IMN430

Chapitre 2
Techniques de base en visualisation
Partie 1: Triangulation

Olivier Godin

Université de Sherbrooke

12 janvier 2017

Plan de la présentation

Introduction

Introduction

Dans ce chapitre-ci, on s'attarde aux méthodes de base permettant d'**exprimer les données sous forme visuelle**.

Tel que mentionné dans le chapitre d'introduction, les **algorithmes de mappage** qui mettent en correspondance des données avec des primitives graphiques sont au coeur de la visualisation.

Ces algorithmes seront catégorisés **selon le type de donnée qu'ils prennent en entrée**.

Introduction

Ici, une attention toute particulière sera portée aux algorithmes de mappage de **données scalaires**.

Par la suite, une introduction aux représentations de **données vectorielles** sera aussi présentée, mais ce sujet sera approfondi au chapitre 5.

Triangulation

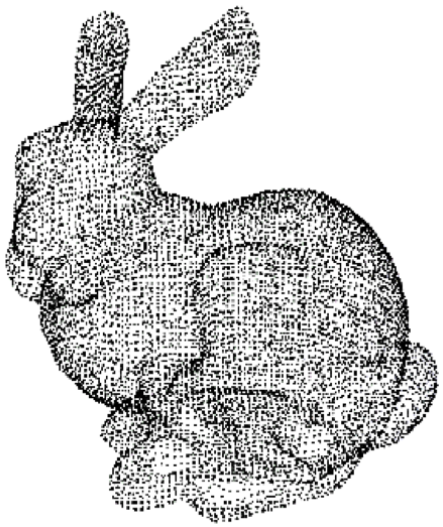
Généralités

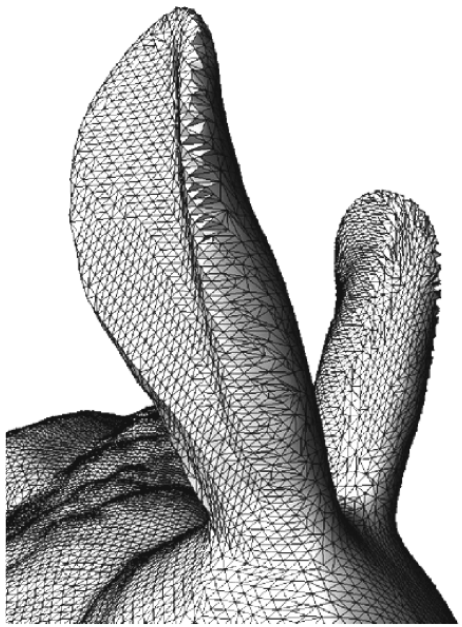
Principe de base

À partir d'un nuage de points 3D, on veut **interpoler la surface qui sous-tend le nuage**. Cela revient à trouver le modèle qui colle le mieux aux données.

Principe de base

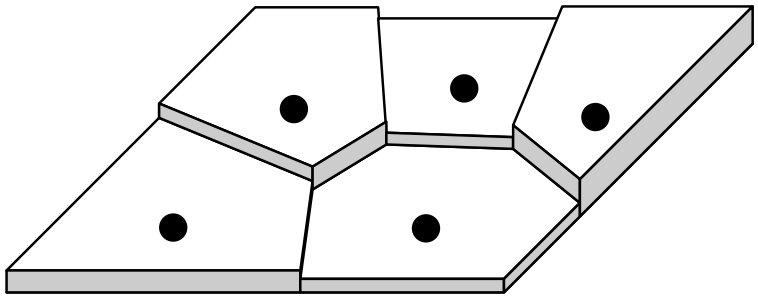
On souhaite interpoler une surface 2D $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ où la valeur $f(x, y)$ est connue **seulement en un nombre fini de points**, que l'on notera P, un sous ensemble du domaine $A \subseteq \mathbb{R}^2$ de f.





Principe de base

Une approcha naïve consiste à **assigner à chaque coordonnées de A la valeur de f au point $p \in P$ le plus près**.

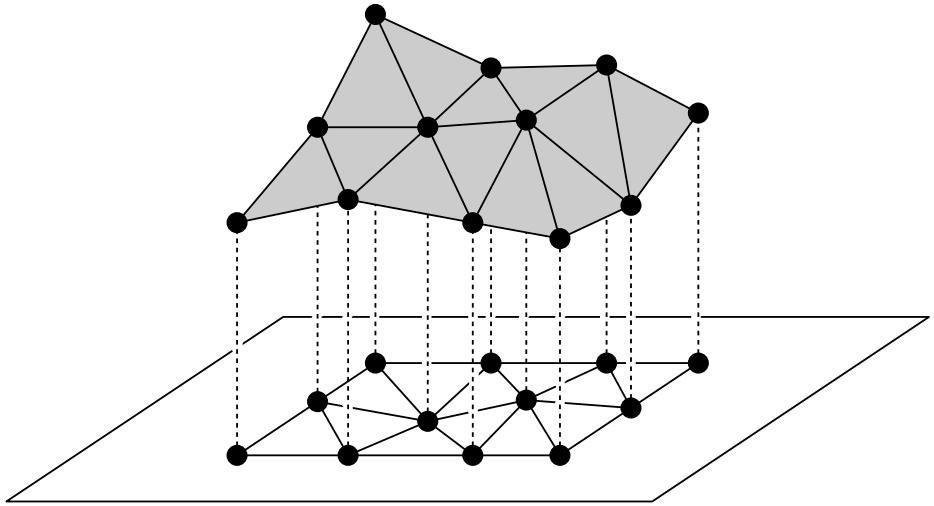


[1]

Une meilleure approche consiste à déterminer une **triangulation de P**. Dans une triangulation, chaque point $p_i = (x_i, y_i) \in P$ est élevé à sa hauteur $f(x_i, y_i)$ et ceux-ci sont **reliés par des triangles en 3D**.

Principe de base

Une triangulation est une **subdivision planaire de** A où chaque division est un triangle dont les sommets sont les éléments de P.



[1]

De plus, dans une triangulation T de P, il est **impossible d'ajouter un segment connectant deux points** sans intersecter un segment existant.

Principe de base

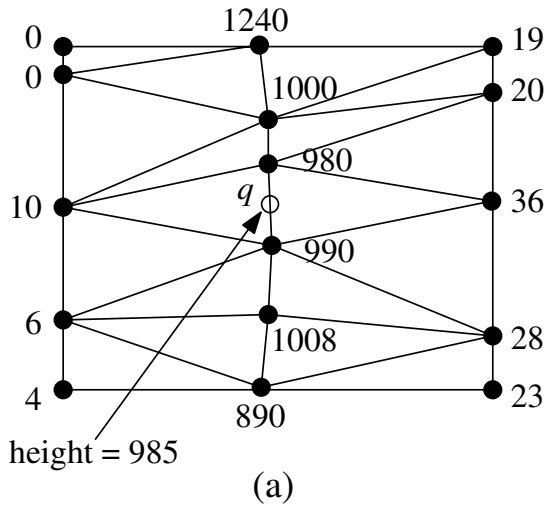
Plusieurs questions restent sans réponse :

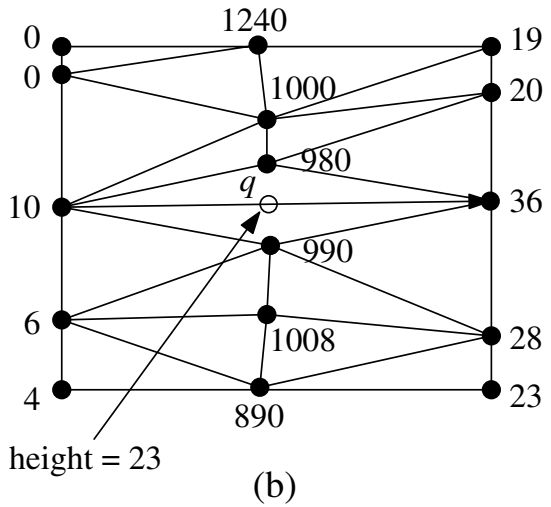
- Est-ce qu'une triangulation pour un ensemble de points P **existe toujours** ?
- Cette triangulation est-elle **unique** ?
- Si elle ne l'est pas, comment savoir si une triangulation est **meilleure qu'une autre** ?

Comme **on ne dispose des valeurs de** $f(x, y)$ **qu'à certaines positions**, toutes les triangulations de P pourraient être percues comme étant **aussi valides les unes que les autres**... Ce n'est toutefois pas le cas.

Principe de base

À titre d'exemple, considérons **deux triangulations d'un même ensemble de points**, où les valeurs numériques associées aux sommets sont les valeurs de $f(x, y)$ à ces points.





[1]

Les deux triangulations sont **identiques à l'exception d'une arête**, mais la triangulation (b) semble moins naturelle.

Principe de base

Comment **convertir cette intuition en un critère strict** qui nous dirait que (a) est une meilleure triangulation que (b) ?

Le problème de la triangulation (b) est que la valeur interpolée au point q est donnée par deux sommets assez éloignés de q. Une telle situation engendrera des **petits angles dans la triangulation** et c'est cela qu'on tentera d'éviter.

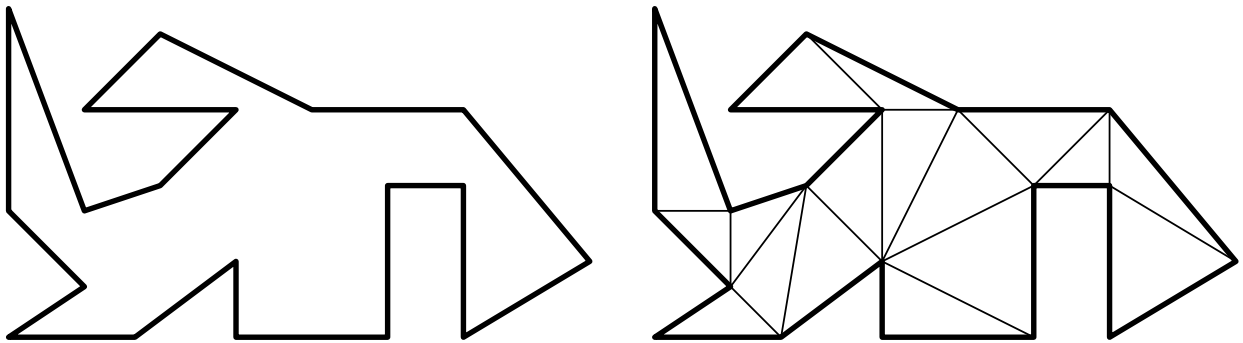
Principe de base

Partant du principe que les petits angles doivent être évités dans une triangulation, on cherchera à **trier celles-ci selon leur plus petit angle**.

Comme il existe un nombre fini de triangulations pour un ensemble de points P donné, on pourra trouver **celle qui maximise l'angle minimal**. Ce sera la triangulation recherchée.

Obtention d'une triangulation

La séparation d'un polygone en triangles à l'aide d'**un nombre maximal de diagonales sans intersection** est une triangulation du polygone.



[1]

Les triangulations ne sont, en général, **pas uniques**.

Obtention d'une triangulation

Le théorème suivant se veut rassurant :

Théorème 3.1

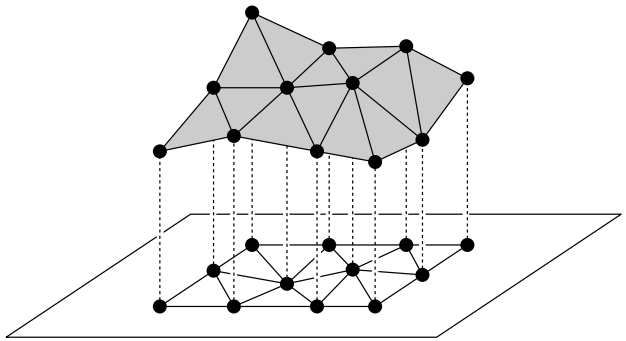
Tout polygone simple admet une triangulation, et toute triangulation d'un polygone simple à n côtés comporte exactement $n - 2$ triangles.

