《计算机图形学原理实践》

实验报告

|  |  |  |
| --- | --- | --- |
| 实验名称 | ： | 球面地理划分线框模型消隐 |
|  |  |  |
| 姓 名 | ： |  |
| 学 号 | ： |  |
| 专业班级 | ： |  |
| 实验时间 | ： |  |

西南科技大学计算机科学与技术学院

1. **实验目的**
2. 掌握球面的地理划分法。
3. 掌握球面的数据结构。
4. 掌握球面线框模型的消隐。

**二、实验步骤**

## 1.实验要求：

（1）自定义屏幕三维右手坐标系，原点位于客户区中心，x 轴水平向右为正，y 轴垂直向上为正，z 轴指向屏幕内部。

（2）建立三维用户右手坐标系{O;x,y,z}，原点 O 位于客户区中心，x 轴水平向右为正，y 轴垂直向上为正，z 轴指向读者。

（3）以用户坐标系的原点为正二十面体的体心建立三维几何模型。对每个 表进行递归划分后，将递归点拉到球面上构造球面。

（4）使用三维旋转变换矩阵计算正二十面体线框模型围绕三维坐标系原点变换前后的顶点坐标。

（5）使用双缓冲技术在屏幕坐标系内绘制球面线框模型消隐后的二维透视投影图。

（6）使用键盘方向键旋转球面线框模型。

（7）使用工具条上的“动画”图标按钮播放或停止球面线框模型的旋转动画。

（8）单击鼠标左键增加视径，单击鼠标右键缩短视径。

**2.实验设计**

（1）读入球面的顶点表

使用 CtestView 类添加成员函数 DrawObject(),对每个三角形表面调用递归函数 SubDivide()进行细分

（2）递归函数

CtestView 类的成员函数 SubDivde 计算三角形的三条边的中点并调用Normalize 函数将中点拉到球面上

（3）球化函数

CtestView 类中的 normalize 函数，将每个三角形中点的位置矢量单位化后再乘以球的半径进行球面化处理

（4）绘制三角形

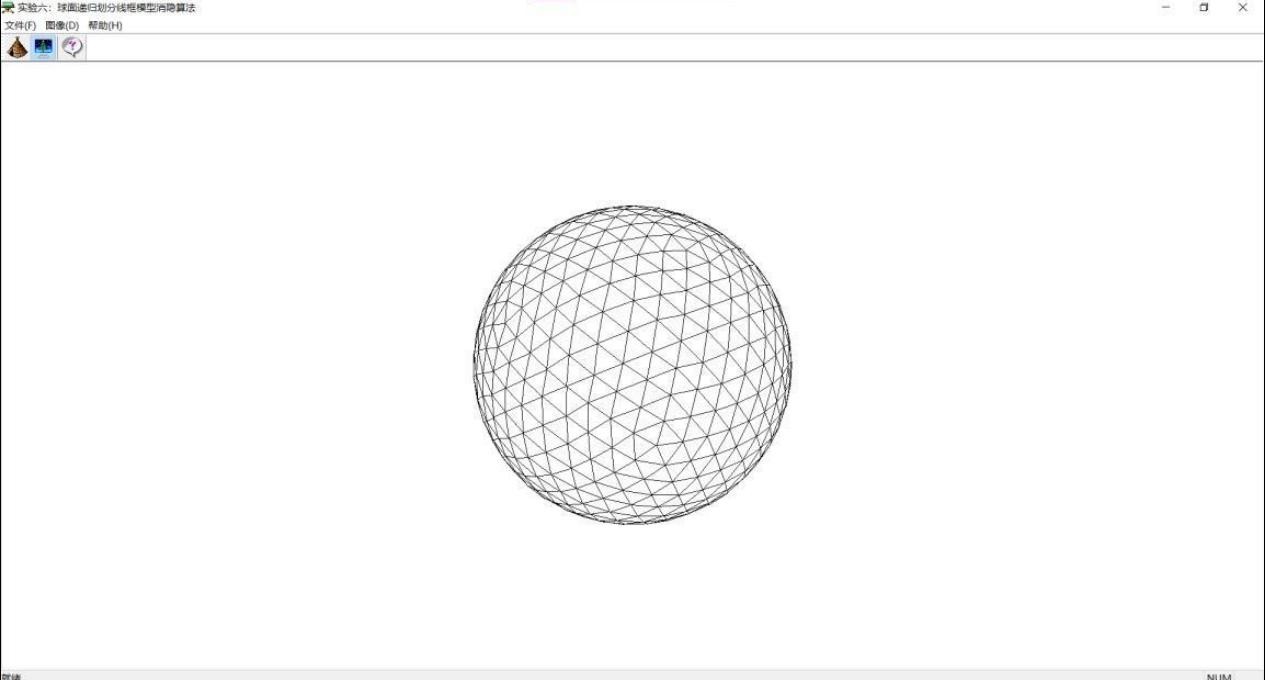
CtestView 类中成员函数 rawTriangle 消隐后使用直线段绘制透视投影后每个三角形小面的线框模型。

