2D Game Programming 01

afewhee@gmail.com

1. Direct3D Device

- Graphic 출력
  - Window API ➔ GDI의 DC (Device Context)
    - GDI는 보편성을 위해 설계
    - 소프트웨어 렌더링 ➔ Game의 경우 화면 출력 속도를 요구하는데 많이 부족 ➔ 하드웨어 가속을 지원하는 라이브러리 필요
  - Game
    - DirectDraw ➔ Microsoft DirectX 7.0의 2D전용 라이브러리
    - Direct3DDevice ➔ Microsoft DirectX 8.0 이후 버전에서의 3D전용 라이브러리. 2D는 3D로 처리
    - OpenGL ➔ 가장 보편적인 3D Graphic Library

- 윈도우 프로그램 순서
  - main() → GetModuleHandle() → RegisterClass() → CreateWindow()
    → GetMessage() → TranslateMessage()→ DispatchMessage()
    → WindowProcedure() → DestroyWindow() → UnRegisterClass()

- 게임 프로그램 순서
  - main() → ... → CreateWindow() → CreateDevice()
    → PeekMessage() → TranslateMessage() → DispatchMessage()
    → GameProcedure() → ReleaseDevice() → DestroyWindow() → ...

- DC: 윈도우 핸들 ➔ GetDC(), BeginPaint()

- Direct3D를 이용한 2D:
  - 윈도우 핸들 ➔ D3D(Direct3D) ➔ Direct3DDevice ➔ D3DXSprite

1. Direct3D Device와 Sprite

- 필요한 헤더 파일 / 라이브러리
  - Direct3D Device
    - #pragma comment(lib, "d3d9.lib")
    - #include <d3d9.h>
  - D3DXSprite
    - #pragma comment(lib, "d3dx9.lib")
    - #include <d3dx9.h>
- 필수 객체
  - LPDIRECT3D9 m_pD3D = NULL;
  - LPDIRECT3DDEVICE9 m_pd3dDevice = NULL;
  - LPD3DXSPRITE m_pd3dSprite = NULL;
- D3D, Device, Sprite 객체 생성
  - D3D: Direct3DCreate9(D3D_SDK_VERSION)
  - Device: pD3D->CreateDevice()
  - Sprite: D3DXCreateSprite(pDevice)
- 객체 소멸:
  - COM을 상속받는 DirectX의 모든 객체는 Release() 함수로 해제

- Direct3D Device 생성하기
  - 디바이스를 생성하기 위해서는 프리젠트 파라메터 구조체가 필요
  - 전부 0으로 설정한 후에 일부만 변수 설정

```
D3DPRESENT_PARAMETERS d3dpp;

ZeroMemory( &d3dpp, sizeof(d3dpp) );

d3dpp.Windowed               = m_bWindow;
d3dpp.SwapEffect             = D3DSWAPEFFECT_DISCARD;
d3dpp.EnableAutoDepthStencil = TRUE;
d3dpp.AutoDepthStencilFormat = D3DFMT_D16;


// D3DADAPTER_DEFAULT: 대부분의 그래픽카드는 모노 듀얼일 경우 이부분을 수정
// D3DDEVTYPE_HAL : 하드웨어 가속(가장 큰 속도)을 받을 것인가.. 하드웨어 지
// 원이 없을 경우 D3D는 소프트웨어로 이를 대체 할 수 있다.

if( FAILED( m_pD3D->CreateDevice( D3DADAPTER_DEFAULT
                                , D3DDEVTYPE_HAL
                                , m_hWnd
                                , D3DCREATE_MIXED_VERTEXPROCESSING
                                , &d3dpp
                                , &m_pd3dDevice ) ) )
{
      if( FAILED( m_pD3D->CreateDevice( D3DADAPTER_DEFAULT
                                      , D3DDEVTYPE_HAL
                                      , m_hWnd
                                      , D3DCREATE_SOFTWARE_VERTEXPROCESSING
                                      , &d3dpp
                                      , &m_pd3dDevice ) ) )
      {
            m_pD3D->Release();
            return -1;
      }
}
```

2. 화면 출력

- 화면 출력 순서는 반드시 지켜야 함.
- 모든 장면은 후면 버퍼에 렌더링

- 후면 버퍼 지우기:
  - pDevice->Clear()

- 3D 장면의 연출 시작:
  - pDevice->BeginScene()

- 2D 스프라이트 객체를 이용한 장면 연출 시작:
  - pSprite->Begin()

- 2D Draw:
  - pSprite->Draw(...)

- 2D 스프라이트 객체 장면 연출 끝:
  - pSprite->End()

- 3D 장면 연출 끝:
  - m_pd3dDevice->EndScene()

- 후면 버퍼와 전면 버퍼 교체:
  - pDevice->Present()

- 텍스처: 3D에서는 이미지 보다 텍스처라는 용어를 많이 사용

- 텍스처 객체 생성
  - 2D 전용 텍스처 함수는 없음
  - 3D에서 사용하는 D3DXCreateTextureFromFileEx() 함수
  - 2D에 맞는 옵션 조정이 필요

- 텍스처 생성 Ex)

```
LPDIRECT3DTEXTURE9 pTx1 = NULL;
D3DXIMAGE_INFO      pImgInf;
...
if( FAILED( D3DXCreateTextureFromFileEx(
        pDevice                 // 디바이스 포인터
        , "텍스처 파일이름"
        , D3DX_DEFAULT
        , D3DX_DEFAULT
        , 1                     // 밉 레벨(2D에서는 반드시 1)
        , 0
        , D3DFMT_UNKNOWN
        , D3DPOOL_MANAGED       // 가장 안전한 메모리 풀
        , 0x0000001             // 필터링
        , 0x0000001             // 밉 필터링
        , 0x00FFFFFF            // 컬러 키
        , &pImgInf              // 텍스처 인포메이션
        , NULL
        , &pTx1                 // 텍스처 포인터
{
        return -1;
}
...
pTx1->Release();
pTx1 = NULL;
```

- 3개의 배경 이미지, 4개의 버튼 이미지를 이용해서 게임의 시작, 플레이 화면, 종료 화면을 구성하시오.

Advanced ➔ 각각의 화면은 시작 클래스, 플레이 클래스, 종료 클래스로 만들어서 화면에 출력하시오.