**案例34-五角星透视投影算法**

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，绘制五角星。

五角星共有12个点和20各面组成。

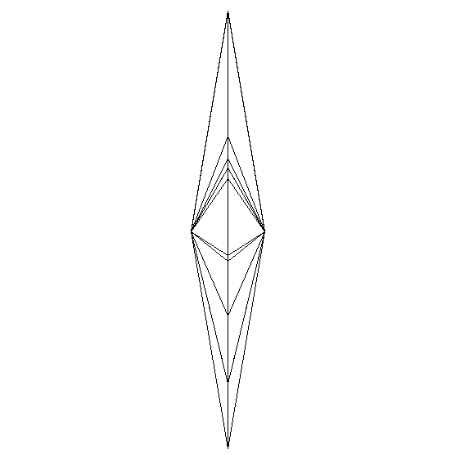
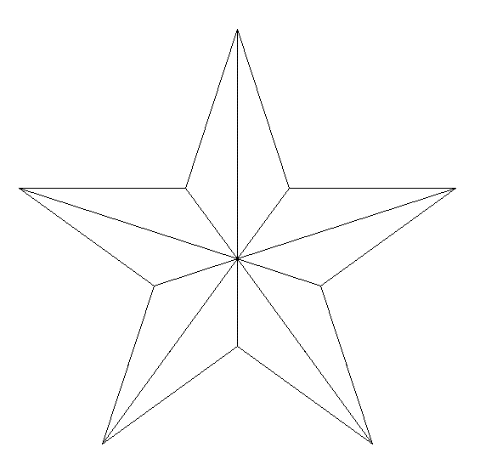


图34-1 主视图 图34-2 侧视图

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CPentastar类。
3. 在CPentastar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. CPentastar类：

public:

CPentastar();

virtual~CPentastar();

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

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

void Draw(CDC\* pDC);//绘制五角星线框

public:

CP3d P[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;//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

CPentastar::CPentastar()

{

R = 1000.0; d = 900.0;

Phi = 90.0; Theta = 0.0;

}

CPentastar::~CPentastar()

{

}

void CPentastar::ReadPoint()//点表

{

//顶点的三维坐标(x,y,z)

double b = 50;

double c = 300;

P[0].x = 0; P[0].y = 0; P[0].z = b;

P[1].x = 0; P[1].y = c; P[1].z = 0;

P[2].x = -c \* cos(72 \* PI / 180) \* tan(36 \* PI / 180); P[2].y = c \* cos(72 \* PI / 180); P[2].z = 0;

P[3].x = -c \* sin(72 \* PI / 180); P[3].y = c \* cos(72 \* PI / 180); P[3].z = 0;

P[4].x = -c \* cos(72 \* PI / 180) \* cos(18 \* PI / 180) / cos(36 \* PI / 180); P[4].y = -c \* cos(72 \* PI / 180)\*sin(18 \* PI / 180) / cos(36 \* PI / 180); P[4].z = 0;

P[5].x = -c \* sin(36 \* PI / 180); P[5].y = -c \* cos(36 \* PI / 180); P[5].z = 0;

P[6].x = 0; P[6].y = -c \* cos(72 \* PI / 180) / cos(36 \* PI / 180); P[6].z = 0;

P[7].x = c \* sin(36 \* PI / 180); P[7].y = -c \* cos(36 \* PI / 180); P[7].z = 0;

P[8].x = c \* cos(72 \* PI / 180) \* cos(18 \* PI / 180) / cos(36 \* PI / 180); P[8].y = -c \* cos(72 \* PI / 180) \* sin(18 \* PI / 180) / cos(36 \* PI / 180); P[8].z = 0;

P[9].x = c \* sin(72 \* PI / 180); P[9].y = c \* cos(72 \* PI / 180); P[9].z = 0;

P[10].x = c \* cos(72 \* PI / 180) \* tan(36 \* PI / 180); P[10].y = c \* cos(72 \* PI / 180); P[10].z = 0;

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

}

void CPentastar::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] = 1; F[0].vI[2] = 2;

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

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

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

F[4].SetNum(3); F[4].vI[0] = 0; F[4].vI[1] = 5; F[4].vI[2] = 6;

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

F[6].SetNum(3); F[6].vI[0] = 0; F[6].vI[1] = 7; F[6].vI[2] = 8;

F[7].SetNum(3); F[7].vI[0] = 0; F[7].vI[1] = 8; F[7].vI[2] = 9;