**三、实验结果**

1.窗口正中间显示一个球面模型

图示

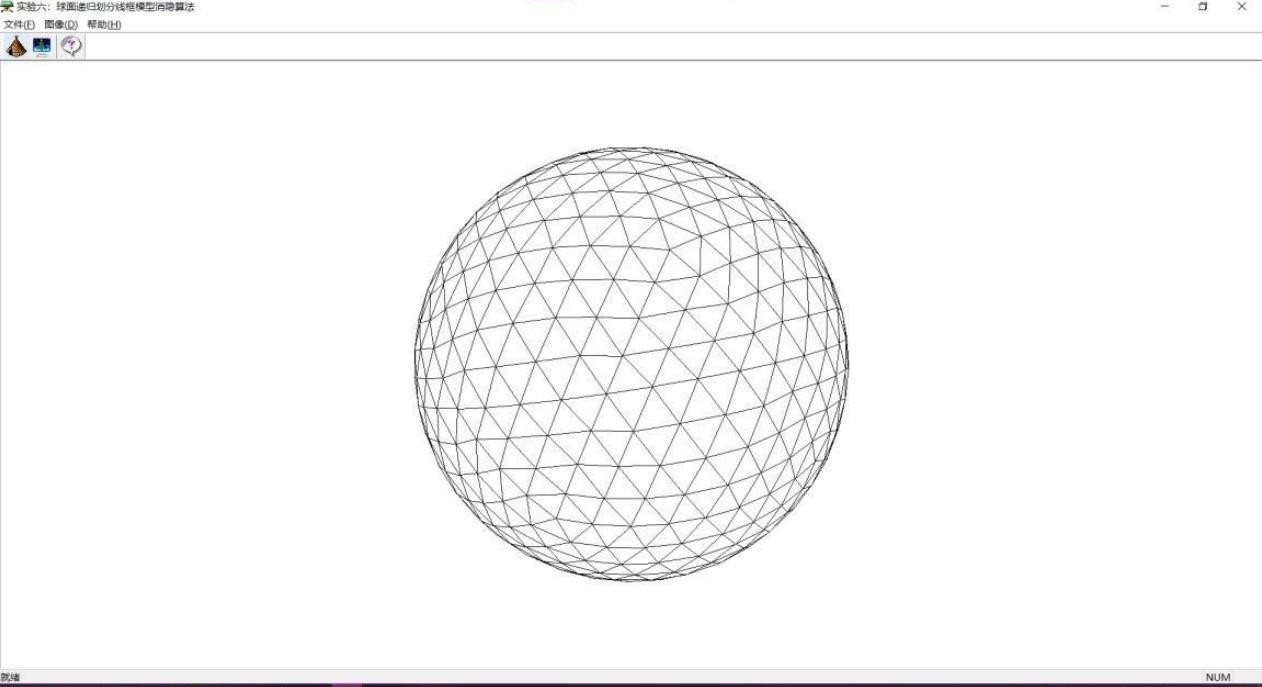
描述已自动生成

2.点击开始，开始旋转，点击暂停后停止旋转，可以看到球体发生了转变。

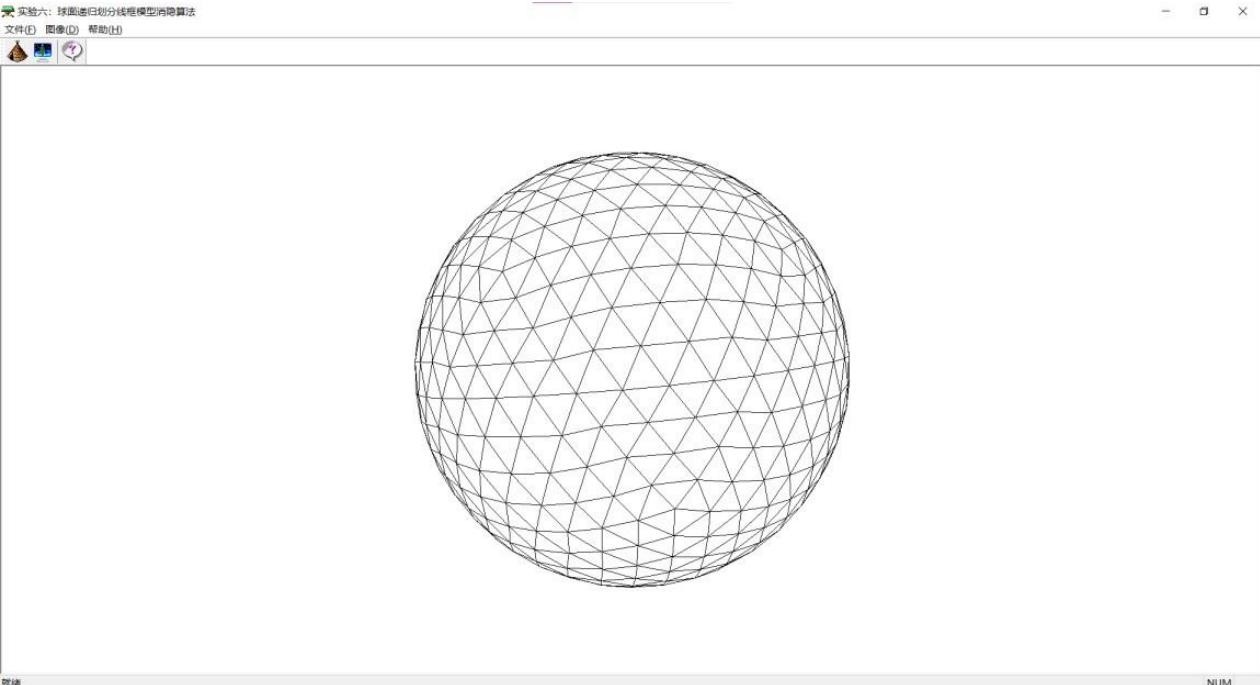
3.点击鼠标左键，球体缩小。

图示

中度可信度描述已自动生成

4.点击鼠标右键，球体放大。

5.按方向键，图像向相应方向旋转



1. **实验总结**

在本次实验中，通过运用学习到的图形学的相关理论知识，掌握了球面的地理划分法以及球面线框模型的消隐方法。通过学习球面的数据结构，和线框模型的消隐，能表示定视点位置的球面坐标。在实现过程中通过创建二维数组读入球面的面表。此次实验碰到了不少难题，所幸在老师的视频讲解以及同学的帮助下顺利完成。

**附录：源代码（关键代码）**

void CTestView::ReadVertex()//点表

{

const double Golden\_Section=(sqrt(5.0)-1.0)/2.0;//黄金分割比例double a=160;//黄金矩形长边的边长

double b=a\*Golden\_Section;//黄金矩形短边的边长Radius=sqrt(a\*a+b\*b);//正二十面体外接球体半径

//顶点的三维坐标(x,y,z) V[0].x=0; V[0].y=a; V[0].z=b;

V[1].x=0; V[1].y=a; V[1].z=-b;

V[2].x=a; V[2].y=b; V[2].z=0;

V[3].x=a; V[3].y=-b; V[3].z=0;

V[4].x=0; V[4].y=-a; V[4].z=-b;

V[5].x=0; V[5].y=-a; V[5].z=b;

V[6].x=b; V[6].y=0; V[6].z=a;

V[7].x=-b; V[7].y=0; V[7].z=a;

V[8].x=b; V[8].y=0; V[8].z=-a;

V[9].x=-b; V[9].y=0; V[9].z=-a;

V[10].x=-a;V[10].y=b; V[10].z=0;

V[11].x=-a;V[11].y=-b;V[11].z=0;

}

void CTestView::SubDivide(CDC \*pDC,CP3 p0, CP3 p1, CP3 p2,int n)//递归函数

