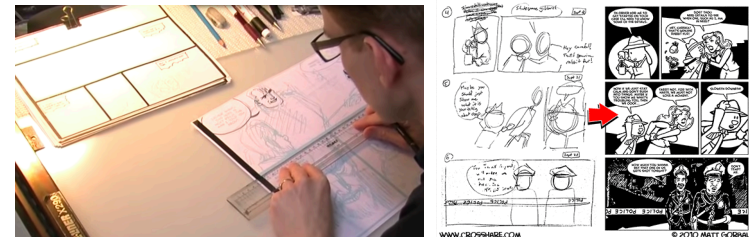


## UI software architectures & Modeling interaction

(part of this content is based on previous classes from A. Bezerianos, S. Huot, M. Beaudouin-Lafon, N. Roussel, O. Chapuis)

## Assignment 1

Design and implement an interactive tool for creating the layout of comic strips



<https://www.lri.fr/~fanis/teaching/ISI2014/assignments/ass1/>

## Software architecture - MVC

### structure of an interactive system

What we see

- output

What we act with

- input

What happens

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



visible part  
« front end »

invisible part  
« back end »

## example 1

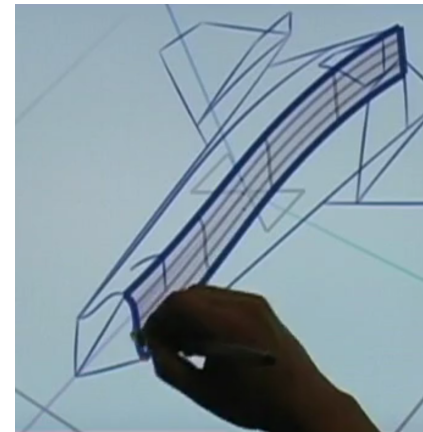


front end

- 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

ID	Name	Email	Address
6625	JUANITA LAMBERT	lmbasley@evernet1.biz	139 MANNING HWY,CLYD,79554
6740	GREG CABRERA	cholder@tmail.biz	736 GENESSEE BLVD,CORDELE,17433
6599	ALISA WISE	lheyee@eyecode.net	205 ALICE RD,CABELLA,14855
9282	SHARON WINTERS	jogand@tmail.com	955 COHEN PIKE,TYRONE,811
2150	KRISTY FRANKS	lgates3@sonama1.com	1471 ALEXIS PKWY,BALDWIN,85
1927	JEFF RICE	dieidits@tmail.org	104 DUNDEE PKWY,HOGANVILLE,3741
7972	TAMARA BRYANT	hotwom@evernet1.us	1382 WOGAN BLVD,CITY OF CALHOUN,43790
5824	ALISHA YANG	foundwong@holmat1.net	716 HOGANS DR,HARDING,58932
1402	JASON NGUYEN	hweethers@tmail.com	637 MICHAEL CRES FORT STEWART,14864
3620	LINDSEY CABRERA	askeddreams@tmail.com	1420 LAZELERE HTS,FORT STEWART,21650
3511	ANTHONY MATHEWS	ctarrel@evernet1.co.uk	1325 OKEY LN,THUNDERBOLT,63656
572	JARED FORD	wrnying@tmail.co.uk	122 EUCALID RD,FORTH,42014
6720	ALTIMIN WILLIAMS	roomwhere@tmail1.net	260 HILL PARK,NORCROSS,58355
3447	MARION BROWN	hwagner@b1tmail.co.uk	739 MIDDLE PATH,MACON,9944
9356	HOPE HAYNES	saerifesi@eyecode.com	1023 COOPERRIDERS CRES,STATENVILLE,342
2259	JEANNIE RANDOLPH	wormto@eyecode.net	672 EDISON PATH,CENTERVILLE,48580
6264	RACHEL CONLEY	ismokewhite2@eyecode.net	1223 STEVENS CT,CARRROLLTON,40084
4224	TRACIE DAVENPORT	evmitaser@eyecode.net	548 VALLEY FAULBRIESTOL,63113
1836	NANCY GOFF	holdfire@gmail1.us	147 FLEMING HWY,MORA,37823
9284	KATHY MORENO	hadcode@evernet1.co.uk	1341 BYRNE DR,SUMNER,60450

