Quintard Livaï - 11508352
Faurobert Emeric - 11508845

Rapport Projet Géométrie Algorithmique

I) Introduction

Le but de ce projet est de réaliser une application de triangulation en 2D, et ce, de manière incrémentale. Afin d'obtenir un maillage propre, plusieurs algorithmes ont été implémentés. Dans un premier temps on a réalisé une triangulation naïve, ordonnée selon les points entrés. Par la suite on rend notre maillage globalement de Delaunay grâce aux flips de Lawson. Nous avons ensuite amélioré cet algorithme afin de le rendre itératif. Finalement, nous avons réalisé l'extraction de squelette par l'algorithme Crust, en utilisant les centres de Voronoï. L'application est pourvue d'une interface permettant de visualiser les différentes étapes au fur et à mesure des transformations effectuées.

II) Structure initiale

Un nuage de points est représenté dans un fichier comme une liste de coordonnées spatiales. Cette liste est lue et sert de base à la triangulation. Un *point* est composé de ses coordonnées dans le repère. Un *sommet* est défini comme un *point* associé à une *face*. Chaque *face* est composée de trois *sommets* et connaît les trois *faces* voisines. Tous ces éléments (faces & sommets) s'identifient entre eux par leur identifiant (un entier). Différents itérateurs permettent de parcourir les sommets et les faces (dans ce dernier cas il est possible d'effectuer un parcours exhaustif ou par visibilité). Les circulateurs permettent quant à eux de naviguer sur les *faces* ou *sommets* autour de chaque *sommet*.

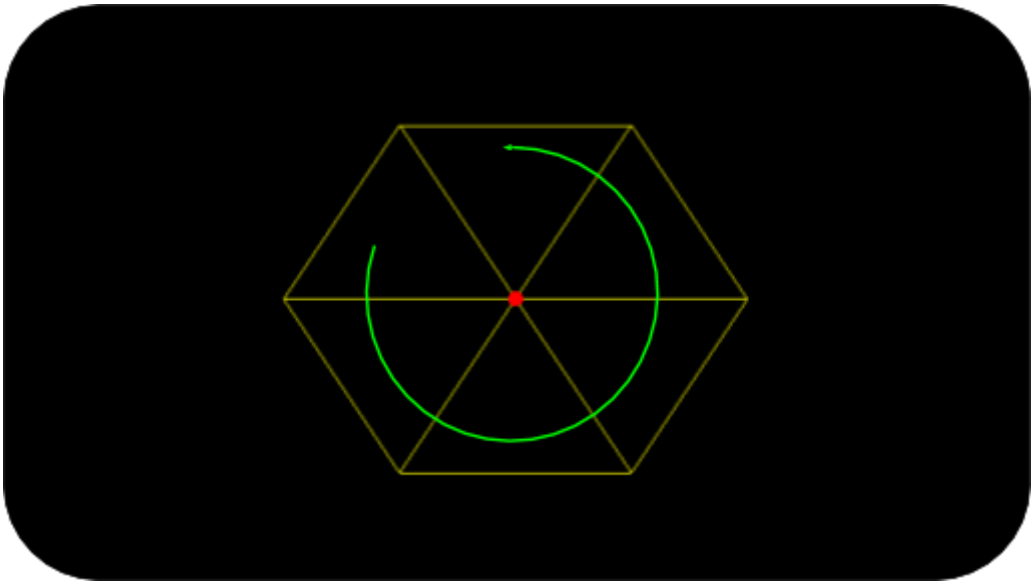

Figure 1 : Circulation autour d'un sommet

Le problème de *faces* sans voisins (en bordure) est résolu par la création d'un point infini auquel on relie ces dernières. Ce point, qui ne fait pas réellement partie du maillage, nous servira par la suite à naviguer autour de notre enveloppe convexe.

