ENS Cachan Bretagne / Rennes
L3 Informatique, parcours R & I
Cours de programmation, 1er semestre 2015–2016

—

Projet 2

Luc Bougé, Martin Quinson*

13 octobre 2015

—

Id: projet2_Delaunay.tex 1595 2015−10−13 07:23:52Z bouge

Présentation du devoir

Ce projet est à faire par groupes de 2 (plus un groupe de 1 si nécessaire). La soutenance de ce projet aura lieu probablement le

*vendredi 6 novembre à **10h15**, sur le créneau du TD (à confirmer).*

Les codes et les diapos de soutenance (format PDF) sont à envoyer à martin.quinson@ens-rennes.fr pour le

*mercredi 4 novembre à **7h00**.*

Le rapport est à envoyer pour le

*dimanche 8 novembre à **18h00**.*

Une pénalité d'un point par heure sera appliquée en cas de retard pour le rendu du rapport.

Modalités de la soutenance. L'exposé de soutenance dure 10 minutes. Il est suivi de 10 minutes de questions. Il est souhaité que le groupe utilise un support visuel permettant de rapidement résumer le travail effectué (comprenant typiquement des captures d'écrans de résultats d'exécutions). Les démonstrations de programmes en direct sont aussi appréciées (quoique non obligatoires), mais faites attention à bien gérer votre temps dans ce cas.

*Rédigé d'après un document de Benoît Le Gouis, repris Pierre Karpman, repris d'un document de Matthieu Dorier, repris d'un document de Sébastien Chédor, repris d'un document de Thomas Gazagnaire.

Contenu du rendu. Le rendu se fait sous la forme d'une archive contenant :

— Un rapport rédigé (en format PDF) expliquant et motivant les choix de conception (non triviaux) que vous avez fait, ainsi que les extensions que vous avez ajoutées au sujet de base.
— Les diapos utilisés pour la soutenance.
— Le code source de vos programmes (si possible, proposez deux versions pour chaque programme : une version de base répondant strictement aux questions et une version complète avec toutes vos extensions).
— Un fichier *README* donnant les commandes nécessaires à la compilation du (des) programme(s).

L'archive sera nommée d'après la convention *Nom1Nom2*, les noms de famille étant ordonnés alphabétiquement (par ex. *Calvin*, *CalvinHobbes*, *CalvinDerkinsHobbes*). Elle sera envoyée par courrier électronique, l'objet duquel utilisera la même convention que ci-dessus, préfixée de *Prog1P1* (par ex. *Prog1P1CalvinHobbes*).

Remarque. *La qualité de la rédaction est très importante. La notation prend en compte le soin, la structure et l'orthographe*[1] *du rapport. L'aspect* esthétique *de vos programmes est aussi important. Vous ferez donc attention à écrire un code commenté, lisible, bien indenté, au choix de vos noms de variables, etc. Il est recommandé d'utiliser un* pretty-printer *Caml professionnel, par exemple* ocp-indent *disponible avec* OPAM

Par ailleurs, nous insistons pour que vous preniez l'habitude dès a présent de concevoir l'intégralité de votre code en anglais, que ce soit dans le nom des fichiers, le nom des variables et des procédure, les commentaires, etc.

Évaluation

Le sujet de projet tel que décrit dans ce document consiste en le *minimum nécessaire* pour obtenir une note passable. Il vous est fortement conseillé de réaliser des extensions aux programmes demandés. Le sujet en donne quelques exemples, mais vous êtes libres d'en choisir d'autres. Prenez cependant garde à ne pas être trop ambitieux ! Une extension modeste mais bien réalisée et fonctionnant correctement sera plus valorisée qu'une qui serait plus complexe mais implémentée de façon brouillonne et imparfaite. (Bien sûr, que ce conseil ne soit pas non plus pris comme un encouragement à ne pas faire de votre mieux !)

Assistance

Martin Quinson et Luc Bougé sont à votre disposition pour répondre à toutes vos questions. Notez que nous ne traitons que les questions posées par l'interface Piazza https://piazza.com/ens-rennes.fr/fall2015/programmationl3ri/.