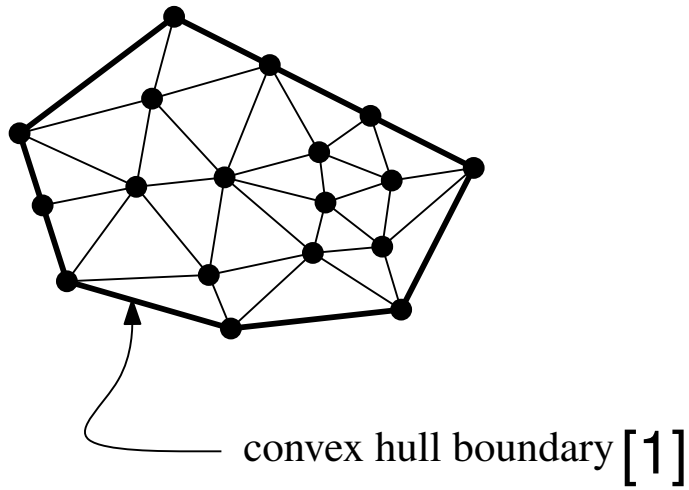
En effet,

- il garantit l'existence d'une triangulation ; et
- il détermine la complexité de celle-ci.

Obtention d'une triangulation

Dans le cas d'un nuage de points 3D, on **projette ceux-ci sur un plan** et l'enveloppe convexe 2D des points projetés est le polygone A dont on cherche la triangulation.





Obtention d'une triangulation

Soit $\mathcal{T}$ une triangulation d'un ensemble de points P composée de m triangles. On s'intéresse aux $3m$ **angles formant les m triangles de** $\mathcal{T}$ triés en ordre croissant.

On note par

$$\alpha_1, \alpha_2, \ldots, \alpha_{3m}$$

cette séquence d'angles, avec $\alpha_i \leq \alpha_j$ si $i < j$.

On appelle $\alpha(\mathcal{T}) = (\alpha_1, \alpha_2, \ldots, \alpha_{3m})$ le **vecteur-angle** de $\mathcal{T}$.

Obtention d'une triangulation

Soit $\mathcal{T}'$ une autre triangulation de P, et soit $\alpha(\mathcal{T}') = (\alpha'_1, \alpha'_2, \ldots, \alpha'_{3m})$ son vecteur-angle. On dira que $\alpha(\mathcal{T}) > \alpha(\mathcal{T}')$ s'il existe une position i, avec $1 \leq i \leq 3m$, telle que

$$\alpha_j = \alpha'_j \quad \text{pour tout } j < i \quad \text{et} \quad \alpha_i > \alpha'_i.$$

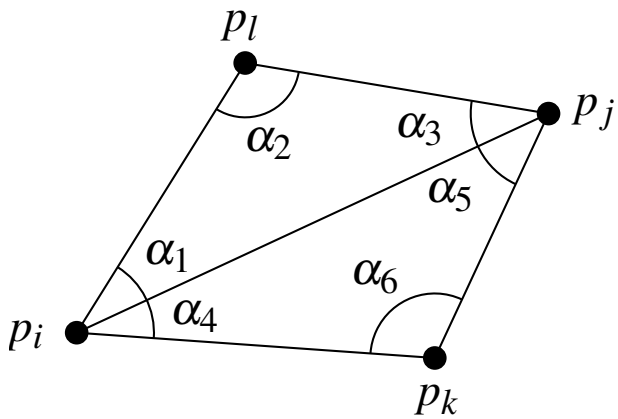
Une triangulation de P est dite **angle-optimale** si $\alpha(\mathcal{T}) > \alpha(\mathcal{T}')$ pour toutes les autres triangulations $\mathcal{T}'$ de P.

Une triangulation angle-optimale **maximisera le plus petit angle** parmi l'ensemble des triangulations possibles.

Obtention d'une triangulation

Considérons une arête $e = \overline{p_i p_j}$ faisant partie d'une triangulation $\mathcal{T}$ d'un ensemble de points P.

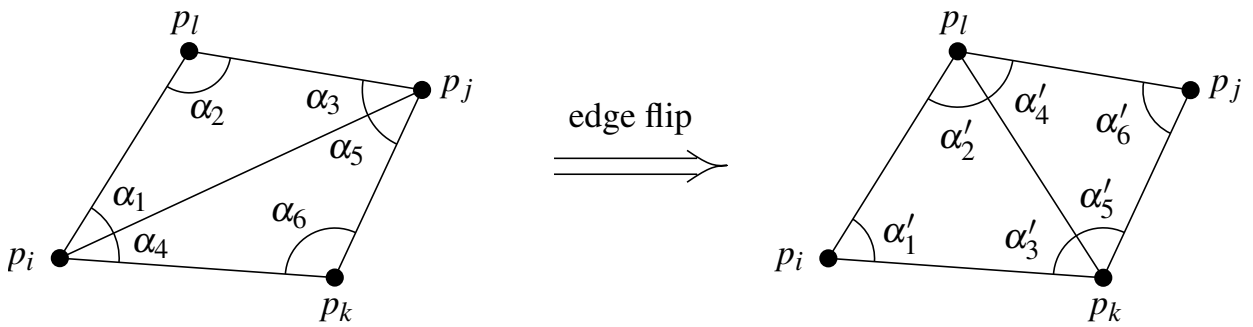
Si e **ne fait pas partie de l'enveloppe convexe** de la triangulation, on sait qu'elle est **partagée par deux triangles** $\triangle p_i p_j p_k$ et $\triangle p_i p_j p_l$.



[1]

Obtention d'une triangulation

Une nouvelle triangulation $\mathcal{T}'$ de P peut alors être obtenue en **remplaçant l'arête $\overline{p_i p_j}$ par l'arête $\overline{p_k p_l}$**.

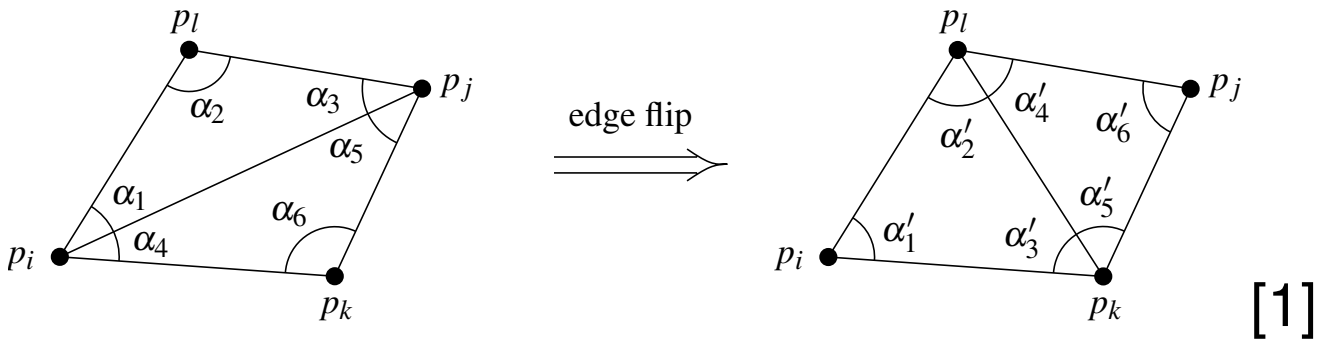


[1]

Ainsi, **les seules différences entre les deux vecteur-angles** sont les six angles $\alpha_1, \ldots, \alpha_6$ dans $\alpha(\mathcal{T})$ qui sont remplacés par $\alpha'_1, \ldots, \alpha'_6$ dans $\alpha(\mathcal{T}')$.

Obtention d'une triangulation

On dira que l'arête $e = \overline{p_i p_j}$ est **illégale** si $\min_{1 \leq i \leq 6} \alpha_i < \min_{1 \leq i \leq 6} \alpha'_i$.



[1]

En d'autres mots, si une arête est illégale, **on peut augmenter localement la valeur du plus petit angle** en permutant les arêtes.

Obtention d'une triangulation

Soit $\mathcal{T}$ une triangulation possédant une arête illégale *e* et soit $\mathcal{T}'$ la triangulation obtenue en permutant *e*. On aura alors que $\alpha(\mathcal{T}') > \alpha(\mathcal{T})$.

On définit alors le concept de **triangulation légale** comme étant une triangulation **ne contenant aucune arête illégale**.

De plus, **toute triangulation angle-optimale est légale**.

Obtention d'une triangulation

En utilisant cette définition, **l'obtention d'une triangulation légale à partir d'une triangulation initiale** est fort simple : il suffit de faire permuter des arêtes jusqu'à ce qu'elles soient toutes légales.

Comme il existe un nombre fini de triangulations pour un ensemble de points P, cette méthode fournira assurément une triangulation légale.

Obtention d'une triangulation

Cela dit, cette technique est trop lente pour être intéressante en pratique. De plus, **on ne peut garantir l'obtention d'une triangulation angle-optimale** par cette méthode, puisque le résultat obtenu dépend du point de départ.

On prendra plutôt un long détour pour trouver la triangulation souhaitée...

Diagrammes de Voronoï

Diagrammes de Voronoï (Principe de base)

Soit P un ensemble de n points distincts sur le plan, appelés sites.

Le **diagramme de Voronoï** de P est la **subdivision du plan** en n cellules (une pour chaque site) telle qu'un point $q \notin P$ est dans la cellule correspondante à un site $p_i \in P$ si et seulement si

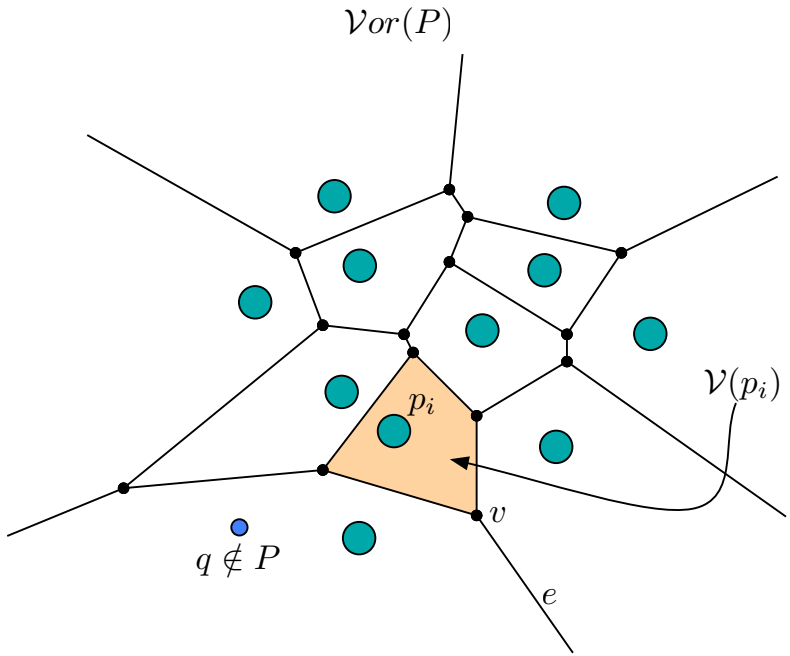
$$d(q, p_i) < d(q, p_j) \qquad \text{pour tout } p_j \in P \text{ avec } i \neq j,$$

où $d(p, q)$ est la distance euclidienne entre les points p et q.

Diagrammes de Voronoï (Principe de base)

Dans le diagramme de droite,

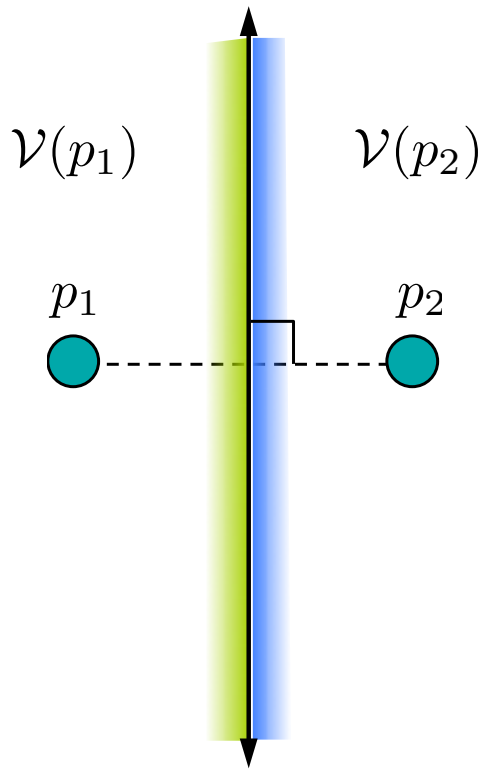
- $\mathcal{V}or(P)$ est le diagramme de Voronoï associé à P
- $\mathcal{V}(p_i)$ est la cellule associée au site p_i
- q est un point quelconque absent de P
- e est une arête de Voronoï
- v est un sommet de Voronoï



Principe de base

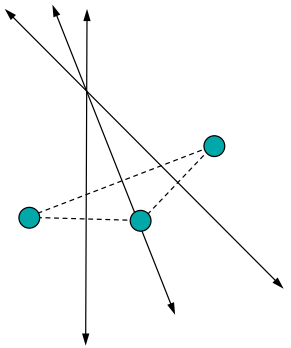
Si P contient seulement deux points, p_1 et p_2, alors la droite bissectrice sépare le plan en 2 demi-plans, $h(p_1, p_2)$ et $h(p_2, p_1)$.

