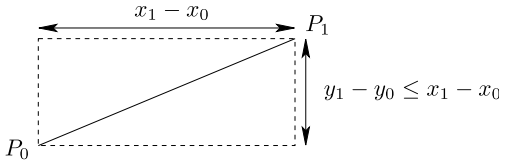
Infographie - M1

Chapitre 2 - Tracés de segments et de cercles

V.Padovani
Equipe Preuves, Programmes et Systèmes, Université Paris 7

1 Tracés de segments

On considère le problème consistant à tracer à l'écran la représentation d'un segment allant d'un point $P_0 = (x_0, y_0)$ à un point $P_1 = (x_1, y_1)$, de manière à ce que les pixels allumés soient les plus proches possibles des points du segment "réel".



On fera les hypothèses suivantes, auxquelles il est toujours possible de se ramener par échanges et/ou symétries :

- P_0 est "à gauche" de P_1, *i.e.* $x_0 \leq x_1$,
- la *pente* du segment $\delta = (y_1 - y_0)/(x_1 - x_0)$ est comprise entre -1 et 1.

On suppose donc que la portion d'écran rectangulaire contenant le segment P_0, P_1 est plus large que haute. Les segments de pixels représentant $[P_0P_1]$ seront donc tous horizontaux. On supposera en outre que P_0 et P_1 sont à coordonnées entières.

1.1 Algorithmes naïfs

On note *Arrondi* la fonction $x \mapsto \lfloor x + \frac{1}{2} \rfloor$ (partie entière inférieure de $x + \frac{1}{2}$). Pour tout réel x, *Arrondi*(x) vaut : la partie entière inférieure de x si $x = n + \frac{1}{2}$ avec $n \in \mathbb{N}$; l'entier le plus proche de x sinon.

1.1.1 Algorithme incrémental

1. $\delta \leftarrow (y_1 - y_0)/(x_1 - x_0)$,
2. $y_c \leftarrow y_0$,
3. pour x_c allant de x_0 à x_1, faire :
   (a) afficher le pixel en $(x_c, \mathit{Arrondi}(y_c))$,
   (b) $y_c \leftarrow y_c + \delta$.

Le problème posé par cette méthode est évidemment son manque de précision. Dans une implémentation réelle, la pente δ sera représentée par une variable de

type `float` ou `double`, et la valeur reçue par cette variable ne sera en général qu'une approximation de la pente réelle du segment. A chaque ajout de δ à y_c, la valeur de y_c s'écartera toujours d'avantage de l'ordonnée du point d'abscisse x_c appartenant réellement au segment.

1.1.2 Algorithme multiplicatif

1. $\delta \leftarrow (y_1 - y_0)/(x_1 - x_0)$,
2. pour x_c allant de x_0 à x_1, faire :
   (a) $y_c \leftarrow y_0 + (x_c - x_0) * \delta$.
   (b) afficher le pixel en $(x_c, \mathit{Arrondi}(y_c))$,

Cette méthode produit un meilleur résultat mais, là encore, la valeur reçue par y_c avant affichage n'est qu'une approximation.

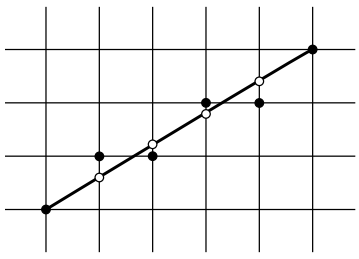
1.2 Algorithme de Bresenham

L'algorithme de Bresenham ou *algorithme de la ligne médiane*, élimine totalement le problème de l'imprécision des calculs, en ne se servant que de variables entières - cette possibilité existe en théorie, en remarquant que si les coordonnées des points sont entières, la pente de la droite est rationnelle, donc finiment représentable. On supposera cette pente de la droite est comprise entre 0 et 1.

1.2.1 Version "réelle"

Voici tout d'abord la version "réelle" de l'algorithme, inexacte, mais servant de point de départ à sa version optimisée. La boucle principale de cette fonction parcourt tous les x_c de x_0 à x_1. On garde en mémoire dans une variable e la différence entre l'ordonnée courante y_c et l'ordonnée du point du segment d'abscisse x_c.

