Kevin King
Professor Zhu
COSC 77: Computer Graphics
Assignment 2 - Technical Implementation

**Step 0: Nonsense Rendering in GLSL**

For this step, I played around in the helloworld.frag file by shuffling the different components of the vectors to manipulate the colors. I also used trigonometric functions and the mix function to experiment with the vector color components. Additionally, in the Update_Vertex_Color_And_Normal_Mesh_For_Mesh_Object function in main.cpp, I added three for loops instead of the initial one that made each of the vertices green. Instead, I had each for loop go through a third of "vn" and specify a different color for each third. I commented these for loops out so that the rest of the steps wouldn't get messed up.

**Step 1: Calculate the Vertex Normal For a Triangle Mesh**

I started with a for loop that iterates through each triangle, gets the positions of its three vertices from the "vertices" data structure, then gets the two edges using those vertices. From there, I used the normal vector (cross product) equation to calculate the normal vector, and then added the normal vector to the normals array. Lastly, I looped through the normals array and normalized each vector inside it.

**Step 2-3: Lambertian and Phong Shading**

For Lambertian shading, the output variables for the vertex shader that I added were the normal vector and the vertex position vector. For the fragment shader, I added constant float variables for the material properties k_a and k_d, as well as constant vectors for their light source colors I_a and I_d. The ambient light is just k_a*I_a. For the diffusive light source, I got the normal vector of the surface point and the directional vector pointing from to the light source as well. Then, I followed the equation in order to get the diffusive light, and frag_color was the sum of the two.

For Phong shading, I did the same thing but also added constant float k_s and vector I_s for the specular light and got the reflection direction from the light source and the direction vector from the surface point to the camera position. Then, the frag_color output was the sum of the three light sources.

**Advanced Lighting Effect**

In my_phong.frag, I created an array of 4 different colors (orange, green, brown, yellow) for my I_d variable to switch between over time. In order to make the color change gradual, I used the mix function on two consecutive colors in the array, setting the weight by using the modulus operation on the iTime variable. The modulus operation makes it so that, as time goes on, the weight that we have will not grow too large as we are using a remainder value.