Nicholas Renna

Instructor: Kurt Diesch

SNHU CS-330 Computer Graphic and Visualization

Project 3D Scene Reflection Paper

**Development choices:**

My scene is based off real objects that were found around my room where I do most of my computer programming and school projects. First the scene starts out with the complex object which is a small bird house. In the project it’s a cube with a pyramid sitting on top, so two different primitive shapes that form as one, the small house. I thought it would be interesting to develop a house because in my personal life I have interest and experience developing houses and remodels. Also, an interesting company in Arizona that is building affordable housing is looking for interns to support IT development. The object in my finished scene also has texture on the pyramid to support a roof shingles look, while the building has a greenish brick wall texture applied and another texture image is applied to create a door. Each part of the house is in their own function and positioned using XYZ coordinate system.

The other objects in my scene are a flashing light which represents a cylinder, a book that represents another cube but shaped differently, and a basketball which is represented as a sphere. All objects are sitting on a plane as well which is a wood texture desk. The cylinder was created with header files and other CPP files that would allow me to accurately create a cylinder which is a more complex object. Then in main (Urender) a call will be made to draw and then state the size of the object, while positioning and rotating the object using scale, rotate and translate. The book cube was created using vertices and indices with a texture applied to support the look of the 2D image. In creating the sphere, I would apply different header/CPP files that would help me create the shape and then call the newly CPP functions to create the basketball (sphere). A free texture of orange leather was found on google and a texture was applied to the basketball.

**User Navigation:**

The 3D scene while in run mode can be navigated using two input devices, a mouse, and a keyboard. The mouse which the cursor is disabled will move the camera motion, in different directions such as up, down, left, and right. For example, in a video game the joystick moves the camera or player’s view, and in my case, it does the same here except using the mouse to move the camera view. To move the camera vertically, horizontally and depth, the WASD and QE keys on the keyboard were implemented to provide extra interaction. On the keyboard the letter W will move the camera forward, A will move the camera left, S will move backwards, D will move the camera right, while Q moves the camera up and E moves the camera down. Also, you can use both the mouse and the keyboard at the same time to move around the scene. For instead, you may want to hover over the top of the scene to look down at it like you were flying above and both the keyboard and mouse will be handy for that type of movement. These movement processes were put into a function called UProcessInput and called to initialize the program at the top of the source code CPP file.

**Custom functions:**

During the planning phase of development, I chose to put each object in their own function because each object was a different shape or the dimensions of the shape was different, such as the cubes I have in my scene. One cube is part of a complex object (house) and the other is a book that is built differently. Inside the functions I have vertices and indices to support the rendering of the objects. Also, in the functions I have implemented vertex arrays to generate multiple VAOs, Buffers for the vertices and indices, and pointers tell OpenGL how it should interpret the data, one for vertex, one for color or normal, and the last pointer for textures. A mesh is used to mesh the objects together and is useful when you need to render multiple objects and in (Urender) a call would be made to draw the objects. Having the objects in their own functions allows them to be more organized and more efficiently to manage when you add more objects to the scene. The book mesh function was customized to support the shapes of the book in the 2D image, and the texture is custom to replicate an actual book.

The cylinder and sphere were designed in different CPP and header files which allows for better change control and keeps the code more modular because you can service the code or review sections in a more organized manner. In the source code I would just include the correct header files using inheritance properties which allows to share code already written saving time and complexly in the long run. Then in main (Urender) I would use a function call to call the cylinder or sphere to render the objects. Another custom feature was the reusing of code functions to create the lights in the scene or similar shapes that could change the size and location in the main (Urender) call.