DirectX Overview & Installation

305890
2008년 봄학기
3/12/2008
박경신

Overview

- Game programming
- DirectX
- DirectX installation
- Win32, DirectX, Direct3D examples

Game Programming

- Graphic
- Sound & Video
- 물리 (Physics)
- 인공지능 (Artificial Intelligence)
- 네트워크 (Networking)
- 스크립팅 (Scripting)
- 사용자 인터페이스 (User Interface)
- 기획

Game Programming

- API(Application Program Interface)
  - HAL(Hardware Abstraction Layer), HEL(Hardware Emulation Layer)
- 3D Graphics API
  - DirectX, OpenGL, Java3D, etc
- Game Design Software
  - DarkBasic, Game Maker, 3D GameStudio, RPG Maker, MUSEN, etc

Game Programming

- 게임 엔진
  - Higher level interface
    - Sprite based
    - Isometric
    - Full 3D
  - Higher level modeling concepts
    - Sprites, Solids, Characters (articulated), …
  - 다양한 디스플레이 기능 제공
    - Mini map, Multiple views, Overlays, Special effects
  - Game Design Software
    - http://www.gamediscovery.com/game-design/game-design-software.asp
  - Game Engine
    - http://en.wikipedia.org/wiki/List_of_game_engines

Game Programming

- Sound & Video
  - 사운드의 추가로 사실성을 극대화, 다양한 clues를 사운드로 제공
  - 사운드 format
    - Wave, AIFF (high quality, lots of memory, fast)
    - MP3 (high quality, compressed, slower)
    - MIDI (lower quality, very low storage, limited, adaptable)
    - CD (Very high quality, fast, limited to background music)
  - 사운드 동시 재생
    - Mixers (hidden in the HAL), Buffer management, Streaming sound
  - Special sound features
    - Positional sound (possibly with Dolby surround)
    - Adaptive music (DirectMusic)
  - 사운드 제작
    - 샘플링이나 리코딩

Game Programming

- AI
  - 게임 AI를 가지고 게임플레이를 지원
  - 게임 내에서 플레이어의 상대적 역할을 수행하면서 마치 플레이어가 다른 사람과 게임을 하고 있다는 착각을 유도
  - Finite State Machine (FSM), Path Finding, Flocking, …
- Physics
  - 실제 환경과 유사하게 물리적인 현상을 시뮬레이션
  - Collisions, terrain changes, water effects, particle system, …
  - 간단한 게임에서는 극히 제한적으로 사용 (또는 전혀 사용하지 않음)
  - Havok, PhysX (NovodeX 후속 버전)

Game Programming

- 사용자 인터페이스
  - Configuration, game data등의 변화를 주시하고 있음
  - Menus나 online help 등을 제공
- Configuration System
  - 하드웨어 스펙 (hardware specs)에 맞추는 기능
  - 게임 플레이어의 성향 (player preferences)에 맞추는 기능
- Online Help
  - 게임 플레이어를 도와주는 역할
  - Players never read the documentation! 따라서 아주 간단한 게임이라 해도 online help는 도움이 됨. 게임이 복잡해 질 수록 online help도 지원할 내용이 많아짐.
  - 일반적으로 화면 위에 (주 화면보다 다르게 해서) overlay해서 보여줌

Game Programming

- Game Data
  - "A game is a database with a fancy interface"
  - Resources
    - graphics models (sprites, characters)
    - sounds, music
    - images, backgrounds, video
  - Level description / Game status / List of events
  - User profile

Game Programming

- Event Handler
  - 대부분의 게임이 event-based model을 사용하고 있음
    - Events가 생기면 게임 status에 따라 게임 엔진의 내용이 변함
  - Events
    - User input, Collisions, Timers (controlled by the logic), 등등
    - Game tokens으로 생성. state diagrams (finite-state machines)과 interaction matrices를 사용해서 behavior를 표현함.
  - Timing
    - 하드웨어 자체의 frame rate을 사용
      - Higher chance of sudden drop in speed
      - Old games run way too fast (100 fps versus 50, or less)

Game Program Structure

- Game Loop
  - Draw playing area (need a way of representing levels)
  - Draw players in playing are
  - Do
    - Read key press, joystick, etc..
    - [Read from Network]
    - [Process AI]
    - Update player positions
    - Process power-ups (double speed, the ability to walk on water, lava, or acid, the ability to jump higher or fall like a feather,…)
    - Check for collisions
    - Update score, status, etc…
    - Redraw player positions
  - Loop until game ends

Game Program Structure

- Game End
  - Die screen
  - Check High score and update table
  - Show high scores
  - Return to start up phase or Main menu
- Game Introduction

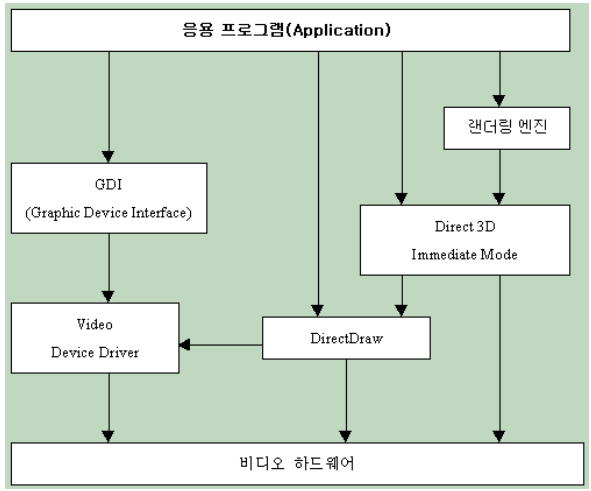
What Is DirectX?

