Justification of Development Choices for the 3D Scene

Alvin Lagoon

CS330

06/27/2024

Selected Objects

For this 3D scene, the objects were chosen based on their ability to demonstrate a range of rendering techniques, including texture mapping, material properties, and lighting effects. The selected objects include:

1. Floor: A large, flat surface to serve as the base of the scene. It allows for the demonstration of texture tiling and UV scaling.
2. Blue Box: A simple geometric shape to showcase basic texture mapping and transformations.
3. Candlewick , Candle Glass and candle base: These objects illustrate more complex materials and lighting effects. The candle includes a semi-transparent glass and a wick, demonstrating multi-material objects and the application of different shaders.
4. Cup: This object is used to showcase tapered cylinder meshes and the application of textures to curved surfaces.
5. Remote: A complex shape with a distinct texture, used to demonstrate detailed texture mapping and material properties.

These objects were chosen because they provide a balance between complexity and simplicity, making them ideal for demonstrating key 3D graphics concepts without overwhelming the scene.

Programming for Required Functionality

The required functionality was programmed using OpenGL for rendering, GLM for mathematics, and custom shader management for handling materials and lighting. Key functionalities include:

* Texture Loading and Binding: Using the CreateGLTexture and BindGLTextures methods to manage texture loading and binding in OpenGL.
* Material Definition: Defining various materials with different properties to be used across different objects in the scene.
* Lighting Setup: Configuring directional and point lights to create realistic lighting effects.
* Transformations: Using transformation matrices to position, rotate, and scale objects in the scene.

User Navigation in the 3D Scene

Virtual Camera Control

The virtual camera was set up to allow users to navigate the 3D scene using different input devices. The setup involves:

* Keyboard Input: For moving the camera forward, backward, and strafing left or right. Typically, the W, A, S, and D keys are used for these movements.
* Mouse Input: For controlling the camera's orientation. Moving the mouse horizontally and vertically adjusts the yaw and pitch of the camera, respectively.
* Scroll Wheel: For zooming in and out, if applicable.

The camera's position and orientation are updated based on these inputs, and the view matrix is recalculated accordingly to reflect the camera's new position and orientation in the scene.

Custom Functions for Modularity and Organization

To keep the code modular and organized, several custom functions were developed. These functions include:

Texture and Material Functions

* \**CreateGLTexture(const char* filename, std::string tag)\*\*: Loads a texture from a file, configures it for OpenGL use, and stores it with a tag for easy retrieval.
* SetShaderTexture(std::string textureTag): Sets the texture in the shader based on the tag provided.
* SetShaderMaterial(std::string materialTag): Passes material properties to the shader for the specified material tag.

These functions abstract the details of texture and material management, making the code more reusable and easier to maintain.

Transformation and Rendering Functions

* SetTransformations(glm::vec3 scaleXYZ, float XrotationDegrees, float YrotationDegrees, float ZrotationDegrees, glm::vec3 positionXYZ): Sets up the model transformation matrix based on scale, rotation, and position parameters.
* RenderFloor(), RenderBlueBox(), RenderCandle(), etc.: Each render function sets up the necessary transformations, textures, and materials, and then draws the object. These functions keep the rendering code organized and modular.

By using these custom functions, the code becomes more modular, making it easier to understand, maintain, and extend. Each function has a specific responsibility, ensuring that the code is not only functional but also clean and well-organized