计算机图形学与虚拟现实

实验报告

2025秋

课程类型 限选□ 选修□

实验题目 **冯氏光照模型的绘制**

姓　名 **邴海诺**

学　号 **2024112353**

学　院 **计算学部**

2025 年　月　日

一、实验目的

了解冯氏光照模型的基本原理，实现基础的光照模型，包括环境光照、漫反射光照和镜面光照

二、实验要求及实验环境

```
OpenGL version: 4.6 (Compatibility Profile) Mesa 25.2.7-cachyos1.2
GLEW version: 2.2.0
```

三、设计思想（本程序中的用到的主要算法及数据结构，没有填写无）

expr3.frag 里面：

环境光：ambient = ambientStrength * lightColor。

漫反射：

```
// 漫反射（Lambert）
// 用法线和光向量的点乘来计算漫反射强度，取 0 到 1
float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = diff * lightColor; // 漫反射分量（与表面材料的颜色相乘）
```

镜面反射：

```
//  镜面高光（Phong）
// dot(viewDir, reflectDir) 越接近 1，说明视线更接近反射方向 -> 高光越强
// pow(..., 32) 控制高光聚集程度（高 -> 更小更亮的高光斑）
float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
vec3 specular = specularStrength * spec * lightColor; // 镜面分量
```

最终：color = (ambient + diffuse) * objColor + specular

```
// ambient 和 diffuse 与物体颜色相乘（表示被物体吸收/反射的光）
// 而 specular 为高光（通常由光源决定，与物体颜色无关）
vec3 result = (ambient + diffuse) * objColor + specular;
```

四、实验结果与分析（根据章节分节）

1. 先定义环境光，环境光是全局的、无方向的散射光，模拟间接光或环境光照。ambient = ambientStrength * lightColor，通常是常量，直接与材质颜色相乘作为基础亮度。

2. 漫反射的实现

光线被粗糙表面按余弦定律散射，亮度与光线入射角余弦成正比。lightDir = normalize(lightPos - FragPos)；diff = max(dot(norm, lightDir), 0.0)； diffuse = diff * lightColor 当表面朝向光源（法线与光向量夹角小）时，diff 接近 1，越亮；背光面为 0。

3. 镜面反射的实现

光在光滑表面形成高光，依赖观察方向与反射方向的接近程度。reflectDir = reflect(-lightDir, norm) ,reflect 函数以入射向量 I 关于 N （法线）求反射向量 spec = pow(max(dot(viewDir, reflectDir), 0.0), shininess) specular = specularStrength * spec * lightColor 其中 shininess 越大 -> 高光越尖锐（小而亮的光斑）；越小 -> 光斑更模糊。specularStrength 相当于镜面的平滑程度，用来调节高光强度。因为高光是由光源和表面反射特性决定，不是被物体颜色吸收，所以镜面项通常不与物体纹理颜色相乘

4. 合成颜色（shader 中常见）

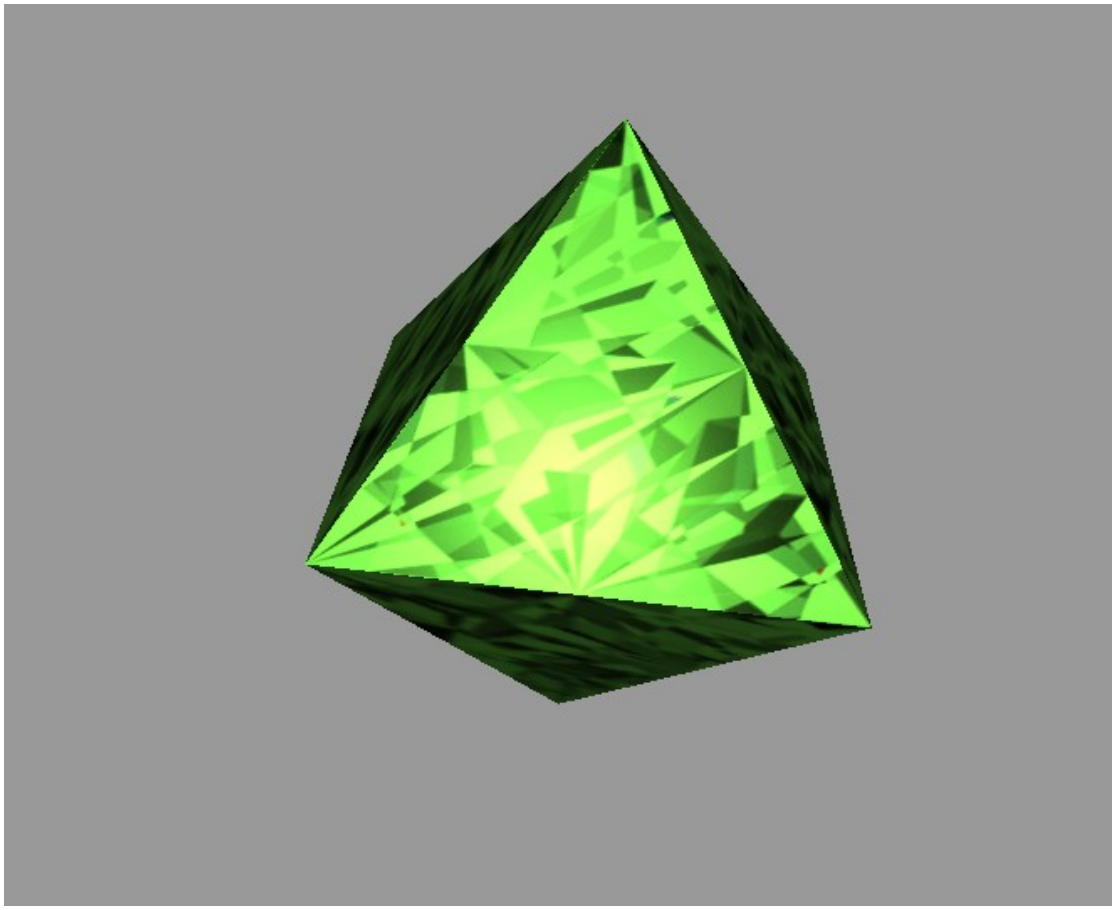
color = (ambient + diffuse) * objColor + specular

