专业：计算机科学与技术
姓名：方彦祺
学号：3220102829
日期：2024.12.17
地点：无

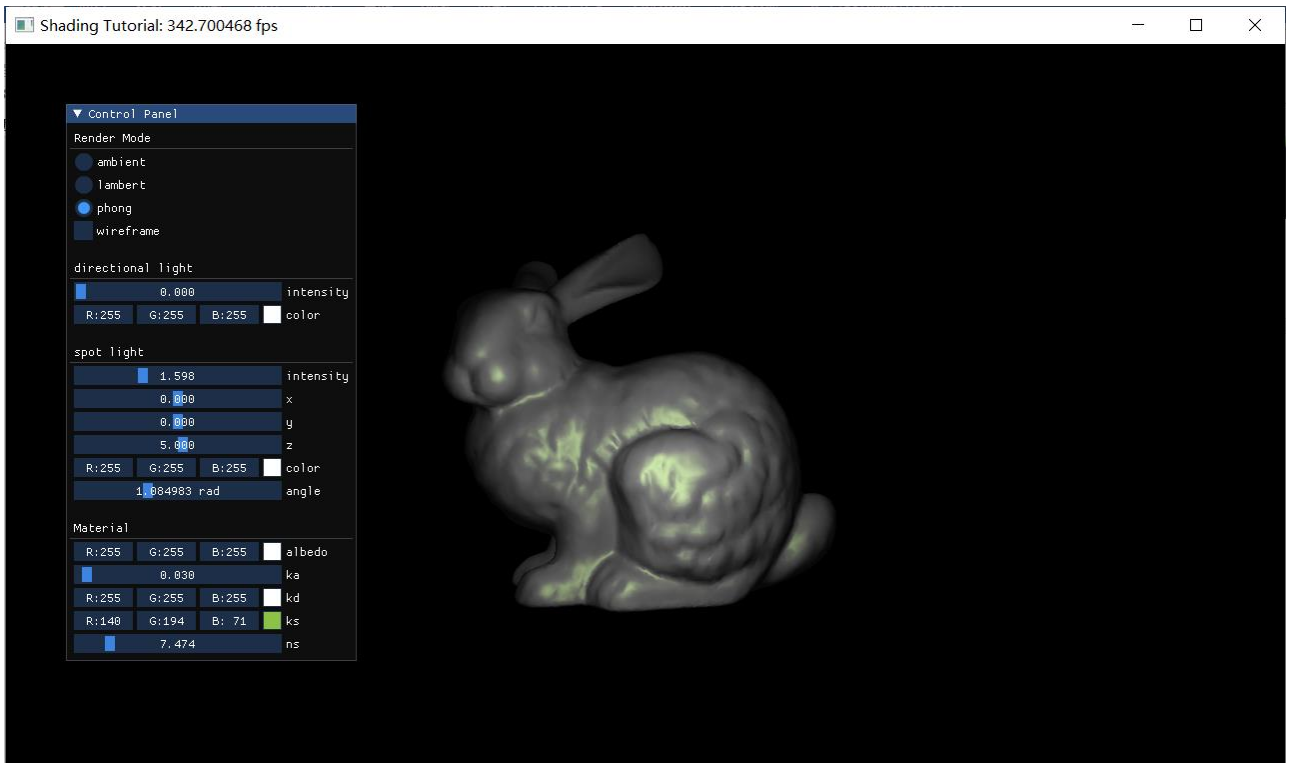
浙江大学实验报告

课程名称：计算机图形学 指导老师：童若锋 成绩：
实验名称：OpenGL 光照模型 实验类型：基础实验 同组学生姓名：无

装 订 线

一、实验目的和要求

掌握着色器语言 GLSL 的基本语法，类比 Lambert 光照模型，利用 GLSL 书写 Phong 光照模型顶点着色器与片段着色器代码，并实现材质对平行光与聚光灯等光源的相应。



