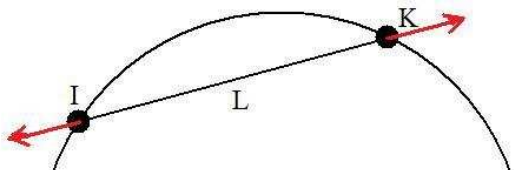
Mouvement de particules qui se repoussent, disposées sur des courbes ou des surfaces

1) Mouvement de particules sur un cercle

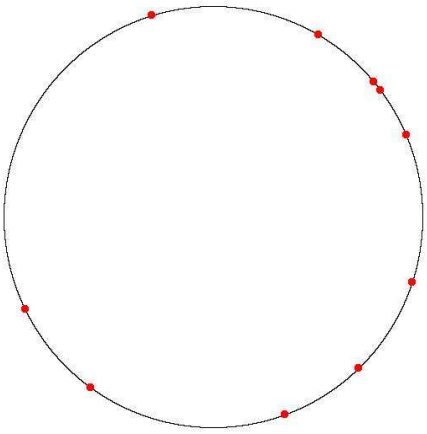
Des particules sont disposées au hasard sur un cercle. Chacune est soumise à des forces de répulsion provenant de toutes les autres particules. Elles vont alors se déplacer tout en assujetties à rester sur le cercle. Ainsi deux particules *I* et *K* quelconques se repoussent mutuellement avec une force inversement proportionnelle à la distance qui les sépare, portée par la droite (*IK*), et dont l'amplitude est de la forme $F = q\ m\ m' / L$, où *m* et *m'* sont les masses des particules et *q* un coefficient. Pour simplifier on prendra des particules ayant toutes la même masse $m = 1$.



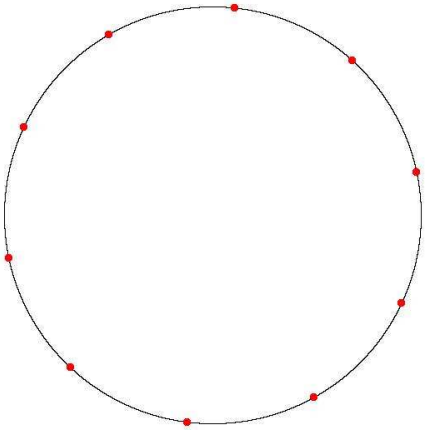
On envisage deux cas.

a) Lâchers par à-coups successifs

A partir de leur position initiale, on lâche les particules pendant un temps bref, de façon que celles-ci parcourent toutes un angle de 1°, puis on les bloque. Ensuite on recommence. Au terme d'un certain nombre de lâchers successifs, les particules finissent par trouver une position d'équilibre (en fait elles oscillent dans un sens et dans l'autre de 1° autour de cette position). A chaque lâcher, l'accélération $\boldsymbol{a}$ est créée par la force à laquelle chaque particule est soumise, et la vitesse, nulle au départ, varie de $\boldsymbol{dv} = \boldsymbol{a}\ dt$. Comme le mouvement se fait sur un cercle, seule la composante tangentielle de la force intervient, et comme on attend un mouvement d'un angle de 1°, il suffit de savoir dans quel sens il se fait.



Conditions initiales



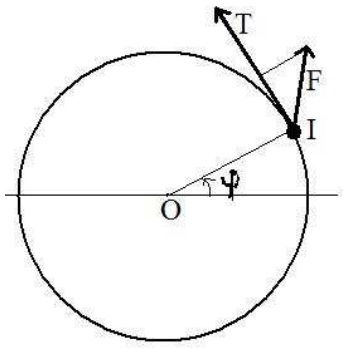
Arrivée à l'équilibre après de petits mouvements

Programme

On prend un cercle de rayon 1, pour simplifier, et l'on place dessus *N* particules au hasard. La position d'une particule *i* est connue par l'angle qui lui correspond sur le cercle (soit φ_i noté *phid*[*i*] lorsqu'il est en degrés, et *phir*[*i*] lorsqu'il est converti en radians. La particule *i* a pour coordonnées (cos φ_i , sin φ_i). Les particules sont ensuite dessinées sur le cercle sous forme de petits disques rouges.

```
cercle(xorig,yorig,zoom, noir);   /* le cercle dessiné sur l'écran avec un zoom */
for(i=0;i<N;i++)
  {  phid[i]=rand()%360;  phir[i]=phid[i]*pi/180.;
     disque(xorig+zoom*cos(phir[i]),yorig-zoom*sin(phir[i]),4,rouge);
  }
SDL_Flip(ecran); pause();
```