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)

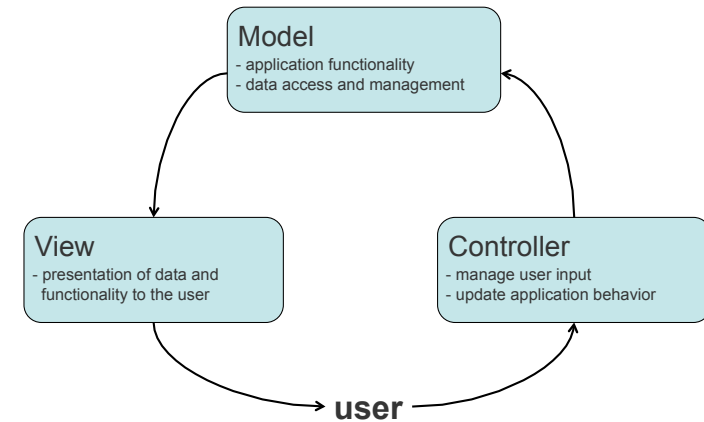
MVC is :

- 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)

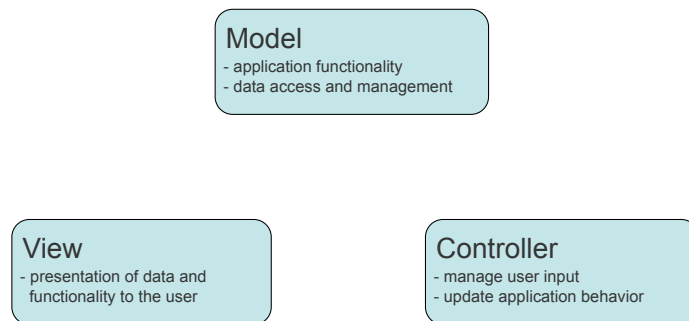
Introduced in 1979 by Trygve Reenskaug

Strongly linked to OO programming (Smalltalk)

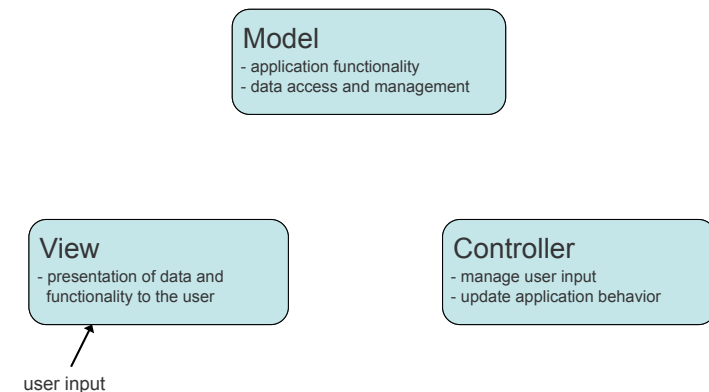
## MVC: *ideal* interactions between components



