2D Node Transformations

2D Node Manipulation Example

To illustrate basic node transformation operations, the *NodeManipulationWidget_2D* and the *DisplayPositionWidget2D* part of the Tutorial Widgets cand be used. An usage example for *NodeManipulationWidget_2D* can be seen in **NodeManipulationSolution_2D** solution provided in the content folder of *cgi_studio_player*.

Please consider:

- A *Node2DToTransform* on which the following transformations are made has to be selected in the widget.
- Only if the widget is *enabled* (this property is also derived from a base class), the setter methods of the widget properties will take effect.
- If a widget property setter changes a node property of the associated node, the visual effect can be seen, but note that the static node properties maintained by SceneComposer are not changed accordingly!
- Widget dynamics are never reflected to static scene structure. It is recommended to use SceneComposer only for property configuration and dynamics only in the Player.

Translations

Once the NodeManipulationWidget_2D is linked with a node from the static scene tree and enabled, the position coordinates can be set in the Player for this property to change the position.

```
            // Set node position
            m_node->SetPosition(m_position);    // set absolute node position
            // End Set node position
```

```
            // Translate node
            m_node->Translate(m_translate);     // translate actual node position
            // End Translate node
```

>

> Candera::Transformable2D::SetPosition sets the absolute position of a node compared toCandera::Transformable2D::Translate, which adds the given vector to the current position. So setting the position first and then translating the node gives a different behavior than vice versa.

Rotations

Similar to translation, rotation can also be applied via Candera::Transformable2D::Rotate or Candera::Transformable2D::SetRotation.

```
            // Rotate node
            m_node->Rotate(m_rotate);
            // End Rotate node
```

Scaling

Again there is also a Candera::Transformable2D::Scale and Candera::Transformable::SetScale method.

```
            // Scale node
            m_node->Scale(m_scale);
            // End Scale node
```

Pivot Point

It is possible to change your pivot point with the Candera::Transformable2D::TranslatePivotPoint (or Candera::Transformable2D::SetPivotPoint) method. The pivot point can be used to define e.g. the rotation point that the node should rotate around. It affects the scaling too.

How to get the screen space coordinates of a node

With the DisplayPositionWidget2D you can retrieve the screen coordinates of an associated node. Please consider that a node and a camera have to be selected in the widget. In the solution the widget is disabled by default, so if you want to see an output of your display position, you need to enable it. To get the screen space coordinates of an associated node, you simply need to add the render target position of your node to the window position.

- First you need the world position of your node. With this position you can retrieve the viewport position, which you need for retrieving the render target position.

```
        Vector2 worldPosition = node->GetWorldPosition();
        Vector2 viewportPos = Math2D::TransformSceneToViewport(*camera, worldPosition);
        Vector2 renderTargetPos = Math2D::TransformViewportToRenderTarget(*camera, viewport
```

- Now you need the window.

```
        Window* w = camera->GetRenderTarget()->GetGraphicDeviceUnit()->ToWindow();
        if (w != 0) {
            Vector2 delta(static_cast<Float>(w->GetX()), static_cast<Float>(w->GetY()));
```

- With these two Candera::Vector2 you can calculate the screen space Candera::Vector2 of your node.

```
            Vector2 displayPos = delta + renderTargetPos;
```

- displayPos now contains the X and Y screen space coordinates of the selected node.

Revision #4
Created 2023-02-21 05:55:57 UTC by Tsuyoshi.Kato
Updated 2023-06-19 00:45:56 UTC by Tsuyoshi.Kato