Puis on passe au mouvement, avec une boucle du temps, où à chaque fois les particules bougent d'un degré dans un sens ou dans l'autre selon l'action de la force qui imprime à chaque particule une petite vitesse. A chaque instant, on obtient un nouvel angle pour chaque particule. Pour avoir la force exercée sur chaque particule I, on fait la somme des forces de répulsion de toutes les autres particules. Pour une de ces particules K, on calcule le vecteur ***IK*** (cos φ_K – cos φ_I, sin φ_K – sin φ_I, puis sa longueur au carré IK^2. Le vecteur unitaire porté par ***IK*** est ***IK*** / *IK*, et comme la force est portée par ***IK*** dans le sens inverse, avec une amplitude de la forme *q* / *IK*, la force exercée par *K* sur *I* est – ***IK*** / IK^2. En cumulant ces forces, on obtient la force ***F*** (*fx*, *fy*). Au point *I*, le vecteur unitaire tangent est ***T*** (– sin φ_I , cos φ_I) et il est dirigé dans le sens trigonométrique (inverse de celui des aiguilles d'une montre). En prenant le produit scalaire ***T F***, on saura dans quel sens bouge le point I. S'il est positif, l'angle sera augmenté de 1°, et s'il est négatif, il diminue de 1°.




```
for(t=0;t<1000;t++)
{  cercle(xorig,yorig,zoom, noir);
   for(i=0;i<N;i++)
    {   fx=0.;fy=0.;
        for(k=0;k<N;k++)    if (k!=i)
          {  ikx=cos(phir[k])-cos(phir[i]);    iky=sin(phir[k])-sin(phir[i]);
             long2=ikx*ikx+iky*iky; if (long2==0.) long2=0.000000001;
             fx-=q*ikx/long2;    fy-=q*iky/long2;
          }
       if (-fx*sin(phir[i])+fy*cos(phir[i])>0.)  newphid[i]=phid[i]+1;
       else  if (-fx*sin(phir[i])+fy*cos(phir[i])<0.) newphid[i]=phid[i]-1;
       newphir[i]=newphid[i]*pi/180.;
    }
  SDL_Flip(ecran);  SDL_FillRect(ecran,0,blanc);
  for(i=0;i<N;i++)   /* on actualise */
       {  phir[i]=newphir[i]; phid[i]=newphid[i];
```

```
        disque(xorig+zoom*cos(phir[i]),yorig-zoom*sin(phir[i]),4,rouge);
      }
}
```

b) Mouvement continu, avec amortissement

Maintenant les particules bougent sous l'effet des forces qui leur sont appliquées. A l'instant initial les particules sont disposées au hasard sur le cercle, avec une vitesse angulaire vr_i nulle. Puis on divise le temps en intervalles *dt* très courts. A chacun de ces instants on calcule la force exercée sur chaque particule, comme précédemment, et l'on en déduit l'accélération angulaire (ou tangentielle) en projetant la force sur la tangente, toujours grâce au produit scalaire ***F T***. Cela permet de connaître la variation *dv* de la vitesse angulaire, et d'en déduire la nouvelle vitesse angulaire vr_i à chaque instant. Grâce à la vitesse, on trouve la nouvelle position de chaque particule. Pour éviter des oscillations infinies, on ajoute une force d'amortissement proportionnelle à la vitesse, soit ici - 0.4 vr_i . On en déduit le programme, qui permet d'observer les oscillations amorties des particules jusqu'à leur position d'équilibre, la même que celle obtenue dans le paragraphe précédent.

```
dt=0.002;
cercle(xorig,yorig,zoom, noir);
for(i=0;i<N;i++)     /* conditions initiales  */
{   phid[i]=rand()%360;  phir[i]=phid[i]*pi/180.;
    disque(xorig+zoom*cos(phir[i]),yorig-zoom*sin(phir[i]),4,rouge);
    vr[i]=0.;
}
SDL_Flip(ecran);   pause();

for(;;)      /* évolution au fil du temps  */
{  cercle(xorig,yorig,zoom, noir);
   for(i=0;i<N;i++)
     {   fx=0.;fy=0.;
         for(k=0;k<N;k++)   if (k!=i)
           {  ikx=cos(phir[k])-cos(phir[i]);  iky=sin(phir[k])-sin(phir[i]);
              long2=ikx*ikx+iky*iky; if (long2==0.) long2=0.000000001;
              fx-=q*ikx/long2;    fy-=q*iky/long2;
           }
         ar=-fx*sin(phir[i])+fy*cos(phir[i])-vr[i]*0.4;  /* accéleration tangentielle */
         vr[i]+=ar*dt;    /* vitesse angulaire */
         newphir[i]=phir[i]+vr[i]*dt;   /* nouvelle position de la particule i  */
     }
   for(i=0;i<N;i++)    /* actualisation et dessin  */
     { phir[i]=newphir[i];
       disque(xorig+zoom*cos(phir[i]),yorig-zoom*sin(phir[i]),4,rouge);
     }
   SDL_Flip(ecran);  SDL_FillRect(ecran,0,blanc);
}
```

