Programmation en GTK (partie 1)

Tristan Colombo 2002

1. Aperçu rapide

Le GTK est un langage qui fonctionne grâce à la création d'objets graphique qui sont tous au départ du même type : les *GtkWidget* (ou Gadget GTK en bon français). Un objet créé peut contenir un autre objet ; par exemple :

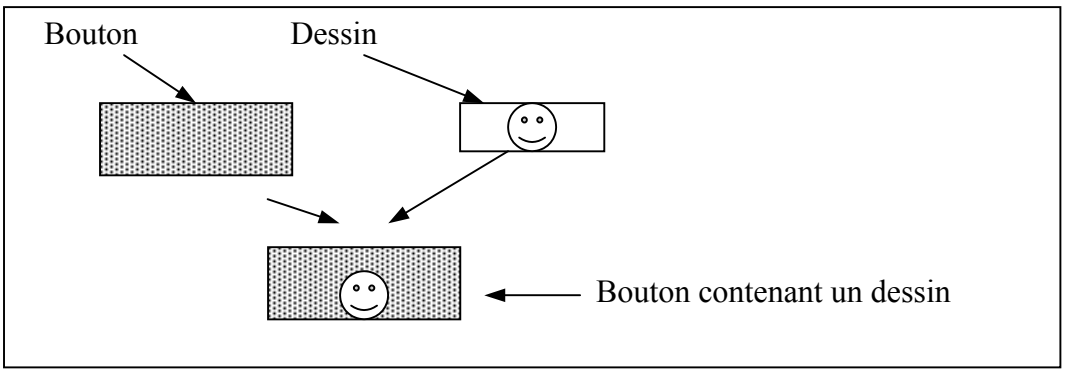

FIG. 1 – Les Gtk_Container

Lorsque l'objet Bouton reçoit l'objet Dessin il devient un container : la commande sera du type *gtk_container_add*(Bouton, Dessin) et Bouton sera alors un bouton contenant un dessin. Chaque objet peut recevoir un signal (la souris est sur l'objet, l'objet est affiché, l'utilisateur clique sur l'objet, etc.). On peut effectuer une opération particulière suivant le signal émis par *gtk_signal_connect*.

2. Ouverture d'une session GTK

Voici un programme très simple (et commenté !) montrant comment créer une fenêtre graphique en GTK (ce programme ne fait que créer la fenêtre et n'affiche rien dedans par contre on peut fermer cette fenêtre en cliquant sur le X en haut à droite) :

```
/* Le fichier d'include des fonctions GTK */
#include <gtk/gtk.h>

int main(int argc, char *argv[])
{
  /* Comme je l'ai fait remarqué précédemment, tous les objets GTK sont */
  /* des GtkWidget. Fenetre représentera notre objet de fenêtre */
  /* principale (celle qui contiendra tous les autres objets) */
  GtkWidget *Fenetre;
  /* Initialisation des bibliothèques de GTK */
  gtk_init(&argc, &argv);
  /* Création de la fenêtre */
  Fenetre = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  /* Connection des signaux « destroy » et « delete_event » : si la */
  /* fenêtre est détruite (destroy) alors on quitte la boucle */
  /* d'attente du GTK (gtk_main_quit) ; si l'on désire supprimer */
  /* la fenêtre (delete_event) alors on appelle la fonction */
  /* gtk_widget_destroy qui, comme son nom l'indique va détruire */
  /* le GtkWidget */
  gtk_signal_connect(GTK_OBJECT(Fenetre), "destroy", (GtkSignalFunc)
gtk_main_quit, &Fenetre);
```

```
  gtk_signal_connect(GTK_OBJECT(Fenetre), "delete_event", (GtkSignalFunc)
gtk_widget_destroy, NULL);
  /* gtk_window_set_title donne un nom à la fenêtre (la barre de */
  /* couleur en haut de la fenêtre */
  gtk_window_set_title(GTK_WINDOW(Fenetre), "Jeu de dames v0.0.1");
  /* Tout ce que l'on fait en GTK est stocké en mémoire pour un */
  /* affichage plus rapide : il faut donc maintenant afficher tout */
  /* ce que nous avons fait */
  gtk_widget_show_all(Fenetre);
  /* La boucle principale du GTK qui ne s'achève que lorsque le dernier */
  /* GtkWidget est détruit */
  gtk_main();
  return 0;
}
```

