GTK+ une alternative à Qt

Daniel Antunes

5 novembre 2014

Table des matières

Introduction à GTK+

- GTK+ : **G**IMP **T**oolk**i**t
- librairie pour création d'interface graphique
- écrit en C
- compatible avec : C/C++, Java, Python, Perl, Ada, Fortan,...
- utilisable avec **Glade**

GTK+ vs. Qt

GTK+	Qt
combine bien avec GLib, GIO, GStreamer, Pango et Cairo	possède de nombreux modules : QtSql, QtXml, QtWebkit, ...
principaux systèmes d'exploitation	principaux systèmes d'exploitation + Android, iOS, Symbian, etc.
performances identiques pour les mêmes tâches	
utilisables sur $\sim$ mêmes langages	
$\sim$ 200 collaborateurs $\sim$ 3000 commits/an	$\sim$ 400 collaborateurs $\sim$ 15000+ commits/an

Pour commencer...

- Pour compiler Gtk+ en C, ajouter à la ligne de commande :
  "$(pkg-config –cflags –libs gtk+-2.0)"
- Dans le programme :
  "#include <gtk/gtk.h>"
- Ensuite, dans le main, il faut utiliser :
  "**void** gtk_init(**int** *argc, **char** ***argv)"
  /*initialisation de GTK+ et éventuellement configuration si valeurs passées en paramètre du main*/
  et plus tard :
  "**void** gtk_main(**void**)"

Pour commencer... II (quelques notions)

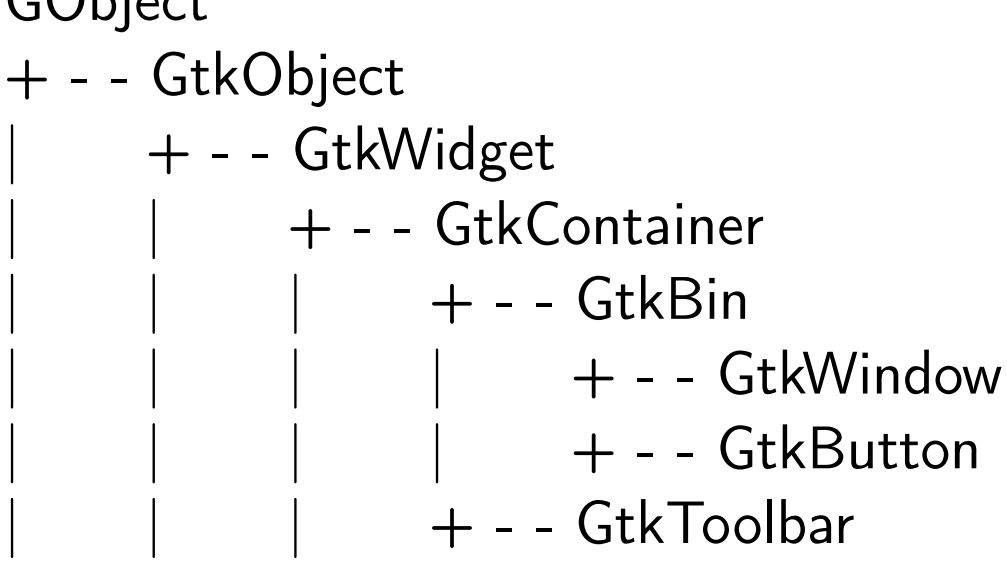
- Les widgets (**Win**dow Gad**get**) :
  - objets GTK (notion d'héritage)
  - initialiser un objet :

```
GtkWidget *maFenetre;
maFenetre = gtk_window_new(GTK_WINDOW_TOPLEVEL);
```

  - chaque type de widget a des fonctions de personnalisation de

syntaxe : **type_retour** gtk_type_effet (**paramètres**) ;
où type = type de widget (ex. window pour une fenêtre)
et effet = action à éffectuer (ex. set_relief, set_title)

Pour commencer... III (héritage GTK)



Casting

GTK_WIDGET(widget)
GTK_WINDOW(window)

GTK_OBJECT(object)
GTK_Container(container)

Des fenêtres et des boutons

- création d'une fenêtre :

```
/* déclaration */
GtkWidget *maFenetre;
/* initialisation */
maFenetre = gtk_window_new(GTK_WINDOW_TOPLEVEL);
/* affichage */
gtk_widget_show(maFenetre);
/* taille par défaut */
gtk_window_set_default_size(GTK_WINDOW(maFenetre),
320 ,200);
```

  - GTK_WINDOW_TOPLEVEL : une fenêtre normale
  - GTK_WINDOW_POPUP : une fenêtre vierge, sans bordures

les objets en GTK sont invisibles par défaut on peut utiliser :

```
void gtk_widget_show_all (GtkWidget *widget);
```

Des fenêtres et des boutons II

- création d'un bouton :

```
/* déclaration */
GtkWidget *bouton;
/* initialisation */
bouton = gtk_button_new();
/* attacher le bouton à la fenêtre */
gtk_container_add(GTK_CONTAINER(maFenetre),bouton);
```

pour l'initialisation de notre bouton on dispose de variantes :

- **GtkWidget*** gtk_button_new_with_label(const **gchar** *label);
- **GtkWidget*** gtk_button_new_with_mnemonic(const **gchar** *label);
- **GtkWidget*** gtk_button_new_from_stock(const **gchar** *stock_id);

D'autres widgets...

