

Human-Computer Interaction

Eric Lecolinet - Télécom ParisTech
www.telecom-parist.fr/~elc

IGR203 / CSC_4IG05/6_TP

Introduction

- foundations, history

Software design (continuation of IG201)

- Statecharts, MVC, etc.
- Web user interfaces

User-centered design, human factors

- Principles, interviews, brainstorming, prototyping, evaluation
- Perception/action, pointing, colors, ergonomical principles

Hardware for HCI

- Electronics prototyping with Arduino

IGR203 / CSC_4IG05/6_TP

Grading

- 50% exams and labs (*will appear as CSC_4IG05_TP*)
- 50% group project (*as CSC_4IG06_TP*)

Resources

- **UE:** <https://perso.telecom-paristech.fr/elc/igr203/>
- **ecampus:** <https://ecampus.paris-saclay.fr/course/view.php?id=20444>
- **Team Project:** <https://perso.telecom-paristech.fr/eagan/class/igr203/>

IGR & IGD

The screenshot shows the official website for the IGR program at Telecom ParisTech. The page title is "Filière IGR 3D et Systèmes Interactifs". It features a grid of small images illustrating various applications of interaction design, such as VR headsets, mobile devices, and 3D models. Below the grid, there's a section titled "Descriptif" with detailed text about the program's goals and curriculum. A large orange arrow points from this section down to the "Option interne IGR & other Masters" text.

<http://www.telecom-paristech.fr/~elc/igr>

Troisième année

Plusieurs possibilités :

- 1. Suivre le [Master IGD](#) (Interaction, Graphics & Design) de l'Institut Polytechnique de Paris
 - L'entrée se fait alors au niveau M2
 - [Site IP Paris et inscriptions](#)
- 2. Suivre l'[Option interne IGR](#)
- 3. Candidater à un M2 associé :
 - [M2 IMA](#) : Image, Spécialité de la mention Informatique (Sorbonne Université)
 - [M2 HCI](#) : Interaction, Human Computer Interaction (Université Paris Saclay), Mention Informatique
 - [M2 MVA](#) : Parcours Mathématiques, Vision et Apprentissage (U. Paris Saclay), Mention Mathématiques et Applications
- 4. Suivre une formation équivalente à l'étranger :
 - Voir [cette page](#) et contacter [James Eagan](#) responsable mobilité internationale de la filière IGR

Il est également possible de choisir un cursus transverse (option entrepreneuriat) ou un des cursus alternatifs.

IGD Master

IGD - Interaction, Graphics & Design Track

Master of Computer Science - IP Paris



[Index](#) • All Courses • Interaction • Design • Mixed Reality • Graphics • Multimodal/Robotics • Programming • Projects • Schedule

Presentation

The IGD Master track is part of the [Computer Science Master](#) of Institut Polytechnique de Paris (IP Paris). Please see the [IGD IP Paris](#) page for a presentation and for application.

Welcome Session

The Welcome Session will take place on Monday September 7, 14:00-16:00 at Télécom Paris (room 0A214).

Courses & Schedules

- [Full list of courses](#)
- [Overall schedule](#)
- The list of chosen courses must first be validated by the student's tutor (see [Policies & Procedures](#) below).
- Course registration should be done in the institute offering the desired course (e.g. Télécom Paris for TP-xxx courses).

Institutional partners

The IGD Master offers a comprehensive range of courses which are held by the following institutes:

- TP = Télécom Paris
- X = École Polytechnique
- TSP = Télécom SudParis
- ENSTA = École Nationale Supérieure des Techniques Avancées
- HCI = Master HCI - Université Paris-Saclay

All these institutes are located on Plateau de Saclay. Maps: [IP Paris - Paris-Saclay](#)

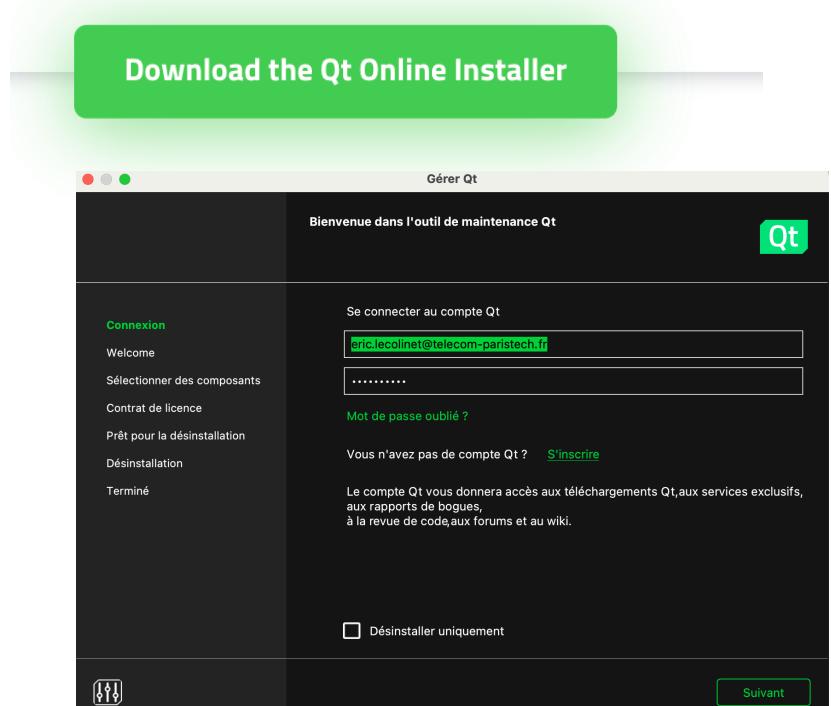
Policies & Procedures

Note on Qt (for first labs)

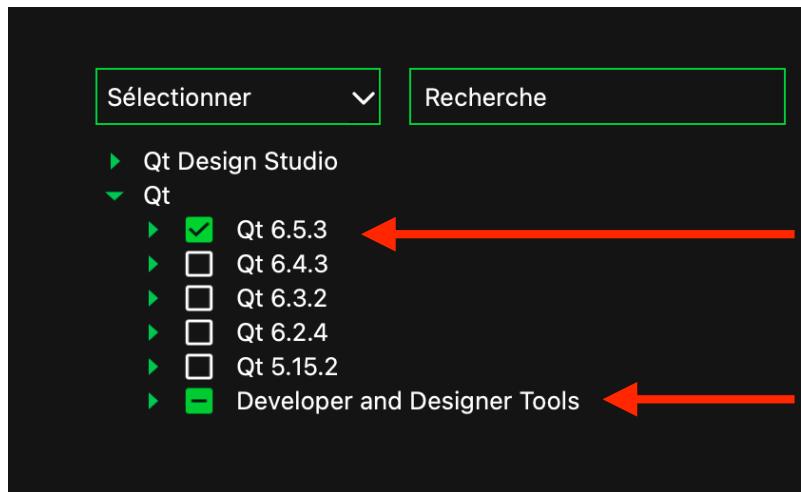
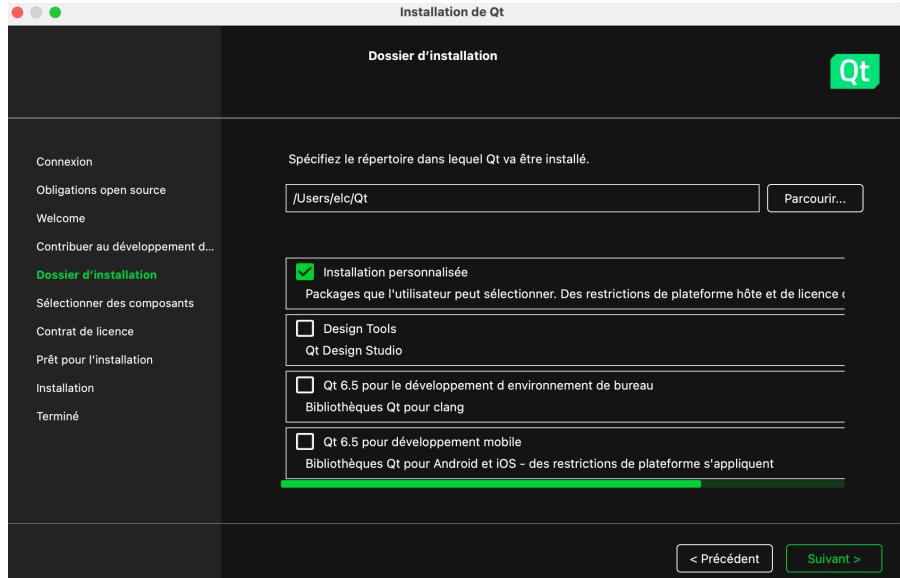
- **Labs:** <http://www.telecom-paristech.fr/~elc/qt>
- **Qt Web site:** <http://www.qt.io/>
- **Documentation:** <http://doc.qt.io/qt-6/>

- Installation (Open Source Version):

- In <https://www.qt.io/download-open-source>
 - ▶ click on "Download the Qt Online Installer"
 - ▶ download Installer for your OS
- Open Qt **MaintenanceTool** app

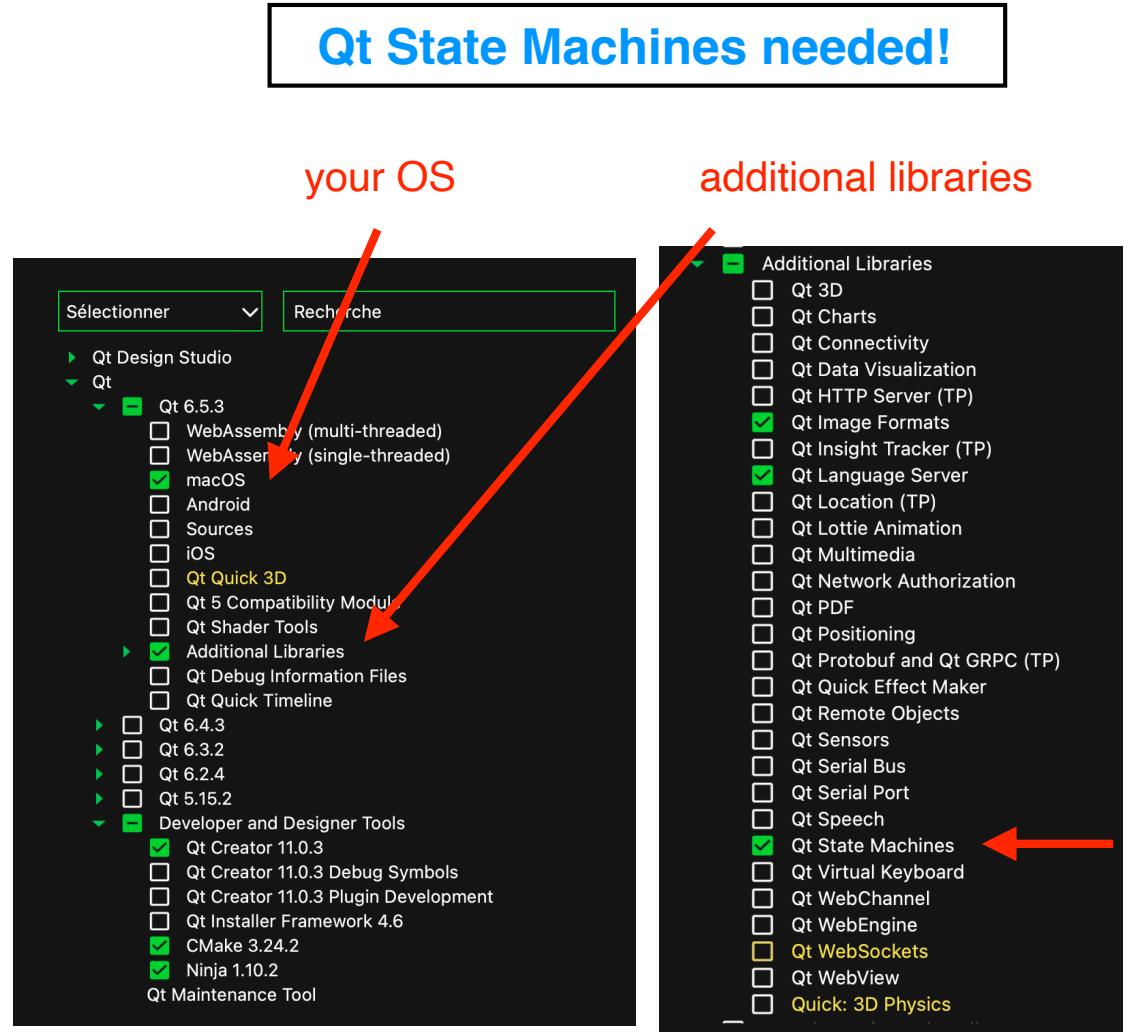


Install (MaintenanceTool)



Qt State Machines needed!

your OS additional libraries



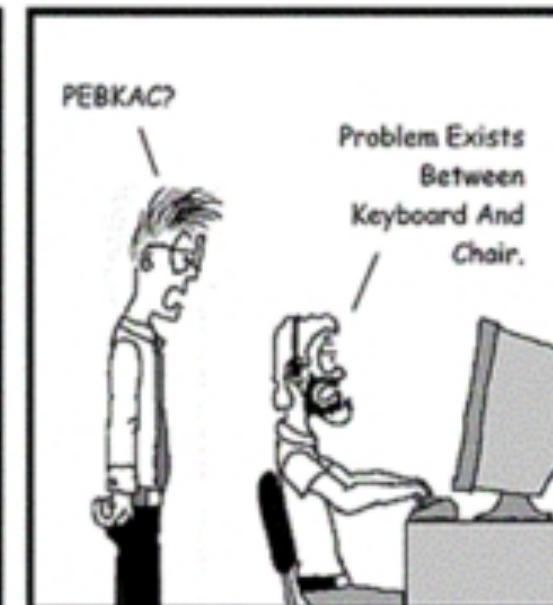
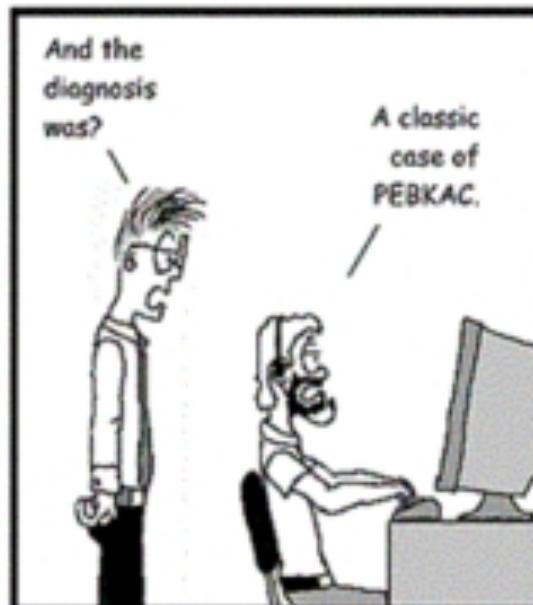
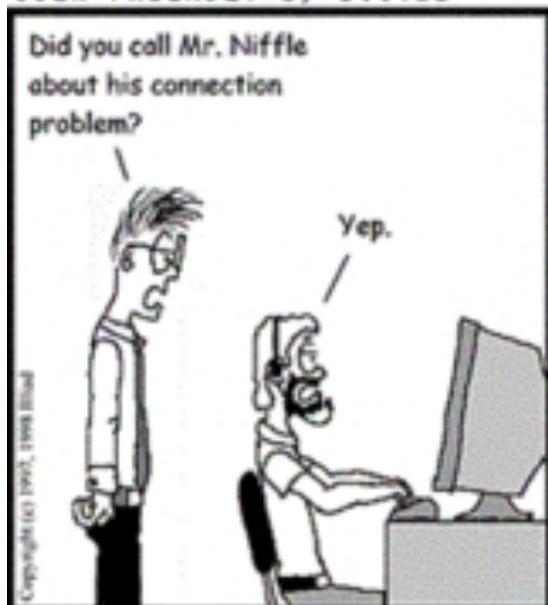
The screenshot shows the 'Qt Maintenance Tool' interface with a sidebar for 'Sélectionner' and a 'Recherche' search bar. The main area displays a tree view of Qt components. A red arrow points from the checked 'macOS' under 'Qt 6.5.3' to the 'Additional Libraries' section on the right, which contains a list of modules. Another red arrow points from the checked 'Qt State Machines' under 'Additional Libraries' back to the 'Qt 6.5.3' section in the tree view. The 'Additional Libraries' list includes: Qt 3D, Qt Charts, Qt Connectivity, Qt Data Visualization, Qt HTTP Server (TP), Qt Image Formats, Qt Insight Tracker (TP), Qt Language Server, Qt Location (TP), Qt Lottie Animation, Qt Multimedia, Qt Network Authorization, Qt PDF, Qt Positioning, Qt Protobuf and Qt GRPC (TP), Qt Quick Effect Maker, Qt Remote Objects, Qt Sensors, Qt Serial Bus, Qt Serial Port, Qt Speech, and Qt State Machines.

Human-Computer Interaction

HCI =

- Design
- Implementation of **interactive** systems... used by **human users!**
- Evaluation

USER FRIENDLY by Illiad



THE PEBKAC PROBLEM...

Human-Computer Interaction

Multi-disciplinary domain

Engineering

- computer science, software development, GUI toolkits, AI & machine learning...
- electronics, hardware, material science...

Psychology, ergonomics, design

- ergonomics, cognitive psychology, physiology...
- sociology...
- industrial design, graphic arts, typography...

Human-Computer Interaction

Goals

1) **Usability:** reliable, easy-to-use, efficient interactive systems

- learning cost

Criteria:

- ability to use advanced functionalities
- speed of execution and fatigue (cf. repetitive actions)
- errors (especially for critical interfaces, but not only)
- user satisfaction

Human-Computer Interaction

Goals

1) Usability: reliable, easy-to-use, efficient interactive systems

2) Reasonable development cost

GUI development tends to take a huge amount of time!

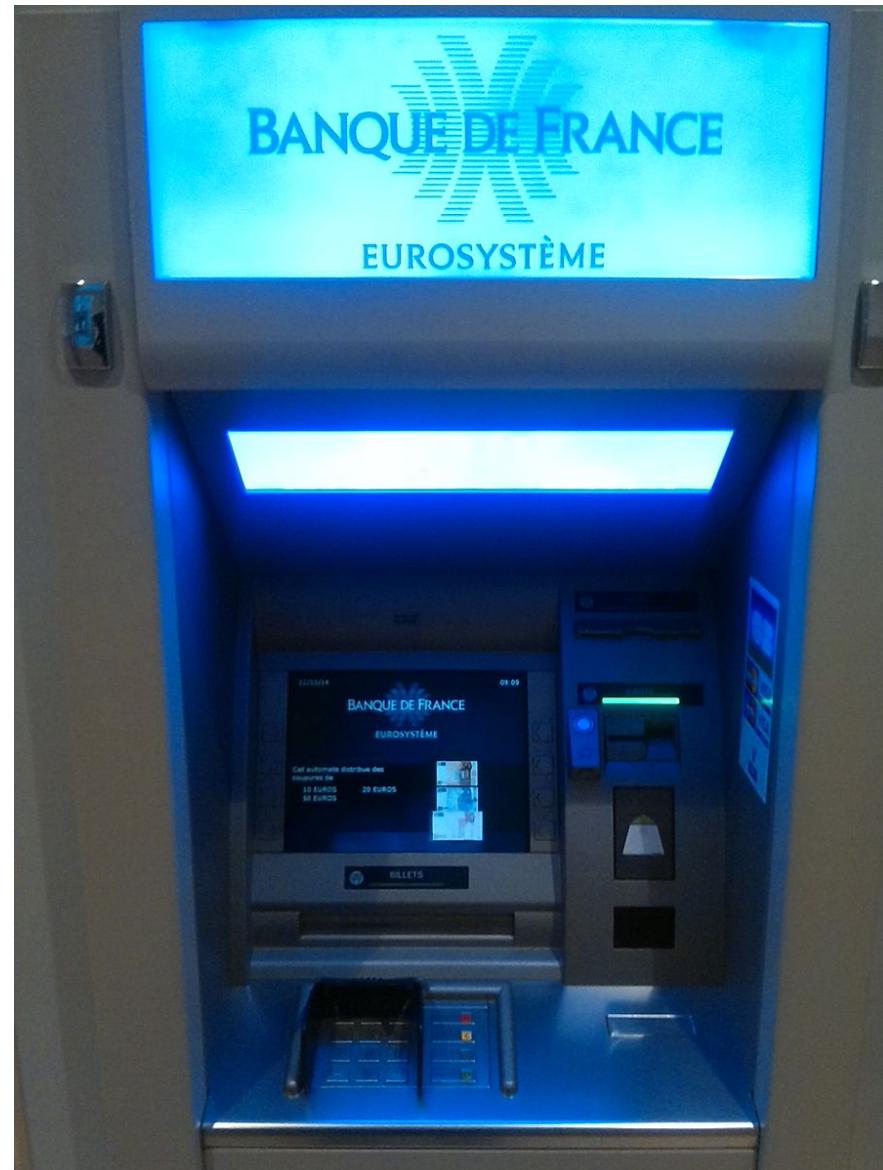
Don't think system first!

=> Dedicated **design methodologies**, user-centric design

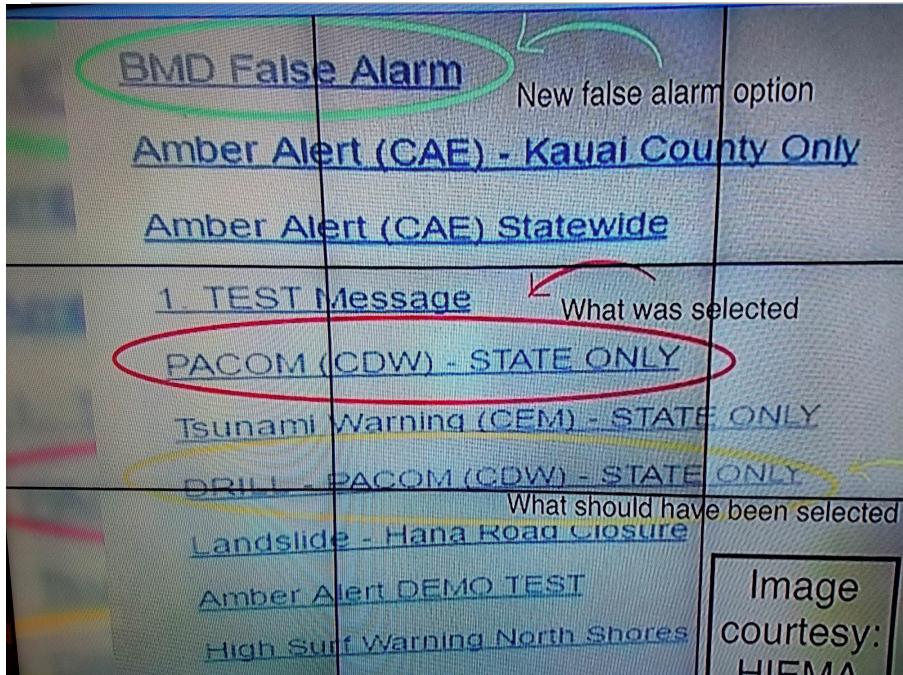
Examples

ATM Machine

- Money then card?
- Card then money?



