Multiple Render Targets

Description

This section explains how to handle Multiple Render Targets by means of a small ping pong like tutorial solution.

Example Solution

- RenderTargetsCamerasSolution in folder *cgi_studio_player/content/Tutorials/07_RenderTargetsCameras*

Handling Multiple Render Targets

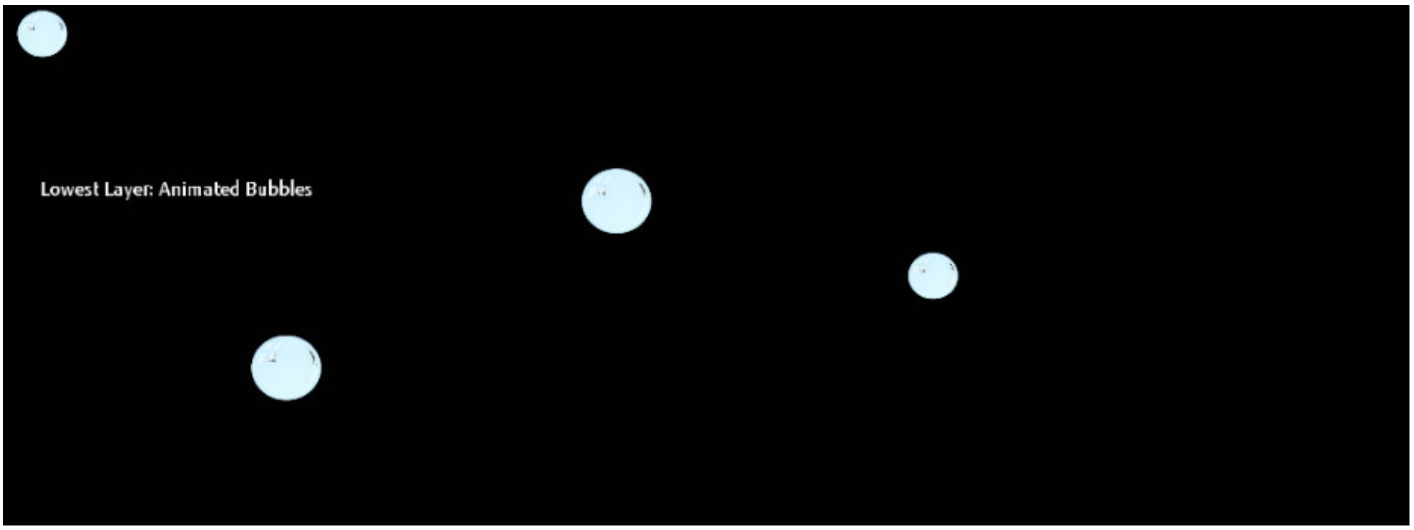
The example solution in *cgi_studio_player/content/Tutorials/07_RenderTargetsCameras* consists of the following:

- a main scene rendered to Candera::iMX6WindowSurface render target
- three different scenes rendered to three different Candera::SoftwareLayer render targets.

The three Candera::SoftwareLayer render targets are linked to the Candera::iMX6WindowSurface render target through their **Compostion Camera** property and will be blended against each other.