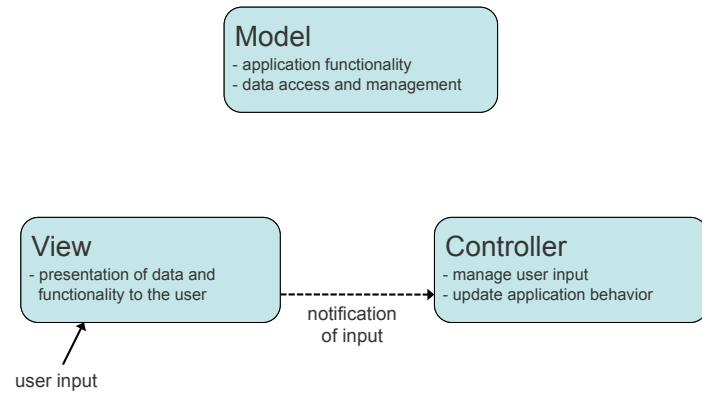
## MVC: interactions between components



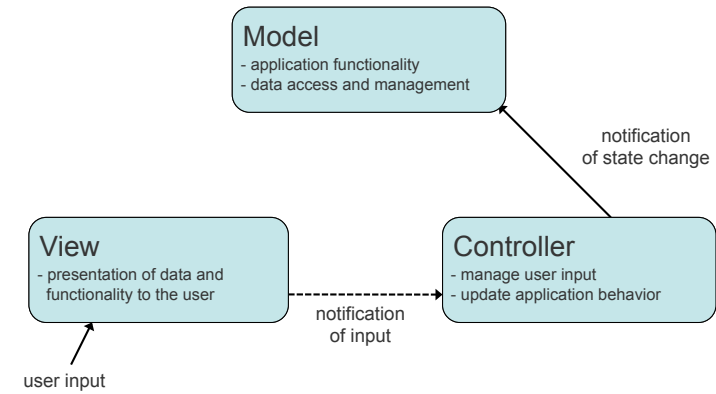
## MVC: interactions between components



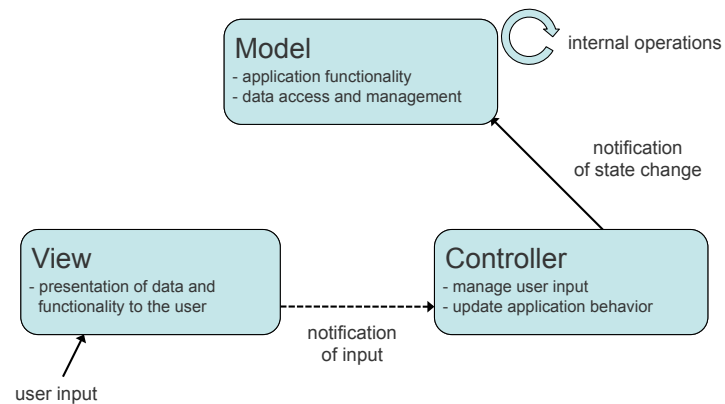
## MVC: interactions between components



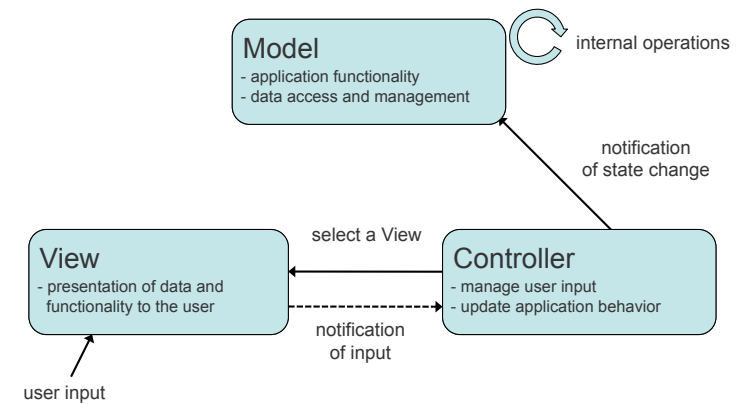
## MVC: interactions between components



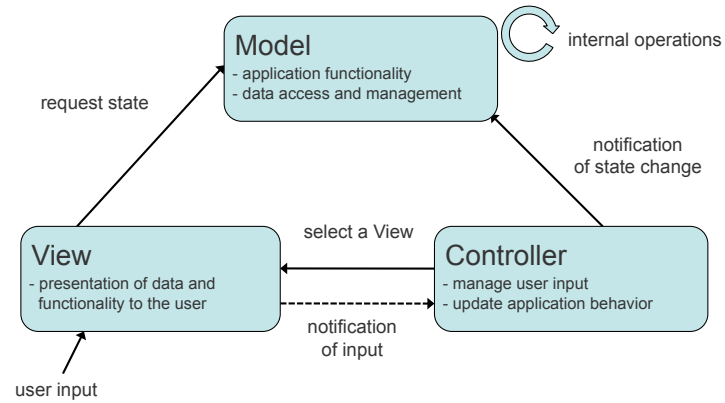
## MVC: interactions between components



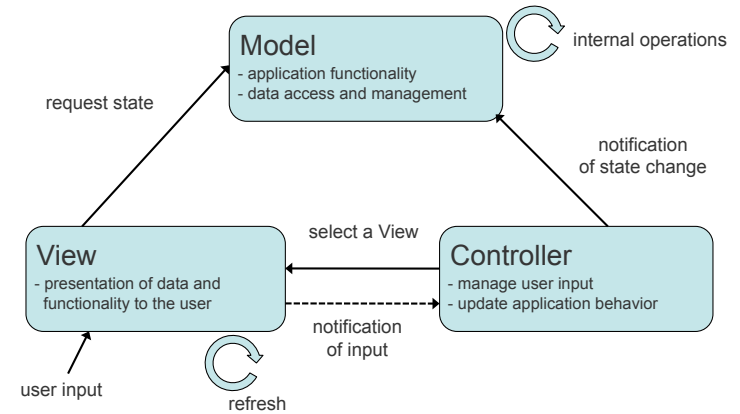
## MVC: interactions between components



## MVC: interactions between components

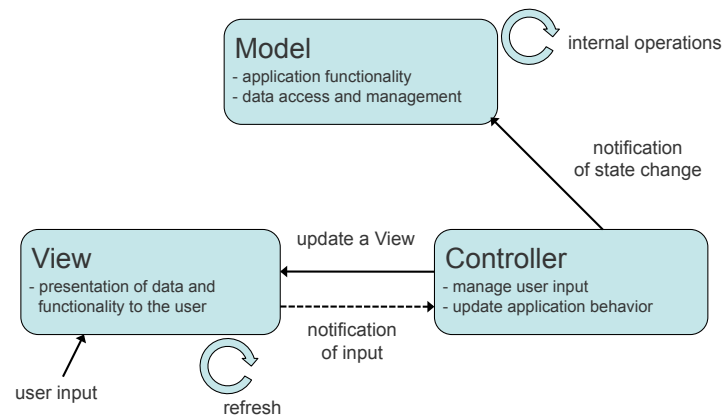


## MVC: interactions between components

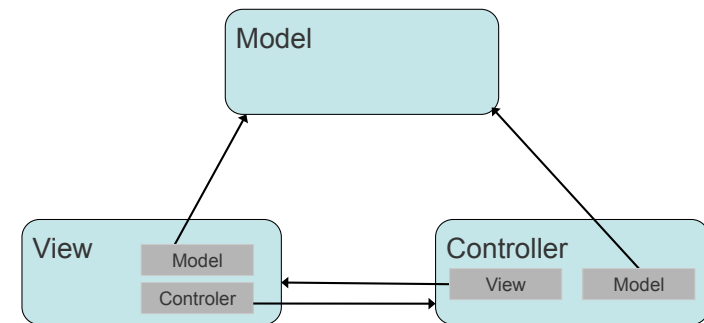


## MVC: interactions between components

### Alternative architecture



## 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 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**

## 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**

## advantages 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

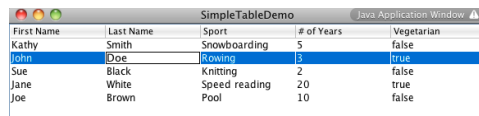
Too many calls between components

- « Spaghetti » code

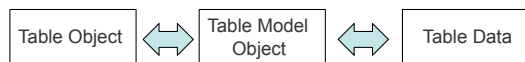
Controller and View are often tightly linked to Model (and often to each other)

➡ need to adapt implementation

## example



First Name	Last Name	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboarding	5	false
John	Doe	Rowing	3	true
Sue	Black	Knitting	2	false
Jane	White	Speed reading	20	true
Joe	Brown	Pool	10	false



`javax.swing.JTable`      `javax.swing.table.TableModel`

## MVC and Java Swing Widgets

Model-View-Controller separation not strict

Model categories:

Visual status of GUI controls, e.g., pressed or armed button

Application-data model, e.g., text in a text area

Swing uses a model by default for each widget

View & Controller (often part of the same UI object)

Look & Feel + Listener

Examples : JButton, JLabel, JPanel, etc.