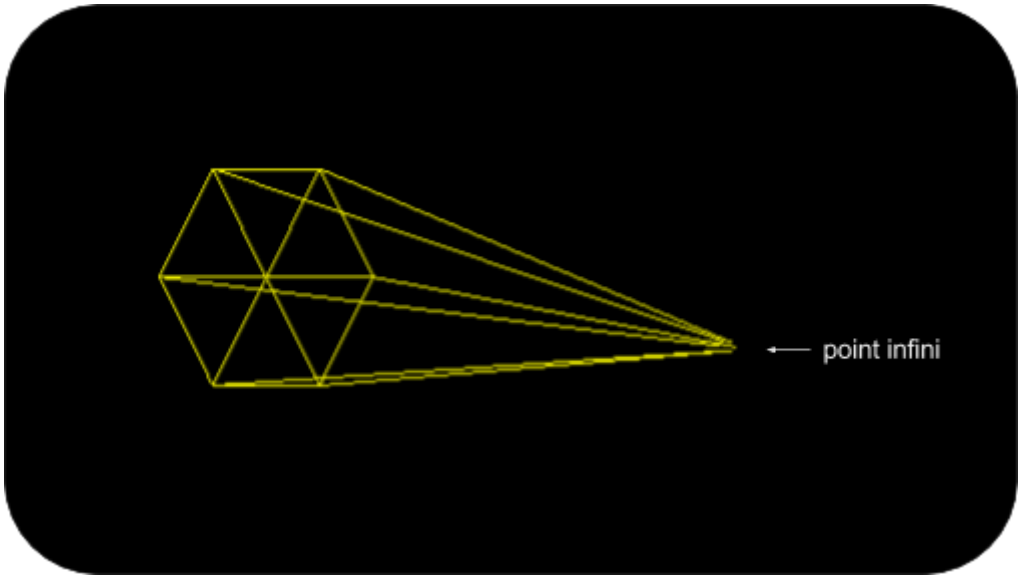

Figure 2 : Représentation du point infini

III) Triangulation naïve

Les trois premiers points lus constituent notre premier triangle. Les suivants seront tous progressivement ajoutés au maillage par la suite.

- Si le point lu est à l'intérieur d'un triangle existant, alors on subdivise ce triangle en 3.

