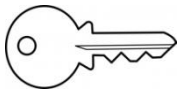
게임프로그래밍

D2D 개요

박종승

Dept. of CSE, Incheon Nat. Univ.
jong@inu.ac.kr
http://ecl.inu.ac.kr

목차

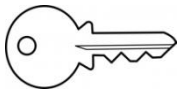


DirectX

- DirectX SDK
  - DirectX Graphics
    - Direct3D
    - Direct2D
    - DirectWrite
    - Windows Imaging Component (WIC)
  - DirectX Audio
    - DirectSound (deprecated)
    - XAudio2
  - DirectX Input
    - DirectInput (deprecated)
    - XInput

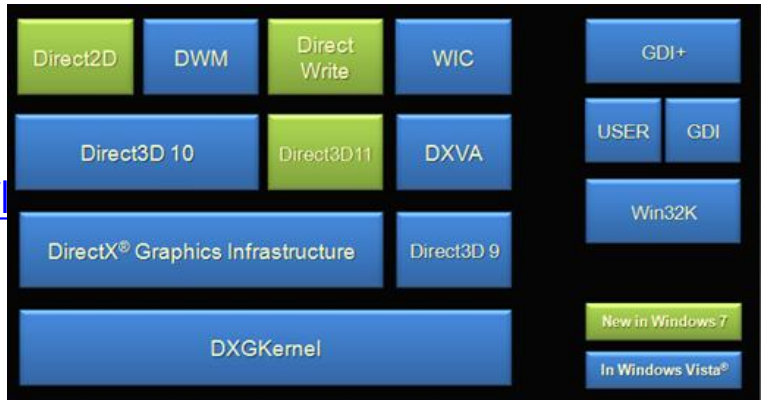
DirectX'

- Abbreviation: (D=Direct) : DX=DirectX, Direct2D=D2D, Direct3D=D3D
- Old : DirectDraw, DirectPlay, DirectMusic, DirectShow
- DirectX SDK Version-ups
  - 버전 10 : 2016.12, Windows Vista, 셰이더모델 4.0
  - 버전 10.1 : 2008.03 , 셰이더모델 4.1
  - 버전 11 : 2009.10, Windows 7 , 셰이더모델 5.0
  - 버전 11.1 : 2012
  - 버전 11.2 : 2013 , 셰이더모델 5.1
  - 버전 11.3 & 버전 12 : 1507
  - 버너 11.4 & 버전 12 update : 1511
  - 버전 11.4 update & 버전 12 update : 1607 (셰이더모델 6.0), 1703, 1709, 1803, 1809
  - 버전 12.1 : (셰이더모델 6.4) Windows 10 1903(April 2019)



Direct2D?

- Direct2D?
  - 새로운 2D 그래픽스 API
    - 하드웨어 가속, 즉시모드(immediate mode)
    - 2D 기하(geometry), 비트맵(bitmap), 텍스트 등의 고성능 고품질 렌더링을 제공함
    - GDI, GDI+, Direct3D와 상호운용
  - More info
    - https://msdn.microsoft.com/en-us/l



- 대상 개발자
  - 대규모의 native 응용 개발자
  - 응용개발자들을 위한 툴킷이나 라이브러리의 개발자
  - Direct3D 그래픽을 사용하면서, (메뉴, UI요소, HUD를 위해) 간단하고 고성능의 2D 및 텍스트 렌더링이 필요한 개발자
- 실행 요구사항
  - Windows 7 이상

참고: 즉시모드와 보류모드의 차이점

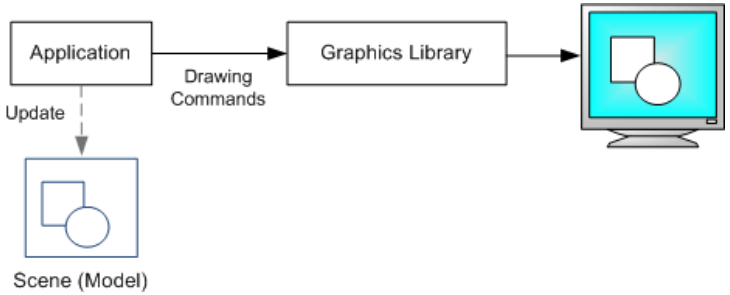
- 즉시모드(immediate mode)
  - 3D 그래픽HW에 대한 복잡하지만 빠른 low-level interface를 제공함
  - 렌더링 프레임워크(DirectX)에 렌더할 것을 명시적으로 명령하여, 자신의 데이터를 직접 렌더함.

