**Final Project Reflection**

Matthew A. Keaton

Southern New Hampshire University

CS 330: Computer Graphics & Visualization

Professor Mike Prasad

June 22nd, 2025

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

When building my 3D scene, I wanted to create something realistic that looked like a real desk setup someone might use in an office or at home, so I chose to replicate my own personal desk setup. I chose commonly used items like a small desk, laptop, mouse, books, a lamp, and a coffee mug because they are familiar and help the viewer feel like the scene is real. Each object also gave me a chance to use different 3D shapes like boxes, cylinders, cones, and spheres, which helped me use more than just one shape and implement multiple others. I placed the books on the back right to add depth and balance, while the coffee mug in the front left adds some attention to that side of the scene. I used rotation, scaling, and translation to move and resize all the objects to make sure they looked like they belonged in the scene. I also used different colors and textures to make each item stand out. For lighting, I added two light sources so the objects would not look flat or have dark spots. One light is a yellowish overhead light and the other is a bluish light that fills in the shadows. All these choices made to add in my program ensured that my scene met all the requirements and gave me full creative control over how each object interacted with the light and environment.

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

To make the scene more interactive, I set it up so users can move around and look at everything from different angles. I used the keyboard and mouse to control the camera. The W, A, S, and D keys let the user move forward, left, backward, and right, like in a video game. The mouse allows the user to look around by moving the camera view up, down, left, or right. This makes it feel like you are inside the 3D world and can explore it naturally. I used built-in camera tools from the GLM library to control how the camera moves and where it looks. This was important because I wanted the user to be able to see every object, no matter where it was placed or how the lights hit it. This also helped me when debugging my code, I was able to move around the environment to get a better view of how the objects fit into the scene or if their positions needed to be changed. By allowing the user to move and look around, they can get a full view of how the lighting and shapes work together, which helps show off the result better.

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

To keep my code clean and easy to work with, I added a few custom functions that helped me organize everything and avoid repeating the same lines over and over. One function is SetShaderTexture(), which makes it simple to apply textures to different objects in the scene. Instead of writing all the OpenGL texture code every time, I just call this function and pass the name of the texture I want to use. This keeps the rendering code for each object short and easy to read. Another function is DefineObjectMaterials(), where I set up material properties like ambient, diffuse, and specular colors for specific objects. This makes it easy to change how certain objects reflect light without changing the shader or repeating material settings throughout the code. Lastly, SetupSceneLights() is the function I use to define the light sources in the scene. This lets me set the position, color, and strength of each light. I used this function to add two lights, one warm with a yellowish color and one cool with blueish color, to make the scene look more realistic. These custom functions helped me stay organized and made it easier to test and update different parts of the project.