ENSAI 3ème année
Filière Systèmes d'Information Statistique

Programmation Objet en C++

Thierry Duval

thierry.duval@irisa.fr

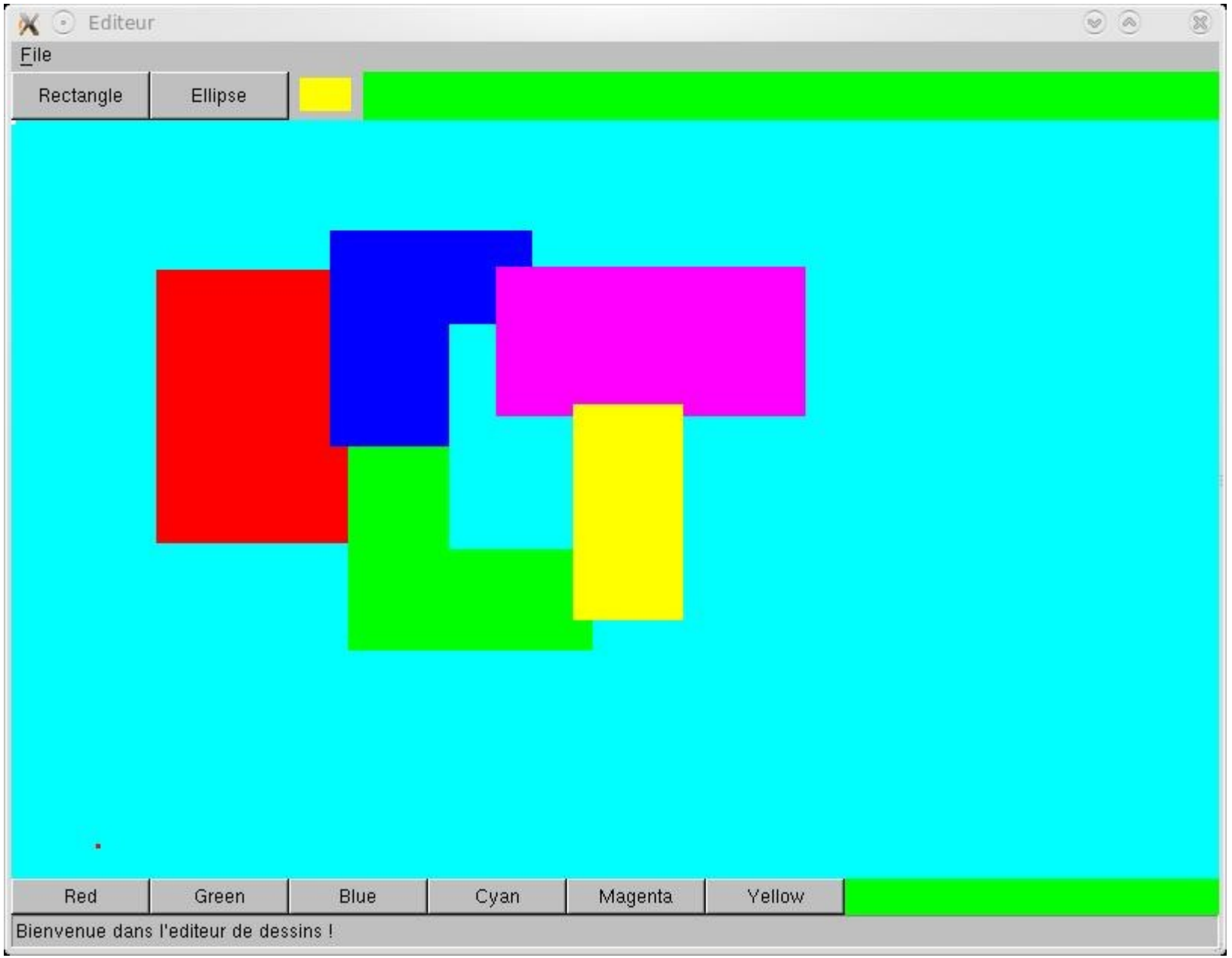
Objectifs du cours

- Approfondir les concepts objets en C++ :
  - Classes
  - Héritage (simple et multiple)
  - Polymorphisme et Liaison Dynamique
  - Généricité
- Découvrir quelques patrons de conception :
  - State, Observer, …
- Interagir autrement qu'avec une souris !
  - Enfin peut-être...

Moyens utilisés pour y parvenir...

- Réaliser un éditeur de dessins...
  - Utilisation de l'API graphique wxWidgets
    - En restant à un niveau d'utilisation assez simple..
  - Création de classes d'édition
    - En respectant certains patrons de conception
  - Création de classes de dessin
    - En profitant du polymorphisme et de la liaison dynamique
- Interfaçage avec un périphérique matériel...
  - Une « Wiimote »à coupler à l'éditeur
    - Via des interfaces (classes abstraites C++ à dériver)

L'éditeur de dessins



L'API wxWidget

- Va fournir les principales fonctionnalités attendues :
  - Affichage de zones rectangulaires et de widgets
    - Fenêtres, panneaux rectangulaires, boutons, …
    - À assembler et gérer dans une hiérarchie arborescente
  - Gestions des entrées de l'utilisateur :
    - Clics souris, déplacements souris, touches clavier
  - Dessins dans des zones rectangulaires
  - Interruptions régulières possibles :
    - Via une horloge qui peut envoyer des événements d'interruption
- Et beaucoup d'autres qu'on utilisera pas...

wxWidget

- http://www.wxwidgets.org/
- Boîte à outils graphique 2D
- « Cross-platform » toolkit :
  - Win32, Mac OS X, GTK+, X11, Motif, WinCE
  - API : C++, Python, Perl, C#/.NET
- « Look and feel » natif :
  - S'appuie sur la plate-forme native sous- jacente
- Extensible, Gratuite, Open-source, Mature

Les fonctionnalités de l'éditeur

- Choix de la forme à tracer
  - Via des boutons...
    - Offrir au moins 2 possibilités (rectangle, ellipse, …)
    - Permettre éventuellement de créer ses propres formes...
- Choix de la couleur de la forme à tracer
  - Via des boutons... ou des moyens plus élaborés (« sliders »)
- Dessin des formes dans des rectangles
  - Dans une zone réservée à cet effet
- Actions sur les dessins
  - Déplacement, suppression, « lancement », ...

