



UNIVERSITÉ  
DE LORRAINE

UFR MATHÉMATIQUES  
INFORMATIQUE MÉCANIQUE

## INITIATION À LA RECHERCHE

MASTER 1

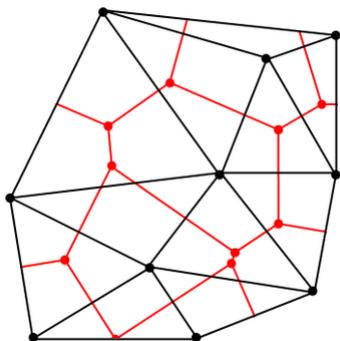
Année 2017-2018

---

# Diagrammes de Voronoï

## Génération d'une carapace de tortue

---



*Auteurs :*  
Samhi JORDAN  
Almeftah TIFAOUT

*Encadrant :*  
M. MICHEL  
LITA, Université de Lorraine



# Sommaire

1	Remerciements . . . . .	1
2	Résumé . . . . .	2
3	Sujet du projet . . . . .	3
4	Organisation du travail . . . . .	3
4.1	Historique des réunions . . . . .	4
5	Introduction . . . . .	5
6	Problématique . . . . .	7
7	Etat de l'art . . . . .	9
7.1	Découvertes . . . . .	9
7.1.1	Diagrammes de Voronoï . . . . .	9
7.1.2	Triangulation de Delaunay . . . . .	11
7.1.3	Relation Triangulation et diagramme de Voronoï . . . . .	12
7.2	Publications marquantes . . . . .	13
7.3	Applications existantes . . . . .	13
8	Propositions, hypothèses . . . . .	14
8.1	Silhouette carapace . . . . .	15
8.2	Sites . . . . .	17
8.3	Triangulation . . . . .	20
8.3.1	Algorithme principal . . . . .	20
8.3.2	Déterminer les triangles à supprimer (DTL) . . . . .	22
8.3.3	Déterminer les triangles à ajouter (NTL) . . . . .	22
8.4	Diagramme de Voronoï . . . . .	24
8.4.1	Boundary representation . . . . .	25
9	Données . . . . .	29
10	Matériel . . . . .	29
10.1	Microsoft Visual Studio 2017 . . . . .	29
10.2	Qt Creator 4.6.0 . . . . .	29
10.3	Git . . . . .	29
11	Expérimentations . . . . .	30
12	Résultats . . . . .	30
13	Synthèse . . . . .	30
14	Conclusion . . . . .	31
15	Perspectives . . . . .	32
A	Sujet d'initiation à la recherche	33

# Table des figures

1	Girafe, Feuille, Ruche . . . . .	5
2	Tortue . . . . .	7
3	1) zones d'influence, 2) Zone de domination, 3) Design, 4) Détection de galaxies, 5) Trajectoire dans zone . . . . .	8
4	Diagramme de Voronoï . . . . .	9
5	Postes d'ambulance d'une ville . . . . .	10
6	Diagramme de Voronoï résultant des postes d'ambulance. . . . .	10
7	Exemple de triangulation de Delaunay . . . . .	12
8	Superposition d'un diagramme de Voronoï et d'une triangulation de Delaunay . . . . .	12
9	Application d'Alex Beutel . . . . .	14
10	Application de Jason Davies . . . . .	14
11	Courbe de Bézier (en noir) et son polygone de contrôle (en vert formé par les points oranges) . . . . .	16
12	Procédure de génération d'une courbe de Bézier . . . . .	16
13	Moitié de contour d'une carapace . . . . .	17
14	Carapaces générées par l'application . . . . .	17
15	Exemple de carapace . . . . .	18
16	Génération uniforme des sites . . . . .	19
17	Point dans carré virtuel . . . . .	19
18	Génération stochastique des sites . . . . .	19
19	Notion d'angle capable . . . . .	21
20	Étapes de l'algorithme de Stéphanie Hahmann . . . . .	21
21	Étapes de la procédure DTL . . . . .	22
22	Étapes de la procédure NTL . . . . .	23
23	Triangulation calculée . . . . .	24
24	Détails diagramme de Voronoï . . . . .	24
25	Diagramme de Voronoï incomplet . . . . .	25
26	Représentation B-rep de la carapace . . . . .	25
27	Deux sortes de faces . . . . .	26
28	Représentation des faces en mémoire . . . . .	28
29	Finalité de l'application développée . . . . .	28

## 1 Remerciements

Tout d'abord, nous tenons à exprimer toute notre gratitude à notre encadrant Monsieur Dominique MICHEL : grâce à son aide précieuse, ce travail n'aurait jamais vu le jour. Il nous a offert la possibilité d'explorer un nouveau monde. Certes, nous avions pu remarquer la structure géométrique assez fine et soigneuse des tissus biologiques dans l'environnement et dans la vie courante avant de travailler sur ce projet, mais nous ne savions pas qu'elle portait le nom de Diagramme de Voronoï. Encore mieux, toujours grâce à lui, nous avons eu l'opportunité de découvrir les domaines d'application de la notion de Diagramme de Voronoï en Informatique notamment dans la recherche combinatoire. Nous lui adressons nos remerciements les plus chaleureux pour ses explications si détaillées et claires, pour son soutien et pour son accompagnement durant tout le projet, cela a été un grand plaisir de travailler avec lui.

Nous tenons à remercier vivement Monsieur Christian MINICH pour son enseignement et ses conseils concernant l'Algorithmique Géométrique. Son aide nous a été très précieuse.

Un grand merci à Monsieur Anass NAGIH : la manière avec laquelle il nous a enseigné la convexité et les graphes ne quittera jamais nos esprits.

Nous n'oublierons jamais les cours d'Algorithmique avec M. Imed KACEM, c'est vrai que nous n'avons pas abordé de PTAS (Polynomial-Time Approximation Scheme) dans notre projet, mais ce que nous avons appris grâce à lui a fait mûrir notre analyse pour trouver des algorithmes afin de résoudre des problèmes. Merci à vous !

Nous remercions de façon générale tous nos professeurs de l'UFR MIM<sup>1</sup>. Merci de nous avoir montré les clés du succès : avoir confiance en soi et en ses capacités, croire en soi et toujours tenter de se dépasser. Merci d'avoir forgé notre esprit analytique lorsque l'on est face à un problème.

Enfin, nous remercions nos camarades de promotion qui ont toujours été là pour nous motiver afin de surmonter la charge de travail de fin d'année.

---

1. Unité de Formation et de Recherche, Mathématiques Informatique Mécanique.

## 2 Résumé

Nous avons commencé par lire de la documentation, des articles et des thèses se référant à la problématique des diagrammes de Voronoï. Nous nous sommes vite rendus compte qu'un tel diagramme est en réalité un graphe dual d'une triangulation attribuée à Boris Delaunay<sup>2</sup>. Nous avons étudié également le processus de génération d'une telle triangulation qui nous faciliterait la génération d'un diagramme de Voronoï. D'un nuage de point, il existe plusieurs algorithmes permettant d'arriver à la triangulation de Delaunay et ainsi générer sans problèmes le diagramme de Voronoï final. Celui que nous avons étudié ayant été découvert par une chercheuse française, membre de l'INRIA<sup>3</sup> à l'université de Grenoble.

Le point de départ de notre travail est donc d'avoir un nuage de point sur lequel appliquer les processus précédents. Nous ne partirons pas d'un nuage de points aléatoire dans le plan. Le sujet du rapport (voir section 3) faisant mention de la génération d'une carapace de tortue, nous devions générer des points plus ou moins contrôlés dans un plan fermé. Se pose désormais le problème du contenant des points. Notre encadrant nous a conseillé d'utiliser des courbes ayant de belles allures pour les contours de la carapace, les courbes de Bézier ont été retenues.

Nous avions donc désormais en main tous les modèles pour développer notre application, le développement devrait se dérouler ainsi :

Contours → Nuage de points → Triangulation → Diagramme de Voronoï

Pour arriver à nos fins nous avons conceptualisé et mis en place des structures pratiques et intéressantes dans le développement de l'application. Notamment la notion de B-rep<sup>4</sup> Nous avons également utilisés des outils facilitant l'intégration de modèle et l'implémentation de nouvelles structures. Après plusieurs tests lors de chaque phase de développement nous avons validé des paliers pour pouvoir passer au suivant.

Finalement, après tant d'heures de travail et de développement nous sommes arrivés à un résultat final satisfaisant à notre niveau. Évidemment l'application peut être améliorée et étendue à d'autres formes ou structures, pourquoi pas ajouter un module "bac à sable" permettant d'interagir directement avec une zone particulière.

---

2. Mathématicien russe du XIX<sup>ème</sup> siècle.

3. Institut national de recherche en informatique et en automatique.

4. Boundary Representation (Représentation de bords).

### **3 Sujet du projet**

Le présent rapport se consacre à l'analyse, par les membres du projet, des structures de Voronoï dans le but de développer un logiciel de génération de textures biologiques. En effet le travail accompli n'a pas pour vocation de révolutionner les structures et/ou algorithmes pré-existants mais d'en faire une étude approfondie pour en avoir une compréhension suffisante pour l'élaboration du logiciel.

Avec notre encadrant nous avons choisi de générer des carapaces de tortues dont le motif n'est rien d'autre qu'un diagramme de Voronoï. Cependant ces derniers se retrouvent dans de nombreuses structures biologiques naturelles tels que les pelages de certains animaux, les carapaces et peaux de certains reptiles et aussi des tissus cellulaires. Le sujet initial disponible en annexe A relate précisément le sujet de base qui a été modifié lors de la première réunion pour ne générer finalement qu'une carapace de tortue. La génération interactive de l'utilisateur pour générer un polygone fermé a été remplacée par la génération de contours de formes à l'aide de courbes de Bézier.

### **4 Organisation du travail**

Le projet étant d'assez grande envergure compte tenu de l'état initial de nos connaissances du sujet, nous avons rapidement contacté notre professeur référent M. MICHEL Dominique<sup>5</sup>. Après avoir eu une idée globale du sujet, nous avons commencé à définir les différentes tâches de chacun en fonction de nos points forts. Pour ce faire, nous nous sommes réunis plusieurs fois afin de bien délimiter nos champs d'action.

M<sup>lle</sup> Almeftah Tifaout étant plus à l'aise avec les mathématiques et la géométrie plane a pris les devants pour s'occuper de la partie la plus théorique afin de constituer le modèle de notre application. M. Samhi Jordan quant à lui s'est occupé de l'aspect graphique et fonctionnel de celle-ci en important, utilisant et adaptant les différentes parties du modèle. Durant tout le temps consacré à notre projet, personne n'a été amené à travailler seul de son côté sur les tâches qui lui ont été affectées, nous avons plutôt opté pour une méthode de travail assez dynamique : nous nous sommes appuyés chacun sur le travail de l'autre en croisant nos recherches et nos idées afin de gagner en efficacité. Les remarques et les commentaires de chacun ont toujours été les bienvenus pour améliorer notre travail commun.

