Plug-ins User Guide

Version 5.6



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 Plug-ins Introduction 12
1.1 Plug-in Color Coding 12
1.2 Supported Pipelines 13
1.3 Related Documents 13
1.4 Feedback and Suggestions 14
2 Basic Plug-ins 15
2.1 Related Documents 15
2.2 Feedback and Suggestions 15
2.3 Geometry Plug-ins 16
2.3.1 Default Geometry Plug-ins 17
2.3.2 Dynamics Geometry Plug-ins 106
2.3.3 Visual Data Tools Geometry Plug-ins 112
2.3.4 Primitives Plug-ins 159
2.3.5 RealFX Geometry Plug-ins 163
2.3.6 Ticker Geometry Plug-ins 169
2.3.7 Topology Geometry Plug-ins 186
2.4 Container Plug-ins 191
2.4.1 Basic Container Plug-ins 192
2.4.2 Arrangement Container Plug-ins 214
2.4.3 Control Container Plug-ins 222
2.4.4 Default Container Plug-ins 311
2.4.5 Feed Container Plug-ins 315
2.4.6 Container FX Plug-ins 319
2.4.7 Global Container Plug-ins 334
2.4.8 MultiTouch Container Plug-ins 402
2.4.9 Presenter Container Plug-ins 420
2.4.10 RealFX Container Plug-ins 429
2.4.11 Script Container Plug-ins 437
2.4.12 Sound Container Plug-ins 438
2.4.13 SplineFX Container Plug-ins 441
2.4.14 TextFX Container Plug-ins 444
2.4.15 Texture Container Plug-ins 467
2.4.16 Ticker Container Plug-ins 499
2.4.17 Time Container Plug-ins 503

2.4.18 Tool Container Plug-ins 508
2.4.19 Topology Container Plug-ins 562
2.4.20 Transformation Container Plug-ins 564
2.4.21 Visual Data Tools Container Plug-ins 568
2.4.22 Text Container Plugins 586
2.4.23 VizMaps Plug-ins 591
2.5 Shaders (Classic) 602
2.5.1 Default Shaders 603
2.5.2 Effect Shaders 604
2.5.3 Filter Shaders 618
2.5.4 Material Shaders 627
2.5.5 RealFX Shaders 637
2.5.6 Texture Shaders 642
2.5.7 RTT Advanced Materials Shaders 657
2.6 Scene Plug-ins 684
2.6.1 Default Scene Plug-ins 685
2.6.2 Scene Image Plug-ins 687
2.6.3 Lineup Scene Plug-ins 690
2.6.4 Scene Control Plug-ins 692
2.6.5 MultiTouch Scene Plug-ins 698
2.6.6 Scene Scripts 700
2.6.7 Scene Texture Plug-ins 701
2.6.8 Scene Tools Plug-ins 705
2.6.9 Scene RealFX Plug-ins 709
2.7 Shaders (Viz Engine Renderer) 711
2.7.1 Particle System 712
2.7.2 XR Draw 715
3 Viz Engine Extension Plug-ins 716
3.1 Lifecycle 716
3.2 Communication 717
3.3 Message API 718
3.4 Sample Extension 718
3.5 Sample Plug-in 719
3.6 Sample Script 720
3.7 Related Documents 720
3.8 Feedback and Suggestions 720
3.9 Plug-in API 721

3.9.1 Callbacks 721
3.9.2 Functions 722
3.10 Script API 724
3.10.1 Functions 724
3.10.2 Callbacks 724
3.11 Extension Plug-ins 725
3.11.1 viz_command_over_websocket 726
3.11.2 viz_probel 727
3.11.3 viz_sealevel 734
3.11.4 viz_tally_tsl 741
3.11.5 viz_webrtc 746
4 DataPool Plug-ins 754
4.1 Related Documents 754
4.2 Feedback and Suggestions 754
4.3 Introduction to DataPool Plug-ins 755
4.3.1 Example 755
4.4 DataPool Plug-ins Overview 757
4.4.1 Overview 757
4.4.2 Example 757
4.5 Installation 759
4.5.1 DataPool Plug-in Groups 760
4.5.2 Installing DataPool 763
4.5.3 Advanced Silent Installation 764
4.6 Architecture 765
4.6.1 DataFields 765
4.6.2 DataPool Hierarchy 767
4.6.3 Data Objects 770
4.6.4 Arrays of Data Objects 772
4.6.5 Tables of Data Objects 773
4.6.6 DataPool Variables Scope 775
4.6.7 DataPool Configuration Files 776
4.6.8 External Interface 778
4.7 Expressions 779
4.8 Common DataPool Parameters 780
4.9 Special DataPool Variables 782
4.10 DataPool Container Plug-ins 784
4.10.1 Data3DObject 786

4.10.2 DataAction 787
4.10.3 DataActionTable 788
4.10.4 DataAlpha 789
4.10.5 DataAnim 790
4.10.6 DataArray 791
4.10.7 DataArrow 792
4.10.8 DataCamera 793
4.10.9 DataCenter 794
4.10.10 DataClick 795
4.10.11 DataCondition 796
4.10.12 DataCountdown 798
4.10.13 DataCounter 799
4.10.14 DataDirector 800
4.10.15 DataDispatcher 801
4.10.16 DataDrawMask 802
4.10.17 DataFeedback 803
4.10.18 DataGeom 804
4.10.19 DataGPI 805
4.10.20 DataGraph 806
4.10.21 DataGraphPoint 807
4.10.22 DataHyperlink 808
4.10.23 DataImage 811
4.10.24 DataInRange 812
4.10.25 DataInterpolate 813
4.10.26 DataKey 814
4.10.27 DataKeyFrame 815
4.10.28 DataKeyFrame2 816
4.10.29 DataKeyText 817
4.10.30 DataKeyTime 818
4.10.31 DataLookup 819
4.10.32 DataLUImage 820
4.10.33 DataManipulate 821
4.10.34 DataMaterial 823
4.10.35 DataMaterialGradient 826
4.10.36 DataMaterialIndex 827
4.10.37 DataMath 828
4.10.38 DataMathObject 829
4.10.39 DataMinMax 830

4.10.40 DataMouseAction 831
4.10.41 DataMousePosition 832
4.10.42 DataMultiParam 833
4.10.43 DataNumber 834
4.10.44 DataObject 835
4.10.45 DataObjectTracker 836
4.10.46 DataParameter 837
4.10.47 DataParamTracker 838
4.10.48 DataPosition 839
4.10.49 DataReader 840
4.10.50 DataRotation 851
4.10.51 DataScale 852
4.10.52 DataScreen 853
4.10.53 DataSelector 854
4.10.54 DataSerialClock 855
4.10.55 DataSerialGPS 856
4.10.56 DataSHM 857
4.10.57 DataSHMTracker 858
4.10.58 DataStructure 859
4.10.59 DataSwitch 860
4.10.60 DataSystem 861
4.10.61 DataTable 862
4.10.62 DataTemo 863
4.10.63 DataText 864
4.10.64 DataTextKerning 866
4.10.65 DataTexture 867
4.10.66 DataTime 868
4.10.67 DataTimer 870
4.10.68 DataViz3Script 871
4.10.69 DataWPosition 872
4.11 DataPool Scene Plug-ins 873
4.11.1 DataClock 874
4.11.2 DataInteractive 875
4.11.3 DataMaterialTable 877
4.11.4 DataMouseSensor 878
4.11.5 DataPool Plug-in 879
5 Viz World Plug-ins 882

5.1 Related Documents....882
5.2 Feedback and Suggestions....882
5.3 Maps Geometry Plug-ins....883
5.3.1 2D Label....884
5.3.2 3D Border....889
5.3.3 3D Line....893
5.3.4 3D Line Control....898
5.3.5 3D Models....899
5.3.6 3D Region....903
5.3.7 3D Region Control....908
5.3.8 3D Roads....909
5.3.9 Atlas....912
5.3.10 C3D Terrain....919
5.3.11 GeoChart....921
5.3.12 GeoImage....926
5.3.13 Globe....928
5.3.14 Label and Go....931
5.3.15 MapScale....936
5.3.16 Pyramid Control....937
5.3.17 ShadowAgent....938
5.3.18 WindFlows....939
5.4 Maps Container Plug-ins....945
5.4.1 3D Border Manager....947
5.4.2 3D Line Manager....948
5.4.3 3D Line Tracer....951
5.4.4 3D Map Telestrator....961
5.4.5 3D Map Telestrator Design....969
5.4.6 3D Region Manager....970
5.4.7 3D Roads Manager....973
5.4.8 Center Map....976
5.4.9 CWM Client....977
5.4.10 Focus On Map....988
5.4.11 Geo Text....991
5.4.12 GeoData Reader....993
5.4.13 Hop It....995
5.4.14 Hop Sync....996
5.4.15 Hops Manager....997

5.4.16 KML Reader 998
5.4.17 Label AddOn 1000
5.4.18 Label and Go Assistant 1001
5.4.19 Label It 1008
5.4.20 LatLongGrid 1012
5.4.21 Locator Control 1013
5.4.22 Map Layers 1015
5.4.23 Map Layers Control 1016
5.4.24 Map Tiler 1017
5.4.25 Map Zoom 1019
5.4.26 Mouse2Memory 1020
5.4.27 Mute 1021
5.4.28 NavCom 1022
5.4.29 NavFade 1029
5.4.30 NavFinder 1032
5.4.31 Navigator 1035
5.4.32 NavScale 1040
5.4.33 NavSlave 1041
5.4.34 Place Finder 1042
5.4.35 Publish To Design 1044
5.4.36 Region2Tex 1045
5.4.37 Trace It 1048
5.4.38 World Position 1050
5.5 Maps Scene Plug-ins 1051
5.5.1 3D Map Setting 1052
5.5.2 Label Manager 1056
5.5.3 Light On Globe 1060
5.5.4 Map Builder 1062
5.5.5 Traffic Manager 1063
5.6 Maps Shader Plug-ins 1064
5.6.1 3D Line Shader 1065
5.6.2 C3D Terrain Shader 1066
5.6.3 FadeTexture 1067
5.6.4 Geo Chart Shader 1068
5.6.5 Rebound 1069
6 PixelFX Plug-ins 1070
6.1 Related Documents 1070

6.2 Feedback and Suggestions....1070
6.3 Introduction to PixelFX Plug-ins....1071
6.4 PixelFX Geometry....1072
6.4.1 pxLensEnergyBolt....1074
6.4.2 pxLensGlowBall....1076
6.4.3 pxLensGlowSpikes....1077
6.4.4 pxLensPolyElement....1078
6.4.5 pxLensRandomPoly....1079
6.4.6 pxLensRays....1080
6.4.7 pxLensSpark....1081
6.4.8 pxLensStripes....1082
6.5 PixelFX Container....1083
6.5.1 PixelFX....1084
6.5.2 pxColorWorks....1087
6.6 PixelFX Shader....1100
6.6.1 Common Properties....1100
6.6.2 PixelFXLensFlare....1101
6.6.3 pxBCubic....1102
6.6.4 pxCCBase....1103
6.6.5 pxEqualize....1104
6.6.6 pxGaussianBlur....1105
6.6.7 pxGradient....1106
6.6.8 pxInvert....1108
6.6.9 pxLensDistort....1109
6.6.10 pxMotionBlur....1110
6.6.11 pxNoise....1111
6.6.12 pxPixelate....1112
6.6.13 pxPosterize....1113
6.6.14 pxRecolor....1114
6.6.15 pxRipple....1115
6.6.16 pxSparkle....1116
6.6.17 pxTurbDissolve and pxTurbWipe....1117
6.6.18 pxTurbulence....1118
6.6.19 pxTwirl....1119
6.6.20 pxWaves....1120
7 Socialize Plug-ins....1121
7.1 Related Documents....1121

7.2 Feedback and Suggestions....1121
7.3 Socialize Container Plug-ins....1122
7.3.1 HTTP Recognizer....1123
7.3.2 Playlist Reader....1125
7.3.3 Split Author....1129
7.3.4 Text Highlight....1131
7.3.5 WordCloud....1136

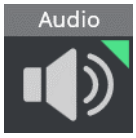
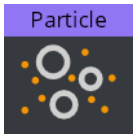
1 Plug-ins Introduction

The Plug-ins User Guide provides details about settings available through its configuration user interface within Viz Artist or Viz Engine.

Note: The Viz World plug-in package is released separately from the Viz World Client. Some of the functionality of these plug-ins are still intended to be used with the Viz World Client.

1.1 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

Copyright © 2026 Vizrt

Icon	Plug-in	Style
	Media Asset plug-ins	Black label

1.2 Supported Pipelines

If a Plug-in is not supported in both Viz Engine Renderer and Classic Renderer, the *Supported Pipelines* section shows which render pipeline is supported:

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

1.3 Related Documents

- *Viz Artist User Guide*: Contains information on how to install Viz Engine and create graphics scenes in Viz Artist.
- *Viz Engine Administrator Guide*: Contains information on how to install the Viz Engine software and supported hardware.
- *Viz Ticker User Guide*: How to install, configure and use the Viz Ticker client, and configure the output channels.
- *Viz Trio User Guide*: How to install, configure and use the Viz Trio client, and configure the output channels.
- *Viz Pilot User Guide*: How to install, configure and use Viz Pilot.
- *Viz Multichannel User Guide*: How to install, configure and use Viz Multichannel.
- *Viz World Classic User Guide*: Contains information on creating 2D maps and geographic animations.
- *Viz World User Guide*: Contains information on creating real-time 3D maps and using the client-server solution.

For more information about all of the Vizrt products, visit:

- www.vizrt.com
- Vizrt Documentation Center
- Vizrt Training Center
- Vizrt Forum

Copyright © 2026 Vizrt

1.4 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 Basic Plug-ins

The following describes the Basic Plugin Package bundled with the Viz Engine and Viz Artist installer.

2.1 Related Documents

- *Viz Artist User Guide*: Contains information on how to install Viz Engine and create graphics scenes in Viz Artist.

2.2 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

2.3 Geometry Plug-ins

The default path for the Geometry plug-ins is: *<viz install folder>\plug-in\<plug-in name.vip>*

- Default Geometry Plug-ins
- Dynamics Geometry Plug-ins
- Visual Data Tools Geometry Plug-ins
- Primitives Plug-ins
- RealFX Geometry Plug-ins
- Ticker Geometry Plug-ins
- Topology Geometry Plug-ins

Copyright © 2026 Vizrt

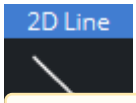
2.3.1 Default Geometry Plug-ins

The following Geometry plug-ins are located in the Default folder:

- 2D Line
- 2D Patch
- 2D Ribbon
- Alpha Map
- Arrow
- Circle
- Cog Wheel
- Cone
- Connector
- Cube
- Cycloid
- Cyclorama
- Cylinder
- Cylinder3
- Displacement Map
- Eclipse
- Fade Rectangle
- Filecard
- Graph
- Graph2D
- Icosahedron
- Image FX
- Multi Panel
- Noggi
- Pointer
- Polygon
- Quad
- Rect
- Rectangle
- Ring
- Roll
- Sphere
- Spline Path
- Spline Strip
- Spring
- Star
- Torus
- Triangle
- Wall
- Wave

Copyright © 2026 Vizrt

2D Line



The 2D Line plug-in draws a 2D line through given 2D coordinates, respective to point values.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

2D Line Properties

- **use LOD:** Enables level of detail. Range: `On - Off`. Default: `On`.
- **Enable Outline:** Enables drawing of an outline. Range: `On - Off`. Default: `On`.
- **Width:** Defines the width of the drawn line. Range: `0.0 - 500.0`. Default: `5.0`
- **Point values:** Lists x,y pairs separated by ':' character. x and y values are separated by blanks. Example: `0 0: 100 100: 200 70: 300 240: 400 280: 500 240: 600 400`.
- **New Line:** Press before values are entered (i.e. Point values), or else the line is rendered as invisible.
- **Clear All:** Deletes all line-segments.

To Create 2D Lines



1. Enter the **Point Values**. For example, `0 0 : 200 45` entered in the Point Values field creates a line that starts at the point X1(0), Y1(0) and ends at point X2(200), Y2(45).
2. Click the **New Line** button to create the line.

Copyright © 2026 Vizrt

2D Patch



The 2D Patch plug-in is a two dimensional planar curved grid of polygons. The grid is defined by control points that are located on the perimeter of the grid. The parameters of each one of the control points is the location X and Y, and the assigned texture coordinates U and V. The calculation of the internal grid points is done with a mix of a one-dimensional cubic spline and a two dimensional bi-linear interpolation.

It is possible to specify the interpolation direction to be just in one direction, or in both. Depending on the shape that should be created, it is important to choose the right direction (for example it is not too difficult to create an annulus with this plug-in, after choosing the right interpolation direction along the radius).

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

This page contains the following topics:

- 2D Patch Properties
- 2D Patch Workflow
- 2D Patch Tips
- 2D Patch Known Limitations

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

2D Patch Properties

- **use LOD:** Enables/disables dynamic level of detail.
- **Geo:**
- **X and Y Number of Intervals:** Sets the number of intervals between the control points in the X and Y direction respectively. Parameter name: *NX/NY*.
- **Interpolation Direction:** Defines the direction of interpolation from the control points on the perimeter to the internal grid points. Parameter name: *interDirect*.
- **X Size, Y Size, Reset XY (button) and Reset UV (button):** Gives initial values for the control points. The values of X and Y-Size takes effect just after hitting the **Reset XY** button. The Reset UV button sets default values to the texture coordinates. Parameter name: *SX/SY*, *resetXY/resetUV*.
- **X Resolution and Y Resolution:** Defines the number of subdivisions to polygons between the control points in the X- and Y-direction respectively.Parameter name: MX/MY.
- **Interactive mode:** A toggle to enable or disable the interactive mode of direction manipulation with the mouse. Parameter name: *interactive*.
    - **Editing Range %:** Sets the amount of effect the movement of one point (move with **SHIFT** pressed) should have on the neighboring points. Parameter name: *EditingRange.*
    - **Editing Mode:** When moving a point, it is possible, by pressing **SHIFT**, to move the other points simultaneously. How the points move depends on the editing mode that is chosen. Parameter name: *EditingMode*.

Copyright © 2026 Vizrt

    - **Constant (button):** The other points move their positions exactly equal to the one you move.
    - **Linear (button):** The other points move in a linear way, based on the distance from the point that is being moved.
    - **Exponential (button):** The other points move in an exponential way, based on the distance from the point it is being moved.
- **Draw Control Points:** A toggle to draw or not to draw the control points. Enables the Control Point Size parameter. Parameter name: *drawControlPoints*.
    - **Control Point Size:** Sets the control point size.
- **Show Control Points Values:** Enables the manipulation of the control points in the rendering window. Parameter name: *controlPointSize*.
- **New edit mode:** When enabled, and you press down **CTRL**, you can see the control points and also you have the opportunity to move the control points. The “Interactive mode” and the handle “script/plug-ins event mode” must be enabled.
- **X, Y, U and V:** Sets position and texture coordinates for each of the control points. These values are typically changed in the rendering window only with the mouse.

⚠ **Note:** There are 12 control points that each have their own X,Y,U and V parameters.

2D Patch Workflow

1. Set the right number of control points, according to your estimation of how complex the patch is you are going to create. This number can be changed at any time. The plug-in redistributes the new points along the already defined patch.
2. Estimate the final length and width of the patch.
3. Move the patch to the right place in your scene. If it is going to be on some horizontal plane, it might be easier first to edit it in the default orientation, and afterwards rotate it to the right place. There is no problem to continue working on it after the rotation has been made, but because of the perspective, it might be less straightforward.
4. Turn on the Draw Control Points toggle and the Interactive toggle. While working with the cursor, you do not have to be very close to the control point that you would like to manipulate. The plug-in finds by itself the nearest control point to where is your cursor is. The selected control point gets yellow color, as compared to all the rest that are white, to indicate that it has been selected for editing.

⚠ **Note:** You are able to do the direct manipulation just if the object is the selected object in the Viz Artist container tree. Remember also, that regardless of the position of your cursor, if the patch is the selected object, and the interactive toggle is on, always one control point becomes selected and is changed while working with the cursor. Hitting the space key on the keyboard cancels the current selection. The cursor then returns to normal functionality until the patch object is selected again.

5. With the left button, according to the selection roles above, you can change the X and Y location of each of the control points.
6. Once the shape of the patch is more or less defined, it is a advisable to set the resolution parameter. Performance wise you should set it as low as possible, but it should be high enough to guarantee a smooth enough look.
7. Choose an image and drag it on the patch in the usual way. After doing it open the image texture editor and set the Mapping property to Vertex, and the Wrap property to Clamp.
8. The next step is to map more accurately the image on the patch. Set first the Texture Length according to your estimation, then choose the Texture Direction, and then play with the Texture Head Location to see the texture flowing along the patch. This is a good point to review again all the parameters you set before.

Copyright © 2026 Vizrt

9. The last step is to define an animation if necessary. Typically with the patch, the animation is just on the Texture Head Location. To learn more about animation, see the **Create Animations** section of the Viz Artist User Guide.

Caution: While defining an animation on the parameters of a 2D Patch, you must make sure that the right window is open in Viz Artist. Without it, the changes that you are doing with the cursor does not take effect with regard to the animation.

2D Patch Tips

- It is sometimes useful to work with 2D Patch while in wireframe mode. If you have a texture with alpha, to see it correctly, you need to turn the image off with the small enable/disable button near the image icon.
- Remember that if the 2D Patch is not selected, the interactive mode is not active. As result of this, the first click with the mouse on unselected object is always with the normal Viz Artist functionality of the cursor, of moving, rotating and so on the object.
- If you are done with the interactive session of defining a 2D Patch, turn the **Interactive** toggle `Off`.
- While working on a 2D Patch in the interactive mode, it is not possible to select another object with the cursor in the Viz Artist Scene Editor. The way to select another object is first to hit *SPACE*, and thereafter select the other object.
- Doing animation on the 2D Patch is quite expensive performance wise. Be aware to it, and try to use as much as possible a small number of control points and polygons.
- For the texture used for the 2D Patch, make sure that along the flow direction, you have on both texture edges a clean line with alpha equal to `0`. Otherwise, you get a wake of the image edges in front or behind it.
- Do not change the number of control point along an animation. It is possible, if necessary, to change the resolution number.
- While defining an animation, make sure to remove from it, if necessary, the toggle values of Show Control Points, Interactive Mode and so on.

2D Patch Known Limitations

- When you use 2D Patch in a scene, by choosing an object in the Viz Artist rendering window, the Viz Artist user interface is not updated automatically.
- While copying a 2D Patch, to make it work in the interactive mode, it is necessary to save the scene first.
- If the texture coordinates on the control points are getting outside the [0,1] range artifacts may show. Be aware to keep it within the [0,1] range.

Copyright © 2026 Vizrt

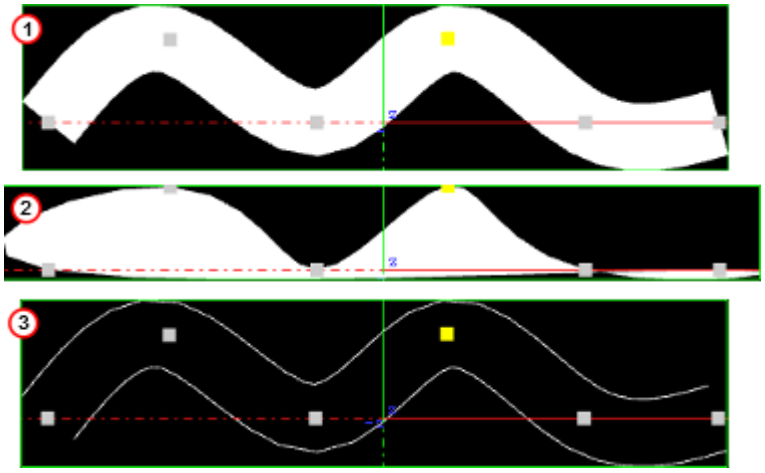
2D Ribbon



The 2D Ribbon plug-in is a curved strip. It is ideal to make customized curves to symbolize, for example, frontiers on a map or similar. The way the object is curved is easily edited in the 2D Ribbon property editor. It is possible to enable inter activity so that counterpoints of the ribbon can be dragged with the cursor.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

The 2D Ribbon can be used in Ribbon (1), Filled Area (2) or Double Outline (3) modes:



This page contains the following topics and procedures:

- 2D Ribbon Properties
  - Ribbon Mode Properties
  - Filled Area Mode Properties
  - Double Outline Mode Properties
- To Create a 2D Ribbon
- 2D Ribbon Tips
- 2D Ribbon Known Limitations

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

2D Ribbon Properties

The properties of the 2D Ribbon plug-in are different for each selected mode.

Copyright © 2026 Vizrt

Ribbon Mode Properties

- **Use LOD:** Enables or disables the dynamic Level of Detail (LOD)
- **Geo:** TBA
- **Working Mode:** Available modes are Ribbon, Filled Area and Double Outline. Default is the Ribbon mode.
- **Number of Control Points:** Sets the number of control points along the ribbon (when **Slave** mode is set to `Off`).
- **Resolution:** Defines the number of subdivisions to polygons between the control points.
- **Closed Curve:** Creates a closed a curve like a circle. The forward end touches the backward end.
- **Constant Width:**
  - **Set to on:** Any width modification is along all of the 2D Ribbon.
  - **Set to off:** Any width modification is around the nearest Control Point.

⚠ **Note: Preserve Constant Width While Possible** is not available when this is selected.

- **Add Caps:** Shows the Begin and End cap parameters when set to `On`. Removes the Begin and End caps when set to `Off`, even if they are set to `On` in the two parameters:
  - **Begin Caps:** Applies a cap at the start of the 2D Ribbon when set to `On`. Also sets the resolution of the start cap.
  - **End Caps:** Applies a cap at the end of the 2D Ribbon when set to `On`. Also sets the resolution of the end cap.
- **Preserve Constant Width While Possible:** Makes the 2D Ribbon try and maintain its set width when modified (available when **Constant Width** is set to `Off`) when set to `On`.
- **Interactive Mode:** Enables or disables the interactive mode of direction manipulation with the cursor:
  - **Editing Range %:** Sets the amount of effect the movement of one point (move with **SHIFT**) should have on the neighboring points.
  - **Editing Mode:** When moving a point, press **SHIFT** to move the other points simultaneously. How the points move depends on the editing mode that is chosen, **Constant**, **Linear** or **Exponential.**
  - **Constant:** The other points move their positions exactly equal to the one you move.
  - **Linear:** The other points move in a linear way, based on the distance from the point that is being moved.
  - **Exponential:** The other points move in an exponential way, based on the distance from the point it is being moved.
- **Reset Options:** Sets the 2D Ribbon to initial values for the Control Points and ribbon size. Changed values of Ribbon **Length** and Ribbon **Width** take effect after Reset has been clicked.
  - Reset **Reset:** Resets all created curves but does not delete the existing control points.
- **Texture Options:** Enables 2D Ribbon Texture options when set to `On`:
  - **Texture Direction:** Shows the texture in different directions. For example, if a pointer right-shows used as texture, the pointer shows to the right side if the **Horizontal** option is activated. Otherwise, to the left side if the
    -Horizontal option is set. Same for the **Vertical** and **-Vertical** option. The rendered pointer shows to the top or to the bottom side.
  - **Texture Head Location:** Moves the texture either to top or bottom direction.
  - **Texture Length:** Stretches the texture.

Copyright © 2026 Vizrt

    - **Use Texture Factor:** Enables the texture factor parameter when set to `On`.
- **Rotate Around X Axis:** Rotates the created 2D Ribbon around the X axis when set to `On`.
- **Show Spline:** Shows the spline of the 2D Ribbon when set to `On`.
- **Use Progressive Visualization:** When set to `On` the activated parameters affect the rendering:
    - **Progress Mode:** Slices the curve in the x-direction by length or by the control point index. Select from two options, **By Length** and **By Control Points**.
    - **Path Position:** Limits the workspace so there are less control points to prepare.
- **Render Bumps:** The curves are top-barbed which is useful for weather broadcasting.
    - **Warm:** Gives a rounded edge to the bumps.
    - **Cold:** Gives a sharper edge to the bumps.
    - **Occluded:** Gives alternating warm and cold bumps.
- **Slave Mode:** Enables one ribbon to act as the master and the other as the slave. The slave ribbon uses the **Number of Control Points** and **Resolution** set for the master.
- **Show Control Points Values:** Enables the manipulation of the Control Points in the render window. X, Y, and W values set the X/Y position and the width of the curve at the different Control Points. These values are typically be changed when edited directly in the Scene Editor.

Filled Area Mode Properties

For details see Ribbon Mode Properties.

Double Outline Mode Properties

For details see Ribbon Mode Properties.

To Create a 2D Ribbon

1. Add the 2D Ribbon plug-in to the Scene Tree.
2. Set **Interactive Mode** to `On`.
3. Set the **Number of Control Points**.

> **Note:** This number can be changed at any time. New points are distributed along the already defined ribbon (2).

4. Open **Reset Options:**
    a. Enter (estimate) the **Ribbon Length** and **Ribbon Width** (enter a new figure and press **ENTER**).
    b. Click Reset to activate the settings.
5. Toggle the **Constant Width**, as required.
6. Position the 2D Ribbon in the Scene.

> **Note:** If the 2D Ribbon is to be on some horizontal plane, it might be easier first to edit it in the default orientation, and afterwards rotate it to the right place. There is no problem to continue work on it after the rotation has been made, but because of the perspective, it might be less straightforward.

7. Click the (Events) button in the Scene Editor.
8. Modify the 2D Ribbon shape, either:

Copyright © 2026 Vizrt

- Use the mouse cursor to modify the Control Points. Click on or near the Control Point. The selected control point shows yellow (1).
- **Left button:** Click and drag the nearest Control Point.
- **Right button:** Modifies the 2D Ribbon width.
  or
- Set **Show Control Points Values** to **on**. Modify the Control Points as required.



⚠ **Note:** If the Container, which contains the 2D Ribbon plug-in, is selected, and **Interactive Mode** is set to `On` , one Control Point is always active.

9. Set the **Resolution**. For best performance set the resolution as low as possible, but high enough to guarantee a smooth enough look in the corners of the 2D Ribbon, and also a smooth enough mapping of the image/texture on the 2D Ribbon.
10. Choose an image and drag it on the 2D Ribbon. Open the Image texture editor and set:
    - **Mapping** to **Vertex**
    - **Wrap** to **Clamp**
11. Map the image more accurately on the 2D Ribbon. Set **Texture Options** to `On` :
    a. Set the **Texture Length** as required.
    b. Set the **Texture Direction** as required.
    c. Set the **Texture Head Location** to see the texture flowing along the 2D Ribbon.

Copyright © 2026 Vizrt



12. This is a good point to review again all set parameters.
13. Define an animation, if required. Typically with the 2D Ribbon the animation is just on the Texture Head Location.

IMPORTANT! While defining an animation on the parameters of a 2D Ribbon, make sure that the right window is open in Viz Artist. Without it the changes that are made with the cursor do not take effect, with regard to the animation.

2D Ribbon Tips

- It is sometimes useful to work with 2D Ribbon with use LOD set to on.
- If you have a texture with alpha, to see it correctly set the image to off with the small enable/disable button near the image icon.
- When finished with the creation of a 2D Ribbon, make sure that **Interactive Mode** is set to `Off`.
- When the 2D Ribbon **Interactive mode** is set to on, and the Scene Editor is active, it is not possible to select another object with the cursor in the Scene Editor. To select another object make sure that **Interactive mode** is set to off and the Scene Editor is inactive.

Copyright © 2026 Vizrt

- Animation on the 2D Ribbon is quite heavy on performance. Try to keep the number of Control Points and Polygons as low as possible.
- When a texture is used with a 2D Ribbon, make sure that, along the flow direction, on the texture edges there is a clean line with alpha equal to `0`. If not there is a wake of the image edges in front or behind it.
- Do not change the number of Control Point along an animation. It is possible, if necessary, to change the resolution number.
- While defining an animation, make sure to remove from it, if necessary, the toggle values of Show Control Points, Interactive Mode and so on.

2D Ribbon Known Limitations

- When you have in the scene a 2D Ribbon, by choosing an object in the Viz Artist Scene Editor, the Viz Artist user interface is not updated automatically.
- While copying a 2D Ribbon, to make it work in the interactive mode, it is necessary to save the scene first.
- If the texture coordinates on a Control Point are close to or outside the [0,1] range, artifacts may show. Keep it within the [0,1] range.

See Also

- 2D Follow
- **Create Animations** section of the Viz Artist User Guide

Copyright © 2026 Vizrt

Alpha Map



The Alpha Map plug-in enables the creation of an alpha map from a grayscale image by translating the intensity of the grayscale image to an opacity (alpha) value. This is useful for entities which do not directly support an image as their alpha channel.

Alpha Map does not accept an arbitrary image even if it only contains grayscale values. The image format must be luminance, for example, a single channel containing intensity values.

If a RGB image in dropped onto the plug-in, an error message is generated:

```
ERROR in plug-in AlphaMap: Image <8AFD1E04-B938-41D1-A743CF251028403B> not a valid luminance or alpha image
```

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Alpha Map Properties

- **Detail:** Sets the detail level of the map, the lower the value the higher the level of detail.
- **Image:** Drag the image to create the map to the drop zone. The image must be a valid alpha image.
- **X-size:** Sets the size of the map along the X-axis.
- **Y-size:** Sets the size of the map along the Y-axis.
- **Texture X Offset %:** Sets the offset of the texture in the X-axis.
- **Texture Y Offset %:** Sets the offset of the texture in the Y-axis.
- **Texture X Width %:** Stretches or compresses the texture in the X-axis.
- **Texture Y Width %:** Stretches or compresses the texture in the Y-axis.
- **Patch mode:**
  - **Triangle Strip:** Uses the same size of triangle polygon on the whole surface.
  - **Optimized Quads:** Uses bigger triangle polygons, where the surface of the displacement map is flat (reduce the total number of polygons and improve performance).
- **Inverse:** Mirror all coordinates.
- **Texture Coordinates:** Draws the texture over again when set to **Repeat** or stretches the texture to fit when the texture is to small to fit onto the Alpha Map rectangle when **Clamp** is selected.
- **Texel Precision:** Uses **Subtexel** for better texture image quality:
  - **Texel:** No sub-pixel or sub-texel corrections are made.
  - **Subtexel:** There is a limited number of pixels available on the screen, if a line does not run through a real pixel, it must be moved to the nearest one, this introduces a positional error. If **Subtexel** is selected Viz Artist breaks up pixels into smaller sub pixels in memory, so that the line can be drawn to the nearest sub pixel.
- **Smooth:** Creates a smooth look of the map, without the reduction of the polygon details of the map. In some cases the number of polygons are required to obtain the correct lightning.

Copyright © 2026 Vizrt

- **Texture Coordinates:** Decides if texture coordinates are to be created, and to which level. Select from **None**, **Full** or **Part (partial**). This is required if a texture is applied with a vertex mapping. Texture mapping consumes resources, if not needed select **None**.
- **Color:** Sets the color of the alpha map rectangle.

To Add an Alpha Map

1. Drag the Alpha Map plug-in to the Scene Tree.
2. Add an alpha (must be luminance only) image to the Alpha Map editor **Image** drop zone.
3. Drag a video source icon onto the alpha map's container.

Copyright © 2026 Vizrt

Arrow



The Arrow plug-in allows you to create and customize an arrow geometry. The arrow is a 2D object.

By clicking on the arrow icon in a container, the arrow editor opens. The default arrow has two heads. At the tip of each arrowhead the reference points are placed. The left point is 1 and the right is 2.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Arrow Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Point 1X/1Y/1Z/2X/2Y/2Z:** Sets the point offsets.
- **Width:** Sets the width of the arrow shaft.
- **Style 1:** Determines whether the arrow end at point 1 should have a head.
- **Style 2:** Determines whether the arrow end at point 2 should have a head.
- **Arrow Width:** Sets the width of the arrow heads.
- **Arrow Length:** Sets the length of the arrow head.
- **Percent:** Scales the arrow in percent of the size defined by X-, Y- and Z-values.
- **Mode:** Defines if the arrow lies in an XY, XZ or YZ plane.

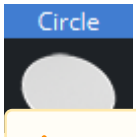
To Create an Arrow

Copyright © 2026 Vizrt

1. Create a group and add the Arrow plug-in to it.
2. Add a material and/or an image to the same container as the Arrow plug-in to add color and/or texture to it.

Copyright © 2026 Vizrt

Circle



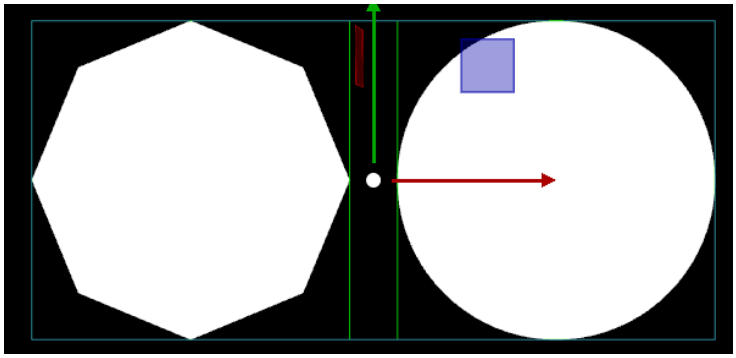
The Circle plug-in creates a circle with different corner levels.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Circle Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Radius:** Sets the radius of the circle.
- **Corners:** Sets the number of corners the circle is to be constructed of. The circle is set together by tiled triangles and the higher the number of corners is set, the more triangles are used to construct the circle. The higher the value set, the more rendering performance used.
- **Mode:** Changes the space perspective.
    - XY (front view)
    - XZ (bottom view)
    - YZ (left view)

To Create a Circle



1. Create a group and add the Circle plug-in to it.
2. Add a material and/or an image to the same container as the Circle plug-in to add color and/or texture to it.
3. Set the circle radius.
4. Set the number of corners.

Copyright © 2026 Vizrt

Cog Wheel



The Cog Wheel plug-in creates a cog wheel.

It has a range of properties to adjust the look of the cog wheel.

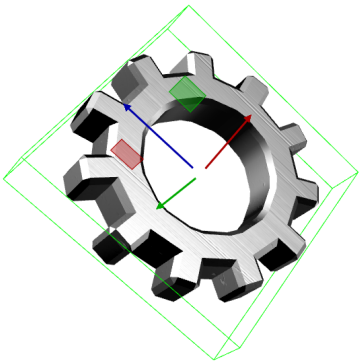
To avoid decreasing the system performance, set the tessellation to a lower value since it does not make much difference in quality.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Cog Wheel Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Tesselation:** Sets the level of detail.
- **Corners:** Sets the number of corners the cogwheel is to be constructed of. The cog wheel gets one tooth for each corner, so the number of teeth changes correspondingly.
- **Height:** Sets the height or width of the cog wheel.
- **Tooth height:** Sets the height of the cog wheel teeth.
- **Diameter:** Sets the diameter of the cog wheel.
- **Hole:** Creates and sets the size of a hole in the cog wheel.
- **Bevel:** Sets the degree of bevel at the cog wheel.
- **Inner Bevel:** Enables or disables bevel in the cog wheel hole.
- **Show Top:** Enables or disables visualization of the top.
- **Show Bottom:** Enables or disables visualization of bottom.
- **Center:** Allows you to select where the geometrical center should be placed on the cog wheel: Either *Center*, *Bottom* or *Top*.

To Create a Cogwheel

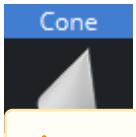


1. Add the Cog Wheel plug-in to a container.
2. Add a material to the same container.
3. Set **Hole** parameter to `50.0`.

Copyright © 2026 Vizrt

4. Set the **transformation** parameters for the container to: **Rotation** X: `90.0` , Y: `45.0` and Z: `-45.0` .

Cone



The Cone plug-in creates a cone geometry.

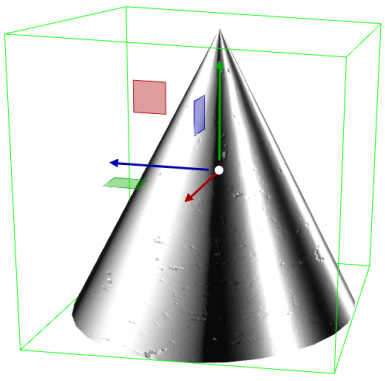
Set the tessellation to a reasonable value since it does not make a great difference in quality.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Cone Properties

- **Use LOD:** Enables or disables dynamic level of detail.
- **Tessellation:** Sets the degree of detail.
- **Height:** Sets the height of the cone.
- **Diameter:** Sets the diameter of the cone.
- **Corners:** Allows you to decide the number of corners the cone is to be constructed of.
- **Show Bottom:** Enables or disables visualization of the cone bottom.
- **Center:** Allows you to select where the geometrical center should be placed on the cone: Either *Center*, *Bottom* or *Top*.
- **Smooth:** Enables a smoothing of the cone edges.
- **Rounded Tip:** Enables the user to adjust the roundness of the cone's tip.
- **Rounded Tip Height:** Sets the parameter for the roundness of the tip. The angle of the cone's side does not change, the rounded tip is created by 'taking away' from the pointed tip. Therefore the cone does not have the full height.

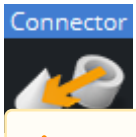
To Create a Cone



1. Add the Cone plug-in to a container.
2. Add a Material to the same container.

Copyright © 2026 Vizrt

Connector



The Connector plug-in connects two objects with a line of a required color and width.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Connector Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Container 1 and 2:** Placeholder for container 1 and 2. Drag the container onto the drop zone. Click **R** to remove the container.
- **Connection Point 1 and 2 By X, Y and Z Axis:** Sets the X, Y and Z-value of container 1 and 2 where the line should start. Can be set to *Min*, *Center*, *Max*, *CenterBB* or *Origin*.
- **Line Width:** Sets the width of the connecting line.
- **Drawing Mode:** Available options are *Line, Mesh and Viz Engine*.

Note: To use the plug-in in the Viz Engine Render Pipeline, the *Viz Engine* option is mandatory.

- **Line Color:** Defines the color of the line.
- **Border Thickness:** Applies a border with the thickness determined by the value.
- **Border Color:** Defines a border color.

To Connect Two Objects



1. Create a new group and add the Connector plug-in to it.

Copyright © 2026 Vizrt

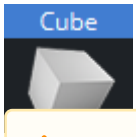
2. Create two new group containers and add a geometry object (for example, Cylinder and Cone) to each group.
3. Open each container's transformation editor and move the objects a part.
4. Add a material and/or an image to the geometry objects.
5. Open the connector plug-in editor and drag and drop the two containers to the Container 1 and Container 2 placeholders, respectively.
6. Adjust the color line.
7. Open the Connector plug-in editor and play with the settings.
8. Add more objects to Container 2 and animate it.

See Also

- Bounding Box

Copyright © 2026 Vizrt

Cube



The Cube plug-in creates cubes with particular widths, heights, depths and other attributes.

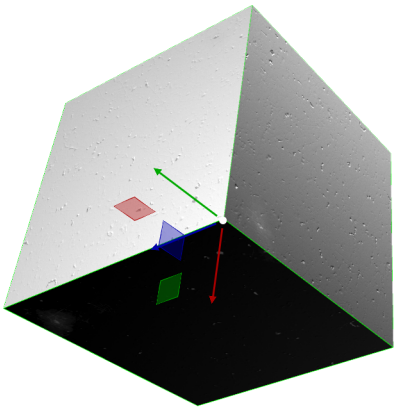
Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Cube Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Size X:** Sets the size of the cube in the X-axis. Parameter name: *size_X*.
- **Size Y:** Sets the size of the cube in the Y-axis. Parameter name: *size_Y*.
- **Size Z:** Sets the size of the cube in the Z-axis. Parameter name: *size_Z*.
- **Center Y:** Sets the position of the cube center along the Y-axis. You can choose between *Center*, *Bottom* or *Top*.
- **Tessellation:** Sets the degree of detail.
- **Adaptive Tessellation:** Adjusts tessellation on each axis separately, depending on the extension of the cube on that axis. The reference value is an extension of `100`. For a face of size `200`, the tessellation-parameter is doubled, for a face of size `50`, the tessellation is halved. The automatically adapted tessellation still never exceeds its maximum value of `100`.
- **Bevel:** Adds a bevel of the given size to corners and edges of the cube. The size of the bevel reduces the size of the cube's axis-aligned faces which: in the case 'Adaptive tessellation' is enabled causes them to become less tessellated accordingly. Tessellation of the bevel itself is affected by 'adaptive tessellation' too.
- **Rounded Bevel:** Rounds the beveled edges and corners when set to `On`. The roundness of the bevel is tessellated is affected by *adaptive tessellation*.
- **Show Top, Bottom, Front, Back, Left, Right:** Shows or hides the cube's face and adjacent corners and edges.
- **Show Size 0:** Enables/disables display of the cube at axis value zero (`0`). If one of the axis values are set to `0`, the cube remains visible if the setting is enabled (`On`). Disabling (`Off`) it makes the cube become invisible as long as one of the axis values are `0`. A situation where this parameter is useful is when creating bar chart animations. In such animations it is possible to hide the bar at value `0` by disabling this setting.

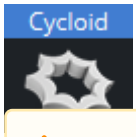
Copyright © 2026 Vizrt

To Create a Cube



1. Create a new group container and add the Cube plug-in to it.
2. Add a material and/or an image to the same container.
3. Open the Cube editor and set Rotation Y and Z values to `45.0`.

Cycloid



The Cycloid plug-in creates a Cycloid geometry.

The path traced by a point on a wheel as the wheel rolls, without slipping, along a flat surface.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Cycloid Properties

- **Shape**
  - **EpiCycloid:** A plane curve produced by tracing the path of a chosen point of a circle which rolls without slipping around a fixed circle.
  - **Hypo Cycloid:** A special plane curve generated by the trace of a fixed point on a small circle that rolls within a larger circle.
- **Corners:** Sets the number of corners the Cycloid is to be made up of.
- **Hole:** Creates a hole in the Cycloid.
- **Radius:** Sets the radius of the hole.
- **Factor Outer:** Sets the outer factor of the Cycloid.
- **Factor Inner:** Sets the inner factor of the Cycloid (when **Hole** is set to `On`).

To Create a Cycloid

Copyright © 2026 Vizrt

1. Create a new container.
2. Add the Cycloid plug-in to it (1: Default Hypo Cycloid).
3. Open the Cycloid editor.
4. Modify the Cycloid parameters, for example (2: Modified Cycloid).
    - **Corners:** `8`.
    - **Hole:** `On`.
    - **Radius:** `20`.
    - **Factor Outer:** `10`.
    - **Factor Inner:** `10`.
    - **Slice:** `5`.
5. If required, add a material and/or a texture.

Copyright © 2026 Vizrt

Cyclorama



The Cyclorama (Cyc) plug-in renders special geometry needed to represent a back wall, one corner and two corner Cyclorama.

Cyclorama is intended to be used in combination with Viz Studio Manager and Tracking Hub.

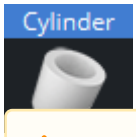
Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

There is no need for a designer to change parameters within this plug-in, as it is mainly used in combination with Viz Studio Manager and Tracking Hub.

For a detailed explanation on how to use this plug-in, please refer to **Working with Holdout and Trash Mattes** in the Viz Artist User Guide.

Copyright © 2026 Vizrt

Cylinder



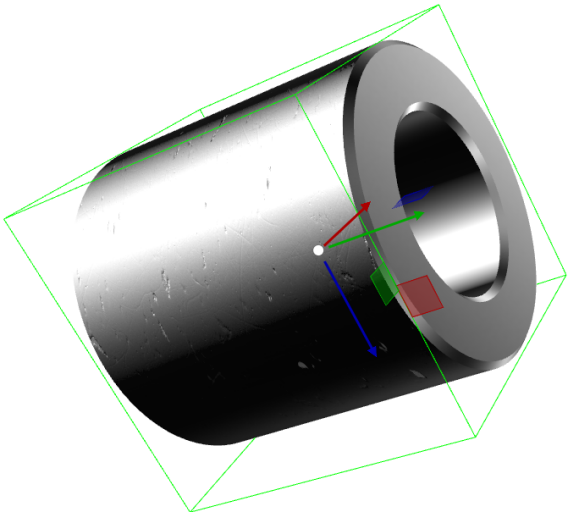
The Cylinder plug-in creates cylinders with different heights, widths and depths and other attributes.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Cylinder Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Height:** Sets the height of the cylinder.
- **Diameter:** Sets the diameter of the cylinder
- **Rotation:** Sets the rotation around X-axis.
- **Angle:** Sets the opening angle from 0-360°. As the value decreases below 360° an increasing angle opens in the cylinder.
- **Corners:** Sets the number of corners the cylinder is to be made up from.
- **Hole:** Creates a hole in the cylinder, making it into a tube.
- **Center:** Sets the position for the center. You can choose between *Center*, *Bottom* or *Top*.
- **Bevel:** Sets the size of bevel at the cylinder.
- **Inner Bevel:** Enables/disables bevel in the cylinder hole if bevel is set at the bevel parameter.
- **Show Top:** Turns off/on visualization of cylinder top.
- **Show Bottom:** Turns off/on visualization of cylinder bottom.
- **Show size 0:** Enables/disables display of the cylinder at height value `0`. If you set height to zero, the cylinder is visible by default. If you disable this option, the cylinder becomes invisible.

To Create a Cylinder



1. Create a new group container.

Copyright © 2026 Vizrt

2. Add the Cylinder or Cylinder3 plug-in.
3. Open the Cylinder editor.
4. Set Hole to `50.0`.
5. Add a material and/or an image to the same Container.
6. Open the cylinder editor.
7. Set Rotation Y to `75.0` and Z to `90.0`.

Copyright © 2026 Vizrt

Cylinder3



The Cylinder3 plug-in creates cylinders.

Compared to the Cylinder plug-in, Cylinder3 provides the designer with more advanced options and settings.

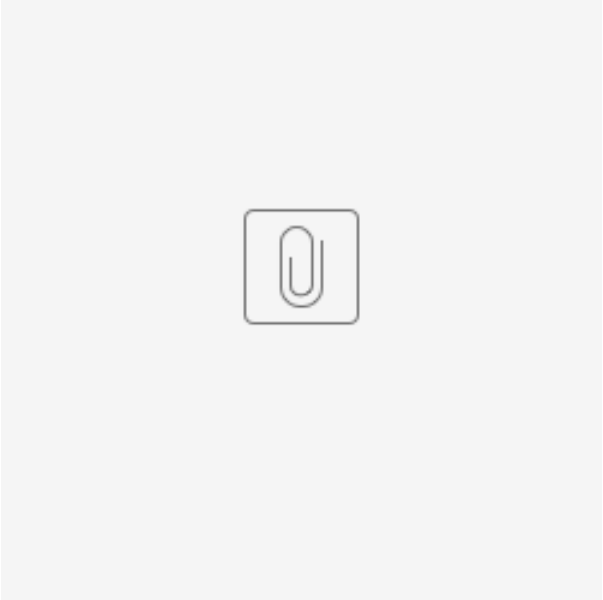
> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Cylinder3 Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Height:** Sets the height of the cylinder.
- **Diameter:** Sets the diameter of the cylinder
- **Rotation:** Sets the rotation around X-axis.
- **Angle:** Sets the opening angle from 0-360°. As the value decreases below 360° an increasing angle opens in the cylinder.
- **Opening Mode:** Describes in which direction the cylinder is opened when adjusting the Angle parameter:
  - **CW:** Opens the cylinder in a clockwise direction.
  - **CCW:** Opens the cylinder in a counter-clockwise direction.
  - **Center:** Opens the cylinder in clockwise and counter-clockwise directions simultaneously.
- **Corners:** Adjusts the resolution of the cylinder by setting the number of corners the cylinder is to be made up from.
- **Hole:** Creates a hole in the cylinder, creating a tube.
- **Center:** Sets the center position of the cylinder to *Center*, *Bottom* or *Top*.
- **Bevel:** Adjusts the bevel size of the cylinder.
- **Inner Bevel:** Enables or disables bevel in the cylinder hole if a value is given for the bevel parameter, even if **Hole** is set to `0.0`. When creating a cylinder with an outer bevel and flat top and bottom surfaces, this setting must be set to `Off`.
- **Show Top:** Turns visualization of the cylinder top surface on or off.
- **Show Bottom:** Turns visualization of cylinder bottom surface on or off.
- **Show size 0:** Enables or disables display of the cylinder when the value of **Height** is set to `0`. By default, the cylinder is visible when Height is set to `0`. By disabling this option, the cylinder becomes invisible. This is useful when designing pie chart visualizations.
- **Show inner bevel:** If this toggle is off, the inner bevel, respectively the inner cylinder is not rendered.
- **Show outer bevel:** If this toggle is off, the outer bevel, respectively the outer cylinder is not rendered.
- **Show caps:** Applicable only to cylinders with an opening angle. This setting determines whether a surface, or cap, should be added to the angle's sides. This setting only affects the geometry object if the angle is less than 360°. In combination with the **Back Face** setting of the Expert container plug-in, this setting allows for the design of cylinders with a cut out section exposing the cylinder's inside.
- **Static texture:** If the texture mapping of the texture is set to **Vertex** mode, this setting defines whether the texture bends around the object.
- **Texture anchor:** Changes the texture orientation in **X**-direction to left or right. Requires the angle to be less than 360°.

Copyright © 2026 Vizrt

To Create a Cylinder



1. Create a new group container.
2. Add the Cylinder3 plug-in and open the Cylinder3 plug-in editor.
3. Add a material or an image to the Container.
4. Open the Cylinder3 plug-in editor and set the desired parameters.
5. Use the transformation editor to set the container’s position and rotation.

Copyright © 2026 Vizrt

Displacement Map



The Displacement Map plug-in enables you to create topographical (height) maps by means of a grey scale image.

The intensity value of the gray scale image is translated to a height value.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Displacement Map Properties

- **Detail:** Sets the detail level of the map, the lower the value the higher the level of detail.
- **Image:** Drag the image you want to use creating the map, onto the placeholder. The image must be a valid luminance or alpha image.
- **X-size:** Sets the size of the map along the X-axis.
- **Y-size:** Sets the size of the map along the Y-axes.
- **Height:** Sets the height of the map.
- **Texture X Offset %:** Sets the offset of the texture in the X-axis.
- **Texture Y Offset %:** Sets the offset of the texture in the Y-axis.
- **Texture X Width %:** Stretches or compresses the texture in the X-axis (`0` is not supported, instead the default value is used).
- **Texture Y Width %:** Stretches or compresses the texture in the Y-axis (`0` is not supported, instead the default value is used).
- **Patch Mode**
  - **Triangle Strip:** Uses the same size of triangle polygon on the whole surface.
  - **Optimized Quads:** Uses bigger triangle polygons where the surface of the displacement map is flat, thereby reducing the total number of polygons and improving performance.
- **Inverse:** Mirrors all topographic coordinates through the zero level. This makes a mountain top to a crater.
- **Texture Coordinates:** Repeat/Clamp sets if the texture is the be repeated or clamped if it is too small to fit onto the displacement maps rectangle. Clamp stretches the texture to make it fit, Repeat starts drawing the texture over again when it reaches the end.
- **Texel Precision**
  - **Texel:** No subpixel/subtexel correction is made.
  - **Subtexel:** There is a limited number of pixels available on the screen, if a line does not run through a real pixel, it must be moved to the nearest one, this introduces a positional error. If subtexel is selected Viz Artist breaks up pixels into smaller sub pixels in memory so that the line can be drawn to the nearest sub pixel.
- **Smooth:** Use this parameter to smooth the look of the map, without reducing the polygon details of the map. In some cases you need the number of polygons you have to obtain the correct lightning.
- **Texture Coordinates:** Decides if texture coordinates are to be created, and to which level, either **None**, **Full** or **Partially**. You need this enabled to some level, if a texture is applied using a vertex mapping. Texture mapping is resource consuming, so if you do not need it, keep it off.

Copyright © 2026 Vizrt

Note: Please be aware not to increase to number of polygons in detail not too much, as this is a common mistake.

To Create a Displacement Map

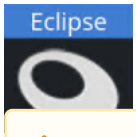


1. Create a new group and add the Displacement Map plug-in to it.
2. Add material to the group container.
3. Open the Displacement Map editor, and drag and drop a gray-scaled image onto the image placeholder.
4. Open the group container's transformation editor and adjust the Rotation values.

Note: An RGB or similar image does not work, and you get an information in the log field. The image must be a valid luminance or alpha image.

Copyright © 2026 Vizrt

Eclipse



The Eclipse plug-in enables you to create an eclipse shape.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Eclipse Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Corners:** Changes the number of triangles the object has. If you want good quality, increase the value. A value of `40` is a reasonable value.
- **Inner Radius 1 and 2:** Changes the inner radius in X-Position (1) and the inner radius in Y-Position (2).
- **Outer Radius 1 and 2:** Changes the outer radius in X-Position (1) and the outer radius in Y-Position (2).
- **Hole Offset X, Y and Z:** Moves the inner hole to X, Y, Z-position. With this parameter you can shape great objects like a volcano.
- **Rotation:** Rotates the eclipse. The result is visible if you change the angle less than 360°.
- **Angle:** Change this to create a view like a cake respectively a divided circle.
- **Create Uniform Normals:** Adapts the varying normals to the majority of normals in same direction.

To Create an Eclipse

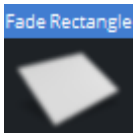


Copyright © 2026 Vizrt

1. Create a new group and add the Eclipse plug-in to it.
2. Add material to the group container.
3. Open the eclipse plug-in editor, and adjust the following values:
    - Set Inner Radius 1 and 2 to `40.0`.
    - Set Hole Offset X to `20.0`.
4. Open the group container's transformation editor and adjust the Rotation values.

Copyright © 2026 Vizrt

Fade Rectangle



The Fade Rectangle plug-in creates a rectangle with four sides that can be set to fade in a user-specified way.

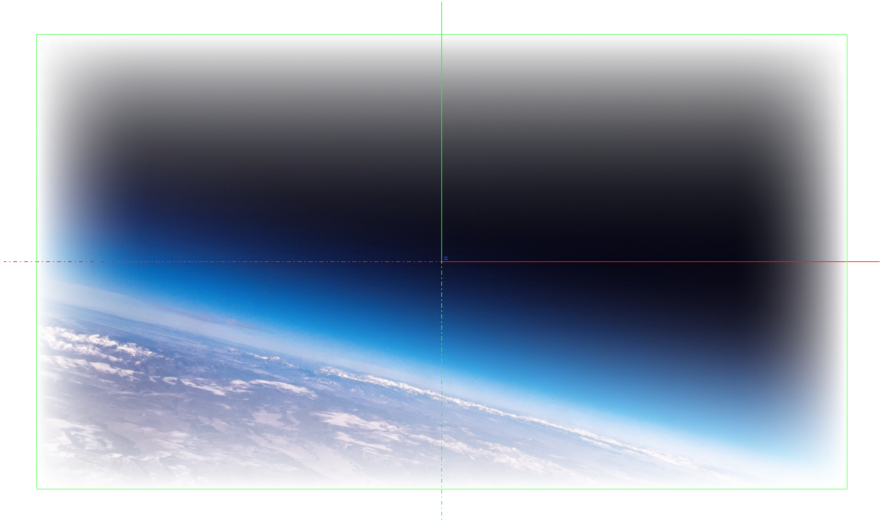
Both the degree of fading and the area influenced by the fading can be customized.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Fade Rectangle Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Tesselation:** Sets the degree of detail.
- **Width:** Sets the width of the fade rectangle.
- **Height:** Sets the height of the fade rectangle.
- **Lock Blend:** Allows you to decide whether to adjust the blending looked for both the X- and Y-axis or single (each axis separately).
- **Blend Range:** Allows you to set the blend range if looked mode is selected.
- **Blend Range X and Y:** Sets the blend range of the X and/or Y axis if single blend is selected.
- **Blend Factor:** Sets the alpha ramp of the fade rectangle. To see the effect set the alpha value to `0` and adjust the blend factor.
- **Color:** Shows the color of the rectangle. Set the color either in the editor below or drag a material from the Server Panel onto the color icon.
- **Mode:** Changes the space perspective.
    - XY (front view)
    - XZ (bottom view)
    - YZ (left view)

To Create a Faded Rectangle

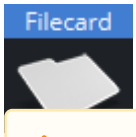


Copyright © 2026 Vizrt

1. Create a new group and add the Fade Rectangle plug-in onto it.
2. Add an image to this group.
3. Open the Fade Rectangle editor and change the required color through the color parameter and adjust the width and the height.

Copyright © 2026 Vizrt

Filecard



The Filecard plug-in creates file cards with different widths and heights and other attributes.

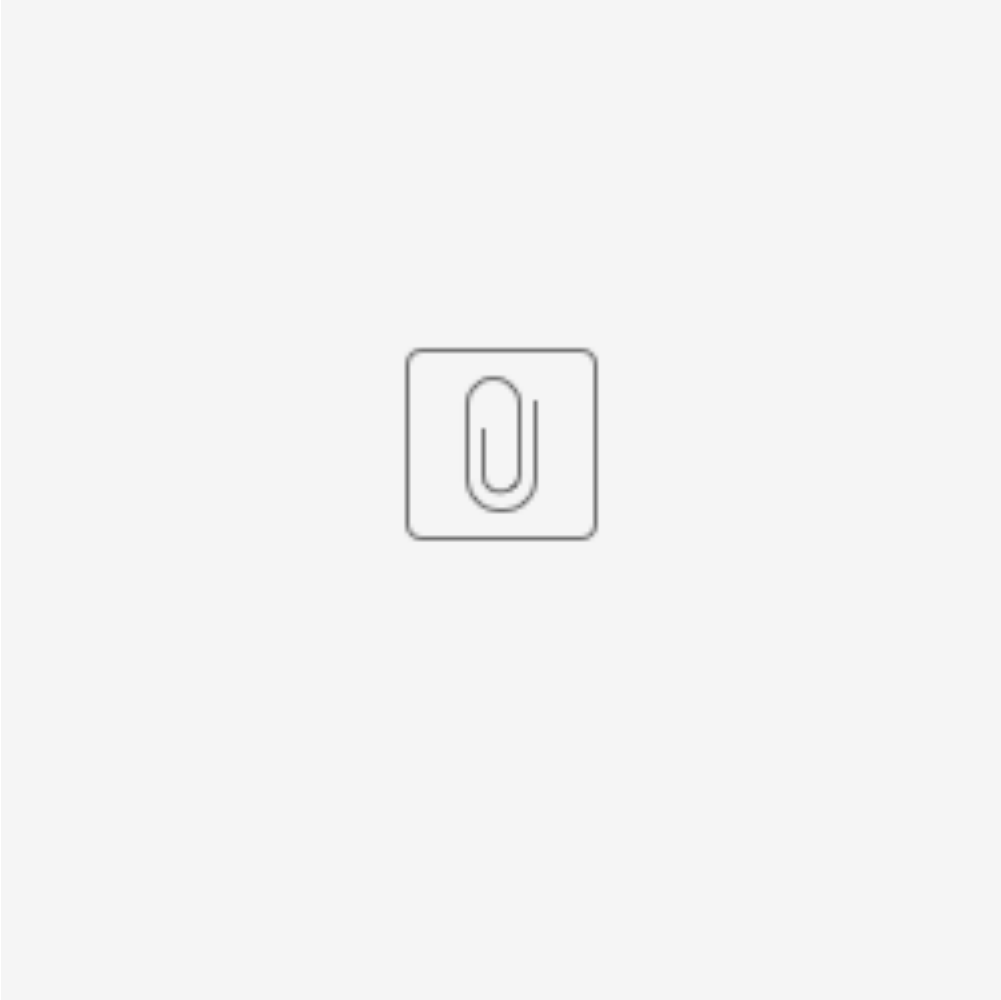
> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Filecard Properties

- **use LOD:** Enables/disables dynamic level of detail.
- **Width:** Sets the width of the file card.
- **Height:** Sets the height of the file card.
- **Tab Position X and Y:** Sets the tab position in x-direction and y-direction.
- **Tab Angle:** Reduces/increases the size of the curve.
- **Bevel:** Reduces/increases edge softness. Increase this value to get a softer edge.
- **Tesselation:** Changes the number of triangles of the rendered file card to increase the visual quality. Please consider that the number of triangles affects the performance of the system.

Copyright © 2026 Vizrt

To Create a Filecard



1. Create a new group and add the Filecard plug-in to it.
2. Add a material and/or a texture to the group.
3. Open the Filecard editor and set the following parameters:
    - Set Width to `700.0`.
    - Set Tab Position X to `70.0`, and Tab Position Y to `60.0`.
    - Set Tab Angle to `45.0`.
    - Set Bevel to `25.0`.

Copyright © 2026 Vizrt

Graph



The Graph plug-in allows you to create a 2D or 3D graph with up to 50 values.

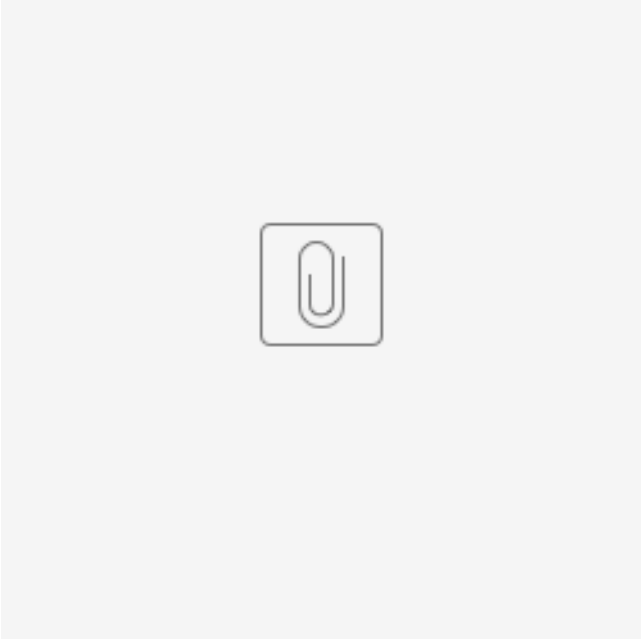
> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Graph Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Width:** Sets the width of the graph.
- **Height:** Sets the height of the graph.
- **Start:** Sets the starting point of the graph.
- **End:** Sets the ending point of the graph.
- **Line Width:** Defines the line width.
- **1st/last segment interpolation:** Switches the interpolation of the first and last segment on or off.
- **3D:** Enables/disables 3D visualization of the graph.
- **Extrusion depth:** Sets the graph's extrusion depth.
- **Add backside:** Enables/disables visualization of the graph backside.
- **Use vertex color:** Enables a base color for the graph. Set the values in the color editor or drag a material onto the small square below the parameter.
- **Line color:** Enables the use of a vertex color.
- **X0/Y0 to X49/Y49:** Enables up to 50 different value points of the graph by giving coordinates on the X- and Y-axis.

Copyright © 2026 Vizrt

To Create a Graph



1. Create a new group and add the graph plug-in to it.
2. Add a material and/or a texture to the group, or open the graph editor and enable **Use Vertexcolor** and set the Line Color.
3. Open the graph editor and set the Start and End points.
4. Enable 3D and set Extrusion Depth to `30.0`.
5. Add points for the X- and Y-directions (i.e. X0/Y0, X1/Y1 and so on).

Copyright © 2026 Vizrt

Graph2D



The Graph2D plug-in creates 2D and 3D graphs, lines and bars from imported files.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

This page contains the following topics and procedures:

- Graph2D Properties
- To Create a Channel File for a Graph2D
- To Create a 2D Graph

Graph2D Properties

- **use LOD:** Renders higher or lower detail for the object. This setting depends on the camera/object distance, and renders with marginal quality casualties if this toggle is switched on. LOD stands for Level of Detail.
- **Channel:** Affects the Channel File to search for. The file holds the vertex and other data to define the whole graph.
- **Use:** Loads the file through a Channel File or remote access.
- **Max (%):** Sets the range for rendering. For example, if the range is set to 50% the graph is cut 50% in width.
- **3D:** Enables 3D rendering of the graph, and the **Show Back** option.
    - **Show Back:** Enables the object to show the back in case the object is rotated.
- **Type:** Following types are available: Graph, Line and Bars. Take a change if you want another visual depiction. **Graph** enables the Reduction and Minimum Points settings. **Line** enables the Constant Width, Line Width (%), Reduction (%) and Minimum Points settings. Bars enables the Bars Width (%) and Bar Animation settings.
    - **Reduction (%):** Reduces points by the Douglas-Peucker algorithm. The Douglas-Peucker algorithm is an algorithm for reducing the number of points in a curve that is approximated by a series of points.
    - **Minimum Points:** Changes the number of points for rendering the graph.
    - **Constant Width:** Sets a constant width.
    - **Line Width (%):** Changes the width of the line.
    - **Reduction (%):** Reduces points by the Douglas-Peucker algorithm. The Douglas-Peucker algorithm is an algorithm for reducing the number of points in a curve that is approximated by a series of points.
    - **Minimum Points:** Changes the number of points for rendering the graph.
    - **Bar Width (%):** Changes the width of all bars.
    - **Bar Animation:** Renders each bar separately when **One by One** is selected. Renders all bars at the same time when **All together** is selected.
- **Use Colors:** Sets the color for positive and negative values. This setting is only available for graphs and bars (not line).
- **Show Frame:** Shows the frame around the graph. Useful for orientation.
- **Aspect Ratio:** Stretches the graph evenly in x-direction.
- **Refresh Data File:** Reloads the Channel File, which holds the data value for the x- and y-points.

Copyright © 2026 Vizrt

To Create a Channel File for a Graph2D

1. Create a text file (for example, **CHANNEL2**), with no file extension.
2. Add this data: `5 0 0 10 20 20 0 30 20 40 0`.
    - The first number (5) holds the number of points available in the file.
    - The other definitions are the X and Y points -> (0/0), (10/20), (20/0) etc.
3. Create this directory: *<viz data folder>\plug-in\graph2D.* and place the channel file in it.

IMPORTANT! The channel file must always be named **CHANNEL<Channel number>**.

To Create a 2D Graph

1. Create a new Container.
2. Add the Graph2D plug-in.
3. Add a Material and/or a Texture.
4. Set the **Channel** parameter to the Channel file (for example: **2** for **CHANNEL2**).
5. Click **Refresh data file** if the Channel file contents have been modified.



Copyright © 2026 Vizrt

Icosahedron



The Icosahedron plug-in creates a Platonic solid composed of twenty faces that span twelve vertices, each face of which is an equilateral triangle.

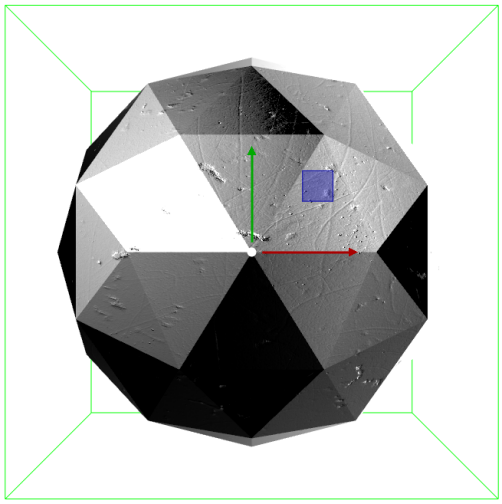
An icosahedron can be considered a rough approximation for a sphere.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Icosahedron Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Smooth:** Enables smoothing of edges.
- **Depth:** Sets the number of planes at the icosahedron. The more planes you set the more it looks like a sphere.
- **Tesselation:** Allows you to choose level of detail setting between *Low*, *Middle* and *High*.

To Create an Icosahedron



1. Create a new group and add the Icosahedron plug-in to it.
2. Add a material and/or a texture to the group.
3. Open the Icosahedron editor and disable the Smooth option, set Depth to `3` and Tesselation to `Middle`.

Copyright © 2026 Vizrt

Image FX



The Image FX plug-in enables you to create a wide variety of transitions between one or more images. The plug-in uses many different effect models that can be customized through parameters. Some properties use advanced mathematical formulas to create the effects, and not all of them can be explained meaningfully in normal words. Because of this, not all properties can be explained completely here, you need to play with some properties to see the effects they create in a given situation (what some properties are set to has impact on the effect that other properties give).

The images are split into a number of user-defined sections and every section is controlled independently based on progress maps. Any number of images of any Viz Artist supported format can be controlled using Image FX. All images are resized (if needed) to the first image size. The image transition order can be modified on the fly and images may be added to the sequence using external commands.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Image FX Properties

At the top of the editor you find a set of buttons to switch the properties.

Image

- **Add and Remove:** Images dropped onto the Add/Remove drop zone are added/removed as sub-containers of the plug-in container in the scene tree.
- **Load from Scenetree:** Loads images that are added directly as sub-containers to the Image FX plug-in container.
- **Clear all:** Removes all sub-containers to the Image FX plug-in container.

Geometry

The image is split into a number of stand-alone sections that can be modified independently.

- **Geometry Type:** Selects which kind of geometry the images are split up into.
  - **Tiles:** GL quads, the number of quads are X tessellation times Y tessellation.
  - **Mesh:** (vertices) GL quad strip, the number of vertices are (X tessellation +1) times (Y tessellation +1). Modification can apply to every vertex in the mesh.
  - **Triangles:** GL triangles, the number of triangles are X tessellation times Y tessellation times 2. Modification can apply to one triangle (3 vertices).
  - **Random Triangles:** GL triangles, triangles are generated using random points. Number of triangles is a minimum of X tessellation times the Y tessellation. Modification can apply to one triangle (three vertices).

Copyright © 2026 Vizrt

- **Random Seed:** Is relevant if you have chosen random triangles. It specifies a seed for the random number generator. Even though Viz Artist use random numbers, the layout of a specific random seed always looks the same. Press the **wireframe** button and click the **new random seed** button to see the effect.

- **Tessellation:** Sets the degree of detail (enable wireframe to view the effect).
- **Spacing:** Sets a spacing between the geometrical parts in the image.
- **Border:** Draws a border with the width defined here between single sections if spacing has a value.
- **Crop:** Enables/disables cropping of the border.
- **Alpha:** Sets an alpha value for the border.
- **Color:** Sets a color for the border.

Trigger

In the trigger section, you can select between different progress maps that defines how every section of the image is to be triggered by the global progress. Each section of the image receives its base value, the global progress is a value that changes as you animate by using one of the options in the control menu. Since every section gets a different base value, it is triggered differently by the global progress.

> **Example:** In a domino effect which starts at the bottom left the section at the bottom left receives a value of 100 and the section at the top right receives a value of 1. As the progress moves, sections with high values are influenced, fast then sections with low values. If we continue with the above example with a fade effect, the bottom left section fades out when progress is 20, but the top right section only fades out when progress is 100.

The options for progress maps are:

- **Clock:** Sets the values for the sections that are computed based on a clock hand movement. It has the following properties:
  - **Center:** Sets the center for the clock hand movement.
  - **Start Angle:** Sets the starting angle for the clock hand.
  - **Whirl:** Applies a whirl effect to the movement.
  - **Direction:** Sets the direction for the clock hand movement.
- **Domino:** Sets the values for the sections that are computed based on a domino movement. It has the following options:
  - **Order:** Sets the order in which the tiles are moved. Choose between **Lines** (Line by line movement), **Columns** (Column by column movement, **Center** (From center and outwards movement), **Bottom Left** (Bottom left tiles first and then diagonal movement towards top right corner) or **Top Left** (Top left tiles first and then diagonal movement towards bottom right).
  - **Direction:** Sets the starting point of the movement.
- **Noise:** Sets the values for the sections that are computed based on user-defined noise map created using noise algorithms (perlin noise).
  - The settings create a noise texture and the image transformations are based on this image.
  - When you edit the noise properties it is a good idea to enable the *show* button. This lets you see the images that are created and see changes instantly.
  - **Wave**, **Cloud**, **Marble** and **Noise** are predefined noise parameter settings. It is normally a good idea to start with one of these and thereafter edit until you get the required effect. The four images below show how the four noise parameters look like when the show option is enabled:

Copyright © 2026 Vizrt



- **Turbulence 1** to **3:** Sets the noise level. The settings must be *played* with to see what kind of effect you get as they influence each other. So what effect one gives depends on the state of the other.
- **Frequency:** Defines something that can be described as the “wave length” of the noise function. A large number gives a small wave length which creates a lot of changes to the image, while a small number creates a smoother image.
- **Exponent:** Controls the ratio between the white areas and the black areas without changing the shape significantly. Exponent zero give a full white picture, and as the value grows, the black area becomes larger and larger.
- **Scale X/Y:** Are quite similar to the usual texture scale, but they are working with reverse logic. In the usual texture scaling, increasing the scale is like zooming into the image, but in Image FX it works like zooming out. Also the units are different, but beside this, they are quite similar.

- **Random:** Computes the values for the sections randomly.
- **Image:** Computes the values for the sections based on pixel values of an image you drag onto the drop zone. What the function does is to create links between the animated image tiles and the pixels of the image and then use the luminance level of each pixel to set the base value for the tiles. By default 100% luminance sets high base value and 0% sets a low base value. This option allows you to generate any progress map you want by using other tools.
- **Static:** Gives all sections the same value. The complete image moves out and the other moves in.
- **Random Seed:** Specifies a seed for the random number generator. Is relevant if you have chosen random order. Even though Viz Artist uses random numbers, the animation for a specific random seed always looks the same.

Effect

The trigger section defined the order in which the tiles would be influenced by the global process. The effect section allows you define what kind of effect/transition that is used.

- **Target Image**

Copyright © 2026 Vizrt

    - **Static:** The target image is not effected at all the by the effect. A typical example would be a reveal effect where target image is “hidden” behind main image and a transition effect on the main image reveals the target image.
    - **Dynamic:** The target image is controlled by the effect. A typical example would be main image flying out of the frame and at the same time target image flying in. If a dynamic target image is used, its effect can be either being the invert of main image or can be defined separately.
- **Interpolate**
    - **Linear:** A linear interpolation is used for the transformations.
    - **Smooth:** A smooth interpolation is used for the transformations.
- **Position > Move:** You can specify a position for the source image, for the target image and a random position.
    - For the source image, which is the image that is currently shown, you specify a source position. This is the position that the image tiles move to when the animation runs.
    - If no target position is specified, the target image animates from the source position to the initial position of the source image.
    - If you specify a target value the target image moves from that position and onto the initial position of the source image.
    - Switch on **X**, **Y** and **Z** to alter the values.
    - **Absolute:** All sections move to the same end position.
    - **Relative:** The sections maintain their original relationship.
- **Position > Explode:** Final positions for the sections are calculated based on a user-controlled explode algorithm. The tiles of the image are treated as particles in an explosion and they are moved in one to three axis depending on what you specify. All tiles are thrown out of a emitter that can be defined to have any opening angle, so the particles can be sent within a narrow angle, like a canon fire, or in all direction like a explosion. The parameters are:
    - **Duration:** Sets the duration for the movement of the tiles/particles.
    - **Opening Angle:** Sets the opening angle for the emitter. Remember to choose more than one axis to get any clear effect.
    - **Angle Rotate X:** Rotates the emitter hole on the X-axis.
    - **Angle Rotate Y:** Rotates the emitter hole on the Y-axis.
    - **Force** sets the degree of force that throws the particles out in space.
    - **Force Spread %:** Randomizes the initial impels of the tiles. Spread 0 % means all items have the same impulse, 100% means a high degree of randomness.
    - **Gravity:** Sets a gravity force for the environment.
    - **Use Axis:** Selects which axes or axis to use.
- **Rotation:** Rotates the sections.
- **Scaling:** Scales the sections.
- **Alpha:** Changes the alpha value of the sections from **Alpha Start** to **Alpha End**.
  To all options (except alpha) a certain degree of user-controlled randomness can be added.

- **Random Seed:** Specifies a seed for the random number generator. Is relevant if you have a random property. Even though Viz Artist uses random numbers, the animation for a specific random seed always looks the same.
  The final effect of transformation can be a combination of effects where the timing between the different effects can be tuned using the mini-stage. Move the start and end Key Frames to time the effects.

Copyright © 2026 Vizrt

Example: A position explode effect can be combined with a rotation effect so the sections rotate as they explode. An alpha effect can be added to fade out the sections at the end.

The pivot properties have relevance when you use rotation on the tiles. The settings decide where the rotation center should be on the tiles. To see the effect, create a low tessellation transition with tiles and a 180 degree X or Y rotation. Change the pivot properties for Y pivot if you have an X rotation or the X pivot if you have an Y rotation and see the effect.

Control

The global progress value that is used to animate the sections can be controlled in different ways:

- **Auto:** When using auto mode, Image FX starts the effect as soon as the **Take** button is pressed. Useful for external control to run a sequence of images.
- **Animation:** The global progress value is animated and the effect follows the stage progress. In this mode you can combine the effect animation with other animations.
- **Global Animation:** When using a sequence of images, the complete progress animates all images.

To Add, Load and Remove Images

1. Add the images you want to create transitions between by dragging them from the Server Panel and dropping them onto the **Add** drop zone.
    - The added images are visible in the scene tree as sub-containers of the Image FX plug-in container.
2. To add images directly as sub-containers to the plug-in container you must click the **Load from Scenetree** button to make the plug-in recognize the added images.
3. Remove images by dragging them from the Server Panel and onto the **Remove** drop zone, or simply remove them directly from the scene tree.
4. To remove all, click the **Clear all** button.

Copyright © 2026 Vizrt

Multi Panel



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. Multi Panel contains the actual video walls dimensions and its alignment in the real studio.

Multi Panel 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).

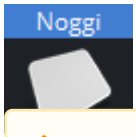
> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Multi Panel Properties

- **Mode:** Defines how the properties are being used to generate the video wall geometry.
- **Width:** Defines the width of the geometry/single segment in centimeters.
- **Height:** Defines the height of the geometry/single segment in centimeters.
- **Angle:** Sets the rotation angle around the y-axis in degrees/angle in degrees between two consecutive shapes.
- **Segments:** Sets the number of subdivisions/consecutive segments.
- **Bottom UV:** UV coordinate for the lower edge, where y = 0.
- **Top UV:** UV coordinate for the upper edge, where y = Height.
- **Left UV:** UV coordinate for the left edge, where x = 0.
- **Right UV:** UV coordinate for the right edge, where:
    - **Geometry:** x = Width
    - **Segment**: x = Width * Segments
- **UV Rotation:** Rotation Angle around the UV center.

Copyright © 2026 Vizrt

Noggi



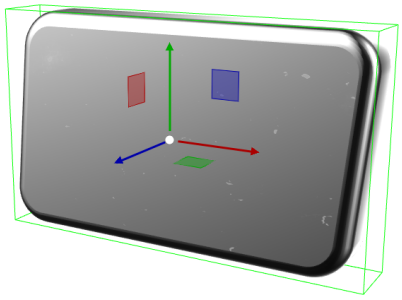
The Noggi plug-in creates geometry objects of beveled rectangles with some attributes.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Noggi Properties

- **Width:** Changes the width in X-direction.
- **Height:** Changes the height in Y-direction.
- **Bevel (top/right, top/left, bottom/left and bottom/right):** Changes the bevel on each corner.
- **Stretch:** The result is an object like a parallelogram. Stretch is in percent (= tan(alpha)). If Geometry Type is Outline.
- **Horizontal Alignment:** Changes the object alignment to left, center or right. This change affects the y-rotation.
- **Vertical Alignment:** Changes the object alignment to bottom, middle or top. This change affects the x-rotation.
- **Geometry Type:** Renders the object as area- or with outline look.
- **Inner Outline:** Changes the vertex sequence in outline mode. Inner Outline is important for face orientation when extruding the geometry.
- **Outline Width:** Changes the thickness of the rendered outline.
- **Edge Points:** Change the number of triangles to render a beveled corner. Increasing the number increases the bevel resolution.
- **Crop:** Enables/disables cropping of the object. Cropping and vertex texture are only available if Geometry Type is Area.
    - **Crop Face (Left, Right, Bottom and Top):** Crops the object on each given side.

To Create a Noggi



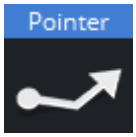
1. Create a new group and add the Noggi plug-in to it.
2. Add a material and/or a texture to the group.
3. Open the transformation editor and set Position X to `-150.0`.

Copyright © 2026 Vizrt

4. Open the Noggi editor and set the following parameters:
    - Set Width and Height to `600.0` and `100.0`.
    - Set Bevel top/right and bottom/left to `0.0`.
    - Set Bevel top/left and bottom/right to `20.0`.

Copyright © 2026 Vizrt

Pointer



The Pointer plug-in is similar to the Arrow plug-in. The main difference is that the pointer has a joint link, so it looks more like a *bend* arrow.

Pointer has two reference points which controls the pointer's position. The first point is in the joint link and the second point is at the arrow's head.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Pointer Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Point 1 X:** Sets the position of point 1 on the X-axis.
- **Point 1 Y:** Sets the position of point 1 on the Y-axis.
- **Point 2 X:** Sets the position of point 2 on the X-axis.
- **Point 2 Y:** Sets the position of point 2 on the Y-axis.
- **Width:** Sets the width of the pointer shaft.
- **Cap Style Begin:** Allows you to select the style on the pointers beginning. The options are *Flat*, *Circle* or *Arrow*.
- **Cap Style End:** Allows you to select the style on the pointers end. The options are *Flat*, *Circle* or *Arrow*.
- **Arrow Length:** Sets the length of the arrow head.
- **Arrow Width:** Sets the width of the arrow head.
- **Circle Radius:** Sets the radius of the circle, if cap style is selected.
- **Circle Segments:** Sets the resolution of the outer side of the pointer angle. Switch to wireframe to see the effect as you change the value.
- **Joint Style:** Sets the style of the pointer joint, either *Round* or *Miter*.

To Create a Pointer

Copyright © 2026 Vizrt

1. Create a new Container.
2. Add the Pointer plug-in to it.
3. Add a material and/or a texture to the group.
4. Open the Pointer editor and set the following parameters:
    - Set Point 1 X and Y to `50.0`.
    - Set Point 2 X and Y to `75.0`.
    - Set Width to `20.0`.
    - Set Cap Style Begin and End to `Circle` and `Arrow`, respectively.
    - Set Arrow Length, Width and Radius to `50.0`.

Copyright © 2026 Vizrt

Polygon

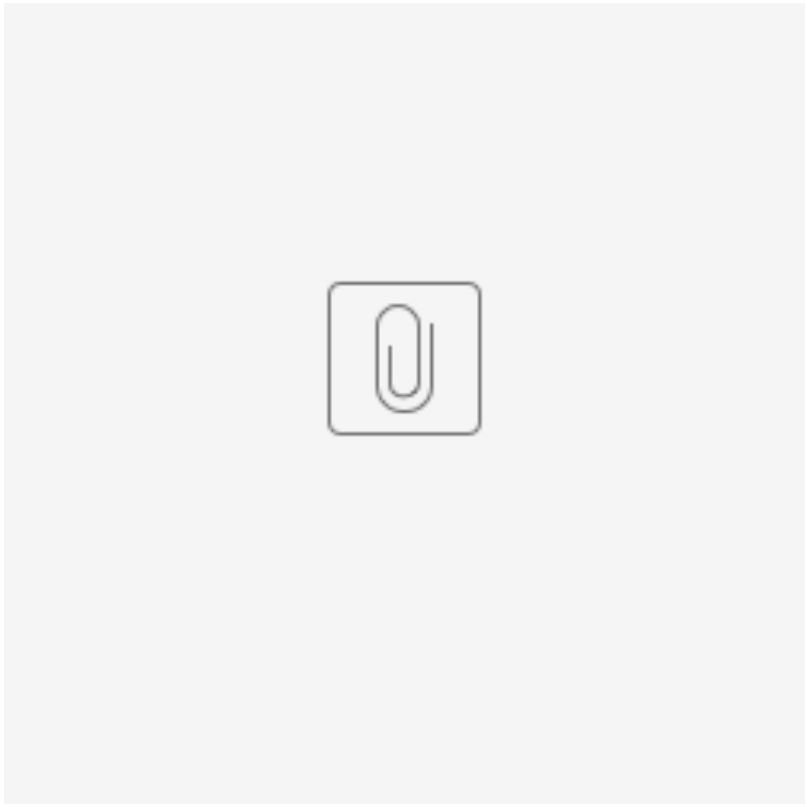


The Polygon plug-in enables the creation of user-defined polygon geometry shapes. The polygons are drawn and edited directly in the Scene Editor. Created Polygons can also be rotated to create a 3D object (see Revolve: Spline Rotation in 3D).

To create a Polygon, add the Polygon plug-in to a container and start drawing by clicking in the Scene Editor to create polygon nodes. A click, hold and drag converts the node into a spline handle for drawing curves. If an existing node is clicked on it removes the node, and if a line or spline is clicked on, between two nodes, it inserts a new node at this location. There are several Polygon plug-in Editor Shortcuts, documented in the Viz Artist User Guide. If more than one polygon is created, in the same container, and if they overlap each other, they act as masks.

Instead of drawing polygons manually, they can be imported as Adobe Illustrator files (*.*ai*) as well. If your external design tool does not support the *.*ai* format, consider importing your polygon as a geometry (see Geometry Import in Viz Artist User Guide).

Note: Only Adobe Illustrator AI 88 format files are supported.



Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

This page contains the following topics and procedures:

- Polygon Properties

Copyright © 2026 Vizrt

- Polygon Scene Editor Context Menu
- To Select an Option from the Context Menu
- Create a Polygon

Polygon Properties

- **Precision:** Sets the precision of the spline calculation.
- **Import:** Imports Adobe illustrator files. Opens a window to select a file and import. Only Adobe Illustrator AI 88 format files are supported.
- **Revolve:** See Revolve: Spline Rotation in 3D.
- **Storage:** Creates a Spline with coordinates. Press **ENTER** to apply.

To Toggle Between Polygon and Container Properties

- Click the Polygon or Transformation icon in the:
  - Scene Tree Container, or
  - The Properties panel
- In the Scene Editor:
  - Outside the polygon bounding box: Right click and select **Properties > Geometry**.
  - Inside the polygon bounding box: Right click and select Edit container.

Polygon Scene Editor Context Menu

Right-click, and hold, in the polygon green bounding box, in the Scene Editor, to open the polygon context menu. To select a menu option see To Select an Option from the Context Menu.

> **Note:** The right-click must be inside the created polygon shape box. If a click is outside the bounding box then the standard Scene Editor menu is shown.

Copyright © 2026 Vizrt



- **New spline:** Creates a new polygon in the same container.
- **Close/Open:** Opens or closes a polygon.
- **Cut:** Cuts the selected polygon or spline nodes.
- **Copy:** Copies the selected polygon or spline nodes.
- **Paste:** Pastes the selected polygon or spline nodes.
- **Mirror X/Mirror Y:** Mirrors the created shape in the X or Y axis.
- **Flip x/Flip y:** Flips the created shape in the X or Y axis.
- **Edit container:** Opens the Container editor and exit polygon edit mode. To edit the polygon again, click on the polygon icon in the properties editor or the Scene Tree.

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.

Create a Polygon

This section contains the following procedures:

To Create a Simple Spline

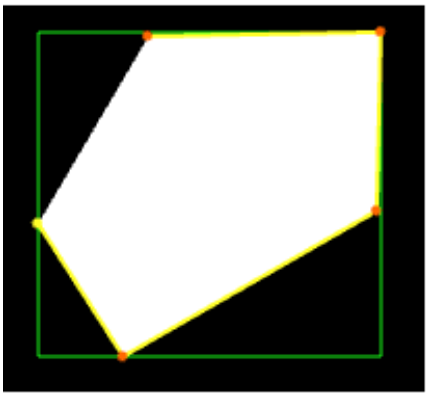
1. Create a Container.
2. Add the Polygon plug-in.
3. Click on the Polygon icon (in the Container or the Properties editor).

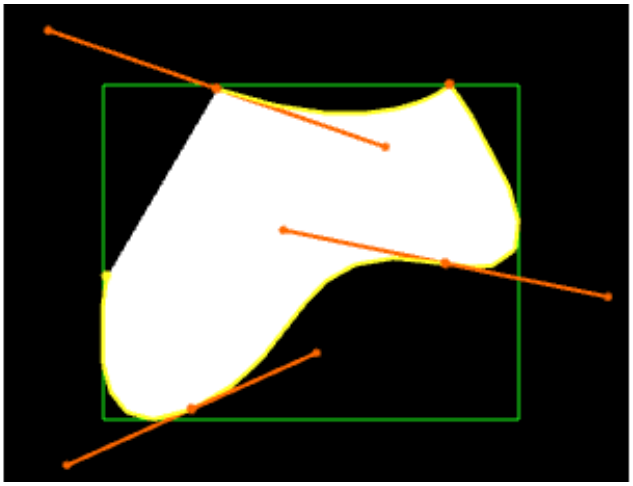
Copyright © 2026 Vizrt

4. The polygon splines are created by each click in the Scene Editor:
    - Click in the Scene Editor to create a spline point
    - Click and drag in the Scene Editor to create a spline node, complete with handles (continue the drag to increase the length of the handles)



5. While the polygon plug-in is active, each click in the Scene Editor creates more polygon points, and creates the polygon shape.





Tip: At any time press and release **DELETE** to remove the last created point. Press and hold **DELETE** to remove more than one spline, or remove the whole polygon.

Once the spline has been created it can be modified, as required, with various shortcuts (see Polygon plug-in Editor Shortcuts in the Viz Artist User Guide). List below are some common examples:

To Draw a New Spline

1. Right-click in an existing spline.

Copyright © 2026 Vizrt

2. Click **New Spline**.
3. Add intersections as detailed in To Create a Simple Spline.

To Move a Spline

1. Select the spline to be moved:
    - Press **S** and click on a spline.
2. Press **ALT** and drag.

Tip: If there is only one spline in the Container, press **ALT** and drag.

To Move More Than One Spline

1. Select the splines to be moved:
    - Press **S** and drag around the splines.
2. Press **ALT + S** and drag.

To Add or Delete a Spline Point

- Click anywhere on a spline to add another spline point.
- Click on a spline point to delete it.

To Move Spline Points

1. Press and hold **CTRL**.
2. Click on a spline point and drag. When **CTRL** is pressed move any spline points as required.

To Create or Delete a Spline Point Handle

1. Press **SHIFT:**
    - **Create:** Click on a spline point and drag.
    - **Delete:** Click on a spline point with a handle.

To Resize a Spline Point Left or Right Handle

1. Press **SHIFT**.
2. Click on a spline point handle and drag. Only the selected handle moves.

To Resize a Spline Point Handle

1. Press **CTRL**.
2. Click on a spline point handle and drag. The left and right handles move as one handle.

Note: The handle clicked on can be lengthened as required. The other handle remains the same length.

Note: If either the left or right handle has been moved individually, this sets the two handles together again.

Copyright © 2026 Vizrt

To Mirror Spline Sections

1. Right-click on a spline, or select particular nodes of it with **S**.
2. Select either:
    - **Mirror X:** Mirrors whole spline along the X axis.
    - **Mirror y:** Mirrors whole spline along Y axis.

To Flip a Spline

1. Right-click on a spline.
2. Select either:
    - **Flip x:** Flips whole spline over X-axis.
    - **Flip y:** Flips whole spline over Y-axis.

Scene Editor Grid

The Scene Editor Grid Tool-bar can be used as a great tool to help draw polygons. Together with the grid snap option polygon points can positioned, simply and accurately, in the 3D space.

Make sure that the polygon is aligned with the grid for the grid snap to work.

To Create a Spline in the Scene Editor Grid Snap

1. In the Scene Editor click  (Grid).
2. Click  (Plane Type).
3. Click  (Free Grid).
4. Click  (Snap to Grid).
5. Create a spline. Each spline point snaps to the nearest cross hairs on the grid.

Copyright © 2026 Vizrt



To Move a Spline in the Grid with Snap

1. In the Scene Editor click (Grid).
2. Click (Plane Type).
3. Click (Free Grid).
4. Click on (Move Object to Plane).
5. Press **ALT**.
6. Click on the Spline and drag.

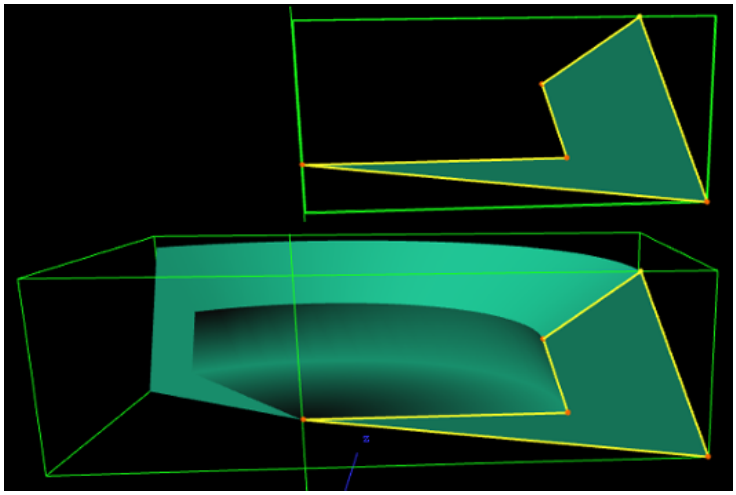
Note: If there is more than one Spline to be moved see To Move More Than One Spline.

The Spline snaps to the next set of cross hairs, and so on, as it is dragged.

Revolve: Spline Rotation in 3D

The Spline rotation creates a 3D mesh by rotating the created spline.

Copyright © 2026 Vizrt



Use the Polygon Revolve Properties panel to modify the Spline as required.

Polygon Revolve Properties

The Polygon Revolve properties are available after the first spline point has been added to the Scene Editor. Click on **Rotate** to activate.

- **Precision:** Sets the precision of the spline calculation.
- **Import:** Imports Adobe illustrator files. Opens a window to select a file and import. Only Adobe Illustrator AI 88 format files are supported.
- **Rotate:** Creates the revolve mesh and activate all its settings.
    - **Rotate in degrees box:** Sets the degree of rotation.
- **Tessellation:** Sets the degree of detail for the mesh.
- **Shading Angle:** Defines how the shading normal are aligned on the revolve object surface.
- **Culling:** Sets which side of the spline to view:
    - **Back:** View the back side of the spline.
    - **Front:** View the front side of the spline.

⚠ **Note:** To view both sides at the same time, add the Expert plug-in to the Container and active **Back Face**.

- **Revolve Around:** Sets the axis to rotate the spline around.
    - **Local Axis:** Calculates the rotation around the Container axis when the Rotation angle is changed.
    - **First to Last:** Rotates around the first and last spline point (**Revolve Axis** is grayed out) when the Rotation angle is changed.
- **Revolve Axis:** Sets which axis to rotate around:
    - **Y:** Rotates around the Y axis.
    - **X:** Rotates around the X axis.
- **Direction:** Sets which direction to view the rotation:
    - **Right-hand:** View from the right hand side.
    - **Left-hand:** View from the left hand side.
- **Show Hull:** Shows the rotated shape (1 and 2).
- **Show Caps:** Shows the original polygon shape at the beginning and end of the rotation (1 and 3).

Copyright © 2026 Vizrt



To Rotate a Spline

1. Create a spline.
2. Click on **Rotate**.
3. Modify the created 3D object with the Polygon Revolve Properties.

See Also

- Revolve: Spline Rotation in 3D

Copyright © 2026 Vizrt

Quad



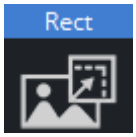
Quad is a very efficient basic 1x1 geometry used mainly for holding textures. It only renders a basic set of vertices and texture coordinates, no normals, binormals, tangents. Its main purpose is to hold basic images with Phong Materials. By default, it renders a 1x1 pixel surface and need to be scaled up correctly.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Copyright © 2026 Vizrt

Rect



The Rectangle geometry is the default geometry to show textures.

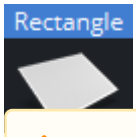
It is added automatically by dragging any image to the scene tree or editor. It does not have any user visible options.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Copyright © 2026 Vizrt

Rectangle



The Rectangle plug-in creates a 2D rectangle with some attributes.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Rectangle Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Tessellation:** Sets the degree of detail.
- **Bevel:** Enables and sets the degree of bevel at the rectangle edges. Bevel may have an effect on the height of the rectangle. For example, if Height is set to `100.0` and Bevel is set to `51.0`, the effective height is `102.0`.
- **Width:** Sets the width of the rectangle.
- **Height:** Sets the height of the rectangle.
- **Corners:** Sets the number of corners the rectangle is build up from, it become visible, when you set a bevel on the rectangle.
- **Use Vertexcolors:** Enables you to set the four color parameters below. If this option is set, the rectangle does not respond to container color anymore.
- **Upper Left:** Sets the color and alpha of the upper left section of the rectangle. Press the colored button to enable color editing on that section and change the color in the color editor below. You can also drag a material from the Server Panel and onto the color icon.
- **Lower left:** Sets the color and alpha of the lower left section of the rectangle. Press the colored button to enable color editing on that section and change the color in the color editor below. You can also drag a material from the Server Panel and onto the color icon.
- **Upper Right:** Sets the color and alpha of the upper right section of the rectangle. Press the colored button to enable color editing on that section and change the color in the color editor below. You can also drag a material from the Server Panel and onto the color icon.
- **Lower Right:** Sets the color and alpha of the lower right section of the rectangle. Press the colored button to enable color editing on that section and change the color in the color editor below. You can also drag a material from the Server Panel and onto the color icon.

Copyright © 2026 Vizrt



- **Mode:** Sets the orientation of the plane to XY, XZ or YZ.

Copyright © 2026 Vizrt

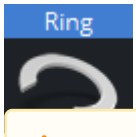
To Create a Rectangle



1. Create a new group and add the Rectangle plug-in to it.
2. Add a material and/or a texture to the group, and/or open the Rectangle plug-in editor and enable and set the vertex colors.
3. Open the transformation editor and set Position Y to `-150.0`.
4. Open the Rectangle editor and set the following parameters:
    - Set Bevel to `15.0`.
    - Set Width to `600.0`.
    - Set Height to `30.0`.

Copyright © 2026 Vizrt

Ring



The Ring plug-in creates an open or closed ring with some attributes.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Ring Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Corners:** Sets the number of corners on the ring.
- **Inner Radius:** Sets the size of the inner radius.
- **Outer Radius:** Sets the size of the outer radius.
- **Rotation:** Rotates the ring like a “turning wheel”. This is typically used in combination with an open angle, to place the angle at the required point.
- **Angle:** Defines an open angle on the ring.
- **Mode:** Changes the object view coordinate. Available options are *XY*, *XZ* and *YZ*.

Copyright © 2026 Vizrt

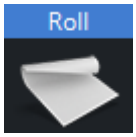
To Create a Ring



1. Create a new group and add the Ring plug-in to it.
2. Add a material and/or a texture to the group.
3. Open the Ring plug-in editor and **animate** the Angle.
    - Set Angle to `0.0` (60/50 fps).
    - Set Angle to `360.0` (120/100 fps).

Copyright © 2026 Vizrt

Roll



The Roll plug-in allows you to create a rectangle that rolls up in different ways. This is typically used for different kinds of unveiling.

To make the roll look realistic, the expert plug-in must be added to the container to enable visualization of backface and two-sided lightening.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Roll Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Size X:** Sets the size of the rectangle on the X-axis.
- **Size Y:** Sets the size of the rectangle on the Y-axis.
- **Tesselation:** Sets the level of detail.
- **Angle:** Sets the angle of the rolling up.
- **Diameter:** Sets the diameter of the roll.
- **Helix factor:** Defines how tight the roll is to be rolled up. With a high helix factor, there is space between the layers of the roll.
- **Drawing:** Selects how the roll rectangle is to be drawn up. If you switch to wireframe mode and toggle between the two settings you see the difference:
  - **Segments:** Draws the rectangle up by using many long lines stretching from edge to edge. This mode does not look as good as **Mesh** but it demands less performance of the render engine.
  - **Mesh:** Draws the rectangle using many small equally sized triangles. The number and size depends on the level of tessellation selected. This drawing mode can be heavy to render if the tessellation is set high. **Mesh** creates a better lightning of the roll rectangle.
- **Style:** Sets the style of the roll:
  - **Cigar roll:** Makes the rectangle roll up in a normal manner.
  - **Wipe:** Rolls the carpet up without creating a roll.
- **Outline:** Enables an outline view for the roll. You must in addition add the expert plug-in and enable the outline setting there.
- **Time:** Sets the time for the roll sequence. Animate this value to create an animation of the roll object. To learn more about animation, see the **Create Animations** section of the Viz Artist User Guide.

Copyright © 2026 Vizrt

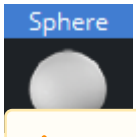
To Create a Roll



1. Create a new group and add the Roll plug-in to it.
2. Add the Expert plug-in to the group.
3. Open the Expert editor and enable (`On`) the Back Face property.
4. Add a material and/or a texture to the group.
5. Open the Roll editor and set the following parameters:
    - Set Size X to `150.0` .
    - Set Size Y to `20.0` .
    - Set Angle to `10.0` .
6. Adjust the Time parameter to see the roll effect.

Copyright © 2026 Vizrt

Sphere



The Sphere plug-in creates a simple sphere with some attributes.

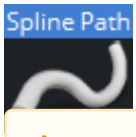
Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Sphere Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Tessellation:** Sets the degree of detail for the sphere.
- **Rotation:** Allows you to rotate the sphere around its Y-axis.
- **Open Angle:** Allows you to create an open angle in the sphere.
- **Texture:** Allows you to select between three different positions of the texture mapping coordinates when a vertex texture is mapped onto the sphere. To visualize the effect of the different settings, add a plain texture onto the sphere, like a chessboard image and set the texture mapping to vertex in the texture editor. Create an opening in the sphere and change the sphere texture parameters. Increase and decrease the opening angle with the different settings enabled and the effect should be visible.
    - **Absolute:** Compresses and expands the texture mapping coordinates as the opening of the sphere changes.
    - **Relative:** Cuts/adds texture coordinates as the opening changes. The cut/add is done on the one side of the opening angle. The coordinates are anchored at the other side.
    - **Centered:** Does the same as the relative mapping, but cuts/adds coordinates from both sides. The texture is anchored in a centered position on the sphere. The center of the sphere is 180° from the point where the opening angle starts.

Copyright © 2026 Vizrt

Spline Path



The Spline Path plug-in creates complex paths with an arrow look.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Spline Path Properties

- **use LOD:** Enables/disables dynamic level of detail.
- **Node:** Defines the node index. For example, node 3 or node 25.
- **X, Y and Z:** Sets the 3D point to define a node.
- **Length:** Stretches the entire path in X-direction.
- **Width:** Stretches the entire path in Y-direction.
- **Threshold:** Increase this value if you do not need a high path accuracy but more performance.
- **Add, Move, Remove and Clear:** Add adds a new node with an specific index. Move moves an existing node to another X, Y or Z-point. Remove removes an existing node. Clear clears all nodes.
- **Mode:** The following options are available:
  - **Tube:** Renders the entire object.
  - **Band XY:** Renders only the front side.
  - **Band XZ:** Renders only the top side.
  - **Band YZ:** Renders only the left side.
- **Arrow:** Creates an array look.
- **Arrow Length:** Sets the arrow length (only when *Arrow* is set to `On`).
- **Arrow Width:** Sets the arrow width (only when *Arrow* is set to `On`).
- **Texture Stretching Mode:** (only behave as described for the *Band* modes.)
  - **"Width+Length" (fit both directions)**: Net effect: exactly **one copy of the texture is stretched over the whole ribbon/tube**. The image always fills the object, but its aspect ratio distorts as you add nodes or change Width — a long path squashes the texture along V.
  - **"Width" (fit across, tile along):** Net effect: **fit across the width, repeat along the length** — one texture tile per 10 units of path length. Growing the path adds more repeats instead of stretching the existing ones, so a dashed/arrow/tread texture keeps a constant length-wise scale.
  - **"-" (no fit; absolute scale in both directions):** Net effect: **the texture tiles in both directions at a fixed world scale of one tile per 10×10 units**. Nothing is stretched — the texture keeps its aspect ratio, and both Width and Length changes only reveal more or fewer repeats.

Copyright © 2026 Vizrt

To Create a Spline Path



1. Create a new group container.
2. Add the Spline Path plug-in.
3. Add the Expert plug-in.
4. Open the Expert editor.
5. Set **Back Face** to On.
6. Add a Material and/or a Texture to the group.
7. Open the Spline Path editor.
8. Set **Node** to `0` / **X** to `50.0` / **Y** to `50.0`.
9. Click Add.
10. Set **Node** to `1` / **X** to `150.0` / **Y** to `25.0`.
11. Click Add.
12. Set **Node** to `2` / **X** to `250.0` / **Y** to `10.0`.
13. Click Add.
14. Set **Node** to `3`, **X** to `350.0`, **Y** to `45.0`.
15. Click Add.

Copyright © 2026 Vizrt

Spline Strip



The Spline Strip plug-in creates 3D interactive loft extrusions which are able to be animated in their length and surface.

You can use modifiable lines, circles or rectangles for the extrusion. Arrows and terrain alignment are also supported.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

Spline Strip Properties

- **Path Node Index:** Defines where Path Nodes get added, moved or removed. The first node within a path owns the index 0. If you add nodes at the end, then the Path Node Index is increased automatically to provide a simple path generation.
- **X, Y and Z:** Sets the Path Node's X, Y and Z-position.
- **TC:** Sets the Path Node's TC-position.
- **Add Node:** Adds a path node at the current Path Node Index with position X, Y, Z and TC.
- **Move Node:** Moves the current path node to your required X, Y, Z and TC position.
- **Remove Node:** Removes a path node at the current Path Node Index.
- **Interactive Mode:** If you have finished the process of setting path nodes then you are able to edit it simpler by mouse. Just drag and drop the shown nodes to a new position. Left Mouse Button: move node in XY. Middle Mouse Button: move node in XZ.
- **Interactive Orientation:** World, Local, View. Move nodes interactive with world-, local-, or viewpoint-coordinates.
- **Node Size:** Sets the interactive node's size.
- **Storage:** Is used for storing your defined path nodes in a #-separated string. It is possible to edit nodes directly in the Storage Text field.
- **Clear Storage:** Clears the spline strip object and its storage.
- **Use Time Code:** Activates Time Code functionality.
- **TC Current:** Contains the current Time Code of an external Time Code source.
- **TC Start:** Sets the starting point of your Time Code animation.
- **TC Reference:** If you are using more than one spline strip then they probably won't start all at the same Time Code position. But you can change that by a simple time shift with the Help of TC Reference.
- **Path Steps:** Defines SplineStrip tessellation.
- **Use Threshold:** You can optimize your spline strip with that option. Path Steps are added at necessary positions only. So the number of Path Steps increases with a decreasing Threshold.
- **Threshold:** Sets the Threshold for Path Steps generation.
- **Path Start:** Sets the starting point of the path geometry.

Copyright © 2026 Vizrt

- **Path Length:** It defines the length of our spline strip object. Feel free to animate that value!
- **Path Alignment:** Fits the spline strip to Terrain plug-ins in Viz Artist/Engine (for example, use Terrain for drawing paths over a mountain). Available options are; Free and Topography.
- **Surface Offset:** Shifts Path on Y axis.
- **Use Surface Normal:** If activated then the surface offset is computed via terrain normals.
- **Show Path:** Shows the Path which you have defined at the beginning. This option is quite helpful when you are editing Path Nodes.
- **Shape Type:** If you activate the Organic Mode, your spline strip starts to wobble in dependency of the adjusted Organic Amplitude, Organic Speed, and Organic Period's Length. Available options are; Static and Organic.
- **Shape:** Defines the shape used for the loft extrusion. Available options are; Line, Circle and Rectangle.
- **Shape Width:** Sets the shape's width.
- **Shape Height:** Sets the shape's height.
- **Shape Tesselation:** Defines how many steps are used to draw the required shape.
- **Shape Rotation:** You are able to move the spline strip shape up to 360 degrees.
- **Shape Orientation:** When set to Free, shapes are rotated in dependency of the adjusted Shape Rotation value. When set to XZ, YZ or YX, shapes are auto rotated to obtain a constant SplineStrip width in the required aspect. When set to Terrain, shapes are rotated to fit the terrain beneath them.
- **Translate Vertices To Terrain:** Moves every single vertex to fit the terrain. This mode only makes sense if you are using Line extrusion.
- **Flip Normals:** Flips spline strip normals.
- **Backface Culling:** Switch off drawing spline strip's backside. This property is only available in Classic Render Pipeline.
- **Organic Amplitude:** Adjusts the animated diameter reduction from 0% to 100%.
- **Organic Speed:** Changes speed of organic mode animation. Also negative values are allowed.
- **Organic Period's Length:** Sets the diameter reduction function's period length.
- **Termination:** Closes our spline strip object at the Beginning, End or at Both sides. Available options are; None, Beginning, End and Both.
- **Arrow:** Puts an Arrow to the Beginning, End or Both sides. Available options are; None, Beginning, End, Both
- **Object Width:** Changes arrow width.
- **Object Height:** Changes arrow height.
- **Object Length:** Changes arrow length.
- **Object Tesselation:** Changes arrow tessellation.
- **Generate Texture Coordinates:** Activate this function if you are using vertex maps on your spline strip.
- **Stretch Texture to Path Length:** If activated then the map is stretched to the actual path length.
- **Draw Mode:** Switch between fill- and wire-frame mode. Available options are Fill and Wire. This property is only available in the Classic Render Pipeline.
- **Buffer Geometry:** The whole spline strip data is buffered to memory. If you use spline strips with a huge number of vertices then Buffer Geometry is able to improve performance. Activate this mode only if you do not use any kind of animation in your current spline strip. This property is only available in the Classic Render Pipeline.
- **Color:** Defines path's and interactive-mode-node's color. Color is also used if Wire is activated and no maps are applied onto your spline strip object.
- **Child Distance:** Defines distance between path end and child container.
- **Rotate Child:** Activates child container rotation.

Copyright © 2026 Vizrt

To Create a Spline Strip

1. Create a new Container.
2. Add the Spline Strip plug-in.
3. Open the Spline Strip editor.
4. Use all default values.
5. Click **Add Node**.
6. Change **X** to `100.0`.
7. Click **Add Node**.
8. Change **X** to `200.0`, **Y** to `50.0`.
9. Click **Add Node**.
10. Set **Shape Width** and **Shape Height** to `50.0`.
11. Set **Shape** to **Circle**.
12. Set **Shape Tesselation** to `20`.
13. Set **Path Length** to `150`.



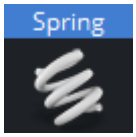
14. Optionally, click **Interactive Mode**.
15. Change **Node Size** to for example `15.0`.
16. Activate script and plug-in events in the Scene Editor by clicking the **E** button.
17. Drag and drop the small Cubes as required.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

Spring



The Spring plug-in creates a spiral spring geometry.

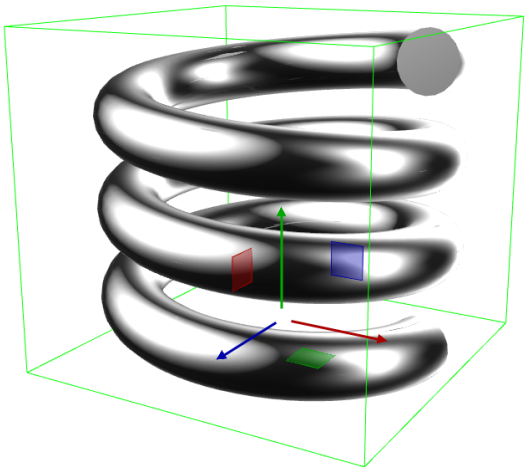
Be aware this object can easily be very heavy to render if a high level of tessellation and corners are selected.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Spring Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Tessellation:** Sets the degree of detail for the spring.
- **Corners:** Sets the number of corners the spring should have.
- **Gradient:** Sets the size of the spring gradient.
- **Diameter torus:** Sets the diameter of the springs.
- **Diameter cross section:** Sets the diameter of the spring cord.
- **Show Bottom:** Shows/hides bottom of the spring.
- **Show Top:** Shows/hides Top.
- **Center:** Sets the location of the spring center: Either *Center*, *Bottom* or *Top*.

To Create a Spring



1. Create a new group and add the Spring plug-in to it.
2. Open the Spring editor and set the following parameters:
    - Set Turns to `6.0`.
    - Set Gradient to `20.0`.
    - Set Diameter torus to `200.0`.

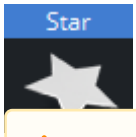
Copyright © 2026 Vizrt

- Set Diameter cross section to `100.0`.
- Set Center to `Center`.

3. Add a material and/or a texture to it.

Copyright © 2026 Vizrt

Star



The Star plug-in creates a star with different attributes.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Star Properties

- **Corners/Tips:** Sets the number of corners on the inner circle of the star (equals the number of tips on the outer circle of the Star). Default: 5, Min: 3, Max: 300.
- **Star Radius:** Sets the radius of the outer circle (where the tips of the star are located). Default: 50, Min: 1, Max: 200.
- **Star Inner Radius Factor:** Determines the factor of the outer radius to the inner radius of the star. A value of 1.0 means that the inner radius of the star is identical to the **Star Radius** value. Default: 2, Min: 0.25, Max: 50.
- **Hole:** Places a hole in the Star geometry. Default: 0 (`Off`).
- **Hole Radius:** Sets the radius of the outer circle where the tips of the hole are located. Default: 25, Min: 1, Max: 200.
- **Hole Inner Radius Factor:** Determines the ratio of the outer radius to the inner radius of the hole. A value of 1.0 means that the inner radius of the hole is identical to the Hole Radius. Default: 2, Min: 1, Max: 50.

Copyright © 2026 Vizrt

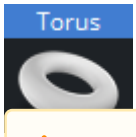
To Create a Star



1. Create a new container.
2. Add the Star plug-in to it (1: Default Star).
3. Open the Star editor.
4. Modify the Star parameters, for example (2: Modified Star).
    - Corners: `10`.
    - Star Radius: `60`.
    - Star Inner Radius Factor: `6.0`.
    - Hole: `On`.
    - Hole radius: `50`.
    - Hole Inner Radius Factor: `9.0`.
5. If required, add a material and/or a texture.

Copyright © 2026 Vizrt

Torus



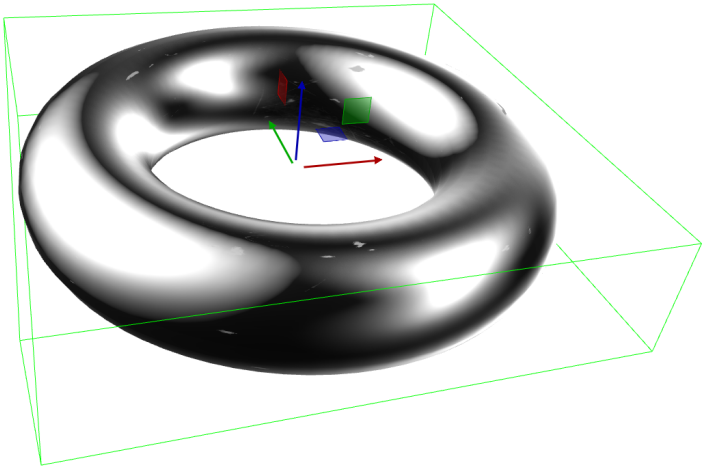
The Torus plug-in creates a torus with different attributes.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Torus Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Tesselation:** Sets the degree of detail for the torus.
- **Corners:** Sets the number of corners on the torus.
- **Radius:** Sets the radius of the torus from its center in the middle of the hole.
- **Radius cross section:** Sets the radius of the torus cross section or “tube”.

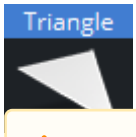
To Create a Torus



1. Create a new group and add the Torus plug-in to it.
2. Open the Torus editor and set the following parameters:
    - Set Radius to `50.0`.
    - Set Radius cross section `20.0`.
3. Add a material and/or a texture to it.

Copyright © 2026 Vizrt

Triangle



The Triangle plug-in creates a simple 2D triangle with some attributes.

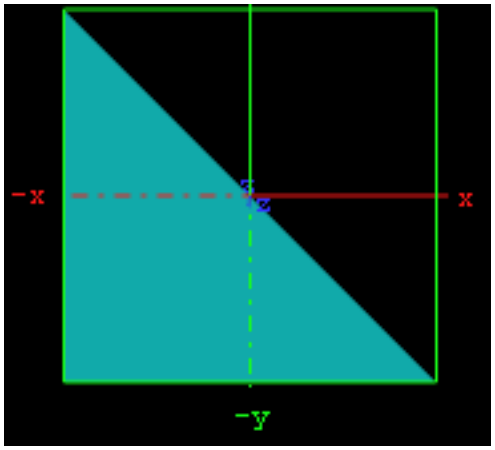
Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Triangle Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **X1:** Sets the x-coordinate for the first corner of the triangle.
- **Y1:** Sets the y-coordinate for the first corner of the triangle.
- **X2:** Sets the x-coordinate for the second corner of the triangle.
- **Y2:** Sets the y-coordinate for the second corner of the triangle.
- **X3:** Sets the x-coordinate for the third corner of the triangle.
- **Y3:** Sets the x-coordinate for the third corner of the triangle.
- **X1=X2:** Sets the same x-coordinate for the first and second corners of the triangle.
- **Y1=Y2:** Sets the same y-coordinate for the first and second corners of the triangle.
- **Y1=Y3:** Sets the same y-coordinate for the first and third corners of the triangle.
- **Y2=Y3:** Sets the same y-coordinate for the second and third corners of the triangle.
- **Use Vertexcolors:** Enables/disables setting color parameters for the corners of the triangle. If this option is set, the triangle does not respond to container color anymore.
    - **Color 1:** Sets the color and alpha of the first corner of the triangle. Press the colored button to enable color editing on that section and change the color in the color editor below. You can also drag a material from the Server Panel and onto the color icon.
    - **Color 2:** Sets the color and alpha of the second corner of the triangle. Press the colored button to enable color editing on that section and change the color in the color editor below. You can also drag a material from the Server Panel and onto the color icon.
    - **Color 3:** Sets the color and alpha of the third corner of the triangle. Press the colored button to enable color editing on that section and change the color in the color editor below. You can also drag a material from the Server Panel and onto the color icon.

Copyright © 2026 Vizrt

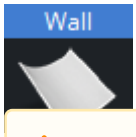
To Create a Triangle



1. Create a new group and add the Triangle plug-in to it.
2. Add a material and/or a texture to the group, and/or open the Rectangle plug-in editor and enable and set the vertex colors.
3. Open the transformation editor and set Position Y to `-150.0`.
4. Open the Triangle editor and set the following parameters:

Copyright © 2026 Vizrt

Wall



The Wall plug-in creates a curved wall object.

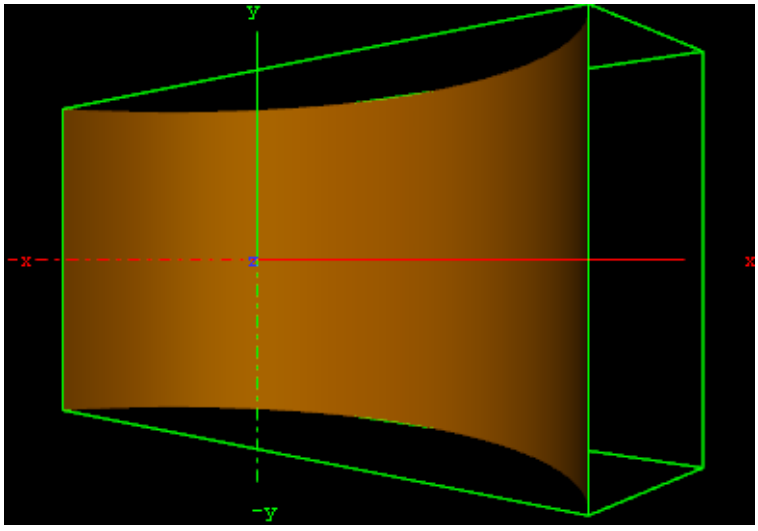
The size and curving of the wall can be customized.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Wall Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Height:** Sets the height of the wall.
- **Width:** Sets the width of the wall.
- **Depth:** Sets the depth of the wall curving. The higher this value is set, the more curved the wall is.
- **Angle:** Sets the angle of the walls curving. Use this parameter together with Depth to achieve the desired curving of the wall.
- **Corners:** Sets the number of internal corners the wall should be built up from.
- **Center:** Sets the center axis of the object. Choose between *Center*, *Bottom* or *Top*.
- **Front:** Lets you switch between front and back.

To Create a Wall

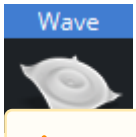


1. Create a new group and add the Wall plug-in to it.
2. Open the Wall editor and set the following parameters:
    - Set Height to `200.0`.
    - Set Width to `400.0`.
    - Set Depth to `400.0`.
    - Set Center to `Center`.
3. Add a material and/or a texture to it.

Copyright © 2026 Vizrt

4. Open the Transformation Editor and set Rotation Y to `-45.0`.

Wave



The Wave plug-in allows you to create a wave sequence on a flat surface.

The properties of the sequence are visualized in a graphical display in the wave editor.

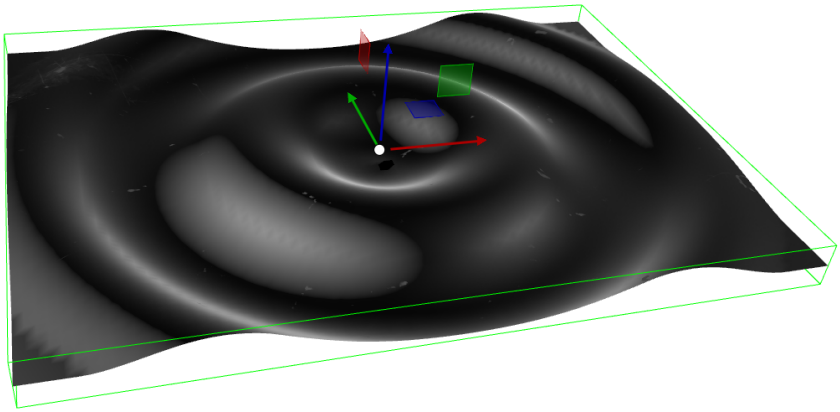
Note: This plug-in is located in: Plugins -> Geom plug-ins -> Default

Wave Properties

- **Size X** and **Y:** Sets the size of the rectangle where the waves show.
- **Tesselation:** The amount of triangles used to create the geometry. As higher as more calculations need to be performed.
- **Geom Type:** Allows you to select between **Circular** and **Linear** waves.
- **Stretch:** Allows you to customize the form of the circular waves. By altering the stretch **X** and **Y** values, you can for instance obtain an oval or a compressed form of the waves.
- **Angle:** Sets the angle of the waves when a linear mode is chosen.
- **Reference Point:** Sets the position of the waves starting point.
- **Offset:** Sets an offset value of the **Ref.Pnt**.
- **Damping Mode:** Enables damping of the waves. The amplitude of the wave is reduced by the damp length calculated from the center of the wave.
- **Damping Length:** Sets the length the distance that is used to damp the waves in the **Damp. mode** function.
- **Wavelength:** Sets the length of the waves.
- **Amplitude:** Amplifies the waves, resulting in both bigger and higher waves.
- **Time:** Is the time-line for a wave sequence. You see a thin red vertical line moving in the graphical display as you alter the time value. You can also alter the value by dragging the red line.
- **Invert OFF ON:** Inverts the wave curves.
- **Position X** and **Y:** Allows you to edit the position values of the wave control points. Select a point by clicking on it. Its color changes to red to show that it is selected. Now you can drag it around.
- **Envelope Generator:** Draws the wave manually when enabled. When disabled the wave is built by a sine wave function.
- **Mirror:** Allows you to decide whether the points controlling the crest of the waves should be edited in a locked manner, with the points controlling the troughs of the waves or conversely, if they are to be edited separately.

Copyright © 2026 Vizrt

To Create a Wave



1. Create a new group and add the Wave plug-in to it.
2. Open the Transformation Editor and set Rotation X to `-75.0`.
3. Add a material and/or a texture to it.
4. Open the Wave editor and set the following parameters:
    - Set Size X and Y to `350.0`.
    - Set W.-Length to `8.0`.
5. Click the **Lock** button to unlock the spline curve editor.
6. Play with the spline curve handles, Time, Position X and Y values to see how the wave behaves with different settings.

Copyright © 2026 Vizrt

2.3.2 Dynamics Geometry Plug-ins

The following Geometry plug-ins are located in the Dynamics folder:

- Cloth
- Cloth Flag
- Flag

Cloth



The Cloth plug-in provides a simulation of an elastic vertex system. The vertex system can be configured with many different parameters. Most of the parameters depend on each other, and some must be set in conjunction with others. The stress of the system is mentioned below, and the rule is that the greater the stress is the greater the chance the system becomes unstable. With low stress factors the system is completely stable.

Adding the Cloth plug-in to a container also adds the Expert plug-in.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Dynamics

This page contains the following topics and procedures:

- Cloth Properties
- To Create a Cloth Effect

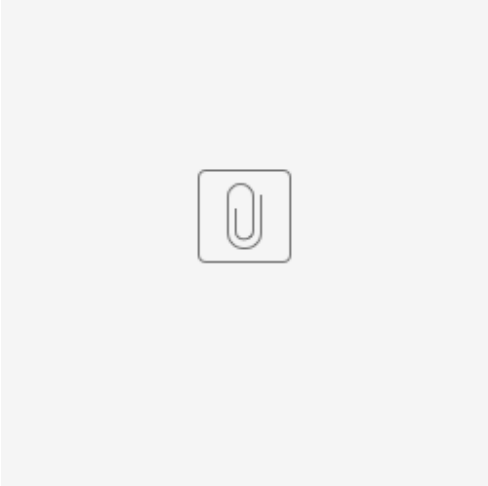
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Cloth Properties

- **Reinitialize:** Restarts rendering of the object.
- **Elastic:** Changes the elastic properties of the object. If set it to `0.1`, the object behavior resembles that of latex, and at `1.0` more like cloth.
- **DampingFct:** Specifies how much speed each vertex should loose. The lower this factor is the less speed the object loses.
- **Pause:** Pauses all calculations and freezes the system's animation.
- **Tessellation direction U and V:** Sets the tessellation in the two directions on the objects' surface. The calculation time needed increases with more vertices.
- **Speed:** Determines how many calculations are needed per frame. The less calculation done the slower the object moves. If the number of calculations are increased, the object moves faster.
- **Wind X, Y and Z:** Gives the speed and direction of the wind affecting the object.
- **TurbulenceSpeed:** Gives the speed and direction of the wind turbulence affecting the object.
- **TurbulenceFactor:** Multiplies the wind speed.
- **Gravity X, Y and Z:** Defines gravity affecting all vertices of the object.
- **Scaling u (width):** Changes the width of the rectangle which is including the animated cloth without increasing the number of triangles.
- **Scaling v (height):** Changes the height of the animated rectangle which is including the animated cloth without increasing the number of triangles.
- **Draw normals:** Shows the normals for each triangle.
- **Draw force:** Shows the appealing wind in color.

Copyright © 2026 Vizrt

To Create a Cloth Effect



1. Add the **Cloth** plug-in to a Container.
2. Add a **material** to the same Container.
3. Set **WindX** to `2.0`.
4. Click the **Reinitialize** button to see the result.

See Also

- Expert

Cloth Flag



The Cloth Flag plug-in provides a simulation of an elastic vertex object. The vertex object can be configured with many different parameters.

Most of the parameters depend on each other, and some must be set in conjunction with others.

Adding the Cloth Flag plug-in to a container also adds the Expert plug-in.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Dynamics

This page contains the following topics and procedures:

- Cloth Flag General Properties
- Cloth Flag Geometry Settings
- Cloth Flag Move Settings
- Cloth Flag Advanced Settings

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Cloth Flag General Properties

- **Wind Rotation, Elevation and Force:** These three factors combined gives the speed and direction of the wind affecting the flag. The allowed values are:
  - **Wind Rotation:** From `-720.0` to `720.0`
  - **Wind Elevation:** From `-90.0` to `90.0`
  - **Wind Force:** From `0.0001` to `100.0`
- **Gravity Direction:** Vectors that define in which direction gravity is affecting the vertices of the flag:
  - **U:** Horizontal.
  - **V:** Vertical.
  - **None:** Gravity is not applied.
- **Pause:** Setting **Pause** to `On` pauses all calculations performed and the animation in both the scene editor and *On Air* output.
- **Reinitialize:** Restarts rendering of the object.

Cloth Flag Geometry Settings

- **Tessellation direction U and V:** Sets the tessellation in the two directions on the systems surface. Calculation time increases with more vertices.
- **Scaling U (width):** Changes the width of the rectangle which is including the animated cloth without increasing the number of triangles.
- **Scaling V (height):** Changes the height of the animated rectangle which is including the animated cloth without increasing the number of triangles.

Copyright © 2026 Vizrt

- **Reinitialize:** Restarts rendering of the object.

Cloth Flag Move Settings

- **Translate X, Y and Z:** Offsets the position output using the position values set for the container.
- **Rotate X, Y and Z:** Offsets the rotation output using the rotation values set for the container.
- **Scale:** Offsets the scale output using the scale value set for the container.
- **Reinitialize:** Restarts rendering of the object.

Cloth Flag Advanced Settings

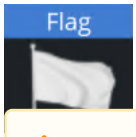
- **Turbulence Speed:** Determines how fast the direction of speed oscillates.
- **Turbulence Factor:** Multiplies wind speed.
- **Elastic:** Changes the elastic properties of the system. If set to `0.1`, the rendered output has properties resembling latex, and at `1.0` more like cloth.
- **Damping:** Specifies how much speed each vertex should lose.
- **Simulation Quality:** Reduces or increases the number of calculations performed for the plug-in, for performance reasons.
- **Wireframe:** Shows the wireframe for the plug-in.
- **Draw normals:** Shows the normals for each triangle.
- **Draw force:** Shows the appealing wind in color.
- **Reinitialize:** Restarts rendering of the object.

See Also

- Expert

Copyright © 2026 Vizrt

Flag



The Flag plug-in provides a simpler simulation of an elastic vertex object than the Cloth Flag plug-in.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Dynamics

Flag General Properties

- **Width:** Defines the width of the object. Accepted values are from `0.0` to `200`.
- **Height:** Defines the height of the object. Accepted values are from `0.0` to `200`.
- **Amount:** Sets the intensity of the effect. Accepted values are from `0.0` to `200`.
- **Wavelength:** Defines the length between the animated waves. Accepted values are from `0.0` to `200`.
- **Wave Speed:** Defines the speed of the animated waves. Accepted values are from `-100` to `100`.
- **Turbulence Speed:** Defines the speed of the turbulence in the animation. Accepted values are from `-100` to `100`.
- **Y Stretch:** Defines the anisotropic value of the Y-axis. Accepted values are from `0.2` to `100`.
- **Tessellation X:** Defines the amount of tessellation on the X-axis. Accepted values are from `1` to `100`.
- **Tessellation Y:** Defines the amount of tessellation on the Y-axis. Accepted values are from `1` to `100`.
- **Plane:** Selects the dimensional plane for the rendered object.
- **Fixed Edge:** Selects which edge should be fixed, or none. The fixed edge acts as a boundary for the animation.
- **Direction:** Automates the animation direction angle if set to `Auto` (default).
- **Dir Angle:** If **Direction** is set to `Custom`, provides the designer with the option of defining the animation direction angle. Accepted values are from `-720` to `720`.

Copyright © 2026 Vizrt

2.3.3 Visual Data Tools Geometry Plug-ins

Overview

Visual Data Tools (VDT) show any kind of statistic data. All the VDT plug-ins use the Shared Memory transport channel.

Shared Memory is a map which has user defined variables, indexed by a string, also known as a key-value pair. There are three types of Shared Memory Maps.

For more information about Shared Memory see the **Shared Memory (SHM)** section of the Viz Artist User Guide.

The VDT plug-in suite consists of:

- **Geometry plug-ins:** Used to draw the chart (detailed in this section).
- **Basic Container Plug-ins:** Used to label, data scale, etc., the chart.

To learn more about the use of Visual Data Tools, download a tutorial on www.vizrt.com under the Training section.

The following Geometry plug-ins are located in the Visual Data Tools folder:

- Area Chart
- Bar Chart
- Line Chart
- Pie Chart
- Scatter Chart
- Stock Chart

See Also

- Basic Container Plug-ins
- Basic Geometry Plug-ins
- Advanced Bar Chart Creation
- Area Stack
- Bar Stack
- Data Fit
- Data Import
- Data Label
- Data Storage
- Line Stack
- **Scripting** section of the Viz Artist User Guide.
- Tutorial on Viz University

Copyright © 2026 Vizrt

Area Chart



The Area Chart plug-in draws an area chart, filled with data out of a Shared Memory Map.

You can use delimited strings or arrays for data transfer via Scene, Global or Distributed Shared Memory.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

This page contains the following topics and procedures:

- Area Chart Properties
- To Create a Scene with Area Chart
- To Create a Scene with Area Chart and Data Storage
- To Create a Area Chart with Data Animation
    - Chart Animation Using Data Import
    - Chart Animation Using Control Chart

Area Chart Properties

- **Shared Mem.:** Changes between **Scene**, **Global** and **Distributed** Shared Memory. Use **Inactive** memory to not forward any values via Shared Memory.
- **Specify X Values:** Enables DataX input.
    - **Key DataX:** Defines Shared Memory key name for X values.
    - **DataX:** Defines parameter for X values if Shared Memory is set to **Inactive.**
- **Key DataY:** Defines Shared Memory key name for Y values.
- **DataY:** Defines parameter for Y values if Shared Memory is set to Inactive.
- **Transfer Mode:** Sets string or array based data transfer (not available when set shared memory is set to **Inactive**).
- **Data Delim.:** Defines the value separator sign(s).
- **Number Format:** Defines if Viz should get the decimal point separator from the `System Locale` or `Custom` settings. If Custom is enabled:
    - **Decimal Symbol:** Defines which symbol is used as decimal point when **Custom Number Format** is set.

⚠ **Note:** If the Area Stack plug-in is used, the following Area Chart properties need to be defined in the Area Stack plug-in.

- **DataX, Y Fit:** Enables data normalization.
    - **DataX, Y Scale:** Scales input by the selected factor.
    - **DataX, Y Offset:** Adds an offset to the incoming data.
    - **DataX, Y Auto Scale:** Enables automatic data normalization.

Copyright © 2026 Vizrt

    - **DataX, Y Fit Size: Total:** Scales the whole chart to the defined borders. **Current** scales the current chart segment to the set borders.
    - **DataX, Y Detect Limits:** Detects minimum and maximum of all values and scales them to adjusted Start and Stop. This option is used to upscale the interesting part of the chart - especially if there are only little changes between the data values.
    - **DataX, Y Threshold:** Adds a definable offset to the detected limit.
    - **DataX, Y Start:** Lower Auto Scale edge.
    - **DataX, Y Stop:** Upper Auto Scale edge.
- **Values Animation:** Sets the vertical animation type to be created when data values change, but the number of nodes is still the same.
    - **Disable:** No animation.
    - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.** Also select an **Animation Mode.**
    - **Director:** Creates data transition animation with the Stage Director, that controls the Animation Progress parameter from 0.0% (old data) to 100.0% (new data). Also select an **Animation Mode.**
- **Animation Mode:**
    - **Concurrent:** Changes all nodes concurrently.
    - **Left to Right:** Transitions from left to right.
    - **Right to Left:** Transitions from right to left.
- **Balance Speed:** Relates node values to the position of transformed data when set to `On` and Animation Mode is set to **Left to Right** or **Right to Left**.

Example:

- **Balance Speed:** If the values of four nodes change from `0,0,0,0` to `1,9,90,900` in **Left to Right** mode at 50% of the transformation.
- **Balance Speed Off:** Only the first two nodes are transformed to new values.
- **Balance Speed On:** The first three nodes are already transformed to new values, and the last node also transformed to 44% of the new values, because the data changes from 0 (0+0+0+0) to 1000 (1+9+90+900), at 50% of transformation (in this case the values is changed for 500 of 1000), the first three nodes are already finished and the last node is changed to 400.

- **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.
- **Data Nodes Animation:** Sets the horizontal animation type to be created when number of data nodes change is detected.
    - **Disable:** No animation.
    - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**
    - **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.

Note: When the number of nodes is changes, area chart looks for added and removed nodes from their values, adds new nodes to the chart from zero width and resizes all nodes to their new size, specified in duration.

- **Horizontal Alignment:** Sets horizontal orientation to left, center or right.
- **Type:** Creates different graph looks. Available options are Normal, Spline, Staircase and Constrained Spline.

Copyright © 2026 Vizrt

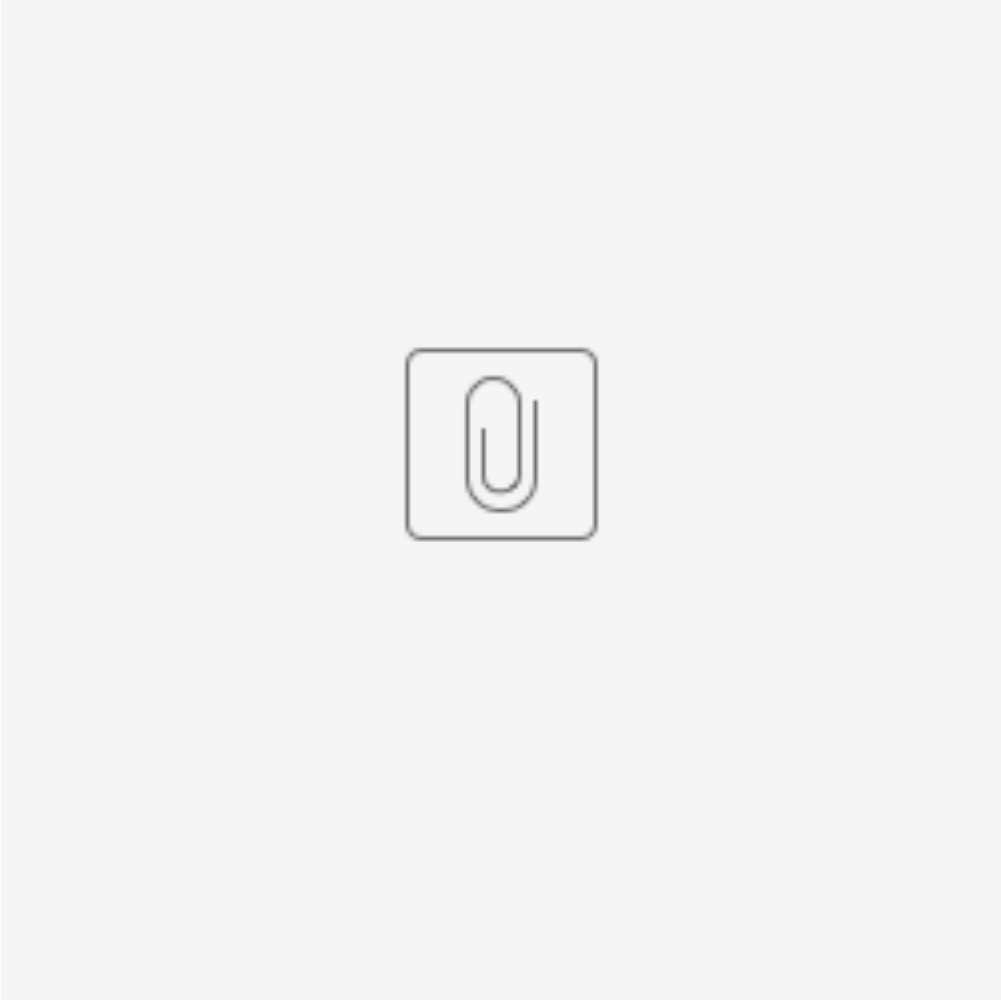
    - **Normal:** Enables direct connections between entered values.
    - **Spline:** Interpolates extra values to chamfer the graph.
    - **Staircase:** Creates a staircase look.
    - **Constrained Spline:** Same as Spline mode but with a different algorithm to prevent overshooting.
- **Stair Width:** Defines width for a single stair or for all stairs if Specify X Values is activated.
- **Chart Width:** Adjusts the chart width.
- **Chart Depth:** Adjusts extrusion.
    - **Tesselation Width:** Sets tessellation in Spline mode.
- **Selection:** Specifies Start and Stop type.
    - **Relative:** 0%: 100%
    - **Absolute:** Depends on the specified data.
    - **ID:** Value ID starting from 0.
- **Start%, abs,:** Graph starting point.
- **Stop%, abs,:** Graph stopping point.
- **Reposition Start:** Translates the whole chart always to the same starting point independent of varying Start and Stop positions.
- **Bevel:** Activates Bevel mode.
    - **Bevel Bottom:** Chamfers the chart's bottom.
    - **Bevel Size:** Adjusts bevel's size.
    - **Bevel Steps:** Sets roundness via the number of segmentation steps.

> ⚠ **Note:** A chamfer is a beveled edge connecting two surfaces.

- **RelativeMapping:** Stretches texture to fit graph width.
- **Pos. Container:** Translates every child container to a bar's top.
    - **Container Offset:** Adds a certain offset to each container.
    - **Container Offset Z:** Adds a Z Axis offset to each container.
    - **Center Container:** Centers each translated container.
- **Pointers Shared Mem.:** Selects the map where the pointer values should be distributed to.
- **Key Data Pointers:** Defines Shared Memory Keyname.
- **Pointer:** Activates the Pointer.
    - **Container:** Defines Container which should be used as pointer.
    - **Selection:** Selects the navigation type.
    - **Position%, abs,:** Sets pointer position in dependency of the selected navigation type.
    - **Offset:** Defines pointer offset from the chart.
    - **Normal Offset:** Rotates offset to the direction of the current surface normal.
    - **Rotate:** Rotates the pointer to the direction of the current surface normal.
    - **Offset Z:** Adds an additional Z offset to the pointer container.
    - **Center Container:** Centers Container - useful for varying pointer sizes.
- **Pointers:** Activates each pointer as required:
    - **Low Pointer** / **High Pointer / Start Pointer / Stop Pointer**
    - **Container:** Selects a container for the pointer.

Copyright © 2026 Vizrt

To Create a Scene with Area Chart



1. Create a new Container.
2. Add the Area Chart plug-in.
3. Open the Transformation Editor.
   Set these parameters:
   - Position X: `-100.0`.
   - Position Y: `-100.0`.
   - Rotation Y: `-25.0`.
4. Add a Material and/or a Texture.
5. Open the Area Chart editor.
6. Enter these parameters:
   - Set DataY values: `10,20,30,20,50,60,80,45,20`.
   - Set **Type** to `Spline`.
   - Set **Chart Width** to `200.0`.

Copyright © 2026 Vizrt

- Set **Chart Depth** to `30.0`.

To Create a Scene with Area Chart and Data Storage

1. Create a new Container.
2. Add the Area Chart plug-in.
3. Add Data Storage to the Area Chart Container.
4. Select the Area Chart plug-in.
    a. Set **SharedMem.** to `Scene`.
    b. Rename the **Key DataY** to `MyDataY`.
5. Open the Data Storage editor.
    a. Enter `MyDataY` in the **Key Data1** field.
    b. Set the **SharedMem.** to `Scene`.
    c. In **Data1** enter: `10,40,50,20,80,90,60,50`.

To Create a Area Chart with Data Animation

Chart Animation Using Data Import

The following steps are to prepare data sets for animation in Excel file.

1. Start Microsoft Excel.
2. Enter **ExcelData1** into cell A1.
3. Add some sample values in the cells below (`A2-A8: 80, 35, 45, 75, 85, 55, 60`).
4. Enter **ExcelData2** into cell A2.
5. Add some sample values in the cells below (`B2-B9: 80, 35, 45, 75, 40, 85, 55, 60`).
6. Enter **ExcelData3** into cell A3.
7. Add some sample values in the cells below (`C2-C9: 40, 60, 75, 85, 80, 55, 45, 35`).
8. Enter **ExcelData4** into cell A4.
9. Add some sample values in the cells below (`D2-D8: 60, 75, 85, 80, 55, 45, 35`).
10. Rename this first sheet to **MyTable** (can be done with a double click on the sheet name at the bottom).
11. Follow one of the following methods to make Viz Engine can read this Excel data file.
    a. **Save and close** the Excel document.
    b. **Share** the Excel document.
        i. On the Review tab, in the Changes group, click the **Share Workbook** button.
        ii. The Share Workbook dialog box appears, and you select the **Allow changes by more than one user at the same time.** check box on the Editing tab.
        iii. Click **OK**.
        iv. Save the Excel document.

IMPORTANT! You must use the same platform (x64/x86) of Microsoft Excel and Viz Engine.

After the data sets are created, the following steps are to create a Data Animation scene.

1. Follow the "To create a Area Chart" instruction.
2. Modify the following parameters of the Area Chart plug-in.
    - Change **Shared Mem.** to **Scene.**
    - Set **Kay DataY** to **DataY.**

Copyright © 2026 Vizrt

- Set **Data Delim.** to **#.**
- Set **Values Animation** to **Timer.**
- Set **Data Nodes Animation** to **Timer.**

3. Add Data Import to the same container.
    a. Set **File** to above prepared Excel file.
    b. Set **Table / Sheet** to **MyTable**.
    c. Set **Column(,Col...)** to **ExcelData1**.
    d. Set **Data Delim.** to **#.**
    e. Set **Shared Mem.** to **Scene.**
    f. Set **Key** to **DataY.**
4. Click the **GetIt** button and animation chart shows from zero to seven nodes.
5. Add one data node in the middle of chart.
    a. Set **Column(,Col...)** to **ExcelData2.**
    b. Click the **GetIt** button.
6. Change values of eight data nodes.
    a. Set **Column(,Col...)** to **ExcelData3.**
    b. Click the **GetIt** button.
7. Remove the first data node.
    a. Set **Column(,Col...)** to **ExcelData4.**
    b. Click the **GetIt** button.

> **Note:** You can add more data columns and play with animation on data change.

Chart Animation Using Control Chart

1. Follow the "To create a Area Chart" instruction.
2. Modify the following parameters of the Area Chart plug-in.
    - Set **Values Animation** to **Timer.**
    - Set **Data Nodes Animation** to **Timer.**
3. Add Control Chart to the same container and save the scene.
4. Open Viz Trio.
    - Import the scene.
    - Click "1 (AreaChart)" in Tab Fields.
    - Edit values in Editing Template.

See Also

- Control Chart
- Data Fit
- Data Import
- Data Label
- Data Storage
- Area Stack

Copyright © 2026 Vizrt

Bar Chart



The Bar Chart plug-in draws a bar chart, filled with data out of a Shared Memory Map.

You can use delimited strings or arrays for data transfer via Scene-, Global- or Distributed-Shared Memory.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

This page contains the following topics and procedures:

- Bar Chart Properties
- To Create a Bar Chart
- To Create a Bar Chart with Data Animation
  - Chart Animation Using Data Import
  - Chart Animation Using Control Chart

Bar Chart Properties

- **Shared Mem.:** Changes between **Scene**, **Global** and **Distributed** Shared Memory. Use **Inactive** memory to not forward any values via Shared Memory and the data is taken from the **Data** field.
- **Data:** Inputs chart data (available when **Inactive** is selected).
- **Key Data:** Determines shared Memory key name (available when **Scene**, **Global** or **Distributed** is selected).
- **Transfer Mode:** Sets **String** or **Array** based data transfer.
- **Data Delim.:** Defines the value separator sign(s).
- **Number Format:** Defines if Viz should get the decimal point separator from the `System Locale` or `Custom` settings. If Custom is enabled:
  - **Decimal Symbol:** Defines which symbol is used as decimal point when **Custom Number Format** is set.

> ⚠ **Note:** If the Bar Stack plug-in is used, the following Bar Chart properties need to be defined in the Bar Stack plug-in.

- **DataX Fit:** Enables data normalization on the horizontal:
  - **DataX Scale:** Scales input by the selected factor.
  - **DataX Offset:** Adds an offset to the incoming data.
  - **DataX Auto Scale:** Enables automatic data normalization.
  - **DataX Fit Size:**
    - **Total:** Scales complete chart as per the defined Start and Stop settings.
    - **Current:** Scales the current chart segment to the set borders.
- **Data Fit:** Enables data normalization:
  - **Data Scale:** Scales input by the selected factor.

Copyright © 2026 Vizrt

    - **Data Offset:** Adds an offset to the incoming data.
    - **Data Origin Offset:** Adds an offset to the smallest/minimum value.
    - **Data Auto Scale:** Enables automatic data normalization.
    - **Data Fit Size:** Scales complete chart in dependency of the defined Start and Stop settings (**Total**) or scales the **Current** chart segment to the set borders.
    - **Data Detect Limits:** Detects minimum and maximum of all values and scales them to adjusted Start and Stop.
    - **Data Threshold:** Adds a definable offset to the detected limit.
    - **Data Start:** Lower Auto Scale edge.
    - **Data Stop:** Upper Auto Scale edge.
- **Compare:** Specifies another BarChart for comparison. You have to set this parameter also in the other chart(s) so that each chart can react on changes of the other(s).
- **Value Animation:** Sets the vertical animation type to be created when data change is detected:
    - **Disable:** No animation**.**
    - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**. Also select an **Animation Mode**.
    - **Director:** Creates data transition animation with the Stage Director, that controls the Animation Progress parameter from 0.0% (old data) to 100.0% (new data). Also select an **Animation Mode**.
- **Animation Mode:**
    - **Concurrent:** Changes all nodes concurrently.
    - **Left to Right:** Transitions from left to right.
    - **Right to Left:** Transitions from right to left.
- **Balance Speed:** Relates node values to the position of transformed data when set to `On` and Animation Mode is set to **Left to Right** or **Right to Left**.

Example:

- **Balance Speed:** If the values of four nodes change from 0,0,0,0 to 1,9,90,900 in **Left to Right** mode at 50% of the transformation.
- **Balance Speed Off:** Only the first two nodes are transformed to new values.
- **Balance Speed On:** The first three nodes are already transformed to new values, and the last node also transformed to 44% of the new values, because the data change from 0 (0+0+0+0) to 1000 (1+9+90+900), at 50% of transformation (in this case the values is changed for 500 of 1000), the first three nodes are already finished and the last node is changed to 400.

- **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.
- **Data Nodes Animation:** Sets the horizontal animation type to be created when number of data nodes change is detected.
    - **Disable:** No animation.
    - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**
    - **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.

Copyright © 2026 Vizrt

⚠ **Note:** When the number of nodes is changed, bar chart looks for added and removed nodes from their values, adds new nodes to the chart from zero width and resizes all nodes to their new size, specified in duration.

- **Horizontal alignment:** Sets horizontal orientation to left, center or right.
- **Corners:** Sets the bar's corners.
- **Chart Width:** Defines width for a single bar or for all bars.
- **Bar Width:** Value of the previous parameter.
- **Max Width:** Allows bars to grow only to a certain width.
- **Bar Gap Ratio:** Adjusts the gap between the single bars. 0% means no gap.
- **Depth:** Defines if the bar depth should equal its width or lets you set a custom depth.
- **Bar Depth:** Adjusts bar depth in custom depth mode.
- **Draw Zero:** If the Bar value is equivalent to `0`, then a very flat bar is drawn instead of nothing.
- **Tessellation Length:** Tessellates bar length.
- **Progress Mode:**
  - **Bars:** Allows each bar grows to its value.
  - **Chart X:** The chart grows horizontally. Each bar comes in at its final height.
  - **Chart** Y: The chart grows vertically. All bars grow together to their final height.
- **Relative Length:** If activated, each bar has its own 100% length (for example, seven bars equal 700%).
- **Const. Speed:** Sets the same animation duration for each bar.
- **Start%:** Determines starting point of the graph.
- **Stop%:** Determines stop point of the graph.
- **Bevel:** Activates Bevel mode.
  - **Bevel Top:** Chamfers bar's top.
  - **Bevel Bottom:** Chamfers bar's bottom.
  - **Bevel Size%:** Adjusts bevel's size from 0 to 100.
  - **Bevel Steps:** Sets roundness via the number of segmentation steps.

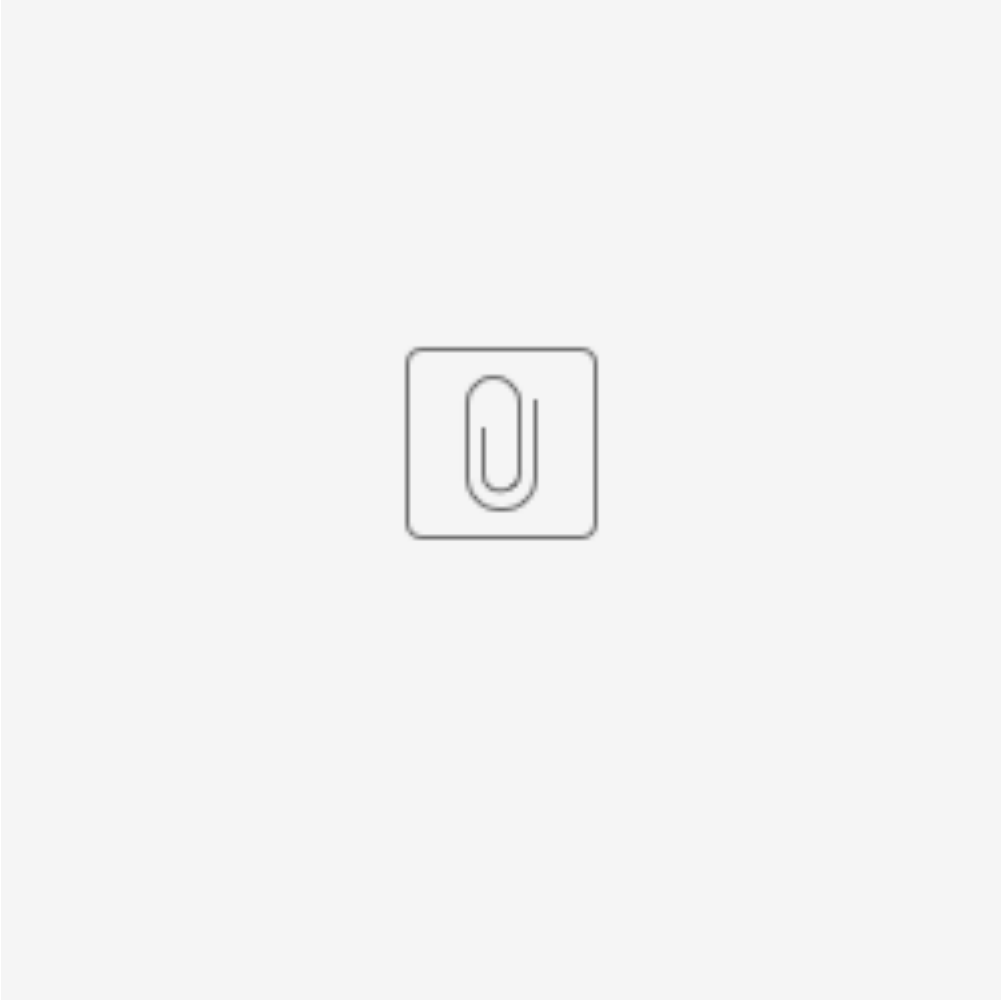
⚠ **Note:** A chamfer is a beveled edge connecting two surfaces.

- **Pos. Container:** Translates every child container to a bar's top.
  - **Container Offset:** Adds a certain offset to each container.
  - **Container Offset X:** Adds a X Axis offset to each container.
  - **Container Offset Y:** Adds a Y Axis offset to each container.
  - **Container Offset Z:** Adds a Z Axis offset to each container.
  - **Center Container:** Centers each translated container.
  - **Size Compensation:** Repositions containers if they can change their size (for example, Text geometries used as chart labels). The labels vary in their width which requires a repositioning if a horizontal BarChart (rotated 90 deg) is used.
  - **None:** Deactivates container size compensation.
  - **Width:** Compensates width changes.
  - **Height:** Compensates height changes.
- **Map:** Calculates texture coordinates for all bars, a single bar or defines vertex colors instead. Colors can be set individually for each bar or value dependent (+/- color). In the Color +/- mode you can override value dependent colors.

Copyright © 2026 Vizrt

- **Color ID:** Moves between the available vertex colors.
- **Color:** Chooses color for the current ID.
- **Positive:** Chooses color for positive values.
- **Negative:** Chooses color for negative values.
- **Color Override:** Activates or deactivates overriding of a positive or negative color with a defined alternative color.

To Create a Bar Chart



1. Create a new group and add the Bar Chart plug-in to it.
2. Open the Transformation Editor and set Position X and Y to `-100.0` and Rotation Y to `-25.0`.
3. Add a material and/or a texture to it.
4. Open the Bar Chart editor and set the following parameters:
    - Set the following DataY values: `10,20,30,20,50,60,80,45,20`.
    - Set Corners to `3`.
    - Set Bar Width to `20.0`.

Copyright © 2026 Vizrt

- Set Bar Gap Ratio[%] to `10.0`.

To Create a Bar Chart with Data Animation

Chart Animation Using Data Import

The following steps are to prepare data sets for animation in Excel file.

1. Start Microsoft Excel.
2. Enter **ExcelData1** into cell A1.
3. Add some sample values in the cells below (`A2-A8: 80, 35, 45, 75, 85, 55, 60`).
4. Enter **ExcelData2** into cell A2.
5. Add some sample values in the cells below (`B2-B9: 80, 35, 45, 75, 40, 85, 55, 60`).
6. Enter **ExcelData3** into cell A3.
7. Add some sample values in the cells below (`C2-C9: 40, 60, 75, 85, 80, 55, 45, 35`).
8. Enter **ExcelData4** into cell A4.
9. Add some sample values in the cells below (`D2-D8: 60, 75, 85, 80, 55, 45, 35`).
10. Rename this first sheet to **MyTable** (can be done with a double click on the sheet name at the bottom).
11. Follow one of the following methods to make Viz Engine can read this Excel data file.
    a. **Save and close** the Excel document.
    b. **Share** the Excel document.
        i. On the Review tab, in the Changes group, click the **Share Workbook** button.
        ii. The Share Workbook dialog box appears, and you select the **Allow changes by more than one user at the same time.** check box on the Editing tab.
        iii. Click **OK**.
        iv. Save the Excel document.

IMPORTANT! You must use the same platform (x64/x86) of Microsoft Excel and Viz Engine.

After the data sets are created, the following steps are to create a Data Animation scene.

1. Follow the "To create a Bar Chart" instruction.
2. Modify the following parameters of the Bar Chart plug-in.
    - Change **Shared Mem.** to **Scene.**
    - Set **Kay Data** to `Data`.
    - Set **Data Delim.** to `#`.
    - Set **Values Animation** to `Timer`.
    - Set **Data Nodes Animation** to `Timer`.
3. Add Data Import to the same container.
    a. Set **File** to above prepared Excel file.
    b. Set **Table / Sheet** to `MyTable`.
    c. Set **Column(,Col...)** to `ExcelData1`.
    d. Set **Data Delim.** to `#`.
    e. Set **Shared Mem.** to `Scene`.
    f. Set **Key** to `Data`.
4. Click the **GetIt** button and animation chart shows from zero to seven nodes.
5. Add one data node in the middle of chart (Horizontal Animation).

Copyright © 2026 Vizrt

    a. Set **Column(,Col...)** to **ExcelData2.**
    b. Click the **GetIt** button.
6. Change values of eight data nodes (Vertical Animation).
    a. Set **Column(,Col...)** to **ExcelData3.**
    b. Click the **GetIt** button.
7. Remove the first data node (Horizontal Animation).
    a. Set **Column(,Col...)** to **ExcelData4.**
    b. Click the **GetIt** button.

> **Note:** You can add more data columns and play with animation on difference data change.

Chart Animation Using Control Chart

1. Follow the "To create a Bar Chart" instruction.
2. Modify the following parameters of the Bar Chart plug-in.
    - Set **Values Animation** to **Timer.**
    - Set **Data Nodes Animation** to **Timer.**
3. Add Control Chart to the same container and save the scene.
4. Open Viz Trio.
    - Import the scene.
    - Click "1 (BarChart)" in Tab Fields.
    - Edit values in Editing Template.

See Also

- Control Chart
- Data Fit
- Data Import
- Data Label
- Data Storage
- Basic Container Presenter
- Bar Stack
- Advanced Bar Chart Creation

Copyright © 2026 Vizrt

Advanced Bar Chart Creation

This section outlines some of the more advanced features of the Bar Chart plug-in. The example provided is aimed at designers familiar with the Viz Artist user interface and prior knowledge of advanced scene design techniques, such as the use of Control plug-ins and scripting.

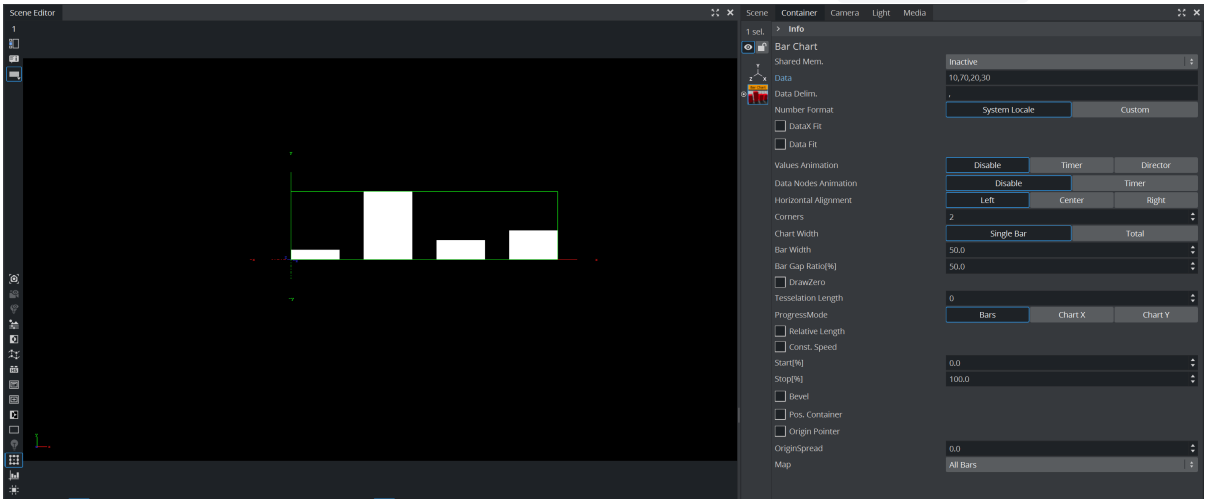
This brief tutorial consists of the following sections and procedures:

- Establishing the Basic Bar Chart
- Adding a Script to Control Data
- Calculating Scale Manually for External Applications or for Scripts
- Adding Labels to the Bars
- Animating the Bars
- Animating the Labels
- Adding Color

Establishing the Basic Bar Chart

In this first section, the basic bar chart is established and set up to be controlled by an External Control Application, such as Viz Trio or Viz Pilot. In an environment where the data is modified directly through the plug-in, there is no need for the shared memory or script as outlined below.

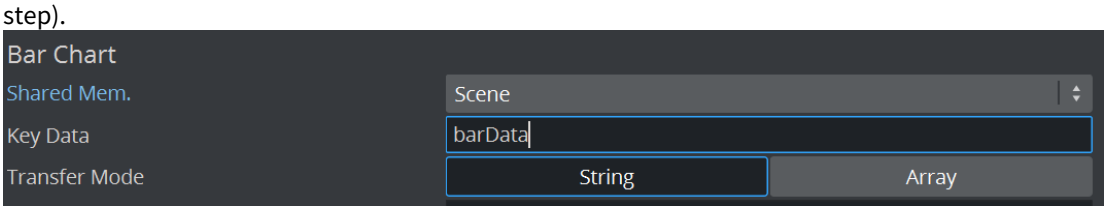
1. Create a new scene and add an empty container. This container is used as the parent container. Rename the container to `object`.
2. Add Bar Chart to a sub-container in the scene tree and name it `BarChart`, then open Bar Chart properties.
3. In the **Data** field, add a series of comma-separated values. Here, the values `10,70,20,30` are used.



> **Note:** By using the **Data Delim.** field, the data delimiter can be changed from comma to any other delimiter, for example the pipe character (`|`), to suit the needs of the production environment the scene is used in.

4. Currently, the **Shared Mem.** parameter is set to `Inactive`. This means that the data used by the plug-in is being read from the **Data** field, as set in step 2. However, in most cases some sort of shared memory would

be used in order to share the same data with other plug-ins (in particular, Data Label introduced in a later step).



In this example, the **Scene** shared memory is used. This Shared Memory setting does not interfere with any other scene(s) or machine, only the local scene. By clicking the `Scene` button, the field **Key Data** becomes available, where the name of the shared memory used for the bar data must be specified. An external control application, such as Viz Pilot or Viz Trio, can now be used to set the values that are to be represented by the bar chart. Enter `barData` as the shared memory name.

5. Add an empty container to the scene tree, and then the Control Parameter and Script plug-ins to the newly created container. Rename the container to `ValuesControl` and open the Script plug-in properties.

Adding a Script to Control Data

1. Next, a script allowing the data to be controlled with Control Parameter needs to be added:

```
Dim oldKey As String = ""
Sub OnInitParameters()
RegisterParameterString("barsValues", "Bar Values", "1,2,3,4,5,6", 40, 1000,
"")
RegisterParameterString("barsSMName", "Shared Mem Name", "barData", 40, 1000,
"")
End Sub

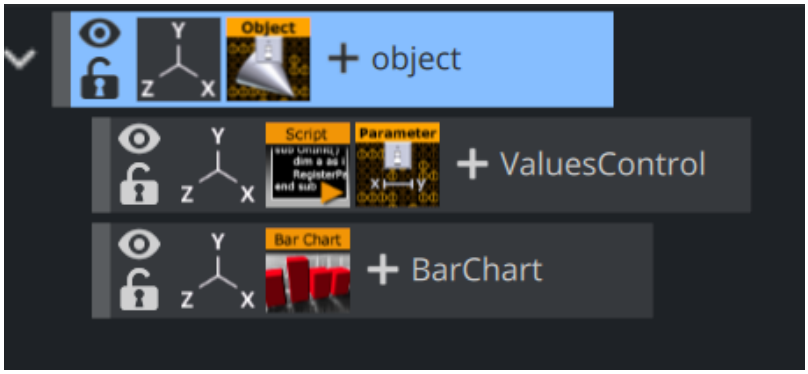
Sub OnParameterChanged(parameterName As String)
If parameterName = "barsSMName" Then
If oldKey <> "" Then Scene.Map.DeleteKey(oldKey)
Scene.Map.CreateKey(GetParameterString("barsSMName"))
oldKey = GetParameterString("barsSMName")
Else
Scene.Map[GetParameterString("barsSMName")] = GetParameterString("barsValues")
End If
End Sub

Sub OnInit()
Scene.Map.CreateKey(GetParameterString("barsSMName"))
Scene.Map[GetParameterString("barsSMName")] = GetParameterString("barsValues")
End Sub
```

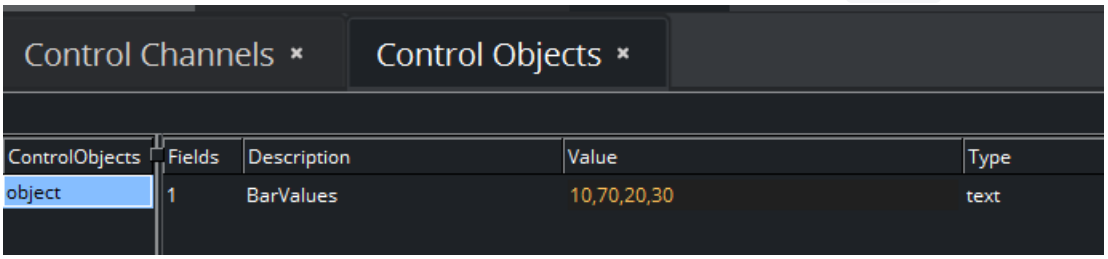
Tip: As alternative to a string with a data delimiter, the data source could also be a *Double-Array*. This can be achieved by selecting `Array` as **Transfer Mode** in the Data Bar plug-in properties. In that case, instead of assigning a string, an `Array[Double]` data type would have been assigned in the example script above. This can be convenient if the data is provided by an external source, for instance a database or as the result of parsing a text file.

Copyright © 2026 Vizrt

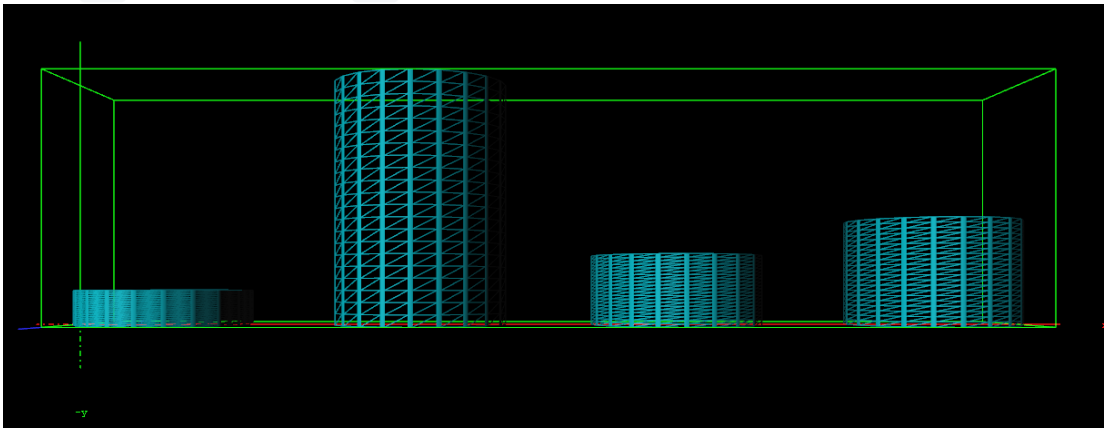
After entering the above script into the script editor, click the **Compile & Run** button. The scene tree should now look like this:



2. Open the Control Parameter properties. Change the **Field Identifier** to `1001`, set the **Data Type** to `Text`, and enter `SCRIPT*INSTANCE*barsValues` in the **Parameter** field. By opening the Control Objects tab in the Control area, the *Bar Values* can now be changed directly. Any changes here should be reflected immediately on the rendered output in the **Preview** window when pressing `Enter`.



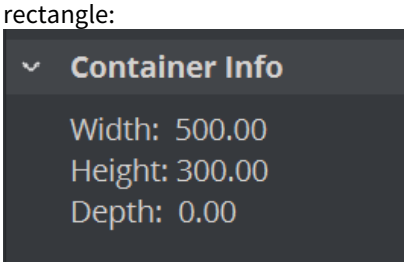
3. Apply a material on the Bar Chart container. The shape of the rendered bars can be controlled by adjusting the **Corners**, **Tesselation Length**, and **Bevel** parameters in Bar Chart properties. For a classic 2-dimensional bar, use a **Corners** value of `2`. For a smooth cylinder shape, the corners value can be increased up to a maximum of `1000`. Activate *Wireframe* rendering mode in the preview window by clicking the `W` button in the scene editor to get a better understanding of these parameters. This is how it may look with Corners set to `25`, a **Tesselation Length** of `20`, with **Bevel**, **Bevel Top** and **Bevel Bottom** set to `On`, **Bevel size (%)** to `15` and **Bevel Steps** to `10`:



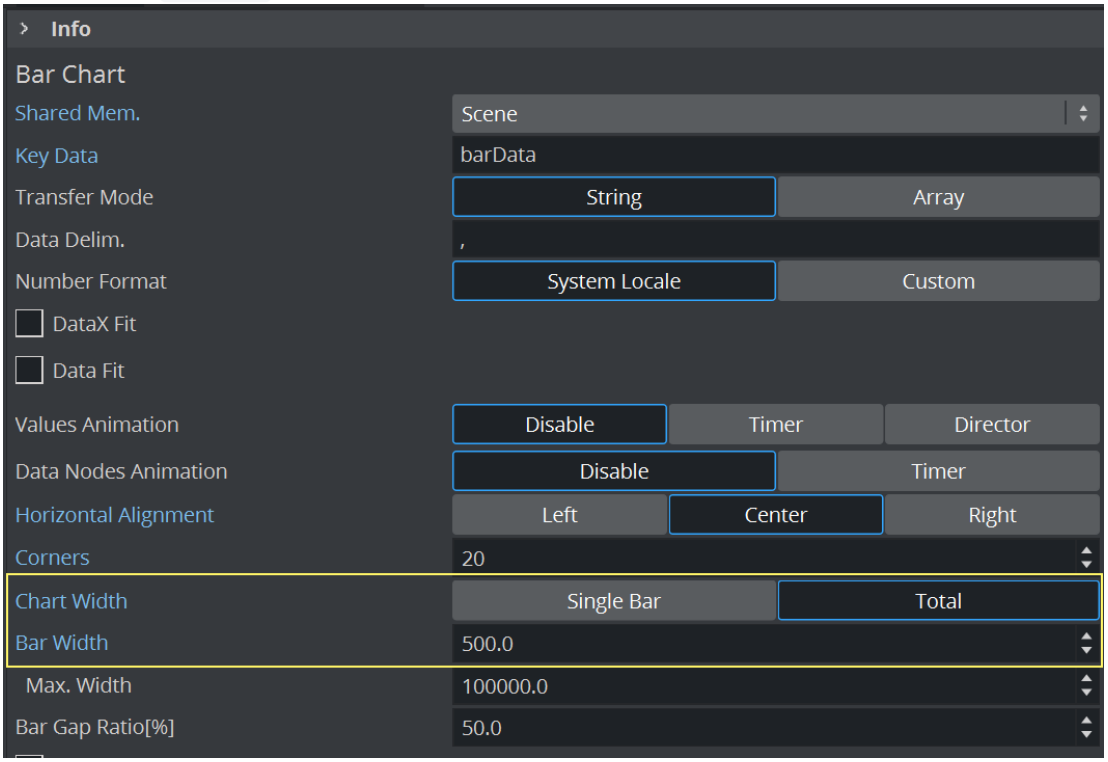
Copyright © 2026 Vizrt

4. Drag a Rectangle to the scene tree to create a new container, and add Expert. In the Expert properties, select `Outline` as **Drawmode** to create a bounding box. Often, it is useful to fit the entire data into a given screen bounding box. This can be achieved positioning the newly added rectangle on the screen, representing the area where the graph should be drawn. Here, the rectangle has width of `500` and a height of `300`. It is positioned at `0.0` in Z-space, as is also the container holding the Bar Chart itself.

   Open the Rectangle Transformation Editor and check the effective width and height of the boundary rectangle:



5. To match the width of the graph to the bounding box, open Bar Chart properties and change the **Chart Width** to `Total` and the **Bar Width** to that of the bounding rectangle (`500.0`). Then set **Horizontal Alignment** to `Center` .

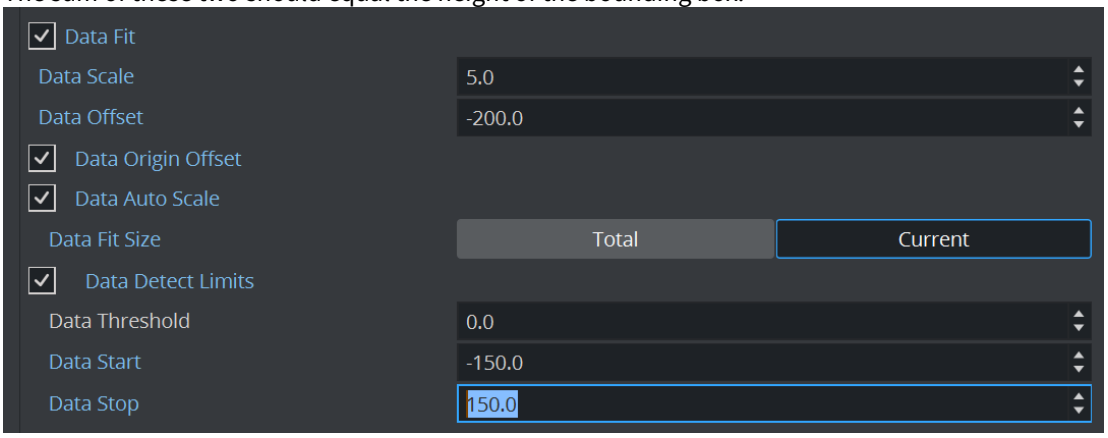


Tip: The **Max. Width** parameter defines how thick a bar can become. If the number of input values are reduced, the bars width increases within the defined **Bar Width** area. If there are only two bars left, they look really big. The Max. Width parameter is useful to limit the impact of this behavior.

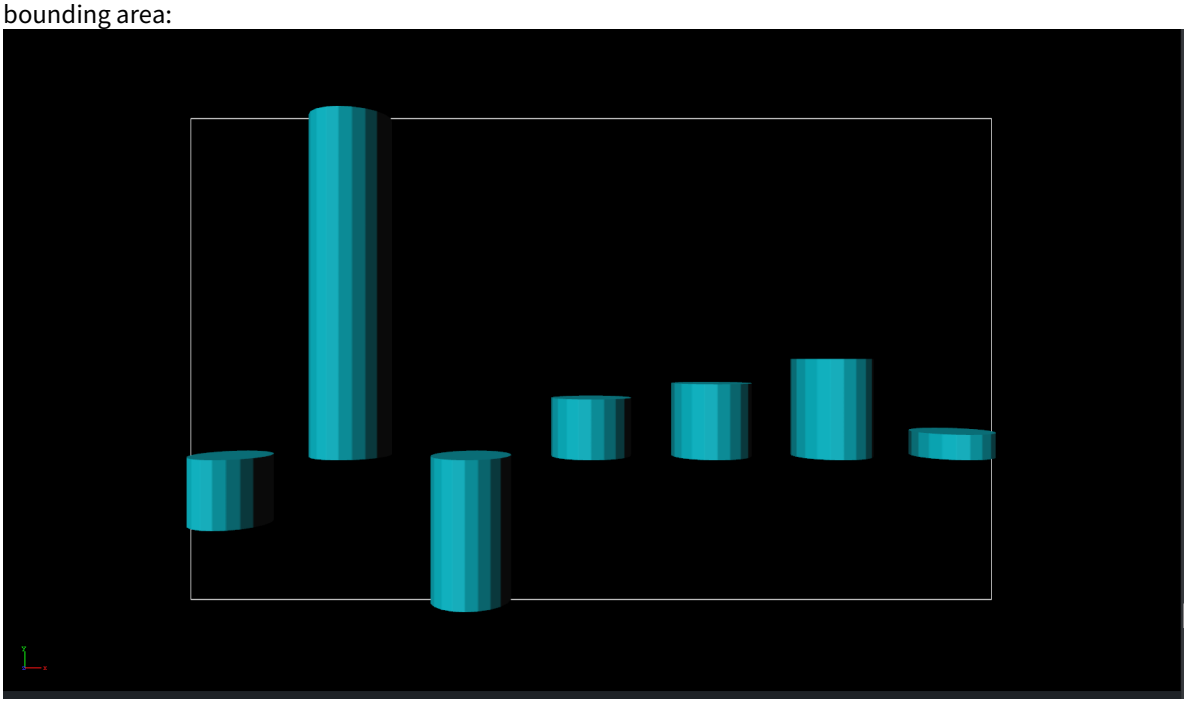
6. Mapping the graph to fit the whole height of the bounding area is done by enabling some of the properties inherent in Bar Chart. In the property panel, set **Data Fit** and **Data Auto Scale** to `On` , then set **Data Fit**

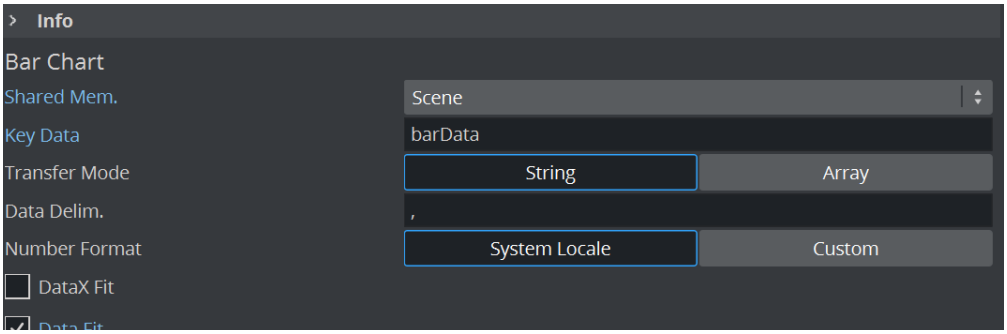
Copyright © 2026 Vizrt

Size to `Current`. Then either type in, or click and drag, the **Data Start** and **Data Stop** values as required. The sum of these two should equal the height of the bounding box.

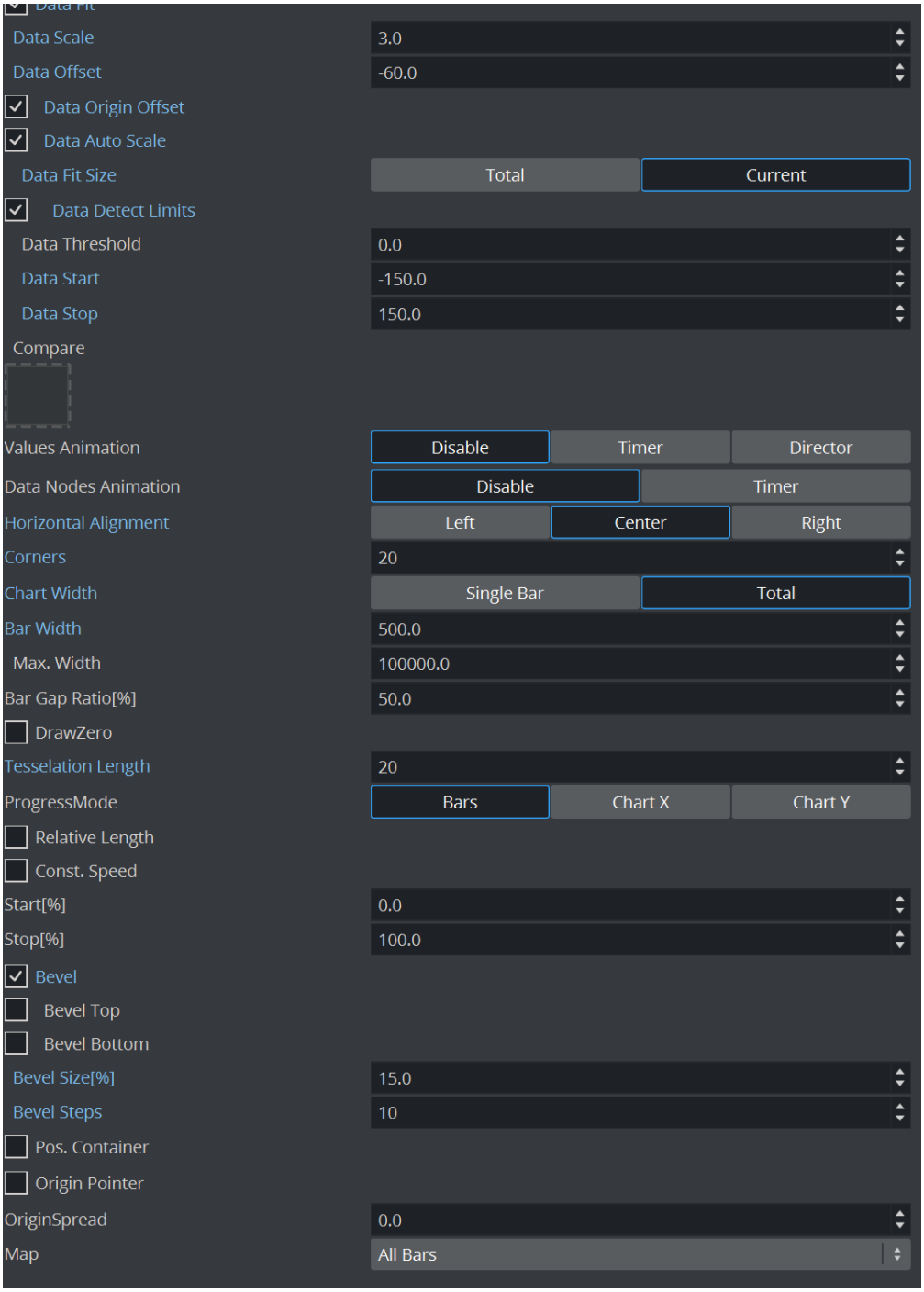


Open the **Control Object** tab in the **Object area**, and enter the values used for the example data scale calculation above, `-14, 70, -30, 12, 15, 20, 5,` as values for the **Bar Values** object. The preview is automatically updated. The result should look like this, with the Bars perfectly matching the bounding area:





Copyright © 2026 Vizrt



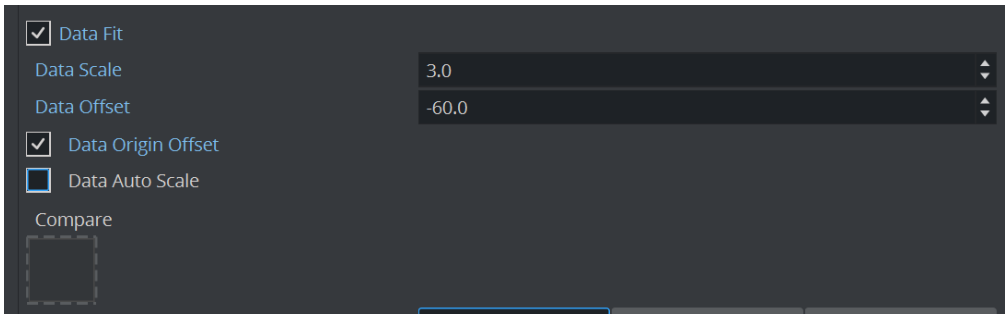
Please go straight to Adding Labels to the Bars to continue with the tutorial. The following section is intended as reference for users wanting to perform this scaling manually, or by an external application or a script.

Calculating Scale Manually for External Applications or for Scripts

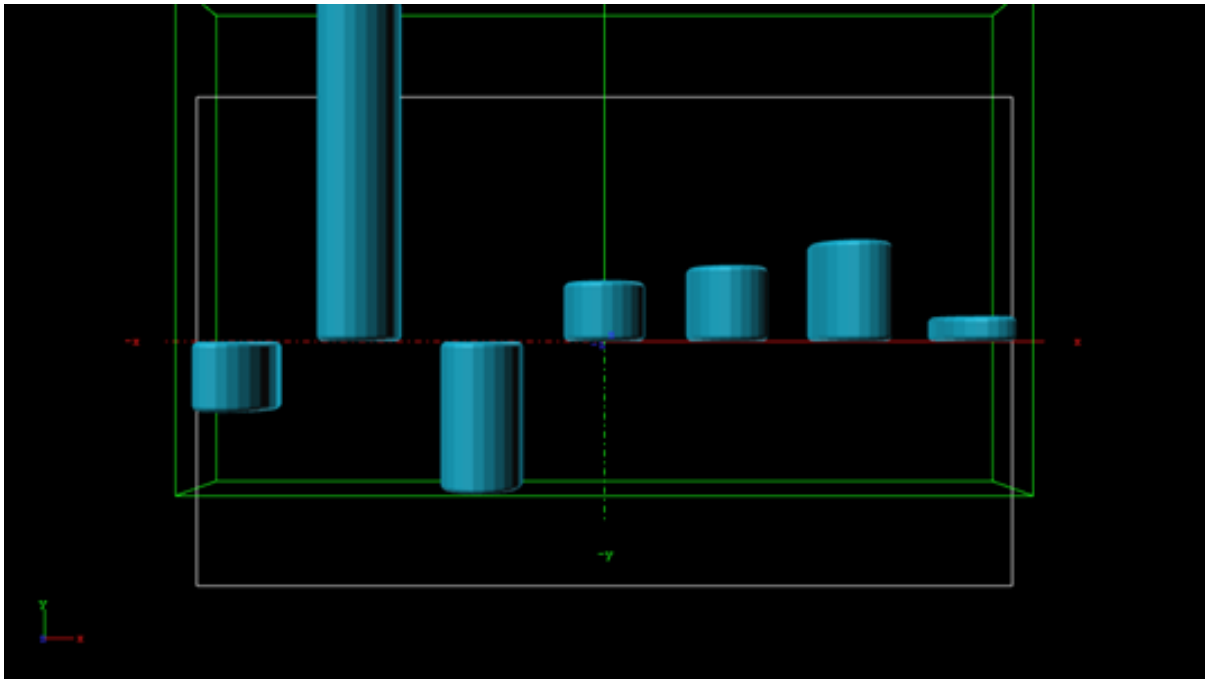
The minimum and maximum values of the data needs to be known to be able to calculate the required scale value. For example, if using the values `-14, 70, -30, 12, 15, 20, 5` as input data, the minimum value

is `-30` and the maximum value is `70`. If negative values are not used, set the minimum value to minimum to `0.0` (more mathematically speaking; `min=minimum(0.0, minimum(inputdata))`). For the example values above, this results in a range of **100** (from min to max). These 100 units must be matched to the *bounding box height*, which is **300**. Dividing `300` by `100` returns a scale value of `3.0`.

Set **Data Fit** to `On` and enter this value in the **Data Scale** field (1):



Open the **Control Object** tab in the **Object area**, and enter the values used for the example data scale calculation above, `-14, 70, -30, 12, 15, 20, 5,` as values for the **Bar Values** object. The preview is automatically updated. Notice how the graph aligns to the bounding box size, however, it is not centered in the middle.



To center the graph within the bounding box, a *data offset* translation value must be calculated. The formula for calculating this value is: $-\frac{height}{2} + min * dataScale$, where *dataScale* is the value calculated in the previous step, and *height* is the bounding height (here: `300`). Using the values in this example, the formula would be $\frac{-300}{2} + (-30) * 3.0 = -60$.

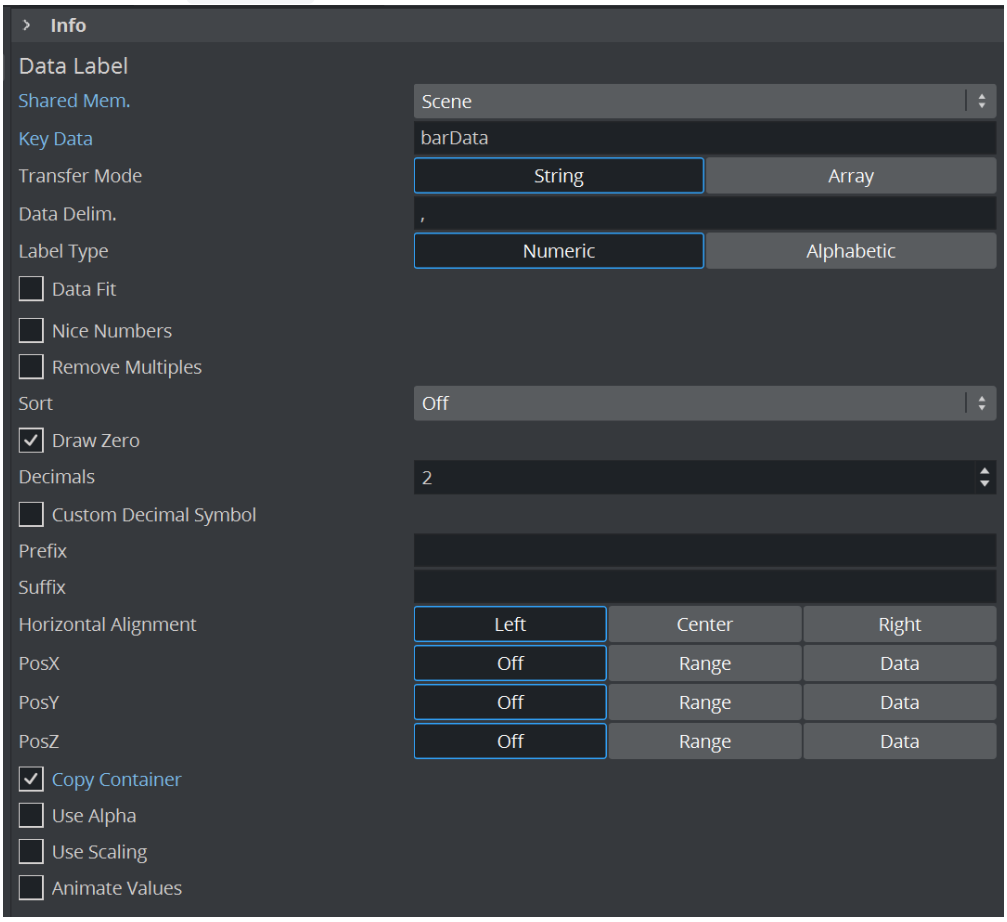
Copyright © 2026 Vizrt

Set **Data Origin Offset** to `On` and enter the resulting value, `-60`, in the **Data Offset** field. The result should look like in step 11, with the bars perfectly matching the bounding area.

Adding Labels to the Bars

To have labels representing the bar value above each bar, Data Label is used.

1. Add Data Label to the **BarChart** container and open Data Label properties. Select `Scene` as **Shared Mem.** and enter `barData` in the **Key Data** field, then set the **Copy Container** parameter to `On`.