Retours vers l'utilisateur

- Indispensables pour une bonne utilisation
  - Informent l'utilisateur de l'état courant
- À plusieurs niveaux :
  - Dessin « en temps interactif » lors de la création
    - Dans la zone de dessin...
  - Curseurs indiquant les actions à venir
    - Dans les dessins
  - Indicateurs de forme et couleur sélectionnées
    - Dans l'éditeur

Quelques éléments wxWidgets

- wxApp
- wxFrame
- wxWindow
- wxTopLevelWindow
- wxPanel
- wxSizer
- wxButton
- wxTimer
- wxPaintDC, wxPen, wx Brush, wxColour

wxApp

- Niveau principal
- Doit posséder une méthode :
  - virtual bool OnInit (void)
    - Chargée de créer au moins une fenêtre de plus haut niveau
    - … et de la déclarer comme fenêtre principale de l'application
- Main implémenté à l'aide d'une macro :
  - IMPLEMENT_APP (…)
  - Instanciera une classe dérivée de wxApp
  - Appellera la méthode OnInit de cette instance

wxFrame

- Hérite de wxTopLevelWindow
  - Est donc une fenêtre sans « parent »
  - Est manipulable par le gestionnaire de fenêtrage
- Peut accueillir des composants graphiques
  - Un seul « enfant » occupera tout l'espace
  - Plusieurs enfants devront se partager l'espace
    - Utiliser pour cela un gestionnaire de positionnement (wxSizer)

wxWindow

- Paramètres de son constructeur :
  - wxWindow * parent, // NULL si pas de parent
  - wxWindowID id, // wx_ID_ANY : donné automatiquement, < 0
  - const wxPoint & pos = wxDefaultPosition,
  - const wxSize & size = wxDefaultSize,
  - long style = 0,
  - const wxString & name = wxPanelNameStr
- Donne accès à son « parent » et à ses « enfants » :
  - virtual wxWindow * GetParent (void) const
  - const wxWindowList & GetChildren () const // + non const...

Exemple : wxApp et wxFrame

Exemple : wxApp et wxFrame

```
#include <wx/wx.h>

class MainFrame : public wxApp {
   virtual bool OnInit (void) ;
} ;

bool MainFrame::OnInit (void) {
   wxFrame * frame = new wxFrame ((wxFrame *)NULL, -1, _T ("wxApp et wxFrame"),
                                                  wxPoint (50,50), wxSize (800,600)) ;
   frame->Show (TRUE) ;
   SetTopWindow (frame) ;
   return TRUE ;
}

IMPLEMENT_APP (MainFrame)
```

wxPanel

- Hérite de wxWindow
  - Est une fenêtre contenue dans n'importe quelle autre
  - Est aussi un conteneur d'autres fenêtres
  - Est un élément très utilisé car facile à étendre par dérivation

Exemple : wxApp, wxFrame et wxPanel

Exemple : wxApp, wxFrame et wxPanel

```
#include <wx/wx.h>

class MainPanel : public wxApp {
   virtual bool OnInit (void) ;
} ;

bool MainPanel::OnInit (void) {
   wxFrame * frame = new wxFrame ((wxFrame *)NULL, -1,
                                  _T ("wxApp, wxFrame et wxPanel"),
                                  wxPoint (50,50), wxSize (800,600)) ;
   wxPanel * rootPanel = new wxPanel (frame, -1) ;
   rootPanel->SetBackgroundColour (wxColor ("White")) ;
   frame->Show (TRUE) ;
   SetTopWindow (frame) ;
   return TRUE ;
}

IMPLEMENT_APP (MainPanel)
```

Organisation de l'arborescence

- Automatiquement à la création d'un composant !
  - En précisant quel est son parent :
    - NULL pour une fenêtre de plus haut niveau
    - Une fenêtre pour tous les autres composants graphiques
- Réorganisation possible :
  - En précisant quel est le nouveau parent :
    - virtual bool wxWindow::Reparent (wxWindow * newParent)
  - En détruisant une fenêtre :
    - virtual bool wxWindow::Destroy()

Quelques détails techniques

- A priori, wxWidgets n'est pas totalement intégré à l'environnement de développement...
  - N'importe quel utilisateur peut l'installer pour son usage personnel, sans privilèges particuliers
- Il va falloir déclarer où se trouvent :
  - Les fichiers d'entête décrivant les composants
    - Pour pouvoir compiler les programmes
  - Les bibliothèques de composant avec lesquelles se lier
    - Pour pouvoir créer des programmes exécutables
  - Les bibliothèques à charger dynamiquement
    - Pour pouvoir exécuter les programmes créés

Les variables d'environnement

- En environnement de type « C shell»

setenv WXWIDGETSDIR /là_où_est_installé_wxwidgets
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${WXWIDGETSDIR}/lib

- En environnement de type « bourne again shell»

export WXWIDGETSDIR=/là_où_est_installé_wxwidgets
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${WXWIDGETSDIR}/lib

