게임프로그래밍

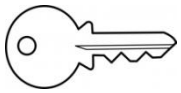
비트맵

박종승

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

목차

- 비트맵 그리기
- 파일로부터 비트맵 로드하기
- 자원으로부터 비트맵 로드하기
- 비트맵을 불투명 마스크로 사용하기



비트맵 그리기

- 비트맵 그리기
  - 비트맵 객체: ID2D1Bitmap
  - 비트맵 객체 그리기: ID2D1RenderTarget::DrawBitmap 함수 호출
- 비트맵 객체 만들기
  - 응용의 자원 파일로부터 비트맵 로드하여 ID2D1Bitmap 객체 생성
    - LoadBitmapFromResource 함수 호출
      - 참고: LoadBitmapFromResource 함수의 구현은 "이후에 설명함"

```
LoadBitmapFromResource( m_pRenderTarget, m_pWICFactory,
                        L"SampleImage", L"Image", 200, 0, &m_pBitmap );
```

- 비트맵 객체는 장치 의존적 자원임
  - 렌더타겟과 수명을 함께 해야 함
- 에러 체크
  - DrawBitmap 를 포함한 모든 DrawXXX 함수는 에러 리턴 안 함
    - 인자가 잘못된 경우에 실제 실행되지 않지만 프로그램은 정상적으로 진행됨
  - 대신, ID2D1RenderTarget::EndDraw의 리턴 결과를 확인해야 함

비트맵 그리기'

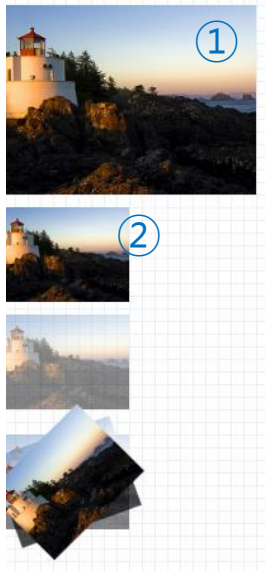
- DrawBitmap 함수 인자
  - 인자1: ID2D1Bitmap* bitmap
    - 그릴 비트맵
  - 인자2: D2D1_RECT_F destinationRectangle
    - 비트맵이 그려질 크기와 위치(렌더타겟 좌표계에서의 DIP 단위)
    - "비트맵을 이 사각형 크기로 스케일링함"
    - 생략가능. 디폴트 NULL(비트맵을 스케일링 없이 렌더타겟의 원점에 그림).
  - 인자3: FLOAT opacity
    - 비트맵에 적용할 불투명도로 [0.0f,1.0f] 사이의 실수
    - 이 값이 비트맵 콘텐츠의 알파값과 곱해져서 적용됨
    - 생략가능. 디폴트 1.0f
  - 인자4: interpolationMode:
    - 비트맵이 스케일링/회전될 경우에 사용될 보간 모드
    - D2D1_BITMAP_INTERPOLATION_MODE_XXX, XXX=NEAREST_NEIGHBOR, LINEAR
    - 생략가능. 디폴트 LINEAR
  - 인자5: D2D1_RECT_F sourceRectangle
    - 그릴 비트맵의 부분의 크기와 위치를 명시 (비트맵 좌표계에서의 DIP 단위)
    - 생략가능. 디폴트 NULL(비트맵 전 영역을 그림).

비트맵 그리기”

