CENTRALE LYON

Calcul et modélisation géométrique pour l'informatique graphique

Rapport de TP

Élèves :
Baptiste GRIGNON

Enseignants :
Raphaëlle CHAINE

15 avril 2025

CENTRALE LYON

Table des matières

CENTRALE LYON

1 Introduction

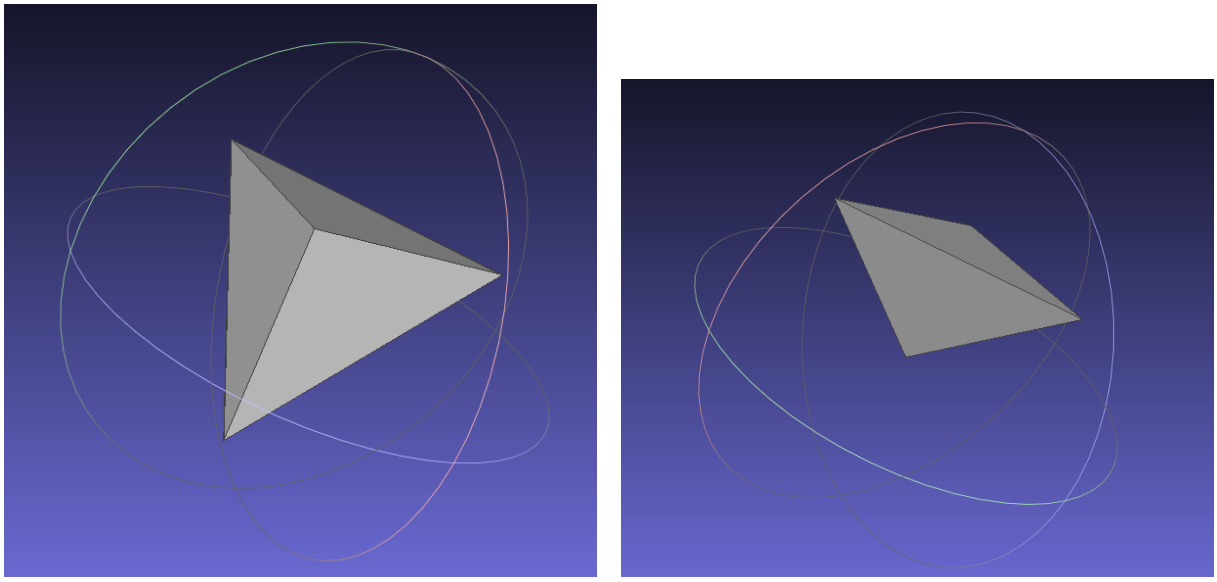
L'objet de ce rapport est la présentation des résultats obtenus au cours des 3 TPs du cours de **Calcul et modélisation géométrique pour l'informatique graphique**, portant respectivement sur la structure des maillages, le calcul d'opérateurs différentiels (Laplacien) sur le maillage, et la création d'un "beau" maillage (triangulation de Delaunay) par l'algorithme de Lawson.

2 TP1

Le but du TP1 est de définir une structure topologique de maillage permettant leur manipulation, d'encoder quelques maillages simples à la main, et de réaliser des fonctions permettant de les lire et de les écrire dans des fichiers OFF (*Object File Format*).

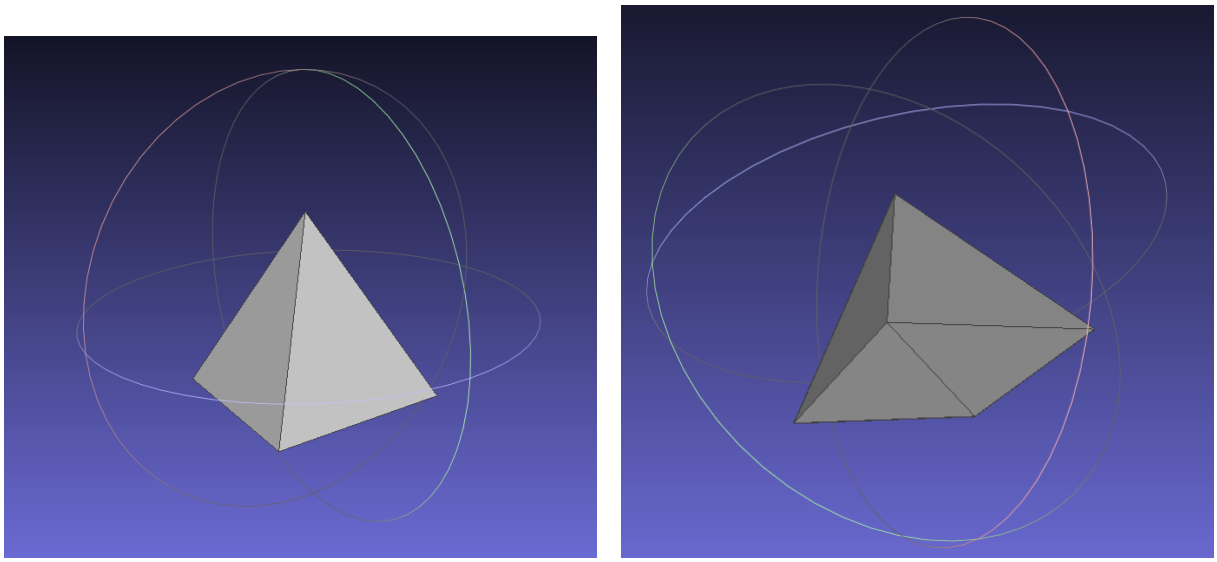
Un maillage est défini par une liste de triangles et une liste de sommets. Un sommet est défini par sa position dans l'espace et par l'indice d'un triangle auquel il est connecté dans la liste des triangles. Un triangle est défini par la liste des indices de ses 3 sommets (dans la liste de sommets), ordonnée dans le sens direct, et par la liste des indices des 3 triangles voisins (connectés au triangle par une arête), ordonnées comme la liste de sommets, de sorte à ce que l'indice "local" (0, 1 ou 2) d'un triangle soit identique à celui du sommet qui lui fait face.

On peut ensuite utiliser cette structure pour encoder 3 maillages simples à la main : un tétraèdre, une pyramide à base carrée, et une bounding box 2D. La bounding box est un rectangle, mais dont les côtés sont reliés à un sommet virtuel "à l'infini" (qui est le premier sommet de la liste), par des triangles (que l'on qualifie également de virtuels). On peut ensuite écrire (automatiquement) ces maillages au format OFF, et les visualiser à l'aide du logiciel Mesh Lab :



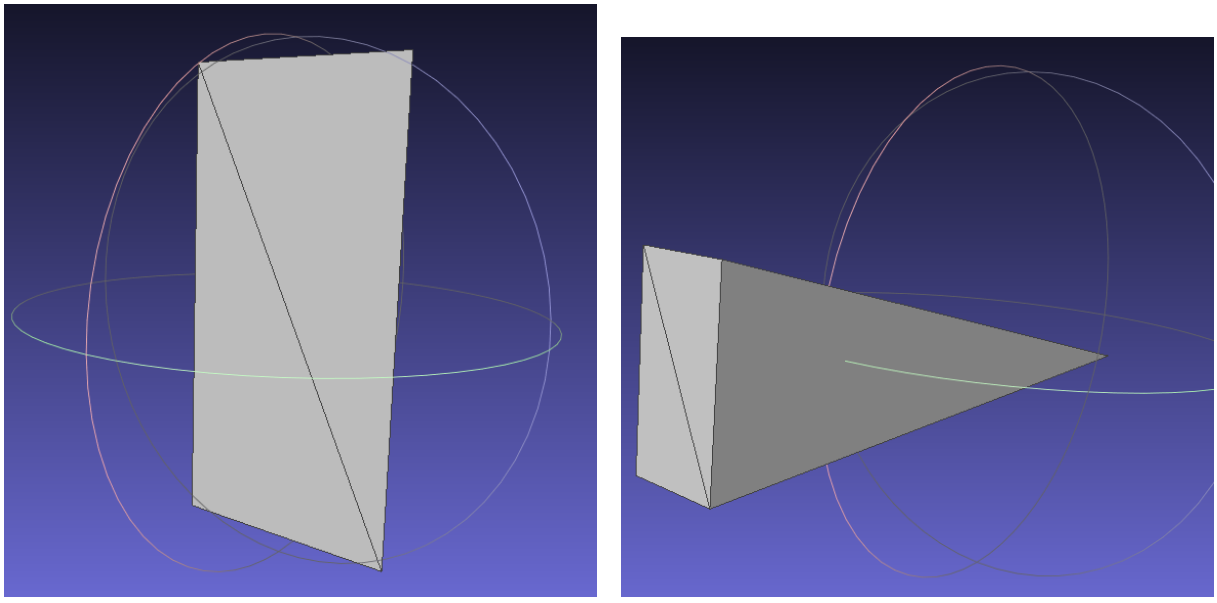
(a) Vue de dessus (b) Vue de dessous

FIGURE 1 – Tétraèdre



(a) Vue de dessus (b) Vue de dessous

FIGURE 2 – Pyramide



(a) Sommet virtuel caché (b) Sommet virtuel visible

FIGURE 3 – Bounding box 2D

On notera que le sommet et les triangles virtuels sont tout simplement ignorés quand on écrit le maillage dans un fichier OFF. Ainsi, à la lecture d'un maillage, on ne pourra pas récupérer cette information topologique. On pourrait chercher les arêtes sur le bord de la surface et les reconnecter à un sommet virtuel, mais cela n'a pas été implémenté.