2) Mouvement de particules sur une sphère

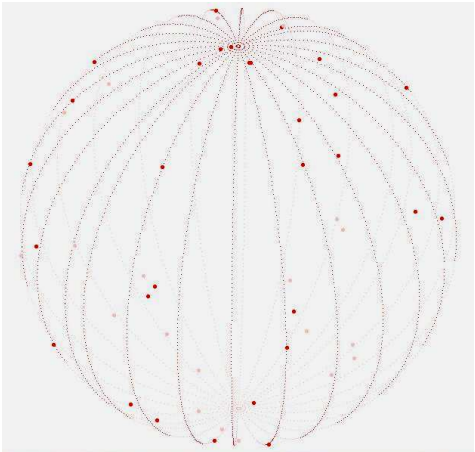
On a déjà vu le mouvement par lâchers successifs (*chapitre 4* : géométrie 3d). Nous traitons ici le mouvement continu. On commence par esquisser le dessin de la sphère en

prenant un certain nombre de points selon leur longitude et leur latitude, en distinguant les points visibles (en rouge vif) et ceux non visibles (en rouge pâle).

```
alpha=0.6 ;
c=sqrt(2.)*tan(alpha);  A=zoom/sqrt(2.);  B=zoom*sin(alpha)/sqrt(2.);  C=zoom*cos(alpha);
for(phi=0.; phi<deuxpi/2.;phi+=deuxpi/25.)  for(lambda=0.;lambda<deuxpi;lambda+=0.02)
  { x=R*cos(lambda)*cos(phi);y=R*cos(lambda)*sin(phi);z=R*sin(lambda);
    xe=xorig +A*(x-y);  ye=yorig-B*(x+y)- C*z;
    if (x+y-c*z<0.) putpixel(xe,ye,rouge);  else putpixel(xe,ye, rougepale);
  }
```

Puis on place au hasard *N* particules sur la sphère, dessinées sous forme de petits disques.

```
for(i=0;i<N;i++)
  {  ph[i]=(double)rand()/32768.*pi; la[i]=(double)rand()/32768.*deuxpi;
     px[i]=R*cos(la[i])*cos(ph[i]);  py[i]=R*cos(la[i])*sin(ph[i]);  pz[i]=R*sin(la[i]);
     xe=xorig +A*(px[i]-py[i]);  ye=yorig-B*(px[i]+py[i])- C*pz[i];
     if (px[i]+py[i]-c*pz[i]<0.)  disque(xe,ye,2,rouge);
    else  disque(xe,ye,2,rougepale);
  }
SDL_Flip(ecran); pause();
```

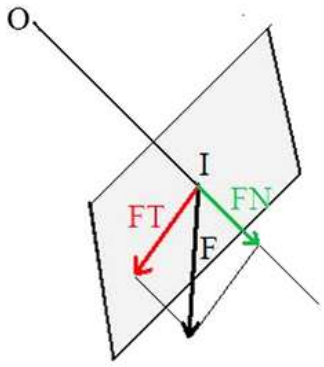


En ajoutant des vitesses nulles, on a les conditions initiales avant de lancer la boucle du temps, découpé en petits intervalles *dt*.

```
for(i=0;i<N;i++) { vx[i]=0.;vy[i]=0.; vz[i]=0.;}
dt=0.001;
```