- DirectX는 Microsoft사에서 Windows 운영체계를 위한 하드웨어지원 고성능 게임 SDK (Software Development Kit)
- DirectX의 첫 번째 버전은 1995년 후반에 개발.
- DirectX가 개발된 배경은 개발자에게 게임 개발에 기반이 되는 교육과 Component를 제공해 그들로 하여금 쉬운 개발 환경을 만들 수 있도록 하는데 있음.
- 기존의 Windows 게임에서 최대의 걸림돌인 느린 화면 출력 속도를 빠르게 개선시키기 위해 나온 것임.
- DirectX는 GDI (Graphic Device Interface)라고 하는 일관된 인터페이스를 경유하지 않고 바로 하드웨어를 접근할 수 있는 방법을 제시하여 그래픽 가속을 함.

What Is DirectX?

- DirectX는 게임 개발자들에게 고성능 하드웨어 (3D 가속칩, 사운드 카드 등)의 향상된 기능을 보다 쉽게 접근/사용할 수 있는 단일 API (Application Programming Interface)를 제공
- 윈도우 기반의 컴퓨터에서 그래픽, 비디오, 3차원 애니메이션, 서라운드 사운드, 네트워크 등 멀티미디어 관련 프로그램을 실행시키기 위한 기반이 되는 기술들의 집합체
- 처음 DirectX가 발표되었을 때는 4개의 API(Application Programming Interface) - 즉 DirectDraw, DirectSound, DirectInput, DirectPlay - 가 지원되었지만, 업그레이드된 DirectX에서는 Direct3D가 추가되어 게임 라이브러리의 기능이 더욱 막강해짐

What Is DirectX?



DirectX SDK Components

- DirectX header and libraries
- DirectX component DLL (run-time)
- DirectX API (응용 프로그램 인터페이스)
- Sample applications and source codes
- 그 외 잡다한 유틸리티 (utility)

Components of DirectX

- DirectSetup
  - 게임제작을 완성하고 배포하는데 사용하는 API
  - 최신 DirectX 버전 체크와 설치하도록 도와주는 기능 제공
- DirectX Graphics
  - 그래픽 output 처리
  - 2차원 3차원 드로잉과 그래픽 초기와 및 해상도 등 그래픽 관련 셋팅.
  - DirectDraw 화면처리
  - Direct3D
  - Direct3D extension (D3DX) utility library
- DirectInput
  - 사용자 input관련된 이벤트처리
  - 키보드, 마우스, 게임패드, 조이스틱, force-feedback 등 입력장치 지원
- DirectPlay
  - Network 지원

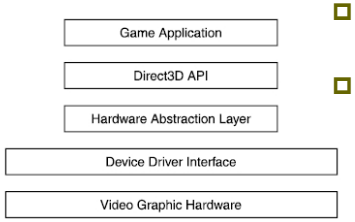
Components of DirectX

- DirectSound
  - 사운드 이팩트와 background music
  - WAV 파일 지원
  - 어떻게 사운드를 재생할 지에 대한 처리 지원
- DirectMusic
  - 동적인 soundtrack을 생성 지원
  - 정해진 시간에 맞춰 소리를 틀어줄 수 있으며 또는 pitch, tempo, volume의 변화에 따라 적응가능
- DirectShow
  - 동영상 혹은 streaming audio을 보다 빠른 속도로 다운 또는 처리
  - AVI, MP3, MPEG, ASF 파일 지원
- DirectAnimation
  - 보다 빠른 그래픽 동영상을 처리하기 위해 사용되는 함수를 내장

How Is DirectX Put Together?

- DirectX는 COM 기술에 바탕을 두고 있음
  - DirectX는 새로운 고성능 하드웨어 사용시 보다 향상된 기능을 쉽게 접근 가능하도록 지원.
  - 새로운 DirectX 버전이 출시되어도 전 버전으로 개발한 게임 소프트웨어 실행을 지원.
- COM (Component Object Model)
  - COM 객체는 개발자들이 DirectX를 사용할 수 있는 interface 집합으로 구성됨.
  - COM 객체는 일반적으로 시스템에 레지스터된 DLL 파일로 구성됨.
  - COM 객체는 메쏘드를 access하는 interface 사용을 요구함. 하나의 COM 객체 안에 다양한 버전의 interface공존 가능.
  - C++, C#, Visual Basic 등 다양한 개발언어 사용가능

