实验二：光照模型

姓 名： 陈嘉乐

学 号： 2021218152

班 级： 计科21-3班

实验地点： 计算机楼

实验时间： 2024.3.26

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明暗处理方法和Phong明暗处理方法。

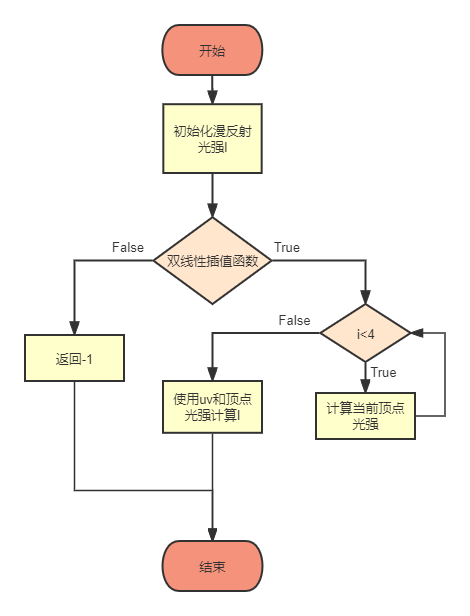
说明:本平台目前仅考虑环境光(Ambient Light)、漫反射光(Diffuse

light)，暂不考虑镜面反射光(Specular Light),而且不考虑光强衰

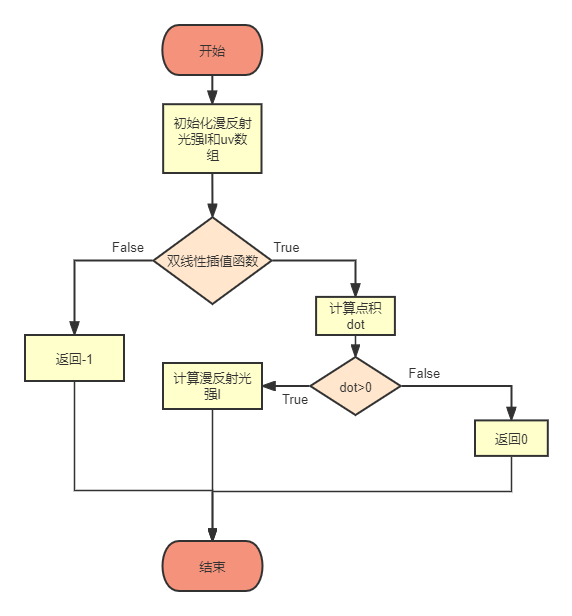
减;

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)
2. 实验结果
   1. 程序流程图

Gouraud函数程序流程图：



Phong函数程序流程图：



* 1. 程序代码

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

int Vertices[4][3],

double Normals[4][3],

int lightPos[3],

double Ip,

double Kd)

{

double I = 0; // 初始化漫反射光强为0

double uv[4]; // 用于存储双线性插值的系数

// 调用双线性插值函数判断点Pt是否在四边形内部，并计算插值系数

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

return -1; // 如果插值失败（即点不在四边形内部），则返回-1

}

double intensityAtVertices[4]; // 存储四个顶点处的光照强度

// 遍历四边形的每个顶点

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

// 计算光源到当前顶点的方向向量

double lightDir[3] = {

lightPos[0] - Vertices[i][0],

lightPos[1] - Vertices[i][1],

lightPos[2] - Vertices[i][2]

};

// 计算方向向量的模（长度）

double norm = sqrt(lightDir[0] \* lightDir[0] +

lightDir[1] \* lightDir[1] +

lightDir[2] \* lightDir[2]);

// 归一化方向向量

lightDir[0] /= norm;

lightDir[1] /= norm;

lightDir[2] /= norm;

// 计算方向向量与顶点法向量的点积

double dot = lightDir[0] \* Normals[i][0] +

lightDir[1] \* Normals[i][1] +

lightDir[2] \* Normals[i][2];

// 确保点积非负

dot = max(dot, 0.0);

// 计算顶点处的光照强度

intensityAtVertices[i] = Ip \* Kd \* dot;

}

// 使用双线性插值系数和顶点光照强度计算点Pt处的光照强度

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

I += intensityAtVertices[j] \* uv[j];

}

// 返回点Pt处的漫反射光强

return I;