- 보류모드(retained mode)
  - 3D 그래픽HW에 대한 사용하기 편리하지만 느리고 융통성이 없고 비대한 higher-level interface를 제공함
  - 렌더링에 대한 명시적인 제어는 없고, 단지 렌더링을 위한 준비들을 내부 데이터 구조에 채워둠.
  - 렌더러는 가능한 시간에 자신의 방식으로 렌더함.
  - 렌더링 대상은 주로 scene graph로 표현됨.

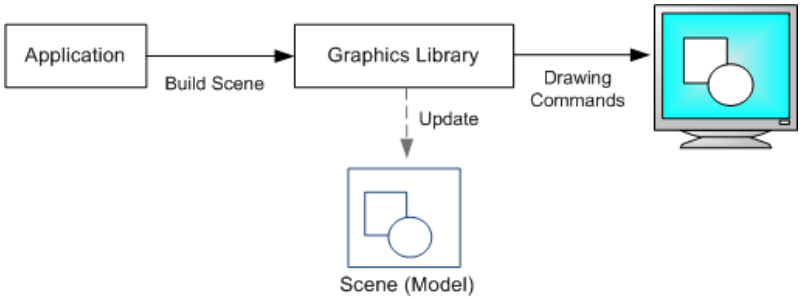
- 어떤 것이 더 좋은가?
  - 즉시모드!!

참고: 즉시모드와 보류모드의 차이점'

- 즉시모드(immediate mode)
  - E.g. Direct2D

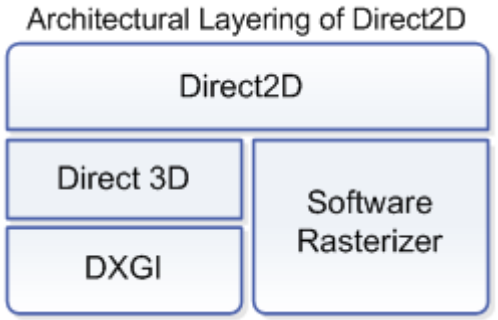


- 보류모드(retained mode)
  - E.g. Windows Presentation Foundation (WPF)



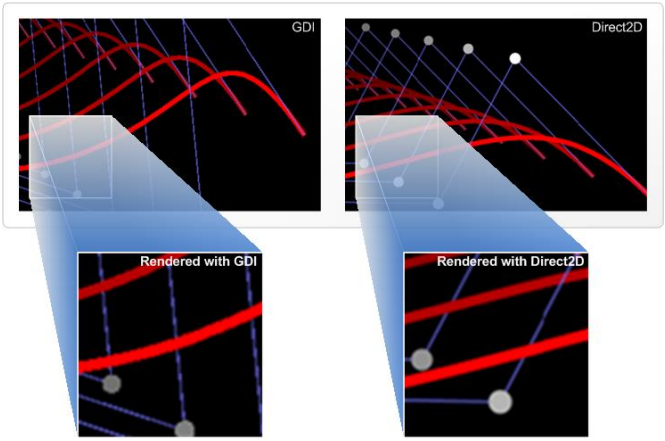
Direct2D에 대해서

- 왜 Direct2D 인가?
  - 시각적으로 높은 수준의 만족도를 제공함
  - 부착된 그래픽 하드웨어의 성능에 적응적인 2D 렌더링 코딩이 가능
  - Direct3D를 사용하기에 벅찬 GDI/GDI+ 개발자를 위한 경우
  - Direct3D 응용에 고품질 2D 그래픽스를 추가하기 위한 경우
- 최대의 가용성을 가진 고성능 구현
  - Direct3D 10.1 API를 사용함
    - 하드웨어 가속 렌더링
  - 하드웨어 가속이 어려운 경우
    - 소프트웨어 rasterizer를 사용.
      - GDI+보다 더 좋음.

