Mouvement d'une particule dans une cuvette

Cet exercice a deux objectifs :

- dessiner une cuvette sphérique
- faire évoluer une particule pesante dans cette cuvette. Pour cela nous utiliserons les caractéristiques de la sphère, mais nous donnerons aussi une méthode générale s'appliquant à toute forme de surface.

1) Dessin d'une cuvette sphérique

a) *Points sur la sphère*

On commence par tapisser la sphère par des points ($x[k]$, $y[k]$, $z[k]$) régulièrement disposés suivant leur longitude et leur latitude, comme on l'a fait dans les chapitres 4 et 7 du cours *graphisme et géométrie*. Puis on dessine ces points en perspective sur l'écran, avec leurs coordonnées écran ($xe[k]$, $ye[k]$), en appliquant les formules de passage vues précédemment.

```
 déclarations à faire
couleurfond=SDL_MapRGB(ecran->format,20,0,0);
for (i=0;i<256;i++) co[i]=SDL_MapRGB(ecran->format,i,0,0);
SDL_FillRect(ecran,0,couleurfond);
alpha=pi/4. ;  /* angle de vision */
c=sqrt(2.)*tan(alpha);A=zoom/sqrt(2.);B=zoom*sin(alpha)/sqrt(2.);C=zoom*cos(alpha);
phi=0.;k=0;dp=2.*pi/(float)N;dl=(pi - 2.*ll)/(float)NN;
for (i=0; i<N; i++)
{   lambda= -pi/2.+ll;
   for (j=0;j<=NN;j++)
   {  x[k]=R*cos(lambda)*cos(phi);  y[k]=R*cos(lambda)*sin(phi);  z[k]=R*sin(lambda);
      dist[k]=x[k]+y[k]-c*z[k]; if (dist[k]<0.) coleur=rouge;  else coleur=vert;
      xe[k]=xorig +A*(x[k]-y[k]);  ye[k]=yorig-B*(x[k]+y[k])- C*z[k];
      putpixel(xe[k],ye[k],coleur);
      lambda+=dl;  k++;
   }
   phi+=dp;
}
SDL_Flip(ecran);  puis effacer
```

b) Facettes de la cuvette sphérique

Des points précédents, on ne garde que ceux qui sont dans la partie basse de la sphère, celle qui forme la cuvette. Pour avoir une demi-sphère, on prend tous les indices k des points tels que : $k\%(NN+1)<NN/2$. On distingue trois zones :

- les points de la partie arrière de la cuvette avec ses facettes rectangulaires
- les points au voisinage du pôle sud avec ses facettes triangulaires.
- les points de la partie avant de la cuvette.

```
for (k=0;k<NNN;k++)
   {  if (dist[k]>0. && k%(NN+1)<NN/2)  /* partie arrière */
      {  ligne( xe[k],ye[k],xe[k+1],ye[k+1],vert);
```

```
      ligne( xe[k],ye[k],xe[(k+NN+1)%NNN],ye[(k+NN+1)%NNN],vert);
      ligne(xe[(k+NN+1)%NNN],ye[(k+NN+1)%NNN],
            xe[(k+NN+2)%NNN],ye[(k+NN+2)%NNN],vert);
      ligne(xe[k+1],ye[k+1],xe[(k+NN+2)%NNN],ye[(k+NN+2)%NNN],vert);
    }
    if (dist[k]>0 && k%(NN+1)==0)  /* pôle sud  */
    {  ligne( xe[k],ye[k],xorig,yorig+C,vert);
      ligne(xe[(k+NN+1)%NNN],ye[(k+NN+1)%NNN],xorig,yorig+C,vert);
    }
    if ( dist[k]<0. && k%(NN+1)<NN/2)  /* partie avant */
    {  ligne( xe[k],ye[k],xe[k+1],ye[k+1],rouge);
      ligne( xe[k],ye[k],xe[(k+NN+1)%NNN],ye[(k+NN+1)%NNN],rouge);
      ligne(xe[(k+NN+1)%NNN],ye[(k+NN+1)%NNN],
            xe[(k+NN+2)%NNN],ye[(k+NN+2)%NNN],rouge);
      ligne(xe[k+1],ye[k+1],xe[(k+NN+2)%NNN],ye[(k+NN+2)%NNN],rouge);
    }
  }
```