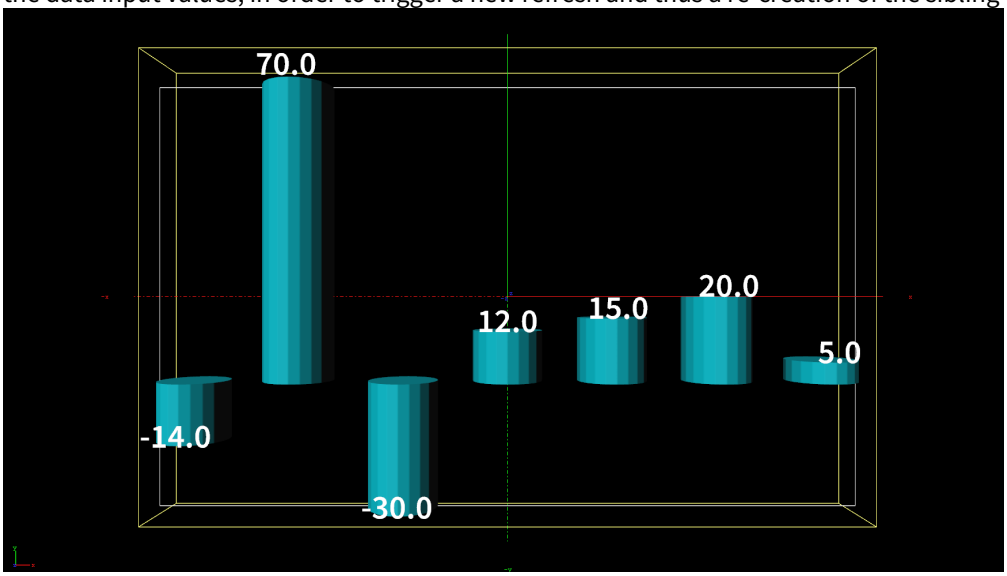


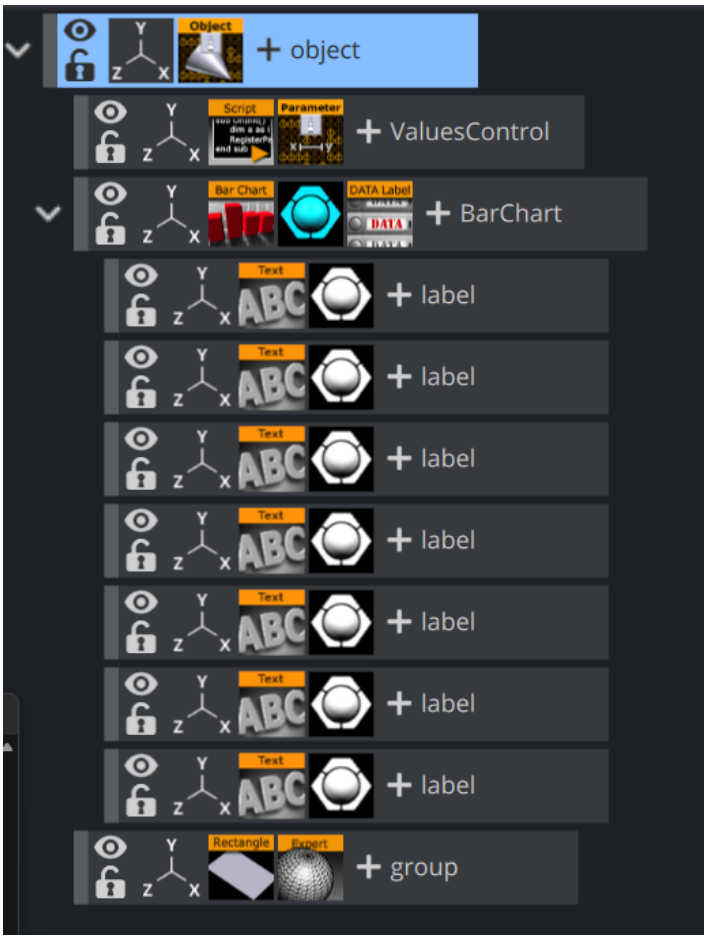
2. Create a new container as a child of the **BarChart** container, and add a font and a material to the new container. Rename the container to `label`. Set the size and text parameters as desired. Here, the font container's **Scaling** is `.25`, and the font orientations are set to **Orientation center** and **Vertical orientation first line**.
3. Open the Bar Chart properties and set **Pos. Container** to `On`. Combined with the **Copy Container** parameter set in step 13, the parameter Pos Container lets Viz copy the **label** container for each label value and position the sub-containers of the BarChart container in relation to the respective bar representing the value. Container offsets can also be defined if desired.

Tip: When creating 3D bars where the **Corner** value is set to more than `2`, the label needs to be offset in Z-space. A way to circumvent this is to add Expert, and set Z-Buffer Ignore to `On`.

Copyright © 2026 Vizrt

4. Open the **Control Area** and change one of the bar values in the **Control Object** tab, in order to trigger a refresh. This should create the new **label** containers and populate them with data. If modifications are required on the label prototype (the first child container), delete all other siblings and alter one or more of the data input values, in order to trigger a new refresh and thus a re-creation of the sibling containers.





Copyright © 2026 Vizrt

Animating the Bars

Basic animation of the bars is typically controlled by the Bar Chart **Stop%** parameter, with values from `0.0` to `100.0`. For a basic animation, set **Stop%** to `0` and add a key-frame at frame `0`, then set the **Stop%** value to `100` and add another key-frame.

There are various other parameters that further influence the animation:

- By activating the **Const. Speed** setting, animation of each bar has the same duration.
- The **Relative Length** setting gives each bar its own 100 percent length, meaning seven bars equal 700 percent.
- When **Progress Mode** is set to `Bars`, each bar subsequently grows to its value; and when set to `Chart X` or `Y`, the whole chart grows horizontally or vertically, respectively, with the incoming bars already at their final height.

> ⚠ **Note:** When **Progress Mode** is set to `Chart Y`, columns representing negative values are not visible.

Animating the Labels

The labels can be animated in a similar fashion, as Data Label has the same **Stop%**, **Relative Length** and **Const. Speed** parameters. Additionally, the values can be animated while the bars grow by setting **Animate Values** to `On`. By adding Alpha to the label containers, and activating the setting **Use Alpha**, the labels fade in as the bars appear.

The parameter **Use Scale** is used when an interpolation on the scale value of each label between `0.0` and `1.0` is required. As a result of the fact that the scaling interpolates to the fixed value of `1.0`, combined with the fact that the font label in most cases is somehow scaled, fonts often appear too big. This problem can be overcome by adding a parent container to the font. Data Label puts the text into the first child container containing a Text Geometry plug-in, so the label content is preserved.

Adding Color

Colors can be assigned to each individual bar, by setting the **Map** parameter of Bar Chart to **Color**. RGB and alpha values can be set by adjusting the **Color ID**. However, Viz Trio and Viz Pilot are not able to control the colors, there is no control plug-in capable to do so. This can be resolved by using a script and to expose the script parameters to the control application. The following script example allows to set a clear color for all bars, and it has a button that calculates the min/max bar and assigns different colors to those bars:

```
Sub calcMinMax()
Dim chart As Container = GetParameterContainer("bars")
If chart = null Then Exit Sub
Dim valueString As String = Scene.map[GetParameterString("barsSMName")]
Dim values as Array[String]
Dim minValue As Double = 1000000
Dim maxValue As Double = -100000
Dim minIndex As Integer = -1
Dim maxIndex As Integer = -1
Dim i As Integer = 0
valueString.split(",", values)
```

Copyright © 2026 Vizrt

```
For Each str In values
If str.length <> 0 Then
If CDbl(str) < minValue Then
minValue = CDbl(str)
minIndex = i
End If

If CDbl(str) > maxValue Then
maxValue = CDbl(str)
maxIndex = i
End If
End If
i = i + 1
Next
Dim col As Color
Dim clearColor As Color = GetParameterColor("clearColor")

For i = 0 To values.Size
If i = maxIndex Then
col.Red = 0.317
col.Green = 0.71
col.Blue = 0.0
ElseIf i = minIndex Then
col.Red = 1.0
col.Green = 0.14
col.Blue = 0.0
Else
col = clearColor
End If

chart.Geometry.SetParameterInt("ColorID", i)
chart.Geometry.SetParameterColor("Color", col)
Next
End Sub

Sub resetColors()

Dim chart As Container = GetParameterContainer("bars")

If chart = null Then Exit Sub

Dim valueString As String = Scene.map[GetParameterString("barsSMName")]
Dim values as Array[String]

Dim i As Integer = 0
valueString.split(",", values)

Dim col As Color = GetParameterColor("clearColor")

For i = 0 To values.Size
chart.Geometry.SetParameterInt("ColorID", i)
chart.Geometry.SetParameterColor("Color", col)
```

Copyright © 2026 Vizrt

```
Next
End Sub

Sub OnExecAction(buttonId As Integer)
If buttonId = 0 Then calcMinMax()
If buttonId = 1 Then resetColors()
End Sub

Sub OnInitParameters()
RegisterParameterContainer("bars", "Bar Chart")
RegisterParameterString("barsSMName", "Shared Mem Name", "bars", 40, 1000, "")
RegisterPushButton("minmax", "Calc Min/Max", 0)

Dim clearColor As Color

clearColor.Red = 1
clearColor.Green = 1
clearColor.Blue = 1

RegisterParameterColor("clearColor", "Clear Color", clearColor)
RegisterPushButton("reset", "Reset Colors", 1)
End Sub

Sub OnInit()
resetColors()
End Sub
```

The **ControlParameter** is set to **Text**, and the parameter to control is `SCRIPT*INSTANCE*clearColor`.

See Also

- Control Chart
- Data Fit
- Data Import
- Data Label
- Data Storage
- Basic Container Presenter
- Bar Stack

Copyright © 2026 Vizrt

Line Chart



The Line Chart plug-in draws a line chart, filled with data out of a Shared Memory Map.

You can use delimited strings or arrays for data transfer via Scene-, Global- or Distributed-Shared Memory.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

This page contains the following topics and procedures:

- Line Chart Properties
- To Create a Line Chart
- To Create a Line Chart with Data Animation
  - Chart Animation Using Data Import
  - Chart Animation Using Control Chart

Line Chart Properties

- **Shared Mem.:** Changes between Scene, Global and Distributed Shared Memory. Use Inactive memory to not forward any values via Shared Memory.
- **Specify X Values:** Enables DataX input.
  - **DataX:** Determines Shared Memory key name for X values. DataX is the default input parameter for X values.
  - **Key DataX:** Determines Shared Memory key name for X values.
- **DataY:** Determines Shared Memory key name for Y values. DataY is the default input parameter for Y values.
- **Key DataY:** Determines Shared Memory key name for Y values.
- **Transfer Mode:** Sets string- or array-based data transfer.
- **Data Delim.:** Defines the value separator sign(s).
- **Number Format:** Defines if Viz should get the decimal point separator from the `System Locale` or `Custom` settings. If Custom is enabled:
  - **Decimal Symbol:** Defines which symbol is used as decimal point when **Custom Number Format** is set.

> ⚠ **Note:** If the Line Stack plug-in is used, the following Line Chart properties need to be defined in the Line Stack plug-in.

- **DataX, -Y Fit:** Enables data normalization.
  - **DataX, -Y Scale:** Scales input by the selected factor.
  - **DataX, -Y Offset:** Adds an offset to the incoming data.
  - **DataX, -Y Auto Scale:** Enables automatic data normalization.
  - **DataX, -Y Fit Size:**

Copyright © 2026 Vizrt

        - **Total:** Scales the whole chart to the defined borders.
        - **Current:** Scales the current chart segment to the set borders.
    - **DataX, -Y Detect Limits:** Detects minimum and maximum of all values and scales them to adjusted Start and Stop. This option is used to upscale the interesting part of a chart - especially if there are only little changes between the data values.
    - **DataX, -Y Threshold:** Adds a definable offset to the detected limit.
    - **DataX, -Y Start:** Lower Auto Scale edge.
    - **DataX, -Y Stop:** Upper Auto Scale edge.
- **Values Animation:** Sets the vertical animation type to be created when data change is detected:
    - **Disable:** Does not create an animation.
    - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**. Also select an **Animation Mode**.
    - **Director:** Creates data transition animation with the Stage Director, that controls the Animation Progress parameter from 0.0% (old data) to 100.0% (new data). Also select an **Animation Mode**.
- **Animation Mode**
    - **Concurrent:** Changes all nodes concurrently.
    - **Left to Right:** Transitions from left to right.
    - **Right to Left:** Transitions from right to left.
- **Balance Speed:** Relates node values to the position of transformed data when set to `On` and Animation Mode is set to **Left to Right** or **Right to Left**.

Example:

- **Balance Speed:** If the values of four nodes change from 0,0,0,0 to 1,9,90,900 in **Left to Right** mode at 50% of the transformation.
- **Balance Speed Off:** Only the first two nodes are transformed to new values.
- **Balance Speed On:** The first three nodes are already transformed to new values, and the last node also transformed to 44% of the new values, because the data change from 0 (0+0+0+0) to 1000 (1+9+90+900), at 50% of transformation (in this case the values is changed for 500 of 1000), the first three nodes are already finished and the last node is changed to 400.

- **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.
- **Data Nodes Animation:** Sets the horizontal animation type to be created when number of data nodes change is detected:
    - **Disable:** No animation.
    - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**. Also select an **Animation Mode**
    - **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.

Note: When number of nodes is changes, line chart looks for added and removed nodes from their values, add new nodes to the chart from zero width and resize all nodes to their new size in specified duration.

- **Horizontal Alignment:** Sets horizontal orientation to left, center or right.
- **Line Type**
    - **Normal:** Direct connections between entered values.
    - **Spline:** Interpolates extra values to chamfer the graph.

Copyright © 2026 Vizrt

    - **Constrained Spline:** Same as Spline mode but with a different algorithm to prevent overshooting.
- **Chart Width:** Adjusts line width (`0` for 2D mode).
    - **Tessellation Width:** Sets tessellation length in spline modes.
- **Thickness:** Adjusts line thickness (`0` for 2D mode).
- **Depth:** Adjusts line depth (`0` for 2D mode).
- **Tessellation Thickness:** Tessellates line thickness.
- **Tessellation Depth:** Tessellates line depth.
- **Const. Thickness:** Builds the geometry with a constant width.
    - **Miter:** Applies a miter cut-off to avoid huge peaks for small data point angles.
- **Selection:** Specifies the Start- and Stop-type. Relative: 0%: 100%. Absolute: depends on your specified data. ID: Value ID starting from 0.
- **Start%, abs,:** Determines starting point of the graph.
- **Stop%, abs,:** Determines stop point of the graph.
- **Reposition Start:** Translates the whole chart always to the same starting point.
- **Bevel:** Activates Bevel mode.
    - **Bevel Start:** Chamfer start.
    - **Bevel Stop:** Chamfer stop.
    - **Bevel Size%:** Adjusts bevel's size from 0 to 100.
    - **Bevel Steps:** Sets roundness via the number of segmentation steps.

> ⚠ **Note:** A chamfer is a beveled edge connecting two surfaces.

- **RelativeMapping:** Stretches texture to fit graph length.
- **Pos. Container:** Translates every child container to a bar's top.
    - **Container Offset:** Adds a certain offset to each container.
    - **Value Dependent Pos.:** Translates the container above or below the line in dependency on the data values.
    - **Container Offset Z:** Adds a Z axis offset to each container.
    - **Center Container:** Centers each translated container.
- **Pointer:** You can use a container as a pointer and move it along the chart's shape.
    - **Container:** This container represents your pointer.
    - **Selection:** Specifies the position-type. Relative: 0%: 100%. Absolute: depends on your specified data. ID: Value ID starting from 0.
    - **Position%, abs,:** Translates pointer by this parameter.
    - **Offset:** Adds an offset to the container's position.
    - **Normal Offset:** Rotates the offset so that it is perpendicular to the chart's surface.
    - **Rotate:** Rotates the container in dependency on the chart's shape.
    - **Offset Z:** Adds a Z axis offset to the pointer.
    - **Center Container:** Uses the container's center for translation and rotation.
- **Low Pointer:** Activates the low pointer.
    - **Container:** Chooses a container for the pointer.
- **High Pointer:** Activates the high pointer.
    - **Container:** Chooses a container for the pointer.
- **Start Pointer:** Activates the Start Pointer.
    - **Container:** Chooses a container for the pointer.

Copyright © 2026 Vizrt

- **Stop Pointer:** Activates the stop pointer.
    - **Container:** Chooses a container for the pointer.

To Create a Line Chart



1. Create a new group and add the Line Chart plug-in to it.
2. Open the Transformation Editor and set **Position X** and **Y** to `-100.0` and **Rotation Y** to `-25.0`.
3. Add a material and/or a texture to it.
4. Open the Line Chart editor and set the following parameters:
    - Set **Shared Mem.** to **Inactive**
    - Set the following **DataY** values: `10,20,30,20,50,60,80,45,20`.
    - Set **Line Type** to **Spline**.
    - Set **Chart Width** to `250.0`.
    - Set **Depth** to `10.0`.
    - Enable **Bevel**, **Bevel Start**, **Bevel Stop**.
    - Set **Bevel Size%** to `100.0`.

Copyright © 2026 Vizrt

- Set **Bevel Steps** to `10.0`.

To Create a Line Chart with Data Animation

Chart Animation Using Data Import

The following steps are to prepare data sets for animation in Excel file.

1. Start Microsoft Excel.
2. Enter **ExcelData1** into cell A1.
3. Add some sample values in the cells below (`A2-A8: 80, 35, 45, 75, 85, 55, 60`).
4. Enter **ExcelData2** into cell A2.
5. Add some sample values in the cells below (`B2-B9: 80, 35, 45, 75, 40, 85, 55, 60`).
6. Enter **ExcelData3** into cell A3.
7. Add some sample values in the cells below (`C2-C9: 40, 60, 75, 85, 80, 55, 45, 35`).
8. Enter **ExcelData4** into cell A4.
9. Add some sample values in the cells below (`D2-D8: 60, 75, 85, 80, 55, 45, 35`).
10. Rename this first sheet to **MyTable** (can be done with a double click on the sheet name at the bottom).
11. Follow one of the following methods to make Viz Engine can read this Excel data file.
    a. **Save and close** the Excel document.
    b. **Share** the Excel document.
        i. On the Review tab, in the Changes group, click the **Share Workbook** button.
        ii. The Share Workbook dialog box appears, and you select the **Allow changes by more than one user at the same time.** check box on the Editing tab.
        iii. Click **OK**.
        iv. Save the Excel document.

> **IMPORTANT!** You must use the same platform (x64/x86) of Microsoft Excel and Viz Engine.

After the data sets are created, the following steps are to create a Data Animation scene.

1. Follow the "To create a Line Chart" instruction.
2. Modify the following parameters of the Line Chart plug-in.
    - Change **Shared Mem.** to **Scene.**
    - Set **Kay DataY** to `DataY`.
    - Set **Data Delim.** to `#`.
    - Set **Values Animation** to `Timer`.
    - Set **Data Nodes Animation** to `Timer`.
3. Add Data Import to the same container.
    a. Set **File** to above prepared Excel file.
    b. Set **Table / Sheet** to `MyTable`.
    c. Set **Column(,Col...)** to `ExcelData1`.
    d. Set **Data Delim.** to `#`.
    e. Set **Shared Mem.** to **Scene.**
    f. Set **Key** to **DataY.**
4. Click the **GetIt** button and animation chart shows from zero to seven nodes.
5. Add one data node in the middle of chart.
    a. Set **Column(,Col...)** to **ExcelData2.**

Copyright © 2026 Vizrt

b. Click the **GetIt** button.

6. Change values of eight data nodes.
    a. Set **Column(,Col...)** to `ExcelData3`.
    b. Click the **GetIt** button.
7. Remove the first data node.
    a. Set **Column(,Col...)** to `ExcelData4`.
    b. Click the **GetIt** button.

> **Note:** You can add more data columns and play with animation on difference data change.

Chart Animation Using Control Chart

1. Follow the "To create a Line Chart" instruction.
2. Modify the following parameters of the Line Chart plug-in.
    - Set **Values Animation** to **Timer.**
    - Set **Data Nodes Animation** to **Timer.**
3. Add Control Chart to the same container and save the scene.
4. Open Viz Trio.
    - Import the scene.
    - Click "1 (LineChart)" in Tab Fields.
    - Edit values in Editing Template.

See Also

- Control Chart
- Data Fit
- Data Import
- Data Label
- Data Storage
- Line Stack

Copyright © 2026 Vizrt

Pie Chart



The Pie Chart plug-in draws a pie chart, filled with data out of a Shared Memory Map.

You can use delimited strings or arrays for data transfer via Scene-, Global- or Distributed-Shared Memory.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

This page contains the following topics and procedures:

- Pie Chart Properties
- To Create a Pie Chart
- To Create a Pie Chart with Data Animation
  - Chart Animation Using Data Import
  - Create a Data Animation scene
  - Chart Animation Using Control Chart

Pie Chart Properties

- **Shared Mem.:** Changes between **Scene**, **Global** and **Distributed** Shared Memory. Use **Inactive** memory to not forward any values via Shared Memory and the data is taken from the **Data** field.
- **Data:** Determines Shared Memory key name. Data is the default input parameter.
- **Transfer Mode:** Sets string- or array-based data transfer.
- **Data Delim.:** Defines the value separator sign(s).
- **Number Format:** Defines if Viz should get the decimal point separator from the `System Locale` or `Custom` settings. If Custom is enabled:
  - **Decimal Symbol:** Defines which symbol is used as decimal point when **Custom Number Format** is set.
- **Data Fit:** Enables data normalization.
  - **Data Offset:** Adds an offset to the incoming data.
- **Values Animation:** Sets the animation type to be created when data change is detected.
  - **Disable:** No animation.
  - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**. Also select an **Animation Mode**.
  - **Director:** Creates data transition animation with the Stage Director, that controls the Animation Progress parameter from 0.0% (old data) to 100.0% (new data). Also select an **Animation Mode**.
- **Animation Mode:**
  - **Concurrent:** Changes all nodes concurrently.
  - **Left to Right:** Transitions from left to right.
  - **Right to Left:** Transitions from right to left.

Copyright © 2026 Vizrt

- **Balance Speed:** Relates node values to the position of transformed data when set to `On` and Animation Mode is set to **Left to Right** or **Right to Left**.

> **Example:**
> - **Balance Speed:** If the values of four nodes change from 0,0,0,0 to 1,9,90,900 in **Left to Right** mode at 50% of the transformation.
> - **Balance Speed Off:** Only the first two nodes are transformed to new values.
> - **Balance Speed On:** The first three nodes are already transformed to new values, and the last node also transformed to 44% of the new values, because the data change from 0 (0+0+0+0) to 1000 (1+9+90+900), at 50% of transformation (in this case the values is changed for 500 of 1000), the first three nodes are already finished and the last node is changed to 400.

- **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.
- **Diameter:** Sets size of pie chart.
- **Total Segment deg:** Sets the pie chart's maximum angle in degrees (default: 100% equals 360 degrees).
- **Inner Cutoff:** Cuts off a round piece from the cake's center.
- **Height:** Adjusts height.
- **Corners:** Defines the minimum amount of corners at the pie's edge. In dependency on the number of pieces an algorithm adds corners to always give the pie chart the same appearance.
- **Keep Proportions:** If enabled, the pie chart would look real. Otherwise each piece's radius is recalculated.
- **Const. Offset:** Sets a constant gap width.
- **Center Offset:** Determines each pieces' distance from the pie's center.
- **Piece Scaling:**
  - **All:** All pieces equal 100%.
  - **Single:** Every piece's size is described by its own 100% (for example, three pieces mean 300%).
  - **Fit:** All pieces equal 100%, but they are scaled relatively to fit their new area with the correct values.
- **Piece Size%:** Parameter for the previous option.
- **Bevel:** Activates Bevel mode.
  - **Bevel Top:** Chamfers cake's top.
  - **Bevel Bottom:** Chamfers pie's bottom.
  - **Bevel Size%:** Adjusts bevel's size from 0 to 100.
  - **Bevel Steps:** Sets roundness via the number of segmentation steps.
- **Move Piece:** Defines the piece for the next parameter operation. Starts with 0, -1 means nothing selected.
  - **Center Offset:** Offsets center for a certain piece.
  - **Height Offset:** Offsets height for a certain piece.
  - **Effect Range:** Sets the range of pieces to move.
  - **Effect Distribution:** Sets the distribution of the effect range.
- **Pos. Container:** Translates every child container to a piece of cake.
  - **Container Offset:** Adds a certain center offset to each container.
  - **Center Container:** Centers each translated container.
  - **PosX, Y, Z:** Activates container translation on the particular axis.
  - **Container Pos. Z%:** Sets the relative position on the z axis for each child container.
- **Map**
  - **Color Bar:** Sets a single color (V texture coordinate) for a certain piece (for example, three pieces: V = `0.0`, V = `0.5` and V = `1.0`).

Copyright © 2026 Vizrt

- **Vertex:** Uses vertex colors for each piece starting with ID 0. **Color ID** moves between the available vertex colors. **Color** lets you choose color for the current ID.

Note: A chamfer is a beveled edge connecting two surfaces.

To Create a Pie Chart



1. Create a new group and add the Pie Chart plug-in to it.
2. Open the Transformation Editor and set Position X and Y to -100.0 and Rotation Y to `-25.0`.
3. Add a material and/or a texture to it.
4. Open the Pie Chart editor and set the following parameters:
   - Set the following DataY values: `10,20,30,20,50,60,80,45,20`.
   - Enable Keep Proportions.
   - Set Center Offset to `6.0`.
   - Set Move Piece to `1`.
   - Set Center and Height Offset to `20.0`.
   - Enable Vertex and set different colors for all Color IDs.

To Create a Pie Chart with Data Animation

Chart Animation Using Data Import

Prepare data sets for animation in Excel file.

1. Start Microsoft Excel.
2. Enter **ExcelData1** into cell A1.
3. Add some sample values in the cells below (`A2-A8: 80, 35, 45, 75, 85, 55, 60`).
4. Enter **ExcelData2** into cell A2.
5. Add some sample values in the cells below (`B2-B9: 80, 35, 45, 75, 40, 85, 55, 60`).
6. Enter **ExcelData3** into cell A3.
7. Add some sample values in the cells below (`C2-C9: 40, 60, 75, 85, 80, 55, 45, 35`).
8. Enter **ExcelData4** into cell A4.
9. Add some sample values in the cells below (`D2-D8: 60, 75, 85, 80, 55, 45, 35`).
10. Rename this first sheet to **MyTable** (can be done with a double click on the sheet name at the bottom).
11. Follow one of the following methods to make Viz Engine can read this Excel data file.
    a. **Save and close** the Excel document.

Copyright © 2026 Vizrt

b. **Share** the Excel document.
   i. On the Review tab, in the Changes group, click the **Share Workbook** button.
   ii. The Share Workbook dialog box appears, and you select the **Allow changes by more than one user at the same time.** check box on the Editing tab.
   iii. Click **OK**.
   iv. Save the Excel document.

IMPORTANT! You must use the same platform (x64/x86) of Microsoft Excel and Viz Engine.

Create a Data Animation scene

1. Follow the "To create a Pie Chart" instruction.
2. Modify the following parameters of the Pie Chart plug-in.
   - Change **Shared Mem.** to `Scene`.
   - Set **Data Delim.** to `#`.
   - Set **Values Animation** to `Timer`.
3. Add Data Import to the same container.
   a. Set **File** to above prepared Excel file.
   b. Set **Table / Sheet** to `MyTable`.
   c. Set **Column(,Col...)** to `ExcelData1`.
   d. Set **Data Delim.** to `#`.
   e. Set **Shared Mem.** to `Scene`.
   f. Set **Key** to `Data`.
4. Click the **GetIt** button and chart shows from zero to seven nodes.
5. Change to the another data set.
   a. Set **Column(,Col...)** to **ExcelData2** or **ExcelData3** or **ExcelData4.**
   b. Click the **GetIt** button.
6. Repeat step 5 with another data set including **ExcelData1**.

Note: You can add more data columns and play with animation on difference data change.

Chart Animation Using Control Chart

1. Follow the "To create a Pie Chart" instruction.
2. Modify the following parameters of the Pie Chart plug-in.
3. Change **Values Animation** property to **Timer.**
4. Add Control Chart to the same container and save the scene.
5. Open Viz Trio.
   - Import the scene.
   - Click "1 (PieChart)" in Tab Fields.
   - Edit values in Editing Template.

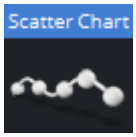
See Also

- Control Pie
- Data Fit
- Data Import

Copyright © 2026 Vizrt

- Data Label
- Data Storage
- Basic Container Presenter

Scatter Chart



The Scatter Chart plug-in draws a scatter chart, filled with data out of a Shared Memory Map.

You can use delimited strings or arrays for data transfer via Scene-, Global- or Distributed-Shared Memory.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

This page contains the following topics and procedures:

- Scatter Chart Properties
- To Create a Scatter Chart
- To Create a Scatter Chart with Data Animation
  - Chart Animation Using Data Import
  - Create a Data Animation Scene
  - Chart Animation Using Control Chart

Scatter Chart Properties

- **Shared Mem.:** Changes between **Scene**, **Global** and **Distributed** Shared Memory. Use **Inactive** memory to not forward any values via Shared Memory and the data is taken from the **Data** field.
- **Key DataX:** Determines Shared Memory key name for X values. DataX is the default input parameter for X values.
- **Key DataY:** Determines Shared Memory key name for Y values. DataY is the default input parameter for Y values.
- **DataZ Position:** Uses DataZ for Z-axis positioning.
- **Key DataZ:** Determines Shared Memory key name for Z values. DataZ is the default input parameter for Z values.
- **Transfer Mode:** Sets string- or array-based data transfer.
- **Data Delim.:** Defines the value separator sign(s).
- **Number Format:** Defines if Viz should get the decimal point separator from the `System Locale` or `Custom` settings. If Custom is enabled:
  - **Decimal Symbol:** Defines which symbol is used as decimal point when **Custom Number Format** is set.
- **DataX, -Y, -Z Fit:** Enables data normalization.
- **DataX, -Y, -Z Scale:** Scales input by the selected factor.
- **DataX, -Y, -Z Offset:** Adds an offset to the incoming data.
- **DataX, -Y, -Z Auto Scale:** Enables automatic data normalization.
- **DataX, -Y, -Z Detect Limits:** Detects minimum and maximum of all values and scales them to adjusted Start and Stop.
- **DataX, -Y, -Z Threshold:** Adds a definable offset to the detected limit.

Copyright © 2026 Vizrt

- **DataX, -Y, -Z Start:** Determines lower Auto Scale edge.
- **DataX, -Y, -Z Stop:** Determines upper Auto Scale edge.
- **Values Animation:** Sets the vertical animation type to be created when data change is detected:
  - **Disable:** Does not create an animation.
  - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**. Also select an **Animation Mode**.
  - **Director:** Creates data transition animation with the Stage Director, that controls the Animation Progress parameter from 0.0% (old data) to 100.0% (new data). Also select an **Animation Mode**.
- **Animation Mode:**
  - **Concurrent:** Changes all nodes concurrently.
  - **Left to Right:** Transitions from left to right.
  - **Right to Left:** Transitions from right to left.
- **Balance Speed:** Relates node values to the position of transformed data when set to `On` and Animation Mode is set to **Left to Right** or **Right to Left**.

Example:
- **Balance Speed:** If the values of four nodes change from 0,0,0,0 to 1,9,90,900 in **Left to Right** mode at 50% of the transformation.
- **Balance Speed Off:** Only the first two nodes are transformed to new values.
- **Balance Speed On:** The first three nodes are already transformed to new values, and the last node also transformed to 44% of the new values, because the data change from 0 (0+0+0+0) to 1000 (1+9+90+900), at 50% of transformation (in this case the values is changed for 500 of 1000), the first three nodes are already finished and the last node is changed to 400.

- **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.
- **Data Nodes Animation:** Sets the horizontal animation type to be created when number of data nodes change is detected.
  - **Disable:** Does not create an animation.
  - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**
  - **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.

Note: When the number of nodes is changes, scatter chart looks for added and removed nodes from their values, adds new nodes to the chart from zero width and resizes all nodes to their new size in specified duration.

- **Horizontal Alignment:** Sets horizontal orientation to left, center or right.
- **Chart Width:** Adjusts the chart width.
- **Container:** Determines the container used for the scatter nodes.
- **DataZ SizeX:** Allows Z Values to influence X scaling.
- **DataZ SizeY:** Allows Z Values to influence Y scaling.
- **DataZ SizeZ:** Allows Z Values to influence Z scaling.
- **Center Container:** Centers each translated container.
- **Fit Width:** Considers the width of the scatter node containers which means that the containers do not reach outside of the specified range.

Copyright © 2026 Vizrt

- **Animate Position:** Sets the translation type: None: no animation, Y: animates the container along the Y axis (suggestion: Scale Y + DataLabel for BarChart alike labeling).
- **Animate Scaling:** Sets the scaling type: None: no scaling at all, XYZ: linear scaling in all directions, Y: scales in y direction only (->BarChart).
    - **Relative Length:** Provides each scatter node with its own 100% size (for example seven nodes equal 700%).
    - **Const. Speed:** Sets the same animation duration for each bar.
    - **Total Length%:** Sets the accumulated size of all scatter nodes in percent.

To Create a Scatter Chart



1. Create a group and add the Sphere plug-in to it.
2. Open the transformation editor and set the Scaling (locked) to `0.2`.
3. Add a material and/or a texture to the sphere.
4. Set the group with the sphere to hidden.
5. Create a new group and add the Scatter Chart plug-in to it.

Copyright © 2026 Vizrt

6. Open the Transformation Editor and set Position X and Y to `-100.0`.
7. Open the Scatter Chart editor and set the following parameters:
    - Set the following DataY values: `10,20,30,20,50,60,80,45,20`.
    - Set Width to `250.0`.
    - Drag and drop the container with the Sphere to the Container placeholder.

To Create a Scatter Chart with Data Animation

Chart Animation Using Data Import

Prepare data sets for animation in Excel file.

1. Start Microsoft Excel.
2. Enter **ExcelData1** into cell A1.
3. Add some sample values in the cells below (`A2-A8: 80, 35, 45, 75, 85, 55, 60`).
4. Enter **ExcelData2** into cell A2.
5. Add some sample values in the cells below (`B2-B9: 80, 35, 45, 75, 40, 85, 55, 60`).
6. Enter **ExcelData3** into cell A3.
7. Add some sample values in the cells below (`C2-C9: 40, 60, 75, 85, 80, 55, 45, 35`).
8. Enter **ExcelData4** into cell A4.
9. Add some sample values in the cells below (`D2-D8: 60, 75, 85, 80, 55, 45, 35`).
10. Rename this first sheet to **MyTable** (can be done with a double click on the sheet name at the bottom).
11. Follow one of the following methods to make Viz Engine can read this Excel data file.
    a. **Save and close** the Excel document.
    b. **Share** the Excel document.
        i. On the Review tab, in the Changes group, click the **Share Workbook** button.
        ii. The Share Workbook dialog box appears, and you select the **Allow changes by more than one user at the same time.** check box on the Editing tab.
        iii. Click **OK**.
        iv. Save the Excel document.

> **IMPORTANT!** You must use the same platform (x64/x86) of Microsoft Excel and Viz Engine.

Create a Data Animation Scene

1. Follow the "To create a Scatter Chart" instruction.
2. Modify the following parameters of the Scatter Chart plug-in.
    - Change **Shared Mem.** to `Scene`.
    - Set **Key DataY** to `DataY`.
    - Set **Data Delim.** to `#`.
    - Set **Values Animation** to `Timer`.
    - Set **Data Nodes Animation** to `Timer`.
3. Add Data Import to the same container.
    a. Set **File** to above prepared Excel file.
    b. Set **Table / Sheet** to `MyTable`.
    c. Set **Column(,Col...)** to `ExcelData1`.

Copyright © 2026 Vizrt

d. Set **Data Delim.** to `#`.
e. Set **Shared Mem.** to `Scene`.
f. Set **Key** to `DataY`.

4. Click the **GetIt** button and animation chart shows from zero to seven nodes.
5. Add one data node in the middle of chart.
   a. Set **Column(,Col...)** to `ExcelData2`.
   b. Click the **GetIt** button.
6. Change values of eight data nodes.
   a. Set **Column(,Col...)** to `ExcelData3`.
   b. Click the **GetIt** button.
7. Remove the first data node.
   a. Set **Column(,Col...)** to `ExcelData4`.
   b. Click the **GetIt** button.

> **Note:** You can add more data columns and play with animation on difference data change.

Chart Animation Using Control Chart

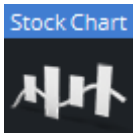
1. Follow the "To create a Scatter Chart" instruction.
2. Modify the following parameters of the Scatter Chart plug-in.
   - Set **Values Animation** to **Timer.**
   - Set **Data Nodes Animation** to **Timer.**
3. Add Control Chart to the same container and save the scene.
4. Open Viz Trio.
   - Import the scene.
   - Click "1 (ScatterChart)" in Tab Fields.
   - Edit values in Editing Template.

See Also

- Control Chart
- Data Fit
- Data Import
- Data Label
- Data Storage

Copyright © 2026 Vizrt

Stock Chart



The Stock Chart plug-in draws a stock chart, filled with data out of a Shared Memory Map.

You can use delimited strings or arrays for data transfer via Scene-, Global- or Distributed-Shared Memory.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

This page contains the following topics and procedures:

- Stock Chart Properties
- To Create a Stock Chart
- To Create a Stock Chart with Data Animation
  - Chart Animation using Data Storage
  - Chart Animation Using Control Chart

Stock Chart Properties

- **Shared Mem.:** Changes between Scene-, Global- and Distributed-Shared Memory. Use Inactive memory to not forward any values via Shared Memory.
- **Key Open:** Determines Shared Memory key name for open values. DataOpen is the default input parameter for open values.
- **Key Close:** Determines Shared Memory key name for close values. DataClose is the default input parameter for close values.
- **Key High:** Determines Shared Memory key name for high values. DataHigh is the default input parameter for high values.
- **Key Low:** Determines Shared Memory key name for low values. DataLow is the default input parameter for low values.
- **Transfer Mode:** Sets string- or array-based data transfer.
- **Data Delim.:** Defines the value separator sign(s).
- **Number Format:** Defines if Viz should get the decimal point separator from the `System Locale` or `Custom` settings. If Custom is enabled:
  - **Decimal Symbol:** Defines which symbol is used as decimal point when **Custom Number Format** is set.
- **Data Open, -Close, -High, -Low Fit:** Enables data normalization.
  - **Data Open, -Close, -High, -Low Scale:** Scales input by the selected factor.
  - **Data Open, -Close, -High, -Low Offset:** Adds an offset to the incoming data.
  - **Data Auto Scale:** Enables automatic data normalization.
- **Value Animation:** Sets the vertical animation type to be created when data change is detected:
  - **Disable:** No animation.

Copyright © 2026 Vizrt

- **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**. Also select an **Animation Mode**.
- **Director:** Creates data transition animation with the Stage Director, that controls the Animation Progress parameter from 0.0% (old data) to 100.0% (new data). Also select an **Animation Mode**.

- **Animation Mode:**
  - **Concurrent:** Changes all nodes concurrently.
  - **Left to Right:** Transitions from left to right.
  - **Right to Left:** Transitions from right to left.
- **Balance Speed:** Relates node values to the position of transformed data when set to `On` and Animation Mode is set to **Left to Right** or **Right to Left**.

Example:
- **Balance Speed:** If the values of four nodes change from 0,0,0,0 to 1,9,90,900 in **Left to Right** mode at 50% of the transformation.
- **Balance Speed Off:** Only the first two nodes are transformed to new values.
- **Balance Speed On:** The first three nodes are already transformed to new values, and the last node also transformed to 44% of the new values, because the data change from 0 (0+0+0+0) to 1000 (1+9+90+900), at 50% of transformation (in this case the values is changed for 500 of 1000), the first three nodes are already finished and the last node is changed to 400.

- **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.
- **Data Nodes Animation:** Sets the horizontal animation type to be created when number of data nodes change is detected.
  - **Disable:** No animation.
  - **Timer:** Creates data transition animation (from an old set of data to a new set) within the specified time set in **Duration sec.**
  - **Duration sec.:** Sets the duration of the animation in seconds for **Timer** mode.

Note: When number of nodes is changes, stack chart looks for added and removed nodes from their values, adds new nodes to the chart from zero width and resizes all nodes to their new size in specified duration.

- **Data Detect Limits:** Detects minimum and maximum of all values and scales them to adjusted Start and Stop.
- **Data Start:** Determines lower auto scale edge.
- **Data Stop:** Determines upper auto scale edge.
- **Draw Open:** Draws open data.
- **Draw Close:** Draws close data.
- **Draw High:** Draws high data.
- **Draw Low:** Draws low data.
- **Candle Corners:** Sets candle segments.
- **Wick Corners:** Sets wick corners.
- **Width Scaling**
  - **Single Candle:** Adjusts size for a single candle.
  - **Fit:** Adjusts all candles to fit a certain width.

Copyright © 2026 Vizrt

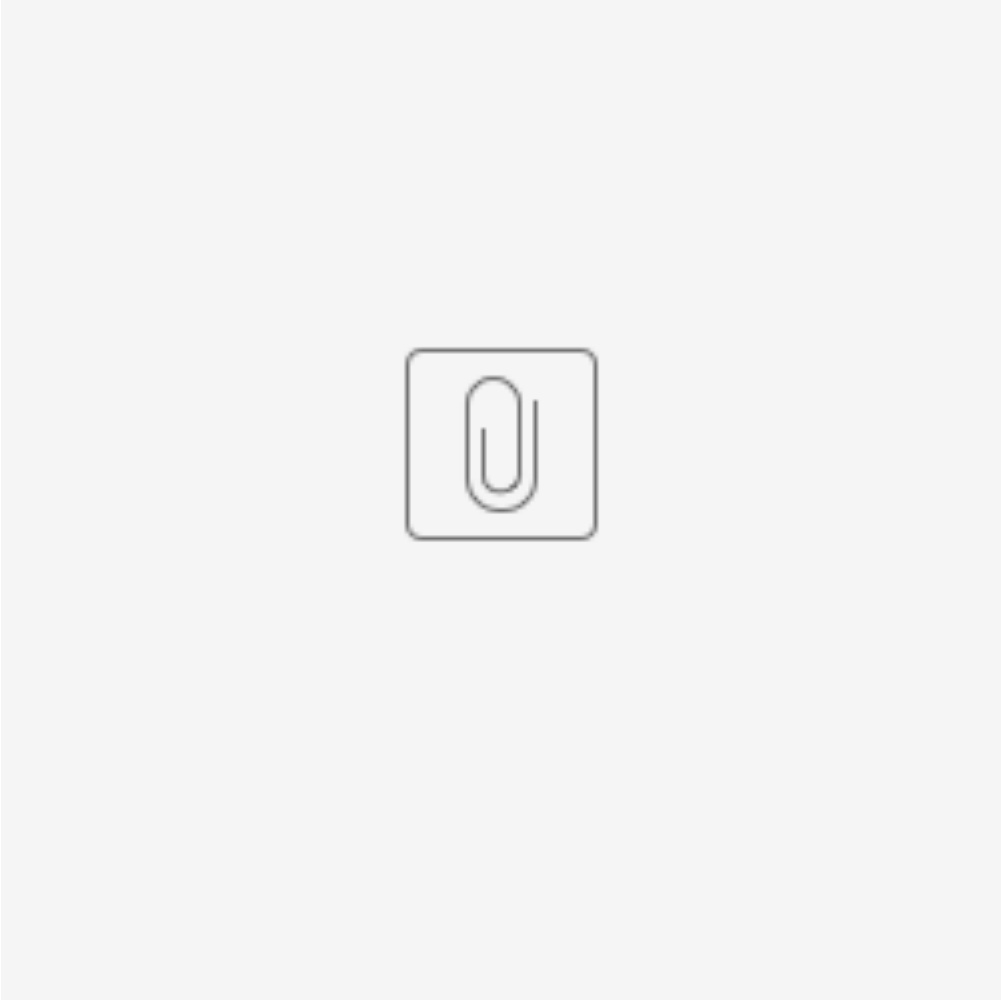
- **Candle Width:** Value for the previous parameter.
- **Candle Gap Ratio%:** 100% means that the gaps have the same size as the candles.
- **Candle Wick Ratio%:** 100% means that the wicks have the same size as the candles.
- **Bevel Candle:** Activates Bevel mode for the candle geometry.
    - **Bevel Candle Top:** Chamfers candle's top.
    - **Bevel Candle Bottom:** Chamfers candle's bottom.
    - **Bevel Candle Size%:** Adjusts bevel's size from 0 to 100.
    - **Bevel Candle Steps:** Sets roundness via the number of segmentation steps.
- **Bevel Wick:** Activates Bevel mode for the wick geometry.
    - **Bevel Wick Top:** Chamfers wick's top.
    - **Bevel Wick Bottom:** Chamfers wick's bottom.
    - **Bevel Wick Size%:** Adjusts bevel's size from 0 to 100.
    - **Bevel Wick Steps:** Sets roundness via the number of segmentation steps.
- **Relative Length**
    - **On:** Sets all accumulated candle lengths to equal 100%.
    - **Off:** Sets each candle to its own 100% (for example, three candles: 300%).
- **Total Length:** Value for the previous parameter.
- **Use Vertex Colors:** Activates the following parameters.
    - **Bull Candle Color:** Defines the color for all candles where the close value is higher than the open value.
    - **Bear Candle Color:** Defines the color for all candles where the close value is lower than the open value.
    - **High Wick Color:** Sets high wick's color.
    - **Low Wick Color:** Sets low wick's color.



Note: A chamfer is a beveled edge connecting two surfaces.

Copyright © 2026 Vizrt

To Create a Stock Chart



1. Create a new group and add the Stock Chart plug-in to it.
2. Open the Transformation Editor and set Position X and Y to `-100.0`.
3. Open the Stock Chart editor and set the following parameters:
    - Set the Data Open parameter to the following values: `10,20,30,40,50,40,30,40,50`.
    - Set the Data Close parameter to the following values: `20,15,35,45,40,60,45,20,20`.
    - Set the Data High parameter to the following values: `25,20,40,50,55,60,50,40,50`.
    - Set the Data Low parameter to the following values: `10,15,30,30,25,30,30,20,15`.
    - Set Candle and Wick Corners to `10`.
    - Set Candle Width to `25.0`.

Copyright © 2026 Vizrt

To Create a Stock Chart with Data Animation

Chart Animation using Data Storage

1. Follow the "To create a Stock Chart" instruction.
2. Modify the following parameters of the Stock Chart plug-in.
    - Change **Shared Mem.** to `Scene`.
    - Set **Values Animation** to `Timer`.
    - Set **Data Nodes Animation** to `Timer`.
    - Set **Width Scaling** to `Fit`.
    - Set **Candle Width** to `325`.
3. Add Data Storage into the same container.
    - Set **Key Data1** to **DataOpen.**
    - Set **Data1** to `10,20,30,40,40,30,40,50`.
    - Set **Key Data2** to **DataClose.**
    - Set **Data2** to `20,15,35,45,60,45,20,20`.
    - Set **Key Data3** to **DataHigh.**
    - Set **Data3** to `25,20,40,50,60,50,40,50`.
    - Set **Key Data4** to **DataLow.**
    - Set **Data4** to `10,15,30,30,30,30,20,15`.
4. Change **Shared Mem.** to **Scene** and animation chart shows from nine to eight nodes.
5. Update data with the same number of nodes.
    - Change **Shared Mem.** to **Inactive.**
    - Set **Data1** to `40,30,50,25,20,40,55,45`.
    - Set **Data2** to `25,45,20,20,30,45,50,40`.
    - Set **Data3** to `55,50,50,40,40,55,55,50`.
    - Set **Data4** to `10,15,10,15,15,20,40,30`.
6. Change **Shared Mem.** to **Scene** and values of nodes change in the animation.
7. Update data by adding two nodes.
    - Change **Shared Mem.** to **Inactive.**
    - Set **Data1** to `40,30,20,50,25,20,40,30,55,45`.
    - Set **Data2** to `25,45,45,20,20,30,45,45,50,40`.
    - Set **Data3** to `55,50,50,50,40,40,55,55,55,50`.
    - Set **Data4** to `10,15,20,10,15,15,20,25,40,30`.
8. Change **Shared Mem.** to **Scene** and animation chart shows from eight to ten nodes.

IMPORTANT! You must use the same platform (x64/x86) of Microsoft Excel and Viz Engine.

Chart Animation Using Control Chart

1. Follow the "To create a Stock Chart" instruction.
2. Modify the following parameters of the Stock Chart plug-in.

Copyright © 2026 Vizrt

- Set **Values Animation** to `Timer`.
- Set **Data Nodes Animation** to `Timer`.
- Set **Width Scaling** to `Fit`.
- Set **Candle Width** to `325`.

3. Add Control Chart to the same container and save the scene.
4. Open Viz Trio.
    - Import the scene.
    - Click "1 (StockChart)" in Tab Fields.
    - Edit values in Editing Template.

See Also

- Control Chart
- Data Fit
- Data Import
- Data Label
- Data Storage

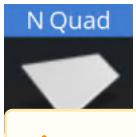
Copyright © 2026 Vizrt

2.3.4 Primitives Plug-ins

The following Geometry plug-ins are located in the Primitives folder:

- N Quad
- P Quad

N Quad



The N Quad plug-in creates a rectangle with different attributes.

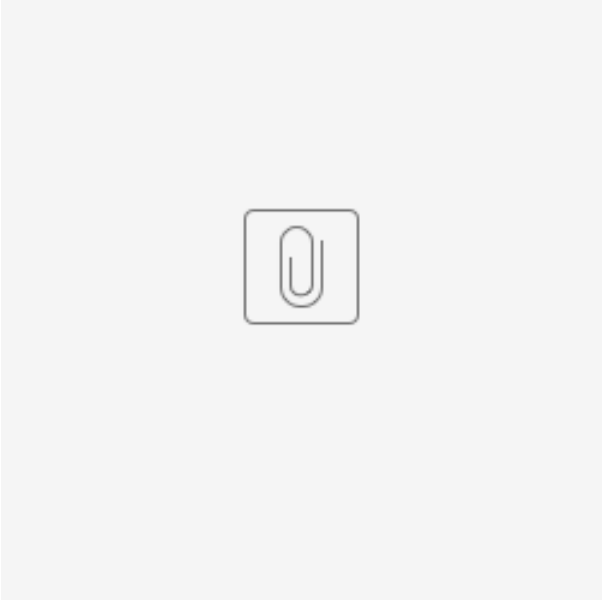
Note: This plug-in is located in: Plugins -> Geom plug-ins -> Primitives

N Quad Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Tessellation U:** Sets the degree of detail.
- **Tessellation V:** Sets the degree of detail.
- **Orientation:** Switches the orientation from **Top Bottom** to **Left Right**.
- **Top/Left Width:** Sets the width.
- **Bottom Right Width:** Sets the width.
- **Height Width:** Sets the height.
- **Depth:** Sets the depth.
- **Center X:** Switches the center in X between **Center**, **Top** and **Bottom**.
- **Center Y:** Switches the center in Y between **Center**, **Left** and **Right**.
- **Shearing:** Sets the shearing value, so the bottom is shifted.
- **Image Mapping:** May be set to **Stretch** or **Tile**.
- **Crop Top Left:** Crops the top.
- **Crop Bottom Right:** Crops the bottom.
- **Border Geometry:** Enables/disables the top and bottom border width parameters. When enabled, only the border outline is visible.
- **Top Border Width:** Sets the width of the top border.
- **Bottom Border Width:** Sets the width of the bottom border.

Copyright © 2026 Vizrt

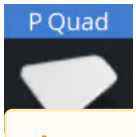
To Create an N Quad



1. Create a new group and add the N Quad plug-in to it.
2. Add a material and/or a texture to the group.
3. Open the transformation editor and set Rotation X and Z to `-45.0`.
4. Open the N Quad editor and set the following parameters:
    - Set Orientation to Left-Right.
    - Set Top/Left Width to `200.0`.
    - Set Bottom/Right Width to `0.0`.
    - Set Height/Width to `300.0`.
    - Set Depth to `100.0`.
    - Set Center X and Y to `Center` and `Bottom`, respectively.

Copyright © 2026 Vizrt

P Quad



The P Quad plug-in creates a rectangle with different attributes.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Primitives

P Quad Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Tesselation:** Sets the degree of detail.
- **Bevel:** Sets the radius of rounded edges.
- **Width:** Sets the width of the quad.
- **Height:** Sets the height of the quad.
- **Corners:** Sets the number of the corners for rounded edges.
- **Use Control Vertices:** Allows the four edges of the quad to be moved individually. With the following parameters, edges can be placed in X and Y directions as desired:
  - **LowerLeft X**
  - **LowerLeft Y**
  - **LowerRight X**
  - **LowerRight Y**
  - **UpperRight X**
  - **UpperRight Y**
  - **UpperLeft X**
  - **UpperLeft Y**
- **Use Vertexcolors:** Allows the color of the edges to be set when enabled. With the following parameters, individual colors can be selected:
  - **UpperLeft**
  - **Lower Left**
  - **Upper Right**
  - **Lower Right**

Copyright © 2026 Vizrt

2.3.5 RealFX Geometry Plug-ins

The RealFX plug-in set enables you to create particle effects in Viz Artist.

Particle systems are a computer graphics technique to simulate certain physics-based effects, which are otherwise very hard to reproduce with conventional rendering techniques. Examples of such effects which are commonly replicated using particle systems include fire, explosions, smoke, weather effects, sparks, falling leaves, dust, meteor tails, or abstract visual effects like glowing trails, magic spells, etc.

The particle effects in Viz Artist/Engine run in real-time, meaning that there are a few inherent constraints that must be taken into account when considering best practices for employing this plug-in set. For example, there is a trade-off between the number of particles and performance optimization; more generally there needs to be a considered balance between performance and visual quality.

RFxSmoke is the baseline plug-in within the RealFX plug-in set. The remaining plug-ins in this set are applied on top of RFxSmoke in any given container. RFxSmoke includes built-in functionality and the ability to host the additional functionality contained in the other plug-ins in this set. Part of the built-in functionality, e.g. turbulence, is kept for compatibility with previous version of Viz Artist.

There are three categories of additional plug-ins:

- **Birth plug-ins:** Refers to where the particles are spawned.
- **Motion control plug-ins:** Governs the position, direction, velocity, size and color of each particle.
- **Texture control plug-ins:** Affects the texture mapping and the “look” of each particle by using pixel shader technology.

Copyright © 2026 Vizrt



The following Geometry plug-ins are located in the RealFX folder:

- RFxSmoke

See Also

- RealFX in Basic Shader Plug-ins
- pxColorWorks in Basic Container Plug-ins

Copyright © 2026 Vizrt

RFxSmoke



The RFxSmoke plug-in is the baseline plug-in within the RealFX plug-in set, which creates realistic visualization of smoke, fire, explosions and much more (the level of creative skill is the only limitation).

RFxSmoke emits particles from a point and moves them in some direction until they die or fade away. With the use of different texture mappings and settings, many effects can be achieved.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> RealFX

This page contains the following topics:

- RFxSmoke Properties
- Spawn
- Impulse
- Forces
- Size
- Rotation
- Color
- Alpha

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

RFxSmoke Properties

- **Number of particles:** Sets the number of particles to be active at the same time. The number of **required** particles is calculated by the **Birth Rate** multiplied by the **Life Span**. If this is lower or equal, the number of particles present is a constant flow of particles, otherwise the plug-in collects particles until the minimum number of particles has been reached and then spawns them.
- **Birth Rate:** Sets the birth rate of particles per second.
- **Life Span:** Sets the life span of the objects in seconds. In the field to the right of Life Span, set a percentage over which the life span of the particles varies. If **Life Span** is set to three and the variation to 50%, the life span of particles vary randomly between 1,5 and 4,5 seconds. This creates more realism, since particles in real smoke or fire does not have equal life span.
- **Preroll (fields):** Sets the amount of fields to calculate before RFxSmoke goes On Air.
- **Reference:** Sets the particles reference to either **Container** or **World**. If **Container** is selected, after emitted the particles relate their position to the containers position. If the emitter object is moved, the particles alter their position correspondingly. If **World** is chosen, the particles maintain their position and path even if the emitter is moved, they have their own world coordinates.
- **Billboard:** If **Particle** is selected, the particles maintain a frontal position against the camera when the camera moves.
- **Sort:** Enables/disables sorting.
- **Array:** Enables/disables the use of array.

Copyright © 2026 Vizrt

- **Reset Particles:** Removes all already emitted particles.
- **Show Forces:** Show help lines that show the forced settings of some of the parameters.
- **Freeze Motion:** Halts the emitting process.

Spawn

Spawn parameters are related to the position of the smoke emitter. In the Spawn panel define the shape and size of the area in which the emitter should emit the particles.

- **Type:** Defines the shape or type of spawn area. Click **Cube**, **Sphere**, **Container** and **Image** to set if the spawn area should be shaped as a **Cube** or a **Sphere**, or if a container or image should be used.
- **Size:** Sets the size, selected from the **X**, **Y** and **Z** values. If **Sphere** is selected and only the **X** value is set, the emitter creates particles while it moves randomly along a line. If a **Y** value is also set, the emitter moves within a circle. If a **Z** value is set, the emitter moves within a sphere. If **Cube** is selected the same applies, but the stages would then be a line, a rectangle and a Cube.
- **Range:** Sets the range, within the **Sphere** or **Cube**, the emitter should use to distribute the particles. Default is 100%, where it uses the whole spawn area. If the value is reduced, a section, which starts from the middle of the spawn area where no particles are being emitted, increases correspondingly. If the value is set to `1%`, the emitter only emits particles at the outer edge or surface of the cube or sphere.

Tip: Container is especially valuable when animating a Scene, as the smoke is emitted from the outline of an animated Container. To set the Container, which the smoke should emit from, drag the Container from the Scene Tree onto the drop zone.

Note: A Container must be 2D, such as a rectangle, circle, square, etc.

Impulse

In the Impulse panel set the parameters for how particles are emitted.

- **Speed:** Sets the speed the particles have when they are emitted. In the field to the right of **Speed**, set a percentage over which the particle speed varies.
- **Rotation X:** Sets the X axis angle of the opening in the emitter, the emitter hole.
- **Rotation Z:** Sets the Z axis angle of the opening in the emitter, the emitter hole.
- **Angle X:** Sets the X axis opening angle of the emitter hole. It can be done locked with the Z axis, or independently. Click **Lock X/Y** to The button to the right of the value field to lock or unlock the axes.
- **Angle Z:** Sets the Z axis opening angle of the emitter hole.
- **Twist:** Creates a twist of the opening angels of the emitter hole. It can also be described as a rotation of the emitter around the Y axis.
- **Align to Center:** Disables all the above impulse parameters and aligns the particles above the emitter.

Forces

In the Forces panel are the parameters to create environmental effects, like wind, gravity and air resistance.

- **Gravity:** Sets the degree of gravity, which affects the path of the particles. If set high, the smoke moves downwards. With a negative gravity, the particles rise faster.
- **Air Resistance:** Sets the degree of air resistance to creates a force that prevents the particles from rising.

Copyright © 2026 Vizrt

- **Turbulence X**, **Y** and **Z:** Defines a simulation of turbulence on the particles as they rise. The turbulence effect is achieved through a jittering of the particles. The **X**, **Y** and **Z** values set the axis of the jittering movement.
- **Wind:**
  - **Force:** Sets a level cross wind. A positive value blows from the left to towards the right, a negative the other way around. For example: If a horizontal wind force is not required, use the two next parameters to adjust the angle of the wind.
  - **Rot X**/**Z:** Sets the rotation of the wind force on the X and Z axis. Modify these values to change the direction of the wind defined in the **Wind Force**.

Size

In the Size panel are the parameters to the size of the particles at the moment of their birth and death. Between the time of birth and death, the size is a product of a linear interpolation.

- **Birth X**/**Y:** Sets the size of each particle at the time it is being emitted.
- **Death X**/**Y:** Sets the size of each of the particle at the time it fades away and dies.
- **Lock X/Y:** Locks or unlocks the axes.

Rotation

In the Rotation panel, you can define a static rotated position or a spin of the particles. Both the static rotated position and the spin can either be set as an offset equal on all particles or as a range where the particles are rotated or span randomly. To see the effect, set the number of particles to a very low value and alter the rotation parameters.

- **Initial**
  - **Offset:** Sets the degree of rotation the particles should have. This rotated position is static and remains throughout the life of each particle.
  - **Range:** Sets a degree range of rotation the particles should have. This rotated position is static and remains throughout the life of each particle. If for example the value is set to `30`, the static rotated position of the particles varies randomly between 30° and -30°.
- **Spin**
  - **Offset:** Sets the degree of spin each particle should have throughout its life time. The higher the value, the higher the number of spins. A positive value creates an counterclockwise rotation, a negative creates a clockwise.
  - **Range:** Sets a spin degree range for the particles. They spin randomly between the parameter value and the same value mirrored through zero.
- **Follow Path:** If the RFxSmoke object is animated, enable this option to make the particles align their position to the animation path in the same way as is the case with the follow path option in the transformation editor.

Color

In the Color panel set the color of the particles throughout their life cycle. The particles can be set to change their color through stages from their birth to their death by the addition of more Key Frames (single click the color stage).

The left side of the color stage is at birth time and the right is at death time. By default there are two color Key Frames at each end, that can be clicked on, to set their color. The color range between two Key Frames is an

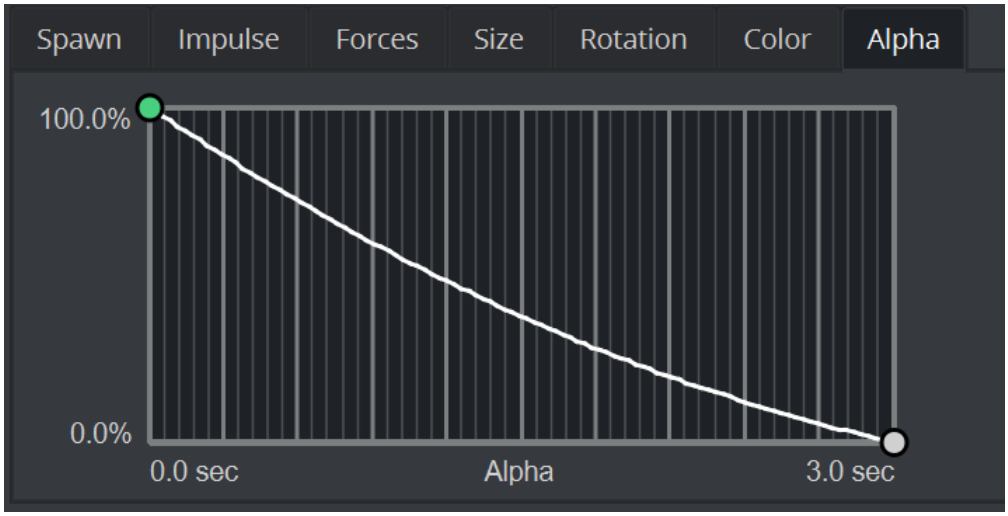
Copyright © 2026 Vizrt

interpolated transformation. More Key Frames can be added to create a more complex color transformation between particles birth and death.

To delete color Key Frames, select them and press **DELETE** or **BACKSPACE**, on the keyboard.

Alpha

In the **Alpha** panel set the alpha values of the particles throughout their life cycle. The alpha editor contains a curve with points that can be moved, in both X and Y directions, to change the form of the curve. The X axis is the life time of the particles and the Y axis is the alpha value.



To select a point, or create a new one, left-click on the graph and drag the point to the required X and Y position. To delete a point, select it and press **DELETE** on the keyboard.

See Also

- **Create Animations** section of the Viz Artist User Guide.

Copyright © 2026 Vizrt

2.3.6 Ticker Geometry Plug-ins

The following Geometry plug-ins are located in the Ticker folder:

- Dexter
- Scroller
- Trio Scroll

Copyright © 2026 Vizrt

Dexter



The Dexter plug-in is a macro-based text-ticker tool that generates static horizontal text crawl and vertical text rolls.

The Dexter editor is organized in five sections, Control, Design, Text, Trigger and Stopper. When adding Dexter, two sub-containers are also added; Animation (visible) and Templates (hidden). In addition, it has a set of scripting possibilities.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

This page contains the following topics and procedures:

- Common Parameters
- Control Parameters
- Design Parameters
- Text Parameters
- Trigger Parameters
- Stopper Parameters
- Dexter Parameters
- Script Programming Parameters
- Script Programming Tokens
- Script Programming Syntax
- Script Programming Templates
- Script Programming Newline Logic
- Script Programming Triggers

Common Parameters

These buttons are found top right of the Dexter editor and are not active in control mode.

- **Control:** See Control Parameters.
- **Design:** See Design Parameters.
- **Text:** See Text Parameters.
- **Trigger:** See Trigger Parameters.
- **Stopper:** See Stopper Parameters.
- **Info:** Shows the delimited borders defined by the mark settings.
- **Update:** Applies the changes on the already existing containers.
- **Build:** Rebuilds all objects.
- **Clear:** Clears the text objects.

Control Parameters

Use the Control section to start, stop, continue and reset the scroll/roll. It also contains buttons to select the velocity mode and speed of the ticker.

- **Start:** Starts the animation.
- **Stop:** Stops the animation.
- **Reset:** Resets the animation.

Copyright © 2026 Vizrt

- **Cont:** Continues the animation.
- **Loop:** Loops the scroller/crawl.

The velocity control defines how the speed of the crawl/roll is to be calculated.

- **Duration (all groups same):** Makes all group use the same amount of time to crawl/scroll from start to end. The duration is measured at the group containing the longest text string.
- **Duration (for each group):** Makes all groups use exactly the same time. As a consequence, longer text strings scroll faster than shorter ones.
- **Speed:** Makes all groups scroll at the same speed, as a consequence long message takes longer to scroll than shorter messages. The speed unit is pixels per frame divided by ten.

Design Parameters

In the Design section of the Dexter editor, the range of parameters determines how the text shows:

- **Screen width:** Sets the width of the screen used for placing the text objects and culling. Possible values range from `20` to `4000`.
- **Screen height:** Sets the height of the screen used for placing the text objects and culling.
- **Loop kerning:** Defines the spacing between two loops. Possible values range from `0` to `1000`.
- **Group kerning:** Sets the vertical distance between group items.
- **Crawl mode:** Creates a normal horizontal scrolling text.
- **Roll mode:** Creates a vertical scroll.
- **Default Font:** Shows the default font. To load a font, drag it from the Server Panel and onto the font drop zone. To unload a font, simply drag it to the trashcan. This must be set to get any text unless you specify a font in the macro and all text are executed with macros.
- **Autosplit text:** Splits long text objects into smaller text objects for better performance when enabled.
- **Corr. val.:** Adds the correction value to the gap between two new objects created when the autosplit feature splits a text object. Although it reads the length of a space character from the font to make a proper placing of the objects, there is a slight difference in length between a split and a non-split text. Therefore, the correction value is inserted to correct text that does not look good because of incorrect placing when the objects are split.
- **Min. object length:** Sets the minimum object length for an auto-split object.
- **Col1:** Sets the default color 1. To set it, either drag a material from the Server Panel onto the col1 icon, or use the color editor below.
- **Col2:** Sets the default color 2. To set it, either drag a material from the Server Panel onto the col1 icon, or use the color editor below.
- **Col3:** Sets the default color 3. To set it, either drag a material from the Server Panel onto the col1 icon, or use the color editor below.

Text Parameters

The Text Editor contains the text area where you enter the script commands. It also contains a list of all tokens and parameters. By clicking on a token or parameter, you automatically insert it into the script text at cursor position.

- **Space=Text:** Forces the parser to interpret spaces at the beginning of a line/after a token has ended as legal text (in its default state those spaces are abandoned).

Copyright © 2026 Vizrt

- **Load File:** Allows you to browse for a script file. The default extension for the file is *.edx*. The script file must contain a macro script with standard syntax. When the file loads, the contents are written onto the text field.

Trigger Parameters

- **Act.:** Evaluates the marker when set to `On`.
- **Mark:** Determines the name of the marker for orientation.
- **Position:** Sets the marker position where an event should occur.
- **Command:** Runs the Viz command at the determined place.

The concept of marks and triggers allows you to send commands to Viz Artist/Engine when the animation reaches a certain position. To make it work you need to define two items:

First of all, mark the trigger positions in the text crawl/roll. This is done using the token `<MARK>`.

Animation starts<MARK A>here.

There are six marks, named A, B, C, D, E and F, that can be put wherever desired in the text. If **Info** is enabled, a representation of the mark is visible on-screen.

In addition to the six user-definable marks, there are two marks that are created automatically, named BEGIN and END. They are located at the beginning and end of the crawl/roll. Please keep in mind that the END mark is located before all the containers created for the loop feature and that marks are not looped.

After setting marks and building the crawl/roll, trigger actions must be specified. All eight marks are listed in the trigger section of the editor. The **on**/**off** switch left to the mark name is used to set that mark active/inactive. The position value right to the mark name sets the position where the triggering should occur (value `0` is at the end of Dexter's bounding box. When animation is running and the mark in the text reaches this position, the specified command is triggered. If you switch on **Info** you see a line representing the trigger position. Switch it on and watch the marks and lines interacting to see how this works. Right to the trigger position you find a text field where you can enter the command which should be sent to Viz Artist when a mark triggers. You may use every command available via external control, for example you may change values in Viz Artist, start animations, execute plug-ins, and so on.

First you may issue two or more commands when a mark reaches its trigger position. Simply split them using a `;` character (semi-colon). Also you may use the DEXTER location to send pseudo-commands to Dexter.

Currently two such pseudo-commands are implemented: The first one is DEXTER*STOP, which stops Dexter's animation, the second one is DEXTER*PAUSE X, where Dexter stops the animation for X seconds (of course this is a floating point value, so DEXTER*PAUSE 0.5 is also possible) and continue with the same speed as before.

> **Note:** These two pseudo-commands are not supported by Viz Artist command interface; they are only available using Dexter's trigger feature. Dexter supports the THIS_SCENE pseudo location as well; you may want to use this to refer to objects/animations in the local scene.

Stopper Parameters

- **Enable:** Evaluates the markers when set to `On`.
- **Use Mark:** Uses the given position.
- **Position:** Sets the position of the `Ease-Out` and `Ease-In` range.
- **Ease Out:** Determines the last position of a text object during movement.
- **Continue**

Copyright © 2026 Vizrt

- **Wait for "Cont":** Makes the text object wait until an `continue` Viz command is sent.
- **Wait Nr Fields:** Makes the text object wait for a specified number of fields. Consider that two fields comply one frame.

- **Ease In:** Slows down the text object and waits for a continue signal or until the number of fields are reached.
- **Global:** Stops global or local to a certain group.

The Stopper is an extension to the triggering ability of Dexter. It allows you to create ease in/out animations at defined points along the crawl/roll. First you need to create a set of marks using the <MARK> token to define the stop points in your text. Then open the Stopper page in the Dexter editor and check the **Enable** button.

Select which marks to use. If you have defined your stop points using the <MARK C> token you need to select C here. If you enable **Info** you see a representation of the animation curve in the editor. Using the position, ease in and ease out slider you can move and alter that animation like you need. (Please keep in mind that the ease out part of the animation is executed before the ease in part: although usually it is the other way round. But here we do not want to start an animation smoothly and then stop it, we want to stop it first and then restart it again).

Using the continue switch you can select what Dexter should do after the animation has stopped: Either wait a certain number of fields or wait until the Continue button in the Control section is pressed. If you have more than one group in crawl mode you may want to check the **Global** as well. If it is checked, all groups halt as soon as a marker in any group reaches the stopper position. If it is unchecked only the group that contains this marker is stopped.

Dexter Parameters

Name	Type	Function
text	string	This is the script Dexter uses for building the text.
screenw	float	Screen width (used for placing the text objects and culling) 20/4000.
screenh	float	Screen height (used for placing the text objects and culling) 20/4000.
time	float	Animation length (in seconds) or animation speed 1/500.
file	string	File name (default extension .*).
loopmode	bool	Loop on or off.
loopkern	float	Defines the spacing between two loops (global) 0/1000.
groupkern	float	Defines the spacing between two groups (global) 0/1000.
showarea	bool	View the culling area on or off.
doautosplit	bool	If on, it splits text objects into pieces.

Copyright © 2026 Vizrt

Name	Type	Function
autosplitcorr	float	Correction value for autosplit feature -1e6/+1e6.
autosplitmin	int	Minimum number of characters for split text objects 0/100000.
time-line	float	Value for preview control (works if animation is stopped) 0/1000.
deffont	string	Default font (for example, Peak/AvantGarde/AvantGarde-Book).
col1	unsigned long	Default color value.
col2	unsigned long	Color value #2.
col3	unsigned long	Color value #3.
durmode	int	Selects animation mode (duration/speed) 0/2.
roll	bool	Switches between vertical (0) and horizontal (1) animation.
trigonbegin	bool	Turns on/off trigger BEGIN (see 2.6).
trigona	bool	Turns on/off trigger A.
trigonb	bool	Turns on/off trigger B.
trigonc	bool	Turns on/off trigger C.
trigond	bool	Turns on/off trigger D.
trigone	bool	Turns on/off trigger E.
trigonf	bool	Turns on/off trigger F.
trigonend	bool	Turns on/off trigger END.
trigposbegin	float	Position of trigger BEGIN -100000/100000.
trigposa	float	Position of trigger A -100000/100000.
trigposb	float	Position of trigger B -100000/100000.

Copyright © 2026 Vizrt

Name	Type	Function
trigposc	float	Position of trigger C -100000/100000.
trigposd	float	Position of trigger D -100000/100000.
trigpose	float	Position of trigger E -100000/100000.
trigposf	float	Position of trigger F -100000/100000.
trigposend	float	Position of trigger END -100000/100000.
trigcombegin	string	Command for trigger BEGIN.
trigcoma	string	Command for trigger A.
trigcomb	string	Command for trigger B.
trigcomc	string	Command for trigger C.
trigcomd	string	Command for trigger D.
trigcome	string	Command for trigger E.
trigcomf	string	Command for trigger F.
trigcomend	string	Command for trigger END.
status	bool	While animation is running, this parameter is set to 1 (read-only).

Boolean values can either be 1 (on) or 0 (off). Color values are four byte unsigned integers, byte 0 = red, 1 = green, 2 = blue, 3 = alpha.

Script Programming Parameters

Name	Default Value	Function
AnchorX, AnchorY, AnchorZ	0	X, Y and Z position of a new text object.
AnimDirection	Default	Animation direction, set to anything but “Default” to reverse animation

Copyright © 2026 Vizrt

Name	Default Value	Function
AutoSplitCorr	0	Correction value for autosplit feature
AutoSplitMin	20	Minimum number of characters for split text objects
Col1, Col2, Col3	255/255/255/255	Holds the color value from the gui/external control (parameter Col1)
Command	-	Sends a command to Viz Artist/Engine.
Detail	Auto	Sets detail/fontstyle of text objects (see above).
DoAutoSplit	0	Enables/disables auto-split feature.
Enlighted	0	Set to 1 if you want enlighted text.
FontStyle	-	Sets the font used to create text objects (for example, Vizrt/AvantGarde/AvantGarde-Book).
GroupKerning	-	Sets horizontal spacing between groups (like Jump).
Jump	0	Sets horizontal spacing between groups.
LastContainer	-	Holds the container number of the last created container.
LastX, LastY	0	If a new text object is created, the last X and Y position are stored here.
MarkOffsX, MarkOffsY	0	X and Y offset for marks (see Script Programming Tokens).
Material	255/255/255/255	The material used for creating text objects.
ObjectKerning	5	Sets spacing between two text objects.
RollAlign	C	Align groups in roll mode. values are R/r (right), L/l (left) and C/c (center).
RollBorder	0	Sets left/right border for groups (if aligned left or right in roll mode).
Scaling	20	Sets scaling in percent (20 means a scaling of 0.2).

Copyright © 2026 Vizrt

Name	Default Value	Function
ShadowDirection	320	Direction of shadow.
ShadowDistance	10	Distance of shadow
ShadowMaterial	204/204/204/204	Material of shadow.
ShadowSharpen	0	Controls **Sharpen** parameter for the shadow of the selected font.
ShadowZOffset	-1	Z offset of shadow.
Sharpen	0	Control **Sharpen** parameter of selected font.
SoftShadowLevel	1	Level of soft shadow (1-4).
Step	15	Sets the amount AnchorX is increased on a newline (see newline logic).
Template	-	Sets template to be used for creating text objects (see Script Programming: Templates).
TextKerning	0	Adjusts text kerning (in text object).
UseShadow	0	Set to 1 to enable shadow.
UseSoftShadow	0	Set to 1 to enable soft shadow.
WordSpacing	0	Adjusts word spacing (in text object).
XScaling, YScaling, ZScaling	20	Scaling value for X, Y and Z axes.

Color values are described in the form r/g/b/a with decimal numbers (base 10). Col1 is also used as the default color value (Material is initialized with this value). Material can also hold an existing material in the material pool (for example, *Vizrt/Artdeco/artdeco.0*). See **Col1, Col2, Col3** in Script Programming Parameters.

Detail can have one of the following values: 1, 2, 3, 4, 5, 6 or T. 1 through 6 are detail levels, T means texture font. Every other value (like the default “Auto” is interpreted as detail level auto). See **Detail** in Script Programming Parameters.

Copyright © 2026 Vizrt

Script Programming Tokens

Name	Syntax	Function
ADD	<ADD Parameter Value>	Adds Value to Parameter (see example below).
CLEAR	<CLEAR Parameter>	Clears Parameter (sets it to "").
CONTAINER	<CONTAINER Path>	Inserts a container into the text (see example below).
ENDMAC	<ENDMAC>	Ends macro definition (nested macros are not allowed).
EXEC	<EXEC Name>	Executes the macro Name.
GROUP	<GROUP>	Begins a new group of containers.
IMAGE	<IMAGE name>	Inserts an image in the text.
MACRO	<MACRO Name>	Defines the macro Name.
MARK	<MARK Type>	Sets a mark (see Script Programming Triggers).
MAX	<MAX Parameter Value>	If Value is higher than Parameter then Parameter = Value.
MIN	<MIN Parameter Value>	Works like MAX but takes the smaller value.
NEWLINE	<NEWLINE>	See Script Programming Newline Logic .
NOTEXT	<NOTEXT>	See Script Programming Newline Logic.
SET	<SET Parameter Value>	Sets Parameter to Value.

The ADD token adds either a string or a float to one. If either Parameter or Value are strings, the two strings can be combined to one. If both are floats, an arithmetic addition is done.

For example: `<SET Test Vi><ADD Test zrt>` results in “Vizrt”, but: `<SET Test 5><ADD Test 6.3>` results in Test holding the value 11.3.

The CONTAINER token is used to define a container path for inserting a container into the text. A container path may either look like “1/3/2/3/1” (the same system as used in the GUI), “$Containername” or “#Containernumber”.

Copyright © 2026 Vizrt

`<CONTAINER 1/3/2/4> <CONTAINER $$Sphere> (or: <CONTAINER “$Sphere”>) <CONTAINER #307>`

The MARK token sets a mark that can be A, B, C, D, E or F.

The GROUP token begins a new group of containers. Instead of writing <GROUP> you may leave one line empty as this has the same effect. Use groups for making lines in roll mode.

Script Programming Syntax

The syntax consists of tokens and text. <tokens> are surrounded by <> (for example, <ENDMAC>). Spaces are used to separate tokens and arguments. Text is everything that is not bracketed. Tokens can be everywhere in the text (for example, Viz is <SET Material 234/34/58/114>great is valid). You do not need to write every token in a new line. Nested tokens are not allowed. A macro cannot execute itself.

In addition to the tokens, there is the escape character **$:**

- **$Parameter** inserts the value of Parameter in the text
- **$_** inserts a space character in the text (for use in tokens)
- **$<** inserts a < character
- **$>** inserts a > character
- **$$** inserts a $ character

The $ character can be placed in tokens and in the text, for example $<Dexter$> says: `<SET Text Hello><ADD Text $_World>$Text`

Inside tokens you may use double quote characters to insert text.
`<SET Text “This is a text”>`

Any $ characters and spaces within two double quote characters are ignored from parsing and executing. Two double quote characters are interpreted as one:
`<SET Name ""Harry"">`

Script Programming Templates

You can define templates for the creation of text objects. Under the Dexter container you find a container named “Templates”. Under this container you can create a set of template containers. These can hold animations, plug-ins, key, alpha, etc. Give every template container a unique name. To access it in the script, you need to set the Template parameter to the name of the template you want to use: `<SET Template AlphaKey1>`

After you set the Template parameter, Dexter uses this template container as basis for every text object. To switch this behavior off set Template to its default value (which is ""): `<CLEAR Template>`

Caution: Certain parameters (like material) are overwritten by Dexter after the container has been created from the template.

Script Programming Newline Logic

The token <NEWLINE> (see Script Programming Tokens) describes a new text object, which is separated from the previous one. Every newline moves AnchorX (see **AnchorX, AnchorY, AnchorZ** in Script Programming Parameters) a

Copyright © 2026 Vizrt

bit further, depending on the value of **Step**. This token is inserted internally to prevent you from typing <NEWLINE> over and over again.

The following text `One Two` is converted to `One <NEWLINE> Two`

No newlines:

- From the beginning of the script until the first occurrence of text.
- After a line that doesn't contain text (if the line is empty a new group is created).
- After the tokens MACRO, ENDMAC and NOTEXT, and the last newline before those tokens are deleted.

For example:

Lines	Explanation
	No new lines because no text occurred.
<SET Camera Bobby>	No text, no new line.
Return of the	Here is the first new line.
Killer Tomatoes II	No new line because of token MACRO in the next line. From here on there are no new lines until the end of the MACRO.
<MACRO SetDir>	No new line because there is no text in the line.
<SET Director Jimmy>	
<ENDMAC>	
<EXEC SetDir>	
Director:	New line.
$Director	New line.
<SET FontStyle Arial>	No new line because there is no text in the line.
Camera:	New line.
$Camera	
<NOTEXT>	The <NOTEXT> token does not add new lines before the next text item occurs (see Script Programming: Tokens).
x	New line.

Copyright © 2026 Vizrt

Lines	Explanation
y	New line.
<SET Material $Col2>	No new line because there is no text in the line.
z	New line.
1<NEWLINE>2	New line for 1 and 2.

The macro SetDir in the example above can be used as follows without creating a newline: `Hello<EXEC SetDir>World`

The same rules applies to automatically inserted **GROUP** tokens (see Script Programming Tokens).

Script Programming Triggers

The concept of marks and triggers allows you to send commands to Viz Artist/Engine when the animation reaches a certain position. Therefore you need to define two items:

- First is setting a mark in the text. You have six different marks to place wherever you think they are useful, together with the automatically created BEGIN and END mark this gives you a total of eight marks.
- The second items are the triggers, where you have as well eight different triggers. While marks are set with a token in the text, triggers can be defined in the trigger section of the Dexter editor or using external control. You can define a position and a command for every trigger. When the corresponding mark reaches the trigger position, the command is issued to Viz Artist/Engine.

Note: Only marks and triggers of the same type work together! If for example mark B reaches the position of trigger D, nothing happens.

Marks are set using the MARK token, for example: Animation starts exactly{{<MARK D>}}here!

User defined marks are numbered A through F and are color coded in the trigger editor and as graphical symbols when show area is turned on. In addition to the user defined marks there are the automatically created marks BEGIN and END, which are inserted at the beginning and ending of the crawl/roll. The commands being issued do not have to be numbered, this is optional. If you do not provide a number, -1 is added before the command is sent to Viz Artist/Engine. you can as well define more than one command, separate them by a semicolon character.

You may invoke two internal commands using the trigger function with the `DEXTER*` location: stop and pause.

```
DEXTER*STOP DEXTER*PAUSE X
```

Note: `X` is the time in seconds.

Copyright © 2026 Vizrt

Scroller



The Scroller plug-in generates a dynamic line of scrolling items, mostly used for creating scrolling tickers.

When used with Viz Ticker the scene is designed as a transition logic scene where each data item is an instance of a scene template designed as part of the background scene. This is unlike traditional transition logic scenes where each data item would be based on a specific foreground scene controlled by the background scene. Based on the scene template that resides with the background scene the Scroller is able to construct and configure the scene templates on demand for Viz Ticker that manages and reuses them.

To design a scrolling carousel, the Scroller and one or more scene templates must be added to the background scene. The scene templates serve as prototypes for the containers that the Scroller plug-in generates. Alignment of the generated containers is done by editing the alignment of the scene templates.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> Ticker

Note: Please note that this plug-in requires special security privileges. When running as Non-Admin, `SeCreateGlobalPrivilege` and `SeCreatePagefilePrivilege` have to be set.

Tip: When saving a Scroller scene, remember to hide the scene template containers. This prevents the scene template from being shown on-screen.

Scroller Properties

- **Element Source:** Element descriptions are added here. Descriptions must be entered in the element source syntax, or in the name of a Shared Memory segment that uses the Shared Memory protocol. When designing and testing a scene, the element source syntax is used to make the system scroll instances of the design containers.
- **Show Advanced:** Shows additional parameters. Defaults to `Off`.
- **Layer:** Defines the layer name for the scroller. The layer name must be identical to the name of the director that holds the in and out stop points for the carousel.
- **Mode:** Defines if elements should be shown as scrolling or flipping elements for scrolling and flipping carousels, respectively.
- **Direction:** Defines the direction of the scroller.
- **Scroll Region Width:** Defines the width of the scroller. This option is only relevant for left-to-right/right-to-left scrollers.
- **Scroll Region Height:** Defines the height of the scroller. This option is only relevant for up-down/down-up scrollers.
- **Scroll Speed:** Defines the speed of the scroller. The x position of the scrolling items are decremented with this value for each frame that passes.
- **Minimum spacing:** The minimum spacing between scrolling elements.
- **Padding:** Defines an extra padding value that is added to the width of the bounding box of an item when calculating the spacing between the items. Negative padding can be used to make items overlap.

Copyright © 2026 Vizrt

- **Start Position:** Set the initial position as a percentage value in the range `0-100`. A value of `100` means the scroller starts fully populated with content. The default value is `0`.
- **Played Out:** Tracks movement of scrolling element and notify ticker service on specified position.
  - **Element:** Position of the scrolling element to be checked with the marker. When set to `Beginning hits`, notify when beginning of element hits the marker. When set to `End leaves`, notify when end of element leaves the marker.
  - **Marker:** Position of the marker related to side of Scroll region.
- **Show Bounds:** If enabled, the area in which the scroller tries to maintain the illusion of a continuously scrolling flow of items is visible on-screen.
- **Active:** Scrolling or flipping is available when enabled.
- **Preload Cache:** Use this on configurations with no ringbuffer, to move all spike-load related to item creation to the initialization phase. Designs named `DESIGN_designname_Xnnn` have `nnn` instances created in advance, where `nnn` is an integer.
- **Trigger Director:** Director to be triggered when the **Trigger** button is pressed.
- **Start with padding:** Sets padding for the first scrolling item, for example when **Start Position** is set to `100%` or the scroller container is hidden at the beginning.
- **No Padding On Empty:** Does not create any extra spacing between elements for elements with zero bounding box.
- **Reinitialize:** Reinitializes the plug-in by deleting all cached objects and reconstructing the initial node structure. This parameter can be used to update the scroller after changing for example the item source.
- **Clear:** Clears all current ticker items in the rendering window.
- **Trigger:** Triggers the director named in the Trigger Director field provided that there are items in queue to be scrolled in. If no director is specified, the scroller is activated.

Tip: The Viz Ticker User Guide contains information on design conventions when using the Scroller plug-in.

See Also

- Viz Ticker User Guide
- Basic Container Control plug-ins
- Toggle plug-in

Copyright © 2026 Vizrt

Trio Scroll



The Trio Scroll plug-in is a Geometry plug-in for positioning and scrolling a fixed set of items in a specified direction, often used for credit lists or crawls.

Scrolling templates can be built in Viz Trio using the Create New Scroll feature.

The scroll scene that is automatically created by Viz Trio can be edited in Viz Artist, but it is also possible to build them manually.

Some of the supported features are:

- Scrolls can be created by receiving item data from Viz Trio's XML format.
- Fine-tuned control of spacing between individual items.
- Easepoints (ease in and out) on particular items.

Scrolls are usually built in Viz Trio, but it is possible to add items in Viz Artist by adding merged objects under the *base_elements* group and clicking the **Initialize** button.

Adding the Trio Scroll plug-in to the scene tree automatically adds the Control Object plug-in (if it is not present).

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Default

Trio Scroll Properties

- **Field Identifier:** When making templates for Viz Trio, this should be a numeric value, Range: 1-n. For Viz Pilot and Viz Ticker, this can also be a descriptive text string. This value is used by the control clients to identify the editable item in the scene. In Viz Trio, the numbers from the control plug-ins are used to create the tab-order between the editable items.
- **Description:** A description of the tab field or editable object. This is used as a description for the items when used in the control clients.
- **Show only in Expert Mode:** If Viz Trio is connected, changes are possible if this toggle is set to Off.
- **Mode:** Selects the direction the scroller should move across the screen.
- **Scroll area width:** Selects the area in width the scroller should use.
- **Scroll area height:** Selects the area in height the scroller should use.
- **Scroll speed:** Sets the scroller speed.

> **Note:** To avoid flickering, a roll rate that is aligned with your output frequency should be used. When the roll rate is twice the field rate, things look fine. This relationship holds for all integer multiples: Roll rates that are odd multiples of the field rate look awful. Even multiples look great. For 59.94 Hz video, good roll rates are 120, 240, 360, and the like. In 50 Hz, the good ones are 100, 200, 300, 400, and so on.

- **Wanted total scroll time:** Sets the scroll time.
- **Actual total scroll time:** Sets the scroll time.
- **Global spacing:** Sets spacing between the pages.
- **Loop:** Enables looping. When set to `Off`, looping does not stop the playlist from being played until the end.
- **Show bounds:** Enables and shows the scroll's bounding-box on the preview and program renderer.
- **Auto-start scroll:** Enables the Auto-scroll on take property.

Copyright © 2026 Vizrt

- **Start position:** Sets the start position for the scroll. This setting affects the total scroll time.
- **Alignment:** Sets the alignment of the scroll. The position is relative to the position of the scene design. Available options are: None, Left/Bottom, Center and Right/Top.
- **End-of-scroll director:** Triggers a Continue on a specified end director. The director is triggered when the scroll leaves the scroll area. It is only supported in normal mode (real-time), not in post rendering or NLE mode.
- **Show Advanced:** All parameters under the Advanced section are for debugging, by Vizrt developers or very advanced users only. For more information contact Vizrt support.
  The nameless text parameter with the value *<entry></entry>* shows the contents of the scroll items in the internal VDOM format. It contains the scroll items that can dragged into a scroll page in the Trio client with drag and drop
  - **Current position:** Scrubs the scroller position.
  - **Command:** Executes various scroller commands. After setting its value the click on **Execute Command**. If a command has a result it is reflected in *Command result*. The following command examples are used from the Trio client:
    - `SET_FOCUS [<item name> [<tab-field name>]]` - This is for example used if you double click in Trio on a scroll item to highlight it in the preview.
    - `MOVE_ELEMENT <from item name> <to item name>` - This is used when changing the order of scroll items by drag and drop.
    - `REMOVE_ELEMENT <item name>` - This is used to remove a scroll item.
    - `UPDATE_TOTAL_TIME` - This commands the plug-in to recalculate its total time, typically after adding a new scroll item.
    - `SET_TEMP_EASEPOINT_VALUE` - Gets used when editing the Easepoint values.
  - **Command result:** See **Command**.
  - **Reload prototypes:** Updates the scroll items due to changes in their templates (scenes) - use a context menu action in Viz Trio to do this (see the Viz Trio User Guide).
  - **Start/Stop/Continue:** Controls the scroll (used in the local preview and the live controls for playback).
  - **Execute command:** See **Command**.

See Also

- **Create New Scroll** in the Viz Trio User Guide.
- Control Object plug-in
- Trio Scroll Element plug-in

Copyright © 2026 Vizrt

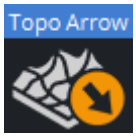
2.3.7 Topology Geometry Plug-ins

The following Geometry plug-ins are located in the Topo folder:

- Topo Arrow
- Topo Rect
- Topo Ring
- Topography

Copyright © 2026 Vizrt

Topo Arrow



The Topo Arrow plug-in creates an arrow geometry. The arrow behaves like it is cut out from cloth and smoothed over the geometry provided by Topography.

This is needed if the arrow is used on a surface which is not 100% planar to prevent parts of the geometry from appearing to be floating above the ground. Ski jumps and golf greens are examples where this should be used.

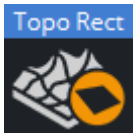
Note: This plug-in is located in: Plugins -> Geom plug-ins -> Topo

Topo Arrow Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Point 1 x:** Determines x-coordinate of starting tip of arrow.
- **Point 1 y:** Determines y-coordinate of starting tip of arrow.
- **Point 1 z:** Determines z-coordinate of starting tip of arrow.
- **Point 2 x:** Determines x-coordinate of ending tip of arrow.
- **Point 2 y:** Determines y-coordinate of ending tip of arrow.
- **Point 2 z:** Determines z-coordinate of ending tip of arrow.
- **Width:** Determines width of the arrow beam.
- **Style 1:** Sets start tip to flat or arrowhead.
- **Style 2:** Sets end tip to flat or arrowhead.
- **Arrow Width:** Determines width of arrowheads. This has no affect on flat ends.
- **Arrow Length:** Determines length of arrowheads. This has no affect on flat ends.
- **Percent:** Sets total arrow-length (tip to tip) to a percentage of length defined by points 1 and point 2. If less than 100%, point 2 is moved accordingly.
- **Mode:** Sets plane in which the arrow is created.
    - **XY:** Creates arrow in XY plane.
    - **XZ:** Creates arrow in XZ plane.
    - **YZ:** Creates arrow in YZ plane.
- **Alignment:** Projects the geometry of the arrow along the y-axis on the geometry provided by a topography plug-in in the same scene if set to **Topography**. This only makes sense for mode XZ.

Copyright © 2026 Vizrt

Topo Rect



The Topo Rect plug-in creates a 3D rectangle geometry. The rectangle behaves like it is cut out from cloth and smoothed over the geometry provided by Topography.

This is needed if the rectangle should be used on a surface which is not 100% planar to prevent parts of the geometry appearing to be floating above the ground. Ski jumps and golf greens are examples where this should be used.

The tessellation needs to be set according to the detail in the topography to eliminate surfaces cutting to the topography or floating above it.

> **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Topo

Topo Rect 3D Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Width:** Determines the width of the rectangle.
- **Height:** Determines the height of the rectangle.
- **Offset:** Offsets each individual vertex from the topography.
- **Tessellation:** Determines the amount of vertices used for the rectangular surface.
- **Plane**
    - **XY:** Creates rectangle in XY plane.
    - **XZ:** Creates rectangle in XZ plane.
    - **YZ:** Creates rectangle in YZ plane.

Copyright © 2026 Vizrt

Topo Ring



The Topo Ring plug-in creates a 3D ring geometry. The ring behaves like it is cut out from cloth and smoothed over the geometry provided by Topography. This is needed if the ring should be used on a surface which is not 100% planar to prevent parts of the geometry from appearing to be floating above the ground. Ski jumps and golf greens are examples where this should be used.

The tessellation needs to be set according to the detail in the topography to eliminate surfaces cutting to the topography or floating above it.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Topo

Topo Ring Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **X:** Determines the X coordinate of the ring's center.
- **Y:** Determines the Y coordinate of the ring's center.
- **Z:** Determines the Z coordinate of the ring's center.
- **Inner Radius:** Sets the inner radius of the ring. If set to `0`, this becomes a disc.
- **Outer Radius:** Sets the outer radius of the ring. The *Outer Radius* must be larger than the *Inner Radius*.
- **Plane**
  - **XZ:** Creates a ring in the XZ plane.
  - **XY:** Creates a ring in the XY plane.
  - **YZ:** Creates a ring in the YZ plane.
- **Segments:** Determines the number of segments along the ring.
- **Slice:** Opens the ring when set to less than `360`.
- **Rotation:** Rotates the ring around the center point.
- **Tessellation:** Determines the amount of vertices used for the ring surface.

Copyright © 2026 Vizrt

Topography



The Topography plug-in is the user interface for the topology in a scene. It is used to either load or build a topology from scratch which can be used by topology aware plug-ins like Topo Arrow, Topo Rect or Topo Ring.

Areas where these plug-ins can be of use are for example ski jumps, golf greens or other surfaces which are not 100% planar. Adding virtual elements which should not float above the ground require the knowledge of the terrain or topographical relief.

Geometry plug-ins can make use of this knowledge and act like a cloth where the deformation along the vertical axis is defined by gravity and the topographic relief.

> ⚠ **Note:** This plug-in is located in: Plugins -> Geom plug-ins -> Topo

Topography Properties

- **Use LOD:** Enables/disables dynamic level of detail.
- **Files** Shows a list of topography files.
- **Name:** Shows the current file name.
- **Load:** Loads the topography from the currently selected file.
- **Save:** Saves the topography to the current file name.
- **VertexX:** Determines the X coordinate of a vertex to add to the topology.
- **VertexY:** Determines the Y coordinate of a vertex to add to the topology.
- **VertexZ:** Determines the Z coordinate of a vertex to add to the topology.
- **NormalX:** Determines the X component of the normal vector for the vertex.
- **NormalY:** Determines the Y component of the normal vector for the vertex.
- **NormalZ:** Determines the Z component of the normal vector for the vertex.
- **AddVertex:** Adds a new vertex to the topology.
- **iA:** Determines the first index of a vertex of a triangular face.
- **iB:** Determines the second index of a vertex of a triangular face.
- **iC:** Determines the third index of a vertex of a triangular face.
- **AddFace:** Adds a new triangular face to the topology using *ia*, *ib* and *ic*.
- **Build:** Attempts to automatically triangulate the vertices which have been added to the topology.
- **Clear:** Deletes all vertices and faces from the topology.

Copyright © 2026 Vizrt

2.4 Container Plug-ins

The default path for Container plug-ins is: *<viz install folder>\plugin*.

- Basic Container Plug-ins
- Arrangement Container Plug-ins
- Control Container Plug-ins
- Default Container Plug-ins
- Feed Container Plug-ins
- Container FX Plug-ins
- Global Container Plug-ins
- MultiTouch Container Plug-ins
- Presenter Container Plug-ins
- RealFX Container Plug-ins
- Script Container Plug-ins
- Sound Container Plug-ins
- SplineFX Container Plug-ins
- TextFX Container Plug-ins
- Texture Container Plug-ins
- Ticker Container Plug-ins
- Time Container Plug-ins
- Tool Container Plug-ins
- Topology Container Plug-ins
- Transformation Container Plug-ins
- Visual Data Tools Container Plug-ins
- Text Container Plugins
- VizMaps Plug-ins

Copyright © 2026 Vizrt

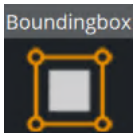
2.4.1 Basic Container Plug-ins

The following Container plug-ins are located in the Container folder:

- BoundingBox
- Cobra
- Coco
- Colin
- Cora
- Corena
- Talent Tracker
- Toggle

Copyright © 2026 Vizrt

BoundingBox



The BoundingBox plug-in allows to override the standard Bounding Box of containers independently of the type of the container.

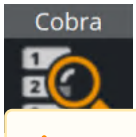
Note: This plug-in is located in: Plugins -> Container plug-ins -> Container

BoundingBox Properties

- **X Left Bottom Back:** Changes the dimension in X-direction starting from left.
- **Left Bottom Back:** Changes the dimension in Y-direction starting from bottom.
- **Left Bottom Back:** Changes the dimension in Z-direction starting from flipside.
- **Right Top Front:** Changes the dimension in X-direction starting from right.
- **Right Top Front:** Changes the dimension in Y-direction starting from top.
- **Right Top Front:** Changes the dimension in Z-direction starting from front.
- **Set to Standard:** Initializes values to enclose the entire container object.

Copyright © 2026 Vizrt

Cobra



The Cobra function is an easy to use function replace (parts) of container names.

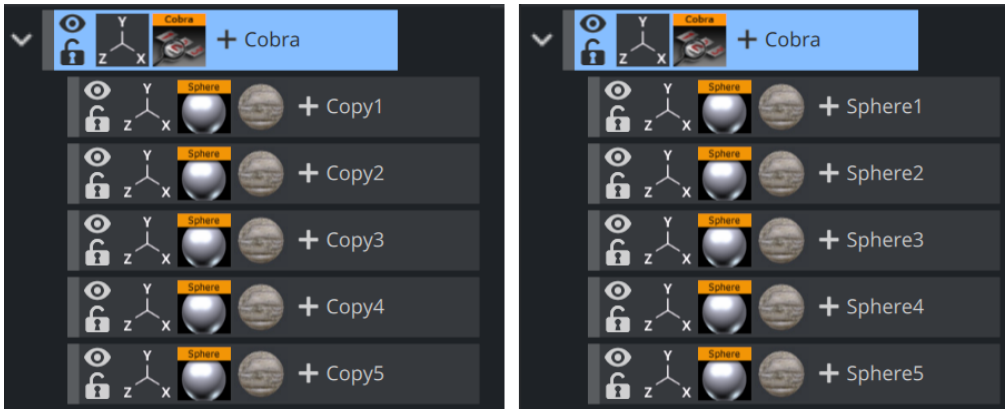
Enter a search pattern and a string the search should be replaced with and execute.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Container

Cobra Properties

- **Search pattern:** Sets the text string to be replaced.
- **Search pattern position:** Defines the position within the string. It can be set to Begin, End and Any.
- **Replace with:** Sets the text string to be used instead of the search pattern.
- **Search level:**
  - **Same:** Affects only containers on the same hierarchy level.
  - **Down:** Affects only Sub-Containers of the current container.
  - **Tree:** Searches the whole Scene Tree.
- **Add Index:** Adds a numerical index to the new name when set to `On`.
  - **Starting Index:** Sets the number to start the index.
  - **Reverse Index:** Uses a reverse index list when set to `On`.
- **Replace:** Starts the replace process.

To Rename Container(s)



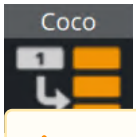
1. Add the Cobra plug-in to a container that, for example, has Sub-Containers that need be renamed.
2. Open the Cobra editor.
3. Enter a **Search pattern** text string (for example, Copy)
4. Set the **Search pattern position** to **Begin**.
5. Enter the **Replace with** text string (for example, Sphere).
6. Set the **Search level** to **Down**.
7. Click **Replace**.

See Also

Copyright © 2026 Vizrt

- Coco
- Corena

Coco



The Coco plug-in allows you to easily create copies of a container.

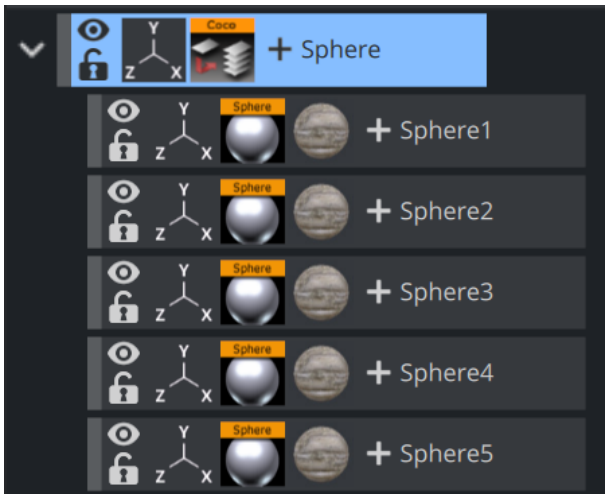
This function is typically used together with the Basic Container Arrange plug-ins.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Container

Coco Properties

- **Number of Copies:** Sets the number of copies to be created when the function is executed.
- **Scene Tree Level:** Allows you to select the hierarchic position of the copies.
  - **same:** Sets the copies on the same level as the source container.
  - **down:** Sets the copies as Sub-Containers to the source container.
- **Clear Transformation:** Sets all transformation values on the new copies, like scaling, position and rotation, back to zero. All copies are stacked at a initial position with all transformation values set to zero.
- **Starting Index:** Allows you to alter the starting point of the index numbering of the copies. By default, the numbering starts on `1`.
- **Container Name:** Allows you to change the name the containers that is given by default.
- **Execute:** After having set the required parameters, click this button to execute the duplicating process.

To Create Copies of Containers



1. Add the Coco plug-in to the container you want to copy.
2. Open the Coco editor and enter the Number of Copies (for example, `10`)
3. Set Scene Tree Level to **down**.
4. Click the **Execute** button.

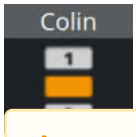
See Also

- Basic Container Arrange

Copyright © 2026 Vizrt

- Cobra

Colin



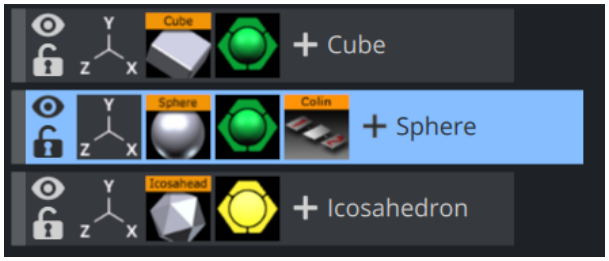
The Colin plug-in allows you to align a container in the scene using two other containers as reference. This makes it easy to maintain a good and controlled symmetry in the scene.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Container

Colin Properties

- **Container 1:** Is the drop zone for the first referential container.
- **Container 2:** Is the drop zone for the second referential container.
- **HOffset %:** Sets the horizontal offset between the referential containers in percent. Horizontal means here an invisible line between the referential containers regardless of the actual boning of the line.
- **VOffset:** Sets the vertical offset from the horizontal line.
- **ZOffset:** Sets the Z offset from the line between the referential containers.
- **Align container rotation**
  - **2D:** Positions and rotates the container in the X- and Y-axis to keep itself aligned on the invisible line between the referential containers as they move when enabled.
  - **3D:** Positions and rotates the container in the X-, Y- and Z-axis to keep itself aligned on the invisible line between the referential containers as they move when enabled.
- **Automatic label orientation:** Adapts the rotation angles automatically. This parameter only works if the **Align container rotation** parameter is switched to **2D** or **3D**.
- **Autoscale object:** Unlocks the **Scaling** value.
- **Scaling:** Scales the container to the provided value.

To Align Containers



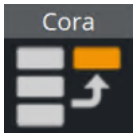
1. Add the Cube, Sphere and Icosahedron geometries to the scene tree.
2. Open the transformation editor for Cube and Icosahedron and set them to Position X `-200.0` and `200.0`, respectively, and Position Y `100.0`.
3. Add the Colin plug-in to the *Sphere* container.
4. Open the Colin editor and drag and drop the Cube and Icosahedron containers onto the Container1 (left) and Container2 (right), respectively. The container is then aligned between the two referential containers.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

Cora



The Cora plug-in allows you to sort a set of Sub-Containers based on multiple criteria. It is a useful tool when your scene starts getting complex and it is difficult to maintain the overview. To make sorts by different criteria it can help you reorganize the scene tree structure. It works on all containers under the container it is added to.

The plug-in is often used together with the Time Displacement plug-in which enables many creative possibilities.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Container

Cora Properties

- **Criteria:** Selects the wanted sort criteria.
  - **Name:** Performs a case sensitive sort. In normal sort order, capitalized letters are listed first. Be aware, use letter case consistently if you want to have a correct result when you perform a sort by name.
  - **X:** Sorts by X-values.
  - **Y:** Sorts by Y-values.
  - **Z:** Sorts by Z-values.
  - **Vis:** Sorts all the visible containers first.
  - **Alpha:** Sorts by the alpha value among those containers who have an alpha function attached. Alpha values set on the material of a container are ignored.
  - **Key:** Sorts containers by key function. Among those with key signal attached, the key alpha value is used for the further sorting.
  - **Rand:** Sorts by random.
- **Mode:** Sets the order of the sorting. Choose between setting the **Highest** or the **Lowest** values.
- **Random Seed:** Is relevant if you have chosen random as your sort criteria. It specifies a seed for the random number generator. Even though Viz Artist use random numbers, the animation for a specific random seed always looks the same.
- **Random Weight:** Defines how random the randomize function sorts. `Random 1.0` means random, `random 0.0` does nothing and `random 0.5` gives you a 50% chance if a container is resorted or stays in place.
- **Execute:** Starts the operation.

To Organize the Scene Tree

Add the Cora plug-in to the container that holds the Sub-Containers you want to sort, select the sort criteria and press the **Execute** button.

See Also

- Time Displacement

Copyright © 2026 Vizrt

Corena



The Corena plug-in renames all Sub-Containers of the container you apply it to.

Corena is often used together with the Time Displacement plug-in which enables many creative possibilities.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Container

Information: This plug-in is not needed anymore and is only available for compatibility reasons. Renaming containers is a built-in function of the Scene Tree.

Corena Properties

- **Name:** Sets the name you want to rename the Sub-Containers to. You can define the number format of the new names by entering a hash in addition to the name for each digit you want in the numbering. If you enter ###newname, the containers are named: *001newname*, *002newname*, and so on.
- **Add Index:** Enables/disables creation of a numbering index.
- **Starting Index:** Sets the number you want the numbering index to start from.
- **Reverse Index:** Creates the index in a reversed order when enabled.
- **Execute:** Starts the operation.

To Rename Container(s)

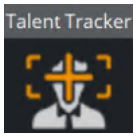
1. Add the Corena plug-in to the container that holds the containers you want to rename.
2. Enter the new name, set the indexing parameters and click **Execute**.

See Also

- Cobra

Copyright © 2026 Vizrt

Talent Tracker



The Talent Tracker plug-in reads the talent position corresponding to a *Person ID* from shared memory and sets the container position according to the talent position.

To use Talent Tracker, drag the Talent Tracking plug-in onto a desired container.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Container

Talent Tracker Properties

- **Person ID:** Sets the container position to the position of a specific talent.

> **Note:** Currently, there is a maximum of one trackable talent, therefore setting the Person ID parameter to anything above 1 is undesired.

The container position is updated according to the position of the talent in the live input.

Talent Tracking

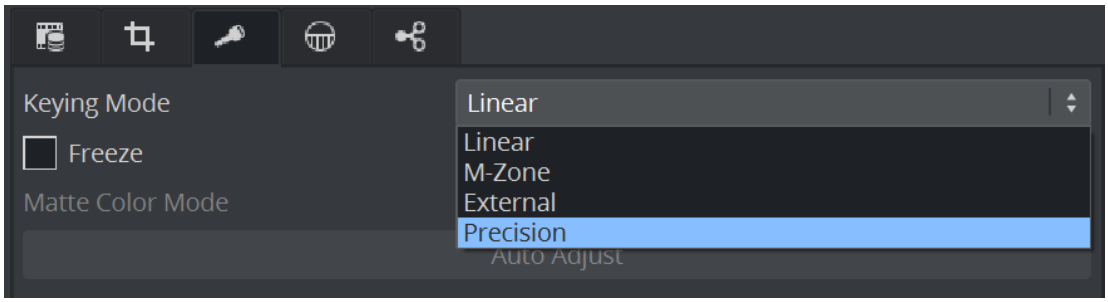
The talent tracking functionality is used to find the talent in live input provided to the Engine. The talent tracking functionality relies on the Precision Keyer, for it analyses the Precision Keyer matte for acquiring the talent position. Currently, a maximum of one person can be tracked.

The positions resulting from the talent tracking analysis is available in shared memory through keys starting with *TRACKING_PERSON*. The value stored along with these keys is a four-dimensional vector of integer values, containing respectively, the *X-coordinate* and *Y-coordinate* with respect to the *width* and *height* of the live input. Such that the format of the value becomes *{x, y, width, height}*.

As the currently supported maximum number of persons tracked is `1`, and shared memory entry besides index `0` should be discarded.

To Enable Talent Tracking

Since the talent tracking functionality relies on the color difference Keyer to be enabled on a provided live input, make sure that the Precision Keyer is enabled on the live input media asset.



The tracking functionality can be enabled by enabling **Talent Reflections** on the media asset.

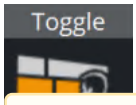
Copyright © 2026 Vizrt



After enabling the talent tracking functionality, the talent positions can be accessed via shared memory, or used directly with Talent Tracker and a container.

Copyright © 2026 Vizrt

Toggle



The Transition Logic engine for Viz. Creates two A/B placeholder containers with cross-fade directors, enabling animated switching between loaded graphics with object caching.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Container

This page contains the following topics and procedures:

- Overview
- Prerequisites
- Description and Methodological Considerations
  - Toggle Placeholder Containers
  - Toggle Directors and Key Frames
  - Director Naming Convention
  - A/B Switching Mechanism
    - Switch Operation
    - Two Trigger Scenarios
  - Parameter Reference
    - Basic
    - Adaptive Scenes
    - Advanced
    - Actions
  - Default Keyframes
  - Object Caching
  - Custom Transitions
  - Interaction with ControlObject
  - Example: Master Scene with Two Layers
  - Typical Broadcast Workflow
  - Technical Notes

Overview

Toggle is the runtime engine behind **Transition Logic**. When added to a container, it:

1. Creates **two placeholder subtrees** (A-side and B-side) as children.
2. Generates **cross-fade directors** (AX/BX) with a default 1-second alpha dissolve.
3. Generates **object directors** (AO/BO) where loaded scene animations are placed.
4. Loads graphics onto the *invisible* placeholder, then **switches** with an animated transition — the viewer never sees a loading state.
5. **Caches** loaded objects for instant re-use.

Key concept — $current and $other The visible placeholder is always named `$current`; the invisible one is `$other`. On switch, the names swap. Content is loaded onto `$other`, data is pushed via ControlObject, then `switch` triggers the cross-fade.

Copyright © 2026 Vizrt

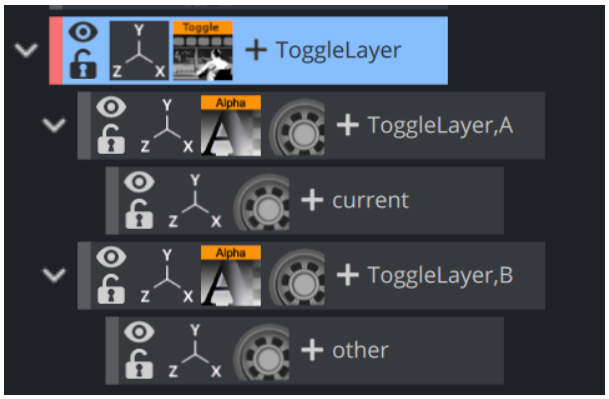
Prerequisites

1. Give the container a **unique, descriptive name** (for example, `BannerLyr`) — this becomes the Toggle layer name and director name.
2. The container must have **no existing children** — Toggle creates its own.
3. Create a **Director** with the **same name** as the container.
4. The Director must have at least two stop points: `$O` (Out, at frame 0) and at least one visible state (for example, `$IN`).
5. Add Toggle to the container.
6. Click **Default Keyframes** to generate the GOTO_TRIO action channels.

Description and Methodological Considerations

Toggle plug-in is one of the main building blocks of Transition Logic, and it is in general a relatively simple plug-in. When the Toggle plug-in is added to a container, it creates two child containers that act as placeholder containers for objects that can toggle on and off screen. In addition, it also creates some utility directors in the stage.

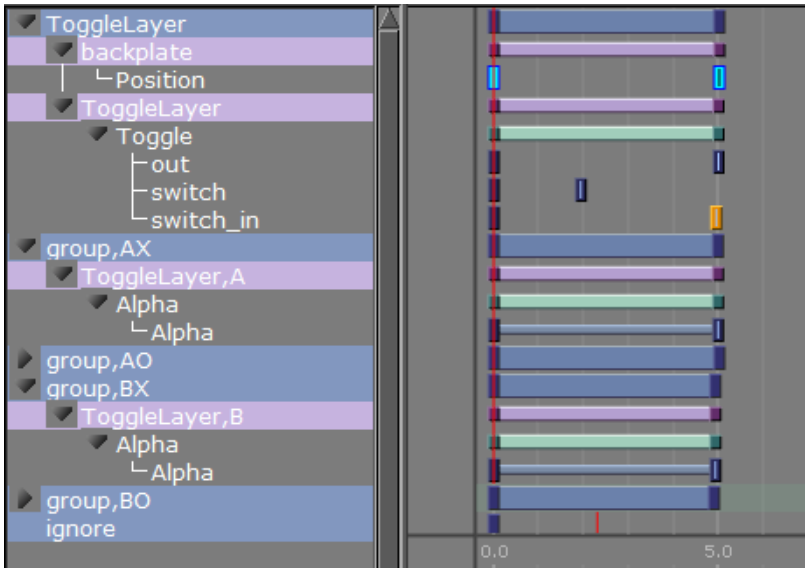
Toggle Placeholder Containers



The two placeholder containers correspond to an **A**-side and a **B**-side. Each of the placeholders again have a child container; one named **current** the other named **other**. The *current* is typically the visible container, while the *other* is not visible. At the time of a switch, these containers switch names. This means that at any given time it is not possible to know if the *current* parent is the *A* side or the *B* side. It is the responsibility of the Toggle plug-in to keep track of this.

Copyright © 2026 Vizrt

Toggle Directors and Key Frames



The Toggle plug-in creates two directors each containing a default one second cross fade animation. These animations are placed in the *AX* and *BX* directors. *AX* can be interpreted as the *A*-side cross fade. The animations can be modified to show any kind of transition, but it is important that the *AX* and *BX* are the same.

In addition to the cross fade directors the Toggle plug-in also creates placeholder directors for the *object* animations. Merged objects that are loaded into these placeholder containers during playback may contain animations embedded in the geometry object. The object animations are automatically added as a sub-director to the *AO* or *BO* director depending on if the object is being loaded on the *A*-side or *B*-side.

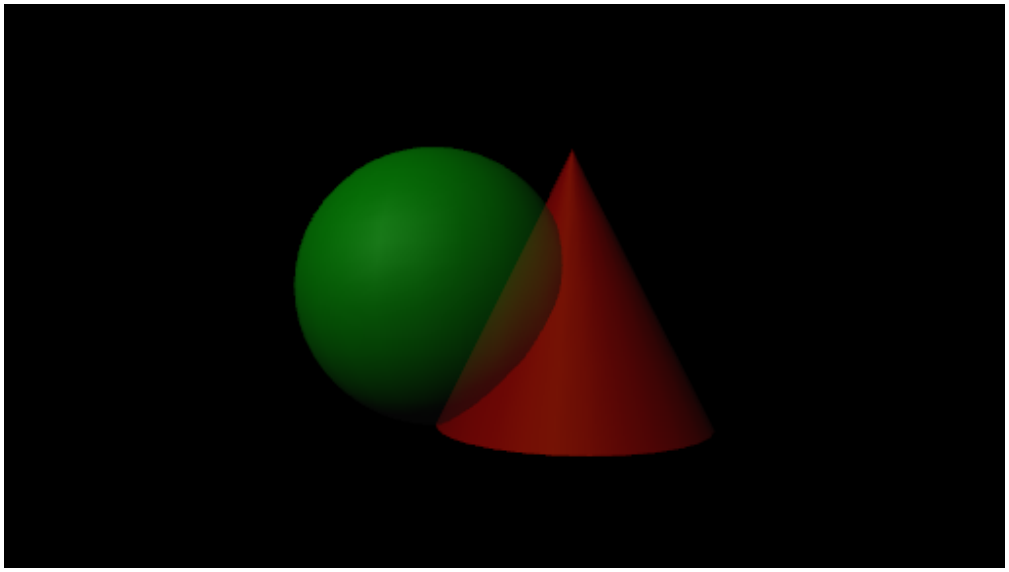
Caution: It is NOT recommended to manually adjust the *AO* and *BO* directors.

There are two different scenarios of when the utility directors are triggered:

1. When there is a state change. When there is a state change, the Toggle-Layer director is animated from the current state to the required next state. In this scenario, the utility directors are triggered according to the toggle Key Frames in the stage. This means that it is the position of these Key Frames that determine the timing of when the utility directors are triggered.
2. No state change, only changing the current object. When there is no state change, there is no movement of the Toggle-Layer director. In this case, the Toggle switch action is triggered, which animates the utility directors at the same time; animating the current object out and the next object in at the same time.

Note: Placeholder containers and directors are only generated if none exists from before.

Copyright © 2026 Vizrt



Toggle has a built-in support for object loading. Enter the path to an object (for example, *GEOM*myfolder/myobject*) into the **Object** field, and the plug-in loads this object on the *other* container. The object is by default invisible and can be loaded with data, before activating the Switch action to show the object.

An object loaded this way is cached, so that requesting the same object again does not trigger loading again. The object cache can also be preloaded by specifying all the geometries that should be preloaded before initializing the Toggle plug-in. Preloading of the objects needed by the Toggle plug-in is done when a playlist is initialized. For this reason the *Preload Object* text field should normally not be manually modified in Viz Artist.

Director Naming Convention

For a Toggle container named `BannerLyr`, the following sub-directors are created:

Director	Purpose	Editable?
`BannerLyr`	**Controlling director** — contains GOTO_TRIO action channels (`switch_in`, `out`, `switch`). This is what Transition Logic addresses.	Stop points only
`BannerLyr,AX`	**A-side cross-fade** — default: alpha 0→100 over 1 second. Customise for slides, wipes, 3D flips.	Yes — but must match BX
`BannerLyr,BX`	**B-side cross-fade** — must always be identical to AX.	Yes — must mirror AX
`BannerLyr,AO`	**A-side object animation** — auto-managed. Loaded scene animations become sub-directors here.	No — auto-managed

Copyright © 2026 Vizrt

Director	Purpose	Editable?
`BannerLyr,BO`	**B-side object animation** — auto-managed.	No — auto-managed

> **AX and BX must always be identical:** Toggle may load content on either side. If the cross-fade animations differ, transitions will look inconsistent depending on which placeholder is active.

Before adding the plug-in to a container in a scene, please make sure that:

1. You give the container a unique and sensible name. This name becomes the Toggle-Layer name. Also, as the Toggle plug-in automatically generates directors, which are named based on the container name and its parents names, it is worth considering if the location of the container that hosts the Toggle plug-in is in the correct position in the scene tree.
2. The container the Toggle plug-in is added to must not have any child containers.
3. A director with the same name as the container (Toggle-Layer name) is created.
4. The director has a minimum of two stop points and that these stop points are named correctly. The names must correspond with the Toggle-Layer states.

> **IMPORTANT!** One of the Transition Logic requirements is to have a stop point named “O”. Best practice is to always name the first stop point “O” (frame 0).

After the Toggle plug-in has been added to the container, the plug-in icon can be clicked to see its properties. Clicking the **Default Key Frames** button creates the most commonly used Key Frames for triggering the toggling between the current and the other objects. Adjusting the timing can easily be done by moving the Key Frames in the stage.

A/B Switching Mechanism

Switch Operation

1. **Animate out:** Plays the X director of the current placeholder (fade out) in reverse.
2. **Flip:** Swaps which side is `$current` and which is `$other`.
3. **Rename:** Renames containers are renamed so external TCP commands always address `$current` / `$other` consistently.
4. **Animate in:** Plays the X director of the new current placeholder (fade in).

Two Trigger Scenarios

Scenario	What happens
State change (e.g. $O → $IN)	The layer Director animates via GOTO_TRIO. Toggle keyframes on the timeline trigger AX/BX at the designed moment.
Object swap (same state, new content)	No Director movement. The `switch` action fires directly, cross-fading from current to other.

Copyright © 2026 Vizrt

Parameter Reference

Basic

Parameter	Default	Description
Object	*empty*	Geometry pool path to load onto `$other` (for example, `GEOM*MyDir/MyTemplate`). Setting this triggers the load.
Manage stops	On	Rebuilds stop-points on loaded objects by calling ControlObject's `restore_stops` / `remove_stops`.
Auto-continue other object on Switch	On	Automatically sends `continue_animation()` to the object director after switch. Respects ControlObject's *Filter Continues* flag.
State control only	Off	Ignores all commands — only GOTO_TRIO state changes are processed. Optimisation for empty overlay scenes.
Controls video	Off	Allocates/clears clip channels per placeholder for ControlVideo integration.

Adaptive Scenes

Parameter	Default	Description
Support Box Transformation	Off	Initialises Box Transformation on Toggle and placeholder containers. Subscene mode only.
Support Flexbox layouting	Off	Creates Flexbox plugins on placeholders; uses `active` toggling instead of position animations.
Format Name	*empty*	Forces subscenes to render in a specific format. If empty, the master scene format is used.

Advanced

Revealed when **Show advanced** is enabled.

Copyright © 2026 Vizrt

Parameter	Default	Description
Preload objects	*template*	List of geometry pool paths to preload (one per line, `#` = comment). Used with the Initialize action.
Director Command	*empty*	Raw director command string to execute on current/ other object director.
Object director preview position	*empty*	Symbolic name or timecode for the `show_position` action.
Dummy	100	Internal parameter used to anchor the controlling director's animation channel. Do not modify.

Actions

Action	Description
Switch	Full A/B cross-fade transition: animate out current → swap → animate in new current.
Out	Animate out the current placeholder (reverse-play X director).
In	Animate in the current placeholder (forward-play X director).
Switch in	Swap other→current and animate in (no out animation on old).
Switch in preview	Hide current without animation, swap, seek to preview position.
Switch without animation	Swap visible containers instantly — no AX/BX playback.
Continue	Send `continue_animation()` to the current object director.
Continue Noanim	Seek object director to next stop without animation.
Default Keyframes	Generate GOTO_TRIO action channels (`switch_in` , `out` , `switch`) on the controlling director.
Delete Default Keyframes	Remove auto-generated keyframes. Run before re-generating after adding new stop points.
Initialize	Clear object cache and reload everything in the preload list.

Copyright © 2026 Vizrt

Action	Description
Run director command on other	Execute the *Director Command* string against `$other`'s object director.
Run director command on current	Execute the *Director Command* string against `$current`'s object director.
Position object director at preview location	Seek the current object director to the timecode in *Object director preview position*.
Move object director to next stop	Seek the current object director to the next stop point.
Delete containers of all cached objects	Re-initialise placeholders and clear all cached containers from the tree.
Delete containers of inactive objects	Delete cached containers that aren't currently active on either A or B.

Default Keyframes

When **Default Keyframes** is invoked:

1. Scans the controlling director (same name as the Toggle container) for all stop points.
2. Creates three **action channels** on that director: `switch_in`, `out`, and `switch`.
3. For each transition between stop points, inserts direction keyframes (±0.02s from boundaries) that tell AX/BX which way to play: forward (fade in) or reverse (fade out).
4. Creates the four sub-directors (`,AX`, `,BX`, `,AO`, `,BO`) with default one second alpha animations.

This is what enables GOTO_TRIO to automatically trigger the correct transition when jumping between states.

Object Caching

Toggle caches every loaded object. Requesting the same scene path again returns the cached version instantly — no Graphic Hub load occurs. The cache can be:

- **Pre-populated:** List paths in the *Preload objects* text area and invoke **Initialize**.
- **Built on demand:** Each first load is cached automatically.
- **Cleared:** Use *Delete containers of all cached objects* or *Delete containers of inactive objects*.

Custom Transitions

Replace the default AX/BX alpha dissolve with any animation style:

- **Slide:** Animates position X/Y instead of opacity.
- **Scale:** Scales current to 0, scale other from 0 to 1.
- **3D flip:** Rotates containers on the Y-axis.

Copyright © 2026 Vizrt

- **Wipe:** Animates crop or stencil masks.
- **Dissolve with blur:** Combines opacity with a blur plug-in.

The only constraint: **AX and BX must always be identical animations.**

Interaction with ControlObject

Interaction	Description
Stop-point management	Before loading, calls `ControlObject.remove_stops`. After loading, calls `ControlObject.restore_stops` to recreate stop points on the object director.
Continue filtering	Before sending `continue_animation()`, checks ControlObject's *Filter Continues* parameter. If true, suppresses the continue.
Video channel allocation	When *Controls video* is on, sends the allocated clip channel number to ControlObject.
Subscene root discovery	Locates the ControlObject plugin on the loaded scene's root to determine the subscene boundary.

Example: Master Scene with Two Layers

```
Scene "MasterScene"
  │
  ├── Container "BannerLyr"           ← Toggle plugin
  │   ├── Director "BannerLyr"        ← stops: $O (frame 0), $IN (frame 50)
  │   ├── Director "BannerLyr,AX"     ← alpha 0→100 cross-fade
  │   ├── Director "BannerLyr,BX"     ← identical to AX
  │   ├── Director "BannerLyr,AO"     ← auto-managed
  │   ├── Director "BannerLyr,BO"     ← auto-managed
  │   ├── $current
  │   │   └── $object (loaded scene A)
  │   └── $other
  │       └── $object (loaded scene B)
  │
  └── Container "TickerLyr"           ← Toggle plugin
      ├── Director "TickerLyr"        ← stops: $O, $TICKER
      ├── ...AX, BX, AO, BO
      ├── $current
      └── $other
```

Typical Broadcast Workflow

1. **Load:** Set `Toggle*object` to a GEOM path → scene loads onto `$other`.

Copyright © 2026 Vizrt

2. **Data:** Push field values to `$other$object*ControlObject*in`.
3. **Take:** Layer director executes `GOTO_TRIO $O $IN` → default keyframes trigger `switch_in` → Toggle cross-fades from nothing to the new graphic.
4. **Update:** Push new data to `$current$object*ControlObject*in` (no animation needed).
5. **Swap:** Load a different template onto `$other`, push data, then `switch` → A/B cross-fade.
6. **Out:** Layer director executes `GOTO_TRIO $IN $O` → default keyframes trigger `out` → current fades away.

Technical Notes

- The container **must have a unique name** before adding the Toggle plug-in. Director names are derived from the container's tree path.
- Toggle only generates placeholders and directors if none exist from before.
- After changing *Support Box Transformation* or *Support Flexbox*, the scene must be **saved and reloaded** to regenerate the Toggle structure.
- In subscene mode, after `SHOW $IN` + `switch INVOKE`, a `continue INVOKE` must follow to complete the animation.
- `sitch_no_anim` is an intentional internal name (not a typo in this documentation).
- Box Transformation / adaptive scenes are supported in **subscene mode only**, not with merged geometries.

See Also

- **Transition Logic** section of the Viz Artist User Guide.

Copyright © 2026 Vizrt

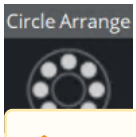
2.4.2 Arrangement Container Plug-ins

The following Container plug-ins are located in the Arrange folder:

- Circle Arrange
- Grid Arrange
- Time Displacement

Copyright © 2026 Vizrt

Circle Arrange



Circle Arrange is a tool for arranging a set of containers into a circular structure.

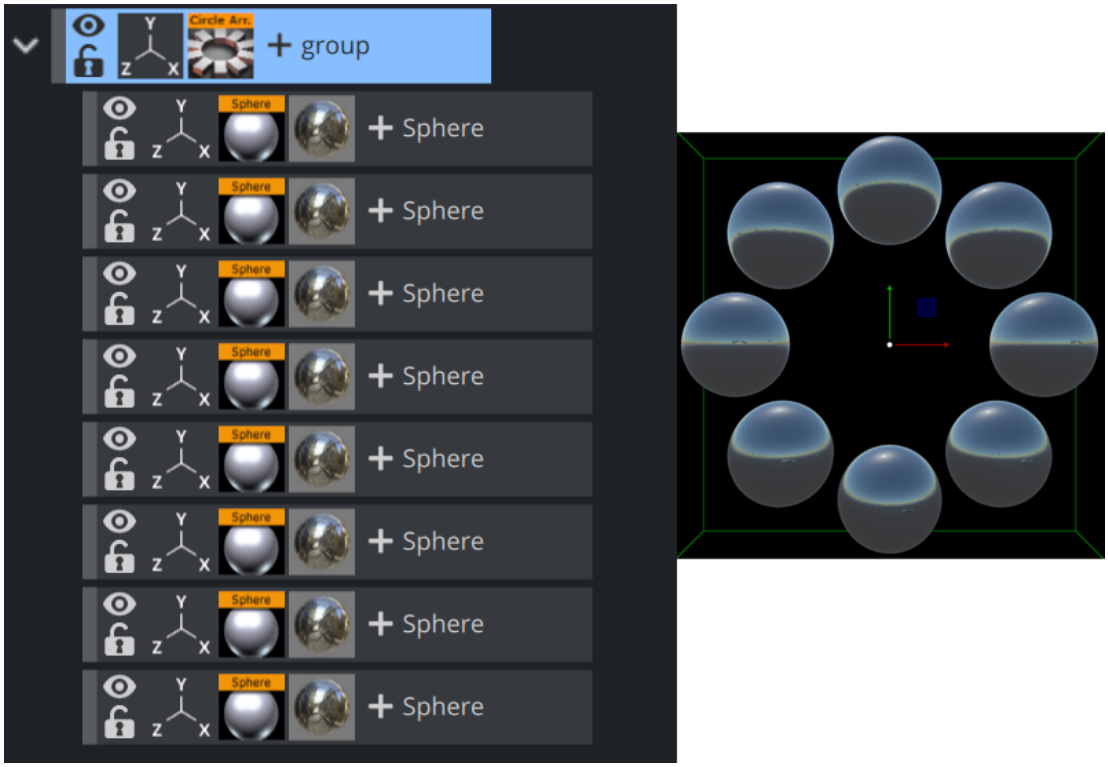
> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Arrange

Circle Arrange Properties

- **Number of items:** Allows you to define how many items the structure is to made up of.
- **Number of Rows:** Sets the number of rows/rings for the circular structure.
- **Distribution:** Allows you to define how the containers are to distributed in the circular shape.
  - **Monospace:** Sets the same number of items in each circle row. The spacing between objects in the outer rows is then bigger than in the inner rows.
  - **Equal:** Distributes the objects between the rows so the distance between the objects in the different rows are equal. The number of containers is larger in the outer rows.
- **Inner Radius:** Sets the radius from the center to the first container ring.
- **Outer Radius:** Sets the radius from the center to the outer ring.
- **Start Angle:** Setting this above -180° creates an open section in the circle structure. The opening is created in a clockwise direction.
- **End Angle:** Setting this below 180 creates an open section in the circle structure. The opening is created in a counterclockwise direction.
- **Orientate:** Orients the containers with their bottom towards the center of the circular structure when enabled.
- **Rebuild Containerlist:** Rebuilds the structure after having added or removed containers form the group.

Copyright © 2026 Vizrt

To Arrange Containers in a Circle



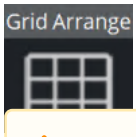
1. Create a group container, and name it `Circle`.
2. Add a **Sphere** as a Sub-Container to the *Circle* container.
3. Add a material and/or a texture to the *Sphere* container.
4. Open the Transformation Editor for the Sphere container, and set **Scaling** (locked) to `0.2`.
5. *Optional*: Animate the *Sphere* object.
6. Make 15 copies of the Sub-Container, totaling the number of Sphere containers to 16. All copies should be placed as Sub-Containers of the Circle container and have the same position as the original.
7. Add the Circle Arrange container plug-in to the *Circle* container.
8. Open the Circle Arrange editor and do the following:
   - Set **Number of items** to `16`.
   - Set **Number of Rows** to `2`.
9. Click the **Rebuild Container list** button. The plug-in positions all Sub-Containers in a circular structure. The order of the containers in the group decides their placing in the circular structure. If more than one row is selected, the first containers end up in the outer circle and the last containers in the inner circle.

See Also

- Grid Arrange

Copyright © 2026 Vizrt

Grid Arrange



The Grid function is an effective tool for arranging a set of containers into a grid or tabular structure.

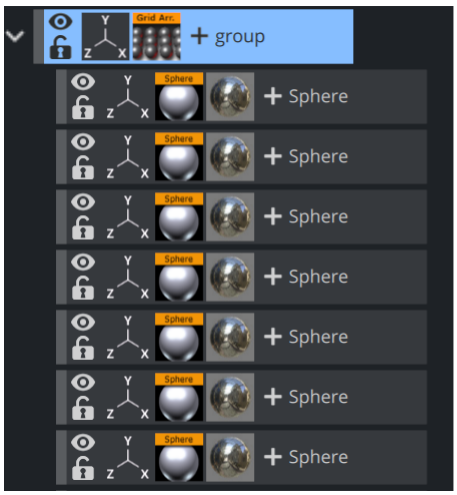
Note: This plug-in is located in: Plugins -> Container plug-ins -> Arrange

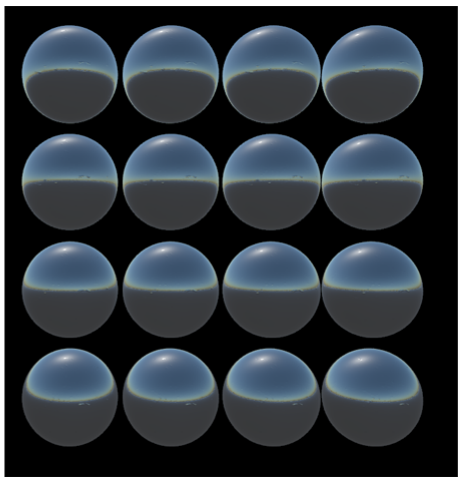
Grid Arrange Properties

- **Number of Rows:** Sets the number of rows the containers are to be distributed over.
- **Number of Columns:** Sets the number of columns the containers are to be distributed over.
- **Row Offset:** Sets the distance between the rows.
- **Column Offset:** Sets the distance between the columns.
- **Start In Corner:** Selects which corner to start the arrangement.

Note: The product of number of rows and number of columns should normally be equal number of containers. If it is higher, all rows are not equal, if it is lower, some containers are not shown, but they are still rendered.

To Arrange Containers in a Grid





1. Add container to the Scene Tree.
2. Name it `Grid`.
3. Add a **Sphere** as a Sub-Container to the *Grid* container.
4. Add a material and/or a texture to the *Sphere* container.
5. Open the Transformation Editor for the *Sphere* container.
6. Set **Scaling** (locked) to `0.2`.
7. *Optional*: Animate the Sphere object.

Copyright © 2026 Vizrt

8. Create 15 copies of the Sub-Container (total of 16 Sphere containers). All copies should be placed as Sub-Containers of the Grid container and have the same position as the original.
9. Add the Grid Arrange plug-in to the *Grid* container.
10. Open the Grid Arrange editor and set these parameters:

- **Number of Rows:** `4`.
- **Number of Columns:** `4`.
- **Row Offset:** `40.0`.
- **Column Offset:** `40.0`.

Grid Arrange positions all Sub-Containers in a grid structure. The order of the containers in the group decides their placing in the grid structure. If more than one row is selected, the first containers end up in the first row(s) and the last containers in the last row(s).

See Also

- Circle Arrange

Copyright © 2026 Vizrt

Time Displacement



Time Displacement allows you to use a gray-scaled alpha image for setting animation time offsets for all the objects in a group container. This is typically useful if you want to create some kind of randomized animation effect or that should follow a certain pattern.

What the function does is to create links between the animated containers and the pixels of the gray-scale image and then use the alpha level of each pixel to set the offset of the containers. The range from 100% alpha to 0% alpha on the pixels sets the offset level. By default 100% alpha sets no offset and 0% sets maximum offset.

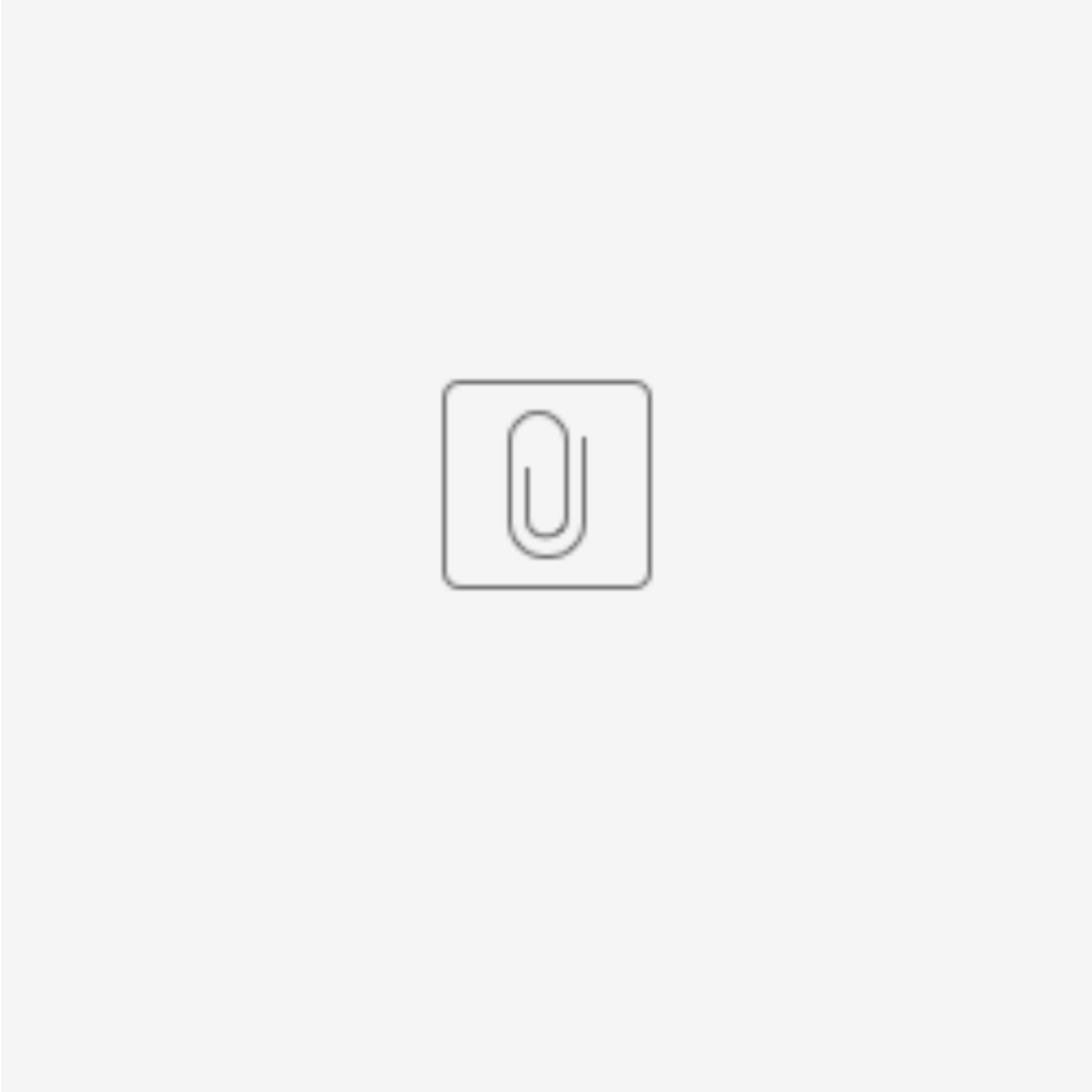
> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Arrange

Time Displacement Properties

- **Image:** Drag the alpha image you want to use for creating the containers offsets onto the drop zone here.
- **Invert:** Inverts the link between the image pixels and offsets. 0% gives no offset and 100% gives maximum.
- **Image Width:** Sets the height of the alpha image.
- **Image Height:** Sets the height of the alpha image. The product of **Image Width** multiplied with **Image Height** should normally be the same as the number of containers that are to be used in the function.
- **Pixel Reference:** Defines the way the pixels should be associated to the animated containers. You can choose between:
  - **Order in Tree:** Decides how the pixels should be associated.
  - **Name of Container:** Sets the order in which the containers are associated with the pixels.
- **Search Tree:** Sets the search level of the function, when it decides which Sub-Containers to include in the function:
  - **One Level:** Includes only the first level of sub-containers in the function.
  - **Recursive:** Includes all levels of sub-containers in the function.
- **Threshold Action:** Executes if the pixel value is lower than the threshold value.
  - **None:** Does not perform an action.
  - **Skip Pixel:** Skips the pixel in the offset process.
  - **Visible**- Makes objects visible when the pixel value is lower than the threshold value.
  - **Invisible**- Makes objects invisible when the pixel value is lower than the threshold value.
- **Threshold:** Sets the threshold value where one of the above selected actions are performed.
- **Stage Range (frames):** Sets the range of the offset distribution. A high value results in bigger differences in offset between containers that are associated with different pixel alpha values.
- **Stage Offset (frames):** Adds a static offset value to all containers.
- **Filter Type**- Filters the containers in or out. Available options are; **None**, **Including** and **Excluding**.
- **Filter String**- Sets the name of the containers that are to be included in the filter. Use of asterisks is supported.
- **Execute:** Starts the operation.

Copyright © 2026 Vizrt

To Set Animation Time Offsets with a Grayscaled Alpha Image



1. Add a group to the scene tree and name it `TimeDisplace`, and the TimeDisplace plug-in to it.
2. Create a Sub-Container of the *TimeDisplace* container and name it `Sphere`.
3. Create a new group container as a Sub-Container of *Sphere*.
4. Add a Sphere geometry plug-in and material to the group container.
5. Animate the group container from Position X `-200.0` to `200.0`.
6. Make six copies of the *Sphere* container under the *TimeDisplace* container.
7. Add the Grid Arrange plug-in to the *TimeDisplace* container.
8. Open the Grid Arrange editor and set Number of Rows to `6` and Row Offset to `30.0`.
9. *Optional*: Remove the Grid Arrange plug-in.
10. Add the Corena plug-in and rename the containers to Sphere, starting at index `0`. This allows you to filter out specific containers using the TimeDisplace filter.

Copyright © 2026 Vizrt

11. *Optional*: Remove the Corena plug-in.
12. Open the *TimeDisplace* editor and add an alpha image to the Image drop zone.
13. Click **Execute**.



14. Click the Stage to see the offsets being added to the containers.
15. Run the animation to see the time offset in action!

See Also

- Grid Arrange
- Corena

Copyright © 2026 Vizrt

2.4.3 Control Container Plug-ins

The following Container plug-ins are located in the Control folder:

- Apply Shared Memory
- Common Control Properties
- Control Action
- Control Action Table
- Control Audio
- Control Bars
- Control Chart
- Control Clip
- Control Clipplayer Texture Renderer
- Control Clock
- Control Condition
- Control Container
- Control Data Action
- Control Datapool
- Control DP Object
- Control FeedView
- Control Field Renamer
- Control Geom
- Control Hide in Range
- Control Hide on Empty
- Control Image
- Control Key Frame
- Control List
- Control ProtoList
- Control Map
- Control Material
- Control Multihop
- Control NDI
- Control Num
- Control Object
- Control Omo
- Control Parameter
- Control Payload
- Control Pie
- Control PreProcessing Texture Renderer
- Control Scaling
- Control Sign Container
- Control SoftClip
- Control Stoppoint
- Control Text
- Control VBI

Copyright © 2026 Vizrt

- Control Video
- Control World
- Placeholder
- Control Texture Slots

Copyright © 2026 Vizrt

Apply Shared Memory



The Apply Shared Memory plug-in allows values to be transferred into Viz Engine using mechanisms provided by shared memory maps that can be applied to control plug-ins such as Control Text. Just as Control Object acts as a distribution hub for content provided by control applications, Apply Shared Memory acts as a distribution hub for content provided by external systems feeding real-time data through the shared memory maps.

The Control plug-ins that Apply Shared Memory can apply content to resides in the same folder.

Just like Control Object, Apply Shared Memory only applies values to Control plug-ins residing in the subtree of the scene tree where Apply Shared Memory resides. Also like Control Object, it matches the keys of incoming content with the field identifiers of individual control plug-ins to determine where to apply which incoming content value.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

This page contains the following topics and procedures:

- Apply Shared Memory Properties
- Visualize Composite Data Structures
- How Shared Memory Keys are Determined
- To Visualize Stock Data from Shared Memory
- To Test Stock Visualization

Apply Shared Memory Properties

- **Shared Memory Source:** Specifies which shared memory map to apply values from.
  The **Scene**, **Global** and **Distributed** choices select the shared memory maps that are referred to as *Scene.Map*, *System.Map* and *VizCommunication.Map* respectively in the section.
- **Memory Base Key:** Provides the key prefix path to combine with the field identifiers of control plug-ins to determine the key to look up in the chosen shared memory map when applying values to a particular field exposed by a control plug-in.
- **Expose Base Key:** Exposes the **Memory Base Key** property as a controllable field. Once this option is turned on the properties **Field Identifier** and **Description** as described in the Common Control Properties section can be specified.
  The **Expose Base Key** property is typically used to allow a control application to control the base key which determines where in the shared memory map to pull values from. However, since the exposed property is an exposed control field like any other, a parent Apply Shared Memory plug-in can be used to make the shared **Memory Base Key** property controllable by external systems through a shared memory map.
- **Shared Memory Keys:** Lists the currently applied shared memory key and value pairs.

Choosing whether to allow the exposed field to be controlled by control applications or external systems feeding the shared memory map is done by placing either a Control Object or another Apply Shared Memory plug-in on a parent container, respectively.

Visualize Composite Data Structures

The **Memory Base Key** property combined with the field identifiers exposed by Control plug-ins can be seen as a way to visualize sets of related key-value pairs stored in shared memory maps. While the **Memory Base Key** property specifies where in the shared memory map to locate the set of related key-value pairs, the exposed

Copyright © 2026 Vizrt

field identifiers specify which of the related key-value pairs from the selected set to visualize in a particular way in the scene subtree.

For example, the **Memory Base Key** property can specify a shared memory path representing a particular stock symbol, while the field identifiers of Control plug-ins pick out particular values such as stock *name*, *value* and *change* from that location in the shared memory map.

Since Apply Shared Memory only affects the scene subtree it is placed on, visualizing multiple composite data structure instances sharing the same structure and presentation can easily be achieved by copying subtrees and only change the Memory Base Key property.

For example, design a stock presentation design once, and copy it many times to create a scene showing several distinct stocks using the same visual style. Turning on the **Expose Base Key** property gives users of control applications control of which stocks to show in each presentation copy, while the data content for each stock is still supplied in real-time through the shared memory map.

How Shared Memory Keys are Determined

The shared memory keys used to look up values in a shared memory map is constructed by the combination of an absolute path specified in **Memory Base Key** property and an either relative or absolute path consisting of the field identifier of an exposed field. Any path starting with a slash (/) is considered an absolute path, while all other paths are considered relative.

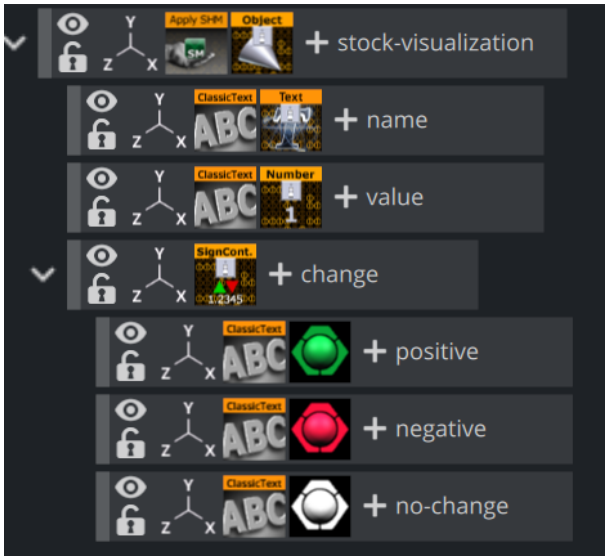
If the field identifier is an absolute path, then that absolute path alone is the result of the resulting shared memory key used. For example, the field identifier */weather/norway/bergen/temperature* results in the shared memory key */weather/norway/bergen/temperature* regardless of what the **Memory Base Key** property is.

If the field identifier is a relative path, the path is interpreted relative to the absolute base path provided in the **Memory Base Key** property, and the absolute path resulting from that interpretation is used as the shared memory key. For example, a **Memory Base Key** property of */weather/sweden/stockholm* and a field identifier of *temperature* results in the shared memory key */weather/sweden/stockholm/temperature*.

Like file system paths, a path segment consisting of a single dot (.) is considered referring to the current directory, while two dots (..) is considered referring to the parent directory. This notation is resolved to a regular absolute path before using the result as a shared memory key. For example, a **Memory Base Key** property of */weather/./norway/trondheim* and a field identifier of *../bergen/windspeed* resolves to the shared memory key */weather/norway/bergen/windspeed*.

Copyright © 2026 Vizrt

To Visualize Stock Data from Shared Memory

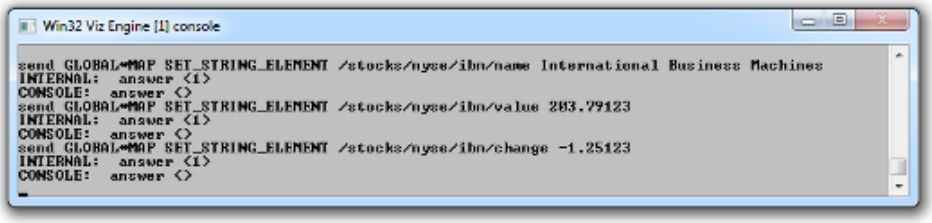


This procedure visualizes composite content fed through a shared memory map to the Control plug-ins Control Text, Control Num and Control Sign Container.

1. Create a new container.
2. Drag and drop Apply Shared Memory onto this container.
3. Set the **Memory Base Key** property of the Apply Shared Memory plug-in to `/stocks/nyse/ibm`.
4. Create two text geometries as children of this container.
5. Drag and drop Control Text into the first text geometry container.
6. Set the **Field Identifier** property of Control Text to `name`.
7. Turn off **Use formatted text** property of Control Text.
8. Drag and drop Control Num onto the second text geometry container.
9. Set the **Field Identifier** property of the of Control Num to `value`.
10. Create a third container as a child of the Apply Shared Memory container.
11. Drag and drop Control Sign Container onto this container.
12. Set the Field Identifier property of Control Sign Container to `value`.
13. Create three text geometries as children of the Control Sign Container container.
14. Drag and drop Control Num onto each of these three text geometry containers.
15. For each of these three Control Num plug-ins, set the **Field Identifier** property to `change`.
16. Arrange the layout of the text geometries, and apply materials for styling.

Copyright © 2026 Vizrt

To Test Stock Visualization



To test that stock visualization commands can be sent to populate the global shared memory map, see the descriptions in the section.



While these commands are suitable for testing the designs, a real external system that feeds the Viz Engine with data would use a more efficient data transfer protocols to populate the shared memory map. See the External Data Input section of the Viz Artist User Guide for information about sending data efficiently using UDP.

When changing a value in the shared memory map that is visualized by a Control plug-in the scene updates immediately to reflect the new value. The shared memory map can be populated with stock values for many stock symbols, including stocks that are not visualized at the moment. As soon as the **Memory Base Key** property of Apply Shared Memory is changed to visualize the values for a different stock symbol, and the scene updates immediately to reflect the value previously populated into the shared memory map for that stock symbol.

Copyright © 2026 Vizrt

Common Control Properties

Most Control plug-ins have a set of common properties:

- **Field Identifier:** Used by the control client applications to identify the editable item in the scene. Only English characters (A-Z, a-z), numeric digits (0-1), dot (.) and dash (-) are allowed. When a dot is used in the identifier (for example, *1.score*) it means this is a sub-field. Multi-level sub-field is allowed. When making templates for Viz Trio, this should be a numeric value, starting at 1. The numbers are used to create the tab-order between the editable items. For Viz Pilot and Viz Ticker, this can also be a descriptive text string.
- **Description:** Describes the tab field or editable object. This is used as a description for the items when used in the control clients.
- **Show only in Expert Mode:** Hides the tab-field properties for the user if Expert Mode is enabled in Viz Trio. For more information, read about Viz Trio's macro commands, and the command *set_expert_mode_enabled*.

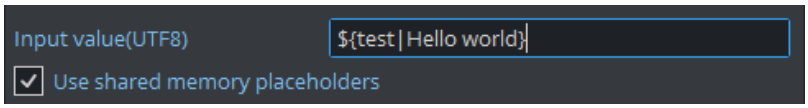
Shared Memory Placeholders

Some Control plug-in supports Shared Memory placeholders. Placeholders can be described by *${shm_key}*, where *shm_key* is the name of a Global Shared Memory Map. You can also define a default value by adding a pipe and the value. For example: *${shm_key|Default}*, where *Default* is used as default value if the Shared Memory Map is empty or does not exist.

Shared Memory maps can be accessed and changed in various ways, like an external command, script etc...

Examples

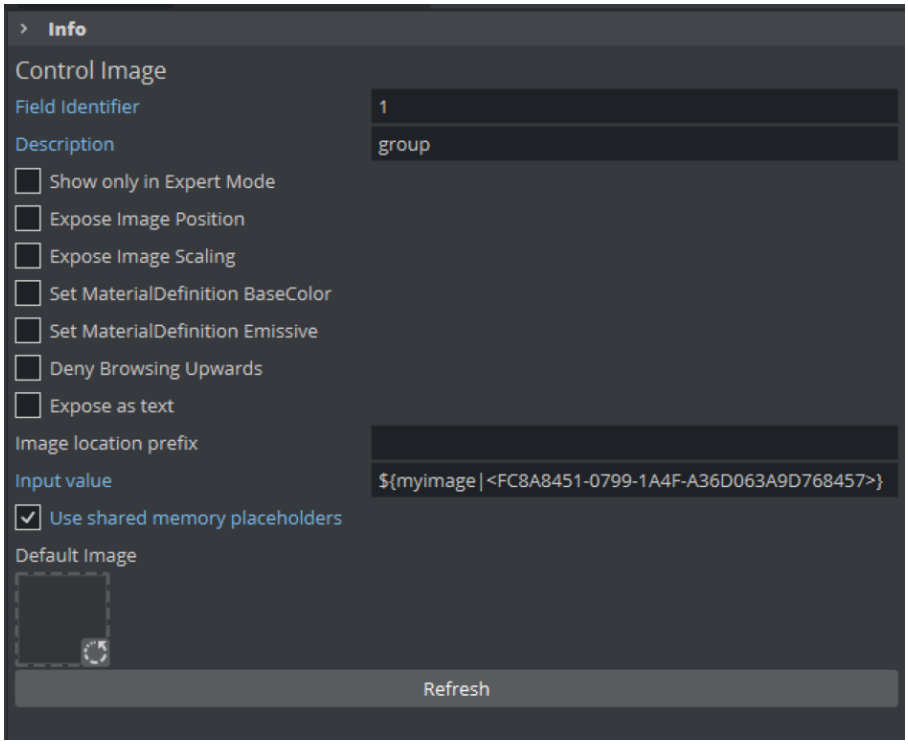
Control Text



This shows "Hello world" by default. By sending **GLOBAL*MAP SET_STRING_ELEMENT test This is my new value from a shared memory** the text changes to the new value.

Control Image

Copyright © 2026 Vizrt



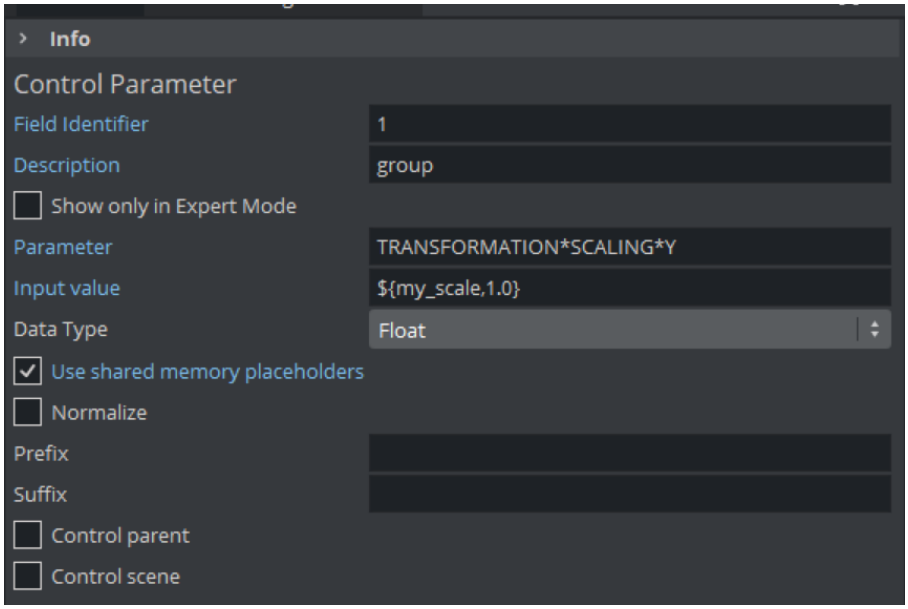
This show loads a certain image one the shared memory variable *myimage* has been changed. Otherwise, it falls back to the given default image. By sending **GLOBAL*MAP SET_STRING_ELEMENT myimage <any_uuid>** the text changes to the new value.

When using the image location prefix, be sure to have the value set correctly (for example, `IMAGE*/Vizrt-Quality_Assurance/Engine/Standard_Library/ControlImage/`).

You can then invoke a change of the SHM variable by sending, for example `GLOBAL*MAP SET_STRING_ELEMENT myimage testimage`. This forces the image */Vizrt-Quality_Assurance/Engine/Standard_Library/ControlImage/testimage* to be loaded and set.

Control Parameter

Copyright © 2026 Vizrt



This applies a value to the Y scaling of the container with 1.0 as default value. By sending `GLOBAL*MAP SET_DOUBLE_ELEMENT my_scale 2.0`, the scaling changes to 2.0.

Information: Shared memory placeholders are available in Viz Plug-ins for Viz Engine 4.4.1 or later.

The following plug-ins support Shared Memory placeholders:

- Control Num
- Control Text
- Control Image
- Control Parameter
- Control PreProcessing TextureRenderer

Copyright © 2026 Vizrt

Control Action



The Control Action plug-in executes an action, for example Viz Engine commands or Control Object commands, when receiving input on the *input* field.

The action can contain more than one command to be called, and the commands must be separated by semicolons.

When **Notify Only When Value Change** is checked the actions are only triggered if the input value differs from the one already stored on the “input” field from previous invocations.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Action Properties

- **Input Value:** Specifies the value input to compare against.
- **Notify Only When Value Change:** Triggers a command when a value is changed.
- **Action Type**
  - **Viz Command:** Triggers an internal Viz Engine command (for example, `"THIS_SCENE*STAGE*DIRECTOR*Audio START"`).
  - **ControlObject Command:** Triggers a Control Object command (for example, `"ON 1 SET abc"`).
- **Action:** Actions to execute if values match.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Action Table



The Control Action Table plug-in allows defining a table of actions similar to Control Action. The actions are associated with a given value.

When Control Action Table receives input, it compares the received data to each of the values. If the data matches one of the values, Control Action Table starts the corresponding action.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Action Table Properties

- **Input Value:** Specifies the value input to compare against.
- **Notify Only When Value Change:** Triggers a command when a value is changed.
- **Value 1/2:** The value that the given action should be associated with.
- **Action 1/2:** Actions to execute if value match.
- **Default Action:** Executes if the input does not match any of the specified values.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Audio



The Control Audio plug-in enables the operator of a control application to choose a sound clip.

To expose the different properties, and change the audio file in one of the control applications, add Control Audio to the same container as Audio.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Audio Properties

- **Director:** Sets the name of the stage director the audio channel is located below.
- **Audio Key Frame:** Sets the ID/name of the audio clip Key Frame.

Note: It is mandatory to add Director and Audio Keyframe name.

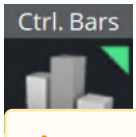
Important: Make sure the audio clip name does not contain any white spaces.

See Also

- Common Control Properties
- Control Object
- Audio

Copyright © 2026 Vizrt

Control Bars



The Control Bars plug-in allows binding of tab field values to the *Bar Value* fields in Bar Values.

Each *Bar Value* property in the Bar Values editor is presented as a separate tab field in Viz Trio.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Bars Properties

- **First Bar Field ID:** Sets the tab-field ID for the first *Bar Value*.
- **Last Bar Field ID:** Sets the tab-field ID for the Last *Bar Value*.
- **Expose Bars Max/Total Value:** Exposes the Bars Max/Total value as separate tab fields when enabled.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Chart



The Control Chart plug-in binds control of chart data to a Control Object. It currently supports the graph geometry plug-in and the bar values, pie values and Basic Geometry Visual Data Tools plug-ins. It serves the purpose to interface all kind of chart data. It covers graph charts, bar charts, pie charts, area charts and generally all kind of plug-in data with a table-like representation.

Control Chart offers the same kind of interface like control list. That means it delivers a schema specification as type encoding. The chart data is communicated as an XML table value fully compatible to control list values. The supported control chart commands mimic the control list commands.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Chart Properties

- **Custom Number of Exposed Rows:** Enables the Number of exposed Rows setting.
  - **Number of exposed rows:** Sets the default number of exposed rows for the scene when the number of rows can be changed by the operator. If the number of rows cannot be changed the number of exposed rows is fixed.
- **Mutable Number of Rows:** Allows the operator can add or delete rows if set to on.
  - **Custom minimum and maximum Number of Rows:** Enables the Minimum and Maximum Number of Rows settings. To the operator, the number of fields can only be added or deleted when inside the range (for example, greater than two and less than ten).

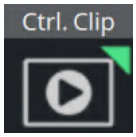
> **IMPORTANT!** If Control Chart is used in combination with Basic Geometry Visual Data Tools plug-ins, make sure the Visual Data Tools plug-in Shared Memory is set to `Inactive`.

See Also

- Common Control Properties
- Control Object
- Basic Geometry Visual Data Tools

Copyright © 2026 Vizrt

Control Clip



The Control Clip plug-in binds control of an AVI clip played by MoViz or SoftClip to a Control Object. With Control Clip, the clips may be changed within the control application.

The MoViz plug-in enables the user to play media files or media streams inside Viz Engine. MoViz uses the Microsoft DirectShow Filtergraph framework to render the media. It is possible to play files (for example, MPEG and QuickTime) or streams from a server. For the latter to work, the system must be appropriately equipped.

SoftClip is a Viz Artist/Viz Engine plug-in that can show AVI clips, either projected on a texture, or rendered directly.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Clip Properties

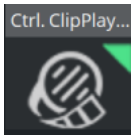
This plug-in does not have any properties or parameters except those that are common to all plug-ins.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Clipplayer Texture Renderer



The Control Clipplayer Texture Renderer sets a Clip file for the Texture Clipplayer to play.

It should be used instead of the legacy MoViz and SoftClip players as the new implementation relies on the VML player implementation.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

To Create a control able Clipplayer Texture instance:

- Click on **Workspace > Add > Clipplayer Texture**.
- Drag the new create Workspace item to the scene tree. A new geometry and a phong material are created.
    - To modify the behavior of your clip player, double click on the workspace item.
- Add the Control Clipplayer Texture plug-in to the container to allow control from a control application like Viz Trio.

Control Clipplayer Texture Properties

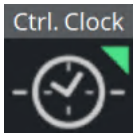
The **Check Change** option only applies a new clip if it differs from the existing one. For example, given a template containing a clip filename and a text, if only the text is changed, the clip continues playing if this element is being taken On Air.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Clock



The Control Clock plug-in allows the user to set time for a clock as well as controlling several clock functions.

The clock object is a text container in the scene with the clock function enabled.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Clock Properties

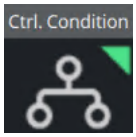
- **Expose Limit:** Enables the user set a limit for the clock.
- **Expose Direction:** Enables the user to choose the direction for the clock.
- **Action:** Sets the clock action that is the default value in the control client's clock editor.
- **Input Value:** Shows the current value for Control Clock. It is not necessary to set any value here. It is normally only used for debugging purposes.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Condition



The Control Condition plug-in triggers actions based on the input given in a tab-field with the same field identifier.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Condition Properties

- **Input Value:** Shows the current input value of Control Condition.
- **Format:** Sets the format for the input value that it should conditionally trigger on.
  - **Condition:** Sets the type of condition if Format is set to number.
- **Arguments:** Sets the arguments for the conditional operation.
- **Action Type / Action:** Sets the action that is to be performed when the condition is met. It can be a control object command like `“ON 2 SET mytext”` or a Viz Engine command prefixed by a zero and a space.
- **Use Else Commands:** Triggers an *else* action when the input data falls within the conditions when enabled.
  - **Else Action:** Sets the else action that is triggered when the input data does not fall within the conditions.

Example Actions

Actions can be Control Object commands in the form like this: `“ON 2 SET Mike Johnson”`

or commands like: `0 RENDERER*STAGE START`

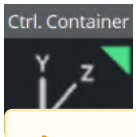
> ⚠ **Note:** Viz Engine commands must be prefixed with a zero followed by a space. This is to enable the plug-in to recognize that the command should be interpreted as a Viz Engine command.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Container



The Control Container plug-in exposes a range of different transformation properties for a container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Container Properties

The following properties can be exposed:

- Visibility
- Object
- X/Y/Z position
- X/Y/Z rotation
- X/Y/Z scaling

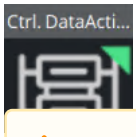
For position, rotation, and scaling a stop point Key Frame can be specified. This enables the user to control a Key Frame in an animation, for instance the end Key Frame in a position animation.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Data Action



The Control Data Action plug-in binds control of a control parameter to a Data Action.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Data Action Properties

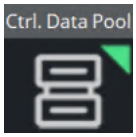
- **Input value:** Shows the current input value.
- **Prefix:** Adds a prefix, if required.
- **Postfix:** Adds a postfix, if required.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Datapool



The Control Datapool plug-in is used with the DataPool Plug-in repository, available separately.

Control Datapool is used to bind a tab field value to a DataPool plug-in variable. Any DataPool Plug-in used in a scene must have its variable listed in the **DataPool** scene plug-in. Control Datapool links a tab-field to these variables.

For more information about the DataPool plug-ins, refer to the DataPool documentation.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Datapool Properties

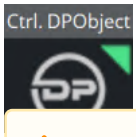
- **DataPool Variable:** The DataPool variable can be controlled by the user.
- **Input Value:** Shows the current input value.
- **Control Type:** Specifies if the tab-field value should be controlled through a DataPool variable or a field name.
- **Variable Type:** Allows the user to define the kind of data the referenced variable is linked to. The type selected here decides what kind of editor is opened in a Viz Trio client.

See Also

- Common Control Properties
- Control Object
- DataPool Plug-ins

Copyright © 2026 Vizrt

Control DP Object



The Control DP Object allows copying or linking of Data Pool objects.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control DP Object Properties

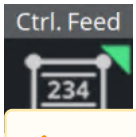
- **Input value is:** Select from Object to COPY, Object to LINK or Direct VALUES.
- **Input value:** Shows the current input value.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control FeedView



The Control FeedView plug-in sets the Feed View locator to a given value.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control FeedView Properties

- **Locator:** Sets the FeedView locator to a given value.
- **Prefix:** Sets prefix to add to the front of the locator key.

Example Input Values

```
/nyse/nasdaq/cisco /nyse/nasdaq/intc
```

If a prefix is set, it is added to the front of the locator key, a prefix of: `/nyse/nasdaq`. With an input of: `IBM`, it resolves to: `/nyse/nasdaq/IBM`

The locator field is the value as received from Control Object. It is trimmed of CR/LF before concatenated into a FeedView key.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Field Renamer



The Control Field Renamer plug-in allows for renaming the field identifier of multiple control plug-ins in a single operation.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Field Renamer Properties

- **Begin from ID:** Enter the numeric value of the first ID in the renaming range.
- **End at ID:** Enter the numeric value of the last ID in the renaming range.
  - **Follow Changes:** Changes the Begin and End IDs if set to `On`.
- **Padding:** Sets the zero padding for the desired output. Zero padding adds one or more prefixing zeros if the ID consists of fewer digits than specified.

Example: If Padding is set to `4`, the ID '14' is changed to `0014`, and the ID '103' is changed to `0103`.

- **Method:** Choose the renaming method:
  - **Set:** In the field **Set lowest ID to**, enter the desired value to change all IDs in the specified range by the same value
  - **Shift:** Shifts all the IDs in the specified range by the provided amount
- **Execute:** Click the **Execute** button to execute the rename process.

To Rename Control Fields

1. Drag the Control Field Renamer plug-in onto a container. When executing the renaming process, all Control plug-ins in that container and all of its child containers are affected.

Tip: To rename all field identifiers of an entire scene, place all containers in the scene inside a parent container and add Control Field Renamer to that parent container.

2. Set the range of IDs to be changed in **Begin from ID** and **End at ID**.

Note: If the **Follow Changes** option is set to on, the above two range fields are also changed.

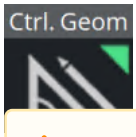
3. Use **Padding** to set the length for zero padding. Set to zero for no padding.
4. Choose the renaming **Method**.
5. Click **Execute** to execute renaming.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Geom



The Control Geom plug-in exposes the control of geometry objects to the user.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Geom Properties

- **Geom location Prefix:** Sets a path to a folder. The geometry location exposed to the control client can be a simple string, instead of a full path. This is relevant if connected to a newsroom system. For example, if a user wants to edit a text and geometry on the same tab-field by just entering a text string. If for instance the user have a folder of weather symbols and the control client receives “sunny” from the external system it loads the geometry object with the path: “geom location prefix + sunny”, which then typically would be an image of a shining sun.
- **Input Value:** Shows the current input value. Enter a value to test the relation of max/min. input and scale values.
- **Deny Browsing Upwards:** Restricts the user from browsing for objects in folders other than the selected folder and its sub-folders when enabled.
- **Use Reference:** Overwrites materials if an object is going to be replaced with another one already within the pool. Turn this option off if you encounter issues with wrong materials assigned. Default is *active*.

Configured Geom location Prefix written without prefix “GEOM*/” loads geoms correctly but is never removed from Trio field values.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Hide in Range



The Control Hide in Range plug-in hides or shows a container if the input value given by the user is within or outside a specified value range.

The input value must be from a font container that holds a numeric value, typically with Control Num.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Hide in Range Properties

- **Input Value:** Shows the current input value.
- **Lower Range Bound:** Sets the upper bound for the range.
- **Upper Range Bound:** Sets the lower bound for the range.
- **Visibility in Range:** Sets whether the container should be visible or hidden when the input value is within the range.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Hide on Empty



The Control Hide on Empty plug-in hides the container it sits on if the input value is blank (text) or zero (numbers). Use this plug-in to hide the graphics items if the user does not enter any value in the connected tab-field.

Together with the Autofollow plug-in, the plug-in can be used to make intelligent scenes that hide objects that receive no value from the Viz Trio user and also rearrange those items that have a value or text and are visible. For example, when using a lower third scene with a bug and a breaking news bar, the bug and the breaking news bar should not be shown all the time. So instead of having to make several explicit variants, make one scene and use Control Hide on Empty to hide the bug and the breaking news only if the page in Viz Trio have no value in those tab-fields.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Hide on Empty Properties

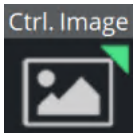
- **Input Value:** Shows the current input value
- **Treat Input as Numerical:** Hides the containers with an input value of `0` or `0.0` when enabled.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Image



The Control Image plug-in creates an image control in the control clients.

Control Image must be placed on the container that holds the image/texture.

If an image/texture is invalid or Viz Engine for some reason cannot load it, then the current image/texture remains visible.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Image Properties

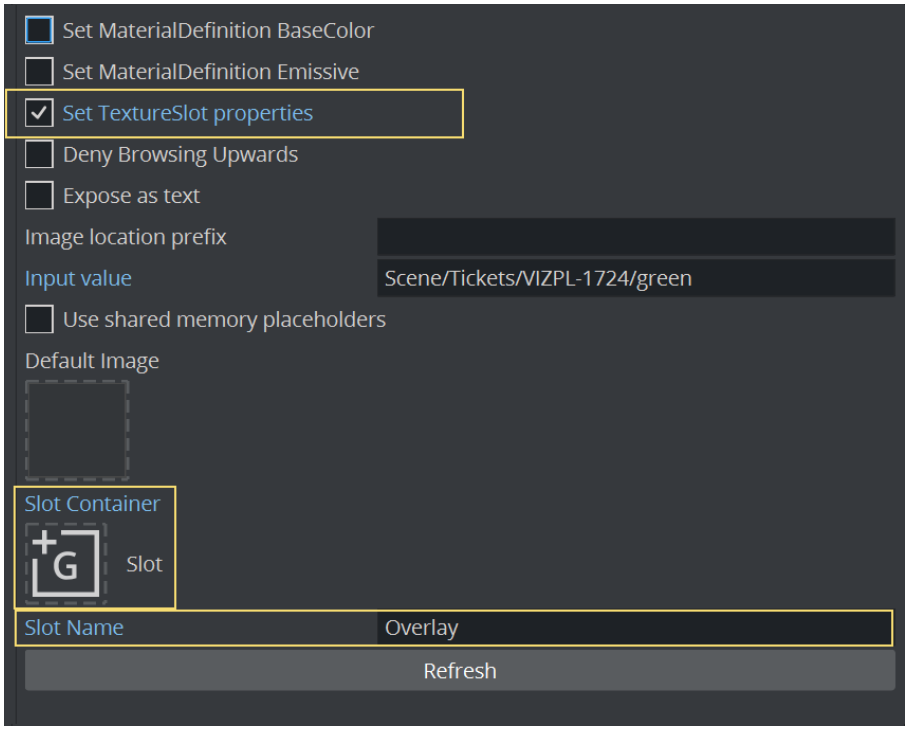
- **Expose Image Position:** Allows the user to change the position of the image/texture.
- **Expose Image Scaling:** Allows the user to change the scaling of the image/texture.
- **Set MaterialDefinition BaseColor:** Sets the given image as BaseColor channel on a PBR or Phong Material.
- **Set MaterialDefinition Emissive:** Sets the given image as Emissive channel on a PBR or Phong Material.
- **Deny Browsing Upwards**- Restricts the user from browsing for images/textures in folders other than the selected folder and its sub-folders when enabled.
- **Expose as text:** Works **ONLY** if the `Image Location Prefix` is defined
- **Image location Prefix:** If the full path to a folder is entered, the image/texture location exposed to the control clients can be a simple string, instead of a full path. This is relevant if connected to a newsroom system. For example if a user wants edit a text and an image/texture on the same tab-field by just entering a text string. This is relevant if a user has a folder containing flag images/textures, and the control client receives “US” from the external system it loads the image/texture with the path: “image location prefix + US”, which then typically would be the “Stars and stripes” (flag of the U.S.A).
- **Input Value:** Shows the current value for the plug-in. It is not necessary to set any value here. It is used for debugging purposes.
- Shared Memory placeholders

Configured **Image location Prefix** written without prefix “IMAGE*/” loads images correctly but is never removed from Trio field values.

To Use with Texture Slot

Control Image can be used to set images onto Texture Slot for Viz Engine Renderer. Enable the *Set TextureSlot properties* and give a unique name inside Texture Slot (for example, *Overlay*). If Texture Slot is located anywhere else the local container, the container must be provided:

Copyright © 2026 Vizrt



See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Key Frame



The Control Key Frame plug-in can be used to give the user control over the value of a single value Key Frame, unlike position values for instance where all three axes must be specified.

Create an animation, for instance an alpha animation, and give the Key Frame to control a name. Put Control Key on the object's container and set the parameters.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Key Frame Properties

- **Expose Time:** Enables the user to adjust the Key Frame along the time-line.
- **Control Parent:** When enabled, this option allows for two Control Key Frame plug-ins to be within one container.

> ⚠ **Note:** In this case the control Key Frame plug-ins must be added as Sub-Containers of the container.

- **Minimum Input:** Sets the minimum allowed input value.
- **Maximum Input:** Sets the maximum allowed input value.
- **Minimum Value:** Sets the minimum value, which is set relative to the minimum input value.
- **Maximum Value:** Sets the maximum value, which is set relative to the maximum input value.
- **Input Value:** Shows the current input value. Enter a value to test the relation of max/min. input and scale values.
- **Key Frame Time:** Sets the time value of the Key Frame.
- **Key Frame:** Sets the name of the Key Frame that is to receive a value by the control client user.
- **Bind Without Key Frame Prefix:** When enabled, the Key Frame name is not appended to the plugins field identifier.
  By default the ControlKeyframe binds to {fieldIdentifier}.{keyframeName}, turing this option on will bind to {fieldIdentifier}

Input Value Example

Example for the input values: If the Key Frame is an alpha Key Frame which in Viz Artist can be float values from `0` to `100`, and the following values are set:

- Minimum input = `1`.
- Maximum input = `10`.
- Minimum value = `10`.
- Maximum value = `90`.

This means that if the user gives the following input:

- `1`, which is the lowest input value allowed, the alpha value in Viz Artist is set to `10`.
- `4`, which is somewhere in between, the relative alpha value is `36,666` ... .

Copyright © 2026 Vizrt

- `10`, which is the highest value allowed, the alpha value is set to `90`.

See Also

- Common Control Properties
- Control Object

Control List



Exposes a **table of repeating rows** to external control systems. Each row is a sub-ControlObject; each cell is a Control* plugin (ControlText, ControlImage, etc.) under that sub-ControlObject.

By following some design conventions when designing the table in Viz Artist, Viz Trio shows a table editor to the user.

This editor supplies a more effective way for filling tables than having one tab field for each table cell.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

This page contains the following topics and procedures:

- Overview
- Scene Tree Structure
- Quick Start
- Parameter Reference
  - Identity
  - Row Count
  - Actions
- Field Addressing
  - Examples
- Data Serialisation
  - GET Response (XML)
  - Schema (LIST Response)
- Row Management
  - How Rows Work
- Interaction with ControlObject
- Tables and Preview Points
- To Design a Table Scene
  - Example: Election Results Table
- Technical Notes

Overview

ControlList turns a set of child sub-ControlObject containers into a **structured table** that newsroom systems (Viz Trio, Viz Pilot, MSE) can populate with data. It:

1. **Discovers rows:** Detects child sub-ControlObjects via namespace binding and registers them as rows.
2. **Exposes a table schema:** Generates VDF/XML type information describing columns (field names, types, min/max).
3. **Manages row visibility:** Shows/hides rows based on the exposed count. Rows are *pre-existing* containers that get activated or deactivated, not dynamically created.
4. **Routes data:** Receives data via the parent ControlObject's `in` protocol and dispatches values to individual cells via `INDEX` addressing.

Copyright © 2026 Vizrt

Key concept: ControlList does **not** clone or create containers at runtime. The scene designer pre-creates enough row containers (sub-ControlObjects). ControlList shows/hides them based on the data received.

Scene Tree Structure

```
Container "object"
  └── ControlObject (root)
      │
      └── Container "Scoreboard"
          └── ControlList (field_id: "scores")
              │
              ├── Container "Row1"
              │   └── ControlObject (sub, namespace: "scores")
              │     ├── ControlText  (field_id: "team")
              │     └── ControlNum   (field_id: "pts")
              │
              ├── Container "Row2"
              │   └── ControlObject (sub, namespace: "scores")
              │     ├── ControlText  (field_id: "team")
              │     └── ControlNum   (field_id: "pts")
              │
              └── Container "Row3"  (hidden if only 2 rows needed)
                  └── ControlObject (sub, namespace: "scores")
                      ├── ControlText  (field_id: "team")
                      └── ControlNum   (field_id: "pts")
```

Quick Start

1. Under the root ControlObject, create a **group container** for the list.
2. Add **ControlList** to that container and set a **Field Identifier** (for example, `scores`).
3. Create **child containers**, one per row. Each must have a **sub-ControlObject** with a namespace matching the ControlList's field ID.
4. Under each sub-ControlObject, add Control* plug-ins (ControlText, ControlImage, ControlNum, etc.) with unique `field_id` values. These are the columns.
5. Duplicate the row containers to create as many rows as your maximum needs.
6. Optionally set **Custom Number of exposed Rows** to control the initial visible count.

Parameter Reference

Identity

Parameter	Default	Description
Field Identifier	*auto*	Field ID used by the parent ControlObject for addressing (for example, `scores`).

Copyright © 2026 Vizrt

Parameter	Default	Description
Description	*empty*	Human-readable description of the list field.
Show Only in Expert Mode	Off	Hides this field from non-expert UIs in Viz Trio / Viz Pilot.

Row Count

Parameter	Default	Description
Custom Number of exposed Rows	Off	When on, the exposed row count is set manually. When off, it is derived from the number of sub-ControlObjects.
Number of exposed Rows	0	How many rows are visible/active. Only editable when *Custom Number of exposed Rows* is on. Range: 0–65 536.
Mutable Number of Rows	On	Allows external systems to dynamically change the visible row count. When data arrives with a different number of rows, the count adjusts automatically.
Custom minimum Number of Rows	Off	Enables a user-defined minimum row count. Only active when *Mutable* is on.
Minimum Number of Rows	0	Lower bound on visible rows. Only editable when both *Mutable* and *Custom minimum* are on.
Custom maximum Number of Rows	Off	Enables a user-defined maximum row count. Only active when *Mutable* is on.
Maximum Number of Rows	65 536	Upper bound on visible rows. Only editable when both *Mutable* and *Custom maximum* are on.

Actions

Action	Description
Refresh	Triggers a full rebind. Re-scans the container tree for sub-ControlObjects and re-syncs the row/namespace list.

Copyright © 2026 Vizrt

Field Addressing

ControlList uses **INDEX-based addressing** to route commands to individual cells. The syntax is:

```
WITH <list_field_id> INDEX <row> ON <cell_field_id> SET <value>
```

Examples

```
# Set team name in row 0
...ControlObject*in SET "WITH scores INDEX 0 ON team SET Yankees"

# Set points in row 2
...ControlObject*in SET "WITH scores INDEX 2 ON pts SET 42"

# Get team name from row 1
...ControlObject*in SET "WITH scores INDEX 1 ON team GET"

# Focus a cell for preview highlighting
...ControlObject*in SET "WITH scores INDEX 0 ON team FOCUS"

# List all fields in a row
...ControlObject*in SET "WITH scores INDEX 0 LIST"
```

Zero-based indexing: Row indices are **zero-based**. `INDEX 0` addresses the first visible row, `INDEX 1` the second, and so on.

Data Serialisation

GET Response (XML)

When the parent ControlObject's `GET_MODEL` or a direct GET is called, ControlList serialises visible rows as XML:

```
<entry>
  <element description="Row 1">
    <entry name="data">
      <entry name="team">Apaches</entry>
      <entry name="pts">40</entry>
    </entry>
  </element>
  <element description="Row 2">
    <entry name="data">
      <entry name="team">Eagles</entry>
      <entry name="pts">35</entry>
    </entry>
  </element>
```

Copyright © 2026 Vizrt

```
</entry>
```

Schema (LIST Response)

The type schema describes the table structure for newsroom templates:

```
<field name="scores" type="list">
  <mincount>0</mincount>
  <maxcount>7</maxcount>
  <schema>
    <field name="team" type="richtext"/>
    <field name="pts" type="integer"/>
  </schema>
</field>
```

Row Management

How Rows Work

- Rows are **pre-existing** sub-ControlObject containers in the scene tree.
- ControlList **shows** the first *N* rows and **hides** the rest by setting containers inactive.
- When data arrives with a different number of `<element>` nodes than the current count, and *Mutable* is on, the exposed count adjusts automatically.
- The count is clamped between *Minimum* and *Maximum Number of Rows* if those constraints are enabled.

⚠️ **Pre-create enough rows**ControlList cannot create new containers at runtime. If your data has 10 items but you only created five row containers, only five are visible. Always pre-create rows up to your maximum expected count.

Interaction with ControlObject

Interaction	Description
Field registration	Each sub-ControlObject registers its fields upward through the namespace binding. ControlList intercepts these and builds compound paths (for example, `WITH scores INDEX 0 ON team`).
Data dispatch	The parent ControlObject's `in` protocol routes `WITH … INDEX …` commands to ControlList, which dispatches to the correct row's sub-ControlObject.
VDF model	ControlList contributes its schema (field names, types, min/max counts) to the parent ControlObject's `GET_MODEL` response.
Namespace propagation	Sub-ControlObjects use their namespace prefix to propagate field IDs upward through the ControlList to the root ControlObject.

Copyright © 2026 Vizrt

Tables and Preview Points

A table scene might be hard to read if you need to show a large table with many rows (for example, 20) and/or columns. In such cases it is not uncommon to only show a subset of the information (for example, five rows), and rather animate the scene to show the next five rows and so on.

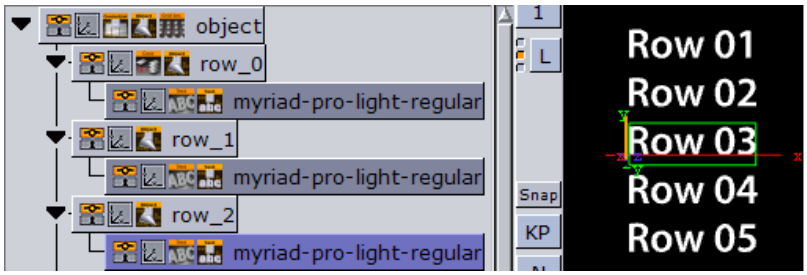
However, in such a situation, while editing table data in Viz Trio, you cannot use a basic jump to Key Frame event for preview, as this only shows the Key Frame that relates to the table's Field identifier (for example, 3), and not the rows or columns of the table.

To enable operators to preview the data in the scene, it is therefore possible to add a preview point that extends to the next starting point (for example, the next five rows of the table), using the table's (Control List) Field identifier in addition to the row number.

So, if the identifier of Control List is 03, you have to define tags with names 03.0, 03.5, 03.10 and 03.15 (zero based row index) for Viz Trio to detect the tags and jump to them on focusing the according rows in Viz Trio's Control List table editor.

- **Basic jump to Key Frame:** When adding Key Frame events (stop points, tags,...) and giving them the name of a control plug-in identifier, Viz Trio jumps the time-line to that Key Frame event when selecting the tab field with that identifier.
- **Extended jump to Key Frame:** If the Key Frame event name is `3_005` and there is a control list with identifier `3`, then Viz Trio jumps the time-line to that Key Frame event. In this case when the sixth row of the table is focused.

To Design a Table Scene



To design a table scene that is correctly interpreted by Control List, there are some design conventions that need to be followed:

1. Add a group as the root container for the table. Give it a descriptive name such as `table`.
2. Add Control List to the *table* group container.
3. Add a new group as a Sub-Container to the *table* container. This container is the design for the rows in the table. Give it a name such as `row_1`.
4. Add Control Object to the newly added row container.
5. As Sub-Containers of the row container, add text containers with backgrounds to form the design for a single table row with the required number of columns.
    - Each of the text containers must get a Control Text and the field identifiers for each must be set from 1-n depending on how many columns the table should have.
    - The Description property in Control Text is picked up by Control List and used as column headers.

Copyright © 2026 Vizrt

6. When the single row looks as it should, duplicate it to make the needed total number of table rows.
    - The *container* plug-in Coco can be used to automate this task.
    - Add Coco to the row container, and set the number of copies, for example 9 (1+9=10), and click **Execute**.
7. Arrange the rows
    - Grid Arrange can be used to automate this task.
    - Add Grid Arrange to the table container, and set the number of rows to the same as above and press **ENTER**.
    - Set the row offset and press **ENTER**.

The table item of the scene is ready and when imported into Viz Trio and read, the table tab-field is shown with a special table editor. When imported into Viz Template Wizard, a template with a custom table editor is generated.

Example: Election Results Table

```
Container "object"
  └── ControlObject (root, TL enabled)
      |
      └── Container "Results"
          └── ControlList (field_id: "results", max rows: 8)
              |
              ├── Container "Party1"
              |   └── ControlObject (namespace: "results")
              |     ├── ControlText  (field_id: "party")  → "Democrats"
              |     ├── ControlNum   (field_id: "seats")  → 218
              |     └── ControlImage (field_id: "logo")   → <UUID>
              |
              ├── Container "Party2"
              |   └── ControlObject (namespace: "results")
              |     ├── ControlText  (field_id: "party")  → "Republicans"
              |     ├── ControlNum   (field_id: "seats")  → 212
              |     └── ControlImage (field_id: "logo")   → <UUID>
              |
              ├── Container "Party3" ... (pre-created, hidden if unused)
              ├── Container "Party4" ...
              └── ... up to Party8
```

Populate from an automation system:

```
WITH results INDEX 0 ON party SET "Democrats"
WITH results INDEX 0 ON seats SET 218
WITH results INDEX 1 ON party SET "Republicans"
WITH results INDEX 1 ON seats SET 212
WITH results INDEX 2 ON party SET "Independents"
WITH results INDEX 2 ON seats SET 5
```

Technical Notes

- Row indices are **zero-based**.

Copyright © 2026 Vizrt

- Row visibility is controlled by setting containers active/inactive, not by creating or destroying scene tree nodes.
- When *Mutable Number of Rows* is on and XML data arrives with fewer `<element>` nodes, excess rows are automatically hidden.
- All sub-ControlObjects must use the **same namespace** (matching the ControlList's field_id) for proper row registration.
- Each row should have **identical column structure** (same field_id values) for consistent schema generation.
- The **Refresh** action should be invoked after manually adding or removing row containers in the scene tree.
- Parameter enable/disable logic is cascading: *min/max count* fields are only editable when both *Mutable* and their respective *Custom* flags are on.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control ProtoList



ControlProtoList is an more advanced plugin based on the Control List plugin.

What it does

Control List gives you a list where every row is the same. ControlProtoList lifts that restriction: you design several **prototypes**, and each row is a copy of one of them. A results list can mix a scoreline row with a photo row; a map can mix a pin with a polygon. Or think about the maps usecase where you want to have different prototypes for e.g. Major cities, Capitals, Counties and so on.

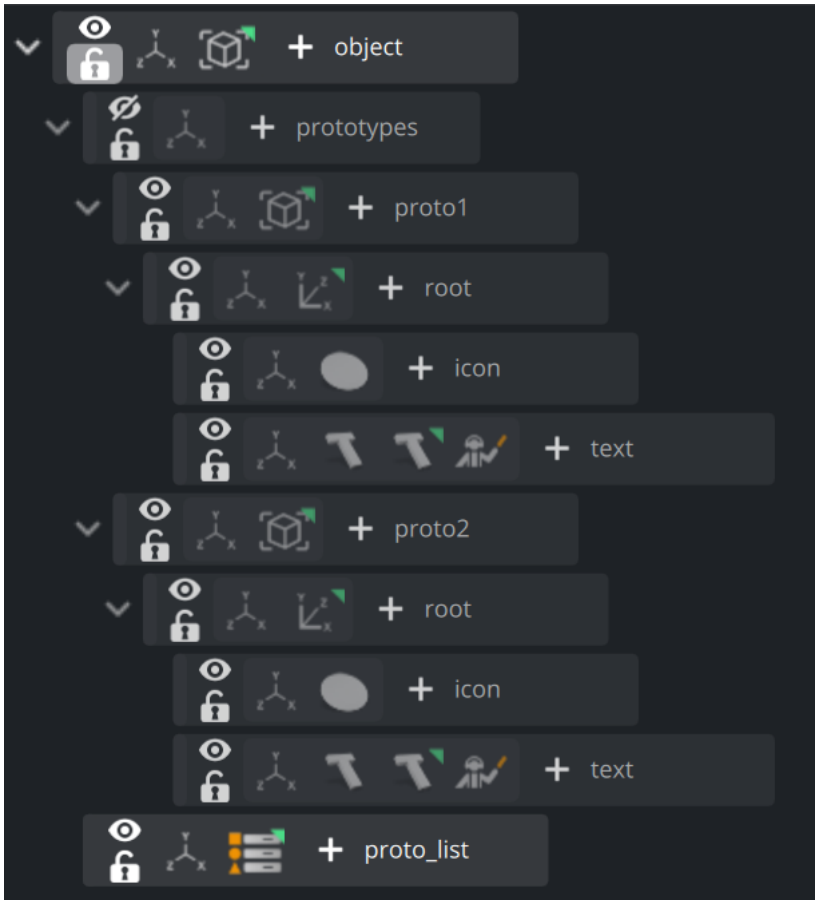
The rule that makes it work: **all prototypes must expose the same control fields.** They can look completely different, but a row has to be fillable the same way whichever prototype it came from. If one prototype exposes a headline and a picture, they all do.

Scene structure

The plugin expects a particular shape. Three things matter: a ControlObject at the top of the scene, a group whose children are the prototypes, and the container carrying ControlProtoList.

Scene tree (Example):

Copyright © 2026 Vizrt



Rules for the prototypes group

- Each prototype must carry a **ControlObject** of its own. A child without one is quietly skipped and never appears as a choice.
- The **container name is the prototype name**. It is what the operator picks from the dropdown, so name them for people: `Headline`, `Photo`, `Quote`.
- **Order matters.** The first child is the default: every new row starts as a copy of it.
- Put the group **outside** the list. Its controls stay out of the template because the plugin reads them, rather than the scene registering them.

> Once the plugin has scanned it, the prototypes group is switched inactive so it does not render. That is the plugin housekeeping, not something you broke.

Setting it up

The order below is the part worth memorizing — most problems with this plugin come from doing step 3 too early or not at all.

1. Build the prototypes first
   Create the group, build each prototype under it, put a ControlObject on each one and add the control plugins for the fields a row needs. Finish this before you touch the list.

Copyright © 2026 Vizrt

2. Add ControlProtoList to the list container
   Drop the plugin on the container that will hold the rows, and give it a **Field Identifier** — that is the name the list appears under in the template.
3. Point *Prototypes* at the group, then press Refresh
   Drag the prototypes group onto the **Prototypes** parameter. **Refresh** makes the plugin scan it and build the list of choices. Press it again any time you add, rename, remove or restructure a prototype — the plugin does not notice on its own.
4. Add rows
   Add Row appends one row, always a copy of the first prototype. **Clear Rows** empties the list.
5. Give each row its prototype
   Use the prototype dropdown on the row. Switching replaces the row with a copy of the chosen prototype, keeps its position, and carries the field values across.

⚠ No **Refresh**, no choices. Until the scan has run, the dropdown is empty and every row you add is a copy of the same prototype with no way to change it.

Working with rows

Rows are real containers below the list, named `row_0`, `row_1` and so on in order. The plugin renumbers them whenever the list changes, so the names always run without gaps — do not rely on a particular row keeping a particular name.

Adding

Adding always produces a copy of the first prototype. That is by design: creating the row and deciding what it is are two separate actions, and the second one is the dropdown.

Switching a row's prototype

The row is rebuilt from the other prototype in the same position, and the values already typed into its fields are carried over. Because all prototypes expose the same fields, nothing is lost in the swap.

Removing

There is no per-row delete button. **Clear Rows** empties the list, and a control client shortens it by sending fewer rows than before.

ℹ The minimum and maximum only apply when their *Custom* checkbox is ticked.
Leave **Custom Maximum** off and the list runs to the built-in ceiling of 65,536 rows, whatever number is sitting in the Maximum box.

Parameters

The first three are specific to ControlProtoList. The rest are the standard list parameters it inherits from ControlList.

		Plugin parameters
Parameter	**Default**	**What it is for**

Copyright © 2026 Vizrt

`Prototypes`	—	The group container whose children are the prototypes. Drag the container onto it.
`Prototype Field Name`	prototype	The field identifier of the choice column. Change it if `prototype` clashes with a field of your own.
`Prototype Field Description`	Prototype	The label the operator sees above the dropdown.
`Field Identifier`	—	The name the whole list appears under in the template.
`Description`	—	The label for the list itself.
`Custom Minimum`	off	Enables the minimum row count below.
`Minimum Number of Rows`	0	Applies only when *Custom Minimum* is on.
`Custom Maximum`	off	Enables the maximum row count below.
`Maximum Number of Rows`	65536	Applies only when *Custom Maximum* is on. Adding stops once the list is full.
`Mutable Number of Rows`	on	Whether the operator may add and remove rows at all.
`Custom Exposed` / `Exposed Rows`	off / 0	How many rows the template offers up front.