Passons maintenant à l'évolution des particules au fil du temps. Au début de chaque étape du temps, on connaît la position et la vitesse de chaque particule. Il s'agit alors de déterminer l'accélération, puis la nouvelle position et la nouvelle vitesse des particules quand le temps augmente de *dt*. Chaque particule en position *I* (*px*[*i*], *py*[*i*], *pz*[*i*]) est soumise à la force de répulsion de toutes les autres *J*. Pour une particule *J* on prend le vecteur ***IJ*** et on le divise par sa longueur *r* pour obtenir un vecteur de longueur unité. Avec une force de répulsion inversement proportionnelle à la distance *r*, cela donne une force $-(\boldsymbol{IJ} / r) / r = -\boldsymbol{IJ} / r^2$. Puis on cumule les forces de toutes les particules *J* autres que *I*, d'où la force ***F*** (*fx*, *fy*, *fz*) agissant sur la particule *I*. Cette force se décompose en

une composante tangentielle et une composante normale, celle-ci étant compensée par la réaction de la surface sphérique. Mais il s'ajoute la force centripète d'amplitude v^2 / R, R étant le rayon de la sphère (nous prenons $R = 1$). Les seules forces agissantes sont la force tangentielle et la force centripète. Pour une particule en position I, la force centripète est alors $-\boldsymbol{OI}\, v^2$, avec O centre de la sphère, d'où $OI = 1$.



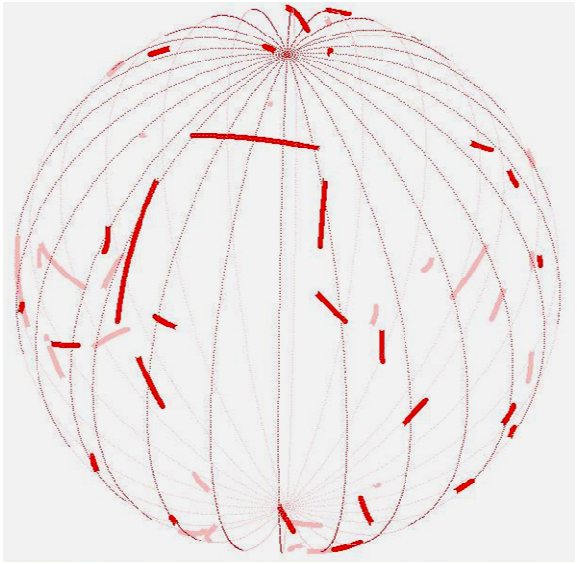
Il reste à projeter la force ***F*** sur le plan tangent à la sphère. On prend d'abord la composante normale ***FN*** portée par (*OI*), et qui vaut (***F OI***) ***OI***. Pour avoir la composante tangentielle ***FT***, on fait ***FT*** = ***F*** – ***FN***.

Finalement, chaque particule est soumise à une accélération ***A*** égale à l'addition de la force tangentielle et de la force centripète, la masse de chaque particule étant prise égale à 1. Cela permet d'obtenir la nouvelle position de la particule, ainsi que la nouvelle vitesse.

