Paulina Weaver

Professor Enkema

CS330

04/19/2025

7-1 Final Project Reflection

For my final project, I chose a candlelit dinner scene because I thought it would be both visually interesting and a great way to show off different 3D elements. This setting gave me the chance to work with a variety of shapes, textures, and materials, like the smooth surface of the table, the transparency of wine glasses, and the warm glow of candlelight. I included objects like a table, two chairs, a wall clock, a wine bottle, a candle, wine glasses, plates, and forks to create a cozy and balanced scene that feels realistic and complete. The scene was built using a range of shapes, including planes, boxes, cylinders, tapered cylinders, cones, spheres, and prisms.

One of the most important design decisions I made involved lighting. Initially, I used a point light for the candle, but it didn’t look right because it was illuminating areas it shouldn’t, especially underneath the table. It created the illusion that the candle was glowing from below rather than casting a soft, realistic light around it. I later switched to a spotlight, which gave me more control over the direction of the light and allowed me to simulate the candle’s glow more realistically on the table. I also added a flickering effect using a sine wave based on glfwGetTime() which gave the candle a more lifelike feel and helped the entire room feel dynamic rather than static. I used different light types (directional, point, and spotlight) to simulate natural and artificial lighting. This trial-and-error process really helped me understand what worked visually and what didn’t.

Users can navigate the scene using their mouse, scroll wheel and keyboard, allowing them to explore the scene from any angle. The WASD keys are used for zooming in (W), zooming out (S), and panning left (A) and right (D). The QA keys control vertical movement, allowing users to move up (Q) and down (E). The user can toggle between orthographic and perspective views using the "O" and "P" keys. Mouse movement lets users look around in any direction, and the scroll wheel adjusts movement speed for smoother navigation.

Keyboard input is handled in the ProcessKeyboardEvents() method, which processes all key-based controls using GLFW callbacks. For mouse movement, the Mouse\_Position\_Callback() method tracks the cursor's position, adjusting the camera's orientation accordingly. Scroll input is handled through Mouse\_Scroll\_Callback(), which adjusts the camera's speed dynamically between 1.0f and 100.0f. This ensures smooth, controlled movement throughout the scene, preventing either overly slow or excessively fast camera navigation. The orthographic and perspective projections also update the camera’s position and angle to provide a proper viewing experience in each mode.

To keep my program clean and easy to manage, I created several custom functions that grouped related tasks together. For example, SetupSceneLights() manages all the lighting properties in one place, making it simple to adjust light color, position, or intensity without digging through unrelated code. Similarly, DefineObjectMaterials() is responsible for setting up the material properties for each object, allowing me to fine-tune details like shininess and transparency without interfering with rendering logic.

Each object in the scene has its own rendering function, like RenderTable(), RenderWineBottle(), and RenderWallClock(). This not only keeps things neat, but also makes it easier to isolate bugs during development. For example, when I was testing transparency for the wine glasses, I only needed to modify RenderWineGlass() without affecting the rest of the scene. This modular approach also made it easier to reposition or rescale objects individually and would support future reuse in other projects.

Overall, organizing the code this way has made the project easier to debug, extend, and manage as it grows. It also sets me up for future improvements, like adding animation or interaction, since the structure is already in place to handle more complexity without overwhelming the rest of the codebase.