Buttons

- **Refresh** — re-scans the prototypes group. Press it after any change to the prototypes.
- **Add Row** — appends one row, a copy of the first prototype.
- **Clear Rows** — removes every row.

What the operator sees

In Trio, Pilot or Template Builder the list appears under its Field Identifier, and each row carries the fields your prototypes expose plus one extra: the prototype dropdown, labelled with your *Prototype Field Description*.

Copyright © 2026 Vizrt

The dropdown is a closed list — it offers exactly the prototypes that were discovered, in the order they sit in the group, and nothing else can be typed into it. Adding a row from the client behaves the same way it does in Artist: a copy of the first prototype, which the operator then changes if they want a different one.

`Worth checking before handover`

If a field appears on some rows and not others, the prototypes have drifted apart. Every prototype needs the same field identifiers, or the template will look inconsistent as soon as an operator switches a row.

Driving it externally

Two things are worth knowing if you script against the plugin or debug a newsroom integration.

Setting a field on one row

Fields inside a row — including the prototype column — are addressed through the scene's ControlObject with a `WITH` clause naming the list and the row index.

Set the prototype of the second row

```
<scene>*TREE*<container>*FUNCTION*ControlObject*in SET WITH items INDEX 1 ON
prototype SET proto2
```

`items` is the list's Field Identifier, `1` the zero-based row, and `prototype` the choice column's field name.

Replacing the whole list

A control client does not usually go row by row. It sends the complete list as one payload, naming a prototype and the field values for each row, and the plugin reconciles what exists against what arrived — adding, switching and removing rows as needed. Sending fewer rows than last time is how rows get removed.

`Reading values back`

Ask the scene for its control data rather than querying a row field directly. The scene's data returns one payload per row, each carrying that row's prototype and field values — which is also exactly what the operator's client sees.

When something looks wrong

- **The prototype dropdown is empty:** The scan has not run, or it found nothing. Press **Refresh**. If it stays empty, check that *Prototypes* really points at the group and that the prototypes are its direct children.
- **One prototype never appears in the list:** It has no ControlObject. Prototypes without one are skipped. Check the container name too — an unnamed container cannot be offered.
- **Every row comes out as the same prototype:** That is what adding does. Rows are created from the first prototype and changed afterwards with the dropdown. If the dropdown will not change them, the scan has not run.
- **I repointed *Prototypes* at another container and nothing changed:** If the new container cannot be found, the plugin keeps the prototypes it already had rather than emptying the list. Check the container is still in the scene, then press **Refresh**.
- **Adding rows stops before the number I set:** The maximum is being applied — but only if *Custom Maximum* is ticked. If it is off, the ceiling is 65,536 and something else is stopping you.

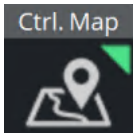
Copyright © 2026 Vizrt

- **A row lost its content after switching prototype:** Values are carried across by field identifier. If a field is named differently on the two prototypes, there is nowhere to put the value. Align the field identifiers across all prototypes.

See also:

Control List

Control Map



The Control Map plug-in binds the control of the Viz World Client plug-in XML property to a Control Object which allows an operator to use the Viz World Client Editor to select maps.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Map Properties

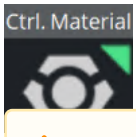
This plug-in does not have any properties or parameters except those that are common to all plug-ins.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Material



The Control Material plug-in exposes the material control to the Viz Trio user.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Material Properties

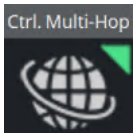
- **Material:** Sets the path to the material that should be used when the page is imported into the control client.
- **Deny Browsing Upwards:** Restricts the user from browsing for materials in folders other than the selected folder and its sub-folders when enabled.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Multihop



The Control Multihop plug-in works with the Multihop Editor (within the World Map Editor) to edit hops on a scene containing Control Multihop within Viz Trio or Viz Pilot.

Control Multihop should be placed next to Hops Manager plug-in.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Multihop Properties

This plug-in does not have any properties or parameters except those that are common to all plug-ins.

See Also

- Common Control Properties

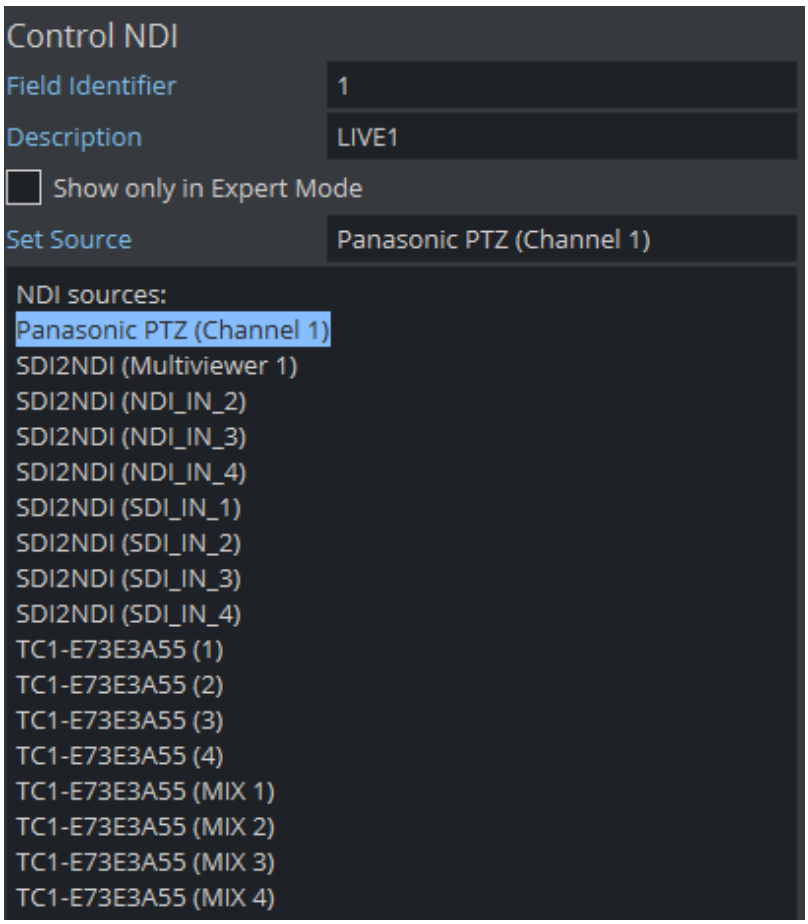
Copyright © 2026 Vizrt

Control NDI



The Control NDI plug-in allows to select and activate a certain NDI source available in your network from within Viz Artist or from a control application like Viz Trio.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

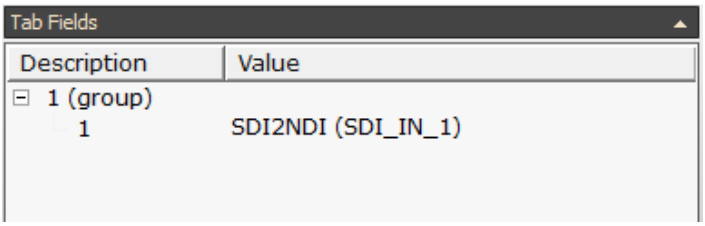


Control NDI exposes the NDI source selector to external applications like Viz Trio or Viz Pilot. An operator can set a new NDI available within the network which is then taken by Control NDI and set for the live input source. It also offers basic testing possibilities for the designer by providing a list of discovered sources. To switch to a new NDI source, copy the source from the NDI sources list and paste it into the **Set Source** field.

Please use the **Refresh** Button, if NDI sources do not show up immediately.

Copyright © 2026 Vizrt

To Use with Viz Trio



You need to manually enter a NDI source into the control field.

Information: Due some limitations on the Viz Trio side, it is currently not possible to do a discovery of sources and to provide a dropdown list.

See Also

- Common Control Properties
- Control Object

Control Num



The Control Number plug-in (also known as Control Num) is used to be able to decide how a number input is to be formatted.

It can be a value given by the control client user or by any external source.

Control Number should be used instead of Control Text when numbers are to be the input value. To make it work, add Control Number to the container that holds the text object.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Num Properties

- **Minimum Input:** Sets the minimum allowed input value.
- **Maximum Input:** Sets the maximum allowed input value.
- **Input Value:** Shows the current input value. Enter a value to test the number formatting.
- **Use shared memory placeholders:** See Shared Memory placeholders.
- **Show Verbatim:** Disables all formatting when enabled.
- **Thousand Separator:** Sets the required thousand separator for the output.
- **Plus Prefix:** Applies the prefix entered here in front of positive values.
- **Minus Prefix:** Applies the prefix entered here in front of negative values.
- **Suffix:** Applies the string entered here after all values.
- **Decimal Point:** Defines the decimal point for the output.
- **Decimal Places:** Sets the number of decimals that are shown by the output.
- **Leading Zero:** Places a zero in front of decimal values between `1` and `-1` when enabled.
- **Zero Replacement:** Sets the string to show instead of zero values.
- **Under replacement:** Sets the string to show instead of the minimum number.
- **Over replacement:** Sets the string to show instead of the maximum number.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Object



The Control Object plug-in provides control channels for subtrees of Viz Scenes.

To create a Viz Trio page template, a Viz Pilot template or a Viz Ticker template; each Scene must have at least one instance of Control Object. The control channel information is stored in the subtree itself, so the Scene subtree can be merged and stored as objects in the object pool. When a Control plug-in is added to a Scene, a Control Object plug-in is automatically added to the root container of the scene subtree to be controlled. Control Object stores and gathers field definitions which describe how values are processed and applied to various properties of the subtree. The easiest way to define fields is to place one or more Control plug-ins on containers within the subtree. These Control plug-ins find the Control Object and register properties related to the container they are placed on.

> **Note:** For Viz Ticker it is possible to have more than one Control Object plug-in. It should be on the root container for the tab objects in the scene.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

This page contains information on the following topics:

- Overview
- Control Object without Transition Logic
- Control Object with Transition Logic Enabled
- Control Object Properties
    - Transition Logic Off
    - Transition Logic On
    - Quick Start: Transition Logic Template
    - Parameter Reference
        - Basic
        - Advanced
        - Actions
    - The "in" Protocol
    - Manual Entry Format
        - Examples
    - Transition Logic Integration
        - Architecture
        - How the Take Works
        - Critical Naming Rules
        - Stop-Point Preservation
    - Namespace Propagation
    - GFX Subscene Support

Overview

ControlObject is the backbone of Vizrt's template-based broadcast workflow. It serves four primary roles:

1. **Field collection:** Discovers child Control* plug-ins (ControlText, ControlImage, ControlNum, etc.) and registers their editable fields into a unified field map.

Copyright © 2026 Vizrt

2. **Command protocol:** Exposes an `in` parameter that external applications (Viz Trio, Viz Pilot, Viz Multichannel, automation) use to read, write, list, and focus fields.
3. **VDF data model:** Generates an XML schema describing all fields, types, defaults, and metadata consumed by newsroom systems for template editing.
4. **Transition Logic integration:** Declares how the object scene should be loaded onto a master scene layer, which state to target, and which background scene to use.

Tip: In a Transition Logic workflow, the **master scene**(background scene) contains Toggle layers and directors. **Object scenes** (templates) each have a ControlObject on their root container that declares which layer and state they belong to.

Control Object without Transition Logic

In the simple scene based mode, Control Object should be put on the root container for the page tab-field containers. A description, which is used as template description in the control clients, should also be entered. To load the graphics in another rendering layer than the Middle (default), choose Front or Back. By using different layers up to three items can be on screen simultaneously without having to design a Transition Logic setup.

Note: Objects in the Back layer are rendered behind objects in the Middle and Front layer and so on, so the scenes must then be designed with that in mind.

Control Object with Transition Logic Enabled

With Transition Logic enabled, logical layer and state can be defined for the scene. These are used by the control clients to find out which transitions and animations to trigger when a page is put On Air or Off Air. These transitions and layers must all be defined in the background scene. A background scene name can also be set if a background scene other than the default (named default) is used.

Control Object Properties

Transition Logic Off

- **Description:** Provides the description of the object the scene subtree represents. This description is typically presented to users in GUI applications that allow the user to fill in data that should be presented by the controlled object.
- **Transition Logic:** Specifies whether the controlled object uses transition logic. Transition logic loads objects onto layers in a background scene, and runs animations to change the background scene layer into a required state.
- **Viz Layer:** Determines the Viz layer to load the scene which contains the scene tree which contains this Control Object. This parameter is only used when transition logic is set to `Off`.
- **Use All Directors:** Controls whether to play every director or only the default director in a non-transition logic scene, and takes effect when the Vizrt Media Sequencer triggers the `TAKE` command. When loading the scene into a control application such as Viz Trio, all directors in the stage start automatically when the scene is taken in. By changing this to `Off`, only directors with `Auto Start` enabled in the **Director Editor** start when the scene is taken in. This parameter is only used when **Transition Logic** is set to `Off`.

Copyright © 2026 Vizrt

Example: The scene design incorporates an animated background which starts automatically when the scene is loaded. To avoid the background animation restarting when the scene is played out, set **Use All Directors** to `Off` .

- **Show advanced:** See Control Object Advanced Properties.
- **Is Prototpye**: By default, the VDF data model automatically collect all properties from all loaded subscenes and exposes them to the control application.
  Sometimes you do not want this to be visible, for example if a scene is used as a prototpye or placeholder. Enabling this option will prevent any control object of such a scene inside a host scene when being used e.g. as Subscene.

This does not affect scenes being loaded from GFX or Superchannels

Transition Logic On

Quick Start: Transition Logic Template

1. Create an **object scene** with a root container named `object` .
2. Add **ControlObject** to the root container.
3. Set **Description** to a human-readable name (for example, "Lower Third – Breaking News").
4. Enable **Transition Logic**.
5. Set **Layer identifier** to match a Toggle container on your master scene (for example, `BannerLyr`).
6. Set **State identifier** to a stop-point on the layer director (for example, `LOWERTHIRD`).
7. Set **Background scene** to the master scene name (for example, `default`).
8. Add child containers with **ControlText**, **ControlImage**, etc.: each with a `field_id` . ControlObject auto-discovers them.

Parameter Reference

Basic

Parameter	Default	Description
Description	*empty*	Human-readable template description shown in Viz Trio / Viz Pilot GUI.
Transition Logic	Off	Enables Transition Logic mode. When on, reveals layer/state/background fields.
Layer identifier	*empty*	Must match a Toggle container name **and** its director on the master scene. Case-sensitive.

Copyright © 2026 Vizrt

Parameter	Default	Description
State identifier	*empty*	Stop-point name on the layer director to animate to (for example, `$LOWERTHIRD`).
Background scene	*empty*	Master scene name (name only, not a full path). Must not contain `/` or `\`.
Filter Continues	Off	Filters CONTINUE commands from Toggle, preventing unintended director advances.
Viz Layer	MIDDLE	Renderer layer for non-TL scenes: FRONT, MIDDLE, or BACK.
Use All Directors	On	When on, triggers the entire stage (all directors). When off, only the Default director plays.
Is Prototype	Off	Exposing parameters to a host scene if used as subscene

Advanced

Revealed when **Show advanced** is enabled.

Parameter	Default	Description
Enable GFX subscene	Off	Allows this ControlObject to discover and forward commands to ControlObjects in GFX channels.
Subscene field definition	Merge	How GFX subscene fields are combined: **Merge** (flat), **Merged subfield** (grouped under one prefix), or **Subfields** (per-channel prefix).
Subscene field id/ prefix	gfx	Prefix for subscene field IDs (for example, `gfx1`, `gfx2`).
Use custom duration	Off	Enables a custom display duration for interstitial/sponsor elements.
Duration	5.0	Display duration in seconds (only visible when custom duration is enabled).
Bind with Unreal Engine	Off	Maps field entries to Unreal Engine exposed ControlObjects.

Copyright © 2026 Vizrt

Parameter	Default	Description
Namespace prefix	*empty*	Prefix for propagating field IDs upward. Child field `1` becomes `Prefix.1` on a parent ControlObject.
Script (deprecated)	*empty*	Trio client script name. Deprecated, use Logicmaker instead.
Entries (manual)	*template*	Manually defined field entries in `key:location:type:min:max:maxchar:description` format.
Remap fields	Off	Gives the control clients the list of entries in a map. For use in scenes with nested control object plug-ins. This feature is mainly for advanced integration use. When `On`, Control Object gives the control clients the list of entries in map, rather than the list of entries registered by control plug-ins below the Control Object in the scene tree, and manually entered entries from entries. When actions are performed on fields, the Control Object uses map to rename the field identifier before dispatching the action.
Input command/ request	*empty*	The "in" protocol endpoint. Writing to this triggers command execution.
Result	*empty*	Read-only output of the last command.
Error	*empty*	Error output from the last command (max 64 KB).
Pending Result Chunks	0	Number of remaining result chunks for payloads exceeding 64 KB.

Actions

Action	Description
Store Stop Points	Serialises director stop-points into a hidden parameter so they survive object-pool merge.
Restore Stop Points	Recreates stop-points from stored data on the Toggle object director.
Refresh UE Control Objects	Reloads Unreal Engine ControlObject entries (only visible when Bind with Unreal is on).

Copyright © 2026 Vizrt

The "in" Protocol

External applications communicate with ControlObject by writing commands to the `in` parameter. The result is returned via `result`; errors via `error`.

Command	Description
`ON <field-id> SET <value>`	Set a field's value. Dispatches to the bound Control* plug-in or TCP location.
`ON <field-id> GET`	Read a field's current value.
`ON <field-id> FOCUS`	Move the GUI focus to the field's container (useful for interactive editing).
`LIST` / `LIST_ALL`	Return all field definitions in `key:location:type:…` format.
`GET_MODEL`	Return the full VDF XML data model (schema + scene data + directors + TL metadata).
`SET_MODEL <xml>`	Apply a VDF payload XML to set multiple fields at once.
`HIT <x> <y>`	Returns the field ID for the container at the given screen coordinates.

> **Chunked I/O**
> For payloads exceeding 64 KB, the protocol supports chunked transfer. Send commands with `CHUNK_START`, `CHUNK`, `CHUNK_END` prefixes. Results larger than 64 KB are split into chunks retrieved via the *Get Next Result Chunk* action, monitored through *Pending Result Chunks*.

Manual Entry Format

The **Entries** text field supports manually defined fields using this format:

```
key:location:type:min:max:maxchar:description:aux_field
```

Examples

```
# Text field bound to a geometry text node
votes:$top$label*GEOM*TEXT:integer:0:100:8:Percentage of votes

# Text field with no constraints
```

Copyright © 2026 Vizrt

```
name:$bottom$name*GEOM*TEXT:::::Name of person

# Float field via direct TCP path
zoom:MAIN_SCENE*TREE*$Browser*FUNCTION*Browser*zoom:float:0.0:-10.0:10.0:Zoom level

# Using control channels (@-addressing)
zoom:MAIN_SCENE*TREE*@zoom:float:0.0:-10.0:10.0:Zoom level
```

Transition Logic Integration

Architecture

A Transition Logic broadcast setup involves two scene types:

- **Master scene** (background scene): Contains Toggle containers and directors for each layer (for example, `BannerLyr`, `TickerLyr`, `ClockLyr`).
- **Object scenes** (templates): Each has a ControlObject on its root that declares which layer and state it targets.

How the Take Works

1. Newsroom app (Trio/Pilot) sends data via the `in` protocol.
2. The object scene is merged into the master scene's Toggle layer.
3. Toggle performs an A/B swap, animating the old graphic out and the new one in.
4. The layer director executes `GOTO_TRIO` from the current stop-point to the state declared by the ControlObject's **State identifier**.

Critical Naming Rules

> ⚠ **Names must match exactly**
>
> - **Layer identifier** must match both the Toggle container name *and* its director name on the master scene.
> - **State identifier** must match a stop-point (`$` -prefixed) on that layer director.
> - **Background scene** is the scene *name only* — not a full Graphic Hub path.
> - All names are **case-sensitive**.

Stop-Point Preservation

When an object scene is merged into the object pool, director stop-points are discarded. Use **Store Stop Points** to serialise them into ControlObject before saving. On load, **Restore Stop Points** recreates them on the Toggle's object director (`<LayerName>O`).

Namespace Propagation

When the **Namespace prefix** is set (for example, `Header`), field IDs propagate upward through parent ControlObjects or ControlLists. A child field with ID `1` becomes accessible as `Header.1` on the parent.

Copyright © 2026 Vizrt

When namespace is **empty**, the ControlObject acts as a *registration boundary*. Field IDs do not propagate upward, and Toggle treats it as a self-contained template.

GFX Subscene Support

When **Enable GFX subscene** is on, ControlObject discovers ControlObjects in active GFX channels and forwards commands to them. Three merge modes control how subscene fields appear:

Mode	Behavior
Merge	Subscene fields are merged flat into the parent's field list.
Merged subfield	Fields are grouped under a single prefix (set by *Subscene field id/prefix*).
Subfields	Each GFX channel gets its own prefix (`gfx1` , `gfx2` , etc.).

See Also

- Common Control Properties
- Toggle

Copyright © 2026 Vizrt

Control Omo



The Control Object Moving (Control Omo) plug-in gives the possibility to add a group of containers and reveal one at the time.

This is done by adding Omo on the root container and in its editor specify which of the Sub-Containers to show. The Omo value can be made accessible for the control client by adding Control Omo on the same container as Omo.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Omo Properties

- **Minimum Value:** Sets the minimum input value.
- **Maximum Value:** Sets the maximum input value.

See Also

- Common Control Properties
- Control Object
- Omo

Copyright © 2026 Vizrt

Control Parameter



The Control Parameter plug-in works with function plug-ins, and is a very flexible plug-in that can be used to control properties that are not exposed in a specific plug-in, and requires some knowledge in the Viz Artist command language. The parameter to control must be identified by its command path relative to the container Control Parameter is located at. To find the correct command path, open the Show Commands window (button in the lower left corner in Viz Artist) and make changes to the parameter to control. The command path is shown among the commands. For instance if the value to be changed is the alpha value in an alpha editor the command window shows the following: `receive <-1 SCENE*noname*TREE*#805*ALPHA*ALPHA SET 99.0>`

#805 is the internal name for the container and the part needed to expose the alpha parameter (`ALPHA * ALPHA`). The `SET` part is added by Control Parameter. To expose the alpha parameter for a container write `ALPHA * ALPHA` in the Parameter section in the plug-in and then set the correct data type which in this case is “Float”.

Another example would be to use Control Parameter to expose and control additional values like controlling both Pie Max and Max Value. These parameters are dependent on each other, but both values are not exposed unless the Control Parameter is used. Adjusting the Pie Max value gives the following: `receive <5679 SCENE*noname*TREE*#378*FUNCTION*PieValues*DATA GET>`

In this case, the `FUNCTION*PieValues*DATA` is the parameter that exposes the object property *Pie Max* so it can be controlled through a template.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Parameter Properties

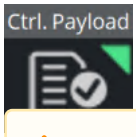
- **Parameter:** Sets the parameter path to the property to enable control for.
- **Input Value:** Shows the current input value.
- **Data Type:** Sets correct data type depending on the type of parameter to enable control for.
- **Normalize:** Enables normalization options to normalize the input range for Control applications. This setting is only active if the Datatype is a number (Integer or Float).
    - **Minimum Input:** Sets the minimum allowed input value.
    - **Maximum Input:** Sets the maximum allowed input value.
    - **Minimum Value:** Sets the minimum value, which is set relative to the minimum input value.
    - **Maximum Value:** Sets the maximum value, which is set relative to the maximum input value.
- **Enclose in Quotes:** Adds quotes to the input value (for example, if you want to set a clip with spaces in the pathname). This setting is only active if the Datatype is not a number.
- **Prefix:** Adds a prefix to all values that are sent to the renderer. This can be used to show specific text or it can be used to add a path to an object pool for instance.
- **Control Parent:** Controls the parent container when enabled.
- **Control Scene:** Control Scene parameters.
- Shared Memory placeholders

See Also

Copyright © 2026 Vizrt

- Common Control Properties
- Control Object

Control Payload



The Control Payload plug-in distributes a whole payload to children Control plug-ins.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Example: Social TV (Feed Streamer) uses this plug-in to send payload fields from a Data Hub to Viz Engine.

If the payload value is empty, the Container is hidden.

Control Payload Properties

- **Text Entry Box:** Data from external application (for example, Viz Trio or Social TV) to send the payload.

Example:

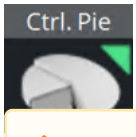
```
<payload xmlns="http://www.vizrt.com/types">
    <field name="topic">
        <value>It\qs Valentine\qs Day, how you doin\q? </value>
    </field>
    <field name="option">
        <value>Hate it! Worst holiday ever|Love it! Wish it were every day|
Thanks for the reminder, need to buy roses</value>
    </field>
    <field name="count">
        <value>238|189|46</value>
    </field>
</payload>
```

See Also

- Common Control Properties

Copyright © 2026 Vizrt

Control Pie



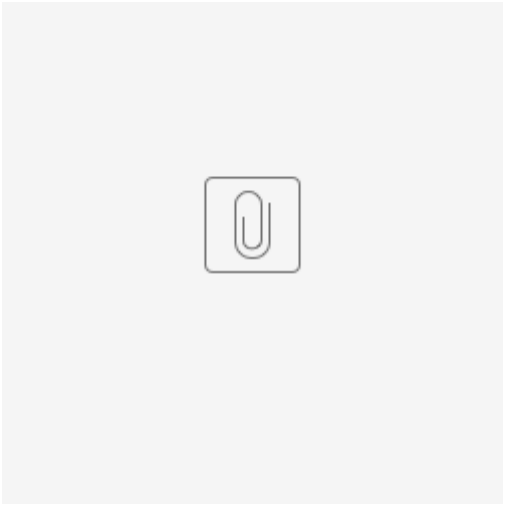
The Control Pie plug-in is able to bind a tab-field to values from Pie Slice plug-ins.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Pie Properties

- **First slice ID:** Sets the tab-field number for the first slice (the first container below the group that holds Control Pie).
- **Last Slice ID:** Sets the tab-field number for the last slice (the difference between Last and First should correspond to the total number of slices in the pie chart).
- **Expose Pie Max/Total Value:** Allows a Viz Trio user to set Max value for a pie slice and Max/Total value for the whole pie chart when enabled.

To Build a Pie Chart



1. Add a group container and name it, for instance `pie`. Add Pie Values and Control Pie to the group.
2. Add a Cylinder object as a Sub-Container to the group. Add material and the presenter plug-in Pie Slice to the new container. Name the container `slice` or similar. The Cylinder object is not visible.
3. Open the Pie Values editor and increase the Value0 property a bit. The Cylinder becomes visible and its size changes.
4. Open the Cylinder editor and set Center to `Top`.
5. Open the Transformation Editor for the pie group, and rotate the pie group around 90 degrees on the X-axis to make the cylinder top face the camera.
6. Add a font as Sub-Container to the slice container, and add a material to make it visible against the background.

Copyright © 2026 Vizrt

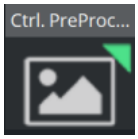
7. Open the Text Editor and set the horizontal justification to Center. Open the Transformation Editor and rotate the font around the X-axis to -90 degrees and set Y position to `1.0`.
8. Scale the font down a bit so it fits over the slice (for example `0.2`).
9. On the slice container, open the Pie Slice editor and set Control Text Values to a data type (for example, *None*, *Integer*, *Float* or *Formatted*). By choosing Integer or Float the slice text changes, which shows the value for the slice.
10. In the Pie Slice editor, adjust the Text Offset to position the value label relative to the center of the pie (for example `70.0`).
11. A single pie slice should be ready. Make the number of slices needed by making copies of the slice container and change their material so they can be differentiated from each other.
12. Set the different values in the Pie Values editor to test.

See Also

- Control Object
- Pie Slice
- Pie Values
- Common Control Properties

Copyright © 2026 Vizrt

Control PreProcessing Texture Renderer



The Control PreProcessing Texture Renderer modifies the setting for the Blur Texture Renderer.

The Blur Texture Renderer allows for an efficient GPU based blurring of textures.

If these effects need to be controlled by a control application, the Control PreProcessing Texture Renderer plug-in is required.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control PreProcessing Texture Renderer Properties

- **Deny Browsing Upwards**- Restricts the user from browsing for images/textures in folders other than the selected folder and its sub-folders when enabled.
- **Image location Prefix:** If the full path to a folder is entered, the image/texture location exposed to the control clients can be a simple string, instead of a full path. This is relevant if connected to a newsroom system. For example if a user wants edit a text and an image/texture on the same tab-field by just entering a text string. This is relevant if a user has a folder containing flag images/textures, and the control client receives “US” from the external system it loads the image/texture with the path: “image location prefix + US”, which then typically would be the “Stars and stripes” (flag of the U.S.A).
- **Input Value:** Shows the current value for the plug-in. It is not necessary to set any value here. It is used for debugging purposes. You can also use Shared Memory placeholders.
- **Default Image:** Usually the image which has already been assigned when applying the plug-in. It acts as a default if no value is set.
- **Blur Factor:** The amount of strength the blur effect is applied to the image.

Configured **Image location Prefix** written without prefix “IMAGE*/” loads images correctly but is never removed from Trio field values.

5%	

Copyright © 2026 Vizrt

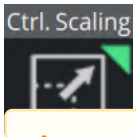
10%	
25%	

See Also

- Control Object
- Common Control Properties

Copyright © 2026 Vizrt

Control Scaling



The Control Scaling plug-in enables the Viz Trio user to edit transformation scaling values for objects. Normalization between input and output values can be set by parameters.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Scaling Properties

- **Minimum & Maximum Input:** Sets minimum and maximum allowed input values.
- **Minimum Scaling:** Sets the minimum transformation value which is set relative to the minimum input value.
- **Maximum Scaling:** Sets the maximum transformation value which is set relative to the maximum input value.
- **Control X, Y, Z**- Controls the X, Y, and Z-axis independently when enabled. The Viz Template Wizard TripletEditor component has three *onChange* events available that accommodates for these parameters.
- **Input Value:** Shows the current input value. Enter a value to test the relation of max/min. input and scale values.
- **Stop Key Frame:** Forces the scaling values entered to be put into the end Key Frame for the animation if the scaled object is animated. For instance if a user is to create a bar graph that animates from zero to a value specified, then this is the case. Name the end Key Frame in the Viz Artist stage and enter that name in the “stop Key Frame” field in the control scaling plug-in.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Sign Container



The Control Sign Container plug-in value can show one of three containers based on the value of a numerical input.

The container that holds Control Sign Container must have three Sub-Containers:

- A negative value triggers to show the first container.
- A positive value triggers to show the second container.
- A zero value triggers to show the third container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Sign Container Properties

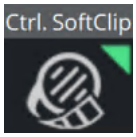
- **Input Value:** Shows the current input value. Enter a value here to test that the correct Sub-Container is shown.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control SoftClip



The Control SoftClip plug-in allows the user to control and browse for video clips played in Viz Artist using the Soft Clip plug-in.

Add the Control SoftClip to the same container as the SoftClip plug-in. Further, specify a Field Identifier and a Description.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control SoftClip Properties

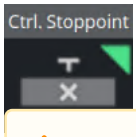
This plug-in does not have any properties or parameters except those that are common to all plug-ins.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Stoppoint



The Control Stoppoint plug-in enables control of the time value on a stop point.

This can typically be used for setting the length of an animation.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Stoppoint Properties

- **Stoppoint:** Sets the name of the stop point to control.
- **Director:** Sets the name of the animation director where the stop point is placed.
- **Stop Time:** Sets the stop time in seconds. This is the value that is set by the Viz Trio operator.

Note: This Control plug-in does not work with transition logic based scenes.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Text



The Control Text plug-in binds the control of Text or Classic Text to a Control Object depending on the container geometry.

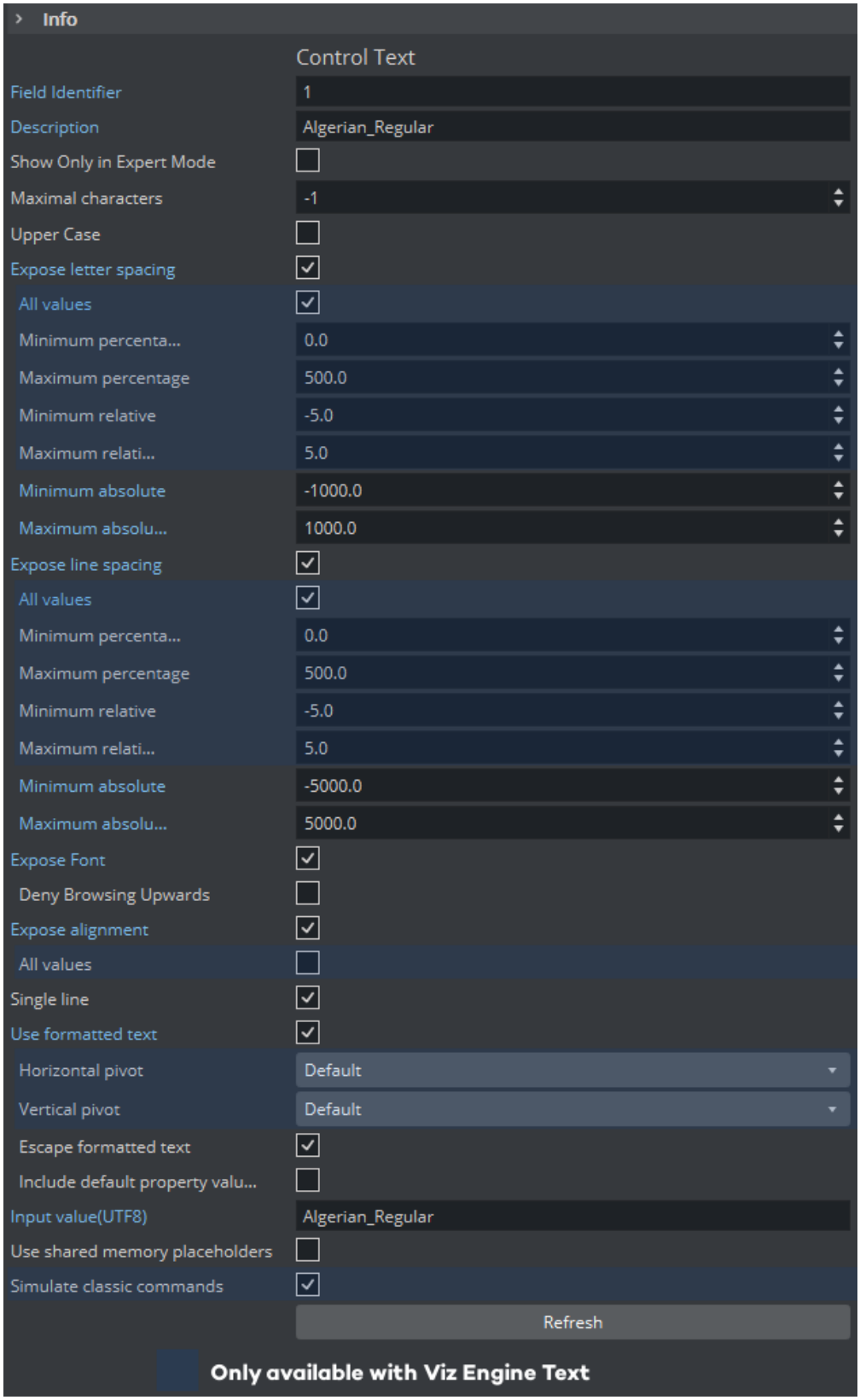
If neither Text or Classic Text is in the container, the Control Text plug-in uses the scene setting to determine which render engine to use.

This plug-in expects the input value in UTF-8 encoding.

⚠ This plug-in is located in: Plugins -> Container plug-ins -> Control

Copyright © 2026 Vizrt

Control Text Properties



Copyright © 2026 Vizrt

- **Maximal characters:** Sets the maximum number of characters which can be typed into the text field.
- **Upper Case:** Converts lower case input into upper case when set to `On`.
- **Expose letter spacing:** Allows changes to the letter spacing parameters and sets the minimum and maximum value when set to `On`. The exposed field is **kerning**.
    - **All values [*1]:** Exposes two more values of letter spacing, percentage and extended relative instead of just extended absolute. The exposed field is changed to a group of three sub-fields:
        - **letter_spacing.percentage**, **Minimum** and **Maximum** value can be set.
        - **letter_spacing.extend_rel**, **Minimum** and **Maximum** value can be set.
        - **letter_spacing.extend_abs**, **Minimum** and **Maximum** value can be set.
- **Expose line spacing:** Allows changes to the line spacing parameters and set the minimum and maximum value when set to `On`. The exposed field is **line_spacing**.
    - **All values [*1]:** Exposes two more values of line spacing, percentage and extended relative instead of just extended absolute. The exposed field is changed to a group of three sub-fields:
        - **line_spacing.percentage**, **Minimum** and **Maximum** value can be set.
        - **line_spacing.extend_rel**, **Minimum** and **Maximum** value can be set.
        - **line_spacing.extend_abs**, **Minimum** and **Maximum** value can be set.
- **Expose Font:** Allows selects another Font when set to `On`. The exposed field is **font** on Classic Text, or **font_face** on Viz Engine Text.
    - **Deny Browing Upward:**
- **Expose alignment:** Allows changes to the text alignment when set to On. The exposed field is **justification**.
    - **All values [*1]:** Exposes both horizontal and vertical alignment. The exposed field is changed to a group of two sub-fields:
        - **alignment.horizontal**
        - **alignment.vertical**
- **Single Line:** Converts multi-line input into a single line when set to `On`.
- **Use Formatted Text:** Enables use of character specific formatting in supported client applications. Character specific formatting includes bold, italic, subscript, superscript, kerning, position, rotation, scaling, alpha, and color(RGB) in Classic Text, modifier, font size, color(ARGB) in Text. Formatted text cannot be used together with Text FX Write.
    - **Horizontal Pivot** and **Vertical Pivot:** Set transformation pivot of characters in **Formatted Text Wrapper** with this formatted text `<fo:wrapper rotate-z="-45">M</fo:wrapper> <fo:wrapper scale-x="2" scale-y="2">N</fo:wrapper>`. Set Horizontal Pivot to *Center* and Vertical Pivot to *Middle* rotates ‘M’ and scales 'N’ at the original center of each character.
    - To use the characters `<` and `>` correctly in Viz Multichannel, make sure that **Use Formatted Text** is set to `OFF`.
    - Character specific formatting is not supported for Viz Pilot.
    - In Viz Trio, the operator enter a required formatting mode via an assigned shortcut key. To change the format of one or more characters, hold down **ALT** and toggle the right and left arrow keys to shift mode. See the Viz Trio User Guide for more information on Character Formatting.
- **Escape formatted text:** Simplifies formatted text output. Use XML escape characters instead of XML <![CDATA[]]> and not add an XML root element. Use `<fo:wrapper/>` element only with formatted words and characters. Control Text still accepts both full formatted and escape formatted text input independent of this property.

Copyright © 2026 Vizrt

- **Include default property values:** Adds all available properties of formatted text output even all values are defaults. This guarantees all supported formats by Control Text can be transferred to another Text via XML.
- **Input Value / Input Value(UTF8):** Shows the current value for the plug-in. It is not necessary to set any value here. It is used for debugging purposes.
- **Use shared memory placeholders:** See Shared Memory placeholders.
- **Simulate classic commands*1:** Allows Viz Trio uses Classic Text commands via Control Object to control styles of Viz Engine Text.

⚠ ***1:** only available with Viz Engine Text

Formatted Text Wrapper

Format single or multiple characters by cover them with <fo:wrapper /> element and add any or multiple of these attribute attributes.

All Available Formatters

	XML Attribute	Data Type	Min	Max	Note	Example
kerning	letter-spacing	float	-	-	• Existing value of text is not changed if no new value of these modifiers is specified. • This formatter affects space between covered and next and all followed characters. • From this example. "END" is moved to the left side of another ("TEST").	<fo:wrapper letter-spacing="80">T</fo:wrapper> <fo:wrapper letter-spacing="40">E</fo:wrapper> <fo:wrapper letter-spacing="10">S</fo:wrapper> <fo:wrapper letter-spacing="-360">T</fo:wrapper>END
color	color	3 of ubyte	#000000	#FFFFFF	Contains value of red, green and blue in hexadecimal from 00 to FF lead by #: • #FF0000: brightest red • #00FF00: brightest green • #0000FF: brightest blue	<fo:wrapper color="#FF0000">T</fo:wrapper> <fo:wrapper color="#00FF00">E</fo:wrapper> <fo:wrapper color="#0000FF">S</fo:wrapper> <fo:wrapper color="#000000">T</fo:wrapper>

Copyright © 2026 Vizrt

	XML Attribute	Data Type	Min	Max	Note	Example
underline	underline	boolean	true/false		• Existing value of text is not changed if no new value of these modifiers is specified.	<fo:wrapper underline="true">TEST</fo:wrapper>
scale	scale-x	float	0.0	-	• Should be only positive value otherwise the backface is on the front side and be invisible until enable "Back-Face" drawing in Expert plug-in.	<fo:wrapper scale-x="3.0" scale-y="3.0">T</fo:wrapper> <fo:wrapper scale-x="2.0" scale-y="2.0">E</fo:wrapper> <fo:wrapper scale-x="1.0" scale-y="1.0">S</fo:wrapper> <fo:wrapper scale-x="0.4" scale-y="0.4">T</fo:wrapper>
	scale-y	float	0.0	-		
	scale-z	float	0.0	-		
position	pos-x	float	-	-	• Relates to default position of each character.	<fo:wrapper pos-y="-30">T</fo:wrapper> <fo:wrapper pos-y="-10">E</fo:wrapper> <fo:wrapper pos-y="10">S</fo:wrapper> <fo:wrapper pos-y="30">T</fo:wrapper>
	pos-y	float	-	-		
	pos-z	float	-	-		
rotation	rotate-x	float	-	-	• One rotation is 360 (counterclockwise) or -360 (clockwise).	<fo:wrapper rotate-x="15">T</fo:wrapper> <fo:wrapper rotate-x="30">E</fo:wrapper> <fo:wrapper rotate-x="45">S</fo:wrapper> <fo:wrapper rotate-x="60">T</fo:wrapper>
	rotate-y	float	-	-		
	rotate-z	float	-	-		

Available Classic Text Formatters

Copyright © 2026 Vizrt

	XML Attribute	Data Type	Min	Max	Note	Example
alpha	opacity	float	0	100	• Existing value of text is not changed if no new value of these modifiers is specified. • Subscript and superscript cannot be true at the same time.	<fo:wrapper opacity="80">T</fo:wrapper> <fo:wrapper opacity="60">E</fo:wrapper> <fo:wrapper opacity="40">S</fo:wrapper> <fo:wrapper opacity="20">T</fo:wrapper>
	bold	boolean	true/false			<fo:wrapper bold="true">TEST</fo:wrapper>
	italic	boolean	true/false			<fo:wrapper italic="true">TEST</fo:wrapper>
	subscript	boolean	true/false			<fo:wrapper subscript="true" superscript="false">TEST</fo:wrapper>
	superscript	boolean	true/false			<fo:wrapper subscript="false" superscript="true">TEST</fo:wrapper>
font	font	FONT				<fo:wrapper font="<00AF0E0D-625C-184B-98AE54ADA55B6239>">TEST</fo:wrapper> <fo:wrapper font="FONT*Fonts/JS-Wansika-Italic">TEST</fo:wrapper> <fo:wrapper font="Fonts/JS-Wansika-Italic">TEST</fo:wrapper>

Available Text Formatters

Copyright © 2026 Vizrt

	XML Attribute	Data Type	Min	Max	Note	Example
color	color	3-4 of ubyte	#00000000	#FFFFFFFF	Contains value of alpha (optional), red, green and blue in hexadecimal from 00 to FF lead by #: • #FF0000: brightest red • #00FF00: brightest green • #0000FF: brightest blue • #00000000: transparent (zero alpha) • #FF000000: opaque (highest alpha) Default alpha value is #FF if not specified	<fo:wrapper color="#FF0000">T</fo:wrapper> <fo:wrapper color="#00FF00">E</fo:wrapper> <fo:wrapper color="#0000FF">S</fo:wrapper> <fo:wrapper color="#000000">T</fo:wrapper> <fo:wrapper color="#40808080">O</fo:wrapper> <fo:wrapper color="#80808080">P</fo:wrapper> <fo:wrapper color="#B0808080">A</fo:wrapper> <fo:wrapper color="#D0808080">Q</fo:wrapper>
modifier	modifier	modifier	subscript/superscript/none		• Existing value of text is not changed if no new value of these modifiers is specified.	<fo:wrapper modifier="subscript">TEST</fo:wrapper> <fo:wrapper modifier="superscript">TEST</fo:wrapper> <fo:wrapper modifier="none">TEST</fo:wrapper>
font	font	FONT_FACE				<fo:wrapper font="<41E57305-1205-4742-96DA817BCA632729>">TEST</fo:wrapper> <fo:wrapper font="FONT_FACE*Fonts/JS_Wansika_Italic">TEST</fo:wrapper> <fo:wrapper font="Fonts/JS_Wansika_Italic">TEST</fo:wrapper>

<fo:wrapper /> element can be covered by another such as:

Copyright © 2026 Vizrt

Formatters	<fo:wrapper bold="true" color="#ff0000"> Th<fo:wrapper color="#ffffff"> a<fo:wrapper color="#0000ff"> il</fo:wrapper> a</fo:wrapper> nd </fo:wrapper>
Result	Th a il a nd

Entities

Support three entities in formatted text for preventing broken formatter XML.

Character	Entity
Character	**Entity**
<	<
>	>
&	&

Example

Formatter	<fo:wrapper color="#FF0000"><sender>John & Smith</sender></fo:wrapper>
Result	<sender>John & Smith</sender>

CDATA Section

CData section can be used for preventing broken formatter XML too, but with more readable text.

Can be used to send plain text into Control Classic Text plug-in with enabled "Use formatted text" but many existing formats such as bold, italic, underline subscript and superscript in current text still are kept.

Example

Formatter	<fo:wrapper color="#ff0000"><![CDATA[<sender>John & Smith</sender>]]></fo:wrapper>
Result	<sender>John & Smith</sender>

Can rest all existing formats by assigning those new values, but the plain text should not contain **"]]>"**.

Example with Classic Text

Copyright © 2026 Vizrt

```
<fo:wrapper letter-spacing="0" color="#000000" opacity="1 0 0" bold="false" italic="false"
underline="false" subscript="false" superscript="false"
font="<00AF0E0D-625C-184B-98AE54ADA55B6239>" ><![CDATA[Any plain text]]></fo:wrapper>
```

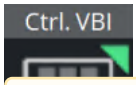
Example with Text

```
<fo:wrapper letter-spacing="0" color="#000000" underline="false" modifier="none"
font="<41E57305-1205-4742-96DA817BCA632729>" ><![CDATA[Any plain text]]></fo:wrapper>
```

See Also

- Common Control Properties
- Control Object

Control VBI



The Control VBI plug-in binds control of scene VBI source to a Control Object.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

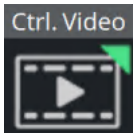
Control VBI Properties

- **VBI Source:** Determines the VBI source to bind.

See Also

- Common Control Properties

Control Video



The Control Video plug-in exposes control over a video codec channel.

Add Control Video to give the Client Application operator and editors (like the Media Search in Viz Trio or Viz Pilot) the ability to search Viz One for video clips.

Control Video works in combination with the Video Clip plug-in. This allows you to merge Key Frames with the object for Transition Logic scenes.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control Video Properties

- **Loop Clip:** Enables or disables continuous looping for a video clip in graphics.
- **Auto Play:** Starts playout on scene load.
- **Toggle Channel:** Enables toggling between two clips in a Transition Logic scene. Enable to create a smooth transition between two clips when loading a new clip.

> **Information:** To use the Toggle Channel option in Control Video plug-in, several options in other plug-ins need to be enabled:
>
> 1. In the master scene, turn on "Control video" in Toggle plug-in.
> 2. In the object scene:
>   - Add a Video Clip plug-in and select Texture target.
>   - Add a Control Video plug-in and turn on Toggle Channel.

- **Codec Channel:** Sets the codec channel for the video board in use:
  - **1:** Is normally used as the default codec channel for full screen video.
  - **2:** Is normally used as the default codec channel for video in graphics.
- **Global Clip:** Plays the selected clip as a Media Asset, instead of adding the clip to the stage when set to `On`.

> **Tip:** Set **Global Clip** to `Off` to access the **Director** and **Clip Key Frame** fields.

- **Director:** Specifies the name of the director that the clip loads to.
- **Clip Key Frame:** Specifies the name of the Key Frame the clip loads to.
- **Viz One:** Transfers Clips From Viz One.
- **Use CUE:** Cues up clips by name only, and does not affect the involved clip channel. Requires **Global Clip** to be `Off`.
- **Expose cuing field:** Setting this to `On` lets client applications control the **Use CUE** option through the sub-field `cue`. Requires **Global Clip** to be `Off`.

To Use Control Video with Viz One

To access, transfer and play clips from Viz One systems, open the Viz One Browser dock and drag and drop your clip directly into the *Clip* field.

Copyright © 2026 Vizrt

To Run the Clip at a Specific Key Frame on a Specific Director

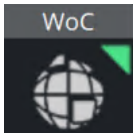
1. Set **Global Clip** to `Off`.
2. Enter the name of the director that controls playout of the clip in the **Director** field.
3. In the **Key Frame** field, enter the name of the Key Frame that triggers playout.

See Also

- Common Control Properties
- Control Object
- Video Clip

Copyright © 2026 Vizrt

Control World



The Control World plug-in allows you to expose a set of properties to the operator.

Control World is a replacement for Control Map with many more options and on-the-fly feedback from Viz Artist/Engine.

> **Note:** Control World and Control Map plug-ins are not compatible. In Template Wizard versions 8.6 and lower, an error message pops up the first time you browse for a map set up with Control World plug-in.

The control can expose different fields based on the container it resides on. When tabbing to the control (in a navigator scene) the camera jumps to map location and all feedback (exact camera position) is immediate.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Control

Control World Properties

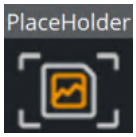
- **Client Mode:** Sets the client mode to Full or Simple. When selecting a map, this opens the full or simple map client in the control application.
- **Input:** Sets the exposed properties. The field is based on the input from NavFinder; however, you may also manually change this field.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Placeholder



The Placeholder plug-in is used to mark placeholder containers when designing library objects for Viz Trio compositing.

A placeholder container is available as a parent to other compositing objects in the scene tree.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Control

Placeholder Properties

- **Sort Order:** Integral sorting order identifier (lower number is sorted first).
- **Description:** A description of the tab field or editable object. This is used as a description for the items when used in the control clients.

See Also

- Common Control Properties
- Control Object

Copyright © 2026 Vizrt

Control Texture Slots



The Control Texture Slots plug-in exposes the slots of a Texture Slot plug-in as image controls in the control clients.

Control Texture Slots must be placed on the same container as the Texture Slot plug-in whose slots it controls.

All exposed slots are grouped under one field (the Field Identifier), with one sub-field per slot, similar to Control Container. This lets one control plug-in drive several texture slots, for example a background image and a logo, from one template.

Note: *This plug-in is located in: Plugins → Container plug-ins → Control*

Control Texture Slots Properties

- **Exposed Slots (empty = all):** A comma separated list of the slot names to expose, for example `Overlay,Logo`. Every listed slot that exists in the Texture Slot plug-in becomes a sub-field. The slots appear in the order of this list. If the list is empty, all slots are exposed in the order of the Texture Slot plug-in. Names that do not match a slot are skipped and a warning is written to the console.
- **Expose UV Offset:** Allows the user to change the UV offset (position) of the image in each exposed slot.
- **Expose UV Scaling:** Allows the user to change the UV scaling of the image in each exposed slot.
- **Image location Prefix:** If the full path to a folder is entered, the image location exposed to the control clients can be a simple string instead of a full path. Works the same way as in Control Image: if the prefix points to a folder of flag images and the control client sends `US`, the image is loaded from "image location prefix + US". Full paths sent by the control client are used as they are and ignore the prefix.
- **Use shared memory placeholders:** Exposes the slot images as text instead of image fields. The text may contain shared memory placeholders such as `Logos/${team|default}`, the same as in Control Image. The images are updated automatically whenever a used shared memory variable changes. See Shared Memory placeholders.
- **Refresh:** Re-reads the slots of the Texture Slot plug-in and updates the exposed fields. Use it after adding, renaming or removing slots in the Texture Slot plug-in. The fields are also updated whenever a property of Control Texture Slots changes.

Copyright © 2026 Vizrt

Configured **Image location Prefix** written without prefix “IMAGE*/” loads images correctly but is never removed from Trio field values.

Exposed Fields

For each exposed slot the following fields are created, where `<field_id>` is the Field Identifier and `<slot>` is the slot name:

Field	Type	Description	Created when
`<field_id>.<slot>`	Image (Text with shared memory placeholders)	The image of the slot. An empty value clears the image.	Always
`<field_id>.<slot>.image_position`	Duplet (U V)	UV offset of the slot, range -1000 to 1000.	Expose UV Offset is enabled
`<field_id>.<slot>.image_scaling`	Duplet (U V)	UV scaling of the slot, range 0 to 1000.	Expose UV Scaling is enabled

The slot name is used as the label of the field in the control client.

> ***Note:*** *In the field identifier, any character of the slot name other than letters, digits,* `_` *and* `-` *is replaced by* `_`*. For example, the slot* `Lower Third` *is exposed as* `<field_id>.Lower_Third`*. If two slot names map to the same field identifier, only the first one is exposed.*

If an image cannot be found, an error is reported on the Control Object.

Example

A Texture Slot plug-in has the slots `Background`, `Overlay` and `Logo`. Control Texture Slots is set up with:

- Field Identifier: `slots`
- Exposed Slots: `Overlay,Logo`
- Expose UV Offset: enabled

The control client then shows the fields:

- `slots.Overlay`
- `slots.Overlay.image_position`
- `slots.Logo`
- `slots.Logo.image_position`

The `Background` slot is not exposed.

Copyright © 2026 Vizrt

See Also

- Texture Slot
- Control Image
- Common Control Properties
- Control Object

2.4.4 Default Container Plug-ins

The following Container plug-ins are located in the Default folder:

- Magnifier Controller
- Screen2World
- Trio Scroll Element

Copyright © 2026 Vizrt

Magnifier Controller



The Magnifier Controller plug-in gives control over the magnifier from inside a plug-in. When the controller is assigned to a container, the magnifier is applied to this container. The magnifier is positioned and scaled according to the positional and scale values of the host container. The plug-in additionally provides access to the magnifier properties **Scale**, **Ellipsis** and **Alpha**.

Please observe that only one Magnifier Control plug-in may be active within a scene at any given time. Because of this, if there are several instances of the Magnifier Controller, the magnifier is only applied to the last instance activated. However, when the currently applied controller is deactivated, the plug-in automatically applies the magnifier to the next container with an active controller plug-in. The magnifier is fully deactivated only when there are no active controllers left.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Default

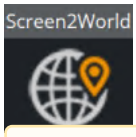
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Magnifier Controller Properties

- **Scale:** Sets the scale of magnification. The minimum value is `1.0`, the maximum value is `100.0`. The default value is `2.0`.
- **Ellipsis:** Creates an ellipsis effect, mimicking the distortion created by a concave or convex lens. The minimum value is `-10.0`, the maximum value is `10.0`. The default value is `0.0`.
- **Alpha:** Sets the alpha value for the magnification. The minimum value is `0.0`, the maximum value is `100.0`. The default value is `100.0`.
- **Active:** Enables or disables magnification. Can be set to `On` or `Off`. By default, this is set to `On`.

Copyright © 2026 Vizrt

Screen2World



The Screen2World plug-in converts the passed screen positions X and Y to world positions X, Y and Z, and sets them for the container the plug-in is applied to.

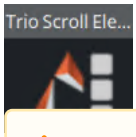
Note: This plug-in is located in: Plugins -> Container plug-ins -> Default

Screen2World Properties

- **Screen Position X:** Sets the X value for the screen position. Accepted values are floating point numbers ranging from `-100000.0` to `100000.0`. The default value is `0.0`.
- **Screen Position Y:** Sets the Y value for the screen position. Accepted values are floating point numbers ranging from `-100000.0` to `100000.0`. The default value is `0.0`.

Copyright © 2026 Vizrt

Trio Scroll Element



The Trio Scroll Element plug-in is used for data storage by the Trio Scroll Geometry plug-in.

It is automatically added to each base element of a Viz Trio scroll scene.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Default

Trio Scroll Element Properties

- **Prototype:** Sets the path and the name of the scene.
- **Extra space before element:** Sets the space before the element.
- **Extra space after element:** Sets the space after the element.
- **Easepoint 0-4 enabled:** Enables the easepoint(s).
  - **Identifier:** Sets the identifier for the easepoint.
  - **Slow down length (EaseOut):** Sets the size in pixels that is used to ease out the scroll element.
  - **Speed up length (EaseIn):** Sets the size in pixels that is used to ease in the scroll element.
  - **Pause time in seconds, or -1 (indefinitely):** Sets the time in seconds for how long the scroll should wait before continuing the scroll.
  - **Element alignment (0.5 for center):** Sets the alignment of the element relative to the height of the element. `0.5` (equal to 50%) is the default value and represents the middle of the element's bounding box area.
  - **Scroll area alignment (0.5 for center):** Sets the scroll alignment for the element in the scroll area. `0.5` (equal to 50%) is the default value and represents the middle of the scroll's bounding box area.

Note: Pause time in seconds, or -1 (indefinitely) affects the total time the scroll is scheduled to use.

See Also

- **Create New Scroll** in the Viz Trio User Guide
- Trio Scroll Geometry plug-in

Copyright © 2026 Vizrt

2.4.5 Feed Container Plug-ins

The purpose of the Feed plug-ins is to receive real-time data and apply it to the state of a Viz Artist Scene in various ways. They are mostly used to create Scenes for use in financial data and stock feed visualizations.

> **Note:** They are now mostly replaced by a combination of the shared memory map, Viz scripting, Control Plugins plug-ins and the Apply Shared Memory plug-in as the preferred way to bring real-time data into Viz Artist Scenes.

The method of use for the plug-ins is as follows:

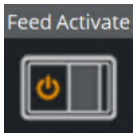
1. Identify text geometry, or container visibility in a scene that an external real-time data feed source is to control.
2. Apply the appropriate Feed plug-in to the container of the identified property.
3. In the plug-in properties, specify a data feed key to look for in the incoming data feed.
4. Send UDP messages to carry the content of the data feed in a YAML format which carries key-value pairs to a specific port on the Viz Engine host, which the plug-in infrastructure listens to.

The following Container plug-ins are located in the Feed folder:

- Feed Activate
- Feed View
- Hide in Range

Copyright © 2026 Vizrt

Feed Activate



The Feed Activate plug-in activates and deactivates plug-ins based on a given value in a FeedRelay hierarchy.

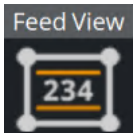
Note: This plug-in is located in: Plugins -> Container plug-ins -> Feed

Feed Activate Properties

- **Is active:** Updates visibility of sub-containers based on container name and value retrieved location.
- **Locator:** Gets data for text from this location.
- **Sub-Locator:** Appends this to locator path when retrieving text.
- **Update Interval:** Determines how often (in number of fields) an update occurs (increase to gain performance).

Copyright © 2026 Vizrt

Feed View



The Feed View plug-in applies value from a real-time feed onto text geometry with various forms of optional formatting.

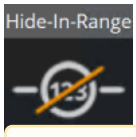
> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Feed

Feed View Properties

- **Is active:** Updates text geometry of container.
- **Locator:** Gets data for text from this location.
- **Sub-Locator:** Appends this to locator path when retrieving text.
- **Update Interval:** Determines how often (in number of fields) an update occurs.
- **Show verbatim:** Shows data as given, does not interpret and reformat.
- **Plus prefix:** Uses this prefix for positive values.
- **Minus prefix:** Uses this prefix for negative values.
- **Suffix:** Appends this suffix.
- **Decimal point:** Uses these characters as a decimal point.
- **Leading zero:** Uses leading zero for values less than one.
- **Zero replacement:** Uses this string when value is zero.
- **Show message when no data:** Sets the contents of the 'Empty Message' field as text if the locator is not found in the cache if checked. If this is unchecked, the text geometry is left unmodified.
- **Empty Message:** Sets text if the locator is unregistered.

Copyright © 2026 Vizrt

Hide in Range



The Hide in Range plug-in makes container either visible or invisible depending on whether value from real-time feed is within a specified range of numerical values.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Feed

Hide in Range Properties

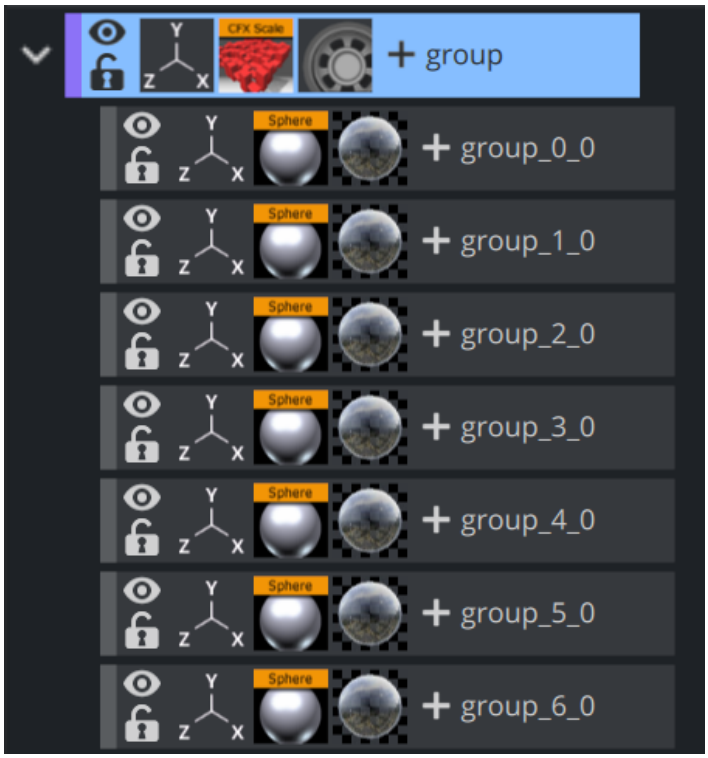
- **Is active:** Updates text geometry of container.
- **Locator:** Gets data for text from this location.
- **Sub-Locator:** Appends this to locator path when retrieving text.
- **Update Interval:** Determines how often (in number of fields) an update occurs.

Copyright © 2026 Vizrt

2.4.6 Container FX Plug-ins

Container FX is much like Basic Container TextFX, except that instead of text, a set of containers are added under the plug-in. Instead of animating the containers, Container FX animates the containers below (aka the container-Set). Those containers can be circles and so on. Most of the Container FX plug-ins share a set of Common Properties.

Like Text FX plug-ins use containers, a Container FX plug-in uses a set of containers under the plug-in and change their position like Text FX does. It is recommended to be familiar with Text FX before designing with Container FX to understand the idea. You can also use the Cloner plug-in to create the necessary containers under the plug-in.



In the above example, the containers under the group container with the Sphere names is the container-set, and is animated here.

The following Container plug-ins are in the Container FX folder:

- Common Container FX Properties
- CFX 2D Follow
- CFX Alpha
- CFX Arrange
- CFX Color
- CFX Explode
- CFX Jitter Alpha
- CFX Jitter Color
- CFX Jitter Position
- CFX Jitter Scale
- CFX Plus Plus

Copyright © 2026 Vizrt

- CFX Rotate
- CFX Scale

Common Container FX Properties

The following properties are common to most of the Container FX plug-ins:

- **Progress %:** 0% progress is the beginning of the effect, 100% the end. Animate this value from 0% to 100% to see the effect or from 100% to 0% to animate the effect backwards.
- **Progress Type**
  - **Absolute:** 100% progress animates all container's set, regardless of how many containers it has.
  - **Relative:** 100% progress animates ten containers. This is needed to adjust the timing of several containers with different sizes. The effect speed should be for example five containers per second, so the animation must be from 0% to 100% in two seconds. This works for with ten containers or less. If you want to use more containers, animate the progress value over 100% (10% for each container).
- **Direction:** Sets the direction of the effect sequence, you can choose between the following options:
  - **Left:** Starts with the first container in the containers-set.
  - **Right:** Starts with the last container in the containers-set.
  - **Random:** Uses a random order.
  - **Static:** Processes all containers at the same time.
  - **Wave:** Starts with the first container, animates the effect from zero: 100% and then down again to 0%.
  - **Center:** Starts the effect from the center of the containers-set.
  - **2 Center:** Starts the effect at the same time from the beginning and the end of the containers-set. They meet at the center.
- **Interpolate:** Chooses between a soft or a linear interpolation of the transition from container to container.
- **Effect Range:** Defines how many containers are processed at the same time. If for example the Effect Range is set to `4`, and you manually increase the progress value, you see that when the fifth container starts to be processed, the first is finished, when the sixth starts, the second is finished, and so on.
- **Random Seed:** Specifies a seed for the random number generator when a random direction is chosen. Even though Viz Artist uses random numbers, the animation for a specific random seed always looks the same. This is typically useful if you combine two different container effects.
- **Container Order:** Sets the container order for the effect. Available options are As Is, Horizontal or Vertical. Horizontal and Vertical enables the containers -set Direction options.
- **Direction:** Sets the horizontal and vertical container-set direction (see Container Order). When the containers-set order is set to Horizontal, the container-set direction must be set to Left to Right or Right to Left. When the container order is set to Vertical, the containers-set direction must be set to Up to Down or Down to Up.

Copyright © 2026 Vizrt

CFX 2D Follow



The Container FX 2D Follow plug-in creates an animation of geometry containers following a path defined by the 2D Ribbon geometry plug-in. 2D Ribbon must be added as a parent container to the container holding CFX 2D Follow. The geometries that are to be animated needs to be added as sub-containers to the container holding CFX 2D Follow.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX 2D Follow Properties

- **Positioning:** *Absolute* positioning evenly distributes the followers across the entire length of the spline. *Relative* means that the followers are added from left to right along the spline.
- **Path Position:** Moves the followers along the 2D Ribbon shape.
- **Position Wrap:** *Interpolate* means that text continues in the direction set by the last section of the spline; *Clamp* stops the string on the spline end-point; *Repeat* causes the string to move to the other spline end-point as soon as it moves over the end-point.
- **Kerning Scale:** Scales the kerning of the spline.
- **Align:** Rotates the followers by Z to align the X-Axis with the tangent of the spline at the followers' position.
- **Translation offset:** Moves the followers from the spline in the XY plane.
- **Rotation:** Rotates the followers using the spline as a rotation axis.

To Set Up the CFX 2D Follow Plug-in

1. Add the 2D Ribbon geometry to the scene tree, then open the 2D Ribbon editor and enable **Show Control Point Values**.
2. Set alternating values of `30.0` and `-30.0` to the **Y axis** values, creating a wave shape.
3. Add a sub-container to the 2D Ribbon container, and add the **Container FX 2D Follow** plug-in to the newly created container.
4. Add the geometries that are to be animated along the 2D Ribbon's path as sub-containers to the Container FX 2D Follow container.
5. Open the **Container FX 2D Follow editor** and animate its **Path Position** value from `0.0` to `100.0`.

See Also

- Common Container FX Properties
- 2D Ribbon

Copyright © 2026 Vizrt

CFX Alpha



The Container FX Alpha plug-in is a container-set effect that creates a fade in effect for the containers in the set.

The effect sequence can be set to go many different ways.

A fade in and out effect can also be achieved.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Alpha Properties

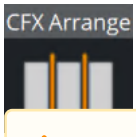
- **Alpha Begin %:** Sets the alpha level of the containers at 0% effect.
- **Alpha End %:** Sets the alpha level of the containers at 100% effect.

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Arrange



The CFX Arrange plug-in arranges containers in either a circular or a wave shape.

The containers can be animated on the selected shape by animating the offset value.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Arrange Properties

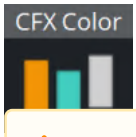
- **Offset %:** Moves the containers on the shape. 100% means one full rotation of the containers-set on the circle.
- **Scale %:** Sets the container-set on the shape. The parameter does not scale the containers, but the center of the bounding box of the containers.
- **Shape:** Changes the shape of the container layout. The options are Circle or Wave.
- **Diameter:** Sets the diameter of the circular shape.
- **Positioning:** Defines the position of the container-set on the circular shape. Relative means that the spacing of the container-set is maintained. Absolute means that the container-set is evenly distributed out on the circle.
- **Direction:** Sets the direction of the container-set on the circle to either Clockwise or Counterclockwise.
- **Rotation:** Rotates the container-set on the X-axis.
- **Align:** Aligns the container along the curve.

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Color



The CFX Color plug-in adds a color effect to the containers-set.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Color Properties

- **Color Start:** Sets the initial color before applying the effect.
- **Color End:** Sets the color the effect should apply.

See Also

- Common Container FX Properties

CFX Explode



The CFX Explode plug-in creates an explosion like function where the containers get thrown away from their initial position.

The speed, direction and spread of the moving containers can be altered with parameters.

Note: Works only if the containers-set is set to texture.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Explode Properties

- **Duration:** Defines the duration of the progress for each of the containers. Simply increase the value if you want it last longer. You could achieve the same by making the gravity stronger and animating the progress slower, but it is easier to increase the duration instead if your explode effect is too short.
- **Opening Angle:** Sets the angle for the spread of the containers. `0` sends them straight up, `360` spreads them in a circular shape.
- **Angle Rotate X:** Rotates the opening angle around the X-axis.
- **Angle Rotate Y:** Rotates the opening angle around the Y-axis.
- **Force:** Sets the force that throws away the containers. A high force makes them go far away, conversely a low force creates only a small motion of the containers.
- **Force Spread %:** Sets a variation of the force among the containers.
- **Use Axis:** Allows you to select on which axis or combination of axes the containers are to spread along.
- **Gravity:** Sets a gravity force that influences the path of the containers to end up going downwards. The higher the value is set, the faster each container diverts from its initial path and starts going downwards.
- **Use Rotation:** Rotates the container as they are being thrown away from their initial position when set to `On`.
- **Rotation Force:** Sets the degree of rotation as the containers are being thrown away.
- **Show Force:** Shows lines showing the containers path and speed in the Scene Editor when set to `On`.

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Jitter Alpha



The CFX Jitter Alpha plug-in creates a jittering motion of the containers by randomly changing the alpha value of each container. The degree of change and the start sequence of the jittering can be altered. To use the plug-in, add it onto a container with a font. To create an animation, animate the progress. Other values can of course be animated as well.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Jitter Alpha Properties

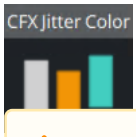
- **Effect What:** Defines where the effect should have an effect. Available options are background (BG), foreground (FG) or both.
- **Alpha Begin:** Sets the alpha level of the container at 0% effect.
- **Alpha End:** Sets the alpha level of the container at 100% effect.
- **Randomness:** Sets the intensity of the jittering alpha changes.

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Jitter Color



The CFX Jitter Color plug-in animates a jittering effect on the color of the container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Jitter Color Properties

- **Anchor Horizontal:** Sets the anchor point for the containers on the horizontal plane.
- **Anchor vertical:** Sets the anchor point for the containers on the vertical plane.
- **Use Axis:** Defines on which axis or axes the containers scale to create the jittering effect.
- **Lock Axis:** Applies the same Min, Max and Randomness settings to all axes when enabled. If you disable randomness, these parameters are visible for each of the axes and must be set individually.
- **Min:** Sets the minimum scaling for the containers.
- **Max:** Sets the maximum scaling for the containers.
- **Randomness:** Sets the intensity of the jittering movement.

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Jitter Position



The CFX Jitter Position plug-in creates a jittering motion of the containers by randomly changing the position of each container. The degree of position change and the starting sequence of the jittering can be altered. To create an animation, animate the progress. Other values can of course be animated as well.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Jitter Position Properties

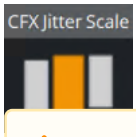
- **Use Axis:** Defines on which axis or axes the containers moves to create the jittering effect.
- **Lock Axis:** Gives the same values for Range and Randomness for all axes when enabled. If you disable it, Range and Randomness for each of the axes is shown and can be set individually.
- **Range:** Sets the range of the jittering movement.
- **Randomness:** Sets the intensity of the jittering movement.

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Jitter Scale



The CFX Jitter Scale plug-in animates a jittering effect on the scale of the container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Jitter Scale Properties

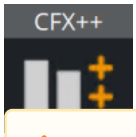
- **Anchor Horizontal:** Sets the anchor point for the containers on the horizontal plane.
- **Anchor vertical:** Sets the anchor point for the containers on the vertical plane.
- **Use Axis:** Defines on which axis or axes the containers scale to create the jittering effect.
- **Lock Axis:** Gives all axes get the same Min, Max and Randomness settings when enabled. If you disable randomness, these parameters are visible for each of the axes and must be set individually.
- **Min:** Sets the minimum scaling for the containers.
- **Max:** Sets the maximum scaling for the containers.
- **Randomness:** Sets the intensity of the jittering movement.

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Plus Plus



The Container FX Plus Plus plug-in allows you to set a number of effects on a container's set.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Plus Plus Properties

- **Position Wrap**
  - **Extrapolate:** Continues the direction of the spline when set above 100% or below 0% of Path Position.
  - **Clamp:** Stops the string on the spline endpoints.
  - **Repeat:** Causes the string to move to the other spline endpoint as soon as it moves over the endpoint.
  - **ContainerFx:** Positions containers by the relative spline of each container using the ContainerFX parameters.
- **Scheme Type:** Defines how the container-set looks at 0% and 100% progress.
  - **In -> In:** at 0% first container is at the beginning of the spline, at 100% last container is at the end of the spline.
  - **In -> Out:** at 0% first container is at the beginning of the spline, at 100% first container is at the end of the spline.
  - **Out -> In:** at 0% last container is at the beginning of the spline, at 100% first container is at the end of the spline.
  - **Out -> Out:** at 0% last container is at the beginning of the spline, at 100% last container is at the end of the spline.
- **Progress (%):** Animates the progress of the effect(s).

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Rotate



The CFX Rotate plug-in allows you to create an effect where the containers rotate on the X-, Y- or Z-axis.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Rotate Properties

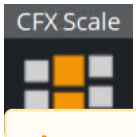
- **Anchor Horizontal:** Sets the anchor point for the containers on the horizontal plane.
- **Anchor Vertical:** Sets the anchor point for the containers on the vertical plane.
- **Use Axis:** Defines on which axis the containers rotate.
- **Begin:** Sets the initial rotated position of the containers.
- **End:** Sets the ending rotated position of the containers.

See Also

- Expert
- Common Container FX Properties

Copyright © 2026 Vizrt

CFX Scale



The CFX Scale plug-in allows you to create a scaling animation of the containers.

Note: This plug-in is located in: Plugins -> Container plug-ins -> ContainerFx

CFX Scale Properties

- **Anchor Horizontal:** Sets the anchor point for the containers on the horizontal plane.
- **Anchor Vertical:** Sets the anchor point for the containers on the vertical plane.
- **Use Axis:** Defines on which axis or axes the containers scale.
- **Lock Axis:** Scales the Begin- and End the same way for all the axes when enabled. If you disable the option, Begin and End must be set for all axes individually.
- **Begin:** Sets the initial size of the containers.
- **End:** Sets the ending size of the containers.

See Also

- Common Container FX Properties

Copyright © 2026 Vizrt

2.4.7 Global Container Plug-ins

Global Plug-ins are built in functions of the Render System. For easier use, they are exposed as Plug-ins.

The following functions are located in the Global folder:

- Alpha
- Audio
- Average Texture
- Clipper
- Expert
- Extrude
- Global Illumination
- Glow
- HDR
- Infotext
- Instancing Plugins
- Key
- Layer
- Light Layer
- Light V4
- Lighting
- Look-At
- Magnifier
- Mask Source and Mask Target
- Masking
- Projector Source and Projector Target
- Script
- Shadow Caster and Shadow Receiver
- Synchronized Properties
- Talent Reflection
- Texture Slot
- Video Clip
- Virtual Window
- Window Mask
- Z-Sort Priority

Copyright © 2026 Vizrt

Alpha



The Alpha plug-in adds an alpha channel to the container.

The alpha channel defines the degree of transparency for the container and its Sub-Containers.

This feature can be used to easily fade in or out a complete part of the scene-tree.

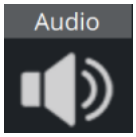
> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Global

Alpha Properties

- **Alpha:** Sets the alpha value for the container.

Copyright © 2026 Vizrt

Audio



The Audio plug-in allows a designer to configure audio channels.

Audio can be applied to any Container. The plug-in has three different audio clip mixing modes, and each mode panel has a test function.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

This page contains the following topics and procedures:

- Default Mode
- FX Mode
- Manual Mode
- To Configure Audio Channel Aliases
- To Test Audio Channel Setup

If the **Mute when invisible** is active, no audio is being played if the contianer is not being rendered. This makes it easier to mute the audio if it should not play if the respective container is also set to be inactive/invisible.

Default Mode

The Default mode uses the Channel settings stored in the clip. For example, *FrontLeft* and *FrontRight* are mixed to the aliases *FRONT_LEFT*, *FRONT_RIGHT* and so on.

If Viz Engine is configured with fewer channels, Viz Engine mixes the remaining clip channels according to the channel geometry. If Viz Engine only has stereo configured, but the clip used has 7.1 surround sound, Viz Engine mixes *LeftBack* and *LeftMiddle* to *FRONT_LEFT* and so on.

The **Pan** slider uses audio aliases when adjusting the volume of the left and right speakers. This is valid for multichannel configurations as well. If the slider is moved to the right in a 7.1 configuration *LeftBack*, *LeftMiddle* and *LeftFront* are muted. The **Volume** field controls the overall audio volume of all clip channels.

FX Mode

In the FX mixing mode, Viz Engine mixes the clip in relation to the listener position which can be defined in the fields X and Y. Viz Engine takes the channel geometry into account. The **Volume** field controls the overall audio volume of all clip channels.

Manual Mode

In Manual mixing mode it is possible to enter the Out Channel alias for every clip channel. The mask can hold more than one channel. This allows the mix of every clip channel to an arbitrary amount of out channels. The **Volume** field controls the overall audio volume of all clip channels.

The syntax for the string is as follows: `CHANNEL_ALIAS%VOLUME+CHANNEL_ALIAS%VOLUME…` or `[CHANNEL_ALIAS%VOLOUME]+`.

Example: [FRONT_LEFT%0.5]+

Copyright © 2026 Vizrt

To Configure Audio Channel Aliases

1. Add an Audio clip to a Scene.
2. Click the **Audio** plug-in icon (opens the Audio editor).
3. Click **Manual**.
4. Enter these channel configurations (as entered in **Audio Settings** in Viz Configuration (see the Viz Engine Administrator Guide):
    - Out Channel 0: *FRONT_LEFT*
    - Out Channel 1: *FRONT_RIGHT*
    - Out Channel 2: *FRONT_CENTER*
    - Out Channel 3: *LOW_FREQUENCY*
    - Out Channel 4: *BACK_LEFT*
    - Out Channel 5: *BACK_RIGHT*
    - Out Channel 6: *SIDE_LEFT*
    - Out Channel 7: *SIDE_RIGHT*
5. Save the scene. Depending on the settings in the channel configuration a clip now plays the different languages.

Note: You can use aliases as configured in the Channels tab (see the **Audio Settings** page in the **Configuring Viz** section of the Viz Engine Administrator Guide.

To Test Audio Channel Setup

1. Add a group container to the scene tree.
2. Add Audio to the group container.
3. Open the Audio editor, and add an audio clip to the *Audio Clip* drop-zone, and click **Play**.

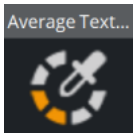
Tip: Always have a set of test clips that provide audio for the different channel setups.

See Also

- Audio Settings (see the **Configuring Viz** section of the Viz Engine Administrator Guide)

Copyright © 2026 Vizrt

Average Texture



The Average Texture plug-in takes an image or a media asset as a parameter and populates its average RGB color onto several places.

You can use this plug-in to set the light color based on the average color of your live video signal.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Info: This plug-in works in the Viz Engine Render Pipeline only!

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Average Texture Properties

- **Target Type:** Specifies where the average color gets applied to. This requires, that the same container also holds either a Viz Light Object or a Viz Material Definition.
  - **Light Color:** Forwards the RGB value to the container light.
  - **Material Color:** The container's Viz Material tint receives the new color value.
  - **None:** Disables any forwarding.
- **Submit:** Enables Shared Memory Support. Once this is turned on, every change calculated by this plug-in is submitted to Viz Engine's shared memory interface.
  - **Target Type:** Defines which shared memory Map to use.
    - **Scene:** The local scene shared memory map is used.
    - **Global:** This is needed if several scenes on the same Viz Instances (for example, GFX channels, etc.) need to be notified about changes.
    - **Distributed:** If multiple Engines need to get these RGB changes. Changes are distributed via all connected Viz Engines on the same Graphic Hub.
  - **Key:** A unique key used to receive the changes.

The following script shows an example on how to use the Shared Memory Interface via Viz Script:

```
sub updateSHM()
  dim c as Container
  dim sourceName as String
  dim avg as Color
  avg =(Color)Scene.Map["average_texture_color"]
  c = Scene.findContainer("shm_color_text")
  c.Material.Color = avg
  println avg
end sub
```

Copyright © 2026 Vizrt

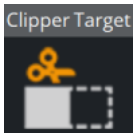
```
sub OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)
  if mapkey="average_texture_color" Then
    println("SHM change: " & mapKey)
    updateSHM()
  end If
end sub

sub OnInit()
  scene.map.RegisterChangedCallback("average_texture_color")
end sub
```

Information: This plug-in doesn't work with DXT compressed images. It is mainly used for Media Assets like Live Input feeds.

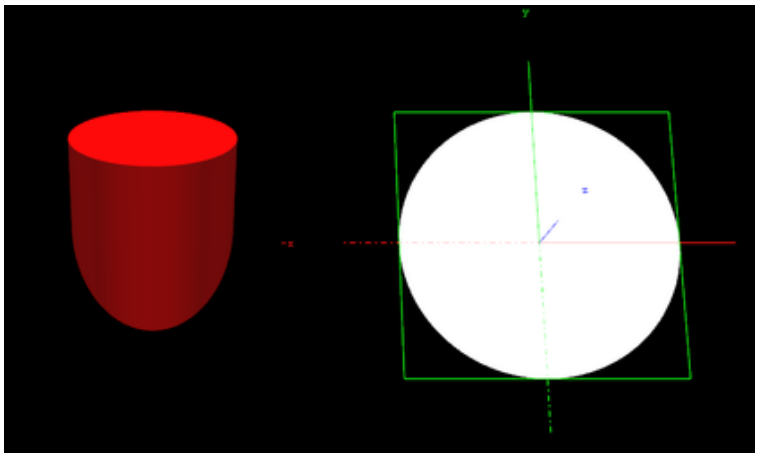
Copyright © 2026 Vizrt

Clipper



The Clipper plug-in is an alternative to the mask function. Objects with Clipper attached that are placed behind the Clipper plane are masked/clipped. If an object is only partially behind a Clipper plane, only the parts of the object that are behind the plane are clipped out, since the clipping is done in true 3D space.

Clipper uses the OpenGL clipping planes. Up to six of them may be used. To define a Clipper plane use the Properties Panel.



All the containers that you want to be affected by the clipping plane, you must have Clipper applied and specify in the editor which of the clipping planes is to mask out/clip the container. It is possible to let a container be affected by multiple clipping planes.

Clipper can replace mask in many situations. Some of the advantages are compared to normal mask are:

- Clipping is done in true 3D space, while masking is done in 2D (although the 2D mask is created by a 3D object).
- There is no performance hit. In fact there is even a performance gain if parts of the object are clipped. Regular masks have a quite high performance cost.

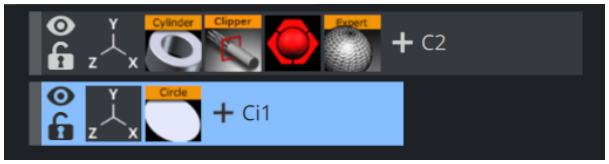
> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Global

Clipper Properties

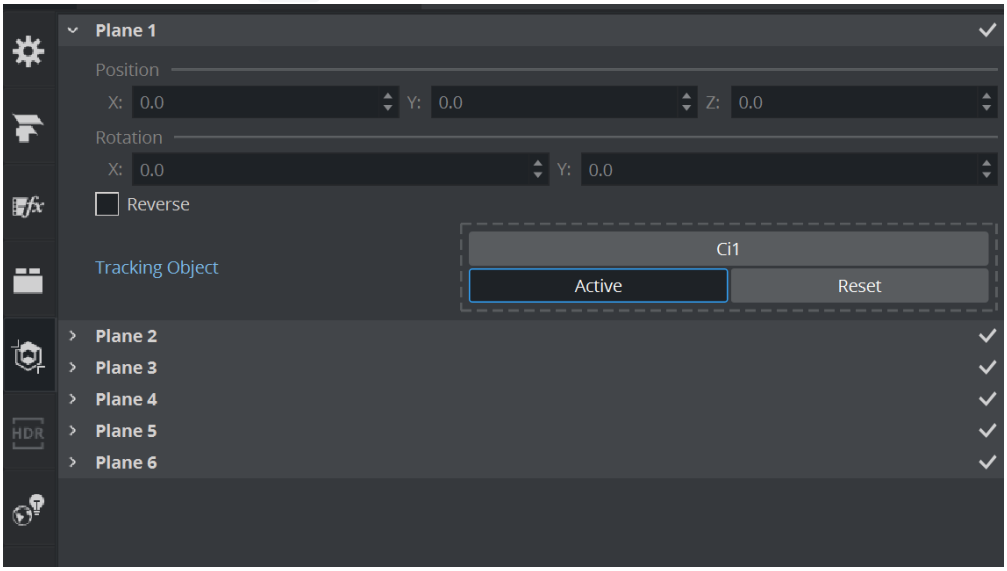
- **Planes:** Enables the planes. By default, `1` is enabled.

Copyright © 2026 Vizrt

To Create a Clipping Plane Effect

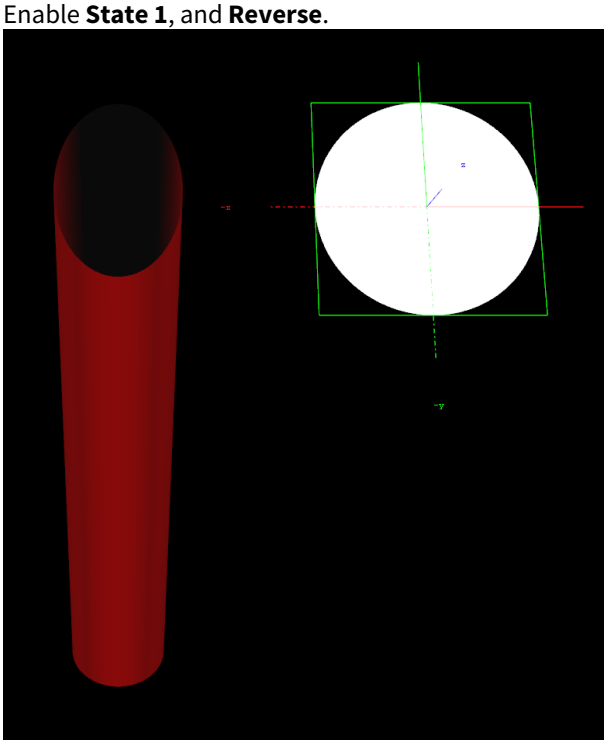


1. Add two group containers to the scene tree (same level) and name them `Cylinder` and `ClipperRectangle`.
2. Add the following to the *Cylinder* container:
   - Cylinder
   - Clipper
   - Material
   - Optional: Expert
3. Add the following to the *ClipperRectangle* container:
   - Rectangle
   - Material
   - Optional: Expert
4. Open the editor for the *Cylinder* container and set **Scaling** (locked) to `2.0`.
5. Open the Cylinder editor and set Hole to `45.0`.
6. Open the Clipper editor and enable **Plane 1** (enabled by default).
7. Optional: Open the Expert editor(s) and enable **Back Face** (On) and **Twosided Lighting**.
8. Open the editor for the *ClipperRectangle* and set the following parameters for the following properties:
   - Set Position Y to `45.0`.
   - Set Rotation X to `-45.0`.
   - Set Rotation Y to `45.0`.
   - Set Scaling (locked) to `2.0`.



Copyright © 2026 Vizrt

9. Open Scene Settings, and select the Clipper Panel tab.
10. Enable **State 1**, and **Reverse**.

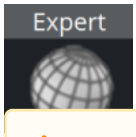


11. Drag and drop the *ClipperRectangle* container onto the Tracking Object drop zone.

Information: In the New Render Pipeline, the Clipper requires a Material to work correctly.

Copyright © 2026 Vizrt

Expert



The Expert plug-in sets some special properties and adds some advanced functions to a Container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

The plug-in shows a different user interface based on the chosen Render Pipeline:

Viz Engine Renderer

Draw Options

- **Draw Mode:** Selects how the container is drawn.
  - **Normal:** Shows the rendered object with material and/or textures.
  - **Wireframe:** Shows the polygons that are drawn to create the object.
  - **Outline:** Shows the outline of the object (needs to be supported by a geometry plug-in).
- **Line Width:** Sets the width of the wires if **Wireframe** or **Outline** draw mode is selected.
- **Disable Backface Culling:** Renders the back face of a geometry. This can be useful when rendering transparent objects.
- **Two-sided Lighting:** Enables two-sided lighting for a single container instead for a whole scene.
- **Multi-sampled Alpha Mask:** Converts the Alpha Channel to a multi-sampled coverage mask.
- **Use Program Point Size:** When enabled, the size of rendered points is taken from the shader program instead of the fixed default point size. Enable this when a custom shader or plugin renders point primitives and controls their size, for example per-point sizes in a point cloud or particle effect. Has no effect on non-point geometry.

Depth Options

- **Write to Depth:** Enables writing into the depth buffer for this geometry.
- **Disable Depth Test:**

GBuffer Options

- **Write to GBuffer**: Enables writing into the GBuffer.

Blending Options

- **Enable Render Mode:** Automatically adds transparency to the object. To disable this transparency in the Viz Engine Render Pipeline, disable *Enable Render Mode*.
- **Render Modes**
  - **Add:** Symbolically: C*A + FC. The source color gets added to the target color. The amount of color that is added depends, as we see from the formula, on the alpha value. However, it is always an addition, so the end result is always a lighter color than the initial frame content. If the frame color has high values on all three color channels (RGB), you might experience that the addition of the new

Copyright © 2026 Vizrt

color takes all channels to values above `255` (saturation). The values are clamped at `255`, which is white.

- **Blend:** Symbolically: C*A+(1-A)*FC. The new color value gets created as a weighted average of the source and the target. The weight factor is the alpha value of the rendered color. That means that if the incoming color has a very low alpha value, its influence on the new color is small, and conversely, if the new color has a high alpha value, its influence on the new color is larger.
- **Subtract:** Symbolically: FC-C*A. The new color is the result of the incoming color being subtracted from the color in the frame buffer and the result is written back into the frame buffer. The alpha value of the incoming color decides how much who gets subtracted. If the incoming color has high values on all three color channels (RGB), you might experience that the subtraction of the new color takes all channel values to `0`, and the result is a black color.
- **Rev-Subtract:** Symbolically: C*A-FC. This is a reversed version of subtractive. The color in the frame buffer gets subtracted from the incoming color and the result is written back into the frame buffer.
- **Multiply:** Symbolically: CA*FC. The new color and alpha get multiplied with the existing values in the frame buffer. The formula presupposes that the colors and alpha re described as values between `0` and `1`. A color rendered with multiply always results in a darker color than both the color being rendered and the color in the frame buffer.
- **Rev-Multiply:** This is a reversed version of multiply.
- **Max:** Higher color values overwrite lower color values (for example, white draws over black - independently from the Z-Sort).
- **Blend (Premultiplied Alpha):** Like Blend, but assuming the color already has the alpha multiplied.
- All Render Modes having ***(Advanced)*** in their name are supported by Viz Engine Renderer and are mimicking the respective Photoshop blend modes.

Mirroring

Enables you to mirror the object over the X-, Y- and Z-axis.

Draw Layer

Defines if the objects are drawn in all layers or only if it shows in a certain layer:

- **Any:** The object is always drawn.
- **Main:** The object is drawn only if it is in a scene, located in the Main Layer.
- **Back:** The object is drawn only if it sits in the Back Layer.
- **Front:** The object is drawn only if it is in the Front Layer.

Classic Renderer

- **Draw Mode:** Selects how the container is drawn.
  - **Normal:** Shows the rendered object with material and/or textures.
  - **Wireframe:** Shows the polygons that are drawn to create the object.
  - **Outline:** Shows the outline of the object (needs to be supported by a geometry plug-in).
- **Line Width:** Sets the width of the wires if **Wireframe** or **Outline** draw mode is selected.
- **Disable Backface Culling:** Renders the back face of a geometry. This can be useful when rendering transparent objects.

Copyright © 2026 Vizrt

- **Two-sided Lighting:** Enables two-sided lighting for a single container instead for a whole scene.
- **Z-Buffer Draw:** Defines if the depth buffer is used for rendering or not.
- **Z-Buffer Ignore:** Ignores the OpenGL Z-buffer draw option when set to `Off`, and draws the object no matter if it is in the back or not.
- **Line Anti-aliasing:** Renders an anti-aliased outline for 2D objects on machines that do not have anti-aliasing with multi-sampling. This extra outline can be removed by disabling this option.
- **Exact Picking:** Exact Picking is usually a bit slower, as it uses pixel values instead of using the bounding box dimension.
- **Separate Specular:** Enables specular highlights for textured geometry.
- **Render Mode:**
    - **Add:** Symbolically: C*A + FC. The source color gets added to the target color. The amount of color that is added depends, as we see from the formula, on the alpha value. However, it is always an addition, so the end result is always a lighter color than the initial frame content. If the frame color has high values on all three color channels (RGB), you might experience that the addition of the new color takes all channels to values above `255` (saturation). The values are clamped at `255`, which is white.
    - **Blend:** Symbolically: C*A+(1-A)*FC. The new color value gets created as a weighted average of the source and the target. The weight factor is the alpha value of the rendered color. That means that if the incoming color has a very low alpha value, its influence on the new color is small, and conversely, if the new color has a high alpha value, its influence on the new color is larger.
    - **Subtract:** Symbolically: FC-C*A. The new color is the result of the incoming color being subtracted from the color in the frame buffer and the result is written back into the frame buffer. The alpha value of the incoming color decides how much who gets subtracted. If the incoming color has high values on all three color channels (RGB), you might experience that the subtraction of the new color takes all channel values to `0`, and the result is a black color.
    - **Rev-Subtract:** Symbolically: C*A-FC. This is a reversed version of subtractive. The color in the frame buffer gets subtracted from the incoming color and the result is written back into the frame buffer.
    - **Multiply:** Symbolically: CA*FC. The new color and alpha get multiplied with the existing values in the frame buffer. The formula presupposes that the colors and alpha re described as values between `0` and `1`. A color rendered with multiply always results in a darker color than both the color being rendered and the color in the frame buffer.
    - **Rev-Multiply:** This is a reversed version of multiply.
    - **Max:** Higher color values overwrite lower color values (for example, white draws over black - independently from the Z-Sort).
- **Mirror:** Enables you to mirror the object over the X-, Y- and Z-axis.
- **Shade Model**
    - **Flat:** Shows the single polygons the object is made of. Lighting is only applied per face and not per vertex.
    - **Smooth:** Calculates a smooth surface.
- **Transparency Mode**
    - **Blend:** Is the standard option.
    - **Mask:** Uses a raster.
    - **Fastest:** Uses the transparency mode which takes the least time to render.
    - **High Quality:** Uses the transparency mode which delivers the best looking result.

Copyright © 2026 Vizrt

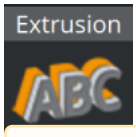
- **Draw Layer:** Defines if the objects are drawn in all layers or only if it shows in a certain layer:
  - **Any:** The object is always drawn.
  - **Main:** The object is drawn only if it is in a scene, located in the Main Layer.
  - **Back:** The object is drawn only if it sits in the Back Layer.
  - **Front:** The object is drawn only if it is in the Front Layer.



Note: The Draw Layer option can be used, for example, for a menu in a Scene which should be only visible if it is loaded in the front layer.

Copyright © 2026 Vizrt

Extrude



The Extrude plug-in sweeps a 2D item through space along its Z-axis. The sweeping path the item is followed during this process is used to create a surface. As a result a 3D item with front, back and sides is created from the 2D item.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Extrude Properties

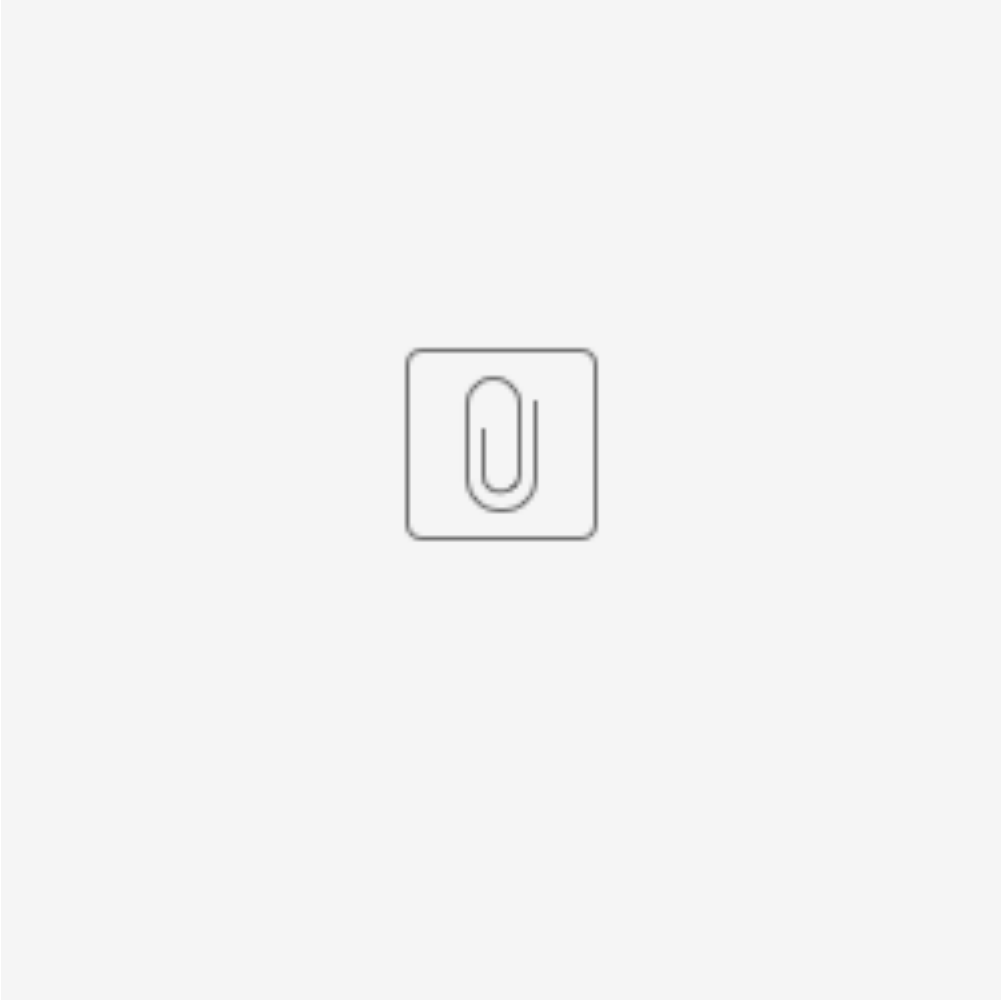
- **Bevel Types:** The buttons marked with different bevel types are used to set the type of bevel to use on the figure.
- **Extrusion Depth:** Sets the extrusion depth on the Z-axis.
- **Bevel Size:** Sets the size of the bevel. This must be set to some value to enable the selection of bevel type.
- **Shading Angle:** Sets the angle of the shading function. To “erase out” unevenness, try increasing the shading angle.
- **Bevel Detail:** Sets the degree of detail of the bevel. The lower the value is set the more detailed the bevel is constructed. The more detailed the bevel is, the smaller tiling it is constructed from.
- **Backface:** Allows you to enable or disable visualization of the backface.

If colors are set to **Active**, you can edit the items **Front**, **Back** and **Side** color by using the sliders or changing the values.

Note: Adding Extrude to a font sets the font’s quality to `Normal` and disables the **Quality** selector in the Text Editor.

Copyright © 2026 Vizrt

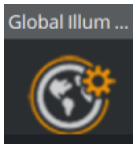
To Extrude a 2D object



1. Create a group and add a Circle and material to it.
2. Add Extrude to the same container.
3. Open the Extrude editor and select one of the bevel effects, and set Extrusion depth to `100.0` and Bevel Size to `10.0`.

Copyright © 2026 Vizrt

Global Illumination



The Global Illumination Settings plug-in defines how Global Illumination is applied to this and all subcontainers. 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*).

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.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Info: This plug-in works in the Viz Engine Render Pipeline only!

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Global Illumination Parameters

- **Visibility**
  - **Render:** The container itself and the hierarchy beyond is visible to the real-time renderer.
  - **Global Illumination:** The container itself and the hierarchy beyond is visible to the Global Illumination system.
- **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 Global Illumination.
- **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.

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.

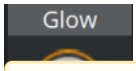
Note: The GI Settings plug-in supersedes the automatic mesh classification of Viz Artist 3. Automatic mesh classification is not supported anymore in the Viz Engine Render Pipeline.

Copyright © 2026 Vizrt

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.

Copyright © 2026 Vizrt

Glow



The Glow plug-in makes the lit faces of objects glow.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Note: This plug-in has been deprecated in the Viz Engine Render Pipeline. It can be replaced with emissive colors and blooming effects.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

Glow Properties (Classic Render Pipeline)

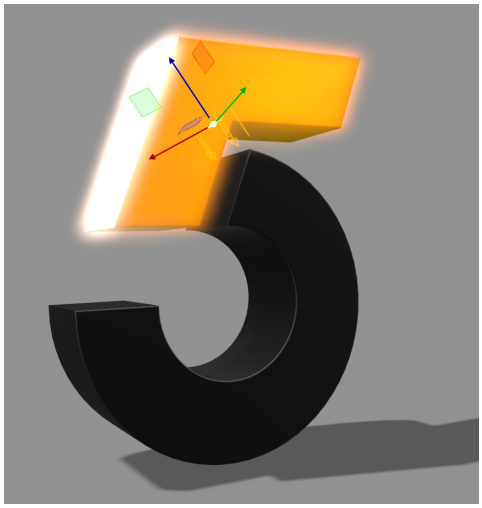
- **Blur Width:** Sets the width of the blur (the size of the glow).
- **Strength:** Sets the intensity/strength of the glow.
- **Object Strength:** Sets the intensity/strength of the glow on the faces of the object.
- **Draw Texture:** Uses the selected Texture for the Glow Effect. When a multi-textured object is used, use the **Unit** drop down box to select which of the textures to use.
- **Use Color:** Changes the color of the glow when set to `On`. If set to `Off`, the glow is based on the Color (Material and Texture) of the Object.
- **Textured:** Applies the Texture, or Textures, of the object on the glow.

Glow Properties (Viz Engine Render Pipeline)

- **Blur Iterations:** Defines how often the blur effect is applied.
- **Intensity:** Sets the intensity/strength of the glow.
- **Object Intensity:** Sets the intensity/strength of the glow within the object.
- **Enable Auto Prioritization:** Enables Viz Engine to organize rendering of the glow effect automatically for this object.
- **Priority:** Sets the priority number used to organize rendering of the glow effect.
- **Enable Color Tint and Color:** Enables and defines a certain color for the glow effect.

Note: For all properties except Blur iterations, a material must be applied first.

Copyright © 2026 Vizrt



Using Priority to Optimize Rendering Performance

Rendering Glow effect in Viz Engine Render Pipeline requires multiple passes (i.e. GPU draw calls), and users are encouraged to minimize such rendering passes based on scene design.

In the typical situation where the number of glowing geometries is limited (i.e. less than 5 - 10), enabling Auto Prioritization is recommended to let Viz Engine group the glowing geometries based on their locations on the screen and minimize the number of passes automatically.

However, if a crowd of glowing geometries is expected, Auto Prioritization should be turned off, and Priority number is supposed to be assigned manually, which hints Viz Engine at mapping the geometry onto the rendering pass. The users are encouraged to group glowing geometries with the Priority number to limit the number of passes. Note that the Priority number is unrelated to the order the base geometries are rendered (i.e. geometries before glowing).

Without Auto Prioritization, two geometries are dedicated to the same rendering pass if and only if the following conditions are satisfied:

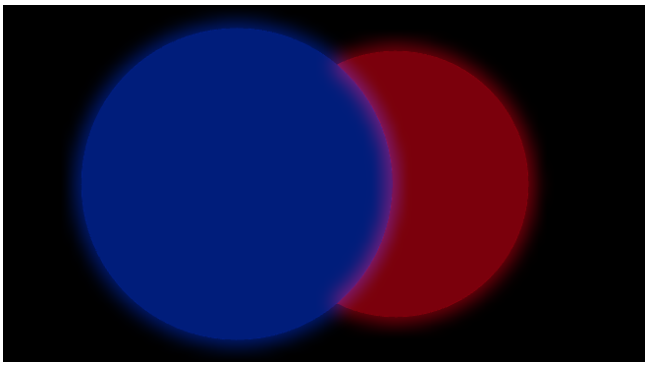
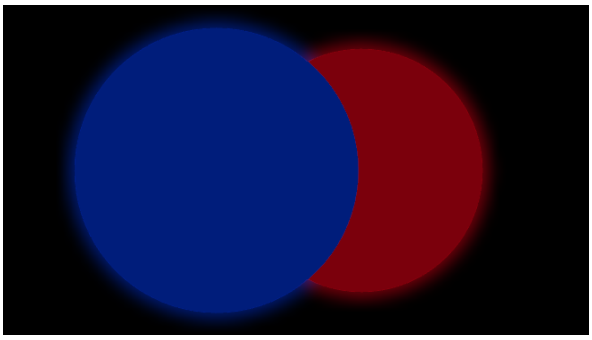
- The Priority numbers are identical.
- The following properties are identical:
    - Blur Iterations
    - Object Intensity

For any geometries dedicated to the same rendering pass, the Glow effect for them is rendered at once, and it results in the loss of light bleeding when there is an overlap in a camera view. Thus, it is strongly recommended to consider such trade-off during scene design.

These are example cases where a single rendering pass for multiple glowing geometries definitely benefits:

- A group of glowing geometries that are supposed to be a single bigger geometry (e.g. a rigid structure or a tree).
- A group of glowing flat geometries that lie on the same plane (e.g. 2D arrows on a football field).
- A repetition of glowing geometries (e.g. aligned arrows or particle sprites).

Copyright © 2026 Vizrt

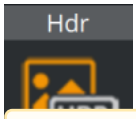
Separated Rendering Passes	Same Rendering Pass
	

Recommendations

- If the geometries are supposed to be a single bigger geometry, always group their containers and assign the Glow plugin to the parent container. This deliberately assigns them with the same Priority number, and thus the same rendering pass.
- Given the coplanar overlap of flat geometries (e.g. same-depth camera-facing circles),
    - If one geometry glows and the another is not, make sure that the non-glowing one is rendered **first**. This could be achieved by either placing the container holding the glowing geometry after (or beyond the bottom of) the container holding the non-glowing geometry in the scene tree or using Z-Sort Priority plugin. And if neither of them is possible:
        - Depending on the selected render sequence, if the non-glowing geometry is a full-screen image (e.g. a talent), set it as a background or foreground of the scene.
        - Migrate it to other layer. This is either the one listed in Layer plugin (only work for multi-layer render sequences) or the scene layer (i.e. Back, Main, Front).
        - Lift up the non-glowing geometry by tweaking its position to separate the depth.
    - If both geometries glow and have the same properties, assign them the same Priority number as suggested in the previous section. In this case, they are spatially attached, and glowing without light bleeding at the connecting edge is expected as a normal phenomenon in the real world.

Copyright © 2026 Vizrt

HDR



The HDR plug-in 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 plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Note: This plug-in has been deprecated in the Viz Engine Render Pipeline.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

HDR Properties

This plug-in does not have any properties or parameters.

To Add HDR

Drag the plug-in on a container and modify the HDR settings under **Scene settings > HDR**. Specify a diffuse and a Cubic Reflection image, as required.

Copyright © 2026 Vizrt

Infotext



The Infotext plug-in allows to add some additional information to a certain container.

This additional information could be, for example, file history on a top container or instructions how to use a certain script, etc..

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Infotext Properties

This plug-in does not have any properties or parameters.

To Add Infotext

Drag Infotext onto a container and enter any additional text into the text box in Infotext properties.

Copyright © 2026 Vizrt

Instancing Plugins



The Instancing and Instancing Producer plug-ins are required for a special way of efficient rendering of multiple objects.

To define the source of your instance, place Instancing on a container and assign a Target container hosting Instancing Producer in the UI. This renders copies of the instanced container underneath the producer container.

For details on this feature, please see **Introduction to Viz Engine Rendering > Instancing** in the Viz Artist User Guide.

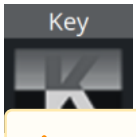
Compatibility Information: This plug-in works in the Viz Engine Render Pipeline only!

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Copyright © 2026 Vizrt

Key



The Key plug-in adds a Key signal to a container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Key Properties

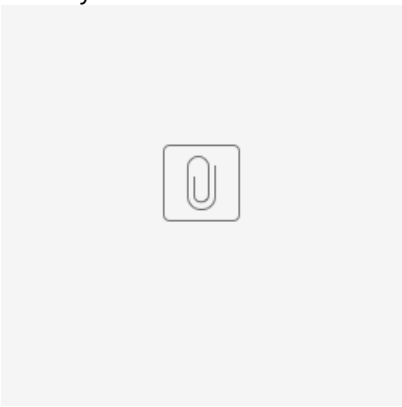
- **Draw RGB:** Enables or disables the graphical object.
- **Draw Key:** Enables or disables the key signal.
- **Automatic Alpha:** Calculates the alpha value automatically when set to `On`. Takes the alpha value entered when set to `Off`. This is normally done to obtain some level of transparency so that, for example, video background is visible through the key object.
- **Alpha as Key Only:** Takes the alpha value of the object and renders the key with this alpha instead of the alpha value of the key function attached. The object itself is rendered opaque. This is usually used to render transparent objects for keyed graphics. The color mixing of the object in the scene and the, for example, video background is done by the external Keyer so the object must be rendered non transparent.
- **Combine with BG chroma key:** Combines the alpha of the Container with the alpha of the background, before blending the foreground with the background (active in a Classic Render Pipeline's virtual sports render sequence only) when set to `On`.
- **Depth information only:** Occludes depth only for rendering with live video. Containers which are marked for **Depth Only** are rendered before Containers which are marked for chroma or linear keying. This could be useful for sports, where real objects should occlude virtual items of the scene. This setting requires the **Render Sequence** to be set to `Arena` in the scene's Global Settings Panel.
- **Separate Depth Render Pass:** Only necessary for Libero render sequence. Renders containers in a second pass after merging depth information. Useful if a container should be always rendered on top (for example, a beam).
- **Render Mode:** Sets the mode in which the alpha values of the key item to be rendered should be mixed with the alpha values already existing in the frame buffer. The mixing is done on a per pixel basis (in the formulas the range of the alpha values is from `0-1`, instead of `0-100` as in the value field).
  - **Add:** Key = SourceKey + TargetKey
  - **Blend:** Key = SourceKey * SourceAlpha + TargetKey * (1-SourceAlpha)
  - **Blend (Premultiplied Alpha):** Key = SourceKey + TargetKey * (1-SourceAlpha)
  - **Subtract:** Key = SourceKey - TargetKey
  - **Rev-Subtract:** Key = TargetKey - SourceKey
  - **Max:** Key = Max (SourceKey, TargetKey)
  - **Min:** Key = Min (SourceKey, TargetKey)

Note: Be aware that with the subtractive and the rev-subtractive, mode the order in which the objects are being rendered then is crucial. Make sure the z-sort is correct.

Copyright © 2026 Vizrt

To Add Key

1. Open a Scene.
2. Add an object.
3. Add Key.

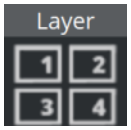


4. Click the **Key** button on the left side of the Scene Editor.



Copyright © 2026 Vizrt

Layer



The Layer plug-in controls which layer the container is rendered to before composition in multi-layer render sequences.

This plug-in works with the following render sequences:

- Multi-Layer Virtual Studio
- Multi-Layer External VS
- Multi-Layer XR Set
- Arena Image-Based
- Libero
- Libero Fill&Key

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Info: This plug-in works in the Viz Engine Render Pipeline only!

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Note: This plug-in introduces in-scene layering, and is unrelated to scene layers (i.e. Back, Main, and Front layers)

Layer Properties

- **Main:** If checked, the container is rendered to the main layer, where every graphics is rendered to in non-multi-layer render sequences.
- **Overlay:** If checked, the container is rendered to the overlay layer, which is for an overlay graphics in general.
- **Auxiliary:** If checked, the container is rendered to the auxiliary layer, which supports or interacts with other layers on rendering.
- **SSR Foreground:** If checked, the container is rendered to the separated layer that is recomposed to the main layer after SSR is rendered. See Screen Space Reflection#%5BinlineExtension%5DMulti-Layer-SSR .

Warning: Every multi-layer render sequence utilizes these layers in different ways. Please check the description of each render sequence before assigning containers to a particular layer.

Copyright © 2026 Vizrt

Configuring for Viz Arena

To prepare containers in a Viz Engine Render Pipeline scene used for Viz Arena, the containers must be assigned to **only one** of these layers:

- Main: The container is linearly keyed by the alpha of the scene background (e.g. the container disappears in the area where the alpha is zero).
- Overlay: The container is rendered on top of the scene background and any graphics in Main layer.

The following table maps the container configuration in Classic Render Pipeline to Viz Engine Render Pipeline.

Classic Render Pipeline	Viz Engine Render Pipeline
Key plug-in with **Combine with BG Chroma Key** set to On	Key and Layer plug-ins with **Main** checked
Key plug-in	Key and Layer plug-ins with **Overlay** checked

For more information, see Viz Libero and Viz Arena Render Sequences#Arena-Image-Based-Render-Sequence-(Image-Based-Tracking) .

Configuring for Viz Libero

To prepare containers in a Viz Engine Render Pipeline scene used for Viz Libero, the containers must be assigned to **only one** of these layers:

- Main: The container is linearly keyed by the alpha of the scene background (e.g. the container disappears in the area where the alpha of the background is zero).
- Auxiliary: The container is rendered on top of the scene background and any graphics in Main layer, while being depth-tested with the depth buffer shared from Viz Libero.
- Overlay: The container is rendered on top of the scene background and any graphics in Main and Auxiliary layers.

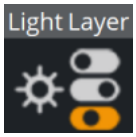
The following table maps the container configuration in Classic Render Pipeline to Viz Engine Render Pipeline.

Classic Render Pipeline	Viz Engine Render Pipeline
Key plug-in with **Combine with BG Chroma Key** set to On	Key and Layer plug-in with **Main** checked
Key plug-in	Key and Layer plug-in with **Auxiliary** checked
Key plug-in with **Separate Depth Render Pass** set to On	Key and Layer plug-in with **Overlay** checked

For more information, see Viz Libero and Viz Arena Render Sequences#Libero-Render-Sequence .

Copyright © 2026 Vizrt

Light Layer



The Light Layer plug-in applies lighting to a certain group of containers.

Every light has a light layer property, the light with its corresponding bit-wise layer number is only applied to the light layer group.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Info: This plug-in works in the Viz Engine Render Pipeline only!

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Light Layer Properties

- **Lights (1-32):** Selects which lights are being applied to this container. If you have two spotlights within your scene, a layer can be set for each single light. You can choose if your geometry is lit by Light 1, Light 2 or both.

Copyright © 2026 Vizrt

Light V4



The Light (V4) plug-in adds a new Viz Engine light onto a container.

More detailed information about how to use Lights in Viz is available at **Light and Shadows** in Viz in the Viz Artist User Guide.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Info: This plug-in works in the Viz Engine Render Pipeline only!

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Light (V4) Properties

- **Light Type:** Selects what kind of light is applied to your container.
  - **Directional**
  - **Point**
  - **Spot**
  - **Area**
- **Color:** Selects the color the light source emits.
- **Intensity:** Determines the strength of the light source.
- **Diffuse Intensity:** Determines how much light diffuse areas emit.
- **Specular Intensity:** Determines how much specular areas of the source are affected. This is also important for Blooming Effects.
- **Radius:** Defines the active area of a Point or Spotlight.
- **Outer and Inner Cone Angle:** Defines the angle of a spot light's beam (available to spot lights only).
- **Layer:** Defines the layer, this light is active. See Light Layer plug-in.

Shadow Settings

- **Shadow On/Off:** Generates shadows from the lights when set to `On` (this setting is not available on Area Lights).
- **Offset Factor:** Defines the polygon offset factor for shadow rendering.
- **Offset Units:** Defines the polygon offset units for shadow rendering.
- **Split Lambda:** The value to blend between log and uniform cascade splitting.
- **Pure Shadow Light:** If enabled, 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%).

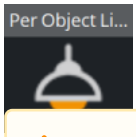
Copyright © 2026 Vizrt

Global Illumination Properties

- **Radiosity Multiplier:** Defines a scale factor for the radiosity contribution of the real-time light source.
- **Bake:** Defines if your light is used as a real-time light or as a baked Global Illumination light.
- **Baked Radiosity Multiplier:** Defines a scale factor for the radiosity contribution of the baked light source.
- **Baked Directional Spread:** Produces basic soft shadows for baked lights when given positive values (for Directional lights only).

Copyright © 2026 Vizrt

Lighting



The Lighting plug-in adds individual lighting per object.

This affects the containers only, whereas global lights affect the whole scene.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Note: This plug-in has been deprecated in the Viz Engine Render Pipeline.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Lighting Properties

- **Type:** Determines the light source type:
  - **Local:** The Local light source is a positional light. It is near or within the scene, and the direction of its rays is taken into account in lighting calculations. Local lights have a greater performance cost than directional lights, due to the additional calculations. A real life equivalent to a local light source is a light bulb. The Local light source has properties for color and position.
  - **Spot:** The Spot light source emits a cone of light. The only objects that are illuminated are those within the cone. The Spot light source has properties for color, position, rotation, and attenuation.
  - **Infinite:** The Infinite light source is a directional light. It is considered to be an infinite distance away from the objects in the scene. Because of the distance, the rays of light are considered parallel by the time they reach the object. A real life equivalent to an infinite light source is sunlight. The Infinite light source has properties for color, position, and rotation. Infinite is selected by default.
- **Shadow:** Turns shadows on or off.
- **Smoothness:** Affects how shadows blend. Value from `1` to `10`. The smoother, the more blended (soft) the shadows are.
- **Shadow Color:** Selects the color and alpha value of the shadow.
- **Color:** Sets the light source color.
- **Position:** Sets the position of the light source along the X, Y, and Z axis.
- **Rotation:** Sets the values for Pan, Tilt, and Twist for Spot or Infinite light sources.
- **Cascaded Shadow Mask:** Determines the frustum distances affected by shadows (i.e. the three-dimensional region with shadow effects specified as Near, Middle and Far regions).
- **Spot:** Sets the concentration of the light within the light cone of the Spot light source. When set to zero, the whole light cone has the same intensity. If set greater than zero, the intensity decreases away from the center.

Copyright © 2026 Vizrt

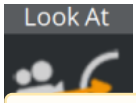
To Add Light



1. Open the Light Editor, and disable global light settings.
2. Add a Sphere geometry to the scene tree, and add material and Lighting to it.
3. Duplicate the Sphere container and place it at the same level (root) as the other container.
4. Open the transformation editor and set Position X to `-100.0` for the first container and Position X to `100.0` for the second.
5. Open the Lighting editors the first container one and set the following parameters:
    - Set Type to `Spot`.
    - Set Light Color to `0,255,0`.
6. Open the Lighting editors the second container one and set the following parameters:
    - Set Type to Spot.
    - Set Light Color to `0,255,0`.
    - Set Position X and Y to `650.0` and `-850.0`, respectively.
    - Set Rotation X and Y to `-650.0` and `850.0`, respectively.

Copyright © 2026 Vizrt

Look-At



The Look-At plug-in is used to rotate a Container so that it faces a defined direction (billboarding). This is especially valuable in animations, it ensures that a geometry always faces towards the camera.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Look-At Properties

- **Auto Rotation**
  - **None:** Does not rotate.
  - **Billboard:** Rotates the container to face the camera.
  - **Follow Path:** Rotates the container to justify at the animation path.
- **Keep Size:**
- **Billboard Axis:** Only with **Billboard:**
  - **X:** Rotates the container around the X-axis to face the camera, even if camera moves.
  - **Y:** Rotates the container around the Y-axis to face the camera, even if camera moves.
  - **XY Camera:** Rotates the container around the X- and Y-axis to face the camera, even if it moves.
  - **XY Screen:** Rotates the container around the X and Y-axis to face the camera, if the camera orbits the container only.

See Also

- **Create Animations** section of the Viz Artist User Guide.

Copyright © 2026 Vizrt

Magnifier



The Magnifier plug-in enlarges certain areas of the render output. Magnifier magnifies all underlying graphics.

Create a container and drag Magnifier onto it.

It automatically magnifies the underlying parts. It can be used multiple times (in contrast to the scene magnifier functionality).



Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

Magnifier Properties

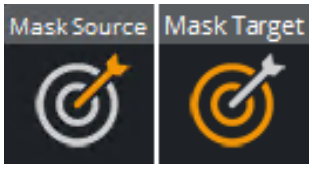
- **Type:** Defines the basic shape of your loupe. It can be either a rounded circle or a square rectangle shape.
- **Position:** Defines the location of the magnification. This can be either defined in Magnifier by modifying the x and y parameters or you use the attached containers position by activating the **Use container position** option.
- **Size:** Defines the location of the magnification. This can be either defined in Magnifier itself or you use the attached containers scale by activating the **Use container scale** option.

Copyright © 2026 Vizrt

- **Scale:** Determines the magnification level.
- **Ellipsis:** Defines how much distortion the lens has.
- **Color:** Defines the color of the magnified area. The color is additively blended.
- **Alpha:** Gives your magnifier a smooth border. A value of `1` gives a sharp edge and the lower the value, the smoother the edges are.
- **Tesselation:** Defines how exact the loupe is build. A value higher than `20` does not make any sense and provides no visual improvement.

Copyright © 2026 Vizrt

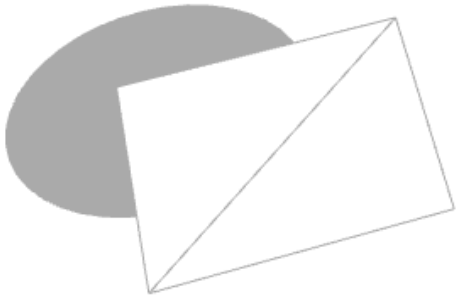
Mask Source and Mask Target



The Mask Source and Mark Target plug-ins allow one Container to act as a mask for other Containers.

The Mask Container is called the source and the Containers which are affected by the source mask are called targets.

A Container affected by a mask becomes transparent where the mask covers it.



Due to graphics hardware limitations, currently Viz Artist supports a maximum of eight different target and source layers for one object. This means that any target within a layer is affected by all masks for the same layer. For example, if a mask has the layers 1, 2 *and* 3 selected, it affects all targets that have 1, 2 *or* 3 selected.

Enable shadows in the Global Settings Panel.

Layers 7 and 8 may not be available because global shadow settings are enabled by default. Shadows use two layers of the mask plug-in, but can be set to `Off` in Global Settings Panel, to use all eight layers.

In the Global Settings Panel, define if a mask layer is to be drawn inverted. Normally the mask cuts a hole in the mask target where it covers it. If a layer is selected to be drawn inverted the opposite situation is the case. Only where the mask covers the mask target is target be visible.

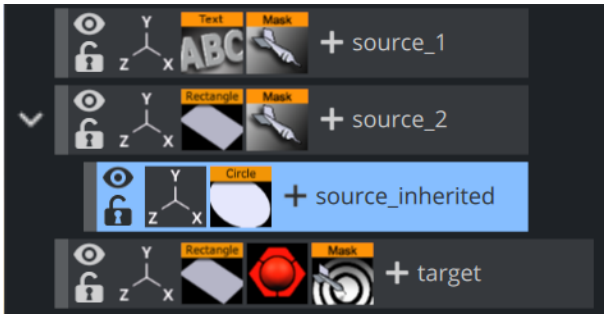
> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Global

Mask Source and Mask Target Properties

- **Layer:** Sets the layer the mask should have an effect on. This allows the creation of more than one mask that has an effect on the target.
- **Inherited:** (Mask Source Only) When enabled (`On`), all Sub-Containers of the mask source mask the target. If disabled (`Off`) all Sub-Containers of the mask source does not mask the target as it no longer inherits the mask properties.
- **Force Visible:** (Mask Source Only) When enabled (`On`), the geometry serving as mask source is rendered in the scene editor, even when not selected. When disabled (`Off`), the mask source geometry is only rendered when selected. This enables the designer to work more efficiently with multiple mask sources when designing the scene. The mask source geometry never gets rendered on the output.

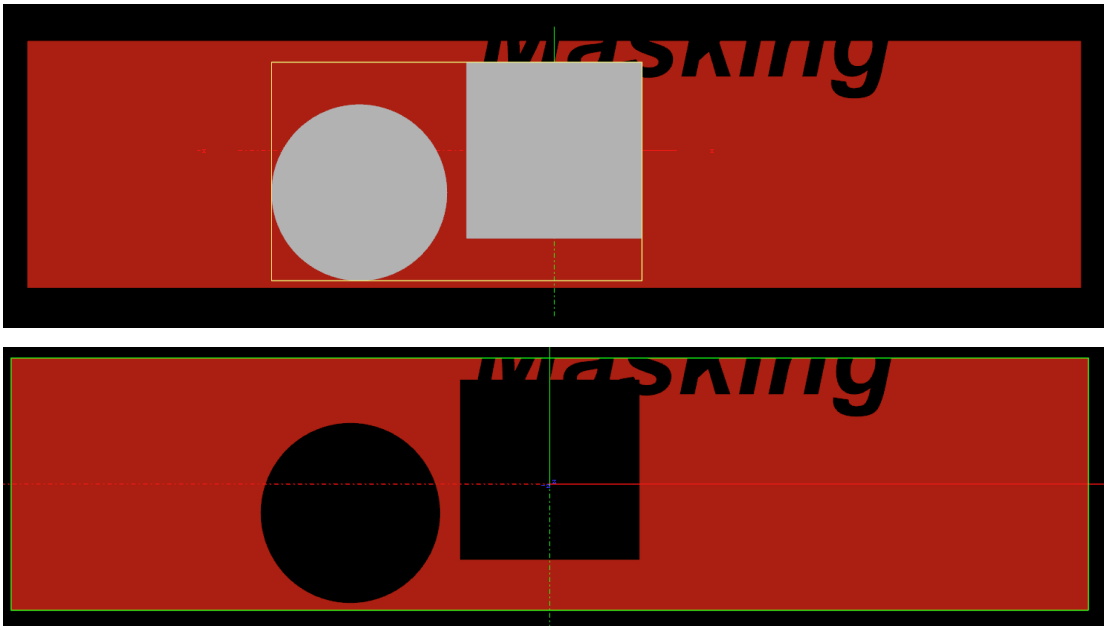
Copyright © 2026 Vizrt

To Add a Mask



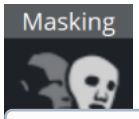
1. Add three Containers to the Scene Tree.
2. Name them `source_1`, `source_2` and `target`.
3. Add a Font and the Mask Source plug-in to the *source_1* Container.
4. Add a Rectangle and the Mask Source plug-in to the *source_2* group.
5. Add a Rectangle, material (for example, red) and the Mask Target plug-in to the *target* Container.
6. Open the *source_1* Transformation Editor:
    a. Set **Axis Center X** and **Y** to `C`.
    b. Reset **Position X** and **Y** to `0.0`.
7. Open the *source_1* Text Editor: Enter some text (for example, Vizrt).
8. Open the *source_1* Mask Source editor: Set *Layer* to `1`.
9. Open the *source_2* Transformation Editor: Set *Position X* to `-120.0` and *Y* to `-70.0`.
10. Open the *source_2* Rectangle editor: Set *Width* and *Height* to `50.0`.
11. Open the *source_2* Mask Source editor: Set *Layer* to `2`.
12. Open the *target* Rectangle editor: Set *Width* to `300.0` and *Height* to `200.0`.
13. Open the *target* Mask Target editor. Enable *Layer 1* and *Layer 2*.
14. Add a Container as a Sub-Container of *source_2*.
15. Name it `source_inherited`.
16. Add a Circle to the *source_inherited* Container.
17. Open the *source_inherited* Transformation Editor: Set *Position X* to `240.0`.
18. Open the *source_inherited* Circle editor: Set *Radius* to `25.0`.
19. Open the *source_2* Mask Source editor.
    a. Disable **Inherited** (`Off`). This should make the Circle disappear as it no longer inherits the Mask Source properties.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

Masking



The Masking plug-in provides a simple masking functionality for your Tonemapping effects.

Compatibility Info: This plug-in works in the Viz Engine Render Pipeline only!

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Masking can be used to turn off Tonemapping and Gamma correction for containers. This could be used for example on live sources, if Post Processing Effects should not be applied.

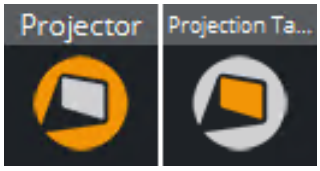
For example if a live input should not be affected by tonemapping or gamma correction:

- Create a container and add a PBR material.
- Drag a live source onto the emissive part of the PBR material.
- Apply Masking and turn off tonemapping and gamma correction.

Note: Masking works for opaque materials only. Transparency is not supported.

Copyright © 2026 Vizrt

Projector Source and Projector Target



The Projector Source and the Projector Target plug-ins are used to project something on something else (for example, an image onto a geometry).

A projector container creates a planar projection of their texture onto container(s) with the projection target function attached. In principle it works similar to a slide-projector. For objects that do not have texture coordinates, the projector function can be used to apply texture to such objects. It also gives the possibility to have two textures on a single object, the objects own texture and the projected texture.

The projector and the projection target can be set up to react on different layers similar to shadows and masks.

Note: The projector plug-in does not work on text.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Note: This plug-in has been deprecated in the Viz Engine Render Pipeline.

This page contains the following topics and procedures:

- Projector Source Properties
- Projector Target Properties
- To Project a Texture onto an Object

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Projector Source Properties

- **fovy:** Sets the field of view for the projector container. The higher the fovy is set, the larger the projected texture on the receiver is. You can picture it as the zoom function of a slide projector.
- **Show:** Allows you to hide/show the projector object.
- **Layer:** Lets you select by which layers the projector and receiver are to perform the function. You can use up to eight different projector layers and source layers for one object. This means that targets with the layer 1 set is affected by all projectors which have the corresponding layer selected. If a projector has the layers 1, 2 and 3 selected it affects all targets with 1, 2 or 3 selected.

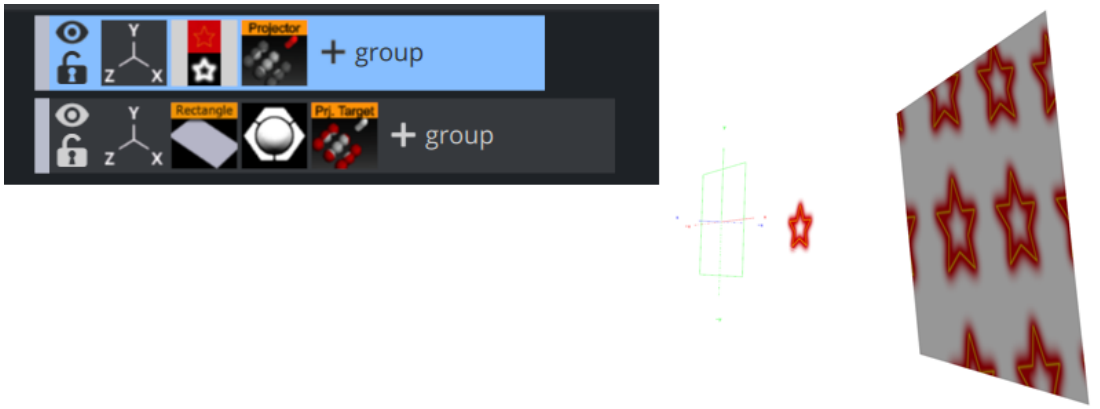
Projector Target Properties

- **Layer:** Lets you select by which layers the projector and receiver are to perform the function. You can use up to 8 different projector layers and source layers for one object. This means that targets with the layer 1 set is

Copyright © 2026 Vizrt

affected by all projectors which have the corresponding layer selected. If a projector has the layers 1, 2 and 3 selected it affects all targets with 1, 2 or 3 selected.

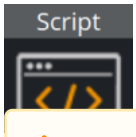
To Project a Texture onto an Object



1. Add a group container to the scene tree, add the Projector and Alpha to it, an image/texture, and name it `Source`.
2. Open the Alpha editor and set the alpha value to `50.0%`.
3. Open the transformation editor for the Source container and set Rotation Y to `135.0`.
4. Add another group to the scene tree, and name it `Target`.
5. Open the transformation editor for the Target container, and set Position X to `130.0`.
6. Add a Sphere geometry, material, image/texture and the Projector Target plug-in to the Target container.
7. *Optional*: Animate the sphere.
8. Open the Projector editor and **disable** the **Show** property.

Copyright © 2026 Vizrt

Script



The Script plug-in allows adding scripts to a Container and storing them with the Scene.

It is possible to archive and import scripts as part of a Scene Archive as with any other plug-in.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

A script can also be saved as its own script plug-in, and used in future Scenes. To save a script as a script plug-in, drag a compiled script into the Basic Container Script folder. The Script plug-ins folder is a special folder that lets users create and name plug-ins.

Note: Script plug-ins are saved to *<viz data folder>\Scriptplug-ins*.

WARNING: A script plug-in is not stored in the database, and is not distributed with an archive, because it is code and not data.

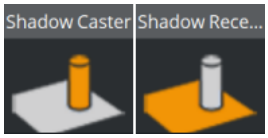
There is an important difference between a script that is not converted into a plug-in, and a script that is. As long as it is not converted the content of the script code is data within the Script plug-in, just like the width property is data within for example a geometry plug-in. Once the script is converted to a plug-in, it is a plug-in itself like any other.

See Also

- **Scripting** section of the Viz Artist User Guide

Copyright © 2026 Vizrt

Shadow Caster and Shadow Receiver



The Shadow Caster and Shadow Receiver plug-ins enable an object's shadow to reflect on another object.To create a shadow item in a scene, two containers at a minimum must be used. Shadow Caster must be attached to one container and Shadow Receiver to the other.

The Shadow Caster container must be positioned between the Shadow Receiver container and a light source.

For correct lighting calculation, a material is also required on the container. Objects that can function as a Shadow Receiver should be a built in 2D geometry or a font. An imported 2D geometry does not work.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Global

This page contains the following topics and procedures:

- Shadow Caster Properties
- Shadow Receiver Properties
- To Cast a Shadow
- Shadow Receiver Known Limitations

Shadow Caster Properties

- **Lights:** Allows you to select which light sources are to shine on Caster object and thereby producing the shadow/shadows.
- **Alpha:** Sets the alpha value and hence the transparency of the shadow being cast on the Receiver objects.
- **Color:** Allows you to define the shadows color.
- **Auto-Fading:** With this option enabled, if you fade out a Caster object either by using the alpha function or by adding a material with a low alpha value, the shadows created by the Caster fades correspondingly. The option is enabled by default, since it looks right that the shadow fades out when the Caster object fades out.
- **Layer:** Allows you to select by which layer(s) the Caster casts a shadow. You can select up to eight layers. Each layer corresponds to the layer selected in the Receiver objects. If you want all shadow Casters to cast shadow on all Receivers, you can set all layers on all Casters and Receivers to `1`. If you want one Caster to create shadow on some Receivers and another Caster to create shadow on different Receivers, you select different layers for the sets of Casters and corresponding Receivers. Casters and Receivers can have multiple layers enabled at the same time so you can set up a great number of combinations of Caster/Receiver combinations.

Shadow Receiver Properties

- **Alpha:** Sets the alpha value.
- **Clipping:** Cuts the shadow on the border of the object when set to `On`. When set to `Off`, the shadow is rendered onto the background outside the object, which normally looks wrong. Rendering with the clipping options set takes more resources than without. If you for instance have an infinite floor or a room with walls, you can switch of clipping without seeing the "false" shadows, and thereby save some rendering performance.

Copyright © 2026 Vizrt

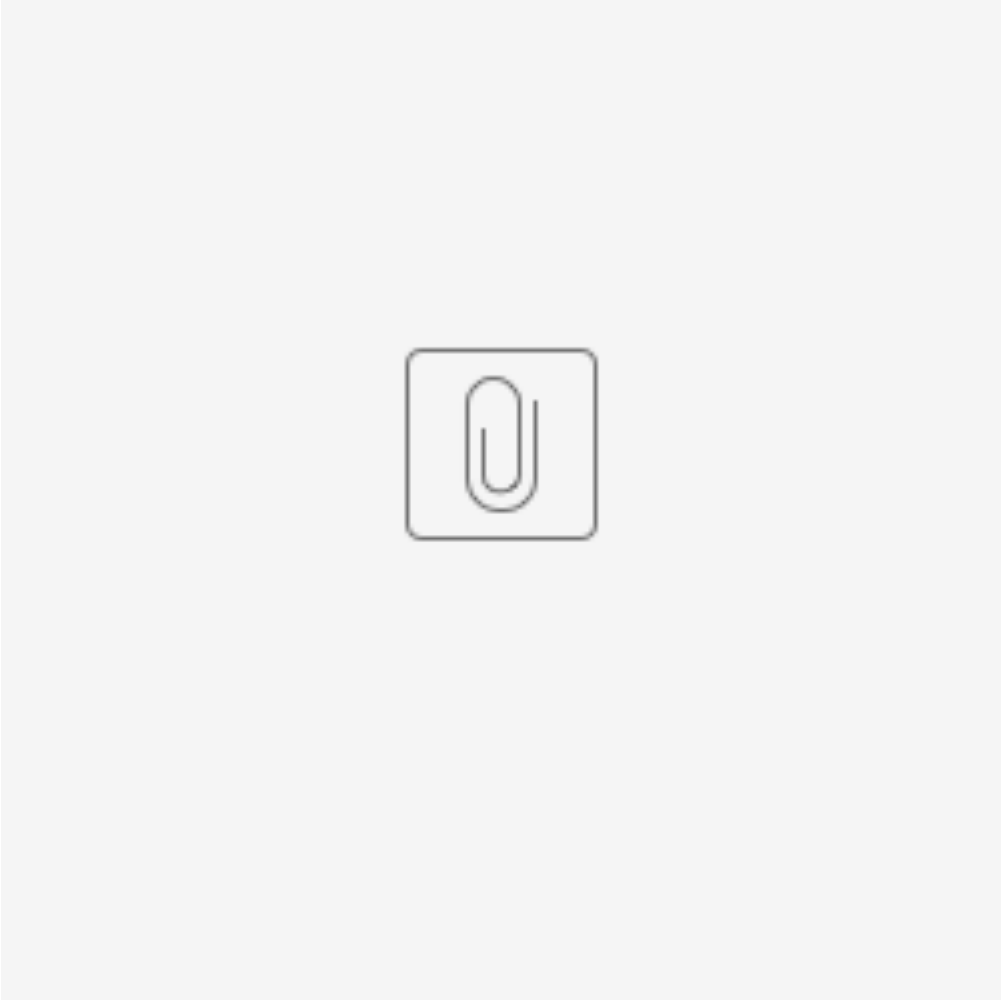
- **Culling:** Decides in advance of the rendering process if the shadow is visible or if it is hidden by other objects. If that is the case, it skips it in the rendering process. If you know for sure that the shadow for an object in a scene should be rendered visible for the whole object, you can disable culling to save some rendering time, but the effect is marginal.
- **Show Receiver:** Hides or shows the Receiver object. If it is hidden, the shadow is still visible.
- **Draw Key:** Adds key to the shadow. When enabled, the Receiver must be hidden for the key to be drawn.
- **Key Value:** Sets the key value.

> ⚠ **Note:** The Key Value setting is not available when Shadow Type in the Global Settings Panel is set to Shadow Map.

- **Layer:** Allows you to select by which layers the Receiver receives shadow. You can select up to eight layers. Each layer corresponds to the layer selected in the Caster objects. If you want all shadow Casters to cast shadow on all Receivers, you can set all layers on all Casters and Receivers to 1. If you want one Caster to create shadow on some Receivers and another Caster to create shadow on different Receivers, you select different layers for the sets of Casters and corresponding Receivers. Casters and Receivers can have multiple layers enabled at the same time so you can set up a great number of combinations of Caster- and Receiver combinations.

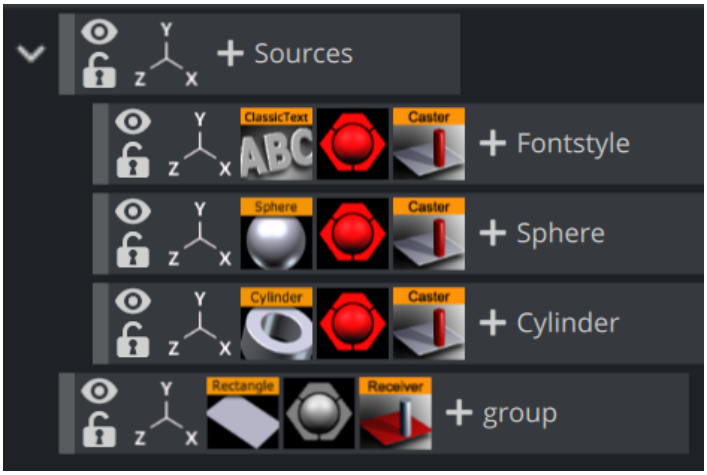
Copyright © 2026 Vizrt

To Cast a Shadow



1. Add a group container to the Scene Tree.
2. Name it `Sources`.
3. Add a Fontstyle and the Sphere and Cylinder Geometries as Sub-Containers of Sources.
4. Add a Material and Shadow Caster to each Sub-Container.
5. Add a Group Container at the same level as the *Sources* Container
6. Add a Rectangle Geometry, a Material and Shadow Receiver to it.

Copyright © 2026 Vizrt



7. Position the Rectangle and the other objects such that the Sphere, Cylinder, and Text casts a shadow on the rectangular surface.

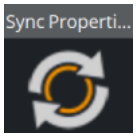
Caution: Containers are rendered twice when shadows are used.

Shadow Receiver Known Limitations

With shadows in Stencil Mode, the current implementation of the shadow Receiver assumes that the receiving surface is flat and a standard geometry object, such as a rectangle, orientated on the XY plane. Non-flat geometries, and planes which are not extending in the XY plane, does not produce a properly casted shadow. However, by activating Shadow Map for the desired light source, shadows render correctly. Always make sure that the receiving object has a material.

Copyright © 2026 Vizrt

Synchronized Properties



The Synchronized Properties plug-in can be used to synchronize container property changes within the same scene loaded on a cluster of Viz Engines. Synchronization needs to be configured for each Viz Engine taking part in the clustered setup. This plug-in works on a per container basis, which keeps network traffic low. Child containers do not inherit the plug-in, and requires their own instance of the plug-in if they are to be synchronized. A scene plug-in is available for use-cases where the entire scene is to be synchronized.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Example: A renderer with touch screen controls a stereoscopic interactive scene which is rendered on two other Engines. All three Engines need to perform all property or transformation changes at the same time.

This feature works for all container properties and plug-in parameters, and requires that the Engines render the same scene, referencing the same UUIDs. This allows designers to work with different versions of the same scene, without any need to update the changed IDs in external applications. If a parameter change is transferred to another Engine, then the container UUID of the last loaded scene is used for the update. If new UUIDs are required, a new scene has to be created, either from scratch, or by merging the complete scene into a container which is then split into a new scene.

To Synchronize Plug-in Property Changes on Multiple Viz Engines



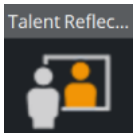
- Configure the involved Engines for synchronization.
- Create the scene that should be synchronized.
- Add Synchronized Properties to those containers that should be synchronized. No scripting is required for the synchronization.

See Also

- For more information about how to synchronize multiple Viz Engines, see the **Global Input** page in the **Configuring Viz** section of the Viz Engine Administrator Guide.

Copyright © 2026 Vizrt

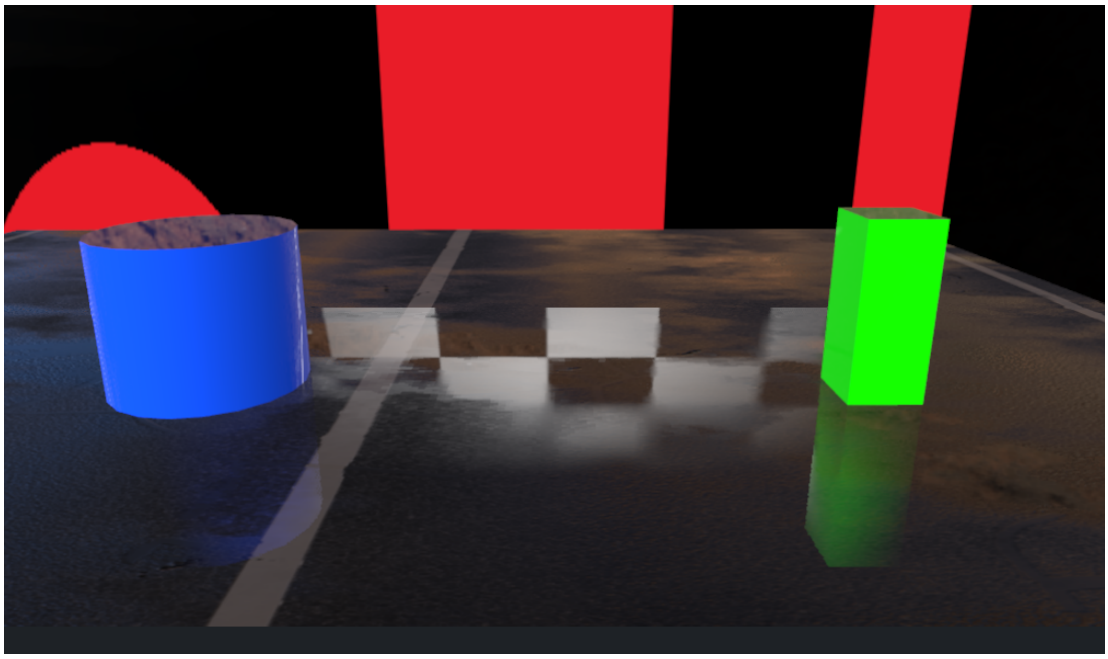
Talent Reflection



The Talent Reflection plug-in is used to render imaginary talent at a given depth only visible in reflections.

With this method, it is possible to render reflections of any live video input on Augmented Reality objects.

Below is an example of a rendered scene with talent reflections, a simple checkerboard texture is used as talent reflection input source.



Note: This plug-in is located in: Plugins -> Container plug-ins -> Global

Compatibility Info: This plug-in works in the Viz Engine Render Pipeline only!

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Talent Reflection Properties

- **Source:** Determines talent input source (Live input, Clip input, Image...).
- **Downscale:** Scales down rendered talent reflection to prevent ghosting artifacts.

Copyright © 2026 Vizrt

Usage

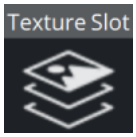
1. Enabling post-processing is required to see the effect in the Scene Editor.
2. Create at least one container with geometry and reflective material.
3. Drag Talent Reflection to a new container. This container is used for positioning. The distance from the container to the camera defines the position where the talent reflections are drawn.
4. Add any geometry as sub-container to see where its positioned inside the scene. This container can be hidden for final rendering.
5. Next, drag any image source (Live input, Clip input, Image...) into the Source placeholder.

Limitations

Normal Screen Space Reflections are not visible behind the talent reflection plane.

Copyright © 2026 Vizrt

Texture Slot



The Texture Slot plug-in allows multi-texturing like in the Classic Render Pipeline for Phong and PBR Materials.

You can create as many slotse as you want and use the textures in these slots to blend with a PBR or Phong material. It can be blended with all available textures in either the Phong or the PBR materials.

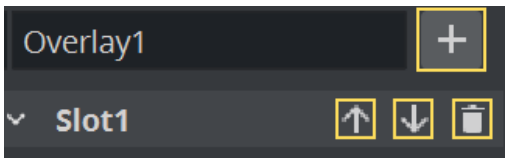
Note: This plug-in is located in: Plugins > Container plug-ins > Global.

Compatibility Info: This plug-in works in the Viz Engine Render Pipeline only!

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Plug-in Settings

To create a new Texture Slot, give it a unique name and hit the **+** button. The new slot is put on top of the existing ones. To reorder your slots, use the up and down arrows. To delete your slot, press the trash icon.



Note: Slots can **not** be renamed after creation.

- **sRGB Enabled:** Provides support for extended colorspace when enabled.

Blend Settings

The Target defines which texture unit you want to blend with.

Target	Available in
BaseColor	PBR/Phong
Emissive	Phong

Information: The Ambient Occlusion texture is not accessible through the Texture Slot plug-in by design.

Copyright © 2026 Vizrt

You can simulate an animated height effect if you, for example, blend an additional normal map to an existing one. The **Color Blend** mode defines how your image is blended with the **Target** image.

Target	Slot
	

Basic Operations

Operation	Explanation	Sample
Keep	Keeps the Target image.	
Replace	Replaces the Target with the Slot image.	

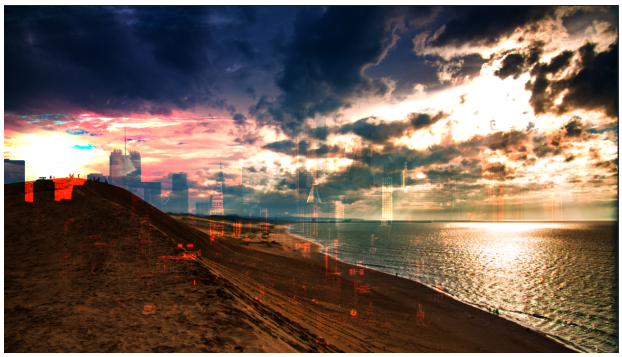
Copyright © 2026 Vizrt

Operation	Explanation	Sample
Average	Uses the average pixel values of both images.	
Normal	*Replaces the Target with the Slot image. Same as Replace*	

Lightening Operations

Operation	Explanation	Sample
Add	Adds the Slot image to the Target image.	
Lighten	Uses the lighter pixel value of both images.	

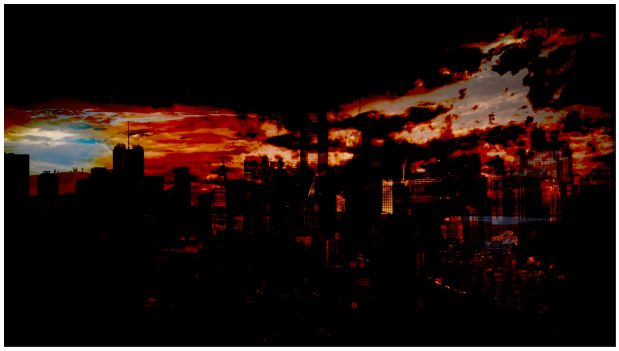
Copyright © 2026 Vizrt

Operation	Explanation	Sample
Screen	With Screen blend mode, the values of the pixels in the two layers are inverted, multiplied, and then inverted again. The result is the opposite of Multiply: wherever either layer was darker than white, the composite is brighter.	
ColorDodge	The Color Dodge blend mode divides the Target by the inverted Slot. This lightens the Target layer depending on the value of the Slot layer: the brighter the Slot, the more its color affects the Target. Blending any color with white gives white. Blending with black does not change the image.	
LinearDodge	Brightens the Target color to reflect the Slot color by increasing the brightness. Blending with black does not have any influence - See LinearBurn	
Reflect	This mode is useful when adding shining objects or light zones to images.	

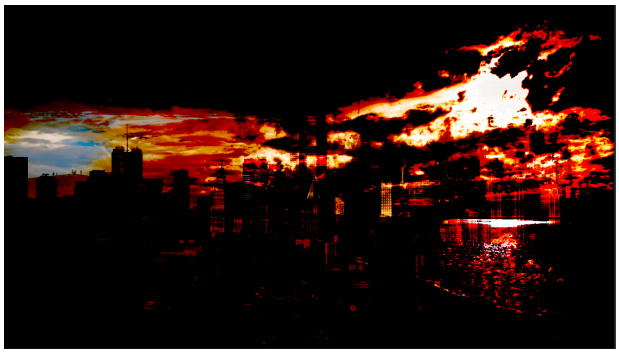
Copyright © 2026 Vizrt

Operation	Explanation	Sample
Glow	This mode is a variation of reflect mode (base and blend color swapped).	

Darkening Operations

Operation	Explanation	Sample
Subtract	Subtracts the Slot image from the Target Image.	
Darken	Uses the darker pixel value of both images	

Copyright © 2026 Vizrt

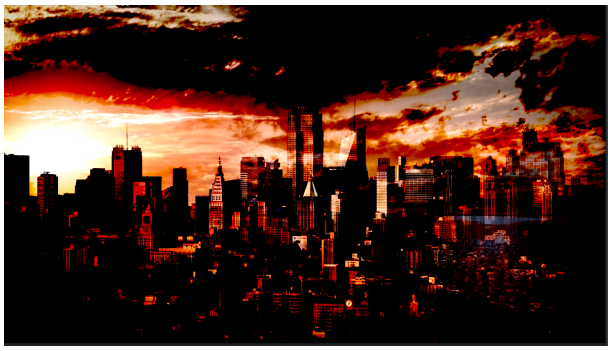
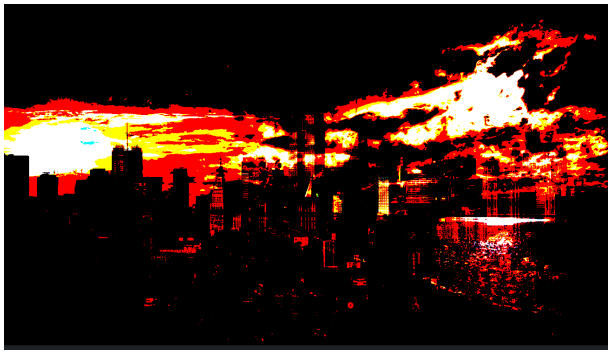
Operation	Explanation	Sample
Multiply	Multiplies the Slot image with the Target image.	
ColorBurn	The Color Burn mode divides the inverted Target by the Slot image, and then inverts the result. This darkens the Slot image increasing the contrast to reflect the color of the Target image. The darker the Target layer, the more its color is used. Blending with white produces no difference.	
LinearBurn	Darkens the Target color to reflect the Slot color by decreasing the brightness. Blending with white does not have any influence - See LinearDodge	

Contrast Effects

Copyright © 2026 Vizrt

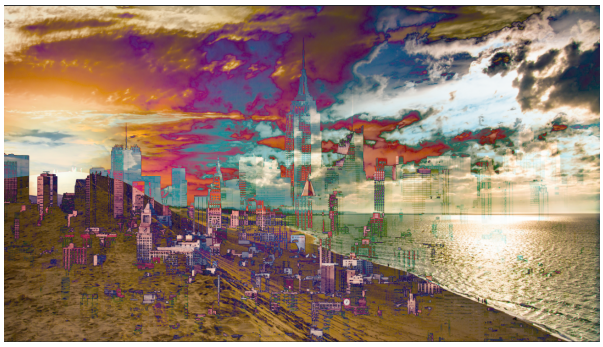
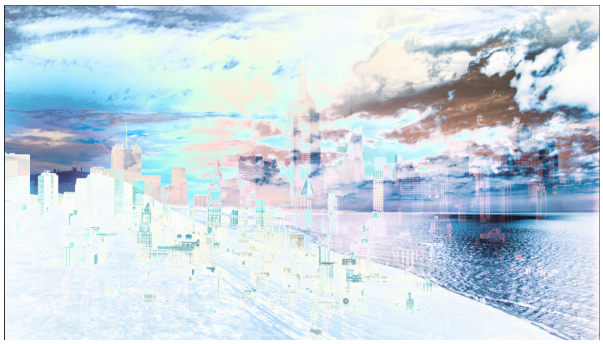
Operation	Explanation	Sample
Overlay	Uses Screen blending on lighter pixels and Multiply mode on darker pixels. It is the opposite operation to the Hard Light blending.	
SoftLight	Similar to Overlay, but appears softer.	
HardLight	Uses Linear-Doge on lighter pixels and Linear Burn on on darker pixels. It is more intense than the Overlay and results in harsher lights.	
VividLight	Uses Color Doge on lighter pixels and Color Burn on on darker pixels. It appears more extreme than the Hard Mix Effect.	

Copyright © 2026 Vizrt

Operation	Explanation	Sample
LinearLight	Uses Linear Dodge blend mode on the lighter pixels and the Linear Burn blend mode on the darker pixels. Similar to the Vivid Light blend mode in overdrive, and typically results in a more extreme effect.	
HardMix	Uses Linear Light as threshold. It is recommended to adjust the Blend Factor to achieve some great results.	
PinLight	Lightens lighter pixels and darkens darker pixels. This is a wild blend mode that can result in patches or blotches (large noise), and it completely removes all mid-tones.	

Inversion Effects

Copyright © 2026 Vizrt

Operation	Explanation	Sample
Difference	Difference subtracts the Target from the Slot or the other way around, to always get a non-negative value. Blending with black produces no change, as values for all colors are 0. Blending with white inverts the picture.	
Exclusion	This mode is basically the same as the Difference blend mode, except when similar colors cancel each other, the resulting color is gray instead of black.	
Phoenix	This subtracts the lighter pixel from the darker pixel, and adds 255, giving a bright result.	
Negation		

Alpha Blend mode defines how the Alpha Channel of the Slot image will be used:

Copyright © 2026 Vizrt

- **Keep:** Keeps the Target Alpha.
- **Replace:** Uses the Slot Alpha.
- **Add:** Adds both Slot and Target Alpha channel.
- **Multiply:** Multiplies both Alpha channels..
- **Subtract:** Subtracts the Slot Alpha Channel from the Target Channel.
- **Reverse Subtract:** Like subtract, but inverted afterwards.
- **Negate:** Blend effect defines how strongly the operation is applied to the Target image. This is defined from `0` (no effect at all) to `1` (fully).

Swizzle

The Swizzle part allows to exchange the various color channels. You can for instance ignore an alpha channel or fill it completely white (1) or black (0).

Texture Coordinates

This defines how the final composition is applied to the Target Texture. Usually this is done in Vertex mode, but you can also overwrite it, for example to use the Slot Image as Reflection map. Additional mapping options can be achieved by using offset, scaling and rotation.

See **Introduction to Viz Engine Rendering > Working with Phong Materials > Texture Generation** in the Viz Artist User Guide for more information.

Copyright © 2026 Vizrt

Video Clip



The Video Clip plug-in works in combination with Control Video. It allows you to use clip Key Frames in the stage for clip playback. When used with Transition Logic, the Key Frames can be merged with the object. When Video Clip is used in Transition Logic scenes, it allows a Toggle between the two clip channels. A single object playbacks any clip. In a Transition Logic scene, Toggle makes sure that the channels toggle correctly, which allows for transitions between two running clips. The plug-in also toggles the clip texture if texture mode is selected.

Video Clip is commonly used in a video workflow with Viz One. It automatically adds a Clip Channel plug-in. The clip channel has to be activated using the **Video Input** in Viz Configuration (see the Viz Engine Administrator Guide).

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Global

Video Clip Properties

- **Channel:** Sets the clip channel to be used. Available options are; Clip channel 1 and 2.
- **Target:** Sets the clip mode. Setting target to DVE adds default Key Frames to the stage. Setting target to Texture adds the **Live Video** Media Asset texture plug-in and default Key Frames to the stage. Available options are; Inactive, DVE or Texture.
- **Volume:** Sets the clip volume in percentage.

To Add a Video Clip

1. Check the Viz Configuration to see if clip channels are activated.
2. Add the Video Clip plug-in to a geometry (for example, a Rectangle).
3. Select Clip Channel 1 and Type Texture.
4. Switch to the Stage and select the Clip under VideoClip.
    - On the right side the clip to be played can be selected.
    - Additional Key Frames can be added to play different clips.

See Also

- Toggle
- Control Video

Copyright © 2026 Vizrt

Virtual Window



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.

Virtual Window Properties

Projection

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:



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

Copyright © 2026 Vizrt

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.

Copyright © 2026 Vizrt

Window Mask



The Window Mask plug-in limits the area into which a Sub-Container of Window Mask is rendered to a user-defined rectangular section of the output.

It is possible to add a tracking object. If a tracking object is used, it's bounding box is projected onto the screen and the resulting rectangle is used as a mask.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Global

Window Mask Properties

- **Position X, Position Y:** Specifies the upper left corner of the rectangle. The values are scaled in such a way that `(0,0)` refers to the upper left and `(100,100)` to the bottom right corner of the screen.
- **Width, Height:** Specifies the dimensions of the rectangle in percentages of the screen dimensions.
- Tracking: Provides an alternative to setting the position and dimensions by a bounding box of a particular container.
    - **Origin:** Specifies where to take a container as a tracking object. Apart from Custom, three neighborhoods of the container holding Window Mask are available:
        - **Previous:** Adjacent container above as shown in the scene tree.
        - **Next:** Adjacent container below as shown in the scene tree.
        - **First Child:** Adjacent child container as shown in the scene tree.
    - **Object:** Specifies a container from the scene tree to be a tracking object. The specified container is taken into consideration if and only if *Origin* is set to *Custom*. To discard the specified container, click on **Reset**.

> **Note:** The underlying mask operation is integral. Use other masks like Mask Source / Mask Target if you need subpixel accuracy.

Copyright © 2026 Vizrt

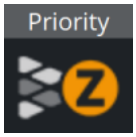
To Use the Window Mask



1. Add a Rectangle geometry to the scene tree, and add material to it.
2. Open the Rectangle editor and set Width to `200.0`.
3. Open the transformation editor and set Position Y to `-36`.
4. Add a new group, and add the Window Mask plug-in to it.
5. Add a Sphere geometry as a Sub-Container of the group container, and add material to it.
6. Open the Window Mask editor, activate Tracking, drag, and drop the Rectangle container onto the Object drop zone. This should give you a sphere that is partly masked due to the rectangle object.

Copyright © 2026 Vizrt

Z-Sort Priority



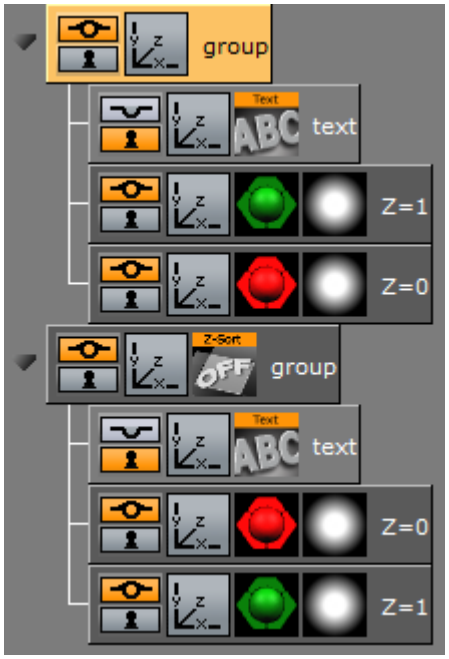
The Z-Sort Priority plug-in disables back to front automatic sorting for the immediate children of that group when applied to a group.

This enables designers to set the rendering order manually.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Global

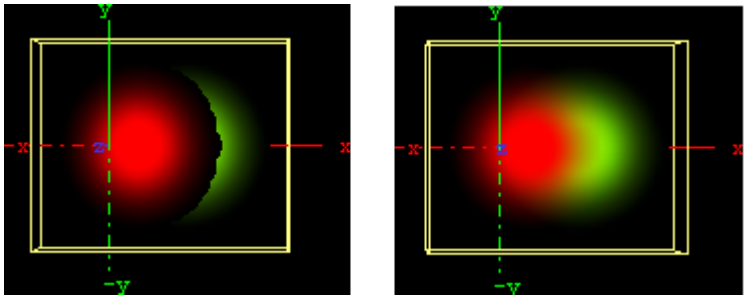
Z-Sort Priority Description

To sort objects of a Scene on the Z-axis is essential for any 3D program to render correctly. The renderer starts by rendering the object furthest behind, and then mixing in objects closer and closer to the camera. Sometimes the machine gets “confused” with what should be rendered first and last because it calculates this from what is the center of the object’s Z position.



A typical example of a Z sort problem is when two colored objects have rotated their Y axis to for example `-12.0`. In this case the default behavior for a render engine is to render the first object (green) with a Z position `0.0` first, and then the second object (red) with Z position `1.0` last.

Copyright © 2026 Vizrt



When objects are rotated, the output is wrong because the red object should be rendered before the green due to the rotation of the objects. This can be corrected by adding Z-Sort Priority, which lets users manually sort objects using the logic of the tree structure. Hence, what is placed at the top of a group in the scene tree gets rendered first, and what is placed below is rendered after.

Sorting is solved by rendering the second (red) object first, and then the first (green) last. Be aware that although the red object has a Z position of `1.0` and the green `0.0`, the red object must be rendered before the green because of the overlapping rotation (Y `-12.0`). Normally objects that are furthest behind should be placed at the top in the scene tree.

Z-Sort Priority disables the automatic Z-sort for a Container. How the items in the Container get sorted in the Z-buffer is then dictated by the order of the objects in the scene tree.

Transparency, Rendering Order Logic and Priority

Questions

- Why does a scene look different when a parent container is set to 99% transparency?
- Why are do some Containers disappear?
- When should Z-Sort Priority be used?

Explanation

The explanation to why changing an alpha plug-in to 99% on a parent container, which affects the way objects are rendered, is the Rendering Order. When rendering polygonal Scenes with transparencies (like those created in Viz Artist) the rendering order is important with the current architecture of graphics cards.

The common practice for 3D applications is to render semi-transparent objects from back to front (or rather from far to near).

Example: Think of a scene that shows a city skyline through of a semi-transparent window. Lets assume that the window is rendered **before** the skyline. Some background color is applied to the screen, then the window is rendered. The graphics card blends the color of the window with the background color (say, blue). In addition to color, the graphics card also marks the distance of the window's pixels from the camera for each pixel, so that farther away pixels do not obstruct the window. This technique for achieving obstruction between objects on per-pixel basis is called Depth-Buffer or Z-Buffer. Now render the skyline. The color of the skyline should have been blended with the color of the window, but it does not. The graphics card rejects every skyline pixel because its distance from the camera is farther away than the distance of the window pixels. This is a case where the positive role of the Depth-Buffer produces undesirable results.

Copyright © 2026 Vizrt

To avoid this undesirable result, Viz Artist splits the rendering of the containers in the Scene into three parts:

1. The mask objects are rendered first to ‘mark’ the areas of masking on the screen.
2. The opaque objects are rendered front to back.
3. The transparent objects are rendered back to front.

Viz Artist/Viz Engine decides whether an object is transparent based on some criteria, for example:

- If the texture of the object has an alpha channel.
- If the material of the object has an alpha value less than 100%.
- If the object has an inherited ALPHA function with an alpha value less than `100`.
- and more...

With all this said, rendering back to front is not perfect because the rendering order is not done per pixel; rendering is not even done per polygon. Viz Artist/Engine renders a scene per complete container; container order is based on the center of the container.

Example: Take a tilted flat map with icons. The map container is large, and its center is in front of some of the icons, but in the back of other icons. Assuming the icons have transparency, the map is actually rendered in between the icons. This behavior is visually incorrect as the map should be rendered first, then all the icons.

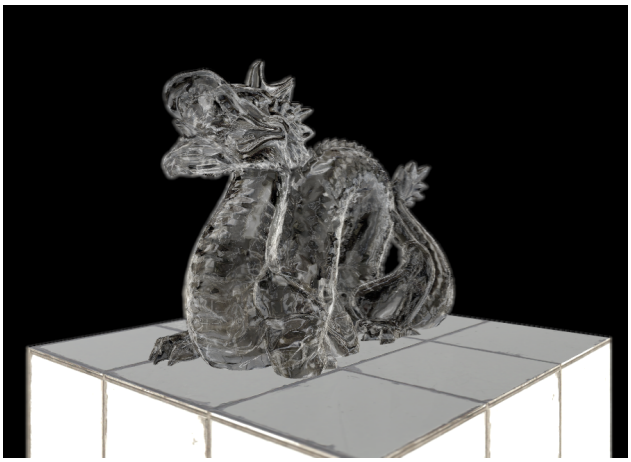
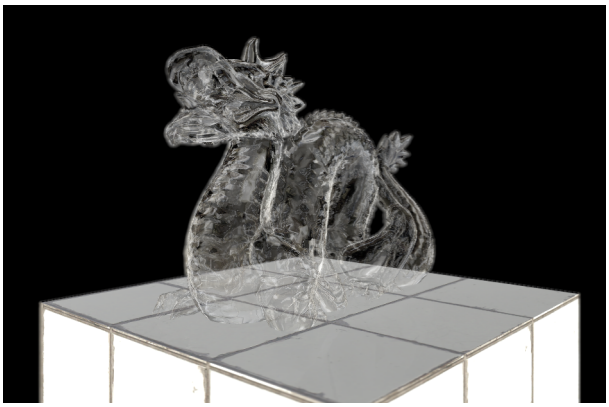
Solution

This behavior can be fine-tuned using Z-Sort Priority. When Z-Sort Priority is applied to a group, it disables back to front automatic sorting for the immediate children of that group. This enables designers to set the rendering order manually.

Examples

The following shows an incorrect rendering of a transparent object on a cube. The Z-ordering is wrong, therefore the transparent parts on the cube appears incorrectly (black). This might only happen when flying around it. When adding Z-Sort Priority to the parent container and reordering the tree, the cube is to be rendered first and the dragon afterwards, the transparency issues are fixed and the dragon appears correctly on the cube, even when rotating.

Copyright © 2026 Vizrt

Incorrect	Fixed
	
	

Notes/Exceptions

- If an Alpha plug-in at 99% is applied on the top container, it can force the whole collection of containers to be sorted back to front, and can solve the problem, but this is not a best practice. There is much more overhead rendering a scene like this. A more permanent solution would be to optimize the scene with Z-Sort Priority.
- Opaque objects are better rendered from front to back because there is some performance gain for each obstructed pixel (it is not being processed after being rejected). With that being said opaque objects can be rendered at any order.

Information: Do not place Z-Sort Priority on top of your scene. Adding Z-Sort Priority disables sorting and optimizing the geometry for rendering and can have a negative impact to your render performance. Please use Z-Sort Priority only where absolutely needed as it disables performance optimizations. A common solution for Z-Sort Priority related issues could also be to change the center point of your geometry.

Copyright © 2026 Vizrt

2.4.8 MultiTouch Container Plug-ins

The MultiTouch plug-ins are used to take incoming touch events, process them and then trigger events, perform transformations on objects, telestrate, or trigger script events. The idea behind these plug-ins is to bundle often used or needed functionality and eliminate the need to write your own scripts or plug-ins for that. They are great tools to bring MultiTouch interactively fast and easily into a Scene. The Container plug-ins require the MtSensor to be available to work. If a MultiTouch plug-in is added to a Scene it automatically adds the MtSensor to the Scene if it is not there.

The MultiTouch plug-ins can communicate with Viz scripts, either through event handlers, Shared Memory variables, or Data Pool variables. The plug-ins can also be controlled by triggering the 'push buttons' and/or setting the 'parameter' variables exposed by the plug-in. Specifics on what features are available for each plug-in are documented in the plug-in specific sections below.

In the Viz Artist editor interface, the  button toggles between enabling and disabling emulated sensor input using the mouse as well as direct input from a sensor. The plug-ins work with multiple layers. Each Scene layer with MultiTouch objects installed in it should also have an instance of the MtSensor automatically installed.

The following Container plug-ins are located in the MultiTouchComp folder:

- Mt2D Control
- Mt3D Control
- MtButton
- MtNavigator
- MtTelestrator
- Plug-in Event and Notification System

Make sure you also read up on the Plug-in Event and Notification System.

Copyright © 2026 Vizrt

Mt2D Control



The Mt2D Control plug-in moves, scales, and rotates an object in 2D. Mt2D Control allows multi-finger 2D manipulation of an object in the scene. When the object is touched, it translates, rotates, and scales to keep the object underneath the user's fingers as closely as possible. It moves within the screen relative 2D plane through the object's center position defined by the objects orientation axes.

The container being manipulated must not have any local non-uniform scale. Mt2D Control forces y and z scale so they are equal to x scale when the object is touched. If non uniform scaling is required on object in the container, put the object's container in a group and scale the parent. Or put Mt2D Control on an identity group and insert the object to be moved as a child container. All transformations set on the manipulated container's parents are preserved.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> MultiTouchComp

This page contains the following topics and procedures:

- Mt2D Control Configuration
- Events and Notifications
- To Create a Simple Scene with Mt2D Control

Mt2D Control Configuration

- **Momentum:** Enables movement or rotation with momentum after the object is released.
  - **Friction:** Sets the amount of friction used when momentum is enabled.

> ⚠ **Note:** Using this feature conflicts with any externally scripted or other animations applied to the container.

- **Lock Translation:** Allows only rotation and scaling of the object.
- **Lock Translation (Screen X):** Prevents movement left-to-right on the screen.
- **Lock Translation (Screen Y):** Prevents movement up-and-down on the screen.
- **Lock Translation (Object X):** Prevents movement along the object's local X axis.
- **Lock Translation (Object Y):** Prevents movement along the object's local Y axis.
- **Lock Rotation:** Prevents rotation of the object.
- **Lock Scale:** Prevents resizing of the object.
- **Limit Position:** Limits the maximum and minimum position of the X and Y axis.
- **Limit Rotation:** Limits the maximum and minimum rotation of the X and Y axis.
- **Limit Scale:** Limits the maximum and minimum scale.
- **Id:** Gives additional context in the handler script, specify a string that identifies any notifications dispatched by Mt2D Control. This is often included as an argument for the event so a common script may handle events from a number of plug-ins.
- **Set Shared Memory:** Enables shared memory to be updated for the plug-in notifications when set to `On`.
  - **Shared Memory Prefix:** This sets a 'prefix name' to be prepended to the shared memory variables maintained by Mt2D Control notifications. For plug-ins that maintain multiple fields each field name has the prefix prepended to it followed by a dot '.' so as to mimic member access to an object: if the prefix is 'Obj' the fields 'field1' and 'field2' would be identified with the strings 'Obj.field1' and

Copyright © 2026 Vizrt

'Obj.field2'. The shared memory field 'Obj' is also maintained and is simply an integer that is modified every time any of its 'subfields' is updated.

- **Shared Memory Type:** Click either Global, Scene, or Distributed. This selects the shared memory area to update.

- **Set Data Pool:** Shows 'you wish' plug-in notifications to set a Data Pool variable.
  - **Data Pool Variable:** Shows the name of the Data Pool variable to have set.
- **Active during screen wide telestration?:** If the button to is to stay active when the screen-wide telestrator is on, set to On. The default is off, which means the button cannot be triggered when telestration is on.
- **Hit Coordinate Type:** Selects the Coordinate system to use. Click either Local, World or Screen.
- **Produce delta values, don't move object:** Does not alter transform of object with attached controller, but instead provide 'deltas' in the event notifications.

> ⚠ **Note:** Rotation scale or position constraints do not operate in this mode.

Events and Notifications

The events dispatched by Mt2D Control are:

OnGrab (when object is 'grabbed'):

sub **ONGrab** (**HitContainer as String, Id as String, HitPoint as Vertex, plug-inContainer as String, FirstChildContainer as String**)

end sub

- **HitContainer as String**
- **Id as String**
- **HitPoint as Vertex**
- **plug-inContainer as String**
- **FirstChildContainer as String**

OnGrabUpdate (when object is moved):

sub **OnGrabUpdate** (**HitContainer as String, Id as String, Position as Vertex, Rotation as Vertex, Scale as Vertex, Pressure as Double**)

end sub

- **HitContainer as String**
- **Id as String**
- **Position as Vertex**
- **Rotation as Vertex**
- **Scale as Vertex**
- **Pressure as Double**

OnAnimation (dispatched when post grab animation (momentum) is started, followed by OnGrabUpdates):

sub **OnAnimation** (**HitContainer as String, Id as String**)

end sub

- **HitContainer as String**
- **Id as String**

OnGrabRelease (when both touch grab and momentum animation are complete):

Copyright © 2026 Vizrt

sub **OnAnimation** (**HitContainer as String, Id as String**)

end sub

- **HitContainer as String**
- **Id as String**

The Shared Memory Field names updated are:

- **HitContainer** (the name of the container that was 'hit')
- **Id** (the user ID string entered in the notification interface)
- **HitPoint**
- **plug-inContainer**
- **FirstChildContainer**
- **Position**
- **Rotation**
- **Scale**
- **Pressure**
- **State**

The Data Pool structure updated is:

- **Mt2D Control:**
  - string HitContainer
  - string Id
  - string HitPoint
  - string plug-inContainer
  - string FirstChildContainer
  - string Position
  - string Rotation
  - string Scale
  - string Pressure
  - string State

To Create a Simple Scene with Mt2D Control

> **Note:** If a touch screen is not available, set **Multi Touch** input to **Mouse** in the **Communication** section of Viz Configuration (see the Viz Engine Administrator Guide).

1. Create a new Scene.
2. Add a new group Container.
3. Add a Cube Geometry to the Container.
4. Add Mt2DControl to the Container.
5. Open the Scene in **On Air**. Move, rotate or scale the Cube with the touch input device.
6. Use the Mt2DControl editor to modify the plug-in behavior as required.

Copyright © 2026 Vizrt

Mt3D Control



The Mt3D Control plug-in moves, scales, and rotates an object in 3D.

Mt3D Control provides an arc ball-like controller for the object it is attached to. When used with one finger, the object can be rotated. When multiple fingers are used, the object can be moved, rotated and scaled in a 2D plane (the same as the Mt2D Control).

Note: This plug-in is located in: Plugins -> Container plug-ins -> MultiTouchComp

This page contains the following topics and procedures:

- Mt3D Configuration
- 2D Configuration
- Output Configuration
- Events and Notifications
- To Create a Simple Scene with Mt3D Control

Mt3D Configuration

- **Config:** Selects 3D Config
- **Axis Lock:** Constrains the rotation to a specific axis, or allow free rotation. Options are:
  - Off
  - X Axis
  - Y Axis
  - Z Axis
- **Axis Lock Local:** Uses the local axis instead of the world axis if axis lock is on.
- **Active during screen wide telestration?:** If the button to is to stay active when the screen-wide telestrator is on, set to On. The default is off, which means the button cannot be triggered when telestration is on.
- **Hit Coordinate Type:** Selects the Coordinate system to use. Click either Local, World or Screen.

2D Configuration

- **Config:** Selects configuration.
- **Momentum:** Enables movement or rotation with momentum after the object is released.
  - **Friction:** Sets the amount of friction used when momentum is enabled.

    Note: Using this feature conflicts with any externally scripted or other animations applied to the container.
- **Lock Translation:** Allows only rotation and scaling of the object
- **Lock Translation (Screen X):** Prevents movement left-to-right on the screen
- **Lock Translation (Screen Y):** Prevents movement up-and-down on the screen
- **Lock Translation (Object X):** Prevents movement along the object's local X axis
- **Lock Translation (Object Y):** Prevents movement along the object's local Y axis
- **Lock Rotation:** Prevents rotation of the object
- **Lock Scale:** Prevents resizing of the object.

Copyright © 2026 Vizrt

- **Active during screen wide telestration**?: Determines whether the button to stays active when the screen-wide telestrator is on, when set to `On`. The default is `Off`, which means the button cannot be triggered when telestration is on.
- **Hit Coordinate Type:** Selects the Coordinate system to use.Click either Local, World or Screen.

Output Configuration

- **Config:** Selects configuration.
- **Id:** Gives additional context in the handler script, specify a string that identifies any notifications dispatched by this plug-in. This is often included as an argument for the event so a common script may handle events from a number of plug-ins.
- **Set Shared Memory:** Enables shared memory to be updated for the plug-in notifications when set to `On`.
    - **Shared Memory Prefix:** This sets a 'prefix name' to be prepended to the shared memory variables maintained by the plug-ins notifications. For plug-ins that maintain multiple fields each field name has the prefix prepended to it followed by a dot '.' so as to mimic member access to an object: If the prefix is 'Obj' the fields 'field1' and 'field2' would be identified with the strings 'Obj.field1' and 'Obj.field2'. The shared memory field 'Obj' is also maintained and is simply an integer that is modified every time any of its 'subfields' is updated.
    - **Shared Memory Type:** Click either Global, Scene, or Distributed. This selects the shared memory area to update.
- **Set Data Pool:** Shows 'you wish' plug-in notifications to set a Data Pool variable
    - **Data Pool Variable:** Shows the name of the Data Pool variable to have set.
- **Active during screen wide telestration?:** Determines whether the button stays active when the screen-wide telestrator is on, when set to `On`. The default is `Off`, which means the button cannot be triggered when telestration is on.
- **Hit Coordinate Type:** Selects the Coordinate system to use. Click either Local, World or Screen.

Events and Notifications

The events dispatched by Mt3dControl are:

OnGrab (when object is 'grabbed'):

sub OnGrab (**HitContainer as String, Id as String, HitPoint as Vertex, plug-inContainer as String, FirstChildContainer as String**)

end sub

- **HitContainer as String**
- **Id as String**
- **HitPoint as Vertex**
- **plug-inContainer as String**
- **FirstChildContainer as String**

OnGrabUpdate (when object is moved):

sub **OnGrabUpdate** (**HitContainer as String, Id as String, Position as Vertex, Rotation as Vertex, Scale as Vertex, Pressure as Double**)

end sub

Copyright © 2026 Vizrt

- **HitContainer as String**
- **Id as String**
- **Position as Vertex**
- **Rotation as Vertex**
- **Scale as Vertex**
- **Pressure as Double**

OnGrabRelease (when object is “released”):

sub OnGrabRelease (HitContainer as String, Id as String)

end sub

- **HitContainer as String**
- **Id as String**

The Shared Memory Field names updated are:

- HitContainer (the name of the container that was “hit”)
- Id (the user ID string entered in the notification interface)
- HitPoint
- plug-inContainer
- FirstChildContainer
- Position
- Rotation
- Scale
- Pressure
- State

The Data Pool structure updated is:

- Mt3dControl:
  - string HitContainer
  - string Id
  - string HitPoint
  - string plug-inContainer
  - string FirstChildContainer
  - string Position
  - string Rotation
  - string Scale
  - string Pressure
  - string State

To Create a Simple Scene with Mt3D Control

Note: If a touch screen is not available, set **Multi Touch** input to **Mouse** in the **Communication** section of Viz Configuration (see the Viz Engine Administrator Guide).

1. Create a new Scene.
2. Add a new group Container.
3. Add a Cube Geometry to the Container.

Copyright © 2026 Vizrt

4. Add Mt3D Control to the Container.
5. Open the Scene in **On Air**. Use the touch input device to transform the object.
6. Use the Mt3D Control editor to modify the plug-in behavior, as required.

Copyright © 2026 Vizrt

MtButton



The MtButton plug-in triggers an action when tapped or pressed. MtButton makes a container or its immediate children act as buttons.

Several methods may be used to trigger the buttons. They are selected in the Viz Artist UI with the 'Trigger Type' radio button. The MtButton can be enabled or disabled. When the button is disabled, any tap or stroke input it receives does not trigger the associated events. The button can be enabled or disabled through MtButton parameters, or through a command. A button which is disabled after a stroke has pressed on a button does NOT prevent generation of events for a stroke that is currently active.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> MultiTouchComp

This page contains the following topics and procedures:

- MtButton Configuration
- Events and Notifications
- To Create a Simple Scene with the MtButton Plug-in

MtButton Configuration

- **Id:** Gives additional context in the handler script, specify a string that identifies any notifications dispatched by MtButton. This is often included as an argument for the event so a common script may handle events from a number of plug-ins.
- **Set Shared Memory:** Enables shared memory to be updated for MtButton notifications when set to `On`.
  - **Shared Memory Prefix:** Sets a 'prefix name' to be prepended to the shared memory variables maintained by MtButton notifications. For plug-ins that maintain multiple fields each field name has the prefix prepended to it followed by a dot (**.**) so as to mimic member access to an object (if the prefix is 'Obj' the fields *field1* and *field2* would be identified with the strings *Obj.field1* and *Obj.field2*. The shared memory field 'Obj' is also maintained and is simply an integer that is modified every time any of its 'subfields' is updated).
  - **Shared Memory Type:** Selects the shared memory area to update. Click either Global, Scene, or Distributed.
- **Set Data Pool:** Shows 'you wish' plug-in notifications to set a Data Pool variable.
  - **Data Pool Variable:** Shows the name of the Data Pool variable to have set.
- **Active during screen wide telestration?:** Determines whether the button remains active when the screen-wide telestrator is on, when set to `On`. The default is `Off`, which means the button cannot be triggered when telestration is on.
- **Trigger type:**
  - **On Touch:** Sets the buttons to generate an *onTrigger* event immediately when they are touched. No other events are generated in this mode.
  - **On Release:** Sets the buttons to generate an *onTrigger* event when the last stroke present over a button is released. onStrokeEnter() and onStrokeLeave() events are generated as strokes enter and leave the button (see protocol below).
  - **On Tap:** Sets the buttons to generate an *onTrigger* event when strokes recognized as 'taps' occur after a button is quickly pressed and released without movement or a finger is held down on a key on an object. onStrokeEnter() and onStrokeLeave() events are also generated in this mode.

Copyright © 2026 Vizrt

    - **Taps Only:** Sets the buttons to emit an *onTap* event when stokes recognized as a 'taps' occur. No other events are generated in this mode.
- **Hit Coordinate Type:** Selects the Coordinate system to use. Click either Local, World or Screen.

Events and Notifications

The MtButton generates an **OnTap** event when it is tapped in 'Taps Only' mode:

sub OnTap(**HitContainer as String, Id as StringHit, Point as VertexTaps as Integer, plug-inContainer as String, FirstChildContainer as String**)

end sub

- **HitContainer as String:** The name of the container that was 'hit'.
- **Id as String:** The user ID string entered in the notification interface.
- **HitPoint as Vertex:** The Point where the object was tapped in local coordinates.
- **Taps as Integer:** The number of taps (2= double tap).
- **plug-inContainer as String:** Name of the container MtButton is attached to.
- **FirstChildContainer as String:** Direct (first level) child container of MtButton, that is a (indirect or direct) parent of the tapped container. If the MtButton container itself is tapped, the MtButton container is returned.

MtButton generates an **OnTrigger** event when it is triggered in any other mode than 'Taps Only':

sub OnTrigger(**HitContainer as String, Id as String, HitPoint as Vertex, Taps as Integer, plug-inContainer as String, FirstChildContainer as String, StrokeId as Integer**)

end sub

- **HitContainer as String:** The name of the leaf container that was 'hit'.
- **Id as String:** The user ID string entered in the notification interface.
- **HitPoint as Vertex:** The Point where the object was tapped in local coordinates.
- **Taps as Integer:** The number of taps if tapped (2= double tap).
- **plug-inContainer as String:** Name of the container MtButton is attached to.
- **FirstChildContainer as String:** First child container below MtButton the trigger occurred on or below.
- **StrokeId as Integer:** Unique ID of stroke causing this event.

The MtButton generates an **OnStrokeEnter** event when it is triggered in any other mode than 'Taps Only':

sub OnStrokeEnter(**Id as String, plug-inContainer as String, FirstChildContainer as String, StrokeId as Integer, StrokeCount as Integer, Active as Integer**)

end sub

- **Id as String:** The user ID string entered in the notification interface.
- **plug-inContainer as String:** Name of the container MtButton is attached to.
- **FirstChildContainer as String:** First child container below MtButton the stroke entered.
- **StrokeId as Integer:** Unique ID of stroke causing this event. Only valid from initial enter to final leave.
- **StrokeCount as Integer:** Count of stroke on the button after entry. Is `1` when the first stroke enters.
- **Active as Integer:** This parameter is deprecated and always returns `0`.

The MtButton generates an **OnStrokeLeave** event when it is triggered in any other mode than 'Taps Only':

Copyright © 2026 Vizrt

sub OnLeave(**Id as String, plug-inContainer as String, FirstChildContainer as String, StrokeId as Integer, StrokeCount as Integer, Active as Integer**)

end sub

- **Id as String:** The user ID string entered in the notification interface.
- **plug-inContainer as String:** Name of the container MtButton is attached to.
- **FirstChildContainer as String:** First child container below MtButton the stroke exited.
- **StrokeId as Integer:** Unique ID of stroke causing this event. Only valid from initial enter to final leave.
- **StrokeCount as Integer:** Count of stroke on the button after exit. Is `0` when the last stroke exits.
- **Active as Integer:** This parameter is deprecated and always returns `0`.

The shared memory field names are:

- **HitContainer:** The name of the container that was 'hit'.
- **Id:** The user ID string entered in the notification interface.
- **HitPoint:** The Point where the object was tapped in the selected coordinate type.
- **Taps:** The number of taps (2= double tap).
- **plug-inContainer:** Name of the container MtButton is attached to.
- **FirstChildContainer:** First child container below MtButton that tap occurred on or below.
- **StrokeId:** Unique ID of last trigger or tap.

The Data Pool structure updated is:

- MtButton:
  - string Id
  - string HitPoint
  - string HitContainer
  - string Taps
  - string plug-inContainer
  - string FirstChildContainer
  - string StrokeId

To Create a Simple Scene with the MtButton Plug-in

> ⚠ **Note:** If a touch screen is not available, set **Multi Touch** input to **Mouse** in the **Communication** section of Viz Configuration (see the Viz Engine Administrator Guide).

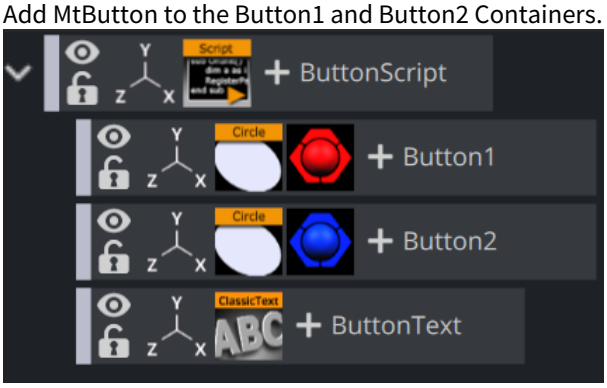
1. Create a new Scene.
2. Add a new group Container.
3. Name the group Container ButtonScript.
4. Add Script to the ButtonScript Container (**Container plug-ins > Global**).
5. Open the Script Editor and type this script:

```
sub OnTap(HitContainer as String, Id as String, HitPoint as Vertex, Taps as
Integer, plug-inContainer as String, FirstChildContainer as String)
    dim bt as Container
    dim text as String
    bt = FindSubContainer("ButtonText")
text = "OnTap(\n HitContainer = " & HitContainer
```

Copyright © 2026 Vizrt

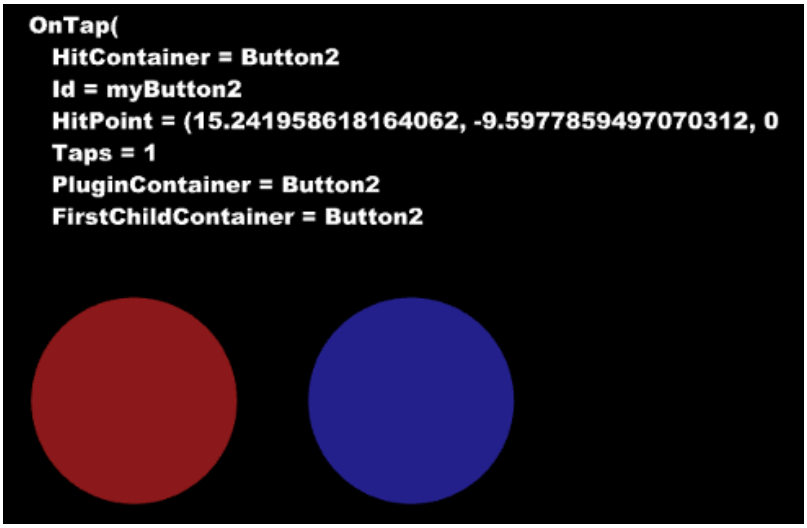
```
text &= "\n Id = " & Id
text &= "\n HitPoint = " & HitPoint
text &= "\n Taps = " & Taps
text &= "\n plug-inContainer = " & plug-inContainer
text &= "\n FirstChildContainer = " & FirstChildContainer & "\n)"
    bt.Geometry.Text = text
end sub
```

6. Click **Compile and Run**.
7. Add three Sub-containers. Name them, in order:
    - Button1
    - Button2
    - ButtonText
8. Add a Circle Geometry to the Button1 and Button2 Containers.
9. Use the Circle Geometry Transformation editors to move the two circles so that both can be seen.
10. Right-click the Button1 and Button2 Containers, in turn, and select **Add Material...** from the context-menu. Open the Material Editor and define a color for both Containers.
11. Add MtButton to the Button1 and Button2 Containers.



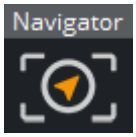
12. In the MtButton editor:
    - Type `myButton1` in the Button1 **Id** field
    - Type `myButton2` in the Button2 **Id** field
    - In **Trigger Type** field click **On Tap** for both containers
13. Add a Font to the ButtonText Container.
14. Open the Scene in **On Air**.
15. Click on each button, in turn, and see the text change with each selection.
16. Use the MtButton editor to modify the button's actions, as required.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

MtNavigator



The MtNavigator plug-in controls Viz World Navigator.

The MtNavigator plug-in allows MultiTouch free navigation over a map generated by the Viz World and Viz Maps plug-ins. Currently it only supports the 'Flat' map model, and not the globe or other 3D models. It sends commands to Navigator to change the position and distance of the map based on MultiTouch input.

Note: This plug-in is located in: Plugins -> Container plug-ins -> MultiTouchComp

Note: The Viz Engine Render Pipeline does not support this plug-in.

MtNavigator Configuration

- **Georeference Container:** Specifies the container which owns Navigator.
- **Momentum:** Enables movement or rotation with momentum after the object is released.
  - **Friction:** Sets the amount of friction used when momentum is enabled.
- **Limit Camera Distance:** Limits the maximum and minimum camera distance.
- **Is free navigation allowed:** Activates or deactivates navigation.
- **Id:** Gives additional context in the handler script, specify a string that identifies any notifications dispatched by MtNavigator. This is often included as an argument for the event so a common script may handle events from a number of plug-ins.
- **Set Shared Memory:** Enables shared memory to be updated for the plug-in notifications when set to `On`.
  - **Shared Memory Prefix:** This sets a 'prefix name' to be prepended to the shared memory variables maintained by the plug-ins notifications. For plug-ins that maintain multiple fields each field name has the prefix prepended to it followed by a dot (**.**) so as to mimic member access to an object (i.e. if the prefix is *Obj* the fields *field1* and *field2* would be identified with the strings *Obj.field1* and *Obj.field2*. The shared memory field *Obj* is also maintained and is simply an integer that is modified every time any of its subfields is updated).
  - **Shared Memory Type:** Selects the shared memory area to update. Click either Global, Scene, or Distributed.
- **Set Data Pool:** Shows 'you wish' plug-in notifications to set a Data Pool variable.
  - **Data Pool Variable:** Shows the name of the Data Pool variable to have set.

Events and Notifications

The events dispatched by MtNavigator are:

OnNavGrab (when navigable object is 'grabbed'):

sub **OnNavGrab** (HitContainer as String, Id as String, Offset as Vertex, Distance as Double, plug-inContainer as String)

end sub

- **HitContainer as String**
- **Id as String**

Copyright © 2026 Vizrt

- **Offset as Vertex**
- **Distance as Double**
- **plug-inContainer as String**
  OnNavGrabUpdate (when navigable object is moved):

sub **OnNavGrabUpdate** (HitContainer as String, Id as String, Offset as Vertex, Distance as Double, plug-inContainer as String)

end sub

- **HitContainer as String**
- **Id as String**
- **Offset as Vertex**
- **Distance as Double**
- **plug-inContainer as String**
- **OnNavGrabRelease** (when navigable object is released):
  - HitContainer as String
  - Id as String

The Shared Memory Field names updated are:

- **HitContainer:** The name of the container that was 'hit'.
- **Id:** The user ID string entered in the notification interface.
- **Offset:** The current lon/lat position in Navigator.
- **Distance:** The current distance in Navigator.
- **plug-inContainer:** The name of the container MtNavigator is attached to.
- **Active:** Set to `1` if 'grabbed', `0` if 'released'.

The Data Pool structure updated is:

- MtNavigator:
  - string Id
  - string HitContainer
  - string Offset
  - string Scale
  - string plug-inContainer
  - string Active

Basic Scene Design

Follow these steps to create a minimal scene which uses MtNavigator:

1. Add GeoImage to a new container named *Nav* and set:
   a. Set **Use Base Map** to `Off`.
   b. Set **Map Size** to `700.0`.
2. Add CWMClient to the container *Nav*.
3. Add Navigator to the container *Nav*.
4. Add MTNavigator to the container *Nav* and set *Nav* as **Georeference Container**.
5. Add NavFinder to the container *Nav* and select **Map-Start** as *Hop Point*.
6. Add a new container *Atlas* as child of *Nav* and add Atlas to it.

Copyright © 2026 Vizrt

MtTelestrator



The MtTelestrator plug-in is used to do telestration on a Container level (used with GraffitiTex, which is added automatically to the same Container).

For scene-wide telestration the MtSensor has a telestration feature.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> MultiTouchComp

> **Note:** The Viz Engine Render Pipeline does not support this plug-in.

