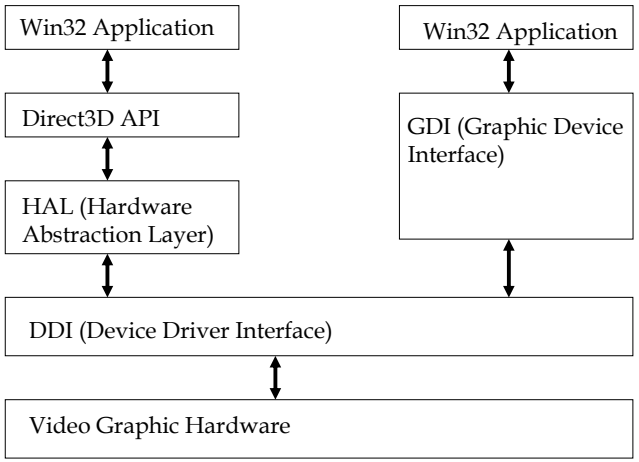
Direct3D Initialization

305890
2008년 봄학기
3/19/2008
박경신

Initializing Direct3D

- Direct3D 개요
- COM
- 약간의 준비
- Direct3D 초기화하기
- 예제

DDI, HAL, GDI 관계도



Direct3D Overview

- Direct3D
  - 3D 가속 하드웨어를 이용해 3D 세계를 표현할 수 있도록 하는 저수준 그래픽 API
  - Application->Direct3D->HAL->그래픽 장치(graphic device)
    - HAL (Hardware Abstraction Layer)는 그래픽 카드마다 다른 처리방식을 해결. HAL에 포함되어 있는 기능들은 장치마다 다름.
    - HAL에서 구현하지 않은 Direct3D함수를 호출하면 오류 발생
- REF (Reference Rasterizer)
  - 그래픽 장치에서는 지원하지 않는 Direct3D 기능을 제공하기 위함.
  - 모든 Direct3D API를 소프트웨어로 emulation하는 REF를 제공함.
  - REF 장치는 개발 목적으로 제공됨. DirectX SDK에만 포함되어 있음. 최종 사용자에게는 배포 불가.
- D3DDEVTYPE
  - HAL장치 – D3DDEVTYPE_HAL
  - REF 장치 – D3DDEVTYPE_REF

COM

- COM (Component Object Model)
  - COM은 DirectX를 프로그래밍 언어에 독립적으로 만들어주고 하위 호환성을 갖출 수 있게 하는 기술
  - COM object를 interface라고 부르며, C++ class와 비슷하게 이용
  - 모든 COM interface는 IUnknown COM interface에서 기능을 상속받음
    - COM object는 자신의 메모리 관리를 스스로 수행함
    - C++ new 키워드가 아닌 다른 COM interface의 method나 특수한 함수를 통해 COM interface의 pointer를 얻음
    - 마찬가지로, COM interface를 이용하는 작업이 모두 끝나면 C++ delete 키워드가 아니라 interface의 Release를 호출함
  - COM interface는 접두어로 "I"를 붙여서 명명
    - E.g., 표면을 나타내는 COM 인터페이스의 이름 IDirect3DSurface9

Direct3D Basics - Surface

- Surface
  - 표면은 Direct3D가 주로 2D image data 를 보관하는데 이용하는 픽셀의 행렬 (matrix of pixels) 임.
  - 표면의 너비 (width)와 높이 (height)는 pixel 단위로 계산
  - Pitch – 표면의 너비(width)의 byte 수
    - 하드웨어 구현에 따라 width*sizeof(pixelFormat)보다 더 클 수 있음

Direct3D Basics - Surface

- IDirect3DSurface9
  - 표면을 이용하도록 하는 interface. 표면에서 직접 데이터를 읽고 쓰는 몇 가지 method와 표면에 대한 정보를 얻는 method 등을 제공
  - IDirect3DSurface9 Method
    - LockRect – surface memory로의 pointer를 제공
    - UnlockRect – surface memory에 대한 작업이 끝난 후에 호출하여 잠금을 해제하도록 함
    - GetDesc – 표면에 대한 정보를 D3DSURFACE_DESC 구조체를 통해 얻음

```
typedef struct _D3DLOCKED_RECT {
        INT Pitch; // surface pitch
        void *pBits; // surface memory
} D3DLOCKED_RECT;
```

