IMAC
INGÉNIEUR
ESIPE

Programmation et Algorithmique II
Projet
Delaunay/Voronoi

Vous arrivez vers la fin des tps d'Algorithme et Programmation, vous avez obtenu les notions de bases des algorithmes avancés. Pour ce fin de semestre, vous allez implémenter un autre algorithme qui permet d'une part de faire la triangulation de Delaunay et d'autre part faire un diagramme de Voronoï.

Triangulation de Delaunay

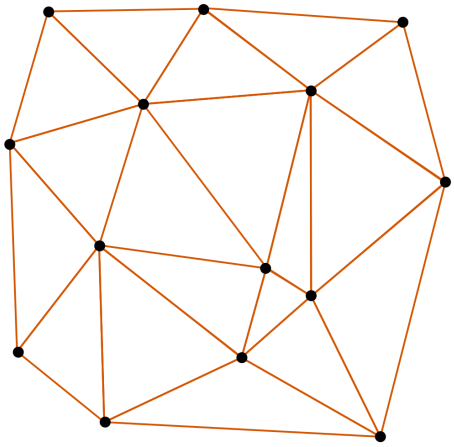
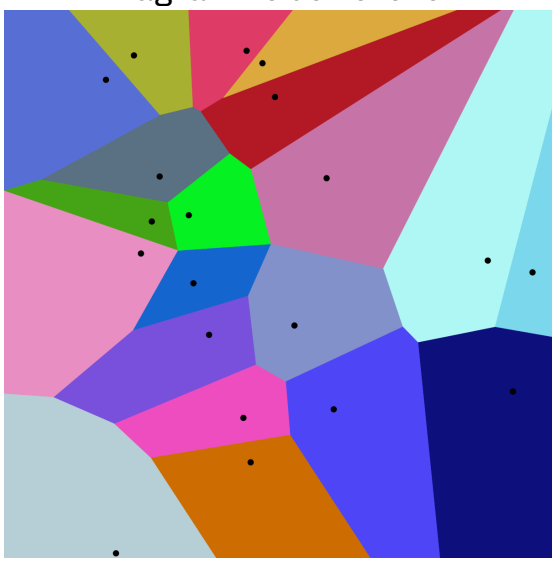


Diagramme de Voronoi



Évaluateurs :
Steeve Vincent : steeve.v91@gmail.com
Enguerrand De Smet : desmet.enguerrand@gmail.com

Modalités de rendu

- **Nombre de personnes par projet :** 2
- **Livrable :**
  - Les sources C/C++ avec un système de compilation permettant de compiler sur un système (Linux ou Windows) avec l'environnement suivant :
    - Package : package SDL 1 et 2, SDL_image, SDL_mixer, SDL_TTF
    - Compilateur : gcc, g++ (version C++≤ 17)
    - Utilitaire de compilation : make, ninja, cmake
  - un fichier texte pour la rédaction demandée dans la section 4
- **Date de rendu :** Lundi 05 Juin 2022 avant 1h00, **Tout retard sera pénalisé par des malus**

1 Géométrie

1.1 Voronoï

Le diagramme de Voronoi décrit les régions de proximité pour un ensemble de point.

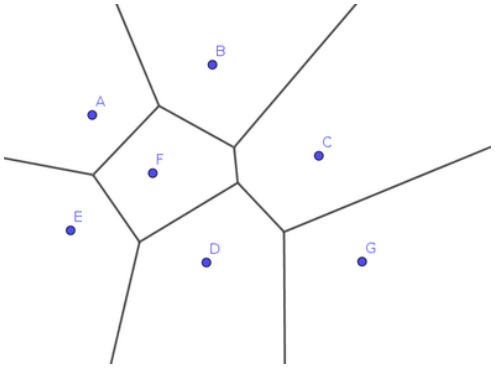


Figure 1 – Diagramme de Voronoï

Chaque point produit un polygon de Voronoï, il s'agit d'un polygon qui contient tous les points du repère 2D qui est plus proche de ce point que tous les autres.

1.2 Delaunay

La triangulation de delaunay dont tous les cercles circonscrits ne contiennent que les points de leur triangle respectif.
Le cercle circonscrit d'un triangle et le cercle qui passe par les trois points du triangle.