}

double CIlluminationView::Phong(int Pt[3], // 给定点的坐标

int Vertices[4][3], // 四边形的四个顶点坐标

double Normals[4][3], // 四个顶点的法向量

int lightPos[3], // 光源的位置

double Ip, // 漫反射光源的强度

double Kd) // 漫反射系数

{

double I = 0; // 初始化漫反射光强为0

double uv[4]; // 用于存储双线性插值得到的系数

// 调用双线性插值函数，如果失败则返回-1

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

return -1;

}

// 初始化插值法向量为0向量

double normalAtPt[3] = {0, 0, 0};

// 根据双线性插值得到的系数和四个顶点的法向量，计算点Pt处的插值法向量

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

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

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

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

}

// 归一化插值法向量

double norm = sqrt(normalAtPt[0] \* normalAtPt[0] +

normalAtPt[1] \* normalAtPt[1] +

normalAtPt[2] \* normalAtPt[2]);

if (norm != 0) { // 避免除以0的情况

normalAtPt[0] /= norm;

normalAtPt[1] /= norm;

normalAtPt[2] /= norm;

}

// 计算光源到点Pt的方向向量

double lightDir[3] = {

lightPos[0] - Pt[0],

lightPos[1] - Pt[1],

lightPos[2] - Pt[2]

};

// 归一化光源方向向量

norm = sqrt(lightDir[0] \* lightDir[0] +

lightDir[1] \* lightDir[1] +

lightDir[2] \* lightDir[2]);

if (norm != 0) { // 避免除以0的情况

lightDir[0] /= norm;

lightDir[1] /= norm;

lightDir[2] /= norm;

}

// 计算光源方向与法向量的点积（即夹角余弦值）

double dot = lightDir[0] \* normalAtPt[0] +

lightDir[1] \* normalAtPt[1] +

lightDir[2] \* normalAtPt[2];

// 如果点积大于0（即光源方向与法向量之间的夹角小于90度），则计算漫反射光强

if (dot > 0) {

I = Ip \* Kd \* dot;