---

5. Maître de conférence au Laboratoire d'Informatique Théorique et Appliquée, Université de Lorraine.

## 4.1 Historique des réunions

**Réunion 1 :** 14 Novembre 2017

**Ordre du jour :** Compréhension du sujet

Première présentation du sujet par M. Michel qui nous a donné une idée globale des diagrammes de Voronoï et de la marche à suivre pour le projet. Nous avons commencé à aborder les thèmes de triangulation, courbes de *Bézier* ainsi que l'algorithmique pour la géométrie. Nous avons ainsi pu voir comment se déroulait étape par étape une formation d'un diagramme de Voronoï à partir d'un nuage de points en utilisant l'algorithme itératif de Stéphanie Hahmann [Hahmann, 2000]. Nous avons abordé l'algorithme de Fortune [Fortune, 1987] qui s'appuie sur un balayage que nous implémenterons si nous avons du temps supplémentaire.

**Réunion 2 :** 29 Janvier 2018

**Ordre du jour :** Approfondissement du sujet

Cette seconde réunion avait pour but de structurer notre démarche pour l'avancement du projet. Nous avons approfondi les thèmes géométriques se rapportant aux nuages de points pour en générer de manière stochastique<sup>6</sup>. Ensuite nous avons vu en détail comment former une courbe de *Bézier* à l'aide de points de contrôle qui nous servira de silhouette pour notre carapace. Nous avons abordé également une notion importante pour la formation d'un diagramme, il s'agit de faces qui sont représentées sous forme de liste mémoire.

Nous avons défini les phases de notre projet ainsi :

1. Génération de la silhouette de la carapace.
2. Génération des sites à l'intérieur de la silhouette.
3. Génération de la triangulation à l'aide des sites.
4. Génération du diagramme à l'aide de la triangulation.

En sortant de cette réunion nous avions toutes les cartes en main pour mener à bien notre projet d'initiation à la recherche.

Nous avons eu également de nombreuses discussions fructueuses par courriel avec notre encadrant pour mettre le point sur notre état d'avancement du projet ainsi que pour répondre aux différentes questions du binôme. Il a su nous conseiller et nous mettre dans la bonne direction et a toujours été à l'écoute lorsque nous en avions besoin.

---

6. Aléatoire contrôlé.

## 5 Introduction

Le présent travail s'inscrit dans le cadre de l'unité d'enseignement intitulée « Initiation à la Recherche » proposée à l'Unité de Formation et de Recherche Mathématiques Informatique et Mécaniques de Metz pour l'année universitaire 2017/2018. Ce rapport porte sur les travaux effectués par les deux membres du groupe, Samhi Jordan et Almeftah Tifaout, étudiants en première année du Master Informatique, au sujet de la génération de textures biologiques, notamment des carapaces de tortues, à l'aide des structures de Voronoï. Nous allons donner une vue globale du travail accompli en suivant et en référençant la structure du rapport afin de guider le lecteur.

C'est avec un grand intérêt que nous avons opté de travailler sur ce sujet sous l'encadrement de M. Dominique Michel. Un choix qui nous a été immédiat pour ses aspects algorithmiques et géométriques qui coïncident avec les points communs de nos personnalités. La lecture du sujet a suscité en nous la curiosité d'en apprendre davantage. M. Michel a accepté chaleureusement de nous encadrer, et nous a donc offert l'occasion de découvrir une nouvelle notion mathématique.

Intéressons nous maintenant au cœur du sujet en nous posant cette question : Quel pourrait être le point commun entre une girafe, une feuille d'arbre et une ruche d'abeilles ?



FIGURE 1 – Girafe, Feuille, Ruche

Comme le montrent les images de la figure 1, mis à part les tissus biologiques que l'on retrouve dans la nature bien évidemment, les motifs que nous pouvons voir sur la peau des girafes, sur la feuille d'arbre et sur la ruche d'abeilles ne semblent pas être générés du hasard. Cette structure fascinante qui varie d'un tissu à un autre, a attiré toute l'attention de certains chercheurs. Elle porte le nom de Diagramme de Voronoï.

La notion de diagramme de Voronoï intervient non seulement sur l'algorithmique géométrique, mais aussi sur l'optimisation combinatoire. Les exemples le plus significatif dans ce cas est celui-ci : recherche des plus proches voisins, problème du voyageur de commerce, ainsi que la résolution d'un certain nombre d'applications d'optimisation spatiale telles que la logistique, l'aménagement du territoire, l'emplacement d'une installation... Ce qui fait de ces diagrammes une référence et un recours pour résoudre certains problèmes d'optimisation.

Le sujet de recherche fait, de manière générale, référence à la géométrie algorithmique. Cependant nous allons appliquer cette géométrie algorithmique pour résoudre des problèmes particuliers que sont les diagrammes de Voronoï. Bien entendu, ces derniers sont déjà étudiés depuis les années 1970 en informatique pour diverses applications. Néanmoins l'algorithme ultime n'a pas encore été trouvé, les chercheurs s'obstinent à essayer de gagner encore en complexité, la meilleure étant actuellement  $O(n \log n)$  en temps et  $O(n)$  en espace occupé dans la mémoire.

En terme de généralité, un diagramme de Voronoï peut être vu, dans le plan euclidien, comme une partition de ce plan en régions partant d'un ensemble fini de points (sites ou germes). Chacune de ces régions ne renfermant qu'un point du plan et contenant tous les points plus proches de ce point que toutes les autres régions. Davantage de détails seront donnés dans la section 7 relative à l'état de l'art ainsi que dans la section 6 pour les définitions. Nous nous appuieront sur cette définition générale pour générer un cas particulier de diagramme sur un plan fermé, c'est-à-dire fini. En effet, il s'agit de générer une carapace de tortue en deux dimensions contenant un nuage de points contrôlés qui fera office de base pour générer le diagramme de Voronoï relatif. Voir section 8 pour la génération pas à pas d'une telle carapace.

Ayant fait mention de complexité algorithmique et de géométrie algorithmique, il va de soi que l'efficacité des programmes écrits doit être au rendez-vous. C'est pourquoi ce projet sera entièrement rédigé avec le langage C++, qui une fois compilé offre de meilleures performances que d'autres. Il est également choisi pour sa facilité d'écriture en mode orienté objet. De plus notre application étant graphique nous allons utiliser une librairie graphique très puissante et très performante qui n'est autre que *Qt* pour le C++.

Nos travaux s'appuieront essentiellement sur des travaux existants. Pour ce qui est des définitions formelles nous nous sommes inspirés en grande partie de Franz Aurenhammer [Aurenhammer, 1991] et Thomas Iwaszko[Iwaszko, 2012]. Enfin pour la partie plus technique et algorithmique nous sommes partis d'une découverte de Stéphanie Hahmann [Hahmann, 2000]. En combinant tout cela, nous avons été capable d'émettre des hypothèses quant aux bons résultats que notre logiciel pouvait fournir. Ces dernières n'ont pas toujours été vérifiées car il existe des cas particuliers, notamment lorsque nous travaillions avec l'aléatoire (voir section 12). Finalement, si l'on se passe de certains résultats infructueux, nous sommes plutôt satisfait du rendu final car utilisant des choses existantes nous avons tout de même dû réfléchir et avons eu des idées concluantes, notamment pour les contours de la carapace ou encore la gestion des bords.

Malgré l'existence de nombreuses applications qui permettent déjà de générer des diagrammes de Voronoï, la plupart partent d'un plan considéré comme infini, notre défi étant de délimiter notre zone d'interaction. De plus nous ne la délimitons pas de n'importe quelle manière mais avec des outils mathématiques pointus, les courbes de Bézier. Ce qui ajoute une satisfaction personnelle pour les deux membres du groupe car ce projet regroupe de nombreuses notion mathématiques et géométrique. Ce qui fait que lors de ce projet nous avons appris énormément. Nous avons surtout pris conscience que la gestion de bords en géométrie est une chose ardue pour les informaticiens.

Nous avons clairement défini les missions de chacun dès le début en fonction de nos points forts et en respectant une charge de travail équitable. La partie modélisation et implémentation des outils mathématiques et géométriques a été attribuée à Tifaout Almeftah. La partie implémentation fonctionnelle du modèle et graphique a été attribuée Jordan Samhi. Bien entendu, nos travaux se croisent et avons travaillé main dans la main. Le présent rapport ayant été rédigé par les deux membres du groupe pour une compréhension optimale, rien n'étant laissé au hasard.

Nous commencerons par faire un petit état de l'art pour connaître les principaux travaux existants ainsi que les principales applications existantes. Nous détaillerons par la suite les propositions que nous avons faites pour l'avancement de notre projet ainsi que les détails de certaines réflexions et implémentations. Nous continuerons par décrire les données avec lesquelles nous avons travaillé ainsi que les outils utilisés. Nous ferons un état de nos différentes expérimentations et de leurs résultats afférents. Enfin nous conclurons et avancerons quelques pistes d'évolution pour l'application développée.

## 6 Problématique

Le sujet de notre projet d'initiation à la recherche comme nous l'avons décrit dans la section 3, porte sur l'implémentation d'un logiciel interactif qui génère des carapaces de tortues telle que l'on peut le voir sur la figure 2.



FIGURE 2 – Tortue

La réalisation de ce projet s'est avérée faire appel à plusieurs notions mathématiques et algorithmiques à savoir la génération du diagramme de Voronoï connaissant un ensemble de points. Il va falloir toutes les étudier afin de bien mener nos réflexions et bien avancer le projet.

Les diagrammes de Voronoï sont étudiés et utilisés depuis plus d'une quarantaine d'années dans le domaine de l'informatique. Notamment dans le domaine de la géométrie algorithmique. En effet les chercheurs essaient de trouver le moyen de calculer le plus efficacement un tel diagramme en partant d'un nuage de points. On comprend vite un tel intérêt face à un nuage très conséquent car le temps de calcul peut rapidement "exploser". Ils permettent également de modéliser de nombreuses structures existantes dans la nature pour les comprendre ou faciliter certaines modélisations de phénomènes (voir les exemples ci-après).

- Définition de zones d'influences optimisées pour le commerce (intérêt économique).
- Modélisation de trajectoires en robotique.
- Modélisation de structures en trois dimensions comme des dômes géodésiques.
- Identifications de galaxies.
- Design architecturale (génération de formes chaotiques contrôlées agréable à regarder).
- Dans le sport désormais pour déterminer la zone de domination d'un joueur.

Les illustrations correspondantes à ces exemples sont disponibles sur la figure 3.