Examples



False Alarm in Hawaii could have caused
Nuclear War (JDD - Jan. 15, 2018)

A €85 billion typo
at Samsung's brokerage
(les Echos - April 11, 2018)



Examples (critical interfaces)

Mont Saint Odile disaster

- Crash of an **Airbus A-320** on January 20, 1992
- 87 dead, 9 survivors
- Charged in January 1997 for "homicide and involuntary injuries" in relation to "**the ergonomics of the descent mode of the aircraft**"

Examples (critical interfaces)

Mont Saint Odile disaster

- The pilot confused the **selected descent mode**
- He entered **33** believing he was in **FPA mode**,
but he was in **VS mode!**
 - **FPA** mode => descent **angle**
 - **VS** mode => vertical **speed**
 - **33** can either mean **3.3°** or **3300 fpm**!

=> VSpeed = **3300 fpm** (16.7 m/s) about **4 times** the normal speed!



Cockpit comparison



Boeing 747 (in the 90')



Airbus A330

Cockpit comparison



- New **type** of cockpit: new possibilities involve new errors!
- No physical **visual cues**, different infos can be displayed at the same place
- Poor visibility of **modes**, no **units** (33 can mean 3.3° or 3300 fpm)

Previous user knowledge

The user may understand something else!

I'm an IT specialist on a hotline and I get a call:

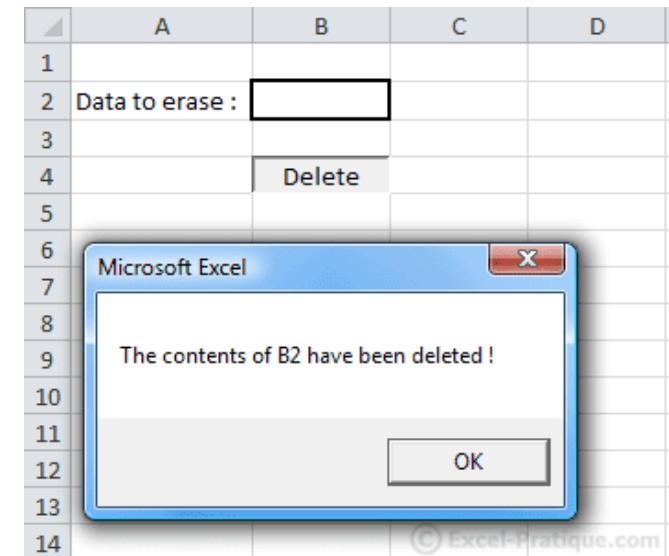
"Sir, just to be sure, if I put the seagull in the box, does that mean I won't have to retype my password?"



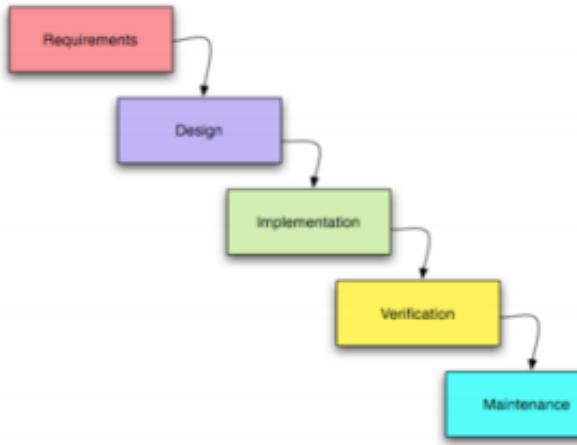
... Seagull in the box = check the box

!

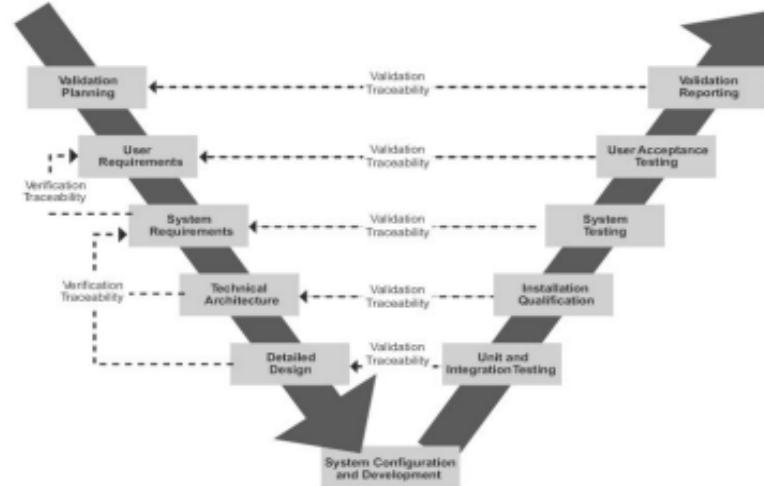
Why OK rather than Do It on dialog boxes?



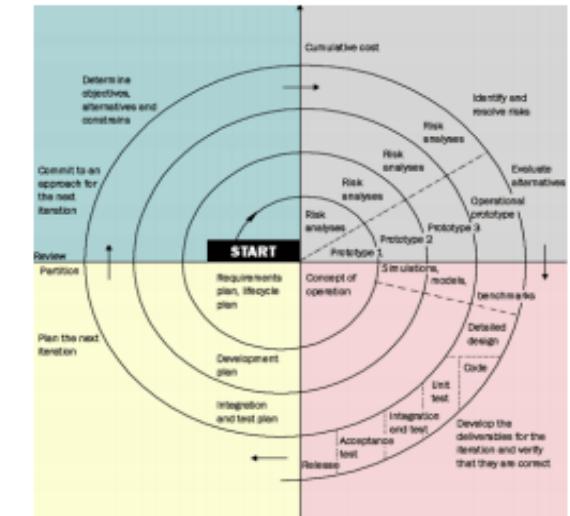
Traditional design models



Waterfall



V



Spiral

Traditional development models:

- involve users at the **beginning** and the **end** of the design process
 - => can't take **human factors** into account effectively

User-centred design



- **iterative design**, trial and error approach
- the users are the **domain experts** => **active role** in the design process
- **mock-ups** in first iterations

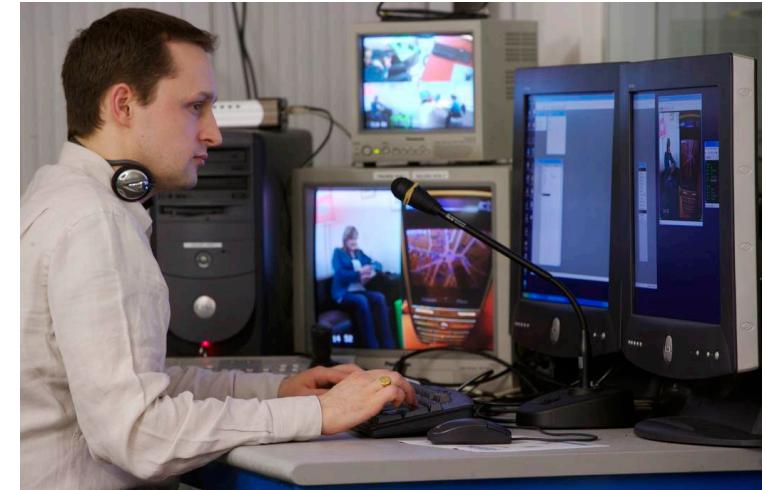
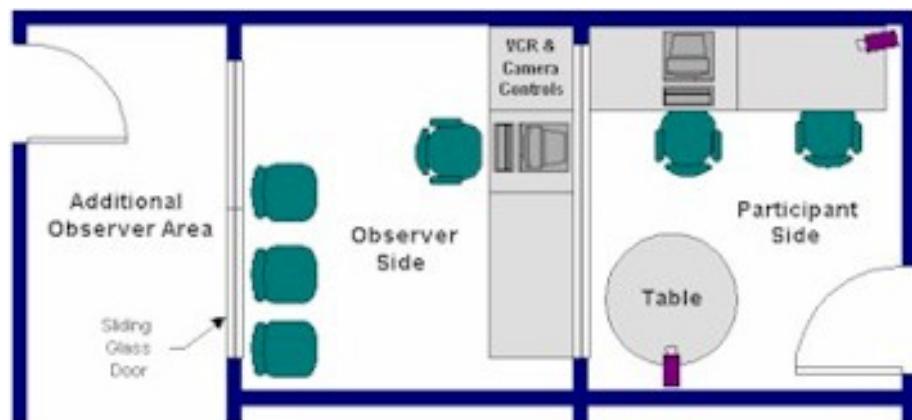
User-centred design

jour	tâche prévue
vendredi	vérification de l'état des features à tester
lundi	développement des tâches de test de la semaine
mardi	reception du prototype. pre-test en interne du proto pour en déterminer les limites
mercredi	test avec sujets les responsables développeurs participent à l'observation
jeudi / ven	analyse des tests, recommandations, synthèses

Example: Word 6 development

- over several months
- 43 user testings / more than 250 participants

Usability Labs

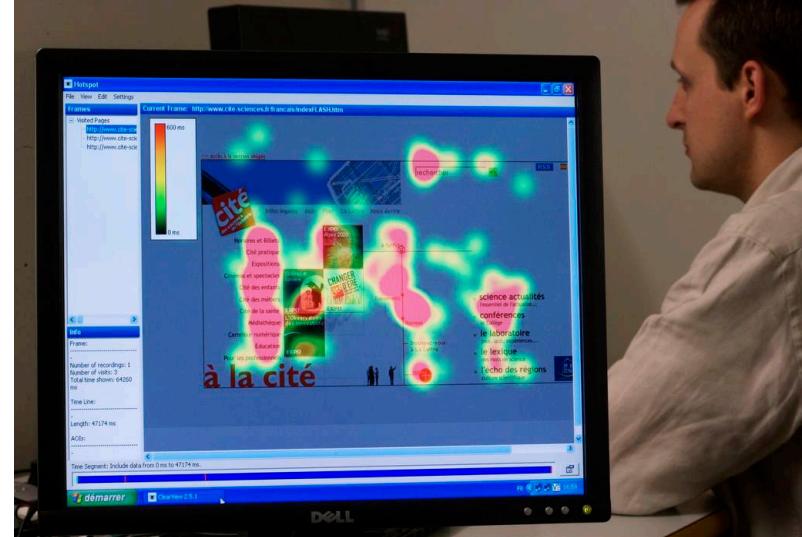


LUTIN Lab (La Villette)

Usability Labs

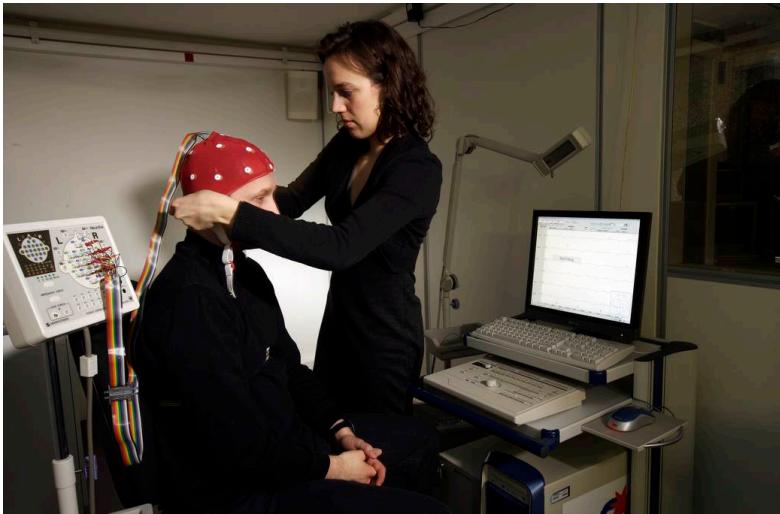


Eye tracking



Heat map

LUTIN Lab (La Villette)



Physiological measures

Interaction styles

Evolution

système conversationnel (shell)	fenêtres (minitel!)	manipulation directe (WIMP)	réalité virtuelle (ou augmentée)
------------------------------------	------------------------	--------------------------------	-------------------------------------



Conversational interaction

- **System-imposed**
- Typical of command languages
 - ex: Unix Shell
- Only for **expert users**

```
drwxr-xr-x  8 elc  elc   272 May  4  2005 Music/
lrwxr-x---  1 elc  elc    9 Feb 22  2006 PHOTOS@ -> Pictures/
lrwxr-x---  1 elc  elc   34 Feb 22  2006 PRESENTATIONS@ -> Desktop/Dem
ampus/Presentations
drwxr-xr-x  28 elc  elc   952 Jun 26 15:03 Papers/
drwxr-xr-x  19 elc  elc   646 Nov 14  2005 Papers.ex/
drwx----- 22 elc  elc   748 Jul 30 13:50 Pictures/
drwxr-xr-x  9 elc  elc   306 May  4  2005 Public/
drwxr-xr-x  7 elc  elc   238 May  4  2005 Sites/
drwx----- 13 elc  elc   442 Jul 30 13:47 Videos/
-rw-r--r--  1 elc  elc    58 May 30  2006 a.sh*
drwxr-x---  8 elc  elc   272 Feb 14  2007 arch/
drwxr-xr-x  18 elc  elc   612 Sep 30  2005 bin/
lrwxr-x---  1 elc  elc    7 Jun 21  2005 desk@ -> Desktop
-rw-r--r--  1 elc  elc  1211 Jun  9  2005 hosts.html
drwxr-x--- 12 elc  elc   408 May 12  2006 images/
-rw-r--r--  1 root  wheel  960 Apr 14  2004 open-x11*
drwxr-x--- 32 elc  elc  1088 Sep 25 00:35 ubit/
drwxr-x--- 41 elc  elc  1394 Aug 18 00:02 ubit-5.0.0/
alias elc ()
```

Interaction styles

Forms

- System-imposed, but with **hints** (labels)
- Limited interaction
- Web 1.0, Minitel...

The screenshot shows a search interface titled "Activité". At the top, there are navigation links: "LES PAGES JAUNES", "LES PAGES BLANCHES", "LES PAGES MARQUES", "LES RUES COMMERCANTES", "LES PAGES WEB", and "LES PAGES TOURISME". Below the title, there are several input fields:

- A dropdown menu labeled "Avec Web" with "Tous les professionnels" selected.
- An input field for "Nom" (Name) with a placeholder "Prénom" (First name).
- A checkbox labeled "Commencant par" (Starting with) with an input field next to it.
- An input field for "Adresse" (Address) with a placeholder "Rue" (Street).
- An input field for "Localité" (Location) with a placeholder "Ville" (City).
- An input field for "Département" (Department) with a placeholder "Code postal" (Post code).

A large blue "RECHERCHER" (Search) button is located at the bottom right of the form area.

The screenshot shows a social networking interface titled "3615 FACEBOOK". At the top, there are links for "mon compte" and "Guide".

Amis

- Fab ParseError
- Bufford Molosse
- Maurice Art&Co
- Ufunk Kelon

[voir mes amis](#)

Mon Statut

[valider le statut](#) [Envoyer](#)
[annuler](#) [Correc.](#)

Actualités

- Bufford Molosse :
Dur retour de soirée... La semaine va être difficile !
- Ufunk Kelon :
A vu des screenshots du prochain Windows 2.10 !
- Maurice Art&Co :
Dimanche = Journée relax ;)

par UFUNK.net

[page suivante](#) [Suite](#)
[page précédente](#) [Retour](#)

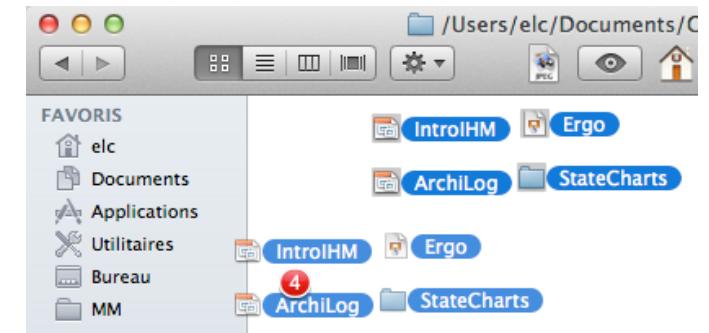
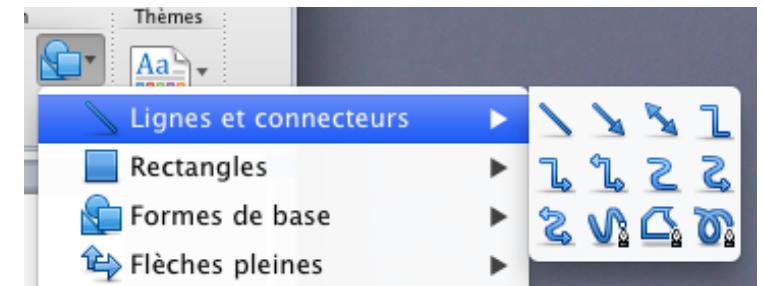
Interaction styles

WIMP

- **Window, Icon, Menu, Pointer**
- **Permanent representation (metaphors)**
- **Direct manipulation:**
fast, incremental, reversible operations
- **Immediate feedback**

Goals

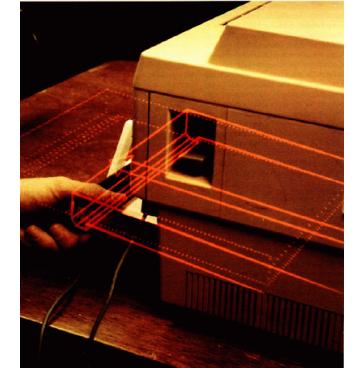
- **Exploratory learning**
- **Transfer** of experience by analogy



Interaction styles

Post-WIMP

- Touch / Mobile interaction
- Multimodal interaction (Touch, Voice, AI...)
- Virtual / Augmented / Mixed Reality
- Tangible/Physical interfaces / Smart Materials
- Robotics / Augmented Human
- etc.



Conferences and journals

Conferences

- ACM CHI
- https://www.youtube.com/watch?v=owjYbVI9eGQ&ab_channel=ACMSIGCHI
- ACM UIST
- IHM (AFIHM: www.afiham.org)
- AVI
- INTERACT
- IEEE Visualization

Journals

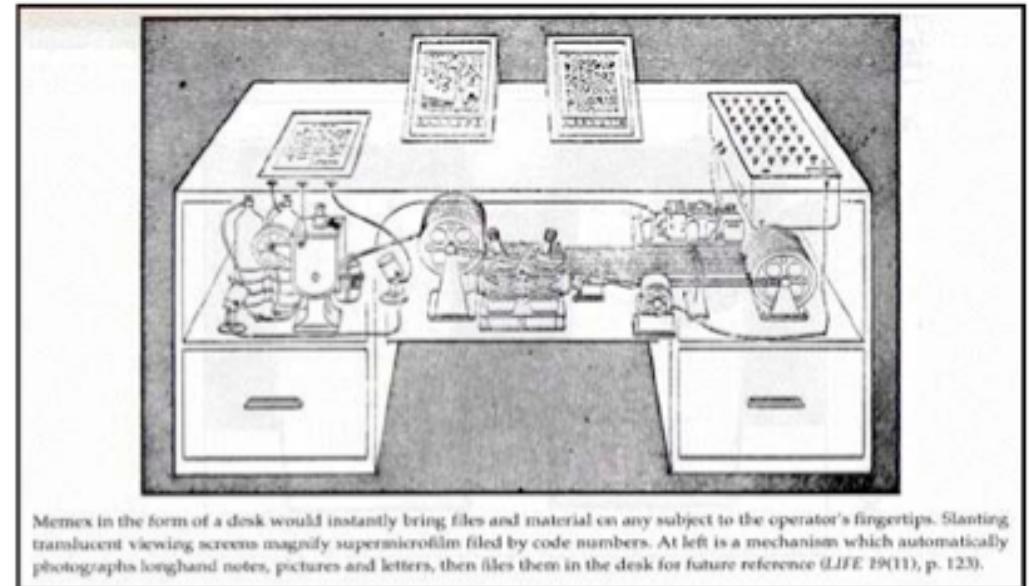
- ACM Interaction
- ACM ToCHI



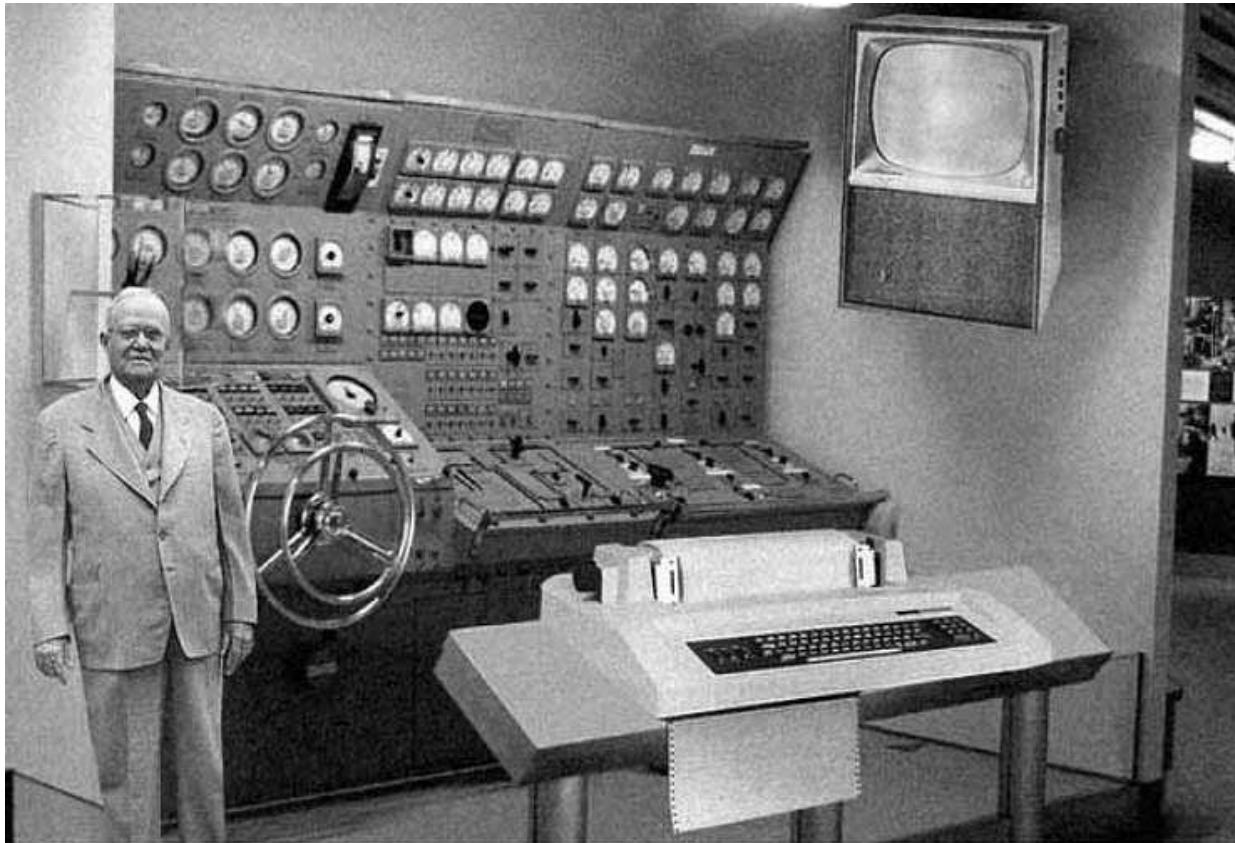
A brief history of HCI

1945: Memex (Vannevar Bush)

- idea of an "external memory"
- **hypertext** concept
 - keywords, references, indexing, links, annotation...
- tech not ready (microfilms!)



