**案例37-HSV颜色模型算法**

文档编写：霍波魏

校稿/修订：孔令德

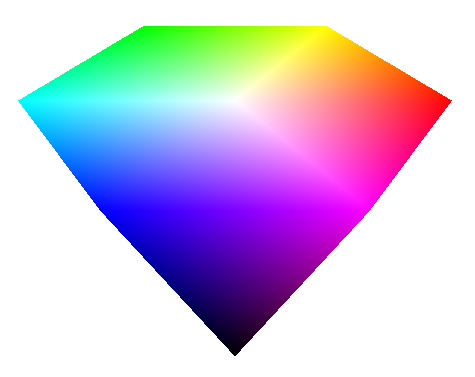
时间2019~2020

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（英文版）开发。

1. **知识点**

HSV颜色模型为一个底面向上的倒置六棱锥，底面中心位于HSV柱面坐标系的原点，如图37-1所示。锥顶为黑色，明度值为*V* = 0；锥底面中心为白色，明度值为*V* = 1；明度用百分比表示。6个顶点分别表示6种纯色。色调*H*在正六棱锥的垂直于*V*轴的各个截面内，沿逆时针方向用离开红色顶点的角度来表示，范围为0°～360°。饱和度*S*由棱锥上的点至*V*轴的距离决定，是所选颜色的纯度和该颜色的最大纯度的比率，用百分比表示。请注意当*S* = 0时，只有灰度，即非彩色光的饱和度为零。沿*V*轴正向，灰度由深变浅，形成不同的灰度等级。



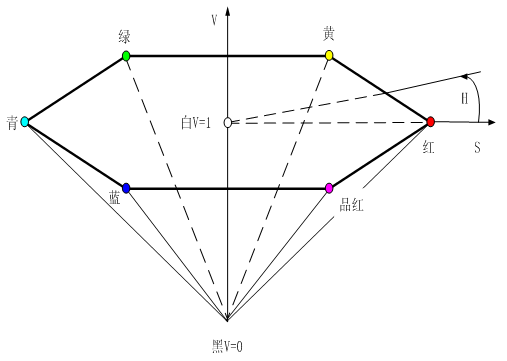


图37-1 线框模型 图37-2 表面模型

1. **实现步骤：**
2. 添加基础类与添加绘制六棱锥的CSixPyramid类。
3. 在CSixPyramid类中计算顶点坐标、读入面表，绘制图形。
4. 在CProjection类中对透视变换参数初始化、设置视点位置以及透视变换。
5. 在CTestView中添加消息响应函数，在OnDraw中调用DoubleBuffer函数。
6. **主要算法**

1. CSixPyramid类：

public:

CSixPyramid();

virtual~CSixPyramid();

void ReadVertex();//读入点表

void ReadFace();//读入面表

void Draw(CDC\* pDC);//绘制六棱锥表面

public:

CP3 V[8];//点表

CFace F[12];//面表

CProjection projection;

CSixPyramid::CSixPyramid()

{

}

CSixPyramid::~CSixPyramid()

{

}

void CSixPyramid::ReadVertex()//点表

{

//顶点的三维坐标(x,y,z),立方体边长为2a

double r = 240, h = 400;

V[0].x = 0; V[0].y = h / 3; V[0].z = 0; V[0].c = CRGB(1.0, 1.0, 1.0);//六棱锥白色底面中心

V[1].x = r; V[1].y = h / 3; V[1].z = 0; V[1].c = CRGB(1.0, 0.0, 0.0);//右侧红色顶点

V[2].x = r \* cos(PI / 3); V[2].y = h / 3; V[2].z = -r \* sin(PI / 3); V[2].c = CRGB(1.0, 1.0, 0.0);

V[3].x = -r \* cos(PI / 3); V[3].y = h / 3; V[3].z = -r \* sin(PI / 3); V[3].c = CRGB(0.0, 1.0, 0.0);

V[4].x = -r; V[4].y = h / 3; V[4].z = 0; V[4].c = CRGB(0.0, 1.0, 1.0);

V[5].x = -r \* cos(PI / 3); V[5].y = h / 3; V[5].z = r \* sin(PI / 3); V[5].c = CRGB(0.0, 0.0, 1.0);

V[6].x = r \* cos(PI / 3); V[6].y = h / 3; V[6].z = r \* sin(PI / 3); V[6].c = CRGB(1.0, 0.0, 1.0);

V[7].x = 0; V[7].y = -2 \* h / 3; V[7].z = 0; V[7].c = CRGB(0.0, 0.0, 0.0);//六棱锥黑色顶点

}

void CSixPyramid::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;//底面

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] = 1;

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

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

F[8].SetNum(3); F[8].vI[0] = 3; F[8].vI[1] = 7; F[8].vI[2] = 4;

F[9].SetNum(3); F[9].vI[0] = 4; F[9].vI[1] = 7; F[9].vI[2] = 5;

