Tanner Hunt

June 12, 2025

CS330 Computer Graphics and Visualization

Final Project Reflection

I chose objects for my scene that I thought would both showcase the rendering technology being built and be realistic to build within the timeframe of this project by using primitives to build a more complicated scene. The glass mug had the opportunity to showcase reflections, the lamp had the opportunity to showcase objects built with multiple primitives, the pencil case had the opportunity to showcase lighting of densely populated objects, and the desk stand had the opportunity to showcase texturing and make positioning objects more interesting. Much of the required functionality was prebuilt, such as the window manager, camera, and shader scripts. To add functionality, I wrote code that translated, rotated, and scaled primitive objects to represent more complex objects, empowered the user to navigate the scene, textured objects, created materials, and added lights to the scene. One approach I considered during early development was building data structures that automatically managed transformations and coloring of objects. This strategy made prototyping the scene quicker but would be difficult to maintain after the remaining features were built.

The user can navigate throughout the scene from the camera using the W, A, S, and D keys to move in relation to the camera’s forward vector. Additionally, the camera can pan superiorly and inferiorly to the camera using the Q and E keys. The user can also use the mouse to angle the camera in the direction they want to move. The mouse has been locked to the viewport to make navigation easier. Without this functionality, the camera stops rotating as soon as the user moves the mouse outside of the window. The user can use the escape key to free their mouse and close the program. The user can also change the projection to orthographic using the O key, or perspective using the P key. Finally, the user can use the scroll wheel to increase or decrease the speed the camera flies through the scene. These controls use inputs from the GLFW library. The benefit of this is that this control paradigm becomes extensible to other controllers. For example, panning could be rebound to a joystick controller using the GLFW joystick commands. By using an extensible library, these graphics could be implemented in a diversity of projects, like computer gaming, animation,

The code has been made more modular by abstracting events into methods with clear naming conventions. For example, primitive shapes are transformed to represent a mug in a method called makeMug(). Because many primitives are programmatically built into the scene, separating the creation of these objects makes the code easier to read and modify. Another example of this are the methods associated with material and light creation, which creates clear expectations of the functionality of certain lines of code. One way the code could become more modular and reusable would be to transform objects and apply textures automatically through an object manager. Manually moving and texturing thousand of objects through code would be unfeasible in a larger project.