Programming of Interactive Systems

Anastasia.Bezerianos@lri.fr

(Nolwenn Maudet)

Week 4:
a. Peripherals, Software Architectures & MVC

Anastasia.Bezerianos@lri.fr

peripherals

structure of an interactive system

What we see
- output

What we act with
- input

visible part
« front end »

What happens
- treatment
- computation
- communication
- data (storage and access)

invisible part
« back end »

structure of an interactive system

Output
"bitmap" screens: cathode ray, LCD, Plasma, OLED.

Size expressed with their diagonal dimension in inches
(1 inch = 2.54 cm, 30 inches~76cm) and the width to height ratio (e.g., 16/9)

Resolution expressed in pixels (e.g., 2560x1600)

Resolution and size gives the density expressed in "dpi": dot[pixel] per inch
(typically 100 dpi ~ 40 pixels per cm, i.e. 1 pixel ~ 0.25 mm)

Color Resolution ("depth" RGB[A]):
8 bits (256 colors), 16 bits (65536 colors) or 24[32] bits (16 millions of colors [+256 levels of "translucence"])

Temporal Resolution expressed in Hz, the number of frames the screen can display per second (typically 60 Hz)

structure of an interactive system

Output

http://youtu.be/u7Gm0OeKxwU

structure of an interactive system

Input

- keyboards
- mice, tablets, joysticks, trackballs
- augmented pens
- speech recognition
- motion capture & computer vision
- interactive surfaces
  - (e.g., mobiles, tangibles)
- hybrid devices (input output)
  - force feedback devices
    http://youtu.be/REA97hRX0WQ
  - touch screens (e.g., vibration)
  - deformable or actuated displays
    http://youtu.be/ouP9xNujkNo

text entry

Input (text entry)

- problem: Optimization of key position
- Dvorak layout: 10 to 15% speed improvement and reduced fatigue compared to Qwerty

- Software keyboard:
  optimization ➔ pointing
- dpy keyboard with keys that have led screens (oled) or projection on keyboard
  http://youtu.be/fhBH6KW2aT4

text entry

Input (alternative to classical keyboard)

- Chord keyboards:
  - few keys (4 or 5)
  - use of multiple keys simultaneously
  - fast input with one hand

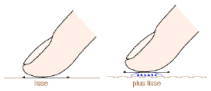
- Mobile phone keyboards:
  - multi tap
  - input can be slow
  - T9 system: one tap per letter can suggest/add words

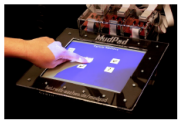
3D and tactile peripheral devices

Input

- 3D peripherals

- haptic/tactile feedback: vibrations, surface changes
  - vibrations when we pass over some targets, can we reproduce true textures?





type and control of peripheral devices

Absolute : transmit a position (x, y)
Examples : tablets, touch screens, optical pens

Relative: transmit a displacement (dx, dy)
Examples : mice, joysticks, trackballs

Order 0: input device displacement corresponds to a displacement of an object
Example : mouse – cursor pair

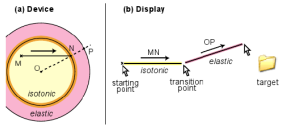
Order 1: input device controls the speed of an object
Example : joystick – cursor pair

Isotonic devices: control position – Order 0, and use clutching for long distances
Examples : mice, touchpads

Elastic devices: have a stable state, and «elastic» return to it
Examples : joystick (tilt ➔ speed)

an isotonic and elastice device

RubberEdge (Casiez et al. 2007): reduce clutching by combining positional and elastic control



Center of touchpad: position
Borders of touchpad: an elastic system for controlling speed displacement

http://youtu.be/kucTPG_zTik

input/output devices: Control – Display Gain

Resolution : number of pulses that the input device can send for a given distance.

dpi = dot[pulses] per inch
[1 inch = 2.54 cm]

Example mouse: between 300dpi (slow), 600dpi (typical) and 2400dpi (max?), or respectively one pulse every 0.083, 0.042 and 0.01 mm

Controle-Display Gain :

$$\text{CDGain} = \frac{\text{Distance traversed by pointer on display}}{\text{Distance traversed by input device}}$$

