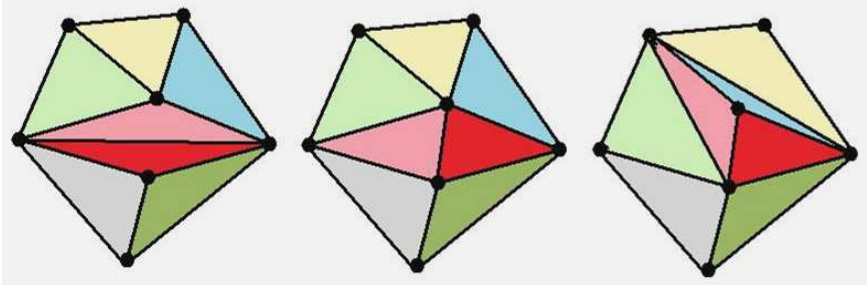
Triangulation de Delaunay
et cellules de Descartes-Dirichlet-Voronoï

Nous avons déjà rencontré les cellules de Descartes, dans le *chapitre 1* (dessin de points, lignes, cercles) du cours *géométrie et graphisme*, en utilisant à ce sujet la méthode des cercles grossissants. Nous allons retrouver ce problème dans un nouveau contexte, en liaison avec celui de la triangulation.

On se place dans le plan, où sont placés *N* points, souvent appelés sites dans ce contexte. Qu'entend-on par triangulation de ces *N* points ? Une telle triangulation est formée de triangles dont les sommets sont les sites, et qui à eux tous constituent une partition de l'enveloppe convexe des *N* points.[1] Il existe plusieurs triangulations, dont l'une est appelée triangulation de Delaunay.[2]



Trois des triangulations possibles pour *N* = 7 sites

Triangulation de Delaunay

Donnons-nous *N* sites. Par définition, on appelle triangle de Delaunay un triangle qui a comme sommets trois des sites, et tel que son cercle circonscrit n'ait en son intérieur strict aucun site. De tels triangles existent, comme on le verra, et l'on démontre que :

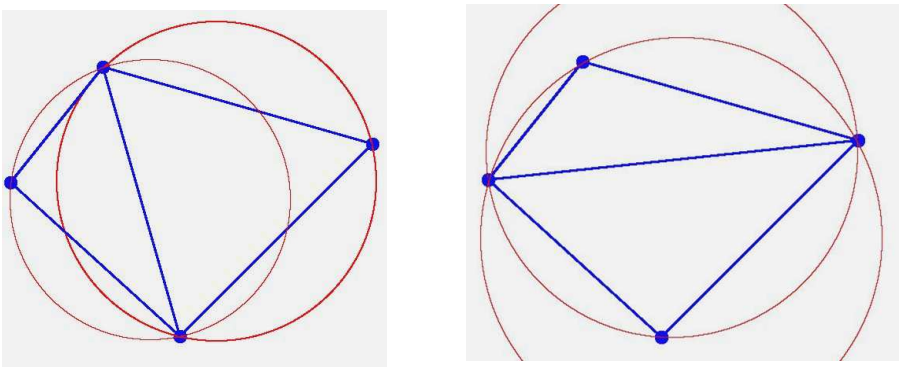
1) L'assemblage de tels triangles de Delaunay pour les *N* sites forme une triangulation des *N* sites, appelée triangulation de Delaunay. Une telle triangulation est unique, sous réserve qu'on n'ait jamais trois sites alignés, ni quatre sites sur le même cercle.

2) A chaque triangulation des *N* sites, on peut associer le mot formé par tous les angles des triangles (il en existe trois par triangle), ces angles étant rangés par ordre croissant dans le mot. Les lettres de ces mots sont donc des nombres entre 0 et 180°, l'ordre alphabétique étant l'ordre croissant des nombres. La triangulation de Delaunay a comme particularité de donner le plus grand mot dans l'ordre alphabétique. Autrement

[1] « Partition » signifie que les triangles recouvrent la surface délimitée par l'enveloppe convexe, sans aucun chevauchement.

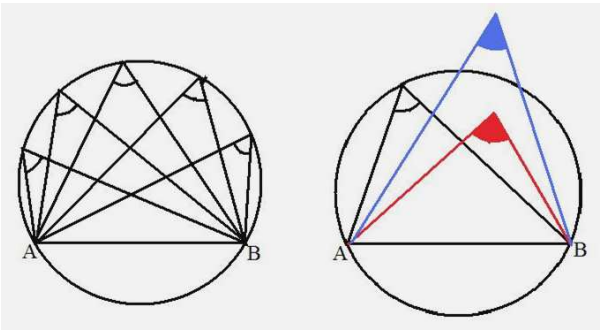
[2] Signalons que Delaunay, ou encore Delone, est un mathématicien russe de prénom Boris Nikolaievitch, mais il a publié certains de ses travaux en français, dans les années 1930, dans le domaine de la cristallographie notamment.

