tutoriel

Tutoriel Glade 3

Hello again.

Voici des techniques pour développer des applications utilisant Glade3.

Notre exemple d'application sera un visualiseur et un éditeur de graphes simple mais entièrement fonctionnel. Voici ce que nous apprendrons avec cet exemple :

- aborder un nouveau projet avec le concepteur d'interface graphique Glade3
- créer des éléments visuels d'application avec Glade3
- créer des objets de stockage de données pour l'application avec Glade3
- connecter les fonctions de rappel aux signaux des widgets.
- dessiner sur des widgets en utilisant la bibliothèque graphique Cairo.
- utiliser Pango de Cairo en utilisant pangocairo.
- créer des menus et une barre d'outils basés sur des actions.

Premiers pas

Le travail d'aujourd'hui sera divisé en deux parties :

- Créer un “plan” de l'interface graphique de notre application en fonction des besoins.
- Rédaction de l'arborescence des widgets, basée sur le plan de la section précédente.

Maintenant, quelles sont les exigences pour notre application. De toute évidence, il doit pouvoir afficher les données sous forme numérique et graphique. Nous voulons également pouvoir ajouter de nouveaux points, en supprimer, en les réorganiser et les modifier. Nous voulons également pouvoir afficher des points de repère, des lignes de connexion ou les deux sur la carte. Et c'est à peu près tout pour les besoins initiaux. Nous allons laisser de la place pour une extension future de nos plans, au cas où nous déciderions d'ajouter quelque chose plus tard.

Maintenant voici la partie amusante, l'interface de dessin. Je préfère le faire sur papier avec un crayon, mais n'hésitez pas à expérimenter. Pour un exemple de projet, je suis venu avec ce design :

Maquette d'interface

Passons maintenant à la dernière chose à faire aujourd’hui: écrire l’arborescence des widgets. Dans GTK +, tout commence par GtkWindow de niveau supérieur, qui servira de racine à l’arborescence de nos widgets.

Notre fenêtre principale sera divisée en quatre sections verticales : une pour la barre de menus, une pour la barre d’outils, une pour la partie centrale où se déroulera toute l’action et une dernière pour la barre d’état. Comme notre fenêtre principale (GtkWindow) ne peut contenir qu'un seul widget enfant, nous avons besoin de GtkVBox dans lequel nous allons emballer, de bas en haut : GtkStatusbar, widget pour la partie centrale, GtkToolbar et GtkMenuBar.

La partie centrale devra être divisée en sections horizontales: une pour le tableau de données, une

pour les contrôles ponctuels et une pour la zone d'affichage. Donc, cela nécessitera GtkHBox. De quels widgets aurons-nous besoin à l'intérieur ? Pour la table de données, nous utiliserons GtkTreeView, qui est compressé dans GtkScrolledWindow pour permettre le défilement. Pour les contrôles de points, nous aurons besoin de GtkVButtonBox qui hébergera nos boutons.

Passons à la zone d'affichage. Nous avons encore deux parties: la partie supérieure affichera le graphique et la partie inférieure qui contiendra les boutons de contrôle. Nous aurons donc besoin d'une autre GtkVBox pour toute la section. La partie inférieure sera représentée par GtkHButtonBox avec GtkToggleButtons, alors que la partie supérieure mérite un autre paragraphe.

Pourquoi un autre paragraphe? Parce que nous aurons besoin d'ajouter quelques widgets pour obtenir la mise en page souhaitée. Comme vous pouvez le constater sur mon dessin, je souhaite que la zone du graphique soit centrée dans la partie d'affichage. Afin de pouvoir ajouter un zoom à la zone de graphique, nous avons également besoin d'une sorte de widget de défilement. Et comment assembler tout cela? Nous allons d'abord ajouter GtkScrolledWindow à GtkVBox du paragraphe précédent. Pour faire défiler le contenu, nous devons placer GtkViewport dans GtkScrolledWindow. Dans GtkViewport, nous ajouterons GtkAlignment, qui se chargera de centrer la zone de la carte. Dans GtkAlignment, nous ajouterons GtkFrame, ce qui ajoutera une ombre à la zone du graphique. Enfin, nous ajoutons une zone GtkDrawing à l'intérieur de GtkFrame. Et nous avons fini.

Si nous convertissons cette description en une représentation arborescente, nous obtenons ceci :

```
GtkWindow
└─ GtkVBox
   ├─ GtkMenuBar
   │  └─ /* Menus here */
   ├─ GtkToolbar
   │  └─ /* Tool buttons here */
   ├─ GtkHBox
   │  ├─ GtkScrolledWindow
   │  │  └─ GtkTreeView
   │  ├─ GtkVButtonBox
   │  │  └─ /* Buttons here */
   │  └─ GtkVBox
   │     ├─ GtkScrolledWindow
   │     │  └─ GtkViewport
   │     │     └─ GtkAlignment
   │     │        └─ GtkFrame
   │     │           └─ GtkDrawingArea
   │     └─ GtkHButtonBox
   │        └─ /* Check buttons here */
   └─ GtkStatusBar
```

Encore une explication. Lors de l'ajout de GtkTreeView à GtkScrolledWindow, je n'ai pas utilisé GtkViewport en tant que widget adaptateur, alors que l'ajout de GtkAlignment en nécessitait un. Pourquoi? En ce qui concerne GtkScrolledWindow, il existe deux types de widgets: ceux qui prennent en charge le défilement natif et ceux qui ne le font pas. GtkTreeView, GtkTextView, GtkIconView et GtkViewport prennent en charge le défilement et peuvent être ajoutés directement dans GtkScrolledWindow. Tous les autres widgets ont besoin de GtkViewport en tant qu'adaptateur.

Pré-requis

Première étape

Autres étapes

Conclusion

Problèmes connus

Voir aussi

- **(en)** http://
- **(fr)** http://

Basé sur « Article » par Auteur.