게임프로그래밍

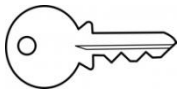
기하 3

박종승

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

목차

- 복합 기하, 기하 그룹
- 기하 그룹 생성의 예
- 참고: 채우기 모드
- 변환된 기하
- 기하를 변환하는 예제
- 기하 연산
- 기하들의 결합



복합 기하, 기하 그룹

- 복합 기하(composite geometry)
  - 기하들을 묶은(group) 기하 또는 기하들을 결합(combine)한 기하
- 복합 기하의 표현
  - 기하 그룹 - ID2D1GeometryGroup

- 기하 그룹
  - 여러 기하들을 하나의 기하로 묶음
  - ID2D1GeometryGroup 를 생성하는 방법
    - ID2D1Factory::CreateGeometryGroup 함수를 호출
      - 인자1: fillMode (나중에 설명)
      - 인자2,3: 기하 그룹에 추가할 기하 객체들의 배열; 기하 객체들의 개수;

기하 그룹 생성의 예

- 예: 두 기하 그룹을 생성함
  - 기하 객체들(네 개의 원)의 배열을 선언
  - 채우기 모드를 달리 하여 기하 그룹을 생성

04.GeometryGroupCircles


```
const D2D1_ELLIPSE ellipse1 = D2D1::Ellipse( D2D1::Point2F(105.0f, 105.0f), 25.0f, 25.0f );
m_pD2DFactory->CreateEllipseGeometry( ellipse1, &m_pEllipseGeometry1 );
const D2D1_ELLIPSE ellipse2 = D2D1::Ellipse( D2D1::Point2F(105.0f, 105.0f), 50.0f, 50.0f );
m_pD2DFactory->CreateEllipseGeometry( ellipse2, &m_pEllipseGeometry2 );
const D2D1_ELLIPSE ellipse3 = D2D1::Ellipse( D2D1::Point2F(105.0f, 105.0f), 75.0f, 75.0f);
m_pD2DFactory->CreateEllipseGeometry( ellipse3, &m_pEllipseGeometry3 );
const D2D1_ELLIPSE ellipse4 = D2D1::Ellipse( D2D1::Point2F(105.0f, 105.0f), 100.0f, 100.0f);
m_pD2DFactory->CreateEllipseGeometry( ellipse4, &m_pEllipseGeometry4 );

ID2D1Geometry *ppGeometries[] = { m_pEllipseGeometry1, m_pEllipseGeometry2,
            m_pEllipseGeometry3, m_pEllipseGeometry4 };

m_pD2DFactory->CreateGeometryGroup( D2D1_FILL_MODE_ALTERNATE,
            ppGeometries, ARRAYSIZE(ppGeometries), &m_pGeoGroup_AlternateFill );

m_pD2DFactory->CreateGeometryGroup( D2D1_FILL_MODE_WINDING,
            ppGeometries, ARRAYSIZE(ppGeometries), &m_pGeoGroup_WindingFill );
```

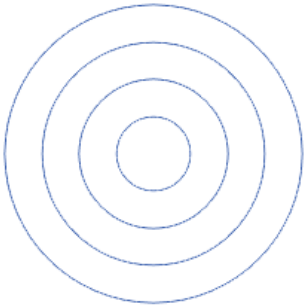
참고: 채우기 모드

- 채우기 모드의 지정
  - 채우기 영역을 결정하는 방법을 명시함
  - ID2D1GeometrySink::SetFillMode 함수 호출
    - BeginFigure 함수 호출 이전에 호출할 것
  - 채우기의 두 가지 모드: alternate 또는 winding
    - D2D1_FILL_MODE_ALTERNATE (디폴트)
      - 해당 점에서 임의의 방향으로 무한히 확장했을 때에, 그 광선이 교차하는 조각(segment)들의 개수를 셈.
      - 홀수이면 해당 점은 채우기 영역 내부에 있음.
    - D2D1_FILL_MODE_WINDING
      - 해당 점에서 임의의 방향으로 무한히 확장했을 때에, 그 광선이 한 조각(segment)을 교차하는 지점을 찾음.
      - 0부터 시작하여 카운트 시작:
      - 조각이 광선을 왼쪽에서 오른쪽으로(오른쪽에서 왼쪽으로) 교차하면 +1(-1)
        - » 왼쪽,오른쪽은 광선의 진행방향 입장에서
      - 카운트 결과가 0이 아니면 해당 지점은 채우기 영역 내부에 있음

