Programmation d'interfaces

Cours 4 - Les composants graphiques avancés

H. Djerroud

LIASD - Université Paris 8

Automne 2020

Plan de cours

- Les Boites de dialogues
- Les ComboBox
- Les Listes

Les boîtes de dialogue

Les boîtes de dialogue sont un moyen pratique de demander à l'utilisateur de saisir ou répondre rapidement à une question ou de saisir quelques informations dans une sous fenêtre. Elles sont utilisée généralement pour :

- Afficher un message
- Poser une question binaire ...

GTK gère les boîtes de dialogue comme une fenêtre fractionnée verticalement (`GtkBox`), l'utilisateur peut alors insérer dans la partie haute les `widget` tel que les `entry`, `lablel`, ... La partie basse est généralement réservée aux boutons, ces boutons sont généralement créés avec la boite de dialogue, mais l'utilisateur peut insérer de nouveaux.

Les boîtes de dialogue

- Créer la boite de dialogue
- Récupérer le conteneur
- Créer des widgets et les insérer dans le conteneur
- Rendre visible les composants de la boite de dialogue
- Afficher la boite de dialogue
- Récupérer la réponse de l'utilisateur
- Détruite la boite de dialogue

Exemple de création d'un boite de dialogue (1/3)

```
void on_click(GtkWidget wgt, gpointer userData){
...
// création de la boite de dialogue
GtkWidget* diag;
diag = gtk_dialog_new_with_buttons
    ("Saisir une valeur",
      GTK_WINDOW(userData->win),
      GTK_DIALOG_MODAL | GTK_DIALOG_DESTROY_WITH_PARENT,
      "OK", GTK_RESPONSE_ACCEPT,
      "Annuler", GTK_RESPONSE_CANCEL,
       NULL);
...
```

Exemple de création d'un boite de dialogue (2/3)

```
...
// Récupération du conteneur
GtkWidget* diagArea = gtk_dialog_get_content_area (GTK_DIALOG (diag));

//Création d'un composant champs de saisie
GtkWidget* entry1 = gtk_entry_new();

//ajouter le champs de saisie à boite de dialogue
gtk_container_add (GTK_CONTAINER (diagArea), entry1);

//Rendre visible les composants
gtk_widget_show_all (diag);
...
```

Exemple de création d'un boite de dialogue (3/3)

```
...
// Affciher la boite de dialogue et récupérer
// la valeur selon  bouton pressé
gint res = gtk_dialog_run(GTK_DIALOG(diag));
switch(res){
  case GTK_RESPONSE_ACCEPT:
    //récupérer la valeur sasise par l'utilisateur
    // dans la boite de dialogue
    gchar* val = gtk_entry_get_text(GTK_ENTRY(entry1));
    ...
    break;
  case GTK_RESPONSE_CANCEL:
  case GTK_RESPONSE_NONE:
    break;
  }
  gtk_widget_destroy(diag);
```

Les types de boîtes de dialogues

`GtkDialogFlags`

Flag	Description
GTK_DIALOG_MODAL	Toujours au premier plan
GTK_DIALOG_DESTROY_WITH_PARENT	Détruisez la boîte de dialogue lorsque son parent est détruit
GTK_DIALOG_USE_HEADER_BAR	Boutons en haut de la fenêtre

Les types de boutons

`GtkResponseType`

Flag	Description
GTK_RESPONSE_REJECT	Rejeter
GTK_RESPONSE_ACCEPT	Accepeter
GTK_RESPONSE_OK	Ok
GTK_RESPONSE_CANCEL	Annumer
GTK_RESPONSE_CLOSE	Fermer
GTK_RESPONSE_YES	Oui
GTK_RESPONSE_NO	Non
GTK_RESPONSE_APPLY	Appliquer
GTK_RESPONSE_HELP	Aide
GTK_RESPONSE_DELETE_EVENT	Click sur le haut
GTK_RESPONSE_NONE	Aucun de ces évènements

