게임프로그래밍

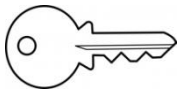
기하 1

박종승

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

목차

- 기하 개요
- 단순한 기하
- 단순한 모양을 그리기
- 획 스타일 속성값
- 경로 기하
- 복잡한 그림 생성하기 예제



기하 개요

- D2D 기하(geometry)란?
  - 기본 그리기 모양들임
  - ID2D1Geometry를 상속한 인터페이스의 객체
- 기하의 종류
  - 단순한 기하
    - 사각형, 모서리가 둥근 사각형, 타원, 원
  - 경로 기하(path geometry)
    - 원호, 곡선, 선분 등의 조각들로 구성된 복잡한 기하 모양
  - 복합 기하(composite geometry)
    - 기하들을 묶은 기하 또는 기하들을 결합한 기하

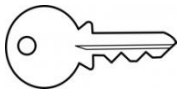
기하 개요'

- D2D 기하의 특징
  - 장치 독립적 자원임
    - ID2D1Factory 를 사용하여 생성함
  - Immutable (생성 후에는 수정 불가)
- 한 기하의 생성 방법
  - ID2D1Factory::CreateXXXGeometry 함수 호출
    - XXX=Rectangle,Path,…
- 한 기하를 렌더링하는 방법
  - 렌더타겟의 DrawGeometry 또는 FillGeometry 함수를 호출

```
virtual void ID2D1RenderTarget::DrawGeometry(
                ID2D1Geometry* geometry,
                ID2D1Brush* brush,
                FLOAT strokeWidth = 1.0f,
                ID2D1StrokeStyle* strokeStyle = NULL );
virtual void ID2D1RenderTarget::FillGeometry(
                ID2D1Geometry* geometry,
                ID2D1Brush* brush,
                ID2D1Brush* opacityBrush = NULL );
```

기하 개요”

- 기하 렌더링의 성능 측면
  - 가급적 더욱 구체적인 그리기 함수를 호출할 것
  - 알려진 기하를 활용할 수 있으므로 렌더링이 더 빠름
    - 예: DrawGeometry 함수를 호출하는 것보다 DrawRectangle를 호출할 것



단순한 기하

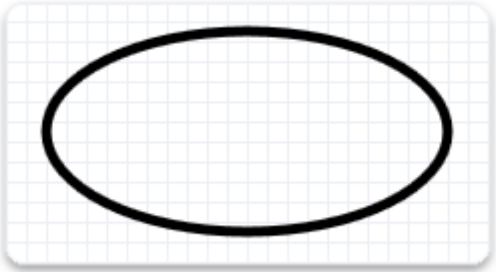
- 단순한 기하
  - ID2D1RectangleGeometry : 사각형
  - ID2D1RoundedRectangleGeometry : 모서리가 둥근 사각형
  - ID2D1EllipseGeometry : 타원, 원

- 단순한 기하의 생성
  - ID2D1Factory::CreateXXXGeometry 함수를 호출
    - XXX=Rectangle,RoundedRectagle,Ellipse
  - 사각형: ID2D1Factory::CreateRectangleGeometry
    - ID2D1RectangleGeometry 객체를 리턴함
  - 둥근 사각형: ID2D1Factory::CreateRoundedRectangleGeometry
    - ID2D1RoundedRectangleGeometry 객체를 리턴함
  - 타원: ID2D1Factory::CreateEllipseGeometry
    - ID2D1EllipseGeometry 객체를 리턴함

단순한 기하'

- 예: 타원을 그리는 예제
  - CreateEllipseGeometry
    - 인자: 중심이 (100, 60)이고, x-/y-반경이 100/50인 타원 구조체
  - DrawGeometry 호출
    - 인자: 타원 기하, 검정 붓(ID2D1SolidColorBrush), stroke width 5

