实验二：光照模型

姓 名： 侯腾跃

学 号： 2022217477

班 级： 计科22-4班

实验地点： 计算机中心楼三机房

实验时间： 2025.4.9

1. 实验目的和要求

理解和掌握简单光照模型的基本原理和方法；并编程实现两种常用 的明暗处理方法： Gouraud 明暗处理方法 与 Phong 明暗处理方法

说明：本平台目前仅考虑环境光(Ambient Light)、漫反射光(Diffuse light)，暂不考虑镜面反射光(Specular Light ),而且不考虑光强衰减；

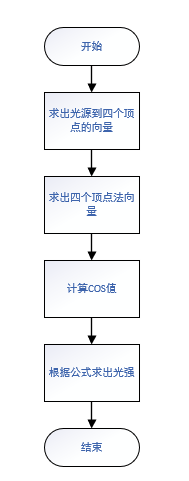
1. 实验环境和工具

开发环境：Visual C++ 6.0

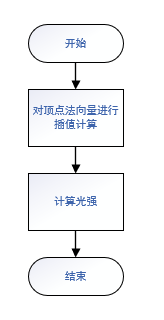
实验平台：Illumination（自制平台）

1. 实验结果
   1. 程序流程图

Gouraud算法：



Phong算法：



* 1. 程序代码

**Gouraud算法：**

double CIlluminationView::Gouraud (int Pt[3],

int Vertices[4][3],

double Normals[4][3],

int lightPos[3],

double Ip,

double Kd)

{

double I = 0;

double uv[4];

if (BilinearInterpolation(Vertices, Pt, uv)) {

double strong[4];//光强

double cos1[4];//计算光源每一点相对于p[i]的角度

double light\_dir[4][3];//光源到p点的向量

double strlen[4];//mo

double len[4];

double Normals2[4][3];

int i, j;

for (i = 0; i < 4; i++) {

for (j = 0; j < 3; j++) {

light\_dir[i][j] = lightPos[j] - (double)Vertices[i][j];

}

}

for (i = 0; i < 4; i++){

strlen[i] = sqrt(light\_dir[i][0] \* light\_dir[i][0] + light\_dir[i][1] \* light\_dir[i][1] + light\_dir[i][2] \* light\_dir[i][2]);

}

for (i = 0; i < 4; i++){

for (j = 0; j < 3; j++){

light\_dir[i][j] = light\_dir[i][j] / strlen[i];

}

}

for (i = 0; i < 4; i++){

len[i] = sqrt(Normals[i][0] \* Normals[i][0] + Normals[i][1] \* Normals[i][1] + Normals[i][2] \* Normals[i][2]);

}

for (i = 0; i < 4; i++){

for (j = 0; j < 3; j++){

Normals2[i][j] = Normals[i][j] / len[i];

}

}

for (i = 0; i < 4; i++){

cos1[i] = light\_dir[i][0] \* Normals2[i][0] + light\_dir[i][1] \* Normals2[i][1] + light\_dir[i][2] \* Normals2[i][2];

}

for (i = 0; i < 4; i++){

strong[i] = Ip \* Kd \* cos1[i];

}

for (i = 0; i < 4; i++){

I = I + strong[i] \* uv[i];

}

}

if (I < 0) {

I = 0;

}

return I;

}

**Phong算法：**

double CIlluminationView::Phong(int Pt[3],

int Vertices[4][3],

double Normals[4][3],

int lightPos[3],

double Ip,

double Kd){

double I = 0; //itensity;

double light[3] = {lightPos[0] - Pt[0], lightPos[1] - Pt[1], lightPos[2] - Pt[2]};

double light\_norm = sqrt(light[0] \* light[0] + light[1] \* light[1] + light[2] \* light[2]);

double light\_dir[3] = {light[0] / light\_norm, light[1] / light\_norm, light[2] / light\_norm};

double normal[3];

double uv[4];

if (!BilinearInterpolation(Vertices, Pt, uv)) {

return 0.0;

}

for (int i = 0; i < 4; i++) {

normal[0] += Normals[i][0] \* uv[i];

normal[1] += Normals[i][1] \* uv[i];

normal[2] += Normals[i][2] \* uv[i];

