Projet 1 : Triangulation de Delaunay

Matthieu Gillet, Maé Mavromatis, Mathieu Poirier
ENS Rennes – ISTIC Université Rennes 1
L3 Informatique – parcours SIF

26 octobre 2018

Résumé

Ce rapport expose le travail fourni pour le projet 1 du module de programmation. Il est construit en deux parties : la première sur une triangulation de Delaunay classique, et la deuxième sur les extensions sur lesquelles nous avons travaillé.

Table des matières

1 Triangulation de Delaunay classique

1.1 Introduction

Ce projet étant centré sur la triangulation de Delaunay, commençons par reprendre les définitions.[3]

Une triangulation d'un ensemble de points de $\mathbb{R}^2$ est un ensemble d'arêtes maximal et reliant ces points sans créer d'intersection ; c'est-à-dire que si une arête est ajoutée dans cet ensemble, elle créera une intersection.

Remarquons que le résultat est un ensemble de triangles. En effet, si la figure contient un quadrilatère, l'ensemble d'arêtes n'est pas maximal car l'ajout d'une des diagonales est possible ; de même pour tout polygone de taille supérieure. L'ensemble de trangles ainsi obtenu justifie le nom de triangulation.

Une triangulation de Delaunay est une triangulation particulière, imposant la condition supplémentaire suivante : aucun point ne se trouve dans le cercle circonscrit d'un triangle.

1.2 Algorithme

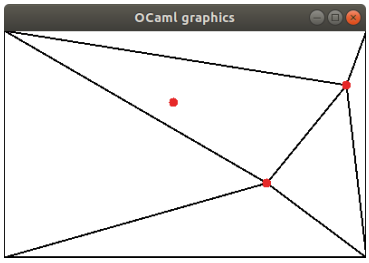
Pour construire la triangulation de Delaunay dans une fenêtre graphique, nous sommes partis des quatre coins, formant une triangulation de base de deux triangles.

La triangulation est construite incrémentalement, c'est-à-dire que l'on ajoute les points du nuage un par un. La procédure est décrite par l'algorithme 1 ci-dessous.

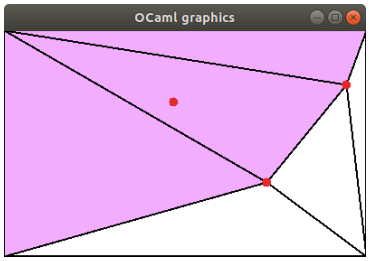
Algorithme 1 : Construction incrémentale

Entrées :
T une triangulation à n points
M un nouveau point
Corps :
pour tout triangle $t \in T$ **faire**
si M appartient au cercle circonscrit à t **alors**
Retirer t de T
fin si
fin pour
Construire la bordure B à partir des triangles retirés
pour (X, Y) adjacents dans B **faire**
Ajouter à T le triangle MXY
fin pour

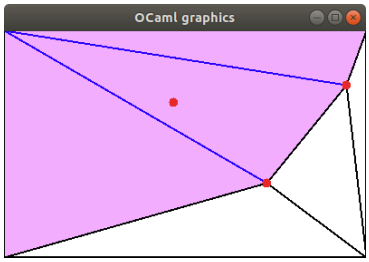
Par ailleurs, la figure 1 illustre cet algorithme dans un exemple simple.



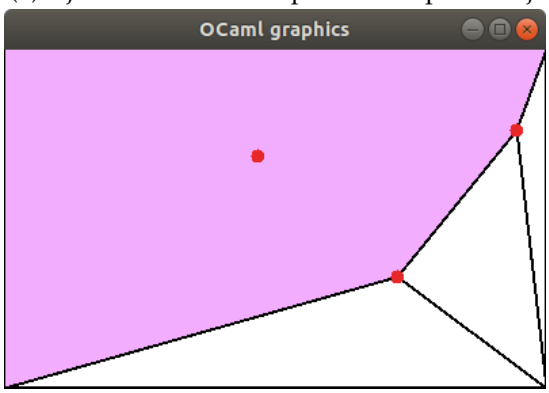
(a) Ajout d'un nouveau point.