De plus, **cette droite est la seule arrête du diagramme de Voronoï** associé à P.



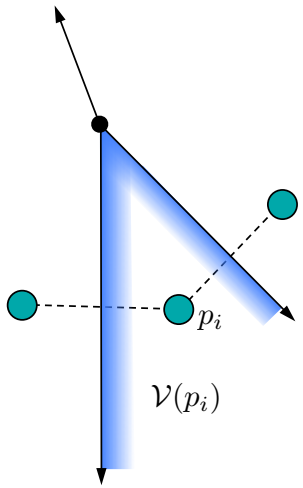
Principe de base

Lorsqu'on ajoute un point à P, on ajoute **une bissectrice pour chacune des paires de points**.



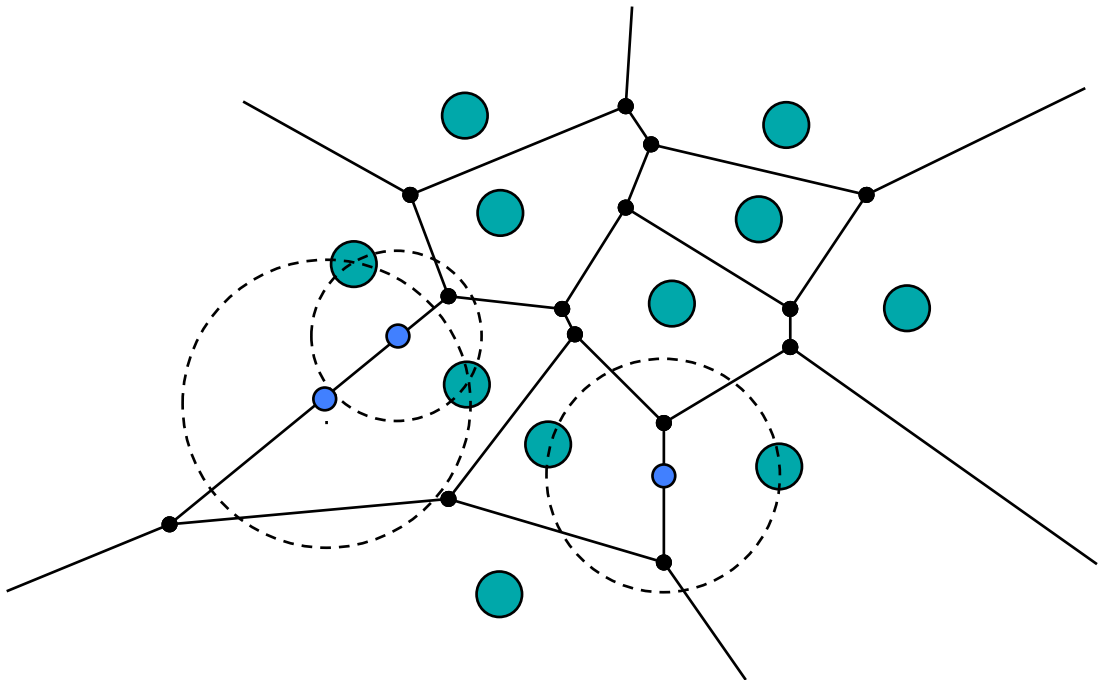
Une cellule est l'**intersection de plusieurs demi-plans** :

$$\mathcal{V}(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ i \neq j}} h(p_i, p_j)$$



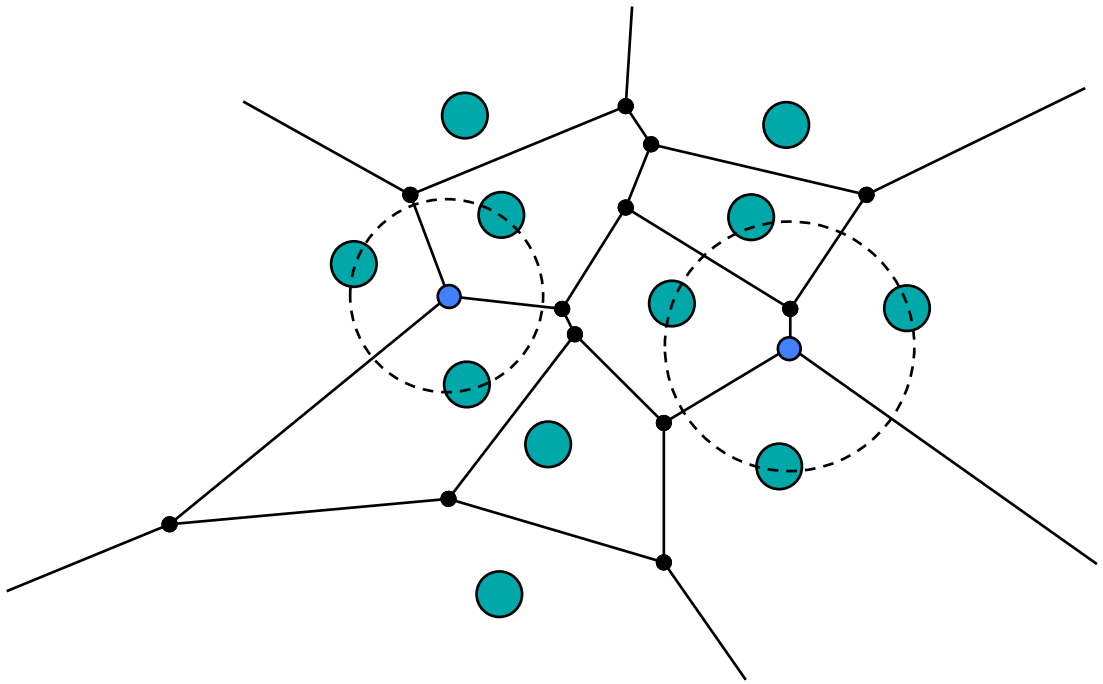
Principe de base

Un point $q \notin P$ est **sur une arête** de $\mathcal{V}or(P)$ entre les sites p_i et p_j si et seulement si le plus grand cercle vide centré en q **touche seulement à** p_i **et** p_j.



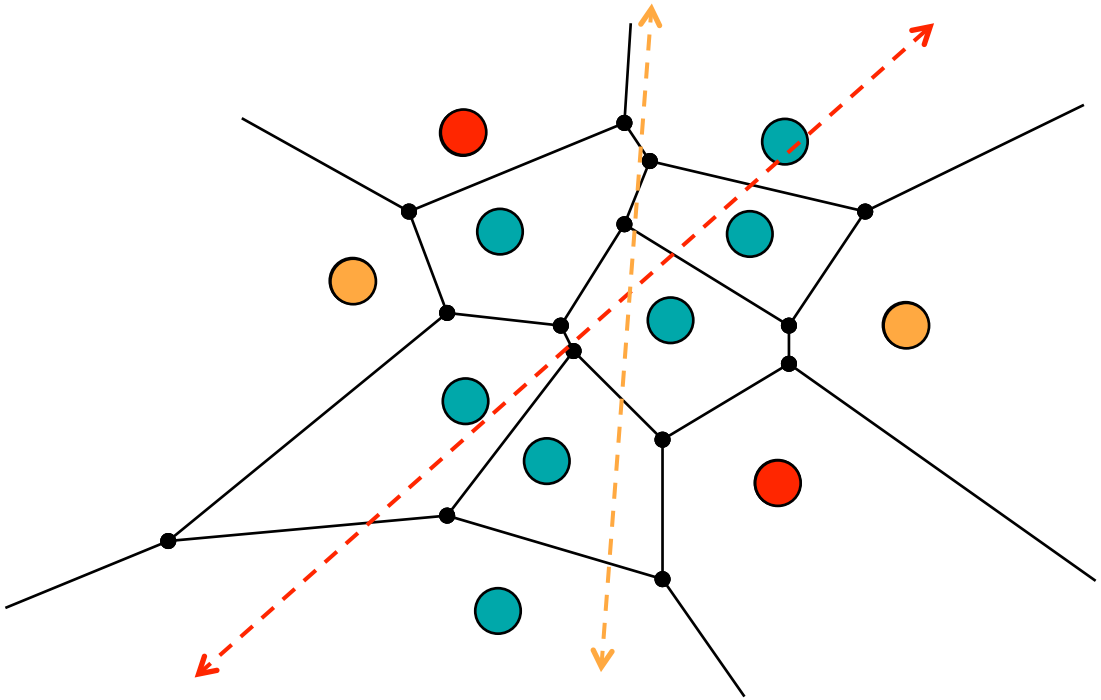
Principe de base

Un point $q \notin P$ est **un sommet** de $\mathcal{V}or(P)$ entre les sites p_i et p_j si et seulement si le plus grand cercle vide centré en q **touche seulement à au moins trois sites**.



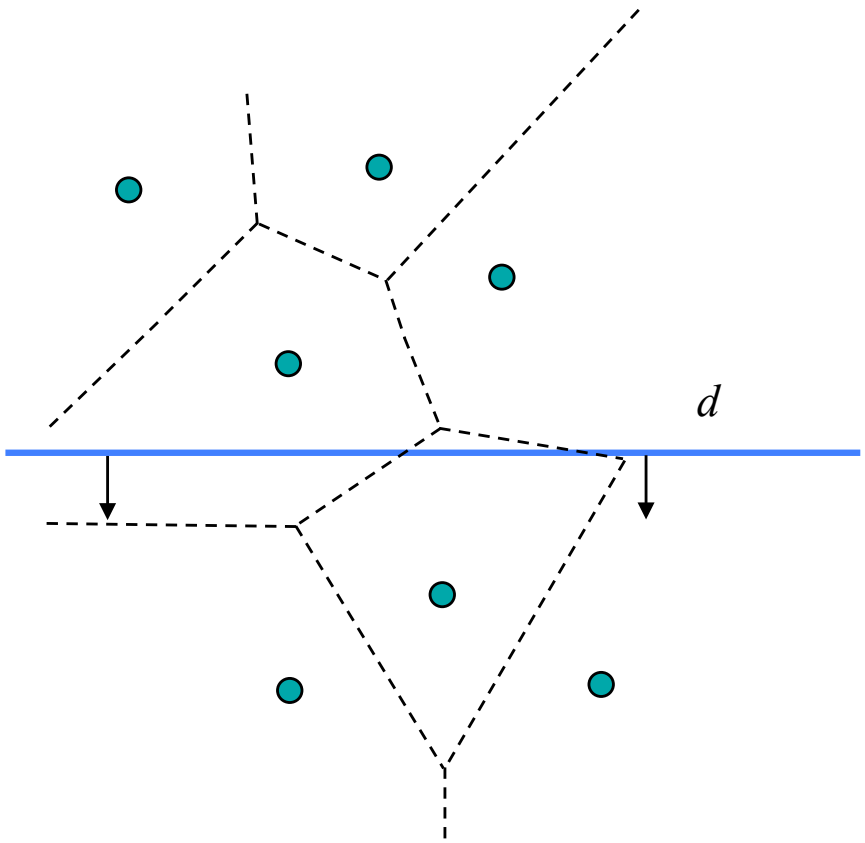
Principe de base

Notons que ce ne sont **pas toutes les bissectrices ni toutes les intersections de bissectrices** qui sont dans $\mathcal{V}or(P)$.



