Quand les segments ressemblent à des escaliers...

André Boileau, Section didactique, Département de mathématiques, UQAM

Quand vous vous servez d'un logiciel de dessin sur ordinateur, celui-ci vous permet de tracer à main levée un certain nombre de formes élémentaires : rectangles, cercles, courbes « libres », etc. Les formes disponibles varient avec le logiciel utilisé, mais je ne connais aucun logiciel de dessin qui ne vous permette de tracer un simple segment reliant deux points. Voici comment ça se passe habituellement : avec la souris, vous amenez le curseur à une extrémité du segment que vous voulez tracer et vous pressez sur un bouton de la souris; tout en maintenant ce bouton enfoncé, vous déplacez la souris jusqu'à l'autre extrémité du segment et vous relâchez le bouton. Pendant tout le temps où le bouton reste enfoncé, le logiciel trace le segment reliant le point initial et la position courante de la souris (en n'oubliant pas d'effacer les segments précédents).

Ce processus nous semble tellement naturel que nous n'y prêtons plus guère attention. Mais essayons de regarder ceci avec des yeux neufs, comme si c'était la première fois. On peut se représenter un écran d'ordinateur comme formé de carrés minuscules appelés pixels, disposés en rangées et en colonnes. De nos jours, pour un écran donné, on peut faire varier ce nombre de pixels. Sur l'écran de l'ordinateur où j'écris ces lignes, on peut choisir entre diverses configurations allant de 480 lignes de 640 pixels jusqu'à 1024 lignes de 1280 pixels.

Dans la figure 1, on imagine avoir agrandi une portion de l'écran dans laquelle deux pixels ont été désignés comme les extrémités d'un segment. On a aussi tracé le segment reliant les milieux des deux pixels; mais un tel tracé ne peut se faire sur l'écran de l'ordinateur, puisque seuls les pixels complets peuvent être colorés. Quels pixels doit-on colorer pour mieux représenter le segment reliant nos deux points? Dans le cas particulier présenté ici, il y a deux solutions possibles, représentées dans les figures 2 et 3.

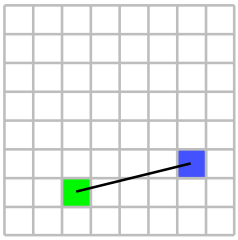

Figure 1

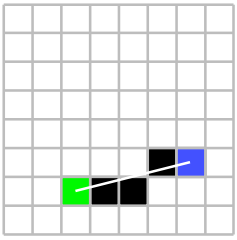

Figure 2

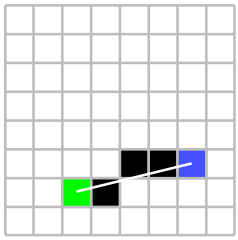

Figure 3

Dans un cas particulièrement simple comme ci-dessus, on peut voir intuitivement les solutions possibles. Mais les concepteurs de logiciels trouveraient très difficile d'incorporer une telle intuition dans leur programmation, d'autant plus que la ou les solutions doivent être trouvées pour *tous* les segments reliant deux pixels. La meilleure façon de procéder consiste donc à faire appel aux mathématiques.

D'un point de vue mathématique, chaque pixel est un carré. Mais, pour étudier la question, on va plutôt représenter chaque pixel par un point bien défini : son centre. Ceci nous permettra de désigner plus simplement nos pixels : si on suppose que le pixel en bas à gauche a (1,1) comme coordonnées et que les pixels carrés ont un côté de longueur unité, alors les deux pixels ci-dessus auront comme coordonnées (3,2) et (7,3).

Trouvons maintenant l'équation de la droite passant par nos deux points-pixels. Si (x,y) est un point de cette droite, on sait qu'on aura égalité des pentes des droites déterminées par les deux points (3,2) et (x,y) d'une part, et les deux autres points (3,2) et (7,3) d'autre part. Ceci peut s'exprimer par l'égalité suivante $\frac{y-2}{x-3}=\frac{3-2}{7-3}$, d'où l'on tire l'équation de la droite : $y=\frac{1}{4}x+\frac{5}{4}$.

