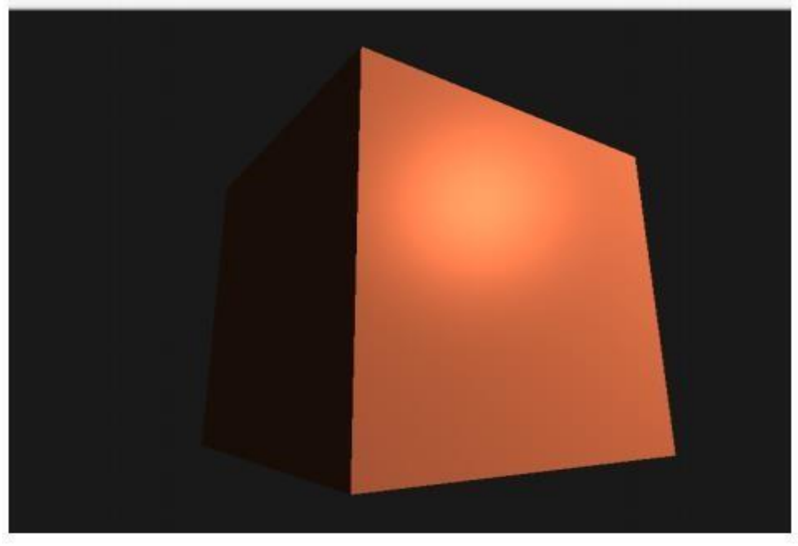
COSC 4370 – Homework 3

Name: Daniel Nguyen
PSID: 1479087
October 2022

1 Problem

The problem for this homework is that we need to implement 3D viewing and Phong shading model. First we must complete varies steps such as, completing the GetViewMatrix() function in Camera.h and projection matrix in main.cpp. Second, we need to write the vertex and fragment shaders for Phong model. If all has been implemented correctly the following image can be reproduced:



2 Method

Methods that were used in this Homework are lookat, perspective, Ambient lighting, Diffuse lighting and Specular lighting. Granted some of these are more commands rather than methods, but we do use these commands to help us recreate the image given in the problem. Using lookat and perspective commands to help us complete the camera and Ambient, Diffuse, and Specular for Phong model.

2.1 References

https://www.geertarien.com/blog/2017/07/30/breakdown-of-the-lookAt-function-in-OpenGL/#:~:text=The%20LookAt%20function%20in%20OpenGL%20creates%20a%20view%20matrix%20that,and%20orientation%20of%20a%20camera.
https://learnopengl.com/Getting-started/Camera

https://learnopengl.com/code_viewer.php?type=header&code=camera
https://learnopengl.com/Getting-started/Camera
https://learnopengl.com/Lighting/Basic-Lighting
https://learnopengl.com/code_viewer.php?code=lighting/basic_lighting-exercise2

3 Implementation

The lookat command "creates a view matrix that transforms vertices from world space to camera space. It takes three vectors as arguments that together describe the position and orientation of a camera" the three vector being position, position + up, up.

Perspective command which allows the user to see the camera's projection. This works because the perspective uploads the matrix to the GPU each frame. The command takes four variables first being the field of view, but we need to replace that with camera.zoom which basically does the same thing. The second variable is the screen size, third being near so 0.1 and fourth being far so we'll leave that value at 100.

Ambient lighting basically known as global illumination because we know that light reflects off surface so there is no single direction or source that light comes from. There is a simple calculation we can use to determine ambient lighting. Ambient is determined by how strong ambient is so we can give it any level of strength * lightcolor. After getting that value we multiply that with the lightcolor giving the object either a stronger or light lightcolor on it depending on the ambient strength.