Direct3D Basics - Surface

- 표면을 잠그고 각각의 pixel을 빨간색으로 지정하는 예

```
// _surface가 IDirect3DSurface9 interface로의 pointer라고 가정
// 32-bit pixel format을 이용한다고 가정

// surface 정보를 얻음
D3DSURFACE_DESC surfaceDesc;
_surface->GetDesc(&surfaceDesc);
// surface pixel data로의 pointer를 얻음
D3DLOCKED_RECT lockedRect;
_surface->LockRect(&lockedRect, 0, 0);
// surface의 각 pixel을 대상으로 반복하여 pixel을 빨간색으로 지정
DWORD* imageData = (DWORD*) lockedRect.pBits;
for (int i=0; i<surfaceDesc.Height; i++) {
    for (int j=0; j<surfaceDesc.Width; j++) {
      // pitch는 byte 단위이며 DWORD당 4bytes이므로 4로 나눔
      int index = i * lockedRect.Pitch / 4 + j;
      imageData[index] = 0xffff0000;            // red
    }
}
_surface->UnlockRect();
```

Direct3D Basics - Multisampling

- Multisampling [그림 1.3 참고]
  - Pixel matrix로 image를 표현함에 따라 나타나는 거친 이미지를 부드럽게 함.
  - 일반적인 용도는 anti-aliasing을 위한 표면 멀티 샘플링
  - D3DMULTISMPLE_TYPE
    - 표면에 적용할 multisampling 레벨을 지정하는 열거형
    - D3DMULTISAMPLE_NONE – 멀티 샘플링을 지정하지 않는다
    - D3DMULTISAMPLE_[1~16]_SAMPLE – 1에서 16까지의 멀티 샘플링 레벨을 지정한다.
  - 멀티 샘플링은 application 속도를 매우 느려지게 함
  - IDirect3D9::CheckDeviceMultiSampleType으로 그래픽 카드의 지원 여부를 확인 할 필요가 있음

Direct3D Basics – Pixel format

- Pixel format
  - Surface나 texture를 만들기 위해서는 pixel format을 지정해야 한다.
- D3DFORMAT
  - D3DFMT_R8G8B8
    - 24-bit pixel format. 8-bit red, 8-bit green, 8-bit blue
  - D3DFMT_X8R8G8B8
    - 32-bit pixel format. 가장 왼쪽 8-bit은 이용되지 않음
  - D3DFMT_A8R8G8B8
    - 32-bit pixel format. 가장 왼쪽 8-bit은 alpha
  - D3DFMT_A16B16G16R16F
    - 64-bit floating point pixel format
  - D3DFMT_A32B32G32R32F
    - 128-bit floating point pixel format
  - 널리 이용되지 않는 포멧을 이용할 때는 보유한 카드에서 원하는 포멧을 지원하는지를 확인해야 한다.

Direct3D Basics – Memory pool

- Memory pool
  - Surface등 다양한 Direct3D 자원들은 여러 종류의 memory pool에 보관됨
- D3DPOOL
  - D3DPOOL_DEFAULT
    - 자원(e.g., video memory, AGP memory, system memory등)의 타입과 이용 방식에 가장 적합한 자원들을 메모리에 보관하도록 Direct3D에 요청함
    - 반드시 IDirect3DDevice9::Reset 호출 이후 초기화되어야 함
  - D3DPOOL_MANAGED
    - Managed pool에 보관된 자원은 Direct3D에 관리됨 (즉, 필요에 따라 자동으로 video memory나 AGP memory로 옮겨짐)
    - 자원의 복사본이 system memory내에 보관되어 application이 자원을 접근/수정할 때 이용하도록 하며, Direct3D는 자동으로 이를 video memory에 갱신함
  - D3DPOOL_SYSTEMMEM
    - 시스템 메모리 내에 보관될 자원을 지정함
  - D3DPOOL_SCRATCH
    - 시스템 메모리 내에 보관될 자원을 지정함. Direct3D가 자원을 접근할 수 없음. 그러나 두 pool 사이에 서로 복사하는 것은 가능함.