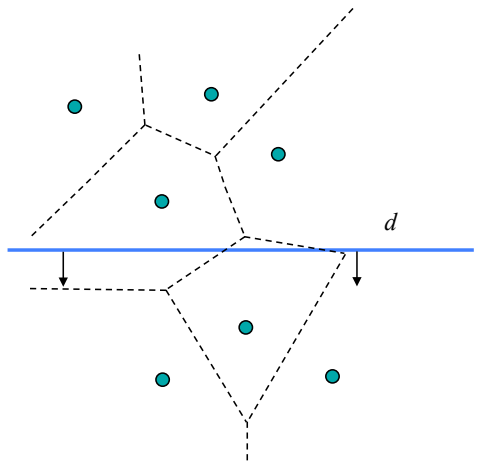
Algorithme de Fortune

L'**algorithme de Fortune** est une approche par balayage d'une droite d de haut en bas de l'ensemble des sites.



Algorithme de Fortune

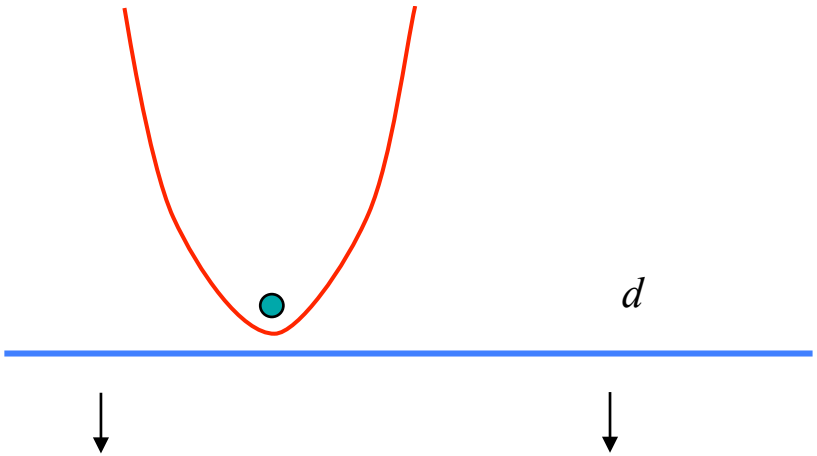
L'idée est de maintenir une représentation des positions des points qui sont **plus près d'un site p_i au dessus de d que de d**.

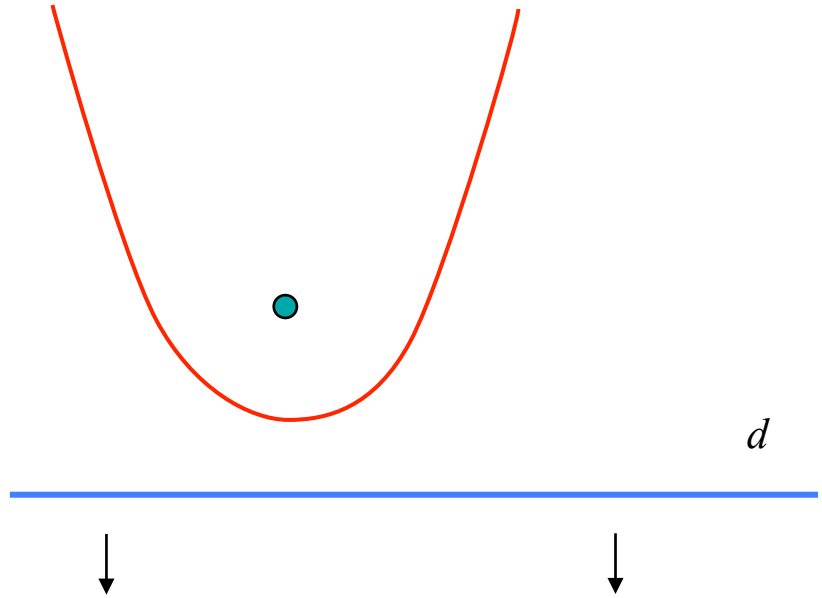


Cette représentation ne changera plus, car un tel point est **forcément plus près de d que de tous les sites en dessous de d**.

Algorithme de Fortune

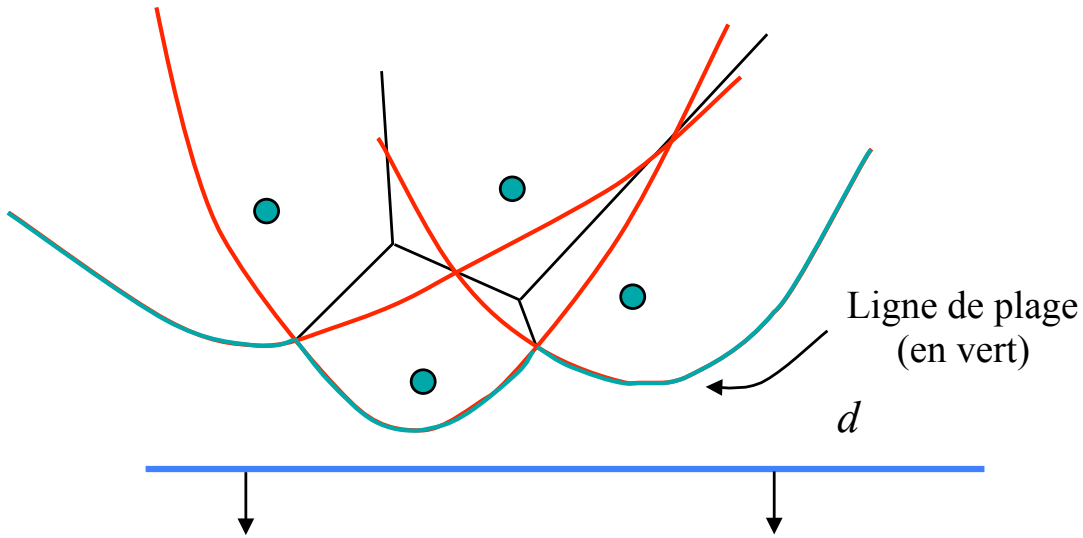
On peut caractériser analytiquement la ligne d'équidistance entre un point et une droite par **une parabole qui s'évase à mesure que la droite s'éloigne du point**.





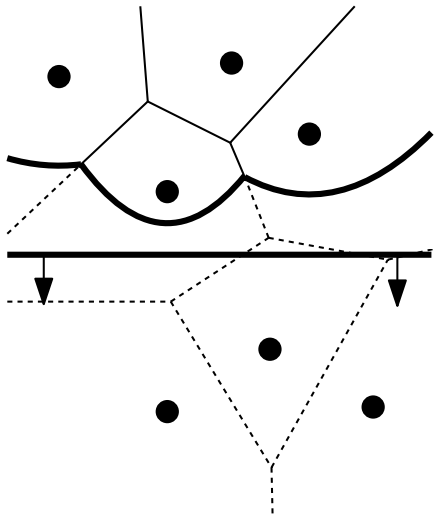
Algorithme de Fortune

Si on a plusieurs sites, chacun d'eux est représenté par une parabole différente. La ligne de front des paraboles est appelée **ligne de plage**.



Algorithme de Fortune

Une arête du diagramme de Voronoï correspond aux intersections des paraboles. La position des intersections évolue à mesure que d descend.



[1]

Ce qui est au dessus de la ligne de plage ne change plus.

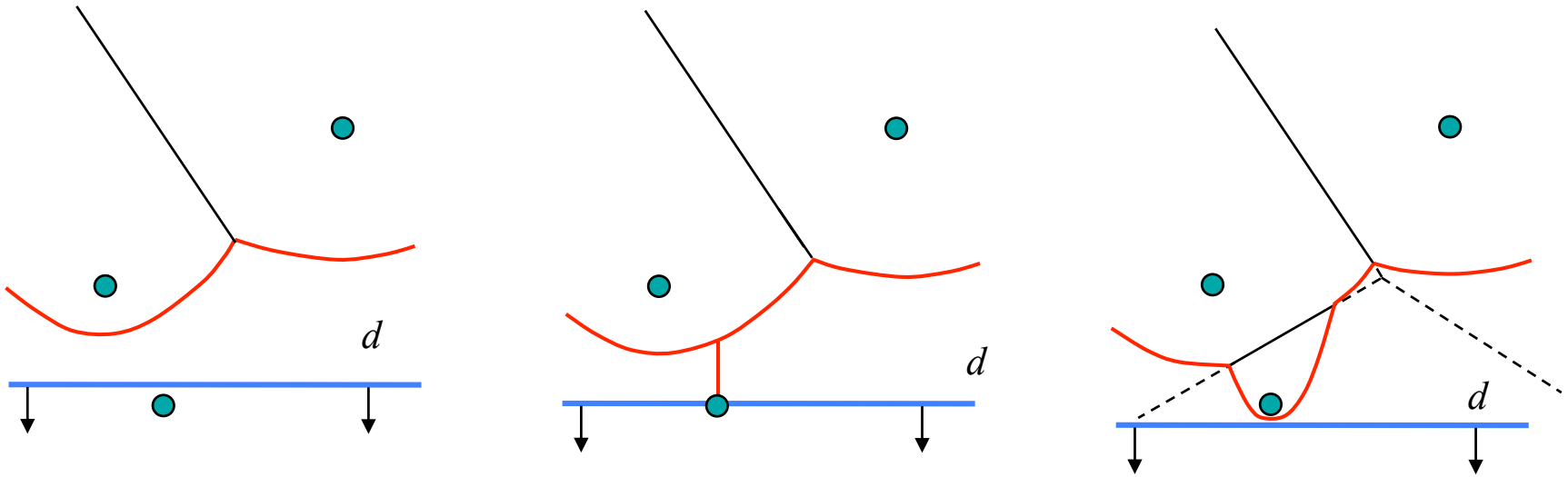
Algorithme de Fortune

À mesure que la droite *d* descend et parcourt l'ensemble des sites, **deux événements peuvent survenir** :

- *d* rencontre un site
- un arc de parabole disparaît

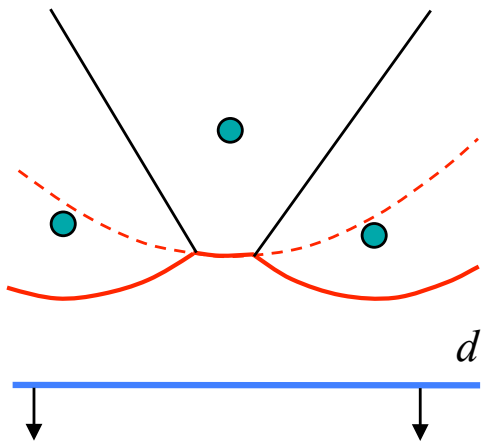
Algorithme de Fortune

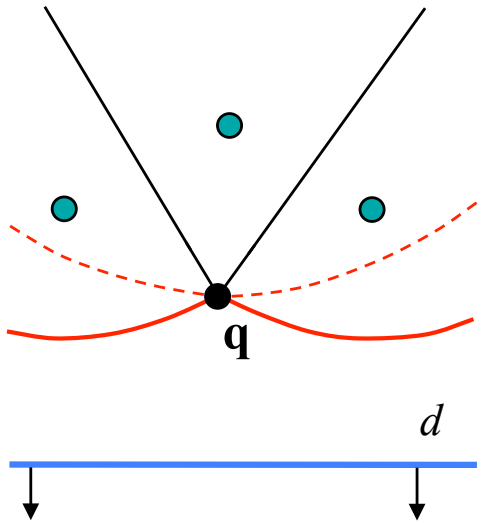
Lorsque *d* rencontre un site, **une nouvelle parabole est créée**, ce qui modifie la ligne de plage, et **une nouvelle arête** s'ajoute au diagramme.