Exemples d'autres widgets :

- des GtkLabel, ils insèrent du texte dans l'interface graphique
- des GtkHBox/GtkVBox, permettent d'aligner des widgets horizontalement/verticalement
- des GtkTable, grille invisible pour la disposition des objets
- des GtkEntry, entrée de saisie
- des GtkImage, affichage d'images
- des boîtes de dialogue
- des GtkMenuBar/GtkMenuItem, barres de menu
- des GtkToolbar, barre d'outils (new, save, open,...)
- des GtkStatusBar, barre de status
- des GtkProgressBar, barre de progression

Les signaux

GTK+ fonctionne avec les principes des "signals" et fonctions "callback"

Action de l'utilisateur sur l'interface

||

Réception d'un signal

||

Réaction avec la fonction callback

Les signaux II

Pour gérer tous ces évenements, une fonction :

void g_signal_connect (**gpointer** *object, const **gchar** *name,
GCallback func, **gpointer** func_data);

- object : le widget dont on veut gérer l'évenement, avec la macro G_OBJECT(widget *)
- name : une chaîne de caractères contenant le nom du signal, chaque signal possible a un nom
- func : une chaîne de caractères contenant le nom de la fonction callback qui sera appelée lors de l'émission du signal, avec la macro G_CALLBACK
- func_data : un pointeur vers d'éventuels paramètres de la fonction callback

Les signaux III

Une fonction s'occupe de la boucle évènementielle :

void gtk_main (**void**) ;

Une seule façon d'y mettre fin :

void gtk_main_quit (**void**) ;

Exemple

```
g_signal_connect(G_OBJECT(boutonQuitter), "clicked",
                 G_CALLBACK(maFonctionCallback), NULL);
```

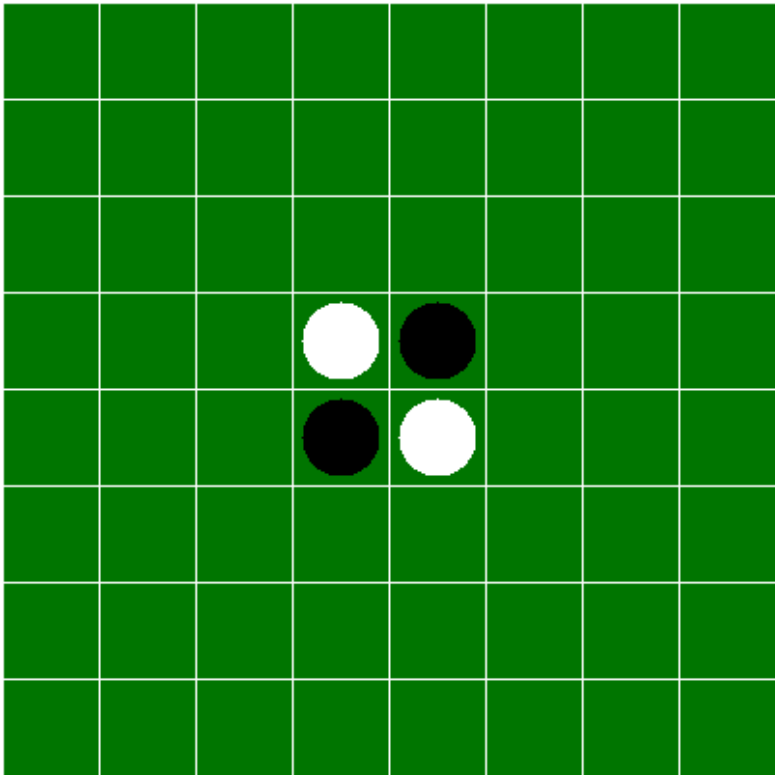
avec la fonction callback :

```
void maFonctionCallback(GtkWidget *widget,gpointer data)
{
    gtk_main_quit();
}
```

GDK

GDK :

- sous-bibliothèque de GTK+
- constituée principalement de fonctions de dessin
- permet de créer des interfaces pour des jeux