dit, elle a comme première lettre (son plus petit angle) le plus grand angle parmi les premières lettres des autres triangulations. Et si deux triangulations ont la même première lettre (le plus petit de leurs angles), la triangulation de Delaunay est celle qui donne la plus grande deuxième lettre, etc. Simplement dit, la triangulation de Delaunay a tendance à donner des triangles moins aplatis, et c'est là son principal intérêt en pratique. Son autre intérêt est d'être lié au diagramme de Voronoï qui est son dual, comme nous le verrons plus bas.



Deux triangulations de $N = 4$ sites. Celle de *gauche* est une triangulation de Delaunay, celle de *droite* ne l'est pas (les cercles circonscrits aux triangles contiennent le quatrième site).

- **Rappel préalable sur le théorème de l'angle inscrit**



Considérons un cercle et une corde *AB*, comme sur le dessin ci-contre. Le théorème de l'angle inscrit (ou de l'arc capable) indique qu'en se plaçant sur l'arc de cercle situé au-dessus de *AB*, on voit toujours la corde *AB* sous le même angle (il en serait de même avec l'arc au-dessous, avec dans ce cas un angle obtus). Par rapport à cet angle inscrit, il en découle que si l'on prend un point au-dessus du cercle, l'angle (*en bleu* sur le dessin) devient plus petit, et qu'avec un point à l'intérieur du cercle, mais toujours au-dessus de *AB*, l'angle (*en rouge*) devient plus grand. Dans l'algorithme qui suit, on appliquera cette propriété de l'angle inscrit.

Algorithme de la triangulation de Delaunay

On commence par placer les N sites numérotés de 0 à $N - 1$ sur l'écran, par exemple en leur donnant des coordonnées entières, dans le repère de l'écran ayant l'origine en bas à gauche (*yorig* = 599).

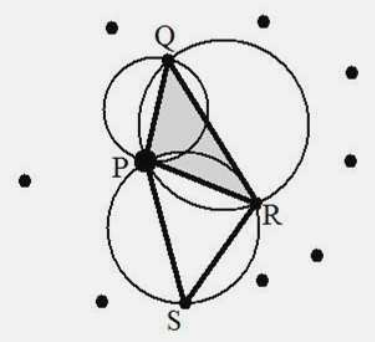
```
for(i=0;i<N;i++)
do {x[i]=rand()%800;y[i]=rand()%600;}
while (troppres(i)==1);
for(i=0;i<N;i++) { filldisc(x[i],yorig-y[i],2,blue);}
```

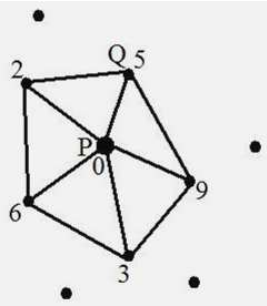
avec la function *troppres*() pour éviter que deux sites soient trop près l'un de l'autre.

```
int troppres(int i)
{  int k;
   for(k=0;k<i;k++)
   if ( abs(x[k]-x[i])<10 && abs(y[k]-y[i])<10 ) return 1;
   return 0;
 }
```

Puis, pour chaque site *P*, on va déterminer tous les triangles de Delaunay qui sont en éventail autour de lui, ces triangles ayant tous comme l'un de leurs sommets ce point *P*. Mais il existe deux cas, selon que l'éventail fait un tour complet autour du site *P*, ou seulement un éventail partiel lorsque le site *P* se trouve sur l'enveloppe convexe de l'ensemble des sites.

Premier cas : le site *P* n'est pas dans l'enveloppe convexe. On commence par chercher le premier triangle de Delaunay ayant *P* comme sommet. Pour cela on prend le site *Q* qui est le plus proche voisin de P. Puis parmi les sites situés en tournant à droite de ***PQ*** on choisit le site *R* tel que l'angle *PRQ* soit maximal. On peut affirmer que le triangle *PRQ* est un triangle de Delaunay.[3] A partir de ce premier triangle, on va obtenir

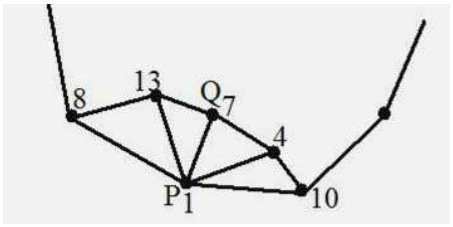




tous les autres triangles situés autour de *P* en tournant toujours vers la droite. On repart de *PR* et on cherche le site *S* situé à droite de ***PR*** et tel que l'angle *PSR* soit maximal. On a encore un triangle de Delaunay.[4] On fait de même à partir de *PS*, et ainsi de suite jusqu'à ce que l'on ait fait un tour complet autour de *P*. On détermine ainsi ce que l'on appelle la liste des voisins du site *P*, par exemple sur le dessin ci-contre, la liste des voisins de 0 est *voisins*[0][]= 5 9 3 6 2 5.

