國立高雄第一科技大學

工科博士班

**電玩物理學**

HW4. Directx 繪製正方體並旋轉

指導老師：黃勤鎰

班 級：電子4A

學生姓名：劉峻瑜

學 號：u0252038

目錄:

(一)題目要求 3

(二)流程圖 3

(三)程式碼 4

(四)成果 11

(一)題目要求:

**第一小題**

1.使用Directx。

2.繪製3D正方體。

3.其表面貼上不同圖形(使用六張texture)

4.優先對y軸順時針旋轉2轉/分,轉1圈後,在對z軸逆時針旋轉4轉/分,轉2圈後,最後再對x軸順時針旋轉8轉/分,轉4圈,重複旋轉。

**第二小題**

1.使用Directx。

2.繪製3D正方體。

3.其表面貼上不同圖形(使用一張texture)

4.優先對y軸順時針旋轉2轉/分,轉1圈後,在對z軸逆時針旋轉4轉/分,轉2圈後,最後再對x軸順時針旋轉8轉/分,轉4圈,重複旋轉。

(二)流程圖:



(三)程式碼:

程式碼僅貼出與範例程式碼(圓柱體貼圖)不同的地方，並未全部貼上

第一小題

#include <Windows.h>

#include <mmsystem.h>

#include <d3dx9.h>

#pragma warning( disable : 4996 ) // disable deprecated warning

#include <strsafe.h>

#pragma warning( default : 4996 )

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

//-----------------------------------------------------------------------------

// Global variables

//-----------------------------------------------------------------------------

LPDIRECT3D9 g\_pD3D = NULL; // Used to create the D3DDevice

LPDIRECT3DDEVICE9 g\_pd3dDevice = NULL; // Our rendering device

LPDIRECT3DVERTEXBUFFER9 g\_pVB = NULL; // Buffer to hold vertices

/\* 前 後 下 \*/

LPDIRECT3DTEXTURE9 g\_pTexture, g\_pTexture2 , g\_pTexture3, g\_pTexture4, g\_pTexture5, g\_pTexture6= NULL; // Our texture

// A structure for our custom vertex type. We added texture coordinates

struct CUSTOMVERTEX

{

D3DXVECTOR3 position; // The position

D3DCOLOR color; // The color

#ifndef SHOW\_HOW\_TO\_USE\_TCI

FLOAT tu, tv; // The texture coordinates

#endif

};

// Our custom FVF, which describes our custom vertex structure

#ifdef SHOW\_HOW\_TO\_USE\_TCI

#define D3DFVF\_CUSTOMVERTEX (D3DFVF\_XYZ|D3DFVF\_DIFFUSE)

#else

#define D3DFVF\_CUSTOMVERTEX (D3DFVF\_XYZ|D3DFVF\_DIFFUSE|D3DFVF\_TEX1)

#endif

//-----------------------------------------------------------------------------

// Name: InitGeometry()

// Desc: Create the Textures and vertex buffers

//-----------------------------------------------------------------------------

HRESULT InitGeometry()

{

// Use D3DX to create a texture from a file based image

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"banana.bmp", &g\_pTexture2)))

{

// If texture is not in current folder, try parent folder

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"..\\banana.bmp", &g\_pTexture2)))

{

MessageBox(NULL, L"Could not find banana.bmp", L"Textures.exe", MB\_OK);

return E\_FAIL;

}

}

// Use D3DX to create a texture from a file based image

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"banana2.bmp", &g\_pTexture)))

{

// If texture is not in current folder, try parent folder

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"..\\banana2.bmp", &g\_pTexture)))

{

MessageBox(NULL, L"Could not find banana.bmp", L"Textures.exe", MB\_OK);

return E\_FAIL;

}

}

// Use D3DX to create a texture from a file based image

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"banana3.bmp", &g\_pTexture3)))

{

// If texture is not in current folder, try parent folder

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"..\\banana3.bmp", &g\_pTexture3)))

{

MessageBox(NULL, L"Could not find banana.bmp", L"Textures.exe", MB\_OK);

return E\_FAIL;

}

}

// Use D3DX to create a texture from a file based image

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"banana4.bmp", &g\_pTexture4)))

{

// If texture is not in current folder, try parent folder

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"..\\banana4.bmp", &g\_pTexture4)))

