Algorithmes de Tracé de Droites

Rapport

BLQIS

Résumé

Les droites discrètes sont une notion fondamentale en informatique graphique et en traitement d'image. Contrairement aux droites continues, qui sont représentées par des équations mathématiques continues, les droites discrètes sont définies sur un ensemble fini de points discrets dans un espace discret tel qu'une grille ou une matrice. Ce concept est crucial dans plusieurs contextes, tel que le tracé approximatif de cercles, les algorithmes de ray-tracing, ou bien encore dans le cadre de remplissage de polygones.

Le tracé de droites discrètes implique la détermination d'une séquence finie de points dans un espace discret pour représenter une droite entre deux points donnés. Les algorithmes de tracé de droites visent à résoudre ce problème de manière efficace, en optimisant le choix des pixels utilisés tout en préservant autant que possible la fidélité visuelle de la droite.

Parmi les algorithmes classiques pour le tracé de droites discrètes, on trouve l'algorithme de Bresenham (dénommé B5), qui est largement utilisé pour sa simplicité et son efficacité. D'autres algorithmes, tels que ceux développés par Rokne, Wyvill & Wu, et Graham, Iyengar, se basent sur cet algorithme de référence et promettent une amélioration considérable en temps, tout en gardant une complexité, un coût d'initialisation et une performance dans le pire cas possible identiques.

L'objectif de ce rapport est d'évaluer la performance de l'algorithme du pas de 3 (1993), considéré comme le plus rapide des algorithmes susmentionnés, et reproduire les gains de performances annoncés[1].

Structure du projet

Les droites sont représentées par des points au sein de structures de tableaux en C, afin de simuler les pixels d'une image numérique. Le temps d'affichage venant grever ces valeurs, il a été omis lors du test de temps.

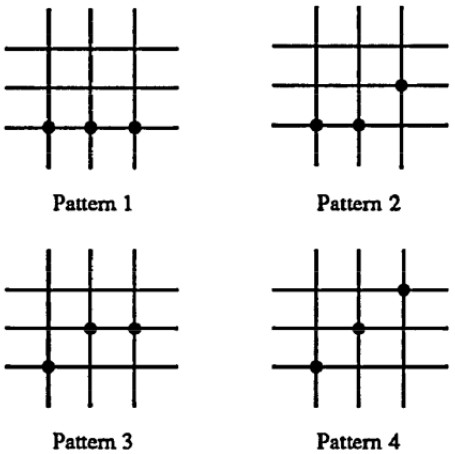
Chaque algorithme dispose cependant d'une fonction d'affichage qui permet la vérification des résultats obtenus et leur cohérence. Ainsi, des tests croisés sont effectués entre chaque algorithme et sont affichés lors de l'exécution du code source.

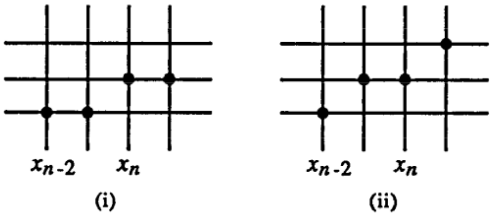
Le taux d'amélioration de performance est également implémenté et a permis une comparaison en temps des différents algorithmes. [**Figure 2**][**Figure 3**]

Algorithmes du pas itératif

Une amélioration de l'algorithme B5 consiste en l'incrément de x de 2 ou 3 en un tour de boucle, et réduire ainsi le nombre total de tours.
L'idée est de séparer l'octant en deux parties :
$0 < dy < dx < 2dy$ et $0 < dy < 2dy < dx$
où chaque partie a un ou plusieurs schémas spécifiques associés.
Ce qui vient fortement affecter la performance de ce type de d'algorithmes à pas itératif est le signe de delta. En effet, dès lors qu'il faut vérifier quel schéma d'incréments choisir (horizontale-oblique, oblique-horizontale, oblique-oblique-horizontal pour le pas de 3, etc...), il est nécessaire de revenir un cran en arrière (*delta(x - 1, y)*) et faire un test sur le signe de delta. Ces retours en arrière sont plus ou moins conséquents selon le type de droites à tracer.

Figure 1 : Schémas de tracé avec incréments horizontales et obliques (XX, XD, DX, DD, XDX, DXD)





Résultats

Figure 2 : Comparaison en µs des algorithmes sur des droites de 100 points.

Droite	B5	P2	P3	% Réduction (B5, P2)	
(100, 0)	18.88	17.35	17.32	8.27	0.20
(99, 13)	16.31	15.76	15.61	4.27	0.94
(97, 21)	17.27	15.30	15.21	11.92	0.58
(96, 25)	16.92	16.76	16.60	1.86	0.91
(95, 30)	17.83	17.49	14.82	16.92	15.28
(93, 34)	18.00	17.91	14.55	19.18	18.76
(92, 38)	17.78	16.07	13.96	21.45	13.09
(88, 46)	16.78	14.36	14.14	15.76	1.53
(84, 53)	14.06	13.82	12.02	14.55	13.02
(70, 70)	16.37	12.84	12.63	22.86	1.67

Figure 3 : Comparaison en µs des algorithmes sur des droites de 1000 points.

Droite	B5	P2	P3	% Réduction (B5, P2)	
(1000, 0)	183.92	176.06	175.27	4.71	0.45
(991, 130)	181.83	175.23	170.30	6.34	2.81
(976, 216)	189.72	169.69	164.60	13.24	3.00
(965, 258)	180.68	178.54	169.70	6.07	4.95
(953, 300)	186.59	173.25	146.71	21.37	15.32
(939, 342)	185.25	184.98	155.55	16.03	15.91
(923, 382)	182.44	176.88	158.49	13.13	10.39
(88, 46)	184.31	153.60	151.90	17.59	1.11
(84, 53)	178.25	163.01	136.60	23.37	16.20
(707, 707)	170.34	132.20	129.87	23.76	1.76

L'algorithme du pas de 3 est le plus efficace lorsque $3dy = dx$ et dy est un entier très grand.
A chaque itération, trois pixels sont dessinés, contre deux avec l'algorithme du pas de 2.

Conclusion

Nous observons bien une nette amélioration de au niveau performance de l'algorithme du pas de 3, par rapport au pas de 2 et B5. Une autre amélioration éventuelle est d'étendre cet algorithme à un pas de n comme l'a suggéré Graeme W. Gill en 1994, ou mieux encore : adapter le pas selon la pente de la droite (Boyer-Bourdin, 2000).

Bibliographie

- Phil Graham and S. Sitharamalyengar, Double- and Triple-Step Incremental Generation of Lines (1993)
- J.E. Bresenham. Run length slice algorithm for incremental lines, in Fundamental Algorithms for Computer Graphics (R.A. Earnshaw, Ed.), NATO ASI Series,Springer-Verlag: New York, 59-104 (1985)
- X. Wu and J.G. Rokne. Double-step incremental generation of lines and circles, Computer Vision, Graphics, and Image Processing 37(3). 331-344 (1987)