**Project Reflection**

Joseph Veneski

Department of Computer Science, Southern New Hampshire University

CS-330 Computer Graphic and Visualization

Professor Philip Enkema

October 15, 2023

**Project Reflection**

**Development Choices:**

In this project the goal was to recreate a three-dimensional scene in OpenGL using primitive shapes based on a photo. Several objects in the scene were simple to build with primitives such as the table represented as a cube object along with four cylinders for legs. The ground setting and the paper on the table were created with plane objects and another cylinder for the bug spray. The baseball cap is built by combining several primitives, a half-sphere is built for the main section, a cylinder is placed intersecting with the table, so it is cut off giving the illusion of a hat brim. Three tori are placed on the hat to represent the vent holes and a compressed sphere is placed on the very top for the button. This is the most complex object in the scene built using six separate primitives of three different types. The screwdriver object is represented using two cylinders for the handle and shaft, with a pyramid for the tip.

The pineapple tiki torch was more difficult than anticipated to build using primitives. A representation has been implemented for now using a thick cylinder for the base with a half-sphere on the top. The base has a white pyramid texture applied that gives an illusion to the real complexity of the tiki torch. This could be enhanced further by implementing normal mapping to give more depth to lighting on the surface rather than building actual pyramids all over the shape (De Vries, 2014).

Textures were applied to all objects in the scene with normal calculations for lighting. Besides the pineapple texturing discussed already, the hat object utilizes four separate textures for the different pieces (brim, base, vents, button) and the screwdriver uses two textures.

Project requirements stated that there must be two separate light sources of different colors. To adhere to this requirement the primary white light originally planned was used in combination with a blue-tinged light source off to the side. This secondary light source was inspired by the off-scene bug zapper that is lit up at night, and when utilized in the scene helps illuminate the table legs and provide ambience to the lighting.

**Navigation:**

A camera object with movement controls is utilized in the scene based on the tutorials from LearnOpenGL.com. Users can navigate the scene using the W, A, S, D keys in combination with the mouse for orientation. Additionally, the Q, and E keys can be used to move “up” and “down” in the scene. Keyboard inputs are grouped and handled under a processInput function and mouse events such as scrolling, clicking, and movement are handled in separate functions allowing high portability. Users can modify movement speed throughout the scene by using the mouse-scroll wheel. The camera perspective can be switched between orthographic and projection modes using the P key. For testing purposes, the primary light source also has movement controls similar to the camera by using the I, J, K, L, U, and O keys.

**Custom Functions:**

The program utilizes several functions for modularity such as initialization, input processing, texture loading, mouse-handlers, rendering, and clean-up functions. The largest function in the program is the render function which handles all the object placement and drawing calls. Each object (sphere, torus, pyramid, cube, cylinder) has its own class which handles generating its own array and buffer objects and vertex information (coordinate, normal, texture). By utilizing separate classes, primitives can very easily be added or removed from the scene which was a major boon to debugging and troubleshooting issues. The primitive classes utilized in this program can be ported to any other project with ease by including the relevant .h and .cpp files. In the main driver of the 3D scene has several components that can be reused such as the user input and mouse-handler functions as well as the main render loop. A user could simply take this program and modify the render loop and texture loading to draw a whole new 3D scene with their own shapes and textures reflecting the high reusability of this code.

**References**

De Vries, J. (2014). *Normal Mapping*. LearnOpenGL. <https://learnopengl.com/Advanced-Lighting/Normal-Mapping>