}

double normal\_norm = sqrt(normal[0] \* normal[0] + normal[1] \* normal[1] + normal[2] \* normal[2]);

double normal\_dir[3] = {normal[0] / normal\_norm, normal[1] / normal\_norm, normal[2] / normal\_norm};

I = Kd \* Ip \* (normal\_dir[0] \* light\_dir[0] + normal\_dir[1] \* light\_dir[1] + normal\_dir[2] \* light\_dir[2]);

if (I < 0.0) {

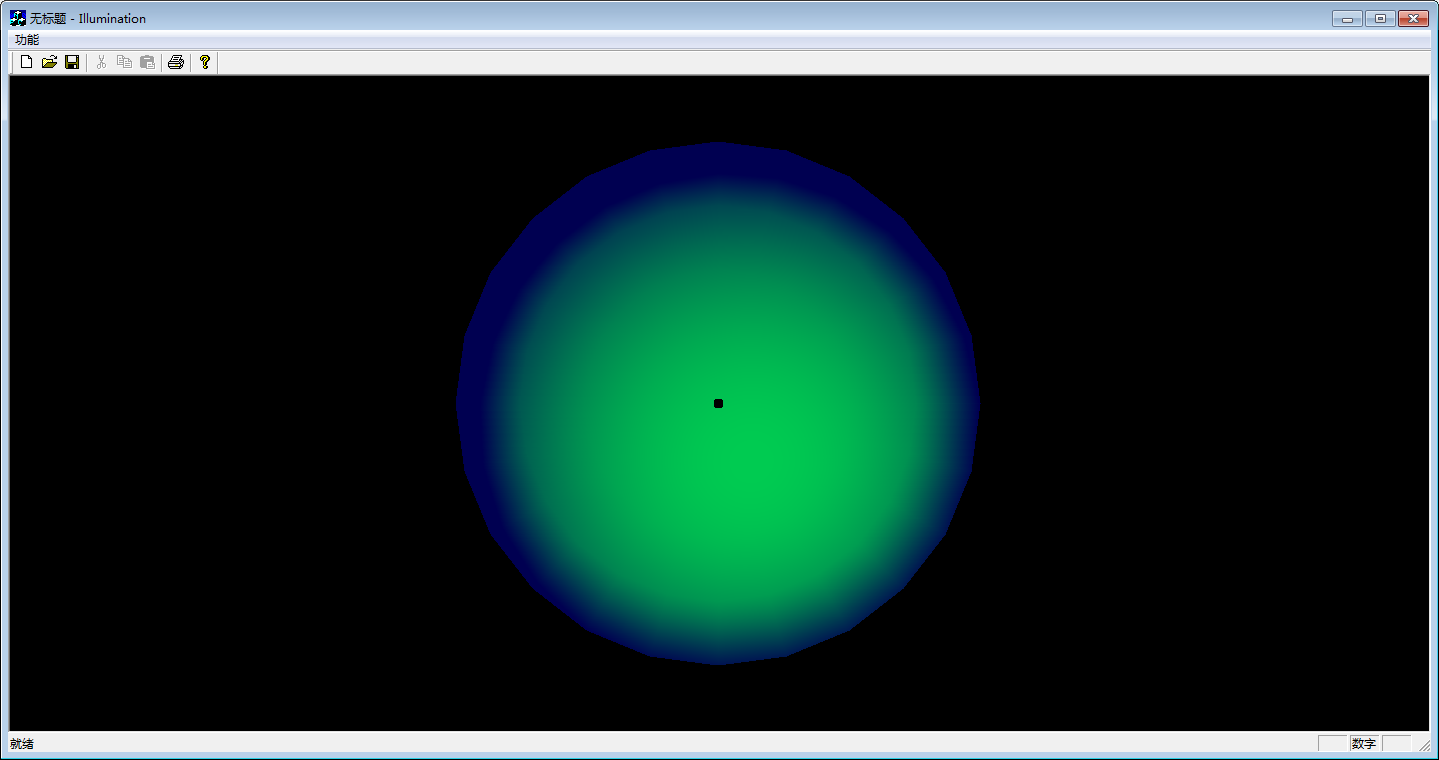
I = 0.0;