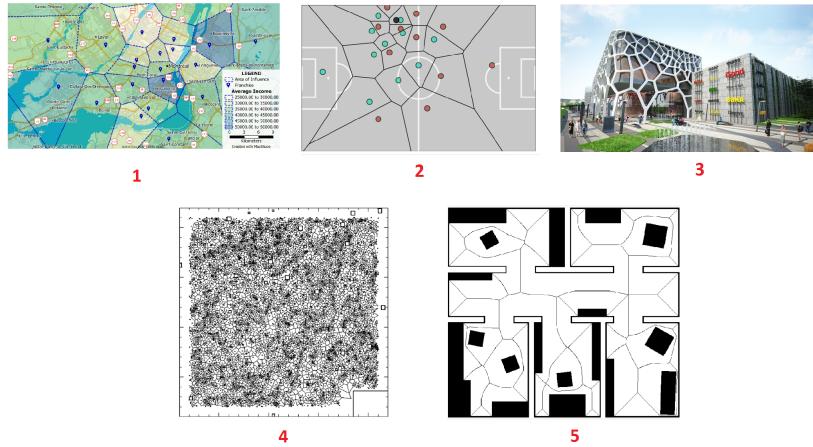


FIGURE 3 – 1) zones d'influence, 2) Zone de domination, 3) Design, 4) Détection de galaxies, 5) Trajectoire dans zone

On peut se demander comment sont générées ces différentes zones, heureusement elles sont définies de manière formelle par les mathématiciens. Cela aide énormément à la mise en œuvre d'algorithme et à la compréhension de leur modélisation. Nous verrons plus loin la définition mathématique pour bien comprendre et avoir en tête ce que représente une cellule de Voronoï.

Dans la littérature, un diagramme de Voronoï définit bien une partition du plan, chaque zone étant définie comme l'ensemble des points les plus proches d'un site donné que de tous les autres sites.

Comme nous avons pu le remarquer sur les images des tissus biologiques précédentes, la surface, qu'elle soit celle de la peau d'une girafe ou celle de la ruche d'abeilles, comporte plusieurs zones délimitées par un contour. Nous nous sommes aperçus que ces zones sont uniformément réparties dans le cas de la ruche d'abeilles, nous avons l'impression qu'il s'agit d'un hexagone qui se répète tandis que ces zones ont une forme différente pour les peaux des girafes et également pour la feuille d'arbre. Ceci a donné naissance à plusieurs questions cruciales : qui est-ce qui détermine la forme de ces zones ? Qu'est-ce qui garantit que les zones que nous aurons à générer auront bien les mêmes formes des zones qui apparaissent sur les carapaces de tortues dans la nature ?

Grâce aux différentes explications qui nous ont été fournies par notre encadrant, nous avons compris que le diagramme de Voronoï, comme le montre la figure 4 comporte à sa génération des cellules ouvertes (infinies), nous nous sommes donc demandé comment le générer d'abord dans le cadre de notre logiciel, puis comment le délimiter ensuite dans un contour fermé (le contour extérieur d'une carapace de tortue).

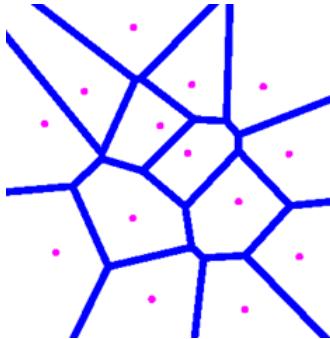


FIGURE 4 – Diagramme de Voronoï

Nous présenterons dans la suite de ce rapport toutes les définitions formelles et mathématiques des notions abordées dans ce travail.

## 7 État de l'art

Nous allons dans cette section faire un état des connaissances actuelles dans le domaine de l'informatique sur les diagrammes de Voronoï. Nous allons mentionner et décrire certains articles existants à propos de ce sujet. Nous noterons que nous avons pu nous inspirer et nous appuyer de certains d'entre eux pour ce projet d'initiation à la recherche.

### 7.1 Découvertes

#### 7.1.1 Diagrammes de Voronoï

Les diagrammes de Voronoï doivent leur nom à un mathématicien ukrainien nommé Gueorgui Voronoï qui vécut dans l'ancien empire Russe. Il était également professeur à l'université de Varsovie. Malheureusement il mourut à l'âge de 40 ans à cause de maladies cependant il a laissé ces fameux diagrammes comme héritage dans le monde scientifique.

Commençons par introduire un exemple :  
 Étant donné les postes d'ambulances dans une ville, en cas d'urgence (modélisée par le point rouge de la figure 5), à partir de quel poste l'ambulance devrait partir ?

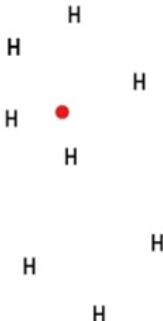


FIGURE 5 – Postes d'ambulance d'une ville

Soit  $S$  : l'ensemble de  $n$  points du plan, ( $n \geq 3$ ), ici  $n = 8$  pour 8 postes d'ambulances. Chaque région de Voronoï est un polygone convexe. Il s'agit en effet de l'intersection d'ensembles convexes : des demi-plans. Le diagramme de Voronoï de  $n$  points partitionne donc le plan en polygones convexes : chaque polygone contient exactement un point  $H$  de  $S$ , et chaque point d'un polygone est plus près de son point central que de tout autre.

Après avoir construit le diagramme de Voronoï comme décrit précédemment avec l'intersection des demi-plans, nous obtenons le résultat de la figure 6.

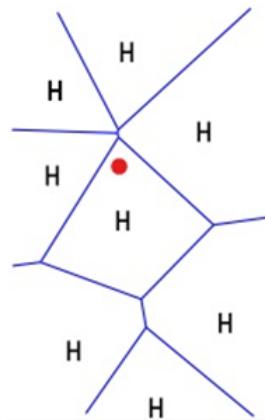


FIGURE 6 – Diagramme de Voronoï résultant des postes d'ambulance.

Et donc, nous pouvons conclure que l'ambulance  $H$  qui est située dans la même zone que le cas d'urgence devrait intervenir pour secourir. Ceci, parce qu'elle représente la plus proche ambulance du lieu où a eu le cas d'urgence.

**Définition (Région de Voronoï)** Nous allons définir formellement ce qu'est la zone<sup>7</sup> d'influence d'un site<sup>8</sup> ainsi :

Soit  $P = \{p_1, \dots, p_n\}$  un ensemble de points d'un plan appelé nuage de points. Nous associons à chacun de ces points  $p_i$  une région  $R$  de Voronoï définie comme l'ensemble des points étant plus proches ou aussi proches du point considéré que de tout autre point  $p_j$  ( $p_i \neq p_j$ ) comme suit :

$$R(p_i) := \{x \in \mathbb{R}^2 \mid \forall p_j \in P, \|x - p_i\| \leq \|x - p_j\|\}$$

On notera que  $\|x - p_i\|$  est la distance entre le point  $x$  et le point  $p_i$  dans le plan.

Notre travail va consister en la modélisation de nombreuses notions de géométrie pour être capable de générer un diagramme de Voronoï efficacement et respectant cette définition.

Dans les nombreuses recherches que nous avons dû faire, nous nous sommes aperçus qu'il existe une autre manière de construire le diagramme de Voronoï, sans passer ni par la division du plan en des demi-plans ni par la définition mathématique du diagramme. Il s'agit en effet de la triangulation de Delaunay.

### 7.1.2 Triangulation de Delaunay

La triangulation de Delaunay doit son nom à un mathématicien russe connu sous le nom de Boris Delaunay. Ses domaines d'applications étaient l'algèbre générale ainsi que la géométrie des nombres. Il s'est servi des résultats de Voronoï pour inventer sa triangulation, 26 ans après la mort de celui-ci. Il faut aussi savoir que Voronoï était le directeur de thèse officieux de Delaunay.

**Définition (Triangulation)** Une triangulation consiste en un maillage d'un nuage de points du plan en formant des triangles dont les sommets sont les points du nuage. Chaque triangle ne chevauche aucun autre triangle et ne peut avoir en commun avec un autre triangle qu'un point du maillage ou une arête d'un autre triangle.

**Définition (Triangulation de Delaunay)** La méthode de Delaunay pour construire sa propre triangulation respecte la définition précédente à cela près qu'elle y ajoute une contrainte. En effet, si l'on considère un triangle  $T$  de la triangulation, alors aucun autre point du nuage n'est à l'intérieur du cercle circonscrit de  $T$ . En d'autres mots, si parmi les triangles construits, chacun de leur cercle circonscrit est vide alors il s'agit d'une triangulation de Delaunay. Cette triangulation permet d'éviter des triangles très allongés en maximisant l'angle minimal de l'ensemble des angles des triangles construits. Nous pouvons voir un exemple sur la figure 7.

---

7. Région de Voronoï.

8. Point du nuage de points.

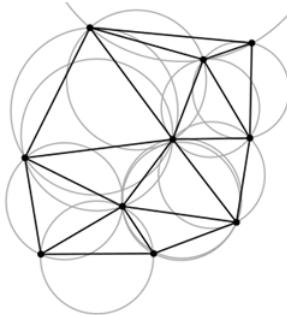


FIGURE 7 – Exemple de triangulation de Delaunay

### 7.1.3 Relation entre triangulation de Delaunay et diagramme de Voronoï

La triangulation de Delaunay d'un ensemble discret  $P$  de points est le graphe dual du diagramme de Voronoï associé à  $P$ .

**Passage de la triangulation de Delaunay au diagramme de Voronoï** Les sommets du diagramme de Voronoï sont les centres des cercles circonscrits des triangles de la triangulation de Delaunay. Les arêtes du diagramme de Voronoï sont sur les médiatrices des arêtes de la triangulation de Delaunay.

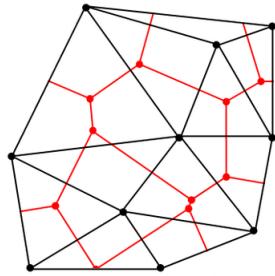


FIGURE 8 – Superposition d'un diagramme de Voronoï et d'une triangulation de Delaunay

La figure 8 représente la superposition d'un diagramme de Voronoï (en rouge) et de sa triangulation de Delaunay (en noir).

**Passage du diagramme de Voronoï à la triangulation de Delaunay** Chaque germe du diagramme de Voronoï constitue un sommet dans la triangulation de Delaunay. Ces sommets sont reliés entre eux par une arête si et seulement si les cellules sont adjacentes.

## 7.2 Publications marquantes

Nous ne pouvons pas nous empêcher de citer le formidable article qu'a écrit M. Aurenhammer Steven [Aurenhammer, 1991] traitant des diagrammes de Voronoï. Il s'agit très justement d'un état de l'art consacré à ces diagrammes datant de 1991. Il commence par confronter les différentes visions des biologistes, des mathématiciens et des informaticiens. Bien entendu le reste de l'analyse est consacré à l'algorithme car il est lui-même informaticien.

La seconde publication qui nous a été extrêmement utile afin de comprendre certaines notions est le rapport de thèse de M. Iwaszko Thomas [Iwaszko, 2012]. Il s'agit d'une généralisation des régions de Voronoï. Évidemment l'ensemble de sa thèse ne nous a pas été d'une grande aide car elle traite de sujets plus vastes que notre simple génération de diagrammes. Il y décrit également des applications dans le monde de l'informatique nous ayant permis de toucher du doigt la réalité des applications.

