Tutorial for Special 2D Render Techniques

2D Shadow Effect

Description

This chapter briefly describes how to use the 2D shadow effect.

> Ensure that this effect is supported by the device platform in use.

2D Configure a Shadow Effect

Shadow Effect

By using Candera::ShadowBitmapBrushBlend, a colored shadow can be applied to a bitmap.

Following properties of the shadow can be configured:

- Color (including alpha value)
- Scale (the shadow size related to the bitmap)
- Position Offset (related to the bitmap)

Following is an example of a grey colored shadow, with a scale of 70% and an offset of 80/80 related to the bitmap:

Refer to SceneComposer User Manual about how to configure a ShadowBitmapBrushBlend effect in a 2D scene.

Modify 2D Shadow Effect Properties

Access Shadow Effect

To modify the shadow effect properties, first the shadow effect needs to be retrieved from its render node:

```
    // Get and cast first effect of render node
    Effect2D* effect2D = renderNode->GetEffect(0);
    ShadowBitmapBrushBlend* shadow = 0;
    if (effect2D != 0){
        shadow = (ShadowBitmapBrushBlend*) effect2D;
    }
    // End Get and cast first effect of render node
```

Set Shadow Color

Specify a color with a red, a blue, a green and an alpha component and apply the color to the Shadow Effect.

Example:

```
Color::Data c;
c.alpha = 0.5f;
c.blue = 0.0f;
c.green = 0.0f;
c.red = 1.0f;
shadow->ShadowColor().Set(c);
```

Set Shadow Scale

For the shadow's scale you need the scale X and Y value. Add the two values to a Vector2 and set the shadow's scale.

Example:

```
Float shadowScaleX = 0.7f;
Float shadowScaleY = 0.7f;
shadow->ShadowScale().Set(Vector2(shadowScaleX, shadowScaleY));
```

Set Shadow Position Offset

It is possible to change the position of the shadow by changing its offset to the node. Therefore you need a Vector2 with the X and Y offset values again.

Example:

```
Float shadowPosOffX = 5.0f;
Float shadowPosOffY = 7.0f;
shadow->ShadowPositionOffset().Set(Vector2(shadowPosOffX, shadowPosOffY));
```

2D Mirror Effect

Description

This chapter briefly describes how to use the 2D mirror effect.

> Ensure that this effect is supported by the device platform in use.