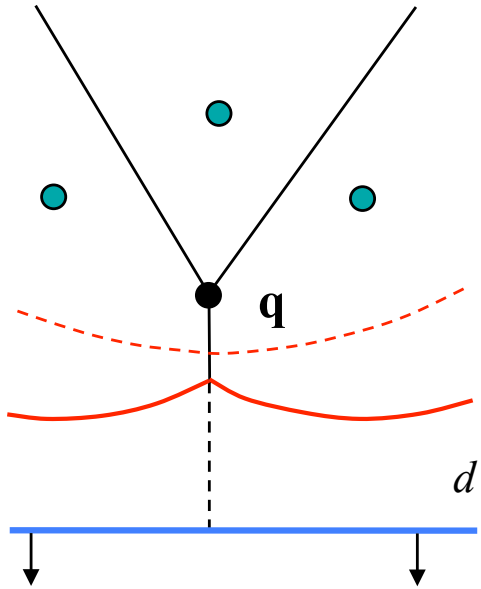


Algorithme de Fortune

Lorsqu'un arc de parabole diminue et disparaît au point q, **un nouveau sommet** et **une nouvelle arête** s'ajoutent au diagramme.

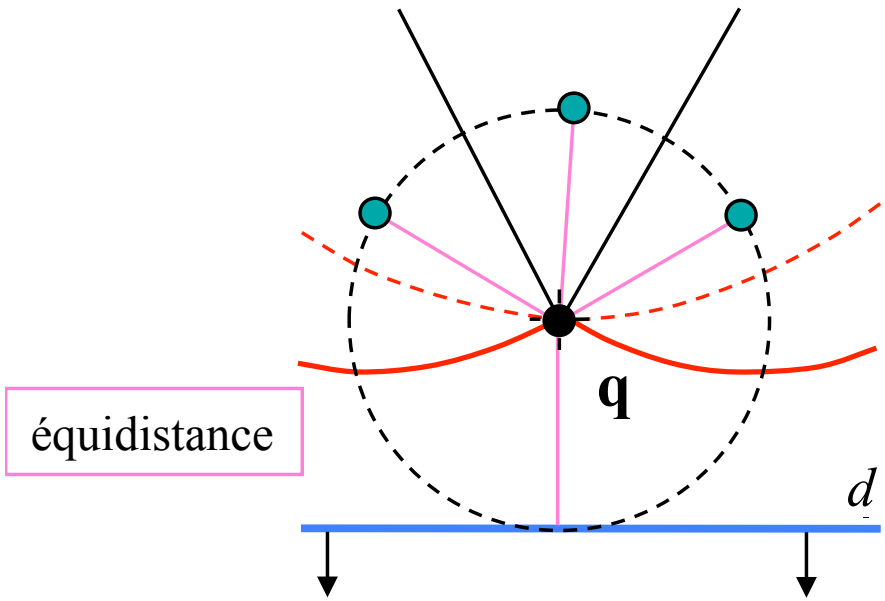






Algorithme de Fortune

De plus, lors de la disparition d'un arc de parabole au point q, **au moins trois sites sont sur la frontière d'un cercle vide** centré en q.



Algorithme de Fortune

Pour l'implantation de l'algorithme de Fortune pour la construction d'un diagramme de Voronoï, on utilise deux structures de données :

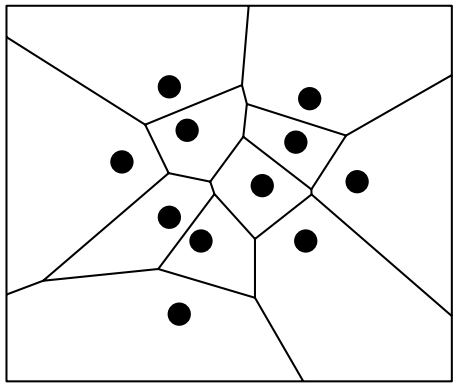
- Un **arbre binaire de recherche** balancé pour stocker la ligne de plage ;
- Une **liste d'arêtes doublement chaînée** pour stocker le diagramme de Voronoï en construction.

La **liste d'arêtes doublement chaînée** (*doubly-connected edge list*, ou DCEL), à la base, est une structure élaborée pour stocker une **subdivision du plan**.

DCEL

Ainsi, une DCEL sert à représenter en mémoire **une subdivision planaire**, mais son fonctionnement se prête très bien au stockage d'un diagramme de Voronoï.

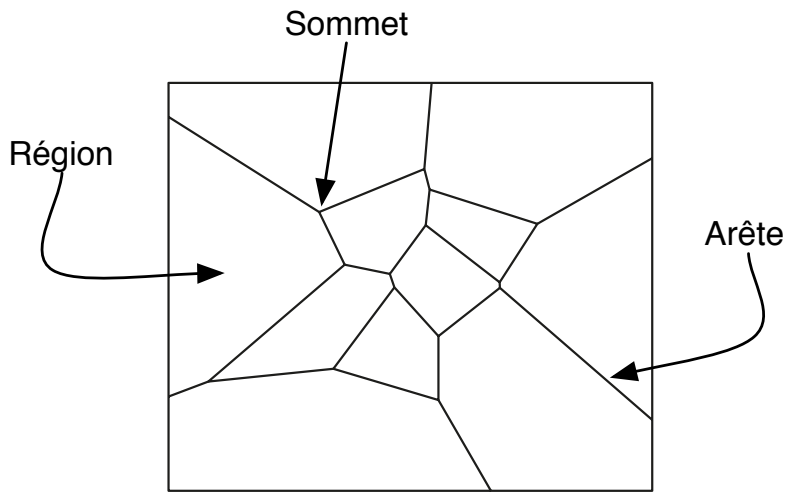
Pour y arriver, on définit un rectangle englobant tous les points formant le diagramme de Voronoï :



[1]

DCEL

Une DCEL contient un enregistrement pour chaque **région**, chaque **arête** et chaque **sommet** du diagramme de Voronoï.



DCEL

L'information géométrique et topologique contenue dans la DCEL doit permettre d'effectuer les opérations de base suivantes :

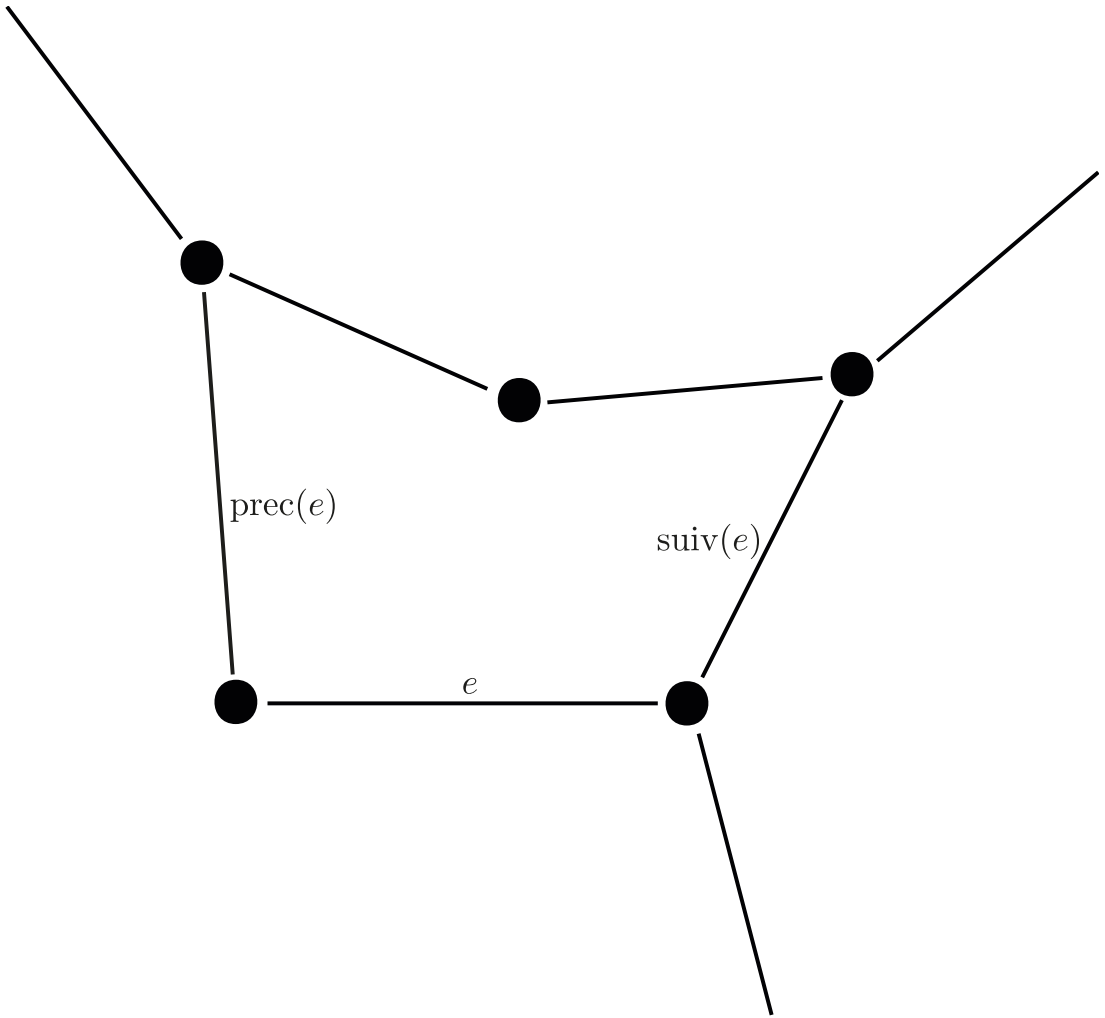
- **Faire le tour d'une région** en passant par chacune de ses arêtes ;
- **Accéder à une région à partir d'une région adjacente** en connaissant l'arrête commune ;
- **Visiter toutes les arêtes** ayant comme extrémité un point donné.

DCEL

Pour pouvoir effectuer la première tâche, c'est-à-dire **se promener d'une arête à l'autre** autour d'une région, on doit stocker **un pointeur de l'arête vers la suivante**.

Qui plus est, ce n'est pas très forçant de permettre en plus le parcours dans l'autre sens. Il suffit alors de conserver aussi **un pointeur vers l'arête précédente**.

DCEL



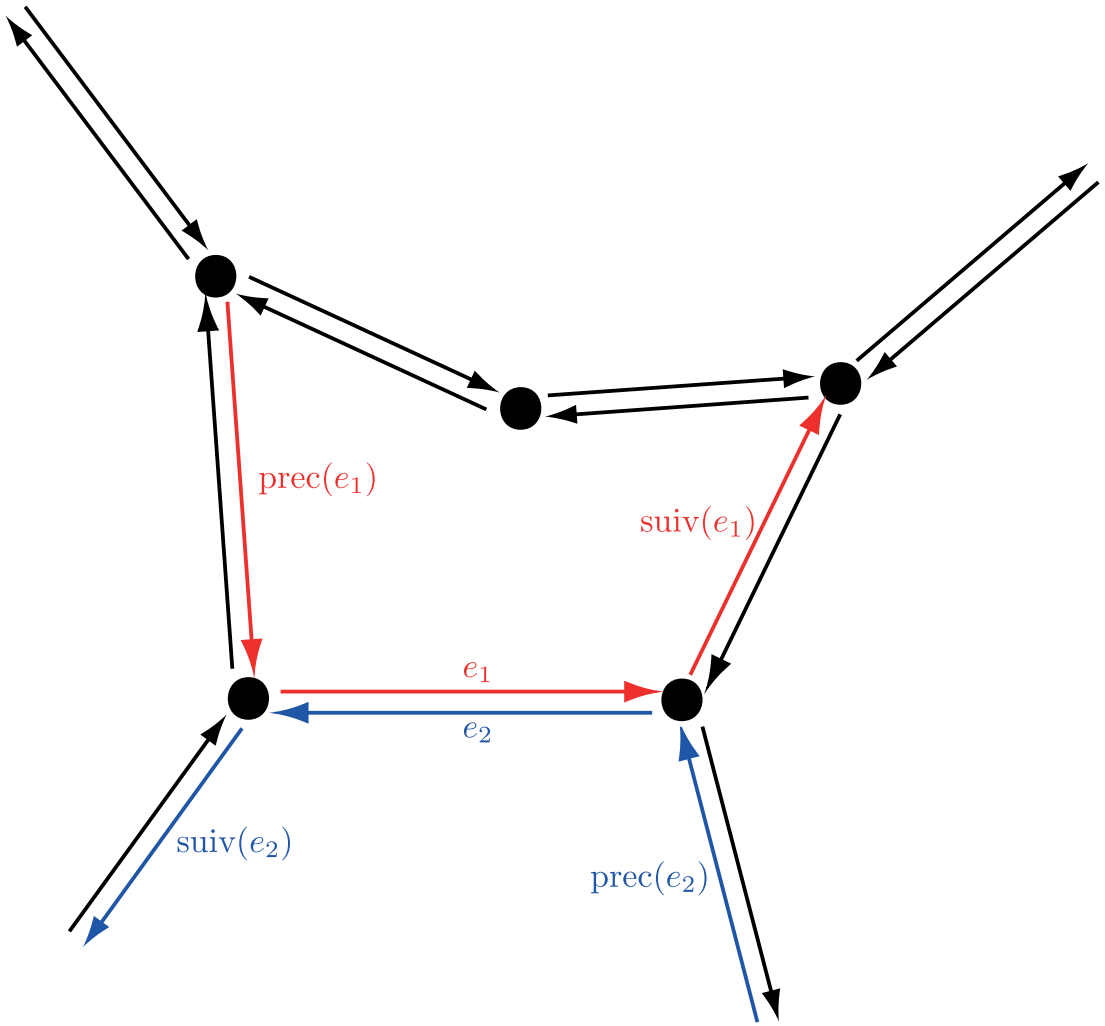
DCEL

Comme une arête se trouve habituellement sur la frontière de deux régions, on devra stocker **deux paires de pointeurs**.

