# 6.9 颜色立方体演示程序

我们已经讲解了足够多的内容，现在我们可以开始编写一个简单的颜色立方体演示程序了。这个例子基本上包含了我们前面讲到的所有内容。读者应该对照前面的几节，仔细研究这些代码，直到把每一行代码都弄懂为止。注意，程序使用了6.8.1节编写的“color.fx”effect。

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// BoxDemo.cpp by Frank Luna (C) 2011 All Rights Reserved.

//

// Demonstrates rendering a colored box.

//

// Controls:

// Hold the left mouse button down and move the mouse to rotate.

// Hold the right mouse button down to zoom in and out.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "d3dApp.h"

#include "d3dx11Effect.h"

#include "MathHelper.h"

struct Vertex

{

 XMFLOAT3 Pos;

 XMFLOAT4 Color;

};

class BoxApp : public D3DApp

{

public:

 BoxApp(HINSTANCE hInstance);

 ~BoxApp();

 bool Init();

 void OnResize();

 void UpdateScene(float dt);

 void DrawScene();

 void OnMouseDown(WPARAM btnState, int x, int y);

 void OnMouseUp(WPARAM btnState, int x, int y);

 void OnMouseMove(WPARAM btnState, int x, int y);

private:

 void BuildGeometryBuffers();

 void BuildFX();

 void BuildVertexLayout();

private:

 ID3D11Buffer\* mBoxVB;

 ID3D11Buffer\* mBoxIB;

 ID3DX11Effect\* mFX;

 ID3DX11EffectTechnique\* mTech;

 ID3DX11EffectMatrixVariable\* mfxWorldViewProj;

 ID3D11InputLayout\* mInputLayout;

 XMFLOAT4X4 mWorld;

 XMFLOAT4X4 mView;

 XMFLOAT4X4 mProj;

 float mTheta;

 float mPhi;

 float mRadius;

 POINT mLastMousePos;

};

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE prevInstance,

 PSTR cmdLine, int showCmd)

{

 // Enable run-time memory check for debug builds.

#if defined(DEBUG) | defined(\_DEBUG)

 \_CrtSetDbgFlag( \_CRTDBG\_ALLOC\_MEM\_DF | \_CRTDBG\_LEAK\_CHECK\_DF );

#endif

 BoxApp theApp(hInstance);

 if( !theApp.Init() )

 return 0;

 return theApp.Run();

}

BoxApp::BoxApp(HINSTANCE hInstance)

: D3DApp(hInstance), mBoxVB(0), mBoxIB(0), mFX(0), mTech(0),

 mfxWorldViewProj(0), mInputLayout(0),

 mTheta(1.5f\*MathHelper::Pi), mPhi(0.25f\*MathHelper::Pi), mRadius(5.0f)

{

 mMainWndCaption = L"Box Demo";

 mLastMousePos.x = 0;

 mLastMousePos.y = 0;

 XMMATRIX I = XMMatrixIdentity();

 XMStoreFloat4x4(&mWorld, I);

 XMStoreFloat4x4(&mView, I);

 XMStoreFloat4x4(&mProj, I);

}

BoxApp::~BoxApp()

{

 ReleaseCOM(mBoxVB);

 ReleaseCOM(mBoxIB);

 ReleaseCOM(mFX);

 ReleaseCOM(mInputLayout);

}

bool BoxApp::Init()

{

 if(!D3DApp::Init())

 return false;

 BuildGeometryBuffers();

 BuildFX();

 BuildVertexLayout();

 return true;

}

void BoxApp::OnResize()

{

 D3DApp::OnResize();

 // 当窗口大小改变时，需要更新横纵比，并重新计算投影矩阵

 XMMATRIX P = XMMatrixPerspectiveFovLH(0.25f\*MathHelper::Pi, AspectRatio(), 1.0f, 1000.0f);

 XMStoreFloat4x4(&mProj, P);

}

void BoxApp::UpdateScene(float dt)

{

 // Convert Spherical to Cartesian coordinates.

 float x = mRadius\*sinf(mPhi)\*cosf(mTheta);

 float z = mRadius\*sinf(mPhi)\*sinf(mTheta);

 float y = mRadius\*cosf(mPhi);

 // 创建视矩阵

 XMVECTOR pos = XMVectorSet(x, y, z, 1.0f);

 XMVECTOR target = XMVectorZero();

 XMVECTOR up = XMVectorSet(0.0f, 1.0f, 0.0f, 0.0f);

