```c
#include <stdio.h>

//~ local
#include "const.h"
#include "graph.h"
#include "moteur_graphique.h"
#include "routage.h"

void afficher_voisin(graphe* G, flame_obj_t * fo, cercle_t * c,int sommet, int deb, int fin)
{
    int j;
    for(j = deb; j < fin; j++) if(G->list[sommet][j] != -1) { afficher_connexion(fo,c,sommet,j,GRIS); }
}


void init_affichage_tier(cercle_t * c, int debut,int fin, int tier,int * x,int * y)
{
    int i;

    for( i = debut ; i < fin ; i ++)
    {
        c[i].rad = TAILLE_CERCLE;
        if(tier == tier1)
        {
            colorer_cercle(&c[i],ROUGE);
            c[i].pos_x = *x * ( 3 * TAILLE_CERCLE);
            c[i].pos_y = *y * ( 3 * TAILLE_CERCLE);

        }
        if(tier == tier2)
        {
            colorer_cercle(&c[i],BLEU);
            c[i].pos_x = *x * ( 3 * TAILLE_CERCLE);
            c[i].pos_y = *y * ( 3 * TAILLE_CERCLE);
        }
        if(tier == tier3)
        {
            colorer_cercle(&c[i],VERT);
            c[i].pos_x = *x * ( 3 * TAILLE_CERCLE);
            c[i].pos_y = *y * ( 3 * TAILLE_CERCLE);
        }
        *x+=1;
        if(*x > 10)
        {
            *y += 1;
            *x = 1;
        }
    }
}

void initialisation_objets_graphique(graphe *G,flame_obj_t * fo,cercle_t * c)
{
    int noeud;

    int x = 1;
```

```
56      int y = 1;
57
58      // > Initialise les coordonnées des cercles
59      init_affichage_tier(c,debTier1,finTier1,tier1,&x,&y);
60      init_affichage_tier(c,debTier2,finTier2,tier2,&x,&y);
61      init_affichage_tier(c,debTier3,finTier3,tier3,&x,&y);
62
63      for (noeud = 0; noeud < TAILLE_GRAPHE; noeud++)
64      {
65          afficher_cercle(fo, &c[noeud]);
66          // > Affichage des connexions mais c'est illisble
67          //~ afficher_voisin(G, fo, c, noeud, debTier1, finTier1);
68          //~ afficher_voisin(G, fo, c, noeud, debTier2, finTier2);
69          //~ afficher_voisin(G, fo, c, noeud, debTier3, finTier3);
70      }
71  }
72
73  int trouve_id(int x,int y)
74  {
75      x -= (2 * TAILLE_CERCLE);
76      y -= (2 * TAILLE_CERCLE);
77      return ((y / (3 * TAILLE_CERCLE))*10) + (x / (3*TAILLE_CERCLE));
78  }
79
80  void affiche_chemin (flame_obj_t * fo,routage* R, cercle_t * c,int deb, int
    fin,enum couleur coul)
81  {
82      int voisin[TAILLE_GRAPHE] = {-1};
83
84      int i = 0, suiv = R->succ[deb][fin];
85      voisin[i] = deb;
86
87      for(i = 1; suiv != fin && i < TAILLE_GRAPHE; i++)
88      {
89          deb = R->succ[deb][fin];
90          voisin[i] = deb;
91          suiv = R->succ[deb][fin];
92      }
93
94      if(suiv == fin)
95      {
96          voisin[i] = fin;
97
98          int d, e, j;
99          for(j = 0 ; j < i; j++)
100         {
101             d = voisin[j]; e = voisin[j+1];
102             afficher_connexion(fo, c, d, e, coul);
103         }
104     }
105     else
106     {
107         printf("error\n");
108     }
109 }
110
111
```

```
112  void interaction_user(graphe * G,flame_obj_t * fo,routage * R,cercle_t * c)
113  {
114      XEvent event;
115      int cmp = 0;
116      int id_1 = 0;
117      int id_2 = 0;
118      int save_id_1 = 0;
119      int save_id_2 = 0;
120
121      int click_x, click_y;
122
123      while (1)
124      {
125          if (XPending(fo->display) > 0)
126          {
127            XNextEvent(fo->display, &event);
128            if(recupere_clavier(event) == 'q') { break; }
129            if (event.type == ButtonPress)
130            {
131                  // Récupere les coordonées
132                  click_x = event.xkey.x;
133                  click_y = event.xkey.y;
134
135                  // colorie et affiche les cercles
136                  colorer_cercle(&c[trouve_id(click_x,click_y)],JAUNE);
137                  afficher_cercle(fo,&c[trouve_id(click_x,click_y)]);
138
139                  if(cmp == 0)
140                  {
141                      save_id_1 = id_1;
142                      save_id_2 = id_2;
143                      id_1 = trouve_id(click_x,click_y);
144
145                      // Permet d'éffacer les traits
146                      affiche_chemin ( fo, R, c, save_id_1, save_id_2, NOIR);
147                      affiche_croix(fo, c[save_id_2].pos_x, c[save_id_2].pos_y, NOIR);
148                      initialisation_objets_graphique ( G, fo, c);
149
150                      // Coloris les cercles
151                      colorer_cercle( &c[id_1], JAUNE);
152
153                      // Affiche les cercles
154                      afficher_cercle(fo, &c[id_1]);
155                  }
156                  else
157                  {
158                      id_2 = trouve_id(click_x,click_y);
159
160                      // Coloris les cercles
161                      colorer_cercle( &c[id_2], JAUNE);
162
163                      // Affiche les cercles
164                      afficher_cercle(fo, &c[id_2]);
165                  }
166                  cmp ++;
167            }
168        }
```

```
169         else
170         {
171           if(cmp == 2)
172           {
173               cmp = 0;
174
175               // Affiche les chemins
176               afficher_chemin (R, id_1, id_2);
177               affiche_chemin (fo, R, c, id_1, id_2, JAUNE);
178
179               // Affiche debut
180
181               // Affiche arrivé
182               affiche_croix(fo, c[id_2].pos_x, c[id_2].pos_y, BLANC);
183           }
184         }
185     }
186   }
187
188   void gestion_fenetre_graphique(graphe* G, routage *R)
189   {
190       // > Initialisation du canvas:
191       flame_obj_t * fo = init_canvas();
192
193       // > Allocation de mémoire de la structure de donnée d'un cercle
194       cercle_t c[TAILLE_GRAPHE];
195
196       // > Initialise les connexions et les cercles
197       initialisation_objets_graphique(G,fo,c);
198
199       // > Gestion des interaction utilisateur machine
200       interaction_user(G,fo,R,c);
201
202       // > Fermeture du canvas
203       flame_close(fo);
204   }
205
```