINEMA 4D has not been installed prior to Viz Artist installation.**
  - Complete the Viz Engine installation.
  - Install CINEMA 4D.
  - Start the Viz Artist installer again and select **Repair mode**. This makes a copy of the LiveLink files to the CINEMA 4D plug-in folder.

Copyright © 2026 Vizrt

- **For Cinema 4D R20 or newer:**
  - Complete the Viz Engine installation.
  - Install Cinema 4D.
    - For Cinema 4D R20 to R22, add the directory *<viz install folder>\CINEMA 4D LiveLink\R17* as plugin directory in Cinema 4D (see https://www.plugins4d.com/installing).
    - For Cinema 4D R23 or newer, add the directory *<viz install folder>\CINEMA 4D LiveLink\R23* as plugin directory in Cinema 4D (see https://www.plugins4d.com/installing).

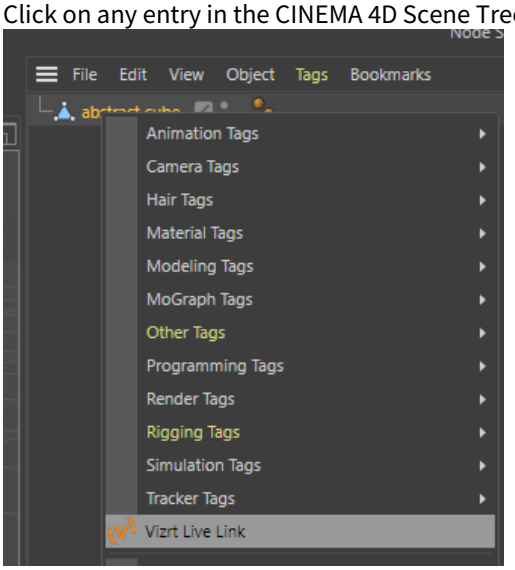
To Open a LiveLink

IMPORTANT! Open the same files in Viz Artist and CINEMA 4D. Because Viz Artist and CINEMA 4D communicate using UUIDs, opening different file versions causes the live link to fail.

1. Open Viz Artist and CINEMA 4D.
2. Import the project in Viz Artist (Import, Import Scene).

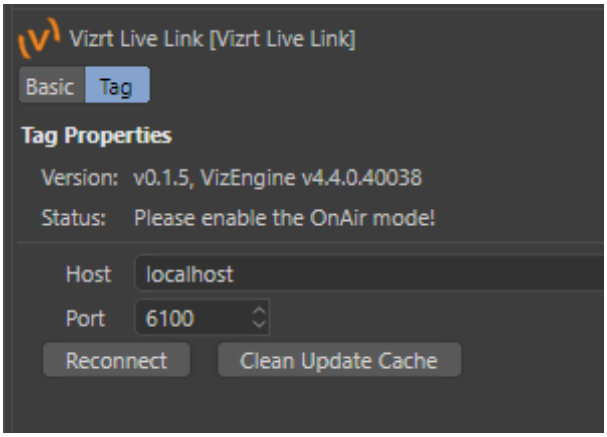
Note: Cinema4D import works for the Classic Render Pipeline only.

3. Set Viz Artist to On Air Mode.
4. In CINEMA 4D:
   a. Click on any entry in the CINEMA 4D Scene Tree.



   b. In the Basic Panel, click on **Enable.**
   c. Enter the Viz Artist/Engine **Host** name.
   d. Enter the Viz Artist/Engine **Port** number.