1. Pensez par exemple à utiliser un correcteur orthographique tel qu'*aspell* si vous rédigez votre rapport avec LaTeX.

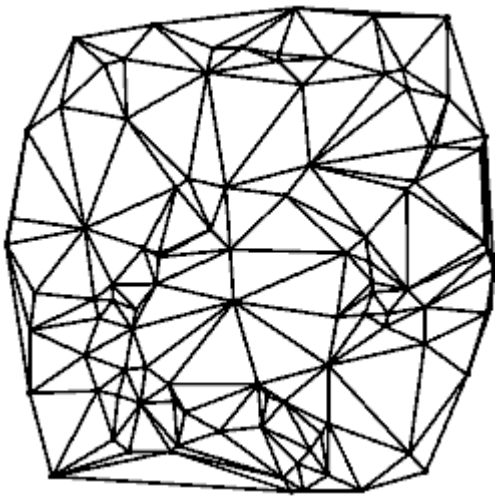

FIGURE 1 – Exemple de triangulation de Delaunay

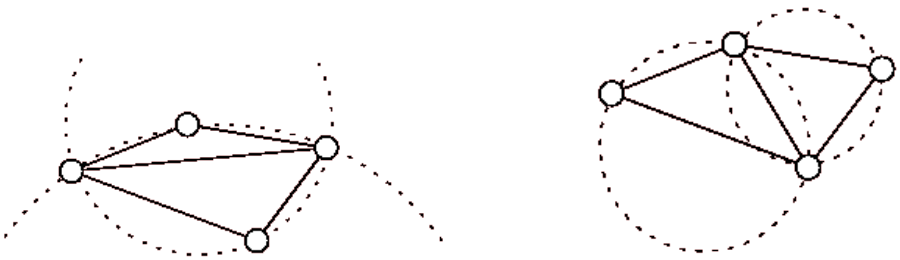

FIGURE 2 – Exemples de triangulations

1 La triangulation de Delaunay

Étant donné un ensemble de points du plan, on veut réaliser un maillage de ce nuage de points par un ensemble de triangles les plus réguliers possibles. Par réguliers, on entend *les plus équilatéraux possibles*, ce qui se traduit par la propriété suivante : l'intérieur du cercle circonscrit de chaque triangle ne contient aucun point du nuages à mailler. Une triangulation qui respecte ce critère est une triangulation de Delaunay. Un exemple de triangulation de Delaunay est donné en Figure 1. La Figure 2 présente une triangulation qui n'est pas correcte (à gauche) et une qui l'est (à droite).

2 Algorithme

Il existe de nombreux algorithmes pour résoudre ce problème. Celui auquel nous allons nous consacrer est un algorithme incrémental non optimal. L'idée est la suivante : on génère dans un premier temps un ensemble aléatoire de points, qui formeront le nuage à mailler. On initialise l'algorithme par un maillage trivial de la fenêtre d'affichage (Figure 3), les points

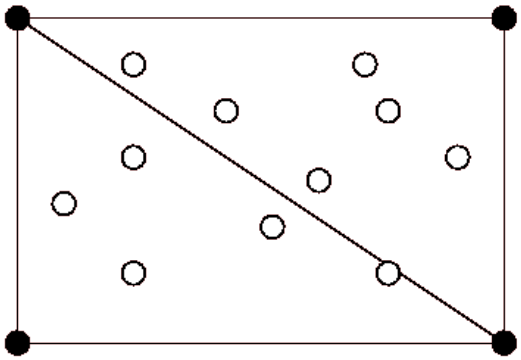

FIGURE 3 – Maillage initial

FIGURE 4 – Ajout d'un point au maillage

blancs sont les points du nuage et les points noirs correspondent aux points extrémaux visibles à l'écran. En d'autres termes, on ajoute les points extrémaux à l'ensemble qu'on veut trianguler, et on les utilise comme points de départ de la triangulation.

À chaque étape de l'algorithme on ajoute un nouveau point du nuage au maillage. Ce nouveau point peut appartenir aux cercles circonscrits à certains triangles du maillage. Il faut, d'une part, enlever ces triangles du maillage (Figure 4, gauche) et, d'autre part, y ajouter le maillage constitué du nouveau point et des points des triangles enlevés (Figure 4, droite).

Il y a donc deux problèmes à résoudre :

1. à partir d'un ensemble de triangles et d'un point, trouver les triangles dont le cercle circonscrit contient le point ;
2. à partir d'un ensemble de triangles à retirer et d'un point, calculer les triangles à ajouter.

Pour résoudre le premier problème on remarquera que le point D est à l'intérieur du cercle circonscrit au triangle (A, B, C) (donné dans le sens direct) si et seulement si le déter-

minant de la matrice :

$$\begin{pmatrix} A_x & A_y & A_x^2 + A_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ C_x & C_y & C_x^2 + C_y^2 & 1 \\ D_x & D_y & D_x^2 + D_y^2 & 1 \end{pmatrix}$$

est strictement positif. Si (A, B, C) est donné dans le sens indirect le déterminant doit être strictement négatif. Pour savoir si (A, B, C) est donné dans le sens direct il suffit de vérifier que le déterminant de la matrice suivante est positif.

$$\begin{pmatrix} B_x - A_x & C_x - A_x \\ B_y - A_y & C_y - A_y \end{pmatrix}$$

Pour résoudre le second problème on remarquera que les triangles à retirer forment une zone convexe du plan. On cherche à calculer la frontière de cette zone. Il suffit pour cela de remarquer qu'une arête appartient à la frontière si et seulement si elle n'apparaît qu'une seule fois dans l'ensemble des arêtes des triangles à retirer. Pour ajouter le point D il suffit donc, pour chacune de ces arêtes (A, B) d'ajouter un triangle (A, B, D).