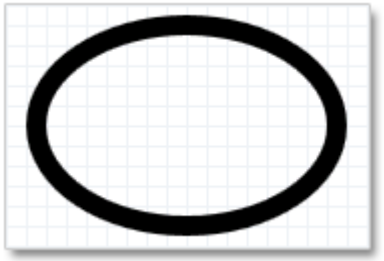
```
ID2D1EllipseGeometry* m_pEllipseGeometry;
hr = m_pD2DFactory->CreateEllipseGeometry(
            D2D1::Ellipse( D2D1::Point2F(100.f, 60.f), 100.f, 50.f ),
            &m_pEllipseGeometry );
m_pRenderTarget->DrawGeometry(m_pEllipseGeometry, m_pBlackBrush, 5);
```



단순한 기하를 그리기

- 타원의 외곽선을 실선 획(solid stroke)으로 그리기
  - 필요한 정보를 준비:
    - 붓: 예: ID2D1SolidColorBrush, ID2D1LinearGradientBrush
    - 타원: D2D1_ELLIPSE 구조체
  - 함수 호출:
    - ID2D1RenderTarget::DrawEllipse
  - 예:
    - 검정색 실선 붓을 생성해서 m_pBlackBrush 에 저장
    - 구조체 D2D1_ELLIPSE를 정의
    - 붓과 타원 구조체를 인자로 사용하여 함수 호출

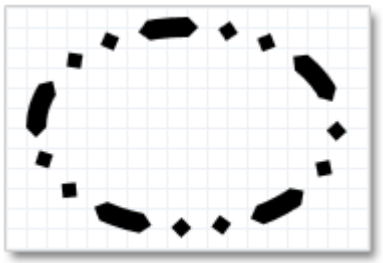
```
m_pRenderTarget->CreateSolidColorBrush(
            D2D1::ColorF(D2D1::ColorF::Black), &m_pBlackBrush );
D2D1_ELLIPSE ellipse = D2D1::Ellipse( D2D1::Point2F(100.f, 100.f), 75.f, 50.f );
m_pRenderTarget->DrawEllipse(ellipse, m_pBlackBrush, 10.f);
```



단순한 기하를 그리기'

- 타원의 외곽선을 점선 획(dashed stroke)으로 그리기
  - ID2D1StrokeStyle를 생성해서 획(stroke)의 외양을 수정할 수 있음
  - 예:
    - 점선 획을 명시하는 ID2D1StrokeStyle 생성
    - DrawEllipse 함수를 호출하여 그리기

```
D2D1_STROKE_STYLE_PROPERTIES strokeStyleProperties = D2D1::StrokeStyleProperties(
    D2D1_CAP_STYLE_FLAT,  D2D1_CAP_STYLE_FLAT,  // The start cap, the end cap.
    D2D1_CAP_STYLE_TRIANGLE, // The dash cap.
    D2D1_LINE_JOIN_MITER, // The line join.
    10.0f, // The miter limit.
    D2D1_DASH_STYLE_DASH_DOT_DOT, // The dash style.
    0.0f // The dash offset.
    );
m_pDirect2dFactory->CreateStrokeStyle(strokeStyleProperties, NULL, 0, &m_pStrokeStyle);
m_pRenderTarget->DrawEllipse(ellipse, m_pBlackBrush, 10.f, m_pStrokeStyle);
```



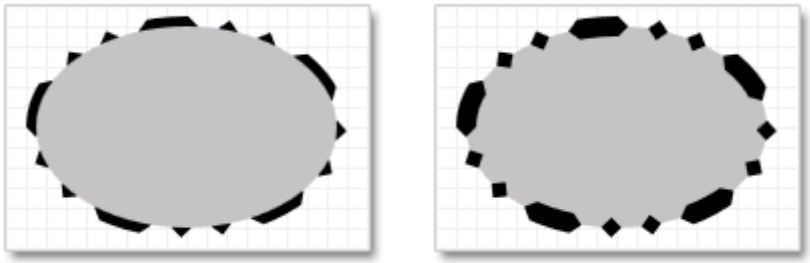
단순한 기하를 그리기”