Copyright © 2026 Vizrt

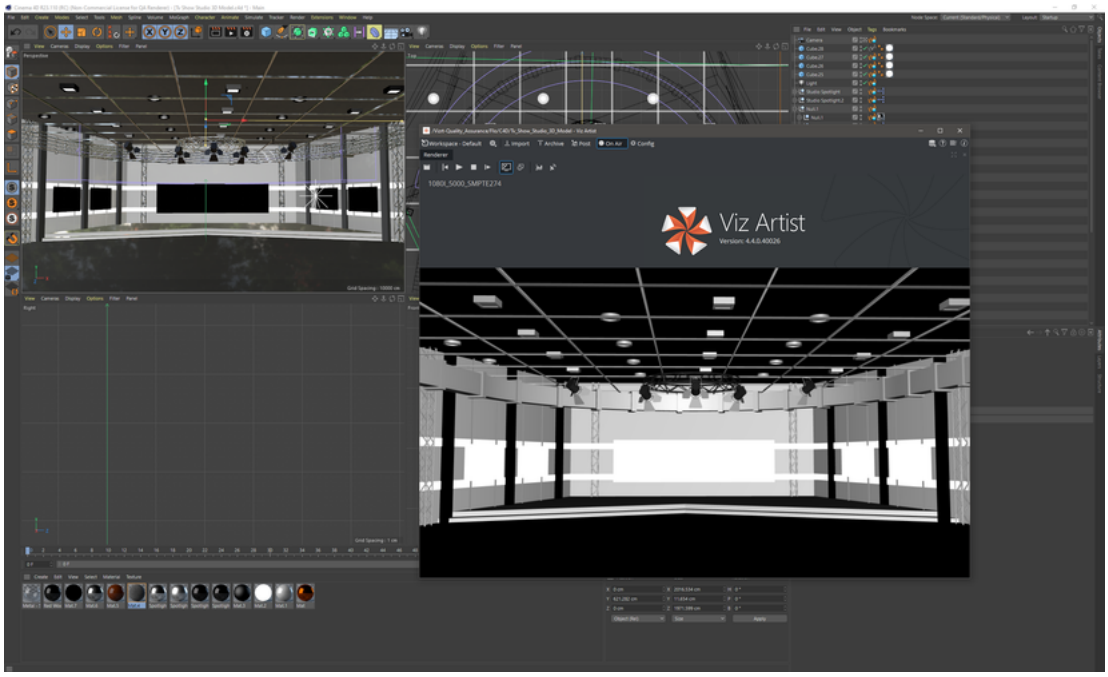


e. Click on **Reconnect.**
f. The Tag properties panel always shows the current connection status.

Icon	Status
	Disabled
	Enabled, but not connected
	Enabled and connected

Changes made to the scene in CINEMA 4D now reflects automatically in the scene in Viz Engine. The Live Link works only one way, from CINEMA 4D to Viz Artist. Moving objects in Viz Artist does not transfer back to CINEMA 4D.

Copyright © 2026 Vizrt



24.3.4 CINEMA 4D Integration Limitations

The following limitations apply to the CINEMA 4D and Viz Artist integration when importing CINEMA 4D scenes:

- LiveLink only updates changes in the position, rotation and scale of objects. It does not support changes in other parameters.

Cameras

Maximum number of cameras in the scene: 16, including the editor camera.

Lights

- Maximum number of lights in the scene: Eight. There is no support for a scale value other than `1.0` for X, Y, Z.
- Supported CINEMA 4D light types:
  - Omni
  - Spot
  - Distant
- Unsupported CINEMA 4D light types:
  - Parallel
  - Rectangular Spot
  - Parallel Spot
  - Parallel Spot Rectangular
  - Photometric
  - Tube

Copyright © 2026 Vizrt

- There is no support for:
    - Light intensity parameter
    - Shadows
    - Near/far clipping
    - Noise
    - Caustics
    - Lens Effects
    - Reflection
    - Visible Light

Materials and Textures

- Multi-texturing is not supported.
- The texture that is imported into Viz Artist is only the one on the Color channel of the last texture tag of an object (the one which shows on the top layer).
- Textures generated by a CINEMA 4D shader (and CINEMA 4D Shaders in general) are not supported.
- There is no support for:
    - Polygon selections
    - Sides
    - Seamless or Repetitions settings

General Limitations

- By default, Viz Engine does not render the backface of objects. To do this, add the Expert container plug-in.
- Sky, Environment, Background, Foreground, and Physical Sky objects, are not supported.
- The Floor Object is imported as a rectangle of the size 2000x2000, just as shown in the Editing mode in CINEMA 4D. After import, you can scale its size, as you require.

Acknowledgements

TV Studio model from c4ddownload.com.

Copyright © 2026 Vizrt

24.4 Adobe After Effects

Viz Artist can import Adobe After Effects project files directly, as it is capable of reading After Effects Project files (*.aep*) and storing its compositions as Scenes.

This section describes how to work with Adobe After Effects and Viz Artist. For the system requirements and integration setup, see System Requirements. The section contains the following topics:

- System Requirements
- Preparation of Your After Effects Project
- To Import After Effect Projects
- Frame Accurate Output

24.4.1 System Requirements

The following are the minimum requirements to import After Effects scenes to Viz Artist:

IMPORTANT! You must install both Viz Artist and Adobe After Effects on the same machine to use the After Effects importer.
Make sure no license or login requests are blocking the startup of Adobe After Effects.

- Graphic Hub 2.4 or higher.
- Matrox video board.
- Adobe After Effects.

Refer to the following version compatibility table:

Viz Artist	Tested Adobe After Effects version
5.5	25.6.4
5.4	24.6.0 (2024), 25.3.0 (2025)
5.3	23.6.0 (2023)
5.1, 5.2	22.6.0 (2022)
4.4	CC 2020, CC 2021
4.3	CC 2020, CC 2021
4.0, 4.1, 4.2	CC 15.0
3.12, 3.13, 3.14	CC 15.0
3.9.0, 3.10, 3.11	CC 2017

Copyright © 2026 Vizrt

Viz Artist	Tested Adobe After Effects version
3.8.3	CC 2015.3
3.8.2, 3.8.1	CC 2015
3.8.0	CC2014
3.7.2, 3.7.1	CC
3.7.0	CS6