{

MessageBox(NULL, L"Could not find banana.bmp", L"Textures.exe", MB\_OK);

return E\_FAIL;

}

}

// Use D3DX to create a texture from a file based image

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"banana5.bmp", &g\_pTexture5)))

{

// If texture is not in current folder, try parent folder

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"..\\banana5.bmp", &g\_pTexture5)))

{

MessageBox(NULL, L"Could not find banana.bmp", L"Textures.exe", MB\_OK);

return E\_FAIL;

}

}

// Use D3DX to create a texture from a file based image

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"banana6.bmp", &g\_pTexture6)))

{

// If texture is not in current folder, try parent folder

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"..\\banana6.bmp", &g\_pTexture6)))

{

MessageBox(NULL, L"Could not find banana.bmp", L"Textures.exe", MB\_OK);

return E\_FAIL;

}

}

// Create the vertex buffer.

if( FAILED( g\_pd3dDevice->CreateVertexBuffer( 50 \* 12 \* sizeof( CUSTOMVERTEX ),

0, D3DFVF\_CUSTOMVERTEX,

D3DPOOL\_DEFAULT, &g\_pVB, NULL ) ) )

{

return E\_FAIL;

}

// Fill the vertex buffer. We are setting the tu and tv texture

// coordinates, which range from 0.0 to 1.0

CUSTOMVERTEX\* pVertices;

if( FAILED( g\_pVB->Lock( 0, 0, ( void\*\* )&pVertices, 0 ) ) )

return E\_FAIL;

for( DWORD i = 0; i < 50; i++ )//前

{

pVertices[2 \* i + 0].position = D3DXVECTOR3(-1.0f+((2.0f/50)\*i), 1.0f, 1.0f);

pVertices[2 \* i + 0].color = 0xffffffff;

pVertices[2 \* i + 0].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 0].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 0].tv = 1.0f;

#endif

pVertices[2 \* i + 1].position = D3DXVECTOR3(-1.0f+((2.0f / 50)\*i), -1.0f, 1.0f);

pVertices[2 \* i + 1].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 1].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 1].tv = 0.0f;

#endif

}

for (DWORD i = 0; i < 50; i++)//後

{

pVertices[2 \* i + 100].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), 1.0f, -1.0f);

pVertices[2 \* i + 100].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 100].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 100].tv = 1.0f;

#endif

pVertices[2 \* i + 101].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), -1.0f, -1.0f);

pVertices[2 \* i + 101].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 101].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 101].tv = 0.0f;

#endif

}

for (DWORD i = 0; i < 50; i++)//下

{

pVertices[2 \* i + 200].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), -1.0f, 1.0f);

pVertices[2 \* i + 200].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 200].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 200].tv = 1.0f;

#endif

pVertices[2 \* i + 201].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i),-1.0f, -1.0f);

pVertices[2 \* i + 201].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 201].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 201].tv = 0.0f;

#endif

}

for (DWORD i = 0; i < 50; i++)//上

{

pVertices[2 \* i + 300].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), 1.0f, 1.0f);

pVertices[2 \* i + 300].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 300].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 300].tv = 1.0f;

#endif

pVertices[2 \* i + 301].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), 1.0f, -1.0f);

pVertices[2 \* i + 301].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 301].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 301].tv = 0.0f;

#endif

}

for (DWORD i = 0; i < 50; i++)//右

{

pVertices[2 \* i + 400].position = D3DXVECTOR3(-1.0f, -1.0f + ((2.0f / 50)\*i), 1.0f);

pVertices[2 \* i + 400].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 400].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 400].tv = 1.0f;

#endif

pVertices[2 \* i + 401].position = D3DXVECTOR3(-1.0f, -1.0f + ((2.0f / 50)\*i), -1.0f);

pVertices[2 \* i + 401].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 401].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 401].tv = 0.0f;

#endif

}

for (DWORD i = 0; i < 50; i++)//右

{

pVertices[2 \* i + 500].position = D3DXVECTOR3(1.0f, -1.0f + ((2.0f / 50)\*i), 1.0f);

pVertices[2 \* i + 500].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 500].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 500].tv = 1.0f;

#endif

pVertices[2 \* i + 501].position = D3DXVECTOR3(1.0f, -1.0f + ((2.0f / 50)\*i), -1.0f);

pVertices[2 \* i + 501].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 501].tu = ((FLOAT)i) / (50 - 1);

pVertices[2 \* i + 501].tv = 0.0f;