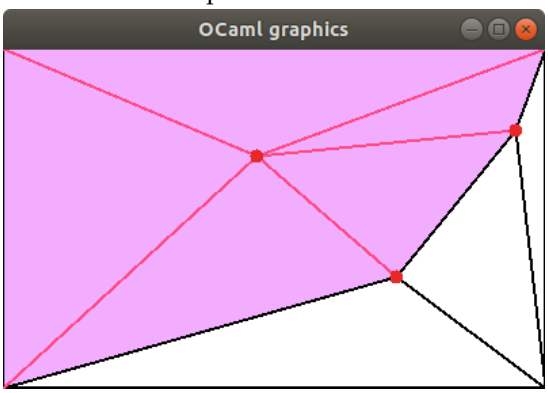
(b) Repérage des triangles dont le cercle circonscrit contient le point à ajouter.



(c) Suppression des triangles repérés.



(d) Construction de la bordure convexe.



(e) Construction des nouveaux triangles.

FIGURE 1 – Exemple d'ajout de point à une triangulation de Delaunay.

1.3 Implémentation

Le diagramme de classes présenté en figure 2 expose la hiérarchie des modules OCaml créés et exploités. Les contenants sont des répertoires et les contenus sont les modules.

La sémantique des flèches est la suivante : un élément A pointe vers un élément B si A nécessite B pour fonctionner. Les flèches qui relient les répertoires signifient qu'au moins un module du répertoire d'origine nécessite au moins un module du répertoire d'arrivée. Explorons cet arbre de dépendance en partant des feuilles.

Tout d'abord, l'ensemble du programme repose sur les modules de `Structures`. Ce répertoire contient les objets géométriques de base et des primitives d'accès rudimentaires de constructeur, accesseurs, et création d'ensemble.

Ensuite, les modules du répertoire `Tools` effectuent des opérations plus avancées sur ces premières structures. On trouve par exemple, dans le module `Geometry`, la fonction de calcul du cercle circonscrit à un triangle.

Le module `Delaunay` implémente quant à lui l'algorithme décrit dans la partie 1.2. Les abstractions déjà créées par l'existence des modules de `Tools` permettent de simplifier la lecture du programme de ce module, qui peut se permettre d'enchaîner des opérations géométriques non triviales sans avoir à se soucier de leur implémentation.

Enfin, le module `Display` permet de gérer l'affichage dans la fenêtre graphique de la triangulation obtenue grâce au module précédent.

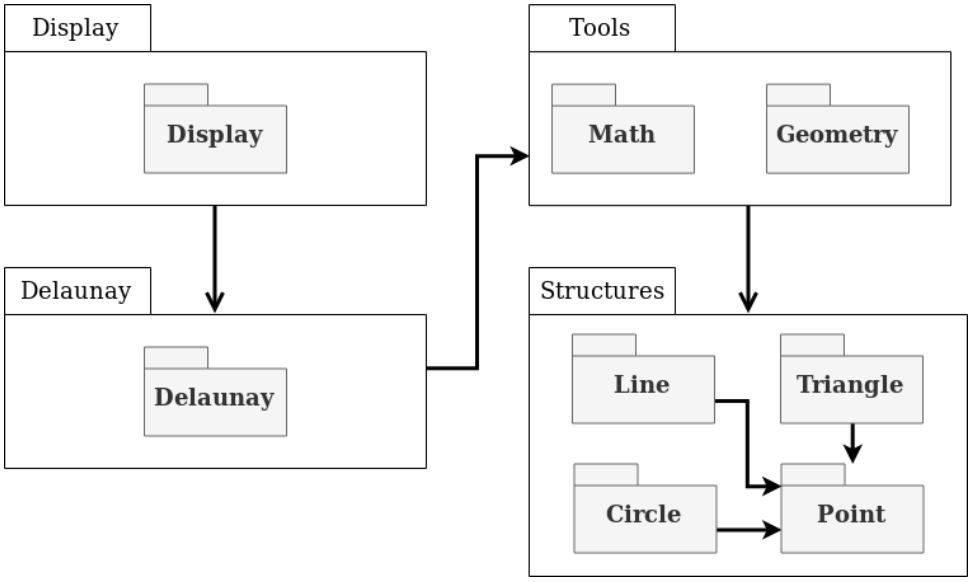


FIGURE 2 – Diagramme de classes pour la triangulation simple.

