**CS330 Project 2 Reflection**

Dante Trisciuzzi

4.11.2023

CS330 - SNHU

Prof. Jeff Phillips

**3D Scene Development**

My approach to development of this scene was highly reiterative. Before getting creative and organizing the objects in the scene, I laid the groundwork for the OpenGL pipeline. My first steps were of course setting up the basic rendering essentials, like a window and shaders. From here, each step builds on the previous steps, spawning objects, lighting them, camera controls, etc. Each of these development phases also required refactoring and re-thinking about some of the previous components. A prime example of this is the lighting layout; I started with basic shaders that just took vertex positions. I added colors, normals, and UV mapping for textures. This process required constant rework of the shaders as well as the vertex generation pipeline I had developed. When lighting came into the equation, much of the work I did for colors and textures needed to be re-written. This process happened over and over for each component added to the system.

When the time came to actually get creative and start replicating the scene, I started my work on my whiteboard. I considered the shape and position of objects relative to the camera, and what each object looks like from the perspective shown in the image I chose. Once I started laying objects out into the scene space, I was very glad to have built out a modular mesh creation system because it gave me creative freedom. I made all the textures myself for these objects, a couple of them are just pictures I took; a static picture of my desktop, and a panoramic image for the cylinder I used to encompass the scene as a background (my skybox, so to speak). The remainder of the textures I created myself in MS-Paint. While the textures I made don’t quite look true to life, I think they get the job done pretty well.

Navigating my application’s scene is simple and intuitive, particularly if you have ever played a computer game with a mouse and keyboard before. Controlling the in-world camera can be performed using the W, A, S, and D keys for movement, the Q and E keys for adjusting the vertical position of the camera, mouse movement to control the direction, and the mouse scroll wheel to change the camera’s move speed. Lastly, one can use the P key to toggle perspective and orthographic mode. Though orthographic mode does not provide a great representation of the scene, since the encompassing cylinder is very large.

Due to the inherently re-iterative nature of the development process, I opted to modularize my code and apply several object-oriented programming practices. Most importantly I used an inheritance for my Mesh management, with a base mesh class and several implementations for each type of primitive shape I would be using. This also allowed me to use polymorphism, or treating different types of objects as a single type, by creating a vector of Mesh pointers, and adding a reference to each mesh created to it. This greatly simplified the rendering loop, while ensuring I could still modify the objects at runtime if needed. Additionally, I used classes and encapsulation for all the main components to the application; the camera, the shader program(s), even the OpenGL window all had dedicated classes and exposed only the necessary actions. During this process, I wrote many wrappers for complex functionality for all of the above classes, specifically the Mesh class. The Mesh class needed all the math hidden away from the programmer to ensure consistency. Lastly, I also wrote some helper functions in two distinct namespaces; Math, and Textures. These helpers just minimized the amount of code that would need to be written in the main application source file, like creating textures and assigning texture IDs for later use.

**Resources**

*GLFW: Reference*. (n.d.). https://www.glfw.org/docs/latest/modules.html

*Learn OpenGL, extensive tutorial resource for learning Modern OpenGL*. (2023). Retrieved April 11, 2023, from https://learnopengl.com/