게임프로그래밍

렌더타겟

박종승

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

목차

- [참고: this→“자원”→“변환”]
- 렌더타겟이란
- 렌더타겟 생성 인자
- 예제: 헬로 월드
- 픽셀 포맷

렌더타겟이란

- 렌더타겟이란?
  - 기능: 그리기 위한 자원들을 생성하고 그리기 연산을 수행함
  - ID2D1RenderTarget 를 상속한 자원

- 렌더타겟의 종류
  - ID2D1HwndRenderTarget : 한 윈도우에 렌더함
  - ID2D1DCRenderTarget : 한 GDI 장치 context에 렌더함
  - Bitmap 렌더타겟 : 한 off-screen bitmap에 렌더함
  - "DXGI 표면 렌더타겟" : 한 "DXGI 표면"에 렌더함. D3D와 연결하여 사용하기 위한 목적임.

- 렌더타겟의 특징
  - 렌더타겟은 특정 렌더링 장치와 연관되어 있으므로, 장치 의존적임
  - 장치가 제거되면 기능이 정지됨

렌더타겟

- 렌더타겟 자원
  - 렌더타겟이 생성하는 모든 자원은 장치 의존적임
    - 렌더타겟 자체가 장치 의존적이므로
  - 생성 가능한 자원의 타입들: bitmap, brush, layer, mesh

- 그리기 명령
  - 그리기 함수들의 호출 묶음 전/후에
    - ID2D1RenderTarget::BeginDraw
    - ID2D1RenderTarget::EndDraw
  - 그리기 함수들: stroke 또는 fill
    - 대부분 인자로 한 shape(한 primitive 또는 한 geometry)와 한 brush를 가짐
  - 렌더타겟이 제공하는 기타 함수들
    - 클리핑, opacity mask 적용, 좌표공간 변환 등
  - D2D는 왼손좌표계를 사용함: +x축이 오른쪽, +y축이 아래쪽

일괄처리로 동작하는 그리기 함수

- 일괄처리 함수
  - 종류: 모든 그리기 함수(DrawXXX,FillXXX) 및 GeometrySink 함수
  - 모든 그리기 함수: BeginDraw와 EndDraw 사이에 배치
    - BeginDraw 이후부터 모든 그리기 명령들을 모아두기 시작함.
    - EndDraw 호출 시에 그리기 명령들의 일괄처리 시작됨
    - 참고: 그리기 명령들의 일괄처리가 시작되는 경우
      - (a) EndDraw 호출 시에
      - (b) Flush 함수가 명시적으로 호출된 경우에
      - (c) 렌더링 명령들을 담고 있는 버퍼가 가득찬 경우;
  - GeometrySink 함수 (ID2D1GeometrySink의 함수)
    - Close 호출 시에 일괄처리 시작됨

그리기의 에러 다루기

- 일괄처리 함수의 에러 다루기
  - 그리기 함수는 일괄처리됨
    - 그리기 함수로부터 그리기 수행 결과의 성공 여부를 즉시 알 수 없음
  - 일괄처리 함수들의 에러는 일괄처리가 완료된 시점에서 에러가 보고됨
  - 그리기 함수의 수행 도중 에러가 발생되면:
    - 내부 에러 상태가 set됨
    - 에러 상태가 set이면 렌더타겟은 그리기를 중지함
  - 에러 여부를 알려면:
    - Flush 함수나 EndDraw 함수의 리턴값을 살펴볼 것
- Flush 함수
  - 용도: 일괄처리를 기다리는 그리기 명령들을 강제 처리되도록 요청함
  - Flush가 호출되면:
    - 기존의 에러 상태를 reset하고 그리기를 재개함
  - (효율적인 구현을 위해서) Flush 사용의 제한
    - 자원의 관리는 D2D에 맡기는 것이 바람직함
      - Flush 함수 호출은 피하는것이 바람직함