#endif

}

g\_pVB->Unlock();

return S\_OK;

}

//-----------------------------------------------------------------------------

// Name: Cleanup()

// Desc: Releases all previously initialized objects

//-----------------------------------------------------------------------------

VOID Cleanup()

{

if (g\_pTexture6 != NULL)

g\_pTexture6->Release();

if (g\_pTexture5 != NULL)

g\_pTexture5->Release();

if (g\_pTexture4 != NULL)

g\_pTexture4->Release();

if( g\_pTexture3 != NULL )

g\_pTexture3->Release();

if (g\_pTexture2 != NULL)

g\_pTexture2->Release();

if (g\_pTexture != NULL)

g\_pTexture->Release();

if( g\_pVB != NULL )

g\_pVB->Release();

if( g\_pd3dDevice != NULL )

g\_pd3dDevice->Release();

if( g\_pD3D != NULL )

g\_pD3D->Release();

}

//-----------------------------------------------------------------------------

// Name: SetupMatrices()

// Desc: Sets up the world, view, and projection transform matrices.

//-----------------------------------------------------------------------------

VOID SetupMatrices()

{ // D3DXMatrixRotationZ( &matWorld, timeGetTime() / 1000.0f );

//D3DXMatrixRotationY(&matWorld, fAngle);

// Set up world matrix

D3DXMATRIXA16 matWorld;

int i = 0;

D3DXMATRIXA16 RotationX;

D3DXMATRIXA16 RotationY;

D3DXMATRIXA16 RotationZ;

static float fps = 0; //我们需要计算的FPS值

static int frameCount = 0;//帧数

static float currentTime = 0.0f;//当前时间

static float lastTime = 0.0f;//持续时间

static int control = 0;//帧数

DWORD dwBegin, dwEnd;

dwBegin = timeGetTime();

UINT iTime = (timeGetTime()% 60000); //Msecond 不精準

FLOAT fAngleY = (iTime) \* ( 2.0f \* D3DX\_PI ) / 30000.0f;// 2 \*pi

FLOAT fAngleZ = (iTime) \* ( 2.0f \* D3DX\_PI) / 15000.0f;// 2 \*pi

FLOAT fAngleX = (iTime) \* ( 2.0f \* D3DX\_PI) / 7500.0f;// 2 \*pi

Sleep(100);

frameCount++;//每调用一次Get\_FPS()函数，帧数自增1

currentTime = timeGetTime()\*0.001f;//获取系统时间，其中timeGetTime函数返回的是以毫秒为单位的系统时间，所以需要乘以0.001，得到单位为秒的时间

//如果当前时间减去持续时间大于了1秒钟，就进行一次FPS的计算和持续时间的更新，并将帧数值清零

if (currentTime - lastTime > 1.0f) //将时间控制在1秒钟

{

control++;

lastTime = currentTime; //将当前时间currentTime赋给持续时间lastTime，作为下一秒的基准时间

frameCount = 0;//将本次帧数frameCount值清零

}

if (control <30)

{

D3DXMatrixRotationY(&matWorld, fAngleY);

}

else if(control <60)

{

D3DXMatrixRotationZ(&matWorld, fAngleZ);

}

else if (control < 90)

{

D3DXMatrixRotationX(&matWorld, fAngleX);

}

else

{

D3DXMatrixRotationX(&matWorld, fAngleX);

control = 0;

}

g\_pd3dDevice->SetTransform( D3DTS\_WORLD, &matWorld );

// Set up our view matrix. A view matrix can be defined given an eye point,

// a point to lookat, and a direction for which way is up. Here, we set the

// eye five units back along the z-axis and up three units, look at the

// origin, and define "up" to be in the y-direction.

D3DXVECTOR3 vEyePt( 0.0f, 3.0f,-5.0f );

D3DXVECTOR3 vLookatPt( 0.0f, 0.0f, 0.0f );

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

D3DXMATRIXA16 matView;

D3DXMatrixLookAtLH( &matView, &vEyePt, &vLookatPt, &vUpVec );

g\_pd3dDevice->SetTransform( D3DTS\_VIEW, &matView );

// For the projection matrix, we set up a perspective transform (which

// transforms geometry from 3D view space to 2D viewport space, with

// a perspective divide making objects smaller in the distance). To build

// a perpsective transform, we need the field of view (1/4 pi is common),

// the aspect ratio, and the near and far clipping planes (which define at

// what distances geometry should be no longer be rendered).