This page contains the following topics and procedures:

- MtTelestrator Configuration
- Events and Notifications
- To Create a Simple Scene with Telestration

MtTelestrator Configuration

- **Active:** Enables or disables telestration input from the user.
- **Minimum Telestration Width:** Determines minimum brush width to assign based on pressure applied to sensor.
- **Maximum Telestration Width:** Determines maximum brush width to assign based on pressure applied to sensor.
- **Color:** Assigns the color for subsequent telestration drawing.
- **Id:** Gives additional context in the handler script, specify a string that identifies any notifications dispatched by this plug-in. This is often included as an argument for the event so a common script may handle events from a number of plug-ins.
- **Set Shared Memory:** Enables shared memory to be updated for the plug-in notifications when set to `On`.
    - **Shared Memory Prefix:** This sets a 'prefix name' to be prepended to the shared memory variables maintained by the plug-ins notifications. For plug-ins that maintain multiple fields each field name has the prefix prepended to it followed by a dot '.' so as to mimic member access to an object: If the prefix is 'Obj' the fields 'field1' and 'field2' would be identified with the strings 'Obj.field1' and 'Obj.field2'. The shared memory field 'Obj' is also maintained and is simply an integer that is modified every time any of its 'subfields' is updated.
    - **Shared Memory Type:** Selects the shared memory area to update. Click either Global, Scene, or Distributed.
- **Set Data Pool:** Shows 'you wish' plug-in notifications to set a Data Pool variable.
    - **Data Pool Variable:** Shows the name of the Data Pool variable to have set.
- **Init:** Clears current telestration and make Telestration active.
- **Clear:** Clears the contents of the attached telestration.

Events and Notifications

The events dispatched by MtTelestrator are:

- `OnTelestrationStateChange` (sent when telestration is activated or deactivated):

Copyright © 2026 Vizrt

    - `plug-inContainer` as String
    - `Id` as String
    - `TelestrationOn` as Boolean
- `OnTelestrationColorChange` (sent when telestration is activated or deactivated):
    - `plug-inContainer` as String
    - `Id` as String
    - `PenColor` as Color
- `OnTelestrationClear` (sent when telestration is cleared):
    - `plug-inContainer` as String
    - `Id` as String
    - `TelestrationOn` as Boolean

      The shared memory field names are:

- **plug-inContainer:** The name of the plug-ins container or '' if attached to a scene
- **Id:** The user ID string entered in the notification interface
- **PenColor:** The current pen color
- **Active:** Is set to `1` if telestration is active, `0` if not.

The Data Pool structure updated is:

- MtTelestrator:
    - string plug-inContainer
    - string Id
    - string PenColor
    - string Active

To Create a Simple Scene with Telestration

> **Note:** If a touch screen is not available, set **Multi Touch** input to **Mouse** in the **Communication** section of Viz Configuration (see the Viz Engine Administrator Guide).

1. Create a new Scene.
2. Add a new group Container.
3. Add a Rectangle Geometry to the Container.
4. Add the MtTelestrator plug-in to the container. This automatically adds GraffitiTex to the container.
5. Set the **Active** parameter for the MtTelestrator and GraffitiTex to `On` in the Properties Panel.
6. Open the Scene in **On Air**. Use the touch input device to create freehand drawing.
7. Use the MtTelestrator and GraffitiTex plug-in editors to modify the telestration affect as required.

See Also

- GraffitiTex

Copyright © 2026 Vizrt

Plug-in Event and Notification System

All the MultiTouch (Mt) plug-ins share a common event notification module and associated configuration interface. For all plug-ins that generate events, events are dispatched up the Container hierarchy from the plug-in Container to the first script found with a handler present for the event being generated.

For plug-ins that optional update values in Shared Memory, select which one of the three maps to update:

- System
- Scene
- Distributed

Each shared memory area updated by a plug-in is composed of one root variable specified in the 'Shared Memory Prefix' field in the plug-ins user interface. This 'root' variable is updated if any of its additional sub-fields are updated. The root variable contains an integer or update ID. Sub-fields are identified by adding 'dot delimited' suffix on the root name. For example, if 'Shared Memory Prefix' is set to 'Name', the sub-field 'field' would be identified as `Name.field` in a Viz script.

For all plug-ins that can set a Data Pool variable, each has a 'structure' defined in a *.dp* file. There is one such file for each plug-in. Both the file and the name of the structure defined by it are the same as the name of the plug-in. For example, in the case of an MtButton, the file is *MtButton.dp* and the name of the structure for sub-item access is *MtButton*.

Copyright © 2026 Vizrt

2.4.9 Presenter Container Plug-ins

The following Container plug-ins are located in the Presenter folder.

- Bar
- Bar Value
- Bar Values
- Pie Slice
- Pie Values



Presenter Plugins have been replaced by Visual Data Tools

Copyright © 2026 Vizrt

Bar



The Bar plug-in tells the Bar Values plug-in that the container is part of the bar chart and it also tells which items of the geometry on the container to control.

Each container that is to be a bar, must hold a geometry and a Bar plug-in. Bar can be used as a standalone plug-in.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Presenter

Bar Properties

- **Standalone:** If the Bar plug-in is going to be used without a Pie Values plug-in to set the values, enable this button. It enables you to set the value directly on the plug-in and you can define normalization factors between the input values and the value that is being set on the geometry.
- **Bar Min / Max:** If in standalone, these values set the minimum and maximum values for the bar. These values are connected to the Obj Min/Max values and by adjusting these together you can get the required effect between input values and visual effect.
- **Value:** Sets the scaling value of the object.
- **Obj Min / Max:** Maps the input values to the native values on the object being controlled by the plug-in. Adjust these together with Bar Max/Min on the Bar Values plug-in or on the Bar plug-in itself if in standalone mode, to get the required effect between input values and visual effect.
- **Param Type:** Sets the type of value to control on the object.
    - **Param:** Controls a named parameter on the object. Use the console to find the correct name for the parameter: Enable **Show commands**, Open the object's editor, change the parameter and see in the console what the parameter name is.
    - **Position:** Position values can be set to control the X, Y or Z axis.
    - **Rotation:** Rotation values can be set to control the X, Y or Z axis.
    - **Scale:** Scale values can be set to control All, X, Y or Z axis.
    - **Text:** Text values can be shown as Integer, Float or Formatted text.
- **Initialize:** Enables the plug-in to initialize and gain control of the parameters.

See Also

- Bar Value
- Bar Values
- Basic Geometry Visual Data Tools

Copyright © 2026 Vizrt

Bar Value



The Bar Value plug-in controls the value of all Bar plug-ins and Counters of its child nodes.

In addition, a set of commands can be specified that are executed when the value passes certain thresholds.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Presenter

Bar Value Properties

- **Bar Min:** Minimum Value of the bar.
- **Bar Max:** Maximum Value of the bar.
- **Value:** The actual value of the bar.
- **control counters:** States if counters should be affected. If this is checked, Value is shown in all Sub-Containers with Counter plug-ins.
- **Type:** Determines whether the command should fire if the threshold is passed from below, or from above. Off does not execute the command at all. Available options are *Off*, *Up* and *Down*.
- **Threshold:** Compares the value against the threshold.
- **Command:** Executes a Viz Artist/Engine command when the threshold is passed.

To Create a Bar that Shows Its Value

1. Create a Container and add the Bar Value plug-in to it.
2. Open the Bar Value editor, and check control counters.
3. Set Threshold to `20`, Type to Up and set a command (for example, `MAIN*ONAIR GET_INFO`).
4. Add two Sub-Containers.
5. Add a Font and a Counter plug-in to the first Sub-Container.
6. Add a Cube and a Bar plug-in to the second Sub-Container.
7. Set the Cube's Y-Axis Center to `(B)ottom`.
8. Open the Bar editor and set Param Type to Scale and Scale to `Y`.
9. Modify Bar Value's Value field. If the value exceeds `20` the information is written to the console.

See Also

- Bar
- Bar Values
- Basic Geometry Visual Data Tools

Copyright © 2026 Vizrt

Bar Values



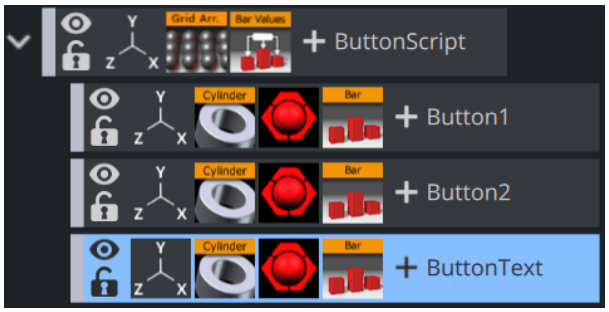
The Bar Values plug-in communicates with all Sub-Containers that hold a Bar plug-in and lets you set values for each of them.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Presenter

Bar Values Properties

- **Num Bars:** Sets the number of bars to be controlled by the Bar Values plug-in. Maximum number of bars is 25.
- **Auto Max:** Allows the user to switch between Manual and Auto mode. Auto mode disables the Bar Max value.
- **Bar Max:** Sets the maximum values of the bar.
- **Bar Min:** Sets the minimum value of the bar.
- **Scale:** Sets the scale of the object.
- **Bar Animation:** Sets the number of bars to be animated.
- **Bar Value 1-25:** Sets the value for the bar(s).

To Create a Bar Chart with the Bar Values Plug-in



1. Add a group container to the scene tree, and add the Bar Values and Grid Arrange to it.
2. Add a group container as a Sub-Container to the first, and name it `Bar`.
3. Add material, the Cylinder and Bar to the Bar container.
4. Open the Cylinder editor and set Center to `Bottom`.
5. Open the Bar editor, set Param Type to **Scale** and Scale to `Y`, and click **Initialize**.
6. Create and place a number of copies of the Bar container (for example, `5`) at the same level as the Bar container.
7. Open the Grid Arrange editor and set **Number of Columns** to **6** and **Column offset** to `30.0`.
8. Open the Bar Values editor and set **Num Bars** to `6` and **Bar Animation** to `6`.

See Also

- Basic Geometry Visual Data Tools

Copyright © 2026 Vizrt

- Control Bars
- Bar
- Bar Value
- Pie Slice
- Pie Values

Copyright © 2026 Vizrt

Pie Slice



The Pie Slice plug-in is used with the Pie Values plug-in to create and animate a pie chart of up to 20 slices.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Presenter

Pie Slice Properties

- **Obj Min / Max:** Maps the input values to the values on the cylinder object. Adjust these together with Bar Max/Min on the Pie Values plug-in to get the required effect between input values and visual effect.
- **Offset:** Offsets the slice from the center of the pie chart.
- **Text Offset:** Offsets the text label from the center of the pie chart.
- **Control Text Values:** Sets the type of text. Available options are No, Integer, Float and Formatted.
- **Initialize:** Initializes the value of the pie slice.
- **Offset Scale with Size** and **Offset Scale: (Engine 5.6 or newer)**

The **Pie Slice** plug-in separates a slice from the pie by moving its container outward along the slice's own centerline (bisector). Because a slice is a wedge, moving it outward by a given distance opens a gap between it and its neighbors that is proportional to the *sine of half the slice angle*. The consequence: with the same **Offset** value on every slice, wide slices visibly separated much further than narrow ones, and uniformly separated "exploded pie" animations were not achievable.

- **Offset Scale With Size** (checkbox, default: **on**)
  - **On** — legacy behavior. **Offset** is the distance the slice container travels outward. The resulting gap between neighboring slices grows with slice size.
  - **Off** — uniform-gap behavior. **Offset** now defines the *gap* between the slice and its neighbors, and the plug-in automatically computes the travel distance needed for each slice to produce that same gap. All slices appear separated by the same distance regardless of their size.
- **Offset Scale** (numeric, default: **4.0**, range **1.0–100.0**)
  - Limits how far a slice may travel when **Offset Scale With Size** is off. The travel distance is capped at **Offset × Offset Scale**.
  - This cap is needed because a very thin slice would require an extremely large travel distance to open the same gap as a wide one, which would fling it off the pie. Raise the value to let thin slices reach their full uniform gap; lower it to keep all slices close to the pie center.

Existing scenes are unaffected. The default (**Offset Scale With Size** on) reproduces the pre-5.6 behavior exactly, so scenes built with earlier versions render identically.

Practical guidance

Uncheck **Offset Scale With Size** on every Pie Slice container in the chart, then animate **Offset** to get an evenly separated "exploding pie". Setting it on only some slices produces mixed behavior.

With the default **Offset Scale** of 4.0, slices of roughly **29° or wider** (about 8% of the pie) reach the full uniform gap; thinner slices hit the cap and separate slightly less than the others. Approximate thresholds for other values:

Copyright © 2026 Vizrt

Offset Scale	Slices reaching the full uniform gap
1.0	180° and wider (cap is effectively always active)
2.0	60° and wider (~17% of the pie)
4.0 (default)	~29° and wider (~8% of the pie)
10.0	~11.5° and wider (~3% of the pie)
100.0	~1° and wider

See Also

- Basic Geometry Visual Data Tools
- Control Bars
- Bar
- Bar Value
- Bar Values
- Pie Values

Copyright © 2026 Vizrt

Pie Values



The Pie Values plug-in is used with the Pie Slice plug-in to create and animate a pie chart of up to 20 slices.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Presenter

Pie Values Properties

- **Pie Min and Max:** Sets the minimum and maximum values for the pie size and the range of the size.
- **Min and Max Value:** Sets the range of the shown pie.
- **Scale:** Sets the scale of the pie. Default is `1`.
- **Slice:** Sets the number of slices to be shown as part of the pie.
- **Value 0 - 19:** Sets the slice value. Available number of slices is `20`.

To Create a Pie Chart with the Pie Values Plug-in



1. Add a group container to the scene tree, and name it `Pie`.
2. Open the transformation editor and set **Rotation X** to `45.0`, and **Scaling Y** (single) to `0.2`.
3. Add the Pie Values plug-in to the Pie container.
4. Open the Pie Values editor and set *Slice* to `6.0`, and enter the following values in **Value 0** to `5`; `10.0, 20.0, 20.0, 10.0, 20.0, 20.0`.
5. Add a Sub-Container to the Pie container and name it *Slice*.

Copyright © 2026 Vizrt

6. Add the Cylinder geometry plug-in, material and Pie Slice plug-in to the Slice container.
7. Open the Pie Slice editor and set Offset to `10.0` and Text Offset to `80.0`.
8. Add a Sub-Container to the *Slice* container and name it `Text`.
9. Open the transformation editor for the Text container and set **Scaling** (locked) to `0.2` and **Rotation X** to `-90.0`.
10. Add a font, material and the Extrude plug-in to the Text container.
11. Open the Extrude editor and set **Extrusion Depth** to `100.0`.
12. Create and place a number of copies of the Slice container (for example, `5`) at the same level as the *Slice* container.
13. Change the colors of the pie slices to distinguish each slice.

See Also

- Basic Geometry Visual Data Tools
- Control Pie
- Bar
- Bar Value
- Bar Values
- Pie Slice

Copyright © 2026 Vizrt

2.4.10 RealFX Container Plug-ins

The RealFX plug-in set enables you to create particle effects in Viz Artist.

Particle systems are a computer graphics technique to simulate certain physics-based effects, which are otherwise very hard to reproduce with conventional rendering techniques. Examples of such effects which are commonly replicated using particle systems include fire, explosions, smoke, weather effects, sparks, falling leaves, dust, meteor tails, or abstract visual effects like glowing trails, magic spells, etc.

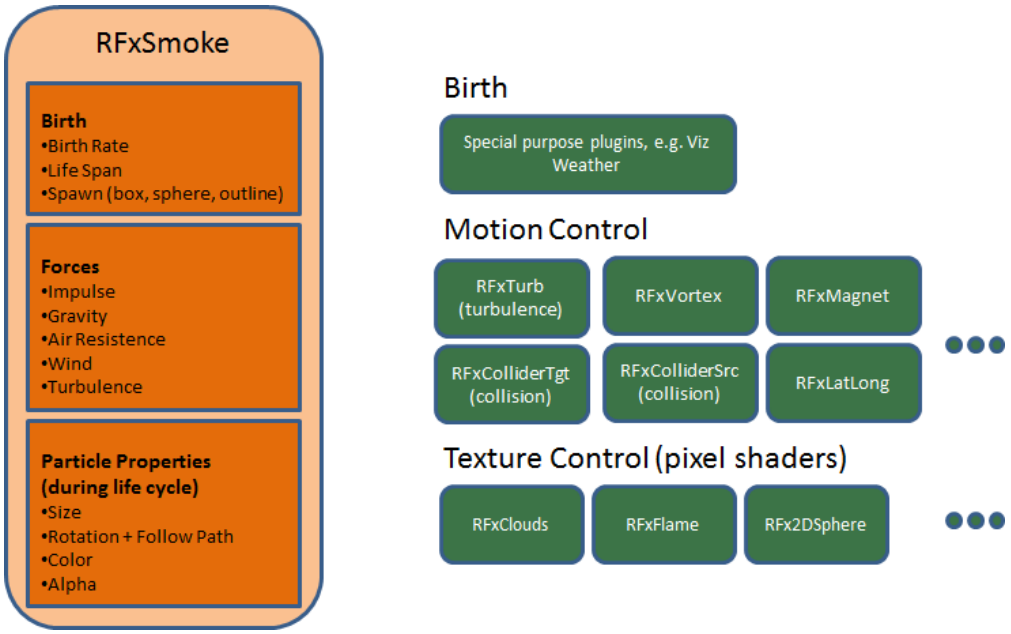
The particle effects in Viz Artist/Viz Engine run in real-time, meaning that there are a few inherent constraints that must be taken into account when considering best practices for employing this plug-in set. For example, there is a trade-off between the number of particles and performance optimization; more generally there needs to be a considered balance between performance and visual quality.

RFxSmoke is the baseline plug-in within the RealFX plug-in set. The remaining plug-ins in this set are applied on top of RFxSmoke in any given Container. RFxSmoke includes built-in functionality and the ability to host the additional functionality contained in the other plug-ins in this set. Part of the built-in functionality (for example, Turbulence) is kept for compatibility with previous versions of Viz Artist.

There are three categories of additional plug-ins:

- **Birth plug-ins:** Refers to where the particles are spawned.
- **Motion control plug-ins:** Governs the position, direction, velocity, size and color of each particle.
- **Texture control plug-ins:** Affects the texture mapping and the “look” of each particle by using pixel shader technology.

RealFX Architecture



The following Container plug-ins are located in the RealFX folder:

- RFxColliderSrc
- RFxColliderTgt
- RFxLatLong
- RFxMagnet
- RFxTurb
- RFxVortex

See Also

- RealFX in Basic Shader Plug-ins
- Basic Geometry RealFX in Basic Geometry Plug-ins

Copyright © 2026 Vizrt

RFxColliderSrc



The RFxColliderSrc plug-in is used in conjunction with RFxColliderTgt to create a collision and collision detection system.

For example, the source could be a notional wall or floor, whereas the target would be the particles themselves. The particle target can work with a number of sources.

Note: When RFxColliderSrc is added to a container, the RFxCollisionManager plug-in is automatically added at the scene level.

Note: This plug-in is located in: Plugins -> Container plug-ins -> RealFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

RFxColliderSrc Properties

Adjust the following parameters as required to achieve the required effect:

- Object Type (Auto, Sphere, Box)
- Bounce
- Random
- Flag 1-8 (on/off)

See Also

- RFxSmoke
- RFxColliderTgt

Copyright © 2026 Vizrt

RFxColliderTgt



The RFxColliderTgt plug-in is used in conjunction with RFxTurb to create a collision and collision detection system.

For example, the source could be a notional wall or floor, whereas the target would be the particles themselves. The particle target can work with a number of sources.

Note: When RFxColliderTgt is added to a container, the RFxCollisionManager plug-in is automatically added at the scene level.

Note: This plug-in is located in: Plugins -> Container plug-ins -> RealFX

RFxColliderTgt Properties

Adjust the following parameters as required to achieve the required effect:

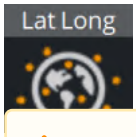
- Flag 1-8 (on/off)
- Proximity Fade and Fade Range: Allows you to fade particles based on their proximity to other objects (only spheres and boxes).

See Also

- RFxSmoke
- RFxTurb

Copyright © 2026 Vizrt

RFxLatLong



The RFxLatLong plug-in fits any other RealFX motion onto a globe surface.

The effects are mainly geographic and large-scale.

Note: This plug-in is located in: Plugins -> Container plug-ins -> RealFX

RFxLatLong Properties

Adjust the following parameters as required to achieve the required effect:

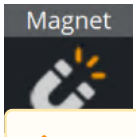
- Radius
- Spawn Min Long
- Spawn Max Long
- Spawn Min Lat
- Spawn Max Lat
- Update from Globe (on/off)
- Spawn Min Alt
- Spawn Max Alt
- Orientation (Horizontal/Radial)
- Trim Radius

See Also

- pxColorWorks

Copyright © 2026 Vizrt

RFxMagnet



The RFxMagnet plug-in creates motion patterns that resemble a magnetic field.

Note: This plug-in is located in: Plugins -> Container plug-ins -> RealFX

RFxMagnet Properties

Adjust the following parameters as required to achieve the required effect:

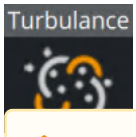
- Amount
- Center X
- Center Y
- Center Z
- Power Law
- Min Distance Clamp

See Also

- RFxSmoke

Copyright © 2026 Vizrt

RFxTurb



The RFxTurb plug-in applies turbulence-like forces to particles.

It can be used to create the effect of random changes in wind force and direction.

Note: This plug-in is located in: Plugins -> Container plug-ins -> RealFX

Note: RFxTurb requires to have RFxSmoke on the same container.

RFxTurb Properties

Adjust the following parameters as required to achieve the required effect:

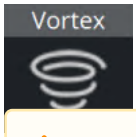
- Amount
- Inertia
- Wavelength
- Update Rate

See Also

- RFxSmoke
- RFxVortex

Copyright © 2026 Vizrt

RFxVortex



The RFxVortex plug-in applies vortex-like forces to particles.

It can be used to create the effect of a tornado-like twister.

Note: This plug-in is located in: Plugins -> Container plug-ins -> RealFX

RFxVortex Properties

Adjust the following parameters as required to achieve the required effect:

- Amount
- Inertia
- Center X
- Center Y
- Center Z
- Rot X
- Rot Y

See Also

- RFxSmoke

Copyright © 2026 Vizrt

2.4.11 Script Container Plug-ins

The Script plug-ins is a folder to save created Container Script plug-ins.

A script can be saved as its own script plug-in, and used in future Scenes (see the Scripting section of the Viz Artist User Guide). To save a Script plug-in, drag a compiled script into the Script plug-ins folder.

> **Note:** Script plug-ins are saved to *<viz data folder>\Scriptplug-ins*.

See Also

- Script plug-in

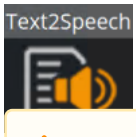
Copyright © 2026 Vizrt

2.4.12 Sound Container Plug-ins

The following Container plug-ins are located in the Sounds folder:

- Text2Speech

Text2Speech



The Text2Speech plug-in is used to create speech from text.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container Plug-ins -> Sounds

Text2Speech Properties

- **Output:**
  - **Video:** Plays the output to the video out.
  - **Speakers:** Plays the output to the computer speakers.
  - **Both:** Plays the output to both the video out and computer speakers. If running without a video card, this causes double playback.
- **Configuration:** Selects predefined configurations for the **Speaker**, **Rate**, and **Pitch** parameters. This section is only visible when a profile file is located.
  - **Custom:** Sets the plug-in parameters manually. This is disabled if Configuration is set to `Profile.`
  - **Profile:** Sets the plug-in parameters based on the profile file selected in the **Profiles** pane. This pane is only visible when a profile file is located. If Configuration is set to `Custom`, the **Profile** button is disabled.
- **Speakers:** Shows all speakers present in the system.
  - Lists all of the speakers currently present in the system, and allows the user to select one. Disabled if Custom configuration is selected.
- **Volume:** Sets the desired output volume.
- **Rate:** Sets the desired output rate.
- **Pitch:** Selects the output pitch. The available settings are:
  - X-Low
  - Low
  - Medium
  - High
  - X-High
- **Text:** Changes made to a text is played out immediately when in On Air mode. Play-out of changes to text during scene design in Viz Artist requires pressing the **Play** button.
- **Let Me Finish:** Defines the behavior when a new text triggers play-out. Used in On air mode only.
  - **Off:** New text stops the current output and start playing the new text. When set to `Off`, **Queue Intermission** and **Minimal Queue Intermission** is disabled.
  - **On:** When text is playing out, adding new text adds this to a queue and play out automatically when the previous text is done, adjusted for the intermission value. When set to `On`, **Queue Intermission** is enabled.
  - **Speed Up (1):** When there is a play-out queue, the output **Rate** is increased by one, and the **Intermission Minimal Value** is used.

Copyright © 2026 Vizrt

    - **Speed Up (2):** When there is a play-out queue, the output **Rate** is increased by one. Depending on the queue size, the output rate can be increased by two. In both cases, the **Intermission Minimal Value** is used.
    - **Shorten:** As when set to On, adding new text adds this to a queue to be played out automatically when the previous text is done, adjusted for the intermission value. However, the text is shortened.
- **Queue Intermission (ms):** Defines the pause in milliseconds between text play-out when there is a text queue. The default value is `1000` (one second). This parameter is disabled when **Let Me Finish** is set to `Off.`
- **Minimal Queue Intermission (ms):** Defines the minimum pause in milliseconds between text play-out when there is a text queue and **Let Me Finish** is set to `Speed Up` or `Shorten`. The default value is `1000` (one second).
- **Play:** Starts play-out of the current text.
- **Pause:** Pauses the on-going audio play-out on the video out output.
- **Stop:** Stops play-out.
- **Read Languages:** Reads the Languages files from *%Programdata%\vizrt\VizEngine\TextToSpeech*.
    - Language files are used to improve the pronunciation of words. The Text2Speech plug-in utilizes the *Universal Phone Set (UPS)* machine-readable phonetic alphabet created by Microsoft for this purpose. The language files are also used to replace defined abbreviations, for example replacing *Av* with *Avenue.*
- **Read Profiles:** Reads profiles file from *%Programdata%\Vizrt\VizEngine*.

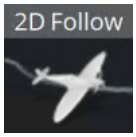
Copyright © 2026 Vizrt

2.4.13 SplineFX Container Plug-ins

The following Container plug-ins are located in the SplineFX folder:

- 2D Follow

2D Follow



The 2D Follow plug-in works together with the 2D Ribbon.

2D Follow to automatically creates an animation path that follows the form of a 2D Ribbon. The 2D Ribbon must be created first.

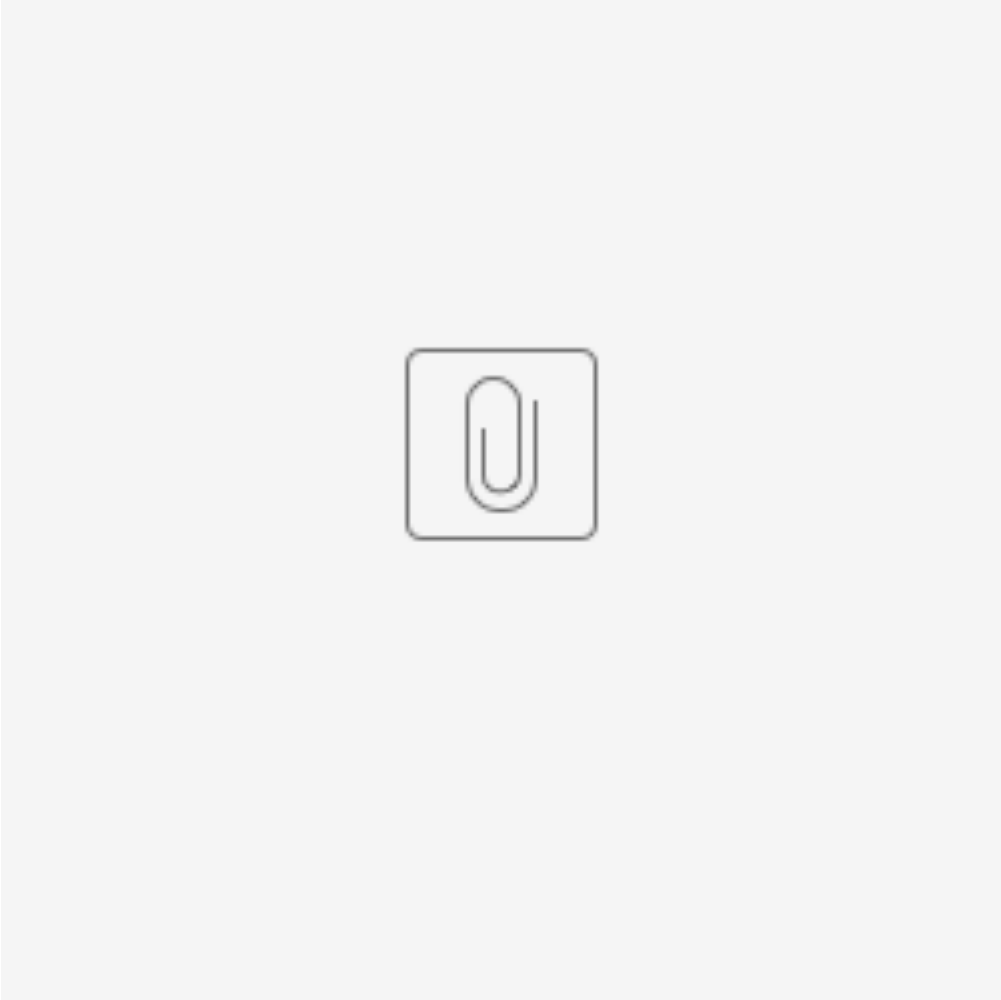
Note: This plug-in is located in: Plugins -> Container plug-ins -> SplineFX

2D Follow Properties

- **Path Position:** Sets the position for the object along the path of the 2D Ribbon object. Animate this value to get an animation in the stage.
- **Position Wrap:** Can be set to **Interpolate**, **Clamp** or **Repeat**.
- **Translation Offset:** Sets an offset between the object and the 2D Ribbon path.
- **Keep Orientation:** Keeps the orientation between the object and the 2D Ribbon path as the object moves along the path when enabled.
- **Orientation Offset:** Offsets the orientation between the object and the 2D Ribbon path.

Copyright © 2026 Vizrt

To Set up the 2D Follow Plug-in



1. Add the 2D Ribbon geometry to the scene tree.
2. Open the 2D Ribbon editor and enable **Show Control Point Values**.
3. Set alternating values of `30.0` and `-30.0` to the Y axis values, creating a wave shape.
4. Add the item(s) (for example, a Sphere) you want to animate along the 2D Ribbon's path as a Sub-Container to the 2D Ribbon container.
5. Add the 2D Follow plug-in to the container.
6. Open the 2D Follow editor and animate its Path Position value from `0.0` to `100.0`.

See Also

- 2D Ribbon
- **Create Animations** section of the Viz Artist User Guide.

Copyright © 2026 Vizrt

2.4.14 TextFX Container Plug-ins

The following Container plug-ins are located in the TextFX folder:

- Common Text FX Properties
- Convert Case
- Mark Text
- Text FX Alpha
- Text FX Arrange
- Text FX Color
- Text FX Emote
- Text FX Explode
- Text FX Jitter Alpha
- Text FX Jitter Position
- Text FX Jitter Scale
- Text FX Plus Plus
- Text FX Rotate
- Text FX Scale
- Text FX Size
- Text FX Slide
- Text FX Vertex Color
- Text FX Vertex Explode
- Text FX Write

Copyright © 2026 Vizrt

Common Text FX Properties

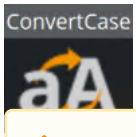
When in text editing mode, any TextFX effects are automatically deactivated to allow text input. As soon as the text editing is deactivated, either by selecting another container, or by just selecting a different container property within the selected container, such as transformation, the TextFX effect is applied again in the scene preview.

The following properties are common to most of the Text FX plug-ins:

- **Progress %:** 0 percent progress represents the beginning of the effect, 100 percent the end. Animate this value from `0` to `100` to see the effect or from `100` to `0` to animate the effect backwards.
- **Progress Type**
  - **Absolute:** 100 percent progress animates all characters of a text, regardless how many characters it has.
  - **Relative:** 100 percent progress animates ten characters. This is needed to adjust the timing of several text objects with different sizes. The effect speed should be for example five characters per second, so the animation must be from 0 to 100 in two seconds. This works for text with ten characters or less. If you want to use longer texts, animate the progress value over 100 percent (ten percent for each character).
- **Direction:** Sets the direction of the text effect sequence, you can choose between the following options:
  - **Left:** Starts with the first character.
  - **Right:** Starts with the last character
  - **Random:** Uses a random order.
  - **Static:** All characters are processed at the same time.
  - **Wave:** Starts with the first character, animates the effect from 0 to 100 and then down again to 0.
  - **Center:** Starts the effect from the center of the text.
  - **2 Center:** Starts the effect at the same time from the beginning and the end of the text. They meet at the center.
- **Interpolate:** Choose between a **soft** or a **linear** interpolation of the transition from character to character.
- **Effect Range:** Defines how many characters are processed at the same time. If for example the Effect Range is set to `4`, and you manually increase the progress value, you see that when the fifth character starts to be processed, the first is finished, when the sixth starts, the second is finished, and so on.
- **Random Seed:** Specifies a seed for the random number generator when a random direction is chosen. Even though Viz Artist uses random numbers, the animation for a specific random seed always looks the same. This is typically useful if you combine two different text effects.
- **Ignore Space:** Ignores space when animating the effect.
- **Ignore Newline:** Ignores new lines when animating the effect.
- **Text Order:** Sets the text order for the effect. Available options are As Is, Horizontal or Vertical. Horizontal and Vertical enables the Text Direction options.

Copyright © 2026 Vizrt

Convert Case



The Convert Case plug-in converts ASCII strings to upper or lower case.

Current container must hold a string.

Note: This plug-in is located in: Plugins -> Container plug-ins -> TextFx

This page contains the following topics and procedures:

- Convert Case Properties
- To Convert a String to Upper or Lower Case

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Convert Case Properties

- **Upper/Lower:** Converts the text for the specific container to upper or lower case.
  `FUNCTION*ConvertCase*text_case - int; 0: upper, 1: lower`
  `FUNCTION*ConvertCase*text - string`

To Convert a String to Upper or Lower Case

1. Add a group to the scene tree, and add a font to it.
2. Add the Convert Case plug-in to the same container.
3. Open the Convert Case editor, and enter text in the editor.
4. Click the Upper button to convert to upper case text, or click the Lower button to convert to lower case text.

Copyright © 2026 Vizrt

Mark Text



The Mark Text plug-in allows you to highlight or underline words in a text string.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	❌

Mark Text Properties

- **Text to Mark:** Sets the text that should be highlighted. The text is case sensitive.
- **Marks spaces:** Also allows to put the marker between words.
  - Please verify your scene when using the Mark Space option on single words located at the end of the textbox. In some rare conditions, it may happen that the selection does not match the expected result.
- **Multiple marks:** Highlights all matched text.
- **Use regular expression:** Allows regex to be used for more sophisticated highlighting.
  - **Only matched group:** Highlights only on capture-groups `(value)`, other matched texts outside capture-groups or inside non-capture-groups `(?:value)` are not.
  - **Case insensitive:** Matches both upper and lower case.
  - **Grammar:** Selects one of the following supported grammars:
    - ECMA
    - Basic
    - Extended
    - Awk
    - Grep
    - Egrep
- **Mark:** Selects which text should be highlighted. Available options are foreground (Front), background (Shadow), Both and None. Background highlights the text's shadow.
- **Size:** Sets the thickness of the highlight.
- **Position:** Sets the y-position of the highlight in relation to the text.
- **Depth:** Sets the z-position of the highlight in relation to the text. This allows the highlight to move behind the text.
- **Foreground Color:** Sets the color and alpha value of the highlight.

To Mark Text

1. Add a group container to the scene tree and name it `Text`.
2. Add a font to the Text container.

Copyright © 2026 Vizrt

3. Open the font editor and enter a text string.
4. Add the Mark Text plug-in to the Text container.
5. Open the Mark Text Editor and enter a word from your text string or regex if enabled.



Note: Do not change the font type between texture and geometry as this does not affect the whole string when using the Mark Text plug-in.

Note: Overlapping marks could occur with narrow character spacing or kerning. This would be visible in case of transparent marks.

Copyright © 2026 Vizrt

Text FX Alpha



The Text FX Alpha plug-in is a text effect that creates a fade in effect for the text characters.

The effect sequence can be set to go many different ways. A fade in and out effect can also be achieved.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Alpha Properties

- **Alpha Begin %:** Sets the alpha level of the character at 0% effect.
- **Alpha End %:** Sets the alpha level of the character at 100% effect.

See Also

- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Arrange



The Text FX Arrange plug-in allows you to arrange characters in either a circular or a wave shape.

The characters can be animated on the selected shape by animating the offset value.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

This page contains the following topics and procedures:

- Text FX Arrange Properties
  - Circle Parameters
  - Wave Parameters

Text FX Arrange Properties

- **Offset %:** Moves the characters on the shape. 100% means one full rotation of the text on the circle.
- **Scale %:** Sets the text on the shape. The parameter does not scale the characters itself, but the kerning of the characters.
- **Shape:** Changes the shape of the character layout to **Circle** or **Wave**.

⚠ **Note:** The shape of the text transformation is not visible in the Scene Editor while the Text Editor is active.

Circle Parameters

- **Diameter:** Sets the diameter of the circular shape.
- **Positioning:** Defines the position of the text on the circular shape:
  - **Relative:** Maintains the text spacing of the text object.
  - **Absolute:** Distributes the text evenly onto the circle.
- **Direction:** Sets the direction of the text on the circle to either **Clockwise** or **Counterclockwise**.
- **Rotation:** Rotates the characters on the X-axis.
- **Align:** Rotates the characters on the Z-axis to align the X-axis with the tangent of the shape at the characters position.

Wave Parameters

- **Length:** Sets the wave length. A high value creates many and small waves, a low value creates fewer and bigger waves.
- **Width:** Sets the width of the wave.

Copyright © 2026 Vizrt

- **Height:** Sets the height of the waves without altering the number of waves as the **Length** does.
- **Shift %:** The wave usually starts at the height `0`. The shift value moves that point, thereby moving the texts position on the wave structure. Setting **Shift %** to 100% brings you back to the beginning.
- **Damping:** Modifies the amplitude of the wave curves.
- **Rotation:** Rotates the characters on the X-axis.
- **Align:** Rotates the characters on the Z-axis to align the X-axis with the tangent of the shape at the characters position.

Copyright © 2026 Vizrt

Text FX Color



The Text FX Color plug-in adds a color effect to the text.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

Note: This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Color Properties

- **Color Begin:** Sets the initial color before applying the effect.
- **Color End:** Sets the color the effect should apply.

See Also

- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Emote



The Text FX Emoticons plug-in allows you to create an emoticon container, add it to the Emoticons plug-in and replace characters (tokens) in a text string with an assigned container.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Note: This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Emoticons Properties

- **Autoscale:** Scales the emoticon containers so their height above the baseline (0) is identical to the height of character A.
- **UTF-8 Encoding:** Enables UTF-8 encoding for the text effect.
- **Token 1 - n:** Sets the placeholder for an emoticon.
- **Container 1 - n:** References the container holding the emoticon. The container should be similar to the layout of a glyph.
- **Precache 1-n:**- Enables Viz Artist/Engine to cache the objects.

Copyright © 2026 Vizrt

Text FX Explode



The Text FX Explode plug-in creates an explosion like function where the characters get thrown away from their initial position.

The speed, direction and spread of the moving characters can be altered with parameters.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

⚠ **Note for the Classic Render Pipeline:** Works only if the text is set to texture (does not work with geometry text).

⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Explode Properties

- **Duration:** Defines the duration of the progressing for each of the characters. If you want a longer progressing, increase this value. You could achieve the same by making the gravity stronger and animating the progress slower, but it is more easy to increase the duration instead if your explode effect is too short.
- **Opening Angle:** Sets the angle for the spread of the characters. `0` sends them straight up, `360` spreads them in a circular shape.
- **Angle Rotate X:** Rotates the opening angle around the X-axis.
- **Angle Rotate Y:** Rotates the opening angle around the Y-axis.
- **Force:** Sets the force that throws away the characters. A high force makes them go far away, conversely a low force creates only a small motion of the characters.
- **Force Spread %:** Sets a variation of the force among the characters.
- **Use Axis:** Allows you to select on which axis or combination of axes the characters are to spread along.
- **Gravity:** Sets a gravity force that influence the path of the characters to end up going downwards. The higher the value is set, the faster each character diverts from its initial path and starts going downwards.
- **Use Rotation:** Rotates characters as they are being thrown away from their initial position when enabled.
- **Rotation Force:** Sets the degree of rotation as the characters are being thrown away.
- **Show Force:** Shows lines showing the characters path and speed in the Scene Editor when enabled.

⚠ **Note:** Show Force is not available in the Viz Engine Render Pipeline.

See Also

- Text FX Vertex Explode
- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Jitter Alpha



The Text FX Jitter Alpha plug-in creates a jittering motion of the characters by randomly changing the alpha value of each character. The degree of change and the start sequence of the jittering can be altered. To use the plug-in, add it onto a container with a font. To create an animation, animate the progress. Other values can of course be animated as well.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Jitter Alpha Properties

- **Effect What:** Defines where the effect should have an effect. Available options are background (BG), foreground (FG) or both.
- **Alpha Begin:** Sets the alpha level of the character at 0% effect.
- **Alpha End:** Sets the alpha level of the character at 100% effect.
- **Randomness:** sets the intensity of the jittering alpha changes.

Copyright © 2026 Vizrt

Text FX Jitter Position



The Text FX Jitter Position plug-in creates an jittering motion of the characters by randomly changing the position of each character. The degree of position change and the starting sequence of the jittering can be altered. To use the function add it onto a container which holds a font. To create an animation, animate the progress. Other values can of course be animated as well.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

Note: This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Jitter Position Properties

- **Use Axis:** Defines on which axis or axes the characters moves to create the jittering effect.
- **Lock Axis:** Sets **Range** and **Randomness** to the same value for all axes when enabled. If you disable it, **Range** and **Randomness** for each of the axes are shown and you can set them individually.
- **Range:** Sets the range of the jittering movement.
- **Randomness:** Sets the intensity of the jittering movement.

See Also

- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Jitter Scale



The Text FX Jitter Scale plug-in animates a jittering effect on the scale of the character.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Jitter Scale Properties

- **Anchor Horizontal:** Sets the anchor point for the characters on the horizontal plane.
- **Anchor vertical:** Sets the anchor point for the characters on the vertical plane.
- **Use Axis:** Defines on which axis or axes the characters scale to create the jittering effect.
- **Lock Axis:** Gives all the axes get the same **Min**, **Max** and **Randomness** settings when enabled. If you disable randomness, these parameters are visible for each of the axes and must be set individually.
- **Min:** Sets the minimum scaling for the characters.
- **Max:** Sets the maximum scaling for the characters.
- **Randomness:** Sets the intensity of the jittering movement.

See Also

- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Plus Plus



The Text FX Plus Plus plug-in works in combination with the Pathfinder plug-in and allows to arrange a text among a defined spline.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

Note: This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Plus Plus Properties

- **Position Wrap**
  - **Interpolate:** Above 100% or below 0% of Path Position continues the direction of the spline.
  - **Clamp:** Stops the string on the spline endpoints.
  - **Repeat:** Causes the string to move to the other spline endpoint as soon as it moves over the endpoint.
  - **TextFx:** Positions characters by the relative spline of each character using the common TextFx parameters.
- **Scheme Type:** Defines how the text looks at 0% and 100% progress.
  - **In -> In:** at 0% first character is at the beginning of the spline, at 100% last character is at the end of the spline.
  - **In -> Out:** at 0% first character is at the beginning of the spline, at 100% first character is at the end of the spline.
  - **Out -> In:** at 0% last character is at the beginning of the spline, at 100% first character is at the end of the spline.
  - **Out -> Out:** at 0% last character is at the beginning of the spline, at 100% last character is at the end of the spline.
- **Progress (%):** Animates the progress of the effect(s).
- **Position:** Positions the characters using the spline as a position axis.
- **Rotation:** Rotates the characters using the spline as a rotation axis.
- **Scale:** Scales the character's X, Y and Z-axis using the spline as a scaling graph.
- **Color:** Applies Color Change on character.
- **Alpha:** Applies Alpha on the characters using the spline as a alpha graph.
- **General:** Enables rotation, offset and align properties.
  - **Rotate X/Y/Z:** Adds extra Rotation to each character.
  - **Text Offset:** Sets the text offset.
  - **Align:** Rotates the characters to follow the spline movements.
- **TextFX:** See Common Text FX Properties.

Copyright © 2026 Vizrt

    - **Direction:** Sets the direction of the text effect sequence. Normal starts with the first character (for example, left). Opposite starts with the last character (for example, right). Random uses a random order (see Random Seed). Static means that all characters are processed at the same time. Wave starts with the first character, animates the effect from 0 to 100% and then down again to 0%. Center starts the effect from the center of the text. 2 Center starts the effect at the same time from the beginning and the end of the text. They meet at the center.
- **Rotation:** Defines the rotation effect to apply on the characters for each axis.
- **Scale:** Defines the scale graph to apply on the characters for each axis.
- **Color:** Defines the color gradient effect to apply on the characters.
- **Alpha:** Defines the alpha effect to apply on the characters.

See Also

- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Rotate



The Text FX Rotate plug-in allows you to create an effect where the characters rotate on the X-, Y- or Z-axis.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	✅

⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Rotate Properties

- **Anchor Horizontal:** Sets the anchor point for the characters on the horizontal plane.
- **Anchor Vertical:** Sets the anchor point for the characters on the vertical plane.
- **Use Axis:** Defines on which axis the characters rotate.
- **Begin:** Sets the initial rotated position of the characters.
- **End:** Sets the ending rotated position of the characters.

See Also

- Expert
- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Scale



The Text FX Scale plug-in allows you to create a scaling animation of the characters.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

Note: This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Scale Properties

- **Anchor Horizontal:** Sets the anchor point for the characters on the horizontal plane.
- **Anchor Vertical:** Sets the anchor point for the characters on the vertical plane.
- **Use Axis:** Defines on which axis or axes the characters scale.
- **Lock Axis:** Scales the **Begin**- and **End** the same for all the axes when enabled. If you disable the parameter, **Begin** and **End** must be set for all the axes individually.
- **Begin:** Sets the initial size of the characters.
- **End:** Sets the ending size of the characters.

See Also

- Text FX Size
- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Size



The Text FX Scale plug-in allows you to create a sizing animation of the characters.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Note: This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Size Properties

- **Begin:** Sets the initial scale of the text before the effect.
- **End:** Sets the final scale of the text after the effect.

See Also

- Text FX Scale
- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Slide



The Text FX Slide plug-in allows you to create a sliding animation of the characters on the X, Y and Z-axis.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Note: This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Slide Properties

- **X, Y and Z Distance:** Defines direction and distance of moved text.
- **Show Direction:** Enables and disables view of movement vector.

See Also

- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Vertex Color



The Text FX Vertex Color plug-in adds a color per vertex effect to the text.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	❌

⚠ **Note:** Works only if the text is set to texture (does not work with geometry text).

⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Vertex Color Properties

- **Color Bbox:** Defines the reference bounding box, global for the entire text, or local for each character.
- **Color Direction:** Defines the color gradient shape. Available options are; top left to bottom left, top right to bottom right, top left to top right, bottom left to bottom right, top right to bottom left and four corners.
- **Color Level:** Sets the level of the color (`0.1` to `1.9`). Default is `1.0` .
- **Top and Bottom Left and Right:** Sets the color and alpha value for each edge.
- **Color use Alpha:** Enables alpha.

See Also

- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Vertex Explode



The Text FX Vertex Explode plug-in allows you to create an effect where text characters are exploded with a vertex effect.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	❌

⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Vertex Explode Properties

- **X, Y and Z Distance:** Defines direction and distance of moved text.
- **Show Direction:** Enables and disables view of movement vector.

See Also

- Text FX Explode
- Common Text FX Properties

Copyright © 2026 Vizrt

Text FX Write



The Text FX Write plug-in allows you to create an effect where text characters get shown as if they were written by the keyboard in real-time.

This creates a sort of Typewriter effect.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

The text field in the font editor does not need to contain any text, as the text for this plug-in is entered in the plug-in editor. Text FX Write expects UTF-8 encoded text as input, if a font encoding other than *Default* is selected (see the **Font/Text Options** page in the **Configuring Viz** section of the Viz Engine Administrator Guide).

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> TextFX

Text FX Write Properties

- **Fill with Space:** Adds a space for every character not yet written when enabled. The bounding box of the text is then sized correctly from the start. If you use an **Auto Follow** function connected to the text, enabling this option creates a better result.
- **Effect:** Allows you to select an effect to be executed as each character gets written on the screen. There are four effects to choose from and they have in addition their own parameters.
  - **Pop:** Creates the basic effect where the characters just pop up on the screen without any other sign being shown.
  - **Count:** Creates a counting sequence by showing a random symbol before showing the character to be shown.
  - **Random:** Writes random characters before the final character is shown (in combination with effect range). Use a non proportional font when you use this function to get the better looking effect.
  - **Cursor:** Shows a cursor that writes the characters. When you select this option, a text field opens below where you can choose which symbol to use as cursor.
- **Effect Speed:** Sets the speed of the effect that shows the preceding symbol. This is an additional parameter that is visible if you have selected **Count** or **Random**.
- **Selection:** Allows you to select between some preset character ranges to use for the effect. This is an additional parameter to the **Random** effect.
- **Cursor:** Allows you to enter the character which is to be used as a cursor.

See Also

- Common Text FX Properties

Copyright © 2026 Vizrt

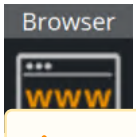
2.4.15 Texture Container Plug-ins

The following Container plug-ins are located in the Texture folder:

- Browser
- GeoGraffiti
- Grabbit
- GraffitiTex
- Image Clip
- Image Proportion
- Light Blur
- MoViz
- Noise
- SoftClip
- Texture Component
- VLC Plug-in

Copyright © 2026 Vizrt

Browser

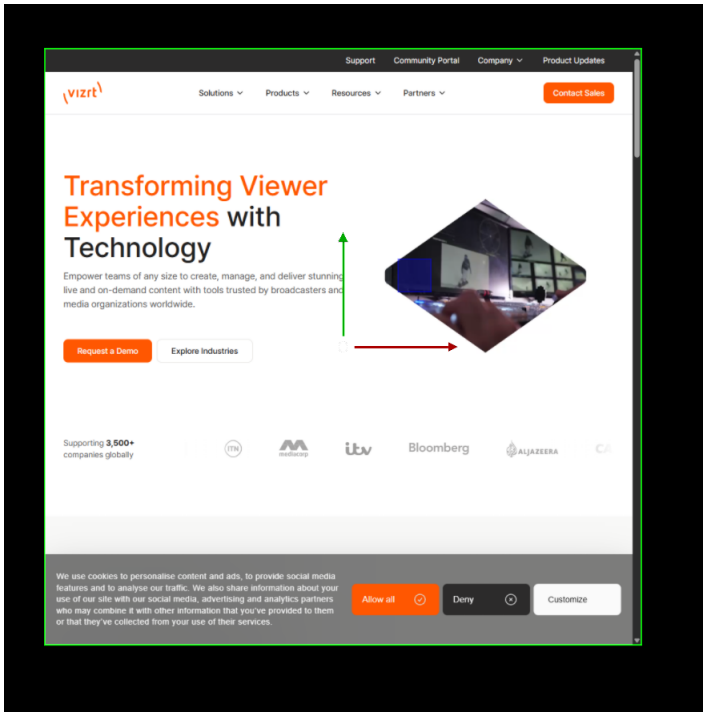


The Browser plug-in gives the ability to stream browser content on a Texture.

Note: Vizrt is not responsible for the content displayed or the technology used to display this content.

This page contains the following topics and procedures:

- Browser Properties
- To Create a Simple Scene with Browser
- Interact with a Scene with JavaScript
- Example HTML Document
- Browser Status Notifications
- Limitations



Browser Properties

- **URL:** Determines the URL address to be opened.
- **Texture Width:** Determines the width in pixels of the browser window.
- **Texture Height:** Determines the height in pixels of the browser window.
- **Allow Popups:** Allows the browser to render popup windows. The popup renders into the current texture. Close the popup by navigating back.

Copyright © 2026 Vizrt

- **Disable JavaScript:** Disables JavaScript execution.

> **Note:** Vector animations like Lottie require JavaScript to run.

- **Generate Mipmaps:** Generates all mipmap levels. Set to `Off` for more performance. Set to `On` for more quality when the texture is rendered on a smaller area on the screen than the width/height of the browser itself.
- **Alpha:** Enables a transparent browser window.

> **Note:** For the Viz Engine Renderer, Material Definitions' Rendering Method has to be set to Transparent for Alpha to work correctly. It is done automatically when Alpha is switched.

- **Snapshot Filename:** Saves a snapshot at the location (full path) entered here. The location can be either a local file or on the Graphic Hub and must be specified as a Uniform Resource Identifier (URI), for example:
  - **File:** *file:///c:/temp/Browser/example.png*
  - **Graphic Hub:** *vizgh:///Browser/example.png*
  - **Graphic Hub UUID:** *vizgh:///<F3F3F087-B85A-4AEE-B5019F78CB6AE5EC>/example.png*
- **Snapshot:** Takes a snapshot of the current browser texture. The snapshot is saved as a file or to the Graphic Hub, as specified in **Snapshot Filename.**
- **Navigate Back:** Navigates back to the previous page.
- **Navigate Forward:** Navigates forward to the next page.
- **Reload Page:** Reloads the current page.
- **Zoom In:** Zooms into the page, incrementing per click.
- **Zoom Out:** Zooms out of the page, incrementing per click.
- **Zoom Reset:** Resets zoom to default value.
- **Zoom:** Determines the desired zoom value. The fields supported range is from `-10.0` to `10.0`. The default value is `0.0`. Like any other parameter, the value of this field can be controlled from a ControlObject. Please enable **Show Advanced** and then add the parameter under **Channel**:

> **ControlObject Exposure with Direct Address**
> `zoom:MAIN_SCENETREE$BrowserFUNCTIONBrowser*zoom:float:0.0:-10.0:10.0:zoom level`

> **ControlObject Exposure with Control Channel Address**
> `zoom:MAIN_SCENETREE@zoom:float:0.0:-10.0:10.0:zoom level`

> **Note:** The Browser plug-in works asynchronously. Therefore, the zoom value needs to be applied after a successful load of an URL. This means that the value stored in the plug-in may not be the currently active zoom value until it has been applied.

- **ScrollBy X:** Sets the scroll in the horizontal axis.
- **ScrollBy Y:** Sets the scroll in the vertical axis.
- **ScrollBy:** Executes ScrollBy X and Y values.
- **Execute:** Executes the code entered in the code entry field.

Copyright © 2026 Vizrt

- **Disable Native Mouse Events:** Disables all mouse events, such as move, click and wheel events. Useful to control mouse events from outside, for example in scripts; or if it interferes with other plug-ins, such as Graffiti.
- **Mouse Position X and Y:** Controls the position of the mouse cursor with the u/v parameters. The coordinates are normalized in [0,1], being (0,0) the lower left corner and (1,1) the upper right corner of the browser window. All mouse click and move events described below relate to this coordinate:
  - **Left Click:** Emulates a click (both mouse down and mouse up events) of the left mouse button.
  - **Left Mouse Down:** Emulates a left mouse button down event.
  - **Left Mouse Up:** Emulates a left mouse button up event.
  - **Right Mouse Down:** Emulates a right mouse button down event.
  - **Right Mouse Up:** Emulates a right mouse button up event.
  - **Mouse Move:** Emulates a mouse move event (might be triggered after the u/v coordinate has been changed.
  - **Wheel Up:** Emulates a mouse wheel up event.
  - **Wheel Down:** Emulates a mouse wheel down event.
  - **Wheel Left:** Emulates a mouse wheel left event.
  - **Wheel Right:** Emulates a mouse wheel right event.
- **Proxy User:** Provides the user name in case the system's HTTP proxy requires authentication.
- **Proxy Password:** Provides the user name password in case the system's HTTP proxy requires authentication.
- **Read Events From Shared Memory:** Enables the browser to receive events from shared memory callbacks.
- **Input Events from Shared Memory:** Prefix key for global shared memory. Valid commands are:
  - `WB_TEXT` : Can be used to emulate string input like 'Hello World!'
  - `WB_KEY` : Can be used for special key strokes. Possible values are *ENTER*, *RETURN*, *BACKSPACE*, *DELETE*, *HOME*, *END*, *LEFT*, *RIGHT*, *UP*, *DOWN*, *PAGE_UP*, *PAGE_DOWN*.
    Sample for viz-scripting supposing ShmemVarName is WB_EVENTS:

    ```
    System.map["WB_EVENTS"] = "WB_TEXT Hello World!"
    System.map["WB_EVENTS"] = "WB_KEY ENTER"
    ```

    Note that a Shared Memory change event only occurs if the memory actually changes. Sending the same text twice, for example, does not trigger a Shared Memory event.

⚠ **Note:** Special characters (non-ASCII) like umlauts, etc. are not supported.

- **Enable Multitouch Events:** Enables native multitouch events (scroll, pinch to zoom, double tap).
- **Cache Browsing Data:** Stores already loaded webpages locally to make browsing faster.
- **Separate Audio Stream:** Integrates browser audio into Viz Engine`s audio system (and can be controlled via Viz Engine Audio volume controls), when enabled. This option can not be used on the fly.

⚠ **Note:** This setting is a global one and is effective after a Viz Engine restart and this scene is loaded for the first time.

Copyright © 2026 Vizrt

To Create a Simple Scene with Browser

1. Add the browser plug-in to a container
2. A quad and a Phong material is added automatically.
3. Open the Browser properties and enter a valid URL.
4. Click on  (enable the handling of interactive script/plug-in events (Scene Editor)).
5. Use the mouse or controls to navigate the website.

Interact with a Scene with JavaScript

It is possible to interact with a Scene from within the JavaScript of an HTML document (for example, *colorpicker.html*).

In the JavaScript section of a HTML document interact with the Scene with the **window.vizrt** object. Append any function name on this object, for example, **window.vizrt.colorChange("255 100 100")**. The function **colorChange** must be defined in the **JavaScript function** section of the **Browser** plug-in.

To add more functions, add them in this GUI section, separated by new lines:

Only one argument is supported for each function, to add more arguments to a function use separators and encode all arguments into one string.

Calls from JavaScript are translated into Viz Shared Memory variable changes, on the **system.map**. To register these changes use the Shared Memory variable composed by the Container ID of the Browser plug-in and the string, for example, `"BROWSER STATUS"`.

IMPORTANT! There must be a space between the double quotation mark (`"`) and BROWSER{{: " BROWSER STATUS"}}.

Script Example: When the script sits on the same Container:

```
sub OnInit()
    System.map.RegisterChangedCallback( this.VizId & " BROWSER STATUS" )
end sub

sub OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)
    println( "shared map changed: " & map[mapKey] )

    Dim str As String = map[mapKey]
    Dim strstr As array[String]

    str.split(" ", strstr)
    if strstr[1] = "javascript" then
        if strstr[2] = "colorChange" then
            scene.findContainer("iso").Material.diffuse.red = cDbl(strstr[3])/255.0
            scene.findContainer("iso").Material.diffuse.green = cDbl(strstr[4])/255.0
            scene.findContainer("iso").Material.diffuse.blue = cDbl(strstr[5])/255.0
        end if
    end if
end sub
```

Copyright © 2026 Vizrt

Example string passed to Shared Memory: `"0 JavaScript colorChange 0 53 4"`

- The first number, either `0` or `1`, can be ignored.
- `JavaScript`, shows that a JavaScript function was invoked.
- The function name (`colorChange`) itself followed by the argument(s).
  Example call in the HTML document: `window.vizrt.colorChange(r + " " + g  + " " + b);`

Example HTML Document

Entered in **URL:** file:///C:/test/colorpicker.htm (see Browser Properties):

```
<script type="text/javascript">

function objectmove(x, y)
{
  PositionX.value = x;
  PositionY.value = y;
}

</script>
<!DOCTYPE html>
<html lang="en">
<head>
    <title id='Description'>The jqxColorPicker widget allows you to easily pick a
color.
    </title>
    <link rel="stylesheet" href="http://www.jqwidgets.com/jquery-widgets-demo/
jqwidgets/styles/jqx.base.css" type="text/css" />
    <script type="text/javascript" src="http://www.jqwidgets.com/jquery-widgets-demo/
scripts/jquery-1.11.1.min.js"></script>
    <script type="text/javascript" src="http://www.jqwidgets.com/jquery-widgets-demo/
scripts/demos.js"></script>
    <script type="text/javascript" src="http://www.jqwidgets.com/jquery-widgets-demo/
jqwidgets/jqxcore.js"></script>
    <script type="text/javascript" src="http://www.jqwidgets.com/jquery-widgets-demo/
jqwidgets/jqxcolorpicker.js"></script>
    <script type="text/javascript" src="http://www.jqwidgets.com/jquery-widgets-demo/
jqwidgets/jqxradiobutton.js"></script>
</head>
<body class='default'>
    <div id='content'>
        <script type="text/javascript">
            $(document).ready(function () {
                // Create jqxColorPicker widgets.
                $("#jqxColorPicker").jqxColorPicker({ width: '250', height: '250'});
                $("#hueMode").jqxRadioButton({ width: '150', height: '25'});
                $("#saturationMode").jqxRadioButton({ checked: true, width: '180',
height: '25'});
                $("#hueMode").on('change', function (event) {
```

```
                    if (event.args.checked) {
                        $("#jqxColorPicker").jqxColorPicker({ colorMode: 'hue' });
                    }
                    else {
                        $("#jqxColorPicker").jqxColorPicker({ colorMode:
'saturation' });
                    }
                });
                $("#jqxColorPicker").on('colorchange', function (event) {
                    console.log(event.args.color);
                    window.vizrt.colorChange(event.args.color.r + " " +
event.args.color.g + " " + event.args.color.b);
                    $("#colorlog").html("<div>Color: #" + event.args.color.hex + "</
div>");
                });
            });
        </script>
        <div id='jqxWidget'>
            <div id="jqxColorPicker">
            </div>
            <div style="font-size: 13px; font-family: verdana; margin-top: 10px;">
                <div style="margin-bottom: 5px;">Color Modes</div>
                <div id="saturationMode">Saturation Color Mode</div>
                <div id="hueMode">Hue Color Mode</div>
            </div>
            <div style="font-family: Verdana; font-size:13px;" id="colorlog"></div>
        </div>
    </div>
</body>
</html>
```

Browser Status Notifications

The browser plug-in writes the following status updates into the System Shared Memory Map:

- **address <uri>:** Called when the browser URL changes.
- **title <title>:** Called when the browser title changed.
- **loadstate <started|stopped>:** Called when loading of a page started or stopped.

Use the container ID where the plug-in resides concatenated with the string *BROWSER STATUS* on the System Map to register for all Browser-related events.

```
sub OnInit()
    System.map.RegisterChangedCallback( this.VizId & " BROWSER STATUS" )
end sub

sub OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)
    if mapKey = this.VizId & " BROWSER STATUS" then
            println( "New status: " & map[mapKey] )
    end if
end sub
```

Limitations

- Because of licensing issues, some features are not enabled in the Browser plug-in. This affects for example the support for H.264/H.265 video, used by some streaming platforms. Visit https://html5test.opensuse.org/ to verify whether features are available.

Note: Rendering of webpages is synchronized to the output refresh rate of Viz Engine.

Copyright © 2026 Vizrt

GeoGraffiti



The GeoGraffiti plug-in draws on a globe.

Alpha is global to all colors and shapes, you must drag an Alpha object onto the container.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

GeoGraffiti Properties

- **Active:** Enables/disables drawing.
- **Texture Width:** Sets dimensions of the texture used for drawing canvas.
- **Brush Type:** Selects color or eraser brush.
- **Brush Image:** Determines the shape of the brush using an image. If empty, a round brush is used.
- **Brush Width:** Sets the width of the brush in pixels. Visible only if the color brush is selected.
- **Eraser Brush Width:** Sets the width of the eraser in pixels. Visible only if the eraser brush is selected.
- **Current Color**
- **Brush Color:** Selects the color of the brush.
- **Recognize Shapes:** Toggles shape recognition mode on or off. When enabled, additional parameters are enabled:
  - **Recognize Ellipse:** Specifies whether shape recognition tries to recognize ellipse shape.
  - **Recognize Circle:** Specifies whether shape recognition tries to recognize circle shape.
  - **Recognize Cross:** Specifies whether shape recognition tries to recognize cross shape.
  - **Shape Delay:** Determines the number of frames to wait from mouse up before trying to recognize shapes.
  - **Draw Arrow:** Specifies whether non-recognizable shapes are converted to an arrow.
  - **Arrow Length:** Sets the length of an arrow head.
  - **Arrow Width:** Sets the width of an arrow head.
  - **Circle Rad:** Determines the radius of the circle replacing a recognized circle. If zero, the radius of the recognized circle is used.
  - **Cross Width:** Determines the width of the cross replacing the recognized cross shape. If zero, the width of the recognized cross shape is used.
- **Enable Undo After Clear:** Enables undo after the Clear button has been used.
- **Clear (button):** Clears the canvas.
- **Undo (button):** Undoes an action.
- **Redo (button):** Redoes an action.

Copyright © 2026 Vizrt

Grabbit



The Grabbit plug-in gives the ability to use captured video inside Viz Engine. Grabbit uses the Microsoft DirectShow Filter graph framework to capture the video.

Devices which are supported by DirectShow and are self-contained such as capture cards or web cameras, can be used. Self-contained means the capture device does not need any further input.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Texture

This page contains the following topics and procedures:

- Grabbit Properties
- Required Software
- Best Practices
- Known Limitations
- To Capture a Video Stream with Grabbit
- To Capture a Video Stream with Grabbit and a VfW Supported Device

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Grabbit Properties

Following is a list of all parameters of the plug-in. Depending on the state of the plug-in, they may be hidden. Some of them are not visible at all.

- **Available Devices:** Decides which video capture device to use. This list is created when the plug-in loads. That means if you have a capture device which is not known at startup of the renderer, it does not show up. On the other hand, it does not know if a capture device is already in use by some other plug-in. Since the list is static, it just shows all capture devices which were available at startup. If the parameter is changed and the plug-in is already started, it stops capturing before loading the new capture device. Changes are effective immediately.
  - Parameter name: `CaptureDevices (list)`.
- **Selected Capture Device:** Contains the simple name for the chosen capture device from the list. This parameter decides which DirectShow Device is taken for capturing. If the parameter is changed and the plug-in is already started, it stops the capturing before loading the new capture device.
  - Parameter name: `CaptureDevice (string)`.

Note: This parameter is hidden from the user interface.

- **Capture Format:** Represents the current capturing format as a human readable string. It shows the width by height, frame rate, video subtype, and bit depth. It is a read only resource and is changed with the configuration dialogs below. This should really be an opaque data type.

Copyright © 2026 Vizrt

- Parameter name: `CaptureFormat (string)`. Parameters are shown if the filter supports the ProcAmp interface.

> ⚠ **Note:** This parameter is hidden from the user interface.

- **Brightness:** Changes the brightness. The range is relative and the value is re-calculated internally according to the possibilities (min/max/stepping) of the device. Changes are effective immediately but not for already captured video.
  - Parameter Name: `Brightness`.
  - Type: float. Range: [`0-100`]. Default: `50`.
- **Contrast:** Changes the contrast. The range is relative and the value is re-calculated internally according to the possibilities (min/max/stepping) of the device. Changes are effective immediately but not for already captured video.
  - Parameter Name: `Contrast`.
  - Type: float. Range: [`0-100`]. Default: `50`.
- **Hue:** Changes the hue. The range is relative and the value is re-calculated internally according to the possibilities (min/max/stepping) of the device. Changes are effective immediately but not for already captured video.
  - Parameter Name: `Hue`.
  - Type: float. Range: [`0-100`]. Default: `50`.
- **Saturation:** Changes the saturation. The range is relative and the value is re-calculated internally according to the possibilities (min/max/stepping) of the device. Changes are effective immediately but not for already captured video.
  - Parameter Name: `Saturation`.
  - Type: float. Range: [`0-100`]. Default: `50`.
- **Sharpness:** Changes the sharpness. The range is relative and the value is re-calculated internally according to the possibilities (min/max/stepping) of the device. Changes are effective immediately but not for already captured video.
  - Parameter Name: `Sharpness`.
  - Type: float. Range: [`0-100`]. Default: `50`.
- **Gamma:** Changes the gamma. The range is relative and the value is re-calculated internally according to the possibilities (min/max/stepping) of the device. Changes are effective immediately but not for already captured video.
  - Parameter Name: `Gamma`.
  - Type: float. Range: [`0-100`]. Default: `50`.
- **Colorenable:** Enables or disables color. Changes are effective immediately but not for already captured video.
  - Parameter Name: `Colorenable`.
  - Type: toggle button. Default: `on`.
- **Whitebalance:** Changes the white balance. The range is relative and the value is re-calculated internally according to the possibilities (min/max/stepping) of the device. Changes are effective immediately but not for already captured video.

Copyright © 2026 Vizrt

    - Parameter Name: `Whitebalance`.
    - Type: float. Range: [`0-100`]. Default: `50`.
- **Backlight Compensation:** Enables or disables the back light compensation. Changes are effective immediately but not for already captured video.
    - Parameter Name: `BacklightComp`.
    - Type: toggle button. Default: `on`.
- **Gain:** Changes the gain. The range is relative and the value is re-calculated internally according to the possibilities (min/max/stepping) of the device. Changes are effective immediately but not for already captured video.
    - Parameter Name: `Gain`.
    - Type: float. Range: [`0-100`]. Default: `50`.

      Parameters are shown if the filter supports the DirectShow dialog interface. Since the dialogs are handled by Windows and not the Viz Artist GUI they may open in the background. If so please bring the dialog to the front and make your settings. Make sure that the dialog is closed before you change anything in the Viz Artist GUI.

The buttons are disabled if there is no such dialog. Some implementations hint that a dialog is available even if it is not. In that case the button is not disabled but no dialog window shows.

- **Video Filter:** The dialog window may be behind the renderer GUI after activation. Please be careful to close the dialog window. Otherwise the renderer and the GUI may lock up. Changes are effective immediately.
    - Parameter Name: `ButtonVideoFilter`.
    - Type: push button.
- **Video Capture Pin:** If available choose RGB24 or UYV as format. Other settings may not work. The dialog window may be behind the renderer GUI after activation. Please be careful to close the dialog window. Otherwise the renderer and the GUI may lock up. Changes are effective immediately.
    - Parameter Name: `ButtonVideoCapPin`.
    - Type: push button.
- **Audio Filter:** The dialog window may be behind the renderer GUI after activation. Please be careful to close the dialog window. Otherwise the renderer and the GUI may lock up. Changes are effective immediately.
    - Parameter Name: `ButtonAudioFilter`.
    - Type: push button.
- **Audio Capture Pin:** The dialog window may be behind the renderer GUI after activation. Please be careful to close the dialog window. Otherwise the renderer and the GUI may lock up. Changes are effective immediately.
    - Parameter Name: `ButtonAudioCapPin`.
    - Type: push button.
- **Crossbar 1:** The dialog window may be behind the renderer GUI after activation. Please be careful to close the dialog window. Otherwise the renderer and the GUI may lock up.
    - Parameter Name: `ButtonCrossbar1`.
    - Type: push button.
- **Crossbar 2:** The dialog window may be behind the renderer GUI after activation. Please be careful to close the dialog window. Otherwise the renderer and the GUI may lock up.
    - Parameter Name: `ButtonCrossbar2`.

Copyright © 2026 Vizrt

    - Type: push button.
- **TV-Tuner Video:** The dialog window may be behind the renderer GUI after activation. Please be careful to close the dialog window. Otherwise the renderer and the GUI may lock up.
    - Parameter Name: `ButtonTvVideo`.
    - Type: push button.
- **TV-Tuner Audio:** The dialog window may be behind the renderer GUI after activation. Please be careful to close the dialog window. Otherwise the renderer and the GUI may lock up.
    - Parameter Name: `ButtonTvAudio`.
    - Type: push button.
      Video for Windows Dialogs:

Parameters are shown if the filter supports the Video for Windows (VfW) dialog interface. Since the dialogs are handled by Windows and not the Viz Artist GUI they may open in the background. If so please bring the dialog to the front and make your settings. Make sure that the dialog is closed before you change anything in the Viz Artist GUI.

Buttons are disabled if there is no such dialog. Some implementations hint that a dialog is available even if it is not. In that case the button is not disabled but no dialog window shows.

- **Cap Source:** If the filter supports the VfW Source dialog this button is present. It can be activated if the capturing is stopped otherwise the button is grayed out. The dialog window may be behind the renderer GUI after activation. please be careful to close the dialog window. otherwise the renderer and the GUI may act in a strange way. Changes are effective immediately.
    - Parameter Name: `ButtonCaptureSource`.
    - Type: push button.
- **Cap Format:** If the filter supports the VfW Format dialog this button is present. It can be activated if the capturing is stopped otherwise the button is greyed out. The dialog window may be behind the renderer GUI after activation. please be careful to close the dialog window. otherwise the renderer and the GUI may act in a strange way. Changes are effective immediately.
    - Parameter Name: `ButtonCaptureFormat`.
    - Type: push button.
- **Cap Display:** If the filter supports the VfW Display dialog this button is present. It can be activated if the capturing is stopped otherwise the button is grayed out. The dialog window may be behind the renderer GUI after activation. please be careful to close the dialog window. otherwise the renderer and the GUI may act in a strange way. Changes are effective immediately.
    - Parameter Name: `ButtonCaptureDisplay`.
    - Type: push button.
- **Auto Scale Texture:** If enabled the captured video is scaled according to the underlying geometry. Changes are effective immediately but not for already captured video.
    - Parameter Name: `AutoScale`.
    - Type: toggle button. Default: `on`.
- **Border Crop Width:** Crop pixels on the horizontal sides of the video. This parameter is shown only if Auto Scale Texture is enabled. Changes are effective immediately but not for already captured video.
    - Parameter Name: `BorderCrop_x`.
    - Type: integer. Range: [0-0x7fff]. Default: 0.
- **Border Crop Height:** Crop pixels on the vertical sides of the video. This parameter is shown only if Auto Scale Texture is enabled. Changes are effective immediately but not for already captured video.

Copyright © 2026 Vizrt

    - Parameter Name: `BorderCrop_y`.
    - Type: integer. Range: [0-0x7fff]. Default: 0.
- **Status Opened:** Shows if the capture graph has been built successfully.
    - Parameter Name: `StatusOpened`.
    - Type: toggle button (show only).
- **Status Started:** Shows if the capture graph has been started.
    - Parameter Name: `StatusStarted`.
    - Type: toggle button (show only).
- **Status Stream Width:** Shows the captured video width.
    - Parameter Name: `StatusStream_x`.
    - Type: integer (show only).
- **Status Stream Height:** Shows the captured video height.
    - Parameter Name: `StatusStream_y`.
    - Type: integer (show only).
- **Start:** Starts capturing. Feedback is given through Status Started and Status Stream Width/Height. If capturing does not start check the Status Opened and Connected. Capturing cannot commence if the filter graph is not connected. Changes are effective immediately.
    - Parameter Name: `ButtonStart`.
    - Type: push button.
- **Stop:** Stops capturing. You need to do this to select the Vfw Dialog buttons. Changes are effective immediately.
    - Parameter Name: `ButtonStop`.
    - Type: push button.
- **Clear:** Clears the texture to opaque white. You may need to hit the button twice for the change to show. Changes are effective immediately.
    - Parameter Name: `ButtonClear`.
    - Type: push button.

Required Software

- DirectShow version 9.0c. It is strongly recommended to have a clean DirectShow installation to start with. Additional source filters may be added later. Find more Information at the
- DirectShow of VfW compatible capture card.

Best Practices

GraphEdit is a tool from the Microsoft DirectShow SDK that allows you to visualize the default Filtergraph that DirectShow builds to render the Media. If you can render the Media inside GraphEdit it should play in Viz Engine, too. If it does not play in GraphEdit, it does not play in Viz Engine.

Known Limitations

- Dynamic listbox for capture devices.
- Handle the capture device list dynamically.
- ProcAmp settings should be updated from dialog settings as well.

Copyright © 2026 Vizrt

- Query for availability in the various ProcAmps.
- The Capture Device must be known to the system at application start. When using USB devices this means they must be connected before Viz Engine is started.

To Capture a Video Stream with Grabbit

1. Add the Rectangle geometry to the Scene Tree.
2. Add the Grabbit plug-in to it.
3. Select the capture device to work with from the **Available Capture Devices** list.
    - You should get a positive **Status Opened** feedback.

> **Note:** Do not select a device which is currently in use (either by another instance of this plug-in or by another application).

4. If you have a device that supports the capture format enumeration and the ProcAmp interface you get a screen similar to the following.
5. Set the **Capture Format** by using the dialogs behind the configuration buttons Video Filter, and so on.
6. Select **Auto Scale Texture** if you want the texture position and scaling set by this plug-in so that the video fills out the entire geometry. If you have selected this feature you may want to crop something on the horizontal or vertical sides of the video. Use the Border Crop Width and Border Crop Height for this effect.
7. Click the **Start** button to start capturing. You should get a positive *Status Started* feedback. In addition, the *Status Stream Width* and *Status Stream Height* should reflect the dimensions of the captured video.
8. Adjust the effect settings. You might want to experiment with them. Some parameters have discreet values as for example the *Gamma* parameter.

> **Note:** The effect settings are highly dependent on the selected device.

To Capture a Video Stream with Grabbit and a VfW Supported Device

> **Note:** If you have a device which supports the VfW Dialogs interface, you should see that your device is tagged with VfW.

Once you have selected the capture device it needs to be configured. You do this by clicking the buttons that leads to the VfW dialog windows.

1. Click **Cap Source** and **Cap Format** and set the parameters according to your needs.
    - The values you set here cannot be changed via the usual control mechanism. You need to pay close attention that the dialog window may open in the behind the Viz Artist GUI.
    - Make sure that each dialog window is closed before any other changes are made in the Viz Artist GUI.
    - You should get a positive **Status Connected** feedback. If you want to go back to change something here the graph must be in Stopped mode.
2. Select **Auto Scale Texture** if you want the texture position and scaling set by this plug-in so that the video fills out the entire geometry.
    - If you have selected this feature you can use the **Border Crop Width** and **Border Crop Height** to crop something on the horizontal or vertical sides of the video.
3. Click the **Start** button to start capturing.
    - You should get a positive **Status Started** feedback. In addition the **Status Stream Width** and **Status Stream Height** should reflect the dimensions of the captured video.

Copyright © 2026 Vizrt

4. Click the **Stop** button to stop the capturing.
    - This sets it back to Stopped mode.
    - You should see the feedback in the **Status Started** toggle.
    - Once the capturing is stopped you can use the **Clear** button to set the underlying texture to opaque white.

Copyright © 2026 Vizrt

GraffitiTex



The Graffiti Texture plug-in gives the ability to draw freehand on top of flat containers, such as rectangles. The freehand draw is created with a brush shape, used with a mouse, 6DOF device, or multi touch device. The plug-in can also recognize some rendered shapes, such as circles, ellipses, crosses, and arrows, and replaces the hand-drawn item with the recognized shape.

6DOF (6 Degrees of Freedom) events are triggered for all layers, but the Main Layer (see Layer Manager) has priority over the Front and Back Layers. This means that if the Front Layer scene grid is to be used for 6DOF, then the grid in the Front Layer must be set to **active,** and the grid in the Main Layer must to be set to **inactive**. If there is no grid defined in any of the three Layers, an orthogonal XY 2D grid is used (see Grid Tool-bar).

The plug-in can either work in combination with Viz Engine's multi-texturing technology or can add a standalone texture.

There are two graffiti plug-ins; the scene plug-in Graffiti which is used globally for the scene, and this one which is applied on the container level.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Texture

This page contains the following topics and procedures:

- GraffitiTex Properties
- To Create a Container Level Graffiti Effect

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

GraffitiTex Properties

- **Active:** Enables/disables drawing.
- **Multi Telestration:** Allows handling of multiple touch events for multi-touch devices. Some devices incorporate a pressure factor; this influences the width of the brush:
  - **Pressure Min %:** Sets the minimum width of the brush.
  - **Pressure Max %:** Sets the maximum width of the brush.
- **I6DOF:** Specifies whether input comes from mouse of 6DOF.
- **Transparent Base:** Specifies whether the base color of the container is the background of the rendered brush or whether the brush is drawn on a transparent background.
- **Texture Width, Texture Height:** Sets dimensions of the texture used for drawing canvas.
- **Brush Type:** Selects color or eraser brush.
- **Brush Image:** Determines the shape of the brush using an image. If empty, a round brush is used.
- **Brush Width:** Sets the width of the brush in pixels. Visible only if the color brush is selected.
- **Eraser Brush Width:** Sets the width of the eraser in pixels. Visible only if the eraser brush is selected.
- **Brush Color:** Selects the color of the brush.
- **Recognize Shapes:** Toggles shape recognition mode on or off.
  - **Recognize Ellipse:** Specifies whether shape recognition tries to recognize ellipse shape.

Copyright © 2026 Vizrt

- **Recognize Circle:** Specifies whether shape recognition tries to recognize circle shape.
- **Recognize Cross:** Specifies whether shape recognition tries to recognize cross shape.
- **Shape Delay:** Determines the number of frames to wait from mouse up before trying to recognize shapes.
- **Draw Arrow:** Specifies whether non-recognizable shapes are converted to an arrow.
- **Arrow Length:** Sets the length of an arrow head.
- **Arrow Width:** Sets the width of an arrow head.
- **Circle Rad:** Determines the radius of the circle replacing a recognized circle. If zero, the radius of the recognized circle is used.
- **Cross Width:** Determines the width of the cross replacing the recognized cross shape. If zero, the width of the recognized cross shape is used.

- **Enable Undo After Clear:** Enables undo after the Clear button has been used.
- **Clear (button):** Clears the canvas.
- **Undo (button):** Undoes an action.

To Create a Container Level Graffiti Effect



- First, add the plug-in to the container and set the plug-in properties.
- Then set Viz Artist in On Air Mode, and start drawing.

Copyright © 2026 Vizrt

Image Clip



The Image Clip plug-in is designed to play back a sequence of still images (for example, TGA or TIFF) rather than playing a movie file. It also supports alpha channel and various play modes (looping, swing, etc). The plug-in loads all images to RAM, then loads only one image at a time to texture memory. The download time is almost linear to the size of each frame. Preparation for this plug-in should be uncomplicated as most video applications has support for saving out TIFF. Since it plays the sequence from memory, one obvious benefit is that once loaded into memory it does not access the hard drive at all. So if it is needed to play back ten different sequences at once, ImageClip is pretty much the only solution. It is ideal for looping small animations such as logos. The drawback is that it consumes system memory.

Remember to keep track of how much memory each sequence needs, to avoid running out of memory on the Viz Engine machine. Running low on memory makes the system start caching parts of memory to the hard drive and it may not play the sequence and graphics in real-time.

Image clip should not be used with very large clips in Continuous and Array mode. Since all images are loaded into memory a large number of images or a large image size would require large amounts of memory.

Memory can be calculated as follows:

- Number of images * Image width * Image height * 3 (or 4 when using alpha)
  The use of Thread mode solves the memory issue, but it takes longer to request a frame. Therefore it should be used only in a slow motion image clip.

Images for an image sequence should be placed in specific folders. For example if you out many flag image sequences, you would have a separate folder for each flag image sequence. Make sure that the images are named sequentially (for example, *england001.tif*, *england002.tif*, *england003.tif* etc).

Note: This plug-in is located in: Plugins -> Container plug-ins -> Texture

Known Issue: If the Image Clip plug-in is used to display compressed image sequences like *.vbn*, please be aware that any image operation like resizing (keep under option) or format conversations (for example, to luminance) will fail because of the missing original data. This is a known limitation.

This page contains the following topics and procedures:

- Image Clip Properties
- To Add an Image Clip

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Image Clip Properties

- **Image:** Sets the image path and the first image you would like to use in the animation. Do not use clip names with numbers (except the counter (for example, 000, 001, 002, etc.)).
- **Play Mode:** Sets the play mode:

Copyright © 2026 Vizrt

    - Once
    - Loop
    - Swing
    - Animation
    - Anim-Dissolve
- **Reverse:** Reverses the animation (not in **Animation** play mode).
- **Animation Position:** Sets the animation position (for example, where to start and stop the animation). The position is a counter for the number of images in the folder referenced by the Image setting. This setting is enabled when **Play Mode** is set to **Animation**.
- **Play Speed:** Sets the speed of the animation.
- **Advanced:** Enables the advanced settings (see below).
- **Time Mode:** Enables time specific settings.
- **Stand Alone Image:** Enables the same clip to be played in different speeds.
- **First Image:** Defines the first image of the animation.
- **Nof Image:** Defines the number of images, relative to the First Image, that should be part of the animation.
- **Geo Ref:**

> **Info:** When changing the *Crop* or *Scale By* parameters, **Reinitialization** is needed.

- **Crop:** Crops the image (by percentage) from the left, right, bottom and top side.
- **Scale By:** Enables the scale setting.
    - **Scale:** Scales all images to the closest power of two. When not selected, automatic texture coordinates are applied so the image fits the texture.
- **Keep Under:** Forces the image size. For example if you have selected 64, the image is trimmed to the size 64 x 64.
- **Base Path:**
- **Format:**
- **Weather:**
- **Blend Images:**
    - **Dissolve Speed:**
- **Control Texture Map:**
- **Memory Type:** Sets the memory type.
    - **Continuous:** Uses one big chunk of memory to store all images.
    - **Array:** Uses divided chunks of memory to store all images.
    - **Thread:** Loads only requested frame on-the-fly, with [n] frames loading time limit.
- **Mapping:**
- **Play (button):** Plays the animation.
- **Stop (button):** Stops the animation.
- **Reinitialize (button):** Reinitializes all settings.

To Add an Image Clip

1. Add a Container to the Scene Tree.
2. Add the Image Clip plug-in to it.
3. In the Image Clip properties click the **Browse** ( ) button to locate the folder that has the required images.
4. Select the first image in the sequence.

Copyright © 2026 Vizrt

5. Click **OK**. The image sequence now loads into the ImageClip plug-in, and is visible in the scene.
6. Click **Play** to test the image sequence. If you change the image sequence and you need to reload it, click the Reinitialize button.
7. Change the Play Mode to Loop to play the image sequence continuously. The image sequence starts playing automatically.
8. Change the Play Mode to Animation to animate the image sequence frame position to control the sequence within the Stage Editor. The image sequence now stops playing and a new property is revealed that is called Animation Position. This refers to the frame number that is currently shown for the image sequence.
9. Animate the Animation Position value. The Animation Position maximum value is restricted to the number of images.
10. Another possibility is to create a file with a *.vln* extension. This file includes the base path and also the names of the images to load. In this case the images must not have a counter number in their filename. You can handle this file as an ordinary text file.
11. Load the *.vln* file instead of loading an image file located in a directory.
    Example:

```
BASE_PATH 'C:/clip/images'
{
  'radar_200504110800.png' 2005_04_11_10:00
  'radar_200504110815.png' 2005_04_11_10:15
  'radar_200504110830.png' 2005_04_11_10:30
  'radar_200504110845.png' 2005_04_11_10:45
  'radar_200504110900.png' 2005_04_11_11:00
  'radar_200504110915.png' 2005_04_11_11:15
  'radar_200504110930.png' 2005_04_11_11:30
  'radar_200504110945.png' 2005_04_11_11:45
}
```

Note: Viz Engine does not interpret between Key Frames, so if you choose to slow down an animation then the motion is not smooth.

See Also

- MoViz
- SoftClip

Copyright © 2026 Vizrt

Image Proportion



The Image Proportion plug-in maps an image to a container, maintaining the correct aspect ratio and allowing some constraints, such as whether the entire image should be visible, or the entire container should be filled.

The plug-in takes into account the container geometry's dimensions, which means it needs a geometry other than the standard image geometry, as the image geometry automatically takes the proportions of the image itself.

Note: This plug-in is obsolete as the functionality is now within Image Channels / Super Channels.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Texture

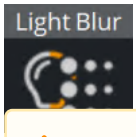
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Image Proportion Properties

- **Border Color:** Adds a border color to the container.
- **Show Entire Image:** Determines whether the entire image is shown.
- **Allow Stretch:** Permits stretching the image to fill the container.
- **Auto convert to RGBA:** Ensures border colors and transparency can be applied independent of the texture format when enabled. This is an expensive operation and causes jittering when an image with another format is put on the same container during playout.

Copyright © 2026 Vizrt

Light Blur



The Light Blur plug-in allows the user to configure and create a light blurred image.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Texture

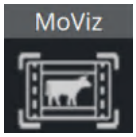
IMPORTANT! The light blur functionality is now a built in feature of the Viz Artist/Viz Engine core. For future use, it is recommended to use the **Dynamic Texture** Media Asset.

Light Blur Properties

- **Width and Height:** Sets the width and height of the blurred light. Available options are `2` to `2048`.
- **Format:** Sets the image format. Options are:
  - RGB
  - RGBA
  - Alpha
  - Luminance
  - Luminance Alpha
- **Save:** Saves the blur as an image.
- **X and Y (%):** Sets the position of the blur on the X and Y axis (50% places the blur at the origin).
- **Spread (%):** Sets the spread of the blur.
- **Start:** Sets the start color of the blur.
- **End:** Sets the end color of the blur.

Copyright © 2026 Vizrt

MoViz



The MoViz plug-in allows media files or streams to be played inside Viz Engine.

MoViz is not a QuickTime player or an AVI player. It uses the Microsoft DirectShow Filter Graph framework to play the media files, which means it can play both types, but only certain installed Filter Graphs (some of the QuickTime and some AVI).

MoViz is not natively QuickTime, it uses the AVI wrappers and filters for QuickTime and AVIs. This means that not all QuickTime codecs can be used, only the ones that are supported within DirectX wrappers and filters.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Texture

Note: Throughout this plug-in documentation, media file and media stream is referred to as media and may be used interchangeably unless stated otherwise.

Warning: Always make sure to install the required codecs for used media files. Missing codecs can lead to Viz Engine crashes.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

MoViz Properties

- **Clip URI:** Click the **Browse** button (...) to load clips from the file system (or enter the location directly). It takes the form of a valid URI (for example, *http://...* or *mms://...*). If the parameter is changed and the plug-in is already started it stops the playback before loading a new clip. Changes are effective immediately.

IMPORTANT! All video clips from the file system should be stored locally on the *D:* drive of the machine that is used to play out the graphics. Remember to save the clip locally on the client machines with preview (for example, Viz Trio) and use the same *D:* drive so that the clip can be seen in preview.

- **Clip URI Stereo:** Path to Clip that is loaded for the right eye camera in a stereo configuration with Stereo Mode = Right Eye. You can load clips from the file system by using the **Browse** button, or by typing the path location directly. It takes the form of a valid URI (for example, http://... or *mms://...*). If you change the parameter after starting the plug-in, it stops the playback before loading the new clip. Changes are effective immediately.
- **Autoplay:** Sets the automatic start of the clip after it has loaded, to On or Off. Changes are effective immediately (Default is off).
- **Loop Clip:** Sets looped playback to On or Off. If the clip position is at the clip out position or EOF it rewinds to the clip in position. Changes are effective for the next EOF, clip out position (Default is `Off`).

Copyright © 2026 Vizrt

- **Sync to Video Counter:** Enables or disables the use of the system retrace counter. If enabled, the clip is played back with the speed of Viz's internal retrace counter. This is the tick count of the SDI output. If disabled, the clip plays back at the speed of its own internal clock.
- **Clip Duration (ms):** Shows the duration of the clip in milliseconds (read only). This parameter makes only sense with media that supports this kind of information.
- **Clip Position (ms):** Gives the current position of the clip in milliseconds. When in pause mode this parameter may be changed to seek to the gives time. Do not change this parameter when not in pause mode. The position may be changed between Clip Position IN and Clip Position OUT. This parameter makes only sense with media that supports this kind of information. Changes are effective immediately.
- **Clip Position IN (ms):** Gives the first frame where the clip should start in milliseconds. This parameter makes only sense with media that supports this kind of information. Changes are effective immediately.
- **Clip Position OUT (ms):** Gives the last frame where the clip should stop in milliseconds. This parameter makes only sense with media that supports this kind of information. Changes are effective immediately.
- **Video Device Format:** Selects a video format for rendering. Changes are effective at load time.
    - **None:** Does not render the video portion.
    - **RGB24:** Delivers the video portion to Viz Artist/Engine with the RGB24 format. This format uses eight bits per Red, Green and Blue component, hence the name RGB24.
    - **ARGB32:** Delivers the video portion to Viz Artist/Engine with the ARGB32 format. This format uses eight bits per Red, Green and Blue component and in addition eight bits for Alpha (8-bits * 4 components), hence the name ARGB32 indicating RGB with Alpha. This format is often referred to as RGBA.
- **Force Opaque:** (ARGB32 only) Sets the alpha value to 100% opaque (if required) if the clip has an alpha channel. Changes are effective for each frame delivered.
- **Auto Scale Texture:** Scales captured video according to underlying geometry when enabled. Changes are effective immediately but not for already delivered video.
    - **Border Crop Width:** Crops pixels on the horizontal sides of the video. Changes are effective immediately but not for already delivered video.
    - **Border Crop Height:** Crops pixels on the vertical sides of the video. Changes are effective immediately but not for already delivered video.
- **Audio Device/Format:** Selects which audio device/format to use for rendering. Changes are effective immediately but not for already delivered audio. For performance reasons, `None` is the recommended setting.
    - **None:** Does not render the audio portion.
    - **Default Device:** Renders the audio portion using the default DirectShow renderer.
- **Status Loaded:** Shows if the clip has loaded.
- **Status EOF:** Shows if the clip is at EndOfFile. Status EOF allows user defined actions upon registering for change requests in a script. Whenever a change in the parameter `StatusEof` is happening and a Viz System Shared Memory (SHM) variable named `MoViz[A,B]_eof` exists, this variable is set to `1` when **Status EOF** is `On`, and to `0` when Status EOF is `Off`. The variable's parameter `A` is the hexadecimal value of the *scene ID* in uppercase A-F range. The variable's parameter `B` is the hexadecimal value of the `Container ID` in uppercase A-F range.

> ⚠ **Note:** Since Viz Engine 3.8.1, this is supported for MoViz Version 2.20.0 onwards.

Copyright © 2026 Vizrt

- **Status Clip Width:** Shows the video width.
- **Status Clip Height:** Shows the video height.
- **Status Video:** Shows if the clip has a usable video component.
- **Status Audio:** Shows if the clip has a usable audio component.
- **Seek Timeout:** Specifies the timeout for seeking to a position in the movie file. Higher values can prevent jump back to the expected position when clip playback takes longer to start then expected. However, the playback time may not be as exact.
- **Eject:** Unloads the clip and destroy the graph. Changes are effective immediately.
- **Load:** Loads the given clip and builds the graph. Changes are effective immediately.
- **Rewind:** Rewinds the clip to the start, when supported. Changes are effective immediately.
- **Stop:** Puts the graph in stop mode. Changes are effective immediately.
- **Play:** Puts the graph in play mode (start playback). Changes are effective immediately.
- **Pause:** Puts the graph in pause mode. Changes are effective immediately.
- **Play From Start:** Rewinds the clip to the start before playback. Changes are effective immediately.
- **Clear:** Clears the texture to opaque white.



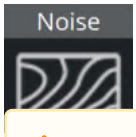
Note: Moviz handles videos with alpha, but the codecs in use must provide a valid ARGB32 stream and Containers you want visible should use the KEY Function, otherwise the alpha channel is blank, making the whole video totally transparent. See also Advanced Issues with Video Codecs in the Viz Artist User Guide.

See Also

- Image Clip
- SoftClip

Copyright © 2026 Vizrt

Noise



The Noise plug-in allows the user to configure and create a noise image.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Texture

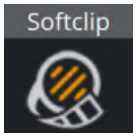
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Noise Properties

- **Width and Height:** Sets the width and height of the noise. Available options are `2` to `2048`.
- **Format:** Sets the image format. Available options are *RGB*, *RGBA*, *Alpha*, *Luminance* and *Luminance Alpha*.
- **Save (button):** Saves the noise as an image.
- **Factor X and Y:** Sets the noise factor on the X and Y axis (range is from `0.000` to `1000.000`).
- **Turbulence 1, 2 and 3:** Sets the turbulence of the noise.
- **Frequency:** Sets the frequency of the noise.
- **Exponent:** Sets the exponent of the noise.
- **Fg:** Sets the foreground color.
- **Bg:** Sets the background color.
- **Wave, Clouds, Marble and Noise (buttons):** Allows the user to select between four different presets.
- **Randomize (button):** Sets random factor, turbulence and frequency. Randomize works together with the Wave, Clouds, Marble and Noise selections.

Copyright © 2026 Vizrt

SoftClip



The SoftClip plug-in is used for playing video clips projected on a texture, and supports AVI files only. This format supports RGB and RGBA (alpha channel).

In its simplest form, the SoftClip plug-in can exist on an empty container to play back a video clip in Viz Artist/Engine. Other plug-ins can be mixed with SoftClip to obtain different results. For example you can apply a SoftClip plug-in to a Sphere and add a Material to affect the overall shape, color and shading of the video clip.

With SoftClip you are able to control the video clip within the stage, unlike with the **Live Video** Media Asset method. The video clip size can also be whatever you decide, however, it is recommended that you keep the dimensions in multiples of 8 (for example, 128x128 or 256x512) for performance purposes.

Use of the SoftClip plug-in might have an affect on performance and quality, hence, it is recommended to use the Performance Editor when working with the SoftClip plug-in.

IMPORTANT! Viz Engine keeps all designer, operator and automation loaded scenes and plug-ins in memory. As a consequence, if 30 different templates that load 30 different scenes are loaded, that each has a single SoftClip plug-in on it, Viz Engine has thirty SoftClip plug-ins with 30 threads activated in memory. If every plug-in has specified that it should have ten preloaded frames, then Viz Engine has 300 uncompressed decoded frames preloaded in memory. This can take a substantial amount of memory, and the combination of having that much data loaded and that many threads running is likely to affect performance. This workflow situation has to be taken into consideration when designing and testing the design. Therefore, unloading scenes and cleaning up the Viz Engine renderer, a function in many Vizrt control client programs, should be used.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Texture

This page contains the following topics and procedures:

- SoftClip Properties
- To Use SoftClip

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

SoftClip Properties

- **Clip file:** Use this function to browse for the clip to be played out.

Note: It is important that all video clips are stored locally on the D: drive of the machine used to play out the graphics. Remember to save the clip locally on the client machines with preview (for example, Viz Trio) and use the same D: drive so that the clip can be seen in preview.

- **Clip length (frames):** Shows the number of frames for the clip.

Copyright © 2026 Vizrt

- **Live Play:** Makes sure that 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**.

Note: There is no frame interpolation done when you change the frame rate from the default specified.

- **Loop Video:** Loops the video automatically when enabled.
- **Use decode thread:** Enables split processing of the clip evenly between the machine's processors. It is recommended to have this `On` (default). On multi-processor machines this gains performance, but threading makes scheduling less predictable and may cause the Viz Engine frame rate to be less stable on a single CPU machine.

Tip: If **Priority** is set to `High`, for best results set **Use Decode Thread** to `off`.

- **Preload frame count:** Works as a buffer to enable smoother playback of the clip. If using Live Play, the Preload frame count can help reducing spike-loads and avoid dropped frames.
- **Frame skipping:** With frame skipping enabled, Viz Engine skips frames if it needs, to try and maintain real-time graphics playback. With frame skipping disabled, the clip plays out in full, as fast as possible but does not try to maintain real-time playback. (Graphics stagger on the output if the renderer drops out of real-time.) When using the video card's ringbuffer functionality the SoftClip frame skipping should be turned off.
- **Force Opaque:** Fills the alpha channel, thus making the video completely opaque. When running videos where the codec leaves the alpha channel blank, the video rendering may become completely transparent.
- **Preferred CPU:** Enables the User to specify a particular processor to handle video play out. The recommended CPU value is based on the specifications of the running machine:
  - If it is a quad core processor the CPU value can be set to `4`, but to avoid errors it is recommended to set the value to `2`.
  - If it is a machine with lower specifications then the recommended value is `1`.
  - The value `-1` enables the machine operating system to select which CPU to use.
- **Priority:** Sets the priority of drawing the video to a custom level in Viz Engine giving it more or less processor priority.
- **Clear before playback:** Clears the image before playback when set to **On**. **Live Play** must be set to **Synchronized** and the **Frame number** to be shown from the clip is smaller than the current frame number +1 (available when **Live Play** is set to Off).
- **Auto scale texture:** Scales the video clip to the size of the underlying rectangle on which the video is placed in the container.

Copyright © 2026 Vizrt

- **Border crop:** Determines how much of the border to crop when using the auto scaling over. Due to bilinear texture interpolation, you might see that color from the underlying texture is bleeding into the video texture area. Use this parameter to crop away the affected border pixels.

To Use SoftClip

1. Add a group container to the Scene Tree.
2. Add the **SoftClip** plug-in.
3. Open the SoftClip editor.
4. Click the **Browse...** button to select and load a clip. Loading a clip makes it visible in the scene. By default, the SoftClip plug-in has the Live Play and Loop video settings enabled, hence, the clip instantly starts playing (and looping) in the render window.
5. Optional: Enable the **Use decode thread** option.
6. Disable the **Live Play** option to enable the **Frame number** setting.
7. Animate the Frame number setting (for example, from frame ten to 30).



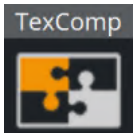
Note: Viz Engine does not interpret between Key Frames, so if you choose to slow down an animation then the motion is not smooth.

See Also

- Image Clip
- MoViz

Copyright © 2026 Vizrt

Texture Component



The Texture Component plug-in selects which component of a texture is used for the texture mapping process.

Note: This plug-in is a legacy plug-in and the functionality is now incorporated into the Texture Editor.

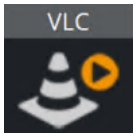
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Texture Component Properties

- **Use Component:** Determines whether to use alpha, color, or both.
  - **Alpha:** Uses only the alpha of the texture. The RGB color of the texture is set to a constant color specified by the **Constant Color** parameter.
  - **Color:** Uses only the RGB color of the texture. The alpha is set to a constant value specified by the **Constant Alpha** parameter.
  - **Alpha+Color:** Uses both the alpha and the RGB color of the texture (default setting).
- **Constant Color:** Sets the constant RGB color of the texture when **Alpha** is selected.
- **Constant Alpha:** Sets the constant alpha value when **Color** is selected.

Copyright © 2026 Vizrt

VLC Plug-in



The VLC plug-in is a media player plug-in that makes use of VLC media player technology. VLC should be placed on a group container for playing the media over a rectangle or on a geometry object, as it is a function plug-in.

The plug-in requires a VLC media player 3.0.11 for 64-bit Windows to be installed on the machine.

Download VLC media player 3.0.11 for 64-bit Windows from http://videolan.org then put *plugins* and *lua* folders in <VizDirectory>. See more information at *<Viz4Directory>\VLC_README.rtf*

Note: This plug-in is located in: Plugins -> Container plug-ins -> Texture

Note: Do not apply a texture to a VLC plug-in container.

VLC Properties

- **Media:** Name of the media to be played. It can be a file supported by VLC player.
- **Loop:** Plays media on repeat when set to `On`.
- **Audio Enabled:** Plays sound from the selected media during playback when set to `On`. When set to `Off`, the sound is muted.
- **Volume:** Controls the volume level if Audio is enabled.
- **Set Clip Time:** Seeks the playing or paused clip to specified time (in milliseconds). It is not possible on a clip with some video codec.
- **Play on Load:** Plays the clip automatically when loading the scene when set to `On`. When set to `Off`, the clip does not play until clicking the **Play** button.
- **Use Proxy:** Determines whether to use a proxy. When set to `On`, additional parameters are enabled:
  - **Proxy Host:** The name of the proxy machine.
  - **Proxy Port:** Proxy port, as defined on the proxy machine.
  - **Proxy User:** User name with permission to access the network via the proxy machine.
  - **Proxy Password:** Password for the proxy user.
- **Play:** Plays the media from the beginning.
- **Stop:** Stops playing the media. The current frame is displayed. It is not possible to resume playing the media after has been stopped.
- **Pause:** Pauses the playing clip at the current frame.
- **Continue:** Continues playing the media from the current frame after it was paused.

Logging: In previous versions, it was possible to enable logging per container separately. This is now depending on the config setting **VerboseConsole**; if it is set to verbose it also enables logging for the VLC plug-in. If enabled, it writes log information into a file named *vlc.log*, residing in the *%ProgramData%\vizrt\VizEngine* folder.

Info: The VLC player is not a replacement for the Matrox clip player!

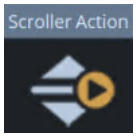
Copyright © 2026 Vizrt

2.4.16 Ticker Container Plug-ins

The following Container plug-ins are located in the Ticker folder:

- Scroller Action

Scroller Action



The Scroller Action plug-in makes it possible to design scrolling tickers that trigger actions when scroller items are played out. Actions can be triggered both before and after reading the next item from the input source, as well as delaying the reading of the next item a certain number of frames.

A scrolling carousel scene must have a Scroller plug-in attached to a container and be configured to receive items from an external source. In most cases this external source is Viz Ticker's Ticker Service running on the same machine. During scene design, it is useful to use a debugging syntax to specify a fake source that refers to graphics templates located directly in the scene.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Ticker

This page contains the following topics and procedures:

- Scroller Action Properties
- Controlling the Progression
- To Control the Progression

Scroller Action Properties

- **Delay (Frames):** Adds a delay after the current item is fully inside the scroller, so that insertion of the next item is postponed the given number of frames.
- **On Inside:** Defines if the item should trigger anything after it fully enters the scroller.
  - **Action**
    - **None:** Does nothing.
    - **Animate:** Allows a director to be triggered and defines the start and stop point of the director animation.
  - **Director:** Defines the director to run when On Inside is triggered.
  - **From Tag:** Defines where to start the director animation.
  - **To Tag:** Defines where to stop the director animation.
- **After Delay:** Defines if the item should trigger anything after the scroller has finished delaying the next item.
  - **Action**
    - **None:** Does nothing.
    - **Animate:** Allows a director to be triggered and defines the start and stop point of the director animation.
  - **Director:** Defines the director to run when After Delay is triggered.
  - **From Tag:** Defines where to start the director animation.
  - **To Tag:** Defines where to stop the director animation.

    The source syntax (separator) (text) creates a scroller showing an endless stream of items alternating between a separator and a text graphics design. The scroller searches through the scene tree for containers named separator and text, use the graphics template found there, and then copies them to create the item instances that are being scrolled.

Copyright © 2026 Vizrt

Controlling the Progression

A typical example of when the Scroller Action plug-in is necessary, is when text messages (SMS) should scroll into the screen, animate to a halt, wait a few seconds, and then restore the scroll speed whenever new text messages are available.

This paragraph provides guidance on how to create an upwards scroll to show text messages, where the scroller should pause for a while after playing out each message to the screen. The scroller should not immediately follow the text design with the next separator design. Instead, the text design should enter the screen fully, and then the scroller should smoothly slow down to a halt. The scroller should wait a given number of frames before slowly starting the scroller again and pushing the next separator item onto the screen. If an external source is used, the scroller should not start again until there is a new item available from the source.

To Control the Progression

1. Create a director named `scroller`. This director animates the scroller speed.
2. On this director, create a stop point, and call it `stop`.
3. Set the stop scroll speed to `0`.
4. Create another stop point, and call it `play`.
5. Set the play scroll speed to a suitable speed. The actual names that are used here are not important, but they must match the names specified in step 8.
6. Add the Scroller Action plug-in to the text template top node (the container holding the Control Object plug-in).
7. In the Scroller Action plug-in editor, define the **Delay** frame count. This setting adds a delay after the current item, so that the insertion of the next item is postponed the given number of frames. The delay counter starts to count from the time the previous item has fully entered the screen, which is a few frames after it would normally try to insert a new item. When delay is set to `0`, the scroller performs as normal, and does not wait until the previous item has fully entered the scroller before inserting a new item. The *delay* parameter specifies the minimum amount of time that must pass between pushing out the current item, which is based on the template graphics with the *Scroller Action* plug-in, and the next item. The actual amount of time might be greater if there are no new items available after the *delay* frame count has expired. Setting this parameter does not affect the speed of the scroller, it only delays the inserting of the next item, in effect increasing the resulting gap between the previous item and the next. The default value of the *delay* frame count parameter is `0`, which means that the scroller attempts to insert the next item immediately after the previous item.
8. In the Scroller Action plug-in editor, click the *On Inside: action* **Animate** button. When *animate* is selected, three additional on *inside* related parameters become available:
   - **On Inside:** director (default value is scroller)
   - **On Inside:** from tag (default value is play)
   - **On Inside:** to tag (default value is stop)

   The parameters should remain at the default values. When the current item has fully entered the screen, the scroller starts to animate the *scroller* director from a *play* tag to a *stop* tag. The rules of the *GOTO_TRIO* director command defines this behavior.
9. In the Scroller Action plug-in editor, click the *After Delay: action* **Animate** button. When *animate* is selected, three additional *after delay* related parameters become available:
   - **After Delay: director** (default value is `scroller`)
   - **After Delay: from tag** (default value is `stop`)

Copyright © 2026 Vizrt

- **After delay: to tag** (default value is `play`)
  The parameters should remain at the default values. When the *delay* frame count expires, the scroller starts to animate the *scroller* director from a *stop* tag to a *play* tag. The rules of the *GOTO_TRIO* director command defines this behavior.

See Also

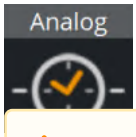
- **Transition Logic** section of the Viz Artist User Guide.

2.4.17 Time Container Plug-ins

The following Container plug-ins are located in the Time folder:

- Analog Watch
- Clock Rotation

Analog Watch



The Analog Watch plug-in lets you create a real-time animated clock of objects.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Time

This page contains the following topics and procedures:

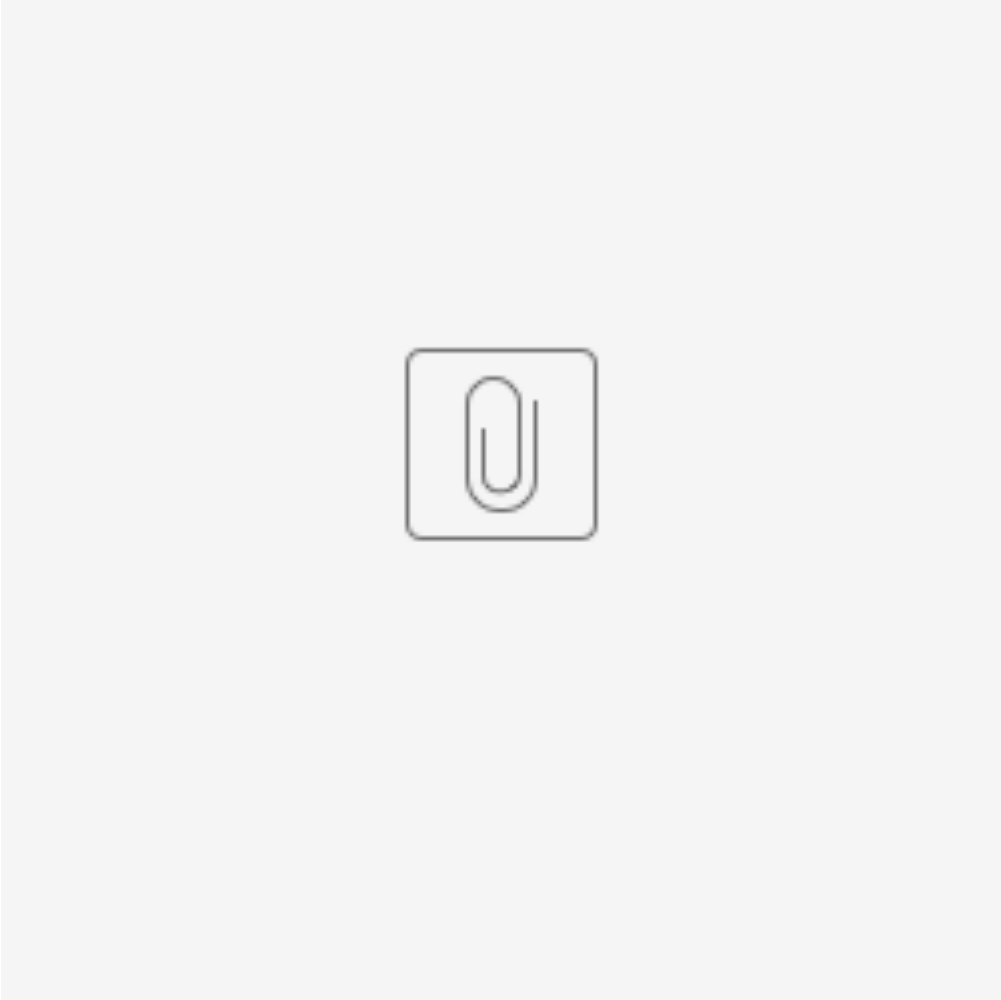
- Analog Watch Properties
- To Create an Analog Watch

Analog Watch Properties

- **Mode:** Sets the mode. Available options are Hour24, Hour12, Minute, Second and Off.
- **Smooth:** Ensures the clock hands rotate smoothly. Check this button to enable smooth rotation. Hour hands always rotate smoothly, so this switch is inactive if the plug-in is set to Hour24 or Hour12 mode.
- **Offset:** Changes the rotation offset. 1200 usually is equal to a rotation of 0 degrees.
    - 1 hour in Hour24 mode: 15 degrees.
    - 1 hour in Hour12 mode: 30 degrees.
    - 1 minute in Minute mode: 6 degrees.
    - 1 second in Second mode: 6 degrees.

Copyright © 2026 Vizrt

To Create an Analog Watch



1. Add a Circle geometry to the scene tree, add material to it, and name it **Clock**.
2. Open the Circle editor and set scaling to `2.0`.
3. Add three Arrow geometries as Sub-Containers of the **Clock** container, add material to them, and name the first **Hours, the** second **Minutes**, and the third as **Seconds**.
4. Open the Arrow editor for Hours and set the following properties:
    - Style1 to `Flat`.
    - Width and Arrow Width to `4.0`.
    - Percent to `30.0`.
5. Open the Arrow editor for Minutes and set the following properties:
    - Style1 to `Flat`.
    - Width and Arrow Width to `3.0`.
    - Percent to `40.0`.
6. Open the Arrow editor for Seconds and set the following properties:

Copyright © 2026 Vizrt

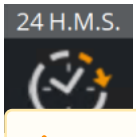
- Style1 to `Flat`.
- Width and Arrow Width to `2.0`.
- Percent to `50.0`.

7. Add the Analog Watch plug-in to the Hours, Minutes and Seconds containers.
8. Open the Analog Watch editor for the Hours container and set it to Hour24 or Hour12.
9. Open the Analog Watch editor for the Minutes container and set it to Minute.
10. Open the Analog Watch editor for the Seconds container and set it to Second.

See Also

- Clock Rotation
- System Time

Clock Rotation



Animates any object as a rotating clock.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

Clock Rotation Properties

- **Clock Unit:** Sets the units for the rotation of the object. Available options are Seconds, Minutes and Hours.
- **Rotation axis:** Sets the axis for the rotation of the object. Available options are X, Y and Z.
- **Motion type:** Sets the motion type for the rotated object. Digital shows a ticking motion, and Chrono a smooth motion.
- **Reverse:** Sets the direction of the rotation. When set to `Off`, the rotation is clockwise, and when set to `On`, the rotation is counter-clockwise.

See Also

- Analog Watch
- System Time

Copyright © 2026 Vizrt

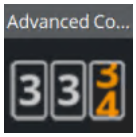
2.4.18 Tool Container Plug-ins

The following Container plug-ins are located in the Tools folder:

- Advanced Counter
- Align Text by Character
- Autofollow
- Autorotate
- Bounding Actions
- Cloner
- Colorize
- Counter
- DVE Follow
- Flexbox
- Heartbeat
- Hide On Empty
- Image Link
- Jack
- Level of Detail
- Magnify
- Match It
- Max Size
- Max Size Lines
- Object Zoom
- Omo
- Parliament
- PathFinder
- Rotation Order
- Slide Show
- System Time
- Temo
- Text Auto Scale
- Text Link
- Text Parameters
- TextBG
- Texture Fit
- TransitionLayers
- VCF Parameter

Copyright © 2026 Vizrt

Advanced Counter



The Advanced Counter plug-in allows you to easily create an animated counting sequence.

Advanced Counter shows values with decimals and it has the possibility to create a user defined format mask. Furthermore, it can have prefixed values.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

This page contains the following topics and procedures:

- Advanced Counter Properties
- To Create an Advanced Counter

Advanced Counter Properties

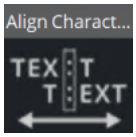
- **Use Integers:** Enables the use of integers only.
- **Value (Dec.), Value (Int.):** Sets the current value of the counter. Animate this value to create the counting sequence. Any number in the valid integer/float range on the computer system is allowed (for 32-bit integers: - 2147483648 to +2147483647, and for single precision floats (32-bit): -3.40282347E+38 to +3.40282347E+38).
- **Leading blanks:** Selects what the counter does with leading blanks. They can either be **Cut** away or put in as **Blank**, asterisk, hash, dot or zeros.
- **Force prefix:** Enables both positive and negative values to have a prefix, not only negative values as is the default (+/-).
- **Delete unnecessary commas:** Removes superfluous commas when a specified format mask has more commas than the number needs. If for instance the mask `$###,###,###,###.##` is defined, and the number `4120.37` is entered, it is output as **$,,4,120.37**. If by enabling this option, the number gets a correct format: **$4,120.37**.
- **Decimal separator:** Switches the decimal separator between point and comma.
- **Format mask:** Allows format mask definition. Each hash symbolizes an item of the total number. To alter, add or remove hashes. You can also add constant values, for example DM, NOK etc.
- **Initialize:** Starts the counter.

To Create an Advanced Counter

1. Add a group container to the scene tree.
2. Add a font and material to the group container.
3. Add Advanced Counter to the group container.
4. Open the Advanced Counter editor and animate the parameters.

Copyright © 2026 Vizrt

Align Text by Character



The Align Text by Character plug-in aligns the text content by centering the text in each line to the specified token.

If the token is found multiple times on the same line, only the first one is used.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Note: This plug-in only works with the Text plug-in in the Viz Engine Render Pipeline.

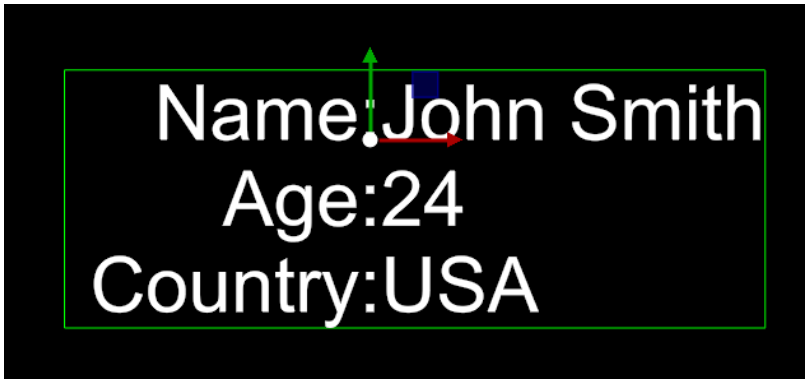
Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Properties

- **Token:** The character to be used as the token to center the text.

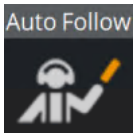
Example

Text aligned on the colon (**:**) character:



Copyright © 2026 Vizrt

Autofollow



The Autofollow plug-in connects one object with another. This object is from then on dependent on the connected plug-in.

Depending on the axis of your Object to follow, the bounding-box grows in the reverse direction.



If you want to follow for example a text object, which writes from left to right, you have to set the axis of your text object to the left side, because the bounding box grows to the right side. You can select for the following object either one axis (X,Y,Z) or two (XY, XZ, YZ).



Drag Autofollow over the object container which should be dependent. It automatically follows the previous object in the scene tree. You may also set this to be the first object or select one specific object by adding it to the object placeholder exposed when selecting the Other option.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Autofollow Properties

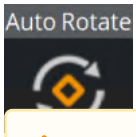
- **X Previous Center:** Defines the alignment of your related object.
- **Default Distance (Min):** Allows you to set a distance from the object to the related one. This is a relative value that is dependent on the size of the related container.
- **Progress (%):** Allows you to animate the related object. In this case you can set a default distance value and a maximum distance. What you need to do is just set Key Frames on the progress. At the beginning of the animation you can for example have 100%, and at the end 0%. The object runs from the maximum to the default distance value.
- **Maximum Distance:** Allows you to set the maximum distance from the object to the related one. This is an absolute value, which is independent of the size of the related container.

Copyright © 2026 Vizrt

- **Reference Container:** With these buttons (**PREV**, **FIRST**, **OTHER**) you can change the related object. The default preference is 'Previous', where the previous container in the scene-tree is selected. If you choose **Other**, you have to drag your wished reference-container onto the empty button, which shows when you select **Other**.
- **Direction:** Allows you to choose in which axes there should be a dependency on. Options are: *X*, *Y*, *Z*, *XY*, *XZ* or *YZ*.
- **Follow Negative:** If you select this feature, all numerical values are negated. For example, when you change *Default Distance (Min)* to a positive value your object is positioned in negative direction of the selected axis or axes.
- **Ignore hidden containers:** If the reference container is hidden, search for the next visible container depending on the reference container setting (only if set to **Previous** or **First**).
- **Ignore default distance on hide:** Allows you to hide the reference container if you do not want the default distance to be considered. In this case, it is null. If you make the reference container visible, the default distance shows again.
- **X Local Center:** Allows you to select your alignment for the local center in X-direction.

Copyright © 2026 Vizrt

Autorotate



The Autorotate plug-in rotates a container continuously around one of its axes.

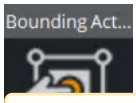
Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Autorotate Properties

- **Axis:** Selects the rotation axis either **X**, **Y** or **Z**.
- **Direction:** Selects the direction of the rotation, either **Right** or **Left**.
- **Velocity:** Sets the speed of the rotation. The unit is degrees per frame.
- **Ease In (frames):** Sets the number of frames the object uses to accelerate smoothly from no motion to the speed set in **Velocity**.
- **Ease Out (frames):** Sets the number of frames the object uses to retard from the rotation speed to a stop.
- **Starting Angle:** Sets the angle from the objects current position, from which the object is to start from.
- **Ping Pong:** Enables a ping pong motion where the object rotates forward and backward between two angles on the axis.
- **Ping Pong Angle 1:** Sets angle 1 for *Ping Pong Mode*.
- **Ping Pong Angle 2:** Sets angle 2 for *Ping Pong Mode*.
- **Ease in:** Initializes the rotation from the starting angle selected and with the **Ease in** value selected.
- **Ease out:** Stops the rotation. The stop motion is influenced by the **Ease out** parameter.
- **Continue:** Starts the rotation from the point where it was halted. The value of **Ease in** is disregarded.
- **Halt:** Stops the rotation without regard to **Ease out**.

Copyright © 2026 Vizrt

Bounding Actions



The Bounding Actions plug-in enables you to run Viz Artist/Engine actions depending on the size of the bounding box.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

This page contains the following topics and procedures:

- Bounding Actions Properties
  - Syntax
- To Use Bounding Actions

Bounding Actions Properties

- **Text box:** Executes the command every time the bounding box changes.
- **Scale X, Y and Z:** Applies a scaling to the values %dx %dy %dz.
- **Offset X, Y and Z:** Adds an offset to the values %dx %dy %dz.

Syntax

- **%dx:** Substitutes the width of the bounding box.
- **%dy:** Substitutes the height of the bounding box.
- **%dz:** Substitutes the depth of the bounding box.
- **%xbool:** 0 if dx <= 0 else 1.
- **%ybool:** 0 if dy <= 0 else 1.
- **%zbool:** 0 if dz <= 0 else 1.
- **%container:** Path of the container the bounding action is applied to.
- **%container[path]:** Relative path of the container the bounding action is applied to.

Example: `THIS_SCENE*TREE*#297*TRANSFORMATION*POSITION SET %dx %dy %dz`

Example: `%container[$textnode1$textnode2]*GEOM*TEXT SET %xbool`

To Use Bounding Actions

1. Drag the Bounding Actions onto a container.
2. Insert a command in the Text box (see examples above). Now every time the bounding box changes (a Sub-Container is moved outside the current bounding box or Bounding Actions is used to modify it), this command is executed.

Copyright © 2026 Vizrt

Cloner



Duplicates the first child container and arranges the copies in a **matrix grid**, **ellipse arc**, or on a **sphere** surface.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Performance tips:

- This plug-in is designed to quickly create and arrange a certain amount of containers. If you are running on the Viz Engine Render Pipeline and using the same geometry, please consider using the Instancing plug-ins.
- Changing only spatial parameters (spacing, radius, angles) triggers a *reposition-only* pass. The performance expensive clone/tree operation only runs when clone count, rows, or columns change.

This page contains the following topics and procedures:

- Overview
- Quick Start
- Parameter Reference
  - General
  - Matrix Mode
  - Ellipse Mode
  - Sphere Mode
  - Actions
  - How It Works
  - Usage Examples
    - Example 1: 5×3 Grid of Cubes
    - Example 2: Circular Gauge (270° Arc)
    - Example 3: Globe Markers
  - Technical Notes

Overview

The **Cloner** plug-in takes the *first child* of the container it is applied to and creates up to 3,000 copies, positioning them in one of three spatial arrangements. It is commonly used to build data-driven bar charts, grids, circular gauges, audience seating, and particle-style scatter layouts.

- **Matrix:** Rectangular grid on a selectable plane (XY / XZ / YZ).
- **Ellipse:** Arc or full ellipse with independent semi-axes and angle range.
- **Sphere:** Surface distribution over azimuth × pitch ranges.

Quick Start

1. Create a group container and add the Cloner plug-in to it (***Plugins > Container > Tools > Cloner***).

Copyright © 2026 Vizrt

2. Add one child under the group. This is the template to be cloned (for example, a cube, a rectangle, or an entire sub-tree).
3. Set **Clones Count**, **Rows**, and **Cols**. The effective count is `min(Count, Rows × Cols)`.
4. Choose a **Shape** (Matrix, Ellipse, or Sphere) and adjust the corresponding layout parameters.
5. For Matrix mode, set **Delta X / Y / Z** to control spacing, and choose whether the delta represents per-step distance or total bounding box.

Parameter Reference

General

Parameter	Default	Range	Description
Clones Count	100	1 – 3,000	Total number of clones to create.
Clones Rows	1	1 – 3,000	Number of rows in the layout grid.
Clones Cols	1	1 – 3,000	Number of columns in the layout grid.
Shape	MATRIX	MATRIX ELIPSE SPHERE	Spatial arrangement mode.
Rename SubTrees	Off	—	When enabled, renames entire sub-trees of each clone (not just the direct child).
Remove Excess Containers	Off	—	Automatically deletes clones that exceed the current count. When off, excess containers are hidden instead.

Matrix Mode

Parameter	Default	Range	Description
Plane	XY	XY · XZ · YZ	Which two axes are used for column/ row layout.
Delta X	0	±10,000,000	Spacing along X axis.
Delta Y	0	±10,000,000	Spacing along Y axis.
Delta Z	0	±10,000,000	Spacing along Z axis.

Copyright © 2026 Vizrt

Parameter	Default	Range	Description
Delta represents	Step	Step Bounding Box	**Step**: delta is the distance between each clone. **Bounding Box**: delta is the total extent, divided evenly among clones.

Ellipse Mode

Parameter	Default	Range	Description
R1	0	0 – 100,000,000	Semi-axis one (horizontal radius).
R2	0	0 – 100,000,000	Semi-axis two (vertical radius).
Min Angle	0	0 – 360	Start angle of the arc (degrees).
Max Angle	0	0 – 360	End angle of the arc (degrees).
Offset Angle	0	0 – 360	Additional rotation applied to each clone so it faces outward along the arc.

Sphere Mode

Parameter	Default	Range	Description
Radius	100	0 – 100,000,000	Distance from center to each clone.
Min Azimuth	0	0 – 360	Longitude start (degrees).
Max Azimuth	360	0 – 360	Longitude end (degrees).
Min Pitch	−90	−90 – 90	Latitude start (degrees, south pole = −90).
Max Pitch	90	−90 – 90	Latitude end (degrees, north pole = 90).

Copyright © 2026 Vizrt

Actions

Action	Description
Remove All	Deletes all cloned children, leaving only the original template container.
Remove Only Excess	Deletes clones that exceed the current *Clones Count*.

How It Works

1. **Cloning:** The plug-in iterates `Rows × Cols` (capped at *Clones Count*). For each slot where a child container does not yet exist, it copies the previous sibling. Each clone is renamed sequentially. If *Rename SubTrees* is on, the entire sub-tree of each clone is also renamed.
2. **Positioning:** Based on the selected *Shape*, each clone's position and rotation are calculated:
   - **Matrix**: Steps through rows × cols applying X/Y/Z offsets. In *Bounding Box* mode, deltas are divided by the row/col count.
   - **Ellipse**: Distributes clones evenly on an elliptical arc between Min Angle and Max Angle. Multi-row ellipses grow radii proportionally and shift in depth.
   - **Sphere**: Uses azimuth (longitude) across columns and pitch (latitude) across rows to place clones on a spherical shell. Each clone is rotated to face outward.
3. **Change detection:** Each frame, current parameter values are compared against cached values. Count/row/col changes trigger a full rebuild; spatial-only changes trigger a cheap reposition pass.
4. **Excess handling:** Containers beyond the current count are either hidden (`active = false`) or deleted, depending on the *Remove Excess Containers* setting.

Usage Examples

Example 1: 5×3 Grid of Cubes

```
Container "CubeGrid"
  └── Cloner  (Shape: MATRIX, Count: 15, Rows: 3, Cols: 5, Delta X: 120, Delta Y:
120)
     └── Cube  (geometry: Cube, 100×100×100)
```

Creates a five column, three row grid of cubes spaced 120 units apart on the XY plane.

Example 2: Circular Gauge (270° Arc)

```
Container "Gauge"
  └── Cloner  (Shape: ELIPSE, Count: 27, Rows: 1, Cols: 27,
              R1: 200, R2: 200, Min Angle: 135, Max Angle: 405)
     └── Tick  (geometry: Rectangle, 4×20)
```

Places 27 tick marks evenly along a 270° arc, each rotated to face outward from center.

Example 3: Globe Markers

Copyright © 2026 Vizrt

```
Container "Globe"
  └── Cloner  (Shape: SPHERE, Count: 50, Rows: 5, Cols: 10,
              Radius: 300, Min Azimuth: 0, Max Azimuth: 360,
              Min Pitch: -60, Max Pitch: 60)
     └── Dot   (geometry: Sphere, scale: 0.1)
```

Distributes 50 markers across a sphere shell, covering ±60° latitude in a 10×5 grid.

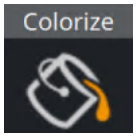
⚠ **Bounding Box vs Step mode:** In *Step* mode, Delta values are the distance between adjacent clones. In *Bounding Box* mode, Delta values define the **total extent** of the grid (the spacing is computed automatically). Switching modes without adjusting values can cause clones to overlap or spread unexpectedly.

Technical Notes

- The effective clone count is `min(Clones Count, Rows × Cols)`.
- The first child is always the **template** (it is never deleted by cleanup actions).
- GUI parameters are dynamically shown/hidden based on the selected Shape (matrix parameters are hidden in sphere mode and vice versa).
- Clone naming is sequential: `Clone_1`, `Clone_2`, etc.
- For large clone counts (1000+), enable *Remove Excess Containers* to prevent hidden containers from consuming memory.

Copyright © 2026 Vizrt

Colorize



The Colorize plug-in allows you to transform the colors of containers within a container group. If you for instance have 15 Sub-Containers in a group, the colorize plug-in assign numbers to each container by the order they have in the group. You can then define for each Sub-Container which color it should have to start with and which color it should change to.

Colorize can define up to ten colors. Each color is able to influence more than one container. The Num Color1-10 parameters allow you to link multiple containers to one color in the colorize plug-in. Each of the Sub-Containers must have its own material for Colorize to work.

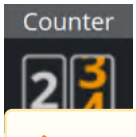
> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

Colorize Properties

- **Colorize:** Runs the color transformation. One number corresponds to one container.
- **Animation Start:** Allows you to select the starting point for the color change.
- **Default Color:** Sets the color for containers in the group which are not selected for color transformation. To set it, use the color editor below or drag a material from the Server Panel onto the small square.
- **Num Color1 A to 10 A:** Assigns the starting colors to containers, from which the color change starts. The value sets the number of containers to be linked with the corresponding color in the Color1 A: 10 A parameters. The function starts counting from the first undefined container. This means that, if color 1 A has the value 2, color 2 A has the value 3 and 3 A has the value 1, the two first containers get the properties of color 1 A, container 3: 5 get the properties of color 2 A and container 6 gets the properties for color 3 A.
- **Color1 A to 10 A:** Sets the colors for 1 A: 10 A. You can set each color using two methods:
  - Either select the color by clicking the color icon and then set the color properties at the color editor at the bottom, or
  - Drag a material from the Server Panel onto the color icon of the color you want to set.
- **Num Color1 B to 10 B:** Assigns the ending color to containers, to which the color change ends. The value sets the number of containers to be linked with the corresponding color in the Color 1 B: 10 B parameters. The function starts counting from the first undefined container. This means that, if color 1 B has the value 2, color 2 B has the value 3 and 3 B has the value 1, the two first containers get the properties of color 1 A, container 3: 5 get the properties of color 2 B and container 6 gets the properties for color 3 B.
- **Color1 B to 10 B:** Sets the color for 1 B: 10 B. You can set each color using two methods:
  - Either select the color by clicking the color icon and then set the color properties at the color editor at the bottom, or
  - Drag a material from the Server Panel onto the color icon of the color you want to set.
- **Rebuild:** After having made changes to colors and assigning containers, click rebuild to apply the changes.

Copyright © 2026 Vizrt

Counter



The Counter plug-in creates a count up or count down at any given range and at any given speed. It shows integral numbers.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

This page contains the following topics and procedures:

- Counter Properties
- To Use the Counter

Counter Properties

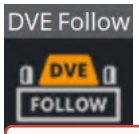
- **Number:** Sets the current number of the counter. Animate this value to create the wanted count-up or count-down. Any number in the valid integer range on the computer system is allowed (-2147483648 to +2147483647 for 32-bit integers).
- **Initialize:** Starts the operation.

To Use the Counter

Add Counter to a container with a font, and click **Initialize**.

Copyright © 2026 Vizrt

DVE Follow



The DVE Follow plug-in connects a video input channel or a clip channel of the containing Scene with the bounding box of a Container. When connected the channel, if set to DVE, follows the bounding box of the Container. The plug-in does not actively set the channel to DVE.

IMPORTANT! Be aware of the fact that the DVE animation is done on the video board while the animation of the ruling container is done together with all other graphics on the GPU. The output delays of the graphics and the DVE animation are different and this can lead to notable offsets between graphics and DVE (especially notable with fast animations).

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

DVE Follow Properties

- **Channel Type:** Type of channel to manipulate:
  - Video Channel
  - Clip Channel
- **Video/Clip Channel Number:** The plug-in supports a maximum of eight video channels and 16 clip channels.
- **Layer:** Selects the layer which contains the Scene in which the DVE is located.
- **Video Width:** Sets the width of the containers bounding box.
- **Video Height:** Sets the height of the containers bounding box.
- **Alpha:** Sets the alpha value of the DVE channel.
- **Priority:** Sets the priority value of the DVE channel.
- **Crop:** Sets the left, right, top and bottom crop value of the DVE channel.

Copyright © 2026 Vizrt

Flexbox



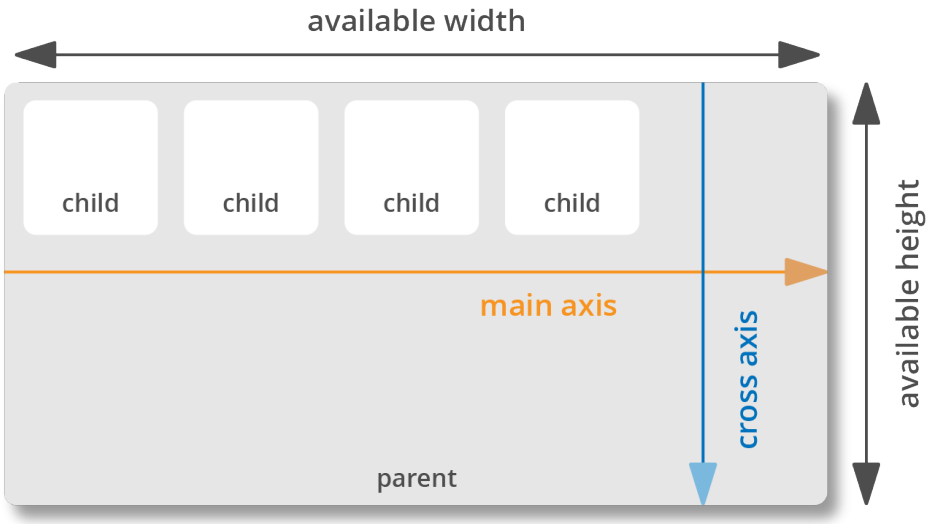
The Flexbox plug-in takes advantage of the Box Transformation and allows to build graphics for adaptive storytelling.

It is based on the flex implementation of CSS and web development.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Information: The Flexbox plug-in requires a valid license and is available for the Viz Engine Render Pipeline only!

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.



Flexbox Properties

Flex

- **Flex Wrap:** Scales down children to fit into a box. By changing the wrapping mode, children are automatically aligned based on the setting:
  - **No Wrap:** Boxes are scaled down until they reach their minimum width.
  - **Wrap** and **Wrap Reverse:** Boxes are scaled down until they reach their minimum size and are then wrapped to fill the space.
- **Flex Direction:** Defines the main axis and defines the direction of all children placed underneath the flexbox container.

Copyright © 2026 Vizrt

- **Direction:** Aligns starting from the **left** or from the **right** side. By default, it **inherits** the setting of the parent flex container.
- **Flex Basis:** 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.
- **Flex Grow:** Defines if and how strongly a container can grow 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.
- **Flex Shrink:** Defines if and how strongly a container can shrink to fit into a parent container. The minimum and maximum width/height values are taken into account. 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).

3D Content

- **Update Box Transformation:** Overrides values of Box Transformation in each field. This flag does not affect the width/height and x/y values of Box Transformation being overridden by FlexBox.
- **Box Transformation Z:** Controls the z value of box information in Box Transformation.
- **Box Transformation Depth:** Controls depth value box information in Box Transformation.

Dimensions

- **Aspect Ratio:** Defines how a flexbox is being scaled. If it is on *auto*, the width and height taken as 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.
- **Sizes:** Sets the width and height of a container.
- **Minimum:** Sets the upper size constraints of a container. These properties have higher priority than all other properties and are always respected.
- **Maximum:** Sets the lower size constraints of a container. These properties have higher priority than all other properties and are always respected.

⚠ **Note:** 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 (This dimension can be influenced by other properties.)
  - **Percentage:** Same as pixels, but based on percentage values.

- **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.
- **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.
- **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.

Copyright © 2026 Vizrt

Position

- **Tracking Container:** Specifies a reference container and Flexbox follows this position.
- **Mode:** 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:** Takes the calculated position and adds the given offset. Offsets can be set as *auto* (calculated automatically when using certain content alignments) or as value in *pixels* or *percentage*.
    - **Absolute:** 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.
- **Offsets:** Shifts the whole container and its content based on the calculated position by the given values.

Content

- **Justify Content:** Describes how child containers align along the Main Axis. This can be used to automatically center items, for example. The following settings are available:
    - **Flex Start:** 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.
- **Align Items:** Describes how child containers align along the cross axis.
    - **Auto:** Uses the align item of the parent container.
    - **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:** Stretches items to fill the cross axis. The cross axis setting 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.

Align Items

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:

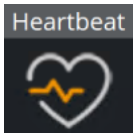
- **Align Self:** Overwrites the Align Items property given by the parent container to is own.
- **Align Content:** Defines how containers are aligned. If the wrap mode forces elements to be distributed in more than on row or column, it has no effect if no wrapping happens.
    - **Auto:** Uses the align 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:** Aligns child containers to the end of the cross axis.

Copyright © 2026 Vizrt

- **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

Heartbeat



The Heartbeat plug-in creates a heart beat animation. Includes animation of:

- Size, along different axes, axis-pairs or all axes
- Rotation, around three main axes
- Alpha
- Color

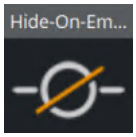
> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

Heartbeat Properties

- **Animation Length:** Sets the length of the animation. Parameter Name: *animLength*.
- **Swing:** Enables the object to have a swinging action (back and forth). Parameter Name: *swing*.
- **Pause:** Enables the object to pause before repeating the action. Parameter Name: *pause*.
    - **Pause Length:** Sets the length of the pause. Parameter Name: *pauseLength*.
- **Change Size:** Enables the change size parameters for changing the vector(s), initial and final weight of the object's animation. Parameter Name: *changeSize*.
    - **X, Y, Z, XY, XZ, YZ, XYZ:** Sets animation vector(s). Parameter Name: *changeVector*.
    - **Initial Weight:** Sets the initial weight of the object. The higher the value, the slower the animation at first. Parameter Name: *initialWeight*.
    - **Final Weight:** Sets the final weight of the object. See also Initial Weight. Parameter Name: *finalWeight*.
- **Change Rotation:** Enables the rotation parameters changing the rotation axes, initial and final angles. Parameter Name: *changeRotation*.
    - **X, Y, Z:** Rotates the object on the X, Y or Z axis. Parameter Name: *rotAxes*.
    - **Initial Angle:** Sets the initial angle of the object. Parameter Name: *initialAngle*.
    - **Final Angle:** Sets the final angle of the object. Parameter Name: *finalAngle*.
- **Change Alpha:** Enables the alpha parameters changing the initial and final alpha values. Parameter Name: *changeAlpha*.
    - **Initial Alpha:** Sets the initial alpha value. Parameter Name: *initialAlpha*.
    - **Final Alpha:** Sets the final alpha value. Parameter Name: *finalAlpha*.
- **Change Color:** Enables the color parameters changing the color of the object. Parameter Name: *changeColor*.
    - **Source Color:** Sets the source color. Parameter Name: *sourceColor*.
    - **Target Color:** Sets the target color. Parameter Name: *targetColor*.

Copyright © 2026 Vizrt

Hide On Empty



The Hide On Empty plug-in hides its container when a target text geometry matches a given condition.

The condition may be an empty text, or text displaying a zero value, such as `0` or `0.0`, depending on the plug-in parameters.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

Hide On Empty Properties

- **Text container:** Specifies the container of text of which to check the content.
  - **Self:** The text geometry is in the same container as the plug-in is applied to.
  - **First child:** The text geometry is in the first child container of the container the plug-in is applied to.
  - **Other:** Enables the **Other text container** drop area.
- **Other text container:** Drag the container containing the text geometry to set the target. Enable by selecting **Other** as *Text container*.
- **Treat as numerical:** Hides the target container if the specified text is `0` or `0.0`, in addition to empty text.
- **Invert visibility:** Displays the container when the condition is satisfied instead of hiding the target container when the condition matches.
- **Update Geometry:** Updates the visibility state of the container holding the plug-in after changes, removal, or assignment of target text geometry, during the design process.

> **Note:** You cannot use the control plug-in Control Hide on Empty in conjunction with Control Text using the same field identifier. In such a case, use the **Hide On Empty** property in the TextBG plug-in.

Copyright © 2026 Vizrt

Image Link



The Image Link plug-in copies the image/texture from the source container to up to ten linked containers.

The image you load in the source container is repeated in all the linked containers.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

This page contains the following topics and procedures:

- Image Link Properties
- To Link an Image to Multiple Containers

Image Link Properties

- **Container 1: 10:** Drop zones for the containers you want to link.
- **Initialize:** Initializes the plug-in.
- **Do it now:** Performs the operation immediately.

To Link an Image to Multiple Containers

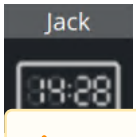
1. Add a group container to the scene tree.
2. Add Image Link and the source image to the group container.
3. Add a number of containers to the scene tree with a dummy image on each.
4. Open the Image Link editor.
5. Drag and drop the containers with the dummy images onto the Container 1-n drop zones.
6. Click **Initialize** to apply change.

See Also

- Text Link

Copyright © 2026 Vizrt

Jack



The Jack plug-in shows a digital clock and date.

Jack is highly customizable allowing you to set your own time and date formats and correction values.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Jack Properties

Time mode has the following properties:

- **Hour format:** Sets how to show hours (either off, on or with a leading zero if the value is less than ten).
- **Min format:** Sets how to show minutes (see Hour format).
- **Sec format:** Sets how to show seconds (see Hour format).
- **Hour/Min separator:** Sets character separation for hours and minutes.
- **Blink mode:** Blinks the separator between hours and minutes when set to `On`. This means every odd second it turns off.
- **Min/Sec separator:** Sets a separator for minutes and seconds.

Note: Although a format is switched off, the separator is still shown.

- **am/pm or 24 hours:** Switches between the am/pm and 24 hour format.
- **Time/meridian separator:** Sets a separator for the am/pm hour format.
- **am/pm format:** Sets the case of am/pm to upper or lower.
- **Hour correction:** Enables you to correct the hour value of the system time. Useful to show the local time of a location in a different time zone.
- **Min correction:** Sets correction value for minutes (see Hour correction).
- **Sec correction:** Sets correction value for seconds (see Hour correction).

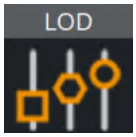
Date mode has the following properties:

- **Year format:** Sets how to show the year format (off, two digits or four digits, please keep in mind that in the year 10000 this is three digits or five digits, due to the algorithm used).
- **Month format:** Sets how the month is shown, either off, on or with a leading zero from January to September.
- **Day format:** Sets how the day is shown (see Month format).
- **First separator:** Sets the separator character for the first two items of the date. If the dot value is selected, an additional blank character is inserted automatically.
- **Second separator:** Sets the separator character for the last two items of the date.
- **Order:** Sets the order of the items of the date.
- **Year correction:** Sets a correction value for the year.
- **Month correction:** Sets a correction value for the month.
- **Day correction:** Sets a correction value for the day.

Note: The order when using multiple date corrections is Day > Month > Year.

Copyright © 2026 Vizrt

Level of Detail



The Level Of Detail (LOD) plug-in is a mechanism for controlling the level of complexity of a 3D object, depending on camera position and object size. As an object becomes smaller on the screen, the objects polygons become smaller. At a certain size, those polygons are small enough to be replaced by a lower number of larger polygons, without loosing realism. The goal is to maintain the overall shape of the object so it does not change significantly, when the number of polygons are changing.

Viz Artist has a built-in LOD mechanism for text objects and internal objects, such as Sphere, Cylinder and so on. This mechanism changes the tessellation of the object, based on its size on screen. However, there is no LOD mechanism for imported objects and models. This is where the LOD plug-in can be used. The LOD plug-in enables switching between different representations (levels) of the same object, based on its distance from the camera and on the zoom level of the camera. A switching range is assigned to each representation to determine the specific distance from camera n which it is replaced by the next representation.

Additionally, the zoom of the camera is taken into consideration; objects that are far away from the camera appear closer (and larger) at a narrow zoom position, or smaller at wide zoom. To accommodate for different zoom values, LOD assumes that the switching ranges are correlative to a reference zoom value. When the actual zoom value differs from the reference zoom value, the switching ranges are automatically adjusted.



To use LOD on an object, you have to import several representations of that object from an external source. The different representations of the object have to be placed under a group node, in the order of detail, this means the most detailed level is on the top of the group. When LOD is placed on the group container it makes only one of its children (levels) visible at a given time, based on the distance between the center of the object and the camera, as well as the current zoom value, and other selectable parameters of the plug-in.

LOD works in conjunction with LOD Manager that controls all the LOD plug-ins in the scene.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

LOD Properties

- **Lock:** Disables distance switching. In certain cases, it is desirable to disable the distance switching and lock LOD to show one of the levels.
- **Lock On Index:** Shows index of the selected level when **Lock** is `On`.
- **Center:** Defines whether the distance from the camera is measured from the origin of the object (**Automatic**) or from a specific point in the object, for example one of the corners (**Manual**)
- **Center X, Y, and Z:** Sets the point from which the distance from the camera is measured (in object coordinates), if the **Center** parameter is set to **Manual**.

Copyright © 2026 Vizrt

- **Range Scale:** Sets a scale factor on the switching ranges. This scale factor allows shortening or lengthening the distance at which object representations are switched.
- **Range 0:** Sets the switching ranges. Those are the distances at which object representations are switched. The number of entries is `n+1` where `n` is the number of children in the group on which LOD is assigned
- **Initialize:** Re-initializes the LOD plug-in. This is required if the number of children of the LOD group has changed after LOD was assigned to the group.

See Also

- LOD Manager

Copyright © 2026 Vizrt

Magnify



The Magnify plug-in offers the possibility to create a simulation of a looking glass and other similar effects.

An area defined by an overlying object an image can be shown in a magnified way.

Compatibility Info: This plug-in works in the Viz Classic Render Pipeline only and only is available if the Render mode is set to Classic.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

This page contains the following topics and procedures:

- Magnify Properties
- To Magnify a Texture

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Magnify Properties

- **Scale:** Sets the scaling of the magnifying effect.
- **Effect Type:** Adds an extra effect, either **Blur** or **Pixel**. This only works on onux.
- **Effect Value:** Sets the value for the selected extra effect.
- **Alpha:** Sets the alpha value for the container.
- **Rebuild:** Applies the changes to the texture.

Copyright © 2026 Vizrt

To Magnify a Texture



1. Add a Container to the Scene Tree.
2. Name it `source`.
3. Add the image or texture that is to be magnified to the *source* Container.
4. Add a Sub-Container to the *source* Container.
5. Name it `magnify`.
6. Add the same image or texture to the *magnify* Container that was used in the *source* container.
7. Add the Circle geometry to the *magnify* Container.
8. Open the transformation editor for the *magnify* Container.
9. Set Position Z to `1.0`.
10. Add Magnify to the *magnify* container.
11. Open the Magnify editor.
12. Set Scale to `2.0`.

Copyright © 2026 Vizrt

Match It



The Match It plug-in sets the axis center (center for rotations) to the camera and moves the object away from the camera by the specified amount of units. By setting an arbitrary camera distance and initializing Match It, the container with Match It is positioned in the specified distance and any rotation is done with respect to the camera position. The container's axis center parameter can also be edited for a more flexible configuration.

Note: Keep in mind that you have to re-initialize the plug-in if the camera is moved.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Match It Properties

- **Initialize:** Resets the pivot to the camera and places the object **CamDistance in Z** units away.
- **CamDistance in Z:** The distance from the camera on the z axis.

Copyright © 2026 Vizrt

Max Size



The Max Size plug-in is used to set max size parameters for text objects.

This allows you to have control over the space the text should use so it does not overflow the designated text area.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

This page contains the following topics and procedures:

- Max Size Properties
- To Set Maximum size

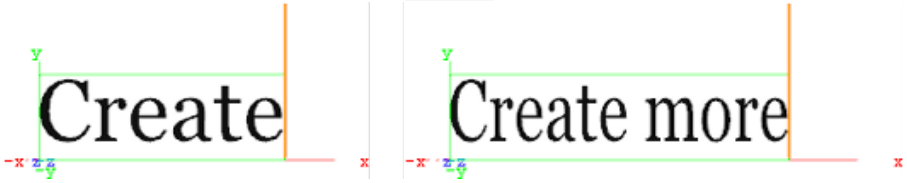
Max Size Properties

- **Default x-scale:** Sets the default x-scale.
- **Default y-scale:** Sets the default y-scale.
- **Max size:** Sets the direction of the size to be controlled:
  - **x:** Controls the width.
  - **y:** Controls the height.
- **Max width/height:** Sets the maximum width or height, depending on whether **x** or **y** is selected for **Max size**.
- **Scale y/x:** Scales the **y** or **x** value accordingly when maximum width or height is reached, respectively.
- **Initialize text scale:** Initializes the plug-in parameters.

To Set Maximum size



1. Add a group container to the scene tree.
2. Add a group container as a Sub-Container of the first (scaled to 1,1,1) and name it `text`.
3. Add a font (for example, Georgia Regular) to the text container.
4. Add MaxSize plug-in to the text container.
5. Click MaxSize and set **Max width** to `200.0`.



6. Click the font and type *Create.*
7. Now type *Create more.* This automatically scales the font's X value as it limits itself to the Max width.

Copyright © 2026 Vizrt



8. *Optional*: Click MaxSize and enable the **Scale Y** parameter to automatically scale the font's Y value.
   When working with scenes that includes vertically aligned text direction, such as East Asian languages where top to bottom columns of text can be desirable, please pay attention to the vertical orientation selected in the Text editor. When the text scale is adjusted with Max size, the selected vertical orientation can affect the position of the text geometry:
   a. **Top:** Centers the geometry at the top of the boundary. Changes to the text scale do not change the geometry position.
   b. **First Line:** Centers the geometry at the base of the first line. Because of this, changes to the text scale also move boundary of the geometry.
   c. **Center:** Moves the boundary when the text scale changes because the center of the geometry is not the same as the center of the boundary, the boundary is moved slightly when the text scale is changed.
   d. **Bottom:** Centers the geometry at the base of the last line of text, but this is not the same as the bottom boundary. This results in the boundary being moved when the text scale is changed. This allows for more flexible choices when designing a scene, and should be selected based on the desired result.

See Also

- Max Size Lines

Copyright © 2026 Vizrt

Max Size Lines



The Max Size Lines plug-in adds multi-line support for Basic Container TextFX with right-to-left languages. Each line of text must be in separate containers, and organized as a sub group of the container holding Max Size Lines.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Max Size Lines Properties

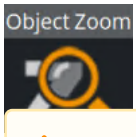
- **Default x-scale:** Sets the default x-scale.
- **Default y-scale:** Sets the default y-scale.
- **Max width:** Sets the max width.
- **Scale y:** Scales the y-value accordingly when max width is reached.
- **Initialize text scale:** Initializes the plug-in parameters.

See Also

- Max Size

Copyright © 2026 Vizrt

Object Zoom



The Object Zoom plug-in lets an object follow the camera and match its orientation.

This creates an effect as if the camera would smoothly zoom to the object.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Default

This page contains the following topics and procedures:

- Object Zoom Properties
- To Zoom an Object Towards the Camera

Object Zoom Properties

- **Counter:** Determines the speed of the zoom operation. Use values from `1-1000`.
- **Dest. Object:** Identifies the name of the container that is affected by this plug-in.
- **Dest. Dist.:** Determines the distance from the camera at which the object stops.
- **Dest. Size %:** Determines the scaling of the object.
- **Path HOffset:** Determines the horizontal offset in camera space.
- **Path VOffset:** Determines the vertical offset in camera space.
- **Initialize (button):** Starts the zoom.

To Zoom an Object Towards the Camera

1. Add Object Zoom to a container.
2. Set the **Dest. Object** parameter to match the containers name.
3. Set the **counter** to at least `1`.
4. Click **Initialize**.
5. Go to **Views** and move the camera to see the container zooming towards the camera.

Copyright © 2026 Vizrt

Omo



The Object Moving (Omo) plug-in gives the user the possibility to create a very realistic animation of a complex imported 3D object, where the object not just changes position or rotates, but where it changes form and moves in a realistic way, like a man walking or a fish swimming.

Omo can also create animation sequences with other sorts of objects. Anything you can put in a container can be used as an item in an Omo animation process.

Omo hides all Sub-Containers and shows them one by one, the one shown is controlled by the **Visible Container** parameter. By animating this value, an animation sequence of all the containers is made. Omo can also be used to show one group at a time.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

Omo Properties

- **Visible Container:** Allows you to select which container that is to be visible.
- **Keep Visible:** Enables you to keep containers visible after they have been revealed one by one.
- **Initialize:** Initializes Omo. All containers in the group are then hidden, except the one selected by the **Visible Container** parameter.

To Animate One Object at a Time



1. Create a group and add Omo to it.
2. Create a number of Sub-Containers to the root container.
3. Add a Sphere geometry and material to each Sub-Container.
4. *Optional*: Animate the Sphere.
5. Open the Omo editor, click the Initialize button, and animate the Visible Container parameter.

See Also

- Control Omo

Copyright © 2026 Vizrt

Parliament

Overview



The Parlament plug-in arranges child containers in a **semicircular (hemicycle) seating layout** — like seats in a legislative chamber or parliament. It is designed for election graphics, political visualizations, and any scene that needs to display items arranged in concentric arc rows.

The plug-in takes the parent container's geometry and material, hides the parent's own geometry, and clones it into N child containers distributed across configurable concentric rows.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Note: This plug-in is compatible with the Classic Render Pipeline only.

This page contains the following topics and procedures:

- Overview
- Parameters
- How It Works
  - Seat Distribution Algorithm
  - Live Update vs. Rebuild
- Setup Guide
  - Basic Parliament Visualization
  - Coloring Seats by Party
  - Partial Arcs
- Usage Example: Election Night Graphic
- Technical Notes

Parameters

Parameter	Default	Range	Description
Number Of Elements	40	1 – 1,000	Total number of seats/objects to place in the hemicycle.
Number Of Rows	3	1 – 50	Number of concentric rows.
Rebuild Geometry	—	—	Deletes all child containers and rebuilds them from scratch.
Inner Radius	70.0	0 – 10,000	Radius of the innermost row.
Outer Radius	200.0	0 – 10,000	Radius of the outermost row.

Copyright © 2026 Vizrt

Parameter	Default	Range	Description
Start Angle	−90°	−180° – 180°	The arc's starting angle. Default −90° starts at the left.
End Angle	90°	−180° – 180°	The arc's ending angle. Default 90° ends at the right, forming a semicircle.
X Scale	1.0	0.01 – 10	Per-element X scale.
Y Scale	1.0	0.01 – 10	Per-element Y scale.
Z Scale	1.0	0.01 – 10	Per-element Z scale.
Lock	On	—	Uniform scaling: when enabled, changing one axis applies to all three.

How It Works

Seat Distribution Algorithm

1. The arc between **Start Angle** and **End Angle** is divided into concentric rings, spaced evenly between the inner and outer radius.
2. Each row's arc length is computed. Seats are distributed proportionally: Outer rows (with longer arcs) receive more seats than inner rows.
3. A greedy balancing algorithm picks the row with the lowest fill ratio for each seat, ensuring even distribution.
4. Each child container is positioned on the arc and **rotated to face the center** of the hemicycle.

Live Update vs. Rebuild

The plug-in supports two update modes:

Mode	When	What happens
Live update	Radius, angles, or scale change	Existing child containers are repositioned and rescaled (no tree modification). Fast and non-destructive.
Rebuild	**Rebuild Geometry** button pressed, or element/row count changed	All child containers are deleted and recreated. Required when the number of elements or rows changes.

Important: The parent container must have a **geometry** assigned (for example, a Sphere, Cube, or custom object) before adding the Parlament plug-in. The plug-in clones this geometry into each child seat. The parent's own geometry is hidden automatically.

Copyright © 2026 Vizrt

Setup Guide

Basic Parliament Visualization

1. Create a container and assign a geometry (for example, a small **Sphere**) and a material.
2. Add the **Parlament** plug-in to the container.
3. Set **Number Of Elements** to the total seat count (for example, `598` for the German Bundestag).
4. Set **Number Of Rows** to the desired row count (for example, `15`).
5. Adjust **Inner Radius** and **Outer Radius** to fit your scene layout.
6. The plug-in creates child containers arranged in a semicircle.



Note: If the object you used for creating the seats contains many polygons, the total product of the parliament container can quickly be too heavy to render.

Copyright © 2026 Vizrt

Coloring Seats by Party

After the plug-in creates the child containers, each seat is a separate container in the scene tree. You can:

- Apply different **materials** to individual seats to represent party colors.
- Use **shared memory** or **ControlObject** to drive seat colors from data.
- Animate individual seats (for example, highlight a seat when a vote is cast).

Partial Arcs

To create layouts other than a semicircle:

- **Full circle:** Set Start Angle to `-180°` and End Angle to `180°`.
- **Quarter arc:** Set Start Angle to `-45°` and End Angle to `45°`.
- **Inverted (seats facing outward):** Swap Start and End Angle values.

Usage Example: Election Night Graphic

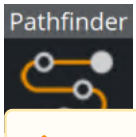
1. Create a Sphere geometry container with a small radius (for example, scale 0.5).
2. Apply a neutral gray material.
3. Add the **Parlament** plug-in with 598 elements across 15 rows.
4. Set Inner Radius to `100`, Outer Radius to `300`.
5. The plug-in generates 598 spheres arranged in the hemicycle pattern.
6. Use a script or ControlObject to iterate through seats and assign party-colored materials based on election results data.

Technical Notes

- The plug-in positions seats in the **XZ plane**: The arc opens along X, with rows stacked along Z (depth). Y is used for the vertical axis.
- Each seat is rotated around the **Y axis** to face the center of the hemicycle.
- The **Scale Lock** feature propagates any scale change to all three axes, ensuring uniform seat sizing.
- Rounding errors in seat distribution are compensated on the last row to ensure the total always matches the requested element count.
- For best performance with large seat counts (500+), avoid triggering **Rebuild** frequently, use live update (radius/angle/scale changes) when possible.

Copyright © 2026 Vizrt

PathFinder



The PathFinder plug-in creates an animation path based on predefined shapes.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

PathFinder Properties

The properties of PathFinder are different for each selected mode:

- Common Properties
- Circle Properties
- Wave Properties
- Spring Properties
- Line Properties

Common Properties

- **Shape Type:**
  - **Circle:** Uses an existing 2D Circle geometry for the animation path.
  - **Wave:** Uses an existing 2D Wave geometry for the animation path.
  - **Spring:** Uses an existing 3D Spring geometry for the animation path.
  - **Line:** Uses an existing 2D Line geometry for the animation path.
- **KeyFrame Smooth Level:** Defines how smooth the spline is.
  - **Linear:** Does not add smoothing.
  - **1-6:** Determines the level of smoothing.
- **Direction:** Determines the direction the spline is built.
  - **Clock:** Builds the spline in a clockwise direction.
  - **Counter Clock:** Builds the spline in a counter clockwise direction.
- **Axis:** Determines the axis orientation.
  - **XY:** Builds over the X and Y axes (facing up).
  - **YZ:** Builds over the Y and Z axes (facing the screen).

Circle Properties

- **Radius:** Determines the radius of the circle.
- **Resolution:** Determines the number of key frames on the circle.
- **Start Point:** Defines the starting angle for the circle. This value can be less than `0` and more than `360`, but the value always corresponds to the degree value of a circle.

Wave Properties

- **Wave Length:** Determines the length of each wave.
- **Wave Amplitude:** Determines the height of each wave.
- **Resolution:** Determines the number of key frames on each wave count.

Copyright © 2026 Vizrt

- **Count:** Determines the number waves to create.
- **Start Point:** Determines how the wave begins. A value of `0` starts at the bottom, while a value of `100` starts at the top.

Spring Properties

- **Spring Type:** Determines the type of spring path.
    - **Slalom:** Creates a smooth shape.
    - **Steps:** Creates a shape more like circles connect to one another.
- **Radius:** Determines the radius of the spring.
- **Radius Change %:** Increases/decreases the radius size. The spring shape turns into a cone when the radius is decreased.
- **Distance:** Determines the height of the spring.
- **Resolution:** Determines the number of key frames on each spring count.
- **Count:** Determines the number of spring iterations.

Line Properties

- **Axis:** Builds line on X, Y or Z axis.
- **Length:** Determines the length of the line.
- **Resolution:** Determines the number of key frames on the line.

Copyright © 2026 Vizrt

Rotation Order



The Rotation Order plug-in changes the rotation order of a container in Viz Artist to match the Softimage XSI rotation order.

Rotation Order is automatically added to some containers when Softimage XSI scenes are imported into Viz Artist.

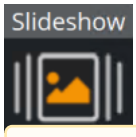
Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Rotation Order Properties

- **Rotation Order:** Sets the order of rotation to either:
  - **XYZ**
  - **ZYX**
  - **ZXY**
  - **YXZ**

Copyright © 2026 Vizrt

Slide Show



The Slide Show plug-in allows you to create a sequence where a group of containers fade in and out one by one in the same manner as a normal slide show.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

This page contains the following topics and procedures:

- Slide Show Properties
- To Create a Slide Show

Slide Show Properties

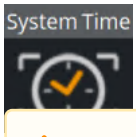
- **Blend Value:** Controls the slide show sequence. Animate this value to create the change from slide to slide.
- **Use Offset:** Enable this option if the slides have the same initial position and you want them to move along some of the axes as the slides are being shown.
- **X, Y, and Z Offset:** Sets the offset values of each slide calculated form the one previously shown.
- **Fade Out:** May be set **Off**, **On** (every slide) or **Last** (last slide only)
- **Start with:** Sets what should be visible if slide show is not started.

To Create a Slide Show

1. Add a group to the scene tree, and name it `slideshow`.
2. Add a number of Sub-Containers that are used to house the slide(s).
3. Add Slide Show to the slideshow container.
4. Position the Sub-Containers in advance using the transformation editors for each container, or open the Slide Show editor and offset them to create a moving sequence.
5. Animate the blend value to create the slide show sequence.
6. Click the Initialize button to initialize the plug-in. Each of the Sub-Containers automatically get an alpha icon.

Copyright © 2026 Vizrt

System Time



The System Time plug-in creates a variety of time and date settings based on the system time.

System Time adheres to the standard **IEEE Std 1003.1, 2004 Ed.**

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

System Time Properties

- **format (aAbBcdDHIijlmMpSUwWxXyYzZ/:,.):** Sets the format of the system time.
- **Language:** Sets the language format of the system time. Options:
  - Default (system specific)
  - English
  - German
  - Norwegian
- **Seconds offset from system clock:** Sets the number of seconds the system clock should be offset (default is `0`).
- **Hours offset from system clock:** Sets the number of hours the system clock should be offset (default is `0`).
- **System time zone:** Sets the system time zone.
- **Show Time Zone:** Shows the result of the system time zone.
- **Lower Case Text:** Shows all text in lower case.

Time Zone: tzn[+|-] hh[:mm[:ss]] [dzn]

- **tzn:** Time zone name (for example, PST).
- **hh,mm,ss:** Offsets from the local timezone to UTC (not the reverse).
  - For time zones ahead of the set one, the time difference is negative.
  - For time zones behind the set one, the time difference is positive.
- **dzn:** Daylight-saving-time zone (for example, PDT).

Examples: EST5EDT, PST5PDT, GST-1GDT

See Also

- Analog Watch
- Clock Rotation

Copyright © 2026 Vizrt

Temo



The Temo plug-in gives the user the possibility to create an image animation sequence, using a method much like the one used for making a cartoon film. The basic input for the plug-in is a single image consisting of many tiled and equally sized squares set up in a matrix.

Each square of the image is made up to be a snapshot of an animation sequence, just like each picture frame is on a normal film. The image must be made up in advance with the aid of an image editing tool. In the property editor, you tell the Temo plug-in how many tiled squares there are in the X- and the Y-axis. The plug-in is then able to show the tiles of the image one by one. By animating this, a “film like” sequence is created. The plug-in has only a small influence on the rendering performance.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

Setup

Temo looks for the input texture differently in the Classic Render Pipeline and the Viz Engine Render Pipeline:

- **Classic Render Pipeline:** Temo looks for an Image plug-in in the same container and use the assigned color texture (Default slot) as the input texture.
- **Viz Engine Render Pipeline:** Temo looks for a Phong material in the same container and uses the assigned color texture (Color slot) as the input texture.

Temo Properties

- **Number of tiles horizontal:** Defines the number of tiles in the X-axis of the source image.
- **Number of tiles vertical:** Defines the number of tiles in the Y-axis of the source image.
- **Show Tile Number:** Selects the tile to be shown. The range of tile numbers is the product of the two above parameters. If there are three in the horizontal and four in the vertical plane, the total is 12. The tile with the coordinates X1Y1 is shown first, then X2Y1 and so on. By animating the whole range of this value, a “film animation” can be created. The quality of the animation depends of course on the number of tiles and on the degree of transformation from one tile to the next throughout the whole sequence.

Copyright © 2026 Vizrt

Text Auto Scale



The Text Auto Scale plug-in automatically scales text within a **Text Box** area as defined in the Text Editor.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Text Auto Scale Properties

- **Enabled:** The plug-in is enabled by default when added to a container. Click the button to disable.
- **Default x-scale:** Sets the default scale value for the X-axis.
- **Default y-scale:** Sets the default scale value for the Y-axis.

Tip: When using Text Auto Scale, Text Box scaling should be set to `Off` to avoid conflict in scaling operations.

Copyright © 2026 Vizrt

Text Link



The Text Link plug-in copies the text from the source container to up to ten linked containers.

The text you enter in the source container is repeated in all the linked containers.

The linked containers must contain a font.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Text Link Properties

- **Container 1: 10:** Drop zones for the containers you want to link.
- **Initialize:** Initializes Text Link.
- **Do it now:** Performs the operation immediately.

To Link Text to Multiple Containers

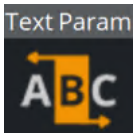
1. Add a group container to the scene tree.
2. Add Text Link and a font to the group container.
3. Add a number of containers to the scene tree with a font on each.
4. Open the Text Link editor.
5. Drag and drop the containers with the fonts onto the Container 1-n drop zones.
6. Click **Initialize** to apply change.

See Also

- Image Link

Copyright © 2026 Vizrt

Text Parameters



The Text Parameters plug-in offers the designer the possibility to animate text parameters, as standard text parameters in the Text Editor cannot be animated. The parameters for the Text Parameters can only be used for animation.

The settings are not visible on the text until an animation update has been made.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Text Parameters Properties

- **Use Kerning:** Enables or disables the use of Kerning.
- **Tracking:** Sets the space between each letter.
- **Leading:** Sets the space between each line of text.
- **Word Spacing:** Sets the space between each word.
- **Use Soft Shadow:** Enables the use of Soft Shadow. Level 4 is the most blurred, softest shadow. The Drop Shadow option must be enabled for the Soft Shadow to be visible.
- **Use Drop Shadow:** Enables or disables Drop Shadow.
- **Direction:** Sets the direction of the Drop Shadow in degrees.
- **Distance:** Sets the distance of the Drop Shadow.
- **Z Offset:** Sets the offset of the Drop Shadow in the Z-axis.
- **Color:** Sets the color of the shadow.

Copyright © 2026 Vizrt

TextBG



The TextBG plug-in creates a dynamically sized background for text, images or video. For example, you can use it to create a background that expands as text is entered in the Text Editor.

TextBG must be added to a container that also holds a Rectangle, Noggi, Fade Rectangle or Cube geometry plug-in. To use TextBG with anything other than fonts, set the **Text Parameter** option to **Scale**.

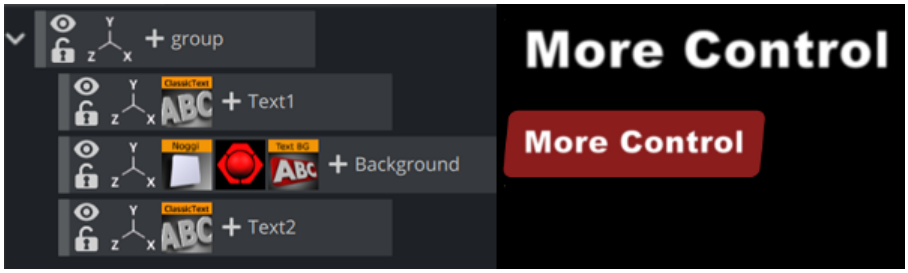
> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

TextBG Properties

- **Size Parameter:** Sets the size of the background to adjust to:
  - **W/H (width/height):** Adjusts the width and height of the background geometry to the size of the typed text.
  - **Scale:** Scales the background to the size of the target container.
- **Text Container:** Sets the inputs source for TextBG:
  - **PREV (previous):** Applies the background geometry to the container previous to the TextBG container.
  - **PARENT:** Applies the background geometry on the parent container.
  - **NEXT:** Applies the background geometry to the next container from the TextBG container.
  - **OTHER:** Enables the **Other container** field.
- **Other container:** Available when **OTHER** is selected. Drag a container from the same group to the drop area. Click on **R** to remove the Container.
- **Direction:** Sets the direction in the background grows when adjusting itself to the text. Available options are **X**, **Y**, or the **XY**-axes in combination.
- **Hide On Empty:** Hides the background container if there is no text. Set to `off` to enable **Always Keep Margins**.
  - **Always Keep Margins:** Makes the background geometry retain the defined margins as minimum width and height, even when the text container is empty. Enabled by setting Hide On Empty to `off`.
- **High Precision:** Adds one decimal point precision to the **Margin** fields.
- **Right, Left, Bottom and Top Margin:** Sets the margin between the text and the background in centimeters. Accepts values in the range `-1000.0` to `1000.0`.
- **Lock Margins:** Sets the margin to be the same for all.
- **Enable Substring:** Enables the **First** and **Last Character** options. Substring cannot be used with multiple texts.
  - **First Character:** Sets on which character the background should start animating.
  - **Last Character:** Sets on which character the background should stop animating.

Copyright © 2026 Vizrt

To Add Background to the Previous or Next Containers



1. Add a group container to the Scene Tree.
2. Add two sub-containers to the `group` container:
    - Name the top one `Text1`.
    - Name the bottom one `Background`.
3. Add a font to the *Text1* container.
4. Open the Transformation Editor of the Text1 container.
    a. Set **Position Z** to `1.0`.
    b. Enter some text. In this example, the text is `More Control`.
5. Copy the *Text1* container and place the copy below the `Background` container in the scene tree. Move the container so that both texts are visible, as seen in the screenshot above.
6. Rename the new container to `Text2`.
7. Add a geometry, for example the Noggi, a material, and TextBG to the `Background` container.
8. Open the TextBG editor.
9. Toggle the **Text Container** property from **PREV** to **NEXT**. The background geometry moves from *Text1*, or the *previous* container, to *Text2,* the *next* container. If you resize the text in the *Text2* container, the background geometry automatically resizes to fit the text.

> **Tip:** To apply a background to a different container, select **OTHER** as **Text Container** and drag the desired container to the drop area.

10. To use TextBG, to apply a background to the parent container, which is `group` in this example, select **PARENT** as **Text Container**. This requires that the parent container contains a text, image or video object working with TextBG.

Copyright © 2026 Vizrt

Texture Fit

Texture Fit

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

Compatibility: V3 Classic, V4 PBR **Category:** Tools

Overview



The Texture Fit plug-in provides advanced control over how textures (images) are displayed on a container. It automatically adjusts UV coordinates so that a texture maps correctly onto a container regardless of the container's dimensions or aspect ratio.

This is especially useful in **adaptive / multi-format** workflows where the same scene is rendered in different aspect ratios (e.g., 16:9 landscape and 9:16 portrait). Texture Fit ensures that images always look correct without manual UV adjustments.

Prerequisites

Texture Fit **requires** the following setup on the container:

Requirement	Why
Box Transformation mode	Texture Fit reads the container's pixel-based width and height. It does **not** work with 3D Transformation.
Flexbox plug-in	Flexbox computes the actual box layout dimensions that Texture Fit reads. Without Flexbox, the container has no calculated box size.
Geometry (e.g., Rectangle)	A renderable geometry is needed to display the texture.
Material with a Texture	A Phong or PBR material definition with an assigned texture (base color slot). Texture Fit modifies the UV scale/offset on this material.

Setup Guide — Using Texture Fit with Box Transformation and Flexbox

Step 1: Enable Box Transformation

Select the container and switch from 3D Transformation to **Box Transformation** in the transformation panel. This gives the container pixel-based width and height properties instead of 3D position/rotation/scale.

Step 2: Add a Flexbox Plug-in

Copyright © 2026 Vizrt

Add the **Flexbox** plug-in to the container (or a parent container). Flexbox calculates the layout box dimensions based on the container hierarchy. Texture Fit depends on these calculated dimensions to determine the container's aspect ratio.

Step 3: Add Geometry and Material

1. Assign a **Rectangle** geometry (or another suitable 2D geometry) to the container.
2. Add a **Phong** or **PBR** material definition.
3. Assign a **texture** (image) to the material's base color slot.

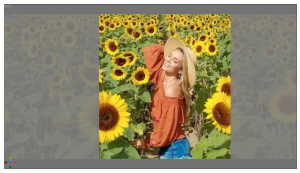
Step 4: Add Texture Fit

Add the **Texture Fit** plug-in to the container's function stack.

Important — Plug-in Order: Texture Fit must appear**after**Flexbox in the function stack. Flexbox needs to compute the box dimensions first; Texture Fit then reads those dimensions to calculate the correct UV mapping. If the order is reversed, Texture Fit may read stale or zero-sized box values.

Step 5: Choose a Mapping Method

Select one of the three mapping methods depending on your use case:

Method	Best For	Behaviour	
Contains	Showing the full image without cropping	The entire image is visible inside the container. Empty space (letterbox/ pillarbox bars) may appear if the aspect ratios don't match.	
Cover	Filling the container completely	The image fills the entire container with no empty space. Parts of the image may be cropped if the aspect ratios differ.	
Region of Interest	Guaranteeing a specific area is always visible	A user-defined rectangle (normalized 0–1 coordinates) is always visible. Other parts of the image are cropped as needed when the container changes size.	

Parameters

Texture Size Mode

Copyright © 2026 Vizrt

Value	Description
auto (default)	Reads the texture dimensions automatically from the material's base color texture.
manual	Lets you enter custom texture dimensions (useful when the texture is not yet assigned or when overriding).

When set to **manual**, the **Texture Width** and **Texture Height** parameters become visible (default: 1920 × 1080).

Mapping Method Parameters

Contains

- **Offset U / Offset V** — Shifts the image within the container (range: −1 to 1).
- **Scale Pivot U / Scale Pivot V** — Sets the anchor point for scaling (range: −1 to 1).

Scale U/V are locked to 1.0 in this mode.

Cover

- **Scale U / Scale V** — Scales the image (range: −1000 to 1000, default: 1.0).
- **Offset U / Offset V** — Shifts the image within the container.
- **Scale Pivot U / Scale Pivot V** — Sets the anchor point for scaling.

Region of Interest

- **X min / X max** — Horizontal region bounds (0 = left, 1 = right).
- **Y min / Y max** — Vertical region bounds (0 = top, 1 = bottom).

All scale/offset/pivot controls are hidden in this mode — the UV mapping is fully automatic based on the defined region.

Adaptive Multi-Format Example

A common use case is a lower-third graphic that must work in both **16:9** (landscape) and **9:16** (portrait) formats:

1. Create a container hierarchy with **Flexbox** managing the layout.
2. The image container uses **Box Transformation** so its width and height adapt to the available space.
3. Add **Texture Fit** with the **Cover** mapping method.
4. As the output format changes, Flexbox recalculates the container dimensions, and Texture Fit automatically adjusts the UV mapping so the image always fills the container — no manual intervention required.

For editorial images where a specific subject must always be visible (e.g., a person's face), use the **Region of Interest** method instead. Define the ROI rectangle around the subject, and Texture Fit ensures it remains visible regardless of the output format.

Technical Notes

- Texture Fit sets the texture sampler wrap mode to **clamp-to-border** on both the base color and emissive texture slots.

Copyright © 2026 Vizrt

- In **auto** mode, the texture dimensions are read from the material's base color texture each frame. If no texture is assigned, the manual fallback values (1920 × 1080) are used.
- The plug-in executes during the render callback (after layout), so it always uses the most up-to-date box dimensions from Flexbox.

Example

The image shown in a landscape and in a portrait situation:

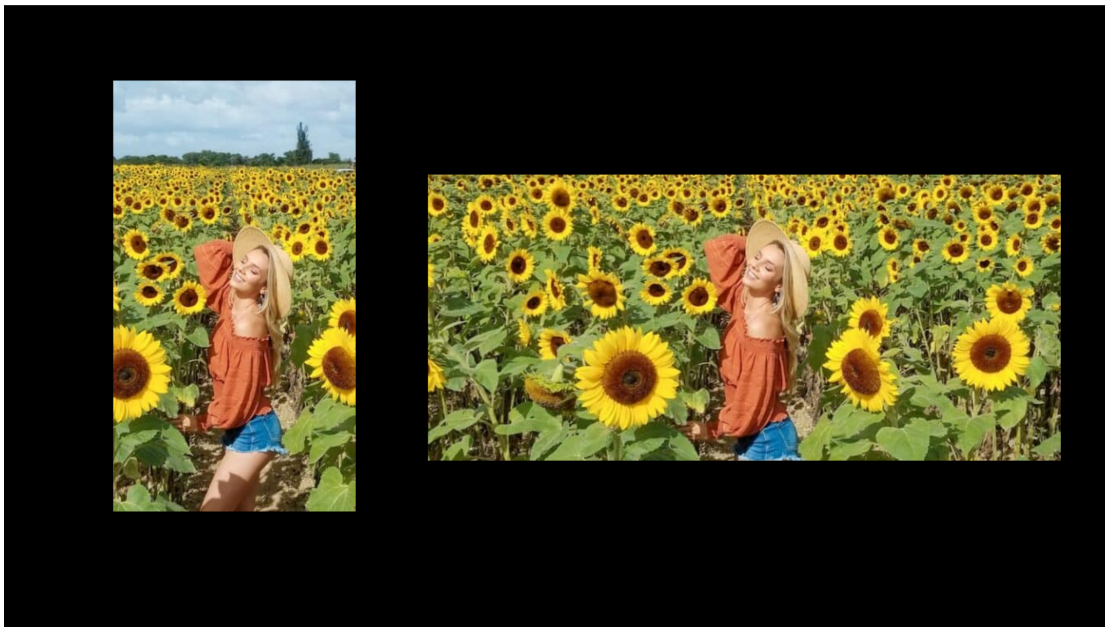


Image Credits: dezistyle.com

TransitionLayers



The TransitionLayers plug-in enables a new way to define the two dynamic images of a scene-transition scene, using the **Dynamic Scene** Media Asset.

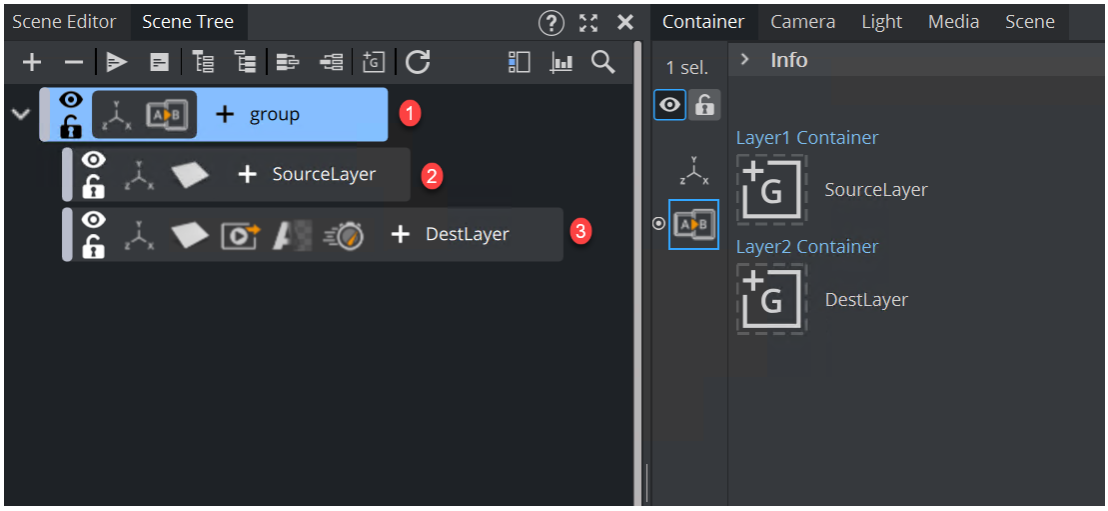
Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

TransitionLayers Properties

- **Layer1 Container:** Container holding the scene to transition *from*. The referenced containers' name is shown to the far right.
- **Layer2 Container:** Container holding the scene to transition *to*. The referenced containers' name is shown to the far right.
- **Reset buttons:** Each layer container has a reset button, which empties the container reference.

Tip: By pressing the group button for the source or destination layers, the corresponding container is selected in the scene tree.

When adding TransitionLayers to a container, TransitionLayers automatically creates the two sub-containers required when either the Layer1 Container or Layer2 Container are clicked in the plug-in properties:



1. Parent container holding TransitionLayers.
2. Sub-container holding the Source Layer (scene to transition from*)*, or *Layer1* Container.
3. Sub-container holding the Destination Layer (scene to transition to), or *Layer2* Container. In addition, a default two second alpha in animation is added for the *Layer2* Container to the **Default** director in the Stage. This default animation can of course be changed as desired.

Tip: Containers holding the dynamic images for the transition scene can also be dragged onto the desired *Layer* container in the plug-in properties pane.

Copyright © 2026 Vizrt

VCF Parameter



The VCF (Virtual Camera Flight) Parameter plug-in works in conjunction with the VCF scene plug-in. This allows you to create a seamless interpolated transition from a VCF to a real camera, and vice versa. This is only a relevant function to set up if you have purchased the Viz Virtual Studio expansion components.

You must have a real camera with data tracking enabled, which is set in remote mode in the camera editor. In case you have several tracked cameras, the virtual camera interpolates its position to the real camera that is selected On Air.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

This page contains the following topics and procedures:

- VCF Parameter Properties
- To Use the Virtual Camera Flight

VCF Parameter Properties

- **Real Camera %:** Defines the percentage the real camera influences the position of the virtual camera.
- **Real Camera Deformation %:** Defines the percentage the real camera influences the lens distortion.
- **Zoom:** Sets the actual zoom (FOV-Y) value of the virtual camera

To Use the Virtual Camera Flight

1. Add VCF to the scene setup plug-ins drop zone.
2. Create a new group in your scene and add two new containers under it. These are to be the objects that defines the virtual camera flight. One defines the position, the other the direction.
3. Name the containers according to the names entered in VCF (e.g. `T_POS` and `T_ROT`).
4. Click **Initialize** to finish.
5. Add VCF Parameter to the container that holds the position object.
6. Animate your virtual camera flight using the two objects to define position, and direction. Do **not** switch the whole container invisible or else the animation does not run. You may switch the objects to be invisible, at any time.
7. Animate the **Real Camera %** and **Real Camera Deformation %** parameters in what frame you want to have the real camera at 0% visibility and when at 100% visibility (the longer the animation is, the smoother the interpolation is: 100 F should be good).
    - If you start with a virtual camera the first Key Frame (0) should have 0% of the real camera, and the last should be 100% of the real camera.
    - The virtual camera generates a full key (masking the FG), if you choose virtual studio in the scene setup: background.
8. Make sure your FG object/talent are not seen while switching to the real camera. They should show in the frame after the switch.

> **Note:** It is not possible to animate the roll of the camera.

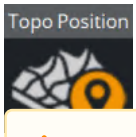
Copyright © 2026 Vizrt

2.4.19 Topology Container Plug-ins

The following Container plug-ins are located in the Topo folder:

- TopoPos

TopoPos



The TopoPos plug-in adjusts the Y value of a container.

Works on all containers under the one TopoPos is dragged onto.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Topo

Topo Pos Properties

- **Offset:** Sets the offset of the container in the Y-axis.

2.4.20 Transformation Container Plug-ins

The following Container plug-ins are located in the Transformation folder:

- Justifier
- Vertex Bone
- Vertex Skin

Justifier



The Justifier plug-in enables you to animate the justification of an object using the object's height, width or depth as its coordinate system rather than using the Viz Artist/Engine coordinate system.

This plug-in acts on the local space of the container it is attached to. So a parent container can be used to rotate the container for use of arbitrary (world space) axes.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Transformation

This section contains information on the following topics:

- Justifier Properties
- To Justify an Object

Justifier Properties

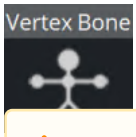
- **Align on X, Y and Z:** Aligns the position. Left, Right and Center enables the Anchor option(s). Adv (advanced) enables the Anchor and Shift option(s).
  - **X, Y and Z Anchor:** Sets the offset for the alignment.
  - **X, Y and Z Shift (%):** Sets the shift of the position values to specify alignment between left/right, top/bottom and front/back (for example, an Anchor at `50.0`, Shift at `-10%`, gives an Anchor at `45.0`.). Range is `-100` to `100`.
- **Use rotated edges:** If this is set to true and the container has been rotated, a new axis aligned bounding box is used to compute the alignment. (I.e. a non rotated bounding box completely containing the rotated container).

To Justify an Object

1. Create a container that can only move up and down.
2. Add the Justify plug-in on the container.
3. Set Align on X to `Center`.
4. Set Align on Z to `Center`. Now the container can only be moved along its y axis.
5. Use the X Anchor and Z Anchor to move it along the respective axis.

Copyright © 2026 Vizrt

Vertex Bone



The Vertex Bone plug-in is used to mark which containers act as bones for Vertex Skinning.

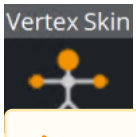
Note: Do not remove this plug-in as this would need a clean, new import (not re-import) to get the scene running again.

Vertex Bone Parameters

This plug-in has no editable parameters.

Copyright © 2026 Vizrt

Vertex Skin



Vertex Skin is the main plug-in that marks a mesh that uses Vertex Skinning.

> **Note:** Do not remove this plug-in as this would need a clean, new import (not re-import) to get the scene running again.

Vertex Skin Parameters

- **Root:** Defines the container under which Vertex Skin finds the bones to use.
- **Show bones:** Verifies the location of each bone visually.
- **Show the weight of a bone:** Visualizes the influence of the selected bone container on the target mesh. Only usable with no other Shader present.
- **Ignore transform on mesh container:** Can be deactivated to incorporate the transformation of the mesh container.

> **Note:** Deactivating is strongly discouraged.

- **Shared Memory Tracking:** Uses data from shared memory to update the transformations of bones.
  - **Shared Mem. Key:** Defines the shared memory key name that receives the data field value.
  - **Bone Key Prefix:** Defines a prefix that is added to Vertex Bones.
  - **Position Root Bone only:** Positions the Root Bone without positioning the other bones.

Motion Capture Live Tracking

To apply live motion capturing data to a Viz geometry, the following steps are required:

- Setup Viz Tracking Hub for Motion Analysis Motion Capturing (See the **Motion Analysis Integration** page in the **Appendices** section of the Tracking Hub User Guide).

> **Note:** Requires Tracking Hub 1.1.2. or newer

- Load motion capture data in Motion Analysis Cortex control application.
- Run Viz Artist with activated Tracking Hub usage.
- Load the imported mesh and navigate to the first Vertex Skin plug-in instance in your scene.
- Activate *Shared Mem Tracking* with the key name set in Cortex. You should see all available tracking bones in the Vertex Skin plug-in's textbox below. This indicates that the information flows correctly from Cortex to Tracking Hub to the Viz Engine.
- Due to accuracy issues of the motion capture system, it could make sense to activate *Position Root Bone only.* This translates only the root bone and all related child bones are rotated.

Copyright © 2026 Vizrt

2.4.21 Visual Data Tools Container Plug-ins

The following container plug-ins are located in the Visual Data Tools folder:

- Area Stack
- Bar Stack
- Data Fit
- Data Import
- Data Label
- Data Storage
- Line Stack

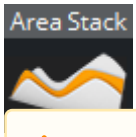
See Also

These plug-ins are helpers working in combiantion with the Visual Data Tools.
Please have a look into

- Area Chart
- Bar Chart
- Line Chart
- Pie Chart
- Scatter Chart
- Stock Chart
- Tutorial on www.vizrt.com under the Training section.

Copyright © 2026 Vizrt

Area Stack



The Area Stack plug-in creates a stacked chart consisting of several Area Charts.

Note: This plug-in is located in: Plugins -> Container plug-ins -> VisualDataTools

This page contains the following topics and procedures:

- Area Stack Properties
- To Create a Scene with Area Stack

Area Stack Properties

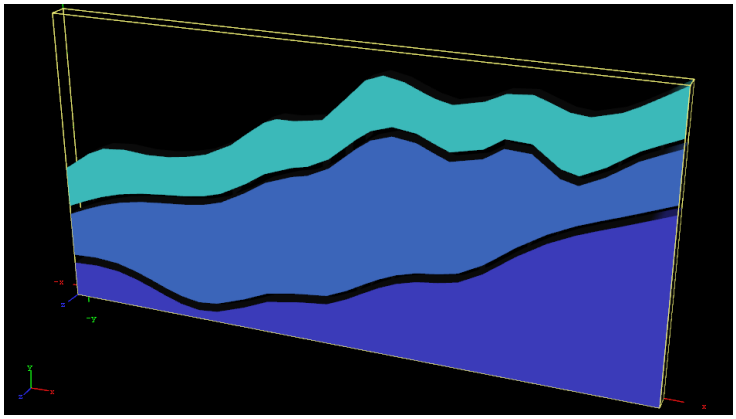
- **Initialize stacked chart:** Refreshes the stacked chart, based on Area Chart plug-ins in the sub-containers. The stacked chart detects when a chart is added/removed from the stack, and updates the chart accordingly. However, some actions, such as re-ordering the charts within a stack, are not updated automatically. In these cases, press **Initialize stacked chart** to update the stack.
- **Stack Gap:** Adds space between each chart.

Note: The remaining properties in the Area Stack plug-in are the same as used in Area Chart. Area Chart features Specify X Values, DataY Compare, and Bevel, are disabled in stacked charts.

To Create a Scene with Area Stack



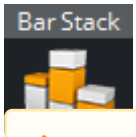
Copyright © 2026 Vizrt



1. Create a new container.
2. Add an Area Stack plug-in into this container.
3. Add two or more Area Chart plug-ins to become children of this container.
4. Add Data Y into each chart. They should have the same number of nodes.
5. In the settings of the Area Stack plug-in, press **Initialize Stacked Chart** to refresh the chart.
6. Use the Area Stack plug-in to set the chart parameters that are common for all charts in the stack.

Copyright © 2026 Vizrt

Bar Stack



The Bar Stack plug-in creates a stacked chart consisting of several Bar Charts.

Note: This plug-in is located in: Plugins -> Container plug-ins -> VisualDataTools

This page contains the following topics and procedures:

- Bar Stack Properties
- To Create a Scene with Bar Stack

Bar Stack Properties

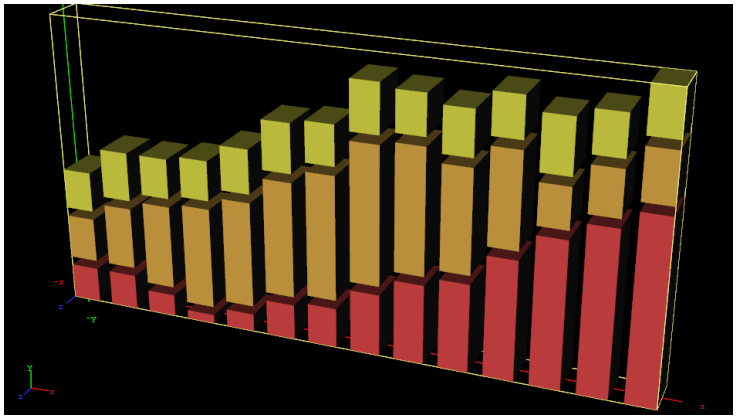
- **Initialize stacked chart:** Refreshes the stacked chart, based on Bar Chart plug-ins in the sub-containers. The stacked chart detects when a chart is added/removed from the stack, and updates the chart accordingly. However, some actions, such as re-ordering the charts within a stack, are not updated automatically. In these cases, press **Initialize stacked chart** to update the stack.
- **Stack Gap:** Adds space between each chart.

Note: The remaining properties in the Bar Stack plug-in are the same as used in Bar Chart. Bar Chart features Specify X Values, DataY Compare, and Bevel, are disabled in stacked charts.

To Create a Scene with Bar Stack



Copyright © 2026 Vizrt



1. Create a new container.
2. Add a Bar Stack plug-in into this container.
3. Add two or more Bar Chart plug-ins to become children of this container.
4. Add Data Y into each chart. They should have the same number of nodes.
5. In the settings of the Bar Stack plug-in, press **Initialize Stacked Chart** to refresh the chart.
6. Use the Bar Stack plug-in to set the chart parameters that are common for all charts in the stack.

Copyright © 2026 Vizrt

Data Fit



The Data Fit plug-in listens for incoming data and modifies and redistributes it to another Shared Memory key.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> VisualDataTools

This page contains the following topics and procedures:

- Data Fit Properties
- To Create a Scene with Data Fit

Data Fit Properties

- **Shared Mem:** Changes between Scene-, Global- and Distributed-Shared Memory. Use Inactive memory to not forward any values via Shared Memory.
- **Transfer Mode:** Sets string- or array-based data transfer.
- **Data Delim. Src.:** Defines the value separator sign(s) for all data sources.
- **Data Delim. Dest.:** Defines the value separator sign(s) for all data destinations.
- **Key Data Src.1-5:** Shared Memory key name for input 1-5.
- **Key Data Dest.1-5:** Shared Memory key name for output 1-5.
- **Scale:** Scales each incoming value by that factor.
- **Offset:** Adds an Offset to each incoming value.
- **Autoscale:** Activates automatic scaling mode. Now all data is scaled to a certain range defined by the following two parameters:
  - **Start:** Determines autoscale starting range.
  - **Stop:** Determines autoscale stopping range.

