COSC 4370 – HOMEWORK 3

NAME: AI NGUYEN

PSID: 1392857

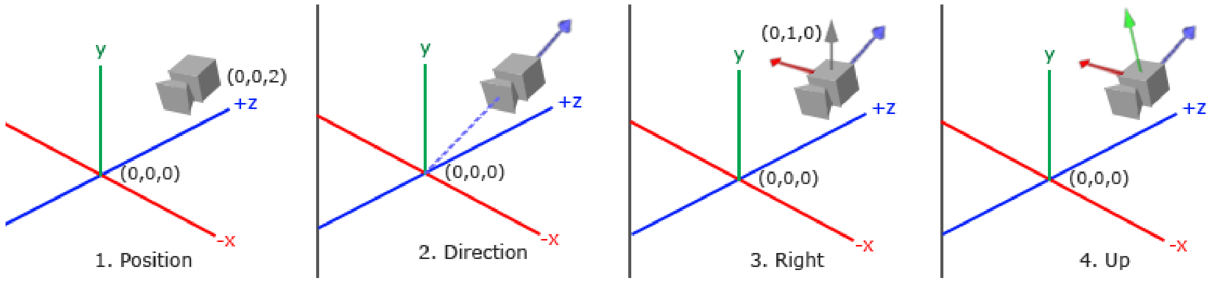
OCTOBER 2021

1. PROBLEM

The assignment requires practicing 3D viewing and dive a little more deeply into OpenGL by implementing the Phong shader model.

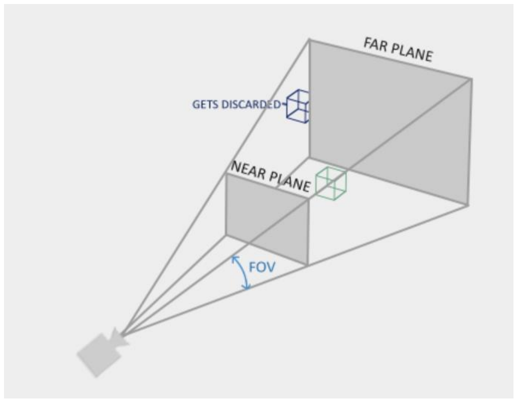
2. METHOD

a) Camera: As shown in the figure, each camera needs to have 4 elements, including position, direction, right axis, and up axis.



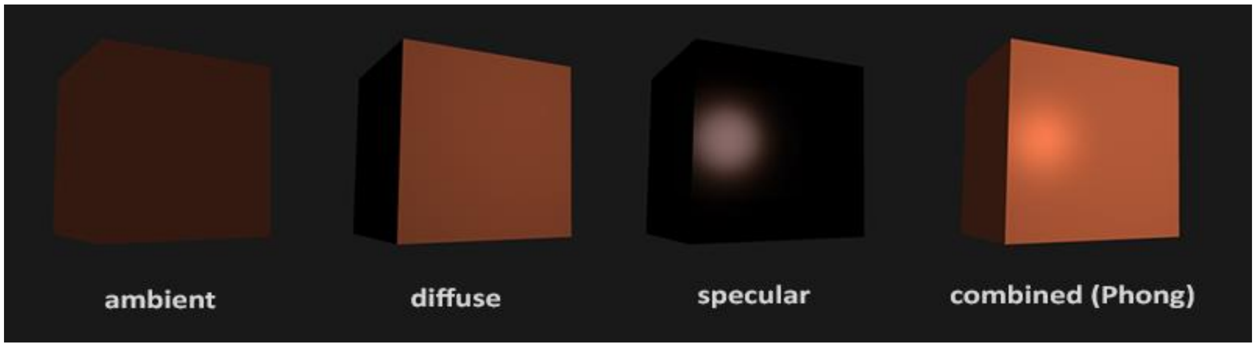
- The position is where the camera is placed to see the object. For example, in the figure the camera is positioned at (0,0,2), that is, on the z-axis and 2 units from the origin.
- The direction is the way the camera pointing to the object, that is, take the vector of the camera position minus the vector of the object.
- The right axis is the vector pointing to the right of the camera, this is the positive x-axis. To find this vector, take the cross product of the upward vector and the direction vector of the camera. The resulting vector will be perpendicular to the two vectors.
- The up axis will be based on the positive y-axis and is found similarly to the right axis, take the cross product between the Right Axis and the direction vector.

b) Coordinate Systems: Projection matrix is one of the transformation matrices used to transform the coordinates from one space to the next coordinate space. It will have two forms each of which will have its own frustum: orthographic projection matrix or a perspective projection matrix.



