DirectX 11 Tutorial 17 Part1 – Textures, Reflections and Cube Mapping

This tutorial focuses on Vertex and Pixel Shaders.

You are required to:

SkyBox Vertex Shader

1. complete the transformation of the vertex positions so that the sky box is always rendered on the far clipping plane
2. output the correct 3D texture coordinates to sample from the cube map texture.

Reflection Mapping

1. Load the shaders for reflection mapping in place of the per pixel lighting shaders Scene::initialiseSceneResources())
2. calculate the reflection vector for the eye direction and use it to sample the reflected colour from the sky box environment (reflection\_mapping\_PS.hlsl)

If you have time after completing these tasks research "Fresnel Bias". How might "Fresnel Bias" be applied to the reflection mapping Pixel shader.

The cBuffers in the code are not used efficiently how might the code be reorganised to make more efficient use of cBuffers.

Download the starting source code for week 17.

Note: in this project the CBufferBasic structure is not used to manage the View and Projection matrices. Instead the Camera class manages the attributes of the camera and updates the cameraCbuffer when its update method is called. The World matrix management is now delegated to the base model class. On calling the update method of the box the modelCBuffer is mapped.

Review the week 16 lecture notes "Lecture 16b - Reflections and Cube Mapping" and complete the following tasks:

Task 1

The first task of this week's tutorial is to complete the sky\_box\_VS.hlsl shader

Search the source code for the comments:

// Add Code Here (ensure .z is on far clipping plane)

Refer to the lecture notes and add the appropriate initialisation code.

Task 2

Search the source code for the comments:

// Add Code Here (Output the correct 3D texture coordinates )

Task 3

Output the correct 3D texture coordinates to sample from the cube map texture.

// Add code here scale the box x1000

box->setWorldMatrix(box->getWorldMatrix()\*XMMatrixScaling(1000, 1000, 1000));

Task 4

Add an orb to the scene

// Add a chrome sphere to reflect the skybox

// The Model class is derived from the BaseModel class with an additional load method

// that loads 3d data from a file

// The load method makes use of the ASSIMP (open ASSet IMPort) Library for loading

// 3d data http://assimp.sourceforge.net/.

**orb = new Model(device, wstring(L"Resources\\Models\\sphere.3ds"),** perPixelLightingEffect, NULL, 0, skyBoxTextureArray, 1);

//Raise orb off the ground

orb->setWorldMatrix(orb->getWorldMatrix()\*XMMatrixTranslation(0, 2, 0));

orb->update(context);

// Render Orb

if (orb)

orb->render(context);

Task 5

Search the source code for the comments (in Scene::initialiseSceneResources()):

// Add Code Here ( Load reflection\_map\_vs.cso and reflection\_map\_ps.cso )

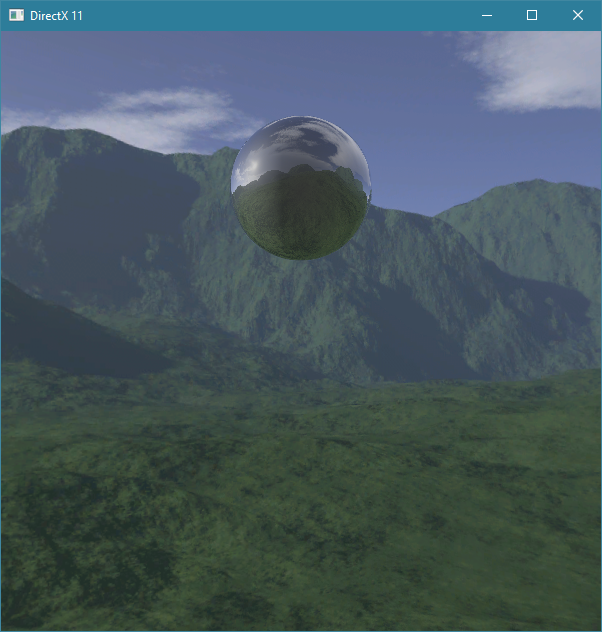
Load the shaders for reflection mapping (reflection\_map\_PS.hlsl and reflection\_map\_VS.hlsl) in place of the per pixel lighting shaders.

Task 6

Search the source code for the comments:

// Add Code Here (Calculate reflection vector ER)

Calculate the reflection vector for the eye direction and use it to sample the reflected colour from the sky box environment (reflection\_mapping\_PS.hlsl)



Screen shot of required outcome.