Direct3D Basics – Double buffering

- Double buffering [그림 1.5 참고]
  - Direct3D는 보통 2~3개의 표면을 하나의 켈렉션으로 관리하며 이를 스왑 체인 (swap chain)이라 부른다.
  - 스왑 체인과 페이지 플리핑(page flipping) 기술은 프레임 간의 부드러운 애니메이션을 제공하기 위한 것이다.
  - 더블 버퍼링
    - 전면 버퍼 – 이 버퍼의 내용물은 현재 모니터에서 보여지고 있다.
    - 후면 버퍼 – 현재 렌더링 처리 중인 프레임이 이 버퍼에 보관된다.
    - 후면 버퍼에서 렌더링을 수행하고 전면 버퍼 표면의 디스플레이가 완료되면 스왑 체인의 끝으로 돌아가 후면 버퍼를 전면 버퍼로 전환한다.

Direct3D Basics – Depth buffer

- Depth buffer [그림 1.6 참고]
  - Pixel의 depth 정보를 가지는 surface임. 물체의 pixel이 다른 pixel을 가리는지의 여부를 판단하기 위해 depth buffering (또는 z-buffering) 기법을 사용.
  - 렌더링된 이미지가 640x480 해상도를 가진다면 640x480개의 깊이 존재. 카메라와 가까운 pixel이 뒤쪽 pixel을 가리는 원리임.
  - Depth buffer의 포멧은 depth test의 정확도를 결정함. 24-bit도 충분함.
- Depth buffer
  - D3DFMT_D32 - 32-bit depth buffer
  - D3DFMT_D24S8 - 24-bit depth buffer, 8-bit stencil buffer
  - D3DFMT_D24X8 - 24-bit depth buffer
  - D3DFMT_D24X4S4 - 24-bit depth buffer, 4-bit stencil buffer
  - D3DFMT_D16 - 16-bit depth buffer

Direct3D Basics – 장치의 특성

- D3DCAPS9
  - Direct3D가 제공하는 모든 기능들을 대응시켜줌.
  - D3DCAPS9 instance에 대응되는 bit나 data member를 확인하면 장치가 특정기능을 제공하는 지를 알 수 있음.

```
// D3DCAPS9 instance가 초기화되었다고 가정
bool supportsHardwareVertexProcessing;

// 장치가 변환과 조명계산을 하드웨어로 처리할 수 있는지 여부
If (caps.DevCaps & D3DDEVCAPS_HWTRANSFORMANDLIGHT) {
   // bit가 켜져 있으므로 기능이 지원됨
   supportsHardwareVertexProcessing = true;
} else {
   supportsHardwareVertexProcessing = false;
}
```

Direct3D Initialization

- Direct3D의 초기화 과정
  1. IDirect3D9 interface로의 pointer를 얻음
  2. 기본 display adapter (즉, 기본 그래픽스 카드)가 하드웨어 vertex processing을 지원하는지 알아보기 위해 D3DCAPS9를 확인함
  3. IDirect3DDevice9 instance의 특성 지정을 위해 D3DPRESENT_PARAMETERS 구조체 instance를 초기화함
  4. 초기화된 D3DPRESENT_PARAMETERS에 따라 IDirect3DDevice9 객체를 형성

Initialization – IDirect3D9 얻기

- IDirect3D9
  - 3D graphics를 화면에 표시하는데 이용될 물리적 하드웨어 장치의 C++객체임
  - 그래픽스 카드가 제공하는 기능, display mode, format등의 특성에 대한 정보를 얻는 과정

```
IDirect3D9* d3d9;
d3d9 = Direct3DCreate9(D3D_SDK_VERSION);
```

Initialization – HW Vertex Processing 확인

- IDirect3DDevice9 객체 생성시
  - 반드시 원하는 vertex processing mode를 지정해야 함.
- D3DCAPS9 instance를 초기화
  - Default display adapter의 특성으로 D3DCAPS9 구조체를 채움

```
HRESULT IDirect3D9::GetDeivceCaps (
    UINT Adapter,          // 특성을 얻고자 하는 physical display adapter
    D3DDEVTYPE DeviceType, // 이용할 장치 type을 지정
    D3DCAPS9 *pCaps
);
```

