**案例30-球面递归划分算法**

1. **案例描述：**

本案例利用球面递归划分算法，绘制递归划分球面。

1. **实现步骤：**
2. 添加基础类与添加绘制立方体的CSphere类。
3. 在CSphere类中计算顶点坐标、读入面表，绘制图形，透视变换参数初始化、设置视点位置以及透视变换。
4. 在CTestView中添加消息响应函数，在OnDraw中调用DoubleBuffer函数。
5. **主要算法**

1. CSphere类：

public:

void ReadPoint();//读入点表

void ReadFace();//读入面表

void InitParameter();//参数初始化

void SetViewPoint();//设置视点

CP2i PersPectiveProjection(CP3d P);//透视投影

void Draw(CDC\* pDC);//绘制球面线框

void SubDivide(CDC \*pDC, CP3d, CP3d, CP3d, int);//递归函数

void DrawTriangle(CDC \*pDC, CP3d, CP3d, CP3d);//绘制三角形函数

void Normalize(CP3d &);//矢量扩展函数

public:

CP3d V[12];//点表

CFace F[20];//面表

int nClientWidth;//屏幕客户区宽度

int nClientHeight;//屏幕客户区高度

int nHWidth, nHHeight;//屏幕客户区的半宽和半高

double R, Theta, Phi, d;//视点在用户坐标系中的球坐标

double k[9];

double Alpha, Beta;//x方向旋转α角,y方向旋转β角

CP3d ViewPoint;//视点三维直角坐标

CP2i ScreenCoorP;//透视二维维坐标

double Radius;//球体半径

void CSphere::ReadPoint()//点表

{

const double Golden\_Section = (sqrt(5.0) - 1.0) / 2.0;//黄金分割比例

double a = 200;//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 CSphere::ReadFace()//面表

{

//面的顶点数和面的顶点索引

F[0].SetNum(3); F[0].vI[0] = 0; F[0].vI[1] = 6; F[0].vI[2] = 2;

F[1].SetNum(3); F[1].vI[0] = 2; F[1].vI[1] = 6; F[1].vI[2] = 3;

F[2].SetNum(3); F[2].vI[0] = 3; F[2].vI[1] = 6; F[2].vI[2] = 5;

F[3].SetNum(3); F[3].vI[0] = 5; F[3].vI[1] = 6; F[3].vI[2] = 7;

F[4].SetNum(3); F[4].vI[0] = 0; F[4].vI[1] = 7; F[4].vI[2] = 6;

F[5].SetNum(3); F[5].vI[0] = 2; F[5].vI[1] = 3; F[5].vI[2] = 8;

F[6].SetNum(3); F[6].vI[0] = 1; F[6].vI[1] = 2; F[6].vI[2] = 8;

F[7].SetNum(3); F[7].vI[0] = 0; F[7].vI[1] = 2; F[7].vI[2] = 1;

F[8].SetNum(3); F[8].vI[0] = 0; F[8].vI[1] = 1; F[8].vI[2] = 10;

F[9].SetNum(3); F[9].vI[0] = 1; F[9].vI[1] = 9; F[9].vI[2] = 10;

F[10].SetNum(3); F[10].vI[0] = 1; F[10].vI[1] = 8; F[10].vI[2] = 9;

F[11].SetNum(3); F[11].vI[0] = 3; F[11].vI[1] = 4; F[11].vI[2] = 8;

F[12].SetNum(3); F[12].vI[0] = 3; F[12].vI[1] = 5; F[12].vI[2] = 4;

F[13].SetNum(3); F[13].vI[0] = 4; F[13].vI[1] = 5; F[13].vI[2] = 11;

F[14].SetNum(3); F[14].vI[0] = 7; F[14].vI[1] = 10; F[14].vI[2] = 11;

F[15].SetNum(3); F[15].vI[0] = 0; F[15].vI[1] = 10; F[15].vI[2] = 7;

F[16].SetNum(3); F[16].vI[0] = 4; F[16].vI[1] = 11; F[16].vI[2] = 9;

F[17].SetNum(3); F[17].vI[0] = 4; F[17].vI[1] = 9; F[17].vI[2] = 8;

F[18].SetNum(3); F[18].vI[0] = 5; F[18].vI[1] = 7; F[18].vI[2] = 11;

F[19].SetNum(3); F[19].vI[0] = 9; F[19].vI[1] = 11; F[19].vI[2] = 10;

}

void CSphere::InitParameter()//透视变换参数初始化

{

k[1] = sin(PI \* Theta / 180);

k[2] = sin(PI \* Phi / 180);

k[3] = cos(PI \* Theta / 180);

k[4] = cos(PI \* Phi / 180);

k[5] = k[2] \* k[3];

k[6] = k[2] \* k[1];

k[7] = k[4] \* k[3];

k[8] = k[4] \* k[1];