De plus, les fichiers OFF ne stockent pas le triangle connecté pour les sommets et de liste de triangles voisins pour les triangles, donc il faut réaliser "la couture" du maillage lors de la lecture du fichier OFF. Trouver le triangle connecté à un sommet est simple, on a juste à compléter l'information à la lecture des indices des sommets formant les triangles. Pour trouver les triangles voisins entre eux, on enregistre chaque arête d'un triangle dans une "map" lorsqu'on lit les informations du triangle. Si l'arête n'existe pas encore dans la map, on l'y met, avec les informations du triangle (indice global du triangle et indice local de l'arête = indice local du triangle connecté à travers l'arête). Si elle existe, on récupère les informations stockées, on la supprime de la map, et on utilise les informations pour mettre à jour la couture.

On peut ainsi, charger et réécrire des modèles plus complexes, comme la reine alien, utilisée pour le TP suivant :

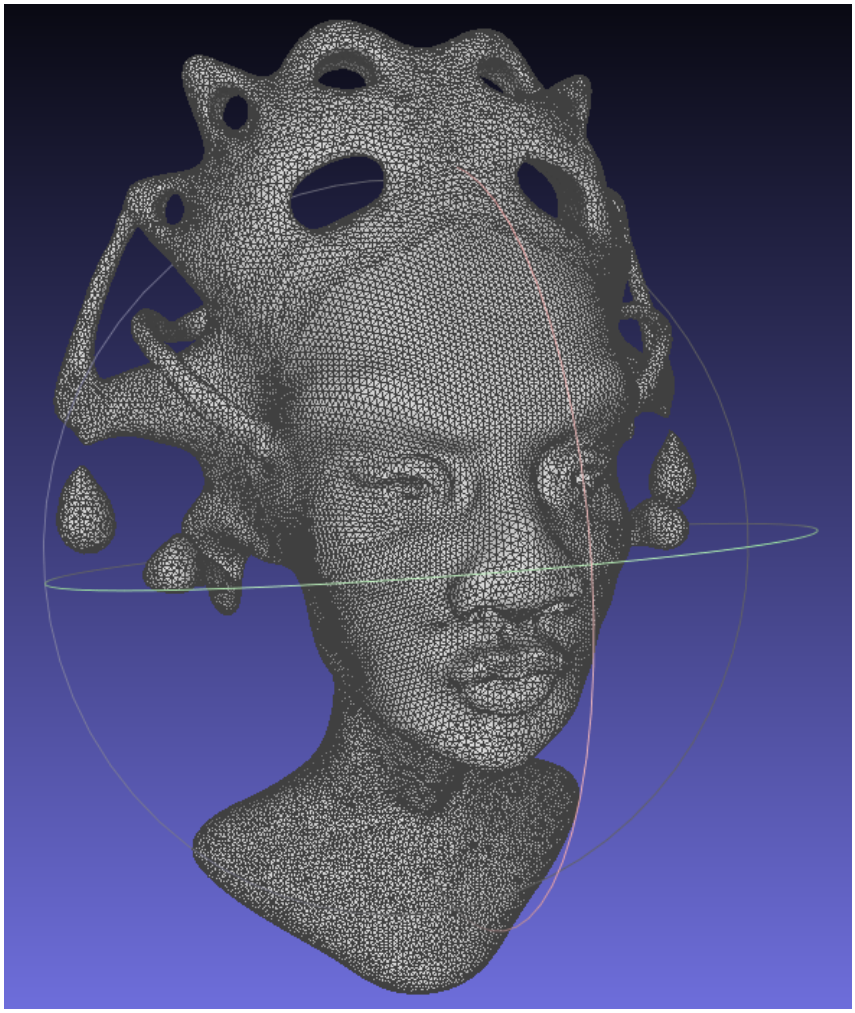


FIGURE 4 – Modèle fourni de reine alien

3 TP2

Le but du TP2 est de calculer le Laplacien Δu (qu'on notera Lu) d'une fonction arbitraire u aux sommets du maillage, et de s'en servir pour calculer la courbure sur le maillage et simuler la diffusion thermique. Pour calculer le Laplacien, on utilise l'équation suivante :

$$(Lu)_i = \frac{1}{2A_i} \sum_j (cot\alpha_{ij} + cot\beta_{ij})(u_j - u_i) \tag{1}$$

où i est l'indice d'un sommet, j ($\neq i$) parcourt les indices des sommets des triangles connectés à i, et où l'aire A_i, l'angle α_{ij} et l'angle β_{ij} sont donnés par :

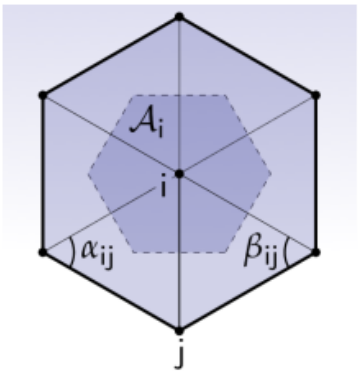


FIGURE 5 – Schéma du voisinage d'un sommet

On approxime A_i par le tiers de la somme des aires des triangles connectés à i. On peut calculer l'aire d'un triangle en divisant par 2 la norme du produit vectoriel de 2 arêtes de même origine, c'est-à-dire si $\vec{AB}$ et $\vec{AC}$ sont 2 arêtes d'un triangle : $A_{triangle} = \frac{|||\vec{AB} \times \vec{AC}||}{2}$

Pour calculer $cot\alpha_{ij}$ (respectivement $cot\beta_{ij}$), on utilise la formule suivante :

$$cot\alpha_{ij} = \frac{\vec{AB} \cdot \vec{AC}}{||\vec{AB} \times \vec{AC}||} \tag{2}$$

où $\cdot$ dénote du produit scalaire, $\times$ dénote du produit vectoriel et où $(\vec{AB}, \vec{AC})$ définit α_{ij} (non orienté) (respectivement βij). Il n'y a pas d'ambiguïté sur le signe du produit vectoriel, car les angles des triangles sont dans $[0, \pi]$.

Enfin, il s'agit de parcourir le voisinage de i en utilisant les informations topologiques pour calculer $(Lu)_i$.

Courbure En calculant le laplacien des coordonnées x, y et z d'un sommet, on peut obtenir le vecteur suivant :

$$(\Delta x, \Delta y, \Delta z) = 2H\vec{n} \tag{3}$$

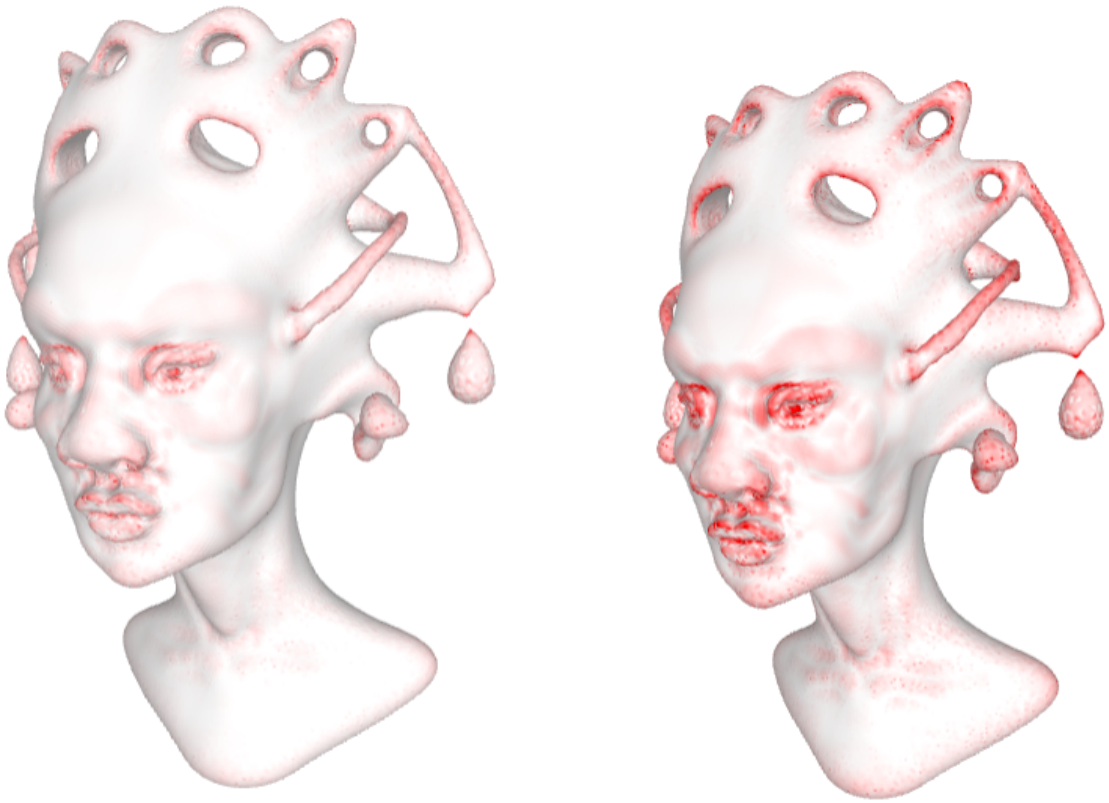
où H est la courbure signée de la surface au sommet et $\vec{n}$ le vecteur normal à la surface au niveau du sommet.