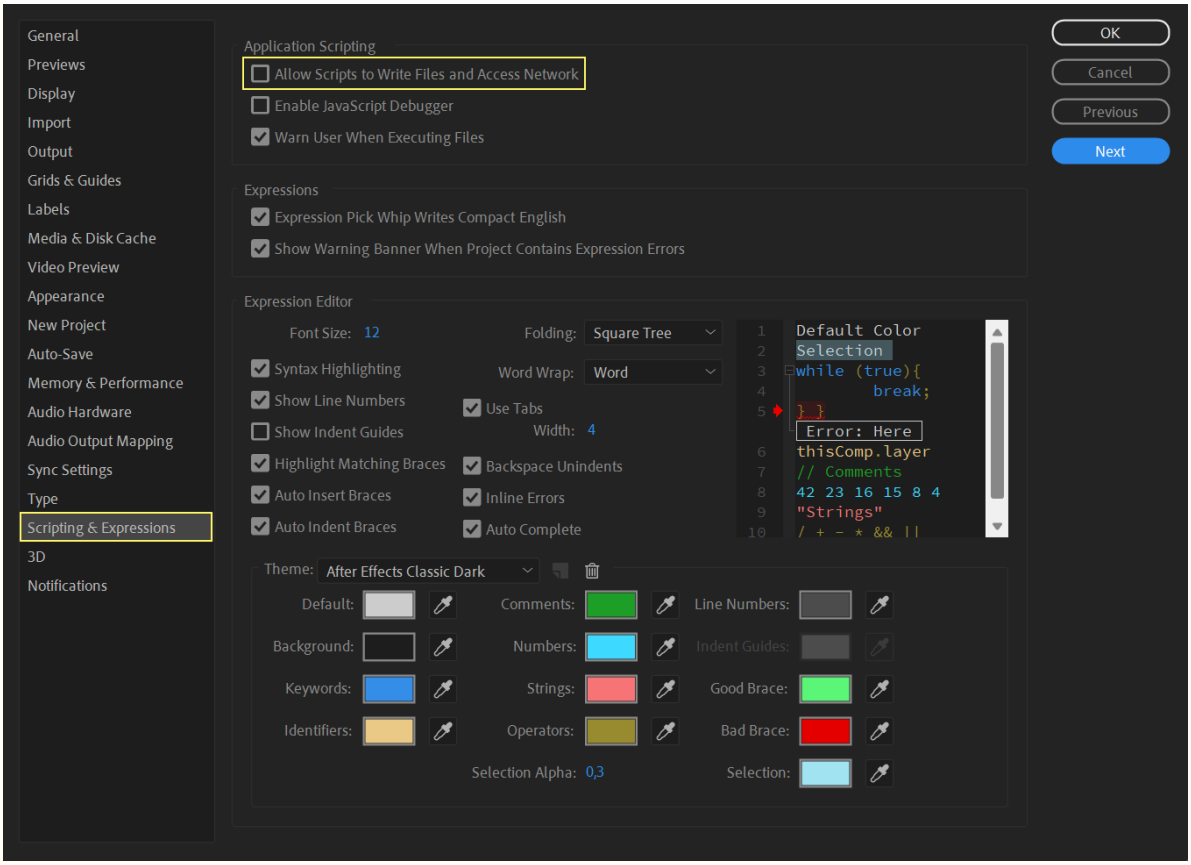
The Viz Engine installer copies the plug-in file, *AEExport.aex*, to the Adobe After Effects folder. If multiple versions of After Effects are found, the installer copies the plug-in to the highest Adobe After Effects version number. If the installer cannot find an existing installation of Adobe After Effects, it does not copy the plug-in to the system during installation.

Because of this, you must install Adobe After Effects before installing Viz Engine for the After Effect Import plug-in to work. If you have installed the software in the wrong order, uninstall Viz Engine and run the Viz Engine installer again. This installs the file, and resolves this issue.

In case, If installer does not copy the plug-in to the system, you can manually copy *AEExport.aex* file from *C:\Program Files\Vizrt\VizEngine\import\AE* to *C:\Program Files\Adobe\Adobe After Effects <version-id>\Support Files\Plug-ins*, and this resolves the issue.

Note: Newer versions of After Effects require to allow the execution of scripts:

