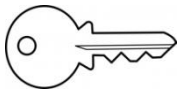
게임프로그래밍

변환

박종승

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



목차

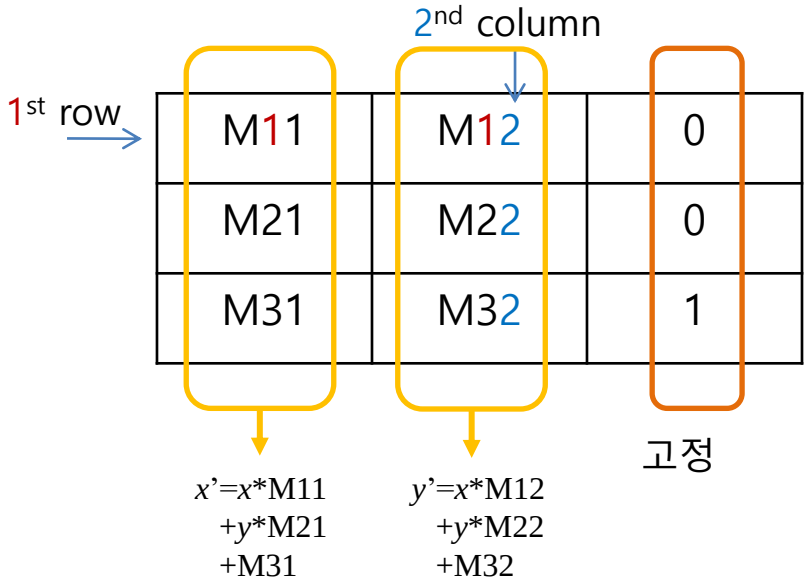
- [참고: “렌더타겟”→“자원”→this]
- D2D 변환이란?
- 변환행렬의 표현과 생성
- 렌더타겟에 변환을 적용
- 여러 변환들을 적용하기
- 붓에 변환 적용하기
- Geometry에 변환 적용하기
- 렌더타겟 변환이 clip에 미치는 영향

변환의 기능, 표현

- 변환의 기능
  - 한 좌표공간에서 정의된 한 객체의 점들을,
    동일 좌표공간에서의 다른 위치로 매핑
    또는, 다른 좌표공간에서의 위치로 매핑
- 변환의 표현
  - 행렬(transformation matrix): 선형변환을 표현
  - 3x3 FLOAT 값들

$$(x,y) \longrightarrow (x',y')$$

$$(x',y',1) = (x,y,1)*M$$



디폴트값

1	0	0
0	1	0
0	0	1

변환의 특성, 좌표 공간

- 변환행렬의 특성
  - scale, rotate, skew, translation

M11	M12	0
M21	M22	0
M31	M32	1

M11	M12	0
M21	M22	0
M31 OffsetX	M32 OffsetY	1

- D2D 좌표 공간
  - D2D는 왼손좌표계를 사용함
  - +x 방향은 오른쪽으로 향하고, +y 방향은 아래쪽으로 향함

변환행렬의 표현, 생성

- 변환행렬의 표현
  - 3x2 부분행렬
  - 구조체 : D2D1_MATRIX_3X2 구조체
  - 클래스 : Matrix3x2F
    - 디폴트 생성자 : 초기화하지 않음
    - 항등행렬(identity matrix) 얻기 : Matrix3x2F::Identity.
      - 항등행렬: 특성을 바꾸지 않는 변환
- 변환행렬의 생성 방법
  - Matrix3x2F 클래스가 여러 정적 함수들을 제공함
    - Rotation
    - Scale
    - Skew
    - Translation

렌더타겟에 변환 적용하기

- 렌더타겟
  - 좌표공간을 변환하는 함수를 제공함
  - ID2D1RenderTarget::SetTransform
    - 인자로 주어진 변환을 렌더타겟에 적용함
  - 이후로의 모든 그리기 연산들은 변환된 공간에서 이루어짐

- 렌더타겟에 그리기
  - 렌더타겟의 그리기 함수들을 사용
  - BeginDraw 함수와 EndDraw 함수 사이에 배치

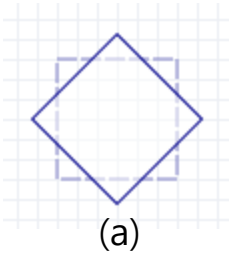
회전

02.HelloWorldRotated

- Matrix3x2F::Rotation

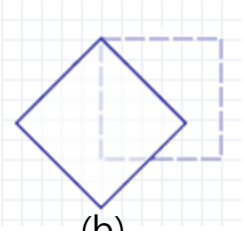
```
static Matrix3x2F Rotation( FLOAT angle,
                            D2D1_POINT_2F centerPoint = D2D1::Point2F() );
```