A brief history of HCI



Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 50 years from now scientific progress is expected to solve these problems. With teletype interface and the Fortran language, the computer will be easy to use.

1954 : A vision of the "home computer in 2004"



1956: a 5 Mb disk

A brief history of HCI

1963: Sketchpad (Ivan Sutherland MIT)

First drawing software with **icons**

- **direct manipulation** (optical pen)
- **high resolution** screen
(oscilloscope)
- OO programming



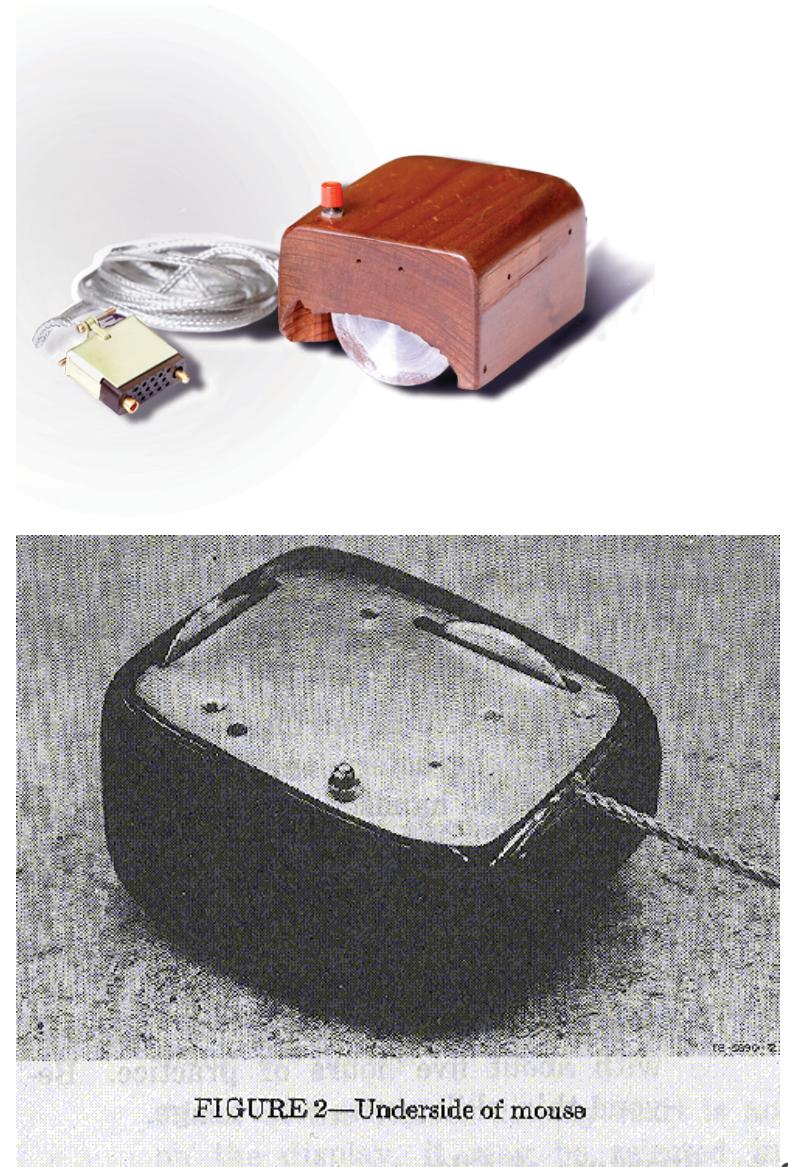
A brief history of HCI



A brief history of HCI

1962+ : Douglas Engelbart

- Invented the **mouse!**
- High resolution screen
- Word processing
- Hypertext and hypermedia
- Electronic messaging
- Videoconference
- Groupware



A brief history of HCI

Xerox PARC (Palo Alto, CA)

- Research center founded in 1970
- Office automation...

Alain Kay

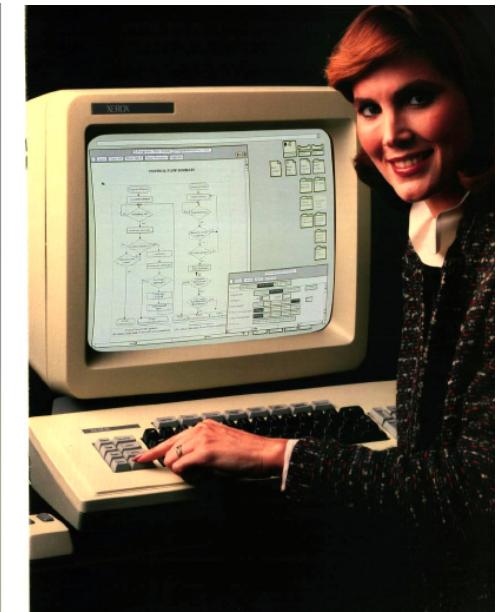
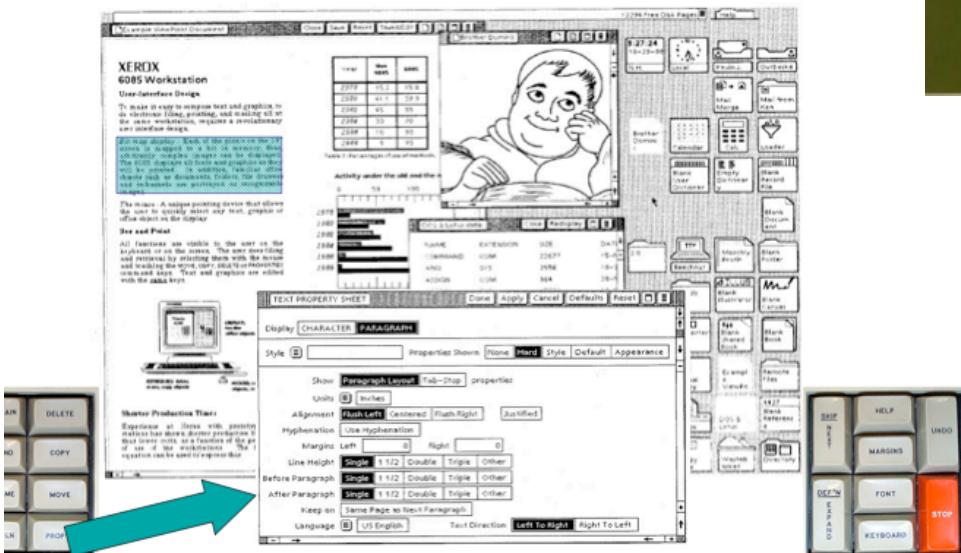
- One of the founders of the PARC
- **Personal Computer** concept in 1969
- A radical vision compared to what existed at that time!



A brief history of HCI

1981: Xerox Star

- First commercial workstation
- Advanced graphical environment
- GUI, "desktop metaphor"
- Very expensive, commercial failure



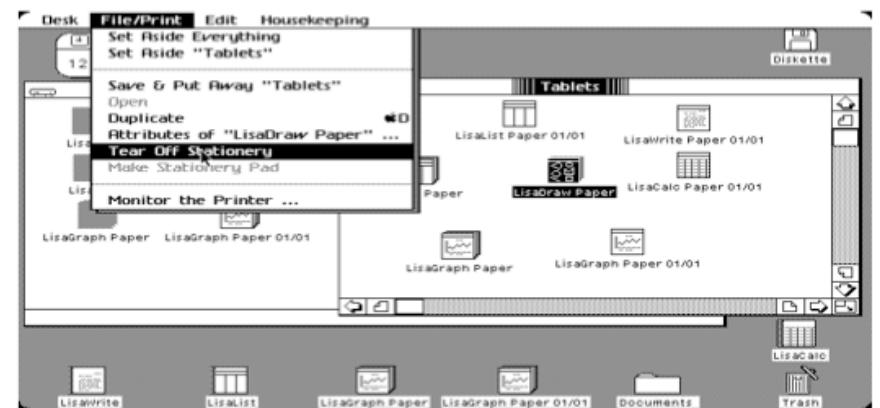
A brief history of HCI

1984: Apple Macintosh

- Inspired from PARC research
- Mature technology, open architecture
- **WYSIWYG**, laser printing
- "Reasonably" priced (\$ 2500)
- **Style guides** (consistency)



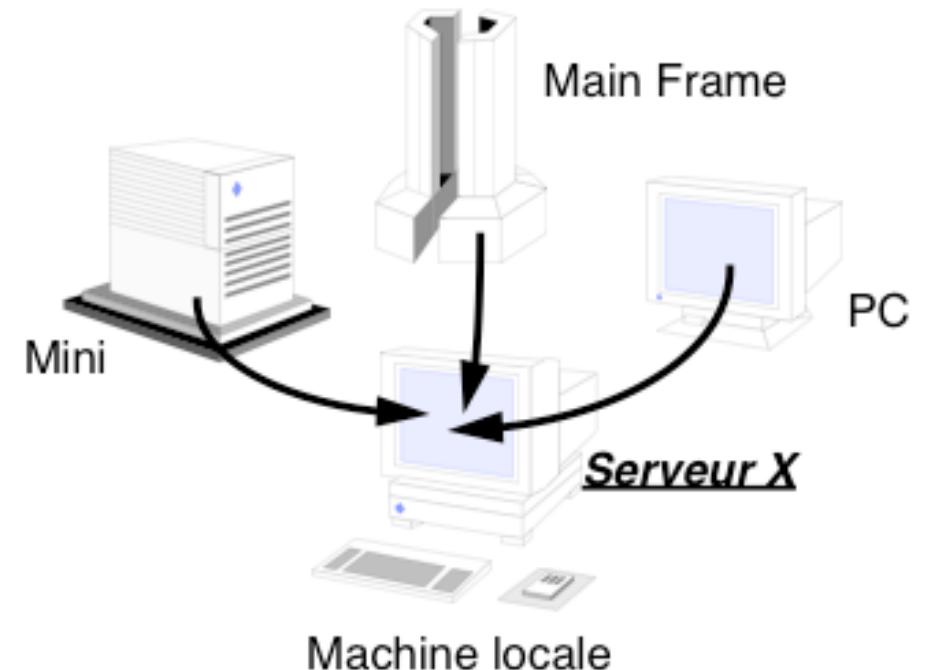
Succesor of **Lisa**,
another commercial failure



A brief history of HCI

1985: X Window System (MIT)

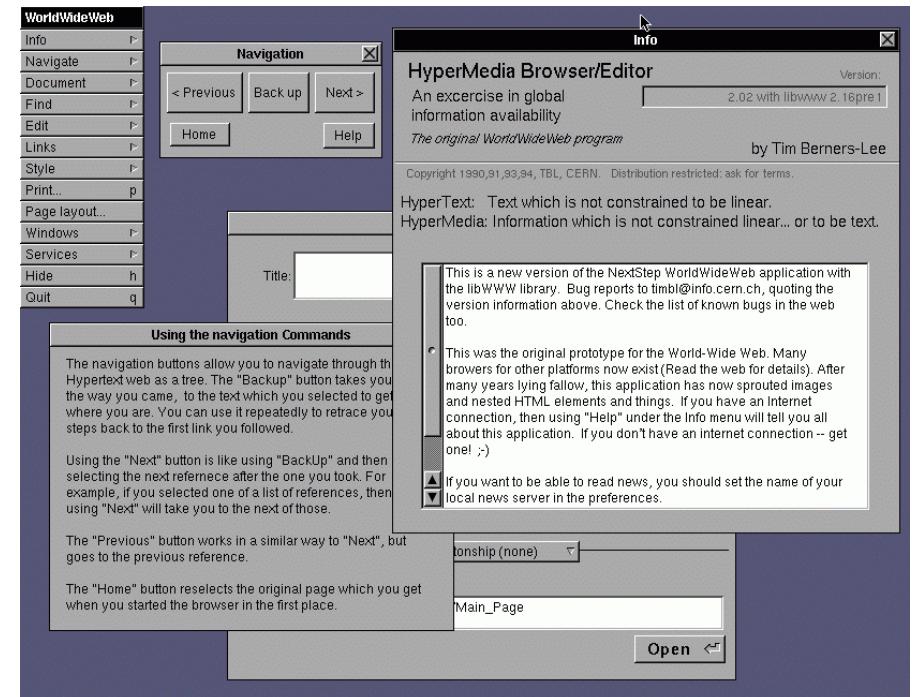
- For high-end workstations
- **Multi-platform**
- Hardware and software independent
- **Networked:** client / server architecture
- **Still used** nowadays! (Linux)



A brief history of HCI

1990/1991: World Wide Web (Tim Berners-Lee - CERN)

- Networked hypertext model
- **Internet + hypertext**
 - creation of URL / HTTP / HTML
- Paper refused at the prestigious ACM Hypertext conference!
- Small project that changed the world!



- Small project that changed the world!

source: Wikipedia

HCI Software Development

Eric Lecolinet - Télécom ParisTech
www.telecom-parist.fr/~elc

Components of a GUI

Components of a Graphical User Interface (GUI)

- **Structure**: relationships between graphical objects (*e.g. HTML*)
- **Presentation**: graphical attributes (*e.g. CSS*)
- **Data** (*e.g. text, table content*)
- **Interaction**: **behavior of the GUI** : Most difficult part!

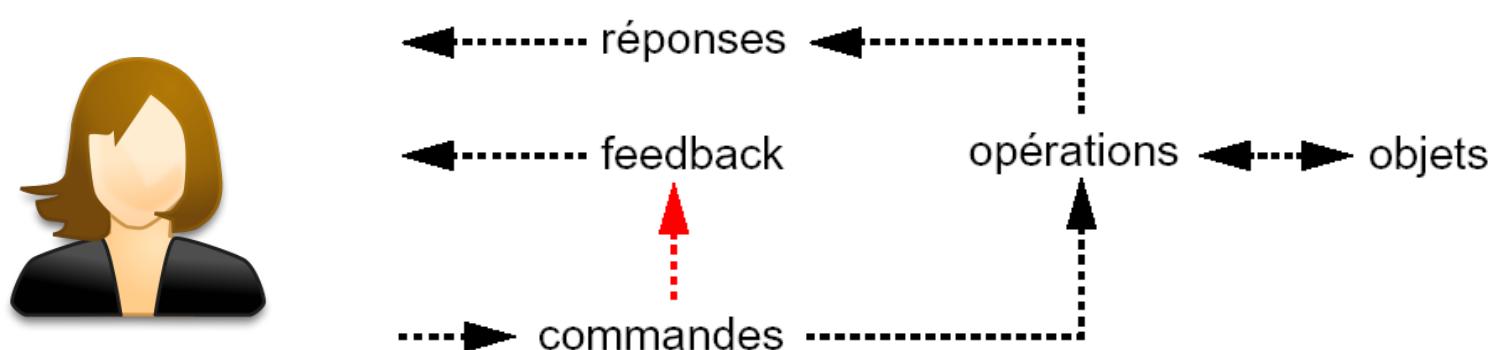
Specifying + encoding interaction

Specification

- How to **specify** the behavior in a **concise** and **unambiguous** manner?

Encoding

- How to **implement** it in a reliable and scalable way



src: M. Beaudoin Lafon

Specification

Specification languages

Example:

UAN : User Action Notations

(R. Hartson et al.)

Table I. Summary of Some Useful UAN Symbols

Action	Meaning
~	move the cursor
[X]	the context of object X, the “handle” by which X is manipulated
~[X]	move cursor into context of object X
~[x, y]	move the cursor to (arbitrary) point x, y outside any object
~[x, y in A]	move the cursor to (arbitrary) point within object A
~[X in Y]	move to object X within object Y (e.g., [OK_icon in dialogue_box])
[X]~	move cursor out of context of object X
v	depress
^	release
Xv	depress button, key, or switch called X
X^	release button, key, or switch X
X^v	idiom for clicking button, key, or switch X
X"abc"	enter literal string, abc, via device X
X (xyz)	enter value for variable xyz via device X
()	grouping mechanism
*	iterative closure, task is performed zero or more times
+	task is performed one or more times
{ }	enclosed task is optional (performed zero or one time)
A B	sequence; perform A, then B (same if A and B are on separate, but adjacent, lines)
OR	disjunction, choice of tasks (used to show alternative ways to perform a task)
&	order independence; connected tasks must all be performed, but relative order is immaterial
↔	interleavability; performance of connected tasks can be interleaved in time
	concurrency; connected tasks can be performed simultaneously
;	task interrupt symbol; used to indicate that user may interrupt the current task at this point (the effect of this interrupt is specified as well, otherwise it is undefined, i.e., as though the user never performed the previous actions)
forall	for all
:	separator between condition and action or feedback

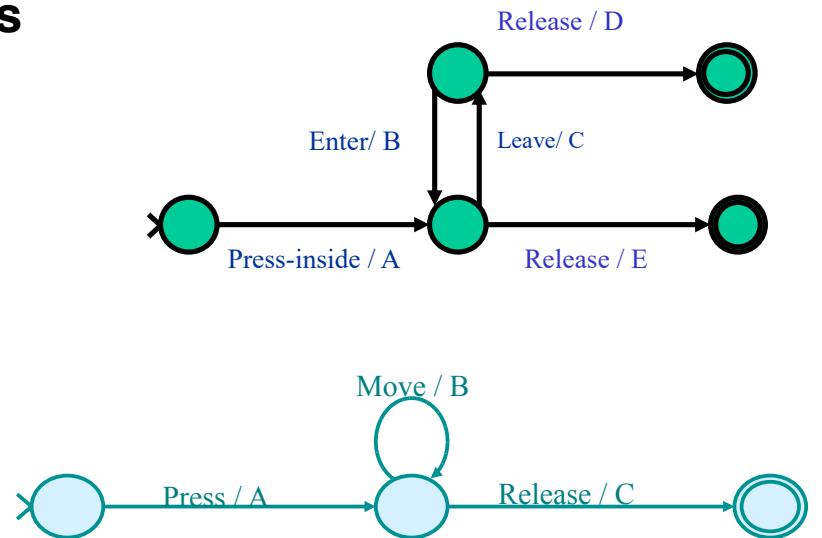
Table 1: UAN specification of the task “select icon”

TASK: select file		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
~[file] M^	file ! forall file' ≠ file : file' -!	
Mv		selected = file

Finite state machines (FSM)

Finite state machines

- Simple and effective for **modeling behaviors**
- Avoid "**spaghetti**" of callback functions
- Compatible with **event-driven** management
(as GUI toolkits)



Double benefit

- Graphical **representation** => **GUI specification**
- Can **generate** source code => **GUI implementation**

State machine example

Rubber banding

Accept press event and get P1;

P2 = P1;

Draw line P1-P2;

Repeat

Erase line P1-P2;

P2 = current_position();

Draw line P1-P2;

Until release event;

Act on line input;



(adapted from Scott Hudson - CMU)

Problem?

State machine example

Rubber banding

Accept press event and get P1;

P2 = P1;

Draw line P1-P2;

Repeat

Erase line P1-P2;

P2 = current_position();

Draw line P1-P2;

Until release event;

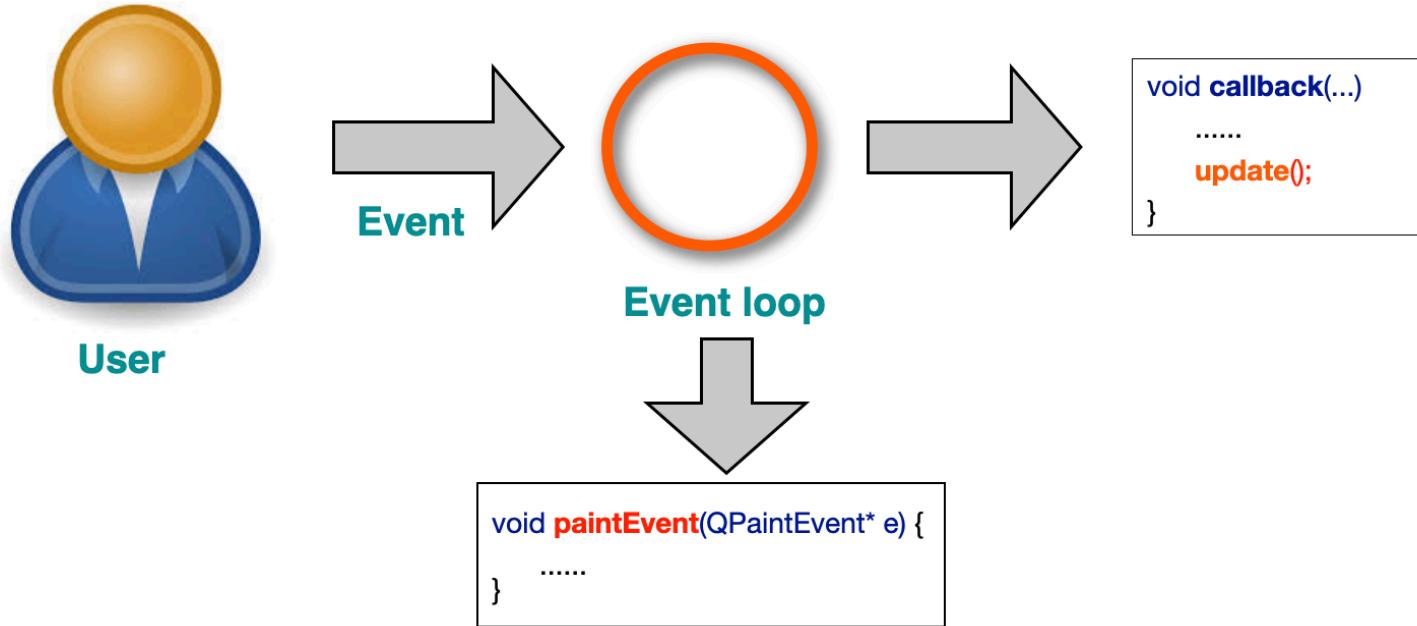
Act on line input;



Problem:

not compatible with **event-driven management!**

Event-driven management



Events are provided by the event loop

- We cannot "*repeat an action*" in a loop
- The event loop must not be **blocked**
- Other events should not be **ignored**

State machine example

Accept press event and get P1;

P2 = P1;

Draw line P1-P2;

A

Repeat

Erase line P1-P2;

P2 = current_position();

