**案例35-钻石透视投影算法**

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：**

本案例利用透视投影，绘制钻石。

钻石主要分为6各部分：桌面外圈（12各点）、腰上刻面中部外圈（12个点）、腰围上圈（24个点）、腰围下圈（24个点）、腰下课面中部外围（12个点）、尖底（1个点）。

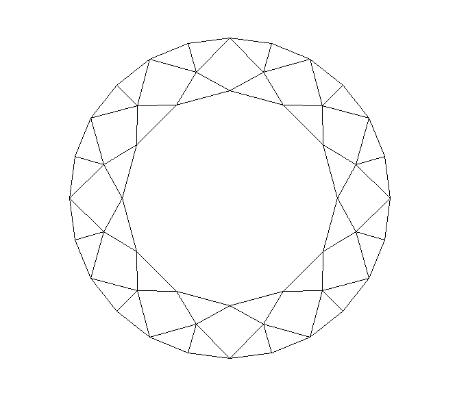
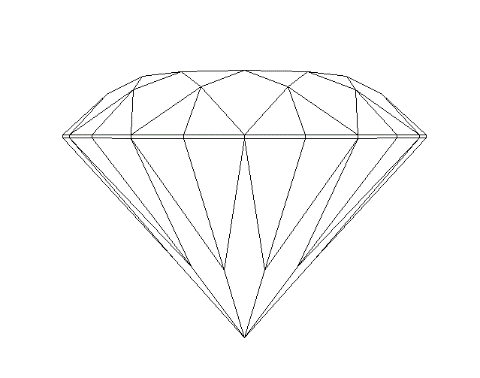


图35-1 主视图 图35-2 俯视图

1. **知识点**

本案例主要讲解的是透视投影，透视投影是从空间一点（称为投影中心或视点）将物体投射到某一观察平面上所得到的图形。透视投影要求三元素：观察者、观察平面、物体。观察平面也称为视平面，通俗地讲就是屏幕。视平面位于观察者与物体之间。视线与视平面的交点就是物体上一点的透视投影。观察者的眼睛位置称为视点，垂直于视平面的视线与视平面的交点称为视心，视点到视心的距离称为视距。视点到物体的距离称为视径。视点是观察坐标系的原点。视心是屏幕坐标系的原点。视距常用d表示，视径常用R表示。

从世界坐标系到屏幕坐标系的透视投影整体变换矩阵为



其中



是观察坐标系到屏幕坐标系的变换，



是世界坐标系到观察坐标系的变换。

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

public:

CDiamond();

virtual~CDiamond();

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

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

void Draw(CDC\* pDC);//绘制钻石线框

void Model(CDC\* pDC, int n, int nface);

public:

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;//R,Theta,Phi视点在用户坐标系的球坐标,d视距

double k[9];//运算常量

CP3d ViewPoint;//视点

CP2d ScreenCoorP;

CP3d P1[12], P2[12], P3[24], P4[24], P5[12], P;

CFace \*\*F;//面表

void CDiamond::ReadPoint()//点表

{

//坐标计算按照比例

double theta1 = 2 \* PI / 12, r1 = 160, orgtheta1 = 0;

double theta2 = 2 \* PI / 12, r2 = 200, orgtheta2 = theta2 / 2;

double theta3 = 2 \* PI / 24, r3 = 260, orgtheta3 = 0;

double theta4 = 2 \* PI / 24, r4 = 260, orgtheta4 = 0;

double theta5 = 2 \* PI / 12, r5 = 104, orgtheta5 = theta5 / 2;

double h1 = 80, h2 = 57, h3 = 0, h4 = -4, h5 = -180, h6 = -300;

for (int i = 0; i < 12; i++)//桌面外圈

{

P1[i].x = r1 \* cos(theta1 \* i + orgtheta1);

P1[i].y = r1 \* sin(theta1 \* i + orgtheta1);

P1[i].z = h1;

}

for (int i = 0; i < 12; i++)//腰上刻面中部外圈

{

P2[i].x = r2 \* cos(theta2 \* i + orgtheta2);

P2[i].y = r2 \* sin(theta2 \* i + orgtheta2);

P2[i].z = h2;

}

for (int i = 0; i < 24; i++)//腰围上圈

{

P3[i].x = r3 \* cos(theta3 \* i + orgtheta3);

