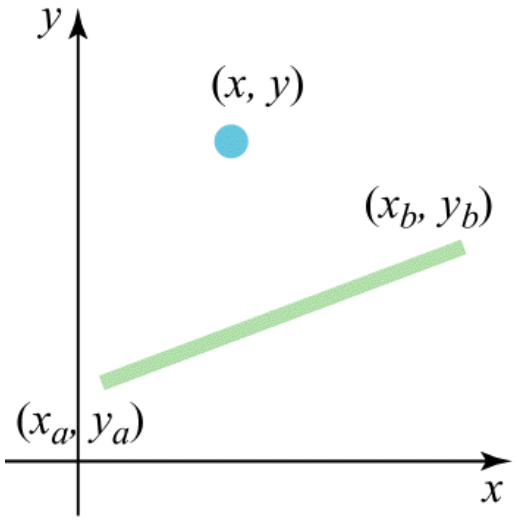
2차원 그래픽스의 기본(1)

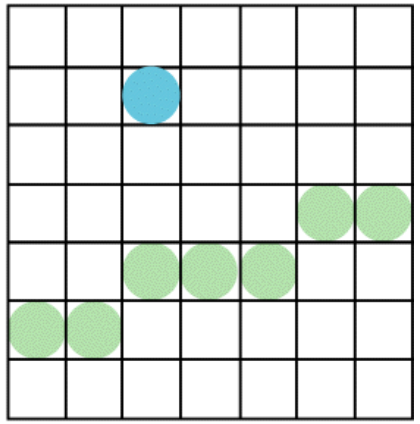
동아대학교 컴퓨터공학과
박영진

점과 선의 정의 및 속성

- 점(Point)
  - 기하적 공간에서 좌표 (x,y)로 정의
  - 속성은 크기(Size), 명암(Intensity) 또는 색상(Color), 모양(Shape) 등



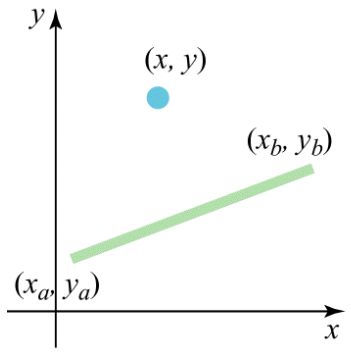
(a) 기하적 공간



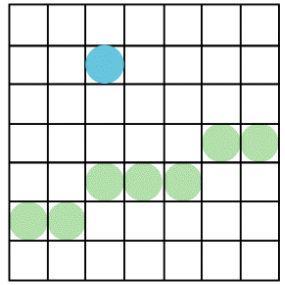
(b) 출력장치의 공간

점과 선의 정의 및 속성

- 선(Line)
  - 시작점 (x_a, y_b)와 끝점 (x_b, y_b)의 절대좌표로 정의
  - 또는 시작점 좌표 (x,y)와 좌표의 증가값 (△x,△y)의 상대좌표로 정의
  - 속성은 선의 유형(Line Type), 굵기(Width), 색상, 선끝 모양(Line Cap) 등



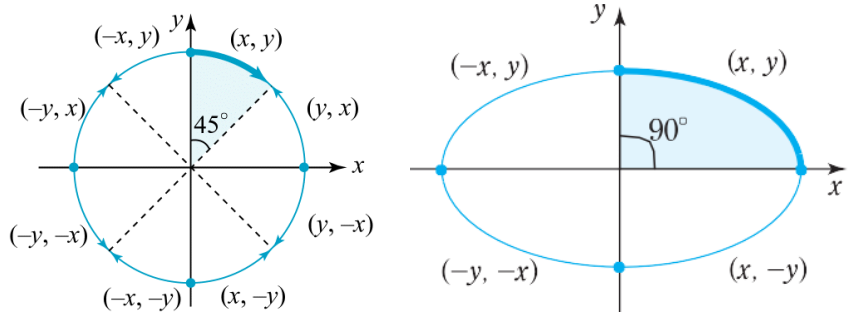
(a) 기하적 공간


(b) 출력장치의 공간

- 다중선(Polyline)
  - 시작점 (x_1, y_1)부터 마지막 n번 째 점 (x_n, y_n)까지의 절대좌표로 정의
  - 또는 시작점 (x_1, y_1)과 각 좌표의 증가값들 (Δx_1, Δy_1)...(Δx_{n-1}, Δy_{n-1})로 정의
  - 선의 속성 이외에 추가로 선 이음(Line Join) 등의 속성