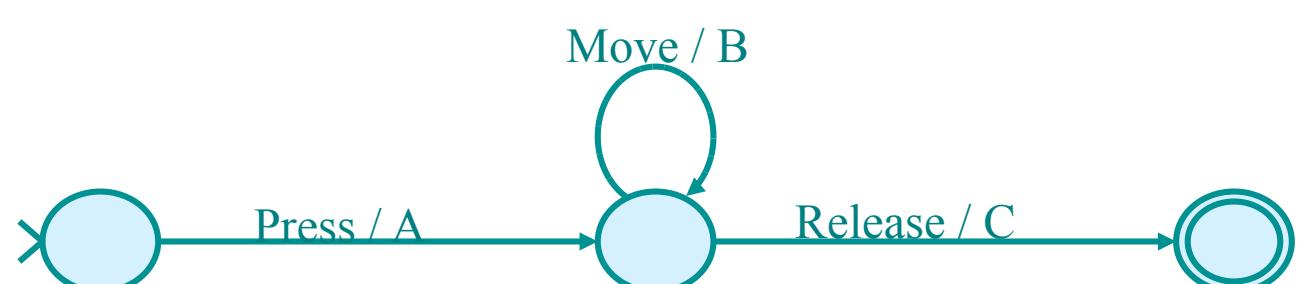
Draw line P1-P2;

B

Until release event;

Act on line input;

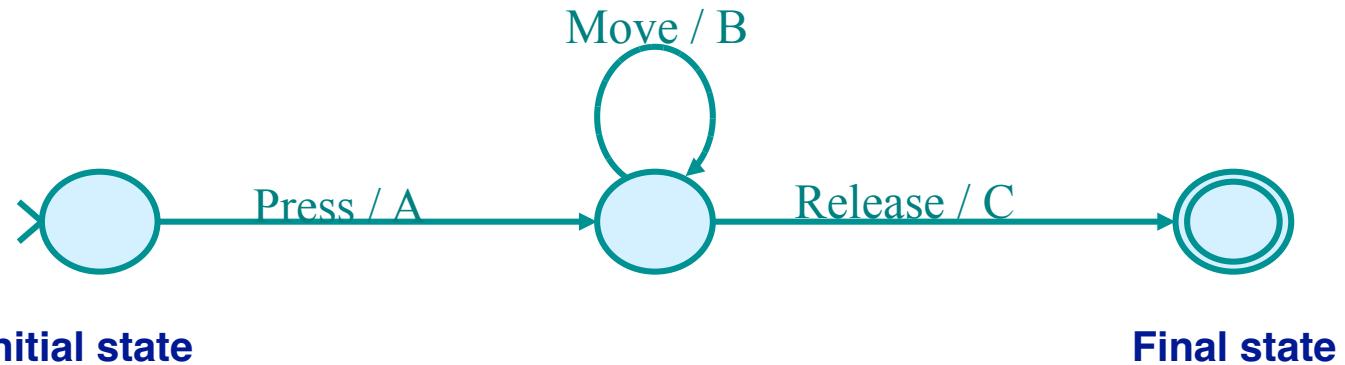
C



States and transitions

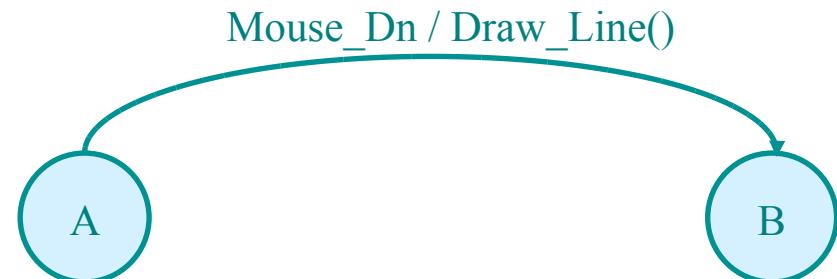
States

- Circles

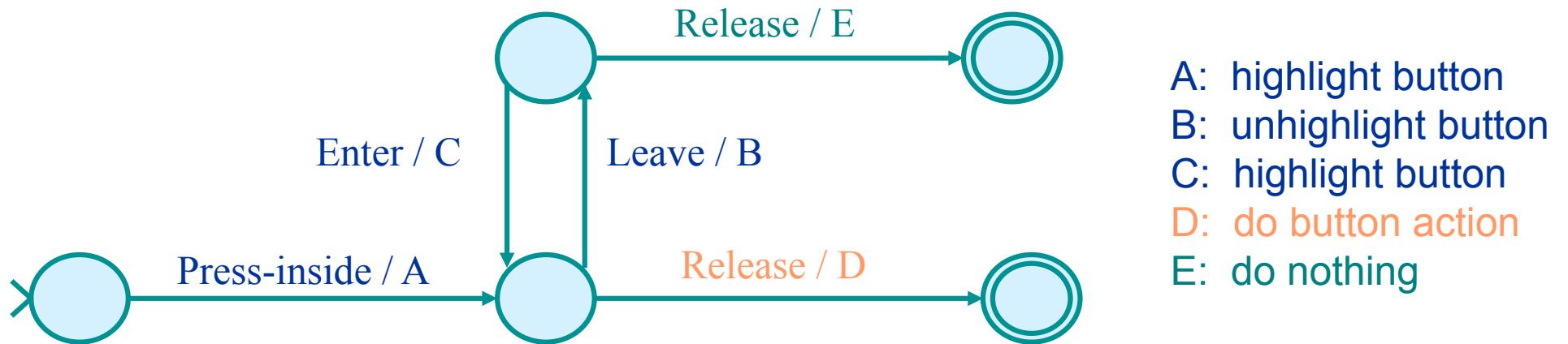


Transitions

- Arcs with an **event / action** label
- When the **event** occurs:
 - the **action** is executed
 - the **state** changes

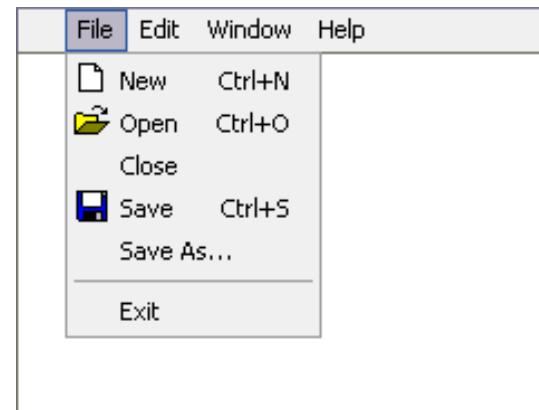


Clicking on a button

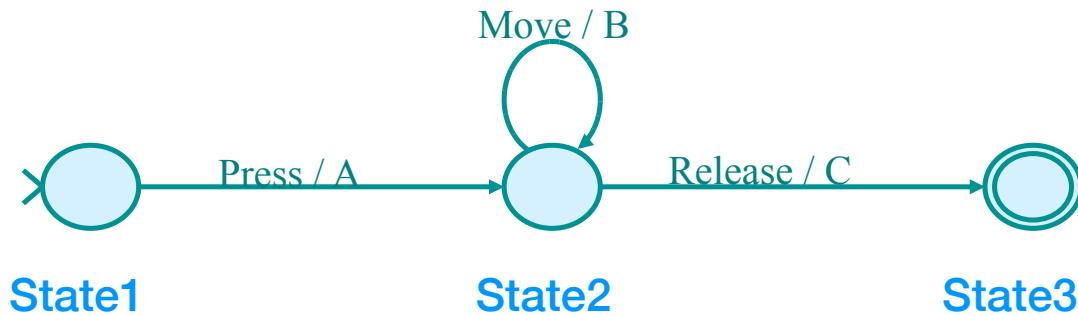


Limitations of this example

- No keyboard interaction
- Menu systems are more complex



FSM Implementation



```
state = start_state;  
while (true) {  
    event = wait_for_event();  
    state = fsm(state, event);  
}
```

Input \ Current state	State A	State B	State C
Input X
Input Y	...	State C	...
Input Z

```
state fsm (state, event) {  
    switch (state) {  
        case State2:  
            switch (event.kind) {  
                case MouseMove:  
                    actionB ();  
                    state = State2;  
  
                case MouseRelease:  
                    actionC ()  
                    state = State3;  
            }  
  
        case State1:  
            switch (event.kind) {  
                case ...  
            }  
    }  
    return state;  
}
```

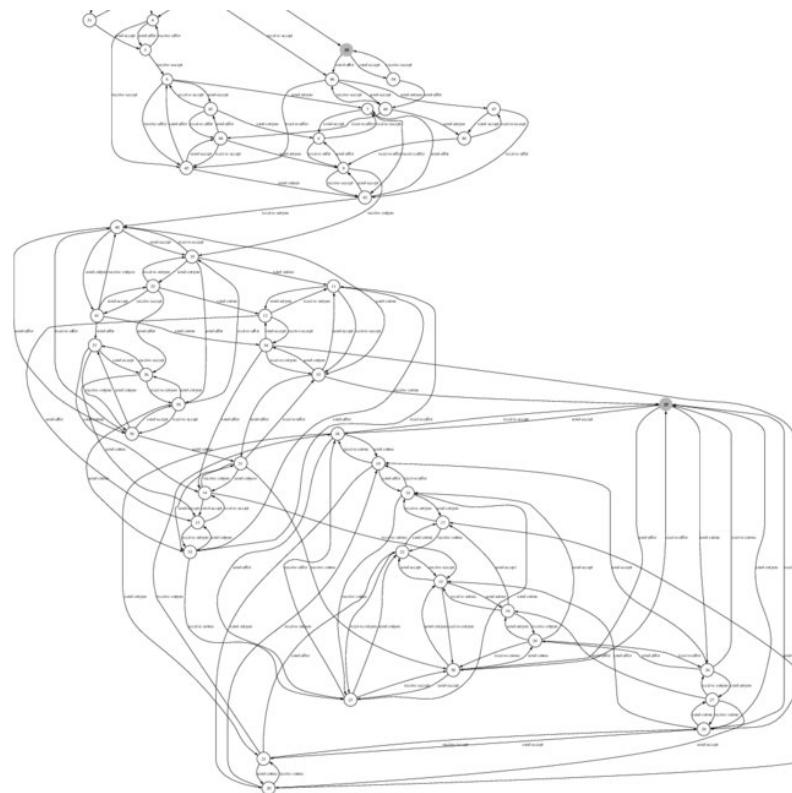
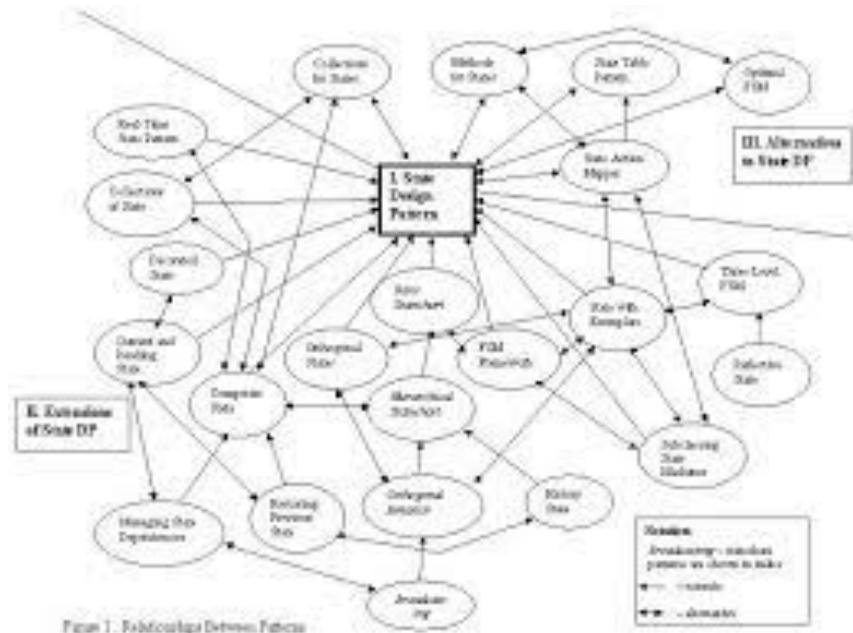
In practice

generally based on
state/transitions tables

Limitations

FSM limitations

- Nice for simple examples
 - But GUIs involve **many interactive objects** => **combinatorial explosion!**



Statecharts

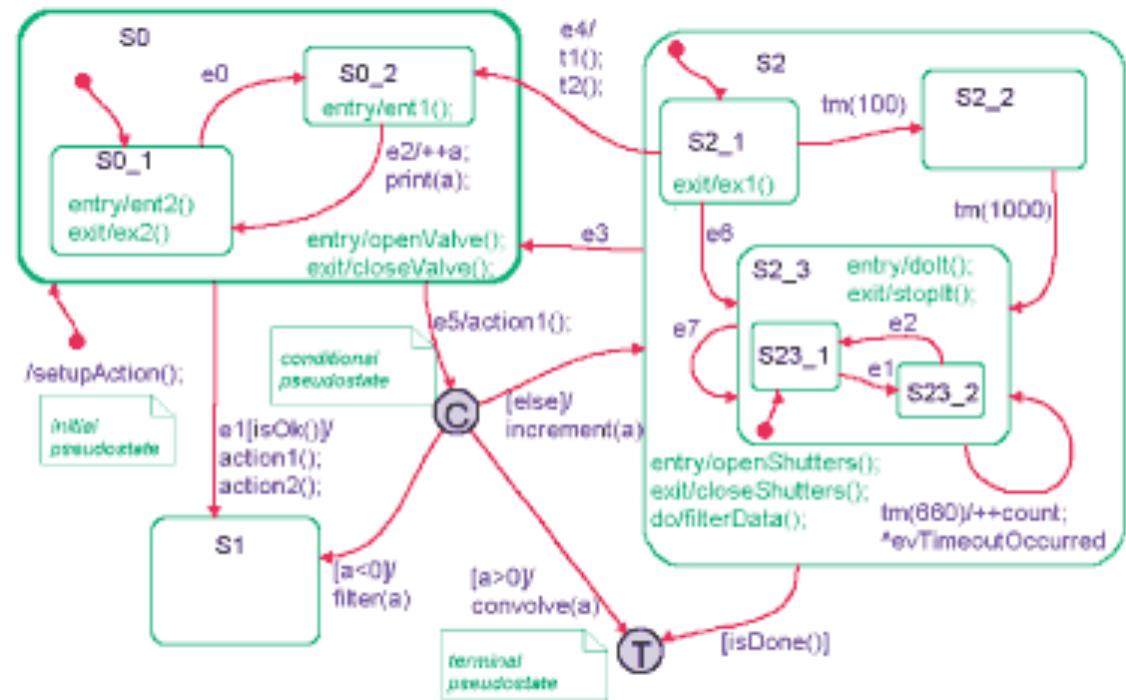
Statecharts

Hierarchical state machines

- => subsystem **decomposition**
- => **considerably reduce**
the number of transitions

Many tools

- including **UML**

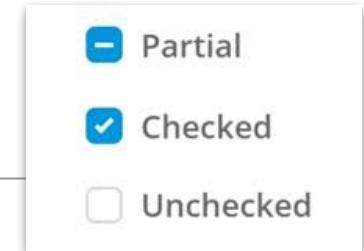


Qt State Machines

Qt State Machine Framework

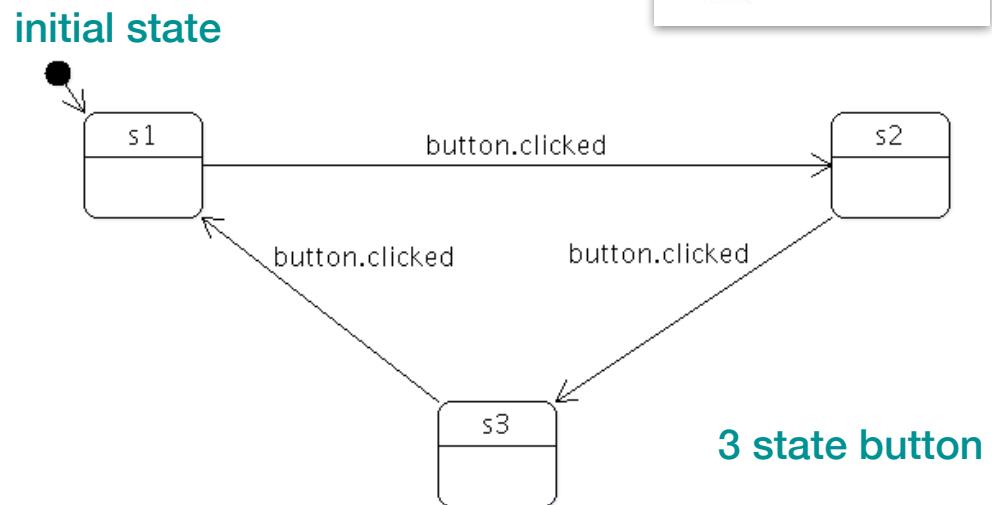
- **Statecharts** based on **SCXML**
- Allow:
 - **grouped states** : for **hierarchies**
 - **parallel states** : to avoid **combinatorial explosion**
 - **history states** : to **save** and **restore** the state
 - **user-defined** events
- Also useful for **animations**

Qt State Machines



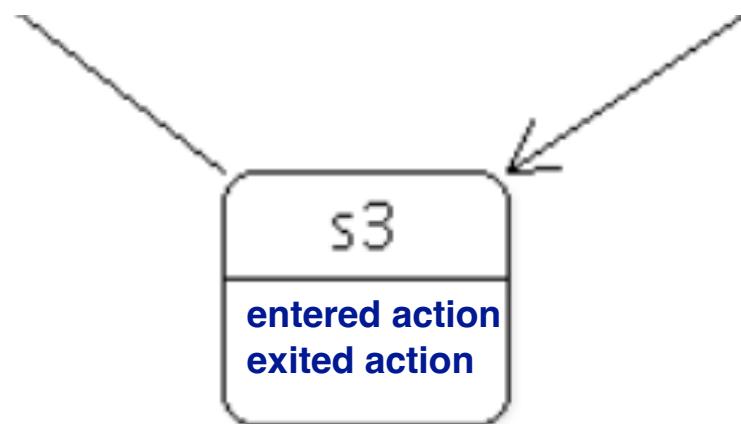
States

- The **initial state** must be specified
- Often GUIs have no **final state**



Actions

- **Qt actions** generally on **states**
 - when **entered**
 - when **exited**
- Can also be on **transitions**
 - (slightly more complicated)



Three state button

- Partial
- Checked
- Unchecked

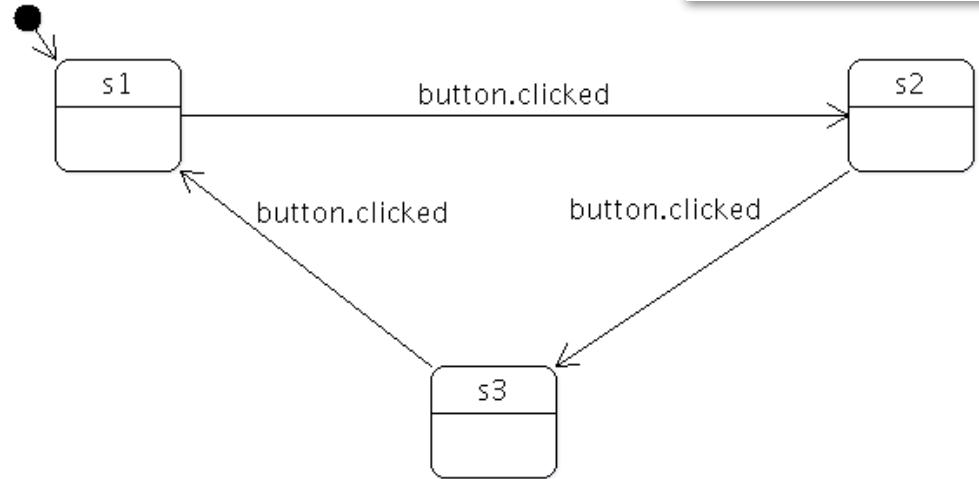
```
// creates the state machine
auto * mac = new QStateMachine();
QState *s1 = new QState();
QState *s2 = new QState();
QState *s3 = new QState();

mac->addState(s1);
mac->addState(s2);
mac->addState(s3);
```

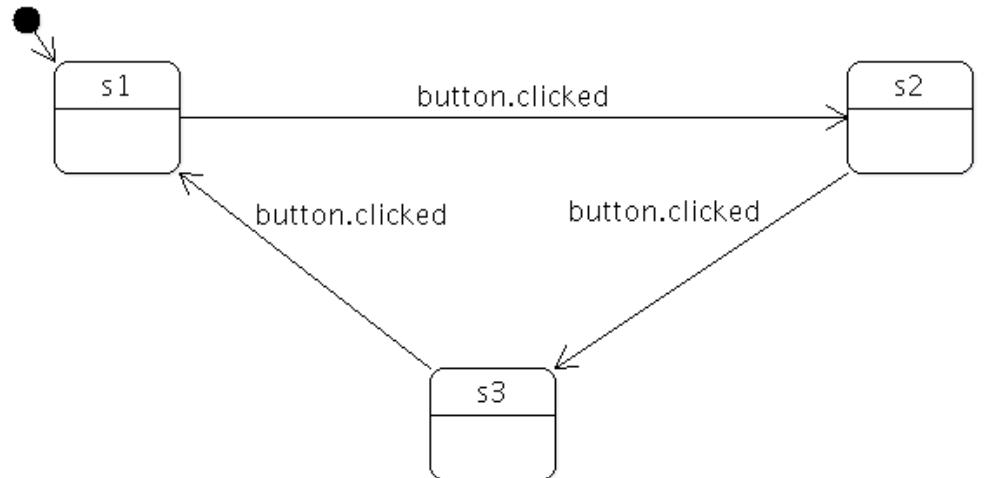
```
s1->addTransition(button, SIGNAL(clicked()), s2); // transition from s1 to s2 when bouton is clicked
s2->addTransition(button, SIGNAL(clicked()), s3);
s3->addTransition(button, SIGNAL(clicked()), s1);

QObject::connect(s3, SIGNAL(entered()), window, SLOT(showMaximized())); // actions are on states
QObject::connect(s3, SIGNAL(exited()), window, SLOT(showMinimized()));

mac->setInitialState(s1); // sets the initial state (don't forget!)
mac->start(); // starts the state machine
```



Three state button + quit



Improvement

- **quit button** that exits the application

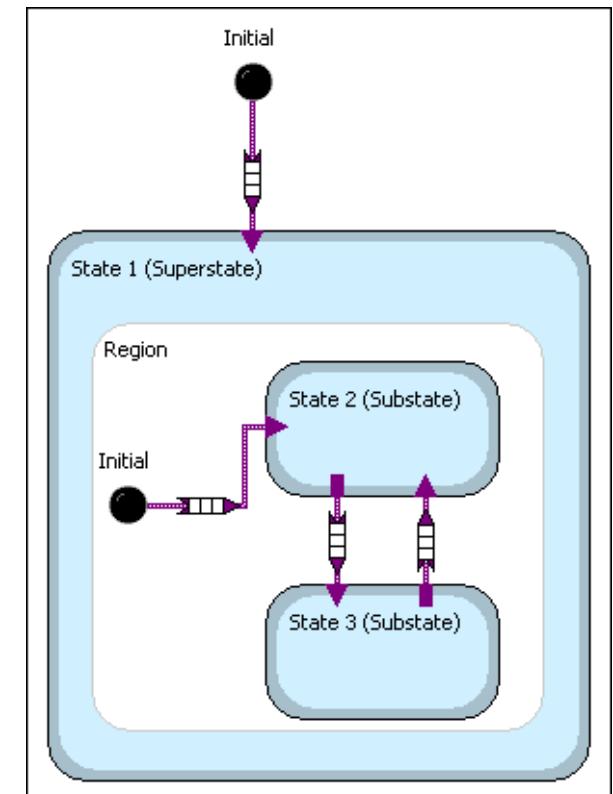
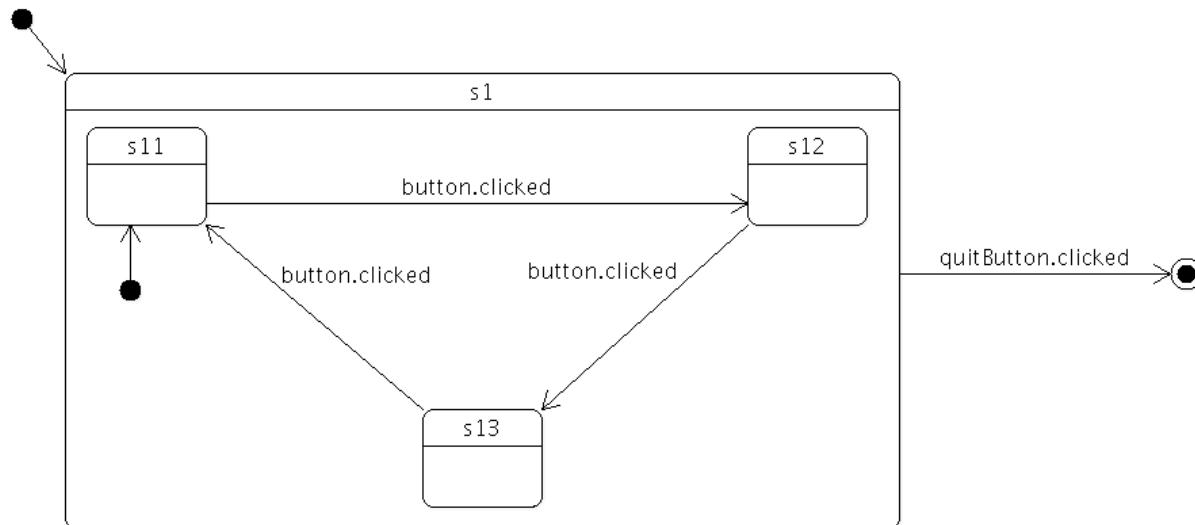
=> we need a transition **from each state to the final state !**

- **Hierarchies** make that simpler!

