TP Synthèse d'images: Geometrie Algorithmique
CPE

durée - 4h

2012/2013

1 Introduction

Ce TP permet de réaliser un algorithme classique de géométrique algorithmique : La construction de polygone convexe d'un ensemble de points. Dans le cas présent, on implémentera l'algorithme dans un espace 2D.

2 Affichage de points

La fig. 1 permet de donner un exemple d'affichage de positions (x, y) directement en C++ sous OpenGL dans le contexte classique des TPs. Il peut s'adapter à tout autre affichage 2D ou 3D.

```
std::vector<v2d> v_vertices;

void scene::load_model()
{
    //fill
    int N=3684542;
    int N_sommets=8000;
    for(int k=0;k<N_sommets;++k)
    {
        v2 p=v2((qrand()%N)/double(N),(qrand()%N)/double(N));
        v_vertices.push_back(p);
    }
}

void scene::draw_scene()
{
    glPointSize(2.0);
    glLineWidth(4.0);
    glDisable(GL_LIGHTING);
    glColor3d(0,0,0);
    glBegin(GL_POINTS);
    for(unsigned int k=0;k<v_vertices.size();++k)
        glVertex2d(v_vertices[k].x(),v_vertices[k].y());
    glEnd();

}
```

positions (x,y)
aléatoires (uniforme)

affichage des positions

FIGURE 1 – Exemple d'initialisation et d'affichage de points du plan.

Question 1 *Initialisez les structures et affichez des points et des lignes dans le plan.*

3 Positionnement de points

Dans le cas le plus classique, les positions du plan sont distribuées aléatoirement suivant une loi normale.

Il est possible de générer des nombres aléatoires possédant une distribution uniforme avec l'appel *qrand()* en Qt.

On rappelle [1] que si a et b sont deux variables aléatoires de distribution uniforme sur $[0, 1]$, alors

$$\begin{cases} \sqrt{-2\ \ln(a)}\ \cos(2\pi b) \\ \\ \sqrt{-2\ \ln(a)}\ \sin(2\pi b)\ , \end{cases}$$

sont deux variables aléatoires qui suivent une distribution normale centrée de variance unitaire.

Question 2 *Implémentez le placement de sommets suivant une distribution normale en* (x, y).

4 Algorithme force-brute

Le premier algorithme de construction de l'ensemble convexe consiste en une approche naive de type force-brute. Son principe est le suivant :

1. On initialise le premier sommet de l'ensemble convexe en considérant un sommet faisant partie de celui-ci. (par exemple le sommet dont la coordonnée x est minimale).
2. On cherche le voisin du sommet courant laissant l'ensemble des autres positions à sa gauche. On obtient ainsi une arête du polygone convexe.
3. On réitère à l'étape précédente en partant du nouveau sommet ainsi trouvé jusqu'à rejoindre la position initiale.

On rappelle que pour tester si une position $\mathbf{p}$ se trouve à droite ou à gauche du vecteur $\mathbf{u} = \mathbf{p}_1 - \mathbf{p}_0$, on calcul le signe du déterminant : $\det(\mathbf{p} - \mathbf{p}_0, \mathbf{p}_1 - \mathbf{p}_0)$.

Complexité : Une évaluation permettant de savoir si un sommet laisse tous les autres à sa gauche nécessite un parcours et un test sur l'ensemble des positions. La complexité associée est donc en $\mathcal{O}(N)$. Cette évaluation se réalise ensuite elle-même sur tout les sommets jusqu'à trouver celui convenant. Trouver ce sommet particulier se fait donc en $\mathcal{O}(N^2)$. Enfin, on recommence l'opération pour tous les sommets du polygone convexe. Dans le cas d'une distribution normale on a environ $\log(N)$ sommets sur le polygone.
La complexité totale de construction du polygone est donc de $\mathcal{O}(N^2 \log(N))$.

Notons que l'initatialisation de la recherche du premier sommet est en $\mathcal{O}(N)$. Elle est donc négligeable en terme de complexité.

5 Algorithme du Quick-Hull

L'algorithme du Quick-Hull permet d'optimiser la complexité par rapport à l'algorithme de force brute.

Supposons que l'on connaisse trois sommets $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ appartenant au polygone convexe. L'algorithme est alors le suivant :

1. Eliminer de l'ensemble tous les sommets situés à l'intérieur du triangle ainsi délimité.
2. Pour chaque arête du triangle parcouru dans le sens trigonométrique, chercher le sommet à sa droite le plus éloigné de ce coté que l'on appellera $\mathbf{p}_i$.
3. Recommencer l'opération en considérant le triangle formé par les deux sommets de l'arête courante et $\mathbf{p}_i$.

1. Méthode de Box-Muller

Pour initialiser cet algorithme, on pourra considérer le triangle plat formé par $(\mathbf{p}_{\min}, \mathbf{p}_{\max}, \mathbf{p}_{\min})$, avec $\mathbf{p}_{\min}$ le sommet ayant la coordonnée x minimal, et $\mathbf{p}_{\max}$ le sommet ayant la coordonnée x maximale.

Notons que pour savoir si un point est situé à l'intérieur d'un triangle, on peut vérifier que celui-ci est situé à gauche de tous les côtés lorsqu'ils sont parcourus dans le sens trigonométrique.

Complexité : La suppression des sommets intermédiaires permet de réaliser des tests sur un ensemble de plus en plus petit de positions. On peut ainsi montrer que pour des distributions standard (ex. distribution normale), la complexité associée est en $\mathcal{O}(N \log(N))$.

Notez que l'on peut accélérer l'initialisation dans de nombreux cas en ne partant pas d'un triangle plat, mais d'un quadrilatère formé par les extrêmes des coordonnées en x et y lorsque celles-ci ne sont pas confondues.

On notera également que la structure de données doit permettre une suppression des sommets en $\mathcal{O}(1)$. En effet, utiliser un vecteur possède une suppression en $\mathcal{O}(N)$ se qui ferait perdre l'avantage d'utiliser un tel algorithme.

6 Travail demandé

Question 3 *Implémentez l'algorithme force brute et observez le temps de calcul en fonction du nombre de sommets. Vous vérifierez le bon fonctionnement de l'algorithme sur des cas simples.*

Question 4 *Implémentez l'algorithme du Quick-Hull et comparez son temps de calcul par rapport au cas précédent.*

Question 5 *Testez des cas de configurations différentes (distribution non normales) où l'algorithme du Quick-Hull est moins efficace.*