To Create a Scene with Data Fit

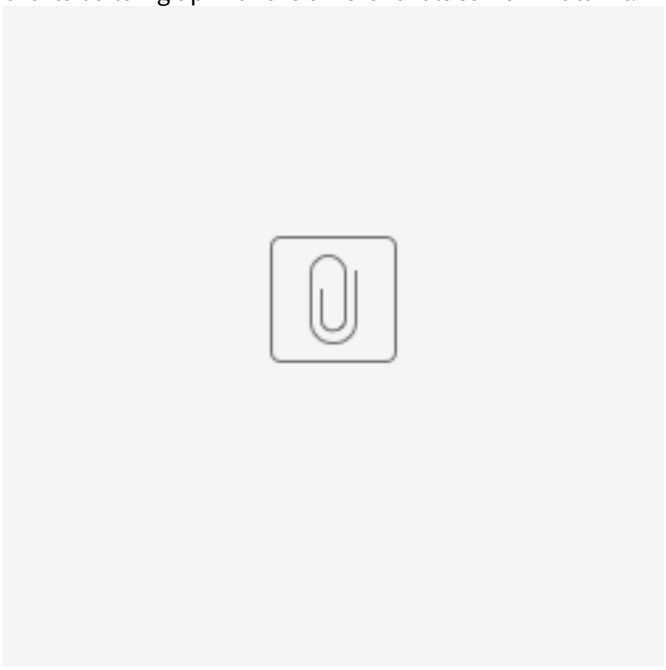


This procedure feeds two charts with data. One with the original and the other with the modified data from Data Fit.

1. Create a new container.
2. Drag and drop a Data Storage and a Data Fit plug-in into this container.
3. Drag and drop two Bar Chart plug-ins to become children of this container.
4. Open the first Bar Chart editor and set Shared Mem. to Scene and KeyData to `MyData`.
5. Do the same for the second Bar Chart but use *MyDataFit* as the key name.

Copyright © 2026 Vizrt

6. Go to the Data Fit plug-in and set its Shared Mem. type also to Scene
7. Set Key Data Src.1 to *MyData* and Key Data Dest.1 to *MyDataFit* > set Scale to `2.0`. The input Shared Mem key is specified with Key Data Src. and the output with Key Data Dest.
8. Now edit Data Storage for the input values: set Shared Mem. to Scene, Key Data1 to MyData and type in some example values for Data1: `80,40,50,70,80,60,80`. As soon as you start typing you see both charts building up with the different values from Data Fit.



Copyright © 2026 Vizrt

Data Import



The Data Import plug-in enables Microsoft Excel (*.xls* and *.xlsx*) and Microsoft Access (*.mdb*) file import via ADO, and distributes it to a text field or a shared memory map.

Data Import supports reading both numbers and text.

To access Excel or Access data using this plug-in, you must first install the Microsoft Access Database Engine 2016 Redistributable.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> VisualDataTools

This page contains the following topics and procedures:

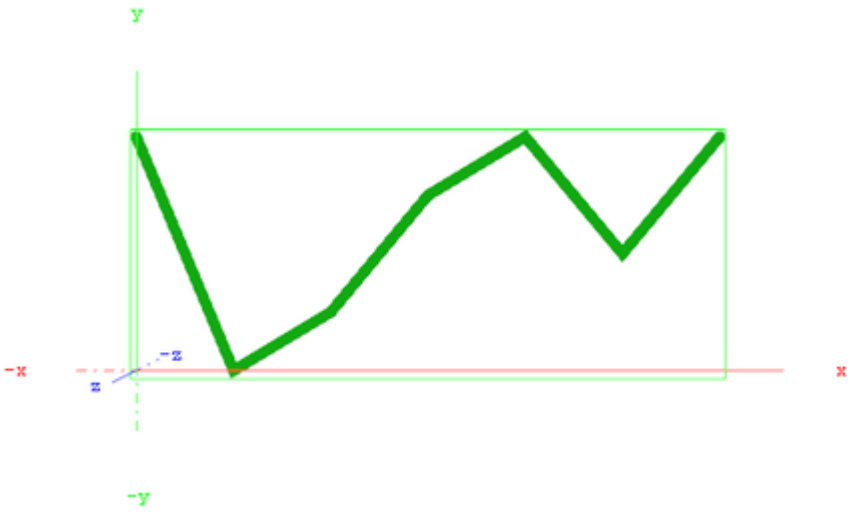
- Data Import Properties
- To Create a Scene with Data Import

Data Import Properties

- **File:** Chooses the required document.
- **Table / Sheet:** Defines the name of the sheet or table.
- **Column(,Col...):** Sets one or more column names to use (comma separated).
- **Area Input:** Uses area information.
- **Column Input:** Uses column information.
- **Row Input:** Uses row information.
- **Data Delim.:** Includes a delimiter after each row.
- **Column Delim.:** Includes a delimiter for column separator sign(s).
- **Shared Mem.:** Sets Scene-, Global- or Distributed-Shared Memory for data transport. Use Inactive memory to not forward any values via Shared Memory.
- **Key:** Determines Shared Memory key name.
- **GetIt:** Reads the document, sends the required data and shows it in the text box.

Copyright © 2026 Vizrt

To Create a Scene with Data Import



This procedure imports data from an Microsoft Excel sheet and feeds Line Chart with data through Shared Memory.

1. Start Microsoft Excel.
2. Enter **ExcelDataY** into cell A1.
3. Add some sample values in the cells below (A2-A8: `80,40,50,70,80,60,80`).
4. Rename this first sheet to **MyTable** (can be done with a double click on the sheet name at the bottom).
5. **Save and close** the Excel document.

IMPORTANT! You must close the document before importing it to Viz Artist.

6. Start Viz Artist and **create a new scene.**

Copyright © 2026 Vizrt



7. Add a group container to the scene tree.
8. Add a material to the group container.
9. Open the Transformation Editor and set **Position X** to `-200.0` and **Position Y** to `-100.0`.
10. Drag and drop the Line Chart and Data Import plug-ins to the same group container.
11. Set **Shared Mem.** to **Scene** with **MyDataY** as Key name for both plug-ins.
12. Set **Data Delim.** to **#** for both plug-ins.
13. For Line Chart, set **ChartWidth** to `500.0`.
14. Activate **DataY Fit**, **DataY Auto Scale** and **DataY Detect Limits.**
15. Adjust **DataY Stop** to `200.0`.
16. Activate **Const. Thickness** to give the chart a constant line width.
17. For Data Import, set the stored Excel file as the **File** parameter.
18. Enter **MyTable** for the Table/Sheet parameter and **ExcelDataY** for Column (,Col…).
19. Click the **GetIt** button and the chart shows.

Copyright © 2026 Vizrt

Data Label



The Data Label plug-in enables graph labeling for all Financial plug-ins.

All containers beneath the current container get labeled with the values from your Shared Memory variables.

Note: This plug-in is located in: Plugins -> Container plug-ins -> VisualDataTools

This page contains the following topics and procedures:

- Data Label Properties
- To Create a Scene with Data Label
- To Create Data Label on VDT Chart
  - For Area Chart, Bar Char, Line Chart and Pie Chart
  - For Scatter Chart
  - For Stock Chart

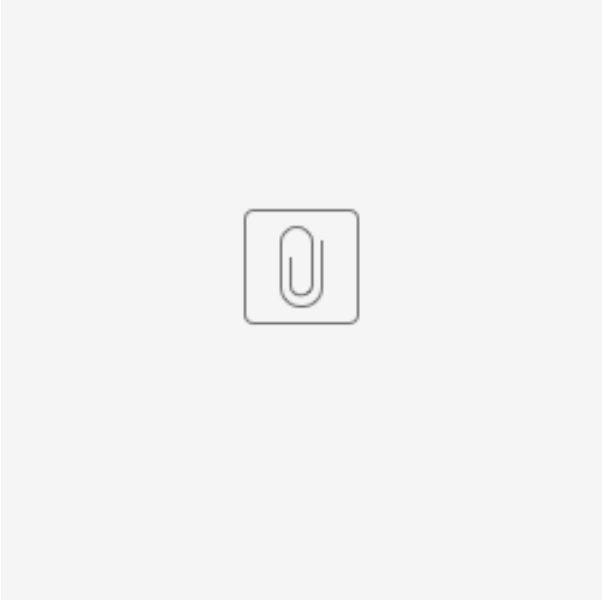
Data Label Properties

- **Shared Mem.:** Changes between **Scene**, **Global** and **Distributed** Shared Memory. Use **Inactive** memory to not forward any values via Shared Memory.
- **Key Data:** Determines Shared Memory key name.
- **Data:** Input Parameter for data.
- **Transfer Mode:** Sets string- or array-based data transfer.
- **Data Delim.:** Defines the value separator sign(s).
- **Label Type:** Sets the label type to numeric values (default) or alphabetic labels (to show weekdays, months, etc.).
- **Data Fit:** Enables data normalization.
  - **Data Threshold:** Adds a definable offset to the detected limit.
- **Data Scale:** Scales input by the selected factor.
- **Data Offset:** Adds an offset to the incoming data.
- **Data Auto Scale:** Enables automatic data normalization.
- **Data Detect Limits:** Detects minimum and maximum of all values and scales them to adjusted Start and Stop.
- **Data Start:** Lower Auto Scale edge.
- **Data Stop:** Upper Auto Scale edge.
- **Nice Numbers:** Rounds to the best fitting next number.
- **Remove Multiples:** Removes multiple values.
- **Sort:** Sorts data items in ascending or descending order.
- **Draw Zero:** Shows or hides zero values.
- **Decimals:** Adjusts decimal digits of all labels.
- **Custom Decimal Symbol:** Enables a custom decimal symbol to be set.
  - **Decimal Symbol:** Specifies the decimal symbol.
- **Prefix:** Adds a label prefix.
- **Unit:** Defines a custom unit for your data values.

Copyright © 2026 Vizrt

- **PosX, Y, Z:** Positions container along the adjusted distance or uses numeric data value input to translate the label containers.
- **OffsetX, Y, Z:** Translates the label container on the current axis.
- **DistanceX, Y, Z:** Sets whole positioning distance for the containers on the current axis.
- **Copy Container:** Adds new containers if there are fewer containers than values.
- **Use Alpha:** Uses input data for alpha scaling.
- **Use Scaling:** Uses input data for alpha scaling.
- **Animate Values:** Lets the current number run from 0 to the defined end value.
- **Relative Length:** If activated, each label has its own 100% (for example, seven labels equal 700%).
- **Const. Speed:** Sets the same animation duration for each label.
- **Total Length%:** Sets the accumulated value of all labels in percent.
- **Alpha max.:** Limits maximum alpha value.
- **Copy Container:** Limits maximum scaling value.

To Create a Scene with Data Label



1. Add an empty container to the scene.
2. Start with an empty scene and drag and drop Data Label and Data Storage plug-ins on a container in the scene-tree.
3. Add a Text child container.
4. Go to the Data Label plug-in, set Shared Mem. to Scene and KeyData to `MyData`.
5. In Data Storage, also set Shared Mem. to Scene, Key Data1 to “MyData” and Data1 to, for example: `80,40,50,70,80,60,80`.
6. Go back to Data Label and activate Copy Container to create multiple instances of the text containers automatically. You see all labels located on the same position now. In the previous examples, the chart plug-in was always responsible for the label positioning. In this example, we want to move that responsibility to Data Label itself.

Copyright © 2026 Vizrt

7. Switch PosX to Range, DistanceX to `500.0` and OffsetX to `-250.0`. We want to position the labels over a certain X, Y or Z range in space.
8. Set PosY to Range, OffsetY to `-75.0` and DistanceY to `150.0`.
9. You can also do a value dependent translation by setting PosX, Y, or Z to Data



10. Set Sort to Ascending to sort the labels. You see that there are labels with the same values. We want to get rid of them and activate *Remove Multiples* for that. You can also try the *Nice Numbers* option now which searches for the next logarithmic style labels. But we continue with deactivated Nice Numbers parameter now.
11. Activate Animate Values and turn down Total Length[%] to `0.0`. Create a Key Frame, set Total Length[%] back to `100.0` and create another Key Frame.
    - This creates an animation where the label values should count up to their final value.
    - Start the animation and see the labels counting and distributing over the specified XY range.

To Create Data Label on VDT Chart

When we put Data Label in the same container of Area Chart, Bar Chart, Line Chart, Pie Chart and Scatter Chart, Pos, Offset and Distance X, Y Z properties of Data Label can be controlled by VDT Chart.

For Area Chart, Bar Char, Line Chart and Pie Chart

1. Go to VDT Chart plug-in and enable Pos. Container property.

Copyright © 2026 Vizrt

2. By default, all position axes, X, Y and Z are enabled with Pos. Container property properties. These properties of Data Label are disabled and be controlled by VDT Chart with enabled Pos. Container property.

For Scatter Chart

1. Add a sub container to the template container of Scatter Chart.
2. Add the font to be used as label to the new sub container and modify to proper size and position, text orientation and another properties.
3. Refresh scatter containers by dropping the template container into Container property of Scatter Chart.
4. Add a Data Label to the same container of Scatter Chart.
5. Update *Data* or *Shared Mem. Key Data*, *Transfer Mode* and *Data Delim* of Data Label to the same as the Scatter Chart plug-in.
6. Click **Resend Data** in Data Storage, if enabled.
7. Pos X, Pos Y and Pos Z properties of Data Label are disabled and are controlled by Scatter Chart.

For Stock Chart

- Follow instructions of To Create a Scene with Data Label and set *Key Data* to the same as *Key Open*, *Key Close*, *Key High* or *Key Low*.
- To set multiple Data Labels, each Data Label has to be in separate container, all of them can be children of the Stock Chart.

Copyright © 2026 Vizrt

Data Storage



The Data Storage plug-in stores data in the current Scene and sends it to the other plug-ins on startup.

Data Storage helps artists during Scene design if there is no data source available, and to hold data within, for example, the Scene.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> VisualDataTools

This page contains the following topics and procedures:

- Data Storage Properties
- To Store and Send Data
- To Enable Data Input from External Sources

Data Storage Properties

- **Shared Mem.:** Changes between **Scene**, **Global** and **Distributed** Shared Memory. Use **Inactive** memory to not forward any values via Shared Memory.
- **Key Data 1-5:** Determines Shared Memory key name.
- **Data 1-5:** Contains (separated) data.
- **Resend Data:** Resends the actual values.

To Store and Send Data

1. Create a chart scene (for example, using Line Chart) and set **Shared Mem.** to **Scene.**
2. Add Data Storage to the scene and set **Shared Mem.** to **Scene.**
3. Open the Data Storage editor and set the following parameters:
    - **Shared Mem.** to **Scene.**
    - **Key Data1** to `Data1.`
    - **Data1** to `10, 20, 15, 30, 25, 40, 20.`
4. Open the chart plug-in editor and set the **Key DataY** value (for example, `Data1`) using the Key Data1 value you set for Data Storage.

> ⚠ **Note:** The various chart plug-ins have different property names (for example, *Data* and *Key Data*). Also, some have X, Y or Z appended at the end, defining the axis.

Copyright © 2026 Vizrt



5. Open the Data Storage editor and click the **Resend Data** button.

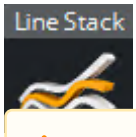
To Enable Data Input from External Sources

The following example builds on the previous example.

1. Add the Control Parameter to the scene tree. Control Object is automatically added.
2. Open the Control Parameter editor.
3. Enter `FUNCTION*DataStorage*Data1` into the **Parameter** field.
4. Set **Data Type** to **Text.** This allows you to pass a string of numbers to Control Parameter that are stored in the shared memory. It also allows your template designer or operator to further extend/enhance the input methods (for example, through scripting).

Copyright © 2026 Vizrt

Line Stack



The Line Stack plug-in creates a stacked chart consisting of several Line Charts.

Note: This plug-in is located in: Plugins -> Container plug-ins -> VisualDataTools

This page contains the following topics and procedures:

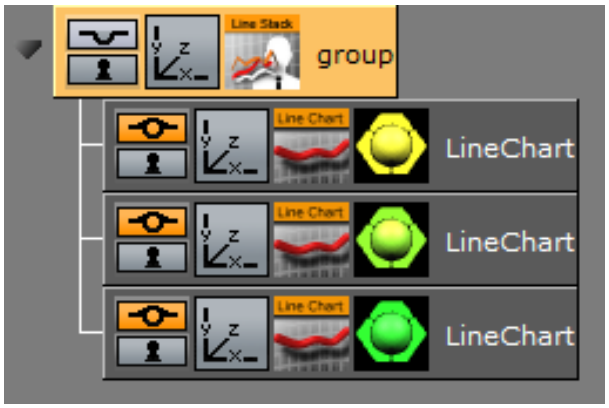
- Line Stack Properties
- To Create a Scene with Line Stack

Line Stack Properties

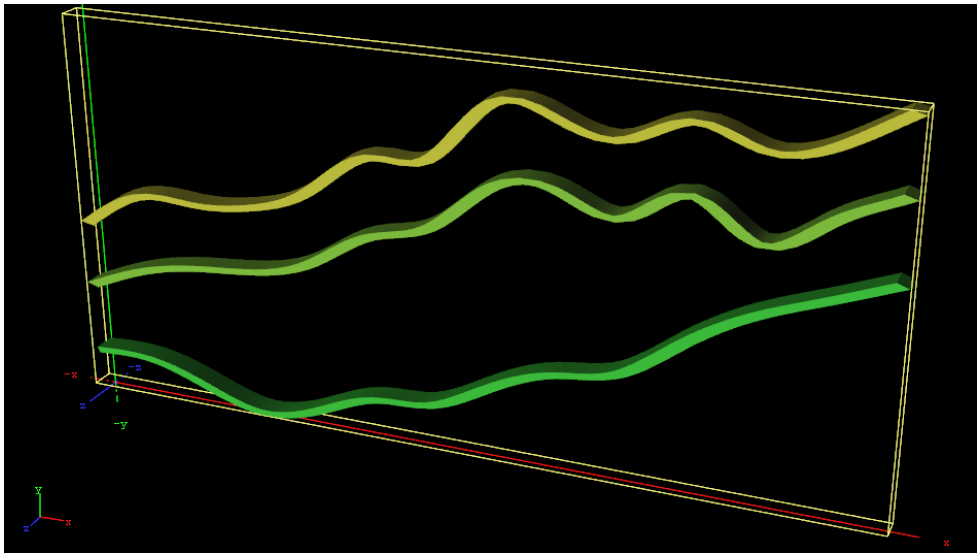
- **Initialize stacked chart:** Refreshes the stacked chart, based on Line Chart plug-ins in the sub-containers. The stacked chart detects when a chart is added/removed from the stack, and updates the chart accordingly. However, some actions, such as re-ordering the charts within a stack, are not updated automatically. In these cases, press **Initialize stacked chart** to update the stack.
- **Stack Gap:** Adds space between each chart.

Note: The remaining properties in the Line Stack plug-in are the same as used in Line Chart. Line Chart features Specify X Values, DataY Compare, and Bevel, are disabled in stacked charts.

To Create a Scene with Line Stack



Copyright © 2026 Vizrt



1. Create a new container.
2. Add a Line Stack plug-in into this container.
3. Add two or more Line Chart plug-ins to become children of this container.
4. Add Data Y into each chart. They should have the same number of nodes.
5. In the settings of the Line Stack plug-in, press **Initialize Stacked Chart** to refresh the chart.
6. Use the Line Stack plug-in to set the chart parameters that are common for all charts in the stack.

Copyright © 2026 Vizrt

2.4.22 Text Container Plugins

Format Number

Location: Plugins → Container Plug-ins → Text **Compatibility:** V3 Classic, V4 PBR **Category:** Text

> Note: This plugin is inactive by default and should only be used for compatibility reasons

Overview



The Format Number plug-in formats a numeric or date string displayed in a **Text geometry** container. It takes the raw text value, parses it as a number or date, and rewrites the text with the chosen formatting (including decimal places, thousand separators, currency symbols, percentage notation, and date layouts).

Negative values can optionally be displayed in a user-defined color for visual emphasis (for example, red for losses in financial graphics).

Format Modes

The plug-in provides four formatting modes, selectable via the **Format** parameter:

Mode	Description	Example
Number	Plain numeric formatting with configurable decimal places, fraction types, and thousand separators.	`1234.5` → `1,234.50`
Percentage	Multiplies the value by 100 and appends `%`. Uses the same numeric formatting options.	`0.75` → `75%`
Currency	Prepends or appends a currency symbol with configurable negative notation.	`-1234` → `(EUR 1,234)`
Date	Parses a date string and reassembles it using the chosen field order, separators, month/year/day styles, and optional weekday.	`07.03.2025` → `March 7, 2025`

Parameters

General

Parameter	Default	Description
Text	*empty*	The input text to format. Typically bound to the container's text content.
Format	None	Formatting mode: `None`, `Number`, `Percentage`, `Currency`, `Date`.

Copyright © 2026 Vizrt

Number / Percentage / Currency Options

Parameter	Default	Description
Negative numbers	`-123`	Display style for negative values: `-123` or `(123)`.
Negative currency	`-EUR 123`	Negative notation in Currency mode: `-EUR 123`, `EUR -123`, `(EUR 123)`, or `EUR (123)`.
Negative Color	White	Per-character color applied when the value is negative.
Fraction Type	Decimal	Decimal display or fractional notation: `Decimal`, `1/2`, `2/4`, `4/8`, `8/16`, `5/10`, `50/100`.
Decimal places	0	Number of decimal digits (0–6). Only visible in Decimal fraction type.
Decimal separator	Point	Character for the decimal point: `.` or `,`.
Replace Zero	Off	When enabled, replaces zero values with a custom string.
Replace Zero by	`–`	The replacement string used when the value is zero.
Use Thousand separator	Off	Enable thousand grouping (e.g., `1,234,567`).
Thousand separator	Comma	Character for thousand grouping: `.` or `,`.
Currency	`EUR`	The currency symbol or text. Only visible in Currency mode.

Date Options

Parameter	Default	Description
Date Input	European	Input parsing order: European (`dd.mm.yyyy`) or American (`mm/dd/yyyy`).

Copyright © 2026 Vizrt

Parameter	Default	Description
Date	DD/MM	Output field arrangement: `DD/MM`, `MM/DD`, `DD/MM/YY`, `MM/DD/YY`, `MM/YY`, `YY/MM`, or `User` (custom format string).
Format	`WW the DD on MM year YY`	Custom format string (User mode). Tokens: `W` = weekday, `D` = day, `M` = month, `Y` = year.
Year	`2009`	Short (`09`) or full (`2009`) year.
Month	`03`	Month display: `03`, `3`, `Mar`, `March`, or `M`.
Day	`07`	Day display: zero-padded (`07`) or plain (`7`).
Day In Week	None	Optional weekday prefix: `None`, long (`Tuesday`), or short (`Tue`).
Month Separator	`/`	Separator between month and year: `/`, `-`, `,`, `.`, or space.
Day Separator	`/`	Separator between day and month.
Week Day Separator	`/`	Separator after the weekday name.

Usage Example

Formatting election results

1. Create a Text geometry container with the raw value (for example, `12345.678`).
2. Add the **Format Number** plug-in to the container's function stack.
3. Set **Format** to `Number`.
4. Set **Decimal places** to `1`, enable **Use Thousand separator**.
5. Result: `12,345.7`.

Copyright © 2026 Vizrt

Currency display with negative coloring

1. Set **Format** to `Currency`, **Currency** to `$`.
2. Set **Negative currency** to `($ 123)`.
3. Set **Negative Color** to red.
4. Input `-500` → displays as `($ 500)` in red.

Technical Notes

- The plug-in writes directly to the Text geometry on the same container. It overwrites the displayed text each frame when the input changes.
- Parameters that are not relevant to the current format mode are automatically hidden in the GUI (for example, date options are hidden in Number mode).
- Fractional display modes (½, ¼, etc.) use Unicode fraction characters when available.

Copyright © 2026 Vizrt

2.4.23 VizMaps Plug-ins

The following Container plug-ins are located in the Container folder:

- Geo Camera
- Geo Camera Link
- Geo Canvas
- Geo Expression
- Geo Navigator

Copyright © 2026 Vizrt

Geo Camera



What it is. GeoCamera is how you point at the world. Rather than positioning a camera in 3D space and working out where that lands on the map, you say it directly: *this* place, *this* close, turned *this* way, tilted *this* much. It is what makes "fly from Berlin to Cairo" a matter of two coordinates and a zoom instead of a hand-keyframed camera move, and every map animation is built on it. It drives the view, and anything anchored to the map moves with it.

It lives on the same container as the GeoCanvas and is added automatically by Viz Artist.

The four you will actually touch are Latitude, Longitude, Zoom and Bearing. The rest are set once and left alone.

See chapter Working with Maps for detailed information

Parameters

Parameter	What it is for
Latitude / Longitude	The point on Earth at the centre of frame. This is your "where" — animate these two and the map travels. Latitude also quietly changes how much ground a zoom level covers, which is why the same zoom feels much closer in Norway than in Kenya.
Zoom	How close you are. Each whole step doubles the scale: zoom 4 is a continent, 10 a city, 16 a few streets, 20 a building. Fractional values are fine and are what make a smooth push-in possible.
Bearing	Which way is up, in degrees. 0 is north-up. Rotate it to follow a route, to line a coastline up with your lower third, or simply because the story reads better at an angle.
Pitch	How far the camera tilts from looking straight down. 0 is a flat overhead map; increase it and the view leans into perspective, with the horizon appearing at high values. This is what turns a map into a landscape shot.
Projection	How the round Earth is flattened. **Mercator** is the familiar web-map look — shapes stay true locally, but areas stretch badly toward the poles (the reason Greenland looks continental). **Equirectangular** keeps latitude and longitude on an even grid, which suits full-world graphics where the polar distortion of Mercator would be distracting. Pick one at design time; it is not something to animate.
Resolution	The frame size the camera reckons against, in pixels. Worth understanding rather than ignoring: zoom is defined *relative* to this, so the same zoom on a different resolution shows a different amount of ground. Keep it matched to your output format unless you have a specific reason not to.

Copyright © 2026 Vizrt

Parameter	What it is for
View bounds	Read-only. The rectangle of the world currently in frame, as west/south/east/north. Useful for checking what is actually visible — whether the region you care about is fully in shot before you commit to a zoom.

Copyright © 2026 Vizrt

Geo Camera Link



What it is
Geo Camera Link links the camara properties of one GeoCanvas to anotherone.
This is needed to build locators and similar objects.

See Adding a Locator for details

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

Geo Canvas



What it is.
GeoCanvas is the map itself. Put it on a container and that container becomes a piece of the world: it fetches map tiles from the Viz Maps backend, assembles them into the basemap you see, and — crucially — establishes the geographic frame that everything else hangs off. Child containers positioned by latitude and longitude, GeoJSON shapes, GeoText labels and the GeoCamera all resolve against the GeoCanvas above them. Without one, there is no map and nothing geographic to anchor to.

See chapter Working with Maps for detailed information

Most scenes need to set exactly one thing: the style. Everything else is tuning for on-air reliability.

Parameters

Parameter	What it is for
Style	The map's appearance — the whole cartographic design: which layers are drawn, in what colours, with what labels. Takes a style reference from the Maps backend (`vizmaps://api/v1/styles/<id>`). This is the one parameter almost every scene sets, and switching it is how you go from a satellite look to a clean political map.
Style ▸ Reload	Re-fetches the current style from the backend without touching anything else. Use it when a designer has published a change to a style you are already pointing at, instead of reassigning the style.
Style ▸ Use reference labels	Draws the style's reference labels — place names and similar text baked into the style. Turn it off when you are supplying your own labelling (GeoText, or a graphic layer) and don't want the style's text competing with it.
Source mask	Selects which of the style's data sources are actually drawn, as a bitmask. Lets one style serve several looks — mask off a source to drop, say, roads or boundaries for a particular scene without authoring a second style.
Mask ▸ Bit	Sets a single bit of that mask on or off by index, rather than writing the whole mask value. The convenient form when you are toggling one source.
Tile prefetch ▸ Rings	How many rings of tiles to load *around* the visible area. Higher values mean a pan has tiles already in hand instead of loading them mid-move. Costs memory and bandwidth.
Tile prefetch ▸ Child levels	How many zoom levels *below* the current one to fetch ahead. This is what makes zooming **in** stay sharp instead of showing a blurred parent tile while it catches up.

Copyright © 2026 Vizrt

Parameter	What it is for
Tile prefetch ▸ Parent levels	How many zoom levels *above* the current one to fetch ahead — the same idea for zooming **out**.
Tile prefetch ▸ World cover	Fetches enough low-zoom tiles to cover the whole globe, so a fully zoomed-out view is never empty. Worth turning on for a scene that starts wide or cuts to a world shot.
Tile ready timeout	How long to wait for tiles before the canvas reports itself ready anyway. `0` means wait indefinitely. This is the on-air safety valve: it decides whether a slow backend delays your take or lets it go with an incomplete map.
Ready	Read-only. `1` once the tiles for the current view have arrived. This is what you poll before taking the scene to air or grabbing a snapshot — reading it is the difference between capturing a finished map and a half-loaded one.
Active	Whether the canvas draws at all. Switches the basemap off while leaving the geographic frame — and everything anchored to it — in place.

Copyright © 2026 Vizrt

Geo Expression



What it is. GeoExpression links the map's zoom to a value somewhere else in your scene. As the camera flies in or out, something changes with it — a border line thins as you approach, a label fades in only once you are close enough, a readout counts down. You draw a curve that says "at this zoom, that value", point it at what you want driven, and it runs continuously from then on.

The pattern is always the same three decisions: the **curve** (what value at what zoom), the **range** it is allowed to produce, and the **target** it lands on. It sits on the container carrying whatever it drives.

The reason it exists is that hand-animating these relationships doesn't survive a live camera. An operator flying the map freely will stop anywhere, and a keyframed animation only looks right if they stop where you expected. A curve against zoom is correct at every point in between.

See chapter Working with Maps for detailed information

Parameters

Parameter	What it is for
Target	What kind of thing the output drives. **Alpha** fades the container — the simplest way to have detail appear only at close zooms. **Line Node Thickness** drives the width of a GeoJSON line, which is the classic case: borders that stay readable whether you are looking at a continent or a county. **Plugin Parameter** drives any parameter of another plugin on the same container. **Script Parameter** hands the value to a Viz script, for anything the other three can't express.
Plugin Name	Which plugin on the container to drive. Only relevant for the Plugin Parameter target — the other three know their own destination.
Plugin Parameter Name	Which parameter of that plugin receives the value. For a Script Parameter target this is the name the script registered, and it's the only field needed since the script is found on the container itself.
Payload	The curve — the heart of it. A set of points, each pairing a zoom level with the value you want at that zoom. Two points give a simple ramp; more let you hold a value across a range and then change quickly, which is usually what looks right on air.
Interpolation	How values between your curve points are found. **Linear** blends smoothly and is what you want for anything continuous — thickness, alpha, size. **Step** holds each value until the next point is reached, for things that should switch rather than slide, like a label that is either on or off.

Copyright © 2026 Vizrt

Parameter	What it is for
Min Value / Max Value	The bounds the output is allowed to take. These are your guard rails: they stop an extreme zoom from producing a hairline too thin to see or a line thick enough to swamp the map. Set them to the range you are happy to broadcast, not the widest the target accepts.
Current Zoom Level	The zoom the expression is working from. Mainly useful while designing — set it by hand to preview what the curve does at a given zoom without flying the camera there and back.
Current Output Value	The value the curve is producing right now. The field to watch when something isn't behaving: if this moves as expected but the target doesn't change, the problem is the target wiring, not the curve.
Status	Feedback from the plugin about its current state. Worth a glance if the expression isn't doing anything.

Building one

Set the target first, then the range, then draw the curve — in that order, because the range decides what the curve's values mean. Preview by typing zoom levels into Current Zoom Level rather than flying the camera; it is far quicker, and it lets you check the extremes deliberately instead of hoping you happen to pass through them.



The output is only ever a single number, so driving several parameters means several GeoExpressions; and Current Output Value is the first place to look when debugging, since it cleanly separates "the curve is wrong" from "the curve is fine but isn't reaching its target".

Copyright © 2026 Vizrt

Geo Navigator



What it is. GeoNavigator builds camera flights across the map. You collect a list of places — a row per destination — and it generates the animation between them on a director. A three-stop tour becomes three rows plus a button press, rather than a hand-keyframed camera move that has to be redone every time the running order changes.

Its own description puts it as *"geo camera navigator with payload row editing, optional camera row apply, and director keyframe animation generation."* In practice the workflow is: fly the camera somewhere you like, capture it as a row, repeat, then generate the animation.

There is a companion **ControlGeoNavigator** that exposes the flight to Trio or Pilot, so an operator can pick destinations without opening the scene.

See chapter Working with Maps for detailed information

Flight settings

Setting	What it is for
Default Motion	The shape of the movement between two places. **Linear** moves at a constant rate. **Sine** eases in and out, which reads better on air for short hops. **Van Wijk** is the one to reach for on long flights — it automatically pulls back and pushes in again so the journey stays legible instead of skimming the ground at high speed. **Great Circle** follows the true shortest path over the globe, which is what you want when the flight itself is the story, like a route between two cities.
Default Duration	How long a flight takes, in the units of your director. Used when timing is authored rather than computed.
Timing Mode	**Authored** uses the duration you set, so every hop takes the same time whether it crosses a city or an ocean. **Auto** derives the duration from the distance, so long flights get proportionally longer. Auto usually feels more natural; Authored is what you want when the flight has to fit a fixed slot.
Default Zoom Lift	How far the camera pulls back at the midpoint of a flight before descending again. A little lift makes a hop feel like travel rather than a slide; too much makes short hops feel melodramatic.

Copyright © 2026 Vizrt

Setting	What it is for
Default Flight Direction	Which way round the globe. **Shortest** picks the shorter of the two, and is right nearly always. **East** and **West** force the direction — the reason they exist is flights near the date line, where the shortest path may not be the one that tells the story.
Director	Which director the generated animation is written to.
Hop Name Prefix	The naming used for the generated hops, so what lands in the director is identifiable later rather than a row of anonymous keys.
Progress	Where you are along the flight, from start to end. This is the value the director animates — and the one to scrub when checking what the middle of a flight actually looks like.
Prefetch Step (frames)	How often along the flight to fetch map tiles ahead of the camera. This is what stops a flight arriving somewhere the map hasn't loaded yet. Denser stepping costs more memory and bandwidth.
Target Row	Which row in the list the actions below apply to.
Search Location	A place to look up and fly to, rather than entering coordinates by hand.
Payload	The list of destinations itself — the flight plan. Normally built with the buttons below rather than edited directly.
Status	Feedback from the plugin. Worth a glance when a flight isn't behaving.

Copyright © 2026 Vizrt

Actions

Button	What it does
Add Row from Camera	Captures where the camera is now as a new destination. The main way a flight gets built: fly, capture, repeat.
Replace Row from Camera	Overwrites the target row with the current camera. How you adjust one stop without rebuilding the flight.
Activate Row	Jumps the camera to a row, to check a stop looks right.
Clear Rows	Empties the flight plan.
Apply Search Location	Takes the place from Search Location and applies it.
Update Animation	Generates the director keyframes from the current rows and settings. **This is the one to remember** — changing rows or settings does not rewrite the animation on its own.
Auto Jump	Moves the camera to a row automatically as you work.
Prefetch Hops / **Prefetch Animation**	Pre-loads map tiles along the destinations or along the generated flight, so the first playback is not also the first tile fetch. Worth running before anything goes to air.

In practice

Build the flight first — capture your stops with the camera and check each with Activate Row — then set motion, timing and lift, then press **Update Animation**. That last step is the one people forget: the rows and the animation are separate things, and the director keeps whatever was generated last until you regenerate it.

Prefetch before air. A flight that looks perfect on the second run and ragged on the first is almost always tiles arriving late, not the animation.

Copyright © 2026 Vizrt

2.5 Shaders (Classic)

The default path for the Shader plug-ins is: *<viz install folder>\plug-in*.

Tip: Most shaders also work on video clips.

- Default Shaders
- Effect Shaders
- Filter Shaders
- Material Shaders
- RealFX Shaders
- Texture Shaders
- RTT Advanced Materials Shaders

Copyright © 2026 Vizrt

2.5.1 Default Shaders

This folder is empty and reserved for custom Shader plug-ins.

Copyright © 2026 Vizrt

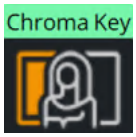
2.5.2 Effect Shaders

The following Shader plug-ins are located in the Effects folder:

- Chroma Keyer
- Fluid
- Frame Mask
- Image Mask
- LED Panel
- Soft Mask
- Water Shader

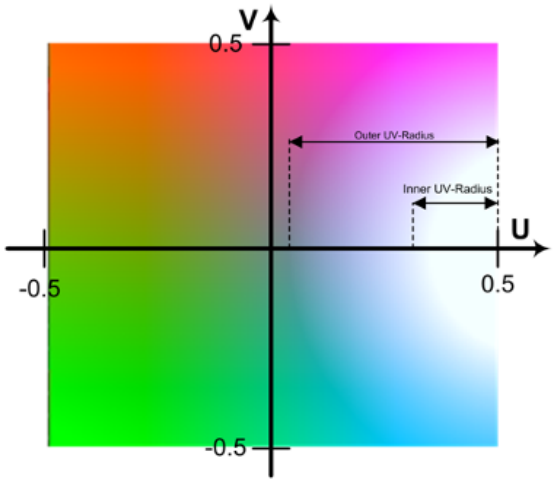
Copyright © 2026 Vizrt

Chroma Keyer



The Chroma Keyer plug-in clears regions in a video defined by a color.

The color-space of this Shader is YUV, which is defined as following:



The luminance (Y) component determines the brightness of the color, while the U and V components determine the color itself (the chroma). Luminance (Y) ranges from `0.0` (dark) to `1.0` (bright). U and V ranges from `-0.5` to `0.5`.

> **Note:** This plug-in is located in: Plugins -> Shader -> Effects

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Chroma Keyer Properties

- **Video:** Sets the input video channel for keying. Be sure that for the set video channel the Keying Mode has been set to M-Zone.
- **Blend:** Sets the alpha for the key signal.
- **Fit:** Alters size to fit Screen or Object.
- **Invert:** Flips the key signal.
- **Sampling Filter:**
  - **Nearest:** Takes color information from one pixel.
  - **1D-Filter:** Takes color information after applying a 1D-linear filter.
  - **2D-Filter:** Takes color information after applying a 2D-linear filter.

Copyright © 2026 Vizrt

- **Luminance:** Sets the luminance.

Basic parameters for first method:

- **HueAngle:** Selects the color to key in YUV space.
- **OpeningAngle:** Determines tolerance for the color in YUV space.
- **KeyGain:** Determines gradient for borders to neighboring colors (`1` =soft, `10` =sharp).
- **SatMin:** Determines lower saturation limit for keyed colors.
- **SatMinGrad:** Determines gradient for lower saturation limit (`0` =sharp, `1` =soft).
- **SatMax:** Determines upper saturation limit for keyed colors.
- **SatMaxGrad:** Determines gradient for upper saturation limit (`0` =sharp, `1` =soft).

Basic parameters for second method:

- **U-Color:** Determines U-value for color keying.
- **V-Color:** Determines V-value for color keying.
- **UV-Diameter:** Removes all colors between the reference UV and this diameter.
- **UVGradient:** Determines gradient for fade out at diameter border (`0` =sharp, `1` =soft).

Removing keying from Highlights and Shadows:

- **LumMin:** Sets lower luminance limit for keyed colors.
- **LumMinGrad:** Sets gradient for lower luminance limit (`0` =sharp, `1` =soft).
- **LumMax:** Sets upper luminance limit for keyed colors.
- **LumMaxGrad:** Sets gradient for upper luminance limit (`0` =sharp, `1` =soft).

Sampling reference point:

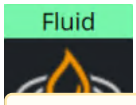
- **UVRangeCheck:** Obtains information about the sampled color.
- **URangeMin:** Defines the minimum UColor that should be accepted.
- **URangeMax:** Defines the minimum UColor that should be accepted.
- **VRangeMin:** Defines the minimum UColor that should be accepted.
- **VRangeMax:** Defines the minimum UColor that should be accepted.
- **SampleTrustiness:** Determines percentage of samples in the limited UV-Color space.
- **Pos. X:** Deternines X-coordinate for sampling.
- **Pos. Y:** Deternines Y-coordinate for sampling.

> ⚠ **Note:** The origin of the coordinate system is in the upper left corner.

- **Width:** Sets number of pixels in x-direction.
- **Height:** Sets number of pixels in y-direction.
- **Sample at Pos:** Replaces XY-coordinates of the object center if Pos. X and Pos. Y are enabled. If disabled, the XY-coordinates are set by values in the interface.
- **Get Sample:** Samples another time the reference point.

Copyright © 2026 Vizrt

Fluid



The Fluid plug-in provides a geometry mesh with some elasticity during animation.

Note: This plug-in is located in: Plugins -> Shader -> Effects

Copyright © 2026 Vizrt

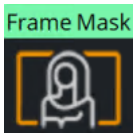
Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	❌

Fluid Properties

- **Inertia:** Sets the inertia of the motion.
- **Strength:** Sets the strength of the motion.

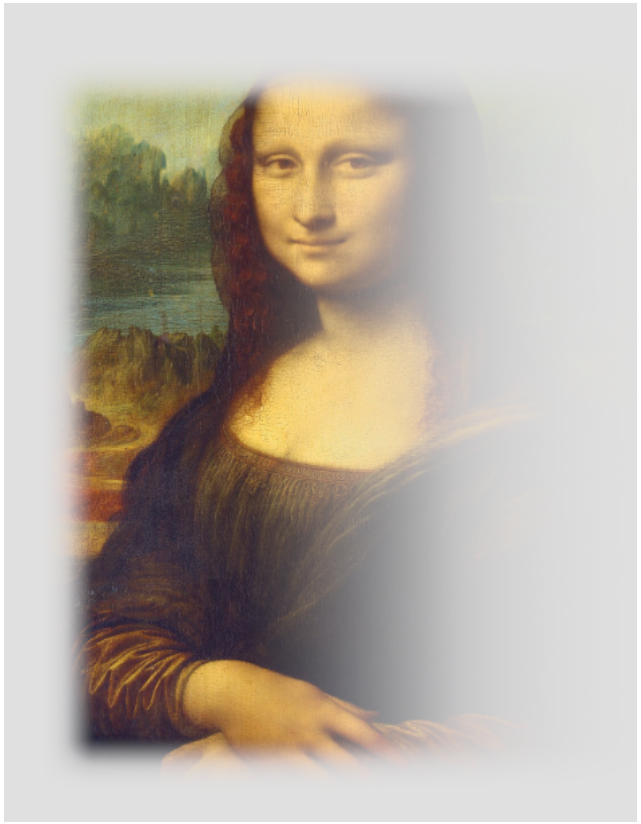
Copyright © 2026 Vizrt

Frame Mask



The Frame Mask plug-in adds a framed mask around an object (e.g. image or geometry). In the properties, the aspect ratio can also be set.

Note: This plug-in is located in: Plugins -> Shader -> Effects



Upgrade information: This plug-in needs to be re-applied when converting from existing Classic Render Pipeline scenes to the Viz Engine Render Pipeline.

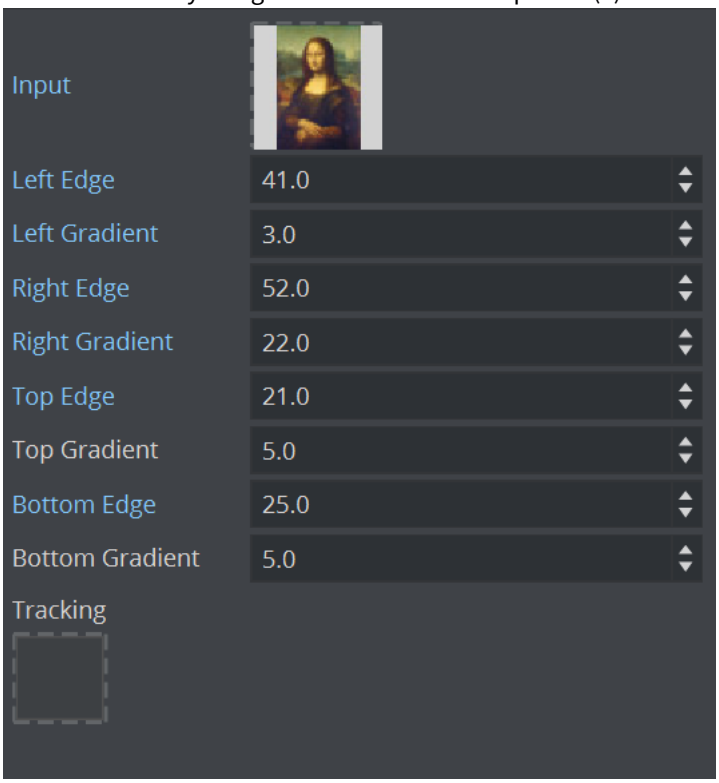
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✓

Frame Mask Properties

- **Target (VER only)**: The image the effect should be applied on.

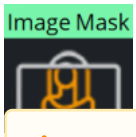
Copyright © 2026 Vizrt

- **Left, Right, Top and Bottom Edge:** Sets the tracked borders (see also **Tracking**).
- **Left, Right, Top and Bottom Gradient:** Sets the gradient from the edge of the frame to the edge of the object.
- **Aspect Ratio (Classic Only):** Sets the aspect ratio (16:9, 4:3 or Custom), if required (default is `Off`).
    - **Custom:** Uses to set a custom aspect ratio (for example, 1.85:1 or 2.39:1).
- **Tracking:** Uses a Container to track the location of the actual frame mask instead of animating all tracked borders manually. Drag a Container to the drop zone (1):



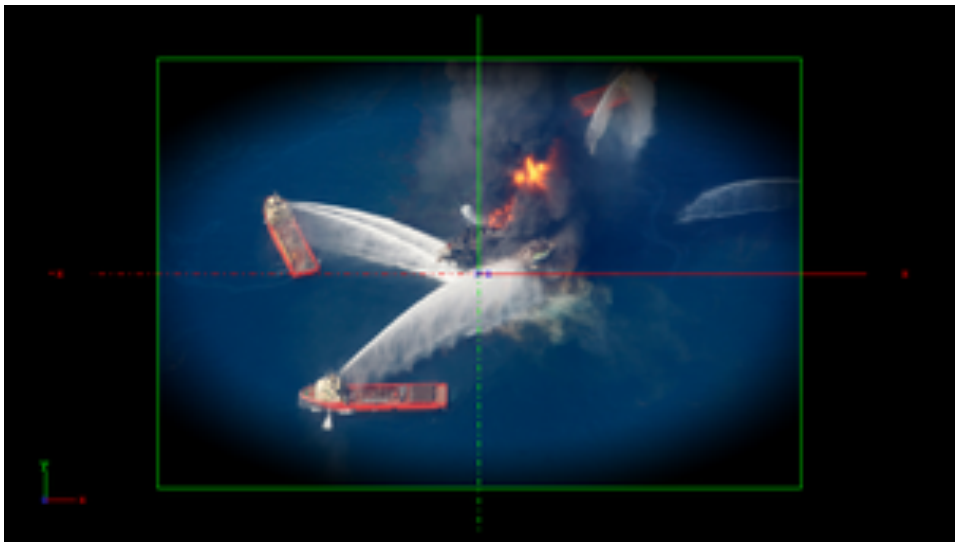
Copyright © 2026 Vizrt

Image Mask



The Image Mask plug-in applies a mask to an image using other images (for example, an alpha image).

Note: This plug-in is located in: Plugins -> Shader -> Effects



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Image Mask Properties

- **Image:** Placeholder for image.
- **Blend:** Blends the image with the object.
- **Position X, Y %:** Sets the position of the mask in percentage for the screen or object.
- **Scale X, Y %:** Sets the size of the mask in percentage for the screen or object.
- **Fit:** Selects the size of the image mask to the screen or the object:
  - **Screen:** Sizes the image mask to the screen. If the object is moved, it moves while the image mask stays fixed.
  - **Object:** Sizes the image mask to the object. If the object is moved, the image mask moves with it.
- **Wrap:**
  - **Repeat:** Applies the image mask multiple times on the screen or object.
  - **Clamp:** Applies the image mask once on the screen or object.
- **Mask Aspect:** Sets the aspect of the image mask to the screen or the object.

Copyright © 2026 Vizrt

- **Invert:** Inverts the image.
- **Alpha only:** Enables the alpha channel for the image (for example, an RGBA image).
- **Soft Mask:** Applies a soft mask (available when **Fit** is set to **Screen**).

Copyright © 2026 Vizrt

LED Panel



The LED Panel plug-in creates an LED panel style texture.

> ⚠ **Note:** This plug-in is located in: Plugins -> Shader -> Effects

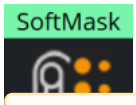
Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	❌

LED Panel Properties

- **Pixel Size:** Determines the size of pixels in the LED board.
- **Billboard Size X:** Determines the width of the LED board (for example, using a *Pixel Size* of `1` and a *Billboard Size* of `10` shows ten LEDs).
- **Billboard Size Y:** Determines the height of the LED board.
- **Tolerance:** Threshold that defines which pixels belong to an LED.
- **Pixel Radius:** Sets the maximum size/radius of the LED.
- **Luminance Steps:** Determines number of discrete luminance steps the LED panel is able to show.
- **Luminance Boost:** Determines factor to increase luminance.
- **Color Boost:** Enhances the RGB values of the image.
- **Noise:** Defines texture where LEDs are already burnt out or damaged (white is interpreted as functional, black is treated as damaged).
- **Wrap:** Sets whether to use texture repeat or clamping mode.
- **Burnt Out Percent:** Determines the percentage of pixels that appear to be broken or burnt out.
- **Background Color:** Determines the background color of the LED board.

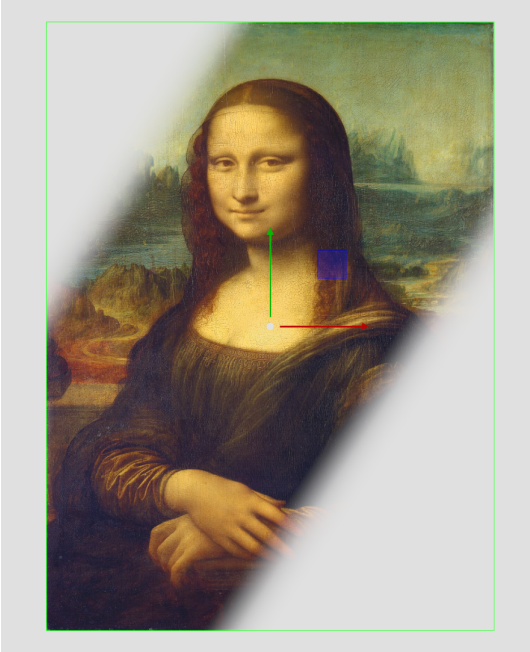
Copyright © 2026 Vizrt

Soft Mask



The Soft Mask plug-in applies a soft mask to an object (for example, image or geometry).

Note: This plug-in is located in: Plugins -> Shader -> Effects



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	⊖ (Use Textureslot UV Space functionality instead)

Upgrade information: This plug-in needs to be re-applied when converting from existing Classic Render Pipeline scenes to the Viz Engine Render Pipeline.

Soft Mask Properties

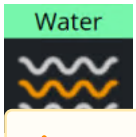
- **Input** (VER pipeline only): The source image the affect should be applied on.
- **Progress:** Moves the mask perpendicular to the mask direction. Default is `0.0` and is based on the configured center position.
- **Rotation:** Rotates the mask around the center.
- **Keep Aspect** (Classic only)**:**

Copyright © 2026 Vizrt

- **Center X and Y:** Defines the X and Y position of the center of the mask.
- **Gradient Width:** Sets the size of the gradient.
- **Invert:** Inverts the mask.
- **Mirror Gradient:** Mirrors the mask on both axis.
- **Blank:** Offsets the area that should not be affected by the mask.
- **Tracking:** Allows the progress to be controlled by a tracked Container to build the progress animation easier. Drag a Container to the drop zone.

Copyright © 2026 Vizrt

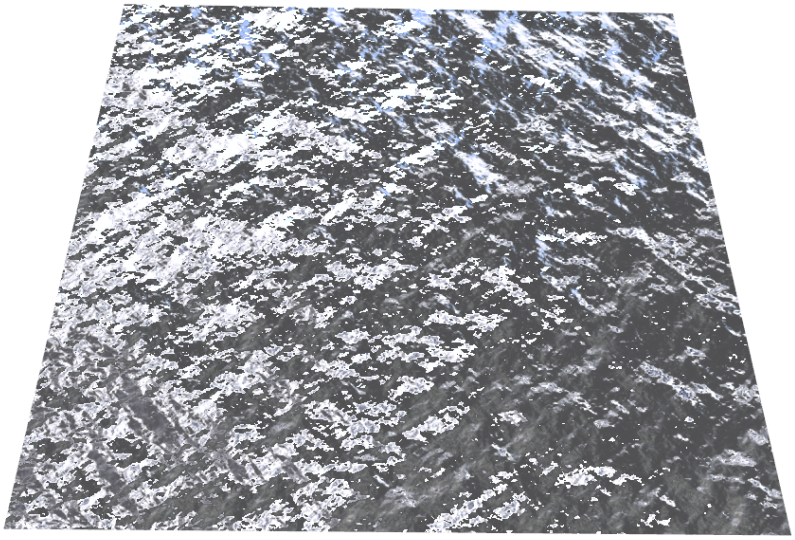
Water Shader



The Water Shader plug-in simulates a water surface.

Note: This plug-in is located in: Plugins -> Shader -> Effects





Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Water Shader Properties

- **Bump Height:** Sets the height of the bump map.
- **Bump Speed:** Moves the bump map of the surface. Useful to simulate a current.
- **Bump Scale:** Scales the bump map.
- **Wave Frequency:** Sets the frequency of the wave.
- **Wave Amplitude:** Sets the amplitude of the wave.
- **Wave Speed:** Defines the big waves of the water. You need geometry with many vertices to get smooth waves.
- **Environment Map:** Defines the texture which is reflected in the water. If no texture is applied, the shader uses a default texture. The size of the environment map should be kept low (512x512 max), because it can consume a lot of texture memory.
- **Deep Water:** Sets the color of deep water.

Copyright © 2026 Vizrt

- **Shallow Water:** Sets the color of shallow water.
- **Reflection:** Sets the color of the reflections.

> **Note:** The blending of deep and shallow water depends on the angle of the camera and how it looks at the water.

- **Water Amount:** Controls how the deep and shallow color affects the look of the water.
- **Reflection Amount, Bias, Power and Multiplier:** Controls the reflection.
- **Blend Texture:** Allows you to blend the texture of the container into the water.

Copyright © 2026 Vizrt

2.5.3 Filter Shaders

The following Shader plug-ins are located in the Filter folder:

- Blend Image
- Blur
- Color Balance
- Radial Blur
- Sepia
- Sharpen

Copyright © 2026 Vizrt

Blend Image



The Blend Image plug-in blends two images in different ways.



Note: This plug-in is located in: Plugins -> Shader -> Filter

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

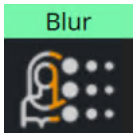
Copyright © 2026 Vizrt

Blend Image Properties

- **Image:** Image placeholder. Reset removes the image.
- **Mode:** Available blend modes are:
    - **Blend:** Ordinary picture blending.
    - **Darken:** Uses the base or blend color, whichever is darker, as the destination color.
    - **Lighten:** Uses the base or blend color, whichever is lighter, as the destination color.
    - **Multiply:** Multiplies the base color with the blend color. The result color is always a darker color.
    - **Screen:** Multiplies the inverse of the base and blend color. The result color is always a lighter color.
    - **Color Burn:** Darkens the base color to reflect the blend color by increasing the contrast of the base and blend color.
    - **Color Dodge:** Brightens the base color to reflect the blend color by decreasing the contrast of the base and blend color.
    - **Overlay:** Screens or multiplies the colors depending of the base color.
    - **Soft Light:** Darkens or lightens the colors, depending on the blend color.
    - **Hard Light:** Multiplies or screens the colors, depending on the blend color.
    - **Add:** Brightens the base color to reflect the blend color by increasing the brightness.
    - **Subtract:** Subtracts the base and blend colors.
    - **Inverse Subtract:** Same as subtract but inverses the base and blend colors.
    - **Difference:** Subtracts either the base color from the blend color or the blend color from the base color depending on which color has the greater brightness value.
    - **Inverse Difference:** Same as difference but inverses the base and blend color.
    - **Exclusion:** Similar to the **Difference** mode, but lower in contrast.
- **Blend:** Blends the image with the object.
- **Position X and Y %:** Positions the image on the object.
- **Scale X and Y %:** Scales the image.
- **Wrap:** Wraps the object with the image. Available options are:
    - **Repeat:** Repeats the image.
    - **Clamp:** Clamps the image (no repeat).

Copyright © 2026 Vizrt

Blur



The Blur plug-in blurs an object with an image texture and/or a material on it.

Values can be animated. You also have a blur plug-in directly on the image editor, but that only works on one image, using the shader all images within a group can be blurred.



The samples above depict the same image with blur quality set to Low, Normal and High with a range of 100.0. The last is the original (reference) with no blur filter.

> **Note:** This plug-in is located in: Plugins -> Shader -> Filter

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Blur Properties

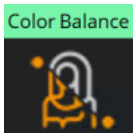
- **Quality:** Sets the quality of the blur filter. Low is more pixelated, whereas high is smooth. Available options are Low, Normal and High.
- **Range:** Sets the range of the blur. Range is from `0.0` to `100.0`.

See Also

- Radial Blur

Copyright © 2026 Vizrt

Color Balance



The Color Balance plug-in determines brightness, contrast, saturation and hue.



Note: This plug-in is located in: Plugins -> Shader -> Filter

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

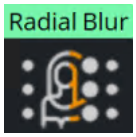
Copyright © 2026 Vizrt

Color Balance Properties

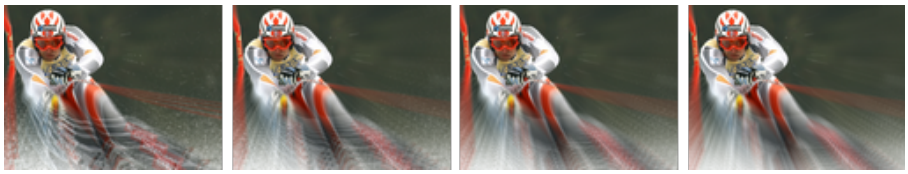
- **Brightness:** Sets the brightness of the object. Range is `0.0` to `1000.0`.
- **Contrast:** Sets the contrast of the object. Range is `-1000.0` to `1000.0`.
- **Saturation:** Sets the saturation of the object. Range is `-1000.0` to `1000.0`.
- **Hue:** Sets the hue of the object. Range is `0.0` to `360.0`.

Copyright © 2026 Vizrt

Radial Blur



The Radial Blur plug-in blurs objects in different ways.



The samples above depict the same image with radial blur quality set to Low, Normal, High and Very High with Center X and Y set to `-25.0` and `25.0`, Inner Range and Scale set to `25.0` and `50.0`, respectively.

> **Note:** Radial blur does not clamp the object.

> **Note:** This plug-in is located in: Plugins -> Shader -> Filter

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Radial Blur Properties

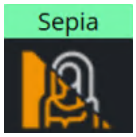
- **Quality:** Sets the quality of the blur filter. Low is more pixelated, whereas very high is smooth. Available options are *Low*, *Normal*, *High* and *Very High*.
- **Center X and Y:** Sets the position of the object's radial blur.
- **Inner Range:** Sets the inner range of the blur.
- **Scale:** Sets the scale of the object's blur.

See Also

- Blur

Copyright © 2026 Vizrt

Sepia



The Sepia plug-in generates a sepia effect using two colors, and to adjust the desaturation and tone of the color for example to blend the image with the color scheme of the overall scene.

Sepia is similar to what is known as duo tone in most photo editing suites.



The samples above depict the same image with tone levels set to `100.0`, `50.0` and `0.0`. All other values are set to default. The last image is the original (reference) without the sepia plug-in.

Note: The color range is from `0.0 – 100.0` divided by 255 (0.392, 0.784 etc.).

Note: This plug-in is located in: Plugins -> Shader -> Filter

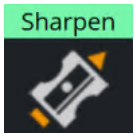
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Sepia Properties

- **Desaturation:** Sets the saturation level. Range is `0.000` (color) to `100.000` (black & white).
- **Tone:** Sets the color tone. Range is `0.000` to `100.000`.
- **Light Color:** Specifies the light color and the strength of it. Range is `0.000` to `100.000`.
- **Dark Color:** Specifies the dark color and the strength of it. Range is `0.000` to `100.000`.

Copyright © 2026 Vizrt

Sharpen



The Sharpen plug-in sharpens the given images on objects.



The samples above depict the same image with Quality set to High, and Scale values at `0.0`, `100.0`, `200.0` and `300.0`, respectively. Range is set to `25.0` (default). The first image is the original (reference) with Scale set to `0.0`.

> **Note:** This plug-in is located in: Plugins -> Shader -> Filter

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Sharpen Properties

- **Quality:** Sets the quality of the sharpen filter. Low is more pixelated, whereas high is smooth. Available options are *Low*, *Normal* and *High*.
- **Range:** Includes more more pixels in the operation when the value is increased.
- **Scale:** Sharpens the image when the value is increased.

Copyright © 2026 Vizrt

2.5.4 Material Shaders

The following Shader plug-ins are located in the Material folder:

- Anisotropic Light
- Bump Map
- Cartoon
- Gooch
- Lighting Shader
- Simple Bump Map

Copyright © 2026 Vizrt

Anisotropic Light



The Anisotropic Light plug-in uses a lookup texture to compute the distribution of the specular component. All the shader properties are defined by the material of the container.

Only Light number 1 is used and the light is always treated as a local light source.

In the figure below, the left image is with anisotropic light and the right image is with a regular material.



Note: This plug-in is located in: Plugins -> Shader -> Material

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

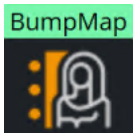
Copyright © 2026 Vizrt

Anisotropic Light Properties

This plug-in does not have any properties or parameters.

Copyright © 2026 Vizrt

Bump Map



The Bump Map plug-in simulates complex surface structures. Instead of using many triangles to render a 3D object with many structures on its surface, use bump mapping to get a similar result.

Use plug-ins for Adobe Photoshop, Gimp, built-in functions of your modeling package or NVIDIA's Melody tool.

Note: Make sure the geometry supports tangent space vectors. Otherwise, use the Simple Bump Map shader.



Note: This plug-in is located in: Plugins -> Shader -> Material

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Bump Map Properties

- **NormalMap:** Stores a direction of normals directly in the RGB data. Move an normal map via drag and drop onto this field.
- **HeightMap:** Move an height map via drag and drop onto this field. The Height Map file is only visible and used when parallax mapping is enabled.
- **Light Source:** Selects the light source in the range 1-8. Please consider that only one light source is possible. By default, the first light source is used.
- **Position X and Y:** Defines the position of the normal map in X-direction and Y-direction.
- **Scale X and Y:** Scales the normal map, respectively changes the width and height size of the image.
- **Wrap:** Repeats or clamps the bump map for the image.

Copyright © 2026 Vizrt

- **Toggle X and Y:** Changes the direction of the normals stored in the normal map image.
- **Scale Height:** Changes the intensity of the bump map effect. Scale Height is visible when parallax mapping is enabled.
- **Parallax Mapping:** Provides an additional map to the texture for more apparent depth and the object is more realism.

See Also

- Simple Bump Map

Copyright © 2026 Vizrt

Cartoon



The Cartoon plug-in renders objects in a cartoon like manner. Shading is done in steps and you can define number of steps, the brightest color and the darkest color. Edge width allows you to blend between the steps. Material and texture of the container are ignored. Only Light number 1 is used and the light is always treated as local light source.



Note: This plug-in is located in: Plugins -> Shader -> Material

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Cartoon Properties

- **Shades:** Defines number of shade levels.
- **Edge Width:** Creates a smooth effect. The transitions are more hard or more soft.
- **Silhouette Width:** Sets the width of the contours around the rendered objects.
- **Silhouette Color:** Sets the color of the contours around the rendered objects. Additionally, you can select the alpha value if you want transparent areas. `0%` (Invisible) - `100%` (Visible).
- **Surface High:** Sets the color in the broad range. Additionally, you can select the alpha value if you want transparent areas. `0%` (Invisible) - `100%` (Visible).
- **Surface Low:** Sets the color in the dark range. Additionally, you can select the alpha value if you want transparent areas. `0%` (Invisible) - `100%` (Visible).

Copyright © 2026 Vizrt

Gooch



The Gooch plug-in is a per pixel light with a reflective high light. Material of the container is ignored, only the shader parameters are used to define the surface color.

Texture mapping works as usual, but the texture can be blended with the surface color using the Texture Alpha parameter. Only Light number 1 is used and the light is always treated as local light source.

In the figure below, the left image is with Gooch and the right image is with a regular material.



Note: This plug-in is located in: Plugins -> Shader -> Material

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Gooch Properties

- **Surface:** Describes the intensity of color like the ambient color of a material.
- **Cool:** Defines the cool color. Both colors cool and warm are merged. The result is a color gradient.
- **Warm:** Defines the warm color.
- **Highlight:** Sets the diameter of the highlight.
- **Highlight Alpha:** Sets the alpha or transparent value of the highlight.
- **Texture Alpha:** Describes the visible value of the texture image in percent. `0%` (Invisible) - `100%` (Visible).

Copyright © 2026 Vizrt

Lighting Shader



The Lighting Shader plug-in performs per-pixel lighting, as opposed to default per-vertex lighting in Viz.

Note: This plug-in is located in: Plugins -> Shader -> Material

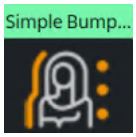
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Lighting Properties

- **Specular Type:** Shows the color that appears when light comes from a particular direction and bounces back from the surface in a mirrored direction. The specular color is 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.

Copyright © 2026 Vizrt

Simple Bump Map



The Simple Bump Map plug-in simulates complex surface structures. Instead of many triangles to render a 3D object with many structures on its surface you can use bump mapping to get a similar result.

You can only use a gray scaled image as normal map. If you want a more accurate result and better effects please use the Bump Map plug-in instead which is using an RGB normal map image.

To generate a Simple Bump Map you can use the plug-ins for Photoshop, Gimp, the built-in functions of your modeling package or NVIDIA's tool Melody. The Height Map (used for parallax mapping) should be a grayscaled image (white -> highest, black -> lowest bumps).

Note: All calculations are done in object space.

Copyright © 2026 Vizrt

Note: This plug-in is located in: Plugins -> Shader -> Material

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Simple Bump Map Properties

- **NormalMap:** Stores a direction of normals directly in the RGB data. Move an normal map via drag and drop onto this field.
- **HeightMap:** Move an height map via drag and drop onto this field. The Height Map file is only visible and used when parallax mapping is enabled.
- **Light Source:** Selects the light source in the range 1-8. Please consider that only one light source is possible. By default the first light source is used.
- **Position X and Y:** Defines the position of the normal map in X-direction and Y-direction.
- **Scale X and Y:** Scales the normal map, respectively changes the width and height size of the image.
- **Wrap:** Repeats or clamps the bump map for the image.
- **Toggle X and Y:** Changes the direction of the normals stored in the normal map image.
- **Scale Height:** Changes the intensity of the bump map effect. Scale Height is visible when parallax mapping is enabled.
- **Parallax Mapping:** Provides an additional map to the texture for more apparent depth and the object is more realism.

See Also

- Bump Map

Copyright © 2026 Vizrt

2.5.5 RealFX Shaders

The RealFX plug-in set enables you to create particle effects in Viz Artist.

Particle systems are a computer graphics technique to simulate certain physics-based effects, which are otherwise very hard to reproduce with conventional rendering techniques. Examples of such effects which are commonly replicated using particle systems include fire, explosions, smoke, weather effects, sparks, falling leaves, dust, meteor tails, or abstract visual effects like glowing trails, magic spells, etc.

The particle effects in Viz Artist run in real-time, meaning that there are a few inherent constraints that must be taken into account when considering best practices for employing this plug-in set. For example, there is a trade-off between the number of particles and performance optimization; more generally there needs to be a considered balance between performance and visual quality.

RFxSmoke is the baseline plug-in within the RealFX plug-in set. The remaining plug-ins in this set are applied on top of RFxSmoke in any given container. RFxSmoke includes built-in functionality and the ability to host the additional functionality contained in the other plug-ins in this set. Part of the built-in functionality (for example, Turbulence) is kept for compatibility with previous versions of Viz Artist.

There are three categories of additional plug-ins:

- **Birth plug-ins:** Refers to where the particles are spawned.
- **Motion control plug-ins:** Governs the position, direction, velocity, size and color of each particle.
- **Texture control plug-ins:** Affects the texture mapping and the “look” of each particle by using pixel shader technology.

Copyright © 2026 Vizrt



The following Shader plug-ins are located in the RealFX folder:

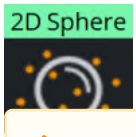
- RFx2DSphere
- RFxClouds
- RFxFlame

See Also

- Basic Geometry RealFX in Basic Geometry Plug-ins
- pxColorWorks in Basic Container Plug-ins
- RFxSmoke

Copyright © 2026 Vizrt

RFx2DSphere



The RFx2DSphere plug-in creates a 2D sphere-like look on each particle.

The sphere is affected by light sources.

Note: This plug-in is located in: Plugins -> Shader -> RealFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

RFx2DSphere Properties

Adjust the following parameters as required to achieve the required effect:

- **Color Mode** (Material, Color, Color-Material)
- **Radius**
- **Blend**

Copyright © 2026 Vizrt

RFxClouds



The RFxClouds plug-in creates a dynamic plume-like texture on each particle, and changes randomly for each particle over time.

It can be used on clouds and smoke.

Note: This plug-in is located in: Plugins -> Shader -> RealFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

RFxClouds Properties

Adjust the following parameters as required to achieve the required effect:

- **Brightness**
- **Turb Scale X**
- **Turb Scale Y**
- **Turb Speed**

Copyright © 2026 Vizrt

RFxFlame



The RFxFlame plug-in creates a dynamic flame-like texture on each particle, and changes randomly for each particle over time.

Note: This plug-in is located in: Plugins -> Shader -> RealFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

RFxFlame Properties

This plug-in has no editable parameters.

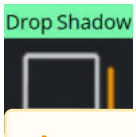
Copyright © 2026 Vizrt

2.5.6 Texture Shaders

The following Shader plug-ins are located in the Texture folder:

- Drop Shadow
- Emboss
- MultiTexture
- Substance

Drop Shadow



The Drop Shadow plug-in generates 2D shadow of a texture.

Note: This plug-in is located in: Plugins -> Shader -> Texture



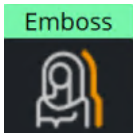
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Drop Shadow Properties

- **Distance:** Sets the distance of the shadow.
- **Direction:** Sets the direction of the shadow in degrees.
- **Threshold:** Sets the shadow threshold.
- **Soft Shadow:** Enables the Soft Distance parameter for applying a soft shadow.
    - **Soft Distance:** Sets the distance of the soft shadow.
- **Color:** Sets the color of the shadow.

Copyright © 2026 Vizrt

Emboss



The Emboss plug-in allows you to raise highlighted surfaces and lower shadows of textures and images creating an embossed look.

Note: This plug-in is located in: Plugins -> Shader -> Texture

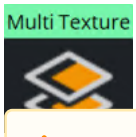
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Emboss Properties

- **Direction:** Sets the direction of the emboss. Options are:
  - None
  - South-East
  - South-West
  - North-East
  - North-West
- **Emboss:** Sets the Emboss level, in percentage.

Copyright © 2026 Vizrt

MultiTexture

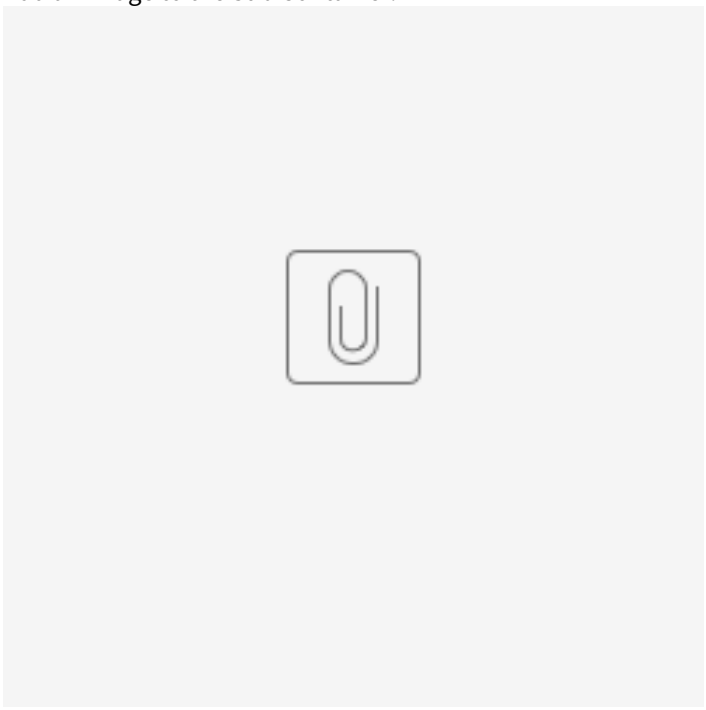


The MultiTexture plug-in blends (linear) two textures together.

Note: This plug-in is located in: Plugins -> Shader -> Texture

To Create a Simple Blended Image Scene

1. Create a new Container.
2. Add an image to the new Container.
3. In the Image Editor set the image to:
    - Unit: Level one
    - Inheritable
4. Create a Sub Container.
5. Add an image to the Sub Container.



6. In the Image Editor set the image to:
    - Unit: Level two
7. Add the MultiTexture plug-in to the Sub Container.
8. Click on the MultiTexture plug-in.
9. Adjust the image blend as required.

Copyright © 2026 Vizrt



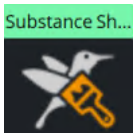
Copyright © 2026 Vizrt

Blended Images Example



Copyright © 2026 Vizrt

Substance

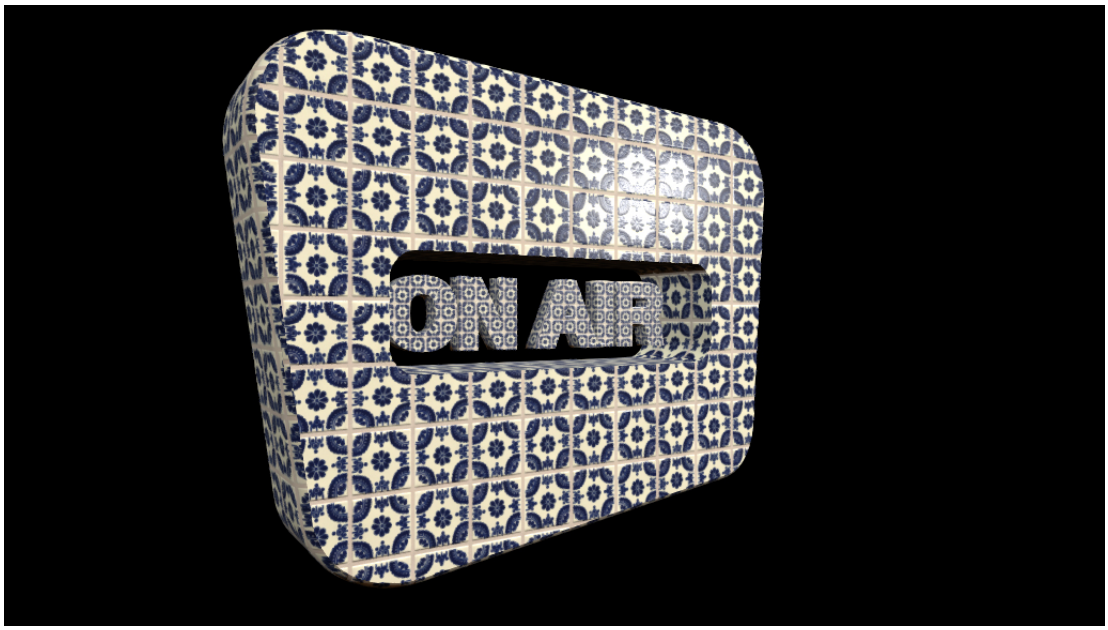


The Substance Shader plug-in renders *Allegorithmic PBR* (Physically Based Rendering) substances in Viz Engine.

Substance Shader takes into account the position, color and enabled/disabled state of the first eight Lights in the Scene.

It also supports materials with alpha properties.

Note: This plug-in is located in: Plugins -> Shader -> Texture



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Substance Properties

Note: The Substance properties panel parameters are different for each selected Substance.

The following options are available in the Basic Tab of the Substance Shader. The substance settings depend on the settings the creator of the material has exposed:

- **Substance:** Uses a PBR Substances item (*.sbsar* file). Drag to the drop zone.
- **Environment Map:** Uses a latitude/longitude panorama environment map image. Drag to the drop zone.

Copyright © 2026 Vizrt

- **Anisotropic Filtering:** Sets the image quality. The default value is `4X`.
- **Relief Amount:** Sets the relief amount that the Shader applies.
- **Tiling:** Sets the tiling.
- **Emissive Intensity:** Sets the intensity of the emissive properties (if the substance has any).
- **Ambient Intensity:** Sets the ambient light taken into account by the Shader.
- **sRGB Base Color:** Converts the base color map to sRGB (default: on).
- **Texture Rotation:** Sets the rotation of the texture on the geometry.
- **Texture Scaling X/Y:** Sets the scaling of the texture on the geometry.
- **Texture Position X/Y:** Sets the position of the texture on the geometry.

To Create a Substance Shader Effect

1. Create a new Container.
2. Add a **Cube** geometry.
3. Add the **Substance Shader**.
4. Drag a **Substance** to the **Substance** drop zone.
5. Drag a latitude/longitude panorama image to the **Environment Map** drop zone.
6. Use the Substance Shader properties to modify the result, if required.
7. Click **Substance Settings** to adjust the properties in the Substance properties panel.

Note: The import and use of Substances in Viz Artist requires a separate license. You obtain this license, as well as a license for the Allegorithmic Substance Designer and the Substance PBR database, from your local Vizrt representative.

Substances use two different file extensions, *.sbs* and *.sbsar*:

- *.sbs* files are editable substance source files. Use Substance Designer with these files.
- *.sbsar* files are performance optimized, published substances.

You can import *.sbsar* files into Viz Artist. They must be Physically Based Rendering (PBR) materials to work correctly in Viz Artist. Make sure you create the correct material type when creating Substance Shaders in Substance Designer.

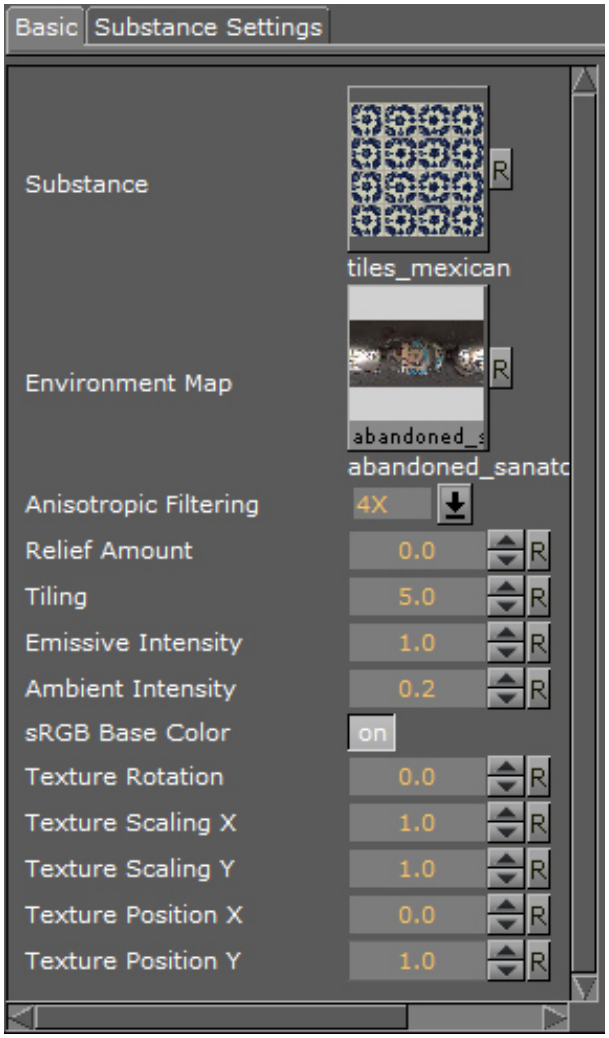
To create a *.sbsar* file, first create a Shader with Substance Designer, and then publish this Shader for import to Viz Artist. To view and adjust the parameters of a Shader, switch to the **Substance Settings** panel. The parameters in the settings panel are different for each loaded substance, depending on the type of Shader. The only supported type of Shader in Viz Artist is PBR Metallic/Roughness.

Tip: If you get the error message *This is not a valid Substance* when you apply the Shader in Viz Artist, you need to make sure you have a PBR Metallic/Roughness material created in Substance Designer.

Substance Basic

Here you can adjust the basic settings of your Substance Shader.

Copyright © 2026 Vizrt



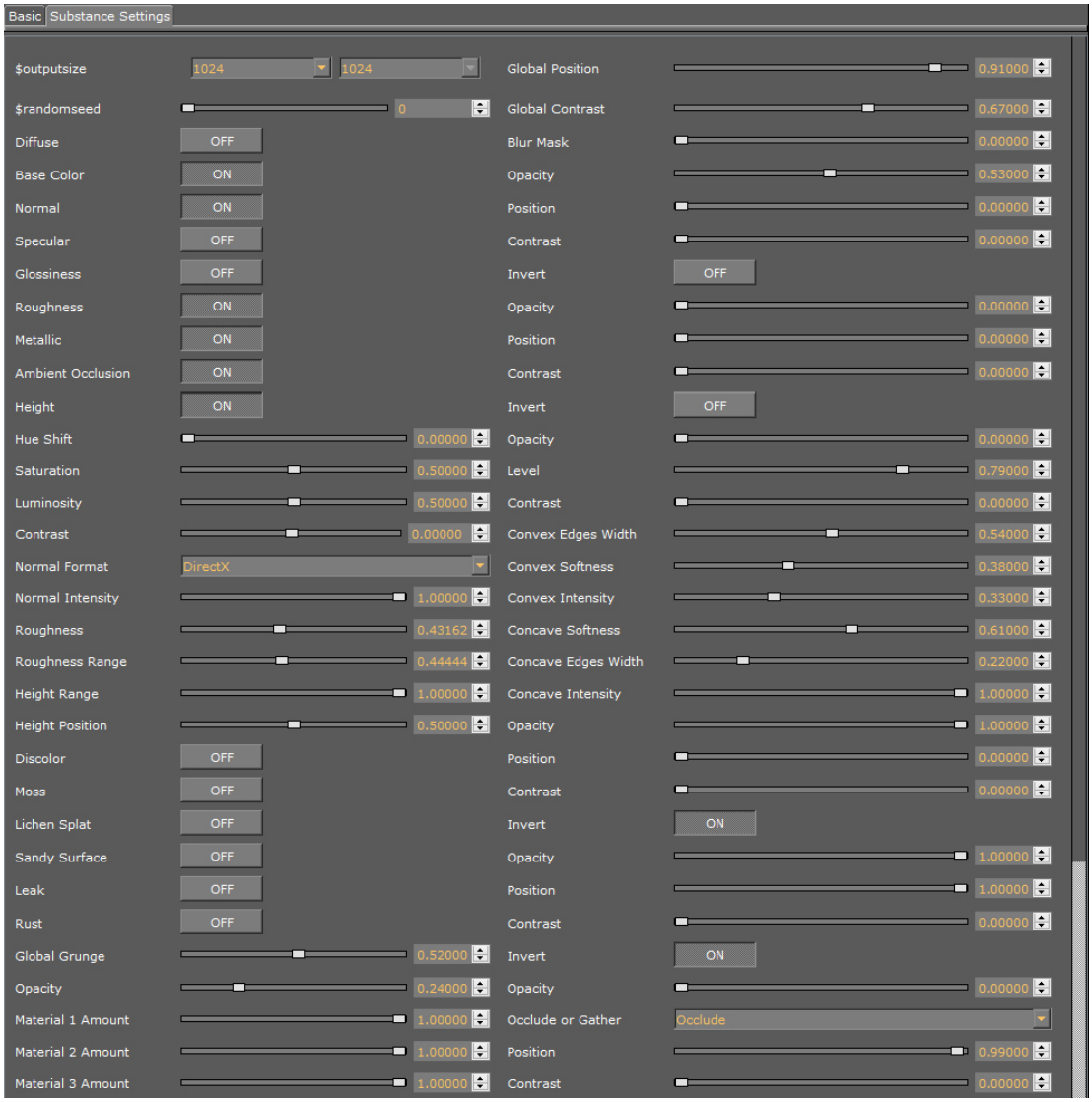
- **Substance:** Drag the desired Substance Shader from the Viz database section to this drop-field.
- **Environment Map:** Drag any RGB or HDR environment map that should be reflected on the surface to this drop-field. The strength is controlled by the **Ambient Intensity** and by the metallic setting exposed individually in a Substance Shader.
- **Anisotropic Filtering:** Increases the load on the GPU during Shader calculation at higher settings. This is usually set between `0-4`.
- **Relief Amount:** You only need to adjust this setting for relief Shaders.
- **Tiling:** Lets you specify the number of tiles.
- **Emissive Intensity:** Change this setting for a higher emission rate of your Shader.
- **Ambient Intensity:** Controls the blending when you have an Environment texture applied to your Substance Shader.
- **sRGB Base Color:** Uses the sRGB color space model when enabled.
- **Texture Rotation:** Rotates the texture applied to your geometry. This rotation has a limitation in the rotation degree of freedom, If the limit is reached, you need to change the rotation in your Substance Shader.
- **Texture Scaling X:** Scales the applied texture/Shader on the x axis.

Copyright © 2026 Vizrt

- **Texture Scaling Y:** Scales the applied texture/Shader on the y axis.
- **Texture Position X:** Changes the `X` position of the applied texture/Shader on the x axis.
- **Texture Position Y:** Changes the `Y` position of the applied texture/Shader on the y axis.

Substance Settings

The settings of each Substance Shader are different, depending on the material itself. All Substance Shaders have in common that they need to generate a texture from the procedural Shader description created in Substance Designer. If you are creating texture based Shaders, you need to ensure that your base texture has enough resolution. All other settings and sliders in the Substance Shader vary with each different Shader, depending on which features are exposed in each individual Shader. Some settings are exposed for almost any existing Shader from the Substance PBR library. These include **Hue Shift**, **Saturation**, **Luminosity**, **Contrast**, **Normal Format**, **Output Size** and **$randomseed**. If you create your own Shaders in Substance Designer, you must expose the modifiers manually from Substance Designer if they are to be editable in Viz Artist.



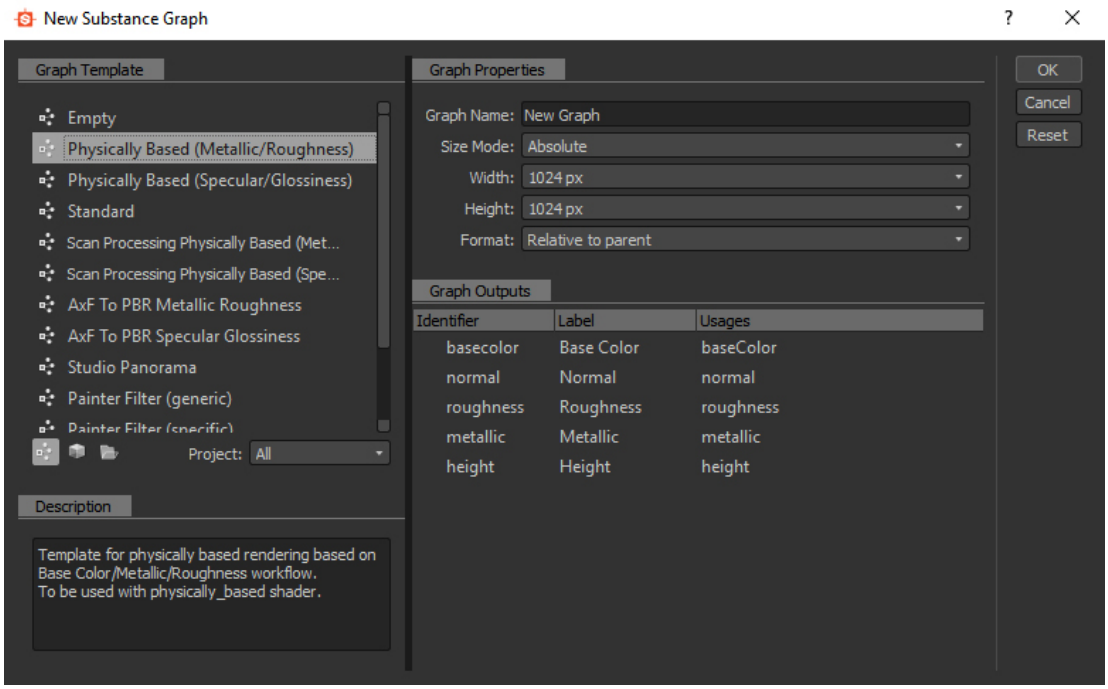
Copyright © 2026 Vizrt

Tip: Check the NVIDIA GPU meter to see performance used by the Shaders.

How to Create Substance Shaders with Substance Designer

This section gives you a brief introduction on how to create a PBR Metallic/Roughness material in Substance Designer. Substance Designer allows you to create procedural materials as well as texture/scan based materials. Please see the Allegorithmic website and their Youtube channel for more detailed information on how to work with Substance Designer: https://www.allegorithmic.com/.

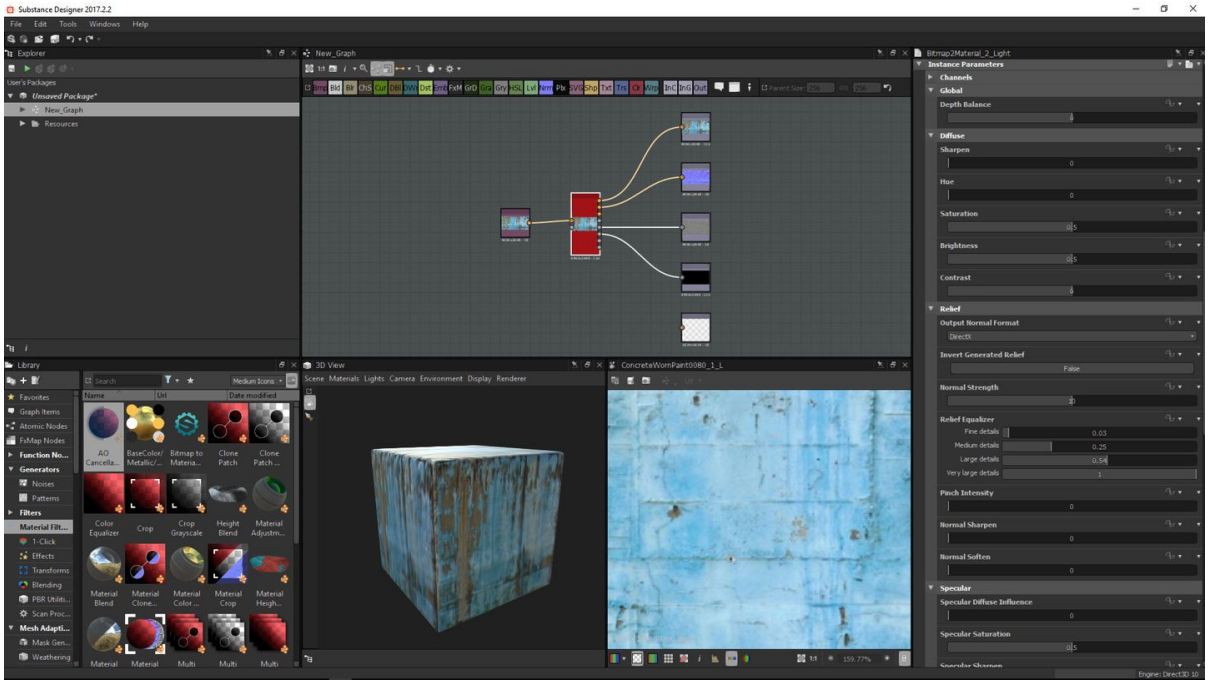
When you create a new Substance in Substance Designer, you need to select the **Physically Based (Metallic/Roughness) Graph Template**. Otherwise, this Shader does not work properly in Viz Artist, causing the error message *This is not a valid Substance*. When saving a Substance Shader/material, it is saved as an *.sbs* file. These files cannot be imported into Viz Artist. To import Substance Shaders in Viz, you need to publish the material to an *.sbsar* file.



The Shader graph of each material looks different. Here are a few examples for different Shader graphs with PBR Metallic/Roughness materials, reaching from simple texture Shader graphs to complex procedural Shader graphs.

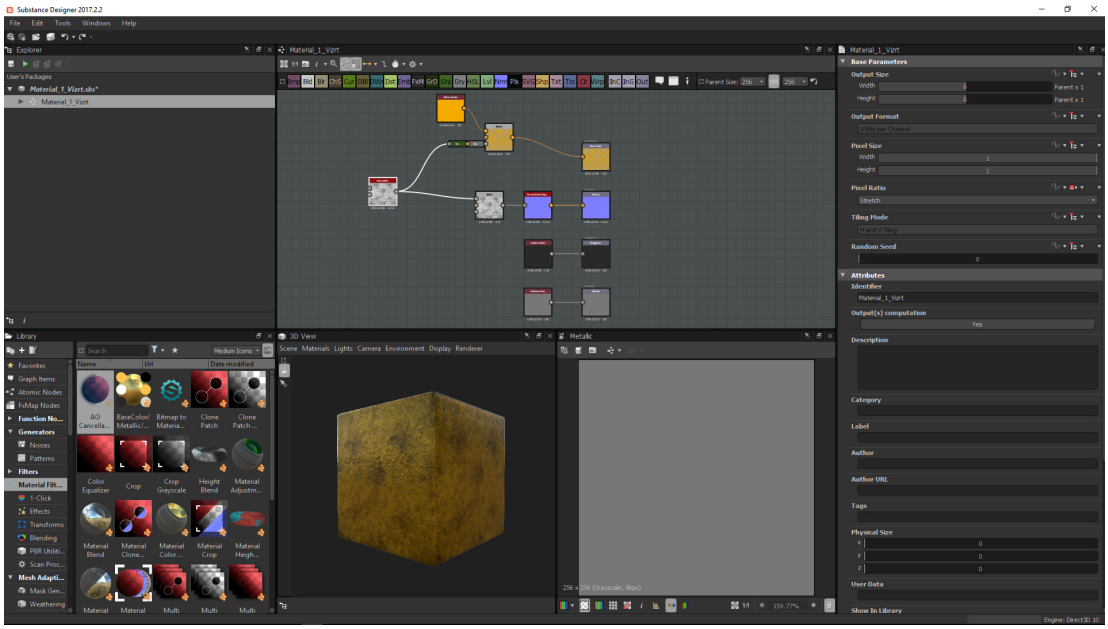
Copyright © 2026 Vizrt

Simple Texture Shader



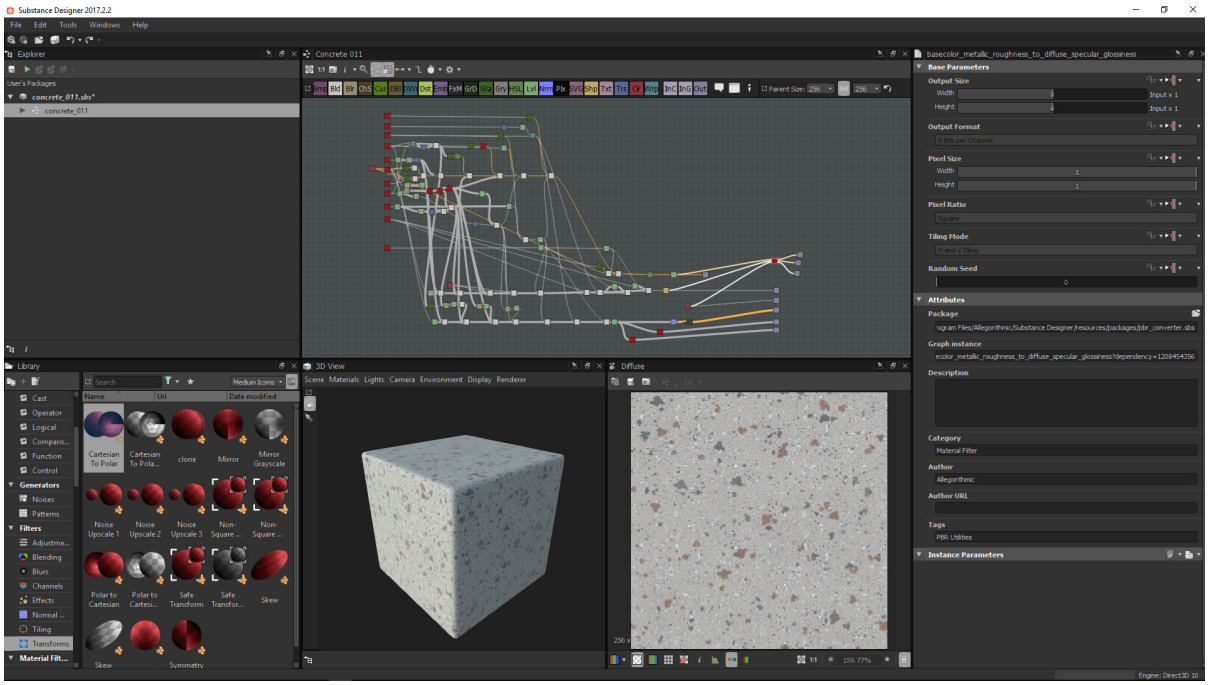
In the Shader graph, you see the five Output channels for the PBR material: **Base color**, **Normal**, **Roughness**, **Metallic**, and **Height**. All outputs are generated automatically from a simple input texture.

Basic Procedural Shader



In the Shader graph, you see the four Output channels for the PBR material: **Base color**, **Normal**, **Roughness**, and **Metallic**. All outputs are generated and mapped manually.

Complex Procedural Shader



Here you see a very complex procedural Shader graph, constructed with different modifiers and functions, generating the necessary output for **Base Color**, **Normal**, **Roughness**, **Metallic** and **Height**.

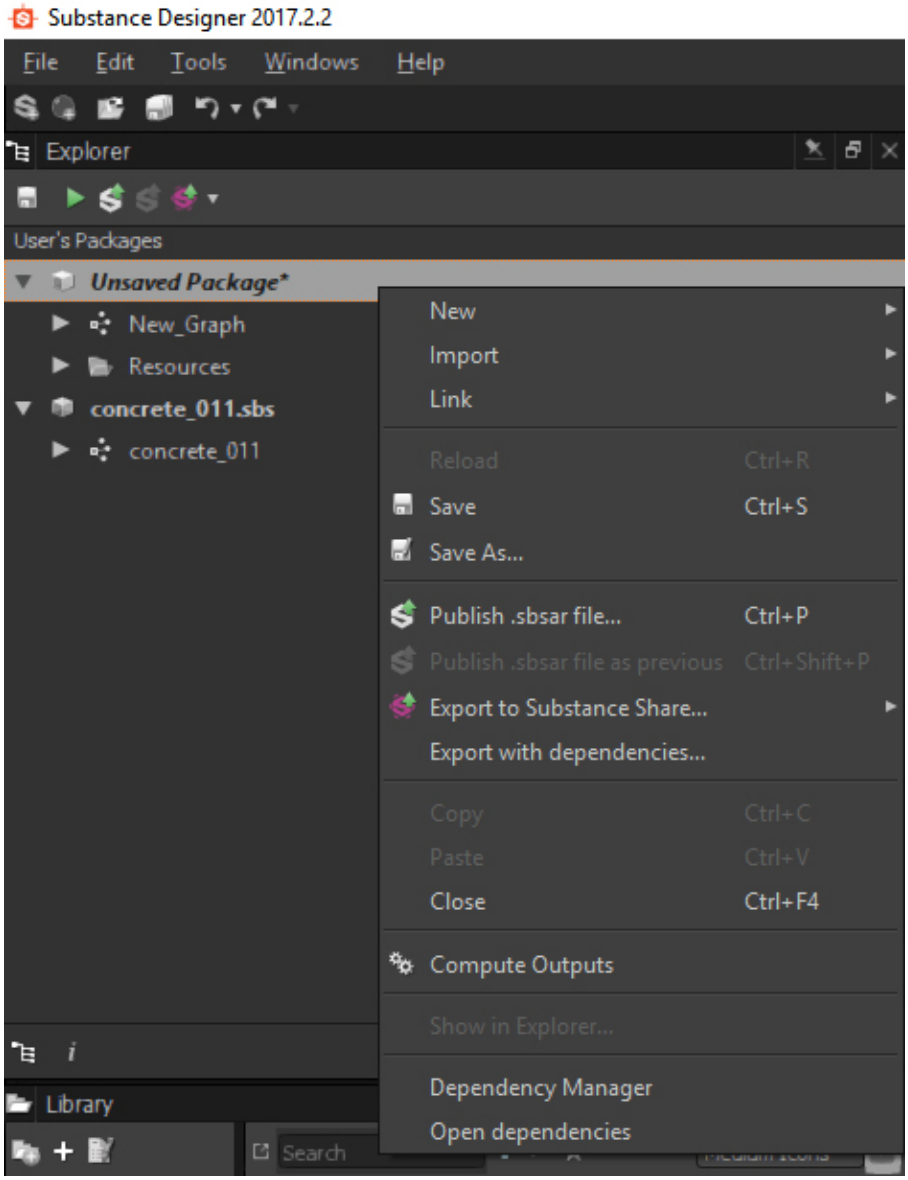
Tip: Only PBR Metallic/Roughness materials from Substance Designer are supported in Viz Artist.

Publishing Shaders

When you want to import the Substance Shaders in Viz Artist, you need to publish them from Substance Designer instead of saving them. The difference between publishing and saving is that the Shader is performance optimized when published. Published Shaders cannot be opened by Substance Designer, so it is important to keep your *.sbs* files and *.sbsar* files together in case you need to make changes to your Shader later on.

To publish the Shader, right click the Shader/package and select **Publish *.sbsar* file**.

Copyright © 2026 Vizrt

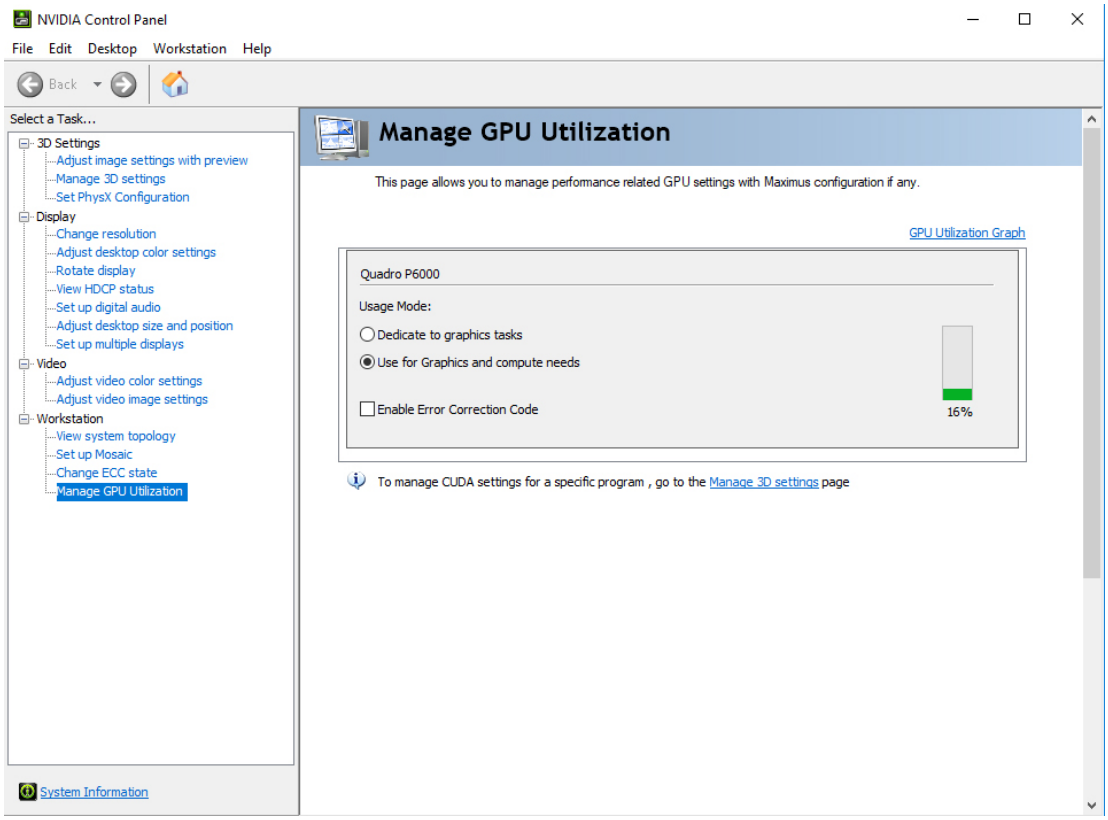


Tip: You can find documentation and training with Substance Designer on https://www.allegorithmic.com/ as well as on the Allegorithmic Youtube channel.

Substances Performance

Substance Shaders are computed on your GPU. Vizrt recommends an NVIDIA M-series graphics card or higher to work efficiently with Substances. Since Substances run exclusively on the GPU, you should check the NVIDIA GPU meter when working with Substances inside Viz to have full overview of the rendering performance. There should be a section called **Manage GPU Utilization** in your **NVIDIA Control Panel**. Check the performance of the Substance Shaders here.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

2.5.7 RTT Advanced Materials Shaders

RTT Advanced Materials plug-ins allow you to create a number of high-quality materials using state-of-the art shader technology. Using the plug-ins is very easy. Simply drag the shader plug-in of your choice onto selected geometry, assign a material and set up the additional parameters as shown by the shader in its own editor. Some plug-ins remain inactive until you have applied all necessary textures, depending on the shader plug-in. This may include a basic texture on your container and/or additional textures to be dropped in the plug-in interface. You can modify all parameters of the applied material on your object, such as Ambient Color, Diffuse, Specular and Emission, Shininess and Alpha. All these parameters also affect the shader.

All RTT Advanced Materials plug-ins (except the Bump Optimized Shader) allow manipulating the texture mapping of each texture map individually. This means you can use a different position, rotation and scaling for each texture. As for the mapping of additional textures inside the Shader, the same mapping method as being assigned to the texture on your container is used. It is also possible to change the alignment of the environment textures to achieve the correct reflections on the respective object surface.

If you do not want to use individual texture transformations, you may switch the option off. Each additional texture (except the environment texture) offers the button **Enable Individual Transform**. With the button turned `Off`, the texture mapping coordinates of the base texture is used.

RTT Shaders support up to eight light sources. The supported light types are infinite and local lights; also the light color of these lights is supported.

If you import the Viz Artist/Engine archive **RTT_ShaderPresets_V1**, you obtain about 60 sample scenes which are meant to give you an idea how to work with the Shaders. These presets are free to use, but in general, they are thought as a guideline for how to work with textures and the Shaders. Correctly working with the textures dramatically improves the look of your Shader. The Shaders use color maps, bump maps (in the normal map format), specular maps and cubic environment maps in HDR and LDR formats.

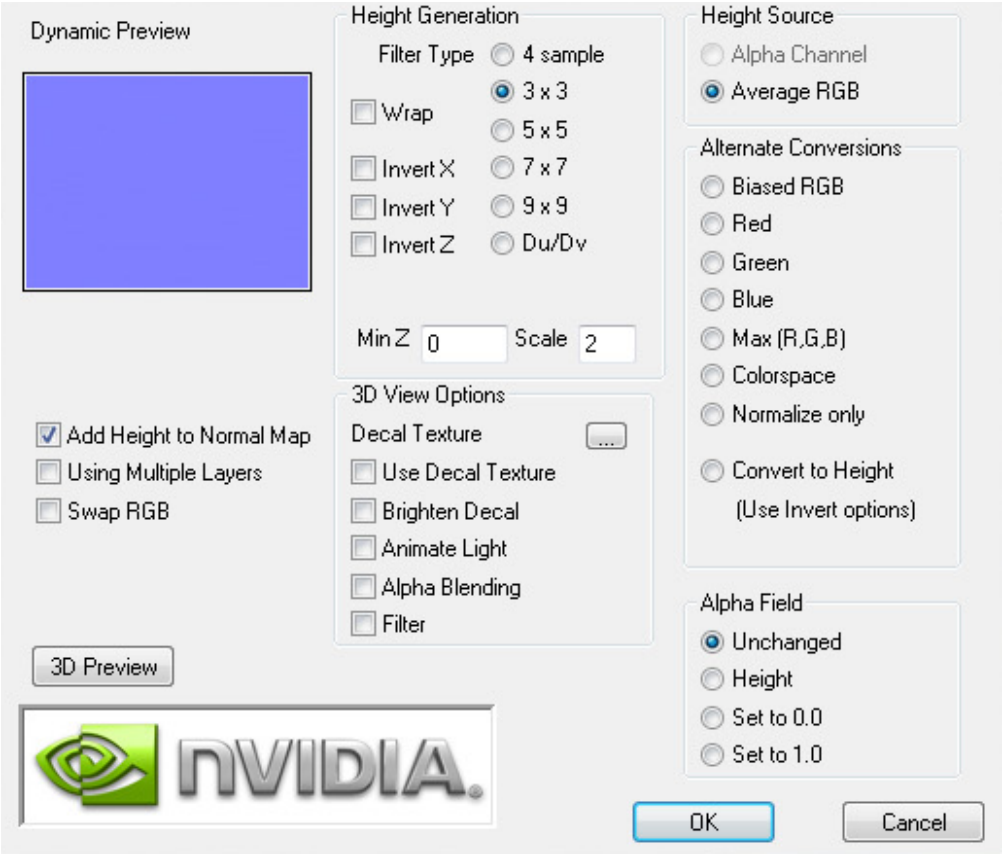
Creating color maps is straightforward. They are regular RGB textures that you can use either on the container or inside the Shader, depending on the Shader you are using.

Creating specular maps is also straightforward. They are grayscale textures where you define, through a gradient from white to black, where light is reflected or not.

Creating normal maps for the bump inside your Shader is not complicated either; just go to NVIDIA's corporate web page and download the relevant NVIDIA Photoshop plug-ins, or use this link NVIDIA Texture Tools for Adobe Photoshop.

After installing the filter in question, you have a new section in the filter category in Adobe Photoshop. By applying this filter, you are shown a window with various settings (Please read the corresponding NVIDIA User’s Guide); yet there is not much work involved. Just have a look at the settings in the NVIDIA Photoshop plug-in Image below and start exploring them on your own.

Copyright © 2026 Vizrt



Creating Cubic Environment maps is no secret either. Cubic Environment Maps can be rendered either with a preferred Render Engine (Mental Ray, V-Ray, etc.), if it should be a 3D generated environment map, or they can be created with HDR Images, which are then transformed with HDR Shop.

Note: HDR Shop has a menu for panoramic transformations, here you can create a cubic environment image by transforming it from latitude/longitude to an cubic environment by just a few mouse clicks (see instructional video).

Note: Creating Cubic Environment Maps (HDR or LDR) is a well known technique for 3D experts. This Viz Artist User Guide does not explain the creation of these Environment Maps in detail, as there are numerous tutorials on the web for this texture creation procedure.

If you do not have a license for the RTT Advanced Materials plug-ins, you may contact either one of your local Vizrt Support people or you can send an e-mail to license@vizrt.com to obtain either a time-limited demo license or a full license. Without a license, you can load the preset scenes, but you cannot modify them; only a few parameters are adjustable.

The following Shader plug-ins are located in the RTTAdvancedMaterials folder:

- Cube Map
- HDR
- Normal Map
- Fresnel

Copyright © 2026 Vizrt

- RTT

Cube Map

Cube map textures are typically used for approximated environmental reflection and refraction. They consist of a set of six two-dimensional textures that form a textured cube centered at the origin. Unlike 2D sphere maps, they grant higher details for the whole environment. The RTT Advanced Materials plug-ins accept only cube maps as reflection textures and require them to be stored in a vertical cross layout. Cube maps can soon be created easily by using the Dynamic Texture plug-in. To achieve the best results with the Shaders we strongly recommend to create your cube maps in HDR format.

Note: The vertical cross cubemap size is required to be one of the following: 192 by 256, 384 by 512, 768 by 1024, 1536 by 2048 or 3072 by 4096 pixels, where a higher resolution indicates higher image quality. The general rule of thumb is that the contained six images need to have power-of-2 resolution.

HDR

HDR (High Dynamic Range) images can store a much wider range of brightness values than common (low dynamic range) images. Therefore, it is possible to create more brilliant highlights and reflections when using HDR images. HDR images can be shot by using special spherical cameras, or created with the aid of HDR image processing tools. Furthermore ready-to-use HDR cube maps are available as DVD collections. All plug-ins with environmental reflections support HDR cube maps. *.hdr* format images are supported.

Normal Map

Normal maps are textures used to calculate bump mapping. They encode surface details (the normals of a surface) as RGB color values and therefore show usually in some shades of blue. They can be created easily from (grayscale) height maps with plug-ins for Adobe Photoshop or Gimp.

Fresnel

The Fresnel effect describes how much light is reflected on a surface and how much light is refracted through this surface. At shallow angles the reflection is strong while there is almost no refraction.

RTT

All RTT Advanced Materials are developed by Realtime Technology AG. Realtime Technology AG is a worldwide leading supplier of 3D real-time visualization technologies and services for industrial applications in the automotive, aircraft and consumer goods industries.

Note: All RTT Shaders can be uninstalled from the Viz Artist/Engine program menu.

Copyright © 2026 Vizrt

Anisotropic Light Shader



The Anisotropic Light Shader plug-in calculates a highlight to simulate advanced surfaces.

The highlight can be controlled in using two parameters. In addition, a color texture can be used to add more details. This shader is particularly useful when no surrounding reflection is required.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

Anisotropic Light Shader Properties

- **Roughness X and Y:** Controls the width and height of the anisotropic highlight.
- **Color Map 2:** Offers the possibility to assign a second diffuse color texture that is blended with the first color texture (according to the chosen technique)
- **Enable Individualized Transform:** Activates the option to define an independent transformation (Position, Rotation and Scale values next to the Image Parameter) for Normal- and/or Bumpmaps instead of using the regular texture coordinates.
- **Technique:** Shows a list of available techniques.
    - **Standard:** Is a technique that creates an anisotropic highlight (to be used with geometry that has no texture coordinates and normals).
    - **Standard_Texture:** Is a technique that mixes a diffuse texture color into the material color.
    - **Standard_MultiTexture_Add:** Is a technique to additively blend Color Map 2 with the texture.
    - **Standard_MultiTexture_Blend:** Is a technique to blend Color Map 2 with the texture according to their alpha values.
    - **Standard_MultiTexture_Subtract:** Is a technique to subtractively blend Color Map 2 with the texture.
    - **Standard_MultiTexture_Modulate:** Is a technique to blend Color Map 2 by multiplying it by the texture.
    - **Binormal:** Is a technique to create an anisotropic highlight using binormals and tangents of the geometry (to be used with geometry having texture coordinates and normals).
    - **Binormal_Texture:** Is a technique to mix a diffuse texture color into the material color.
    - **Binormal_MultiTexture_Add:** Is a technique additively blend Color Map 2 with the texture.
    - **Binormal_MultiTexture_Blend:** Is a technique to blend Color Map 2 with the texture according to their alpha values.
    - **Binormal_MultiTexture_Subtract:** Is a technique to subtractively blend Color Map 2 with the texture.
    - **Binormal_MultiTexture_Modulate:** Is a technique to blend Color Map 2 by multiplying it by the texture.

Best Practices

Apart from the parameters in the plug-in container, it is necessary to assign a material. When applying a texture technique it is compulsory to assign a texture as a color/basic texture. In case of multi texturing, an additional second texture must be assigned in the corresponding Shader rollout. Therefore, do not forget to adjust material parameters in addition, such as ambient, diffuse, specular color and shininess.

Copyright © 2026 Vizrt

Brushed Metal Shader



The Brushed Metal Shader plug-in uses bump mapping and reflections from a cube map, the impression of more sophisticated metal surfaces, such as brushed metals, is created.

The surface is lit by an anisotropic highlight. To describe the surface structure, a normal map is used.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

Brushed Metal Shader Properties

- **Roughness X and Y:** Controls width and height of the anisotropic highlight.
- **Bump Height:** Affects the height of the created bumps and wrinkles on the surface.
- **Environment Reflectivity:** Manipulates the influence of the reflection color from the environment map on the resulting color.
- **Reflection Multiplier / Exposure:** Affects the exposure of the environment map. This is especially helpful when working with HDR images.
- **Gamma of EnvMaps:** Controls the gamma correction of the environment map. This is once again particularly helpful when working with HDR images.
- **Bump Map / Normal Map:** Is the texture that defines the surface structure with encoded normals.
- **Enable Individualized Transform:** Activates the option to define an independent transformation (Position, Rotation and Scale values next to the Image Parameter) for Normal- and/or Bumpmaps instead of using the regular texture coordinates.
- **Environment Map:** Is a vertical cross cube map that describes the environmental surrounding used to calculate reflections.
- **Technique:** Shows a list of available techniques.
    - **Standard Bump:** Is a technique that should be used with geometry that has no standard UV texture coordinates and normals.
    - **Binormal Bump:** Is a technique that should be used with geometry having texture coordinates and normals.

Best Practices

The Brushed Metal Shader should be preferred to Anisotropic Light Shader whenever you want to create highly detailed metal surfaces. However, the shader is more complex and requires more rendering time. Therefore, it may be unsuitable, depending on the complexity of the other parts of the scene and also on the graphics hardware available.

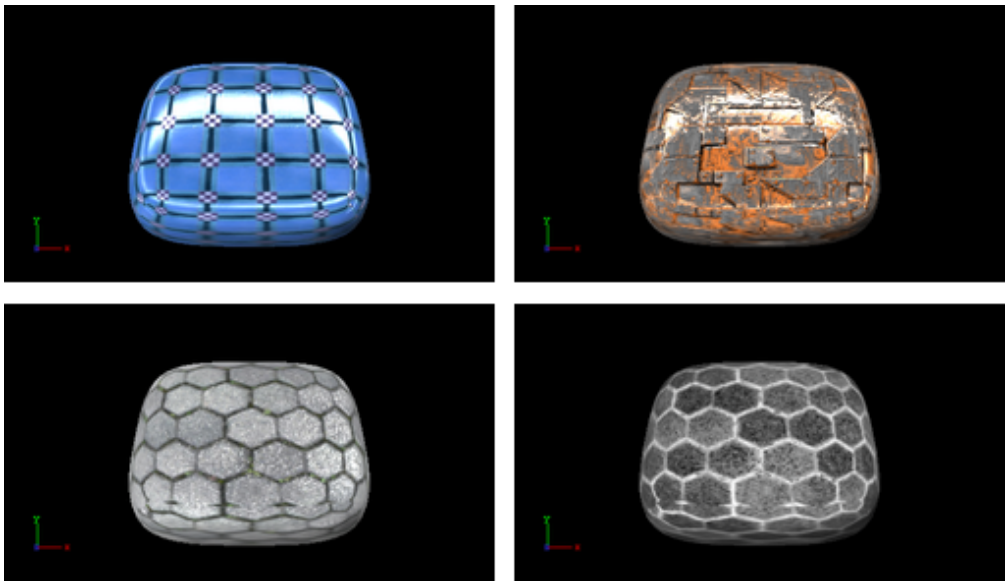
Apart from the parameters in the plug-in container, it is necessary to assign a texture as a color/basic texture and a material. Do not forget to adjust material parameters in addition, such as ambient, diffuse, specular color and shininess, to achieve the required look for your surface.

Copyright © 2026 Vizrt

Bump Optimized Shader



The Bump Optimized Shader plug-in uses bump mapping, the impression of detailed surface structures is created while the structures are defined by normal maps. In addition to a highlight, the reflection of the surrounding is calculated (from a cube map). It is possible to add blurriness to the reflection to create rougher surfaces. A parameter is available to modulate whether the reflections follow the surface structures or behave like a clear coat layer on the top of it. The specular map describes where on the surface highlight and reflections are shown and where the surface shows only diffuse lighting.



This plug-in offers the same functionality as the Bump Shader plug-in, but as an optimization, it does not allow separate texture mapping for bump and specular maps. It consequently requires less rendering time and should be used whenever additional parameters are not required, i.e. you have matching sets of textures that can be used with the shader or you do not need to animate the textures. This shader should be preferred especially in complex scenes.

Note: All RTT shaders can be uninstalled from the Viz Artist/Engine program menu.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

Bump Optimized Shader Properties

- **Bump Height:** Manipulates the height of the created bump on the surface.
- **Modulate Bumpy Reflection:** Controls whether the environment reflection is smooth or influenced by surface details. To calculate reflections on a smooth clear coat layer, value `0.0` is set by default. On the other hand, value `1.0` is used to incorporate surface details.
- **Environment Reflectivity:** Manipulates the influence of the reflection color from the environment map on the resulting color.
- **Reflection Multiplier/Exposure:** Affects the exposure of the environment map. This is especially helpful when working with HDR images.
- **Gamma of EnvMaps:** Controls the gamma correction of the environment map. This is once again particularly helpful when working with HDR images.
- **Blur Reflection:** States how much the environment reflection is blurred to create the impression of a rougher surface.
- **Fresnel Minimum, Fresnel Scale and Fresnel Exponent:** Affect the Fresnel effect upon the reflection.
- **Bump Map/Normal Map:** Is the texture that defines the surface structure/bump with encoded normals.
- **Specular Map:** Is the texture that defines where the surface is lit by specular highlight and reflection.
- **Environment Map:** Is a cube map that describes the environmental surrounding used to calculate reflections.
- **Technique:** Shows a list of available techniques.
    - **Standard Bump:** Is a technique to be used with geometry that has no standard UV texture coordinates and normals.
    - **Binormal Bump:** Is a technique to be used with geometry having texture coordinates and normals.

Copyright © 2026 Vizrt

Bump Shader



The Bump Shader plug-in uses bump mapping, the impression of detailed surface structures is created while the structures are defined by normal maps. In addition to a highlight, the reflection of the surrounding is calculated (from a cube map). It is possible to add blurriness to the reflection to create rougher surfaces. A parameter is available to modulate whether the reflections follow the surface structures or behave like a clear coat layer on the top of it. The specular map describes where on the surface highlight and reflections are shown and where the surface shows only diffuse lighting.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials

Supported Pipelines	
Classic Render Pipeline	✓

Copyright © 2026 Vizrt

Supported Pipelines	
Viz Engine Render Pipeline	✗

Bump Shader Properties

- **Bump Height:** Manipulates the height of the created bump on the surface.
- **Modulate Bumpy Reflection:** Controls whether the environment reflection is smooth or influenced by surface details. To calculate reflections on a smooth clear coat layer, value `0.0` is set by default. On the other hand, value `1.0` is used to incorporate surface details.
- **Environment Reflectivity:** Manipulates the influence of the reflection color from the environment map on the resulting color.
- **Reflection Multiplier/Exposure:** Affects the exposure of the environment map. This is especially helpful when working with HDR images.
- **Gamma of EnvMaps:** Controls the gamma correction of the environment map. This is once again particularly helpful when working with HDR images.
- **Blur Reflection:** States how much the environment reflection is blurred to create the impression of a rougher surface.
- **Fresnel Minimum, Fresnel Scale and Fresnel Exponent:** Affect the Fresnel effect upon the reflection.
- **Bump Map/Normal Map:** Is the texture that defines the surface structure/bump with encoded normals.
- **Specular Map:** Is the texture that defines where the surface is lit by specular highlight and reflection.
- **Environment Map:** Is a cube map that describes the environmental surrounding used to calculate reflections.
- **Technique:** Shows a list of available techniques.
    - **Standard Bump:** Is a technique to be used with geometry that has no standard UV texture coordinates and normals.
    - **Binormal Bump:** Is a technique to be used with geometry having texture coordinates and normals.

Best Practices

If you use this shader in complex scenes, you should favor the Bump Optimized Shader. The difference between the Bump Shader and the Bump Optimized Shader is that you can modify the texture coordinates inside the Bump Shader, which, however, needs more rendering performance. If you do not have to adjust the texture coordinates individually, or animate them separately, you should use the Bump Optimized Shader to save rendering performance.

Apart from the parameters in the plug-in container, it is necessary to assign a texture as a color texture and a material.

Copyright © 2026 Vizrt

Fabric Shader



The Fabric Shader plug-in creates the impression of detailed surface structure on a dull surface by using bump mapping.

The structure is defined by a normal map. A highlight is placed along the edges of the object to increase the impression of fabric surfaces.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

Fabric Shader Properties

- **Velvety (Limb) Parameter:** Controls the strength of the velvety effect at grazing angles of view.
- **Bump Height:** Manipulates the height of the created bumps and wrinkles on the surface.
- **Bump Map/Normal Map:** Is the texture that defines the surface structure with encoded normals.
- **Technique:** Shows a list of available techniques.
    - **Standard_Bump:** Is a technique has to be used with geometry that has no standard UV texture coordinates and normals.
    - **Binormal_Bump:** Is a technique should be used with geometry having texture coordinates and normals.

Copyright © 2026 Vizrt

Glass Shader



The Glass Shader plug-in takes refraction and reflection properties from the surroundings (cube map) and blends them to create the dielectric effect of glass.

Apart from the parameters in the plug-in container, it is necessary to assign a texture as a color/basic texture and a material. Do not forget to adjust material parameters in addition, such as ambient, diffuse, specular color and shininess, to achieve the required look for your surface.

In addition, lighting by a highlight and a color texture are used to describe the material in detail. The opacity of the object can be controlled to modulate between the solidly lit surface and the transparent impression of glass.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials

Copyright © 2026 Vizrt

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Glass Shader Properties

- **Refraction Index:** Controls the refraction index of the material, and accordingly the strength of the refraction.
- **Fresnel Minimum, Fresnel Scale and Fresnel Exponent:** Affects the Fresnel effect upon the reflection.
- **Environment Reflectivity:** Manipulates the influence of the reflection color from the environment map on the resulting color.
- **Bump Height:** Manipulates the height of the created bumps on the surface.
- **Reflection Multiplier/Exposure:** Affects the exposure of the environment map.
- **Gamma of EnvMaps:** Controls the gamma correction of the environment map. This is particularly helpful when working with HDR images.
- **Blur Reflection:** States to which extent the environment reflection and refraction are blurred to create the impression of a rougher surface.
- **Opacity:** Influences the mixing of the solid material color and the dielectric glass effect (reflection and refraction).
- **Bump Map/Normal Map:** Is the texture that defines the surface structure with encoded normals.
- **Environment Map:** Is a cube map that describes the environmental surrounding used to calculate reflections and refractions.
- **Technique:** Shows a list of available techniques.
    - **Glass:** Is a technique to be used for plain glass.
    - **Glass_Texture:** Is a technique to mix a diffuse texture color into the material color.
    - **Glass_BumpStandard:** Is a technique to create bump mapping with the given normal map on the surface of the geometry that has no standard UV texture coordinates and normals.
    - **Glass_BumpStandard_Texture:** Is a technique to use a diffuse color texture and bump mapping on the surface of the geometry that has no texture coordinates and normals.
    - **Glass_BumpBinormal:** Is a technique to create bump mapping for geometry with texture coordinates and normals.
    - **Glass_BumpBinormal_Texture:** Is a technique to use a diffuse color texture and bump mapping on the surface of geometry with texture coordinates and normals.

Glass Shader Best Practices

Apart from the parameters in the plug-in container, it is necessary to assign a material and a texture as a color/basic texture. Do not forget to adjust material parameters in addition, such as ambient, diffuse, specular color and shininess, to achieve the required look for your surface.

Copyright © 2026 Vizrt

Gooch Shader



The Gooch Shader plug-in is used to calculate the lighting of the surface. Accordingly, a warm color is shown in lit areas while a cold color is used in unlit areas. A highlight is further added. Besides, a color texture may be used to add more details. This shader is particularly useful when no surrounding reflection is required.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

Gooch Shader Properties

- **Cold Color:** Assigns the color used for unlit surfaces.
- **Warm Color:** Assigns the color used for lit surfaces.
- **Back Splash:** Controls a factor to increase the perception of curvature on grazing angles. The resulting color is weighted according to the back splash factor.
- **Texture Alpha:** Defines to which amount the diffuse color texture is blended into the resulting color.
- **Highlight Alpha:** Defines to which extent the calculated highlight is blended into the resulting color.
- **Color Map 2:** Offers the possibility to assign a second diffuse color texture that is blended with the first color texture (according to the chosen technique).
- **Technique:** Shows a list of available techniques.
    - **GoochShading:** Is a technique to use Gooch shading.
    - **GoochShading_Texture:** Is a technique to mix a diffuse texture color into the material color.
    - **GoochShading_MultiTexture_Add:** Is a technique to additively blend Color Map 2 with the texture.
    - **GoochShading_MultiTexture_Blend:** Is a technique to blend Color Map 2 with the texture according to their alpha values.
    - **GoochShading_MultiTexture_Subtract:** Is a technique to subtractively blend Color Map 2 with the texture.
    - **GoochShading_MultiTexture_Modulate:** Is a technique to blend Color Map 2 by multiplying it by the texture.

Gooch Shader Best Practices

Apart from the parameters in the plug-in container, it is necessary to assign a material. Do not forget to adjust material parameters, such as ambient, diffuse, specular color and shininess. When applying a texture technique it is compulsory to assign an texture as a color/basic texture. In case of multi texturing an additional, second texture must be assigned in the corresponding Shader rollout.

Copyright © 2026 Vizrt

Lacquered Surfaces Shader



The Lacquered Surface Shader plug-in simulates smooth lacquered surfaces lit by a highlight.

On the clear coat layer, the surrounding is reflected (from a cube map). The reflection blurriness can be controlled to create the appearance of a rougher surface.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

Lacquered Surfaces Shader Properties

- **Environment Reflectivity:** Manipulates the influence of the reflection color from the environment map on the resulting color.
- **Reflection Multiplier/Exposure:** Affects the exposure of the environment map. This is especially helpful when working with HDR images.
- **Gamma of EnvMaps:** Controls the gamma correction of the environment map. This is once again particularly helpful when working with HDR images.
- **Blur Reflection:** States how much the environment reflection is blurred to create the impression of a rougher surface.
- **Fresnel Minimum, Fresnel Scale and Fresnel Exponent:** Affects the Fresnel effect upon the reflection.
- **Environment Map:** Is a cube map that describes the environmental surrounding used to calculate reflections.
- **Technique:** Shows a list of available techniques.
    - **PixelShading:** Is a technique to be used for finer details on the surface.
    - **VertexShading:** Is a technique to be used for objects in the distance or for highly tessellated objects.

Lacquered Surfaces Shader Best Practices

Apart from the parameters in the plug-in container, it is necessary to assign a material and a texture as a color/basic texture. Do not forget to adjust material parameters in addition, such as ambient, diffuse, specular color and shininess, to achieve the required look for your surface.

Copyright © 2026 Vizrt

Metal Reflection Shader



The Metal Reflection Shader plug-in lights objects with a highlight and by the reflection of the surrounding (from a cube map).

Both highlight and reflection take into account the material color of the object to generate a realistic metal effect.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

Metal Reflection Shader Properties

- **Environment Reflectivity:** Manipulates the influence of the reflection color from the environment map on the resulting color.
- **Reflection Multiplier/Exposure:** Affects the exposure of the environment map. This is especially helpful when working with HDR images.
- **Gamma of EnvMaps:** Controls the gamma correction of the environment map. This is once again particularly helpful when working with HDR images.
- **Environment Map:** Is a cube map that describes the environmental surrounding used to calculate reflections.

Metal Reflection Shader Best Practices

Apart from the parameters in the plug-in container, it is necessary to assign a material and a texture as a color/basic texture. Do not forget to adjust material parameters in addition, such as ambient, diffuse, specular color and shininess, to achieve the required look for your surface. To achieve a photo realistic, metallic look, it is recommended to assign any color but white as a specular color.

Copyright © 2026 Vizrt

Microstructure Shader



The Microstructure Shader plug-in simulates a detailed surface structure by bump mapping and lights it with the aid of a highlight.

Additionally, a clear coat layer reflects the surrounding (from a cube map). A normal map is used to define the surface structure.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials



Supported Pipelines	
Classic Render Pipeline	✓

Copyright © 2026 Vizrt

Supported Pipelines	
Viz Engine Render Pipeline	✗

Microstructure Shader Properties

- **Environment Reflectivity:** Manipulates the influence of the reflection color from the environment map on the resulting color.
- **Reflection Multiplier/Exposure:** Affects the exposure of the environment map. This is especially helpful when working with HDR images.
- **Gamma of EnvMaps:** Controls the gamma correction of the environment map. This is once again particularly helpful when working with HDR images.
- **Bump Map / Normal Map:** Is the texture that defines the surface structure with encoded normals.
- **Environment Map:** Is a cube map that describes the environmental surrounding used to calculate reflections.
- **Technique:** Shows a list of available techniques.
    - **Standard_Bump:** Is a technique to be used with geometry that has no standard UV texture coordinates and normals.
    - **Binormal_Bump:** Is a technique to be used with geometry having texture coordinates and normals.

Microstructure Shader Best Practices

Apart from the parameters in the plug-in container, it is necessary to assign a material and a texture as a color/basic texture. Do not forget to adjust also the material parameters like ambient, diffuse, specular color and shininess, to achieve the required look for your surface

Copyright © 2026 Vizrt

Monitor Shader



The Monitor Shader plug-in simulates the appearance of a flat screen display where saturation decreases at grazing angles of view. The shown color texture blends into the diffuse color of the material.

Additionally, a reflection from the surrounding (from a cube map) is added, increasing in strength towards more acute angles of view.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	❌

Monitor Shader Properties

- **Saturation/Reflection/Color Gradient:** Defines to which amount the material's diffuse color is mixed into the resulting color at grazing angles.
- **Environment Reflectivity:** Manipulates the influence of the reflection color from the environment map on the resulting color.
- **Reflection Multiplier/Exposure:** Affects the exposure of the environment map. This is especially helpful when working with HDR images.
- **Gamma of EnvMaps:** Controls the gamma correction of the environment map. This is once again particularly helpful when working with HDR images.
- **Environment Map:** is a cube map that describes the environmental surrounding used to calculate reflections.
- **Technique:** Shows a list of available techniques.
    - **Saturation_and_Reflection_Gradient:** Is a technique to influence saturation and reflection appearance alike.
    - **Reflection_Gradient:** Is a technique to influence only the reflection appearance.

Monitor Shader Best Practices

Apart from the parameters in the plug-in container, it is necessary to assign a material and a texture as a color/basic texture. For this shader, a background image should be selected as the basic texture to achieve a highly realistic monitor effect. Do not forget to adjust material parameters in addition, such as ambient, diffuse, specular color and shininess, to achieve the required look for your surface.

Copyright © 2026 Vizrt

Velvet Shader



The Velvet Shader plug-in creates a highlight along the edges of the surface is created to generate velvety impression. The highlight is controlled in using two parameters that describe the transition towards diffuse lighting. Furthermore, a surface color can be added to brighten the barely lit areas. A color texture may be used to add more surface details. This shader is particularly useful when no surrounding reflection is required.

Note: This plug-in is located in: Plugins -> Shader -> RTTAdvancedMaterials



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

Velvet Shader Properties

- **Surface-Color:** Assigns the color of the surface (underneath the velvety hairs).
- **Specular-Color:** Assigns the specular color of the material.
- **Edge Rolloff:** Defines how much of the surface color is to be seen in barely lit areas.
- **Specular low-cut and specular high-cut:** Defines the sharpness of the velvety highlight.
- **Color Map 2:** Offers the possibility to assign a second diffuse color texture that is blended with the first color texture.
- **Technique:** Shows a list of available techniques.
    - **Velvet:** Is a technique to use velvety shading,
    - **Velvet_Texture:** Is a technique to mix a diffuse texture color into the material color.
    - **Velvet_MultiTexture_Add:** Is a technique to additively blend Color Map 2 with the texture.
    - **Velvet_MultiTexture_Blend:** Is a technique to blend Color Map 2 with the texture according to their alpha values.
    - **Velvet_MultiTexture_Subtract:** Is a technique to subtractively blend Color Map 2 with the texture.
    - **Velvet_MultiTexture_Modulate:** Is a technique to blend Color Map 2 by multiplying it by the texture.

Velvet Shader Best Practices

Apart from the parameters in the plug-in container, it is necessary to assign a material. Therefore, do not forget to adjust material parameters in addition, such as ambient, diffuse, specular color and shininess. When applying a texture technique it is compulsory to assign an texture as a color/basic texture. In case of multi texturing an additional, second texture must be assigned in the corresponding Shader rollout.

Copyright © 2026 Vizrt

2.6 Scene Plug-ins

Scene plug-ins are global functions that have influence on the whole Scene. They are mainly used when external programs interface with Viz Artist.

The default path for the Scene plug-ins is *<viz install folder>\plug-in*.

- Default Scene Plug-ins
- Scene Image Plug-ins
- Lineup Scene Plug-ins
- Scene Control Plug-ins
- MultiTouch Scene Plug-ins
- Scene Scripts
- Scene Texture Plug-ins
- Scene Tools Plug-ins
- Scene RealFX Plug-ins
- Scene Libero Plug-ins

Copyright © 2026 Vizrt

2.6.1 Default Scene Plug-ins

The following Scene plug-in is located in the Default folder:

- VCF

VCF



The VCF scene plug-in works in conjunction with the VCF Parameter plug-in that allows you to create a seamless interpolated transition from a virtual camera flight (VCF) to a real camera - and conversely. This is only a relevant plug-in to set up if you have purchased the virtual studio expansion components.

You must have a real camera with data tracking enabled which is set in remote mode in the camera editor. In case you have several tracked cameras, the virtual camera interpolates its position to the real camera that is selected On Air.

> **Note:** This plug-in is located in: Plugins -> Scene plug-ins -> Default

This page contains the following topics and procedures:

- Properties
- To Animate a Virtual Camera Flight

Properties

- **Tracking Object (POS):** Defines the position or eye-point of the virtual camera flight. Use the name of the container in which center the camera should be placed.
- **Tracking Object (ROT):** Defines the direction of the virtual camera flight. Use the name of the container the camera is looking at.
- **Initialize (button):** Places the camera in position and sets the direction.

To Animate a Virtual Camera Flight

1. Start by adding the VCF scene plug-in under the Video Clip Playback Considerations plug-in tab.
2. Create a new group in your scene and add two new containers under it. These are to be the objects that define the virtual camera flight. One defines the position, the other the direction.
3. Name the containers according to the names you entered in the VCF scene plug-in (for example `T_POS` and `T_ROT`).
4. Click **Initialize** to finish.
5. Add the VCF Parameter plug-in onto the container that holds the position object.
6. Animate your virtual camera flight using the two objects to define position and direction.

- Do **not** switch the whole container invisible because the animation does not run.
- You may switch the objects to be invisible at any time.
- It is not possible to animate the roll of the camera.

Copyright © 2026 Vizrt

2.6.2 Scene Image Plug-ins

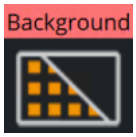
The following Scene plug-in is located in the Image folder:

- Background Clip

> **Note:** This plug-in is deprecated and should not be used anymore. Please use any of the Media Assets as a Scene Background instead.

Copyright © 2026 Vizrt

Background Clip



The Background plug-in plays back a sequence of still images (tga, tiff, etc.) in the background of your scene rather than playing an AVI file.

Since it plays the sequence from memory, the scene has a higher loading time and the drawback is that it consumes system memory which can influence the system stability. This plug-in should not be used with very large images or a large number of images. So keep track of how much memory each sequence needs so you do not run out of memory on the render engine.

Scene plug-ins are added under the Scene Settings plug-in tab.

Since all images are loaded into memory a large number of images or a large image size would require large amounts of memory. Memory can be calculated as follows: `number of images * image width * image height * 3 (or 4 for alpha).`

> **Note:** This plug-in is located in: Plugins -> Scene plug-ins -> Image

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Background Clip Properties

- **Image:** Allows you to enter the name of the image sequence or browse for it. Select the first image you would like to use in your clip. Do not use clip names with numbers (except the counter).
- **Play Mode:** Determines the method of playback. Can be set to:
  - **Once:** Plays the clip once.
  - **Loop:** Repeats the clip over and over again.
  - **Swing:** Plays the clip forwards and backwards in a loop.
- **Reverse:** Plays the clip backward when enabled.
- **Play speed:** Lets you alter the speed in percent.
- **Advanced:** Opens advanced parameters if set.
- **Stand alone image:** Plays the same clip at different speeds.
- **First image:** Select the number of the image which you want to start with.
- **Nof images:** Determines the number of images to be played in a clip, otherwise all images are played back. Nof stands for “number of images”.
- **Crop:** Crops the image in percent from the left, right, bottom and top side.
- **Scale By:** Sets whether to scale all images. When not selected an automatic textures coordinates are applied so the image fits the texture.
- **Scale (%):** Scales all images by the percentage entered.
- **Keep under:** Forces the image size. For example, if you have selected 64, the image is trimmed to the size 64 x 64.
- **Base Path:** Sets the folder for the images.

Copyright © 2026 Vizrt

- **Format:** Loads the file in original condition if **Image** format is selected. Please select another format if you want load a black and white image (“Alpha”) or full color image (“RGB”, “RGBA”: with alpha value).
- **Play, Stop and Reinitialize:** Enables you to play, stop and reinitialize the clip.

To Add a Background Image Clip

1. Create a directory and populate it with image files, for instance `abc001.png`, `abc002.png`, `abc003.png`, etc.

- The images are played as a sequence based on the filename including numbers.
- After you have created the images, load the first image of the sequence and the clip is now visible in the Scene Editor.

1. Another possibility is to create a file with a **.vln* extension. This file includes the base path and also the names of the images to load. In this case the images must not have a counter number in their filename. You can handle this file as an ordinary text file.
2. Load the *.vln* file instead of loading an image file located in a directory.

Example

```
BASE_PATH ‘C:/clip/images' {
‘radar_200504110800.png’ 2005_04_11_10:00
‘radar_200504110815.png’ 2005_04_11_10:15
‘radar_200504110830.png’ 2005_04_11_10:30
‘radar_200504110845.png’ 2005_04_11_10:45
‘radar_200504110900.png’ 2005_04_11_11:00
‘radar_200504110915.png’ 2005_04_11_11:15
‘radar_200504110930.png’ 2005_04_11_11:30
‘radar_200504110945.png’ 2005_04_11_11:45
}
```

See Also

- Image Clip

Copyright © 2026 Vizrt

2.6.3 Lineup Scene Plug-ins

The following Scene plug-in is located in the Lineup folder:

- Tree Status

Tree Status



The Tree Status plug-in is needed by the Lineup template in Viz Pilot to collect information about the scene hierarchy.

The Tree Props container plug-in is required for use of the Tree Status plug-in. Move the Tree Props plug-in onto the group holding the transformation which is to be controlled by Viz Pilot. Viz Pilot is then able to build its own internal tree properties list.

> **Note:** This plug-in is located in: Plugins -> Scene plug-ins -> Lineup

Tree Status Properties

- **Result:** Contains the entire information about the complete scene tree.
- **S2VResult:** Holds the information X-, Y-, Z-translation and the Y-rotation given in the card file virtual Studio. S2V stands for *scene to virtual*.
- **Command:** Allows a Viz command to take place.
- **Initialize (button):** Initializes the *Result* field.
- **InitializeS2V (button):** Initializes the *S2VResult* field.

Copyright © 2026 Vizrt

2.6.4 Scene Control Plug-ins

The following scene plug-in is located in the Control folder:

- Presets

Presets



The Presets scene plug-in manages Super Channel-based preset master scenes. It has functionality for creating, updating and removing named presets, as well as the ability to transition between such presets. The presets define different layouts of the Super Channels in the scene. Preset state and the transition animations between presets are represented by directors in the stage. The Presets plug-in also provides a command interface that is utilized by the Media Sequencer Engine for management and playout of named presets.

> ⚠ **Note:** This plug-in is located in: Plugins -> Scene plug-ins -> Control

Presets Properties

- **Show advanced:** Toggles advanced properties.
  - **Current preset:** The last preset taken, cut or selected in the dropdown.
  - **Command in:** Receives commands in order to perform an operation on the preset master scene.
  - **Command result:** Returns a result using the result parameter when a command is successfully executed.
  - **Command error:** Sets an error message using the error parameter on invalid commands or command failure.
- **Add superchannel:** Adds a Super channel and a default content transition director to the scene, and applies Super Channel settings needed for playout from the Media Sequencer.
- **Preset name:** Specifies the name of a new preset.
- **No layout transitions (new preset):** Specifies whether the new preset should be transitionless. When selected, new presets added do not support layout transitions to and from other presets. When transitioning to and from such presets, the plug-in always falls back to cut (immediately applies the layout without animation).
- **Add preset:** Adds a new preset with the name specified by the preset name parameter. The layout of the new preset is determined by the layout of Super Channels in the scene at the moment of adding the preset. This does not overwrite existing presets.
- **Preset:** Determines the preset for editing.
- **Discard changes:** Discards the changes made to the Super Channel layout and shows the layout for the preset currently selected in the dropdown list.
- **Apply changes:** Applies changes to the preset currently selected in the dropdown list. This updates the preset director and preset transition directors with the layout currently visible in the editor.
- **Remove preset:** Removes the preset currently selected in the dropdown list from the scene.

Creating a Preset Master Scene

This section describes how to create a preset master scene from scratch with Viz Artist. Start with an empty scene.

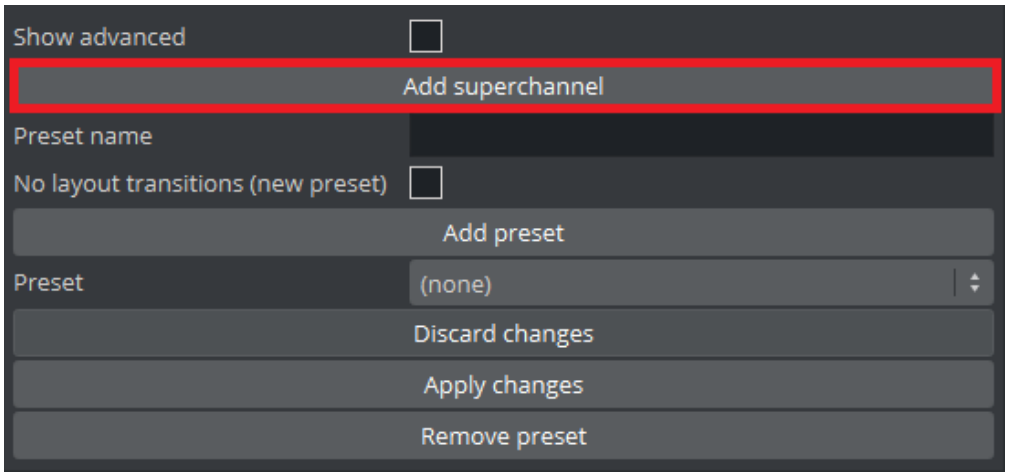
Adding the Presets Scene Plug-in

The first step is to add the Presets scene plug-in to the scene.

Copyright © 2026 Vizrt

Adding Super Channels

The preset scene manages named presets that define layouts of Super Channels. We need to add Super Channels to the scene. Super Channels can be added to the scene one by one with the plug-in:



It is possible to add Super Channels to the scene manually in the DVE media assets tab, however this is discouraged because the default Super Channel configuration does not support playout from the Media Sequencer. The following settings are set when adding Super Channels with the Presets plug-in:

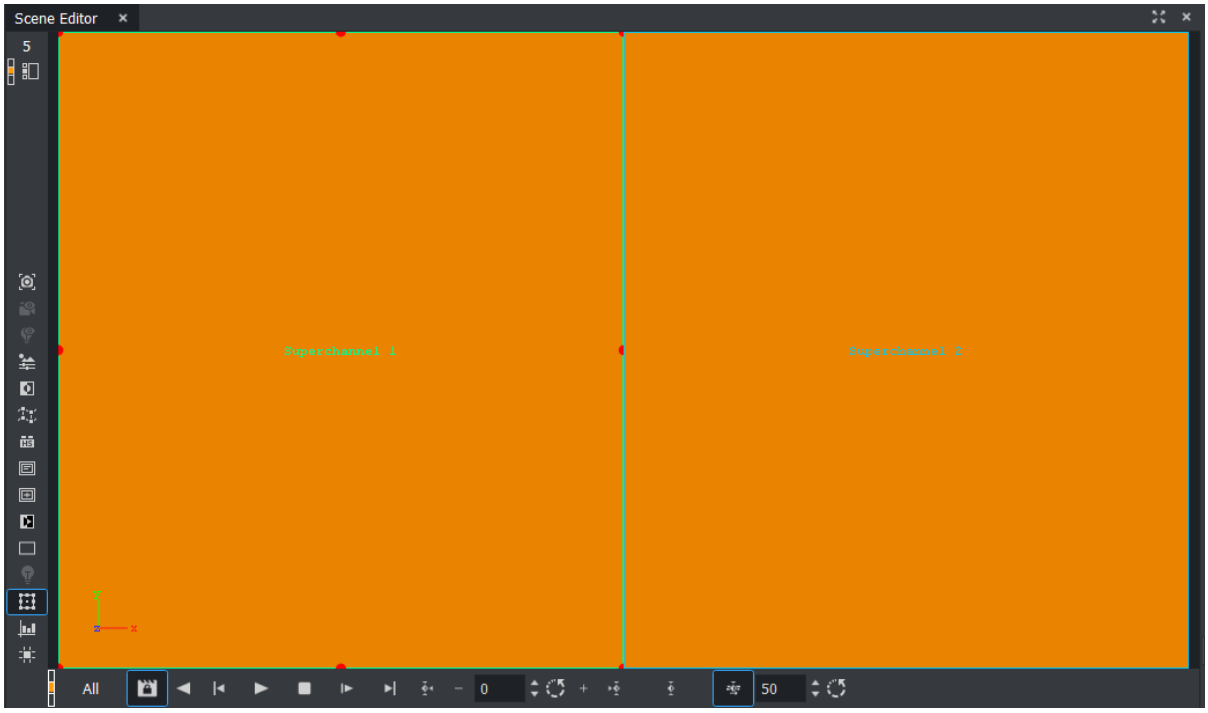
- Super Channel asset type is set to *Texture.*
- Super Channel start mode is set to *Loaded / Cleared.*
- Directors for transitioning from subchannels A to B and from B to A are set to content transition directors generated by the plug-in.
- Start mode for Super Channel subchannels are set to *Transition Start.*
- Image zoom mode is set to *Scale to fit min* for Super Channel subchannels.
- Image border color is set to transparent (alpha 0%).

Creating Presets

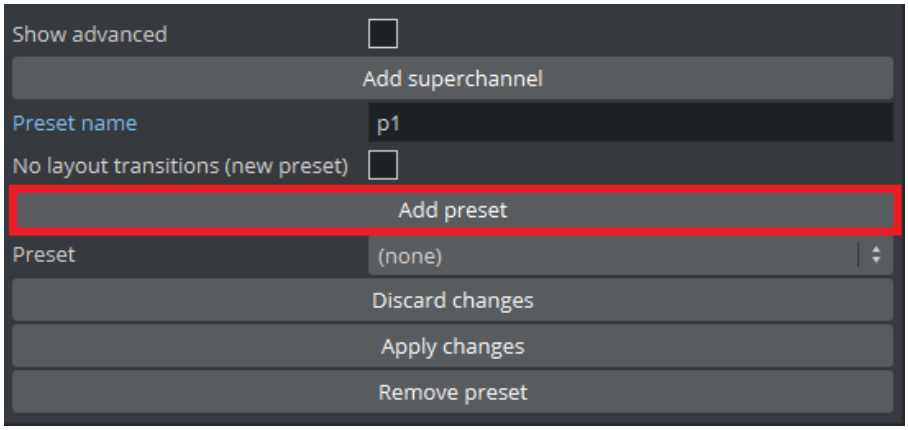
When the Presets plug-in and Super Channels are added to the scene, we can create presets, defining different layouts of Super Channels.

To create a new preset, first arrange the Super Channels to the desired layout and set the desired Super Channel properties:

Copyright © 2026 Vizrt



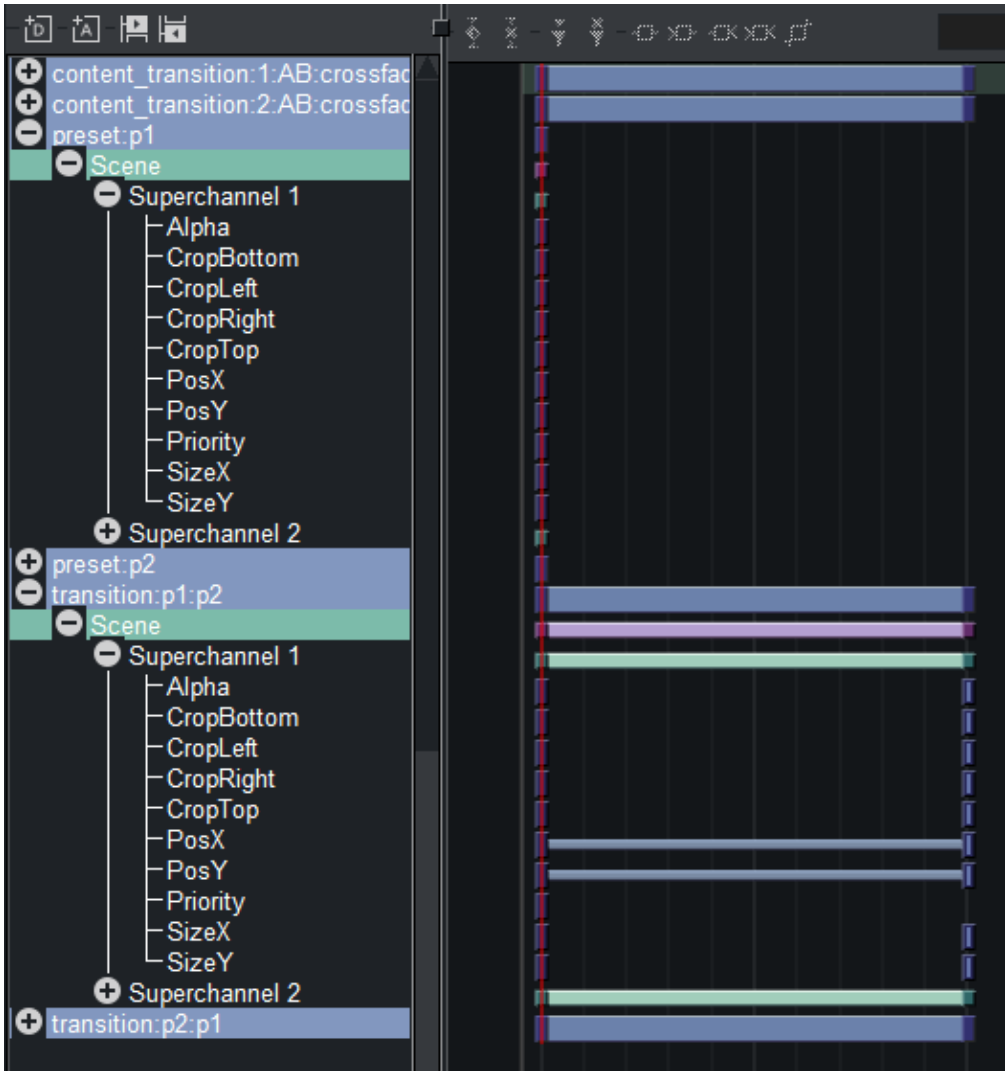
When the desired layout is set, use the Presets plug-in interface to add a new preset:



Specify a unique name for the new preset before pressing the **Add preset** button. Existing presets are not overwritten. For adding more presets to the scene, repeat the process of adjusting the Super Channel layout, choosing a new preset name and pressing the **Add preset** button. If it is desired to add a preset without layout transitions to and from other presets, the option **No layout transitions (new preset)** should be enabled before pressing the **Add preset** button.

The presets and transitions between presets are now represented by directors in the stage, created by the plug-in:

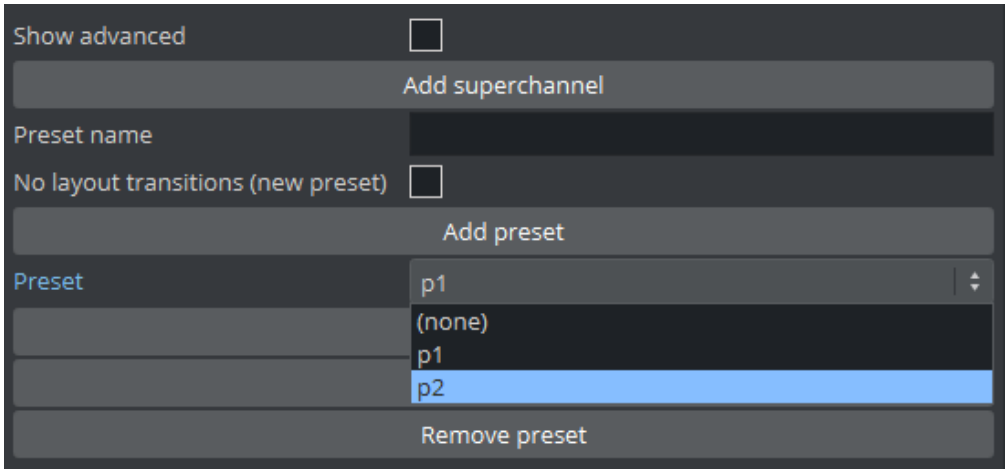
Copyright © 2026 Vizrt



Editing Presets

To edit a preset, first make sure it is selected in the dropdown list:

Copyright © 2026 Vizrt



Now you can rearrange the Super Channels to the desired layout.

Note: Selecting a different preset discards any unapplied changes currently made in the editor. Changes can be applied to the selected preset with the **Apply changes** button, or discarded with the **Discard changes** button.

Copyright © 2026 Vizrt

2.6.5 MultiTouch Scene Plug-ins

The following Scene plug-ins are located in the MultiTouchComp folder:

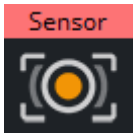
- MtSensor

All the MultiTouch Plugins Container plug-ins require the MtSensor to be available. If a MultiTouch Plugins Container plug-in is added to a Scene it automatically adds the MtSensor to the Scene if it is not there.

The plug-ins in this folder are inactive by default. To view the folder and use the plug-ins, set all plug-ins to active in **Plug-ins** (see the **Configuring Viz** section of the Viz Engine Administrator Guide).

Copyright © 2026 Vizrt

MtSensor



The MtSensor plug-in manages global telestration and plug-in settings. The MtSensor plug-in is the scene plug-in that must be available for any MultiTouch feature to work. It is automatically added to the Scene if any of the Container or Geometry plug-ins that require it are inserted in the Scene. It can also be dragged from the Server Panel under **Plugins > SP > MultiTouchComp**, to the plug-in tab under Scene Settings.

The MtSensor plug-in also provides a screen-wide telestration facility. It can be controller through plug-in parameters and buttons that are available for scripting.

> ⚠ **Note:** This plug-in is located in: Plugins -> Scene plug-ins -> MultiTouchComp

This page contains the following topics and procedures:

- MtSensor Configuration
- Events

MtSensor Configuration

- **Active:** Enables/disables telestration input from user.
- **Minimum Telestration Width:** Assigns minimum brush width based on pressure applied to sensor.
- **Maximum Telestration Width:** Assigns maximum brush width based on pressure applied to sensor.
- **Color:** Assigns a color for subsequent telestration drawing.
- **Id:** Provides additional context in the handler script, specify a string that identifies any notifications dispatched by this plug-in. This is often included as an argument for the event so a common script may handle events from a number of plug-ins.
- **Set Shared Memory:** Enables shared memory to be updated for the plug-in notifications if set to `On`.
- **Shared Memory Prefix:** Sets a 'prefix name' to be prepended to the shared memory variables maintained by the plug-ins notifications. For plug-ins that maintain multiple fields each field name has the prefix prepended to it followed by a dot, so as to mimic member access to an object. For example, if the prefix is *Obj*, the fields *field1* and *field2* would be identified with the strings *Obj.field1* and *Obj.field2*. The shared memory field 'Obj' is also maintained and is simply an integer that is modified every time any of its subfields is updated.
- **Shared Memory Type:** Selects the shared memory area to update. Can be either **Global**, **Scene**, or **Distributed**.
- **SetDataPool:** Shows 'you wish' plug-in notifications to set a DataPool variable.
- **DataPoolVariable:** Shows the name of the DataPool variable one wishes to have set.
- **Init:** Clears current telestration and make Telestration active.
- **Clear:** Clears the contents of the attached telestration.

Events

The MtSensor plug-in does not generate any events for the scripting engine. The scene-wide telestration feature emits the same events that the MtTelestrator plug-in emits.

Copyright © 2026 Vizrt

2.6.6 Scene Scripts

The Script plug-ins is a folder to save created Scene Script plug-ins.

A script can be saved as its own script plug-in, and used in future Scenes (see Create Script-based plug-ins in the Viz Artist User Guide). To save a Script plug-in, drag a compiled script into the Script plug-ins folder.

Note: Script plug-ins are saved to *<viz data folder>\Scriptplug-ins*.

Copyright © 2026 Vizrt

2.6.7 Scene Texture Plug-ins

The following Scene plug-in is located in the Texture folder:

- Graffiti

Graffiti



The Graffiti scene plug-in allows telestration on top of scenes in the screen space. Telestration is done by drawing with a brush shape using a mouse or a 6DOF interface. Graffiti can recognize some rendered shapes and replace the hand-drawn items with the recognized shapes (for example, circle, ellipse, cross, arrow).

The graffiti scene plug-in is not intended for use on video output. To use graffiti textures on video output, use the GraffitiTex container plug-in.

> **Note:** This plug-in is located in: Plugins -> Scene plug-ins -> Texture

This page contains the following topics and procedures:

- Graffiti Properties
- To Create a Scene Level Graffiti Effect
- 6DOF (6 Degrees of Freedom)

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Graffiti Properties

- **Active:** Enables/disables drawing.
- **Multi Telestration:**
- **Stylus:**
- **Pressure Min [%]:**
- **Pressure Max [%]:**
- **Pointer Source:** Supports input from the mouse, or from another plug-in via a dispatcher interface. This button selects whether the plug-in listens to the mouse or the dispatcher for input.
- **Mask Container:** Points to either a container or a group of containers. These containers mask the area over which the brush is rendered.
- **Alpha:** Adds transparency to the brush.
- **Key:** Selects whether the brush is rendered in the key signal
- **Brush Type:** Selects color or eraser brush.
- **Brush Image:** Determines the shape of the brush using an image (optional). If empty, a round brush is used.
- **Brush Width:** Determines width of the brush in pixels. Visible only if the color brush is selected.
- **Eraser Brush Width:** Determines width of the eraser in pixels. Visible only if the eraser brush is selected.
- **Brush Color:** Selects the color of the brush.
- **Recognize Shapes:** Enables/disables shape recognition mode.
- **Recognize Ellipse:** Specifies whether shape recognition tries to recognize ellipse shape.
- **Recognize Circle:** Specifies whether shape recognition tries to recognize circle shape.
- **Recognize Cross:** Specifies whether shape recognition tries to recognize cross shape.
- **Shape Delay:** Sets number of frames to wait from mouse up before trying to recognize shapes.

Copyright © 2026 Vizrt

- **Draw Arrow:** Specifies whether non-recognizable shapes are converted to an arrow.
- **Arrow Length:** Determines length of arrow head.
- **Arrow Width:** Determines width of arrow head.
- **Circle Rad:** Sets radius of the circle replacing a recognized circle. If zero, the radius of the recognized circle is used.
- **Cross Width:** Sets width of the cross replacing the recognized cross shape. If zero, the width of the recognized cross shape is used.
- **Max. Undos:** Sets maximum number of undo operations.
- **Clear (button):** Clears the canvas.
- **Undo (button):** Undoes an operation.

To Create a Scene Level Graffiti Effect



Copyright © 2026 Vizrt

Add the plug-in to the Plug-in Panel in **Scene Settings**, set the plug-in properties, set Viz Artist in On Air Mode and start drawing.

6DOF (6 Degrees of Freedom)

6DOF is a tool for artists, script writers and plug-in developers to position and orientate objects in the 3D space (three translation and three rotation axes => 6DOF).

6DOF works directly with the built in Scene-Grids (see Grid Tool-bar) and uses them to calculate the intersection of the projected user input (mouse or touchscreen position) with the current grid plane. As soon as a touch or click input is detected, it triggers the corresponding 6DOF callbacks (OnButtonDown6DOF, OnMove6DOF, etc...) and passes the calculated position and rotation.



For example: This could be used to move players on a football field interactively without doing any math at all.

Additionally, 6DOF has a distribution mechanism which can send the user input to multiple engines, at the same time. This is configured in the **Global Input** of the Viz Configuration (see the Viz Engine Administrator Guide)

See Also

- GraffitiTex (container plug-in)

Copyright © 2026 Vizrt

2.6.8 Scene Tools Plug-ins

The following Scene plug-ins are located in the Tools folder:

- LOD Manager
- Scene Synchronized Properties

Copyright © 2026 Vizrt

LOD Manager



The LOD Manager scene plug-in works in conjunction with the Level of Detail container plug-in which is rendering objects with different details depending on the camera range; however, an integration with the LOD Manager is not required as Level of Detail is capable of initializing the parameter values by itself. The LOD Manager plug-in is used to change the visual view of the complete scene during rendering depending on the camera and scale settings.

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Tools

This page contains the following topics and procedures:

- Level Of Detail (LOD) Manager Properties
- To Use the LOD Manager Plug-in

Level Of Detail (LOD) Manager Properties

- **Reference Zoom:** Sets the reference zoom that is used to correlate switching ranges based on the actual camera zoom value.
- **Range Scale:** Sets the global range scale which is used to scale all LOD ranges on all Level of Detail plug-ins in the scene.
- **Reference Width:** Not used.
- **Reference Height:** Not used.

To Use the LOD Manager Plug-in



1. Start by adding the LOD Manager scene plug-in under the Scene Settings' plug-in tab.

Copyright © 2026 Vizrt

2. Add a group container to your scene tree and add at least two new sub containers of the group container.
3. Add a Sphere geometry to the sub containers as it has many triangles.
4. Add a material to each sub container.
5. Open the Sphere editor for the all sub containers and set different Tesselation values for each to change the number of triangles and therefore the level of detail of an object.

> ⚠ **Note:** All spheres should, for the sake of example, be placed in the same position.

6. Add the Level of Detail plug-in to the group container.
7. Open the LOD editor and change the *Range* parameter values provided by the Level of Detail plug-in.
   For example, set Range 0 to `500`, Range 1 to `1000` and so on.
   - These settings affects the instant of time when an object 1, object 2 is rendered, which you have placed in step 3.
   - The high resolution object is rendered at first and if the camera to object distance is increasing, the next object is rendered because now a low-resolution object has near the same visual quality despite fewer triangles. Consider that the visual quality depends on your settings and object graduation.
8. Finally, change the LOD Manager parameters *Reference Zoom* and *Range Scale* to adjust the scene view additionally.

Copyright © 2026 Vizrt

Scene Synchronized Properties



The Scene Synchronized Properties plug-in can be used to synchronize all property changes within the same scene loaded on a cluster of Viz Engines. Synchronization needs to be configured for each Viz Engine taking part in the clustered setup. This plug-in works on a per scene basis. A container plug-in is available, allowing synchronization to take place on a per container basis.

This feature works for all container properties and plug-in parameters, and requires that the Engines render the same scene, referencing the same UUIDs. This allows designers to work with different versions of the same scene, without any need to update the changed IDs in external applications. If a parameter change is transferred to another engine then the container UUID of the last loaded scene is used for the update. If new UUIDs are required, a new scene needs to be created, either from scratch, or by merging the complete scene into a container which is then split into a new scene.

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Tools

To Synchronize Scene Property Changes on Multiple Viz Engines

- Configure the involved Viz Engines for synchronization.
- Create the scene that should be synchronized.
- Add the **Scene Synchronize Properties** plug-in to the Scene Setting's Plug-in Panel. No scripting is required for synchronization.

See Also

- Synchronized Properties Container plug-in
- For more information about how to synchronize multiple Viz Engines, see the **Global Input** page in the **Configuring Viz** section of the Viz Engine Administrator Guide.

Copyright © 2026 Vizrt

2.6.9 Scene RealFX Plug-ins

- Collision Manager

Collision Manager



Collision Manager Properties

- **Spheres:** Sets the number of spheres
- **Boxes:** Sets the number of boxes.

Copyright © 2026 Vizrt

2.7 Shaders (Viz Engine Renderer)

The default path for the Viz Engine Shader plug-ins is: *<viz install folder>\plugin*.

- Particle System
- XR Draw

2.7.1 Particle System



The Particle System plug-in simulates and renders complex visual effects and natural phenomena (for example, fire, smoke, rain, snow, fog, dust, and explosions).

The Viz Engine Render Pipeline supports GPU-accelerated particle systems.

Supported Pipelines	
Classic Render Pipeline	✗
Viz Engine Render Pipeline	✓

Note: The Classic Render Pipeline particle modifiers like Vortex or Turbulence are not supported.

This page contains information on the following topics:

- Particle System Properties
  - Common Properties
  - Emission
  - Force
  - Texture
  - Size
  - Color
  - Alpha

Particle System Properties

Common Properties

- **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.

Note: The sorting of particles should always be turned on when alpha images are in use to guarantee a correct blending.

Copyright © 2026 Vizrt

Emission

- **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**
        - **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**
        - **Cube Dimension X, Y, Z** is the cube size on each axis.
    - **Face**
        - **Face Dimension X, Z** is the face (plane) size on each axis.

Force

- **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.
- **Rotation Speed (Z axis):** Sets particle angular velocity around the Z axis.
- **Rotation Speed Spread (Z axis):** Controls the randomness for the particle rotation speed (Z axis) of each particle. A random value between *-Particle rotation speed spread (Z axis)* and *Particle rotation speed spread (Z axis)*, is added to the particle rotation speed (Z axis).

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.

Copyright © 2026 Vizrt

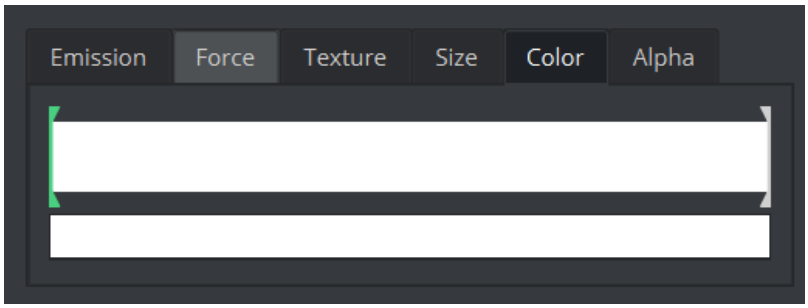
Texture

- **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.

Size

- **Size Scale X,Y:** Scales for the size of the particle, where a particle is represented by a rectangle.
- **Size Spread X,Y:** 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.
- **Size X, Y:** 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

- **Alpha:** Controls the particle alpha during its lifetime. Horizontal axis is the lifetime in seconds, vertical axis controls the alpha value.

Copyright © 2026 Vizrt

2.7.2 XR Draw



XR Draw is a Viz Engine Shader plug-in that dynamically generates and renders an extruded mesh geometry along a path.

It can be used for 3D Telestration in an XR-Set Environment.

> **Note:** This plug-in is located in: Plugins -> Shader -> Material

XR Draw Properties

- **Point X**: Current input position in X.
- **Point Y**: Current input position in Y.
- **Point Z**: Current input position in Z.
- **Threshold**: Minimum distance between points on path.
- **Radius**: Radius of the extruded base geometry.
- **Contour points**: Number of contour points of the extruded base geometry.
- **Color**: Sets the color used to render the geometry.
- **Texture**: Determines texture used to render the geometry.
- **Scale U**: Sets scale factor for the generated UV coordinates in U direction.
- **Scale V**: Sets scale factor for the generated UV coordinates in V direction.
- **Start drawing**: Starts recording points and generating the geometry along the recorded path.
- **Stop drawing**: Stops recording points.
- **Clear Points**: Clears all recorded points and reset the drawn geometry.

> **Note:** The recorded points are not being saved, as the purpose of the plug-in is for live drawing.

Copyright © 2026 Vizrt

3 Viz Engine Extension Plug-ins

Extension plug-ins provide details about settings available through Viz Engine and Viz Script. Extension plug-ins are similar to normal plug-ins, but they are loaded during Viz Engine startup and are alive until shutdown (where regular plug-ins are bound to the scene lifetime). It is possible to provide data or services for scripts and plug-ins or control Viz Engine.

This section contains information on the following topics:

- Plug-in API
- Script API
- Extension Plug-ins
    - viz_command_over_websocket
    - viz_probel
    - viz_sealevel
    - viz_tally_tsl
    - viz_webrtc

3.1 Lifecycle

The extension plug-ins have a different lifecycle then normal plug-ins. They are loaded on startup and life until Viz Engine shuts down. This allows extension plug-ins to provide all kind of services which are not necessarily bound to a scene.

Copyright © 2026 Vizrt



3.2 Communication

There are many ways to communicate with an Extension. One of the most common ways is to use SHM. The extension can provide data by reading/writing the key/values of Viz Engine SHM.

Another method introduced with these Extensions is the Message API which is described below. SHM API and Message API are provided by Viz Engine, but there are many other methods like using a socket connection.

Copyright © 2026 Vizrt

3.3 Message API

The Message API is similar to a REST API. A service/extension can provide multiple endpoints and clients can request data by sending requests to such endpoints. Like the REST API, such endpoints are described by a path. Extension endpoints are prefixed with */Extensions/<extension-name>*. For example, the extension named *Greetings* could have an endpoint *greet* which could be requested with */Extensions/Greetings/greet*.

A message request can be sent from plug-ins or extensions and consists of the endpoint and a message body (which can be empty). Viz Engine then dispatches those messages to the corresponding extensions. The extension then responds with a message which is dispatched by Viz Engine back to the plug-in/script.

The whole communication is asynchronous. Requests can be sent from any thread. *PLUGIN_REQUEST/ PLUGIN_RESPONSE* callbacks are always called on the main thread. This way of communicating also allows extensions to not immediately respond to requests. Furthermore, it is possible to send a response without receiving requests. This is useful to provide some kind of notification mechanism.

3.4 Sample Extension

The simple extension plug-in with Message support to greet. This plug-in responds to any Message received with "*Hello <request-message>*".

```
#include <PluginLib/evPlugin_PLUGINEXTERN.h>
#include <PluginLib/evPlugin_Extensions.h>
#include <Pluginlib/evPlugin_PLUGINFUNC.h>
#include <Pluginlib/evPlugin_DynInterface.h>
#include <Pluginlib/api_functions.h>

struct PLUGIN_DATA
{
};

VIZPLUGIN_API int
PLUGIN_INIT_EX()
{
    // Register plugin
    evRegisterPlugin("SampleExtension");
    evRegisterPluginFolder("Extension");
    evRegisterPluginType(EV_EXTENSION);

    evRegisterPluginUACfolders();

    evRegisterTotalSize(sizeof(PLUGIN_DATA));

    return 0;
}

VIZPLUGIN_API void
PLUGIN_INIT_FUNCTION(void* pvDataPtr)
{
```

Copyright © 2026 Vizrt

```
    new (pvDataPtr)(PLUGIN_DATA)();
}

VIZPLUGIN_API void
PLUGIN_CLEANUP_FUNCTION(void* pvDataPtr)
{
    auto pluginPtr = reinterpret_cast<PLUGIN_DATA*>(pvDataPtr);
    pluginPtr->~PLUGIN_DATA();
}

VIZPLUGIN_API
void PLUGIN_REQUEST(void* pvDataPtr, int senderId, const char* location, const char*
msg)
{
    auto pluginPtr = reinterpret_cast<PLUGIN_DATA*>(pvDataPtr);
    std::string response = std::string("Hello ") + msg;
    _api__respond(senderId, location, response.c_str());
}
```

3.5 Sample Plug-in

A Viz Engine plug-in that communicates with the Sample Extension above. On initialization, a message request is sent to the extension and the response set to the plug-in parameter *Output*, which can be viewed in Viz Artist.

The output is "*Hello Plugin*".

```
struct PLUGIN_DATA
{
    char* output;
};

VIZPLUGIN_API void PLUGIN_INIT()
{
    evRegisterPlugin( "SamplePlugin" );
    evRegisterPluginFolder( "Sample" );
    evRegisterPluginType( EV_FUNCTION_CONTAINER );
    evRegisterPluginUACfolders();

    evRegisterParameterText("Output", "", 400, 600);
    evRegisterTotalSize(sizeof(PLUGIN_DATA));
}

VIZPLUGIN_API void PLUGIN_INIT_FUNCTION(void *dataptr)
{
    // cast dataptr to our parameter structure
    auto data = (struct PLUGIN_DATA*)dataptr;
    new (data) PLUGIN_DATA;

    auto pluginObjectId = _api__get_plugin_objectid();
    _api__request(pluginObjectId, "/Extensions/SampleExtension", "Plugin");
```

Copyright © 2026 Vizrt

```
}

VIZPLUGIN_API void PLUGIN_RESPONSE(void* pvDataPtr, const char* location, const char*
msg)
{
    struct PLUGIN_DATA* data = (struct PLUGIN_DATA*)pvDataPtr;
    std::strcpy(data->output, msg);
    evSendGuiRefresh();
}
```

3.6 Sample Script

Similar to the Sample plug-in, this script sends a message request to the extension.

The output is:

"*/Extensions/SampleExtension*"

"*Hello Script*"

```
Request("/Extensions/SampleExtension", "Script")

sub OnResponse(location as string, message as string)
    Println(location)
    Println(message)
end sub
```

3.7 Related Documents

- *Viz Engine Administrator Guide*: Contains information on how to install the Viz Engine software and supported hardware.

3.8 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

3.9 Plug-in API

3.9.1 Callbacks

PLUGIN_EXTENSION_CONFIGURE

Signature

int PLUGIN_EXTENSION_CONFIGURE(void pvDataPtr, const char* configStr)*

Required

No

Description

Called to configure the extension plug-in. This function is called if a configuration file for this extension exists with a <*configStr*> parameter.

PLUGIN_EXTENSION_CONFIG_GET

Signature

const char PLUGIN_EXTENSION_CONFIG_GET(void* pvDataPtr)*

Required

No

Description

Called to get the current configuration of the extension plug-in. The return value is expected to be the content of the serialized configuration.

PLUGIN_REQUEST

Signature

void PLUGIN_REQUEST(void pvDataPtr, int connection, const char* location, const char* msg)*

Required

No

Copyright © 2026 Vizrt

Description

Called whenever a message is received for this extension.

> **Note:** This function is always called on the main thread.

- connection: object-id of sender (must be used for _api__respond)
- location: message endpoint
- msg: message payload

PLUGIN_RESPONSE

Signature

void PLUGIN_RESPONSE(void pvDataPtr, const char* location, const char* msg)*

Required

No

Description

Callback for incoming responses [Messages].

> **Note:** This function is always called on the main thread.

After calling _api__request this function is called with the response message.

- location: message endpoint
- msg: message payload

3.9.2 Functions

_api__get_plugin_objectid

Signature

int _api__get_plugin_objectid();

Description

Returns the object ID of the current plug-in. Can be used for _api__request.

Copyright © 2026 Vizrt

_api__request

Signature

void _api__request(int iObjectID, const char location, const char* msg)*

Description

Sends a message request.

Note: Function can be called at any time and is thread-safe.

- iObjectID: object-id of the current plug-in. Can be retrieved by calling _api__get_plugin_objectid.
- location: Message endpoint
- msg: optional message payload

Example: If an extension provides a message endpoint of "/Extensions/Sample/greet".
You could make a requests with _api__request(myObjectId, "/Extensions/Sample/greet", "")

_api__respond

Signature

void _api__respond(int iObjectID, const char location, const char* msg);*

Description

Can be used to respond to a message request. If a message is received by getting PLUGIN_REQUEST called. You can respond with this function (immediately or at another time).

- iObjectID: object-id which got passed by PLUGIN_REQUEST
- location: location which got passed by PLUGIN_REQUEST
- msg: the response message body

Note: Function can be called at any time and is thread-safe.

Copyright © 2026 Vizrt

3.10 Script API

3.10.1 Functions

Signature

Request(location as String, message as String)

Description

Sends a message request. On response the callback, *OnResponse* is called.

3.10.2 Callbacks

Signature

sub OnResponse(location as string, message as string)

Description

Response callback which gets called for each message request issued by this script.

Copyright © 2026 Vizrt

3.11 Extension Plug-ins

The default path for the Extension plug-ins is: *<viz install folder>\plug-in\<plug-in name.vip>*

The following Extension plug-ins are located in the Default folder:

- viz_command_over_websocket
- viz_probel
- viz_sealevel
- viz_tally_tsl
- viz_webrtc

Copyright © 2026 Vizrt

3.11.1 viz_command_over_websocket

The viz_command_over_websocket plug-in allows Viz Engine to interpret websocket messages as Viz Engine commands.

Open a websocket connection and send for each command one message. The extension then responds with the corresponding command response from Viz Engine. The command can be prepended with an integral number which is used as Command ID. This way the response can be associated with the request message. If left empty, the config parameter **default-id** is used instead.

Configuration

To Configure this plugin create a file called ***{VizEngine-Config-dir}****/extensions/CommandOverWebsocket.json*. This file is used for every running instance of Viz Engine. To have a specific config for a running instance, you can create configuration files like ***{VizEngine-Config-dir}****/extensions/CommandOverWebsocket-**{Instance-Number}**.json*. For this extension, this is required if multiple instances are started, as it is not possible to open multiple websocket servers on the same port.

Supported config parameters:

- **port:** Port where the websocket server will listen.
- **default-id:** Default Command-ID used if the incoming message does not include it.

The **CommandOverWebsocket.json** file could then look like this:

```
{
    "default-id": "-1",
    "port": 6900
}
```

Limitations

Currently, the websocket server uses the ws protocol which is not protected. Using this extension in OEM versions is not possible.

Copyright © 2026 Vizrt

3.11.2 viz_probel

The viz_probel plug-in allows the dynamic assignment of IP streams (2110) to the Viz Engine input channel.

Viz Engine manages a number of external sources to map them to a smaller number of available input channels. Viz Engine maintains control over which input channels have fixed sources from startup and which input channels can request a new source. The viz_probel plug-in maintains a table that maps a source to the Viz Engine input channel. Since each input channel in the Viz Engine corresponds to a physical Matrox connection. The plug-in effectively maps external sources to a particular Matrox physical connection.

The external component that communicates with viz_probel plug-in is usually a Virtual Studio Monitor (VSM) which communicates with the viz_probel plug-in via SW-P-08 general remote protocol.

VSM has a matrix composed of the sources and its corresponding output. This matrix state is saved in the viz_probel plug-in.

The viz_probel plug-in has the following tables:

- VSM output and corresponding sources.
- VSM output and corresponding Viz Engine input channel.
- Matrox physical connection and Viz Engine input channel. Configured by the user.
- Matrox physical connection and VSM output. Configured by the user.

Message Interface

Communication via message passing is supported which is similar to a REST API. Messages can be sent via `Request(location, message)` in the Viz Script.

The response is asynchronous and is only given to the plug-in user when there is change in the internal mappings.

Register

The following must be called at least once to be able to connect to the VSM.

Location: */Extensions/Probel/register*

Response:

```
[{"error": false}]
```

Unregister

The following must be called to disconnect from VSM.

Location: */Extensions/Probel/unregister*

Response:

```
[{"error": false}]
```

Copyright © 2026 Vizrt

Sources

The following returns the corresponding Viz Engine input channel. This generally changes the matrix in VSM but not always. {} can also be *releaseall* or *requestall*.

releaseall attempts to release all sources while *requestall* attempts to request all available sources.

Location: */Extensions/Probel/sources/{}*

Response:

```
[{"error": false}]
```

The following releases a channel in the matrix in VSM.

Location: */Extensions/Probel/sources/{}/release*

Response:

```
[{"error": false}]
```

The following releases an all channel in the matrix in VSM except a few of them. {} can have multiple sources separated by "-". Example */Extensions/Probel/sources/releaseall/except/3-4-5-6*

Location: */Extensions/Probel/sources/releaseall/except/{}*

Response:

```
[{"error": false}]
```

Debug

The following is used for debugging SW-P-08 implementation. If {} is `0`, it sends an ImplementationRequest. If {} is `1`, it sends a CrossPointTallyDumpRequest.

Location: */Extensions/Probel/debug/{}*

Response:

```
[{"error": false}]
```

The following is used for debugging SW-P-08 implementation. If {} is `0`, it sends a CrossPointInterrogate, the next {} is a parameter indicating the output. If {} is `1`, it sends a AllSourceNamesRequest, the next {} is a parameter indicating the names length.

Location: */Extensions/Probel/debug/{}/{}*

Response:

Copyright © 2026 Vizrt

```
[{"error": false}]
```

The following is used for debugging SW-P-08 implementation. If {} is `0`, it sends a CrossPointConnect, the next {} are parameters indicating the output and source. If {} is `1`, it sends a SingleSourceNameRequest, the next {} are parameters indicating the names length and source.

Location: */Extensions/Probel/debug/{}/{}/{}*

Response:

```
[{"error": false}]
```

Response

The following is how the asynchronous response looks like. {} corresponds to the source.

Location: */Extensions/Probel/sources/{}/change*

Response:

```
[
    {
    "channelId": 0,
    "sourceId": 0
    }
]
```

Configuration

To Configure this plug-in create a file called ***{VizEngine-Config-dir}****/extensions/Probel.json*. This file is used for every running instance of Viz Engine. To have a specific configuration for a running instance you can create configuration files like ***{VizEngine-Config-dir}****/extensions/Probel-****{Instance-Number}****.json*.

Supported config parameters:

- **address:** Defines the port that should be used in VSM if connection mode is configured as server (depending on the connection-mode configuration), or the address and port of VSM if configured as client.
- **backup-address:** Defines the port that should be used in VSM backup if connection mode is configured as server (depending on the connection-mode configuration), or the address and port of VSM backup if configured as client.
- **connection-mode:** Determines whether the plug-in should act as a server or client. `0` for server or `1` for client.
- **fixed-entry-offset:** Sources that should not be changed. This sources can not be released by plug-in the user.
- **delay-in-fields:** Sets the field delay before giving the response to the plug-in user.
- **disconnected:** Defines what should be considered as disconnected sources. Several sources can be defined separated by comma.

Copyright © 2026 Vizrt

- **range:** Limits the range at which the plug-in user can operate. This is used in the event multiple Viz Engine instances are operating using the same VSM server but must operate only on a subset of sources.
- **vizch-vsmout:** Shows which Viz Engine input channel corresponds to which VSM output.
- **vizch-matroxch:** Shows which Viz Engine input channel corresponds to which Matrox physical connector. This should match with the configuration in the usual Viz Engine configuration file.
- **extended-message:** Enables extended support for Probel which adds support for a large matrix with more than 1024 sources.

The Probel *.json* file could then look like this:

```
{
    "address": "10.29.39.11:4211",
    "backup-address": "0.0.0.0:27015",
    "connection-mode": "1",
    "delay-in-fields": "10",
    "disconnected": "1022,65535",
    "fixed-entry-offset": "4",
    "range": "0-15",
    "vizch-matroxch":
"0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,26,27,28,29,30,31",
    "vizch-vsmout":
"0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,3
1"
}
```

Sample Script

The following is an example of most of the functionality presented above:

```
sub OnInitParameters()
RegisterParameterInt("req", "ReqSource", 1, 1, 1023)
RegisterPushButton("request", "Request Source", 1)
RegisterPushButton("release", "Release Source", 2)
RegisterPushButton("releaseall", "Release All", 3)
RegisterPushButton("requestall", "Request All", 4)
RegisterPushButton("register", "Register", 5)
RegisterPushButton("unregister", "Unregister", 6)
end sub

sub OnResponse(location as string, message as string)
    PrintLn "OnResponse triggered..."
    Println location
    Println message
end sub

sub OnExecAction(buttonId As Integer)
if buttonId = 1 Then
Request("/Extensions/Probel/sources/"&cstr(getParameterInt("req")-1), "")
elseif buttonid = 2 Then
```

Copyright © 2026 Vizrt

```
Request("/Extensions/Probel/sources/"&cstr(getParameterInt("req")-1)&"/release", "")
elseif buttonid = 3 Then
for x = 0 to 31
println "Release : "&x
Request("/Extensions/Probel/sources/"&cstr(x)&"/release", "")
next
elseif buttonid = 4 Then
for x = 0 to 32
println "Request : "&(x mod 32)
Request("/Extensions/Probel/sources/"&cstr((x mod 32)), "")
next
elseif buttonid = 5 Then
Request("/Extensions/Probel/register", "")
elseif buttonid = 6 Then
Request("/Extensions/Probel/unregister", "")
end if
end sub
```

As per the documentation, the user needs to call first register to be able to use the other functions.

Debug

To enable debug logs go to the usual Viz Engine configuration file and enable ACE logging:

`ACELogParams = -f OSTREAM|STDERR|VERBOSE_LITE|VERBOSE -p STARTUP|SHUTDOWN|~TRACE|DEBUG|INFO|WARNING|NOTICE|ERROR|CRITICAL|ALERT|EMERGENCY`

This allows for the user to see the raw messages that are being sent and received between the VSM and the plug-in.

Troubleshooting

To be able to troubleshoot it is important to understand the console log messages.

Suppose we have the following tables:

(MatroxCh#, VSMOut#): (0, 15), (1, 0), (2, 3), (3, 12), (4, 4), (5, 5), (6, 6), (7, 7).
(MatroxCh#, VizCh#): (0, 1), (1, 0), (2, 3), (3, 2), (4, 4), (5, 5), (6, 6), (7, 7).
(VSMOut#, VizCh#): (0, 0), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (12, 2), (15, 1).
(VSMOut#, SourceId#): (0, 3), (1, 2), (2, -1), (3, 10), (4, 0), (5, 12), (6, 8), (7, 5), (8, 0), (9, -1), (10, -1), (11, -1), (12, 11), (13, -1), (14, -1), (15, 4).

All operations are done only with the tables **(VSMOut#, VizCh#)** and **(VSMOut#, SourceId#). (MatroxCh#, VSMOut#)** and **(MatroxCh#, VizCh#)** are redundant.

First case: Requested SourceId# has a VSMOut# that is associated with VizCh#:

```
INFO: VizEngine-0[15996]:viz::swp08::Probel_SWP08_Handler::PrintMap: (MatroxCh#,
VSMOut#): (0, 15), (1, 0), (2, 3), (3, 12), (4, 4), (5, 5), (6, 6), (7, 7).
INFO: VizEngine-0[15996]:viz::swp08::Probel_SWP08_Handler::PrintMap: (MatroxCh#,
VizCh#): (0, 1), (1, 0), (2, 3), (3, 2), (4, 4), (5, 5), (6, 6), (7, 7).
INFO: VizEngine-0[15996]:viz::swp08::Probel_SWP08_Handler::PrintMap: (VSMOut#,
VizCh#): (0, 0), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (12, 2), (15, 1).
```

Copyright © 2026 Vizrt

```
INFO: VizEngine-0[15996]:viz::swp08::Probel_SWP08_Handler::PrintMap: (VSMOut#,
SourceId#): (0, 3), (1, 2), (2, -1), (3, 10), (4, 0), (5, 12), (6, 8), (7, 5), (8,
0), (9, -1), (10, -1), (11, -1), (12, 11), (13, -1), (14, -1), (15, 4).
INFO:
VizEngine-0[15996]:viz::swp08::Probel_SWP08_Handler::RequestChannelFromSourceId: foun
d vsmout (0) in (VSMOut#, SourceId#) that has an associated vizch (0) in (VSMOut#,
VizCh#) with given sourceId (3).
INFO:
VizEngine-0[15996]:viz::swp08::Probel_SWP08_Handler::RequestChannelFromSourceId: (VSM
Out#, SourceId#) = (0, 3) (VSMOut#, VizCh#) = (0, 0).
```

Second case: Requested SourceId# does not have a VSMOut# that is associated with VizCh#. In this case we need to do a **CrossPointConnect** using VSMOut that is both associated with a non-used SourceId# and is also associated with VizCh#. In this case we could not find it: None of the VSMOut in **(VSMOut#, VizCh#)** has a SourceId# = -1 in **(VSMOut#, SourceId#)**. A release is needed first.

```
INFO: VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::PrintMap: (MatroxCh#,
VSMOut#): (0, 15), (1, 0), (2, 3), (3, 12), (4, 4), (5, 5), (6, 6), (7, 7).
INFO: VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::PrintMap: (MatroxCh#,
VizCh#): (0, 1), (1, 0), (2, 3), (3, 2), (4, 4), (5, 5), (6, 6), (7, 7).
INFO: VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::PrintMap: (VSMOut#,
VizCh#): (0, 0), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (12, 2), (15, 1).
INFO: VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::PrintMap: (VSMOut#,
SourceId#): (0, 3), (1, 2), (2, -1), (3, 10), (4, 0), (5, 4), (6, 8), (7, 5), (8, 0),
(9, -1), (10, -1), (11, -1), (12, 3), (13, -1), (14, -1), (15, 4).
INFO:
VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::RequestChannelFromSourceId: vsmo
ut NOT found in (VSMOut#, SourceId#) that has an associated vizch in (VSMOut#,
VizCh#) with given sourceId (12).
INFO:
VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::RequestChannelFromSourceId: coul
d not find first vsmout in (VSMOut#, VizCh#) with sourceId (-1) in (VSMOut#,
SourceId#).
```

Third case: From the above state, after releasing we can request again.

```
INFO: VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::PrintMap: (MatroxCh#,
VSMOut#): (0, 15), (1, 0), (2, 3), (3, 12), (4, 4), (5, 5), (6, 6), (7, 7).
INFO: VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::PrintMap: (MatroxCh#,
VizCh#): (0, 1), (1, 0), (2, 3), (3, 2), (4, 4), (5, 5), (6, 6), (7, 7).
INFO: VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::PrintMap: (VSMOut#,
VizCh#): (0, 0), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (12, 2), (15, 1).
INFO: VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::PrintMap: (VSMOut#,
SourceId#): (0, -1), (1, 2), (2, -1), (3, 10), (4, 0), (5, 4), (6, 8), (7, 5), (8,
0), (9, -1), (10, -1), (11, -1), (12, -1), (13, -1), (14, -1), (15, 4).
INFO:
VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::RequestChannelFromSourceId: vsmo
ut NOT found in (VSMOut#, SourceId#) that has an associated vizch in (VSMOut#,
VizCh#) with given sourceId (12).
```

Copyright © 2026 Vizrt

```
INFO:
VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::RequestChannelFromSourceId: foun
d first vsmout (0) in (VSMOut#, VizCh#) with sourceId (-1) in (VSMOut#, SourceId#).
INFO:
VizEngine-0[16012]:viz::swp08::Probel_SWP08_Handler::RequestChannelFromSourceId: (VSM
Out#, VizCh#) = (0, 0) (VSMOut#, SourceId#) = (0, 12).
```

3.11.3 viz_sealevel

The viz_sealevel plug-in allows Viz Engine to provide GPIO data coming from SeaLevel Devices for plug-ins and scripts.

Both SeaLevel PCI and ethernet Devices are supported. The GPIO data is provided in via SHM key/values and via message passing.

SHM Interface

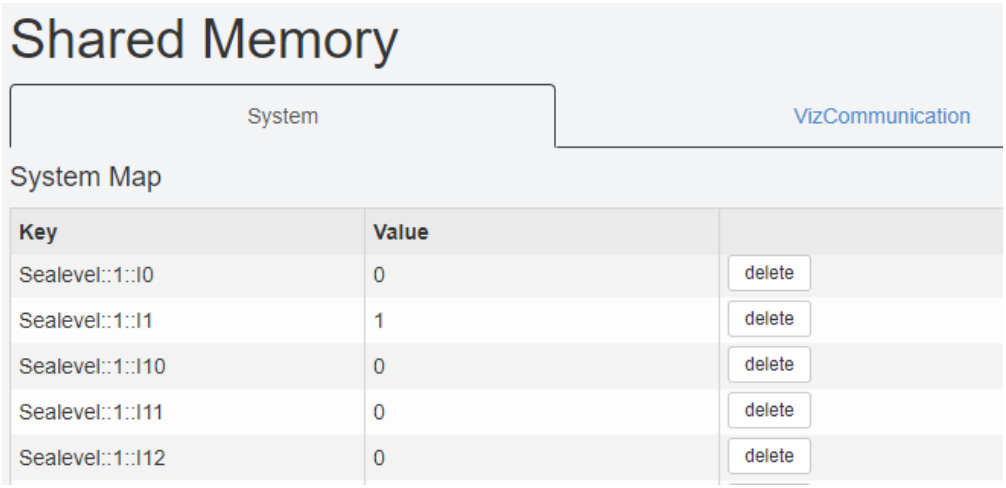
Each GPIO pin is mapped to a SHM key.

Format

The pins are mapped with following format:

- **Inputs:** `Sealevel::<device-id>::I<pin-id>::<0 or 1>`
- **Outputs:** `Sealevel::<device-id>::O<pin-id>::<0 or 1>`

SMM Values in Viz Engine REST Interface



Inputs

These SHM values are only for reading and are updated when a input pin changes. By registering to SHM changes this can then be used to perform tasks in scripting or plug-ins.

Outputs

These values are meant to set from inside scripts or plug-ins (for example to switch lights on and off).

Copyright © 2026 Vizrt

Message Interface

The Viz Engine plug-in API supports communication via message passing which is similar to a REST API. Messages can be send via `Request(location, message)` in Viz Script.

