**武汉纺织大学**

**《Direct3D图形编程》上机实验报告**

**题目:** **模板（2）**

**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26日**

1. **实验1**
2. 题目

绘制一个正方形，用ice.bmp对正方形做纹理映射；在正方形前绘制一个黄色茶壶并绘制茶壶在正方形中的镜像；通过模板缓存控制茶壶镜像只出现在镜子区域内；通过VK\_LEFT，VK\_RIGHT，VK\_UP和VK\_DOWN按键来调整茶壶在X轴和Y轴的位置，具体如下：

[1] VK\_LEFT：x轴上的值减少；

[2] VK\_RIGHT：x轴上的值增加；

[3] VK\_UP：y轴上的值增加；

[4] VK\_DOWN：y轴上的值减少。

1. 实现代码

在d3dInit程序框架上修改代码：

1. 全局变量：

IDirect3DDevice9\* Device = 0;

IDirect3DVertexBuffer9 \* VB=0;

IDirect3DTexture9 \* Tex=0;

D3DMATERIAL9 mtrl;

ID3DXMesh\* Teapot=0;

D3DMATERIAL9 TeapotMtrl;

D3DXVECTOR3 TeapotPos(0.0f, 0.0f, -5.0f);

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(

6 \* sizeof(Vertex),

D3DUSAGE\_WRITEONLY,

Vertex::FVF,

D3DPOOL\_MANAGED,

&VB,

0);

Vertex\* v;

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

v[0] = Vertex(-10.0f,-10.0f,0.0f,0.0f,0.0f,-1.0f,0.0f,1.0f);

v[1] = Vertex(-10.0f,10.0f,0.0f,0.0f,0.0f,-1.0f,0.0f,0.0f);

v[2] = Vertex(10.0f,10.0f,0.0f,0.0f,0.0f,-1.0f,1.0f,0.0f);

v[3] = Vertex(-10.0f,-10.0f,0.0f,0.0f,0.0f,-1.0f,0.0f,1.0f);

v[4] = Vertex(10.0f,10.0f,0.0f,0.0f,0.0f,-1.0f,1.0f,0.0f);

v[5] = Vertex(10.0f,-10.0f,0.0f,0.0f,0.0f,-1.0f,1.0f,1.0f);

VB->Unlock();

mtrl = d3d::WHITE\_MTRL;

TeapotMtrl = d3d::YELLOW\_MTRL;

D3DXCreateTeapot(Device,&Teapot,0);

D3DLIGHT9 dir;

::ZeroMemory(&dir, sizeof(dir));

dir.Type = D3DLIGHT\_DIRECTIONAL;

dir.Diffuse = d3d::WHITE;

dir.Specular = d3d::WHITE \* 0.2f;

dir.Ambient = d3d::WHITE \* 0.6f;

dir.Direction = D3DXVECTOR3(0.707f, 0.0f, 0.707f);

Device->SetLight(0, &dir);

Device->LightEnable(0, true);

Device->SetRenderState(D3DRS\_NORMALIZENORMALS, true);

D3DXCreateTextureFromFile(

Device,

"ice.bmp",

&Tex);

Device->SetSamplerState(0,D3DSAMP\_MAGFILTER,D3DTEXF\_LINEAR);

Device->SetSamplerState(0,D3DSAMP\_MINFILTER,D3DTEXF\_LINEAR);

Device->SetSamplerState(0,D3DSAMP\_MIPFILTER,D3DTEXF\_POINT);

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);

d3d::Release<ID3DXMesh\*>(Teapot);

1. Display()函数：

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

Device->BeginScene();

Device->SetRenderState(D3DRS\_STENCILENABLE,true);

Device->SetRenderState(D3DRS\_STENCILFUNC,D3DCMP\_ALWAYS);

Device->SetRenderState(D3DRS\_STENCILREF,0x1);

Device->SetRenderState(D3DRS\_STENCILZFAIL,D3DSTENCILOP\_KEEP);

Device->SetRenderState(D3DRS\_STENCILFAIL,D3DSTENCILOP\_KEEP);

Device->SetRenderState(D3DRS\_STENCILPASS,D3DSTENCILOP\_REPLACE);

D3DXMATRIX W;

D3DXMatrixTranslation(&W,0.0f,0.0f,0.0f);

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

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

Device->SetFVF(Vertex::FVF);

