게임프로그래밍

레이어

박종승

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

목차

- [학습 순서: "붓"→"불투명 마스크"→this]
- 레이어란?
- 레이어의 생성과 사용
- PushLayer 함수 인자
- 레이어의 내용 범위 제한
- 레이어를 사용한 클리핑
- 불투명 마스크를 레이어에 적용하기
- 축정렬 클립 사각형으로 클리핑하기

레이어란?

- 레이어(layer)란
  - ID2D1Layer 객체로 표현
  - 여러 그리기 연산들의 그룹을 다룰 수 있도록 함
  - 렌더타겟이 생성하는 자원으로, 장치 의존적임 (붓처럼)
- 레이어의 사용 방법
  - 한 레이어를 렌더타겟에 push함
  - 이후로의 렌더타겟의 그리기 연산은 레이어로 전달됨
  - 레이어의 사용이 끝나면 해당 레이어를 렌더타겟으로부터 pop함
  - pop할 때에 레이어의 내용이 렌더타겟에 합성됨
- 효율적인 사용
  - 여러 효과를 구현할 수 있는 강력한 렌더링 기법을 제공함
  - 너무 많은 개수의 레이어를 사용하면 성능이 크게 낮아짐
  - 레이어 자원을 가급적 재사용하도록 노력해야 함

레이어의 생성과 사용

- 레이어의 생성
  - ID2D1RenderTarget::CreateLayer 함수 호출
- 레이어 사용 시작: PushLayer 함수 호출
  - 호출 시점: BeginDraw 함수의 호출 이후에
  - 인자로 주어진 레이어를 렌더타겟에 추가함
  - 호출 이후부터, 렌더타겟의 모든 그리기 연산은 레이어로 넘겨짐
  - 레이어의 위치는 렌더타겟에 지정된 변환에 영향을 받음
- 레이어 사용 종료: PopLayer 함수 호출
  - 레이어의 내용을 렌더타겟에 합성함
  - 호출 시점: EndDraw 호출 이전에
  - 각 PushLayer은 대응되는 PopLayer 호출을 가져야 함
    - 모든 레이어가 pop되기 전에 Flush 함수가 호출되면 에러 발생됨
  - 에러발생 여부의 확인
    - 다른 그리기 연산처럼, EndDraw나 Flush 함수의 리턴값으로 확인해야 함

PushLayer 함수 인자

- PushLayer의 인자: D2D1_LAYER_PARAMETERS 구조체
- 구조체 인자 (A)
  - contentBounds: 레이어 내용의 범위를 명시하는 사각형
- 구조체 인자 (B)
  - geometricMask: 레이어가 클리핑되도록 하는 기하 마스크.
  - maskAntialiasMode: 클리핑에서 적용될 antialiasing 모드를 명시함.
  - maskTransform: 레이어를 합성할 때에 기하 마스크에 적용될 변환.
- 구조체 인자 (C)
  - opacity: 이 레이어의 불투명도
  - opacityBrush: 레이어의 불투명도를 바꾸기 위한 붓.
- 구조체 인자 (기타)
  - layerOptions: 텍스트 렌더링을 표시.
    - 디폴트 off. ClearType antialiasing으로 텍스트를 렌더링할 때에 on함.

PushLayer 함수 인자'

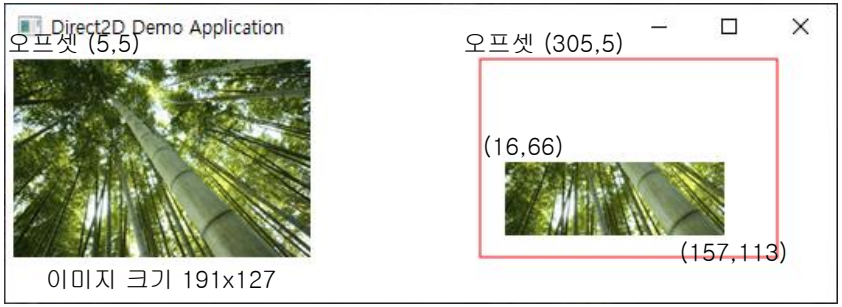
- D2D1::LayerParameters()
  - D2D1_LAYER_PARAMETERS 구조체를 생성하는 편리함수

```
D2D1_LAYER_PARAMETERS LayerParameters(
  const D2D1_RECT_F &          contentBounds      = D2D1::InfiniteRect(),
  ID2D1Geometry *              geometricMask      = NULL,
  D2D1_ANTIALIAS_MODE          maskAntialiasMode  = *_PER_PRIMITIVE,
  D2D1_MATRIX_3X2_F            maskTransform      = D2D1::IdentityMatrix(),
  FLOAT                        opacity            = 1.0,
  ID2D1Brush *                 opacityBrush       = NULL,
  D2D1_LAYER_OPTIONS           layerOptions       = *_NONE
);
```

