中山大學
SUN YAT-SEN UNIVERSITY

《计算机图形学作业》

实验报告

（作业六）

学 院 名 称 ：	数据科学与计算机学院
学 生 姓 名 ：	姚雪辉
学　　号 ：	15355119
专业（班级）：	16 软件工程四（7）班
时　　间 ：	2019 年 5 月 4 日

Basic

Phong Shading

Phong 光照模型 = 环境光照+漫反射光照+镜面光照

计算环境光照：不需要考虑法向量，光线方向，只要对光照本身处理就可以了。

```
vec3 ambient = ambientStrength * lightColor;
```

漫反射光照：需要（1）法向量：一个垂直于顶点表面的向量。（2）定向的光线：作为光源的位置与片段的位置之间向量差的方向向量。所以在顶点着色器中要加入法向量aNormal，传递给片段着色器，并且转换成世界坐标系，然后分别得到法向量和定向的光线，归一化之后，点乘得到影响的程度，取最大值是为了防止影响为负数。

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

镜面光照：需要（1）视点的方向：摄像机位置与片段的位置之间向量差的方向向量，（2）反射方向：光线方向与法向量的影响。

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

顶点着色器：

```
const char *lightVertexShaderSource1 = "#version 330 core\n"
"layout (location = 0) in vec3 aPos;\n"
"layout (location = 1) in vec3 aNormal;\n"
"out vec3 FragPos;\n"
"out vec3 Normal;\n"
"uniform mat4 model;\n"
"uniform mat4 view;\n"
"uniform mat4 projection;\n"
"void main()\n"
"{\n"
"    FragPos = vec3(model * vec4(aPos,1.0));\n"
"    Normal = mat3(transpose(inverse(model)))*aNormal;\n"
"    gl_Position = projection * view * vec4(FragPos, 1.0);\n"
"}\0";

const char *objectFragmentShaderSource1 = "#version 330 core\n"
"out vec4 FragColor;\n"
```

```
"in vec3 Normal;\n"
"in vec3 FragPos;\n"
"uniform vec3 lightPos;\n"
"uniform vec3 viewPos;\n"
"uniform vec3 lightColor;\n"
"uniform vec3 objectColor;\n"
"uniform float ambientStrength;\n"
"uniform float diffuseStrength;\n"
"uniform float specularStrength;\n"
"void main()\n"
"{\n"
"    vec3 ambient = ambientStrength * lightColor;\n"
"    vec3 norm = normalize(Normal);\n"
"    vec3 lightDir = normalize(lightPos - FragPos);\n"
"    float diff = max(dot(norm, lightDir), 0.0);\n"
"    vec3 diffuse = diffuseStrength * diff * lightColor;\n"
"    vec3 viewDir = normalize(viewPos - FragPos);\n"
"    vec3 reflectDir = reflect(-lightDir, norm);\n"
"    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);\n"
"    vec3 specular = specularStrength * spec * lightColor;\n"
"    vec3 result = (ambient + diffuse + specular) * objectColor;\n"
"    FragColor = vec4(result, 1.0);\n"
"}\n";
```

Gouraud Shading

Gouraud 模型和 Phong 模型的不同在于，gouraud 的光照是在顶点着色器中实现的，而 Phong 模型是在片段着色器中实现的。

顶点着色器:

```
const char *lightVertexShaderSource2 = "#version 330 core\n"
"layout(location = 0) in vec3 aPos;\n"
"layout(location = 1) in vec3 aNormal;\n"
"out vec3 LightingColor;\n"
"uniform vec3 lightPos;\n"
"uniform vec3 viewPos;\n"
"uniform vec3 lightColor;\n"
"uniform mat4 model;\n"
"uniform mat4 view;\n"
"uniform mat4 projection;\n"
"uniform float ambientStrength;\n"
"uniform float diffuseStrength;\n"
"uniform float specularStrength;\n"
"void main()\n"
"{\n"
```

```
"    gl_Position = projection * view * model * vec4(aPos, 1.0);\n"
"    vec3 Position = vec3(model * vec4(aPos, 1.0));\n"
"    vec3 Normal = mat3(transpose(inverse(model))) * aNormal;\n"
"    vec3 ambient = ambientStrength * lightColor;\n"
"    vec3 norm = normalize(Normal);\n"
"    vec3 lightDir = normalize(lightPos - Position);\n"
"    float diff = max(dot(norm, lightDir), 0.0);\n"
"    vec3 diffuse = diffuseStrength * diff * lightColor;\n"
"    vec3 viewDir = normalize(viewPos - Position);\n"
"    vec3 reflectDir = reflect(-lightDir, norm);\n"
"    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);\n"
"    vec3 specular = specularStrength * spec * lightColor;\n"
"    LightingColor = ambient + diffuse + specular;\n"
"}\n";
```

片段着色器：

```
const char *objectFragmentShaderSource2 = "#version 330 core\n"
"out vec4 FragColor;\n"
"in vec3 LightingColor;\n"
"uniform vec3 objectColor;\n"
"void main()\n"
"{\n"
"    FragColor = vec4(LightingColor * objectColor, 1.0);\n"
"}\n";
```

切换 shading&参数调节

切换 shading

把编译 shader 的那部分放到循环体里面，用三个参数控制， isPhongShading，isGouraudShading，isChanging 分别表示是否是 Phong 光照模型，Gouraud 光照模型，以及是否需要重新编译。这样可以保证切换不同的 shader 的时候，都只编译一次。

整体结构类似如下

```
        if (isPhongShading && isChanging) {
            isChanging = false;
            ...
        }
        if (isGouraudShading && isChanging) {
            isChanging = false;
            ...
        }
```

```
ImGui::Checkbox("isPhongShading", &isPhongShading);
if (isPhongShading) {
    isGouraudShading = false;
    isChanging = true;
}
ImGui::Checkbox("isGouraudShading", &isGouraudShading);
if (isGouraudShading) {
    isPhongShading = false;
    isChanging = true;
}
```

参数调节

如果参数固定的话，是在 glsl 的 main 函数中申明，这样外界无法改变参数，所以将三个参数申明为 uniform,然后通过 glUniform1f 函数改编参数。

```
"uniform float ambientStrength;\n"
"uniform float diffuseStrength;\n"
"uniform float specularStrength;\n"

unsigned int ambientStrengthLoc = glGetUniformLocation(objectShader,"ambientStrength");
glUniform1f(ambientStrengthLoc, ambientStrength);
unsigned int diffuseStrengthLoc = glGetUniformLocation(objectShader,"diffuseStrength");
glUniform1f(diffuseStrengthLoc, diffuseStrength);
unsigned int specularStrengthLoc =
glGetUniformLocation(objectShader,"specularStrength");
glUniform1f(specularStrengthLoc, specularStrength);
```

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

这部分比较简单，只要改变 lightPos 就可以了，因为 glm::vec3 lightPos(1.2, 1.0f, 2.0f);设定为开始的变量，在 shader 中会使用到该变量，所以改变 lightPos 的值会直接改变光照的影响，而不能直接改变代表 light 的 cube 的 position。