Configure a 2D Mirror Effect

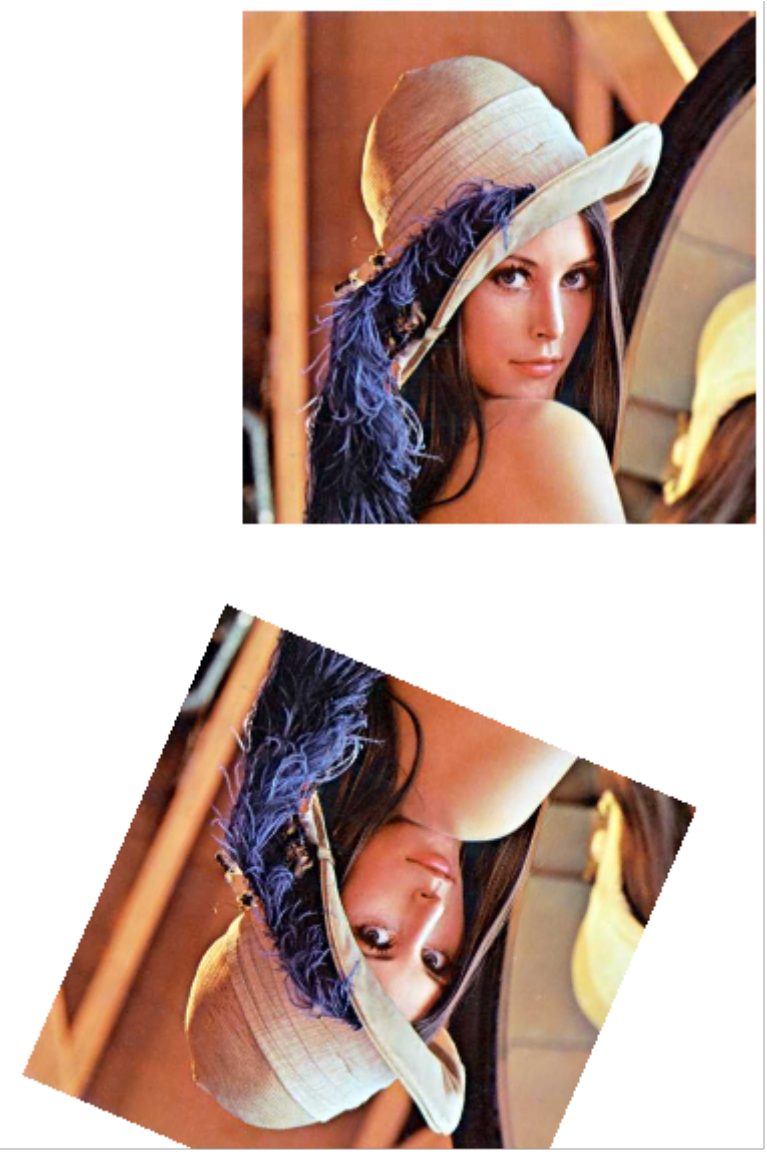
Mirror Effect

By using Candera::MirrorBitmapBrushBlend, a mirror reflection can be applied to a bitmap node.

Following properties of a mirror effect can be configured:

- Mirror Axis Modifications
- Alpha Value

Following is an example of a mirror effect with a diagonal mirror axis and an alpha value of 1.0f:



Refer to SceneComposer User Manual about how to configure a MirrorBitmapBrushBlend effect in a 2D scene.

Modify 2D Mirror Effect Properties

Set Mirror Axis

To set the Mirror Axis you need to set two vectors.

- The first vector defines the start point of the mirror axis (the startpoint's X and Y value),
- and the second vector defines the end of the axis.

The following example shows how to implement a mirror axis from the startpoint (0, 400) to the endpoint (700, 400).

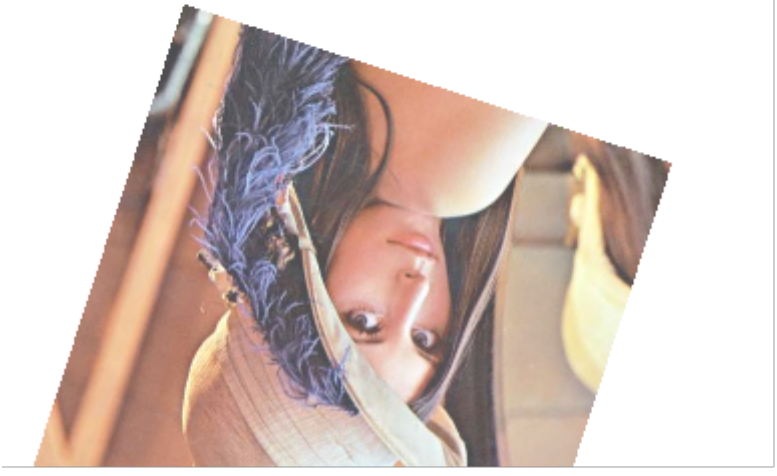
```
Float fromX = 0.0f;
Float fromY = 400.0f;
Float toX = 700.0f;
Float toY = 400.0f;

mirror->MirrorAxisFrom().Set(Vector2(fromX, fromY));
mirror->MirrorAxisTo().Set(Vector2(toX, toY));
```

Set Mirror Alpha Value

It is possible to set the mirrors alpha value, so that the mirror looks more or less intensive.

```
Float mirrorAlpha = 0.7f;
mirror->Alpha()= mirrorAlpha;
```



2D Blur Effect

Description

This chapter briefly describes how to use the 2D blur effect.

> Ensure that this effect is supported by the device platform in use.

