Lee Kitchen  
CS-330

21APR2024

7-1 Final Project Design Decisions

The project code manages the rendering of a 3D scene, depicting a desk setup with a keyboard mat, monitor, computer riser and a watch. Each item is selected with precision to accurately represent real-world desk objects from the source image. For example, the desk's surface is rendered as a flat plane to represent the top of the desk. The riser, composed of a top and three legs, is crafted from rectangles. The transformations of scaling, rotation, and translation are applied to position each object realistically within the scene. Moreover, texture mapping and setting of materials are assigned using the functions `SetShaderTexture` and `SetShaderMaterial` to heighten the scene's realism. My development approach aimed to accurately and authentically represent the scene to be a visual cohesive representation of the source image of the desktop environment.

For users, the user can navigate the 3D scene using keyboard and mouse inputs. The `Mouse\_Position\_Callback` method tracks mouse movement within the window, updating the camera's yaw and pitch angles accordingly to simulate looking around. Next, the `Mouse\_Wheel\_Callback` function processes mouse wheel scrolling events to change the speed of the camera around the scene. The keyboard inputs are as followed, the `ProcessKeyboardEvents` function handles various actions. Users can move the camera forwards and backwards (`W` and `S` keys), strafe left and right (`A` and `D` keys), and ascend and descend (`Q` and `E` keys). Pressing the `L` key toggles the camera lock, allowing users to lock or unlock the camera's movement which is a function I created early on to help figure out camera positions. Moreover, the `P` key switches the projection mode to perspective, while the `O` key switches to orthographic projection. Overall, this setup provides users with intuitive controls for exploring the 3D scene, offering both mouse-based rotation and keyboard-based movement to navigate through the environment effectively.

The RenderScene() method orchestrates the rendering of a 3D scene by transforming and drawing various basic 3D shapes. Its modular and organized structure relies on several custom functions, each serving a specific purpose. Firstly, the SetTransformations() function is responsible for setting the scale, rotation, and position transformations for each mesh before rendering. This encapsulation allows for easy management and reuse of transformation parameters across different objects in the scene, enhancing the code's modularity. Secondly, the SetShaderColor() function dynamically sets the color of the shader used for rendering, providing a reusable way to customize object appearances. While the code comments suggest a shift towards texture assignments, this function still highlights the flexibility in shader properties adjustment. Additionally, the SetShaderTexture() function assigns textures to meshes, enhancing realism and visual appeal, while promoting code reuse and simplifying texture management. Finally, the SetShaderMaterial() function assigns materials to meshes, influencing how light interacts with their surfaces. By encapsulating material assignment in a separate function, it facilitates easy customization of material properties across different objects, further enhancing modularity and organization. Overall, these custom functions streamline the rendering process, promote code reuse, and enable easier customization of the scene's appearance.