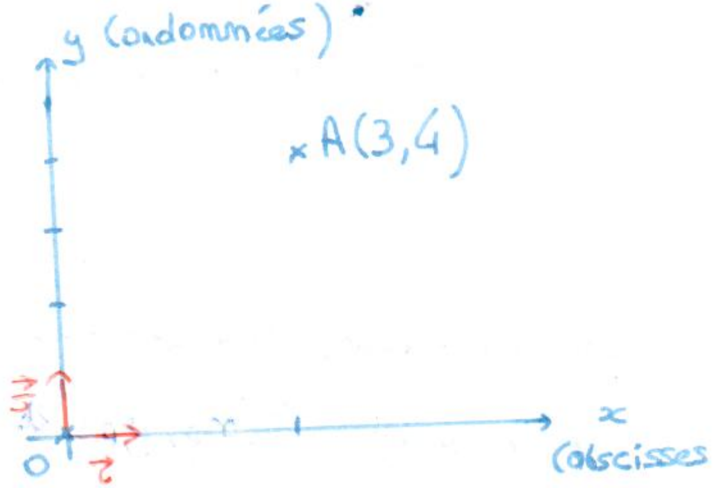
Tracé de droites discrètes

0 Rappel : équations de droites

On se place dans le plan (dim. 2) muni d'un repère orthonormé $(O, \vec{i}, \vec{j})$.



Un point A du plan est repéré par ses coordonnées x_A et y_A avec :

$$\vec{OA} = x_A \vec{i} + y_A \vec{j}$$

Notation : $A(x_A, y_A)$

Une droite est un ensemble de points tous alignés. Dans le plan, cela correspond à l'ensemble des points de coordonnées (x,y) satisfaisant une équation de la forme

$$ax + by + c = 0$$

(équation cartésienne)

$$y = mx + p$$

(équation affine)

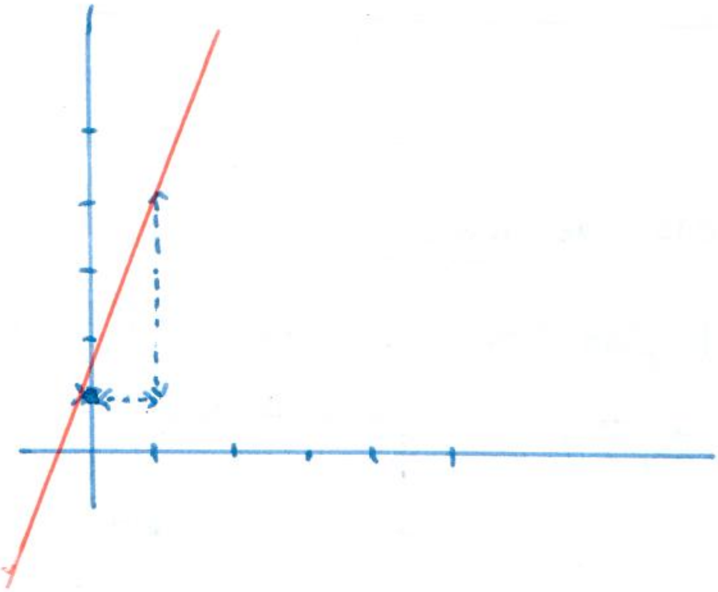
$$m = -\frac{a}{b}$$

$$p = -\frac{c}{b}$$

m : coefficient directeur

p : ordonnée à l'origine

Ex : Droite d'équation $y = 3x+1$



Passe par $(0,1)$
Pente égale à 3

On considère une droite $(\mathcal{D}) : y = mx + p$ passant par deux points $P(x_P, y_P)$ et $Q(x_Q, y_Q)$. Alors :

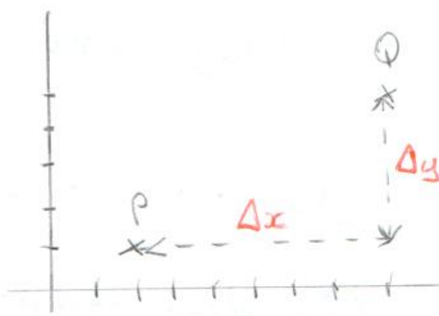
$$m = \frac{y_Q - y_P}{x_Q - x_P} := \frac{\Delta y}{\Delta x}$$

