2D Scene Graph Dynamics

Description

Usually all required nodes are already specified and preconfigured via SceneComposer, so in most cases it won't be necessary to make any adaptations within the scene graph loaded from an asset at runtime.

However, some use cases might require such adaptations. This tutorial covers the most common means to manipulate the scene graph structure dynamically.

Example Solution

- SceneGraphDynamicsSolution_2D in folder *cgi_studio_player/content/Tutorials/03_SceneGraphAndNodes*

Adding and Removing 2D Nodes

Scene Graph Dynamics 2D Example

There are some differences between a 2D and a 3D scene graph. This chapter will explain the differences in detail.

For 2D scene graph dynamics, the following examples can be used:

- **SceneGraphDynamicsSolution_2D** from folder *cgi_studio_player/content/Tutorials/03_SceneGraphAndNodes*
- **SceneGraphDynamicsWidget_2D**

Adding and Removing 2D Nodes

In the widget a RenderNode with text is created and added/removed to/from the 2D scene graph. For adding and removing 2D nodes on the scene graph the following properties can be used:

- **AddToNode2D:** Select the node where the new RenderNode should be attached. This node can be any sub-type of Candera::Node2D.

```
        CdaProperty(AddToNode, Candera::Node2D*, GetAddToNode, SetAddToNode)
            CdaDescription("Node2D, where a new TextNode shall be added to")
            CdaCategory("Adding and Removing 2D Nodes")
        CdaPropertyEnd()
```

- **TextNodeStyle :** Sets the style to be used for the text node. This allows the user to provide a custom style for the dynamic text.

```
CdaProperty(TextNodeStyle, Candera::TextRendering::SharedStyle::SharedPointer,
GetTextNodeStyle, SetTextNodeStyle)
 CdaDescription("Style for the new Text Node ")
 CdaCategory("Adding and Removing 2D Nodes ")
```

- **AddTextNode:** If AddTextNode is enabled, a new text node with the selected font is created and attached to the selected node. If disabled, the text node is removed from the scene graph again.

```
        CdaProperty(AddTextNode, bool, GetAddTextNode, SetAddTextNode)
            CdaDescription("If AddTextNode is enabled, a new TextNode with Selected
TextNodeStyle Will be added to the selected AddToNode property. If disabled the new TextNode
will be removed.")
            CdaCategory("Adding and Removing 2D Nodes")
        CdaPropertyEnd()
```

Creating a new Text Node

To display new text, a new TextNode2D is created. TextNode2D represents a Render Node (Candera::TextNode2D). It requires a BitmapBrush effect for rendering. The Candera::BitmapTextNodeRenderer then generates the bitmap images from the given text, and these images are finally rendered using the attached Bitmap Brush effect.

Create an instance of BitmapBrushEffect and a text node for the scene graph using the following code used.

```
BitmapBrushColorBlend::SharedPointer bbc = BitmapBrushColorBlend::Create();
bbc->GetColorEffect().Color().Set(Color(255 / 255, 255, 255 / 255, 1));
m_addedNode = TextNode2D::Create();
m_addedNode ->AddEffect(bbc.GetPointerToSharedInstance());
m_addedNode ->SetName("AddedTextNode");
```

```
m_addedNode ->SetText("AddedNode");
m_addedNode ->SetStyle(m_textNodeStyle);
m_addedNode ->SetTextNodeRenderer(BitmapTextNodeRenderer::Create());
m_addedNode ->SetPosition(100, 200);
```

Adding Text Node to Scene Graph

Next the text node is uploaded to VRAM and appended to the selected node.

```
        static_cast<void>(m_addedNode->Upload());
        static_cast<void>(m_addToNode->AddChild(m_addedNode));
```

Removing and Destructing Text Node

A 2D node can be removed by calling Candera::Node2D::RemoveChild. This is also described in the 3D part of this chapter (see Removing and destructing the Billboard). For removing and completely destructing the node, following code is used:

```
        static_cast<void>(m_addedNode->Unload());
        m_addedNode->Dispose();
        m_addedNode = 0;
```

The text node is unloaded from VRAM. The method Candera::Node2D::Dispose removes the node from its parent and frees the resources.

> In SceneComposer the manipulated scene graph and the changed properties do not appear because it is only changed dynamically by the widget.

> The widget must control memory and VRAM management of its internal dynamic subtree autonomously.

Replacing a 2D Effect

Replacing an Effect

In the SceneGraphDynamicsWidget_2D example, the effect of a Candera::RenderNode can be either replaced by a Candera::SolidColorBrushBlend or a Candera::BitmapBrushBlend effect. Of course the same procedure works for Text effects.

