Programming of Interactive Systems

Anastasia.Bezerianos@lri.fr

Week 4:
a. Peripherals, Software Architectures & MVC

Anastasia.Bezerianos@lri.fr

peripherals

structure of an interactive system

What we see/perceive

- 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
(100 dpi ~ 40 pixels per cm, i.e. 1 pixel ~ 0.25 mm; 300 dpi ~ 118 pixels per cm, 1 pixel ~ 0.08 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

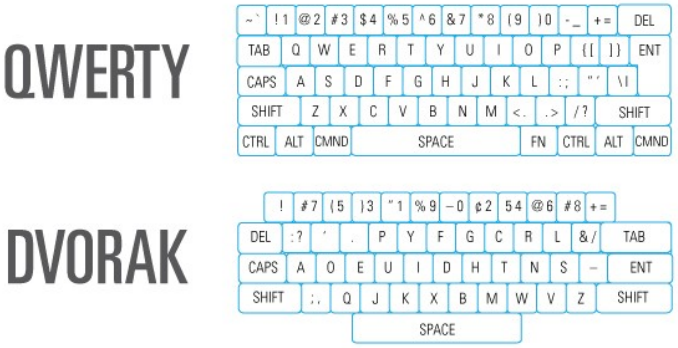
    http://youtu.be/ouP9xNujkNo

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

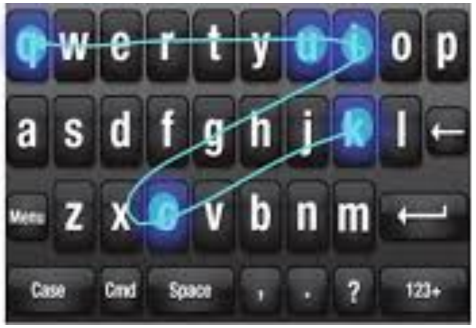
text entry

Input (alternative to classical keyboard)

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

1
ABC 2
DEF 3
GHI 4
JKL 5
MNO 6
PQRS 7
TUV 8
WXYZ 9
*
0
#

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

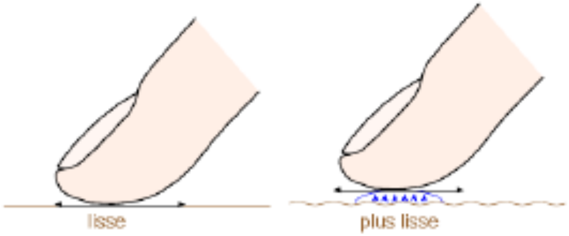


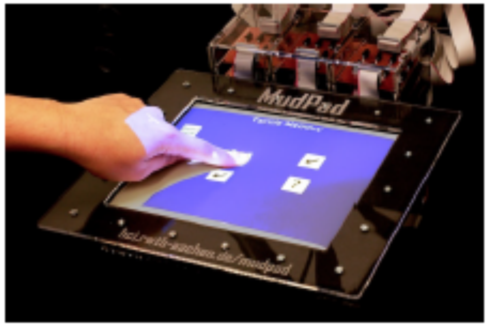
3D and tactile peripheral devices

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 a (virtual) 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

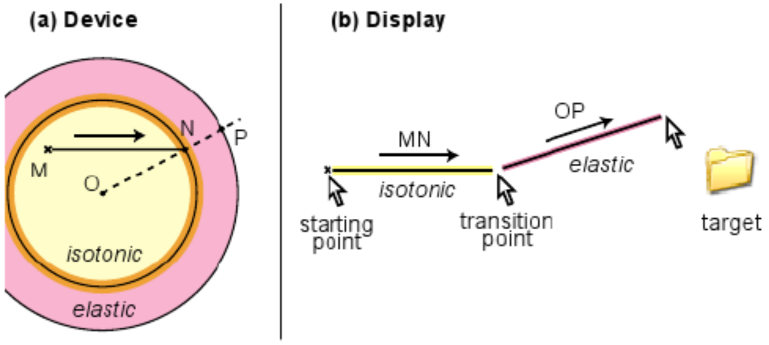
Examples : mice, touchpads

Elastic devices: have a stable state, and «elasticly» return to it

Examples : joystick (tilt ➔ speed)

an isotonic and elastic 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 (high), or respectively one pulse every 0.083mm, 0.042mm and 0.01 mm

Control-Display Gain :

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

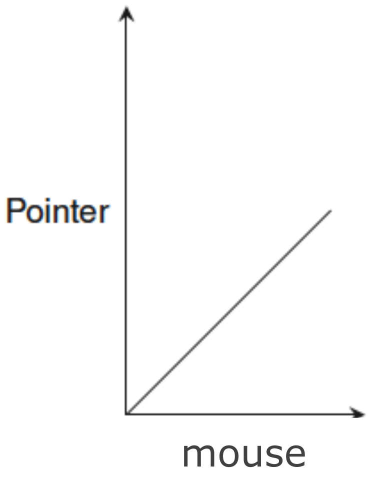
Examples:
Tablet with same size as display, and direct/absolute: CDGain = 1
Screen 100 dpi, mouse 600 dpi and one pulse per pixel: CDGain=6