P3[i].y = r3 \* sin(theta3 \* i + orgtheta3);

P3[i].z = h3;

}

for (int i = 0; i < 24; i++)//腰围下圈

{

P4[i].x = r4 \* cos(theta4 \* i + orgtheta4);

P4[i].y = r4 \* sin(theta4 \* i + orgtheta4);

P4[i].z = h4;

}

for (int i = 0; i < 12; i++)//腰下刻面中部外圈

{

P5[i].x = r5 \* cos(theta5 \* i + orgtheta5);

P5[i].y = r5 \* sin(theta5 \* i + orgtheta5);

P5[i].z = h5;

}

P.x = 0; P.y = 0; P.z = h6;//尖底

}

void CDiamond::ReadFace()//面表

{

for (int i = 0; i < 7; i++)//动态数组

{

F = new CFace \*[6];

}

F[0] = new CFace[1]; //桌面

F[1] = new CFace[12];//三角刻面

F[2] = new CFace[12];//斜面

F[3] = new CFace[24];//上腰刻面

F[4] = new CFace[24];//腰围

F[5] = new CFace[24];//下腰刻面

F[6] = new CFace[12];//尖底

CP3d p3[3], p4[4], p12[12];

for (int i = 0; i < 12; i++)//桌面

{

p12[i] = P1[i];

}

F[0][0] = CFace(12, p12);

for (int i = 0; i < 12; i++)//三角刻面

{

int j = (i + 1) % 12;

p3[0] = P1[i];

p3[1] = P2[i];

p3[2] = P1[j];

F[1][i] = CFace(3, p3);

}

for (int i = 0; i < 12; i++)//斜面

{

int j = (i + 1) % 12;

int k = (2 \* i + 2) % 24;

p4[0] = P2[i];

p4[1] = P3[k];

p4[2] = P2[j];

p4[3] = P1[j];

F[2][i] = CFace(4, p4);

}

for (int i = 0; i < 24; i++)//上腰刻面

{

int j = (i + 1) % 24;

if (i % 2 == 0)

{

p3[0] = P3[i];

p3[1] = P3[j];

p3[2] = P2[i / 2];

F[3][i] = CFace(3, p3);

}

else

{

p3[0] = P3[i];

p3[1] = P3[j];

p3[2] = P2[(i - 1) / 2];

F[3][i] = CFace(3, p3);

}

}

for (int i = 0; i < 24; i++)//腰围

{

int j = (i + 1) % 24;

p4[0] = P3[i];

p4[1] = P4[i];

p4[2] = P4[j];

p4[3] = P3[j];

F[4][i] = CFace(4, p4);

}

for (int i = 0; i < 24; i++)//下腰刻面

{

int j = (i + 1) % 24;

if (i % 2 == 0)

{

p3[0] = P4[i];

p3[1] = P5[i / 2];

p3[2] = P4[j];

F[5][i] = CFace(3, p3);

}

else

{

p3[0] = P4[i];

p3[1] = P5[(i - 1) / 2];

p3[2] = P4[j];

F[5][i] = CFace(3, p3);

}

}

for (int i = 0; i < 12; i++)//底部刻面

{

int j = (i + 1) % 12;

p4[0] = P5[i];

p4[1] = P;

p4[2] = P5[j];

int k = (2 \* i + 2) % 24;

p4[3] = P4[k];

F[6][i] = CFace(4, p4);

}

}

void CDiamond::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 CDiamond::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;

ViewCoorP.c = P.c;

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

ScreenCoorP.y = d \* ViewCoorP.y / ViewCoorP.z;

ScreenCoorP.c = ViewCoorP.c;

}

void CDiamond::SetViewPoint()//设置视点

{

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

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

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

}

void CDiamond::Draw(CDC\* pDC)//绘制钻石线框模型

{

for (int nFace = 0; nFace < 1; nFace++)//桌面

Model(pDC, 0, nFace);

for (int nFace = 0; nFace < 12; nFace++)//三角刻面

Model(pDC, 1, nFace);

for (int nFace = 0; nFace < 12; nFace++)//斜面

Model(pDC, 2, nFace);

for (int nFace = 0; nFace < 24; nFace++)//上腰刻面

Model(pDC, 3, nFace);

for (int nFace = 0; nFace < 24; nFace++)//腰围

Model(pDC, 4, nFace);

for (int nFace = 0; nFace < 24; nFace++)//下腰刻面

Model(pDC, 5, nFace);

for (int nFace = 0; nFace < 12; nFace++)//底部刻面

Model(pDC, 6, nFace);

