TP wxWidgets n°2

Objectif général

Le but des 5 TP est d'apprendre à créer et manipuler des interfaces graphiques, en créant un petit programme destiné à l'affichage et à la saisie de triangles. Pour cela, nous allons utiliser la librairie multi-plateforme wxWidgets (anciennement wxWindows), à titre d'exemple. Vous devrez programmer en C++. Tout au long des TP, vous pouvez utiliser la documentation présente sur le site web de wxWidgets :
http://docs.wxwidgets.org/stable/wx_classref.html#classref

Objectif de ce TP

Dans ce deuxième TP, nous allons associer des comportements aux différents widgets et aux événements qui peuvent leur arriver.

Manipulations

1. Evénements liés à la sélection d'une rubrique dans un menu

Création des fonctions et lien avec les événements
Lorsqu'une rubrique (un élément de la classe wxMenuItem) est sélectionnée dans un menu, une action doit être éffectuée. Cette action dépend de la rubrique. On aura donc une action, autrement dit une fonction, par rubrique différente.
Pour lier les événements aux fonctions décrivant les actions à effectuer, on place toutes ces relations dans la table des événements. Cette table des événements commence toujours par une instruction `BEGIN_EVENT_TABLE`, et termine toujours par une instruction `END_EVENT_TABLE`, et est généralement placée avant le constructeur de la classe correspondant aux widgets concernés. C'est entre ces deux instructions que se placent les associations. La table des événements correspondant aux rubriques des menus est celle qui est placée juste avant le constructeur de CMainFrame. Vous devez donc ajouter à cette table tous les événements susceptibles de se produire.
Une association événement-widget-fonction se déclare par une instruction du type :

```
NOM_EVENEMENT ( ID_WIDGET, classe::nom_fonction)
```

Dans le cas présent, les événements pouvant survenir sont des événements de type `EVT_MENU` (signifiant "on a cliqué sur une rubrique de menu"). Les `ID_WIDGETS` sont ceux des différentes rubriques qui ont été ajoutées aux menus lors du TP précédent par les instructions `Append`, et qui sont stockés dans un enum. Les fonctions n'ont pas encore été déclarées, mais nous allons le faire ensuite.
Exemple d'association à ajouter :

```
EVT_MENU(MENU_NEW, CMainFrame::OnNew)
```

Nous créerons par la suite une fonction `OnNew`, qui sera alors appelée automatiquement dès que la rubrique "Nouveau" est cliquée.
De la même façon, créez toutes les associations correspondant aux rubriques des menus que vous avez créées lors du précédent TP, en donnant toujours aux fonctions des noms commençant par "`On`…". Par convention, on désigne ainsi les fonctions répondant à des actions de l'utilisateur.

Ensuite, il faut créer toutes les fonctions correspondantes. Ce seront des fonctions membres privées de CMainFrame. Exemple :

- déclaration dans mainframe.h, comme membre de CMainFrame :

```
void OnNew(wxCommandEvent& event);
```

- définition dans mainframe.cpp :

```
void CMainFrame::OnNew(wxCommandEvent& event)
{
}
```

Ces fonctions prennent en argument une variable de type `wxCommandEvent` qui sera l'événement qui s'est produit et qui a conduit à l'appel de cette fonction. Cet événement sera envoyé de façon totalement automatique.
Créez les fonctions correspondant aux noms que vous avez donnés dans la table des événements. Elles seront pour l'instant vides.

Contenu des fonctions

Nous allons maintenant mettre du contenu dans certaines de ces fonctions.
Commençons tout d'abord par la fonction correspondant à la sélection de la rubrique "Quitter". Cette fonction doit fermer l'application. Nous placerons donc dans cette fonction un appel à la fonction `Close`, membre de `wxWindow` dont dérive `wxFrame` dont dérive `CMainFrame` :

```
Close(TRUE);
```

Désormais, on peut fermer l'application en utilisant la rubrique "Quitter".

Nous allons pour l'instant laisser vides les fonctions d'ouverture, de sauvegarde, et de nouveau fichier triangle.

Occupons-nous de la fonction qui gère le masquage / affichage de la barre d'outils, que nous appellerons par exemple `OnToolBar`. Il faut lui ajouter du code spécifiant que si la barre d'outils est visible, alors on doit la cacher, et inversement. Utilisez pour cela des fonctions membres de la classe `wxWindow` dont dérive `wxToolBar`.