La variable e est aussi appelée *variable de décision* : c'est à partir de sa valeur que l'on décide ou non d'incrémenter l'ordonnée courante y_c. Dans la version ci-dessous, on incrémente y_c dès que e dépasse $\frac{1}{2}$.



1. $\delta_x \leftarrow (x_1 - x_0)$
2. $\delta_y \leftarrow (y_1 - y_0)$
3. $y_c \leftarrow y_0$
4. $e \leftarrow 0$
5. pour x_c allant de x_0 à x_1, faire :
   (a) afficher le pixel en (x_c, y_c),
   (b) si $e + \delta_y/\delta_x > 1/2$,
   - $y_c \leftarrow y_c + 1$,
   - $e \leftarrow e + \delta_y/\delta_x - 1$

   sinon
   - $e \leftarrow e + \delta_y/\delta_x$

Voici à présent comment on obtient la version *discrète* de l'algorithme de la ligne médiane. Dans l'algorithme initial, ci dessus, l'incrémentation de y_c est soumise à la condition

$$e + \frac{\delta_y}{\delta_x} > \frac{1}{2}$$

Une condition équivalente, ne nécessitant que des sommes et produits de valeurs entières, est :

$$(2 \times \delta_x \times e) + (2 \times \delta_y) - \delta_x > 0$$

Reprenons l'algorithme précédent. On effectue le changement de variable

$$e_d \leftarrow (2 \times \delta_x \times e) + (2 \times \delta_y) - \delta_x$$

(e_d pour "erreur discrète"). L'initialisation $e \leftarrow 0$ devient :

$$e_d \leftarrow (2 \times \delta_y) - \delta_x$$

Le test $e + \delta_y/\delta_x > 1/2$ devient simplement :

$$e_d > 0$$

Puisque e_d vaut $(2 \times \delta_x \times e) + (2 \times \delta_y) - \delta_x$, ajouter δ_y/δ_x à e revient à ajouter $(2 \times \delta_x \times (\delta_y/\delta_x))$ à e_d, soit encore à lui ajouter $2 \times \delta_y$. L'affectation $e \leftarrow e + \delta_y/\delta_x$ devient donc

$$e_d \leftarrow e_d + (2 \times \delta_y)$$

De même, ajouter $\delta_y/\delta_x - 1$ à e revient à ajouter $(2 \times \delta_x \times (\delta_y/\delta_x - 1))$ à e_d, soit encore à lui ajouter $2 \times \delta_y - 2 \times \delta_x$. L'affectation $e \leftarrow e + \delta_y/\delta_x - 1$ devient donc

$$e_d \leftarrow e_d + (2 \times \delta_y) - (2 \times \delta_x)$$

Voici à présent, avec ce changement de variable, la version discrète de l'algorithme de la ligne médiane, c'est-à-dire l'algorithme de Bresenham. Les deux incréments possibles pour e_d sont pré-calculés, afin de minimiser encore le nombre d'opérations :

1. $\delta_x \leftarrow (x_1 - x_0)$
2. $\delta_y \leftarrow (y_1 - y_0)$
3. $y_c \leftarrow y_0$
4. $incr_1 \leftarrow (2 \times \delta_y)$
5. $incr_2 \leftarrow incr_1 - (2 \times \delta_x)$
6. $e_d \leftarrow incr_1 - \delta_x$
7. pour x_c allant de x_0 à x_1, faire :
   (a) afficher le pixel en (x_c, y_c),
   (b) si $e_d > 0$,
   - $y_c \leftarrow y_c + 1$,
   - $e_d \leftarrow e_d + incr_2$

   sinon
   - $e_d \leftarrow e_d + incr_1$

2 Tracés de cercles

Le problème du tracé de cercles admet plusieurs solutions naïves, récursives ou non, basées sur le calcul de sinus et de cosinus. Ces solutions sont à la fois imprécises et gourmandes en temps de calcul. L'algorithme ci-dessous, dû à Bresenham, résout ces deux problèmes en ne considérant que des variables entières et des opérations élémentaires - il faut bien sûr supposer que les coordonnées du centre du cercle et son rayon sont entiers.

