CS330

Aidan Farhi

2025-04-20

CS330 Final Project

While building my 3D scene representing a real-world image, I leveraged the OpenGL library using the C++ programming language. OpenGL is one of the most popular and ubiquitous libraries for 3D rendering available. The real-world scene I chose was that of a well-maintained garden. It included arches, grass, hedges, a roof, and a paved sidewalk. I chose this scene because it contained a rich variety of objects that were simple enough to model yet still complex enough to provide a challenge using the tools available. To represent the various objects of the scene, I used several basic shapes available within the library. For the paved sidewalk and roof, I used a Plane. While creating the arches, I used a combination of Torus' and Prisms. The grass was represented with a Box and the hedges were created using Tapered Cylinders. Each shape had a texture and material applied to it to make it more realistic. In addition to rendering shapes, materials, and textures, I also used various lighting techniques, including Point Lights and a Direction Light to enhance realism. Another feature of the project was utilizing input devices to allow a user to navigate around the 3D scene. Using a combination of basic shapes, texturing, lighting, navigation, and other techniques I was able to create a 3D representation of the scene in a way that closely represents the original.

Included in this project is the ability for a user to navigate the 3D scene. This was achieved by mapping events from input devices to control the camera movement. All of the logic for this was contained within the ViewManager class. This class has methods for processing a variety of events from the mouse and keyboard. For example, there were methods called Mouse_Position_Callback( ) and Mouse_Scroll_Callback( ) that hooked into mouse movement and mouse scrolling, updating the perspective of the camera and adjusting the movement speed respectively. The WASD keys allowed the user to go forward, left, right, and backwards on the XYZ axis. The Q and E keys moved the camera up and down on the Y axis. By moving the mouse, the user can see different perspectives of the scene. Using the O and P keys the user can toggle between orthographic and perspective views of the 3D scene. The speed of camera movement was adjustable using the mouse scroll wheel. If the user scrolled up, the speed increased. If the user scrolled down, the speed decreased. Enabling the user to navigate the scene using a variety of inputs provides a rich and immersive experience.

The project was organized into several C++ classes. Each class was responsible for a different aspect of the project. Within each class, there were several methods that were used to implement a separate smaller unit of functionality. By breaking down the project into these modular units, it was easier to navigate the codebase, logically separate concerns, and promote code re-use. The SceneManager class was responsible for loading and setting textures, defining materials, setting up lighting, and rendering the scene. In the RenderScene( ) method, the logic for drawing and transforming the 3D was implemented. This method included calls to the SetTransformations( ), SetShaderTexture( ), and SetShaderMaterial( ) methods for each shape of the scene. The SetTransformations( ) method was leveraged to set the transform buffer. SetShaderTexture( ) takes in a texture ID and then sets the texture within the shader. The SetShaderMaterial( ) method passed the material values into the shader. By re-using these methods while rendering each shape, it reduced the amount of code necessary to maintain and allowed for a more modular and organized project. By combining basic shapes, textures, lighting, navigation, and other techniques, I successfully created a 3D scene that closely mirrors the original representation.