List Devices

Location: */Extensions/Sealevel/devices*
Response:

```
[
    {
        "id": "1",
        "name": "410io_",
        "status": "connected",
        "status-description": "Connected"
    }
]
```

Get Device Info

Location: */Extensions/Sealevel/devices/1*
Response:

```
{
    "id": "1",
    "name": "410io_",
    "status": "connected",
    "status-description": "Connected",
    "pins": [
        {
            "id": "0",
            "is-input": true,
            "value": 0
        },
        {
            "id": "1",
            "is-input": true,
            "value": 0
        },
        {
            "id": "2",
            "is-input": true,
            "value": 0
        },
        {
            "id": "3",
            "is-input": true,
```

Copyright © 2026 Vizrt

```
            "value": 0
        },...
    ]
}
```

Register for Pin Changes

Location: */Extensions/Sealevel/devices/1/register*
Response:

```
{
    "error": false
}
```

If then a pin changes the script or plug-in would receive such a response:
Location: */Extensions/Sealevel/devices/1/change*
Response:

```
{
    "device": 1,
    "pin": 1,
    "value": true
}
```

Get All Pin Data for a Given Device

Location: */Extensions/Sealevel/devices/1/pins*
Response:

```
[
    {
        "id": "0",
        "is-input": true,
        "value": 0
    },
    {
        "id": "1",
        "is-input": true,
        "value": 0
    },
    {
        "id": "2",
        "is-input": true,
        "value": 0
    }, ...
]
```

Copyright © 2026 Vizrt

Get Pin Data for a Given Device and Pin

Location: */Extensions/Sealevel/devices/1/pins/0*
Response:

```
{
    "id": "0",
    "is-input": true,
    "value": 0
}
```

Set Pin Data for a Given Device and Pin

Location: */Extensions/Sealevel/devices/1/pins/0*
Message:

```
{
    "value": 1
}
```

Response:

```
{
    "id": "0",
    "is-input": true,
    "value": 1
}
```

Configuration

To Configure this plugin create a file called ***{VizEngine-Config-dir}****/extensions/Sealevel.json*. This file is used for every running instance of Viz Engine. To have a specific config for a running instance you can create config files like ***{VizEngine-Config-dir}****/extensions/Sealevel-****{Instance-Number}****.json*.

Supported config parameters:

- **auto-reconnect:** Automatically reconnect to device.
- **reconnect-interval:** Interval for reconnecting.
- **discover-ethernet-devices:** Automatically detect SeaLevel network devices.
- **ethernet-devices:** Array of SeaMax devices connected via Ethernet.
    - **address:** IP address.
    - **name:** Associated name for this Card.
- **seaio-devices:** Array of SeaIO device (PCI-Cards).
    - **adapter:** Adapter number for PCI-Card (For two installed PCI-Cards possible adapters would be 0 and 1).
    - **name:** Associated name for this Card.

Copyright © 2026 Vizrt

- **serial-devices:** Array of SeaMax devices connected via serial interface.
  - **port:** COM port for this device.
  - **name:** Associated name for this Card.

The *Sealevel.json* file would then look like this:

```
{
    "auto-reconnect": true,
    "discover-ethernet-devices": true
    "ethernet-devices": [
        {
            "port": "COM1",
            "name": "serial_card_0"
        }
    ],
    "reconnect-interval": 60000,
    "seaio-devices": [
        {
            "adapter": 0,
            "name": "pci_card_0"
        }
    ],
    "serial-devices": [
        {
            "port": "COM1",
            "name": "serial_card_0"
        }
    ]
}
```

Sample Scripts

List Devices

```
sub getDevices()
    Request("/Extensions/Sealevel/devices", "")
end sub

sub onDevices(devices as Json)
    if devices.Size == 0 Then
        exit sub
    end if

    dim devCount as Integer = devices.Size - 1
    for i = 0 to devCount
        dim deviceObj as Json = devices.At(i)
        Println(deviceObj.Dump())
    Next
```

Copyright © 2026 Vizrt

```
end sub

sub OnResponse(location as string, message as string)
    if location == "/Extensions/Sealevel/devices" Then
        dim j as Json
        j.Load(message)
        onDevices(j)
    end if
end sub

getDevices()
```

Get Pin Values

```
sub getPinValue(devId as String, pinId as String)
    Request("/Extensions/Sealevel/devices/" & devId & "/pins/" & pinId, "")
end sub

sub OnResponse(location as string, message as string)
    dim j as Json
    j.Load(message)

    onPinValue(j.GetString("id"), j.GetInteger("value"))
end sub

sub onPinValue(pinId as String, value as Integer)
    Println("Pin " & pinId & " value = " & value)
end sub

' Inputs
for i = 0 to 15
    getPinValue("0", "" & i)
next

' Outputs
for i = 16 to 31
    getPinValue("0", "" & i)
next
```

Set Output Pins

```
sub getPinValue(devId as String, pinId as String)
    Request("/Extensions/Sealevel/devices/" & devId & "/pins/" & pinId, "")
end sub

sub setPinValue(devId as String, pinId as String, value as Integer)
    dim payload as String = "{\"value\": " & value & "}"
```

Copyright © 2026 Vizrt

```
    Request("/Extensions/Sealevel/devices/" & devId & "/pins/" & pinId, payload)
end sub

sub OnResponse(location as string, message as string)
    dim j as Json
    j.Load(message)

    onPinValue(j.GetString("id"), j.GetInteger("value"))
end sub

sub onPinValue(pinId as String, value as Integer)
    Println("Pin " & pinId & " value = " & value)
end sub

setPinValue("0", "16", 1)
setPinValue("0", "18", 1)

' Outputs
for i = 16 to 31
    getPinValue("0", "" & i)
next
```

Listen for Pin Changes

```
sub OnResponse(location as string, message as string)

    if location.StartsWith("/Extensions/Sealevel/devices") and
location.EndsWith("change") Then
        dim j as Json
        j.load(message)
        onPinChanged(j.GetInteger("pin"), j.GetInteger("value"))
    end if
end sub

sub onPinChanged(pinId as Integer, value as Integer)
    Println("Pin " & pinId & " changed to " & value)
end sub

Request("/Extensions/Sealevel/devices/0/register", "")
```

Copyright © 2026 Vizrt

3.11.4 viz_tally_tsl

The viz_tally_tsl plug-in allows Viz Engine to control tallies via message passing from plug-ins and scripts using the TSL UMD Protocol Specification.

The extension acts as client and connects to an TSL server. Currently only V3.1 of TSL UMD Protocol is implemented for both TCP and UDP. The communication for TSL UMD Protocol is unidirectional. So it is not possible to get the current status from the Server. Therefore, the extensions keeps record of the current state. On startup, the extensions assumes that all tallies are off. From this point on, the extensions stores which tallies were enabled. If the server connection is lost, then the extension attempts to reconnect. As soon as it is reconnected, it resends all tally states to the server.

Message Interface

The Viz Engine Plug-in API now supports communication via message passing which is similar to a REST API. Messages can be sent via `Request(location, message)` in the Viz Script.

List Servers

Location: */Extensions/Tsl_Tally/servers*
Response:


```
[
    {
        "host": "120.120.120.120",
        "id": 0,
        "links": [
            "self",
            {
                "href": "/Extensions/Tsl_Tally/servers/0"
            },
            "tally",
            {
                "href": "/Extensions/Tsl_Tally/servers/0/tally"
            }
        ],
        "name": "tsm_test_server",
        "port": 6112,
        "status": "disconnected"
    },
    {
        "host": "localhost",
        "id": 1,
        "links": [
            "self",
            {
                "href": "/Extensions/Tsl_Tally/servers/1"
            },
            "tally",
            {
```

```
                "href": "/Extensions/Tsl_Tally/servers/1/tally"
            }
        ],
        "name": "local",
        "port": 34567,
        "status": "connected"
    }
]
```


Get Tally Status for a Given Server and Address

Location: */Extensions/Tsl_Tally/servers/:server-id/tally/:address*
Response:


```
[
    {
        "active": false,
        "brightness": 1.0
    },
    {
        "active": false,
        "brightness": 1.0
    },
    {
        "active": false,
        "brightness": 1.0
    },
    {
        "active": false,
        "brightness": 1.0
    }
]
```


Get Tally Status for a Given Server and Address

Location: */Extensions/Tsl_Tally/servers/:server-id/tally/:address*
Message:


```
[
    {
        "active": true,
        "brightness": 1.0
    },
    {
        "active": true,
        "brightness": 1.0
    },
    {
        "active": false,
```


Copyright © 2026 Vizrt

```
        "brightness": 1.0
    },
    {
        "active": false,
        "brightness": 1.0
    }
]
```

Response:

```
[
    {
        "active": true,
        "brightness": 1.0
    },
    {
        "active": true,
        "brightness": 1.0
    },
    {
        "active": false,
        "brightness": 1.0
    },
    {
        "active": false,
        "brightness": 1.0
    }
]
```

Get Tally Status for a Given Server, Address and Tally

Location: */Extensions/Tsl_Tally/servers/:server-id/tally/:address/:tally-index*
Response:

```
{
    "active": false,
    "brightness": 1.0
}
```

Set Tally Status for a Given Server, Address and Tally

Location: */Extensions/Tsl_Tally/servers/:server-id/tally/:address/:tally-index*
Message:

```
{
    "active": true,
    "brightness": 1.0
```

```
}
```

Response:

```
{
    "active": true,
    "brightness": 1.0
}
```

Configuration

To Configure this plugin create a file called ***{VizEngine-Config-dir}****/extensions/Tsl_Tally.json*. This file is used for every running instance of Viz Engine. To have a specific configuration for a running instance you can create configuration files like ***{VizEngine-Config-dir}****/extensions/Tsl_Tally-****{Instance-Number}****.json*.

Supported config parameters:

- **reconnect-interval-seconds:** Interval in seconds to reconnect to a disconnected server.
- **servers:** Array of TSM servers.
    - **host:** Hostname of the TSM server.
    - **port:** Port of the TSM server.
    - **name:** Associated name for this server.

The *Tsl_Tally.json* file could then look like this:

```
{
    "reconnect-interval-seconds": 60,
    "servers": [
        {
            "host": "10.211.1.153",
            "name": "tsm_adrian",
            "port": 6112
        },
        {
            "host": "localhost",
            "name": "local",
            "port": 34567
        }
    ]
}
```

Sample Scripts

Switch Tally on

```
sub turnOnTally(serverId as Integer, address as Integer, tally as Integer)
```

Copyright © 2026 Vizrt

```
    Request("/Extensions/Tsl_Tally/servers/"&serverId&"/tally/"&address&"/"&tally&"",
"{\"active\": true}")
end sub

sub turnOffTally(serverId as Integer, address as Integer, tally as Integer)
    Request("/Extensions/Tsl_Tally/servers/"&serverId&"/tally/"&address&"/"&tally&"",
"{\"active\": false}")
end sub

turnOnTally(0, 0, 0)
turnOffTally(0, 0, 0)
```

Switch Tally on with Brightness to 0.4

```
sub setTally(serverId as Integer, address as Integer, tally as Integer)
    Request("/Extensions/Tsl_Tally/servers/"&serverId&"/tally/"&address&"/"&tally&"",
"{\"active\": true, \"brightness\": 0.4}")
end sub

setTally(0, 0, 0)
```

Copyright © 2026 Vizrt

3.11.5 viz_webrtc

The viz_webrtc plug-in allows Viz Engine to use peer-to-peer communication to be able to stream Viz Engine output to multiple clients. It opens a websocket connection between the extension and the client (browser) and then waits for the SDP response from the client to establish the peer connection. When the connection is complete, the video track is sent to the client as soon as a scene is loaded in Viz Engine. This extension also takes incoming input events from the client to be handled by Viz Engine allowing the user to trigger events, perform transformation on objects or trigger script events.

Important: WebRTC output does not properly support interlaced resolutions. Jitter is expected when using these resolutions.

WebRTC in Software I/O Only Mode

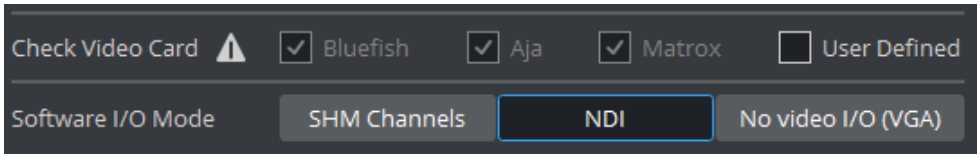
Licensing

To operate viz_webrtc in Software I/O mode **Parallel Outputs** needs to be licensed.

Configuration

To configure Viz Engine using WebRTC without Matrox support:

- Uncheck *User Defined* in Check Video Card.
- Set the Software I/O Mode to **NDI**.



WebRTC with Matrox Topology

Installation

The plug-in can be used in combination with a Matrox topology board, which means that DSX.core client must be installed if no Matrox board is installed.

Important: This procedure needs to be followed after installing DSX.core:

1. Unregister *mvfDsxCore.dll*.
   a. Click **Start > Run** (or use the Windows command line: **Search > CMD >** (Right click) **Run as Administrator**).
   b. Type `REGSVR32 /U "C:\Program Files\Matrox DSX-TopologyUtils\System64\mvfDsxCore.dll"` and press **ENTER**.
2. Shut down *X.info* in the task manager.

Copyright © 2026 Vizrt

3. Delete *mvfDsxCore.dll* from the folder *C:\Program Files\Matrox DSX-TopologyUtils\System64*.
4. Start *X.info*.

Licensing

To operate viz_webrtc **Parallel Outputs** needs to be licensed. If DSX.core is used, an additional licenses for either **DSX.core HD License** or **DSX.core UHD License** is necessary.

Common Configuration for Both Modes

The Viz Engine output enables WebRTC stream if a valid license is available and configured and the output is enabled in the Viz Configuration File.

Information: To use WebRTC output, it has to be explicitly enabled by setting `WebrtcOut1.Enable = 1` in section **MATROX_CONFIG** in your Viz Configuration File.

It is also possible to enable WebRTC output or check if it is enabled using the following commands:

```
CONFIGURATION*CHANNELS*WEBRTCOUT_0*ENABLE SET 1
CONFIGURATION*CHANNELS*WEBRTCOUT_0*ENABLE GET
```

Regarding to websocket port configuration, it's possible to change the default port (`9092`) in Viz Configuration File.

Information: Settings for WebRTC websocket port can be found in section **MATROX_CONFIG** in your Viz Configuration File: `WebrtcOut1.WebsocketPort` .

It is also possible to change it or check the current port using the following commands:

```
CONFIGURATION*CHANNELS*WEBRTCOUT_0*PORT SET <port_number>
CONFIGURATION*CHANNELS*WEBRTCOUT_0*PORT GET
```

Note: This configuration must be done up front. It does not take effect during runtime.

WebRTC Versions

The following table lists the WebRTC version Viz Engine is built with:

Viz Engine	WebRTC
5.5	

Copyright © 2026 Vizrt

Viz Engine	WebRTC
5.4	m125
5.3	m125
5.2	m85
5.1	m85

WebRTC Client Support

The HTTP server is not implemented by viz_webrtc plug-in. The following information is to help every user to build their webrtc client to be able to interact with webrtc server.

Configuration

To configure webrtc client to establish a peer connection with webrtc server built into the viz_webrtc plug-in, it is necessary to create a websocket client that is responsible for handling the peer connection and also notifying the plug-in via a data channel for input handling.

WebSocket Settings

Regarding to websocket configuration, the default port is `9092` in Viz Configuration File. Make sure that you keep the same websocket port on client side when you update it on Viz Configuration File.

```
this.connection = new WebSocket("ws://127.0.0.1:9092/ws")
```

Peer Connection Settings

To establish a peer connection we need to set the ice server with the same configuration that was defined in viz_webrtc plug-in.

```
const peerConnectionConfig = {
iceServers: [
{
urls: "stun:stun.l.google.com:19302",
},
],
};
```

Once the RTCPeerConnection is created it is necessary to create an SDP answer on client. As the SDP offer was already created in the remote peer (server side) the websocket connection can be used to listen for incoming offers. Once that is done, the two peers have set both the local and remote session descriptions so they know the capabilities of the remote peer. This doesn't mean that the connection between the peers is ready. For this to work, we need to collect the ICE candidates at each peer and transfer to the other peer.

Copyright © 2026 Vizrt

To receive the remote tracks that were added by the server, we register a listener on the local *RTCPeerConnection* listening for the track event. The *RTCTrackEvent* contains an array of *MediaStream* objects that have the same *MediaStream.id* values as the peer's corresponding local streams.

```
this.peerConnection.ontrack = (e) => {
        console.log("Remote stream added");
        this.playerElement = document.getElementById('remote-video');
        this.playerElement.srcObject = e.streams[0];
      };
```

More information can be found here https://webrtc.org/getting-started/peer-connections on how to create a webrtc peer connection on client side.

Events Handling Support

To be able to notify the webrtc server when a new input event is triggered, it is necessary to follow the defined message standard so that the server can recognize the content of the received message.

Supported Actions Label

```
const Action = {
      Move: 0,
      Rotate: 1,
      Zoom: 2,
      KeyUp: 3,
      KeyDown: 4
  };
```

Supported Input Events Label

```
const InputEvent = {
    Keyboard: 0,
    MouseMove: 1,
    MouseLeft: 2,
    MouseRight: 3,
    MouseWheel: 4,
    Touch: 5,
    ButtonClick: 6
  };
```

Key up

```
let key = {
            "key": e.keyCode,
            "inputEvent": {
```

Copyright © 2026 Vizrt

```
              "event": InputEvent.Keyboard,
              "action": Action.KeyUp,
            },
            "type":"key"
          };
          console.log(key)
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(key));
```

Key down

```
let key = {
            "key": e.keyCode,
            "inputEvent": {
              "event": InputEvent.Keyboard,
              "action": Action.KeyDown,
            },
            "type":"key"
          };
          console.log(key)
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(key));
```

Mouse down

```
// Mouse left down

if(e.which === 1) {
          this.leftClicked = true;
          let coordinates = {
            "coordinates":{
                "x":this.x,
                "y":this.y,
                "delta":this.mousedelta
            },
            "inputEvent": {
              "event": InputEvent.MouseLeft,
              "action": Action.KeyDown,
            },
            "type":"coordinates"
          };
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(coordinates));
        }

// Mouse right down
        if(e.which===3){
          this.rightClicked = true;
```

Copyright © 2026 Vizrt

```
          let coordinates = {
            "coordinates":{
                "x":this.x,
                "y":this.y,
                "delta":this.mousedelta
            },
            "inputEvent": {
              "event": InputEvent.MouseRight,
              "action": Action.KeyDown,
            },
            "type":"coordinates"
          };
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(coordinates));
        }

// Mouse wheel down
        if(e.which===2){
          let coordinates = {
            "coordinates":{
                "x":this.x,
                "y":this.y,
                "delta":this.mousedelta
            },
            "inputEvent": {
              "event": InputEvent.MouseWheel,
              "action": Action.KeyDown,
            },
            "type":"coordinates"
          };
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(coordinates));
        }
```

Mouse up

```
// Mouse left up

if(e.which === 1) {
          this.leftClicked = true;
          let coordinates = {
            "coordinates":{
                "x":this.x,
                "y":this.y,
                "delta":this.mousedelta
            },
            "inputEvent": {
              "event": InputEvent.MouseLeft,
              "action": Action.KeyUp,
            },
            "type":"coordinates"
```

```
          };
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(coordinates));
        }

// Mouse right up
        if(e.which===3){
          this.rightClicked = true;
          let coordinates = {
            "coordinates":{
                "x":this.x,
                "y":this.y,
                "delta":this.mousedelta
            },
            "inputEvent": {
              "event": InputEvent.MouseRight,
              "action": Action.KeyUp,
            },
            "type":"coordinates"
          };
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(coordinates));
        }

// Mouse wheel up
        if(e.which===2){
          let coordinates = {
            "coordinates":{
                "x":this.x,
                "y":this.y,
                "delta":this.mousedelta
            },
            "inputEvent": {
              "event": InputEvent.MouseWheel,
              "action": Action.KeyUp,
            },
            "type":"coordinates"
          };
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(coordinates));
        }
```

Mouse Wheel zoom

```
let coordinates = {
          "coordinates":{
              "x":this.x,
              "y":this.y,
              "delta":this.mousedelta
          },
          "inputEvent": {
```

```
            "event": InputEvent.MouseWheel,
            "action": Action.Zoom,
          },
          "type":"coordinates"
        };
        this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(coordinates));
```

Mouse Move

```
let coordinates = {
          "coordinates":{
              "x":this.x,
              "y":this.y,
              "delta":this.mousedelta
          },
          "inputEvent": {
            "event": InputEvent.MouseMove,
            "action": Action.Rotate,
          },
          "type":"coordinates"
        };
        this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(coordinates));
```

Client Resize

```
const resolution = {
            "resolution":{
                "width":this.innerWidth,
                "height":this.innerHeight
            },
            "type":"resolution"
          };
          this.resizeVideo();
          this.connection && this.connection.readyState == 1 &&
this.connection.send(JSON.stringify(resolution));
```

Note: It's important that webrtc server is aware of client size changes. The client resolution affects directly the coordinates that are sent to the server, so you must notify the server each time the resolution changes. Furthermore, the coordinates sent to the server must refer to the video element that is tracking the webrtc stream that comes from the server. For this reason, each time the browser is resized, the video element must also be resized accordingly.

Copyright © 2026 Vizrt

4 DataPool Plug-ins



The DataPool Plug-ins provide details about settings available through its configuration user interface within Viz Artist.

4.1 Related Documents

- *Viz Artist User Guide*: Contains information on how to install Viz Engine and create graphics scenes in Viz Artist.

4.2 Feedback and Suggestions

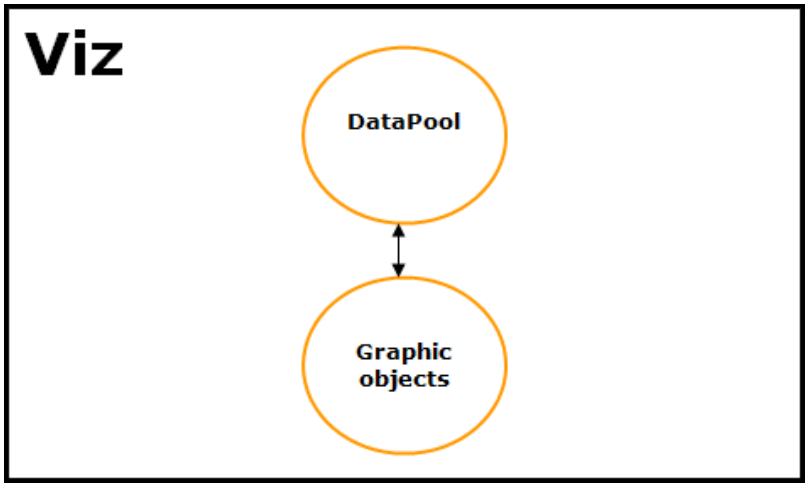
We encourage suggestions and feedback about our products and documentation. To give feedback and/or suggestions, please contact your local Vizrt customer support team at www.vizrt.com.

Copyright © 2026 Vizrt

4.3 Introduction to DataPool Plug-ins

DataPool is a mechanism for managing data in Viz. The mechanism is implemented through a group of container function plug-ins and a group of scene plug-ins added to the Viz installation.

DataPool implementation allows Viz to keep information in named cells called DataFields. The data is managed at a separated, virtual, work space, linked to the graphics thru a set of plug-ins that make the connection between the DataFields and the graphic objects.



Some of the DataPool plug-ins have no effect over the graphics and are used only for managing and manipulating data. Some DataPool plug-ins are used for assigning data to DataFields and some plug-ins control Viz behavior upon changes in DataField values, affecting the graphic objects.

The native communication protocol to Viz is command oriented. Any change of the graphic objects, requires a separate command that is sent to the renderer. Each command includes the path to the container and the operation performed on the container.

4.3.1 Example

In a scene with a cube object, the cube can be scaled, moved, its bevel parameter changed, etc.

This is done by sending a command to Viz that specifies the name of the cube, its location in the Viz scene tree, the parameter to be changed and the new value for the parameter. If we add a font object to the same scene, we would be able to modify different parameters of the object, like setting its font, contents, kerning and so on.

If we want to implement a histogram of one single bar formed by the cube and the text object, setting the bar to the value `10`, we need to send Viz the following commands:

- Set the font object to display the string `10` (maybe add a suffix or prefix like a dollar sign: `$10`).
- Calculate the size of the cube and send the correct scaling value command Viz.

To change the value of the bar, two commands were sent, containing a different format for displaying the same graphic information: setting the bar to the value `10`.

Copyright © 2026 Vizrt

Another issue is that each external command sent to Viz contains the address of the container to which it is addressed. If a software program needs to send the two commands to Viz, it has to know the names of the containers and their location in the scene hierarchy. If an external application is written controlling a scene, the scene hierarchy and container names cannot be changed in the scene.

All these issues mentioned previously create difficult maintenance problems. DataPool is designed to solve these problems. It extends the external command mechanism of Viz, allowing the user, or the controlling software, to send data rather than container specific operations. The data is addressed to named fields (DataFields). The scene designer uses special purpose plug-ins, used for processing and handling the incoming data.

If we modify the previous example to use the DataPool mechanism, a DataText plug-in is added to the text object and a DataScale plug-in is added to the cube object. Both plug-ins are registered to receive data from the same DataField named *Bar Value*. As soon as *Bar Value* changes both plug-ins react, the first changing the string of the text object and the second scaling the cube. The data is sent to the scene through a scene plug-in called DataPool. This plug-in is just an interface to enter the data. Once this plug-in is defined in the scene, any external software can address it without the need of knowing the scene hierarchy or the container names.

Copyright © 2026 Vizrt

4.4 DataPool Plug-ins Overview

The DataPool plug-ins in Viz are responsible for handling incoming data and for affecting the objects in the scene according to the DataFields values.

4.4.1 Overview

The DataPool plug-ins are found in Viz in the Function plug-ins pool, under a folder called **Data**. There are DataPool DP Container Plug-ins and DataPool DP Scene Plug-ins. The scene plug-ins manage incoming data from different sources, inserting it to DataFields and maintain the DataFields Table. For example, the DataPool plug-in receives information from external sources and inserts it to the different DataFields. The DataMouseSensor senses mouse position and events and stores them in predefined DataFields.

The container DataPool plug-ins trigger graphic changes in the container they are attached to, when the DataField registered to them changes. Most of the container plug-ins have an argument called **Field Name**. The field name value is the name of the DataField to which the plug-in is registered to. The Field Name can take the following forms:

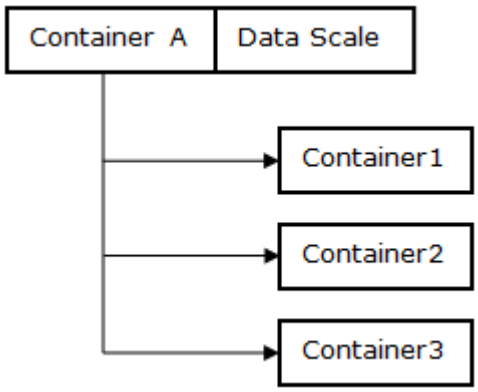
`<name>`

`<name>[number of items]`

If the field name is **VALUE** then the DataField is defined as a single datum field. However, if the field name is `VALUE[100]` then the name of the DataField is **VALUE** and it has 100 data items (values). The field name, scene hierarchy and the way the information is sent to the DataPool plug-in determine the container plug-in behavior.

4.4.2 Example

This is a hierarchy of a graph scene:



The above figure shows a container A with three children 1, 2, and 3. A DataScale plug-in is attached to A. The DataScale plug-in controls the scaling of the container according to the data it receives.

If the *Field Name* in the DataScale plug-in is defined to be `VALUES[3]` (a vector of 3 items named VALUES), DataScale acts in two different ways:

Copyright © 2026 Vizrt

1. If the data is entered without specifying an item index, DataScale affects container A.

> **Example:** If the data is entered as VALUES=0.5 0.5 0.5; then the container A is scaled by `0.5` in each axis.

2. If the data specifies an item index, DataScale works on the respective child container of A.

> **Example:** If the data is entered as `VALUES[2] = 0.1 0.1 0.1;` , then container3 is scaled by `0.1` in each axis (Note that the index value is zero based).

Copyright © 2026 Vizrt

4.5 Installation

DataPool is installed into the plugin directory under the last Viz installation. The plug-ins are installed according to the version found on the machine. The plug-ins are divided into four DataPool Plug-in Groups.

This section contains information on the following topics:

- DataPool Plug-in Groups
- Installing DataPool
- Advanced Silent Installation

Copyright © 2026 Vizrt

4.5.1 DataPool Plug-in Groups

The plug-ins are divided into four groups:

- DataPool
- Interactive
- Project Specific
- Script

For most users, installing the DataPool group is sufficient. For advanced applications install other plug-in groups as well.

The installation includes the *DataPoolLib.dll* and an empty *config.dp* file which are installed in the Viz directory. When installing the DataScript plug-in, *js32.dll* is also installed to the Viz directory.

DataPool

Main plug-in group (default). Includes most of the plug-ins:

- Data3DObject
- DataAction
- DataActionTable
- DataAlpha
- DataAnim
- DataArray
- DataArrow
- DataClock
- DataCondition
- DataCountDown
- DataCounter
- DataDevice (not installed for Viz 3.X)
- DataDirector
- DataFeedback
- DataGeom
- DataGraph
- DataGraphPoint
- DataImage
- DataKey
- DataKeyFrame
- DataKeyFrame2
- DataKeyTime
- DataLUImage
- DataMaterial
- DataMaterialGradiant
- DataMaterialIndex
- DataMaterialTable
- DataMath
- DataMinMax

Copyright © 2026 Vizrt

- DataMultiParam
- DataNumber
- DataObject
- DataObjectTracker
- DataParameter
- DataParamTracker
- DataPool
- DataPosition
- DataReader
- DataRotation
- DataScale
- DataSelector
- DataSHM (Viz Engine 3.X only)
- DataSHMTracker (Viz Engine 3.X only)
- DataStructure
- DataSwitch
- DataSystem
- DataTable
- DataText
- DataTextKerning
- DataTexture
- DataTime
- DataTimer
- DataViz3Script (Viz Engine 3.X only)

Interactive

All plug-ins that are based on mouse/keyboard events:

- DataClick
- DataFeedBack
- DataHyperLink
- DataInteractive
- DataKeyText
- DataManipulate
- DataMouseAction
- DataMousePosition
- DataMouseSensor

Project Specific

Plug-ins written for specific projects but can be used in other applications.

- DataCamera
- DataDispatcher
- DataDrawMask

Copyright © 2026 Vizrt

- DataScreen

Script

Contains the DataScript plug-in and the script library.

- DataScript (Viz Engine 2.X only)

Copyright © 2026 Vizrt

4.5.2 Installing DataPool

First-time Installation

1. Run the installation file and follow the instructions. When installing DataPool for the first time three installation types are available:
    - **Typical:** Installs only the DataPool plug-in group.
    - **Custom:** Installs the plug-in groups that the user selects.
    - **Full:** Installs all the DataPool plug-ins (all the groups).
2. Select the installation type and continue. An information window displays the path of the installation and selected plug-in groups.

> **Note:** The plug-ins are installed to the plug-in directory under the latest Viz installation.

> **Note:** This User Guide is installed with the DataPool group.

To Upgrade an Existing Installation

1. When upgrading the installation three options are displayed:
    - **Modify:** Changes the installed groups on the machine.
    - **Repair:** Reinstalls the installed groups (**.dp* files are not overwritten).
    - **Remove:** Uninstalls DataPool.
2. Select one of the options and click the next button.

Copyright © 2026 Vizrt

4.5.3 Advanced Silent Installation

The silent installation enables the user to install Viz DataPool without using the installer GUI. This option is used when a large number of machines require the software to be installed. To use the silent installation mode:

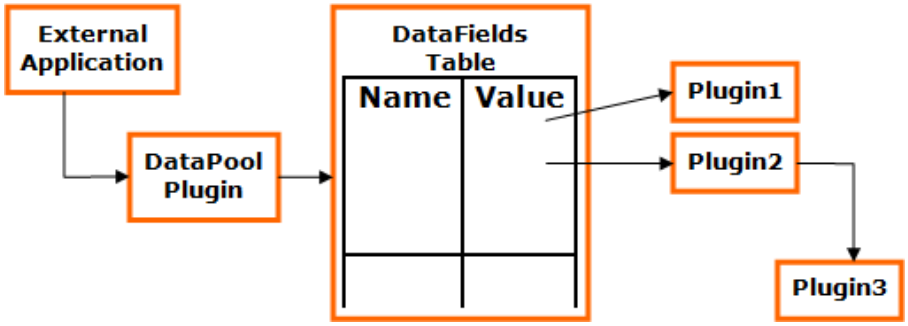
Examples

In a command shell, run the following command:

- To install DataPool from the *.msi* located in path *C:\temp\VizPlugins-DataPool-4.2.0.msi* without GUI (for progress installation progress GUI `/qn` should be replaced with `/qb`) and create a log file for the installation steps in path *C:\temp\install.log* and deploy the DataPool plug-ins in the plugin folder under *C:\program files\vizrt\VizEngine*:
  ```
  msiexec /i "c:\temp\VizPlugins-DataPool-4.2.0.msi" /qn /log c:\temp\install.log VIZ_ENGINE_INSTALL_DIR="c:\program files\vizrt\VizEngine"
  ```
- To uninstall installed DataPool without GUI and create a log file for the uninstallation steps in *C:\temp\uninstall.log*:
  ```
  msiexec /x "c:\temp\VizPlugins-DataPool-4.2.0.90.msi" /qn /log c:\temp\uninstall.log
  ```

Copyright © 2026 Vizrt

4.6 Architecture



DataPool implements a table of DataFields. Each DataField has a unique name by which it is accessed, a value (data) and a list of data changes handlers. The handlers are implemented as plug-ins in Viz, where each plug-in affects specific parameters of a container in Viz. The plug-in instances are registered to DataFields when attached to a container in the scene. The plug-in modifies the container parameters when the value of the DataField changes. It doesn't do anything unless it is called.

The external software sends the data to the DataPool through the DataPool plug-in. In the sent message, the application specifies which data should be assigned to which DataField. The DataPool plug-in accesses the DataFields table using the name of the field to be accessed, gets the relevant field and assigns the data to it.

This assignment invokes a series of triggers to all the plug-ins (handlers) registered to the field.

4.6.1 DataFields

As stated before, the DataField is a cell formed by three elements:

- Name (or address).
- Data.
- A list of handlers. The name is a zero terminated ASCII string.

The data is a vector of strings. The DataField can host one string of data, or a series of strings. The user can set the size (the number of items) of the DataField. Each one of the DataField items can be accessed separately or they can be accessed as a group, altogether.

The process of creation of the DataField is as follows: When a DataPool plug-in is specified to register to a certain DataField the plug-in looks for the specific DataField in the DataFields table. If the field doesn't exist the plug-in creates it and registers itself to the DataField. In order to send data to a certain DataField, at least one plug-in has to be registered to that DataField.

This section describes the architecture of DataPool and contains information on the following topics:

- DataPool Hierarchy
- Data Objects
- Arrays of Data Objects
- Tables of Data Objects
- DataPool Variables Scope
- DataPool Configuration Files

Copyright © 2026 Vizrt

- External Interface

4.6.2 DataPool Hierarchy

DataNodes and DataFields

DataPool is a hierarchical data structure composed of DataFields and nodes. DataFields are the lowest level of the hierarchy. The DataFields don't have children. Nodes are groups of DataFields. The DataPool hierarchy has a root node. Each of the DataFields in the hierarchy has a parent node, either the root node or any other sub-level node.

Example 1

The following example shows a hierarchy with a root, two nodes (NA and NB) and six DataFields. As can be seen each of the DataFields reside below a node. Setting the value of the DataField FA under the node NA follows the hierarchical structure to the required DataField: `NA/FA=<some information>;`

In the case of field FC under NB: `NB/FC=<some other information>;`



To set the data of all DataFields under the node NB, the following commands are sent:

Copyright © 2026 Vizrt

```
NB/FA=<info for A>;
NB/FB=<info for B>;
NB/FC=<info for C>;
```

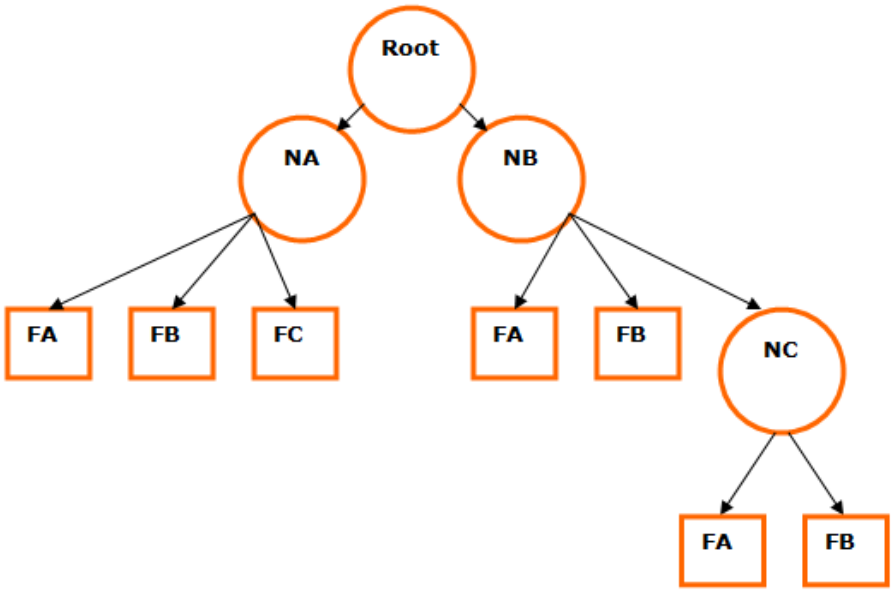
Note that there is a certain amount of redundancy because NB appears three times. To avoid using the node name every time, the following alternative format is used:

```
NB={
    FA=<info for A>;
    FB=<info for B>;
    FC=<info for C>;
};
```

The brackets determine the scope of the contained statements. In the above command the statement "FA=<info for A>;" is in the scope of object NB.

Example 2

Consider the following example:



In this example, NB has a child node called NC. To set data to the field FB under NC the following command is sent:
`NB/NC/FB = <info for B>;`

To populate all the fields in NC:

```
NB/NC= {
    FA = <info for A>;
    FB = <info for B>;
```

Copyright © 2026 Vizrt

```
};
```

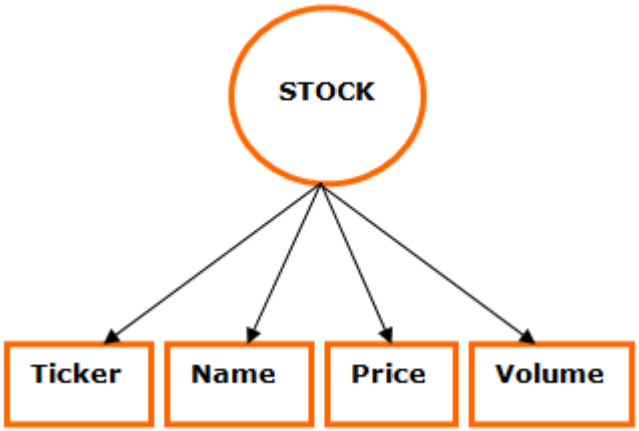
If we want to populate the whole hierarchy with data in one command:

```
NA = {
    FA =<info for A>;
    FB=<info for B>;
    FC=<info for C>;
};
NB = {
    FA =<info for A>;
    FB=<info for B>;
    NC = {
        FA=<info for A>;
        FB=<info for B>;
    };
};
```

4.6.3 Data Objects

Data objects use the structure of DataNodes and DataFields to describe properties of objects used for displaying information in Viz scenes.

An example of such an object is a stock. A stock object can be represented in the following way:



The DataPool representation of this stock object would be:

```
Stock={
    string Ticker;
    string Name;
    string Price;
    string Volume;
};
```

The meaning of this definition is: A Stock object is defined by four fields: Ticker, Name, Price and Volume.

The type string preceding each property of the stock means that incoming data is a text string and should be handled as such by the DataPool plug-ins.

To send information to an object called MyStock, defined as a Stock object, the following format is used:

```
MyStock={
    Ticker=IBM;
    Name=International Business Machines;
    Price=85.26;
    Volume= 5,188,500;
};
```

This information is sent to the DataPool plug-in and used by the different DataPool plug-ins in the scene. Other examples for objects are cities weather conditions, election results, etc.

Copyright © 2026 Vizrt

DataPool object definitions are saved in configuration files with the special suffix *.dp*, saved under Viz folder. DataPool loads all the objects defined in all of the *.dp* files, and makes the object types defined available to DataPool plug-ins such as DataObject or DataArray.

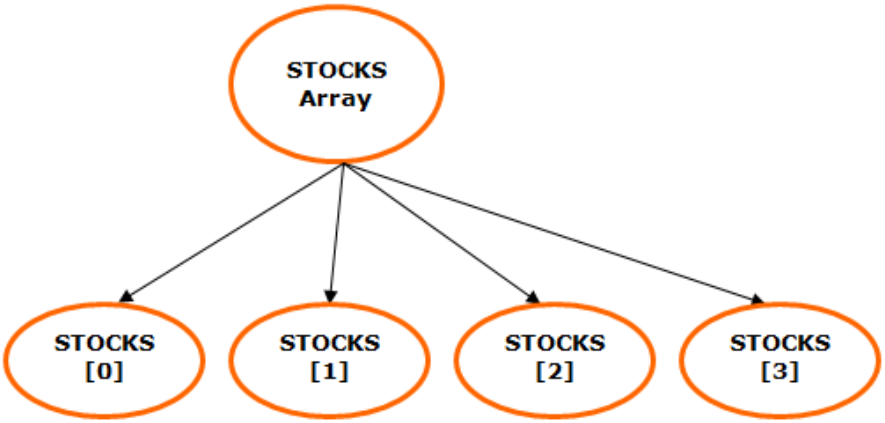
The default configuration file is called *config.dp* and it is installed empty during DataPool installation. When upgrading DataPool, the *config.dp* file is never overwritten. Multiple configuration files can be used to define various objects. All files must be saved with the *.dp* suffix.

Copyright © 2026 Vizrt

4.6.4 Arrays of Data Objects

When dealing with multiple instances of the object type, an array of object types can be defined. Defining an array of objects simplifies data management when sending the information to the different objects.

Sometimes there is a need to maintain lists of objects. An example could be the display of a histogram of stocks. Each Viz object is attributed a DataPool object. But in this case, it would be very unwise to give a name to each of the objects because the number of the objects may be unknown. DataPool solves this problem by providing with a means of array of objects. Like in the case of the single object, the array has an object type. When defined the array builds a set of objects as its children. This scheme is shown in the following figure:



In the figure, an array of stocks of size four is shown. As can be seen each of its children are named Stocks[0], Stocks[1], and so on. To enter data to the array element number 2:

```
Stocks[2]/Price=85.3;
```

If we want to enter information to the whole array:

```
Stocks[0-3]={
    {Ticker=IBM;Price=85.26;},
    {Ticker=SGI;Price=1.09;},
    {Ticker=AOL;Price=14.71;},
    {Ticker=AA;Price=23.85;},
};
```

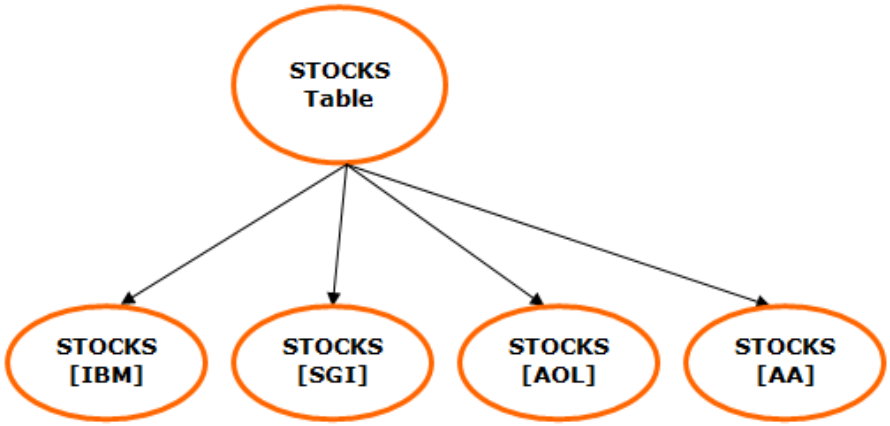
Note: There are two sets of brackets in the above statement: One for the array itself and a second for the objects themselves. Also note that because this is an array, the objects are separated by commas rather than by semicolons.

Copyright © 2026 Vizrt

4.6.5 Tables of Data Objects

In the DataArray plug-in, we can organize the objects in an array structure and refer to the whole array or to single objects within the array using the array name and an index.

Sometimes it is useful to organize the objects and keep the object's names. For example, rather than keeping a list of stocks we could keep a table in which IBM's stock is stored under the name IBM, SGI's stock under the name SGI and so on. As in the array, the table has an object type, but unlike the array the table doesn't have a size. Every time an object is inserted in the table the table grows. The DataTable plug-in stores the data sent to it and it is able to recall this data when requested. That data can later be used with the copy functionality of DataPool, to effect the graphic objects thru the DataFields.



A typical table of stocks could look as follows:

In the above figure a table of stocks named "Stocks" can be seen.

The children objects are called "Stocks[IBM]", "Stocks[SGI]", etc…

To send information to the stock IBM: `Stocks[IBM]/Price = 85.26;` or

```
Stocks[IBM]={
    {
        Volume=5,188,500;
        Price=85.26
    ;}
};
```

To enter information to a number of stocks:

```
Stocks[IBM, AOL]={
    {
        Ticker=IBM;
        Price=85.26;
    },
    {
```

```
        Ticker=AOL;
        Price=14.71;
    }
};
```

4.6.6 DataPool Variables Scope

When referring to DataPool variables, the common and wide spread meaning is addressing global variables by their name and addressing structured variables by specifying the full path to the variable (i.e. *<structure name>/../<variable name>*).

In some DataPool plug-ins (where applicable), one of the parameters defines the DataPool variable scope. The options are Local and Global:

- **Local Variables:** When referring to Local DataPool Variables, the meaning is variables within a DataPool object/data structure. When the scope is set to Local, the DataPool variable is addressed by its name only, even when it is a part of a DataPool structure.
- **Global Variables:** When referring to Global DataPool Variables, the meaning is variables that are not defined as part of a DataPool structure.

⚠ **Note:** This parameter only has an effect on DataPool expressions and variables that reside under the DataObject container, i.e. child containers within the object's hierarchy.

Example

The following definitions are set in the *config.dp* file:

```
string MyText;
MyObject{
    string MyText;
    string TextLength;
};
```

When addressing the variable MyText:

- MyText="This Text"; → the Global variable MyText is set to *This Text*.
- MyObject/MyText="This Text"; → The variable MyText in the MyObject structure is set to *This Text*.
  When the scope parameter is set to Local and setting the following assignment in one of the structure's children MyText="This Text"; then the effect is like defining the full path: MyObject/MyText="This Text";.

Copyright © 2026 Vizrt

4.6.7 DataPool Configuration Files

This page contains information on the following topics:

- DP Configuration Files
    - Example Configuration File
- DataPool.ini File
- Conversion Table
    - File Syntax

Warning: If using Viz Engine 4 or newer: When editing DataPool configuration files, always modify the files located in *%ProgramData%/vizrt/VizEngine*! The files in *\Program Files* are just templates which are copied to *%ProgramData%* at installation time. The Engine reads the files from *%ProgramData%*.

DP Configuration Files

DataPool data fields and objects are used across scenes and engine machines to support Viz scenes and data distribution. The DataPool configuration files, used to define DataPool structures and DataPool variables, use the suffix *.dp*. All files with *.dp* suffix are loaded during Viz load and the data fields and structures are defined in the DataPool memory. The configuration files can be copied to all relevant Viz machines or saved in a common folder and defined in the *DataPool.ini* file.

When installing DataPool, the file *datapool.dp* is installed to the Viz Program Files folder and copied to *%ProgramData%/vizrt/VizEngine*. This file is an example and the users can modify the file to define data fields. Additional *.dp* files can be used to organize the DataPool variables and data structures.

Example Configuration File

```
Example_STOCK = {
    string Example_ID;
    string Example_Name;
    string Example_symbol;
    string Example_Open;
    string Example_Close;
    string Example_High;
    string Example_Low;
    string Example_Volume;
    string Example_Price;
    string Example_TimeRange;
};
Example_RESULTS = {
    string Example_NAME;
    string Example_SCORE;
    string Example_COLOR;
};
```

The file shows two DataPool structures.

Copyright © 2026 Vizrt

Note: When using multiple configuration files, make sure your data structures are unique. Multiple structures with the same name are not supported.

DataPool.ini File

The *DataPool.ini* file is installed to the Viz Program Files folder and copied to *%ProgramData%/vizrt/VizEngine*. It contains the following parameters:

- **ConfigurationFolder:** Defines the location of the *.dp* files to be loaded. If omitted, the default path is set to .\ (Viz folder).
- **MultiDatapool:** Defines if a separate Viz DataPool segment is assigned to each scene loaded to Viz.
- **ReloadConfigFiles:** Defines whether to reload the configuration files (*.dp* files) while loading each scene.

Conversion Table

The conversion tables file is a comma separated file installed in the Viz Program Files folder and copied to *%ProgramData%/vizrt/VizEngine*. All the conversion tables are defined in this file name, using a fixed format. The tables are used to convert input data values to another value defined in the table. The first column in the table is the input value and the second column is the converted value (output).

Note: The file name is hard coded: *DP_ConvTable.csv*.

File Syntax

The file contains a few reserved tokens used for defining conversion tables and other parameters:

- **__VERSION__:** This entry is used for future file support and version compatibility issues.
- **__DELIMITER__:** This entry is used to define a different file delimiter (other than comma). If omitted comma is used.
- **__TABLE__:** This entry defines the beginning of a conversion table. The table end is defined by another table entry or end of file.
  Table example:

```
__TABLE__,colors,
red,255 0 0
green,0 255 0
blue,0 0 255
yellow,255 255 0
cyan,0 255 255
purple,255 0 255
__DEFAULT__, 0 0 0
,
```

  In the example, a table called colors is defined. When the conversion table is used in a plug-in, an input string *red* is converted to the output string `255 0 0` and sent back to the plug-in for processing.
- **__DEFAULT__:** This token is used in any of the conversion tables as a default conversion value (any value not found in the table is converted to the default value).

Copyright © 2026 Vizrt

4.6.8 External Interface

The external interface to DataPool is based on Viz messages sent to the DataPool plug-in. Each message contains a series of data assignments to different DataFields.

The structure of the message is:
`command1 command2 command3 ...`

The structure of each command is as follows:
`<Name>[items range] = datum1, datum2, datum3, ...;`

Where `name` is the name of the DataField to assign data to and `items range` is a description of items to which the data is addressed.

> **Note:** The semicolon is part of the command and should be added at all times. It is used to define the end of the command.

The item ranges are a comma-separated list of indexes, or a from-to index.

Examples

1. Send data to a field named **VALUE**: `VALUE=30;`
2. Send data to the first five values of a field called **VALUES**: `VALUES[0-4] = 0, 1, 2, 3, 4;`
   The above is equivalent to sending: `VALUES[0]=0; VALUES[1]=1; VALUES[2]=2; VALUES[3]=3; VALUES[4]=4;`

> **Note:** The indexes of the items are zero-based (start from 0).

3. Send data to the items 1-3, 55-57 and 101: `VALUES[1-3,55-57, 101] = 0, 1, 2, 3, 4, 5, 6;`
   This is equivalent to sending: `VALUES[1]=0; VALUES[2]=1; VALUES[3]=2; VALUES[55]=3; VALUES[56]=4; VALUES[57]=5; VALUES[101]=6;`

Copyright © 2026 Vizrt

4.7 Expressions

An expression is a text parameter combining constant strings and references to DataPool variable's (DataField's) value. The reference syntax is `$(DataField)`. When using the expression, the string `$(DataField)` is replaced by the value of the DataField specified in the parentheses. For example, suppose we need to run different animation directors in Viz and we want to send the name of the director as a parameter. The command in Viz to run a director is:

```
0 RENDERER*STAGE*DIRECTOR*<name> START
```

So we need to send the entire command for each director that we want to run. By using expressions, only the name of the director to be run is sent to Viz. We define a DataField called DIR that contains the name of the director to run.

> ⚠ **Note:** Predefined DataFields are defined in the config.dp file, residing in the Viz folder.

Now we can use the following syntax for running the animations:

```
0 RENDERER*STAGE*DIRECTOR*$(DIR) START
```

$(DIR) is replaced by the incoming data, so if the data was *DIR=mydirector*; the whole expression is interpreted to be:

```
0 RENDERER*STAGE*DIRECTOR*mydirector START
```

> ⚠ **Note:** The string enclosed between the parentheses is considered to be either a DataPool variable, a special DataPool command or a special DataPool variable.

A common DataPool variable is **$(INDEX)**. This variable is used to access the full index range of a DataField. For example, if we have two DataFields called DST[10] and SRC[10] and we want to copy all the values from vector SRC to vector DST, the expression should state `DST[$(INDEX)]=SRC[$(INDEX)]`. A useful DataPool special command is **$(REFRESH)**. The effect of having an expression like `$(REFRESH) $(VALUE)` is to trigger all the plug-ins registered to the DataField called VALUE without having to receive external data. Each data plug-in has its own set of special purpose variables. They are explained in the explanation of each plug-in.

Copyright © 2026 Vizrt

4.8 Common DataPool Parameters

- **Field Name:** Defines the name of the DataField from which the plug-in receives the data.
- **Data Min and Data Max:** Specifies value mapping between incoming data and a Viz value. These parameters are always used in conjunction with two parameters representing Viz values (usually named Value Min and Value Max). The Data Min value is converted to a minimum Viz value and the Max Data is converted to a maximum Viz value. All incoming values are converted respectively to Viz values. When the Clamp parameter is set `On`, Values out of this range (smaller than Data Min or larger than Data Max) is cropped to the Min/Max values. When Clamp is `Off`, incoming values are divided by the range between Data Min and Data Max and the resulting value is used as the incoming value and interpolated to Viz values.

  ```
  ## Example
  Data Min is 0
  Data Max is 100
  Viz Min is 0
  Viz Max is 200
  Sent value is 125
  ```

  If Clamp is On the Viz value is set to `200` (all values above Data Max (100) are cropped to the Data Max Value and interpolated to Viz Values)
  If Clamp is Off, Viz value is set to `250`.
- **Prefix:** Specifies a prefix string that is added in front of the incoming data string.
- **Suffix:** Specifies a suffix string that is added at the end of the incoming data string.
- **Clamp:** Clamps the incoming data values to the data min/max range when set to `On`. All DataPool plug-ins that use the Data Min and Data Max parameters have a clamp button. Values out of this range (smaller than Data Min or larger than Data Max) are cropped to the Min/Max values when the Clamp parameter is set `On`.
- **Notify Only On Change:** Triggers all DataPool plug-ins registered to that DataField when a DataField value is changed. This happens also if the DataField was set to the same values. Notifies the data plug-ins only if the field value was actually changed when enabled. The default behavior is `Off` (as it was in previous DP releases, before adding this feature).
- **Use Other Container:** Allows one container to affect another container, or allows another instance of the same plug-in to be used on the same container. DataPool container plug-ins affect the containers they are attached to, and cannot affect other containers directly. A common example of using this parameter is changing multiple keyframes in an animation. Since only one plug-in can be attached to the container, dummy containers with DataKeyFrame plug-in, using this parameter can be used to affect the remaining keyframes (This parameter makes the DataKeyFrame2 plug-in unnecessary. The plug-in is installed for backwards compatibility only). When enabled, additional options are also enabled:
  - **PARENT:** The DataPool plug-in affects the parent container (the container residing above the current container in the scene hierarchy).
  - **GrParent:** The DataPool plug-in affects the grandparent container (the container residing two levels above the current container in the scene hierarchy).
  - **GrGrParent:** The DataPool plug-in affects the great grandparent container (the container residing three levels above the current container in the scene hierarchy).

Copyright © 2026 Vizrt

- **REMOTE:** When remote is selected another parameter, Remote Container, is enabled. Drag the container to be effected to the container place holder.

> ⚠ **Note:** When using the Parent, GrParent or GrGrParent options, the link to the other container is relative. When using the Remote option the link to the other container is absolute. Using the relative options is recommended when using object designs that are duplicated in the scene and controlling containers within the object hierarchy (like in Viz Curious Maps Client labels or Viz Weather objects).

> ⚠ **Note:** The REMOTE option maintains compatibility with previous versions of DataPool using the Use Remote Container option.

- **Incremental Change:** Specifies if the incoming data is added to the current value of the data field (`On`) or replaces the current value of the data field (`Off`).
- **Use Conversion Table:** Enables a Conversion Table Name parameter when enabled. Enter the table name as it appears in the DP_ConvTable.csv file, without the proceedings leading and proceeding underscore characters.

Copyright © 2026 Vizrt

4.9 Special DataPool Variables

- **$(INDEX):** Returns the indexes of sub-containers of the selected container.
- **$(REFRESH) $(DataField_Name):** Triggers all plug-ins registered to DataField_Name as if the DataField DataField_Name was changed.
- **$(SCENE), $(THIS_SCENE):** Returns the name of the current scene.
- **$(CONTAINER):** Returns the container ID of the container that the DataPool plug-in is attached to.
- **$(CONTNAME):** Returns the name of the container hosting the DataPool plug-in.
- **$(CHILD):** Returns the index of the selected child.
- **$(DATA):** Returns the string value of a text object.
- **$(PARENT):** Represents the parent DataPool object/structure of the current DataPool field. This variable enables the user to change the values of its "brother" data fields. This special variable is applicable only in plug-ins that have an Action parameter.
  Example: In the given structure

```
DEMO={
    string Field1
    string Field2
};
```

  If a DataPool plug-in, with an Action parameter, on the Field1 container refers to $(PARENT)/Field2, it sets the value of Field2 in the same structure:
  `$(PARENT)/Field2="This is the Field1" $(Field1)`
  The result of this action is that *Field2* contains the quoted text and the content of *Field1* data field.

IMPORTANT! When working with DataPool structures/objects, the $(PARENT) is the recommended way of addressing other data fields in the DataPool structure/object.

Note: Using the $(PARENT) variable is similar to using the Global/Local parameter in some of the DataPool plug-ins. Both make the distinction between a DataPool variable that is part of a data structure/object and a DataPool variable that is used not a part of a structure/object.

Note: The Field Name parameter (the DataPool variable that triggers the action should be a part of the structure. The $(PARENT) value is defined according to this data field.

- **$(PARENT_NAME):** This special variable is applicable only in plug-ins that have an Action parameter. The $(PARENT_NAME) returns the parent DataPool object/structure name, of the current DataPool field.

Note: The Field Name parameter (the DataPool variable that triggers the action should be a part of the structure. The $(PARENT_NAME) value is defined according to this data field.

See $(PARENT) description for additional details.

- **$(PARENT_FULL_NAME):** Returns the full hierarchy string of the parent DataPool object/structure, of the current DataPool field. If the structure/object has only one level of variables in the hierarchy, then the $

Copyright © 2026 Vizrt

(PARENT_FULL_NAME) and the $(PARENT_NAME) returns the same value. If the structure/object has more than one level, the $(PARENT_FULL_NAME) returns the entire path of the structure from the requested level. The full name is returned in the format of: xx/yy/zz/. This special variable is applicable only in plug-ins that have an Action parameter.

Note: The Field Name parameter (the DataPool variable that triggers the action should be a part of the structure. The $(PARENT_FULL_NAME) value is defined according to this data field.

See $(PARENT) description for additional details.

- **$(FIELD_NAME):** Returns the name of the DataField when used in a child container.
- **$(FIELD_FULL_NAME):** Returns the full name of the DataField when used in a child container.
- **$(FIELD_DATA):** Returns the value of a DataField.
- **$(SCENE_FULL_NAME):** Returns the full path and scene name.

Copyright © 2026 Vizrt

4.10 DataPool Container Plug-ins

The container plug-ins are listed in this section in alphabetic order.

- Data3DObject
- DataAction
- DataActionTable
- DataAlpha
- DataAnim
- DataArray
- DataArrow
- DataCamera
- DataCenter
- DataClick
- DataCondition
- DataCountdown
- DataCounter
- DataDirector
- DataDispatcher
- DataDrawMask
- DataFeedback
- DataGeom
- DataGPI
- DataGraph
- DataGraphPoint
- DataHyperlink
- DataImage
- DataInRange
- DataInterpolate
- DataKey
- DataKeyFrame
- DataKeyFrame2
- DataKeyText
- DataKeyTime
- DataLookup
- DataLUImage
- DataManipulate
- DataMaterial
- DataMaterialGradient
- DataMaterialIndex
- DataMath
- DataMathObject
- DataMinMax
- DataMouseAction
- DataMousePosition

Copyright © 2026 Vizrt

- DataMultiParam
- DataNumber
- DataObject
- DataObjectTracker
- DataParameter
- DataParamTracker
- DataPosition
- DataReader
- DataRotation
- DataScale
- DataScreen
- DataSelector
- DataSerialClock
- DataSerialGPS
- DataSHM
- DataSHMTracker
- DataStructure
- DataSwitch
- DataSystem
- DataTable
- DataTemo
- DataText
- DataTextKerning
- DataTexture
- DataTime
- DataTimer
- DataViz3Script
- DataWPosition

Copyright © 2026 Vizrt

4.10.1 Data3DObject



The Data3DObject plug-in receives a name of an object from the Viz object library, and it loads it to the container it is placed on.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

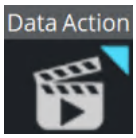
Note: Built in objects cannot be used.

Prefix is used to add a Viz object path to a folder where the objects are stored.

For example: If the FieldName is *AAA* and the object is *spaceship* then the command is: `AAA=spaceship;`

Copyright © 2026 Vizrt

4.10.2 DataAction



The DataAction plug-in executes an action (Viz commands or DataPool commands) when the DataField registered to it changes.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

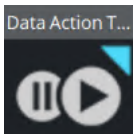
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataAction Properties

- **Action:** Specifies the action to be invoked. If the action string begins with `0`, then the command is handled as a Viz command, otherwise it is considered to be a DataPool command. The action can contain more than one command to be called. The commands must be separated by semicolons (**;**). Expressions can be used in the action utilizing the special DataPool variables.
- **Set to Mute:** Specifies if the action is triggered when the data field changes.

Copyright © 2026 Vizrt

4.10.3 DataActionTable



The DataActionTable plug-in allows defining a table of actions (as described in DataAction). Each of these actions is identified by a value of the *FieldName* variable.

When DataActionTable receives data (a value of the FieldName variable) it compares the data to each of the values. If the data matches one of the values, DataActionTable invokes the corresponding action.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

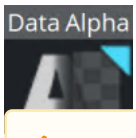
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataActionTable Properties

- **Numeric Values Only:** Enforces the plug-in to use only numeric values received as the Field Name value, and uses only numeric values in the `Value<i>` parameters, when set to `On`.
- **Value<i>:** Specifies the value of the FieldName variable that invokes the Action specified in `Action<i>`.
- **Action<i>:** Specifies the action to call in the case the data matches `Value<i>`.
- **Default Action:** Specifies the action to invoke if the data doesn't match any of the specified values.

Copyright © 2026 Vizrt

4.10.4 DataAlpha



The DataAlpha plug-in controls the value of the Alpha function plug-in.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

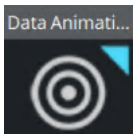
Information: DataPool plug-ins only work properly on a single channel on the same machine.

Values are between `0` and `100` (`0` =transparent, `100` =opaque).

Note: An Alpha plug-in must be assigned on the controlled container.

Copyright © 2026 Vizrt

4.10.5 DataAnim



The DataAnim plug-in animates a value and sends the data of this value, every field, to the DataField that is defined in the Output Field Name.

DataAnim is different from other DataPool plug-ins since it is not triggered by a change of a DataField. It is activated from the animation director it is located in, and plays a predefined animation.

DataAnim acts as a data generator for the specified output DataField.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataAnim Properties

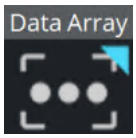
- **Output Field Name:** Defines the name of the DataField that receives the animated value of the *Value* parameter every video field.
- **Animation Type:** Defines the type of value that is sent to the Output Field Name. Value types are integer or float.
- **Value:** Generates and sends the data (the animated parameter of the plug-in) to the DataField defined in Output Field Name.

Example

Define the Output Filed Name as `AAA`, and create an animation of the Value parameter from `0` to `10`. When running the animation, `AAA` is changed every field as the animation progresses from `0` to `10`. All DataPool plug-ins registered to the DataField `AAA` is then triggered every field and set to the current value of the Value parameter.

Copyright © 2026 Vizrt

4.10.6 DataArray



The DataArray plug-in enables a simpler way of controlling a complex structure of objects that are defined in the *config.dp* file. See DataObject for detailed information about defining object structures and using the *config.dp* file.

DataArray defines the size of the array (the number of its child objects) and manages the data flowing to the objects.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataArray Properties

- **Field Name:** Defines the name and size of the array.
- **Type:** Defines the type of the array objects. The list of types is created from the types defined in the configuration files (*.dp* files).

Example

The Field Name `Stocks[32]` defines an array named Stocks that has 32 objects of the type Stock selected in the Type parameter.

Copyright © 2026 Vizrt

4.10.7 DataArrow



The DataArrow plug-in works specifically on the Arrow geometry plug-in in Viz.

It controls the start and end points of the arrow object, enabling incoming data to control the arrow object.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

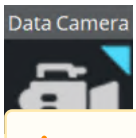
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataArrow Properties

- **Format:** Specifies the format of the expected data. The plug-in processes the data according to the selected format. The options are *Point1* (start point), *Point2* (end point) or *Point1&2* (both end points).

Copyright © 2026 Vizrt

4.10.8 DataCamera



The DataCamera plug-in controls camera position and target according to the incoming data.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

If the container that the plug-in controls has child containers, then the specified camera locks its position to the first child container and the destination of the camera is locked to the second child container. In this case, the data should be of the following format:

```
DATA=<camera> <x position> <y position> <z position> <x target> <y target> <z target>;
```

Note: Position and target values are used as offsets from the child containers location.

If the container doesn't have any child containers then the data should be of the following format:

```
DATA=<camera> <x position> <y position> <z position> <pan> <tilt> <twist>;
```

Note: Camera position change is relative to the container that the plug-in works on.

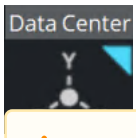
The plug-in works in two different modes: First, the data contains both the camera ID, the position and target of the camera and second, the data contains just the camera ID and the position and the target of the camera are calculated from the center of the first two children of the container. In the second mode, the container must have two children. The center of the first child determines the center of the camera and the center of the second child determines the target of the camera.

DataCamera Properties

- **Set Current:** Defines whether a camera switch in the render window occurs if the incoming data refers to a different camera than the current camera. If set to `On`, then when incoming data refers to a camera other than the current camera selected in the render window, the camera switches to the camera defined in the incoming data. When set to `Off`, the camera defined in the incoming data is moved, but the current camera remains the same.
- **Function:** This parameter is inactive. The parameter is kept for backwards compatibility and should not be used.

Copyright © 2026 Vizrt

4.10.9 DataCenter



The DataCenter plug-in changes the axis of the container according to the received data.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataCenter Properties

- **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- **Incremental Change:** Adds incoming data to the current value of the data when set to `On`. When set `Off`, the data replaces the existing value of the data field.

Example

If Field Name is `VALUE` and the format is `X`, the data sent is `VALUE=10.5`. The container moves the X axis to a position of `10.5`.

Copyright © 2026 Vizrt

4.10.10 DataClick



The DataClick plug-in triggers an action when the mouse is clicked within the drawn area of the container it is assigned to or on any of its child containers.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataClick Properties

- **Settings:** Defines the displayed editor type. When set to **Advanced**, every click triggers two events: mouse down (mouse button pressed) and mouse up (mouse button released). When set to **Simple**, only mouse click is used.
- **On Click:** Triggers a DataPool action when the mouse is clicked within the container's drawn area and any of its child containers.
- **On Click Child:** Triggers a DataPool action when the mouse is clicked within the drawn area of a child container under the container affected by the DataClick plug-in.
- **On Mouse Down:** Triggers a DataPool action when the mouse button is pressed within the container's drawn area and any of its child containers.
- **On Mouse Down Child:** Triggers a DataPool action when the mouse button is pressed within the drawn area of a child container under the container affected by the DataClick plug-in.
- **On Mouse Up:** Triggers a DataPool action when the mouse button is released within the container's drawn area and any of its child containers.
- **On Mouse Down Child:** Triggers a DataPool action when the mouse button is released within the drawn area of a child container under the container affected by the DataClick plug-in.
- **On Enter:** Triggers a DataPool action when the mouse enters the container's drawn area and any of its child containers.
- **On Enter Child:** Triggers a DataPool action when the mouse enters the drawn area of a child container under the container affected by the DataClick plug-in.
- **On Leave:** Triggers a DataPool action when the mouse leaves the container's drawn area.
- **On Leave Child:** Triggers a DataPool action when the mouse leaves the drawn area of a child container under the container affected by the DataClick plug-in.
- **Behavior:** Defines in what way the mouse events trigger the actions or cause the DataField to change. When set to **On Click**, only the on click events are enabled. When set to **On Click and Move** additional action (On Enter and On Leave) fields are enabled.

Copyright © 2026 Vizrt

4.10.11 DataCondition



The DataCondition plug-in evaluates the value of the DataField it is registered to using a defined comparison function and executes an action accordingly.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataCondition Properties

- **Format:** Defines the way incoming data is handled. Data can be used as a number, string, a set of X Y values or a set of X Y Z values. The following table shows how the different formats are evaluated. Cells filled with `EVAL` are evaluated by the selected operator. Cells filled with `TRUE` always return true. Cells filled with `FALSE` always return false. Other text in a cell shows the behavior of the chosen operator.
  - **Number:** Performs a simple numeric evaluation.
  - **Text:** Performs an alphanumeric evaluation.
  - **X Y:** Calculates the sum of the absolute value of the difference between the first and second values of the arguments and compares it to the precision value multiplied by two: |x1 - x2| + |y1 - y2| < 2 * precision.
  - **X Y Z:** Calculates the sum of the absolute value of the difference between the first, the second and the third values of the arguments and compares it to the precision value multiplied by three: |x1 - x2| + |y1 - y2| + |z1 - z2| < 3 * precision.

	<	>	<=	>=	=
Number	EVAL	EVAL	EVAL	EVAL	EVAL
Text	EVAL	EVAL	EVAL	EVAL	EVAL
X Y	FALSE	!=	=	TRUE	EVAL
X Y Z	FALSE	!=	=	TRUE	EVAL

Note: The X Y and X Y Z formats only support equal to (**=**) and not equal to (**!=**).

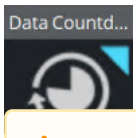
- **Condition:** Specifies the evaluation function to apply. Evaluation is done left to right (incoming data is placed at the left side of the comparison operator). `True` always runs the action and `False` always runs the *else* action (if enabled). The *If* option evaluates the DataField value: i.e. all values other than zero runs the action, zero runs the *else* action. `Between` checks if the value is greater or equal (>=) than the first and lower (<) than the second argument.

Copyright © 2026 Vizrt

- **Arguments:** Defines the expression that is compared to the incoming data. An argument can be a constant (i.e. a fixed number/text) or a data field value. To specify a data field value as an argument, use the following syntax: `$(DataFieldName)`.
- **Compare Vectors:** Defines incoming data as a vector (an array of DataFields with definition of the number of items) and the argument expression is taken to be a vector of data when set to `On`. The comparison is then done item by item and the action is executed accordingly on each of the children. If this parameter is `Off` then each item in the incoming data is compared against one single argument.
- **Precision:** Affects the evaluation of X Y and X Y Z formats only. It defines the margin in which the evaluation results as true (see X Y or X Y Z format explanation).
- **Do Once On Change:** Triggers an Else action. When set `ON`, only if the condition result was changed, i.e. if it was changed from true to false or vice versa, the action (or Else action) is triggered. When set `Off`, every time the condition is checked the action (or Else action) is triggered accordingly.
- **Action:** Specifies the action to invoke. If the action string begins with `0`, then the command is handled as a Viz command and sent directly to Viz Engine, otherwise it is considered to be a DataPool command. The action can contain more than one command to be called. The commands must be separated by semicolons (**;**). Expressions can be used in the action utilizing the special DataPool variables.
- **Else:** Enables the Else action when set to `On`. When an else action is defined it is executed if the evaluation result is false. When set `Off`, the Else action field is disabled and no action is executed when the evaluation result is false.
- **Else Action:** Defines the action to execute if the evaluation result is false.

Copyright © 2026 Vizrt

4.10.12 DataCountdown



The DataCountdown plug-in counts up or down to a defined target time.

The target time is an absolute time (future or past).

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataCountdown Properties

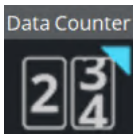
- **DP variables Prefix:** Variable prefix is added to a set of fixed names, based on time scale, for different time variables. The prefix is used to distinguish one set of time parameters from the other. The default prefix is CD1. For example, if Prefix is set to DDay, the corresponding variables are:

```
DDay_DAYS
DDay_HOURS
DDay_HOURS_TOTAL
DDay_MINS
DDay_MINS_TOTAL
DDay_SECS
DDay_SECS_TOTAL
DDay_SIGN (i.e + for future and: for past)
```

- **Target Day in month:** Defines the countdown target day. Values are `1-31`.
- **Target Month:** Defines the countdown target month. Values are `1-12`.
- **Target Year:** Defines the countdown target year. Values are `1970-2050`.
- **Target Hour:** Defines the countdown target hour. Values are `0-23`.
- **Target Minute:** Defines the countdown target year. Values are `0-59`.
- **Target Second:** Defines the countdown target year. Values are `0-59`.
- **Leading Zero All:** Adds a leading zero to all the single digit numbers (`0-9`) when set to `On`.
- **Leading Zero Minutes, Seconds:** Adds a leading zero to the single digit numbers (`0-9`) of the minutes values and the seconds values when set to `On`. Others are displayed as a single digit.
- **Action On Time:** Defines an action that is executed when the counter reaches the target time. If the action string begins with `0` then the command is handled as a Viz command, otherwise it is considered to be a DataPool command. The action can contain more than one command to be called. The commands must be separated by semicolons (**;**). Expressions can be used in the action utilizing the special DataPool variables.
- **Action On T-offset:** Defines the action that is executed when the counter reaches an offset point from the target time.
- **Offset (Seconds):** Defines the offset from the target time that triggers the action on *t*-offset (in seconds).
- **Set Target to NOW:** Sets the current system time as the target time when pressed.

Copyright © 2026 Vizrt

4.10.13 DataCounter



The DataCounter plug-in increases/decreases its value every time the DataField changes (if set to begin counting at `1` with an incremental value of `1`, the plug-in counts the number of times that the DataField changed).

DataCounter starts counting at the *From* parameter value, adding the *Step* parameter value to counter every time the DataField defined in Field Name changes, until the value *To* is reached.

When the *To* value is reached, the counter starts over again from the beginning.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

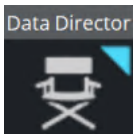
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataCounter Properties

- **From:** Defines the initial value of the counter.
- **To:** Defines the ending value of the counter.
- **Step:** Defines the value to increment.
- **Output Field:** Defines a DataPool variable that the counter value is assigned to.
- **Output Scope:** Defines whether the DataPool variable defined in the output field is a global variable or a local variable.

Copyright © 2026 Vizrt

4.10.14 DataDirector



The DataDirector plug-in enables the control of an animation director by assigning stage commands to the DataField it is registered to.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataDirector Properties

- **Use Other Container:** Defines if the plug-in controls the director of a different container (available options: parent, grand parent, grand grand parent, remote).
- **Use the container's director:** Defines if the plug-in controls the director for the animation in a container.

Note: If animation of a remote Container is used (by checking Use Other Container->Remote), then the animation director in the remote container is triggered.

If none is enabled, then the default director is controlled.

Stage Commands

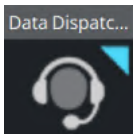
The stage commands that can be assigned to the DataField: `START` , `CONT` , `START_REV` (v5.0 only), `CONT_REV` (v5.0 only), `STOP` .

Example

Create an animation for a container (for example, a rectangle). Add the DataDirector plug-in to the container. Set the *Field Name* of the plug-in to `AAA` . Send the command: `AAA=START;` .

Copyright © 2026 Vizrt

4.10.15 DataDispatcher



The DataDispatcher plug-in enables data transfer between plug-ins.

DataDispatcher calls the dispatcher of a plug-in in the controlled container, and transfers data via DataPool variables.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataDispatcher Properties

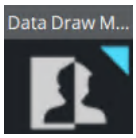
- **Parameter Type:** Defines destination plug-in type. Data is formatted accordingly.
- **Function Name:** Defines the name of the plug-in whose dispatcher is to be called.

Data Format

The data should arrive in the following format: `<dispatcher message ID> <dispatcher message body>`

Copyright © 2026 Vizrt

4.10.16 DataDrawMask



The DataDrawMask plug-in is used when using viewports (multiple camera windows in Viz render window).

DataDrawMask defines the mask type of the container it controls and the way it is rendered in each of the viewports.

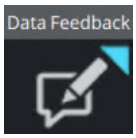
The DataField (defined in Field Name) value contains this information.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Copyright © 2026 Vizrt

4.10.17 DataFeedback



The DataFeedBack plug-in sends information from the DataPool to a specific UDP socket in a remote host.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataFeedback Properties

- **Destination Host:** Defines the hostname to which the data is sent to.
- **Destination Port:** Defines the socket number to which the data is sent to.
- **Connect:** Initializes the connection after the host and port are defined.
- **Connection Type:** Defines the connection type to open: UDP or TCP.
- **Message:** Defines the expression to send to the remote socket when the plug-in is triggered.
- **Add EOL (CR &LF) to TCP:** Defines if new line and line feed characters to add to the TCP message.
- **Add Delimiter to TCP:** Defines if a delimiting character to add to the message.
- **Char Of TCP Delimiter:** Defines the character to use as a delimiter and add to the message.

Copyright © 2026 Vizrt

4.10.18 DataGeom



The DataGeom plug-in sets the geometry of the container it is assigned to according to the received data. A group with child geometry containers is defined in the scene tree and the DataField receives the index of the selected geometry (index is zero based). The same result can be achieved by using Data3DObject, but Data3DObject accesses the disk when used. To avoid real time rendering problems in heavy scenes, it is better to use DataGeom since it does not access the disk.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataGeom Properties

- **Group Parent:** Defines the parent container of the geometries in the scene tree. The container is dragged to the container area in the DataGeom editor.
- **Copy Container:** Defines if the geometry is copied as a child container to the DataGeom controlled container when sent.

Example

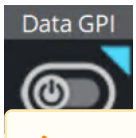
Given the following scene tree:

The DataPool assignment `Geom=0` ; changes the geometry of the GEOM container to sphere, `Geom=1` ; changes it to cube and so on.

Note: If an index value that does not exist is sent, then zero value is used (the first geometry under the group).

Copyright © 2026 Vizrt

4.10.19 DataGPI



The DataGPI plug-in controls visual output and target according to the incoming data.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

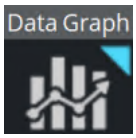
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataGPI Properties

- **Output Field Name:** Defines the name of the DataField that receives the animated value of the value parameter every video field.
- **Port (COM) Number:** Defines the serial communication port for information.
- **Box Type:** Defines the visual output device.
  - **In Active:** Uses the active connection.
  - **SRC-16:** Uses a serial connection.

Copyright © 2026 Vizrt

4.10.20 DataGraph



The DataGraph plug-in works on the Graph geometry plug-in in Viz. It enables controlling the values of the graph points.

Incoming data values of the DataField must contain all the point values according to the selected format, separated by spaces.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataGraph Properties

- **Format:** Defines the axis that is controlled by DataGraph. If one axis is selected (X or Y) then the data should contain one value per point, if X Y is selected, then two values should be sent per point.
- **Normalize:** Forces the graph points to use the entire height range (the range between Position Min and Position Max parameter) according to the minimum and maximum values received when set to `On`. The minimum value is used as the Data Min value and the maximum value is used as the Data Max, using the entire graph range for the received values.

Example

If format is set to X Y, then for drawing four point of the graph, eight numbers should be sent: `Abc=0 0 10 5 12 10 8 15;`

- The first point is located at `X=0`, `Y=0`.
- The second point is located at `X=10`, `Y=5`.
- The third point is located at `X=12`, `Y=10`.
- The fourth point is located at `X=8`, `Y=15`.

Copyright © 2026 Vizrt

4.10.21 DataGraphPoint



The DataGraphPoint plug-in enables to control values of a selected point in a Graph object, 2D Ribbon object and 2D Patch object.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataGraphPoint Properties

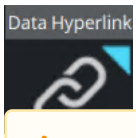
- **Use DataPool Container:** Replaces the Graph Container parameter with the Container Field parameter when set to `On`. This enables a dynamic control of different graph objects defined in the Container Field parameter.
- **Container Field:** Defines a DataPool variable which contains a container name, replacing the Graph Container parameter placeholder. This parameter is used from external applications sending data to DataPool enabling the application to change the controlled graph object.
- **Graph container:** Defines the container to be controlled. Drag the geometry object from the Viz tree to the container area.

Note: Graph Container and Container Field parameters operate the same as the Use Remote Container parameter. The Use Remote Container parameter has no effect in the plug-in.

- **Graph Point ID:** Defines the point to control. ID is the index of the point in the geometry plug-in. ID 1 refers to the first point and so on.
- **Point Offset:** Defines an offset of the point index from the first point. The actual point ID is the sum of Graph Point ID and Point offset.
- **Format:** Defines the axis or width to be controlled by the plug-in. If one axis is selected (X or Y) or if width is selected (W), then the data should contain one value. If X Y is selected, then two values should be sent. If X Y W are selected, three values should be sent.

Copyright © 2026 Vizrt

4.10.22 DataHyperlink



The DataHyperlink plug-in receives mouse events and triggers actions accordingly.

The actions are Viz and DataPool commands.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

This plug-in gets the mouse events from Viz and it does not require the DataMouseSensor plug-in added to the scene. When a DataHyperlink plug-in is used in a scene, a set of predefined DataPool variables is created and updated on every change in the scene. These variables contain information about the mouse cursor location and other mouse events:

- **MX:** Defines cursor location on the X axis.
- **MY:** Defines cursor location on the Y axis.
- **MXY:** Defines cursor location on the X axis and Y axis.
- **MXYZ:** Defines cursor location on the X axis, Y axis and Z axis.
- **MOUSE_DRAG:** Indicates whether a mouse drag is performed.
- **MOUSE_CLICK:** Indicates whether a mouse button is currently pressed.
- **DRAGGED_CONT:** Returns the ID of the currently dragged container
- **DROPPED_CONT:** Returns the ID of the container that was dropped on the container hosting the DataHyperlink plug-in
- **DROP_TYPE:** Defines the type of object that was dropped on the container: No object, geometry, image.

Note: When using DataHyperlink in a scene and no other data is sent externally, adding the DataPool scene plug-in is unnecessary.

DataHyperlink Properties

- **Search Children:** Triggers actions defined on the container that has the plug-in attached to it on mouse events of child containers when set to `On`. When set to `Off`, only mouse events on the container that has the plug-in attached to it trigger the actions.
- **Pick Button:** Defines which mouse button (Left, Middle or Right) triggers the Press and Release events.
- **On Press:** Defines the action that is triggered when the selected button is pressed and the cursor is within the container area.
- **On Release:** Defines the action that is triggered when the selected button is released and the cursor is within the container area.
- **On Enter:** Defines the action that is triggered when the cursor enters the container area.
- **On Leave:** Defines the action that is triggered when the cursor leaves the container area.
- **On Mouse Move:** To be implemented.
- **On Mouse Over:** Defines the action that is triggered while the cursor hovers within the container area.
- **On Mouse Drag:** Defines the action that is triggered while the container is dragged.
- **On Mouse Drop:** Defines the action that is triggered when another object is dropped within the container area.

Copyright © 2026 Vizrt

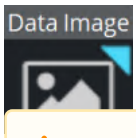
- **Delay Click:** Delays actions triggered by the mouse by five fields when set to `On`. When set to `Off`, no delay is used.
- **Drag For Position:** Enables to container to be dragged and re-positioned in the render window when in On Air mode. Positioning occurs when the left mouse button is pressed and the mouse is dragged in the render window when the related container is selected. When set to `Yes`, additional parameters are enabled.
- **Use Drop Z-Buffer Value:** Changes perspective and container area when dragging an object around. This parameter defines whether the cursor location follows the container area through out the dragging of the object or the object moves according to the mouse movements. Due to perspective changes, the cursor might exit the container area while dragging.
- **Limit X Position:** Defines whether to limit the movement of the object on the X axis while dragging. When set to `Yes`, **Min X Position** and **Max X Position** are enabled, defining the minimum/maximum position on the X axis that the object can be dragged to.
- **Limit Y Position:** Defines whether to limit the movement of the object on the Y axis while dragging. When set to `Yes`, **Min Y Position** and **Max Y Position** are enabled.
- **Limit Z Position:** Defines whether to limit the movement of the object on the Z axis while dragging. When set to `Yes`, **Min Z Position** and **Max Z Position** are enabled.
- **Drag For Rotation:** Enables container to be dragged and rotated in the render window when in On Air mode. Rotation occurs when the center mouse button is pressed and the mouse is dragged in the render window when the related container is selected. When set to `Yes`, additional parameters are enabled.
- **Limit X Rotation:** Defines whether to limit the rotation of the object on the X axis while dragging. When set to `Yes`, **Min X Rotation** and **Max X Rotation** are enabled.
- **Limit Y Rotation:** Defines whether to limit the rotation of the object on the Y axis while dragging. When set to `Yes`, **Min Y Rotation** and **Max Y Rotation** are enabled.
- **Limit Z Rotation:** Defines whether to limit the rotation of the object on the Z axis while dragging. When set to `Yes`, **Min Z Rotation** and **Max Z Rotation** are enabled.
- **Rotation Sensitivity:** Defines the relation between the mouse movement and the object's rotation. The higher the sensitivity is, the larger mouse movements required to rotate the object to the same point.
- **Drag For Scale:** Enables to container to be scaled when the mouse is dragged in the render window when in On Air mode. Scaling occurs when the right mouse button is pressed and the mouse is dragged in the render window when the related container is selected. When set to `Yes`, additional parameters are enabled.
- **Limit Scale:** Defines whether to limit the scaling of the object while dragging. When set to `Yes`, **Min Scale** and **Max Scale** are enabled.
- **Scaling Sensitivity:** Defines the relation between the mouse movement and the object's scale. The higher the sensitivity, the larger mouse movements required to scale the object to the same size.
- **Use Another Container Bounding Box:** Enables Container For Bounding Box parameter when set to `Yes`. When set to `No`, the container's bounding box is used to capture mouse events.
- **Container For Bounding Box:** References this container's bounding box for events when a container is dragged to the placeholder.
- **Drop Type:** Defines which drop action triggers an event.
    - **NONE:** Does not trigger an event.
    - **Cont Geom:** Triggers a drag event for a dropped geometry.
    - **Cont Image:** Triggers a drag event for a dropped image.

Copyright © 2026 Vizrt

    - **Other Cont Geom:** Enables a Dropped Container parameter when selected. When a container is dropped in the defined Dropped Container parameter, it triggers a drop event.
    - **Other Cont Image:** Enables a Dropped Container parameter when selected. When an Image is dropped in the defined Dropped Container parameter, it triggers a drop event.
- **Use Viewports:** Enables a Camera Number parameter when set to `Yes`. Set the number of camera. The viewport area of the selected camera triggers the events.
- **Force Scene Redraw:** Redraws the render window every field when set to `Yes`. When set to `Off`, Viz renders the graphics normally.

Copyright © 2026 Vizrt

4.10.23 DataImage



The DataImage plug-in changes the image or texture applied to the container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

The data specifies the name of the image applied to the container.

DataImage Properties

- **Set MaterialDefinition BaseColor:** Sets the given image as BaseColor channel on a V4 PBR Material.
- **Set MaterialDefinition Emissive:** Sets the given image as Emissive channel on a V4 PBR Material.

Format

Image names should have the format:

- **IMAGE*<name>:** Where `<name>` is the full path to an image in Viz images library. The prefix `IMAGE*` can be omitted from the name.
- **BUILT_IN*IMAGE*<name>:** For special built-in images like *VIDEO1*, *VIDEO2*…

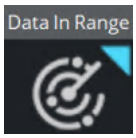
Note: The container must have an initial image attached to it for the plug-in to take effect.

Example

A container with three children and with a DataImage plug-in attached to it. If the Field Name is `IMAGES[3]` and the data is entered as `IMAGES[0-2]=IMAGE*image1, IMAGE*image2, BUILT_IN*IMAGE*VIDEO1;` then the first child shows *image1*, the second *image2* and the third shows *VIDEO1*.

Copyright © 2026 Vizrt

4.10.24 DataInRange



The DataInRange plug-in converts a range of values into a single value.

The value in the Field name parameter is compared to the defined values in the Range Descriptions parameter.

If the input value is found in one of the ranges, it is converted to the value assigned to the range and the result is set in the Result Field Name.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Note: Range values must be numeric values.

DataInRange Properties

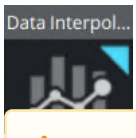
- **Result Field Name:** Defines the name of the DataPool variable that the result of the range conversion is assigned to.
- **Result Field Scope:** Defines the scope of result field. If set to `Local`, the Result Filed Name value is defined as part of a DataPool structure used in an object. If defined as `Global`, the Result field is defined as a regular data field.
- **Range Descriptions:** This parameter contains the different ranges. A range is defined inside brackets, with minimum and maximum values separated by comma. When a round bracket (opening or closing the range), the value next to it is not included in the range (but limiting the range). When square brackets are used the values are included in the range.

Example

In the following line the range of numbers between 1 and 10, excluding 1 and including 10, is converted to the text *Little Numbers*: `(1,10] = Little Numbers;`

Copyright © 2026 Vizrt

4.10.25 DataInterpolate



The DataInterpolate plug-in copies a numeric value to another field.

The value in the Field Name parameter is multiplied with the defined value in the Type parameter.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

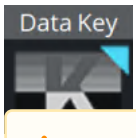
Note: This is an experimental plug-in.

DataInterpolate Properties

- **Result Field Name:** Defines the name of the DataPool variable that the result of the range conversion is assigned to.
- **Result Field Scope**
  - **Local:** Defines the result field as part of a DataPool structure used in an object.
  - **Global:** Defines the result field as a regular data field.
- **Range Descriptions:** Unused parameter.
- **Type**
  - **Two:** Multiplies input value by two.
  - **Ten:** Multiplies input value by ten.

Copyright © 2026 Vizrt

4.10.26 DataKey



The DataKey plug-in toggles the Key function plug-in.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataKey Properties

- **0:** Sets the key to `Off`.
- **Any other Value:** Sets the key to `On`.

Note: Key function plug-in must be assigned to the container.

Copyright © 2026 Vizrt

4.10.27 DataKeyFrame



The DataKeyFrame plug-in affects a named keyframe of an animation defined on the container the plug-in is attached to.

The Keyframe name must be unique.

When the FieldName changes, the plug-in looks for the keyframe and assigns new values to it.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataKeyFrame Properties

- **KeyFrame Name:** Defines the name of the keyframe the plug-in works on. The keyframe name must be unique.
- **Reset KeyFrame:** Defines the name of a keyframe containing initial parameters to be used when a `RESET` string is received. If a `RESET` data is accepted, the keyframe values are set to the values of the Reset KeyFrame. The keyframe name must be unique.
- **Value Format:** Defines the format of the received data.

Note: If data format is not as defined in the Value Format parameter, unpredictable behavior occurs.

Note: When using a combination of DataAction and DataKeyframe2, it's important to ensure that the DataAction plug-in appears after the DataKeyframe2 plug-in in the scene tree.

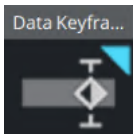
- If all the keyframes are within the same container, the DataAction plug-in must be positioned after the DataKeyframe2 plug-in within that container.
- If multiple containers have DataKeyframe2 plug-ins and you want to use DataAction, the DataAction plug-in should be placed in a separate container positioned below all the other containers containing DataKeyframe2 plug-ins.

Known Issues

When using DataKeyFrame inside a Transition Logic scene, be sure to use unique keyframe names, as searching for a keyframe inside a merged geometry only returns the first one.

Copyright © 2026 Vizrt

4.10.28 DataKeyFrame2



The DataKeyFrame2 plug-in behavior is the same as DataKeyFrame behavior, with the same parameters.

DataKeyFrame2 was created because it is not possible to add the same function plug-in twice to the same container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Note: When using a combination of DataAction and DataKeyframe2, it's important to ensure that the DataAction plug-in appears after the DataKeyframe2 plug-in in the scene tree.

- If all the keyframes are within the same container, the DataAction plug-in must be positioned after the DataKeyframe2 plug-in within that container.
- If multiple containers have DataKeyframe2 plug-ins and you want to use DataAction, the DataAction plug-in should be placed in a separate container positioned below all the other containers containing DataKeyframe2 plug-ins.

Note: This plug-in is deprecated. DataKeyFrame2 became obsolete when the parameter *Use Remote Container* was added to the DataPool plug-in. The is continued for backwards compatibility only, to support scene already using this. When creating new scenes, use the DataKeyFrame with the Use Remote Container option set to `On`.

Copyright © 2026 Vizrt

4.10.29 DataKeyText



The DataKeyText plug-in allows entering text to a container with text geometry using the keyboard.

It allows defining actions to be taken as soon as the data arrives. In order for the data to arrive, the container must be *in focus*.

The container is in focus when it is selected using the mouse.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

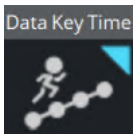
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataKeyText Properties

- **Click To Focus:** Specifies whether the container is always in focus or just when the mouse selects it.
- **On Focus:** Defines what to do when the container has focus. There are three options: First, nothing happens (no effect). Second, a Viz Action is invoked, or third, a DataPool Action is executed. DataPool Action is a series of commands in the same format as entered in the DataPool plug-in.
- **On Defocus:** Defines what to do when the container loses focus. There are three options: First, nothing happens (no effect). Second, a Viz Action is invoked, or third, a DataPool Action is executed. DataPool Action is a series of commands in the same format as entered in the DataPool plug-in.
- **On Enter:** Defines what to do when the container is in focus and the Carriage Return or **ENTER** button is clicked. There are three options: First, nothing happens (no effect). Second, a Viz Action is invoked. Third, a DataPool Action is executed, or fourth, the new line is started in the text.
- **Defocus on Enter:** Removes container focus when a Carriage Return or **ENTER** button is pressed and the On Defocus action is invoked.
- **On Change:** An action that is invoked every time a character is entered to the container. The options are No Effect, Viz Action or DataPool Action.

Copyright © 2026 Vizrt

4.10.30 DataKeyTime



The DataKeyTime plug-in moves a defined keyframe on the animation timeline to the time specified by the value of the DataField. The keyframe must have a unique name assigned to it.

When used with the Remote Container option, multiple keyframes of the same animation can be manipulated by multiple DataKeyTime plug-ins.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataKeyTime Properties

- **Keyframe Type:** Defines the type of animation property to affect: Keyframe, Stop or Pause point, or Action tag.

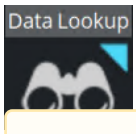
Note: Any property selected must be named uniquely. Otherwise the plug-in does not affect the selected point.

- **Director Name:** Defines the name of the director containing the stop point or pause point to affect. The name is defined in the stage editor of Viz. This parameter is enabled when Key Frame Type is Stop/Pause Point or Action.
- **Channel Name:** Defines the name of the channel containing the action tag to affect. The name is defined in the stage editor of Viz. This parameter is enabled when Key Frame Type is Action.
- **Keyframe Name:** Defines the keyframe that is controlled by the plug-in. The name is defined in the stage editor of Viz and it must be unique.
- **Time Format:** Defines if incoming data value is processed as Frame units (video frames) or seconds.

Note: Key Frames are moved to the absolute time value in the timeline. Abnormal behavior might occur when moving key frames on the timeline, since key frames can change their sequential order in the animation.

Copyright © 2026 Vizrt

4.10.31 DataLookup



The DataLookup plug-in searches data inside complex data objects using index and field name.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataLookup Properties

- **Source:** Defines the Data Object to be searching.
- **Attribute:** Defines the field name of Data Table inside Data Object after selected by index or hash key and get only value of this field.
- **Target:** Defines the DataField that receives the result of the operation.
- **Find Until:** Use when **Source** parameter assigned to DataHashTable with ordered numeric key and searching for element with maximum key value less than value of hash key
- **Return Key:** Use with **Find Until** parameter to get only the matched key value, not use **Attribute** parameter to get the field value inside.

Copyright © 2026 Vizrt

4.10.32 DataLUImage



The DataLUImage plug-in defines a table of aliases which link to images in the Viz Images folders. The value sent in the DataField is one of the alias names. When the plug-in receives the data, it compares the value to all the aliases defined in the table. If an alias matches the value sent, the plug-in replaces the image on the controlled container to that image.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataLUImage Properties

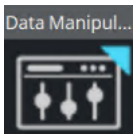
- **Image0..Image19:** Defines image placeholders for defining the images in the table. The images are dragged from the image library in Viz to the image place holders. To remove an image from the table, press the **Reset** button or drag a new image to it.
- **Value0..Value19:** Contains the aliases for each of the images.

Example

If Field Name is set to *MyImage*, then to change the image on the container, use the following command: `MyImage=Value1;` (Or any other value assigned to an image).

Copyright © 2026 Vizrt

4.10.33 DataManipulate



The DataManipulate plug-in enables interactive manipulation of the object that the plug-in is placed on, when in On Air mode.

The left mouse button controls position manipulation, center Mouse button affects rotation and the right mouse button affects scaling.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataManipulate Properties

- **Virtual Manipulation:** Defines if object manipulation is based on Data Fields values. When set to `On`, additional parameters are enabled:
  - **Virtual Position FieldName:** Defines a DataPool variable that controls the position of the manipulated container. The value of the variable defines the delta of the object position from the current position.
  - **Virtual Rotation FieldName:** Defines a DataPool variable that controls the rotation of the manipulated container. The value of the variable defines the delta of the object rotation from the current rotation value.
  - **Virtual Scale FieldName:** Defines a DataPool variable that controls the scale of the manipulated container. The value of the variable defines the delta of the object scale from the current scale.
- **Single Button:** Allows all mouse buttons to control one property of the object when set to `On`. An additional parameter is enabled, setting the controlled attribute of the object:
  - **Single Button Mode:** Selects the control mode of the mouse.
- **Navigation Model:** Selects the type of axis used to manipulate the object.
  - **Cartesian:** Uses the Cartesian coordinate system to calculate object movement.
  - **Polar:** Uses the Polar coordinate system to calculate object movement.
  - **Navigation Model:** Selects the type of axis used to manipulate the object.
- **Other Container is Hot Area:** Defines a remote container as the hot area instead of the container the plug-in is attached to. When set to `On`, an additional parameter is enabled: Hot Area Container, and a container placeholder is added.
- **Is Moveable:** Moves the object when selected and dragged while the left mouse button is pressed, when set to `On`. When set `Off` the object does not move when dragged. Additional parameters are enabled when set to `On`:
  - **Movement:** Determines how the selected object can move. When 2D is selected, the object can move only on the X and Y axes. When 3D is selected, the object can be moved in all three axes.
  - **Limit Move X:** Limits the object's position values according to the values set in the Move Min X and Move Max X when set to `On`. When set to `Off` the object can be dragged all over the render window.

Copyright © 2026 Vizrt

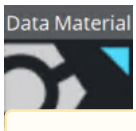
    - **Limit Move Y:** Limits the object's position values according to the values set in the Move Min Y and Move Max Y when set to `On`. When set to `Off` the object can be dragged all over the render window.
    - **Limit Move Z:** Limits the object's position values according to the values set in the Move Min Z and Move Max Z when set to `On`. When set to `Off` the object can be dragged all over the render window.
- **Is Rotation able:** Rotates the object when selected and dragged while the center mouse button is pressed, when set to `On`. When set `Off` the object does not rotate when dragged. Additional parameters are enabled when set to `On`:
    - **Limit Rotate X:** Limits the object's rotation values according to the values set in the Rotate Min X and Rotate Max X when set to `On`. When set to `Off` the object can be rotated to any angle.
    - **Limit Rotate Y:** Limits the object's rotation values according to the values set in the Rotate Min Y and Rotate Max Y when set to `On`. When set to `Off` the object can be rotated to any angle.
    - **Limit Rotate Z:** Limits the object's rotation values according to the values set in the Rotate Min Z and Rotate Max Z when set to `On`. When set to `Off` the object can be rotated to any angle.
    - **Rotation Sensitivity:** Defines the relation between the mouse movement size and the rotation size.
- **Is Scaleable:** Scales the object when selected and dragged while the right mouse button is pressed, when set to `On`. When set `Off` the object does not scale when dragged. Additional parameters are enabled when set to `On`:
    - **Limit Scale:** Limits the object's scaling values according to the values set in the Min Scale and Max Scale, when set to `On`. When set to `Off` the object can be scaled to any size.
    - **Scale Sensitivity:** Defines the relation between the mouse movement size and the scaling size.
- **Writeable:** Controls a text object. If the plug-in is controlling a text object and writable is `On`, when selecting the object the user can type and change the content of the text value. When set to `Off`, the text object cannot be modified.
- **Show Bounding Box:** Toggles visibility for the object's bounding box. When set to `On`, the object's bounding box is visible when the object is selected (in On Air mode). When set to `Off`, the bounding box is not visible when the object is selected.



Note: DataManipulate plug-in does not work if the scene has a DataInteractive plug-in added.

Copyright © 2026 Vizrt

4.10.34 DataMaterial



The DataMaterial plug-in changes the material parameters of the controlled container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

The data specifies any combination of the ambient, diffuse, specular, emission and alpha parameters of the material. The data format is:

```
[AMBIENT r g b] [DIFFUSE r g b] [SPECULAR r g b] [EMISSION r g b] [ALPHA a]
```

This format can specify new values for separate material components.

- **[COLOR r g b]:** Specifies a color using red green and blue values.

Note: All color values (r g b) are numbers between `0` and `255`. Alpha values are float numbers between `0` and `100`.

- **[MATERIAL name]:** Specifies a material name from the Viz material library. The message format is: `MATERIAL MATERIAL*<name>`, where `<name>` is the full path to a material in Viz material library.

Example

- If the received data is `MAT[0-3]=DIFFUSE 255 0 0, DIFFUSE 0 255 0, DIFFUSE 0 0 255;` then the diffuse component of the material of the first child is red, the diffuse component of the material of the second child is green and, the diffuse component of the material of the third child is blue.
- If the data received is `MAT=COLOR 255 0 0;` then all the container materials are set to red.
- If the data received is `MAT=MATERIAL MATERIAL*blue;` then the material is set to the blue material in Viz material library.
- If the data received is `MAT=MATERIAL MATERIAL*<UUID>;` then the material is set to the material of the given UUID.

Note: DataMaterial plug-in does not work unless the controlled containers have a material on them. If controlling child containers from a top node, each of the controlled child containers must have a material attached to it. The top container hosting the DataMaterial plug-in doesn't need to have a material attached to it.

Note: Only first level of child containers with materials are affected, grandchildren containers with materials are not affected.

Copyright © 2026 Vizrt

Advanced Color Control

The **Advanced Color Control** simplifies the way to set the DataMaterial field.

Color Property Mode

If the **Advanced Color Control** option is set to `On`, the drop-down list **Color Property Mode** is shown. The list shows all available modes:

COLOR, AMBIENT, DIFFUSE, SPECULAR, EMISSION, ALPHA, MATERIAL.

The effect when using the **Advanced Color Control** is that the needed Prefix for DataPool is set automatically by the plug-in.

Color Property Example

On a container with a geometry, a material and the plug-in DataMaterial with field set to "mat", these are the DataPool settings:

- With **Advanced Color Control** set to `Off`: `mat=MATERIAL MATERIAL*GLOBALS/redMaterial`
- With **Advanced Color Control** set to `On` and **Color Property Mode** set to *MATERIAL*: `mat=MATERIAL*GLOBALS/redMaterial`

Color Format

> ⚠ **Note:** The Color Format is not enabled for **ALPHA** and **MATERIAL**.

This parameter provides multiple input formats for colors. Supported are following formats:

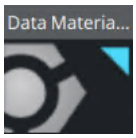
Format	Range RGB	Range Alpha	Example
R G B (int)	0 - 255	-	236 132 39
R G B (float)	0.0 - 1.0	-	0.9255 0.5176 0.1529
RGBA (hex)	00 - FF	00 - FF	EC84277F
ARGB (hex)	00 - FF	00 - FF	7FEC8427
RGB (hex)	00 - FF	-	EC8427
R G B A (int)	0 - 255	0 - 100	236 132 39 50
R G B A (float)	0.0 - 1.0	0.0 - 1.0	0.9255 0.5176 0.1529 0.5

Copyright © 2026 Vizrt

Format	Range RGB	Range Alpha	Example
A R G B (int)	0 - 255	0 - 100	50 236 132 39
A R G B (float)	0.0 - 1.0	0.0 - 1.0	0.5 0.9255 0.5176 0.1529

Copyright © 2026 Vizrt

4.10.35 DataMaterialGradient



The DataMaterialGradient plug-in changes the material color of the controlled container to a color within a predefined spectrum (the plug-in changes the hue of the material).

A minimum value is assigned to one color and a maximum value is assigned to another color.

Any value between the minimum and maximum values is translated to a color, and affects the container's material.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataMaterialGradient Properties

- **Source material:** Assigns the Data Min value to the material/alpha.
- **Target material:** Assigns the Data Max value to the material/alpha.

Example

Data Min is `0` and Data Max is `100`. Source material is red and Target material is green. If the received data is `50` the material that is set on the container is yellow.

Note: DataMaterialGradient plug-in does not work unless the controlled containers have a material on them. If controlling child containers from a top node, each of the controlled child containers must have a material attached to it. The top container hosting the DataMaterialGradient plug-in doesn't need to have a material attached to it.

Copyright © 2026 Vizrt

4.10.36 DataMaterialIndex



The DataMaterialIndex plug-in changes the material of the controlled container. The scene must have a DataMaterialTable scene plug-in with a defined material table.

DataMaterialIndex uses the indices of the defined materials in the DataMaterialTable to change the material on the containers. Data format is an index number (integer). DataMaterialIndex copies the material related to the given index from the table to the controlled container. The controlled container must have material attached to it.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Example

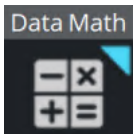
To apply the index to a series of subcontainers, place DatamaterialIndex on the parent container and enter a Field Name (for example, *matindex[0-9]*). This means it changes the material color of the first ten containers to the given values (1-7).

```
MatIndex[0-9]=1,2,3,4,5,6,7;
```

The material on the first child is changed to the material defined in entry 1 in the material table, the second is changed to the material defined in entry 2 of the material table, and so on. The last containers (index 7-9) are set to the material defined in index 7 of the table.

Copyright © 2026 Vizrt

4.10.37 DataMath



The DataMath plug-in performs mathematical calculations with DataPool variables and writes the result to a specified DataField.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataMath Properties

- **Argument 1 and Argument 2:** Defines the arguments used for the mathematical operation. *argument1* is used at the left hand side of the operator. Both arguments must be DataPool variables (DataFields) or constants. If the operation is unitary, only *argument1* is used.

Note: Both arguments (DataFields) trigger the DataMath plug-in and the result is calculated.

- **Result:** Defines the DataField that receives the result of the operation.
- **Operation:** Defines the operator that is used for the calculation.
  - **+ - *:** Adds/subtracts/multiplies the arguments.
  - **/:** Divides *argument1* by *argument2*.
  - **Div:** Provides integral part of the division of *argument1* by *argument2*.
  - **Mod:** Gives the remainder of the division of *argument1* by *argument2*.
  - **^:** Raises *argument1* to the power of *argument2*.
  - **AND:** Performs a logical AND of the arguments: If both are different than zero the result is `1`.
  - **OR:** Performs a logical OR of the arguments: If both are zero the result is `0`.
  - **Cos / Sin:** Determines cosines/sines of *argument1*.
  - **Rnd:** Gives a random value between the two values.
  - **Time ++/--:** Increases/decreases *argument1* by one.
  - **Min/Max:** Performs a numeric comparison of the arguments and returns the lower/higher value.
  - **Avrg:** Calculates the average of the arguments.
- **Decimal Places:** Determines the number of decimal places for the result of the operation.

Copyright © 2026 Vizrt

4.10.38 DataMathObject



The DataMathObject plug-in performs mathematical calculations with DataPool objects and variables and writes the result to a specified result.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataMathObject Properties

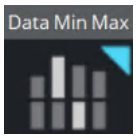
- **Field Name:** Defines the first argument for the mathematical operation. This argument is a DataPool variable or a DataPool Object.
- **Second Argument:** Defines the second argument used for the mathematical operation. The order of the arguments in the mathematical expression is defined in the Operation Order argument.
- **Second Argument Scope:** Defines the scope of the second argument (whether the argument is a local variable, part of an object, or a global DataPool variable).
- **Result:** Defines the DataField that receives the result of the mathematical operation.
- **Result Scope:** Defines the scope of the result (whether the argument is a local variable, part of an object, or a global DataPool variable).
- **Operation Type:** Defines type of operation to perform. According to the selected operation type, a list of operations are displayed. The selected operator is used in the mathematical expression.
- **Operation Order:** Defines the order of the arguments in the mathematical expression.
- **Decimal Places:** Determines the number of decimal places for the result of the operation.

Notes

- If the operation is unitary, only *argument1* is used.
- Only the Field Name parameter triggers DataMathObject and the result is calculated.
- If one of the variables is not defined it is ignored (it is not created automatically like in other DataPool plug-ins).

Copyright © 2026 Vizrt

4.10.39 DataMinMax



The DataMinMax plug-in receives a DataPool array name and returns two DataFields: One containing the minimum value of the array and the other containing the maximum value of the array.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

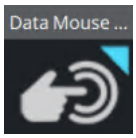
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataMinMax Properties

- **Field Name:** Defines the name of the array to be checked.
- **Min Destination Field:** Assigns the minimum value of the array into a DataField.
- **Max Destination Field:** Assigns the maximum value of the array into a DataField.

Copyright © 2026 Vizrt

4.10.40 DataMouseAction



The DataMouseAction plug-in receives mouse events from the DataMouseSensor plug-in and runs Viz actions accordingly.

DataMouseAction is not registered to a DataField. It is attached to a container and it is triggered by the mouse actions when the cursor is within the container area in the render window.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

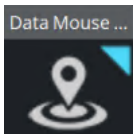
Note: DataPool commands cannot be used.

DataMouseAction Properties

- **Click To Activate:** Determines whether the user has to select the container in order to activate the actions. If set to `On`, it is selected when the first action that is executed (the Mouse Down action).
- **Mouse Threshold:** Determines the minimum number of pixels that the mouse arrow has to be moved before a movement is recognized.
- **Mouse Down:** Calls a Viz action as soon as a mouse button is pressed on top of the container the plug-in is attached to.
- **Mouse Up:** Calls a Viz action when a mouse button is released and the container the plug-in is attached to is the previously selected container.
- **Drag Left:** Calls a Viz action when the selected container is dragged (moved) to the left and the movement is larger then the Mouse Threshold value.
- **Drag Right:** Calls a Viz action when the selected container is dragged (moved) to the right and the movement is larger then the Mouse Threshold value.
- **Stop Horizontal Drag:** Calls a Viz action when a horizontal mouse dragging movement stops or the mouse button that started all the drag movement stops.
- **Drag Down:** Calls a Viz action when the selected container is dragged (moved) down and the movement is larger then the Mouse Threshold value.
- **Drag Up:** Calls a Viz action when the selected container is dragged (moved) up and the movement is larger then the Mouse Threshold value.
- **Stop Vertical Drag:** Calls a Viz action when a vertical mouse dragging movement stops or the mouse button that started all the drag movements stop.

Copyright © 2026 Vizrt

4.10.41 DataMousePosition



The DataMousePosition plug-in receives mouse events from DataMouseSensor and enables dragging the container to attach it to, when in On Air mode.

DataMousePosition is not registered to a DataField. DataMousePosition is attached to a container and it is triggered by the mouse actions when the container is in focus (selected).

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataMousePosition Properties

- **Format:** Specifies in which axis or plane the container moves.
- **X Sensibility:** Defines the sensibility of the mouse movement in the X-axis.
- **Y Sensibility:** Defines the sensibility of the mouse movement in the Y-axis.
- **Click To Activate:** Defines whether the user must click on the container in order to activate it.
- **Snap To Grid:** Specifies whether the object snaps to an imaginary grid whenever the movement is over.
- **Origin X, Y and Z:** Defines the origin of the imaginary grid to be used when Snap To Grid is on.
- **Size X, Y and Z:** Defines the dimensions of the cells of the imaginary grid to be used when Snap To Grid is on.

Copyright © 2026 Vizrt

4.10.42 DataMultiParam



The DataMultiParam plug-in extends the DataParameter plug-in options.

It enables DataPool to control multiple parameters. Incoming data contains all the values for the defined parameters.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataMultiParam Properties

- **Format:** Defines the container parameters to be controlled by incoming data. The format is specified in lines, each containing the following information:
  `<Plug-in Type> <Plug-in Name> <Parameter Name>`
  - `<Plug-in Type>` is either GEOMETRY or FUNCTION.
  - `<Plug-in Name>` is the name of the plug-in that is controlled.
  - `<Parameter Name>` is the name of the parameter in the specified plug-in to control.

Note: Plug-in name and parameter name must be identical to the names used in Viz (names are case sensitive). To find out the correct names, open the Viz command console and change the parameter required. The correct spelling is displayed in the command console as part of the messages sent from Viz.

- **Separator:** Defines a character in the incoming data that separates between the values of the defined parameters. Default separator is a pipe character (|).

Example

If DataMultiParam is attached to a Cylinder object, with an Autorotate function plug-in, Field name is *VALUE* and the controlled parameters are:

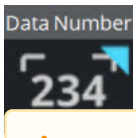
- GEOMETRY Cylinder angle
- GEOMETRY Cylinder hole
- FUNCTION AutoRotate rotspeed

Using the default separator, incoming data is `VALUE=90 | 100 | 230`. The result is:

- Parameter *angle* of the Cylinder is set to the value `90`.
- Parameter *hole* of the Cylinder is set to the value `100`.
- Parameter *rotspeed* of the AutoRotate plug-in is set to the value `230`.

Copyright © 2026 Vizrt

4.10.43 DataNumber



The DataNumber plug-in works on containers with text geometry.

DataNumber formats the display of numeric values according to the parameters set in the plug-in.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataNumber Properties

- **Negative Numbers:** Defines whether to show a negative number with a minus prefix -123,0 or use the parenthesis format (123).
- **Decimal Separator:** Defines which character to use as a decimal separator: a comma or a dot.
- **Replace Zero:** Defines whether to replace a zero value with the string defined in the Replace Zero By parameter.
- **Replace Zero By:** Defines a string that is displayed when the numeric value equals zero.
- **Decimal Places:** Defines how many digits are displayed on the right side of the decimal separator.
- **Force Fraction:** Shows a fraction with the defined decimal places even though the value is an integer or has a fraction with less digits than defined when enabled.
- **Use Thousand Separator:** Defines whether a thousand separator is displayed.
- **Thousand Separator:** Defines which character to use as a thousand separator: A comma or a dot.

Note: When displaying accounting information it is common to use the parenthesis to show negative numbers.

Note: The fraction is rounded according to the value defined in the Decimal places parameter.

Copyright © 2026 Vizrt

4.10.44 DataObject



The DataObject plug-in allows to create a new typed object and link it to a Viz hierarchy. When a DataObject plug-in is attached to a container, a field name and an object type is specified the DataObject plug-in searches Viz hierarchy to look for a parent for the object and children underneath.

Every Data plug-in found under the container where the DataObject is found is tested to see whether it's a field of the object. If it is so, it is attached as a child of the object.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataObject Properties

- **Type:** Defines the type of the object. The list of the types is taken from the configuration file.

Copyright © 2026 Vizrt

4.10.45 DataObjectTracker



The DataObjectTracker plug-in tracks the controlled container and sends the selected attribute values to a defined DataPool variable (DataField).

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

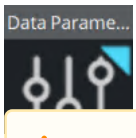
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataObjectTracker Properties

- **Destination Field:** Assigns container values to *DataObjectTracker* from the named *DataField*.
- **Track Type:** Defines which container attributes are tracked by the *DataObjectTracker*.
- **Format:** Defines the format by which the attribute values are sent to the Destination Field.
- **Track when hidden:** Defines if the object tracks when it is hidden.
- **Offset X/Y/Z:** Adds a fixed value to the current values of the container, in the X/Y/Z axis, before setting the values in the Destination Field.
- **Scale X/Y/Z:** Multiplies a fixed value by the current values of the container's scale, in the X/Y/Z axis, before setting the values in the Destination Field.
- **Absolute Position:** Refers all read values from the container in relation to its top parent node (i.e. the returned values include all position, rotation or scaling values of all parent containers of the controlled container) when set to `On`. When set to `Off`, all values read from the container refer to the container itself in relation to its position, rotation or scaling.

Copyright © 2026 Vizrt

4.10.46 DataParameter



The DataParameter plug-in controls other plug-ins (geometry or function plug-ins) parameter values.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataParameter Properties

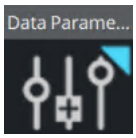
- **Parameter Type:** Defines the controlled plug-in type. The options are `GEOMETRY`, `FUNCTION` or `SCENE`. If Function or Scene is selected, the Function Name parameter is enabled.
- **Function Name:** Sets the name of the function (or scene) plug-in that the DataParameter plug-in controls.
- **Parameter Name:** Defines the parameter that is controlled. The parameter name must be identical to the name used by Viz (case sensitive). To find the exact parameter name use the show command option and change the parameter from the user interface. The name of the parameter is printed by Viz in the commands console.

Example

Suppose a group has three children, each child is a cube. If we want to change the height parameter of each cube according to a certain value received from the field `VALUES[3]`. `Field Name = VALUES[3]`, `Parameter Type = GEOMETRY` (cube is a geometry plug-in), `Function Name = Cube` and `Parameter Name=size_Y` (name of the parameter in the cube plug-in that changes the height of the cube) If the data is `VALUES[0-2]=1, 2, 3;` the cube's heights becomes 1, 2 and 3 respectively.

Copyright © 2026 Vizrt

4.10.47 DataParamTracker



The DataParamTracker plug-in tracks the controlled container and sends the defined parameter values to a defined DataPool variable (DataField).

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

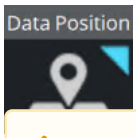
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataParamTracker Properties

- **Destination Field:** Assigns container parameter values to *DataParamTracker* according to the name of the *DataField*.
- **Parameter Type:** Defines the type of the tracked plug-in.
- **Function Name:** Defines what function plug-in is tracked. The function plug-in name must be identical to the name used by Viz (case sensitive). To find the exact parameter name use the show commands console.
- **Parameter Name:** Defines the parameter that is tracked. The parameter name must be identical to the name used by Viz (case sensitive). To find the exact parameter name use the `SHOW COMMAND` option and change the parameter from the user interface. The name of the parameter is printed by Viz in the command console.

Copyright © 2026 Vizrt

4.10.48 DataPosition



The DataPosition plug-in changes the position of the container according to the received data.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataPosition Properties

- **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- **Incremental change:** Defines if the received data is added to the current value of the data field. When set to `On` the received data is added to the data field. When set to `Off` the received data replaces the value of the data field.
- **Separate Parameters:** Enables additional parameters when set to `On`, allowing the user to define minimum and maximum values to all parameters separately:
  - **X Min/Max:** Defines a minimum/maximum value for the received X position of the object.
  - **X Position Min/Max:** Defines a minimum/maximum Viz value for the received X position of the object.

Note: (Y and Z axis have the same parameters. The parameters are enabled according to the selected format).

Example

If Field Name is `VALUE` and the format is X, the data sent is `VALUE=10.5;`. The container moves on the X axis to a position of `10.5`.

Copyright © 2026 Vizrt

4.10.49 DataReader



The DataReader plug-in allows for reading feeds of information from several different types of sources of information.

DataReader supports reading Excel files, databases using SQL, feeds using XML and JSON. The plug-in also supports different types of outputs.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

The plug-in shows different arguments depending on the inputs and the outputs used:

- Functionality
- Input Type - Excel
- Example
    - Preparation
        - Debug Information
        - Data Source
        - DataPool Type
            - Stocks.dp
    - Setup
        - DataReader
        - DataArray
            - READER_CONSOLE
            - DataPool Dump
        - DataTable
            - READER_CONSOLE
            - DataPool Dump
        - DataStructure
            - READER_CONSOLE
            - DataPool Dump
        - DataFieldArray
            - READER_CONSOLE
            - DataPool Dump
        - Shared Memory
            - READER_CONSOLE
- Input Type - SQL
- Input Type - XML
- Input Type - JSON

Functionality

All of the different sources of information behave the same way: They all search for records and translate the records fields to a certain format according to the requested output type. In the case of DataPool output types, the

plug-in converts each incoming record to a DataPool command and executes it. In the case of Shared Memory, the plug-in creates a character separated string of fields and pushes to shared memory.

Important! To read from Microsoft Excel files, Microsoft Access Database Engine is necessary. If the Microsoft Office installer did not install Microsoft Access Database Engine, it needs to be installed with the same software architecture (x64 or x86) as Microsoft Office.

Input Type - Excel

In this mode the plug-in reads a sheet from an Excel file. The parameters in this mode are:

- **File Name:** Determines the name of the file to be read.
- **Table/Sheet:** Determines the name of the sheet to be read from within the file.
- **Key:** Determines the name of the column in the Excel file from which the keys to the DataTable should be taken.
- **From row:** Determines the row number in the Excel sheet from where to start reading the records.
- **Number of rows:** Determines the maximum number of records to read, starting from the row stated in *From row*.
- **Fields to Read:** Specifies by name the columns to read.
- **DP Field Name:** Specifies the field name in DataPool where to write the output.
- **Field is a:** Specifies the type of the target. In case of *Shared Memory*, the following parameters are changed:
  - **DP Field Name:** Hidden.
  - **Shared Memory Variable:** The variable for Shared Memory to be used.
  - **Fields Delimiter:** The delimiter to use for different fields.
  - **Rows Delimiter:** The delimiter to use for different rows.
- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target.
- **Show Data:** Dumps the data the plug-in reads to a DataPool variable called `READER_CONSOLE` when set to `On`. This is extremely useful for development and debugging purposes. This dump includes error messages too.
- **Return Variables Prefix:** Specifies a prefix to add to the name of the DataPool variables that the plug-in uses. For example, if the prefix is `STV_`, then the plug-in dumps the debug information to `STV_READER_CONSOLE`.
- **Load Automatically:** Reads the information automatically every certain amount of time. This amount of time is defined in *Automatic Load Period (in seconds)*.
- **Automatic Load Period (in seconds):** Specifies the period of time between consecutive automatic reads.
- **Load:** Invokes the action of reading the data.

Copyright © 2026 Vizrt

Example

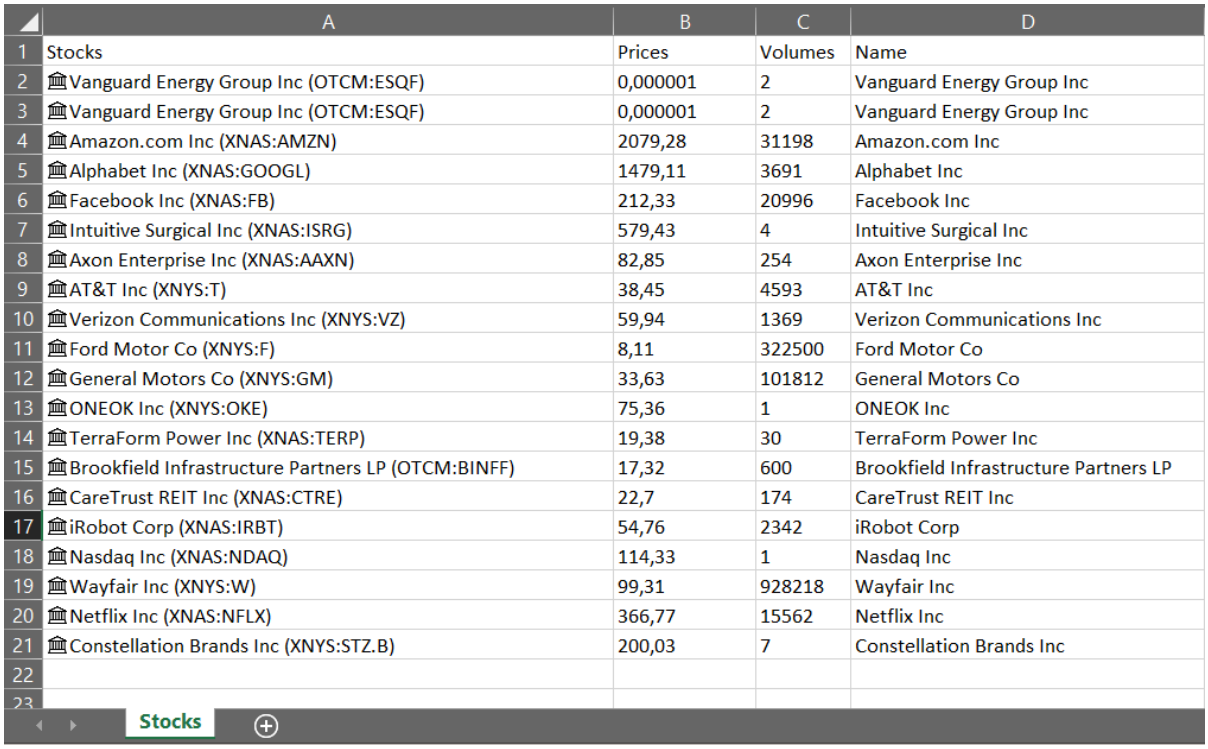
Preparation

Debug Information

For these examples, we want to see if any errors occur or if the data is correct. To prepare this debug mode, create a container with a Font and a DataText plug-in. Set the *Field Name* of the DataText plug-in to *READER_CONSOLE*.

Data Source

The following is a typical example of a table in an Excel sheet we would be interested in reading from using *DataReader*. Create a similar file (the headings have to match) and save it under *C:\temp\Indexes.xlsx*.



	A	B	C	D
1	Stocks	Prices	Volumes	Name
2	🏛Vanguard Energy Group Inc (OTCM:ESQF)	0,000001	2	Vanguard Energy Group Inc
3	🏛Vanguard Energy Group Inc (OTCM:ESQF)	0,000001	2	Vanguard Energy Group Inc
4	🏛Amazon.com Inc (XNAS:AMZN)	2079,28	31198	Amazon.com Inc
5	🏛Alphabet Inc (XNAS:GOOGL)	1479,11	3691	Alphabet Inc
6	🏛Facebook Inc (XNAS:FB)	212,33	20996	Facebook Inc
7	🏛Intuitive Surgical Inc (XNAS:ISRG)	579,43	4	Intuitive Surgical Inc
8	🏛Axon Enterprise Inc (XNAS:AAXN)	82,85	254	Axon Enterprise Inc
9	🏛AT&T Inc (XNYS:T)	38,45	4593	AT&T Inc
10	🏛Verizon Communications Inc (XNYS:VZ)	59,94	1369	Verizon Communications Inc
11	🏛Ford Motor Co (XNYS:F)	8,11	322500	Ford Motor Co
12	🏛General Motors Co (XNYS:GM)	33,63	101812	General Motors Co
13	🏛ONEOK Inc (XNYS:OKE)	75,36	1	ONEOK Inc
14	🏛TerraForm Power Inc (XNAS:TERP)	19,38	30	TerraForm Power Inc
15	🏛Brookfield Infrastructure Partners LP (OTCM:BINFF)	17,32	600	Brookfield Infrastructure Partners LP
16	🏛CareTrust REIT Inc (XNAS:CTRE)	22,7	174	CareTrust REIT Inc
17	🏛iRobot Corp (XNAS:IRBT)	54,76	2342	iRobot Corp
18	🏛Nasdaq Inc (XNAS:NDAQ)	114,33	1	Nasdaq Inc
19	🏛Wayfair Inc (XNYS:W)	99,31	928218	Wayfair Inc
20	🏛Netflix Inc (XNAS:NFLX)	366,77	15562	Netflix Inc
21	🏛Constellation Brands Inc (XNYS:STZ.B)	200,03	7	Constellation Brands Inc
22				
23				

Stocks

DataPool Type

A matching DataPool type is needed for this example. Create a file *%PROGRAMDATA%\Vizrt\VizEngine\DataPool\Stocks.dp* with following content:

Stocks.dp

```
Stocks = {
    string Prices;
    string Volumes;
```

Copyright © 2026 Vizrt

```
    string Name;
};
```

Setup

DataReader

Create a container with a DataReader plug-in on it. In this case, configure the DataReader plug-in as following:

- Query type = EXCEL
- File name = *C:\temp\Indexes.xlsx*
- Table/Sheet = Stocks
- Key = Name
- From Row = 5
- Number of Rows = 4
- Fields To Read = Prices,Volumes,Name
- DP Field Name = Indexes
- Show Data = On

DataArray

Set the DataReader parameter *Field is a:* to *DataArray*.

Create a new container with a DataArray plug-in and set following configuration:

- Field Name = Indexes[4]
- Type = Stocks

Now click the **Load** button in the DataReader plug-in to load the data into the DataArray. You should see the results on the DataText container:

READER_CONSOLE

```
Indexes[0]={
    {
        Prices=""579,43"";
        Volumes=""4"";
        Name=""Intuitive Surgical Inc"";
    };
};

Indexes[1]={
    {
        Prices=""82,85"";
        Volumes=""254"";
        Name=""Axon Enterprise Inc"";
    };
};

Indexes[2]={
    {
```

Copyright © 2026 Vizrt

```
        Prices=""38,45"";
        Volumes=""4593"";
        Name=""AT&T Inc"";
    };
};

Indexes[3]={
    {
        Prices=""59,94"";
        Volumes=""1369"";
        Name=""Verizon Communications Inc"";
    };
};

READER_NUM_RECORDS=9;
```

Click the **Dump** button on the scene plug-in *DataPool* and the console prints your DataPool data:

DataPool Dump

```
Indexes[0-3]={
    {
            Name=Intuitive Surgical Inc;
            Prices=579,43;
            Volumes=4;
    },
    {
            Name=Axon Enterprise Inc;
            Prices=82,85;
            Volumes=254;
    },
    {
            Name=AT&T Inc;
            Prices=38,45;
            Volumes=4593;
    },
    {
            Name=Verizon Communications Inc;
            Prices=59,94;
            Volumes=1369;
    }
};
```

DataTable

Set the DataReader parameter *Field is a:* to *DataTable*. Create a new container with a DataTable plug-in and set following configuration:

- Field Name = Indexes
- Type = Stocks

Copyright © 2026 Vizrt

Now click the **Load** button in the DataReader plug-in to load the data into the DataArray. You should see the results on the DataText container:

READER_CONSOLE

```
Indexes[Intuitive Surgical Inc]={
    {
        Prices=""579,43"";
        Volumes=""4"";
        Name=""Intuitive Surgical Inc"";
    };
};

Indexes[Axon Enterprise Inc]={
    {
        Prices=""82,85"";
        Volumes=""254"";
        Name=""Axon Enterprise Inc"";
    };
};

Indexes[AT&T Inc]={
    {
        Prices=""38,45"";
        Volumes=""4593"";
        Name=""AT&T Inc"";
    };
};

Indexes[Verizon Communications Inc]={
    {
        Prices=""59,94"";
        Volumes=""1369"";
        Name=""Verizon Communications Inc"";
    };
};
```

Click the **Dump** button on the scene plug-in *DataPool* and the console prints your DataPool data (note that the *Key* parameter in the DataReader plug-in is used to obtain the column from the indexes of the table):

DataPool Dump

```
Indexes[Axon Enterprise Inc,Intuitive Surgical Inc,AT&T Inc,Verizon Communications
Inc]={
        {
                Name=Axon Enterprise Inc;
                Prices=82,85;
                Volumes=254;
        },
        {
                Name=Intuitive Surgical Inc;
```

Copyright © 2026 Vizrt

```
                Prices=579,43;
                Volumes=4;
        },
        {
                Name=AT&T Inc;
                Prices=38,45;
                Volumes=4593;
        },
        {
                Name=Verizon Communications Inc;
                Prices=59,94;
                Volumes=1369;
        }
};
```

DataStructure

Set the DataReader parameter *Field is a:* to *DataStructure* and remove the *DP Field Name*. Create a new container with a DataStructure plug-in and set following configuration:

- Field Name = Axon Enterprise Inc

Create three containers under the DataStructure container level and add a font and DataText to every of them. Set the Field Names of the three DataTexts to *Prices*, *Volumes* and *Name*. Now click the **Load** button in the DataReader plug-in to load the data into the DataArray. You should see the results on the DataText container:

READER_CONSOLE

```
Intuitive Surgical Inc={
    Prices=""579,43"";
    Volumes=""4"";
    Name=""Intuitive Surgical Inc"";
};

Axon Enterprise Inc={
    Prices=""82,85"";
    Volumes=""254"";
    Name=""Axon Enterprise Inc"";
};

AT&T Inc={
    Prices=""38,45"";
    Volumes=""4593"";
    Name=""AT&T Inc"";
};

Verizon Communications Inc={
    Prices=""59,94"";
    Volumes=""1369"";
    Name=""Verizon Communications Inc"";
};
```

Copyright © 2026 Vizrt

Click the **Dump** button on the scene plug-in *DataPool* and the console prints your DataPool data (note that the *Key* parameter in the DataReader plug-in is used to obtain the column from the indexes of the table):

DataPool Dump

```
Axon Enterprise Inc ={
    Name=Axon Enterprise Inc;
    Prices=82,85;
    Volumes=254;
}
```

DataFieldArray

Set the DataReader parameter *Field is a:* to *Data Field Array*. Create a new container with a DataText plug-in and set following configuration:

- Field Name = Indexes[2-3]
  This fetches a sub array from the Indexes Data Field Array starting with index 2 (3rd element).

Create a containers under the DataText container level and add a font to it.

Info: We could use multiple sub containers to show multiple values of the created sub array (Indexes[2-3]). This array has two values, so we could create two font containers below the DataText container to display them.

Now click the **Load** button in the DataReader plug-in to load the data into the Data Field Array and display the third value in the DataText's sub container. You should see the results on the DataText container:

READER_CONSOLE

```
DFA[0-8]=""Axon Enterprise Inc"",""AT&T Inc"",""Verizon Communications Inc"",""Ford Motor Co"";READER_NUM_RECORDS=9;
```

Click the **Dump** button on the scene plug-in *DataPool* and the console prints your DataPool data (note that the *Key* parameter in the DataReader plug-in is used to specify the column from the values):

DataPool Dump

```
DFA[0-2]=Axon Enterprise Inc,AT&T Inc,Verizon Communications Inc
```

Shared Memory

Set the DataReader parameters:

- Field is a: = Shared Memory
- Shared Memory Variable = Indexes

Create a new container with a DataText plug-in, a SHMTracker plug-in and a font and set following configuration:

- DataText plug-in:
  - Field Name = Output
- SHMTracker plug-in:

Copyright © 2026 Vizrt

- Output Field Name = Output
- Shared Memory Key Name = Indexes

Now click the **Load** button in the DataReader plug-in to load the data into the Shared Memory. You should see the results on the debug DataText container and the Output DataText container:

READER_CONSOLE

```
579,43|4|Intuitive Surgical Inc|82,85|254|Axon Enterprise Inc|38,45|4593|AT&T Inc|59,94|1369|Verizon Communications Inc|
```

Input Type - SQL

In this mode, the plug-in reads a sheet from a database that supports SQL. The parameters in this mode are:

- **Connection String:** Connects to the database using the connection string. Each type of database has a different connection string. A good reference on connection strings for several different databases can be found in http://www.connectionstrings.com/.
- **SQL Query:** Queries the database using SQL command.
- **Key:** Determines the name of the column in the read query or table from which the keys to the DataTable should be taken.
- **From row:** Determines the row number in the query/table from where to start reading the records.
- **Number of rows:** Determines the maximum number of records to read, starting from the row stated in *From row*.
- **Fields to Read:** Specifies by name the columns to read.
- **DP Field Name:** Specifies the field name in DataPool where to write the output.
- **Field is a:** Specifies the type of the target. In case of **Shared Memory** following parameters are changed:
  - **DP Field Name:** Hidden.
  - **Shared Memory Variable:** The variable for Shared Memory to be used.
  - **Fields Delimiter:** The delimiter to use for different fields.
  - **Rows Delimiter:** The delimiter to use for different rows.
- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target.
- **Show Data:** Dumps the data the plug-in reads to a DataPool variable called `READER_CONSOLE` when set to `On`. This is extremely useful for development and debugging purposes. This dump includes error messages too.
- **Return Variables Prefix:** Specifies a prefix to add to the name of the DataPool variables that the plug-in uses. For example, if the prefix is `STV_`, then the plug-in dumps the debug information to `STV_READER_CONSOLE`.
- **Load Automatically:** Reads the information automatically every certain amount of time. This amount of time is defined in *Automatic Load Period (in seconds)*.
- **Automatic Load Period (in seconds):** Specifies the period of time between consecutive automatic reads.
- **Load:** Invokes the action of reading the data. The different types of outputs are exactly as in the case of Excel.

Copyright © 2026 Vizrt

Input Type - XML

This mode allows for reading XML feeds. These can be read from a remote server or from a local file.

These can be read from a remote server or from a local file.

The parameters in this mode are:

- **File Name:** Shows the full name of a file or a URI to a remote file to be read.
- **Use Authentication:** Allows entering a user name and password In the case the *File Name* is a URI and the remote server requires Basic Authentication.
- **Avoid Cache:** Avoids using cache mechanisms that avoid the refresh of data changing in the server side in the case of a remote file. This is useful when reading information that changes. The caching mechanism avoids the plug-in to get the updates. When “Avoid Cache” is on the plug-in gets the freshest information.
- **XPath:** States what records to search for. For an explanation of the XPath syntax please refer to http://www.w3.org/TR/xpath/. For examples of how to use XPath please refer to https://msdn.microsoft.com/en-us/library/ms256086.
- **Namespaces:** Specifies namespaces for use in XPath expressions when it is necessary to define new namespaces externally. Namespaces are defined in the XML style, as a space-separated list of namespace declaration attributes. You can use this property to set the default namespace as well. An example of the definition of namespaces could be: `xmlns:na='http://myserver.com' xmlns:nb='http://yourserver.com'`
- **Key:** Determines the name of the field in the read records from which the keys to the DataTable should be taken.
- **From row:** Determines he row number in the read record from where to start reading the records.
- **Number of rows:** Determines the maximum number of records to read, starting from the row stated in *From row*.
- **Fields to Read:** Specifies by name the fields to read from the records.
- **DP Field Name:** Specifies the field name in DataPool where to write the output.
- **Field is a:** Specifies the type of the target. In case of **Shared Memory** following parameters are changed:
  - **DP Field Name:** Hidden.
  - **Shared Memory Variable:** The variable for Shared Memory to be used.
  - **Fields Delimiter:** The delimiter to use for different fields.
  - **Rows Delimiter:** The delimiter to use for different rows.
- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target.
- **Input Format:** Specifies the format of the XML file.
- **Output Format:** Specifies the format for the DataPool output.
- **Show Data:** Dumps the data the plug-in reads to a DataPool variable called READER_CONSOLE when set to `ON`. This is extremely useful for development and debugging purposes. This includes error messages too.
- **Return Variables Prefix:** Specifies a prefix to add to the name of the DataPool variables that the plug-in uses. For example, if the prefix is `STV_`, then the plug-in dumps the debug information to `STV_READER_CONSOLE`.
- **Load Automatically:** Reads the information automatically every certain amount of time. This amount of time is defined in *Automatic Load Period (in seconds)*.
- **Automatic Load Period (in seconds):** Specifies the period of time between consecutive automatic reads.

Copyright © 2026 Vizrt

- **Load:** Invokes the action of reading the data.
- **Use Custom Headers:** Adds custom HTTP headers to the request. For example, this can be used to specify an API token for web services.

Input Type - JSON

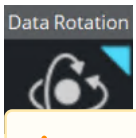
This mode allows for reading JSON feeds. These can be read from a remote server or from a local file.

The parameters in this mode are:

- **File Name:** Shows the full name of a file or a URI to a remote file to be read.
- **XPath:** States what records to search for. This is a subset of the standard XPath.
- **Key:** Determines the name of the field in the read records from which the keys to the DataTable should be taken.
- **From row:** Determines the row number in the read record from where to start reading the records.
- **Number of rows:** Determines the maximum number of records to read, starting from the row stated in *From row*.
- **Fields to Read:** Specifies by name the fields to read from the records.
- **DP Field Name:** Specifies the field name in DataPool where to write the output.
- **Field is a:** Specifies the type of the target. In case of **Shared Memory** following parameters are changed:
  - **DP Field Name:** Hidden.
  - **Shared Memory Variable:** The variable for Shared Memory to be used.
  - **Fields Delimiter:** The delimiter to use for different fields.
  - **Rows Delimiter:** The delimiter to use for different rows.
- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target.
- **Show Data:** Dumps the data the plug-in reads to a DataPool variable called `READER_CONSOLE` when set to `On`. This is extremely useful for development and debugging purposes. This dump includes error messages too.
- **Return Variables Prefix:** Specifies a prefix to add to the name of the DataPool variables that the plug-in uses. For example, if the prefix is `STV_`, then the plug-in dumps the debug information to `STV_READER_CONSOLE`.
- **Load Automatically:** Reads the information automatically every certain amount of time. This amount of time is defined in *Automatic Load Period (in seconds)*.
- **Automatic Load Period (in seconds):** Specifies the period of time between consecutive automatic reads.
- **Load:** Invokes the action of reading the data.
- **Proxy on/off**
  - **Proxy Mode:**
    - **Auto Detect:** Detects if there is a proxy set in the system or Engine.
    - **System Settings:** Sets the proxy for HTTP requests to the system proxy.
    - **Manual Configuration:** Configures a proxy manually with a *Proxy URL*, *Proxy User Name* and *Proxy Password*.
- **Use Custom Headers:** Adds custom HTTP headers to the request. This can for example be used to specify an API token for web services.

Copyright © 2026 Vizrt

4.10.50 DataRotation



The DataRotation plug-in rotates the container according to the received data.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataRotation Properties

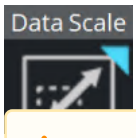
- **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- **Incremental change:** Defines if the received data is added to the current value of the data field. When set to `On` the received data is added to the data field. When set to `Off`, the received data replaces the value of the data field.

Example

If Field Name is `VALUE` and the format is XY, the data sent is `VALUE=10.5 5.0;`. The container rotates around the X axis to an angle of `10.5` and around the Y axis to an angle of `5.0`.

Copyright © 2026 Vizrt

4.10.51 DataScale



The DataScale plug-in scales the container according to the received data.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataScale Properties

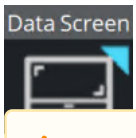
- **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- **Incremental change:** Defines if the received data is added to the current value of the data field. When set to `On` the received data is added to the data field. When set to `Off` the received data replaces the value of the data field.

Example

The current scaling values of the container are `2.0 3.0 4.0`. If the parameters are: `Field Name: VALUE`, `Format: Y`, `Data Min: -100`, `Data Max: 100`, `Scale Min: 0` and `Scale Max: 1` and the data entered is `VALUE=0;`. The resulting scaling values of the container are `2.0 0.5 4.0`

Copyright © 2026 Vizrt

4.10.52 DataScreen



The DataScreen plug-in is used to control screen coordinates transformations of objects.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

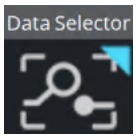
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataScreen Properties

- **Position Format:** Defines if the object's position is affected by incoming data and the format the data. Data is used as screen coordinate values, where the top left corner of the render window is the 0 0 position and the bottom right corner of the render window is 720 576 position.
- **Size Format:** Defines if the object's scaling is affected by incoming data and the format of the data.

Copyright © 2026 Vizrt

4.10.53 DataSelector



The DataSelector plug-in receives the number of a child container as data.

DataSelector toggles visibility for all the child containers of the controlled container and activates only the *n*-th child container, as received in the data.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataSelector Properties

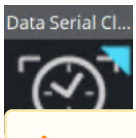
- **First Container Is:** Select `0` or `1` as the first container index. Incoming data is applied accordingly.
- **Value ReScaling:** Disables *First Container is* parameter and enables additional parameters when set to `On`, allowing rescaling of incoming data to the defined values. All incoming data exceeding the Data Min or Data Max is cropped.
- **Switch all on until value:** Switches on visibility for all child containers until the first value is accepted in the DataField when set to `On`. After the first value is set, the DataSelector only switches one child container at a time. When set to `Off`, child containers keep their current status until the first value is accepted in the DataField.
- **Enable Dissolve:** Uses a fade transition between the current container and the next container.

Example

`Field Name=VALUE`. If the data entered is `VALUE=3` then all the children containers are invisible out of the fourth (0-3) child.

Copyright © 2026 Vizrt

4.10.54 DataSerialClock



The DataSerialClock plug-in controls a clock through a serial port connection.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataSerialClock Properties

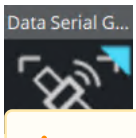
- **DP Variables Prefix:** Adds prefix to a set of fixed names, based on time scale, for different time variables. The prefix is used to distinguish one set of time parameters from the other. The default prefix is *CD1*. For example, if Prefix is set to *DDay*, the corresponding variables are:

```
DDay_DAYS
DDay_HOURS
DDay_HOURS_TOTAL
DDay_MINS
DDay_MINS_TOTAL
DDay_SECS
DDay_SECS_TOTAL
DDay_SIGN (i.e + for future and: for past)
```

- **Port (COM) Number:** Defines the serial communication port for information.
- **Leading Zero:** Adds a leading zero to all single digit numbers (0-9) when set to `On`.

Copyright © 2026 Vizrt

4.10.55 DataSerialGPS



The DataSerialGPS plug-in sends and receives GPS data through a serial port connection.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

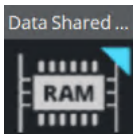
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataSerialGPS Properties

- **Port (COM) Number:** Defines the serial communication port for information.

Copyright © 2026 Vizrt

4.10.56 DataSHM



The DataSHM (Shared Memory) plug-in uses a defined data field or a DataPool expression to update a string type shared memory entry.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

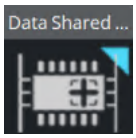
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataSHM Properties

- **Shares Memory Type:** Selects the shared memory in Viz 3 that the memory key is defined in. For additional information about Shared Memory, refer to the Viz Artist User Guide.
- **Shares Memory Key Name:** Defines the shared memory key name that receives the data field value or DataPool expression.
- **Transferred Argument:** Selects the argument type that is sent to the shared memory key. Select **Variable Value** to set the data field value to the memory key. Select **DP Expression** to send a DataPool expression to the shared memory key. When DP Expression is selected, an additional parameter is enabled: Expression String. Define the DataPool expression that results in a string. The resulting string is sent to the shared memory key.

Copyright © 2026 Vizrt

4.10.57 DataSHMTracker



The DataSHMTracker plug-in monitors a string type shared memory entry.

When the shared memory entry is changed, the plug-in uses the updated value to update the related DataPool variable.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

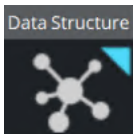
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataSHMTracker Properties

- **Shares Memory Type:** Selects the shared memory in Viz 3 that the memory key is defined in. For additional information about Viz 3 shared memory, refer to Viz 3 user guide.
- **Shares Memory Key Name:** Defines the tracked shared memory key name. Whenever the shared memory key changes, its value is copied to the defined data field.

Copyright © 2026 Vizrt

4.10.58 DataStructure



The DataStructure plug-in uses structures that were defined in the DataPool scene plug-in.

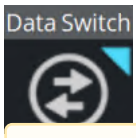
The structure format is the same as in the *config.dp* file but the structure is defined in the scene making it available only to variables in the scene (Unlike structures defined in the *config.dp* file which are available for all scenes to use). The Field name relates to a variable defined as the type of a structure.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Copyright © 2026 Vizrt

4.10.59 DataSwitch



The DataSwitch plug-in toggles the visibility of the container according to the incoming data.

Any value other than zero sets the container's visibility to `On`, zero sets the visibility to `Off`.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

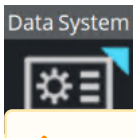
Information: DataPool plug-ins only work properly on a single channel on the same machine.

Example

`Field Name=VALUES[3]`. If the data entered is `VALUES[0-2]=0, 1, 0;`, then the first and third children of the container the plug-in is attached to is deactivated and the second child is activated. If the data entered is `VALUES=0`, the container itself is deactivated. If the data is `VALUES=1`, the container itself is activated.

Copyright © 2026 Vizrt

4.10.60 DataSystem



The DataSystem plug-in enables running external applications/commands.

Incoming data contains the command to run.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataSystem Properties

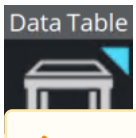
- **No Wait:** Defines whether Viz waits for the launched application to finish its run.

Example

Field Name is CMD. Incoming data is CMD=`calc`;. The result is that the windows calculator opens.

Copyright © 2026 Vizrt

4.10.61 DataTable



The DataTable plug-in defines a table of typed objects.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataTable Properties

- **Table Name:** Defines the name of the table.
- **Type:** Defines the type of the objects in the table. The list of the types is taken from the configuration file.

Copyright © 2026 Vizrt

4.10.62 DataTemo



The DataTemo plug-in creates an image animation sequence, using a method much like the one used for making a cartoon film.

The basic input for DataTemo is a single image consisting of many tiled and equally sized squares set up in a matrix.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataTemo Properties

- **Format:** Specifies the format of the incoming data. Format options specify the axis which the data relates to.
- **Matrix Width (Cells in row):** Determines the number of cells for each row.
- **Matrix Height (Cells in column):** Determines the number of cells for each column.

Note: Texture mapping should be *Vertex* to prevent an incorrect texture ratio, especially if *Matrix Width* and *Matrix Height* are different.

Example

Show Japanese day of week texture from a single image.



1. Import an image that contains all seven day of week symbols, starting with Monday.
2. Drop the image into the scene.
3. Change texture mapping to **Vertex**.
4. Add DataTemo to the same container of the image.
5. Keep Format property as **X** because this example image contains all symbols in horizontal order.
6. Change Matrix Width property to `7`. You may have to change the scaling of the container or add a Rectangle geometry to correct the size and scale of clipped symbol.
7. Set Field Name property to DataPool variable that contains day of week index start from 0 (Monday). To use current day of week from DataClock, follow these steps:
    a. Go to **Built Ins > Scene Plugins** tab and **Scene Settings > Plugin** tab.
    b. Add DataClock to the scene.
    c. Go back to **Built Ins > Container Plugins** tab, add DataMath into the same container.
    d. Set Argument 1 property to `WEEKDAY`.
    e. Set Result property to `WEEKDAY0` because WEEKDAY from DataMath starts from `1`.
    f. Set Operation property to --.
    g. Set Field Name property of DataTemo to `WEEKDAY0`.

Copyright © 2026 Vizrt

4.10.63 DataText



The DataText plug-in controls text objects.

DataText receives incoming data, adds a defined prefix and a defined suffix to the data, and changes the text value (string) to the result.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataText Properties

- **Trim Option:** Enables/disables trimming. When set to `Off`, incoming data is not trimmed. When set to `On`, additional trim options for incoming data are added:
  - **Remove Prefix Until:** Removes all characters from the beginning of the string to the defined delimiting substring (including) when set to `On`. An additional parameter is enabled: Sub String, to define a limiting substring.
  - **Remove Sufix From:** Removes all characters from the defined substring (inclusive) to the end of the string when set to `On`. An additional parameter is enabled: Sub String, to define the substring.
  - **Choose an entry:** Enables additional parameters when set to `On`.
  - **Delimiter:** Defines a delimiting character.
  - **Number:** Defines the entry number of the beginning with the delimiter character. The incoming string is split into substrings, using delimiter Y as the split point (end of sub string), and substring X is used as the data (delimiting characters are not included in the substrings).
  - **Choose Characters:** Enables the Range parameter when set to `On`. The data used by the plug-in is a simple range of bytes X-Y defined in Range (zero is not a valid value as a character number).

Note: When using more than one trim option the AND operator is used. If all options are used the following result is used: Remove prefix AND remove suffix AND split data AND select substring number X AND select bytes number X-Y.

- **Replace backslash n by EOL:** Enables/disables replacement of control characters. When set to `Off`, incoming data is not changed. when set to `On`, all `\n` (backslash n) control characters are replaced with the End Of Line control character.
- **Convert Case:** Selects the required option for data case conversion:
  - **None:** Does not change any data strings.
  - **Lower:** Converts all data strings to lower case only.
  - **Upper:** Converts all data strings to upper case only.
  - **Word:** Converts the first letter of every word to upper case.
  - **Sentence:** Converts the first letter of every sentence to upper case.

Copyright © 2026 Vizrt

Example

Field Name= `PRICE`

Prefix= `$`

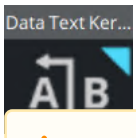
Suffix= `is the price`.

If the data is PRICE=123 then the result is: `$123 is the price.`

> **Note:** When sending special characters, or characters used as DataPool separator characters, to *DataText* plug-in, use double-double quotes at the beginning and ending of the string: ""*What's the frequency, Kenneth?*"" If the quotes are omitted, the string is not displayed correctly.

Copyright © 2026 Vizrt

4.10.64 DataTextKerning



The DataTextKerning plug-in sets the kerning of a text object.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

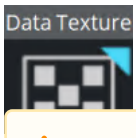
Information: DataPool plug-ins only work properly on a single channel on the same machine.

Example

Field Name=VALUE. If the entered data is VALUE= `12` ; then the kerning of the text object is assigned the value `12` .

Copyright © 2026 Vizrt

4.10.65 DataTexture



The DataTexture plug-in controls various texture attributes such as mapping, scaling, etc.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Incoming data contains the parameter name of the texture and the value.

DataTexture Properties

- **TEXT:** Creates a texture on the controlled container. Value is `IMAGE*<image name>` where image name is a full path to an image in Viz images library.
- **MAPT:** Defines the mapping type of the texture. Values are `1` (LINEAR), `2` (VERTEX), `3` (REFLECT).
- **ENVT:** Defines the environment type of the texture. Values are `3042` (blend), `8449` (decal) or `8448` (modulate).
- **QUAL:** Defines the texture quality type. Values are `1` (PIXEL), `2` (LINEAR), `3` (MIPMAP), `4` (SHARPEN).
- **COLO:** Defines the color quality of the texture. Values are `1` (32 bits), `2` (16 bits).
- **EFFT:** Defines an effect type used on the image. Values are `0` (Smooth), `1` (Mosaic).
- **WRAP:** Defines the wrapping type of the texture. Values are `10497` (REPEAT), `10496` (CLAMP).
- **POSX:** Position of the texture on the X axis of the container. Value should be a float number.
- **POSY:** Position of the texture on the Y axis of the container. Value should be a float number.
- **ROTZ:** Rotation of the texture around the Z axis. Value should be float number.
- **SCAX:** Scales texture on the X axis. Value should be float number.
- **SCAY:** Scales texture on the Y axis. Value should be float number.
- **BLEN:** Provides blend value of the texture. Value should be a float number.
- **EFFV:** Defines the amount of the effect defined in the `EFFT` (effect type) that is applied to the texture. Value should be a float number.
- **IMGS:** Changes the image of a given container. Value is `<Xvalue> <Yvalue>`.
- **SETI:** Replaces the image on the container. Data format is `IMAGE*<image Name>` where image name is the full path to an image in the Viz images library.
- **DRAW:** Sets the scale and bias values of the texture. Values are *<texcoord scale> <float bias s> <float bias t>*. all values are float numbers.

Copyright © 2026 Vizrt

4.10.66 DataTime



The DataTime plug-in translates the value of the incoming data into date/time elements and stores these values to a set of predefined DataPool variables.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Predefined Variables

The predefined variables are:

- **_DAY:** Day of the month (`1-31`).
- **_DAYNAME:** Name of the day in the week. Names are defined in the Day Names parameter.
- **_FHOUR:** Hours/min/sec described in floating point number (decimal) i.e. `6:30` is converted to `6.5` .
- **_HOUR:** Hour of the day (`0-23`).
- **_HOURNAME:** Set of name description for the hours of the day. Names are defined in the Hour Names parameter.
- **_MIN:** Minute of the hour.
- **_MONTH:** Month number in the year (`1-12`).
- **_MONTHNAME:** Name of the month. Names are defined in the Month Names parameter.
- **_MSEC:** One thousandth of a second in the second (`0-999`). This variable is used only when the Resolution parameter is set to milliseconds.
- **_SEC:** Second number in a minute (`0-59`).
- **_WEEKDAY:** Number of the day in the week (`1-7`).
- **_YEAR:** Number of the year using four digits.
- **_RELDAY:** Relative day number, where `0` (zero) is today. This variable returns an integer value only (the difference between the received data and the current system time).
- **_RELDAYNAME:** Relative day name, where yesterday, today and tomorrow are used for `-1` , `0` , `1` values received in the `RELDAY` variable. All other days are displayed by name (Sunday, etc.).
- **_HOURS_SHORT:** Show 12 hour clock time.
- **_AMPM:** Display AM or PM (used with `HOURS_SHORT`).

DataTime Properties

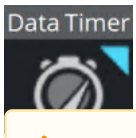
- **DP variables Prefix:** Defines a prefix that is added to the DataTime variables. The prefix is used to distinguish one set of time parameters from the other, enabling the usage of multiple time values. Default is `T1` . If the prefix is set to `T1` , the corresponding variables are:
  - T1_DAY
  - T1_DAYNAME

Copyright © 2026 Vizrt

    - T1_FHOUR
    - T1_HOUR
    - T1_HOURNAME
    - T1_MIN
    - T1_MONTH
    - T1_MONTHNAME
    - T1_MSEC
    - T1_SEC
    - T1_WEEKDAY
    - T1_YEAR
- **DP variables Scope:** Uses data variables as part of the structure the variables were defined in when set to `Local`. When set to `Global`, the time data variables are used as global DataPool variables.
- **Input is Machine Time:** Defines if the time is read from the machine's clock or from the data field. If set to `On`, the Field Name parameter is disabled and additional parameters are enabled:
- **Input Field Format:** Selects the input field units.
- **Output Field Time Zone:**
    - **As Is:** Leaves timezone data as set.
    - **UTC -> Local:** Converts UTC timezone to local timezone.
    - **Local -> UTC:** Converts local timezone to UTC.
- **Offset Field:** Defines the offset units.
- **Offset:** Defines the number of offset units from the current machine time.
- **Leading Zero:** Defines if the zero character is added to the single digit values (`0-9`). This option is useful when formatting the time display.
- **Day Names:** Defines a set of names for the days of the week, to be used by the `_DAYNAME` variable. The parameter should contain seven names, comma separated. If the parameter is left blank, *DataTime* uses the default of Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.
- **Month Names:** Defines a set of names for the months of the year, to be used by the `_MONTHNAME` variable. The parameter should contain twelve names, comma separated. If the parameter is left blank, *DataTime* uses the default of January, February, etc.
- **Hour Names:** Defines a set of names for the hours of the day, to be used by the `_HOURNAME` variable. The parameter should contain twenty four names, comma separated. If the parameter is left blank, *DataTime* uses the default of Midnight, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, Noon, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23.
- **Replace Yesterday By:** Allows the replacement of the term *yesterday* for another one. This feature is useful when working on a language other than English.
- **Replace Today By:** Allows the replacement of the term *today* for another one. This feature is useful when working on a language other than English.
- **Replace Tomorrow By:** Allows the replacement of the term *tomorrow* for another one. This feature is useful when working on a language other than English.
- **Initialize:** Calculates variable values now.
- **Reset:** Recalculates variable values now.
- **Week Director:** Specifies a Director that is controlled by the DataTime value. The Director's animations start on a new week and ends with that week.

Copyright © 2026 Vizrt

4.10.67 DataTimer



The DataTimer plug-in runs an action (Viz action or DataPool action) at a fixed time interval.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

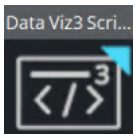
Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataTimer Properties

- **Action:** Defines the action that is executed at the interval time.
- **Enabled:** Executes the action at the defined time intervals when set to `On`. When set to `Off` the plug-in is disabled and the action is not executed.
- **Interval:** Defines the time length, in milliseconds, between executions of the action.

Copyright © 2026 Vizrt

4.10.68 DataViz3Script



The DataViz3Script plug-in invokes a predefined Viz 3 scripting subroutine or function, with the updated value of its defined data field (DataPool variable) or with a DataPool expression as an argument.

In the case of a function, the returned value of the function can be assigned to a DataPool variable.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataViz3Script Properties

- **Script Type:** Defines the location of the script, containing the requested function/procedure. Select **Container** to use a function in a container script or **Scene** to use a function in the scene setting script module.
- **Function Name:** Sets the name of the function/procedure to be called when the data field changes.
- **Transferred Argument:** Selects the argument that is sent to the function/procedure. Select **Variable Value** to set the data field value to the memory key and to use the data field value as the function argument. Select **DP Expression** to send a DataPool expression to the function/procedure. When DP Expression is selected, an additional parameter is enabled: Expression String. Define the DataPool expression that results in a string. The resulting string is sent as an argument (or arguments) to the function/procedure.
- **Use Return Value:** Defines if the returned value from the defined function is re-used by the plug-in. When set to `Off`, no return value is used. When set to `On`, additional parameters are enabled:
  - **Output Field:** Sets the data field name that the returned value is assigned to.
  - **Output Scope:** Defines the output field scope: Local or global. The value returned from the function is assigned to the defined output field.

Copyright © 2026 Vizrt

4.10.69 DataWPosition



The DataWPosition plug-in affects World Position (a Viz World plug-in) attached to the controlled plug-in.

The DataPool variable defined in the Field Name should contain the values of the Longitude, Latitude and Altitude to be sent to the World Position plug-in.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Copyright © 2026 Vizrt

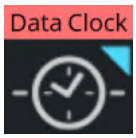
4.11 DataPool Scene Plug-ins

The scene plug-ins are listed in alphabetical order:

- DataClock
- DataInteractive
- DataMaterialTable
- DataMouseSensor
- DataPool Plug-in

Copyright © 2026 Vizrt

4.11.1 DataClock



The DataClock plug-in is a scene plug-in that sends the current system time to predefined DataPool DataFields.

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataClock Properties

- **Leading Zero on All:** Adds a leading zero to all the single digit numbers (0-9) of hours, minutes and seconds.
- **Leading Zero on MINS, SECS:** Adds a leading zero to the single digit numbers (0-9) of minutes and seconds only.

Fields

The fields DataClock generates are:

- **DAY:** Current day (`1-31`).
- **MONTH:** Current month (`1-12`).
- **YEAR:** Current year (four digits).
- **WEEKDAY:** The day of the week (`1-7`). The week starts with Sunday (1) and ends with Saturday (7).
- **DAY_NAME:** The textual name of the day.
- **MONTH_NAME:** The textual name of the month.
- **HOURS:** Hour of the day (`0-23`).
- **MINS:** Minute of the hour (`0-59`).
- **SECS:** Second of the minute.
- **AMPM:** AM or PM.
- **COUNTER:** A running counter of seconds. Its value is equal to 60*60*HOURS + 60*MINS + SECS.

Copyright © 2026 Vizrt

4.11.2 DataInteractive



The DataInteractive plug-in receives mouse and keyboard events and triggers actions accordingly. The actions are DataPool commands.

This plug-in is a scene plug-in and it gets the mouse events from Viz (DataMouseSensor is not required).

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Notes

- The interactive events are sent only when Viz is in On Air mode.
- When using DataInteractive in a scene and no other data is sent externally, adding the DataPool plug-in scene plug-in is unnecessary.
- When using DataInteractive plug-in, DataManipulate and DataClick cannot be used.

DataInteractive Properties

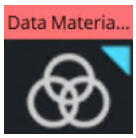
- **Click Type Left:** Executes the action defined in the Click Action Left Mouse parameter every time the left mouse button is clicked when set to `Always`. When set to `Clear Area Only`, the action is triggered if the mouse is clicked on the background (not on any container area). When set to `Never`, the action is disabled.
- **Click Action Left Mouse:** Defines a DataPool action that is performed when the mouse is clicked.
- **Click Type Middle:** Executes the action defined in the Click Action Middle Mouse parameter every time the left mouse button is clicked when set to `Always`. When set to `Clear Area Only`, the action is triggered if the mouse is clicked on the background (not on any container area). When set to `Never`, the action is disabled.
- **Click Action Middle Mouse:** Defines a DataPool action that is performed when the mouse is clicked.
- **Click Type Right:** Executes the action defined in the Click Action Right Mouse parameter every time the left mouse button is clicked when set to `Always`. When set to `Clear Area Only`, the action is triggered if the mouse is clicked on the background (not on any container area). When set to `Never`, the action is disabled.
- **Click Action Right Mouse:** Defines a DataPool action that is performed when the mouse is clicked.
- **Mouse Action 0 (to 9) Active:** Enables additional parameters when set to `On`. The action is disabled when set to `Off`.
    - **Alt For Action 0 (to 9):** Defines if the **ALT** button must be pressed in conjunction with the defined key in the parameter Key For Action 0 (to 9).
    - **Ctrl For Action 0 (to 9):** Defines if the **CTRL** button must be pressed in conjunction with the defined key in the parameter Key For Action 0 (to 9).

Copyright © 2026 Vizrt

- **Shift For Action 0 (to 9):** Defines if the **SHIFT** button must be pressed in conjunction with the defined key in the parameter Key For Action 0 (to 9).

Copyright © 2026 Vizrt

4.11.3 DataMaterialTable



The DataMaterialTable plug-in enables the user to create a table of materials and store this table to a file, enabling the same table to be used in different scenes.

When a table is defined at the scene level (by loading a table from a file or defining material entries) it is accessible by DataMaterialIndex.

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

Note: The material table has a maximum of 256 entries. Index range is `0-255`. When saving/reading the material table file all the entries are saved/read.

DataMaterialTable Properties

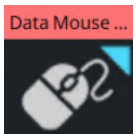
- **Save Table to File:** Saves table data to the defined file (in the Table File Name parameter) when pressed.
- **Load Table from File:** Loads table data to the defined file (in the Table File Name parameter) when pressed.
- **Table File Name:** Defines the name of a file for storing/reading the material table. Suffix for DataMaterialTable files is *.dmt*.
- **Auto Load Table File:** Loads the color table file during scene load when set to `On`. When set to `Off`, the scene loads without reloading the color table file.
- **Per Scene Table:** Loads and stores the color table used in the scene in a separate area from the common color table when set to `On`. When set `Off`, the scene uses a common color table defined in the table file.

Note: When using the common color table, every time the color table is loaded it overwrites the existing table. When using the Per Scene Table option, a color table is created for each scene.

- **Index:** Defines the entry in the material table (index) to be edited. Index is zero based (the first entry is `0`, the second is `1`, and so on).
- **Enlighted:** Defines whether the material in the entry index is lit.
- **Ambient:** Defines the ambient component of the material in the entry index of the table.
- **Diffuse:** Defines the diffuse component of the material defined in the current index of the table.
- **Specular:** Defines the specular component of the material defined in the current index of the table.
- **Emission:** Defines the emission component of the material defined in the current index of the table.
- **Shininess:** Defines the shininess component of the material defined in the current index of the table.
- **Copy from container:** Allows copying a material from a given container to the material table. The material parameters are set when the container is dragged to the container place holder, in the plug-in editor. The material table is not changed if the material of the container is changed.

Copyright © 2026 Vizrt

4.11.4 DataMouseSensor



The DataMouseSensor plug-in receives mouse events and keyboard events from Viz and stores the events in special DataPool variables.

Once the events are stored in DataPool variables, this information is accessible to other DataPool plug-ins.

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

The variables that it creates are:

- **MPOS:** Returns the mouse position when a button is clicked. If a button is pressed while moving the cursor MPOS updates dynamically. When the button is released the mouse position value is stored in MPOS. The position is displayed in X Y values.
- **MBUT:** Returns the number of the last clicked mouse button (`1` =left), whether the button is currently pressed or released, and the X Y values like in MPOS.
- **MSELCONT:** Returns the ID of the selected container.
- **KEY:** Returns the ASCII code of the last pressed key on the keyboard and whether it is currently pressed (pressed= `1` , released= `0`).

Note: Not all keyboard buttons are displayed when using the KEY variable.

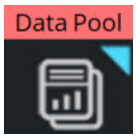
- **MONOFF:** Indicates whether a mouse button is currently clicked or released.
- **MBUTTON:** Returns the number of the last clicked mouse button. `1` =left button, `2` =center button, `3` =right button.

The DataMouseSensor is a scene plug-in and it should be used in scenes that use mouse/keyboard DataPool plug-ins.

The DataMouseSensor plug-in has no parameters.

Copyright © 2026 Vizrt

4.11.5 DataPool Plug-in



The DataPool plug-in is a scene plug-in that manages the DataPool mechanism and data transfer in the scene level.

DataPool manages the DataFields, values, and registered plug-ins table.

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Data

Information: DataPool plug-ins only work properly on a single channel on the same machine.

DataPool Properties

- **Data:** Allows the user or external software to define new DataFields and assign data to DataFields. The field content is saved with the scene when the scene is saved in Viz. Data is a text parameter. All commands/ assignments should be semicolon separated.
  For example: `A=5; B=3; aa[0..2]=1,3,5; Stocks[1]={Name=TEST, Value=4.5};`
- **Data1:** Sends information from external applications to the DataPool mechanism. The *Data1* field has a larger size and can accept larger information blocks than the Data parameter. This parameter is hidden. The data that is sent is not saved with the scene.
- **DataRequest:** Requests data from the DataPool (*DataRequest* is a text parameter). The user or external software can query DataFields by name using this parameter. The format of the query is as follows:
  `fieldname[indexes-range]; fieldname[indexes-range]....`
  The result of the query appears in the *DataOutput* parameter.
- **DataOutput:** Contains the data returned as a result of a query entered in *DataRequest* (*DataOutput* is a string parameter). The user or external software can use the returned information for manipulating the graphics. For example: External usage of *DataRequest* and *DataOutput*.

```
Enter data to the vectors VALUE and AAA:
0 RENDERER*FUNCTION*DataPool*Data SET VALUE[0-3]=V0, V1, V2,V3;
AAA[0-5]=A0, A1, A2, A3, A4, A5;

Request the last two items of VALUE and the third value of AAA:
0 RENDERER*FUNCTION*DataPool*DataRequest SET VALUE[2-3]; AAA[3];

Read the requested information:
0 RENDERER*FUNCTION*DataPool*DataOutput GET

Viz replies:
0 VALUE[2-3]=A2, A3; AAA[3]=A3;
```

- **DataCopy:** Copies data from one DataPool variable or structure to another. For example:
  - a=b; is assigned the value of b to a.
    Or, if Stock1 and Stock2 is of Stocks type then:
  - Stock2=Stock1; assigns all the values from Stock1 to the values of Stock2.

Copyright © 2026 Vizrt

⚠ **Note:** The copy action does not take place unless the Copy Automatically parameter is set `On`, or the **Execute Copy** button is pressed.

- **DataLink:** Links one DataPool variable or structure to another. For example:
  - a=b; links the value of a to b.
    or, if Stock1 and Stock2 is of Stocks type then:
  - Stock2->Stock1; links Stock2 to Stock1.

⚠ **Note:** The link action does not take place unless the Link Automatically parameter is set `On`, or the **Execute Link** button is pressed.

- **Dump:** Triggers a printout of all DataPool variables, structures and their values to the Viz console and to the file *datapool_dump.txt* located in *%ProgramData%\Vizrt\VizEngine*.
- **Dump Spec:** Triggers a printout of all DataPool variables and structures to the Viz console.
- **Initialize:** Triggers a rebuild of the DataPool hierarchy in the scene and the assignment of values to the DataPool variables.
- **Execute Copy:** Triggers an immediate copy of the variables or structures defined in the *DataCopy* field.
- **Execute Link:** Triggers an immediate link of the variables or structures defined in the *DataLink* field.
- **Reload Configuration Files:** Allows reloading all the **.dp* files.
- **Reload Conversion Tables:** Allows reloading all the conversion tables.
- **Copy Automatically:** Defines if the copy of the values, defined in the *DataCopy* field, is triggered automatically when the *DataCopy* parameter is changed. When set `OFF` the copy operation does not happen until the **Execute Copy** button is pressed. When set to `On`, every time the *DataCopy* parameter changes or any of the copied DataPool variables defined in the *DataCopy* field is modified, DataPool copies the values as defined in the *DataCopy* parameter.
- **Link Automatically:** Defines if the link of the values, defined in the *DataLink* field, is triggered automatically when the *DataLink* parameter is changed. When set to `Off` the copy operation does not happen until the **Execute Link** button is pressed. When set to `On`, every time the *DataLink* parameter changes or the linked DataPool variables are modified, DataPool executes the link as defined in the *DataLink* parameter.
- **Update DataPool On Load:** Loads and initializes the data stored in the Data parameter when set to `On`.
- **Enable UDP:** Enables external applications to control the DataPool via UDP when set to `On`. When set to `On` additional parameters are enabled.
- **Show UDP Messages:** Defines whether to display incoming UDP messages in the Viz console.
- **UDP Host Name:** Defines the name of the machine that the UDP messages should arrive from.
- **UDP Port:** Defines the port number that DataPool listens for UDP messages.
- **Enable Multicast:** Enables external applications to control the DataPool via MULTICAST when set to `On`. When set to `On` additional parameters are enabled.
- **Show Multicast Messages:** Defines whether to display incoming MULTICAST messages in the Viz console.
- **Multicast Port:** Defines the port number that DataPool listens for MULTICAST messages.
- **Multicast Address:** Defines the address of the machine sending the MULTICAST messages.

Copyright © 2026 Vizrt

- **Track DP_COMMAND in Scene SHM:** Tracks a reserved shared memory key `DP_COMMAND` in the scene shared memory when set to `On`. When the key is changed its content is parsed and interpreted as one of the following commands:
    a. If `DP_COPY` prefix is used, the remaining string is used as a DataPool copy command, copying one or multiple DataPool fields.
    b. If `DP_LINK` prefix is used, the remaining string is used as a DataPool link command, linking one or multiple DataPool fields.
    c. If `DP_SET` prefix is used, the remaining string is used to set the value of DataPool fields.
    d. When none of the above options are detected, the content of `DP_COMMAND` is used to set the value of DataPool fields.
- **Track DP_COMMAND in Global SHM:** Tracks a reserved shared memory key `DP_COMMAND` in the scene shared memory when set to `On`. The same as the above section, regarding Viz 3 Global shared memory.
- **Track DP_COMMAND in Distributed SHM:** Tracks a reserved shared memory key `DP_COMMAND` in the scene shared memory when set to `On`. The same as the above section, regarding Viz 3 Distributed shared memory.

Copyright © 2026 Vizrt

5 Viz World Plug-ins



The World Plug-ins provide details about settings available through its configuration user interface within Viz Artist.

Note: This plug-in package is released separately from the Viz World Client. Some of the functionality of these plug-ins are still intended to be used with the Viz World Client.

5.1 Related Documents

- *Viz Artist User Guide*: Contains information on how to install Viz Engine and create graphics scenes in Viz Artist.
- *Viz World Classic User Guide*: Contains information on creating 2D maps and geographic animations.
- *Viz World User Guide*: Contains information on creating real-time 3D maps and using the client-server solution.

5.2 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

5.3 Maps Geometry Plug-ins

This chapter describes geometry plug-ins. The geometry plug-ins are found in the following plug-in folders:

- **Maps:** Contains standard plug-ins.
- **Maps-Adv:** Contains advanced plug-ins.
- **Maps-Lab:** Contains experimental plug-ins. Since these plug-ins are experimental and not supported, they are not documented here.
- **Maps-Obs:** Contains obsolete plug-ins, installed only for backward compatibility. These plug-ins should **not** be used when designing new scenes. Since these plug-ins are obsolete and not supported, they are not documented here.

See the following sections for more information:

- 2D Label
- 3D Border
- 3D Line
- 3D Line Control
- 3D Models
- 3D Region
- 3D Region Control
- 3D Roads
- Atlas
- C3D Terrain
- GeoChart
- GeoImage
- Globe
- Label and Go
- MapScale
- Pyramid Control
- ShadowAgent
- WindFlows

Copyright © 2026 Vizrt

5.3.1 2D Label



The 2D Label plug-in creates two dimensional labels on the map, based on labels received from the selected map or automatically generated labels generated by the Label Manager according to the map information received from Navigator. It must have a container with text content as a child to work properly.



Note: When adding 2D Label to a container, World Position and Alpha are added automatically to the same container. Label Manager has to be added manually to the scene when using 2D Label without Navigator.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

2D Label Properties

Label

The Label tab contains general geometric properties for label such as margins (geometry length or width around the bounding box of the text), outline width and colors.

- **Margin Options**
  - **Equal Spaced:** Uses equal spacing for the side margins and top/bottom margins.
  - **Separate:** Uses different margins for each side of the labels.
- **Width Margins, Left/Right Margins:** Sets the margin between the sides of the text and the label background edges.

Copyright © 2026 Vizrt

- **Height Margins, Top/Bottom Margins:** Sets the margin between the top and bottom of the text and the label background edges.
- **Outline Width:** Sets the width of the label's background outline.
- **Label Color:** Sets the color of the label's background fill.
- **Outline Color:** Sets the color of the label's background outline.
- **Offset X:** Defines an additional offset in Viz units in the X-axis where the Label is drawn.
- **Offset Y:** Defines an additional offset in Viz units in the Y-axis where the Label is drawn.

Note: The numeric values next to the color parameters are the alpha values of the label and the label outline.

Note: The color palette effects only the selected color parameter (label or outline), and it appears when one of the color parameter is selected (Viz 3).

Bevel

The Bevel tab contains options dealing with rounding of corners.

- **Bevel:** Defines the bevel value of the label's background corners.
- **Roundness:** Defines whether the background corners is rounded.
- **Detail:** Defines the quality of the label object (when a higher value is selected, more polygons are used to build the label object).
- **Lock Aspect:** Applies the bevel and roundness parameters to the corners of the label object equally when set to `On`. When set to `Off`, the bevel and roundness parameters are applied to the corners using the object's height and width ratio.

Caption

The Caption tab defines the position of the label.

- **Pivot:** Defines the label's object shape and location in relation to the label's geographic location:
  - **Pointer:** Includes a pointer going out of the label background in the label object and points at the label's geographical position. When selected, additional parameters are enabled: *Width* and *Connection Bevel*.
  - **Outline:** Gives the label object a rectangular shape and the outline of the shape is on the label's geographical position.
  - **Center:** Places the center of the label over the label's geographical position.
  - **Offset:** Places the label at the defined offset from the label's geographical position.
  - **Position:** Suggests a few possible locations to Auto Street Labels for the label (positions along the street) so that Label Manager can pick one of them.
- **Width Type:** Defines whether the (strap) width should change as the caption is moving away from the tip or stay fixed.
- **Width:** Defines the width of the pointer's base overlapping the label background. This parameter is only enabled when Pointer is selected.
- **Connection Bevel:** Defines whether the area connecting between the pointer and the label background is to be rounded or sharp. This parameter is only enabled when Pointer is selected.

Copyright © 2026 Vizrt

- **Rotate Labels:** Defines whether labels, created in Viz (3D labels), are rotated like the labels in World Map Editor or Map Editor Classic (WME). If set to `Off`, all labels display horizontally. If set to `On`, labels that were rotated in Map Editor are rotated in Viz.

Caption Source

The options **Default Position**, **WME Directions** and **Label Manager Presets** include the properties described below.

Default Position

Default Position is used when adding labels in WME when no direction is selected.

- **Direction (deg):** Sets the angle of the label in relation to its geographic position.
- **Distance (cm):** Sets the distance of the label from its geographical position.
- **WME Offsets Distance:** Enables the WME to offset the distance (default `On`). When disabled (`Off`) only the direction offsets from WME is used and the distance ignored.

WME Compass Rose

This allows you to set and fine tune offsets for every direction available in WME. When selected, the labels are placed as they were placed on the map in the Map Editor. When manually set inside WME they always take priority over the presets and Default Position settings.

- **Presets:** Corresponds to the available directions inside the WME.
- **Direction Offset (Degrees):** Defines values for fine-tuning the position of the label.
- **Distance (Viz units):** Defines the distance offset for fine-tuning the position of the label.

Label Manger Presets

When this is selected, Label Manager uses the defined presets to place the labels over the map. Label Manager optimizes the label position such that the labels do not overlap.

- **Number of Presets:** Defines the number of label position presets available to the user (one to four presets).
- **Current Preset:** Selects the preset number to be configured, using the Direction and Distance parameters. Each preset should be selected and the label position should be adjusted.
- **Direction (deg):** Sets the angle of the label in relation to its geographic position.
- **Distance (cm):** Sets the distance of the label from its geographical position.

Special

The Special tab defines special options for the pointer.

Copyright © 2026 Vizrt



- **Shape:** Defines the pointer shape:
  - **None:** Gives the pointer a sharp point shape.
  - **Circle:** Gives the pointer a circle at the tip of the pointer.
  - **Square:** Gives the pointer a circle at the tip of the pointer.
  - **Straight:** Gives the pointer straight lines at the tip of the pointer.
- **Size:** Defines the size of the shape at the tip of the pointer. The parameter is only enabled if the pointer is selected and a tip shape is set.
- **Navigator Overlay:** Defines how the label is displayed over the map. Available options are Disabled, Fixed, Scaling, Near Scale and Far Scale.
  - **Disabled:** Places the label on the map using its geographical referencing.
  - **Fixed:** Places the label by keeping its geographical referencing but using a different camera (either with dynamic image or with a front layer). The label size remains fixed.
  - **Scaling:** Places the label by keeping its geographical referencing but using a different camera (either with dynamic image or with a front layer). The Label scales trying to imitate the camera movement.
  - **Near Scale:** Defines the maximum size of the label on the screen (the final size of the label when zooming in).
  - **Far Scale:** Defines the minimum size of the label on the screen (the final size of the label when zooming out).
- **Static Map Scale:** Defines whether scaling of the label is performed over a static map (no Navigator plug-in used). When disabled (`Off`), no scaling is applied to the labels. When enabled (`On`), an additional parameter is enabled:
  - **Scale:** Sets the scaling factor of the labels when used over a static map (no Navigator plug-in).
- **Collision Mode:** Defines how the labels are placed when an overlap or collision between two labels occur.
  - **Tip Based:** Allows the pointers of overlapping labels can cross or touch, but no overlap of label bodies are allowed.
  - **BBox Based:** Calculates a bounding box around the entire label (label body and pointer). Overlap between a label's bounding box is not allowed.
- **Hide if Empty:** Hides the container of the label if the Label text is empty.
- **Enable Overlay:** Renders label geometry on different camera.

Fade

The Fade tab defines the fade effect parameters to be used with the duplicated labels. Fade has two fade modes: Stand Alone and Controlled.

Copyright © 2026 Vizrt

Note: The *Fade On Distance* parameter is only enabled if Navigator Overlay is set to `Scaling`. See the plug-in section Special.

- **Stand Alone:** Defines the label appearance manually with the additionally enabled parameters:
  - **Fade On Time:** Defines a label fade effect, beginning at a relative point to the defined hop duration. An additional parameter is enabled, *Time To Hop*, defining when the fade occurs.
  - **Fade On Distance:** Defines a label fade effect, beginning at a relative distance from the hop final location.
  - **Fade On Lat/Long:** Defines a label fade effect, beginning at a Longitude and Latitude offset from the hop final location. An additional parameter is enabled, *Lat/Long*, defining the offset from in degrees.
- **Controlled:** Sets label appearance automatically with the 2D Label plug-in. It can also be based on Label Manager settings and Navigator animation (hops).
  - **Step:** Controls when the label fades in and out in relation to an animation. In general, the fade can be based on the camera distance (For example: Capitals are in view when distance is below 1000 kilometers) or on timing in relation to the hop:
    - **Auto:** Fades in and out based on distance to hop when a label is of type point (added by the user). If the label is of type place/region it fades in and out based on the distance set in Label Manager plug-in. If the hop is not close enough for the label to show and the label was added by the user, it fades in based on hop timing and not distance.
    - **On Hop:** Links the fade to the hop timing.
    - **Point 1/Point 2:** These are reserved for labels where the distance is configured by Label Manager.
    - **Hop and Above:** Turns on at the hop and stays on thereafter.
  - **Selected Label Timing:** Sets the time in relation to the hop time if the label's fading is based on hop timing. It is disabled if Step is set to Point 1 or Point 2. Since the label appearance is automatically calculated, this timing offset is used in the calculation. Select one of the options:
    - **At End:** Places labels at the end of the animation.
    - **Close to End:** Places labels just before the end of the animation.
    - **Ahead:** Places labels before the end of the animation.
    - **Well Ahead:** Places labels well before the end of the animation.
  - **Label Priority:** Defines the priority of the currently edited label in relation to other labels when a conflict between label positions occurs. The highest priority is preferred when displaying the labels.
    - **Auto:** Sets the label priority using Label Manager.
    - **Normal, High:** Allows Label Manager to decide which label to show when there is a conflict between two normal (or high) priority labels.
    - **Always:** Displays the label even if it conflicts with another label (with any priority).

Copyright © 2026 Vizrt

5.3.2 3D Border



The 3D Border plug-in draws the borders with fixed/scaling width options.

Controlled mode refers to the 3D Border Control plug-in, turn on this option to allow 3D Border Control to take effect.

Note: The 3D Border Control plug-in has been deprecated.



This plug-in is used for applying graphic designs to the border data retrieved by 3D Map Setting. Each 3D Border is used to filter and define which borders are drawn.

Note: When adding 3D Border to a container, 3D Line Shader is added automatically to the same container. 3D Map Setting must be manually added when using 3D Border.

The Data tab is used for defining which borders are displayed by the plug-in. The Width tab is used for defining the border width and other graphical related attributes. The Outline tab is used for adding an outline to the borders.

Copyright © 2026 Vizrt

The Effect tab is used for defining an animation of the border. The Advanced tab is used for defining general parameters of the border.

> **Note:** This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

3D Border Properties

Data

- **Draw:** Sets the border type drawn by the plug-in. Available options are Country, Region and All.
  - **Country:** Draws country borders only. When Country is selected, additional options are made available: Country, Coastline and Both (see Select).
  - **Region:** Draws region borders only. When Region is selected, additional options are made available: Region, Sub Region and Both (see Select).
  - **All:** Draws all available borders retrieved by 3D Map Setting.
- **Select:** Displays the available Country or Region options when either of those two options is selected.
  - **Country:** Draws inland country borders only.
  - **Coastline:** Draws country coastline borders only.
  - **Both (Country & Coastline):** Draws both inland and country coastline borders.
  - **Region:** Draws region borders only.
  - **Sub Region:** Draws sub region borders only.
  - **Both (Region & Sub Region):** Draw both region and sub region borders.
- **Selection:** Defines whether the plug-in uses the selected regions in the map (received from CWMClient). If enabled, only the borders of the selected regions in the map are drawn. If disabled, all borders are drawn according to the plug-in settings (country, region, and so on).

Width

- **Fixed (Pixels)–** Uses a fixed width when drawing the line. This parameter causes the line to maintain the same width regardless of camera position/distance. Available parameters are Width and Fade Edge.
  - **Width (Pixels):** Sets the line width in pixels.
  - **Fade Edge (%):** Sets the percentage of softness added to the edges of the line. When set to `0%`, the line edges are sharp and when set to `100%`, the edges are soft.
- **Scaling:** Varies the line width according to the camera distance from the map when selected. Available parameters that allow the user to set the line attributes are Width, Minimum Width, Maximum Width, Minimum Draw Width and Fade Edge.
  - **Width (km):** Sets the line width in meters on the map. The closer the camera to the map, the wider the line drawn.
  - **Minimum Width (px):** Sets the minimum line width in pixels. If the calculated line width (according to the Width parameter) is smaller than the Minimum Width value, then the Minimum Width value is used.
  - **Maximum Width (px):** Sets the maximum line width in pixels. This value is used when the camera distance is small and the line width should have been larger than the Maximum Width value (in pixels).

Copyright © 2026 Vizrt

    - **Minimum Draw Width (px):** Sets the minimum line width in pixels. If the calculated line width (according to the Width parameter) is smaller than the Minimum Draw Width value, and larger than the Minimum Width parameter, then the line is not drawn.
    - **Fade Edge (%):** Sets the percentage of softness added to the edges of the line. When set to `0%`, the line edges are sharp and when set to `100%`, the edges are soft.
- **Fixed (Mm)–** Uses a fixed width, in Viz units, when drawing the line. This parameter causes the line to maintain the same width regardless of camera position/distance. Available parameters are Width and Fade Edge.
    - **Width:** Sets the line width in pixels.
    - **Fade Edge (%):** Sets the percentage of softness added to the edges of the line. When set to `0%`, the line edges are sharp and when set to `100%`, the edges are soft.
- **See Me Now:** Calculates (when width is set to scaling) the width needed for the line to be visible at a given distance.

Outline

- **None:** Does not draw an outline to the borders.
- **Master:** Acts as the master plug-in for outline behavior. Other 3D Border plug-ins can be set as clients of this plug-in. The same outline attributes are applied to borders drawn by the master plug-in and by all other slave plug-ins. When selected, the *Outline Width (%), Outline Fade (%)* and *Outline Color* parameters are also available.
- **Slave:** Draws the outline according to the defined outline in the master plug-in above it in the scene hierarchy. All other parameters are disabled, displaying the master's values.
- **Stand Alone:** Defines the outline parameters for the borders drawn by this 2D Label only. When selected, the *Outline Width (%), Outline Fade (%)* and *Outline Color* fields are also available.
- **Outline Width (%):** Sets the width of the outline, as a percentage of the border width, where `0%` is the border width.
- **Outline Fade (%):** Sets the percentage of softness applied to the outline edges.
- **Outline Color:** Sets the color of the outline and the alpha value of the outline.

> ⚠ **Note:** The hierarchy structure is important when using the master/slave outline configuration. The master plug-in should always reside as the first container in the group of 3D Border containers. Expert should be added to the map (above the 3D Roads containers) and Z-Buffer Draw should be set to `Off`.

Effect

The Effect tab is used for creating an animation on the line size. After the line object is created, it can be animated by setting keyframes of the length parameter.

- **Border Length:** Sets the line length, where the value `100` represents 100% of the line length.
- **Fade:** Defines the softness that is added to the line edge as the length animation advances. When the length is `100`, the end of the line is not affected by the Fade parameter.

Copyright © 2026 Vizrt

Advanced

The Advanced tab is used for defining general parameters for the 3D Border object.

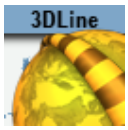
- **Height Offset:** Offsets the borders from the map (on the fly).
- **Border Quality:** Selects the quality of the border line:
  - **Controlled:** Calculates border quality by the navigator distance from the border object.
  - **High:** Uses high quality when drawing the border line (performance is slower).
  - **Medium:** Uses medium quality when drawing the border line.
  - **Low:** Uses low quality when drawing the border line. The border line looks pixelized when zooming into the map.
- **Add Level Down (Shape File):** Defines whether the sub region borders are drawn with the region borders, or if regions are drawn with countries.
- **Control Mode:** Defines whether the 3D Border object is externally controlled by 3D Border Manager. Border objects are controlled by groups.
- **Set Groups:** Set the group number for the object.
- **Line Shader:** Line shader has two variants. The first is **Simple** with less options (and better performance), while the second is more **Advanced** and allows for more options (but at the cost of performance).
- **Update Texture Mapping:** Determines whether the texture coordinates should be updated based on line width.
- **Low Angle Compensation:** Compensates for perspective distortion. Lines become too thin at low camera angles as a result of the perspective distortion. When that happens, there are not enough pixels to support a smooth, anti-aliased line, and the lines look jagged and aliased. This mode compensates for that by both widening the lines and applying transparency when they are viewed at too low angles.
- **Debug:** Enables debug messages in the console.
- **Min Angle:** Does not change line width and transparency if the angle between the camera and the ground below the line is lower than **Min Angle**.
- **Max Angle:** Increases the line width by the **Width Factor** and scales the transparency by the **Alpha Factor** if the angle between the camera and the ground below the line is higher than **Max Angle**. If the angle is between Min Angle and Max Angle then the width and transparency are interpolated.
- **Outline Angle Offset:** Applies an offset to the angle calculation for outline width in order to make the outlines affected at higher angles than the lines themselves.
- **Width Factor (%):** Determines the factor for modifying line width when applicable.
- **Alpha Factor (%):** Determines the factor for modifying line alpha when applicable.
- **Wireframe:** Shows the border object as a wireframe.
- **Print Sub Area Data:** Prints to the console all the sub regions in the plug-in (used for elections).

Rebuild

The Rebuild button triggers the plug-in to redraw the borders according to the plug-in parameters. Some parameters are updated as the parameters are changed and do not require a rebuild command, but it is good practice to rebuild the borders after setting the parameters.

Copyright © 2026 Vizrt

5.3.3 3D Line



The 3D Line plug-in has several graphic uses for drawing lines:

- Drawing shape lines created in the Map Editor (WME) when selecting a map. A line design is created in the hierarchy and defined in CWMClient (see 3D Objects and Shapes).
- Drawing a line between label locations, defined in the WME, from the first label in the list to the last label in the list. Labels in CWMClient must be enabled when using this mode.
- Drawing a line along hop points defined in a Navigator scene. The line follows the path of the Navigator animation between the hops.
- Drawing a line using a Long/Lat coordinates list. The line is drawn from the first Long/Lat pair to the last.

Note: When adding 3D Line to a container, 3D Line Shader is added automatically to the same container. 3D Map Setting has to be added manually when using 3D Line with Navigator. 3D Line Manager is used for controlling and creating 3D Line objects.

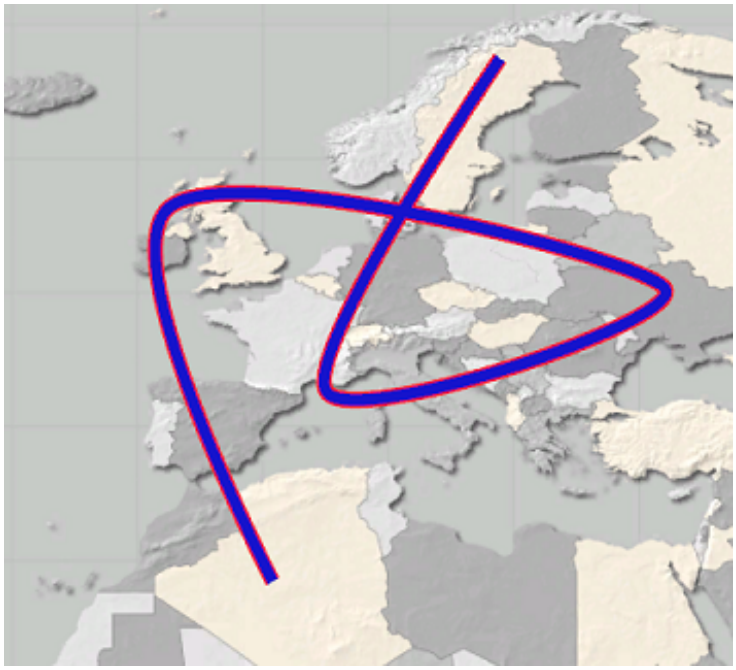
Note: Some of the uses of 3D Line as described above requires the use of Label Manager in the scene.

This plug-in also works together with plug-ins such as: Trace It.

This section contains information on the following topics:

- 3D Line Properties
  - Width
  - Outline
  - Effect
  - Control
  - Advanced
  - Rebuild

Copyright © 2026 Vizrt



Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

3D Line Properties

Width

Note: Width parameters, Fixed (px) and Scaling, affects the line object only if the scene uses Navigator. In case of a static map, use the Fixed (mm) to set the line width.

- **Fixed (px):** Uses a fixed width when drawing the line. This parameter causes the line to maintain the same width regardless of camera position/distance. Available parameters are Width and Fade Edge.
  - **Width (px):** Sets the line width in pixels.
  - **Fade Edge (%):** Sets the percentage of softness added to the edges of the line. When set to `0%`, the line edges are sharp. When set to `100%`, the edges are soft.
- **Scaling:** Varies line width according to the camera distance from the map when selected. Available parameters that allow the user to set the line attributes are Width, Minimum Width, Maximum Width, Minimum Draw Width and Fade Edge.
  - **Width (m):** Sets the line width in meters on the map. The closer the camera to the map, the wider the line drawn.
  - **Minimum Width (px):** Sets the minimum line width in pixels. If the calculated line width (according to the Width parameter) is smaller than the Minimum Width value, then the Minimum Width value is used.

Copyright © 2026 Vizrt

    - **Maximum Width (px):** Sets the maximum line width in pixels. This value is used when the camera distance is small and the line width should have been larger than the Maximum Width value (in pixels).
    - **Minimum Draw Width (px):** Sets the minimum line width in pixels. If the calculated line width (according to the Width parameter) is smaller than the Minimum Draw Width value, and larger than the Minimum Width parameter, then the line is not drawn.
    - **Fade Edge (%):** Sets the percentage of softness added to the edges of the line. When set to `0%`, the line edges are sharp. When set to `100%`, the edges are soft.
- **Fixed (mm)–** Uses a fixed width, in viz units, when drawing the line. This parameter causes the line to maintain the same width regardless of camera position/distance. Available parameters are Width and Fade Edge.
    - **Width:** Sets the line width in pixels.
    - **Fade Edge (%):** Sets the percentage of softness added to the edges of the line. When set to `0%`, the line edges are sharp. When set to `100%`, the edges are soft.
- **See Me Now:** Calculates (when width is set to scaling) the width needed for the line to be visible at a given distance.

Outline

- **Outline:** Enables (`On`) or disables (`Off`) outline.
    - **Outline Width (%):** Sets the width of the outline, as a percentage of the border width.
    - **Outline Fade (%):** Sets the percentage of softness applied to the outline edges.
    - **Outline Color:** Sets the color of the outline and the alpha value of the outline.

> ⚠ **Note:** The color palette is visible in all tabs of the editor, but it only affects the outline color.

Effect

The Effect tab is used for creating an animation on the line size. After the line object is created it can be animated by setting keyframes on the length parameter.

- **Length:** Sets line length, where `100` is 100% of the line length.
- **Fade:** Defines the softness added to the line edge as the length animation advances. When length is `100`, the end of the line is not affected by the Fade parameter.
- **Animation Length (frames):** Sets the default length of the reveal animation in frames (relates to the Run Animation option) and allows you to run a simple reveal animation by pressing the **Run Animation** button (used by 3D Map Telestrator).
- **Lock To Navigator:** Animates the line with the Navigator animation when using 3D Line with Navigator. The starting point of the line is the first hop location and the ending point of the line is the last hop position. The line animates as the hop animations are playing.
- **Run Animation button:** Plays the animation of that particular line.