La programme s'ensuit. Mais si l'on se contente de dessiner les particules à chaque instant en conservant leurs positions antérieures la sphère va vite être recouverte de trajectoires et l'on ne voit plus rien. Si au contraire on dessine les particules à chaque instant en effaçant leurs positions précédentes, on ne voit pas bien le mouvement non plus. On va donc dessiner les particules avec une trainée derrière elles, correspondant à leur position antérieure pendant un nombre fixé d'étapes de temps (100 étapes de temps dans le programme). Comment faire ? On utilise ici un tableau *c*[*i*][*p*] de longueur 100 pour chaque particule *i*, dans lequel vont se trouver les coordonnées écran *xe* et *ye* de la particule. Jusqu'à l'étape de temps 99, le tableau est rempli avec les positions successives de la particule, de 0 à 99, et on les envoie sur l'écran. Puis à l'étape 100, la position 0 de la particule est effacée, et remplacée par la position 100. La lecture des 100 cases du tableau donne un dessin correspondant à la traînée. De même à l'étape 101, on efface la position 1 et l'on met à la place la position 101. Le tableau contient les positions 100, 101, 2, 3, …, 99, et leur dessin simultané donne la traînée derrière la position 101.Et ainsi de suite, avec un parcours cyclique du tableau grâce à une variable *p* qui parcourt le tableau et se remet à 0 quand elle atteint la fin du tableau. Autrement dit, à chaque nouvelle étape, on efface la queue - la dernière position de la traînée- (d'où dans le programme *cercle*(*cx*[*i*][*p*], *cy*[*i*][*p*], 2, *blanc*)) et on lui ajoute une nouvelle tête (on actualise par *cx*[*i*][*p*] = *xe* et *cy*[*i*][*p*] = *ye* et on fait *cercle*(*cx*[*i*][*p*], *cy*[*i*][*p*], 2, *rouge*)). Dernier détail : comme le mouvement des particules efface peu à peu tous les points de la sphère, on décide de la redessiner régulièrement.

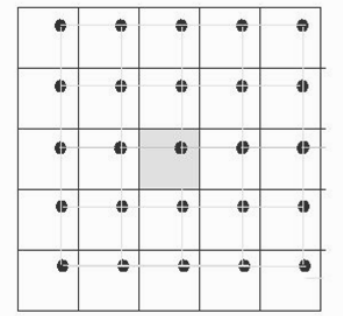
```
for(etape=0;etape<15000;etape++)
{  for(i=0;i<N;i++)
      { xe=xorig +A*(px[i]-py[i]);  ye=yorig-B*(px[i]+py[i])- C*pz[i];
        if (etape>=100) cercle(cx[i][p],cy[i][p],2,blanc);
       cx[i][p]=xe; cy[i][p]=ye;
       if (px[i]+py[i]-c*pz[i]<0.) cercle(cx[i][p],cy[i][p],2,rouge);
       else cercle(cx[i][p],cy[i][p],2,rougepale);
       fx=0.;fy=0.;fz=0.;
       for(j=0;j<N;j++)  if (j!=i)  /* cumul des forces de répulsion sur la particule i  */
        {  r2=(px[j]-px[i])*(px[j]-px[i])+(py[j]-py[i])*(py[j]-py[i])+(pz[j]-pz[i])*(pz[j]-pz[i]);
           fx-=(px[j]-px[i])/r2;fy-=(py[j]-py[i])/r2;  fz-=(pz[j]-pz[i])/r2;
        }
```

```
      ps=fx*px[i]+fy*py[i]+fz*pz[i];  /* produit scalaire  F  OI  */
      ftx=fx-ps*px[i];  fty=fy-ps*py[i];  ftz=fz-ps*pz[i];  /* force tangentielle */
      v2=vx[i]*vx[i]+vy[i]*vy[i]+vz[i]*vz[i];   /* vitesse au carré  */
      accx=ftx-px[i]*v2; accy=fty-py[i]*v2; accz=ftz-pz[i]*v2;;  /* acceleration  */
      px[i]+=vx[i]*dt+0.5*accx*dt*dt;;  py[i]+=vy[i]*dt+0.5*accy*dt*dt;
      pz[i]+=vz[i]*dt+0.5*accz*dt*dt;   /* nouvelle position  */
      vx[i]+=accx*dt; vy[i]+=accy*dt; vz[i]+=accz*dt;  /* nouvelle vitesse  */
     }
  if(p==99)   for(phi=0.; phi<deuxpi/2.;phi+=deuxpi/25.)  /* re-dessin de la sphere */
  for(lambda=0.;lambda<deuxpi;lambda+=0.01)
    { x=R*cos(lambda)*cos(phi);y=R*cos(lambda)*sin(phi);z=R*sin(lambda);
      xe=xorig +A*(x-y);  ye=yorig-B*(x+y)- C*z;
      if (x+y-c*z<0.) putpixel(xe,ye,rouge);  else putpixel(xe,ye, rougepale);
    }
 SDL_Flip(ecran);    p++;    if (p==100) p=0;
}
```



