**案例36-RGB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**

RGB颜色模型可以用一个单位立方体表示。若归一化R, G, B分量到区间[0, 1]内，则所定义的颜色位于RGB立方体内部。原点(0, 0, 0)代表黑色，顶点(1, 1, 1)代表白色。坐标轴上的3个立方体顶点(1, 0, 0), (0, 1, 0), (0, 0, 1)分别表示RGB三原色红色、绿色、蓝色；余下的3个顶点(1, 0, 1), (1, 1, 0), (0, 1, 1)则表示三原色的补色品红、黄色、青色。立方体对角线上的颜色是互补色。在立方体的主对角线上，颜色从黑色过渡到白色，各原色的变化率相等，产生了由黑到白的灰度变化，称为灰度色。灰度色是指纯黑、纯白及两者间的一系列过渡色，灰度色中不包含任何色调。例如，(0, 0, 0)代表黑色，(1, 1, 1)代表白色，而(0.5, 0.5, 0.5)代表其中一个灰度。只有当R, G, B三原色的变化率不同步时，才会出现彩色。

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

public:

CCube();

virtual~CCube();

void ReadVertex();//读入点表

void ReadFace();//读入面表

void Draw(CDC\* pDC);//绘制立方体表面

public:

CP3d V[8];//点表

CFace F[6];//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

CProjection projection;

void CCube::ReadVertex()//点表

{

//顶点的三维坐标(x,y,z),立方体边长为2a

double a = 150;

V[0].x = -a; V[0].y = -a; V[0].z = -a; V[0].c = CRGB(0.0, 0.0, 0.0);

V[1].x = +a; V[1].y = -a; V[1].z = -a; V[1].c = CRGB(1.0, 0.0, 0.0);

V[2].x = +a; V[2].y = +a; V[2].z = -a; V[2].c = CRGB(1.0, 1.0, 0.0);

V[3].x = -a; V[3].y = +a; V[3].z = -a; V[3].c = CRGB(0.0, 1.0, 0.0);

V[4].x = -a; V[4].y = -a; V[4].z = +a; V[4].c = CRGB(0.0, 0.0, 1.0);

V[5].x = +a; V[5].y = -a; V[5].z = +a; V[5].c = CRGB(1.0, 0.0, 1.0);

V[6].x = +a; V[6].y = +a; V[6].z = +a; V[6].c = CRGB(1.0, 1.0, 1.0);

V[7].x = -a; V[7].y = +a; V[7].z = +a; V[7].c = CRGB(0.0, 1.0, 1.0);

}

void CCube::ReadFace()//面表

{

//面的顶点数和面的顶点索引

F[0].SetNum(4); F[0].vI[0] = 4; F[0].vI[1] = 5; F[0].vI[2] = 6; F[0].vI[3] = 7;//前面

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

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

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

F[4].SetNum(4); F[4].vI[0] = 2; F[4].vI[1] = 3; F[4].vI[2] = 7; F[4].vI[3] = 6;//顶面

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

}

void CCube::Draw(CDC\* pDC)//绘制立方体表面

{

CPi2 Point[4];//透视投影后面的二维顶点数组

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

{

CFill \*fill = new CFill;//动态分配内存

CVector3 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//透视投影

Point[nVertex].x = (nHWidth + projection.ScreenP.x);

Point[nVertex].y = ROUND(nHHeight - projection.ScreenP.y);

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

}

fill->SetPoint(Point, 4);//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;

}

}

2.CTestView类：

public:

void DoubleBuffer(CDC\* pDC);//双缓冲绘图

void DrawObject(CDC\* pDC);// 绘制立方体表面

protected:

CTransform3 tran;//变换对象

BOOL bPlay;//动画开关

CCube cube;

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

{

CRect rect;//定义客户区

GetClientRect(&rect);//获得客户区的大小

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

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

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

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

CDC memDC;

memDC.CreateCompatibleDC(pDC);

CBitmap NewBitmap, \*pOldBitmap;

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

pOldBitmap = memDC.SelectObject(&NewBitmap);

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

DrawObject(&memDC);

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

memDC.SelectObject(pOldBitmap);

NewBitmap.DeleteObject();

}

void CTestView::DrawObject(CDC \* pDC)

{

cube.Draw(pDC);

}

1. **实现效果**

RGB颜色模型效果如图36-1所示。

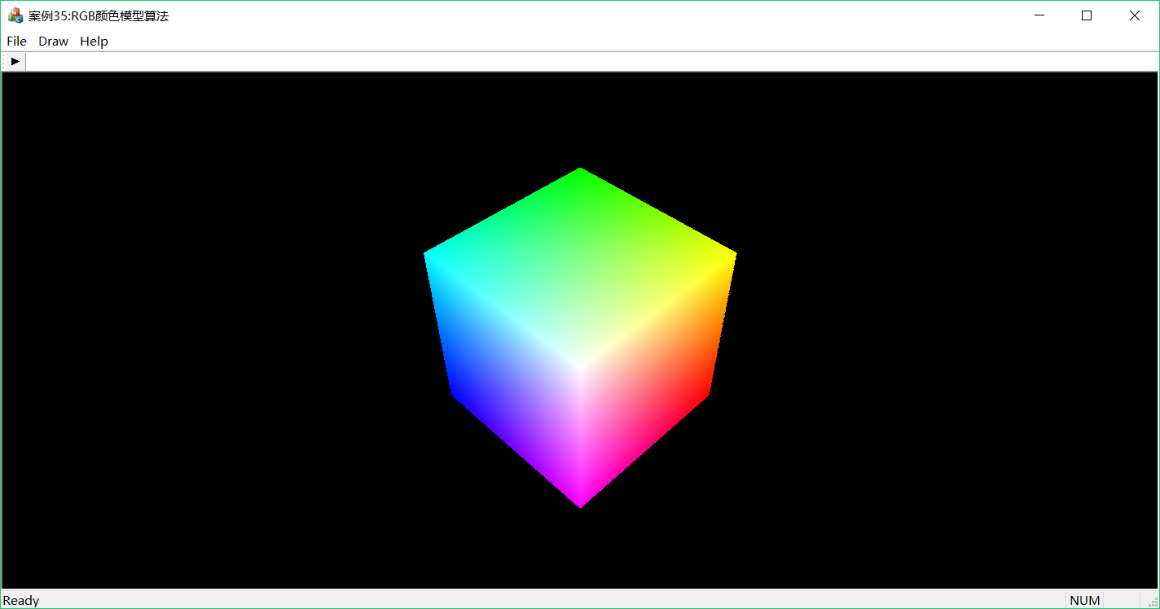


图36-1 RGB颜色模型效果图