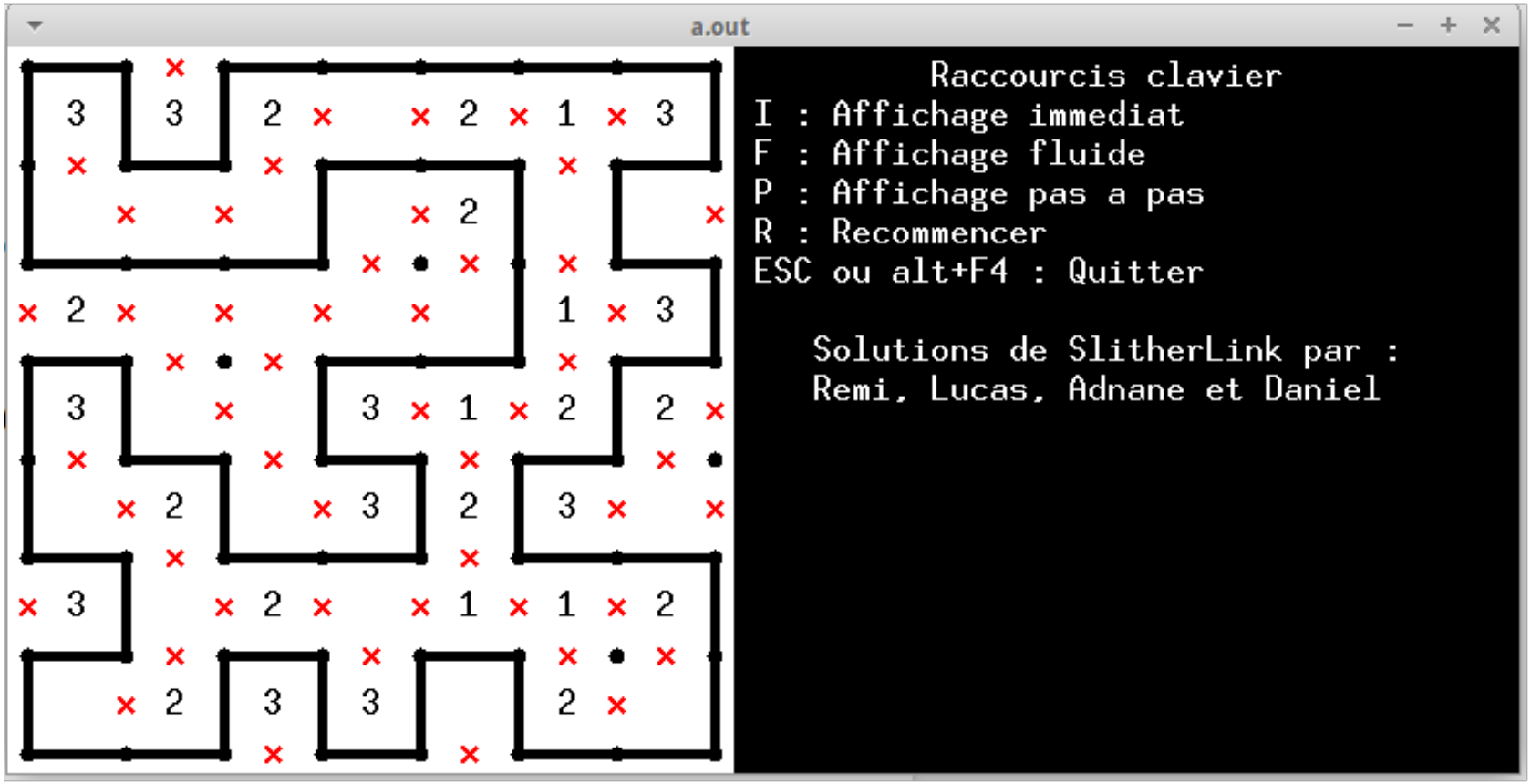


Quelques fonctions GDK

```
/* dessiner un point */
void gdk_draw_point (GdkDrawable *drawable, GdkGC *gc, gint x,
                     gint y);
/* dessiner une ligne */
void gdk_draw_line (GdkDrawable *drawable, GdkGC *gc, gint x1,
                    gint y1, gint x2, gint y2);
/* dessiner un rectangle */
void gdk_draw_rectangle (GdkDrawable *drawable, GdkGC *gc,
                         gboolean filled, gint x, gint y, gint width,
                         gint height);
/* dessiner un arc ou un cercle (arc de 360 degrés) */
void gdk_draw_arc (GdkDrawable *drawable, GdkGC *gc,
                   gboolean filled, gint x, gint y, gint width,
                   gint height, gint angle1, gint angle2);
```

Projet informatique de L2

Voici une interface graphique en GDK :



Glade

- Glade est un outil de conception d'interfaces graphiques GTK+.
- on peut donc y développer visuellement son interface
- Glade génère un fichier XML
- on racorde notre interface à un fichier ".c" :

```
GtkBuilder *builder;
builder=gtk_builder_new();
gtk_builder_add_from_file(builder, "interface.xml", &err);
```

Fin

Merci de votre attention !