Assignment 1:
Exploring OpenGL and Phong Lighting

NAME: BINGNAN LI
STUDENT NUMBER: 2020533092
EMAIL: LIBN@SHANGHAITECH.EDU.CN

1 INTRODUCTION

- Loading mesh file from .obj files has been done.
- Customized camera systems based on Euler angles has been done.
- Phong lighting system with ambient, diffuse and specular lights has been done.
- Multi-lights and spotlights system has been done.
- Play with geometry shader.

2 IMPLEMENTATION DETAILS

(1) Loading mesh from .obj files:

First loaded vertices into self.Vertices, then I loaded normal vectors into a temporary variable. Lastly, I merge them by the indices given by faces.

(2) VAO, VBO, EBO and shaders:

I created VAO, VBO, EBO in mesh class, then I binded and transferred data to VBO and EBO.

For shaders, I compiled shaders in shader class and link them to shader program.

(3) Camera system:

I initialized yaw angle to -90 degree since we initially face to negative z direction. Then I use pitch and yaw angles to calculate Front variable, which is used to indicate the view direction. Then we set positive y direction unit vector as WorldUp variable to indicate the unchanged up direction in world coordinate. With Front and WorldUp, we can calculate Right base with cross product of Front and WorldUp. After that, we can get Up base by calculating cross product of Right and Front. Thus, we built the camera coordinates with (Front, Right, Up).

Then I build the calling functions of keyboard inputs and Cursor movements. For forward and backward, leftward and rightward movements, upward and downward movements, camera position changes along Front, Right and Up vectors respectively. Then for cursor movements, I stored last cursor position in last frame and calculate difference with new cursor position. What needs to be specified is that the position that function "glfwSetCursorPosCallback" returns is the relative position from the left-up corner of display. Then we use the offset along two axes to adjust the value of Yaw and Pitch so that we can calculate new Front with those two angles.

(4) Phong lighting:

Phong lighting has three components: ambient, diffuse and specular.

For ambient light, I set an ambient strength to control the intensity of ambient light, then I multiply ambient strength with light color as the ambient light.

For diffuse light, I calculated the dot product of inverse light direction and normal vector as the diffuse intensity, then I multiply this intensity with light color as diffuse light.

For specular light, I first calculate the reflection light, then I calculated the dot product of reflection light and view direction. After that I calculated the 32 pow of this dot product in order to get a more smooth high light. Finally, I use multiply it with specular intensity and light color as specular light.

Lastly, I added those three parts together and then multiply it with object color as the result color.

(5) Multi-light and spotlight system:

I instantiated a new shader to render spheres in each light sources to represent real bodies of light sources.

I render three kinds of lights: directional light, point light and spotlight moving with camera. For each kind of light, I build different render functions in fragment shader source file.

First, for directional light, the corresponding rendering function is similar to basic phong lighting system.

Second, for point light, I added a decay into the result light with distance. I firstly calculated the distance between fragment point and light source, then I applied

$$attenuation = \frac{1}{K_0 + K_1 \times distance + K_2 * distance^2}$$

to get the decay factor, then I multiply it with result to implement light intensity decay.

Third, for spotlight, I firstly set light position to camera position and set light direction to camera Front so that the spotlight we move with camera. Then I set a cut-off angel to specify the range of lighting. For each fragment points, I determine whether it is in the range of the spotlight. In order to smooth light circle edge, I added an outer cut-off angle variable, for fragment position between cut-off angle and outer cut-off angle, the intensity of light we decay linearly.

(6) Play with geometry:

For each input triangle primes, I first calculated the geometry center of the prime by averaging three vertices, then I calculated a new normal vector by averaging three normal vectors of vertices. Next I translate the geometry center along with the outer normal direction. Then I add three new faces

by combine any two of the three vertices and the new generated point. Finally, I calculated the corresponding normal vectors by cross product.

3 RESULTS

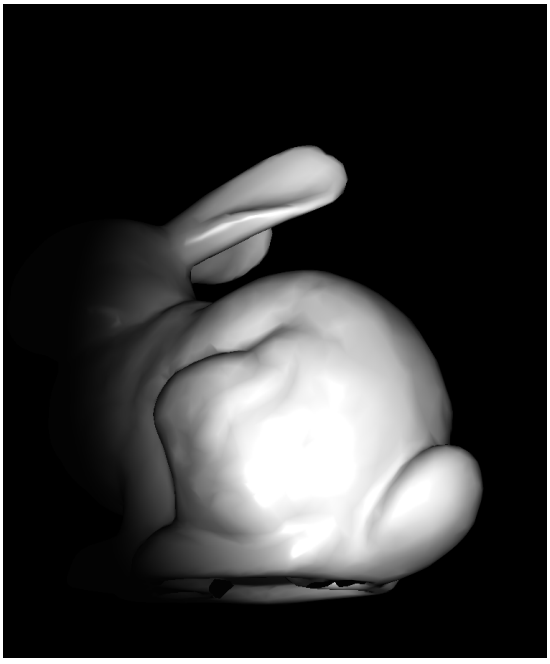

Fig. 1. Bunny with only spotlight system and phong lighting system

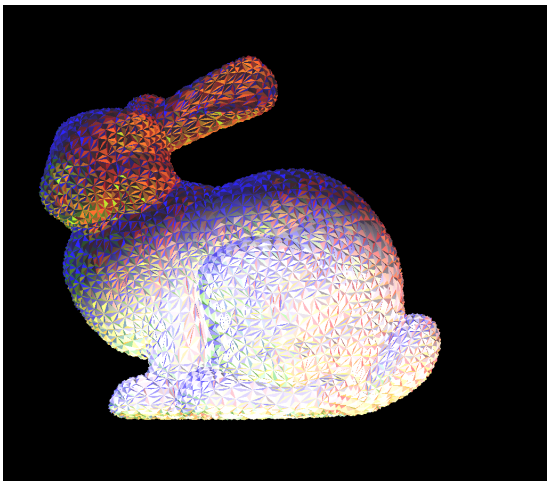

Fig. 2. Bunny with geometry shader and multi-light system

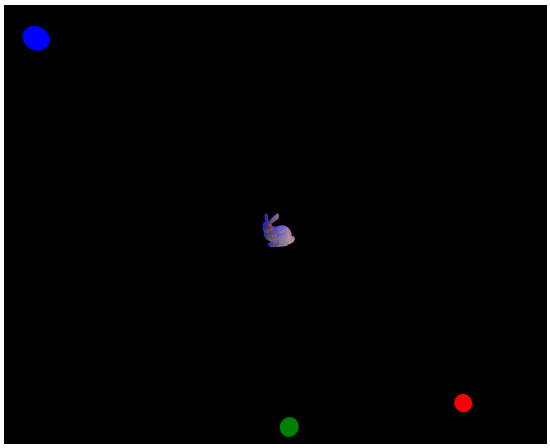

Fig. 3. overview of the whole scene