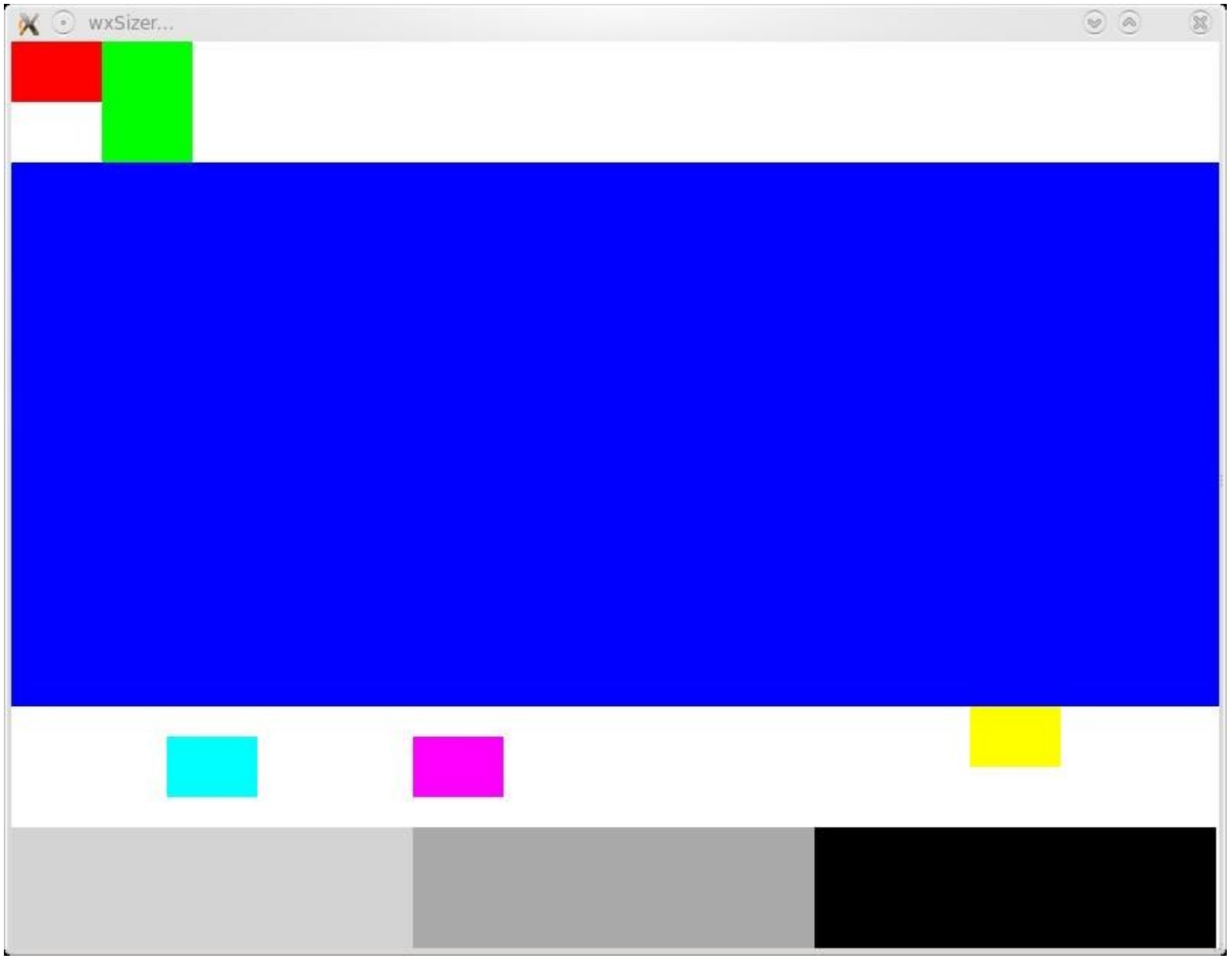
Le Makefile

```
C_FLAGS = -g -c `$(WXWIDGETSDIR)/bin/wx-config --cflags`
L_FLAGS = -g `$(WXWIDGETSDIR)/bin/wx-config --libs`
GCC = g++

all : exempleFrame exemplePanel

exempleFrame : exempleFrame.o
	$(GCC) -o exempleFrame $(L_FLAGS) exempleFrame.o

exempleFrame.o : exempleFrame.cxx
	$(GCC) $(C_FLAGS) exempleFrame.cxx

exemplePanel : exemplePanel.o
	$(GCC) -o exemplePanel $(L_FLAGS) exemplePanel.o

exemplePanel.o : exemplePanel.cxx
	$(GCC) $(C_FLAGS) exemplePanel.cxx
```

wxSizer

- Gestionnaire de positionnement
- Peut contenir :
  - Un autre wxSizer
  - Un composant graphique
- Associer le wxSizer principal à une fenêtre
- Plusieurs spécialisations, dont :
  - wxBoxSizer : alignement en lignes et colonnes
  - wxGridSizer : organisation en grille (rigide)
  - wxFlexGridSizer : organisation en grille (semi-souple)
  - wxGridBagSizer : organisation en grille (souple)

Exemple : wxSizer



Exemple : wxSizer (1/5)

```
#include <wx/wx.h>

class MainSizer : public wxApp {
   virtual bool OnInit (void) ;
} ;

bool MainSizer::OnInit (void) {
   wxFrame * frame = new wxFrame ((wxFrame *)NULL, -1, _T ("wxSizer..."),
                                                  wxPoint (50,50), wxSize (800,600)) ;
   wxPanel * rootPanel = new wxPanel (frame, -1) ;
   rootPanel->SetBackgroundColour (wxColor ("White")) ;
```

Exemple : wxSizer (2/5)

```
wxBoxSizer * hbox1 = new wxBoxSizer (wxHORIZONTAL) ;

wxPanel * panel_1_1 = new wxPanel (rootPanel, -1) ;
panel_1_1->SetClientSize (60, 40) ;
panel_1_1->SetBackgroundColour (wxColor ("Red")) ;
hbox1->Add (panel_1_1, 0) ; // 0 : taille fixe

wxPanel * panel_1_2 = new wxPanel (rootPanel, -1) ;
panel_1_2->SetBackgroundColour (wxColor ("Green")) ;
hbox1->Add (panel_1_2, 1, wxEXPAND) ; // 1 : taille automatique

wxBoxSizer * hbox2 = new wxBoxSizer (wxHORIZONTAL) ;

wxPanel * panel_2 = new wxPanel (rootPanel, -1) ;
panel_2->SetBackgroundColour (wxColor ("Blue")) ;
panel_2->SetClientSize (60, 40) ; // pas pris en compte ici
hbox2->Add (panel_2, 1, wxEXPAND) ; // 1 : taille automatique
```

