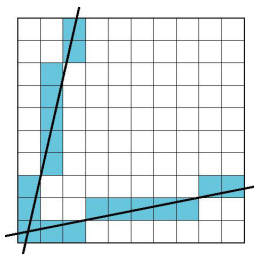
8. Line Rasterization

> 💡 **Line rasterization algorithms**
>
> - **DDA**
> - **Brensenham's algorithm**

Rasterization

- rasterization : 2D 를 pixel 로 바꿔주는 것
- line segment rasterization, polygon rasterization 을 살펴보자



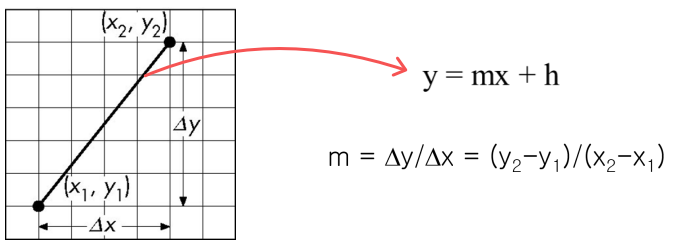
Rasterization of Line Segments

(가정 1) 시작점, 끝점 좌표가 integer 라고 가정

- integer 가 아니면, 반올림 or 내림 등의 방법으로 정수로 만들면 된다.

(가정 2) line 과 가장 근접한 좌표를 찾는 과정까지만 배우고, 해당 좌표를 어떻게 display 할 지는 배우지 않는다.

- (x, y) 가 주어지면, 해당 pixel 을 켜는 `setPixel(x, y)` 라는 함수가 있다고만 배운다.
- 그 함수의 내부는 배우지 않는다. 그 구현 방법은 매우 다양하기 때문



DDA

(이해)

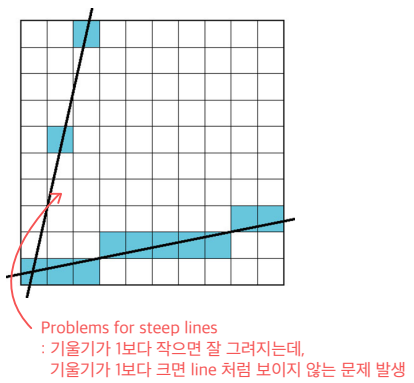
- x 를 시작점부터 하나씩 증가시키면서, y 를 증가시켜야 할지 아닌지를 결정

```
m = (y2-y1)/(x2-x1);  기울기 구하고

for(x=x1, y=y1; x<=x2; x++)
{                 시작점   x 가 x2 가 될 때까지 1 씩 증가
  setPixel(x, round(y));
  y+=m;
}
```

setPixel() 에 들어가는 x, y 는 픽셀 값이므로 정수여야 한다
y 는 기울기를 더한 값이라 정수가 아닐 수 있다
따라서 round() 해줘야 한다

Δx 가 1 이므로
m = Δy/Δx = Δy
즉, y 는 기울기(m)만큼 증가한다



(결론)

- **(1 ≥ |m| ≥ 0 인 경우)** 위 알고리즘 그대로 이용

```
m = (y2-y1)/(x2-x1);

for(x=x1, y=y1; x<=x2; x++)
{
  setPixel(x, round(y));
  y+=m;
}
```

- **(|m| > 1 인 경우)** x 와 y 의 역할을 swap

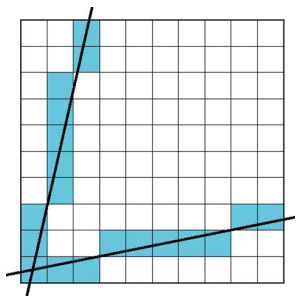
(코딩방법 1) y 가 1 증가하면, x 가 1/m 증가

```
m = (y2-y1)/(x2-x1);
for(x=x1, y=y1; y<=y2; y++)
{
  setPixel(round(x), y);
  x+=1/m;
}
```

Δy 가 1 이므로
m = Δy/Δx = 1/Δx
즉, x 는 1/m 만큼 증가한다

(코딩방법 2) 코드로 x, y swap

```
swap x with y;
m = (y2-y1)/(x2-x1);
for(x=x1, y=y1; x<=x2; x++)
{
  setPixel(x, round(y));
  y+=m;
}
swap x with y;
```



