Courbe de Bézier en mouvement

Nous avons vu comment tracer la courbe de Bézier de quatre points dans le *chapitre IV* (*récursivité*) du cours d'*algorithmique*. Rappelons le programme de la fonction correspondante, où dans le cas présent nous avons pris les coordonnées des points en flottants pour atténuer les erreurs d'arrondi lors des divisions par deux :

```
void bezier(float xa,float ya,float xb,float yb,float xc,float yc, float xd, float yd, int n)
{    float xab,yab,xbc,ybc,xcd,ycd,xabc,yabc,xbcd,ybcd,xabcd,yabcd;
  if (n>0)
  {  xab=(xa+xb)/2; yab=(ya+yb)/2;  xbc=(xb+xc)/2; ybc=(yb+yc)/2;
     xcd=(xc+xd)/2; ycd=(yc+yd)/2;
     xabc=(xab+xbc)/2; yabc=(yab+ybc)/2;   xbcd=(xbc+xcd)/2; ybcd=(ybc+ycd)/2;
    xabcd=(xabc+xbcd)/2;yabcd=(yabc+ybcd)/2;
    bezier(xa,ya,xab,yab,xabc,yabc,xabcd,yabcd,n-1);
    bezier(xabcd,yabcd,xbcd,ybcd,xcd,ycd,xd,yd,n-1);
  }
  else if (n==0)
  {  ligne(xa,ya,xb,yb,vert); ligne(xb,yb,xc,yc,vert);   ligne(xc,yc,xd,yd,vert);  }
}
```

Dans le programme principal, il suffisait de se donner quatre points *A B C D* par leurs coordonnées sur l'écran, soit *xA*, *yA*, *xB*, *yB*, etc, puis d'appeler la fonction *bezier* () sur les huit coordonnées de ces points avec un indice *n* de récursivité égal à 4 ou 5.

Mais maintenant nous allons faire bouger les quatre points par le biais de la souris et suivre l'évolution en temps réel de la courbe de Bézier suivant le mouvement des points. A ce propos, à la fin du *chapitre I* (*initiation à C et SDL*) du cours d'*algorithmique*, nous avons vu comment faire bouger un objet sur l'écran de l'ordinateur en utilisant les mouvements de la souris. Cela s'appelle la gestion d'évènements. Ici, les évènements, ce sont avant tout les mouvements de la souris : le fait d'appuyer sur un bouton de la souris, ou de le relâcher. Ceux-ci vont se répercuter par des mouvements sur l'écran.

Commençons par dessiner les quatre points *A B C D* (ou 0 1 2 3) sous forme de carrés rouges de 20 sur 20, grâce aux surfaces *pointcarré*[] et aux couleurs *rouge*[], chaque point ayant sa propre nuance de rouge. Voici la première partie du programme principal :