[3] Lorsque l'on choisit *Q* le plus proche de *P*, cela signifie qu'à l'intérieur du cercle de centre *P* et de rayon *PQ*, il n'y a aucun autre site (en vertu du théorème de l'angle inscrit), ni à fortiori à l'intérieur du cercle de diamètre *PQ* (voir dessin). Puis en prenant *R* à droite tel que l'angle *PRQ* soit maximal, cela signifie qu'à l'intérieur du cercle circonscrit à *PRQ*, à droite de *PQ* il n'y a aucun site (car sinon l'angle serait trop grand). Et sur la partie gauche non plus, car elle est située à l'intérieur du cercle de diamètre *PQ*. On a bien trouvé un triangle de Delaunay situé à droite de *PQ*. On aurait aussi bien pu faire de même à gauche.

[4] A l'intérieur du cercle circonscrit au triangle *PSR*, il n'y a aucun autre site situé à droite de ***PR*** (l'angle serait trop grand), ni du côté gauche car cette partie est incluse dans le cercle circonscrit au triangle précédent *PRQ*.



- Deuxième cas : le point *P* est sur l'enveloppe convexe. On procède comme auparavant, mais on ne fait plus un tour complet. En tournant d'abord autour de *P* vers la droite, à partir de *Q*, on est bloqué lorsque l'on tombe sur l'enveloppe convexe. On met alors -1 dans la liste des voisins et l'on s'arrête. Dans l'exemple du dessin ci-dessous, on a comme liste de voisins 7 4 10 -1. Puis on repart de *Q* mais en tournant maintenant vers la gauche, jusqu'à être bloqué sur l'enveloppe convexe, d'où la liste 13 8 -1. On mélange enfin les deux listes de voisins pour avoir une seule liste en tournant vers la droite, soit dans l'exemple *voisins*[1][]= 8 13 7 4 10 -1, où *voisins*[1][0]=8 est le site adjacent à *P* (point 1) situé à sa gauche.

Dans le programme, si l'on tombe sur *vsd* = -1, on enregistrera le point *i* comme étant sur l'enveloppe convexe, et éventuellement, on pourra dessiner celle-ci par des jonctions entre *i* et *voisins*[*i*][0], soit :

```
for(i=0;i<N;i++)
if(envconv[i]==OUI) line(x[i],yorig-y[i],x[voisins[i][0]],yorig-y[voisins[i][0]],black);
```

Remarquons que cet algorithme constitue une démonstration constructive de l'existence et de l'unicité de la triangulation de Delaunay, sous réserve d'éviter toute ambiguïté : trois sites ne doivent pas être alignés, ni quatre situés sur le même cercle. Il existe d'autres algorithmes plus performants. Celui que nous présentons ici a le gros désavantage de donner plusieurs fois les mêmes triangles, puisque ceux-ci se trouvent dans les éventails de chacun de leurs trois sommets. Son avantage est d'obtenir pour le même prix l'enveloppe convexe, et surtout de permettre d'en déduire facilement le diagramme de Voronoi, grâce aux triangles de Delaunay adjacents entourant chaque site, que l'algorithme nous donne.

Pour réaliser le programme, on a besoin des deux fonctions suivantes:

- *plusprochevoisin*(*i*) ramène le site le plus proche du site *i* :

```
int plusprochevoisin (int i)
{ int j,jmin; long int dist2,  distmin=900000000 ;
  for(j=0;j<N;j++)  if (i!=j)
    { dist2=(x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j]);
      if (dist2<distmin) {distmin=dist2; jmin=j; }
    }
  return jmin;
}
```

- *voisinsuivantdroite*(*i*,*j*) ramène le site *vsd* tel que le triangle *i j vsd* soit un triangle de Delaunay situé à droite de l'arête *i j* obtenue au coup précédent, lors du parcours à droite en éventail autour du site *i* . S'il n'y a pas de voisin, on met *vsd* à -1. Cela permet de savoir si le point *i* est sur l'enveloppe convexe.