}

CP2i CSphere::PersPectiveProjection(CP3d P)//透视投影

{

CP3d ViewCoorP;//观察坐标系内的点

ViewCoorP.x = k[3] \* P.x - k[1] \* P.z;//观察坐标系三维坐标

ViewCoorP.y = -k[8] \* P.x + k[2] \* P.y - k[7] \* P.z;

ViewCoorP.z = -k[6] \* P.x - k[4] \* P.y - k[5] \* P.z + R;

ScreenCoorP.x = int(d \* ViewCoorP.x / ViewCoorP.z);//屏幕坐标系二维坐标

ScreenCoorP.y = int(d \* ViewCoorP.y / ViewCoorP.z);

return ScreenCoorP;

}

void CSphere::SetViewPoint()//设置视点

{

ViewPoint.x = R \* k[6];

ViewPoint.y = R \* k[4];

ViewPoint.z = R \* k[5];

}

void CSphere::Draw(CDC\* pDC)//绘制球面线框模型

{

int n = 3;//递归深度

for (int nFace = 0; nFace < 20; nFace++)//面循环

{

CP3d Point[3];//透视投影后面的三维顶点数组

for (int nVertex = 0; nVertex < F[nFace].vN; nVertex++)//顶点循环

Point[nVertex] = V[F[nFace].vI[nVertex]];

SubDivide(pDC, Point[0], Point[1], Point[2], n);

}

}

void CSphere::SubDivide(CDC \*pDC, CP3d p0, CP3d p1, CP3d p2, int n)//递归函数

{

if (0 == n)

{

DrawTriangle(pDC, p0, p1, p2);

return;

}

else

{

CP3d p01, p12, p20;

p01 = (p0 + p1) / 2.0;

p12 = (p1 + p2) / 2.0;

p20 = (p2 + p0) / 2.0;

Normalize(p01);//扩展模长

Normalize(p12);

Normalize(p20);

SubDivide(pDC, p0, p01, p20, n - 1);//递归调用

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

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

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

}

}

void CSphere::Normalize(CP3d &p)

{

if (0 == p.Mag())

return;

p /= p.Mag();//模长单位化

p \*= Radius;//扩展到球面上

}

void CSphere::DrawTriangle(CDC \*pDC, CP3d p0, CP3d p1, CP3d p2)

{

CLine \*line = new CLine;

//先剔除背面然后透视变换

CP2i point[3];

CVector3 ViewVector(p0, ViewPoint);//面的视矢量

ViewVector = ViewVector.Normalize();//单位化视矢量

CVector3 V01(p0, p1);//面的一条边矢量

CVector3 V02(p0, p2);//面的另一条边矢量

CVector3 FNormal = Cross(V01, V02);//面的法矢量

FNormal.Normalize();//单位化法矢量

if (Dot(ViewVector, FNormal) >= 0)//背面剔除

{

PersPectiveProjection(p0);//透视投影p0

point[0] = ScreenCoorP;

PersPectiveProjection(p1);//透视投影p1

point[1] = ScreenCoorP;

PersPectiveProjection(p2); //透视投影p2

point[2] = ScreenCoorP;

line->MoveTo(pDC, nHWidth + point[0].x, nHHeight - point[0].y);

line->LineTo(pDC, nHWidth + point[1].x, nHHeight - point[1].y);

line->LineTo(pDC, nHWidth + point[2].x, nHHeight - point[2].y);

line->LineTo(pDC, nHWidth + point[0].x, nHHeight - point[0].y);

}

delete line;

}

2.CTestView类：

public:

void DoubleBuffer(CDC\* pDC);//双缓冲绘图

void DrawObject(CDC\* pDC);//绘制球面线框

protected:

CTransform3 tran;//变换对象

BOOL bPlay;//动画开关

CSphere sphere;

void CTestView::DoubleBuffer(CDC \*pDC)//双缓冲绘图

{

CRect rect;//定义客户区

GetClientRect(&rect);//获得客户区的大小

sphere.nClientWidth = rect.Width();//屏幕客户区宽度

sphere.nClientHeight = rect.Height();//屏幕客户区高度

sphere.nHWidth = sphere.nClientWidth / 2;//屏幕客户区半宽

sphere.nHHeight = sphere.nClientHeight / 2;//屏幕客户区半高

CDC memDC;

memDC.CreateCompatibleDC(pDC);

CBitmap NewBitmap, \*pOldBitmap;

