Felix Carela

6/24/2024

CS-330

7-1 Final Project Submission

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

I chose a variety of objects to a 3D scene. These objects include a round table, a book, an apple, headphones, and a can. Each object was selected to demonstrate different aspects of 3D modeling, texture mapping, and lighting effects. The table provides a base for placing other objects and serves as a reference point for scaling and positioning. The book demonstrates the use of different textures on various faces, highlighting the importance of texture mapping and UV coordinates. A spherical object like an apple helps to showcase smooth shading and texture wrapping on curved surfaces. A torus (headband) for the headphones adds complexity to the scene with its curved shape. The can and its top illustrate how different objects can be combined and textured separately to create a cohesive model. The program includes functionalities such as texture loading, binding, and application to 3D models. Additionally, the use of lighting and materials enhances the visual realism of the scene. The development choices were guided by the need to fulfill these functionalities while keeping the code modular and organized.

**User Navigation in the 3D Scene**

The user can navigate the 3D scene using a virtual camera controlled by mouse and keyboard inputs. The mouse movement adjusts the camera's view direction, while the keyboard keys allow for movement and zooming. Mouse movements are processed through the Mouse\_Position\_Callback function, which calculates the offset and updates the camera's orientation accordingly. Keyboard inputs enable various types of camera movement: the W and S keys move the camera forward and backward, the A and D keys pan the camera left and right, and the Q and E keys move the camera up and down. Additionally, pressing the P key switches to perspective projection, while the O key switches to orthographic projection. These controls are managed within the ProcessKeyboardEvents and Mouse\_Position\_Callback functions, ensuring smooth and intuitive navigation.

**Custom Functions for Modular and Organized Code**

Several custom functions were developed to keep the code modular and reusable. The CreateGLTexture function loads textures from image files, configures the texture parameters, generates mipmaps, and registers the texture with a unique tag. It is reusable for loading any texture required in the scene. The SetTransformations function sets the transformation matrix for scaling, rotating, and translating objects in the scene, centralizing the transformation logic and making it easier to apply consistent transformations across different objects. The SetShaderTexture function binds the appropriate texture to the shader based on the provided tag, abstracting the texture binding process and ensuring that textures are correctly applied to objects without duplicating code. The SetShaderMaterial function applies material properties to the shader for the specified material tag, allowing for easy reuse of material settings across different objects. The DefineObjectMaterials function defines and stores material properties for different types of objects, enabling consistent material application and easy adjustments to material properties. The LoadSceneTextures function loads all the textures required for the scene, encapsulating the texture loading logic and making it easier to manage and update texture assets. These custom functions not only keep the code organized but also promote reusability, making it easier to extend and maintain the 3D scene. Each function is designed to perform a specific task, ensuring that the codebase remains clean and modular.