3. Affichage d'un pixmap

Pour afficher un pixmap (voir la fiche sur les fichiers xpm), nous disposons de quelques functions (je ne rentrerai pas dans les détails mais sachez que le GTK est en fait composé de 2 couches : une couche graphique appelée GDK et une couche de traitement englobant le GDK et appelée GTK) :

```
#include « mon_image.xpm »

GdkPixmap *Pix;
GdkBitmap *Masque;
GdkColor Transparent;
GtkWidget *Pixmap;

/* Instruction obligatoire pour afficher des pixmaps */
gtk_widget_realize(Fenetre);
/* Création du pixmap GDK */
Pix = gdk_pixmap_create_from_xpm_d(Fenetre->window,
                                   &Masque,
                                   &Transparent,
                                   mon_image_xpm);
/* Conversion en pixmap GTK */
Pixmap = gtk_pixmap_new(Pix, Masque);
/* On a plus besoin du pixmap GDK, ni de son masque donc on peut */
/* récupérer la place qu'ils occupaient en mémoire */
gdk_pixmap_unref(Pix);
gdk_pixmap_unref(Masque);
/* On place le pixmap dans la fenêtre (qui devient un container) */
gtk_container_add(GTK_CONTAINER(Fenetre), Pixmap);
```

Note : Pour fonctionner cet exemple doit bien sûr être intégré à la partie de code créant la fenêtre (voir section 2. Ouverture d'une session GTK).

4. Les gtk_tables

Pour pouvoir disposer aisément les objets dans la fenêtre principale, le plus simple est d'utiliser une gtk_table. Il s'agit d'un tableau permettant de placer un objet dans une ou plusieurs cases.

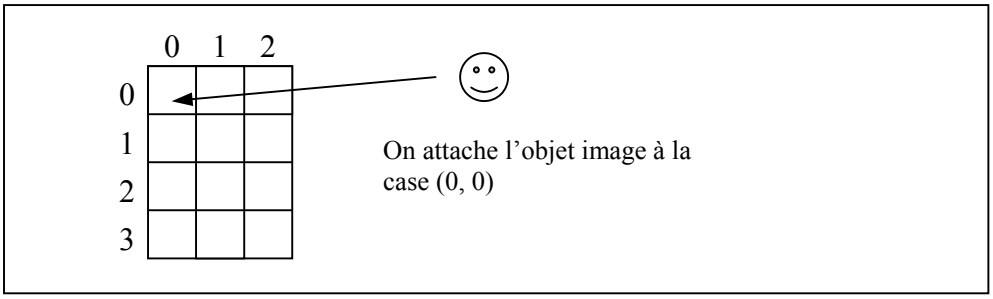

FIG. 2 – gtk_table

Voici un exemple de création d'une gtk_table et d'attachement du pixmap que nous avons créé tout à l'heure :

```
/* Déclaration du GtkWidget qui deviendra la table */
GtkWidget *Table;

/* Création d'une table de 11 x 11 cases */
Table=gtk_table_new(11, 11, TRUE);

/* Ajout de la table dans la fenêtre principale */
gtk_container_add(GTK_CONTAINER(Fenetre), Table);

/* Attachement d'un objet (ici le pixmap défini en section 3. */
/* Affichage d'un pixmap) à la case définie par les points (0, 0) et */
/* (1, 1) soit la première case en haut à gauche. Attention !!! les */
/* coordonnées des points A1(x1, y1) et A2(x2, y2) sont données dans */
/* l'ordre x1, x2, y1, y2 */
gtk_table_attach(GTK_TABLE(Table), Pixmap, 0, 1, 0, 1, GTK_FILL, GTK_FILL,
0, 0);
```