- 예제: DrawBitmap 함수로 비트맵 그리기

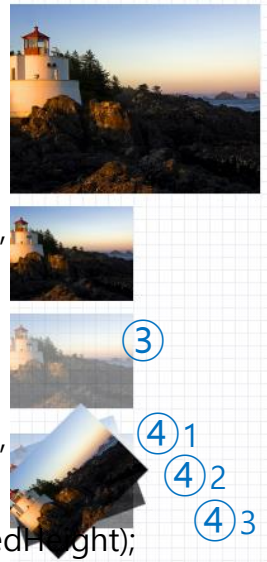
01.DrawBitmapExample

```
D2D1_SIZE_F renderTargetSize = m_pRenderTarget->GetSize(); //렌더타겟의 크기 얻기
m_pRenderTarget->BeginDraw();
m_pRenderTarget->SetTransform(D2D1::Matrix3x2F::Identity());
m_pRenderTarget->Clear(D2D1::ColorF(D2D1::ColorF::White));
m_pRenderTarget->FillRectangle( //배경 그리드 칠하기
                D2D1::RectF(0.0f, 0.0f, renderTargetSize.width, renderTargetSize.height),
                m_pGridPatternBitmapBrush );
D2D1_SIZE_F size = m_pBitmap->GetSize(); //비트맵 크기 얻기

//비트맵① 그리기
D2D1_POINT_2F upperLeftCorner = D2D1::Point2F(100.f, 10.f);
m_pRenderTarget->DrawBitmap( m_pBitmap,
                D2D1::RectF( upperLeftCorner.x, upperLeftCorner.y,
                                upperLeftCorner.x + size.width, upperLeftCorner.y + size.height) );

//비트맵② 그리기; 절반 크기로;
upperLeftCorner.y = upperLeftCorner.y + size.height + 10.f; //이전 비트맵 아래로
float scaledWidth = size.width / 2.f;  float scaledHeight = size.height / 2.f; //절반 크기로 크기조정
m_pRenderTarget->DrawBitmap( m_pBitmap,
                D2D1::RectF( upperLeftCorner.x, upperLeftCorner.y,
                                upperLeftCorner.x + scaledWidth, upperLeftCorner.y + scaledHeight ),
                1.0, D2D1_BITMAP_INTERPOLATION_MODE_LINEAR ); //선형보간
```

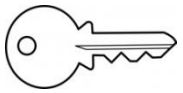


비트맵 그리기'''

- 예제: DrawBitmap 함수로 비트맵 그리기'




```
//비트맵③ 그리기; 절반 크기 및 절반 불투명도로;
upperLeftCorner.y = upperLeftCorner.y + size.height / 2.f + 10.f;
m_pRenderTarget->DrawBitmap( m_pBitmap,
                D2D1::RectF( upperLeftCorner.x, upperLeftCorner.y,
                             upperLeftCorner.x + scaledWidth, upperLeftCorner.y + scaledHeight),
                0.5, D2D1_BITMAP_INTERPOLATION_MODE_LINEAR );
// 비트맵들④1~3을 그리기; 다른 불투명도 및 회전각도로; 비트맵④1 그리기;
upperLeftCorner.y = upperLeftCorner.y + scaledHeight + 20.f;
m_pRenderTarget->DrawBitmap( m_pBitmap,
                D2D1::RectF( upperLeftCorner.x, upperLeftCorner.y,
                             upperLeftCorner.x + scaledWidth, upperLeftCorner.y + scaledHeight),
                0.5, D2D1_BITMAP_INTERPOLATION_MODE_LINEAR );
//비트맵④2 그리기; -20도 회전; 75% 불투명도로;
D2D1_POINT_2F lowerLeftCorner = D2D1::Point2F(upperLeftCorner.x, upperLeftCorner.y + scaledHeight);
D2D1_POINT_2F imageCenter = D2D1::Point2F( upperLeftCorner.x + scaledWidth / 2,
                                           upperLeftCorner.y + scaledHeight / 2 );
m_pRenderTarget->SetTransform( D2D1::Matrix3x2F::Rotation(-20, imageCenter) );
m_pRenderTarget->DrawBitmap( m_pBitmap,
                D2D1::RectF( upperLeftCorner.x, upperLeftCorner.y,
                             upperLeftCorner.x + scaledWidth, upperLeftCorner.y + scaledHeight),
                0.75, D2D1_BITMAP_INTERPOLATION_MODE_LINEAR );
//비트맵④3 그리기; -45도 회전; 완전 불투명으로;
m_pRenderTarget->SetTransform( D2D1::Matrix3x2F::Rotation(-45, imageCenter) );
m_pRenderTarget->DrawBitmap( m_pBitmap,
                D2D1::RectF( upperLeftCorner.x, upperLeftCorner.y,
                             upperLeftCorner.x + scaledWidth, upperLeftCorner.y + scaledHeight),
                1.0, D2D1_BITMAP_INTERPOLATION_MODE_LINEAR );
m_pRenderTarget->EndDraw();
```



파일로부터 비트맵 로드하기

LoadBitmapFromFile

02.LoadBitmapExample

- D2D에서의 비트맵 로드
  - 비트맵: ID2D1Bitmap 객체로 표현
  - D2D는 WIC를 사용하여 비트맵을 로드함
    - WIC(Windows Imaging Component)
      - COM기반 이미지 코덱 프레임워크임
- 비트맵 로드 절차
  - 단계 0: WIC 팩토리 IWICImagingFactory 객체 생성
  - 단계 1: IWICBitmapDecoder를 생성함
    - IWICImagingFactory::CreateDecoderFromFileName 함수 호출

```
IWICBitmapDecoder* pDecoder = NULL;
pIWICFactory->CreateDecoderFromFilename( L".₩₩sampleImage.jpg", //URI
            NULL, GENERIC_READ, WICDecodeMetadataCacheOnLoad, &pDecoder );
```