2.1 Algorithme de Bresenham pour les cercles

La méthode suivante permet de dessiner le second octant (entre les positions "12h" et "13h30") du cercle de rayon entier R centré en $(0, 0)$. Le traitement général s'obtient par symétries et translations. Noter que cette version de l'algorithme de Bresenham est encore optimisable, mais ses versions optimales sont plus difficilement lisibles.

On part du pixel $(x, y) = (0, R)$. A chaque étape, on allume le pixel (x, y). Le couple (x, y) est ensuite transformé :

- soit en $(x + 1, y)$ (le point "haut"),
- soit en $(x + 1, y - 1)$ (le point "bas").

Le traitement est répété tant qu'on a $y > x$. Pour déterminer laquelle des deux ordonnées choisir, on garde en mémoire les deux quantités suivantes (h pour "haut", b pour "bas") :

- $\Delta_h(x,y) = (x+1)^2 + y^2 - R^2$,
- $\Delta_b(x,y) = (x+1)^2 + (y-1)^2 - R^2$.

Noter que $\Delta_b(x,y) = \Delta_h(x,y) - 2y + 1$. Voici un peu plus précisément à quoi correspondent ces deux quantités. Le point *réel* du cercle d'abscisse $x+1$ est de coordonnées $(x+1, y_r)$ avec

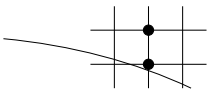
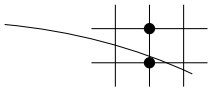
$$(x+1)^2 + y_r^2 - R^2 = 0$$

On donc :

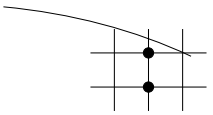
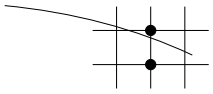
$$\Delta_h(x,y) - y^2 + y_r^2 = 0$$
$$\Delta_h(x,y) = y^2 - y_r^2$$
$$\Delta_b(x,y) - (y-1)^2 + y_r^2 = 0$$
$$\Delta_b(x,y) = (y-1)^2 - y_r^2$$

Autrement dit, $\Delta_h(x,y)$ et $\Delta_b(x,y)$ valent les erreurs signées entre les *carrés des ordonnées* des deux points candidats et celle du point appartenant réellement au cercle. Les propriétes qui suivent sont facilement vérifiables :

$0 \leq \Delta_b(x,y)$ $\quad$ $0 < -\Delta_b(x,y) < \Delta_h(x,y)$

$\Delta_h(x,y) \leq 0$ $\quad$ $0 < \Delta_h(x,y) \leq -\Delta_b(x,y)$

- Si $0 \leq \Delta_b(x,y)$ alors :
  - $(x+1, y-1)$ est au dessus du point réel.
  - $(x+1, y)$ est plus au dessus encore.
  - donc $(x+1, y-1)$ est le plus proche du point réel.
- Si $\Delta_h(x,y) \leq 0$ alors :
  - $(x+1, y)$ est en dessous du point réel.
  - $(x+1, y-1)$ est plus en dessous encore.
  - donc $(x+1, y)$ est le point le plus proche du point réel.

Dans tous les autres cas $\Delta_b(x,y) < 0 < \Delta_h(x,y)$, donc y_r est compris entre y et $(y-1)$, mais n'est égal à aucun des deux – sinon, l'une des deux quantités $\Delta_b(x,y), \Delta_h(x,y)$ serait nulle. Clairement,

- $(x+1, y-1)$ est le point le plus proche si $y - y_r > y_r - (y-1)$,
- $(x+1, y)$ est le point le plus proche si $y - y_r < y_r - (y-1)$

Nous allons démontrer les deux équivalences suivantes :

$$\begin{array}{ll} (1) & y - y_r > y_r - (y-1) \Leftrightarrow 0 < -\Delta_b(x,y) < \Delta_h(x,y) \\ (2) & y - y_r < y_r - (y-1) \Leftrightarrow 0 < \Delta_h(x,y) \leq -\Delta_b(x,y) \end{array}$$

