**案例22-立方体正交投影线框模型算法**

1. **主要知识点**

设物体上任意一点的三维坐标为，正投影后的三维坐标为,则投影方程为

齐次坐标矩阵表示为：

正交投影变换矩阵为：

1. **案例描述：**

本案例利用正交投影算法，绘制立方体正交投影线框模型。

1. **实现步骤：**
2. 添加基础类：CRGB、CP2i、CP2d、CP3d、Cline、CFace、CTransform3。
3. 在CTestView中计算顶点坐标、读入面表，绘制图形。
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**

CTestView类：

public:

void ReadPoint();//读入点表

void ReadFace();//读入面表

void DoubleBuffer(CDC \*pDC);//双缓冲函数

void DrawObject(CDC \*pDC);//绘制立方体线框

protected:

CP3d P[8];//点表

CFace F[6];//面表

int nClientWidth, nClientHeight;//屏幕客户区宽度和高度

int nHWidth, nHHeight;//屏幕客户区的半宽和半高

double Alpha, Beta;//绕X轴旋转α角，绕Y轴旋转β角

BOOL bPlay;//动画开关

CTransform3 tran;//三维几何变换对象

void CTestView::ReadPoint()//点表

{

double a = 150;

P[0].x = -a; P[0].y = -a; P[0].z = -a;

P[1].x = +a; P[1].y = -a; P[1].z = -a;

P[2].x = +a; P[2].y = +a; P[2].z = -a;

P[3].x = -a; P[3].y = +a; P[3].z = -a;

P[4].x = -a; P[4].y = -a; P[4].z = +a;

P[5].x = +a; P[5].y = -a; P[5].z = +a;

P[6].x = +a; P[6].y = +a; P[6].z = +a;

P[7].x = -a; P[7].y = +a; P[7].z = +a;

}

void CTestView::ReadFace()//面表

{

//面的顶点数和面的顶点索引

F[0].SetVNum(4); F[0].vIndex[0] = 4; F[0].vIndex[1] = 5;

F[0].vIndex[2] = 6; F[0].vIndex[3] = 7;//前面

F[1].SetVNum(4); F[1].vIndex[0] = 0; F[1].vIndex[1] = 3;

F[1].vIndex[2] = 2; F[1].vIndex[3] = 1;//后面

F[2].SetVNum(4); F[2].vIndex[0] = 0; F[2].vIndex[1] = 4;

F[2].vIndex[2] = 7; F[2].vIndex[3] = 3;//左面

F[3].SetVNum(4); F[3].vIndex[0] = 1; F[3].vIndex[1] = 2;

F[3].vIndex[2] = 6; F[3].vIndex[3] = 5;//右面

F[4].SetVNum(4); F[4].vIndex[0] = 2; F[4].vIndex[1] = 3;

F[4].vIndex[2] = 7; F[4].vIndex[3] = 6;//顶面

F[5].SetVNum(4); F[5].vIndex[0] = 0; F[5].vIndex[1] = 1;

F[5].vIndex[2] = 5; F[5].vIndex[3] = 4;//底面

}

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

{

CRect rect;//定义客户区

GetClientRect(&rect);//获得客户区大小

nClientWidth = rect.Width();//屏幕客户区宽度

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

nHWidth = nClientWidth / 2;//屏幕客户区半宽

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

CDC memDC;

memDC.CreateCompatibleDC(pDC);

CBitmap NewBitmap, \*pOldBitmap;

NewBitmap.CreateCompatibleBitmap(pDC, nClientWidth, nClientHeight);

pOldBitmap = memDC.SelectObject(&NewBitmap);

DrawObject(&memDC);

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

memDC.SelectObject(pOldBitmap);

NewBitmap.DeleteObject();

}

void CTestView::DrawObject(CDC \*pDC)//绘制立方体线框

{

CP3d Vertex[4];

CLine \*line = new CLine;

for (int nFace = 0; nFace < 6; nFace++)//面循环

{

for (int nVertex = 0; nVertex < F[nFace].vNum; nVertex++)