1.4 Conclusion

Nous avons écrit un algorithme incrémental de triangulation du plan en utilisant la méthode de Delaunay. Nous avons travaillé sur la modularité de notre code et avons implémenté une structure opaque dont les dépendances sont hiérarchisées. Nous avons ainsi obtenu une triangulation de Delaunay de la fenêtre graphique en utilisant les quatre coins de la fenêtre comme cas de base.

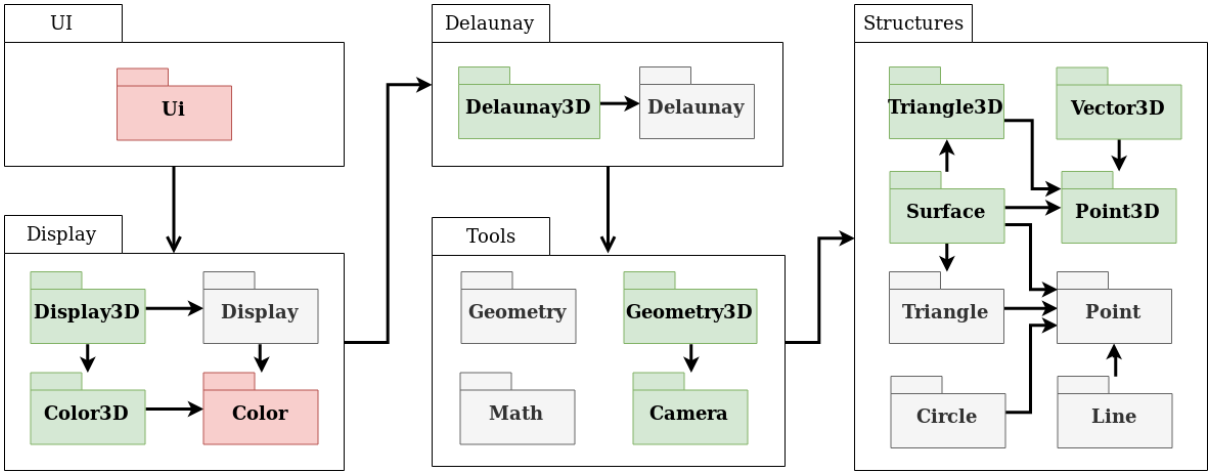


FIGURE 3 – Diagramme de classes avec les extensions.

2 Extensions

Après l'obtention de résultats satisfaisants concernant la triangulation de Delaunay classique, nous avons décidé d'étendre notre travail.

2.1 Extension du diagramme de classes

Nous avons implémenté deux extensions, qui seront détaillées par la suite. Pour y parvenir, nous avons enrichi notre diagramme de classes, comme l'illustre la figure 3.

Sur cette figure, les modules présents auparavant sont représentés en couleur neutre.

Les modules en rouge, `Color` et `Ui`, sont les modules nécessaires à la première extension. Celle-ci est centrée sur la coloration des figures obtenues par notre programme initial.

Enfin, tous les autres modules, représentés en vert, sont ceux utiles pour la seconde extension qui consiste en la projection d'une triangulation sur une surface en trois dimensions.

Davantage de modules ont été nécessaires à cette extension, puisque l'ajout d'une dimension demande de redéfinir les modules de base dans `Structures` ainsi que de nouveaux outils de calcul dans `Tools`. `Delaunay` présente aussi un nouvel algorithme principal, et `Display` propose un nouveau mode d'affichage plus complexe, basé sur l'utilisation d'une caméra.

2.2 Coloration d'une triangulation 2D

2.2.1 Différents modes de coloration

Cette extension, à but esthétique, consiste en la coloration des résultats obtenus. La première idée qui nous est venue est d'utiliser une couleur aléatoire pour chaque élément affiché : sommets, segments et triangles. On peut observer un résultat en figure 4, ce qui donne une mosaïque assez exotique.