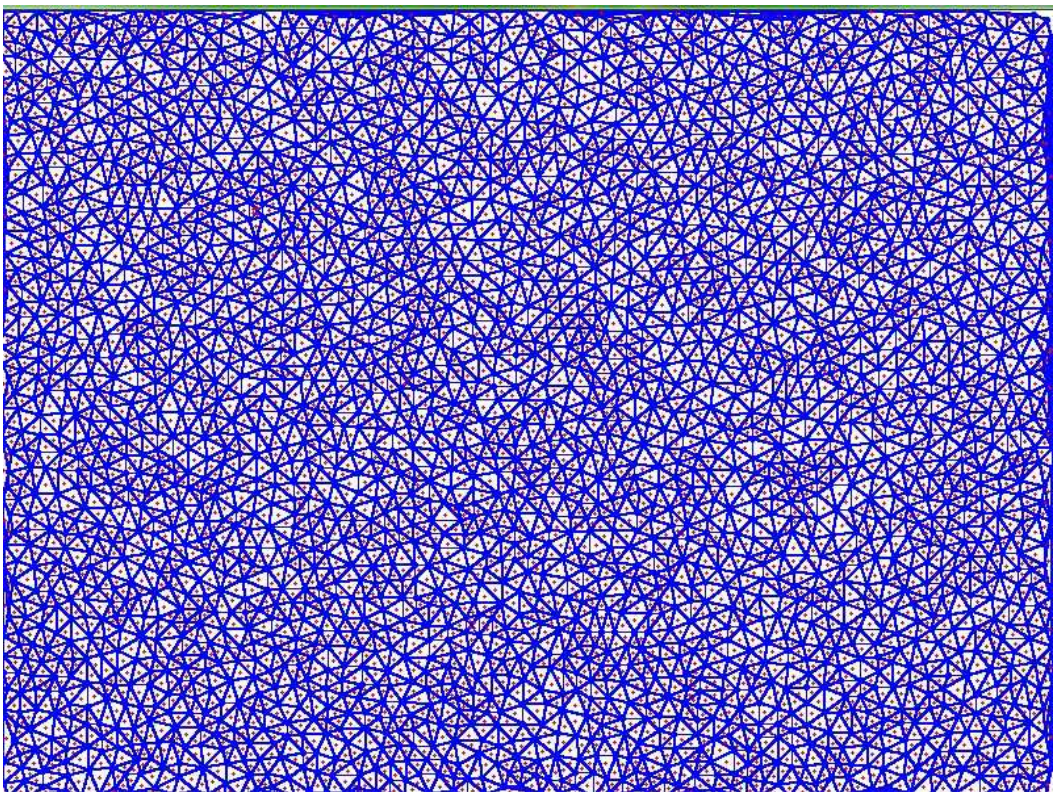
```
int voisinsuivantdroite(int i,int j)
{ int k; int v1x,v1y,v2x,v2y,numerovoisin=-1;
   double prodscal,longki2,longki,longkj2,longkj,coskij,cosmin,det;
```

```
  cosmin=1.;
  for(k=0;k<N;k++) if (k!=i && k!=j)
   { v1x=x[j]-x[i],v1y=y[j]-y[i];  v2x=x[k]-x[i],v2y=y[k]-y[i];  det=v1x*v2y-v1y*v2x;
     if (det<0)  /* on cherche un point à droite */
      { prodscal= (double)(x[i]-x[k])*(double)(x[j]-x[k])
                  +(double)(y[i]-y[k])*(double)(y[j]-y[k]);
        longki2=(double)(x[i]-x[k])*(double)(x[i]-x[k])
                +(double)(y[i]-y[k])*(double)(y[i]-y[k]);
        longkj2=(double)(x[j]-x[k])*(double)(x[j]-x[k])
                +(double)(y[j]-y[k])*(double)(y[j]-y[k]);
        longki=sqrt(longki2); longkj=sqrt(longkj2);
        coskij=prodscal/(longki*longkj); /* on veut le cosinus le plus petit possible */
        if (coskij<cosmin) {cosmin=coskij; numerovoisin=k;}
      }
    }
return numerovoisin;
}
```

On a aussi besoin d'une fonction *voisinsuivantdroite*() : elle est identique à la précédente, sauf que l'on fait maintenant le test : *if* (*det*>0).

Le programme principal en découle :

```
for(i=0;i<N;i++)  /* on prend chaque site */
 { ppv=plusprochevoisin(i); voisins[i][0]=ppv;  longueur[i]=1;   kk=0;
   do { vsd=voisinsuivantdroite(i,voisins[i][kk]); longueur[i]++; voisins[i][kk+1]=vsd; kk++;
      }
  while(vsd!=ppv && vsd!=-1);
  if (vsd==-1)
   { envconv[i]=1;
     do  { vsg=voisinsuivantgauche(i,voisins[i][0]);
           if (vsg!=-1)
             { for(kkk=longueur[i]-1;kkk>=0;kkk--)  voisins[i][kkk+1]=voisins[i][kkk];
               voisins[i][0]=vsg;      longueur[i]++;
             }
         }
     while(vsg!=-1);
   }
 }
 for(i=0;i<N;i++)  /* dessin des triangles autour de i */
  for(ii=0;ii<longueur[i]-1;ii++)  if (voisins[i][ii+1]!=-1)
   { line(x[i],yorig-y[i],x[voisins[i][ii]],yorig-y[voisins[i][ii]],blue);
     line(x[i],yorig-y[i],x[voisins[i][ii+1]],yorig-y[voisins[i][ii+1]],blue);
     line(x[voisins[i][ii]],yorig-y[voisins[i][ii]],x[voisins[i][ii+1]],
          yorig-y[voisins[i][ii+1]],blue);
   }
```