}

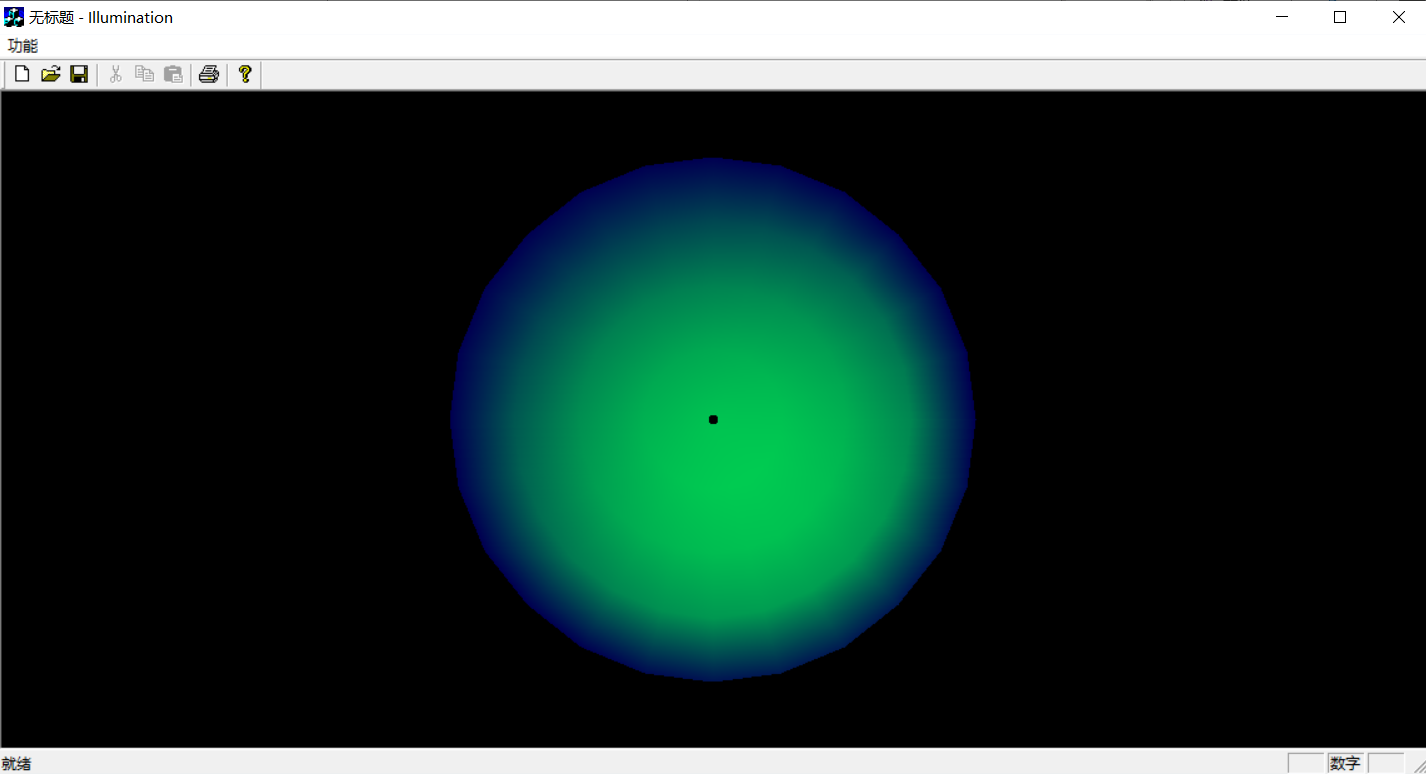
// 返回漫反射光强

return I;