On peut ainsi facilement extraire la valeur absolue de la courbure :

$$|H| = \frac{|(\Delta x, \Delta y, \Delta z)|}{2} \tag{4}$$

En mappant les valeurs de courbure (absolue) à des couleurs ($[H_{min}, H_{max}] \rightarrow [0, 256]$), et en les enregistrant dans un fichier OFF, on peut ainsi visualiser la courbure. Pour chaque sommet, on enregistre (en plus de sa position) la couleur (format RGB) correspondant à sa courbure, ce qui n'est pas prévu par le standard définissant OFF. Mesh Lab ne peut pas lire ces fichiers OFF non standards, mais le visualiseur en ligne `3dviewer.net` le peut : la couleur sur les triangles est un dégradé obtenu par interpolation des couleurs des sommets.

On peut donc visualiser la courbure absolue du modèle de reine alien, dans la figure suivante. "Couleur saturée" signifie qu'on réalise le mapping $[H_{min}, H_{max}/2] \rightarrow [0, 256]$ pour mieux mettre en évidence les faibles valeurs.

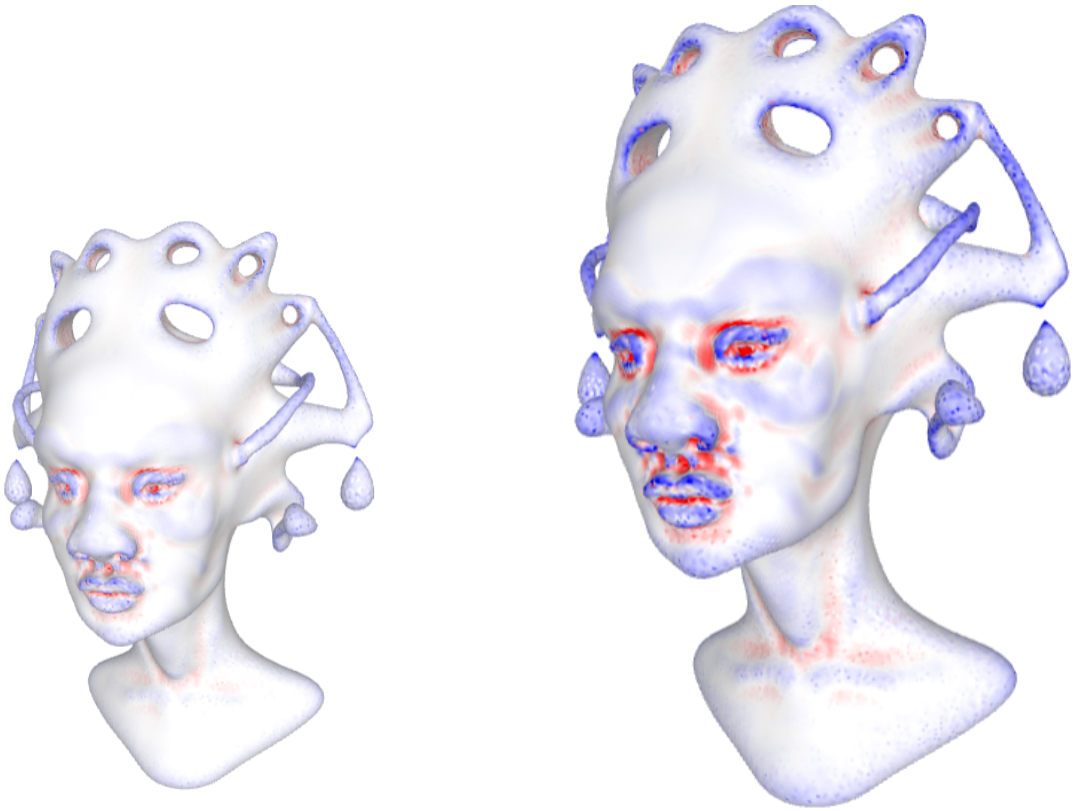


(a) Couleur normale (b) Couleur saturée

FIGURE 6 – Courbure absolue de la reine alien

On peut également essayer d'extraire la courbure signée en supposant que la normale en un sommet est proche de la normale des triangles qui lui sont connectés (les triangles visibles sur la figure 5 par exemple). Ainsi, on extrait les normales des triangles qui entourent un sommet et on réalise le produit scalaire entre ces normales et $(\Delta x, \Delta y, \Delta z)$. Si un de ces produits scalaires est positif, c'est que $(\Delta x, \Delta y, \Delta z)$ est dans le même sens que le vecteur normal au triangle, et donc que la courbure H est positive. Sinon, ils sont dans des sens opposés et la courbure H est négative. On réalise le test pour chaque triangle connecté au sommet, et on choisit le signe de la courbure par un vote majoritaire.

La figure suivante présente le modèle de reine alien avec la courbure signée. Une courbure positive est représentée en rouge et une courbure négative est représentée en bleu.



(a) Couleur normale (b) Couleur saturée

FIGURE 7 – Courbure signée de la reine alien

On remarque une courbure positive aux endroits concaves et une courbure négative aux endroits convexes.

Diffusion thermique On peut également utiliser l'opérateur Laplacien pour simuler la diffusion thermique. L'équation de diffusion est :

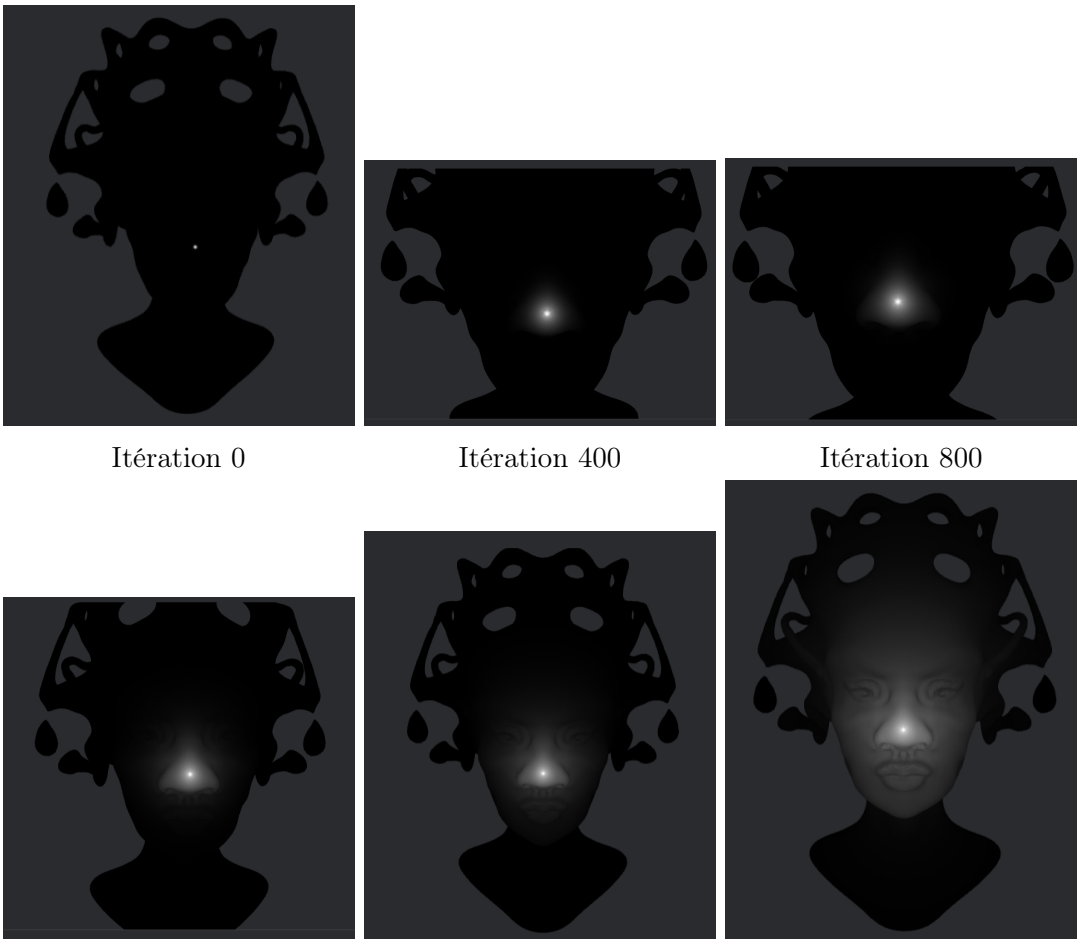
$$\Delta u = \alpha \frac{\delta u}{\delta t} \tag{5}$$

où α est la diffusivité thermique. En utilisant un schéma explicite de discrétisation pour l'équation 5, on obtient :

$$u_{i+1} = u_i + \alpha (Lu)_i \Delta t \tag{6}$$

où $\Delta t = t_{i+1} - t_i$ est le pas temporel. Il doit être choisi tel que $\Delta t \leq \frac{L^2}{2\alpha}$ avec L la plus petite longueur du maillage pour que le schéma soit stable.

On force un sommet au bout du nez à la température $u_i = 100$°C. On calcule Lu_i pour chaque sommet, et on utilise l'équation 6 pour mettre à jour u_i. On répète l'opération pour N itérations. Sur la figure suivante, on présente les résultats pour certains nombres d'itérations :



Itération 0 | Itération 400 | Itération 800

Itération 4000 | Itération 10000 | Itération 50000

FIGURE 8 – Diffusion de la température, représentée en nuance de gris, non saturée

FIGURE 9 – Diffusion de la température, représentée en nuance de gris, à l'itération 150 000, non saturée

4 TP3

Le but du TP3 est d'implémenter un processus de création d'une triangulation de Delaunay à partir d'une liste de points (x, y, z) de l'espace. Pour ce faire, on commence par implémenter la création d'une triangulation naïve, qu'on améliore ensuite à l'aide de l'algorithme de Lawson. On réalisera la triangulation en 2D dans le plan (x, y) en mettant de côté les valeurs d'altitude z, qu'on réintroduira à la fin.