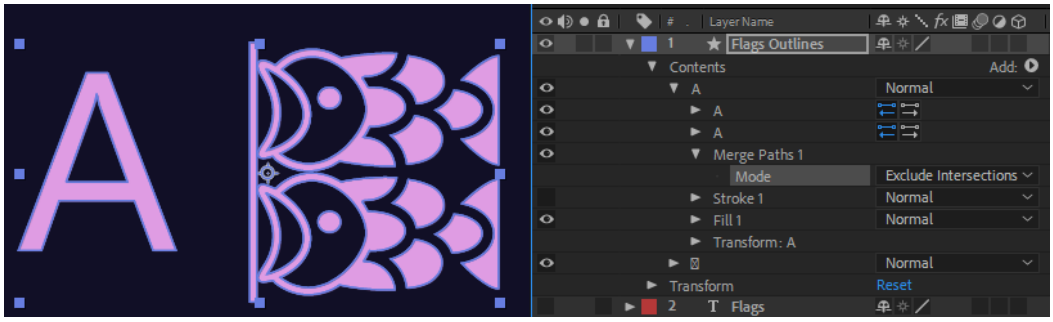
Copyright © 2026 Vizrt



24.4.2 Preparation of Your After Effects Project

1. Shape layer in After Effects can be imported to the scene as two types of objects, video and geometry plug-ins, depends on its complexities.
    a. Rectangle created by Rectangle or Rounded Rectangle tools are imported as a Noggi object.
    b. Ellipse created by Ellipse tool are imported as Eclipse object.
    c. Star and Polygon are imported as Star object if both of the following are true:
        i. Inner and Outer roundness is `0`.
        ii. Star point is integer number.
    d. Shape created by Pen tool are imported into Polygon object.
    e. Converted shape from text, each character of it contains multiple paths. The shape of each character merge all paths in a special mode (be decided by After Effects among Add, Subtract, Intersect and Exclude Intersections depends on the overlapping and direction of each path) that we not support it even most of character works with our simple merging method **Exclude Intersections**, manually change Merge Paths mode of all characters to "Exclude Intersections" can make this layer be import into Polygon object instead of video.
        i. Use Horizontal or Vertical Type Tool to create a text layer.
        ii. Right click on the text layer and select **Create Shapes from Text**.
        iii. Expand **Merge Path** and change Mode to **Exclude Intersections**.

Copyright © 2026 Vizrt



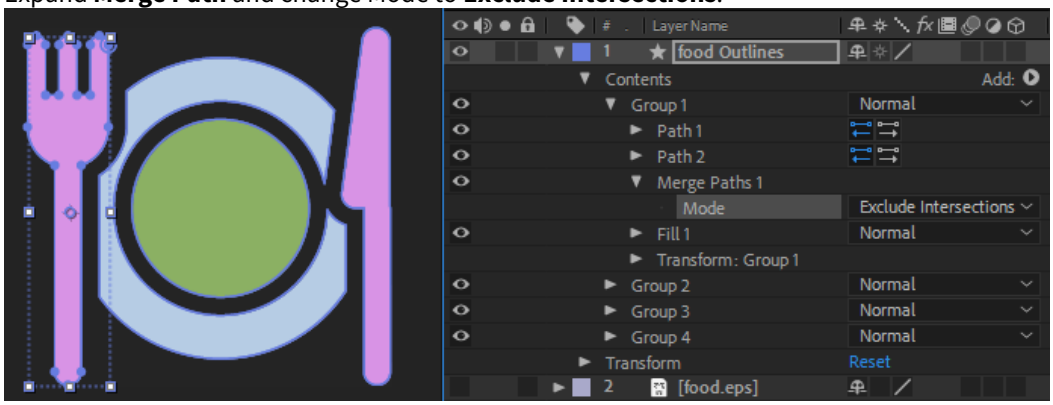
f. Convert shape from imported external vector graphic file such as *.eps*. This is similar to the converted text but it might have many groups of paths. To import an *.eps* file to After Effects and make it be able to imported as as Polygon object in Viz Artist:

i. Press **CTRL + I** and select a vector file.

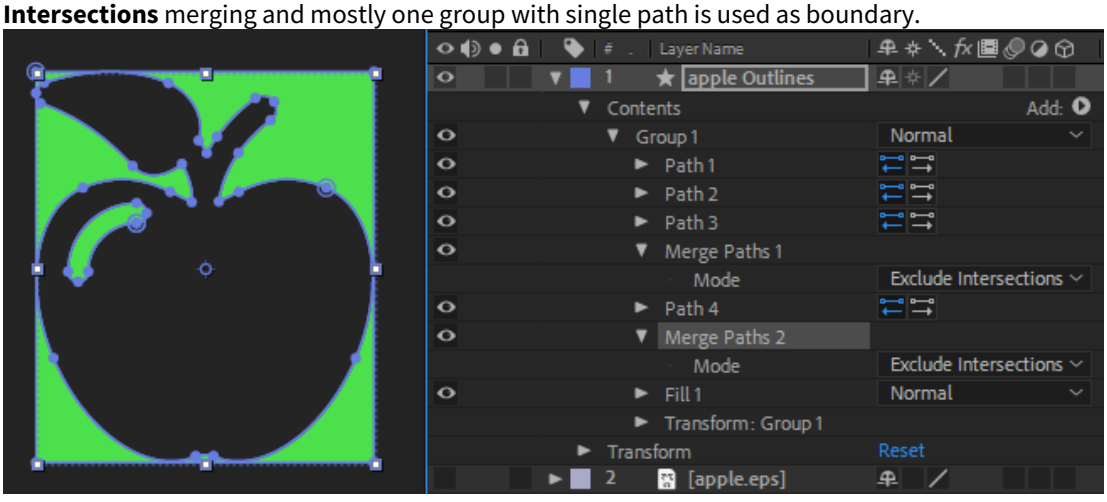
ii. Right click on the vector layer of automatically created composition and select **Create Shapes from Vector Layer**.

iii. Expand Contents and every groups inside, each group is the same as character of converted text.

iv. Expand **Merge Path** and change Mode to **Exclude Intersections**.

