Multidimensional Graphics Implementation Using DirectX API Class Library

Ana'am S. Nasser* & **Jumana Basheer Muhammed****

Received on:4/5/ 2008
Accepted on :7/5/ 2009

Abstract

Several techniques or capabilities are provided by DirectX, each of which can add effects on the designed graphics by using special functions in the application or program that written to design that graphics. Some of these capabilities are that, Lighting, Rotation, Texturing, Fog, Environment Mapping and Stencil Buffer capabilities.

In this research, implementation of two capabilities are produced, the first implementation is of the Texturing capability and show several states of texturing by using variables values of tu and tv texture coordinates.
The second implementation is of the Environment Mapping capability which shows how a shiny teapot is rotated and only one face of the skybox is reflected by it.

DirectX can be written in many programming languages, such as C, C++, Visual C++, and Visual Basic.

When using the DirectX capabilities in many applications, not all cards support all that capabilities at the implementation time, some of these like Lighting, Rotation and Texturing capabilities can be implemented with 64-Mega Byte VGA cards, but others like Fog, Environment Mapping and Stencil Buffer capabilities are implemented with 128-Mega Byte and over.

Keywords: DirectX; DirectX Capabilities; Texturing Capability; Environment Mapping

تطبيق الرُسوماتُ المتعددة الأبعادُ بإستخدام مكتبةِ الـــ DirectX API

الخلاصة

تم تجهيز العديد من التقنيات او القابليات مِن قِبلDirectX ، كلّ منها يُمْكِنُ أنْ يُضيف تأثيراتَ على الرُسوماتِ المُصَمَّمةِ بإستعمال الوظائفِ الخاصّةِ في التطبيق أو البرنامج المكتوبِ لتَصميم تلك الرُسوماتِ. البعض مِنْ هذه القابلياتِ هي Lighting, Rotation, Texturing, Fog, Environment Mapping and Stencil Buffer. في هذا البحثِ، تم تطبيق قابليتين ، التطبيق الأول مِنْ قابليةِ Texturing ويُظهر عِدّة حالات للـــ texturing بإستعمال قِيَم متغيّراتِ لاحداثيات الـــ texturing وهي tu و tv. التطبيقَ الثانيَ مِنْ قابليةِ Environment Mapping ويظهركيفية دوران اريق مضيء وانعكاس وجه واحد فقط من الـــ skybox بواسطته. ممكن برمجة الـــ DirectX باستخدام العديد مِنْ لغاتِ البرمجة، مثل C، C++ ، visual C++ و Visual Basic.عند إستعمال قابلياتِ DirectX في العديد مِنْ التطبيقاتِ، ليسَ كلّ البطاقات تَدْعمُ كلّ تلك القابلياتِ في وقتِ التطبيقَ، البعض مِنْ هذه التطبيقات مثل Lighting،

*Computer Center, University of Technology / Baghdad
**Laser& Optoelectronics Engineering Department, University of Technology / Baghdad

https://doi.org/10.30684/etj.27.11.7
2412-0758/University of Technology-Iraq, Baghdad, Iraq

ممكن تطبيقها باستخدام كارت شاشة ذو سعة 64 ميغابايت، لكن Texturingو Rotation قابليات اخرى مثل Fog, Environment Mapping and Stencil Buffer ممكن تطبيقها باستخدام كارت شاشة ذو سعة 128 ميغابايت فما فوق.

1. Introduction

Computers have become a powerful tool for the rapid and economical production of pictures. There is virtually no undertaking in which graphical displays cannot be used to some advantage, and so it is not surprising to find the use of computer graphics so widespread. Although early applications in engineering and science had to rely on expensive and cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. Today, computer graphics are used routinely in such diverse fields as science, art, engineering, business, industry, medicine, government, entertainment, advertising, education, training, and home applications. And graphical images can be even transmitted around the world using the Internet [1].

DirectX is a set of multimedia Application Programming Interfaces (API's) written by Microsoft. It is a collection of Dynamic Link Libraries (DLLs) that contain functions useful to a wide range of multimedia programmers, but are all almost entirely Microsoft windows platform independent. This allows programmers access to fast graphics, sound, and input functions while not needing their applications to test for the capabilities of the computer on which their program is running. DirectX will evaluate these capabilities and if they are not present, DirectX may attempt (in many cases) to emulate the functions in software instead of hardware [2, 3]. DirectX provides a standard interface to interact with graphics and sound cards, input devices and more. Without this standard set of APIs, different code would have to be written for each combination of graphics and sound cards and for each type of keyboard, mouse and joystick. DirectX abstracts the user from the specific hardware and translates a common set of instructions into the hardware specific commands.