렌더타겟 생성 인자1

- 렌더타겟 생성을 위한 인자1
  - D2D1_RENDER_TARGET_PROPERTIES

```
struct D2D1_RENDER_TARGET_PROPERTIES {
  D2D1_RENDER_TARGET_TYPE     type;
  D2D1_PIXEL_FORMAT           pixelFormat;
  FLOAT                       dpiX;
  FLOAT                       dpiY;
  D2D1_RENDER_TARGET_USAGE    usage;
  D2D1_FEATURE_LEVEL          minLevel;
};
```

- 인자
  - type
    - D2D1_RENDER_TARGET_TYPE_XXX
      - XXX=DEFAULT,SOFTWARE,HARDWARE
      - DEFAULT: 하드웨어가 가용하면 사용하고, 그 외에는 소프트웨어 사용
  - pixelFormat
    - 렌더타겟의 픽셀 포멧
    - 편리함수: D2D1::PixelFormat()
      - 인자를 생략하면, D2D가 적절한 픽셀 포멧을 추천함

렌더타겟 생성 인자1'

- 인자'
  - dpiX,dpiY
    - 렌더타겟의 수평,수직 DPI
    - 둘다 모두 0,0으로 명시하면 디폴트 DPI를 사용함. 디폴트 DPI:
      - Compatible 렌더타겟의 경우: 부모 렌더타겟의 DPI
      - ID2D1HwndRenderTarget인 경우: 렌더타겟의 ID2D1Factory로 얻은 시스템 DPI
      - 그 외의 렌더타겟의 경우: 96
  - usage
    - D2D1_RENDER_TARGET_USAGE_XXX
      - XXX=NONE,FORCE_BITMAP_REMOTING,GDI_COMPATIBLE
      - 렌더타겟의 원격 사용 방식을 명시; GDI호환성을 명시;
  - minLevel
    - D2D1_FEATURE_LEVEL _XXX
      - XXX=DEFAULT,9,10
      - 하드웨어 렌더링을 위한 비디오카드의 최소 Direct3D feature level
      - DEFAULT로 지정하면, D2D가 현 장치의 feature level이 적절한지를 결정함

렌더타겟 생성 인자1”

- 편리함수
  - 간편한 사용방법
    - 인자를 명시하지 않으면 모두 디폴트가 지정됨:

```
D2D1_RENDER_TARGET_PROPERTIES RenderTargetProperties(
  D2D1_RENDER_TARGET_TYPE type = D2D1_RENDER_TARGET_TYPE_DEFAULT,
  const PIXEL_FORMAT pixelFormat = D2D1::PixelFormat(),
  FLOAT dpiX = 0.0,
  FLOAT dpiY = 0.0,
  D2D1_RENDER_TARGET_USAGE usage = D2D1_RENDER_TARGET_USAGE_NONE,
  D2D1_FEATURE_LEVEL minLevel = D2D1_FEATURE_LEVEL_DEFAULT
);
```

  - 사용 예

```
RECT rc;
GetClientRect(m_hwnd, &rc);
D2D1_SIZE_U size = D2D1::SizeU( rc.right - rc.left, rc.bottom - rc.top );
m_pD2DFactory->CreateHwndRenderTarget(
            D2D1::RenderTargetProperties(),
            D2D1::HwndRenderTargetProperties(m_hwnd, size),
            &m_pRenderTarget );
```

렌더타겟 생성 인자2

- 렌더타겟 생성을 위한 인자2
  - D2D1_HWND_RENDER_TARGET_PROPERTIES

```
struct D2D1_HWND_RENDER_TARGET_PROPERTIES {
  HWND                    hwnd;
  D2D1_SIZE_U             pixelSize;
  D2D1_PRESENT_OPTIONS    presentOptions;
};
```

- 인자
  - hwnd
    - 렌더타겟의 출력을 보낼 윈도우 핸들
  - pixelSize
    - 렌더타겟의 픽셀단위의 크기
  - presentOptions
    - present할 때의 옵션

렌더타겟 생성 인자2’