Hierarchies

Substates and superstates

- Substates inherit the transitions of their superstates
- Substates and superstates can be both entered
- Less transitions => simpler diagrams!



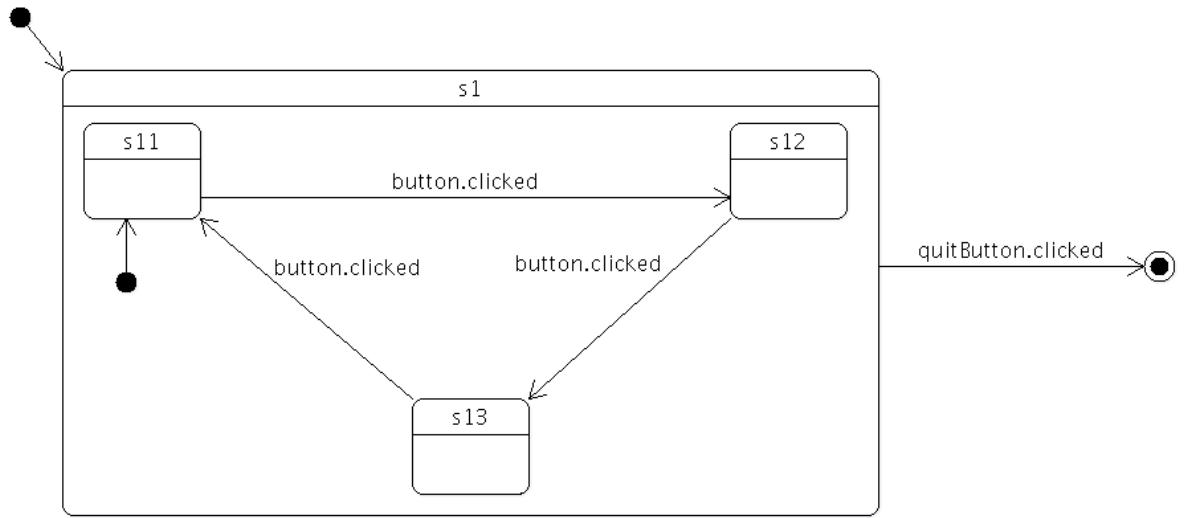
Hierarchies

```
QState *s1 = new QState();
mac->addState(s1);

// s11, s12, s13 are grouped in s1
// i.e. s1 is their superstate
QState *s11 = new QState(s1);
QState *s12 = new QState(s1);
QState *s13 = new QState(s1);

s11->addTransition(button, SIGNAL(clicked()), s12);
s12->addTransition(button, SIGNAL(clicked()), s13);
s13->addTransition(button, SIGNAL(clicked()), s11);
s1->setInitialState(s11);
```

```
QFinalState *s2 = new QFinalState();           // s2 is a final state
mac->addState(s2);                          // s1 children inherit the s1->s2 transition
s1->addTransition(quitButton, SIGNAL(clicked()), s2); // finished() is emitted when the final state is reached
connect(mac, SIGNAL(finished()), QApplication::instance(), SLOT(quit()));
mac->setInitialState(s1);
mac->start();
```



Hierarchies

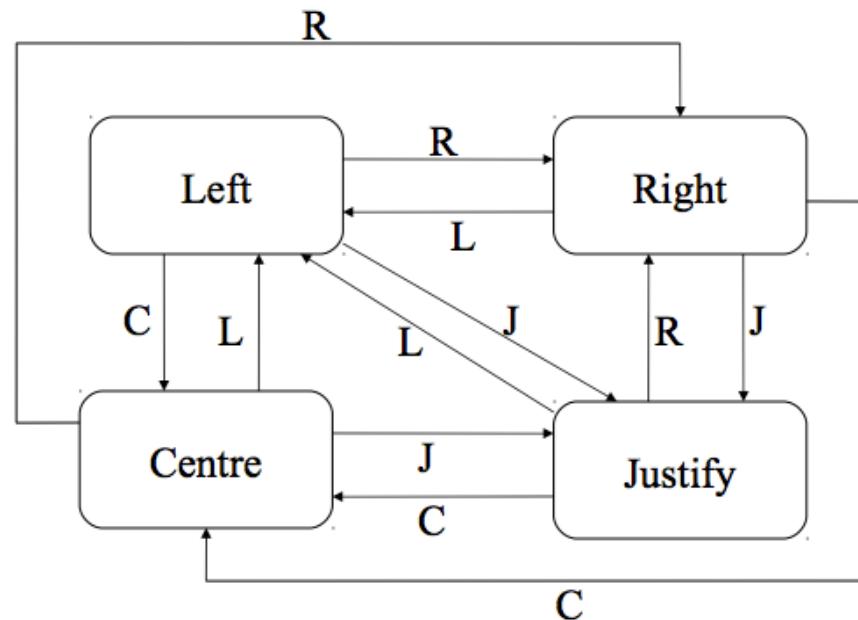
Second exemple

4 radio buttons => 4 **exclusive** states

=> Transition needed between each **couple** of states



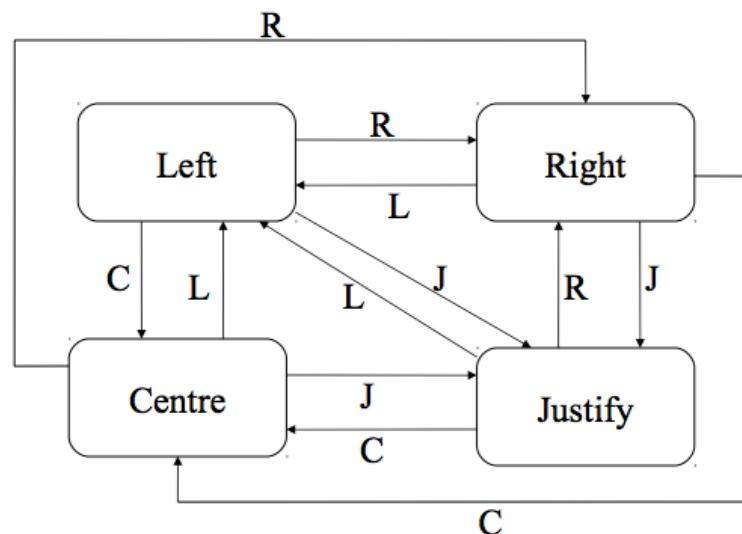
text alignment



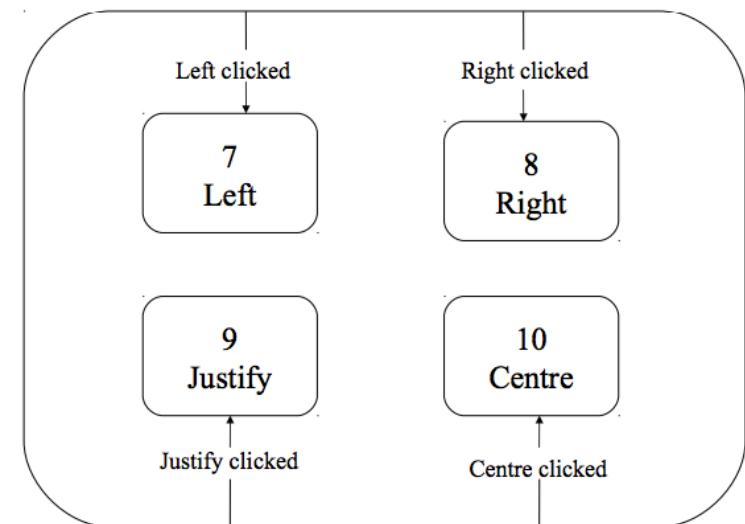
Quite **complex**
for such a **simple** case!

Hierarchies

- A **hierarchy** makes it simple!
- Only 4 **transitions**, all **inherited**
- The superstate is **always entered**



flat diagram

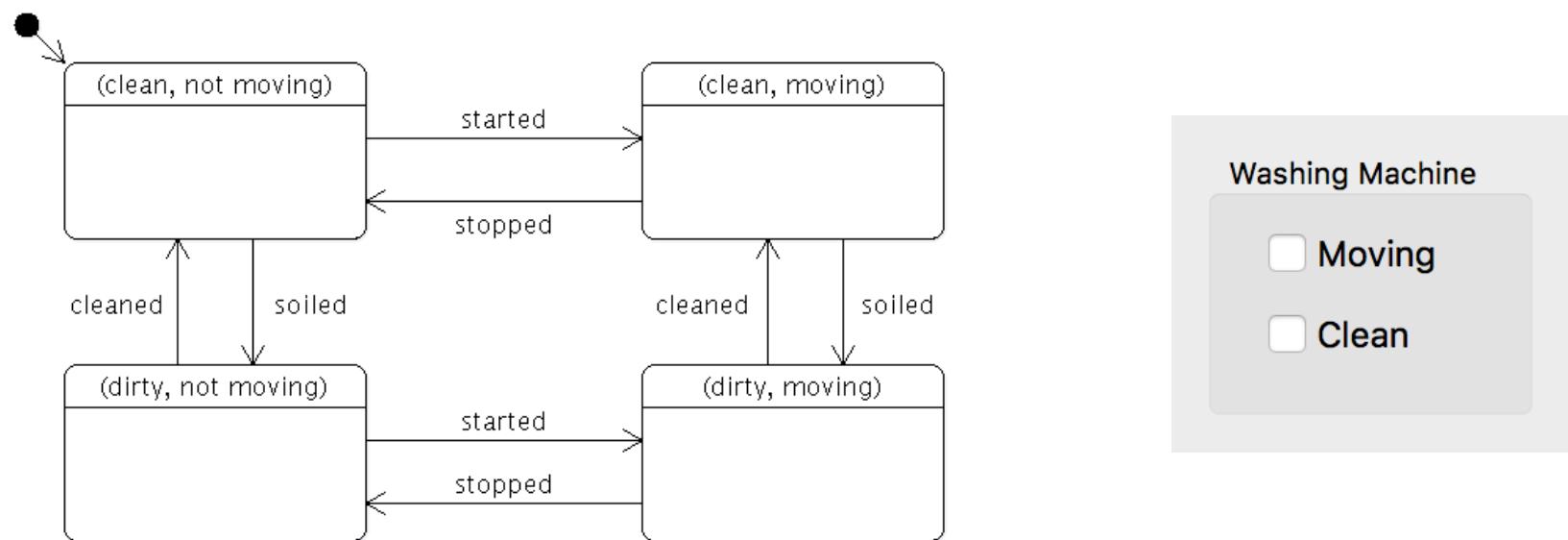


hierarchical diagram

Parallel states (concurrency)

Exemple

- Two **independant** check boxes
- Each check box has **two states** (on/off) => all **combinations** must be considered

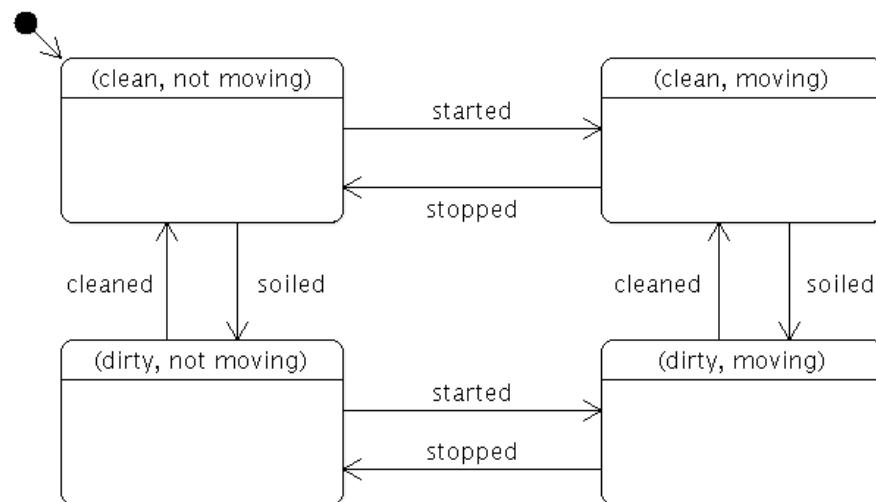


Parallel states

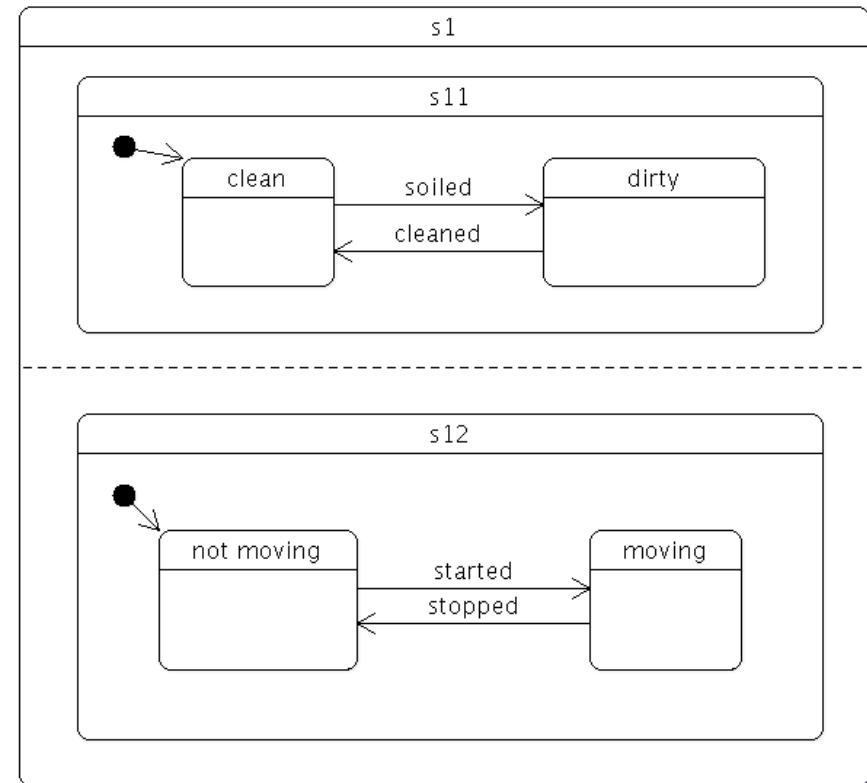
```
QState *s1 = new QState(QState::ParallelStates);  
QState *s11 = new QState(s1);  
QState *s12 = new QState(s1)
```

Solution: parallel states

- several states are **simultaneously entered**



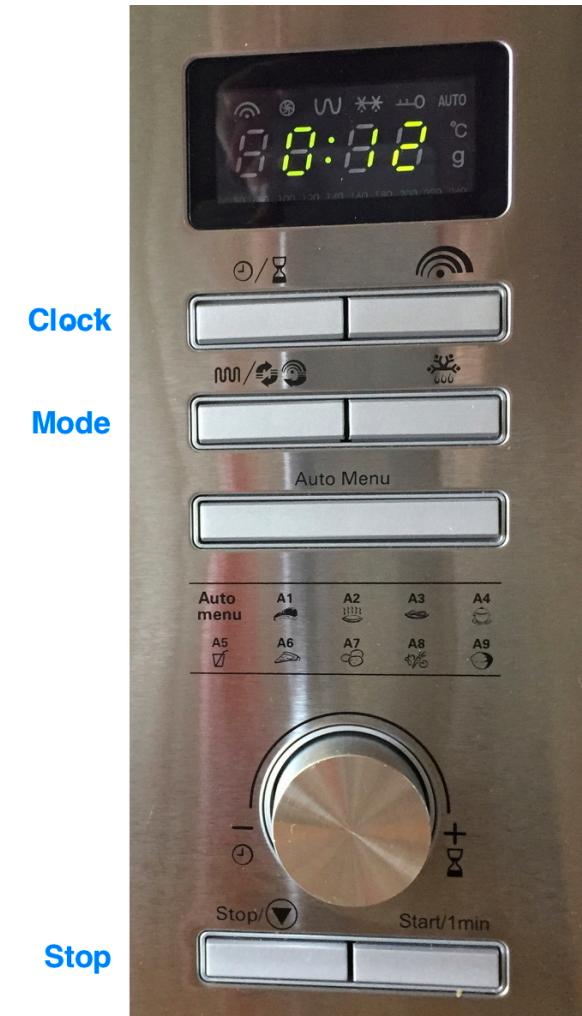
flat diagram



hierarchical diagram

Lab

Goal: implement the GUI and behavior of a microwave oven
TP3 on <https://perso.telecom-paris.fr/elc/qt/>



Clock

Mode

Stop

Power

Defrost

Auto Menu

Dial

Start

Lab

If possible, use qmake (not CMake !)

With Qt6 the State Machine module must be added explicitely

- MaintenanceTool

- Add or remove components
 - Additional Libraries
 - Qt State Machine

In the .pro file add statemachine

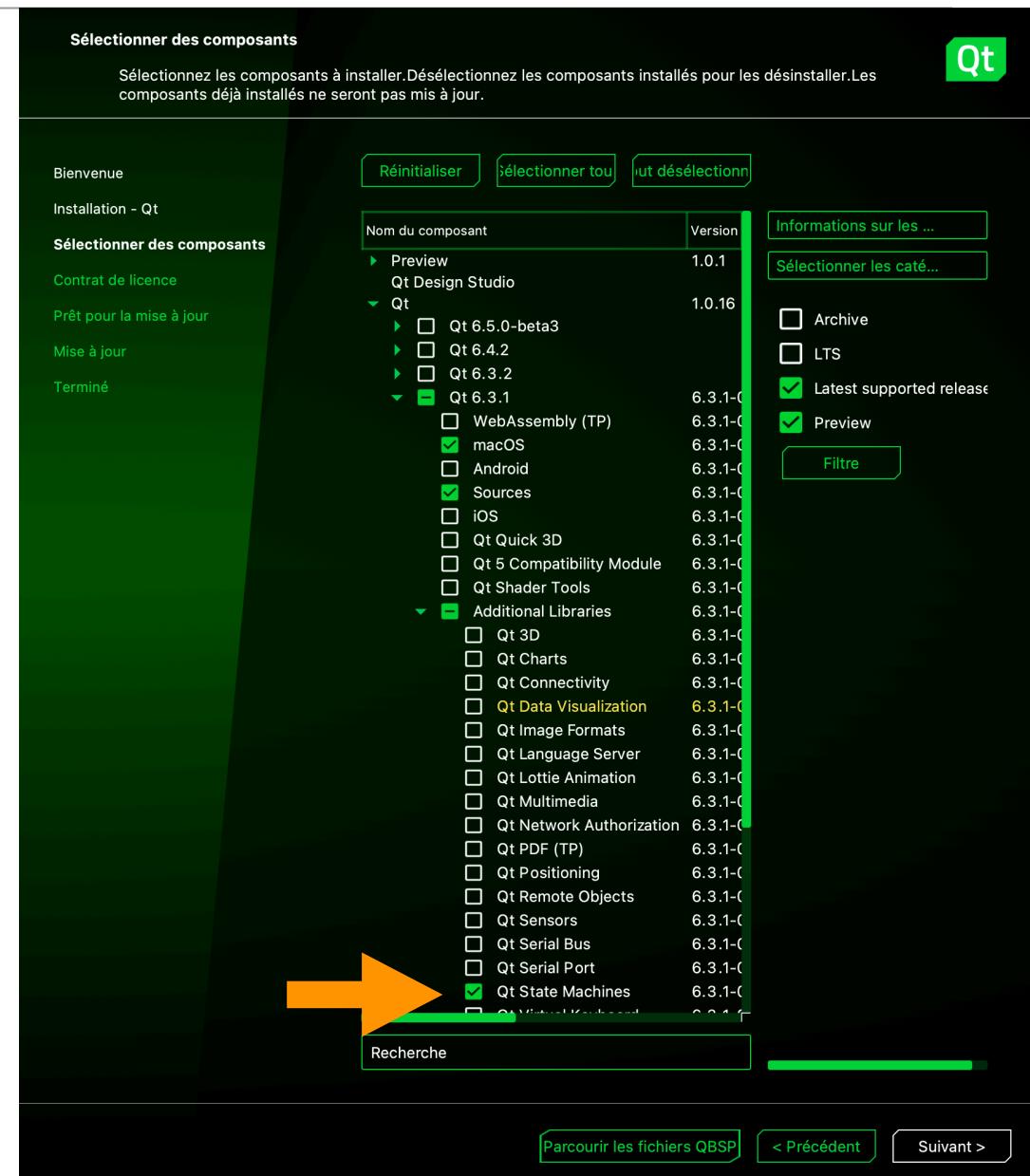
```
QT += core gui widgets statemachine
TARGET = proj
TEMPLATE = app

SOURCES += main.cpp MainWindow.cpp

HEADERS += MainWindow.h

FORMS += MainWindow.ui
```

MaintenanceTool



Timeouts

State transitions can also be fired by timers

- To execute an action after a **delay**:

```
QTimer * timer = new QTimer();
state1->addTransition(timer, SIGNAL(timeout()), state2);
....
timer->start(delay);
....
timer->stop();
```

Action on timeout

```
connect(timer, SIGNAL(timeout()), widget, SLOT(doSomething()));
```

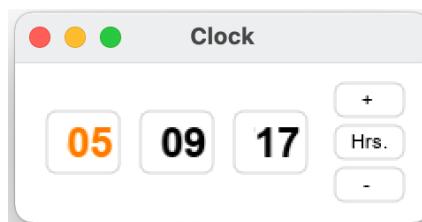
Is the state entered?