c) Remplissage des facettes avec les ombres

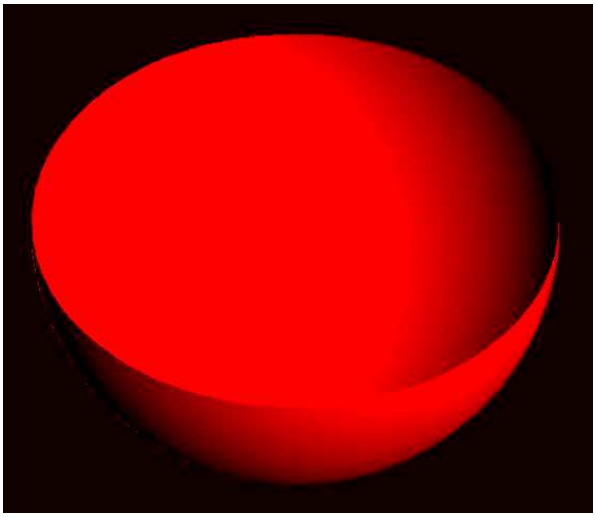
On commence par dessiner la partie arrière de la cuvette (*dist*[*k*]>0 && *k*%(*NN*+1)<*NN*/2), en prenant les normales aux facettes dirigées non plus vers l'extérieur, mais vers l'intérieur, puisque la lumière vient frapper cette surface arrière. On considère non seulement les facettes rectangulaires mais aussi les facettes triangulaires du pôle sud.

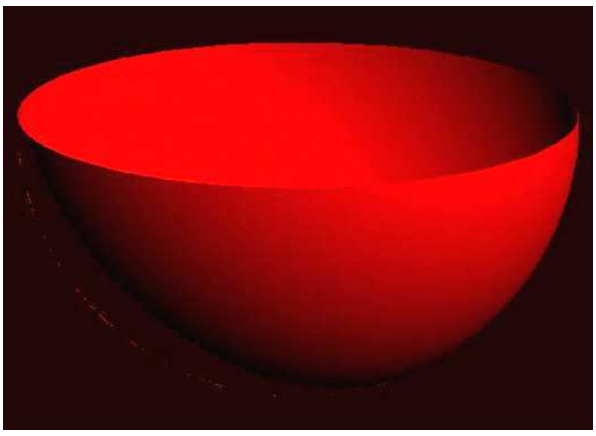
```
 /* vecteur S dirigé vers le soleil */
 sx=0.;sy=-1.; sz=1.;ls=sqrt(sx*sx+sy*sy+sz*sz); sx/=ls; sy/=ls; sz/=ls;

for (k=0;k<NNN;k++)   if ( dist[k]>0.&& k%(NN+1)<NN/2.)
    {  vx=x[(k+NN+1)%NNN]-x[k]; vy=y[(k+NN+1)%NNN]-y[k];
       vz=z[(k+NN+1)%NNN]-z[k];
       vvx=x[k+1]-x[k]; vvy=y[k+1]-y[k]; vvz=z[k+1]-z[k];
       nx=vy*vvz-vvy*vz; ny=vz*vvx-vvz*vx; nz=vx*vvy-vvx*vy;
       ln=sqrt(nx*nx+ny*ny+nz*nz);
       nx/=ln; ny/=ln; nz/=ln; nx=-nx;ny=-ny;nz=-nz;
       icolor=60.+180. *(nx*sx+ny*sy+nz*sz);
       xq[0]=xe[k];yq[0]=ye[k];
       xq[1]=xe[(k+NN+1)%NNN];yq[1]=ye[(k+NN+1)%NNN];
       xq[2]=xe[(k+NN+2)%NNN]; yq[2]=ye[(k+NN+2)%NNN];
       xq[3]=xe[k+1];yq[3]=ye[k+1];
       if (icolor>0)   remplirquadri(co[icolor]);  else remplirquadri(couleurfond);
       if  ( k%(NN+1)==0)
         { vx=x[(k+NN+1)%NNN]-x[k]; vy=y[(k+NN+1)%NNN]-y[k];
           vz=z[(k+NN+1)%NNN]-z[k];
           vvx=x[k]; vvy=y[k]; vvz=z[k]+1.;
           nx=vy*vvz-vvy*vz; ny=vz*vvx-vvz*vx; nz=vx*vvy-vvx*vy;
           ln=sqrt(nx*nx+ny*ny+nz*nz);nx/=ln; ny/=ln; nz/=ln;nx=-nx;ny=-ny;nz=-nz;
           icolor=60.+180. *(nx*sx+ny*sy+nz*sz);
           xq[0]=xe[k];yq[0]=ye[k];
           xq[1]=xe[(k+NN+1)%NNN];yq[1]=ye[(k+NN+1)%NNN];
           xq[2]=xorig; yq[2]=yorig+C;
           if (icolor>=0) remplirtriangle(co[icolor]);    else remplirtriangle(couleurfond);
         }
```

```
    }
  SDL_Flip(ecran);
```