5. Les boutons

Un objet très utile en GTK est le bouton : on peut choisir le style d'affichage de ce dernier, y incorporer du texte ou une image (comme vu en section *1. Aperçu rapide*), et il s'agit d'un objet « clickable » que l'utilisateur peut actionner. Cette fois-ci, dans l'exemple nous allons attaché un bouton contenant un dessin à une table (ne pas conserver la partie de code de la section *4. Les gtk_tables* et insérer celui qui suit à la place).

```
/* Déclaration du GtkWidget qui deviendra la table */
GtkWidget *Table;
/* Déclaration du GtkWidget qui deviendra le bouton */
GtkWidget *Bouton;

/* Création d'une table de 11 x 11 cases */
Table=gtk_table_new(11, 11, TRUE);

/* Création du bouton */
Bouton = gtk_button_new();

/* Définition du style de relief du bouton */
gtk_button_set_relief(GTK_BUTTON(Bouton), GTK_RELIEF_NONE);
```

```
/* Bouton va devenir un container et contenir l'image Pixmap */
gtk_container_add(GTK_CONTAINER(Bouton), Pixmap);

/* On attache le bouton qui contient le dessin à la table (première case */
/* en haut à gauche) */
gtk_table_attach(GTK_TABLE(Table), Bouton, 0, 1, 0, 1, GTK_FILL, GTK_FILL,
0, 0);

/* Ajout de la table dans la fenêtre principale */
gtk_container_add(GTK_CONTAINER(Fenetre), Table);
```

Comme vous pouvez le voir il manque un petit quelque chose … lorsque l'on clique sur le bouton rien ne se passe ! Il faut connecter le signal « clicked » du bouton à une fonction … c'est ce que nous verrons en section *7. Les signaux*.

6. Affichage simple de texte

Pour afficher un petit texte : les labels. On crée un objet label dans lequel on place un texte. Dans l'exemple suivant on place un tel objet dans la case de droite du bouton.

```
/* Création du GtkWidget qui deviendra un Label */
GtkWidget *Label;

/* Définition du label avec son message */
Label=gtk_label_new("Test");

/* Attachement du label à la table en case (1, 0) – (2, 1) */
gtk_table_attach(GTK_TABLE(Table), Label, 1, 2, 0, 1, GTK_FILL, GTK_FILL,
0, 0);
```

7. Les signaux

Revenons à notre bouton … nous aimerions que le fait de cliquer dessus déclenche une action. Il faut alors le relier à un signal et à une fonction qui s'exécutera à la réception de ce signal.

```
/* Définition de la variable string à passer en paramètre */
char s[10];
sprintf(s, « coucou »);

/* On relit le signal « clicked » de l'objet Bouton à la fonction */
/* Affichage_coucou qui prend en paramètre une chaîne de caractères */
/* (impossible de transmettre plus d'une variable) */
gtk_signal_connect(GTK_OBJECT(Bouton), "clicked", (GtkSignalFunc)
Affichage_coucou, (gpointer) s);
```

« Mais … et la fonction Affichage_coucou … comment la définit-on ? » me direz-vous. Voici un donc un exemple de fonction de rappel (ça s'appelle comme ça …).

```
/* Une fonction de rappel prend toujours en premier argument un */
/* GtkObject *objet qui est l'objet qui a appelé cette fonction */
/* et éventuellement un paramètre de type gpointer (qui peut être */
/* de n'importe quel type donc il faut penser à caster (donner le */
/* type de la variable avant son nom comme dans (char *) s, ou (int) i) */
void Affichage_coucou(GtkObject *objet, gpointer s)
{
  fprintf(« %s\n », (char *) s) ;
}
```

Note : pensez ici à ajouter `#include <stdio.h>`.

8. Dernières remarques

Pensez à :

- bien indenter votre travail.
- commenter toutes les fonctions et éventuellement indiquer à quoi correspondent les variables (pas les variables de boucles bien sûr …).
- noter au fur et à mesure ce que vous faites (ça ira plus vite pour faire le rapport !!!)
- tester votre code au fur et à mesure (vous pouvez utiliser `fprintf(stderr, « la variable i vaut %d\n », i);` pour déboguer) ; sachez qu'il est beaucoup plus facile de trouver une erreur dans 10 lignes de code que dans 100 (en supposant qu'il n'y ait qu'une seule erreur dans les 100 lignes de code …).

May the force be with you
(parce-qu'il est pas vraiment évident votre sujet …)

9. Références

[1] ODIN (D.), *Programmation Linux avec GTK+*, Eyrolles, 2000.
[2] Le site officiel de GTK (avec tutorial, **A**pplication **P**rogrammer **I**nterface , …) : *http://www.gtk.org*