En ce qui concerne la triangulation préalable au calcul du diagramme de Voronoï à proprement parler, nous nous sommes appuyés sur les travaux de Stéphanie Hahmann [Hahmann, 2000] nous permettant de la construire de manière itérative en ajoutant les points un par un dans le plan. Cela nous a permis de travailler sur un algorithme plus simple que celui de Fortune [Fortune, 1987].

## 7.3 Applications existantes

Il existe une multitude d'applications pour générer des diagrammes de Voronoï, la plupart et les meilleures parmi celles que nous avons retenues sont des applications web. Il faut aussi garder à l'esprit que de gros logiciels de cartographie ainsi que Matlab permettent de générer de tels diagrammes.

Nous commencerons par décrire l'application d'Alex Beutel<sup>9</sup>, chercheur spécialisé dans l'apprentissage neuronal mais qui a également une spécialisation dans le calcul géométrique pour la modélisation de terrains. Son application est disponible sur son site web<sup>10</sup> et est très simple, en effet il s'agit d'une surface plane sur laquelle nous avons simplement à ajouter des points, le diagramme étant construit automatiquement au relâchement du clic de la souris (voir figure 9).

Le type d'algorithme qu'il utilise pour son application n'étant pas mentionné sur la page web de son application, nous avons décidé de le contacter. Il nous a rapidement répondu en nous envoyant un article<sup>11</sup> qu'il a publié avec des collègues à lui. En lisant l'article nous nous sommes rendu compte qu'il n'a pas utilisé le même type d'algorithme que nous, en effet il a préféré un algorithme récursif qui partitionne en plusieurs cases le plan.

9. Senior Research Scientist, Google.

10. <http://alexbeutel.com/webgl/voronoi.html>.

11. Natural Neighbor Interpolation Bases Grid DEM Construction Using a GPU.

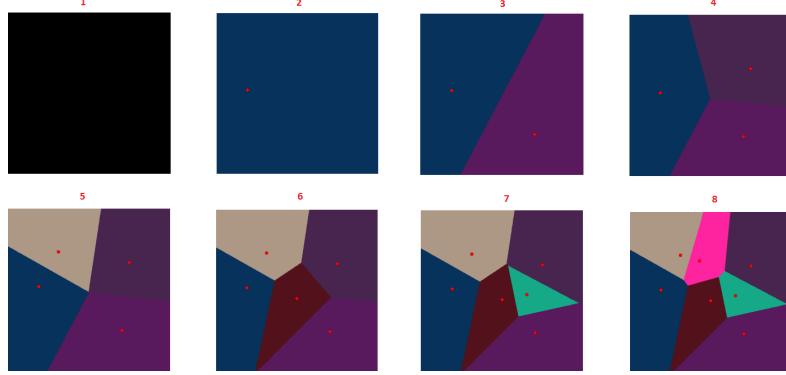


FIGURE 9 – Application d’Alex Beutel

Nous souhaitons également mentionner le joli travail de Jason Davies<sup>12</sup> qui a développé une application permettant de générer une sphère avec des sites sur sa surface. Il est possible de rajouter des sites, le diagramme se forme instantanément, il est également possible de visionner la triangulation de Delaunay associée ainsi que tous les cercles circonscrits de ces triangles (voir figure 10). Pour sa part il a également construit la triangulation de manière récursive et a ainsi déduit le diagramme de Voronoï.

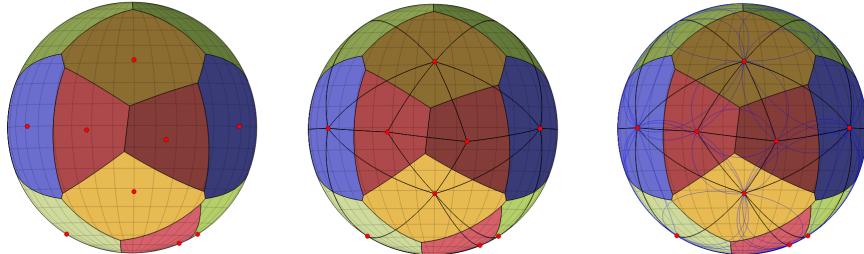


FIGURE 10 – Application de Jason Davies

## 8 Propositions, hypothèses

Nous avons ordonné notre travail et avons suivi l’ordre des tâches défini lors de la seconde réunion avec notre encadrant (voir section 4.1). Chacune des quatre étapes faisant appel à des notions différentes, nous allons mettre en évidence leur déroulement séparément.

---

12. Ingénieur logiciels anglais.

## 8.1 Silhouette carapace

Derrière notre silhouette de carapace se cache en réalité la notion de courbe de Bézier. Nous avons besoin de générer de telles courbes pour avoir des contours harmonieux.

Le mécanisme des courbe de Bézier a été découvert par Pierre Bézier<sup>13</sup> en 1962. Elles remplacent les fameuses courbes dites splines qui présentent de nombreux défauts dûs à leur dépendance au repère considéré. Avant de définir formellement les courbes de Bézier, nous allons faire apparaître une autre notion essentielle qui servira de base pour les calculs des équations de Bézier, il s'agit des polynômes de Bernstein.

**Définition (polynôme de Bernstein)** Ces polynômes ont été découverts par Sergéï Bernstein<sup>14</sup>, ils permettent entre autres choses d'aider à la génération des courbes de Bézier. Nous omettrons les autres cas d'utilisation de ces polynômes car il dépassent le champ d'application de notre sujet d'étude. Ils sont définis comme suit.

Pour tout degré  $n \geq 0$ , il existe  $n + 1$  polynômes de Bernstein  $B_0^n, \dots, B_n^n$  définis sur l'intervalle  $[0, 1]$ , par :

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

Avec  $\binom{n}{i}$  défini comme le coefficient binomial.

On notera que ces polynômes possèdent plusieurs propriétés importantes permettant de prouver efficacement que des fonctions polynomiales peuvent approcher des fonctions continues définies sur des segments. Cela étant hors propos nous ne détaillerons pas ces propriétés que l'on peut consulter dans le théorème de Weierstrass.

**Définition (courbes de Bézier)** Soit  $\Gamma = (P_0, \dots, P_n)$  un ensemble de points appelés points de contrôles (l'ensemble de ces points forment également ce que l'on appelle un polygone de contrôle), alors la courbe de Bézier relative à ces points est définie comme suit.

$$P(t) = \sum_{i=0}^n B_i^n(t) \cdot \mathbf{P}_i$$

Avec  $t \in [0, 1]$  et  $B_i^n$  un polynôme de Bernstein.  $P(t)$  représente l'ensemble des points de la courbe de Bézier.

On notera que le point de départ de la courbe ainsi générée est le point  $P_0$  et le point final est le point  $P_n$ , cependant elle ne passe à priori par aucun autre point étant donné qu'ils servent à définir l'apparence de celle-ci. On peut voir sur la figure 11 une courbe de Bézier avec son polygone de contrôle.

---

13. Ingénieur Arts et Métiers pour Renault.

14. Mathématicien russe du XIX<sup>ème</sup> siècle.

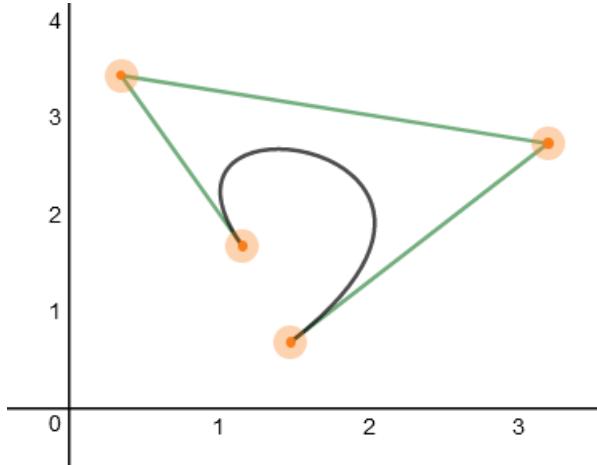


FIGURE 11 – Courbe de Bézier (en noir) et son polygone de contrôle (en vert formé par les points oranges)

Après avoir défini ces deux notions nous nous sommes lancés dans leur modélisation et implémentation au sein de notre application. Pour cela nous avons simplement défini une classe *Bezier* qui va nous permettre de calculer chaque point de notre courbe finale en fonction du paramètre  $t \in [0, 1]$ , cette classe s'appuie elle-même sur une seconde classe *Bernstein* qui permet de calculer le polynôme associé au calcul en cours. La sortie de cette procédure est une liste de points correspondant à la courbe de Bézier voulue. Notons que nous pouvons échantillonner le paramètre  $t$  à notre guise, plus on le fait varier, plus on aura une courbe avec une belle allure. Nous pouvons nous rendre facilement compte du procédé avec la figure 12.

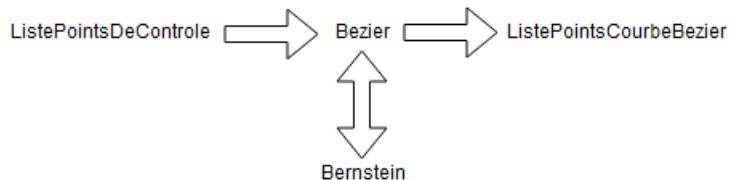


FIGURE 12 – Procédure de génération d'une courbe de Bézier

Se pose maintenant le problème des points de contrôle. En effet sans ces derniers nous ne pouvons pas générer notre courbe car elle repose sur eux. Mais n'oublions pas que notre projet se propose de produire une carapace de tortue, ce qui simplifie grandement le problème. Souvenons-nous, le premier et le dernier point de la courbe correspondent aux premiers et derniers points de contrôle, cela nous fixe deux points. Puis la moitié de la silhouette d'une carapace de tortue étant une simple courbe montante puis descendante (voir figure 13), nous pouvons nous contenter d'un seul point de contrôle supplémentaire. Pour des raisons d'harmonie visuelle la projection de ce point de contrôle sur l'axe des abscisses devra se trouver parfaitement entre les deux projections des deux autres points.

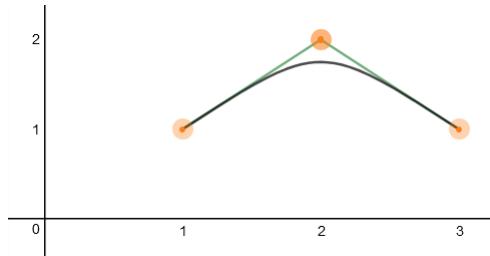


FIGURE 13 – Moitié de contour d'une carapace

On s'aperçoit rapidement qu'une fois cette étape passée, notre silhouette de carapace est terminée. En effet nous nous retrouvons avec une liste de points pour la moitié, il suffit de générer une autre liste de points en prenant l'opposé des ordonnées actuelles. Ainsi il suffit d'assembler les deux listes de points dans notre application et la carapace se retrouve générée.