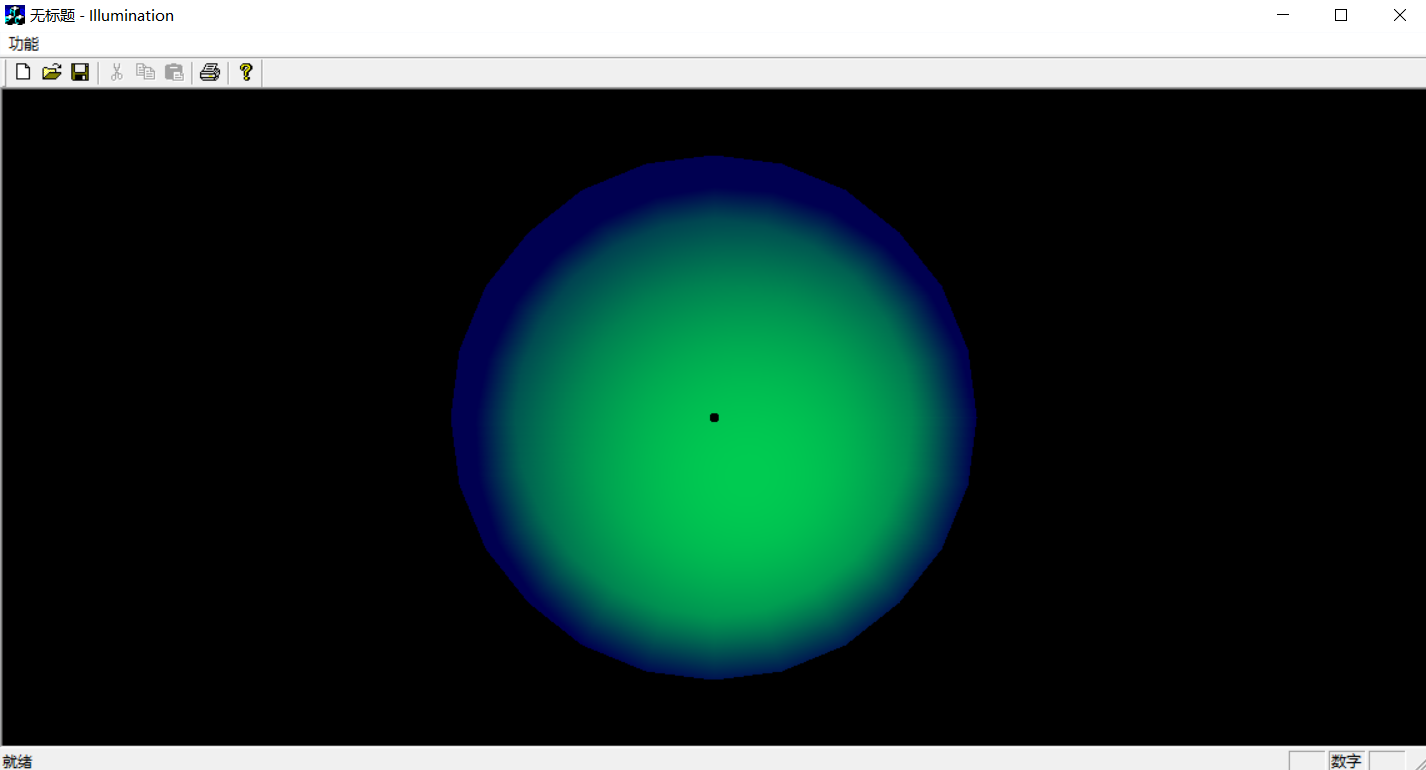
}

* 1. 运行结果

Gouraud明暗处理：



Phong明暗处理：



* 1. 运行结果分析

Gouraud 算法在顶点处计算光照，通过插值得到像素光照值，适用于平滑过渡的表面，但处理细节和曲面丰富的对象时可能出现边缘和光照不连续。而 Phong 算法在每个像素处计算光照，考虑表面法线和视线向量，能呈现更平滑、自然和真实的光照效果。

实验与预期相同，符合实验结果。

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？

有。在原函数中，double dist = (L[0]-N[0])\*(L[0]-N[0]) + (L[1]-N[1])\*(L[1]-N[1]) + (L[2]-N[2])\*(L[2]-N[2]);dist = sqrt(dist);它是通过计算向量L到向量N在欧几里得空间中的距离。但这实际上计算的是两个单位向量之间的弦长，用于后续的角度计算。double theta = 2.0 \* asin(dist/2.0);这里使用了球面三角形的性质，其中 dist 是弦长，theta 是对应的中心角。

正确的计算两个单位向量之间夹角余弦值的方法是使用点积。具体代码如下：

// 将光源位置转换为双精度浮点数

double lightPosDouble[3];

lightPosDouble[0] = static\_cast<double>(lightPos[0]);

lightPosDouble[1] = static\_cast<double>(lightPos[1]);

lightPosDouble[2] = static\_cast<double>(lightPos[2]);

// 计算从光源到点的向量

double L[3];

L[0] = lightPosDouble[0] - Pt[0];

L[1] = lightPosDouble[1] - Pt[1];

L[2] = lightPosDouble[2] - Pt[2];

// 单位化向量L

double lenL = sqrt(L[0]\*L[0] + L[1]\*L[1] + L[2]\*L[2]);

if (lenL > 0.0) {

L[0] /= lenL;

L[1] /= lenL;

L[2] /= lenL;

} else {

// 如果L是零向量，则返回0光强

return 0.0;

}

// 假设N已经是单位向量，直接计算点积

double dotProduct = N[0]\*L[0] + N[1]\*L[1] + N[2]\*L[2];

// 点积结果即为两个单位向量之间夹角的余弦值

double cosTheta = dotProduct;

// 确保余弦值在[-1, 1]范围内（虽然理论上点积结果应该已经在这个范围内）

cosTheta = std::max(-1.0, std::min(1.0, cosTheta));

// 计算漫反射光强

double Id = Ip \* Kd \* std::max(0.0, cosTheta); // 确保光强非负

return Id;

4.2 如何实现光强衰减效果？如何实现镜面反射光效果？

关于如何实现光强衰弱效果：可以计算给定点到光源的距离，然后利用光强衰减的公式，再重新获得光强I并返回即可。部分代码如下。

//给定点到光源的距离

double dx = Pt[0] - lightPos[0];

double dy = Pt[1] - lightPos[1];

double dz = Pt[2] - lightPos[2];

double dis = sqrt(dx \* dx + dy \* dy + dz \* dz);

double D = 1000 / 1 + dis + dis \* dis;

