NSY116 - Multimédia et interaction humain-machine - 2005-6

ED - l'image fixe - corrigé

Question 1 : Morphing

On peut faire une interpolation linéaire : à la K-ème (K = 0..N-1) étape, on a :

image(i,j) = (K*fin(i,j) + (N-1-K)*debut(i,j))/N

Résultats avec N=8

```
#include <stdio.h>
#define CMAX 1500
#define NIM 8

int imaD[CMAX][CMAX];
int imaF[CMAX][CMAX];
int L,W,NIVM,i,j,v,k,c;
char dummy[200], nf[50];
FILE *f;

main(){

 /*-- lecture de l'image depart*/
 /* on suppose un format PGM avec NIVM niveaux de gris */

 f = fopen("depart.pgm","r");
 fscanf(f,"%s",dummy);
 fscanf(f,"%d%d",&W,&L);
 if (W>CMAX){printf("image trop large pour moi\n");exit(0);}
 if (L>CMAX){printf("image trop haute pour moi\n");exit(0);}
 fscanf(f,"%d",&NIVM);
 for (i=0;i<L;i++){
  for (j=0;j<W;j++){
   fscanf(f,"%d ",&v);
   imaD[i][j]=v;
  }
 }
 fclose(f);
```

```
  /*-- lecture de l'image finale*/
 /* on suppose un format PGM avec NIVM niveaux de gris */

 f = fopen("finale.pgm","r");
 fscanf(f,"%s",dummy);
 fscanf(f,"%d%d",&W,&L);
 if (W>CMAX){printf("image trop large pour moi\n");exit(0);}
 if (L>CMAX){printf("image trop haute pour moi\n");exit(0);}
 fscanf(f,"%d",&NIVM);
 for (i=0;i<L;i++){
  for (j=0;j<W;j++){
   fscanf(f,"%d ",&v);
   imaF[i][j]=v;
  }
 }
 fclose(f);

 /*-- production des N images intermediaires */
 for (k=0;k<NIM;k++){
  sprintf(nf,"film%d.pgm",k);
  f = fopen(nf,"w");
  fprintf(f,"P2\n%d %d\n%d\n",W,L,NIVM);
  c = 0;
  for (i=0;i<L;i++){
   for (j=0;j<W;j++){
    v = (k*imaF[i][j]+(NIM-1-k)*imaD[i][j])/NIM;
    fprintf(f,"%d ",v);
    c++;if (c%20==0){fprintf(f,"\n");c=0;}
   }
  }
  fclose(f);
 }
}
```

Question 2 : Algorithme de Floyd & Steinberg (1975)

```
/* binarisation d'un fichier en niveaux de gris */
/* P. Cubaud decembre 2005  */

#include <stdio.h>
#define CMAX 1500
#define NIM 8

int ima[CMAX][CMAX];
int L,W,NIVM,i,j,lemax,lemin,seuil,erreur,v,c;
char dummy[200], nf[50];
FILE *f;

main(){

 /*-- lecture de l'image depart*/
 /* on suppose un format PGM avec NIVM niveaux de gris */

 f = fopen("lennaG.pgm","r");
 fscanf(f,"%s",dummy);
 fscanf(f,"%d%d",&W,&L);
 if (W>CMAX){printf("image trop large pour moi\n");exit(0);}
 if (L>CMAX){printf("image trop haute pour moi\n");exit(0);}
 fscanf(f,"%d",&NIVM);
 for (i=0;i<L;i++){
  for (j=0;j<W;j++){
   fscanf(f,"%d ",&v);
   ima[i][j]=v;
  }
 }
 fclose(f);

 /*-- recherche des niveaux min et max reels */
```

```
 lemin = NIVM;lemax = 0;
 for (i=0;i<L;i++){
  for (j=0;j<W;j++){
   if (ima[i][j] < lemin) lemin = ima[i][j] ;
   if (ima[i][j] > lemax) lemax = ima[i][j] ;
  }
 }
 seuil = (lemin + lemax)/2;
 printf("min : %d  max : %d seuil : %d \n",lemin, lemax, seuil);

 /*-- algo de diffusion de l'erreur de Floyd-Steinberg */
 for (i=0;i<L-1;i++){
  for (j=0;j<W-1;j++){
   if (ima[i][j] > seuil) { erreur =  - lemax + ima[i][j]; ima[i][j] = 0 ; /*
0=blanc pour un PBM */}
   else { erreur = ima[i][j] - lemin; ima[i][j] = 1 ; /* 1=noir pour un PBM */}
   ima[i+1][j] += 3*erreur/8;
   ima[i][j+1] += 3*erreur/8;
   ima[i+1][j+1] += erreur/4;
  }
 }

 /*-- traitement des bords du bas : on peut mieux faire */
 for (i=0;i<L;i++){
  ima[i][W-1] =  (ima[i][W-1] > seuil) ? 0 : 1 ;
 }
 for (j=0;j<W;j++){
  ima[L-1][j] =  (ima[L-1][j] > seuil) ? 0 : 1 ;
 }

 /*-- ecriture de l'image finale */
  f = fopen("lennaBW.pbm","w");
  fprintf(f,"P1\n%d %d\n",W,L);
  c = 0;
  for (i=0;i<L;i++){
   for (j=0;j<W;j++){
    fprintf(f,"%d ",ima[i][j]);
    c++;if (c%20==0){fprintf(f,"\n");c=0;}
   }
  }
  fclose(f);
}
```