The perspective of the projection matrix is to create a large frustum. Each coordinate, if it appears in frustum, will also appear on the user's screen. Each element in the vertex coordinates (x, y, z) will be divided by w (width), it makes the object farther away will be smaller from the user's perspective.

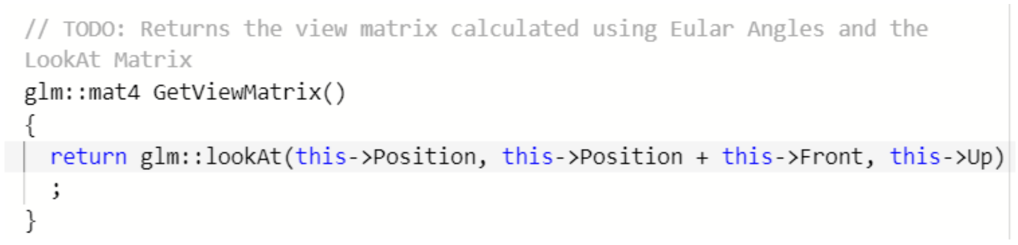
c) Basic Lighting:



- Ambient lighting: In any given condition, at least one source of light shines on the object.
- Diffuse lighting: The main light source shines the object, making the object stand out from the surrounding. The more the surface of an object is exposed to light, the brighter the object will be. Its color is mainly the color of the object, not the color of the light source.
- Specular lighting: This position is the highlight in an object, brighter than the other position. Its color is mainly the color of the light source, not the color of the object.

3. IMPLEMENTATION

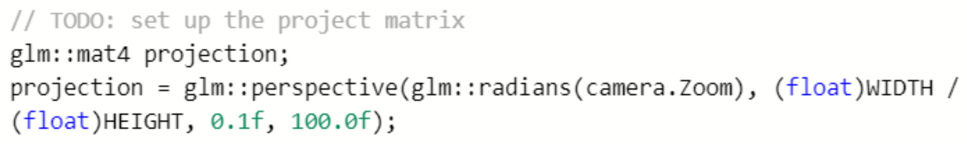
a) Camera



```
// TODO: Returns the view matrix calculated using Eular Angles and the
LookAt Matrix
glm::mat4 GetViewMatrix()
{
  return glm::lookAt(this->Position, this->Position + this->Front, this->Up)
  ;
}
```

The LookAt function requires a camera position vector, a position vector of the object the camera is aiming for, and an up vector.

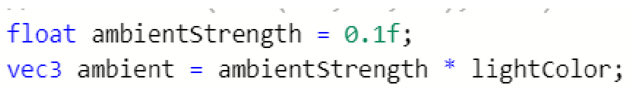
b) Projection matrix



```
// TODO: set up the project matrix
glm::mat4 projection;
projection = glm::perspective(glm::radians(camera.Zoom), (float)WIDTH /
(float)HEIGHT, 0.1f, 100.0f);
```

When the field of view becomes smaller, the scene's projected space gets smaller.

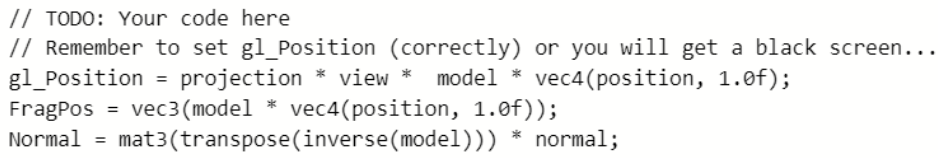
c) Ambient lighting



```
float ambientStrength = 0.1f;
vec3 ambient = ambientStrength * lightColor;
```

Multiply the color of the light by the ambient factor, multiplying by the color of the object. Finally get the result to calculate the fragment's color.

d) Diffuse color: it needs normal vector and directed light ray to calculate



```
// TODO: Your code here
// Remember to set gl_Position (correctly) or you will get a black screen...
gl_Position = projection * view *  model * vec4(position, 1.0f);
FragPos = vec3(model * vec4(position, 1.0f));
Normal = mat3(transpose(inverse(model))) * normal;
```

- Normal vector is the vector perpendicular to the surface.
- Lighting calculations performed in the fragment shader
- Forward the normal vectors from the vertex to the fragment shader
- Calculate the direction vector between the light and the position of fragment
- Applying transpose and inverse function in the matrix shader to get normal vector.