二、实验内容和原理

本次实验需要我们填写的 TODO 部分主要涉及的是风氏光照模型(Phong Lighting Model)的部分，在这里参考给出的参考链接介绍有关该模型的原理。

风氏光照模型的主要结构由 3 个分量组成：环境(Ambient)、漫反射(Diffuse)和镜面(Specular)光照。为这三个分量指定一个颜色，我们就能够对表面的颜色输出有细粒度的控制了。然后再添加一个反光度(Shininess)分量，结合上述的三个颜色，我们就有了全部所需的材质属性。ambient 材质向量定义了在环境光照下这个表面反射的是什么颜色，通常与表面的颜色相同。diffuse 材质向量定义了在漫反射光照下表面的颜色。漫反射颜色（和环境光照一样）也被设置为我们期望的物体颜色。specular 材质向量设置的是表面上镜面高光的颜色（或者甚至可能反映一个特定表面的颜色）。最后，shininess 影响镜面高光的散射/半径。

此外，还有关于投光物的理论原理，将光投射(Cast)到物体的光源叫做投光物(Light Caster)。分类包括定向光(Directional Light)、点光源(Point Light)、聚光(Spotlight)。当我们使用一个假设光源处于无限远处的模型时，它就被称为定向光，因为它的所有光线都有着相同的方向，它与光源的位置是没有关系的。点光源是处于世界中某一个位置的光源，它会朝着所有方向发光，但光线会随着距离逐渐衰减。聚光是位于环境中某个位置的光源，它只朝一个特定方向而不是所有方向照射光线。这样的结果就是只有在聚光方向的

特定半径内的物体才会被照亮，其它的物体都会保持黑暗。OpenGL 中聚光是用一个世界空间位置、一个方向和一个切光角(Cutoff Angle)来表示的。在本次实验的 TODO 部分需要我们使用多个光源。

三、主要仪器设备

Visual Studio 2022

四、操作方法和实验步骤

我们给出 TODO 部分中填写的代码，逐一对其进行分析。

```
const char* fsCode =
    "#version 330 core\n"
    "in vec3 fPosition;\n"
    "in vec3 fNormal;\n"
    "out vec4 color;\n"

    // material data structure declaration
    "struct Material {\n"
    "    vec3 ka; // ambient reflectivity\n"
    "    vec3 kd; // diffuse reflectivity\n"
    "    vec3 ks; // specular reflectivity\n"
    "    float ns; // shininess\n"
    "};\n"

    // light data structure declaration
    "struct DirectionalLight {\n"
    "    vec3 direction;\n"
    "    vec3 color;\n"
    "    float intensity;\n"
    "};\n"

    "struct AmbientLight {\n"
    "    vec3 color;\n"
    "    float intensity;\n"
    "};\n"

    "struct Spotlight {\n"
    "    vec3 position;\n"
    "    vec3 direction;\n"
    "    vec3 color;\n"
    "    float intensity;\n"
    "    float angle;\n"
    "    float kc; \n"
    "    float kl;\n"
    "    float kq;\n"
    "};\n"
```

```
"vec3 calcDirectionalLight(vec3 normal) {\n"
"    vec3 lightDir = normalize(-directionalLight.direction);\n"
"    vec3 diffuse = directionalLight.color * max(dot(lightDir, normal), 0.0f) * "
"material.kd;\n"
"    return directionalLight.intensity * diffuse ;\n"
"}\n"

"vec3 calcSpotLight(vec3 normal) {\n"
"    vec3 lightDir = normalize(spotLight.position - fPosition);\n"
"    float theta = acos(-dot(lightDir, normalize(spotLight.direction)));\n"
"    if (theta > spotLight.angle) {\n"
"        return vec3(0.0f, 0.0f, 0.0f);\n"
"    }\n"
"    vec3 diffuse = spotLight.color * max(dot(lightDir, normal), 0.0f) * material.kd;\n"
"    float distance = length(spotLight.position - fPosition);\n"
"    float attenuation = 1.0f / (spotLight.kc + spotLight.kl * distance + spotLight.kq * "
"distance * distance);\n"
"    return spotLight.intensity * attenuation * diffuse;\n"
"}\n"
```

