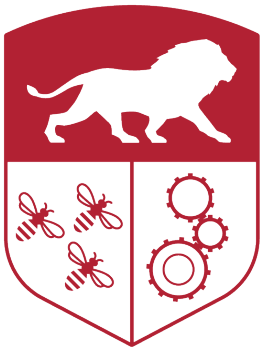


CENTRALE
LYON

Informatique graphique

Élèves :
Paul LACROIX

Enseignants :
Raphaëlle CHAINE

6 avril 2025

Table des matières

1 Introduction

L'objectif de ce rapport est des présenter et d'expliquer le code effectué pour l'algorithme de Delaunay.

Pour les fichiers de test, il sera nécessaire de changer le path des fichiers OFF (qui sont dans le code adapté à ma configuration).

La majorité du code pour l'algorithme de Delaunay se trouve dans le fichier "mesh.cpp".

2 Présentation des défauts du code

Par manque de temps, l'implémentation est naïve sur certain points :

- L'algorithme de Lawson, (fonction "LawsonAlgorithm") qui permet de transformer une triangulation naïve 2D en triangulation de Delaunay, regarde si la triangulation est de Delaunay. Si non, elle parcourt toutes les demi-arrêtes et flip toutes celles qui ne sont pas localement de Delaunay.
- Dans la fonction "initFromPoints", après avoir insérer un point dans la triangulation, parcours toutes les demi-arrêtes et flip celles qui ne sont pas localement de Delaunay. Ainsi, l'algorithme est lent pour initialiser des maillages contenant beaucoup de points.
- L'insertion d'un point à l'extérieur du maillage n'est pas totalement fonctionnelle. Pour pallier à ce problème, lorsque l'on souhaite créer un maillage à partir d'un nuage de points, la fonction "initMeshBoundingBox" est appelée. Elle permet de calculer les minimums et maximums en x et y du nuage de points pour ensuite créer un Mesh initial de 4 points qui contiendra l'ensemble du futur maillage. Ainsi, tous les points ajoutés seront à l'intérieur du maillage.

3 Explication de Delaunay

L'algorithme de Delaunay parcours toutes les arrêtes et flip celles qui ne sont pas localement de Delaunay. Le flip prend en paramêtre 2 faces adjacentes, dont on veut flipper l'arrête qui les sépare. Elle se repose (comme d'autre fonctions), sur la fonction "getPointsFrom2Faces".

Cette dernière prend en argument 2 faces. Si c'est deux fois la même face, où qu'elles ne sont pas adjacentes, cela déclenche une erreur. Si l'une des faces est la face infinie, elle retourne que des -1. Sinon, elle permet d'obtenir les 2 points communs entre les deux faces (donc l'arrête), ainsi que les deux points appartenant à chacune des face. Les points seront toujours dans le même ordre, c'est à dire que le premier, le deuxième et le troisième points sont les points de la première face données à la fonction, dans le sens trigonométrique, et le deuxième, premier et troisième points forment dans cette ordre la deuxième face dans le sens trigonométrique.

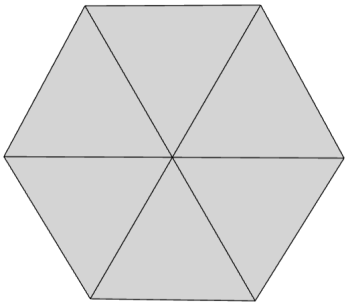
On peut ainsi connaître de façon sûre l'orientation des points, et permet ensuite beaucoup plus facilement de faire les modifications qui permettent de garde l'intégrité du maillage, c'est-à-dire mettre à jour les faces adjacentes des différentes faces.

4 Insertion d'un point dans le maillage

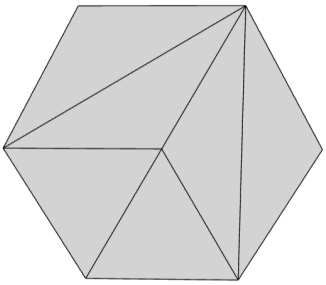
L'insertion d'un point dans le maillage se fait avec la fonction "insertion2D". Parcours d'abord toutes les faces pour voir si le point est dedans. Si oui, alors la face est découpé en 3 nouvelles faces avec la fonction "splitFace". Si le points est dans aucune des faces (donc à l'extérieur du maillage), alors on appelle la fonction "addPointOutsideMesh".

Dans le cas où le point est à l'extérieur du maillage, on parcours toutes les arrêtes, pour calculer la distance entre son milieu est le nouveau point, et trouver celle la plus proche. Ensuite, on créer une nouvelle face qui contient les deux points de l'arrête et le nouveau point. Ensuite, avec la fonction "makeConvexShape", on regarde si on doit créer des faces supplémentaires pour garder la forme convexe du maillage.

