3D Viewing and Phong Shading Model

Eric Parsons

2295804

The purpose of this assignment was to implement the 3D viewing and Phong shading model using OpenGL.

Initially, I got a black screen, which gave me a panic attack, as my 3D and Phong shading model was projecting the emptiness and blackness of my heart and emotions, but I then started implementing the 3D viewing model.

I first wrote code for the vertex shader, as it passes information to the fragment shader. I transformed the vertex position into world space, then the normal into world space. I then calculated the clip space position. I had to search many hours on the internet to get this to implement properly, as I was stuck on getting the lighting calculations correct in the fragment shader.

In the phong.frag file, the shader only had a placeholder that set the output color to red and didn't actually have any lighting calculations. So, like Albert Einstein said (probably at some point in his life, idk actually) we needed to change that. I implemented ambient lighting to simulate the light that would affect all surfaces equally. This essentially gave the object a basic level of illumintation. I added diffuse lighting to make a more direct lighting effect and then threw in specular lighting with a strength of 0.5 so it wouldn't be too shiny or too dull. My favorite number is 32 so I set the 'shininess factor' to 32. Since the final color is calculated as a combination of ambient, diffuse, and specular components, multiplied by the object's base color, I did exactly that in the code to produce the color.

The camera.h was trivial. I used sample code available on many internet forums, and set the main.cpp to accept the function.

In main.cpp all I did was setup the camera implementation from camera.h. I created a basic projection Matrix to simulate the 3D view. Basically it creates depth by having objects further away appear smaller. The projection matrix gets passed to the vertex shader. I got the view matric to accept the camera movements from WASD. The light and camera positions were added to the shaders. I established the lighting and object colors, as well as the lighting position location. The vertex array object and vertex buffer object were set up (this was difficult) to hold the cube's vertex data.

FINALLY, I wrote the code to draw the cube based on the transformation matrix.

Then, I sat in a corner and cried and then did a few stretches because my back hurt from sitting in a chair and coding for 11 hours straight. I finally relaxed then realized I needed to type this report as I accidently forgot to do it for the last assignment.

In the end we get an orange cube with a viewport that can be controlled with the mouse and keyboard.