**Name: Fabian Weiand**

**Date: 6/20/21**

**CS 330 Module Seven – Final Project Reflection**

In creating the 3d scene using openGL for this class I initially set out to render a coffee table with a tea set including teapot, teacup, and saucer. As I continued development, I found I had to change my plan slightly. The teapot proved too complex to develop without more time. Instead, I opted to omit the teapot in favor of a carpet that is laid under the coffee table. In total there are 6 objects in my scene: The plane upon which everything rests, the carpet, coffee table, saucer, teacup, and two windows to provide light. I chose these objects because I felt confident in creating the necessary vertex data and they fulfilled the required functionality of the assignment. One requirement was to use 4 of the primitive shapes within our scene. The carpet is a plane, the table is made up of connected cubes, the saucer represents the very base of sphere, and the teacup is a cylinder with a half torus handle. Another rewuirment is that there would be two light sources of different colors. One window emits a white light, while th other emits a red light.

Navigating around the 3d scene is simple. You cand use the W, A, S, and D keys on the keyboard to transport the camera view forward, left back, and right respectively. You can also use the Q and E keys to move the camera up and down vertically. In addition to these movements, the camera view can also be adjusted by using the mouse to change the view angle and the mouse wheel can adjust the sensitivity of how quickly this view angle changes based on mouse input. One addition functional requirement is to be able to switch between orthographic (2D) and perspective (3D) views at will. This can be achieved via the P key on the keyboard.

To keep my code more organized and limit the amount of code redundancy I opted to created several custom functions. One such function sets the uniform variables of the shaders in my code called setShaderPrograms(). This made it easier to adjust the specific locations of the various objects on the fly. Another custom function I implemented was the createTexturePrograms() function which created and bound textures to specific shader programs allowing to more easily se several textures and shaders in my scene.