TP wxWidgets n°4

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 4ème TP, nous allons afficher les triangles avec OpenGl, et voir comment interagir entre la fenêtre OpenGl et la souris pour saisir des triangles.

Manipulations

1. Création du canevas OpenGl

Commençons par créer le canevas OpenGl. Pour cela, nous allons créer une nouvelle classe, que nous nommerons `OpenGLCanvas`, dans un nouveau fichier nommé par exemple openglcanvas.h. Cette classe devra dériver de la classe de base `wxGLCanvas`. Ses fonctions seront implémentées dans un fichier openglcanvas.cpp.
Afin de pouvoir compiler ces nouveaux fichiers, n'oubliez pas de modifier le makefile. Vous devez également ajouter dans le makefile l'option --gl-libs à la suite de --libs, pour pouvoir faire le lien avec les librairies OpenGl et produire correctement l'exécutable.

Dans cette nouvelle classe, on va d'abord redéfinir le constructeur et le destructeur :

```
OpenGLCanvas::OpenGLCanvas(wxWindow *parent, wxWindowID id,
                        const wxPoint& pos, const wxSize& size,
                        long style, const wxString& name):
                        wxGLCanvas(parent, id, pos, size, style, name)
{
}

OpenGLCanvas::~OpenGLCanvas(void)
{
}
```

On va également créer quelques fonctions membres privées, qui seront appelées lors des rafraîchissements de fenêtre. Ces fonctions sont : OnPaint (toujours appelée lorsque la fenêtre doit être redessinée), OnSize (appelée lorsque l'utilisateur change la taille de la fenêtre), OnEraseBackground (nécessaire pour éviter le phénomène de "flashing").
Voici leurs profils :

```
void OnPaint( wxPaintEvent& event );
void OnSize( wxSizeEvent& event );
void OnEraseBackground( wxEraseEvent& event );
```

Ne pas oublier de créer une table des événements, et de faire un lien entre les événements devant déclencher le rafraîchissement (`EVT_PAINT, EVT_SIZE, EVT_ERASE_BACKGROUND`) et ces fonctions.

1.1 La fonction OnPaint
Cette fonction étant appelée à chaque fois qu'on redessine la fenêtre, elle doit contenir tout ce qui est nécessaire pour tracer les triangles, qui devront être tracés à chaque fois que la fenêtre est redessinée.

Attention à ne pas trop surcharger cette fonction d'instructions, il ne faut y mettre que le strict nécessaire. En effet, lorsqu'on manipulera les objets à la souris, cette méthode pourra être rappelée jusqu'à une centaine de fois par seconde. Si ce code est trop lourd, l'application ne pourra pas suivre et donnera une impression de lenteur.

Dans cette fonction, on doit donc trouver tout d'abord la mise en place d'un "device context" pour le dessin, ainsi qu'une instruction pour le déclarer comme contexte courant :

```
wxPaintDC dc(this);
SetCurrent();
```

Ensuite viennent les instructions OpenGl pour le dessin à proprement parler, que nous placerons dans une fonction séparée appelée par exemple Draw(), et enfin une instruction pour changer de buffer (on est en mode double buffer) :

```
SwapBuffers();
```

1.2 La fonction OnSize
Dans cette fonction, on doit d'abord commencer par appeler la fonction OnSize de la classe de base, afin que soient effectuées toutes les opérations de base nécessaires lors d'un redimensionnement :

```
wxGLCanvas::OnSize(event);
```

Ensuite, on va ajouter des informations de redimensionnement de l'espace réservé à OpenGl. On récupère la taille de la fenêtre cliente, c'est-à-dire l'endroit de la fenêtre d'application dans lequel on va dessiner les triangles :

```
int w, h;
GetClientSize(&w, &h);
```

puis on indique à OpenGl ces dimensions pour initialiser la taille de l'espace à réserver pour le dessin :

```
glViewport(0, 0, (GLint) w, (GLint) h);
```

1.3 La fonction OnEraseBackground
Cette fonction ne contient rien.

Afin de pouvoir utiliser ce canevas OpenGl, vous devez en créer une instance. On va donc ajouter un pointeur vers un `OpenGLCanvas` comme variable publique de la classe `CMainFrame`. Ce pointeur sera initialisé en ajoutant un `"new OpenGLCanvas(…);"` quelque part (ex: là où on a créé les menus, ou bien dans le constructeur de CMainFrame).

2. Dessin des triangles avec OpenGl

Occupons-nous de la fonction Draw que nous avons évoquée tout à l'heure.
Dans cette fonction, nous allons regrouper toutes les instructions OpenGl concernant le dessin.