Les quatre fonctions suivantes, appelées lorsqu'on sélectionne une des quatre rubriques des menus "Options" et "Aide", doivent chacune ouvrir la boîte de dialogue correspondante. Ecrivez le code permettant de le faire.
De plus, nous allons également demander à la fonction correspondant à la rubrique "Gestion des triangles" d'effectuer quelques opérations sur les widgets de sa boîte de dialogue. Les informations modifiant le contenu ou l'apparence des widgets doivent être effectuées avant la demande d'affichage de la boîte de dialogue pour être prises en compte. Nous allons tout d'abord demander l'effacement de tout contenu de la listbox grâce à la fonction `Clear` membre de `wxListBox`. Ensuite, nous allons placer deux chaînes de caractères (par exemple "triangle 1" et "triangle 2") comme éléments de cette listbox, grâce à la fonction `Append` membre de `wxListBox`. Nons allons enfin préciser que la sélection par défaut doit être placée sur le deuxième élément de la listbox grâce à la fonction `SetSelection` membre de `wxListBox`. Vérifiez bien que tout fonctionne.

2. Evénements liés à des actions sur des widgets

Evénements liés aux widgets de la barre d'outils

Les trois premiers boutons de la barre d'outils doivent produire les mêmes résultats que lorsqu'on clique sur les 3 premières rubriques du menu "Fichier". Cela sera fait automatiquement sans intervention de notre part, puisque lors de leur création nous leur avons affecté les mêmes identifiants, de façon à rediriger les événements des 2 sortes de widgets vers les mêmes fonctions de traitement.
Par contre, le quatrième bouton n'a pas encore de fonction de traitement.
Vous devez donc la créer.

Evénements liés aux widgets des boîtes de dialogue
Il reste maintenant à créer les fonctions qui seront appelées automatiquement lors de l'utilisation des widgets des différentes boîtes de dialogue.
Vous devez pour cela ajouter toutes les informations nécessaires dans les fichiers dialogs.h et dialogs.cpp : pour chaque boîte de dialogue, listez les événements possibles, ajoutez les associations événement-widget-fonction dans la table des événements de cette boîte de dialogue, et créez et déclarez les fonctions correspondantes.

Attention cependant, les boutons OK présents sur chaque boîte de dialogue ne nécessitent pas de fonction de traitement ni d'association dans la table des événements. En effet, les événements qui leur sont liés sont traités automatiquement par wxWidgets car nous leur avons donné l'identifiant réservé wxID_OK. Il serait tout de même possible de court-circuiter ce traitement automatique pour ajouter des traitements juste avant la fermeture de la boîte de dialogue, mais nous ne le ferons pas dans le cadre de notre TP.

Attention également, ici les événements ne seront plus du type "`EVT_MENU`". Vous trouverez les types d'événements dans la doc. Par exemple, un événement de type "appui sur un bouton" est `EVT_BUTTON`, décrit dans la page sur la classe `wxButton` de la doc.

Les événements pour une boîte de dialogue sont à ajouter dans la table d'événements qui précède les implémentations de fonctions de la classe correspondant à la boîte de dialogue.
Vous devez entre autres ajouter une fonction correspondant à l'appui sur le bouton "Propriétés" de la boîte de dialogue de "Gestion des triangles", dans laquelle on va ouvrir la boîte de dialogue "Propriétés", et faire quelques initialisations :

- ajouter dans la zone de texte, le texte qui était sélectionné dans la listbox de la boîte de dialogue "gestion des triangles"
- mettre une valeur par défaut au spin button
- mettre une sélection par défaut à la radiobox

3. Ajout de quelques variables membres à CMainFrame

A vous de jouer !...
Vous devez maintenant ajouter quelques variables membres publiques à CMainFrame, et ajouter le code nécessaire au maintien à jour de toutes ces variables en fonction des actions de l'utilisateur :

- une variable *epaisseurtraitcourante* de type entier, qui sera mise à jour par l'utilisation du slider de la boîte de dialogue "Epaisseur trait"
- une variable *couleurcourante* de type wxColour, qui sera mise à jour par l'utilisation de la radiobox de la boîte de dialogue "Couleur"
- une variable booléenne *is_drawing*, qui nous indiquera si le bouton de dessin sur la barre

d'outils est enfoncé ou non.