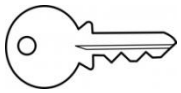


Direct2D에 대해서'

- 시각적 품질
  - GDI보다 우월함
  - Per-primitive anti-aliasing 기법을 사용함
    - 사용하지 않도록 할 수도 있음 (GDI 흉내)
  - 투명도 및 알파블랜딩을 완전하게 지원함
- 상호운용성
  - GDI, GDI+, Direct3D와 함께 사용 가능
    - 이미 개발된 응용의 일부만을 Direct2D로 바꾸어도 됨
  - DirectWrite, WIC(Windows Imaging Component)의 사용도 쉬워짐



GDI사용(aliased) vs Direct2D사용(antialiased)



D2D 실행 환경 준비

- Visual Studio 설치
  - 최신버전 권장, C/C++

- 참고:
  - 독립적으로 배포되는 DirectX SDK 설치
    - 다운로드 위치: http://msdn.microsoft.com/directx
    - 독립적으로 배포되는 마지막 버전: "June 2010"
    - 경로설정
      - DXSDK 경로 (DXSDK를 설치한 후 환경변수가 자동으로 추가됨)
        - » 예: DXSDK_DIR=C:₩Program Files₩Microsoft DirectX SDK (June 2010)₩
  - Windows 8 부터 DirectX SDK는 Windows SDK에 포함되어 배포
    - Windows 8 (2011년부터) 부터는 별도 설치과정 필요없음
      - DirectX 11.1, Direct2D, DirectWrite, DXGI 1.2, WDDM 1.2, DirectXMath, Feature Level 11.1 devices, XINPUT 1.4, XAUDIO 2.8, HLSL compiler
    - Visual Studio 2012 이상을 설치하면 최신 Windows SDK가 자동 설치됨
    - Visual Studio 2010 사용자는 최신 Windows SDK를 별도 설치해야 함
    - 참고: (link)(link)(link)

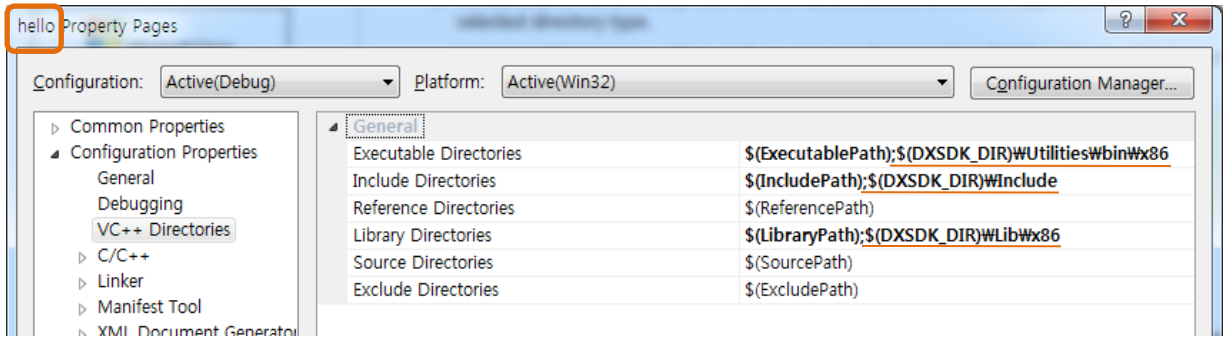
참고: Visual Studio 경로 설정 (독립 배포 버전의 경우)