Puis on dessine la partie avant de la cuvette, qui va recouvrir partiellement la partie arrière.

```
  for (k=0;k<NNN;k++)  if ( dist[k]<0.&& k%(NN+1)<NN/2.)
     {  vx=x[(k+NN+1)%NNN]-x[k]; vy=y[(k+NN+1)%NNN]-y[k];
        vz=z[(k+NN+1)%NNN]-z[k];
        vvx=x[k+1]-x[k]; vvy=y[k+1]-y[k]; vvz=z[k+1]-z[k];
        nx=vy*vvz-vvy*vz; ny=vz*vvx-vvz*vx; nz=vx*vvy-vvx*vy;
        ln=sqrt(nx*nx+ny*ny+nz*nz);
        nx/=ln; ny/=ln; nz/=ln;
        icolor=60.+180. *(nx*sx+ny*sy+nz*sz);
        xq[0]=xe[k];yq[0]=ye[k];
        xq[1]=xe[(k+NN+1)%NNN];yq[1]=ye[(k+NN+1)%NNN];
        xq[2]=xe[(k+NN+2)%NNN]; yq[2]=ye[(k+NN+2)%NNN];
        xq[3]=xe[k+1];yq[3]=ye[k+1];
        if (icolor>0)   remplirquadri(co[icolor]);
        else remplirquadri(couleurfond);
     }

 }
```





2) Trajectoire d'une particule pesante

a) Equation de la surface

Afin de généraliser ce qui est fait avec la cuvette sphérique, on considère que la surface sur laquelle se déplace la particule est connue par son équation, de la forme $Z = f(X, Y)$.. Dans le cas présent, l'équation de la sphère est $X^2 + Y^2 + Z^2 = R^2$, ou encore $Z^2 = R^2 - X^2 - Y^2$, et celle de la cuvette sphérique est $Z = -\sqrt{R^2 - X^2 - Y^2}$. D'où la fonction programmée :

```
double f(double XX, double YY)
{ double ZZ;
   ZZ=-sqrt(R*R-XX*XX-YY*YY); return ZZ;
}
```

