**CS-330: Module 7-1: Final Project Submission: Reflection**

**Justin Carlson**

**Justify Development Choices for your 3D scene:**

I chose a mostly modular approach for my development of my 3D scene. For the most part, I utilized the Cylinder class to build out many cylinders (every single one of my objects contained one or more cylinders), as well as cubes of varying shapes and sizes. I chose the Cylinder class that was pre-built as it was the simplest way to develop it, as opposed to starting from scratch. When it comes to the cubes and the (singular) pyramid, I simply created sets of vertices with coordinates and texture coordinates, and then modified them with model manipulation to place them in my 3D world at the appropriate locations and angles. At first I tried to do everything with placement by just changing all the vertices in my arrays, but that was very time consuming and error-prone. Once I better understood the model transformations, I was able to make changes and try new things out much more efficiently!

**Explain how a user can navigate your 3D scene:**

A user can navigate my scene with both the keyboard, as well as the mouse/trackpad. By using the WASD keys, the user can move to the left or right, or backwards and forwards. When scrolling with the mouse, you can control the camera speed of the world, and by simply moving the mouse, you can move the object(s) around in the world from the viewers point of view. Additionally, the user can use the “P” key to change from a perspective to an orthographic view of the world. I was able to program this relatively simply by hooking up the Input keys and then querying them each time we went through the rendering loop. This is obviously useful for a user of my application, however, I also found it quite helpful when I was debugging and troubleshooting object rendering and placement. For future development, one could also modify this code to handle other input devices, such as a phone with a gyroscopic sensor, or a touchscreen instead of a mouse/trackpad.

**Explain the custom functions in your program that you are using to make your code more modular and organized:**

I have a few good examples here of where I extracted out logic to its own method to allow it to be more modular, organized, and extensible in the future.

1. The “UCreateTexture” function takes a file location and texture ID, and then loads the file into a texture that can be used by OpenGL. I have about 7 different texture files, so having a singular method that I can just call repeatedly definitely makes this easier.
2. The “UDestroyTexture” function – similar to above, this one handles destroying the textures
3. The “UCreateMesh” function – I put all of my vertex and VAO/VBO creation and initialization in this function. This means I only have one spot I need to look at in order to setup all my vertices, instead of all over the place
4. The “UProcessInput” function – this handles all of the user input from the keyboard – so in my render loop, I can simply call that function and make the code more readable, rather than having all of that code in the main loop.

Overall, I feel this shows a good example of clean, modern, commented OpenGL code, that should hopefully be easy for anyone else to read or modify.