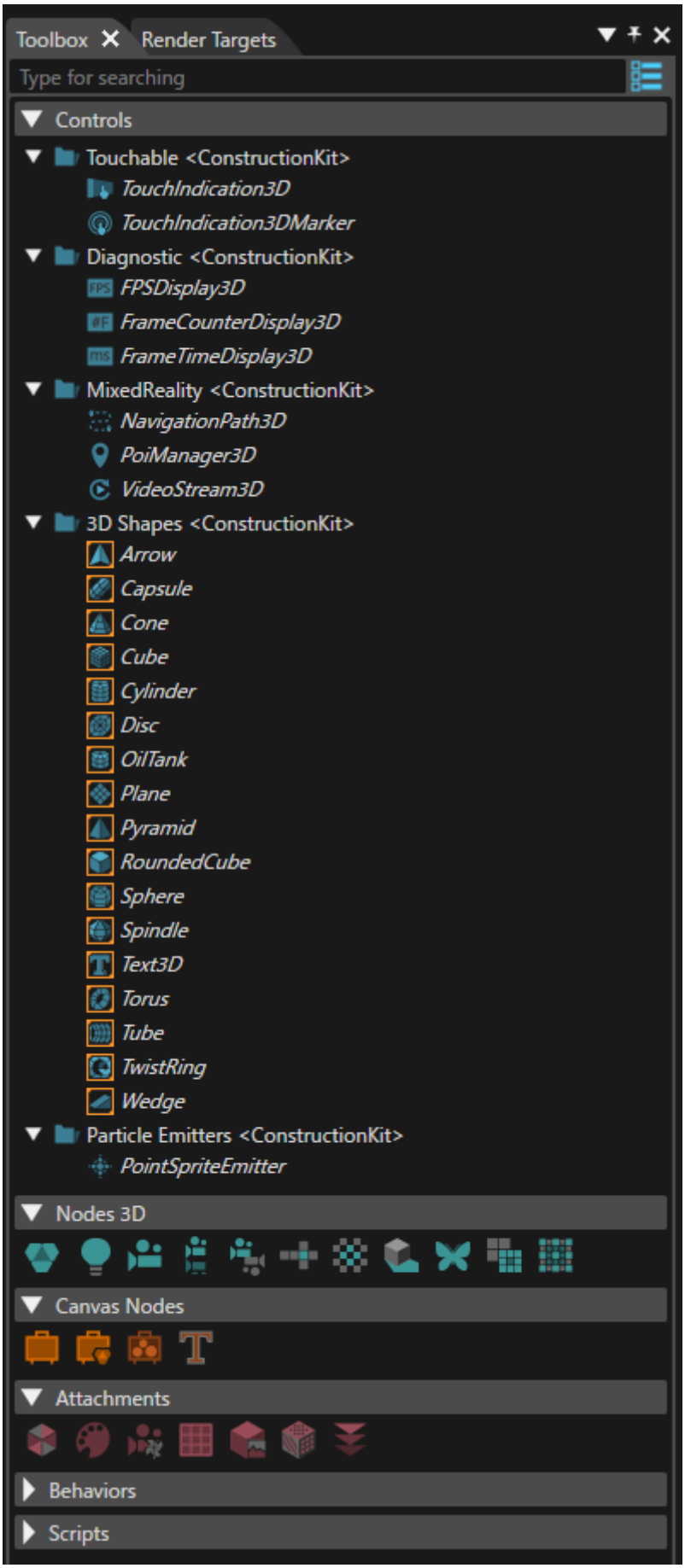
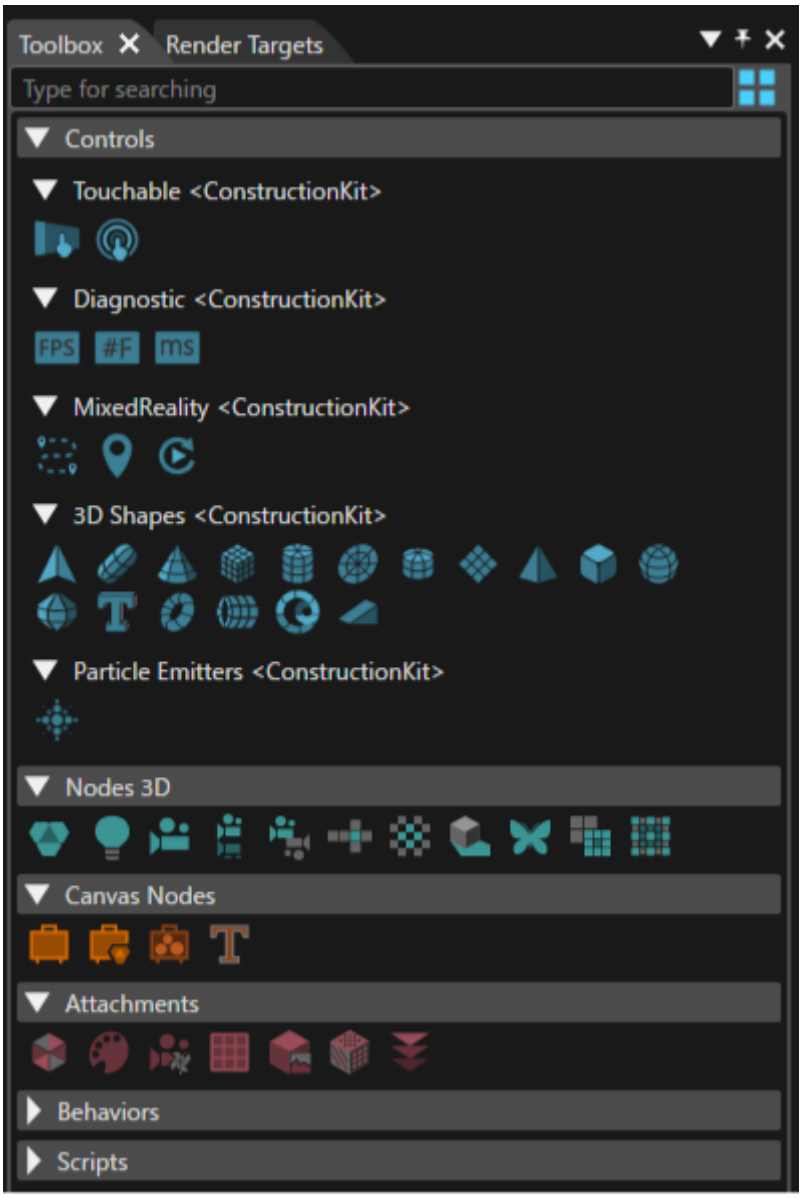
3D Nodes

3D Toolbox

The Toolbox for 3D scenes provides the following items:


Toolbox
Render Targets
Type for searching
Controls
Touchable <ConstructionKit>
TouchIndication3D
TouchIndication3DMarker
Diagnostic <ConstructionKit>
FPSDisplay3D
FrameCounterDisplay3D
FrameTimeDisplay3D
MixedReality <ConstructionKit>
NavigationPath3D
PoiManager3D
VideoStream3D
3D Shapes <ConstructionKit>
Arrow
Capsule
Cone
Cube
Cylinder
Disc
OilTank
Plane
Pyramid
RoundedCube
Sphere
Spindle
Text3D
Torus
Tube
TwistRing
Wedge
Particle Emitters <ConstructionKit>
PointSpriteEmitter
Nodes 3D
Canvas Nodes
Attachments
Behaviors
Scripts

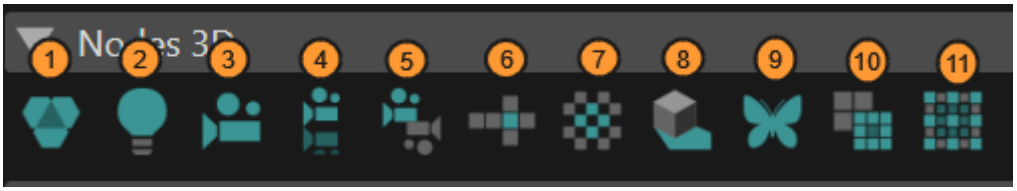
> A "Show Controls as Big Icons/List" is available in the upper right corner. If this buttonis used, a different presentation mode - as icons instead of a text list of items - will be made available (see the image below).



Create 3D Nodes

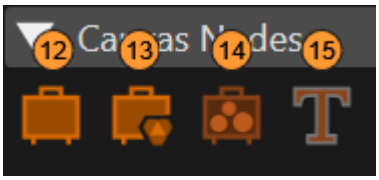
Following items can be used to create a new node in the scene graph:

3D Nodes



1	**Group:** Represents a parent container for nodes or other groups. Candera::Group
2	**Light:** Different types of light could be added to illuminate the scene objects. Read more... Candera::Light
3	**Camera:** To be linked to a render target, where the render result of the camera will be drawn. Read more... Candera::Camera
4	**Reflection Camera:** Creates a reflection of a specific camera on a given reflection plane. Read more... Candera::ReflectionCamera
5	**Stereo Camera:** Creates the effect of stereo 3D. Read more... Candera::StereoCamera
6	**Sky Box:** A type of mapping that gives the user the feeling of immersion into a large environment. Read more... Candera::SkyBox
7	**Billboard:** A textured or colored 2D area. Candera::Billboard
8	**Planar Shadow:** Provides the possibility to "throw" some shadows on a configurable planar surface. Read more... Candera::PlanarShadow
9	**MorphingMesh:** A mesh that stores weight values for morphing. Read more... Candera::MorphingMesh
10	**LodNode:** A node which accommodates multiple nodes, each of them representing a certain, predefined level of detail. Read more... Candera::LodNode
11	**PointSprite:** Displayed as a flat square polygon which is always facing the camera (screen-aligned). It can be textured and scaled to simulate perspective. Candera::PointSprite

Canvas Nodes



12	**Canvas:** A node which acts as a surface. Read more... Candera::Canvas
13	**CanvasGroup:** A node which can contain other canvas nodes. Read more... Candera::CanvasGroup
14	**CanvasSprite:** A node which represents a solid object, offering the possibility to render a texture, a flat colored rectangle or other visual effects. Read more... Candera::CanvasSprite
15	**CanvasText:** A node which facilitates displaying text on Canvas. Candera::CanvasText

Create a new 3D node by drag & drop

- on the Scene Tree to sort the node properly into the scene graph
- directly into the render view of the Scene Editor to create the node on a specific display coordinate.

New scene nodes are named like the item type, for more than one item suffixed with a number.

Refer to the Properties panel for inspecting all properties specific for the type of node. Tooltips are available to explain the purpose of the property. Refer also to the Properties Overview listing the Tooltip descriptions.

Light

Types of Light

Different kinds of light sources are available:

1 Ambient Light

Ambient light illuminates the scene from all directions. The light intensity is the same everywhere in the scene.
Position and direction of an ambient light source have no effect.

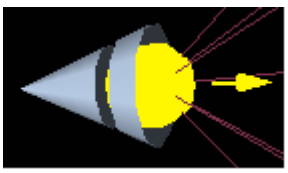
2 Directional Light

Directional light corresponds to sunlight in the real world. Illuminates all objects in the scene from the same direction, and with a constant intensity.
Position and attenuation of a directional light source have no effect.

3 Point Light

Point light casts equal omnidirectional illumination from the position of the light source. The intensity of light coming from a point light source can diminish with distance.
The direction of a point light has no effect.

4 Spot Light



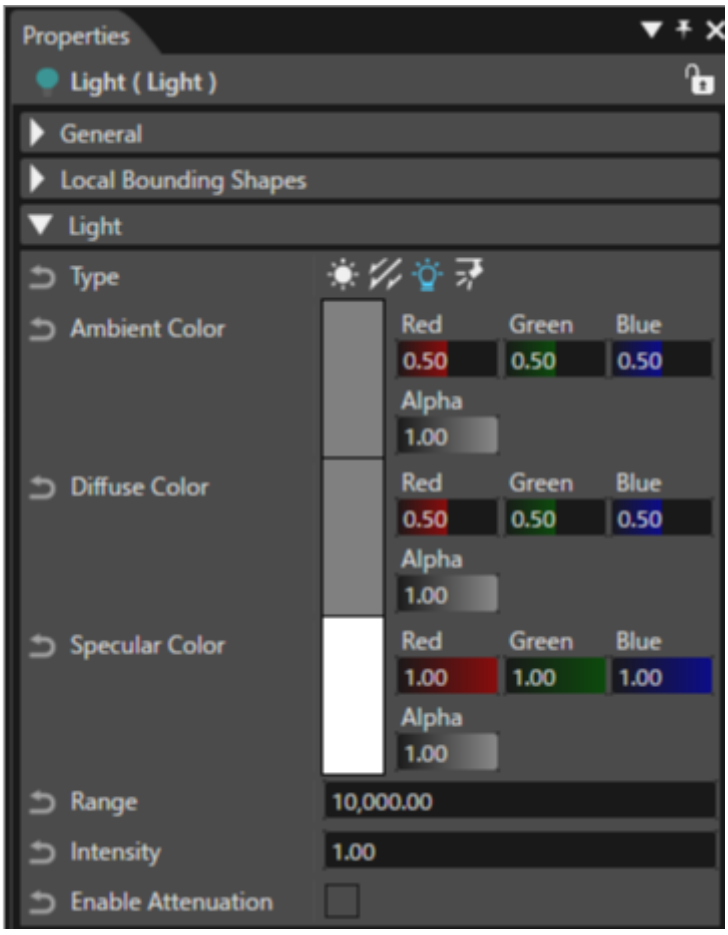
Spot light casts a light cone from a position centered around a direction.
A spot light has both: a direction and a position.

Light Properties

Please refer to section Light Gizmos for more details on how to edit lights by using gizmos.

Light properties that can be edited from the Properties panel (Some properties are only applicable depending on the selected light type):

- **Type:** The type of a light source can be changed at any time. This might be useful for switching to a simpler lightning model. Illumination is enabled, if light has rendering enabled and if the node is in the same scope as the light.
- **Ambient Color:** defines the ambient color component that illuminates the ambient color component of the material.
- **Diffuse Color:** illuminates the diffuse color component of the material. It is ignored if type of light is Ambient.
- **Specular Color:** defines the specular color component that illuminates the specular color component of the material. Ignored if type of light is Ambient.
- **Direction:** defines the direction of the light. It is ignored if type of light is Ambient or Point.
- **Range:** defines the range from light to geometry in world space where illumination is applied. It is ignored if type of light is Ambient or Directional.
- **Intensity:** The intensity factor for the diffuse color of the light. Ignored if type of light is Ambient.
- **Enable Attenuation:** Enables or disables the attenuation of light. The greater the distance of an object's geometry to the light, the less illumination is applied. This property will be ignored if the light type is Ambient or Directional.
- **Constant Attenuation Factor:** defines the constant attenuation factor of the light. The attenuation equation is: 1 / (constant + (linear * distance) + (quadratic * distance^2)). It is ignored if type of light is Ambient or Directional.
- **Linear Attenuation Factor:** defines the linear attenuation factor of the light. It is ignored if type of light is Ambient or Directional.
- **Quadratic Attenuation Factor:** defines the quadratic attenuation factor of the light. It is ignored if type of light is Ambient or Directional.
- **Spot Angle:** defines the spot angle of the light cone in degrees. It is ignored if type of light is Ambient, Directional, or Point.
- **Spot Exponent:** the distribution of the light within the spot light cone. It is ignored if type of light is Ambient, Directional, or Point.