- 그리기와 채우기
  - 타원의 윤곽선 그리기: DrawEllipse 함수
  - 타원의 내부를 채우기: FillEllipse 함수
  - 예: DrawEllipse 호출 후에 FillEllipse 호출

```
m_pRenderTarget->DrawEllipse(ellipse, m_pBlackBrush, 10.f, m_pStrokeStyle);
m_pRenderTarget->FillEllipse(ellipse, m_pSilverBrush);
```

  - 예: FillEllipse 호출 후에 DrawEllipse 호출

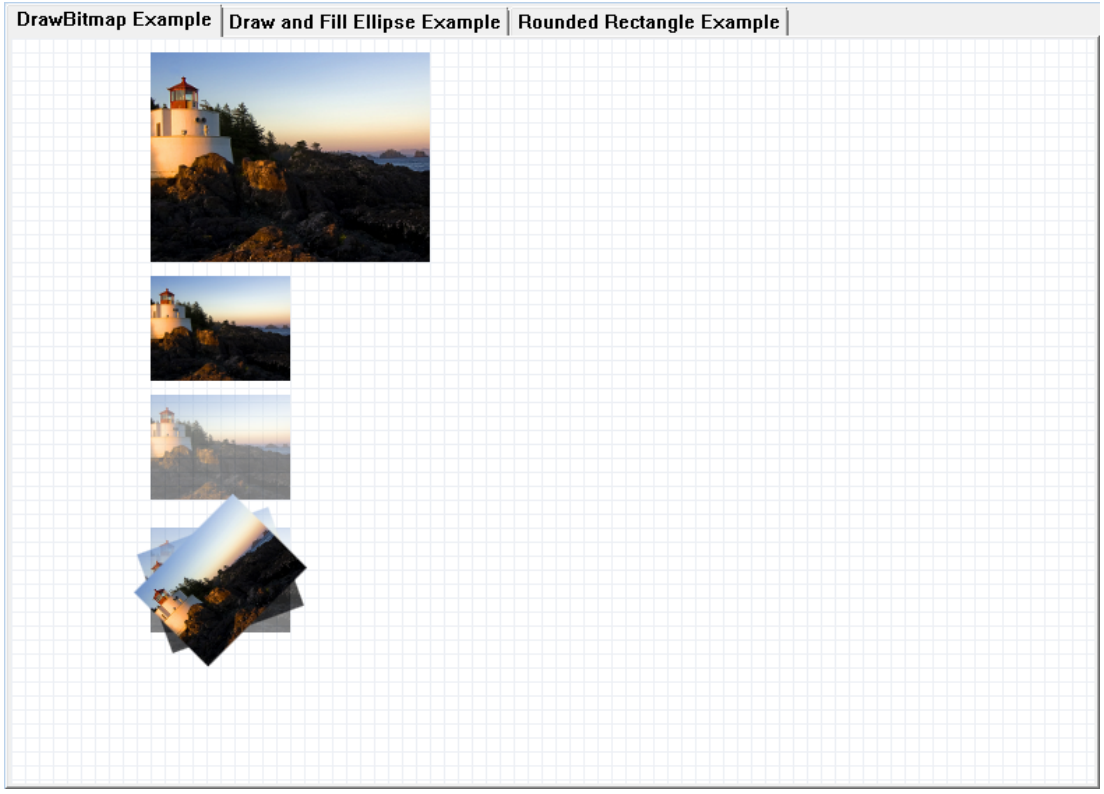
```
m_pRenderTarget->FillEllipse(ellipse, m_pSilverBrush);
m_pRenderTarget->DrawEllipse(ellipse, m_pBlackBrush, 10.f, m_pStrokeStyle);
```



단순한 기하를 그리기'''