D3DXMATRIXA16 matProj;

D3DXMatrixPerspectiveFovLH( &matProj, D3DX\_PI / 4, 1.0f, 1.0f, 100.0f );

g\_pd3dDevice->SetTransform( D3DTS\_PROJECTION, &matProj );

}

//-----------------------------------------------------------------------------

// Name: Render()

// Desc: Draws the scene

//-----------------------------------------------------------------------------

VOID Render()

{

// Clear the backbuffer and the zbuffer

g\_pd3dDevice->Clear( 0, NULL, D3DCLEAR\_TARGET | D3DCLEAR\_ZBUFFER,

D3DCOLOR\_XRGB( 0, 0, 255 ), 1.0f, 0 );

// Begin the scene

if( SUCCEEDED( g\_pd3dDevice->BeginScene() ) )

{

// Setup the world, view, and projection matrices

SetupMatrices();

// Setup our texture. Using Textures introduces the texture stage states,

// which govern how Textures get blended together (in the case of multiple

// Textures) and lighting information. In this case, we are modulating

// (blending) our texture with the diffuse color of the vertices.

#ifdef SHOW\_HOW\_TO\_USE\_TCI

// Note: to use D3D texture coordinate generation, use the stage state

// D3DTSS\_TEXCOORDINDEX, as shown below. In this example, we are using

// the position of the vertex in camera space (D3DTSS\_TCI\_CAMERASPACEPOSITION)

// to generate texture coordinates. Camera space is the vertex position

// multiplied by the World and View matrices. The tex coord index (TCI)

// parameters are passed into a texture transform, which is a 4x4 matrix

// which transforms the x,y,z TCI coordinates into tu, tv texture coordinates.

// In this example, the texture matrix is setup to transform the input

// camera space coordinates (all of R^3) to projection space (-1,+1)

// and finally to texture space (0,1).

// CameraSpace.xyzw = (input vertex position) \* (WorldView)

// ProjSpace.xyzw = CameraSpace.xyzw \* Projection //move to -1 to 1

// TexSpace.xyzw = ProjSpace.xyzw \* ( 0.5, -0.5, 1.0, 1.0 ) //scale to -0.5 to 0.5 (flip y)

// TexSpace.xyzw += ( 0.5, 0.5, 0.0, 0.0 ) //shift to 0 to 1

// Setting D3DTSS\_TEXTURETRANSFORMFLAGS to D3DTTFF\_COUNT4 | D3DTTFF\_PROJECTED

// tells D3D to divide the input texture coordinates by the 4th (w) component.

// This divide is necessary when performing a perspective projection since

// the TexSpace.xy coordinates prior to the homogeneous divide are not actually

// in the 0 to 1 range.

D3DXMATRIXA16 mTextureTransform;

D3DXMATRIXA16 mProj;

D3DXMATRIXA16 mTrans;

D3DXMATRIXA16 mScale;

g\_pd3dDevice->GetTransform( D3DTS\_PROJECTION, &mProj );

D3DXMatrixTranslation( &mTrans, 0.5f, 0.5f, 0.0f );

D3DXMatrixScaling( &mScale, 0.5f, -0.5f, 1.0f );

mTextureTransform = mProj \* mScale \* mTrans;

g\_pd3dDevice->SetTransform( D3DTS\_TEXTURE0, &mTextureTransform );

g\_pd3dDevice->SetTextureStageState( 0, D3DTSS\_TEXTURETRANSFORMFLAGS, D3DTTFF\_COUNT4 | D3DTTFF\_PROJECTED );

g\_pd3dDevice->SetTextureStageState( 0, D3DTSS\_TEXCOORDINDEX, D3DTSS\_TCI\_CAMERASPACEPOSITION );