Question 4 : Projection et caméra

• 1ère étape : conversion dans le repère de l'observateur
1 - Translation de l'origine O en O' => repère (X1,Y1,Z1)
2 - Rotation du repère (X1,Y1,Z1) de q-90° autour de Z1 => repère (X2,Y2,Z2)
3 - Rotation du repère (X2,Y2,Z2) de 90°+f autour de l'axe X2
4 - Conversion en un repère "main gauche"

$$\begin{cases} x' = -x.\sin\theta + y.\cos\theta \\ y' = -x.\cos\theta.\sin\phi - y.\sin\theta.\sin\phi + z.\cos\theta \\ z' = -x.\cos\theta.\cos\phi - y.\sin\theta.\cos\phi - z.\sin\phi + R \end{cases}$$

• 2ème étape : projection du système observé sur l'écran
Si projection perspective :

$$X = D\frac{x'}{z'} \qquad Y = D\frac{y'}{z'}$$

• 3ème étape : traiter les arêtes qui sortent de la zone de dessin (phase de "clipping"). pour l'éviter, on va rechercher les coord. X et Y extrèmes pour leur faire ensuite correspondre les extrémités de la zone de dessin.

• 4ème étape : conversion en coordonnées écran
- On choisit deux points de référence dans la scène, en général, une diagonale de la fenêtre de projection : (Xinf, Yinf) et (Xsup, Ysup)
- On veut que ces points soient situés respectivement en (Xa,Ya) et (Xb,Yb)
- Pour tous les autres points :

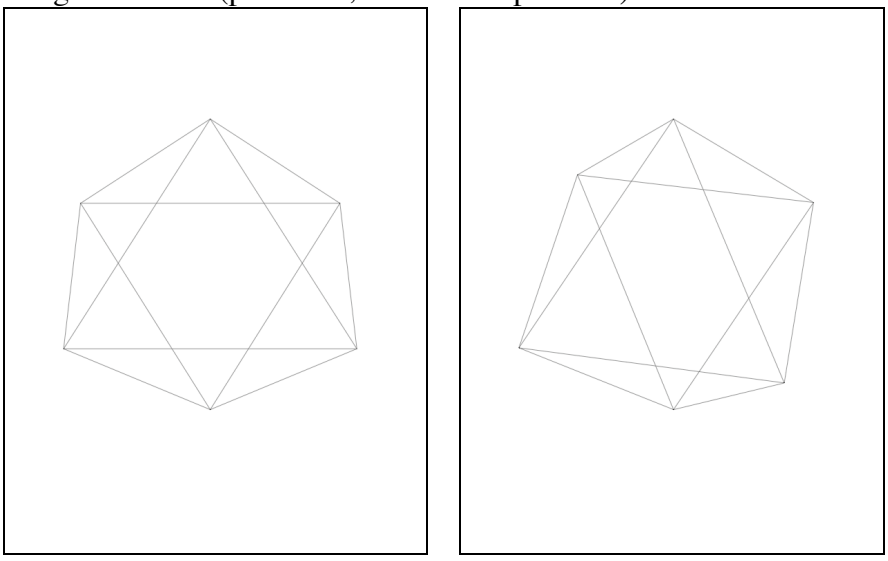
$$\begin{cases} X_{ecran} = \dfrac{X_b - X_a}{X_{\max} - X_{\min}}\left(X - X_{\min}\right) + X_a \\ Y_{ecran} = \dfrac{Y_b - Y_a}{Y_{\max} - Y_{\min}}\left(Y - Y_{\min}\right) + Y_a \end{cases}$$