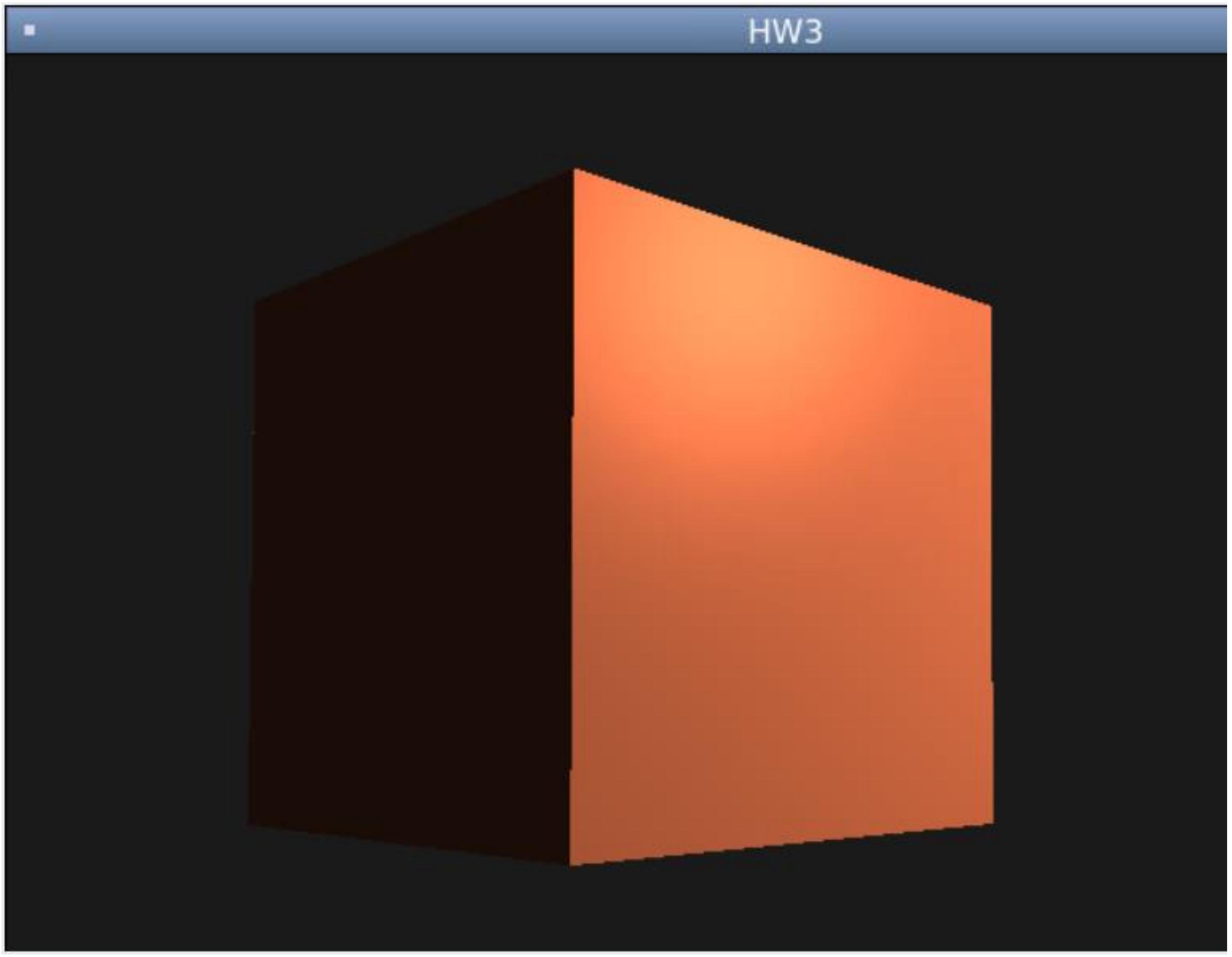
e) Specular lighting

```
float specularStrength = 0.5f;
vec3 viewDir = normalize(viewPos - FragPos);
vec3 reflectDir = reflect(-lightDir, norm);
float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
vec3 specular = specularStrength * spec * lightColor;
```

- 0.5f is just enough because if it's too large, it means that the color of the object is too bright and overwhelms the color of the light source, making the user not see the highlight.
- Calculating dot product view direction and reflect direction.
- Number 32 is the ratio of light to dark of the highlight. The larger the number, the brighter the color of the object, the darker the color of the light source, and vice versa.

4. RESULTS

I use what I learned from Dr. Deng and do further research on the website learnopengl.com to complete this assignment. I feel like my product is almost the same as the question requires.



5. REFERENCE

Basic Lighting - https://learnopengl.com/Lighting/Basic-Lighting

Camera - https://learnopengl.com/Getting-started/Camera

Coordinate System - https://learnopengl.com/Getting-started/Coordinate-Systems

Shaders - https://learnopengl.com/Getting-started/Shaders