**Final Project Reflection**

**Justify development choices for your 3D scene**.

Initially, the idea was to try and recreate my work-space in 3D but, that was before clearly analyzing everything that would be involved with creating a 3D scene in OpenGL. My workspace has complex items, many of which would require 2 to 3 primitive shapes to form the 3D replica. So, I decided to go back to the basics and look for objects that look like primitive shapes. The selected objects in the 3D scene are naturally shaped like primitive shapes. The orange is a sphere, the roll of tape is a torus, the case is an elongated cube, and the jug is a combination of a cylinder and a pyramid. In this case instead of a pyramid a tapered cylinder was used because it was easier to code and produced the desired effects.

I was unable to incorporate the handle for the gallon container, I tried a leaning cylinder, but I could never get it to lean like the handle. The role of tape did give me issues with positioning it in the right place as well as orienting it in the right way. The torus produced was upright and the render would not lay it down to a 180 degree angle. The cap on top of the gallon container kept getting rendered inside the tapered cylinder, this was an issue with the translation of the cap. Texturing the gallon container posed some issues because the gallon in the 2D picture is white and finding a white texture that would actually stand out like texture and not just look like a white colored container was challenging.

**Explain how a user can navigate your 3D scene**.

Navigating the 3D scene is easy because of the camera controls added to the scene. The camera controls in the 3D scene involve a combination of keyboard and mouse functions. On the keyboard, the keys WASD are used for directional control and zoom. W is to zoom out, S is to zoom in, A is to go left, D is to go right. The Q and E buttons move the camera up and down respectively. These key functions WASDQE, are invoked in the UProcessInput function of the code (*line 363*). The mouse scroll button also zooms in and out and the mouse movements move the camera around the 3D scene. These mouse movements are controlled in three functions, the UMouseScrollCallback function (*line 484*) which is responsible for zooming in and zooming out, the UMouseButtonCallback function (*line491*) which controls the button and action values used by the function and the UMousePositionCallback function (*line 463*) which controls the X and Y screen coordinates whenever the mouse is moved.

**Explain the custom functions in your program that you are using to make your code more modular and organized**.

Shaders are one of the types of functions used in this program. Their main purpose is to facilitate the ability to draw on a screen using coordinates. They use GLSL, an Open GL shader language that runs on a GPU (graphics processing unit). Some of the custom functions in use for this program include the Vertex Shader Source 1, Fragment Shader Source 1, Lamp Vertex Shader, Lamp Fragment Shader just to mention a few reusable functions. All the shader programs are running the GLSL version 440. The Vertex Shader Source 1 is a function used in the program to map out shapes in 3D drawing coordinates. This function uses information from the vertex data to position 3D drawing coordinates in our environment. The meshes.cpp file provides the information needed by the Vertex Shader Source 1 program for this project.

The Fragment Source Shader 1 function is responsible for rendering, coloring and texturing of the shapes and or vertices in the 3D scene. In this project, the Fragment Shader Source 1 function initializes the Phong lighting method in its main function during the function’s entry point (*line 151*). Phong lighting is a combination of ambient, diffuse and specular lighting. A Lamp Shader Vertex function is also used in this program to calculate the light position in 3D drawing coordinates. The lamp or light is also an object in our environment that needs to be rendered, positioned, have buffers created for it and all. The last function I will highlight used in this program is the Lamp Fragment Shader. This is used for the light color. These four functions can be easily used to create any different 3D scenes with just the exception of renaming variables. The main function (*line 236*) is the entry point for Open GL and here is where the code tells the program to create the shaders, as well as the meshes, loading and setting any textures to be used, activating the program using: glUseProgram(gProgramId1);.