Opérateurs On commence par définir 2 opérateurs topologiques élémentaires sur le maillage :

- le "flip" d'une arête. Cette opération s'applique sur 2 triangles reliés par une arête. Leur union forme un quadrilatère dont l'arête est l'une des diagonales. L'opération "flip" d'une arête revient à l'effacer et la remplacer par l'autre diagonale du quadrilatère.
- le "split" d'un triangle. Cette opération s'applique à un triangle et prends en entrée un sommet P non relié à un triangle. Elle consiste en la division du triangle en 3 : on relie à P par 2 nouvelles arêtes chaque arête du triangle original.

Ces opérations sont topologiques, c'est-à-dire qu'elles ne tiennent pas compte de la géométrie des objets, uniquement de leur structure. Par exemple, l'opération split peut parfaitement s'appliquer si le nouveau sommet P est en dehors du triangle. De plus, il faut faire très attention lors de l'implémentation de ces opérations de mettre à jour la structure des objets pour que le maillage reste toujours bien cousu.

Il nous faut également définir quelques prédicats pour réaliser des tests :

- le prédicat d'orientation : soit 3 points A, B et C, le résultat du prédicat donne le sens (direct ou non) du triangle. Pour cela, on test le signe de $(\vec{BC} \times \vec{BA}) \cdot \vec{n}$ où $\vec{n}$ est la normale du triangle.
- le prédicat d'inclusion dans un triangle : soit 4 points A, B, C et P, le résultat du prédicat donne si P est dans le triangle ABC ou non. Ce prédicat utilise le prédicat d'orientation pour déterminer l'orientation des 3 triangles qui seraient issus du split de ABC par P. S'ils sont tous orientés dans le même sens, P est dans le triangle, sinon il n'y est pas.
- le prédicat d'inclusion dans un cercle circonscrit : soit 4 points A, B, C et P, le résultat du prédicat donne si P est dans cercle circonscrit au triangle ABC ou non.
- le prédicat de convexité : soit 4 points A, B, C et D, le résultat du prédicat donne si le quadrilatère est convexe ou non. Ce prédicat est similaire au prédicat d'orientation, c'est-à-dire qu'on vérifie que tous les triplets successifs de sommets sont orientés dans le même sens.

Enfin, on définit un opérateur d'élévation, utilisé dans le prédicat d'inclusion dans un cercle circonscrit :

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3, \quad (x, y) \mapsto (x, y, x^2 + y^2) \tag{7}$$

Cet opérateur projette un point du plan dans l'espace sur un paraboloïde. On s'en servira plus tard pour visualiser que notre triangulation est bien de Delaunay.

Triangulation naïve Pour réaliser la triangulation, on commence par créer une fonction *insertPoint* permettant d'insérer un point P dans la triangulation. Dans cette fonction, commence par itérer sur tous les triangles pour chercher (à l'aide de tests utilisant le prédicat dédié) si P appartient (géométriquement) à l'un d'eux. Si c'est le cas, on split le triangle avec P comme nouveau sommet. Sinon, on liste toutes les arêtes sur le bord de l'enveloppe convexe du maillage, c'est-à-dire les arêtes connectées à un triangle virtuel. Ensuite, à l'aide du prédicat d'orientation, on cherche les arêtes visibles depuis P, c'est-à-dire les arêtes $\vec{AB}$ telles que le triangle ABP ne soit pas orienté dans le sens direct. On précisera que comme tous les triangles du maillage sont orientés dans le sens direct, les arêtes $\vec{AB}$ sont orientées dans le même sens.

Une fois qu'on connaît les arêtes visibles par P sur le bord de l'enveloppe convexe, on en choisit une (la dernière par exemple) et on split par P le triangle virtuel qui lui est connecté. Ensuite, on parcourt la liste de ces arêtes vers la gauche (respectivement la droite) en partant de l'arête choisie et on flip l'arête virtuelle (c'est-à-dire une arête qui relie 2 triangles virtuels, ou de manière équivalente, une arête dont l'un des sommets est le sommet virtuel à l'infini) de droite (respectivement de gauche) du triangle virtuel connecté à l'arête visible courante. Ainsi, on inclut P dans la triangulation en étendant l'enveloppe convexe.

Pour réaliser la triangulation naïve à partir d'une liste de points de l'espace (lus depuis un fichier texte), on commence par y chercher 3 points formant un triangle orienté dans le sens direct. Pour cela, on prend les 2 premiers points A et B de la liste et on la parcourt jusqu'à trouver (en utilisant le prédicat d'orientation) un point C tel que ABC soit orienté dans le sens direct. Cela suppose que C existe, mais si les points fournis sont suffisamment nombreux et ne sont pas ordonnés de manière particulière, l'hypothèse est raisonnable.

Ainsi, le point de départ de la triangulation est le sommet virtuel à l'infini, relié par 3 faces virtuelles au premier triangle direct trouvé, formant un "tétraèdre". Ensuite, on utilise la fonction *insertPoint* pour insérer itérativement tous les points restants de la liste dans la triangulation (soit en splittant un triangle, soit en étendant l'enveloppe convexe). On précise que les points insérés sont en fait les projections sur le plan (x,y) des points de l'espace. Les altitudes z sont stockées pour plus tard, ainsi qu'un mapping permettant de faire correspondre l'indice d'un point dans la liste des altitudes avec son indice dans la liste des sommets.

Triangulation de Delaunay Une fois la triangulation naïvement réalisée, on l'améliore avec l'algorithme de Lawson :

1. On parcourt toutes les arêtes. Si une arête n'est pas localement de Delaunay, si le quadrilatère formé par la réunion des 2 triangles qu'elle connecte est convexe et qu'aucun des 2 triangles n'est virtuel, alors on l'ajoute à une queue.

2. On flip toutes les arêtes dans la queue jusqu'à ce que celle-ci soit vide. À chaque flip, on vérifie pour chacune des 4 arêtes correspondant aux côtés du quadrilatère formé par la réunion des 2 triangles connectés par l'arête flippée si elle vérifie la condition mentionné en 1. Si oui, on l'ajoute à la queue.

Pour tester si une arête est localement de Delaunay, on utilise le prédicat d'inclusion dans le cercle circonscrit à un triangle. On considère ABC l'un des 2 triangles reliés par une arête AC, et D le point qui fait face à l'arête dans l'autre triangle. Pour que l'arête AC soit localement de Delaunay, il faut que D ne soit pas inclus dans le cercle circonscrit à ABC et que B ne soit pas inclus dans le cercle circonscrit au triangle ACD.

On notera que dans la plupart du programme, les arêtes sont représentées par le couple (indice global de triangle, indice local d'arête), mais que dans la queue de l'algorithme de Lawson, on les représente par les indices de leurs extrémités dans la liste des sommets. En effet, les flips font "bouger" les triangles et peuvent invalider la première représentation, tandis que la deuxième est robuste aux flips. Grâce à la structure de données, la traduction entre ces 2 représentations est très rapide.

Petite triangulation Le code a d'abord été testé sur un petit maillage réalisé à la main, qui a également permis de tester les opérations élémentaires. La triangulation construite à la main est la suivante :

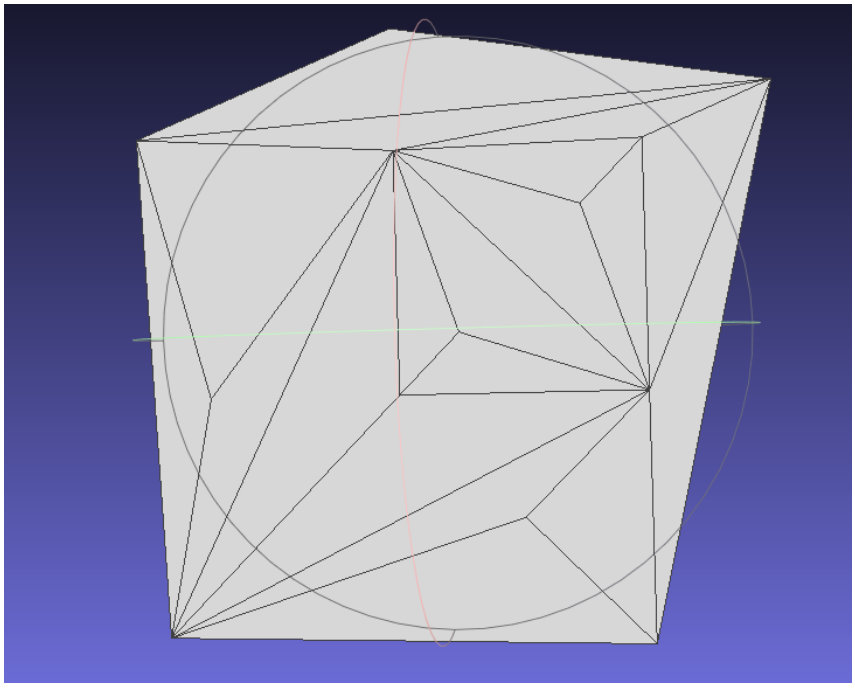
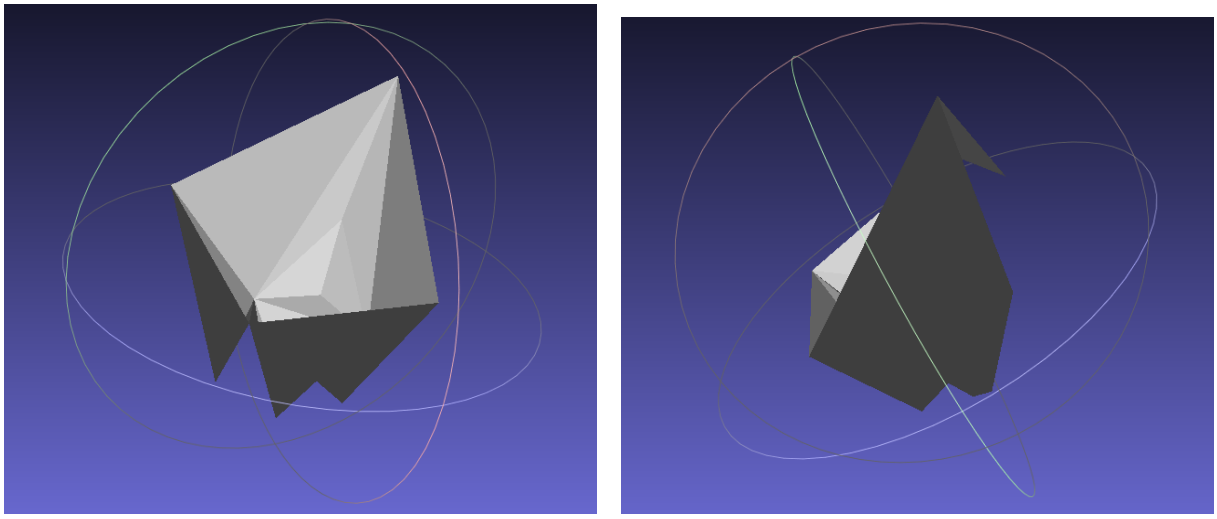


