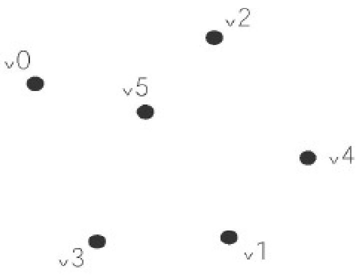
Algorithmes en matériel

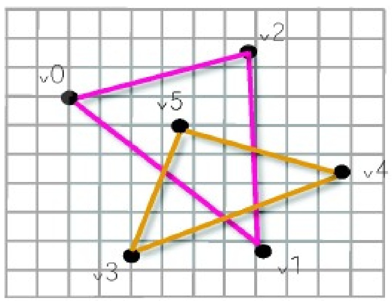
Création des fragments

Les fragments représentent un état graphique intermédiaire entre la sortie du pipeline graphique et les pixels à l'écran :

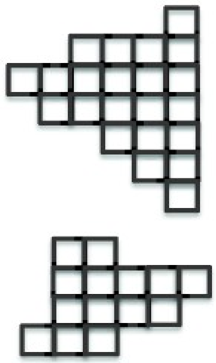
- Un fragment est un élément de géométrie de la taille du pixel.
- C'est le tramage des primitives crée des fragments



sommets



primitive



fragments

ÉCOLE POLYTECHNIQUE MONTRÉAL

Création des fragments

- Procède-t-on point par point pour créer des fragments pour des primitives simples (segment, arc de cercle, polygone, ...) ?
- *Non!* Il existe des algorithmes de conversion spécialisés.

  Par exemple:
  - Algorithme incrémental initial pour un segment de droite
  - Algorithme de Bresenham* (point milieu) pour un segment de droite ou un cercle
- Ces algorithmes créent les fragments pour les primitives 1D.
- Les fragments peuvent ensuite être interpolés, par tramage, sur toute une surface 2D.

* Jack Elton Bresenham, Algorithm for Computer Control of a Digital Plotter, IBM Systems Journal, 4(1):25-30, 1965.

Affichage incrémental initial

- Pour la droite $y=m\bullet x+b$, si l'on fixe Δx, on peut calculer Δy, ou vice versa:

$$\Delta y = m \bullet \Delta x$$

$$\Delta x = \Delta y \,/\, m$$

- Si $0 \le m \le 1$, on peut tracer un segment de droite en fixant $\Delta x = 1$ et en calculant $\Delta y = m \bullet \Delta x = m$

Affichage incrémental initial

- On obtient un algorithme simple:

  x_0 = x1

  y_0 = y1

  tant que x_i < x2

  $x_{i+1} = x_i + 1$

  $y_{i+1} = y_i + m$

- On peut généraliser pour d'autres valeurs de m
- y et m doivent être des valeurs réelles
- y doit être arrondi

Algorithme de Bresenham

- Conversion de segments pour les écrans à balayage linéaire
- N'utilise que des opérations sur les entiers
- Implanté habituellement en matériel
- On incrémente d'une unité la valeur de

