

LIFAP4 - CM6

Interface Graphique "GUI" Graphical User Interface

- Boucle d'événements : l'app Pacman sert de fil conducteur
 - Programmation Évènementielle
- Bibliothèques "IHM"
 - Texte (~Ncurses)
 - SDL2 (graphique, image) / 2 mots sur SFML
 - imGUI
 - Qt (menu, boîte dialogue, etc.)

<http://licence-info.univ-lyon1.fr/LIFAP4>

Alexandre Meyer



Boucle d'événement
menu

Menu

- En texte, pour gérer un menu (tp de LIFAP1)
 - Afficher le menu
 - Poser la question à l'utilisateur
 - Traiter requête utilisateur
 - Appeler la fonction selon ...
 - On recommence

```
Select a Language
```

```
0. English
1. French
2. German
3. Italian
4. Japanese
5. Korean
6. Simplified Chinese
7. Spanish
8. Swedish
9. Traditional Chinese
```

```
Please make a choice (0 - 9), or press h or ? for help:
```

Menu

- En texte, pour gérer un menu

```
// Module Menu.h/.cpp
enum ChoixMenu { MENU_CH1=0, MENU_CH2=1, etc. };
void menuRun()
{
    ChoixMenu ch;
    do
    {
        menuAff();
        scanf("%d", &ch); // ca marche parce que enum=entier
        switch(ch)
        {
            case MENU_CH1 : fonction1(); break;
            case MENU_CH2 : fonction2(); break;
            case MENU_QUIT : libereMemoire(); break;
            default: erreurAff(); break;
        }
        resultatAff();
    } while( ch!= MENU_QUIT );
}
```

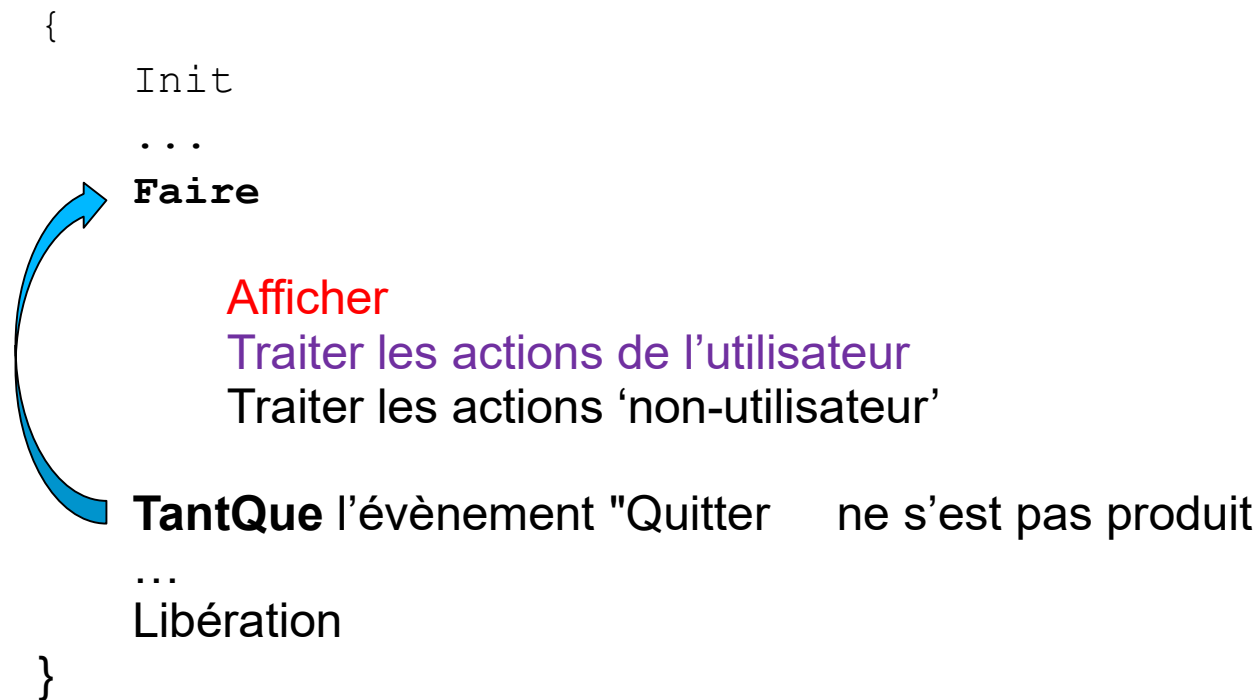
Menu

- En texte, pour gérer un menu
 - Boucle = **boucle de gestion des événements**
 - On la retrouve dans toutes les applications

```
// Module Menu.h/.cpp
enum ChoixMenu { MENU_CH1=0, MENU_CH2=1, etc. };
void menuRun()
{
    ChoixMenu ch;
    do
    {
        menuAff();
        scanf("%d", &ch); // ca marche parce que enum=entier
        switch(ch)
        {
            case MENU_CH1 : fonction1(); break;
            case MENU_CH2 : fonction2(); break;
            case MENU_QUIT : libereMemoire(); break;
            default: erreurAff(); break;
        }
        resultatAff();
    } while( ch!= MENU_QUIT );
}
```

Menu


- En texte, pour gérer un menu
 - Boucle = **boucle de gestion des événements**
 - On la retrouve dans toutes les applications



Menu

- En texte, pour gérer un menu
 - Boucle = **boucle de gestion des événements**
 - On la retrouve dans toutes les applications

```
{  
    Init  
    ...  
    Faire  
    Afficher  
  
    Tant qu'il existe un évènement non traité Faire  
        Récupérer l'évènement  
        Traiter l'évènement  
    FinTantQue  
  
    Traiter les actions indépendant de l'utilisateur  
  
    TantQue l'évènement "Quitter" ne s'est pas produit  
    ...  
    Libération  
}
```



Menu

- On parle de **programmation événementielle**
 - réaction à un événement
- Utilisation de lib
 - parfois la boucle d'évènements est cachée dans la librairie
par exemple : glutLoop (pour la lib GLUT)
 - On programme alors en donnant les fonctions *fonctionN()* à la lib --> notion de **callback**

Cf. Qt plus loin

Menu

Comment écrire une librairie réutilisable de gestion de menu ?

➔ Voir le code menu2 (.h/.cpp)

- Menu = tableau de lignes
- Ligne = un texte + une fonction d'action

➔ Notion de callback = **pointeur de fonction**

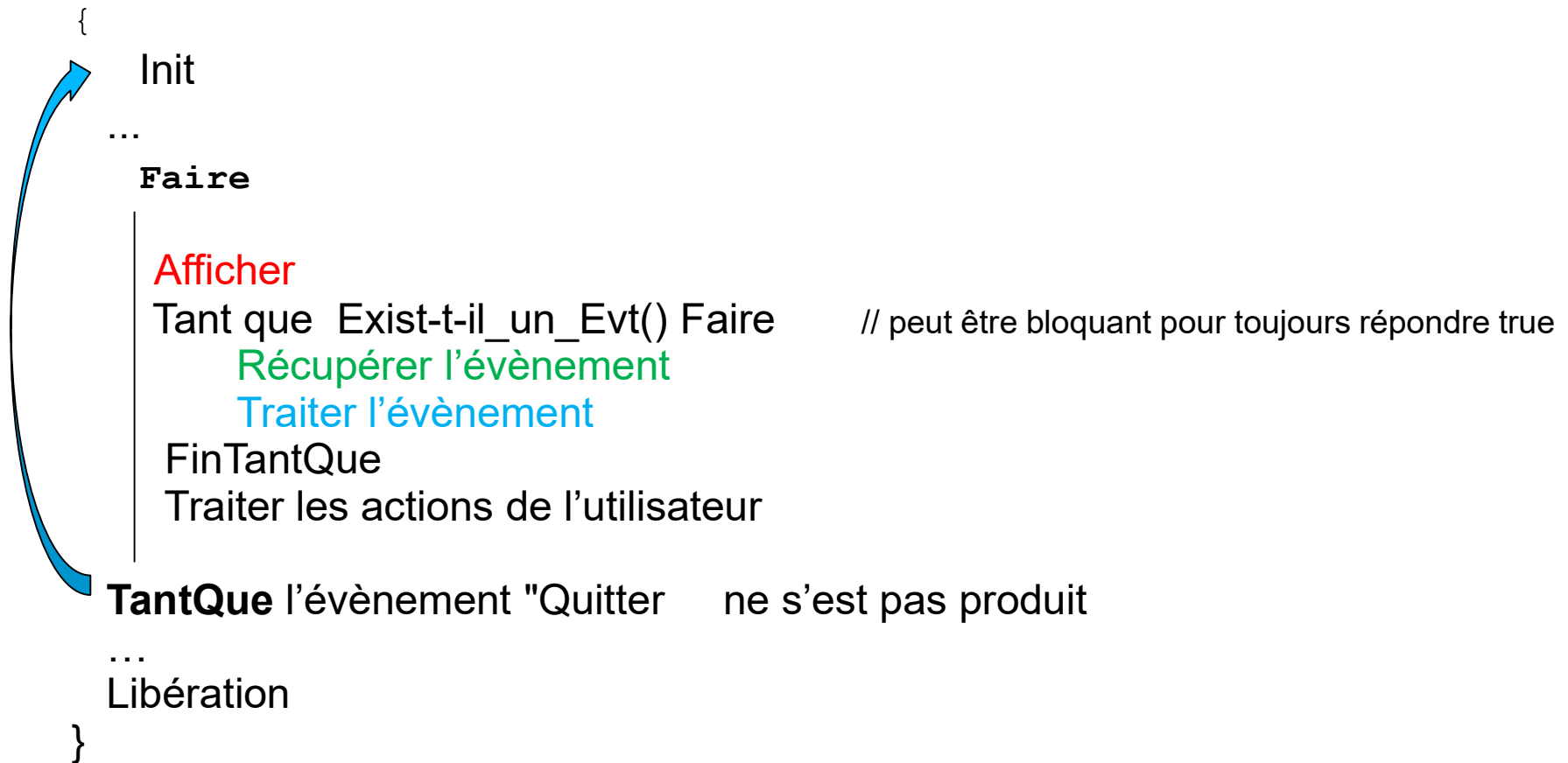
Interaction

```
cin>>choix; ou scanf("%d", &choix);
```

- scanf attend la touche 'entrée' après le choix
- Pas valable pour une application interactive
par ex. pacman : on appuie sur la touche et pacman bouge (pas besoin d'appuyer sur 'Entrée')
- Ceci n'est pas possible avec scanf ou cin>>t;
==> utilisation de bibliothèques :
 - Voir code WinTXT/Pacman (ou NCURSES) en mode texte
 - SDL en mode graphique (ou autres)

Interaction

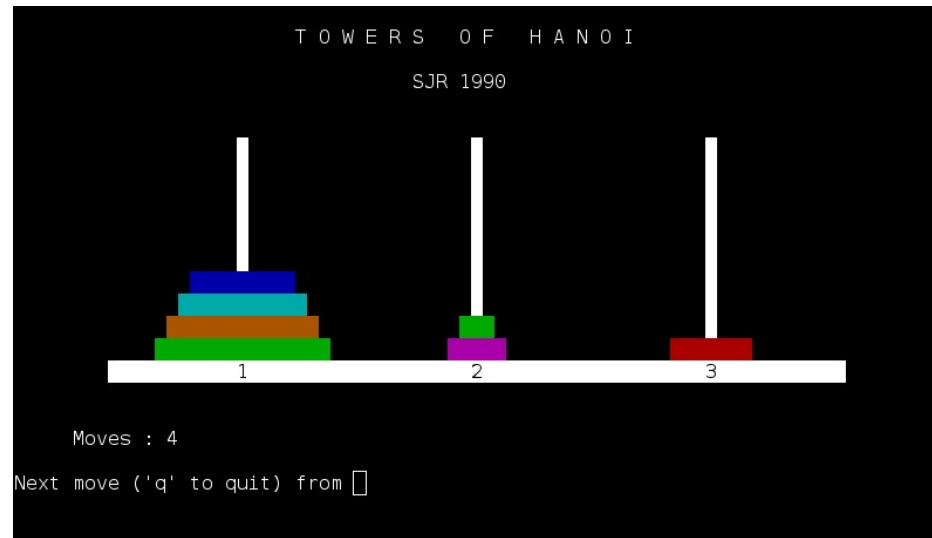
- Récupérer un évènement peut-être bloquant ou non



Interaction

```
cin>>choix; ou scanf("%d", &choix);
```

- scanf est bloquant (cad attend une touche+"Entrée")
- Pas valable pour une application interactive
par ex. pacman, les fantômes bougent même si
l'utilisateur ne tape pas sur une touche
- Remplacer scanf par un test du genre
 - Si UneToucheAEteEnfoncee alors
choix=LireTouche();
- Ceci existe dans SDL et toutes les autres lib d'IHM
(notion d'évènement)