{

if (0 == n)

{

DrawTriangle(pDC, p0, p1, p2); return;

}

else

{

CP3 p01, p12, p20; p01 = (p0 + p1) / 2.0; p12 = (p1 + p2) / 2.0; p20 = (p2 + p0) / 2.0;

Normalize(p01); Normalize(p12); Normalize(p20);

SubDivide(pDC, p0, p01, p20, n - 1); SubDivide(pDC, p1, p12, p01, n - 1); SubDivide(pDC, p2, p20, p12, n - 1); SubDivide(pDC, p01, p12, p20, n - 1);

}

void CTestView::Normalize(CP3 &p)

{

if (0 == p.Mag()) return;

p /= p.Mag(); p \*= Radius;

}

void CTestView::DrawTriangle(CDC \*pDC,CP3 p0, CP3 p1, CP3 p2)

{

CLine\* line = new CLine; CP3 point[3];

CVector ViewVector(p0, ViewPoint); ViewVector = ViewVector.Normalize(); CVector V01(p0, p1);

CVector V02(p0, p2);

CVector FNormal = Cross(V01, V02); FNormal.Normalize();

if (Dot(ViewVector, FNormal) >= 0)

{

PerProject(p0); point[0] = ScreenP; PerProject(p1); point[1] = ScreenP; PerProject(p2); point[2] = ScreenP;

line->MoveTo(pDC, point[0].x, point[0].y); line->LineTo(pDC, point[1].x, point[1].y); line->LineTo(pDC, point[2].x, point[2].y); line->LineTo(pDC, point[0].x, point[0].y);

delete line;

}