Example

- Clock displayed **differently** depending on the **active state**



idle state



set-hour state



set-min state

```
void MainWindow::setClock(int value) {  
    if (hour_state->active()) {  
        setHour(value);  
    }  
    else if (min_state->active()) {  
        setMin(value);  
    }  
}
```

(only available in recent versions of Qt)

Note: no active state during the transition!

```
QSet<QAbstractState*>  
config = machine->configuration();  
  
if (config.contains(hour_state)) {  
    ...  
}  
else if (config.contains(min_state)) {  
    ...  
}
```

Transitions.h

Transitions.h header

- set of **helper functions** : <http://www.telecom-paris.fr/~elc/qt/Transitions.h>
- **actions** on **transitions** (Qt default: actions on states)
- **actions** on **events** and **button events** (Qt default: actions on signals)
- more compact syntax

Qt :

```
state1->addTransition (button, SIGNAL(clicked( )), state2);  
QObject::connect (state2, SIGNAL(entered( )), this, SLOT(dolt( )));
```

action on state

Transitions.h :

```
addTrans (state1, state2, button, SIGNAL(clicked( )), this, SLOT(dolt( )));
```

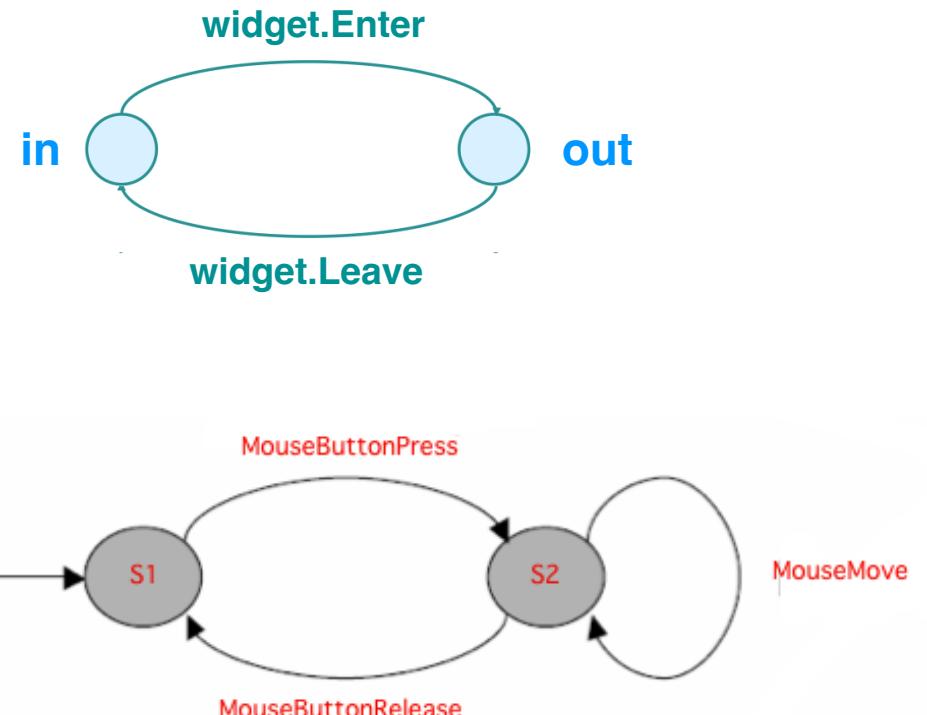
action on transition

Transitions on events

Events

```
addEventTrans (out, in, widget, QEvent::Enter);
```

```
addEventTrans (in, out, widget, QEvent::Leave);
```



Mouse Events

when left button is pressed

```
addMouseTrans (s1, s2, canvas, QEvent::MouseButtonPress, Qt::LeftButton, this, SLOT(newLine( )));
```

```
addMouseTrans (s2, s2, canvas, QEvent::MouseMove, Qt::NoButton, this, SLOT(adjustLine( )));
```

```
addMouseTrans (s2, s1, canvas, QEvent::MouseButtonRelease, Qt::LeftButton);
```

Transitions on events

Modifiers example: shift + (left or right button)

```
addMouseTrans (s1, s2, canvas, QEvent::MouseButtonPress, Qt::LeftButton | Qt::RightButton)
->setModifierMask(Qt::ShiftModifier);
```

Get Mouse position

```
QPoint mousepos;
addMouseTrans (s2, s2, canvas, QEvent::MouseMove, Qt::NoButton, mousepos,
this, SLOT(adjustLine( )) );
```

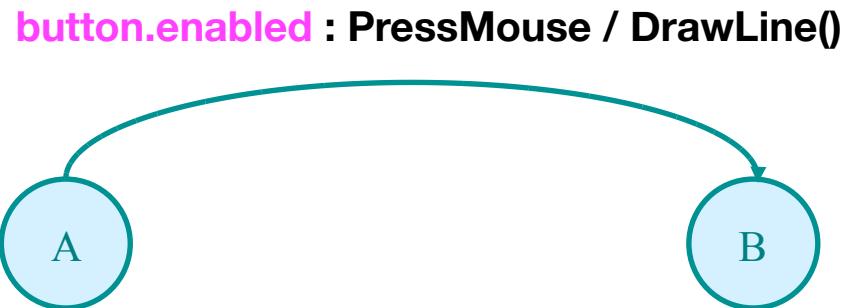
Another way to get mouse position (slightly inaccurate on XWindows)

```
QPoint cursorPos(QWidget* w) { return w->mapFromGlobal(QCursor::pos()); }
```

Guards

Guards

- The transition is performed **only if a predicate is true**
- Notation: **predicate : event / action**



With Transitions.h

```
addTrans (guard, state1, state2, SIGNAL(clicked( )));  
addTrans (guard, state1, state2, SIGNAL(clicked( )), this, SLOT(dolt( )) );
```

guard = bool variable (or object that can be compared to a bool)

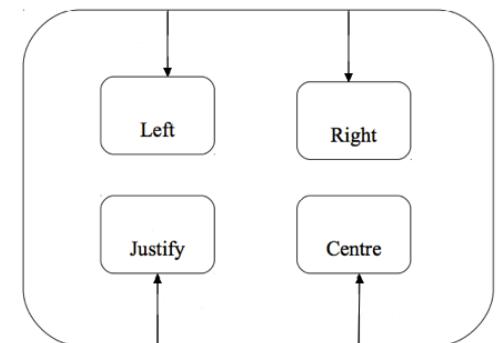
Custom events

Custom Events

- The user can create her own events: `QEvent(int type)`
 - `type` = the type of the event (between `QEvent::User` et `QEvent::MaxUser`)
- `postEvent(QEvent*)` posts an event to a state machine
- `postDelayedEvent(QEvent*, delay)` posts with a delay

Transitions and events

```
addUserEventTrans(s, s_left, LEFT);
addUserEventTrans(s, s_right, RIGHT);
...
mac.postEvent( new QEvent(LEFT) ); // triggers the transition
```



Transition filters

- A transition can be made only if its `eventTest()` method is called and returns **true**
- Technique for implementing **guards** or **user-defined behaviors**

```
class MouseTransition : public QMouseEventTransition {  
    QPoint& pos;  
  
    bool eventTest(QEvent * e) {  
        if (! QMouseEventTransition::eventTest(e)) return false;  
        QEvent* realEvent = static_cast<QStateMachine::WrappedEvent*>(e)->event(); // get event  
        switch (realEvent->type( )) {  
            case QEvent::MouseMove:  
            case QEvent::MouseButtonPress:  
            case QEvent::MouseButtonRelease:  
                _pos = static_cast<QMouseEvent*>(realEvent)->pos(); // update mouse position  
        }  
        return true; // true => the event triggers the transition  
    }  
};
```

exemple: updates the mouse position at each transition

Animations and properties

Properties

```
s1->assignProperty( label, "text", "In state s1" );
```

```
s1->assignProperty( button, "geometry", QRectF(0, 0, 50, 50) );
```

Especially useful for animations

```
s1->assignProperty( button, "geometry", QRectF(0, 0, 50, 50) );
```

```
s2->assignProperty( button, "geometry", QRectF(0, 0, 100, 100) );
```

the size of the button will change smoothly:

```
s1->addTransition( button, SIGNAL(clicked()), s2 )
    ->addAnimation( new QPropertyAnimation( button, "geometry" ) );
```


Software architecture models

Arch Model

- Analogy with **natural language**
- three layers

Presentation (lexical view)

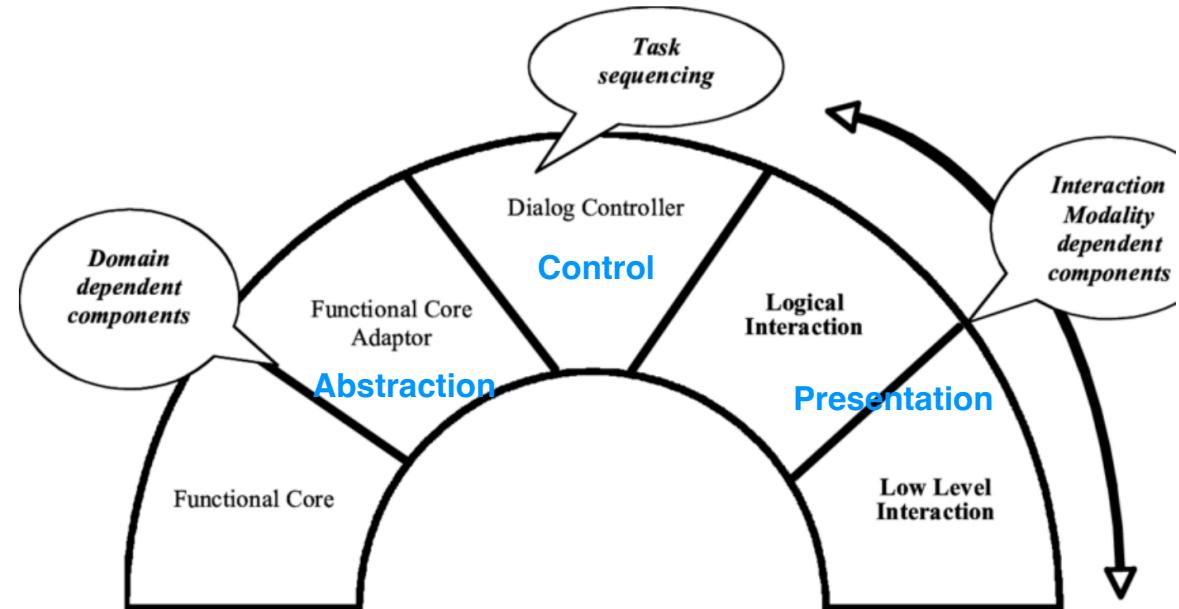
- Graphical objects and primitives

Control (syntax)

- UI logics

Abstraction (semantics)

- API between UI and **functional core** (the non UI part)



from Bastide et al.

Dialogue control

Syntax examples

- object verb

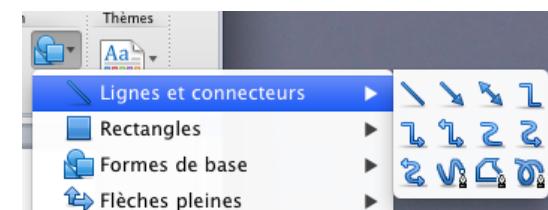
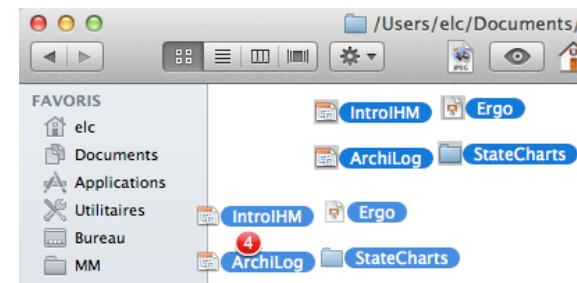
- (object)* verb

- (object)* verb object

- verb object

- verb (objects)*

- verb object or (object)* verb



Surligner un ou plusieurs mots dans Word

Dialogue control

Syntax examples

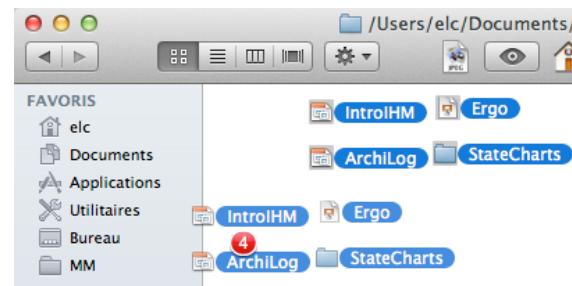
- object verb

- put text in orange



- (object)* verb

- destroy several files

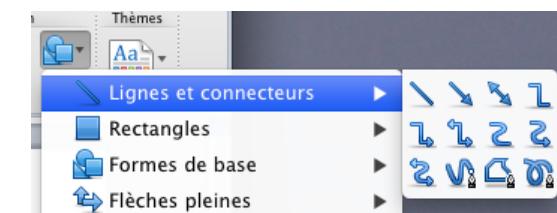


- (object)* verb object

- drag multiple files

- verb object

- draw a single line



- verb (objects)*

- draw several lines (implies a mode)

- verb object or (object)* verb

- MSWord highlighter



Surligner un ou plusieurs mots dans Word

Spatial modes

the effect of an action depends on **where** it is performed

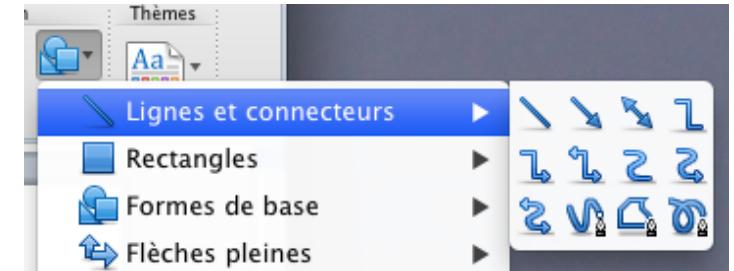
examples:

- buttons, menus...



Advantages

- visibility
- discovery
- memorization (always at the same place)



Temporal modes

the effect of an action depends on **when** it is performed
often used for creating **new objects**

examples:

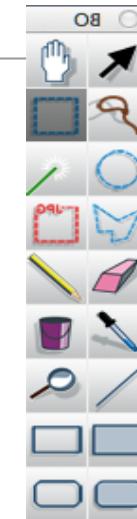
- drawing tool palettes
- modal dialog box

Drawbacks

- lack of **visibility** => risk of **errors** (what is the current state?)



Temporal modes

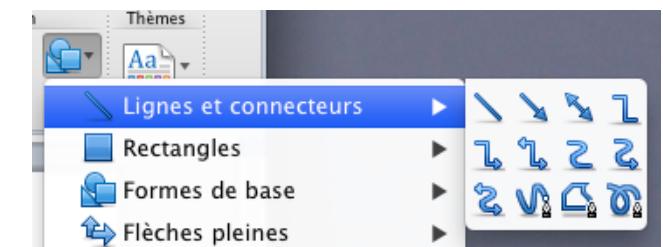


Persistent modes

- remain activated
 - e.g.: draw several lines in a row
 - syntax: **verb (objects)***

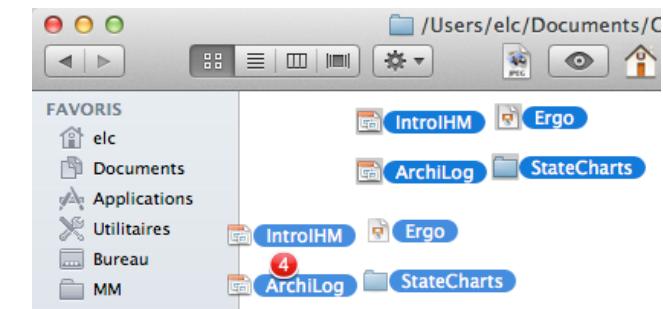
One shot

- takes effect only on the **next action**
 - e.g.: draw a single line
 - syntax: **object verb**



Quasi modes

- **automatically disabled** at the end of the action
 - Ctrl, Alt modifiers
 - Drag and drop



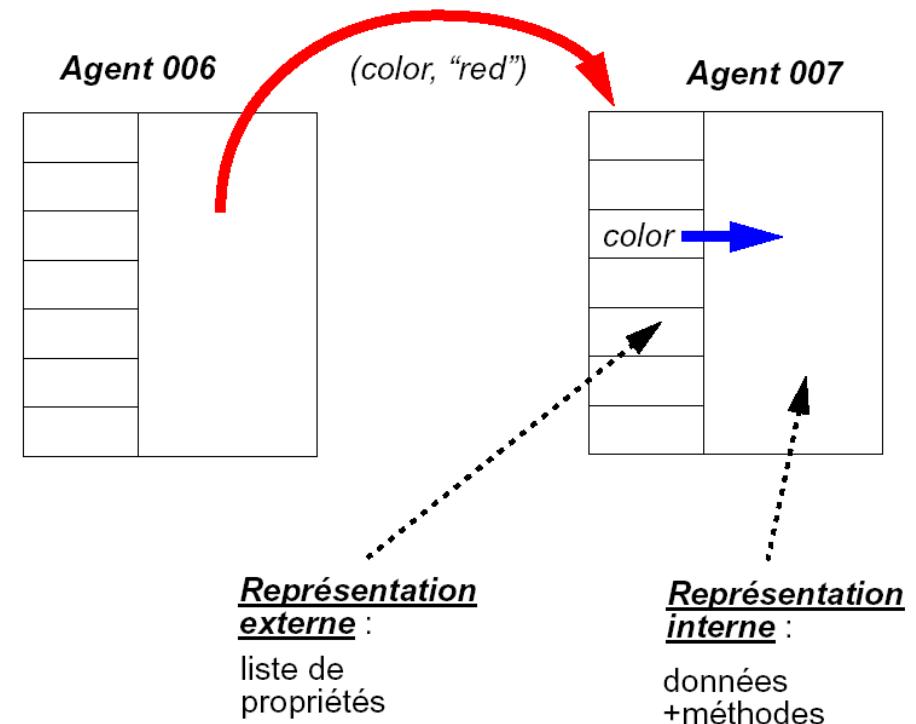
Multi-agent models

Vision: interactive system =

- set of **agents**
- react to / produce **events**

Agent = treatment system with:

- **memory**
- **states**
- a "**processor**"
- a set of event **senders / receivers**



Inspiration of various models

- XWindow, **MVC**, etc.

MVC

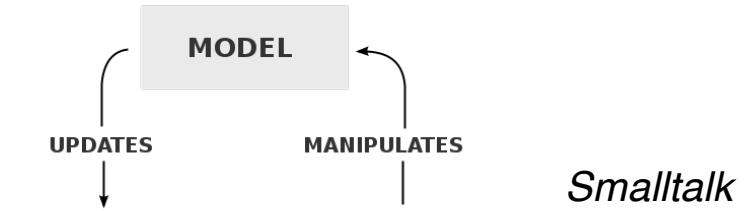
Goals

- Better **structure** applications:
 - Clear separation between **data** and **GUI**
 - Ex: *spreadsheet* : data vs. graphical representation
- Allow managing **multiple views**
- Help **reusing** source code
- Help **team** development

MVC

Original model

- Smalltalk GUI toolkit (1978)

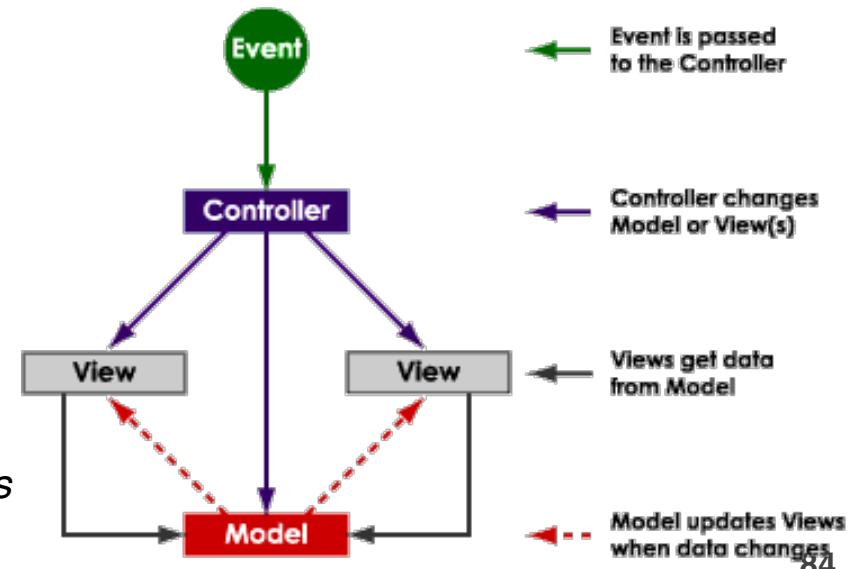


Smalltalk

Three components

- **Model** : data (and data logics)
- **View** : visual representation
- **Controller** : input management and behavior of interactive system

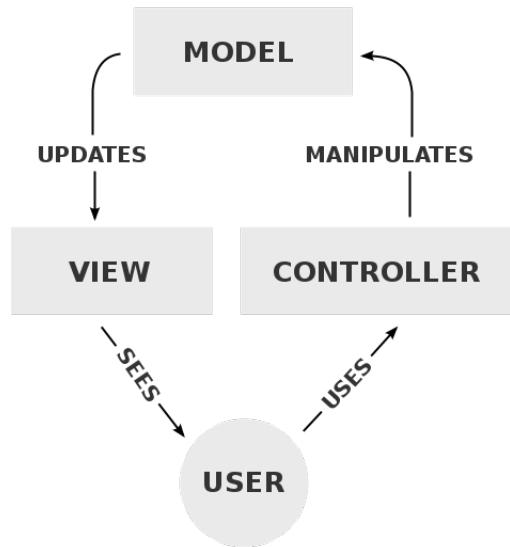
variant, multiple views



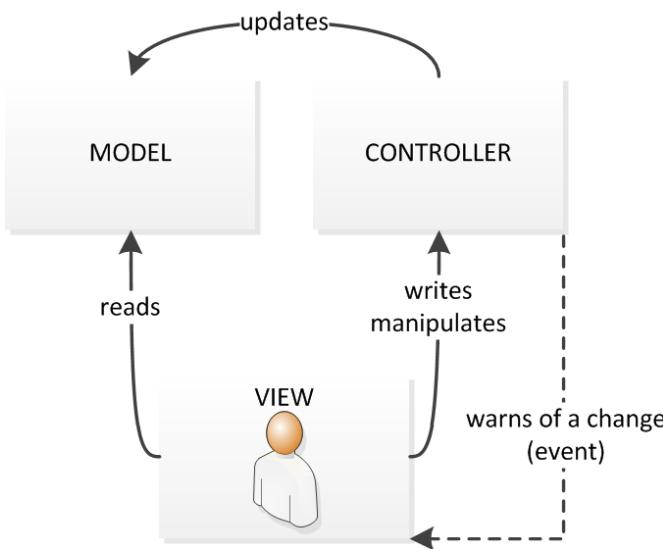
MVC variants

Various flavors

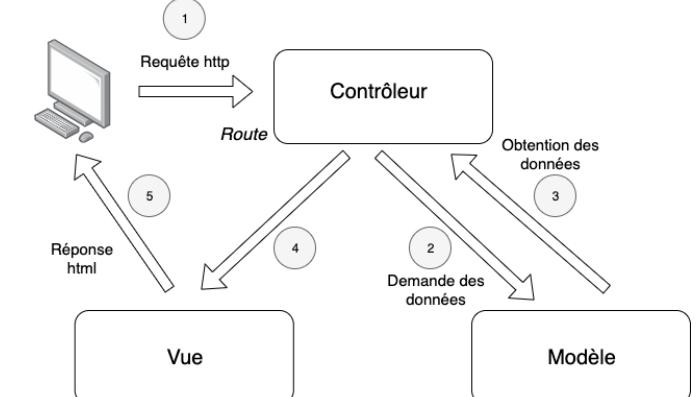
- for graphical interfaces, Web development, etc.