 XMMATRIX V = XMMatrixLookAtLH(pos, target, up);

 XMStoreFloat4x4(&mView, V);

}

void BoxApp::DrawScene()

{

 md3dImmediateContext->ClearRenderTargetView(mRenderTargetView, reinterpret\_cast<const float\*>(&Colors::LightSteelBlue));

 md3dImmediateContext->ClearDepthStencilView(mDepthStencilView, D3D11\_CLEAR\_DEPTH|D3D11\_CLEAR\_STENCIL, 1.0f, 0);

 md3dImmediateContext->IASetInputLayout(mInputLayout);

 md3dImmediateContext->IASetPrimitiveTopology(D3D11\_PRIMITIVE\_TOPOLOGY\_TRIANGLELIST);

 UINT stride = sizeof(Vertex);

 UINT offset = 0;

 md3dImmediateContext->IASetVertexBuffers(0, 1, &mBoxVB, &stride, &offset);

 md3dImmediateContext->IASetIndexBuffer(mBoxIB, DXGI\_FORMAT\_R32\_UINT, 0);

 // Set constants

 XMMATRIX world = XMLoadFloat4x4(&mWorld);

 XMMATRIX view = XMLoadFloat4x4(&mView);

 XMMATRIX proj = XMLoadFloat4x4(&mProj);

 XMMATRIX worldViewProj = world\*view\*proj;

 mfxWorldViewProj->SetMatrix(reinterpret\_cast<float\*>(&worldViewProj));

 D3DX11\_TECHNIQUE\_DESC techDesc;

 mTech->GetDesc( &techDesc );

 for(UINT p = 0; p < techDesc.Passes; ++p)

 {

 mTech->GetPassByIndex(p)->Apply(0, md3dImmediateContext);

 // 立方体有36个索引

 md3dImmediateContext->DrawIndexed(36, 0, 0);

 }

 HR(mSwapChain->Present(0, 0));

}

void BoxApp::OnMouseDown(WPARAM btnState, int x, int y)

{

 mLastMousePos.x = x;

 mLastMousePos.y = y;

 SetCapture(mhMainWnd);

}

void BoxApp::OnMouseUp(WPARAM btnState, int x, int y)

{

 ReleaseCapture();

}

void BoxApp::OnMouseMove(WPARAM btnState, int x, int y)

{

 if( (btnState & MK\_LBUTTON) != 0 )

 {

 // Make each pixel correspond to a quarter of a degree.

 float dx = XMConvertToRadians(0.25f\*static\_cast<float>(x - mLastMousePos.x));

 float dy = XMConvertToRadians(0.25f\*static\_cast<float>(y - mLastMousePos.y));

 // Update angles based on input to orbit camera around box.

 mTheta += dx;

 mPhi += dy;

 // Restrict the angle mPhi.

 mPhi = MathHelper::Clamp(mPhi, 0.1f, MathHelper::Pi-0.1f);

 }

 else if( (btnState & MK\_RBUTTON) != 0 )

 {

 // Make each pixel correspond to 0.005 unit in the scene.

 float dx = 0.005f\*static\_cast<float>(x - mLastMousePos.x);

 float dy = 0.005f\*static\_cast<float>(y - mLastMousePos.y);

 // Update the camera radius based on input.

 mRadius += dx - dy;

 // Restrict the radius.

 mRadius = MathHelper::Clamp(mRadius, 3.0f, 15.0f);

 }

 mLastMousePos.x = x;

 mLastMousePos.y = y;

}

void BoxApp::BuildGeometryBuffers()

{

 // 创建顶点缓冲

 Vertex vertices[] =

 {

 { XMFLOAT3(-1.0f, -1.0f, -1.0f), (const float\*)&Colors::White },

 { XMFLOAT3(-1.0f, +1.0f, -1.0f), (const float\*)&Colors::Black },

 { XMFLOAT3(+1.0f, +1.0f, -1.0f), (const float\*)&Colors::Red },

 { XMFLOAT3(+1.0f, -1.0f, -1.0f), (const float\*)&Colors::Green },

 { XMFLOAT3(-1.0f, -1.0f, +1.0f), (const float\*)&Colors::Blue },

 { XMFLOAT3(-1.0f, +1.0f, +1.0f), (const float\*)&Colors::Yellow },

 { XMFLOAT3(+1.0f, +1.0f, +1.0f), (const float\*)&Colors::Cyan },

 { XMFLOAT3(+1.0f, -1.0f, +1.0f), (const float\*)&Colors::Magenta }

 };