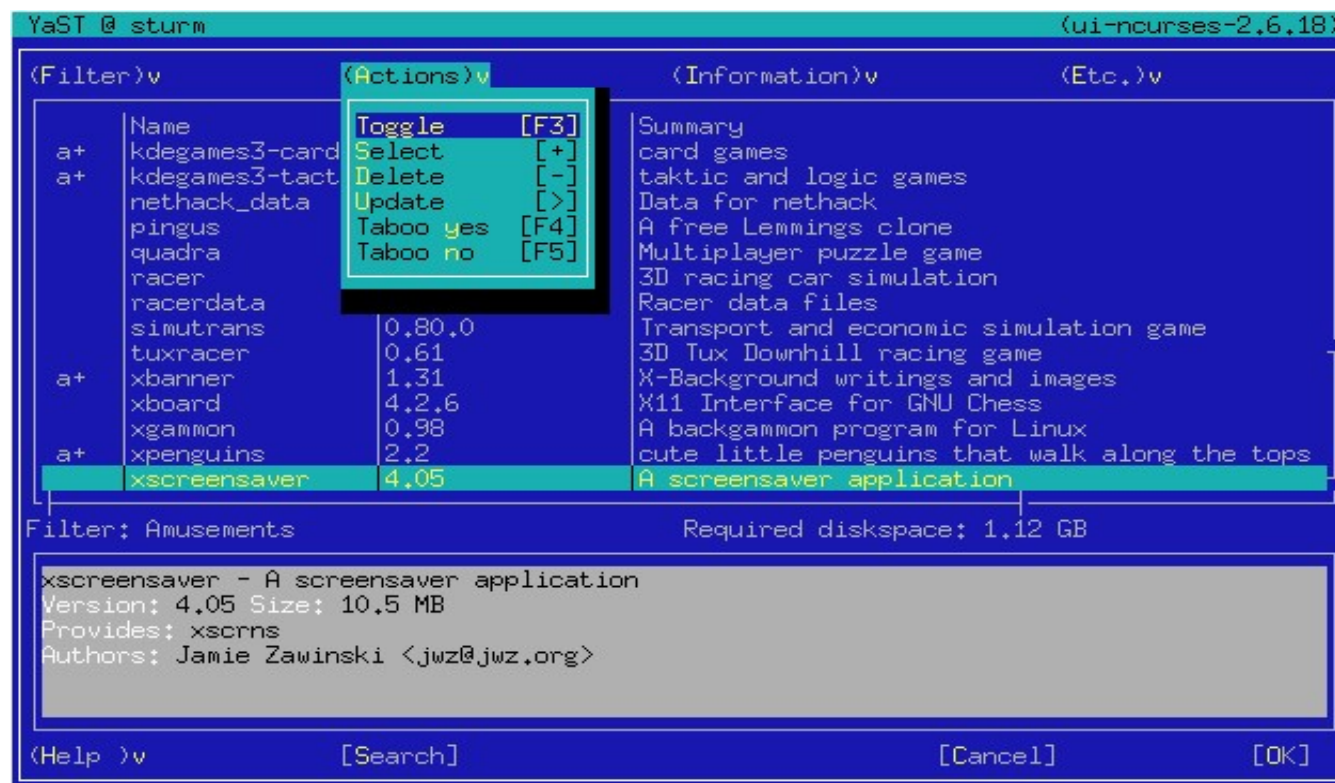


Une application en mode texte

(souvent juste le prototype de base v0.1)

Fenêtres et Texte

- Gestion de l'écran texte avec la notion de fenêtres
 - Code facile à faire (cf. un TP de LIF1 + pacman)
 - Il existe la librairie NCURSES



WinTXT.h / .cpp

affichage texte

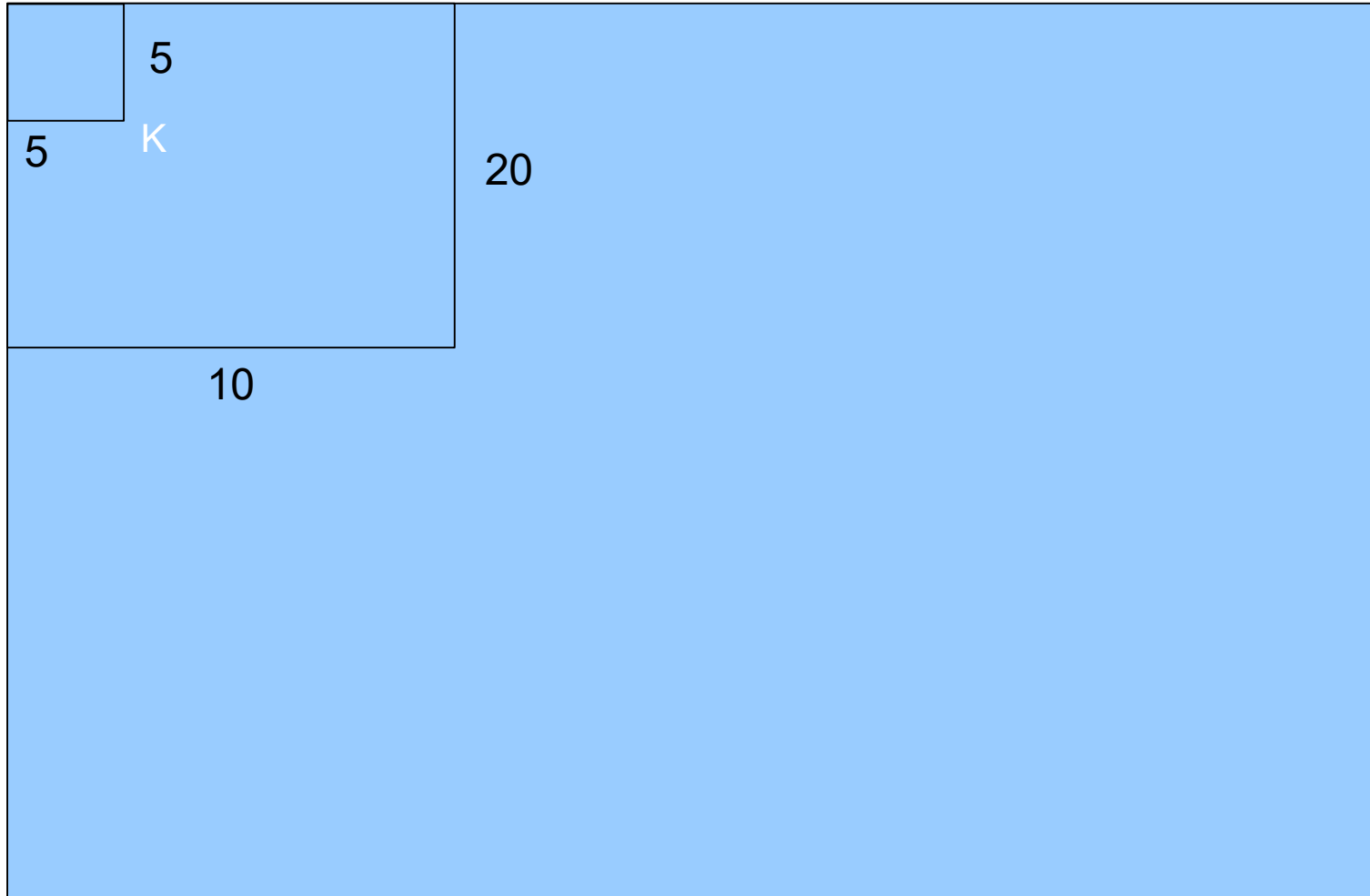
[illegible]

WinTXT

un exemple d'utilisation

```
WinTXT win( 20, 10);      // une fenêtre de taille 20 x 10
ok = true;
do
{
    win.Draw();           // Affichage de la fenêtre sur la console
    c = win.getCh();       // lit une touche (non bloquant), c=0 si aucune touche
    switch(c)
    {
        case 'k' :
            win.print(5, 5, 'k');
                        // place le caractère k dans la case 5,5
            break;
        case 'q':
            ok = false;
            break;
    }
} while(ok);
```


WinTXT : la base



L'utilisateur appuie sur la touche 'k' du clavier ...

WinTXT.h / .cpp

- Regardons le fichier .cpp ...
 - De nombreux appels bas niveau aux fonctions systèmes gérant le terminal
 - Linux != Windows mais un unique fichier et surtout un unique .h → portable

→ Copier les 2 fichiers dans votre projet pour la beta version 0.1

PacmanBeta et interface interchangeable

- Regardez le code de PacmanBeta
 - Les modules « coeurs » du jeu
 - Pacman.h/.cpp
 - Fantome.h/.cpp
 - Terrain.h/.cpp
 - Jeu.h/.cpp
 - Le module spécifique pour l'affichage en TXT
txtJeu.h/.cpp
 - Comment compiler : Makefile
 - ➔ Le projet codeblocks utilise le Makefile

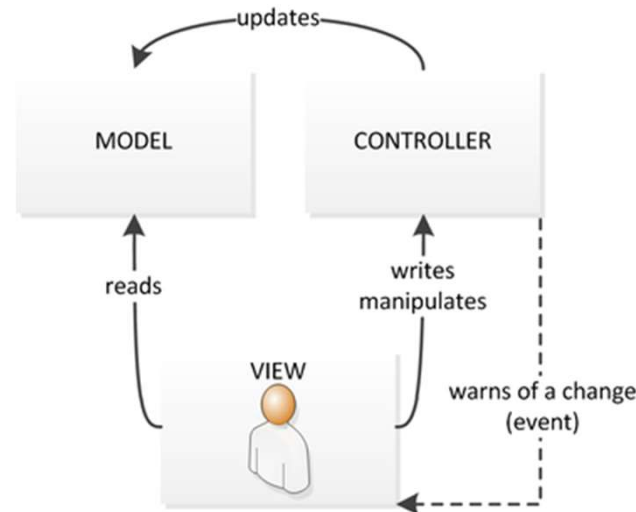
PacmanBeta et interface interchangeable

```
#####
#.....##.....#
#.#####.##...####.#
#.....##.....#
#.....#.....#
#.....#...##.F....#
#####.##...#.....#
#.....##...#.....#
#.....#...#.....#
#.....#.....#
#.....#.....#
#.....#.....#
#P.....#.....#
#####
```

+ DEMO de l'appli

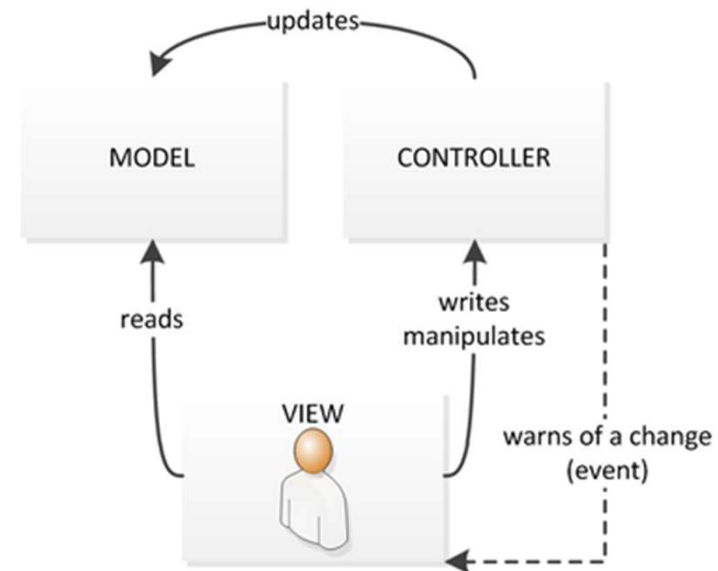
Principe d'organisation du code

Introduction à la notion de MVC Modèle / Vue / Contrôleur



MVC

- **Modèle / Vue / Contrôleur**
- **Design Pattern = patron de conception**
 - arrangement caractéristique de modules,
 - reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels
- MVC est un pattern

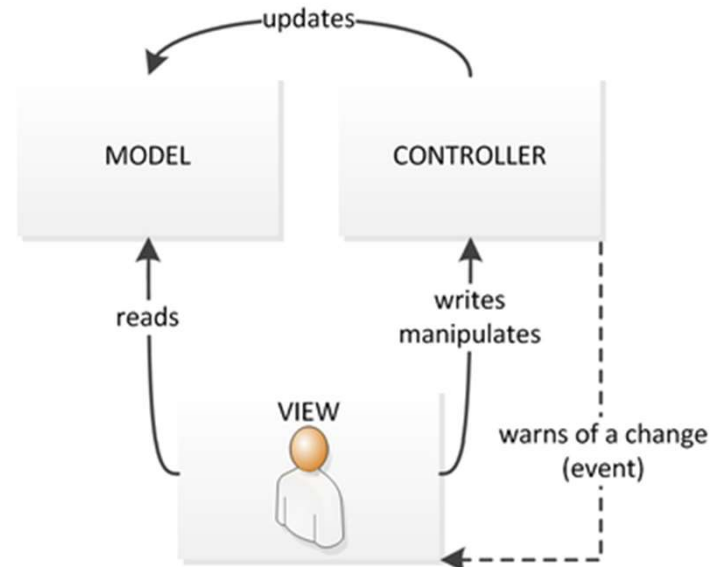


MVC

Modèle / Vue / Contrôleur

En résumé, lorsqu'un client envoie une requête à l'application :

- la requête envoyée depuis la vue est analysée par le contrôleur (via par exemple un *handler* ou [callback](#))
- le contrôleur demande au modèle approprié d'effectuer les traitements et notifie à la vue que la requête est traitée.
- la vue notifiée fait une requête au modèle pour se mettre à jour (par exemple affiche le résultat du traitement via le modèle).



Interface interchangeable

- Que faut-il changer dans PacmanBeta pour remplacer WinTXT par une autre librairie?
- Réponse => ?

Interface interchangeable

- Que faut-il changer dans PacmanBeta pour remplacer WinTXT par une autre librairie?
 - Réponse => que le module ncursJeu
 - 2 fonctions!! presque rien ...
 - L'application a été construite pour être indépendante de l'affichage
- => du coup moins d'efforts, moins de bug, moins de stress ... des développeurs plus heureux!!