Control

The Control tab is used for defining external control parameters. External control is done by 3D Line Control.

Copyright © 2026 Vizrt

- **Control Mode:** Defines whether the object is externally controlled or not.
- **Set Groups:** Defines the groups that the current 3D Line object is a member of. The same group names should be used in the 3D Line Control.
- **Shared Memory:** Determines the type of shared memory that can be used to control the lines.
- **Identifier:** Determines the shared memory name and general purpose identifier for this line. Can be used to control the line or to share a display list data between scenes.

Advanced

- **Cap Edges:** Sets the line's cap shape.
  - **None:** Does not change the line edges.
  - **Round:** Adds a filled round shape to the line caps.
  - **Square:** Adds a filled square shape to the line caps.
  - **Feather:** Adds a feather shape to the line caps. An additional parameter is enabled when Feather is selected: **Cap Fade Size**.
- **Length Effect:** Determines whether the separate polygons in the line animate at the same time (Per Polygon) or one after another (Continuous).
- **Wireframe:** Draws the 3D Line object as a wireframe when enabled (`On`).
- **Height Offset:** Sets the height offset for the 3D Line object on the map (on the fly).
- **Line Offset:** Sets the line offset for the 3D Line object on the map (on the fly).
- **Enable World Periodicity:** Determines how line objects that cross the date line act. When enabled (`On`), a line object that crosses the date line continues from the other side of the map. When disabled (`Off`), it continues across the date line.
- **Update Texture Mapping:** Determines if the texture coordinates are updated based on line width. You have the following options: Disabled, Always or Stationary. Stationary only updates the texture mapping when navigator is moving.

> ⚠ **Note:** When enabled, symbols are easier to control, but as line width changes the cause texture changes which might be disturbing.

- **Low Angle Compensation:** Compensates for perspective distortion. Lines become too thin at low camera angles as a result of the perspective distortion. When that happens, there are not enough pixels to support a smooth, anti-aliased line, and the lines look jagged and aliased. This mode compensates for that by both widening the lines and applying transparency when they are viewed at too low angles. When set to **Advanced**, the following additional parameters are available:
  - **Debug:** Enables debug messages in the console.
  - **Min Angle:** Does not change line width and transparency if the angle between the camera and the ground below the line is lower than **Min Angle**.
  - **Max Angle:** Increases the line width by the **Width Factor** and scales the transparency by the **Alpha Factor** if the angle between the camera and the ground below the line is higher than **Max Angle**. If the angle is between Min Angle and Max Angle then the width and transparency are interpolated.
  - **Outline Angle Offset:** Applies an offset to the angle calculation for outline width to make the outlines affected at higher angles than the lines themselves.
  - **Width Factor (%):** Determines the factor for modifying line width when applicable.
  - **Alpha Factor (%):** Determines the factor for modifying line alpha when applicable.

Copyright © 2026 Vizrt

Rebuild

The Rebuild button triggers the plug-in to redraw the lines according to the plug-in parameters. Some parameters are updated as the parameters are changed and do not require a rebuild command, but it is good practice to rebuild the lines after setting the parameters.

Copyright © 2026 Vizrt

5.3.4 3D Line Control



The 3D Line Control plug-in controls 3D Line object groups.

The Groups are then controlled in terms of width and color. The groups are derived from the settings in the 3D Line objects.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps_Adv

3D Line Control Properties

Control

- **Container:** Sets the top container above the controlled 3D Line objects.
- **Group:** Defines the controlled group number. A group is defined in every 3D Line object.

Width

- **Width:** Selects options to control the line width, Fixed (in pixels or in mm), or Scaling.
  - **Width:** Sets the width value when fixed width is selected (pixels or mm). Fixed width the line maintains the width value through all camera zoom range. When Scaling is selected, width is set in actual kilometers and the line width is calculated according to the camera zoom.
  - **Minimum Width (px):** Sets the minimum value of line width in pixels. this value is used if the calculated width according to the width parameter is smaller than the minimum width value (when zooming far out).
  - **Maximum Width (px):** Sets the maximum value of line width in pixels. this value is used if the calculated width according to the width parameter is larger than the maximum width value (when zooming in).
  - **Minimum Draw Width (px):** Defines the minimum line width drawn.
  - **Fade Edge (%):** Defines the edge fade width as a percentage of the line width.

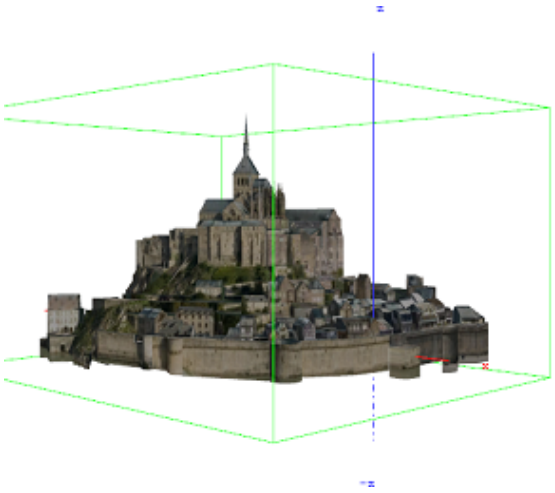
Copyright © 2026 Vizrt

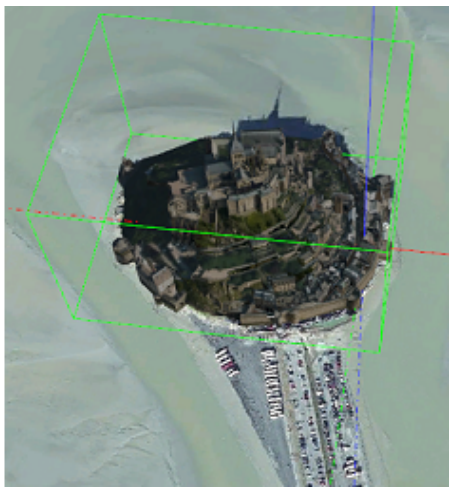
5.3.5 3D Models



The 3D Models plug-in adds 3D models to a graphics scene.

As the models also contain geographical referencing, you may also place the model on a map.





The images above show a 3D model of Mont Saint-Michel in Normandy, France. The model was downloaded from http://sketchup.google.com/3dwarehouse and then placed on a satellite image from Digital Globe.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

3D Models Properties

General

- **File:** Selects the file to load. Capable of loading files in Collada format with the **.dae* extension or **.kml*/ **.kmz* extension (which has geographic information for the model and a link to a **.dae* file inside).
- **Fill Mode:** Selects the rendering mode; solid, wireframe or dotted.
- **One Size:** Recalls the loaded model to a fixed size. Useful in non-georeferenced mode.
- **Automatic Fetch:** Enables fetching pieces of the city as you move around the map (Interactive) or animate (Streaming). Used when loading 3D Cities rather than single models.

Advanced

The Advanced tab is more suitable for a streaming mode of the plug-in, when the plug-in loads/streams data out of a large data bank which cannot be loaded entirely due to its size.

- **GPU Cache Size:** Limits the amount of data that is loaded to the GPU (loaded data may consume more size than expected on GPU).

Copyright © 2026 Vizrt

- **Model Index:** Enables to run through loaded models (mostly for debug purposes) when multiple models are loaded. When a specific index is set, the model is in focus, so the bounding box is shown and **Fly To** and **Info** buttons can be used on that model.
- **Level of Detail:** Defines how many of the largest levels of the texture pyramid (*.dxt* format) can be neglected during loading. May be useful in cases where the texture data is heavy and presents a higher resolution than that required to be shown.
- **Height Offset:** Height offset during fetch of the models.
- **Show Coverage:** Shows the total available geo coverage in green color and currently loaded coverage in orange when models are loaded/streamed from the large database.
- **Fetch Method:** Attempts to load as much data around the fetch point as possible (the amount is limited by the **GPU Cache Size** parameter) when models are fetched either by pressing the **Fetch Locally** button or in streaming mode.
    - **Navigator:** Defines the fetch point as the current Navigator position.
    - **Manual:** Enables manual definition of the fetch point.
- **Loading Type:** Determines how to load multiple models. The Serial mode may be preferred for recording (using Viz Post), when you have to be sure that all the models were loaded up until the next frame.
- **Fade Time (sec):** Defines the time period during which a new model fades from transparent to opaque, to create a more pleasant looking effect on load.

Consolidation

The Consolidation tab is used for compiling a binary format of the plug-in out of Collada files.

- **Model Optimization:** Defines the level of optimization for the model while the model is loading. More optimization may increase loading time while improving rendering performance in some models.
- **Downscale Textures:** Defines how much the first levels of a texture pyramid may be skipped.

Buttons

- **Fly To:** Used to take the camera to the model, if it is georeferenced (if the original Collada file had **.kml* or **.kmz* files containing the geolocation). Requires the presence of NavFinder on the container in order to work.
- **Reload:** Reloads the model.
- **Info:** Prints information about the model to the console. It prints information about the dataset in the case when more than one model is loaded, and prints information regarding a specific model if only one model is loaded or when the *Model index* parameter is not zero.
- **Fetch Locally:** Fetches models around the fetch point (defined manually with *Fetch Method* parameter). The number of models that are fetched is limited either by dataset or by the *GPU Cache Size* parameter.

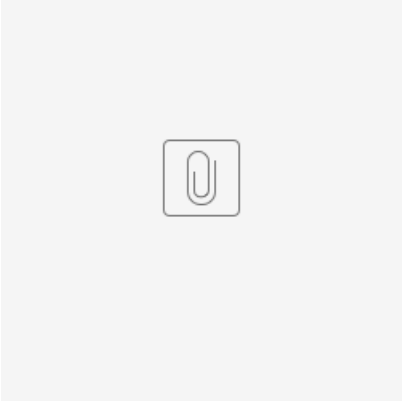
Working with 3D Models

This section provides a quick introduction on how to add a 3D model to your scene, and also how to place the model on a map or satellite imagery.

> **Tip:** Download models for testing from http://sketchup.google.com/3dwarehouse.

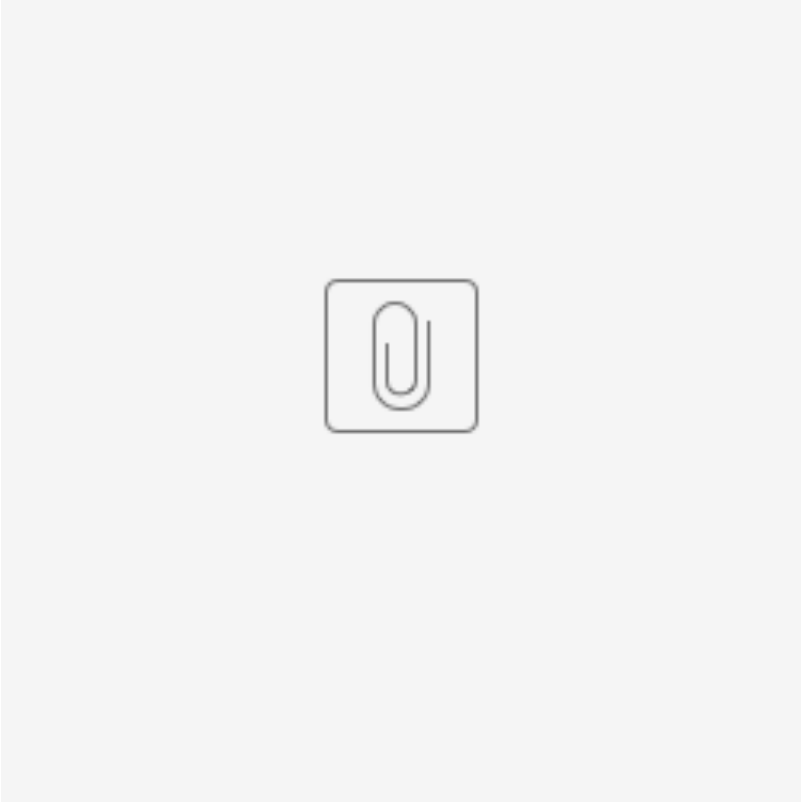
Copyright © 2026 Vizrt

To Add 3D Models to a Scene



1. Start Viz Artist.
2. Add 3D Models to an empty scene tree.
3. Open the 3D Models editor and load a KMZ file.

To Add 3D Models to a Map



1. Start Viz Artist.

Copyright © 2026 Vizrt

2. Add Navigator to an empty scene tree.
3. Add Atlas to the **GeoReferenceMap** container.
4. Open the Atlas editor and set it to use **Mercator projection.**
5. Add a **new group** as a sub-container of the **first** container and name it **position.**
6. Add World Position to the position group.
7. Add a **new group** as a sub-container of the position group and name it **model.**
8. Add 3D Models, Expert, and NavFinder to the model group.
9. Open the 3D Models editor and load a **KMZ** file (for example, Tower Bridge in London).
10. Click **Go To Model.** This should bring the model into view.
11. Open the Navigator editor, select **Advanced**, and then enable (`On`) the **Pan and Tilt Animation** option.
12. Open the 3D Models editor again and click **Go To Model.**
13. Open the transformation editor for the model container.
14. Open the **NavFinder editor** in order to adjust the distance, pan and tilt of the map.

Tip: Press **SHIFT + CTRL** while adjusting the **Distance**, and **SHIFT** while adjusting the **Pan** and **Tilt**.

See Also

- Expert

Copyright © 2026 Vizrt

5.3.6 3D Region



The 3D Region plug-in applies a graphic design to selected regions of the map.

The designs are defined in the scene and linked in CWMClient to create the regions received from the maps server.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps



The plug-in has the following four editor views:

- **General:** Quality and performance parameters.
- **Region:** The object's graphic properties.
- **Sub Regions:** The object's sub regions graphic properties. This tab is enabled when the Sub Regions parameter, under the Advanced tab's Data settings, is enabled (`On`).
- **Advanced:** Defines plug-in behavior in special operation scenarios.

3D Region Properties

General

- **LOD mode:** Sets the level of detail (LOD). When set to auto, the plug-in manages the level of details according to the camera distance from the region object. When set to manual, the level of details do not change when the camera moves, and remain fixed. When set to Manual, the *Polygon Quality* parameter is enabled.
- **Polygon Quality:** Sets the quality of the generated region object. The **Highest** setting uses more polygons to build the region object, but affects performance when many regions are selected. The **Lowest** setting

Copyright © 2026 Vizrt

enables better performance when adding a lot of regions, but looks poor when moving the camera closer to the region objects.

- **Geo Reference:** References the region object to the map above 3DRegion container when set to `On`. When set to `Off`, the longitude and latitude numbers are transferred to the Viz Artist position X, Y values. The object can still be georeferenced (for example, using Locator Control or World Position).
- **Auto Rescale:** Scales all regions to the same size. Can be used when the region is not geographically referenced (on screen normal) and you want all regions to be the same size (e.g. Israel uses the same screen real estate as a map of America) in Viz units.
- **Copy Map to Region:** Copies overlay the map texture on top of the region geometry.
- **Center X Y:** Defines where the axis of the region object is placed: if *Region* is selected, the axis center is the middle of the region object. If *Map* is selected, the axis is placed in the middle of the reference map.
- **Sensible Culling:** Defines whether to cut off areas that belong to the selected region but are geographically remote. If set to `Off`, all marked regions are built. Use this feature to improve performance (usually set to `On`).
- **Culling Threshold:** Defines the level of details displayed in the regions. A higher threshold value draws fewer details (that is small islands, small regions, and so on).
- **Use Cache:** Saves the objects to a cache folder when set to `On`. When using cache, the generated objects and parameters are saved in a cache folder and when switching between parameters the regions are not rebuilt, but taken from the cache (if they exist). If no cache (`Off`) is used, every change in the plug-in interface triggers a rebuild of the region objects.

Region

Fill

- **Fill:** Adds a region object to the map, and if fill is applied, how regions are built on the map (either flat on the map (no width) or extruded).
  - **None:** Does not add a region object to the map.
  - **Extrude:** Adds and extrudes regions over the map. When extrude is used, additional parameters are enabled.
  - **Flat:** Adds a flat object (Extrude= `0`).
- **Include Bottom:** Builds the bottom of the region when extruded.
- **Center Z:** Sets the Z axis location of the region object, with relation to the map: bottom, center or top.
- **Type:** Defines how the Height parameter is calculated.
  - **Absolute Height:** Defines the regions extrusion height value as Viz units.
  - **Relative Height:** Defines the regions extrusion height value as a percentage of the regions size.
- **Show Terrain:** Enables 3D Region to take into account any terrain data for a selected region. Adjust the Terrain Height Scale parameter. For this to work you need to enable the **Fetch Terrain Data** option in CWMClient.
  - **Terrain Height Scale:** Adjusts the terrain height.

Copyright © 2026 Vizrt

Border

The border parameters define whether a border is drawn around the region objects, border color, width and the alpha. If the 3D Region object is in Controlled Mode (*On*) the border properties parameters are disabled.

- **Borders:** Defines whether the object is drawn with a border. When enabled (*On*), the border is drawn and the border properties parameters are enabled.
- **Line Width:** Sets the number of GL lines for the border width.
- **Line Color:** Uses the color pallet to set the border color.
- **Alpha:** Sets the alpha value for the border. This value is controlled by the number to the right of the Line Color selection.

> ⚠ **Note:** The color palette is visible in all 3D Region editors but it **only** affects the border color.

Sub Regions

The Sub Regions tab has the same fields as the Region tab, except that Sub Regions does not have the Show Terrain and Terrain Height Scale fields. It also has the following field:

- **Sub Regions:** Defines whether the object uses the sub regions data in the drawn object. When enabled (*On*), the sub regions data are reflected in the geometry and the Sub Regions tab displays the sub regions parameters.

Advanced

Common Parameters

- **Freeze:** Enables the designer to freeze the region object to Viz data folders (the region is saved as an image containing the region's information).
- **Wireframe:** Draws a wireframe of the object.

> ⚠ **Note:** Multiple parameters in Election and Alpha tabs are unavailable if Sub Regions are not enabled.

Data

The **Data** tab defines how data is processed.

- **Controlled Mode:** Defines whether the 3D Region object is controlled by 3D Region Control in the hierarchy.

Election

All options work together with 3D Region Control.

- **Command:** Contains the text with region information used for elections.
- **Selection Data:** Since the region data can be shared between scenes, setting the Selection Data to **Local** means you only influence the local scene whereas with **Global** you influence all scenes.

If Sub Regions is enabled (`On`), additional parameters are enabled.

Copyright © 2026 Vizrt

- **Print Sub Region Data:** Prints the data in the Viz console.
- **Save Sub Regions Strings:** Saves back the sub regions data to the file.
- **Save Sub Region:** Saves the current specific sub region information back to the file.
- **Convert Names:** Converts the names in the plug-in to a different set of names.
- **Rebuild:** Rebuilds the region objects from the file.

Alpha

The **Alpha** tab is an advanced tab used mainly for elections. Alpha tab parameters are enabled when the Sub Regions parameter is enabled (*On*). The parameters affect the object only when Alpha is added to the object and 3D Region is in Controlled Mode. These parameters define what part of the region data is affected by Alpha.

- **Draw:** Defines what part of the object is affected by the alpha settings.
  - **All:** Draws both region and sub regions in the object.
  - **Region:** Draws only the region.
  - **Sub Region:** Draws only the sub regions.
  - **None:** Sets region and sub regions to be transparent.
- **Region Alpha Mode:** Defines what part of the data is affected by Alpha.
  - **Region:** Affects only the region areas.
  - **Sub Regions:** Affects only sub region areas.
  - **None:** Does not affect data when using Alpha.
- **Border Alpha Mode:** Defines what part of the border data is affected by Alpha.
  - **Region:** Affects only the region borders.
  - **Sub Regions:** Affects only sub region borders.
  - **None:** Does not affect data when using Alpha.
- **Create Alpha Channel:** Creates an alpha channel animation automatically.
  - **Length:** Determines the length of alpha channel animation
- **Extrude Length (F):** Animates sub regions height (extrude). This parameter sets the time for the animation.

Draw

The **Draw** tab defines advanced parameters for drawing the region objects.

- **International Date Line:** Relates to countries that reside on the 180 and -180 longitude line when using flat maps.
  - **None:** Draws the regions on both sides of the map.
  - **United:** Draws the regions together.
- **Border Data:** Defines the source data of the borders.
  - **Polygon:** Draws borders using polygon data.
  - **Border:** Draws borders using border data (higher memory consumption).
- **Border Quality:** Sets the required border quality.
- **Morphing:** (Off, Start, End, Start & End)
  - **Start:** Allows multiple start shapes to be used to create multiple region morphing shapes. You must also design a separate region using **End** in order to define how the shape should end.
  - **Start & End:** Allows setting two shapes (start and end), and the region morphs between the two shapes.
- **Morphing Samples:** Determines the number of data samples between the start and end animation.
- **Morphing Value (%):** Used to animate between the first and last frame of the shape.

Copyright © 2026 Vizrt

Note: The color palette is visible in all 3D Region editors but it **only** affects the border color.

Copyright © 2026 Vizrt

5.3.7 3D Region Control



The 3D Region Control plug-in controls one or more 3D Region objects by changing and applying graphic properties to the objects.

3D Region Control is typically used for producing election graphics where it is useful to distinguish regions by, for example, color.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps_Adv

3D Region Control Properties

- **3D Regions Parent Container:** Container placeholder for the container holding the region objects.
- **Control Mode:** Defines the data segment of the object to be controlled. Select *Region* or *Sub Region* and then select what part of the region/sub-region to control:
  - **Fill:** Enables control of the region/subregion's fill properties.
  - **Border:** Enables control of the region/subregion's border properties.
- **Fill Mode:** Defines the fill of the regions.
  - **Interpolate 3:** Sets three colors that define the region's color range. The region's color is derived from the range of colors and the number of regions.
  - **Interpolate 2:** Sets two colors that define the region's color range. The region's color is derived from the range of colors and the number of regions.
  - **Group:** Controls the fill of all the 3D Region objects under the defined 3D Regions Container (apply the material added to the 3D Region Control container).
  - **Highlighted:** Controls the selected regions on the map only.
  - **Not Default:** Allows 3D Region Control to draw any sub region in the target 3D Region which is not in group one.
- **Apply Sender Matrix:** Applies a matrix to the regions. Available options are None, Multiply, Translate, Scale and Full.
  - **None:** Uses the 3D Region matrix.
  - **Multiply:** Uses the 3D Region matrix multiplied by the 3D Region Control matrix.
  - **Translate:** Uses only the translated part (x, y and z position) of the 3D Region Control matrix.
  - **Scale:** Uses the scale part (x, y and z scaling) of the 3D Region Control matrix.
  - **Full:** Uses the 3D Region Control matrix.
- **Apply Sender Matrix:** Applies a matrix to the regions. Available options are None, Multiply, Translate, Scale and Full.
- **Apply On:** Applies the sent matrix on all regions or only the selected sub regions.

Copyright © 2026 Vizrt

5.3.8 3D Roads



The 3D Roads plug-in overlays road data onto a map.

This plug-in is mainly used with a 3D Line Shader plug-in.



The plug-in has three different editor views:

- **Data:** Defines what type of roads are drawn by the selected 3D Roads object. Select one or more types of roads to be displayed, using the plug-in graphic properties.
- **Width:** Defines the width of the roads drawn by the plug-in.
- **Outline:** Defines the objects outline properties and behavior.
- **Advanced:** Defines extra settings on how to handle texture mapping and low angle compensation when drawing streets.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

3D Roads Properties

Data

- **Freeways/Motorways, Primary, Main, Secondary and Other Roads:** Draws roads rated according to the selection in the loaded roads data when enabled (`On`).
- **Major and Other Railways:** Draws railways rated according to the selection in the loaded railways data when enabled (`On`).

Width

Selects how the road width is calculated.

- **Fixed (Pixels):** When set to *Fixed* the road maintains a fixed width during camera zoom movements. Available options are Width (pixels) and Fade Edge (%).
  - **Width (Pixels):** Sets the width of the roads in number of pixels.
  - **Fade Edge (%):** Sets the percentage of softness applied to the road edges.
  - **Height Offset:** Offsets the borders from the map (on the fly).

Copyright © 2026 Vizrt

- **Scaling:** When set to Scaling the road width varies according to the camera zoom movements. When selected, the following parameters are available:
    - **Width (Meters):** Sets the physical road width (in Meters). This value is translated into drawn pixels according to the camera zoom position and the size of the map.
    - **Minimum Width (Pixels):** Sets the minimum size of the roads drawn. If the calculated road size in pixels is lower than the minimum width, the road is not drawn.
    - **Maximum Width (Pixels):** Sets the maximum size of the roads drawn. If the calculated road size in pixels is higher than the maximum width, the road width is set to the maximum width.
    - **Minimum Draw Width (Pixels):** Sets the minimum size of the roads drawn. When zooming into an area, this is the point where the roads begin to fade in and are drawn over the map.
    - **Fade Edge (%):** Sets the percentage of softness applied to the road edges.
    - **See Me Now:** Calculates (when width is set to scaling) the width needed for the line to be visible at a given distance.
- **Fixed (mm):** When set to *Fixed (mm)* the road maintains a fixed width during camera zoom movements. When selected, the *Width (mm)* and *Fade Edge (%)* parameters are available.

Outline

- **Outline:** Selects one of the options for adding outline to the roads. Available options are None, Master, Slave and Stand Alone.
    - **None:** Does not add an outline to the roads drawn by the plug-in.
    - **Master:** Controls other 3D Roads plug-ins which are in slave mode. The same outline attributes are applied to roads drawn by the master plug-in and by all its slave plug-ins. When selected, additional parameters are Outline Width (%), Outline Fade (%) and Outline Color.
    - **Slave:** Draws the outline according to the defined outline in the master plug-in above it in the scene hierarchy. All other parameters are disabled, displaying the master's values.
    - **Stand Alone:** Defines the outline parameters for the roads drawn by this 3D Roads plug-in only. When selected, available additional parameters are *Outline Width (%)*, *Outline Fade (%)* and *Outline Color*.

⚠ **Note:** The master setting ONLY controls other plug-ins, it does not draw any outlines itself. If there are no slave plug-ins, then no outlines are visible.

- **Outline Width (%):** Sets the width of the outline, as a percentage of the road width, where `0%` is the road width.
- **Outline Fade (%):** Sets the percentage of softness applied to the outline edges.
- **Outline Color:** Sets the color of the outline and the alpha value of the outline.

⚠ **Note:** The hierarchy structure is important when using the master/slave outline configuration. The master plug-in should always reside as the first container in the group of 3D Roads containers. Expert should be added to the map (above the 3D Roads containers) and Z-Buffer Draw should be set to `Off`.

⚠ **Note:** Road data should be loaded (using CWMClient) before working on roads design.

Copyright © 2026 Vizrt

Advanced

- **Line Shader:** Line shader has two variants. The first is **Simple** with fewer options (and better performance), while the second is more **Advanced** and allows for more options (but at the cost of performance).
- **Update Texture Mapping:** Defines when to update the mapping of textures applied in the container:
    - **Disabled:** Does not ever update the mapping of the texture.
    - **Always:** Updates the mapping of the texture while animating.
    - **Stationary:** Updates the mapping of the texture only when the scene is stationary or not an animation.
- **Low Angle Compensation:** Used when creating graphics that have a low angle point of view or tilt in order to remove aliasing on lines far away from the camera.
    - **Wireframe:** Displays the roads as wireframe.
    - **Debug:** Enables debug messages in the console.
    - **Min Angle:** If the angle between the camera and the ground below the line is lower than *Min Angle*, then line width and transparency are not changed.
    - **Max Angle:** If the angle between the camera and the ground below the line is higher than *Max Angle*, then the line width is increased by the *Width Factor* and the transparency is scaled by the *Alpha Factor*. If the angle is between Min Angle and Max Angle then the width and transparency are interpolated.
    - **Outline Angle Offset:** For outline width, apply an offset to the angle calculation in order to make the outlines affected at higher angles than the lines themselves.
    - **Width Factor (%):** Determines the factor for modifying line width when applicable. Percentage value for the width of the line far away from the camera. For example, 400% thicker than when it is close to the camera.
    - **Alpha Factor (%):** Determines the factor for modifying line alpha when applicable. Percentage value for the alpha of the line far away from the camera.

Copyright © 2026 Vizrt

5.3.9 Atlas



The Atlas plug-in displays satellite imagery from Microsoft Bing, Maxar (formerly Digital Globe), Google and other providers. This object plug-in is used with a navigator plug-in, enabling interactive navigation to any location on Earth. The plug-in can be used to display imagery for a Navigator animation (Hops), while creating the required image tiles for a smooth and complete camera movement. It can also be used with Globe.

To create an interactive Touchscreen Application, Atlas must be used together with Navigator.

IMPORTANT! To ensure that image tiles appear correctly, use a base map with the correct projection.

The plug-in has the following editor tabs:

- **Data:** Defines imagery parameters and data providers.
- **General:** Contains General settings.
- **Advanced:** Details configuring connections and operation parameters.
- **Communication:** Contains proxy and IP settings.
- **Animation:** Contains advanced options on how different animation slides are displayed.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

Atlas Properties

Pre-Fill Texture Cache Option: Fully initialize GPU Memory for tiles rendering. This ensures that no textures are created at playout. This increases texture memory usage, because it allocates textures for tiles that might never be rendered.

Data

The **Data** tab is used to define imagery parameters.

- **Data Source:** Sets the data provider. Available providers are:
  - Microsoft Bing
  - Vizrt's CMR (Curious Multi Resolution)
  - Maxar (Digital Globe)
  - WMS (Web Map Service) geo server
  - Nokia
  - Map Server
  - Google

Note:

- Bing, Maxar, WMS and Google require an internet connection.
- Bing, Google and Maxar require a license.
- My Radar, WDT and Blom are removed, existing Scenes fall-back to *Maxar*.

Copyright © 2026 Vizrt

- **Hide Attribution:** Hides the provider's attribution. Some data sources provide this possibility, but it is different for each data source. Some provide a separate license to enable this feature.

Bing

- **Legacy Protocol:** Allows use of the **Data Type** parameter for choosing either satellite imagery or road and border maps with labels, or a combination of the two. If you combine the two road and border maps with labels, they are imposed over the satellite imagery. The API used in previous versions of Viz World is used.
  - **Satellite:** Displays ONLY satellite imagery.



  - **Map:** Displays the map using the hybrid styling, also displaying labels.

Copyright © 2026 Vizrt



- **Combined:** Displays a mix of satellite imagery and labels.

Copyright © 2026 Vizrt



- **Traffic:** Draws a traffic layer.

- **New Protocol:** Allows selecting whether to include labels in the map, and what language to use for these labels. A new API is used, providing more control over what is displayed in the map. The following parameters are available:
  - **Map:** Displays satellite imagery, or displays the map using the hybrid styling.
  - **Labels:** Turns the labels in the map on/off.
  - **Culture:** Selects the language of the labels.
  - **Vector:** Determines whether the map includes vector data (streets/ borders) with the satellite imagery.
  - **Hills:** Determines whether the synthetic map includes hills topography.

CMR

- **CMR File:** Sets the path to the CMR file.
- **CMR Data Status:** Indicates whether the file was loaded correctly.

Maxar

Note: For Maxar datasource, the Viz World server needs to be running with a valid license, to provide a license key to the Atlas plug-in.
- If an MGP LICENSED license is available, the new MGP API for tiles is used.

Copyright © 2026 Vizrt

- If not, Atlas retrieves SecureWatch license credentials from Viz World and uses the legacy API for tiles.
- As a final fallback, Atlas attempts to get an MGP EVAL key from Viz World and uses the new MGP API (tiles are watermarked).

- **Projection:** Defines the projection of the map, either *Unprojected* or *Mercator*.
- **Digital Globe Profile:** Enables you to select from the most common profiles of Digital Globe maps
  Note: Imagery from different profiles may differ in some places and be the same in others.

In addition, the following parameters can be configured in the *VizWorld.ini* configuration file (by default located at *%ProgramData%\vizrt\VizEngine\Maps*):

- DigitalGlobePrefix
- FirstLookPrefix

If these parameters are used (present and not commented with hash) then Atlas uses these strings as the basic connection string.
Projection, tile coordinates and additional parameters like profile or header are added in Atlas itself.

WMS

- **Projection:** Defines the projection of the map, either *Unprojected* or *Mercator*.
- **URL:** Sets the URL of the WMS geo server.
- **Layer:** Comma-separated list of one or more map layers. Values in list correspond to the layer <Name> values by the geo server's Capabilities file. Layer list is optional if the Styled Layer Descriptor (SLD) parameter is present in the request.
- **Free text:** Sets **optional parameters** like transparency and background color (for example, `TRANSPARENT=TRUE&BGCOLOR=#E98300&`). For more information on setting optional parameters in the free text field, see the WMS specification documentation.
- **Version:** Request version. Valid values are `1.0.0`, `1.1.0`, or `1.1.1`.

Nokia

- **Data Type:** Sets how the map is displayed. For more information, see the description of the Data Type parameter in the Bing tab.

Map Server

This allows the use of the TPLs and stylesheets in an available map server.

- **Server Configuration:**
  - **Default:** Uses the default server configured in the engine (see the *Configuring Viz* section of the Viz Engine Administrator Guide).
  - **Manual:** Allows using a different server than what is configured in the Viz Engine.
- **Projects:** Lists all TPLs that are available in the map server.
- **Project:** Displays the selected TPL.
- **Pick Style:** Shows available style sheets in the selected TPL by numerical index .
- **Layer:** Shows the name of the selected style sheet.

Copyright © 2026 Vizrt

Note: In Viz Artist, a maximum of 256 TPLs can be displayed.

Google

- **Profile XML Status:** Shows if the file *GoogleMaps_Profiles.xml* (by default located at *%ProgramData%\vizrt\VizEngine\Maps*) was read correctly.
- **Google Profile:** Lists all found profiles from the file *GoogleMaps_Profiles.xml*.

Note: To add or modify Google profiles, edit the file *GoogleMaps_Profiles.xml*. The Google Maps Tile documentation lists all possible configurations.

General

- **Use Base Map:** Uses base map. This option is used when working with a base map above the Atlas container, and the base map is projected.
  Note that Microsoft Bing imagery is unprojected, so in order to have the images positioned correctly over the map, set this parameter to *On*. The same applies when using Globe.
- **Map Size:** Defines the size of the Atlas image.
- **Fade Time:** Defines the duration of the transition between tile resolutions when zooming in/out of the image.

Advanced

The **Advanced** tab is used for configuring connection and operation parameters.

- **Temp Folder:** Defines a specific cache folder for the retrieved imagery.
- **Mode:** Defines the mode of retrieving the tiles from the Microsoft Bing server. When set to *Serial*, and new image tiles are required, Atlas sends a request to the server and waits for a reply. Only after the reply is another tile requested, and so on. During the time that Atlas waits for a reply, the UI is locked and the user cannot perform other operations. When set to *Multithreaded*, images are requested by a thread of the plugin and the UI is not locked.
- **Requests per Frame:** Defines the maximum image requests from the server in a frame.
- **Layers:** Defines the number of images (with different resolution) per one area.
- **Texture Compression:** Selects one of the required compression modes for the images.
- **Texture Quality:**
  - **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 appears on the two other qualities when the object is being moved away along the Z axis or shrunk, by avoiding the observable "noise" when a texture constantly scales to try to fit.
- **Quality:** Defines the quality of the images requested from the server.
- **Memory Buffer Size:** Controls memory buffer used by Atlas. Useful for scenes with multiple Atlas plugins.

Copyright © 2026 Vizrt

- **Layer Type:** Maintains transparency while stacking multiple layers on top of each other when a transparent layer type is selected. Selecting *opaque* consequently blocks anything below it. WMS layers tend to be transparent (for example, power lines or flood area). This parameter is only visible when the Data Source is set to *WMS*.
- **Stop Support Margins:** Retrieves additional image tiles surrounding the displayed area to enable immediate display of high resolution images when the user moves the image when set to `On`.
- **Limit Disk Read Time (ms):** Specifies the maximum time period for Atlas to be occupied with loading. If while loading the tiles for the current frame, Atlas recognizes that loading is taking too much time, it stops loading and renders the frame.
- **Use Hysteresis:** Resolves the 'flickering problem' that may occur if the Navigator position moves back and forth around a point where Atlas decides to change resolution levels. Hysteresis uses history data in the calculation of a point where the level should be changed, such that the switch for zoom in and zoom out does not happen in the same point, avoiding the flickering.
  If enabled, set the **Hysteresis size**.
- **Tessellation:** Increases tessellation of each tile in Atlas. This may be useful when projection of the basemap differs from the projection of Atlas data source (and one of the projections is not linear). By using tessellation, the tile fits more accurately to the basemap (less distortion). Raising this parameter affects performance, so only use when distortion due to a difference of projections is visibly unacceptable. Only visible when Atlas is used under basemap.

Communication

- **Use Proxy:** Enables proxy configuration parameters when set to `On`. Use this option when working an a network with proxy:
  - **IP:** Determines proxy server IP number.
  - **Port:** Determines proxy port number.
  - **Username/Password:** Sets the username and password.
- **Bind to IP Address:** Forces Atlas to use the IP address of a specific network card in the computer (used when multiple network cards are part of the computer) when set. When not set, it defaults to the first network connection.

Animation

The animation properties are only valid if the Data Source is set either to *My Radar* or to *Weather Decision Technologies*. These properties offer advanced display options for various animation slides.

Copyright © 2026 Vizrt

5.3.10 C3D Terrain



The C3D Terrain plug-in displays terrain objects. The terrain is retrieved from the Viz World Server (WoS) when CWMClient is added to a C3D Terrain object.

When CWMClient is added to the C3D Terrain plug-in, terrain tessellation parameters are enabled in the Miscellaneous tab of CWMClient.

The plug-in has three views:

- Terrain
- Pyramid
- Advanced

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

C3D Terrain Properties

Terrain

- **Height Scale Mode:** Sets the height scaling option for the terrain:
  - **Manual:** Sets the height scale manually by changing the Height Scale parameter. When selected, additional parameters are enabled:
  - **Soft:** Sets the terrain surface to be softened by interpolating points over the terrain. This option prevents the sharp edges over the terrain surface.
  - **Normal:** Uses the same terrain elevation values received from the server.
  - **Exaggerate:** Applies a large scaling factor to the terrain height differences, exaggerating the terrain surface.
  - **Tiling:** Does not take the terrain height into account for the map size. Setting *Height Scale Mode* to `Absolute` may therefore be needed for the terrain tiling, or else the tiles might not connect properly (due to height differences).
- **Globe:** Draws the terrain as part of a globe and enables the Globe Radius parameter when set to `On`.
- **Globe Radius:** Sets the size of the globe used for drawing the terrain.

Note: The Globe and the Globe Radius parameters are disabled if Use Base Map is enabled (`On`).

- **Height Scale:** Sets the scaling value for the terrain elevation. Using a low value flattens the terrain surface.
- **Delta Z:** Sets the Z axis offset for the terrain (mainly used when using the base map).

Pyramid

A Pyramid is a set of map textures in different resolutions, used for zooming into a defined area. When the camera is far from the map, a low resolution map texture is used (covering a wide area). As the camera zooms into the map, it zooms into an area with higher resolution texture, until the final map, with the highest resolution, is used. To enable the Atlas Pyramid parameters, CWMClient must be attached to its container. Note that this disables the Atlas *Terrain File Name* parameter. See the Terrain editor view.

Copyright © 2026 Vizrt

- **Pyramid:** Defines whether a pyramid of map textures is created for the terrain area. When enabled (`On`) the Max Height and 3D Levels parameters are enabled.
  - **Max Height:** Defines the number of textures that are created.
  - **3D Levels:** Defines the number of terrain objects that are created.
- **Blend Textures:** Defines whether the edges of the maps are soft, blending into the larger map of the pyramid. When enabled (`On`), the edges are softened and the **Blend Amount** parameter is enabled.
  - **Blend Amount:** Defines the amount of softness added to the map edges.
- **Rebuild Pyramid:** Builds the pyramid levels.

> ⚠ **Note:** The parameter Rebuild Pyramid is enabled only if Pyramid is enabled (`On`) and it is visible in all tabs.

Advanced

- **Sampling Resolution:** Defines the number of points used to calculate the terrain. The higher the sampling resolution, the less detail shown. When Pyramid mode is enabled (`On`), this parameter is disabled.
- **Use Base Map:** Defines the geographical referencing of the terrain. If Use Base Map is enabled (`On`), the terrain moves to its geographic location on the base map, and the *Base Container* parameter is enabled.
  - **Base Container:** Uses the first map above the C3D Terrain container in the hierarchy as the base map if empty and **Use Base Map** is set to `On` . To use a specific map as the base map, drag a map container to the Base Container place holder. If Use Base Map is set to `Off` , the Globe parameter is enabled and the terrain is drawn as part of a globe. Set the *Globe Radius* to modify the terrain size and curve.
- **Wireframe:** Displays the terrain as wireframe.
- **Sides Smoothing:** Draws the edges of the terrain as flat lines (height is zero) when enabled (`On`). This option is useful when using the terrain object over the base map. When enabled (`On`), additional parameters are enabled:
  - **Sides Smoothing Factor:** Defines the width of the area, close to the edges, that is interpolated to create the smooth transition from terrain info to a flat edge.
  - **Uniform Smoothing:** Applies smoothing for all sides. Enabling Uniform smoothing (`On`), sets smoothing for all sides, and hides the individual parameters. Available individual parameters are; Left, Right, Top and Bottom smoothing.
- **Texture Compression:** Sets the compression level for the texture (DTX5 is the highest compression level; hence, less texture quality).
- **Terrain Status:** Displays the terrain object status. If the terrain object was rebuilt successfully, the indicator displays OK, otherwise, the indicator displays Not Built.

Buttons

- **Rebuild:** Forces a rebuild of the geometry, which is necessary when changing parameters that cannot be updated in real time.
- **Freeze Terrain:** Saves the terrain image and data as Viz images for faster loading and for archiving.

Copyright © 2026 Vizrt

5.3.11 GeoChart



The GeoChart plug-in represents geo-located values as 3D bars on a map. The data format can be a grid of geo positions (locations with constant distances from one another) or a list of coordinates.

In addition, data may be split by timeframes and the plug-in is able to animate the data chart throughout the frames.

To use GeoChart, it should be put beneath the geo reference and load data (see Supported Formats). GeoChart adds the Geo Chart Shader plug-in to the container automatically.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps_Adv

GeoChart Properties

General

- **Load Data:** Selects the data file to be loaded.
- **Frame Number:** Allows selecting the desired timeframe with this control when data comes in frames (see Supported Formats).
  Please note that although it is possible to animate data through the Frame Number parameter it is not recommended because when the frame is switched the data is read from the hard disk such that such an animation is more performance consuming. Moreover there is no interpolation between frames such that the animation jumps from timeframe to timeframe. Use Animation Timeline instead.
- **Animation Mode:** Analyzes the whole frame sequence and builds an animation if the data has timeframes. The data is interpolated smoothly between the frames such that the animation looks smooth.
- **Animation Timeline:** Animates data through timeframes when an animation is built.
- **Animation Buffer Size:** Defines the size of the memory buffer in which the created animation is stored. If the buffer size is less than that required to build the animation from all the available frames, then not all of the frames are included (You can check the total available frames by dragging the Frame Number parameter to the right and seeing the largest frame shown, and the frames included in the animation by dragging the Animation Timeline parameters to the far right).
- **Bar Type:** Defines the geometry of the bar. Please note that Cylinder geometry is the most performance consuming while Prism is the lightest.
- **Bevel Bars:** Bevels bars edges.
- **Bar Height:** Scales the height of the bars.
- **Bar Width:** Scales the width of the bars.
- **Bar Length:** Alters bar's length starting from the bottom.
- **Mask Source:** Clips data to a specific region such that only bars inside the region are seen. For this purpose a mask image is required. You can drag the image into the *Mask* rectangle in the *Image* option or specify a container with a mask image in the *Container* option. See Creating Mask Images.
- **Rebuild:** Reloads the parameter values after a change. Only three parameters of the GeoChart plug-in can be changed on the fly: *Bar Height*, *Bar Width* and *Bar Length*. Altering other parameters requires a Rebuild.

Copyright © 2026 Vizrt

Color

- **Logarithmic Color:** Applies logarithmic scale to a color transform. This may be used when the data has dramatic differences between low and high values (in such cases the high values are red by default and all the low values are blue). Logarithmic scale makes this difference less dramatic, such that more in-between colors appear on lower values. An example of such data may be world population density. In large cities this value is considerably higher than in its surroundings, making it difficult to see less dramatic differences in low population areas.
- **Set Input Range:** Constrains input range to a specified **Input Minimum** and **Input Maximum** value. All values below the minimum are interpreted as lowest possible bar height (zero by default) and all values higher than the maximum are interpreted as a maximum bar height. Note that when *Set Input Range* is pressed the actual minimum and maximum data values from the loaded data are seen.
- **Colors:** Defines the color map by choosing colors and a number of in-between color ramp stages.
- **Color Image:** Uses an image to define the color map. The image must be dragged to this control. An example of an image defining color map:



- **Posterization:** Enables a continuous color map to be split into a desired number of discreet colors.

Advanced

- **Bar Response:** Defines which property of the bar is affected by the value it represents.
- **Logarithmic Data:** Puts the values that the bars represent on the logarithmic scale. This gives a better indication of differences between low values, and makes the difference between low and high values less dramatic. May be useful in cases where there is a large difference between low and high values. (e.g. world population density). Consider using this option in combination with the *Logarithmic Color* option under the Color tab.
- **Data Simplification:** Resamples the data and create different resolution levels to show lower resolution from farther distance. Moreover, data is spit into tiles by a culling mechanism. The data file that is being loaded to a plug-in may contain a huge amount of data and can become very performance intensive during rendering. This parameter defines how may resolution levels are created out of the original data, and therefore effects loading time and the animation built process.
- **Fix To Resolution Level:** Uses a fixed resolution level, selected by the **Resolution Level** parameter. In some cases you would not like the data to be resampled, since this means a loss of some information. A zero value means that raw data is used. Raw data is not resampled and not split to tiles.
- **Invert Output:** Gives maximum values the lowest bar heights and vice versa.
- **Output Minimum:** Defines the height of the bar representing the minimum found value.
- **Output Maximum:** Defines the height of the bar representing the maximum found value.
- **Use Input Range:** Defines the input range such that values lower than **Input Minimum** are considered as having the Input Minimum value, and values higher than **Input Maximum** are considered as having the Input Maximum value. This may be useful when focusing on a specific range in values.
- **Clip Data:** Enables clipping the bars representing values lower or higher than defined in the **Clip Bottom** and **Clip Top** parameters. Note that when this button is switched to `On`, the *Clip Bottom* and *Clip*

Copyright © 2026 Vizrt

Top controls appear with values corresponding to the actual minimum and maximum found values in the data.

Note: Unlike using input range, the clipped bars are not seen at all rather than just representing minimum and maximum values defined in clipping.

Supported Formats

The following formats are supported:

- ASC File
- Viz Weather Data Format
- TAB File

ASC File

In the **.asc* format, the ASCII file represents a grid of values in defined geographic range, defining the number of rows and columns of such data. For example, the following shows a part of a file showing a grid of world values:

```
ncols         360
nrows        180
xllcorner     -180.0
yllcorner     -90.0
cellsize      1.000000000000
NODATA_value  -9999
-9999 -9999 -9999 0.1252774 0.1192886 …
```

Viz Weather Data Format

The data is contained in one **.ini* file and a corresponding list of binary files representing each timeframe. This data format enables the creation of an animation through timeframes. The *.ini* file defines what to expect from the binaries, including the region, projection and resolution (distance between samples of a grid data). The binaries are a list of values corresponding to one timeframe in float precision. Example of an *.ini* file:

```
[Grid]
ModelType=GFS
ModelName=GFS
DataName=Temperature
DataType=TEMPERATURE
FileType=BINARY
FromTime=201204102100
ToTime=201204132100
TimeStep=180
Region=-180.000000/180.000000/-90.000000/90.000000
Resolution=1
Compression=UNCOMPRESSED
[Projection]
Type=UNPROJECTED
```

Copyright © 2026 Vizrt

```
[Files]
File0=grid_0.b
File1=grid_1.b
File2=grid_2.b
[Times]
Time0=201204102100
Time1=201204110000
Time2=201204110300
[Mins,Maxs]
Vals0=-64.059998,37.739994
Vals1=-63.859997,35.239994
Vals2=-64.260010,36.139988
```

TAB File

The **.tab* format is used for non-grid type data. It contains a list of data samples defining the date, hour, coordinate and value of each. The first time such data is read, it is processed such that all the samples corresponding to the same hour represent one timeframe, and the processed data is saved as a cache (in the same folder where the **.tab* file is located) containing a **.bin* header file and numerous **.dat* files each corresponding to one frame. (The filenames of **.dat* files define the date and hour they represent). For example:

```
2013-06-29 00 -90.087890625 179.912109375 112
2013-06-29 00 -41.30859375 174.7265625 1
2013-06-29 01 -40.95703125 174.814453125 1
2013-06-29 01 -39.0234375 -68.115234375 2
2013-06-29 01 -38.759765625 -72.685546875 1
2013-06-29 02 -38.49609375 -63.6328125 1
2013-06-29 02 -38.232421875 144.31640625 7
2013-06-29 02 -37.96875 145.107421875 1
2013-06-29 02 -37.880859375 144.931640625 32
```

Creating Mask Images

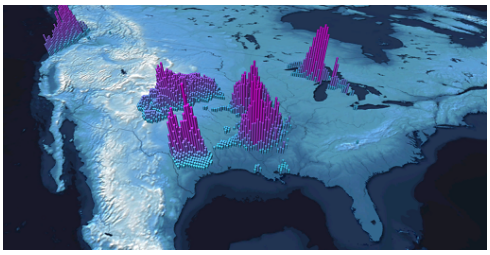
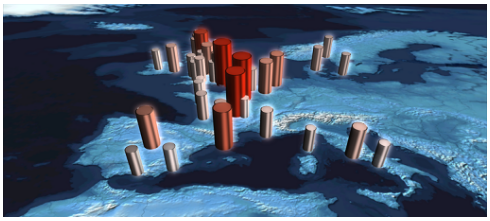
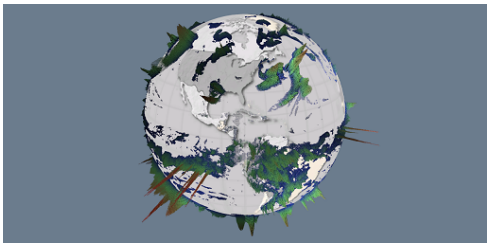
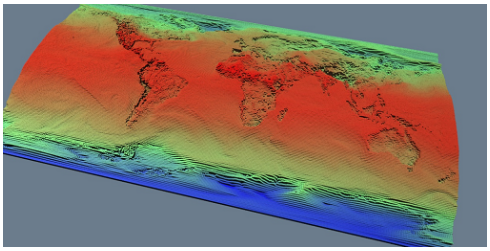
A mask image is created with the help of CWMClient.

The image should correspond exactly to a georeference on which GeoChart is placed, so the basemap region must be defined in the CWMClient. In the case when Atlas is used as georeference, put it beneath the CWM so that it corresponds exactly.

1. Go to the CWMClient defining the georeference and change the **Texture Compression** parameter to `None`. You can find it under the **Texture tab**.
2. Open Viz World Map Editor and select the desired region.
3. Put Map Layers Control on a container with CWM (the georeference).
4. In Map Layers Control, switch **Control** to `Enable` and deselect all of the options except the **Selected Regions**, and press **Refresh Map**.
5. Go to the CWMClient Miscellaneous tab and press **Freeze**.
6. Drag the resulting image from the container to the image pool. You can now use it as a mask for the selected region. You can unfreeze the CWMClient, remove Map Layers Control and change back the texture compression, but be careful not to change the size and range of the georeference map.

Copyright © 2026 Vizrt

Examples

Example	Description
	Weather precipitation data.
	Weather precipitation data clipped to a region.
	Social TV scene showing population of a particular word in Twitter.
	Weather precipitation data on the globe.
	World temperature values.

Copyright © 2026 Vizrt

5.3.12 GeoImage



The GeoImage plug-in enables geographic referencing options, used as a base object with maps.

GeoImage is short for Geographical Reference Image.

The plug-in has three views:

- Basic
- Pyramid
- Advanced

> **Note:** This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

GeoImage Properties

Basic

- **Use Base Map:** Defines the geographical referencing of the GeoImage. If Use Base Map is enabled (`On`), the GeoImage moves to its geographic location on the base map and resizes accordingly.
- **Map Size:** Sets the size of the GeoImage geometry if *Use Base Map* is set to `Off` .
- **Tessellation:** Changes the number of triangles of the rendered filecard to increase the visual quality.

> **Note:** The number of triangles affect the performance of the system.

- **Base Container:** Uses the first geo-referenced map above the GeoImage container in the hierarchy as the base map if empty. To use a specific map as the base map, drag a map container to the Base Container place holder.
- **Delta Z:** Sets a Z offset for the GeoImage geometry.
- **Map Status:** Displays the status of the imagery and whether the imagery contains geo-referencing.
- **Apply Fade Plugin:** Applies a Fade Texture plug-in to the container when clicked.

Pyramid

- **Pyramid:** Uses local parameters when to set to None or Local. When set to Global, it uses the parameters as defined in Hops Manager.
- **Map Style:** Defines the map style to be used for the pyramid maps:
  - **Source:** Uses the top CWM Client container (Map Tiler) style selection when creating the pyramid maps.
  - **Target:** Uses the hop CWM Client style selection when creating the pyramid maps.
  - **Split:** Splits the pyramid layers style, based on image latitude and longitude size. If the image size is smaller then the threshold the target style is used, if larger then the source style is used. Additionally, it is possible to turn on the color correction option in Pyramid Control which color corrects the target image to match the source images. Set the **Satellite Threshold (deg)**.

Copyright © 2026 Vizrt

- **Max Pyramid Height:** Defines the maximum number of maps that are created in the pyramid. The optimal number of pyramid maps is calculated by Pyramid Control. If the optimal number exceeds the *Max Pyramid Height* value, then the plug-in generates the maximum number defined.
- **Pyramid Factor:** Calculates the number of maps required defining the size factor between the maps of the pyramid.
- **Pyramid Steps:** Defines how the map coverage area grows from pyramid to pyramid:
  - **Equal:** The size factor is based on map pyramid height, pyramid factor and the resolution difference between the base map and the final map. The actual number might be different than the pyramid factor. The number of levels might also be smaller than the max pyramid height
  - **Constant:** Uses the value of the pyramid factor as is, with the value of max pyramid height.
- **Create Hole:** Creates a hole on each pyramid tile.
- **Image Aspect:** Calculates the closest power of two size, based on the selected image size of CWM Client texture size. The target uses exactly the same size as CWM Client texture size.
  Note: It is recommended to use this setting to improve performance.
- **Apply Fade:** Defines whether the maps used in the pyramid uses soft edges. Available options are Off, Target and Pyramid.
  - **Off:** Does not use soft edges. The transition between the maps is visible (the maps have sharp edges).
  - **Target:** Sets the last map (target) to have soft edges only.
  - **Pyramid:** Sets all maps in the pyramid to have soft edges.
- **Fade Range:** Sets the fade level (the area of the image that the fade is applied to) when the *Apply Fade* parameter is used.
- **Texture Compression:** Selects one of the required compression modes for the images.
- **Hide Lower Resolution:** Defines whether the map created by CWM Client, located on the Map Pyramid container, is turned off when the texture resolution of that map is lower than the Globe or GeoImage map tiles resolution. If it is enabled (`On`), the maps with lower resolution is turned off automatically by Map Pyramid. If it is disabled (`Off`), Map Pyramid does not turn off the low resolution maps.
- **Map Status:** Determines the status of the map.

Advanced

When the *Use Base Map* option on the Basic tab is turned on, the Advanced tab displays parameters for cropping the imagery applied on this container.

- **Crop Mode:** Defines whether the GeoImage is cropped:
  - **None:** Does not crop the map.
  - **Manual:** Sets the crop values using manual values for the map: **West** and **East** set the Longitude value for the western and eastern edge of the GeoImage map. **North** and **South** and set the Latitude value for the northern and southern edge of the GeoImage map.
  - **Base Map:** Crops the GeoImage according to the base map, that is, if the GeoImage exceeds the base map edges, it is cropped accordingly.

Copyright © 2026 Vizrt

5.3.13 Globe



The Globe plug-in enables geographic referencing options over a globe, used as a base object with maps.

The globe object geographically references the map at the correct location over the globe.



Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

This page contains information on the following topics:

- Globe Properties
  - Basic
  - Pyramid
  - Advanced
- Buttons

Globe Properties

Basic

- **Tessellation:** Defines the number of polygons used to create the Globe geometry.
- **Radius:** Defines the radius of the globe.
- **Map Status:** Reports if the map was loaded as expected (*Ok*), geographic referencing of the map, and so on.

Copyright © 2026 Vizrt

Pyramid

- **Pyramid:** Uses local parameters when to set to None or Local. When set to Global, it uses the parameters as defined in Hops Manager.
- **Map Style:** Defines the map style to be used for the pyramid maps:
  - **Source:** Uses the top CWMClient container (Map Tiler) style selection when creating the pyramid maps.
  - **Target:** Uses the hop CWMClient style selection when creating the pyramid maps.
  - **Split:** Splits the pyramid layers style, based on image latitude and longitude size. If the image size is smaller then the threshold the target style is used, if larger then the source style is used. Additionally, it is possible to turn on the color correction option in Pyramid Control which color corrects the target image to match the source images. Set the **Satellite Threshold (deg)**.
- **Max Pyramid Height:** Defines the maximum number of maps that are created in the pyramid. The optimal number of pyramid maps is calculated by the Map Pyramid plug-in. If the optimal number exceeds the *Max Pyramid Height* value, then the plug-in generates the maximum number defined.
- **Pyramid Factor:** Calculates the number of maps required defining the size factor between the maps of the pyramid.
- **Pyramid Steps:** Defines how the map coverage area grows from pyramid to pyramid:
  - **Equal:** The size factor is based on map pyramid height, pyramid factor and the resolution difference between the base map and the final map. The actual number might be different than the pyramid factor. The number of levels might also be smaller than the max pyramid height
  - **Constant:** Uses the value of the pyramid factor as is, with the value of max pyramid height.
- **Create Hole:** Creates a hole on each pyramid tile.
- **Image Aspect:** Calculates the closest power of two size, based on the selected image size of CWM Client texture size. Target uses exactly the same size as CWM Client texture size.

> ⚠ **Note:** It is recommended to use this setting to improve performance.

- **Apply Fade:** Defines whether the maps used in the pyramid uses soft edges. Available options are Off, Target and Pyramid.
  - **Off:** Does not use soft edges. The transition between the maps is visible (the maps have sharp edges).
  - **Target:** Sets the last map (target) to have soft edges only.
  - **Pyramid:** Sets all maps in the pyramid to have soft edges.
- **Fade Range:** Sets the fade level (the area of the image that the fade is applied to) when the *Apply Fade* parameter is used.
- **Texture Compression:** Selects one of the required compression modes for the images.
- **Hide Lower Resolution:** Defines whether the map created by CWM Client, located on the Map Pyramid container, is turned off when the texture resolution of that map is lower than the Globe or GeoImage map tiles resolution. If it is enabled (`On`), the maps with lower resolution is turned off automatically by the Map Pyramid plug-in. If it is disabled (`Off`), the Map Pyramid does not turn off the low resolution maps.
- **Map Status:** Determines the status of the map.

Copyright © 2026 Vizrt

Advanced

- **Sub Tessellation:** Divides each of the Globe flat areas, to increase texture coordinates and improve the way the textures look on a globe.
- **Texture Mapping:** Defines how the texture (map) is mapped over the Globe:
  - **Absolute:** Crops the globe object according to the values set in the West, East, South and North parameters.
  - **Relative:** Crops the globe object according to the geographical properties of the map applied to the Globe.
  - **GeoRef:** Maps the map over the globe object according to its geographical properties.
- **West/East:** Sets the Longitude value for the western/eastern edge of the Globe. The globe object is cropped at that Longitude.
- **South/North:** Sets the Latitude value for the southern/northern edge of the Globe. The globe object is cropped at that Latitude.
- **Enable Hole:** Saves performance when building pyramids using globe objects. When the globe objects for the pyramid layers are created, a hole is created where the higher resolution maps are, to avoid rendering multiple pixel layers (and to save performance). It can be used manually to create a hole in a globe object by setting values which are in the area of the map in use. When enabled (`On`) additional parameters are enabled:
  - **Hole West (From lon):** Defines an inner Longitude in the map area where the hole begins (West side).
  - **Hole East (To lon):** Defines an inner Longitude in the map area where the hole begins (East side).
  - **Hole South (From lat):** Defines an inner Latitude in the map area where the hole begins (South side).
  - **Hole North (To lat):** Defines an inner Latitude in the map area where the hole begins (North side).
- **Avoid Radius Cheat:** Contains a Radius setting that can be avoided if required.

> ⚠ **Note:** This is a very advanced feature which should be used with caution.

Buttons

- **Build Pyramid:** Rebuilds the pyramids.

If Pyramid is set to None in the Pyramid tab, then the available buttons are:

- **Apply Fade Plugin:** Applies a Fade Texture plug-in to the container when clicked.
- **Face Z Axis:** Rotates and centers the globe object facing the Z-axis. This button can be used when not a full globe is rendered (for example, only Asia). Do not use this feature with Navigator, as it assumes no rotation.

Copyright © 2026 Vizrt

5.3.14 Label and Go



The Label and Go plug-in draws multiple labels on screen. It is mainly used for interactive scenes, but it can also be used for regular scene design. Traditionally, label drawing has been done by repeatedly copying a design. This method gives you full control over the look of the label, but limits the number of labels which can be used as every label has at least two containers. Also, if the number of labels exceed 500 containers, it may cause a performance issue.

Label and Go gives fewer options for design, but since all design is done for one container performance is still good using 70,000 labels.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

This page contains information on the following topics and procedures:

- Label and Go Properties
  - Global
  - Design
    - Caption
    - Background
    - Pointer
  - Communication
  - Shared Memory Communication with Label and Go
  - Example
- Working with Label and Go
  - To Create a Label and Go Scene
  - To Use Viz Config for Interactive Communication
  - Notes

Label and Go Properties

Global

- **Data source:** Defines the type of source the plug-in should render. Label and Go supports the following data sources:
  - **Auto labels:** Uses labels of the entire world.
  - **CWM labels:** Uses labels created by CWM Client.
  - **Streets:** Uses street labels (created by 3D Road Manager or CWM Client).
  - **Other:** Uses other labels. Currently Viz World supports the KML Reader.
  - **Local file:** Determines the shape file that contains names and geo positions.
  - **Shields:** Allows the designer to set designs for road types such as highways, interstates, etc., and to use an iconic representation of the street instead of the street name. This is based on street data and currently only supports Tom-Tom data for the USA. During import of the street data, a CSV file is created under *plugin\data\maps\DesignMappings*, which contains all the types of shields found in the data. The Mapping column in this file can be edited to map the shield types to specific user designs, and then the Label and Go plug-in is used to read the file.

Copyright © 2026 Vizrt

- **Labels Draw Style**
  - **On Map:** Places the labels on the same camera as the navigator camera.
  - **Overlay:** Places the labels on the second camera (normally front layer).
- **Alpha fade time:** Determines the amount of time (fields) for new labels to appear or disappear.
- **Max number of labels:** Determines the maximum number of label to render including all the labels so far (other Label and Go are taken into account as well).
- **Culling bounding box:** Scales the real bounding box to avoid too many labels on screen.
- **Line spacing:** Determines line spacing when using two lines.

Design

- **Font:** Sets the font used by all labels.
- **Label Type:** Selects from the default types of labels when using auto-labels on Data Source (Country, Capital, etc). Using POI-1, POI-2 and so on, allows definition of the name of the style, for example: Earthquake, Explosion, etc.
- **Render:** Renders the selected label design type (for example, Country, POI-1 and so on) when set to `On`.
- **View Distance:** Sets the distance from where the label style should be visible.
- **Set Distance:** Sets the View Distance property based on the current distance of the map.
- **Priority:** Overwrites the default priority which is based on type and town size.
- **Caption/Background/Pointer:** The Caption, Background and Pointer properties are different for each type of design, and are described below.
- **Buttons**
  - **All Off:** Turns off all labels, except the one currently being edited.
  - **All On:** Turns on all labels.
  - **Copy setting from type above/below:** Copies the settings from the label type above/below.
- **Outline Quality:** Determines the quality when background is set to outline.
- **Outline Quality Mask:** Selects the mask to use so it does not interfere with the scene regular makes.
- **Debug:** Draws a green rectangle around all/current labels. (useful to see the bounding box).

Caption

- **Caption**
  - **Hide:** Displays only the pointer of each label.
  - **Show:** Displays both the pointer and the text of each label.
- **Text color:** Sets the color of the text.
- **Text Size:** Sets the size of the text. Text size influences all sizes.
- **Max size:** Limits the size of the text on either X or X/Y axis when set to either Scale X or ScaleX/Y.
- **Max width:** Sets the maximum width the text should take and scales the text either on X or XY based on the Max Size setting.
- **X/Y Offset:** Offsets the text on X/Y from its latitude and longitude.
- **Rotate:** Rotates labels to follow the street when set to `On`. When set to `Off` they are not rotated.

Background

- **Background:** Defines the background type for the text (none/rectangle/image/outline).
- **Size:** Sets the size of the background in relation to the text in percent (%).

Copyright © 2026 Vizrt

- **Image:** Sets a background image for the text.
- **Background color:** Sets the background color.
- **Outline color:** Sets the outline color.
- **Outline width:** Sets the outline width.
- **Background Fixed Size, Background X, Background Y:** Defines the X/Y size of the background when set to fixed size.

Pointer

- **Point:** Sets the type of point.
- **Caption position:** Sets the position of the label in relation to the point.
- **Point Color:** Sets the color of the point.
- **Point Size:** Sets the size of the point.
- **Point Distance (%):** Sets the distance of the label in relation to the point in percent (%).
- **Point BG:** Enables (`On`) or disables (`Off`) the Point BG properties.
- **Point BG Color:** Sets the points background color.
- **Point BG Size (%):** Sets the points background size in percent (%).

Communication

- **Interactive Mode:** This property is here for backwards compatibility only. The recommended way is to use MTButton instead.
- **Shared Memory Type:** Changes between Scene-, Global- and Distributed-Shared Memory.
- **Shared Memory Identifier:** Sets the Shared Memory key name.
- **Set Data Pool:** Enables the distribution of the data to a DataPool structure using the same name as the Shared Memory Identifier.

Shared Memory Communication with Label and Go

To send labels via shared memory to Label and Go, you need to send:

```
Id,Label,long,
lat style ( optional )
```

Each label must start with `id` .

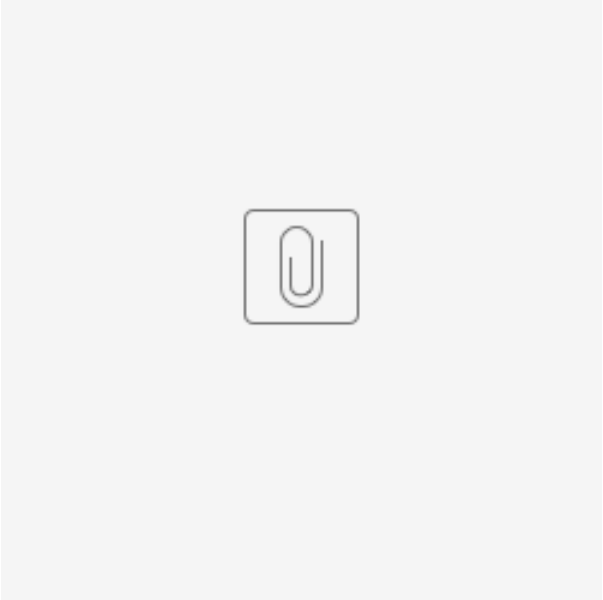
Example

```
dim Variable as string
dim array as Array[string]
\qMy data separated my new line
dim data as String = GetParameterString("Data")
data.Split("\n\r", array)
\qDepending on label@Go setting
Variable = "LABELANDGO_"
Variable.Append("ADDLABELS")
```

```
Scene.Map[Variable] = array
```

Working with Label and Go

To Create a Label and Go Scene



1. Add Atlas to your scene tree.
2. Add Navigator to the same container as Atlas.
3. Open the Navigator editor, click Miscellaneous and set **Interactive Mode** to **Always.**
4. Add a **sub container** to the GeoReferenceMap.
5. Add the **Label and Go** plug-in to the new container.

Copyright © 2026 Vizrt



6. Click the **Scene Editor's E** button. **E** enables the handling of interactive script/plug-in events. Interactive scripts and plug-ins are those related to mouse or keyboard actions.
7. Click the scene and **zoom in and out** with your mouse to see the labels appear.
8. Open the Label and Go editor to test the settings.

To Use Viz Config for Interactive Communication

When using MTButton for interactivity together with Label and Go:

1. The input method from Viz Config is used (for example, Win 7, PPI, TUIO, mouse).
2. The clicked label on the map is published to shared memory.

Notes

- All data sources may select from ten different designs.
- There are currently no limits as to how many Label and Go plug-ins can be used by one scene.
- Label collision is calculated and taken into account. Other types of labels (for example, design using Street Labels) are not taken into account.
- Label and Go currently works on flat maps only.

Copyright © 2026 Vizrt

5.3.15 MapScale



The MapScale plug-in displays the scale factor of the map.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

MapScale Properties

General

- **Geo Source:** Bases the scale or zoom factor data on the navigator plug-in, when set to Navigator. When set to Map, the values are based on the defined Map container.
- **Screen Size (%):** Defines the screen geometry size, measured as percentage of the screen on the X axis.
- **Height (%):** Defines the height geometry, measured as percentage of the screen on the Y axis.
- **Units:** Displays units in Kilometers or Miles.
- **Labeling Mode:** The **Large/Small Limps** options try to find the closest round number in the visible range. While gradually zooming in or out, the rectangle containing the map scale is scaled according to a visible number. When the range limit is reached, the number changes and the rectangle jumps to a new range. This may result in an unpleasant visual effect, so the **Continuous** mode is used where the rectangle showing the scale remains unchanged and the numbers showing the range change continuously without jumps.
    - **Large Limps:** Changes the text display when the scale factor changes significantly (for example, 1000KM, and next limp 500KM).
    - **Small Limps:** Changes the text display when the scale factor is a smaller change (for example, 1000KM, and next limp 900KM).
    - **Continuous:** Provides an exact number of the coverage area of the geometry.
    - **Ruler:** Combines Continuous and Limps modes. While the total width of the rectangle remains unchanged, notches are added showing the rounded range.
- **Shape:** Determines the shape of the edges: Rectangular edges, Rounded Edges or Notched left and right. When using Notched, an additional parameter is visible for the direction for the notches (Up/Down/Both and the height and width of each notch).
- **Expansion:**
- **Outline:** Toggles the outline of the geometry.
- **Outline Width:** Defines the width of the outline when outline is enabled (`On`).
- **Outline Color:** Defines the color of the outline when outline is enabled (`On`).

Advanced

- **Camera:** Determines the camera for placing the geometry.
- **Centimeter/Meter/Kilometers/Feet/Miles Label:** Defines the suffix for each unit.

Copyright © 2026 Vizrt

5.3.16 Pyramid Control



The Pyramid Control plug-in sorts overlapped layers of pyramids in the scene.

This is so tiles with higher resolution are not hidden by tiles with lower resolution.

Note: The plug-in must be located in the hierarchy of Map Tiler.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps

Pyramid Control Properties

- **Fade Tiles:** Applies a fade animation to each tile level based on the distance to the hop when enabled.
- **Start/End Fade (%):** Sets the distance (percentage) to the hop for starting and ending the fade when *Fade Tiles* is enabled.
- **Bind to Hop:** Synchronizes the visibility of pyramids based on the hop that they belong to when enabled. When disabled (default), the pyramids are shown based on the highest resolution available throughout the entire scene.

Note: When using Control MultiHop, the *Bind To Hop* property is set to *Enabled* by default.

- **Pyramid Fade Start (%):** Determines the duration that the pyramids are fully visible, in percentage of total hop time.
- **Target Fade Start (%):** Determines the duration that the target image are fully visible, in percentage of total hop time.
- **Pyramid Fade Length (%):** Determines the time taken for pyramids to fade, in percentage of total hop time.
- **Target Fade Length (%):** Determines the time taken for target to fade, in percentage of total hop time.
- **RenderTile:** Selects which pyramid image to display, from all the pyramid levels. Used mainly for debugging the graphics.
- **Wire Frame:** Shows the pyramids in wire frame mode, to help understand the coverage of the pyramids. Used mainly for debugging the graphics.
- **Show Bounding Boxes:** Renders the bounding box of the pyramids.
- **Refresh All Pyramids (button):** Forces a refresh of all pyramids for all hops in the scene.

Copyright © 2026 Vizrt

5.3.17 ShadowAgent



The Shadow Agent plug-in is an agent for the Shadows parameters covered by Label Manager.

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps_Adv

Shadow Agent Properties

- **Render With Camera:** Defines for which camera shadows apply.
  - **All:** Renders shadows on all cameras (Compatible with previous versions).
  - **Label:** Renders shadows only for the defined off screen labels camera.
  - **Map:** Renders shadows only for the defined map camera.
  - **Shadow:** Renders shadows only for the defined shadows camera, as defined in Label Manager (Shadows tab).

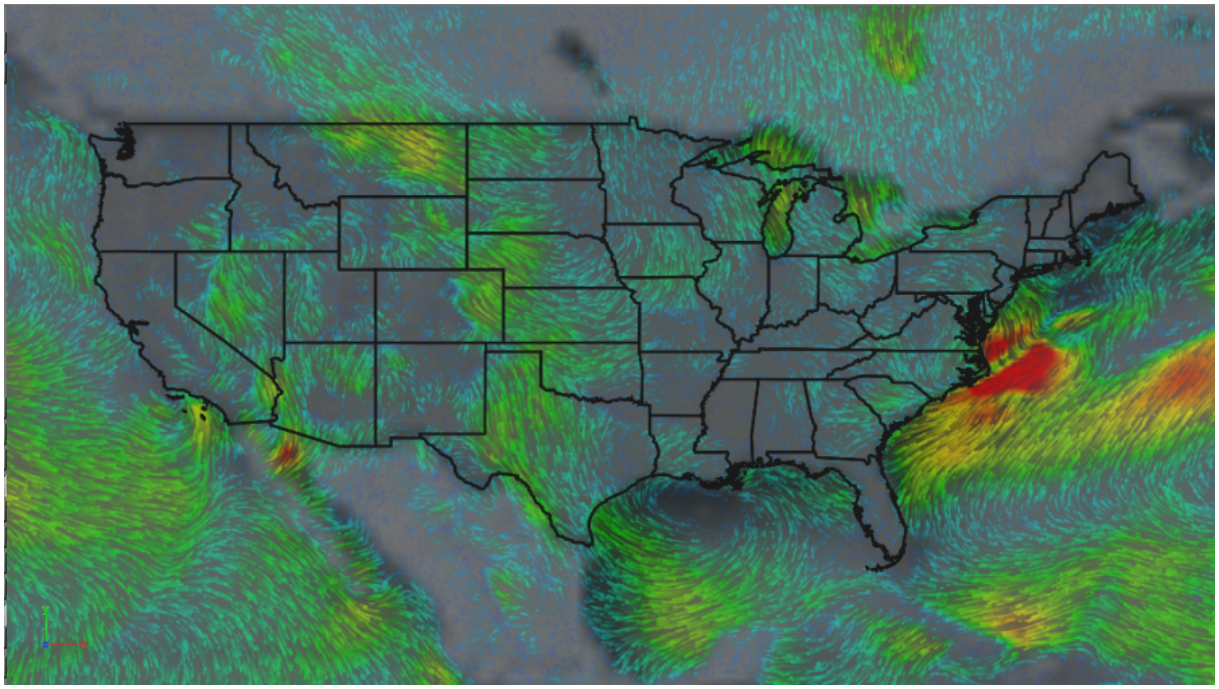
Copyright © 2026 Vizrt

5.3.18 WindFlows

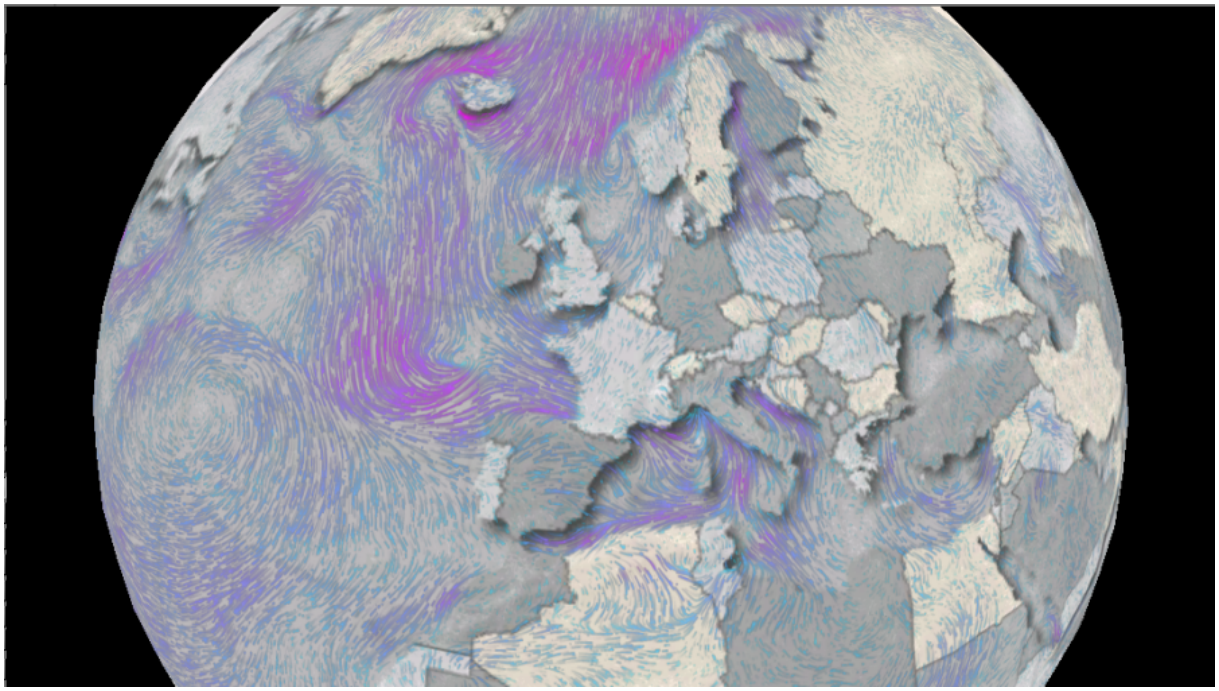


The WindFlows plug-in visualizes geo-located winds by representing it as running flows of various length and colors on top of a map.

The flow direction corresponds to the wind direction. WindFlows can load multiple time frames of wind velocities and directions, and produce a smooth animation over time.



Copyright © 2026 Vizrt



This section contains the following information:

- WindFlows Properties
  - General
  - Color
  - Advanced
- Appendix A - Showing Winds on Top of Color Heat Map
- Appendix B - Example of Masking to Desired Region

Note: This plug-in is located in: Plugins -> Geometry plug-ins -> Maps_Adv

WindFlows Properties

General

- **Data Path:** Specifies data files or relevant data (comes in u and v components). This should be a weather grid model data of type Wind. An *hours* folder of windU or windV component should be specified as input.

Example: `W:\DemoData\WindSpeed\EC025\windU\contourdata\hours`

- **Start Time / End Time:** Loads only relevant data for that time frame if time frame is specified.
- **Frame Number:** Selects time frame of data.
- **Frame Interpolation Mode:** Loads all available frames of animation to produce a smooth animation between time frames.

Copyright © 2026 Vizrt

⚠ The **Data Buffer Size [mb]** parameter under the **Advanced** options limits the number of time frames that is being loaded to animation according to desired size.

- **Timeline (frames):** Control time progress between frames while in interpolation mode. Only this parameter and not *Frame Number* should be animated.
- **Flows:** Determines the number of flows representing wind. Flows are randomly distributed across the canvas.
- **Line Width:** Defines width of strokes. Width is specified in pixels of canvas, changing canvas size causes flows to look as if their width was changed.
- **Max Age:** Defines maximum age in frames of the flows. The age (length) of each stroke is a random quantity; however, flows presented in area of strong winds move faster and those appear longer.
- **Speed:** Controls general speed level of flows, although faster winds always result in faster flows.
- **Fade Rate (frames):** Determines the number of frames it takes to completely fade out the stroke.
- **Canvas Resolution:** Renders flows to a canvas and then this canvas is applied to map like a texture. This parameter allows changing the size of the canvas. Large canvases are useful when small zoom-ins are required without losing the resolution. Flows widths, however, are affected by canvas size. When using **Dynamic** mode, there is no reason to go beyond **Normal** canvas size.
  - **Normal:** Corresponds to HD resolution (1920X1080).
  - **High:** Corresponds to 4K (3840x2160).
  - **Ultra High:** Corresponds to 7680x4320.

Color

- **Color Source:** Defines whether color presets are taken from image or a manual preset.
- **Color Presets:** Predefines color presets to start with. Every change sets the position to custom.
- **Colors:** Defines number of colors to be used in color-map, as well as their values.
- **Color Image:** Uses an image to define the color map. Image should be dragged to this control. An example of an image defining color map:



- **Posterization:** Splits a continuous color map to a desired number of discrete colors.
- **Color Levels:** Defines number of desired levels for posterization.
- **Use Flows Alpha:** Allows wind strength to affect flow transparency when set to `On`.

Advanced

- **Wireframe:** Displays a wireframe of the canvas geometry (may be useful for globe geography).
- **Data Buffer Size (mb):** Defines limit for the amount of data to be loaded to the plug-in, used to limit number of frames in animation.
- **Invert Depth Test:** Masks the desired area and shows winds only above that area. A masking image should be put on top of wind flows and this option enabled. See example of the result in Example of Masking to Desired Region.

Copyright © 2026 Vizrt

- **Canvas Positioning:** Defines canvas coverage behavior. Flows are actually rendered to canvas, and then this canvas is placed on top of geo geometry.
    - **Dynamic:** Calculates canvas size in each frame. Canvas is of the size of the visible geographic area.
    - **Static:** Covers basemap/data range and stays in place. As a result, if you zoom-in in static mode the resolution of flows deteriorates.
    - **Use Window:** Limits the visualization area to a specific geographic region manually defined by west/east/north/south borders. Enabled only in *Static* mode.
- **Autodetect Max Value:** Detects maximum value of loaded data and normalizes the visualization accordingly when set to `On`. When set to `Off`, it enables manual setting of the maximum value for normalization. All data larger than maximum defined value obtains maximum value for visualization (maximum alpha and color representing strongest wind).
- **Clip Data:** Discards flows larger or smaller of specified clipping values.
- **Clip Bottom:** Discards flows smaller of specified clipping value.
- **Clip Top:** Discards flows larger of specified clipping value.
- **Adaptive Density:** Reduces the number of flows dynamically according to the zoom level such that there are fewer flows as you zoom-in when set to `On`.
- **Density Upper Bound/Density Lower Bound:** Defines percentage of flows (out of the number defined in General tab) for the current visible geographic range (When value is set, the visible geo area at that moment is considered). Any zoom-in level below the one saved for the lower bound or zoom-out above the one saved for the upper bound has lower/upper amount of flows correspondingly. In between, the number is interpolated.
- **Render On Move:** Defines behavior of the rendering during movement in dynamic canvas positioning mode. Since it is problematic to move canvas with already rendered data dynamically, add new data and still look correct a number of behaviors were defined:
    - **None:** Shows nothing during movement.
    - **Defer:** Preserves the previous canvas state until movement stops.
    - **One Frame:** Renders only one frame of flows, with the canvas being reset after each frame.
    - **Continuous:** Produces a spread-blur effect.
- **Smooth:** Defines smoothing quality of the flows. Be careful with **Multisample HQ**, as it can be very expensive performance-wise.

Appendix A - Showing Winds on Top of Color Heat Map

It is possible to show flows on top of Color heat-map representing wind strength. IsoGrid is used to show heat-map and WindFlows shows flows on top of it. Some change is required in configuration of the fetch to produce another folder with wind data representing wind strength (instead of just one vector component). This folder is called windS and it should be loaded to IsoGrid in order to show the heat map.

Open WD Administration Tool, go to Providers Info and add the following line for each provider:

```
<entry name="isoScriptOutput">contours.log &amp; WindSpeed.py --dataset
&quot;PATH_TO_DATA_FOLDER&quot; -R_MODEL_GEO_BOUNDS -I_MODEL_RESOLUTION</entry>
```

- `PATH_TO_DATA_FOLDER` is a path to folder containing WindU and WindV folders.
- `MODEL_GEO_BOUNDS` are the bounds found in *config.ini* file of each model under *Region* section.

Copyright © 2026 Vizrt

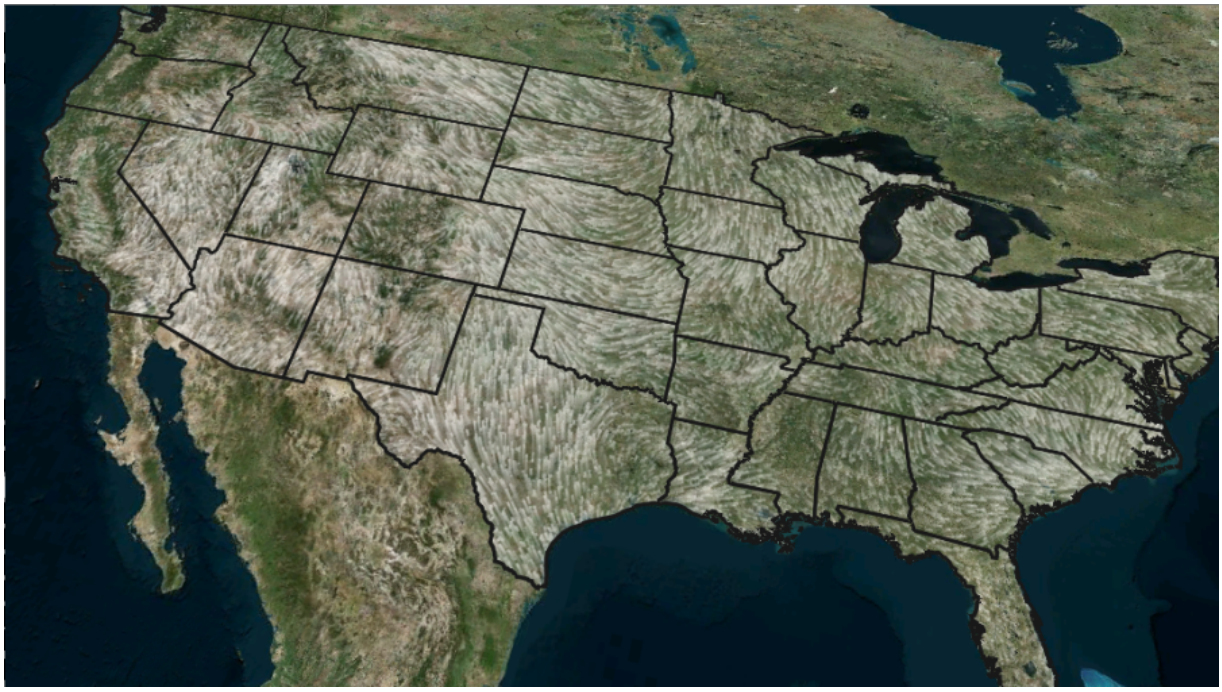
- `MODEL_RESOLUTION` is resolution of data in degrees, also found in the *config.ini* file.

Example of such a line for GFSX model (0.5 degrees resolution):

```
<entry name="isoScriptOutput">contours.log &amp; WindSpeed.py --dataset &quot;C:
\Program Files (x86)\vizrt\vizWeather2\SharedData\FetchData\Wind\GFSX&quot;
-R-180.000000/180.000000/-90.000000/90.000000 -I0.5</entry>
```

As mentioned above, after adding this line fetch application should fetch three folders for wind data – WindU, WindV and WindS representing strength.

Appendix B - Example of Masking to Desired Region



Mask image is created with the help of CWM client plug-in. The image should correspond exactly to a georeference on which GeoChart is placed, so make sure the basemap region is defined in CWM Client (in a case when Atlas is used as georeference put it beneath the CWM to correspond exactly).

1. Go to the CWM Client defining the georeference and change the **Texture Compression** parameter to `None`. You can find it under the **Texture tab**.
2. Open Viz World Map Editor and select desired region.
3. Put the Map Layers Control on a container with CWM (the georeference).
4. In the Map Layers Control, switch **Control** to `Enable` and deselect all options except the **Selected Regions** and press **Refresh Map**.
5. Go to the CWM Client Miscellaneous tab and press **Freeze**.

Copyright © 2026 Vizrt

6. Drag the result image from container to the image pool. You can now use it as a mask for the selected region. You can unfreeze the CWM Client, remove Map Layers Control and change back texture compression, but be careful not to change the size and range of georeference map.

Copyright © 2026 Vizrt

5.4 Maps Container Plug-ins

This chapter describes container plug-ins. The container plug-ins are found in the following plug-in folders:

- **Maps:** Contains standard plug-ins.
- **Maps-Adv:** Contains advanced plug-ins.
- **Maps-Lab:** Contains experimental plug-ins. Since these plug-ins are experimental and not supported, they are not documented here.
- **Maps-Man:** Contains a set of Manager plug-ins used for batch creation of 3D Objects such as regions, borders, and so on.
- **Maps-Obs:** Contains obsolete plug-ins, installed only for backward compatibility. These plug-ins should **not** be used when designing new scenes. Since these plug-ins are obsolete and not supported, they are not documented here.
- **Tools:** Contains tool plug-ins.

See the following sections for more information:

- 3D Border Manager
- 3D Line Manager
- 3D Line Tracer
- 3D Map Telestrator
- 3D Map Telestrator Design
- 3D Region Manager
- 3D Roads Manager
- Center Map
- CWM Client
- Focus On Map
- Geo Text
- GeoData Reader
- Hop It
- Hop Sync
- Hops Manager
- KML Reader
- Label AddOn
- Label and Go Assistant
- Label It
- LatLongGrid
- Locator Control
- Map Layers
- Map Layers Control
- Map Tiler
- Map Zoom
- Mouse2Memory
- Mute
- NavCom
- NavFade

Copyright © 2026 Vizrt

- NavFinder
- Navigator
- NavScale
- NavSlave
- Place Finder
- Publish To Design
- Region2Tex
- Trace It
- World Position

Copyright © 2026 Vizrt

5.4.1 3D Border Manager



The 3D Border Manager plug-in creates 3DBorder objects based on shape files.

The plug-in uses a 3DBorder design to create borders according to the defined settings.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Man

3D Border Manager Properties

- **Design:** Sets the 3DBorder design container that is used for building lines. The design container should be built with a 3DBorder object and a material. No special naming convention is required.
- **Target:** Defines the container to be used as the parent container for the border containers:
  - **Me:** Builds 3DBorder objects under the current container (holding the 3D Border Manager plug-in).
  - **Next:** Builds 3DBorder Objects under the next container (next container in the tree and at the same level as the 3D Border Manager container).
  - **Down:** Builds 3DBorder objects under the first child container.
  - **Container:** Builds 3DBorder objects under the container dragged into the *Target* container place holder.
- **Target Container:** Specifies the container that holds all the 3DBorder objects.
- **Shape File:** Defines a path to the shape file (*.shp*), containing the border definitions. Shape files are bought from vendors specialized in Geographic Information System (GIS) and holds the actual shape data; polygons, splines, and others.

Note: Shape files must be stored in individual folders.

- **Simplify Threshold:** Sets the detail reduction factor for the shape borders.
- **Persistent Data:** Defines whether the data is removed from Viz memory when the scene is closed or not. When enabled references are kept, and load time is quicker.
- **Border Type:** Defines which border type associated with the created objects. If Advanced is selected, additional parameters are enabled allowing the configuration of border type according to the data associated with the shape file.
  - **Borders Type:** Specifies the column name that holds each border type. **Country ID, Region ID, Sub Region ID and Coastline ID:** Specifies the string in the database file (*.dbf*) that matches to each type (for example, *Country*, *Region* and so on).
- **Clear and Build:** Removes all child objects of the target container and rebuilds the objects, using the plug-in settings.
- **Clear Target Tree:** Removes all child objects of the target container.

Copyright © 2026 Vizrt

5.4.2 3D Line Manager



The 3D Line Manager plug-in controls and creates 3DLine objects. The plug-in uses a 3DLine design to create lines according to the defined settings.

The plug-in allows you to build 3D Line in four different modes: Navigator, Command, from WME and from a Shape file.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps_Man

3D Line Manager Properties

Selecting the **Data Source** defines how the 3DLine objects are created:

Navigator

Creates a 3D Line object using the hop locations defined in the Navigator plug-in. The line starts at the first hop location and end at the last hop location. Note that when using the Navigator as the data source, the 3D Line Manager should reside under a Navigator container.

- **Design:** Sets the region design container that is used for building sub-regions. The design container should be built with a 3DLine object and a material. No special naming convention is required.
- **Target:** Defines the container to be used as the parent container for the sub regions containers:
  - **Me:** Builds the 3DLine objects under the current container (holding the 3D Line Manager plug-in).
  - **Next:** Builds the 3DLine objects under the next container (next container in the tree and at the same level as the 3D Line Manager container).
  - **Down:** Builds the 3DLine objects under the first child container.
  - **Container:** Builds the 3DLine objects under the container dragged into the **Target** container place holder.
- **Simplify Threshold (m):** Sets the detail reduction factor for the shape of the lines.
- **Spline:** Determines how to display the spline.
- **Tessellation:** Sets the degree of detail for use with spline.
- **Start Hop:** Determines the starting point for the hop.
- **Number of Hops:** Determines the number of hop points to use.
- **Rebuild:** Forces a rebuild on the cache files generated for the selected shape file.

Command

Creates the 3DLine object from a list of Long/Lat pairs defined by the user, for example: `MyLine: 0,0 0,50 30,30`.

- **Design:** Sets the region design container that is used for building sub-regions. The design container should be built with a 3DLine object and a material. No special naming convention is required.
- **Target:** Defines the container to be used as the parent container for the sub regions containers:
  - **Me:** Builds the 3DLine objects under the current container (holding the 3D Line Manager plug-in).
  - **Next:** Builds the 3DLine objects under the next container (next container in the tree and at the same level as the 3D Line Manager container).

Copyright © 2026 Vizrt

    - **Down:** Builds the 3DLine objects under the first child container.
    - **Container:** Builds the 3DLine objects under the container dragged into the **Target** container place holder.
- **Simplify Threshold (m):** Sets the detail reduction factor for the shape of the lines.
- **Spline:** Determines how to display the spline.
- **Tessellation:** Sets the degree of detail for use with spline.
- **Rebuild:** Forces a rebuild on the cache files generated for the selected shape file.

WME Labels

Allows you to build a line between locations chosen in Map Editor Classic (WME) as WME labels (no line is built if WME labels are not defined). Also, the WME source container must be specified and refresh should be applied by clicking the **Refresh** button in CWMClient (Rebuild button is disabled in this mode). The line between the labels starts and ends in the order they where added (as seen in WME's Map Details list). The first label location is the label at the top of the list, and the last is at the bottom of the list. Rearrange the list to create a new order.

- **Design:** Sets the region design container that is used for building sub-regions. The design container should be built with a 3DLine object and a material. No special naming convention is required.
- **Target:** Defines the container to be used as the parent container for the sub regions containers:
    - **Me:** Builds the 3DLine objects under the current container (holding the 3D Line Manager plug-in).
    - **Next:** Builds the 3DLine objects under the next container (next container in the tree and at the same level as the 3D Line Manager container).
    - **Down:** Builds the 3DLine objects under the first child container.
    - **Container:** Builds the 3DLine objects under the container dragged into the **Target** container place holder.
- **Simplify Threshold (m):** Sets the detail reduction factor for the shape of the lines.
- **Spline:** Determines how to display the spline.
- **Tessellation:** Sets the degree of detail for use with spline.
- **Set Navigator Hops:** Enables the scene to do a hop animation between the lines.
- **Exclude Hops:** Excludes the hop animation for the following labels; None, First, Last, or First and Last. Manual Shift allows you to define which destination is the starting point (limited to a selection of 30 destinations).
- **Set Hops Height:** Determines the height of the hops.

Shape File

Uses a shape file as data source.



Note: In WME Labels option, the 3DLineManager should be attached to a CWMClient container. The Labels in the CWMClient plug-in must be enabled.

- **Design:** Sets the region design container that is used for building sub-regions. The design container should be built with a 3DLine object and a material. No special naming convention is required.
- **Target:** Defines the container to be used as the parent container for the sub regions containers:
    - **Me:** Builds the 3DLine objects under the current container (holding the 3D Line Manager plug-in).
    - **Next:** Builds the 3DLine objects under the next container (next container in the tree and at the same level as the 3D Line Manager container).

Copyright © 2026 Vizrt

    - **Down:** Builds the 3DLine objects under the first child container.
    - **Container:** Builds the 3DLine objects under the container dragged into the **Target** container place holder.
- **Shape File:** Defines a path to the shape file (**.shp*), containing the line definitions. Shape files are bought from vendors specialized in Geographic Information System (GIS) and holds the actual shape data; polygons, splines, and others. Note that shape files must be stored in individual folders.
- **Name Column:** Determines the column to search for names in the shape file.
- **ID Column:** Determines the column to search for IDs in the shape file.
    - **Compare as:** Compares line types as strings or numbers. Numbers allow for a range to be entered.
- **Line Type Column:** Determines the column to search for line types in the shape file.
- **Line Types:** Determines line type to use when importing shape files.
- **Simplify Threshold (m):** Sets the detail reduction factor for the shape of the lines.
- **Spline:** Determines how to display the spline.
- **Tessellation:** Sets the degree of detail for use with spline.
- **Rebuild:** Forces a rebuild on the cache files generated for the selected shape file.

Flow and Routing

Defines the Simplify Threshold and FRC level for the selected data source. **Simplify Threshold** reduces the resolution of the data from the shape file selected. **FRC level** (High/Medium/Low) sets the FRC level, for example, *High* would draw highways and major roads.

- **Shape File:** Defines a path to the shape file (**.shp*), containing the line definitions. Shape files are bought from vendors specialized in Geographic Information System (GIS) and holds the actual shape data; polygons, splines, and others. Note that shape files must be stored in individual folders.
- **Simplify Threshold (m):** Sets the detail reduction factor for the shape of the lines.
- **Level Target:** Container placeholder for selected level of roads.
- **Motorway/Freeway, Major Road, Other Major Road, Secondary Road, Local Connecting Road, Local Road of High Importance:** When enabled (*On*), the plugin draws roads rated according to the selection in the loaded roads data.
- **Rebuild:** Forces a rebuild on the cache files generated for the selected shape file.
- **Build Hierarchy and Data:** Builds base scene hierarchy for traffic flows scene type and generate needed cache data from the selected shape file.

Copyright © 2026 Vizrt

5.4.3 3D Line Tracer



The 3D Line Tracer plug-in works in conjunction with the 3DLineManager.

It allows to trace lines, build Routes of one point or two points when the Data Source is in Flow and Routing and you have the correct shape file (TomTom TMC Data).

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

3D Line Tracer Properties

- **Tracing**
  - **None:** No tracing.
  - **One Line:** Traces one line. A mouse click selects a line.
  - **Two Points:** Traces two points. The first mouse click starts and the second mouse click ends.
  - **Continuous:** Traces continuously. Hold down mouse button and drag.
- **Shortest Line Holder:** Container holding 3DLine to show the shortest route.
- **Fastest Line Holder:** Container holding 3DLine to show the fastest route.
- **Picking Threshold (px):** Determines the distance (in pixels) from line for picking to capture.
- **Max Junctions:** Determines the maximum number of junctions to pass when calculating shortest route before giving up (due to performance).
- **Set Data Pool:** Sends output to DataPool.
- **Use Only TMC:** Uses TomTom shape file data only.
- **Shared Memory Type:** Selects the shared memory area to update. Click either None, Global, Scene, or Distributed.
- **Field Name Prefix:** Determines the shared memory field name to use with DataPool.
- **Start Building Route:** Starts building (and recording) a route with many clicks for better control on the selected route.
- **Pause Building Route:** Pauses building (for example, to move the map) the route.
- **Continue Building Route:** Continues building the route.
- **End Building Route:** Stops building the route.
- **Rebuild from Saved Data:** Rebuilds the line form the data saved in the command box above.
- **Clear:** Clears the contents of the line tracer.
- **Picking Active:** Specifies the place on the screen where can routes can be selected. There are three options:
  - **All:** Allows selecting routes anywhere on the screen.
  - **Inside Rectangle:** Allows selecting routes anywhere inside the rectangle.
  - **Outside Rectangle:** Allows selecting routes anywhere outside the rectangle.

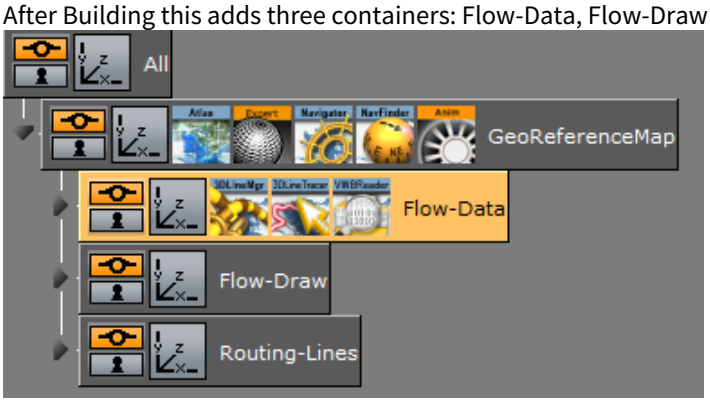
How to Create a Scene Showing Flow and Routing

1. Create a base map using Atlas, add a Navigator plug-in.

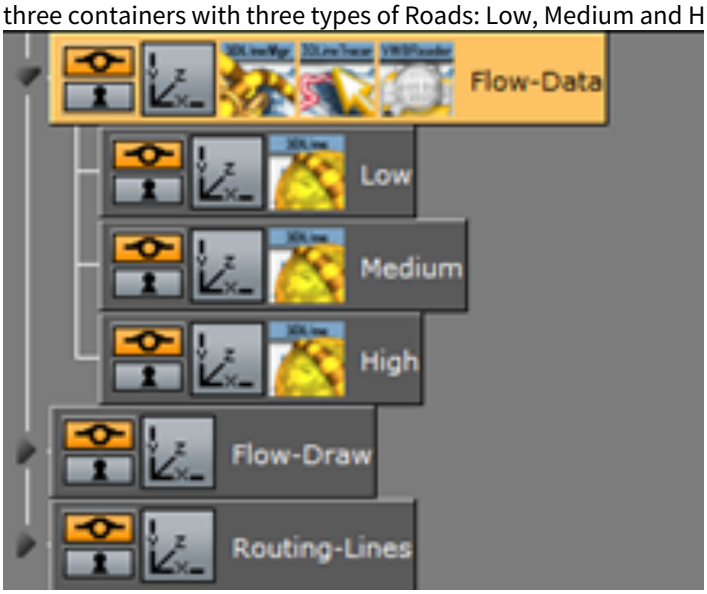
Copyright © 2026 Vizrt



2. Add a 3DLineManager to the Hierarchy. Set the Data Source to be Flow and Routing, Load the correct Shape File (TomTom TMC Data) and Click **Build Flow Hierarchy**, Viz builds the data for Routing.
3. After Building this adds three containers: Flow-Data, Flow-Draw and Routing-Lines.

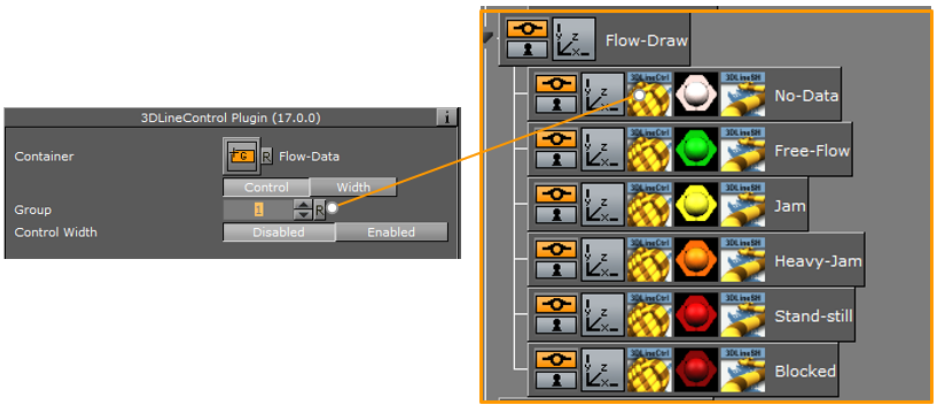


4. **Flow - Data:** The 3DLine Tracer plug-in and the VWBReader are added to the container. Underneath it adds three containers with three types of Roads: Low, Medium and High.



5. In 3DLineManager, you can set the types of Roads that enter in each group, you can change the Roads according to your selection.
6. In the Flow-Draw container: This are the painters that draw the data.

Copyright © 2026 Vizrt



7. When there is no data, all the lines are in Group number 1 and has a light pink color.



How to Load the Traffic Data

1. In VWBReader, change the data type to `Custom`, Load the VWB File (file that is created with the Feedstreemer -this file is updated constantly with the traffic) and Click **Build**.



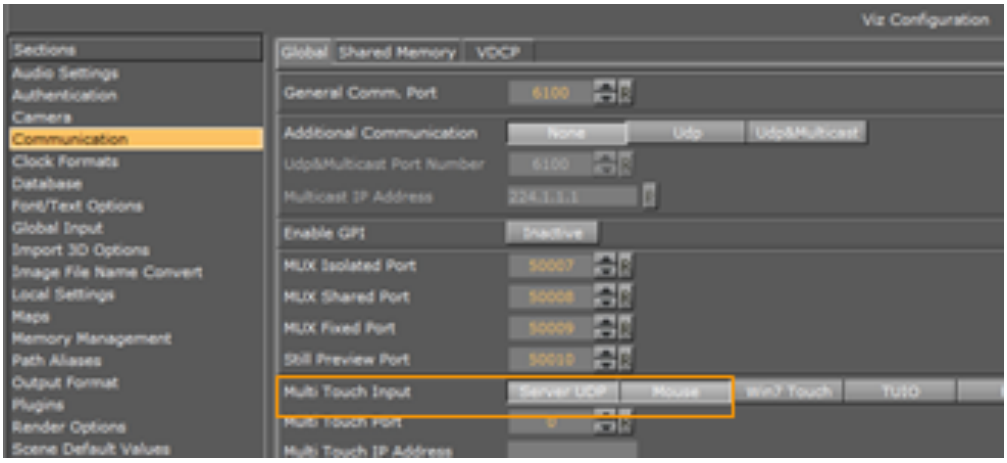
2. It draws all the data: Free flow to green, Jam to Yellow, Heavy Jam to Orange ...

Copyright © 2026 Vizrt

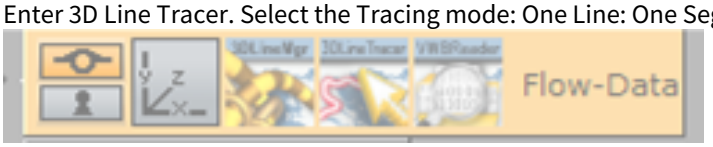


How to Trace the Lines Using the 3D Line Tracer Plug-in

1. To make a selection with the mouse the Viz configuration should be **Communication - Multi Touch input - Mouse**.

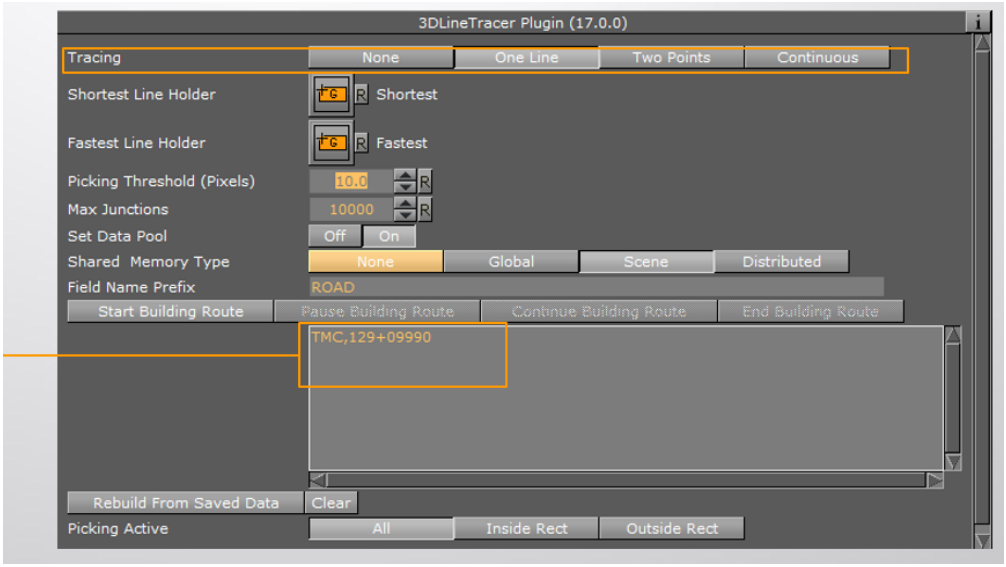


2. Enter 3D Line Tracer. Select the Tracing mode: One Line: One Segment.



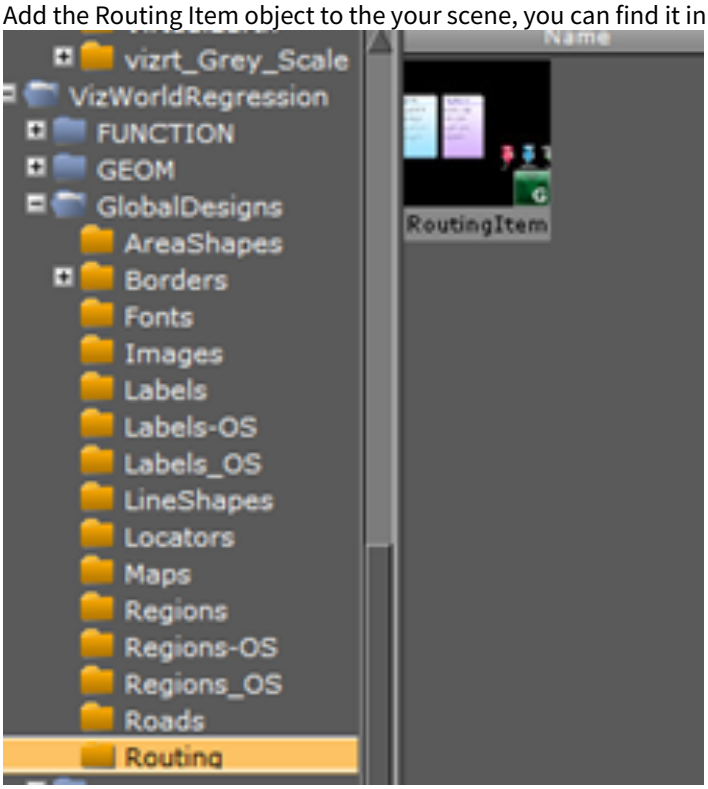
3. When one of the segments is selected you can see the name and ID number.

Copyright © 2026 Vizrt



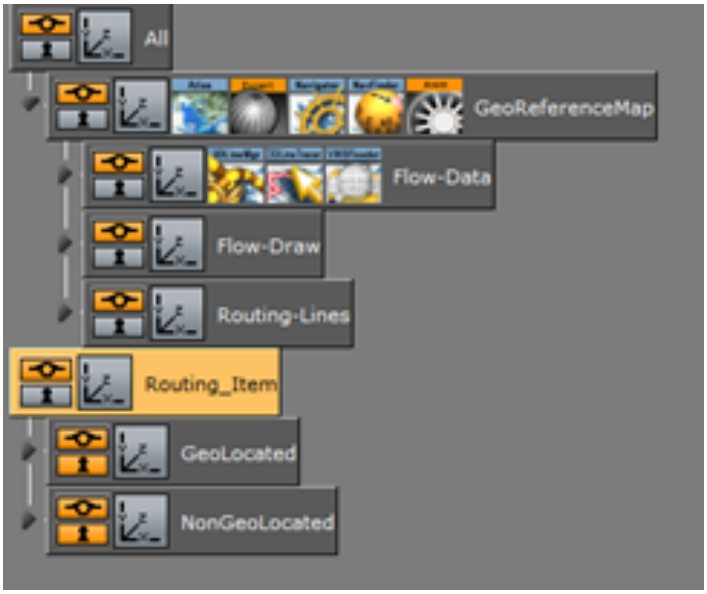
How to Show the Route Information

1. Add the Routing Item object to the your scene, you can find it in the server under: Global Designs.

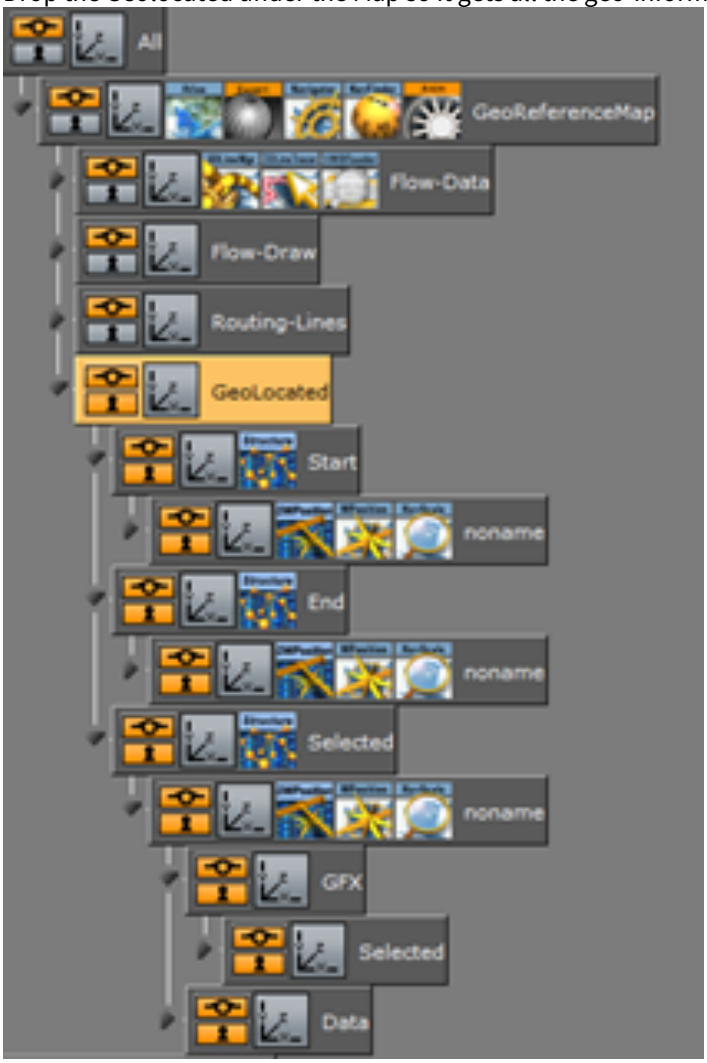


2. Routing folder. You find two Routing items: *GeoLocated* and *Non GeoLocated*.

Copyright © 2026 Vizrt



3. Drop the Geolocated under the Map so it gets all the geo-information.



Copyright © 2026 Vizrt

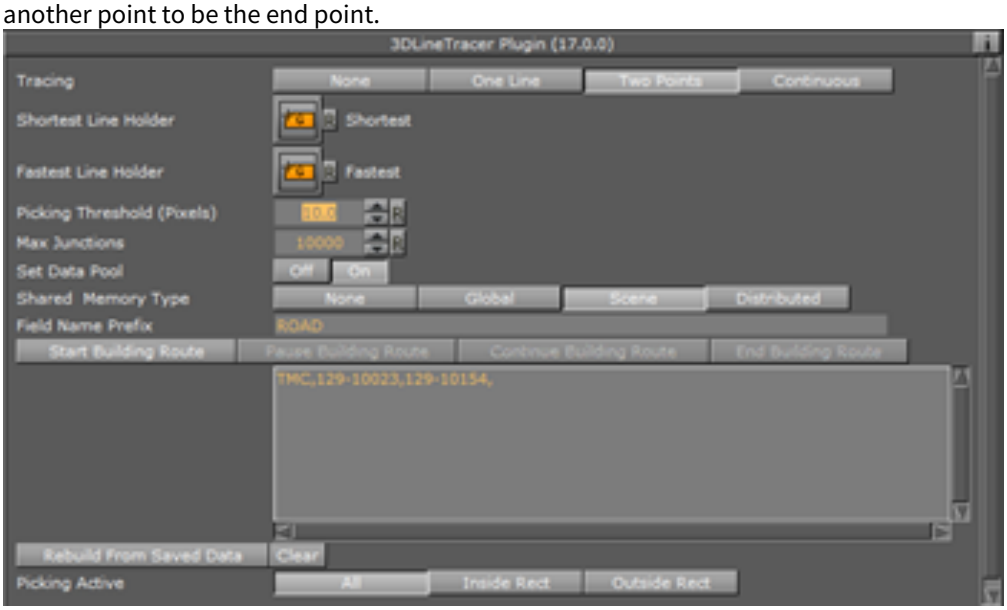
Works with Datapool plug-ins.

4. By selecting a segment, it updates the Shortest panel showing the distance, the time, the speed and the flow type.



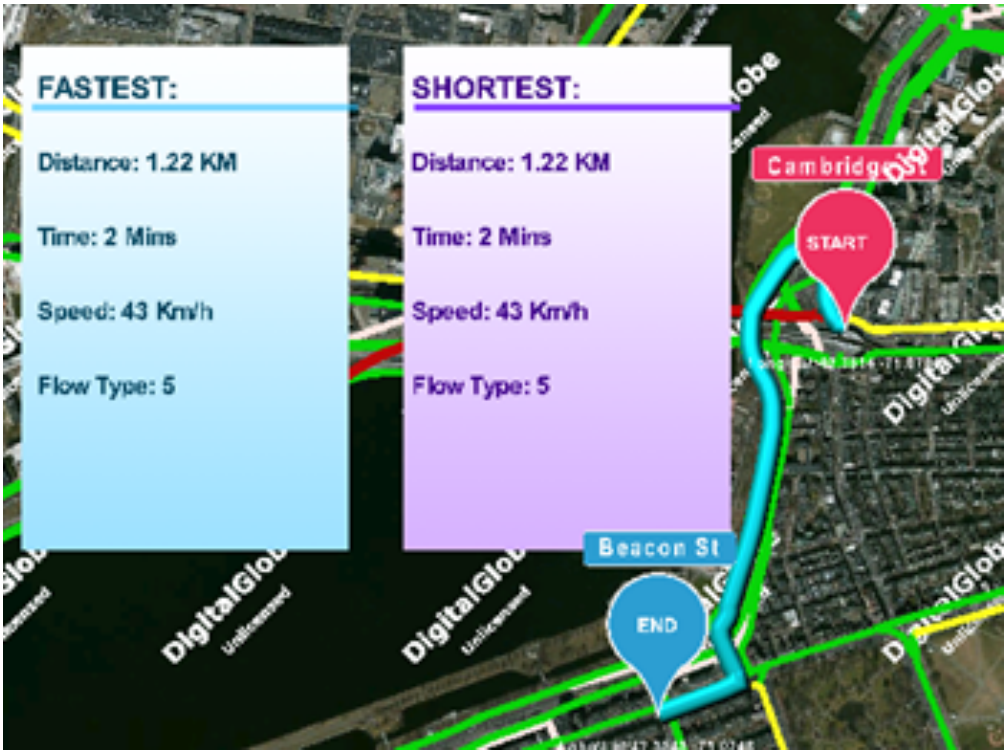
Tracing Two Points

1. Change the Tracing to `Two Points`, select one Point on the Map. That is the start point and select another point to be the end point.



2. It gives you two routes: One Fastest and another Shortest.

Copyright © 2026 Vizrt



3. The information for all the TMC points in a route is exposed via shared memory, using the shared memory keys *ROAD_ROUTE_DATA_SHORTEST* and *ROAD_ROUTE_DATA_FASTEST*.

The value for each TMC point has the following format:

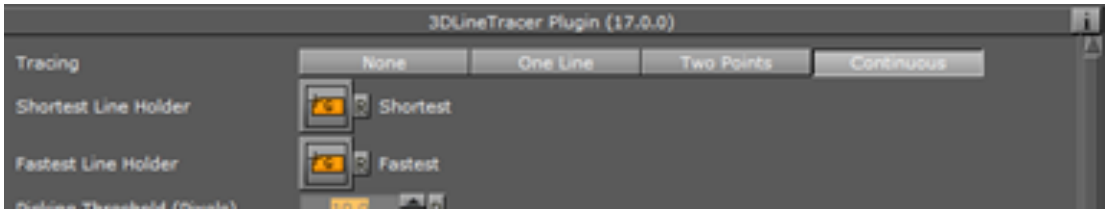
TMC id	time (minutes)	distance (meters)	speed (meters/minute)

As an example, a route could expose the following key, containing three TMC points (separated by pipes):

ROAD_ROUTE_DATA_SHORTEST	116+04106,0.332059,498.057,1499.9| 116+04107,0.296999,495.02,1666.74| 116+04107,0.207084,345.155,1666.74

Continuous Tracing

You can use this option to select a Route by dragging and hold the mouse.

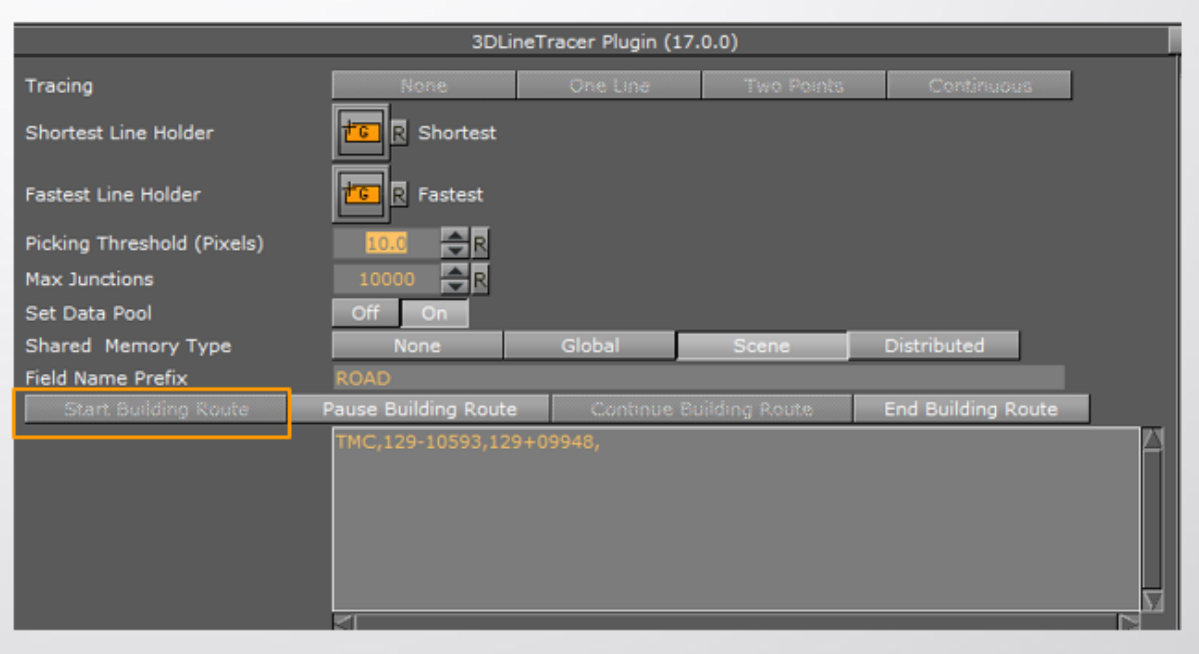


Copyright © 2026 Vizrt

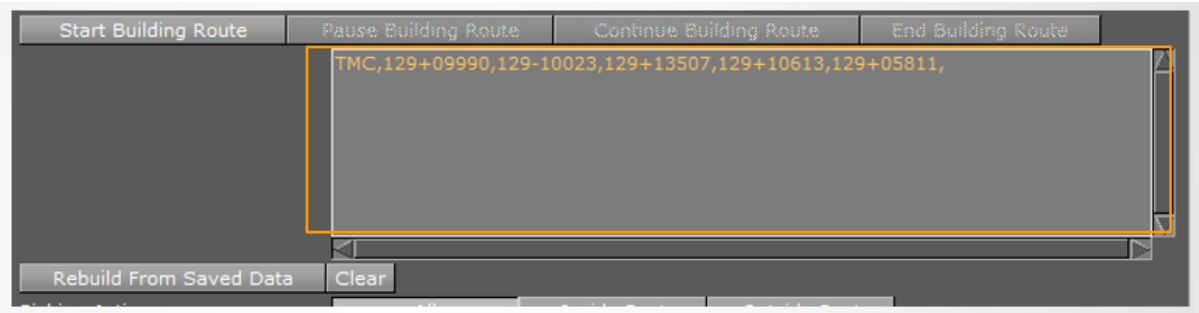
How to Build a Specific Route

Click **Start Building Route**, select the Route in the Map.

> **Note:** Every click selects a segment of the Road.



Here you can see the route information. It can be saved into a text document and then you can send the information. The **Clear** option is to Clear from the Scene and the **Rebuild** option rebuilds the route from saved data.



The **Picking Active** option specifies the place on the screen where can routes can be selected. There are three options:

- **All:** All Screen.
- **Inside Rect:** Inside the Rectangle.
- **Outside:** Outside the Rectangle.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

5.4.4 3D Map Telestrator



The 3D Map Telestrator plug-in enables you to draw strokes on maps (including globes) and arbitrary flat objects using screen input interface like a mouse or touch screen. The strokes, whether with geo or non-geo coordinates, may be passed to other scenes on distant computers through the mechanism of shared memory. 3D Map Telestrator uses 3D Line for strokes visualization and thus it benefits from the rich design capabilities of this plug-in. The obtained strokes may be smoothed to an arbitrary degree to obtain better looking final result of the drawing.

This section contains information on the following topics:

- 3D Map Telestrator Properties
  - General
  - Shape Recognition
  - Examples
- Creating a 3D Map Telestrator Scene
  - To Create a 3D Map Telestrator Scene
- Non Geo-Reference Telestration
- Shared Memory Mechanism
- Using Multitouch Interface
- Using Perceptive Pixel Interface
- Brush Design Using 3D Line
  - Width
  - Outline
  - Effect
  - Advanced

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

3D Map Telestrator Properties

General

- **Camera:** Defines working camera to accept screen input (which is useful in a scene with layers where more than one camera is used).
- **Line Pixel Resolution:** Defines the required resolution of line during drawing, such that for example if resolution of `5` is set strokes less than five pixel size is disregarded.
- **Designs Holder:** Searches beneath the selected container for the required design.
- **Design Name:** Selects the design to implement in the scene. A couple of designs may be implemented in the scene and here the desired one for current use is set. If the required design does not exist, the first one found is used.
- **Target:** Defines where to build line strokes. If the Container option is used, the desired container should be specified in **Target Container** section.
- **Active:** Defines when the plug-in is active. It may be switched off, active always or only be activated while pressing **P** button on the keyboard. This is useful when other interactive capabilities are used in the scene (for example interactive navigation) and care should be taken not to interfere with these modes.

Copyright © 2026 Vizrt

- **Spline:** Smoothens the final result for a better look. The three spline types define three different methods for smoothing.
    - **Control Points:** Uses a spline that paths between the points of the original curve. This results in the smooth nice curve, but may be not accurate enough since the line doesn't actually follow the original path and may deviate from it.
    - **Curve Points:** Makes the spline path through points of interest of the original curves. This method may result in some artifacts causing the route to deform between points of interest due to the constraint that the curve must be smooth.
    - **Combined:** Compromises between Control and Curve points (see Examples).
- **Smoothness:** Allows the smoothness to define the threshold if spline is used in order to obtain smooth results. The higher the threshold the softer the curve is (see the examples at the end of this section).
- **Close Stroke:** Closes strokes automatically when set to `On`.
- **Control Mode:** Evokes a shared memory mechanism to send and receive strokes to and from other 3D Map Telestrator plug-ins when set to `On`.
- **Shared Memory:** Defines the type of shared memory:
    - **Global:** Shares memory within the region of computer.
    - **Scene:** Shares memory within the region of current scene only.
    - **Distributed:** Shares memory between all the scenes and computers in the network.
- **Shared Memory Name:** Defines the name for connection, such that only 3D Map Telestrators with the same name shares data between them.
- **Input Interface:** Selects interface type for line creation. See further remarks about PPI and Multitouch use.
- **Animation Control:** Runs animation that is located on design container after line is created (mouse button released).
- **Build on Load:** Determines whether the lines created when the scene was saved are built when the scene is loaded.
- **Clear/Undo/Redo:** Clears the whole drawing or undoes/redoes last strokes.

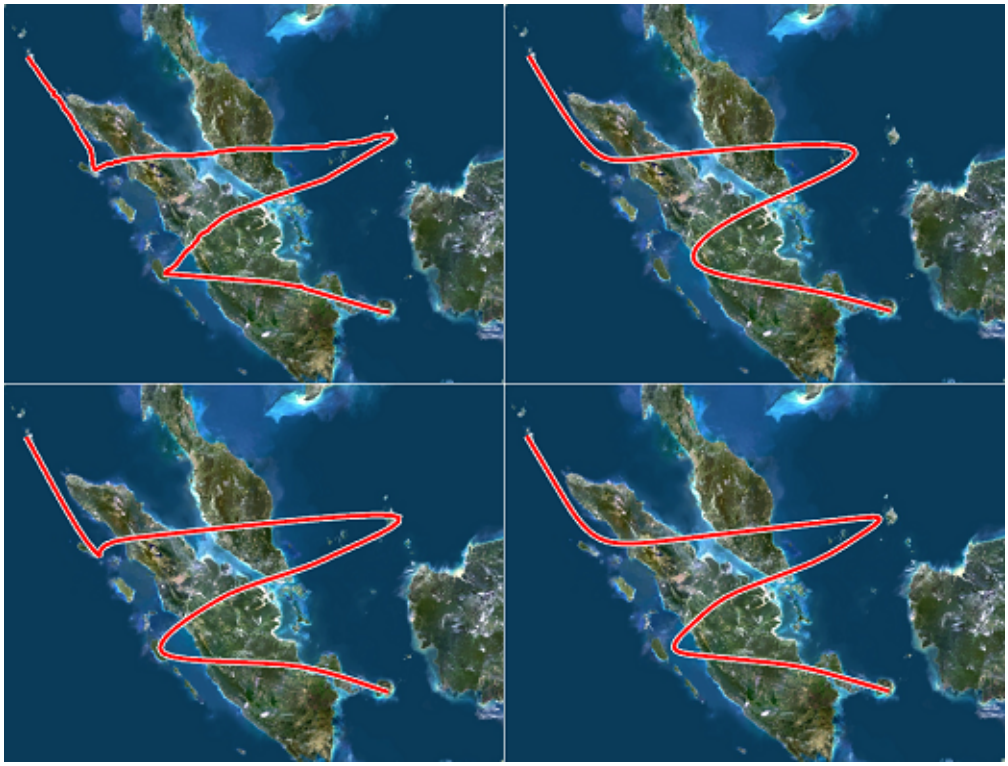
Shape Recognition

- **Build on Load:** Determines whether the lines created when the scene was saved are built when the scene is loaded.
- **Recognize Ellipse, Square, Triangle and Line:** Recognizes four shapes: ellipse (including circle), square, triangle and line. When the plug-in recognizes a shape it rebuilds it to be of ideal shape (a hand-drawn circle is replaced by a perfect circle of the same size). The **Try Harder** option means that there is more probability to recognize the requested shape (for example the circle may be less ideal and still be recognized) running the risk that it might replace shapes that are not (for example, circles). When **Try Harder** is enabled only one shape can be recognized as opposed to the **Try** option which enables the plug-in to recognize multiple shapes simultaneously.

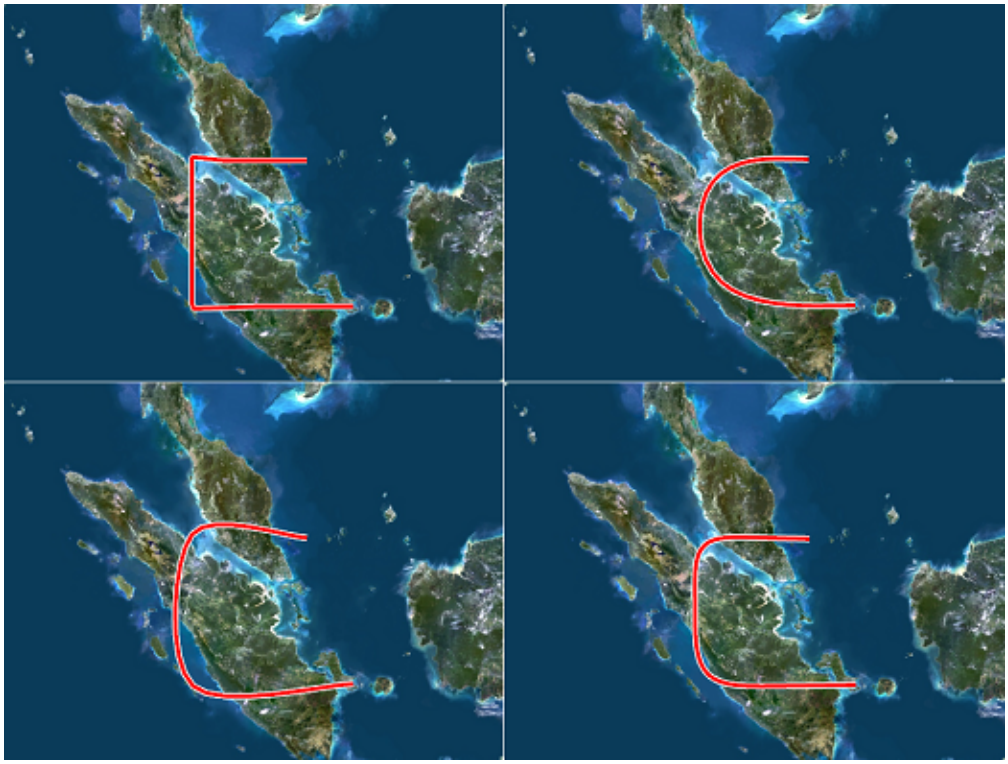
Examples

Influence of the **curves** parameter set to original, control points, curve points and combined:

Copyright © 2026 Vizrt



Influence of the **curves** parameter set to original, control points, curve points and combined:



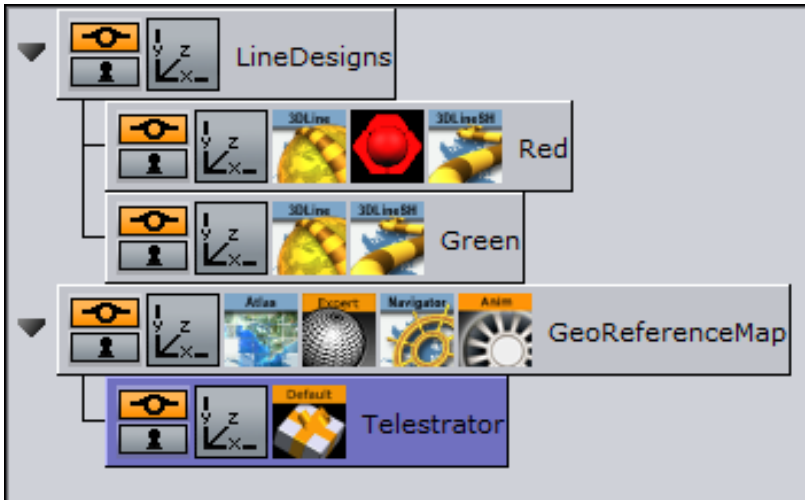
Copyright © 2026 Vizrt

Influence of the **smoothness** parameter set to original, low, medium and high:



Creating a 3D Map Telestrator Scene

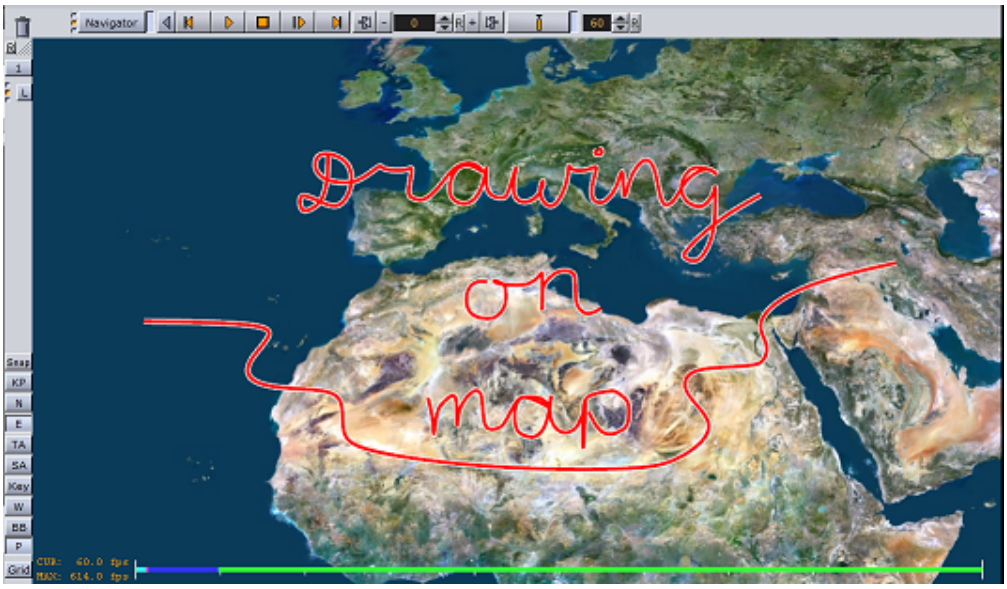
To Create a 3D Map Telestrator Scene



1. Prepare a blank workspace for the new scene, go to **Geom Plugins > Maps** and drag Atlas to the scene.
2. Open the Atlas editor, and in under **Data Source** switch to **CMR** (Curious Multi Resolution), and in the **CMR File** locator locate the CMR with the desired map. This creates a map to draw on. If you have an Internet connection and license for Microsoft Bing maps you see the world map as soon as you drop Atlas and there is no need to load CMR.

Copyright © 2026 Vizrt

3. Go to **Container Plugins -> Maps** and drop Navigator onto the same container as Atlas. Remember to enable interactivity by pressing the **E** button (to the left of the scene editor). Navigation through the map is possible by pressing **i** button and using the mouse. If the map suddenly disappears after dropping Navigator, click the **Center Map** button on the bottom of it.
4. Add a container above the one just prepared and name it **LineDesigns.**
5. Add a new container as a **child of LineDesigns** and name it **Red.**
6. Add 3DLine to the Red container.
7. Open the 3D Line editor and set the **Width** parameter to **Fixed (Pixels).**
8. Set the Width's sliding parameter to `4`.
9. Switch to the Outline option on the topmost radio button row and set **Outline** to **On** and give it a width of `50` (Outline Width (%) parameter).
10. Add material to the container and make it red. This creates the first design for the line.
11. Add another container beneath the Red container and name it **Green.**
12. Add 3DLine to the Green container.
13. Open the 3D Line editor and set the **Width** parameter to **Fixed (Pixels).**
14. Add a material with a green color to the container. This defines the second design for the line named *Green*.
15. Add a new container as a **child of the Atlas** container and name it **Telestrator** and add 3D Map Telestrator to it.
16. Open the 3D Map Telestrator editor and drag the **LineDesigns** container onto the **Designs Holder placeholder.**
17. Set the **Design Name** parameter to **Red** (in order to choose red design).
18. Set the **Active** parameter to the **On P** position.
19. Set **Shared Memory** parameter to **Distributed.**
20. Save the scene.



You can draw on the map by pressing the **P** key on the keyboard while holding the left mouse button down. If you open the same scene on another computer in the network, you should see the same drawings appearing on it too.

Non Geo-Reference Telestration

In the procedure on how to create a 3D map telestrator scene, the drawing was done on the map (the path has geographic coordinates and as such, it is placed on the same geographical route when sent to other computers via

Copyright © 2026 Vizrt

the Shared Memory Mechanism). The other map in the other scene may have different style and projection. It may also have globe geometry, and the path is still geographically correct.

There is an option to use 3D Map Telestrator without map and geographic coordinates. To use this option two conditions must be met:

- The plug-in must not be located under geo-reference (in the example scene the Atlas was used as geo-reference).
- The flat geometry about to be drawn on must be placed on the same container with 3D Map Telestrator.

Shared Memory Mechanism

As explained in the 3D Map Telestrator Properties section, the plug-in uses a shared memory mechanism in order to send and receive strokes to and from other 3D Map Telestrator plug-ins. For the two plug-ins to connect, they should have the same name for the shared memory data they are listening to. The given name should have a prefix of `Telestrator_`, such that when a user enters the name *alpha* for the group of 3D Map Telestrator plug-ins, the actual name of the shared memory parameter is: `Telestrator_alpha`.

The format of data consists of a unique ID that each stroke receives, name of required design and data path that comes as a sequence of pairs of numbers representing longitude and latitude (when non-geo-reference mode is used the flat geometry's bounding box is treated as if having the lat/long coordinates of full world). All components of the data including design name and unique ID are separated by commas.

Using Multitouch Interface

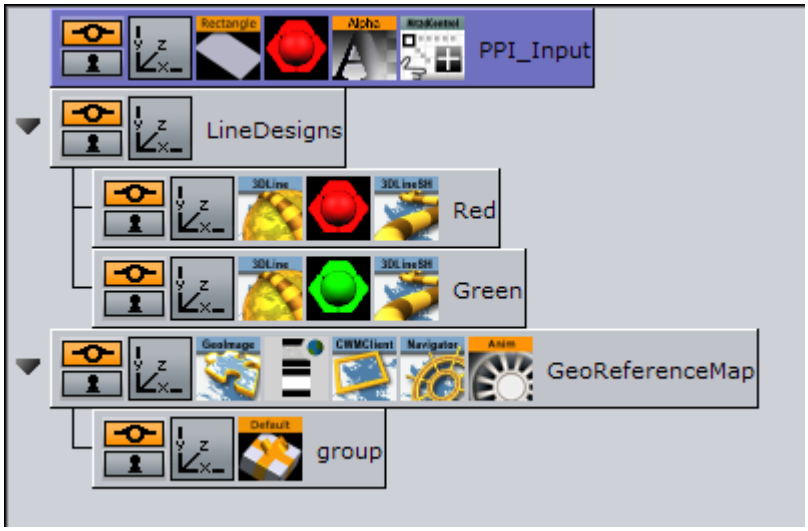
This interface uses touch screen of Multitouch type. For the plug-in to receive information from the screen, the Input Interface button should be switched to Multitouch mode and a container with the plug-in should contain a geometry. Screen touches outside of this geometry are disregarded. If the plug-in is used in a non-georeference mode, there is no problem since it already contains geometry for drawing on. If it is used in a georeference mode, the best practice is to place 3D Map Telestrator on the container with the map.

Using Perceptive Pixel Interface

In a case of using a Perceptive Pixel Interface (PPI) screen, the setup requires a couple of additional things. A special plug-in named Mt2dControl, that is a part of the PPI plug-in package, accepts and processes the inputs from the touch screen and the processed data is sent to 3D Map Telestrator through a shared memory mechanism.

Continuing the procedure above on how to create a 3D map telestrator scene, you can add multi-touch capabilities to your graphics scene. The procedure and example scene tree below expands on the aforementioned procedure.

Copyright © 2026 Vizrt



1. Open the 3D Map Telestrator editor and set **Input Interface** to **PPI** (see Properties) and chose a name for a shared memory connection with the PPI interface (a PPI variable name). Note that this must differ from shared memory connections with other 3D Map Telestrator plug-ins.
2. Add the container somewhere in the scene which is not a part of any hierarchy.
3. Add a rectangle to the container and scale it to the maximum possible size.
4. Add Mute on the container to avoid rendering it.
5. Add Mt2dControl on the container.
6. Open the Mt2dControl editor and set the following parameters:
    - **Lock Rotation** to `On`.
    - **Lock Scale** to `On`.
    - **Shared Memory** to `On`.
    - **Shared Memory Type** to `Scene`.
    - **Shared Memory Prefix** should be the same as the PPI variable name in 3DMapTelestrator.
    - **Hit Coordinate Type** to `World`.
7. **Save** the scene. Now you can draw lines by touching the screen.

Brush Design Using 3D Line

3DLine is used by the 3D Map Telestrator to visualize the strokes. The desired look of the line is set by various parameters in the design (inside 3D Line) and it is important to understand these parameters to get the desired look and behavior for the line.

The following are the most important parameters:

Width

The options for the **Width** tab defines the width behavior of the line. Three options may be used to define width behavior:

- **Fixed (pixels):** Defines constant width of line measured in pixels.
- **Scaling:** Varies line width according to zoom and the minimum/maximum parameters set.
- **Fixed (mm):** Sets a fixed width in geographic units.

Copyright © 2026 Vizrt

- **Fade Edge (%):** Applies a fade effect to the side-edges.

Outline

By using these options an outline of arbitrary width, fade amount and color can be set to a line.

Effect

When a line comes to 3D Map Telestrator, it plays an animation running from single point to a full path by default. The attributes of such an animation can be set beneath this tab. The length of animation in time is controlled by the Animation Length parameter and if it is set to `0`, no animation is played. A fade effect to a running edge of the line can be set by the Fade parameter.

Advanced

A couple of options are available for this tab, the most important of which are:

- **Cap edges:** Defines shape for the beginning and end edge of the curve.
- **Height Offset:** Defines an offset line from surface it is put on.
- **Update Texture Mapping:** Updates line width during zooms according to the setting we made in Width tab, while line length remains unchanged (texture mapped on the line may deform). To keep same aspect ratio for the texture this option must be enabled.

See Also

- 3D Map Telestrator Design
- Using Multitouch Interface

Copyright © 2026 Vizrt

5.4.5 3D Map Telestrator Design



The 3D Map Telestrator Design plug-in controls 3D Map Telestrator options through design setup. When this plug-in is added to the design its setup is imposed on the 3D Map Telestrator (parameters set in 3D Map Telestrator Design plug-in overrides parameters set for the 3D Map Telestrator), and thus it is possible to control spline type and smoothness, close option and shape recognition setup from a design perspective.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

3D Map Telestrator Design Properties

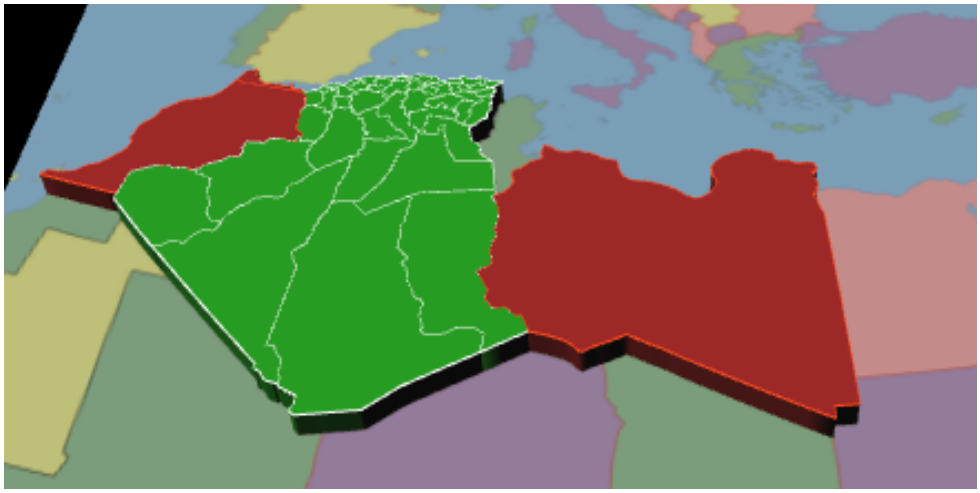
- **Smoothness:** Allows the smoothness to define the threshold if spline is used in order to obtain smooth results. The higher the threshold the softer the curve is (see the examples in the 3D Map Telestrator Properties).
- **Close Stroke:** Closes strokes automatically when set to `On`.
- **Recognize Ellipse, Square, Triangle and Line:** Recognizes four shapes: Ellipse (including circle), square, triangle and line. When the plug-in recognizes a shape it rebuilds it to the ideal shape (i.e. a hand-drawn circle is replaced by a perfect circle of the same size). The **Try Harder** option means that there is more probability to recognize the requested shape (for example the circle may be less ideal and still be recognized) running the risk that it might replace shapes that are not e.g. circles. When **Try Harder** is enabled, only one shape can be recognized as opposed to the **Try** option which enables the plug-in to recognize multiple shapes simultaneously.

Copyright © 2026 Vizrt

5.4.6 3D Region Manager



The 3D Region Manager plug-in creates sub regions of a selected region on the map, using a 3D Region design. The plug-in receives the region name and gets all the sub regions of that region from the server or a shape file. The 3D Region design is duplicated for each sub region and displayed on the map.



Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Man

3D Region Manager Properties

Common

- **Design:** Sets the region design container that is used for building sub-regions. The design container should be built with a 3D Region object and a material. No special naming convention is required.
- **Target:** Defines the container to be used as the parent container for the sub regions containers:
  - **Me:** Builds the 3D Region objects under the current container (holding the 3D Region Manager plug-in).
  - **Next:** Builds the 3D Region objects under the next container (next container in the tree and at the same level as the 3D Region Manager container).
  - **Down:** Builds the 3D Region objects under the first child container.
  - **Container:** Builds the 3D Region objects under the container dragged into the **Target** container place holder.

Build Region

- **Build Region:** Defines the source for which the 3D Region objects are created.
  - **Local:** Builds the 3D Region objects for the container that the 3D Region Manager is applied to. The container must be a 3D Region object for the sub regions to be built for the defined region.

Copyright © 2026 Vizrt

- **WME:** Enables the user to select the regions or sub-regions inside the WME.
- **Shape File:** Allows the user to select a shape file that contains the regions to be created.
- **Command:** Allows the user to set a command with the parameters of the regions to build.
- **Region ID:** Enables the user to show a specific region by region ID after having prepared the cache files.

Local

- **3D Border Visible Filter:** Sets which borders should be cropped to map. Any border lower or equal to the selection is cropped to the map. Available options are Country, Region and None.

WME

- **3D Border Visible Filter:** Sets which borders should be cropped to map. Any border lower or equal to the selection is cropped to the map. Available options are Country, Region and None.
- **Scan:** Defines what regions to display.
  - **Selected:** Displays only the selected regions, selected by pressing the **Select Region** button.
  - **Recursive:** Displays all selected regions and their sub regions.
- **Select Region (WME):** Opens the World Map Editor, enabling the selection of regions to be used in Selection mode. Press the **Select Region** button to select the regions to be built.
- **Select Region (Classic):** Opens the Map Editor Classic, enabling the selection of regions to be used in Selection mode.

Shape File

- **Shape Regions:** Defines how the regions are built. Available options are Merge, Split and Advanced.
  - **Merge:** Creates all regions in the file as one object.
  - **Split:** Creates all regions in the file as separate objects.
  - **Advanced:** Enables the additional parameters Conversion File, State Column and Name Column. **Conversion File** defines a text file for converting regions indexes into region names. **Parent Id Column**, **Parent Name Column** and **Names Column** refer to columns in the database (**.dbf*) file that comes with the shape file. The database files describes what is attached on the shape file. Note that database files can be opened with Microsoft Office Excel.
- **Shape File:** Defines a path to the shape file (**.shp*), containing the region definitions. Shape files are bought from vendors specialized in Geographic Information System (GIS) and hold the actual shape data (polygons, splines, and others) for the container that the 3D Region Manager is applied to. The container must be a 3D Region object for the sub-regions to be built for the defined region. Note that shape files must be stored in individual folders.
- **Conversion File:** Determines files used to convert IDs in the file to Viz World IDs.
- **Parent ID/Name Column:** Determines which column which holds the ID/Name of the parent of the region.
- **Parent Name Length:** Truncates the parent name (`0` = no truncation).
- **IDs/Names Column:** Determines which column which holds the ID/Name of the region.
- **Prepare Cache Files:** Minimizes memory use by pre-creating of the cache files and then building the required regions only (directly from the cache files), instead of building large amounts of regions from a shape file (which may overload the system). Only prepare Cache Files (without loading the data to Viz) the files can be used later in Viz, based on the ID.

Copyright © 2026 Vizrt

Command

- **Command:** Builds a 3D Region object from the specified command. The command defines the region coordinates using pairs of longitude and latitude values. The command format is as follows:

```
<3D RegionContainerName>: Long1,Lat1 Long2,Lat2...
```

Region ID

- **Shape File:** Defines a path to the shape file (*.*shp*), containing the region definitions. Shape files are bought from vendors specialized in Geographic Information System (GIS) and hold the actual shape data; polygons, splines, and others for the container that the 3D Region Manager is applied to. The container must be a 3D Region object for the sub regions to be built for the defined region. Note that shape files must be stored in individual folders.
- **Region ID:** Sets the region's ID for the required region.

Common

- **Design:** Sets the region design container that is used for building sub -regions. The design container should be built with a 3D Region object and a material. No special naming convention is required.
- **Target:** Defines the container to be used as the parent container for the sub regions containers:
  - **Me:** Builds the 3D Region objects under the current container (holding 3D Region Manager).
  - **Next:** Builds the 3D Region objects under the next container (next container in the tree and at the same level as the 3D Region Manager container).
  - **Down:** Builds the 3D Region objects under the first child container.
  - **Container:** Builds the 3D Region objects under the container dragged into the **Target** container place holder.

Buttons

- **Clear and Build:** Deletes all previously built objects and rebuilds the regions according to the current settings.
- **Add to Existing:** Builds the new objects without deleting the old 3D Region objects from the hierarchy.
- **Clear Target Tree:** Removes all objects built by the plug-in from the Viz scene hierarchy.
- **Freeze All:** Freezes the regions to the scene instead of recreating them on load.

Copyright © 2026 Vizrt

5.4.7 3D Roads Manager



The 3D Roads Manager plug-in creates 3D Roads objects.

The plug-in uses a shape file design to create roads according to the defined settings.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Man

3D Roads Manager Properties

- **Roads Mode**
  - **Global:** Applies one set of road designs for all roads.
  - **Design:** Uses a design per CWMClient. The **Design** field sets the roads design container that is used for building border data from a shape file. The design container should be built with a 3D Roads object and a material. No special naming convention is required.
- **Target:** Defines the container to be used as the parent container for the road designs when Roads Mode is set to Design.
  - **Me:** Builds the 3D Roads objects under the current container (holding the 3D Roads Manager plug-in).
  - **Next:** Builds the 3D Roads objects under the next container (next container in the tree and at the same level as the 3D Roads Manager container).
  - **Down:** Builds the 3D Roads objects under the first child container.
  - **Container:** Builds the 3D Roads objects under the container dragged into the Target container place holder. The **Target Container** specifies the container that holds all the 3D Roads objects.
- **Source:** Selects street/road source.
  - **WME:** Uses Viz World Map Editor (WME) using the CWM client plug-in to connect to the map server.
  - **Shape File:** Determines the shape file that contains street data to use.
  - **Cloud Made:** Loads street data from Cloud Made. This is a web service which requires a license.
  - **Data Set:** Loads street data from the converted data supplied by an external provider into Viz World format

WME

- **Simplify Threshold:** Sets the detail reduction factor for the shape of the roads.
- **Select Streets (WME):** Opens the World Map Editor.
- **Select Streets (Classic):** Opens Map Editor Classic.
- **Labels Mode:** Determines how labels are created.
  - **None:** Does not create labels.
  - **Partial:** Creates labels only visible on the view.
  - **All:** Creates all labels.
- **Labels Only:** Creates labels only, not the roads when set to `On`.

Cloud Made

- **Simplify Threshold:** Sets the detail reduction factor for the shape of the roads.
- **Labels Mode:** Determines how labels are created.

Copyright © 2026 Vizrt

    - **None:** Does not create labels.
    - **Partial:** Creates labels only visible on the view.
    - **All:** Creates all labels.
- **Labels Only:** Creates only labels, but not roads when set to `On.`

Data Set

- **Data Set:** Location of data
- **Simplify Threshold:** Sets the detail reduction factor for the shape of the roads.
- **Labels Mode:** Determines how labels are created.
    - **None:** Does not create labels.
    - **Partial:** Creates labels only visible on the view.
    - **All:** Creates all labels.
    - **Label & Go:** Uses Label and Go.
- **Labels Only:** Creates only labels, but not roads when set to `On`.

Shape File

- **Shape File:** Defines a path to the shape file (**.shp*), containing the border definitions. Shape files are bought from vendors specialized in Geographic Information System (GIS) and holds the actual shape data; polygons, splines, and others. Note that shape files must be stored in individual folders.
- **Sort Roads By:** Sets the loaded roads from the shape file in a category.
    - **Category:** Enables the user to select one predefined category of street data. Available **Road Types** are **Freeways**, **Primary**, **Main**, **Secondary** and **Other**.
    - **Advanced:** Splits the data into different categories using specific string types. **Road Type Column** specifies a column name that holds each road category. **Freeways/Motorways**, **Primary Roads**, **Main Roads**, **Secondary Roads** and **Other Roads** specify what string in the database file (**.dbf*) that matches the road type.
- **Persistent Roads:** Defines whether the roads data is removed from Viz memory when the scene is closed or not.
- **Sort Roads by:** Defines which road type is associated with created objects. If Advanced is selected, additional parameters are enabled allowing the configuration of road types according to the data associated to the shape file.
    - **Category:** Enables the selection of road types. Available options are: Freeways, Primary Main, Secondary, Other.
    - **Advanced:** Sets parameters according to the data in the files associated with the shape file. Defines the road type column and the specific road type ID in the file.
- **Road Name/ID Column:** Determines which column which holds the Name/ID of the road.

Buttons

- **Rebuild All:** Rebuilds all data. If Roads mode is set to Design, 3D Roads objects are created from design. If Roads mode is set to Global, global 3D Roads renders built data.
- **Rebuild All (force):** Deletes any cache and force a rebuild of the cache as well as rebuilding the Viz geometry.

Copyright © 2026 Vizrt

- **Rebuild Labels:** Rebuilds only the labels, but does not affect already built streets.
- **Clean All:** Cleans Target container from subcontainers.

Copyright © 2026 Vizrt

5.4.8 Center Map



The Center Map plug-in is used with map objects to keep a point on the map in the center of the map object. The map is moved while the defined point remains centered. With a Flat map, the map is moved, but with a Globe map the globe is rotated. Also see Map Tiler for more information on Flat and Globe maps.

Note: The Center Map plug-in should only be applied to map objects.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

Center Map Properties

- **Position:** Defines how the map is centered.
  - **User:** Enables the user to manually define a center point by setting *Map Center Longitude* and *Map Center Latitude* values.
  - **WPosition:** Centers the map at the location of a *WPosition* container dragged into the *Remote Container* place holder.
  - **Navigator:** Centers the map at the location of a *Navigator* container dragged into the *Remote Container* place holder.
  - **WPoint:** Centers the map at the location of a *WPoint* container dragged into the *Remote Container* place holder.
  - **Map:** Centers the map at the center of a dragged map into the *Remote Container* map.
  - **Frame:**
- **Remote Container:** When Position is set to WPosition, you must drag the container that holds World Position to this field to get the information.
- **Map Center Longitude/Latitude:** Defines information for when Position is set to User.
- **Local Scale:** Sets the scale parameter for the map.
- **Animate to Target:** Animates the map whenever the longitude/latitude value changes when enabled. This is either with a fixed defined time or using length time, it is relative to the distance it has to travel to the new value.
- **Animation Time:** Target value (in fields) for the animation.
- **Shared Memory:** Writes the longitude/latitude values to the shared memory variable defined in the Identifier field when set to `Master`. When set to `Client` it listens for values in the shared memory key.
- **Type:** Determines the type of shared memory to use (for more information on Shared Memory see the Viz Artist User Guide):
  - **Global:** Scene.Map: This is the map local to the current Scene. Every scene has one map that can be used to exchange data among the scripts in the Scene.
  - **Scene:** System.Map: The system-wide map allows for data sharing among the scenes currently loaded into memory.
  - **Distributed:** VizCommunication.Map: A distributed map that enables data sharing among the computers connected to one Graphic Hub.
- **Identifier:** Identifier to be used for shared memory communication.

Copyright © 2026 Vizrt

5.4.9 CWM Client



The CWMClient plug-in is the main plug-in for Viz World Client (WoC). The CWMClient plug-in launches the World Map Editor (WME) and retrieves the map from the Viz World Server (WoS) after the user has applied the changes to the map in WME.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

CWMClient Properties

Texture

The texture section defines the geo map image parameters.

- **Lock Aspect Ratio:** Affects texture height and texture width parameters. When enabled (`On`) any change in the texture height or width affects both parameters. When disabled (`Off`), each parameter is controlled separately.
- **Texture Size:** Presets for texture sizes.
  - **Auto Adjust To Globe:** Defines whether the texture size is modified when mapped on a globe if texture size is set to a setting other than manual. This option is used when displaying maps that are close to Earth's poles.
- **Adjust for Globe (advanced):** Adjusts a 2D selection in WME converting it to 3D.
- **Adjust for Zoom (advanced):** Fetches a wider map when enabled. Normally, the selected map is zoomed in by approximately ten percent in order to avoid the fade area.
- **Texture Width:** Defines the number of pixels in the map width. When aspect ratio is locked, changing the texture width automatically changes the *Texture Height* parameter.
- **Texture Height:** Defines the number of pixels in the map height. When aspect ratio is locked, changing the texture width automatically changes the *Texture Width* parameter.
- **Texture Compression:** Sets the compression level for the texture (DTX5 is the highest compression level, i.e. less texture quality).
- **Texture Quality**
  - **Linear:** Uses the same image resolution in the entire zoom range.
  - **Mipmap:** Changes resolution according to the distance from the image (managed automatically in Viz Artist).
- **Texture Anisotropic Filter:** Turns on the relevant anisotropic in the image on the same container with CWMClient (similar to Mipmap, above). Available settings are `Off` / `2x` / `4x` / `8x` / `16x` .

Copyright © 2026 Vizrt

Labels

- **3D Designs & Holder:** Defines the labels usage and behavior over the map, and has three available options:
  - **Disabled:** Displays labels as part of the map texture. No manual definitions are required.
  - **Local:** Requires manual definition of the 3D Designs & Holder path (e.g. On Map Designs: `Vizrt/VizWorld/GlobalDesigns/Regions/`)
  - **Global:** Enables you to use a global design defined by the 3D Map Setting scene plug-in.

IMPORTANT! Global labels do not work with on screen labels so you still need to configure them (if needed).

Local and Global

- **Visible Filter:** Select *All* to build all labels defined in WoC. Select *Visible* to build only labels that are shown on the selected map.
- **Backslash as New Line:** When Enabled you can add labels with more than one line typing backslash (\) as a separator for each new line.
- **Labels Type:** Defines the source of the label designs. Available options are Dynamic and Scene. Selecting Viz-DB or Scene defines where the label designs are stored.
  - **Dynamic:** Creates new labels dynamically based on label designs created in the scene itself or fetched from the Viz Graphic Hub (Viz DB). All designs are referenced by the On-Map Designs placeholder. When selecting labels in the Map Editor (WME), all labels are dynamically added underneath the CWMClient's container and receive the correct label type (provided it is defined by your label design). Note that this approach does not allow you to use control plug-ins. As the number of labels are dynamic this approach does not allow you to use control plug-ins.
  - **Static:** Does not create new labels underneath the CWMClient's container, but instead uses the number of labels already defined by the designer (achieved by adding copies of the label designs as sub containers of the CWMClient's container). As the number of labels and label types are static, this approach allows use of control plug-ins.
  - **Viz-DB:** Uses label source from a merged object from the Viz objects library containing the label designs.
  - **Scene:** Uses label source from a group container in the scene hierarchy containing the label designs.
- **On-Map Designs:** Defines the source of the labels that are displayed on the map (Local only). When Viz DB (Viz Graphic Hub) is selected, define the path to a merged object, containing the label designs, in Viz object library. When Scene is selected, drag a group container with the label designs to the container place holder.
- **On Screen Designs:** Defines the source of the labels that are displayed on a plane in front of the screen. When Viz DB is selected, define the path to a merged object, containing the label designs, in Viz object library. When Scene is selected, drag a group container with the label designs to the container place holder.
- **On-Map Holder:** Uses this container for grouping all the generated labels on the map (Local only). When a map with labels is selected, the plug-in duplicates the label designs and creates the labels. The duplicated labels are placed under the *On Map Holder* container.
- **On Screen Holder:** Uses this container for grouping all the on screen labels (labels that are not geo-referenced) used in the map (Local only). When a map with on screen labels is used and Viz is selected, the

Copyright © 2026 Vizrt

on screen label designs are copied to the holder container and the label information is sent to the copied designs.

Note: Back slash (/) in label names are treated as a new line (carriage return).

3D Objects

The 3D Objects section defines the other 3D properties of objects on the map, other than labels (regions, roads and so on).

3D Objects and Regions

- **3D Designs & Holder:** Defines whether the regions are part of the received texture or 3D objects. When set to *Disabled*, regions are displayed as part of the texture. When set to *Local* or *Global* the *Viz-DB* and *Scene* parameters are made available. Also, when set to Global, the global region designs and holder are used to create the on map 3D Region objects. On Screen parameters are enabled to define the on screen region designs and holder.

  Note: The global designs and holder are defined in the 3D Map Setting scene plug-in.

  - **Viz-DB:** Sets region source to be a merged object from the Viz objects library, containing the region designs.
  - **Scene:** Sets region source to be a group container in the scene hierarchy, containing the region designs.
- **On-Map Designs:** Defines the source of the 3D Region objects that are displayed on the map. When Viz DB is selected, define the path to a merged object, containing the region designs, in Viz object library. When *Scene* is selected, drag a group container with the region designs to the container place holder.
- **On Screen Designs:** Defines the source of the 3D Region objects that are displayed on a plane in front of the screen. When *Viz DB* is selected, define the path to a merged object, containing the region designs, in Viz object library. When *Scene* is selected, drag a group container with the region designs to the container place holder.
- **On-Map Holder:** Uses this container for grouping all the generated 3D Region objects on the map. When a map with regions is selected, the plug-in duplicates the region designs and create the 3D Region objects. The duplicated regions are placed under the *On Map Holder* container.
- **On Screen Holder:** Uses this container for grouping all the on screen regions (regions that are not geo-referenced) used in the map. When a map with on screen regions is used and *Viz* is selected, the on screen region designs are copied to the holder container and the region information is sent to the copied designs.

3D Objects and Borders

- **3D Designs & Holder:** Defines whether the border line is part of the received texture or drawn as a 3D object. When set to *Disabled*, borders are displayed as part of the map texture. When set to *Local* the *Viz-DB* and *Scene* parameters are made available and the shapes added in the WME are drawn as a 3D object by Viz. When set to Global, the global border designs and holder are used to create the 3DBorder objects.

Copyright © 2026 Vizrt

Note: The global designs and holder are defined in the 3D Map Setting scene plug-in.

- **Recursive level:** Sets the filter level for border details. If *None* is selected, all border data is drawn. If *Region* is selected, Sub Region data is not drawn. If *Country* is selected, region and sub-region borders are not drawn.
  - **Viz-DB:** Sets the border source to be a merged object from Viz objects library, containing the border designs.
  - **Scene:** Sets the border source to be a group container in the scene hierarchy, containing the border designs.
- **On-Map Designs:** Defines the source of the 3DLine objects that are displayed on the map. When *Viz DB* is selected, define the path to a merged object, containing the border designs, in Viz object library. When *Scene* is selected, drag a group container with the border designs to the container place holder.
- **On-Map Holder:** Uses this container for grouping all the generated 3D Border objects on the map. When a map with borders is selected, the plug-in duplicates the border designs and create the 3DLine objects. The duplicated borders are placed under the *On Map Holder* container.

3D Objects and Shapes

- **3D Designs & Holder:** Defines whether the shapes are part of the received texture or drawn as a 3D object. When set to *Disabled*, shapes are displayed as part of the map texture. When set to *Enabled* the *Viz-DB* and *Scene* parameters are made available and the shapes added in the WME are drawn as a 3D object by Viz. When set to Global, the global shape designs and holder are used to create the 3D shape objects.

  Note: The global designs and holder are defined in the 3D Map Setting scene plug-in.

  - **Viz-DB:** Sets the shape source to be a merged object from Viz objects library, containing the shape designs.
  - **Scene:** Sets the shape source to be a group container in the scene hierarchy, containing the shape designs.
- **On-Map Line Designs:** Enables the user to add vector line data to the map. Either by using the draw option in the WME (Add Shape), or from selecting an existing line data (e.g. street).
- **On-Map Area Designs:** Enables vector area designs. This option has support for the area draw option in the WME.
- **On-Map Line Holder:** Defines the source of the 3D Region objects that are displayed on the map. When *Viz DB* is selected, define the path to a merged object, containing the shape designs, in Viz object library. When *Scene* is selected, drag a group container with the shape designs to the container place holder.
- **On-Map Area Holder:** Uses this container for grouping all the generated 3D shape objects on the map. When a map with shapes is selected, the plug-in duplicates the shape designs and create the 3D Region objects. The duplicated shapes are placed under the *On Map Holder* container.

3D Objects and Roads

- **3D Roads:** Defines whether the road data is drawn on the map and the way the roads are drawn. Available options are None, Crop To Map and All.
  - **None:** The roads data is not available to be drawn on the map.

Copyright © 2026 Vizrt

    - **Crop To Map:** Enables the *Visibility Filter* and *Visibility Factor (%)* settings, limiting the roads data.
    - **All:** Loads the selected road data.
    - **Labels Only:** Draws only the road labels.
- **Crop Visibility Filter:** Sets the highest level of road type that is cropped. Available options are Freeway, Primary, Main, Secondary and Other.
    - **Freeway:** Crops all roads rated as freeways and lower (that is all roads) using the *Visibility Factor*.
    - **Primary:** Crops all roads rated as primary roads and lower (that is primary, main, secondary and other) using the *Visibility Factor*.
    - **Main:** Crops all roads rated as main roads and lower (that is main, secondary and other) using the *Visibility Factor*.
    - **Secondary:** Crops all roads rated as secondary roads and lower (that is secondary and other) using the *Visibility Factor*.
    - **Other:** Crops all roads rated as other roads (that is none of the above) using the *Visibility Factor*.
- **Visibility Factor (%):** Defines the cropping area of the roads on the map. 100% means the roads are cropped at the map edges and cover the entire map area. A lower value causes the selected road types in the *Visibility Filter* to be cropped (evenly from the map edges).
- **Simplify Threshold (m):** Applies a simplifying algorithm on road data. The number represents the biggest allowed error.
- **Ignore Filter:** Disables creation of roads that are lower or equal to the selection (i.e. Freeway, Primary, Main, Secondary, and Other).
- **Roads Mode:** Applies one set of road design for all roads (Global), or uses a design per CWMClient (Design).
    - **Source:** Sets the source for road designs. Available options are *Viz-DB* and *Scene*. **Viz-DB** sets road source to be a merged object from Viz objects library, containing the road designs. **Scene** sets road source to be a group container in the scene hierarchy, containing the road designs.
    - **On-Map Designs:** Defines the source of the 3DRoads objects that are displayed on the map. When *Viz-DB* is selected, define the path to a merged object, containing the road designs, in Viz object library. When *Scene* is selected, drag a group container with the road designs to the container place holder.
    - **On-Map Holder:** Uses this container for grouping all the generated 3DRoads objects on the map. When a map with roads is selected, the plug-in duplicates the road designs and create the 3DRoads objects. The duplicated roads are placed under the *On Map Holder* container.

3D Objects and Advanced

- **Regions Visible Filter:** Defines whether *all* the selected regions are created (when using 3D regions) or only the regions that are *visible* on the selected map.
- **Sort Regions:** Defines how the labels and 3D objects are duplicated in the scene tree.
    - **No:** Creates the duplicated labels and 3D objects in the scene tree grouped by geographic levels that is for each country, first the country, then the regions, then the sub regions, and so on.
    - **Ascending:** Creates the duplicated labels and 3D objects created in the scene tree sorted by ascending geographic levels, that is first the sub regions, then the regions, then the country container.
    - **Descending:** Creates the duplicated labels and 3D objects in the scene tree sorted by descending geographic levels, that is first the sub regions, then the regions, then the country container.
- **Copy Map to Region:** Sets if a map of the region is applied to the 3D Region object.
- **Auto Borders:** Defines whether borders are added automatically to the map and the level of the borders:
    - **Off:** Does not add borders.

Copyright © 2026 Vizrt

    - **Highest:** Draws country borders for the selected area (or region borders if only a region was selected).
    - **Intermediate:** Draws region borders (or sub-regions if only a region was selected).
    - **Lowest:** Draws sub-region borders.
- **Auto Regions:** Automatically adds region designs based on location.
    - **Off:** Does not add regions.
    - **On Screen:** Adds region on screen automatically.
    - **On Map:** Adds adds region on map automatically.
- **Auto Regions Level:** Determines the level when creating auto 3D Regions.
- **Set 3D Models:** Determines whether the 3D models plug-in is called to generate 3D buildings in the selected map area.

Miscellaneous

The Miscellaneous section includes additional general parameters of the map.

- **WME Aspect–** Defines the aspect of the map in the WME window:
    - **Image:** Sets the aspect by the texture size defined in the texture screen of the CWMClient plug-in when selected.
    - **User:** Allows manual setting of the aspect when selected. When selected, the *Aspect* parameter is enabled. The aspect is modified by changing the aspect value.
    - **Camera:** Sets the aspect to be the same as the current camera in use. Selecting this option opens the WME in the same aspect as the current camera.
    - **Auto:** Looks for a Navigator plug-in in the hierarchy, above the CWMClient plug-in when selected. If a Navigator is found, the camera aspect is used. If the Navigator plug-in was not found, the image aspect is used.
- **Freeze:** Retrieves the map from from the server and saves it as an image in the Viz image library and used as a static geo-referenced image when set to `On`. All map dynamic parameters are hidden.

> ⚠ **Note:** When freezing a map, Hop scenes with pyramids will no longer work. This is a known limitation.

- **Obey Hops Manager:** Uses settings of labels/regions/3D Objects as defined in the Hop Manager plug-in when set to `On`. When set to `Off`, settings are local to this CWMClient plug-in.
- **Transition Logic Mode:** Requests the map from the server or cache when the scene is loading when set to `Off`. When set to `On`, the scene loads without requesting a map since the control application (e.g. Viz Trio or another external application) sends the parameters for a map to use. This option saves time during initialization of scenes using a dynamic map.
- **Lock Project & Style:** Locks the CWMClient plug-in to only use the selected TPL and style sheet and ignores any changes made from the WME.
- **Is Linked to Master Map:** Permits the CWMClient (slave) to be controlled by another CWMClient (master) when set to `On`. To control another CWMClient, drag and drop a container with a CWMClient (slave) plug-in onto the CWMClient (master) plug-in's Linked Map placeholder (see next parameter).
- **Linked Map:** Defines the map (slave) that is linked to the current map (master) in a container placeholder when set. The linked map is another CWMClient container that receives the same map as the map selected from the WME.

Copyright © 2026 Vizrt

- **Linked Map Size (%):** Defines the size of the linked map as a percentage from the main map, that is if 50% is defined the linked map shows half of the area defined in the main map.
- **Linked Type (advanced):** Determines if the link type only include bounding box or should it include all the data (regions, labels etc.).
- **Shared Memory Link:** Defines which is the master (source) and client (target) when the same Shared Memory Identifier is used.
- **Shared Memory Type:** Defines the accessibility of the shared memory.
  - **Global:** Makes shared memory accessible to all scenes currently loaded in memory. This is useful when working with Transition Logic scenes where your Viz World map can be one scene and the locator a different one and data can easily be transferred between the two.
  - **Scene:** Makes shared memory accessible only locally and to the current scene. Every scene has one shared memory map that can be used to exchange data among the scripts in the scene.
  - **Distributed:** Makes shared memory accessible to all computers connected to one Viz Graphic Hub.
- **Shared Memory Identifier:** Defines the identifier for the shared memory map.
- **URL Images:** Container holding the URL Image manager which is invoked when the map changes.
- **Always Fetch Terrain Data:** Fetches terrain data always when enabled.
- **Terrain Tessellation Locked:** Appears when a C3DTerrain plug-in is combined with the CWMClient plug-in. Tessellation is the terrain resolution, which is the number of polygons used to build the terrain object. The higher the tessellation, the more detailed the terrain is. Use the *Width Tessellation* and *Height Tessellation* parameters to fine tune the quality versus the performance of the scene.
  - When set to `On`, only the *Width Tessellation* parameter can be changed; hence, the setting applies for both *Width* and *Height*.
  - When set to `Off`, the parameters can be set individually.

Imagery

The Imagery section includes multiple imagery configuration options.

- **Data Source:** Defines the source of the imagery. Available options are Bing (MIcrosoft Bing server), CMR (Curious Multi-Resolution imagery), Digital Globe (DG) or Blom.
  When enabled (On) additional parameters are made available, and the WME button is disabled.
  - **Data Resolution:** Available resolutions in pixels (i.e. width).
  - **Temp Folder:** Sets a temporary download folder.
  - **Data Type:** (Visible when Data Source is Bing). Available data types are satellite, map or a combined type of imagery.
  - **CMR:** (Visible when Data Source is CMR). Defines a CMR file to be used as a source for the map.
  - **Digital Globe Projection:** (Visible when Data Source is DG). Allows you to select the DG project. Currently available as Mercator and Unprojected.
- **Longitude:** Sets the map longitude.
- **Latitude:** Sets the map Latitude.
- **CMR Units:** Sets the Curious-Multi Resolution imagery's units to degrees, kilometers, miles or nautical miles.
- **Width (deg):** Sets the area width in degrees, kilometers, miles or nautical miles.
- **Status:** Displays the current CMR operation status.
- **Get Image:** Retrieves the defined imagery (of the area defined by Lon, Lat and Width parameters).
- **Center Longitude Latitude:** When clicked, the CMR is created with the longitude and latitude values as the center of the CMR.

Copyright © 2026 Vizrt

Note: The longitude and latitude values must be within the CMR area. If the values are not within the CMR area no map is created.

Buttons

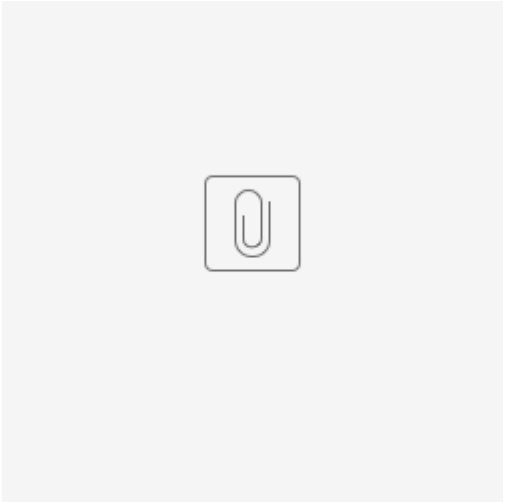
The following default buttons apply to the plug-in as a whole.

- **Viz World Map Editor (WME):** Opens the WME. WME connects to Viz World Server that is defined in Viz Config, and retrieves the current map or opens the defined default project if no map exists (fresh plug-in instance).
- **Refresh Map:** Re-draws the map and refreshes the related Viz objects using Viz World Client (WoC) plug-ins in the scene hierarchy. For example if a label design has been changed in Viz Artist, clicking Refresh Map redraws the map with the new label design without opening WME.
- **Refresh Map (Force New):** Recreates the map on the server and saves it to the cache. This operation forces the map creation, even if the map exists in the cache folder.
- **WME (Classic):** Opens WME Classic interface instead of the new WME

Search Order

The CWMClient can hold designs for labels and 3D objects such as regions, borders and shapes. As the designs can be stored in the scene or on the Viz Graphic Hub (i.e. Viz-DB), you need to understand the order in which these designs are searched for. The following rules apply when searching for designs under the Design holders.

Example I - Add Label Map Tool Bar



If, when using the World Map Editor or Map Editor Classic, you add a label and set the style to be *Traffic_Accident*, the search order is based on the following rules:

1. Search the Scene - that may have multiple designs where each design has a name that corresponds to its design.

Copyright © 2026 Vizrt

2. Search Viz Graphic Hub (i.e. Viz-DB) - that may have multiple designs stored in a design folder where each design is a merged object with a name that corresponds to the design.

First, the Scene design holder is searched (if used) in the following order:

1. By Style
2. By Design (Point_Label)
3. By Default Design (Default_Design)

This means that it searches for and uses:

1. Traffic_Accident, and if not found then
2. Point_Label, and if not found then
3. Default_Design
   If none were found, the Viz-DB design holder is searched in the same order.

Example II - Add label Search Tab



If, when using the World Map Editor or Map Editor Classic, you search for and add a Town with more than one million (Towns > 1m) inhabitants as a label to the map and for the CWMClient set the style to *Bigtown* the search order is based on the following rules:

1. Search by style only if different than Place
2. By detail (Capital / Town_1m / Town_100k / Town_10k / Town_1k / Town)
3. By default town design (Default_Town)
4. By style (if = Place)
5. By default design (Default_Design)

This means that a search searches for and uses:

1. Bigtown, and if not found then
2. Town_1m, and if not found then
3. Default_Town, and if not found then
4. Place,
5. Default_Design

Copyright © 2026 Vizrt

Naming Conventions

> **Note:** Names using letters other than [a-z A-Z 0-9] are converted to _ (underscore).

As there are several types of labels the following is the search order for each label type:

1. Capital
2. Town 1M
3. Town 100K
4. Town 10K
5. Town 1K
6. Town
   - Search by style only if different than Place
   - By detail (Capital / Town_1m / Town_100k / Town_10k / Town_1k / Town)
   - By default town design (Default_Town)
   - By style (if = Place)
   - By default design (Default_Design)
7. Region:
   - By style
   - By region level (Country / Region / Sub_Region)
   - By detail (Region)
   - By default design (Default_Design)
8. Point Label:
   - By style
   - By design (Point_Label)
   - By default design (Default_Design)
9. None of the above:
   - Search by style only if different than Place
   - By default design (Default_Design)
10. All Others:
    - Search by style only if different than Place
    - By detail (Region / Region_Name / River_Name ...)
    - By style (if = Place)
    - By default design (Default_Design)

    List of details name:

- Capital
- Town_1m
- Town_100k
- Town_10k
- Town_1k
- Town
- Tourist_Attraction
- Reserve_Or_Park
- Region_Name
- Physical_Area_Name
- Physical_Water_Name

Copyright © 2026 Vizrt

- River_Name
- Mountain_Name
- Label
- Street_Address
- Intersection
- Traffic_Incident
- Traffic_Accident
- Traffic_Construction
- Traffic_Speed
- Point_Label
- None_Of_The_Above
- Region

Copyright © 2026 Vizrt

5.4.10 Focus On Map



The Focus On Map plug-in creates a single hop (high resolution map over a large referencing map) without using a navigator plug-in or animation. This plug-in is used when creating a large map (reference map) with 3D Objects over it (roads, shapes, and so on). It enables the user to display a high resolution area of the large map without recreating the 3D objects when changing the displayed area.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

Focus On Map Properties

Common Properties

- **Geometry Offset X/Y (%):** Offsets from the true position of the geometry in geometry units.
- **Min/Max Distance:** Sets the minimum/maximum distance from the camera to the map.
- **Get Map:** When clicked, the camera jumps to the defined area.
- **?:** Displays help.

Position

The **Position Tab** enables the parameters for the requested map area where the Position Source defines the source of the viewed area.

- **Map:** Fits the selected map in the Focus On Map container to the screen according to the *Fit To Screen By* selection.
  - **Fit To Screen By:** Sets the map attribute that is used to fit the map to the screen: Width, Height, Min (the minimum value of the map's width and height), Max (the maximum value of the map's width and height), Average (the average of the map's width and height).
  - **Longitude Offset:** Sets a longitude offset from the center of the selected map.
  - **Latitude Offset:** Sets a latitude offset from the center of the selected map.
  - **Distance Offset:** Sets a distance offset from the center of the selected map.
  - **Pan:** Sets a pan value for the camera.
  - **Tilt:** Sets a tilt value for the camera.
- **Absolute:** Enables the user to manually set the parameters for the viewed area.
  - **Longitude:** Sets the Longitude of the viewed area (center).
  - **Latitude:** Sets the Latitude of the viewed area (center).
  - **Distance offset:** Sets a fixed distance offset from the selected map.
  - **Distance:** Sets a distance from the map.
  - **Diameter:** Sets the desired view as Diameter (and not distance).
  - **Pan:** Sets a pan value for the camera.
  - **Tilt:** Sets a tilt value for the camera.
- **Geometry:** The selected map in the Focus On Map container is fitted to the screen according to the Fit To Screen By selection.

Copyright © 2026 Vizrt

- **Fit To Screen By:** Select the 3D Region attribute that is used to fit the map to the screen: Width, Height, Min (the minimum value of the map's width and height), Max (the maximum value of the map's width and height), Average (the average of the map's width and height).
- **Longitude Offset:** Sets a longitude offset from the center of the selected 3D Region.
- **Latitude Offset:** Sets a latitude offset from the center of the selected 3D Region.
- **Distance Offset:** Sets a distance offset from the center of the selected 3D Region.
- **Pan:** Sets a pan value for the camera.
- **Tilt:** Sets a tilt value for the camera.

> ⚠ **Note:** Pan and Tilt parameters are disabled unless the *Pan and Tilt Animation* parameter in Navigator is enabled (`On`).

Camera

The **Camera Tab** defines the camera parameters such as camera number and minimum and maximum distance of the camera from the map.

- **Camera:** Defines the camera number that is affected by the position parameters.
- **Navigation Method:** Allows you to select whether the camera or the container should also move when a map changes position.
  - **Camera:** Moves the camera when the map is repositioned, potentially moving other objects out of frame.
  - **Container:** Moves the container instead of the camera, keeping other objects in view as the camera is still. In other words, moving the base map instead of the camera to see other parts of the map. This setting also means you do not have to use the front layer using two cameras to achieve the same effect as when moving the container. Borders and other elements on the map can be preloaded once for the base map, but this can only be done with a flat map (not a globe).
- **Clipping Plane Control:** Defines the selected camera's clipping plane.
  - **Static (Viz):** Draws objects within the clipping plane values defined in Viz. For Viz 3.x see **Scene Settings > Renderer > Camera Clipping Plane**.
  - **Focus On Map:** Adjusts the clipping plane values according to the camera position. This is automatically done by based on the *Near Distance* and *Far Distance* parameters.
  - **Near distance (%):** Defines the minimum distance of the camera from the map.
  - **Far Distance (%):** Defines the maximum distance of the camera from the map.

If **Position Source** is set to **Absolute** then the parameter **Diameter** can also be set here. If **Position Source** is set to **Geometry** then the parameters **Geometry Offset X/Y** and **Min/Max Distance** can also be set here.

Labels

The **Labels Tab** defines label parameters for how labels should be displayed.

- **Labels:** Defines how the labels are displayed:
  - **Overlay:** Displays labels as a layer on the screen, but the layer is not affected by the camera movement when a new area is selected.
  - **On Map:** Displays labels on the map, moving with the map as the selected area is changed.
  - **On Map Scaling:** Displays labels on the map

Copyright © 2026 Vizrt

- **Label Camera:** Defines the camera number used for displaying the on screen labels.

If **Position Source** is set to **Absolute** then the parameter **Diameter** can also be set here. If **Position Source** is set to **Geometry** then the parameters **Geometry Offset X/Y** and **Min/Max Distance** can also be set here.

Copyright © 2026 Vizrt

5.4.11 Geo Text



The Geo Text plug-in displays the longitude and latitude values received from a variety of sources.

The geographic data is displayed in two text objects (or one) defined by the longitude and latitude containers.



Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

Geo Text Properties

The **Source** defines the source of the geographic data that is displayed in the defined containers under the Containers tab.

- **Source:** Defines the source of the geographic data that is displayed in the defined containers under the Containers tab.
  - **User:** Sets the value manually in the Compass, Longitude and Latitude parameters for the position and range values. The Compass, Longitude and Latitude parameters are enabled only when Position is set to User.
  - **WPosition:** Displays the values received from World Position on the same container as Geo Text in the defined Containers.
  - **Navigator:** Receives the geographic data from Navigator in the scene. The data received data from Navigator is the location of the center of the Navigator point of view and the difference between the edges of the current Navigator map. Geo Text also has placeholders (all requiring a running Viz World server) for defining the administration levels:
    - **Country:** Gets the current Country
    - **Admin 1:** Gets the current administration level 1 (for example, Region)
    - **Admin 2:** Gets the current administration level 2 (for example, Town)
    - **Location:** Gets the lowest administration level (for example, Place)

Copyright © 2026 Vizrt

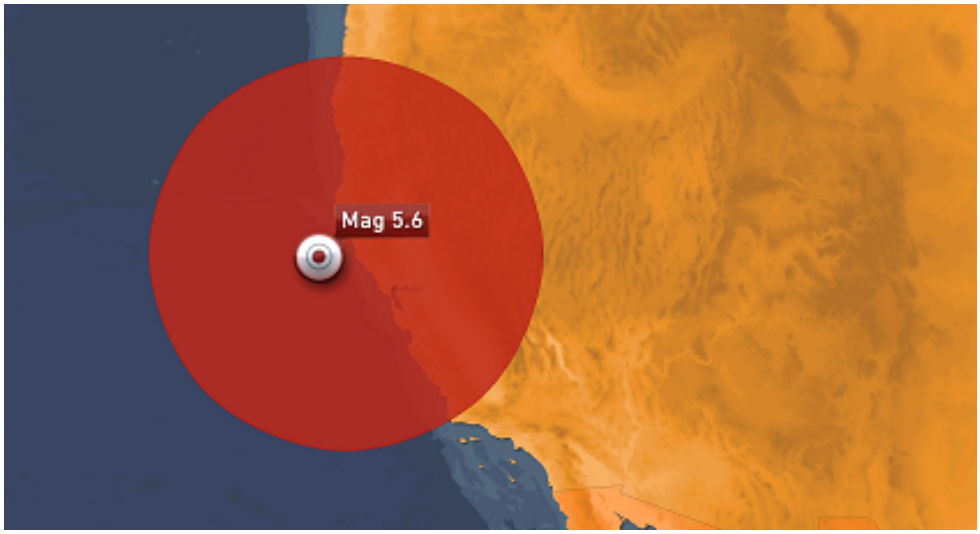
- **WPoint:** Displays the value received from WPosition on the same container as Geo Text in the defined Containers.
- **Map:** Sends the center of the map data to the Longitude and Latitude containers and the difference between the edges of the map to the range containers when placed on a map container.
- **Line:** Sends the center of the line data to the defined containers in the Containers tab when placed on a 3DLine container.
- **Position:** Defines the position text format. Select the required format from the Degrees Format list.
- **Range:** Defines the range text format. Select the units for the displayed values and select the required format from the Degrees Format list. If Kilometers, Miles or Nautical Miles are selected, Dot or Comma can be selected as a separator with a fixed decimal point.
- **Containers:** Defines the font objects that display the longitude and latitude values of the position and range:
    - **Longitude Position:** Links to the text object displaying the longitude position data received from Geo Text.
    - **Latitude Position:** Links to the text object displaying the latitude position data received from Geo Text.
    - **Compass:** Links to an object that is rotated to show the North (based on navigator direction).
    - **Longitude Range:** Links to the text object displaying the longitude range data received from Geo Text.
    - **Latitude Position:** Links to the text object displaying the latitude range data received from Geo Text.
    - **Legend:** Scales view width and view height values based on the object used for the legend.
    - **Length:** Links to the text object displaying the length of the line object as received from Geo Text.

Copyright © 2026 Vizrt

5.4.12 GeoData Reader



The GeoDataReader plug-in gets a geodata stream or file (*.shp*/*.kml*/*.kmz*/*.gdb*/*.mdb*) and inserts the data into Viz. The plug-in can use local scene containers or the global ones that were defined in the 3D Map Setting for the design and data. The shapes can be grouped into different containers, by their geometry, or grouped together in one container.



The names of the available services can be found inside the `Name` tag under `FeatureType`:

```
<FeatureType> <Name>glacier_outlines</Name> ... </FeatureType>
```

For example:

```
http://nsidc.org/cgi-bin/atlas_south?
service=WFS&version=1.1.0&request=GetFeature&typename=glacier_outlines
```

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

GeoDataReader Properties

General

- **Source Type:** Sets whether the source is a local file or a web stream.
- **<Geometry> Design and Holder**
  - **Disabled:** Does not add this type.
  - **Local:** Takes the design and the holder containers locally.
  - **Global:** Takes the design from Graphic Hub.

Copyright © 2026 Vizrt

- **Designs Path:** Sets the path to a design folder that is placed inside the Graphic Hub.
- **<Geometry> Styles Field Name:** Determines the name of the field that contains the style name that each feature has.
- **<Geometry> Split**
    - **Disabled:** Groups all features in one container.
    - **On:** Gives each feature its own container.
    - **By Style:** Groups features by the style name.
- **Clear Existing Features on Read**
    - **Disable:** Adds the features on each Read command.
    - **Enable:** Erases the container that holds the source features, then adds the new features.

Proxy

- **Use Proxy:** Enables proxy configuration parameters when set to `On`. Use this option when working an a network with proxy:
    - **IP:** Determines proxy server IP number.
    - **Port:** Determines proxy port number.
    - **Username/Password:** Sets the username and password.
- **Read:** Adds the features from the proxy.
- **Read and Save As Viz World Binary (VWB):** Adds the features from the proxy and saves them.

Copyright © 2026 Vizrt

5.4.13 Hop It



The Hop It plug-in is a utility plug-in which connects the specific design with a hop.

It is used when the designs (for example, labels, regions etc.) were not generated from the hop you want to connect them to (or from no hop at all).

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

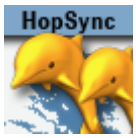
Hop It Properties

- **Hop Mode**
  - **Automatic:** Sets hop points automatically (based on searching all hops locations and connecting to the one which is closest).
  - **Manual:** Sets hop points using the *Hop Point* parameter.
- **Hop Point:** Sets the hop point in the hop sequence. Note that the animation is built in the same sequential order as the list of Hop Points (Map-Start, Destination-1, Destination-2, and so on). If two hops use the same Hop Point, the animation does not work properly.

The three plug-ins that are influenced are Hop Sync, 2D Label and Label It.

Copyright © 2026 Vizrt

5.4.14 Hop Sync



The Hop Sync plug-in coordinates between labels and other 3D objects with built-in animations and hop animations. The Hop Sync plug-in is applied to the same container as Label It, or to a container above the label design containers but below the top design container which is used to create the label merged object. To use Hop Sync plug-in, the label designs must include a merged object containing the design. The plug-in defines a point in the label animation that is matched with the hop point in the Navigator animation.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps

Hop Sync Properties

- **Director:** Defines to which director the label animation is copied to.
  - **Navigator:** Copies the label animation to the Navigator director. This option is used when the label animation should end before or at the time the Navigator animation has reached a hop point. When the Navigator director reaches a hop point and Continue is pressed (or pause time ends) the label animation also continues, causing the label to disappear.
  - **Hop Director:** Copies the label animation to a new director. This option is used when the label animation should end after the hops animation has stopped (or paused). If the label animation is copied to the Navigator director, the label animation stops before the entire label is revealed.
  - **Separate:** Copies the label animation to a new director (not the Hop or Navigator director) to give you more options. Note that it does not perform animation offsets.
- **Animation Start Time:** Defines the point in the label animation that is matched to the hop point in the Navigator animation. The value is set by typing in a number (in seconds), or clicking and sliding the mouse over the parameter until the animation point is reached. Another option is to use the *Set Highlight Time* button. Play the animation and when reaching the requested point in the label animation, stop and click the **Set Highlight Time**. The current label animation value is copied to the *Animation Start Time* field.
- **Order Offset:** Enables/disables an offset between the animation of the objects.
  - **Animation Order Offset Time:** Determines the offset for the animation (for example, when setting this value to `1.2`, the third detail animation starts after 2.4 seconds after the animation has reached the destination).

Copyright © 2026 Vizrt

5.4.15 Hops Manager



The Hops Manager plug-in controls multiple instances of several plug-ins from one interface.

The plug-in should only be used if all parameters for the control plug-ins are the same. If for any reason different values are needed, the plug-in should not be used.

The plug-in controls CWMClient, Map Pyramid and NavFinder; however, only instances which are part of a hop are controlled. The plug-in has two main options; **global values** and **hops**.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps_Man

Hops Manager Properties

Global Values

Global values are used in order to update values globally on all control plug-ins.

Under the global values tab all parameters are the same as those found in the actual plug-ins you want to control. For more information see:

- CWMClient
- NavFinder

Hops

Hops are used to jump to a specific hop. From this tab you can select and refresh a map and fine tune pan values from NavFinder.

- **Multi Hop:** Opens the Multihop Editor in the World Map Editor when the Multihop button is pressed.

Copyright © 2026 Vizrt

5.4.16 KML Reader



The KML Reader plug-in adds objects from KML files to the scene.