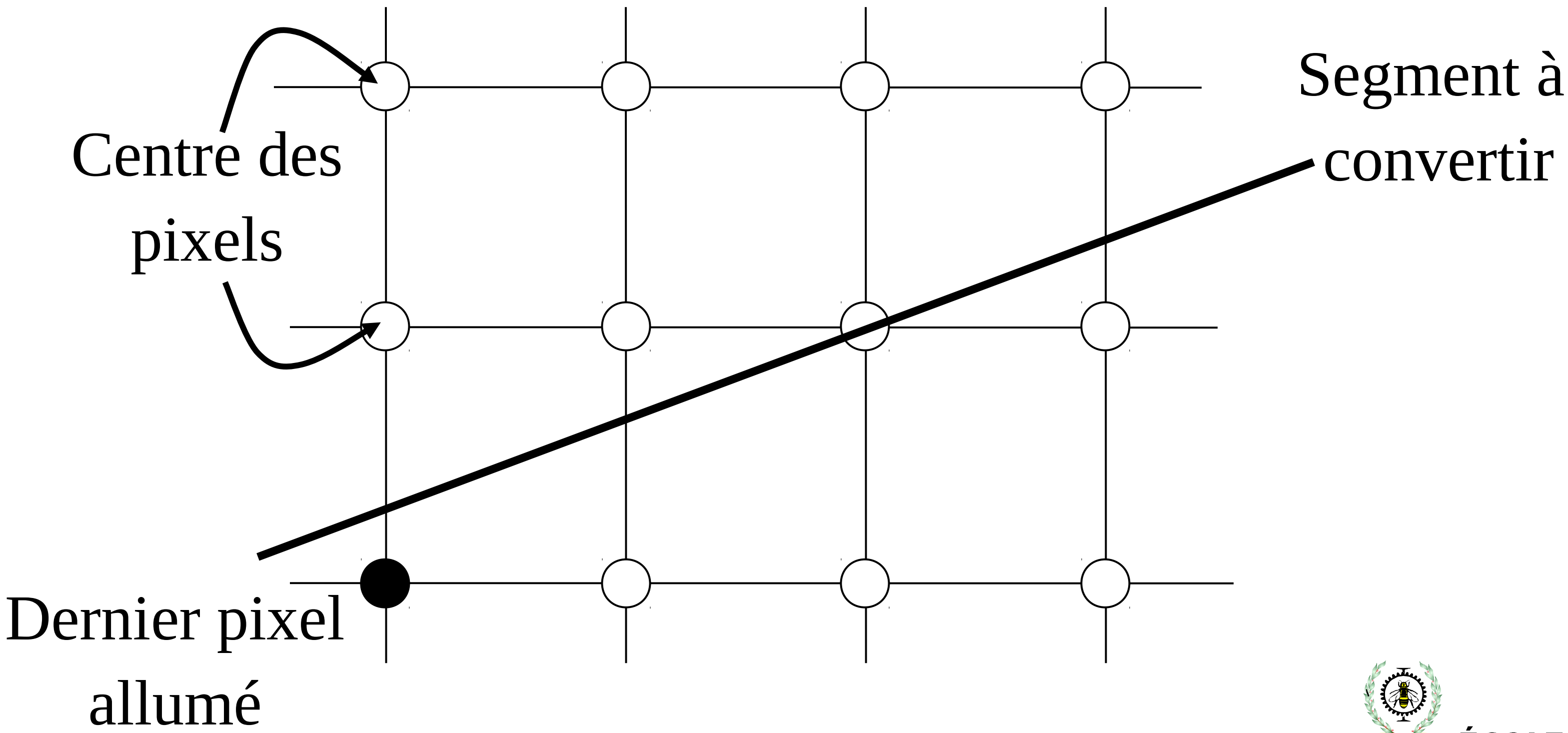
  x_i à $x_i + 1$

et il s'agit de déterminer s'il faut allumer le pixel

$(x_i + 1, y_i+1)$ ou $(x_i + 1, y_i)$

Algorithme de Bresenham

- Valide dans le premier octant et avec pente positive m<1
- Fonctionne pour des pentes entre 0 et 1, avec dx > 0 et dy > 0
- De façon générale, on nomme ce type d’algorithmes ***point milieu.***