Il devient alors pertinent de voir les deux côtés d'une arête **e comme deux demi-arêtes distinctes $\mathbf{e}_1$** et **$\mathbf{e}_2$**, de manière à définir sans ambiguïtés les deux paires d'arêtes précédente et suivante, dans le sens antihoraire (par convention).

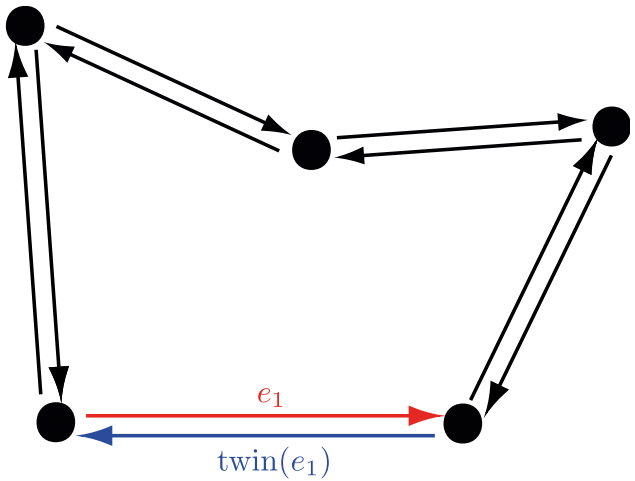
Cela a aussi comme conséquence plaisante qu'**une demi-arête sera sur la bordure d'une seule région**.

DCEL



DCEL

Les deux demi-arêtes associées à une arête sont appelés **jumeaux**.

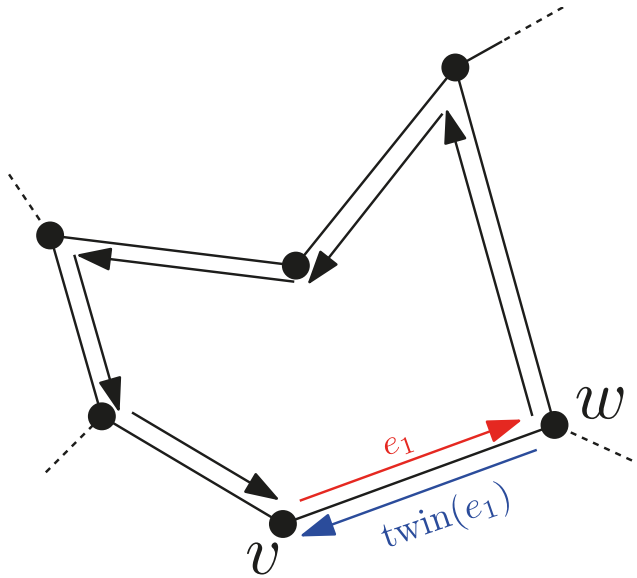


Soit une demi-arête **e**. Définir la demi arête suivante dans le sens antihoraire a pour conséquence que **la région bornée par celle-ci se trouvera toujours du côté gauche**.

DCEL

Comme les arêtes sont orientées, on peut parler de **point d'origine** et de **point d'arrivée** de celles-ci.

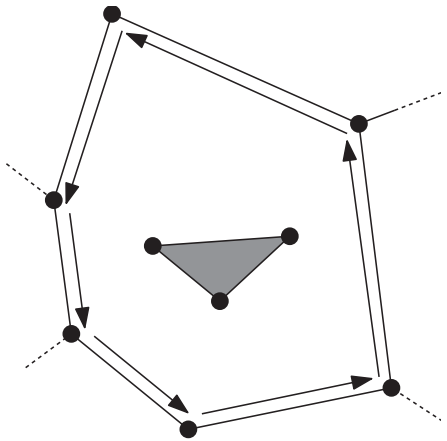
Si une arête **e** possède *v* comme point d'origine et *w* comme point d'arrivée, alors **son arête jumelle** a *w* comme point de départ et *v* comme point d'arrivée.



DCEL

Finalement, un enregistrement de région dans la liste doit posséder **un pointeur vers une demi-arête sur sa frontière**, de manière à permettre d'en faire le tour.

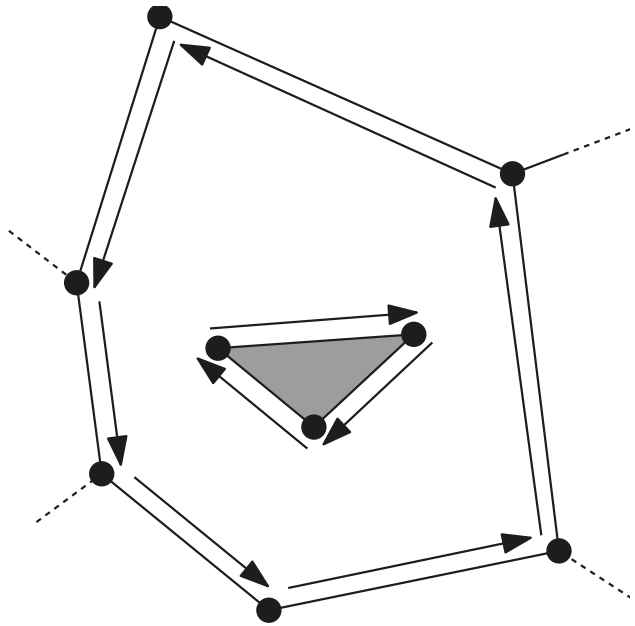
Une situation qui n'a pas encore été gérée est celle des **trous dans une région**.



Dans ce cas, en parcourant les arêtes formant le trou dans le sens antihoraire, on aura que **la région se trouvera à la droite** de celles-ci.

DCEL

Comme il est préférable de toujours avoir les régions du même côté des arêtes (à gauche), on choisira de **parcourir les arêtes dans le sens horaire** dans le cas d'un trou dans une région.



DCEL

La gestion des trous dans les régions signifie aussi que le fait de **posséder un unique pointeur vers une des demi-arêtes** composant son contour n'est plus suffisant.

Il est plutôt nécessaire pour une région de posséder **un pointeur vers une demi-arête de chaque frontière qui la compose**, que celle-ci soit associée au contour ou à un trou dans la région.

DCEL

En résumé, une DCEL compte **trois collections d'enregistrements** : une pour les sommets, une pour les régions et une pour les demi-arêtes.

L'enregistrement pour un sommet *v* comprend les coordonnées de *v* ainsi qu'un pointeur vers une quelconque demi-arête ayant *v* comme extrémité.

L'enregistrement pour une région *f* stocke quant à elle un pointeur vers une quelconque demi-arête sur sa bordure, de même qu'une liste de pointeurs vers une demi arête quelconque faisant partie de la frontière de chaque trou dans la région.

DCEL

Finalement, **l'enregistrement pour une demi-arête e** doit inclure

- Un pointeur vers son sommet d'origine ;
- Un pointeur vers sa demi-arête jumelle ;
- Un pointeur vers la région qu'il borne.

Il n'est toutefois pas nécessaire de stocker **un pointeur vers son sommet d'arrivée**, puisque celui-ci correspond à l'origine de la demi-arrête jumelle.

DCEL

L'enregistrement d'une demi-arête doit aussi contenir **des pointeurs vers les arêtes suivante et précédente**.

Les enregistrements pour les sommet et les arêtes occupent **une quantité de mémoire fixe**, tandis que l'espace occupé par un enregistrement de région est **variable** en raison de la liste de pointeurs vers les trous dans la région.

Algorithme de Fortune

Algorithm VORONOIDIAGRAM(P)
Input. A set $P := \{p_1, \ldots, p_n\}$ of point sites in the plane.
Output. The Voronoi diagram $\mathrm{Vor}(P)$ given inside a bounding box in a doubly-connected edge list $\mathcal{D}$.

1. Initialize the event queue $\mathcal{Q}$ with all site events, initialize an empty status structure $\mathcal{T}$ and an empty doubly-connected edge list $\mathcal{D}$.
2. **while** $\mathcal{Q}$ is not empty
3. **do** Remove the event with largest y-coordinate from $\mathcal{Q}$.
4. **if** the event is a site event, occurring at site p_i
5. **then** HANDLESITEEVENT(p_i)
6. **else** HANDLECIRCLEEVENT(γ), where γ is the leaf of $\mathcal{T}$ representing the arc that will disappear
7. The internal nodes still present in $\mathcal{T}$ correspond to the half-infinite edges of the Voronoi diagram. Compute a bounding box that contains all vertices of the Voronoi diagram in its interior, and attach the half-infinite edges to the bounding box by updating the doubly-connected edge list appropriately.
8. Traverse the half-edges of the doubly-connected edge list to add the cell records and the pointers to and from them.

Algorithme de Fortune

HANDLESITEEVENT(p_i)

1. If $\mathcal{T}$ is empty, insert p_i into it (so that $\mathcal{T}$ consists of a single leaf storing p_i) and return. Otherwise, continue with steps 2– 5.
2. Search in $\mathcal{T}$ for the arc α vertically above p_i. If the leaf representing α has a pointer to a circle event in $\mathcal{Q}$, then this circle event is a false alarm and it must be deleted from $\mathcal{Q}$.
3. Replace the leaf of $\mathcal{T}$ that represents α with a subtree having three leaves. The middle leaf stores the new site p_i and the other two leaves store the site p_j that was originally stored with α. Store the tuples $\langle p_j, p_i \rangle$ and $\langle p_i, p_j \rangle$ representing the new breakpoints at the two new internal nodes. Perform rebalancing operations on $\mathcal{T}$ if necessary.
4. Create new half-edge records in the Voronoi diagram structure for the edge separating $\mathcal{V}(p_i)$ and $\mathcal{V}(p_j)$, which will be traced out by the two new breakpoints.
5. Check the triple of consecutive arcs where the new arc for p_i is the left arc to see if the breakpoints converge. If so, insert the circle event into $\mathcal{Q}$ and add pointers between the node in $\mathcal{T}$ and the node in $\mathcal{Q}$. Do the same for the triple where the new arc is the right arc.

Algorithme de Fortune

HANDLECIRCLEEVENT(γ)

1. Delete the leaf γ that represents the disappearing arc α from $\mathcal{T}$. Update the tuples representing the breakpoints at the internal nodes. Perform rebalancing operations on $\mathcal{T}$ if necessary. Delete all circle events involving α from $\mathcal{Q}$; these can be found using the pointers from the predecessor and the successor of γ in $\mathcal{T}$. (The circle event where α is the middle arc is currently being handled, and has already been deleted from $\mathcal{Q}$.)
2. Add the center of the circle causing the event as a vertex record to the doubly-connected edge list $\mathcal{D}$ storing the Voronoi diagram under construction. Create two half-edge records corresponding to the new breakpoint of the beach line. Set the pointers between them appropriately. Attach the three new records to the half-edge records that end at the vertex.
3. Check the new triple of consecutive arcs that has the former left neighbor of α as its middle arc to see if the two breakpoints of the triple converge. If so, insert the corresponding circle event into $\mathcal{Q}$. and set pointers between the new circle event in $\mathcal{Q}$ and the corresponding leaf of $\mathcal{T}$. Do the same for the triple where the former right neighbor is the middle arc.

