**武汉纺织大学**

**《Direct3D图形编程》上机实验报告**

**题目:** **融合技术**

**成 绩：**

**学 号： 1604240705**

**姓 名： 胡 莲**

**班 级： 计算机11603**

**指导教师： 李 敏**

**报告日期：2019 年4月14日**

1. **实验1**
2. 题目

绘制一个有4个面的锥体，并用图1所示的图像对锥体的四个面做纹理映射，使得每个面看起来颜色都不一样。

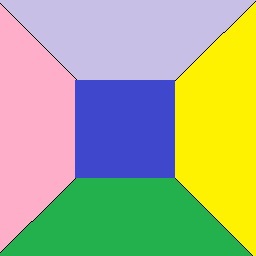


图1 纹理图像

1. 实现代码

在d3dInit程序框架上修改代码：

1. 全局变量：

IDirect3DDevice9\* Device = 0;

const int Width = 640;

const int Height = 480;

IDirect3DVertexBuffer9 \* VB=0;

IDirect3DTexture9 \* Tex=0;

1. 定义灵活顶点格式：

struct Vertex

{

Vertex(){}

Vertex(float x, float y, float z, float nx, float ny, float nz, float u, float v)

{

\_x = x; \_y = y; \_z = z; \_nx = nx; \_ny = ny; \_nz = nz; \_u = u; \_v = v;

}

float \_x, \_y, \_z;

float \_nx, \_ny, \_nz;

float \_u,\_v;

static const DWORD FVF;

};

const DWORD Vertex::FVF = D3DFVF\_XYZ | D3DFVF\_NORMAL | D3DFVF\_TEX1;

1. Setup()函数：

Device->CreateVertexBuffer(

12 \* sizeof(Vertex),

D3DUSAGE\_WRITEONLY,

Vertex::FVF,

D3DPOOL\_MANAGED,

&VB,

0);

Vertex\* v;

VB->Lock(0,0,(void\*\*)&v,0);

// front face

v[0] = Vertex(-1.0f, 0.0f, -1.0f, 0.0f, 0.707f, -0.707f,0.0f,1.0f);

v[1] = Vertex( 0.0f, 1.0f, 0.0f, 0.0f, 0.707f, -0.707f,0.5f,0.5f);

v[2] = Vertex( 1.0f, 0.0f, -1.0f, 0.0f, 0.707f, -0.707f,1.0f,1.0f);

// left face

v[3] = Vertex(-1.0f, 0.0f, 1.0f, -0.707f, 0.707f, 0.0f,0.0f,1.0f);

v[4] = Vertex( 0.0f, 1.0f, 0.0f, -0.707f, 0.707f, 0.0f,0.5f,0.5f);

v[5] = Vertex(-1.0f, 0.0f, -1.0f, -0.707f, 0.707f, 0.0f,0.0f,0.0f);

// right face

v[6] = Vertex( 1.0f, 0.0f, -1.0f, 0.707f, 0.707f, 0.0f,1.0f,1.0f);

v[7] = Vertex( 0.0f, 1.0f, 0.0f, 0.707f, 0.707f, 0.0f,0.5f,0.5f);

v[8] = Vertex( 1.0f, 0.0f, 1.0f, 0.707f, 0.707f, 0.0f,1.0f,0.0f);

// back face

v[9] = Vertex( 1.0f, 0.0f, 1.0f, 0.0f, 0.707f, 0.707f,1.0f,0.0f);

v[10] = Vertex( 0.0f, 1.0f, 0.0f, 0.0f, 0.707f, 0.707f,0.5f,0.5f);

v[11] = Vertex(-1.0f, 0.0f, 1.0f, 0.0f, 0.707f, 0.707f,0.0f,0.0f);

VB->Unlock();

D3DXCreateTextureFromFile(

Device,

"1.jpg",

&Tex);

Device->SetRenderState(D3DRS\_LIGHTING,false);