Modules interchangeables

- Le coeur de l'application doit être pensée pour être le plus “étanche” possible
 - cad indépendant des choses comme l'affichage, le réseau, etc.
- Les fonctionnalités d'affichage, de réseau, etc. doivent pouvoir être remplacées facilement » par d'autres
 - ==> adaptation à la machine cible, évolution dans le temps, etc.

Interface interchangeable

- Alors allons-y remplaçons WinTXT par SDL dans PacmanBeta pour avoir un peu de graphisme ...



SDL (version 2) pour une application graphique interactive (type jeu)

SDL est une API multimédia

- multi-plateforme
 - Gratuite
- Utilisable en C, C++, C#, Pascal, python, etc.

<http://www.libsdl.org>

SDL2, SDL_image, SDL_ttf, ...

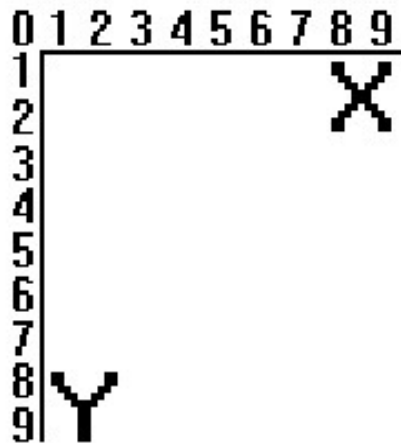
- SDL est composé de 4 bibliothèques(*lib*)
 - SDL_image : gestion du chargement de différents format d'images
 - SDL_ttf : gestion de polices de caractères et de l'affichage de texte
 - SDL_mixer : gestion du son
- Installation
 - Linux

```
apt-get install libsdl2-dev libsdl2-image-dev libsdl2-ttf-dev
```

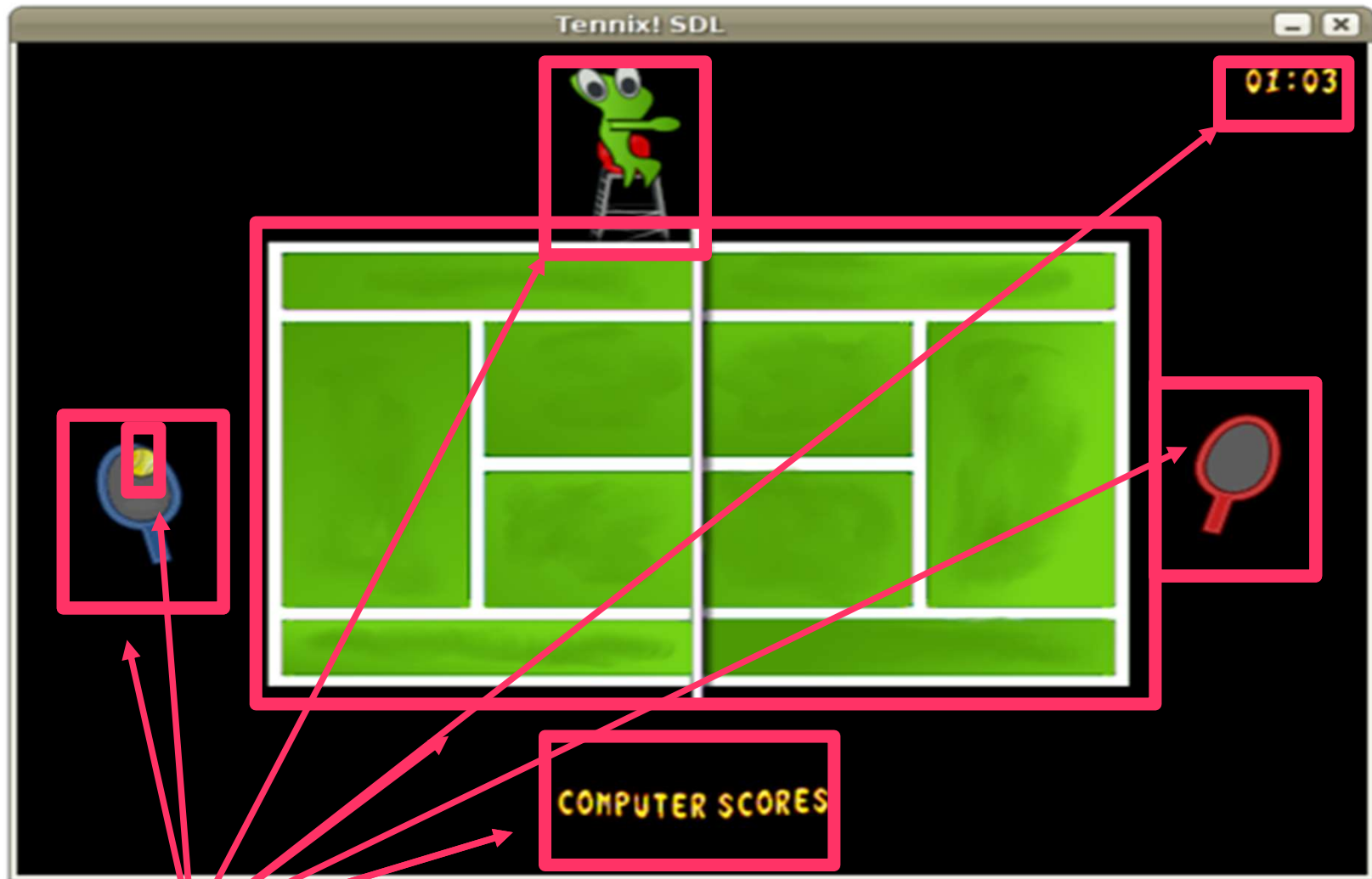
(attention n'installer pas SDL1 ...)
 - Windows : le plus simple prendre le répertoire 'extern' de Pacman

SDL2 : la base

- Notion de surface/texture
 - Une surface/texture en SDL = un rectangle
 - SDL_Surface/SDL_Texture = WinTXT en texte
 - Largeur x Hauteur (=DimX et DimY) + une position à l'écran (posX, posY)
 - axe des Y inversé



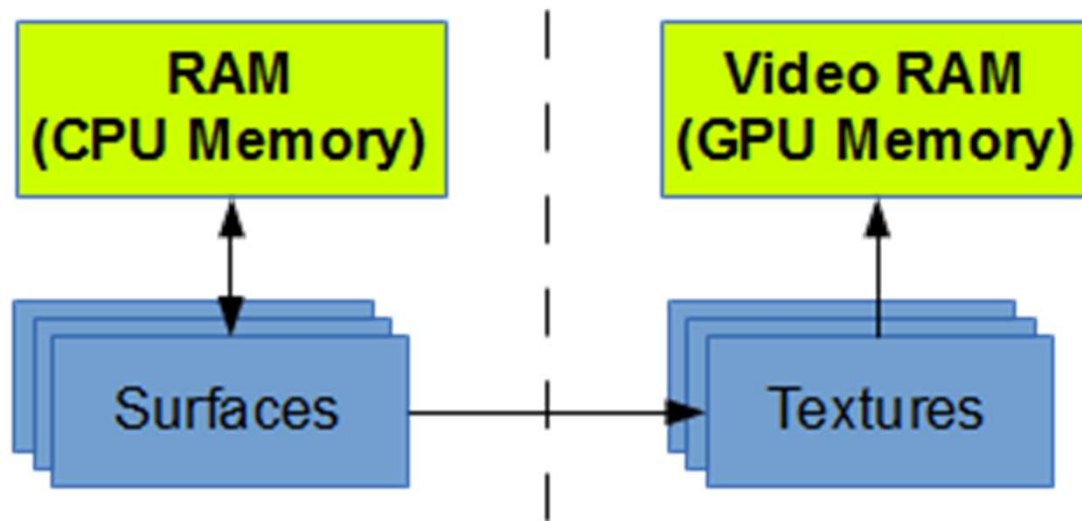
SDL2 : la base



SDL_Surface / SDL_Texture

SDL2 : surface/texture

- Différence entre Surface et Texture
 - Surface → CPU mémoire centrale
 - Très proche du module Image (du TD)
 - Texture → GPU mémoire de graphique



SDL2, la base : init

- **Initialisation, 3 libs donc 3 init :**

```
if (SDL_Init(SDL_INIT_VIDEO) < 0)
{
    std::cout << "Erreur lors de l'initialisation de la SDL : " <<
    SDL_GetError() << std::endl;
    SDL_Quit();
}

if (TTF_Init() != 0) ...

int imgFlags = IMG_INIT_PNG | IMG_INIT_JPG;
if( !( IMG_Init( imgFlags ) & imgFlags ) ) ...
```

- **La fenêtre principale : SDL_Window* window;**

```
sj.window = SDL_CreateWindow(    "Pacman v0.1",
                                SDL_WINDOWPOS_CENTERED,    // X coin haut/gauche
                                SDL_WINDOWPOS_CENTERED,    // Y coin haut/gauche
                                dimx, dimy,
                                SDL_WINDOW_SHOWN | SDL_WINDOW_RESIZABLE);
```

SDL2, la base : Renderer

- Renderer
 - L'outil indispensable à tout affichage : rectangle, ligne, image, etc.
 - Fait le lien avec l'accélération graphique (GPU)

```
SDL_Renderer *renderer;
```

```
renderer = SDL_CreateRenderer(  
    window,          // un renderer est lié à une  
fenetre  
    -1,              // index du driver, -1=default  
    SDL_RENDERER_ACCELERATED); // Accelération  
GPU?
```

SDL2, la base : Surface/Texture

Dans Pacman on vous propose une structure Image

(attention : != du module Image du TD)

- Surface / Image

```
struct Image
{
    SDL_Surface* surface;          // l'image stockée sur
    CPU

    SDL_Texture* texture;          // le renderer ne sait
    afficher que ça (GPU)

    bool has_changed;              // si la surface change il
    faut mettre

};                                // à jour la texture ➔ role de
ce booléen
```

- Création d'une image

SDL2, la base : Surface/Texture

```
Image image(const char* filename, SDL_Renderer * renderer)
{
    Image res;
    res.surface = IMG_Load(filename);

    SDL_Surface * surfaceCorrectPixelFormat =
        SDL_ConvertSurfaceFormat(
                                                    res.surface,

SDL_PIXELFORMAT_ARGB8888, 0);
    SDL_FreeSurface(res.surface);
    res.surface = surfaceCorrectPixelFormat;

    res.texture = SDL_CreateTextureFromSurface( renderer,
res.surface);
    return res;
}
```

SDL2 : afficher une texture

- Recopier une texture sur la fenêtre principale

```
SDL_RenderCopy(renderer, im.texture, NULL, &r);
```

```
SDL_Rect r;    qui contient
```

```
    r.x, r.y           // position en x,y
```

```
    r.w                // largeur de la texture
```

```
    r.h                // hauteur de la texture
```

```
w et h peuvent être différent de la surface  
(taille initiale de l'image)
```

- Avec la structure Image proposée dans Pacman

```
void image_draw(Image& im, SDL_Renderer * renderer, int x, int  
y, int w, int h)
```

SDL2 : afficher une texture

```
void image_draw(Image& im, SDL_Renderer * renderer, int x, int y, int
w, int h)
{
    SDL_Rect r;

    r.x = x;

    r.y = y;

    r.w = (w<0)?im.surface->w:w;
    r.h = (h<0)?im.surface->h:h;

    if (im.has_changed)                // Si la surface a changé, on met à
jour la texture
    {
        ok = SDL_UpdateTexture( im.texture, NULL,
                                im.surface->pixels, im.surface->pitch);

        im.has_changed = false;
    }

    ok = SDL_RenderCopy(renderer, im.texture, NULL, &r);
```