Device->SetMaterial(&mtrl);

Device->SetTexture(0, tex);

Device->DrawPrimitive(D3DPT\_TRIANGLELIST,0,2);

Device->SetRenderState(D3DRS\_STENCILFUNC,D3DCMP\_EQUAL);

Device->SetRenderState(D3DRS\_STENCILPASS,D3DSTENCILOP\_KEEP);

D3DXMatrixTranslation(&W,TeapotPos.x,TeapotPos.y,TeapotPos.z);

D3DXMATRIX R,p;

D3DXPLANE plane(0.0f,0.0f,1.0f,0.0f);

D3DXMatrixReflect(&R,&plane);

p = W \* R;

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

Device->SetMaterial(&TeapotMtrl);

Device->SetTexture(0, 0);

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

Device->SetRenderState(D3DRS\_ALPHABLENDENABLE,true);

Device->SetRenderState(D3DRS\_SRCBLEND,D3DBLEND\_DESTCOLOR);

Device->SetRenderState(D3DRS\_DESTBLEND,D3DBLEND\_ZERO);

Device->SetRenderState(D3DRS\_CULLMODE,D3DCULL\_CW);

Teapot->DrawSubset(0);

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

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

Device->SetRenderState(D3DRS\_CULLMODE,D3DCULL\_CCW);

D3DXMatrixTranslation(&W,TeapotPos.x,TeapotPos.y,TeapotPos.z);

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

Device->SetMaterial(&TeapotMtrl);

Device->SetTexture(0, 0);

Teapot->DrawSubset(0);

Device->EndScene();

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

1. WndProc()函数：

case WM\_KEYDOWN:

if( wParam == VK\_ESCAPE )

::DestroyWindow(hwnd);

if ( wParam == VK\_LEFT )

TeapotPos.x -= 0.1f;

if ( wParam == VK\_RIGHT )

TeapotPos.x += 0.1f;

if( wParam == VK\_UP )

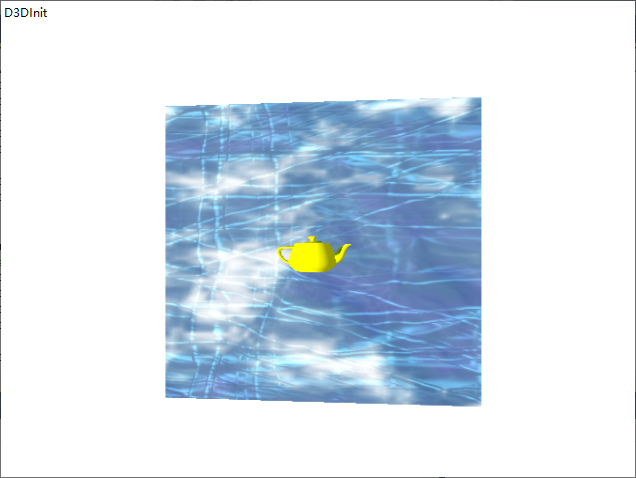
TeapotPos.y += 0.1f;

if( wParam == VK\_DOWN )

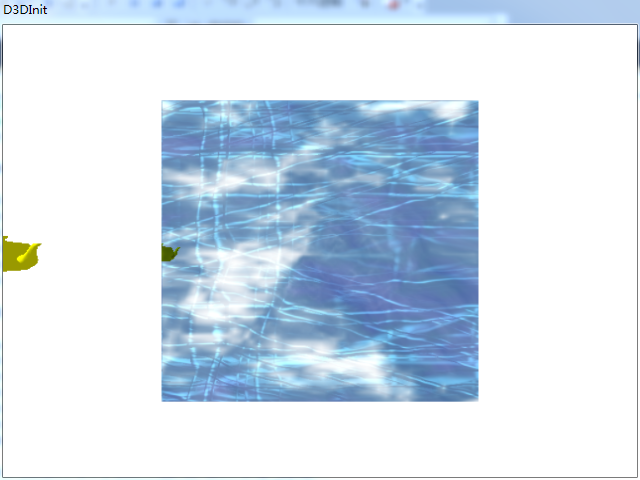
TeapotPos.y -= 0.1f;

break;