선 이음의 모양

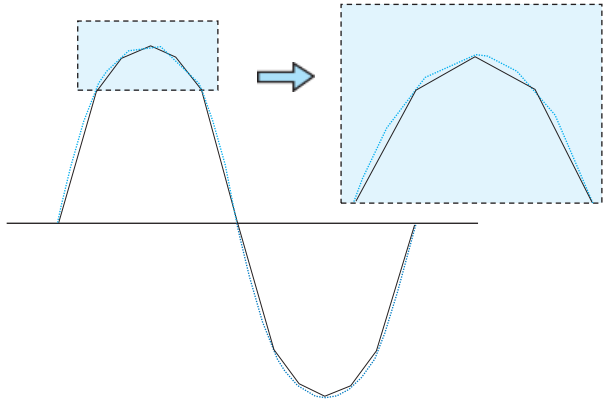
원 그리기



- 극 좌표계(Polar Coordinate)를 이용하는 방법
  - $x^2 + y^2 = r^2$ 일 때 $x = r\cos\theta, y = r\sin\theta$
  - $(x - xc)2 + (y - y_c)^2 = r^2$ 일 때, $x = xc + r\cos\theta, y = y_c + r\sin\theta$
    - 매개변수 θ의 일정 간격으로 원주 상의 점을 구한 후 선분으로 연결
- 타원 그리기
  - $(x - xc)2 / rx^2 + (y - y_c)^2 / ry^2 = 1$일 때 $x = xc + r_x\, cos\,\theta, y = y_c + r_y\, sin\theta$

기타 곡선 그리기

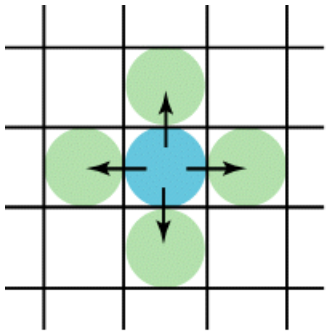
- 함수 $y = f(x)$ 로 표현 가능한 곡선
  - 적당한 간격의 x값에 해당하는 곡선 상의 점을 구한 후 선분으로 연결
  - 삼각(Sine) 함수, 지수(Exponential) 함수, 다항식(Polynomial) 함수, 스플라인(Spline) 함수 등



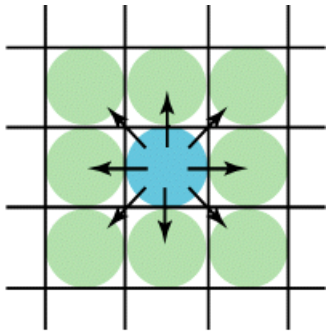
함수 형태의 곡선 그리기

영역 및 다각형 채우기

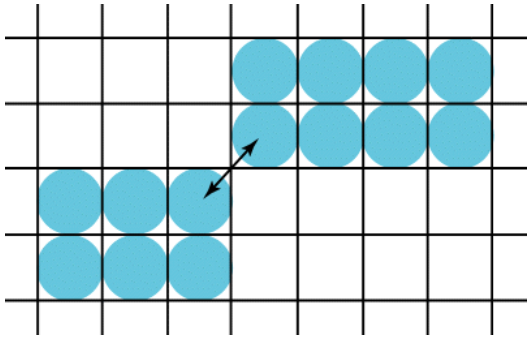
- 이웃한(Adjacent 또는 Connected) 픽셀 간의 연결 방식
  - 4방향연결(4-Connected) 방식
  - 8방향연결(8-Connected) 방식



(a) 4방향 연결



(b) 8방향 연결

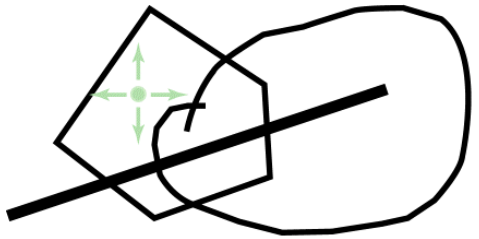


영역 연결 방식의 예

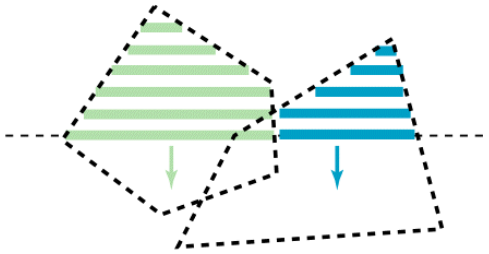
채우기 방식

- 시드채우기 방식
  - 그림이 래스터 버퍼에 그려진 후 이미지에서 영역의 채우기를 실행
  - 영역 내부의 한 픽셀이 시드로 주어지고 이 픽셀에서부터 채워나간다
  - 주로 페인팅 소프트웨어나 대화식 이미지 처리 프로그램에서 사용

