TP wxWidgets n°5

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 5ème TP, nous allons apprendre à utiliser les menus contextuels surgissants, ainsi qu'à fournir un fichier d'aide.

Manipulations

1. Menus contextuels

Ces menus surgiront lorsqu'on appuiera sur le bouton droit de la souris, et le contenu du menu devra dépendre de l'endroit où se trouvait le curseur de la souris au moment où l'on a appuyé sur le bouton. Pour cela, nous allons commencer par définir une fonction `OnRightDown` dans `OpenGLCanvas`.

1.1 Menu contextuel par défaut
Dans cette fonction `OnRightDown`, commençons par créer un premier menu "popup" qui sera celui par défaut. Ce menu sera structuré de la façon suivante :

- un menu général à afficher, de type `wxMenu` nommé par exemple popupmenu
- 3 sous-menus de type `wxMenu*` : "Fichier", "Gestion", "Valeurs courantes"
- des items à ajouter dans chaque sous-menu :
  - Fichier : "Ouvrir fichier" et "Sauvegarder fichier"
  - Gestion : "Gestion des triangles"
  - Valeurs Courantes : "Couleurs courantes" et "Epaisseur courante"

Pour ajouter un item à un sous-menu, on procède comme lors du premier TP, grâce à la fonction `Append` avec comme arguments un identificateur de menu et une chaîne de caractères à afficher. Par exemple :

```
submenu1->Append(MENU_OPEN, wxT("Ouvrir fichier"));
```

Ajoutons ainsi tous les items dans les sous-menus.
Ensuite, ajoutons les sous-menu au menu général, également grâce à la fonction `Append` :

```
popup.Append(POPUP_SUB1, wxT("Fichier"), submenu1);
```

Il ne reste plus qu'à afficher ce menu grâce à la fonction PopupMenu de la classe `wxWindowBase` :

```
PopupMenu( &popup, event.GetX(), event.GetY() );
```

1.2 Clic droit sur un triangle

Lorsque l'on clique sur un triangle, on souhaite pouvoir accéder directement à certaines fonctionnalités sur ce triangle : supprimer ce triangle, afficher les propriétés de ce triangle.

Pour cela, il nous faut tout d'abord détecter si l'on a cliqué sur un triangle, et si c'est le cas, il faut déterminer sur quel triangle on a cliqué.
Ajoutons le fichier triangle.cpp à notre projet. Il contient la définition d'une nouvelle fonction de la classe `Triangle` qui nous permettra de savoir si un point est à l'intérieur de ce triangle ou non. Ajoutez le profil de cette nouvelle fonction dans le fichier triangle.h.

Dans la fonction `OnRightDown`, il faut donc commencer par faire un test : si l'on n'a pas cliqué sur un triangle, alors afficher le menu par défaut, et si l'on a cliqué sur un triangle, alors afficher un autre menu : ce nouveau menu sera composé uniquement de 2 items (sans sous-menu) à "Propriétés de ce triangle" et "Supprimer ce triangle".

Pour ce test, il est préférable de décomposer le problème pour éviter de surcharger la fonction `OnRightDown` à faire une petite fonction `int Est_dans_triangle(int x, int y)`, qui parcourt les triangles et donne le numéro de celui sur lequel on a cliqué, ou -1 sinon.

Attention à bien faire ici aussi la conversion coordonnées de la fenêtre <-> coordonnées OpenGl.

On peut stocker le résultat de ce test dans un nouvel entier membre de la classe `OpenGLCanvas`, que l'on nommera par exemple `selected_tri`.

Il faut ensuite créer les deux fonctions `OnContextPptes` et `OnContextSuppr` qui seront appelées lorsqu'on choisira un des deux éléments de ce menu contextuel. Ces fonctions utiliseront la variable `selected_tri` pour savoir sur quel triangle on travaille.

2. Fichier d'aide

Nous souhaitons proposer à l'utilisateur une aide.
Pour cela, nous allons ajouter une nouvelle rubrique dans le menu Aide de l'application : rubrique "Ouvrir l'aide".

Il faut également ajouter :

- la fonction (nommée par exemple `OnHelp`) qui sera appelée lorsqu'on choisira cette rubrique, ainsi que sa déclaration
- le lien avec l'événement dans la table des événements
- une variable "help" de type `wxHtmlHelpController`
- les lignes suivantes dans MyApp::OnInit(), pour initialiser le fichier d'aide :

```
if ( !m_MainFrame->help.Initialize("HELP") )
{
 wxLogError(wxT("Cannot initialize the help system, aborting."));
 return FALSE;
}
```

- la ligne suivante dans `OnHelp`, afin de lancer le visionnage du fichier d'aide:

```
help.DisplayContents();
```

Pour pouvoir tester cela, vous devez télécharger les fichiers suivants :
HELP.hhc, HELP.hhp, Introduction.htm, Demarrage.htm, Raccourcis.htm