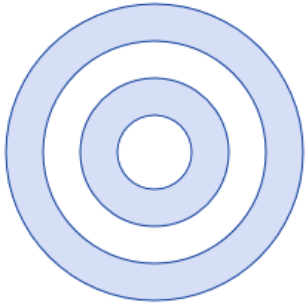
참고: 채우기 모드'

- 예

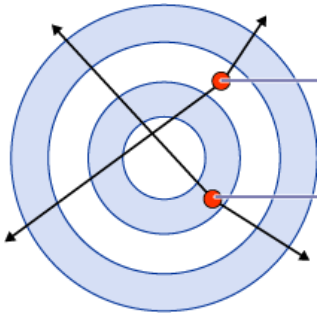
04.GeometryGroupCircles

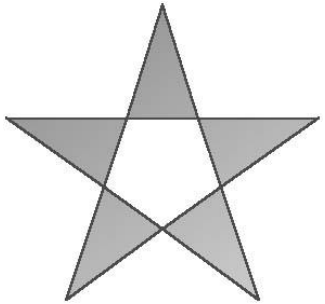


alternate

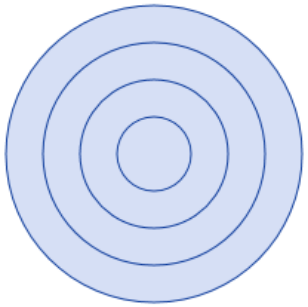


D2D1_FILL_MODE_ALTERNATE

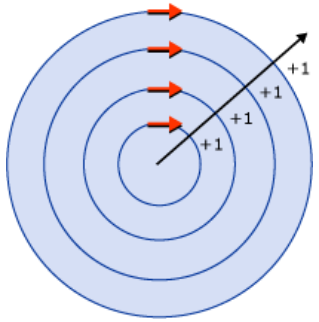


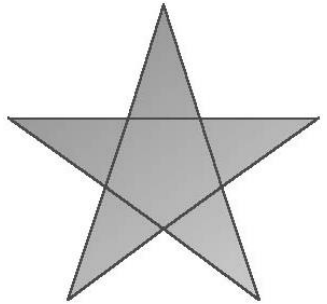


winding

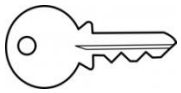


D2D1_FILL_MODE_WINDING





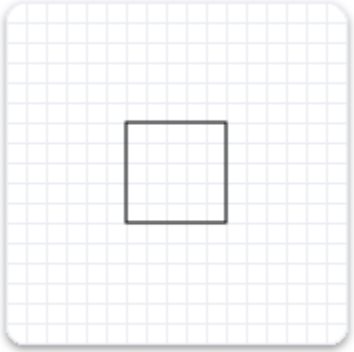
계산 원리



변환된 기하

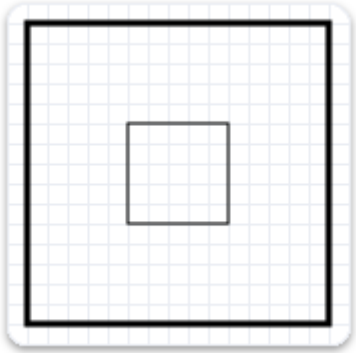
- 기하를 변환하는 방법
  - 방법1: 렌더타겟이 그리는 모든 것들을 변환
    - 렌더타겟의 SetTransform 함수
    - 모든 것에 영향을 미침. 획의 두께도 변환됨.
  - 방법2: 해당 기하에 직접 명시
    - ID2D1Factory::CreateTransformedGeometry 함수를 호출
      - ID2D1TransformedGeometry 생성
    - 모양의 좌표값들에만 영향을 미침. 획의 두께는 변환되지 않음.
- 예1
  - 사각형 기하(ID2D1RectangleGeometry)를 생성하여 변환 없이 그리기

```
m_pD2DFactory->CreateRectangleGeometry(
                D2D1::RectF(150.f, 150.f, 200.f, 200.f),
                &m_pRectangleGeometry );
m_pRenderTarget->DrawGeometry(
                m_pRectangleGeometry, m_pBlackBrush, 1 );
```