这部分完成对 Phong 模型着色器的初始化，上面给出的部分均可以从已有的代码中直接 copy，我们着重分析自己完成的的代码部分。

```
"vec3 calcSpecularLight(vec3 normal) {\n"
"    vec3 lightDir = normalize(lightPos - fPosition);"
"    vec3 cameraDir = normalize(cameraPos - fPosition);"
"    vec3 reflectDir = reflect(-lightDir, normal);"
"    float spec = pow(max(dot(cameraDir, reflectDir), 0.0), material.ns);"
"    return material.ks * spec * lightColor;"
"}\n"
```

上面这部分计算镜面光的部分，各句所做的工作分别为:归一化光源方向；归一化视角（相机）方向；计算反射方向；计算镜面反射；返回镜面光的贡献值。

```
"void main() {\n"
"    vec3 normal = normalize(fNormal);\n"
"    vec3 diffuse = calcDirectionalLight(normal) + calcSpotLight(normal);\n"
"    vec3 ambient = material.ka * ambientLight.color * ambientLight.intensity;\n"
"    vec3 specular = calcSpecularLight(normal);"
"    vec3 result = ambient + diffuse + specular;"
"    color = vec4(result,1.0f);"
"}\n";
```

主函数部分，计算 Phong 模型的各项参数的实际值，并相加得到最终颜色。

```
// 2. TODO: transfer the camera position to the shader
// write your code here
// ----------------------------------------------------------------
// _phongShader->set...
// ----------------------------------------------------------------
_phongShader->setUniformVec3("cameraPos", _camera->transform.position);
```

相机位置用于计算视线方向，该句为相机在世界空间中的位置。

```
// 3. TODO: transfer the material attributes to the shader
// write your code here
// ----------------------------------------------------------------
// _phongShader->set...
// ----------------------------------------------------------------
_phongShader->setUniformVec3("material.ka", _phongMaterial->ka);
_phongShader->setUniformVec3("material.kd", _phongMaterial->kd);
_phongShader->setUniformVec3("material.ks", _phongMaterial->ks);
_phongShader->setUniformFloat("material.ns", _phongMaterial->ns);
```