}

void CDiamond::Model(CDC\* pDC, int n, int nface)

{

CVector3 VS(F[n][nface].FPoint[0], ViewPoint);

if (Dot(F[n][nface].Normal, VS) >= 0)

{

CP2d t;

for (int nEdge = 0; nEdge < F[n][nface].En; nEdge++)

{

PersPectiveProjection(F[n][nface].FPoint[nEdge]);

if (nEdge == 0)

{

pDC->MoveTo(ROUND(nHWidth + ScreenCoorP.x), ROUND(nHHeight - ScreenCoorP.y));

t = ScreenCoorP;

}

else

{

pDC->LineTo(ROUND(nHWidth + ScreenCoorP.x), ROUND(nHHeight - ScreenCoorP.y));

}

}

pDC->LineTo(ROUND(nHWidth + t.x), ROUND(nHHeight - t.y));

}

}

2.CTestView类：

public:

void DoubleBuffer(CDC\* pDC);//双缓冲绘图

void DrawObject(CDC\* pDC);//绘制五角星线框

protected:

CTransform3 tran;//变换对象

BOOL bPlay;//动画开关

CDiamond diamond;//钻石对象

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

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

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

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

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

CDC memDC;

memDC.CreateCompatibleDC(pDC);

CBitmap NewBitmap, \*pOldBitmap;

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

pOldBitmap = memDC.SelectObject(&NewBitmap);

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

DrawObject(&memDC);

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

memDC.SelectObject(pOldBitmap);

NewBitmap.DeleteObject();

}

void CTestView::DrawObject(CDC \* pDC)

{

diamond.Draw(pDC);

}

1. **实现效果：**

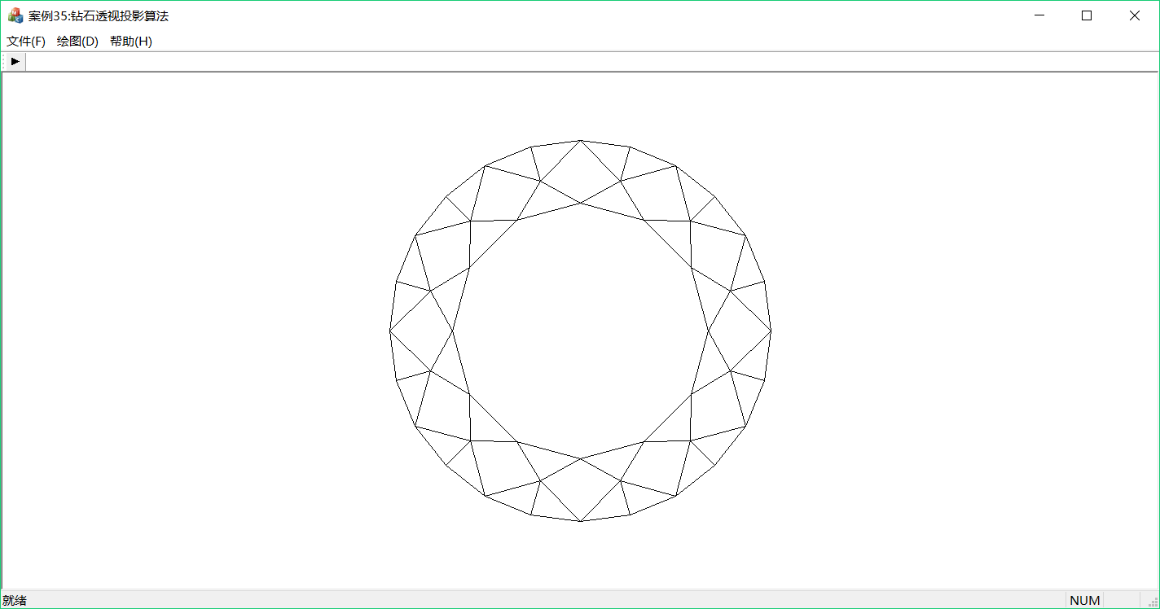


图34-3钻石透视投影效果图