**案例32-圆锥面隐线算法**

文档编写：霍波魏

校稿/修订：孔令德

时间2019~2020

联系方式：QQ997796978

**说明：**本套案例由孔令德开发，原版本为Visual C++6.0，配套于孔令德的著作《计算机图形学-基于MFC三维图形开发》一书。孔令德计算机工程研究所的学生霍波魏在学习计算机图形学期间，对本套案例进行了升级并编写了学习文档。现在程序的编写和程序的解释都是基于Windows 10操作系统，使用Microsoft visual studio 2017平台的MFC（中文版）开发。

1. **案例描述：**

本案例利用隐线算法，绘制圆锥面并进行消隐。

假设圆锥周向网格数为n1，纵向网格数为n2，则网格侧面定点数为n1×n2；加上锥顶和地面中心顶点，圆锥网格中共有顶点数n1+n2+2。

圆锥参数：周向网格n1：36；纵向网格n2为：36。

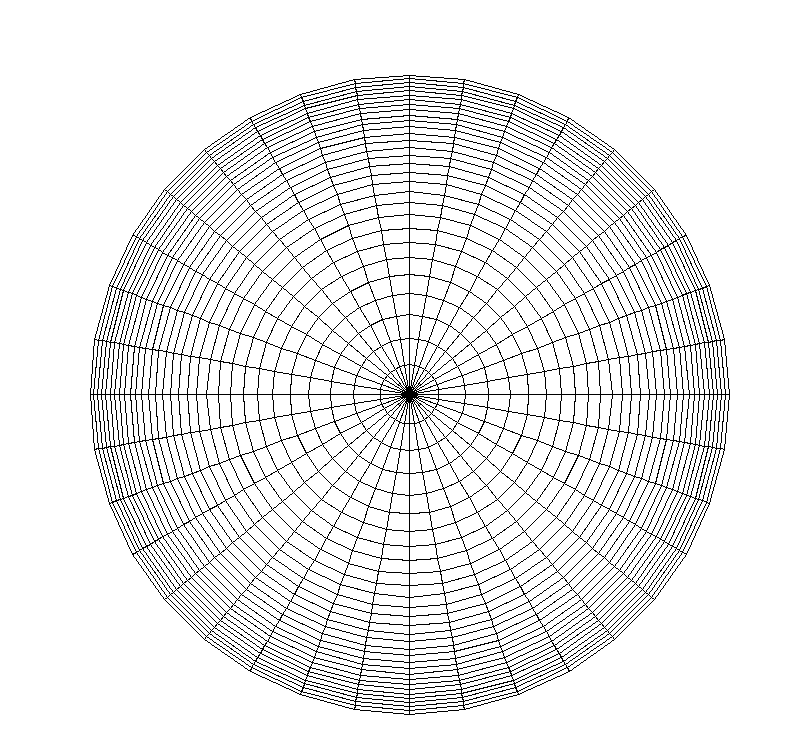
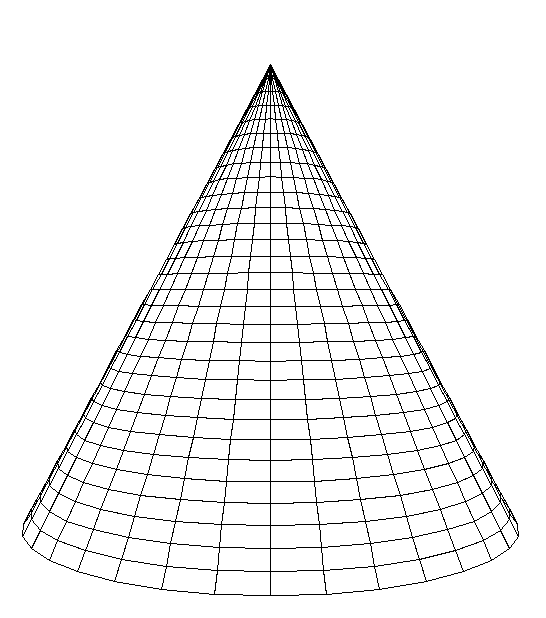


图32-1 主视图 图32-2 俯视图

1. **知识点**

本案例主要讲解的是隐线算法，也称为背面剔除算法，他是解决凸面体消隐问题比较简单的一种方法，背面剔除算法的关键是给出测试其每个表面可见性的判别式，可以根据其外法向量与视向量V（从表面上的一个顶点指向视点）的夹角θ来进行可见性检测。当夹角θ位于之间时，该表面可见，绘制边界线；当夹角θ位于之间时，表面不可见，不绘制边界线。

1. **实现步骤：**
2. 添加基础类与添加绘制圆柱面体的CCone类。
3. 在CCone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. CCone类：

public:

CCylinder(void);

virtual~CCylinder(void);

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 DrawObject(CDC\* pDC);//绘制圆柱线框

public:

double h;//圆柱高度

double r;//圆柱半径

double k[9];

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

double Alpha, Beta;//x方向旋转α角,y方向旋转β角

int N1, N2;//N1为周向网格,N2纵向网格

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

CP3d ViewPoint;//视点球坐标位置

CP2i ScreenCoorP;//屏幕坐标系的二维坐标点

CP3d \*V;//球面顶点一维数组

CFace \*\*F;//小面的二维数组径

void CCone::ReadPoint()//点表

{

r = 160;//圆锥底面半径

h = 300;//圆锥的高

int cTheta = 10;//面片夹角

N1 = 360 / cTheta;//N1周向网格

int cNum = 10;//纵向间距

N2 = ROUND(h / cNum);//N2为纵向网格数

V = new CP3d[N1 \* N2 + 2];

double cTheta1, cNum1;

V[0].x = 0;

V[0].y = 0;

V[0].z = 0;//圆锥底面中心

for (int i = 0; i < N2; i++)//纵向

{

cNum1 = i \* cNum;

for (int j = 0; j < N1; j++)//周向

{

cTheta1 = j \* cTheta \* PI / 180;

double r1 = (N2 - i) \* r / N2;

V[i \* N1 + j + 1].x = r1 \* cos(cTheta1);

V[i \* N1 + j + 1].y = i \* cNum;

V[i \* N1 + j + 1].z = r1 \* sin(cTheta1);

}

}

V[N1 \* N2 + 1].x = 0; V[N1 \* N2 + 1].y = h; V[N1 \* N2 + 1].z = 0;//圆锥顶点

}

void CCone::ReadFace()//面表

{

//设置二维动态数组

F = new CFace \*[N2 + 1];//设置行

for (int n = 0; n < N2 + 1; n++)

F[n] = new CFace[N1];//设置列

for (int j = 0; j < N1; j++)//构造底部三角形面片

{

int tempj = j + 1;

if (tempj == N1) tempj = 0;//面片的首尾连接

int BottomIndex[3];//底部三角形面片索引号数组

BottomIndex[0] = 0;

BottomIndex[1] = j + 1;

BottomIndex[2] = tempj + 1;

F[0][j].SetNum(3);

for (int k = 0; k < F[0][j].vN; k++)//传入面中点的索引

F[0][j].vI[k] = BottomIndex[k];

}

for (int i = 1; i < N2; i++)//构造圆锥侧面的四边形面片

{

for (int j = 0; j < N1; j++)

{

int tempi = i + 1;

int tempj = j + 1;

if (tempj == N1) tempj = 0;

int BodyIndex[4];//圆锥体四边形面片索引号数组

BodyIndex[0] = (i - 1) \* N1 + j + 1;

BodyIndex[1] = (tempi - 1) \* N1 + j + 1;

BodyIndex[2] = (tempi - 1) \* N1 + tempj + 1;

BodyIndex[3] = (i - 1) \* N1 + tempj + 1;

F[i][j].SetNum(4);

for (int k = 0; k < F[i][j].vN; k++)

F[i][j].vI[k] = BodyIndex[k];

}

}

for (int j = 0; j < N1; j++)//构造侧面顶角附近的三角形面片

{

int tempj = j + 1;

if (tempj == N1) tempj = 0;

int TopIndex[3];//顶部三角形面片索引号数组

TopIndex[0] = N1 \* N2 + 1;

TopIndex[1] = (N1 \* (N2 - 1)) + tempj + 1;

TopIndex[2] = (N1 \* (N2 - 1)) + j + 1;

F[N2][j].SetNum(3);

for (int k = 0; k < F[N2][j].vN; k++)

F[N2][j].vI[k] = TopIndex[k];

}

}

void CCone::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];

}

void CCone::SetViewPoint()//设置视点

{

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

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

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

}

CP2i CCylinder::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 CCone::DrawObject(CDC \* pDC)

{

CLine \*line = new CLine;

CP2i Point3[3], t3;//顶面与底面顶点数组

CP2i Point4[4], t4;//侧面顶点数组

for (int i = 0; i < N2 + 2; i++)//N2+2

{

for (int j = 0; j < N1; j++)

{

CVector3 ViewVector(V[F[i][j].vI[0]], ViewPoint);//面的视向量

ViewVector = ViewVector.Normalize();//单位化视向量

F[i][j].SetFaceNormal(V[F[i][j].vI[0]], V[F[i][j].vI[1]], V[F[i][j].vI[2]]);