The Architecture of DirectX



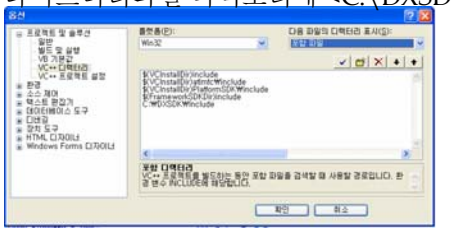
- DirectX는 API와 HAL (Hardware Abstraction Layer) 에 바탕으로 구성
- API layer는 HAL을 통하여 하드웨어와 통신을 수행
- HAL (Hardware Abstraction Layer)는 하드웨어의 가속 기능과 프로그램 간의 통신을 담당해 개발자로 하여금 다양한 하드웨어의 가속 기능을 사용할 수 있도록 함
- HEL (Hardware Emulation Layer)는 하드웨어가 지원하지 못하는 기능을 시뮬레이션함으로써 특정 시스템에서 작동되지 않을 수 있는 비호환성 문제를 해결. DirectX8.0 이후 제공되지 않음.

DirectX Installation

- DirectX SDK (November 2007)
- It contains the DirectX Runtime and all DirectX software required to create DirectX compliant applications in C/C++ and C#
- http://www.microsoft.com/downloads/details.aspx?FamilyId=4B78A58A-E672-4B83-A28E-72B5E93BD60A&displaylang=en
- **Quick Details**
  - File Name:dxsdk_november2007.exe
  - Version:9.21.1148
  - Date Published:10/26/2007
  - Language:English
  - Download Size:427.8 MB

DirectX9 Installation

- Microsoft Visual Studio .NET 2005 설치하기
- DirectX SDK (November 2007) 설치하기
  - http://www.microsoft.com/downloads/details.aspx?FamilyId=4B78A58A-E672-4B83-A28E-72B5E93BD60A&displaylang=en
- VisualStudio .NET에 DirectX9 header 파일과 library 경로 연결하기
  - 도구->옵션->프로젝트 및 솔루션->VC++ 디렉토리
  - 포함파일 디렉토리에 <C:\DXSDK>\Include 추가
  - 라이브러리파일 디렉토리에 <C:\DXSDK>\Lib 추가



DirectX9 Installation

- Library
  - d3d9.lib 기본
  - d3dx9.lib 유용한 함수가 다수 포함되어 있어 편리함
  - winmm.lib 타이머 함수 등의 사용을 위해 필요
  - 예제 프로그램을 빌드하기 위해 d3d9.lib d3dx9.lib winmm.lib 파일을 프로젝트에 링크

일반적인 win32 프로그램 구조

- RegisterClassEx()로 생성하고자 하는 윈도우의 클래스 등록
- CreateWindow()로 윈도우 생성
- ShowWindow(), UpdateWindow()로 윈도우를 화면에 표시
- GetMessage(), TranslateMessage(), DispatchMessage()로 이루어진 메시지 루프 수행
- 메시지 루프를 빠져나올 경우 프로그램을 종료

Win32 Program 작성 예

- 윈도우 클래스 (window class)

```
typedef struct {
    UINT cbSize;
    UINT style;
    WNDPROC lpfnWndProc;
    int cbClsExtra;
    int cbWndExtra;
    HINSTANCE hInstance;
    HICON hIcon;
    HCURSOR hCursor;
    HBRUSH hbrBackground;
    LPCTSTR lpszMenuName;
    LPCTSTR lpszClassName;
    HICON hIconSm;
} WNDCLASSEX;
```

- 메시지 (Message)

```
typedef struct {
    HWND hwnd;
    UINT message;
    WPARAM wParam;
    LPARAM lParam;
    DWORD time;
    POINT pt;
} MSG;
```

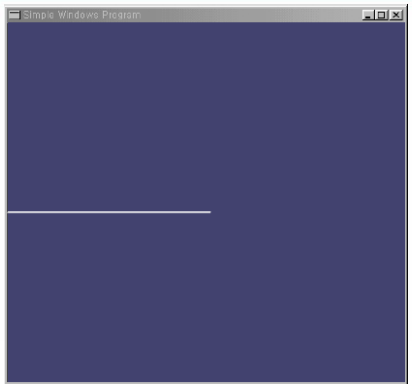
- 디바이스 문맥 (DC, Device Context)
- GDI (Graphics Device Interface)


```
#include "stdafx.h"

LRESULT CALLBACK WndProc(HWND, UINT,
                         WPARAM, LPARAM);

int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine, int nCmdShow) {
  HWND hwnd; MSG msg; WNDCLASSEX wcex;
  static char szAppName[] = "SimpleWinProg";

  wcex.cbSize = sizeof(WNDCLASSEX);
  wcex.style = CS_HREDRAW | CS_VREDRAW;
  wcex.lpfnWndProc = (WNDPROC) WndProc;
  wcex.cbClsExtra = 0; wcex.cbWndExtra = 0;
  wcex.hInstance = hInstance;
  wcex.hIcon = LoadIcon(NULL,IDI_APPLICATION);
  wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
  wcex.hbrBackground = (HBRUSH)
                CreateSolidBrush(RGB(66, 66, 111));
  wcex.lpszMenuName  = NULL;
  wcex.lpszClassName = szAppName;
  wcex.hIconSm   = LoadIcon(NULL,
                            IDI_APPLICATION);
  RegisterClassEx(&wcex);

  hwnd = CreateWindow(szAppName,
            "SimpleWindows Program",
            WS_OVERLAPPEDWINDOW,
            CW_USEDEFAULT, CW_USEDEFAULT,
            500, 500, NULL, NULL, hInstance, NULL);

  ShowWindow(hwnd, nCmdShow);
  UpdateWindow(hwnd);

  while (GetMessage(&msg, NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
  }
  return msg.wParam;
}
```