  - 인자1: 각도(시계방향으로의 회전각도; degree단위)
  - 인자2: 중심점(회전중심; 객체의 좌표계로 표현됨)
    - (a)회전중심점이 사각형 중심인 경우 vs.
    - (b)사각형 왼쪽상단모서리인 경우



(a)

점선=변환 전



(b)

- 예제
  - 사각형을 시계방향으로 사각형중심에 대해서 45도 회전

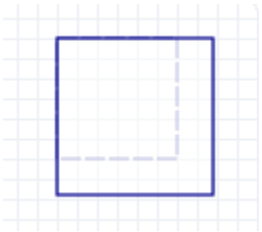
```
D2D1_RECT_F rectangle = D2D1::Rect(438.0f, 301.5f, 498.0f, 361.5f); //사각형 생성하기
m_pRenderTarget->DrawRectangle( rectangle, m_pOriginalShapeBrush, //사각형 그리기
            1.0f, m_pStrokeStyleDash );
m_pRenderTarget->SetTransform( //회전 변환을 렌더타겟에 적용
            D2D1::Matrix3x2F::Rotation( 45.0f, D2D1::Point2F(468.0f, 331.5f)) );
m_pRenderTarget->FillRectangle(rectangle, m_pFillBrush); //사각형 채우기
m_pRenderTarget->DrawRectangle(rectangle, m_pTransformedShapeBrush); //변환된 사각형 그리기
```

크기조정

- Matrix3x2F::Scale

  Matrix3x2F::Scale(D2D1_SIZE_F scalefactor, D2D1_POINT_2F centerpoint)
  Matrix3x2F::Scale(float scalex, float scaley, D2D1_POINT_2F centerpoint)

  - 인자1: scalex, scaley: x/y축으로의 두 개의 scale factor (1.5는 150%임)
  - 인자2: centerpoint 중심점(크기조정 중심)
    - 디폴트 (0,0)



- 예제
  - 한 정사각형을 130% 증가시킴.
  - 중심점은 원래 정사각형의 좌측상단 모서리

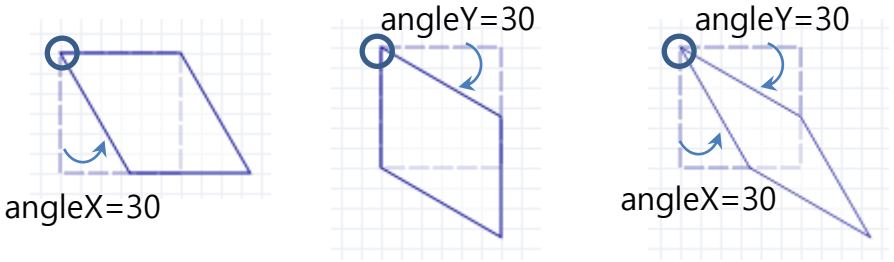
```
D2D1_RECT_F rectangle = D2D1::Rect(438.0f, 80.5f, 498.0f, 140.5f); //사각형 생성
m_pRenderTarget->DrawRectangle( rectangle, m_pOriginalShapeBrush, //사각형의 외곽선 그리기
            1.0f, m_pStrokeStyleDash );
m_pRenderTarget->SetTransform(  //렌더타겟에 크기조정 변환을 적용
            D2D1::Matrix3x2F::Scale( D2D1::Size(1.3f, 1.3f),
                                     D2D1::Point2F(438.0f, 80.5f)) );
m_pRenderTarget->FillRectangle(rectangle, m_pFillBrush); //사각형의 내부 색칠
m_pRenderTarget->DrawRectangle(rectangle, m_pTransformedShapeBrush); //사각형의 외곽선 그리기
```

기울임

- Matrix3x2F::Skew

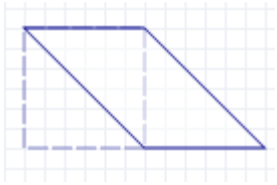
```
static Matrix3x2F Matrix3x2F::Skew( FLOAT angleX, FLOAT angleY,
        D2D1_POINT_2F centerPoint = D2D1::Point2F() );
```

