Fonctions graphiques en C avec SDL
Structure générale d'un programme C avec SDL

Nous donnons ici les fonctions graphiques essentielles pour faire du dessin sous SDL. Leurs programmes sont donnés ci-dessous. Enfin nous donnons la structure d'un programme utilisant SDL. Il conviendra de faire des copier-coller pour vos propres programmes.

1. Fonctions

1.a. Ce à quoi elles servent

- *pause() : l'image est figée sur l'écran jusqu'à ce que l'on appuie sur une touché, ou sur la petite croix en haut à droite de la fenêtre.*
- *putpixel() : le pixel de coordonnées xe, ye est colorié sur l'écran avec la couleur c.*
- *getpixel() : ramène la couleur du pixel de coordonnées xe, ye.*
- *circle() : dessine un cercle de centre xo, yo, de rayon R, avec la couleur c.*
- *filldisc() : remplit le disque de centre xo, yo, de rayon R, avec la couleur c.*
- *line() : dessine une ligne entre les points (x0, y0) et (x1, y1) avec la couleur c. Deux fonctions sont disponibles. La première dessine la droite avec de petits traits horizontaux ou diagonaux. La deuxième dessine la droite avec des petits traits horizontaux et verticaux. Cette deuxième fonction doit être utilisée si l'on veut faire un floodfill dans une zone délimitée par des droites.*
- *rectangle() : dessine un rectangle dont le sommet en haut à gauche est (x1, y1) et le sommet en bas à droite (x2, y2), les côtés ayant la couleur c.*
- *arrow() : dessine un trait avec une flèche entre deux points (x1, y1, (x2, y2) avec la couleur c. Utilisée notamment pour les graphes, elle permet aussi de dessiner un petit cercle lorsque l'on va d'un point à lui-même.*
- *linewithwidth() : dessine une ligne entre deux points (x1, y1) et (x2, y2) avec la couleur c, et une certaine épaisseur width.*
- *floodfill() : remplit une surface délimitée par une courbe de bordure de couleur cb, à partir d'un point intérieur à la surface(x, y), avec une couleur de remplissage cr. Le point (x, y) joue le rôle de source et la couleur cr s'étale sur la surface par écoulement, jusqu'à la bordure de couleur cr. Si la bordure contient une ou des lignes, utiliser pour les tracer la fonction line() en marches d'escalier.*
- *arc…() : ces fonctions permettent de tracer des arcs de cercles (dans certains cas particuliers).*

Attention : ne pas utiliser les variables de ces fonctions (style x0, xo, etc.) comme variables globales.