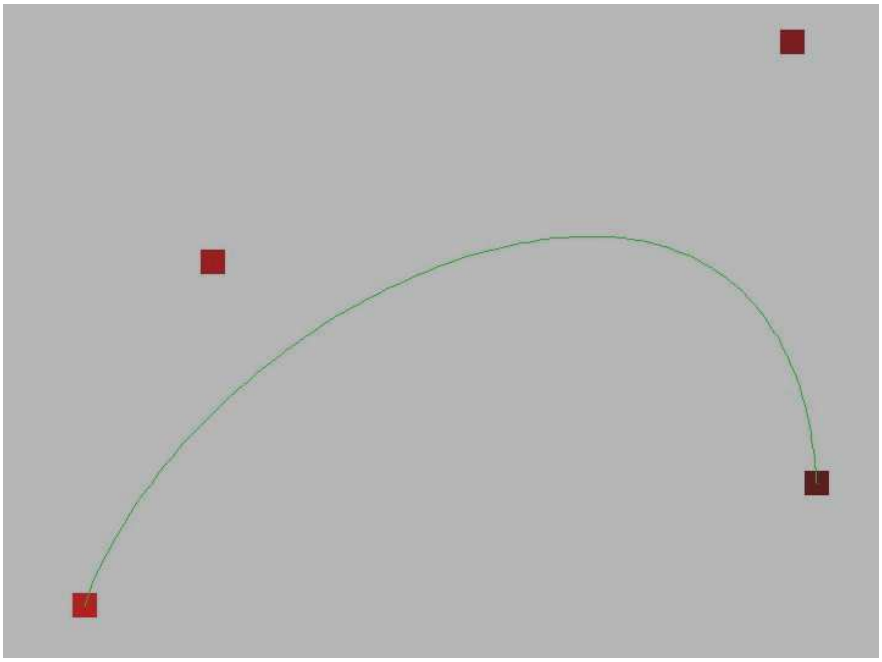
partie 1

```
#include <stdlib.h>
#include <SDL/SDL.h>
#define OUI 1
#define NON 0
void ligne(int x0,int y0, int x1,int y1, Uint32 c);
void cercle( int xo, int yo, int RR, Uint32 couleur);
Uint32 getpixel(int xe, int ye);
```

```
void bezier(float xa,float ya,float xb,float yb,float xc,float yc,float xd,float yd, int n);
Uint32 blanc,rouge[4],vert; SDL_Surface *ecran,*pointcarre[4];

int main ( int argc,char *argv[])
{  int flag, i, continuer;    SDL_Event event;   SDL_Rect position[4];
   SDL_Init( SDL_INIT_VIDEO );
   ecran=SDL_SetVideoMode(800, 600, 32,
                                  SDL_HWSURFACE | SDL_DOUBLEBUF);
   blanc=SDL_MapRGB(ecran->format,255,255,255);
   vert=SDL_MapRGB(ecran -> format, 0,255, 0);
  for(i=0;i<4;i++)
   {  pointcarre[i]= SDL_CreateRGBSurface(SDL_HWSURFACE,20,20,32,0,0,0,0);
     rouge[i]=SDL_MapRGB(ecran -> format, 255-50*i, 0, 0);
   }
   SDL_FillRect(ecran,NULL,blanc);
   for(i=0;i<4;i++) SDL_FillRect(pointcarre[i],NULL,rouge[i]);
   position[1].x = 30; position[1].y = 30;     /* on se donne la position des coins en haut à
   position[2].x = 500; position[2].y = 200;     gauche de chaque carré */
   position[0].x = 100; position[0].y = 500;
   position[3].x = 700; position[3].y = 400;
```

Si le programme s'arrêtait là, il suffirait de faire un *blit* des carrés sur l'écran, d'appeler la fonction *bérier*() et de faire un *flip* pour voir la courbe se dessiner. Mais nous allons faire beaucoup mieux.



Nous allons faire un programme « événementiel », constitué d'une grande boucle, ici *while (continuer==OUI)*, où le programme est dans l'attente du moindre évènement, par le biais de *SDL_WaitEvent(&event)*. Si nous ne faisons rien (aucun évènement ne se produit), le programme va tourner éternellement, et à chaque tour de la boucle, la fenêtre est mise en blanc, puis les quatre points sont affichés en rouge sur l'écran (*blit*),

et la courbe de Bézier dessinée [1], avec pour finir le *flip* qui permet ces dessins, Grâce au double buffer, il ne se produit aucun clignotement, et même si l'image ne cesse de se réafficher sur l'écran, on a l'impression d'une image fixe.

Maintenant ajoutons l'évènement habituel : *SDL_QUIT*. Si nous cliquons sur la croix en haut à droite de la fenêtre, la variable *continuer* passe à *NON*, la fenêtre se ferme et le programme s'arrête. Passons à la gestion de la souris : lorsqu'on la bouge, on voit une petite flèche se déplacer sur l'écran. Mais que veut-on exactement ? On veut que si l'on place la flèche de la souris dans un carré rouge (un des quatre points) et que l'on appuie sur un bouton de la souris, alors le point se trouve agrippé à la souris, et si l'on déplace la souris, le carré rouge va suivre le mouvement, celui-ci ne s'arrêtant que lorsqu'on relâche le bouton de la souris. Pour cela on utilise un drapeau *flag* qui vaut -1 lorsqu'aucun point n'est accroché à la souris, et qui vaut *i*, entre 0 et 3, lorsque la souris est accrochée au point *i*.