FIGURE 10 – Petite triangulation naïve

La qualité de la triangulation est mauvaise. On peut également le voir en projetant les points dans l'espace des sphères (opérateur ϕ). On observe alors que la triangulation ne

forme pas un "beau" paraboloïde, comme on l'attendrait d'une triangulation de Delaunay. En effet, des "replis" sont présents :



(a) Vue de "devant" (b) Vue de "derrière"

FIGURE 11 – Petite triangulation naïve dans l'espace des sphères

Après application de l'algorithme de Lawson, on obtient la triangulation suivante :

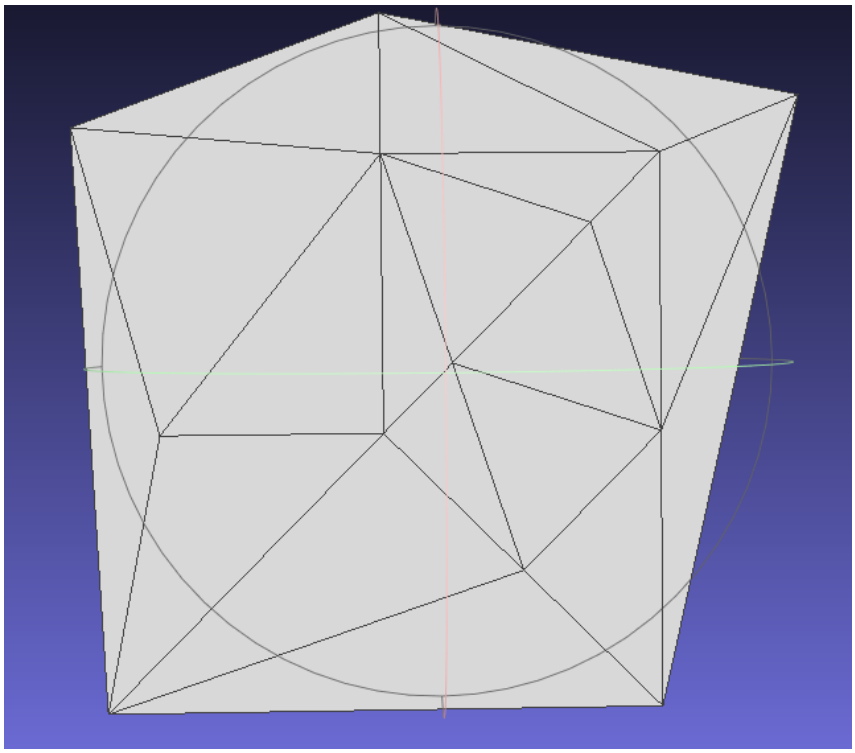
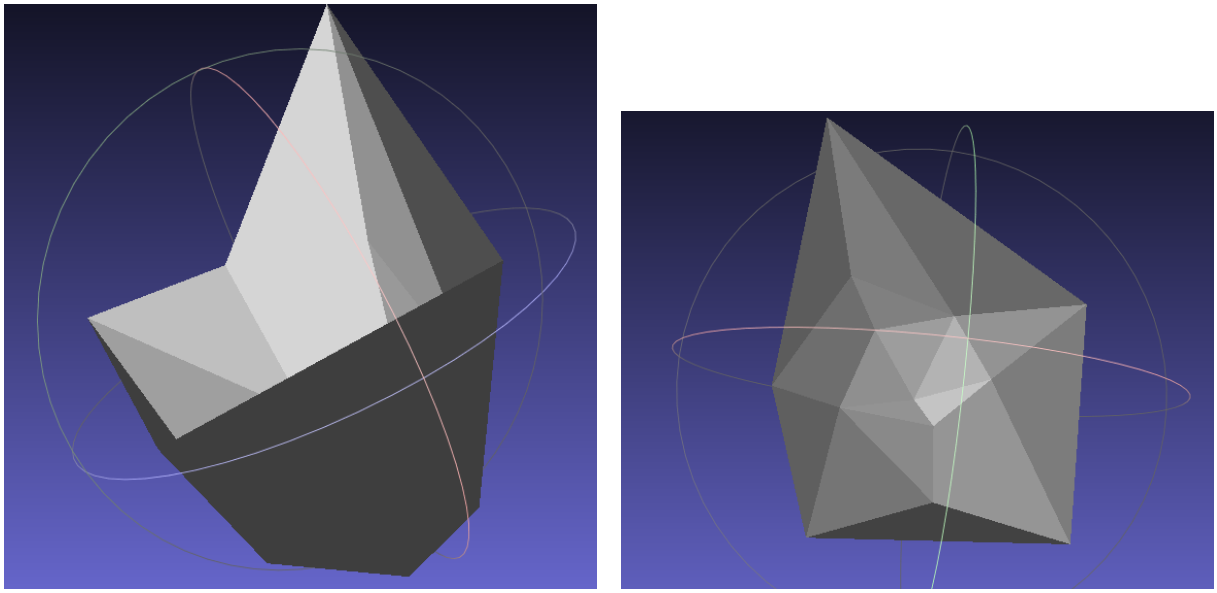


FIGURE 12 – Petite triangulation de Delaunay

La triangulation a l'air significativement plus jolie. On peut le confirmer avec la projection sur le paraboloïde, qui est maintenant nette et sans replis :



(a) Vue de coté (b) Vue de dessus

FIGURE 13 – Petite triangulation de Delaunay dans l'espace des sphères

Grandes triangulations On applique maintenant le code sur les nuages de points fournis. En particulier, pour le nuage de points "alpes poisson", on obtient la triangulation naïve suivante :

