**Computer Graphics**

**Assignment 1**



Before I started to code I wanted to define which functions I wanted to use.

Most of the basic OpenGL template code is based on our tutorials so I just want to show the ones that weren’t or coded by me.

// Loading data  
std::string s;  
std::string head;  
GLfloat vertex[3030];  
GLushort face1[6066];  
int i = 3, j = 0;  
  
std::ifstream ifs("eight.uniform.obj");  
  
while (std::getline(ifs, s))  
{  
 if (s[0] == 'v') //vertex  
 {  
 std::istringstream sin(s);  
 sin >> head >> vertex[i] >> vertex[i + 1] >> vertex[i + 2];  
 i += 3;  
 }  
 else if (s[0] == 'f') //face  
 {  
 std::istringstream sin(s);  
 sin >> head >> face1[j] >> face1[j + 1] >> face1[j + 2];  
 j += 3;  
 }  
}

I have used this code to open and read the obj file in order to get the necessary vertices and faces form the file. I opened the obj file in .txt format to better understand how to call the data correctly.

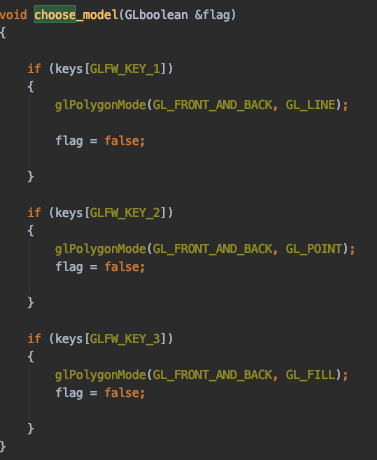




The movement function allowed me to control the camera as I pleased. -, = allows me to come closer and further from the object. The reason that I have used = instead of + is that I use a MacBook and it was more convenient to control this way. Also A , D, W,S allowed me to move the camera left, right, forward and backward respectively.

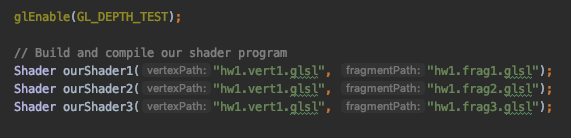


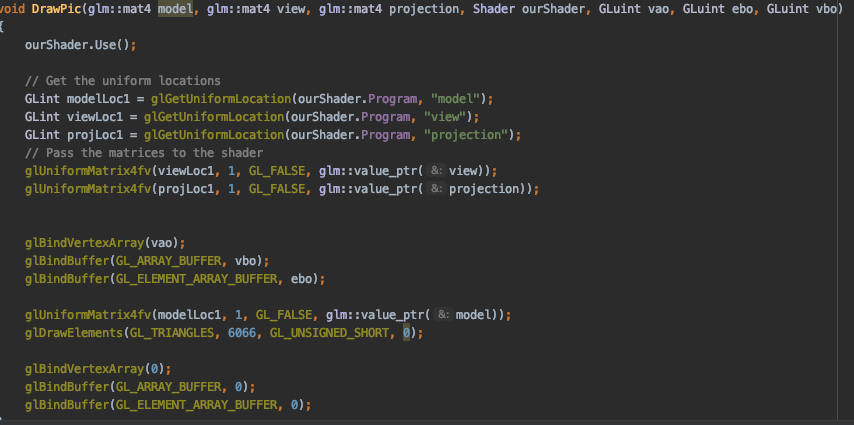
This function allowed me to rotate the object within the x y and z coordinates respectively.

This choose model function allowed me to switch in between models. I was able to call the appropriate draw function with the correct shaders to display the accurate object model that I wanted to display. So the 1, 2 ,3 and 4 keys allowed me to do and the C key allowed me to change the fragment shader colour in my wireframe model

As you can see above, in order to change the colours for my appropriate purpose, I have created 3 fragment and 1 vertex shader file.







Finally I want to briefly talk about my DrawPic() function, I have added the ourShader.Use() function to call our shaders. Get our uniform locations. Passing the matrices to our shaders and bind our vertex array and buffer to display our object in our window.