b) Plan tangent à la surface en un point, et son vecteur normal

Nous donnons ici une méthode générale, pour une surface quelconque. Dans le cas particulier de la sphère, il existerait une méthode beaucoup plus simple. On se place en un point M (X, Y, $f(X, Y)$) de la surface. Pour avoir le plan tangent en M, on prend deux points très proches de M, $M_1(X_1=X+dX,\ Y_1=Y,\ Z_1=f(X+dX,\ Y))$ et $M'_1(X,\ Y+dY,\ f(X, Y+dY))$, et pour améliorer la précision, deux autres points de l'autre côté, M_2 ($X_2=X-dX$, $Y_2=Y$, $Z_2=f(X-dX, Y)$) et M'_2 (X, $Y-dY$, $f(X, Y-dY)$). Les deux vecteurs $\boldsymbol{M_2M_1}$ et $\mathbf{M'_2M'_1}$ déterminent le plan tangent. Pour avoir un vecteur perpendiculaire au plan, on fait le produit vectoriel de ces deux vecteurs, et l'on obtient $\boldsymbol{N}$ (nx, ny, nz) de coordonnées $-(Z_1 - Z_2).2dY$, $-(Z'_1 - Z'_2).2dX$, $4dX\ dY$, ou encore $-(Z_1 - Z_2) / 2dX$, $-(Z'_1 - Z'_2) / 2dX$, 1. On lui donne enfin la longueur 1. Précisons que ce vecteur est orienté vers l'intérieur de la sphère, à cause du produit vectoriel (règle du tire-bouchon). Si l'on veut la normale dirigée vers l'extérieur (vers le bas pour la demi-sphère), on prend $(Z_1 - Z_2) / 2dX$, $(Z'_1 - Z'_2) / 2dX$, -1.

c) Evolution dans le temps : première méthode

La force verticale est le poids, d'amplitude $-\ m\ g$ (on prend $m = 1$ et $g = 1$). On ajoute éventuellement une force de frottement $-\ q\ \boldsymbol{V}$ proportionnelle à la vitesse. Cela donne la force $\boldsymbol{F}$ ($-\ q\ VX$, $-\ q\ VY$, $-1 - q\ VZ$). On la projette sur le plan tangent, d'où $\boldsymbol{FT} = \boldsymbol{F} - (\boldsymbol{F}\ \boldsymbol{N})\ \boldsymbol{N}$. D'où $\boldsymbol{FT}$ (FTX, FTY, FTZ). Mais Il faut aussi tenir compte de la force centripète, perpendiculaire au plan tangent, soit $\boldsymbol{FN} = -(V^2 / R)\ \boldsymbol{N}$, où R est le rayon de courbure de la trajectoire, soit dans le cas présent le rayon de la sphère. Dans le programme on a pris la normale $\boldsymbol{N}$ dirigée vers le bas, d'où la présence d'un signe – dans $\boldsymbol{FN}$. Cela donne les composantes tangentielle et normale de l'accélération $\boldsymbol{A}$, d'où finalement (AX, AY, AZ).