Triangulation de Delaunay pour 2800 points. Les points rouges ajoutés sont les centres de gravité des triangles, ils peuvent servir au coloriage futur des surfaces des triangles

Remarque : dans ce programme, nous traitons aussi bien les points qui ne sont pas sur l'enveloppe convexe, et ceux qui sont dessus. Mais à eux seuls, les points qui ne sont pas sur l'enveloppe, et dont on fait le tour complet, suffisent pour donner toute la triangulation.

Diagramme de Voronoï, et lien avec la triangulation de Delaunay

Reprenons notre ensemble formé de *N* sites (ou points). Rappelons que la cellule de Voronoï d'un site est l'ensemble des points du plan, formant une surface appelée cellule, qui sont plus prés de ce site que de tous les autres. Chaque cellule de Voronoï d'un site est un polygone convexe, ou plus largement une surface polygonale car la cellule d'un site situé sur l'enveloppe convexe n'est pas bornée. Le pavage du plan par les cellules de Voronoï des *N* sites forme ce que l'on appelle le diagramme de Voronoi.

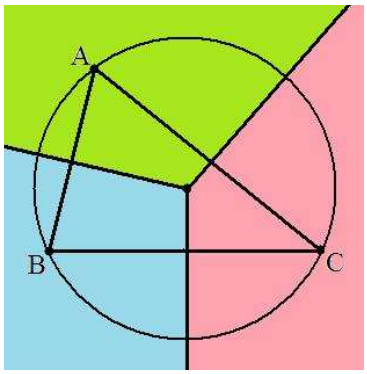
Signalons que G. Voronoï est un mathématicien russe des années 1900. Avant lui, d'autres personnes s'étaient intéressées à ce genre de problème, notamment Dirichlet (vers 1850) et surtout Descartes (vers 1640) dans son traité sur la lumière, à propos de la fragmentation du cosmos (voir *cours de géométrie et graphisme, chapitre* 1). Plus récemment, les cellules de Voronoi ont servi à l'étude de certains phénomènes naturels, et elles s'apparentent à la notion d'espace vital pour diverses espèces vivantes.

- Quelques exemples simples

1) Diagramme de Voronoi de deux points *A* et *B* : dans ce cas, les deux cellules forment deux demi-plans séparés par la médiatrice de [*AB*]. Rappelons que la médiatrice

de [*AB*] est la droite perpendiculaire à (*AB*) passant par le milieu de [*AB*]. Elle a comme caractéristique d'être l'ensemble des points situés à égale distance de *A* et de *B*, et elle coupe le plan en deux demi-plans, celui contenant l'ensemble des points plus prés de *A* que de *B*, et celui avec les points plus près de *B* que de *A*.

2) Cellules de Voronoi de trois points non alignés : il s'agit de trois secteurs angulaires délimités par trois demi-droites issues du centre du cercle circonscrit au triangle ABC, et qui sont les trois médiatrices de ce triangle.



Comment avoir le cercle circonscrit à un triangle *ABC*

Il s'agit du cercle (unique) passant par les trois sommets *A*, *B*, *C* du triangle. Son centre *C* est le point d'intersection des trois médiatrices du triangle. Pour l'avoir, il suffit de prendre le point d'intersection de deux médiatrices, par exemple celles des côtés *AB* et *AC*. Le vecteur ***AB*** a comme coordonnées *xab = xb – xa*, et *yab = yb – ya*, différence des coordonnées des points. De même avec ***AC*** (*xac*, *yac*). Le milieu *I* de [*AB*] a comme coordonnées *xi* =(*xa* + *xb*)/2, et *yi* = (*ya* + *yb*)/2. La médiatrice de [*AB*] passe par *I* et est perpendiculaire à *AB*. Avec *M* (*X*, *Y*) point courant sur cette droite, le fait d'avoir ***IM*** perpendiculaire à ***AB*** donne l'équation de la médiatrice :

$xab(X - xi) + yab(Y - yi)=0$.