Configure a 2D Blur Effect

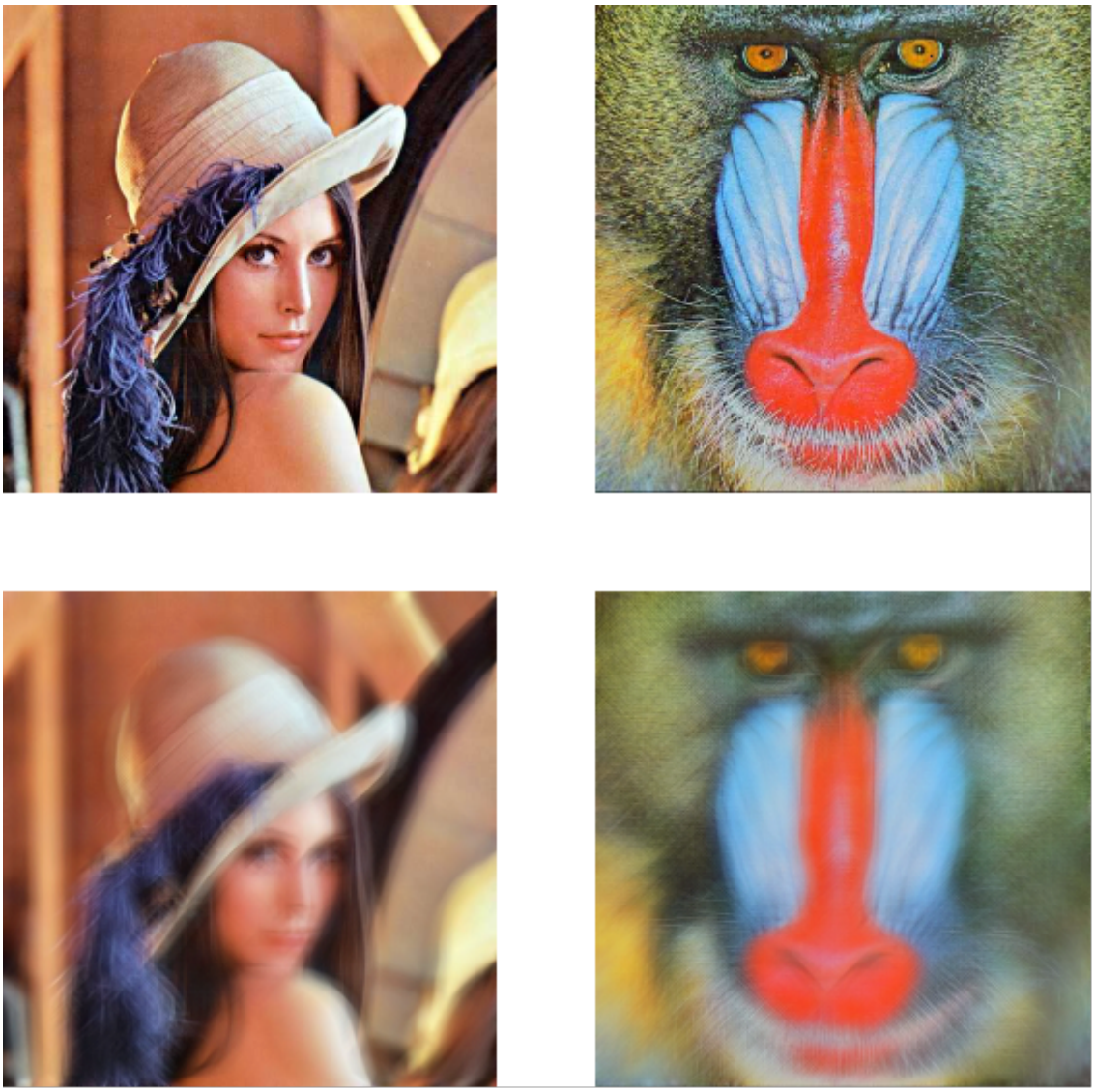
Blur Effect

By using Candera::BlurBitmapBrushBlend, a bitmap node can be blurred, applying a gaussean filter convolution.

Blur Effect supports the following property:

- Filtersize (Number of neighbour pixels)

In this example, the upper two pictures are the original ones, the lower two are blurred with a filter size of 15.



Refer to SceneComposer User Manual about how to configure a BlurBitmapBrushBlend effect in a 2D scene.

Modify 2D Blur Effect Property

Set Blur Filtersize

When using this effect you only have to set the filtersize. This value defines the blur factor of, for example, an image. The bigger the value, the more the image is blurred. The following example shows how to set the filtersize.

```
UInt8 factor = 5;
blur->FilterSize() = factor;
```

2D Shear Effect

Description

This chapter briefly describes how to use the 2D shear effect.

> Ensure that this effect is supported by the device platform in use.