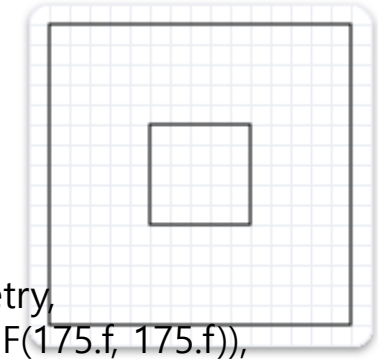
변환된 기하'

- 예2
  - 렌더타겟을 변환하여 사각형을 3배 크기조정
  - 굵은 선 사각형: 변환을 적용한 사각형
    - 획의 두께가 동일하게 1이지만, 더 두꺼움



```
m_pRenderTarget->SetTransform(
            D2D1::Matrix3x2F::Scale( D2D1::SizeF(3.f, 3.f),
                                     D2D1::Point2F(175.f, 175.f)) );
m_pRenderTarget->DrawGeometry( m_pRectangleGeometry, m_pBlackBrush, 1 );
```

- 예3
  - 기하를 변환하여 사각형을 3배 크기조정
    - CreateTransformedGeometry 함수를 호출
    - 사각형 크기만 커지고 획은 변함이 없음



```
m_pD2DFactory->CreateTransformedGeometry( m_pRectangleGeometry,
            D2D1::Matrix3x2F::Scale( D2D1::SizeF(3.f, 3.f), D2D1::Point2F(175.f, 175.f)),
            &m_pTransformedGeometry );

m_pRenderTarget->SetTransform(D2D1::Matrix3x2F::Identity()); //이전 변환을 초기화
m_pRenderTarget->DrawGeometry( m_pTransformedGeometry, m_pBlackBrush, 1 );
```

기하를 변환하는 예제

- 기하를 변환

05.TransformGeometryHourglass

```
// 한 경로 기하를 생성
m_pD2DFactory->CreatePathGeometry(&m_pPathGeometry);

// 기하 싱크를 이용하여 경로 기하에 쓰기 시작
m_pPathGeometry->Open(&pSink);
pSink->BeginFigure(D2D1::Point2F(0, 0), D2D1_FIGURE_BEGIN_FILLED );
pSink->AddLine(D2D1::Point2F(200, 0));
pSink->AddBezier(D2D1::BezierSegment( D2D1::Point2F(150, 50),D2D1::Point2F(150, 150),D2D1::Point2F(200, 200)) );
pSink->AddLine(D2D1::Point2F(0, 200));
pSink->AddBezier(D2D1::BezierSegment( D2D1::Point2F(50, 150),D2D1::Point2F(50, 50),D2D1::Point2F(0, 0)));
pSink->EndFigure(D2D1_FIGURE_END_CLOSED);
pSink->Close();
SafeRelease(&pSink);

// 변환된 기하를 생성. 원본을 45도 시계방향으로 회전.
m_pD2DFactory->CreateTransformedGeometry( m_pPathGeometry,
                D2D1::Matrix3x2F::Rotation(45.f, D2D1::Point2F(100,100)),
                &m_pTransformedGeometry );
```

기하를 변환하는 예제'

- 붓 준비 ("붓" 파트에서 설명)

```
// 검정색 단색 붓
m_pRenderTarget->CreateSolidColorBrush(
                D2D1::ColorF(D2D1::ColorF::Black), &m_pBlackBrush );

// 선형 계조 붓
static const D2D1_GRADIENT_STOP stops[] = {
                {   0.f,  { 0.f, 1.f, 1.f, 0.25f }  },
                {   1.f,  { 0.f, 0.f, 1.f, 1.f }  },  };
m_pRenderTarget->CreateGradientStopCollection( stops, ARRAYSIZE(stops), &pGradientStops );
m_pRenderTarget->CreateLinearGradientBrush(
            D2D1::LinearGradientBrushProperties( D2D1::Point2F(100, 0), D2D1::Point2F(100, 200)),
            D2D1::BrushProperties(), pGradientStops, &m_pLGBrush );
SafeRelease(&pGradientStops);
```