if(D > 1){

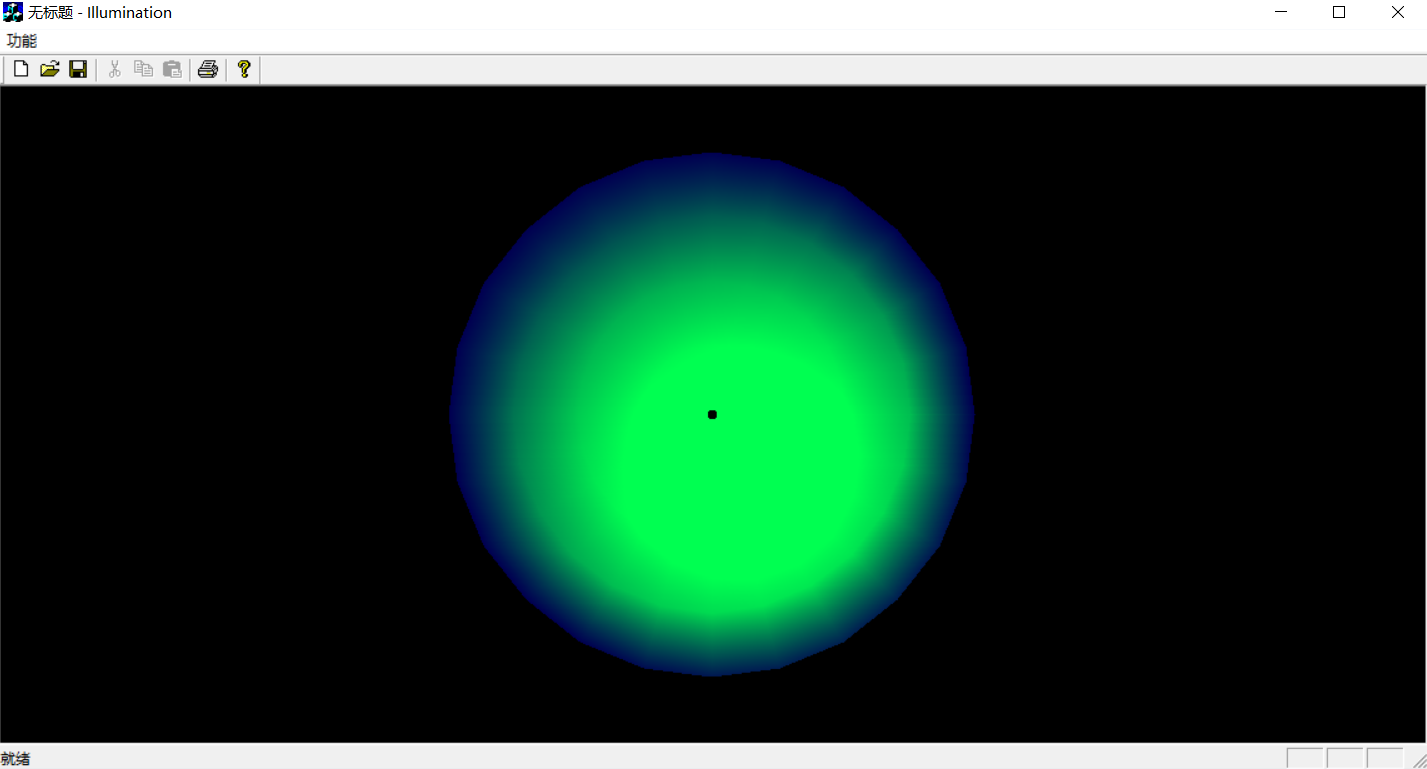
D = 1;

}

I = I \* ((1700 \* 1700 / ((dis \* dis) + 1700)));

//I = I \* dis \* 0.001;

return I;



关于如何实现镜面反射效果：

首先，需要定义包括顶点位置、顶点法向量、光源位置以及光源和材质的属性。接着，我们需要计算表面上的任意点相对于光源的入射光向量和该点指向观察者的视线向量。然后，为了计算镜面反射，我们需要先找到入射光向量和表面法向量的反射。这通常涉及到将入射光向量与法向量做点积，然后将结果乘以2倍的法向量，并从入射光向量中减去这个结果。这样我们就得到了反射向量。接下来，我们需要计算视线向量和反射向量之间的夹角余弦值。这个值将用于确定镜面反射光的强度，因为它反映了观察者的视线与反射光线的对齐程度。最后，我们利用镜面反射系数、高光反射指数以及之前计算的夹角余弦值来计算出镜面反射光的强度。

部分代码如下

// 自定义视线的位置

double eysPosition[3] = {0.0};

for(int m = 0; m < 3; m++)

{

eysPosition[m] = lightPos[m] + (m\*100 + 50);

}

// 计算点Pt到观察者的方向向量, 即V向量

double viewDir[3] = {0};

for (int n = 0; n < 3; n++)

{

viewDir[n] = eysPosition[n] - Pt[n];

}

// 求反射向量R

double refDir[3] = {0.0};

double temp\_up = up(lightPos,pt\_normal, 3);

for(int p = 0; p < 3; p++)

{

refDir[p] = 2 \* (temp\_up \* pt\_normal[p]) - lightPos[p];

}

// 定义镜面反射光属性

double ks = 0.8;

double m\_Shininess = 5;

double Ie = 1;