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

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

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

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

F[12].SetNum(3); F[12].vI[0] = 11; F[12].vI[1] = 9; F[12].vI[2] = 8;

F[13].SetNum(3); F[13].vI[0] = 11; F[13].vI[1] = 8; F[13].vI[2] = 7;

F[14].SetNum(3); F[14].vI[0] = 11; F[14].vI[1] = 7; F[14].vI[2] = 6;

F[15].SetNum(3); F[15].vI[0] = 11; F[15].vI[1] = 6; F[15].vI[2] = 5;

F[16].SetNum(3); F[16].vI[0] = 11; F[16].vI[1] = 5; F[16].vI[2] = 4;

F[17].SetNum(3); F[17].vI[0] = 11; F[17].vI[1] = 4; F[17].vI[2] = 3;

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

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

}

void CPentastar::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];

}

CP3d CPentastar::PersPectiveProjection(CP3d P)//透视投影

{

CP3d ViewCoorP;//观察坐标系内的点

CP3d ScreenCoorP;//屏幕坐标系内的点

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;

return ScreenCoorP;

}

void CPentastar::SetViewPoint()//设置视点

{

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

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

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

}

void CPentastar::Draw(CDC\* pDC)//绘制五角星线框模型

{

CP3d Vertex[3];

CLine\* line = new CLine;

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

{

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

Vertex[nVertex] = PersPectiveProjection(P[F[nFace].vI[nVertex]]);

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

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

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

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

}

delete line;

}

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

CPentastar pentastar;//五角星对象

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

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

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

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

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

CDC memDC;

memDC.CreateCompatibleDC(pDC);

CBitmap NewBitmap, \*pOldBitmap;

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

pOldBitmap = memDC.SelectObject(&NewBitmap);

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

DrawObject(&memDC);

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

memDC.SelectObject(pOldBitmap);

NewBitmap.DeleteObject();

}

void CTestView::DrawObject(CDC \* pDC)

{

pentastar.Draw(pDC);

}

1. **实现效果：**

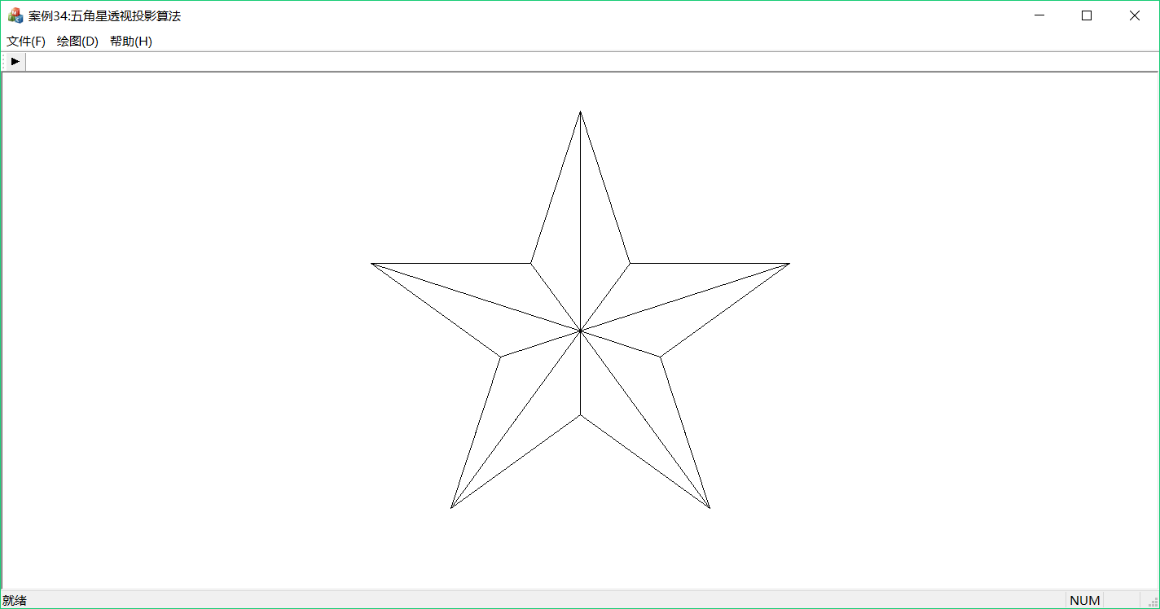


图34-3 五角星透视投影效果图