Elijah Harris

CS-330

Gray

6/22/2024

1-7 Final Project Reflection

Embarking on the development of a 3D scene with proper lighting and material properties was both an exciting and challenging journey. Throughout this process, I had the opportunity to apply theoretical knowledge gained from coursework in a practical setting, while also honing my problem-solving skills. This reflection delves into the various stages of development, the challenges faced, and the custom methods that helped me through it.

The initial stage of the project involved setting up the scene and creating the basic shapes required for the desk setup. This included loading meshes for the plane (desk), box (monitor, keyboard, and PC tower), torus (PC tower) and cylinder (mouse). Each of these shapes had to be carefully positioned, rotated, and scaled to create a realistic desk scene. Adding textures to the objects was the next step. This involved loading texture images and mapping them to the corresponding objects. At first my paths were started from the C drive, but I got feedback that relative paths would work better for sharing the code. I started with a simple ambient light to provide a base level of illumination. Understanding the Phong lighting model was essential for this stage. I learned how ambient light provides uniform lighting, while diffuse and specular lights add depth and realism by simulating how light interacts with surfaces.

One of the most challenging aspects was fine-tuning the lighting parameters. This involved adjusting the position, color, and intensity of the lights to achieve the desired effect. For instance, the overhead light needed to provide global illumination, while the monitor light had to be directional to simulate the screen emitting light. Fine-tuning these values required a lot of trial and error, making small adjustments and visually comparing the results. This iterative process highlighted the importance of patience and persistence in achieving the right balance. Making the monitor screen appear as if it was emitting light was particularly challenging. Initially, the light was affecting the wrong parts of the scene, leading to an unnatural appearance. Through research and experimentation, I learned how to use directional light properties effectively.

Users can move through the scene using a combination of keyboard and mouse inputs. The keyboard controls allow for movement in all directions: forward, backward, left, right, up, and down. Specifically, the W and S keys move the camera forward and backward, while the A and D keys allow for strafing left and right. Vertical movement is controlled by the Q and E keys, moving the camera up and down respectively. Mouse inputs are used to control the orientation of the camera. By moving the mouse, users can look around the scene, simulating a free-look perspective commonly found in first-person navigation systems. The camera's yaw and pitch are adjusted based on the mouse movement, allowing users to look left, right, up, and down. The scroll wheel is used to adjust the movement speed, making navigation faster or slower as needed.

The ProcessInput function is responsible for handling keyboard inputs to control the camera's movement within the scene. This function is called every frame to ensure that the camera responds to user inputs in real-time. It checks for key presses and updates the camera's position based on the direction keys pressed (W, A, S, D, Q, E). This function makes it easy to add or modify camera controls without affecting other parts of the code. The mouse\_callback function handles mouse movement to control the camera's orientation. By calculating the offset of the mouse movement, this function updates the camera's yaw and pitch angles, allowing the user to look around the scene.