3) Mouvements sur un tore

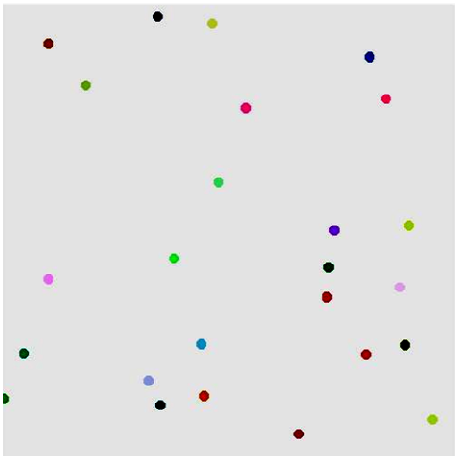
En fait il s'agit de circuler dans un carré dont les côtés sont cycliques : si une particule dépasse la limite à droite, elle est mise à gauche, et de même en haut et en bas. A son tour, chaque particule est démultipliée dans les carrés qui entourent le carré initial. Une particule est en fait répétée une infinité de fois, mais comme on prend une force de répulsion en $1 / r^2$, on se contente de prendre quelques carrés du voisinage (ici cela fait 25 carrés). Chaque particule est ainsi repoussée par toutes les autres, elles-mêmes répétées 25 fois. Cela correspond à ce qui se passerait sur un tore (un pneu) : une particule est reliée à une autre par plusieurs chemins sur la surface du tore, selon le nombre de tours que l'on fait sur lui. Mais en faisant cela on suppose que les forces sont portées par ces lignes courbes, ce qui est moins réaliste.

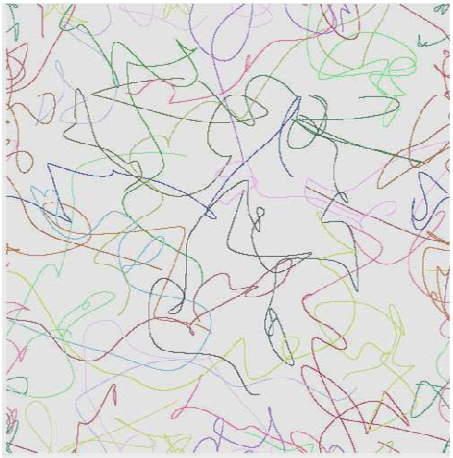


Dans le programme qui suit, on travaille dans un carré de côté 1, on prend une force de répulsion de la forme k / r^2, et on lui ajoute une force d'amortissement $– q \boldsymbol{V}$, afin d'atteindre un état d'équilibre, à partir de conditions initiales prises au hasard. On constate expérimentalement que plusieurs états d'équilibre sont possibles. Pour avoir les coordonnées des reproductions de chaque particule, on utilise les tableaux pré-calculés *bx*[] et *by*[]. Les coordonnées de la particule *j* et de ses duplications sont *x*[*j*] + *bx*[*k*] et *y*[*j*] + *by*[*k*], avec *k* de 0 à 25.