**Dessin** Nous l'avons mentionné précédemment, pour la partie graphique de notre application nous utilisons l'interface de programmation applicative (API) *Qt*. Celle-ci nous permet de manipuler les objets géométriques simplement. Pour nous simplifier le travail, nous utilisons une classe que nous avons mise en place appelée *Point* qui contient deux coordonnées *x* et *y*. *Qt* nous permet, à partir d'une liste de points, de générer ce qu'il appelle un chemin passant par ceux-ci.

Pour pouvoir générer des carapaces de différentes tailles nous avons ajouté un caractère aléatoire dans la position du point central en ordonné. Cependant nous avons contrôlé sa variation pour que sa valeur reste entre 40% et 90% de la distance entre le premier et le dernier point. Ceci afin d'éviter d'avoir des carapaces trop aplaties ou en cloche. Vous trouverez en figure 14 un exemple de différentes carapaces générées par notre logiciel.

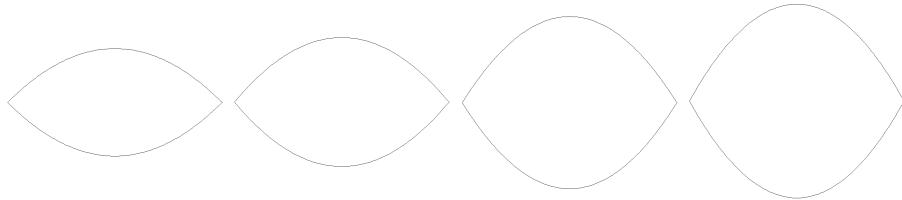


FIGURE 14 – Carapaces générées par l'application

## 8.2 Sites

Les contours de la carapace étant en place, il nous faut désormais générer les points, aussi appelés sites ou germes, à l'intérieur de celle-ci. Ces points serviront de base pour générer la future triangulation. Comment générer efficacement de tels points pour rester réaliste par rapport à une vraie carapace de tortue ? Nous allons étudier cela par la suite.

Le problème étant posé, la solution va découler d'elle-même. En effet, nous pourrions très bien écrire une procédure qui va nous générer aléatoirement des points dans cette carapace, mais le résultat serait évidemment catastrophique dans le sens où, malgré la présence d'un diagramme de Voronoï, nous n'aurons pas une carapace fidèle à la réalité. Nous avons donc observé plusieurs carapaces de tortues pour voir quel schéma pouvait être répété afin de nous guider au mieux. Si l'on observe bien la figure 15 on s'aperçoit que finalement la nature fait bien les choses et nous simplifie la tâche. Nous avons mis en exergue les potentiels sites sur cette carapace en jaune et en bleu afin de bien se rendre compte de ce que nous allons faire.

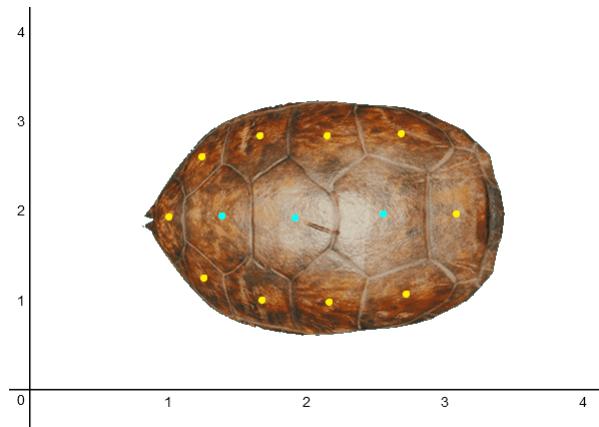


FIGURE 15 – Exemple de carapace

**Remarques** Nous pouvons voir deux choses, premièrement que les sites en bleu sont sur une ligne droite et traversent toute la longueur de la carapace à intervalle plutôt régulier sans passer par les premier et dernier points de contrôle. Deuxièmement on remarque que les autres points suivent simplement le contour de la carapace à intervalle aussi régulier.

Nous allons utiliser cette modélisation pour générer les sites de notre application. L'astuce que nous avons utilisée réside dans le fait de réutiliser les points de contrôles de la génération des contours. En effet pour les points qui suivent le contour, il s'agit simplement après observation d'une seconde silhouette de la même carapace mais plus faiblement échantillonnée. C'est exactement ce que nous avons décidé de faire, en faisant varier le paramètre  $t$  beaucoup moins entre 0 et 1. Ce n'est pas tout, nous ne pouvons pas nous contenter de réutiliser exactement les premiers et derniers points de contrôle, on voit que la nouvelle silhouette avec les points jaunes ne passe pas par ceux-ci, nous les avons donc rapprochés l'un de l'autre d'une dizaine de pourcents de la longueur qui les sépare vers l'intérieur. Pour les points dans l'axe de la longueur de la carapace nous avons également pris quelques points. Pour le dessin à l'aide de Qt cette fois-ci nous n'avons pas dessiné un chemin mais simplement de petits cercles noirs remplis pour chaque point afin de bien visualiser les sites sur l'application. On peut voir un aperçu sur la figure 16.

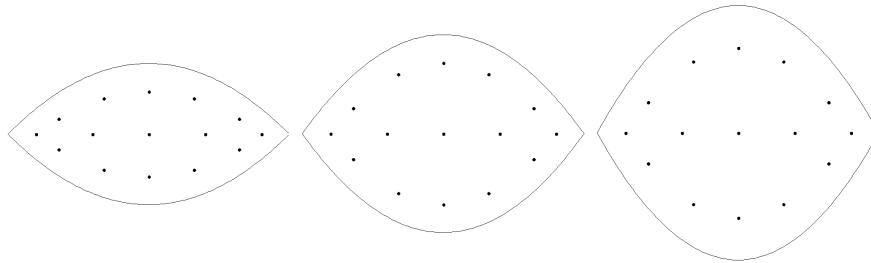


FIGURE 16 – Génération uniforme des sites

Cependant ce n'est pas très satisfaisant car si l'on s'en tient à cela, toutes les carapaces seront les mêmes. Nous ne voulons donc pas de génération purement aléatoire ni purement régulière, il s'agit donc de combiner les deux. En effet l'aléatoire contrôlé (stochastique) va nous permettre de faire varier chacun de ces sites localement. Pour rendre cela possible nous avons raisonné simplement, nous avons considéré que chaque point faisait partie virtuellement d'un petit carré local dont il est le centre. Sur la figure 17 on voit que le point noir peut potentiellement se retrouver n'importe où dans son carré, par exemple aux endroits où les points gris sont situés.

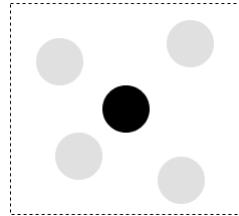


FIGURE 17 – Point dans carré virtuel

En faisant varier les coordonnées des points dans leur carré respectif, nous pouvons donc faire varier tous les points aléatoirement sans détruire la structure régulière globale de ces points. Ainsi nous nous retrouverons à chaque génération avec une carapace différente (voir figure 18).

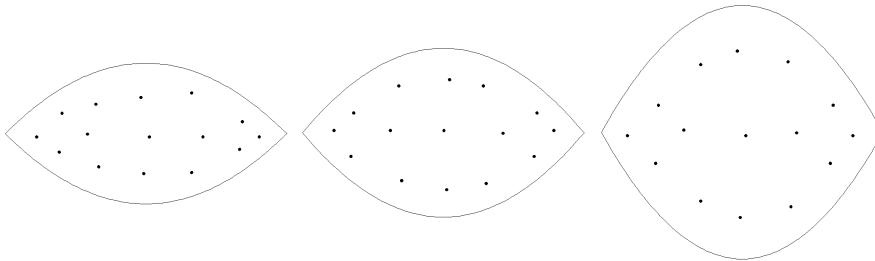


FIGURE 18 – Génération stochastique des sites

## 8.3 Triangulation

La génération d'un diagramme de Voronoï s'appuie sur une triangulation spéciale des points du plan. Il s'agit de la triangulation de Delaunay<sup>15</sup> qui permet d'éviter une triangulation non régulière avec des triangles longs.

Nous allons donc devoir générer une triangulation de Delaunay à partir de la liste de points générée dans la section 8.2. Pour ce faire nous allons implémenter une classe Triangulation avec une méthode prenant en entrée une telle liste et qui produira en sortie une liste de triangles. Comme nous l'avons déjà écrit, nous avons implémenté l'algorithme itératif de Stéphanie Hahmann [Hahmann, 2000]. Ce n'est pas l'algorithme le plus efficace en temps mais un des plus simples à comprendre, c'est pourquoi nous l'avons choisi.

**Description de l'algorithme** Il fait partie de la famille des algorithmes itératifs, dans le sens où à chaque itération il calculera la nouvelle triangulation en fonction de l'ancienne, des points déjà insérés et du nouveau point inséré dans l'itération courante. Cela suppose d'avoir dès la première itération une triangulation correcte et respectant les points à insérer. La solution à ce problème est plutôt simple, elle consiste à générer un rectangle contenant tous les points du nuage et à le diviser en deux triangles adjacents, puis de les supprimer à la fin. L'élaboration d'un tel rectangle est aussi simple, il suffit de prendre les coordonnées des points les plus extrêmes du plan et de créer quatre points correspondant au rectangles avec leurs coordonnées.

### 8.3.1 Algorithme principal

Il s'appuie sur deux listes essentielles :

- Une liste des triangles qui seront ajoutés dans la triangulation courante, notée NTL.
- Une liste des triangles qui seront effacés de la triangulation courante, nommée DTL.

Ceci étant posé, regardons les cinq grandes étapes de l'algorithme principal :

1. Trouver le triangle dans lequel se trouve le point à ajouter à la triangulation courante.
2. Déterminer les triangles de la triangulation précédente à supprimer de la triangulation courante à partir du triangle calculé dans l'étape précédente, en parcourant ses voisins.
3. Déterminer les nouveaux triangles à ajouter à la triangulation courante.
4. Supprimer les triangles précédemment calculés de la triangulation courante.
5. Ajouter le point courant à la liste des points de la triangulation finale.

Trouver le triangle dans lequel se trouve le nouveau point à ajouter à la triangulation courante semble très facile à première vue. Cependant, ceci constitue en lui-même une notion géométrique très importante. Il faut parcourir toute la triangulation et voir, pour chaque triangle, si le nouveau point y est contenu ou pas. Il existe plusieurs méthodes que nous avons explorées lors de nos recherches pour déterminer si un point est contenu dans un polygone (ici un triangle), mais nous avons préféré implémenter

---

15. Boris Delaunay, Mathématicien russe.

la méthode que M. Christian MINICH<sup>16</sup> nous a expliqué lors de l'UE Infographie. Il s'agit d'un algorithme qui s'appuie sur la notion d'*angles capables*.

Un angle capable est l'angle  $\alpha$  formé par un point  $P$  et une arête  $[P_1, P_2]$  comme représenté sur la figure 19.