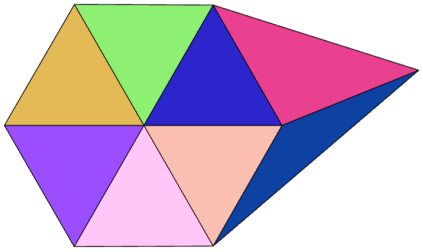
Cependant cette fonction part du principe que les arrêtes adjacentes qui forment le bord du maillage est situé sur une autre face. Ainsi, si dans un maillage, une face possède deux arrêtes sur le bord du maillage, cela fait planter le code (illustration en figure 1). Pour remédier à ce problème, on pourrait créer une classe qui permetterait d'itérer sur le bord du maillage. Ainsi, cela éviterait de parcourir toutes les arrêtes pour trouver la plus proche (en créant une surcharge des opérateurs ++ et - - selon le sens dans lequel on veut parcourir le bord), et aussi plus facilement garder la forme convexe du maillage.



(a) Chaque arrête sur le bord appartiennent à des faces différentes



(b) La face à droite possède deux arrêtes sur le bord



(c) Le point a correctement été inséré de le premier maillage. La même opération de marche pas pour le deuxième maillage

FIGURE 1 – Exemple de limite de l'algorithme d'insertion d'un point en dehors du maillage

5 Résultats

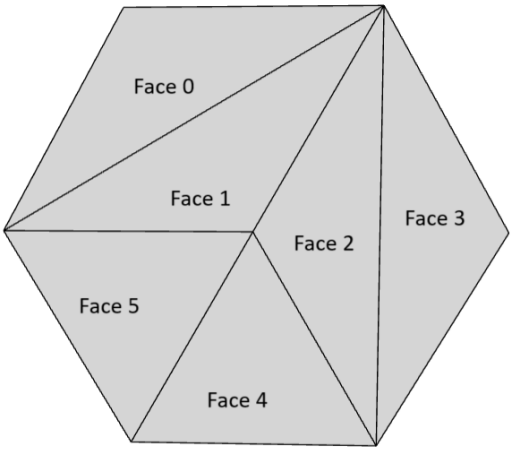


FIGURE 2 – Maillage sur lequel seront fait les opérations

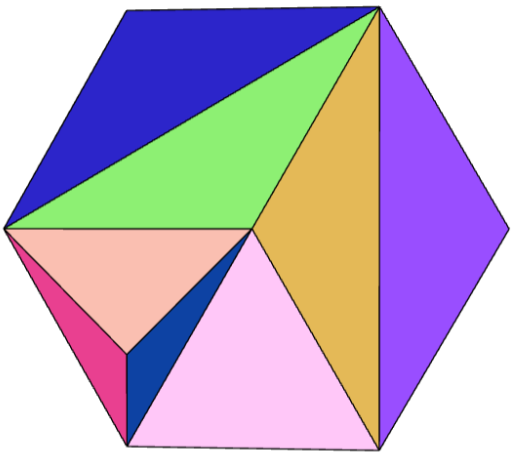


FIGURE 3 – Exemple de split Face (face 5)

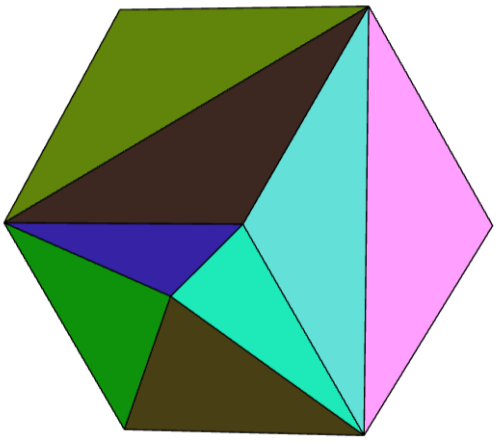


FIGURE 4 – Exemple de split Edge (entre les face 4 et 5)