- 다각형 주사변환 방식
  - 매 주사선 별로 다각형의 내부 구간을 판단하여 해당 픽셀을 칠한다.
  - 주사선채우기(Scanline Fill)라고도 한다.
  - 주로 벡터방식의 그리기 소프트웨어에서 사용



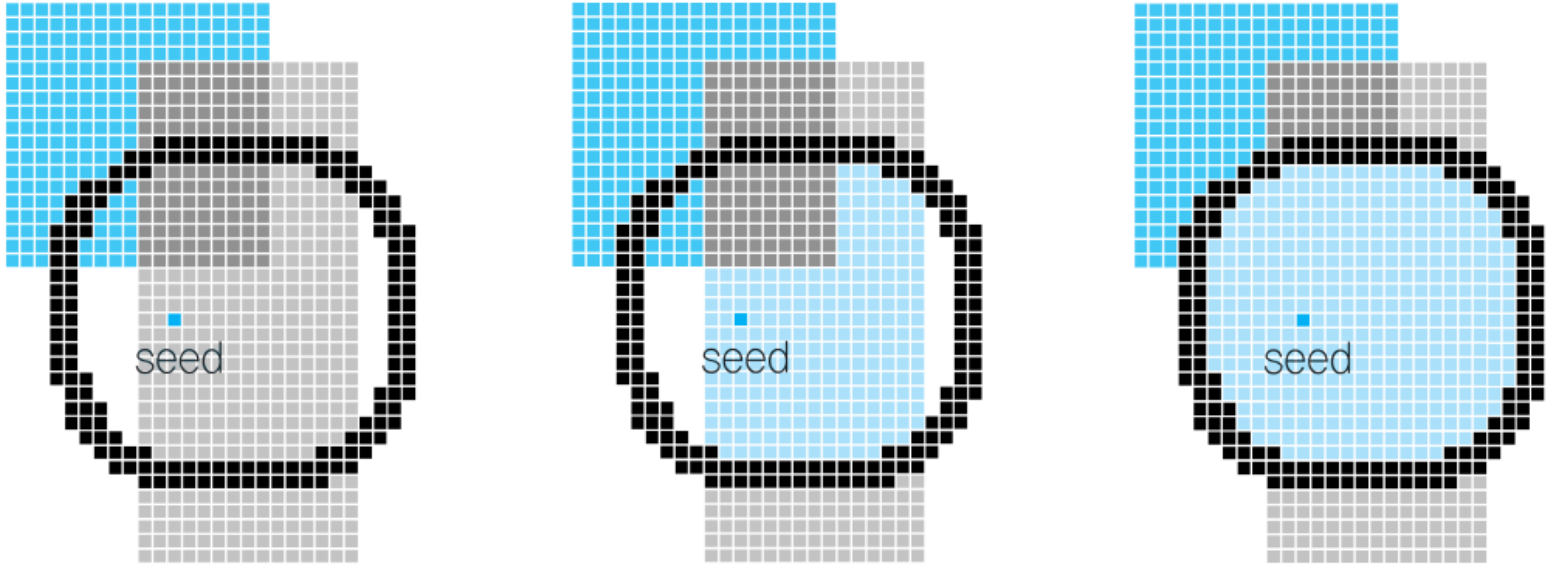
(a) 시드채우기 방식



(b) 다각형 주사변환 방식

시드 채우기(Seed Fill) 방식

- 내부 영역에 대한 판단 및 채우기 방식
  - 내부로 정의된(Interior-defined) 영역 채우기 ➔ 범람채우기(Flood-Fill)
  - 경계로 정의된(Boundary-defined) 영역 채우기 ➔ 경계채우기(Boundary-Fill)