Light properties that can be edited in ShaderParamSetter:

- **CoordinateSpace:** defines the coordinate space for lighting calculations in shader. It has two possible options:
  > Object - Light is transformed into object space to calculate vertex illumination
  > World - vertex and normal are transformed into world space to calculate vertex illumination. Object space is less computation intensive and is preferred.
  However, to assure accurate illumination of non-uniform scaled objects, lighting in world space is indicated.

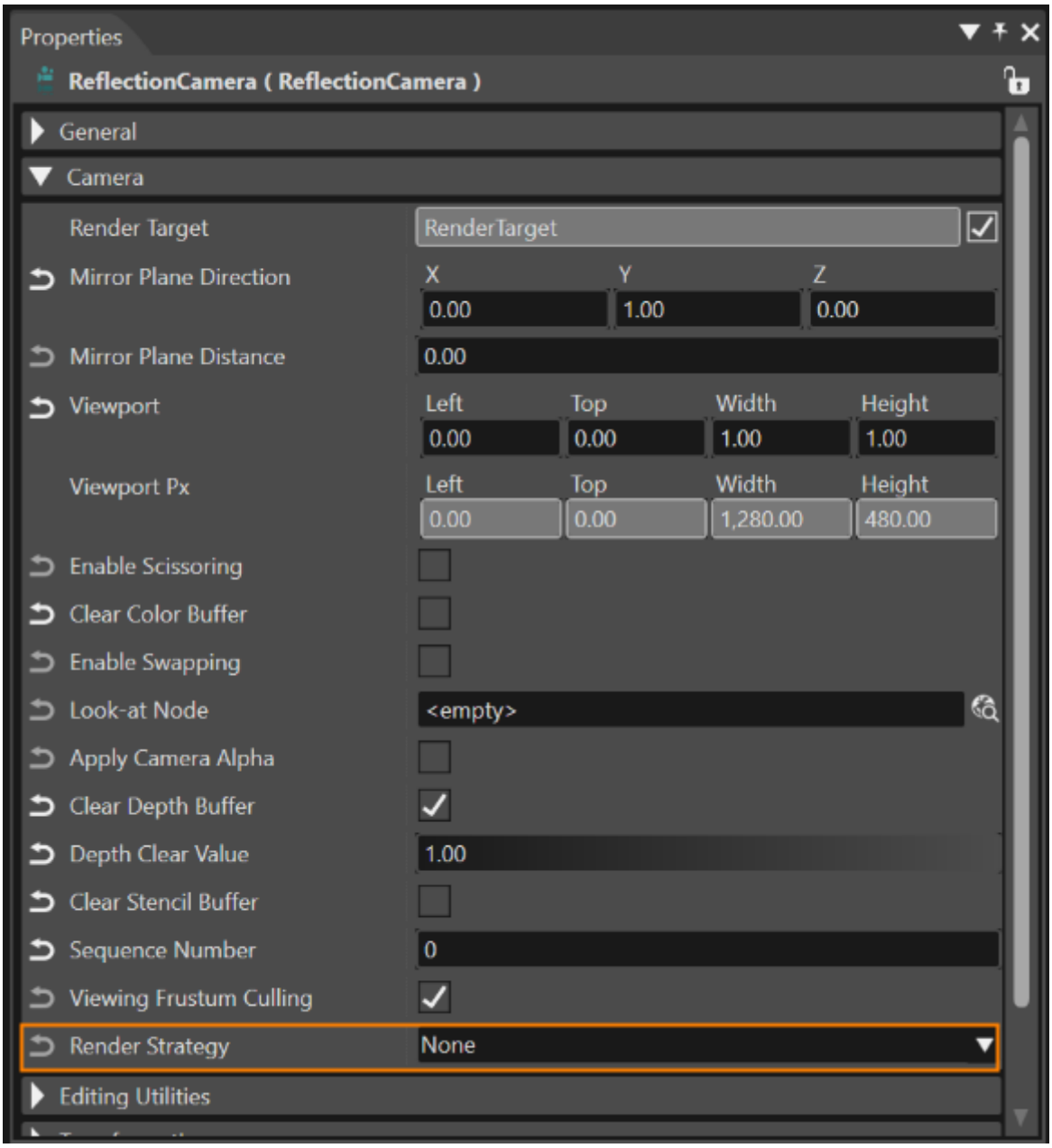
Camera

Refer to the Properties panel for inspecting all properties specific for a 3D camera.
Tooltips are available to explain the purpose of the property.

Please refer to section Camera Manipulation for details on how to manipulate cameras.

Camera Render Strategy

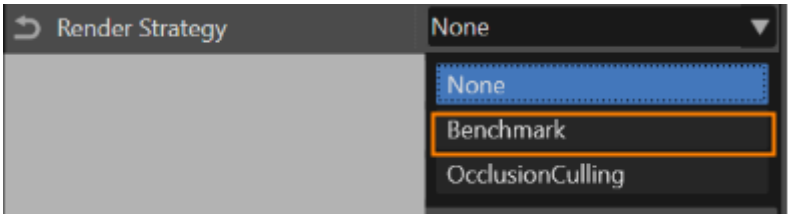
Camera, Reflection camera and LeftEye/RightEye (StereoCamera) objects have a property (available in Properties panel) called "RenderStrategy" that only affects the processing of that camera and its nodes.



Per default "none" is selected as camera render strategy, which means the nodes will be rendered inside a render pass according to the defined render order.

Benchmark Render Strategy

Select "Benchmark" as render strategy to split a render call into several passes.



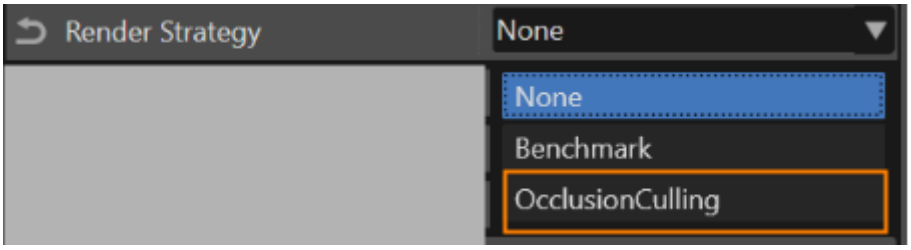
In a render pass, only that part of the scene graph is rendered, which node's benchmarks are beyond a given threshold. In the next render call, rendering will resume at the previous stop point, until it is complete.

Two properties are relevant to achieve this behavior:

- **"Render Benchmark":** A render measure value of the node, given an average time required to rendered that node. Render benchmarks can be calculated by CGI Analyzer and imported into SceneComposer.
- **"Benchmark Threshold":** Only nodes with an accumulated node render benchmark value smaller than or equal to the given threshold will be rendered during a single render call. Otherwise rendering is paused and the remaining nodes will only be rendered in the next step.

Occlusion Culling

Select "Occlusion Culling" as render strategy to have a new property available in property grid: QueryType.

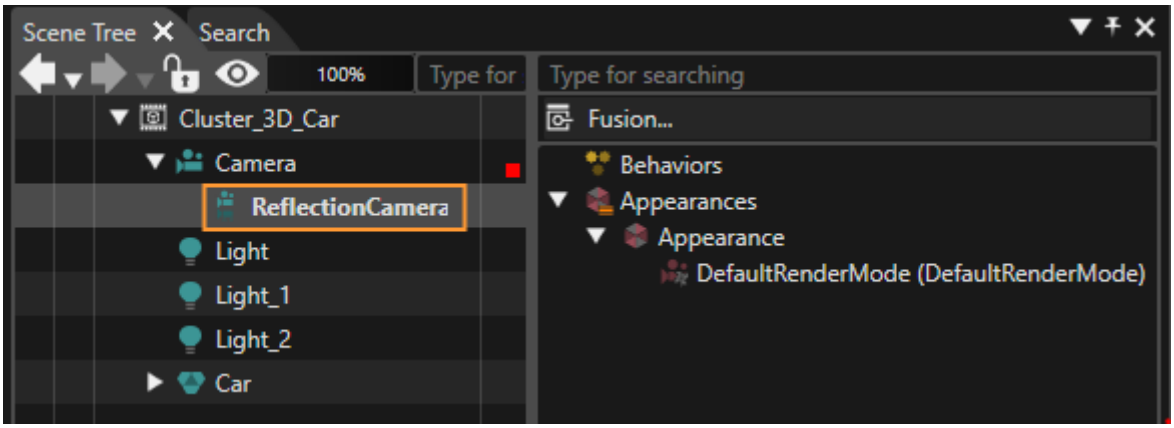


> Occlusion culling render strategy applies only to OpenGLES 3.0 platforms.

The render strategy uses a dynamic property (QueryProperty) per node to determine if an occlusion query should be issued or not. It also stores the query result for that node in that property.

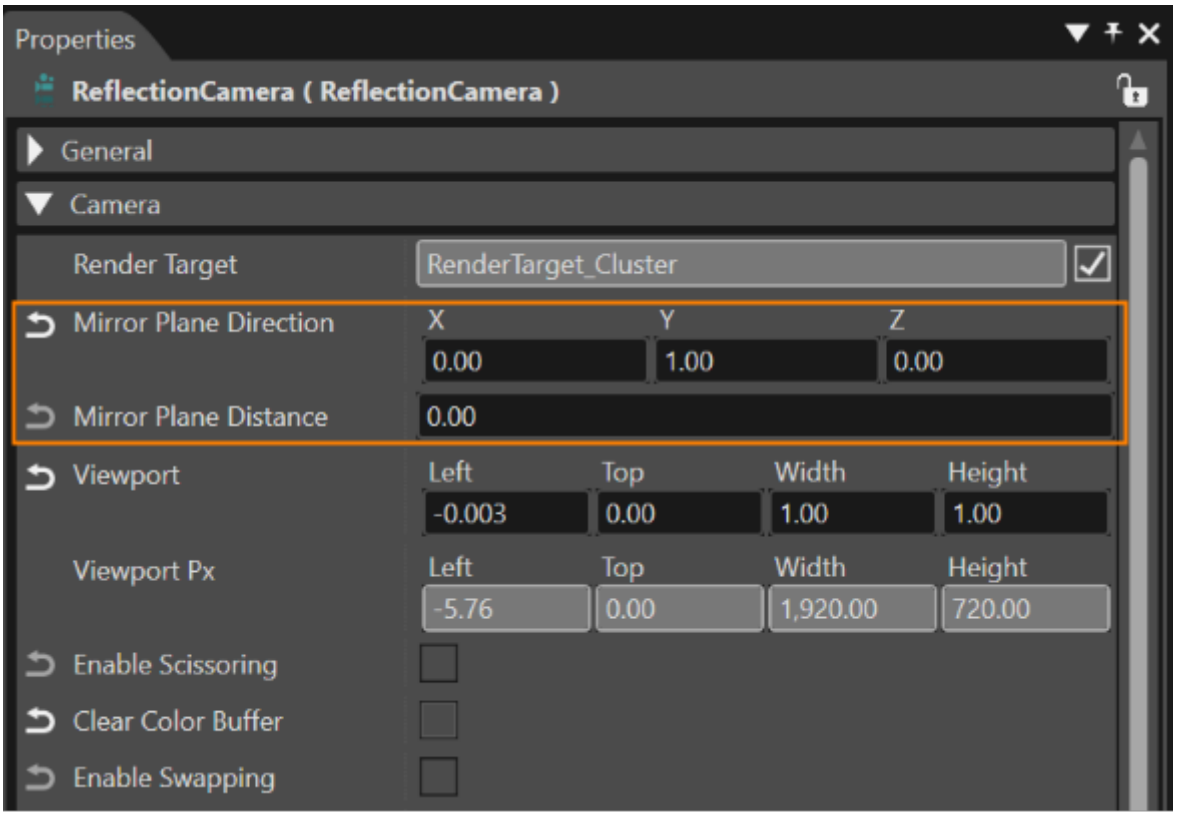
Reflection Camera

Reflection camera is a special item in the Toolbox that can be added in scenes only under camera nodes using drag & drop.



The camera will have some modified properties after adding a reflection camera on it. After the reflection camera is removed, the camera may still have some modified properties which have been previously required for the reflection camera.