  - 단계 2: 이미지로부터 프레임을 얻기
    - 얻은 프레임을 IWICBitmapFrameDecode 객체에 저장함

```
IWICBitmapFrameDecode* pSource = NULL;
pDecoder->GetFrame(0, &pSource); //0번 프레임(단일 프레임이므로 유일함)
```

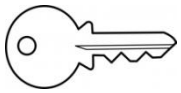
파일로부터 비트맵 로드하기'

- 비트맵 로드 절차'
  - 단계 3: 비트맵을 D2D 호환 포맷으로 변환
    - 호환포맷의 하나인 32bppPBGRA로 변환 (참고: link: MSDN)
      - GUID_WICPixelFormat32bppPBGRA = DXGI_FORMAT_B8G8R8A8_UNORM + D2D1_ALPHA_MODE_PREMULTIPLIED
    - 먼저, IWICFormatConverter 객체 생성
      - IWICImagingFactory::CreateFormatConverter 함수 호출
    - 다음, IWICFormatConverter 객체의 초기화 함수 호출하여 변환 수행

```
IWICFormatConverter* pConverter = NULL;
pIWICFactory->CreateFormatConverter(&pConverter);
pConverter->Initialize( pSource,
            GUID_WICPixelFormat32bppPBGRA,
            WICBitmapDitherTypeNone, NULL, 0.f, WICBitmapPaletteTypeMedianCut );
```

  - 단계 4: 비트맵(ID2D1Bitmap) 객체 생성
    - CreateBitmapFromWicBitmap 함수 호출
      - WIC 비트맵으로부터 D2D 비트맵 생성

```
ID2D1Bitmap* pBitmap = NULL;
pRenderTarget->CreateBitmapFromWicBitmap( pConverter, NULL, &pBitmap );
```



자원으로부터 비트맵 로드하기

LoadBitmapFromResource

02.LoadBitmapExample

- 응용의 "자원 정의 파일"(.rc 파일)
  : 자원을 정의하는 스크립트 파일
  - 컴파일 시에, 자원 정의 파일로부터 "자원 파일"(.res 파일)을 생성함
  - 링크 시에, 실행 파일에 합쳐짐
- 비트맵 로드 절차
  - 파일에서 로드하는 절차와 유사함
  - 단계 1: 응용의 "자원 정의 파일"에서 자원을 정의함
    - 예: 자원 정의 파일에 다음과 같이 정의함
      - 자원이름="SampleImage", 자원타입="Image"

```
#include "windows.h"
SampleImage  Image  "sampleImage.jpg"
```

  - 단계 2: 응용의 자원 파일로부터 이미지를 로드함

```
//자원을 찾기
HRSRC imageResHandle = NULL;
imageResHandle = FindResourceW(HINST_THISCOMPONENT, L"SampleImage", L"Image");
if (! imageResHandle) { /*에러: 자원을 찾을 수 없음*/ }
//자원을 로드하기
HGLOBAL imageResDataHandle = NULL;
imageResDataHandle = LoadResource(HINST_THISCOMPONENT, imageResHandle);
if (! imageResDataHandle) { /*에러: 자원을 로드할 수 없음*/ }
```

자원으로부터 비트맵 로드하기'

- 비트맵 로드 절차'
  - 단계 3: 해당 자원을 lock함; 이미지의 크기를 계산함;

```
//자원을 잠그고 메모리 포인터를 얻어옴
void *pImageFile = NULL;
pImageFile = LockResource(imageResDataHandle);
if (! pImageFile) { /*에러: lock할 수 없음*/
//이미지의 크기를 계산함.
DWORD imageFileSize = 0;
imageFileSize = SizeofResource(HINST_THISCOMPONENT, imageResHandle);
if (! imageFileSize) { /*에러: 이미지 크기를 계산할 수 없음 */ }
```

  - 단계 4: 자원을 메모리로 매핑하기 위해서 IWICStream 객체를 생성
    - IWICImagingFactory::CreateStream 함수를 호출
    - IWICStream::InitializeFromMemory 함수를 호출
      - 주어진 메모리 블록을 스트림으로 다룰 수 있도록 함

```
//WIC 스트림을 생성
IWICStream *pStream = NULL;
pIWICFactory->CreateStream(&pStream);
//스트림을 초기화
pStream->InitializeFromMemory( reinterpret_cast<BYTE*>(pImageFile), imageFileSize );
```