De même pour la médiatrice de [*AC*] avec son milieu *J* (*xj*, *yj*). Leur point d'intersection *C* (*xc*, *yc*) est obtenu en résolvant le système des deux équations, ce qui a aboutit à :

$$xc = \frac{(xab\ xi + yab\ yi)yac - (xac\ xj + yac\ yj)yab}{\det}$$

$$yc = \frac{(xac\ xj + yac\ yj)xab - (xab\ xi + yab\ yi)xac}{\det} \quad \text{avec } \det = xab\ yac - xac\ yab$$

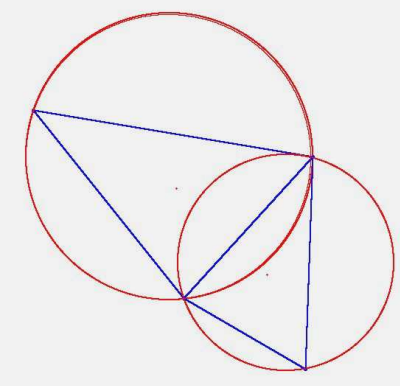
où *det* est le déterminant associé aux deux droites. Connaissant le point *C*, on en déduit le rayon *R = CA* du cercle circonscrit.

Dans le programme qui suit, la fonction *cerclecirconscrit*() prend comme variables les numéros des trois points du triangle, les coordonnées d'un point numéro *i* étant *x*[*i*] et *y*[*i*] données en entiers, correspondant aux pixels de l'écran (sans zoom, avec seulement l'origine des coordonnées en bas à gauche de l'écran : *yorig*=599).

```
void cerclecirconscrit(int a,int b,int c)
{ int xab, yab, xac, yac, xmab, ymab, xmac, ymac,rayon,det,centrex,centrey;
  float rayon2;
  xab=x[b]-x[a];yab=y[b]-y[a];  xac=x[c]-x[a]; yac=y[c]-y[a];
```

```
  xmab=(x[a]+x[b])/2; ymab=(y[a]+y[b])/2; xmac=(x[a]+x[c])/2; ymac=(y[a]+y[c])/2;
  det=xab*yac-yab*xac;
  centrex=(yac*(xab*xmab+yab*ymab) -yab*(xac*xmac+yac*ymac))/det;
  centrey=(-xac*(xab*xmab+yab*ymab) +xab*(xac*xmac+yac*ymac))/det;
  rayon2=(float)(x[a]-centrex)*(float)(x[a]-centrex)
          +(float)(y[a]-centrey)*(float)(y[a]-centrey);
  rayon=(int)sqrt(rayon2);
  circle(centrex,yorig-centrey,rayon,red);
 }
```

Remarque : il existe une autre méthode pour avoir le centre du cercle circonscrit à un triangle *ABC*. Ce point est le barycentre des trois points affectés respectivement des masses *sin* 2*A*, *sin* 2*B*, *sin* 2*C*.

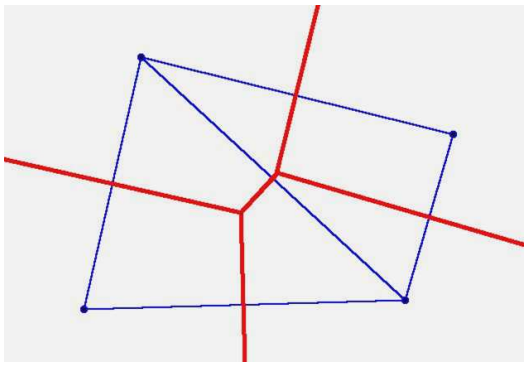


Deux triangles adjacents avec leurs cercles circonscrits. Pour les quatre sites qui sont les sommets des triangles, il s'agit d'une triangulation de Delaunay : aucun des deux cercles ne contient le quatrième sommet autre que ceux par lesquels passe chacun des cercles.

- Ce que nous avons fait avec deux ou trois points peut se généraliser. La bordure des cellules de Voronoi est toujours formée de morceaux de médiatrices. Pour chaque point, il suffit de prendre les médiatrices de ce point avec tous les autres. La cellule de Voronoi de ce point est la surface entourant le point et délimitée par les médiatrices, étant entendu que seules quelques-unes d'entre elles sont concernées pour former la bordure de la cellule.

- Lien avec la triangulation de Delaunay

Comme cela se démontre, les arêtes des cellules de Voronoi ne sont autres que des parties de médiatrices de triangles de Delaunay, et les sommets des cellules sont les centres des cercles circonscrits de triangles de Delaunay. Plus précisément, deux sommets extrémités d'une arête d'une cellule de Voronoi (quand on n'est pas sur la bordure) sont les centres de cercles circonscrits de deux triangles de Delaunay adjacents.