Pacman SDL

- la boucle

```
Uint32  t = SDL_GetTicks(), nt;
bool stop = false;
SDL_Event e;
while (!stop)
{
    nt = SDL_GetTicks();
    if (nt-t>500)
    {
        jeuActionsAutomatiques(sj.jeu);
        t = nt;
    }

    if (SDL_PollEvent(&e)) {
        if (e.type == SDL_QUIT) stop = true;
        else if ( e.type == SDL_KEYDOWN )
        {
            switch ( events.key.keysym.scancode )
            {
                case SDL_SCANCODE_UP:
                    jeuActionClavier( sj.jeu, 'b');
                    break;
                    .....
            }

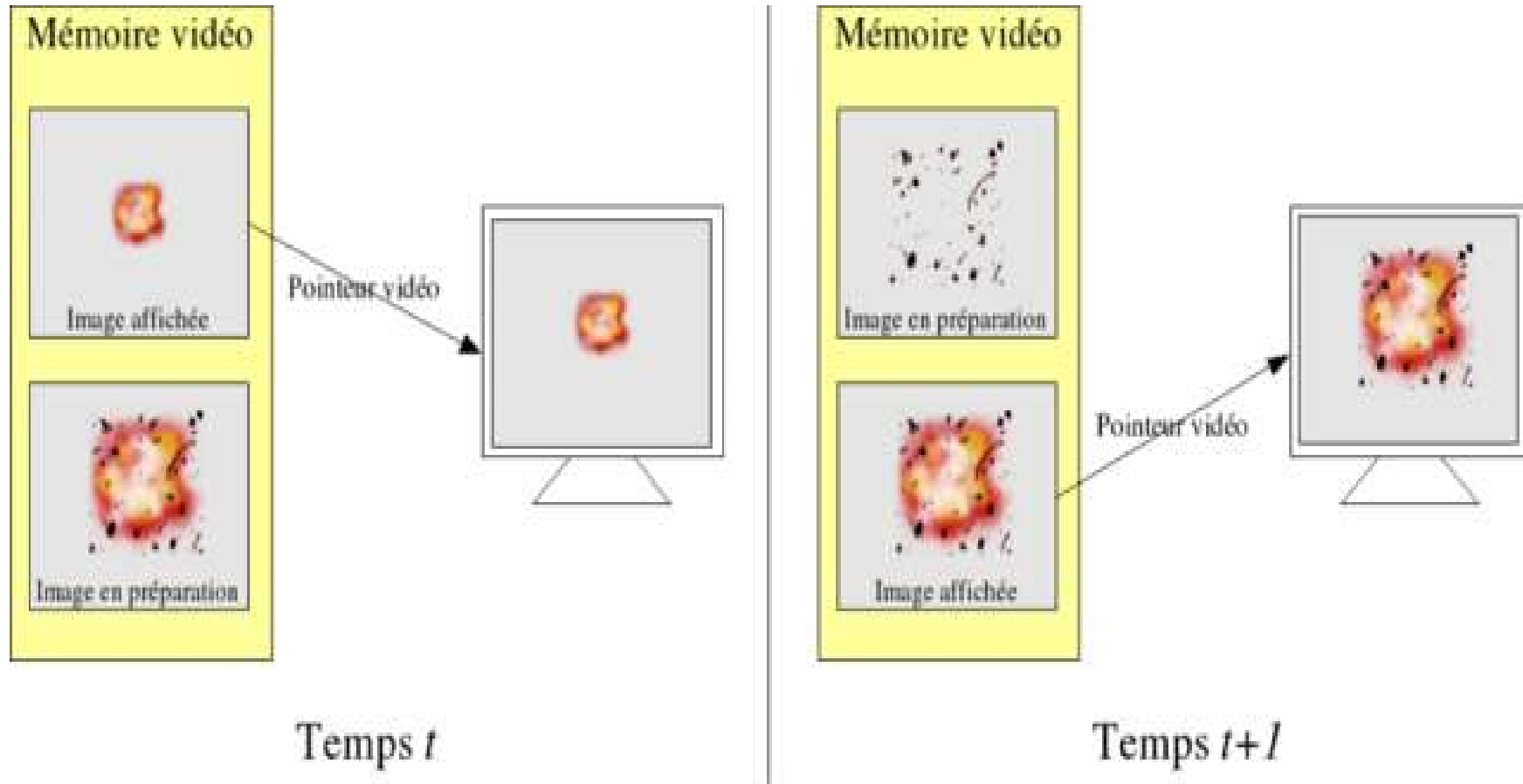
            sdljeuAff( sj );
            SDL_RenderPresent(renderer);
        }
    }
}
```

**Gestion du temps pour les actions
non utilisateur → fantômes bougent
toutes les 500ms**

**Gestion du clavier : ici réaction à un
évènement 'KEYDOWN'.**

**Il est possible également de tester si
une touche est enfoncée n'importe
où dans le programme :**
const Uint8 *state = SDL_GetKeyboardState(NULL);
if (state[touche]) ...

SDL2 : Double buffering



`SDL_RenderPresent(sj.renderer); // permute les 2 buffers`

SDL2 : fermeture

- `SDL_DestroyRenderer(ren);`
- `SDL_DestroyWindow(win);`
- `IMG_Quit();` // `SDL_image`
- `TTF_Quit();` // `SDL_ttf`
- `MIX_CloseAudio();` // `SDL_mixer` : son
- `SDL_Quit();`

Exemple SDL minimaliste - Init

```
#include "SDL.h"

int main(int argc, char *argv[])
{
    SDL_Window *win = NULL;
    SDL_Renderer *renderer = NULL;
    SDL_Texture *bitmapTex = NULL;
    SDL_Surface *bitmapSurface = NULL;
    int posX = 100, posY = 100, width = 320, height = 240;

    SDL_Init(SDL_INIT_VIDEO);

    win = SDL_CreateWindow("Hello World", posX, posY, width, height);

    renderer = SDL_CreateRenderer(win, -1, SDL_RENDERER_ACCELERATE);

    bitmapSurface = IMG_Load("img/hello.bmp");
    bitmapTex = SDL_CreateTextureFromSurface(renderer, bitmapSurface);
    SDL_FreeSurface(bitmapSurface);

    ...
}
```

Exemple SDL minimaliste

- la boucle
- quitter

```
bool stop = false;
SDL_Event e;
while (!stop)
{
    if (SDL_PollEvent(&e))
    {
        if (e.type == SDL_QUIT) stop = true;
    }

    SDL_RenderClear(renderer);                // Efface l'écran
    SDL_RenderCopy(renderer, bitmapTex, NULL, NULL); // Dessine texture
    SDL_RenderPresent(renderer);              // Permute les 2 buffers
}

SDL_DestroyTexture(bitmapTex);
SDL_DestroyRenderer(renderer);
SDL_DestroyWindow(win);
SDL_Quit();

return 0;
```

Interface interchangeable

- Alors allons-y remplaçons WinTXT par SDL dans PacmanBeta pour avoir un peu de graphisme ...

==> Regardons le module sdlJeu.h/.cpp

+ DEMO de l'appli

SFML

- SFML, très similaire à SDL2
 - SFML offre une interface simple vers les différents composants de votre PC, afin de faciliter le développement de jeux ou d'applications multimedia. Elle se compose de cinq modules : système, fenêtrage, graphisme, audio et réseau.
 - Multi plateforme
 - Multi langage
 - <https://www.sfml-dev.org/index-fr.php>



imgui

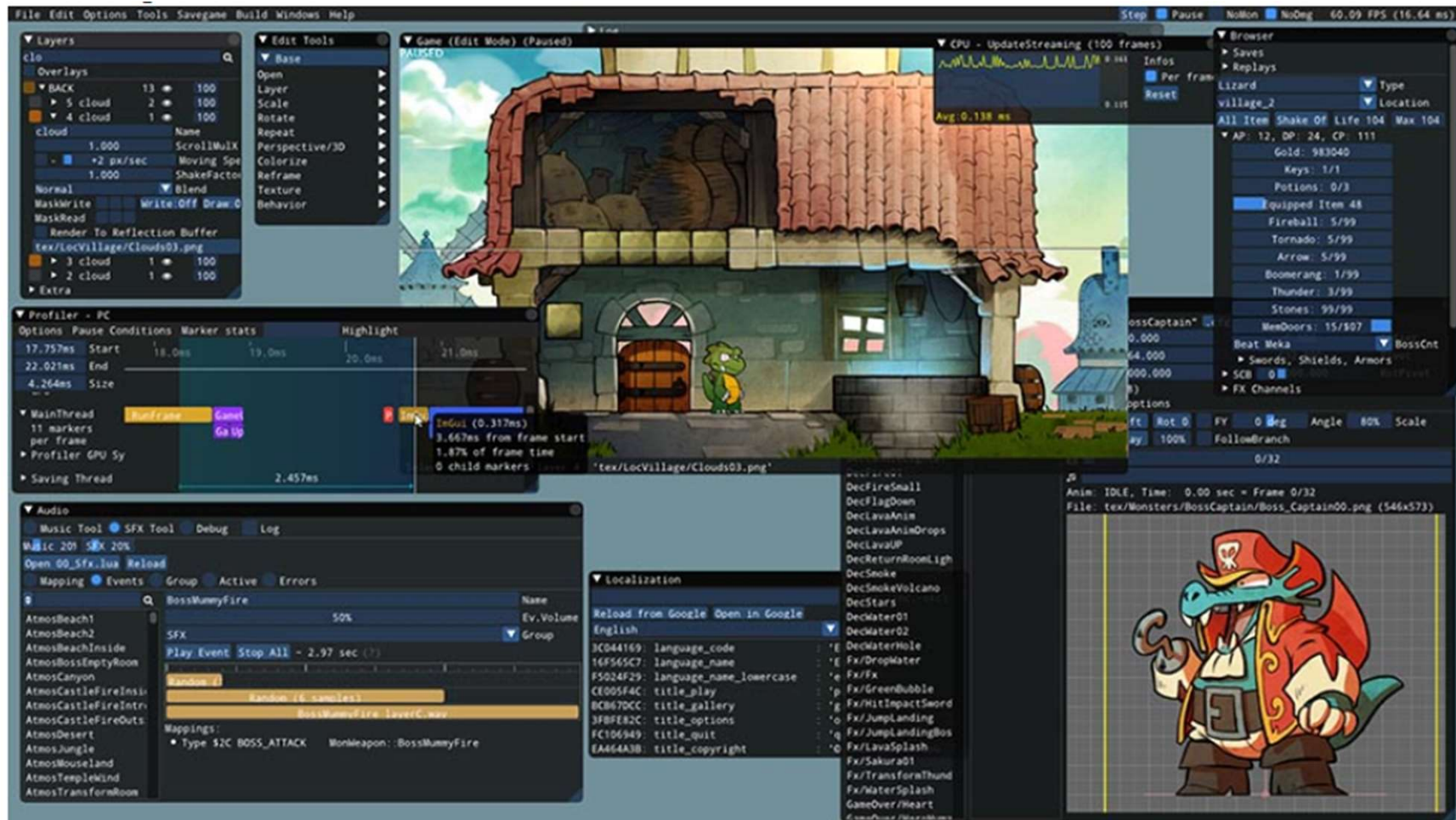
- ImGui
 - <https://github.com/ocornut/imgui>
 - Multi plateforme
 - Multi langage
 - Se greffe par-dessus de nombreux frameworks
 - SDL2, Qt, SFML, etc.
- ImGui par-dessus SDL
 - Voir par exemple
<https://retifrav.github.io/blog/2019/05/26/sdl-imgui/>