  - 인자1: angleX, angleY: x/y축으로의 기울임 각도 (degree단위)
    - angleX는 y-축에서부터 반시계방향, angleY는 x-축에서부터 시계방향으로 측정
  - 인자2: centerPoint: 기울임이 수행되는 지점
  - 기울임(skew 또는 shear)은 x/y-축으로 주어진 각도만큼 기울임을 의미함
  - 예시
    - 세 경우 모두, centerPoint는 왼쪽 상단 모서리



기울임'

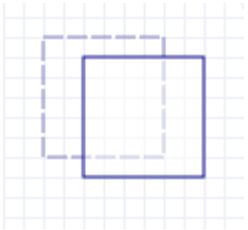
- 예제
  - angleX=45, centerPoint=왼쪽상단 모서리
    - y-축으로부터 반시계방향으로 45도 기울임



```
D2D1_RECT_F rectangle = D2D1::Rect(126.0f, 301.5f, 186.0f, 361.5f); //사각형을 생성
m_pRenderTarget->DrawRectangle( rectangle, m_pOriginalShapeBrush, //사각형의 외곽선을 그림
            1.0f, m_pStrokeStyleDash );

m_pRenderTarget->SetTransform( //기울임 변환을 렌더타겟에 적용함
            D2D1::Matrix3x2F::Skew( 45.0f, 0.0f, D2D1::Point2F(126.0f, 301.5f)) );

m_pRenderTarget->FillRectangle(rectangle, m_pFillBrush); //사각형의 내부를 색칠함
m_pRenderTarget->DrawRectangle(rectangle, m_pTransformedShapeBrush); //사각형의 외곽선을 그림
```

이동

- Matrix3x2F::Translation



```
static Matrix3x2F Matrix3x2F::Translation( D2D1_SIZE_F size );
static Matrix3x2F Matrix3x2F::Translation( FLOAT x, FLOAT y );
```

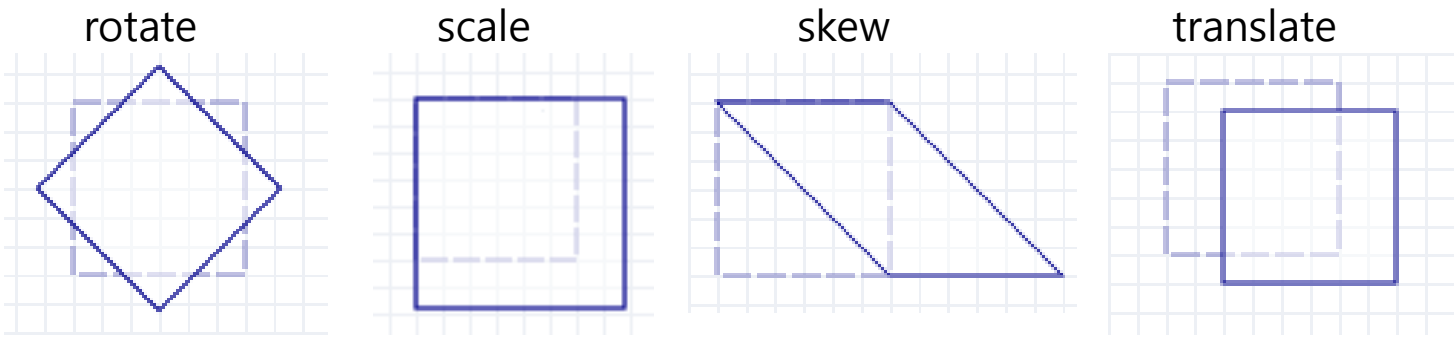
  - 인자1: transX, transY: x/y축으로의 이동 거리
    - transX는 오른쪽으로 이동, transY는 아래쪽으로 이동
- 예제
  - 정사각형을 x-축을 따라 오른쪽으로 20, y축을 따라 아래쪽으로 10만큼 이동

```
D2D1_RECT_F rectangle = D2D1::Rect(126.0f, 80.5f, 186.0f, 140.5f); //사각형을 생성
m_pRenderTarget->DrawRectangle( rectangle, m_pOriginalShapeBrush, //사각형의 외곽선 그리기
            1.0f, m_pStrokeStyleDash );

m_pRenderTarget->SetTransform( D2D1::Matrix3x2F::Translation(20, 10) ); //렌더타겟에 이동 변환 적용

m_pRenderTarget->FillRectangle(rectangle, m_pFillBrush); //사각형의 내부 색칠하기
m_pRenderTarget->DrawRectangle(rectangle, m_pTransformedShapeBrush); //사각형의 외곽선 그리기
```