## example

The data

```
Object[][] data = {  
    {"Kathy", "Smith", "Snowboarding", new Integer(5), new Boolean(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

### The controller

```
public class MySelectionListener implements ListSelectionListener {
    private JTable table;

    public MySelectionListener(JTable table){
        this.table = table;

        table.setCellSelectionEnabled(true);
        ListSelectionModel cellSelectionModel = table.getSelectionModel();
        cellSelectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        cellSelectionModel.addListSelectionListener(this);
    }

    public void valueChanged(){
        ...
    }
}
```

## Modeling Interaction



# WIMP interfaces

WIMP: Window, Icons, Menus and Pointing

Presentation

- Windows, icons and other graphical objects

Interaction

- Menus, dialog boxes, text input fields, etc

Input

- pointing, selection, ink/path

Perception-action loop

- feedback

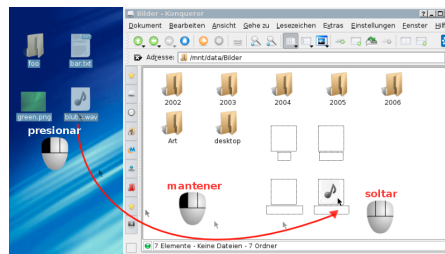


## direct manipulation: examples

Editing documents WYSIWYG: What You See Is What You Get  
text editors (e.g., Word, OpenOffice)  
bitmap/vector graphics (e.g., Photoshop, Illustrator).  
Counter-example: LaTeX ...

Icon interaction:

- Generic interface
- Use of metaphors
- drag-and-drop

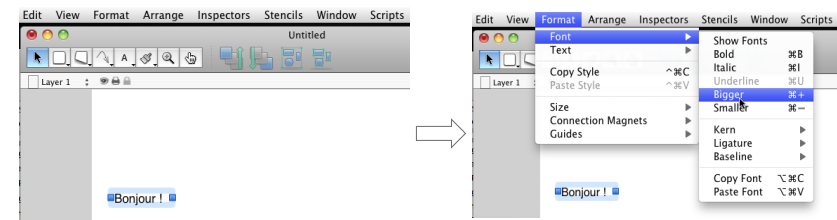


## direct manipulation

Ben Shneiderman (1983)

1. Persistent representation of objects of interest
2. Use of physical actions instead of complex syntax
3. Operations are quick, incremental, reversible, and their effect on objects is immediately visible (**feedback**)
4. Incremental learning, to permit use of the interface with little prior knowledge

## direct manipulation?



## direct manipulation problems

Identifying objects of interest

- example: styles in Word

Immediate feedback difficult when there is a delay between action and result

Direct or indirect manipulation?

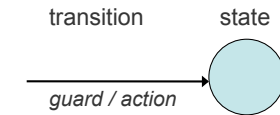
- menus, dialog boxes, scroll-bars, etc.

## describing interactions: state machines

Finite Automata

State = interaction state

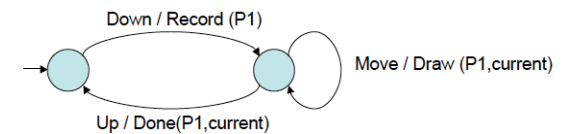
Transition = input events



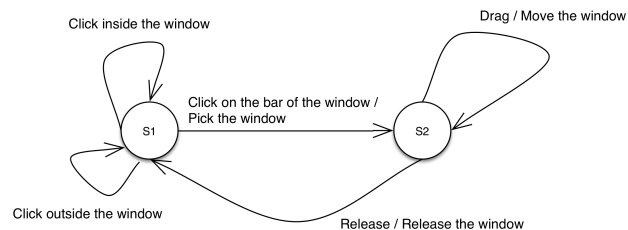
State Machine

- boolean expressions of events associated to transitions (guard)
- actions associated to transitions (not always present)

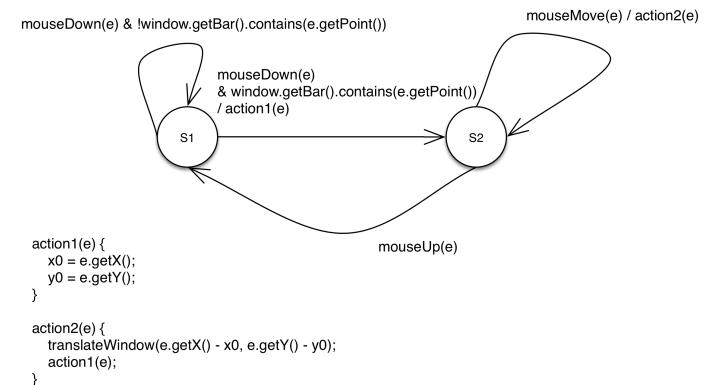
Example:



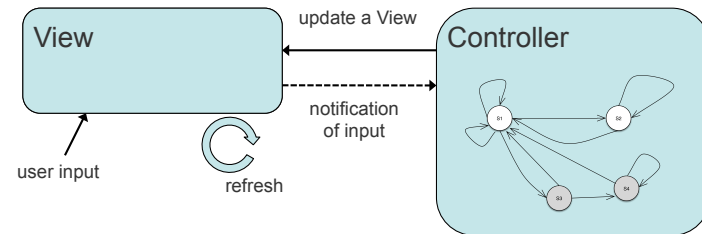
## example: dragging windows



## example: dragging windows



## state machines & MVC



## representing states

Common approach: use of global variables within a controller