Exemple de codage :

```
/******* Pierre Cubaud cubaud@cnam.fr dec. 2005 ******/

#include <stdio.h>
#include <math.h>

/* structure pour decrire la scene */
#define MAXFACES 500
double OB[MAXFACES][3][3]; /* coord. des sommets de l'objet etudie */
double O2[MAXFACES][3][2]; /* coord. projetees des sommets */
int NB;                    /* nombre reel de facettes dans l'objet */

/* position de l'observateur et distance de l'ecran */
#define R 10
#define theta 45
#define phi 30
#define D 5

/* zone de dessin de la scene en coord. ecran (cm) */
#define xa 3
#define ya 7.35
#define xb 18
#define yb 22.25

/* variables diverses */
int i,j;
double sthe,cthe,sphi,cphi,x,y,z,xx,yy,zz,minx,maxx,miny,maxy;
double px,py,pz,qx,qy,qz,nn,xo,yo,zo;

main()
{
 /*--- coord. cartesiennes de l'observateur ---*/
 sthe=sin(theta*3.1416/180);cthe=cos(theta*3.1416/180);
 sphi=sin(phi*3.1416/180);cphi=cos(phi*3.1416/180);
 xo= R*cthe*cphi;
 yo= R*sthe*cphi;
 zo= R*sphi;
 /*--- lecture des coordonnees ---*/
 NB=0;i=0;
 while (scanf("%lg %lg %lg\n",&OB[NB][i][0],&OB[NB][i][1],&OB[NB][i][2])!=EOF)
 {i++;
  if (i==3) {i=0;NB++;}
 }
 /*--- projection perspective ---*/
 for (i=0;i<NB;i++)
 {
  for (j=0;j<3;j++)
  {
```

```
   x=OB[i][j][0];y=OB[i][j][1];z=OB[i][j][2];
   xx= -x*sthe+y*cthe;
   yy= -x*cthe*sphi-y*sthe*sphi+z*cthe;
   zz= -x*cthe*cphi-y*sthe*cphi-z*sphi+R;
   O2[i][j][0]= D*xx/zz;
   O2[i][j][1]= D*yy/zz;
  }
 }
 /*--- pour eviter le clipping : calcul de la fenetre maximale ---*/
 minx=O2[0][0][0];maxx=minx;
 miny=O2[0][0][1];maxy=miny;
 for (i=0;i<NB;i++)
 {
  for (j=0;j<3;j++)
  {
   if (O2[i][j][0]<minx) {minx=O2[i][j][0];}
   if (O2[i][j][0]>maxx) {maxx=O2[i][j][0];}
   if (O2[i][j][1]<miny) {miny=O2[i][j][1];}
   if (O2[i][j][1]>maxy) {maxy=O2[i][j][1];}
  }
 }
 /*--- calcul des coordonnees ecran  ---*/
 for (i=0;i<NB;i++)
 {
  for (j=0;j<3;j++)
  {
   O2[i][j][0]=xa + (xb-xa)*(O2[i][j][0]-minx)/(maxx-minx);
   O2[i][j][1]=ya + (yb-ya)*(O2[i][j][1]-miny)/(maxy-miny);
  }
 }
 /*--- generation des ordres de trace en Postcript ---*/
 printf("/cm {72 mul 2.54 div} def \n");
 printf("1 cm 1 cm scale \n");
 printf("0.014 setlinewidth \n");
 printf("newpath \n");
 for (i=0;i<NB;i++)
 {
  printf("%.5f %.5f moveto \n",O2[i][0][0],O2[i][0][1]);
  printf("%.5f %.5f lineto \n",O2[i][1][0],O2[i][1][1]);
  printf("%.5f %.5f lineto \n",O2[i][2][0],O2[i][2][1]);
  printf("%.5f %.5f lineto \n",O2[i][0][0],O2[i][0][1]);
 }
 printf("stroke \n");
}
```