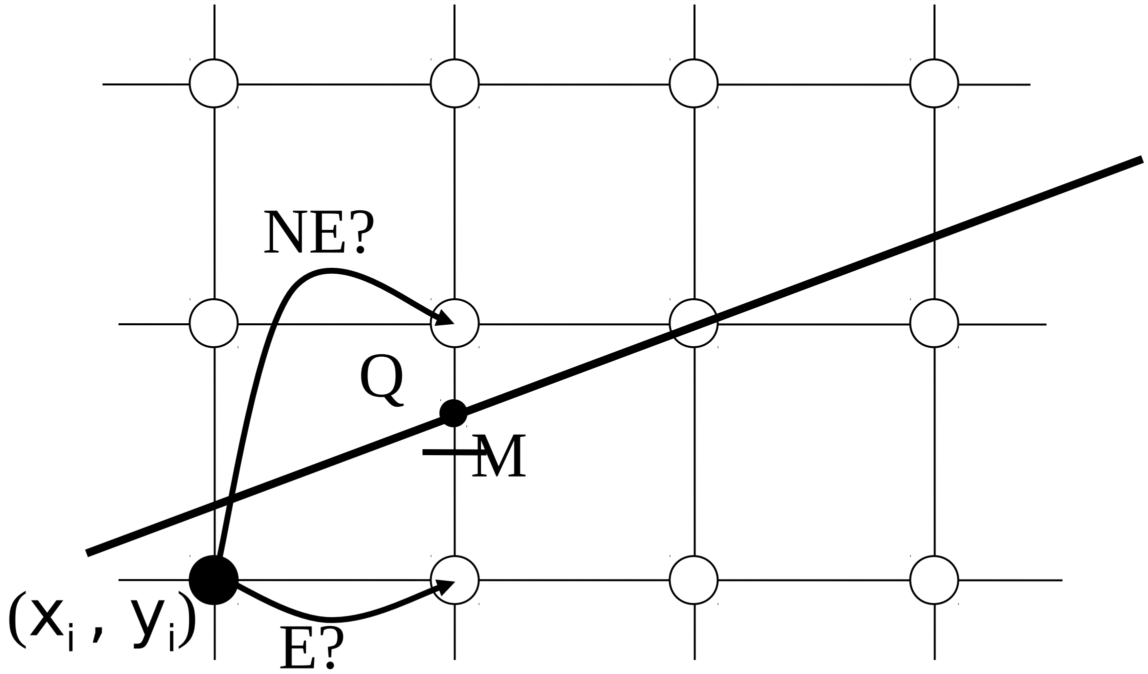


ÉCOLE POLYTECHNIQUE MONTRÉAL

Algorithme de Bresenham

À partir de (x_i, y_i) on doit choisir d'aller vers:

l'EST	$(x_i + 1, y_i)$
le NORD-EST	$(x_i + 1, y_i + 1)$

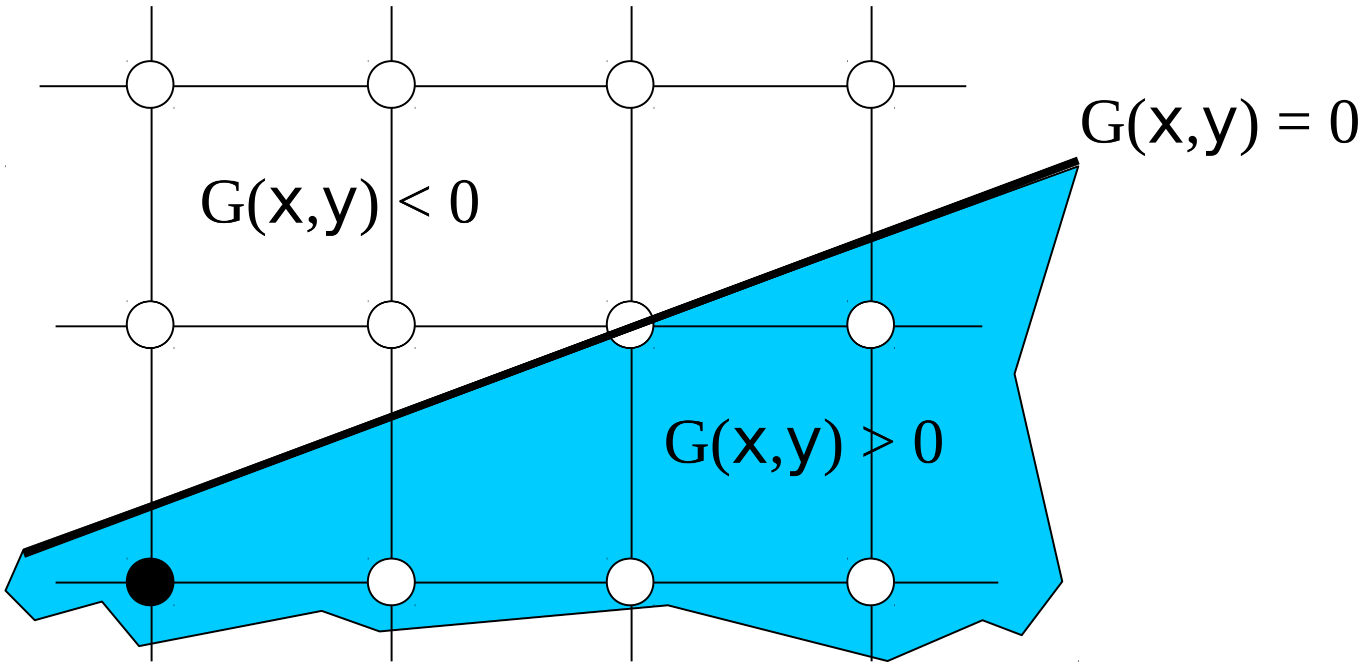


Solution: comparer l'intersection Q avec le point milieu M

Algorithme de Bresenham

Équation de la droite : y = (dy / dx) x + b

On utilise sa forme implicite : G(x, y) = dy · x – dx · y + dx · b



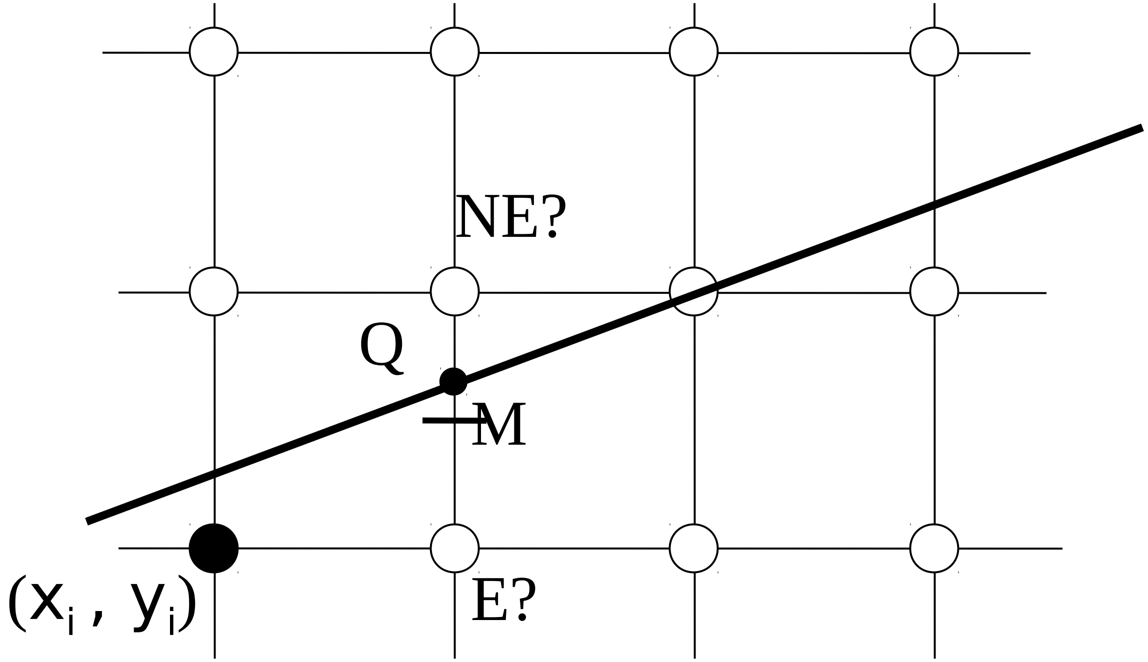
Les propriétés sont conservées si on prend:

F(x,y) = 2G(x,y) = 2dy · x – 2dx · y + 2dx · b

Algorithme de Bresenham

On calcule la valeur de F(x,y) au point M, noté d_i:

$$d_i = F(x_i+1, y_i+½) = 2\,dy\,(x_i + 1) - 2\,dx\,(y_i + ½) + 2dx \cdot b$$



$d_i > 0$: M est sous la droite, Q est au-dessus de M, on va au NE

$d_i \leq 0$: M est au-dessus de la droite, Q est sous M, on va à l'E

Algorithme de Bresenham

Que vaut d_0 ?

