- TP 3 : Triangulations d'un ensemble de points -

Le but de ce TP est de calculer la triangulation d'un ensemble de points du plan, d'abord de façon incrémentale, puis d'améliorer cette triangulation afin d'obtenir une triangulation de Delaunay.

Le langage à utiliser est laissé libre. Toutefois, des primitives d'affichage et de tri ainsi que des trames de programme sont fournies en C++ à l'adresse :

http ://www.lirmm.fr/∼bessy/AlgoGeo/accueil.html

- Triangulation incrémentale -

- Exercice 1 -

Le but de l'exercice est d'implémenter le calcul d'une triangulation d'un ensemble de points du plan par l'algorithme incrémental vu en cours. Le résultat à obtenir est illustré Figure 1.

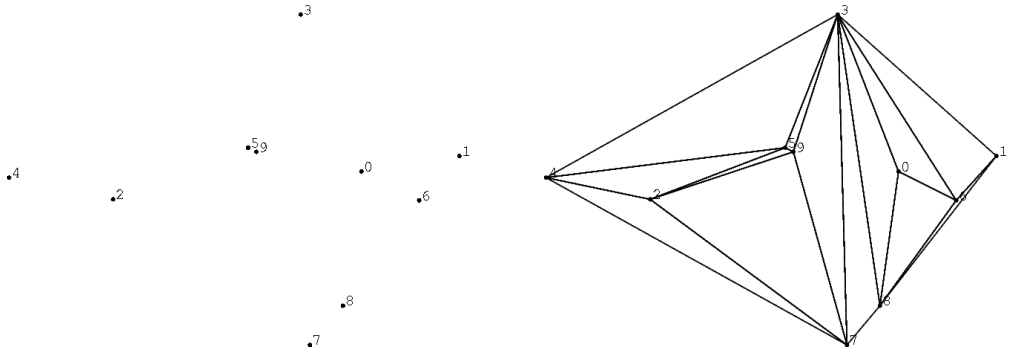

FIG. 1 – Exemple : Un ensemble P de points du plan, puis le même ensemble triangulé.

Dans le fichier *Triangulation.cc*, se trouvent implémentées notamment les fonctions et structures suivantes :

- **typedef struct** { **int** a **; int** b **; int** c **; } triangle ;** : indices des sommets d'un triangle.
- **void PointAuHasard(int** n**,int** $sommet$[][2]**)** : génération de sommets aléatoirement dans le plan.
- **void AffichageTriangulation(int** n**, int** $sommet$[][2]**, int** t**, triangle** T[]**)** : affichage des sommets et de la triangulation dont les triangles sont contenus dans T.
- **void TriLexicographique(int** n**, int** $sommet$[][2]**, int** t**, int** tri[]**)** : tri lexicographique des sommets, leurs indices, triés sont récupérés dans tri.

Leur usage est détaillé dans le fichier *Triangulation.cc*.
Il reste à compléter la fonction **TriangulIncrementale**, plus précisément :

1. Calcul du premier triangle et initialisation de l'enveloppe convexe.
2. Puis pour l'insertion d'un nouveau point d'indice $tri[i]$:
   (a) Calcul de $khaut$, l'indice du sommet de l'enveloppe convexe en cours le plus loin dans le sens direct visible depuis le sommet d'indice $tri[i]$.
   (b) Calcul de $kbas$, l'indice du sommet de l'enveloppe convexe en cours le plus loin dans le sens indirect visible depuis le sommet d'indice $tri[i]$.
   (c) La mise-à-jour de l'enveloppe convexe.

- Triangulation de Delaunay -

- Exercice 2 -

Le but de l'exercice est d'implémenter le calcul d'une triangulation de Delaunay d'un ensemble de points à partir d'une triangulation quelconque de ces points (par exemple, celle calculée dans l'exercice 1). Le résultat à obtenir est illustré Figure 2. Dans le fichier *Delaunay.cc*, se

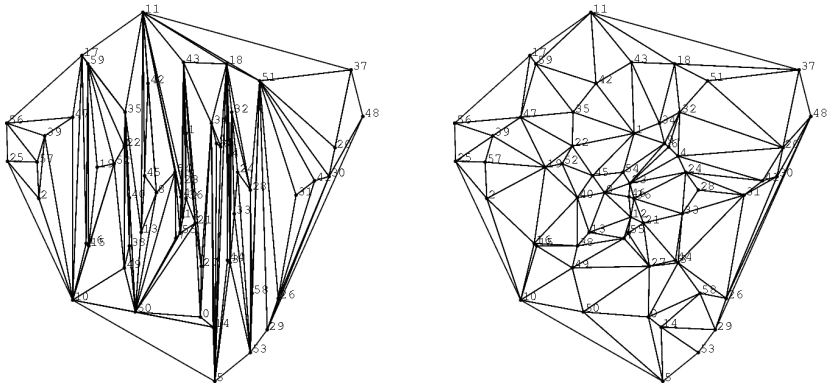

FIG. 2 – Exemple : Un triangulation d'un ensemble de points du plan, puis sa triangulation de Delaunay.

trouve une implémentation en C++ à compléter. Plus précisément :

1. Compléter la partie intitulée *Repérer un point par rapport au cercle circonscrit à un triangle*. Testez votre code avec la fonction **AffichageTestCercleCirconscrit**.
2. Compléter le cœur de l'algorithme, c'est-à-dire la fonction **Delaunay**. Il est rappelé qu'une triangulation est de Delaunay si, et seulement si, pour chaque triangle t, le cercle circonscrit à t ne contient strictement aucun point des triangles voisins à t. Les fonctions suivantes sont implémentées :
   - **int TroisiemePoint(triangle** s**, triangle** t**)** : retourne l'indice du point du triangle s qui n'est pas dans le triangle t.
   - **void Flip(int** i**, int** j**, int** t**, triangle** $T[]$**, int** $voisin[][3]$**)** : effectue le 'flip' de l'arête commune entre les triangles $T[i]$ et $T[j]$, le tableau $voisin[][3]$ est remis à jour.

   Vous utiliserez aussi la fonction **StrictementDansLeCercleCirconscrit** que vous avez implémenté dans la question 1.
   L'usage de ces fonctions est détaillé dans le fichier *Delaunay.cc*.

- Exercices supplémentaires -

- Exercice 3 - Arbre couvrant euclidien minimum -

Calculer un arbre couvrant euclidien minimum d'un ensemble de points aléatoirement répartis. Faire afficher celui-ci en surimpression sur la triangulation de Delaunay de l'ensemble de points choisis, et vérifier que les arêtes de l'arbre sont des arêtes de la triangulation.

- Exercice 4 - Diagramme de Voronoï -

A l'aide de la triangulation de Delaunay obtenue précédemment, écrire une fonction qui calcule les points du diagramme de Voronoï.
Ecrire une seconde fonction qui finit le calcule et trace effectivement ce diagramme.
Effectuer le calcul du diagramme de Voronoï directement, à l'aide de l'algorithme de S.Fortune vu en cours.