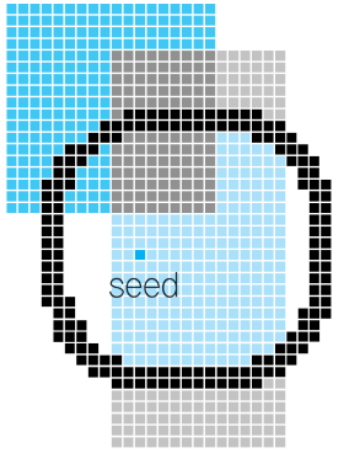


주어진 시드 Flood-Fill Boundary-Fill

알고리즘 비교

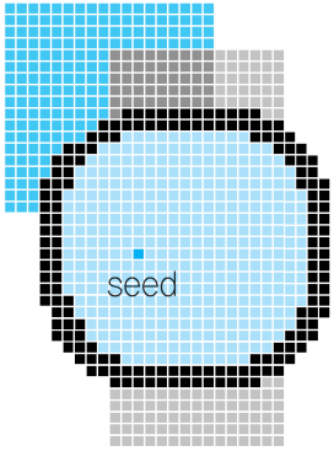
- Flood-Fill

```
void flood_fill(int x, int y) {              // 시드 (x,y) 에서 시작
        if (read_pixel(x, y) == bgColor) {  // 현재 픽셀이 배경색 'bgColor'이면,
                write_pixel(x, y, fillColor);       // 채우기 색 'fillColor'로 칠한다.
                flood_fill(x+1, y);          // 오른쪽으로 반복
                flood_fill(x-1, y);          // 왼쪽으로 반복
                flood_fill(x, y+1);          // 아래로 반복
                flood_fill(x, y-1);          // 위로 반복
        }
}
```



- Boundary-Fill

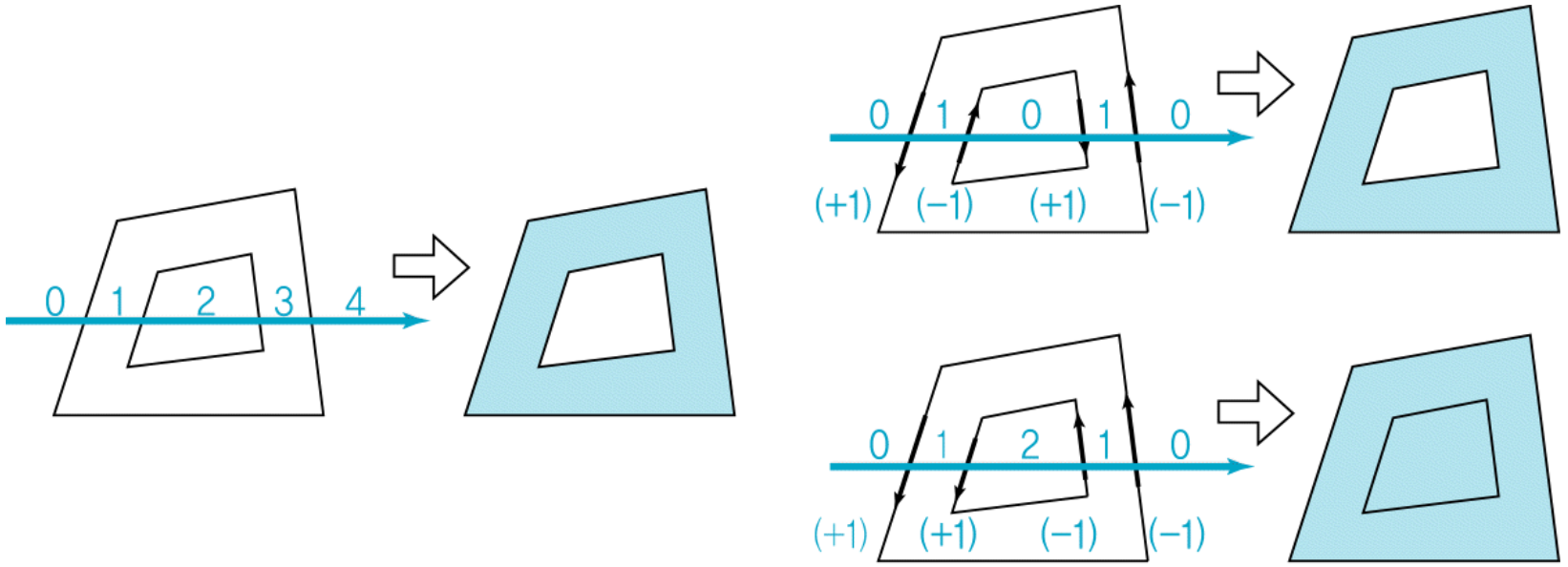
```
void boundary_fill(int x, int y) {      // 시드 (x,y) 에서 시작
        current = read_pixel(x, y);
        if ((current != bdColor) && (current != fillColor)) { // 경계 및 채울 색인지 확인
                write_pixel(x, y, fillColor);       // 내부를 채우기 색 'fillColor'로 칠한다.
                boundary_fill(x+1, y);          // 오른쪽으로 반복
                boundary_fill(x-1, y);          // 왼쪽으로 반복
                boundary_fill(x, y+1);          // 아래로 반복
                boundary_fill(x, y-1);          // 위로 반복
         }
}
```