2D Configure a Shear Effect

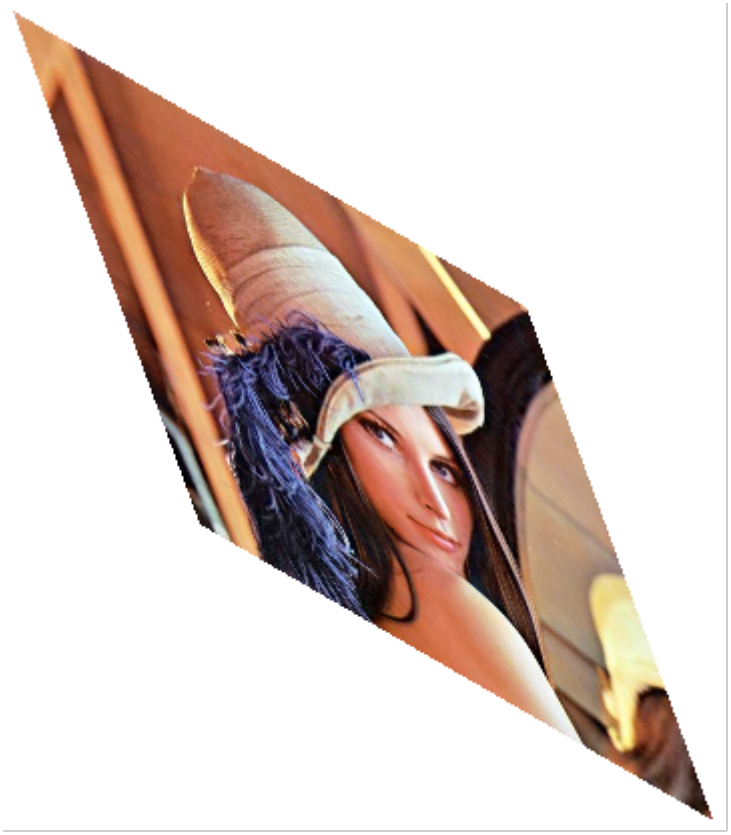
Shear Effect

By using Candera::ShearBitmapBrushBlend, the effect shears the incoming surface horizontally and vertically using the given angles.

Following properties for shearing can be configured:

- Shear Angle X
- Shear Angle Y

Following is an example of a skewed bitmap, with the angles (in degree): X = 20 and Y = 30.



Refer to SceneComposer User Manual about how to configure a ShearBitmapBrushBlend effect in a 2D scene.

Modify 2D Shear Effect Properties

Set Shear Angles

Horizontal and a vertical skew can be modified by setting the ShearAngle to proper values. You can set both, horizontal and vertical skew, to values from -180 to 180.

Example:

```
    // Set Shear Angles X and Y
    shear.ShearAngleX().Set(30.0f);
    shear.ShearAngleY().Set(20.0f);
    // End Set Shear Angles X and Y
```

2D HSL Effect

Description

This chapter briefly describes how to use the 2D HSL effect.

> Ensure that this effect is supported by the device platform in use.