(variation en y sur variation en x)

Pour trouver p, on injecte alors les coordonnées (x_P, y_P) dans l'équation :

$$y_P = \frac{y_Q - y_P}{x_Q - x_P} x_P + p$$

d'où $$p = y_P - \frac{y_Q - y_P}{x_Q - x_P} x_P$$



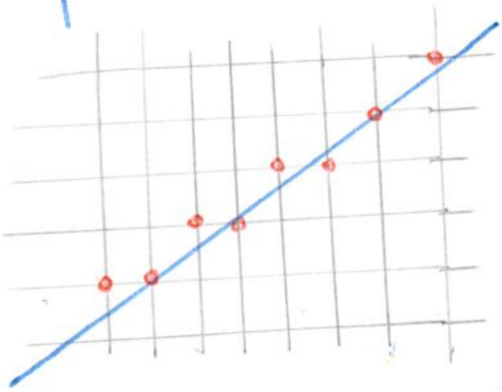
On a alors pour équation

$$y = \frac{y_Q - y_P}{x_Q - x_P} x + y_P - \frac{y_Q - y_P}{x_Q - x_P} x_P$$

$$= \boxed{y_P + (x - x_P) \times \frac{y_Q - y_P}{x_Q - y_P}}$$

I Droite discrète et algorithme naïf

Objectif: On cherche à tracer graphiquement une droite sur l'ordinateur en dessinant l'ensemble des pixels les plus proches de la droite réelle

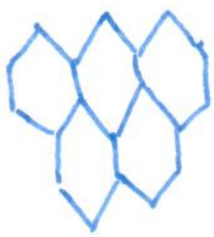


On se déplace selon un chemin 8-connexe:

→ ← ↑ ↓ ↗ ↙ ↘ ↖

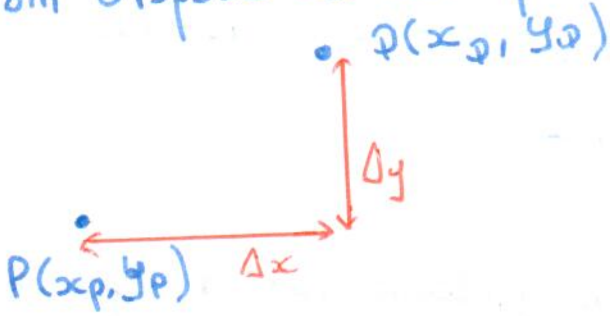
Remarque: Certains algorithmes cherchent des droites discrètes en utilisant de la 6-connexité :

↗ ↘ ↙ ↖ ↓ ↑



Ce type de pavage garantit la distance la plus faible possible entre deux voisins

Si on dispose de deux points $P(x_P, y_P)$ et $Q(x_Q, y_Q)$



Valeur réelle au point de abscisse x:

$y = y_P + (x - x_P) \times \frac{y_Q - y}{x_Q - x}$

```
void droite(int xp, yp, xQ, yQ){
    int x,y;
    for (x = xp; x <= xQ; x++){
        y = xp + ((x-xp) * (yQ-yP))/(xQ - xp);
    }   afficher(x,y);
```

Problème : Le calcul produit un résultat flottant ; en réalisant la division entière, on va choisir l'entier strictement inférieur (partie entière).

Exemple : Pour tracer la droite passant par $P(\underset{x_P}{\underset{\shortparallel}{10}}, \underset{y_P}{\underset{\shortparallel}{12}})$

et $Q(\underset{x_Q}{\underset{\shortparallel}{12}}, \underset{y_Q}{\underset{\shortparallel}{36}})$

x_P	10		
y_P	12		
x_Q	12		
y_Q	36		
x	10	11	12
y	12	24	36

$$12 + (10-10) \times (36-12) / (12-10)$$

Quand $x = 11$,

$$y = 12 + (11-10) \times (36-12) / (12-10)$$
$$= 12 + 24/2$$
$$= 24$$

Dans l'algo, à partir du calcul flottant, on a 3 possibilités :