On a :

$$\begin{array}{rc} & 0 < -\Delta_b(x,y) < \Delta_h(x,y) \\ \Leftrightarrow & \Delta_h(x,y) + \Delta_b(x,y) > 0 \\ \Leftrightarrow & \Delta_h(x,y) + \Delta_h(x,y) - 2y + 1 > 0 \\ \Leftrightarrow & 2\Delta_h(x,y) - 2y + 1 > 0 \\ \Leftrightarrow & 2\Delta_h(x,y) - 2y \geq 0 \end{array}$$

La toute dernière équivalence résulte du fait que $\Delta_h(x,y)$ et y sont des quantités entières. On a d'autre part :

$$\begin{array}{rc} & y - y_r > y_r - (y-1) \\ \Leftrightarrow & y - y_r > y_r - y + 1 \\ \Leftrightarrow & 2y - 1 > 2y_r > 0 \\ \Leftrightarrow & (2y-1)^2 > (2y_r)^2 \\ \Leftrightarrow & 4y^2 + -4y + 1 > 4y_r^2 \\ \Leftrightarrow & 4y^2 - 4y_r^2 - 4y + 1 > 0 \\ \Leftrightarrow & 4\Delta_h(x,y) - 4y + 1 > 0 \\ \Leftrightarrow & 2\Delta_h(x,y) - 2y + \frac{1}{2} > 0 \\ \Leftrightarrow & 2\Delta_h(x,y) - 2y \geq 0 \end{array}$$

Autrement dit, l'équivalence (1) est bien vérifiée.

Démontrons (2). Rappelons que l'on a supposé $\Delta_b(x,y) < 0 < \Delta_h(x,y)$. En considérant les négations des composantes de (1), on obtient :

$$(3) \quad y - y_r \leq y_r - (y-1) \Leftrightarrow 0 < \Delta_h(x,y) \leq -\Delta_b(x,y)$$

Un raisonnement similaire à la seconde partie de la démonstration de (1) permet de démontrer

$$y - y_r = y_r - (y-1) \Leftrightarrow 2\Delta_h(x,y) - 2y + \frac{1}{2} = 0$$

mais la somme d'une quantité entière et de $1/2$ ne peut être nulle, donc

$$(4) \quad y - y_r \leq y_r - (y-1) \Leftrightarrow y - y_r < y_r - (y-1)$$

La conjonction de (3) et (4) donne immédiatement (2). En conclusion :

- Si $0 < -\Delta_b(x,y) < \Delta_h(x,y)$, alors $(x+1, y-1)$ est le point le plus proche.
- Si $0 < \Delta_h(x,y) \leq \Delta_b(x,y)$, alors $(x+1, y)$ est le point le plus proche.

On remplace (x, y) par (x', y'). Les nouvelles valeurs de Δ_h et Δ_b peuvent être recalculées comme suit :

- Si $(x', y') = (x + 1, y)$ on a :

  $$\Delta_h(x+1, y) - \Delta_h(x, y) = (x+2)^2 - (x+1)^2 = 2x + 3$$
  $$\Delta_b(x+1, y) - \Delta_b(x, y) = (x+2)^2 - (x+1)^2 = 2x + 3$$

  donc

  $$\Delta_h(x', y') = \Delta_h(x, y) + 2x + 3$$
  $$\Delta_b(x', y') = \Delta_b(x, y) + 2x + 3.$$

- Sinon $(x', y') = (x + 1, y - 1)$ et :

  $$\begin{aligned}\Delta_h(x+1, y-1) - \Delta_h(x, y) &= (x+2)^2 - (x+1)^2 + (y-1)^2 - y^2 \\ &= (2x+3) - 2y + 1 \\ &= 2x - 2y + 4 \\ \Delta_b(x+1, y-1) - \Delta_b(x, y) &= (x+2)^2 - (x+1)^2 + (y-2)^2 - (y-1)^2 \\ &= (2x+3) - 2y + 3 \\ &= 2x - 2y + 6\end{aligned}$$

  donc

  $$\Delta_h(x', y') = \Delta_h(x, y) + 2(x - y) + 4.$$
  $$\Delta_b(x', y') = \Delta_b(x, y) + 2(x - y) + 6.$$