Exemple : wxSizer (3/5)

```
wxGridSizer * grid = new wxGridSizer (3) ; // 3 colonnes

wxPanel * panel_3_1 = new wxPanel (rootPanel, -1) ;
panel_3_1->SetBackgroundColour (wxColor ("Cyan")) ;
panel_3_1->SetClientSize (60, 40) ;
grid->Add (panel_3_1, 0, wxALIGN_CENTER_HORIZONTAL |
                         wxALIGN_CENTER_VERTICAL) ;

wxPanel * panel_3_2 = new wxPanel (rootPanel, -1) ;
panel_3_2->SetBackgroundColour (wxColor ("Magenta")) ;
panel_3_2->SetClientSize (60, 40) ;
grid->Add (panel_3_2, 0, wxALIGN_CENTER_VERTICAL) ;

wxPanel * panel_3_3 = new wxPanel (rootPanel, -1) ;
panel_3_3->SetBackgroundColour (wxColor ("Yellow")) ;
panel_3_3->SetClientSize (60, 40) ;
grid->Add (panel_3_3, 0, wxALIGN_CENTER_HORIZONTAL) ;
```

Exemple : wxSizer (4/5)

```
wxPanel * panel_3_4 = new wxPanel (rootPanel, -1) ;
panel_3_4->SetBackgroundColour (wxColor ("LightGrey")) ;
panel_3_4->SetClientSize (60, 40) ;
grid->Add (panel_3_4, 1, wxEXPAND) ;

wxPanel * panel_3_5 = new wxPanel (rootPanel, -1) ;
panel_3_5->SetBackgroundColour (wxColor ("DarkGrey")) ;
panel_3_5->SetClientSize (60, 60) ;
grid->Add (panel_3_5, 1, wxEXPAND) ;

wxPanel * panel_3_6 = new wxPanel (rootPanel, -1) ;
panel_3_6->SetBackgroundColour (wxColor ("Black")) ;
panel_3_6->SetClientSize (60, 80) ;
grid->Add (panel_3_6, 1, wxEXPAND) ;

wxBoxSizer * hbox3 = new wxBoxSizer (wxHORIZONTAL) ;

hbox3->Add (grid, 1 , wxEXPAND) ;
```

Exemple : wxSizer (5/5)

```
  wxBoxSizer * vbox = new wxBoxSizer (wxVERTICAL) ;

  vbox->Add (hbox1, 0) ; // 0 : taille fixe
  vbox->Add (hbox2, 1, wxEXPAND) ; // 1 : taille automatique
  vbox->Add (hbox3, 0, wxEXPAND) ; // 0 : taille fixe

  rootPanel->SetSizer (vbox) ;

  frame->Show (TRUE) ;
  SetTopWindow (frame) ;
  return TRUE ;
}

IMPLEMENT_APP (MainSizer)
```

wxButton

- Composant de « contrôle »
- Permet de capter des événements graphiques
  - Souris
  - Clavier
- Se place dans n'importe quel type de fenêtre
- Se gère donc graphiquement à l'aide d'un wxSizer
- Peut être activé ou désactivé :
  - virtual bool Enable (bool enable = true)
  - bool Disable () // <=> Enable (false)

Exemple : wxButton (1/2)

```
#include <wx/wx.h>

class MainButton : public wxApp {
   virtual bool OnInit (void) ;
} ;

IMPLEMENT_APP (MainButton)
```

Exemple : wxButton (1/2)

```
bool MainButton::OnInit (void) {
   wxFrame * frame = new wxFrame ((wxFrame *)NULL, -1, _T ("wxButton..."),
                                                  wxPoint (50,50), wxSize (200,100)) ;
   wxPanel * rootPanel = new wxPanel (frame, -1) ;
   rootPanel->SetBackgroundColour (wxColor ("White")) ;
   wxBoxSizer * hbox = new wxBoxSizer (wxHORIZONTAL) ;

   wxButton * rectangle = new wxButton (rootPanel, -1, wxT ("Rectangle")) ;
   wxButton * ellipse = new wxButton (rootPanel, -1, wxT ("Ellipse")) ;

   hbox->Add (rectangle, 0) ;
   hbox->Add (ellipse, 0) ;
   rootPanel->SetSizer (hbox) ;

   frame->Show (TRUE) ;
   SetTopWindow (frame) ;
   return TRUE ;
}
```

La gestion des événements

- Possible pour les composants dits de contrôle
- Connexion de l'apparition d'un type d'événement avec une méthode à déclencher (notion de callback) :
  - void Connect (
    - int id,
    - wxEventType eventType,
    - wxObjectEventFunction function,
    - wxObject * userData = NULL,
    - wxEvtHandler * eventSink = NULL)
  - Idem sans le premier argument

La fin de gestion des événements

- Possibilité de déconnexion de la méthode à déclencher sur l'apparition d'un type d'événement :
  - bool Disconnect (
    - int id = wxID_ANY,
    - wxEventType eventType = wxEVT_NULL,
    - wxObjectEventFunction function = NULL,
    - wxObject * userData = NULL,
    - wxEvtHandler * eventSink = NULL)
  - Idem sans le premier argument

Types d'événements (1/4)

- Définis dans wx/event.h
- Quelques événements de déclenchement :
  - wxEVT_COMMAND_BUTTON_CLICKED
  - wxEVT_COMMAND_CHECKBOX_CLICKED
  - wxEVT_COMMAND_CHOICE_SELECTED
  - wxEVT_COMMAND_LEFT_CLICK
  - ...
- Événement lié au temps :
  - wxEVT_TIMER

Types d'événements (2/4)

- Quelques événements souris :
  - wxEVT_LEFT_DOWN, wxEVT_LEFT_UP
  - wxEVT_MIDDLE_DOWN, wxEVT_MIDDLE_UP
  - wxEVT_RIGHT_DOWN, wxEVT_RIGHT_UP
  - wxEVT_MOTION
  - wxEVT_ENTER_WINDOW, wxEVT_LEAVE_WINDOW
  - wxEVT_MOUSEWHEEL
  - wxEVT_LEFT_DCLICK
  - ...