다각형 내부 판단 규칙

- 홀짝 규칙(Even-Odd Rule)
  - 주사선 별로 에지가 홀수 번째 교차하면 내부가, 짝수 번째 교차하면 외부가 시작된다고 판단

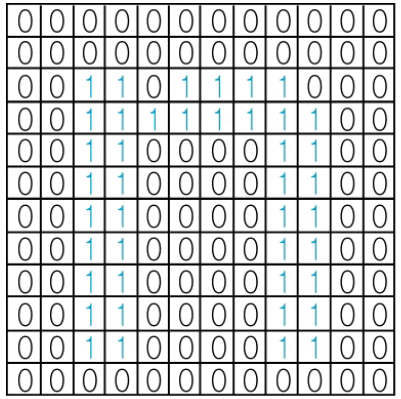
- 접기횟수 규칙(Non-zero Winding Rule)
  - 주사선 별로 아래쪽 에지와 교차하면 접기 횟수를 1 증가, 위쪽 방향의 에지와 교차하면 1 감소
  - 이 때 접기 횟수가 0보다 큰 구간은 다각형의 내부영역으로 판단



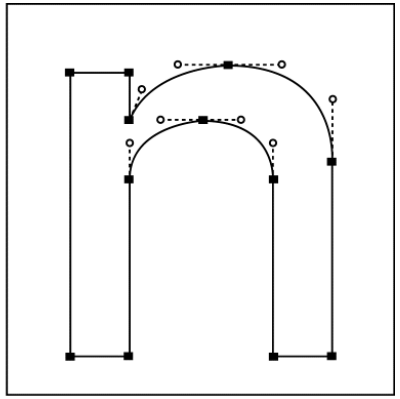
(a) 홀짝 규칙 (b) 접기횟수 규칙

문자의 표현 : 폰트의 종류

- 래스터 폰트(Raster Font 또는 Bitmap Font)
  - 글자 크기에 해당하는 사각형 그리드의 픽셀에 1과 0으로 표현
  - 메모리 내에서 비트맵에 대한 연산으로 처리하므로 출력 속도가 매우 빠르다
  - 제작은 용이하나 확대하면 계단현상(Aliasing)
  - 글자의 확대, 회전, 밀림 등 기하변환은 매우 어려우며 변환시 출력 품질 저하
- 벡터 폰트(Vector Font 또는 Outline Font)
  - 글자의 윤곽선을 여러 부분으로 나누어 직선, 원호, 곡선 등으로 표현하고, 이들에 대한 제어점을 저장
  - MS 윈도우의 TrueType 폰트는 2차 B-스플라인(B-Spline) 곡선을 사용
  - PostScript의 Type1 폰트는 3차 베지어(Bezier) 곡선을 사용



(a) 래스터 폰트


(b) 벡터 폰트

폰트의 종류

- 벡터 폰트의 처리과정

  1) 제어점의 좌표들에 대해 축소하거나 회전하는 등 기하변환을 수행

  2) 윤곽선의 곡선 부분을 선 조각으로 나누어 다각형 형태로 표현

  3) 다각형 주사선 알고리즘을 이용하여 글자의 내부영역 채우기를 수행

- 벡터 폰트의 특징
  - 확대 또는 축소를 하여도 출력 품질이 동일하게 유지
  - 회전 등의 기하변환이 용이, 단, 기하변환을 위한 계산 시간 증가