// SNHU

// CS-330: Computational Graphics and Visualization

// Student name: THOMAS SEIBERT

// MOD 7: Final Project – Design Decisions

1. **Justification of development choices for 3D scene**

In terms of why I chose the selected objects for this scene, they (these objects in particular) did initially meet the assignment rubric. Conveniently, these objects were located within roughly 15 feet of my computer desk, and so therefore the choices weren't difficult. I did, however, simplify the building of the 3D cup object from three shapes to two shapes. Initially when I took the first photo of the scene, I was under the assumption that it would be the most “complicated” shape, but that turned out to be not the case.

I was able to build the program up to the required functionality by studying LearnOpenGL's tutorials, along with the other module resources which were provided.

1. **How a user can navigate this 3D scene**

A user can navigate the scene by using both a mouse and a keyboard. Pressing certain buttons on the keyboard allows the user to navigate around the scene using the perspective of the camera. For example, the 'W' button allows the user to move their camera perspective forward within the scene, while the 'Q' button allows the user to move their camera perspective upward, the 'E' allows for downward movement, while other keys allow for movement in other directions.

When the scene opens in a new window, the mouse becomes locked within the center of the window, which allows for more perspective changes using the mouse. For example, if the user moves the mouse forward, the camera perspective looks upward, and the same is true of any other direction. If the user moves the mouse backward, this shifts the camera perspective downward.

Overall, using a combination of both the mouse and the keyboard allows the user to fully explore the scene.

1. **Custom functions in this program to make code more modular and organized**

Many of the customized functions in this program are located within the SceneManager class. For example, the function DefineObjectMaterials() allows for the configuration of the various objects' material settings for every object within the scene. Within this function, properties such as "ambientStrength" and "shininess "were utilized. The property known as "ambientStrength" defines how much ambient light is emitted, and "shininess" tells how shiny the material is and how much specular light will be reflected off the surface.

Another customized function titled SetupSceneLights() allows for both the addition and configuration of light sources in the scene. Properties in this function were used to define the attributes of a particular light source. For example, "ambientColor" defines the color of the ambient light being emitted, while "focalStrength" defines the focus of the light beam being emitted.

Then, there is the customized LoadSceneTextures() function, which allows for the addition and loading of various textures which are to be mapped to objects within the scene. In this method, I utilized the CreateGLTexture() method and added unique textures that I found through online searches as a parameter to this method. This method allows a tag to be associated with each texture, which made the process of utilizing textures much quicker.

Finally, there is the RenderScene() function where the shapes are drawn, and the scene itself is rendered. This function is highly customizable in that any developer can add, change, and place any shape available within the given parameters of the program. The most common shape that I utilized in the scene that I built was the cylinder.

Every function that I have listed above, and their properties are reusable in the sense that a developer can, with sufficient learning and practice, quickly swap out one parameter for another, particularly in the case of the CreateGLTexture() method.