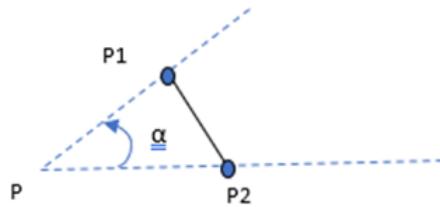


FIGURE 19 – Notion d'angle capable

Défini ainsi, l'algorithme qui vérifie l'appartenance d'un point  $P$  à un triangle  $T_i$  peut être écrit comme suit :

- On parcourt chaque paire de sommets  $[P_i, P_{i+1}]$  de  $T_i$  et on calcule l'angle capable qu'il forme avec  $P$ .
- On calcule  $S$  : la somme de ces angles capables.
- $P$  est à l'intérieur du triangle  $T_i$  si et seulement si  $S$  est égale à  $\pm 2\pi$  (sinon elle est nulle, et il n'y appartient pas).

Comme nous l'avions dit, l'algorithme de Stéphanie Hahmann est relativement simple. Il ne nous reste que deux procédures importantes à détailler, il s'agit de la détermination des triangles à supprimer et ceux à ajouter dans la triangulation courante. Les autres algorithmes géométriques ne seront pas détaillés car ils ne présentent aucun intérêt ici. On peut voir un aperçu des différentes étapes sur la figure 20.

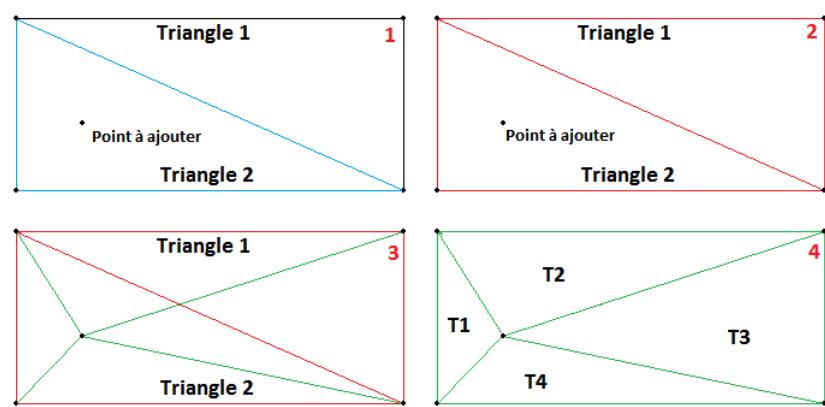


FIGURE 20 – Étapes de l'algorithme de Stéphanie Hahmann

<sup>16</sup>. Maître de conférence au laboratoire de conception, optimisation et modélisation des systèmes de l'Université de Lorraine.

Chaque image correspond à une étape de l'algorithme principale, le triangle bleu dans la première image est le triangle qui contient le nouveau point à ajouter. Sur la seconde on voit les deux triangles en rouge, il s'agit des deux triangles à supprimer déterminés par la procédure DTL. Sur la troisième on retrouve les 4 nouveaux triangles déterminés par la procédure NTL. Enfin sur la dernière on voit la triangulation finale avec la suppression des anciens triangles.

### 8.3.2 Déterminer les triangles à supprimer (DTL)

Cet algorithme reçoit en entrée le point à ajouter dans la triangulation ainsi que le triangle qui contient ce point.

1. On ajoute le triangle d'entrée à la liste des triangles à supprimer.
2. On détermine les voisins du triangle d'entrée qui ne sont pas dans la liste des triangles à supprimer et dont le point d'entrée est dans leur cercle circonscrit.
3. On rappelle cette procédure récursivement avec tous les triangles déterminés dans l'instruction précédente.

A la fin on se retrouve avec une liste des triangles dont le cercle circonscrit contient le point d'entrée. Ce sont ces triangles que nous supprimerons dans l'algorithme principal. Cette procédure est décrite sur la figure 21.

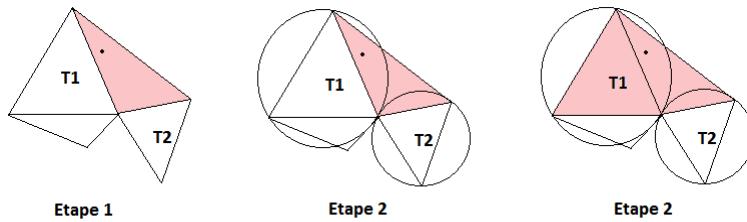


FIGURE 21 – Étapes de la procédure DTL

Sur l'image de gauche on voit le point à rajouter ainsi que le triangle qui le contient, il est donc à supprimer. Ensuite pour tous les voisins de ce triangle on regarde si le point est dans leur cercle circonscrit. C'est le cas pour T1 mais pas pour T2, on ajoute donc T1 à la liste des triangles à supprimer, et ainsi de suite.

### 8.3.3 Déterminer les triangles à ajouter (NTL)

Cet algorithme reçoit en entrée le point à ajouter et la liste des triangles à supprimer calculée précédemment.

1. Déterminer pour chaque triangle à supprimer :
  - (a) Si un des voisins est vide, création d'un triangle ayant pour sommet le point d'entrée et comme arête opposée, l'arête commune entre le triangle à supprimer et son voisin considéré. Ajouter ce triangle à la liste des triangles à ajouter dans la triangulation.
  - (b) Si un des voisins n'est pas vide et que ce voisin n'est pas dans la liste à supprimer, création d'un triangle ayant comme sommet le point d'entrée et

comme arête opposée, l'arête commune entre le triangle à supprimer et son voisin considéré. Mettre à jour la liste des voisins du triangle nouvellement créé. Ajouter ce triangle à la liste des triangles à ajouter dans la triangulation.

2. Pour chaque couple de triangles distincts de la liste des triangles à ajouter à la triangulation, établir leur adjacence si besoin.

A la fin de cet algorithme on se retrouve avec une nouvelle liste de triangles à ajouter à la triangulation courante. Cette procédure est décrite sur la figure 22.

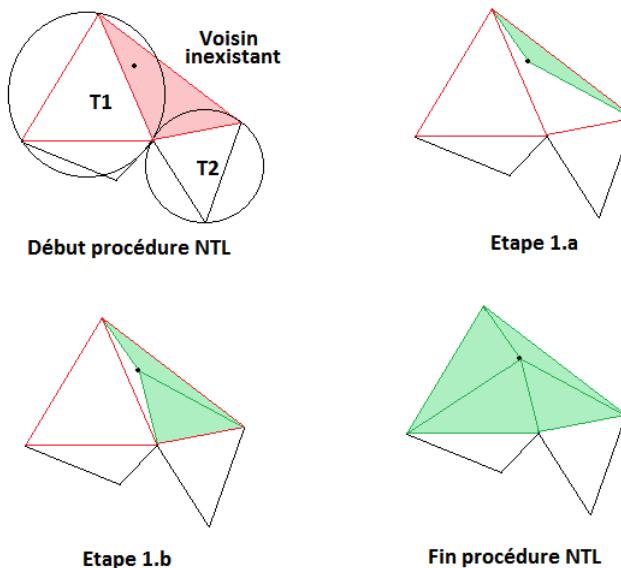


FIGURE 22 – Étapes de la procédure NTL

On considère le triangle rouge à supprimer et T1 son voisin aussi à supprimer car le point à ajouter est dans son cercle circonscrit. Le triangle T2, voisin du triangle considéré n'est pas à supprimer car le nouveau point n'est pas dans son cercle circonscrit.

L'étape 1.a consiste à créer un triangle avec les voisins vides (inexistants), l'étape 1.b consiste à créer un triangle avec les voisins qui ne sont pas à supprimer. Finalement à la fin de la procédure on se retrouve avec les quatre nouveaux triangles verts à ajouter à la triangulation courante. L'algorithme ayant continué à considérer le triangle T1 par la suite.

A la fin de notre algorithme principal, nous nous retrouvons avec la liste des triangles générés, la partie *vue* de notre application se charge de les dessiner un par un pour avoir un rendu tel que l'on peut le voir sur la figure 23.

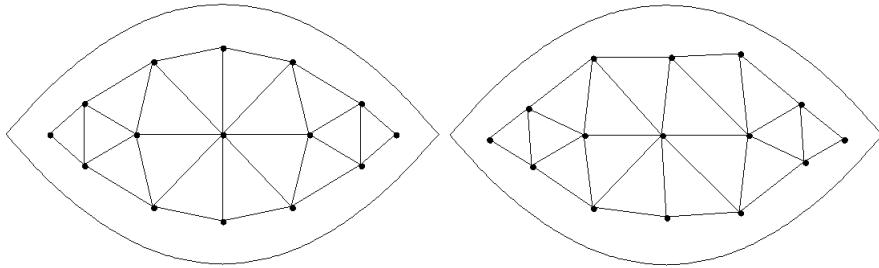


FIGURE 23 – Triangulation calculée

#### 8.4 Diagramme de Voronoï

Nous avons déjà décrit les diagrammes de Voronoï dans la section 6 comme étant l'ensemble des régions de Voronoï du plan calculées à partir du nuage de point du plan et de la triangulation de Delaunay intermédiaire. Un avantage dans la génération de notre diagramme de Voronoï a été ce qu'à décrit Aurenhammer dans l'article étudié [Aurenhammer, 1991] à la page 13. Il spécifie qu'un diagramme de Voronoï et une triangulation de Delaunay sont duals en terme de graphe. Les sommets de Voronoï correspondant aux triangles de Delaunay et les régions de Voronoï correspondant aux sites. En effet si l'on regarde de prêt, le centre de chaque cercle circonscrit de chaque triangle de la triangulation précédemment calculée représente un sommet de Voronoï. Pour notre application il suffit simplement de calculer ces points une fois notre triangulation terminée. Ensuite il suffirait de relier ces points entre eux pour avoir le diagramme souhaité (voir figure 24).

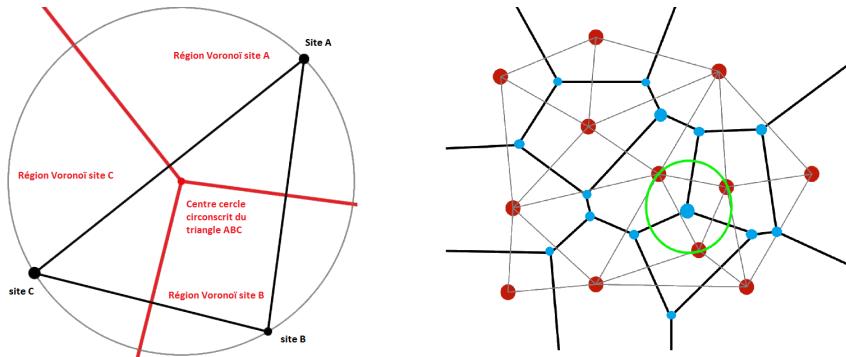


FIGURE 24 – Détails diagramme de Voronoï

Sur l'image de gauche, nous avons trois sites en noir formant un triangle (faisant partie d'une triangulation de Delaunay), on voit que le croisement des trois médiatrices (centre du cercle circonscrit) de ce triangle correspond à un sommet du diagramme de Voronoï, en effet ce point est la jonction entre les trois régions de Voronoï de cette image.