보통 D3DDEVTYPE_HAL로 지정

```
D3DCAPS9 caps;
d3d9->GetDeviceCaps(D3DADAPTER_DEFAULT, deviceType, &caps);
int vp =0;
if (caps.DevCaps & D3DDEVCAPS_HWTRANSFORMANDLIGHT) {
    vp = D3DCREATE_HARDWARE_VERTEXPROCESSING;
} else {
    vp = D3DCREATE_SOFTWARE_VERTEXPROCESSING;
}
```

Initialization – D3DPRESENT_PARAM 설정

- D3DPRESENT_PARAMETERES 구조체 instance채우기
  - IDirect3DDevice9 객체의 성격을 결정함

```
typedef struct _D3DPRESENT_PARAMETERS {
    UINT BackBufferWidth;          // 후면버퍼 너비
    UNIT BackBufferHeight;         // 후면버퍼 높이
    D3DFORMAT BackBufferFormat; // 후면버퍼 픽셀 포멧
    UINT BackBufferCount;          // 이용할 후면버퍼 수 – 보통 "1" 지정
    D3DMULTISAMPLE_TYPE MultiSampleType; // 후면버퍼에 쓸 멀티샘플링타입
    DWORD MultiSampleQuality;  // 멀티샘플링 레벨
    D3DSWAPEFFECT SwapEffect; // 스와핑 방법 – 보통 D3DSWAPEFFECT_DISCARD
    HWND hDeviceWindow;            // 서비스와 윈도우 핸들
    BOOL Windowed;                 // 윈도우 모드로 실행할 때는 true
    BOOL EnableAutoDepthStencil; // 자동으로 깊이/스텐실 버퍼 관리하려면 true
    D3DFORMAT AutoDepthStencilFormat; // 깊이/스텐실 버퍼 포멧
    DWORD Flags;                   // 0 (플래그 없음)
    UINT FullScreen_RefreshRateInHz; // 재생율 지정
    UNIT PresentationInterval;     // 시연간격 D3DPRESENT_INTERVAL_DEFAULT
} D3DPRESENT_PARAMETERS;
```

Initialization – IDirect3DDevice9 생성

- IDirect3DDevice9 객체생성

```
IDirect3DDevice9::CreateDevice(
    // 객체와 대응될 physical display adapter를 지정
    UINT Adapter,
    // HW 장치 (D3DDEVTYPE_HAL)
    // 또는 래퍼런스 래스터기 장치 (D3DDEVTYPE_REF)
    D3DDEVTYPE DeviceType,
    // 장치와 연결될 window handle
    HWND hFocusWindow,
    // 정점처리를 어디서 할 지 결정하는 값
    // 하드웨어 처리 (D3DCREATE_HARDWARE_VERTEXPROCESSING)
    // 또는 소프트웨어 처리 (D3DCREATE_SOFTWARE_VERTEXPROCESSING)
    DWORD BehaviorFlags,
    // 초기화된 D3DPRESENT_PARAMETERS instance 지정
    D3DPRESENT_PARAMETERS *pPresentationParameters,
    // 생성된 장치 return
    IDirect3DDevice9 **ppReturnedDeviceInterface
)
```

Example – d3dUtility.h

```
#include <d3dx9.h>
#include <string>
namespace d3d {
  bool InitD3D( //주응용윈도 초기화, Direct3D초기화.
    HINSTANCE hInstance,  // [in] Application instance.
    int width, int height, // [in] Backbuffer dimensions.
    bool windowed, // [in] Windowed (true) or full screen (false).
    D3DDEVTYPE deviceType, // [in] HAL or REF
    IDirect3DDevice9** device);// [out] The created device.
  int EnterMsgLoop( //응용의 message loop를 포장함.
    bool (*ptr_display)(float timeDelta)); //그리기 코드를 구현할 함수.
  LRESULT CALLBACK WndProc( //주응용윈도를 위한 윈도 procedure선언.
    HWND hwnd,
    UINT msg,
    WPARAM wParam,
    LPARAM lParam);

  template<class T> void Release(T t) { //COM interface를 release.
    if( t ) {
      t->Release();
      t = 0;    }
  }
  template<class T> void Delete(T t) { //storage객체를 제거.
    if( t ) {
      delete t;
      t = 0;    }
  }
}
```