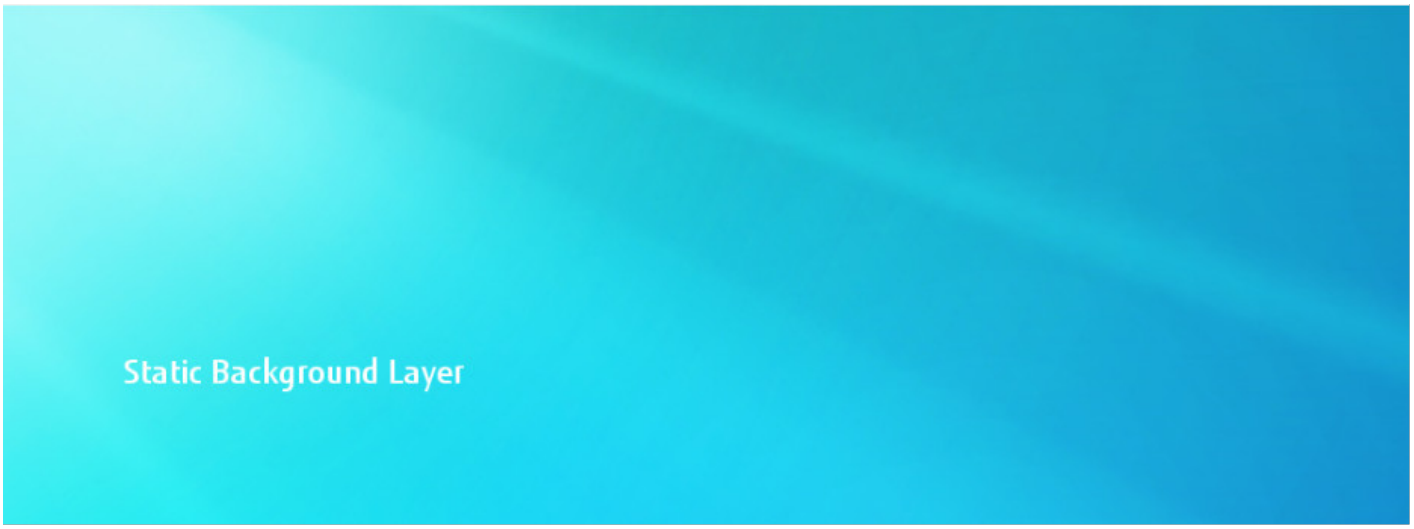
Layer 1: Animated 2D Background

The lowermost layer is a collection of bubbles which are animated to bubble endlessly from the bottom of the 2D screen to the top. It is the lowest layer, because the dreary, transparent bubbles shall be colored by blending with a more colorful layer.



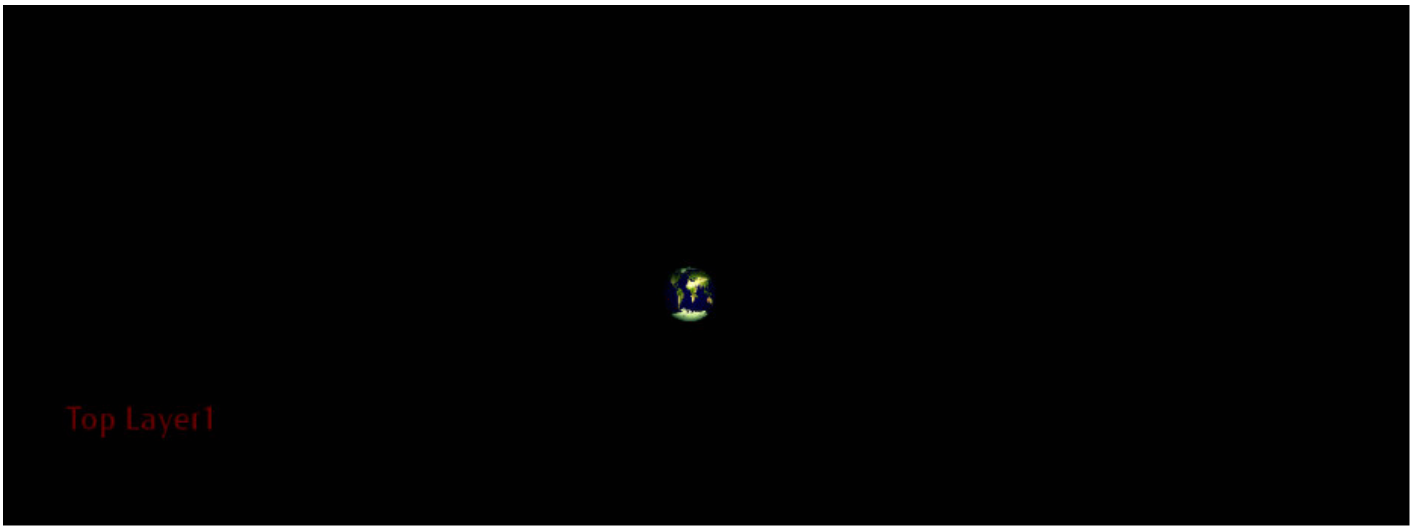
Layer 2: Static 2D Background

This is the layer which will give the animated bubbles some colors.



Layer 3: Dynamic 3D Top Layer

Finally, the top layer shows a 3D scene with a sphere.



The sphere can be linked with the "BouncingNodeWidget". Once this widget is enabled in the Player, it will send the sphere into several directions like in a Ping Pong game. This is the reason, why the camera of this scene needs an OrthographicProjection: The widget needs to match the sphere position on the x and y axis with the boundaries of the 2D projection plane.