Triangulation de Delaunay

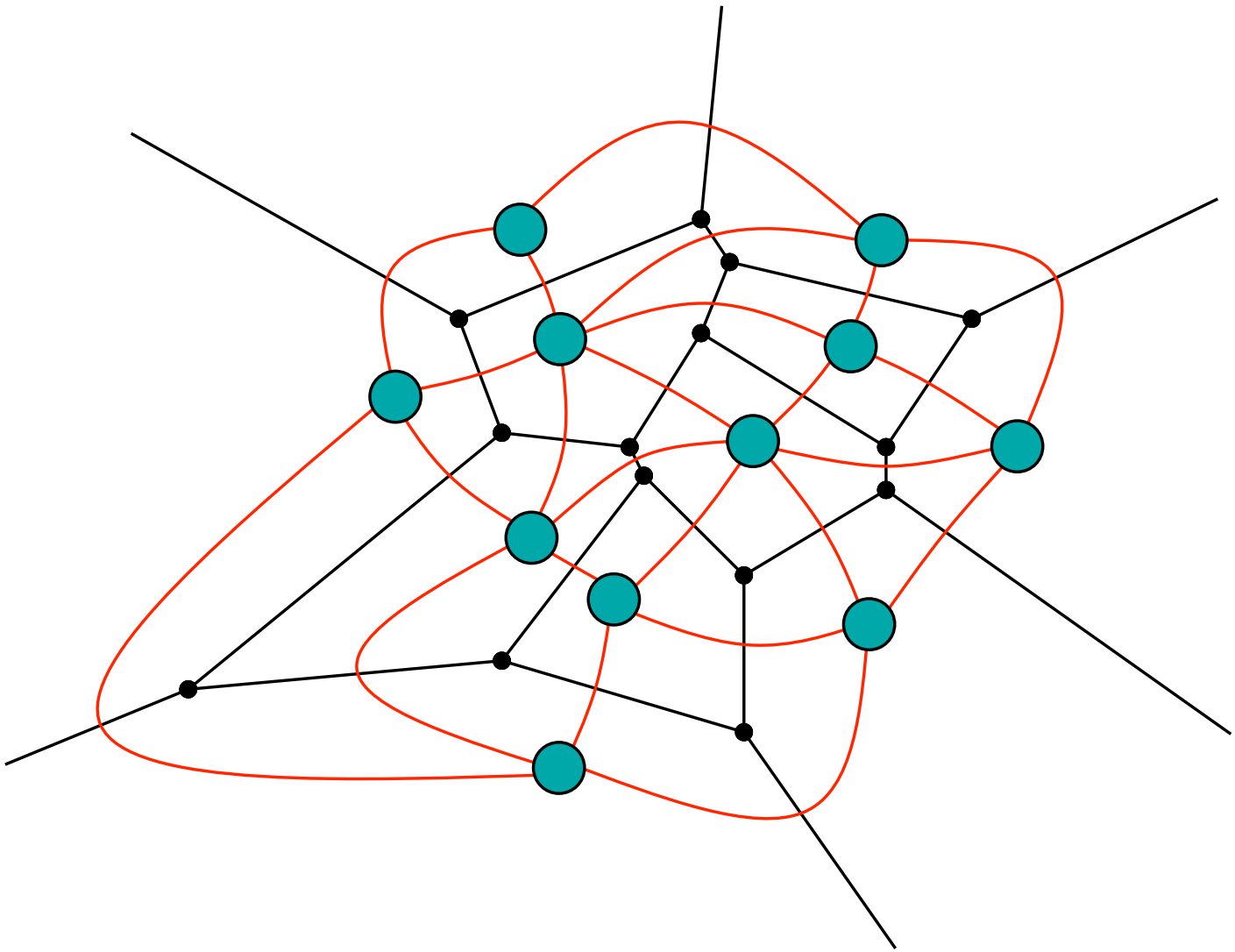
Méthode duale

Soit P un ensemble de n points et $\mathcal{V}or(P)$, son diagramme de Voronoï. On définit $\mathcal{G}(P)$ comme le **graphe dual** de $\mathcal{V}or(P)$.

On a donc que

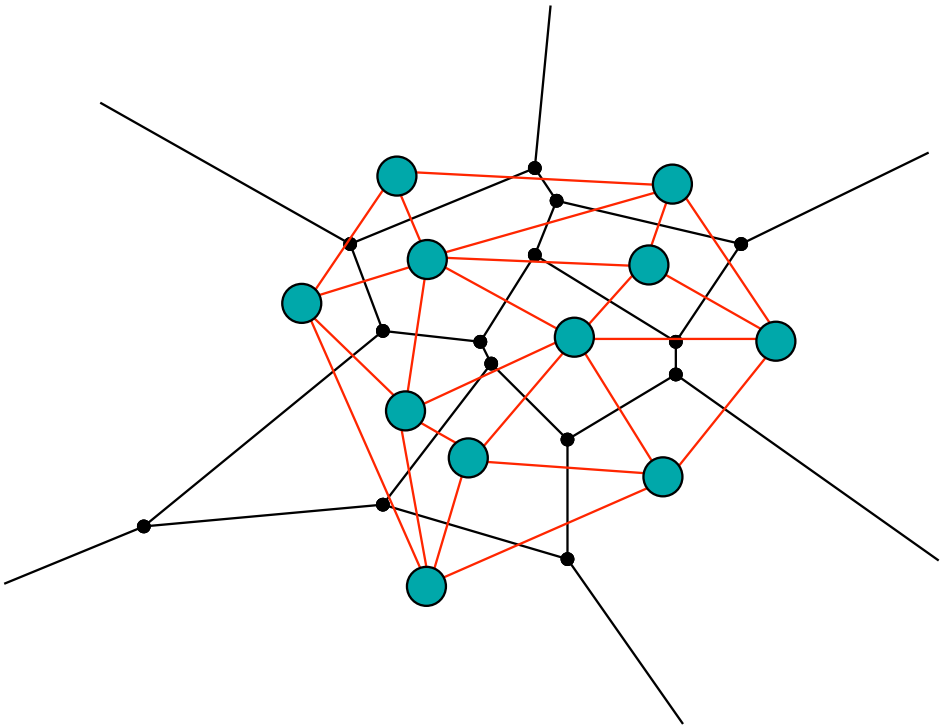
- Pour **chaque cellule** de $\mathcal{V}or(P)$, il y a **un sommet** dans $\mathcal{G}(P)$.
- Pour **chaque arête** de $\mathcal{V}or(P)$, il y a **une arête** dans $\mathcal{G}(P)$.
- Pour **chaque sommet** de $\mathcal{V}or(P)$, il y a **une cellule bornée** dans $\mathcal{G}(P)$.

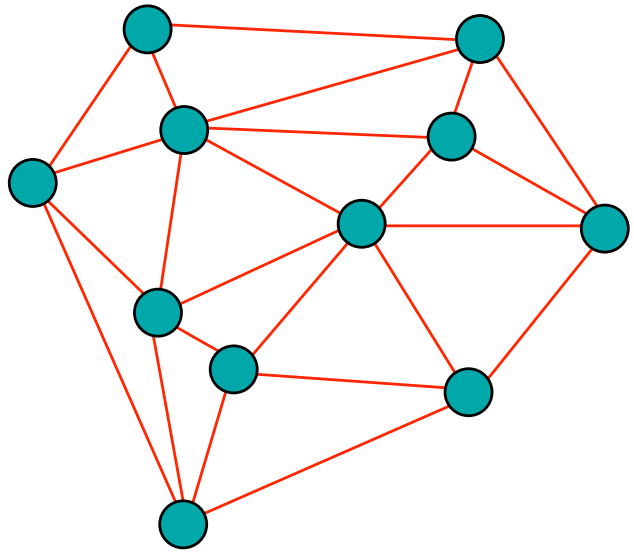
Méthode duale



Méthode duale

On appelle **graphe de Delaunay**, noté $\mathcal{GD}(P)$, le graphe $\mathcal{G}(P)$ où les arêtes sont **des segments de droite entre les sommets**.

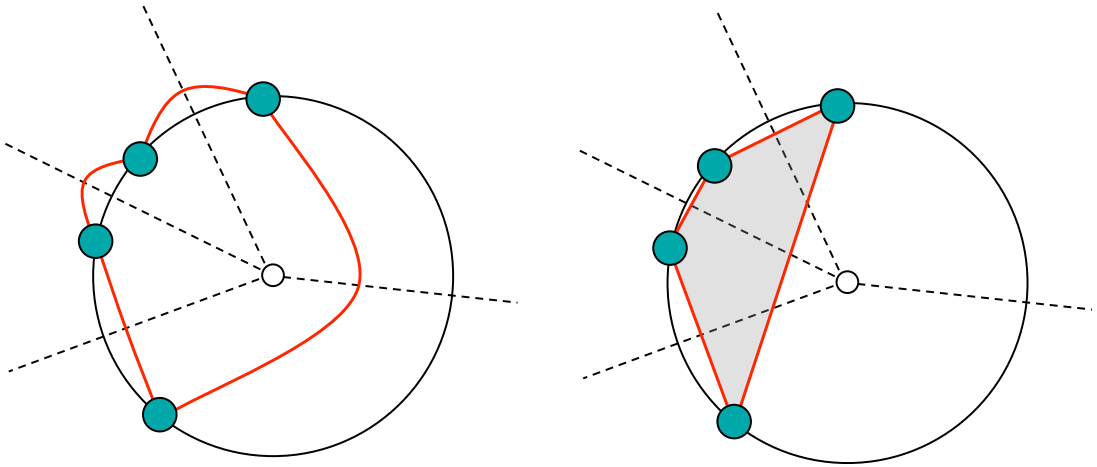




Méthode duale

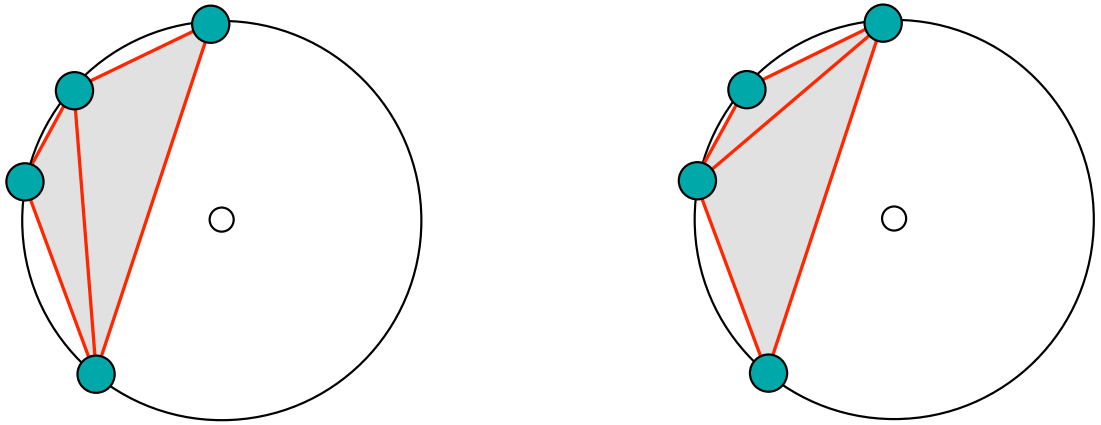
Le graphe de Delaunay n'est **pas nécessairement une triangulation**.

Si un sous-ensemble de **plus de trois points de P se trouve sur un même cercle** vide, ces points forment **un polygone convexe vide**. Cela se produit lorsqu'un sommet de $\mathcal{V}or(P)$ à un degré supérieur à 3.



Méthode duale

La **triangulation de Delaunay**, notée $\mathcal{TD}(P)$, est obtenue en ajoutant 0 ou plus arêtes au graphe de Delaunay $\mathcal{GD}(P)$.



On ajoute le segment **maximisant localement l'angle minimal**.

Méthode duale

La triangulation de Delaunay est **toujours légale**. Elle est aussi **angle-optimale**.