#endif

// Render the vertex buffer contents

g\_pd3dDevice->SetStreamSource( 0, g\_pVB, 0, sizeof( CUSTOMVERTEX ) );

g\_pd3dDevice->SetFVF( D3DFVF\_CUSTOMVERTEX );

g\_pd3dDevice->SetTexture(0, g\_pTexture); //前

g\_pd3dDevice->DrawPrimitive( D3DPT\_TRIANGLESTRIP, 0, 98);

g\_pd3dDevice->SetTexture(0, g\_pTexture2);//後

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 100, 98);

g\_pd3dDevice->SetTexture(0, g\_pTexture3);//左

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 200, 98);

g\_pd3dDevice->SetTexture(0, g\_pTexture4);//右

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 300, 98);

g\_pd3dDevice->SetTexture(0, g\_pTexture5);//上

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 400, 98);

g\_pd3dDevice->SetTexture(0, g\_pTexture6);//下

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 500, 98);

// End the scene

g\_pd3dDevice->EndScene();

}

// Present the backbuffer contents to the display

g\_pd3dDevice->Present( NULL, NULL, NULL, NULL );

}

第二小題

#include <Windows.h>

#include <mmsystem.h>

#include <d3dx9.h>

#pragma warning( disable : 4996 ) // disable deprecated warning

#include <strsafe.h>

#pragma warning( default : 4996 )

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

//-----------------------------------------------------------------------------

// Global variables

//-----------------------------------------------------------------------------

LPDIRECT3D9 g\_pD3D = NULL; // Used to create the D3DDevice

LPDIRECT3DDEVICE9 g\_pd3dDevice = NULL; // Our rendering device

LPDIRECT3DVERTEXBUFFER9 g\_pVB = NULL; // Buffer to hold vertices

/\* 前 後 下 \*/

LPDIRECT3DTEXTURE9 g\_pTexture= NULL; // Our texture

// A structure for our custom vertex type. We added texture coordinates

struct CUSTOMVERTEX

{

D3DXVECTOR3 position; // The position

D3DCOLOR color; // The color

#ifndef SHOW\_HOW\_TO\_USE\_TCI

FLOAT tu, tv; // The texture coordinates

#endif

};

// Our custom FVF, which describes our custom vertex structure

#ifdef SHOW\_HOW\_TO\_USE\_TCI

#define D3DFVF\_CUSTOMVERTEX (D3DFVF\_XYZ|D3DFVF\_DIFFUSE)

#else

#define D3DFVF\_CUSTOMVERTEX (D3DFVF\_XYZ|D3DFVF\_DIFFUSE|D3DFVF\_TEX1)

#endif

HRESULT InitGeometry()

{

// Use D3DX to create a texture from a file based image

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"allin.png", &g\_pTexture)))

{

// If texture is not in current folder, try parent folder

if (FAILED(D3DXCreateTextureFromFile(g\_pd3dDevice, L"..\\allin.png", &g\_pTexture)))

{

MessageBox(NULL, L"Could not find banana.bmp", L"Textures.exe", MB\_OK);

return E\_FAIL;

}

}

// Create the vertex buffer.

if( FAILED( g\_pd3dDevice->CreateVertexBuffer( 50 \* 12 \* sizeof( CUSTOMVERTEX ),

0, D3DFVF\_CUSTOMVERTEX,

D3DPOOL\_DEFAULT, &g\_pVB, NULL ) ) )

{

return E\_FAIL;

}

// Fill the vertex buffer. We are setting the tu and tv texture

// coordinates, which range from 0.0 to 1.0

CUSTOMVERTEX\* pVertices;

if( FAILED( g\_pVB->Lock( 0, 0, ( void\*\* )&pVertices, 0 ) ) )

return E\_FAIL;

for( DWORD i = 0; i < 50; i++ )//前

{

pVertices[2 \* i + 0].position = D3DXVECTOR3(-1.0f+((2.0f/50)\*i), 1.0f, 1.0f);

pVertices[2 \* i + 0].color = 0xffffffff;

pVertices[2 \* i + 0].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 0].tu = ((((FLOAT)i) / (150 - 1)));

pVertices[2 \* i + 0].tv = 0.5f;

#endif

pVertices[2 \* i + 1].position = D3DXVECTOR3(-1.0f+((2.0f / 50)\*i), -1.0f, 1.0f);

pVertices[2 \* i + 1].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 1].tu = ((((FLOAT)i) / (150 - 1)));

pVertices[2 \* i + 1].tv = 0.0f;

#endif

}

for (DWORD i = 0; i < 50; i++)//後

{

pVertices[2 \* i + 100].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), 1.0f, -1.0f);

pVertices[2 \* i + 100].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 100].tu = ((((FLOAT)i) / (150 - 1))) + 0.33;

pVertices[2 \* i + 100].tv = 0.5f;