$$F(x_0+1, y_0+½) = 2\,dy\,(x_0+1) – 2\,dx\,(y_0+½) + 2\,dx·b$$
$$= 2\,dy·x_0 – 2\,dx·y_0 + 2\,dx·b + 2\,dy - dx$$
$$= F(x_0, y_0) + 2\,dy – dx$$

Comme (x_0, y_0) fait partie de la droite, F(x_0, y_0) = 0.

$d_0 = F(x_0+1, y_0+½) = 2\,dy – dx$

Algorithme de Bresenham

Que vaut d_{i+1} ?

- Si $d_i \leq 0$ alors on est allé à l'E :

  $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$

  $d_{i+1} = F(x_i + 2,\ y_i + ½) = d_i + 2\ dy$

- Sinon, on est allé au NE :

  $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i + 1)$

  $d_{i+1} = F(x_i + 2,\ y_i + 1.5) = d_i + 2\ dy – 2\ dx$

Algorithme de Bresenham

```
FONCTION Bresenham( x1, y1, x2, y2 )
   dx = x2 – x1
   dy = y2 – y1
   d  = 2*dy – dx
   AllumePixel( x1, y1 )
   TANT QUE  x1 < x2
    SI d <= 0 ALORS    // EST
        d  = d + 2*dy
    SINON              // NORD-EST
        d  = d + 2*(dy-dx)
        y1 = y1 + 1
    FIN SI
    x1 = x1 + 1
    AllumePixel( x1, y1 )
   FIN TANT QUE
FIN FONCTION
```


ÉCOLE POLYTECHNIQUE MONTRÉAL

Algorithme de Bresenham

Que faire si:

- la pente n’est pas entre 0 et 1 ?
- dx < 0 ou dy < 0 ?

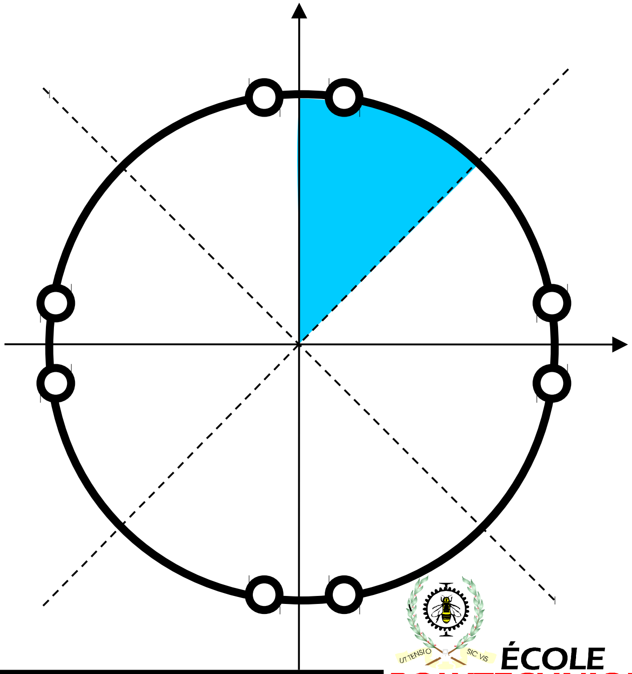
Utiliser la symétrie, selon le cas:

- Balayage par rapport à y plutôt qu’à x
- Inverser le point de départ et le point d’arrivée
- Remplacer y1 = y1 + 1 par y1 = y1 – 1
- Combiner ces stratégies

Algorithme du point milieu, cercle

- On suppose le centre à l'origine
- Le rayon r est un entier
- On trace seulement le deuxième octant
- Les autres octants sont tracés par symétrie