- 렌더타겟 생성을 위한 인자2’
  - 디스플레이 드라이버 동작 특성
    - EndDraw() 호출 시에 그린 내용들이 화면에 보여짐(present함)
      - LCD 디스플레이가 refresh해야 화면에 보이게 됨
        - » 통상 refresh 간격은 16ms (즉 60Hz)
    - 현재 EndDraw 시에 사용될 디스플레이용 버퍼가
      - 가장 최근의 과거 EndDraw 시의 사용된 내용을 유지하도록 할 것인지의 여부
        - » 디폴트 “no”. 즉 디스플레이용 버퍼의 메모리 위치가 바뀔 수 있음.
  - D2D1_PRESENT_OPTIONS
    - *_NONE(디폴트) : 렌더타겟은 present를 위해서 디스플레이가 refresh할 때까지 기다림. Present 시에 이전 프레임은 discard함.
    - *_RETAIN_CONTENTS : Present 시에 이전 프레임을 discard하지 않음.
    - *_IMMEDIATELY : 렌더타겟은 디스플레이가 refresh할 때까지 기다리지 않음
      - 즉시 디스플레이에 갱신되도록 함. (16ms를 기다리지 않음)

렌더타겟 생성 인자2”

- 편리 함수
  - 앞의 두 인자는 명시함

```
WINAPI HwndRenderTargetProperties(
  __in  HWND hwnd,
  __in  D2D1_SIZE_U pixelSize = D2D1::Size(static_cast<UINT>(0), static_cast<UINT>(0)),
  __in  D2D1_PRESENT_OPTIONS presentOptions = D2D1_PRESENT_OPTIONS_NONE
);
```

  - 사용 예

```
RECT rc;
GetClientRect(m_hwnd, &rc);
D2D1_SIZE_U size = D2D1::SizeU( rc.right - rc.left, rc.bottom - rc.top );
m_pD2DFactory->CreateHwndRenderTarget(
            D2D1::RenderTargetProperties(),
            D2D1::HwndRenderTargetProperties(m_hwnd, size),
            &m_pRenderTarget );
```

예제: 헬로 월드

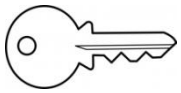
01.HelloWorld

- 예제
  - 한 윈도우에 텍스트를 렌더링하는 예
    - CreateHwndRenderTarget를 호출하여 ID2D1HwndRenderTarget을 생성

```
RECT rc;
GetClientRect(m_hwnd, &rc);
D2D1_SIZE_U size = D2D1::SizeU( rc.right - rc.left, rc.bottom - rc.top );
m_pD2DFactory->CreateHwndRenderTarget(
            D2D1::RenderTargetProperties(),
            D2D1::HwndRenderTargetProperties(m_hwnd, size),
            &m_pRenderTarget );
```

    - ID2D1HwndRenderTarget를 사용하여 텍스트를 그림

```
static const WCHAR sc_helloWorld[] = L"Hello, World!";
D2D1_SIZE_F renderTargetSize = m_pRenderTarget->GetSize();
m_pRenderTarget->BeginDraw();
m_pRenderTarget->SetTransform(D2D1::Matrix3x2F::Identity());
m_pRenderTarget->Clear(D2D1::ColorF(D2D1::ColorF::White));
m_pRenderTarget->DrawText( sc_helloWorld, ARRAYSIZE(sc_helloWorld) - 1,
            m_pTextFormat, D2D1::RectF(0, 0, renderTargetSize.width, renderTargetSize.height),
            m_pBlackBrush );
m_pRenderTarget->EndDraw();
```