Tout d'abord, indiquons quel mode de projection nous allons utiliser. Etant donné que nous souhaitons tracer uniquement des triangles 2D, nous utiliserons une simple projection orthogonale. Nous placerons le centre du repère (point de coordonnées (0;0)) au centre de la fenêtre :

```
glMatrixMode( GL_PROJECTION );
glLoadIdentity();
int w, h;
GetClientSize(&w, &h);
glOrtho(-w/2., w/2., -h/2., h/2., -1., 3.);
```

Initialisons ensuite l'autre matrice à l'identité :

```
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
```

Puis effaçons le fond en lui donnant une couleur bleutée :

```
glClearColor( .3f, .4f, .6f, 1 );
glClear( GL_COLOR_BUFFER_BIT);
```

Il ne reste plus qu'à faire une boucle pour dessiner tous les triangles du tableau grâce aux primitives de dessin vues en cours (`glBegin, glVertex, glColor, glEnd`). Les triangles devront être dessinés avec un bord noir et un remplissage de la couleur définie dans la variable *colour*. Pour faire cela, il faudra tracer d'abord un triangle plein avec la couleur adéquate, puis redessiner par-dessus un deuxième triangle de mêmes coordonnées, non rempli, de bord noir. L'épaisseur du bord noir est définie par la variable *thickness* de la structure *triangle*. La primitive `glLineWidth(GLfloat epaisseur)` permet de modifier l'épaisseur du bord d'un triangle non rempli.

Pour toute interrogation sur les fonctions OpenGl, merci de consulter la doc en ligne (manuel de référence : blue book) :
http://www.glprogramming.com/blue/
avant d'appeler au secours…

Pour tester, essayez de charger le fichier triangles fourni lors du précédent TP, et d'afficher les triangles qu'il contient.

3. Interaction avec la souris : tracé de triangles

Pour interagir avec la souris, vous devez définir trois fonctions : une qui sera appelée dès que la souris bouge, et que nous nommerons `OnMouseMove`, une qui sera appelée dès qu'on appuie sur le bouton de gauche, et que nous nommerons `OnLeftDown`, et une qui sera appelée dès qu'on relâche le bouton de gauche, et que nous nommerons `OnLeftUp`.
Ces trois fonctions prennent en paramètre un événement souris de type `wxMouseEvent&`.
Faites les bonnes associations dans la table des événements (trouvez les bons événements dans la doc).

Le dessin d'un triangle se fait en 3 phases :

1. On saisit le premier point en appuyant sur le bouton gauche. A partir de ce moment, si l'on bouge la souris, un segment doit apparaître entre l'endroit du premier point et la position courante de la souris. Ce segment doit être mis à jour à chaque fois que la souris bouge. A tout moment, ce segment doit être noir et avoir l'épaisseur choisie dans la boîte de dialogue de l'épaisseur.
2. On saisit le deuxième point. Après cela, si l'on bouge la souris, un triangle doit apparaître reliant les deux premiers points saisis et la position courante de la souris. Ce triangle doit être mis à jour à chaque fois que la souris bouge. Ce triangle doit être rempli avec la couleur définie par la boîte de dialogue de choix de couleur, et doit avoir un bord noir de l'épaisseur choisie par la boîte de dialogue de l'épaisseur.
3. On saisit le troisième point ce qui mémorise définitivement le triangle dans le tableau. Après cela, on peut à tout moment commencer un nouveau triangle, à condition que le tableau de triangles ne soit pas déjà plein.

Voici donc ce que doivent faire les fonctions :

- `OnMouseMove` :
  Si on est en mode de dessin, selon que l'on en est à l'étape 1 ou à l'étape 2 on doit tracer un segment ou un triangle.
- `OnLeftUp` :
  Si on est en mode de dessin, selon l'étape à laquelle on est, on doit mémoriser le premier point et les informations de couleur et d'épaisseur, ou bien mémoriser le second point, ou bien mémoriser le troisième point et placer le triangle complet dans le tableau. Si aucun triangle n'était présent avant dans le tableau, et que l'on vient de dessiner un premier triangle, alors il ne faut pas oublier de dégriser la rubrique "Gestion des triangles" dans le menu.

Pour manipuler les informations venant de la souris, vous aurez besoin d'utiliser entre autres les fonctions `GetX()` et `GetY()` de l'événement souris passé en paramètre (voir la doc sur `wxMouseEvent`).

A vous de jouer !...

Pour plus d'informations sur OpenGL, vous pouvez également consulter le manuel du programmeur (red book) :
- HTML : http://www.glprogramming.com/red/
ou encore le site officiel d'OpenGL : http://www.opengl.org