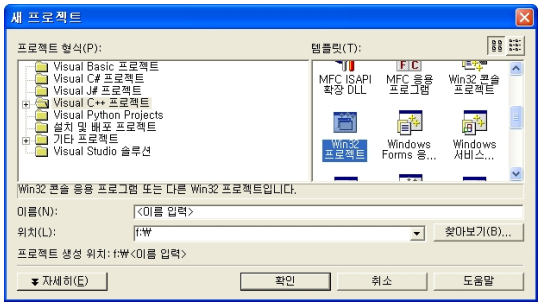
```
LRESULT CALLBACK WndProc(HWND hwnd, UINT
    message, WPARAM wParam,
             LPARAM lParam) {
  HDC hdc;
  PAINTSTRUCT ps;

  switch (message) {
    case WM_PAINT:
      hdc = BeginPaint(hwnd, &ps);
      SelectObject(hdc, GetStockObject(WHITE_PEN));
      MoveToEx(hdc, 0, 250, NULL);
      LineTo(hdc, 250, 250);
      EndPaint(hwnd, &ps);
      break;
    case WM_CHAR:
      if (wParam == 'q') {
        PostQuitMessage(0);
      }
      break;
    case WM_DESTROY:
      PostQuitMessage(0);
      break; \\
```

```
    default:
      return DefWindowProc(hwnd, message,
                           wParam, lParam);
    }
    return 0;
}
```



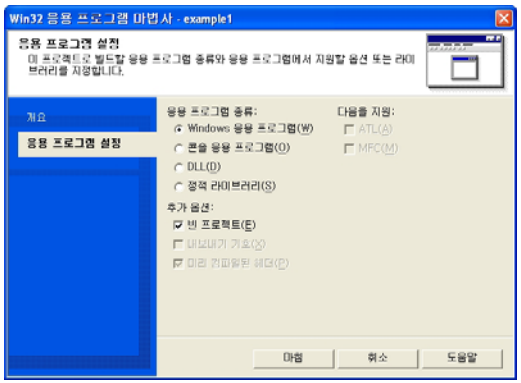
Your First DirectX Program

- 프로젝트 새로 만들기
  - 메뉴에서 파일->새로 만들기->프로젝트
  - 프로젝트 이름을 example1이라고 넣고 프로젝트 템플릿 리스트에서 Visual C++->Win32-> Win32 프로젝트를 선택한다.
  - 프로젝트 디렉토리 위치도 지정한 후 확인 버튼을 누른다



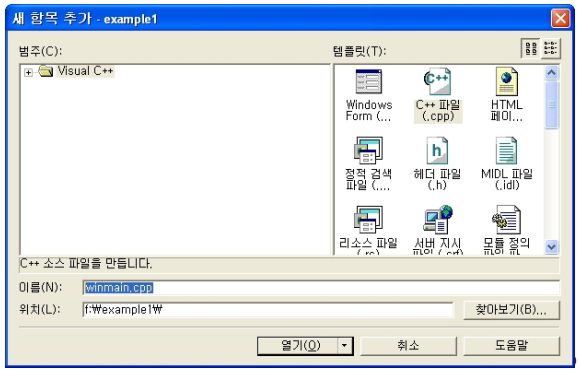
Your First DirectX Program

- 응용 프로그램 설정을 선택하고 "Windows 응용 프로그램"과 "빈 프로젝트"를 선택한 후 "마침" 버튼을 누른다. 그렇게 되면 비주얼 스튜디오에는 빈 프로젝트가 한 개 만들어진 상태가 된다.



Your First DirectX Program

- Windows 코드를 추가
  - 프로젝트->새항목추가 선택, 그리고 템플릿에서 C++ 파일을 선택. 마지막으로 파일 이름을 winmain.cpp라고 입력하고 열기 버튼을 누른다.




```
// Include the Windows header file that's needed for all Windows applications
#include <windows.h>

HINSTANCE hInst;            // global handle to hold the application instance
HWND wndHandle;           // global variable to hold the window handle

// forward declarations
bool initWindow( HINSTANCE hInstance );
LRESULT CALLBACK WndProc( HWND, UINT, WPARAM, LPARAM );

// This is winmain, the main entry point for Windows applications
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPTSTR lpCmdLine, int nCmdShow )
{
    // Initialize the window
    if ( !initWindow( hInstance ) )
        return false;

    // main message loop:
    MSG msg;
    ZeroMemory( &msg, sizeof( msg ) );
    while( msg.message!=WM_QUIT )
    {
        // Check the message queue
        while (GetMessage(&msg, wndHandle, 0, 0) )
        {
            TranslateMessage( &msg );
            DispatchMessage( &msg );
        }
    }
    return (int) msg.wParam;
}
```

Message Loop

GetMessage 함수는 WM_QUIT를 조사할 때까지 프로그램의 모든 메시지 큐를 처리함. 사용자 인풋이나 시스템 메시지를 기다림. GetMessage 함수는 메시지 큐에 대기중인 메시지가 없을 경우 메시지가 전달될 때까지 무한 대기. DispatchMessage함수에 의해 해당 윈도우의 윈도우 프로시저로 보내짐.