D3DXVECTOR3 position(0.0f, 0.0f, -3.0f);

D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);

D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);

D3DXMATRIX V;

D3DXMatrixLookAtLH(&V, &position, &target, &up);

Device -> SetTransform(D3DTS\_VIEW, &V);

D3DXMATRIX proj;

D3DXMatrixPerspectiveFovLH(

&proj,

D3DX\_PI \* 0.5f,

(float)Width / (float)Height,

1.0f,

1000.0f);

Device -> SetTransform(D3DTS\_PROJECTION, &proj);

return true;

1. Cleanup()函数：

d3d::Release<IDirect3DVertexBuffer9\*>(VB);

d3d::Release<IDirect3DTexture9\*>(Tex);

1. Display()函数：

Device->Clear(0, 0, D3DCLEAR\_TARGET | D3DCLEAR\_ZBUFFER, 0xffffffff, 1.0f, 0);

D3DXMATRIX Ry;

static float y = 0.0f;

D3DXMatrixRotationY(&Ry,y);

y+=timeDelta;

Device->SetTransform(D3DTS\_WORLD,&Ry);

Device->BeginScene();

Device->SetStreamSource(0,VB,0,sizeof(Vertex));

Device->SetFVF(Vertex::FVF);

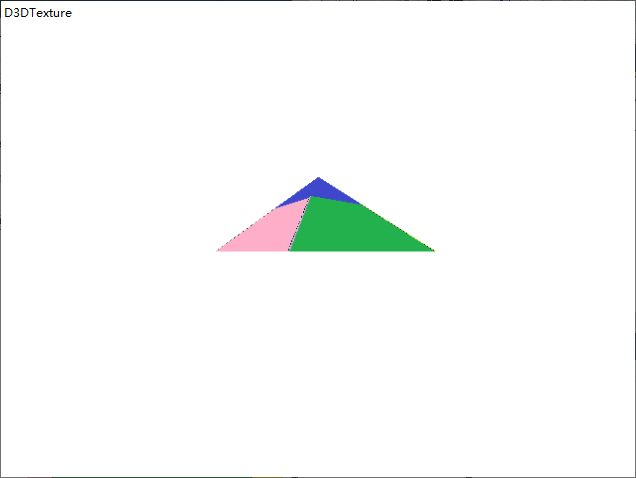
Device->SetTexture(0, Tex);

Device->DrawPrimitive(D3DPT\_TRIANGLELIST,0,4);

Device->EndScene();

Device->Present(0, 0, 0, 0);

1. 程序运行结果



1. **总结**

纹理贴图的顺序要与顶点缓存的顺序一致。

1. **实验２**
2. 题目

绘制一个立方体，并用图3所示的图像对立方体的每个面做纹理映射，使立方体看起来像是一个骰子。

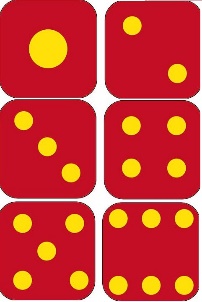


图3 纹理图像

1. 实现代码

在实验一的基础上修改代码：

1. Setup()函数：

Device->CreateVertexBuffer(

36 \* sizeof(Vertex),

D3DUSAGE\_WRITEONLY,

Vertex::FVF,

D3DPOOL\_MANAGED,

&VB,

0);

Vertex\* v;

VB->Lock(0,0,(void\*\*)&v,0);

//V0(-1,0,-1) V1(-1,0,1) V2(1,0,1) V3(1,0,-1)