21.RenderTargetExamples

- 예제: 단순한 모양을 그리기
  - 비트맵 그리기
  - 단순한 모양 그리기
  - 복잡한 모양 그리기

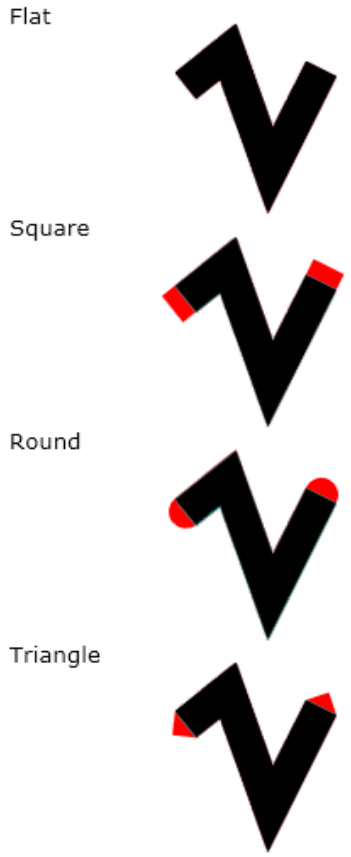


획 스타일 속성값

22.StrokeStyleExample

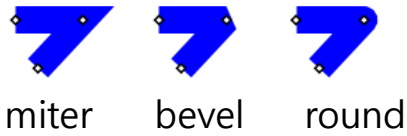
- D2D1_STROKE_STYLE_PROPERTIES 구조체
  - 인자 1,2: startCap/endCap
    - stroke로 그리는 모든 열린 figure의 시작/끝 캡 스타일
  - 인자 3: dashCap: 각 점선 조각들의 양 끝점 캡
  - 인자 4: lineJoin: 각 조각이 조인되는 방식
  - 인자 5: miterLimit (다음 쪽)
  - 인자 6,7: dashStyle, dashOffset (다음 쪽)

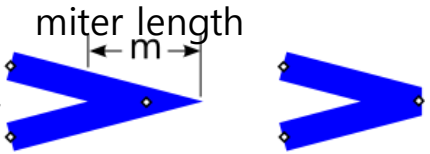
- 인자1,2,3: 캡 스타일
  - 캡 스타일: D2D1_CAP_STYLE_XXX, XXX=
    - FLAT, SQUARE, ROUND, TRIANGLE
      - FLAT: 라인의 끝점에서 끝남.
      - SQUARE: 정사각형 절반이 라인의 끝점에서 더해짐.
      - ROUND: 원모양 반쪽이 라인의 끝점에 더해짐
      - TRIANGLE: 직각 이등변 삼각형이 라인의 끝점에 더해짐

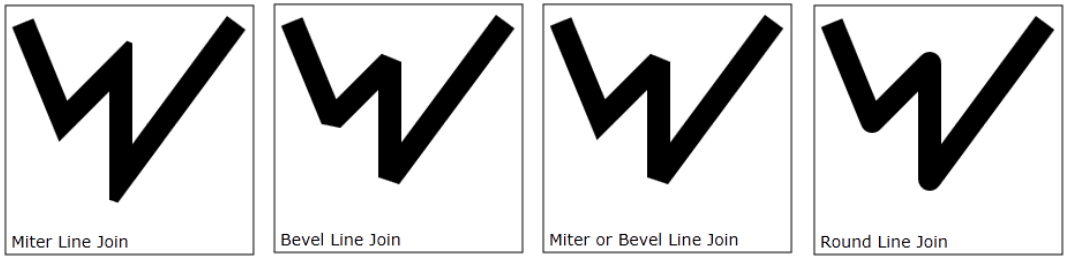


획 스타일 속성값'

- 인자4,5: 라인 조인 스타일
  - 라인 join 스타일: D2D1_LINE_JOIN_XXX, XXX=
    - MITER, BEVEL, ROUND, MITER_OR_BEVEL
    - MITER_OR_BEVEL
      - 기본은 Miter로 그리되, 뾰족이 길게 돌출된 부분들은 Bevel로 그림
      - 판단 기준은 miterLimit로(인자5:FLOAT) 결정함
        - » [miter_length / (stroke_thickness/2)] > miterLimit ?
        - » miterLimit는 뾰족한 돌출 부분의 최소 두께를 명시함