Voici ce qui correspond au moment où l'on appuie sur le bouton de la souris. Si jamais au même moment on se trouve à l'intérieur d'un des quatre carrés rouges, alors on force la souris à se placer au centre du carré, grâce à la fonction *SDL_WarpMouse*. Par la même occasion, *flag* prend le numéro du point concerné.

```
if (flag == -1 && event.type==SDL_MOUSEBUTTONDOWN )
  { for(i=0;i<4;i++)
    if ( getpixel(event.button.x,event.button.y)==rouge[i])
      { flag= I ;   SDL_WarpMouse(position[i].x+10,position[i].y+10); break; }
  }
```

A son tour, chaque mouvement élémentaire de la souris, ne serait-ce que d'un pixel, est un évènement, noté *SDL_MOUSEMOTION*, et la position correspondante de la souris est à chaque instant enregistrée dans les variables *event.button.x* et *event.button.y*. Si l'on veut que le carré rouge numéro *i* suive le mouvement de la souris (ce qui sous-entend que l'on continue d'appuyer sur le bouton de la souris), il suffit de faire :

```
if( flag !=-1 && event.type== SDL_MOUSEMOTION)
   {position[flag].x=event.button.x-10;   position[flag].y=event.button.y-10;}
```

Enfin, dernier évènement, relâchons le bouton de la souris : le point qui bougeait jusque là se fixe, même si l'on continue de faire bouger la souris. Il suffit de remettre *flag* à -1, soit :

```
if (flag !=-1 && event.type==SDL_MOUSEBUTTONUP)  flag=-1;
```

En mettant bout à bout tous ces évènements, on arrive à la deuxième et dernière partie du programme principal:

[1] Remarquons que l'on appelle la fonction *bezier* sur *position*[0].*x*+10, *position*[0].*y*+10, etc, car *position* correspond au sommet en haut à gauche du carré représentant le point, et que le point est pris au centre du carré. Il en sera de même lorsque l'on fixera la souris au centre du carré, comme on le verra ci-dessous.

partie 2

```
   flag=-1; continuer=OUI;
   while (continuer==OUI)
    { SDL_WaitEvent(&event);
      if (event.type==SDL_QUIT)     continuer=NON;
      else if (flag==-1 && event.type==SDL_MOUSEBUTTONDOWN )
       { for(i=0;i<4;i++)
         if ( getpixel(event.button.x,event.button.y)==rouge[i])
                  { flag=i;   SDL_WarpMouse(position[i].x+10,position[i].y+10); break; }
       }
      else if (flag!=-1 && event.type==SDL_MOUSEBUTTONUP)  flag=-1;
      else if( event.type== SDL_MOUSEMOTION)
           {position[flag].x=event.button.x-10;   position[flag].y=event.button.y-10;}

      SDL_FillRect(ecran,NULL,blanc);
      for(i=0;i<4;i++)  SDL_BlitSurface(pointcarre[i], 0, ecran, &position[i]);
      bezier(position[0].x+10,position[0].y+10,position[1].x+10, position[1].y+10,
              position[2].x+10,position[2].y+10,position[3].x+10,position[3].y+10, 5);
      SDL_Flip(ecran);
    }

for(i=0;i<4;i++) SDL_FreeSurface(pointcarre[i]);
return 0;
}
```

On peut ainsi déplacer chacun des quatre points définissant la courbe de Bézier, chacun à son tour, et l'on voit la courbe de Bézier suivre le mouvement des points. Il nous a suffi d'utiliser judicieusement les quatre évènements associés à la souris :

- *SDL_WarpMouse*
- *SDL_MOUSEBUTTONDOWN*
- *SDL_MOUSEBUTTONUP*
- *SDL_MOUSEMOTION*