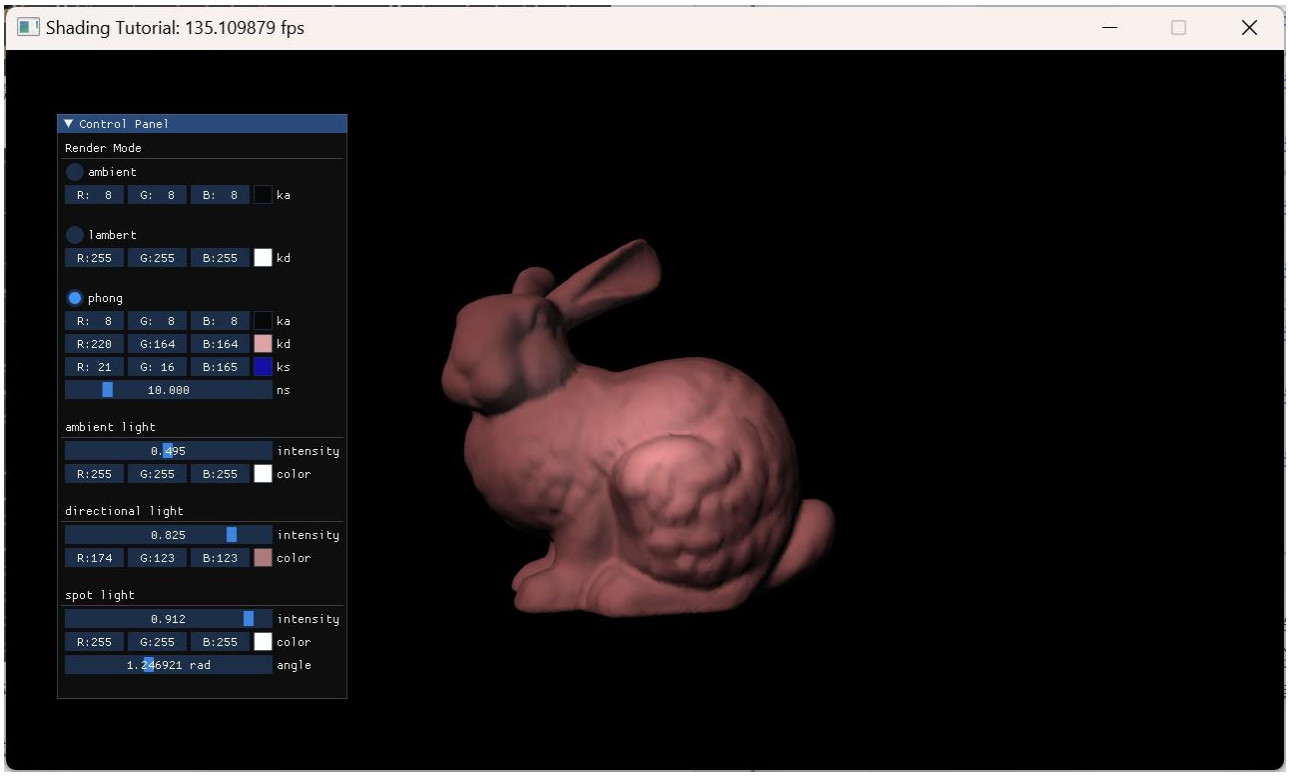
将材质属性传递给着色器，包括Phong模型的三个组成部分环境(Ambient)、漫反射(Diffuse)和镜面(Specular)光照的反射率以及材料的光泽度。

```
// 4. TODO: transfer the light attributes to the shader
// write your code here
// ----------------------------------------------------------
// _phongShader->set...
// ----------------------------------------------------------
_phongShader->setUniformVec3("ambientLight.color", _ambientLight->color);
_phongShader->setUniformFloat("ambientLight.intensity", _ambientLight->intensity);

_phongShader->setUniformVec3("directionalLight.direction", _directionalLight->transform.getFront
_phongShader->setUniformFloat("directionalLight.intensity", _directionalLight->intensity);
_phongShader->setUniformVec3("directionalLight.color", _directionalLight->color);

_phongShader->setUniformVec3("spotLight.position", _spotLight->transform.position);
_phongShader->setUniformVec3("spotLight.direction", _spotLight->transform.getFront());
_phongShader->setUniformFloat("spotLight.intensity", _spotLight->intensity);
_phongShader->setUniformVec3("spotLight.color", _spotLight->color);
_phongShader->setUniformFloat("spotLight.angle", _spotLight->angle);
_phongShader->setUniformFloat("spotLight.kc", _spotLight->kc);
_phongShader->setUniformFloat("spotLight.kl", _spotLight->kl);
_phongShader->setUniformFloat("spotLight.kq", _spotLight->kq);
```

将光源属性传递给着色器，包括环境光源、方向光源和聚光源的各项属性。

五、实验结果与分析

在这里给出一些尝试调整各项参数的截图，具体请见生成的.exe 文件。



六、讨论、心得

本次实验涉及的原理较多，但实际需要完成的部分较为简单，着色器的编写部分参照已给出的代码能完成大部分，将个属性传递给着色器的值这一操作也较为公式化，只要设置对应的变量值即可。

七、参考链接

[1] 颜色 - LearnOpenGL CN (learnopengl-cn.github.io)

[2] 基础光照 - LearnOpenGL CN (learnopengl-cn.github.io)

[3] 材质 - LearnOpenGL CN (learnopengl-cn.github.io)

[4] 投光物 - LearnOpenGL CN (learnopengl-cn.github.io)

[5] 多光源 - LearnOpenGL CN (learnopengl-cn.github.io)