Si l'on s'écarte de cette image et l'on prend un diagramme de plus loin, sur l'image de droite par exemple. On s'aperçoit que cela reste vrai, les points rouges étant les sites et les points bleus étant les sommets du diagramme de Voronoï. Ces sommets correspondent à chaque fois au croisement des médiatrices de chaque triangle, donc des centres des cercles circonscrits de chaque triangle.

Nous avons donc utilisé cette stratégie pour construire notre diagramme de Voronoï. Nous avons défini, à partir de la liste des triangles de notre triangulation précédente, les centres des cercles circonscrits de chacun d'entre eux. Ensuite pour chaque centre nous les avons reliés à chaque centre de chaque triangle voisin. Cependant si nous nous arrêtons à cette seule stratégie nous n'allons pas avoir un diagramme complet, voyons pourquoi sur la figure 25.

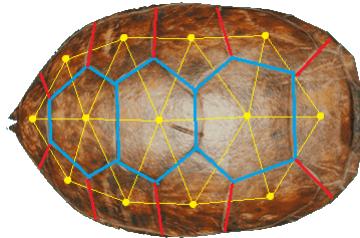


FIGURE 25 – Diagramme de Voronoï incomplet

Les lignes bleues correspondent aux liens entre les centres des cercles circonscrits de chaque triangle. On voit mieux à travers une image les limites de cette stratégie. En effet il nous manque tous les liens rouges avec les contours de la carapace. Pour cela nous allons introduire une nouvelle notion, la représentation des frontières (B-rep).

#### 8.4.1 Boundary representation

Cette notion vient faciliter la prise en compte de contours non réguliers. Mais pas seulement, elle va nous permettre de représenter toute la carapace selon une représentation géométrique. Si l'on considère que notre carapace est terminée, il s'agit donc d'une forme composée de faces (les régions de Voronoï), elles-mêmes composées d'arêtes, elles-mêmes composées de sommets (les centres des cercles circonscrits), voir la figure 26.

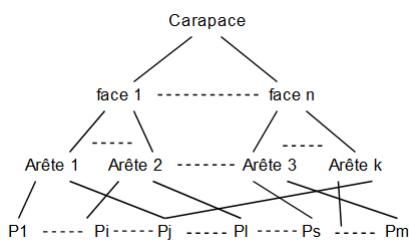


FIGURE 26 – Représentation B-rep de la carapace

Cela permet de mieux représenter les formes composées et ainsi de les formaliser. On voit que finalement l'ensemble est constitué de petites structures indépendantes et permet de travailler sur ces dernières plus facilement au besoin.

On en déduit une structure qui nous permettra de représenter cela en mémoire. Il s'agit d'un graphe dont les sommets seront les centres des cercles circonscrits et les arêtes seront justement les arêtes en ces centres. Nous représenterons également dans notre graphe une liste de faces qui pointeront vers les arêtes de notre liste d'arêtes.

Revenons en aux faces bleues de la figure 25, celles-ci seront des faces composées d'arêtes dont les points sont les centres des cercles circonscrits. Mais comment calculer les faces dont les arêtes sont en rouge ?

Il faut en revenir à notre courbe de Bézier, il s'agit d'un ensemble de points continu. Donc si nous échantillons correctement notre courbe qui représente le contour, il nous sera possible de construire une face composée d'une multitude d'arêtes minuscules vers les bords. Cela nous permettra de nous retrouver avec deux sortes de faces pour notre carapace, voir la figure 27.

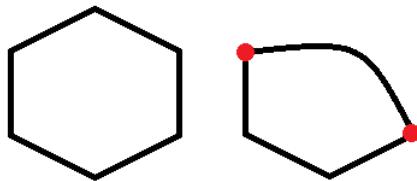


FIGURE 27 – Deux sortes de faces

L'image de gauche représente une face normale constituée de 6 arêtes. La suivante représente une autre face avec cette fois-ci 3 arêtes normales dont les sommets sont les centres des cercles circonscrits et le reste des arêtes forme le contour de la carapace. Il s'agit de minuscules arêtes contiguës.

Un problème subsiste, comment trouver les deux points représentés en rouge sur la figure 27 ? Avec de l'observation on s'aperçoit qu'en réalité ces points sont connectés à des points correspondants aux centres des cercles circonscrits des triangles dont l'arête considérée ne possède aucun voisin (voir figure 25). Ce n'est pas tout, l'arête ainsi formée est également perpendiculaire avec l'arête du triangle qui ne possède pas de voisin étant donné que les arêtes des régions de Voronoï sont les médiatrices des triangles. Une fois les médiatrices calculées, il nous suffit de calculer le point de rencontre entre celle-ci et un point de notre courbe de Bézier pour ainsi relier les deux points calculés.

L'algorithme que nous avons imaginé est le suivant (pseudo-code) :

---

**Algorithm 1** Calculer Diagramme de Voronoï à partir de la triangulation

---

```
for all point ∈ sites do
    Récupérer Triangles : les triangles contenant ce point
    for all T1 ∈ Triangles, T2 ∈ Triangles do
        if T1 ≠ T2 ET T1 voisin avec T2 then
            Créer une arête entre leur centre de cercle circonscrit
        end if
    end for
    if nombre d'arête précédemment créées == nombre de Triangles then
        {Face fermée, il s'agit des points centraux}
        Créer la face avec les arêtes
    else
        {Face ouverte, il s'agit des point externes}
        Récupérer les triangles qui n'ont pas 3 voisins {triangles côté extérieur}
        Récupérer leur centre de cercle circonscrit
        Tracer les médiatrices passant par ces centres avec leur triangles respectif avec
        les faces sans voisin. {côté carapace}
        Récupérer l'intersection de cette médiatrice avec le contour
        Créer une arête entre le centre et cette intersection
        Créer autant d'arête qu'il y a de couple de points contigus entre les deux points
        d'intersection avec le contour.
        Créer la face avec ces arêtes.
    end if
end for
```

---

Cet algorithme nous permet de générer des diagrammes de Voronoï en temps réel dans notre application.

Nous nous retrouvons donc à la fin avec un graphe muni de sommets et d'arêtes. Mieux encore nous avons en notre possession une liste de faces représentant les régions de Voronoï qui s'appuie sur les arêtes de ce graphe, elles-même s'appuyant sur les sommets du graphe.

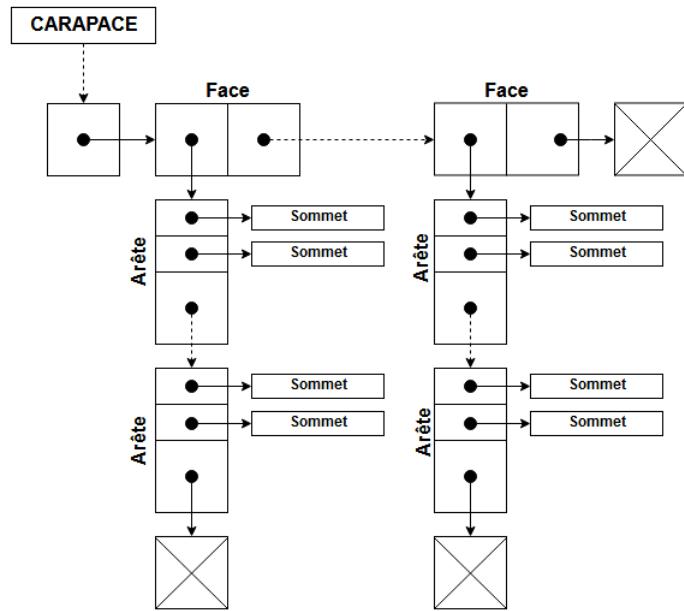


FIGURE 28 – Représentation des faces en mémoire

Sur la figure 28 on peut voir réellement à quoi ressemble notre liste de faces en mémoire. Les liens en pointillés représentent une coupure dans la liste car il peut y avoir plus de faces et d'arêtes mais par manque de place nous représentons les chaînons manquants de cette manière. Il s'agit en réalité d'une liste de listes, avec notre carapace de départ ayant un lien vers la liste de faces car c'est sa représentation. On respecte donc notre représentation en *B*-rep de notre carapace, qui est ainsi formée de faces en deux dimensions.

Finalement notre application a bien évolué en marquant des étapes dans son développement, nous sommes désormais fiers du résultat qui est visible sur la figure 29.

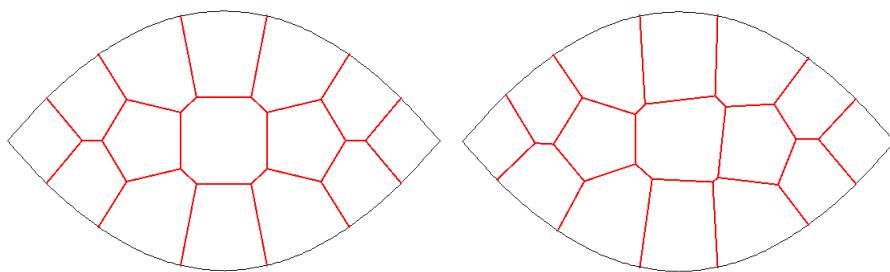


FIGURE 29 – Finalité de l’application développée

## 9 Données

Nous ne traitons aucune donnée externe à notre application dans l'état actuel. Nous pourrions envisager par la suite de définir un format de fichier qui permettrait de charger en mémoire une carapace avec des sites déjà définis dans ce fichier. Ainsi il nous suffirait simplement de calculer la triangulation associée ainsi que le diagramme de Voronoï associé.

Par soucis d'apprentissage et d'utilisation de certains outils mathématiques, nous avons préféré générer nos propres données. Il s'agit des contours des carapaces et des points à l'intérieur de celle-ci. Ces données serviront d'entrée pour l'algorithme de génération de la triangulation de Delaunay que nous avons implémenté et la sortie de cet algorithme deviendra la donnée d'entrée de l'algorithme de génération du diagramme de Voronoï. Nous nous retrouverons finalement avec une liste de faces correspondant aux régions de Voronoï de notre nuage de points.

## 10 Matériel

La plus grande partie de ce projet étant de la réflexion, nous avons passé beaucoup de temps à réfléchir sur papier pour ensuite implémenter nos trouvailles sur de simples ordinateurs. Les principaux outils utilisés étant des environnements de développement intégrés pour le langage C++, nous les avons exploités sur des systèmes Windows 7 et Windows 10.

### 10.1 Microsoft Visual Studio 2017

Ce logiciel est capable de fournir à un utilisateur de nombreux outils pour développer dans plusieurs langages applicatifs ou web. Nous l'avons utilisé pour développer la partie modèle de notre application car il facilite la création de projets et surtout le débogage grâce à la possibilité d'utiliser des points d'arrêts.

