**7-1 Final Project**

Amanda Purnhagen

Southern New Hampshire University

CS 330: Computational Graphics and Visualization

Professor Eugenio Rodriguez

October 15, 2023

**7-1 Final Project**

For my final project, I rendered a scene with a wooden floor, a cylindrical Bluetooth speaker, a Toy Story themed ball, a rectangular calculator, and a pencil. I chose these objects for my scene because I knew I could make them out of the basic shapes available in the assignment prompt. The wooden floor was made using a plane, the Bluetooth used a cylinder, the ball used a sphere, the calculator used a rectangle, and the pencil serves as my composite object and used a cylinder and a pyramid.

To create these objects, I sourced code from the module tutorials, the CylinderExample project, and the MultipleObjectsExample project. All of these played a key role in supplying me with the foundation that I built on top of. I included all necessary header files and added additional C++ files to my project folder and troubleshooted all the compatibility issues. I also sourced free stock images to use for the pencil’s textures.

From there, I created the necessary position vertices, lighting normals, and texture coordinates for each 3D object. I successfully added two lights sources, and they seem to be shining light. I believe the first light shows a green color. However, I had a hard time making sure the fill light's intensity was only 10%.

The user can navigate my 3D scene using the W, A, S, D, Q, and E keys as well as the mouse position and the scroll wheel. Each of these inputs changes the location of the virtual camera (the user’s view). For example, pressing the W key will move the camera forward. Pressing E will move the camera down relative to the user’s current view. The mouse position can be used to move the direction that the camera is pointing in, and the scroll wheel is used to zoom in and zoom out. Additionally, the P key can be used to alternate between an orthographic (2D) view and a perspective (3D) view.

The custom functions I created include additional UCreateMesh() functions. I admit that I could have made this part of the code more modular and organized. Originally, I was trying to put multiple vertex arrays contained within the same UCreateMesh() function, but I could not figure out how to distinguish between them when subsequently attempting to render the separate arrays in my URender() function. In order to complete this project on time, I ended up creating multiple UCreateMesh() functions with different names. Other than that, the same functions exist with some modifications to meet the criteria for my 3D scene. Ideally, I would have liked to make the code far more modular and organized to promote the reuse of functions to complete common tasks.