Homework 6 – lights and shading

注：本次作业的完整演示请见/doc/pic/*.gif

Basic:

1. **实现 Phong 光照模型：**
   - **（1） 场景中绘制一个 cube**

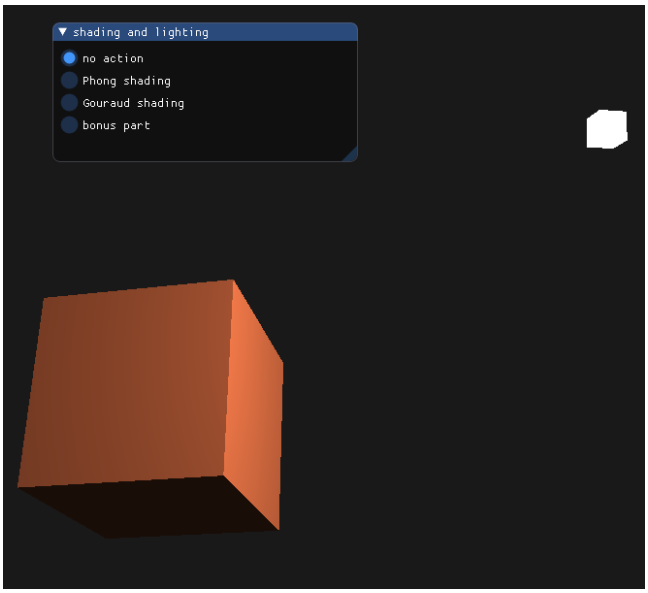
     和前面的作业过程相同，定义顶点坐标，通过几次变换操作将 cube 放置到合适的大小、位置。重写 lampshader 如下：

     lampshader.vs:

```
#version 330 core
layout(location = 0) in vec3 aPos;
uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
void main() {
    gl_Position = projection * view * model * vec4(aPos, 1.0);
}
```

     lampshader.fs:

```
#version 330 core
out vec4 FragColor;
void main() {
    FragColor = vec4(1.0);
}
```



(2) 自己写 shader 实现两种 shading: Phong Shading 和 Gouraud Shading，并解释两种 shading 的实现原理

使用 GLSL 编写 Phong 和 Gouraud shading 的着色器文件，并在渲染时使用之前的 shader 类进行处理。

Phong Shading:

冯氏光照模型由环境光照、漫反射光照和镜面光照三个分量组成，其中，环境光照是用光的颜色乘以一个很小的常量环境因子，再乘以物体的颜色，然后将最终结果作为片段的颜色；漫反射光照需要法向量和光源及片段的位置向量，在片段着色器中标准化得到光的方向向量，并与法向量计算点乘，结果乘以光的颜色就得到了漫反射向量（需要注意点乘取非负值）；对于镜面光照，则需要定义镜面强度、反光度，计算沿法线轴的反射向量和镜面分量。

pshader.vs

```
#version 330 core
layout(location = 0) in vec3 aPos;
layout(location = 1) in vec3 aNormal;
out vec3 FragPos;
out vec3 Normal;
uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main() {
    FragPos = vec3(model * vec4(aPos, 1.0));
    Normal = mat3(transpose(inverse(model))) * aNormal;
    gl_Position = projection * view * vec4(FragPos, 1.0);
}
```

pshader.fs

```
#version 330 core
out vec4 FragColor;
in vec3 Normal;
in vec3 FragPos;
uniform vec3 lightPos;
uniform vec3 lightColor;
uniform vec3 objectColor;
uniform vec3 viewPos;
void main() {
    float ambientStrength = 0.1;
```

```
    vec3 ambient = ambientStrength * lightColor;

    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(lightPos - FragPos);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;

    float specularStrength = 0.5;
    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;

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

Gouraud Shading:
实际上是在顶点着色器中实现的冯氏光照模型，顶点着色器的颜色是顶点的颜色值，片段的颜色值是由插值光照颜色所得来的。

gshader.vs

```
#version 330 core

layout(location = 0) in vec3 aPos;
layout(location = 1) in vec3 aNormal;

out vec3 LightingColor;

uniform vec3 lightPos;
uniform vec3 lightColor;
uniform vec3 viewPos;

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

void main() {
    gl_Position = projection * view * model * vec4(aPos, 1.0);
    vec3 Position = vec3(model * vec4(aPos, 1.0));
```

```
    vec3 Normal = mat3(transpose(inverse(model))) * aNormal;

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

    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(lightPos - Position);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;

    float specularStrength = 1.0;
    vec3 viewDir = normalize(viewPos - Position);
    vec3 reflectDir = reflect(-lightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
    vec3 specular = specularStrength * spec * lightColor;
    LightingColor = ambient + diffuse + specular;