The DirectX architecture is made up of many components [4]

- *Direct3D*
- *DirectDraw*
- *DirectInput*
- *DirectMusic*
- *DirectPlay*
- *DirectSetup*
- *DirectSound*

Direct3D is that portion of DirectX which can draw three dimensional shapes with smooth Gouraud shading and scientifically correct lighting. With Direct3D, it can create a program that allows the operator to dynamically adjust his position with respect to a 3D object, to zoom in and out, and to rotate the object about its central axis. It can even fly right through the object to view its interior [5]. Direct3D first appeared in DirectX 2.0, which came out in 1996. In the short time since its introduction, Direct3D has become the most popular 3D API on the market. Many graphics and games

use it, and almost every 3D graphics accelerator produced supports it.

Direct3D applications work with graphics acceleration hardware similarly in both Retained Mode and Immediate Mode. They use hardware through the HAL, if it's available—otherwise, they use the HEL. Because Direct3D is an interface to a DirectDraw object, Microsoft refers to this HAL as the DirectDraw/Direct3D HAL. Figure 1 illustrates how Direct3D works with the Microsoft Win32 environment and the available hardware accelerators [6, 7, and 8].

2. DirectX Key Concepts

DirectX is a collection of API's interface the Hardware Abstraction Layer (HAL) layered on top of the standard hardware-specific drivers and hardware. DirectX creates a common hardware independent layer providing support of many different types of hardware. When using DirectX the program is written on a single hardware and DirectX will guarantee that the program will be run on all other DirectX compatible hardware components.

DirectX is an API designed to give the programmer a set of functions to access the GPU (Graphics Processing Unit) of a computer [9].

DirectX is a mighty and powerful interface, providing the programmer with a lot of functions to do anything it can be thought of. But the price is that using DirectX (in this case, the 9th generation) is a kind of an enigma to most programmers who might need it for home use [10].

3. DirectX Capabilities

DirectX provides many capabilities that can be used in many applications, each of these capabilities can be used to add some effect to the application that the user works with, such capabilities are Lighting, Rotation, Texturing, Fog, Light Mapping and Environment Mapping and Stencil Buffer capability.

The DirectX capabilities such as Lighting, Rotation, Texturing and Fog can be used with any object in an application to add some effect to that object. These capabilities can be done by using some functions as with Lighting and Texturing capabilities or by using *CMyD3DApplication* class as with Fog capability. This class contains several member functions; each member function is responsible for some thing. One of the programming languages used to implement these capabilities is the Visual C++ language; two capabilities will be produced here one for Texturing capability and other for Environment Mapping capability.

3.1 Texturing Capability

It can be thought of a texture as a bitmap of pixel colors that applied to the surface of a 3D object. In the real world, *texture* refers to an object's color, pattern, and tactile characteristics. Although textures are only bitmaps, they can be applied to 3D primitives to make them look like real-world objects.

3.2 Environment Mapping Capability

Environment mapping is a technique in which the environment surrounding a 3D object (such as the lights, etc.) is put into a texture map, so that the object can have complex lighting effects without expensive lighting calculations [11].

. 4. Implementation of DirectX Capabilities

DirectX, as mentioned earlier provides many capabilities that can be used in many applications, here two capabilities are produced and implemented, the first is the Texturing capability and the second is the Environment Mapping capability. The Texturing capability is done by using some functions, while the Environment Mapping capability is done by using *CMyD3DApplication* class, this class contains several member functions; each member function is responsible for some thing.

4.1 Texturing Capability Implementation

To implement this capability, a cylinder is taken as an object drawn by creating vertex buffer using the *CreateVertexBuffer ( )* function, then filling this vertex buffer with set of vertices represent the cylinder by using the following loop:

```
for( DWORD i=0; i<50; i++ )
  {
    FLOAT theta = (2*D3DX_PI*i)/(50-1);
    pVertices[2*i+0].position = D3DXVECTOR3( sinf(theta),-1.0f, cosf(theta) );
    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( sinf(theta), 1.0f, cosf(theta) );
    pVertices[2*i+1].color = 0xff808080;
#ifndef SHOW_HOW_TO_USE_TCI
    pVertices[2*i+1].tu = ((FLOAT)i)/(50-1);
    pVertices[2*i+1].tv = 0.0f;
#endif
  }
```

In this loop the *tu* and *tv* texture coordinates are set and range from 0.0 to 1.0 to apply the texture which is an image represent the Iraqi flag.

