Wurzburg Brumaire

Philip Enkema

CS-330-12507-M01

10 December 2024

7-1 Final Project Reflection

For this project, I chose to replicate a desktop workstation as the 3D scene. This concept felt fitting because I was working on a computer desktop while creating a digital representation of one. The scene includes a desk, a computer screen, a keyboard, a mouse, and a desk lamp elements that together form a functional workstation. This choice allowed me to explore the interplay between textures, materials, and lighting, while also adding a layer of relatability and immersion to the scene. Each object was carefully chosen to demonstrate key modeling techniques and highlight functionality within the program. For example, the desk was modeled with a wood texture to give it a natural appearance, while the computer screen included a sleek material and texture that simulate its modern design.

To ensure interactivity, I implemented a virtual camera that allows users to navigate the scene seamlessly. The keyboard and mouse provide control over the camera's movement and perspective. Users can move forward, backward, left, and right using the W, A, S, and D keys, and adjust the vertical position with Q and E. Mouse input allows for smooth adjustment of the viewing angle, while the scroll wheel enables zooming in and out. This interactivity was achieved through the integration of a Camera class and custom callback functions, such as Mouse\_Position\_Callback and Mouse\_Scroll\_Wheel\_Callback. These functions process input data to update the camera's position and orientation dynamically, creating an intuitive navigation experience.

The development of custom functions played a key role in making the program modular and efficient. Functions like SetTransformations(), SetShaderTexture(), and SetShaderColor() were designed to handle specific tasks and to be highly reusable. For example, SetTransformations() applies scaling, rotation, and translation transformations to objects based on parameters, ensuring flexibility for different components of the scene. Similarly, SetShaderTexture() simplifies texture assignment by using tags, allowing me to decouple texture management from object rendering. These modular functions reduced redundancy in the code and made it easier to modify or expand the scene. The RenderScene() function ties everything together by calling individual render methods like RenderTable() and RenderComputerScreen() in an organized way, ensuring clarity and maintainability.

This project offered a balance of technical and creative challenges. The focus on modularity and efficient code management allowed me to streamline the rendering process while keeping the program extensible for future additions. The interactive navigation system enhances user engagement, making the scene both functional and visually appealing. Overall, this project deepened my understanding of computational graphics and reinforced best practices in programming for complex systems. It was a rewarding experience that combined creativity, problem-solving, and technical expertise to bring a 3D scene to life.