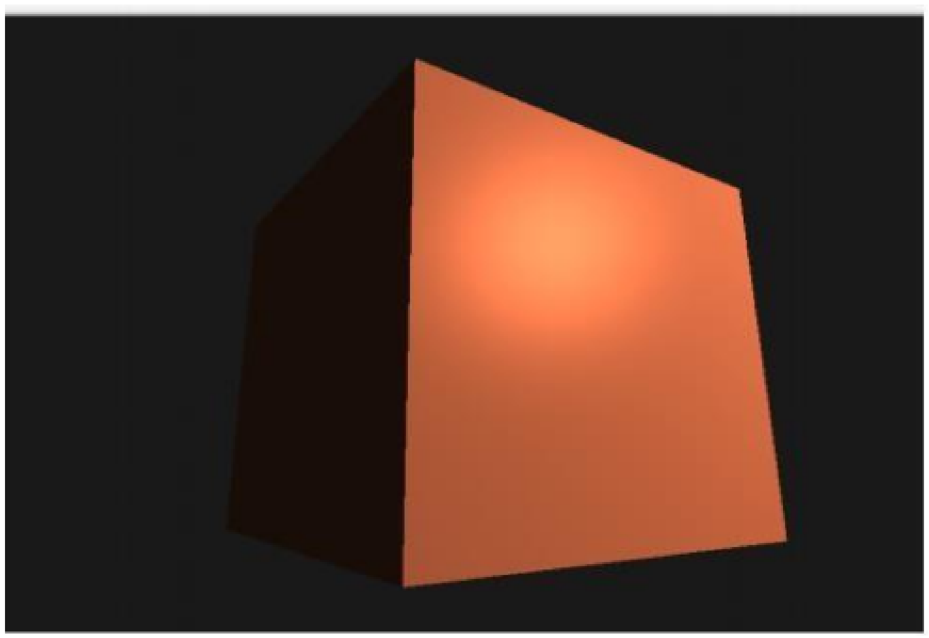
COSC 4370 - Homework 3

Name: Jose Ricardo Sanchez Gonzalez PSID: 1891683

October 27, 2022

1 Problem

The goal of this assignment is to implement the 3D viewing and Phong shading model. By implementing it, the goal is to recreate the following image:



2 Method

First of all, we had to prepare the environment. All we needed was given to us in the files attached with the assignment. After setting it up, we were ready to begin the assignment. We would first have to setup the camera and then focus on implementing the Phong shading model.

3.1 Implementation

The first step was on $main.cpp$, where I first had to setup the project matrix. The code that will allow me to do that is

```
projection = glm::perspective(glm::radians(camera.Zoom),(float)WIDTH / (float)HEIGHT,0.1f,100.0f);
```

Once I added that, my projection matrix was correctly setup.

3.2 Implementation

After setting up the project matrix, what is needed is to setup $GetViewMatrix()$ in $Camera.h$ so it returns the view matrix calculated using Eular Angles and the LookAt Matrix. To achieve that, I implemented:

```
glm::mat4 GetViewMatrix()
    {
      return glm::lookAt(Position, Position + Front, Up);
    }
```

By doing that, GetViewMatrix() was setup.

3.3 Implementation

My next task was to modify the main function in $phong.vs$. In that file, we had the following variables already given:

```
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;

out vec3 Normal;
out vec3 FragPos;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
```

All I had to do was to correctly the gl_position so I could see the cube correctly displaying on my screen. To do that, I set gl_Posititionequals to the following code:

```
gl_Position = projection * view * model * vec4(position, 1.0);
```

After that, I also had to setup Normal and FragPos because they are outgoing variables that I would need later:

```
Normal = normal;
FragPos = vec3(model * vec4(position, 1.0));
```

3.4 Implementation

When gl_Position was setup correctly, and with the following code that was given in $phong.frag$:

```
color = vec4(vec3(1.f,0.f,0.f), 1.0f);
```

I was able to see a working red cube on my screen. What I had to to next, was to correctly implement the Phong shading model correctly. In $phong.frag$ the first thing I was to do was to comment out the code given above, because a red cube was not necessary, but it was just a guide for us to be able to see something on screen before implementing the Phong shading model. The first thing I had to implement was to get the ambient color. I first had to setup an ambientStrength equals to 0.1, and then multiply ambientStrengt times lightColor and store it in ambient.

```
//Getting ambient color
float ambientStrength = 0.1;
vec3 ambient = ambientStrength * lightColor;
```

The next step was to obtain the diffuse color and store it in diffuse. I first normalized the variable Normal and stored in norm. I then normalized the substraction of FragPos to lightPos, and stored it in lightDir. I then stored the max dot of norm, lightDir to finally multiply that times lightColor storing it in diffuse

```
//Getting diffuse color
vec3 norm, lightDir;
norm = normalize(Normal);
lightDir = normalize(lightPos - FragPos); //light position- fragment position
```

```
float diff = max(dot(norm, lightDir), 0.0);

vec3 diffuse = diff * lightColor;
```

Following that, the next step to get the Phong shading model working correctly was tu get the specular light. I first had to set the specularStrength to 0.5. Then normalized the difference between viewPos and FragPos and stored it into viewDir. After that I stored the reflect of -lightDir and norm into reflectDir. I got the spec by the formula and finally got the specular by multiplying the specularStrenght times spec times lightColor.