#endif

pVertices[2 \* i + 101].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), -1.0f, -1.0f);

pVertices[2 \* i + 101].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 101].tu = ((((FLOAT)i) / (150 - 1))) + 0.33;

pVertices[2 \* i + 101].tv = 0.0f;

#endif

}

for (DWORD i = 0; i < 50; i++)//下

{

pVertices[2 \* i + 200].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), -1.0f, 1.0f);

pVertices[2 \* i + 200].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 200].tu = ((((FLOAT)i) / (150 - 1))) + 0.66;

pVertices[2 \* i + 200].tv = 0.5f;

#endif

pVertices[2 \* i + 201].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), -1.0f, -1.0f);

pVertices[2 \* i + 201].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 201].tu = ((((FLOAT)i) / (150 - 1))) + 0.66;

pVertices[2 \* i + 201].tv = 0.0f;

#endif

}

for (DWORD i = 0; i < 50; i++)//上

{

pVertices[2 \* i + 300].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), 1.0f, 1.0f);

pVertices[2 \* i + 300].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 300].tu = ((((FLOAT)i) / (150 - 1)));

pVertices[2 \* i + 300].tv = 1.0f;

#endif

pVertices[2 \* i + 301].position = D3DXVECTOR3(-1.0f + ((2.0f / 50)\*i), 1.0f, -1.0f);

pVertices[2 \* i + 301].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 301].tu = ((((FLOAT)i) / (150 - 1)));

pVertices[2 \* i + 301].tv = 0.5f;

#endif

}

for (DWORD i = 0; i < 50; i++)//右

{

pVertices[2 \* i + 400].position = D3DXVECTOR3(-1.0f, -1.0f + ((2.0f / 50)\*i), 1.0f);

pVertices[2 \* i + 400].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 400].tu = ((((FLOAT)i) / (150 - 1))) + 0.33;

pVertices[2 \* i + 400].tv = 1.0f;

#endif

pVertices[2 \* i + 401].position = D3DXVECTOR3(-1.0f, -1.0f + ((2.0f / 50)\*i), -1.0f);

pVertices[2 \* i + 401].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 401].tu = ((((FLOAT)i) / (150 - 1))) + 0.33;

pVertices[2 \* i + 401].tv = 0.5f;

#endif

}

for (DWORD i = 0; i < 50; i++)//左

{

pVertices[2 \* i + 500].position = D3DXVECTOR3(1.0f, -1.0f + ((2.0f / 50)\*i), 1.0f);

pVertices[2 \* i + 500].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 500].tu = ((((FLOAT)i) / (150 - 1))) + 0.66;

pVertices[2 \* i + 500].tv = 1.0f;

#endif

pVertices[2 \* i + 501].position = D3DXVECTOR3(1.0f, -1.0f + ((2.0f / 50)\*i), -1.0f);

pVertices[2 \* i + 501].color = 0xffffffff;

#ifndef SHOW\_HOW\_TO\_USE\_TCI

pVertices[2 \* i + 501].tu = ((((FLOAT)i) / (150 - 1))) + 0.66;

pVertices[2 \* i + 501].tv = 0.5f;

#endif

}

g\_pVB->Unlock();

return S\_OK;

}

//-----------------------------------------------------------------------------

// Name: SetupMatrices()

// Desc: Sets up the world, view, and projection transform matrices.

//-----------------------------------------------------------------------------

VOID SetupMatrices()

{ // D3DXMatrixRotationZ( &matWorld, timeGetTime() / 1000.0f );

//D3DXMatrixRotationY(&matWorld, fAngle);

// Set up world matrix

D3DXMATRIXA16 matWorld;

int i = 0;

D3DXMATRIXA16 RotationX;

D3DXMATRIXA16 RotationY;

D3DXMATRIXA16 RotationZ;

static float fps = 0; //我们需要计算的FPS值

static int frameCount = 0;//帧数

static float currentTime = 0.0f;//当前时间

static float lastTime = 0.0f;//持续时间

static int control = 0;//帧数

DWORD dwBegin, dwEnd;

dwBegin = timeGetTime();