픽셀 포맷

- 렌더타겟에 픽셀 포맷을 명시하기
  - 렌더타겟을 생성할 때에, 픽셀 포맷을 명시해야 함
  - D2D1_PIXEL_FORMAT 구조체
    - 렌더타겟이 사용하는 픽셀포맷과 알파모드를 명시함
    - format : 각 픽셀 포멧을 명시 DXGI_FORMAT_XXX
    - alphaMode : 알파정보의 해석 방법을 명시. D2D1_ALPHA_MODE_XXX
  - 알파 채널
    - 픽셀포맷의 일부로, coverage 정보나 opacity 정보를 명시함
      - Coverage 정보: 그려야 할 기하의 모양 정보로 그려질 부분과 그려지지 않을 부분을 구분하는 정보임
      - Opacity 정보: 불투명도를 나타내며 100%이면 완전히 불투명임.
    - 렌더타겟이 알파채널을 사용하지 않는 경우
      - D2D1_ALPHA_MODE_IGNORE로 지정하여 렌더타겟을 생성해야 함
  - 픽셀 포맷의 명시
    - ID2D1Factory::CreateHwndRenderTarget 함수 등의 인자
    - D2D1_RENDER_TARGET_PROPERTIES 구조체의 pixelFormat 필드

픽셀 포맷'

- 편리함수

```
D2D1_PIXEL_FORMAT PixelFormat(
  __in  DXGI_FORMAT format = DXGI_FORMAT_UNKNOWN,
  __in  ALPHA_MODE alphaMode = D2D1_ALPHA_MODE_UNKNOWN
);
```

- 사용 예

```
RECT rc;
GetClientRect(m_hwnd, &rc);
D2D1_SIZE_U size = D2D1::SizeU( rc.right - rc.left, rc.bottom - rc.top );

D2D1_PIXEL_FORMAT pixelFormat = D2D1::PixelFormat(
            DXGI_FORMAT_B8G8R8A8_UNORM,
            D2D1_ALPHA_MODE_IGNORE );

D2D1_RENDER_TARGET_PROPERTIES props = D2D1::RenderTargetProperties();
props.pixelFormat = pixelFormat;

m_pD2DFactory->CreateHwndRenderTarget( props,
            D2D1::HwndRenderTargetProperties(m_hwnd, size),
            &m_pRT );
```

ID2D1HwndRenderTarget가 지원하는 픽셀 포맷

- 렌더타겟 ID2D1HwndRenderTarget
  - 생성시의 명시한 타입: D2D1_RENDER_TARGET_TYPE_DEFAULT
    - 지원하는 (format, alphaMode) 조합:
      - DXGI_FORMAT_XXX,D2D1_ALPHA_MODE_XXX
        - » B8G8R8A8_UNORM, PREMULTIPLIED
        - » B8G8R8A8_UNORM, IGNORE
        - » B8G8R8A8_UNORM, UNKNOWN
        - » UNKNOWN, PREMULTIPLIED
        - » UNKNOWN, IGNORE
        - » UNKNOWN, UNKNOWN
      - 위의 조합들은 하드웨어 및 소프트웨어 모두 지원함
    - 디폴트
      - DXGI_FORMAT_UNKNOWN = DXGI_FORMAT_B8G8R8A8
      - D2D1_ALPHA_MODE_UNKNOWN = D2D1_ALPHA_MODE_IGNORE

기타 렌더타겟이 지원하는 픽셀 포맷

- 다음 종류의 렌더타겟에 대해서는 (link: 참고)
  - Compatible 렌더타겟이 지원하는 픽셀 포맷
    - Compatible 렌더타겟 = ID2D1RenderTarget::CreateCompatibleRenderTarget 함수로 생성된 ID2D1BitmapRenderTarget
    - 생성한 렌더타겟이 지원하는 범위를 상속 받음
  - DXGI 표면 렌더타겟이 지원하는 픽셀 포멧
    - DXGI 표면 렌더타겟 = ID2D1Factory::CreateDxgiSurfaceRenderTarget 함수가 생성한 ID2D1RenderTarget
  - WIC 비트맵 렌더타겟이 지원하는 픽셀 포멧
    - WIC 비트맵 렌더타겟 = ID2D1Factory::CreateWicBitmapRenderTarget 함수가 생성한 ID2D1RenderTarget
  - ID2D1DCRenderTarget 이 지원하는 픽셀 포멧
  - 비트맵(ID2D1Bitmap)의 픽셀포멧 명시
    - ID2D1RenderTarget::CreateSharedBitmap나 ID2D1RenderTarget::CreateBitmapFromWicBitmap로 비트맵을 생성할 때 픽셀 포멧 명시