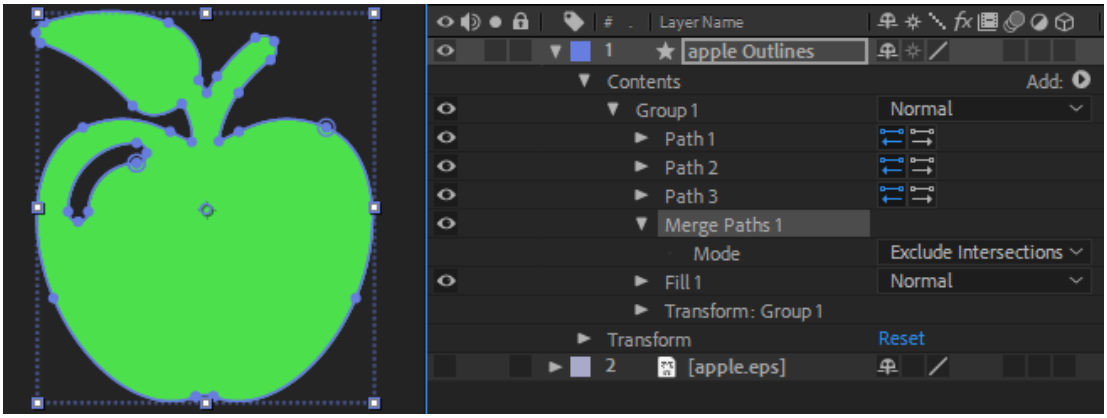


Some converted vector shapes have multiple group of merged paths and not work with **Exclude Intersections** merging and mostly one group with single path is used as boundary.



In this case, that group (Merge Paths and its above Path) can be deleted and other groups can use **Exclude Intersections** creates correct result.

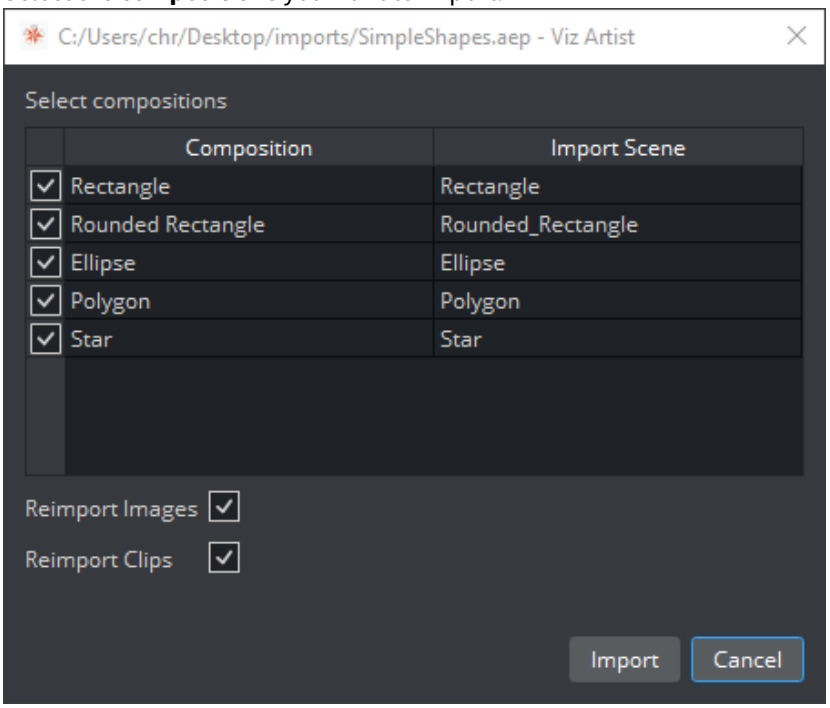
Copyright © 2026 Vizrt



Note: Layer animation is supported, but not object animation inside. The layer is imported as video if any properties of them have a keyframe.

24.4.3 To Import After Effect Projects

- Make sure your setup has a Matrox board or DSX Core enabled.
- Start Viz Artist.
- Choose **Import** and switch the type to *Scene*.
- Locate your *.aep* file and drag it to your location in Asset View.
- Select the **compositions** you want to import.



- Select if **images** or **clips** need to be re-imported.
- If a warning shows up about last supported Adobe After Effects version, click **OK.**

Copyright © 2026 Vizrt

24.4.4 Frame Accurate Output

The imported clip is the Master Clip for the animation's director. To secure frame accurate output of the animation and clip playback, set the following settings in the **Communication > Global** section of Viz Configuration:

- **Frame Accurate Viz Comm.:** `On`.
- **Command Field Dominance:** `Odd Retrace Counter`.
- In the **Video Input > Clip Input** section, set **Texture Delay** to `0`.
- Finally, you need to edit the configuration file manually to set the following flags:
  - **Matrox0.VideoIn1.VideoDelayDVE =** `0`.
  - **ClipIn1.VideoDelayDVE =** `0`.

