# 3D Scene

## Reference Image

A computer desk with multiple monitors and a keyboard

Description automatically generated

## 3D Scene Progress Pictures

1. A white rectangular object with red and blue buttons

   Description automatically generated
2. A keyboard on a black surface

   Description automatically generated
3. A computer keyboard with a colorful light

   Description automatically generated
4. A computer screen with multiple screens

   Description automatically generated with medium confidence
5. A black object with a cross on it

   Description automatically generated

# Reflection

Given my 3D scene selection I felt the objects I displayed capture critical components of my 3D scene such as the monitors that uses cubes and planes, the MacBook uses cubes, and the keyboard uses cubes, I included the monitor mount uses cubes, torus, cylinder and really tied the scene together. Although I did not complete the mouse object, I plan to refine the scene even after this class. Most of my objects were created utilizing cubes with various transformations and translations. Although objects such as the MacBook could use additional objects or radiuses around the edges for a more realistic look, I feel they use of cubes captured the overall design well. Successfully completing the scene up to this point required dedication and countless trial and errors until concepts were understood. Meeting the project requirements involved me adding details to certain objects such as a torus to represent the backing ring of my monitors. These details may not be noticeable at first but contribute largely to the detail of the scene.

Navigation within my scene is accomplished through various key and mouse inputs. WASD keys will provide movement based on the viewer’s current camera position, W will bring the camera position forward, S will move the camera position backwards, A will move the camera position to the left and D will move the camera position to the right. Q will move the camera position up y+ regardless of the user’s camera position, E will move the camera position y- down regardless of the user’s camera position. P will set a 2D projection of the screen. Mouse controls are accomplished using yaw and pitch with yaw handling horizontal movement and pitch handling vertical movement. The user will load the scene and be placed in a default position set by a first mouse Boolean that then sets the position of the default last x and last y variables. Next calculations are done to figure out the offset or how much the mouse has been moved to determine the directions to point the user, code is set in place to ensure the users view does not flip. These controls were set up through several functions, one for handling the keyboard input, one for the mouse input, one for the scroll wheel input as well as a specialized one for the switching of views. The input controller class holds all input related functions and is modular allowing for additional inputs to be added if required.

I refactored my code several times during the development process and could still use a deep refactoring, as I learned OpenGL as well as C++ better I found ways to refactor my code to allow for more modular ability and easier readability, a few examples would be my Scene Drawer class, this class utilizes my MeshData.cpp and Buffer.cpp to allow for the creation a function called create Buffers that can then be utilized once within my main to allow for a single line of code to create all buffers for my objects. Another example would be my Draw Scene function that calls other drawing functions to allow for a single line of code to be implemented into my main for the drawing of all objects within my scene. Another modular code could be my draw Cube function that is used many times to replicate the arms on my monitor mount. This method passes scale factors and translation factors as well as a rotation degree to position objects where I want. This code can be utilized numerous times and draw cubes of all sizes. My shader class allows for multiple fragments and shader source to be added and compiled into a shader program as well.