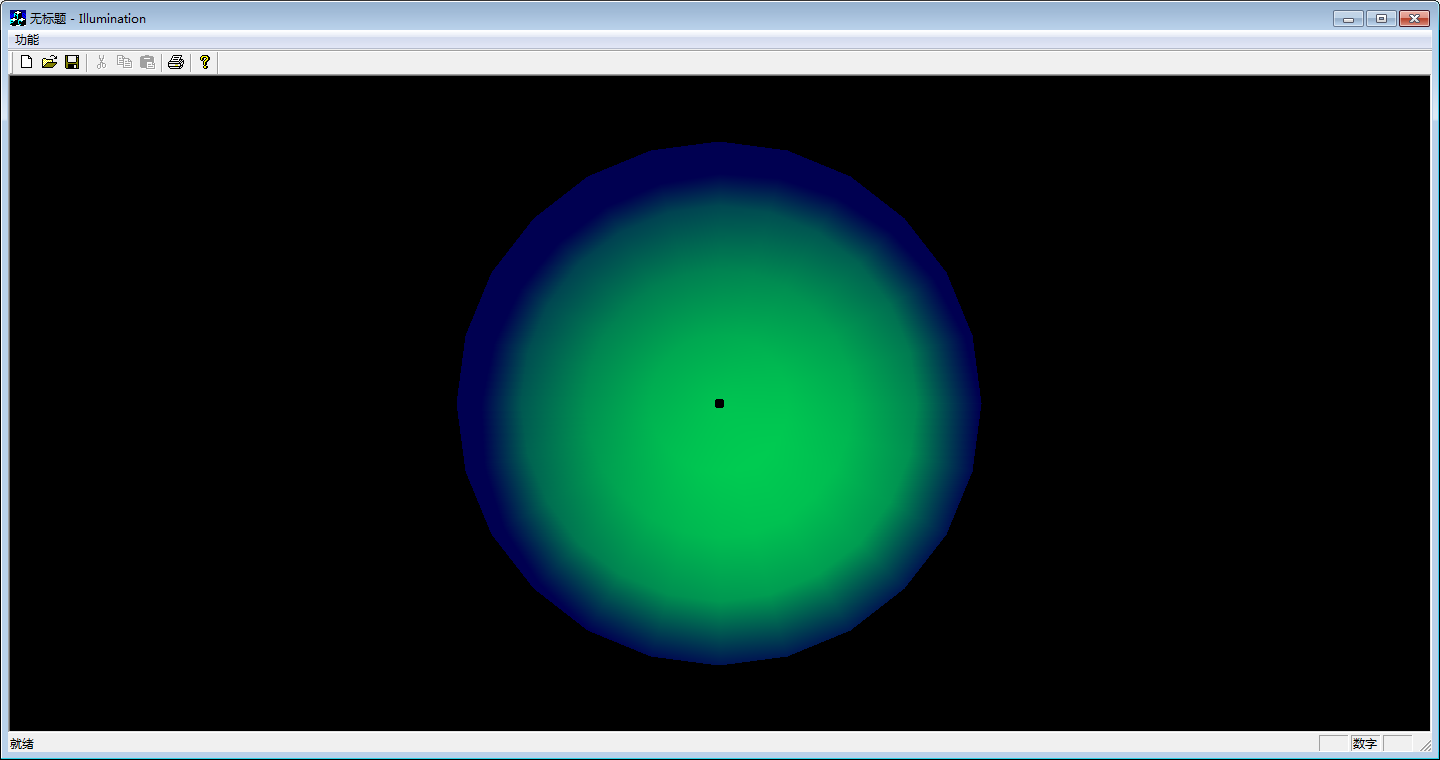
}

return I;

}

* 1. 运行结果





* 1. 运行结果分析

实验结果正确，符合预期。

1. 思考题（可选）
   1. 函数: double CIlluminationView::DensityOnPoint( int Pt[3], double N[3], int lightPos[3], double Ip, double Kd)计算光亮度值过程中对角度 的余弦函数值求解, 还有没有其它求解方法？