Your First DirectX Program

- InitWindow
  - 응용프로그램은 윈도우 클라스로 등록해야 한다.

```
WNDCLASSEX wcex;

// Fill in the WNDCLASSEX structure. This describes how the window
// will look to the system
wcex.cbSize             = sizeof(WNDCLASSEX);  // the size of the structure
wcex.style              = CS_HREDRAW | CS_VREDRAW; // the class style
wcex.lpfnWndProc        = (WNDPROC)WndProc;    // the window procedure callback
wcex.cbClsExtra         = 0;       // extra bytes to allocate for this class
wcex.cbWndExtra         = 0;       // extra bytes to allocate for this instance
wcex.hInstance          = hInstance;   // handle to the application instance
wcex.hIcon              = 0;  // icon to associate with the application
wcex.hCursor            = LoadCursor(NULL, IDC_ARROW);// the default cursor
wcex.hbrBackground      = (HBRUSH)(COLOR_WINDOW+1);   // the background color
wcex.lpszMenuName  .    = NULL;    // the resource name for the menu
wcex.lpszClassName      = "DirectXExample"; // the class name being created
wcex.hIconSm            = 0;       // the handle to the small icon
RegisterClassEx(&wcex);
```

Your First DirectX Program

- CreateWindow
  - Window를 생성하고 보여준다

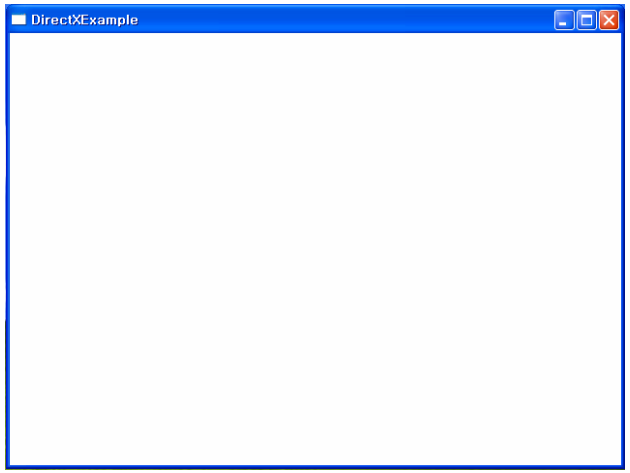
```
// create the window
wndHandle = CreateWindow(
                "DirectXExample",       // the window class to use
                "DirectXExample",       // the title bar text
                WS_OVERLAPPEDWINDOW,    // the window style
                CW_USEDEFAULT,          // the starting x coordinate
                CW_USEDEFAULT,          // the starting y coordinate
                640,                    // the pixel width of the window
                480,                    // the pixel height of the window
                NULL,                   // the parent window; NULL for desktop
                NULL,                   // the menu for the application; NULL for none
                hInstance,              // the handle to the application instance
                NULL);                  // no values passed to the window

// Make sure that the window handle that is created is valid
if (!wndHandle) return false;

// Display the window on the screen
ShowWindow(wndHandle, SW_SHOW);
UpdateWindow(wndHandle);
return true;
```


Your First DirectX Program

- WndProc
  - 시스템으로부터 이벤트(예를 들어, 마우스 이벤트 등)를 처리한다.

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    // Check any available messages from the queue
    switch (message)
    {
        case WM_DESTROY:
            PostQuitMessage(0);
            break;

    }
    // Always return the message to the default window
    // procedure for further processing
    return DefWindowProc(hWnd, message, wParam, lParam);
}
```


Your First DirectX Program



Direct3D Programming

- Direct3D 프로그램 예제
  1. Direct3D 객체 생성
  2. Direct3D 디바이스 생성
  3. 디바이스에 그림을 그려줌

- winmain.cpp 프로그램에 DirectX header file을 추가

```
#include <d3d9.h>
```

- 전역변수

```
LPDIRECT3D9 pD3D;                   // Direct3D 객체
LPDIRECT3DDEVICE9 pd3dDevice;  // Direct3D 디바이스
```

- initDirect3D 함수를 추가

```
// called after creating the window
if (!initDirect3D())
        return false;
```

Direct3D Programming

- Direct3D 객체 생성
  - Direct3DCreate9함수는 IDirect3D9 interface를 생성해서 그 주소값을 리턴한다.
  - 이 함수가 제대로 불리지 않았을 경우 NULL을 리턴한다.