环境与漫反射先与材质颜色相乘，再加上不与材质颜色相乘

的镜面高光。

五、结论

这里换了个模型改了个颜色。



六、参考文献

七、附录：源代码

```
#version 460 core

out vec4 FragColor;

in vec2 TexCoords;
```

```
in vec3 Normal;

in vec3 FragPos;


// 在 main.cpp 中:

// shader_cube.setInt("texture1", 0);

// shader_cube.setVec3("lightPos", 1.5f, 1.0f, 1.2f);

// shader_cube.setVec3("viewPos", camera.position);

// shader_cube.setVec3("lightColor", 1.0f, 1.0f, 1.0f);

// shader_cube.setMat4("model", glm::mat4(1.0f));

// shader_cube.setMat4("view", camera.getViewMatrix());

// shader_cube.setMat4("projection", ... );

// 纹理在 main.cpp 中用 loadTexture 加载为 texture，并用
glActiveTexture(GL_TEXTURE0)

// 绑定到 GL_TEXTURE0，其后 shader_cube.setInt("texture1", 0) 将采样
器绑定到单元 0

uniform sampler2D texture1; // 纹理采样器

uniform vec3 lightColor; // 光的颜色

uniform vec3 lightPos; // 光源位置（世界坐标）

uniform vec3 viewPos; // 相机/观察者位置（世界坐标）

void main()

{

// 镜面/漫反/环境光参数
```

```
float specularStrength = 0.9; // 控制高光强度
float ambientStrength = 0.1; // 环境光强度

// 方向向量（都归一化）
// 视线方向：从片元指向观察者（相机）
vec3 viewDir = normalize(viewPos - FragPos);

// 法线向量：从顶点属性获取，再归一化
vec3 norm = normalize(Normal);

// 光照方向：从片元指向光源
vec3 lightDir = normalize(lightPos - FragPos);

// 反射向量：反射入射光线（用 -lightDir 表示从光源到片元的入射向量）
// reflect(I, N) 计算向量 I 关于法线 N 的反射向量
vec3 reflectDir = reflect(-lightDir, norm);

// 镜面高光（Phong）
// dot(viewDir, reflectDir) 越接近 1，说明视线更接近反射方向 -> 高光越强
// pow(..., 32) 控制高光聚集程度（高 -> 更小更亮的高光斑）
float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
vec3 specular = specularStrength * spec * lightColor; // 镜面分量

// 漫反射（Lambert）
// 用法线和光向量的点乘来计算漫反射强度，取 0 到 1
```

```
float diff = max(dot(norm, lightDir), 0.0);

vec3 diffuse = diff * lightColor; // 漫反射分量（与表面材料的颜色相乘）


// 环境光（常量项）

vec3 ambient = ambientStrength * lightColor;


// 从纹理中采样得到表面颜色（rgb）

vec3 objColor = texture(texture1, TexCoords).rgb;


// ambient 和 diffuse 与物体颜色相乘（表示被物体吸收/反射的光）

// 而 specular 为高光（通常由光源决定，与物体颜色无关）

vec3 result = (ambient + diffuse) * objColor + specular;


FragColor = vec4(result, 1.0);

}
```