Connaissant la position (X, Y, Z) de la particule et sa vitesse (VX, VY, VZ) à un instant t donné, on calcule l'accélération, puis la nouvelle position à l'instant $t+dt$, soit $X+=VX*dt$, et de même pour les autres coordonnées. puis la nouvelle vitesse, soit $VX+=AX*dt$, etc. D'où le programme :

```
dt=0.0001;  dX=0.0001; dY=0.0001;
X=0.95;Y=0.; Z=f(X,Y);  /* conditions initiales */
```

```
VX=0.;VY=0.703;VZ=0.;
for( temps=0;temps<1500000; temps++)   /* boucle du temps */
  {  putpixel(xorig+A*(X-Y), yorig-B*(X+Y)-C*Z,blanc);
     FX=-q*VX; FY=-q*VY; FZ=-1.-q*VZ;
     X1=X+dX;Y1=Y; Z1=f(X1,Y1);  X2=X-dX;Y2=Y; Z2=f(X2,Y2);
     XX1=X; YY1=Y+dY; ZZ1=f(XX1,YY1);   XX2=X; YY2=Y-dY; ZZ2=f(XX2,YY2);
     NPX=(Z1-Z2)/(2.*dX); NPY=(ZZ1-ZZ2)/(2.*dY); NPZ=-1.;
     LNP=sqrt(NPX*NPX+NPY*NPY+NPZ*NPZ);
     NPX/=LNP; NPY/=LNP; NPZ/=LNP;

     FN=FX*NPX+FY*NPY+FZ*NPZ;  /* produit scalaire F N  */
     FTX=FX-FN*NPX; FTY=FY-FN*NPY;  FTZ=FZ-FN*NPZ;
     VIT=sqrt(VX*VX+VY*VY+VZ*VZ);  /* amplitude de la vitesse */
     AX=FTX-NPX*VIT*VIT; AY=FTY-NPY*VIT*VIT; AZ=FTZ-NPZ*VIT*VIT;
     X+=VX*dt+0.5*AX*dt*dt; Y+=VY*dt+0.5*AY*dt*dt; Z+=VZ*dt+0.5*AZ*dt*dt;
    VX+=AX*dt; VY+=AY*dt; VZ+=AZ*dt;
  }
SDL_Flip(ecran);pause();
```

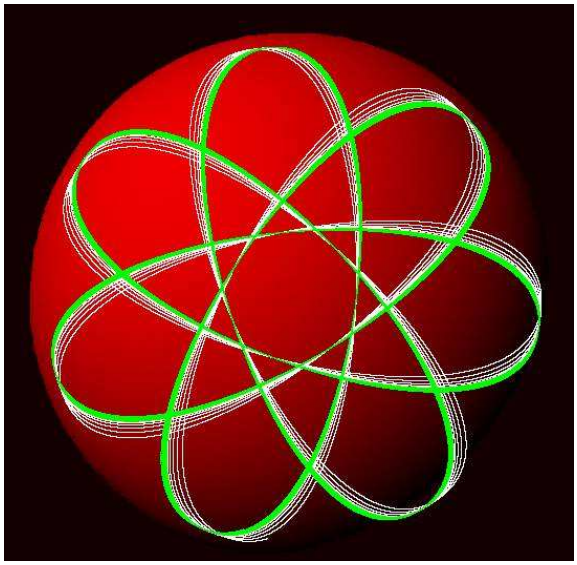
d) Deuxième méthode, plus générale

La méthode précédente ne marche que pour la sphère car on connaît dans ce cas le rayon de courbure *R*. Mais comment faire pour une surface quelconque ? Pour éviter une grosse complication dans les calculs, avec le calcul de *R*, on va se concentrer sur la vitesse tangentielle, celle qui donne son mouvement à la particule. [1] On n'a plus à considérer la force centripète, on prend seulement l'accélération provoquée par le poids et les frottements, soit $\boldsymbol{A}$ ($-q\,VX$, $-q\,VY$, $-1-q\,VZ$). A partir de la position du point et de sa vitesse à l'instant *t*, on calcule la projection de la vitesse sur le plan tangent, soit $\boldsymbol{VT}$ (*VTX*, *VTY*, *VTZ*) avec $\boldsymbol{VT} = \boldsymbol{V} - (\boldsymbol{VN})\,\boldsymbol{N}$. Celle-ci permet d'avoir la nouvelle position du point, par $X += VTX\,dt$, et de même pour les autres coordonnées. Puis la nouvelle vitesse s'obtient en faisant $\boldsymbol{V} = \boldsymbol{VT} + \boldsymbol{A}\,dt$.