기하를 변환하는 예제”

- 변환된 기하를 그리기
  - DrawGeometry, FillGeometry 함수

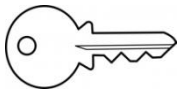
```
// 클라이언트 영역 계산하기
RECT rc;
GetClientRect(m_hwnd, &rc);
D2D1_SIZE_U size = D2D1::SizeU( rc.right - rc.left, rc.bottom - rc.top );

// 이제부터 그리는 것은 20만큼 이동 변환됨
m_pRenderTarget->SetTransform( D2D1::Matrix3x2F::Translation(20.f, 20.f) );

// 왼쪽상단에 모래시계 모양을 그림
m_pRenderTarget->DrawGeometry(m_pPathGeometry, m_pBlackBrush, 10.f);
m_pRenderTarget->FillGeometry(m_pPathGeometry, m_pLGBrush);

// 이제부터 그리는 것은 절반크기로 화면중심 오른쪽하단에 그림
m_pRenderTarget->SetTransform(
                D2D1::Matrix3x2F::Scale( D2D1::SizeF(0.5f, 0.5f), D2D1::Point2F(0.f, 0.f) ) *
                D2D1::Matrix3x2F::Translation( size.width/2 - 50.f, size.height/2 - 50.f)  );

// 원래 기하와 변환된 기하를 그림
m_pRenderTarget->FillGeometry(m_pPathGeometry, m_pLGBrush);
m_pRenderTarget->FillGeometry(m_pTransformedGeometry, m_pLGBrush);
```



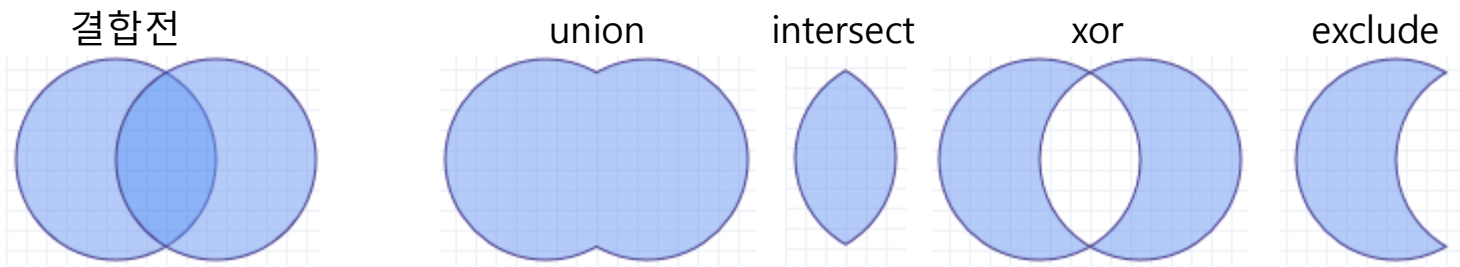
기하 연산

- 인터페이스 ID2D1Geometry는 여러 기하 연산 함수들을 제공함
  - 기하를 조작하거나 측정하는 기능
- 함수들
  - CombineWithGeometry
  - ComputeLength; ComputeArea, ComputePointAtLength
  - FillContainsPoint, StrokeContainsPoint
  - CompareWithGeometry
  - 기타1
    - GetBounds
    - GetWidenedBounds
    - Tessellate
  - 기타2
    - Simplify
    - Outline
    - Widen

기하들의 결합

06.CombineGeometryCircles

- CombineWithGeometry
  - 결합 옵션
    - D2D1_COMBINE_MODE_UNION (union)
      - the union of both geometries
    - D2D1_COMBINE_MODE_INTERSECT (intersect)
      - the overlapping region between the two geometries
    - D2D1_COMBINE_MODE_XOR (xor)
      - the region that is (*A*-*B*) + (*B*-*A*)
    - D2D1_COMBINE_MODE_EXCLUDE (exclude)
      - the region that is *A*-*B*