단일 변환 예제

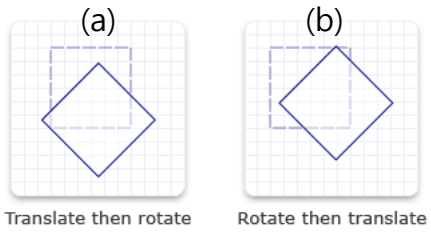
03.SingleTransforms

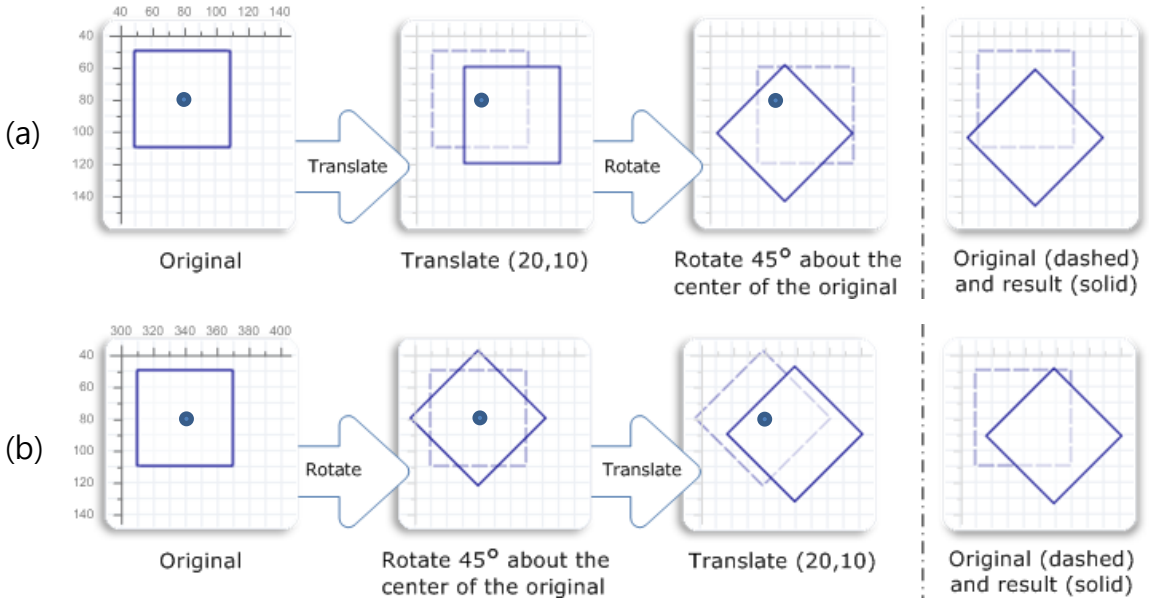
- 단일 변환 예제
  - rotate, scale, skew, translate



여러 변환들을 적용하기

- 여려 변환들을 적용하기
  - "여러 변환들을 하나의 변환으로 결합하기"를 의미
  - 행렬의 적용 순서 (즉 곱셈 순서)가 중요함
  - 예: (a) 이동한 다음 회전 vs. (b) 회전한 다음 이동





여러 변환들을 적용하기'

04.MultipleTransforms

- 예제

(a)

```
D2D1_RECT_F rectangle = D2D1::Rect(40.0f, 40.0f, 100.0f, 100.0f);
m_pRenderTarget->DrawRectangle( rectangle, m_pOriginalShapeBrush, 1.0f, m_pStrokeStyleDash );

D2D1_MATRIX_3X2_F translation = D2D1::Matrix3x2F::Translation(20.0f, 10.0f);
D2D1_MATRIX_3X2_F rotation = D2D1::Matrix3x2F::Rotation( 45.0f, D2D1::Point2F(70.0f, 70.0f) );

// 먼저 (20,10) 만큼 이동하고, 그 다음, 원래 사각형의 중심에 대해서 회전함
m_pRenderTarget->SetTransform( translation * rotation );

// 변환된 공간에서 사각형 그리기
m_pRenderTarget->FillRectangle(rectangle, m_pFillBrush);
m_pRenderTarget->DrawRectangle(rectangle, m_pTransformedShapeBrush);

m_pRenderTarget->SetTransform(D2D1::Matrix3x2F::Identity()); //변환을 리셋
```