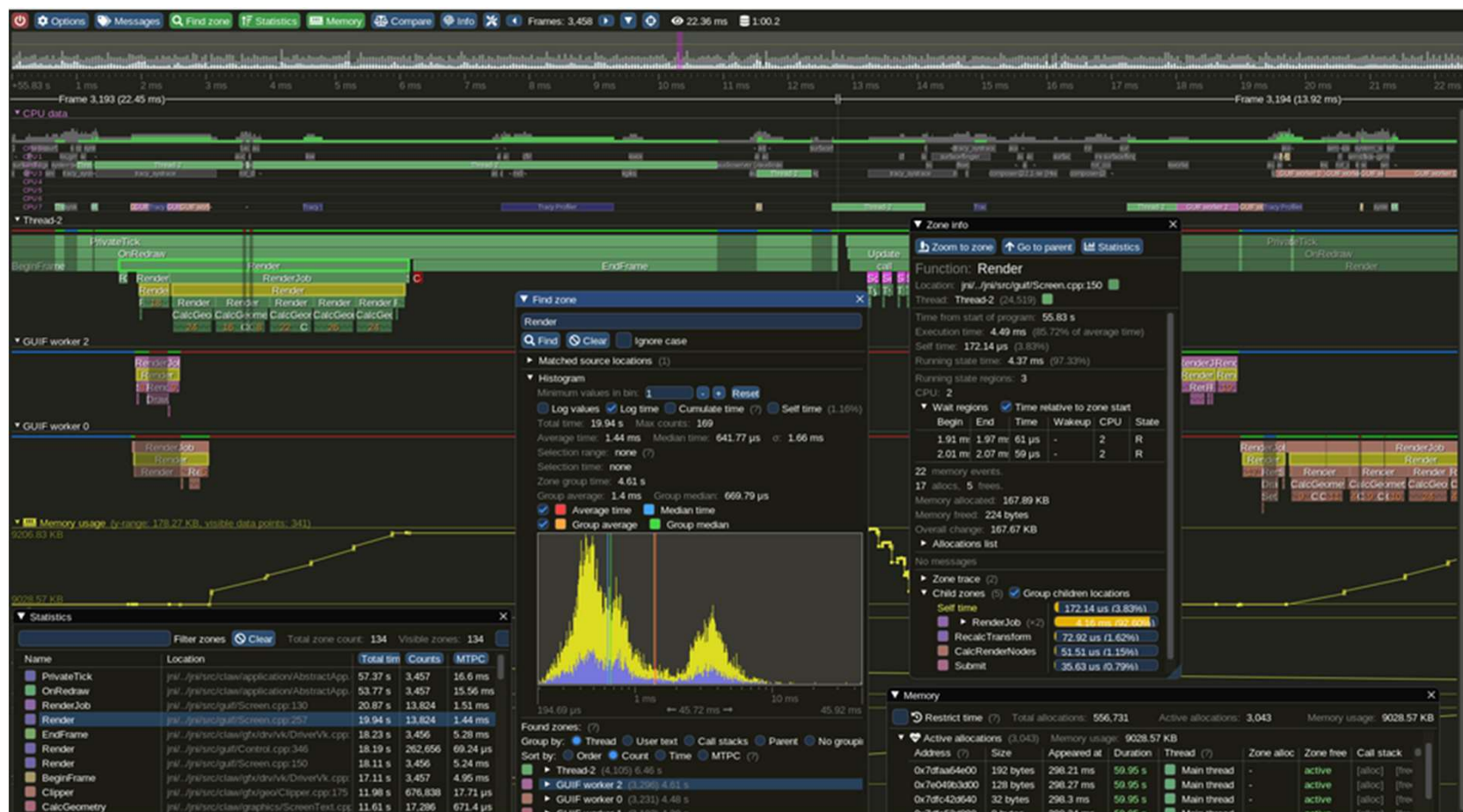
imgui

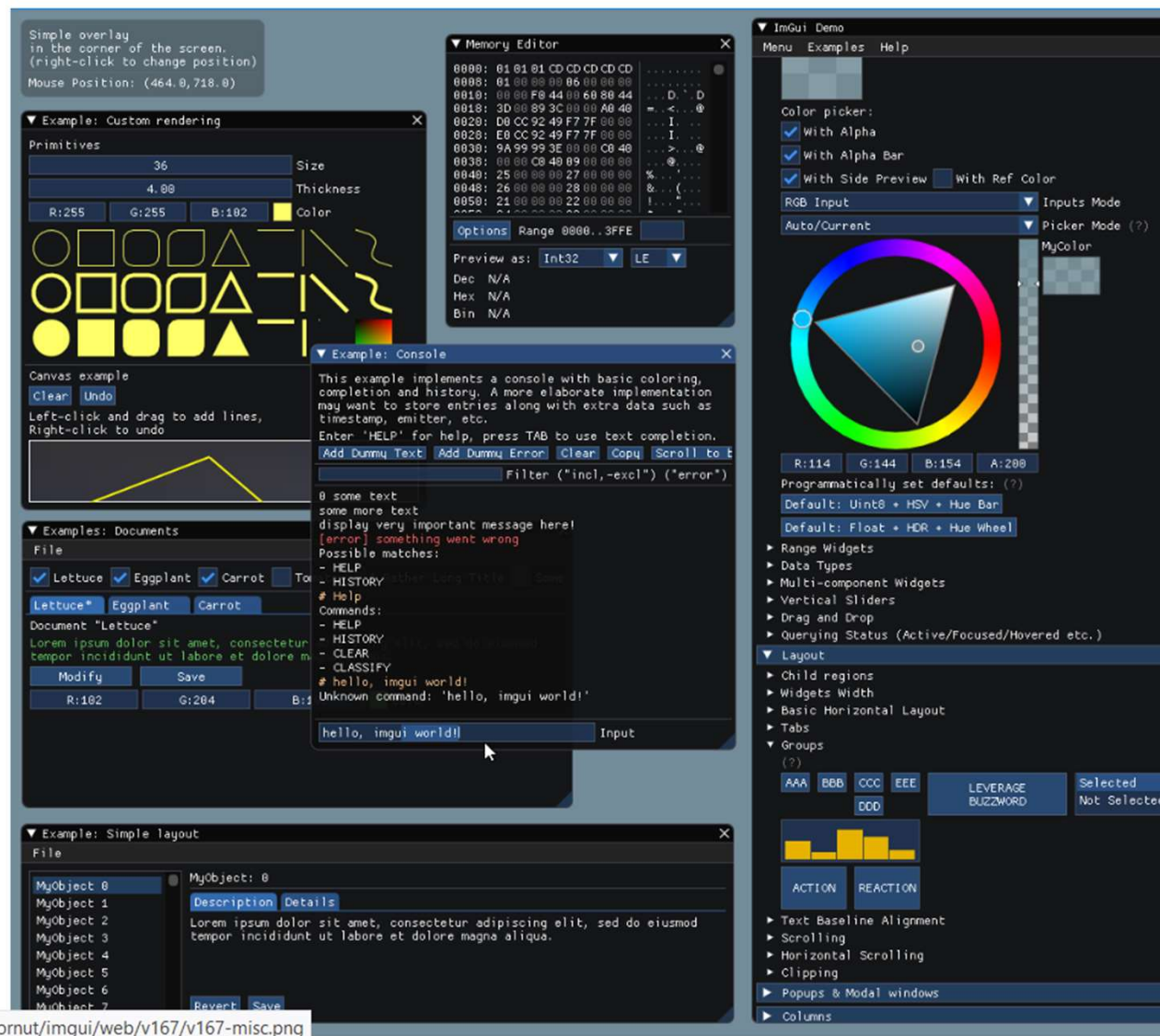
- **The core of Dear ImGui is self-contained within a few platform-agnostic files** which you can easily compile in your application/engine. They are all the files in the root folder of the repository (imgui.cpp, imgui.h, imgui_demo.cpp, imgui_draw.cpp etc.).
- **No specific build process is required.** You can add the .cpp files to your existing project.
- You will need a backend to integrate Dear ImGui in your app. The backend passes mouse/keyboard/gamepad inputs and variety of settings to Dear ImGui, and is in charge of rendering the resulting vertices.
- **Backends for a variety of graphics api and rendering platforms** are provided in the [backends/](#) folder, along with example applications in the [examples/](#) folder. See the [Integration](#) section of this document for details. You may also create your own backend. Anywhere where you can render textured triangles, you can render Dear ImGui.

imgui

- Franchement COOL ! Pour LIFAP4



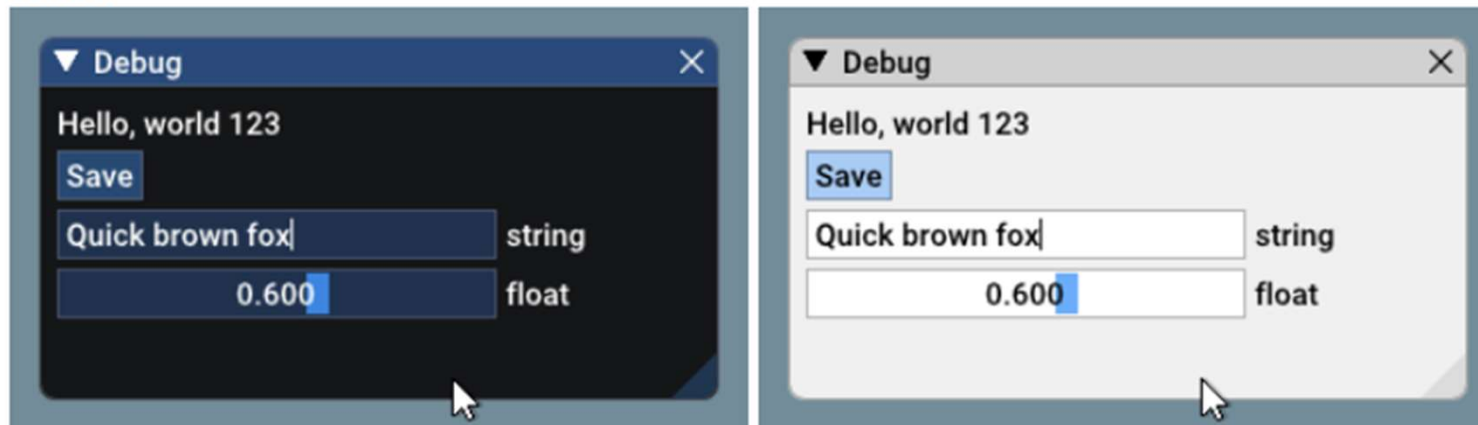




ImGui : exemple de code

```
ImGui::Text("Hello, world %d", 123);  
if (ImGui::Button("Save"))  
    MySaveFunction();  
ImGui::InputText("string", buf, IM_ARRAYSIZE(buf));  
ImGui::SliderFloat("float", &f, 0.0f, 1.0f);
```

Result:



(settings: Dark style (left), Light style (right) / Font: Roboto-Medium, 16px)

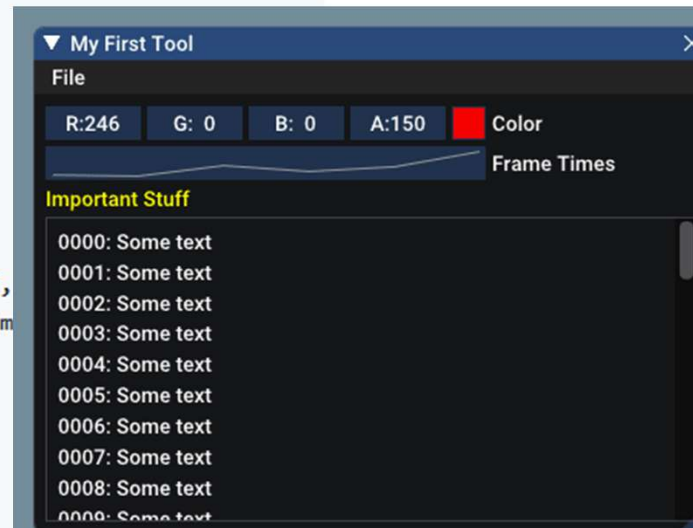
ImGui : exemple de code

```
// Create a window called "My First Tool", with a menu bar.
ImGui::Begin("My First Tool", &my_tool_active, ImGuiWindowFlags_MenuBar);
if (ImGui::BeginMenuBar())
{
    if (ImGui::BeginMenu("File"))
    {
        if (ImGui::MenuItem("Open..", "Ctrl+O")) { /* Do stuff */ }
        if (ImGui::MenuItem("Save", "Ctrl+S")) { /* Do stuff */ }
        if (ImGui::MenuItem("Close", "Ctrl+W")) { my_tool_active = false; }
        ImGui::EndMenu();
    }
    ImGui::EndMenuBar();
}

// Edit a color (stored as ~4 floats)
ImGui::ColorEdit4("Color", my_color);

// Plot some values
const float my_values[] = { 0.2f, 0.1f, 1.0f, 0.5f, 0.9f,
ImGui::PlotLines("Frame Times", my_values, IM_ARRAYSIZE(m

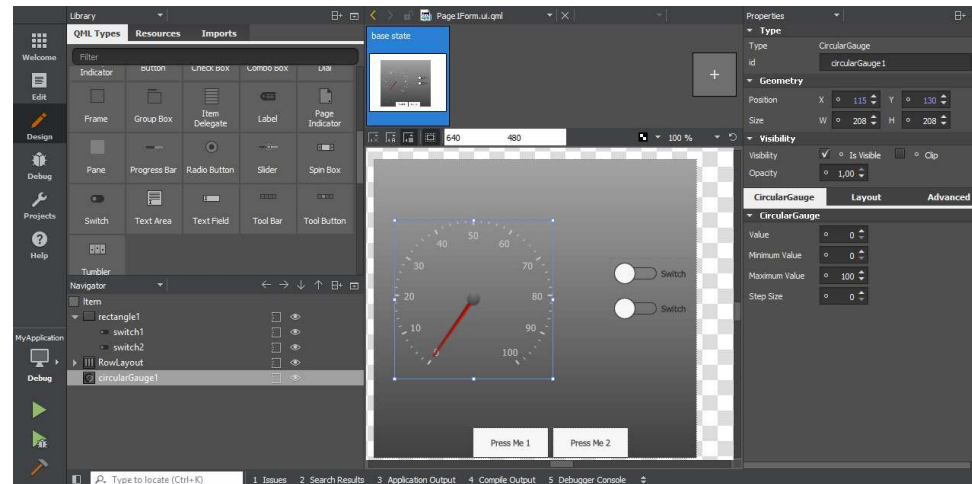
// Display contents in a scrolling region
ImGui::TextColored(ImVec4(1,1,0,1), "Important Stuff");
ImGui::BeginChild("Scrolling");
for (int n = 0; n < 50; n++)
    ImGui::Text("%04d: Some text", n);
ImGui::EndChild();
ImGui::End();
```





Code less.
Create more.
Deploy everywhere.

Qt



Survol pendant ce cours

Beaucoup à apprendre durant votre projet

Qt



- Librairie graphique écrite en C++ par la société TrollTech
- Mécanisme pour interagir
 - avec l'utilisateur (bouton, liste déroulante, ..) avec le système (OpenGL, Xml, SQL, sockets, plugin ...)!
 - Multi-Plateforme !
 - Gratuit (GPL)
- mais dispose aussi d'une licence commerciale

Qt

- Utilisateurs de Qt
 - Nokia, Nasa, Adobe, Motorola
Google, ... !
- Bindings (java, python, c#)



Qt 5 : 10 000 téléchargement par jour



Qt in Automotive
Infotainment



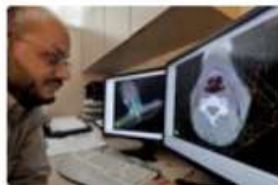
Qt in Aerospace



Qt in Home Media



Qt in IP Communication



Qt in Medical



Qt in Oil & Gas

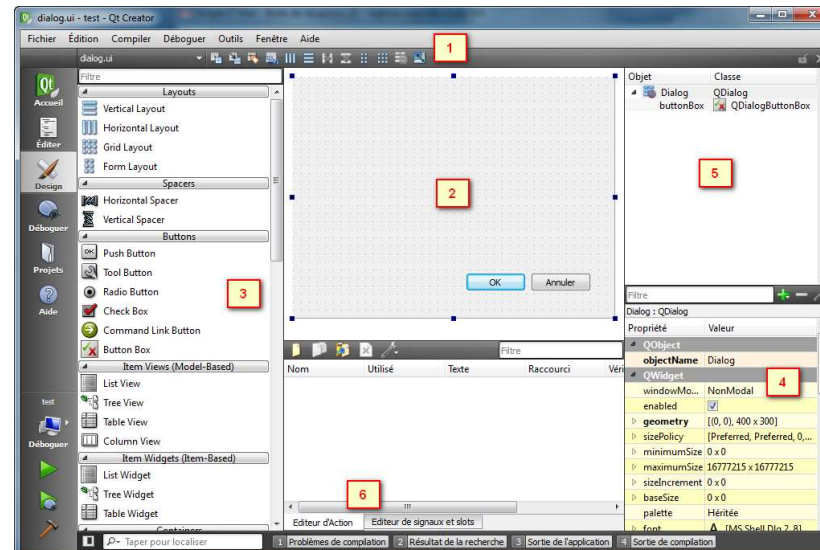


Qt in Visual Effects

Avantages 1/2



- **Cross platform GUI framework**
 - Développer sur une machine et portage immédiat sur une autre (windows, linux, macos, android, etc.)
 - Plus qu'une bibliothèque car fournit des outils, etc.
- **API riche (class/C++ library)**
- **IDE : qtcreator**
- **Système de compilation/projet : qmake / fichier .pro**
- **Graphique : performant + outils de création d'interface**