KML files are XML formatted files containing information about geographic objects: Labels, shapes, lines. KML Reader reads the files and generates objects according to the file's content.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Man

KML Reader Properties

General

- **KML File:** Defines the path to the KML file.
- **Line Designs and Holder:** Defines the graphic designs for lines.
  - **Disabled:** Does not use line designs. Lines found in the file are not drawn.
  - **Local:** Uses specific line design containers and holder container. When selected additional parameters are enabled: Line Designs and Line Holder. Assign the required designs and holder to the plug-in.
  - **Global:** Uses the global line designs and holder container. The global designs and holder are defined in 3D Map Setting.
- **Line Split**
  - **Off:** Places all lines go into one container.
  - **On:** Places every line into its own container.
  - **By style:** Places lines that share a style together in a container
- **Connect Labels to Line:** Creates a line by connecting all labels into a line.
- **Update on Load:** Sets whether the data is read when the scene is loaded.
- **Label Designs and Holder:** Defines the graphic designs for labels.
  - **Disabled:** Does not use label designs. Labels found in the file are not drawn.
  - **Local:** Uses specific label design containers and holder container. When selected additional parameters are enabled: Label Designs and Label Holder. Assign the required designs and holder to the plug-in.
  - **Global:** Uses the global label designs and holder container. The global designs and holder are defined in 3D Map Setting.
- **Shape Designs and Holder:** Defines the graphic designs for shapes.
  - **Disabled:** Does not use shape designs. Shapes found in the file are not drawn.
  - **Local:** Uses specific shape design containers and holder container. When selected, additional parameters are enabled: Shape Designs and Shape Holder. Assign the required designs and holder to the plug-in.
  - **Global:** Uses the global Shape designs and holder container. The global designs and holder are defined in 3D Map Setting.
- **Maximum Number of Shapes:**
- **Georef Images:** Defines whether georef images found in the KML file are used.
  - **Disabled:** Does not draw images found in the file.

Copyright © 2026 Vizrt

    - **Enabled:** Draws images defined in the file and geographically references them. When selected an additional parameter is enabled: *Georef Images Holder*. Assign the required holder to the plug-in.
- **Maximum Number of Labels:**

Hop Control

- **Set Navigator Hops:** Creates hop points based on the information in the KML file. Selects which information to use (line, shape, etc.).
- **Exclude Hops:** Excludes the hop animation for the following labels; None, First, Last, or First and Last. Manual Shift allows you to define which destination is the starting point (limited to a selection of 30 destinations).
- **Start Hop:** Determines the starting point for the hop.
- **Point Data Diameter (km):** Sets the diameter used for each hop point when creating hops based on labels.
- **Set Navigator Hops:** Enables the scene to do a hop animation between the labels.
- **Print Debug Information:** Prints debug messages to the Viz console when set to `On`.
- **Read File:** Triggers the plug-in to read the files and re-draw the objects.

Proxy

- **Use Proxy:** Defines the network uses a proxy server. When set to `On`, additional parameters are enabled:
    - **IP:** Sets the IP address of the proxy server.
    - **Port:** Sets the port number to be used.
    - **Username, Password:** Sets the login credentials.

Copyright © 2026 Vizrt

5.4.17 Label AddOn



The Label AddOn plug-in assigns images, geometries and text based on label parameters.

Labels can have up to four extra parameters.

- Label parents: Country, Admin1, Admin2 (also see Place Finder)
- Label type: Type

Based on the container name (case sensitive), CWMClient updates the Value field in the plug-in with the correct value. For example, if you have selected London and you have a Label AddOn plug-in on a container named *Country* it receives the value of `United Kingdom` and if you have another Label AddOn plug-in on a container named *Type* It receives the value of `Capital`. A use case for this plug-in is whenever you want to have flags for all countries in the world in a folder and use the right flag based on the country. Another use case is having different images for different types and using the correct image based on the type.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

Label AddOn Properties

- **Type**
  - **Images:** Can be applied from the database on a local container.
  - **Geometries:** Can be applied from the database on a local container.
  - **Text:** Can be changed on the local container.
- **Value:** Provides values used to seraph the object or set the text. Can be edited for testing.
- **Prefix:** Sets the prefix used for searching the database. For example `MyImages/Flags/` `Suffix` or `_small.`
- **Hide if Empty:** Hides the container if an object/image is not found.
- **Update:** Tests the prefix and suffix.

Copyright © 2026 Vizrt

5.4.18 Label and Go Assistant



The Label and Go Assistant plug-in works in conjunction with Label and Go to add or delete new labels.

When working with streets it allows editing labels and replacing names and styles of roads. Changes can be saved in a *.csv* file that can be read afterwards by Label and Go.

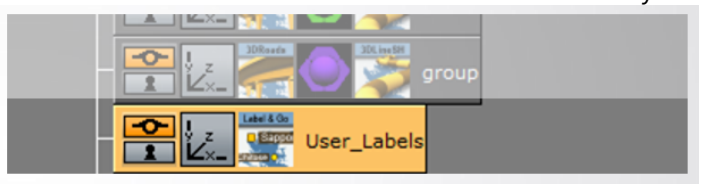
Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

Label and Go Assistant Properties

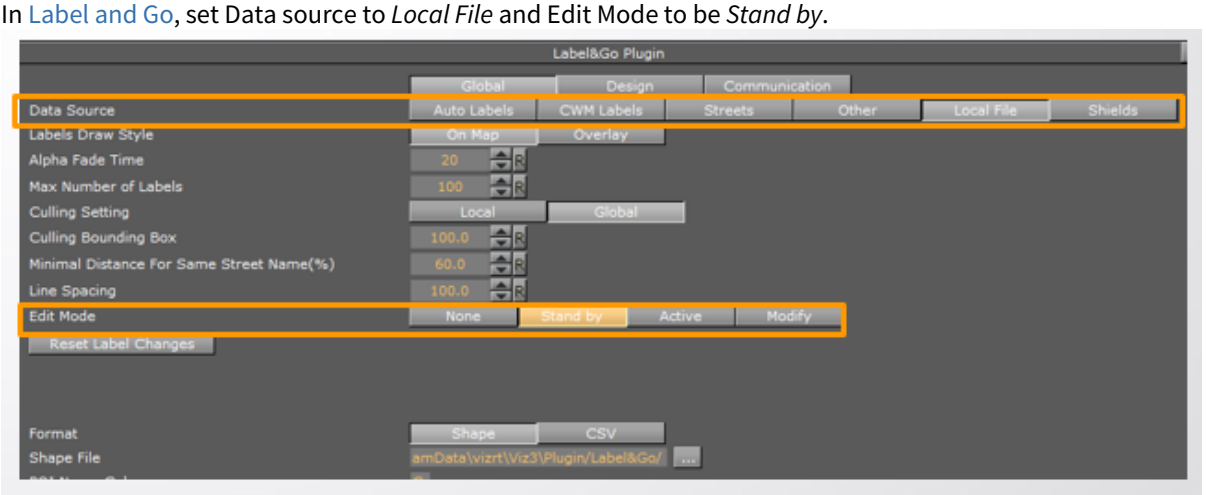
- **Label:** Names the new label.
- **Input Interface:** Toggles whether to use interactive input (combined supports touch) or mouse only.
- **Interactive Input:** Toggles revert or replace (in roads mode).
- **Action:** Selects whether to replace name, style or both.
- **Select Labels (WME):** Adds user label from server (WPF).
- **Select Labels (Classic):** Adds user label from server (classic).

Adding User Labels

1. Add Label and Go to the new container under the hierarchy.



2. In Label and Go, set Data source to *Local File* and Edit Mode to be *Stand by*.

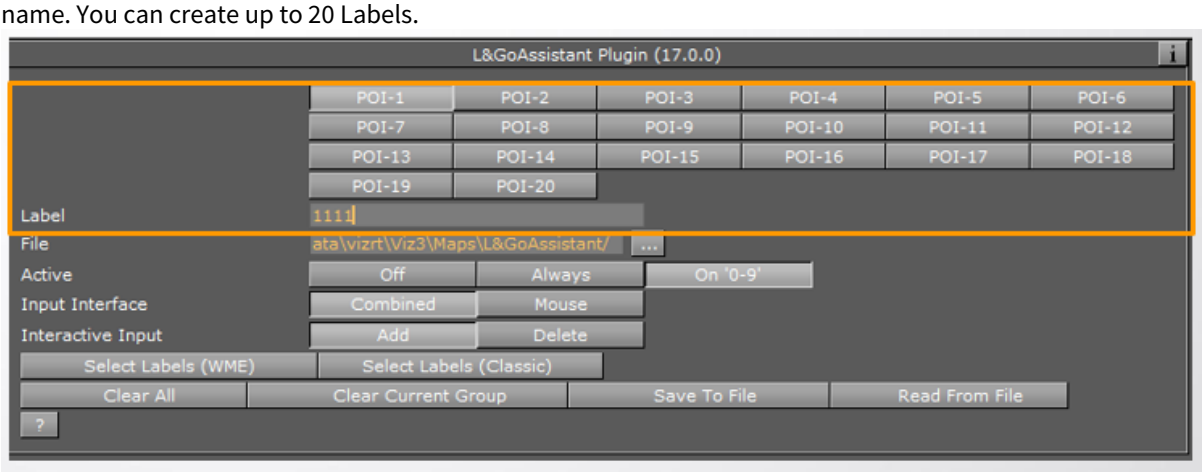


3. There are three modes:
    - **Stand by:** Allows you to select the place where you want to add or remove your labels.
    - **Active:** Allows to add or remove labels depending in your selection.
    - **Modify:** Allows you select a label and then you can move it or rename it.

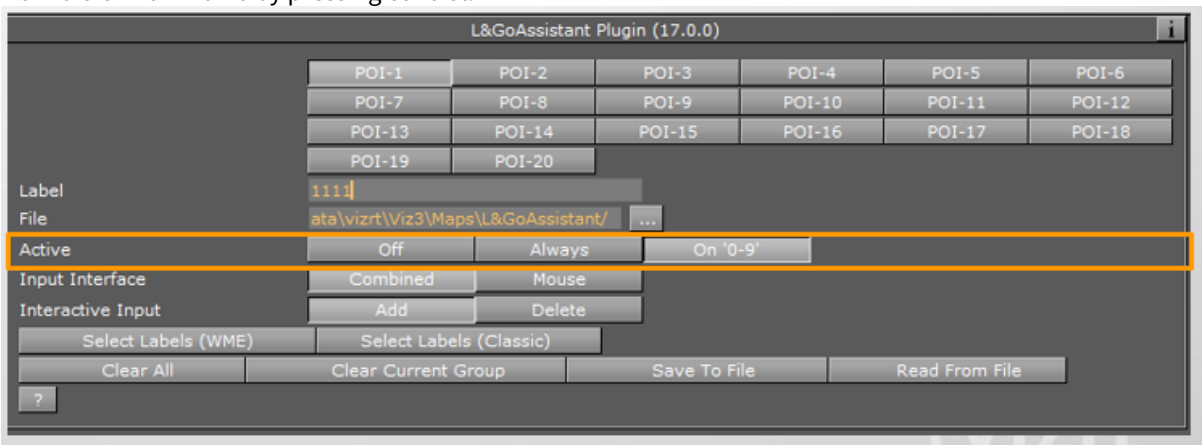
Copyright © 2026 Vizrt

Tip: You can toggle between the Stand by Mode and the Active Mode by pressing **E** on the keyboard.

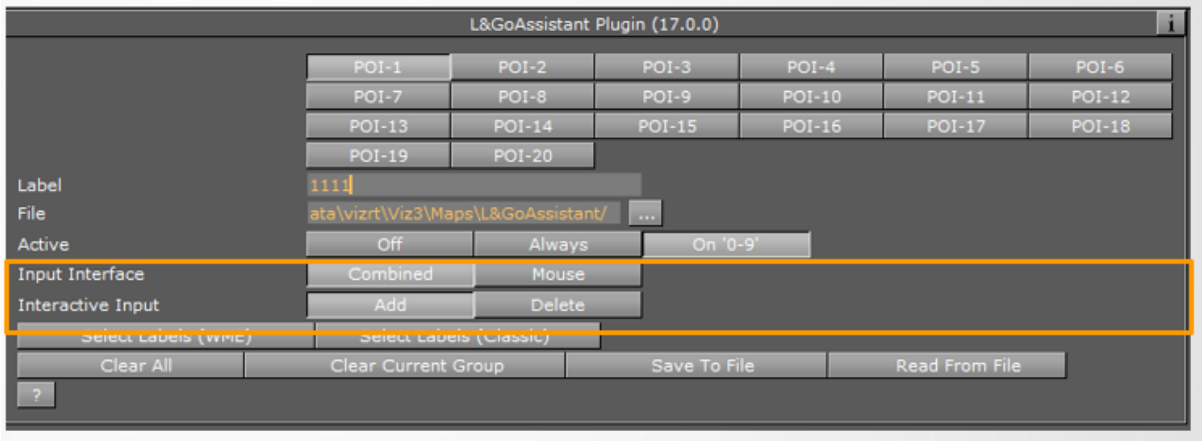
4. Add the Label and Go Assistant plug-in in the same container that Label and Go resides in.
5. Select *PQI-1* for Label number 1 Write the name of the Label. Select *PQI-2* for Label number 2 and change the name. You can create up to 20 Labels.



6. Active set to **On '0-9'** means that hitting the actual number on the keyboard adds labels. You can toggle from 0-9 or from 10-20 by pressing control.



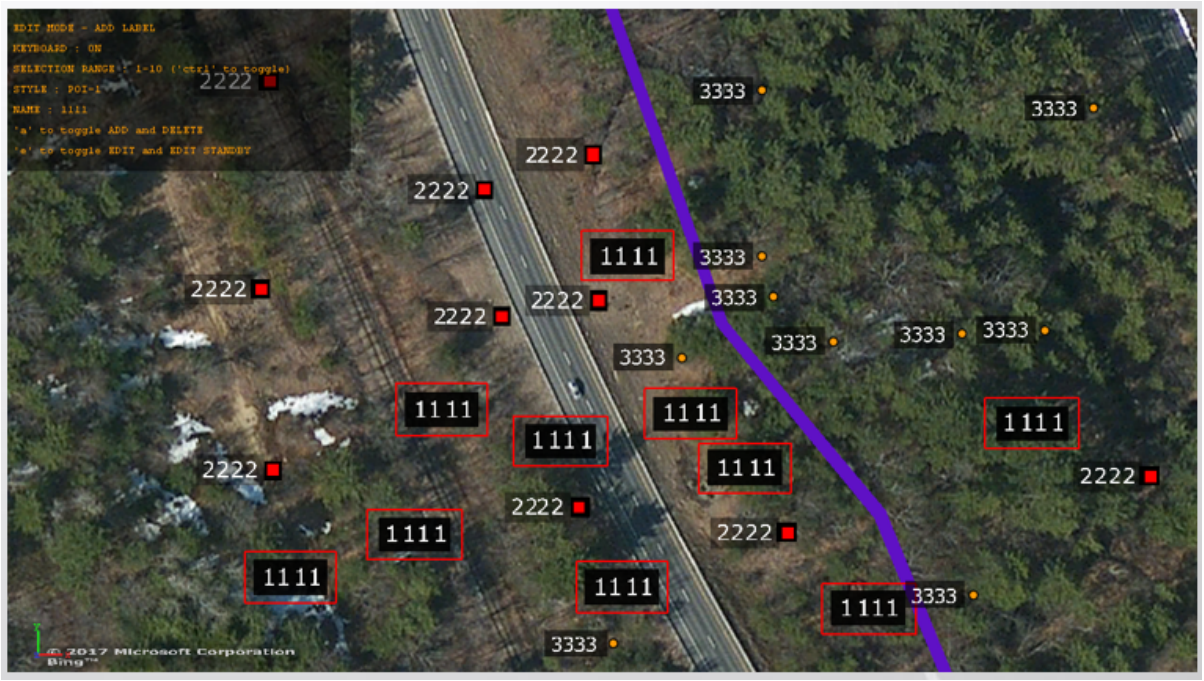
7. Set the Input Interface: Combined - Touch screen and Mouse - Only Mouse.



8. Set the Interactive input: To Add labels, set to **Add**. To Delete Labels, set to **Delete**.

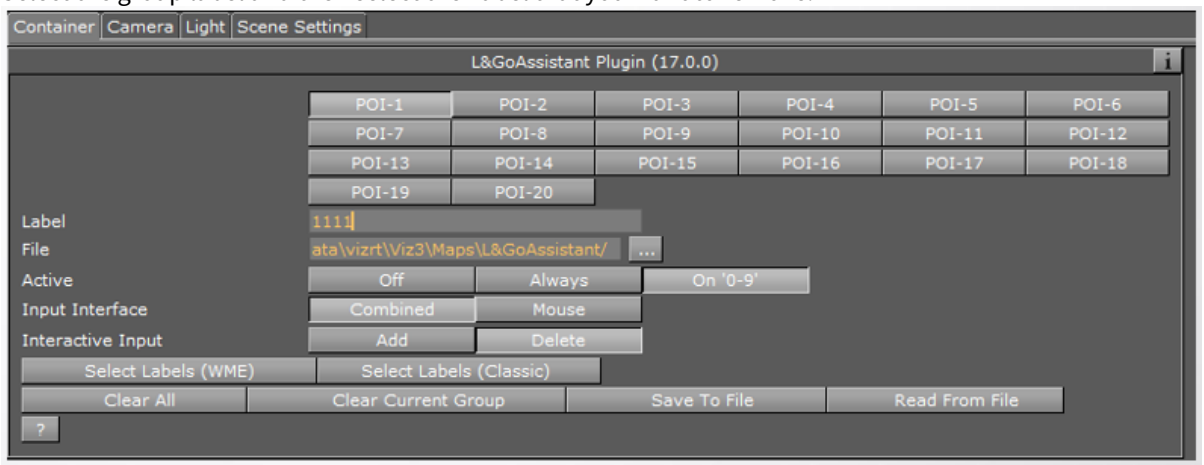
Copyright © 2026 Vizrt

9. To Add Labels from group number 1: Navigate to the specific area, press **1** on the keyboard and click the left button on the mouse. Every time you click a label appears. To change to group 2: Press **2** and click on the mouse.



Deleting User Labels

1. Set the Interactive input to **Delete**.
2. Select the group label and then select the Label that you want to remove.



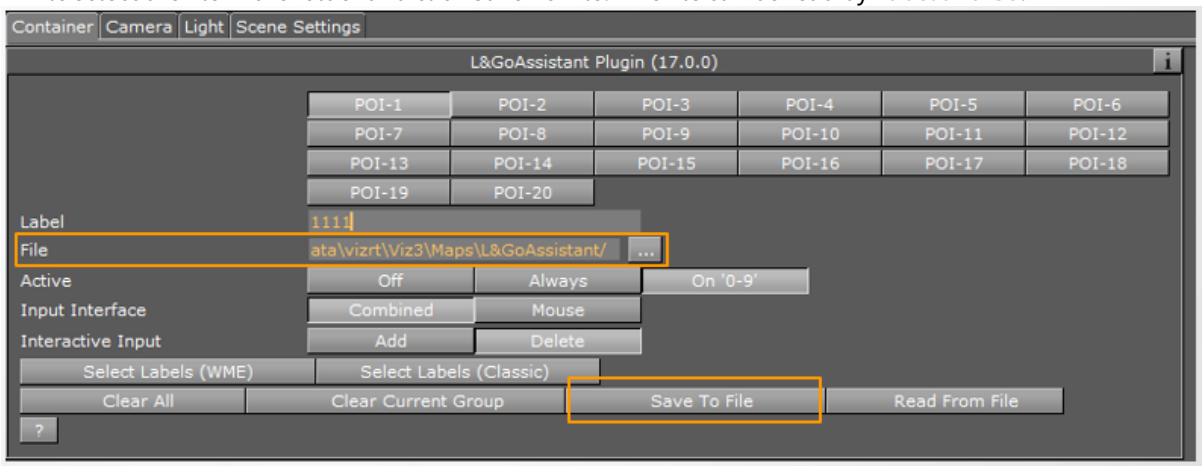
3. In order to delete a group of Labels: Press the Label group number in the keyboard to select the all group, Press **SHIFT** to draw a rectangle around the labels that you want to remove.

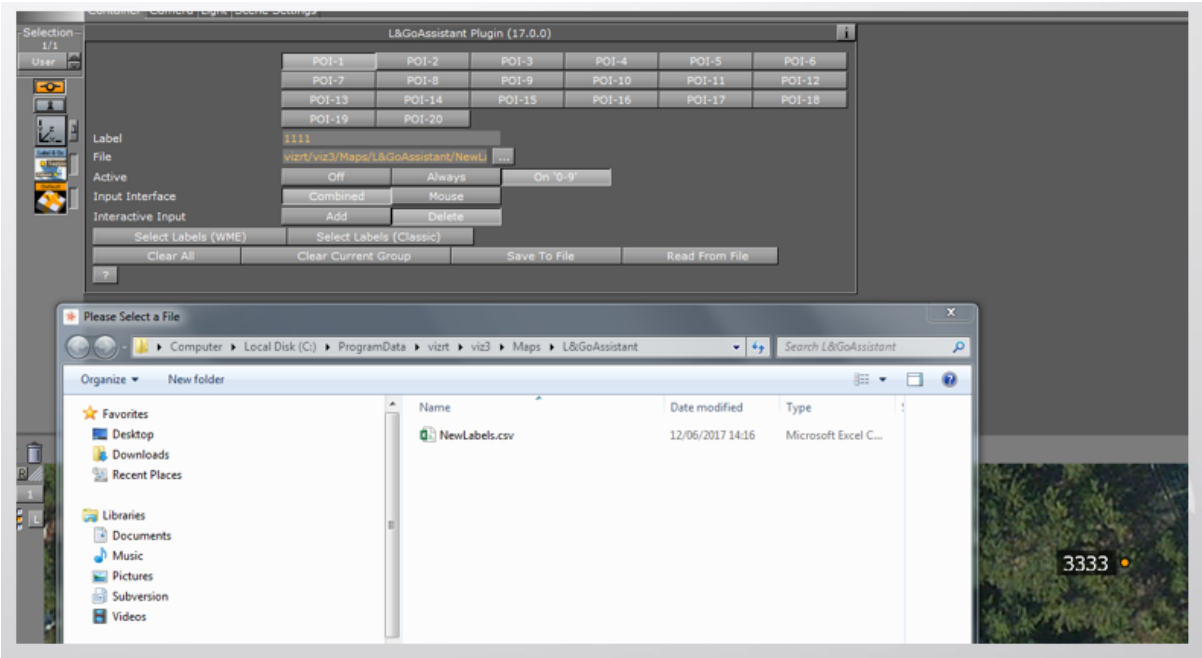
Tip: By pressing **E** in the keyboard you can switch from edit mode to none to see the actual results.

Copyright © 2026 Vizrt

Saving New Labels to a File

1. Navigate to Folder: *%ProgramData%\vizrt\VizEngine\Maps\L&GoAssistant/.*
2. Create a *.txt* document and change it to *.csv*.
3. In File select the file in the folder and click Save To File. This file can be read by Label and Go.

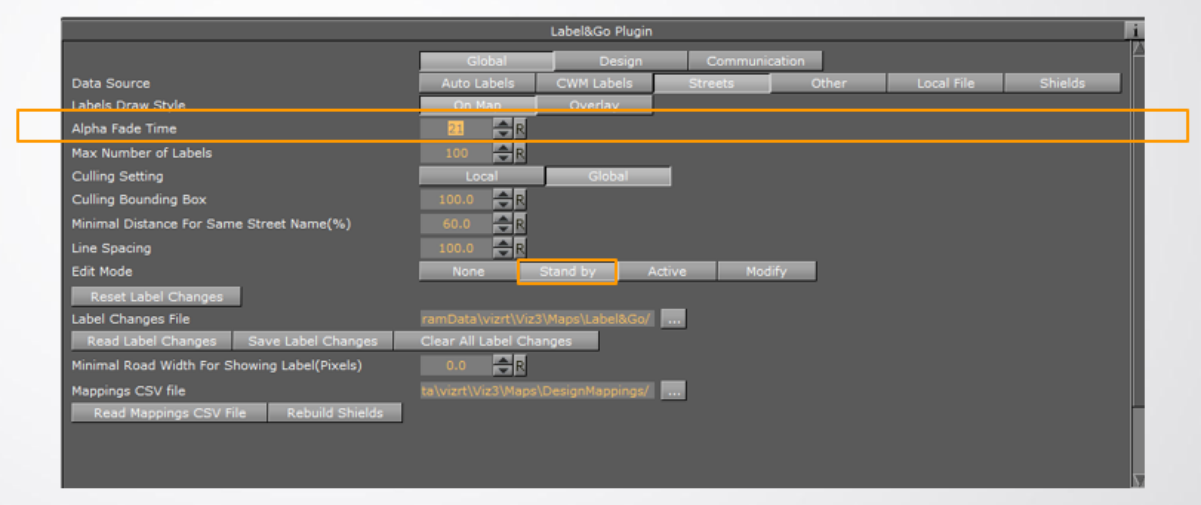




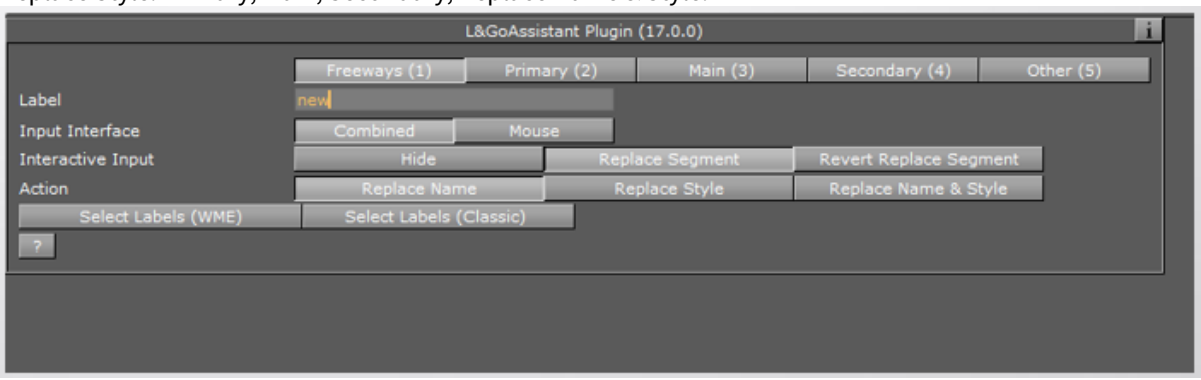
Editing a Label Using the Label and Go Assistant Plug-in

1. We can change a name of a Road or a street. In Label and Go, set the Data Source to Streets. Set the Edit Mode to Stand By. Zoom to the area that you want to edit the Labels.

Copyright © 2026 Vizrt



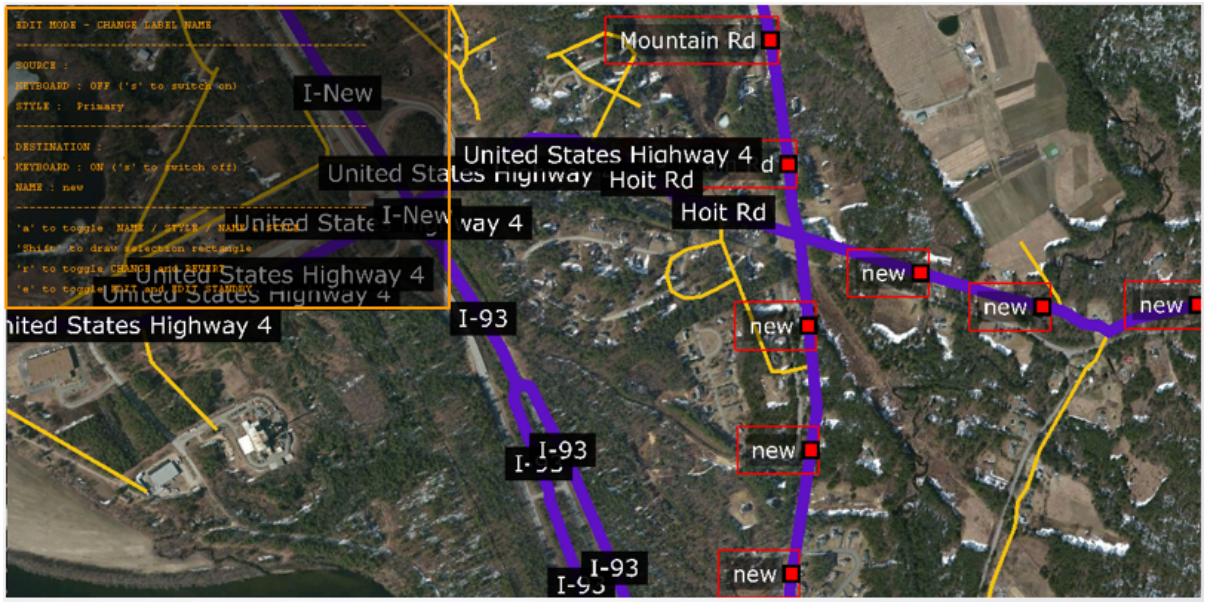
2. Change the Edit Mode to Active.
3. Select the Interactive Input: Replace Segment, Select the Action: Replace Name: Name of the segment, Replace Style: Primary, Main, Secondary, Replace Name & Style.



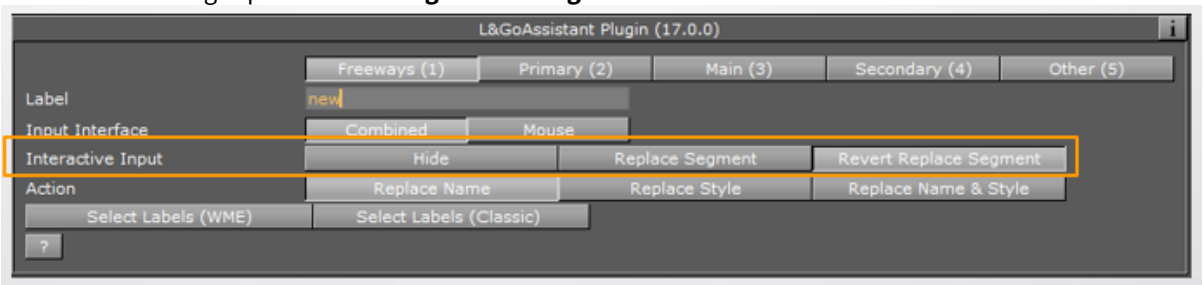
Tip: You can toggle between the options pressing **A** in the keyboard.

4. When you are in Active mode you have an information panel on the left side of the screen.

Copyright © 2026 Vizrt

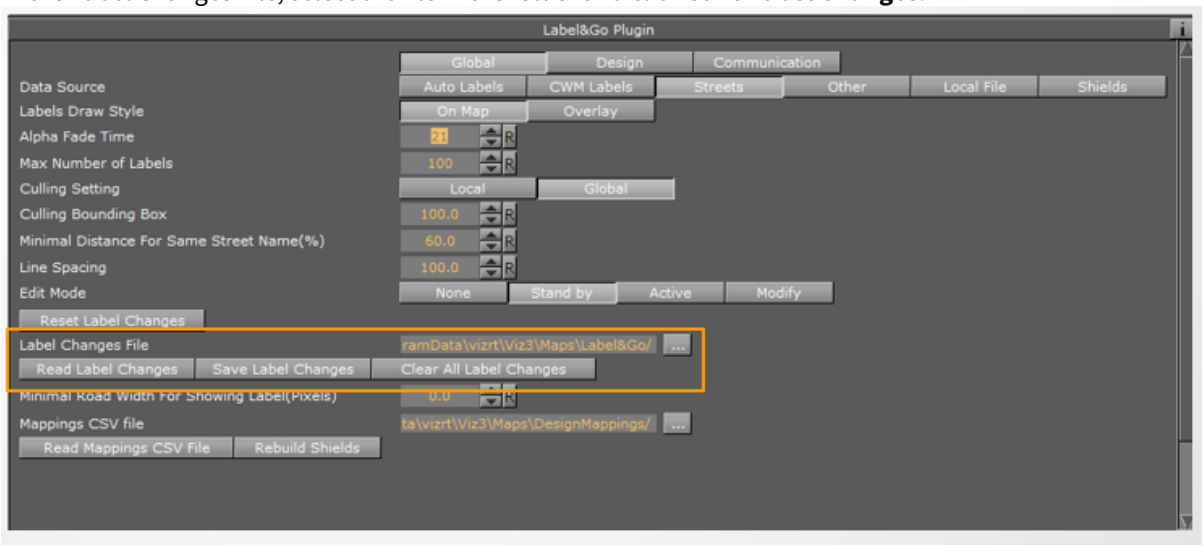


5. To revert the changes press **Revert Segment Changes**.



Saving the Label Changes to a File

1. Enter to Folder: *%ProgramData%\vizrt\VizEngine\Maps\L&GoAssistant/*.
2. Create a text file and rename it to *.csv*.
3. In the Label Changes File, select the file in the folder and click **Save Label Changes**.



Copyright © 2026 Vizrt

By pressing the **Clear All Label Changes,** all changes are cleared, you can Load the file by pressing **Read Label Changes**.

Copyright © 2026 Vizrt

5.4.19 Label It



The Label It plug-in manages 3D labels and place indicators.

The plug-in creates a hierarchy under its container for adding a caption, body and pointer objects.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps

Label It Properties

Caption

Caption defines the relation between the label components (text and pointer). When disabled, duplicated labels use the label design exactly. When enabled, the labels use the following options:

- **Caption Pivot:** Defines the connection point between the caption and the body when the caption is moved (rotated along the tip). There are two options, **Fixed** and **Edge**.
  - **Fixed:** Sets the connection point at the center of the caption.
  - **Edge:** Sets the connection point depending on the correct rotation. If the caption is above the tip, the connection point is at the middle lower side of the caption. If the caption is on the right side of the tip, it is on the left side of the caption. For example: A line moving from the tip to the center of the caption and where it crosses the caption bounding box is the connection point.
- **Rotate Labels:** Defines whether labels, created in Viz (3D labels), are rotated like the labels in the Map Editor. When set to `Off`, all labels are displayed horizontally. When set to `On`, labels that were rotated in the Map Editor are also rotated in Viz.
- **Caption Source:** The options **Default Position**, **WME Directions**, and **Label Manager Presets** include the properties described below.

Default Position

This is used when adding labels in WME when no direction is selected.

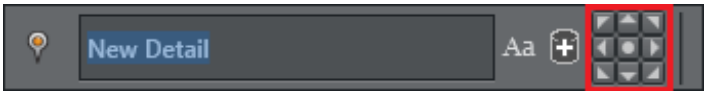
- **Direction (deg):** Sets the angle of the label in relation to its geographic position.
- **Distance (cm):** Sets the distance of the label from its geographical position.
- **WME Offsets Distance:** Enables the WME to offset the distance when set to `On` (default). When set to `Off`, only the direction offsets from WME are used and the distance is ignored.

WME Directions

This allows you to set and fine tune offsets for every direction available in WME. When selected, the labels are placed as they were placed on the map in the Map Editor. When manually set inside WME they always take priority over the presets and Default Position settings. The default pointer direction (angle) and default distance between the pointer and the label is set by the map label parameters and used by the Map Editor (WME). If the direction and distance are changed while using the WME, the *Default Direction* and *Default Distance* values are ignored.

- **WME Compass Rose:** Corresponds to the available directions inside the WME, as shown in the following image.

Copyright © 2026 Vizrt



- **Direction Offset (Degrees):** Determines direction offset for fine-tuning the position of the label.
- **Distance (Viz units):** Determines the distance offset for fine-tuning the position of the label.

Label Manger Presets

When this is selected, Label Manager uses the defined presets to place the labels over the map. Label Manager optimizes the label position such that the labels do not overlap.

- **Number of Presets:** Defines the number of label position presets available to the user (one to four presets). This parameter is only enabled when Caption Mode is set to Controlled.
- **Current Preset:** Selects the preset number to be configured, using the Direction and Distance parameters. Each preset should be selected and the label position should be adjusted.
- **Direction (deg):** Sets the angle of the label in relation to its geographic position.
- **Distance (cm):** Sets the distance of the label from its geographical position.
- **Collision Mode:** Defines how the labels are placed when an overlap or collision between two labels occur. With Tip, the pointers of overlapping labels can cross or touch, but no overlap of label bodies are allowed. With Bounding Box, a bounding box is calculated around the entire label (label body and pointer). An overlap between labels bounding boxes is not allowed.

Overlay

Navigator Overlay defines how the label is displayed over the map. Available options are Disabled, Fixed, Scaling, Near Scale and Far Scale.

- **Disabled:** Places the label on the map using its geographical referencing.
- **Fixed:** Places the label by keeping its geographical referencing, but using a different camera (either with dynamic image or with a front layer). The label size remains fixed.
- **Scaling:** Places the label by keeping its geographical referencing, but using a different camera (either with dynamic image or with a front layer). The label scales to imitate the camera movement.
    - **Near Scale:** Defines the maximum size of the label on the screen (the final size of the label when zooming in).
    - **Far Scale:** Defines the minimum size of the label on the screen (the final size of the label when zooming out).

On Map Scale

- **Static Map Scale:** Defines a scaling factor for the duplicated labels on a static map without camera movement when set to `On`. When set to `Off`, no scaling is used.

Copyright © 2026 Vizrt

Fade

Stand Alone

- **Fade:** Defines the fade effect parameters to be used with the duplicated labels. Available options are **Stand Alone** and **Controlled**. Stand Alone enables the user to define the fade parameters of the labels, while Controlled can be enabled to define the fade parameters when working with Label Manager.
- **Fade on Time:** Defines a label fade effect, beginning at a relative point to the defined hop duration. An additional parameter is enabled, **Time to Hop**, defining when the fade occurs.
- **Fade on Distance:** Defines a label fade effect, beginning at a relative distance from the hop final location.

> **Note:** The *Fade On Distance* parameter is enabled only if *Overlay* is set to *Scaling*.

- **Fade on Lat/Long:** Defines a label fade effect, beginning at a Longitude and Latitude offset from the hop final location. An additional parameter is enabled, **Lat/Long**, defining the offset from in degrees.

Controlled

- **Step:** Controls when the label fades in and out in relation to an animation. In general the fade can be based on the camera distance (for example; capitals are in view when distance is below 1000KM) or on timing in relation to the hop:
  - **Auto:** Fades in and out based on distance to hop when a label is of type point (added by the user). If the label is of type place/region, it fades in and out based on the distance set in Label Manager. If the hop is not close enough for the label to show and the label was added by the user, it fades in based on hop timing and not distance.
  - **On Hop:** Links the fade to the hop timing.
  - **Point 1/Point 2:** Reserves labels where the distance is configured by Label Manager.
  - **Hop and Above:** Turns on at the hop and stays on thereafter.
- **Selected Label Timings:** Sets one of the timing options for the fade to occur:
  - **At End:** Fades the label after the animation stops and before it continues.
  - **Close To End:** Fades the label just before the animation stops and just after the animation continues.
  - **Ahead:** Fades the label before the animation stops and after the animation continues.
  - **Well Ahead:** Fades the label long before the animation stops and long after the animation continues.
- **Label Priority:** Sets the duplicated labels priority. This parameter is useful for a large number of onscreen labels and not all can be displayed. The priority levels define which labels are displayed and which labels are hidden.
  - **Auto:** Allows Label Manager to set priorities.
  - **Normal**, **High:** Sets priorities to normal and high.
  - **Always:** Displays labels every time.

Practical Use

When adding a Label It plug-in, three containers are added under the Label It container: *CAPTION*, *BODY* and *TIP*. Drag a text object under the *CAPTION* container. The *BODY* container is the pointer's body and the *TIP* is the pointer's end.

Copyright © 2026 Vizrt

IMPORTANT! A text object under the label design must be named *label* to receive the name of the object from CWMClient.

Copyright © 2026 Vizrt

5.4.20 LatLongGrid



The LatLongGrid plug-in draws latitude and longitude lines on the map.

The lines are generated using 3D Line.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

LatLongGrid Properties

- **Spacing**
  - **20/10/5 Degrees:** Spacing in degrees for each line in the grid.
  - **Manual:** Defines the **Spacing Units** (Degrees or Meters) using manual input and defines separate **spacing for longitude** and latitude lines and the **Origin Latitude/Longitude** (starting point for the grid).
- **Prune Longitude Lines at Pole:** Continues to draw lines at the poles instead of pruning (cropping) the geometry at the poles when Disabled.
- **Latitude/Longitude Designs:** When set to **Single**, both longitude and latitude lines are drawn using the same 3D Line plug-in. When set to **Separate** it allows using separate 3D Line plug-ins for each to achieve a different look for latitude and longitude lines.

Copyright © 2026 Vizrt

5.4.21 Locator Control



The Locator Control plug-in links an object positioned on one map to another map, showing its position.

The object marking the point on the linked map must reside under a geographically referenced map. The linked object shows the position of the linked map or object on the map its parent map.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps

Locator Control Properties

- **Position Type:** Defines the link between the object and the map. Available options are **Map**, **Navigator**, **WPosition**, **WPoint** and **User**.

Map

Map links the object to a map, showing its entire area over the parent map.

- **Remote Container:** Defines the linked map container.
- **Control Size:** Marks the entire area of the linked map on the parent map when set to `Off`. When set to `On`, the size of the marked area can be controlled.
- **Basic Size:** Defines the percentage of the linked map area to be marked.
- **Min Size:** Defines the minimum size of the linked map area to be marked.
- **Max Size:** Defines the maximum size of the linked map area to be marked.
- **Shared Memory Link:** Defines which is the master (source) and client (target) when the same Shared Memory Identifier is used.
- **Type:** Defines the accessibility of the shared memory.
  - **Global:** Makes shared memory accessible to all scenes currently loaded in memory. This is useful when working with Transition Logic scenes where your Viz World map can be one scene and the locator a different one and data can easily be transferred between the two.
  - **Scene:** Makes shared memory accessible locally and to the current scene. Every scene has one shared memory map that can be used to exchange data among the scripts in the scene.
  - **Distributed:** Makes shared memory accessible to all computers connected to one Viz Graphic Hub.
- **Identifier:** Defines the identifier for the shared memory map.

Navigator

Navigator links the object to a Navigator plug-in, showing its position on the parent map. The plug-in locates the navigator container above it in the scene tree. The Navigator tab displays the same settings as Map with the exception of the Remote Container setting.

WPosition

WPosition links the object to a World Position plug-in, showing its position on the parent map. **Remote Container** defines the linked World Position container.

Copyright © 2026 Vizrt

WPoint

WPoint links the object to a WPoint plug-in, showing its position on the parent map (the WPoint plug-in is used in Viz Weather). **Remote Container** defines the linked WPoint container.

User

User enables the user to position the object on the map.

- **Map Center Longitude:** Defines a Longitude value for the object.
- **Map Center Latitude:** Defines a Latitude value for the object.
- **Tangent To Globe:** Rotates the controlled object to match the globe surface when set to `On`.

> ⚠ **Note:** Tangent To Globe is only visible when the locator is under a globe object.

Copyright © 2026 Vizrt

5.4.22 Map Layers



The Map Layers plug-in exposes map layers to an external application.

Map layers are labels, regions, and so on.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

Map Layers Properties

- **Mode:** Defines the layer maps mode Static and Dynamic.
  - **Static:** Enables maps to be imported to Viz Artist's image library and saved with the scene.
  - **Dynamic:** Enables pyramid maps to be loaded from the cache and temporary folder. When the CWMClient plug-in receives a new map, all layers are updated.
- **Texture Compression:** Sets the compression level for the texture (DTX5 is the highest compression level, which is less texture quality).
- **Texture Quality:** Sets linear for using the same image resolution in the entire zoom range, or Mipmap to change resolution according to the distance from the image (managed automatically in Viz Artist).
- **Texture Mapping:** Defines how the texture (map) is mapped.
- **Texture Anisotropic Filter:** Turns on the relevant anisotropic in the image on the same container with CWMClient (similar to Mipmap, above). Available settings are `Off` / `2x` / `4x` / `8x` / `16x`.
- **Edited Layer:** Sets the layer number for editing. Each layer is then assigned map properties to display.
- **Active:** Activates or disables the selected layer.
- **Container:** Assigns the container for holding the created layer map.
- **Flags:** Enables (`On`) the requested property to expose the map property or feature in the selected layer. *Flags* refer to all the settings from Base Map to Overlay Data.
- **Build Tiles:** Builds the map layers.

Copyright © 2026 Vizrt

5.4.23 Map Layers Control



The Map Layers Control plug-in controls map layers in an individual map and in map tiles (Pyramid or Map Tiler tiles).

The plug-in should reside on the CWMClient container which generates the maps and tiles.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps

Map Layers Control Properties

Select an option to configure: **Map** or **Tiles**. Each can be enabled and configured separately.

- **Control:** Enables layer parameters when enabled. See Map Layers for the layers description.

Copyright © 2026 Vizrt

5.4.24 Map Tiler



The Map Tiler plug-in builds a set of map tiles for displaying an area in high resolution. The maps are created as GeoImage (flat map) or Globe objects and placed under a low resolution map of the entire area for geographical referencing. Map Tiler is also used for managing Pyramid object maps when used in a scene.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

Map Tiler Properties

Default

- **Selected Region:** Defines the area of the map for which the tiles are created.
  - **World:** Creates tiles for the entire world.
  - **Selected Map:** Creates tiles for the area of the selected map in CWMClient attached to the Map Tiler container.
  - **Region List:** Displays a list of regions. An item from the list can be selected to create map tiles for the selected area.
  - **Long Lat:** Sets the Longitude and Latitude minimum and maximum values. The map tiles are created for the defined area. When Long Lat option is selected, additional fields are enabled.
- **Map Status:** Displays plug-in status messages. When configuring the tiles, the Map Status field displays the number of tiles required and tile size according to the current plug-in setup.
- **Requested Number of Tiles (2K2K):** Allows the user to set the number of tiles to create.
- **Geometry Type:** Selects the object type of the created tiles.
  - **Globe:** Creates the tiles using Globe.
  - **Flat:** Creates the tiles GeoImage.

Selected Map

- **Texture Compression:** Sets the compression level for the texture (DTX5 is the highest compression level which is less texture quality).
- **Get Map Resolution/(Classic):** Uses the WME Editor to zoom to the closest level you want the tiles to support. If you change the extent (in Selected Map mode) you need to set the resolution again.

Region List

- **Region Area:** Selects the region you want to build tiles for.

Long Lat

- **West:** Sets the Longitude value for the western edge of the map.
- **East:** Sets the Longitude value for the eastern edge of the map.
- **South:** Sets the Latitude value for the southern edge of the map.
- **North:** Sets the Latitude value for the northern edge of the map.

Copyright © 2026 Vizrt

Custom

The following additional parameters are available:

- **Radius/Map Size:** Sets the size (in Viz units) of the globe or map tiles (all together).
- **Tessellation:** Determines the number of polygons used in the object. The higher the number is, the smoother the object is drawn.
- **Texture Quality:** Uses the same image resolution in the entire zoom range when set to **Linear**. When set to **Mipmap** the resolution is changed according to the distance from the image (managed automatically in Viz Artist).
- **Overlap (%):** Determines the overlapping percentage for each tile.
- **Build Terrain:** Builds terrain data for the created tiles.
- **Smooth Terrain:** Smoothens terrain edges so they blend with the base map surface. When set to `On`, the Terrain Smoothing Factor is enabled.
- **Terrain Smoothing Factor:** Sets the percentage of the smoothed area.
- **Terrain Height Scale:** Sets the scaling factor for terrain height. The higher the factor is, the more extreme the terrain is.
- **Terrain Resolution:** Sets the total terrain resolution for all tiles. A high resolution value results in a more detailed terrain (affecting performance).

Buttons

- **Build Tiles:** Builds the geo-reference map and tiles.
- **Change All Maps:** Builds the geographical reference map, tiles and applies style changes made in CWMClient to all child containers (recursively) under the Map Tiler container.
- **Build Tiles (Force New):** Builds the geographical reference map and tiles without checking the cache for existing maps (from the server).
- **Change All Maps (Force New):** Builds the geographical reference map, tiles and applies style changes made in CWMClient to all child containers (recursively) under the Map Tiler container, without checking the cache for existing maps (from the server).
- **Calculate Tiles Info:** Calculates the tiles information, without building the tiles, and displays it in the Map Status field.

Copyright © 2026 Vizrt

5.4.25 Map Zoom



The Map Zoom plug-in builds single destination animated scenes. Because the scene is treated as an independent Map Object it can be added into any graphic including Transition Logic scenes.

The map is created as GeoImage (flat map) and adds all necessary plug-ins to the hierarchy.

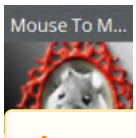
> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps

Map Zoom Properties

- **Zoom:** Defines the zoom in percent (`0-100%`), in order to animate to the final destination.
- **Labels:** Takes the size of the label from the original design when set to **Fixed**. When set to **Auto Scale**, the size is calculated based on the size of the map.

Copyright © 2026 Vizrt

5.4.26 Mouse2Memory



The Mouse2Memory plug-in sends mouse events to shared memory.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Tools

Mouse2Memory Properties

Shared Memory

- **Map Type:** Defines the accessibility of the shared memory.
  - **Global:** Makes shared memory accessible to all scenes currently loaded in memory. This is useful when working with Transition Logic scenes where your Viz World map can be one scene and the locator a different one and data can easily be transferred between the two.
  - **Scene:** Makes shared memory accessible locally and to the current scene. Every scene has one shared memory map that can be used to exchange data among the scripts in the scene.
  - **Distributed:** Makes shared memory accessible to all computers connected to one Graphic Hub.
- **Reset Before Change:** Resets the data before sending (ensures the same data is not sent twice due to shared memory blocking it).
- **Key:** Defines the shared memory key name that receives the data field value.
- **Value:** Defines the data field value for the shared memory key.
- **Execute:** Sends data to shared memory.

Picking

- **Camera:** Determines the camera to use for the projection.
- **Picking Active:** Limits picking to a rectangle.
- **Position X (%):** Determines X value of rectangle.
- **Position Y (%):** Determines Y value of rectangle.
- **Width (%):** Determines width of rectangle.
- **Height (%):** Determines height of rectangle.
- **Preview:** Displays the rectangle.
- **Execute:** Sends data.

Copyright © 2026 Vizrt

5.4.27 Mute



The Mute plug-in determines if a container is drawn or not. The difference between using Mute and setting the visibility off is that when using the plug-in only containers with the plug-in are not drawn, but all child objects are drawn and the geometry properties are used in the hierarchy (geographical reference, and so on). When setting the visibility off, the entire container and its child containers are disabled and not used in any way.

> ⚠ **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Tools

Mute Properties

- **Draw Geometry:** Draws geometry when enabled (`On`). When disabled (`Off`), the geometry is not drawn, but its child containers are drawn.

Copyright © 2026 Vizrt

5.4.28 NavCom



The NavCom plug-in controls the Navigator plug-in. The plug-in is an example showing how to externally control Navigator when special applications should be used to control the scene. Elections are an example of a special case where external control of Navigator is required. When using Viz 3.x scripting, complicated logic and commands can be used with the NavCom plug-in.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

NavCom Properties

Commands

The Commands tab is used for communicating with Navigator in means of sending formatted commands and receiving formatted responses from Navigator.

The Command text field is where you enter the command. NavCom receives predefined, fixed syntax, commands and returns a reply from Navigator. The Response field contains the reply returned from Navigator. The format of the reply is defined in the Advanced tab. The Immediate Response option defines whether the command is sent while typing (every change to the command field sends the command to Navigator), or only after pressing the **Execute Command** button.

The following is a list of commands, example commands and return values:

- **Mouse To Lon Lat:** Converts screen coordinates to Long/Lat values from the map. This needs Navigator.
  - Command: `MTLL x y`
  - Response: `lon lat`

```
MTLL 360 288 -116.998 38.088
```

- **Mouse To Region:** Returns the region name in which the given screen coordinates reside. This command requires Navigator.
  - Command: `MTR x y`
  - Response: `CountryID "Country Name" RegionID "Region Name" SubRegionID "Sub Region Name"`

```
MTR 360 288 1000000000003600 "United States of America" 1000000000038901
"Texas" 1000000000111302 "Atascosa County"
```

  All **Fly To commands** require Navigator and the NavFinder. Also set Navigator's Interactive Anim setting to `On`.
- **Fly To Region ID:** Triggers a Navigator animation from the current location to the sent region (RegionID).
  - Command: `FTRID RegionID`

Copyright © 2026 Vizrt

```
FTRID 81000000000F8285
```

- **Fly To Country:** Triggers a Navigator animation from the current location to the sent country (country name).
  - Command: `FTC "country name"`

```
FTC "united states of America"
```

- **Fly To Region:** Triggers a Navigator animation from the current location to the sent region (Region name).
  - Command: `FTR "country" "Region"`

```
FTR "united states of America" "Texas"
```

- **Fly To Sub Region:** Triggers a Navigator animation from the current location to the sent sub region (by name).
  - Command: `FTSR "country" "Region" "Sub Region"`

```
FTSR "united states of America" "Texas" "young county"
```

- **Fly To Lon Lat Level:** Triggers a Navigator animation from the current location to the sent level area (country, region, or sub region) that the long/lat data resides in.
  - Command: `FTLLL lon lat level`

```
FTLLL -116.998 38.088 1
```

⚠ **Note:** Levels are: 0 = Country, 1 = Region, 2 = Sub Region

- **Search database:** Searches the map database and as a response generates an XML. This command requires a running Viz World server.
  - Command: `SMDB <query>`

  Search web: Searches the web and as a response generates an XML. This command requires a running Viz World server.
  - Command: `SWEB <query>`
- **Search All:** Searches both resources (map database and web) and as a response generates an XML. This command requires a running Viz World server.
  - Command: `SALL <query>`
  - Shared memory reply: `WS_REPLY`
- **Nearby:** Searches locations in the vicinity of the coordinates of your “mouse” and as a response generates an XML. This command requires a running Viz World server. This command requires Navigator.
  - Command: `NEARBY [mouse-x] [mouse-y]`

Copyright © 2026 Vizrt

    - Shared memory reply: `NAV_NEARBY_REPLY`

      Shared **Navigation** memory of Navigator can be activated by setting **Publish Geo Data** to **Bounding box** or **Location**. If Bounding box is selected you get, in addition to Bounding box data, Center and Distance data. If Location is selected you get, in addition to the Location data, all the others. If None is selected, no data is shared. All can be used with scripting to define e.g. the administration levels the script should fetch from the shared data and consequently display it on screen. For interactive scenes such as touch screens this can be used for adding graphics to maps based on your selection (thereby avoiding over-populating the map with information). For testing you can enable the interactive mode by clicking the **E** button in Viz Artist's Scene Editor.
- **Navigation location:** Shared data holding the current geographic location(s) of Navigator. To enable it, select Publish Geo Data's Location option. The data is updated when the camera is not moving. Navigation location also generates an XML with the current geographic location(s) (in the same format as `SMDB`, `SWEB`, `SALL` and `NEARBY`). This requires a running Viz World Server.
    - Name: `NAV_LOCATION_DATA`
- **Navigation bounding box:** Shared memory data holding the current (frame by frame) geographic bounding box (lat lon) of Navigator. To enable it, select Publish Geo Data's Bounding box or Location option.
    - Reply: `NAV_BB_DATA`
- **Navigation center:** Shared memory data holding the current (frame by frame) geographic center of Navigator. To enable it, select Publish Geo Data's Bounding box or Location option.
    - Reply: `NAV_CENTER_DATA`
- **Navigation distance:** Shared memory with the current distance to world of Navigator. To enable it, select Publish Geo Data's Bounding box or Location option.
    - Reply: `NAV_DISTANCE_DATA`
- **Navigation direction:** Shared memory with the current pan and tilt direction of Navigator. To enable it, select Publish Geo Data's Bounding box or Location option.
    - Reply: `NAV_DIRECTION_DATA`

Control

The Control tab is used to directly control Navigator by changing the values of the parameters.

- **Longitude:** Sets the longitude value for Navigator. The Navigator camera moves to the specified longitude value.
- **Latitude:** Sets the latitude value for Navigator. The Navigator camera moves to the specified latitude value.
- **Distance:** Sets the Navigator distance from the map. The Navigator camera moves to the specified distance value.
- **Pan:** Sets the pan value of the Navigator camera.
- **Tilt:** Sets the tilt value of the Navigator camera.
- **Interactive Mode:** Defines the interactive behavior of Viz.
    - **None:** Disables interactive mode.
    - **Editor:** Enables interactive mode during scene editing.
    - **On Air:** Enables interactive mode when Viz Engine is in On Air mode.
    - **Always:** Enables interactive mode during scene editing and when Viz Engine is in On Air mode.
- **Minimum Distance:** Sets the minimum distance from the map in Viz units.
- **Zoom Center:** Used in interactive mode.

Copyright © 2026 Vizrt

    - **Position:** Zooms the map to center on the position of the fingers.
    - **Center:** Zooms to the center of the map.
- **Momentum Mode:** Sets how much **friction** should be applied to the momentum when set to Linear.

> ⚠ **Note:** The Navigator control parameters affect the Navigator immediately.

Advanced

The Advanced tab is used for defining NavCom's general behavior patterns.

- **Shape File Support:** Defines whether NavCom includes shape objects information in the response. The Shape objects must reside under the NavCom container.
    - **Disabled:** Does not return shape object information.
    - **Limited:** Scans the shape files below the container. When a command is received, NavCom returns information about the shapes that overlap the requested data in the command in the response string.
    - **Only:** Scans the shape files below the container. When a command is received, NavCom returns only the information about the shapes that overlap the requested data in the command in the response string.
- **Limit Picking:** Limits the geographical area that NavCom performs any of the commands on. When Limit Picking is enabled the Limit to Country parameter is enabled.
    - **Limit to Country:** Sets the country name. If the requested data in the command does not exist for the defined country area, the command is not executed. Only commands relating to points within the country area are executed.
- **Shared Memory Prefix:** Distinguishes between the data sent to each plug-in when using more than one NavCom plug-in in the scene, and when using scripting to send and receive information from these plug-ins. In each NavCom plug-in, set the Shared Memory Prefix to a different prefix (up to four).
- **Protocol Mode:** Defines the data displayed in the Response field.
    - **Full:** Includes data on names and IDs for countries, regions and sub-regions.
    - **Name:** Includes data on names for the countries, regions and sub-regions.
    - **ID:** Includes only ID data (as defined in the Viz World Server) for the countries, regions and sub-regions.
- **Protocol Separator:** Defines the separating character between the Response data. The defined separator is used between country and region, region and sub-region, and so on.
- **Use Quotes:** Defines whether the response string is quoted or not.
- **Suffix FIPS:** Used for USA regions and sub-regions.
- **Publish Geo Data:** Allows you to publish navigation data to shared memory based on your interaction with the graphics scene (for example, touchscreen or mouse). Shared Navigation memory can be activated by setting to **Bounding box** or **Location**. See the available Navigation data under the Commands section.
    - **Bounding box:** Sends Center and Distance data in addition to Bounding box data.
    - **Location:** Sends all the others in addition to the Location data.
    - **None:** Does not share data.
- **Execute Command:** Executes the defined command when pressed.
- **Initialize:** Rescans the shape objects if the shape objects under the NavCom container are changed. This button is used when using shape file support.

Copyright © 2026 Vizrt

NavCom Scripting

Viz 3.X scripting ability is a powerful tool for implementing complex logic into a scene. In a Navigator scene, NavCom can be used in the scripts to enable such advanced Navigator operations. The following script example demonstrates how to send and receive data from Navigator through to NavCom.

Example: Script example: main.txt

```
dim level as Integer
dim ignore as Integer
dim CurCountry as string
dim CurCountryId as string
dim CurState as string
dim CurStateId as string
dim CurCounty as string
dim CurCountyId as string

Sub UpdateCurrent(temp As String)
    dim position as Integer
    dim temp2 as String
    println temp
    CurCountryId = temp.left(16)
    position = temp.Find("\"")
    temp = temp.GetSubstring(position+1,temp.Length-(position+1) )
    temp2 = temp position = temp2.Find("\"")
    CurCountry = temp2.Left(position)
    println "<" & CurCountryId & "><" & CurCountry & ">"
    position = temp.Find("\"")
    temp = temp.GetSubstring(position+2,temp.Length-(position+2) )
    CurStateId = temp.left(16)
    position = temp.Find("\"")
    temp = temp.GetSubstring(position+1,temp.Length-(position+1) )
    temp2 = temp
    position = temp2.Find("\"")
    CurState = temp2.Left(position)
    println "<" & CurStateId & "><" & CurState & ">"
     position = temp.Find("\"")
    temp = temp.GetSubstring(position+2,temp.Length-(position+2) )
    CurCountyId = temp.left(16)
    position = temp.Find("\"")
    temp = temp.GetSubstring(position+1,temp.Length-(position+1) )
    temp2 = temp
    position = temp2.Find("\"")
    CurCounty = temp2.Left(position)
    println "<" & CurCountyId & "><" & CurCounty & ">"
End Sub

Sub OnInit()
    ignore = 0
     level = 1
```

Copyright © 2026 Vizrt

```
     if Scene.Map.ContainsKey("MTR") = false Then
        Scene.Map.CreateKey("MTR")
    End If
    if Scene.Map.ContainsKey("FTRID") = false Then
        Scene.Map.CreateKey("FTRID")
    End If
    if Scene.Map.ContainsKey("FTC") = false Then
        Scene.Map.CreateKey("FTC")
    End If
    if Scene.Map.ContainsKey("FTR") = false Then
        Scene.Map.CreateKey("FTR")
    End If
    if Scene.Map.ContainsKey("FTSR") = false Then
        Scene.Map.CreateKey("FTSR")
    End If
    if Scene.Map.ContainsKey("FTLLL") = false Then
        Scene.Map.CreateKey("FTLLL")
    End If
    if Scene.Map.ContainsKey("MTR_REPLY") = false Then
        Scene.Map.CreateKey("MTR_REPLY")
    End If
    if Scene.Map.ContainsKey("FTRID_REPLY") = false Then
        Scene.Map.CreateKey("FTRID_REPLY")
    End If
    Scene.Map.RegisterChangedCallback("MTR_REPLY")
    System.Map.RegisterChangedCallback("REGION_L")
 End Sub

Sub OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)
     println mapKey
    dim temp as String
    If mapKey = "REGION_L" Then
       println "ignore pre"
       ignore = 1
    else If mapKey = "MTR_REPLY" Then
       temp = Scene.Map["MTR_REPLY"]
       UpdateCurrent(temp)
       println System.Map["REGION_L"]
          if System.Map["REGION_L"] = "1" then
          Scene.Map["FTRID"] = CurCountryId
       elseif System.Map["REGION_L"] = "2" then
          Scene.Map["FTRID"] = CurStateId
       elseif System.Map["REGION_L"] = "3" then
          Scene.Map["FTRID"] = CurCountyId
       End If
    End If
    If mapKey = "MTLL_REPLY" Then
        println Scene.Map["MTLL_REPLY"]
    End If
 End Sub

sub OnLButtonDown()
    dim temp as String
```

```
    if System.MouseX < (System.RenderWindowWidth: 120 ) then
       Scene.Map["MTR"] = (String)
 System.MouseX & " " & (String)System.MouseY
    end if
end sub
```

5.4.29 NavFade



The NavFade plug-in defines the visibility of an object that the NavFade is attached to in a Navigator scene. The Navigator point of view (distance from the map) determines when the object becomes visible. The NavFade uses an alpha plug-in to control the object's appearance. The alpha plug-in is added automatically when adding NavFade to the container.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

NavFade Properties

Height

The object fades in or out depending on camera height (above the map).

- **Fade Invert:** Fades starting at 100% alpha to 0% alpha, instead of from 0% alpha to 100% when set to `On`.
- **Fade Time:** Defines the fade transition duration in fields.
- **Visibility Range:** Defines how the fade point is calculated.
  - **Below:** Makes the object visible when the defined height is higher than the current Navigator distance from the map.
  - **Above:** Makes the object visible when the defined height is lower than the current Navigator distance from the map.
  - **Range:** Makes the object visible when the defined height is between the defined low and high values.

Below or Above

When Visibility Range parameter is set to Below or Above:

- **Height:** Defines the height value that NavFade uses as the show/hide point of the object when *Visibility Range* parameter is set to Below or Above.
- **Set Height:** Inserts the current camera height into the height field when clicked.

Range

When Visibility Range parameter is set to Range:

- **Low/High:** Sets the lowest/highest value of the height range that the object is visible in. If the current Navigator height is between the low and high values, the object is visible.
- **Set Range High/Low:** Copies the current Navigator height value to the High/Low parameter.

Hops

The object fades in or out depending on the defined hop point and the animation time to/from the selected hop point.

- **Fade Invert:** Fades starting at 100% alpha to 0% alpha, instead of from 0% alpha to 100% when set to `On`.
- **Fade Time (fields):** Defines the fade transition duration in fields.

Copyright © 2026 Vizrt

- **Time to Hop (%):** Sets the point in which the object appears/disappears. The time is set as a percentage of the hop duration.
- **Hop Mode:** Defines whether NavFade affects the manually selected hop or if the hop is auto selected when the designs copied by CWMClient.
- **Hop to Link:** Sets the number of the hop points that NavFade uses as a reference. When animating to and from the selected hop, the object appears/disappears.
- **Hop and Above:** Considers all hops with a higher number than the selected hop when enabled. When enabled, all hops which number is higher than the selected hop is considered as the selected hop. Defines whether NavFade affects the manually selected hop or if the hop is selected by NavFade.
  The **Constant** option causes the object to fade on at *Hop to Link* and stay on from that point onwards.
  The **Related** option allows you to relate hops to a set of containers. You can, for example, use NavFinder to define three hops, and then add three text containers as sub-containers of a container holding NavFade. Setting NavFade to Related allows the NavFinder hops to relate to the text containers found under the NavFade container and fade them in and out as part of the hop animation.

Distance

The object fades in or out depending on the distance from the map.

- **Fade Time:** Defines the fade transition duration in fields.
- **Visibility Range:** Defines how the fade point is calculated. Available options are Below, Above and Range.
  - **Below:** Makes the object visible when the defined distance is higher than the current Navigator distance from the map.
  - **Above:** Makes the object visible when the defined distance is lower than the current Navigator distance from the map.
  - **Range:** Makes the object visible when the defined distance is between the defined low and high values.

Below or Above

When Visibility Range parameter is set to Below or Above:

- **Distance:** Defines the distance value that NavFade uses as the show/hide point of the object.
- **Set Distance:** Copies the current Navigator distance value to the Distance parameter.

Range

When Visibility Range parameter is set to Range:

- **Low/High:** Sets the lower/higher value of the distance range that the object is visible in. If the current Navigator distance is between the low and high values, the object is visible.
- **Set Range High/Low:** Copies the current Navigator height value to the High/Low parameter.

Angle

The object fades in or out depending on the angle between the camera and the map.

- **Fade Invert:** Fades starting at 100% alpha to 0% alpha, instead of from 0% alpha to 100% when set to `On`.
- **Fade Time:** Defines the fade transition duration in fields.

Copyright © 2026 Vizrt

- **Visibility Range:** Defines how the fade point is calculated. Available options are Below, Above and Range.
  - **Below:** Makes the object visible when the defined angle is higher than the current Navigator angle between the camera and the map.
  - **Above:** Makes the object visible when the defined angle is lower than the current Navigator angle between the camera and the map.
  - **Range:** Makes the object visible when the defined angle is between the defined low and high values.

Below and Above

When Visibility Range parameter is set to Below or Above:

- **Angle:** Defines the angle value that NavFade uses as the show/hide point of the object.
- **Set Angle:** Copies the current Navigator angle value to the angle parameter.

Range

When Visibility Range parameter is set to Range:

- **Low/High:** Sets the lowest/highest value of the angle range that the object is visible in. If the current Navigator angle is between the low and high values, the object is visible.
- **Set Range High/Low:** Copies the current Navigator height value to the High/Low parameter.

Movement

Alpha values fade on/off based on if Navigator is currently moving or stationary.

Copyright © 2026 Vizrt

5.4.30 NavFinder



The NavFinder plug-in sets hop points over a given map.

The NavFinder must reside under a Navigator plug-in container and a map.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

NavFinder Properties

Common Properties

- **Hop Point:** Sets the hop point in the hop sequence. Note that the animation is built in the same sequential order as the list of Hop Points (Map-Start, Destination-1, Destination-2, and so on). If two hops use the same Hop Point, the animation does not work properly.
- **Position Source:** Defines how the hop location is set. Available options are Map, Absolute, Geometry and Link.

Buttons

- **Goto Map:** Jumps to the defined hop point on the map.
- **Take Current Values:** Takes the current position and uses that as the hop position. This option is normally used with interactive mode in the Navigator plug-in. The user moves the object to the required position and sets the values for the NavFinder.
- **Fly to Map:** Creates an instant animation and run from the current map location to the current hop point defined by the NavFinder plug-in. This feature is active only if the *Interactive Anim* parameter in the Navigator plug-in is enabled (`On`).
- **Get Map:** The user can navigate manually to any destination and by pressing the Get Map button, the currently viewed map is used. CWMClient must be on the same container and if a pyramid plug-in is used, the pyramid layers are built.
- **Add Linked Hop:** Adds a hop (container with NavFinder set to Link) under the current NavFinder container. See Link tab description for more information.

Tip: *Take Current Values* can be used in interactive mode. The user can position itself using the mouse and then tell NavFinder to use the current values.

Map

The Position Source Map uses the current location from the map (center of the screen values) and adds offsets for fine tuning.

- **Fit To Screen By:** Defines what the end frame of the animation (hop) is. When the hop is defined as a 3D Region object on the map, the animation ends when the bounding box of the object fills the frame. Available parameters for defining how the bounding box is calculated are *Width*, *Height*, *Min*, *Max* and *Average*.
    - **Width:** Uses the width of the object's bounding box to calculate the last frame of the hop animation.

Copyright © 2026 Vizrt

  - **Height:** Uses the height of the object's bounding box to calculate the last frame of the hop animation.
  - **Min:** Uses the minimum value between the width and the height of the object to calculate the last frame of the hop animation.
  - **Max:** Uses the maximum value between the width and the height of the object to calculate the last frame of the hop animation.
  - **Average:** Uses the average value between the width and the height of the object to calculate the last frame of the hop animation.
- **Longitude offset:** Defines Longitude offset based on the current position.
- **Latitude offset:** Defines Latitude offset based on the current position.
- **Distance offset (%):** Changes the distance zoom from the map (zoom in or out).
- **Pan:** Sets a pan value for the camera.
- **Tilt:** Sets a tilt value for the camera.

> ⚠ **Note:** Pan and Tilt parameters are disabled unless the *Pan and Tilt Animation* parameter in the Navigator plug-in is enabled (`On`).

Absolute

Absolute sets the Longitude and Latitude values of the hop point location. Changes the Distance and Distance Zoom parameters:

- **Longitude:** Defines the longitude for the hop position.
- **Latitude:** Defines the latitude for the hop position.
- **Distance Offset (%):** Sets an offset to the distance of the camera from the destination.
- **Distance:** Changes the distance from the map.
- **Diameter:** Sets the desired view as diameter (and not distance)
- **Pan:** Sets a pan value for the camera.
- **Tilt:** Sets a tilt value for the camera.

> ⚠ **Note:** Pan and Tilt parameters are disabled unless the *Pan and Tilt Animation* parameter in the Navigator plug-in is enabled (`On`).

Geometry

Geometry uses the current location from the 3D Region plug-in (center of the region values). Add offsets for fine tuning. See the Map and Absolute sections for descriptions of the parameters.

> ⚠ **Note:** Pan and Tilt parameters are disabled unless the *Pan and Tilt Animation* parameter in the Navigator plug-in is enabled (`On`).

Copyright © 2026 Vizrt

Link

The **Link** mode is used when one hop resides as a child of another hop. the child hop is set to link. When changing the top hop, the child hop changes accordingly, maintaining the same animation that was created during the design.

In **Country** and **Region** mode the navigator position is based on the country or region where the destination (link source) is, whereas in **Level up** mode the navigator position is one administration level (e.g. Place is leveled up to Town) above the destination (link source). In Country, Region and Level Up mode you have the following settings:

- **Distance Offset:** Sets an offset from the calculated distance after the top hop was changed.
- **Pan Offset:** Sets an offset from the calculated pan after the top hop was changed.
- **Tilt Offset:** Sets an offset from the calculated tilt after the top hop was changed.

Interpolation mode has the following setting:

- **Interpolation:** Interpolates navigator position between the gap above and below (useful for a drill down with a pause in the middle)

Offset mode has the following settings:

- **Longitude Offset:** Sets an offset from the calculated hop longitude location after the top hop was changed.
- **Latitude Offset:** Sets an offset from the calculated hop latitude location after the top hop was changed.
- **Min Distance/Max Distance:** Sets the minimum and maximum distance from the map. Using the same parameters for both settings gives you a fixed distance. See the Map and Absolute sections for descriptions of the other parameters.

Frame

- **Source <#>:**
- **Source <#> Active:**

Copyright © 2026 Vizrt

5.4.31 Navigator



The Navigator plug-in enables the user to create realistic animations from one point to another on the map (for example fly over a flat map or globe).

It is also used for navigating on a map (moving the camera) to a defined location using pan and tilt values.

Note: Only containers with a NavFinder plug-in are refreshed.

Note: Orthographic cameras are not currently supported by Viz World.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

Navigator Properties

Camera

Position

The **Position** button displays the camera position parameters.

- **Camera:** Defines the camera that is affected by the position parameters.
- **Longitude:** Defines the Longitude camera position on the map/globe.
- **Latitude:** Defines the Latitude camera position on the map/globe.
- **View:** Defines the Camera's view distance from the map.
- **Pan:** Defines the Camera's pan value.
- **Tilt:** Defines the Camera's tilt value.

Advanced

The **Advanced** button displays the advanced camera parameters.

- **Roll:** Sets the roll value of the camera. This parameter is enabled only if the *Keep The Horizon Horizontal* parameter is set `Off`.
- **Keep The Horizon Horizontal:** Maintains the horizon and disables roll movement when set to `On`.
- **View Units:** Sets the way the number entered in the View field (see Position) is treated. *Distance* defines the number of Viz units from the camera to the map. *Visible View* is based on the view (i.e. how many km/miles and so on) you see in units defined by the 3D Map Setting plug-in.
- **Clipping Plane Control:** Defines the distance range that is drawn by the camera. Objects located closer to the camera than the _Near _parameter and objects located farther that the _Far _parameter is not drawn.
    - **Static (Viz):** Draws the objects within the clipping plane values defined in Viz. For Viz 2.x see **Setup > Camera > Camera Clipping Plane**. For Viz 3.x see **Scene Settings > Renderer > Camera Clipping Plane**.

Copyright © 2026 Vizrt

    - **Navigator:** Adjusts the clipping plane values according to the camera position. This is automatically done by the Navigator plug-in based on the *Near Distance* and *Far Distance* parameters.
- **Near Distance (%):** Defines the percentage of the camera distance from the map, which is used as the Near distance of the clipping plane.
- **Far Distance (%):** Defines the percentage of the camera distance from the map, which is used as the Far distance of the clipping plane.
- **Navigation Method:** Allows you to select whether the camera or the container should also move when a map changes position.
    - **Camera:** Moves the camera when the map is repositioned, potentially moving other objects out of frame.
    - **Container:** Moves the container instead of the camera, keeping other objects in view as the camera is still. In other words, moving the base map instead of the camera to see other parts of the map. This setting also means you do not have to use the front layer using two cameras to achieve the same effect as when moving the container. Borders and other elements on the map can be preloaded once for the base map, but this can only be done with a flat map (not a globe).
- **Update from Container Transformations:** Updates the values (lat, long, distance) in the Navigator plug-in when the navigator container is moved or scaled when enabled. Used in interactive mode.

Animation

The **Animation** button displays the camera animation parameters. There are two parameter tabs: Basic and Advanced.

Basic

- **Number of Hop Points:** Sets the number of key frames (hops) used in the animation.
- **Hop Duration Mode:** Defines the time gap between two key frames.
    - **Fixed:** Uses the same duration as set in the Hop Duration parameter for all hops.
    - **Auto:** Calculates the duration of the animation between hops automatically. minimum hop duration is based on the Hop Duration parameter and the Hop Min Time parameter.
    - **Manual:** Disables the Hop Duration parameter. Enables the user to set hop duration manually in the stage editor.
- **Hop Duration:** Sets the animation length, between one hop to another, in seconds.
- **Hop Min Duration (%):** Sets a minimum time for each hop when Hop Time Mode is set to Auto.
- **Stop Type:** Defines the animation behavior at each hop.
    - **Disabled:** Enables the animation to only use stage properties.
    - **Stop Points:** Enables the animation to stop at each hop point and wait for a continue command.
    - **Pause Points:** Adds a pause point to each hop. An additional parameter, Pause Time, is added to define the pause length in seconds.
    - **Flyover:** Simulates a flight pass over the hop points in a spline path, using the Flyover minimum Height parameter. When selected, additional parameters are enabled.
    - **Per Hop:** Sets the hop stop type in the NavFinder plug-in. Different stop types can be set for each hop.
- **Stop at First Hop:** Defines whether a stop point is added to the first hop (the beginning of the animation). The parameter is enabled only when using Stop Points or Pause Points as the Stop Type value.

Copyright © 2026 Vizrt

- **Pan and Tilt Animation:** Enables or disables the pan and tilt values of the camera of each hop in the animation. When set to `On`, this setting enables the Pan and Tilt values for the NavFinder plug-in.
- **Progress Profile:** Defines timing profile between stop points of animation. Smooth option causes the animation to ease in and out of hop points.

Basic - Flyover

- **Flyover Path Profile (Stop Type is set to Flyover):** Defines flying curve profile (curvature) of animation path.
- **Lag On Point:** Defines timing profile of the flyover animation (similar to *Flyover Path Profile* button).
- **Flyover Min Height:** Sets the minimum flyover height in centimeters.

Advanced

- **Preferred Flight Height:** Defines the camera height that is used in the animation between the hops.
- **Preferred Flight Style:** Defines the camera movement between the hops.
  - **Zoom:** Enables linear movement from hop point to the high point and back into the next hop point.
  - **Auto:** Calculates the movement according to distance, height, and so on.
  - **Flyover:** Enables a smooth movement from one hop to another.
- **Preferred Landing:** Defines the animation behavior when approaching the hop points.
  - **Helicopter:** Uses a steeper approach to the hop point.
  - **Auto:** Calculates the approach according to distance, height, and so on.
  - **Airplane:** Uses a flatter approach to the hop point.

⚠ **Note:** The drifting options allow camera pan and tilt drifting during the animation. This option is used to give the animation movement a kind of satiate feeling.

- **Pan/Tilt Drifting (deg):** Defines the amount of pan/tilt change for each cycle. The cycle is defined by the parameters **Drifting Pan Time (Frames)** and **Drifting Tilt Time (Frames)**.
- **Drifting Pan Time (frames):** Defines the time to complete a full turn of the pan.
- **Drifting Tilt Time (frames):** Defines the time to complete a full turn of the tilt.
- **Drift Phase Shift (%):** Defines the offset between the pan movement and the tilt movement.
- **Drifting Pause:** Defines whether the drifting should stop at the start point or during stop points. If Never is selected, drifting does not stop.
- **Flight Direction:** Forces the direction of the flight. Default is the shortest route to the next hop. If East or West is selected, flight route is set according to the selected option.

Miscellaneous

The **Miscellaneous** button displays the editor view for setting interactivity and label related parameters.

Interactive

- **Interactive Mode:** Defines the interactive behavior of Viz. Available modes are None, Editor, On-Air, Always Fly To and On 'i'.
  - **None:** Disables interactive mode.
  - **Editor:** Enables interactive mode during scene editing.

Copyright © 2026 Vizrt

    - **On Air:** Enables interactive mode when Viz Engine is in On Air mode.
    - **Always:** Enables interactive mode during scene editing and when Viz Engine is in On Air mode.
    - **Fly to:** Defines destination properties. Available *Fly To* options in Interactive mode are Country, Region, Sub Region and Home. **Country** makes the camera animate to the country in which the mouse was clicked. Animation stops when the camera reaches a distance from the country as defined by the *Extra Zoom Country* parameter. **Region** makes the camera animate to the region in which the mouse was clicked. Animation stops when the camera reaches a distance from the region as defined by the *Extra Zoom Region* parameter. **Sub Region** makes the camera animate to the sub region in which the mouse was clicked. Animation stops when the camera reaches a distance from the sub region as defined by the *Extra Zoom Sub Region* parameter. **Home** makes the camera animate to the position defined by the *Home Lan*, *Home Lat* and *Home Distance* parameters.
    - **On 'i':** Enables interactive mode during scene editing and when Viz Engine is in On Air mode, when pressing the **i** key while using the mouse to navigate.
- **Interactive Animation:** Enables user activated animation from the current map position to the current selected hop when set to `On`. This animation is triggered by the user in the NavFinder plug-in, by pressing the *Fly To Map* button. The *Fly To* option enables the user to select a point on the map, by clicking the mouse, and the animation runs from the current camera position to the selected point.
- **Extra Zoom Country:** Defines the extra zoom value added to the camera animation when animation destination is a country. The camera zooms in to the selected country until the bounding box of the country fills the render window. The extra zoom defines an additional zoom value to the final camera position calculations.
- **Extra Zoom Region:** Defines the extra zoom value added to the camera animation when animation destination is a region. The camera zooms in to the selected region until the bounding box of the region fills the render window. The extra zoom defines an additional zoom value to the final camera position calculations.
- **Extra Zoom Sub Region:** Defines the extra zoom value added to the camera animation when animation destination is a sub region. The camera zooms in to the selected region until the bounding box of the sub region fills the render window. The extra zoom defines an additional zoom value to the final camera position calculations.
- **Home Lon:** Defines a longitude value for a home point.
- **Home Lat:** Defines a latitude value for a home point.
- **Home Distance:** Defines a distance from a home point.

Labels

- **Labels:** Defines the label behavior. The Labels setting overrides all labels (in all levels of the hierarchy) under the navigator container.
    - **Overlay:** Places labels on a plane in front of the map (see Label It for description). When this mode is selected an additional parameter is enabled, Label Camera, which defines the camera to be used for the labels.
    - **On Map:** Places labels on the map.
    - **On Map Scaling:**

Advanced

The **Advanced** button displays additional animation parameters.

Copyright © 2026 Vizrt

- **Helicopter Lift Duration:** Sets the time, in seconds, for the helicopter to go out from a hop point to the high point in the middle.
- **Helicopter Smooth Level:** Sets the animation smoothness between one hop to another. This parameter affects the animation path when a helicopter flight is simulated.
- **Airplane Smooth Level:** Sets the animation smoothness between one hop to another. This parameter affects the animation path when a Airplane flight is simulated.
- **Tilt Smoothing:** Sets the required smoothing value for the tilt animation. This parameter is enabled when the Pan and Tilt parameter in the animation tab is enabled. When set to `On`, an additional parameter, Tilt Smoothing Rate, is enabled.
- **Distance Mode:** Defines how the camera distance from the map during the animation and at the hop points are calculated:
    - **Regular:** Sets the distance that is calculated by the Navigator plug-in based on the hop locations and distance from map as set by the user.
    - **Fixed:** Sets a fixed distance for the camera while animating between the hops and at the hop point. When set to fixed, User selected distance at the hop point is ignored.
    - **minimum:** Sets the minimum distance to which the camera descends at the hop points and during the animation. If the calculated distance is larger than the minimum value, the camera uses the calculated distance.
    - **Boundaries:** Enables you to define a minimum and maximum distance.
- **Min Distance:** Sets the minimum distance value.
- **Max Distance:** Sets the maximum distance value.
- **Face North:** Faces the camera towards the map's north.
- **Update Stage in On Air:** Defines whether the stage jumps to the hop position when updating a map in On Air mode.
- **World Center Mode:** Offsets the world center to use the values entered instead of the center of the base map.
    - **World Center Longitude/Latitude:** Sets the world center at the values provided.
    - **Update World Center:** Updates all plug-ins when pressed after values are changed.
    - **Use Current Values:** Takes the current values from current camera position and use them.

Common Buttons

- **Calculate Animation:** Re-builds the animation between the hops using the parameters defined in the plug-in.
- **Center Map:** Aligns the center of the map with the center of the screen.
- **Refresh All Maps:** Makes the Navigator plug-in search its sub-tree for containers with CWMClient and NavFinder and refreshes the CWMClient maps.

Known Issues

- Rotation, scaling and translation above the Navigator container might affect the plug-in behavior. Do not apply any rotations above the container in the hierarchy and use only the pan/tilt parameters of the plug-in to control the orientation of the camera.

Copyright © 2026 Vizrt

5.4.32 NavScale



The NavScale plug-in maintains the scale of an object, related to the screen, during the Navigator animation.

The plug-in is placed on a child container under the Navigator plug-in, and it maintains the defined scaling throughout the animation.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

NavScale Properties

- **Scale on Distance Factor:** Sets the value to the required number by modifying the scale factor and checking the result in the renderer.

Note: This factor does not use any measurement units, but it calculates the object's scaling using a number of parameters from the Navigator plug-in.

Copyright © 2026 Vizrt

5.4.33 NavSlave



The NavSlave plug-in creates a relation between its container and a Navigator plug-in container in the scene.

This plug-in locks the NavSlave container to the longitude and latitude values of the Navigator plug-in. The plug-in searches the hierarchy above it for the navigator container.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

NavSlave Properties

- **Altitude:** Defines the altitude of the object in relation to the Navigator container.
- **Tangent to Globe:** Sets the object to move over the globe surface when set to `On`.

5.4.34 Place Finder



The Place Finder plug-in generates a map or map animation without having to actually select the map, but uses rules to define what map is generated. There are two ways of selecting a map location. The first is simply by entering longitude and latitude values, the second is based on searching the map database for the location and using the first value (that was ranked the highest). Since a user cannot select the final location from a list, it is important to enter more information to make sure the correct location is found.

If you are looking for Paris, Texas in USA, entering `Paris` results in Paris, France because the capital of France is ranked the highest. Searching for `Paris USA` provides results for towns named Paris in the USA; however, searching for `Paris TX USA` or `Paris Texas` provides a more specific result.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps

Place Finder Properties

Location

- **Longitude:** Provides map center Longitude. Changing the value does not affect the map until you hit **Take**.
- **Latitude:** Provides map center Latitude. Changing the value does not affect the map until you hit **Take**.
- **Long/Lat Format:** Sets the format of the incoming data as either Long/Lat or Lat/Long.
- **Longitude/ Latitude:** Provides the map center Longitude and Latitude as one field (very useful for external control). Changing the value does not affect the map unless the last character is a semi-colon (**;**).
- **Search:** Determines the map location to search for. Click the **Take** button to perform the search.
- **Frame by Search Result:** Frames the location from the result of the search when set to `On`.
- **Zoom level:** Determines how the map should be framed. This option is relevant if the map center is based on longitude and latitude values (and not a search location), or if the **Frame by Search Result** is disabled.
  - *Default Size* frames the map based on Default Map Size.
  - *Country, Admin1, Admin2* is first trying to find the relevant information (e.g. country and so on) and fails for example if map center is in the sea. Next it uses the bounding box of the user selection (please read about the Texas example in the Place Finder introduction section).
  - If the user selects *Admin2* the map's frames are based on Lamer County which is Admin2 in that specific Longitude and Latitude.
  - If user selects *Admin1* the map's frames are based on Texas which is Admin1 in that specific Longitude and Latitude.
  - If user selects *Country1* the map's frames are based on USA which is the country for that specific Longitude and Latitude
- **Map Center:** Centers the map based on a point or a frame. If you search for `Beijing, China`, the map can be centered around the point identified as Beijing. This may, depending on your zoom level, leave parts of Beijing out of the frame. Alternatively, you can center your map based on the frame that captures the entire Beijing area in view.
- **Default Map Size (deg):** Defines a default map size for those areas that cannot be framed using predefined administration levels (e.g. international waters).

Copyright © 2026 Vizrt

- **Extra Zoom:** Adds an extra zoom to the map in percentage (`0.001-100%`).

Labels

- **Add Search Result Label:** Adds a label based on the search result.
- **Add Region Label:** Adds a region label based on the selected zoom level.

> ⚠ **Note: Short** only adds the location name, while **Long** adds more information (e.g. country and so on).

- **Add Event Label:** Adds event based labels (e.g. floods, fire, festival). To use different designs, a pipe symbol (**|**) can be used as separator (for example, *Fire*|*Fire in LA* uses a design named *Fire* and the text is *Fire in LA*. Changing the value does not affect the map unless the last character is a semi-colon (**;**).
- **Label Designs:** Determines the location for Label designs. Normally, the plug-in resides next to the CWMClient plug-in and it uses its designs, or the global designs.
- **Label Holder:** Determines the location for Label holder. Normally, the plug-in resides next to the CWMClient plug-in and it uses its holders, or the global holders.
- **Auto Labels Holder:** Defines the location of the Automatic Labels Holder. Automatic labels are labels generated based on the boundaries of the map.

Regions

- **3DRegions:** Determines whether 3D Regions are added. Relevant only when **Zoom level** is set to one of the region types (i.e. Region/Country/Admin1/Admin2/Sub_region).
- **3DRegion Designs:** Determines the location for 3D Region designs. Normally, the plug-in resides next to CWMClient plug-in and it uses its designs, or the global designs.
- **3DRegions Holder:** Determines the location for 3D Region holder. Normally, the plug-in resides next to CWMClient plug-in and it uses its holders, or the global holders.

Advanced

- **Search:** Defines which server data to search in (i.e. map data base, street data (map data base + street data), or Web) This works the same way as in the Classic client.
- **Immediate Response**
  - **Disabled:** Needs a user action to initiate the search: Either type the term with a semi-colon (**;**) at the end or press the **Take** button. Disabled is the default setting.
  - **Enabled:** Changes the search so that every letter typed initiates a search.
- **Status:** Provides the status of the search.

Copyright © 2026 Vizrt

5.4.35 Publish To Design



The Publish To Design plug-in enables updated parameters to be set directly to the design they came from (either container hierarchy or object pool) by pressing Publish To Design or create a new design out of modified data by assigning a new name under the New Design tab.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps_Adv

Publish To Design Properties

- **Publishing Target:** Publishes using the original design or creates a new design.
- **Design Name:** Sets the new name for the new design or selects a design from the Designs list.
- **Publish to Design:** Sends the label back to where it came from, but it does not refresh the scene.
- **Publish and Refresh:** Sends the label back to where it came from and refreshes the scene.

Copyright © 2026 Vizrt

5.4.36 Region2Tex



The Region To Texture plug-in simulates shadow effects for 3D Regions. This is achieved by creating a texture of a region's contour.

The plug-in works in two modes: By creating a single texture that applies to all regions in the scene or by creating multiple textures that applies to each individual region.



To work with a **single texture**, the plug-in must be placed on the Region Holder container.



To work with **multiple textures**, the plug-in must be placed on a container above the 3D Region container.

Copyright © 2026 Vizrt

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

Region To Texture Properties

- **Dynamic Shift:** Shifts shadow texture more when Navigator is farther away. This could be used to better see the shadow from farther distances.
- **Dynamic Shift Amount:** Sets the extent of the Dynamic Shift.
- **Attach to Light:** Considers light direction according to the positioning of the shadow texture when set to `On`. When set to `Off`, *Shift Direction* allows you to manually set the direction of the shadow.
- **Light ID:** Sets the ID of the light to be considered.
- **Texture Size:** Sets the size of the shadow texture. The larger the texture, the better is the visible quality; however, it also requires more rendering time.
- **Blur Amount:** Determines the extent to which the texture is blurred to resemble the appearance of a shadow.
- **Shift Direction:** Sets the shift direction of the shadow manually and according to the region. This setting can only be used if the *Attach To Light* setting is disabled (`Off`).
- **Shift Amount:** Sets the amount of shift.

Working with Region to Texture

To Create a Simple Region to Texture Scene

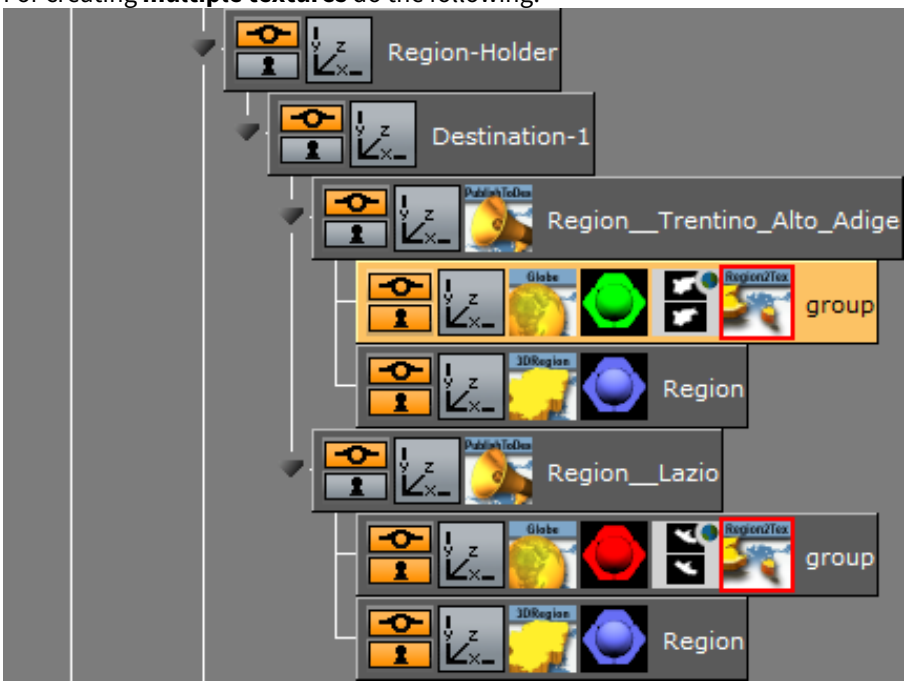
1. Add the Map Builder plug-in to the Scene Settings.
2. **Save** the scene.
3. Open the Map Builder editor and click **Launch Wizard.** This opens the Map Builder. Note that it may open behind Viz Artist.
4. From the **Choose Scene Type** dialog box select **Navigator.**
5. Click the **Base Map.** This opens the Viz World Maps Editor.
6. Select a **stylesheet** for your map and click **OK.**
7. Click the **map** for Destination 01. This opens the Viz World Maps Editor.
8. Click the **Browse Map** button (see Map Tool Bar) and select two regions (for example, Trentino-Alto Adige and Lombardia in Italy) and click **OK.**
9. Click **Build.**
10. Save the map template file to your desired location (for example, `C:\Temp\Maps`). Once saved, the map scene is generated.
11. For creating a **single texture** do the following:



Copyright © 2026 Vizrt

a. Navigate the scene tree to the **Region-Holder** container found under **Object > MapAndHops > GeoReferenceMap > Holders.**
b. **Add** the **Region to Texture** plug-in to the Region-Holder container.

12. For creating **multiple textures** do the following:



a. Navigate the scene tree and **split** the **Region__<name>** container found under **Object > MapAndHops > GeoReferenceMap > Holders > Region-Holder > Destination-n.**
b. Add a new **group** as a sub-container of the **Region__<name>** container.
c. **Add** the **Region to Texture** plug-in to the new group container.

13. Open the Publish To Design editor and click the **Publish and Refresh** button.
14. To adjust the textures, split the merged region containers and adjust the settings available in the Region to Texture editor.

Copyright © 2026 Vizrt

5.4.37 Trace It



The Trace It plug-in places the 3D object it is attached to over a line, created with the 3D Line or Shape to Spline plug-in, and follows the line's end point.

The object follows the line animation as the object with the Trace It plug-in move with the 3D Line/ Shape to Spline end point.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> Maps

Trace It Properties

General

- **Progress:** Indicates the animation position of the 3D Line/Shape to Spline object. When using Shape to Spline, Trace It uses the ShapeToSpline field *Trim End* to control the progress. *Trim Start* and *Trim Offset* must be `0`. If the data originates from a KML file, the progress is calculated in geographic units, otherwise viz units.
- **Smooth Direction (%):** Defines the object's motion behavior when changing direction. When set to a low value, the direction changes faster.
- **Quality:** Tells the plug-in which resolution to track when tracing a border that by default has three levels of detail (LOD). Available options are High, Medium and Low.
- **3DLine:** Defines the 3D Line/Shape to Spline container that the Trace It plug-in follows.
  - **Previous:** Follows the animation of the 3D Line/Shape to Spline plug-in in the previous container.
  - **Above:** Follows the animation of the 3D Line/Shape to Spline plug-in in the above container.
  - **Container:** Follows the animation of the 3D Line/Shape to Spline plug-in in the container dragged to its *container* place holder.
- **Progress Control:** Defines how the object's progress is controlled:
  - **None:** Uses the progress ID manually created by the user (stage animation).
  - **Navigator:** Follows the animation of the Navigator plug-in.
  - **Slave:** Follows the source of the trace from the 3Dline plug-in.
  - **Master:** Controls the 3D line plug-in progress, which is the source of the trace.

Camera

The **Camera** tab defines the mode for camera tracking.

- **Progress:** Indicates the animation position of the 3D Line/Shape to Spline object.
- **Trace Camera:** Controls a navigator's animation between a specified hop and the corresponding one, or manually driven (standalone). Available options are Off, Navigator, Standalone, General and Locks.

  > **Note:** When using Shape to Spline, no tracing is possible in non-geo mode, either with camera or without.

  - **Off:** Does not trace camera animation.
  - **Navigator:** Traces the camera animation between the selected hop and the following hop.

Copyright © 2026 Vizrt

    - **Standalone:** Traces the camera animation between the selected hop and the following hop.
- **General:** Defines the tracing parameters.
    - **Smooth Camera Position (%):** Smoothens camera position path.
    - **Smooth Camera Direction (%):** Smoothens camera direction path.
    - **Start Hop:** Selects a hop in the Navigator animation. The camera trace is inserted between the selected hop and the next one.
    - **Update Hop Data:** Updates data in NavFinder plug-ins where the camera trace is inserted (actually sets start and end values of the tracing route to insert camera trace smoothly to the Navigator animation).
- **Locks:** Enables the user to lock camera animation parameters:
    - **Lock Pan:** Makes the camera pan to follow the path direction when set to `On`. When set `Off`, pan animation uses the Navigator parameters.
    - **Pan Offset:** Sets the pan offset to path directions.
    - **Lock Tilt:** Makes the camera tilt follow the path direction when set to `On`. When set `Off`, tilt animation uses the Navigator parameters.
    - **Tilt Offset:** Sets the tilt offset to path directions.
    - **Lock Distance:** Uses the Distance parameter value to set the camera distance from the map during the animation when set to `On`. When set to `Off`, the distance during the animation uses the Navigator parameters.
    - **Distance:** Sets the distance to use during the animation.

Advanced

- **Segmentation:** Tells the Trace It plug-in what segment to track when a line is split into different segments (for example a region might have several islands) when tracking. Available options are Auto, Largest, Sectioned and All.

IMPORTANT! Maximum number of line segments are `10000`. Shape files that exceed this limit results in lines not being drawn.

- **Overlay:** Transfers the object from a position on the actual map to a position in a different camera (similar to the Label It plug-in's Overlay options).

Note: Shape to Spline has no button for controlling splitting dateline (whereas 3D Line uses *Enable World Periodicity*).

Copyright © 2026 Vizrt

5.4.38 World Position



The World Position plug-in places an object over a geographically referenced map by setting the Longitude, Latitude and Altitude parameters. When the object is moved over the map, the current values of Longitude, Latitude and Altitude are updated in the World Position plug-in. The object with the World Position plug-in must be placed under a map in the hierarchy.

Note: This plug-in is located in: Plugins -> Container plug-ins -> Maps

World Position Properties

Simple

- **Longitude:** Sets the parameter to the requested longitude. The object moves over the map to the requested location. Another option is to move the object and read its longitude value from this field.
- **Latitude:** Sets the parameter to the requested latitude. The object moves over the map to the requested location. Another option is to move the object and read its latitude value from this field.
- **Altitude Units:** Selects the units that the altitude parameter uses.
- **Altitude:** Sets the parameter to the requested altitude. The object moves over the map to the requested location. Another option is to move the object and read its altitude value from this field.
- **Move to Map Center:** Moves the object to the center of the parent map when clicked.
- **Update Lat/Long From Container Position:** Updates the latitude and longitude position parameters when clicked.

Advanced

In addition to the fields in the Simple tab, the Advanced tab has the following fields:

- **Tangent to Globe:** Keeps the object parallel to the globe surface when set to `On`. The parameter is enabled when the World Position plug-in is placed in a child container of a container with a Globe plug-in.
- **Update Long/Lat from Position:** Gets the current container location and updates the longitude, latitude and altitude parameters:
  - **Auto:** Updates the longitude and latitude values once when a new position is dragged over the container and never again.
  - **Never:** Updates the longitude and latitude values, but never updates the object's position.
  - **Always:** Updates the World Position when it changes and always checks the object's position.
- **Follow Terrain:** Reads the height (Altitude) from the terrain in case of a terrain geometry.
- **Longitude/Latitude:** Determines the position in degrees:minutes:seconds format.
- **Long/Lat Format (Long/Lat)(Lat/Long):** Sets the format of incoming data.
- **Lat/Long (+/-dd:mm:ss.xx):** Sets both latitude and longitude values in a single string.
- **Offset/Fine Tuning**
  - **Longitude/Latitude Offset:** Positions the object at a given offset from the actual longitude/latitude when **Offset** is selected.
  - **Longitude/Latitude Seconds:** Provides fine tuning values in seconds when **Fine Tuning** is selected.

Copyright © 2026 Vizrt

5.5 Maps Scene Plug-ins

This chapter describes all scene plug-ins. The container plug-ins are found in the following plug-in folders:

- **Maps:** Contains standard plug-ins.
- **Maps-Adv:** Contains advanced plug-ins.

See the following sections for more information:

- 3D Map Setting
- Label Manager
- Light On Globe
- Map Builder
- Traffic Manager

Copyright © 2026 Vizrt

5.5.1 3D Map Setting



The 3D Map Setting plug-in manages border data from the server.

The border data is retrieved from the Viz World Server (WoS), according to the setting in the 3D Map Setting plug-in, and is used for applying a graphic design to the borders in the map, drawn by the 2D Label plug-ins.

This section contains information on the 3D Map Setting plug-in properties:

- 3D Map Setting Properties
  - General
  - Border Data
  - Window
  - GUI
  - Advanced
- Buttons

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Maps

3D Map Setting Properties

General

- **Geographic Container:** Draws the border of the selected map. Drag the CWMClient container that defines the map area to the container place holder.
- **3D Global Designs Path:** Sets the path to a Viz folder containing the global designs.
- **3D Global Designs:** Holds the place for global designs container.
- **3D Global Holder:** Defines the container that holds all the global 3D objects created from the global designs.
- **On Screen Global Holder:** Defines the container that holds all of the on-screen global 3D objects created from the global designs.
- **Units:** Defines the type of units used to measure how many degrees, kilometers, miles or nautical miles are seen. The selection made here affects Navigator's Height Units setting and the label appearance Units in the Label Manager plug-in (see Definitions section).

Border Data

- **World:** Defines whether the border data is fetched for the entire world (country borders) or other data as defined in the *Additional Region Data* parameter.
  - **None:** Retrieves the border data as defined in the *Additional Region Data* parameter.
  - **Full:** Retrieves the country borders data for the entire world (country borders only).
- **Additional Region Data:** Defines additional border data that is retrieved from the server with the data defined in the *World* parameter. The additional data is limited to the selected region in the *Region List* parameter. This parameter cannot be changed without an instance of Viz World Server (WoS).
  - **None:** Does not use additional data.
  - **Country:** Uses country border data in the selected region.

Copyright © 2026 Vizrt

    - **Region:** Uses region border data in the selected region.
    - **Sub Region:** Uses sub-region border data in the selected region.
- **Region List:** Defines an area of the world for which the additional region data is retrieved from the server. The parameter limits the data size retrieved from the server.
- **Country Borders:** Creates country borders for the selected region in the region list property.
- **Rebuild:** Retrieves the information from the Viz World Server (WoS).
- **Rebuild (Force New):** Recreates the data on the Viz World Server (WoS) and saves it to the cache folder, even if cached data already exists.

Window

- **Window Active:** Enables user definition of an area in the renderer as an active window. All maps data created by the Viz World Client is redrawn to fit into the defined window. When set to `On` additional parameters are enabled:
- **Effect Culling:** Defines whether the window affects the culling of the vector data (streets, borders, etc.).
- **Effect Position:** Offsets the map either by (1) Using the window width and height and window center X and Y properties, or (2) By turning `On` container mode and dragging a container to the Window Container parameter, which is used to track the size and position for offsetting the map camera.
- **Window Mode:** Defines the source of the window aspect: Manual sets the window aspect to be user defined, Camera sets the window aspect to be the same as the render window, and Image sets the window aspect to be the same as the image aspect of the map.
- **Window Units:** Defines the units used to set the window size and position. When set to Percents, the window size is calculated as the defined percentage of the Viz render window size. The window position is calculated as the defined offset percentage of the render window.
- **Window Container:** Allows tracking a container from the scene tree that affects the size and position of the window.
- **Window Width:** Defines the width of the window in percents or pixels.
- **Window Height:** Defines the height of the window in percents or pixels. This parameter is enabled only if window mode is set to *Manual*.
- **Window Center Offset X:** Defines the X position (percents or pixels) of the window in relation to the render window (center to center).
- **Window Center Offset Y:** Defines the Y position (percents or pixels) of the window in relation to the render window (center to center).
- **Control Win Mask:** Defines whether a WindowMask plug-in (added to the map) is controlled by the 3D Map Settings plug-in to mask the defined window.
- **Window Mask Scale:** Defines the scale of the mask over the defined window.
- **Show Debug Window Mask:** Displays a red rectangle around the defined window when set to `On`.

GUI

GUI defines general parameters for controlling container names and container colors in the Viz GUI (effecting the Viz scene tree display), and the creation of Control Channels from added Viz World Client objects. The Control Channels in Viz are displayed under the Control tab and serve as an index for the scene tree. For additional information about Control Channels please refer to the Viz Artist User Guide.

- **Hops:** Defines GUI parameters for hop containers in the scene tree:

Copyright © 2026 Vizrt

    - **None:** Does not apply control channel or name conversion to the hop containers.
    - **Control:** Adds only a control channel for the hop containers.
    - **Name:** Names the created hop containers Hop-1, Hop-2, etc., according to the hop point selected in the NavFinder plug-in. No control channel is added.
    - **Full:** Adds a control channel for every hop container and the hop containers are renamed Hop1, Hop2, etc.
- **Designs:** Defines GUI parameters for any design containers (region designs, road designs, label designs, and so on) in the scene tree:
    - **None:** Does not apply control channel or name conversion to the design containers.
    - **Control:** Adds only a control channel for the design containers used in the scene. The containers are not renamed.
    - **Name:** Renames the design containers (dragged to the CWMClient plug-in) Label-Designs, Region-Designs, and so on. No control channel is added.
    - **Full:** Adds a control channel for every design container and the design containers are renamed.
- **Holders:** Defines GUI parameters for any object holder containers (regions, roads, labels , and so on) in the scene tree:
    - **None:** Does not apply control channel or name conversion to the holder containers.
    - **Control:** Adds only a control channel for the holder containers used in the scene. The containers are not renamed.
    - **Name:** Renames the holder containers (dragged to the CWMClient plug-in) Label-Holder, Region-Holder, and so on. No control channel is added. If the holder container is dragged to a hop CWMClient, it is suffixed indicating the hop number: Label-Holder-H1, Label-Holder-H2, and so on.
    - **Full:** Adds a control channel for every design container and the Holder containers are renamed.
- **Hops Color:** Sets the color index, as defined in the User Interface parameter in Viz Config that is used for the Hop containers in the scene tree.
- **Design Color:** Sets the color index, as defined in the User Interface parameter in Viz Config that is used for the design containers in the scene tree.
- **Holder Color:** Sets the color index, as defined in the User Interface parameter in Viz Config that is used for the generated objects holder container in the scene tree.

Advanced

- **Culling Threshold:** Sets the size of polygons to be culled (not rendered). It is generally better to cull small polygons, as they may not look good when rendered.
- **Polygon Quality:** Defines the quality of the drawn border lines. The higher the quality, the smoother the line is.
- **Polygon Quality Factor (%):** Enables the user to change the automatic polygon quality levels by setting a factor that changes the border quality. Values under `100%` decrease the quality of the lines. Values above `100%` increase the quality of the lines.
- **Language:** Allows the user to set the language of labels for the scene.

Tip: For more information about Viz Config and its configuration options, see the Viz Engine Administrator Guide.

Copyright © 2026 Vizrt

Buttons

- **Reset Design Holders:** Cleans up all design holders (labels, regions, borders and so on).
- **Update Project List:** Updates the list of Viz World map projects available to the designer. By default, Viz Artist always checks the Viz World Server for a list of projects and if a a scene is opened and its project does not exist an error message is displayed. However, if a project is added after Viz Artist is started you can click the Update Project List button in order to update Viz Artist and to avoid the error message.
- **Update All Map Elements:** Checks all map elements in the scene (CWMClients, Place Finder and so on) and refresh them.
- **Sync Local Cache Folders:** Synchronizes the local cache folder with the primary cache folder.
- **Start Logging:** Logs all map related activities into a file.
- **End Logging:** Saves the log file under `C:\Program Files (x86)\vizrt\Common\Maps\logs`.

Copyright © 2026 Vizrt

5.5.2 Label Manager



The Label Manager plug-in retrieves label information from Viz World Server and control the label's appearance when working in an automatic label mode. The Label Manager generates labels, based on the defined label designs, according to parameters defined in the Navigator plug-in, Label It objects and 2D Label objects in the label designs.

For the Label Manager to resolve conflicts, it needs different presets to work with. The more presets and the bigger difference between them, the better chance overlaps are resolved. When working with Label Manager, pay attention to and check all presets it uses to make sure that they are acceptable for the plug-in. Do not create a preset where the label is under the marker if you do not want to see it. Labels without markers (e.g. country) can have presets as well, which means their position can change slightly.

Label Manger basically has two ways of detecting conflicts:

1. Based on the marker (zero size) labels (bounding box) and the line between them. You set this for the 2DLabel (see Special section, Collision Mode = Tip Based) and when used with good and different presets it solves almost any problem, but if a you have a big marker there is always a chance of a marker overlap.
2. Based on the bounding box, the entire (square) bounding box is calculated and used. In this case, there is less chance of resolving conflicts, but it has no overlaps.

This section contains information on the Label Manager plug-in properties:

- Label Manager Properties
  - Map Data
  - Auto Labels
    - Examples
  - Definitions
  - Shadows

> **Note:** This plug-in is located in: Plugins -> Scene plug-ins -> Maps

Label Manager Properties

Map Data

The Map Data tab defines settings for the label types received from the Viz World Server (WoS). These settings are used when calculating label appearance in automatic mode. For every type of label selected in the list, set the required parameters.

- **Type:** Sets the type of labels data to display.
- **Start View:** Defines the map size from which the labels of the selected type appears during the Navigator animation. Parameter units are defined in the Definitions tab's *Units* parameter.
- **Life Span:** Defines the duration of the label appearance on screen. Note that this setting is most useful for country and region types.
  - **Full:** Displays the label at the defined Start View and remains on screen throughout the entire animation.
  - **Long:** Displays the label on screen for a long period of time.
  - **Regular:** Displays the label on screen for a medium period of time.
  - **Short:** Displays the label on screen for a short period of time.

Copyright © 2026 Vizrt

    - **Manual:** Displays the label on screen until the defined *End View* value is reached.
- **End View:** Defines the map size at which the labels of the selected type disappears. This parameter is only enabled if the Life Span is set to Manual. Parameter units are defined in the Definitions tab's Units parameter.

> ⚠ **Note:** The Label Manager plug-in controls and manages the automatic labels appearance, using various parameters from different plug-ins. The Life span and Auto Label parameters vary between labels during the animation (hop), to optimize labels display.

Auto Labels

The Auto Labels tab defines whether automatic labeling is enabled and the label designs used for automatic labeling.

- **Auto Labels:** Sets the auto labels mode to disabled, enabled or interactive.
    - **Disabled:** Does not use automatic labeling in the scene.
    - **Enabled:** Uses automatic labeling in the scene based on the defined designs and labels holder container.
    - **Interactive:** Enables the Auto Labels Quantity.
- **Auto Labels Quantity:** Allows you to set the amount of labels that should be shown. Few show approximately 75 labels (or less), Average 100 labels (or less) and Many 150 labels (or less). Manual sets the distance where the labels should appear (see Start View under Map Data) providing a consistent behavior (e.g. Town 1K always appears/disappears at the same distance).
- **Manual Labels:** Defines whether manual labels, defined for the map, are displayed with the auto labels. When Enabled is set, the Select Manual Labels button appear at the bottom of the editor. When clicked, WME opens, enabling the user to set manual labels.
- **Road Labels:** Defines the level (number) of road labels that are displayed. Select the required option, ranging from `0-None` to `5-All`.
- **Street Labels Style:** Defines the label style of the scene. Options are Viz Design and Internal (Open GL) labels.
    - **Viz Design:** Allows user definition of Viz labels. Note that this limits the number of labels that can be used in Viz. Depending on your system, 300-500 labels may cause Viz Engine to not render in real time. Using Viz Design also gives the option to set other parameters such as the visibility of the label and local and global placeholders for the label design (see 3D Designs & Holder).
    - **Internal (Open GL):** Allows using up to 100.000 labels; however, this limits design options to the font used and relative position and scale. When used with the Street Labels plug-in, this setting only works for Navigator scenes.
- **Labels Visibility (%):** Used when calculating size of street labels for the final map.
- **Gap for Same Street Name (%):** Used when calculating the minimum distance between two street labels of the same name.
- **3D Designs and Holder:**
    - **Designs (text field):** Defines a path in Viz's objects database that contains label designs. Automatic label designs are based on 2D Label and Label It plug-ins.
    - **Designs (place holder):** Displays the selected (drag and drop) top container holding the label designs.

Copyright © 2026 Vizrt

    - **Holder (place holder):** Displays the selected (drag and drop) container that resides under the map. This container is used by the Label Manager plug-in to store the scene's generated labels.
- **Limit Auto Labels:** Defines whether the auto labels are limited to a defined region/s or not. When enabled, the Limit To Region text field is enabled.
- **Limit to Region:** Defines the region to which the auto labels are generated.

Auto Labels can be limited to specified countries and/or regions. The format of valid input is:

- A semi-colon (**;**) between locations.
- A backslash (**/**) or slash (****) to specify the path to the regions.

> ⚠ **Note:** Abbreviations can be used (after defining such list in the WoS)

Examples

- USA;Mexico
- USA\TX;Mexico;Canada
- United States of America\Florida;United States of America\Georgia;USA\NY

Definitions

The Definitions tab is used to set general parameters for Label Manager behavior.

- **Intersection Mode:**
    - **Full:** Selects the best preset for each label and when labels intersect, one of them is turned off.
    - **Offsets:** Selects the best preset for each label.
    - **None:** Creates auto labels only. No intersections are calculated.
- **Screen Culling Mode:** Defines whether a label is drawn when it is outside the Viz renderer's scope/view.
    - **Center:** Fades out the label object when the center of the object is outside the Viz renderer's scope/view.
    - **Anything:** Fades out the label object when any part of the object is outside the Viz renderer's scope/view.
    - **Complete:** Fades out the label object when the entire object is outside the Viz renderer's scope/view.
    - **None:** Does not use culling. The labels are drawn according to the map data parameters only.
- **Map Culling Mode:** Defines whether a label is drawn when it is outside the map area but inside Viz renderer's scope/view.
- **User Labels Ignore Distance Rules:** Ignores the distance rules that apply to labels associated with a hop when enabled.
- **Intersection Distance:** Determines the distance to be used when calculating the best label setup to be used to avoid collision:
    - **Max distance:** Calculates labels at the max distance defined for the label type.
    - **Distance:** Uses the final hop position.
    - **2/4/8* Distance:** Uses the hop position area multiplied by the selected value.
- **Scene Type:** Determines which algorithm to use for a scene. Label Manager uses different algorithms for an interactive scene or a regular one. User should declare the scene type to get best results.
- **Units:** Defines the units used to calculate map size for the Start View and End View parameters. Available options are Degrees, Kilometers, Miles and Nautical Miles, where the default measurement unit is Kilometer. Note that this setting is also affected by the Units option in the 3D Map Setting plug-in (see Label Manager).

Copyright © 2026 Vizrt

- **Fade Time (fields):** Defines the label's fade duration in fields.
- **Labels Draw Style:** Determines where labels are drawn.
    - **On map:** Draws on the map layer.
    - **Overlay:** Draws on a different layer (dynamic image/layer).
    - **On map scaling:** Draws on the map layer, but maintains the same size (avoid NavScale plug-in).
- **Label Camera:** Defines the camera that is used to draw the labels (normally camera two).
- **Debug Info Mode:** Shows different levels of debug information regarding the auto layout.

Shadows

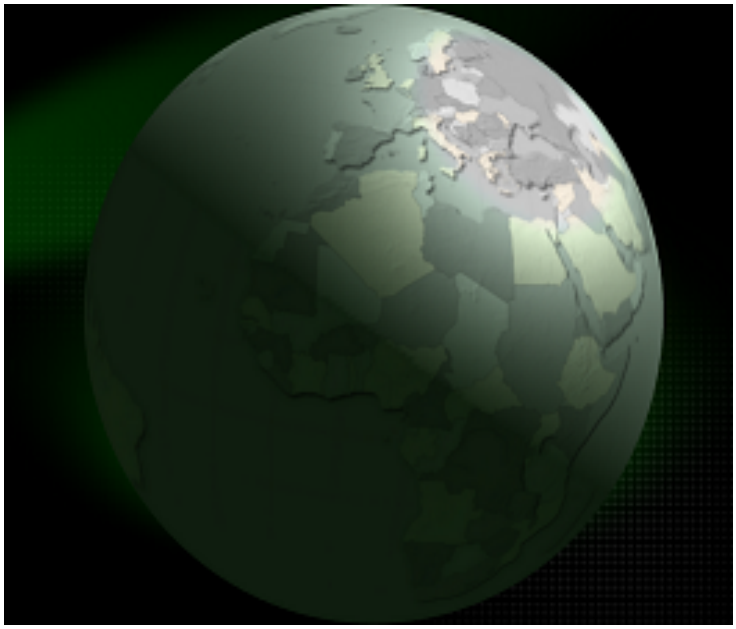
- **Cast Shadows:** Turns option to cast label shadows on/off.
- **Light Type:** Simulates different light types to cast the shadows.
- **Horizontal Angle:** Defines the horizontal angle position of the light.
- **Vertical Angle:** Defines the vertical angle position of the light.
- **Distance:** Defines light distance.
- **Shadow Plane:** Casts planar shadows on an imaginary plane and this option defines the spatial rotation of such a plane. Select one of the options:
    - **Straight:** Casts the plane parallel to the screen.
    - **Up Tilt:** Tilts the casted plane 45 degrees up from the screen.
    - **Down Tilt:** Tilts the casted plane 45 degrees down from the screen.
    - **Manual:** Allows the pan and tilt values of the casted plane to be set manually.
    - **World:** Casts the plane tangent to the world.
- **Tangent Point:** Sets the pivot point for connection shadow to the object.
- **Shadow Pan:** Determines a manual Shadow Pan.
- **Shadow Tilt:** Determines a manual Shadow Tilt.
- **Shadow Distance:** Determines the distance of the shadow plane from the object.
- **Shadows Camera:** Selects a camera for rendering the shadows. Since shadows are done by first rendering objects in a distant place and then overlaying their black silhouettes on the screen, this option defines the camera that looks at such distant place where objects are rendered. (Actually this camera is used for overlay).
- **Show Light Direction:** Visualizes light direction on screen. The light direction is visualized by showing a lit ball or an arrow.

Copyright © 2026 Vizrt

5.5.3 Light On Globe



The Light On Globe plug-in applies light sources to a globe object. The plug-in is required when designing a *hops* scene and the animation is going from the lighted area of the globe to the dark area of the globe. When using the Light On Globe plug-in, the lights follow the camera animation. The light sources are Viz lights, and the lighting parameters should be adjusted in the Viz light editor. The Light On Globe plug-in locks the light sources to the selected camera in the plug-in parameters.



Note: This plug-in is located in: Plugins -> Scene plug-ins -> Maps

Light On Globe Properties

- **Camera:** Sets the camera number for setting the light sources. The light sources are locked to the selected camera number.
- **Key Light Type:** Sets the main light source type. Available types are Local, Spot, Infinite or None.
- **Key Light Angle:** Sets the angle of the key light source, which is the longitude value for the center of the light projected on the globe.
- **Key Light Elevation:** Sets the elevation of the key light source, which is the latitude value for the center of the light projected on the globe.
- **Distance Scale:** Sets a scale value on the globe distance so the light can be closer or further away.
- **Attenuation:** Sets the level of light attenuation.
- **Back Light Type:** Sets the back light source type. Available types are Local, Spot, Infinite or None.
- **Back Light Angle:** Sets the angle of the back light source, which is the longitude value for the center of the light projected on the globe.
- **Back Light Elevation:** Sets the elevation of the back light source, which is the latitude value for the center of the light projected on the globe.

Copyright © 2026 Vizrt

- **Distance Scale:** Sets a scale value on the globe distance so the light can be closer or further away.
- **Fill Light Type:** Sets the fill light source type. Available types are Local, Spot, Infinite or None.
- **Fill Light Angle:** Sets the angle of the fill light source, which is the longitude value for the center of the light projected on the globe.
- **Fill Light Elevation:** Sets the elevation of the fill light source, which is the latitude value for the center of the light projected on the globe.
- **Distance Scale:** Sets a scale value on the globe distance so the light can be closer or further away.
- **Camera Light Type:** Sets the camera light source type. Available types are Local, Spot, Infinite or Off. The camera light follows the camera movements.
- **Camera Light Horizontal Shift:** Sets the horizontal shift of the light source in relation to the camera location.
- **Camera Light Vertical Shift:** Sets the vertical shift of the light source in relation to the camera location.
- **Toggle Mode:** Sets the lights behavior mode during camera movement:
    - **Static:** All light sources, except for the camera light, remain in a fixed location in relation to the globe.
    - **Dynamic:** All light sources maintain a fixed location in relation to the defined camera (that is it moves with the camera).

Copyright © 2026 Vizrt

5.5.4 Map Builder



The Map Builder plug-in builds a scene from a map template (*.mtpl) file, and saves the map scene changes to the map template file.

Note: This plug-in is located in: Plugins -> Scene plug-ins -> Maps

Map Builder Properties

- **Map Template Filename:** Sets the name of the scene template file or click the browse (**...**) button to select a file.
- **Rebuild Scene:** Updates the scene tree with changes from the Map Builder.
- **Launch Wizard:** Runs the Map Builder application.
- **Clean Scene:** Cleans the scene hierarchy.
- **Build Scene from Scratch:** Cleans the scene tree and rebuilds the scene hierarchy using the defined map template file.

Copyright © 2026 Vizrt

5.5.5 Traffic Manager



The Traffic Manager plug-in connects to DataHub for traffic flow and routing information.

> **Note:** This plug-in is located in: Plugins -> Scene plug-ins -> Maps_Adv

Traffic Manager Properties

- **DataHub Host:** Sets the hostname of the DataHub to use.
- **DataHub Port:** Sets the port number to use.
- **Source Name:** Defines the name of the source.
- **Update from DataHub:** Updates information from the DataHub.
- **Flow Data:** Defines the location of the traffic flow data.
- **Flow and Routing Shape File:** Defines the location of the traffic flow and routing shape file.
- **Build Flows Data and Update Flows:** Builds flows data and updates traffic flows (calls 3DLineManager).
- **Update Flows:** Updates the traffic flows.

Copyright © 2026 Vizrt

5.6 Maps Shader Plug-ins

This chapter describes shader plug-ins. The shader plug-ins are found in the following plug-in folders:

- **Default:** Contains the Rebound plug-in.
- **Maps:** Contains standard plug-ins.
- **Maps-Adv:** Contains advanced plug-ins.
- **Maps-Lab:** Contains experimental plug-ins. Since these plug-ins are experimental and not supported, they are not documented here.
- **Maps-Obs:** Contains obsolete plug-ins, installed only for backward compatibility. These plug-ins should **not** be used when designing new scenes. Since these plug-ins are obsolete and not supported, they are not documented here.
- **Texture:** Contains the FadeTexture plug-in.

See the following sections for more information:

- 3D Line Shader
- C3D Terrain Shader
- FadeTexture
- Geo Chart Shader
- Rebound

Copyright © 2026 Vizrt

5.6.1 3D Line Shader



The 3D Line Shader plug-in is used by the 3D Map Setting plug-in to draw the borders using the parameters set in the various 2D Label plug-ins.

It is added automatically when adding a 2D Label object to the scene tree.

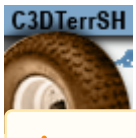
Note: This plug-in is located in: Plugins -> Shader plug-ins -> Maps

3D Line Shader Properties

This plug-in has no editable parameters.

Copyright © 2026 Vizrt

5.6.2 C3D Terrain Shader



The C3D Terrain Shader plug-in is added automatically when using the Atlas plug-in.

Note: This plug-in is located in: Plugins -> Shader plug-ins -> Maps

C3D Terrain Shader Properties

This plug-in has no editable parameters.

5.6.3 FadeTexture



The FadeTexture plug-in applies soft edges to a texture and to crops the texture.

Note: This plug-in is located in: Plugins -> Shader plug-ins -> Texture

FadeTexture Properties

- **Curve:** The plug-in has three plug-in editor views that enable different curve control options.
  - **Constant:** Controls texture edges separately, but no softness is applied to the texture edges.
    - **Crop Top:** Sets the crop value for the top of the texture.
    - **Crop Bottom:** Sets the crop value for the bottom of the texture.
    - **Crop Left:** Sets the crop value from the left of the texture.
    - **Crop Right:** Sets the crop value for the right of the texture.
  - **Spline:** Controls texture edges separately and a common softness value is applied to all edges. Spline has the same parameters as *Constant*.
    - **Fade:** Sets the softness value for the edges of the texture.

Note: If an edge is not cropped, the softness affects the edge.

- **Uniform:** Controls all texture edges together with a fixed softness value applied.
  - **Uniform Crop:** Sets the crop value for all edges of the texture (fixed softness is added to all edges).

Copyright © 2026 Vizrt

5.6.4 Geo Chart Shader



The GeoChart Shader plug-in is the shader for the GeoChart plug-in.

Note: This plug-in is located in: Plugins -> Shader plug-ins -> Maps_Adv

GeoChart Shader Properties

This plug-in has no editable parameters.

5.6.5 Rebound



The Rebound Shader plug-in stops shader inheritance. By default, all shader behavior is shared by all siblings of the container where the shader resides.

The Rebound shader stops this inheritance. It should be placed below the shader you would like to avoid inheriting from.

Note: This plug-in is located in: Plugins -> Shader plug-ins -> Default

Rebound Properties

This plug-in has no editable parameters.

Copyright © 2026 Vizrt

6 PixelFX Plug-ins

The PixelFX Plug-ins provide details about settings available through its configuration user interface within Viz Artist.

6.1 Related Documents

- *Viz Artist User Guide*: Contains information on how to install Viz Engine and create graphics scenes in Viz Artist.

6.2 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

6.3 Introduction to PixelFX Plug-ins

The PixelFX plug-ins affect pixels and geometry, and produce various special effects. They do three main things:

- Color correction
- Lens flares
- Other pixel-based effects, such as noise, gradients, various distortion effects, transitions, and similar effects.

The icon for each of the plug-ins have mini icons that show certain properties of the particular plug-in.



- At the bottom left, there is a performance bar, with three blocks (like a cellphone battery indicator). One block means the plug-in does not have a drastic performance effect, and three means the plug-in is taxing and should therefore be used with caution.
- At the bottom right the orange stack shows if the plug-in is stackable, meaning it can be used in conjunction with color correction plug-ins.
- Some plug-in icons themselves are divided in half with one side showing the visual effect with the plug-in and the other without it.

Some of the PixelFX plug-ins work only when applied to a Container that also has a Texture. The rest function on the Renderer's pixel buffer. For example, you may apply color correction to the root Container in a Scene, therefore perform a color correction to the whole scene. Another example is pxNoise- apply it to the root level container of a scene or a whole tree branch to add a kind of film-grain effect.

One of the parameters that feature in a number of the PixelFX plug-ins is Seed. This defines the pseudo-random rule for changed parameters. Once a value is chosen, although the original state is random, the effect actually looks the same on other computers as well.

Copyright © 2026 Vizrt

6.4 PixelFX Geometry

The Lens Flare plug-ins, which are part of the PixelFX plug-ins set, simulate the effect of streaks and spots of light, caused in real cameras when light enters and bounces inside a camera lens. This plug-in set contains geometries that simulate different shapes, which can be used for lens flare or simple flare effects. Each plug-in can take on very different appearance depending on how the parameters are adjusted. The best advice for working with these plug-ins is to simply play around with the parameter configurations.

The Flare draws itself as a combination of a geometry shader and a pixel shader. When a Flare Container is created a non-editable pixel shader is added to the created Flare container.

In addition, the Expert Container plug-in is automatically added and configured in such a way as to cause the shape used to always be seen in front of all other objects, blending with the rest of the scene to resemble the effect of light bouncing off the lens.



Copyright © 2026 Vizrt

Note: Lens flare plug-ins reside in their own containers. Applying these plug-ins to a container with geometry overwrites the geometry and disable any shaders in child containers.

The baseline plug-in within this set is the pxLensMulti plug-in, which is a container plug-in, whose purpose is to manage other geometry plug-ins. The one exception in this set is the pxLensEnergyBolt, which does not simulate lens flare, but rather the effect of electricity or lightning and it is not managed by pxLensMulti.

The default path for the Geometry plug-ins is: *<viz install folder>\plugin\<plug-in name.vip>*

The following Geometry plug-ins are located in the PixelFX folder:

- pxLensEnergyBolt
- pxLensGlowBall
- pxLensGlowSpikes
- pxLensPolyElement
- pxLensRandomPoly
- pxLensRays
- pxLensSpark
- pxLensStripes

See Also

- PixelFXLensFlare

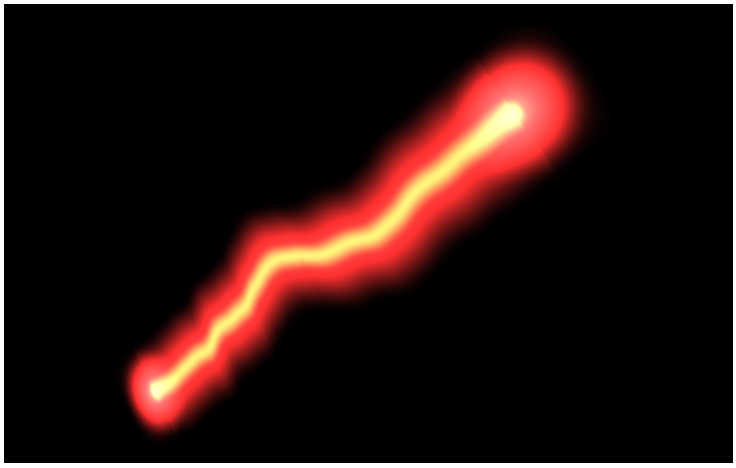
Copyright © 2026 Vizrt

6.4.1 pxLensEnergyBolt



The pxLensEnergyBolt plug-in produces the effect of electricity-like lightning between two points in space.

These points may be defined manually by entering positions, or by tracking the positions of other containers.



Note: This plug-in is located in: Plugins -> Geom plug-ins -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

pxLensEnergyBolt Properties

- **Use LOD:** Enables level of detail.
- **Glow Color:** Determines the color of the glow.
- **Radius:** Determines the glow size.
- **Core Color:** Determines the color of the core of the bolt.
- **Core:** Determines the size of the core of the bolt.

Note: If *Core* is a larger value than *Radius*, it creates a spillover of the color.

- **From X:** Determines the starting *x* position.
- **From Y:** Determines the starting *y* position.
- **From Container:** Uses a container as a starting position.
- **To X:** Determines the ending *x* position.

Copyright © 2026 Vizrt

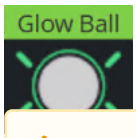
- **To Y:** Determines the ending *y* position.
- **End Caps:** Applies a cap at the end of the energy bolt when set to `On`.
- **Distortion:** Determines the average distance of the line distortion, where `0` is straight and larger or smaller values create a zigzag effect.
- **Speed:** Determines the change rate of the effect.
- **N:** Determines the number of distortions.
- **Seed:** Applies a randomization value to the effect.

See Also

- PixelFXLensFlare

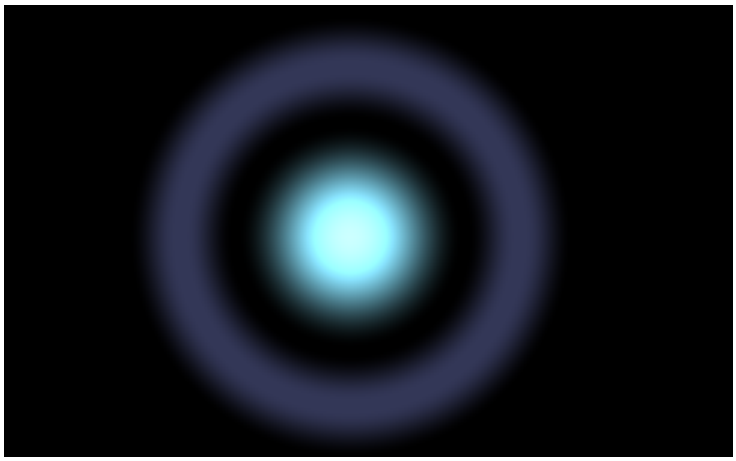
Copyright © 2026 Vizrt

6.4.2 pxLensGlowBall



The pxLensGlowBall plug-in produces the effect of a glowing ball.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> PixelFX



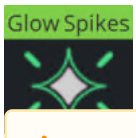
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

pxLensGlowBall Properties

- **Use LOD:** Enables level of detail.
- **Glow Color:** Determines the color of the glow.
- **Radius:** Determines the glow size.
- **Ring Color:** Determines the color of the ring.
- **Ring Radius:** Determines the radius of the ring.
- **Ring Width:** Determines the size of the ring.
- **Ring Softness:** Determines the blurriness of the ring.
- **Size:** Determines the size of the entire glowball.

Copyright © 2026 Vizrt

6.4.3 pxLensGlowSpikes



The pxLensGlowSpikes plug-in produces the effect of a glowing star of spikes.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> PixelFX



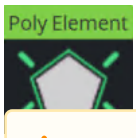
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

pxLensGlowSpikes Properties

- **Color:** Determines the color of the spikes.
- **Sides:** Determines the number of spikes.
- **Size:** Determines the size of the effect.
- **Angle:** Rotates the resulting spikes.

Copyright © 2026 Vizrt

6.4.4 pxLensPolyElement



The pxLensPolyElement plug-in creates a polygon shape.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> PixelFX



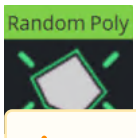
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

pxLensPolyElement Properties

- **Use LOD:** Enables level of detail.
- **Color:** Determines the color of the polygon.
- **Sides:** Determines the number polygon sides.
- **Size:** Determines the size of the effect.
- **Softness:** Determines the blurriness of the polygon.
- **Angle:** Rotates the resulting spikes.

Copyright © 2026 Vizrt

6.4.5 pxLensRandomPoly



The pxLensRandomPoly plug-in creates a polygon shape with random angles.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> PixelFX



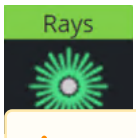
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

pxLensRandomPoly Properties

- **Use LOD:** Enables level of detail.
- **Ray Color:** Determines the color of the polygon.
- **Sides:** Determines the number polygon sides.
- **Angle Randomness:** Randomizes the angles.
- **Radius Randomness:** Randomizes the distance between the center and the angles.
- **Warp:** Pulls the non-angle edges towards the center of the polygon.
- **Warp Softness:** Determines the blurriness at the edges of the polygon.
- **Size:** Determines the size of the effect.
- **Angle:** Rotates the resulting polygon.
- **Seed:** Applies a randomization value to the effect.

Copyright © 2026 Vizrt

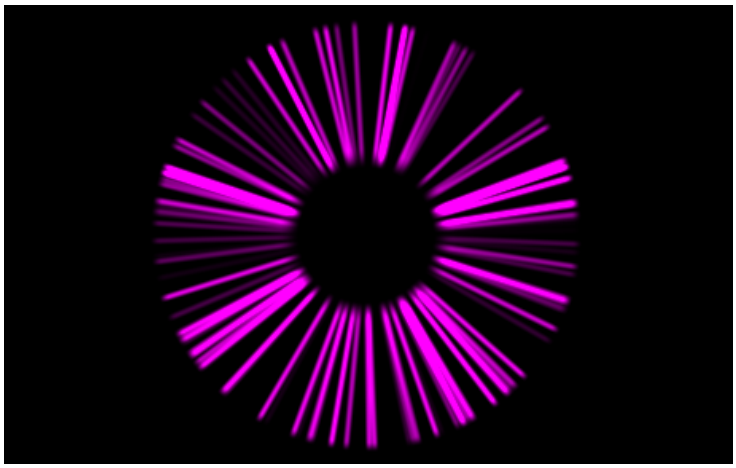
6.4.6 pxLensRays



The pxLensRays plug-in can be used to produce the effect of rays.

The rays can be static or move randomly based on the Speed parameter.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> PixelFX



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

pxLensRays Properties

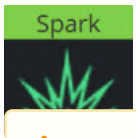
- **Ray Color:** Determines the base color of the ray.
- **Presets:** Selects user-defined (Custom) or predefined (Rainbow or Red > Green) gradient ramp.
- **Color Ramp:** Modifies the color of user-defined portions of each ray.
- **Image:** Uses a gradient ramp from the image dropped into the Image box.
- **Rays:** Determines the number or density of rays.
- **Size:** Determines the length of rays.
- **Inner Radius:** Sets the size of the inner radius.
- **Angle:** Sets the angle value between the first and last ray.
- **Width:** Determines the thickness of rays.
- **Speed:** Determines the change rate of the effect.

See Also

- PixelFXLensFlare

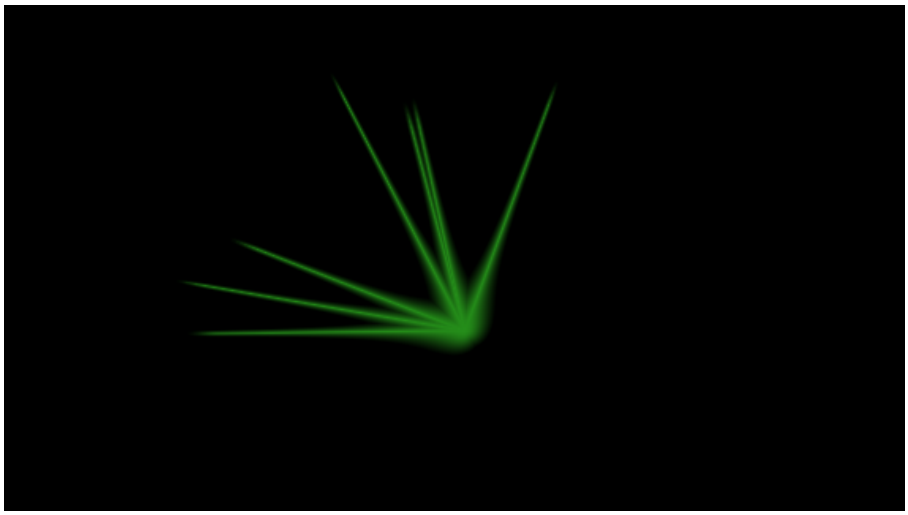
Copyright © 2026 Vizrt

6.4.7 pxLensSpark



The pxLensSpark plug-in creates a spark effect.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> PixelFX



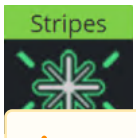
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

pxLensSpark Properties

- **Use LOD:** Enables level of detail.
- **Ray Color:** Determines the color of the effect.
- **Rays:** Determines the number of sparks.
- **Size:** Determines the size of the effect.
- **Seed:** Applies a randomization value to the effect.

Copyright © 2026 Vizrt

6.4.8 pxLensStripes



The pxLensStripes plug-in produces a symmetric stripe effect.

Note: This plug-in is located in: Plugins -> Geom plug-ins -> PixelFX



Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

pxLensStripes Properties

- **Use LOD:** Enables level of detail.
- **Symmetry:** Determines the number of axes of symmetry.
- **Length:** Determines the length of the stripe.
- **Width:** Determines the width of the stripe.
- **Size:** Determines the size of the stripes.
- **Angle:** Rotates the resulting effect.
- **Pinch:** Makes the ends of the stripes narrower than the set *Width*.

Copyright © 2026 Vizrt

6.5 PixelFX Container

The default path for Container plug-ins is: *<viz install folder>\plugin*.

- PixelFX
  - pxLensMulti
- pxColorWorks
  - pxAddSubtract
  - pxBlackAndWhite
  - pxBrightContrast
  - pxColorMatch
  - pxGamma
  - pxHueRotate
  - pxMask
  - pxSaturation
  - pxStack
  - pxTint
  - pxInvertColor

Copyright © 2026 Vizrt

6.5.1 PixelFX

The PixelFX plug-ins are part of the PixelFX Plug-ins set.

The following Container plug-ins are located in the PixelFX folder:

- pxLensMulti

pxLensMulti



The pxLensMulti plug-in serves as a parent container for all the other PixelFX Geometry Plug-ins. It simplifies the process of animating lens flares so that they resemble realistic behavior.

In the child containers, any number of lens flare shapes can be placed. pxLensMulti positions, colors, governs the opacity of all according to the configurable parameters.

The lens flare shapes at origin are a flat geometry. pxLensMulti makes sure that all the shapes it handles are constantly facing the camera.

> **Note:** pxLensMulti creates a notional line along which all shapes are scattered.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> PixelFX

pxLensMulti Properties

Under the Spread tab, the following parameters can be configured to handle shapes' position:

- **Mode:** Defines options for positioning of the multiple lens flares:
  - **Polar:** Places lens flares on a screen space and controls their positions using **Pos Radius** and **Pos Angle** parameters (the notional line is defined by the angle and radius originating from the screen's center).
  - **Cartesian:** Places lens flares on a screen space and controls their positions using **Pos X** and **Pos Y** parameters (the notional line is created between the screen's center and given an x-y location in the screen's coordinates).
  - **Container:** Uses a container to imitate the light position so that the notional line starts from that position and goes through the screen's center. If the container is visible, lens flares are positioned on an axis that goes from light source through the middle of the screen plane (Note that in these mode lens flares positions changed with camera or light movement). In Container mode, you need to drag a container from the scene tree into the Container box.

> **Note:** In Container mode, one point of the notional line is in the coordinates and the second point, through which the notional line passes, is defined in the screen coordinates. As such, camera movement in Container mode actually defines the shapes' movement.

- **Distance:** Defines the notional line that passes through the screen's center. The distance of this center point from the camera is defined by the this parameter's value.
- **Count:** Multiplies the quantity of each shape in each child container by the value entered for this parameter.
- **Radius Scale:** Scales the notional line from the origin onward.
- **Radius Rand:** Scales the notional line's length, regardless of origin.
- **Single Seed:** Allows only one seed to be used for each shape in the child containers when set to `On`. When set to `Off`, a different seed is used for each shape in the child containers.
- **Seed:** Defines the pseudo-random rule for randomly changed parameters. Once a value is chosen, although the original statement is random, the effect looks the same on other computers as well.

Under the Attributes tab, the following can be configured to handle non-positional parameters:

Copyright © 2026 Vizrt

- **Amount:** Determines the amount of light used in lens flares.
- **Amount Rand:** Randomizes the amount parameter between different flares.
- **Size Rand:** Randomizes the light.
- **Color Rand:** Randomizes the color.

See Also

- pxColorWorks
- PixelFX Plug-ins in Geom plug-ins
- PixelFXLensFlare in Shader plug-ins

Copyright © 2026 Vizrt

6.5.2 pxColorWorks

The color correction plug-ins, which are part of the PixelFX plug-ins set, are generally fast and efficient. There is a certain performance penalty when animating them in a stacked scenario.

> **Note:** The PixelFX plug-ins are a separate package from the standard Viz Artist installation, but function within the Viz Artist environment.

The following Container plug-ins are located in the pxColorWorks folder:

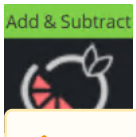
- pxAddSubtract
- pxBlackAndWhite
- pxBrightContrast
- pxColorMatch
- pxGamma
- pxHueRotate
- pxMask
- pxSaturation
- pxStack
- pxTint
- pxInvertColor

See Also

- PixelFX in Basic Shader Plug-ins
- PixelFX Geometry Plug-ins in Basic Geometry Plug-ins

Copyright © 2026 Vizrt

pxAddSubtract



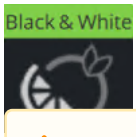
The pxAddSubtract plug-in adds or subtracts a constant color value to each pixel.

It can be applied to Everything, Shadows, Mid tones and Highlights.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

Copyright © 2026 Vizrt

pxBlackAndWhite

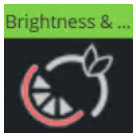


The pxBlackAndWhite plug-in applies gray scale to an RGB image or a tree branch holding geometry. It can be applied to Everything, Shadows, Mid tones and Highlights.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

Copyright © 2026 Vizrt

pxBrightContrast



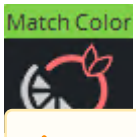
The pXBrightContrast plug-in adjusts the brightness and the contrast on each of R, G and B on a given image or a tree branch holding geometry.

It can be applied to Everything, Shadows, Mid tones and Highlights.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

Copyright © 2026 Vizrt

pxColorMatch



The pxColorMatch plug-in replaces one color with another.

It works on images and tree branches holding geometry alike.

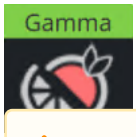
Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

pxColorMatch Properties

- **Weighting:** Defines the color range proximity to the selected color.
- **Inertia:** Defines the smoothness of the range proximity graph.
- **Count:** Defines the number of color replace pairs you wish to use.

Copyright © 2026 Vizrt

pxGamma

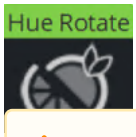


The pXGamma plug-in adjusts the gamma correction on each of R, G and B on a given image.

It can be applied to Everything, Shadows, Mid tones and Highlights.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

pxHueRotate



The pxHueRotate plug-in adjusts the hue rotation on an image.

It can be applied to Everything, Shadows, Mid tones and Highlights.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

Copyright © 2026 Vizrt

pxMask



The pxMask plug-in applies a mask for a color correction operation.

The mask is defined by a black and white image that is dragged into the pxMask parameters. pxMask can function in conjunction with a mask.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

Copyright © 2026 Vizrt

pxSaturation



The pxSaturation plug-in adjusts the color saturation of an image.

It can be applied to Everything, Shadows, Mid tones and Highlights. You also have the option to preserve highlights, or not.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

Copyright © 2026 Vizrt

pxStack



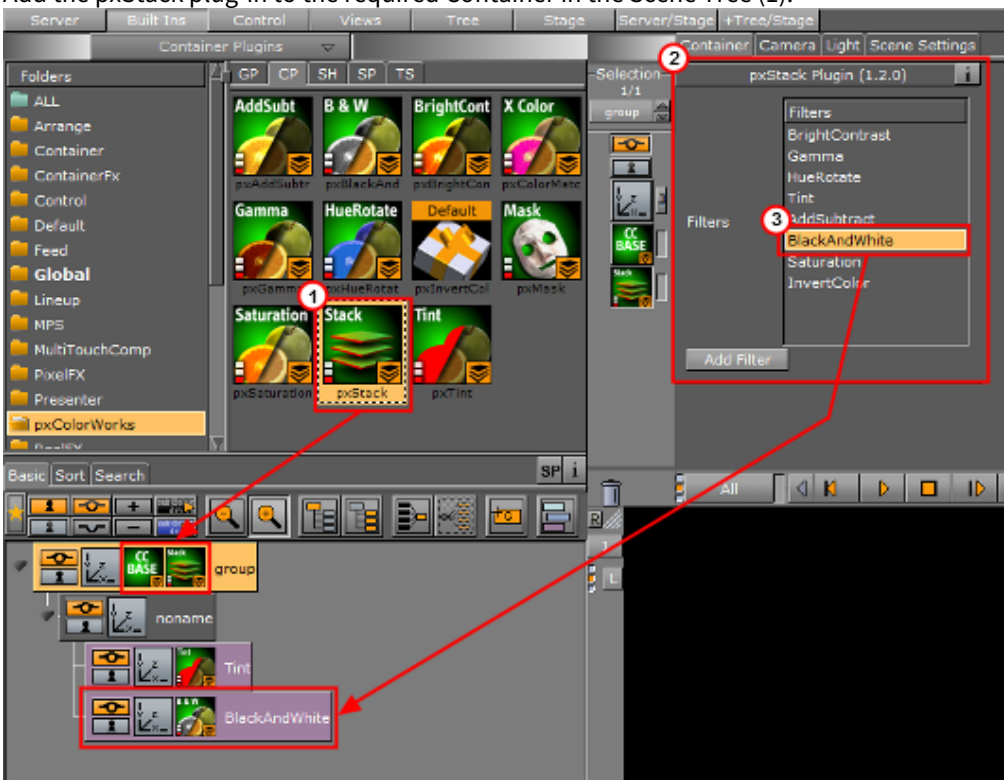
The pxStack plug-in applies multiple color corrections to an image in a specific order.

It can be applied to Everything, Shadows, Mid tones and Highlights. You also have the option to preserve highlights, or not.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

To Apply the pxStack Plug-in

1. Add the pxStack plug-in to the required Container in the Scene Tree (1).



Note: When the pxStack plug-in is added to a Container, it automatically adds the pxCCBase plug-in to the same Container. Also, it automatically generates a child Container to host the color correction nodes you define later.

2. Click the pxStack icon in the Container. The plug-in editor (2) shows a list of color correction filters.
3. Click on a color correction filter.
4. Click **Add Filter**. The color correction filter is added as a new Container under the 'noname' Container in the Scene Tree (3). This termed as a 'Color Correction Node'.
5. Repeat to add more color correction filters. Each color correction filter applied is added (stacked) as a new Color Correction Node under the 'noname' Container in the Scene Tree.
6. Click on a Color Correction Node to open its editor. Modify the effect properties as required.
7. Change the order of the stacked properties, as required.

Copyright © 2026 Vizrt

IMPORTANT! Selected properties act in the sequence in which they are ordered in the scene tree.

Note: To disable or enable a Color Correction Node click the Locked/Unlock icon in the top left of the Container.

Note: To remove a color correction function, delete the plug-in.

Copyright © 2026 Vizrt

pxTint



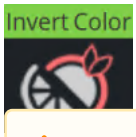
The pxTint plug-in adds a color tint and applies it to an image in the required amount.

It can be applied to Everything, Shadows, Mid tones and Highlights. You also have the option to preserve highlights, or not.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

Copyright © 2026 Vizrt

pxInvertColor



The pxInvertColor plug-in inverts the color value of an image.

It can be applied to Everything, Shadows, Mid tones and Highlights.

Note: This plug-in is located in: Plugins -> Container plug-ins -> pxColorWorks

Copyright © 2026 Vizrt

6.6 PixelFX Shader

The Pixel Shader plug-ins are part of the PixelFX plug-ins set.

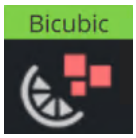
Viz Artist does not support more than one general Shader on a Container, or in a nested fashion. In some cases, PixelFX, by its nature, does support multiple Shaders per Container. Some have high cautionary indications. Some are very light. Some are stackable, and others are not.

The default path for the Geometry plug-ins is: *<viz install folder>\plug-in\<plug-in name.vip>*

The following Shader plug-ins are located in the *PixelFX* folder:

- PixelFXLensFlare
- pxBCubic
- pxCCBase
- pxEqualize
- pxGaussianBlur
- pxGradient
- pxInvert
- pxLensDistort
- pxMotionBlur
- pxNoise
- pxPixelate
- pxPosterize
- pxRecolor
- pxRipple
- pxSparkle
- pxTurbDissolve and pxTurbWipe
- pxTurbulence
- pxTwirl
- pxWaves

6.6.1 Common Properties



The PixelFX Shader plug-in icons themselves are divided in half with one side showing the visual effect with the plug-in and the other without it.

Copyright © 2026 Vizrt

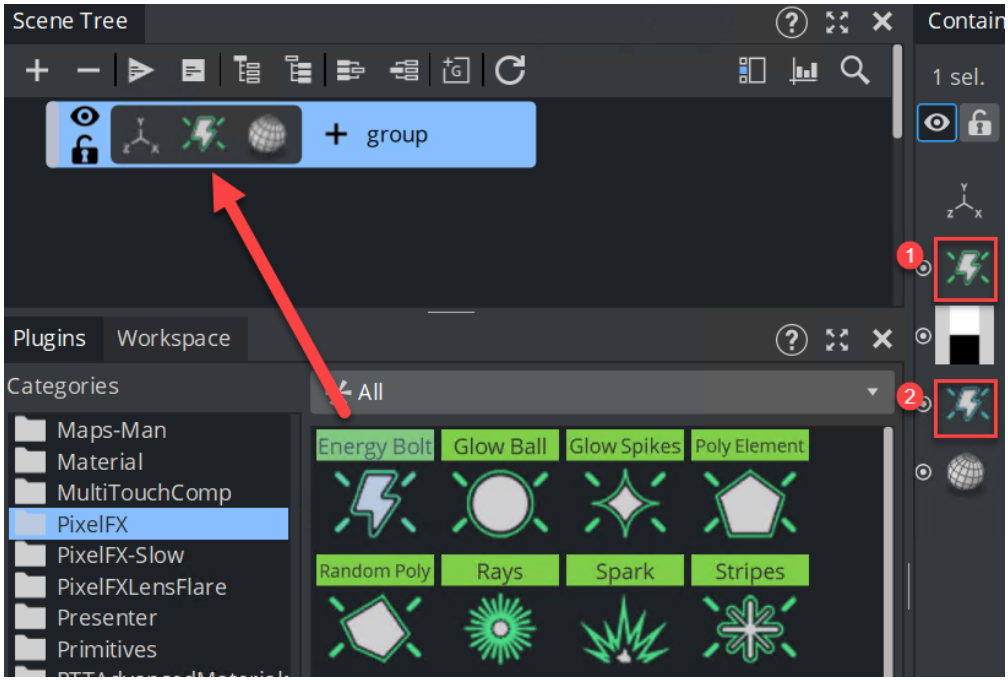
6.6.2 PixelFXLensFlare

The Lens Flare plug-ins, which are part of the PixelFX plug-ins set, simulate the effect of streaks and spots of light caused in real cameras when light enters and bounces inside a camera lens. This plug-in set contains geometries that simulate different shapes, which can be used for lens flare or simple flare effects.

The Flare draws itself as a combination of geometry Shader and pixel Shader.

When a Flare Container is created (1) a non-editable pixel Shader (2) is automatically added to the created Flair Container.

The non editable Shaders are needed to do more (detailed, on a per pixel basis) graphical effects on the GPU (manipulating colors, textures, lighting, etc.), which cannot be done with Geometries only.



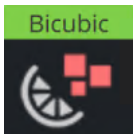
Note: This plug-in is located in: Plugins -> Shader -> PixelFX

See Also

- PixelFX Geometry Plug-ins in Basic Geometry Plug-ins

Copyright © 2026 Vizrt

6.6.3 pxBCubic



The BCubic plug-in introduces three bi-cubic algorithms. When dealing with images that contain line art or lines of high contrast, you may want to protect yourself from the less than the desirable effect of pixelation when an image is over scaled.

Each is useful in different contexts of its respective type. It is very easy to find the best adjustment to your specific scenario.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

6.6.4 pxCCBase



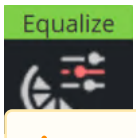
The pxCCBase plug-in (Color Correction) applies a cumulative set of color corrections using the pxStack plug-in.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

6.6.5 pxEqualize



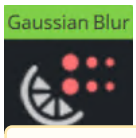
The pxEqualize plug-in allows you to adjust the black, mid-range and white color levels in an image.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

6.6.6 pxGaussianBlur



The pxGaussianBlur plug-in applies a Gaussian blur to the texture, smoothing uneven pixels in the image by removing outliers.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX-Slow

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Adjust the following parameters as required to achieve the required effect:

- X Radius
- Y Radius
- Quality
- Flip XY
- Debug

Copyright © 2026 Vizrt

6.6.7 pxGradient



The pxGradient plug-in has a similar functionality to the gradient function in Photoshop.

It can be applied on a container that holds a geometry, an image, or even an empty container. You can define as many gradient stops as you wish, as well as their origin points, angles, and extent. You can also define the gradient on the alpha channel (256 levels) and introduce turbulence noise that complies with the defined gradient.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

At the top of the pxGradient parameters, you see the controls that let you define the direction, radius and origin of the gradient. Below that you see a two radio button sets. The first lets you set the gradient type, followed by three Repeat types.

Below that, you see the color slider, the color values and alpha ramp settings.

Note: Under the title Color Ramp and Alpha Ramp, you have an active/inactive option, should you want to have an RGB ramp and no alpha ramp, or vice versa.

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Below the RGB and Alpha area of the editor, you find the option of adding Turbulence to the gradient. The turbulence has three simple parameters. Amount determines how aggressive the turbulence looks. Wavelength determines how big or small the turbulence increments are. Progress (similar to that in TextFX plug-ins) allows you to breathe life into the turbulence, providing interesting an effect when animated.

At the very bottom, it is important to make sure that you turn off the Show GUI button, whose default is `On`; if shown On Air, the effect is more embarrassing than dramatic.

Best Practices

It is important to understand that pxGradient is actually drawing pixels. If this plug-in is applied to an empty group or if it is applied to a geometry object that does not contain a texture, the gradient plug-in generates the texture into which it draws its pixels. So if you have a cube and add pxGradient to it, it behaves exactly as a texture. However it can also work in conjunction with a texture. If you then take an image and drop it on the cube, you give a color wash to the image that works in a gradient-like fashion. The gradient-generated texture can also be edited via the texture editor as any other texture would be.

The biggest benefit of using pxGradient is conservation of texture memory. Instead of storing very many different alpha and color ramps in your image pool, you can achieve the overwhelming majority of these effects by well thought out and judicious use of pxGradient. It has a minimum performance penalty and it can be animated. If you use ramps often in your designs, pxGradient is one of your handiest tools.

Copyright © 2026 Vizrt

Tips and Tricks

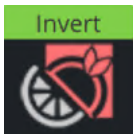
Often, reflection maps are placed on a translucent geometry to create a glass-like reflection (think of a light box or an aquarium).

For the reflection texture, one often uses an image of static noise or ramps of sorts. Since the texture is mapped on the object using a reflection mapping mode, it is very difficult to create the right intensity of noise in your texture so that the resulting reflection looks good.

The pxGradient turbulence functionality allows you to set the intensity of the noise of the texture “on the fly”, minimizing the round-trips to Photoshop.

Copyright © 2026 Vizrt

6.6.8 pxInvert



The pxInvert plug-in allows you to invert each of the color channels RGB and Alpha individually, and to use the RGB as alpha and the alpha as RGB.

Often you create an image with an alpha channel in Photoshop, import it to Viz Artist/Engine and only then realize that your alpha channel is inverted. Or you find an image in the Viz Artist/Engine pool that you want to use, but the way you want to use it is where the RGB is actually the alpha channel.

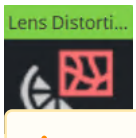
pxInvert allows you to perform these operations without adding yet another image to your image pool, or overloading your scenes with a few versions of the same image, where the difference is simple flips of the channels.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

6.6.9 pxLensDistort



The pxLensDistort plug-in creates a “fish eye” lens distortion effect on an image.

pxLensDistort can also be used to create the effect of an old television with a curved screen.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

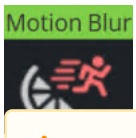
The amount of the effect can be configured using the following parameters:

- Amount
- 2nd Order
- Scale

The center around which the distortion goes can be configured by using the X Center and Y Center parameters.

Copyright © 2026 Vizrt

6.6.10 pxMotionBlur



The pxMotionBlur plug-in applies motion blur to the texture.

pxMotionBlur supports pxColorWorks and pxStack plug-ins.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

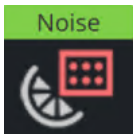
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Adjust the following parameters as required to achieve the required effect:

- X Center
- Y Center
- Rotate
- Zoom
- X Shift
- Y Shift
- Fix Edges (on/off)
- Samples (enter a value)

Copyright © 2026 Vizrt

6.6.11 pxNoise



The pxNoise plug-in is a white noise generator. It deploys the same turbulence as seen in other PixelFX plug-ins.

Similar to pxGradient, it generates its own texture if applied to a container without an image, or blends with the existing texture if the container has an image applied.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

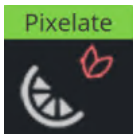
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Adjust the following parameters as required to achieve the required effect:

- **Size**
- **Amplitude:** Intensity/contrast of grains.
- **Color**
- **Speed:** Turbulence noise.
- **Stretch:** Stretches the grain to achieve a rain look of noise.
- **Stretch Angle**
- **Movement:** Transforms the generation of noise.
- **Transform as Texture**
  - **On:** Conforms to texture coordinates.
  - **Off:** Ignores texture coordinates and behave as stencil mapping.
- **Apply to: RGB, RGBA, Alpha:** Determines the channels in which the noise is generated.

Copyright © 2026 Vizrt

6.6.12 pxPixelate



The pxPixelate plug-in effect appears as if you reduce the resolution of your image, similar to the pixelate function in Adobe Photoshop. You may work in a proportional (locked) or disproportional mode.

To achieve a nice animated transition, tweak the Smoothness parameter so that the change between the different pixelation levels are smoother or harsher.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

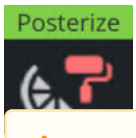
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Adjust the following parameters as required to achieve the required effect:

- Lock X/Y (on/off)
- X Size
- Y Size
- Smoothness
  Supports pxColorWorks and pxStack plug-ins.

Copyright © 2026 Vizrt

6.6.13 pxPosterize



The pxPosterize plug-in recolors an image using only a limited number of colors for a posterize effect.

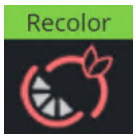
You can adjust the number of levels and set the transition to one of **Off**, **Fade** or **Glow**.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

6.6.14 pxRecolor



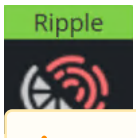
The pxRecolor plug-in ‘false-colors’ an image by using either Red, Green, Blue or Luma as the input to a color ramp.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Copyright © 2026 Vizrt

6.6.15 pxRipple



The pxRipple plug-in creates a pond ripple effect.

pxRipple supports pxColorWorks and pxStack plug-ins.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

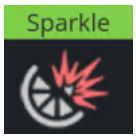
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Adjust the following parameters as required to achieve the required effect:

- X Center
- Y Center
- Radius
- Amount
- Wavelength
- Progress

Copyright © 2026 Vizrt

6.6.16 pxSparkle



The pxSparkle plug-in generates a sparkle pattern as a ray emitting from a point. The pxSparkle editor provides options to control the density of the rays, the x,y position from where they are emitted, the brightness (Amount) of rays, as well as other parameters, which is covered specifically in context. Before delving into these parameters however, it is important to understand the pxSparkle, like other PixelFX shaders (pxGradient, pxNoise) generates its own pixels. It also has the ability to be applied on top of an image and can blend with the image's pixels.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

When pxSparkle is applied to a container without an image, it creates sparkle results. You can also apply it on a container with an image. In this case, you need to decide whether the sparkle pattern should blend on top of the image or whether it cuts the image. If the *Cut Image* option is on, the host image is seen only in areas that are brighter than zero. The brighter the sparkle, the more apparent the host image is.

pxSparkle can be used as a flare to achieve its effect in a radius (the size of the hot spot). The *Inner Amount* (the brightness of the hot spot) and the *Radius* crops or fades the sparkle radially. You can also set the *Speed* parameter to set the sparkle in motion.

- **Transform as texture:** Forces the rendering of the sparkle to ignore the viewmatics. The sparkle is always facing the eye-point. This option is handy when applying the sparkle on top of a hierarchy.
- **Aspect:** Set the proportion of the sparkle independently of the texture transformation. Often, you stretch textures to get what you want. When applying pxSparkle to a distorted texture, you may want to maintain its roundness. You can use this parameter to compensate for the texture distortion.
- **Angle:** Rotates the sparkle without manipulating the texture coordinates. This option is handy when you have a static image on top of which you want to apply a rotating sparkle.

Copyright © 2026 Vizrt

6.6.17 pxTurbDissolve and pxTurbWipe



Both pxTurbDissolve and pxTurbWipe plug-ins utilize the native multi-texturing support of Viz Artist.

pxTurbWipe includes the parameters of Angle and Softness.

Supported Pipelines	
Classic Render Pipeline	✅
Viz Engine Render Pipeline	❌

To Apply pxTurbDissolve and/or pxTurbWipe

1. Insert an image into the scene tree.
2. Go to the image's texture editor and make sure that the texture Unit is set to `1` and the Inheritable option is `active`. This is your Image A in an A-B transition paradigm.
3. Place a child image underneath the first one. This is Image B.
4. In Image B's texture editor, set the Unit to `2` and make sure that the Inheritable option is `inactive`.
5. Apply pxTurbDissolve to Image A (the parent image).
6. In the plug-in editor, first adjust the Transition, followed by the other parameters, as per your liking.

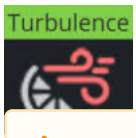
⚠ **Note:** This plug-in is located in: Plugins -> Shader -> PixelFX

See Also

- pxGradient

Copyright © 2026 Vizrt

6.6.18 pxTurbulence



The pxTurbulence plug-in creates a distortion effect using a Perlin turbulence function.

pxTurbulence supports pxColorWorks and pxStack plug-ins.

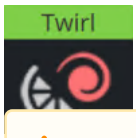
Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Adjust the wavelength, amplitude and speed to achieve the required effect.

Copyright © 2026 Vizrt

6.6.19 pxTwirl



The pxTwirl plug-in creates a distortion effect that twists an image around a central point.

pxTwirl supports pxColorWorks plug-ins and pxStack plug-ins.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

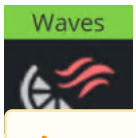
Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Adjust the following parameters as required to achieve the required effect:

- X Center
- Y Center
- Radius
- Amount
- Edge Softness

Copyright © 2026 Vizrt

6.6.20 pxWaves



The pxWaves plug-in creates a distortion effect that makes an image look wavy.

pxWaves supports pxColorWorks plug-ins and pxStack.

Note: This plug-in is located in: Plugins -> Shader -> PixelFX

Supported Pipelines	
Classic Render Pipeline	✓
Viz Engine Render Pipeline	✗

Adjust the following parameters as required to achieve the required effect:

- Type: Sine, Noise, Triangle, Square
- Angle
- Amount
- Wavelength
- Progress
- Seed

Copyright © 2026 Vizrt

7 Socialize Plug-ins



The Viz Social Plug-ins provide details about settings available through its configuration user interface within Viz Artist.

7.1 Related Documents

- *Viz Artist User Guide*: Contains information on how to create graphics scenes in Viz Artist.
- *Viz Engine Administrator Guide*: Contains information on how to install the Viz Engine software and supported hardware.
- *Viz Ticker User Guide*: How to install, configure and use the Viz Ticker client, and configure the output channels.
- *Viz Trio User Guide*: How to install, configure and use the Viz Trio client, and configure the output channels.
- *Viz Pilot User Guide*: How to install, configure and use Viz Pilot.
- *Viz Multichannel User Guide*: How to install, configure and use Viz Multichannel.
- *Feed Streamer User Guide*: Contains information on including information and graphics from various sources into scenes in Viz Artist.

7.2 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

7.3 Socialize Container Plug-ins

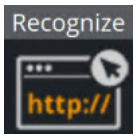
Viz Social offers a set of plug-ins that are bundled with the Viz Social installer. Most of the Viz Social plug-ins work in combination with the DataPool plug-ins.

The following Viz Social plug-ins are located in the Container Plugin tab (CP), under the SocialTV folder:

- HTTP Recognizer
- Playlist Reader
- Split Author
- Text Highlight
- WordCloud

Copyright © 2026 Vizrt

7.3.1 HTTP Recognizer



The STV_HTTP Recognizer plug-in recognizes a HTTP link in text geometry when placed on a text container.

In some cases, messages coming from social media networks such as Twitter or Facebook contain links to other sites. This plug-in hides the link from the message as these links provide no meaningful information for broadcasts.

The plug-in can additionally store the value of this HTTP link in a DataPool or shared memory variable.

> **Note:** This plug-in is located in: Plugins -> Container plug-ins -> SocialTV

HTTP Recognizer Properties

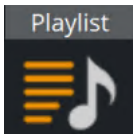
- **Work On Text Changes:**
- **Hide Links:** Specifies whether or not discovered links are to be hidden.
- **Target:**
  - **DataPool:**
  - **Shared Memory:**
- **Variable:** Sets the name of the variable in which the URLs of discovered links are to be stored.
- **>Use Object Container:** Allows one container to affect another container, or allows another instance of the same plug-in to be used on the same container. DataPool container plug-ins affect the containers they are attached to, and cannot affect other containers directly. When enabled, additional options are also enabled:
  - **Which:**
    - **PARENT:** The DataPool plug-in affects the parent container (i.e. the container residing above the current container in the scene hierarchy).
    - **GrParent:** The DataPool plug-in affects the grandparent container (i.e. the container residing two levels above the current container in the scene hierarchy).
    - **GrGrParent:** The DataPool plug-in affects the great grandparent container (i.e. the container residing three levels above the current container in the scene hierarchy).
    - **REMOTE:** When remote is selected another parameter, Remote Container, is enabled. Drag the container to be effected to the container place holder.
- **Shared Memory Variable Type:** Selects the shared memory area to update. Can be either *Global*, *Scene* or *Distributed*.
- **Apply:**
- **Init:**

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

7.3.2 Playlist Reader



The STV_PlaylistReader plug-in is used to read the list of pages inside a playlist, the playlist can be either from a Viz Trio Show or Viz Pilot/MOS Playlist.

STV_PlaylistReader is used in interactive applications to control and read all the pages that are in the Media Sequencer from the given Group of pages. The information retrieved from Media Sequencer is sent to a DataPool Array.

Note: This plug-in is located in: Plugins -> Container plug-ins -> SocialTV

Playlist Reader Properties

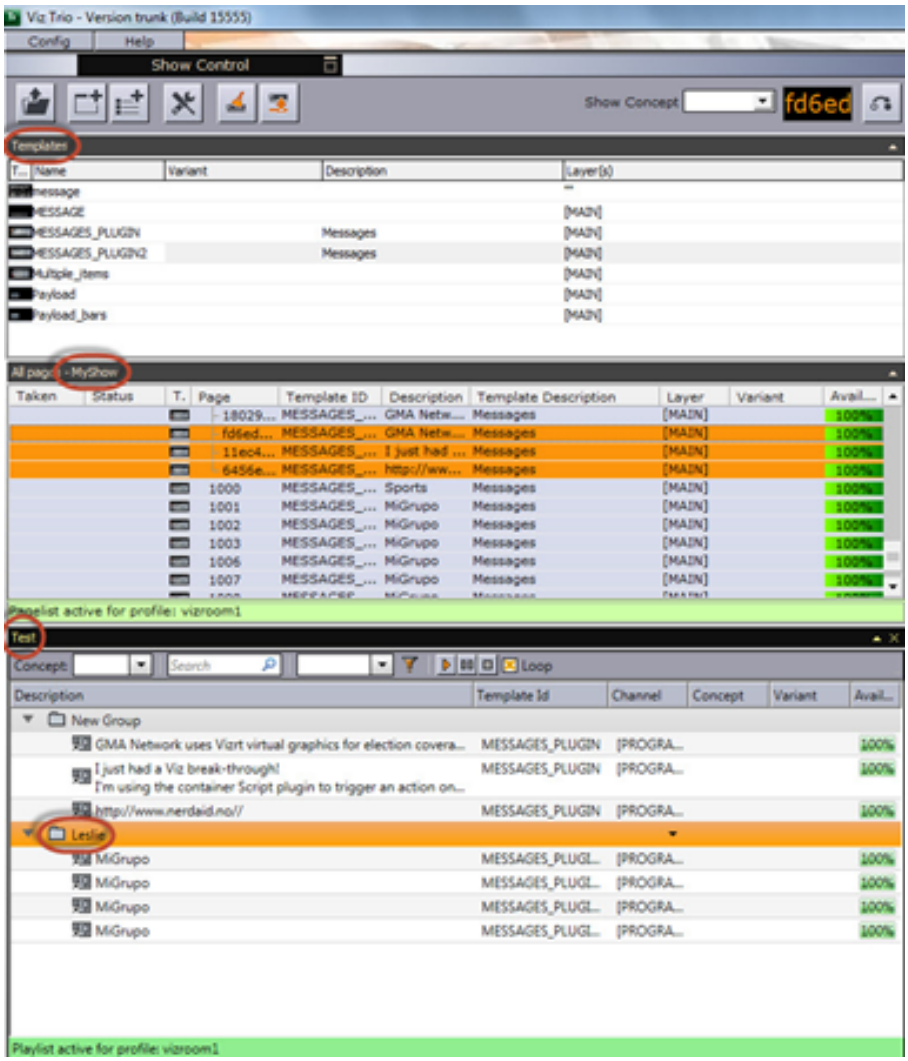
- **MSE Host:** Determines which Media Sequencer to connect to.
- **DP Field Name:** Defines where to send the information.
- **Mode:** Determines which mode to run in.
  - **Trio**
    - **Show Name:** Name of the Viz Trio Show (if not provided, it assumes the playlist is a Content Pilot or MOS playlist).
    - **Playlist:** Name of the Playlist.
    - **Group Name:** Name of the group of pages to retrieve.
    - **Page to take:** Index of page to control, starting from 0.
    - **Profile:** Name of the profile to use, profile must be defined in profile configuration in the Media Sequencer.
  - **Pilot/MOS**
    - **Playlist:** Name of the Playlist.
    - **Group Name:** Name of the group of pages to retrieve.
    - **Page to take:** Index of page to control, starting from 0.
    - **Profile:** Name of the profile to use, profile must be defined in profile configuration in the Media Sequencer.
  - **Ticker**
    - **CarouselIndex:**
    - **Channel Name:**
- **Post Update Action:** Specifies commands (either DataPool or Viz commands) to be executed right after the data is read and sent to the target. Multiple commands should be separated by semicolon. To distinguish between Viz Commands and DataPool commands, Viz Commands need to be prefixed with a zero and a space.
- **Show Data:** Dumps the data the plug-in reads to a DataPool variable called defined in Console Field Name when set to `On`. This is extremely useful for development and debugging purposes. This dump includes error messages too.
- **Console Field Name:** Determines which DataPool field to send to.
- **Number of Read Records Field:**
- **Log Level:** Selects the log level:
  - **Fatal:** Displays only fatal errors. This is the lowest level.
  - **Error:** Displays Fatal and Error messages.

Copyright © 2026 Vizrt

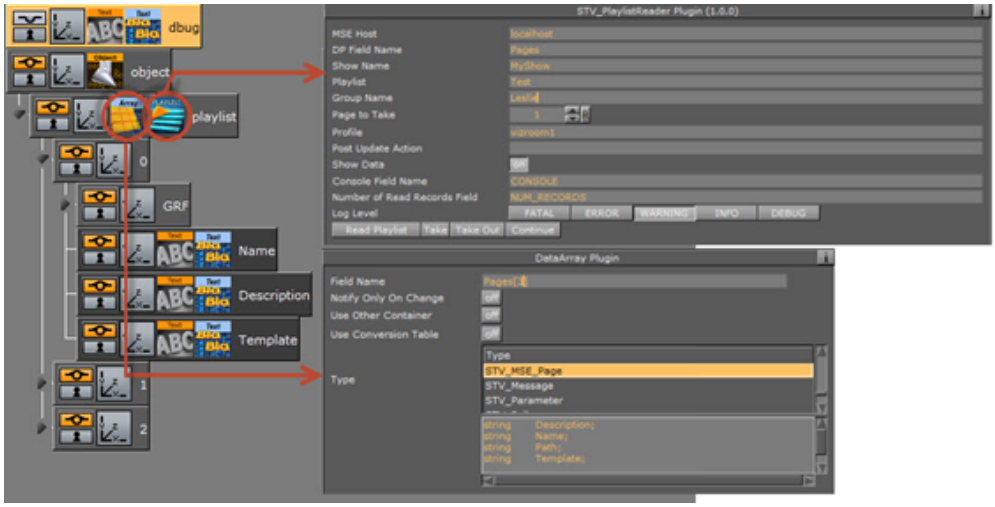
- **Warning:** Displays Fatal, Error and Warning messages.
- **Information:** Displays Fatal, Error, Warning and Information messages.
- **Debug:** Displays Fatal, Error, Warning, Information and Debug messages. This is the highest level.

- **Control buttons**
  - **Read Playlist:** Reads the items based on the properties defined in the plug-in.
  - **Take, Take Out and continue:** Performs the given action to the page defined in the *Page to Take property.*

This is an example of a playlist in Viz Trio:



Using the STV_Playlist plug-in in the Scene:





Configuration File

The Social TV package installs a config-SocialTV file located at: *C:\Program Files (x86)\vizrt\VizArtist*. The Config DP file includes a base type of a STV_MSE_Page:

```
STV_MSE_Page = {
          string Name;
          string Description;
          string Template;
          string Path;
}
```

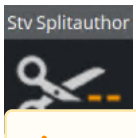
- **Name:** Returns the value of the page name.
- **Description:** Returns the value of the description in the Playlist. The Description is defined in the ControlObjectPlugin on the top of the Scene template.

Copyright © 2026 Vizrt

- **Template:** Returns the value of the Description from the template.

Copyright © 2026 Vizrt

7.3.3 Split Author



The STV_SplitAuthor plug-in recognizes the at (**@**) character from text.

This allows hiding either the user ID or user name of the author when placed on a text container.

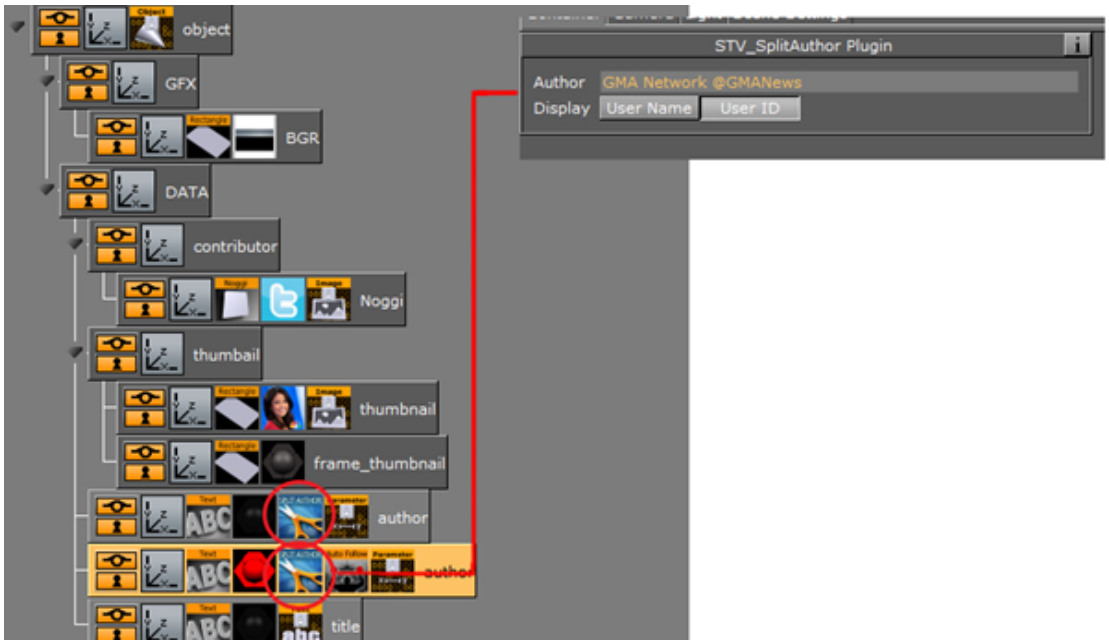
Note: You must send text to the *author* property of this plug-in using ControlParameter.

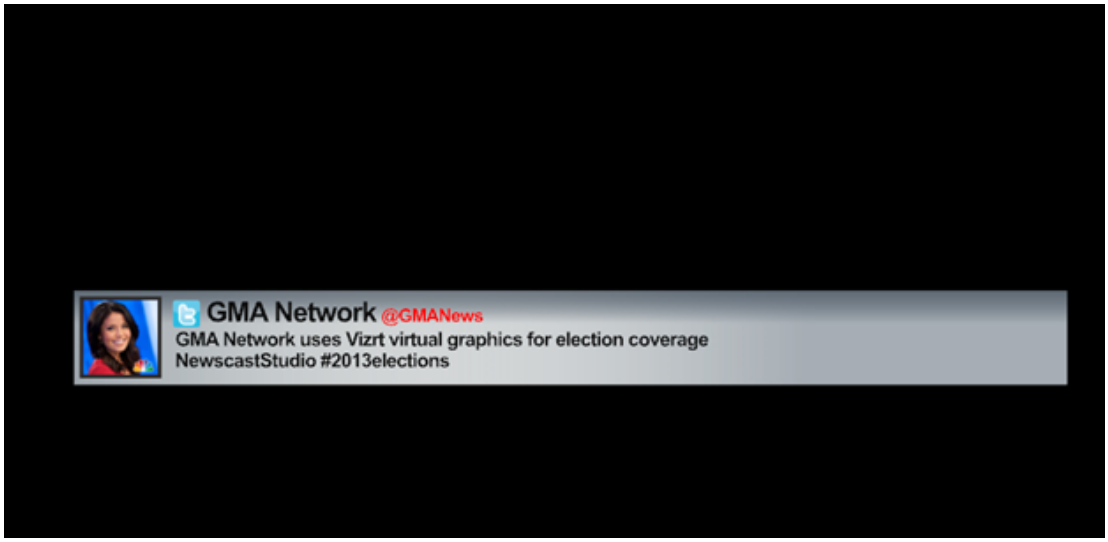
Note: This plug-in is located in: Plugins -> Container plug-ins -> SocialTV

Split Author Properties

- **Author:** Shows the author name (User name and User ID).
- **Display:** Selects whether to show the User Name or User ID.

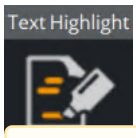
Example





Copyright © 2026 Vizrt

7.3.4 Text Highlight



The STV_TextHighlight plug-in recognizes certain characters and allows color changes to any word containing that character (for example: #tags @User http://) when placed on a text container the plug-in.

Note: You must send text to the *text* property of this plug-in using ControlParameter.

Note: This plug-in is located in: Plugins -> Container plug-ins -> SocialTV

Text Highlight Properties

Coloring

When Text Highlight is placed on a text container, the plug-in recognizes certain characters and lets you change the color of any word containing that character (for example, *#tags*, *@User*, *http://*).

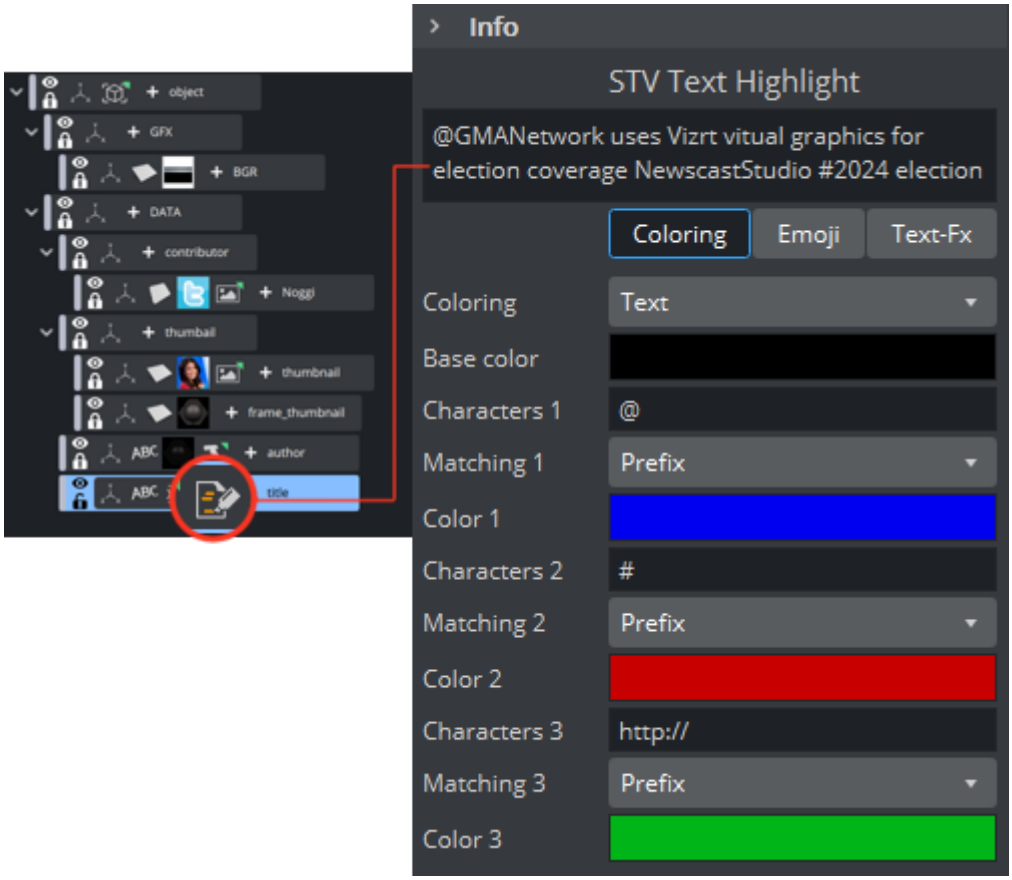
- **Coloring:** Toggles whether to highlight matched words.
- **Base color:** Sets color of characters that don't match any set.

Three sets of word searching methods:

- **Characters:** Characters, word or symbol to search for words to be highlighted.
- **Matching:** Searching methods:
  - **Prefix:** Searches all words started with specified **characters**.
  - **Suffix:** Searches all words ended with specified **characters**.
  - **Regex:** Uses **characters** as regular expression for searching (for example, `.*\d.*` matches all words that contain digits).
- **Color:** Sets color of characters in matched words.

Copyright © 2026 Vizrt

Example





Emoji

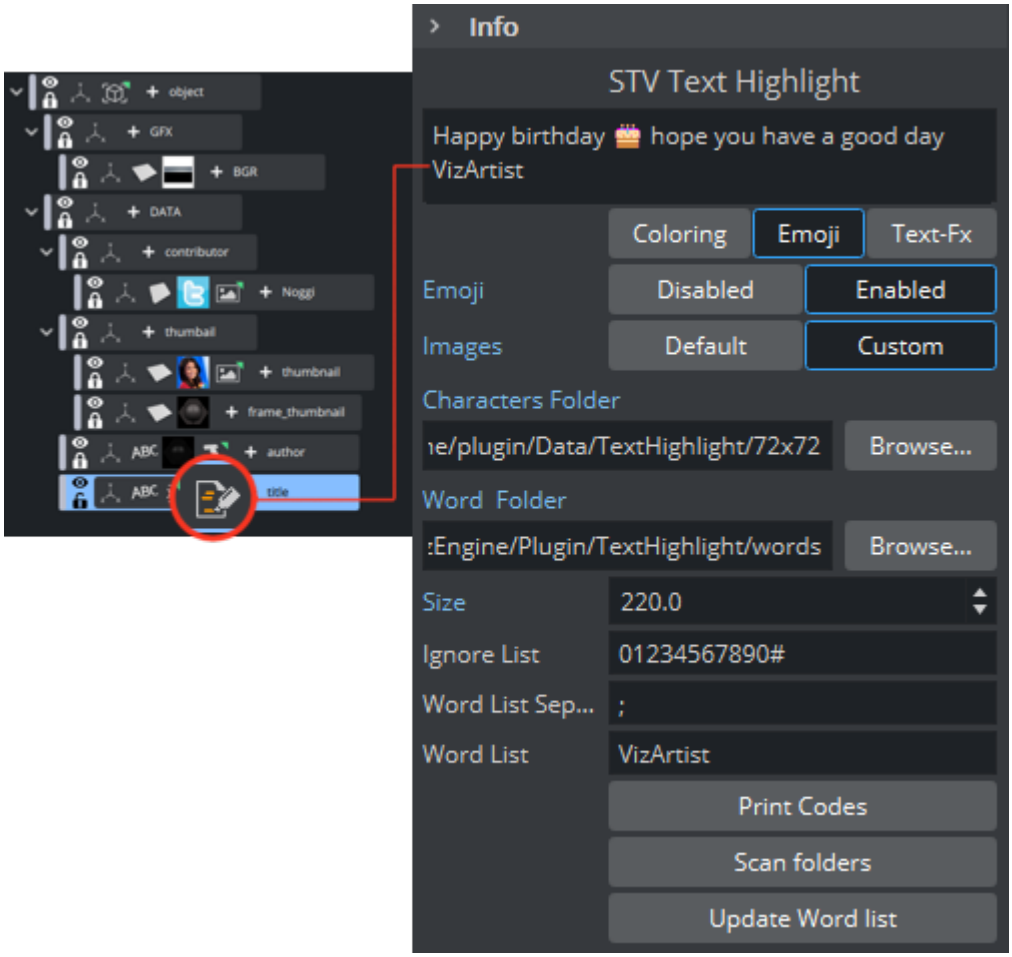
- **Emoji:** Toggles whether to highlight emoji content.
- **Images:** Default/Custom.
  - When using *default* the plug-in uses the images located in *C:\Program Files\Vizrt\VizArtist\plugin\data\TextHighlight\72x72.*
  - When using *custom* the user defines a path to their own images.

Copyright © 2026 Vizrt

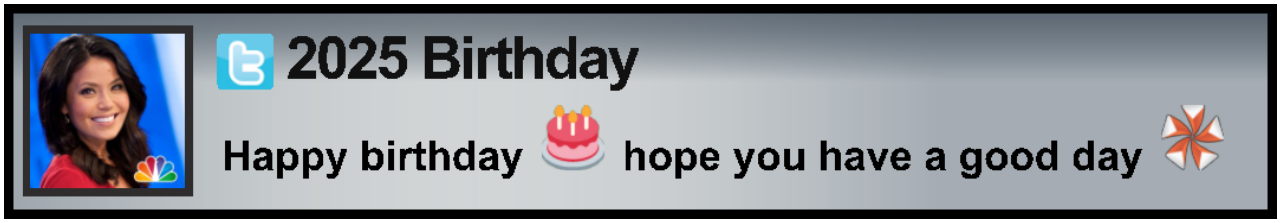
Note: The custom path should be accessible to all machines running Viz Engine, including Viz Trio and Preview Engines.

- **Characters Folder:** Defines a custom folder of images containing characters that should be replaced by emojis. The default folder is `%ProgramFiles%\Vizrt\VizEngine\Plugin\Data\TextHighlight\72x72`.
- **Word Folder:** Defines a folder of images containing words that should be replaced by emojis.
- **Size:** Sets size of emojis (percentage).
- **Ignore List:** Defines a list of characters that should be ignored when replacing with emojis.
- **Word List Separator:** Sets the character to be used as a separator of the list of words to replace by emoji.
- **Word List:** Defines a list of words to be replaced by emojis.
- **Print Codes:** Prints Unicode and matches image of all characters in the text in the console in hexadecimal.
- **Scan Folders:** Scans and updates list of images in **Characters Folder.**
- **Update Word List:** Scans and updates list of images in **Word Folder** and update **Word List.**

Example



Copyright © 2026 Vizrt



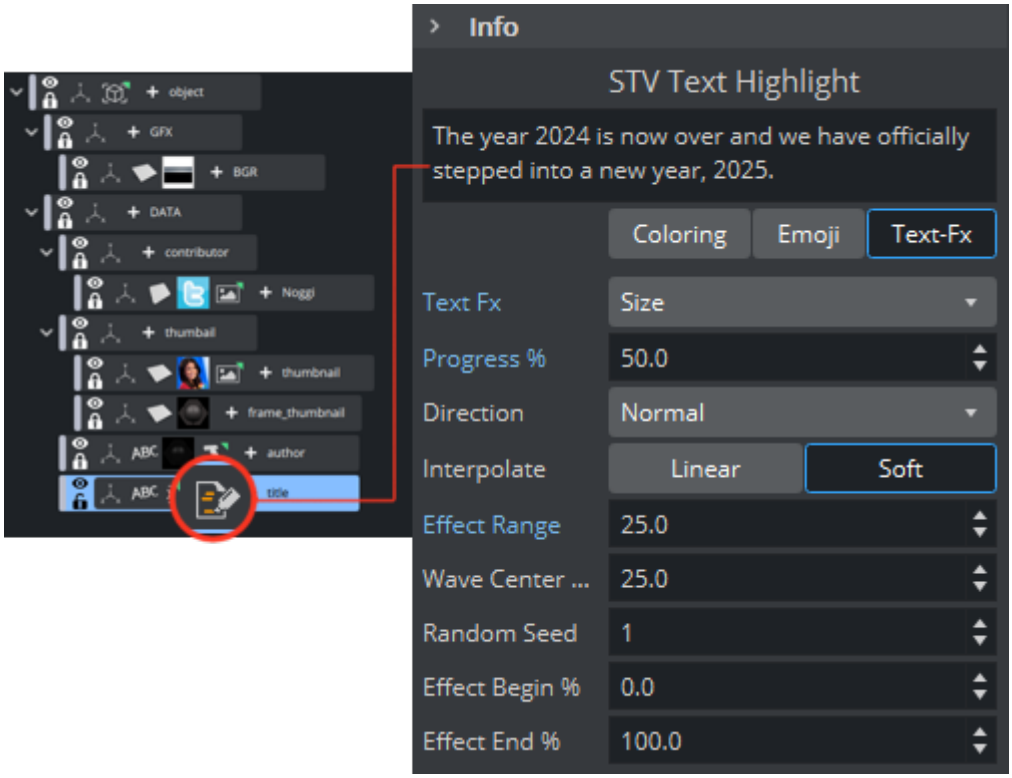
Text FX

The Text FX option allows users to apply similar effects as Text FX plug-ins when using emojis. For more information please refer to section Basic Container TextFX.

- **Text Fx:** Toggles whether to apply progress effects, alpha, size and scale.
- **Progress %:** 0 percent progress represents the beginning of the effect, 100 percent the end. Animate this value from `0` to `100` to see the effect or from `100` to `0` to animate the effect backwards.
- **Direction:** Sets the direction of the text effect sequence.
  - **Normal:** Starts with the first character.
  - **Opposite:** Starts with the last character
  - **Random:** Uses a random order.
  - **Static:** Processes all characters at the same time.
  - **Wave:** Starts with the first character, animates the effect from 0 to 100 and then down again to 0.
  - **Center:** Starts the effect from the center of the text.
  - **2 Center:** Starts the effect at the same time from the beginning and the end of the text. They meet at the center.
- **Interpolate:** Transitions from character to character. Choose between a **soft** or a **linear** interpolation.
- **Effect Range:** Defines how many characters are processed at the same time. If for example the Effect Range is set to `4`, and you manually increase the progress value, you see that when the fifth character starts to be processed, the first is finished, when the sixth starts, the second is finished, and so on.
- **Wave Center Size %:**
- **Random Seed:** Specifies a seed for the random number generator when a random direction is chosen. Even though Viz Artist uses random numbers, the animation for a specific random seed always looks the same. This is typically useful if you combine two different text effects.
- **Anchor Horizontal:** X-pivot of characters scaling.
- **Anchor Vertical:** Y-pivot of characters scaling.
- **Use Axis:** Scales characters on an axis.
- **Effect Begin %:** Effect at 0% progress.
- **Effect End %:** Effect at 100% progress.

Copyright © 2026 Vizrt

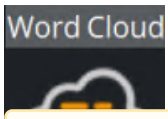
Example





Copyright © 2026 Vizrt

7.3.5 WordCloud



The WordCloud plug-in reads an XML file to create word cloud type visualizations.

WordCloud allows users to read items from the DataHub that are compatible with the plug-in.

Note: This plug-in must be placed on top of a text geometry.

Note: This plug-in is located in: Plugins -> Container plug-ins -> SocialTV

Examples are feeds from Dizplai World Cloud, Spredfast Top Terms, etc. The plug-in then visualizes the data in the form of a Word Cloud.

WordCloud is compatible with Dizplai's XML format, but it can read any XML in the following structure:

```
<datasource>
<entry>
<field name="text">This</field>
<field name="count">967</field>
</entry>
<entry>
<field name="text">is</field>
<field name="count">772</field>
</entry>
<entry>
<field name="text">a</field>
<field name="count">150</field>
</entry>
<entry>
<field name="text">sample</field>
<field name="count">144</field>
</entry>
<entry>
<field name="text">XML</field>
<field name="count">119</field>
</entry>
<entry>
<field name="text">file</field>
<field name="count">106</field>
</entry>
</datasource>
```

WordCloud Properties

- **Font Type:** Regular Complex
- **DataHub Host: (Deprecated)** Defines the hostname of the DataHub to communicate with.
- **DataHub Port: (Deprecated)** Sets the port for connecting to the DataHub. The default is `8089`.
- **DP Field Name:** Returns data to this field.

Copyright © 2026 Vizrt

- **Load Automatically**: Retrieves data automatically from the DataHub at determined intervals.
- **Group Name:** Includes messages that come from specific groups.
- **Source Name:** Includes messages that come from specific sources.
- **Provider Name:** Includes messages that come from a specific provider (for example Facebook, Twitter etc.).
- **Free Text Search:** Searches messages for text included in this parameter.
- **Disable Sort:**
- **Custom Sort:** Sorts retrieved data by the selected parameter.
    - **All:** Loads all messages.
    - **Approved:** Loads approved messages.
    - **Rejected:** Loads the rejected messages.
- **Filter Rejected:**
- **Favorites:** Includes the messages from the Favorites group.
- **Area Width:** Determines the width of the word cloud text region.
- **Area Height:** Determines the height of the word cloud text region.
- **Margin:** Determines the margin around the word cloud text region.
- **Layout:** Creates either a square or round word cloud layout.
- **Seed:** Applies some randomization value to the layout of words.
- **Number of Words:** Limits the number of words to use in the Word Cloud visualization (uses the highest volume words).
- **Color:** Applies an image for coloring each word in the cloud when set to `Image`.
- **Shared Memory:** Scene, Global, Distributed.
- **Scale Ranks:** Allows scaling words based on ranking.
- **Clip Ranks:** Allows clipping words with a value lower or higher than what is defined in the plug-in.
- **Render Bounding Boxes:** Renders the bounding box.

Copyright © 2026 Vizrt