- Visual Studio에서의 환경설정
  - 경로목록의 종류
    - Executable/Include/Library Directories 등을 설정할 수 있음
    - 경로 구분은 ";"으로 함
  - 각 프로젝트 당 경로목록 설정
    - "Solution Explorer"에서, 메뉴 "Project"≫"Properties"를 선택한 후,
    - 대화상자의 "Configuration Properties"≫"VC++ Directories"를 클릭한 후,
    - 오른쪽 목록에서 해당 목록을 클릭하고 "<Edit...>"를 눌러 추가
  - 각 사용자 당 경로목록 설정
    - 메뉴 "View"≫"Property Manager"를 선택한 후,
    - "Property Manager"에서 프로젝트 노드를 펼치고,
    - "Debug | Win32" 하부의 "Microsoft.Cpp.Win32.user"를 더블클릭하고,
    - 대화상자의 "Common Properties"≫"VC++ Directories"를 클릭한 후,
    - 오른쪽 목록에서 해당 목록을 클릭하고 "<Edit...>"를 눌러 추가

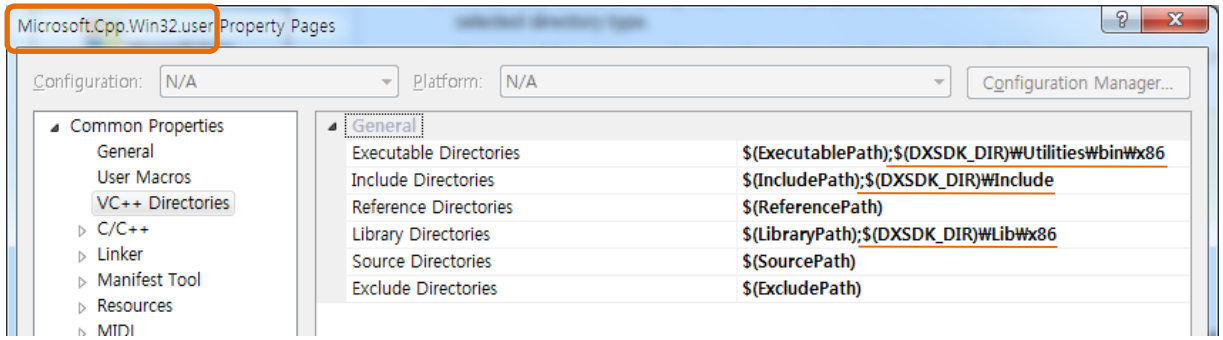
참고: Visual Studio 경로 설정 (독립 배포 버전의 경우)'

- 필요에 따라서 경로 추가
  - 예시: DirectX SDK 관련 경로 추가
    - Executable Directories: $(DXSDK_DIR)₩Utilities₩bin₩x86;$(ExecutablePath)
    - Include Directories: $(DXSDK_DIR)₩Include;$(IncludePath)
    - Library Directories: $(DXSDK_DIR)₩Lib₩x86;$(LibraryPath)

사용자 당



프로젝트 당



D2D 사용 준비

- 헤더 파일
  - d2d1.h
    - D2D API의 핵심 헤더 파일임. 반드시 포함해야 함.
    - 내부적으로 d2dbasetypes.h, d2derr.h 를 포함하고 있음
  - d2d1helper.h
    - 도움 함수들 및 기타

```
#include <d2d1.h>
#include <d2d1helper.h>

#include <dwrite.h>
#include <wincodec.h>
```

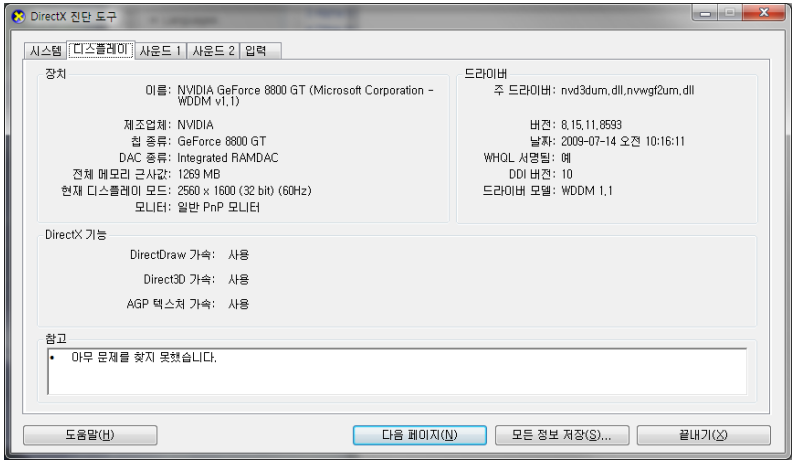
- 라이브러리 파일
  - d2d1.lib
  - dwrite.lib, WindowsCodecs.lib
  - 설정: 프로젝트 속성 > 링크 > 입력 > 추가 종속성