 D3D11\_BUFFER\_DESC vbd;

 vbd.Usage = D3D11\_USAGE\_IMMUTABLE;

 vbd.ByteWidth = sizeof(Vertex) \* 8;

 vbd.BindFlags = D3D11\_BIND\_VERTEX\_BUFFER;

 vbd.CPUAccessFlags = 0;

 vbd.MiscFlags = 0;

 vbd.StructureByteStride = 0;

 D3D11\_SUBRESOURCE\_DATA vinitData;

 vinitData.pSysMem = vertices;

 HR(md3dDevice->CreateBuffer(&vbd, &vinitData, &mBoxVB));

 // 创建索引缓冲

 UINT indices[] = {

 // 前表面

 0, 1, 2,

 0, 2, 3,

 // 后表面

 4, 6, 5,

 4, 7, 6,

 // 左表面

 4, 5, 1,

 4, 1, 0,

 // 右表面

 3, 2, 6,

 3, 6, 7,

 // 上表面

 1, 5, 6,

 1, 6, 2,

 // 下表面

 4, 0, 3,

 4, 3, 7

 };

 D3D11\_BUFFER\_DESC ibd;

 ibd.Usage = D3D11\_USAGE\_IMMUTABLE;

 ibd.ByteWidth = sizeof(UINT) \* 36;

 ibd.BindFlags = D3D11\_BIND\_INDEX\_BUFFER;

 ibd.CPUAccessFlags = 0;

 ibd.MiscFlags = 0;

 ibd.StructureByteStride = 0;

 D3D11\_SUBRESOURCE\_DATA iinitData;

 iinitData.pSysMem = indices;

 HR(md3dDevice->CreateBuffer(&ibd, &iinitData, &mBoxIB));

}

void BoxApp::BuildFX()

{

 DWORD shaderFlags = 0;

 #if defined( DEBUG ) || defined( \_DEBUG )

 shaderFlags |= D3D10\_SHADER\_DEBUG;

 shaderFlags |= D3D10\_SHADER\_SKIP\_OPTIMIZATION;

 #endif

 ID3D10Blob\* compiledShader = 0;

 ID3D10Blob\* compilationMsgs = 0;

 HRESULT hr = D3DX11CompileFromFile(L"FX/color.fx", 0, 0, 0, "fx\_5\_0", shaderFlags,

 0, 0, &compiledShader, &compilationMsgs, 0);

 // compilationMsgs中包含错误或警告信息

 if( compilationMsgs != 0 )

 {

 MessageBoxA(0, (char\*)compilationMsgs->GetBufferPointer(), 0, 0);

 ReleaseCOM(compilationMsgs);

 }

 // 就算没有compilationMsgs，也需要确保没有其他错误

 if(FAILED(hr))

 {

 DXTrace(\_\_FILE\_\_, (DWORD)\_\_LINE\_\_, hr, L"D3DX11CompileFromFile", true);

 }

 HR(D3DX11CreateEffectFromMemory(compiledShader->GetBufferPointer(), compiledShader->GetBufferSize(),

 0, md3dDevice, &mFX));

 // 编译完成释放资源

 ReleaseCOM(compiledShader);

 mTech = mFX->GetTechniqueByName("ColorTech");

 mfxWorldViewProj = mFX->GetVariableByName("gWorldViewProj")->AsMatrix();

}

void BoxApp::BuildVertexLayout()

{

 // 顶点输入布局描述

 D3D11\_INPUT\_ELEMENT\_DESC vertexDesc[] =

 {

 {"POSITION", 0, DXGI\_FORMAT\_R32G32B32\_FLOAT, 0, 0, D3D11\_INPUT\_PER\_VERTEX\_DATA, 0},

 {"COLOR", 0, DXGI\_FORMAT\_R32G32B32A32\_FLOAT, 0, 12, D3D11\_INPUT\_PER\_VERTEX\_DATA, 0}

 };

 // 创建顶点输入布局

 D3DX11\_PASS\_DESC passDesc;

 mTech->GetPassByIndex(0)->GetDesc(&passDesc);

 HR(md3dDevice->CreateInputLayout(vertexDesc, 2, passDesc.pIAInputSignature,

 passDesc.IAInputSignatureSize, &mInputLayout));

}

****

**图6.7 立方体演示程序的屏幕截图**