```
IDirect3D9 *Direct3DCreate9(D3D_SDK_VERSION);
```

인자값은 D3D_SDK_VERSION으로 고정

- IDirect3D9 멤버
  - GetAdapterCount
    - 시스템에 있는 어댑터의 수를 돌려준다.
    - IDirect3D9 인터페이스가 인스턴스화 된 시점에서 시스템에 존재하고 있던 어댑터의 수를 나타내는 UINT 값.
  - 1차 디스플레이 비디오 카드가 주요 어댑터이다.

Direct3D Programming

- Direct3D 디바이스 생성
  - 디스플레이 어댑터를 나타내기 위한 장치를 생성한다.
  - 디바이스가 생성된 후에, 화면에 그려질 다른 Direct3D 방법들이 불려질 수 있다.

```
HRESULT CreateDevice(UINT Adapter,
                     D3DDEVTYPE DeviceType,
                     HWND hFocusWindow,
                     DWORD BehaviorFlags,
                     D3DPRESENT_PARAMETERS *pPresentationParameters,
                     IDirect3DDevice9 ** ppReturnedDeviceInterface);
```


Direct3D Programming

- CreateDevice의 return 값
  - CreateDevice가 성공했으면, D3D_OK를 돌려준다
  - 에러가 생기면,
    - D3DERR_INVALIDCALL 메서드의 호출이 무효이다. 예를 들어 메서드의 인자에 무효인 값이 설정되어 있는 경우.
    - D3DERR_NOTAVAILABLE 이 장치는 문의한 처리를 지원하지 않음.
    - D3DERR_OUTOFVIDEOMEMORY 이 값은 Direct3D가 처리를 실시하는데 충분한 디스플레이 메모리가 없다.

Direct3D Programming

- CreateDevice의 parameters
  - Adapter
    - 디스플레이 어댑터를 나타내는 서수.
    - D3DADAPTER_DEFAULT는 항상 1차 디스플레이 어댑터임.
  - DeviceType
    - D3DDEVTYPE 열거형 멤버.
      - D3DDEVTYPE_HAL: 하드웨어 가속과 래스터화
      - D3DDEVTYPE_REF: Microsoft 레퍼런스 래스터기
      - D3DDEVTYPE_SW: 소프트웨어 디바이스
  - hFocusWindow
    - Direct3D 장치로 포커스를 설정하는 윈도우 핸들
  - BehaviorFlags
    - 정점처리를 어디서 할지를 결정하는 값
    - 장치 생성을 제어하는 1개 또는 복수의 옵션의 편성
    - E.g. D3DCREATE_SOFTWARE_VERTEXPROCESSING
      - 소프트웨어에 의한 정점 처리를 지정한다.

Direct3D Programming

- CreateDevice의 parameters
  - PresentationParameters
    - D3DPRESENT_PARAMETERS 구조체의 포인터
    - 생성하는 장치의 presentation parameter가 기술

```
D3DPRESENT_PARAMETERS d3dpp;
ZeroMemory( &d3dpp, sizeof(d3dpp) );
d3dpp.Windowed = TRUE; // 윈도우모드로 실행
d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD; // Back버퍼를 사용
d3dpp.BackBufferFormat = D3DFMT_UNKNOWN;
d3dpp.BackBufferCount  = 1;
d3dpp.BackBufferHeight = 480;
d3dpp.BackBufferWidth  = 640;
d3dpp.hDeviceWindow    = wndHandle;
```

  - ppReturnedDeviceInterface
    - 돌려받는 IDirect3DDevice9 interface 포인터 주소. 생성된 장치를 나타낸다.

Direct3D Programming

- Clear
  - 장면을 지정한 색으로 채워 넣는 일을 하는 함수

```
HRESULT Clear(DWORD  Count, const D3DRECT *pRects,
              DWORD Flags, D3DCOLOR Color,
              float Z, DWORD Stencil);
```

  - Count
    - The number of rectangles that will be cleared
    - If this value is 0, pRects must be NULL. In this case, the entire viewing area of the screen will be cleared
    - If > 0, pRects must point to an array of D3DRECT structures designating the rectangular areas of the screen to be cleared

Direct3D Programming

- Flags
  - Specifies the buffer to be cleared
  - Three possible values
    - D3DCLEAR_STENCIL
    - D3DCLEAR_TARGET
    - D3DCLEAR_ZBUFFER
- Color
  - D3DCOLOR value
  - The color to clear the render target to
- Z
  - Holds the value to store in the depth buffer
  - Ranges from 0.0f to 1.0f
- Stencil
  - Holds the value to store in the stencil buffer. When not in use, set it to 0

Direct3D Programming

- Present
  - Back 버퍼에 그려진 장면을 화면에 디스플레이 해준다. Back버퍼의 내용을 전면 버퍼로 바꿔주는 것이다.