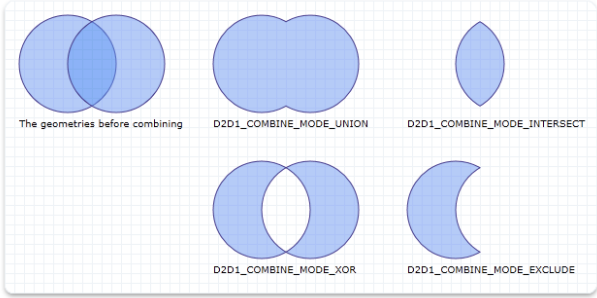
기하들의 결합 예제

- CombineWithGeometry 예

06.CombineGeometryCircles

```
// 첫 번째 원
const D2D1_ELLIPSE circle1 = D2D1::Ellipse( D2D1::Point2F(75.0f, 75.0f), 50.0f, 50.0f );
m_pD2DFactory->CreateEllipseGeometry( circle1, &m_pCircleGeometry1 );
// 두 번째 원
const D2D1_ELLIPSE circle2 = D2D1::Ellipse( D2D1::Point2F(125.0f, 75.0f), 50.0f, 50.0f );
m_pD2DFactory->CreateEllipseGeometry(circle2, &m_pCircleGeometry2);

// 두 기하를 결합
m_pD2DFactory->CreatePathGeometry(&m_pPathGeometryUnion);
ID2D1GeometrySink* pGeometrySink = NULL;
m_pPathGeometryUnion->Open(&pGeometrySink);
m_pCircleGeometry1->CombineWithGeometry( m_pCircleGeometry2,
            D2D1_COMBINE_MODE_UNION,
            NULL, NULL, pGeometrySink );
pGeometrySink->Close();
SafeRelease(&pGeometrySink);
```



기하의 길이, 면적 계산

06.CombineGeometryCircles

- ComputeLength
  - 기하의 길이를 계산함
    - 한 라인으로 쭉 폈을 때의 길이임
    - 닫힌 기하의 경우, 마지막의 닫는 세그먼트도 길이에 포함시킴

```
float length;
m_pCircleGeometry1->ComputeLength( D2D1::IdentityMatrix(), &length );
```

- ComputeArea
  - 기하의 면적을 계산함

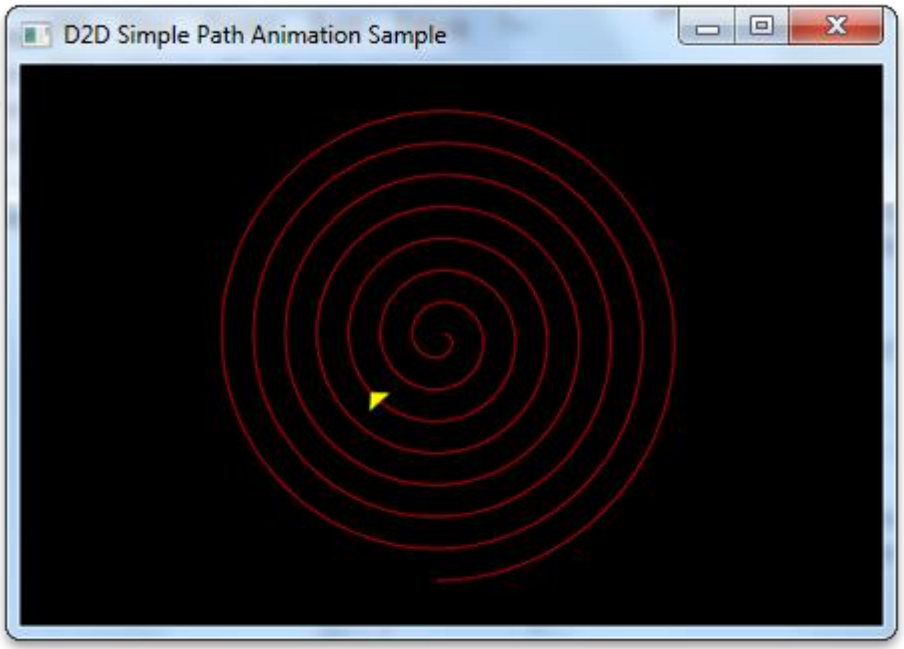
```
float area;
m_pCircleGeometry1->ComputeArea( D2D1::IdentityMatrix(), &area );
```