F[i][j].fNormal.Normalize();//单位化法矢量

if (Dot(ViewVector, F[i][j].fNormal) >= 0)//背面剔除

{

if (3 == F[i][j].vN)//三角形面片

{

for (int m = 0; m < F[i][j].vN; m++)

{

PersPectiveProjection(V[F[i][j].vI[m]]);

Point3[m] = ScreenCoorP;

}

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

{

if (0 == n)

{

line->MoveTo(pDC, nHWidth + Point3[n].x, nHHeight - Point3[n].y);

t3 = Point3[n];

}

else

line->LineTo(pDC, nHWidth + Point3[n].x, nHHeight - Point3[n].y);

}

line->LineTo(pDC, nHWidth + t3.x, nHHeight - t3.y);//闭合多边形

}

else//四边形面片

{

for (int m = 0; m < F[i][j].vN; m++)

{

PersPectiveProjection(V[F[i][j].vI[m]]);

Point4[m] = ScreenCoorP;

}

for (int n = 0; n < 4; n++)

{

if (0 == n)

{

line->MoveTo(pDC, nHWidth + Point4[n].x, nHHeight - Point4[n].y);

t4 = Point4[n];

}

else

{

line->LineTo(pDC, nHWidth + Point4[n].x, nHHeight - Point4[n].y);

}

}

line->LineTo(pDC, nHWidth + t4.x, nHHeight - t4.y);//闭合多边形

}

}

}

}

delete line;

}

2.CTestView类：

public:

void DoubleBuffer(CDC\* pDC);//双缓冲绘图

void Draw(CDC\* pDC);

protected:

CTransform3 tran;//变换对象

BOOL bPlay;//动画开关

CCone cone;

void CTestView::DoubleBuffer(CDC \* pDC)

{

CRect rect;//定义客户区

GetClientRect(&rect);//获得客户区的大小

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

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

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

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

CDC memDC;

memDC.CreateCompatibleDC(pDC);

CBitmap NewBitmap, \*pOldBitmap;

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

pOldBitmap = memDC.SelectObject(&NewBitmap);

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

Draw(&memDC);

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

memDC.SelectObject(pOldBitmap);

NewBitmap.DeleteObject();

}

void CTestView::Draw(CDC \* pDC)

{

cone.DrawObject(pDC);

}

1. **实现效果：**

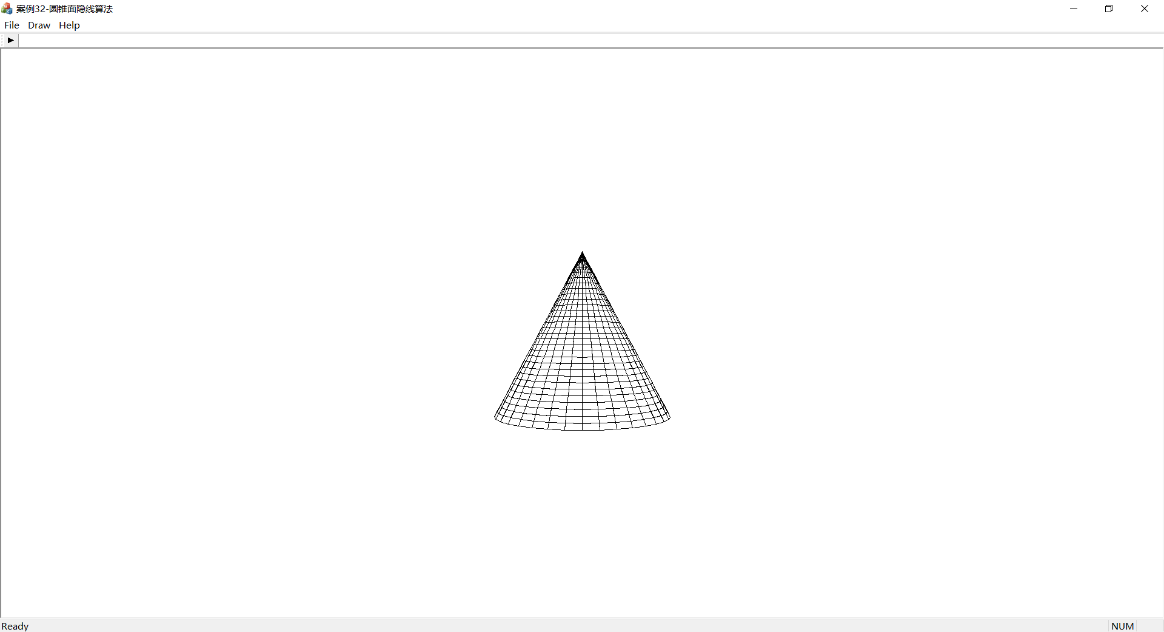


图32-3圆锥面隐线算法效果图