Example – d3dUtility.cpp

```
#include "d3dUtility.h"
bool d3d::InitD3D(
  HINSTANCE hInstance,    int width, int height,
  bool windowed,    D3DDEVTYPE deviceType,
  IDirect3DDevice9** device) { //Create main application window.
  WNDCLASS wc;
  wc.style         = CS_HREDRAW | CS_VREDRAW;
  wc.lpfnWndProc   = (WNDPROC)d3d::WndProc;
  wc.cbClsExtra    = 0;    wc.cbWndExtra    = 0;
  wc.hInstance     = hInstance;
  wc.hIcon         = LoadIcon(0, IDI_APPLICATION);
  wc.hCursor       = LoadCursor(0, IDC_ARROW);
  wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
  wc.lpszMenuName  = 0;    wc.lpszClassName = "Direct3D9App";
  if( !RegisterClass(&wc) ) {
    ::MessageBox(0, "RegisterClass() - FAILED", 0, 0);
    return false;
  }
  HWND hwnd = 0;
  hwnd = ::CreateWindow("Direct3D9App", "Direct3D9App",
    WS_EX_TOPMOST,    0, 0, width, height,
    0 /*parent hwnd*/, 0 /* menu */, hInstance, 0 /*extra*/);
  if( !hwnd ) {
    ::MessageBox(0, "CreateWindow() - FAILED", 0, 0);
    return false;
  }
  ::ShowWindow(hwnd, SW_SHOW);
  ::UpdateWindow(hwnd);
```

Example – d3dUtility.cpp

```
  // Init D3D : Step 1,2,3,4.
  HRESULT hr = 0;
  // Step 1: Create the IDirect3D9 object.
  IDirect3D9* d3d9 = 0;  d3d9 = Direct3DCreate9(D3D_SDK_VERSION);
  if( !d3d9 ) {
    ::MessageBox(0, "Direct3DCreate9() - FAILED", 0, 0);
    return false;  }
  // Step 2: Check for hardware vp.
  D3DCAPS9 caps;
  d3d9->GetDeviceCaps(D3DADAPTER_DEFAULT, deviceType, &caps);
  int vp = 0;
  if( caps.DevCaps & D3DDEVCAPS_HWTRANSFORMANDLIGHT )
    vp = D3DCREATE_HARDWARE_VERTEXPROCESSING;
  else  vp = D3DCREATE_SOFTWARE_VERTEXPROCESSING;
  // Step 3: Fill out the D3DPRESENT_PARAMETERS structure.
  D3DPRESENT_PARAMETERS d3dpp;
  d3dpp.BackBufferWidth  = width;  d3dpp.BackBufferHeight = height;
  d3dpp.BackBufferFormat = D3DFMT_A8R8G8B8;
  d3dpp.BackBufferCount = 1;
  d3dpp.MultiSampleType           = D3DMULTISAMPLE_NONE;
  d3dpp.MultiSampleQuality         = 0;
  d3dpp.SwapEffect                 = D3DSWAPEFFECT_DISCARD;
  d3dpp.hDeviceWindow = hwnd;  d3dpp.Windowed = windowed;
  d3dpp.EnableAutoDepthStencil     = true;
  d3dpp.AutoDepthStencilFormat     = D3DFMT_D24S8;
  d3dpp.Flags                      = 0;
  d3dpp.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;
  d3dpp.PresentationInterval       = D3DPRESENT_INTERVAL_IMMEDIATE;
```

Example – d3dUtility.cpp

```
  // Step 4: Create the device.
  hr = d3d9->CreateDevice(
    D3DADAPTER_DEFAULT, // primary adapter
    deviceType,         // device type
    hwnd,               // window associated with device
    vp,                 // vertex processing
    &d3dpp,             // present parameters
    device);            // return created device
  if( FAILED(hr) ) {
    // try again using a 16-bit depth buffer
    d3dpp.AutoDepthStencilFormat = D3DFMT_D16;
    hr = d3d9->CreateDevice(
      D3DADAPTER_DEFAULT,
      deviceType,
      hwnd,
      vp,
      &d3dpp,
      device);
    if( FAILED(hr) ) {
      d3d9->Release(); // done with d3d9 object
      ::MessageBox(0, "CreateDevice() - FAILED", 0, 0);
      return false;
    }
  }
  d3d9->Release(); // done with d3d9 object
  return true;
}
```

