Rotation en trois dimensions

Rappelons qu'une rotation de centre O (origine du repère) et d'angle φ dans le plan muni d'un repère orthonormé fait passer d'un point $M(x, y)$ à un point $M'(x', y')$ par la formule :

$\begin{cases} x' = x\cos\varphi - y\sin\varphi \\ y' = x\sin\varphi + y\cos\varphi \end{cases}$. La matrice correspondante est $\begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$.

Plaçons maintenant en trois dimensions. Commençons par faire une rotation d'axe Oz et d'angle φ. Dans un repère orthonormé $Oxyz$, cela revient à faire une rotation plane dans le plan xOy ou dans un plan parallèle, tout en laissant fixes les points de l'axe Oz. Cette rotation a pour matrice $R = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Mais qu'en est-il dans le cas général, où l'on fait une rotation d'angle φ autour d'un axe passant toujours par O mais de direction quelconque ? Un vecteur directeur de cet axe est (a, b, c), on le suppose de longueur 1 pour faciliter les calculs ($a^2 + b^2 + c^2 = 1$).

Nous allons voir que la matrice de cette rotation est :

$$M = (1-\cos\varphi)\begin{pmatrix} a^2 & ab & ac \\ ba & b^2 & bc \\ ca & cb & c^2 \end{pmatrix} + \cos\varphi\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \sin\varphi\begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}$$

Pour démontrer cette formule, nous allons passer du repère orthonormé $Oxyz$ originel à un nouveau repère orthonormé $OXYZ$ où l'axe OZ est l'axe de la rotation, de vecteur directeur unité de coordonnées (a, b, c) par rapport au repère initial. Un point M a pour coordonnées (x, y, z) dans le repère originel et pour coordonnées (X, Y, Z) dans le nouveau repère. La matrice de passage P donnant les anciennes coordonnées en fonction des nouvelles est une matrice orthogonale à cause des repères orthonormés. Son inverse n'est autre que sa transposée P^T, et l'on a la relation $R = P^T M P$ où M est la matrice de la rotation dans le repère initial et R celle de la rotation dans le nouveau repère. D'où $M = P R P^T$. On connaît R. Il reste à déterminer P.

Le plan perpendiculaire à l'axe de la rotation et passant par O a pour équation dans le repère originel : $ax + by + cz = 0$. Ce plan coupe le plan horizontal Oxy suivant une droite d'équations :

$\begin{cases} ax + by = 0 \\ z = 0 \end{cases}$

Cette droite qui appartient au plan perpendiculaire à OZ est elle-même perpendiculaire à OZ, et elle admet comme vecteur directeur unité $\boldsymbol{K}$ $(b, -a, 0)$, ou encore, en posant :

$A = \sqrt{a^2 + b^2} = \sqrt{1 - c^2}$, elle a comme vecteur directeur unitaire $\boldsymbol{I}(b/A, -a/A, 0)$ dans le repère originel. C'est ce vecteur $\boldsymbol{I}$ que nous prenons pour définir l'axe OX. Il reste à construire l'axe OY. Pour cela il nous faut un vecteur unité qui soit à la fois orthogonal à OX et OZ, et de façon à avoir une trièdre direct. Un tel vecteur est le produit vectoriel

K* . *I : il aura pour longueur 1, et le trièdre obtenu sera orthonormé direct. On obtient le vecteur ***J*** (ac/A, bc/A, $(c^2 - 1)/A$).

On sait que la matrice de passage P a pour vecteurs colonnes les nouveaux vecteurs de base avec leurs coordonnées dans le repère initial. Ainsi :

$$P = \begin{pmatrix} b/A & ac/A & a \\ -a/A & bc/A & b \\ 0 & (c^2-1)/A & c \end{pmatrix}$$