- partie entière $\lfloor y_R \rfloor$
- entier supérieur $\lceil y_R \rceil$
- entier arrondi (le plus proche) $[y_R]$

Pour obtenir l'entier arrondi, on calcule

$$y = \underset{\text{cast entier}}{(int)} \quad (y_P + (x - x_P) * (y_Q - y_P) / (float)(x_Q - x_P) + 0.5)$$

car pour un nombre réel x, $\lceil x \rfloor = \lfloor x + 0.5 \rfloor$

↳ Arrondi au supérieur dans les cas limites !!

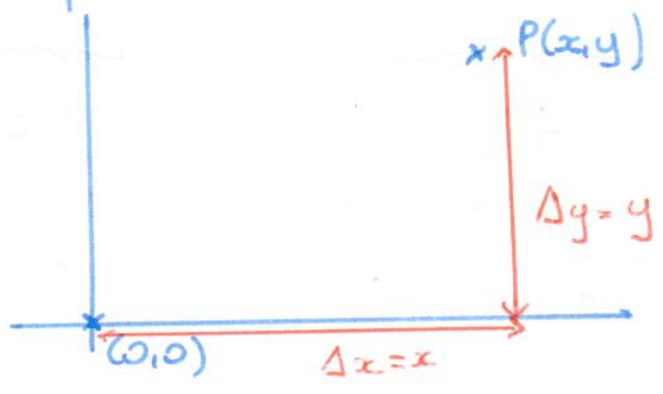
Algo 1

```
void droiteTrivial (xp, yp, xQ, yQ){
    int x,y;
    for (x= xp; x<= xQ; x++){
        y= (int) (yp+(x-xp)*(yQ-yp)/((float)(xQ-xp) +0.5);
        afficher (x,y);
    }
}
```

<u>Remarque</u>: Ce n'est toutefois pas très propre de passer par des calculs flottants pour récupérer des entiers à partir d'entiers.

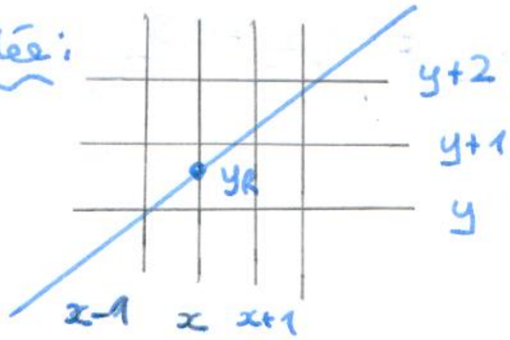
2 Algorithme de Bresenham: la référence 1963

* Par une simple translation, on peut supposer que les droites partent de (0,0). Du coup, on peut simplement utiliser dx et dy en paramètres



* On va se limiter aux droites contenues dans le premier octant (0° ~> 45°). On pourra alors tracer les autres droites par une simple transformation (rotation, symétrie)

Idée :



y_R = valeur réelle de l'ordonnée au point x

On va comparer la distance entre
- y et y_R
- $y+1$ et y_R

et choisir la plus petite.

$$\boxed{\text{Distance entre } y \text{ et } y_R : \ |y - y_R|}$$

Si $|y - y_R| > |y+1-y_R| \Rightarrow y \leftarrow y+1$ $(y{+}{+})$

Sinon, y ne bouge pas

$$y_R = x \times \frac{dy}{dx}$$

~> On regarde si

$$\underbrace{\left| y - x\frac{dy}{dx} \right|}_{\substack{A \\ \text{distance entre} \\ y \text{ et } y_R}} > \underbrace{\left| y+1 - x\frac{dy}{dx} \right|}_{\substack{B \\ \text{distance entre} \\ y+1 \text{ et } y_R}} \Rightarrow y{+}{+}$$

B = A + 1

On étudie, ce qui se passe selon les signes de A et B :

A	B	
≥ 0	≥ 0	y constant ($\lvert A\rvert < \lvert A+1\rvert$)
≥ 0	< 0	$\emptyset$ (Impossible)
< 0	≥ 0	$\boxed{??}$
< 0	< 0	y++

