Austin Antles

CS 330

June 22 2025

**Reflection**

The objects in this project (specifically, a desk, a book, a globe, a mug, and a pyramid) were chosen for their geometric diversity. The goal was to create a realistic and familiar scene while demonstrating a wide range of 3D modeling skills. The flat desk provides a foundational plane, the book a simple box, the pyramid its unique angular shape, and the mug combines a cylinder and a torus. The globe, made of a sphere and a cylindrical stand, serves as a more complex object built from multiple primitive shapes. This variety allowed for a comprehensive application of texturing and lighting. Programming the scene followed a clear, repeatable pattern for each object: first, its position, rotation, and scale were set using transformations; second, its appearance was defined by applying a specific texture and material properties for shininess and reflection; and finally, the command was given to draw the object's mesh. This "transform, style, draw" process made it straightforward to build the scene piece by piece.

A user can explore the 3D scene using standard keyboard and mouse controls, similar to those found in many modern 3D applications and games. The keyboard handles all movement through space: the W, A, S, and D keys move the camera forward, left, backward, and right, while the Q and E keys move it vertically up and down. For more nuanced control, the mouse is used to change the camera's orientation. Simply moving the mouse cursor allows the user to look up, down, left, and right, as if turning their head. Additionally, the mouse scroll wheel directly controls the camera's movement speed, allowing users to slow down for precise inspection of an object or speed up to travel quickly across the scene. This control scheme was programmed by capturing input from these devices and using that data to call methods in the Camera class, which handles all the underlying calculations for movement and viewing angles.

To keep the project organized and easy to manage, the code was broken down into several key, reusable functions. The PrepareScene() function, for example, acts as a one-time setup routine that loads all necessary 3D models and textures into memory before the main rendering begins. This is efficient because it prevents reloading assets every frame. The lighting was consolidated into a dedicated SetupSceneLighting() function, which configures all light sources, their colors, and intensities. This makes it incredibly easy to change the entire mood of the scene, from daylight to sunset, for instance, by only modifying one function. Finally, the main RenderScene() function is called every frame and is responsible for drawing all the objects. By keeping the setup, lighting, and drawing logic separate, the code is highly modular. You can completely change the objects displayed in the scene by only editing RenderScene(), without ever touching the code that controls the lighting or camera.