Arnold Limberg

Professor Aly

CS 330

8/24/24

3D Scene Development Design Decisions

I made a number of important decisions about object selection when creating this 3D scene. In addition to satisfying the project specifications, the items I selected enabled me to increase the complexity as my abilities improved over the course of the semester. I chose a desk, a coffee cup with a handle, a speaker with a mesh texture, a monitor with a screen and stand, and a mouse pad. I created many of these items using reusable and manipulable fundamental shapes such as cylinders, boxes, planes, spheres, and tori. This method helped me become more and more comfortable with shape manipulation as the project progressed.  
 I worked my way up to more complicated items, such as the coffee cup with its own handle, after starting with smaller ones, like the desk plane. In order to achieve the necessary functionality, I used the ViewManager for camera control and view switching, and the ShaderManager for handling textures and lighting. The separation of concerns made the code easier to manage and more extensible.  
 The camera was configured in the ViewManager class. One of its components is a camera object, which records zoom level, front vector, location, and up vector. In order to change camera position and movement speed, mouse callbacks managed mouse movement and scrolling. The scene is controlled with keyboard and mouse inputs: the QE keys for upward and downward movement, the WASD keys for forward, backward, left, and right movement, the ESC key quits the application, while the P and O keys alternate between perspective and orthographic projections. Furthermore, by adjusting the camera's orientation with the mouse and changing its speed with the mouse scroll wheel, the user can pan around the scene.  
 With this configuration, users may easily and quickly adjust the camera's settings, allowing them to examine the 3D environment from different perspectives and distances.  
I wrote a number of custom functions to enhance the code's modularity and organization; these functions are crucial for engaging with the 3D environment.  
 The SceneManager class's DefineObjectMaterials function sets up the material attributes of the different objects in the scene. This function specifies the ways in which various objects interact with light to provide distinctive visual properties. This feature makes it simple to create and alter materials, as well as provide visual fidelity to the scene. For example, the monitor screen material has a high shininess value to create a glossy, shiny appearance, whereas the wood material has a low shininess value to create a matte surface.  
The `SetupSceneLights()` function is essential for establishing the scene's lighting environment. By configuring two light sources with distinct properties, this function produces a lighting scene that is more lively and lifelike. The second light source, a softer light from the right, and the first, a harsh white light from behind, give the scene more depth and dimension.

The SceneManager class has methods like `LoadSceneTextures()` and `RenderScene()` for loading and rendering objects. Texture images are loaded and converted to OpenGL texture data using the `LoadSceneTextures()` function. Together, these features construct a modular, adaptable framework capable of defining, loading, and rendering intricate 3D scenes with realistic lighting and materials. The division of responsibilities between material specification, lighting setup, texture loading, and object rendering simplifies scene expansion and maintenance.  
 I was able to write a more intricate scene while maintaining understandable and manageable code by structuring the code in this modular fashion. As the project progressed, this method also made iterative development and debugging simpler. I discovered during the semester that this modular approach made it much easier to add new features and improve ones that already existed, which made a big difference in the project's overall success.  
 In conclusion, this project has been a priceless way to learn 3D graphics programming. It has taught me the value of organized, modular code design in handling complicated projects, as well as the technical aspects of dealing with OpenGL and 3D rendering. The journey from basic lighting to more intricate material and light interactions, as well as from simple shapes to more complex objects was worthwhile for an upcoming developer like me.