**向量点乘法：**

如果已知两个单位向量 A 和 B，它们之间的夹角余弦值可以直接通过点乘公式计算：cosθ=A⋅B

在光照计算中，通常需要计算光源方向向量 L 和表面法线向量 N 的夹角余弦值。如果这两个向量已经归一化，可以直接使用点乘。

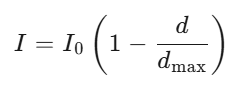
**查找表法：**预先计算并存储一系列角度的余弦值，然后在需要时通过查找表获取结果。这种方法可以避免实时计算余弦值，从而提高效率。

**近似公式：**使用多项式近似或其他数学近似方法来计算余弦值。例如，泰勒级数展开可以在一定精度范围内近似余弦函数。

4.2 如何实现光强衰减效果？如何实现镜面反射光效果？

**光强衰减效果**

1.线性衰减

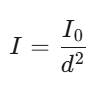
光强随距离线性衰减，公式为：

其中，I0是初始光强，d是光源到物体的距离，d max是衰减范围。

1. 指数衰减

光强随距离指数衰减，公式为： 

1. 二次衰减

根据物理定律，光强随距离的平方衰减，公式为：

**镜面反射光效果**

1.Phong模型：

镜面反射光强计算公式为：

​

其中，Il是光源强度，R 是反射向量，V 是观察方向向量，ns是镜面高光指数。

2.Blinn-Phong模型：

使用半角向量 H 替代反射向量 R，计算公式为：



其中，H 是光源方向向量和观察方向向量的半角向量。

3.GLSL内置函数：

在图形着色器中，可以使用GLSL的reflect函数直接计算反射向量，然后结合点乘和高光指数计算镜面反射光。

1. 实验心得

Gouraud明暗处理是一种基于顶点的光照模型。它的核心思想是在多边形的每个顶点处计算光照强度，然后通过插值在多边形内部生成平滑的光照效果。这种方法的优点是计算效率高，因为光照计算仅在顶点处进行，而插值过程相对简单。然而，Gouraud明暗处理的缺点也很明显：它无法很好地处理高光效果，因为高光通常出现在多边形内部，而不是顶点上。此外，当多边形较大时，插值可能会导致光照效果的不自然过渡。

我通过计算每个顶点的光照强度，然后使用线性插值在多边形内部生成光照值。虽然实现过程相对简单，但在观察结果时，我发现高光部分不够明显，且在多边形边界处的光照过渡不够平滑。这让我意识到Gouraud明暗处理在处理复杂光照场景时的局限性。

在实现Phong明暗处理时，我首先计算了每个顶点的法线向量，然后通过插值在多边形内部的每个像素上得到法线向量的近似值。接着，我在每个像素上应用Phong光照模型，计算最终的光照强度。虽然实现过程比Gouraud方法复杂，但结果让我非常满意。Phong明暗处理能够生成更平滑、更逼真的光照效果，尤其是在高光部分的表现上。

通过这次实验，我深刻体会到了不同光照模型在实现细节和视觉效果上的差异。Gouraud明暗处理虽然计算效率高，但在处理高光和复杂光照场景时存在局限性；而Phong明暗处理虽然能够生成更逼真的光照效果，但计算成本较高。这让我认识到，在实际应用中，选择合适的光照模型需要根据具体需求和性能要求进行权衡

**（在学生的实验报告电子版的最后）**

**表. 实验成绩评定表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 评价内容 | | 权重 | 得分 |
| **验收** | 实验原理是否理解；程序能否运行；实验结果是否正确；任务是否全部完成。 | | 0.5 |  |
| **实验报告** | 1 | 报告格式是否规范，语言使用是否规范，行文是否流畅，是否图文并茂； | 0.2 |  |
| 2 | 实验原理、实验步骤描述是否正确、详实；  程序流程图是否规范，代码实现是否正确；  实验数据记录是否完整，实验结果是否正确；  实验结果的分析、对比是否充分； | 0.2 |  |
| 3 | 实验体会是否正确，是否提出了自己独到见解。 | 0.1 |  |
| 合计 |  | | | |
| 指导教师（签章）： 年 月 日 | | | | |