Comment utiliser cette équation pour décider quels pixels colorer? Une possibilité toute naturelle est de considérer toutes les colonnes intermédiaires, dont les centres ont des coordonnées entières en x : 4, 5 et 6 dans notre exemple. On utilise alors l'équation de la droite pour calculer les ordonnées correspondantes : $\frac{9}{4}=2{,}25$, $\frac{10}{4}=2{,}5$ et $\frac{11}{4}=2{,}75$ dans notre exemple. Puis on choisit de colorer les pixels des colonnes correspondantes dont les centres sont les plus près des valeurs calculées, soit 2, 2 ou 3, et 3. On voit donc pourquoi on avait deux solutions au départ : comme le point (5, $\frac{10}{4}$) est à égale distance des points-pixels (5,2) et (5,3), on peut choisir de colorer l'un ou l'autre de ces points.

Une première tentative de solution générale

Comme nous l'avons souligné, nous sommes à la recherche d'une solution *générale* du problème de tracer un segment reliant deux pixels. Notre étude d'un cas particulier nous a permis de débroussailler le chemin, et nous sommes maintenant prêts à proposer une méthode générale :

Pour relier les pixels (a,b) et (c,d) par un segment (méthode #1)

1. Calculer l'équation de la droite passant par ces deux points

   À partir de l'égalité $\frac{y-b}{x-a}=\frac{d-b}{c-a}$, on obtient l'équation $y=\left(\frac{d-b}{c-a}\right)x+\left[b-a\left(\frac{d-b}{c-a}\right)\right]$.

   Si nous désignons $\left(\frac{d-b}{c-a}\right)$ par m et $\left[b-a\left(\frac{d-b}{c-a}\right)\right]$ par B, l'équation devient $y=mx+B$.

2. Pour toutes les valeurs entières de x comprises entre a et c
   (ces valeurs entières correspondent aux colonnes intermédiaires)
   - Utiliser cette équation pour calculer la valeur correspondante de y : $y=mx+B$.
   - Arrondir cette valeur à l'entier n le plus proche : $n=arrondir(x)$.
     (Notons que, dans certains cas, nous aurons à choisir entre deux possibilités.)
   - Colorer le pixel (x,n).

Certains d'entre vous sont peut-être satisfaits de la méthode ci-dessus, et d'autres entretiennent peut-être des doutes. Mais, dans un cas comme dans l'autre, il est plus sage de faire une bonne vérification expérimentale. On pourra alors observer que cette méthode est couronnée de succès dans bon nombre de cas, comme l'illustrent les exemples ci-dessous.

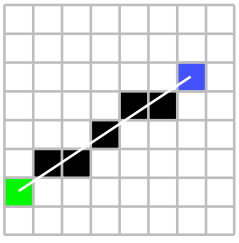

Figure 4

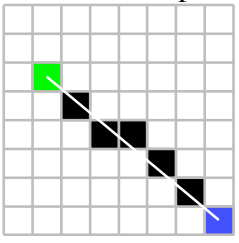

Figure 5

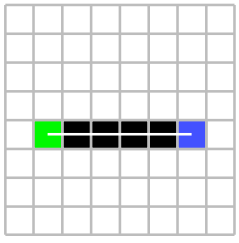

Figure 6

Mais on constatera aussi que la méthode est prise en défaut dans plusieurs autres cas, comme nous le révèlent les exemples suivants : la chaîne de pixels reliant nos deux pixels initiaux comporte des « trous »... Pire encore, dans le cas où nos pixels sont sur une droite verticale, le programme provoque une erreur d'exécution (puisqu'on demande alors de calculer une pente infinie)!

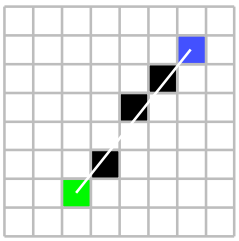

Figure 7

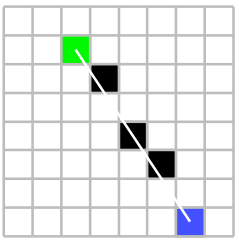

Figure 8

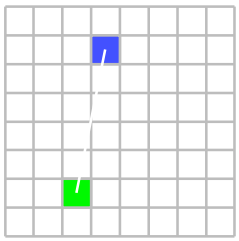

Figure 9

Dans quels cas notre méthode fonctionne-t-elle correctement, et dans quels cas est-elle prise en défaut? L'exemple de la figure 9 est particulièrement révélateur : comme il n'y a *aucune* colonne intermédiaire, les deux pixels ne pourront évidemment pas être reliés entre eux. Plus généralement, si on veut espérer boucher les « trous » entre nos pixels initiaux, il faut que le nombre de pixels tracés par notre méthode (qui est égal au nombre de colonnes intermédiaires entre nos pixels initiaux) soit au moins égal au nombre de lignes intermédiaires, ce qui n'est évidemment pas toujours le cas[1].