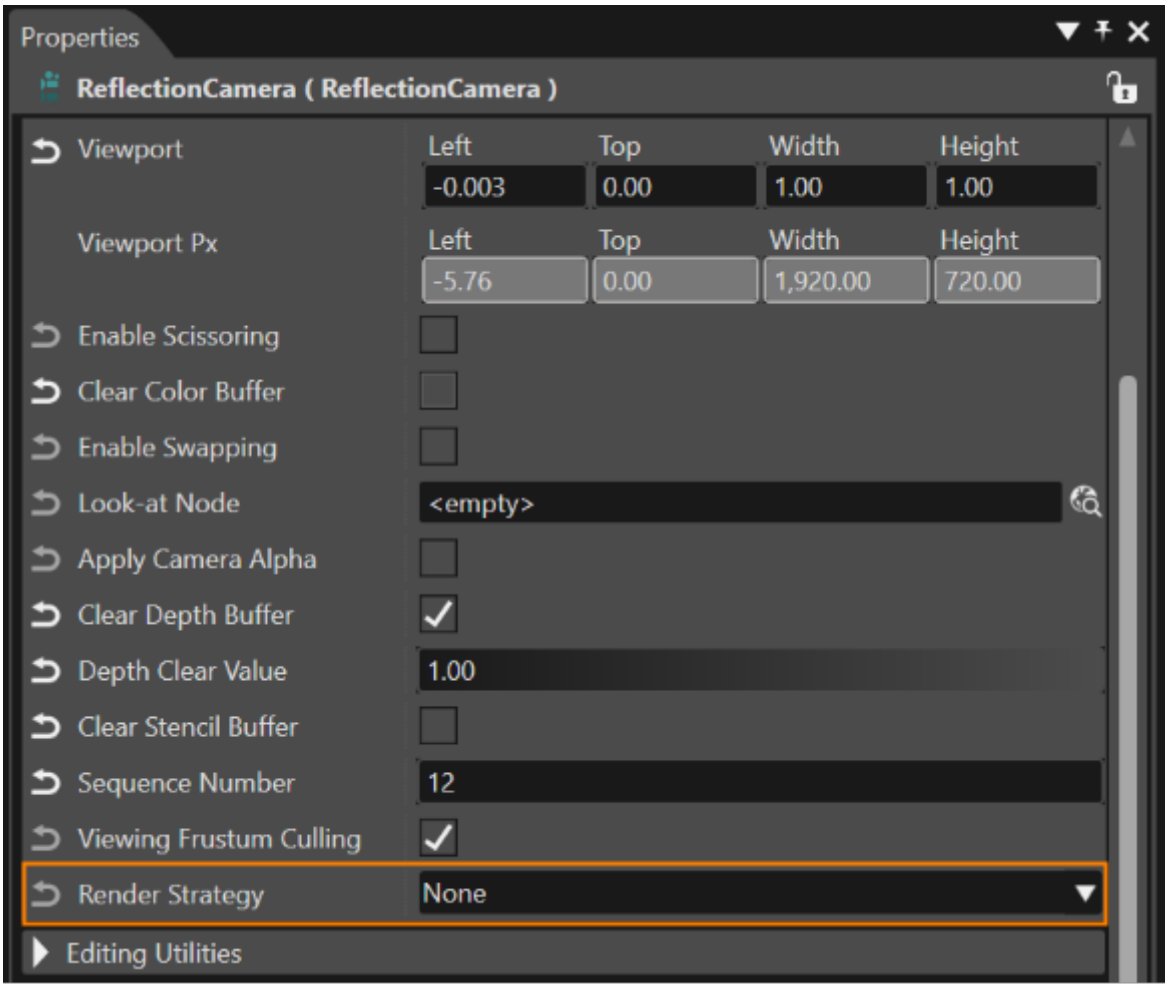
Configure the properties "Mirror Plane Direction" and "Mirror Plane Distance" to visualize the reflection plane.



The reflection will not be rendered in predefined views (Perspective, Orthogonal). Depending on the desired visual effect, set the proper values for the properties: IsColorClearEnabled and SequenceNumber.

Camera Render Strategy

Both Camera and Reflection camera objects have a property (available in Properties panel) called "CameraRenderStrategy" that only affects the processing of that camera and its nodes.

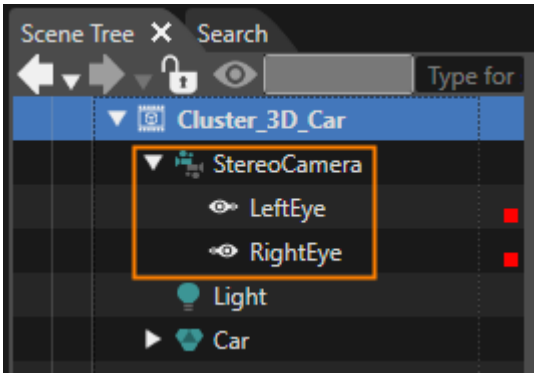


Per default "none" is selected as camera render strategy, which means the nodes will be rendered inside a render pass according to the defined render order.

Stereo Camera

To use the StereoCamera object two Camera objects have to be assigned as the left and the right eye. Only cameras that have no parent, or their parent is the StereoCamera itself, are allowed to be set as eyes. If they have no parent, they'll automatically get assigned as children of this StereoCamera. On these cameras position, lookAtVector, upVector and the projection get set automatically with respect to the StereoCamera objects setting.

A StereoCamera object is available in the Toolbox. In order to use it, the user should drag it on a scene. After this first step, a StereoCamera object will become available in the Scene Tree.



A LeftEye camera and a RightEye camera will be assigned to the StereoCamera. The LeftEye camera and the RightEye camera may be configured from the properties panel as any other camera. The properties specific for the StereoCamera may be configured from the Properties panel that is available when the StereoCamera is selected. Those are:

- Eye Separation and
- Convergence distance

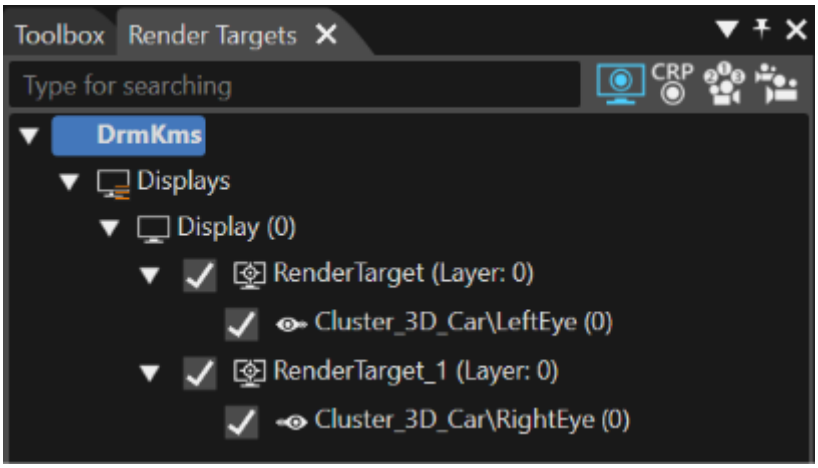
Eye separation controls the interaxial offset of the two Camera objects positions relative to the StereoCamera's position. Here for both camera's the position is set to half of the separation in +X and -X direction.

Convergence distance is the distance from the camera to the plane on which both projections would produce the same image, and therefore no parallax occurs.

> The stereo camera configuration is hardware dependent and it reflects the way in which the stereo effect is achieved in hardware. Without a special designed hardware it is not possible to properly display any stereo content. Stereo hardware most commonly provides the viewer True 3D visuals by interlacing or overlapping the two video outputs on the same display unit.

Without a special designed hardware the only possibility to check the Stereoscopy on SceneComposer is to link every Eye Camera to a different Render Target. For instance, it is possible to associate the LeftEye with a Left Render Target and the RightEye with a Right Render Target. The association of any Eye Camera should be done by following the usual two steps:

- Create a Display and two Render Targets. Read more...
- Associate every of the two cameras to one specific display (RighEye Camera to the R Render target and LeftEye Camera to the L Render target). Read more...



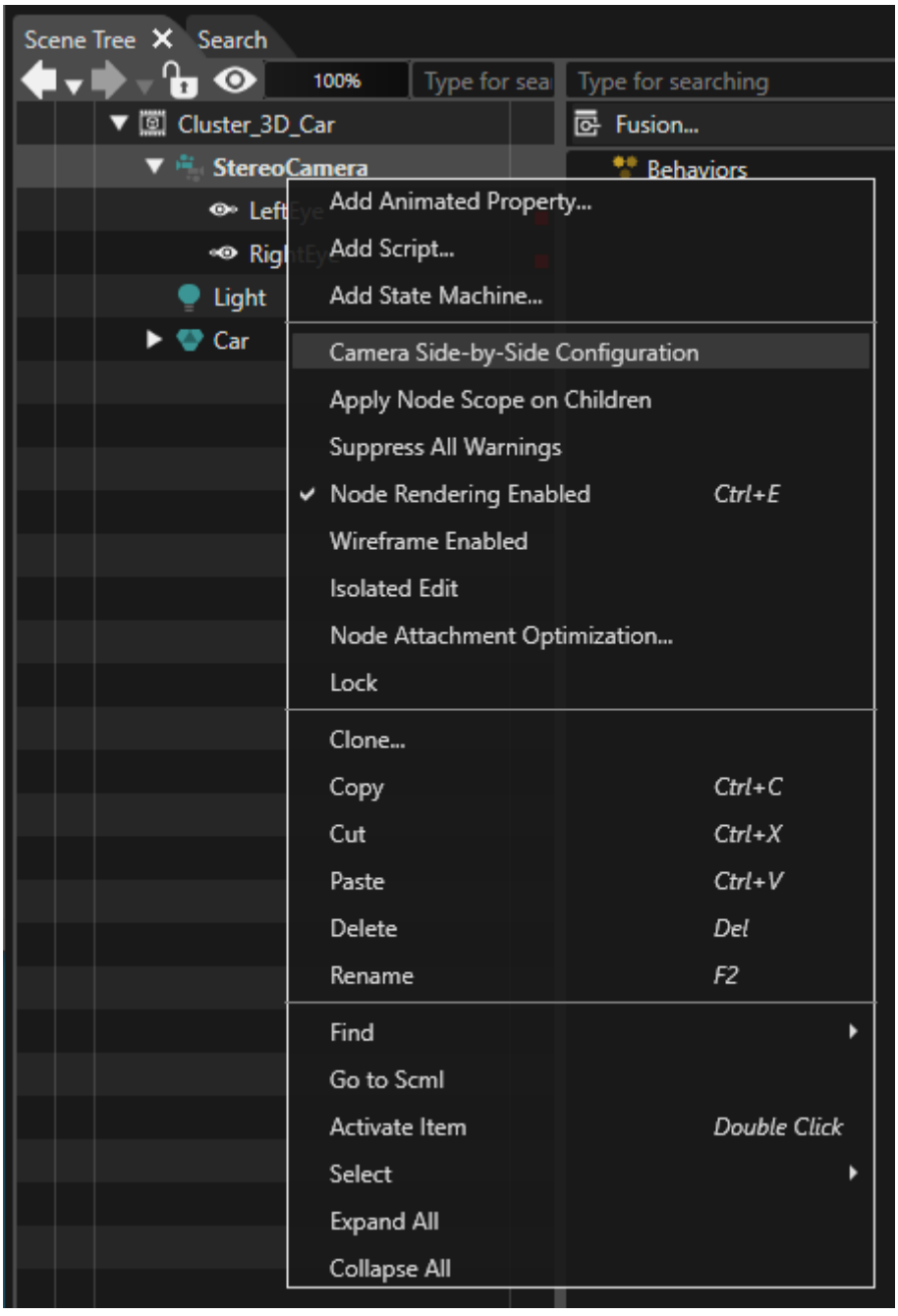
Both Render Targets are expected to be kept grouped under the same Display as it can be noticed in the image above.

Stereo Camera Side by Side

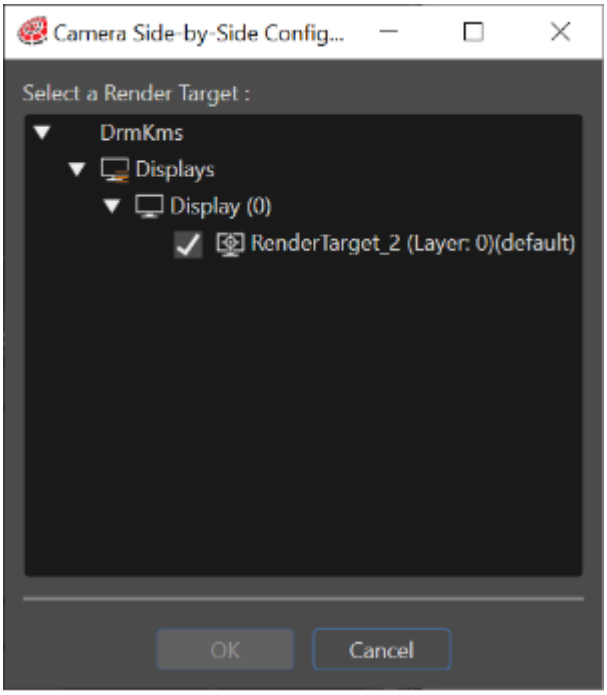
One of the Stereoscopic Camera configurations is the so called "side by side" (SBS) configuration. This type of configuration is based on display that combines the left and right eye images into a single image by squeezing the two together horizontally.

To facilitate the creation of SBS configuration, SceneComposer provides a built in configuration that can be used.

First drag and drop a Stereoscopic Camera from the Toolbox in a 3D scene. Second, right click on the Stereo Camera in the Scene Tree. Select "Camera Side-by-Side Configuration" from the context menu.



If a RenderTarget already includes cameras, you can not choose it. So, remove cameras in the RenderTarget and select "Camera Side-by-Side Configuration" menu to be appeared its dialog. After a display and a render target are created, select the Render Target. After this step click "OK".



In the Render Targets panel both "eyes" associated with the Stereo Camera will be displayed.

> The Viewport property is automatically set. Practically speaking, all the cameras are automatically configured. The dimension of the render target is the only setting that has to be configured manually.

Sky Box

One of the most spectacular types of environment mapping is the so called "sky box". This type of mapping gives the user the feeling of immersion into a large environment. The name is related with one of its first usages into pc games industry, especially in relationship with clouds and sky projected into a 3D level of a computer game.

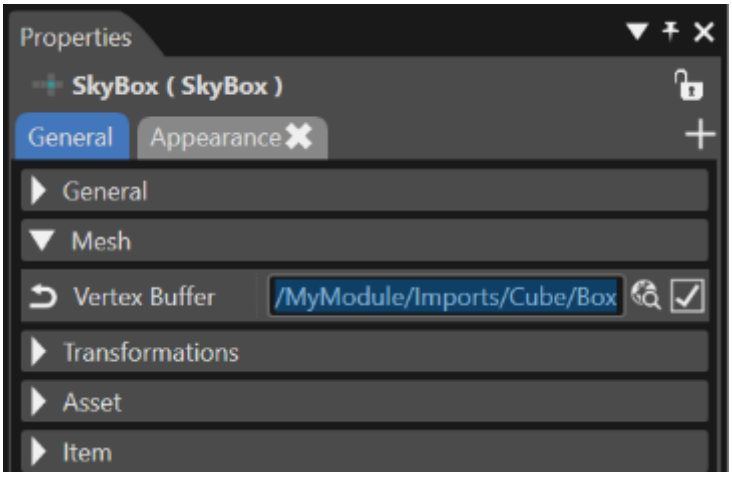
It is possible to create this type of virtual environment by using the SkyBoxMesh from toolbox. Associated with a camera, this mesh will be placed on every renderization on the same position as camera's position. When a skybox is attached to a camera, any of the clear color properties will not be taken into consideration.

The sky box should only be seen if the active camera is the same one with that camera to which the sky box is attached.

> The sky box cannot be manipulated by using gizmos and its bounding box cannot be displayed when it is selected.