NewBitmap.CreateCompatibleBitmap(pDC, sphere.nClientWidth, sphere.nClientHeight);

pOldBitmap = memDC.SelectObject(&NewBitmap);

memDC.FillSolidRect(rect, pDC->GetBkColor());

DrawObject(&memDC);

pDC->BitBlt(0, 0, sphere.nClientWidth, sphere.nClientHeight, &memDC, 0, 0, SRCCOPY);

memDC.SelectObject(pOldBitmap);

NewBitmap.DeleteObject();

}

void CTestView::DrawObject(CDC \* pDC)

{

sphere.Draw(pDC);

}

1. **实现效果：**

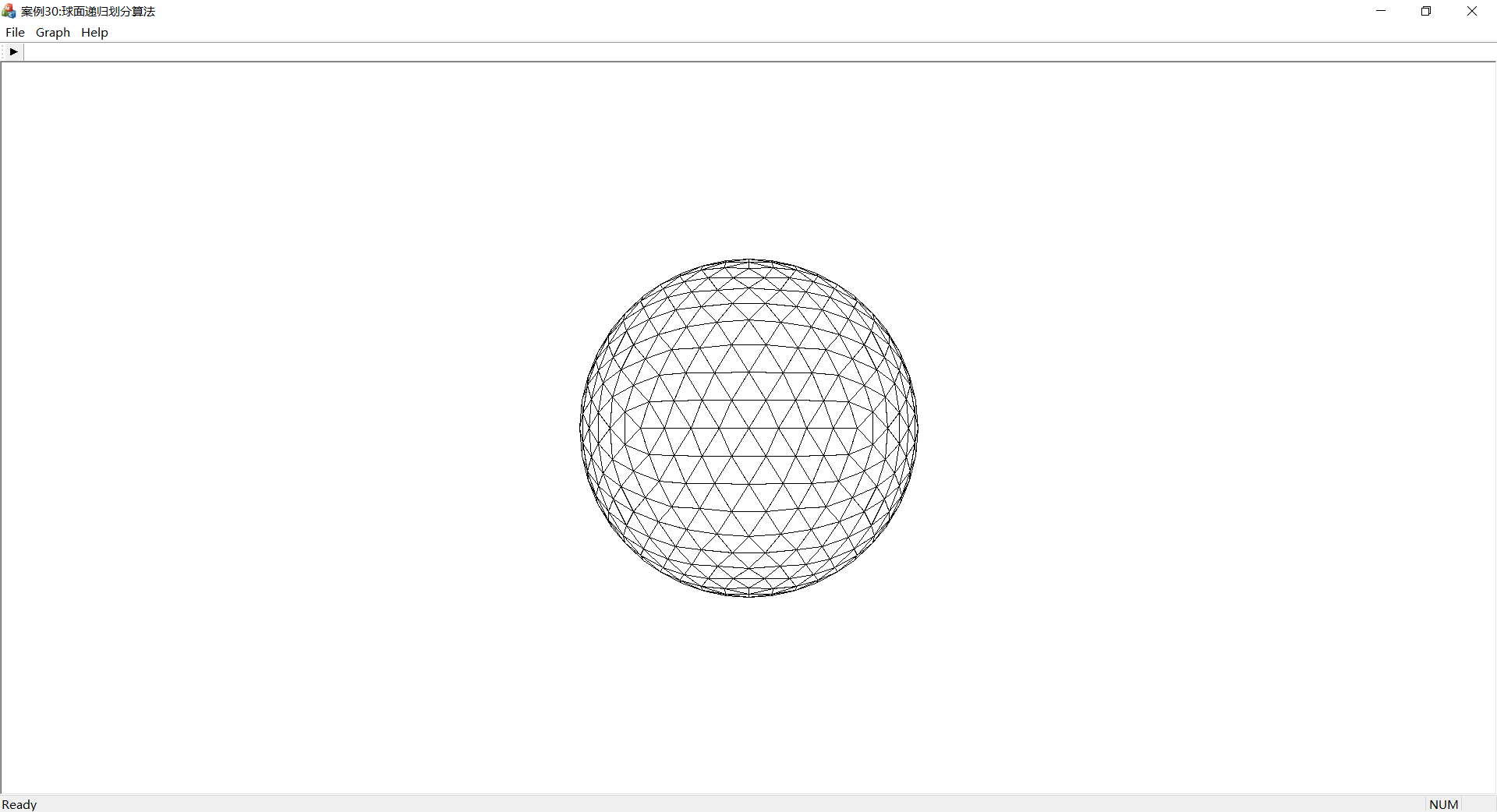


图1球面递归划分算法效果图