자원으로부터 비트맵 로드하기”

- 비트맵 로드 절차”
  - 단계 5: 스트림에 대한 디코더 (IWICBitmapDecoder 객체)를 생성함
    - IWICImagingFactory::CreateDecoderFromStream 함수를 호출

```
IWICBitmapDecoder *pDecoder = NULL;
pIWICFactory->CreateDecoderFromStream( pStream, NULL,
                                WICDecodeMetadataCacheOnLoad, &pDecoder );
```

  - (이후 단계 6-8은 이전의 파일로부터 비트맵 로드하기에서의 단계 2-4와 동일함)

비트맵을 불투명 마스크로 사용하기 예제

Ref. 08brush: **04.OpacityMaskBitmap**

- 예제
  - 단계1

```
LoadBitmapFromResource(m_pRenderTarget,m_pWICFactory, L"GoldFishMask", L"Image", &m_pBitmapMask);
```

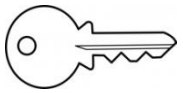
  - 단계2

```
LoadBitmapFromResource(m_pRenderTarget,m_pWICFactory, L"GoldFish", L"Image", &m_pOrigBitmap);

D2D1_BITMAP_BRUSH_PROPERTIES propertiesXClampYClamp = D2D1::BitmapBrushProperties(
        D2D1_EXTEND_MODE_CLAMP, D2D1_EXTEND_MODE_CLAMP,
        D2D1_BITMAP_INTERPOLATION_MODE_NEAREST_NEIGHBOR );
m_pRenderTarget->CreateBitmapBrush( m_pOrigBitmap,
                propertiesXClampYClamp, &m_pOriginalBitmapBrush );
```

  - 단계3

```
m_pRenderTarget->SetAntialiasMode(D2D1_ANTIALIAS_MODE_ALIASED);
m_pRenderTarget->FillOpacityMask( m_pBitmapMask, m_pOriginalBitmapBrush,
            D2D1_OPACITY_MASK_CONTENT_GRAPHICS, &rcBrushRect, NULL );
m_pRenderTarget->SetAntialiasMode(D2D1_ANTIALIAS_MODE_PER_PRIMITIVE);
```





비트맵의 효율적인 사용

- 비트맵의 사용
  - 비트맵 자원의 생성 및 소멸은 비용이 비쌈
    - GPU 텍스처 메모리 할당과 관련된 비용
      - 하드웨어로 자원을 생성 및 소멸하는 것은 매우 비쌈
      - 특히 빈번히 사용되는 비트맵을 비디오 카드에서 생성하는 것은 매우 비쌈
  - 빈번한 비트맵의 생성을 자제
    - 응용의 속도향상을 위해서 bitmap을 재사용하자
- 비트맵을 재사용 (비트맵 내용의 갱신)
  - 기존의 비트맵의 내용을 갱신
    - 비트맵을 생성하는 경우보다 비용이 매우 저렴함
  - 갱신 관련 함수: ID2D1Bitmap::CopyFromXXX
    - XXX=Bitmap,RenderTarget,Memory
  - 함수의 기능
    - 인자로 주어진 비트맵/렌더타겟/메모리의 영역을 현재 비트맵으로 복사함
    - 비트맵 사이즈는 바뀌지 않음

비트맵의 효율적인 사용'

- 비트맵 개수의 최소화
  - 여러 작은 비트맵 들을 생성해서 사용하는 것은 낭비의 소지가 있음
    - 비디오카드에서 최소 메모리 할당 크기가 있음.
      - 더 작은 크기의 자원의 경우 최소 크기가 할당되므로 그 차이만큼은 낭비됨
    - 하나의 큰 비트맵을 할당하고 모든 작은 비트맵 콘텐츠들을 담자
      - 이런 용도의 큰 비트맵을 atlas라고 함
      - 작은 bitmap이 필요할때마다 사각형을 명시해서 atals로부터 추출할 수 있음
        - » 예: 여러 icon들을 하나의 bitmap으로 구성
      - 참고: atlas 비트맵이 너무 크면 에러 발생할 수 있음
        - » 비디오 메모리에서 생성되는 비트맵에는 최대 크기의 제한이 있음. (adapter에 따라서 다름)
  - 추천: 각 비트맵의 크기 제약
    - 대부분의 경우, 최소 크기가 64 KB 이상 되도록 하자.
    - 4 KB보다 작은 비트맵들은 가급적 없도록 하자.