Usually you need to specify only texture to show skybox (CubeMap Texture). And you do not need to specify VertexBuffer and Shader. But if you want to specify your own SkyBoxMesh:

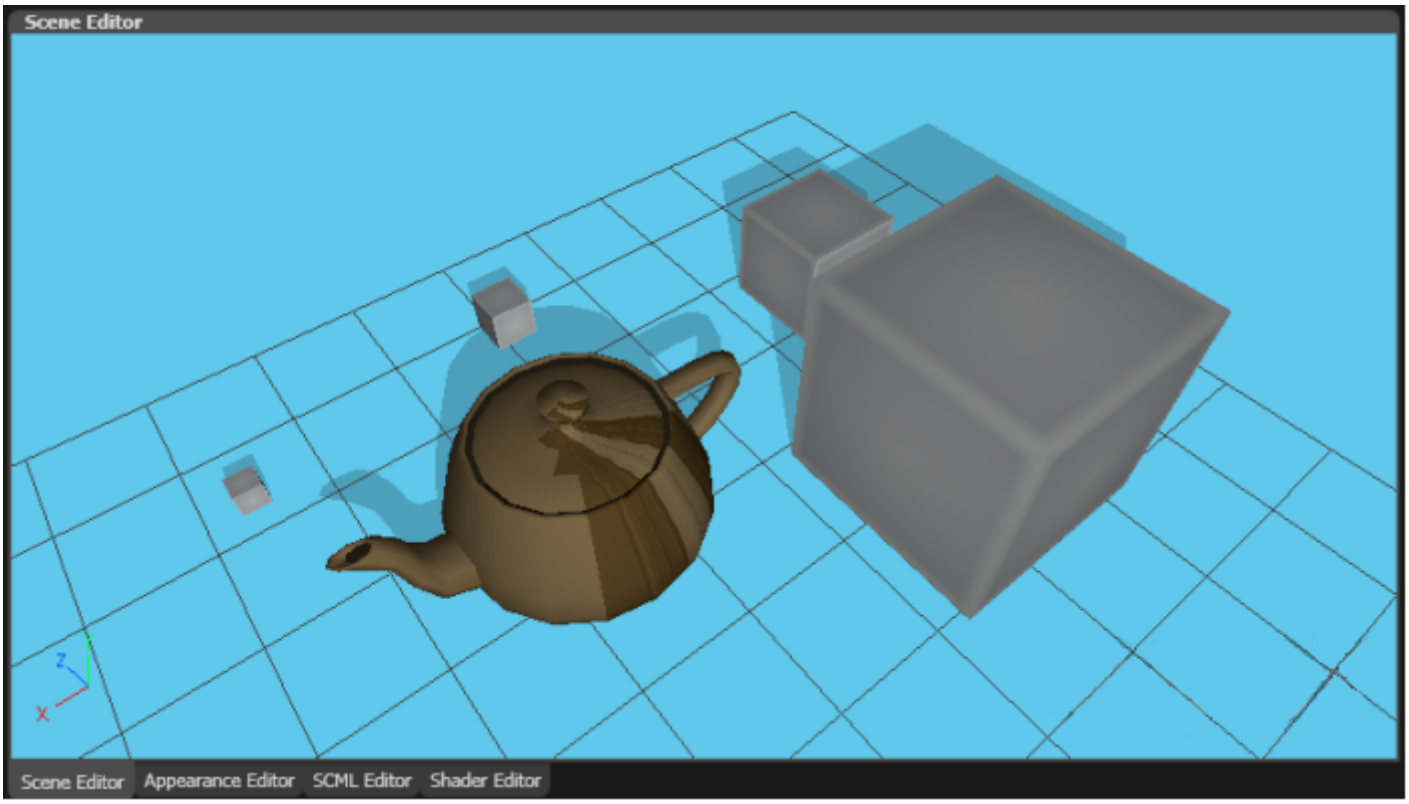
- If the Render Mode is used, Enable Depth Write and Enable Depth Test have to be set as "false";
- An adequate CubeMapTexture with six sides has to be set as texture;
- In Appearance the Shader Program that must be selected is RefTransCubeMap_RefCubeMapTex;



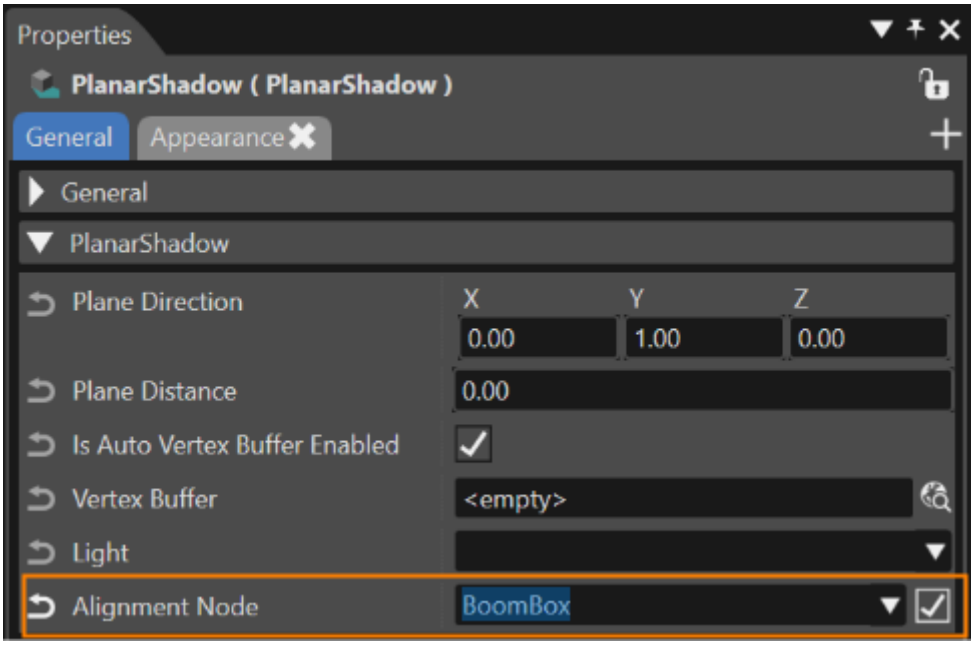
Planar Shadow

To configure it the following steps have to be taken:

- Set the light property (the source which will generate the shadow of the object)
- To set the display it is mandatory for the render target to contain a stencil buffer (stencil bits = 8)
- Shadow plane might be configured by using the PlaneDistance and PlaneDirection properties. Please, note that the shadow is on object by itself so its color (material) or render mode can be changed.
- The default shader is adequate for the renderization process.



The AlignmentNode property exposes Candera's automatic alignment behavior. When the alignment node is set, Candera automatically aligns the shadow plane to the alignment node (which usually is the shadow receiver).



Related with the planar shadow there are many other important aspects:

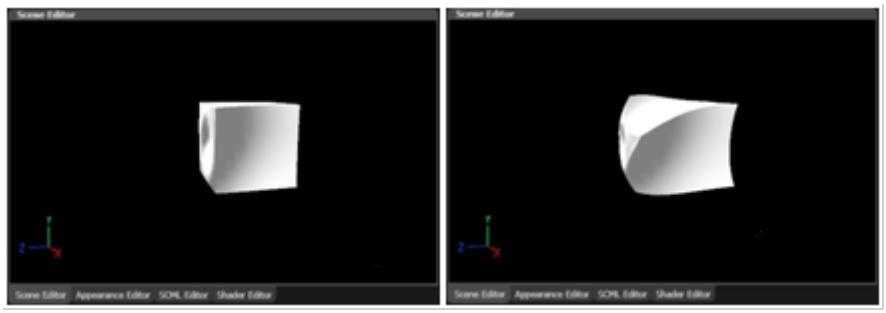
- The shadow cannot be edited by using gizmos (translate, rotate, scale);
- The shadow bounding box is the only selection indicator;
- The correct renderization is related with a stencil buffer;
- It is not possible to add children on shadows;

- The shadow is a mesh; so, a vertex buffer can be associated. If the user will explicitly associate a vertex buffer, the shadow can render with that vertex buffer just if the "IsAutoVertexBufferEnabled" property is deactivated.

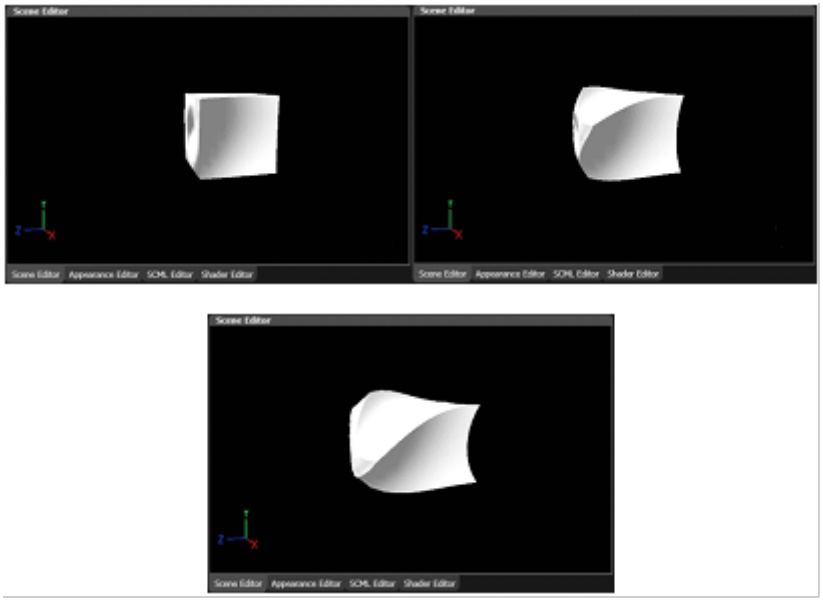
Morphing Mesh

Shape Morphing

Shape Morphing allows a smooth transition from an origin 3D model shape into a destination shape at runtime, e.g. to change a cube to a sphere. In contrast to model blending a single shape is rendered at every point in time which actually forms the morphed model. This model supports virtually all features of regular models.



CGI Studio supports not only a single shape being morphed into another but allows contributing multiple models to the resulting shape. The degree of contribution of a single model to the result is controlled by "morph weight" attribute of the regarding model.



Morphing of multiple models can be used for various fancy effects such as 'chaotic' vertex movement, explosion effects, and simple particle effects.

Technical Background

Geometries (vertex buffer) of different shapes supposed to be morphed are combined into a special, single vertex buffer for "Morphing Mesh" nodes including all vertex buffer attributes such as position, normals, texture coordinates per original vertex. This combined vertex buffer is uploaded to GPU, the regarding vertex attributes can be accessed in vertex shader. Uniforms applied to vertex shader during rendering by Candera include morphing weights for all contributing shapes. The vertex shader calculates the vertex attributes of the resulting shape by adding all regarding vertex attributes of single shapes multiplied by morphing weight.

For example, the vertex position is calculated in vertex shader as

Vertex Position = Vertex Position 0 * Morph Weight 0 + Vertex Position 1 * Morph Weight 1;

Other vertex attributes that are subject to morphing calculation are normals, bi-normals, and in special cases, texture coordinates. This calculation moves the vertices between the corner points of the model according to their morph weight.

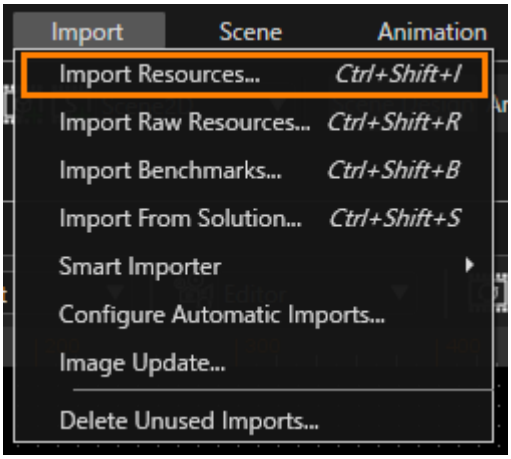
If color or textures are subject of morphing as well, the resulting color needs to be calculated in an according fragment shader considering morph weights.

Prerequisites

3D models need to be prepared for shape morphing. For each "corner" shape, a single model needs to be provided. All models contributing to a morphed shape need to have homologous vertices. This means that all models must have the same number of vertices and the vertices must "semantically correlate". For instance a vertex of model 1 addressed by a certain index is located in a front face. The vertex of model 2 with the same index need to be located on the corresponding front face of the second model, and not, e.g. on a back-side face.

Create a MorphingMesh

To create a MorphingMesh at least one .fbx file has to be imported. Choose "Import > Import Resources..." from the menu. This will open the "Import Resources" dialog. Browse to your FBX file an click [Open] to import the file.



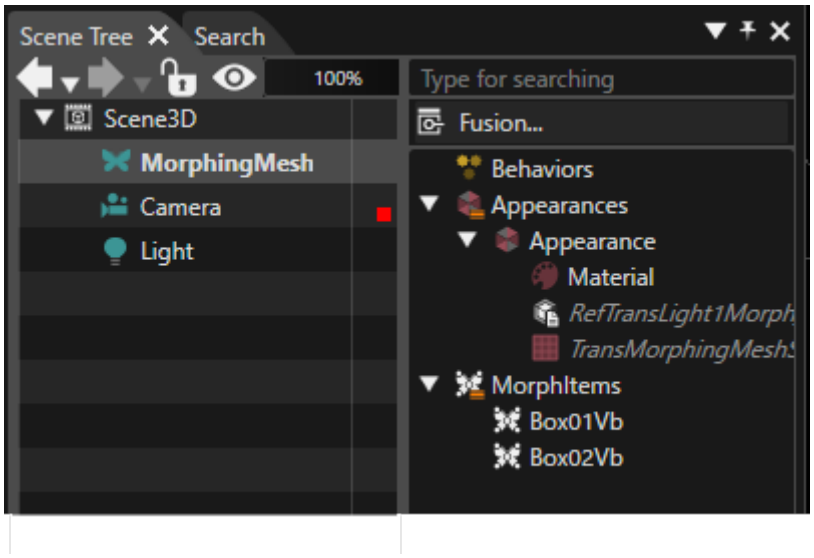
To import the FBX file in Import dialog click "Select Files..." button.

> It is possible to import FBX files in SceneComposer by dragging them on "Solution Explorer" panel.

