Kennedy Uzoho

CS-330 Final Project Reflection.

Southern New Hampshire University.

6/20/2023

CS-330 Final Project Reflection Submission.

Development Choices for the 3D Scene

In developing my 3D scene, I made several choices based on the desired functionality and visual requirements. Here are the justifications for some of the selected objects in the scene.

Table Plane: I decided to include a plane-shaped object to create a realistic table shape that will hold all the other 3 objects in the scene. The cup, laptop, mouse, and two lamps were all sitting on top of the plane-shaped object (Table), which provides a natural ground surface for the other objects in the scene and enhances the overall visual appeal. It also allows for more diverse camera viewpoints and exploration.

Laptop Box: I chose to include a laptop-shaped object as one of the main objects in the scene because it is a simple geometric shape that provides a visually interesting element. The laptop shape adds depth and dimension to the scene and serves as a focal point for the user.

Cup Cylinder: I decided to include a tapered cylinder-shaped object to create a realistic cup-shaped 3d object. The cup provides a realistic environment for the other objects in the scene and enhances the overall visual appeal. It also allows for more diverse camera viewpoints and exploration.

Mouse Sphere: I incorporated a sphere and box-shaped object to complement the entire scene. The mouse adds realism and depth perception by creating the illusion of a vast scene of a table with items on top of it. It enhances immersion and visual experience for the user.

Programming for Functionality

To achieve the required functionality in the 3D scene, I implemented various features such as:

Camera Control: I programmed the virtual camera to enable user navigation in the 3D scene. The camera can be controlled using input devices like the mouse or keyboard; the ‘A’ key left to move left, the ‘D’ key to move right, the ‘W’ key to move up, the ‘S’ key to move down towards the screen, and finally I added extra camera control options that let the user move the camera upward and downward using ‘Q’ and ‘E’ key respectively. The user can pan, zoom, and orbit around the objects to explore the scene from different angles. The mouse cursor can be used to change the orientation of the camera to either look up or down. The mouse scroll can be used to adjust the speed of movement.

Lighting: I implemented lighting techniques, such as ambient, directional, and point lighting direction to enhance the visual appearance of the objects in the scene. The lighting can be adjusted dynamically to create different moods and atmospheres.

Texturing: I incorporated texture mapping to apply realistic textures to the objects in the scene. This adds detail and visual richness to the objects, making them more visually appealing and immersive.

Custom Functions for Modularity and Organization

In my code, I have developed custom functions to make the code more modular and organized. Here are a few examples:

CreateMesh(): This function is responsible for generating a mesh object with the specified vertex positions, texture coordinates, and indices. It encapsulates the process of creating a mesh and allows for easy reuse when generating different objects in the scene.

LoadTexture(): This function loads a texture from a file and returns the corresponding texture ID. It abstracts away the detail of loading textures and provides a reusable function to easily load textures for different objects in the scene.

SetCameraPosition(): This function sets the virtual camera's position in the 3D scene. It allows for easy positioning of the camera at different locations within the scene, providing flexibility and control over the camera's initial position.

These custom functions enhance the modularity of the code by encapsulating specific tasks into reusable functions. They promote code organization, readability, and maintainability, as they can be easily understood and modified independently of other parts of the code. Additionally, these functions contribute to code reusability, allowing for the efficient development of additional objects and functionality in the scene by leveraging the existing modular functions.

Perspective and orthographic displays of the 3D world: I implemented functions to handle switching from perspective view to orthographic view, which enhanced the 3D world and made it more flexible to visualize the objects in the scene. You can use the 'O' key to switch to an orthographic view and the 'P' key to switch to perspective respectively.

Thank you for reading.

Kennedy Uzoho