F[10].SetNum(3); F[10].vI[0] = 5; F[10].vI[1] = 7; F[10].vI[2] = 6;

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

}

void CSixPyramid::Draw(CDC\* pDC)//绘制立方体表面

{

CPi2 Point[3];//透视投影后面的二维顶点数组

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

{

CVector ViewVector(V[F[nFace].vI[0]], projection.ViewPoint);//面的视矢量

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

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

F[nFace].fNormal.Normalize();//单位化法矢量

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

{

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

{

projection.PerProject(V[F[nFace].vI[nVertex]]);//透视投影

Point[nVertex].x = projection.ScreenP.x;

Point[nVertex].y = Round(projection.ScreenP.y);

Point[nVertex].c = projection.ScreenP.c;

}

CFill \*fill = new CFill;//动态分配内存

fill->SetPoint(Point, 3);//设置顶点

fill->CreateBucket();//建立桶表

fill->CreateEdge();//建立边表

fill->Gouraud(pDC);//填充面片

delete fill;//撤销内存

}

}

}

2.CTestView类：

public:

void DoubleBuffer(CDC\* pDC);//双缓冲绘图

void DrawObject(CDC\* pDC);// 绘制六棱锥表面

protected:

CTransform3 tran;//变换对象

BOOL bPlay;//动画开关

CSixPyramid sixpyramid;

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

{

CRect rect;//定义客户区矩形

GetClientRect(&rect);//获得客户区的大小

pDC->SetMapMode(MM\_ANISOTROPIC);//pDC自定义坐标系

pDC->SetWindowExt(rect.Width(), rect.Height());//设置窗口范围

pDC->SetViewportExt(rect.Width(), -rect.Height());//设置视区范围,x轴水平向右，y轴垂直向上

pDC->SetViewportOrg(rect.Width() / 2, rect.Height() / 2);//客户区中心为原点

CDC memDC;//内存DC

memDC.CreateCompatibleDC(pDC);//创建一个与显示pDC兼容的内存memDC

CBitmap NewBitmap, \*pOldBitmap;//内存中承载的临时位图

NewBitmap.CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());//创建兼容位图

pOldBitmap = memDC.SelectObject(&NewBitmap);//将兼容位图选入memDC

//memDC.FillSolidRect(rect,pDC->GetBkColor());//按原来背景填充客户区，否则是黑色

memDC.SetMapMode(MM\_ANISOTROPIC);//memDC自定义坐标系

memDC.SetWindowExt(rect.Width(), rect.Height());

memDC.SetViewportExt(rect.Width(), -rect.Height());

memDC.SetViewportOrg(rect.Width() / 2, rect.Height() / 2);

rect.OffsetRect(-rect.Width() / 2, -rect.Height() / 2);

DrawObject(&memDC);//向memDC绘制图形

pDC->BitBlt(rect.left, rect.top, rect.Width(), rect.Height(), &memDC, -rect.Width() / 2, -rect.Height() / 2, SRCCOPY);//将内存memDC中的位图拷贝到显示pDC中

memDC.SelectObject(pOldBitmap);//恢复位图

NewBitmap.DeleteObject();//删除位图

}

void CTestView::DrawObject(CDC \* pDC)

{

sixpyramid.Draw(pDC);

}

1. **实现效果**

HSV颜色模型效果如图37-1。

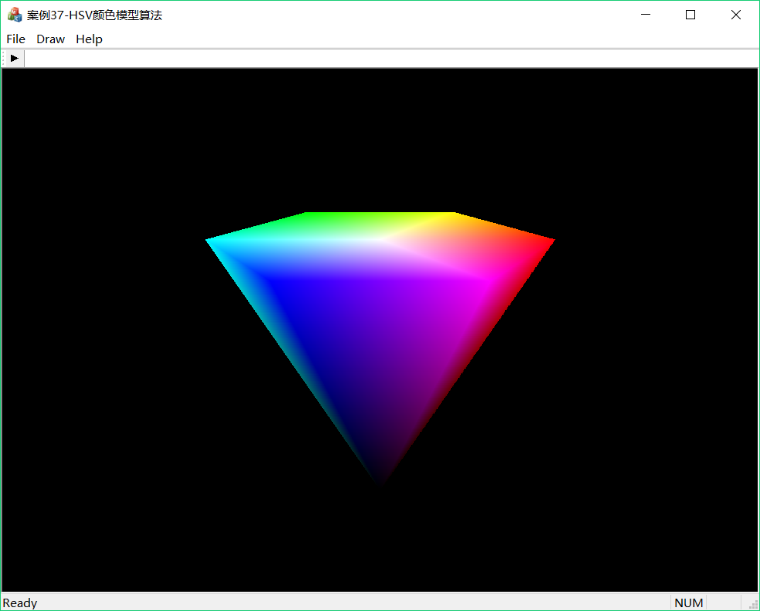


图37-1 HSV颜色模型效果图