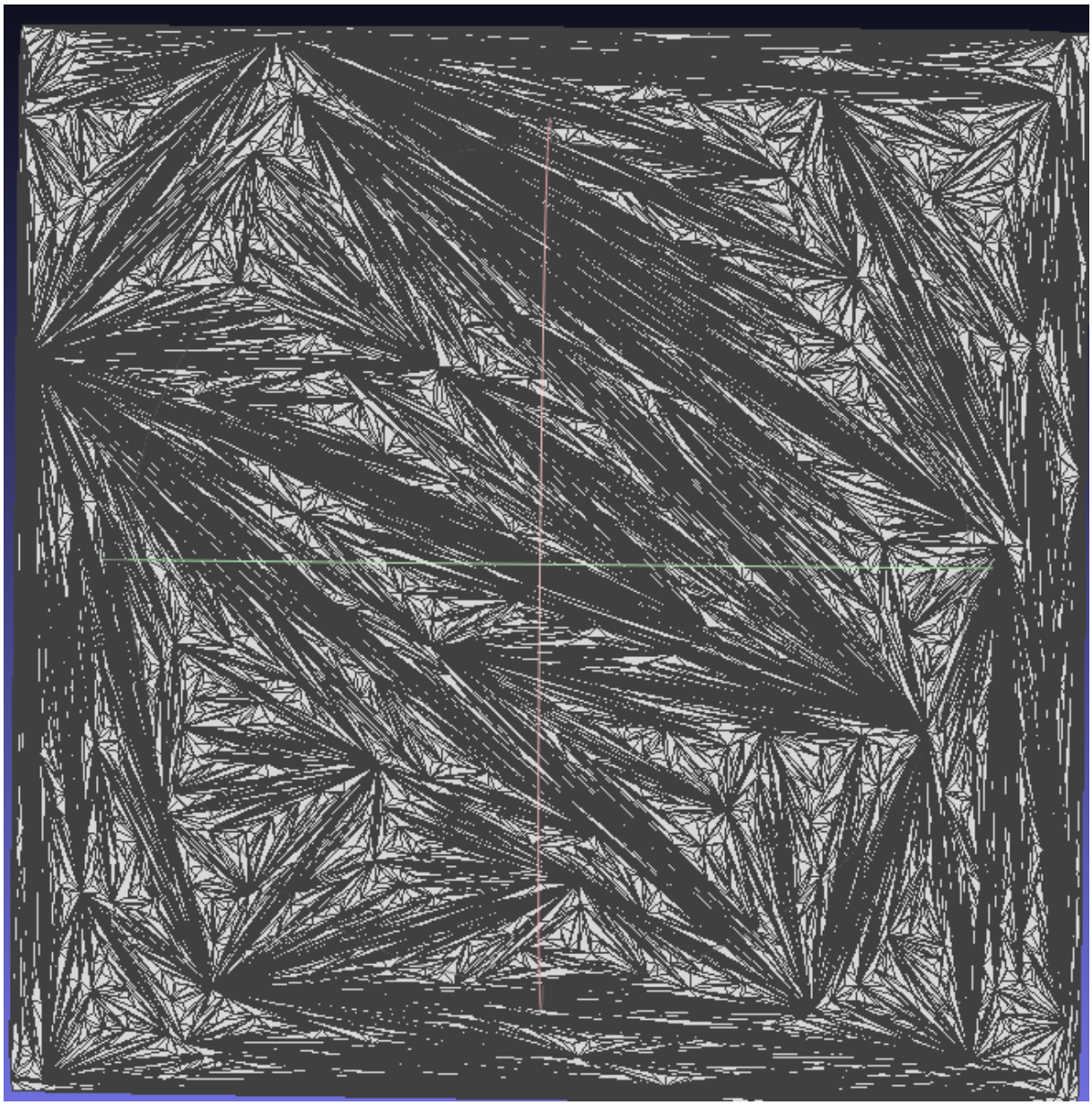


FIGURE 14 – Triangulation naïve du nuage de points "alpes poisson"

La triangulation est particulièrement hideuse, avec tes triangles tellement aplatis que certaines arêtes apparaissent comme des droites parallèles. Après application de l'algorithme de Lawson, la triangulation est très nettement améliorée :

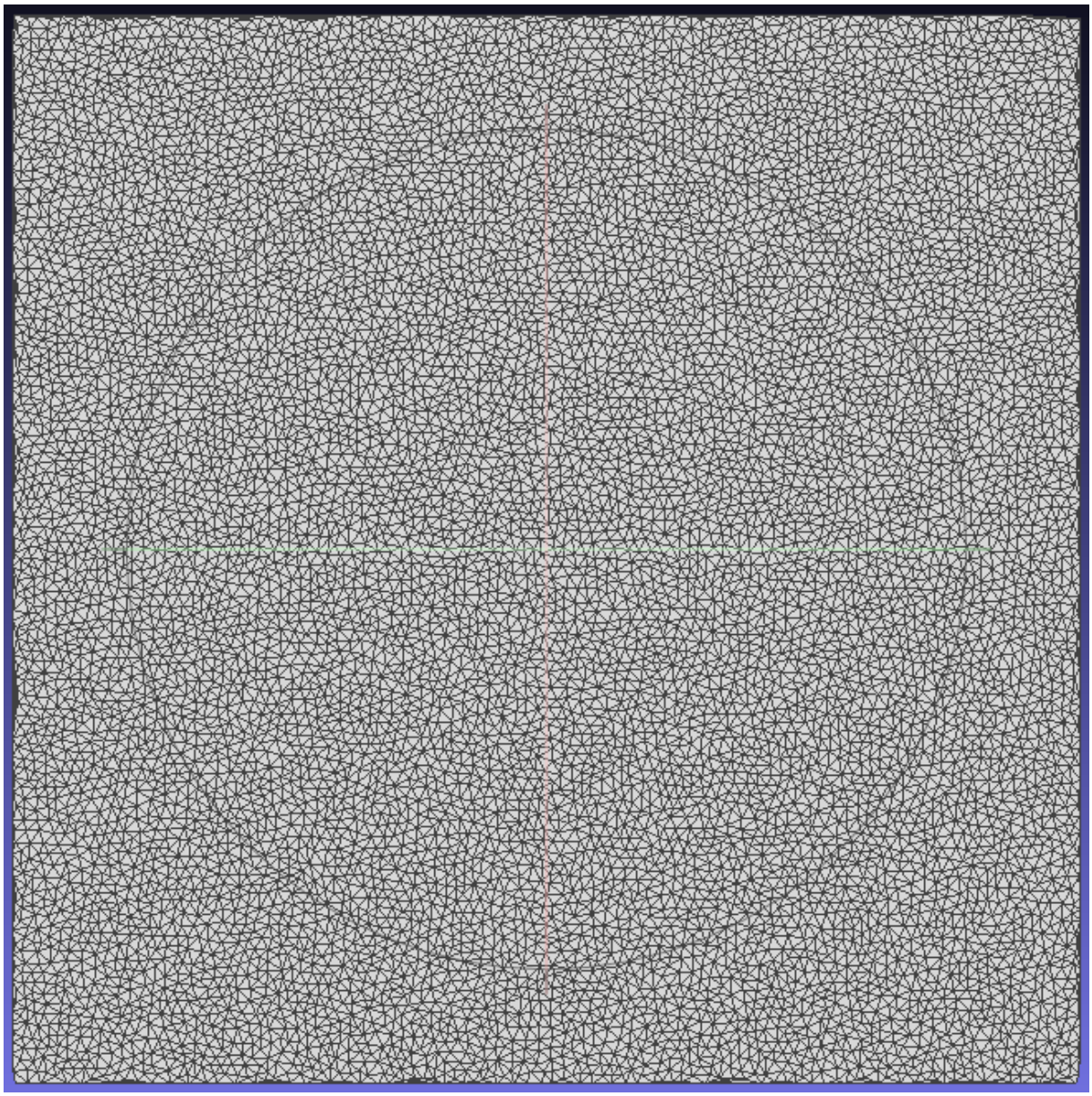


FIGURE 15 – Triangulation de Delaunay du nuage de points "alpes poisson"

De plus, elle forme une paraboloïde bien lisse dans l'espace des sphères :

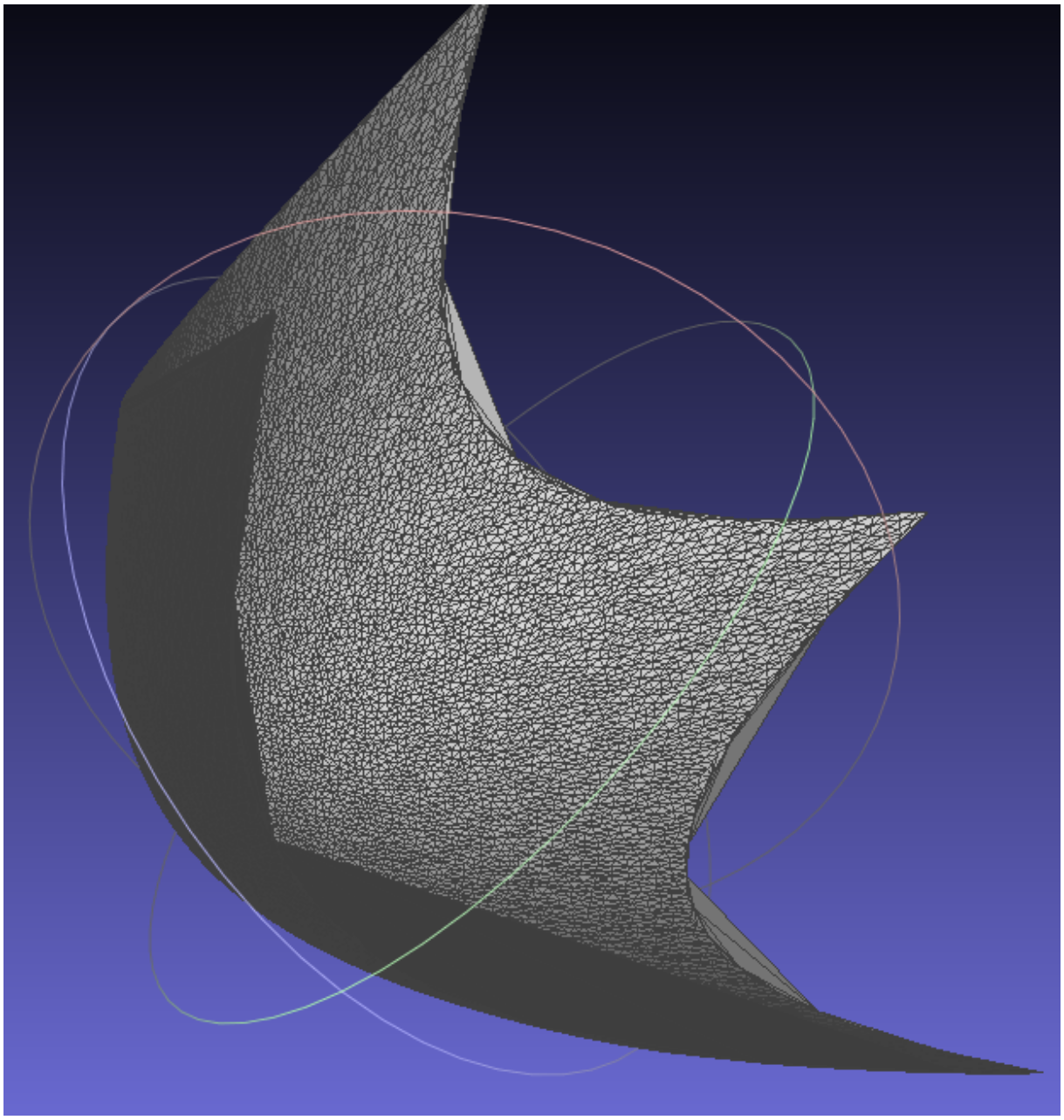


FIGURE 16 – Triangulation de Delaunay du nuage de points "alpes poisson" dans l'espace des sphères