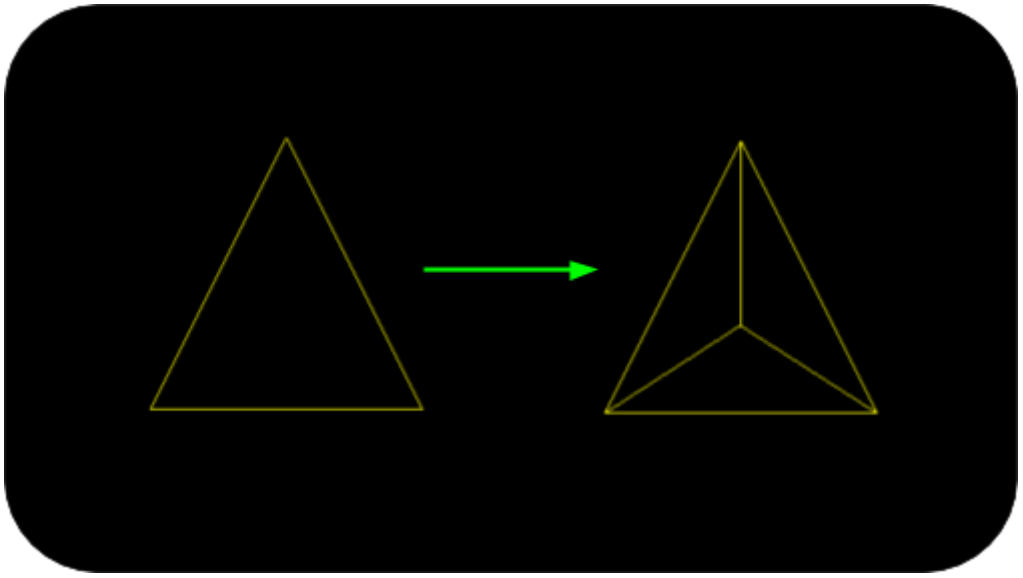

Figure 3 : Ajout d'un point dans un triangle

- Si le point se situe à l'extérieur du maillage, alors on l'y ajoute tout en le reliant à l'enveloppe convexe (bordure). Pour déterminer à quel endroit l'insérer, on parcourt les arêtes de cette enveloppe convexe (en circulant autour du point infini). On relie notre point à la première arête qu'il peut "voir" (càd qu'il n'y a pas d'autre triangle entre l'arête et le point). Si tel est le cas un nouveau triangle est formé avec les points concernés, et le triangle précédemment lié au point infini est subdivisé. Dans tous les cas, les triangles et sommets qui ont été créés ou modifiés voient leurs informations mises à jour (ex : voisins).

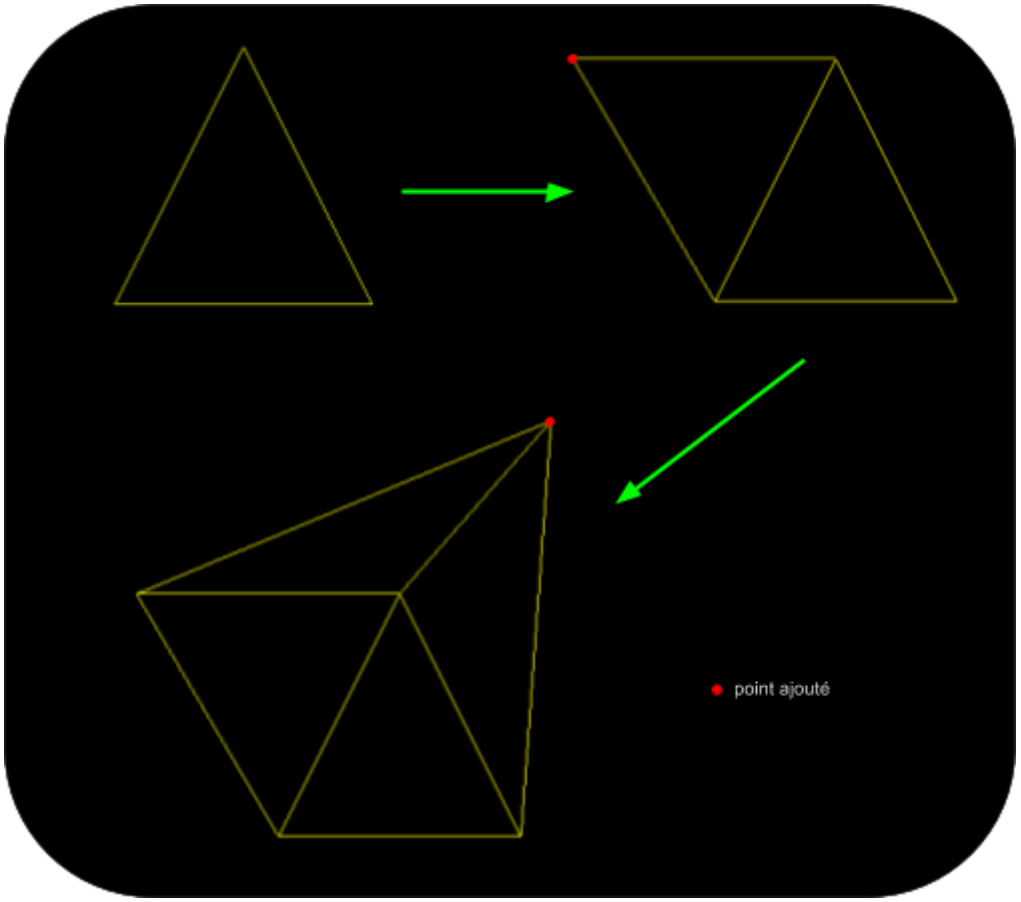

Figure 4 : Ajouts de points à l'extérieur

Cependant, la triangulation naïve est fortement dépendante de l'ordre dans lequel sont insérés les points. Par conséquent, le résultat obtenu peut être extrêmement mauvais.