Copyright © 2026 Vizrt

24.5 NDI and TriCaster

Vizrt's TriCaster is a popular vision mixer for small to medium budget operations. With the Viz Engine integration for TriCaster, Viz Engine provides a fill and key IP stream which is automatically detected by TriCaster as a source for graphics overlay. Two different graphic streams can be assigned in the TriCaster (net1 and net2). Each stream is a fill key combination.

As part of the Viz Artist and TriCaster integration package, Viz Trio is included. If the integration package runs on a Viz Trio One Box, animated preview is available within Viz Trio. Viz Trio can also be controlled directly from the TriCaster through macros that can be mapped to keyboard and control panel shortcuts, and are routed to Viz Trio by Viz Engine. See To Set TriCaster Macros.

A second integration package includes Viz Trio on a HP ZBook, with still preview to create an ultra-portable graphics and video production solution for mobile production trucks (see Viz Trio and Viz Engine for a TriCaster Production on the Same Laptop).

This section contains the following topics and procedures:

- System Requirements
- NDI Protocol
- To Connect a Viz Engine to TriCaster
- To Enable Audio Passthrough from Viz Engine to TriCaster
- To Set TriCaster Macros
- Viz Trio and Viz Engine for a TriCaster Production on the Same Laptop

24.5.1 System Requirements

- Viz Artist/Engine: 3.6.3 and higher
- TriCaster: All versions of TriCaster since:
  - **Build Number:** 2-2-150507B
  - **Driver** = 1.99 1-Apr-2014
  - **Core** = 24e8 29-Jul-2014

24.5.2 NDI Protocol

Support for the NDI protocol from Vizrt group was introduced in Viz Artist/Viz Engine version 3.8.3, with the latest version of the TriCaster software. This enables dual channel output to the TriCaster on one system via named video pipes. Other Viz systems may also be sending on the same network. Systems still running the TriCaster Standard Edition software need to upgrade to the Advanced Edition 2.0 version. Contact Vizrt to obtain the required installer.

Within the TriCaster, all Viz systems on the network can be seen and freely selected to any inputs. With NDI it is possible to check the number of clients actually connected to the Engine, and only render when they are connected. Thus it is possible to expose more channels than you want to actually render and only dedicate rendering time to those that are being used.

Similar to how Viz Artist configuration files are named, a particular Viz Engine appears to the TriCaster with the following naming convention: *computerName-VizEngine-instance*

The StudioMonitor tool, available from Vizrt, allows any video source or output on the TriCaster and/or NDI network to be viewed.

Copyright © 2026 Vizrt

To run Viz Engine in Tricaster mode with a video board installed on the system, set **Check Video Card** to `None` in the **Video Board** section of Viz Configuration.

24.5.3 To Connect a Viz Engine to TriCaster

1. Open **Viz Configuration**.
2. Select the **Video Board** section.
3. Set **Check Video Card** to `None`.
4. Set **Software I/O Mode** to `NDI`.
5. Make sure the Viz Artist dongle has the TriCaster feature enabled.
6. In the TriCaster user interface, click on the drop-down menu (next to **net1** or **net2**).
7. Using the drop-down menu from step 5, connect either net1 or net2.

24.5.4 To Enable Audio Passthrough from Viz Engine to TriCaster

To enable audio pass through from the Viz Engine to TriCaster, set `AudioOn=1` in the Viz configuration file. This is most easily done via **Artist > Config > Audio Settings > Audio Active** button = `On`. With this option set, the network stream to the TriCaster has audio embedded in the signal.

24.5.5 To Set TriCaster Macros

1. Open the macro configuration panel from the TriCaster Macros menu.
2. Create a new macro called **Trio Take**.
3. Edit the macro:
    - **Shortcut:** net1 (tells TriCaster that the network address is the one where Viz Trio is installed)
    - **Value:** vizrt
    - **Key0:** trio
    - **Value0:** take
4. Assign a keyboard shortcut to the macro.
5. Start streaming from the Viz Engine, which is on the same host as Viz Trio.
6. Connect TriCaster **net1** to it.

24.5.6 Viz Trio and Viz Engine for a TriCaster Production on the Same Laptop

On a laptop, there is only one GPU and it must be shared between Viz Trio preview and Viz Engine program out. Since this uses one GPU to create the preview and the program, it is advisable to use a ringbuffer on the Viz Engine output. Depending on how heavy the Scenes are, this can go up to 15 frames. To run Viz Trio and Viz Engine, Viz Engine should be installed for a Viz Trio One Box setup by selecting the **Trio Box CG** platform type during installation.

1. Start Viz Trio client with the `-vizpreview localhost:50010` command line option.

> ⚠ **Note:** The `50010` port is a dedicated preview port (check in the Viz configuration file (<*viz data folder*>).

