

# PoP C++ Série 6 corrigé

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

La solution ci-dessous choisit de faire afficher un passage à la ligne seulement quand on clique sur le destinataire (Madame ou Monsieur). L'ensemble du code source étant disponible dans un fichier archive GTKLabel, on ne reprend ici que layoutbuttons.cc

Remarquer la différence d'initialisation de m\_Box\_Top d'une part (Verticalement) et de M\_Box1 et m\_Box2 d'autre part (Horizontalement). Le label m\_Fin apparaît sur 2 lignes dans l'interface grâce à \n



L'organisation spatiale de l'interface est obtenu avec la bonne répartition des widgets entre m\_Box\_Top et les deux autres.

```
#include <iostream>
#include "layoutbuttons.h"
using namespace std;
```

```
LayoutButtons::LayoutButtons() :
    m_Box_Top(Gtk::ORIENTATION_VERTICAL),
    m_Box1(Gtk::ORIENTATION_HORIZONTAL, 10),
    m_Box2(Gtk::ORIENTATION_HORIZONTAL, 10),
    m_Label_Salutation("Formule de Salutation"),
    m_Label_Destinataire("A qui s'adresse-t-on?"),
    m_Label_Fin("Les meilleures choses ont une fin\n"
               " éventuellement sur plusieurs lignes"),
    m_Button_Hello("Hello"),
    m_Button_Bonjour("Bonjour"),
    m_Button_Monsieur("Monsieur"),
    m_Button_Madame("Madame"),
    m_Button_Fin("Fin")
{
    // Set title and border of the window
    set_title("layout buttons");
    set_border_width(0);

    // Add outer box to the window (because the window
    // can only contain a single widget)
    add(m_Box_Top);

    //Put the inner boxes and the separator in the outer box:
    m_Box_Top.pack_start(m_Label_Salutation);
    m_Box_Top.pack_start(m_Box1);
    m_Box_Top.pack_start(m_Separator1);
    m_Box_Top.pack_start(m_Label_Destinataire);
    m_Box_Top.pack_start(m_Box2);
    m_Box_Top.pack_start(m_Separator2);
    m_Box_Top.pack_start(m_Label_Fin);
    m_Box_Top.pack_start(m_Button_Fin);
```

```

// Set the inner boxes' borders
m_Box2.set_border_width(10);
m_Box1.set_border_width(10);

// Put Hello / Bonjour buttons in Box1:
m_Box1.pack_start(m_Button_Hello);
m_Box1.pack_start(m_Button_Bonjour);

// Put Madame / Monsieur buttons in Box2:
m_Box2.pack_start(m_Button_Madame);
m_Box2.pack_start(m_Button_Monsieur);

// Connect the clicked signal of the button to
// thier signal handler
m_Button_Hello.signal_clicked().connect(sigc::mem_fun(*this,
    &LayoutButtons::on_button_clicked_Hello) );

m_Button_Bonjour.signal_clicked().connect(sigc::mem_fun(*this,
    &LayoutButtons::on_button_clicked_Bonjour) );

m_Button_Monsieur.signal_clicked().connect(sigc::mem_fun(*this,
    &LayoutButtons::on_button_clicked_Monsieur) );

m_Button_Madame.signal_clicked().connect(sigc::mem_fun(*this,
    &LayoutButtons::on_button_clicked_Madame) );

m_Button_Fin.signal_clicked().connect(sigc::mem_fun(*this,
    &LayoutButtons::on_button_clicked_Fin) );

// Show all children of the window
show_all_children();
}
LayoutButtons::~~LayoutButtons() {}

void LayoutButtons::on_button_clicked_Hello() {
    cout << "Hello " ;
}

void LayoutButtons::on_button_clicked_Bonjour() {
    cout << "Bonjour " ;
}

void LayoutButtons::on_button_clicked_Monsieur() {
    cout << "Monsieur" << endl;
}

void LayoutButtons::on_button_clicked_Madame() {
    cout << "Madame" << endl;
}

void LayoutButtons::on_button_clicked_Fin() {
    hide(); //to close the application.
}

```

**Exercice 3 (niveau 1) : Coordinates conversion from the Model space to the Application's window width and height / preventing distortion (in English)**

a) [see source code archive folder serie6\_corrige\_exo3a\_GTK\_conversion\_avec\_translate\_et\_scale]

For clarity purpose the solution defines a **struct Frame** that gathers both the Model space framing = [xMin,xMax] and [yMin,yMax], the window parameters **width** and **height** and the aspect-ratio width/height named **asp**.

A default variable of this type named **default\_frame** is used to initialize the added **frame** attribute to the MyArea class. Its value is set by the default constructor.

The static function **orthographic\_projection** is hidden within the class implementation as it is only defined to highlight its purpose of setting the conversion from the Model space to the window space (principle of abstraction). It is called within the overridden method **on\_draw()** after the **frame** attribute is update with the current window **width** and **height**. After the call to this function the drawing is done with coordinates expressed in the Model space.

When changing the window size, the conversion is updated BUT the scaling factor along X and Y might be different in absolute value, hence introducing a distortion (the lines may not be orthogonal anymore).

expected result with initial window size :



Distorsions introduced when changing the window size :



b) [see source code archive folder serie6\_corrige\_exo3b\_GTK\_conversion\_sans\_distorsion]

The difference with answer a) is the addition of the **adjustFrame** method that changes the Model framing so that :

- 1) its aspect ratio  $(xMax-xMin)/(yMax-yMin)$  becomes the same as the one the window : **width/height**,
- 2) the drawing is always centered in the window

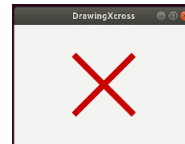
There are 2 strategies for adjustment depending on the value of the new window aspect-ratio **width/height**

3.1) if it is bigger or equal than the default one (i.e. the new window is more flat than the initial one) Then keep the default yMin and yMax and adjust xMin and xMax to get the same aspect ratio

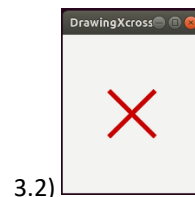
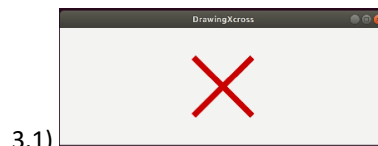
3.2) if it is smaller than the default one (i.e. the new window is less flat than the initial one) Then keep the default xMin and xMax and adjust yMin and yMax to get the same aspect ratio

The **adjustFrame** method has to be called at the beginning of **on\_draw()** to update the Model framing based on the new values of width and height. After the Model framing adjustment is done the conversion and the drawing calls are put in place as in the previous question.

expected result with initial window size :



NO MORE Distorsions introduced when changing the window size :



**Exercice 4 (niveau 1) : convertir le type paramétré Form de la Série4 en une hiérarchie de classes tirant parti du Polymorphisme [see source code archive folder hierarchie\_classe\_solution]**