Le problème de trouver une triangulation qui **maximise l'angle minimum** est donc réduit à **trouver une triangulation de Delaunay**.

Algorithme Trouver une triangulation de Delaunay pour $\mathcal{P}$

ENTRÉES: Ensemble de points $\mathcal{P}$

SORTIES: $\mathcal{TD}(\mathcal{P})$

1: Calculer $\mathcal{V}or(\mathcal{P})$
2: Trouver le graphe dual $\mathcal{G}(\mathcal{P})$
3: Redresser les arêtes pour obtenir $\mathcal{GD}(\mathcal{P})$
4: Ajouter des arêtes pour terminer la triangulation au besoin

Méthode incrémentale

D'autres méthodes permettent d'obtenir la triangulation de Delaunay pour un ensemble de points P.

La **méthode incrémentale aléatoire** en est une, et elle a l'avantage de ne pas nécessiter le calcul du diagramme de Voronoï.

Méthode incrémentale

Algorithme Trouver une triangulation de Delaunay pour $\mathcal{P}$ par la méthode incrémentale aléatoire

ENTRÉES: Ensemble de points $\mathcal{P}$

SORTIES: $\mathcal{TD}(\mathcal{P})$

1: Initialiser la triangulation avec un triangle assez gros, de telle sorte qu'il entoure tous les points de $\mathcal{P}$

2: Choisir un point de $\mathcal{P}$ au hasard : $\mathbf{p}_r$

3: Trouver le triangle dans lequel se trouve $\mathbf{p}_r$

4: Subdiviser ce triangle en 3 triangles plus petits ayant $\mathbf{p}_r$ comme sommet

5: Tourner des arêtes jusqu'à ce que toutes les arêtes soient légales

6: Répéter les étapes 2, 3, 4 et 5 jusqu'à ce que tous les points de $\mathcal{P}$ soient dans la triangulation

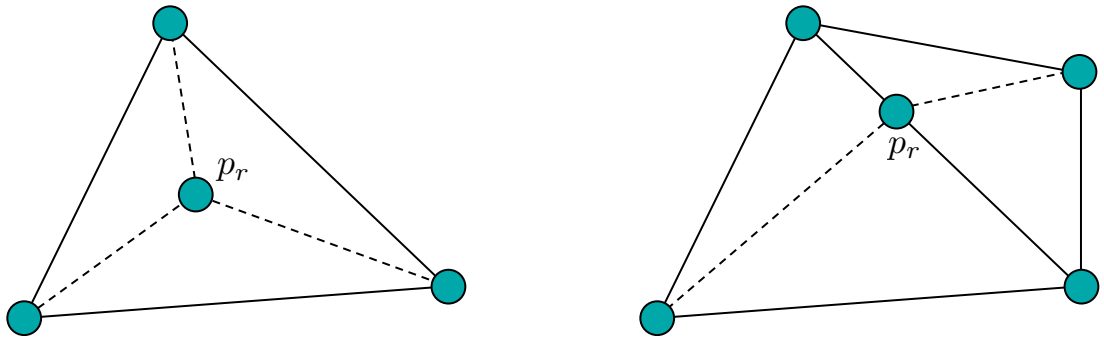
7: Enlever les 3 points du triangle englobant et tous les triangles reliés

Méthode incrémentale

À l'étape 4, avant l'ajout de p_r et la substitution, **toutes les arêtes sont légales**.

Deux cas sont alors possibles lors de l'ajout d'un point :

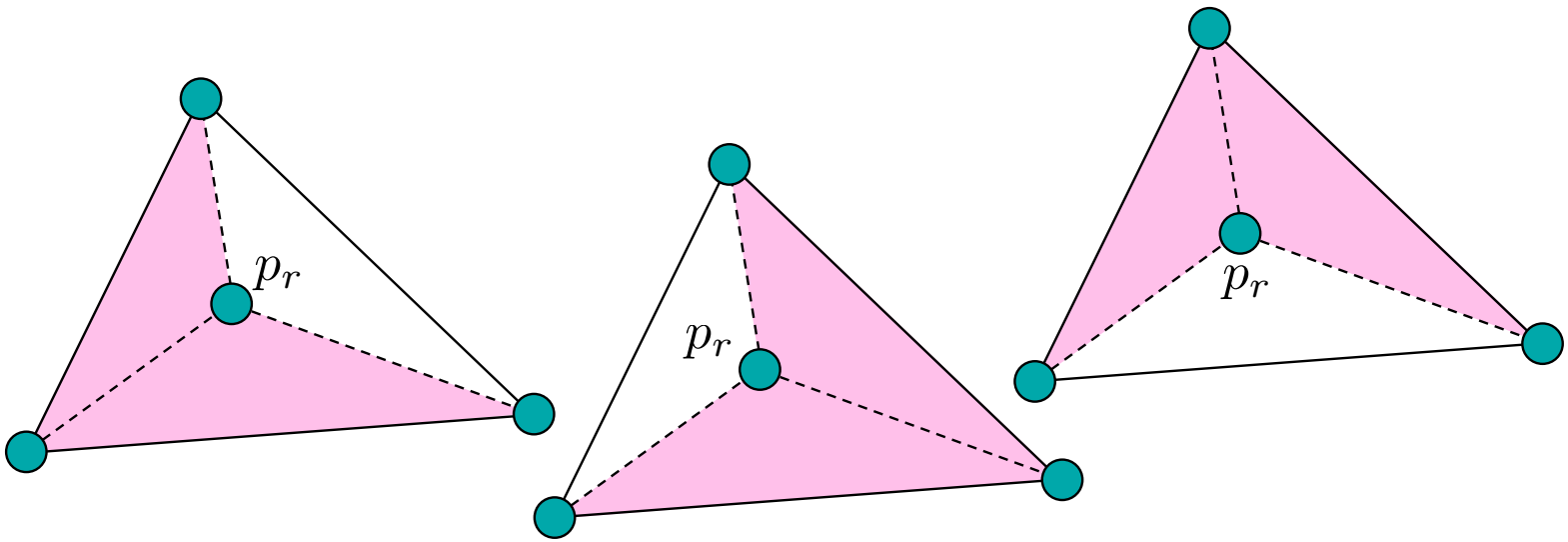
- p_r tombe à l'intérieur d'un triangle
- p_r tombe sur une arête existante



Méthode incrémentale

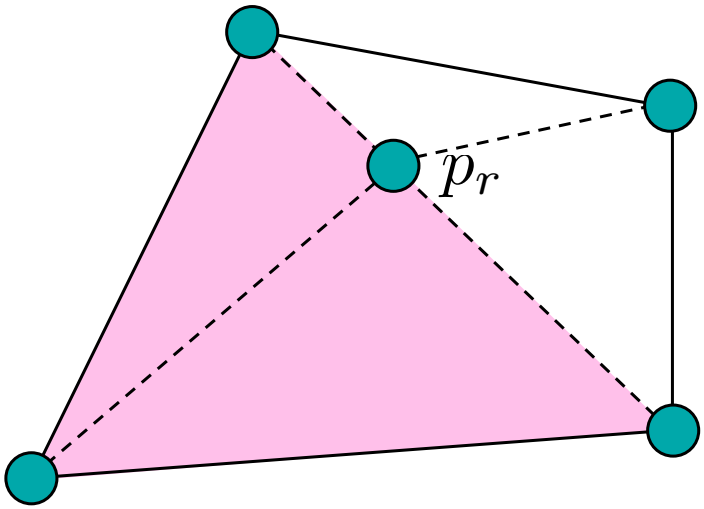
À l'étape 5, on doit déterminer **quelles arêtes sont illégales**. Or, après la subdivision, **toutes les arêtes adjacentes à p_r sont légales**.

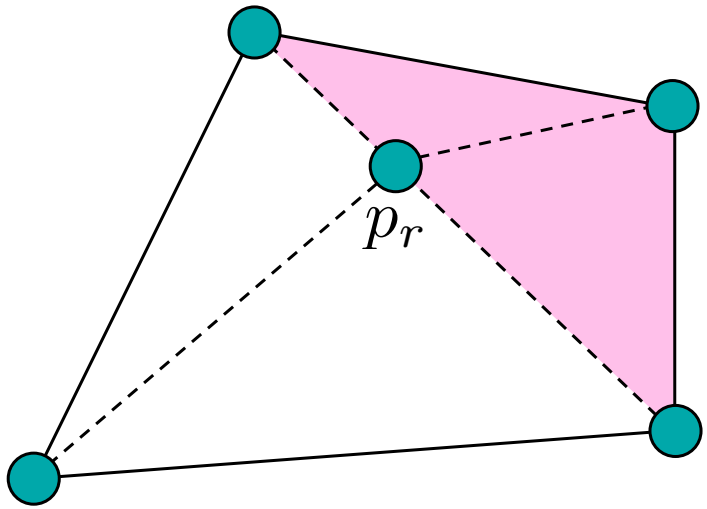
Si p_r tombe à l'intérieur d'un triangle, **les quadrilatères ne sont pas convexes**. On ne peut donc permuter aucune arête.



Méthode incrémentale

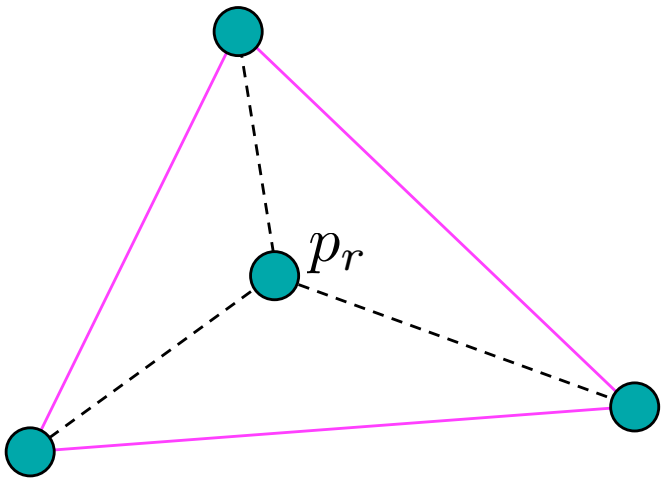
Si p_r tombe sur une arête existante, **deux des quadrilatères sont convexes**. Il est par contre impossible de permuter des arêtes.

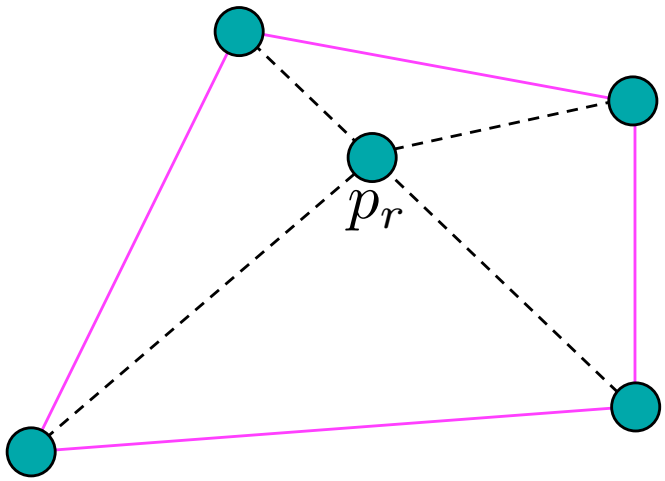




Méthode incrémentale

Une arête devient illégale **si un de ses triangles adjacents a changé**. Les arêtes des nouveaux triangles qui sont non adjacentes à p_r peuvent donc être illégales.





Méthode incrémentale

Il faut donc **vérifier chacun des nouveaux triangles** créés :

- Est-ce qu'on a un quadrilatère convexe ?
- Si oui, a-t-on une arête légale ? Si ce n'est pas le cas, on permute.

Notons que comme la permutation crée deux nouveaux triangles, il faut **reprendre la vérification pour ceux-ci**.

Références

Références I

M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars.
Computational geometry, 2010.

C. D. Hansen and C. R. Johnson.
The visualization handbook, 2004.

W. Schroeder, K. Martin, and B. Lorensen.
The visualization toolkit : An object-oriented approach to 3d graphics, 2006.

A. C. Telea.
Data visualization : Principles and practice, 2008.