1.b. Leurs programmes

```
void pause(void)
{
   SDL_Event evenement;
   do   SDL_WaitEvent(&evenement);
   while(evenement.type != SDL_QUIT && evenement.type != SDL_KEYDOWN);
}

void putpixel(int xe, int ye, Uint32 c)
{ Uint32 * numerocase;
numerocase= (Uint32 *)(screen->pixels)+xe+ye*screen->w;    *numerocase=c;
}

Uint32 getpixel(int xe, int ye)
{ Uint32 * numerocase;
  numerocase= (Uint32 *)(screen->pixels)+xe+ye*screen->w;   return (*numerocase);
}
```

```
void circle( int xo, int yo, int R, Uint32 c)
 {
    int x, y, F, F1, F2,newx,newy;
    x=xo; y=yo+R; F=0;
    if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
    if (x<800 && x>=0 && 2*yo-y>=0 && 2*yo-y<600) putpixel (x,2*yo-y, c);
    while( y>yo)
      {
        F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
        if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
        else {y-=1; F=F2;}
        if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
         newx=2*xo-x ; newy=2*yo-y ;
        if (x<800 && x>=0 && newy>=0 && newy<600) putpixel(x, newy,c);
        if (newx<800 && newx>=0 && y>=0 && y<600) putpixel( newx,y,c);
        if (newx<800 && newx>=0 && newy>=0 && newy<600) putpixel(newx,
        newy, c);
      }
     if (xo+R<800 && xo+R>=0) putpixel(xo+R,yo,c);
     if (xo-R<800 && xo-R>=0) putpixel(xo-R,yo, c);
 }

 void filldisc( int xo, int yo, int R, Uint32 c)
 {
    int x, y, F, F1, F2,newx,newy,xx;
    x=xo; y=yo+R; F=0;
    if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
    if (x<800 && x>=0 && 2*yo-y>=0 && 2*yo-y<600) putpixel (x,2*yo-y, c);
    while( y>yo)
      {
        F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
        if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
        else {y-=1; F=F2;}
        newx=2*xo-x ; newy=2*yo-y ;
        for(xx=newx; xx<=x; xx++)if (xx<800 && xx>=0 && y>=0 && y<600 )
        putpixel(xx,y,c);
        for(xx=newx; xx<=x; xx++)if (xx<800 && xx>=0 && newy>=0 && newy<600 )
        putpixel(xx,newy,c);
      }
     if (xo+R<800 && xo+R>=0&& y>=0 && y<600) putpixel(xo+R,yo,c);
     if (xo-R<800 && xo-R>=0&& y>=0 && y<600) putpixel(xo-R,yo, c);
 }

void line(int x0,int y0, int x1,int y1, Uint32 c)
{
  int dx,dy,x,y,residu,absdx,absdy,stepx,stepy,i;
  dx=x1-x0; dy=y1-y0;  residu=0;       x=x0;y=y0;  putpixel(x,y,c);
  if (dx>0) stepx=1;else stepx=-1; if (dy>0) stepy=1; else stepy=-1;
  absdx=abs(dx);absdy=abs(dy);
  if (dx==0) for(i=0;i<absdy;i++) { y+=stepy;   putpixel(x,y,c); }
  else if(dy==0) for(i=0;i<absdx;i++){ x+=stepx;   putpixel(x,y,c);  }
  else if (absdx==absdy)
  for(i=0;i<absdx;i++) {x+=stepx; y+=stepy;  putpixel(x,y,c); }
  else if (absdx>absdy)
  for(i=0;i<absdx;i++)
     { x+=stepx; residu+=absdy;
       if(residu >= absdx) {residu -=absdx; y+=stepy;}
       putpixel(x,y,c);
     }
  else for(i=0;i<absdy;i++)
```

```
        {y+=stepy; residu +=absdx;
         if (residu>=absdy)   {residu -= absdy;x +=stepx;}
         putpixel(x,y,c);
        }
}

/*  stair line, must be used for floodfill  (ligne en marches d'escalier) */
void line(int x0,int y0, int x1,int y1, Uint32 c)
{
int dx,dy,x,y,residu,absdx,absdy,pasx,pasy,i;
dx=x1-x0; dy=y1-y0;  residu=0;       x=x0;y=y0;  if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c);
if (dx>0) pasx=1;else pasx=-1; if (dy>0) pasy=1; else pasy=-1;
absdx=abs(dx);absdy=abs(dy);
if (dx==0) for(i=0;i<absdy;i++) { y+=pasy;
                                  if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c); }
else if(dy==0) for(i=0;i<absdx;i++){ x+=pasx;
                                     if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c);  }
else if (absdx==absdy)
  for(i=0;i<absdx;i++) {x+=pasx; if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c);
                        y+=pasy;