```
HRESULT Present(CONST RECT *pSourceRect,
                CONST RECT *pDestRect,
                HWND hDestWindowOverride,
                CONST RGNDATA *pDirtyRegion);
```

Direct3D Programming

- pSourceRect
  - A pointer to a RECT structure containing the source rectangle to display from the back buffer. If NULL, the entire back buffer is used
- pDestRect
  - RECT that contains the destination rectangle
- hDestWindowOverride
  - The destination window to use as the target area
  - NULL if you want to use the window specified earlier in the presentation parameters structure
- pDirtyRegion
  - The region within the buffer that needs to be updated
  - NULL to update the whole buffer

Direct3D Programming

- Release
  - 프로그램이 종료될 때 D3D 객체와 디바이스도 release해준다.

```
if (pd3dDevice != NULL)
    pd3dDevice->Release();

if (pD3D != NULL)
    pD3D->Release();
```


Changing the Message Loop

- GetMessage는 메시지가 없으면 메시지가 들어올 때까지 기다리고 있다.
- PeekMessage는 메시지가 있는지를 확인한 후 바로 값을 return해준다. 따라서 메인 메시지 루프에서 다른 함수를 넣어 줄 수 있다.

```
if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
}
else {
        render();
}
```


Init Function

```
bool initDirect3D(void)
{
    pD3D = NULL;
    pd3dDevice = NULL;

    // Create the DirectX object
    if( NULL == ( pD3D = Direct3DCreate9( D3D_SDK_VERSION ) ) )
    {
        return false;
    }

    // Fill the presentation parameters structure
    D3DPRESENT_PARAMETERS d3dpp;
    ZeroMemory( &d3dpp, sizeof( d3dpp ) );
    d3dpp.Windowed = TRUE;
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
    d3dpp.BackBufferFormat = D3DFMT_UNKNOWN;
    d3dpp.BackBufferCount  = 1;
    d3dpp.BackBufferHeight = 480;
    d3dpp.BackBufferWidth  = 640;
    d3dpp.hDeviceWindow    = wndHandle;

    // Create a default DirectX device
    if( FAILED( pD3D->CreateDevice( D3DADAPTER_DEFAULT,
                                    D3DDEVTYPE_REF,
                                    wndHandle,
                                    D3DCREATE_SOFTWARE_VERTEXPROCESSING,
                                    &d3dpp,
                                    &pd3dDevice ) ) )
    {
        return false;
    }
    return true;
}
```

Setting the app in windowed mode with 640x480 resolution

Would make it run on most platforms

Render Function

```
void render(void)
{
    // Check to make sure you have a valid Direct3D device
    if( NULL == pd3dDevice )
        return;// Clear the back buffer to a blue color
    pd3dDevice->Clear( 0, NULL, D3DCLEAR_TARGET,
                       D3DCOLOR_XRGB( 0,0,255 ), 1.0f, 0 );

    // Present the back buffer contents to the display
    pd3dDevice->Present( NULL, NULL, NULL, NULL );
}
```

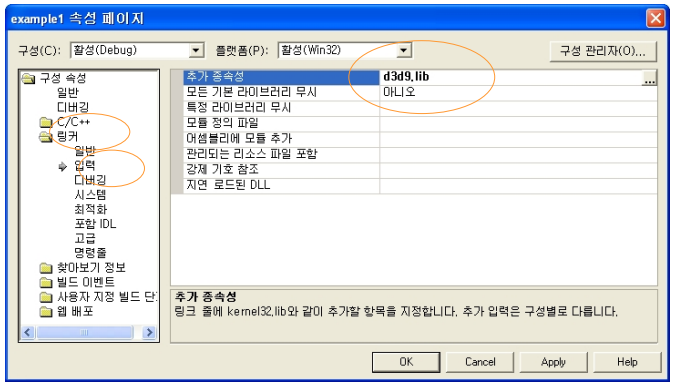
Clearing the screen with blue color

CleanUp Function

```
void cleanUp (void)
{
    // Release the device and the Direct3D object
    if( pd3dDevice != NULL )
        pd3dDevice->Release( );

    if( pD3D != NULL )
        pD3D->Release( );
}
```


Adding the DirectX Libraries

- 프로젝트->속성 메뉴 항목을 선택한 후 아래 그림에서 보인 것처럼 링커를 선택한다. 그리고 입력을 선택한 후 d3d9.lib를 추가 종속성에 추가한다.



Direct3D Programming

Taking the Game Full Screen

- Fullscreen 윈도우 프로그램으로 만들기
- Fullscreen을 만들려면 윈도우 스타일을 바꾼다.

```
wndHandle = CreateWindow("DirectXExample",
                         "DirectXExample",
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT,
                         CW_USEDEFAULT,
                         640,
                         480,
                         NULL,
                         NULL,
                         hInstance,
                         NULL);
```

Change to
WS_EX_TOPMOST | WS_POPUP | WS_VISIBLE

Taking the Game Full Screen

- initDirect3D함수에서 D3DPRESENT_PARAMETERS의 BackBufferFormat과 Windowed를 바꾼다.