### 10.2 Qt Creator 4.6.0

Notre application doit pouvoir afficher les modèles que nous calculons, il nous faut donc une couche graphique pour cela. Nous aurions pu réutiliser des outils graphiques utilisés les années passées avec JAVA ou Python mais nous aurions perdu en flexibilité. Nous avons donc choisi d'apprendre à utiliser Qt. Il s'agit d'une interface de programmation applicative offrant des composants pour l'interfaçage graphique.

La société éditant Qt délivre aussi un environnement de développement intégré nommé *Qt Creator* que nous avons utilisé pour construire toute la partie graphique de notre application. Une fois le modèle défini par un des membres du groupe, il fallait l'importer dans l'environnement Qt creator pour le réutiliser et générer les *items* graphiques correspondant au modèle.

### 10.3 Git

Un projet bien mené ne peut se faire sans l'usage d'un gestionnaire de versions. Git étant simple d'utilisation et intégré à nos environnements de développement, nous l'avons naturellement choisi pour gérer notre projet. Le serveur qui héberge ce projet

est celui de Github, ce qui nous permet d'avoir notre projet à portée à condition d'avoir une connexion Internet.

## 11 Expérimentations

Dans chacune des quatre grandes étapes de notre projet nous avons effectué les mêmes expérimentations. C'est à dire que nous avons commencé par générer des sites de manière uniforme que nous avons appelés *sites parfaits* pour avoir une idée du rendu de la disposition des ces sites, de la triangulation ainsi que du diagramme de Voronoï avec cette configuration. Ensuite nous les avons générés de manière stochastique pour pouvoir comparer les résultats avec la génération régulière. Dans cette dernière configuration le résultat final n'est pas toujours celui attendu pour une carapace de tortue réaliste.

## 12 Résultats

Comme énoncé dans les expérimentations, les résultats étaient très probants pour des générations de sites réguliers. Cependant lors d'une génération de sites stochastique, de temps en temps les points avaient un caractère trop aléatoire dans leur placement, ce qui déformait la triangulation et ainsi le diagramme de Voronoï final. On peut associer ce caractère trop aléatoire à la formation de tissus cancéreux, c'est à dire chaotique et ne permet pas d'avoir une forme finale très contrôlée.

## 13 Synthèse

On pourrait croire que ce projet fut très facile et que notre chemin pour y arriver fut sans embûche d'après la façon avec laquelle nous avons présenté ce rapport. Cependant il faut garder à l'esprit que nous avons relaté les phases d'analyses, de conception, de modélisation ainsi que les résultats. Les phases d'implémentations nous ont donné du fil à retordre car certaines notions sont délicates à manipuler. Notamment certaines listes de listes. Mais l'on sait que pour mener à bien un projet il faut des difficultés et c'est en les surmontant que le projet prend de la valeur. C'est pourquoi nous pensons avoir atteint les objectifs de ce projet d'initiation à la recherche, même si seul le jury peut donner l'avis final.

A la fin de nos développements, nous avons pris un certain recul pour regarder le travail accompli et avons savouré le fruit de notre travail. Nous nous sentons grandis intellectuellement, tant dans la façon de travailler que sur les nouvelles connaissances acquises ainsi que la prise en main de nouveaux outils.

On s'est aussi rendu compte qu'à chaque fois que l'on validait une étape parmi les quatre de départ, on franchissait également un niveau de difficulté croissant. Le défi était au rendez-vous.

Nous n'avons pas la prétention que nos solutions sont les meilleures en termes d'implémentation mais pensons tout de même qu'elle est correcte. Il faut aussi penser que notre application ne génère pas un grand nombre de points dans le plan, la

complexité en terme de temps de calcul ne peut pas se faire ressentir pour le moment. Ce n'est qu'en ajoutant un très grand nombre de points que l'on pourrait se dire qu'il faut optimiser un maximum toutes nos routines.

Nous n'avons fait qu'effleurer la théorie autour des diagrammes de Voronoï. Par exemple nous n'avons pas exploré le fait de pouvoir ajouter une région de Voronoï dans un diagramme existant, ou encore la fusion de deux diagrammes comme le décrit Aurenhammer [Aurenhammer, 1991]. Il est également possible de se lancer dans des diagrammes en trois dimensions ou encore à partir d'un élément en trois dimensions de faire une projection sur le plan euclidien pour ensuite faire le diagramme afférent comme le décrit bien Iwaszko [Iwaszko, 2012] dans sa généralisation des diagrammes de Voronoï.

## 14 Conclusion

Le présent rapport relate des informations que nous avons puisé dans diverses sources que l'on peut voir dans la bibliographie. Notre but étant de comprendre comment former des diagrammes de Voronoï à partir d'un nuage de points. Nous avons suivi un raisonnement logique de succession d'étapes dans le développement de notre projet. Ce qui a permis de mettre à contribution les deux membres du groupe de façon proportionnée.

Les thèmes de géométrie algorithmique étudiés ont été compris par les deux membres du groupe ce qui a aidé à la modélisation, la conception et la mise en place des modèles. Cependant certaines contraintes du sujet ont corsé le développement. Prenons par exemple la partie où il faut relier notre diagramme à notre contour qui est une courbe de Bézier. Nous avons relevé le défi en usant de ruse pour faciliter cela. Pour être plus général, on peut dire que nous avons utilisé certaines astuces dans chaque étape de développement de notre application. Notamment la génération des points dans la carapace ainsi que la gestion des contours avec le diagramme de Voronoï.

Pour ce qui est des résultats, ils sont très positifs même s'il nous arrive de tomber sur des cas où les carapaces ne correspondent pas à la réalité. Disons qu'il s'agit de malformation de carapaces de tortues dues aux déchets présents dans la nature.

Comme nous l'avons déjà spécifié, notre application ne révolutionne rien et n'avait pas pour but de révolutionner quelque chose. L'idée étant de nous faire toucher du doigt certaines notions et d'apprendre à rechercher de l'information comme des chercheurs.

Pour terminer nous aimeraions rajouter que ce projet nous a apporté beaucoup de choses positives pour notre parcours universitaire. Premièrement, la recherche d'informations dans des articles ou des thèses était passionnantes et nous a fait découvrir plus que ce que nous recherchions. Ensuite, nous avons appris à mieux travailler en équipe, en effet le sujet étant de grande envergure il a fallut bien s'organiser et diviser le travail en petits modules pour ne pas perdre le fil. Aussi nous avons appris tant de choses, qu'elles soient mathématiques, géométrique, algorithmique ou encore la manipulation de certains outils informatique. Nous garderons ce projet en mémoire pour notre futur car nous pensons qu'il nous a fait grandir.

## 15 Perspectives

Le stade actuel de notre application est plutôt restreint dans le sens où nous ne pouvons actuellement générer que des carapaces de tortues. Même si ces dernières restent différentes les une des autres, il serait intéressant de continuer à l'améliorer pour pouvoir par exemple choisir de quel animal nous voulons générer un pelage, un tissu ou une carapace. Les silhouettes pouvant être générées entièrement à l'aide des courbes de Bézier, il nous serait possible d'implémenter cette fonction.

Une idée judicieuse pour pouvoir faire des tests rapidement serait de pouvoir interagir directement avec l'application à la manière d'un outil de dessin. Ceci afin de pouvoir définir des bornes en dessinant un contour à la main. Enfin il pourrait être possible de mettre à l'intérieur de ces frontières des points représentants les différents sites. Enfin un diagramme de Voronoï pourrait se générer à partir de ce brouillon. Cela permettrait de tester plusieurs formes ou schémas avant de les implémenter formellement dans le modèle de l'application.

Une autre idée serait de mettre en place un système permettant de générer des structures en trois dimensions avec des sites situés dans l'espace. Ainsi comme l'a décrit Aurenhammer [Aurenhammer, 1991] et l'a repris Iwaszko [Iwaszko, 2012] il serait possible de générer une triangulation en 3 dimensions pour générer aussi un diagramme de Voronoï dans l'espace.

Il pourrait aussi être très utile de pouvoir associer le travail que nous avons achevé avec des autres applications informatiques notamment la génération d'un diagramme de Voronoï dans des problèmes combinatoires. Ces problèmes pourraient être de savoir quelle serait la plus proche ambulance d'un lieu d'urgence comme nous l'avons cité dans l'exemple introductif, ou savoir où est-ce qu'on pourrait construire un nouvel entrepôt de manière qu'il soit le plus proche possible des magasins qui lui sont autour et dont il est fournisseur, ou encore d'introduire notre travail dans le cas d'une ville qui souhaiterait construire une nouvelle résidence pour les étudiants de manière qu'elle soit bien entendu proche des facultés ou des écoles et en même temps la plus proche possible du centre-ville ou des arrêts de bus.

Par manque de temps, de connaissances et de recul nous n'avons pas pu être capables de proposer une solution pour améliorer un des algorithmes étudiés ou encore de proposer un nouvel algorithme plus efficace que ceux existants. Cela pourrait être une très bonne perspective de recherche pour essayer de toujours gagner en complexité algorithmique. Car le cœur de notre sujet d'initiation à la recherche est vraiment ici, l'application étant simplement une mise en œuvre de choses déjà existantes.

## Annexe A

# Sujet d'initiation à la recherche

**Diagramme de Voronoï** De nombreuses structures biologiques tels que tissus cellulaires, peau de reptiles, carapaces de tortues, motifs de pelages d'animaux suivent un modèle géométrique connu sous le nom de diagramme de Voronoï.

**Sujet** Le but de ce TER est la réalisation d'un logiciel de création de textures biologiques simples. L'exécution du programme comprend trois étapes :

1. l'utilisateur du programme définit interactivement un contour polygonal plan fermé
2. le programme « remplit » le contour avec des points qu'il place suivant une certaine distribution
3. le programme remplit le contour avec des cellules correspondant au diagramme de Voronoï des points de l'étape 2

**Distribution de points** Plusieurs modèles de distribution de points seront étudiés : aléatoire, régulier et stochastique. Le but recherché étant le réalisme

Les notions mathématiques et informatiques principales intervenant dans la réalisation de ce TER sont :

- mise en œuvre de graphes
- simulation de variables aléatoires
- algorithmique géométrique plane

**Réalisation** Le développement sera réalisé en C/C++.

**Note** Les aspects mathématiques et géométriques feront l'objet d'une étude bibliographique et seront discutés avec l'encadrant.

**Mots clés** Diagramme de Voronoï, triangulation de Delaunay, génération de texture, génération de nombres pseudo-aléatoires.

**Nombre d'étudiants : 2**

# Bibliographie

- [Aurenhammer, 1991] Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, page Volume 23 Issue 3.
- [Fortune, 1987] Fortune, S. (1987). A sweepline algorithm for voronoi diagrams. *Algorithmica*.
- [Hahmann, 2000] Hahmann, S. (2000). Triangulation de delaunay itérative. *ENSIMAG - DESS IM*.
- [Iwaszko, 2012] Iwaszko, T. (2012). *Généralisation du diagramme de Voronoï et placement de formes géométriques complexes dans un nuage de points*. PhD thesis, université de haute alsace école doctorale Jean-Henri Lambert.