Jeffrey Flores

OpenGL Final Project Reflection

October 20, 2024

For this project, I chose to create a beach-themed 3D scene inspired by an image of a sunset on the shore. This scene contains many key elements, such as an umbrella, a palm tree, seashells, and a reflective plane representing the sandy beach. Each of these objects was selected because of their simplicity and ability to be broken down into basic geometric shapes, which fulfills the requirements for low-polygon models. The primary development choices were driven by ensuring that the objects could be effectively textured and lit within the constraints of OpenGL while maintaining smooth performance.

The decision to include the umbrella, palm tree, seashell, and ground plane was based on the need to replicate the image selected for the project. Each object is made of simple 3D shapes that are easy to model while also visually representing elements from the original scene. The umbrella consists of a cone for the canopy and a cylinder for the pole, both of which keep the polygon count under control. The palm tree trunk is modeled using a cylinder, and its leaves are represented by cones. The seashell is a sphere, flattened using scaling transformations, to create a natural appearance. The ground plane is modeled as a large flat plane that serves as the beach.

The low polygon counts of these objects ensure that the performance remains smooth, even when textures and lighting are applied. The polygon limit of under 1,000 triangles for each object was maintained by utilizing basic shapes, which not only reduced complexity but also made the UV mapping for textures straightforward.

Textures were applied to give a more realistic feel to the scene. I selected royalty-free textures that were 1024x1024 pixels in resolution, ensuring that they were high-quality while being appropriate for OpenGL rendering. Textures were mapped onto the umbrella (wood for the pole and fabric for the canopy), the ground plane (sand texture), and the seashell (a natural shell texture). I utilized OpenGL’s glTexImage2D function to load and map the textures, ensuring they fit correctly on each shape by applying UV scaling where necessary.

For lighting, I used two sources which were a directional light simulating sunlight and a point light to illuminate areas that may fall into shadow. The directional light was positioned at an angle to simulate the effect of the sun, casting both ambient and diffuse lighting across the scene. The point light was introduced to highlight objects such as the seashell and ensure no part of the scene was left in complete darkness, providing a polished look. Both lights were set up to use Phong shading, incorporating ambient, diffuse, and specular components to create realistic lighting effects.

Navigating the 3D scene is important for providing the user with an interactive experience. The camera in the scene is controlled using keyboard and mouse inputs. The user can move the camera forward, backward, left, and right using the W, A, S, and D keys, respectively. Vertical movement is controlled using the Q and E keys, allowing the user to move up and down in the scene. The mouse is used to control the camera’s orientation. Moving the mouse left or right adjusts the yaw (horizontal orientation), while moving the mouse up or down adjusts the pitch (vertical orientation). This enables the user to look around the scene freely. To implement these controls, I created a Camera class that encapsulates the camera’s position, orientation, and movement logic.

To keep the program modular and organized, I developed several custom functions that handle specific tasks, ensuring the code is reusable and maintainable. The “Set Transformations” function is responsible for applying translation, rotation, and scaling transformations to the 3D objects. It takes parameters for position, scale, and rotation (in degrees) and constructs a model matrix. The “Set Shader Texture” function simplifies the process of binding a texture to an object before rendering it. By passing the texture tag, the function retrieves the appropriate texture from memory and binds it to the current object. This abstraction ensures that textures can be easily applied to different objects without repeating code for texture setup. The “SetDirectionalLight and SetPointLight” functions encapsulate the setup of lighting parameters for the scene. They pass the directional and point light properties (ambient, diffuse, and specular values) to the shader. By organizing the code into small, well-defined functions, I was able to make the program more modular and maintainable. Each function is reusable and serves a clear purpose, reducing redundancy and keeping the overall codebase clean and efficient.

In conclusion, the development choices for my 3D scene were focused on simplicity, performance, and visual quality. By selecting low-polygon objects, applying realistic textures, and setting up lighting according to the Phong shading model, I was able to create a polished scene that is both interactive and visually appealing. The camera controls allow the user to explore the scene freely, and the modular design of the program ensures that the code is easy to manage and extend in future developments.