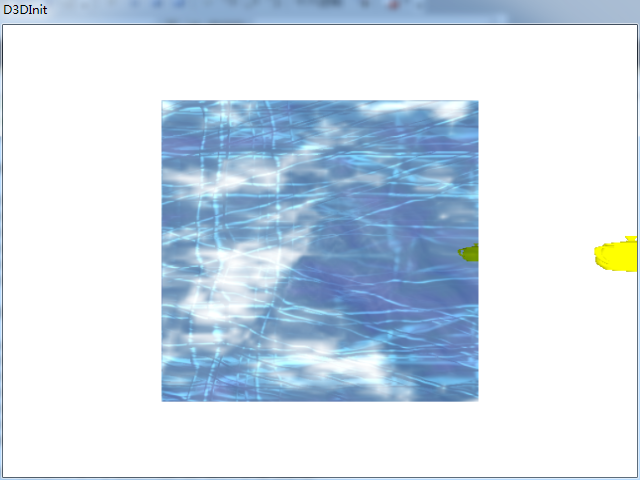
1. 程序运行结果
   1. 初始画面



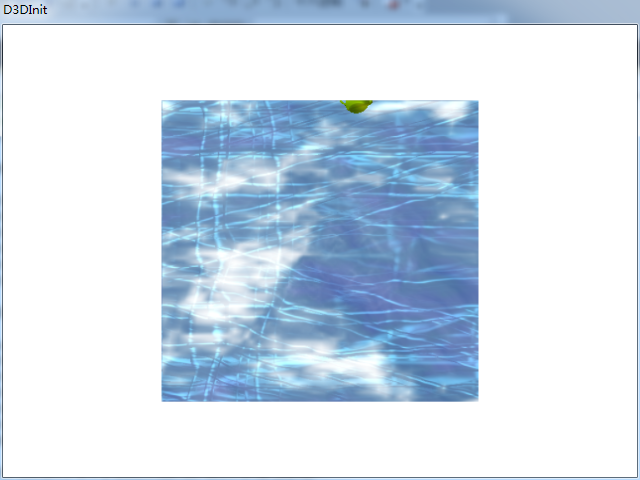
* 1. 按下VK\_LEFT后的画面



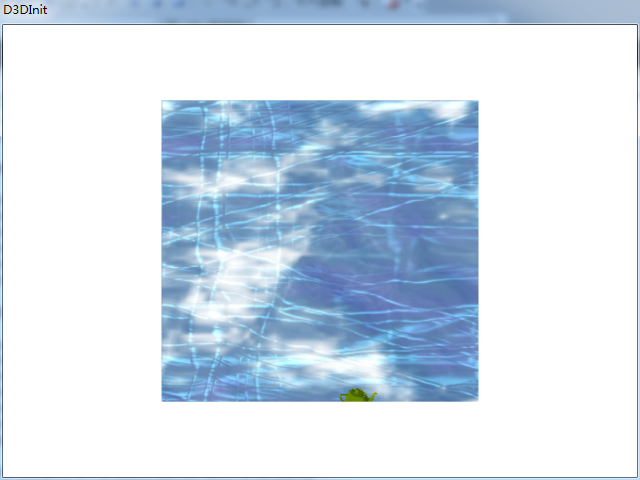
* 1. 按下VK\_RIGHT后的画面



* 1. 按下VK\_UP后的画面



* 1. 按下VK\_DOWN后的画面



1. **总结**
2. 问题：看不到茶壶镜像。

解决方法：开启模板的代码要放在绘制镜子前，且模板的比较代码要放在绘制镜子后、绘制镜像前。

1. **实验2**
2. 题目

绘制一个正方体，用red.jpg，green.jpg和blue.jpg三个纹理文件对正方体的6个面做纹理映射，具体为：前面/后面（red）, 左面/右面（green）, 上面/下面（blue）;并且使得正方体绕Y轴连续旋转。

1. 实现代码

在d3dInit程序框架上修改代码：

1. 全局变量：

IDirect3DDevice9\* Device = 0;

IDirect3DVertexBuffer9\* VB = 0;

IDirect3DTexture9\* tex1 = 0;

IDirect3DTexture9\* tex2 = 0;

IDirect3DTexture9\* tex3 = 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(

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);//前面 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,1.0f,0.0f);