Enfin, on peut réassigner à chaque point son altitude z originale :

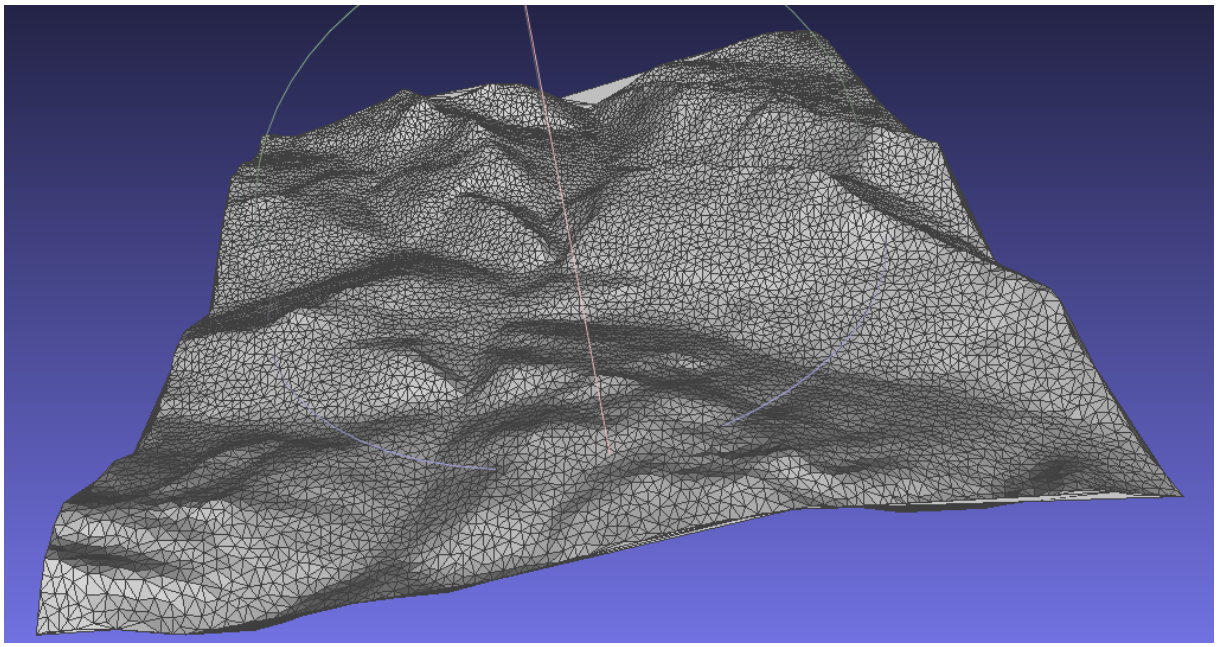
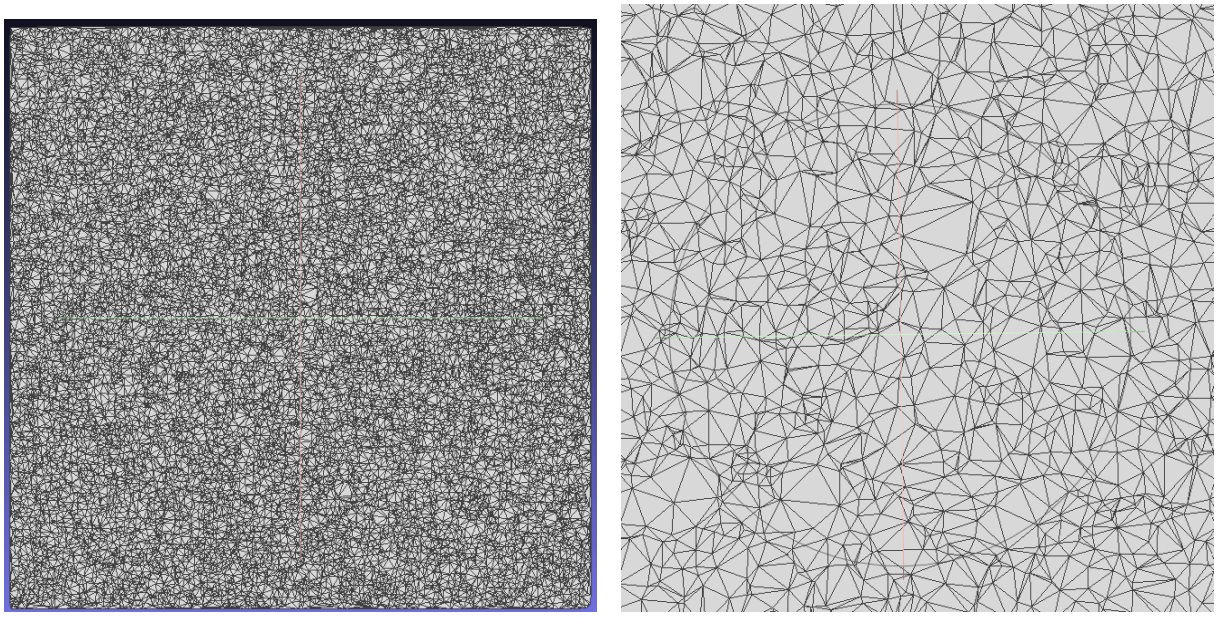


FIGURE 17 – Triangulation de Delaunay du nuage de points "alpes poisson"

Le résultat final est plutôt joli et ressemble bien au relief d'une montagne.

On peut appliquer le processus sur un autre nuage de points, par exemple nuage de points "noise random 2" et faire les mêmes constatations :



(a) Vue large (b) Vue zoomée

FIGURE 18 – Triangulation de Delaunay du nuage de points "noise random 2"

La triangulation naïve n'a pas été reproduite ici car elle a la même allure que pour le nuage de points précédent. On remarque cependant que pour la triangulation de Delaunay, les triangles ne sont pas aussi uniformes que pour le nuage de points précédent. En effet, le nuage des points précédent est (probablement) issu de l'algorithme de Poisson pour la reconstruction de surface, donc ses points sont "bien choisis", tandis que les points de ce nuage sont échantillonnés aléatoirement. Néanmoins, les triangles formés à partir de ces points sont les meilleurs possibles, ce qu'on peut constater sur la projection de la triangulation dans l'espace des sphères, qui forme à nouveau un beau paraboloïde :

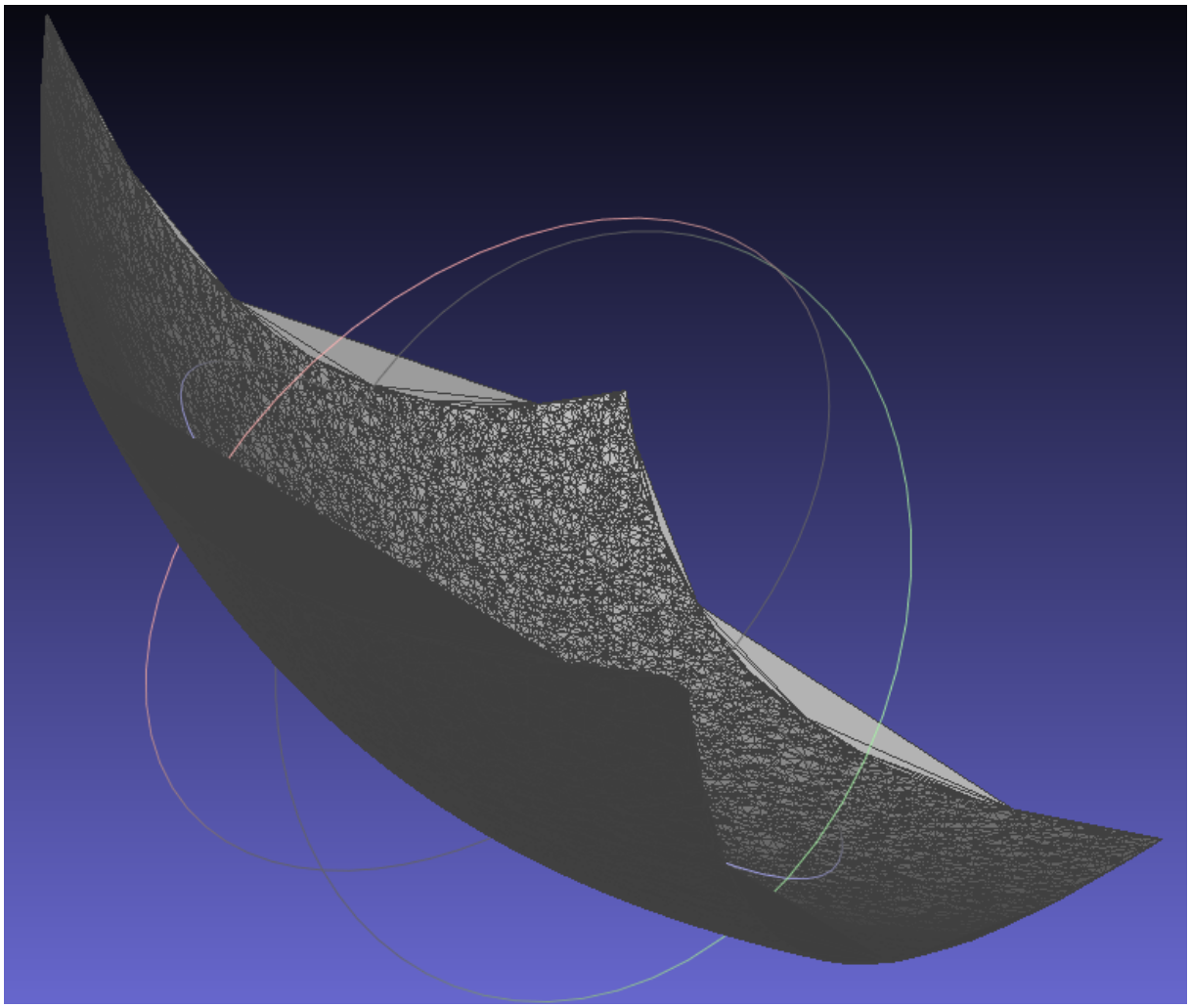


FIGURE 19 – Triangulation de Delaunay du nuage de points "noise random 2" dans l'espace des sphères