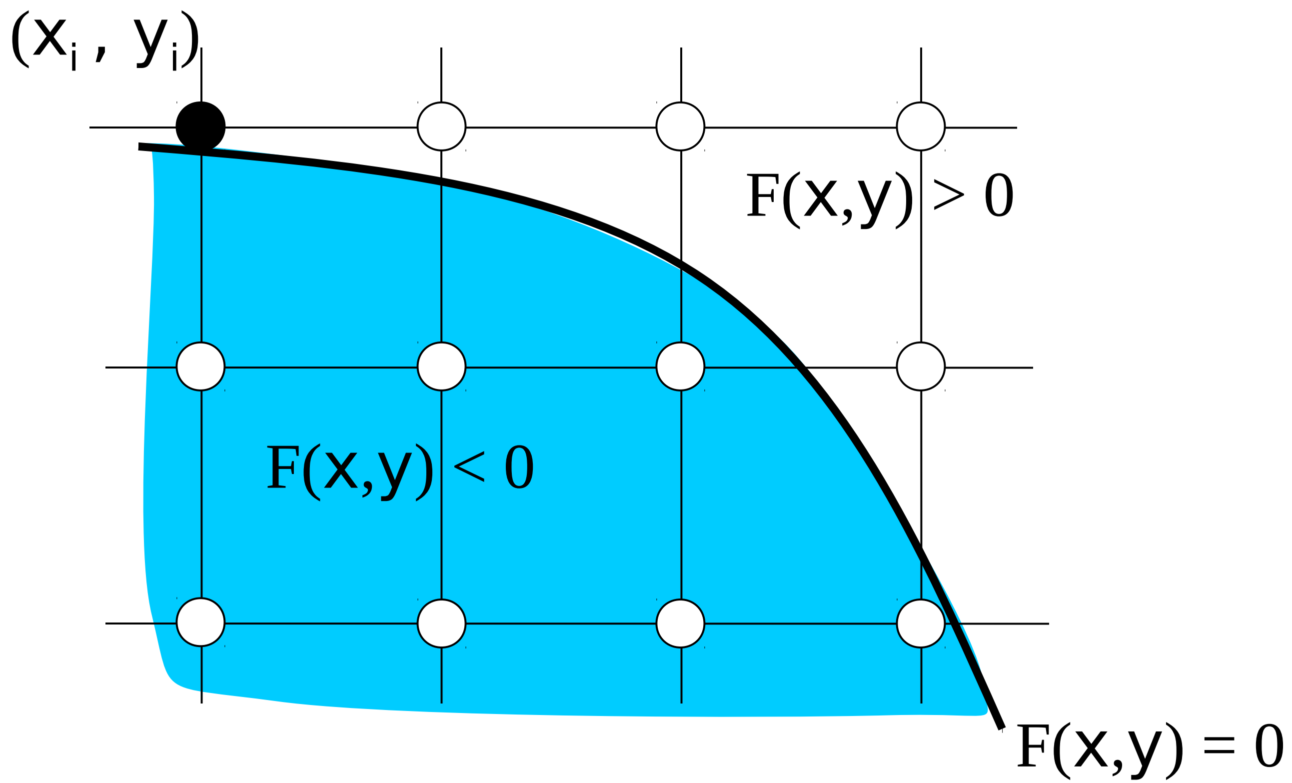
```
FONCTION AllumePixelEtSymétries( x, y )
 AllumePixels(
  ( x, y), ( y, x),
  (-x, y), (-y, x),
  ( x,-y), ( y,-x),
  (-x,-y), (-y,-x) )
FIN FONCTION
```