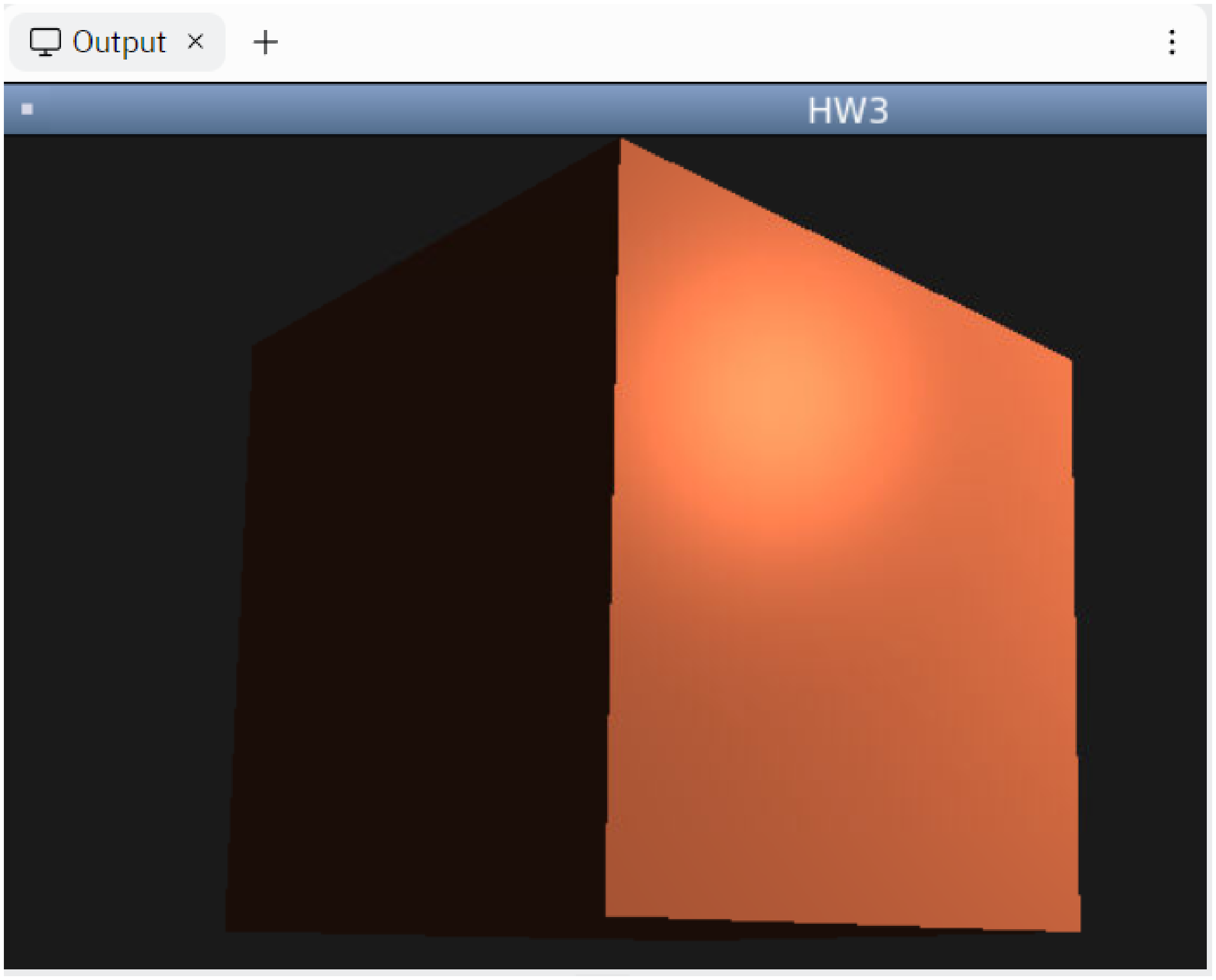
```
//Getting specular light
float specularStrength = 0.5;

vec3 viewDir = normalize(viewPos - FragPos);
vec3 reflectDir = reflect(-lightDir, norm);

float spec = pow(max(dot(viewDir, reflectDir), 0.0), 64);
vec3 specular = specularStrength * spec * lightColor;
```

The final step was to display them all on the screen. I first added the three variables ambient, diffuse and specular and multiplied them by the objectColor. The result was then chosen as part of color. and with that, the code was finished.

```
vec3 result = (ambient + diffuse + specular) * objectColor;
color = vec4(result, 1.0);
```

4 Result

The result is a cube displaying correctly the Phong shading model as intended. You can move the camera as intended too. A picture of the result is the following:



It seems the same as the one given on the assignment.