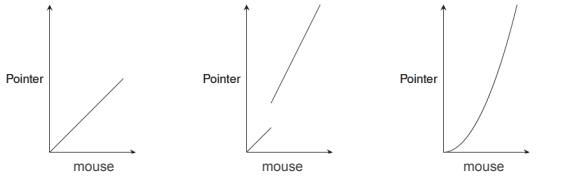
Examples:
Tablet with same size as display, and direct/absolute: CDGain = 1
Screen 100 dpi, mouse 600 dpi and one pulse per pixel: CDGain=6

1 inch mouse = 600 dpi
= 600 pixels of movement
= 6 * 100 dpi screen res.
= 6 inces on screen

input/output devices: acceleration

Problem: if CDGain is too large, accuracy is hard. If it is too small we need to clutch our mouse to travel large distances

Acceleration: dynamic adjustment of CDGain as a function of mouse speed. The faster we move, the bigger the CDGain



c_i Constant CDGain

X Windows:
2 values for CDGain

MacOS X and Windows:
Progressive chance of CDGain

software architecture, MVC

structure of an interactive system

What we see
- output

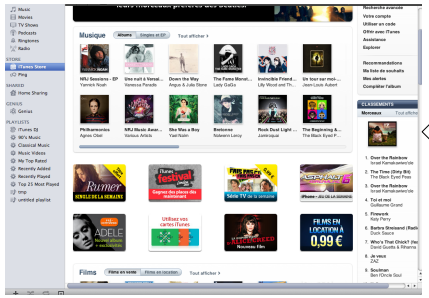
What we act with
- input

visible part
« front end »

What happens
- treatment
- computation
- communication
- data (storage and access)

invisible part
« back end »

example 1

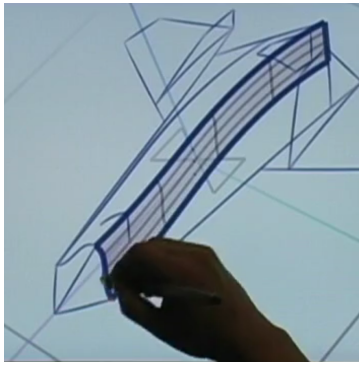


front end ⇔

- data model (albums, artists, categories, etc.)
- communication with iTunes server
- manage queries
- manage sales
- security

back end

example 2



front end ⇔

- geometric models
- calculations (transformations, rendering, etc.)
- store and access designs

back end

example 3

front end ⇔

- tabular structure
- storage and data access

back end

link between the two parts

… programming using an organization model

organize, structure an interactive application by separating:

- Data and their treatment: **the Model**
- Data representation: **the View**
- Application behavior to input: **the Controller**

Model «Model–View–Controller» (MVC)

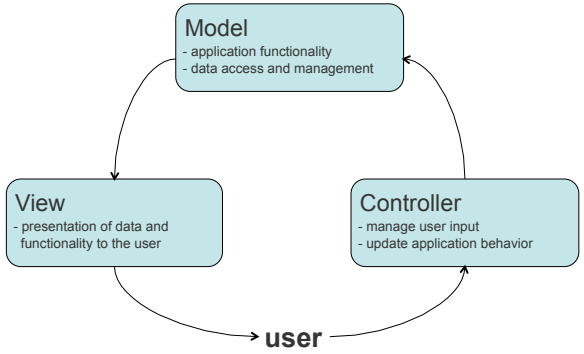
MVC is :

- A *design pattern* (standardized design solution independent of programming language)
- A *software architecture* (a way to structure an application or a set of software packages)

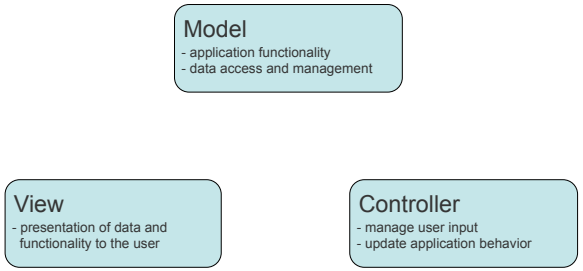
Introduced in 1979 by Trygve Reenskaug

Strongly linked to OO programming (Smalltalk)