ÉCOLE POLYTECHNIQUE MONTRÉAL

Algorithme du point milieu, cercle

Forme implicite du cercle : $F(x, y) = x^2 + y^2 - r^2$

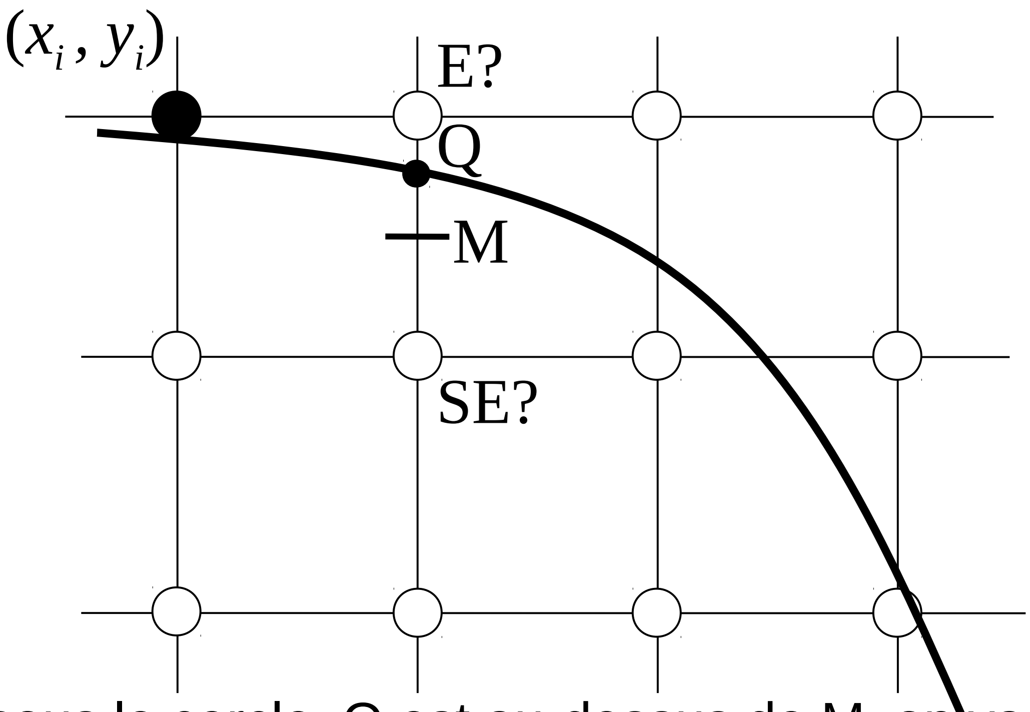


ÉCOLE POLYTECHNIQUE MONTRÉAL

Algorithme du point milieu, cercle

On calcule la valeur de F(x, y) au point M, noté d_i:

$$d_i = F(x_i + 1, y_i - ½) = (x_i + 1)^2 + (y_i - ½)^2 - r^2$$



$d_i < 0$: M est sous le cercle, Q est au-dessus de M, on va à l'E

$d_i \geq 0$: M est au-dessus du cercle, Q est sous M, on va au SE

Algorithme du point milieu, cercle

Comme pour Bresenham, on peut trouver que:

- Au départ :

  $d_0 = 1.25 - r$

- Si $d_i < 0$ alors on est allé à l'E :

  $d_{i+1} = d_i + 2\,x_i + 3$

- Sinon, on est allé au SE :

  $d_{i+1} = d_i + 2\,x_i - 2\,y_i + 5$

Algorithme du point milieu, cercle

On utilise la fraction 1.25, comment peut-on l'enlever?

Si on pose $h_i = d_i - 0.25$, on peut remplacer l'initialisation par:

$h_0 = 1 - r$

On devrait donc aussi modifier la comparaison par:

« Si $h_i < -0.25$ alors … »

Cependant, on remarque que h_i sera toujours un entier donc:

$h_i < -0.25 \Leftrightarrow h_i < 0$

On peut donc seulement changer l'initialisation et remplacer d_i par h_i ailleurs.

ÉCOLE POLYTECHNIQUE MONTRÉAL

Algorithme du point milieu, cercle

```
FONCTION Cercle( rayon )
   x = 0
   y = rayon
   h = 1 - rayon
   AllumePixelEtSymétries( x, y )
   TANT QUE  y > x          // ON RESTE DANS 2e OCTANT
    SI h < 0 ALORS          // EST
        h  = h + 2*x + 3
    SINON                   // SUD-EST
        h  = h + 2*(x-y) + 5
        y = y - 1
    FIN SI
    x = x + 1
    AllumePixelEtSymétries( x, y )
   FIN TANT QUE
FIN FONCTION
```