Configuring shape morphing in SceneComposer:

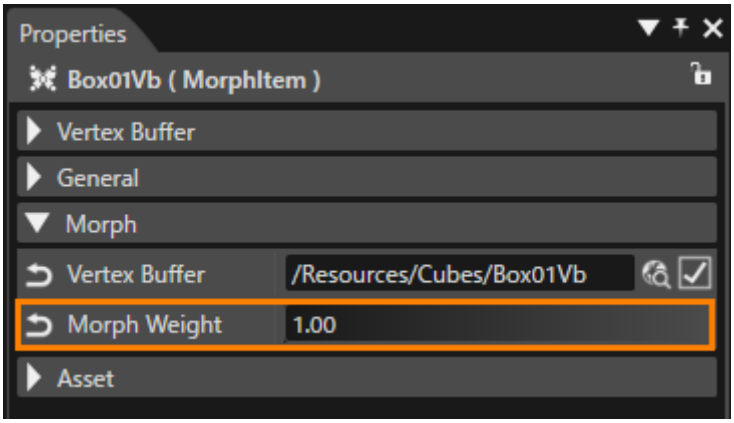
- Create a "MorphingMesh" in a scene by drag and drop from the 3D Toolbox.
- Drag and drop all vertex buffers having the same amount of vertices, which shall be used for morphing to the MorphingMesh.

> The vertex buffers can be dragged just from the Solution Explorer to the MorphingMesh



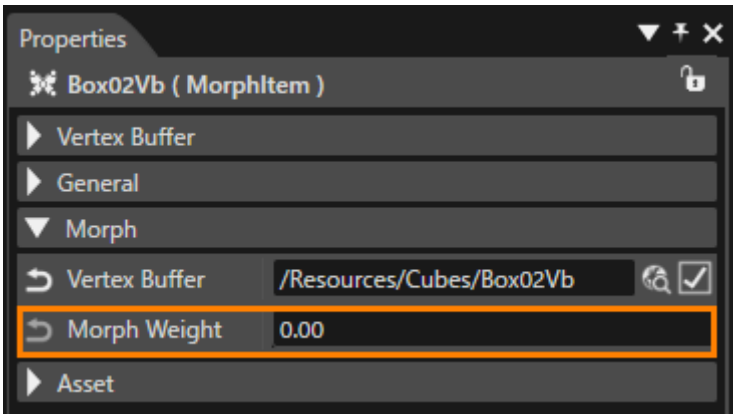
Configure Morphing Weights

In order to get the desired morphing effect, select the morphing mesh in the right side of the Scene Tree panel and set the weight of each morphing mesh item in the Properties panel accordingly.



> The sum of the weight of all morphing mesh items part of a MorphingMesh must be 1.00 to get correct morph behavior.

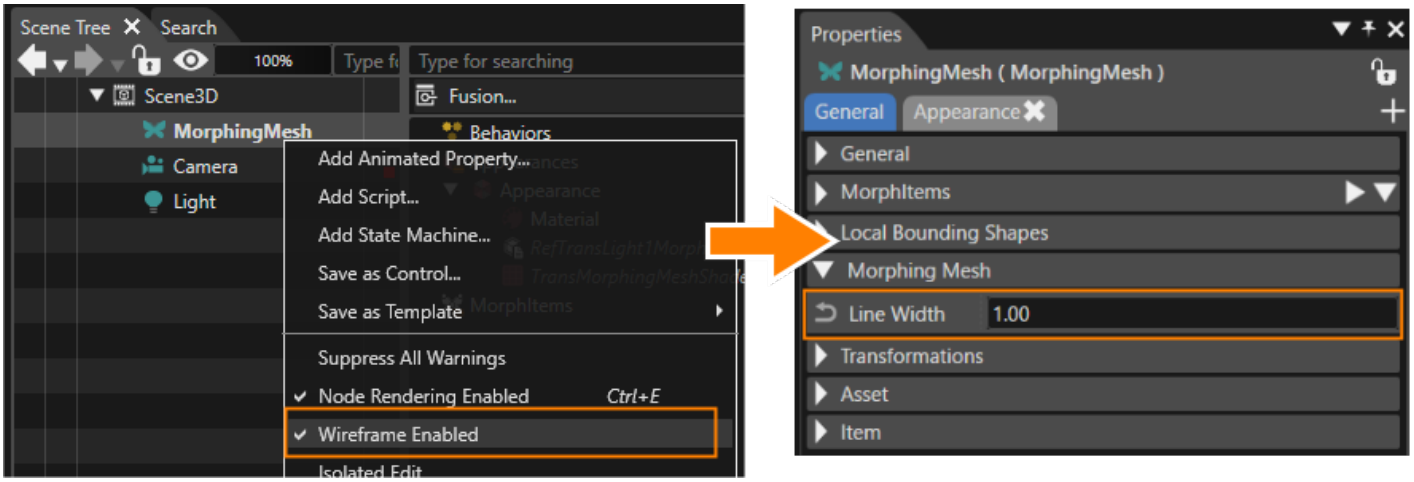
By default, after creating morphing mesh items, the morph weight of the first item is set to 1.00, while morph weights for all other items are set to 0.00.



Wireframe Line Width

When a wireframe vertex buffer is applied to a morphing mesh and the “Wireframe Enabled” option is enabled from the context menu in the scene tree, the properties item “Line Width” is added to the Morphing Mesh. This specifies the line width of the Morphing mesh when it is rendered as wire.

- The actual line width is determined by rounding the specified width to the nearest integer.
- If the setting is less than “1”, it is always set to “1”.



Morphing Shader

The reference shaders for morphing:

- RefTransLight1Morph_RefColor and
- RefTransLight1Morph_RefColorTex

support morphing between maximal 2 vertex buffers.

Level of Detail (LOD)

With Level of Detail, 3D render performance can be improved by using low polygon models for far away objects and high polygon models only for objects near to the camera.

Create a LodNode

Using the level of details concept in SceneComposer:

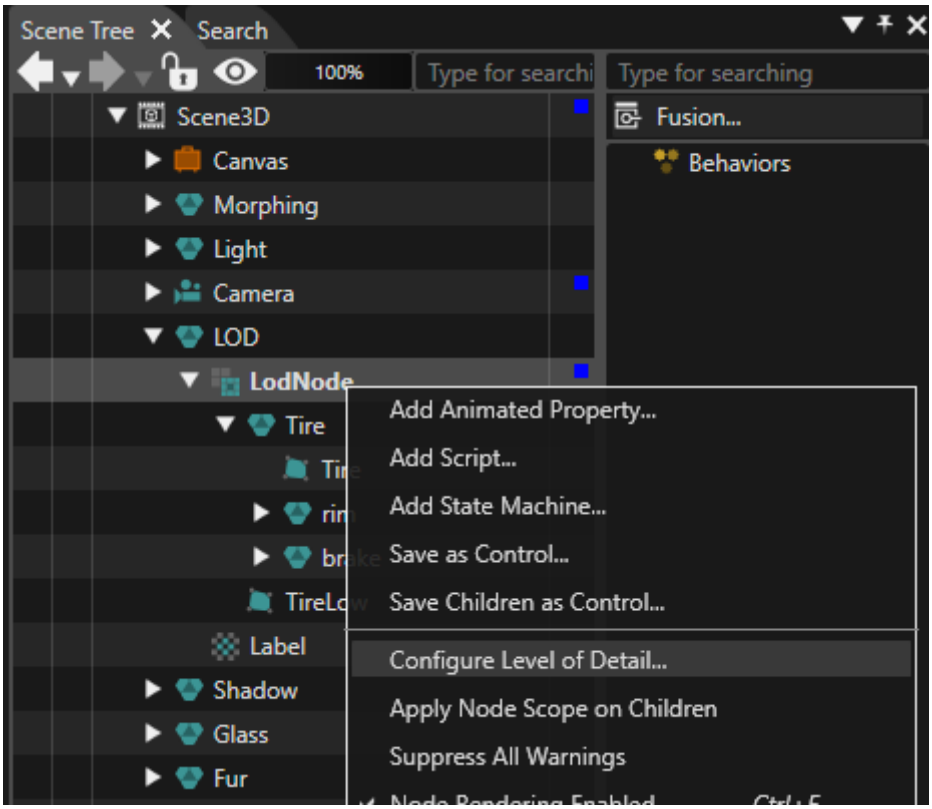
1. Create a lod node item called "LodNode" in a scene by dragging it from the Toolbox and dropping it to the scene.
2. Add children (for example: 2 Billboards) to the LodNode, providing models for each desired lod level. Note that each child node needs to have an own RenderMode attached.

Configure Lod Levels

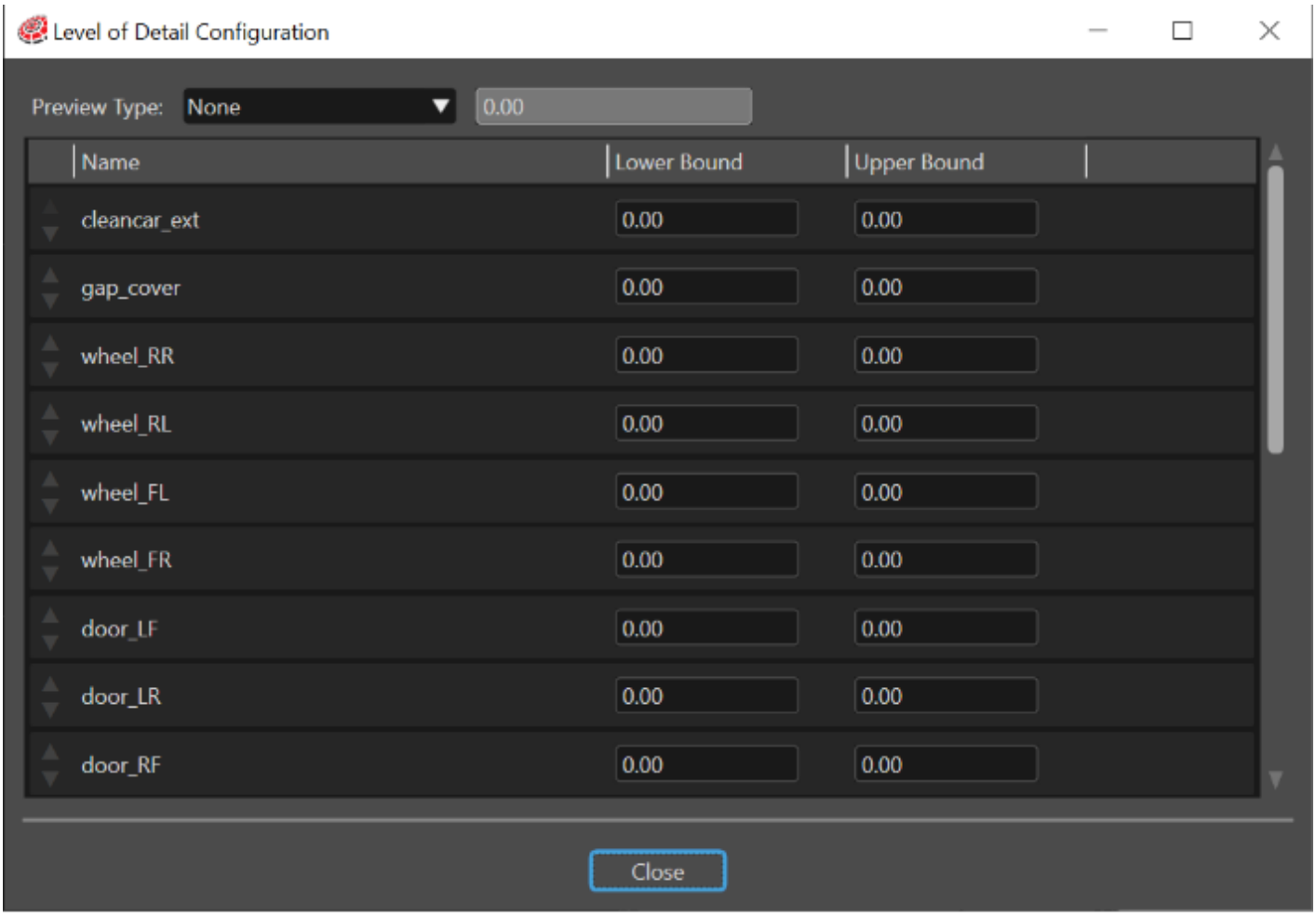
Each child of a LodNode must be associated to a specific LodLevel by configuring following properties:

- index,
- lower bound of visibility range and
- upper bound of visibility range.

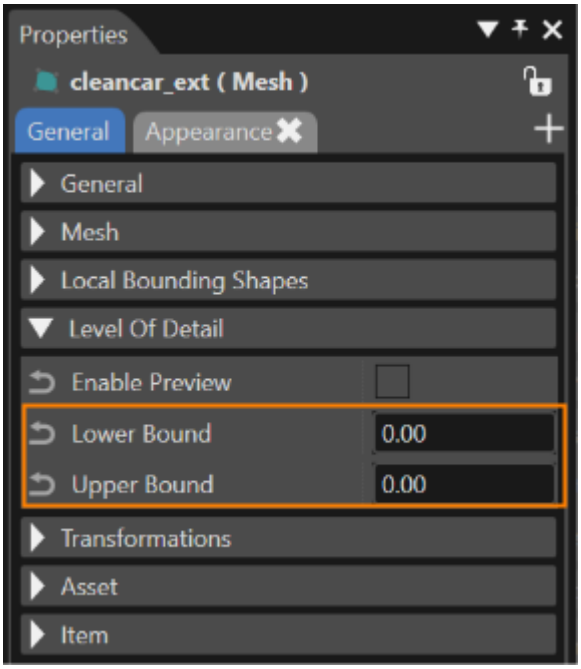
While the index is given by the order of the LodNode children in the scene graph, lower and upper bound can be configured either in the Properties panel or in the "Configure Level of Detail" dialog. For this, use the context menu item "Configure Level of Detail" on the selected LodNode in Scene Tree panel:



First of all, the "Level of Detail" dialog allows to specify the Lower Bound and the Upper Bound for any child part of the LodNode.



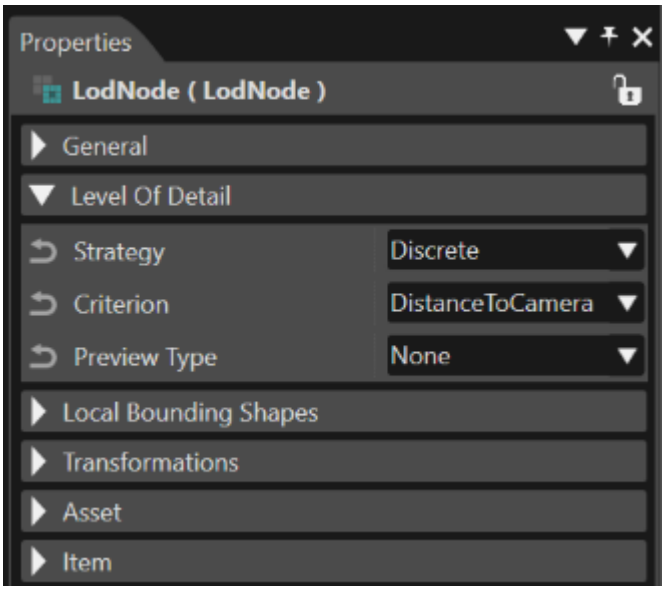
The same operation can be done in the Properties panel, whenever a LodNod child is selected in the Scene Tree panel:



Configure LodNode

Select the LodNode in the Scene Tree panel to edit the following LoD node properties in the Properties panel:

- LoD render strategy,
- LoD criterion and the
- preview type



The preview type property can be edited in the Level of Detail Configuration dialog, too.