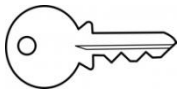
D2D 사용 준비'

- 문서
  - Visual Studio 2010부터 Internet Explorer 창으로 도움말을 보여줌
    - Visual Studio의 "Help"≫"View Help"를 클릭하면 온라인 MSDN 사이트를 보여줌
  - 문서 계층구조:
    - "MSDN Library" ≫ "Windows Development" ≫ "DirectX Graphics and Gaming" ≫ "Direct2D"
  - MSDN Direct2D 문서: (link: Direct2D)
    - https://docs.microsoft.com/ko-kr/windows/desktop/Direct2D

D2D 사용 준비”

- 참고: 드라이버 추천
  - 하드웨어 렌더링에서 최선의 성능을 위해서는:
  - Windows 7용 WDDM 1.1 드라이버를 사용할 것
    - WDDM(Windows Display Driver Model): Windows Vista에서부터 새로 만들어진 그래픽 카드 드라이버 모델
    - 확인 방법: 시작메뉴의 실행 대화상자에서 “DxDiag”를 실행
    - Windows 7/8/8.1/10의 WDDM 버전은 각각 1.1/1.2/1.3/2.0





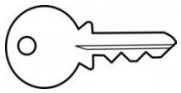
D2D에서의 기본 타입들

- 데이터 타입
  - 기본 타입
    - UINT32: 타입 접미사 "U"에 해당
    - FLOAT (디폴트): 타입 접미사 "F"에 해당
  - 기본 구조체들 ("D2DBaseTypes.h")
    - D2D_POINT_2U struct { UINT32 x, y;}
    - D2D_POINT_2F struct { FLOAT x, y;}
    - D2D_RECT_U struct { UINT32 left, top, right, bottom;}
    - D2D_RECT_F struct { FLOAT left, top, right, bottom;}
    - D2D_SIZE_U struct { UINT32 width, height;}
    - D2D_SIZE_F struct { FLOAT width, height;}
    - D2D_COLOR_F struct { FLOAT r, g, b, a;}
    - D2D_MATRIX_3X2_F struct { FLOAT _11, _12, _21, _22, _31, _32;}
- 버전별 타입 ("D2D1.h")
  - typedef D2D_XXX D2D1_XXX; (XXX=위의 8개 구조체)
  - D2D1_XXX를 사용할 것

D2D에서의 기본 타입들'

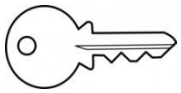
- 도움 함수 ("D2D1Helper.h")
  - 이름공간 "D2D1"
- 생성 도움 함수 공통
  - 인자 타입은 F의 경우 FLOAT, U의 경우 UINT32임
  - 인자의 디폴트는 모두 0(0.f)임.

- 생성 도움 함수
  - D2D1_POINT_2F Point2F (x,y)
  - D2D1_POINT_2U Point2U (x,y)
  - D2D1_SIZE_F SizeF (width,height)
  - D2D1_SIZE_U SizeU (width,height)
  - D2D1_RECT_F RectF (left,top,right,bottom)
  - D2D1_RECT_U RectU (left,top,right,bottom)



에러 다루기 정책

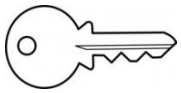
- 기본적인 에러 확인 방법
  - 함수는 HRESULT값을 리턴함
    - 함수가 올바로 처리될 수 없는 경우 실패를 알림
- 유효하지 않은 함수의 인자
  - 함수의 출력용 포인터 인자 ("[out]"으로 표시된 인자)
    - 인자: 특정 객체, 배열, 구조체의 주소
    - 호출 시에 NULL로 명시되어 있으면: 에러("access violation") 발생!
    - 예외: 출력용 포인터가 선택인 경우("[optional]"로 표시된 인자)
      - NULL로 지정해도 됨. (예: EndDraw, Flush)
  - 인자 유효한 값이 제공되어야 하는 함수의 입력용 인자
    - NULL값이 전달되면 에러("access violation") 발생!
    - 예외: 입력 인자가 "optional"인 경우
      - NULL로 지정하면 적절한 디폴트값들이 사용됨