UINT iTime = (timeGetTime() % 60000); //Msecond 不精準

FLOAT fAngleY = (iTime) \* (2.0f \* D3DX\_PI) / 30000.0f;// 2 \*pi

FLOAT fAngleZ = (iTime) \* (2.0f \* D3DX\_PI) / 15000.0f;// 2 \*pi

FLOAT fAngleX = (iTime) \* (2.0f \* D3DX\_PI) / 7500.0f;// 2 \*pi

Sleep(100);

D3DXMATRIX Rx, Ry;

frameCount++;//每调用一次Get\_FPS()函数，帧数自增1

currentTime = timeGetTime()\*0.001f;//获取系统时间，其中timeGetTime函数返回的是以毫秒为单位的系统时间，所以需要乘以0.001，得到单位为秒的时间

//如果当前时间减去持续时间大于了1秒钟，就进行一次FPS的计算和持续时间的更新，并将帧数值清零

if (currentTime - lastTime > 1.0f) //将时间控制在1秒钟

{

control++;

lastTime = currentTime; //将当前时间currentTime赋给持续时间lastTime，作为下一秒的基准时间

frameCount = 0;//将本次帧数frameCount值清零

}

if (control <30)

{

D3DXMatrixRotationY(&matWorld, fAngleY);

}

else if (control <60)

{

D3DXMatrixRotationZ(&matWorld, fAngleZ);

}

else if (control < 90)

{

D3DXMatrixRotationX(&matWorld, fAngleX);

}

else

{

D3DXMatrixRotationX(&matWorld, fAngleX);

control = 0;

}

g\_pd3dDevice->SetTransform( D3DTS\_WORLD, &matWorld );

// Set up our view matrix. A view matrix can be defined given an eye point,

// a point to lookat, and a direction for which way is up. Here, we set the

// eye five units back along the z-axis and up three units, look at the

// origin, and define "up" to be in the y-direction.

D3DXVECTOR3 vEyePt( 0.0f, 3.0f,-5.0f );

D3DXVECTOR3 vLookatPt( 0.0f, 0.0f, 0.0f );

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

D3DXMATRIXA16 matView;

D3DXMatrixLookAtLH( &matView, &vEyePt, &vLookatPt, &vUpVec );

g\_pd3dDevice->SetTransform( D3DTS\_VIEW, &matView );

// For the projection matrix, we set up a perspective transform (which

// transforms geometry from 3D view space to 2D viewport space, with

// a perspective divide making objects smaller in the distance). To build

// a perpsective transform, we need the field of view (1/4 pi is common),

// the aspect ratio, and the near and far clipping planes (which define at

// what distances geometry should be no longer be rendered).

D3DXMATRIXA16 matProj;

D3DXMatrixPerspectiveFovLH( &matProj, D3DX\_PI / 4, 1.0f, 1.0f, 100.0f );

g\_pd3dDevice->SetTransform( D3DTS\_PROJECTION, &matProj );

}

//-----------------------------------------------------------------------------

// Name: Render()

// Desc: Draws the scene

//-----------------------------------------------------------------------------

VOID Render()

{

// Clear the backbuffer and the zbuffer

g\_pd3dDevice->Clear( 0, NULL, D3DCLEAR\_TARGET | D3DCLEAR\_ZBUFFER,

D3DCOLOR\_XRGB( 0, 0, 255 ), 1.0f, 0 );

// Begin the scene

if( SUCCEEDED( g\_pd3dDevice->BeginScene() ) )

{

// Setup the world, view, and projection matrices

SetupMatrices();

// Setup our texture. Using Textures introduces the texture stage states,

// which govern how Textures get blended together (in the case of multiple

// Textures) and lighting information. In this case, we are modulating

// (blending) our texture with the diffuse color of the vertices.

#ifdef SHOW\_HOW\_TO\_USE\_TCI

// Note: to use D3D texture coordinate generation, use the stage state

// D3DTSS\_TEXCOORDINDEX, as shown below. In this example, we are using

// the position of the vertex in camera space (D3DTSS\_TCI\_CAMERASPACEPOSITION)

// to generate texture coordinates. Camera space is the vertex position

// multiplied by the World and View matrices. The tex coord index (TCI)

// parameters are passed into a texture transform, which is a 4x4 matrix

// which transforms the x,y,z TCI coordinates into tu, tv texture coordinates.

// In this example, the texture matrix is setup to transform the input

// camera space coordinates (all of R^3) to projection space (-1,+1)

// and finally to texture space (0,1).