Types d'événements (3/4)

- Quelques types d'événements clavier :
  - wxEVT_CHAR
  - wxEVT_CHAR_HOOK
  - wxEVT_NAVIGATION_KEY
  - wxEVT_KEY_DOWN
  - wxEVT_KEY_UP
  - wxEVT_SET_CURSOR
  - ...

Types d'événements (4/4)

- Quelques types d'événements système :
  - wxEVT_SIZE
  - wxEVT_MOVE
  - wxEVT_CLOSE_WINDOW
  - wxEVT_PAINT
  - wxEVT_ERASE_BACKGROUND
  - ...

Exemple d'événement sur des boutons (1/3)

```
#include <wx/wx.h>

class MainButton : public wxApp {
   virtual bool OnInit (void) ;
   virtual void OnRectangle (wxCommandEvent & event) ;
   virtual void OnEllipse (wxCommandEvent & event) ;
} ;

IMPLEMENT_APP (MainButton)
```

Exemple d'événement sur des boutons (2/3)

```
bool MainButton::OnInit (void) {
   wxFrame * frame = new wxFrame ((wxFrame *)NULL, -1,
                                  _T ("wxButton..."), wxPoint (50,50), wxSize (200,100)) ;
   wxPanel * rootPanel = new wxPanel (frame, -1) ;
   rootPanel->SetBackgroundColour (wxColor ("White")) ;

   wxBoxSizer * hbox = new wxBoxSizer (wxHORIZONTAL) ;

   wxButton * rectangle = new wxButton (rootPanel, -1, wxT ("Rectangle")) ; // id attribué...
   wxButton * ellipse = new wxButton (rootPanel, -1, wxT ("Ellipse")) ; // id attribué...

   hbox->Add (rectangle, 0) ;
   hbox->Add (ellipse, 0) ;

   rootPanel->SetSizer (hbox) ;
```

Exemple d'événement sur des boutons (3/3)

```
   Connect (rectangle->GetId (), wxEVT_COMMAND_BUTTON_CLICKED,
                (wxObjectEventFunction) &MainButton::OnRectangle) ;
   Connect (ellipse->GetId (), wxEVT_COMMAND_BUTTON_CLICKED,
                (wxObjectEventFunction) &MainButton::OnEllipse) ;

   frame->Show (TRUE) ;
   SetTopWindow (frame) ;
   return TRUE ;
}

void MainButton::OnRectangle (wxCommandEvent& event) {
   std::cout << "MainButton : OnRectangle" << std::endl ;
}

void MainButton::OnEllipse (wxCommandEvent & event) {
   std::cout << "MainButton : OnEllipse" << std::endl ;
}
```

L'interruption par une horloge

- Possibilité de :
  - Créer une horloge
  - La démarrer
  - Associer une callback à la production de chaque tic d'horloge
- Permet d'obtenir des informations en provenance du monde extérieur !
  - Sinon on ne peut capter que des actions de l'utilisateur puisqu'on ne maitrise pas le main et sa boucle de gestion d'événements
  - De plus, les interruptions extérieures intempestives peuvent provoquer des erreurs d'exécution...

wxTimer

- Une horloge associée à un composant
- Quelques méthodes :
  - wxTimer (wxEvtHandler *owner, int id = -1)
  - bool Start (int milliseconds = -1, bool oneShot = false)
  - void Stop (void)

Exemple d'interruption par une horloge (1/4)

```
#include <wx/wx.h>

class MainTimer : public wxApp {
   virtual bool OnInit (void) ;
   virtual void OnTimer (void) ;
   virtual void OnStart (wxCommandEvent & event) ;
   virtual void OnStop (wxCommandEvent & event) ;
   wxTimer * timer ;
   wxButton * start ;
   wxButton * stop ;
} ;

IMPLEMENT_APP (MainTimer)
```

Exemple d'interruption par une horloge (2/4)

```
bool MainTimer::OnInit (void) {
   wxFrame * frame = new wxFrame ((wxFrame *)NULL, -1,
                              _T ("wxTimer..."), wxPoint (50,50), wxSize (200,100)) ;
   wxPanel * rootPanel = new wxPanel (frame, -1) ;
   rootPanel->SetBackgroundColour (wxColor ("White")) ;

   start = new wxButton (rootPanel, -1, wxT ("Start timer")) ;
   stop = new wxButton (rootPanel, -1, wxT ("Stop timer")) ;

   wxBoxSizer * hbox = new wxBoxSizer (wxHORIZONTAL) ;
   hbox->Add (start, 0) ;
   hbox->Add (stop, 0) ;
   rootPanel->SetSizer (hbox) ;

   Connect (start->GetId (), wxEVT_COMMAND_BUTTON_CLICKED,
            (wxObjectEventFunction) &MainTimer::OnStart) ;
   Connect (stop->GetId (), wxEVT_COMMAND_BUTTON_CLICKED,
            (wxObjectEventFunction) &MainTimer::OnStop) ;
```

Exemple d'interruption par une horloge (3/4)

```
   start->Enable (true) ;
   stop->Enable (false) ;

   timer = new wxTimer (this, 1) ;
   Connect (wxEVT_TIMER, (wxObjectEventFunction) &MainTimer::OnTimer) ;

   frame->Show (TRUE) ;
   SetTopWindow (frame) ;
   return TRUE ;
}

void MainTimer:OnTimer (void) {
   std::cout << "MainTimer : OnTimer" << std::endl ;
}
```

Exemple d'interruption par une horloge (4/4)

```
void MainTimer::OnStart (wxCommandEvent& event) {
   timer->Start (200) ;
   start->Disable () ;
   stop->Enable () ;
}

void MainTimer::OnStop (wxCommandEvent & event) {
   timer->Stop () ;
   start->Enable (true) ;
   stop->Enable (false) ;
}
```

