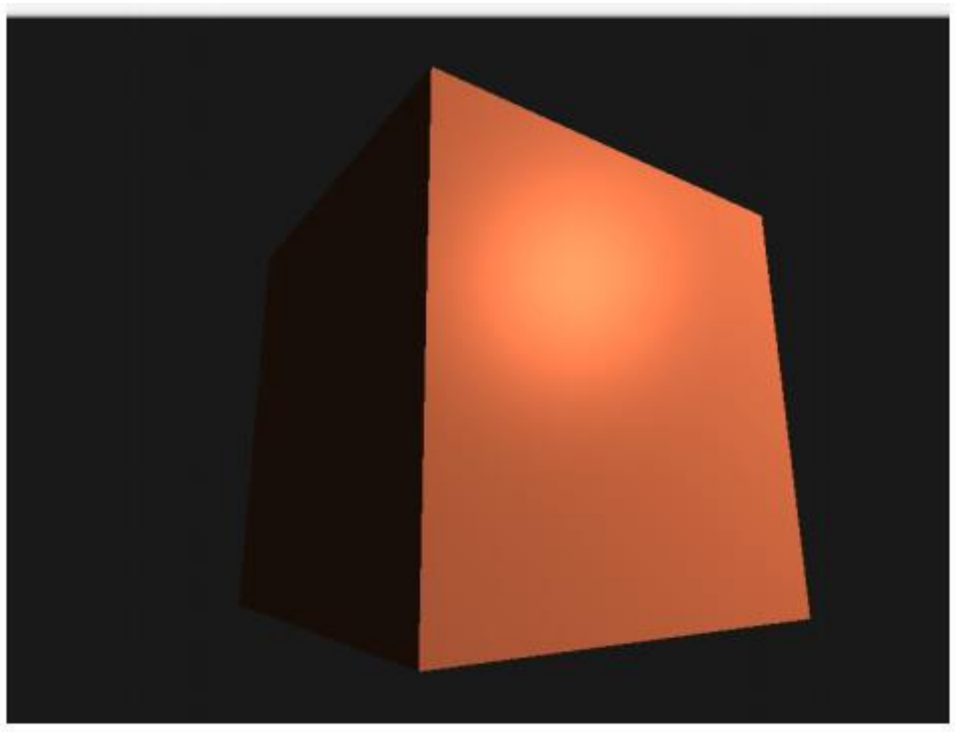
Homework 4 Report

Objective

Practice 3D viewing and implement the Phong shader model using OpenGL.

Use the template code given to reproduce the given image:



Complete the GetViewMatrix() function and projection matrix, then write the vertex and fragment shaders for the Phong model to shade the cube.

Method

First I completed the GetViewMatrix() function by using the lookAt function with arguments already given in the template code. lookAt() takes 3 vectors: camera position, target position, and a vector representing where "up" is in world space.

After returning the view matrix, I set up the projection matrix with OpenGL's projection function. This function sets the camera to view as if you would in the real world rather than orthographically. Parameters were filled with arguments given in template code.

After this I moved to shading. I used https://learnopengl.com/ as a guide.

I did ambient first, then diffuse lighting, then specular lighting.

For ambient lighting, I first multiplied the light color by a constant (0.1). Then, I multiplied the cube's color by the ambient lighting product. This resulted in a darkened cube of a solid color.

For diffuse lighting, the closer a light ray's angle to a surface is to 90 degrees, the brighter it will be. By calculating the angle between the light ray and the normal vector to the surface, I could determine how much the light affects the object's color.

Finally, for specular lighting, the object lighting is affected by the viewing angle as well as the light ray vector and the object's normal vector. If the viewing angle is the same as the light reflection angle, the specular lighting is the strongest at that point.

First, a specular intensity is set depending on the object's intended material. Then, the view direction vector and light reflection vector are calculated. The specular component is then calculated by getting the dot product of the reflection vector and view direction and raising that to a constant value.

```
float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
```

In this case I used 32. The higher the value, the more shiny or glossy it becomes. A lower value scatters the light more while a higher one acts more like a mirror.

At this point ambient lighting, diffuse lighting, and specular lighting had been complete and I used this formula for the final result:

```
vec3 result = (ambient + diffuse + specular) * objectColor;
```

By setting the object color to this, my cube matched very closely to the given image.