Example – d3dUtility.cpp

```
int d3d::EnterMsgLoop( bool (*ptr_display)(float timeDelta) )
{
  MSG msg;
  ::ZeroMemory(&msg, sizeof(MSG));

  static float lastTime = (float)timeGetTime();

  while(msg.message != WM_QUIT) {
    if(::PeekMessage(&msg, 0, 0, 0, PM_REMOVE))
    {
      ::TranslateMessage(&msg);
      ::DispatchMessage(&msg);
    }
    else { //게임 코드를 실행.
      float currTime  = (float)timeGetTime();
      float timeDelta = (currTime - lastTime)*0.001f;

      ptr_display(timeDelta); //display 함수를 호출한다.

      lastTime = currTime;
    }
  }
  return msg.wParam;
}
```

Example Framework

- D3D 초기화
  - 공통 함수 (WndProc, WinMain 외)
    - Bool Setup() – 자원의 할당이나 장치 특성의 확인, 응용 상태의 설정 등 실행을 위한 준비작업
    - Void Cleanup() – Setup()에서 할당된 자원을 해제하는 작업
    - Bool Display(float timeDelta) – 모든 그리기 코드를 구현. 물체의 위치 수정 등 매 frame마다 일어나야 하는 작업을 수행. timeDelta는 각 프레임의 경과 시간으로 초당 프레임과의 애니메이션 동기화에 이용됨.

Example Framework – D3D Init

```
#include "d3dUtility.h"
// Globals
IDirect3DDevice9* Device = 0;
// Framework Functions
bool Setup() {
  // Nothing to setup in this sample.
  return true;
}
void Cleanup() {
  // Nothing to cleanup in this sample.
}
bool Display(float timeDelta) {
  if( Device ) { // Only use Device methods if we have a valid device.
    //Instruct the device to set each pixel on the back buffer black -
    //D3DCLEAR_TARGET: 0x00000000 (black) - and to set each pixel on
    //the depth buffer to a value of 1.0 - D3DCLEAR_ZBUFFER: 1.0f.
    Device->Clear(0 /* DWORD Count : pRects 배열 내 자각형의 수 */,
        0 /* const D3DRECT* pRects : clear할 화면 사각형의 행렬 */,
        D3DCLEAR_TARGET|D3DCLEAR_ZBUFFER /* DWORD Flags: 어떤 표면을 */
          /* clear할지를 지정. D3DCLEAR_TARGET/ZBUFFER/STENCIL. */,
        0x00000000 /* D3DCOLOR Color : render 대상을 clear할 색상 */,
        1.0f /* float Z: Z-buffer에 지정하고자 하는 값 */,
        0 /* DWORD Stencil : stencil buffer에 지정하고자 하는 값 */);

    // Swap the back and front buffers.
    Device->Present(0, 0, 0, 0);
  }
  return true;
```

Example Framework – D3D Init

```
LRESULT CALLBACK d3d::WndProc(HWND hwnd, UINT msg,
       WPARAM wParam, LPARAM lParam) {
  switch( msg ) {
  case WM_DESTROY:
    ::PostQuitMessage(0);
    break;
  case WM_KEYDOWN:
    if( wParam == VK_ESCAPE )
      ::DestroyWindow(hwnd);
    break;
  }
  return ::DefWindowProc(hwnd, msg, wParam, lParam);
}
int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE prevInstance,
           PSTR cmdLine, int showCmd) {
  if(!d3d::InitD3D(hinstance,
    640, 480, true, D3DDEVTYPE_HAL, &Device)) {
    ::MessageBox(0, "InitD3D() - FAILED", 0, 0);
    return 0;
  }
  if(!Setup()) {
    ::MessageBox(0, "Setup() - FAILED", 0, 0);
    return 0;
  }
  d3d::EnterMsgLoop( Display );
  Cleanup();
  Device->Release();
  return 0;
```