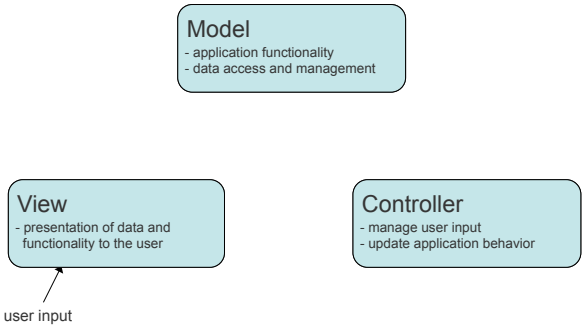
MVC : *ideal* interactions between components



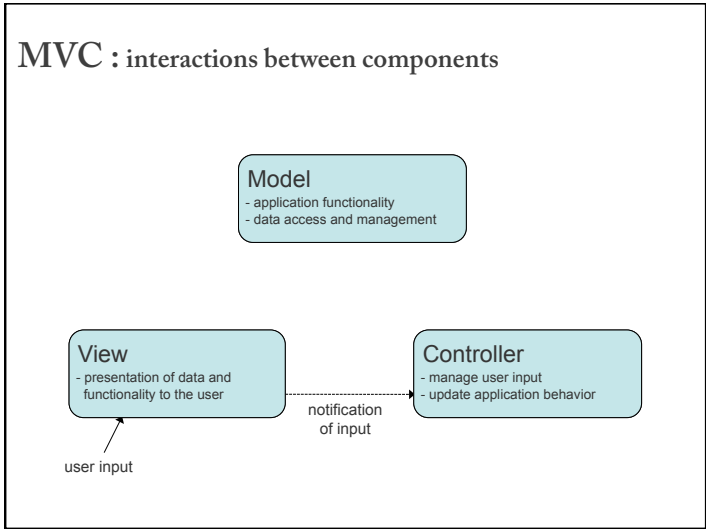
MVC : interactions between components



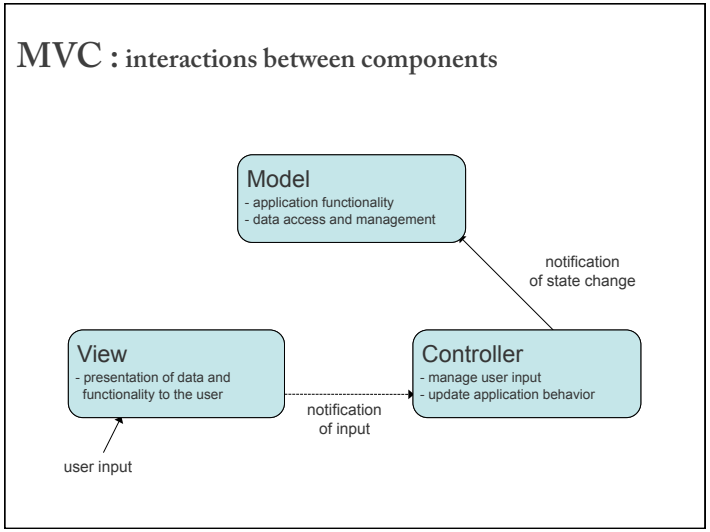
MVC : interactions between components



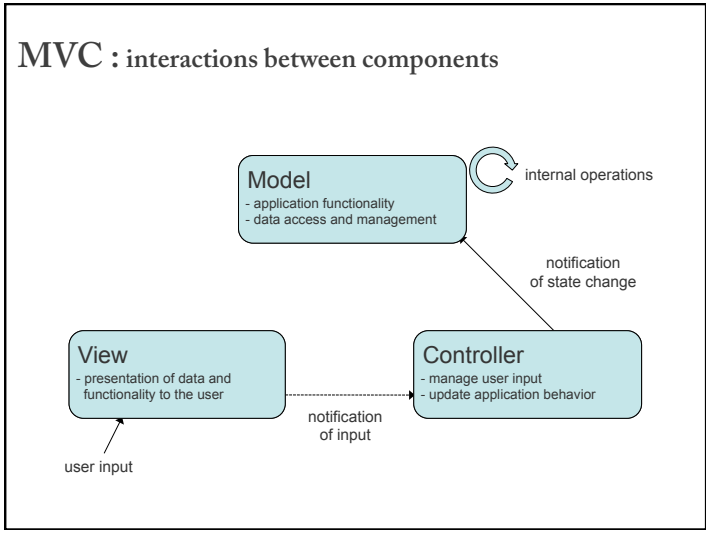
MVC : interactions between components



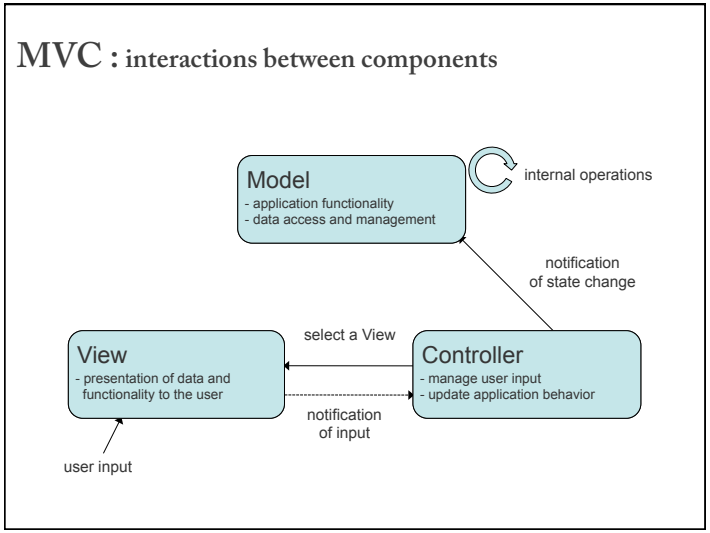
MVC : interactions between components