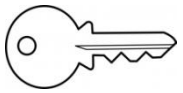
알파 모드

- 픽셀 포멧에 명시하는 alphaMode
  - D2D1_ALPHA_MODE_XXX
    - UNKNOWN
      - 사용하는 렌더타겟 종류에 적합한 알파모드로 지정함 (참고: MSDN)
    - STRAIGHT
      - 알파값은 원래 컬러의 투명도를 의미함
        - » 알파값을 칼라채널값에 미리곱하지 않음.
      - 예: RGBA값이 (255,0,0,153)이라면 60%불투명도의 (255,0,0) 붉은색으로 해석함
      - D2D1_COLOR_F 는 항상 straight 모드로 해석함
        - » 렌더타겟의 알파 모드가 어떻게 지정되어 있는지에 상관 없음
    - PREMULTIPLIED
      - 각 칼라값은 먼저 알파값을 곱해서 스케일링되어 있음.
        - » 따라서, 알파채널값은 어떤 칼라채널값보다 큼
      - 예: RGB값이 (255,0,0)인 붉은색을 60% 불투명도로 명시하려면,
        - » (153/255)=60%를 RGB에 곱한 (153,0,0,153)으로 표시함.
    - IGNORE
      - 알파값을 무시함

알파 모드'

- 붓의 불투명도
  - 렌더타겟의 알파 모드와 상관없이 붓의 불투명도가 작동함

- (참고) 픽셀 포맷 및 알파모드 관련
  - "Supported Pixel Formats and Alpha Modes" (link: MSDN)



참고: Compatible A8 렌더타겟

22.TextAnimationSample

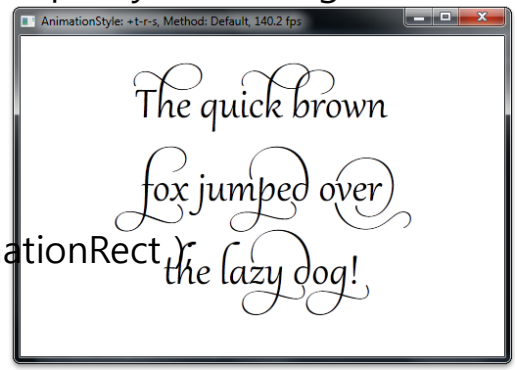
- compatible A8 렌더타겟이란?
  - ID2D1BitmapRenderTarget 객체로,
  - ID2D1RenderTarget::CreateCompatibleRenderTarget 함수로 생성
  - A8 픽셀 포맷을 사용 (DXGI_FORMAT_A8_UNORM)
- 용도
  - 텍스트 애니메이션 시의 부드러운 transition에 적합함. 특히 다음에 적합,
  - 단순한 애니메이션만을(예: 이동,회전,크기조정,칼라변화) 포함하는 텍스트(또는 anti-aliased 기하)를 렌더링하는 응용의 프레임율 향상
  - 텍스트를 쭉 늘이거나 줄이는 응용의 시각적 연속성을 향상

참고: Compatible A8 렌더타겟

22.TextAnimationSample

- 소스 예제(상): Text Animation Sample
  - [참고: 텍스트 렌더링 및 불투명 마스크의 이해가 필요함]
  - 텍스트를 불투명 마스크 용도로 compatible A8 렌더타겟에 그림
  - 변환을 불투명 마스크에 적용하여 빠른 렌더링 결과를 얻음
  - 설명생략 (참고: link: MSDN)

