Assignment 7-1

Jarrod Thomisee

Jarrod.thomisee@snhu.edu

2023-10-15

**Justifying Development Choices for the 3D Scene:**

The main objective when developing this 3D scene was to create a realistic virtual representation of a computer desk setup. The objects and their chosen shapes were intentional:

**Cylinders**: They were aptly used for the base of the monitor and the mouse, emulating real-life design where rounded bases often provide stability for monitors and a cylindrical, elongated shape is ergonomic for a mouse.

**Cubes**: The computer, monitor screen, and keyboard utilize cube shapes because these items tend to have rectangular profiles. The use of cubes simplifies their representation while still capturing their essence.

**Plane**: A plane was a natural choice for the desk. It provides a flat, expansive surface upon which the rest of the items can be placed.

**Torus**: This was incorporated to represent green case lights in the computer. This gives the computer a modern feel, as many custom PC builds nowadays have circular RGB lights for aesthetics.

**Navigating the 3D Scene:**

The navigation mechanism for this scene is both intuitive and user-friendly:

**WASD and Q/E and R**: By pressing the WASD keys, users can move forward, backward, left, or right. Additionally, the Q and E keys allow for vertical navigation. This kind of navigation is common in many 3D applications and games, making it familiar to many users. In the case that the user wants to reset to the initial view that they started with, they can reset by pressing R.

**Mouse**: The user can use the mouse to look around, enabling them to get a panoramic view of the entire setup, ensuring no detail goes unnoticed. This kind of first-person view control is again very intuitive for users familiar with 3D environments.

**Mouse Scroll**: Using the scroll callback, users can adjust the camera speed using the mouse wheel. This offers a way to fine-tune movement speed, making the navigation smoother and more precise.

In addition, the scene is designed to be responsive. If the window or screen is resized, all the images within the scene will adjust dynamically to fit the new dimensions.

**Custom Functions:**

Having custom functions in the program greatly aids in keeping the code modular and organized. Let's delve into some of these:

**generateCylinder()**: This function is designed to generate the vertices for a cylinder given its radius, height, and number of segments. This function can be reused to draw any cylindrical objects in the scene or even in other applications.

**generateTorus():** This function produces the vertices required to render a torus (a doughnut shape) based on specified inner and outer radii and the number of major and minor segments. This can be repurposed for any other scenes or applications where a torus shape is needed.

**resetView():** This function returns the viewer to the default viewpoint, effectively resetting any changes the user made to their perspective within the 3D environment.

**bindTexture():** Given how often I was binding textures, this simplifies the process. It associates specified textures with their intended models, allowing for detailed and textured renderings.

**loadTexture()**: Responsible for correctly loading different image files from the application's resources and preparing them to be used as textures within the 3D scene.

**compileShader()**: This function takes the source code of a shader and processes it, ensuring it's ready for rendering. It checks for any errors in the shader code and returns a compiled shader ready for linking.

**linkShaderProgram():** After shaders are compiled, they need to be linked together into a program that can be used in rendering. This function handles that task, ensuring that vertex, fragment, and other shaders can work together.

Various functions (such as drawMonitor(), drawKeyboard()) that handle the drawing of each item in the scene. By splitting the rendering process into distinct functions, the main rendering loop remains organized and more manageable.

The design decisions implemented in this 3D scene were meticulous, combining aesthetic appeal with efficiency. The intuitive navigation, empowered by WASD controls and mouse interactions, offers users an immersive experience. The introduction of dedicated functions for shapes like cylinders and toruses, as well as shader operations and texture bindings, the codebase achieves both clarity and modularity. These enhancements collectively allow for a seamless development experience and an enriched user engagement.