Objectives

1. To understand how to render objects into textures rather than to the screen.
2. To learn how to implement the render-to-texture technique in DirectX 11
3. To understand some simple texture processing techniques
4. To Create a glowing effect

The lab exercises described below are based on DirectX 11 Tutorial 7, they can be completed directly based on what you have achieved during the lab in Week 8 on Texture mapping.

Exercise 1. Render to texture

1. The first thing you need to do is to create a texture object to store the rendered scene. This is similar to create an ordinary 2D texture object. First, we first declare a 2D texture object using the Direct3D 11 interface ID3D11Texture2D:

```
ID3D11Texture2D* g_pRenderableTexture = nullptr;
```

2. Unlike an ordinary texture, this newly created texture object will be used both as a render target and a shader resource. Thus two views of the texture need to be created:

```
ID3D11RenderTargetView* g_pRenderableTexture_RTV = nullptr;
ID3D11ShaderResourceView* g_pRenderableTexture_SRV = nullptr;
```

3. Describes the 2D texture by using the D3D11_TEXTURE2D_DESC structure, make sure this texture object can be bound to the Direct3D 11 output merger stage both as a render target and a shader resource:

```
D3D11_TEXTURE2D_DESC renderableTextureDesc;
ZeroMemory(&renderableTextureDesc, sizeof(renderableTextureDesc));

renderableTextureDesc.Width = width;
renderableTextureDesc.Height = height;
renderableTextureDesc.MipLevels = 1;
renderableTextureDesc.ArraySize = 1;
renderableTextureDesc.Format = DXGI_FORMAT_R32G32B32A32_FLOAT;
renderableTextureDesc.SampleDesc.Count = 1;
renderableTextureDesc.Usage = D3D11_USAGE_DEFAULT;
renderableTextureDesc.BindFlags = D3D11_BIND_RENDER_TARGET | D3D11_BIND_SHADER_RESOURCE;
renderableTextureDesc.CPUAccessFlags = 0;
renderableTextureDesc.MiscFlags = 0;

g_pd3dDevice->CreateTexture2D(&renderableTextureDesc, NULL,
                                                &g_pRenderableTexture);
```

4. Create a render target view corresponds to the texture created above. Make sure that the format of render target view is the same as the texture's format

```
D3D11_RENDER_TARGET_VIEW_DESC renderable_RTV_Desc;
renderable_RTV_Desc.Format = renderableTextureDesc.Format;
renderable_RTV_Desc.ViewDimension = D3D11_RTV_DIMENSION_TEXTURE2D;
renderable_RTV_Desc.Texture2D.MipSlice = 0;


g_pd3dDevice->CreateRenderTargetView(g_pRenderableTexture, nullptr,
                                        &g_pRenderableTexture_RTV);
```

5. Create a shader resource view for this newly created texture object. Same as creating the render target view, the format of the shader resource view should be the texture's format.

```
D3D11_SHADER_RESOURCE_VIEW_DESC renderable_SRV_Desc;

renderable_SRV_Desc.Format = renderableTextureDesc.Format;
renderable_SRV_Desc.ViewDimension = D3D11_SRV_DIMENSION_TEXTURE2D;
renderable_SRV_Desc.Texture2D.MostDetailedMip = 0;
renderable_SRV_Desc.Texture2D.MipLevels = 1;
```

```
g_pd3dDevice->CreateShaderResourceView(g_pRenderableTexture, &renderable_SRV_Desc,
                                       &g_pRenderableTexture_SRV);
```

6. In Render( ) method, render the cube into a texture by setting the texture created above as the render target at the output merger stage.

```
g_pImmediateContext->OMSetRenderTargets(1, &g_pRenderableTexture_RTV,
                                        g_pDepthStencilView);
```

7. Rendering the object by issuing draw calls.

```
g_pImmediateContext->VSSetShader( … …);
… …
… …
g_pImmediateContext->DrawIndexed( 36, 0, 0 );
```

8. Reset the render target back to the screen.

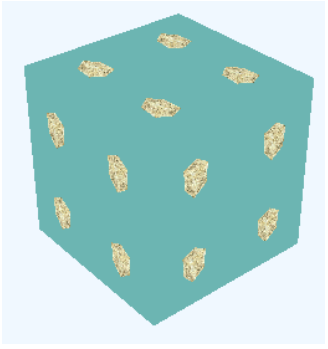
```
g_pImmediateContext->OMSetRenderTargets(1, &g_pRenderTargetView,
                                        g_pDepthStencilView);
g_pImmediateContext->ClearDepthStencilView(g_pDepthStencilView,
                                        D3D11_CLEAR_DEPTH, 1.0f, 0);
```

9. Once the texture has been generated in step 7, it can be used as an ordinary texture. For instance, to map the texture on to the cube made, you need just to set the rendered texture as a pixel shader resource.

```
g_pImmediateContext->VSSetShader( … …);
… …
… …
g_pImmediateContext->DrawIndexed( 36, 0, 0 );
```

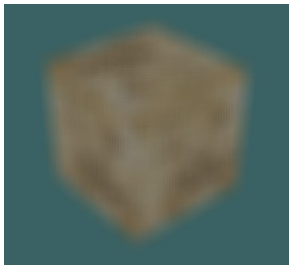
Screen image

Screen image used a texture



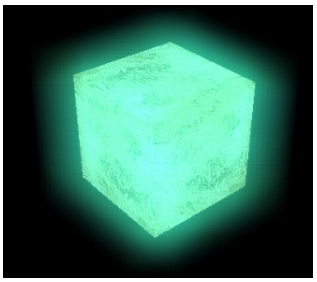
Exercise 2. Texture smoothing

Render an object into a texture and use one of the image smoothing techniques introduced in the lectures to smooth the texture.



Exercise 3. Glowing effect

Combine the original unprocessed rendered texture and the smoothed texture to create a glowing effect.



Exercise 4. Object on fire animation

Use the smoothed rendered texture as the basic fire shape to implement a texture-based fire animation effects shown below.