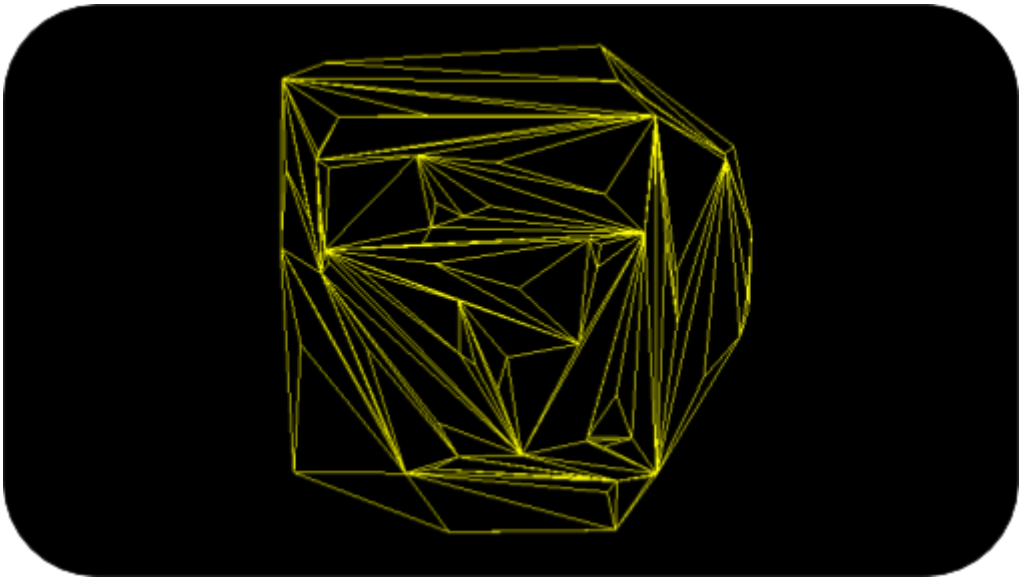

Figure 5 : Triangulation naïve "moche"

IV) Triangulation de Delaunay

Afin de résoudre ce problème, le "mauvais" maillage va être modifié pour en faire une triangulation globalement de Delaunay. Un triangle est dit localement de Delaunay si son cercle circonscrit ne contient aucun des sommets appartenant aux triangles adjacents (à l'exception des sommets communs). Une méthode permettant de rendre tous les triangles localement de Delaunay est le flip d'arête de Lawson. Celle-ci consiste à intervertir l'arête commune entre 2 triangles voisins dont l'un n'était pas localement de Delaunay.