Une seconde tentative, couronnée de succès

Notre méthode semble donc fonctionner quand le nombre de colonnes intermédiaires est supérieur ou égal au nombre de lignes intermédiaires. On peut imaginer mettre au point une seconde méthode, applicable quand cette condition n'est pas vérifiée, puis de combiner ces deux méthodes en une méthode générale.

En fait, cette seconde méthode ressemblera beaucoup à la première : si le nombre de lignes intermédiaires dépasse le nombre de colonnes intermédiaires, on peut imaginer tourner notre écran de 90 degrés. De cette façon, les lignes deviennent colonnes et les colonnes des lignes, on peut appliquer notre méthode initiale dans ce nouveau contexte, puis retourner à nouveau notre écran de 90 degrés dans l'autre sens pour retrouver à la position initiale avec notre solution.

Une façon équivalente de procéder est de chercher à établir l'équation de la droite sous la forme $x = f(y)$. Si on calcule maintenant la pente dans le système (y,x) plutôt que (x,y), on obtient l'égalité[2] $\frac{x-a}{y-b} = \frac{c-a}{d-b}$, d'où l'on tire l'équation $x = \left(\frac{c-a}{d-b}\right)y + \left[a - b\left(\frac{c-a}{d-b}\right)\right]$.

Si nous désignons $\left(\frac{c-a}{d-b}\right)$ par m' et $\left[a - b\left(\frac{c-a}{d-b}\right)\right]$ par B', l'équation devient $x = m'y + B'$.

---

[1] Le lecteur intéressé pourra vérifier que c'est le cas précisément quand la valeur absolue de la pente de la droite est inférieure ou égale à 1.

[2] Je pose la question au lecteur : dans ce cas (comme dans le cas précédent), pourquoi est-on certain que ces rapports seront toujours égaux, et ce quel que soit le point (x,y) choisi sur la droite passant par (a,b) et (c,d)? Et pourquoi est-on certain que ces rapports seront inégaux si on choisit le point (x,y) en dehors de la droite passant par (a,b) et (c,d)?

Nous sommes maintenant prêts à décrire précisément notre deuxième méthode

Pour relier les pixels (a,b) et (c,d) par un segment (méthode #2)
1. Calculer l'équation de la droite passant par ces deux points : $x = m' y + B'$
2. Pour toutes les valeurs entières de y comprises entre b et d
   (ces valeurs entières correspondent aux lignes intermédiaires)
   - Utiliser cette équation pour calculer la valeur correspondante de x.
   - Arrondir cette valeur à l'entier n le plus proche : $n = arrondir(x)$.
     (Notons que, dans certains cas, nous aurons à choisir entre deux possibilités.)
   - Colorer le pixel (n,y).

puis notre méthode globale, combinant les deux précédentes :

Pour relier les pixels (a,b) et (c,d) par un segment (méthode globale)
1. Calculer u le nombre de colonnes intermédiaires entre nos deux pixels
   et v le nombre de lignes intermédiaires entre nos deux pixels.
2. Si $u \leq v$ alors appliquer la méthode #1
   sinon appliquer la méthode #2.

Cette fois, êtes-vous convaincus que notre méthode globale fonctionnera toujours correctement, ou pensez-vous qu'elle n'aura pas réglé tous les cas? Quelle que soit votre opinion, il est plus prudent d'effectuer une vérification expérimentale : on constate alors qu'elle semble toujours fonctionner correctement.

Rendre notre méthode plus efficace : le cas de la multiplication qui disparaît

Si nous voulions résoudre le problème du tracé informatique des segments pour notre propre compte, nous pourrions maintenant nous déclarer satisfaits : nous avons mis au point une solution générale, qui fait l'affaire dans la plupart des situations. Mais si nous nous mettons dans la peau d'un informaticien voulant développer une solution qui sera utilisée par tous les utilisateurs d'un ordinateur donné, il va sans dire que nous devons rechercher une méthode qui soit la plus rapide possible. Il y a des utilisateurs qui vont chercher à tracer des figures complexes comportant un grand nombre de segments (comme celle de la figure 10, ou pire encore), et qui vont comparer la vitesse d'exécution à celle de votre compétiteur. Comment peut-on rendre notre méthode plus efficace?