If pVertices[2*i+0].tv = 0.5f and pVertices[2*i+1].tv = 0.0f are taken, the above half of the used texture will be shown on the object as in figure 2.

If pVertices[2*i+0].tv = 0.5f and pVertices[2*i+1].tv = 1.0f are taken, the down half of the used texture will be shown as in figure 3.

However, figure 4 shows the application output of Texturing capability with Rotation capability.

4.2 Environment Mapping Capability Implementation

In the implementation of this capability, two objects are used; the first object is a shiny teapot which represents the chrome sphere. The second object is a box named here skybox, which is like a room of six faces each of which take a single texture and only one of theme will be reflected on the shiny teapot. These two objects are x files stored and loaded from DirectX by using *Create ( )* function and the texture that reflected on the teapot represent bitmap image named here spheremap also stored and loaded from DirectX using *D3DUtil_CreateTexture ( )* function as bellow:

```
if( FAILED( m_pShinyTeapot->Create( m_pd3dDevice, _T("teapot.x") ) ) )
    return D3DAPPERR_MEDIANOTFOUND;
```

```
if( FAILED( m_pSkyBox->Create( m_pd3dDevice, _T("lobby_skybox.x") ) ) )
    return D3DAPPERR_MEDIANOTFOUND;
if( FAILED( D3DUtil_CreateTexture( m_pd3dDevice, _T("spheremap.bmp"), &m_pSphereMap ) ) )
    return D3DAPPERR_MEDIANOTFOUND;
```

Also *Render ( )* member function is used to rendering the skybox and shiny teapot, to rendering the skybox the following code is used:

```
D3DXMATRIXA16 matWorld;
D3DXMatrixScaling( &matWorld, 10.0f, 10.0f, 10.0f );
D3DXMATRIXA16 matView(m_matView);
matView._41 = matView._42 = matView._43 = 0.0f;
m_pd3dDevice->SetTransform( D3DTS_WORLD, &matWorld );
m_pd3dDevice->SetTransform( D3DTS_VIEW, &matView );
m_pd3dDevice->SetTransform( D3DTS_PROJECTION, &m_matProject );
m_pd3dDevice->SetTextureStageState( 0, D3DTSS_COLORARG1, D3DTA_TEXTURE );
m_pd3dDevice->SetTextureStageState( 0, D3DTSS_COLOROP,

D3DTOP_SELECTARG1 );
m_pd3dDevice->SetSamplerState( 0, D3DSAMP_MINFILTER,

D3DTEXF_LINEAR );
m_pd3dDevice->SetSamplerState( 0, D3DSAMP_MAGFILTER,

D3DTEXF_LINEAR );
if( (m_d3dCaps.TextureAddressCaps & D3DPTADDRESSCAPS_MIRROR) ==

D3DPTADDRESSCAPS_MIRROR )
{
    m_pd3dDevice->SetSamplerState( 0, D3DSAMP_ADDRESSU,

D3DTADDRESS_MIRROR );
    m_pd3dDevice->SetSamplerState( 0, D3DSAMP_ADDRESSV,

D3DTADDRESS_MIRROR );
 }
```

It needed always passing z-test to avoid clearing color and depth buffers.

```
m_pd3dDevice->SetRenderState( D3DRS_ZFUNC, D3DCMP_ALWAYS );
m_pSkyBox->Render( m_pd3dDevice );
m_pd3dDevice->SetRenderState( D3DRS_ZFUNC, D3DCMP_LESSEQUAL );
```

To render the shiny teapot the following code is used, it is needed first setting transform state:

```
D3DXMATRIXA16 matViewProject;
D3DXMatrixMultiply( &matViewProject, &m_matView, &m_matProject );
D3DXMATRIXA16 matViewInv;
D3DXMatrixInverse( &matViewInv, NULL, &m_matView );
D3DXVECTOR4 vecPosition(matViewInv._41, matViewInv._42, matViewInv._43, 1.0f);m_pEffect->SetMatrix( "matWorld", &m_matWorld );
```

```
m_pEffect->SetMatrix(
"matViewProject", &matViewProject
);
m_pEffect->SetVector(
"vecPosition", &vecPosition );
```

Figure 5 shows the application output of SphereMap Capability (the teapot is rotated around y-axis).

The shiny teapot here is rotated around the y-axis inside the skybox and only one face of that skybox can reflected by it, it can rotate the skybox by the mouse using the function and note that however it rotates, the teapot still reflect only one face as bellow:

5. Conclusions

DirectX has many techniques or capabilities which can be added to give the graphics or games some effects, these capabilities can be added to the graphics or games by using set of API functions provided in DirectX.