Configure a 2D HSL Effect

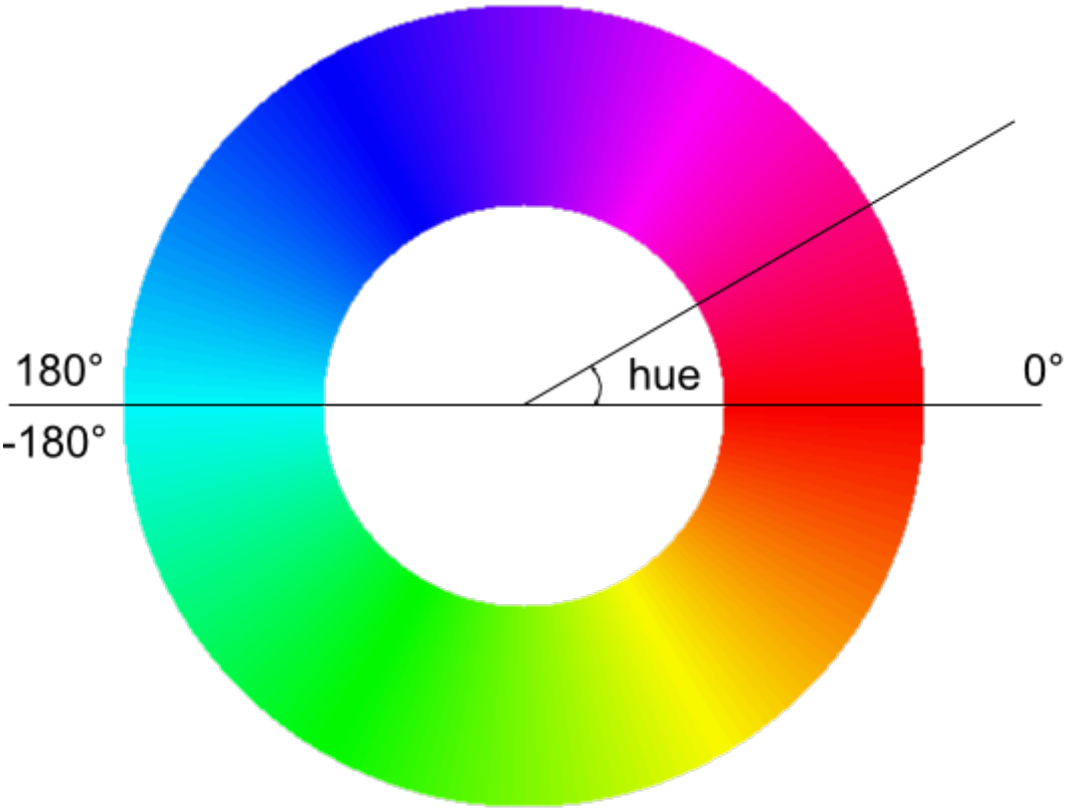
HSL Effect

The BitmapBrushHslBlend effect outputs an bitmap image having applied hue, saturation and lightness correction and blended with the store buffer.

Following properties of a HSL effect can be configured:

Hue

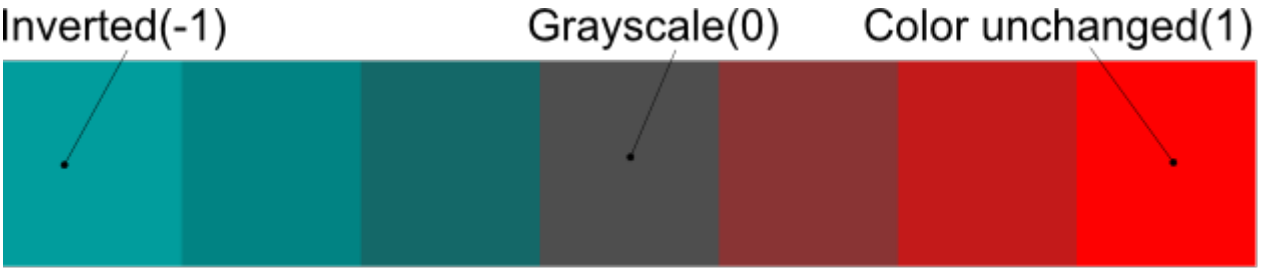
Hue is one of the main properties of a color, defined, technically, as the degree to which a stimulus can be described as similar to or different from stimuli that are described as red, green, blue, yellow, orange and violet. The image below shows the circle of hues for the color red (R=255, G=0, B=0).



On the HSL effect, the hue is expressed as degrees in the interval [-180 .. 180]; the original color is at a hue value of 0.

Saturation

Saturation describes the intensity of a color, defined as a range from pure color to gray.



On the HSL effect, "0" means grayscale, "1" means color unchanged and "-1" means color inverted.

Lightness

Lightness measures the relative degree of black or white that's been mixed with a given hue.

On the HSL effect "0" means black, "1" means the color unchanged.

Example

Following is an example of a HSL effect configured with hue = 120, saturation = -1 and lightness = 1.3:

Refer to SceneComposer User Manual about how to configure a BitmapBrushHslBlend effect in a 2D scene.

Modify 2D HSL Effect Properties

Set Hue, Saturation and Lightness

In order to change the BitmapBrushHslBlend effect properties, call the corresponding method on the HslCorrectionEffect as shown bellow:

```
            m_bitmap_hslBlend = BitmapBrushHslBlend::Create();
            m_bitmap_hslBlend->GetBitmapBrush().Image().Set(m_image.GetPointerToSharedInstance()

            HslCorrectionEffect& hslCorrection = m_bitmap_hslBlend->GetHslCorrectionEffect();
            hslCorrection.Hue() = 17.0F;
            hslCorrection.Saturation() = -0.5F;
            hslCorrection.Lightness() = 1.2F;
```

2D Alpha Mask Effect

Description

This chapter briefly describes how to use the 2D alpha mask effect.