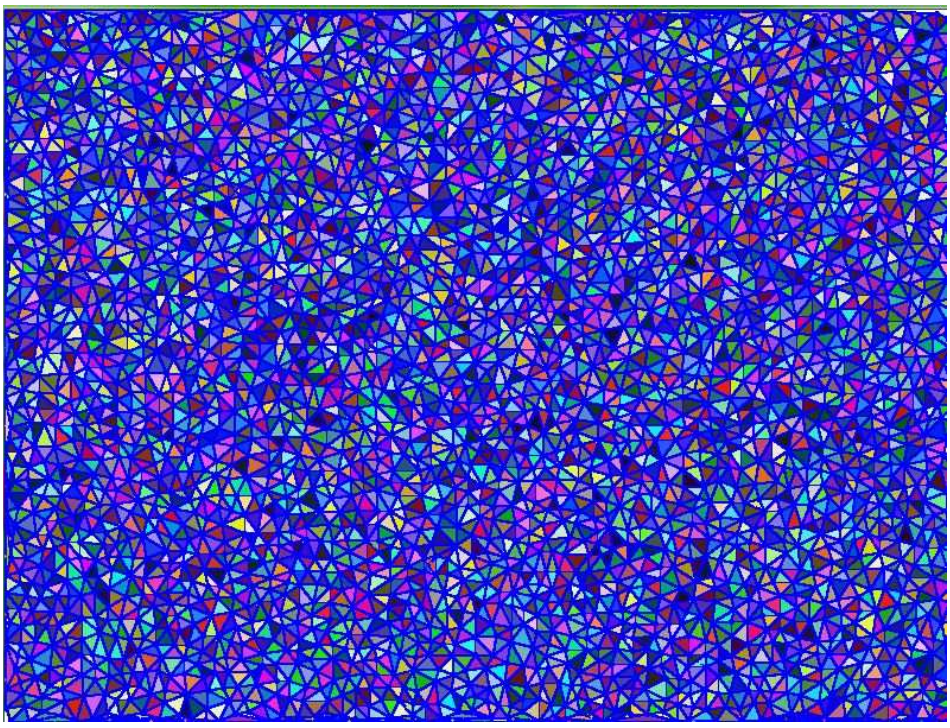
triangulation de Delaunay de quatre sites (*en bleu*)
et cellules de Voronoi (*en rouge*)

L'algorithme précédent nous donne justement des triangles de Delaunay adjacents. Il suffit de déterminer les centres des cercles circonscrits de ces triangles et de les joindre. Un éventail de triangles autour d'un site donne ainsi la cellule de Voronoi entourant le site par jonction fermée des centres des cercles circonscrits. Pour cela il suffit de reprendre le programme du cercle circonscrit, maintenant appelé *cc*() dans le programme qui suit, pour avoir les coordonnées de son centre (*centrex*, *centrey*), sans avoir besoin de le dessiner ni d'avoir besoin de son rayon. Reste le problème des éventails partiels, limités par l'enveloppe convexe. On connaît les sites situés sur cette enveloppe (*envconv*[*i*]=*OUI*) ainsi que le site qui leur est adjacent sur l'enveloppe (*voisins*[*i*][0]). Pour chacun de ces segments de l'enveloppe convexe, une des bordures séparant les deux cellules de Voronoi devient une demi-droite : il s'agit de la médiatrice dd ce segment, dont on prend la demi-droite partant du centre du cercle circonscrit du triangle de Delannay correspondant et allant vers l'extérieur de l'enveloppe convexe. On utilise pour cela une fonction demi-droite *dr*(*xx*, *yy*, *vvx*, *vvy*) partant d'un point *xx*, *yy*, et allant dans la direction et le sens du vecteur (*vvx*, *vvy*), ce vecteur étant perpendiculaire au vecteur joignant le site *i* et le site *voisins*[*i*][0].[5] Cette demi-droite doit aussi être bloquée aux limites de l'écran. On reprend pour cela ce que l'on a fait pour tracer des trajectoires dans un jeu de billard (*cours géométrie et graphisme, chapitre* 2). On aboutit ainsi au programme final pour les cellules de Voronoi se déduisant de la triangulation de Delaunay faite précédemment.

```
for(i=0;i<N;i++)
 {filldisc(x[i],yorig-y[i],2,black);
  for(ii=0;ii<longueur[i]-1;ii++) if (voisins[i][ii+1]!=-1)
   { cc(i,voisins[i][ii],voisins[i][ii+1]);  filldisc(centrex,yorig-centrey,1,black);
     if (ii==0) {cx0=centrex;cy0=centrey;}  /* le premier centre */
     if (ii>0) line(oldcx,yorig-oldcy,centrex,yorig-centrey,black);
     if (ii==longueur[i]-2) line(centrex,yorig-centrey,cx0,yorig-cy0,black);
     oldcx=centrex;oldcy=centrey;
     if (envconv[i]==1)
      { vvx=y[i]-y[voisins[i][0]]; vvy=x[voisins[i][0]]-x[i];
        if (cx0>0 && cx0<800 && cy0>0 && cy0<600) dr(cx0,cy0,vvx,vvy);
      }
   }
 }
```