간단한 그리기: GDI 버전

- 한 사각형 그리기: GDI 버전

```
switch (message) {
    case WM_PAINT: {
            PAINTSTRUCT ps;
            BeginPaint( hwnd, &ps );

            RECT rc; GetClientRect(hwnd, &rc);  // 그리기 영역의 크기를 얻음

            // 원래 객체를 저장함
            HGDIOBJ original = SelectObject( ps.hdc, GetStockObject(DC_PEN) );

            HPEN blackPen = CreatePen(PS_SOLID, 3, 0); // 펜을 생성함
            SelectObject(ps.hdc, blackPen); // 펜을 선택함

            // 사각형을 그림
            Rectangle( ps.hdc, rc.left + 100, rc.top + 100, rc.right - 100, rc.bottom - 100);

            DeleteObject(blackPen);

            SelectObject(ps.hdc, original); // 원래 객체를 복원함
            EndPaint(hwnd, &ps);
        }
        return 0;
    // 다른 메시지들을 다루는 코드...
```



간단한 그리기: Direct2D 버전

01.DrawRectangle

- 한 사각형 그리기 : Direct2D 버전 절차
  - 단계 0: Direct2D 헤더 포함
  - 단계 1: ID2D1Factory 생성
  - 단계 2: ID2D1HwndRenderTarget 생성
  - 단계 3: 붓 생성
  - 단계 4: 사각형 그리기
  - 단계 5: 자원 반납

단계 0, 단계 1

- 단계 0: Direct2D 헤더 포함
  - #include <d2d1.h>
- 단계 1: ID2D1Factory 생성
  - Direct2D의 사용을 위한 출발점임
    - Direct2D의 자원 생성을 위해서 필요함
  - 장치와 독립적임
    - 한번 생성하고 응용이 종료되기 전까지 유지함
  - 인자: 단일 스레드 또는 다중 스레드 중에서 선택
    - 여러 스레드가 이 장치관련 자원을 접근하는 것에 대한 동기화 지원 여부
    - SINGLE_THREADED: 단일 스레드라면 최적의 성능
    - MULTI_THREADED: CPU에서 여러 스레드가 이 장치관련 자원을 동시 처리하는 경우에 적합
  - 예

```
ID2D1Factory* pD2DFactory = NULL;
D2D1CreateFactory( D2D1_FACTORY_TYPE_SINGLE_THREADED, &pD2DFactory );
```

단계 2

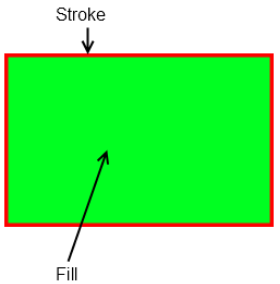
- 단계 2: ID2D1HwndRenderTarget 생성
  - 렌더타겟: 다음을 수행하는 장치임
    - 그리기 연산을 수행함
    - 장치와 관련된 그리기 자원(예: 브러시 등)을 생성할 수 있음
    - 여러 종류가 있음
      - 예: ID2D1HwndRenderTarget: 스크린의 일부에 렌더함
    - 렌더타겟은 작업수행을 위해서,
      - 가능하면 GPU를 사용하고, 가용하지 않은 경우 CPU를 사용함
    - 렌더타겟의 행위 명시: D2D1_RENDER_TARGET_TYPE
  - 렌더타겟의 생성 예

```
RECT rc; GetClientRect(hwnd, &rc); // 그리기 영역의 크기를 얻음

// Create a Direct2D render target
ID2D1HwndRenderTarget* pRT = NULL;
HRESULT hr = pD2DFactory->CreateHwndRenderTarget(
    D2D1::RenderTargetProperties(),
    D2D1::HwndRenderTargetProperties(hwnd, D2D1::SizeU(rc.right-rc.left, rc.bottom-rc.top)),
    &pRT );
```