Vertex[nVertex] = P[F[nFace].vIndex[nVertex]];

CP3d ViewPoint(Vertex[0].x, Vertex[0].y, Vertex[0].z + 1000);

CVector3 ViewVector(Vertex[0], ViewPoint);//面的视矢量

ViewVector.Normalize();//单位化视向量

CVector3 V01(Vertex[0], Vertex[1]);//面的一条边矢量

CVector3 V02(Vertex[0], Vertex[2]);//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

{

line->MoveTo(pDC, ROUND(nHWidth + Vertex[0].x),

ROUND(nHHeight - Vertex[0].y), CRGB(1.0, 1.0, 1.0));

line->LineTo(pDC, ROUND(nHWidth + Vertex[1].x),

ROUND(nHHeight - Vertex[1].y), CRGB(1.0, 1.0, 1.0));

line->LineTo(pDC, ROUND(nHWidth + Vertex[2].x),

ROUND(nHHeight - Vertex[2].y), CRGB(1.0, 1.0, 1.0));

line->LineTo(pDC, ROUND(nHWidth + Vertex[3].x),

ROUND(nHHeight - Vertex[3].y), CRGB(1.0, 1.0, 1.0));

line->LineTo(pDC, ROUND(nHWidth + Vertex[0].x),

ROUND(nHHeight - Vertex[0].y), CRGB(1.0, 1.0, 1.0));

}

}

delete line;

}

1. **实现效果：**

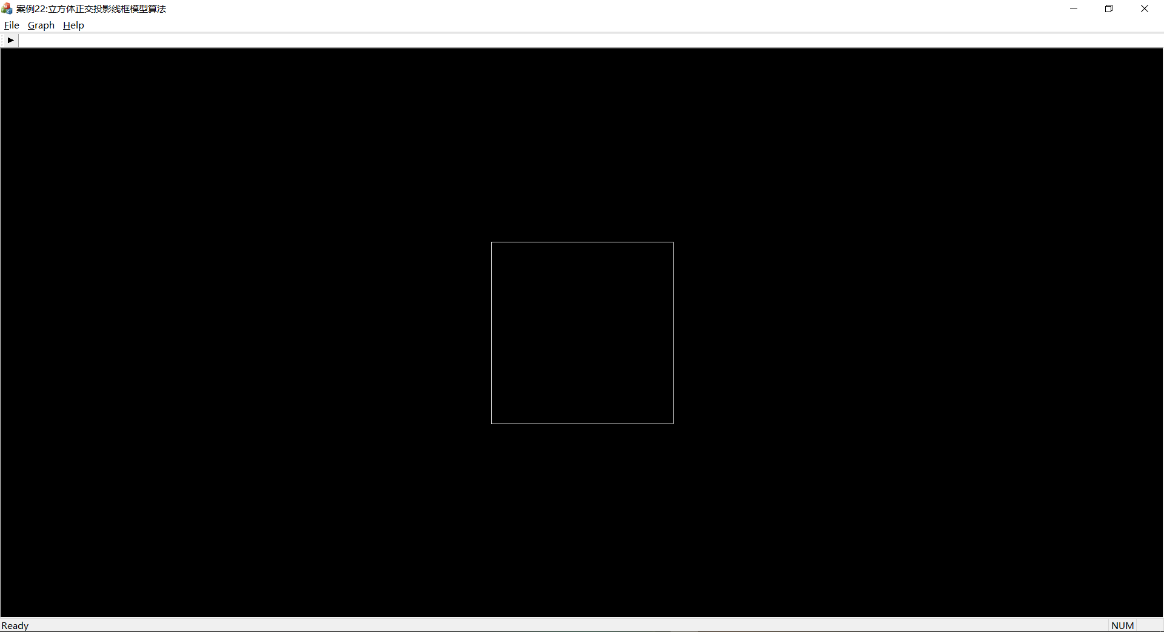


图1立方体正交投影线框模型算法效果图