Ce résultat n'était pas suffisament satisfaisant, nous nous sommes donc orientés vers l'utilisation de dégradés. On peut observer en figure 5 les résultats obtenus respectivement pour

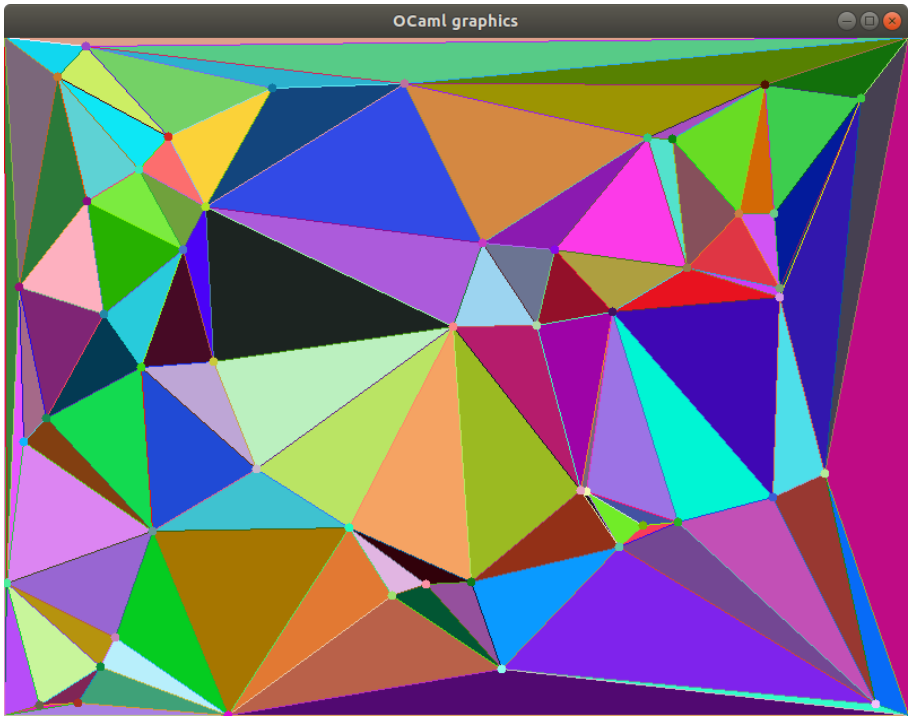


FIGURE 4 – Une triangulation de Delaunay avec des couleurs aléatoires.

un dégradé horizontal, vertical et circulaire.

En terme d'implémentation, nous avons besoin de disposer d'une notion de barycentre de couleur. Chaque dégradé a ses deux couleurs extrêmes comme attributs, et tout objet affiché est coloré par une couleur calculée comme barycentre de ces deux couleurs.

Il nous faut, pour tout objet affiché, le coefficient de ce calcul de barycentre. Pour cela, chaque élément à afficher est assimilé à un point le décrivant au mieux. Cela nous permet de nous ramener dans tous les cas au calcul du coefficient pour un point. Un point est donc assimilé à lui-même ; un segment à son milieu ; et un triangle à son centre de gravité.

Pour un point donné, il ne reste donc qu'à calculer le coefficent à utiliser pour le barycentre de couleur. Ce n'est qu'à cette étape que nous distinguons les différents types de dégradé. Dans le cas du dégradé horizontal, nous disposons d'une abscisse maximale. Le coefficient du barycentre est alors la proportion obtenue entre l'abscisse du point considérée et cette abscisse maximale. Le dégradé vertical se fait de façon analogue en considérant l'ordonnée cette fois-ci. Enfin, pour un dégradé circulaire, nous disposons d'une structure de cercle, donc d'un centre et d'un rayon. Dans ce cas, le coefficient est le rapport entre la distance au centre et le rayon.

Le code tire parti de cette factorisation possible afin de réunir au maximum les différents cas.

(a) Gradient horizontal. (b) Gradient vertical. (c) Gradient circulaire.

FIGURE 5 – Différents modes de coloration utilisant le gradient.

2.2.2 Interface utilisateur

Le dégradé circulaire nous a donné l'idée d'une interaction avec l'utilisateur : donner à celui-ci la possibilité de faire évoluer le centre du cercle utilisé pour le dégradé en fonction de la position de la souris.

