**Detail report for COSC4370 Fall 20 HW3 - OpenGL**

**Objective**

Objective of this Project practice little more deeply into OpenGL with 3D scene and dealing with camera and Projection along with implementing the Phong shader model.

**Environment**

In this project we will be using the GLUT/freeglut, GLEW, GLFW and GLM library for our OpenGL code. If you are on Windows, we can find a freeglut 3.0.0 package for Visual Studio (MSVC) at http://www.transmissionzero.co.uk/software/freeglutdevel/. On OS X, GLUT should be installed by default. Finally, if we are on Linux and don’t have GLUT already installed, you should be able to install it by running sudo apt-get install freeglut3-dev or sudo yum install freeglut devel in a terminal. We have written the included Makefile so that the project should compile with it on Linux or OS X. For Windows, we will need to create a project in Visual Studio rather than using the Makefile.

For windows :

Put glut.h inside “C:\Program  
Files(x86)\Microsoft Visual Studio  
10.0\VC\include\**GL**\” (you may need to create the directory **GL with** yourself).  
λ Put glut32.lib inside “C:\Program  
Files\Microsoft Visual Studio 9.0\VC\lib\”.  
**Windows 32-Bit Users**: Put glut32.dll inside  
“C:\Windows\System32\”.  
**Windows 64-Bit Users**: Put glut32.dll inside  
“C:\Windows\SysWOW64\”.

For other Libraries:-

* <https://sourceforge.net/projects/glew/files/glew/1.13.0/>
* <https://www.glfw.org/download.html>
* <https://github.com/g-truc/glm/releases/tag/0.9.9.8>

For linking all these libraries to my project I watched youtube videos which I have mentioned in the reference section.

**Project Details**

In this project I created three scenes differently i.e. one with using ambient light, another with diffuse light and third with specular light. And as a final scene I rendered by complete phong shading model.

For this assignment all files were provided expect phong.vs and phong.frag. I finished Camera.h and main.cpp with few lines of coding which I had to finished in the mentioned TO DO section.

**CODING PART:-**

## Main.cpp

**glm::mat4 view = camera.GetViewMatrix();**

**glm::mat4 projection = glm::perspective(camera.Zoom, (float)WIDTH / (float)HEIGHT, 0.1f, 100.0f);**

**glUniformMatrix4fv(glGetUniformLocation(lightingShader.Program, "view"), 1, GL\_FALSE, glm::value\_ptr(view));**

**glUniformMatrix4fv(glGetUniformLocation(lightingShader.Program, "projection"), 1, GL\_FALSE, glm::value\_ptr(projection));**

**// Set light uniforms**

**glUniform3fv(glGetUniformLocation(lightingShader.Program, "lightPos"), 1, &lightPos[0]);**

**glUniform3fv(glGetUniformLocation(lightingShader.Program, "viewPos"), 1, &camera.Position[0]);**

**// Set Color**

**glUniform3fv(glGetUniformLocation(lightingShader.Program, "color"), 1, &color[0]);**

This is what I wrote to main.cpp

I used glGetUniformLocation, glUniformMatrix4fv, glUniform3fv functions which pass my uniform values to phong.vs(vertex) and phong.frag(fragment).

## Phone.vs

**#version 330 core**

**layout (location = 0) in vec3 position;**

**layout (location = 1) in vec3 normal;**

**layout (location = 2) in vec2 texCoords;**

**out phong {**

**vec3 FragPos;**

**vec3 Normal;**

**vec2 TexCoords;**

**} vs\_out;**

**uniform mat4 projection;**

**uniform mat4 view;**

**void main()**

**{**

**gl\_Position = projection \* view \* vec4(position, 1.0f);**

**vs\_out.FragPos = position;**

**vs\_out.Normal = normal;**

**vs\_out.TexCoords = texCoords;**

**}**

This code is for vertex to view and projection

## Phone.frag

**#version 330 core**

**out vec4 FragColor;**

**in phong {**

**vec3 FragPos;**

**vec3 Normal;**

**vec2 TexCoords;**

**} fs\_in;**

**uniform vec3 lightPos;**

**uniform vec3 viewPos;**

**uniform vec3 color;**

**void main()**

**{**

**// Ambient**

**vec3 ambient = 0.05 \* color;**

**// Diffuse**

**vec3 lightDir = normalize(lightPos - fs\_in.FragPos);**

**vec3 normal = normalize(fs\_in.Normal);**

**float diff = max(dot(lightDir, normal), 0.0);**

**vec3 diffuse = diff \* color;**

**// Specular**

**vec3 viewDir = normalize(viewPos - fs\_in.FragPos);**

**vec3 reflectDir = reflect(-lightDir, normal);**

**float spec = 0.0;**

**vec3 halfwayDir = normalize(lightDir + viewDir);**

**spec = pow(max(dot(normal, halfwayDir), 0.0), 32.0);**

**vec3 specular = vec3(0.3) \* spec;**

**// FragColor = vec4(ambient, 1.0f);**

**// FragColor = vec4(diffuse, 1.0f);**

**// FragColor = vec4(specular, 1.0f);**

**FragColor = vec4(ambient + diffuse + specular, 1.0f);**

**float gamma = 2.2;**

**FragColor.rgb = pow(FragColor.rgb, vec3(1.0/gamma));**

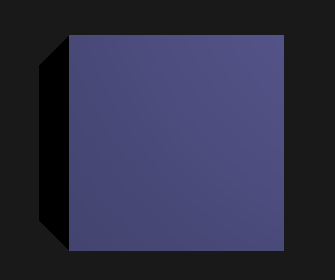
**}**

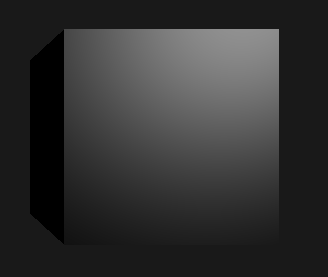
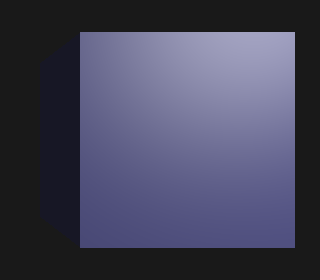
This code is for fragment to set color, texture and light

As you can see, it is allowed to change code to show ambient, diffuse, specular and phong reflection(add all).

**Result:**

After successfully running the code I got following results:-

 Ambient  Diffuse

 Specular  Phong Reflection

**Reference:**

For this assignment our class lecture helped me a lot. Beside our class lecture I took a reference of following links which is youtube video and course:-

* <https://learnopengl.com/>
* <https://www.youtube.com/watch?v=W3gAzLwfIP0&list=PLlrATfBNZ98foTJPJ_Ev03o2oq3-GGOS2&ab_channel=TheCherno>