// 求解镜面反射光

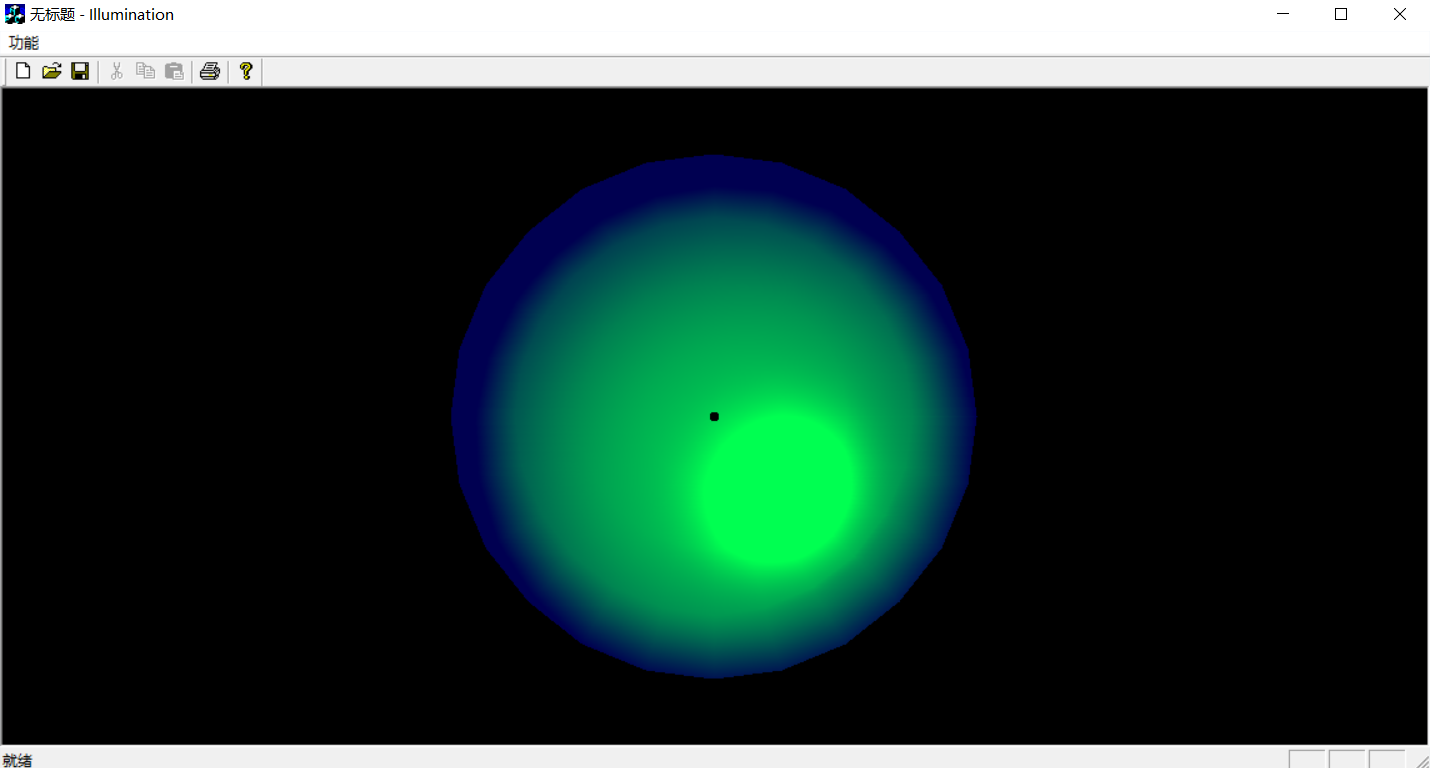
double cosAlpha = up(viewDir, refDir, 3) / (down(viewDir, 3) \* down(refDir, 3));

if(cosAlpha < 0)

cosAlpha = 0.0;

double specular = Ie \* ks \* pow(cosAlpha, m\_Shininess);

I += specular;



1. 实验心得

在这次学习中，我深入研究和实现了Gouraud和Phong两种明暗处理方法，对简单光照模型的基本原理有了深刻的理解。

通过实践，我了解到Gouraud方法在顶点处计算光照强度，通过插值得到颜色，实时性能较好但过渡可能不够平滑。而Phong方法在像素级别计算，虽然计算量大，但颜色过渡更平滑，效果更真实。

这次学习让我深刻体会到光照模型在图形渲染中的重要性，以及不同明暗处理方法的优缺点。通过编程实践，我不仅锻炼了技能，还加深了对图形渲染技术的理解。

**（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 | | | | |