Direct X Midterm: Starscape

Objective

Create a space scene using your knowledge of Geometry, Shaders and Texturing.

Exam Instructions

You have 6 hours to complete as much of this exam as possible. You may reference any previously completed labs, as well as slides and any documentation.

**Repeaters:** **NO** **OBJ** **FILES!** Your ability to find and use model files is not what I am testing.

Scoring Breakdown

Ground is drawing. 5 points

Ground is warped demonstrating it is in fact a grid. 10 points

Ground is textured with correct texture. 5 points

Ground texture repeats the proper amount of times. 5 points

The Black squares on the Ground are Invisible. 10 points

Camera is positioned in the correct location. 5 points

Star is drawing. 10 points

Star is has no culling issues. 5 points

Star is positioned, rotating & scaled correctly. 5 points

Star is reflection mapped with gold texture according to formula. 10 points

Ground undulates smoothly based on time and provided formula. 10 points

Skybox cube map is rendering properly in the background. 10 points

Skybox rotates smoothly displaying the distant environment. 10 points

**Total 100 points**

Explanation of Tasks

**The Ground:**

***Geometry:***

The ground is a flat horizontal plane of **50x50** vertices centered on the origin. Each vertex is spaced apart from each other vertex by **0.1**. Besides position, each vertex contains **a** **UV coordinate as well**. To render these vertices as a **list of triangles**, you must build an **index list** to form all the needed triangles. The following algorithms can be used to assist you when **building the index array**.

**numTriangles = (TotalGridWidth -1) \* (TotalGridLength -1) \* 2**

(the below formula might be used during a nested loop, forming your triangles)

**vertexStartIndex = currGridRow \* GridWidth + currGridColumn**

There are **multiple ways to form the vertex and index buffers you will need**, (I actually used indexed triangle strips) but the algorithms above can be used to **assist** you when attempting to figure out how many triangles you will need, and how to convert a 2D grid location into a 1D vertex index location.

***The Texture:***

The ground uses the “checkerboard.dds” 2D texture. It should **repeat** across the surface **3 times**.

***Making the Ground Undulate:***

Use the Following formula in the Vertex Shader to manipulate a vertex before rasterizing it:

**LocalSpaceY = SIN( COS( LocalSpaceX ) \* 4 + TotalTimeElapsed ) \* 0.1**

**LocalSpaceY + = SIN( COS( LocalSpaceZ ) \* 4 + TotalTimeElapsed ) \* 0.1**

***Hiding the Black Tiles:***

This is actually trivial to do. In the Pixel Shader for the ground, **grab the color from the texture** for the pixel. Now **add together the Red, Green and Blue** Channels. **IF** the **result is less than 0.001**, then call the **“discard”** HLSL intrinsic. This will prevent the pixel from being output.

**The Star:**

The Star is exactly the same as the star you made in Lab 2. The only difference being is that you will apply a cube map texture to create a shiny golden effect.

***The Star Points: (Feel free to use your own values; these are the exact values I used)***

Here are the 20 points that make up the star, starting from the top and going clockwise.

Front: **0.0x, 2.0y, -0.25z** Back: **0.0x, 2.0y, 0.25z**

Front: **0.5x, 0.5y, -0.25z** Back: **0.5x, 0.5y, 0.25z**

Front: **2.0x, 0.5y, -0.25z** Back: **2.0x, 0.5y, 0.25z**

Front: **0.7x,-0.5y, -0.25z** Back: **0.7x,-0.5y, 0.25z**

Front: **1.0x,-2.0y, -0.25z** Back: **1.0x,-2.0y, 0.25z**

Front: **0.0x,-1.0y, -0.25z** Back: **0.0x,-1.0y, 0.25z**

Front: **-1.0x,-2.0y, -0.25z** Back: **-1.0x,-2.0y, 0.25z**

Front: **-0.7x,-0.5y, -0.25z** Back: **-0.7x,-0.5y, 0.25z**

Front: **-2.0x, 0.5y, -0.25z** Back: **-2.0x, 0.5y, 0.25z**

Front: **-0.5x, 0.5y, -0.25z** Back: **-0.5x, 0.5y, 0.25z**

***Positioning the Star:***

Translation[ **0x 2.5y 0z** ] Rotation[ **57.3 degrees per second** ] Scaling[ **50%** ]

***The Golden Shine:***

To re-create the shiny effect apply the **“GoldenShine.dds” cube map** to the star. To do this you **will not actually need any UV coordinates.** Instead **sample** the cube texture by **using a direction vector** calculated in the following way: (you can do this in the VS then send to PS)

**CubeMapSampleVector = MatrixMultiply ( LocalSpaceXYZasDirection, WorldMatrix )**

**The Background:**

The stars & nebula in the background are from **a slowly rotating skybox** like the one found in homework 3. To draw the skybox, just do the following steps:

**Create a cube** just like the one in homework 2 / Lab 3.

**Position the cube where the camera is** in world space and bind the “PurpleNebula.dds” map.

Before drawing anything else, **draw the cube inside out**. (CCW)

Use the same exact formula from “*The Golden Shine*” to **sample the nebula cube map**.

Finally **clear your Z buffer again** and draw the rest of the scene.

***Rotating the Sky:***

To make the sky appear to shift over time you must slowly rotate the skybox once it is surrounding the camera matrix.

Rate of Rotation: **2.865 degrees per second on Local Y then Local Z axis.**

**Misc Values:**

***Initial Camera Location:*** Eye[**0x,1y,-3z**] LookAt[**0x,1y,0z**] Up[**0x,1y,0z**]

***Projection Values:*** Field of View[**75degrees**] Znear[**0.1**] Zfar[**100**]

***Back-Buffer Resolution:*** [**1280w x 768h**]

**Hints & Tips:**

If you can figure out what the starting vertex for your indexed triangle is, the others are close by.

Use pencil & paper if you are having trouble building the Star or Ground.

**Turn In:**

Please be sure your project compiles. A project that has compiler errors is a ZERO.

**Take a screenshot of your exam and include it with your turn-in**.(printScreen/Fraps)

Please release all COM objects and check that your program runs with the DEBUG libraries.

Turn in your midterm on Sidekick and **carefully follow the Turn-In procedure** shown.

You may only leave early if the staff confirms that you have fully completed the exam.