PoP C++ Série 6

H1 :
1) Usage de GTKmm 3 pour l'interface graphique utilisateur : séparation des fonctionalités

H2 : MOOC MOOC Introduction à la programmation orientée objet (en C++)
Série semaine 5: Polymorphisme

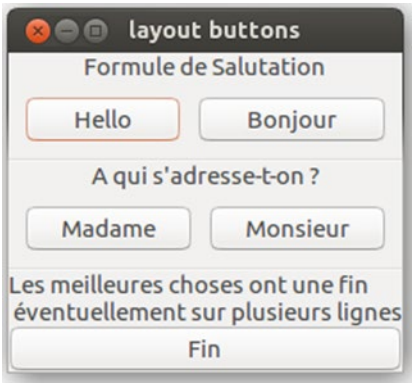
Usage de GTKmm 3 pour l'interface graphique utilisateur : séparation des fonctionnalités

Exercice 1. (niveau 0) : Rappel pour le dessin

Cet exemple comporte deux modules ; le module **myevent** étend la classe **MyArea** vue la semaine dernière et la combine avec la classe **MyEvent** qui établit un lien entre l'action des boutons et l'affichage.

Exercice 2 (niveau 1) : **Layout et Label**

reproduire cette interface :



A partir du concept de **Gtk ::Box** exploité dans l'exercice 1 ci-dessus, il s'agit ici de produire un programme qui combine l'affichage de texte avec **Gtk ::Label** et des Buttons sur plusieurs lignes et (éventuellement) colonnes.

Un clic sur l'un des 4 boutons Hello, Boujour, Madame, Monsieur doit produire l'affichage de ce mot dans le terminal. Un clic sur Fin doit quitter l'application.

Pour le layout avec **Gtk ::Box** relire l'exercice1 ; il n'y a pas de nouveauté.

Pour afficher du texte le widget **Gtk ::Label** est le type à utiliser.

Pour chaque chaine de caractère que l'on veut afficher :

- Dans l'interface du module définissant la classe **layoutbuttons**.h :
  - ajouter un attribut dans la liste des « Member widgets » (avec les attributs Buttons). Par ex :
    - **Gtk ::Label m_Label_Salutation ;**

- Dans l'implémentation du module **layoutbuttons**.cc
    - Initialiser le Label dans la liste d'initialisation du constructeur
        - **m_Label_Salutation(**`"Formule de Salutation")`

    - Pour avoir le texte sur plusieurs lignes il suffit d'insérer le carectère de contrôle **\n** dans la chaîne à afficher

    - Utiliser la méthode **pack_start** pour placer le label dans une **Gtk ::Box** comme traité dans l'exercice 1

Un Label n'a pas de signal_handler.

Le dernier élément nécessaire pour bien reproduire l'image précédente est un séparateur. On dispose du type **Gtk ::Separator** pour cela. Chaque séparateur doit avoir un attribut dans la liste des « Members widgets ». Par ex : **Gtk ::Separator m_Separator1 ;**

Un séparator n'a pas besoin d'être initialisé. On doit seulement le placer au bon moment dans l'interface en utilisant **pack_start** sur les **Gtk ::Box**.

Exercice 3 (niveau 1) : **Coordinates conversion for the Model space to the Application's window width and height system (in English)**

a) Write the code of the 2 conversion functions Xf and Yf from Slide 12 of the lecture from previous week (Programmation orientee Projet: Bases de GTKmm / Serie 5 niveau0 + conversions). It should be a small program that asks (Xm, Ym) values as input and write in the terminal the resulting (Xf, Yf) values for some predefined values of the framing [XMIN, XMAX] x [YMIN, YMAX] and of the window height and aspect ratio (defined as width/height). Choose the values of the framing so that they are different from the default window space, otherwise this question is pointless. Just pay attention to choose the width of the window so that both rectangular spaces (the framing and the window) have the same aspect ratios.

b) By combining the result of (a) with the example from week 5 (GTKddrawingarea) create an Application window drawing a red cross ("X") in the center of the chosen framing [XMIN,XMAX] x[YMIN,YMAX] screen. Verify that your conversion functions ensures that the cross is also located in the center of the graphic window. Then if you manually change the size of the window[1] the X shape should deform so that it is always inside the new window size (without changing the conversion functions).

c) Now let's suppose we want to have the same initial state = the X cross is centered in the window BUT we don't want to introduce a distortion when changing the window size. A distortion means that the angles between the X bars are changing as we change the window size because the scaling factor is difference along X and Y. How should we adapt the conversion formulas to ensure 1) seeing the full X and 2) without distortion.

[1] Hint: on_draw() from MyArea class is called each time you modify the size of your window