                        if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c);
                        }
else if (absdx>absdy)
       for(i=0;i<absdx;i++)
                { x+=pasx; if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c);
                  residu+=absdy;
                  if(residu >= absdx) {residu -=absdx; y+=pasy;
                                       if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c);
                                       }
                }
else for(i=0;i<absdy;i++)
          {y+=pasy; if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c);
           residu +=absdx;
           if (residu>=absdy)   {residu -= absdy;x +=pasx;
                                 if (x>=0 && x<800 && y>=0 && y<600) putpixel(x,y,c);
                                 }
          }
}

void rectangle(int x1,int y1, int x2, int y2, Uint32 c)
{
   line(x1,y1,x2,y1,c);line(x1,y2,x2,y2,c);line(x1,y1,x1,y2,c);line(x2,y2,x2,y1,c);
}

void arrow(int x1, int y1, int x2, int y2, Uint32 c)
{
int dx,dy;
float xf1,yf1,xf2,yf2,d,dx1,dy1,ndx1,ndy1,ndx2,ndy2,angle=M_PI/6.;
line(x1,y1,x2,y2,c);
dx=x2-x1; dy=y2-y1;       d=sqrt(dx*dx+dy*dy);
if (d!=0.)
{ dx1=6.*(float)dx/d; dy1=6.*(float)dy/d;
  ndx1=dx1*cos(angle)-dy1*sin(angle);
  ndy1=dx1*sin(angle)+dy1*cos(angle);
  xf1=0.3*x1+0.7*x2; yf1=0.3*y1+0.7*y2;     xf2=xf1-ndx1; yf2=yf1-ndy1;
  line(xf1,yf1,xf2,yf2,c);
  ndx2=dx1*cos(-angle)-dy1*sin(-angle);
  ndy2=dx1*sin(-angle)+dy1*cos(-angle);
  xf2=xf1-ndx2; yf2=yf1-ndy2;    line(xf1,yf1,xf2,yf2,c);
}
```

```
else
    {circle(x1+10,y1,10,c); line(x1+20,y1,x1+23,y1-6,c);
     line(x1+20,y1,x1+15,y1-5,c);
    }
}

/* add  #include <math.h> */
void linewithwidth(int x1, int y1, int x2, int y2, int width,Uint32 c)
{
int dx,dy;
float k,xf1,yf1,xf2,yf2,d,dx1,dy1,ndx1,ndy1,ndx2,ndy2,angle=M_PI/2.;
line(x1,y1,x2,y2,c);
dx=x2-x1; dy=y2-y1;       d=sqrt(dx*dx+dy*dy);
if (d!=0.)      /* si le vecteur n'est pas nul */
{ dx1=(float)width*(float)dx/d; dy1=(float)width*(float)dy/d;
  ndx1=dx1*cos(angle)-dy1*sin(angle);
  ndy1=dx1*sin(angle)+dy1*cos(angle);
  ndx2=dx1*cos(-angle)-dy1*sin(-angle);
  ndy2=dx1*sin(-angle)+dy1*cos(-angle);
  for(k=0;k<=1.;k+=0.1/d)
  {
  xf1=(1.-k)*x1+k*x2; yf1=(1.-k)*y1+k*y2;
  xf2=xf1-ndx1; yf2=yf1-ndy1;  line(xf1,yf1,xf2,yf2,c);
  xf2=xf1-ndx2; yf2=yf1-ndy2;    line(xf1,yf1,xf2,yf2,c);
  }
}
}

void floodfill( int x,int y, Uint32 cr,Uint32 cb)
{   int xg,xd,xx;
   if (getpixel(x,y) !=cb && getpixel(x,y) !=cr)
   { putpixel(x,y,cr);
     xg=x-1;
     while(xg>0 && getpixel(xg,y)!=cb)  {putpixel(xg,y,cr); xg--;}
     xd=x+1;
     while(xd<800 && getpixel(xd,y)!=cb)  {putpixel(xd,y,cr); xd++ ;}
     for(xx=xg; xx<xd;xx++)
       { if (y>1 ) {floodfill(xx,y-1,cr,cb);}
         if (y<599 ) {floodfill(xx,y+1,cr,cb);}
       }
   }
}

void arc0_90( int xo, int yo, int R, Uint32 c)
 {
    int x, y, F, F1, F2,newx,newy;
    x=xo; y=yo+R; F=0;
    if (x<800 && x>=0 && 2*yo-y>=0 && 2*yo-y<600) putpixel (x,2*yo-y, c);
    while( y>yo)
       {
          F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
          if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
          else {y-=1; F=F2;}
           newy=2*yo-y ;
          if (x<800 && x>=0 && newy>=0 && newy<600) putpixel(x, newy,c);
       }
     if (xo+R<800 && xo+R>=0) putpixel(xo+R,yo,c);
 }

 void arc270_360( int xo, int yo, int R, Uint32 c)
```

```
 {
    int x, y, F, F1, F2,newx,newy;
    x=xo; y=yo+R; F=0;
    if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
    while( y>yo)
      {
        F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
        if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
        else {y-=1; F=F2;}
        if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