> Ensure that this effect is supported by the device platform in use.

Configure a 2D Alpha Mask Effect

Alpha Mask Effect

The alpha mask effect creates the appearance of partial or full transparency of an image by using the information from the alpha channel of a mask.

This appearance can be applied to a node by using one of the following effects:

- GlBitmapBrushMaskBlend
- GlBitmapBrushColorMaskBlend

Configuration

Following properties can be configured:

Image

Represents the image attached to the effect.

Mask

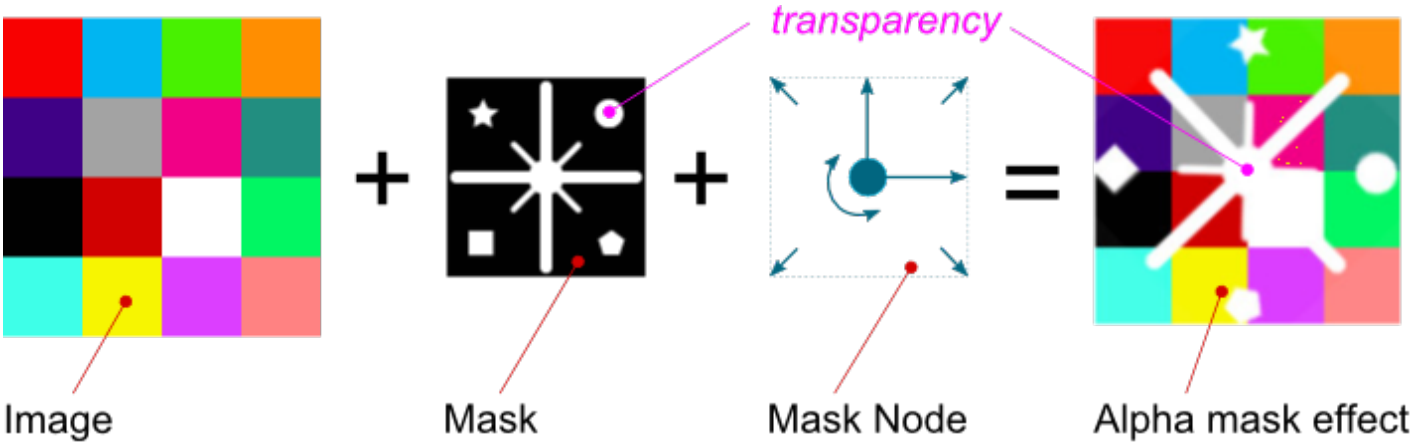
It's the image used as mask. Only the alpha channel of the mask is used.

Mask Node

It's the node used to control the transformation of the mask (position, rotation and scaling). If set to 0 the transformation of the current node is used.

Effect Example

The image below presents an alpha mask effect example. The mask node is rotated 45 degrees, the pivot offset is set to (-50;-50) and scaled to (1.5;1.5).



Refer to SceneComposer User Manual about how to configure an alpha mask effect in a 2D scene.

Modify 2D Alpha Mask Effect Properties

Set Effect Properties

```
//create effect
bitmap_maskBlend = GlBitmapBrushMaskBlend::Create();

//set image property
bitmap_maskBlend->GetBitmapBrush().Image().Set(m_image.GetPointerToSharedInstance());

//set mask property
SharedPointer<BitmapImage2D> maskBitImage = BitmapImage2D::Create();
maskBitImage->SetBitmap(mask_bitmap);
maskBitImage->Upload();
GlMaskEffect& maskEffect = bitmap_maskBlend->GetMaskEffect();
maskEffect.Mask() = maskBitImage.GetPointerToSharedInstance();

//set mask node
RenderNode* maskNode = RenderNode::Create();
maskNode->SetPosition(400.f, 30.f);
maskNode->SetScale(0.2f, 0.2f);
maskNode->SetPosition(m_bitmapNode5->GetPosition() + Vector2(8, 8));
maskEffect.MaskNode() = maskNode;

bitmap_maskBlend->Update();

//attach effect to the node
node = RenderNode::Create();
node->AddEffect(bitmap_maskBlend.GetPointerToSharedInstance());
```