Smalltalk



Variant



Web development

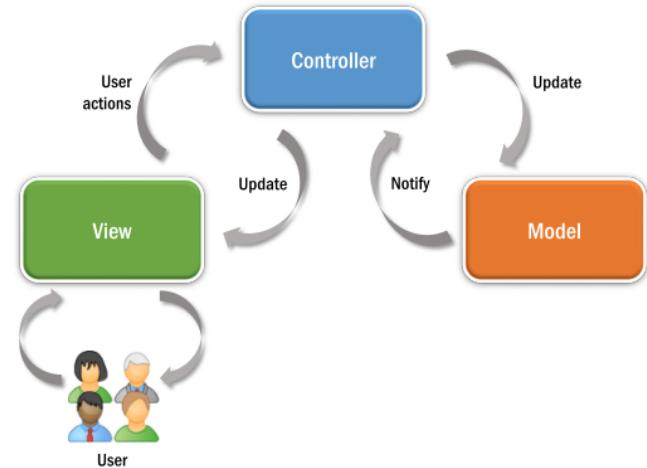
MVC variants

ARCH related

- **View = Presentation**
- **Controller = Control**
- **Model = Abstraction**

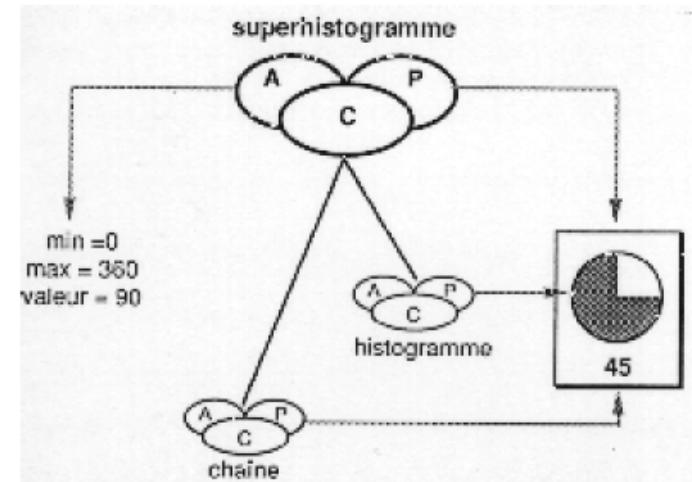
Exemple: **PAC** model (J. Coutaz, LIG)

PAC is **hierarchical**



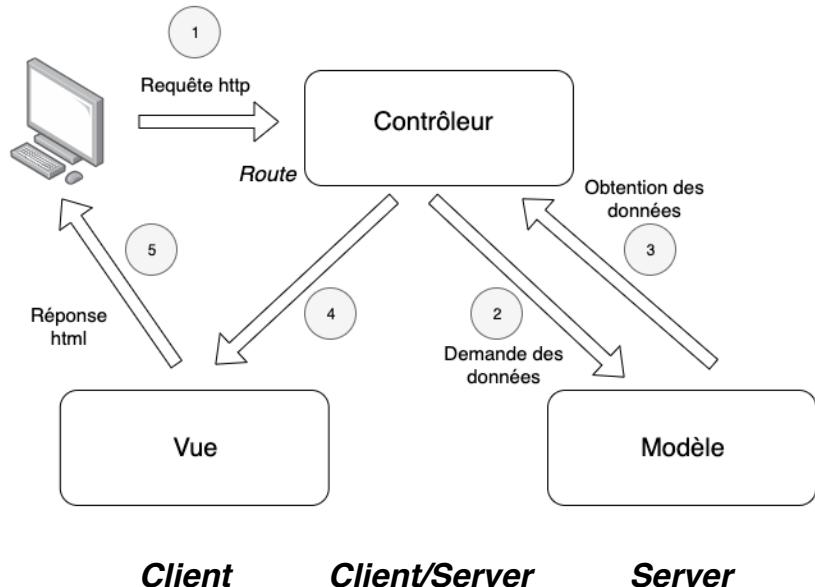
Many Variants

- model–view–adapter (MVA)
- model–view–presenter (MVP)
- model–view–view model (MVVM)
- hierarchical model–view–controller (HMVC)
- etc.

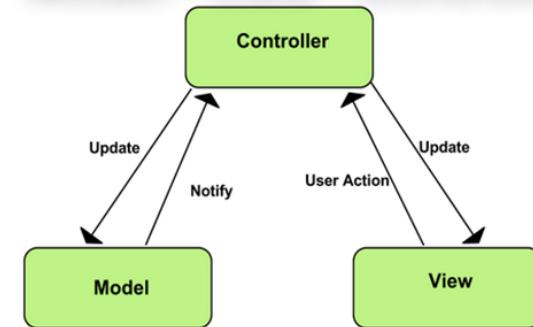


PAC

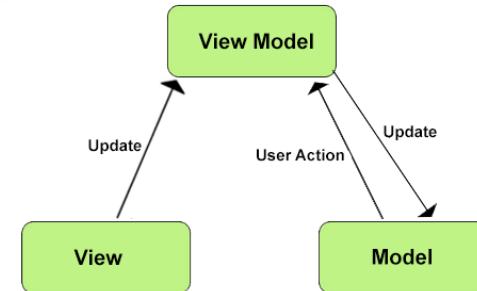
MVC & Web



Model - View - Controller



Model View View Model



Angular framework

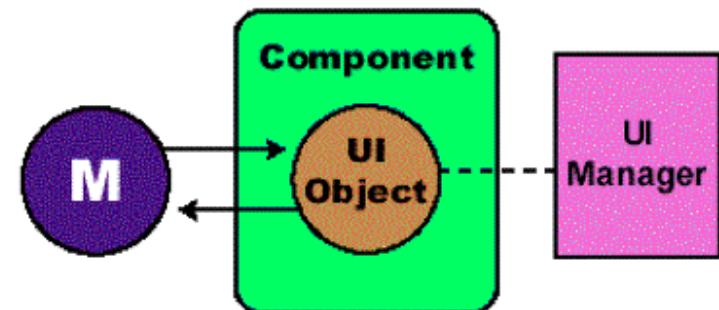
- **View**: client on Web browser
 - **Controller**: application logics
(on server or partly on client)
 - **Model**: database and data logics
- **ViewModel**: model of the View
 - **coordinates** view's interactions with models

<https://deepakjosecodes.com/mvvm-architecture-in-angular-guide-with-code-examples/>

MVC & Java Swing

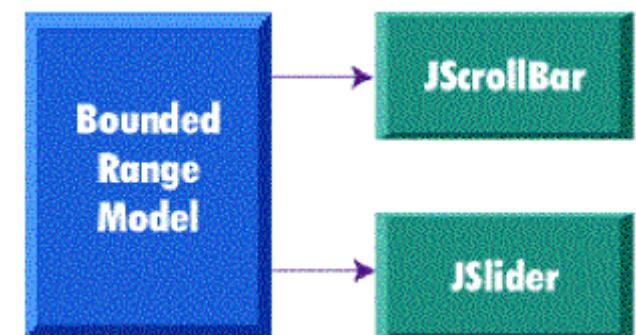
Internal M+VC architecture

- **JComponent** =
 - **Model** (for components managing data)
 - + **UIComponent**
- **UIComponent** = **Controller** + **View** (mixed)
- **Models** can be shared



Example

- **JScrollBar**, **JSlider** own a **BoundedRangeModel**



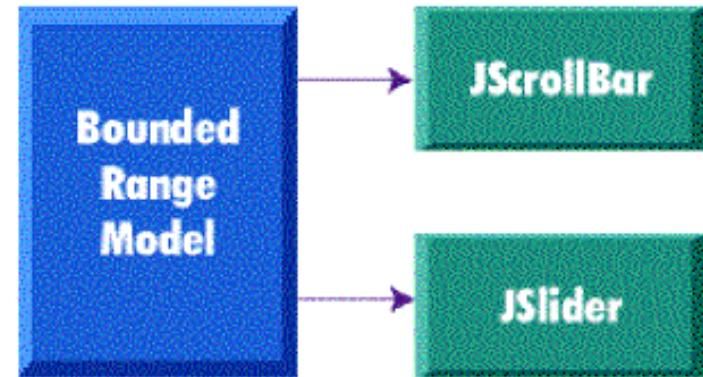
MVC & Java Swing

In the JSlider API

```
public BoundedRangeModel getModel();
public void setModel(BoundedRangeModel);
```

Changing the JSlider model

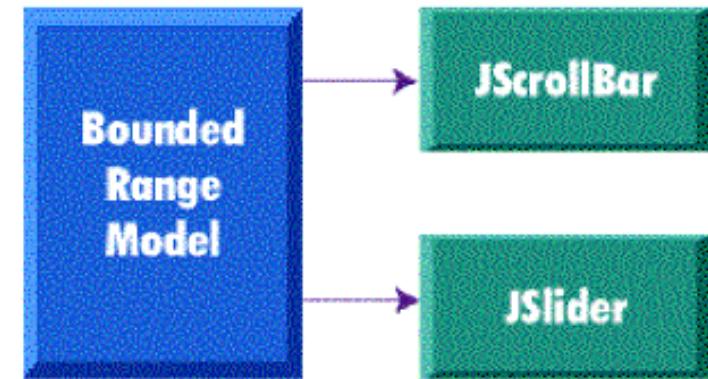
```
JSlider slider = new JSlider();
BoundedRangeModel model =
    new DefaultBoundedRangeModel() {
        public void setValue(int n) {
            System.out.println("SetValue: "+ n);
            super.setValue(n);
        }
    });
slider.setModel(model);
scrollbar.setModel(model);
```



MVC & Java Swing

In the JSlider API

```
public BoundedRangeModel getModel();
public void setModel(BoundedRangeModel);
```



One can also ignore models

```
JSlider slider = new JSlider();
int value = slider.getValue();

// coherent because:
public int getValue() {return getModel().getValue(); }
```

MVC & Java Swing

Pluggable Look and Feel

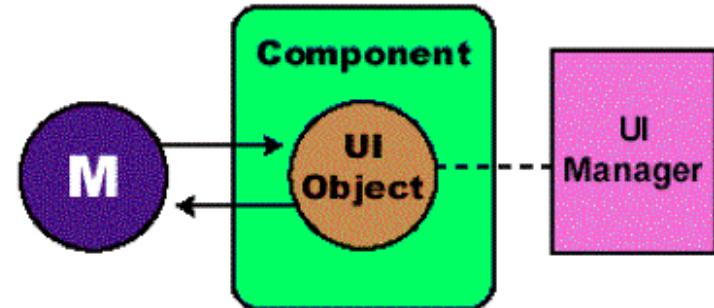
The **UIManager** changes the **ComponentUIs** dynamically

Java Metal:

```
public static void main(String[ ] args) {
    try {
        UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
    } catch (Exception e) { }
}
```

Windows:

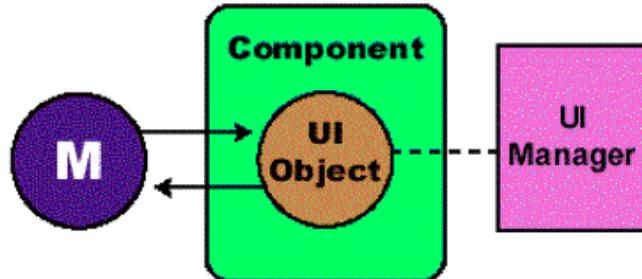
```
UIManager.setLookAndFeel( "com.sun.java.swing.plaf.windows.WindowsLookAndFeel" );
```



MVC & Java Swing

Internal M+VC architecture

- **JComponent** =
 - **Model**
 - **View+Controller**



Higher-level point of view

- **Controllers** = Java **Listeners**
- **Views** = **JComponents**
(include low level **Controllers**)
- **Models** = application **data**

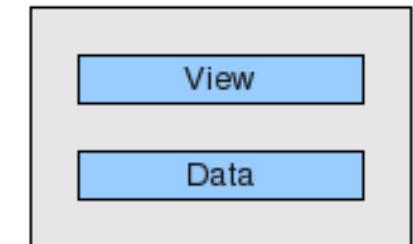
MVC

- can be used in different ways
- at different levels of abstraction

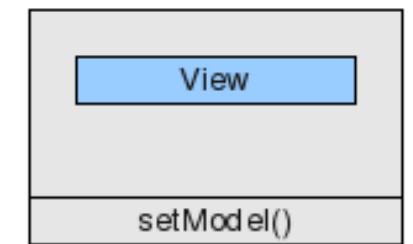
MVC & Qt

Some widgets have 2 implementations

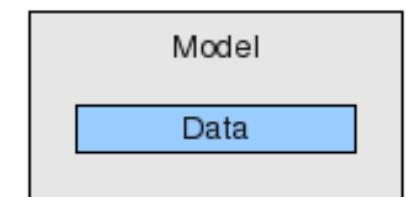
- **Standard** widget implementation
- **Model/View** implementation
 - Manage **external data**
 - Make it easy to deal with **multiple views**



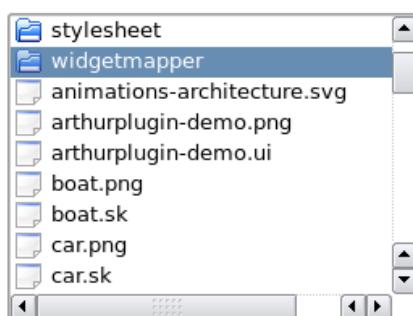
standard



Model



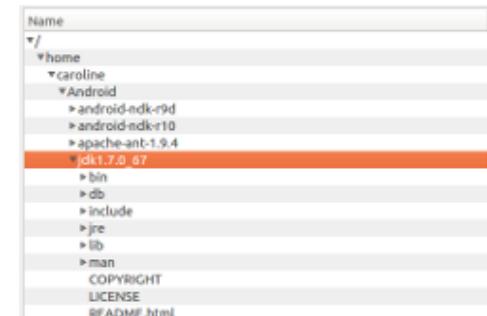
model / view



QListWidget
QListView



QTableWidget
QTableView



QTreeWidget
QTreeView

MVC & Qt: example

Example

- List of **Persons**
- Each **Person** has :
 - a **first name**
 - a **last name**
 - a **phone number**
- The list is stored in a **std::vector**

	1	2	3
1	Jean	Dupond	01 01
2	Alfred	Einstein	03 14
3	Désiré	Landru	66 66

```
struct Person {  
    std::string firstname, lastname, phonenumber;  
};  
  
std::vector<Person> persons;
```

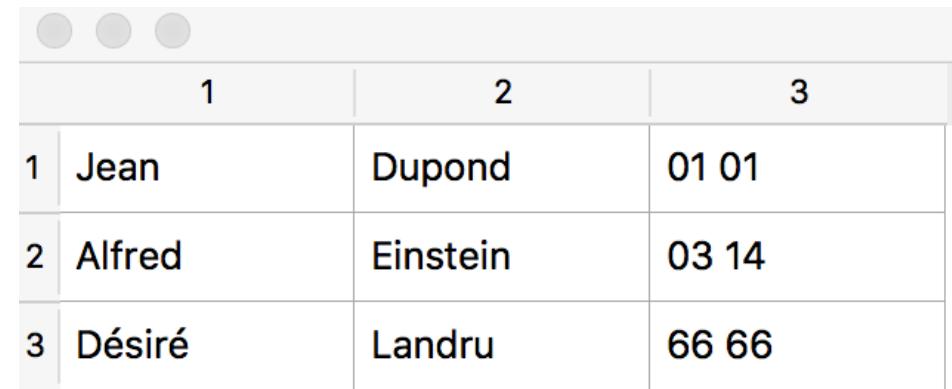
MVC & Qt: example

main.cpp

```
#include <QApplication>
#include <QtWidgets/QTableView>
#include "mymodel.h"

int main (int argc, char *argv[])
{
    QApplication app (argc, argv);

    MyModel model;
    QTableView view;
    view.setModel(&model);
    view.show();
    return app.exec();
}
```



	1	2	3
1	Jean	Dupond	01 01
2	Alfred	Einstein	03 14
3	Désiré	Landru	66 66

Note :

- objects can be created in the stack only in the main() function!
- why?

mymodel.hpp

```
#include <QAbstractTableModel>
#include <string>
#include <vector>
```

```
class MyModel : public QAbstractTableModel {
    Q_OBJECT
```

```
public:
```

```
    MyModel (QObject *parent);
```

number of rows / columns

```
    int rowCount(const QModelIndex& = QModelIndex()) const override { return persons.size(); }
```

```
    int columnCount(const QModelIndex& = QModelIndex()) const override {return 3;}
```

read / change a cell in the model

```
    QVariant data(const QModelIndex& index, int role = Qt::DisplayRole) const override;
```

```
    bool setData(const QModelIndex & index, const QVariant & value, int role);
```

```
    Qt::ItemFlags flags(const QModelIndex & index) const override;    options
```

```
private:
```

```
    struct Person { std::string firstname, lastname, phonenumber; };
```

```
    std::vector<Person> persons;
```

```
};
```

	1	2	3
1	Jean	Dupond	01 01
2	Alfred	Einstein	03 14
3	Désiré	Landru	66 66

mymodel.cpp

```
#include "mymodel.h"

MyModel::MyModel(QObject *parent) : QAbstractTableModel(parent) {
    persons = { {"Jean", "Dupond", "01 01"}, {"Alfred", "Einstein", "03 14"}, {"Désiré", "Landru", "66 66"} };
}

QVariant MyModel::data(const QModelIndex &index, int role) const {      read a cell in the model
    if (role == Qt::DisplayRole) { display mode
        switch (index.column()) {
            case 0:
                return QString( persons[index.row()].firstname.c_str() );
                break;
            case 1:
                return QString( persons[index.row()].lastname.c_str() );
                break;
            case 2:
                return QString( persons[index.row()].phonenumber.c_str() );
                break;
        }
    }
    return QVariant();
}
```

	1	2	3
1	Jean	Dupond	01 01
2	Alfred	Einstein	03 14
3	Désiré	Landru	66 66

mymodel.cpp (continued)

```
bool MyModel::setData(const QModelIndex & index, const QVariant & value, int role) {  
    if (role == Qt::EditRole) {   edit mode  
        switch (index.column()) {  
            case 0:  
                persons[index.row()].firstname = value.toString().toStdString();  
                break;  
            case 1:  
                persons[index.row()].lastname = value.toString().toStdString();  
                break;  
            case 2:  
                persons[index.row()].phonenumber = value.toString().toStdString();  
                break;  
        }  
        emit dataChanged (index,index);      signals that this cell was modified  
    }  
}
```

change
a cell in the
model

```
Qt::ItemFlags MyModel::flags (const QModelIndex &index) const {  
    return Qt::ItemIsEditable | QAbstractTableModel::flags(index);  
}
```

options: mean that
the table can be edited

MVC & Qt

Synchronizing 2 views

- The **selection** can be synced
- Works also with **other data types** than text
- **Other widgets** can use models
(see **QDataWidgetMapper**)

not only text

```
MyModel * model = new MyModel;  
QTableView * view1 = new QTableView;  
QTableView * view2 = new QTableView;  
  
view1->setModel (model);  
view2->setModel (model);
```

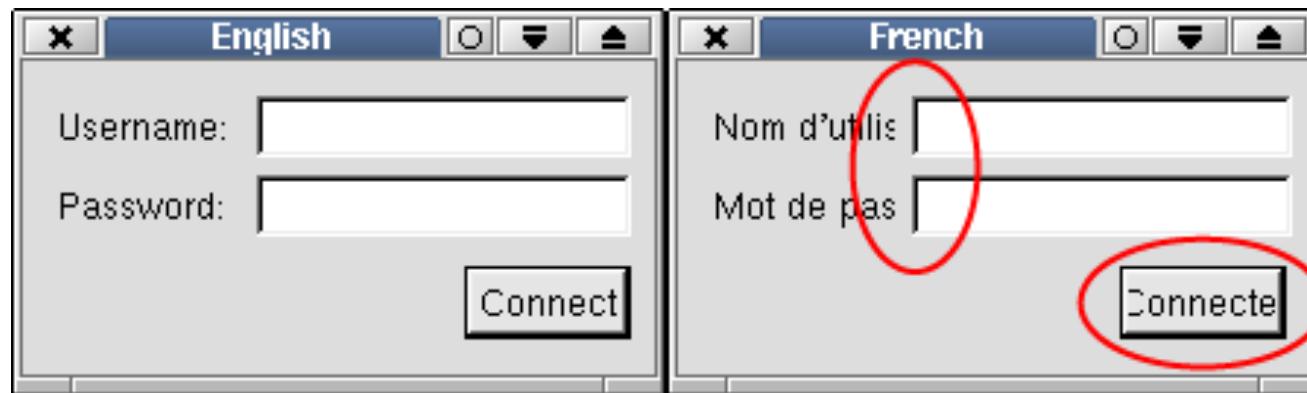
two views

stardelegate				
	Title	Genre	Artist	Rating
1	Mass in B-Minor	Baroque	J.S. Bach	★★★★★
2	Three More Foxes	Jazz	Maynard Ferguson	★★★★★
3	Sex Bomb	Pop	Tom Jones	★★★
4	Barbie Girl	Pop	Aqua	★★★★★

Layout management

Manual layout

- Easy when using an **IDE** (by dragging and dropping widgets)
- **Drawback:** the UI is not **localizable** (word sizes depend on languages)



Layout management

Automatic layout

- Automatic calculation of positions and sizes

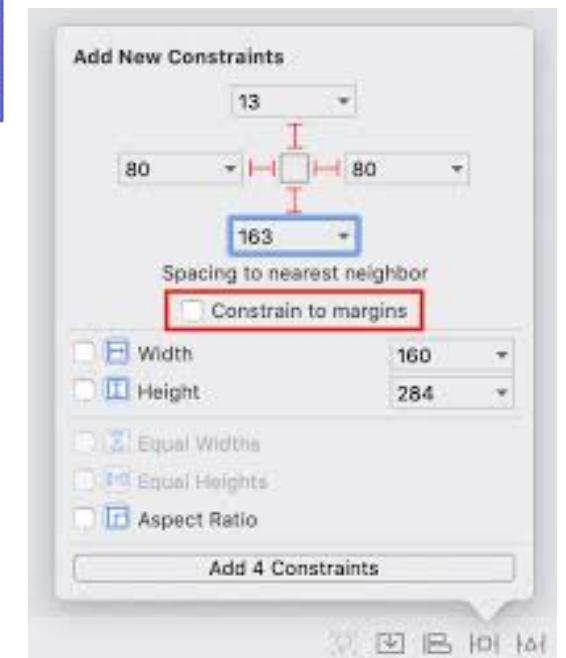
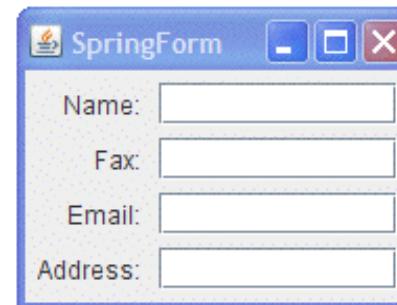
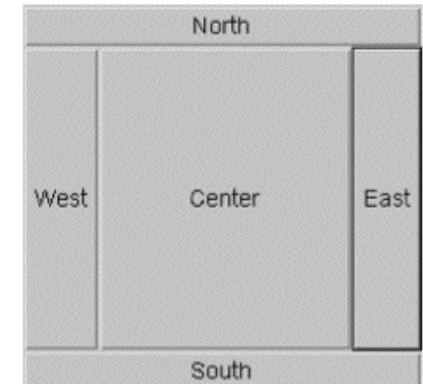
- Two approaches:

- **Layout managers**

- Java Swing, JavaFX, Qt, etc.
- Not as nice?