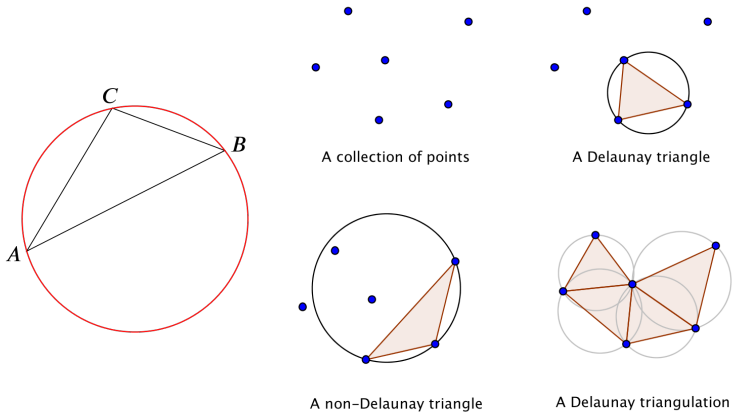


Figure 2 – Cercle circonscrit du triangle ABC et Triangulation de Delaunay

Cela a pour conséquence de former un maillage de triangles sans superposition mais plus encore il existe une corrélation entre la triangulation de Delaunay et le diagramme de Voronoï.

1.3 Graphe dual

Le graphe dual d'un ensemble de polygon P forme un graphe G dont chaque point est à l'intérieur d'un polygon de $P1$ et chaque arête croise un segment de $P1$. Aussi, chaque polygon de $P1$ contient un unique point de G et chacun de ses segments est croisé par une unique arête de G.

Nous, nous allons nous intéresser à un graphe dual particulier qui produit des polygons à partir d'un ensemble de triangles. Chaque point de ce graphe dual est l'intersection des médiatrices de chaque triangle de l'ensemble de base. Ces points sont par ailleurs les centres des cercles circonscrit des triangles en questions.
Il se trouve que ce genre de graph dual, appliqué à une triangulation de Delaunay, forme un diagramme de Voronoï.

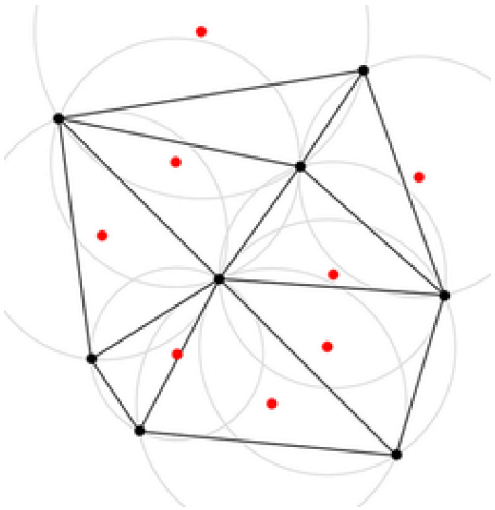


Figure 3 – Triangulation de Delaunay avec les cercles circonscrits

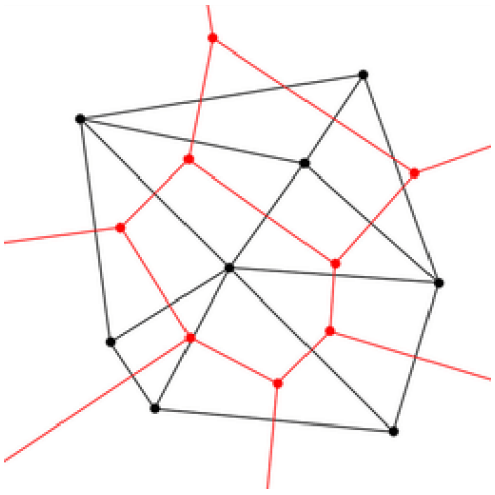


Figure 4 – Graphe dual de la triangulation de delaunay

2 Programme

A partir d'un programme C++, créer le diagramme voronoi d'un ensemble de points défini par l'utilisateur. Vous pouvez utiliser le template de code SDL fourni qui permet de créer un ensemble de points à partir de cliques utilisateur.

Milestone 0 (Prendre en main le code SDL)

- Faire triangles à partir des points générés.

Milestone 1 (Delaunay)

- Faire une triangulation de Delaunay, pour ce faire, utiliser l'algorithme suivant :

Algorithme 1 : A Triangulation Algorithm Written by Paul Bourke

```
Données : Liste de points 2D
Résultat : Liste de triangles

Trier les points selon x

Vider la liste existante de triangles
Créer un très grand triangle (-1000, -1000), (500, 3000), (1500, -1000)
    Le rajouter à la liste de triangles déjà créés

pour chaque point P du repère faire
    créer une liste de segments LS
    pour chaque triangle T déjà créé faire
        si le cercle circonscrit contient le point P alors
            Récupérer les différents segments de ce triangles dans LS
            Enlever le triangke T de la liste
    pour chaque segment S de la liste LS faire
        si un segment est un doublon d'un autre* alors
            le virer ;
    pour Pour chaque segment S de la liste LS faire
        créer un nouveau triangle composé du segment S et du point P ;
```

* : $(segment1.p1 == segment2.p2)\&\&(segment1.p2 == segment2.p1)$
Le principe de l'algorithme de créer des triangles itérativement en ajoutant un nouveau point à chaque fois dans le processus, ce nouveau point va rendre caduque les triangles actuels dont le cercle circonscrit rendre en collision et en recrée de nouveaux dont les cercles circonscrits sont à nouveau bons.