//V4(-1,1,-1) V5(-1,1,1) V6(1,1,1) V7(1,1,-1)

v[0] = Vertex(-1.0f,0.0f,-1.0f,0.0f,0.0f,-1.0f,0.0f,1.0f/3.0f);//前¡ã面? V0 V4 V7

v[1] = Vertex(-1.0f,2.0f,-1.0f,0.0f,0.0f,-1.0f,0.0f,0.0f);

v[2] = Vertex(1.0f, 2.0f, -1.0f,0.0f,0.0f,-1.0f,0.5f,0.0f);

v[3] = Vertex(-1.0f,0.0f,-1.0f,0.0f,0.0f,-1.0f,0.0f,1.0f/3.0f);//V0 V7 V3

v[4] = Vertex(1.0f, 2.0f, -1.0f,0.0f,0.0f,-1.0f,0.5f,0.0f);

v[5] = Vertex(1.0f,0.0f,-1.0f,0.0f,0.0f,-1.0f,0.5f,1.0f/3.0f);

v[6] = Vertex(-1.0f,2.0f,-1.0f,0.0f,1.0f,0.0f,0.5f,1.0f/3.0f);//顶£¤面?V4 V5 V6

v[7] = Vertex(-1.0f,2.0f,1.0f,0.0f,1.0f,0.0f,0.5f,0.0f);

v[8] = Vertex(1.0f,2.0f,1.0f,0.0f,1.0f,0.0f,1.0f,0.0f);

v[9] = Vertex(-1.0f,2.0f,-1.0f,0.0f,1.0f,0.0f,0.5f,1.0f/3.0f);//V4 V6 V7

v[10] = Vertex(1.0f,2.0f,1.0f,0.0f,1.0f,0.0f,1.0f,0.0f);

v[11] = Vertex(1.0f,2.0f, -1.0f,0.0f,1.0f,0.0f,1.0f,1.0f/3.0f);

v[12] = Vertex(1.0f,0.0f,-1.0f,1.0f,0.0f,0.0f,0.0f,2.0f/3.0f);//右®¨°面? V3 V7 V6

v[13] = Vertex(1.0f,2.0f, -1.0f,1.0f,0.0f,0.0f,0.0f,1.0f/3.0f);

v[14] = Vertex(1.0f,2.0f,1.0f,1.0f,0.0f,0.0f,0.5f,1.0f/3.0f);

v[15] = Vertex(1.0f,0.0f,-1.0f,1.0f,0.0f,0.0f,0.0f,2.0f/3.0f);//V3 V6 V2

v[16] = Vertex(1.0f,2.0f,1.0f,1.0f,0.0f,0.0f,0.5f,1.0f/3.0f);

v[17] = Vertex(1.0f,0.0f,1.0f,1.0f,0.0f,0.0f,0.5f,2.0f/3.0f);

v[18] = Vertex(-1.0f,0.0f,-1.0f,-1.0f,0.0f,0.0f,0.5f,2.0f/3.0f);//左Á¨®面? V0 V5 V4

v[19] = Vertex(-1.0f,2.0f,1.0f,-1.0f,0.0f,0.0f,1.0f,1.0f/3.0f);

v[20] = Vertex(-1.0f,2.0f,-1.0f,-1.0f,0.0f,0.0f,0.5f,1.0f/3.0f);

v[21] = Vertex(-1.0f,0.0f,-1.0f,-1.0f,0.0f,0.0f,0.5f,2.0f/3.0f);//V0 V1 V5

v[22] = Vertex(-1.0f,0.0f,1.0f,-1.0f,0.0f,0.0f,1.0f,2.0f/3.0f);

v[23] = Vertex(-1.0f,2.0f,1.0f,-1.0f,0.0f,0.0f,1.0f,1.0f/3.0f);

v[24] = Vertex(-1.0f,0.0f,1.0f,0.0f,0.0f,1.0f,0.0f,1.0f);//背À3面? V1 V6 V5

v[25] = Vertex(1.0f,2.0f,1.0f,0.0f,0.0f,1.0f,0.5f,2.0f/3.0f);

v[26] = Vertex(-1.0f,2.0f,1.0f,0.0f,0.0f,1.0f,0.0f,2.0f/3.0f);