Cas restant : $A < 0$ et $B \geq 0$ ($\lvert A\rvert = -A$, $\lvert B\rvert = B$)

on regarde si $-A > B \Rightarrow 0 > A+B$ donc

on regarde le signe de $A+B$

En fait, cela fonctionne dans tous les cas :

Si : $A+B \geq 0$, y constant

Si : $A+B < 0$, y++

$$A+B = y - x\frac{dy}{dx} + y+1 - x\frac{dy}{dx} = 2y+1-2x\frac{dy}{dx}$$

$$A+B < 0 \iff 2y+1-2x\frac{dy}{dx} < 0$$

$\times dx$

$$\iff 2ydx + dx - 2xdy < 0$$

On va donc en résumé étudier le signe de

$$\boxed{2y\,dx + dx - 2x\,dy := \text{delta}(x,y)}$$

Si: $\text{delta}(x,y) < 0$, $\quad$ y++

Sinon, $\quad$ y constant

<u>Remarque:</u> les valeurs de dx et dy sont constantes tout au long de l'exécution.

<u>P°</u> Comment évolue delta (x,y) en fonction de x et y?

2 possibilités :

1. x++ , y constant (pas horizontal →)

$$\text{delta}(x+1, y) = 2y\,dx + dx - 2(x+1)\,dy$$
$$= \text{delta}(x,y) - 2dy$$

Incrément horizontal

2. x++ , y++ (pas oblique ↗)

$$\text{delta}(x+1, y+1) = 2(y+1)\,dx + dx - 2(x+1)\,dy$$
$$= \text{delta}(x,y) + 2dx - 2dy$$

Incrément oblique

```
void droite_bR(int dx, int dy){
    int delta, incO, incH;
    incH = -dy - dy;
    delta = dx + incH;        valeur de delta(1,0)
    incO = delta + dx;
    for (x=0, y=0; x<= dx; x++){
        afficher (x,y)
        if (delta <0){
            y++;
            delta += incO;
        }
        else{ delta += incH;
        }
    }
}
```

3 Améliorations de Bresemham

Idée: Essayer d'incrémenter x de 2 au lieu de 1, pour faire deux fois moins de tours de boucles

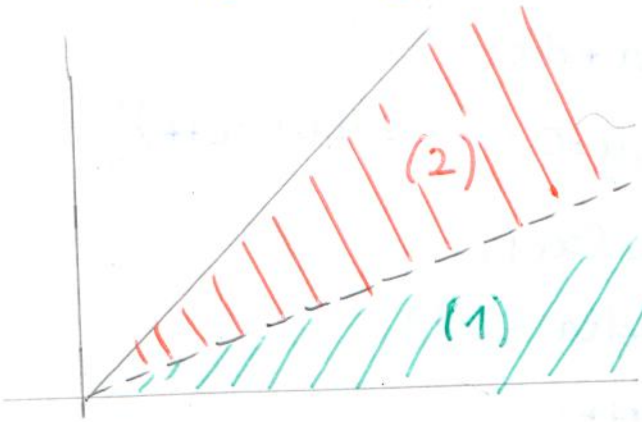
Algorithme " Pas de deux ", Rokme - Wyvill - Wu 1993

on a alors 3 possibilités pour le nouveau y.

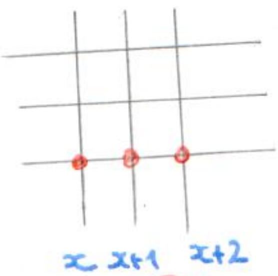
Idée de l'article : Séparer l'octant en deux parties

(1) $0 < dy < dx < 2dy$

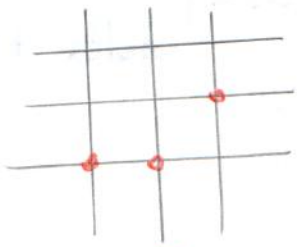
(2) $0 < dy < 2dy < dx$



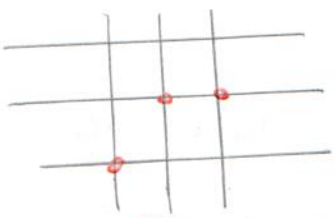
Lorsque x s'incrémente de 2, il y a 4 patterns possibles pour le tracé de droite discrète :