To replace the effect in the example, the following properties can be used:

- **NodeToChange:** Select any render node with an effect whose appearance should be changed. Ensure that just a render node which has an effect is selected (in the sample solution this could be e.g. the node BitmapNode_BarTop). For further information see Adding Replacing Effect.

```
        CdaProperty(NodeToChange, Candera::Node2D* , GetNodeToChange, SetNodeToChange)
            CdaDescription("Select a RenderNode which effect should be exchanged")
            CdaCategory("Replacing a 2D Effect")
        CdaPropertyEnd()
```

- **ChangeEffect:** If ChangeEffect is enabled, the effect of the selected NodeToChange will be removed and a Candera::SolidColorBrushBlend effect is added to the node, instead. If disabled, the actual effect will be removed too and a Candera::BitmapBrushBlend effect is added to the node.

```
        CdaProperty(ChangeEffect, bool, GetChangeEffect, SetChangeEffect)
            CdaDescription("Change
effect of selected NodeToChange. Set either a SolidColorBrushAlphaBlend (enabled) or BitmapBrus
            CdaCategory("Replacing a 2D Effect")
        CdaPropertyEnd()
```

Creating a Color Effect

First a new effect is created by creating an instance of Candera::SolidColorBrushBlend.

> 2D effects are part of the device package, therefore in order to create a concrete instance of the effect, the effect must be enabled for the device. Please see chapter **2D Effects** of this tutorial to see how to enable an effect for a certain device.

```
        Candera::SolidColorBrushBlend::SharedPointer brush =
Candera::SolidColorBrushBlend::Create();
        brush->GetSolidColorBrush().Size().Set(m_Rect);
        brush->GetSolidColorBrush().Color().Set(m_ColorBlue);
        brush->GetBlendEffect().SetBlendMode(RenderDevice2D::SourceAlpha, RenderDevice2D::Invers
```

Adding Replacing Effect

Once the effect is created it can be added to the render node. In the following code snippet the selected NodeToChange is cast to a Candera::RenderNode (for this a RenderNode **must** be selected, otherwise the cast will fail). The first effect of the node is removed via its reference and the new created effect added.

```
    RenderNode* rn = Dynamic_Cast<RenderNode*>(m_nodeToChange);
    if (rn == 0) {
        return;
    }

    static_cast<void>(rn->Unload());

    //remove old effect
    Effect2D* oldEffect = rn->GetEffect(0);
    static_cast<void>(rn->RemoveEffect(oldEffect));

    //add new effect
    static_cast<void>(rn->AddEffect(effect));
    static_cast<void>(rn->Upload());
```

Creating a Bitmap Effect

Creating a Candera::BitmapBrushBlend effect is similar to creating other effects (see Creating a Color Effect). Just apply other properties.

```
        Candera::MemoryManagement::SharedPointer<Candera::BitmapBrushBlend> brush =
Candera::BitmapBrushBlend::Create();
        Bitmap::SharedPointer bitmap = Base::GetAssetProvider()->GetBitmapById(CgiAssetNames::cc
        if (m_image != 0) {
            static_cast<void>(m_image->SetBitmap(bitmap));
        }
        brush.GetSharedInstance().GetBitmapBrush().Image().Set(m_image);
        brush.GetSharedInstance().GetBlendEffect
().SetBlendMode(RenderDevice2D::SourceAlpha, RenderDevice2D::InverseSourceAlpha, RenderDevice2D:
```

> In SceneComposer the manipulated scene graph and the changed properties do not appear because it is only changed dynamically by the widget.

> The widget must control memory and VRAM management of its internal dynamic subtree autonomously.

>

> Don't forget to set a name for the dynamically created nodes.

Cloning 2D Nodes

Cloning 2D nodes is equivalent to cloning of 3D nodes. In accordance to Tree Cloning Strategies, the following support is provided for deep cloning:

- Candera::TreeCloner2D is a specialization for the 2D scene tree of Candera::TreeClonerBase.
- Candera::DeepNode2DCloneStrategy provides a generic node strategy that clones the delegates the cloning of derived Nodes to specialized strategies.
- Candera::DeepCompositeGroup2DCloneStrategy resolves node binding by use of Candera::TreeMatch2D.
- Candera::DeepRenderNodeCloneStrategy resolves the sharing of effects by use of a map.
- Candera::DeepTreeCloner2D is a wrapper of Candera::DeepNode2DCloneStrategy that provides an interface similar to Candera::TreeCloner2D.

Refer to

- Candera::Node2D::Clone

as well as the example for cloning 3D nodes:

- Cloning 3D Nodes

Revision #15
Created 2023-02-21 05:58:49 UTC by Tsuyoshi.Kato
Updated 2026-08-24 07:32:25 UTC by Kanai Tomoaki