Model-View Pattern

Les composants GTK qui affiche des listes de données tel que

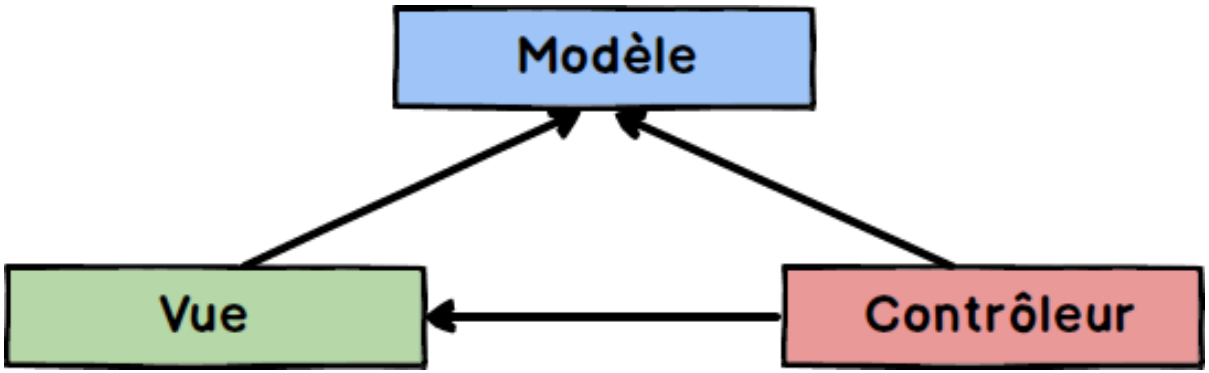
- Les combobox
- Les listes, . . .

Utilisent : *model-view pattern ou MVC*, c'est à dire une structure de données bien déterminée qui stock les données sous forme d'un arbre ou d'une liste, puis cette liste est passée à un composant graphique qui se charge d'afficher ces données.L'utilisateur quand à lui utilise des fonctions (contrôleurs) pour accéder à ces donner.

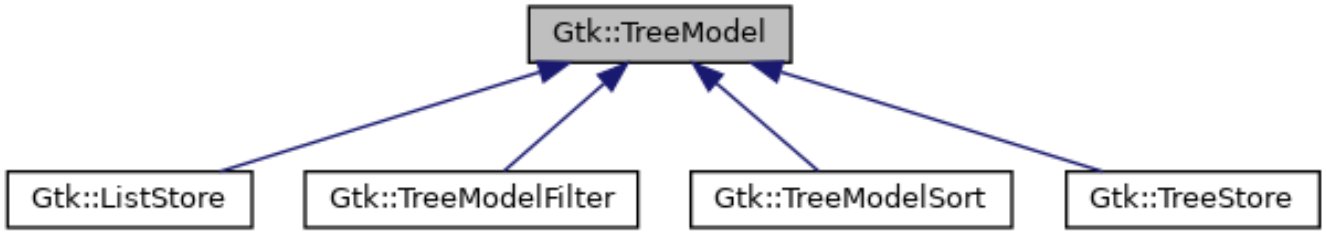
Model-View Pattern

Donc 3 composants :

- Modèle : Les données
- View : Le composant graphique
- Contrôleur : Accéder aux données via les fonction (get/set)



Les modèles avec GTK



- ListStore :Organisation de données plate sans hiérarchie (liste)
- TreeStore :Organisation de données avec hiérarchie (arbre)

ListStore

- Pour créer une ListStore, il faut indiquer le nombre de colonnes ainsi que leurs type.

```
...
GtkListStore *lst = gtk_list_store_new(2,
                                       G_TYPE_INT,
                                       G_TYPE_STRING);
...
```

Quelques type de données

Type	Description
G_TYPE_INT	Entier
G_TYPE_STRING	Chaîne de caractère
G_TYPE_BOOLEAN	Booléenne
GDK_TYPE_PIXBUF	Image
...	...

ListStore

- Pour remplir les données il faut ajouter une ligne à l'aide `gtk_list_store_append`, cette fonction donne un pointeur (iterator) sur la nouvelle ligne
- `gtk_list_store_set` insère la donnée à la position `iter` en indiquant le numéro de sa colonne et la valeur, le (-1) indique la fin de la saisie.

```
...
GtkTreeIter iter;
for (gint i = 0; i< 10 ; i ++){
    DataLang dLang;
    gtk_list_store_append (lst, &iter);
    gtk_list_store_set(lst, &iter, 0, i , 1, "Valeur", -1);
  }
...
```

Les Contrôleurs avec GTK

- Pour accéder aux donnée on utilise un genre de pointeur : *iterator*
- `TreeIter` est un *iterator* pour les `ListStore`

```
...
gint id;
gchar* val;
GtkTreeIter iter;
iter = ....;
gtk_tree_model_get (tmpLst, &iter, 0 , &id, 1, &val, -1);
...
```