단계 2’

- 단계 2: ID2D1HwndRenderTarget 생성’
  - 렌더타겟의 생성 함수: CreateHwndRenderTarget
    - 첫번째 인자: D2D1_RENDER_TARGET_PROPERTIES 구조체
      - 원격 디스플레이 옵션을 명시함: 렌더 주체(소프트웨어/하드웨어), DPI 등 명시
      - 디폴트 렌더타겟 속성을 사용하려면: 도움함수 D2D1::RenderTargetProperties 를 사용
    - 두번째 인자: D2D1_HWND_RENDER_TARGET_PROPERTIES 구조체
      - 다음을 명시함: 컨텐츠가 렌더될 HWND, 렌더타겟의 초기 크기(픽셀 단위), 표현 옵션
      - HWND와 초기 크기를 명시하는 도움 함수: D2D1::HwndRenderTargetProperties
    - 세번째 인자: 리턴될 렌더타겟 주소
  - 렌더타겟의 유지
    - 렌더타겟이 생성될 때에 자원들은 GPU에서 할당됨 (하드웨어 가속 기능이 가용한 경우에)
    - 한번 생성 후에 응용이 종료되기 전까지 생성된 렌더타겟을 유지해야 함.
    - 중간에 D2DERR_RECREATE_TARGET 에러가 발생된다면 렌더타겟 및 관련 자원들을 다시 생성해야 함.

단계 3



- 단계 3: 붓(brush) 생성
  - 렌더타겟을 사용하여 붓(brush) 생성
    - 붓: 한 영역을 색칠(stroke,fill)하는 객체임
      - stroke는 외곽선을 그림. fill은 모양을 채움.
      - 참고: 어떤 API에서는 stroke를 위해서 펜이라는 객체를 사용함.
        - » D2D에서 stroke를 위해서는, 그리기 함수의 인자로, 붓(brush)과 더불어, 획(stroke) 스타일(ID2D1StrokeStyle 인터페이스)을 지정하면 됨.
  - 붓(brush)의 타입
    - 단일색(solid color) 붓 : 명시된 색으로 그림
    - 계조(gradient) 붓 : 선형/방사형(linear/radial) 계조로 그림
    - 비트맵(bitmap) 붓 : 비트맵이나 패턴으로 그램
  - 사용
    - 브러시를 생성한 렌더타겟만 그 브러시를 사용할 수 있음.
      - 일반적으로, 브러시를 한번 생성한 후에, 렌더타겟의 수명 동안 함께 유지됨.
      - 예외: ID2D1SolidColorBrush는 가벼운 자원이므로, 수시로 생성 및 소멸시켜도 됨
      - 단일 ID2D1SolidColorBrush 객체만 생성한 후에 컬러만 바꿔가며 사용해도 됨
  - 예

```
ID2D1SolidColorBrush* pBlackBrush = NULL;
pRT->CreateSolidColorBrush( D2D1::ColorF(D2D1::ColorF::Black), &pBlackBrush );
```

단계 4, 단계 5

- 단계 4: 사각형 그리기
  - DrawRectangle 함수를 사용
    - 인자: 그릴 사각형, 사각형의 외곽선을 그리는데 사용할 브러시
    - 선택적 인자: stroke width(디폴트 1), stroke style(ID2D1StrokeStyle)
  - 그리기 명령
    - 그리기 명령들의 호출 전/후에: BeginDraw/EndDraw 함수 호출

```
pRT->BeginDraw();
pRT->DrawRectangle( D2D1::RectF(rc.left+100.0f, rc.top+100.0f,
                                rc.right-100.0f, rc.bottom-100.0f), pBlackBrush);
HRESULT hr = pRT->EndDraw();
```

- 단계 5: 자원 반납
  - 렌더타겟을 반납할 경우에, 그 렌더타겟으로 생성한 모든 자원들도 반드시 반납

```
SafeRelease(pRT);
SafeRelease(pBlackBrush);
```

  - 응용이 종료될 때에, D2D factory도 반납

```
SafeRelease(pD2DFactory);
```