Avantage 2/2



- **Cross platform GUI framework**
 - Développer sur une machine et portage immédiat sur une autre (windows, linux, macos, android, etc.)
 - Plus qu'une bibliothèque car fournit des outils
-
- Chaines de caractère avec système de traduction pour l'internationalisation
- Timers élégant et puissant (cf. pacman)
- Structuration des objets hiérarchique sous forme d'arbre
- Pointeurs "intelligent" (QPointer)
 - à 0 quand la référence disparaît

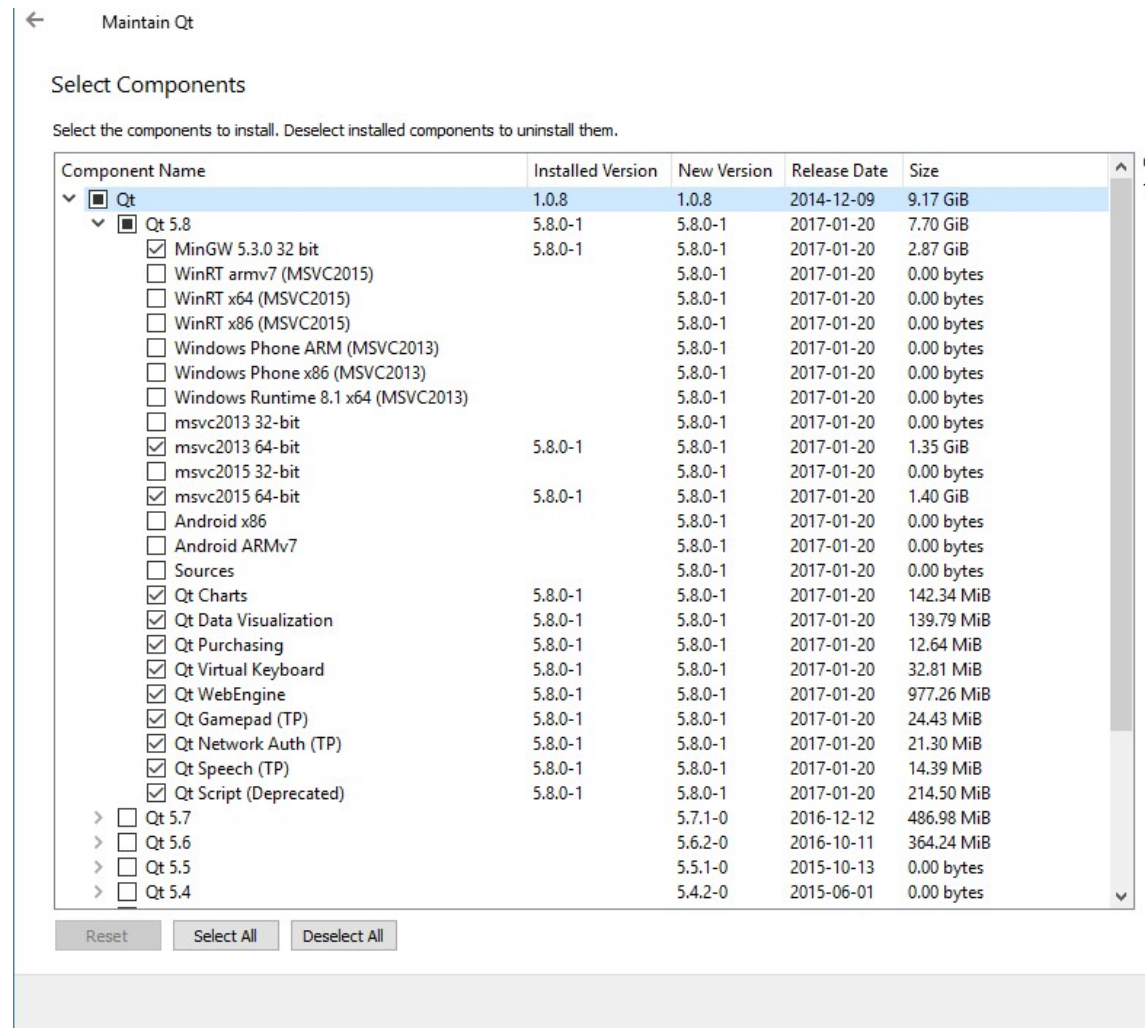
Outils



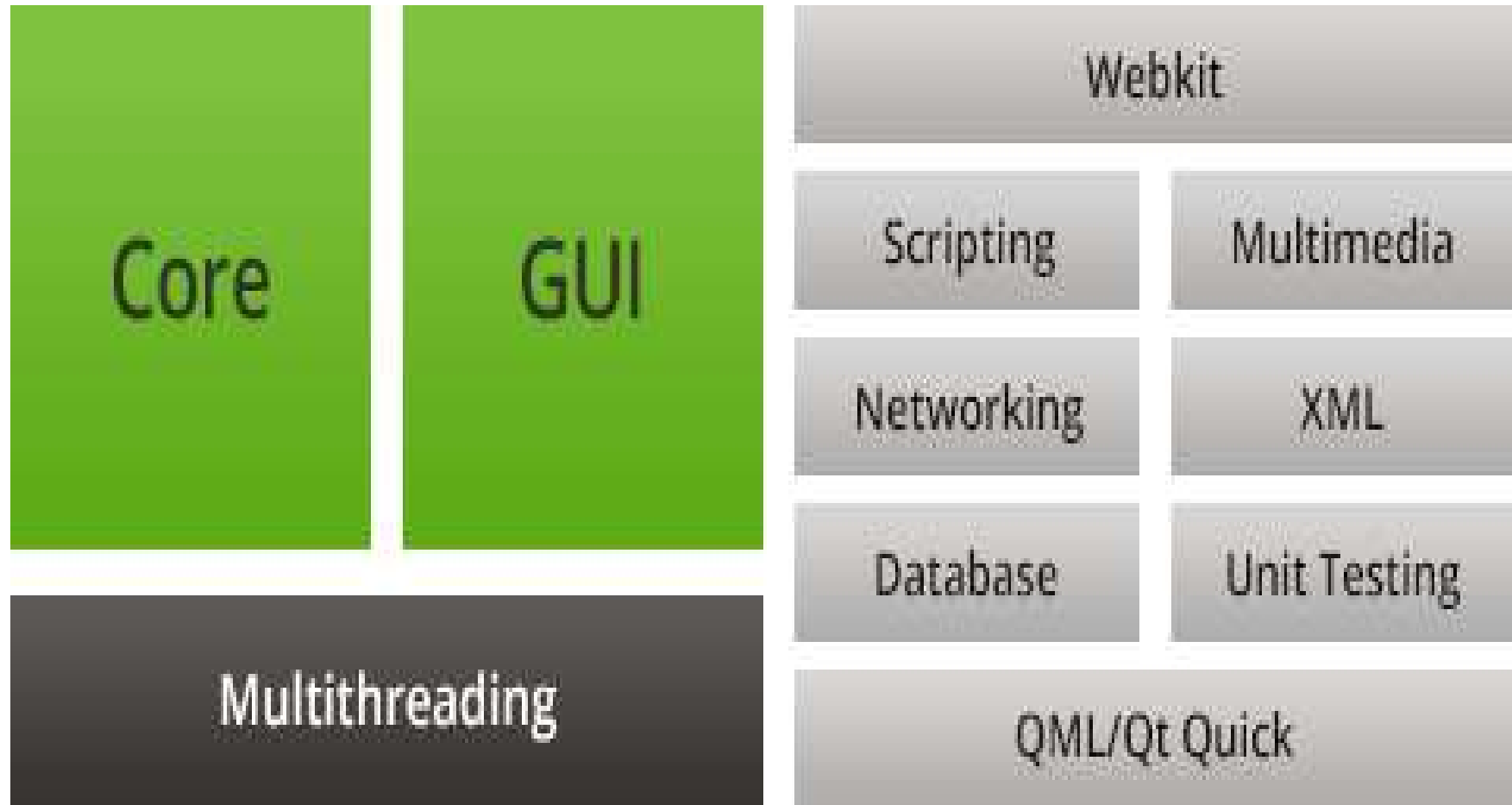
- Qt Creator : Cross platform IDE
- Qt Designer: GUI layout and forms builder
- Qt Linguist: Internationalization toolset
- Qt Assist: Customizable documentation reader
- Qt Qmake: Cross platform build tool
- Plugin for other IDE: Integration with Visual Studio and Eclipse
- Configure: Tool to configure Qt on any specific platform
- Qt SDK: Rich C++ library

Qt : installation

- <https://www.qt.io/download/>



Qt Modular Class Library



Compilation (Qmake)

Qmake

- en entrée un fichier Project (.pro)
- génère un Makefile pour les plateformes cibles.

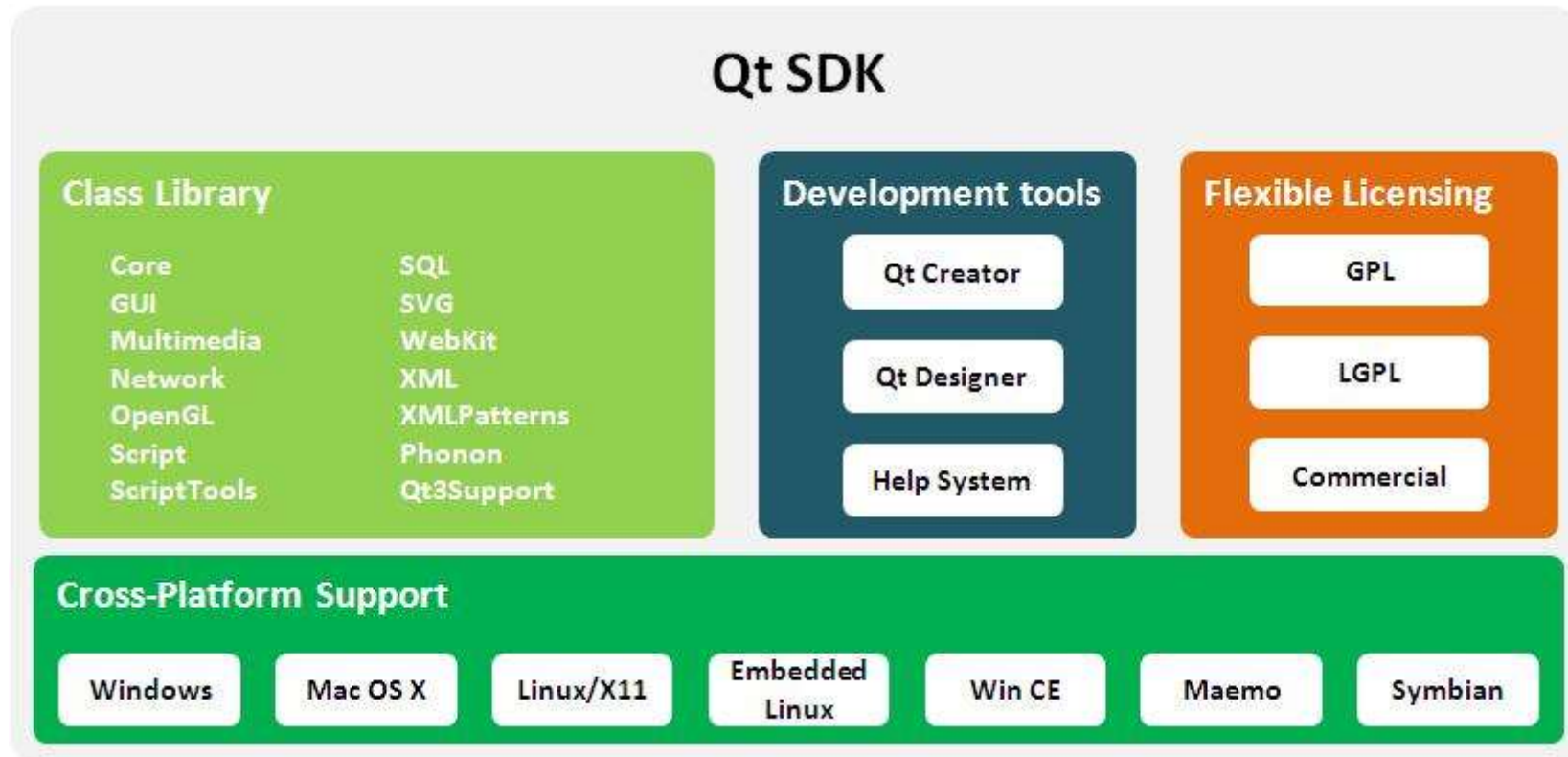
\$Qmake –project (generates “hello.pro”)

\$qmake hello.pro(generates “Makefile”)

\$make (use nmakeif running on Windows)

Qmake génère les règles de compilations + les invocations à moc pour les .h spécifique à Q-Object

Qt Architecture



Qt SDK

Bibliothèque de classes fournit toutes les fonctions de haut niveau pour une applications robuste