Enfin, on réapplique l'élévation, pour trouver un paysage différent, issu d'un bruit :

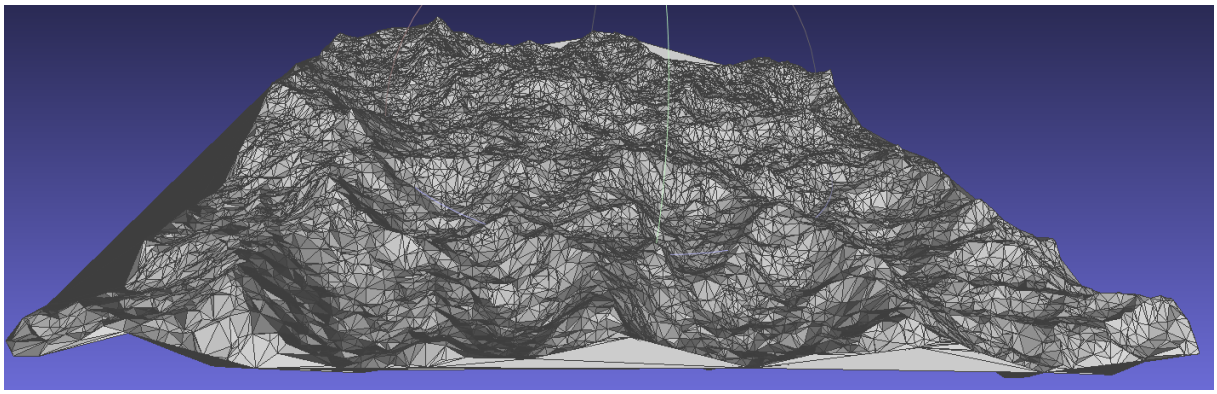


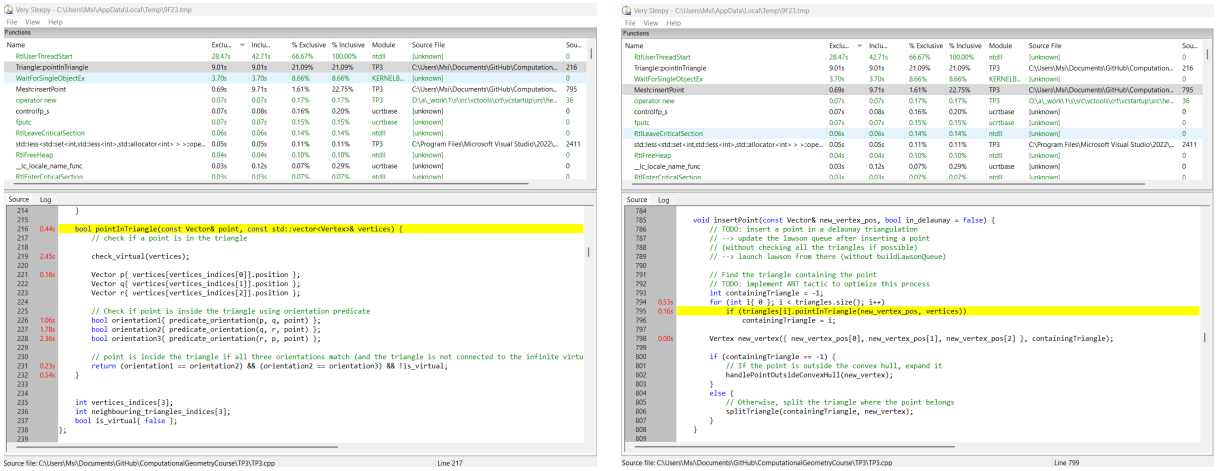
FIGURE 20 – Triangulation de Delaunay du nuage de points "noise random 2"

Remarques en vracs En observant l'évolution de la longueur de queue de l'algorithme de Lawson, on remarque que le nombre d'arêtes à flipper stagne voire augmente un petit peu au début, avant de décroître progressivement :

```
Improving the triangulation (Delaunay) using Lawson algorithm ...
Loop : 0 Edges to flip : 6241
Loop : 1000 Edges to flip : 6270
Loop : 2000 Edges to flip : 6290
Loop : 3000 Edges to flip : 6273
Loop : 4000 Edges to flip : 6254
Loop : 5000 Edges to flip : 6223
Loop : 6000 Edges to flip : 6179
Loop : 7000 Edges to flip : 6103
Loop : 8000 Edges to flip : 6029
Loop : 9000 Edges to flip : 5929
Loop : 10000 Edges to flip : 5782
Loop : 11000 Edges to flip : 5636
Loop : 12000 Edges to flip : 5466
Loop : 13000 Edges to flip : 5273
Loop : 14000 Edges to flip : 5086
Loop : 15000 Edges to flip : 4876
Loop : 16000 Edges to flip : 4649
Loop : 17000 Edges to flip : 4426
Loop : 18000 Edges to flip : 4219
Loop : 19000 Edges to flip : 4009
Loop : 20000 Edges to flip : 3792
Loop : 21000 Edges to flip : 3563
Loop : 22000 Edges to flip : 3361
Loop : 23000 Edges to flip : 3133
Loop : 24000 Edges to flip : 2950
Loop : 25000 Edges to flip : 2756
Loop : 26000 Edges to flip : 2568
Loop : 27000 Edges to flip : 2362
Loop : 28000 Edges to flip : 2215
Loop : 29000 Edges to flip : 2018
Loop : 30000 Edges to flip : 1901
Loop : 31000 Edges to flip : 1722
Loop : 32000 Edges to flip : 1570
Loop : 33000 Edges to flip : 1421
Loop : 34000 Edges to flip : 1285
Loop : 35000 Edges to flip : 1127
Loop : 36000 Edges to flip : 985
Loop : 37000 Edges to flip : 878
Loop : 38000 Edges to flip : 734
Loop : 39000 Edges to flip : 611
```

FIGURE 21 – Évolution de la longueur de la queue de l'algorithme de Lawson

En profilant l'exécution de la triangulation à l'aide du logiciel Very Sleepy, on remarque que la plupart du temps d'exécution du programme est passé à chercher à quel triangle appartient un point lors de son insertion dans la triangulation :

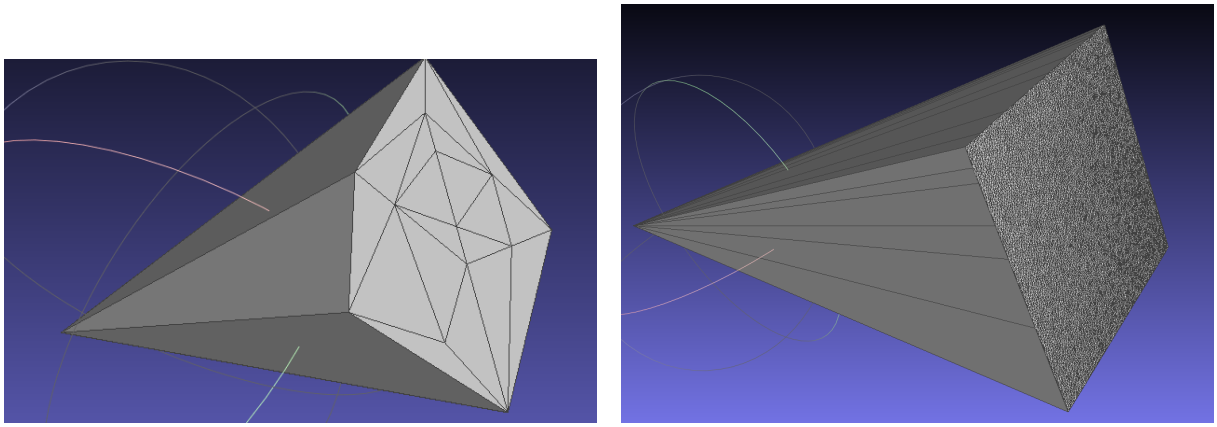


(a) (b)

FIGURE 22 – Captures d'écran du logiciel Very Sleepy

En effet, la complexité de ce processus est en O(mn) avec m le nombre de sommets à insérer et n le nombre de triangles. On pourrait l'optimiser en implémentant une stratégie de type "fourmi" : on sélectionne un triangle aléatoire sur lequel on fait le test premier d'inclusion, puis on voyage de triangle en triangle en direction du point à tester. Cette stratégie n'a pas été implémenté par manque de temps.

Enfin, on peut représenter les triangulations de Delaunay avec le point virtuel à l'infini :



(a) Petite triangulation (b) "alpes poisson"

FIGURE 23 – Triangulations de Delaunay avec point virtuel à l'infini