Les vues avec GTK

- Pour chaque cellule il faut paramétrer l'affichage

```
...
MyComboBox *b = gtk_combo_box_new ();
...
GtkCellRenderer *cell1;
cell1 = gtk_cell_renderer_text_new ();
gtk_cell_layout_pack_start (GTK_CELL_LAYOUT (b), cell1, FALSE);
...
GtkCellRenderer *cell1;
cell1 = gtk_cell_renderer_text_new ();
gtk_cell_layout_pack_start (GTK_CELL_LAYOUT (b), cell1, FALSE);
...
```

Les switch

- `GtkSwitch` est un widget qui a deux états : *activé* ou *désactivé*. L'utilisateur peut contrôler quel état doit être actif en cliquant sur la zone vide ou en le faisant glisser.

Exemple Switch (1)

```
int main(int argc, char* argv[]){
  GtkWidget *win, *sw;
  gtk_init(&argc, &argv);
  win = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_container_set_border_width (GTK_CONTAINER (win), 10);
  g_signal_connect (G_OBJECT(win),"delete_event",
                    G_CALLBACK(gtk_main_quit) ,NULL);
  sw = gtk_switch_new();
  gtk_switch_set_active(GTK_SWITCH(sw), TRUE);
  g_signal_connect (G_OBJECT(sw),"notify::active",
                    G_CALLBACK(on_switch) ,NULL);
  gtk_container_add(GTK_CONTAINER(win),sw);
  gtk_widget_show_all(win);
  gtk_main();
}
```

Exemple Switch (2)

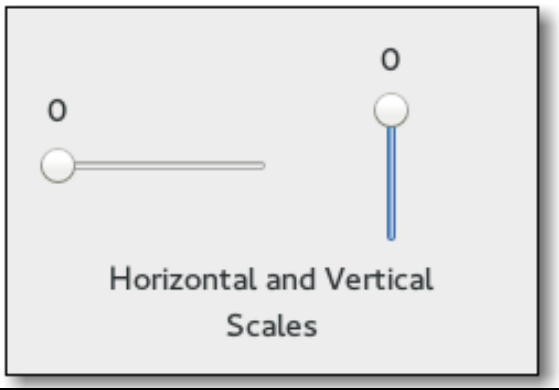
```
...
void on_switch (GtkSwitch *psw, gpointer   user_data){
  g_print("Changement d'état : ");
  if (gtk_switch_get_active (GTK_SWITCH (psw))){
    g_print("Active \n");
    return;
  }
  g_print("Not active \n");
}
...
```

Les GtkRange

- *GtkRange* est la classe de base commune pour les *widgets* qui implémente un ajustement, par exemple `GtkScale` et `GtkScrollbar`.
- En plus des signaux de surveillance des paramètres de réglage, `GtkRange` fournit des propriétés et des méthodes pour influencer la sensibilité des «steppers». Il fournit également des propriétés et des fonctions pour définir un *niveau de remplissage* sur les widgets de plage comme la fonction `gtk_range_set_fill_level()`.

Les Scales

- Un `GtkScale` est un contrôle de curseur utilisé pour sélectionner une valeur numérique. Pour l'utiliser, vous voudrez probablement étudier les méthodes de sa classe de base, *GtkRange*, en plus des méthodes de *GtkScale* lui-même. Pour définir la valeur d'une échelle, vous utiliseriez normalement `gtk_range_set_value ()`. Pour détecter les changements de valeur, vous utiliseriez normalement le signal «valeur modifiée».



Exemple Scale (1)

```
int main(int argc, char* argv[]){
  GtkWidget *win, *scal;
  gtk_init(&argc, &argv);
  win = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_container_set_border_width (GTK_CONTAINER (win), 10);
  g_signal_connect (G_OBJECT(win),"delete_event",G_CALLBACK(gtk_main_qu
  scal = gtk_scale_new_with_range(GTK_ORIENTATION_HORIZONTAL,0,100,1);
  gtk_widget_set_size_request (scal,200,-1);
  g_signal_connect (G_OBJECT(scal),"change-value",G_CALLBACK(on_scale),
  gtk_container_add(GTK_CONTAINER(win),scal);
  gtk_widget_show_all(win);
  gtk_main();
}
```

Exemple Scale (2)

```
...
void on_scale (GtkRange *prange, gpointer   user_data){
  gint val =  gtk_range_get_value(GTK_RANGE(prange));
  g_print("val = %i \n", val);
}
...
```