Enveloppe convexe et triangulation

— TD —

1° Donnez une suite de points du plan pour laquelle l'algorithme de Graham ne donne pas l'enveloppe convexe des points de cette suite.

2° Montrez que le calcul de l'enveloppe convexe d'un ensemble de n points du plan est un problème demandant un temps de calcul en $\Omega\left(n \log n\right)$

3° Montrer que l'algorithme de triangulation incrémental nécessite également un temps en $\Omega\left(n \log n\right)$

— TP —

Les points du plan sont repérés par leurs coordonnées supposées entières et appartenant à $[0, x_{max}] \times [0, y_{max}]$. On pourra, par exemple, fixer $x_{max} = 1024$ et $y_{max} = 1024$.

4° Définissez une structure `Point` pour stocker un point sous forme de ses coordonnées.

5° Écrire une fonction `randomizedPoints` qui génère un ensemble de n points du plan uniformément dans un disque centré en le centre de la zone d'affichage (par exemple centre $(511, 511)$ et rayon 512). Il suffit pour cela de tirer un nombre au hasard dans un angle $[0, 360[$ et un rayon $[0, 511]$. Attention cela ne garantit pas un tirage totalement uniforme (cf `http://www.afapl.asso.fr/Tiralea.htm`) mais on s'en contentera.

6° Écrire une fonction `concave` qui prend en entrée trois points (r, s, t) du plan et renvoie `True` si t est strictement à droite de (rs) orientée de r à s et `False` sinon.

7° Implémentez l'algorithme de Graham

8° Générez un ensemble de points aléatoirement à l'aide de la fonction `randomizedPoints`, et calculez l'enveloppe convexe de ces points.

9° Implémentez une sortie visible de l'algorithme de Graham [1]

10° Implémentez une fonction de tri lexicographique d'un ensemble de n points (vous pouvez utilisez la fonction `qsort` pour vous simplifier le travail).

11° Définissez une structure `Triangle` et un moyen de stocker un ensemble de `Triangle`.

12° Les points étant supposé triés, une triangulation $\mathcal{T}_i$ ayant été déjà calculé, implémentez une fonction permettant de tester si une arête de $\mathcal{T}_i$ est visible d'un point $p_j, j > i$.

13° Implémentez l'algorithme incrémental de triangulation

14° Générez un ensemble de points aléatoirement à l'aide de la fonction `randomizedPoints`, triangulez ces points, et générez un fichier au format `.PLY` [2] afin de les visualiser dans un logiciel approprié, par exemple Meshlab (`http://www.meshlab.net/`).

1. Vous pouvez utiliser le format proposé pour la triangulation pour visualiser le résultat de l'algorithme de Graham
2. format `.PLY` : `http://paulbourke.net/dataformats/ply/` ou `https://en.wikipedia.org/wiki/PLY_(file_format)`