```c
/**
 * Bibliothèque de gestion d'un graphe:
 * Cette bibliothèque permet de dessiner des cercles, des lignes et des croix.
 * Elle a été créer afin de manipuler des graphes simplements.
 * */

#include <time.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <unistd.h>
#include <inttypes.h>

#include "flame.h"
#include "moteur_graphique.h"

#include "pi.h"

#define A_STEP 0.01


void remplir_cercle(flame_obj_t *fo, cercle_t *b)
{
    flame_set_color(fo, b->r, b->g,b->b);
    for (double angle = 0.0; angle < 2 * PI; angle += A_STEP)
    {
        flame_draw_line(fo, b->pos_x, b->pos_y, b->pos_x + b->rad * cos(angle),
        b->pos_y + b->rad * sin(angle));
    }
}

void afficher_cercle(flame_obj_t *fo, cercle_t *b)
{
    flame_set_color(fo, b->r, b->g,b->b);
    for (double angle = 0.0; angle < 2 * PI; angle += A_STEP)
    {
        flame_draw_point(fo, b->pos_x + b->rad * cos(angle), b->pos_y + b->rad *
        sin(angle));
    }
}

void afficher_ligne(flame_obj_t *fo,int x1,int y1,int x2,int y2)
{
    flame_draw_line(fo, x1, y1,x2,y2);
}

char recupere_clavier(XEvent event)
{
    char c;
    if (event.type == KeyPress)
    {
        c = XLookupKeysym(&event.xkey, 0);
        return c;
    }
    return -1;
```

```
 56  }
 57
 58  flame_obj_t * init_canvas()
 59  {
 60      return flame_open("Graphe",TAILLE_ECRAN_HAUTEUR,TAILLE_ECRAN_LARGEUR);
 61  }
 62
 63  void colorer_cercle(cercle_t * c,enum couleur coul)
 64  {
 65      switch(coul)
 66      {
 67          case ROUGE :
 68          {
 69              c->r = 255;
 70              c->g = 0;
 71              c->b = 0;
 72              break;
 73          }
 74          case BLEU :
 75          {
 76              c->r = 0;
 77              c->g = 0;
 78              c->b = 255;
 79              break;
 80          }
 81          case VERT :
 82          {
 83              c->r = 0;
 84              c->g = 255;
 85              c->b = 0;
 86              break;
 87          }
 88          case BLANC :
 89          {
 90              c->r = 255;
 91              c->g = 255;
 92              c->b = 255;
 93              break;
 94          }
 95          case JAUNE :
 96          {
 97              c->r = 255;
 98              c->g = 255;
 99              c->b = 0;
100              break;
101          }
102          default:
103          {
104              perror("switch");
105              exit(EXIT_FAILURE);
106          }
107      }
108  }
109
110  void afficher_connexion(flame_obj_t *fo, cercle_t * c, int id_1, int id_2, enum
     couleur coul)
111  {
```

```
112       if(coul == BLANC) flame_set_color(fo, 255, 255, 255);
113       if(coul == ROUGE) flame_set_color(fo, 255, 0, 0);
114       if(coul == GRIS) flame_set_color(fo, 200, 200, 200);
115       if(coul == NOIR) flame_set_color(fo, 0, 0, 0);
116       if(coul == JAUNE) flame_set_color(fo, 255, 255, 0);
117       afficher_ligne(fo, c[id_1].pos_x, c[id_1].pos_y, c[id_2].pos_x, c[id_2].pos_y);
118   }
119
120   void affiche_croix(flame_obj_t *fo, int x, int y, enum couleur coul) {
121       if(coul == BLANC) flame_set_color(fo, 255, 255, 255);
122       if(coul == NOIR) flame_set_color(fo, 0, 0, 0);
123
124       afficher_ligne(fo, x - 10, y - 10, x + 10, y + 10);
125       afficher_ligne(fo, x - 10, y + 10, x + 10, y - 10);
126   }
127
```