- Core Framework
- GUI Framework framework
- SQL Framework framework
- XML Framework
- Networking Framework
- OpenGL Framework
- Multimedia Framework
- WebKit Framework
- Phonon Framework
- OpenGL framework
-2D with Painter
- Scene Graph
- SVG Framework

Qt Class Library

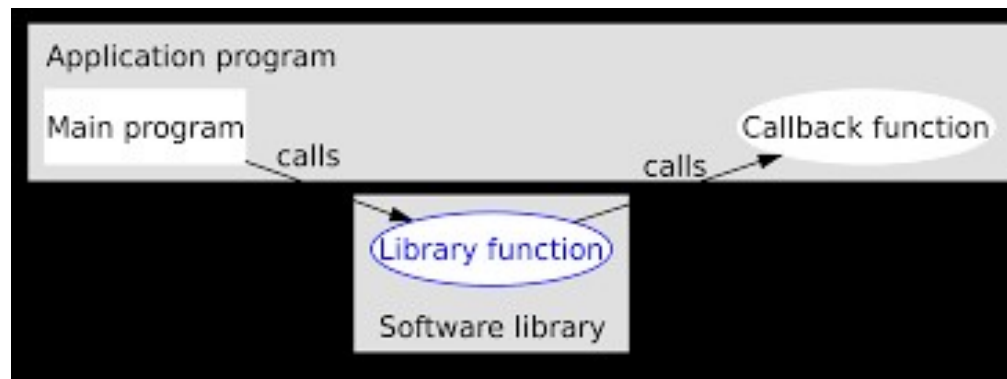
QtCore: File I/O, event and object handling, multi-threading and concurrency, plugins, setting management signals and slots inter-object communications mechanism

QtGui: Set of customizable widgets, 2D graphics canvas and OpenGL® integration, powerful font and layout engine style engine and widget style sheets, anti-aliasing, vector deformation, and SVG support, advanced graphics effects support for ARGB top-level widgets

Callbacks

- Callbacks (un pointer vers une fonction)
- Problèmes
 - pas de vérifications de type (void*, voir partie Menu de ce cours)
 - est-ce que le programmeur a bien respecté les types ?

➔ Qt propose un système de signal/slot/connexion avec une surcouche du compilateur : exécutable moc



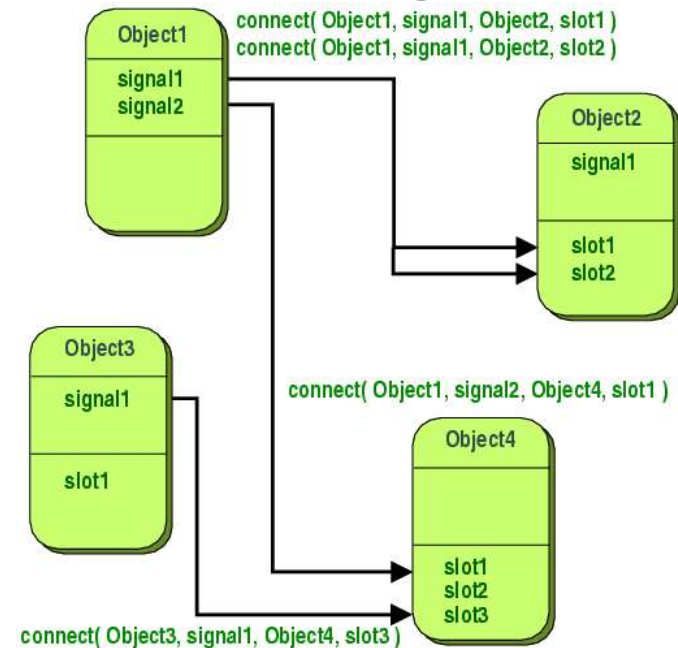
Communication Inter-Object

- L'utilisateur clique sur le bouton OK
 - Que se passe-t-il?
 - Comment réagir ?

Réponse avec Qt

Signals and slots

~ comme un callbacks mais avec le type vérifié (type safe)

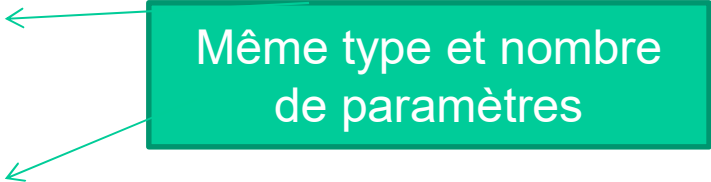


- “**Signals**” sont émis par les objets (ici le bouton emet un « clicked »)
- “**Slots**” sont des fonction qui réagissent aux “signals” mais dont le type est vérifié par Qt/moc
- « Signals » sont connectés aux « Slots » par des appels à « **Connect** »

Signals, Slots et Connect

- Signal et Slots sont émis et exécuté pendant que le programme tourne
- La fonction “connect” branche un signal et un slots

```
connect( slider,  
        SIGNAL( valueChanged( double ) ) ,  
        plot,  
        SLOT(setIntervalTime(double)));
```



Même type et nombre
de paramètres

- Il y a des “signals” et “slots” prédéfinis
- Vous pouvez en définir des nouveaux en fonction de ce que vous avez besoin
- Dans le .h (header file)
 - Q_SIGNALS(signals):
 - Q_SLOTS(slots):
- Analysé par Qt’s Meta Object Compiler (moc)

Signals, slots et events

Signals

- Un signal est un moyen d'avertir un “observateur” que quelque chose s'est passé
- Par exemple
 - Un PushButton est appuyé où la valeur du Slider a change
- Dans votre code vous pouvez émettre des “signals”

Slots

- La fonction slot ne renvoie rien (void)
- Prend en paramètre un Event

Events

- Une instance de la classe Event (ou dérivée) est créée et arrive à la fonction Slots

Type d'*Event*

- Il existe plusieurs types d'"events" en fonction des besoins
 - [QResizeEvent](#)
 - [QPaintEvent](#)
 - [QMouseEvent](#)
 - [QKeyEvent](#)
 - [QCloseEvent](#)
- Par exemple

```
void keyPressEvent(QKeyEvent *keyEvent);
```

Hello World

```
4  #include <QWidget>
5  #include <QLabel>
6
7  namespace Ui {
8  class Widget;
9  }
10
11 class Widget : public QWidget
12 {
13     Q_OBJECT
14
15     public:
16         explicit Widget(QWidget *parent = 0);
17         ~Widget();
18
19     private:
20         Ui::Widget *ui;
21
22         QLabel      *label;
23 };
24
```

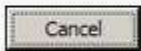
Hello World

```
Widget::Widget(QWidget *parent) :  
    QWidget(parent),  
    ui(new Ui::Widget)  
{  
    ui->setupUi(this);  
    label = new QLabel("Hello World!", this);  
}  
  
Widget::~~Widget()  
{  
    delete ui;  
}
```

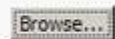
```
4 int main(int argc, char *argv[])  
5 {  
6     QApplication a(argc, argv);  
7     Widget w;  
8     w.show();  
9  
10    return a.exec();  
11 }
```

Qt : galerie de widgets

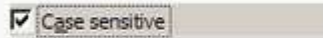
Buttons



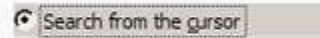
The `QPushButton` widget provides a command button.



The `QToolButton` class provides a quick-access button to commands or options, usually used inside a `QToolBar`.



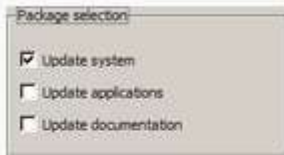
The `QCheckBox` widget provides a checkbox with a text label.



The `QRadioButton` widget provides a radio button with a text or pixmap label.

Qt : galerie de widgets

Containers



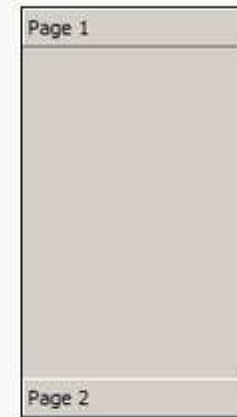
The **QGroupBox** widget provides a group box frame with a title.



The **QTabWidget** class provides a stack of tabbed widgets.



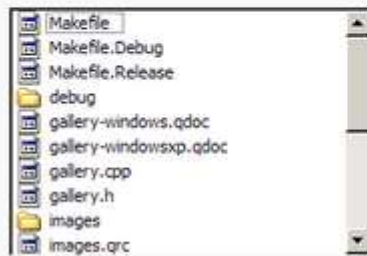
The **QFrame** widget provides a simple decorated container for other widgets.



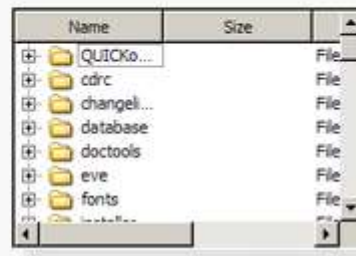
The **QToolBox** class provides a column of tabbed widget items.

Qt : galerie de widgets

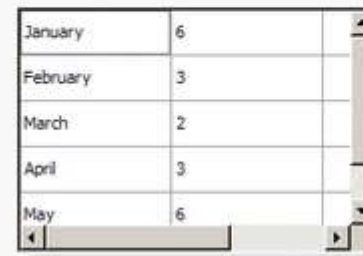
Item Views



The `QListView` class provides a default model/view implementation of a list/icon view. The `QListWidget` class provides a classic item-based list/icon view.



The `QTreeView` class provides a default model/view implementation of a tree view. The `QTreeWidget` class provides a classic item-based tree view.



The `QTableView` class provides a default model/view implementation of a table view. The `QTableWidget` class provides a classic item-based table view.

Qt : galerie de widgets

Display Widgets



The `QProgressBar` widget provides a horizontal progress bar.

Text Label

The `QLabel` widget provides a text or image display.



The `QLCDNumber` widget displays a number with LCD-like digits.

Input Widgets

One-line text editor

The `QLineEdit` widget is a one-line text editor.

Date editor

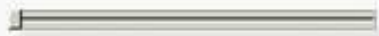
The `QDateEdit` class provides a widget for editing dates.

Time editor

The `QTimeEdit` class provides a widget for editing times.

Date and time editor

The `QDateTimeEdit` class provides a widget for editing dates and times.



The `QSlider` widget provides a vertical or horizontal slider.

Windows style

The `QComboBox` widget is a combined button and pop-up list.



The `QSpinBox` class provides a spin box widget.

Qt : galerie de widgets



The `QFontComboBox` widget is a specialized combobox that enables fonts to be selected from a pop-up list containing previews of available fonts.



The `QDoubleSpinBox` class provides a spin box widget that allows double precision floating point numbers to be entered.



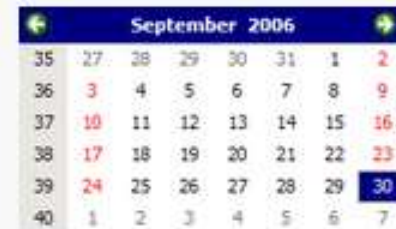
The `QScrollBar` widget provides a vertical or horizontal scroll bar. Here, we show a scroll bar with horizontal orientation.



The `QDial` class provides a rounded range control (like a speedometer or potentiometer).

The `QTextEdit` class provides a widget that is used to edit and display both plain and rich text. `QTextEdit` is an advanced WYSIWYG viewer/editor that can display images, links and tables.

The `QTextEdit` class provides a widget that is used to edit and display both plain and rich text.

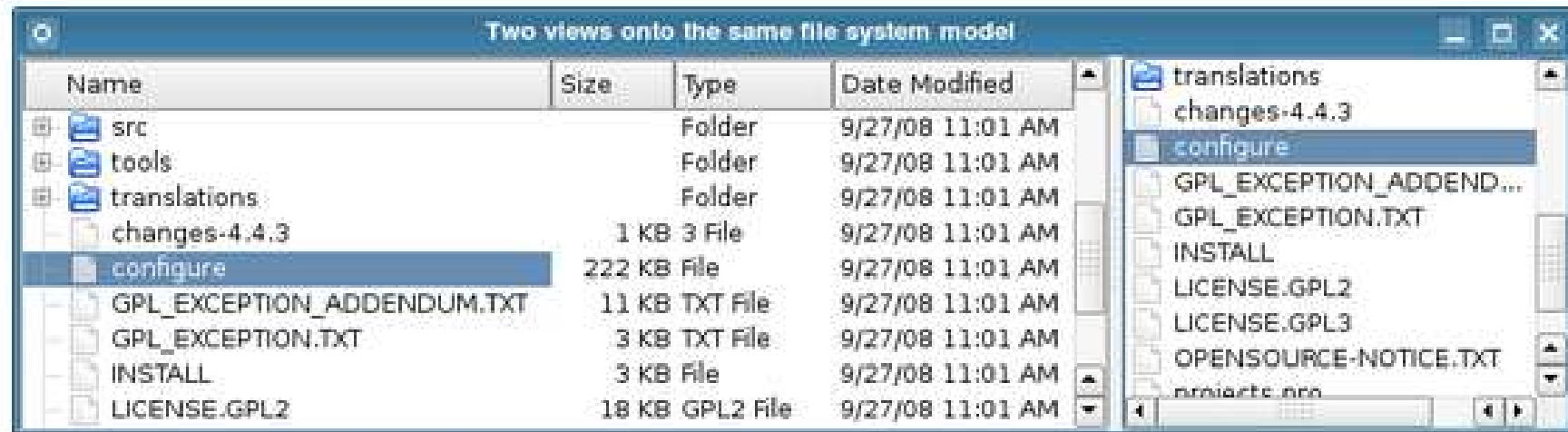


The `QCalendarWidget` class provides a monthly calendar widget that can be used to select dates.