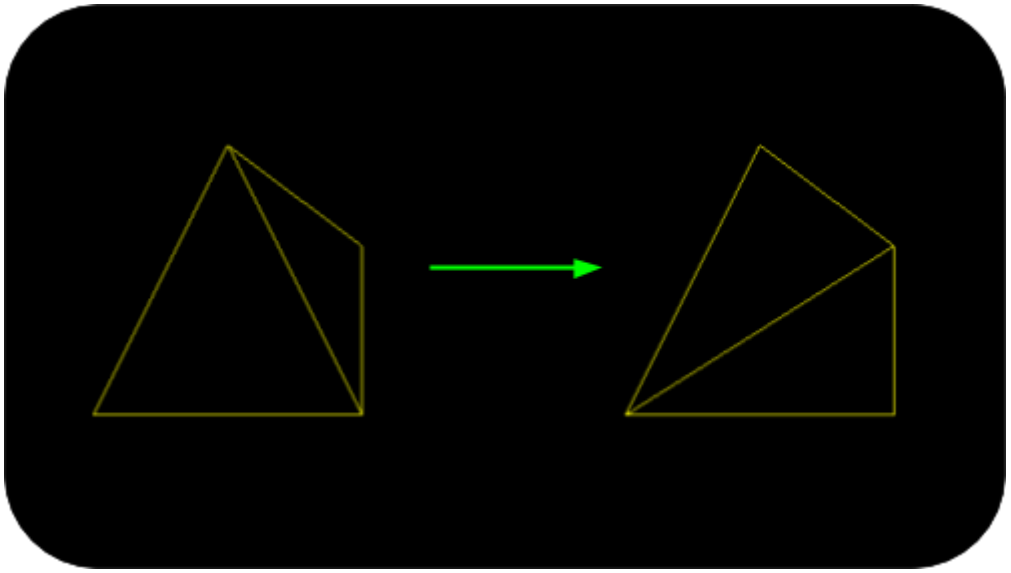

Figure 6 : Flip d'arête

Dans un premier temps, nous avons développé une version naïve de cet algorithme qui parcourait tous les triangles, flippait ceux non localement de Delaunay puis réitérait tant qu'il y avait des modifications possibles. Cependant, une telle implémentation est très coûteuse puisque beaucoup d'opérations sont nécessaires, y compris sur des triangles non concernés par de potentielles modifications.

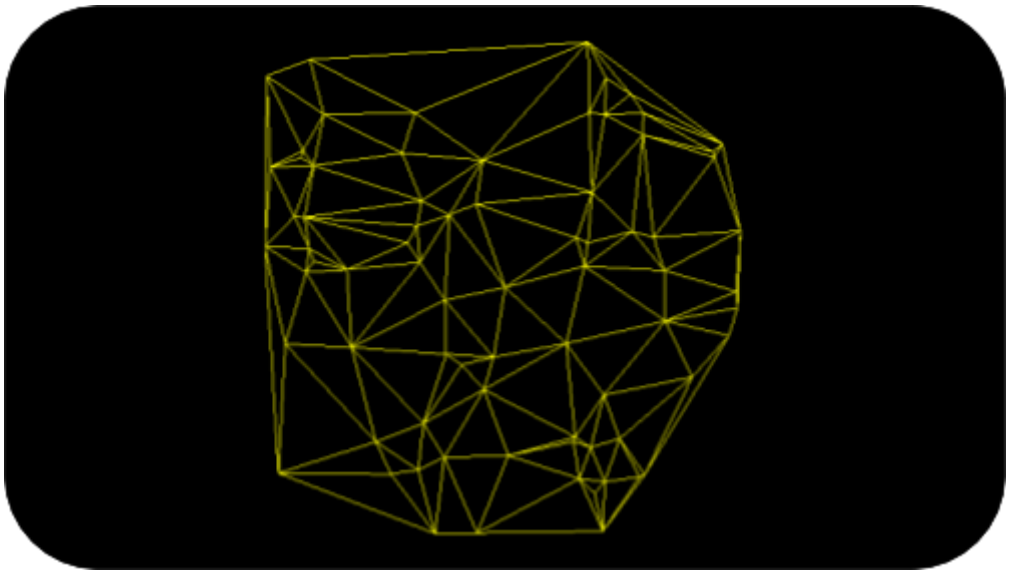

Figure 7 : Triangulation de Delaunay

Afin d'améliorer les performances, une version itérative de cet algorithme a été développée. Cette dernière agit après chaque ajout de point dans une triangulation qui était déjà de Delaunay. Contrairement à la version naïve, celle-ci se contente de parcourir les triangles de proche en proche (propagation) pour effectuer les flips de Lawson. C'est pourquoi elle est beaucoup plus performante, car elle se contente de ne vérifier que les triangles pouvant être potentiellement impactés.