Discrete LoD Render Strategy

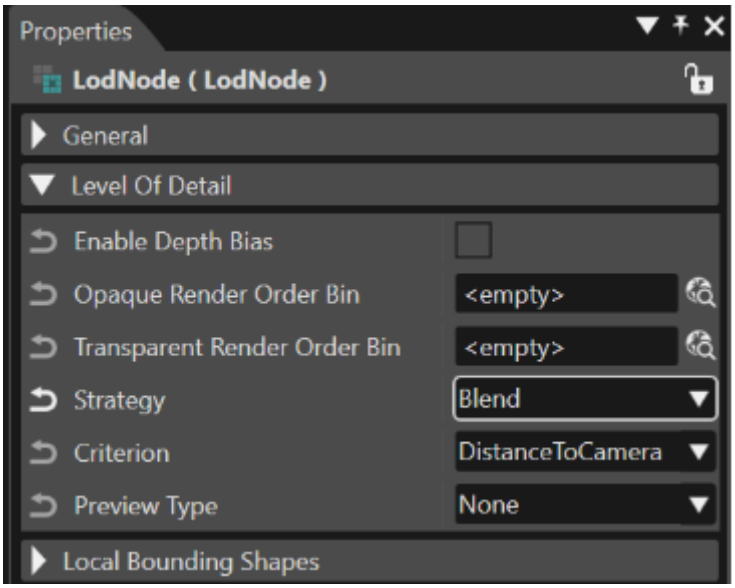
Select "Discrete" for the Level of Detail render strategy to configure a stepwise transition between the several Lod levels.

After choosing "DistanceToCamera" Lod criterion, the LoD level will be switched depending on the distance of the LodNode to the camera.

Blend Lod Render Strategy

Select "Blend" for the Level of Detail render strategy to configure a smooth alpha blending transition between the several Lod levels. In this case, following additional properties can be specified:

- Enable Depth Bias
- Opaque and Transparent Render Order Bin



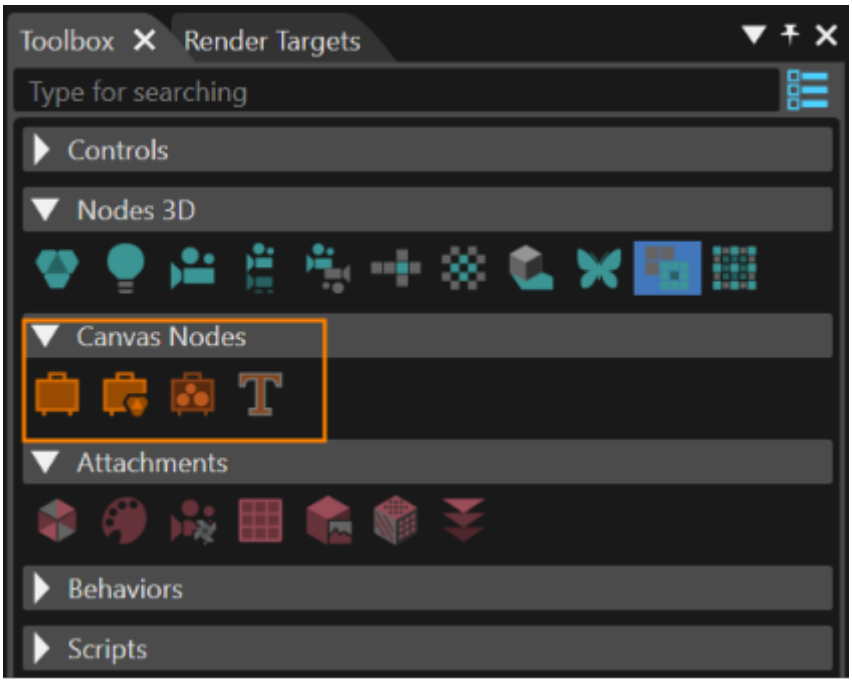
Again, after choosing "DistanceToCamera" Lod criterion, the LoD level will be switched depending on the distance of the LoDNode to the camera.

Canvas

Canvas is a node that acts as a surface which allows the user to introduce 2D content in a 3D scene. Canvas nodes can have appearances, just like normal 3D nodes so the user can achieve complex visual effects by using appropriate shaders, materials and textures.

Canvas, CanvasGroup, CanvasSprite and Canvas Text (placed under the "Canvas Nodes" section of the Toolbox) behave like normal nodes when it comes to transformations and selection. Bounding box will be rendered for each of them. They can be translated, rotated, scaled. Canvas acts as a surface. CanvasGroup nodes, CanvasSprite nodes and Canvas Text nodes can be inserted into it.

For each Canvas object, a new editing camera is added in the scene camera list and becomes available in "Canvas" category. This camera can be selected and exhibits a special behavior: is automatically oriented towards the canvas, regardless of canvas transformations, so that the Canvas and its children will always appear like they would be viewed from a "Front" camera. A Canvas camera provides an orthographic projection.



Canvas Sprite

CanvasSprite is a node that represents a solid object, offering the possibility to render a texture, a flat colored rectangle or other visual effects, depending on its appearance configuration.

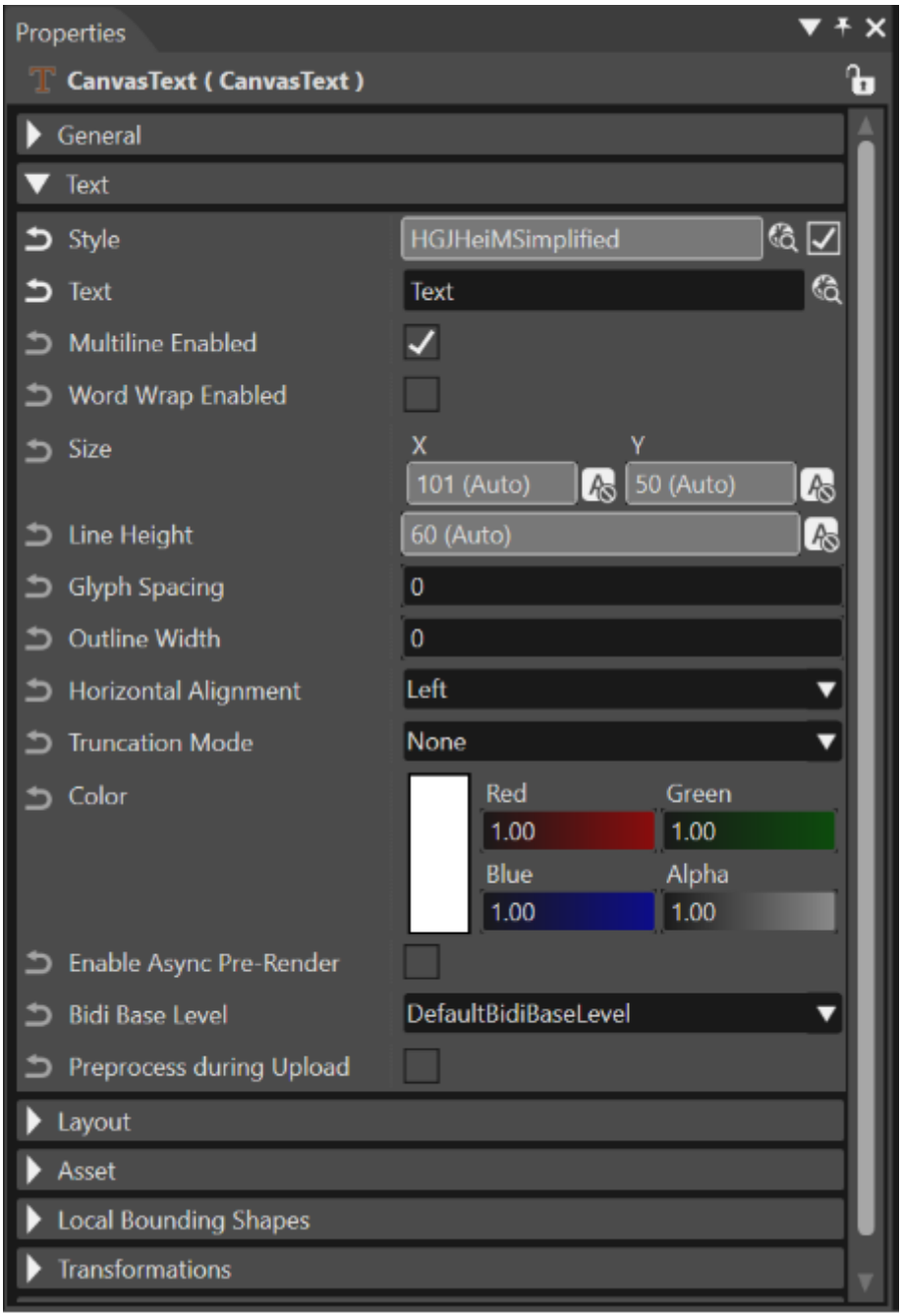
Four new shaders will be available in the default solution for configuring the CanvasSprite:

- RefCanvasTrans_RefTex
- RefCanvasTransLight1_RefColor
- RefCanvasTransLight1_RefColorTex
- RefCanvasTransUniDiffuseMat_RefCanvasUniDiffuseMat

A new uniform setter template will be available in the default solution for configuring CanvasSprite: CanvasTransShaderParamSetter.

Canvas Text

Canvas Text is a node that allows the creation of 2D texts on a Canvas in a 3D scene. When Canvas Text node is selected, specific properties become available in the Properties panel.



In this way the text can be set as desired.

> The Canvas automatically batches Canvas Texts together to minimize the amount of draw calls. To achieve high batching rates, it is crucial that the user shares Appearances, Render Order Ranks, Render Order Bins, and Scope Masks between Canvas Texts as much as possible. Only Canvas Texts with the same properties can be batched together.

Some of the most important Canvas Text properties are the following:

Alignment: The text has a property for horizontal alignment. This property aligns the text within Text::Size. If no Text::Size is set, the Layout::Size is used. The layout has a horizontal and vertical layout alignment property which is used to align the ‘block’ of the measured text within the Layout::Size. The defined layout alignment can be propagated to the text alignment when the text

alignment is set to ‘Auto’.
Otherwise, it is possible to right align the text within Text::Size but left align this block of right aligned text within Layout::Size. The properties will be inverted if the layout direction is set to ‘right to left’.

Word wrap and truncation: Both properties prepare a text for use cases in which they do not fit into a size. Therefore, enabling one of them forces the text to fit into the provided smallest size (Layout::Size or Text::Size).

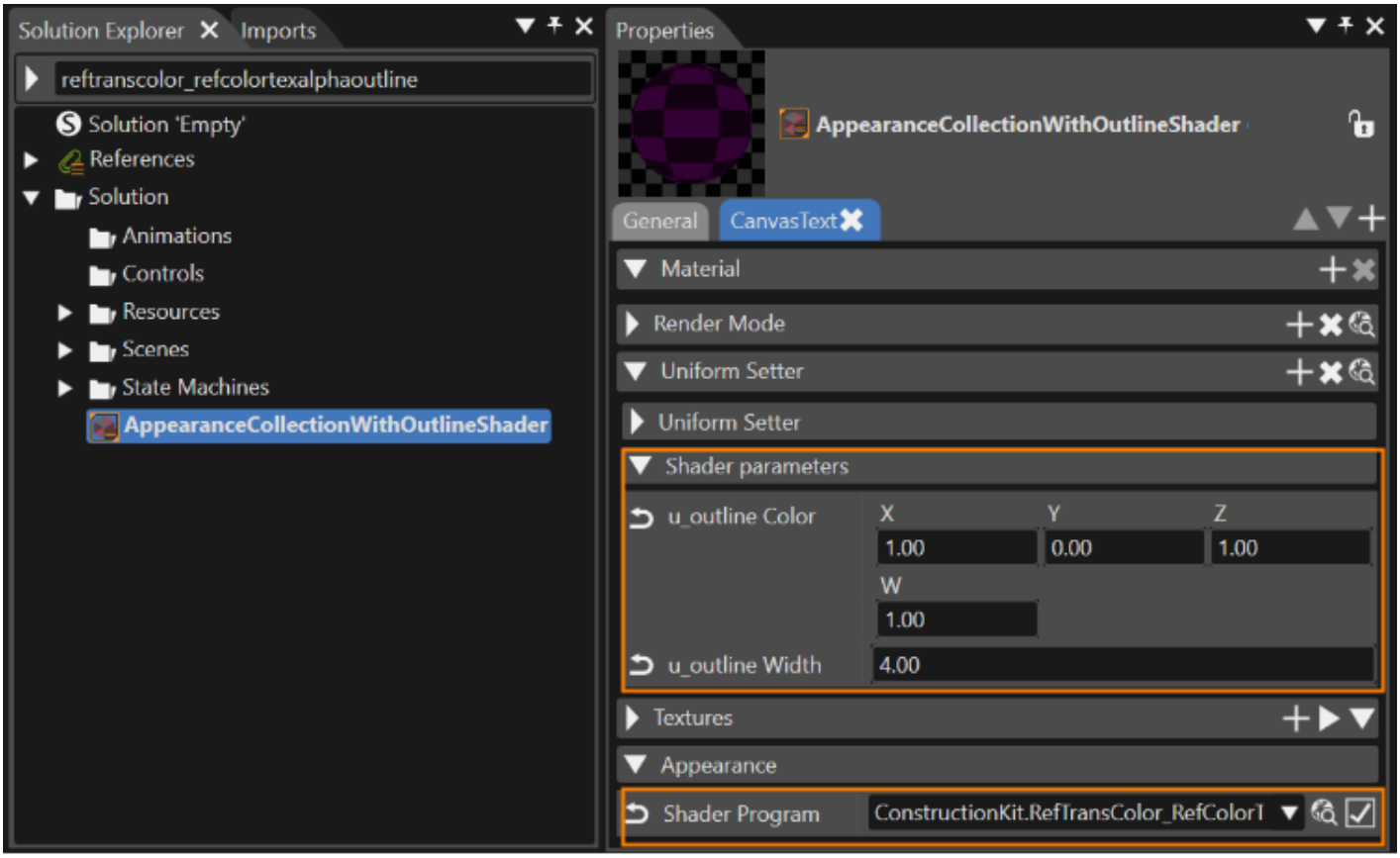
For many other details see the chapter Canvas Text - Size Property.