2. Start Viz Engine.

Copyright © 2026 Vizrt

24.6 Photoshop Import

The revised Photoshop import of Viz Engine now allows a designer to import an Adobe Photoshop project as a scene instead of a single image. This improves the workflow as the Photoshop structure can be directly used in Viz Engine. There is no need for a designer to first export work as intermediate graphics and import them again. The import keeps all your structure and layers, masks and groups.

It turns this:

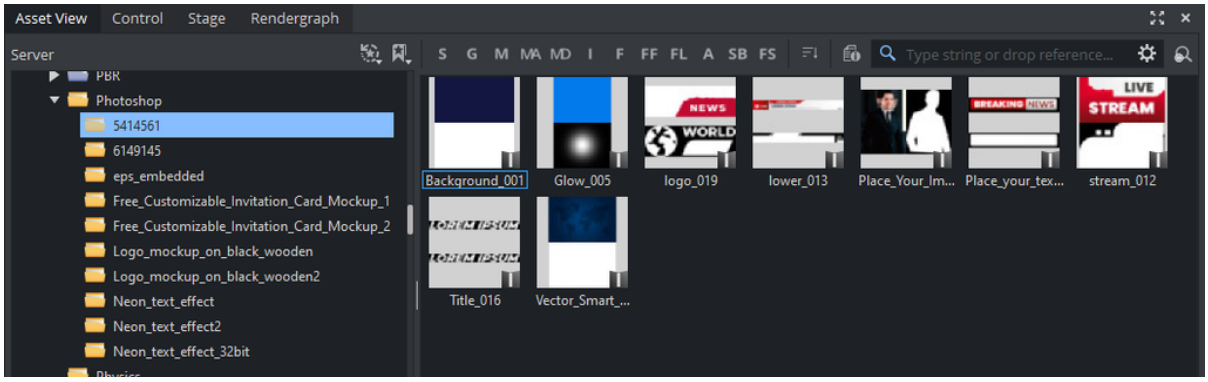


into this:



Copyright © 2026 Vizrt

To import a Photoshop document as a Viz Scene, drag the *.psd* file into the Asset View or choose **Import**, select **Scenes** as type and import the *.psd* file. The Photoshop File is then available as a Viz Scene, all required textures are stored in a subfolder of your Graphic Hub.



The structure is kept (in reverse order). Do not remove the Expert and Z-Sort Off plug-ins on top of your scene tree. These are required to replace all text elements with Viz Engine texts and add control plug-ins if required.

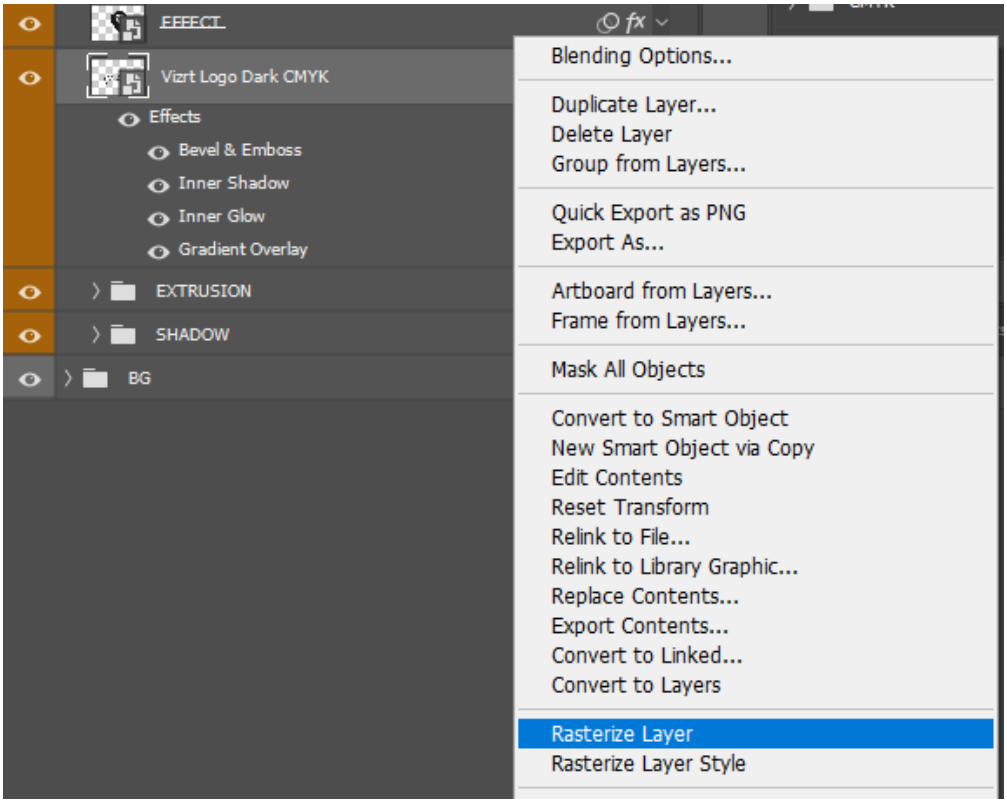
24.6.1 What is Supported

- **Layer hierarchy:** All layers, groups, visibility, etc. are kept and represented in the Viz Scene Tree.
- **Transparency:** Opacity settings in Photoshop are replaced by an Alpha plug-in without pre-multiplying the content.
- **Images:** All images will be imported and automatically stored in a subfolder of your target directory to keep the Graphic Hub clean. Images are imported as quad geometries and a Phong Viz Engine Material holding the texture..
- Masks are fully supported and will be part of the Phong Viz Engine Material texture.
- Vector/Smart Objects are rasterized and imported.
- 8-bit, 16-bit and 32-bit images are supported.

