### 330 Final Project

For my 3D scene, I wanted to choose shapes that were simple and common in everyday life. The knife I used is a triangle attached to a cylinder, the speaker is a simple cube, and the Pez container is a rectangular prism with a sphere on top. These shapes are representative of everyday objects that can be found in any home, and I thought they would provide a good variety of shapes to use. Some of the shapes however, proved very difficult to recreate. Creating shapes in OpenGL is done by combining three points in space and connecting them in the form of a triangle. This triangle is added to as many other triangles as necessary to create its final shape. For instance, a rectangular prism has six surfaces, but each surface consists of two triangles. This means that in order to create rounded edges, many different triangles have to be connected at various angles. There are formulas and available scripts to help accomplish this, but for the ones that I found, I was unable to both create the sphere or cylinder and position it in the appropriate position. In the end, instead of using a cylinder for the handle of the knife and a sphere for the Pez, I represented the handle of the knife with a rectangular prism and the top of the Pez container with a pyramid. The plain that the objects rest on and the speaker were easy and simple to create. The entire scene consists of forty-seven different triangles.

The scene is fully navigable and is able to be viewed from any angle. Both the speed of the camera and the perspective of the scene can be changed. The camera is tied to a listener that checks and listens for button input or mouse movement. The process engine then relays this information to the renderloop, moving the camera for each new render. The listeners I have set for the keyboard are W and S to move forward and backward, A and D to move side to side, Q and E to move up and down, P to change perspective. On the mouse, the wheel controls the speed of the camera and the x and y access controls the pitch and yaw of the camera.

In this program, there are many modular components that make changing and adjusting the code an easy process. One useful and reusable function is the UInitialize function. This allows the program to build the window, any necessary data, and set up any listeners needed for other functions. Another useful function is the UProcess input function. This function takes the information from the listener and can apply commands based on keypress information. A developer can easily add or remove commands for different keys, making this function reusable. The next modular function in this program is the URender function. This function houses building and creation of the meshes. It also controls transitions, textures, and lighting details for each object rendered. This data is passed to the shader and the shader applies it to the mesh. Transitions and other data can be adjusted and lights can be added as long as the shader can apply it. UCreate mesh is a function that contains a list of points and a list of triangles. It then uses this data to create meshes to be used by other functions. UCreate texture is a function that helps to implement textures and apply effects such as wrapping and tiling. This can be used to manipulate textures in different ways if called multiple times. The last function is the UCreate shader program. This allows the program to easily create multiple shades for multiple objects. A developer can use this to create custom shaders for objects.

Altogether most of the program ran the way it was intended. The biggest issue I encountered was how to apply different textures to different triangles. There were many other issues along the way, but after lots of reformatting and experimenting, most were solved. I’ve learned a lot about creating graphical content and how to apply effects. In the future, I hope to hone my skills and better understand how to create and develop 3D graphics.