Katelynn Thompson

Sherin Aly

CS-330

08/22/24

Final Project

When creating my 3D scene using OpenGL, I carefully selected objects I see or use every day, as well as showing off some of my personality and aligning with the practical aspects of 3D modeling and interaction. The chosen objects—a laptop, my Red Bull can, frog planter, headphones, and computer mouse—serve dual purposes: they are everyday items that I use, enhancing the relatability of the scene, and they provide a range of surfaces, shapes, and materials to demonstrate and practice my skills in 3D modeling and rendering.

The selection of these objects was driven by their different shapes, complexities, and textural requirements. For instance, the computer mouse, with its curved body and buttons that wrap around the sides, presented a challenge in achieving smooth, organic shapes, contrasting with the laptop's more angular and straightforward form. This combination allows me to showcase a diverse skill set, from managing simple, flat surfaces to more complex, curved geometries. Furthermore, the frog planter introduces an element of organic, natural texture, which contrasts with the technological elements of the scene, creating a balanced composition that highlights both artificial and natural aesthetics.

Programming the required functionality for this scene involved several key considerations, particularly regarding how users interact with and navigate the environment. The scene's navigability was a critical aspect of its design, ensuring that users could seamlessly explore the virtual space and appreciate the details of each object. I implemented a virtual camera system that allows users to pan, zoom, and orbit around the 3D scene using keyboard and mouse inputs to achieve this. The WASD keys are used for horizontal and vertical movement, while the QE keys allow upward and downward movement, providing a full range of motion. The mouse cursor controls the camera's orientation, enabling users to look around the scene, while the mouse scroll adjusts the movement speed, allowing for finer control during close inspections or faster traversal when needed.

This setup was carefully chosen to provide an intuitive and responsive user experience. The decision to use the WASD keys, which are familiar to many gaming users, was made to reduce the learning curve and ensure fluid navigation. The mouse input for camera orientation was implemented to mimic real-world head movement, where the gaze's direction changes with the head's movement. This natural input mapping to camera control enhances user immersion, making the virtual environment more interactive and engaging.

To ensure my code remains modular and organized, I developed several custom functions that encapsulate specific tasks, making the codebase more manageable and reusable. One such function is the update camera position, which handles all aspects of camera movement based on user input. This function takes in parameters for direction, speed, and current camera orientation and calculates the camera's new position in the scene. By isolating this logic in a single function, I can easily modify how the camera behaves without altering code in multiple places, thereby reducing the risk of errors and simplifying future adjustments.

Another custom function is applyMaterialProperties, which I use to configure the material settings for different objects in the scene. This function takes parameters such as diffuse, specular, and ambient lighting coefficients and applies them to the selected object. This modular approach allows me to reuse the same function across multiple objects, ensuring consistent application of material properties while allowing for easy tweaking of individual object appearances. For example, I used this function to differentiate the glossy plastic material of the mouse from the matte ceramic texture of the frog planter.

By structuring my code in this way, I make it easier to manage and ensure that it is adaptable to future changes or extensions. This modular approach is particularly beneficial when working with complex scenes where multiple objects and interactions must be managed simultaneously. It allows for clean, readable code that can be easily debugged and expanded upon, which is crucial in developing sophisticated 3D environments.

In conclusion, my development choices for this 3D scene were guided by a desire to create an accurate environment for my work and gaming desk. The selected objects provided diverse modeling challenges while implementing a flexible and intuitive navigation system ensured that users could fully engage with the scene. Using custom functions to organize my code further enhanced the maintainability and scalability of the project, positioning it as a robust example of 3D scene development in OpenGL.