      }
     if (xo+R<800 && xo+R>=0) putpixel(xo+R,yo,c);
 }

void arc90_180( int xo, int yo, int R, Uint32 c)
 {
    int x, y, F, F1, F2,newx,newy;
    x=xo; y=yo+R; F=0;
    if (x<800 && x>=0 && 2*yo-y>=0 && 2*yo-y<600) putpixel (x,2*yo-y, c);
    while( y>yo)
      {
        F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
        if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
        else {y-=1; F=F2;}
         newx=2*xo-x ; newy=2*yo-y ;
        if (newx<800 && newx>=0 && newy>=0 && newy<600) putpixel(newx,
        newy, c);
      }
     if (xo-R<800 && xo-R>=0) putpixel(xo-R,yo, c);
 }

void arc180_270( int xo, int yo, int R, Uint32 c)
 {
    int x, y, F, F1, F2,newx,newy;
    x=xo; y=yo+R; F=0;
    if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);

    while( y>yo)
      {
        F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
        if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
        else {y-=1; F=F2;}
         newx=2*xo-x ;
        if (newx<800 && newx>=0 && y>=0 && y<600) putpixel( newx,y,c);
      }
     if (xo-R<800 && xo-R>=0) putpixel(xo-R,yo, c);
 }

void arc0_180( int xo, int yo, int R, Uint32 c)
 {
    int x, y, F, F1, F2,newx,newy;
    x=xo; y=yo+R; F=0;

    if (x<800 && x>=0 && 2*yo-y>=0 && 2*yo-y<600) putpixel (x,2*yo-y, c);
    while( y>yo)
      {
        F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
        if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
        else {y-=1; F=F2;}
        newx=2*xo-x ; newy=2*yo-y ;
```

```
            if (x<800 && x>=0 && newy>=0 && newy<600) putpixel(x, newy,c);
            if (newx<800 && newx>=0 && newy>=0 && newy<600) putpixel(newx,
            newy, c);
          }
        if (xo+R<800 && xo+R>=0) putpixel(xo+R,yo,c);
        if (xo-R<800 && xo-R>=0) putpixel(xo-R,yo, c);
  }

  void arc180_360( int xo, int yo, int R, Uint32 c)
  {
     int x, y, F, F1, F2,newx,newy;
     x=xo; y=yo+R; F=0;
     if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
     while( y>yo)
       {
          F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
          if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
          else {y-=1; F=F2;}
          if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
           newx=2*xo-x ;
          if (newx<800 && newx>=0 && y>=0 && y<600) putpixel( newx,y,c);
       }
        if (xo-R<800 && xo-R>=0) putpixel(xo-R,yo, c);
  }

void arc270_90( int xo, int yo, int R, Uint32 c)
  {
     int x, y, F, F1, F2,newx,newy;
     x=xo; y=yo+R; F=0;
     if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
     if (x<800 && x>=0 && 2*yo-y>=0 && 2*yo-y<600) putpixel (x,2*yo-y, c);
     while( y>yo)
       {
          F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
          if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
          else {y-=1; F=F2;}
          if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
            newy=2*yo-y ;
          if (x<800 && x>=0 && newy>=0 && newy<600) putpixel(x, newy,c);
       }
        if (xo+R<800 && xo+R>=0) putpixel(xo+R,yo,c);
  }

void arc90_270( int xo, int yo, int R, Uint32 c)
  {
     int x, y, F, F1, F2,newx,newy;
     x=xo; y=yo+R; F=0;
     if (x<800 && x>=0 && y>=0 && y<600) putpixel(x,y,c);
     if (x<800 && x>=0 && 2*yo-y>=0 && 2*yo-y<600) putpixel (x,2*yo-y, c);
     while( y>yo)
       {
          F1=F+2*(x-xo)+1; F2=F-2*(y-yo)+1;
          if ( abs(F1)<abs(F2))  { x+=1; F=F1;}
          else {y-=1; F=F2;}
           newx=2*xo-x ; newy=2*yo-y ;

          if (newx<800 && newx>=0 && y>=0 && y<600) putpixel( newx,y,c);
          if (newx<800 && newx>=0 && newy>=0 && newy<600) putpixel(newx,
          newy, c);
       }
```

```
    if (xo-R<800 && xo-R>=0) putpixel(xo-R,yo, c);
 }
```