Milestone 2 (Voronoi)

- Déterminer les centres des cercles circonscrits et connecter les points des triangles adjacents pour en faire des polygons

Fonctionnalités additionnelles

- Gérer des vrais polygons et pas juste des segments
- Mettez des couleurs, faites un truc joli quoi.

3 Développement

3.1 SDL

Pour faire ce projet, il est recommandé d'utiliser la base fourni par les évaluateurs. Cette base utilise la librairie [1] SDL2 qui permet de gérer des fenêtres et leurs contenus indépendamment de la plateforme. Dans cette base vous allez retrouver aussi un main qui gère l'affichage de la fenêtre ainsi que la ***main loop***. Cette main loop permet de gérer les événements tels que les cliques utilisateur. Vous pouvez vous contenter d'implémenter la fonction ***construitVoronoi***, elle prend en paramètre la structure principale qui est ***Application***, contenant les informations nécessaire au déroulement du programme.

3.1.1 Application

La structure **Application** contient quelques informations pour le fonctionnement du programme. il contient notamment, la résolution de la fenetre, la liste des points ajoutés par l'utilisateur, la liste des triangles et des segments à afficher.
Pour savoir comment manipuler les points, les triangles et les segments, vous pouvez vous référer à la documentation des vector.

3.1.2 Structures

Vous avez certaines structures à votre disposition, notamment :
Coords :

- coordonnée X : entier
- coordonnée Y : entier

Segment :

- point 1 : Coords
- point 2 : Coords

Triangle :

1. Une librairie est un ensemble de code utilitaire pré-compilé

- point 1 : Coords
- point 2 : Coords
- point 3 : Coords

Vous pouvez modifier ces structures pour les adapter à votre code.

3.2 CircumCircle

La fonction ***CircumCircle*** vous permet de déterminer d'une part si un point est dans un cercle circonscrit et d'autre part les caractéristiques de ce cercle (notamment centre et rayon). Vous devez passer à cette fonction les coordonnées du point à tester, ceux des points du triangle ainsi que des pointeurs pour récupérer les infos du cercle.

3.2.1 Standard C++ container

Pour vous faciliter la tache, le programme utilise des containers standard C++, il s'agit de structure spéciales permettant de gérer un ensemble d'élément sous forme de tableau, de liste, d'arbre binaire ou d'autres systèmes standards.
cppreference:containers standards

3.2.2 C++ sort

Il existe une fonction ***sort*** qui permet de trier des containers. Pour l'exercice, vous pouvez notamment utiliser
$std :: sort(app.points.begin(), app.points.end(), compareCoords);$
compareCoords étant une fonction présent dans main.cpp permettant de déterminer si un point est plus "petit" qu'un autre en se basant d'abord sur la coordonnée X.

4 Rédaction

Pas de rapport pour ce projet, le projet n'est pas suffisamment déclinable pour en faire un rapport intéressant. En revanche, nous vous demandons de rédiger quelques paragraphes sur les containers list, vector, deque et map. Pour chaque container, il faudra décrire grossièrement son fonctionnement ainsi que ses avantages et inconvénients. Vous pouvez illustrer les spécificité à travers des exemples où il est plus pertinent d'utiliser un container plutôt qu'un autre.
(et pas de ChatGPT, sinon je vous goume)
Un fichier texte à la racine du dossier suffira.

5 Notation

La note du projet sera prise en compte dans la note final de la matière Programmation et Algorithmique en plus des TPs. La notation du projet dépend de l'implémentation réussi de l'algorithme et des description des containers, voici une vague idée du barème

0-10 Algorithme défaillant

10-11 Triangulation de Delaunay

12-13 Diagramme de Voronoi

14-15 Propreté du code, Esthétique, Fonctionnalités supplémentaires

5 pts Rédaction sur les containers C++

6 Remarques

— Vous pouvez utiliser un outil de répartition et suivi de tâches, par exemple Trello, Azendoo ou Notion

— N'oubliez pas de tester votre application à chaque spécification implémentée. Il est impensable de tout coder puis de tout vérifier après. Ne visez pas directement votre oeuvre finale, tester les implémentations basiques sur des cas simples.

— Vos chargés de TD et CM sont là pour vous aider. Si vous avez des doutes/des difficultés sur un point, n'attendez pas la soutenance pour nous en parler.