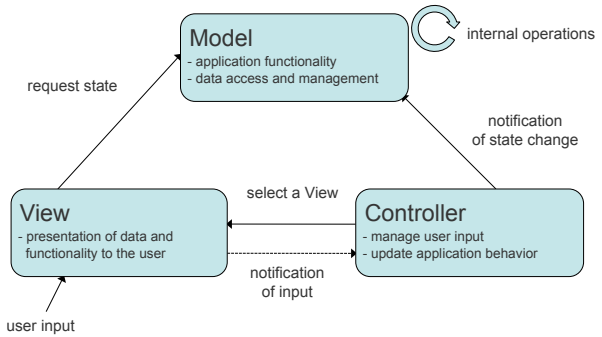
MVC : interactions between components



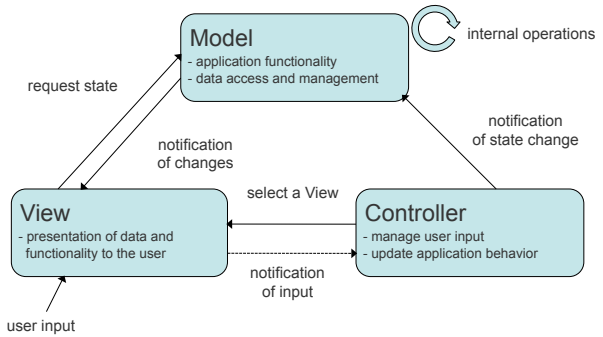
MVC : interactions between components



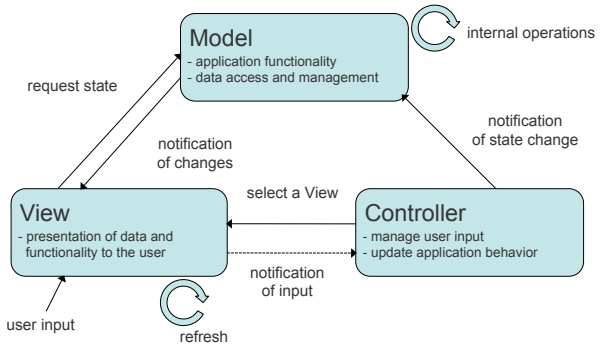
MVC : interactions between components



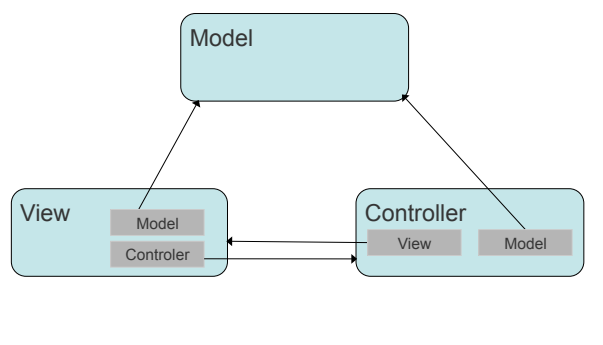
MVC : interactions between components



MVC : interactions between components



MVC : referencing between components



MVC : the model

The model:

- Represents data
- Gives access to data
- Gives access to data management functionality
- Exposes the application functionality

Functional layer of the application

MVC : the view

The view:

- Shows the (or one) representation of the data in the model
- Ensures consistency between data representation and their state in the model (application)

Output of the application

MVC : the controller

The controller:

- Represents the application behavior w.r.t. user actions
- Translates user actions to actions on the model
- Calls the appropriate view w.r.t. the user actions and the model updates

Effect and treatment of input

advanteges of MVC

Clean application structure

Adapted to concepts of O-O programming

Independence of
data – representation – behavior

Modular and **reusable**

disadvantages of MVC

Implementation complex for large applications

Too many calls between components
- « Spaghetti » code

Controller and View are often tightly linked to Model (and often to each other)

⇨ **need to adapt implementation**

naming conventions

Packages:

Controllers	`package application.controllers;`
View	`package application.views;`
Model	`package application.models;`

Classes:

Controllers	`ControllerNameClass.java`
View	`ViewNameClass.java`
Model	`ModelNameClass.java`

MVC and Java Swing

Model-View-Controller separation not strict

Even for simple widgets
- **Model** : abstract behavior of widget
- **View & Controller** : Look & Feel + Listener

Examples : JButton, JLabel, JPanel, etc.

Most often we do not touch the model of widgets
- Swing uses a model by default for each widget

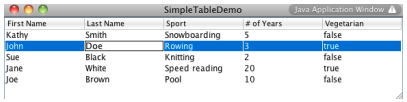
Swing : types of models

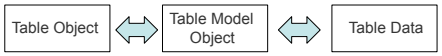
Look & Feel
- Interfaces : ButtonModel, ListSelectionModel
- Default Classes : DefaultButtonModel, DefaultListSelectionModel

Data
- Interfaces : ListModel, TableModel, TreeModel
- Default Classes : DefaultListModel, DefaultTableModel, DefaultTreeModel

Look & Feel + Data
- For some widgets
- Examples : BoundedRangeModel for JSlider

example





javax.swing.JTable *javax.swing.table.TableModel*

example

The data

```
Object[][] data = {
     {"Kathy", "Smith","Snowboarding", new Integer(5), new Boolen(false)},
     {"John", "Doe", "Rowing", new Integer(3), new Boolean(true)},
     {"Sue", "Black","Knitting", new Integer(2), new Boolean(false)},
     {"Jane", "White","Speed reading", new Integer(20), new
             Boolean(true)},
     {"Joe", "Brown","Pool", new Integer(10), new Boolean(false)}
};
```

example

The model

```
class MyTableModel extends AbstractTableModel {
     private String[] columnNames = …
     private Object[][] data = …

     public int getColumnCount() {
             return columnNames.length;
     }

     public int getRowCount() {
             return data.length;
     }

     public String getColumnName(int col) {
             return columnNames[col];
     }

     public Object getValueAt(int row, int col) {
             return data[row][col];
     }
     …
}
```

example

The view

```
TableModel dataModel = new MyTableModel();

JTable table = new JTable(dataModel);
JScrollPane scrollpane = new JScrollPane(table);
```

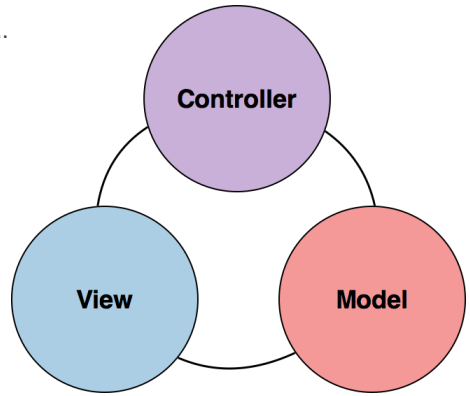
example: revisit our thermometer (java)

App simulating a thermometer, where users can control the temperature

App:
- shows current temperature measured (°C,°K,°F)
- has a controller for changing temperature
- has a controller for changing measuring unit

MVC and Java Swing

An example …



MVC and Java Swing

Problems in terms of the MVC model?
How close is the actual implementation?

Did we even follow the non-ideal diagram?

Are the widgets part of the View or the Controller?

Could the View be part of …