Dessin dans un wxPanel

- À l'aide d'un wxPaintDC
- Quelques méthodes :
  - wxPaintDC (wxWindox * window) ;
  - void SetPen (const wxPen & pen)
  - void SetBrush (const wxBrush & brush)
  - void DrawRectangle (wxCoord x, wxCoord y, wxCoord width, wxCoord height)
  - void DrawEllipse (wxCoord x, wxCoord y, wxCoord width, wxCoord height)

wxBrush, wxPen, wxColour

- Brosse (dessein plein) :
  - wxBrush (const wxColour & colour, int style = wxSOLID)
- Crayon (dessin au trait) :
  - wxPen (const wxColour& c, int width = 1, int style = wxSOLID)
- Couleur :
  - wxColour (unsigned char r, unsigned char g, unsigned char b, unsigned char alpha = wxALPHA_OPAQUE)
  - wxColour (const wxString& colourNname)

Exemple de dessin dans un wxPanel (1/3)

```
#include <wx/wx.h>

class Dessin : public wxPanel {
   public :
      Dessin (wxWindow * parent, int id) ;
      virtual void OnSize (wxSizeEvent & event) ;
      virtual void OnPaint (wxPaintEvent & event) ;
} ;

Dessin::Dessin (wxWindow * parent, int id)
      : wxPanel (parent, id) {
   Connect (wxEVT_PAINT, wxPaintEventHandler (Dessin::OnPaint)) ;
   Connect (wxEVT_SIZE, wxSizeEventHandler (Dessin::OnSize)) ;
   SetBackgroundColour (wxColor ("Yellow")) ;
   SetForegroundColour (wxColor ("Red")) ;
}
```

Exemple de dessin dans un wxPanel (2/3)

```
void Dessin::OnSize (wxSizeEvent & event) {
   std::cout << "Dessin::OnPaint" << std::endl ;
   Refresh () ;
}

void Dessin::OnPaint (wxPaintEvent & event) {
   std::cout << "Dessin::OnPaint" << std::endl ;
   wxPaintDC dc (this) ;
   dc.SetPen(wxPen (GetForegroundColour (), 1)) ;
   dc.SetBrush(wxBrush (GetForegroundColour ()));
   dc.DrawRectangle (20, 20, GetSize ().GetWidth () - 40, GetSize ().GetHeight () - 40) ;
}
```

Exemple de dessin dans un wxPanel (3/3)

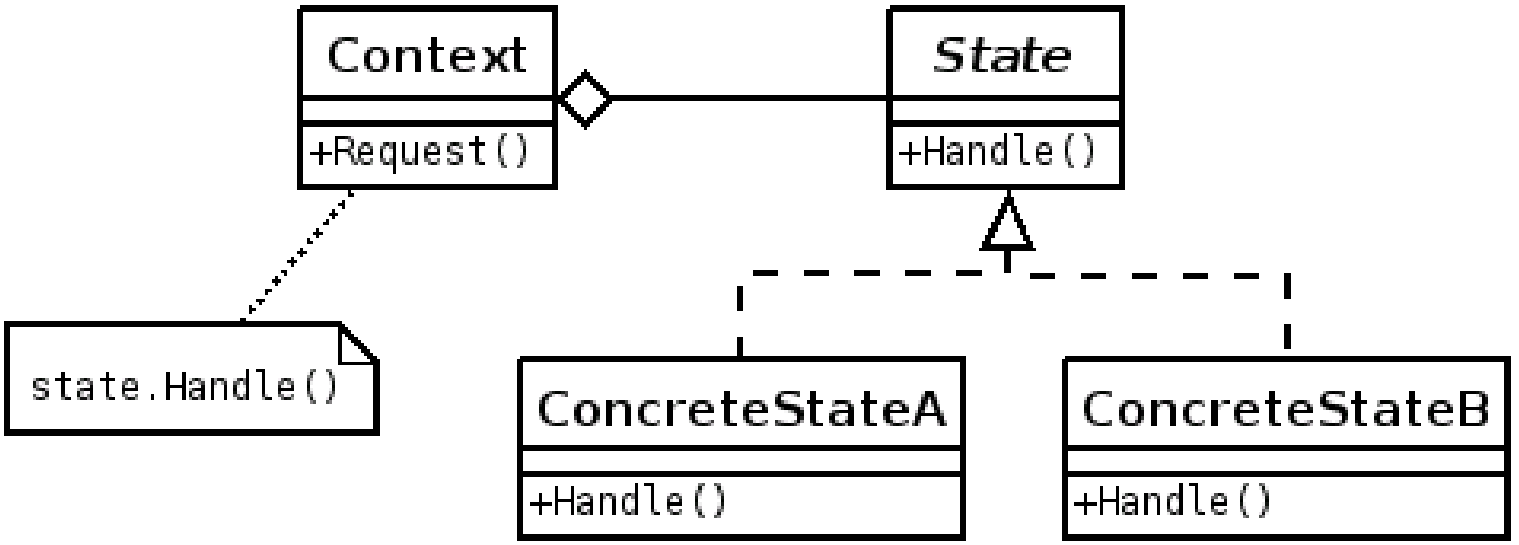
```
class MainDessin : public wxApp {
   virtual bool OnInit (void) ;
} ;

bool MainDessin::OnInit (void) {
   wxFrame * frame = new wxFrame ((wxFrame *)NULL, -1,
                                  _T ("Dessin..."), wxPoint (50,50), wxSize (200,100)) ;
   wxPanel * rootPanel = new Dessin (frame, -1) ;

   frame->Show (true) ;
   SetTopWindow (frame) ;
   return (true) ;
}

IMPLEMENT_APP (MainDessin)
```

Les patrons de conception...