C'est cette fonctionnalité qu'implémente le module `Ui`. Pour cela il a suffi d'exploiter les fonctions de la bibliothèque `Graphics` qui interrogent les événements système. En pratique, une triangulation de Delaunay est calculée une fois, puis affichée. L'ensemble des éléments à afficher est mémorisé afin de ne pas recalculer cette information. À chaque déplacement de souris, on actualise l'affichage des éléments mémorisés selon le dégradé circulaire induit par la position de celle-ci.

Cette extension montre que la modularisation du code nous permet d'ajouter des fonctionnalités en ajoutant peu de nouvelles lignes à notre programme.

2.3 Triangulation en 3D

Pour aller plus loin, nous avons implémenté une triangulation en trois dimensions.

Afin d'obtenir une triangulation d'une surface, nous avons choisi de réaliser la triangulation de Delaunay d'un nuage de points du plan, puis de projeter cette triangulation sur la surface.

Une surface est simplement décrite par une application $f : \mathbb{R}^2 \to \mathbb{R}$.

2.3.1 Projection d'une triangulation du plan sur une surface

La projection d'une triangulation du plan sur une surface est obtenue par ajout d'une coordonnée aux points du plan, qui correspond à la hauteur de ce point. Il suffit d'utiliser l'image par l'application f de chaque point de chaque triangle de la triangulation du plan pour obtenir des points en trois dimensions.

2.3.2 Utilisation d'une caméra

Le rendu 3D repose sur l'utilisation d'une caméra. Elle est caractérisée par la position de son centre, et par deux flottants qui donnent son inclinaison dans l'espace.

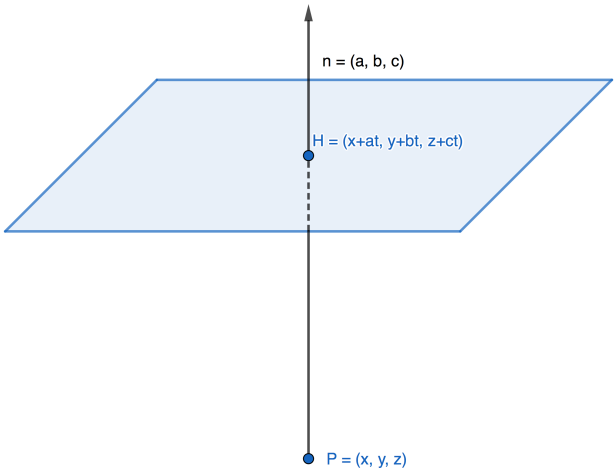


FIGURE 6 – Projection d'un point sur un plan.

Il est alors possible de déterminer le vecteur directeur de la caméra, et le plan qui lui est orthogonal. Ce dernier sera notamment utile pour afficher les points de l'espace, par projection, comme indiqué sur la figure 6.

Nous utilisons la caméra pour changer le point de vue, qui est obtenu par projection des points de la triangulation sur le plan de la caméra, *i.e.* par projection orthographique.

2.3.3 Coloration en fonction de l'altitude

Pour rendre la 3D plus représentative, nous avons décidé de colorer les triangles de la surface en fonction de leurs hauteurs. La coloration reprend la fonctionnalité de gradient du module `Color`, implémentée pour les extensions en deux dimensions (*cf.* 2.2).

À partir d'un triangle en trois dimensions, nous calculons son barycentre dans l'espace. Puis une couleur est attribuée au triangle en fonction de la hauteur de son barycentre.

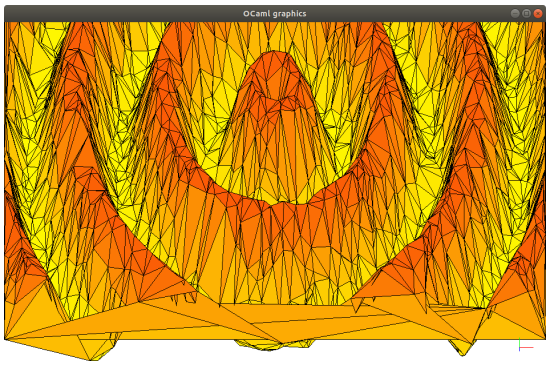
Nous pouvons ainsi obtenir la figure 7, où la couleur claire (resp. foncée) est plus prononcée sur les points de hauteur faible (resp. importante).