레이어의 내용 범위 제한

06.LayerContentBounds

- 구조체 인자 (A)
  - contentBounds: 레이어 내용의 범위를 명시하는 사각형
    - 레이어 내용의 범위를 명시하는 사각형
    - 명시된 범위 이내에 대해서만 렌더타겟에 합성됨
      - 이 범위 외부의 내용들은 그려지지 않음
    - 디폴트값은 InfiniteRect()로 얻을 수 있음
      - 이 값은 렌더타겟 범위가 되도록 함.
- 예: 원본 이미지가 클리핑됨
  - 클리핑 사각형: 왼쪽위 (16,66),
    오른쪽아래 (157,113)




```
ID2D1Layer *pLayer = NULL;
pRT->CreateLayer(NULL, &pLayer);
pRT->SetTransform(D2D1::Matrix3x2F::Translation(305, 5));
pRT->PushLayer( D2D1::LayerParameters( D2D1::RectF(16, 66, 157, 113) ), pLayer );
pRT->DrawBitmap(m_pBambooBitmap, D2D1::RectF(0, 0, 191, 127));
pRT->PopLayer();
SafeRelease(&pLayer);
```

레이어를 사용한 클리핑

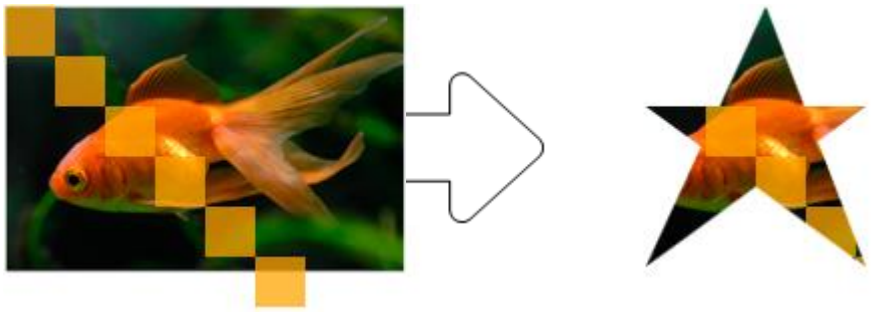
- 구조체 인자 (B)
  - geometricMask: 레이어가 클리핑되도록 하는 기하 마스크.
    - 디폴트 NULL.
    - 선택사항임. NULL로 지정하면 클리핑하지 않음.
    - 명시하는 기하(ID2D1Geometry)가 정의하는 영역에 의해 클리핑됨.
      - 해당 기하 내부만 보여짐
  - maskAntialiasMode: 클리핑에서 적용될 antialiasing 모드를 명시함.
    - 디폴트 on.
    - 클리핑은 geometricMask 필드에서 정의한 기하 마스크의 의해 수행됨
  - maskTransform: 레이어를 합성할 때에 기하 마스크에 적용될 변환.
    - 디폴트 IdentityMatrix()
    - 이 변환은 렌더타겟의 변환에 상대적으로 적용됨

레이어를 사용한 클리핑'

07.LayerGeometricMask

- 단계1: geometricMask 만들기

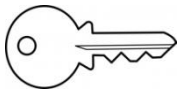
```
ID2D1PathGeometry* m_pPathGeometry;
m_pD2DFactory->CreatePathGeometry(&m_pPathGeometry);
ID2D1GeometrySink *pSink = NULL;
m_pPathGeometry->Open(&pSink);
pSink->SetFillMode(D2D1_FILL_MODE_WINDING);
pSink->BeginFigure(D2D1::Point2F(20, 50), D2D1_FIGURE_BEGIN_FILLED);
pSink->AddLine(D2D1::Point2F(130, 50));
pSink->AddLine(D2D1::Point2F(20, 130));
pSink->AddLine(D2D1::Point2F(80, 0));
pSink->AddLine(D2D1::Point2F(130, 130));
pSink->EndFigure(D2D1_FIGURE_END_CLOSED);
pSink->Close();
SafeRelease(&pSink);
```



레이어를 사용한 클리핑”

- 단계2: 기하 마스크를 사용하여 레이어에 그리기

```
ID2D1Layer* pLayer = NULL;
pRT->CreateLayer(NULL, &pLayer);
pRT->SetTransform(D2D1::Matrix3x2F::Translation(350, 50));
pRT->PushLayer( D2D1::LayerParameters(D2D1::InfiniteRect(), m_pPathGeometry), pLayer );
pRT->DrawBitmap(m_pOrigBitmap, D2D1::RectF(0, 0, 200, 133));
pRT->FillRectangle(D2D1::RectF(0.f, 0.f, 25.f, 25.f), m_pSolidColorBrush);
pRT->FillRectangle(D2D1::RectF(25.f, 25.f, 50.f, 50.f), m_pSolidColorBrush);
pRT->FillRectangle(D2D1::RectF(50.f, 50.f, 75.f, 75.f), m_pSolidColorBrush);
pRT->FillRectangle(D2D1::RectF(75.f, 75.f, 100.f, 100.f), m_pSolidColorBrush);
pRT->FillRectangle(D2D1::RectF(100.f, 100.f, 125.f, 125.f), m_pSolidColorBrush);
pRT->FillRectangle(D2D1::RectF(125.f, 125.f, 150.f, 150.f), m_pSolidColorBrush);
pRT->PopLayer();
SafeRelease(&pLayer);
```