- Les principaux sont décrits dans le GoF :
  - Design Patterns: Elements of Reusable Object-Oriented Software, (Addison-Wesley Professional Computing Series)
    - Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides
- Autres ressources à propos des patrons de conception :
  - http://www.vincehuston.org/dp/
  - http://www.dofactory.com/Patterns/Patterns.aspx
  - http://www.mindspring.com/~mgrand/pattern_synopses.htm
- Et à propos du C++ :
  - http://www.cppreference.com/
  - http://www.cplusplus.com/ref/

Le « State » (GoF 305)

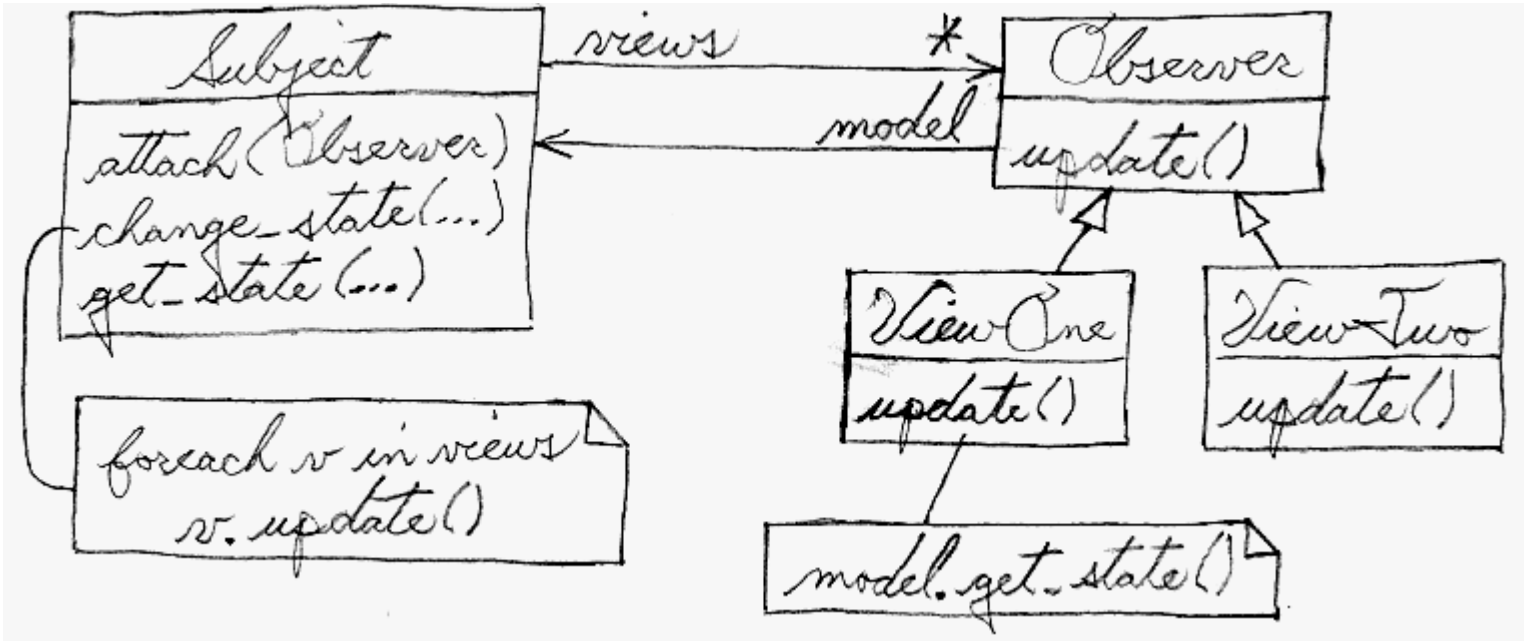
- Basé sur :
  - L'héritage
  - Le polymorphisme
  - La liaison dynamique



L'éditeur et le « State »

- Context :
  - La zone de dessin
- State :
  - Le créateur de dessins (une classe abstraite !)
  - Handle (virtuelle pure : « = 0 ») :
    - c'est la méthode de création de dessins
- ConcreteStateA :
  - Le créateur de rectangles
- ConcreteStateB :
  - Le créateur d'ellipses

L' « Observer » (GoF 293)



L'éditeur et l'« observer »

- Subject : la zone de dessin
  - Sa commande de création de dessin et sa couleur de dessin
    - Et pourquoi pas sa couleur de fond, le nombre de dessins, ...
  - Doit donc disposer d'attributs et méthodes permettant :
    - D'enregistrer des observateurs
    - De propager ses changements aux observateurs
    - De mettre à disposition les valeurs obervées
- Observer :
  - Un nouveau widget
  - Ajouté quelque part dans l'éditeur
  - Doté d'une méthode permettant à l'observé de le prévenir

Autre implémentation possible

- Sur le modèle des « PropertyChangeListener » java :
  - Méthode update paramétrée par :
    - Le nom de la propriété qui a changée
    - La nouvelle valeur de cette propriété (void *)
  - Dans update, on trouve alors un « switch »
  - Ou alors autant de méthodes update que de propriétés...

Wiimote et éditeur de dessins

- Proposition d'utilisation d'une Wiimote pour piloter l'éditeur de dessins :
  - Besoin d'utiliser un driver : cwiid
  - Besoin d'un support bluetooth sur les machines
- Utilisation d'une callback static de cwiid :
  - À l'aide de « cwiid_set_mesg_callback »
  - Ne devra pas interrompre la boucle graphique de d'éditeur
    - Utilisation d'une méthode d'activation par horloge pour aller regarder les données récupérées par la callback
- Classe WiimoteDriver fournie
  - S'appuie sur des classes abstraites à dériver

Les classes compatibles WiimoteDriver

- LimitedWiimoteButtonUser
  - Boutons A et B, enfoncement et relâchement
- CompleteWiimoteButtonUser
  - Tous les boutons, enfoncement et relâchement
- WimmoteXYUser
  - X et Y acquis par la caméra infrarouge
- WiimoteAccelerationUser
  - Ax, Ay et Az acquis par l'accéléromètre
- CompleteWiimoteUser
  - L'ensemble...