Qt : file system

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QSplitter *splitter = new QSplitter;

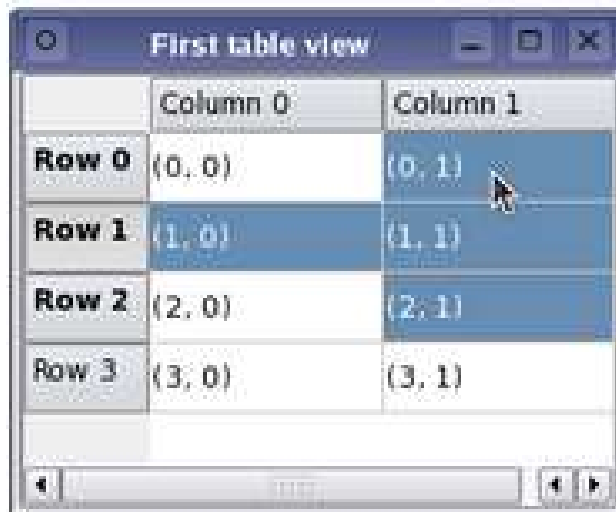
    QFileSystemModel *model = new QFileSystemModel;
    model->setRootPath(QDir::currentPath());
```



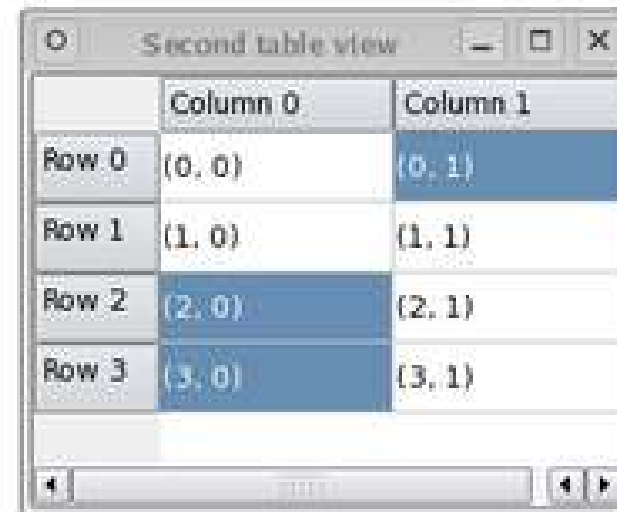
Qt : Model/View

- Un exemple

```
QTableView *firstTableView = new QTableView;  
QTableView *secondTableView = new QTableView;  
  
firstTableView->setModel(model);  
secondTableView->setModel(model);
```



	Column 0	Column 1
Row 0	(0, 0)	(0, 1)
Row 1	(1, 0)	(1, 1)
Row 2	(2, 0)	(2, 1)
Row 3	(3, 0)	(3, 1)



	Column 0	Column 1
Row 0	(0, 0)	(0, 1)
Row 1	(1, 0)	(1, 1)
Row 2	(2, 0)	(2, 1)
Row 3	(3, 0)	(3, 1)

Qt layout

- › A `QHBoxLayout` lays out widgets in a horizontal row, from left to right (or right to left for right-to-left languages).



```
QWidget *window = new QWidget;
QPushButton *button1 = new QPushButton("One");
QPushButton *button2 = new QPushButton("Two");
QPushButton *button3 = new QPushButton("Three");
QPushButton *button4 = new QPushButton("Four");
QPushButton *button5 = new QPushButton("Five");

QHBoxLayout *layout = new QHBoxLayout;
layout->addWidget(button1);
layout->addWidget(button2);
layout->addWidget(button3);
layout->addWidget(button4);
layout->addWidget(button5);

window->setLayout(layout);
window->show();
```


Qt layout

- › A **QHBoxLayout** lays out widgets in a horizontal row, from left to right (or right to left for right-to-left languages).



- › A **QVBoxLayout** lays out widgets in a vertical column, from top to bottom.



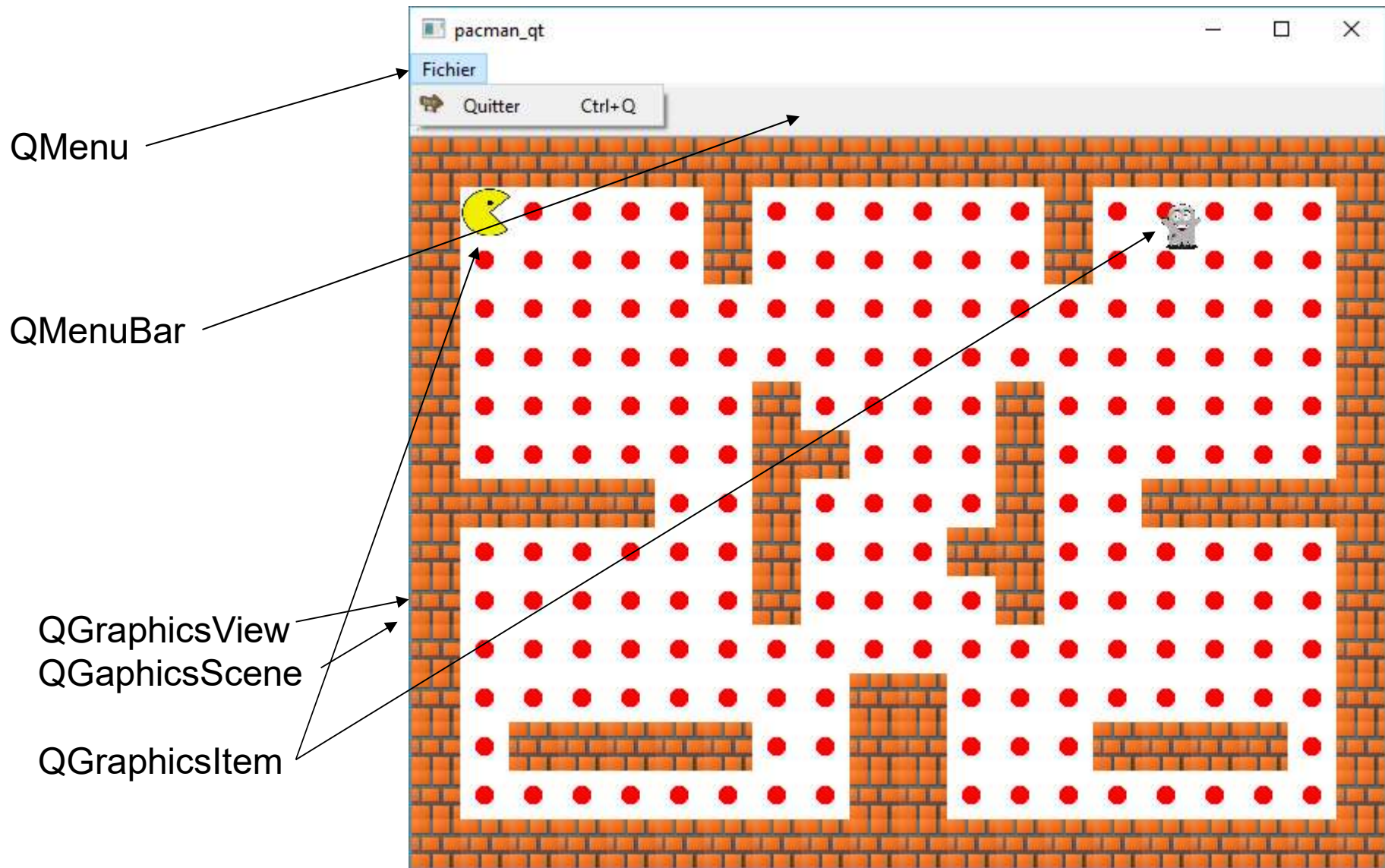
- › A **QGridLayout** lays out widgets in a two-dimensional grid. Widgets can occupy multiple cells.



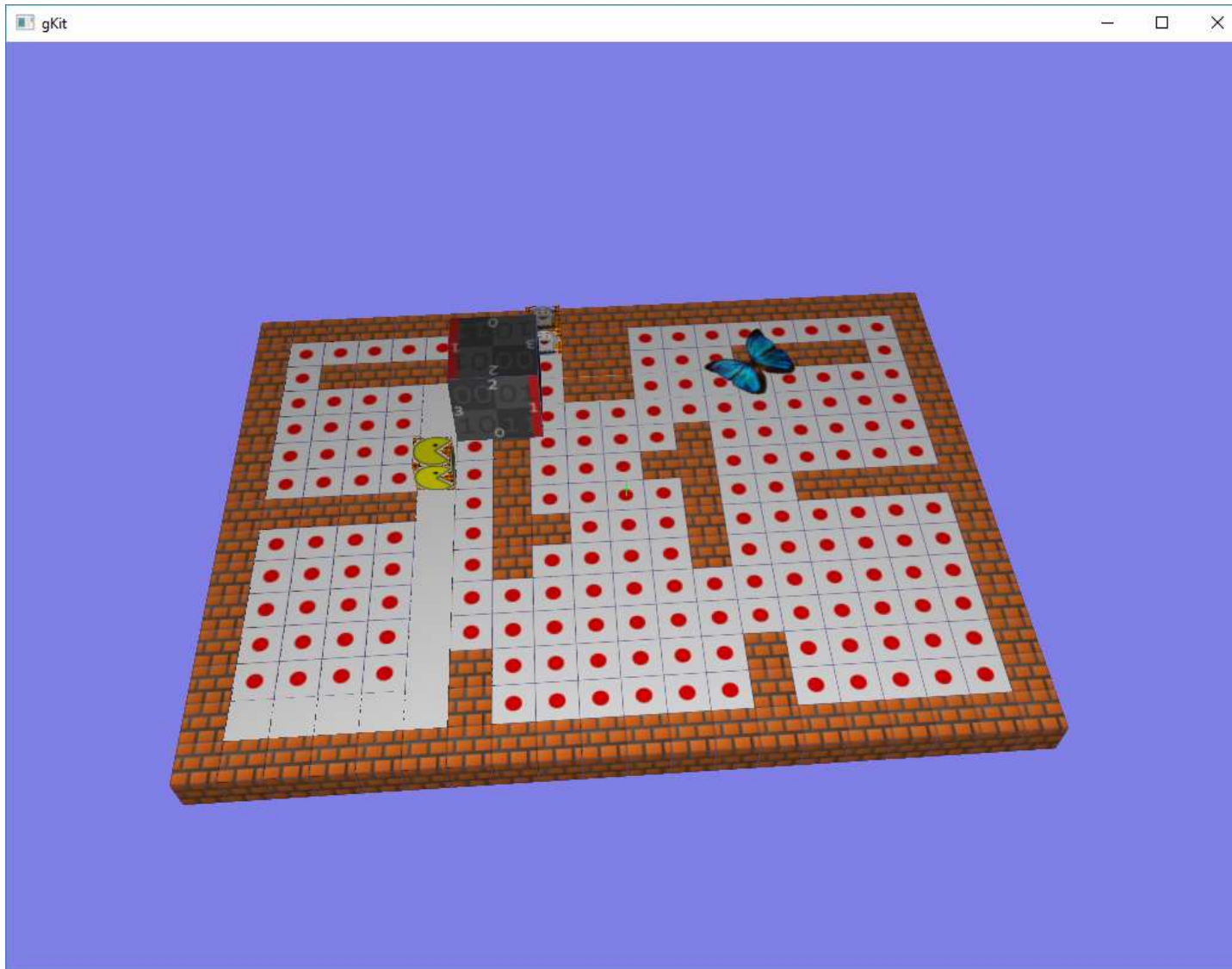
- › A **QFormLayout** lays out widgets in a 2-column descriptive label- field style.



Qt et Pacman : voir le code



OpenGL



Pour votre projet de LIFAP4 ...

Pour votre projet ...

Projet

- ~3 personnes
- 10 semaines
 - 40h de TP encadrés
 - avancer en dehors (par ex. les jeudi après-midi)
- Vous choisissez votre sujet
 - Soyez original, inspirer vous des jeux sur mobiles!
- Cahier des charges définit
 - Les fonctionnalités, les tâches et un 1^{er} diagramme des classes
 - Diagramme des classes sera à jour à la fin aussi
- Démo à rendre à mi-parcours

Pour votre projet ...

- Développement version alpha
 - version texte de votre application pour bien comprendre le cœur du problème
 - TXT est bien pour cela
 - Ne pas rester longtemps en texte
 - une fois les grandes lignes du projet comprises
 - ...

Pour votre projet ...

- Pour comprendre les lib d'interface graphique, le mieux est de pratiquer ... TP++!!
- Graphique : quelle librairie?
 - SDL2 : pour des jeux ou autres avec plutôt des images
 - Qt : pour les applications plus conventionnelles (menu, boîte de dialogue, etc.)
 - D'autres librairies existent ...

Exercices

- Ecrivez la/les boucles d'évènement qui gère en texte un menu appelant un jeu ayant lui-même une boucle d'évènement
- Par exemple un jeu de devinette d'un nombre :
 - L'ordinateur pense » à un nombre et répond à la question est-ce X? » par
Non, mon chiffre est plus grand » ou non, mon chiffre est plus petit » ou Gagné en N tentatives»

Le menu propose

- Lancer le jeu avec des nombre entre 1 et 10
- Lancer le jeu avec des nombre entre 1 et 100
- Lancer le jeu avec des nombre entre 1 et 1000
- Afficher le meilleur score
- Quitter