불투명 마스크를 레이어에 적용하기

- 불투명마스크를 레이어에 적용할 경우
  - Layer에 그린 모든 모양들에 적용됨
- PushLayer 호출
  - ID2D1Layer 를 렌더타겟에 push함
  - 불투명 마스크로 붓을 사용할 수 있음
    - D2D1_LAYER_PARAMETERS 구조체 이용
- 효율성의 고려
  - 레이어는 비쌈
    - 단일 객체에 대해 불투명 마스크를 적용할 경우에는 FillOpacityMask 나 FillGeometry 를 사용하자

불투명 마스크를 레이어에 적용하기'

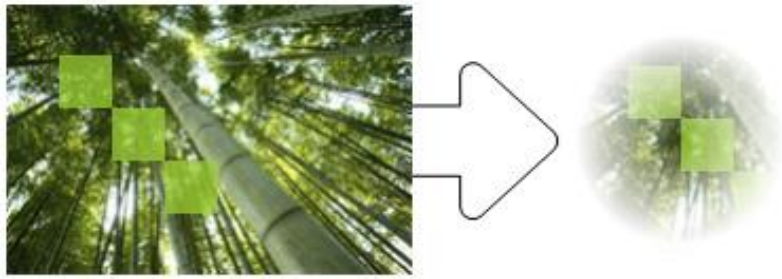
- 구조체 인자 (C)
  - opacity: 이 레이어의 불투명도
    - 디폴트 1.0.
    - 최종 불투명도 = opacity * "레이어에 그리는 각 객체의 불투명도"
  - opacityBrush: 레이어의 불투명도를 바꾸기 위한 붓.
    - 디폴트 NULL.
    - 선택 사항임. 불투명도 마스크와 무관하면 NULL로 지정.
    - 이 붓을 레이어에 매핑함
      - 최종 알파값 = "각 붓 픽셀의 알파값" * "대응되는 레이어 픽셀의 알파값"

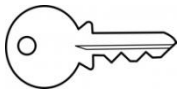
불투명 마스크를 레이어에 적용하기”

08.LayerOpacityMask

- 예: 붓을 불투명 마스크로 사용함
  - ID2D1RadialGradientBrush

```
ID2D1Layer* pLayer = NULL;
m_pRenderTarget->CreateLayer(NULL, &pLayer);
m_pRenderTarget->SetTransform(D2D1::Matrix3x2F::Translation(300, 250));
m_pRenderTarget->PushLayer(
        D2D1::LayerParameters( D2D1::InfiniteRect(), NULL,
                D2D1_ANTIALIAS_MODE_PER_PRIMITIVE, D2D1::IdentityMatrix(), 1.0,
                m_pRadialGradientBrush, D2D1_LAYER_OPTIONS_NONE ),
        pLayer );
m_pRenderTarget->DrawBitmap(m_pBambooBitmap, D2D1::RectF(0, 0, 190, 127));
m_pRenderTarget->FillRectangle( D2D1::RectF(25.f, 25.f, 50.f, 50.f), m_pSolidColorBrush );
m_pRenderTarget->FillRectangle( D2D1::RectF(50.f, 50.f, 75.f, 75.f), m_pSolidColorBrush );
m_pRenderTarget->FillRectangle( D2D1::RectF(75.f, 75.f, 100.f, 100.f), m_pSolidColorBrush );
m_pRenderTarget->PopLayer();
SafeRelease(&pLayer);
```





축정렬 클립 사각형으로 클리핑하기

- 레이어를 많이 사용하면:
  - 성능을 저하시킴
  - 사용하는 레이어 수를 최소한으로 유지하자
- 레이어의 contentBounds를 사용하여 클리핑하는 경우
  - 레이어는 비싸므로 바꾸자!
  - 상대적으로 저렴함 PushAxisAlignedClip/PopAxisAlignedClip 를 사용
- 축정렬(axis-aligned) 클립 사각형으로 이미지를 클립하는 방법
  - 단계1: 원본 이미지를 로드
  - 단계2: 클립 사각형을 명시
    - ID2D1RenderTarget::PushAxisAlignedClip 를 호출
    - 이후로의 그리기 작업들은 명시된 클립 사각형에 대해 클리핑됨
  - 단계3: 원본 이미지를 그리기
  - 단계4: 최근 클립 사각형을 렌더타겟으로부터 제거
    - ID2D1RenderTarget::PopAxisAlignedClip 를 호출

축정렬 클립 사각형으로 클리핑하기'

09.ClipWithAxisAlignedClip

- 클리핑 예:
  - 왼쪽: 원본 이미지: 200*133 픽셀
  - 오른쪽: 클릭 사각형 [(20,20),(100,100)]에 클리핑된 원본 이미지

```
pRT->PushAxisAlignedClip(
                D2D1::RectF(20, 20, 100, 100),
                D2D1_ANTIALIAS_MODE_PER_PRIMITIVE );
pRT->FillRectangle(D2D1::RectF(0, 0, 200, 133), m_pOriginalBitmapBrush);
pRT->PopAxisAlignedClip();
```