LimitedWiimoteButtonUser

```
class LimitedWiimoteButtonUser {

   public :

      virtual void OnWiimoteAPressed (void) = 0 ;
      virtual void OnWiimoteAReleased (void) = 0 ;
      virtual void OnWiimoteBPressed (void) = 0 ;
      virtual void OnWiimoteBReleased (void) = 0 ;

} ;
```

CompleteWiimoteButtonUser

```
class CompleteWiimoteButtonUser : public LimitedWiimoteButtonUser {
   public :
      virtual void OnWiimote1Pressed (void) = 0 ; virtual void OnWiimote2Pressed (void) = 0 ;
      virtual void OnWiimoteUpPressed (void) = 0 ;
      virtual void OnWiimoteDownPressed (void) = 0 ;
      virtual void OnWiimoteRightPressed (void) = 0 ;
      virtual void OnWiimoteLeftPressed (void) = 0 ;
      virtual void OnWiimotePlusPressed (void) = 0 ;
      virtual void OnWiimoteMinusPressed (void) = 0 ;
      virtual void OnWiimoteHomePressed (void) = 0 ;
      virtual void OnWiimote1Released (void) = 0 ; virtual void OnWiimote2Released (void) = 0 ;
      virtual void OnWiimoteUpReleased (void) = 0 ;
      virtual void OnWiimoteDownReleased (void) = 0 ;
      virtual void OnWiimoteRightReleased (void) = 0 ;
      virtual void OnWiimoteLeftReleased (void) = 0 ;
      virtual void OnWiimotePlusReleased (void) = 0 ;
      virtual void OnWiimoteMinusReleased (void) = 0 ;
      virtual void OnWiimoteHomeReleased (void) = 0 ; } ;
```

WimmoteXYUser

```
class WiimoteXYUser {

   public :

      virtual void OnWiimoteXYChanged (const float x, const float y) = 0 ;

} ;
```

WiimoteAccelerationUser

```
class WiimoteAccelerationUser {

   public :

      virtual void OnWiimoteAccelerationChanged (const float ax,
                                                 const float ay,
                                                 const float az) = 0 ;

} ;
```

CompleteWiimoteUser

```
class CompleteWiimoteUser : public CompleteWiimoteButtonUser,
                            public WiimoteXYUser,
                            public WiimoteAccelerationUser {

} ;
```

WiimoteDriver (1/3)

```
class WiimoteDriver {

   public :

      WiimoteDriver (CompleteWiimoteUser * cwu, const std::string wiimoteAddress) ;
      ~WiimoteDriver (void) ;
      void compute (void) ;

   protected :

      CompleteWiimoteUser * cwu ;

      std::list<std::string> producedEvents ;
      std::map<const std::string, void (CompleteWiimoteUser::*) (void)> correspondances ;

      …

} ;
```

WiimoteDriver (2/3)

```
WiimoteDriver::WiimoteDriver (CompleteWiimoteUser * wu, const std::string wiimoteAddress)
                    : cwu (wu),
                    … {

  correspondances ["wiimote_button_a_pressed"] =
    &CompleteWiimoteUser::OnWiimoteAPressed ;
  correspondances ["wiimote_button_a_released"] =
    &CompleteWiimoteUser::OnWiimoteAReleased ;
  correspondances ["wiimote_button_b_pressed"] =
    &CompleteWiimoteUser::OnWiimoteBPressed ;
  ...

}
```

WiimoteDriver (3/3)

```
void WiimoteDriver::compute (void) {
   ...
   cwu->OnWiimoteXYChanged (1 - _xWMNormalized, _yWMNormalized) ;
   cwu->OnWiimoteAccelerationChanged (_aWM [0], _aWM [1], _aWM [2]) ;

   std::list<std::string>::iterator ite ;
   std::list<std::string>::iterator end = producedEvents.end () ;
   for (ite = producedEvents.begin () ; ite != end ; ite ++) {
      if (correspondances.find (*ite) != correspondances.end ()) {
         (cwu->*(correspondances [*ite])) () ;
      }
   }
   producedEvents.clear () ;
}
```

Utilisation de la STL

- Standard Template Library
- Pour manipuler les enfants d'une fenêtre :
  - list<wxWindow *> → wxWindowList
- Utilisation des itérateurs pour :
  - Parcourir l'ensemble des éléments
  - Invoquer des méthodes sur ces éléments
- Les itérateurs pourront être const :
  - Pour invoquer des méthodes qui le sont aussi

wxWindowList

```
void printWindowsPositions (const wxWindowList & children) {
   wxWindowList::const_iterator ite, end = children.end () ;
   wxWindow * current ;
   for (ite = children.begin () ; ite !=end ; ite ++) {
      current = *ite ;
      std::cout <<current->GetPosition ().x << " " << current->GetPosition ().y << std::endl ;
   }
}

void printWindowsSizes (const wxWindowList & children) {
   wxWindowList::const_reverse_iterator ite, end = children.rend () ;
   wxWindow * current ;
   for (ite = children.rbegin () ; ite !=end ; ite ++) {
      current = *ite ;
      std::cout <<current->GetSize ().GetWidth () << " "
                << current->GetSize ().GetHeight () << std::endl ;
   }
}
```

Espaces de nommage

- Utilisation classique :
  - std::cout << "TP C++ - Éditeur de dessins" << std::endl ;
- Déclaration d'utilisation de l'espace de nommage :
  - using namespace std ;
- Utilisation après déclaration :
  - cout << "TP C++ - Éditeur de dessins" << endl ;
- Possibilité de créer ses propres espaces :
  - Déclarer des classes dans un espace de nommage
  - Déclarer les méthodes dans cet espace de nommage ou préfixer les méthodes par cet espace de nommage

Espaces de nommage (1/2)

```
#include <wx/wx.h>

namespace WF {

   class WindowFinder {
      public :
         static wxWindow * findWindowAt (...) ;
         static wxWindow * reverseFindWindowAt (...) ;
   } ;

} ;
```

Espaces de nommage (2/2)

```
wxWindow * WF::WindowFinder::findWindowAt (...) {
   …
}

namespace WF {

   wxWindow * WindowFinder::reverseFindWindowAt (...) {
      …
   }

}
```