Canvas Text Outline

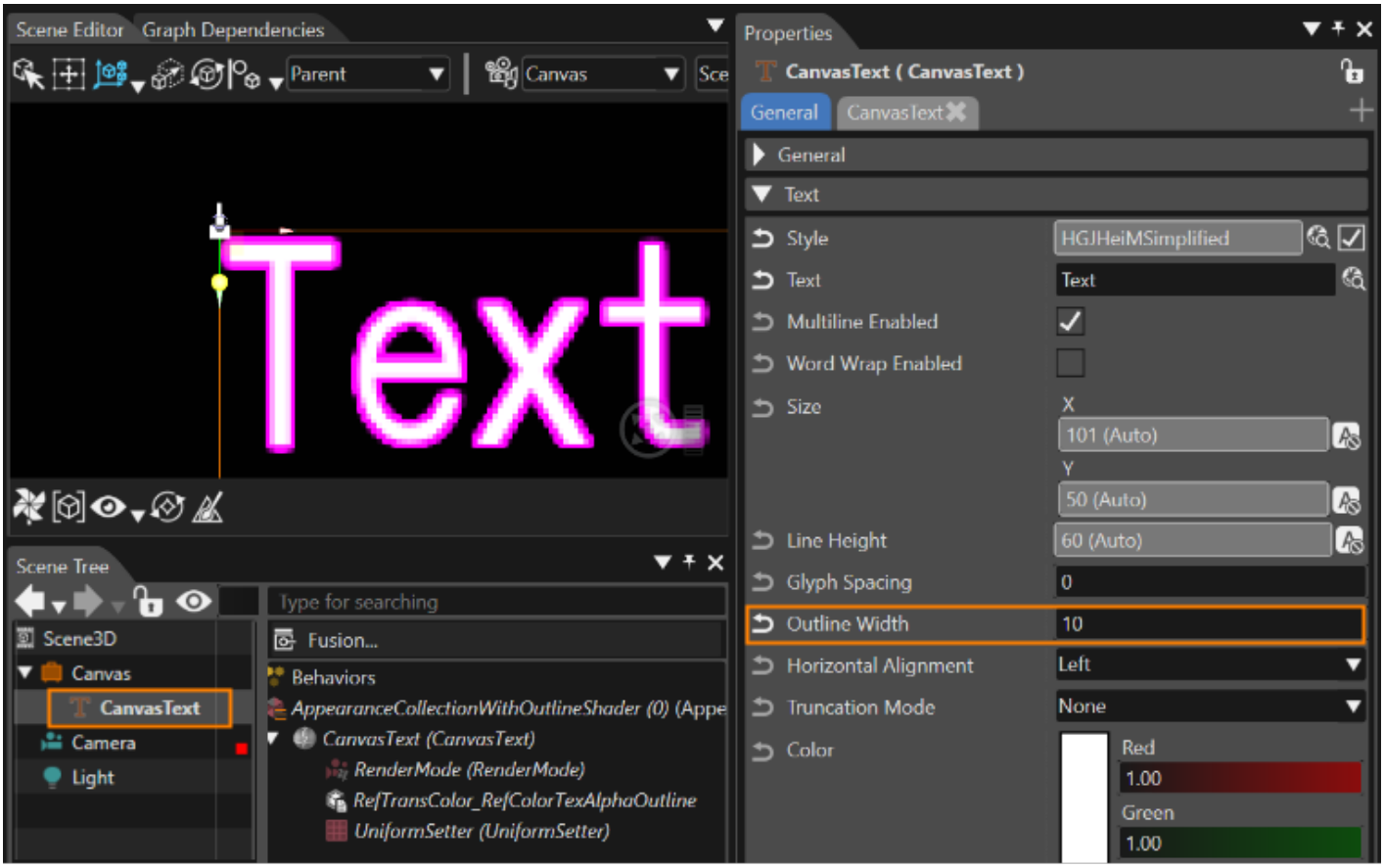
By using this feature, it is possible to use shader effects such as Outline on any canvas text node on a 3D scene. This property allows the user to add an outline on any canvas text on a 3D scene.

To use the "Canvas Text Outline" feature, it is mandatory to use an Appearance Collection that has set "RefTransColor_RefColorTexAlpha" as shader.

In order to obtain such a collection, first, in "Solution Explorer" copy the "CanvasText" appearance collection which can be found in "References > SCL:ConstructionKit > Resources > Appearances > CanvasText" from "Templates" into your solution. After this step, in "Solution Explorer" rename the copied "CanvasText" appearance collection (e.g.: "AppearanceCollectionWithOutlineShader") then in Properties select the "CanvasText" tab and under the "Appearance" section set "RefTransColor_RefColorTexAlphaOutline" as shader program.



From now on it is possible to use this appearance collection to set an outlined canvas text. The color and width properties are available under "Shader parameters" section (see the image above). To see how this works, first, when "AppearanceCollectionWithOutlineShader" is selected in "Solution Explorer", modify the u_outline Color as follows: X=1.00/Y=0.00/Z=1.00/W=1.00 then set the u_outline Width to be "4". Create a 3D scene and in it drag and drop from Toolbox a Canvas node with a Canvas Text in it. In Scene Tree select the "CanvasText" node and drag and drop on it - from "Solution Explorer" - the "AppearanceCollectionWithOutlineShader". In Scene Tree select again the "CanvasText" node then in Properties modify the "Outline Width" to be at least 1 or 2.



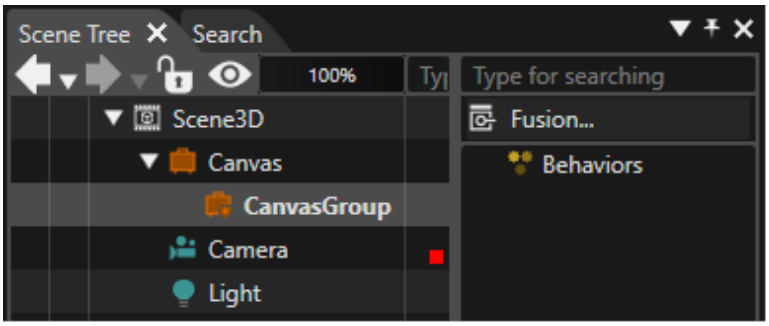
In this way the outline of the text can be set as desired.

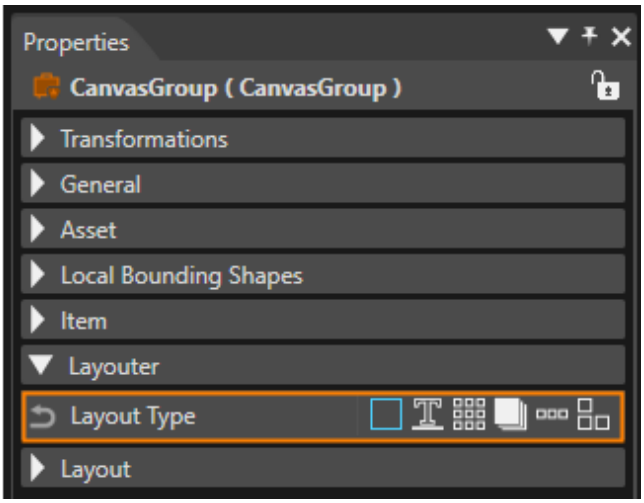
Canvas Layouter

The functionality of 3D Layout for canvas nodes is similar with that of the layouters known from SceneComposer 2D scenes. By using this feature a Canvas Group is able to align its nodes (group, sprite, or text) according to the configured layouter.

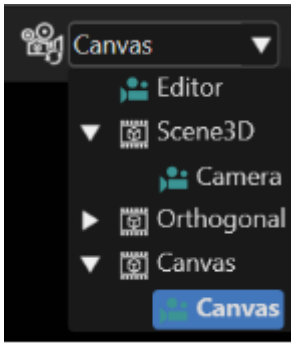
It's not possible to animate the same property for parent and child nodes. Group nodes are resolved and all properties are applied on their child nodes. Either animate the parent group property or the child properties.

The settings for this feature can be done through a category with the same name available in the Properties panel. To access it, first, a Canvas Group - included in a Canvas - has to be selected:





> In order to get the best perspective on a scene where canvas groups/nodes are used, a special canvas camera is available (see the image below).



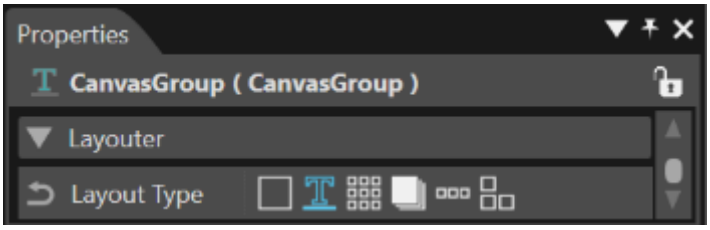
By using this feature a Canvas Group is able to align its nodes - group, sprite, or text - according to the configured layouter. The available layout types are identical with those used for the 2D layouters:

- BaseLine Layouter
- Grid Layouter
- Overlay Layouter
- Stack Layouter
- DockPanel Layouter

> The "Ctrl+L" command is a shortcut for the "Do Layout" operation. Please, note, that in the case of 3D Canvas layouter, this command will update the position and scale' values for the selected object.

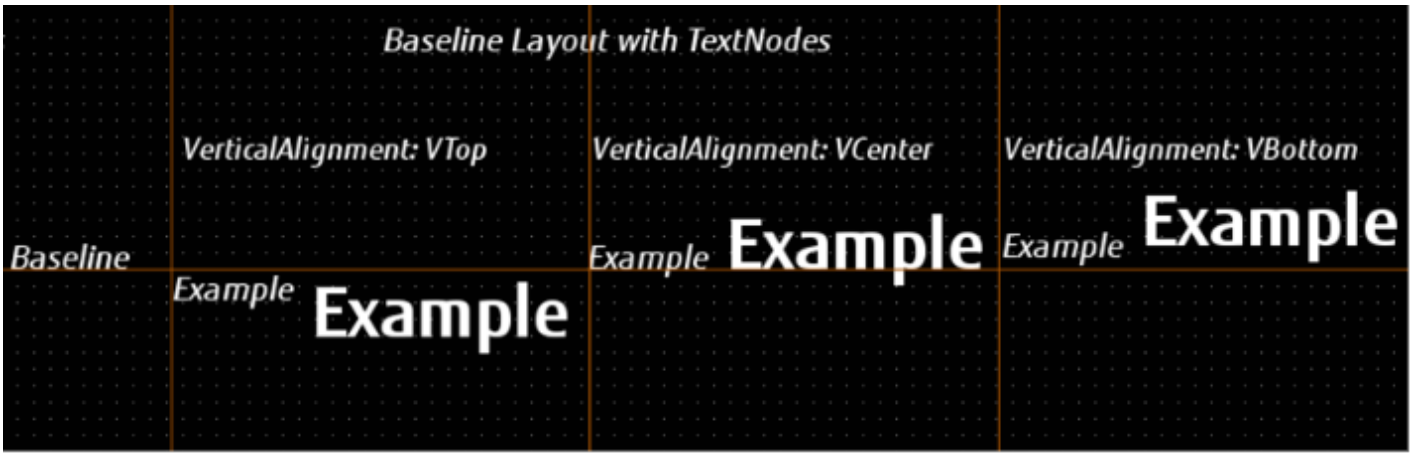
Baseline Layouter

The layout type "Baseline" realizes a stacked laying out in the horizontal direction. It behaves similar to the Horizontal Stack Layouter. The major difference consists in its given reference: it aligns the objects to a line instead of aligning them to the client area. The main purpose of this layouter is to align different sizes formatted texts along the same baseline. For instance, if the center is chosen for alignment, the baseline of the text will be aligned to the given line (= baseline offset). The order of the elements is given by the order of the child nodes within the layouted group.



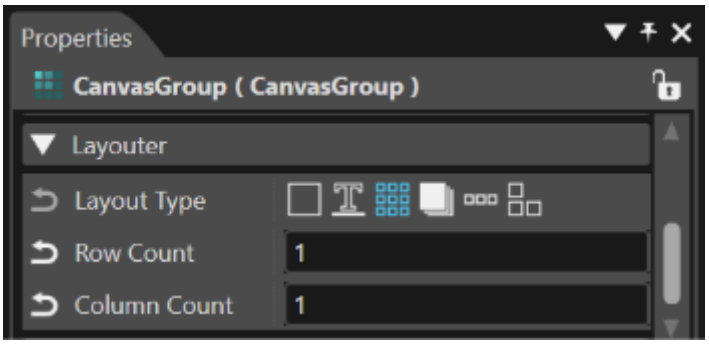
Configure the Baseline Layouter by following these steps:

1. Select the Canvas Group holding the nodes to be layouted in the Scene Tree panel
2. Choose LayoutType "Baseline" from the Properties panel
3. Set any value for property Base Line Offset in the Properties panel, depending on the desired laying out



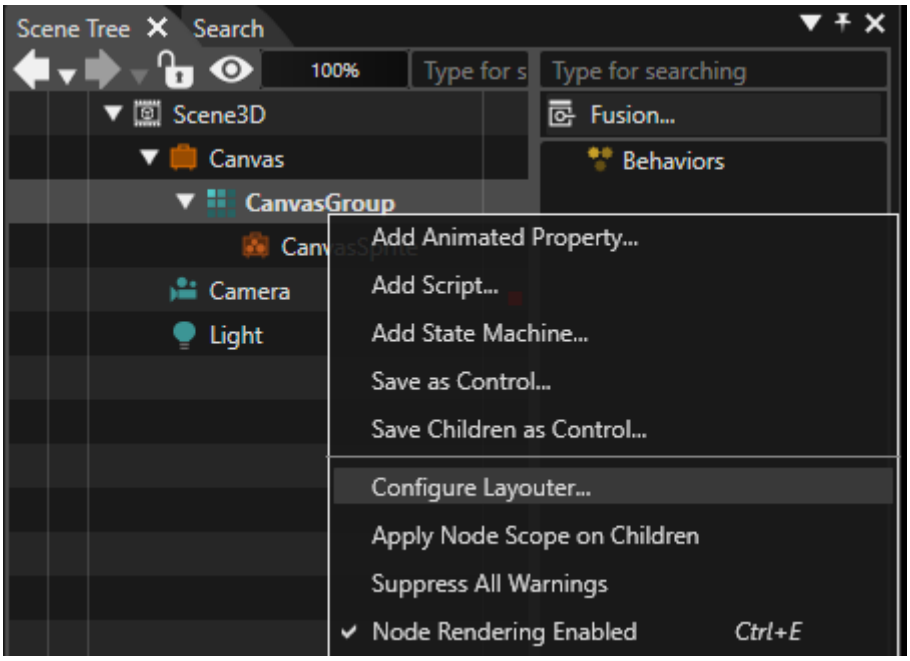
Grid Layouter

In a Grid layout, every node part of the layouted canvas group is assigned to a cell of the grid. A Grid Layout Editor is available to configure the grid and to arrange nodes in the grid's cells by drag & drop operation.