(b)

```
D2D1_RECT_F rectangle = D2D1::RectF(300.0f, 40.0f, 360.0f, 100.0f);
m_pRenderTarget->DrawRectangle( rectangle, m_pOriginalShapeBrush, 1.0f, m_pStrokeStyleDash );

D2D1_MATRIX_3X2_F rotation = D2D1::Matrix3x2F::Rotation( 45.0f, D2D1::Point2F(330.0f, 70.0f) );
D2D1_MATRIX_3X2_F translation = D2D1::Matrix3x2F::Translation(20.0f, 10.0f);
// 먼저 사각형 중심에 대해서 회전하고, 그 다음, (20,10) 만큼 이동함
m_pRenderTarget->SetTransform( rotation* translation );

// 변환된 공간에서 사각형 그리기
m_pRenderTarget->FillRectangle(rectangle, m_pFillBrush);
m_pRenderTarget->DrawRectangle(rectangle, m_pTransformedShapeBrush, 1.0f);
```

붓에 변환 적용하기

- 붓에 변환을 적용하기
  - ID2D1Brush::SetTransform 함수를 호출
- 개념 설명
  - 붓(brush)=벽지로 이해하면 유사함
  - 여러 렌더링 객체들(텍스트,geometry,사각형 등)=스텐실
  - 붓에 변환을 적용하면, 스텐실 하단에 있는 벽지를 움직이는 것과 같음
    - 스텐실의 위치는 고정되어 있음
  - 적용 사례: 렌더링 객체의 외양이 fade되는 효과를 쉽게 구현할 수 있음
- 붓 변환
  - 붓 좌표들이 어떻게 렌더타겟의 좌표공간에 매핑될 지를 명시함
    - 붓 변환이 항등 변환(identity transform)인 경우: 붓은 렌더타겟의 좌표공간과 동일하게 나타남
- 붓 공간
  - 붓 공간 = "렌더타겟의 현재 좌표계"에 "붓 변환"을 적용한 공간임
  - 참고: 붓 공간은 그릴 객체에 상대적이지 않음
    - 붓을 객체에 상대적으로 채우기 위해서는,
      - 붓 공간의 원점을 물체의 bounding box의 왼쪽 상단 모서리로 translate하고,
      - 기본 타일이 객체의 bounding box를 가득 채우도록 붓 공간을 scaling.

Geometry에 변환 적용하기

- Geometry에 변환 적용하기
  - 특정 geometry에 대해서만 변환을 적용할 수 있음
  - 그 geometry의 좌표들을 변환함
- 비교
  - 렌더타겟에 적용하는 변환
    - Geometry의 좌표 및 stroke 등 모든 것에 영향을 미침
  - Geometry에 적용하는 변환
    - 그 geometry의 좌표에만 영향을 미침
      - 그리는 단계에서 적용되는 stroke 관련 내용(stroke thickness등)은 바뀌지 않음
- Geometry에 변환을 적용하는 방법
  - ID2D1Factory::CreateTransformedGeometry 를 호출하여 ID2D1TransformedGeometry 객체를 생성

렌더타겟 변환이 클리핑에 미치는 영향

- 클리핑
  - ID2D1RenderTarget::PushAxisAlignedClip
    - 호출 이후의 그리기 연산들은 인자로 명시된 clipRect에 대해 클리핑됨
- 렌더타겟 변환이 클리핑에 미치는 영향
  - PushAxisAlignedClip가 호출되면,
    - clipRect 인자가 현재의 렌더타겟 변환으로 변환되고,
    - 변환후의 clipRect 에 대해서, AABB가 계산됨
      - 용어: AABB: 축정렬경계상자; axis-aligned bounding box;
    - 그릴 객체들은 (원래의 clipRect가 아니라) 계산된 AABB에 의해 클리핑됨
      - 효율성을 고려하였음
- 주의
  - PopAxisAlignedClip의 호출
    - 명시되었던 clipRect를 렌더타겟으로부터 회수함
      - 더 이상 클리핑되지 않음
    - PushAxisAlignedClip에 매치되도록 호출되어야 함

렌더타겟 변환이 클리핑에 미치는 영향’

- 예시: 한 회전 변환이 렌더타겟에 적용된 경우
  - 1. 검은색 사각형은 렌더타겟을 의미함
  - 2. 회전 변환을 렌더타겟에 적용.
    - 붉은색 점선 사각형은 변환된 후의 렌더타겟임
  - 3. PushAxisAlignedClip가 호출된 후, 회전 변환이 clipRect에 적용됨
    - 파란색 사각형이 변환된 clipRect임.
  - 4. AABB가 계산됨
    - 녹색 점선 사각형이 계산된 AABB임
    - 모든 그릴 객체들은 이 AABB에 대해서 클리핑됨