**案例24-立方体颜色渐变算法**

1. **主要知识点**

颜色线性插值：

1. **案例描述：**

本案例利用立方体颜色渐变算法，绘制颜色渐变立方体。

1. **实现步骤：**
2. 添加基础类：CRGB、CP2i、CP2d、CP3d、Cline、CFace、CTransform3。
3. 添加绘制立方体的CCube类。
4. 在CCube中计算顶点坐标、读入面表，绘制图形，透视投影。
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.CCube类：

public:

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

CP2d PersPectiveProjecttion(CP3d P);//透视投影

void Draw(CDC\* pDC);//绘制立方体线框

public:

CP3d P[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

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

void CCube::ReadPoint()//点表

{

//顶点的三维坐标(x,y,z),立方体边长为2a

double a = 150;

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

//黑色

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

//红色

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

//黄色

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

//绿色

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

//蓝色

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

//品红

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

//白色

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

//青色

}

void CCube::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 CCube::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];

SetViewPoint();

}

void CCube::SetViewPoint()//设置视点

{

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

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

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

}

CP2d CCube::PersPectiveProjecttion(CP3d worldP)//透视变换

{

CP3d ViewP;

CP2d screenP;

ViewP.x = worldP.x \* k[3] - worldP.z \* k[1];//观察坐标系三维坐标

ViewP.y = -worldP.x \* k[8] + worldP.y \* k[2] - worldP.z \* k[7];

ViewP.z = -worldP.x \* k[6] - worldP.y \* k[4] - worldP.z \* k[5] + R;

ViewP.c = worldP.c;

screenP.x = d \* ViewP.x / ViewP.z;//屏幕二维坐标系

screenP.y = d \* ViewP.y / ViewP.z;

screenP.c = ViewP.c;

return screenP;

}

void CCube::Draw(CDC \* pDC)//绘制立方体线框模型

{

CP2d 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++)//顶点循环

Vertex[nVertex] = PersPectiveProjecttion(P[F[nFace].vIndex[nVertex]]);

CVector3 ViewVector(P[F[nFace].vIndex[0]], ViewPoint);//面的视矢量

ViewVector.Normalize();//单位化视向量

CVector3 V01(P[F[nFace].vIndex[0]], P[F[nFace].vIndex[1]]);//面的一条边矢量

CVector3 V02(P[F[nFace].vIndex[0]], P[F[nFace].vIndex[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), Vertex[0].c);

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

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

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

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

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

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

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

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

}

}

delete line;

}

2.CTestView类：

public:

void DoubleBuffer(CDC\* pDC);//双缓存绘图

void DrawObject(CDC\* pDC);//绘图

protected:

BOOL bPlay;//动画开关

CTransform3 tran;//变换对象

CCube cube;立方体类

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

GetClientRect(&rect);

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);

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：**

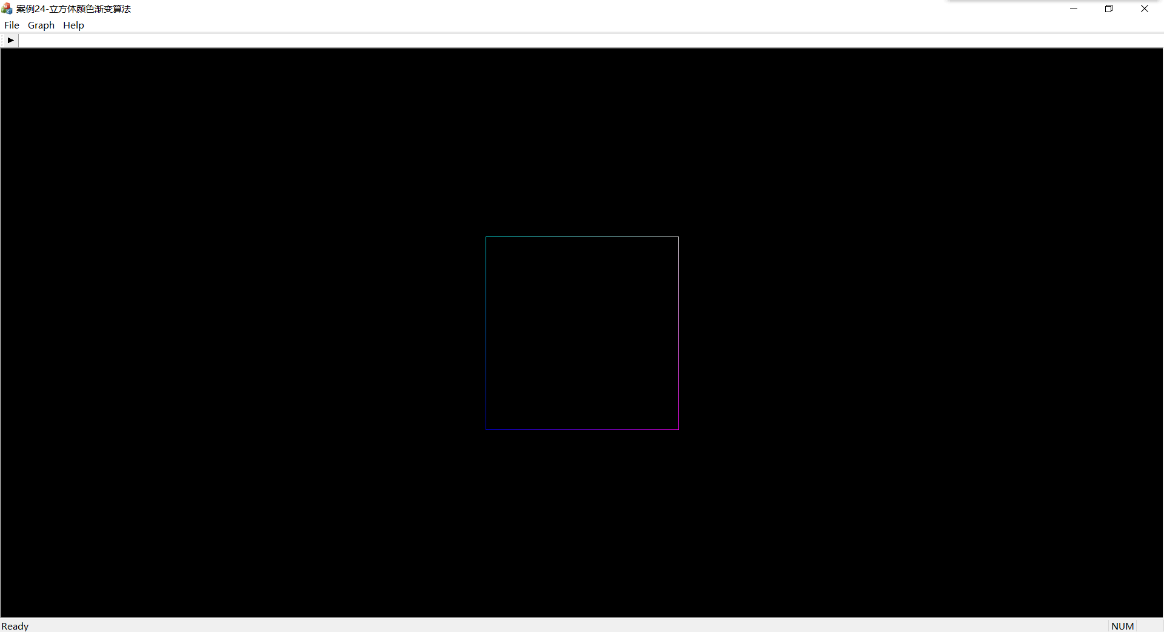


图1立方体颜色渐变算法效果图