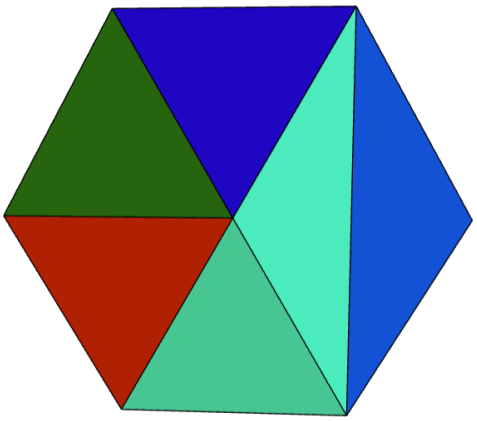


FIGURE 5 – Exemple de flip Edge (entre les faces 0 et 1

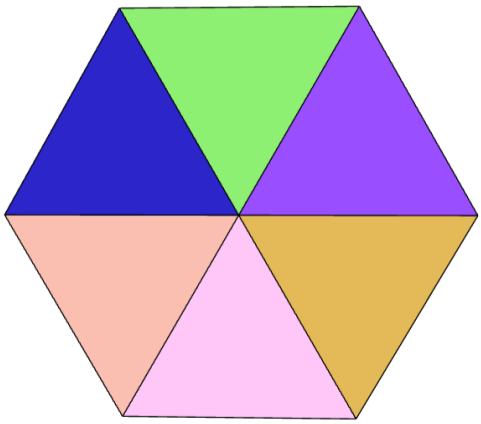


FIGURE 6 – Exemple de delaunay simple

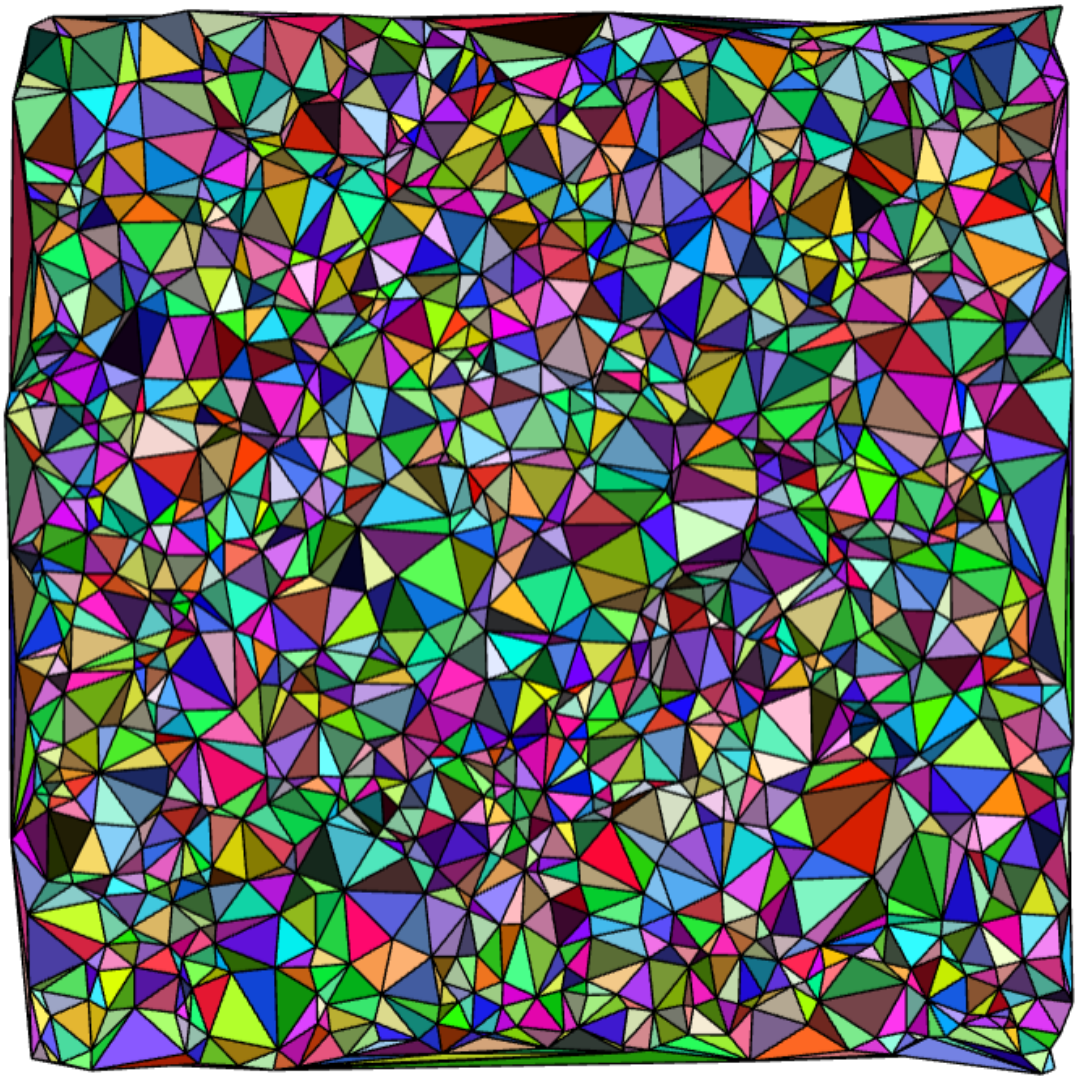


FIGURE 7 – Delaunay sur le fichier "noise_poisson.txt" limité à 1000 points