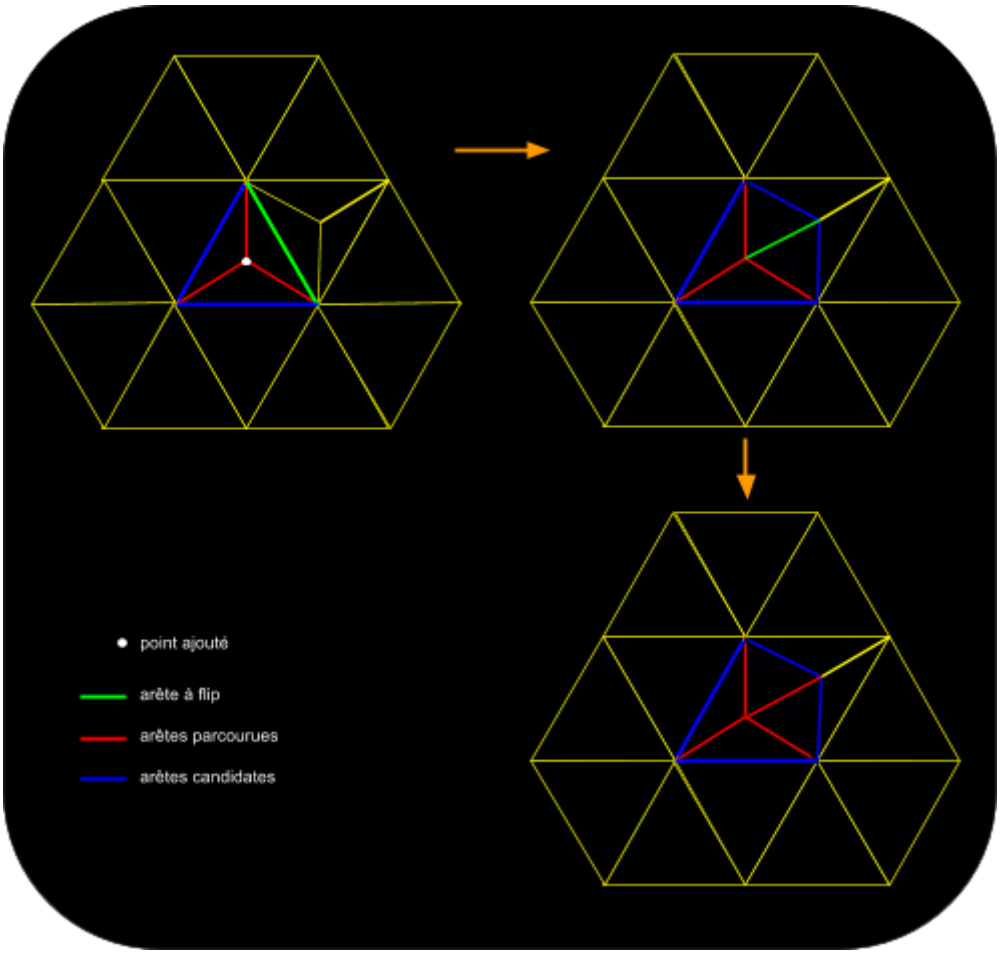

Figure 8 : Delaunay incrémental

V) Crust

Le but de l'algorithme Crust est de trouver le squelette d'une triangulation de Delaunay. Pour ce faire, Crust a besoin des centres des cellules de Voronoï.

A) Voronoï

Dans un premier temps, il faut construire les cellules de voronoï. Ces cellules sont constituées des centres des cercles circonscrits à chacun des triangles. On obtient alors le graphe suivant :