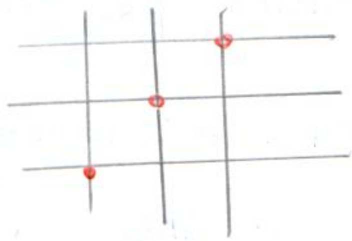
P1



P2



P3



P4

Remarques : • P1 n'est possible que dans la partie basse de l'octant (cas ①)

• P4 n'est possible que dans la partie haute de l'octant (cas ②)

• P2 et P3 sont possibles dans les deux cas.

Posons incr = 2 * dy - dx

→ on commence par tester le signe de incr

<u>Si</u> incr < 0 : partie basse de l'octant

incH = 4dy (pas de 2, delta (x+2, y) = delta (x, y) + 4dy

delta = incH - dx;

while (x < dx + 1){

<u>si</u> delta < 0

Dessiner Pattern 1

delta += incH

<u>Sinon</u> on revient un pas en arrière

<u>si</u> delta - 2dy < 0

Pattern 2

<u>Sinon</u> Pattern 3

y++

delta += 2 * incr;

x += 2;

}

<u>Sinon</u> incr >= 0 : partie haute de l'octant

A ADAPTER

<u>Remarque</u>: Dans les deux cas, si on doit différencier entre pas horizontal puis oblique ou oblique puis horizontal, on doit revenir au cran précédent (delta (x-1, y)) et faire un test sur le signe de ce delta.

⚠ Ici, on prend

$$\text{delta}(x,y) = 2x\,dy - 2y\,dx - dx$$

Si delta ≥ 0 => y++

$$\text{delta}(x+2,y) = \text{delta}(x,y) + 4dy$$

Pas horizontal : $+ 4 * dy$; Pas oblique : $-2dx + 4dy$

$$\text{delta}(x+1,y) = \text{delta}(x+2,y) - 2dy$$

- <u>Extensions</u>

Pas de deux	Rokne, Wyvill, Wu	'93
Pas de trois	Graham, Iyengar	'94
Pas de quatre	Gill	'94
Pas auto-adaptatif	Boyer-Bourdin	'2000.

→ Calculer la pente et décider quel pas utiliser en fonction

Pas idéal = $\dfrac{dx - dy}{dy}$

≃ 4x plus rapide

<u>Remarque</u> : • En mesurant le nombre de tests supplémentaires pour le pas de deux, on se rend compte que ce n'est pas exactement 2x plus rapide que Bresenham.

• Pour le pas de trois, on garde la même séparation de l'octant

Partie basse : 3 patterns | Partie haute : 3 patterns

4- Angel & Morrison '91

On reprend $y = x \frac{dy}{dx}$

Idée : Calculer le pgcd de dx et dy

$d := pgcd(dx, dy)$

On va alors calculer la droite discrète passant par $(0,0)$ et $\left(\frac{dx}{p}, \frac{dy}{p}\right)$, et répéter le tracé de cette droite p fois

Exemple : Pour tracer la droite avec $\begin{cases} dx = 6 \\ dy = 22 \end{cases}$, il suffit de répéter deux fois la droite $\begin{cases} dx = 3 \\ dy = 11 \end{cases}$

Remarque : Angel et Morrison indiquent dans leur article en 1991 que leur algo est 3 fois plus rapide que Bresenham puisque (?) le pgcd moyen sur de grands échantillons de nombres est 4.

Problème : 1 - Calculer le pgcd

2 - 60% des couples de nombres ont un pgcd égal à 1 ! Dans ce cas, on ne gagne rien...

Finalement, si $\begin{cases} t = \text{temps Bresenham} \\ t' = \text{temps avec PGCD} \end{cases}$

$$t' = \frac{60t}{100} + \frac{40}{100}\frac{t}{10} = \frac{64t}{100}$$