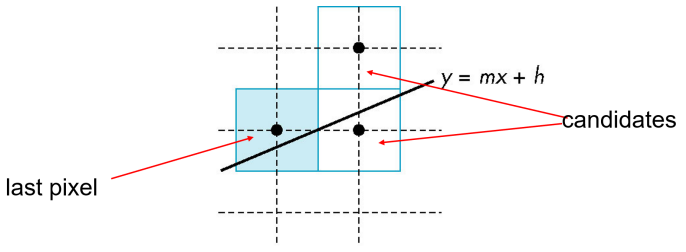
Bresenham's Algorithm

(이해)

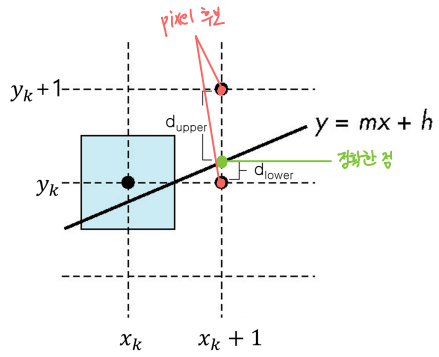
- DDA 코드를 최적화한 방법

> 💡 DDA 는 input, output 모두 integer 좌표계인데, 중간 과정에서 floating point 연산 존재
>
> **floating point 연산을 없애서 최적화하자**

- **(1 ≥ |m| ≥ 0 인 경우)** 만 살펴보고, **(|m| > 1 인 경우)** 는 앞과 동일하게 symmetry 이용해서 처리하면 됨
- 기울기가 1보다 작기 때문에, 현재 픽셀을 찾았으면 **그 다음 픽셀의 후보는 오직 2개이다.**



- 2개의 후보 중에 어떤 것을 선택할 지 알려주는 것이 **Decision Variable p_k**



$p_k = \Delta x\,(d_{lower} - d_{upper})$

d_{lower}, d_{upper} 는 정수가 아니다 (보다 작거나)
Δx를 곱해주면 p_k가 정수가 된다

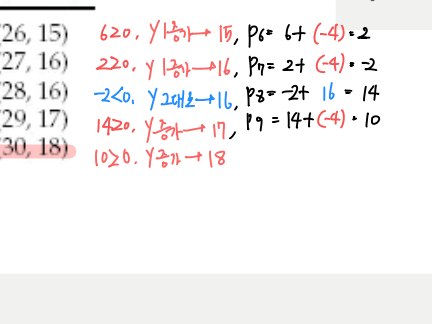
p_k is an integer
$p_k < 0$ use lower pixel
$p_k > 0$ use upper pixel

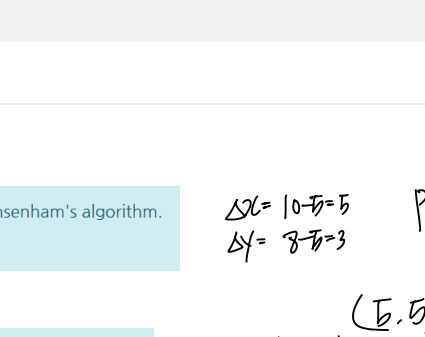
(Point 1) p_k 가 정수
(Point 2) p_{k+1} 은 p_k 로부터 쉽게 구할 수 있음

(결론)

1. 기울기 확인
   - **(1 ≥ |m| ≥ 0 인 경우)** 그대로
   - (**|m| > 1 인 경우)** 앞뒤에서 x, y swap 필요
2. 초기값 계산
   - `setPixel(x₀, y₀)`
   - p_o 계산. **$p_o = 2\Delta y - \Delta x$**
3. Decision Variable **p_k** 계산하며, 그 다음 픽셀 결정. 3 반복
   - **$p_k < 0$** 이면,
     - (x_k+1, y_k) 선택 (즉, y 그대로 유지)
     - **$p_{k+1} = p_k + 2\Delta y$**
   - **$p_k \geq 0$** 이면,
     - (x_k+1, y_k+1) 선택 (즉, y 1 증가)
     - **$p_{k+1} = p_k + 2\Delta y - 2\Delta x$**

(p_o 의 유도 과정과 증명)




Bresenham's Line-Drawing Algorithm for $|m| < 1.0$

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
2. Set the color for frame-buffer position (x_0, y_0); i.e., plot the first point.
3. Calculate the constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as
   $$p_0 = 2\Delta y - \Delta x$$
4. At each x_k along the line, starting at $k = 0$, perform the following test. If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and
   $$p_{k+1} = p_k + 2\Delta y$$
   Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and
   $$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$
5. Perform step 4 $\Delta x - 1$ times.