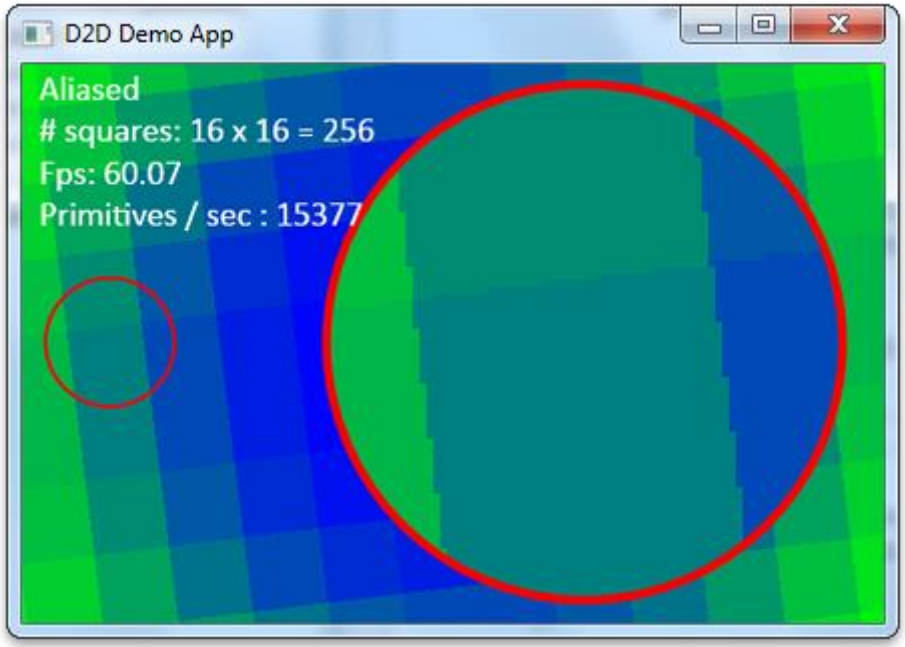
```
// Compatible A8 렌더타겟을 생성
 ID2D1BitmapRenderTarget *m_pOpacityRT;
D2D1_PIXEL_FORMAT alphaOnlyFormat = D2D1::PixelFormat(
                DXGI_FORMAT_A8_UNORM, D2D1_ALPHA_MODE_PREMULTIPLIED);
m_pRT->CreateCompatibleRenderTarget( NULL, &maskPixelSize, &alphaOnlyFormat,
                D2D1_COMPATIBLE_RENDER_TARGET_OPTIONS_NONE, &m_pOpacityRT );
D2D1_RECT_F destinationRect = D2D1::RectF( roundedOffset.x, roundedOffset.y,
                 roundedOffset.x + opacityRTSize.width, roundedOffset.y + opacityRTSize.height );
// 비트맵을 꺼냄
ID2D1Bitmap *pBitmap = NULL;
m_pOpacityRT->GetBitmap(&pBitmap);
// 비트맵을 렌더링함
m_pRT->FillOpacityMask( pBitmap, m_pBlackBrush,
                D2D1_OPACITY_MASK_CONTENT_TEXT_NATURAL, &destinationRect );
pBitmap->Release();
```



참고: 예제: MSAA Rendering Sample

24.MSAARenderingSample

- 예제(상): MSAA Rendering Sample
  - 설명생략 (참고: link: MSDN)



성능에 대한 고려: 렌더타겟의 크기와 장면 복잡도

- 성능 분석에서 과부하 지점을 알아내기 위해서
  - 장면이 fill-rate bound인지 vertex bound인지를 알아야 함
    - fill-rate bound : 처리 속도가 그래픽 카드가 1초당 렌더해서 비디오메모리에 쓸 수 있는 픽셀개수에 제한됨
    - 정점 바운드 : 처리 속도가 장면이 포함하는 geometry의 복잡성에 제한됨
- 장면의 복잡도 이해하기
  - 렌더타겟의 크기를 바꾸어보았을 때
    - 성능이 크게 개선된다면: 응용이 fill-rate bound라는 의미임
    - 아니면, 장면의 복잡도가 성능의 bottleneck이라고 보면 됨
- 장면이 fill-rate bound라면
  - 렌더타겟의 크기를 줄이자
- 장면이 vertex bound라면
  - Geometry의 복잡도를 줄이자
    - 그러나, 출력영상의 품질이 떨어진다는 점을 염두에 두어야 함