24.6.2 Known Limitations

- Import works for Viz Engine Scenes only.
- Text layers are rasterized as Textures. As the import of Fonts used in the Photoshop document can currently not be handled, it is required to replace the font elements afterwards with Text.
- Photoshop's Blend modes like Color Burn, Overlay, Soft-light, etc. are not supported. Only Normal is used.
- Adjustment layers are not supported. It is required to rasterize all of these effects before importing them into Viz Engine.

Copyright © 2026 Vizrt



Acknowledgements

Photoshop template by freepik.com.

Copyright © 2026 Vizrt

24.7 Alembic Import

- Introduction
- Supported Features
- How Alembic Operates in Viz Engine
    - Sample-to-Time Conversion

24.7.1 Introduction

The Alembic (*.abc*) file format is a widely adopted standard in 3D production pipelines for storing and exchanging complex animated scenes. It supports efficient handling of large datasets, including geometry, hierarchy, and animation data.

Starting with Viz Engine 5.4, **sampled meshes** (meshes with time-varying vertex data) are **natively supported** in the Viz Engine Renderer. This functionality is provided through the `DynamicGeometry` system and its associated `DynamicGeometryOmoNode`, enabling real-time playback of baked simulations such as cloth, fluids or character deformations directly from Alembic files.

In addition, **animated transformations** (position, rotation scale over time) are supported, allowing for dynamic and accurate scene playback.

24.7.2 Supported Features

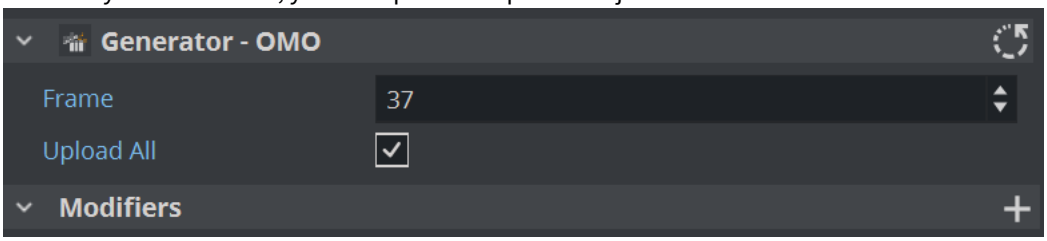
- **Sampled Meshes (Time-Varying Geometry)**
    - Supported via `DynamicGeometry` and `DynamicGeometryOmoNode`.
    - Meshes with vertex positions that change over time (for example, baked simulations or deformations) are imported and rendered correctly.
- **Scene Tree Hierarchy**
    - Preserved from the Alembic file.
    - Parent-child relationships and groupings are maintained.
- **Static and Animated Transformations**
    - Both static and time-varying transformations (position, rotation, scale) are supported.
    - Transform animations are correctly imported and evaluated during playback.
    - Hierarchical transformations are respected.

24.7.3 How Alembic Operates in Viz Engine

- **Importing Alembic Files**
    - Alembic files are imported via the **Import section of Viz Artist**.
    - During import, the Alembic content is converted into a **merged GEOMETRY object** and stored in Graphic Hub.
- **Resulting Object**
    - The imported Alembic file appears as a **GEOMETRY** asset in Graphic Hub.
    - All meshes and hierarchy are merged into a single geometry object for simplified handling.
- **Playback and Rendering**

Copyright © 2026 Vizrt

- If the Alembic file contains sampled mesh data, it is handled by the `DynamicGeometry` system. A "OMO" Generator is added automatically.
- To modify the animation, you can split the imported object.



Sample-to-Time Conversion

When importing animated data from Alembic scene files, each animation sample must be accurately mapped to a time value in seconds. This conversion is essential for ensuring temporal consistency across different digital content creation (DCC) tools and playback environments.

To convert sample indices to time values, the importer follows a prioritized strategy determining the frame rate (FPS):

1. **DCC FPS Metadata**
   If the Alembic file contains embedded metadata specifying the frame rate used in the original DCC application, this value is honored as the authoritative source. This ensures fidelity to the original animation timing.
2. **Engine Output Format Frame Rate**
   If no metadata is available, the importer falls back to the frame rate defined by the engine's output format settings. This allows for consistent integration with the target rendering or playback environment.
3. **Default Frame Rate (24 FPS)**
   In cases where neither metadata nor Engine settings provide a usable frame rate, a default of **24 frames per second** is applied.

Copyright © 2026 Vizrt