# **7-1 Final Project: Design Decisions**

Mark Baligad

Southern New Hampshire University

CS-330: Computer Graphics and Visualization

Professor John Graham

12/15/2024

**Design Decisions**

**Development Choices**

For my project with Triangle and Cube Studios, I recreated the scene for an image of four pieces of furniture in a sunlit concrete room. Each piece of furniture is differently shaped from the other and requires a combination of basic 3D shapes to recreate in OpenGL. To make my selection, I inspected the furniture to determine how many faces each object had and whether they were round or square-shaped. Once each component of the furniture piece was broken down, it became easier to match them up to a 3D shape, such as a plane, cube, cylinder, or sphere.

Beginning with the floor and wall, I used two planes textured with concrete. For the potted plant on the left of the scene, a cylinder worked to represent the pot and the three legs. I used elongated box shapes for the three plant stems and then four elongated green spheres to represent basic leaf shapes. The mirror begins simply as a vertically elongated cube but becomes more complex as I create individual cube shapes to frame the mirror in. Box shapes were used for the stool for all four legs, and then a flattened cylinder served as the seat. The table at the right of the scene consisted of cylinders for the legs and a cube for the tabletop. Finally, on top of the table rests a small gray cylinder for the pot and three small green spheres to recreate the cactus plant.

Lighting my scene was necessary to recreate the warm sunlight light originating from the left of the scene. To accurately portray the lighting, I used wood, clay, and cement materials to configure how the lighting interacts with each material. The wood in my scene displays highlights where the light shines on it, so I configured a higher shininess value compared to the cement. I began with a dim white ambient light for the light sources to ensure the entire scene was illuminated. To recreate the sunlight, I added a warm-colored diffuse light positioned to the upper left of the scene with a large focal strength. Finally, I created a dim white light positioned above the scene to ensure nothing in the scene was too dark.

**Scene Navigation**

To handle navigation in the 3D scene, I updated the ViewManager class to process keyboard and mouse events. The WASD keys control horizontal movement, the Q and E keys control vertical movement, mouse movement reorients the camera, and the mouse scroll wheel can increase or decrease movement speed. To control movement speed, I set minimum and maximum allowed values and then set the sensitivity to limit how much movement speed increased by scrolling forward or decreased by scrolling backward. Finally, the O key was programmed to swap the camera between orthographic and perspective projection.

**Custom Functions**

The code project used to render the scene utilizes multiple classes and methods to render the scene in OpenGL. The ViewManager class handles projecting the 3D scene to create a camera view, moving the camera, and processing user input to move the camera. The CreateDisplayWindow method creates a window that will display the OpenGL render and also defines the callback functions that will monitor user input from the keyboard and mouse. The Mouse\_Position\_Callback method tracks mouse movement, which pans the camera, and the Mouse\_Scroll\_Callback method increases and decreases movement speed. The ProcessKeyboardEvents method reports when the specified keys are pressed and calls the ProcessKeyboard method to move the camera through the scene horizontally or vertically. Finally, the PrepareScene method handles camera movement according to the framerate and sets the camera projection as either orthographic or perspective.

The SceneManager class is used to render the scene, including the lighting and objects. Within the scene manager, I use the CreateGLTexture method to load image files as textures; the SetTransformations method transforms objects using rotation matrices, and SetShaderColor and SetShaderMaterial apply color to the objects. The DefineObjectMaterials method defines how an object should interact with lighting, including ambient, diffuse, or specular lighting. The SetupSceneLights method places light sources on the scene and defines the strength and color of the ambient, diffuse, and specular lighting. Finally, the PrepareScene method loads all of the scene data to memory, and the RenderScene method draws and transforms the shapes in the scene.

When I began adding in multiple shapes, it became clear that the code required to render each 3D object would require a lot of repetition and be difficult to read and maintain. Rendering each mesh requires calling methods to set the scale, rotation, position, textures, and shape. To solve this problem, I added a custom RenderMesh() method to encapsulate the steps for transforming, texturing, and rendering the 3D shapes. Creating a MeshType Enum list also made passing a specific 3D shape to the RenderMesh() method easy. This saves me from repeating the code to declare variables and calling each sub-function for every object in the RenderScene() method.

**A mirror and stool in a room

Description automatically generated**

Reference Image

A screenshot of a video game

Description automatically generated

Render

**References**

Wahyu, N. (2019). *Brown wooden table with chair* [Photograph]. Pexels. Retrieved October 29, 2024, from [Pexels](https://www.pexels.com/photo/brown-wooden-table-with-chair-3097112/)