- **Constraints**

- Apple Xcode, etc.
- Can be somewhat complex



Layout management

Automatic layout + CSS

- Cascaded **Style Sheets**
- Typical in **Web interfaces**
- Flexibility, configurability
- Many modern GUI toolkits support CSS-like styles:

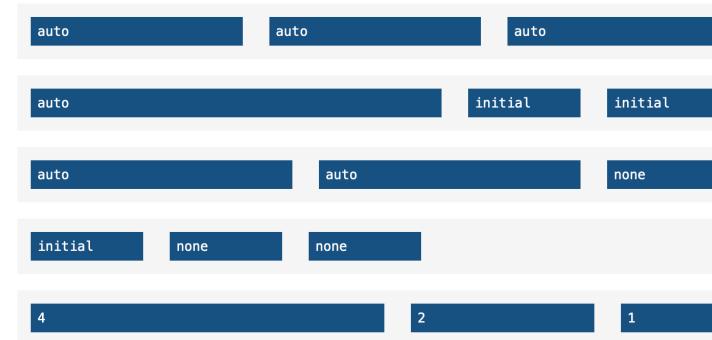
Qt, JavaFX, .NET, etc

CSS Demo: flex

RESET

The interface shows four boxes on the left with their flex properties: 'flex: 1;', 'flex: 2;', 'flex: 1 30px;', and 'flex: 1 1 100px;'. On the right, there is a 'Change me' button and three boxes labeled 'flex: 1' with a blue border.

<https://developer.mozilla.org/fr/docs/Web/CSS/flex>

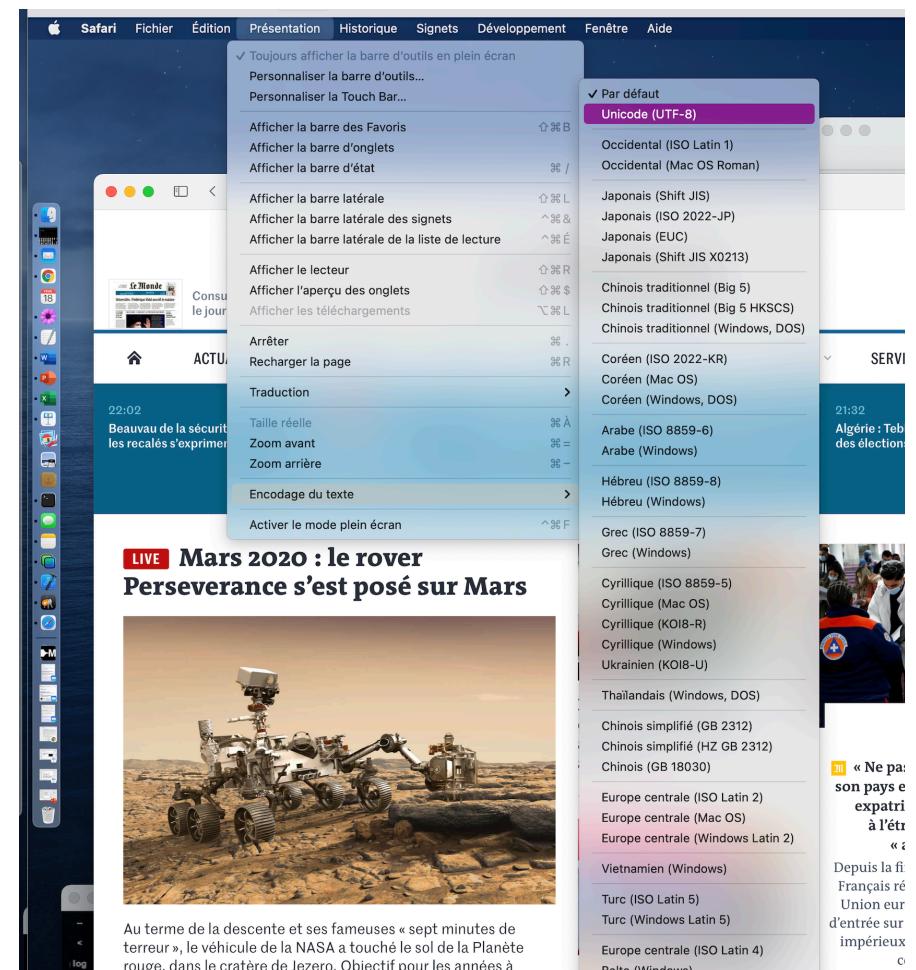


Localization

Process of adapting a product's translation to a specific country or region

- Language, numbers, dates, currency, etc.
 - Beware of dates **10/09 vs 09/10**
 - And numbers: **1,000.1 vs 1000,1**
- Application "locale"
- Encoding of text

```
dhcpwifi-22-132:~> locale
LANG="fr_FR.UTF-8"
LC_COLLATE="fr_FR.UTF-8"
LC_CTYPE="fr_FR.UTF-8"
LC_MESSAGES="fr_FR.UTF-8"
LC_MONETARY="fr_FR.UTF-8"
LC_NUMERIC="fr_FR.UTF-8"
LC_TIME="fr_FR.UTF-8"
LC_ALL=
```



Unicode

Standard for consistent encoding, representation and handling of text

UTF8

- **Web**, Linux, Windows (partly)
- **Variable length**: one symbol = **1 to 4 bytes**
- Compatible with **ASCII**



UTF16

- Java, Qt, Windows (now moving to UTF8)
- **Variable length**: one symbol = **2 or 4 bytes**
- **Almost fixed** in practice: 1 symbol = 2 bytes for most languages and ideographs

UTF32

- 1 symbol = 4 bytes

Unicode

C++

- **std::string** is compatible with **UTF8**
- but 1 symbol = 1 to 4 char(s)!

```
std::string str = "a"; // 1 byte  
std::string str = "à"; // 2 bytes  
std::string str = "😊"; // 4 bytes
```

Qt: Unicode strings

- A **QString** contains **QChars** (16 bits each)

```
QFile file("output.txt");  
  
if (file.open(QIODevice::WriteOnly)) {  
    QTextStream out (&file);  
    out << "Result: " << 10 << " " << 3.14;  
}
```

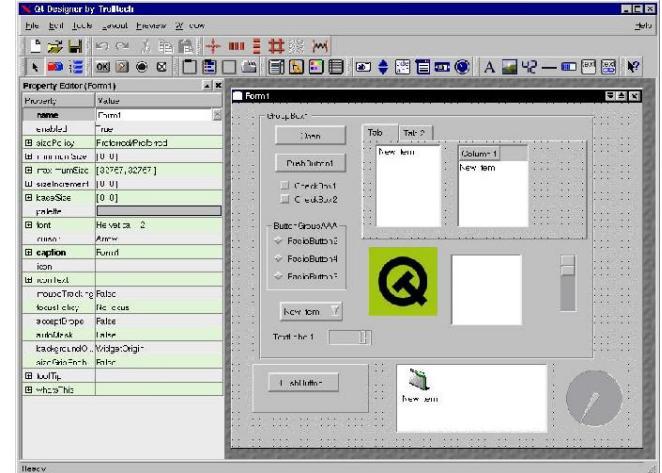
Conversions:

- **qPrintable(string)** : converts to **local encoding**
- **toAscii()** : **ASCII** 8 bits
- **toUtf8()** : **UTF-8** Unicode multibyte
- **toLocal8Bit()** : local encoding 8 bits
- etc.

Visual tools

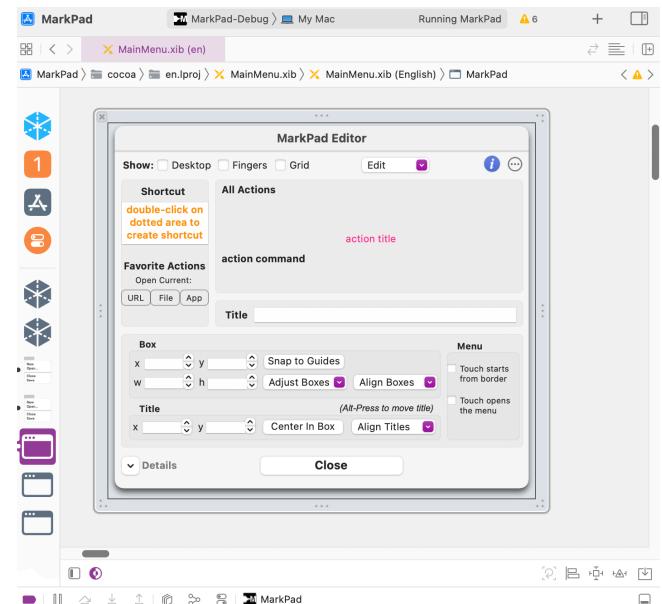
IDEs and Interface Builders

- Many tools: Eclipse, Visual Studio, Xcode, NetBeans, QtCreator, etc.



Benefits

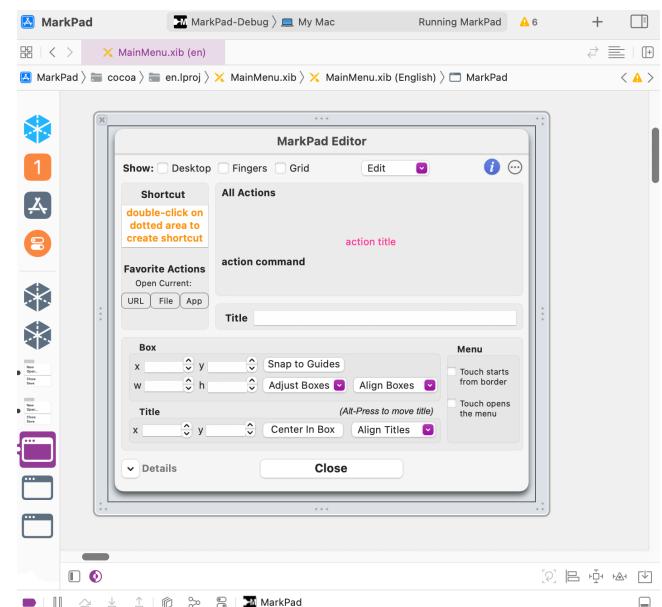
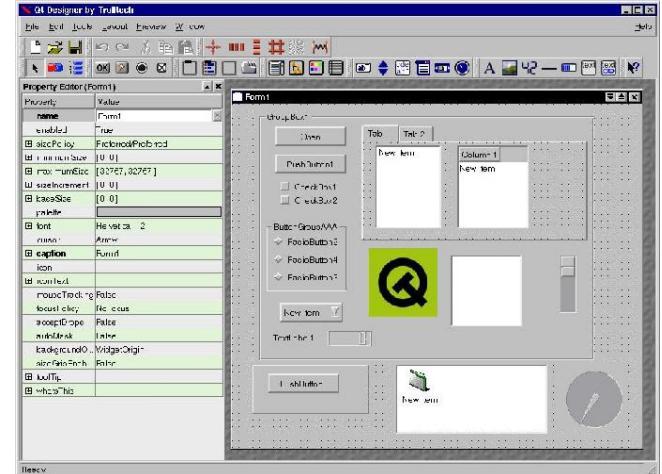
- Helpful for **prototyping**
- Importantly, can be used by **non-programmers**



Visual tools

Limitations

- Mostly useful for creating **static** user interfaces (menus, forms...)
- Little support for **controlling the interaction**, drag & drop, **dynamic** aspects...
- Layout not **localizable** if drag & drop is used
- Restrictions on **generated code**: can't be edited freely



Dedicated languages

User Interface Markup Languages

- Two kinds

1) Specify GUI structure

- **XAML** (Microsoft), **FXML** (JavaFX), **XUL** (Mozilla), etc.
- Compatible with visual tools
- Most of **interaction** specified elsewhere (Javascript, Typescript)

2) Specify structure *and* interaction

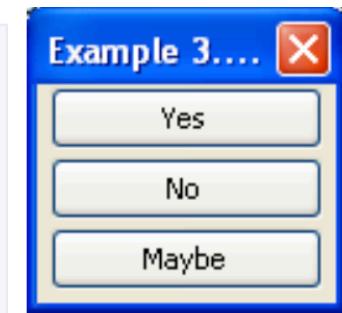
- **UIML**, **QML** (Qt), etc.

Dedicated languages

XUL (Mozilla)

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window id="vbox example" title="Example 3...."
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.x
ul">
  <layout>
    <button id="yes1" label="Yes" />
    <button id="no1" label="No" />
    <button id="maybe1" label="Maybe" />
  </layout>
</window>
```



Dedicated languages

QML (Qt Modeling Language)

- **Declarative** language for specifying GUIs
- Simplifies **interaction** and **animations**
- Allows inline **JavaScript** code

```
Rectangle {  
    id: canvas  
    width: 250  
    height: 200  
    color: "blue"  
  
    Image {  
        id: logo  
        source: "pics/logo.png"  
        anchors.centerIn: parent  
        x: canvas.height / 5  
    }  
}
```

properties can reference
other properties

Bindings

- **Constraints** that avoid writing callback functions
- A property is **automatically updated** when the value of another property **changes**
- Similar mechanism in other GUI toolkits (e.g. **JavaFX**)

```
Rectangle {  
    width: otherItem.width  
    height: otherItem.height  
}
```

```
Rectangle {  
    function calculateMyHeight() {  
        return Math.max(otherItem.height, thirdItem.height);  
    }  
    anchors.centerIn: parent  
    width: Math.min(otherItem.width, 10)  
    height: calculateMyHeight()  
    color: width > 10 ? "blue" : "red"  
}
```

calls method
color changes when width changes

States

Objects can have states

Example:

- *myRect* is located at (0,0)
- In the ***moved state***, it is located at (50,50)
- Clicking within the mouse area changes the **state**

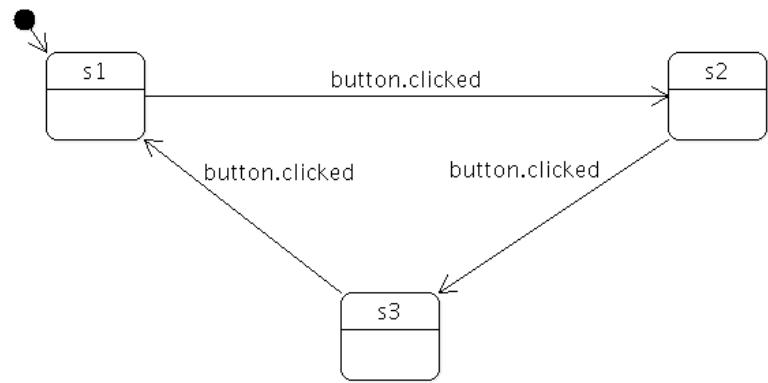
```
Item {  
    id: myItem  
    width: 200; height: 200  
  
    Rectangle {  
        id: myRect id  
        width: 100; height: 100  
        color: "red"  
    }  
    states: [  
        State {  
            name: "moved" state  
            PropertyChanges {  
                target: myRect  
                x: 50  
                y: 50  
            }  
        }  
    ]  
    MouseArea {  
        anchors.fill: parent  
        onClicked: myItem.state = 'moved'  
    }  
}                                            changes state on event
```

Statecharts

Can be drawn interactively

Example

- 3 state button



```
StateMachine {
    id: stateMachine
    // set the initial state
    initialState: s1

    // start the state machine
    running: true

    State {
        id: s1
        // create a transition from s1 to s2 when the button is clicked
        SignalTransition {
            targetState: s2
            signal: button.clicked
        }
        // do something when the state enters/exits
        onEntered: console.log("s1 entered")
        onExited: console.log("s1 exited")
    }

    State {
        id: s2
        // create a transition from s2 to s3 when the button is clicked
        SignalTransition {
            targetState: s3
            signal: button.clicked
        }
        // do something when the state enters/exits
        onEntered: console.log("s2 entered")
        onExited: console.log("s2 exited")
    }
}
```

Animations

Properties can
be animated

```
transitions: [           transition that triggers an animation
  Transition {
    from: "*"
    to: "moved"
    NumberAnimation { properties: "x,y"; duration: 500 }
  }
]
```

```
Rectangle {
  id: rect
  width: 120; height: 200

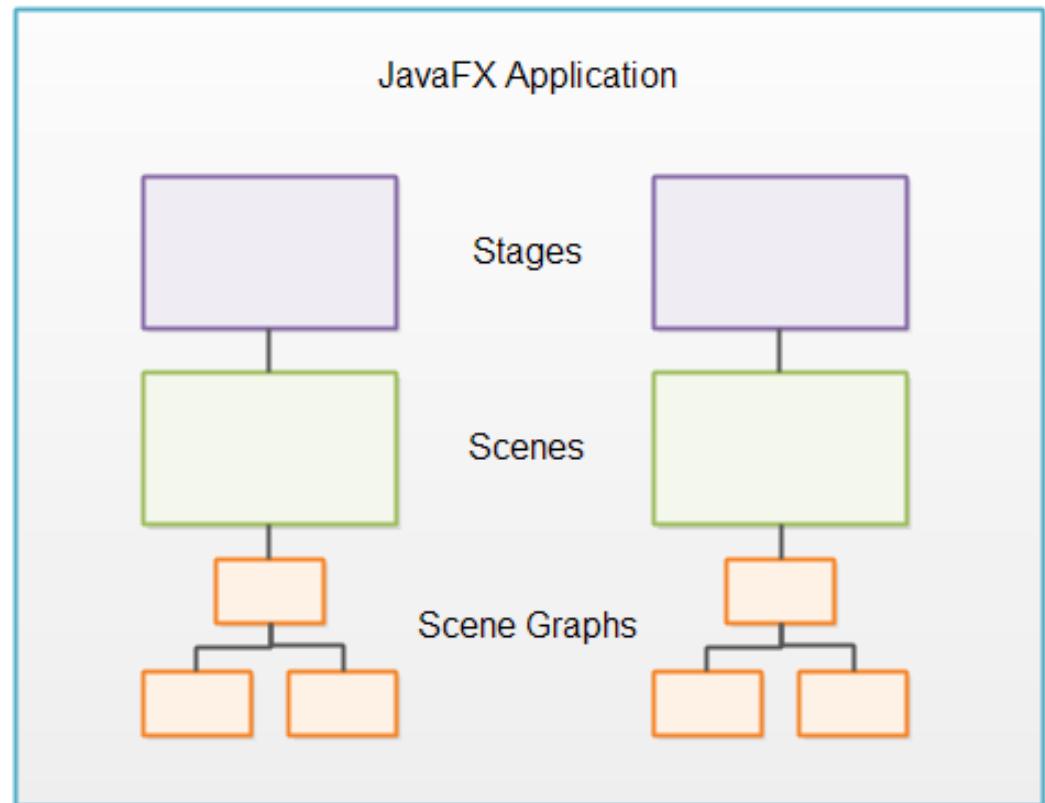
  Image {
    id: img
    source: "pics/qt.png"
    x: 60 - img.width/2
    y: 0

    SequentialAnimation on y {
      loops: Animation.Infinite
      NumberAnimation { to: 200 - img.height; easing.type: Easing.OutBounce; duration: 2000 }
      PauseAnimation { duration: 1000 }
      NumberAnimation { to: 0; easing.type: Easing.OutQuad; duration: 1000 }
    }
  }
}
```

JavaFX

A GUI is a Scene Graph

- **Nodes :**
 - widgets, 2D and 3D shapes, images, medias, etc.
- **States :**
 - geometric transforms, visual effects, etc.
- **Effects :**
 - change the visual aspect



JavaFX

Widgets

- Similar to **Swing** widgets
- Styles can be specified using **CSSs**
- The GUI can be written in **FXML**
 - can contain small scripts
- **SceneBuilder** editing tool



Properties and binding

- Widgets have **properties**
- It's possible to **observe** when their value change
- They can be **bound** together => powerful **synchronization** mechanism
 - **Unidirectional binding**:
 - `label.textProperty().bind(text.textProperty());`
 - **Bidirectional binding** (2 ways):
 - `text2.textProperty().bindBidirectional(text.textProperty())`

```

import javafx.application.*;
import javafx.stage.*;
import javafx.scene.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.scene.control.*;

public class RolePlayer extends Application {
    TextField character;
    TextField actor;

    @Override public void start(Stage primaryStage) {
        character = new TextField();
        actor = new TextField();

        // omitted details ...

        Label role1 = new Label("The role of ");
        Label role2 = new Label();
        Label role3 = new Label(" will be played by ");
        Label role4 = new Label();

        role2.textProperty().bind(character.textProperty());
        role4.textProperty().bind(actor.textProperty());

        Label scene = new Label(pane);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Role Player");
        primaryStage.show();
    }
}

```



source: dummies.com