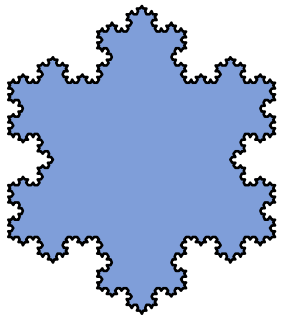

Figure 10

Il semble clair qu'on ne pourra économiser sur le temps nécessaire au tracé des pixels eux-mêmes : quelle que soit la méthode utilisée, le nombre de pixels dessinés sera le même. Il nous faut plutôt regarder les calculs nécessaires pour déterminer quels pixels seront colorés, puisque c'est une opération qui sera effectuée plusieurs fois[3] pour chaque segment.
On voit que le calcul $n = arrondir(mx + B)$ [cas où la méthode#1 est utilisée] ou $n = arrondir(m' y + B')$ [cas où la méthode#2 est utilisée] est effectué autant de fois que le nombre de pixels tracés. Ce calcul comporte une multiplication, une addition et un arrondi. Mais,

[3] Plus précisément, ce nombre de fois sera le maximum des deux nombres suivants : le nombre de colonnes intermédiaires et le nombre de lignes intermédiaires entre nos pixels aux extrémités.

on réfléchissant un peu, on constate qu'on peut éliminer la multiplication qui, pour l'ordinateur comme pour l'humain, est une opération beaucoup plus lourde que l'addition.

En effet, supposons que nous venons de calculer, dans le cadre de la méthode #1, $y = mx + B$. Avant d'arrondir, nous gardons précieusement la valeur y. Comme nous procédons par colonnes, la prochaine valeur de x sera en fait x+1 et la nouvelle valeur de y sera alors

$$m(x+1) + B = (mx + B) + m = y + m.$$

Il suffira donc d'ajouter m à la valeur précédente de y pour en obtenir la nouvelle valeur, qu'il nous faudra ensuite arrondir.

On peut faire encore mieux

À première vue, on ne voit pas trop comment améliorer encore notre méthode de tracé de segments : on a quand même réussi à éliminer une multiplication, pour ne garder qu'une addition et un arrondi. Pour voir comment continuer, nous devons remarquer que notre méthode amène l'ordinateur à calculer sur des nombres décimaux[4]. En effet, la pente (que ce soit m ou m') est définie comme un quotient d'entiers, et ne sera qu'exceptionnellement entière. Et l'on peut voir qu'il en est de même pour B et de B'. Dans nos calculs ($mx + B$ ou $m'y + B'$), deux des trois nombres risquent donc d'être décimaux.

Ne pourrait-on pas transformer notre méthode de façon à ce qu'elle n'utilise que des nombres entiers, ce qui assurerait des calculs considérablement plus rapides? Nous allons voir que oui, en étudiant d'abord un exemple pour généraliser par la suite. Considérons donc la situation illustrée ci-contre par la figure 11. Notre stratégie sera la suivante : à partir de l'extrémité inférieure gauche du segment, nous allons placer horizontalement des pixels tout en calculant nos erreurs de placement (par rapport au segment idéal); et nous passerons à la ligne supérieure quand nos erreurs de placement nous indiqueront que le pixel de la ligne du haut est une meilleure approximation.

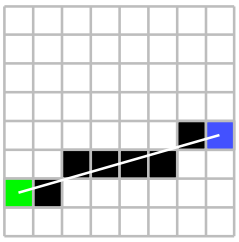

Figure 11

Rendons tout ceci plus précis. Comme les extrémités du segment sont (1,2) et (8,4), l'équation de la droite passant par ces points sera $y = \frac{2}{7}x + \frac{12}{7}$. Au départ, on place un pixel en (1,2) avec une erreur de 0 puisque l'extrémité est exactement sur la droite. On a vu, à la section précédente, qu'à chaque fois qu'on passait à la colonne suivante, la valeur théorique de y augmentait de m, qui est $\frac{2}{7}$ dans notre exemple. Donc, si on place le pixel sur la même ligne, l'erreur augmentera de $m=\frac{2}{7}$; tandis que si on le place à la ligne supérieure, l'erreur augmentera de m-1$=-\frac{5}{7}$. Voici donc comment ceci nous conduira à procéder dans notre exemple :