Il ne reste plus qu'à faire le produit $P\ R\ P^{\mathrm{T}}$ pour avoir M. Après un calcul un peu laborieux, on trouve la formule annoncée ci-dessus.

```
#include <SDL/SDL.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define R 1.
#define zoom 250.
#define N 100
#define NN 40
#define NNN (N*(NN+1))
#define deuxpi 6.283185
#define pi 3.14159
#define ll (deuxpi*5./180.)
#define L 200
#define xorig 400.
#define yorig 300.

float A,B,C,c, alpha,kk;
void pause(void);
void putpixel(int xe, int ye, Uint32 couleur); Uint32 getpixel(int xe, int ye) ;
void cercle(int xo, int yo, int RR, Uint32 couleur) ;
void disque( int xo, int yo, int RR, Uint32 couleur);
void ligne(int x0,int y0, int x1,int y1, Uint32 c);
void facettes(void);
void carre(int x1,int y1,int x2,int y2);
SDL_Surface * ecran;
Uint32 rouge,blanc,bleu, noir, gris, vert,coleur,pal[256],ncolor;
int xe[NNN],ye[NNN];
float dist[NNN],x[NNN],y[NNN],z[NNN],xx[NNN],yy[NNN],zz[NNN];

int main(int argc, char ** argv)
{
float lambda,phi, dl,dp,dangle;
int i, j,k,angle,a;

SDL_Init(SDL_INIT_VIDEO); srand(time(NULL));
 ecran=SDL_SetVideoMode(800,600,32, SDL_HWSURFACE|SDL_DOUBLEBUF);
```

```
    blanc=SDL_MapRGB(ecran->format,255,255,255);noir=SDL_MapRGB(ecran-
>format,0,0,0);
    vert=SDL_MapRGB(ecran->format,0,255,0);bleu=SDL_MapRGB(ecran->format,0,0,255);
    rouge=SDL_MapRGB(ecran->format,255,0,0);gris=SDL_MapRGB(ecran-
>format,100,100,100);
    for(i=0;i<256;i++) pal[i]=SDL_MapRGB(ecran->format,i,i,i);
    SDL_FillRect(ecran,0,blanc);

   alpha=pi/4. ;
   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];
        lambda+=dl;
        k++;
      }
     phi+=dp;
   }
   facettes();
   SDL_Flip(ecran);
   dangle=pi/360.;

   for(angle=0;angle<60;angle++)
    {   k=0;
       for(i=0; i<N; i++) for(j=0;j<=NN;j++)
         {  xx[k]=x[k];yy[k]=y[k]-dangle*z[k]; zz[k]=dangle*y[k]+z[k];k++;}
         k=0;
       for(i=0.; i<N; i++) for(j=0;j<=NN;j++)
         { x[k]=xx[k];y[k]=yy[k];z[k]=zz[k]; k++;}
         k=0;
       for(i=0.; i<N; i++) for(j=0;j<=NN;j++)
         {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]; k++;
         }
       facettes(); SDL_Flip(ecran);
       /*for(i=0;i<10000000;i++) a=1;*/  SDL_FillRect(ecran,0,blanc);
      }

     float aa=0.5, bb=0.866, cc=0.;
     /*float aa=0.3535, bb=0.3535, cc=0.866;*/
      for(angle=0;angle<720;angle++)
     {   k=0;  for(i=0; i<N; i++) for(j=0;j<=NN;j++)
           {  xx[k]=x[k]-cc*dangle*y[k]+bb*dangle*z[k];
             yy[k]=y[k]+cc*dangle*x[k]-aa*dangle*z[k];
             zz[k]=-bb*dangle*x[k]+aa*dangle*y[k]+z[k];   k++;
           }
           k=0;   for(i=0.; i<N; i++) for(j=0;j<=NN;j++)
                       { x[k]=xx[k];y[k]=yy[k];z[k]=zz[k]; k++;}
```

```
            k=0;   for(i=0.; i<N; i++) for(j=0;j<=NN;j++)
            {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]; k++;
            }
          facettes(); SDL_Flip(ecran);
          /*for(i=0;i<1;i++) a=1;*/  SDL_FillRect(ecran,0,blanc);
        }

    SDL_Flip(ecran);
    pause(); return 0;
    }

    void facettes(void)
    {  int k;
       for(k=0;k<NNN;k++) if ( (k+1) % (NN+1) !=0  && dist[k]<0.)
       {
        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);
       }
    }
```