}
```

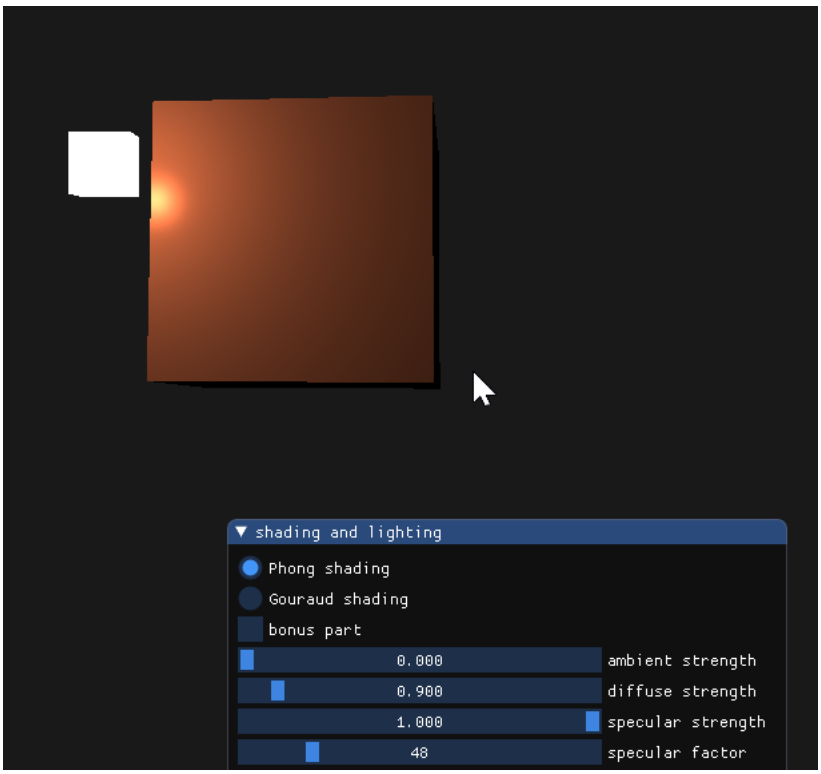
gshader.fs

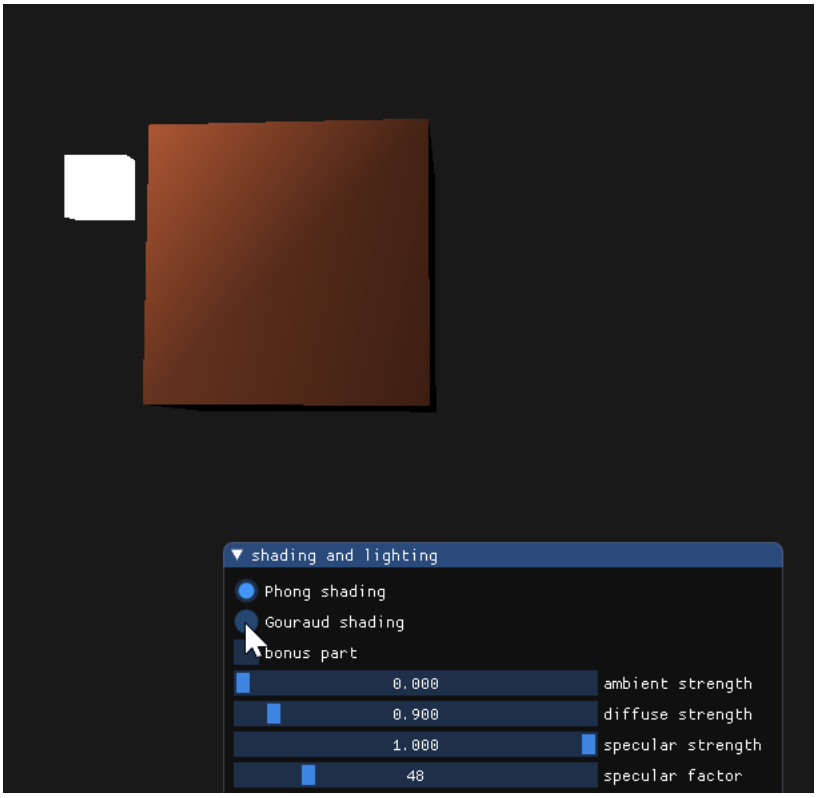
```
#version 330 core
out vec4 FragColor;
in vec3 LightingColor;
uniform vec3 objectColor;

void main() {
    FragColor = vec4(LightingColor * objectColor, 1.0);
}
```

（3）合理设置视点、光照位置、光照颜色等参数，使光照效果明显显示

对比效果如下图, phong shader 的效果更自然。





2. **使用 GUI，使参数可调节，效果实时更改：**

   （1） GUI 里可以切换两种 shading

   使用 imgui 的 radiobutton 切换不同 shading

   ```
   ImGui::Begin("shading and lighting");
   ```

```
ImGui::RadioButton("Phong shading", &mode, 1);
ImGui::RadioButton("Gouraud shading", &mode, 2);
ImGui::Checkbox("bonus part", &rotating);
```

（2）使用如进度条这样的控件，使 ambient 因子、diffuse 因子、specular 因子、反光度等参数可调节，光照效果实时更改

GUI 部分:

```
ImGui::SliderFloat("ambient strength", &ambient, 0.0f, 1.5f);
ImGui::SliderFloat("diffuse strength", &diffuse, 0.5f, 5.0f);
ImGui::SliderFloat("specular strength", &specular, 0.1f, 1.0f);
ImGui::SliderInt("specular factor", &refl, 0, 256);
```

shader 部分:

```
shader.setFloat("ambientStrength",ambient);
shader.setFloat("diffuseStrength", diffuse);
shader.setFloat("specularStrength", specular);
shader.setInt("refl", refl);
```

并且在 shader 的 glsl 文件内定义:

```
uniform float ambientStrength;
uniform float specularStrength;
uniform float diffuseStrength;
uniform int refl;
```

完整演示见/doc/pic/example.gif

Bonus:
当前光源为静止状态，尝试使光源在场景中来回移动，光照效果实时更改。

只需要在前面的基础上加入判断即可:

```
if (rotating) {
    lightPos.x = sin(glfwGetTime()) * 1.0f;
    lightPos.z = cos(glfwGetTime()) * 1.0f;
}
```

其中 rotating 根据 checkbox 是否勾选取值。
具体效果见**/doc/pic/bonus.gif**