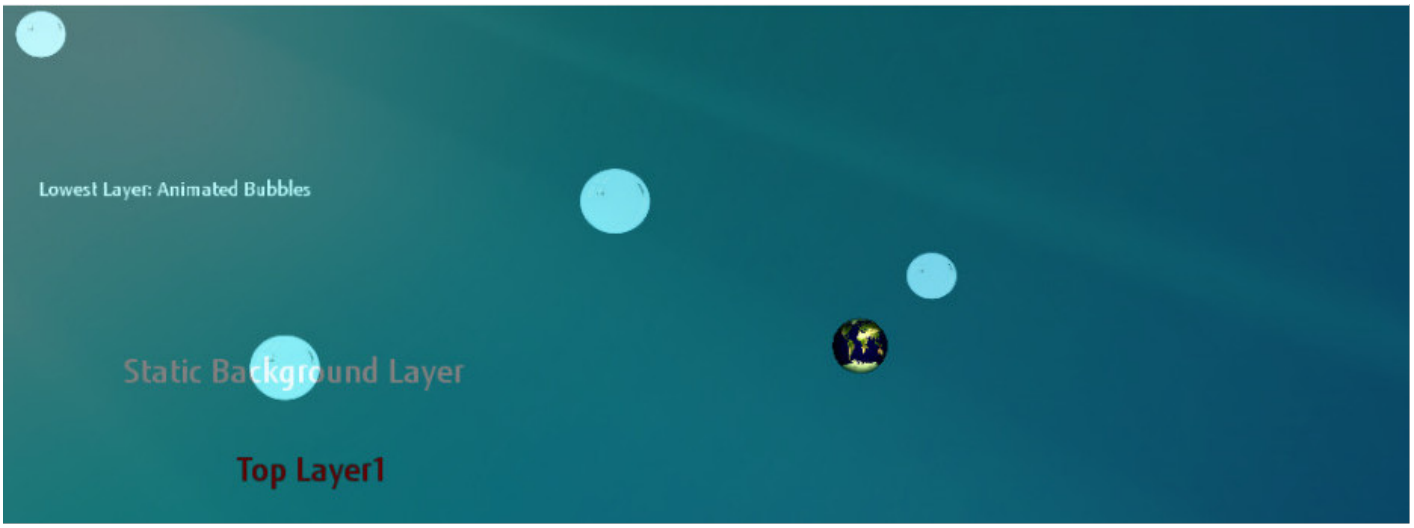
Use the velocity widget property to play with different velocities.

Layer Order and Layer Blending

Layer Order from Lowest to Top

The correct layer order must be specified on the property "Layer" on the render targets, the bigger value for the top layer:

- Lowest Layer: Layer-ID 0
- Medium Layer: Layer-ID 1
- Top Layer: Layer-ID 2



Export Asset and Load with the Player

- Export the example solution to a simulation asset
- When the asset is loaded in the Player, take care to first activate the main scene (Scene3DOwner from solution) and then the other scenes.

Render Target Manipulation

Accessing Render Target Properties

The BouncingNodeWidget uses a property to specify the name of a layer. Width and height of this layer is used by the widget to let the node bounce within these boundaries.

Refer to the following code example to access width and height of this layer:

```
        if (m_layer != 0){
            RenderTarget3D* rt3d = m_layer->ToRenderTarget3D();
            if (rt3d != 0){
                m_layerWidth = rt3d->GetWidth();
                m_layerHeight = rt3d->GetHeight();
            }
        }
```

Accessing Render Target Properties by MetaInfo

Same properties could be accessed by meta information as follows:

```
    const MetaInfo::GraphicDeviceUnitMetaInfo *meta = DevicePackageDescriptor::GetMetaInformatic
    if (meta){
        if (meta->LookupItem("Width") != 0){
            meta->LookupItem("Width")->Get(gdu,m_width,sizeof(m_width));
        }
        if (meta->LookupItem("Height") != 0){
            meta->LookupItem("Height")->Get(gdu,m_height,sizeof(m_height));
        }
        if (meta->LookupItem("BlendMode") != 0){
            meta->LookupItem("BlendMode")->Get(gdu, m_blendMode,sizeof(m_blendMode));
        }
        // ...
    }
```

Accessing Software Layer Properties

The code snippet below shows how to set the software layer properties for an iMX6 device.

```
    Candera::GraphicDeviceUnit* m_gdu = Candera::DevicePackageInterface::CreateGraphicDeviceUnit
    static_cast<Candera::SoftwareLayer*>(m_gdu)->GetSoftwareLayerProperties().CompositionCamera(
    //...
```

Revision #5
Created 2023-02-22 08:35:21 UTC by Tsuyoshi.Kato
Updated 2025-01-17 08:32:59 UTC by Elisabeth Zauner