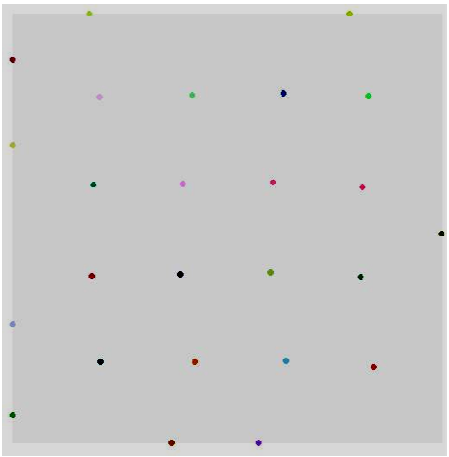
```
#define N 25    /* nombre de particules  */
#define q 0.5   /* coefficient d'amortissement  */
#define kk 0.1  /* coefficient de la force de repulsion  */
#define xorig 80
#define yorig 80
#define zoom 500.
int bx[25]={0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2,-1,-1,-1,-1,-1,-2,-2,-2,-2,-2};
int by[25]={0, 1, 2,-1,-2, 0, 1, 2,-1,-2, 0, 1, 2,-1,-2, 0, 1, 2,-1,-2, 0, 1, 2,-1,-2  };

SDL_Init(SDL_INIT_VIDEO);
ecran=SDL_SetVideoMode(800,600,32,SDL_HWSURFACE|SDL_DOUBLEBUF);
blanc=SDL_MapRGB(ecran->format,255,255,255);
couleurfond=blanc; couleurcarre=SDL_MapRGB(ecran->format,240,240,240);
for (i=0;i<N;i++)
co[i]=SDL_MapRGB(ecran->format,rand()%256,rand()%256,rand()%256);
SDL_FillRect(ecran,0,couleurfond);   srand(time(NULL));
rectangle= SDL_CreateRGBSurface(SDL_HWSURFACE,zoom,zoom,32,0,0,0,0);
position.x=xorig;position.y=yorig ;
SDL_FillRect(rectangle,NULL,couleurcarre);
SDL_BlitSurface(rectangle,NULL,ecran,&position);  /* carré cyclique initial */
for(i=0;i<N;i++)   /* mise en place des particules au hasard à l'instant initial  */
    { x[i]=(float)rand()/32768.;y[i]=(float)rand()/32768.;  }
for(i=0;i<N;i++)
    { vx[i]=0.;vy[i]=0.;   disque(xorig+zoom*x[i],yorig+zoom*y[i],5,co[i]);  }
dt=0.0005;
for(etape=0;etape<100000; etape++)   /* boucle du temps  */
{   for(i=0;i<N;i++)
      {  fx=0.;fy=0.;
        for(j=0;j<N;j++)  if(i!=j)    for(k=0;k<25;k++)
          {  xj=x[j]+bx[k];yj=y[j]+by[k];
             dij=sqrt((xj-x[i])*(xj-x[i])+(yj-y[i])*(yj-y[i]));
             if (dij<0.00001) dij=0.00001;
             d3=dij*dij*dij;
             fx-=kk*(xj-x[i])/d3;    fy-=kk*(yj-y[i])/d3;
          }
        ax=fx-q*vx[i];ay=fy-q*vy[i];
        x[i]+=vx[i]*dt+0.5*ax*dt*dt;    y[i]+=vy[i]*dt+0.5*ay*dt*dt;
        if (x[i]<0.) x[i]+=1.;   if (x[i]>=1.) x[i]-=1.;
        if (y[i]<0.) y[i]+=1.;   if (y[i]>=1.) y[i]-=1.;
        vx[i]+=ax*dt; vy[i]+=ay*dt;
        putpixel(xorig+zoom*x[i],yorig+zoom*y[i],co[i]);
      }
if (etape%100==99) SDL_Flip(ecran);
}
```

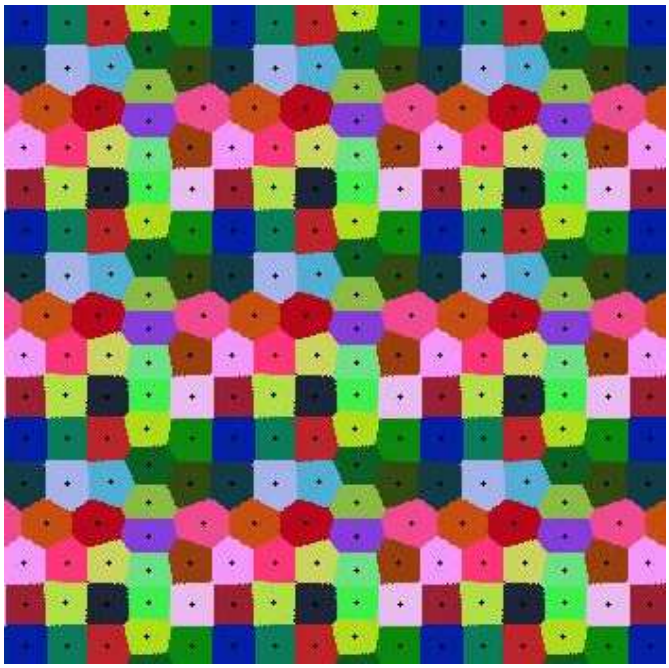
puis afficher les particules dans leur position finale sur le carré cyclique





Conditions initiales (*à gauche*), puis mouvement amorti des N=25 particules





Etat d'équilibre final (*à gauche*) et démultiplication des particules *à droite*, avec les cellules de Voronoi des particules lorsque l'équilibre est atteint