Nyzheir Warner

CS-330

08/16/2025

Final Project Reflection

**Justifying Development Choices**

Objects in my Scene consist of the following: 1x Table (Plane), 1x Wall (Plane) 1x Chessboard(Box, Plane), 16x Pawns (Half Circle, 2x Torus, Tapered Cylinder, Sphere), 2x Kings (2x Half Circle, 2x Torus, 2x Tapered Cylinder, 2x Box), 2x Queens (2x Half Circle, 2x Torus, 2x Tapered Cylinder, Sphere), 4x Bishop (Half Circle, 2x Torus, Tapered Cylinder, Bishop Head, Sphere), 4x Knight (Half Circle, Torus, Knight Head), 4x Rook (Half Circle, Torus, Tapered Cylinder, Rook Head).

For most of these objects, I attempted to use the simplest models possible, utilizing only the provided basic shape meshes. This is true for all pieces except for the knight, rook, and bishop. For these three pieces, I elected to venture out and create some custom meshes as I felt I had already met the requirement of 4 complex objects with the basic meshes, i.e., (Pawn, Queen, King, Chessboard). These three pieces in particular needed some special attention outside of the basic shapes, as most of them require some sort of deformation, which was not possible with the tool set provided. To achieve the look, I wanted for my project, I ventured to create custom meshes for the rook, knight, and bishop in Blender, an open source 3d modeling tool. After I had modeled the meshes, I used Blender's built-in scripting hub to export the vertex and index data in order to export them to our C++ application as custom meshes. Doing this allowed me to truly achieve the original picture I had in mind when selecting a chessboard for my project. Furthermore, it allowed me to explore the possibilities of incorporating more advanced tools into the workflow to achieve stunning visuals.

**User Navigation**

All of the user navigation for our application is housed in a method in ViewManager.cpp.cpp.cpp called ProcessKeyboardEvents(). This method is called whenever we want to process a keyboard input from the user. The following is a keymap that the user can reference to interact with our 3d scene:

            ESC – Close the Window

            W – Zoom Camera In

            S – Zoom Camera Out

            A – Pan Camera Left

            D – Pan Camera Right

            Q – Move Camera Up

            E – Move Camera Down

            P – Switch to Perspective View

            O – Switch to Orthographic View

            1 – Front View

            2 – Right Side View

            3 – Left Side View

            4 – Top-Down View

            L – Turn Flashlight ON/OFF

**Custom Functions**

There was a total of 9 custom functions we created in order to help us create modular and reusable code. These functions have helped remove redundant code, Such as the Create ShapeTransformation method, which helps define each mesh size, orientation, and position. There are also methods that define each object in my scene, including the chessboard and all of its pieces. These methods help contain the code that defines what that piece is, as well as provide an easy way to broadly manipulate each piece, requiring variables such as the xPos, yOffset, zPos, and color. This helps propagate changes to all of the chess pieces to act as one single object. There is an additional variable needed for the bishop, rook, and knight called yRotation, which is needed as the orientation of these pieces’ matters based on which side they are on.