// CameraSpace.xyzw = (input vertex position) \* (WorldView)

// ProjSpace.xyzw = CameraSpace.xyzw \* Projection //move to -1 to 1

// TexSpace.xyzw = ProjSpace.xyzw \* ( 0.5, -0.5, 1.0, 1.0 ) //scale to -0.5 to 0.5 (flip y)

// TexSpace.xyzw += ( 0.5, 0.5, 0.0, 0.0 ) //shift to 0 to 1

// Setting D3DTSS\_TEXTURETRANSFORMFLAGS to D3DTTFF\_COUNT4 | D3DTTFF\_PROJECTED

// tells D3D to divide the input texture coordinates by the 4th (w) component.

// This divide is necessary when performing a perspective projection since

// the TexSpace.xy coordinates prior to the homogeneous divide are not actually

// in the 0 to 1 range.

D3DXMATRIXA16 mTextureTransform;

D3DXMATRIXA16 mProj;

D3DXMATRIXA16 mTrans;

D3DXMATRIXA16 mScale;

g\_pd3dDevice->GetTransform( D3DTS\_PROJECTION, &mProj );

D3DXMatrixTranslation( &mTrans, 0.5f, 0.5f, 0.0f );

D3DXMatrixScaling( &mScale, 0.5f, -0.5f, 1.0f );

mTextureTransform = mProj \* mScale \* mTrans;

g\_pd3dDevice->SetTransform( D3DTS\_TEXTURE0, &mTextureTransform );

g\_pd3dDevice->SetTextureStageState( 0, D3DTSS\_TEXTURETRANSFORMFLAGS, D3DTTFF\_COUNT4 | D3DTTFF\_PROJECTED );

g\_pd3dDevice->SetTextureStageState( 0, D3DTSS\_TEXCOORDINDEX, D3DTSS\_TCI\_CAMERASPACEPOSITION );

#endif

// Render the vertex buffer contents

g\_pd3dDevice->SetStreamSource( 0, g\_pVB, 0, sizeof( CUSTOMVERTEX ) );

g\_pd3dDevice->SetFVF( D3DFVF\_CUSTOMVERTEX );

g\_pd3dDevice->SetTexture(0, g\_pTexture);

g\_pd3dDevice->DrawPrimitive( D3DPT\_TRIANGLESTRIP, 0, 98);

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 100, 98);

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 200, 98);

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 300, 98);

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 400, 98);

g\_pd3dDevice->DrawPrimitive(D3DPT\_TRIANGLESTRIP, 500, 98);

// End the scene

g\_pd3dDevice->EndScene();

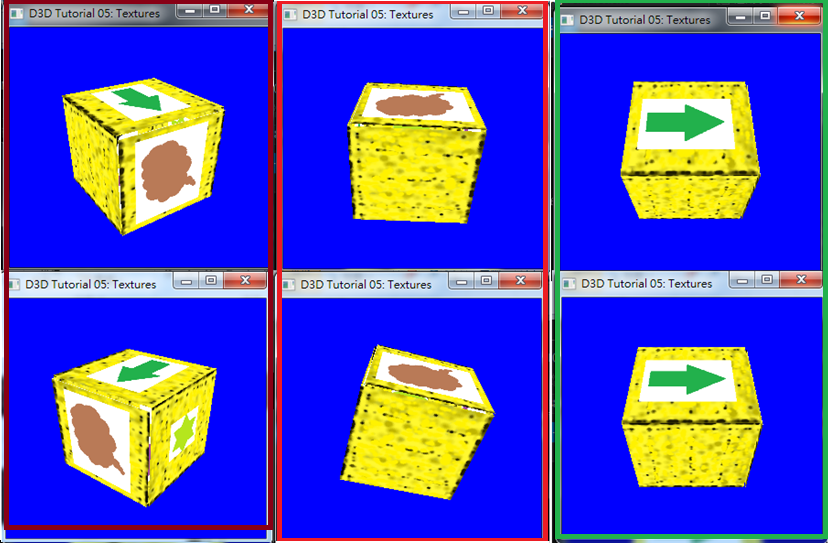
}

// Present the backbuffer contents to the display

g\_pd3dDevice->Present( NULL, NULL, NULL, NULL );

}

(四)成果:



(**圖)向Y軸旋轉 向Z軸旋轉 向X軸旋轉**