```
public enum State {S1, S2, S3, S4}
private State state = State.S1;
```

**or (use of multiple variables)**

```
private boolean buttonPressed = false, mouseMoved = false;
```

In the following lecture, we'll introduce *SwingStates*, a Java library for modeling interaction through states, state transitions, and state machines

## common problems

Getting trapped to states with no transitions (deadlocks)

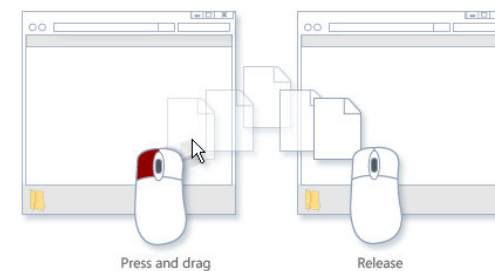
Maintaining the code to capture new or unforeseen states is usually hard

An interaction can involve several UI components. Not always clear how to divide interaction between multiple controllers and state machines.

## drag & drop

Which UI objects are involved?

Which controller handles this interaction?



## interaction modes

Mode: distinct state of the UI where the same user input has a different interpretation

- text vs. drawing mode in an editing tool
- typing capital or small characters

Mode switching

- e.g., Caps lock key, specialized button

Quasimode: mode being active through some constant action from the user

- e.g., use of modifier keys such as Shift, Alt, Control while typing or pointing

## interaction modes: problems

« modes are a significant source of errors, confusion, unnecessary restrictions, and complexity in interfaces »

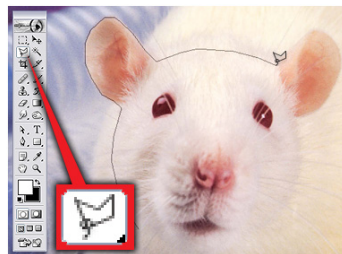
Jef Rusklin

Ruskin advocated for modeless interfaces. He also recommended the use of quasimodes instead of explicit modes.

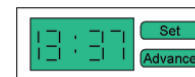
Other points of view (Jacob Nielsen)

- « users cannot cope with everything at once »
- «...need the interface to narrow their attention »
- « Real life is highly moded »

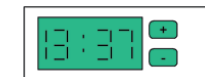
## making modes visible



## eliminating modes



Special mode for changing time



No modes, direct editing

(credits to Niall Murphy)

What are the trade-offs in these designs?