Image obtenue : (phi = 30°, theta = 45° puis 55°)



Exemple de fichier d'entrée :

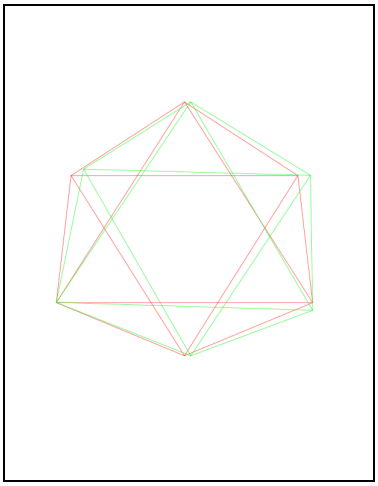
```
1.00000 0.00000 0.00000
0.00000 0.00000 1.00000
0.00000 1.00000 0.00000
0.00000 1.00000 0.00000
0.00000 0.00000 1.00000
-1.00000 0.00000 0.00000
-1.00000 0.00000 0.00000
0.00000 0.00000 1.00000
0.00000 -1.00000 0.00000
0.00000 -1.00000 0.00000
0.00000 0.00000 1.00000
1.00000 0.00000 0.00000
1.00000 0.00000 0.00000
0.00000 1.00000 0.00000
0.00000 0.00000 -1.00000
0.00000 1.00000 0.00000
-1.00000 0.00000 0.00000
0.00000 0.00000 -1.00000
-1.00000 0.00000 0.00000
0.00000 -1.00000 0.00000
0.00000 0.00000 -1.00000
0.00000 -1.00000 0.00000
1.00000 0.00000 0.00000
0.00000 0.00000 -1.00000
```

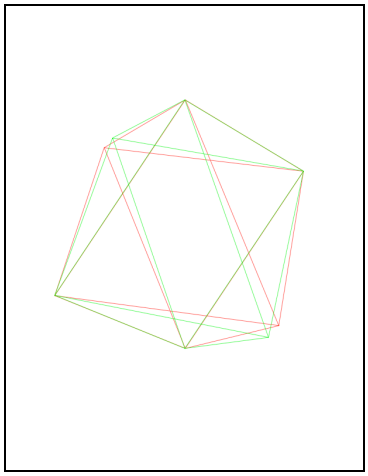
Résultat :

```
/cm {72 mul 2.54 div} def
1 cm 1 cm scale
0.014 setlinewidth
newpath
3.00000 10.46942 moveto
10.50001 22.25000 lineto
18.00000 10.46941 lineto
3.00000 10.46942 lineto
18.00000 10.46941 moveto
10.50001 22.25000 lineto
17.13447 17.92879 lineto
18.00000 10.46941 lineto
17.13447 17.92879 moveto
10.50001 22.25000 lineto
3.86558 17.92880 lineto
17.13447 17.92879 lineto
3.86558 17.92880 moveto
10.50001 22.25000 lineto
3.00000 10.46942 lineto
3.86558 17.92880 lineto
3.00000 10.46942 moveto
18.00000 10.46941 lineto
10.50001 7.35000 lineto
3.00000 10.46942 lineto
18.00000 10.46941 moveto
17.13447 17.92879 lineto
10.50001 7.35000 lineto
18.00000 10.46941 lineto
17.13447 17.92879 moveto
3.86558 17.92880 lineto
10.50001 7.35000 lineto
17.13447 17.92879 lineto
3.86558 17.92880 moveto
3.00000 10.46942 lineto
10.50001 7.35000 lineto
3.86558 17.92880 lineto
stroke
```

Question 5 : Anaglyphe

Le programme précédent est modifié avec deux calculs de vue en séquences : l'un avec la caméra en theta (et tracé en rouge) et l'autre avec theta+2° (et tracé en vert)





Autres exemples :