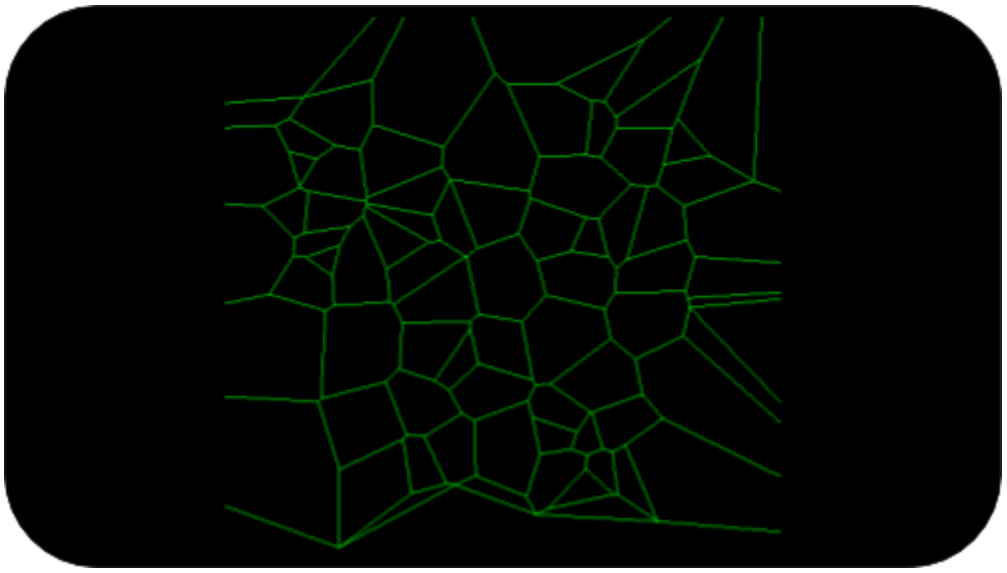

Figure 9 : Cellules de Voronoï

B) Crust

Un nouveau maillage doit être créé. Ce maillage est composé des sommets initiaux ainsi que des centres de voronoï. Après transformation de tous ces points en triangulation de Delaunay, on obtient un nouveau maillage:

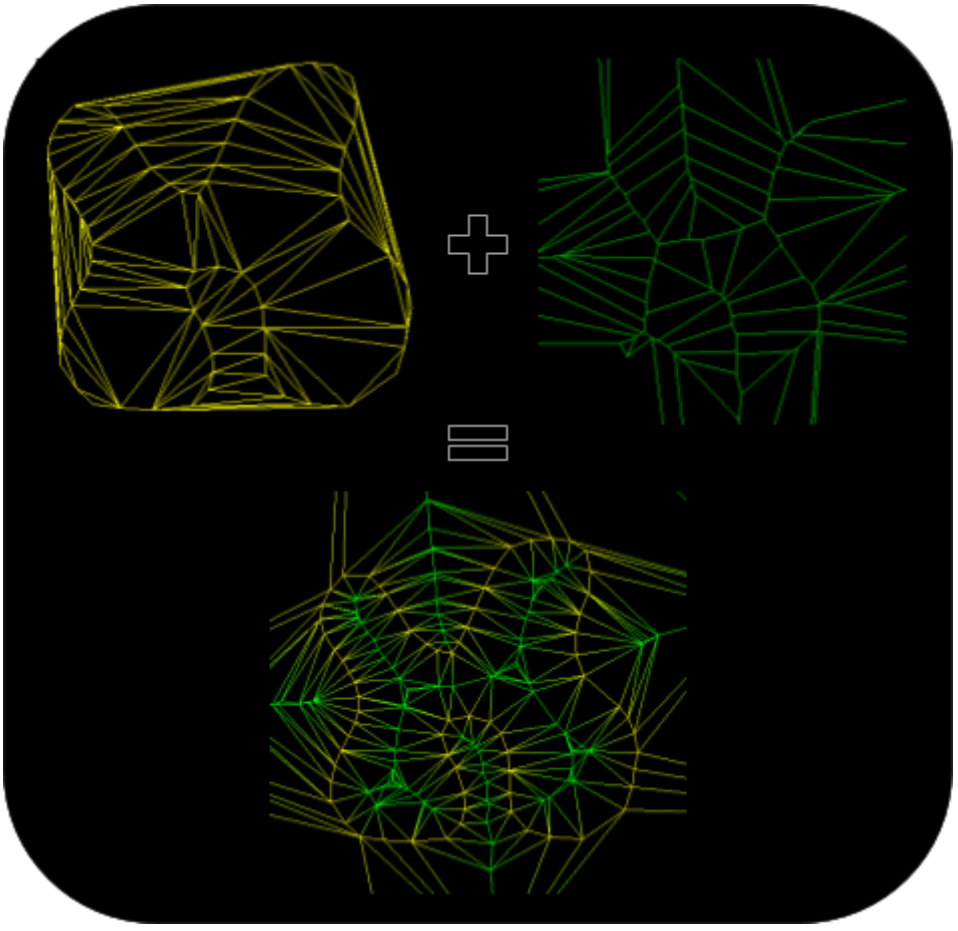

Figure 10 : Maillage Pre-Crust

Il suffit ensuite d'exploiter ce maillage en ne gardant que les arêtes reliant deux sommets du maillage initial. Les arêtes "centre de Voronoï - centre de Voronoï" et "centre de Voronoï - sommet du maillage initial" sont donc ignorées. Ce filtrage permet de récupérer le squelette du maillage initial :

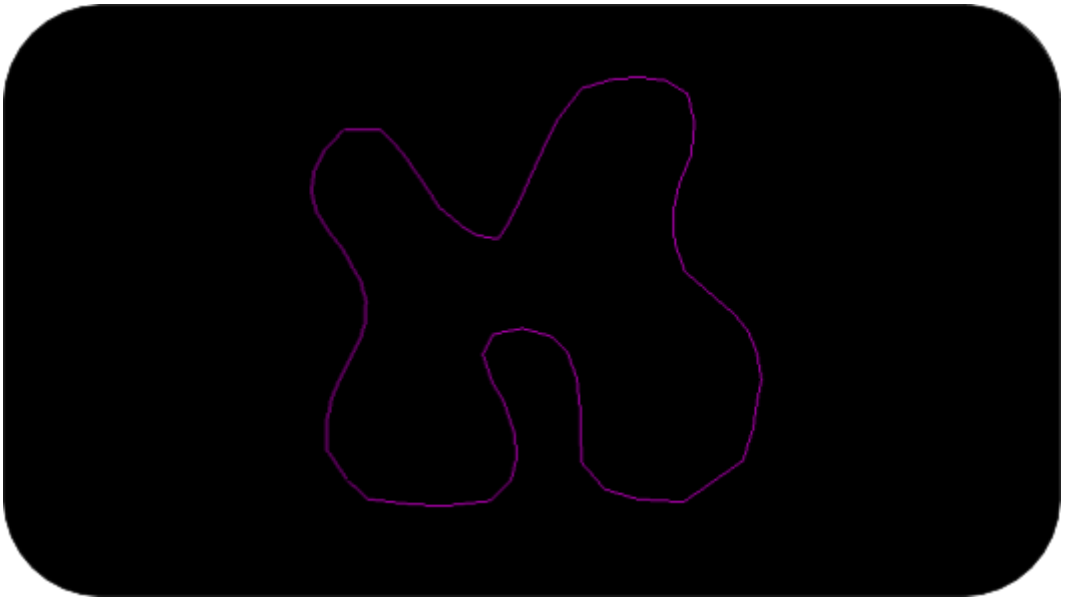

Figure 11 : Crust du maillage

VI) Conclusion

Ce TP aura été une bonne introduction au domaine de la géométrie algorithmiques et plus particulièrement aux méthodes de reconstruction d'un maillage a partir de nuages de points. Il aurait été tout aussi intéressant d'extrapoler ce sujet en 3D afin de pouvoir observer comment générer des volumes. en outre les maillages obtenus comportent toujours certains défauts. En effet il est possible que ces derniers contiennent de nombreux triangles très fin répondant aux contraintes de Delaunay. Or de tels triangles peuvent être problématiques (si l'on souhaite effectuer une simulation sur notre maillage par exemple). Pour régler ce problème, l'algorithme de Ruppert aurait pu être utilisé.