```
if (bFullScreen)
{
    d3dpp.BackBufferFormat = D3DFMT_X8R8G8B8;
    d3dpp.Windowed = FALSE;
}
else
{
    d3dpp.BackBufferFormat = D3DFMT_UNKNOWN;
    d3dpp.Windowed = TRUE;
}
```

Represents 32bit format
8bit red, 8bit green, 8bit blue,
and 8bits are unused

Just use the current setting

Video Modes and Formats

- 비디오카드의 정보 보기
- IDirect3D9::GetAdapterIdentifier
  - IDirect3D9 인터페이스가 인스턴스화 된 시점에서 시스템에 존재하고 있던 물리적인 디스플레이 어댑터를 기술한다.

```
HRESULT GetAdapterIdentifier(UINT Adapter, DWORD Flags,
                             D3DADAPTER_IDENTIFIER9 *pIdentifier);
```

  - return값은
    - 성공했을 경우, D3D_OK를 돌려준다
    - Adapter의 값이 범위 외의 경우, Flags가 인식 불가능한 인자인 경우, 또는 pIdentifier가 NULL 혹은 쓰기 불가능한 메모리를 가리키고 있는 경우는 D3DERR_INVALIDCALL를 돌려준다.

Video Modes and Formats

- parameter
  - Adapter
    - 비디오카드 디스플레이 어댑터를 나타내는 서수
    - D3DADAPTER_DEFAULT는 항상 1차 디스플레이 어댑처이다.
    - 최소값은 0이고 최대값은 IDirect3D9::GetAdapterCount로 돌려받는 값으로부터 1을 뺀 값
  - Flags
    - D3DADAPTER_IDENTIFIER9 의 WHQLLevel 멤버를 설정하는 flag
    - D3DENUM_WHQL_LEVEL 또는 0 을 설정. 디폴트값은 0 .
    - D3DENUM_WHQL_LEVEL 를 지정 하면, 이 호출로 인터넷에 접속해, Microsoft® Windows® Hardware Quality Labs (WHQL)의 새로운 증명서를 다운로드할 수 있다.
  - pIdentifier
    - D3DADAPTER_IDENTIFIER9 구조체의 포인터로 어댑처를 기술하는 정보가 저장된다.
    - 시스템에 있는 어댑터의 수 이상의 값을 Adapter로 지정하면 이 구조체는 0이 설정된다.

Getting the Display Modes for an Adapter

- IDirect3D9::GetAdapterModeCount
  - 어댑터로 이용 가능한 디스플레이 모드의 수를 돌려준다

UINT GetAdapterModeCount(UINT Adapter, D3DFORMAT Format);

  - return값은 지정한 어댑터의 디스플레이 모드의 수를 돌려준다
  - parameters
    - Adapter
      - 디스플레이 어댑터를 나타내는 서수.
      - D3DADAPTER_DEFAULT는 항상 1차 디스플레이 어댑터이다
    - Format
      - D3DFORMAT를 사용하는 표면 타입의 포맷을 식별한다.
      - 유효한 포맷을 조사하려면 IDirect3D9::EnumAdapterModes를 사용한다.

Getting the Display Modes for an Adapter

- IDirect3D9::EnumAdapterModes
  - 이용 가능한 어댑터 모드 D3DDISPLAYMODE를 열거한다.

HRESULT EnumAdapterModes(UINT Adapter, D3DFORMAT Format,
UINT Mode, D3DDISPLAYMODE *pMode);

  - return값은
    - 이 어댑터로 장치를 사용할 수 있는 경우는 D3D_OK를 돌려준다
    - Adapter가 시스템의 디스플레이 어댑터의 수 이상의 경우, D3DERR_INVALIDCALL를 돌려준다
    - 표면 포맷이 지원되지 않은지, 지정된 포맷으로 하드웨어 가속화를 사용할 수 없는 경우는 D3DERR_NOTAVAILABLE를 돌려준다

Getting the Display Modes for an Adapter

- IDirect3D9::EnumAdapterModes
  - parameters
    - Adapter
      - 열거하는 디스플레이 어댑터를 나타내는 서수
      - D3dADAPTER_DEFAULT는 항상 1차 디스플레이 어댑터이다
    - Format
      - 이용 가능한 픽셀 포맷.
      - 유효한 포맷은 D3DFMT_X8R8G8B8, D3DFMT_A8R8G8B8, D3DFMT_A2R10G10B10, D3DFMT_X1R5G5B5, D3DFMT_A1R5G5B5, D3DFMT_R5G6B5
    - Mode
      - 어댑터의 모드를 나타낸다. 0~ GetAdapterModeCount가 돌려주는 값보다 1을 뺀 값까지의 범위에서 디스플레이 모드의 인덱스를 설정할 필요가 있다
    - pMode
      - 디스플레이 모드의 배열. 각 요소는 D3DDISPLAYMODE 형이다. width, height, refresh rate, format 등의 정보를 제공한다.

References

- http://www.monstertv.co.kr/board/content.asp?bd=b_codec&num=5&page=1&kind=&keyword= DirectX 소개
- http://www.gameis.org/Korean/Game_Dx/About-DX.htm DirectX 소개
- http://telnet.or.kr/directx/graphics/reference/d3d/d3dreference.htm Direct3D C/C++ reference