[5] A partir d'un vecteur (*a*, *b*), un vecteur perpendiculaire est soit (-*b*, *a*) soit (*b*, -*a*). Il reste à choisir celui qui se déduit en tournant de +90°, tout en tenant compte du changement de repère pour les ordonnées.

```
/*********************** demi-droite ******************************/
void dr(int xx,int yy, int vx,int vy)
{ float k1,k2; int xextr,yextr;
  if (vx>0 && vy>0)
    { k1=(float)(L-xx)/(float)vx; k2=(float)(l-yy)/(float)vy;
      if (k1>k2) { xextr=xx+k2*vx;yextr=l;}
      else { xextr=L;yextr=yy+k1*vy;}
    }
  else if (vx<0 && vy<0)
    { k1=(float)-xx/(float)vx; k2=(float)(-yy)/(float)vy;
      if (k1>k2) { xextr=xx+k2*vx;yextr=0;}
     else { xextr=0;yextr=yy+k1*vy;}
    }
  else if (vx>0 && vy<0)
    { k1=(float)(L-xx)/(float)vx; k2=(float)(-yy)/(float)vy;
      if (k1>k2) { xextr=xx+k2*vx;yextr=0;}
     else { xextr=L;yextr=yy+k1*vy;}
    }
  else if (vx<0 && vy>0)
    { k1=(float)(-xx)/(float)vx; k2=(float)(l-yy)/(float)vy;
      if (k1>k2) { xextr=xx+k2*vx;yextr=l;}
      else { xextr=0;yextr=yy+k1*vy;}
    }
  else  if (vy==0 && vx>0) { xextr=799;yextr=xx;}
  else if  (vy==0 && vx<0) { xextr=0;yextr=xx;  }
  else if (vx==0 && vy>0) { xextr=xx;yextr=599;}
  else if (vx==0 && vy<0) { xextr=xx;yextr=0;}
  line(xx,yorig-yy,xextr,yorig-yextr,black);
}
```



Triangulation de Delaunay pour $N = 2800$ points, avec les triangles coloriés

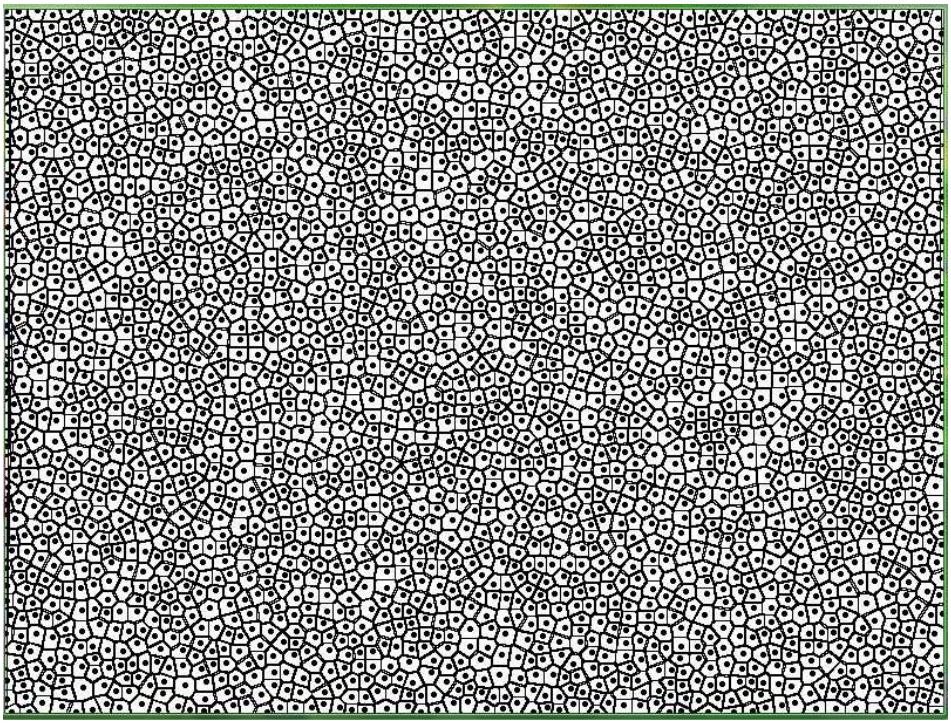


Diagramme de Voronoï déduit de la triangulation de Delaunay précédente