**Project 1 Reflection**

Zachary Mohler

Southern New Hampshire University

CS330

Malcolm Wabara

October 13, 2022

**Justification**

For this project, and throughout the course, my most prominent resource was the *LearnOpenGL* website by Joey de Vries as linked in our weekly resources. A result of this prominent resource is that my formatting, framework, and methodology are derived from my own take on the tutorials provided by the website. I rarely found myself using the GitHub tutorials as I found them to be a crutch rather than a learning tool. I found it much more beneficial to my learning to write my own code from scratch based on *LearnOpenGL* tutorial’s structure.

Naturally, writing my own code from scratch allowed me to better understand the content of each lesson, and therefore develop my own style/omit certain things not pertinent to the project at hand. For example, the tutorials teach the use of lightmaps to simulate textural differences per-object. I opted to not implement lightmaps in this specific project, as all the objects in the scene are made up of one material, and therefore wouldn’t benefit from a lightmap. Another deviation from the tutorials is more technical, and not really visible in the final executable. The tutorials recommended implementing functions within the fragment shader code for creating multiple light sources. I opted to simply ‘hard-code’ the two point lights in the project, this is due to the project having a constant number of point lights. My per-object vertices might also be considered “excessive”. i.e., multiple shared vertices between multiple polygons in single objects, but I found this to be necessary to accurately map texture coordinates as well as normal coordinates for each face.

**User Controls**

The controls available to the user for navigating the scene are simple “FPS” style controls: the W, A, S, and D keys are used for spatial movement—that is—forward, backward, strafing side to side. The Q and E keys are used to move up and down in space. The mouse is used to move the camera’s direction, which is sometimes referred to as ‘mouse-look’ for simplicity. The P key is used to change between orthographic and perspective views. The Perspective view is the typical 3D view that we are used to seeing in 3D applications. The orthographic view is a 2D representation of the same view. It is interesting to have this as a feature to explore how different perspectives appear when the orthographic view is toggled.

**Modularity**

The project has many member functions to simplify certain aspects of creating the OpenGL environment for the final scene. The creation of each GLmesh for primitive types (cube, cylinder, plane, etc.) is handled by an individual function that creates the vertices and binds the buffers. Each callback function used for user input is also a function that can be called during rendering to manipulate the scene. Destructors for each of the meshes, programs and some other aspects of the program are modularized into functions for simplicity, and to apply to each mesh with a standardized call. As previously mentioned, there are more opportunities for modularization that I considered unnecessary for a project of this scale. The project is easily templated using the current modularization—I actually created a template project with a basic cube to start with on each assignment, so from experience, the modularization is great for reusability.