v[27] = Vertex(-1.0f,0.0f,1.0f,0.0f,0.0f,1.0f,0.0f,1.0f);//V1 V2 V6

v[28] = Vertex(1.0f,0.0f,1.0f,0.0f,0.0f,1.0f,0.5f,1.0f);

v[29] = Vertex(1.0f,2.0f,1.0f,0.0f,0.0f,1.0f,0.5f,2.0f/3.0f);

v[30] = Vertex(-1.0f,0.0f,-1.0f,0.0f,-1.0f,0.0f,0.5f,1.0f);//底Ì¡Á面? V0 V2 V1

v[31] = Vertex(1.0f,0.0f,1.0f,0.0f,-1.0f,0.0f,1.0f,2.0f/3.0f);

v[32] = Vertex(-1.0f,0.0f,1.0f,0.0f,-1.0f,0.0f,0.5f,2.0f/3.0f);

v[33] = Vertex(-1.0f,0.0f,-1.0f,0.0f,-1.0f,0.0f,0.f,1.0f);//V0 V3 V2

v[34] = Vertex(1.0f,0.0f,-1.0f,0.0f,-1.0f,0.0f,1.0f,1.0f);

v[35] = Vertex(1.0f,0.0f,1.0f,0.0f,-1.0f,0.0f,1.0f,2.0f/3.0f);

VB->Unlock();

D3DXCreateTextureFromFile(

Device,

"2.jpg",

&Tex);

Device->SetRenderState(D3DRS\_LIGHTING,false);

D3DXVECTOR3 position(0.0f, 1.0f, -5.0f);

D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);

D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);

D3DXMATRIX V;

D3DXMatrixLookAtLH(&V, &position, &target, &up);

Device -> SetTransform(D3DTS\_VIEW, &V);

D3DXMATRIX proj;

D3DXMatrixPerspectiveFovLH(

&proj,

D3DX\_PI \* 0.5f,

(float)Width / (float)Height,

1.0f,

1000.0f);

Device -> SetTransform(D3DTS\_PROJECTION, &proj);

return true;

1. Display()函数：

Device->Clear(0, 0, D3DCLEAR\_TARGET | D3DCLEAR\_ZBUFFER, 0xffffffff, 1.0f, 0);

D3DXMATRIX Ry,Rx;

static float y = 0.0f;

D3DXMatrixRotationY(&Ry,y);

y+=timeDelta;

D3DXMatrixRotationX(&Rx,3.14f/4.0f);

D3DXMATRIX p = Rx \* Ry;

Device->SetTransform(D3DTS\_WORLD,&p);

Device->BeginScene();

Device->SetStreamSource(0,VB,0,sizeof(Vertex));

Device->SetFVF(Vertex::FVF);

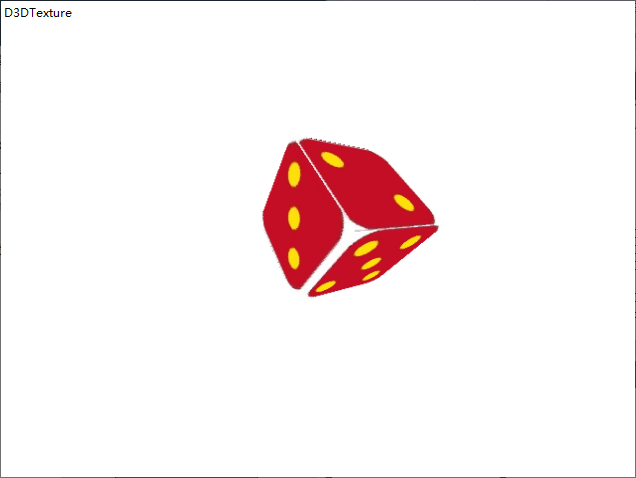
Device->SetTexture(0, Tex);

Device->DrawPrimitive(D3DPT\_TRIANGLELIST,0,12);

Device->EndScene();

Device->Present(0, 0, 0, 0);

1. 程序运行结果



1. **总结**