3 Structures de données et fonctions à implémenter

Dans un premier temps on se limite aux structures permanentes : pas de références, pas de champs mutables, etc. Les types et les fonctions proposés ici sont *obligatoires* lors de votre première implémentation de l'algorithme.

3.1 Types

```
type point = {x: float; y: float}
type triangle = {p1: point; p2: point; p3: point}
type point_set = point list
type triangle_set = triangle list
```

3.2 Fonctions

— val random: **int** −> **int** −> **int** −> point list :
l'appel random nb max_x max_y génère nb points situés entre $(0, 0)$ et $(\text{max\_x}, \text{max\_y})$.

— val ccw: point −> point −> point −> **bool** :
l'appel ccw a b c retourne **true** si a, b et c sont donnés dans le sens direct.

— val in_circle: triangle −> point −> **bool** :
l'appel in_cricle triangle point retourne **true** si le point point est à l'intérieur du cercle circonscrit au triangle triangle.

— val border: triangle_set −> (point ∗ point) list :
border triangles retourne une liste de couples (p, q) tels que les points p et q sont deux points consécutifs de la frontière de la zone convexe définie par l'ensemble triangles. Il faudra faire attention car les arêtes (p, q) et (q, p) sont identiques.

— val add_point: triangle_set −> point −> triangle_set :
l'appel add_point triangles point ajoute le point point à la triangulation courante triangles en utilisant l'algorithme proposé plus haut.
— val delaunay: point_set −> **int** −> **int** −> triangle_set :
l'appel delaunay points max_x max_y construit la triangulation de Delaunay du nuage de points points en initialisant l'algorithme avec un maillage de l'espace de travail compris entre $(0, 0)$ et $(\mathsf{max\_x}, \mathsf{max\_y})$.
— val draw_points: point_set −> unit :
l'appel draw_points points affiche à l'écran le nuage de points points.
— val draw_triangles: triangle_set −> unit :
l'appel draw_triangles triangles affiche à l'écran la triangulation triangles.
— val delaunay_step_by_step: point_set −> unit :
l'appel delaunay_step_by_step points construit en affichant chaque étape la triangulation de delaunay du nuage de points points.

Pour la suite vous devrez toujours respecter ces signatures. Les seules choses qui peuvent être modifiées sont les types présentés plus haut. Le but est que les travaux de tous les groupes puissent être combinés en un seul gros programme.

4 Quelques extensions possibles

Comme pour le premier projet, une extension ne sera corrigée et notée que si tout ce qui précède est terminé de manière satisfaisante. Dans l'idéal, cherchez à implémenter des extensions différentes pour chaque groupe. Voici quelques suggestions d'extensions, mais ce n'est pas du tout limitatif, bien au contraire, ô futurs chercheurs !

Triangulation sans points extrémaux. On peut souhaiter trianguler le nuage de point tel qu'il est généré, sans avoir à inclure les points extrémaux de la fenêtre dans la triangulation. Une façon de faire est de calculer l'enveloppe convexe du nuage, puis d'exécuter le même algorithme que précédemment mais en partant cette fois d'une triangulation de l'enveloppe convexe.

Ajout d'une coordonnée aux points. Ceci permet de tracer des approximations de courbes 3D en associant une couleur à chaque triangle selon sa hauteur. On peut aussi travailler sur l'illumination d'un objet en 3D, la couleur d'un triangle est alors fonction de son orientation par rapport à la source lumineuse. Enfin on peut afficher une vue 3D en gérant la position et l'orientation d'une caméra (avec gestion intelligente de l'ordre d'affichage des triangles).

Ajout d'un état aux triangles ou aux points. Ceci permet de construire un automate cellulaire sur une topologie un peu tordue.

Triangulation dynamique. On a alors un ou plusieurs points mobiles.

Amélioration de la complexité. En utilisant des structures de données plus complexes (attention, les fonctions doivent garder la signature proposée plus haut). On tracera des courbes pour évaluer expérimentalement la complexité et vérifier si elle est vraiment meilleure que précédemment.