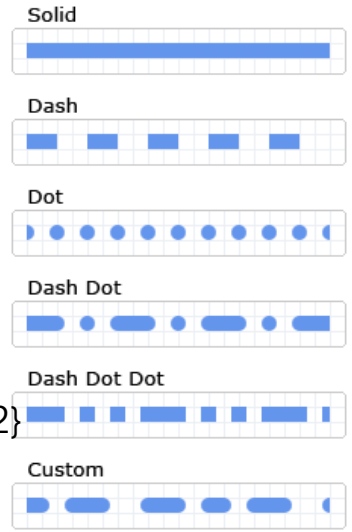


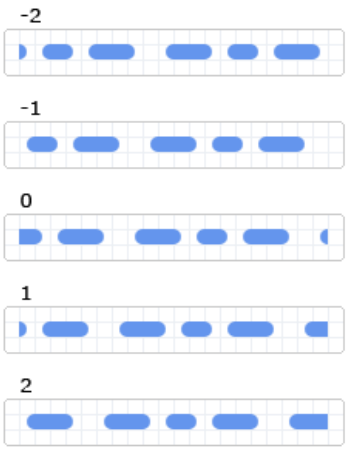




획 스타일 속성값”

- 인자6,7: 점선 스타일
  - 점선 스타일: D2D1_DASH_STYLE_XXX, XXX=
    - SOLID, DASH, DOT, DASH_DOT, DASH_DOT_DOT, CUSTOM
    - 점선배열: stroke_width의 배수 단위로 명시함
      - SOLID: 끊어짐 없음
      - DASH: (dash,gap)의 연속. 즉 dash_array={2,2}
      - DOT: (dot,gap)의 연속. 즉 dash_array={0,2}
      - DASH_DOT: (dash,gap,dot,gap)의 연속. 즉 dash_array={2,2,0,2}
      - DASH_DOT_DOT: (dash,gap,dot,gap,dot,gap)의 연속,
        - » 즉 dash_array={2,2,0,2,0,2}
      - CUSTOM: 점선 패턴을 dash_array로 직접 명시함
  - 점선 오프셋(인자7:FLOAT)
    - 점선 나열에서의 시작 오프셋. 단위는 stroke_width임.
      - 양수/음수는 왼쪽/오른쪽으로 시프트
    - 대부분 0





획 스타일 속성값‴

- 점선 스타일′
  - 예: 점선 스타일을 Custom으로 지정

```
float dashes[] = {1.0f, 2.0f, 2.0f, 3.0f, 2.0f, 2.0f};
m_pD2DFactory->CreateStrokeStyle(
        D2D1::StrokeStyleProperties( D2D1_CAP_STYLE_FLAT, D2D1_CAP_STYLE_FLAT,
            D2D1_CAP_STYLE_ROUND, D2D1_LINE_JOIN_MITER,
            10.0f, D2D1_DASH_STYLE_CUSTOM, 0.0f),
        dashes, ARRAYSIZE(dashes),
        &m_pStrokeStyleCustomOffsetZero );
m_pRenderTarget->DrawLine( D2D1::Point2F(0, 310), D2D1::Point2F(200, 310),
                m_pCornflowerBlueBrush, 10.0f, m_pStrokeStyleCustomOffsetZero );
```

※ ARRAYSIZE(a): 배열 a의 원소의 개수를 리턴하는 매크로 함수

- 효율성을 위한 주의
  - 점선 스타일의 사용은 성능 저하를 유발할 수 있음
    - 알고리즘이 복잡하여, 매우 비싼 연산임
    - 대안으로, 비트맵을 타일 형태로 그려 빠르게 그릴 수 있음