Sur les bords de la figure, on peut voir des triangles qui apparaissent devant certains autres : ce sont en fait des effets de bord. Au bord de la figure, les triangles ont tendance à être plus plats, et donc à fausser l'affichage ; cette caractéristique était présente dès la figure 5.

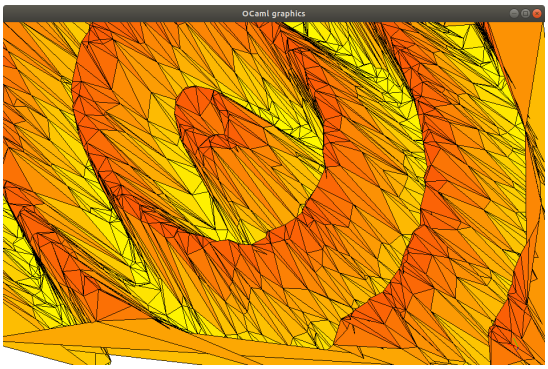
2.4 Conclusion

Nous avons étendu notre travail selon deux axes.

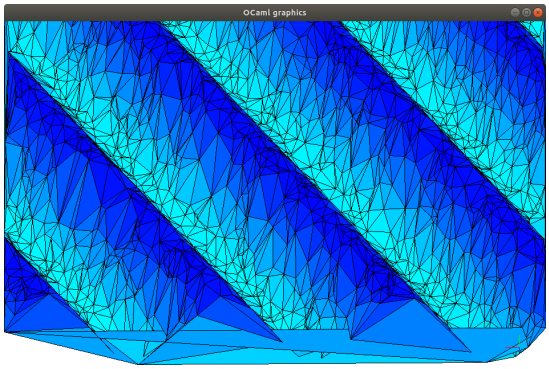
D'une part, nous avons amélioré le rendu visuel de notre triangulation classique en proposant à l'utilisateur plusieurs modes de coloration statique, ainsi qu'une coloration interactive de la triangulation en deux dimensions.



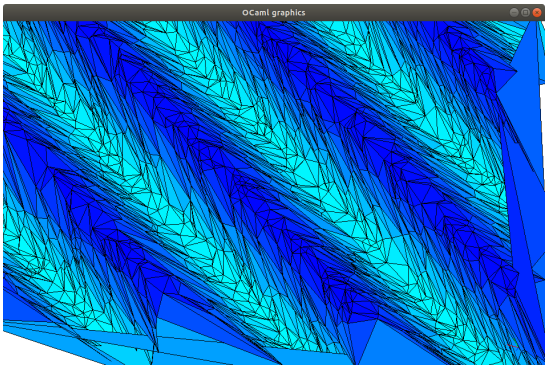
(a) Onde, vue de dessus.



(b) Onde, vue de côté.



(c) Sinusoïde, vue de dessus.



(d) Sinusoïde, vue de côté.

FIGURE 7 – Surfaces de l'espace projetées et affichées dans une fenêtre graphique.

D'autre part, nous avons implémenté un algorithme de triangulation d'une surface de l'espace qui s'appuie sur la triangulation classique. Nous avons également coloré cette triangulation en trois dimensions en fonction de l'altitude des triangles.

3 Conclusion

Finalement, nous avons pris plaisir à aborder ce projet. En lisant sur le sujet, nous avons pu avoir un aperçu des nombreuses applications[2] de la triangulation de Delaunay ainsi que des façons diverses de la mettre en place.[1]

Notre trinôme a bien fonctionné. Notre organisation en amont et une bonne communication nous ont permis de nous impliquer et de nous soutenir dans notre travail. Nous avons ainsi évolué sereinement et avons bien respecté les délais que nous avions estimé en début de projet.

4 Bibliographie

Références

[1] C. Lemaire. Triangulation de delaunay et arbres multidimensionnels, 1997.

[2] C. mohand Oulhadj, I. Belaidi, A. Belaidi, and G. Ishiomin. Application et adaptation de la triangulation de delaunay pour la reconstruction de surfaces. surface reconstruction using and adapting delaunay triangulation, 11 2004.

[3] G. Palmirotta. Triangulation de delaunay, 2015.