**Project Reflection**

Justin Starr

Department of STEM

CS 330 – Comp Graphic and Visualization

Professor Jeff Phillips

December 10, 2023

**Project Reflection**

The objects that I included in my 3D scene for the project were a pen, a baseball, a roll of duct tape, a Rubik’s cube, and a tube of Chapstick. The reason why I chose these objects is that they were all objects that I could include that would demonstrate my ability to add objects in a 3D scene and at least one of those objects (the pen) would be a complex object that is a combination of two or more primitive shapes (a cylinder for the body and a pyramid for the tip of the pen). I also selected these objects because it included a variety of different shapes, a pyramid, a cube, a sphere, and cylinders. This demonstrates that I can create more than one or two different shapes and helps give the scene some character and variety. I was able to program for the required functionality by implementing various functions that addressed each portion of the project. For example, to create the objects, I implemented UCreate functions that when called would create the specific shape and store that shape’s data in the Vertex Array Object. To apply textures, a UCreateTexture function was implemented that creates a texture by loading an image resource and assigning it to a specific texture variable. When each object is drawn, the textureId for that object is passed along to the vertex and fragment shader source to be applied to the object. Similar functionality was implemented for lighting. To achieve movement functionality, two forms of input are included and processed, keyboard and mouse input. First, keyboard input is processed through the UProcessInput function, and mouse input is processed with three different functions, UMousePositionCallback, UMouseScrollCallback, and UMouseButtonCallback.

Having the ability to navigate the created 3D scene is essential for being able to see that the objects are, in fact, 3-Dimensional. It also gives the user the ability to navigate the world and see each of the objects from all of their sides. To navigate the world, specific keyboard keys were assigned actions, in the UProcessInput function, that would control the virtual camera. ‘A’, ‘S’, ‘D’, and ‘W’ keys controlled the camera’s forward, backward, left, and right directional movements and the ‘Q’ and ‘E’ keys control the up and down movement of the camera. Similarly, mouse input is processed that controls the camera’s pitch and yaw and combined with the keyboard input, allows the user to navigate the virtual environment.

Custom functions were implemented in the program to make the code more modular and organized. Specifically, separate functions were used to create the various types of objects. There were four different types of objects used in the 3D scene, a cylinder, pyramid, cube, and a sphere. Each object has its own function to create the object, UCreateCylinder for a cylinder, UCreatePyramid for a pyramid, UCreateSphere for a sphere, and UCreateCube for a cube. Anytime I needed to create one of these objects I could call the same function for the object and simply pass in the object’s mesh ID that the function would use to create the specific object. This allowed me to not have to write a UCreateCylinder function more than once even though I used 4 different cylinders in my scene. Additionally, functions such as UCreateTexture and UDestroyTexture only had to be written once but were called multiple times to create the various textures that the program used and then to destroy them when they were no longer being used. Other files were included in the project such as camera.h and stb\_image.h. Camera.h handles functionality specific to the camera and stb\_image.h handles functionality specific to the texture resources. This helped to reduce the amount of code in the main, also helping to keep the project more modularized and organized.