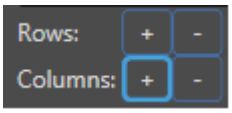


Configuring the Grid Layouter:

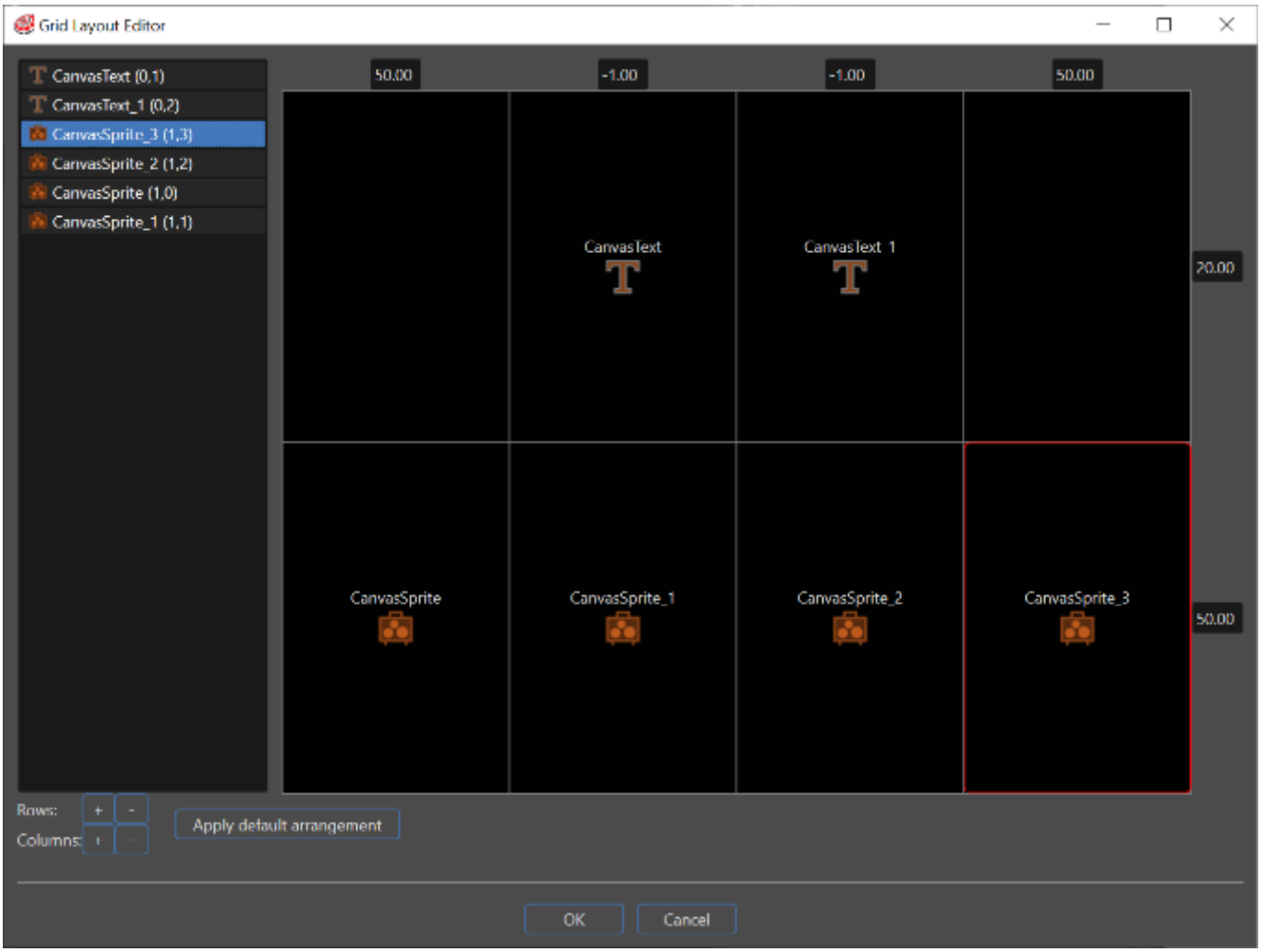
1. Select the Canvas Group holding the canvas nodes to be layouted in the Scene Tree panel
2. Choose LayoutType "Grid" from the Properties panel
3. Optional: Set the number of rows and columns of the grid by editing the properties RowCount and ColumnCount in the Properties panel
4. Use the context menu item "Configure Layouter" on the selected Group in the Scene Tree panel to open the "Grid Layout Editor" dialog



Optional: Use the buttons "+","-" to configure the desired number of rows and columns in the "Grid layout editor" dialog, if not done already in the Properties panel.



Arrange the nodes of the group in the grid's cells by drag & drop operation.



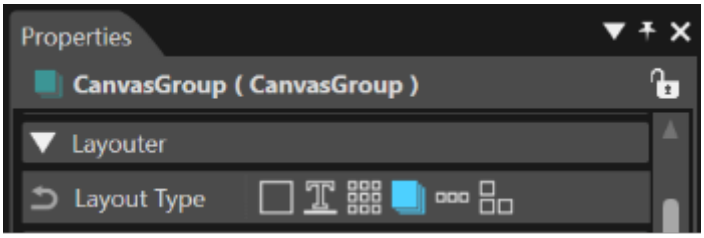
Configure width and height for rows and columns by editing the values on the border of the grid. Values within the range [0...1] are considered as percentages and values greater than 1 as dimensions.

Confirm the layouter configuration by pressing the "OK" button of the dialog "Grid layout editor".

Call the Do-Layout command using the shortcut keys "Ctrl+L" or from the "Edit" > "Do-Layout" menu.

Overlay Layouter

The layout type "Overlay" arranges the objects by laying out them on top of each other or by aligning the nodes according to their general layout properties. The order of the elements is given by the order of the child nodes within the layouted group.

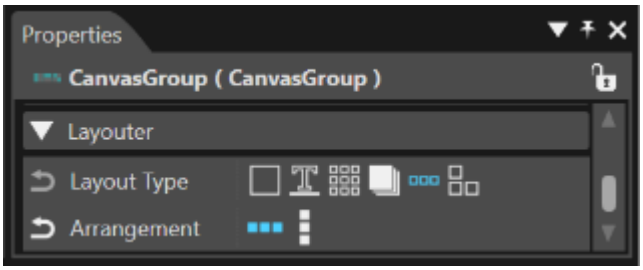


Configuring the Overlay Layouter:

1. Select the Canvas Group holding the canvas nodes to be layouted in the Scene Tree panel
2. Choose LayoutType "Overlay" from the Properties panel

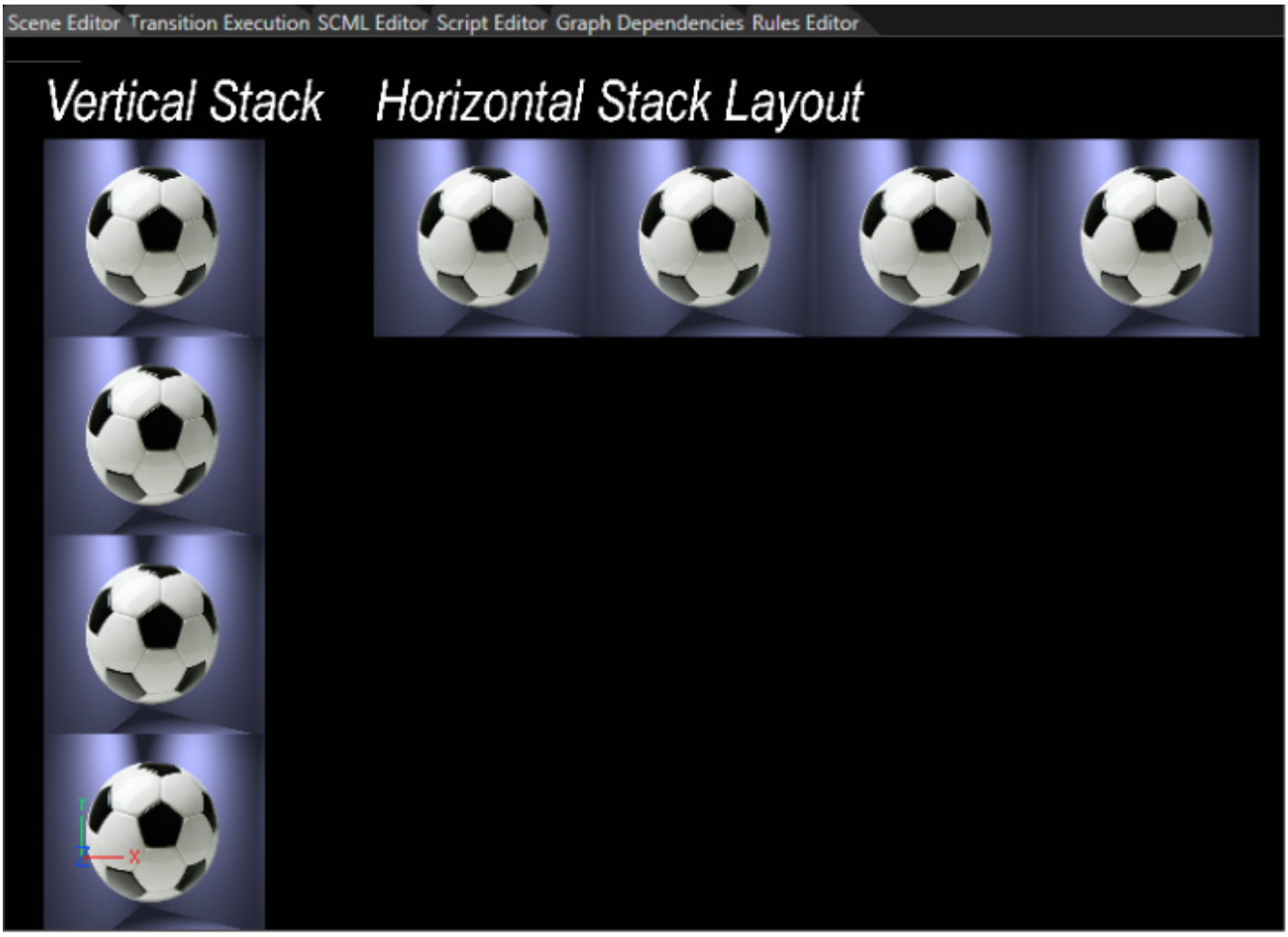
Stack Layouter

The layout type "Stack" realizes a stacked laying out either in horizontal or vertical direction. The order of the stack elements is given by the order of the child nodes within the layouted group.



Configuring the Stack Layouter:

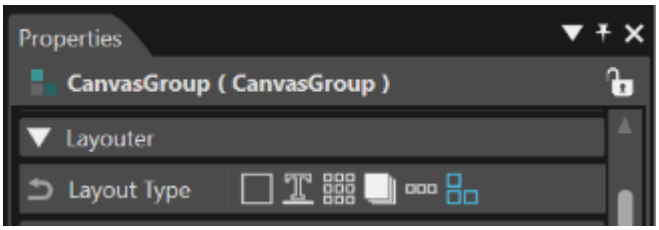
1. Select the Group holding the nodes to be layouted in the Scene Tree panel
2. Choose LayoutType "Stack" from the Properties panel
3. Choose "Horizontal" or "Vertical" value for property Arrangement in the Properties panel, depending on the desired laying out



DockPanel Layouter

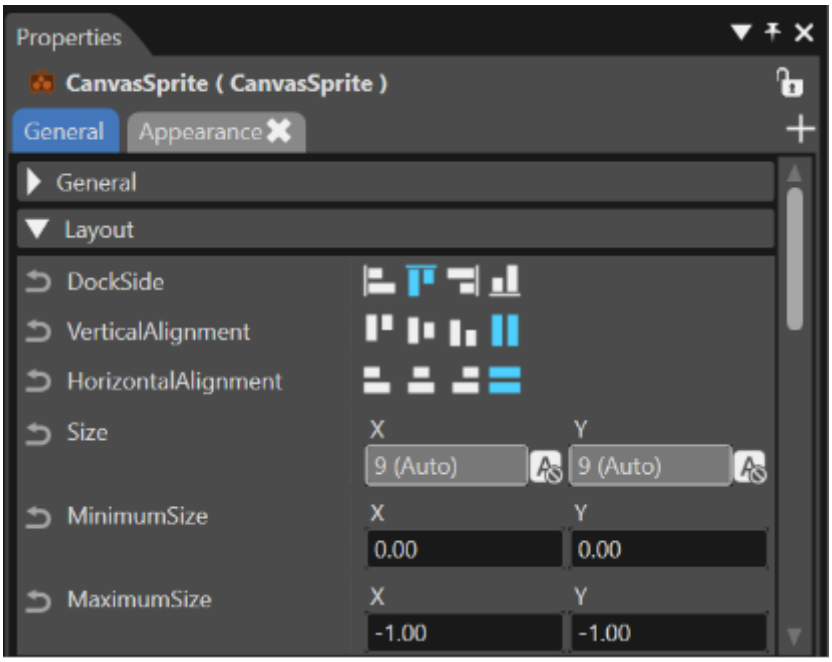
The "DockPanel" layout provides a layout area within it is possible to arrange canvas child elements around the edge of the screen based on four direction: Top, Bottom, Left, and Right.

This type of layout arranges the elements by laying out them in the remaining space after the previous element was set. To dock an element to the center of the panel, it must be the last child of the panel.



Configure the DockPanel Layouter by following these steps:

1. Select the Canvas Group holding the canvas nodes to be layouted in the Scene Tree panel
2. Choose Layout Type "DockPanel" from the Properties panel
3. For any canvas node inside the Canvas Group a new property will become available: "Dock Side". Select it and choose any of the fours available laying outs: "DockLeft", "DockTop", DockRight", "DockBottom".



Revision #40
Created 2023-02-14 13:06:58 UTC by Tsuyoshi.Kato
Updated 2025-09-09 23:59:37 UTC by Tsuyoshi.Kato