- Au départ, on place le pixel en (1,2) et l'erreur est de 0.
- On laisse le prochain pixel sur la même ligne, donc en (2,2); l'erreur est de $0+\frac{2}{7}=\frac{2}{7}$.
- Si on laissait le prochain pixel sur la même ligne, l'erreur serait de $\frac{2}{7}+\frac{2}{7}=\frac{4}{7}>\frac{1}{2}$.
  On place donc le prochain pixel à la ligne supérieure, en (3,3), et l'erreur est de $\frac{2}{7}-\frac{5}{7}=-\frac{3}{7}$.
- On laisse le prochain pixel sur la même ligne, donc en (4,3); l'erreur est de $-\frac{3}{7}+\frac{2}{7}=-\frac{1}{7}$.
- On laisse le prochain pixel sur la même ligne, donc en (5,3); l'erreur est de $-\frac{1}{7}+\frac{2}{7}=\frac{1}{7}$.

[4] Plus précisément, il s'agit de nombres rationnels. Mais on sait que l'ordinateur, dans le cas qui nous occupe, va devoir en utiliser une approximation décimale.

- On laisse le prochain pixel sur la même ligne, donc en (6,3); l'erreur est de $\frac{1}{7} + \frac{2}{7} = \frac{3}{7}$.
- Si on laissait le prochain pixel sur la même ligne, l'erreur serait de $\frac{3}{7} + \frac{2}{7} = \frac{5}{7} > \frac{1}{2}$.
  On place donc le prochain pixel à la ligne supérieure, en (7,4); l'erreur est de $\frac{3}{7} - \frac{5}{7} = -\frac{2}{7}$.
- On laisse le prochain pixel sur la même ligne, donc en (8,4); l'erreur est de $-\frac{2}{7} + \frac{2}{7} = 0$.
  Naturellement, l'erreur redevient zéro à l'autre extrémité.

Le lecteur aimerait peut-être impatiemment me faire remarquer que nous n'avons pas éliminé du tout les fractions! Mais on peut observer que si, au lieu de l'erreur, on utilise plutôt l'erreur multipliée par le dénominateur de la pente (soit 7 dans notre exemple), tous les calculs d'erreur se font sur des entiers. Et si on multiplie aussi par 2, on évite une comparaison avec $\frac{1}{2}$, et toute référence aux fractions est ainsi éliminée. Généralisons maintenant tout ceci, dans le cas[5] où les différences $\Delta x \quad [= c - a]$ et $\Delta y \quad [= d - b]$ sont positives, et où $\Delta y \leq \Delta x$:

Pour relier les pixels (a,b) et (c,d) par un segment (méthode entière, cas #1)
1. Poser u = a, v = b et erreur = 0.
2. Pour toutes les valeurs entières de u comprises entre a et c :
   - Tracer le pixel en (u,v).
   - Ajouter 1 à u, et Δy à l'erreur.
   - Si (erreur+erreur) dépasse Δx, alors ajouter 1 à v et soustraire Δx à l'erreur.

Et pour terminer

Ne nous le cachons pas, l'exposé que vous venez de lire était sévèrement handicapé par le médium utilisé : le livre. Vous n'avez pu effectuer vous-même les vérifications expérimentales, nécessaires non seulement pour valider méthodes étudiées, mais aussi pour développer une expérience concrète du sujet. Pour pallier en partie à ces lacunes, je vous donne rendez-vous à la page Web

http://www.math.uqam.ca/_boileau/Segments.html

où vous trouverez des ressources informatiques, d'éventuelles corrections à apporter, et mon adresse pour le cas où vous voudriez me contacter.

Nous n'avons fait qu'effleurer les bases mathématiques et algorithmiques du graphisme informatique. Le lecteur pourra aisément imaginer une foule de problèmes, allant du tracé de simples cercles à la représentation de figures tridimensionnelles sur un écran bidimensionnel, où les mathématiques jouent un rôle crucial. S'il veut en savoir plus, ce ne sont pas les livres sur le sujet qui manquent.

---

[5] On peut supposer que $\Delta x \geq 0$, puisqu'on peut toujours interchanger les points-pixels de départ. Nous laissons au lecteur le soin d'adapter notre méthode dans les autres cas, et de réunir tous ces cas en une méthode générale.