$y = mx+h$ 식에 x_k 대입

$$y = m(x_k + 1) + b$$

$$d_{lower} = y - y_k = m(x_k + 1) + b - y_k$$

$$d_{upper} = (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b$$

d_{lower}와 d_{upper} 중 어느 것이 더 가까운지 비교해, 빼기

$$d_{lower} - d_{upper} = 2m(x_k + 1) - 2y_k + 2b - 1$$

$m = \frac{\Delta y}{\Delta x}$,

$$p_k = \Delta x(d_{lower} - d_{upper}) = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$ ← 정리

k번째 $p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$

$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$, $x_{k+1} - x_k = 1$

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

최종

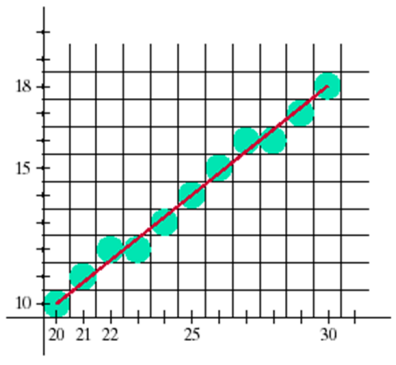
정리 $p_0 = 2\Delta y(x_0+1) + \Delta x(-2y_0 + 2b - 1)$ ← 대입 $(x_0, y_0) \rightarrow y_0 = \frac{\Delta y}{\Delta x}x_0 + b$

$$p_0 = 2\Delta y - \Delta x$$

(예제)

$p_0 = 2\Delta y - \Delta x = 6$

$\Delta x = 10$, $\Delta y = 8$ (20-10, 18-10)



(20, 10)

k	p_k	(x_{k+1}, y_{k+1})	
0	6	(21, 11)	다음, y 증가→11, $p_1 = 6 + (-4) = 2$
1	2	(22, 12)	다음, y 증가→12, $p_2 = 2 + (-4) = -2$
2	−2	(23, 12)	다음, y 그대로→12, $p_3 = -2 + 16 = 14$
3	14	(24, 13)	다음, y 증가→13, $p_4 = 14 + (-4) = 10$
4	10	(25, 14)	다음, y 증가→14, $p_5 = 10 + (-4) = 6$

k	p_k	(x_{k+1}, y_{k+1})	
5	6	(26, 15)	다음, y 증가→15, $p_6 = 6 + (-4) = 2$
6	2	(27, 16)	다음, y 증가→16, $p_7 = 2 + (-4) = -2$
7	−2	(28, 16)	다음, y 그대로→16, $p_8 = -2 + 16 = 14$
8	14	(29, 17)	다음, y 증가→17, $p_9 = 14 + (-4) = 10$
9	10	(30, 18)	10≥0, y 증가→18

$p_k < 0$, $p_{k+1} = p_k$ **+ 2Δy**

$p_k \geq 0$, $p_{k+1} = p_k$ **+ 2Δy - 2Δx**

⇒ **2Δy** (16) 계산해 놓고, **음수**일 때 p_k 에 더하기

⇒ **2Δy - 2Δx** (-4) 계산해 놓고, **양수**일 때 p_k 에 더하기

제대로 했는지 확인하는 방법 → end point (30, 18) 에서 끝나는 지 확인

DDA 결과와 **Bresenham's Algorithm** 결과는 **같음**

Bresenham's Algorithm 은, DDA 과정 중에서 floating point 계산을 없앤 것일 뿐

Quiz) Line Rasterization

Given a line segment $\overline{P_1P_2}$ connecting two points $P_1(5,5)$ and $P_2(10,8)$, rasterize it using Brensenham's algorithm. The first rasterized point (x_0, y_0) is the same as P_1.

$\Delta x = 10 - 5 = 5$, $\Delta y = 8 - 5 = 3$

$p_0 = 2\Delta y - \Delta x = 6 - 5 = 1$

$2\Delta y = 6$

$2\Delta y - 2\Delta x = -4$

Find (x_1, y_1).

하나를 선택하세요.
- a. (6, 5)
- b. (6, 7)
- c. (6, 8)
- d. (6, 6) ✔

Find (x_2, y_2).

하나를 선택하세요.
- a. (6, 7)
- b. (7, 8)
- c. (7, 7)
- d. (7, 6) ✔

Find (x_3, y_3).

하나를 선택하세요.
- a. (8, 7) ✔
- b. (8, 9)
- c. (8, 8)
- d. (7, 7)

Find (x_4, y_4).

하나를 선택하세요.
- a. (9, 7) ✔
- b. (8, 8)
- c. (9, 9)
- d. (9, 8)

(5, 5)

k	p_k	(x_{k+1}, y_{k+1})
0	1	(6, 6)
1	−3	(7, 6)
2	3	(8, 7)
3	−1	(9, 7)
4	5	(10, 8)