LES TRIANGULATIONS

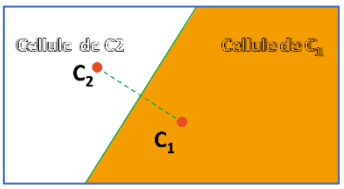
CELLULE DE VORONOÏ

Soit un ensemble de points S appelés centres.
La cellule de Voronoï associée au centre p est défini par

$Vor_S(p) = \{x \in \mathbb{R}^2 / \forall q \in S \ ||x-p|| < ||x-q||\}$

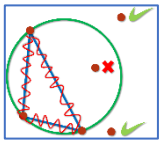
✓ où $||x - p||$ désigne la distance entre x et p

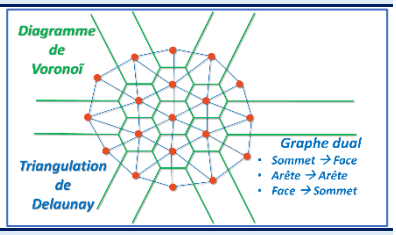
- L'ensemble des cellules de Voronoï définit le diagramme de Voronoï



TRIANGULATION DE DELAUNAY

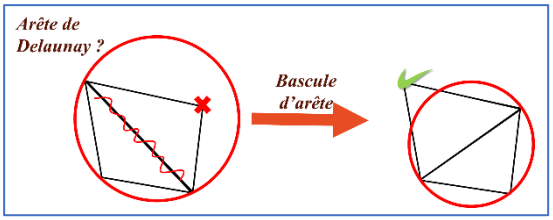
- Une triangulation de Delaunay est le graphe dual d'un diagramme de Voronoï
- Une triangulation de Delaunay est une Subdivision planaire du plan où toutes les faces sont des triangles de Delaunay :
  ✓ Un triangle de Delaunay est un triangle qui vérifie la propriété du cercle vide : Son cercle circonscrit ne contient aucun point dans son intérieur propre.





CALCUL DE LA TRIANGULATION DE DELAUNAY : ALGORITHME INCREMENTAL PAR BASCULE D'ARETES

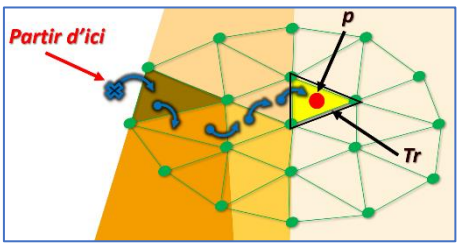
```
1.  Dt : Triangulation de Delaunay
2.  Dt.size = 0 // Nombre de sommets de la triangulation
3.  TableV = null // Table des sommets
4.  TableE = null // Table des aretes
5.  TableF = null // Table des faces
6.  Acces: TableV[i], TableE[i], TableF[i] // i ∈ [0, 1, 2]
7.  insert (Point p)
8.  Switch (Dt.size)
9.     case 0 :
10.       TableV.insert(p)
11.    case 1 :
12.       TableV.insert(p)
13.       TableE.insert(p, TableV[0])
14.    case 2 :
15.       TableV.insert(p)
16.       TableE.insert(p, TableV[0])
17.       TableE.insert(p, TableV[1])
18.       TableF.insert(p,TableV[0],TableV[1])
19.    default :
20.       Face f = Locate(p) // Localiser le point p
21.       Insert(p, f)
22.       Diffuse_Delaunay(p)
23. End Switch
```



Retablir la propriete de Delaunay
Depart : Le nouveau sommet
Pour chaque arete autour du sommet
Si l'arete n'est pas de Delaunay
Basculer l'arete
Continuer la verification pour chaque nouvelle arete

LOCALISATION DANS UNE TRIANGULATION DE DELAUNAY

- Marche par visibilité – Principe ?
  ✓ Partir de n'importe quel triangle, et marcher dans la triangulation : Passer d'un triangle à autre jusqu'à arriver au triangle recherché Tr.




```
1.  locate (Point p)
2.  f = Border face
3.  For i in (1, 2, 3)
4.     dir(i)= Orientation (p, f.vertex[ccw(i)], f.vertex[cw(i)])
5.  While (¬dir(1),¬dir(2),¬dir3) do
6.     For i in (1, 2, 3)
7.        If (¬dir(i))
8.        f = f.neighbor[i] ;
9.        For i in (1, 2, 3)
10.          dir(i)= Orientation (p, f.vertex[ccw(i)], f.vertex[cw(i)])
11. End While
12. return f
```