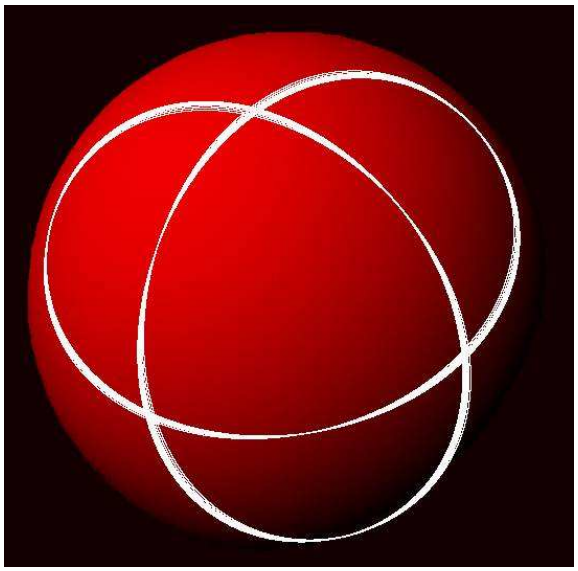
```
dt=0.0001;  dX=0.0001; dY=0.0001;
X=0.95;Y=0.;Z=f(X,Y);   VX=0.;VY=0.703;VZ=0.;
for( temps=0;temps<1500000; temps++)
  {   putpixel(xorig+A*(X-Y), yorig-B*(X+Y)-C*Z,vert);
     AX=-q*VX; AY=-q*VY; AZ=-1.-q*VZ;
     X1=X+dX;Y1=Y; Z1=f(X1,Y1);   X2=X-dX;Y2=Y; Z2=f(X2,Y2);
     XX1=X; YY1=Y+dY; ZZ1=f(XX1,YY1);   XX2=X; YY2=Y-dY; ZZ2=f(XX2,YY2);
     NPX=-(Z1-Z2)/(2.*dX); NPY=-(ZZ1-ZZ2)/(2.*dY); NPZ=1.;
     LNP=sqrt(NPX*NPX+NPY*NPY+NPZ*NPZ);
     NPX/=LNP; NPY/=LNP; NPZ/=LNP;
     VN=VX*NPX+VY*NPY+VZ*NPZ;
     VTX=VX-VN*NPX;   VTY=VY-VN*NPY;  VTZ=VZ-VN*NPZ;
     X+=VTX*dt; Y+=VTY*dt; Z+=VTZ*dt;
     VX=VTX;VY=VTY;VZ=VTZ;   VX+=AX*dt; VY+=AY*dt; VZ+=AZ*dt;
  }
```

[1] Cette méthode a été élaborée et utilisée sur de nombreuses formes de surfaces par S. Djabouabdallah dans son mémoire de maîtrise « trajectoires de particules sur des surfaces ».

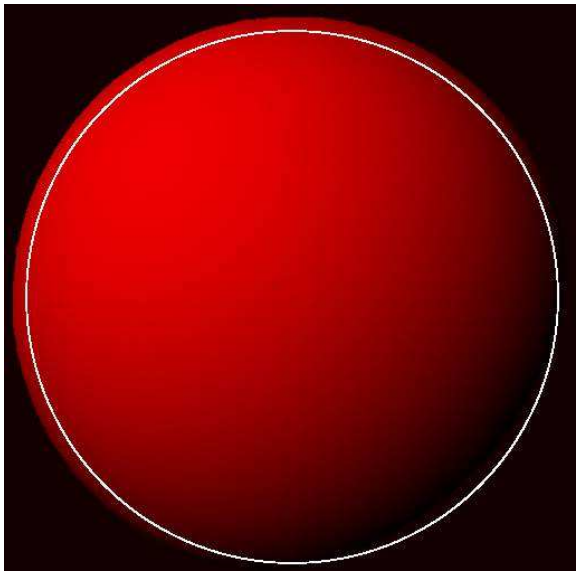
Vue de haut des trajectoires ($\alpha = \pi/2$) , avec comme point de départ de la particule :
$X = 0.95$, $Y = 0$, $Z = f(X,Y)$, et une certaine vitesse initiale :



VX=0 VY=0.305 VZ=0



VX=0 VY=0.703 VZ=0



VX=0 VY=1.7 VZ=0