Trevor Mayo

CS 330

Final Project

The objects in the 3D scene were chosen based on their simplicity and availability around the house. The hemisphere-shaped kong, coffee mug, circuit marker, and an altoid can were all readily accessible and could be captured easily with a camera. The selected objects were also simplistic and distinct in shape, making it easier to create them in the 3D scene.

The functionality required for the 3D scene included navigation and perspective control. To achieve this, I programmed the WASD keys for navigation, Q/E keys for vertical movement, the mouse to look around, and the scroll wheel for speed control. Additionally, the P key was used to toggle orthographic perspective, allowing the user to switch between a 2D and 3D view of the scene.

Navigating the 3D scene involves using the WASD keys to move forward, backward, left, and right. To move vertically, the user can press the Q and E keys to go up and down, respectively. The mouse can be used to look around the scene, and the scroll wheel adjusts the speed at which the user moves. Pressing the P key toggles between orthographic and perspective views.

Custom functions such as "renderPen" or "renderKong" were used to make the code more modular and organized. The renderPen function, for example, would be responsible for rendering a pen in the 3D scene. This function could be called whenever the pen needed to be rendered in the scene, rather than duplicating the code each time the pen was used. Similarly, the renderKong function would be responsible for rendering the kong object in the scene. The function input was updated by including float values to enable movement of the entire object, instead of making adjustments to individual objects (in the case where the object was composed of multiple parts). This modification allows for the object's translation in different directions.

The incorporation of fragment and vertex shader classes also greatly improve the efficiency and readability of the code. By separating the functionality of the shaders into their own classes, I’m able to create more modular and reusable code, making it easier to maintain and modify. Additionally, the use of a shader class provides a convenient way to load and handle textures and other resources needed by the shader program. This further enhances the reusability of the code and saves time and effort in the long run. By leveraging these techniques, I was able to create more efficient and flexible code that’s reusable for future applications.

In general, all the custom functions make the code more reusable since they can be called multiple times throughout the program without having to rewrite the same code. Additionally, these functions make the code more modular, making it easier to update and maintain the codebase.

In conclusion, the 3D scene was developed with simplicity and accessibility in mind, utilizing objects readily available around the house. The scene's functionality includes navigation and perspective control, programmed using input devices such as the keyboard, mouse, and scroll wheel. Custom render functions along with shader classes were used to make the code more modular and organized, improving its reusability and maintainability.