Remarque

Pour toutes les fonctions précédentes, la fenêtre écran (*screen*) est supposée de dimension 800 sur 600 pixels, comme cela est fait dans la déclaration :

screen=SDL_SetVideoMode(800,600,32, SDL_HWSURFACE|SDL_DOUBLEBUF);

Nous avons aussi choisi le double buffer *SDL_DOUBLEBUF*, afin de rendre fluide les animations d'images.

La fonction *putpixel*() n'étant pas sécurisée, il convient de maintenir le point concerné à l'intérieur strict de la fenêtre écran, sinon on aura une erreur fatale. Par contre les fonctions traçant des lignes ou des cercles sont sécurisées.

2. Structure générale d'un programme sous C et SDL

```
#include <SDL/SDL.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define … éventuellement

void pause(void);
void putpixel(int xe, int ye, Uint32 c);
Uint32 getpixel(int xe, int ye);

et declarations d'autres fonctions, éventuellement

SDL_Surface * screen;
Uint32 white, black, couleur[100];   déclarations des couleurs

déclaration de variables globales, éventuellement

int main(int argc, char ** argv)
{
   déclaration des variables locales

   SDL_Init(SDL_INIT_VIDEO);
   screen=SDL_SetVideoMode(800,600,32, SDL_HWSURFACE|SDL_DOUBLEBUF);

   /* exemples de couleurs avec leurs trois composantes RGB */
   couleur[0]=SDL_MapRGB(screen->format,255,255,255);   /* white */
   couleur[1]=SDL_MapRGB(screen->format,255,0,0);    /* red */
   couleur[2]=SDL_MapRGB(screen->format,0,250,0);   /* green */
   SDL_FillRect(screen,0,couleur[0]);  /* donne un fond blanc à la fenêtre */

   ….. mettre ici le programme concerné

  SDL_Flip(screen);  /* Cette fonction affiche l'image issue du programme sur l'écran. Si on ne le fait pas,
                        on ne verra rien. On sera souvent amené à l'utiliser plusieurs fois dans le programme pour
                        voir  ce qui se passe  */
  pause();return 0;
}

void pause(void)
{
   SDL_Event evenement;
```

```
   do   SDL_WaitEvent(&evenement);
   while(evenement.type != SDL_QUIT && evenement.type != SDL_KEYDOWN);
}

void putpixel(int xe, int ye, Uint32 c)
{ Uint32 * numerocase;
numerocase= (Uint32 *)(screen->pixels)+xe+ye*screen->w;    *numerocase=c;
}

Uint32 getpixel(int xe, int ye)
{ Uint32 * numerocase;
  numerocase= (Uint32 *)(screen->pixels)+xe+ye*screen->w;   return (*numerocase);
}
```

Pour plus de détails, on pourra se reporter aux documents :

- *Petite initiation au langage C et au graphisme SDL*
- *Graphisme SDL : points, droites, cercles*

dans le *cours d'algorithmique*, rubrique *enseignements*,

- ainsi qu'au cours *Graphisme et géométrie*, rubrique *enseignements*.