The present work has reached to the following conclusions:-

- In Environment Mapping technique (capability), it is noted that not all cards support all features for all the various environment mapping techniques (such as cube mapping and projected textures), this capability can not implemented with 64-bit AGP card, but implemented with 128-bit AGP.
- Accelerated Graphics Port (AGP) memory is an excellent tool for texturing applications. It can achieve excellent performance using AGP memory, and it's the best option for using multiple, high-resolution textures.

6. References

[1] Donald Hearn & M. Paulin Baker, "Computer Graphics with Open GL", 3'd Edition 2004.

[2] Mungler, "www.d-silence.com, Just what is DirectX?", 21 Sep 2004.

[3] GPWiki, "http://gpwiki.org, DirectX", 2007.

[4] Keith Sink, "www.samspublishing.com, Game programming with DirectX", 20 Aug 2001.

[5] John Kopplin, "www.codeproject.com, Use Direct3D 8 to Fly Through The Munsell Color Solid", 20 Jul 2004.

[6] Peter J. Kovach, "Inside Direct3D", Microsoft Company, Feb 2000.

[7] David Joffe, "www.scorpioncity.com, Game Programming with DirectX", 2006.

[8] Microsoft Corporation, "DirectX 7 SDK", 2 Sep 1999.

[9] James Gupta, "www.codeproject.com, Managed DirectX Tutorials", 2007.

[10] Thomas Bock, "www.CodeGuru.com, DirectX Programming Using DirectX 9", 6 May 2003.

[11] Microsoft Corporation, "DirectX 9.0 SDK", 2002.

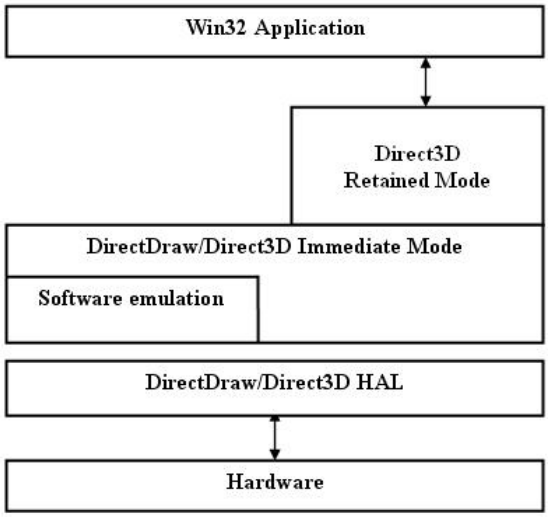

Figure (1) The relationship of DirectDraw/Direct3D to the DirectX environment and to the system

Figure (2) The application output of Texturing capability with the above half of the used texture is shown

Figure (3) The application output of Texturing capability with the down half of the used texture is shown

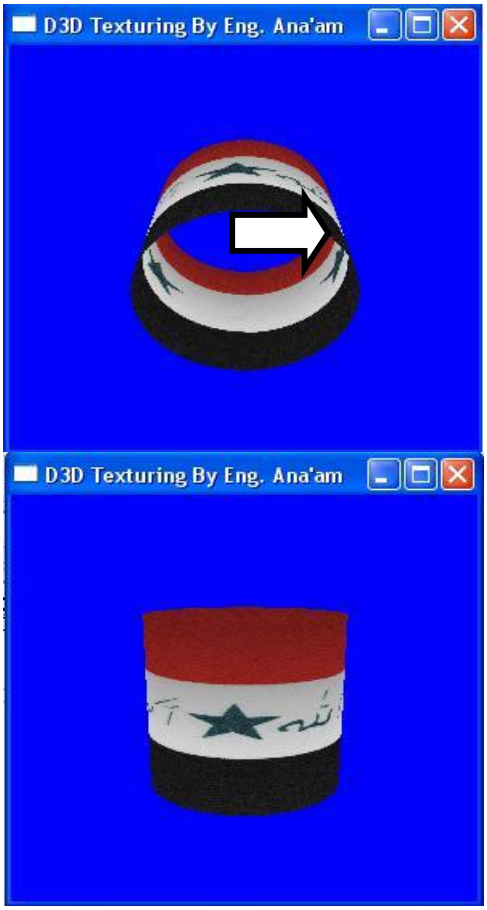

Figure (4) The Texturing capability application with full texturing

Figure (5) The SphereMap Capability application

Figure (6) shows that teapot still reflect one face of skybox however it rotate