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

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

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

v[6] = Vertex(-1.0f,0.0f,1.0f,0.0f,0.0f,1.0f,0.0f,1.0f);//背面 V1 V6 V5

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

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

v[9] = Vertex(-1.0f,0.0f,1.0f,0.0f,0.0f,1.0f,0.0f,1.0f);//V1 V2 V6

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

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

v[12] = Vertex(1.0f,0.0f,-1.0f,1.0f,0.0f,0.0f,0.0f,1.0f);//右面 V3 V7 V6

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

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

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

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

v[17] = Vertex(1.0f,0.0f,1.0f,1.0f,0.0f,0.0f,1.0f,1.0f);

v[18] = Vertex(-1.0f,0.0f,-1.0f,-1.0f,0.0f,0.0f,0.0f,1.0f);//左面 V0 V5 V4

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

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

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

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

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

v[24] = Vertex(-1.0f,2.0f,-1.0f,0.0f,1.0f,0.0f,0.0f,1.0f);//顶面V4 V5 V6

v[25] = Vertex(-1.0f,2.0f,1.0f,0.0f,1.0f,0.0f,0.0f,0.0f);

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

v[27] = Vertex(-1.0f,2.0f,-1.0f,0.0f,1.0f,0.0f,0.0f,1.0f);//V4 V6 V7

v[28] = Vertex(1.0f,2.0f,1.0f,0.0f,1.0f,0.0f,1.0f,0.0f);

v[29] = Vertex(1.0f,2.0f, -1.0f,0.0f,1.0f,0.0f,1.0f,1.0f);

v[30] = Vertex(-1.0f,0.0f,-1.0f,0.0f,-1.0f,0.0f,0.0f,1.0f);//底面 V0 V2 V1

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

v[32] = Vertex(-1.0f,0.0f,1.0f,0.0f,-1.0f,0.0f,0.0f,0.0f);

v[33] = Vertex(-1.0f,0.0f,-1.0f,0.0f,-1.0f,0.0f,0.0f,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,0.0f);

VB->Unlock();

Device->SetRenderState(D3DRS\_NORMALIZENORMALS, true);

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

D3DXCreateTextureFromFile(

Device,

"red.jpg",

&tex1);

D3DXCreateTextureFromFile(

Device,

"green.jpg",

&tex2);

D3DXCreateTextureFromFile(

Device,

"blue.jpg",

&tex3);

Device->SetSamplerState(0, D3DSAMP\_MAGFILTER, D3DTEXF\_LINEAR);

Device->SetSamplerState(0, D3DSAMP\_MINFILTER, D3DTEXF\_LINEAR);

Device->SetSamplerState(0, D3DSAMP\_MIPFILTER, D3DTEXF\_LINEAR);

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

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

return true;

1. Cleanup()函数：

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

d3d::Release<IDirect3DTexture9\*>(tex1);

d3d::Release<IDirect3DTexture9\*>(tex2);

d3d::Release<IDirect3DTexture9\*>(tex3);

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, tex1);

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

Device->SetTexture(0, tex2);

Device->DrawPrimitive(D3DPT\_TRIANGLELIST,12,4);

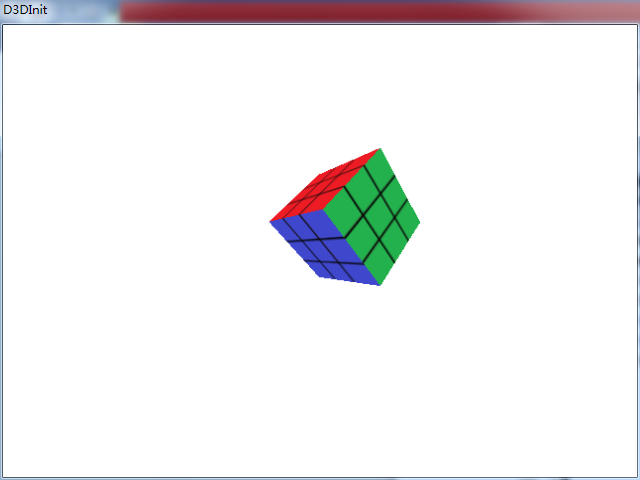
Device->SetTexture(0, tex3);

Device->DrawPrimitive(D3DPT\_TRIANGLELIST,24,4);

Device->EndScene();

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

1. 程序运行结果



1. **总结**

纹理贴图的顺序要严格按照顶点缓存的顺序。