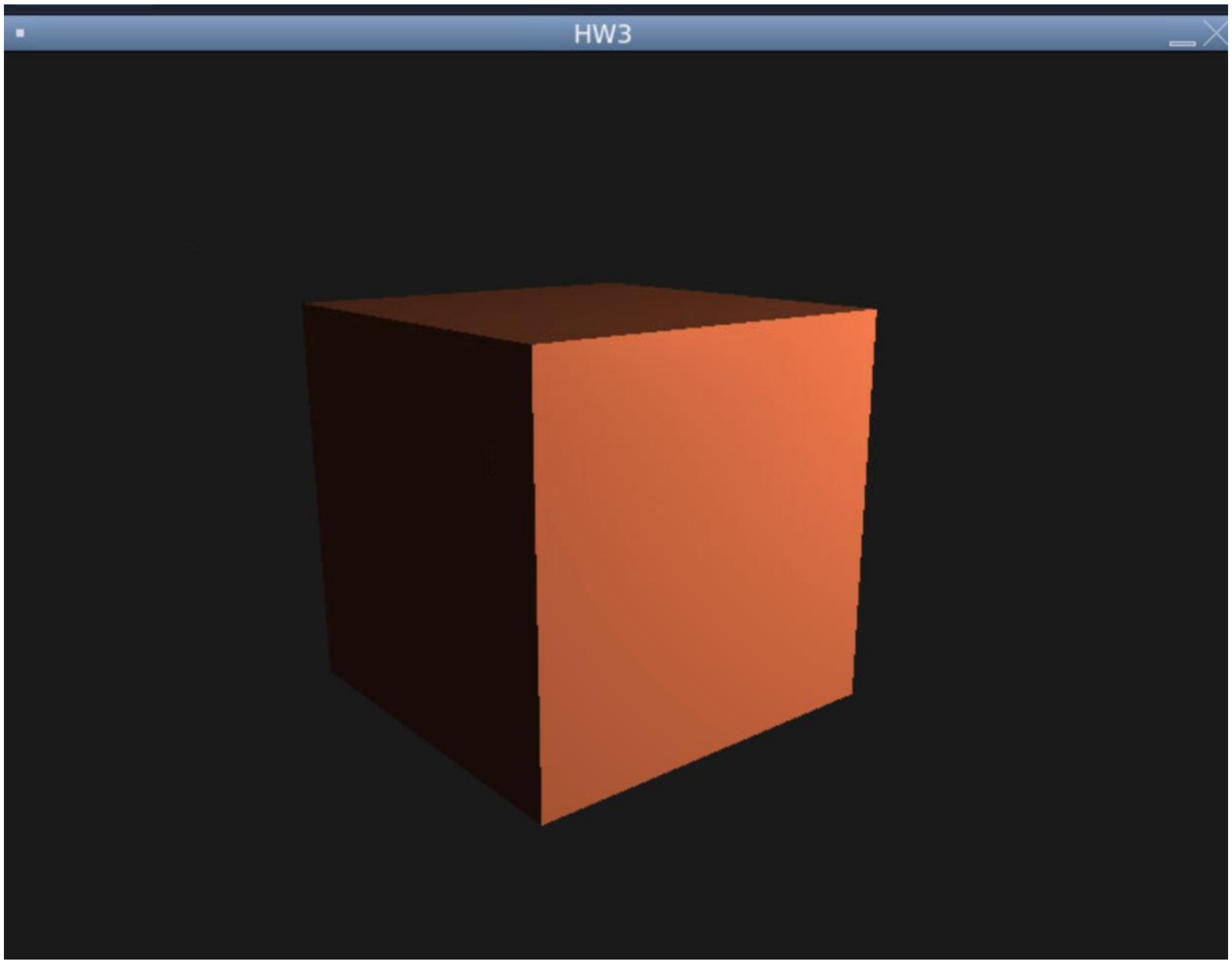
Diffuse Lighting "gives the object more brightness the closer its fragments are aligned to the light rays from a light source." We need a few calculation for this part, first we need to find the direction vector between the light source and the fragment position using lightpositon – fragmentposition, second we need to find the impact of the light on the fragment using the dot product on norm and lightdirection(from the first step) using this value we multiple with the light color getting the diffuse component.

Specular Lighting is light diffuse lighting but is based on view direction. We will take some of the calculations used in Diffuse lighting such as lightdirection in the first step. Next, we find the viewDirection which is the similar equation instead we use viewposition therefore, viewpositon – fragmentposition. Next calculate the view direction vector and the corresponding reflect vector along the normal axis reflect (-lightdirection, norm). Then calculate the dot product between the view direction and the reflect direction and raise it to the power of shininess value.

Overall result or end result of phong model is the result of adding the ambient, diffuse and specular then multiplying that by the object color.

4 Results

Using all the methods mentioned above I was able to output a 3D cube with Phong shading implemented on it, shown below. Implementing the camera in Camera.h and main.cpp allowed us to see and move freely in the space. Creating the cube with the given vertices in main.cpp along with implementing Phong model in phong.vs and phong.frag.


HW3