- ComputePointAtLength
  - 기하를 따라갈 때에 명시한 거리에서의 위치와 접선(tangent)벡터
    - 두 번째 인자에 행렬이 주어지면, 주어진 행렬로 변환한 후에 계산함

```
D2D1_POINT_2F point;
D2D1_POINT_2F tangent;
FLOAT length = 10;
m_pPathGeometry->ComputePointAtLength( length, NULL, &point, &tangent);
```

선택: 고급 예제

- 예제(중상): Simple Path Animation Sample 24.SimplePathAnimationSample
  - 한 경로를 따라가면서 한 기하를 애니메이션 함.
  - 함수 사용 예시: ComputeLength, ComputePointAtLength
  - 설명생략 (참고: link: MSDN)



기하가 주어진 점을 포함하는지의 여부 검사

- FillContainsPoint
  - 기하를 채운 영역이 명시한 점을 포함하는지의 여부
    - 사용 예: 충돌 테스팅을 위해 사용
  - 인자2: 테스트 전에 기하에 적용할 변환

```
BOOL containsPoint1;
m_pCircleGeometry1->FillContainsPoint( D2D1::Point2F(0,0), D2D1::Matrix3x2F::Identity(),
                                       &containsPoint1 );
if (containsPoint1) { /* contains */ }
```

- StrokeContainsPoint
  - 기하의 획이 명시한 점을 포함하는지의 여부
    - 사용 예: 충돌 테스팅을 위해 사용
  - 인자2,인자3: strokeWidth, strokeStyle
  - 인자4: 테스트 전에 기하에 적용할 변환

```
BOOL containsPoint1;
m_pCircleGeometry1->StrokeContainsPoint( D2D1::Point2F(0,0),
                10, NULL, NULL, &containsPoint1 );
if (containsPoint1) { /* contains */ }
```

두 기하의 교차 관계 판단

06.CombineGeometryCircles

- CompareWithGeometry
  - 해당 기하와 인자로 주어진 기하의 교차 관계를 판단함
  - 교차 관계의 종류
    - D2D1_GEOMETRY_RELATION_DISJOINT (disjoint)
      - 두 기하가 전혀 교차하지 않음
    - D2D1_GEOMETRY_RELATION_IS_CONTAINED (is contained)
      - 해당 기하가 인자로 주어진 기하의 내부에 완전히 포함되어 있음
    - D2D1_GEOMETRY_RELATION_CONTAINS (contains),
      - 해당 기하가 인자로 주어진 기하를 완전히 포함하고 있음
    - D2D1_GEOMETRY_RELATION_OVERLAP (overlap)
      - 두 기하가 중첩되지만 서로 완전히 포함되는 관계는 아님
  - 예

```
D2D1_GEOMETRY_RELATION result = D2D1_GEOMETRY_RELATION_UNKNOWN;
m_pCircleGeometry1->CompareWithGeometry( m_pCircleGeometry2,
                D2D1::IdentityMatrix(), 0.1f, &result );
if (result == D2D1_GEOMETRY_RELATION_OVERLAP) { /* Two circles overlap. */ }
```

기타1, 기타2

- 기타1
  - GetBounds
    - 기하의 바운드 사각형을 얻는다.
  - GetWidenedBounds
    - 주어진 strokeWidth와 strokeStyle로 기하의 폭을 넓힌 다음의 기하의 바운드 사각형을 얻는다.
  - Tessellate
    - 기하를 커버하는 삼각형들을 생성함.
- 기타2
  - Simplify
    - 기하로부터 원호나 2차베지어곡선을 제거하여 선분들만 가지도록 함
  - Outline
    - 기하의 외곽선을 계산함. 자기가 자신을 교차하지 않는 외곽선임
  - Widen
    - 명시한 획 만큼 기하의 폭을 넓힘.