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

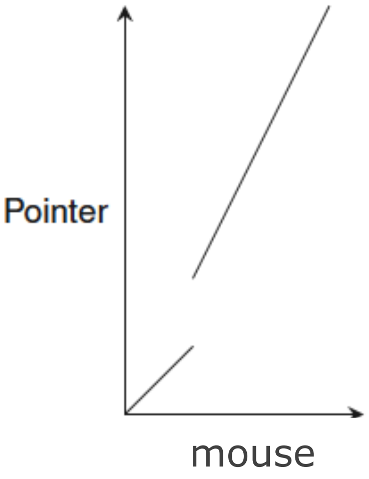
input/output devices: acceleration

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

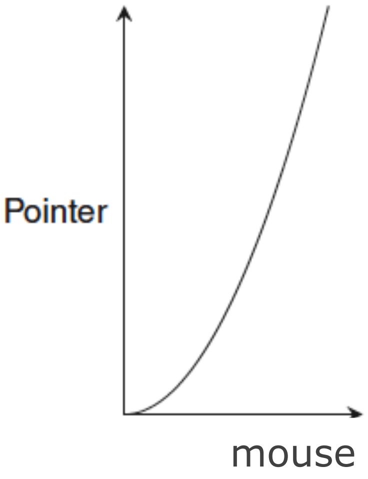
Acceleration: dynamic adjustment of CDGain as a function of mouse speed. The faster we move, the bigger the CDGain



Constant CDGain



X Windows:
2 values for CDGain



MacOS X and Windows:
Progressive change of CDGain

software architecture, MVC

structure of an interactive system

What we see

- output

What we act with

- input

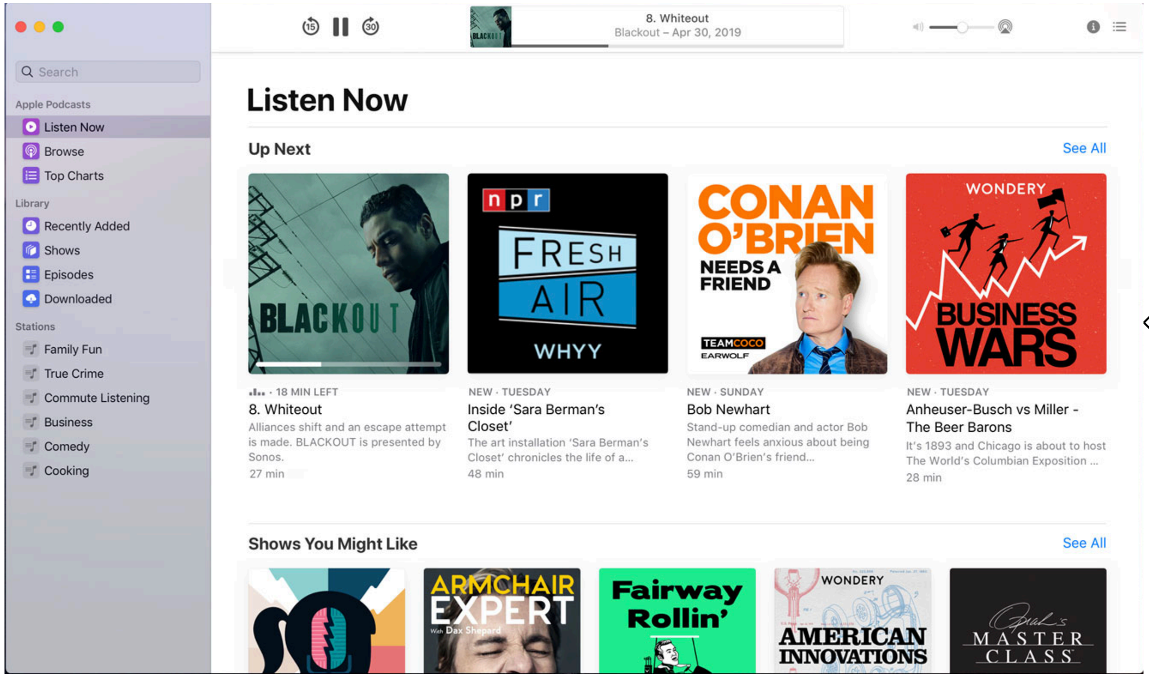
visible part
« front end »

What happens

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

invisible part
« back end »

example 1a



18 MIN LEFT
8. Whiteout
Alliances shift and an escape attempt is made. BLACKOUT is presented by Sonos.
27 min

NEW · TUESDAY
Inside 'Sara Berman's Closet'
The art installation 'Sara Berman's Closet' chronicles the life of a...
48 min

NEW · SUNDAY
Bob Newhart
Stand-up comedian and actor Bob Newhart feels anxious about being Conan O'Brien's friend...
59 min

NEW · TUESDAY
Anheuser-Busch vs Miller - The Beer Barons
It's 1893 and Chicago is about to host The World's Columbian Exposition ...
28 min

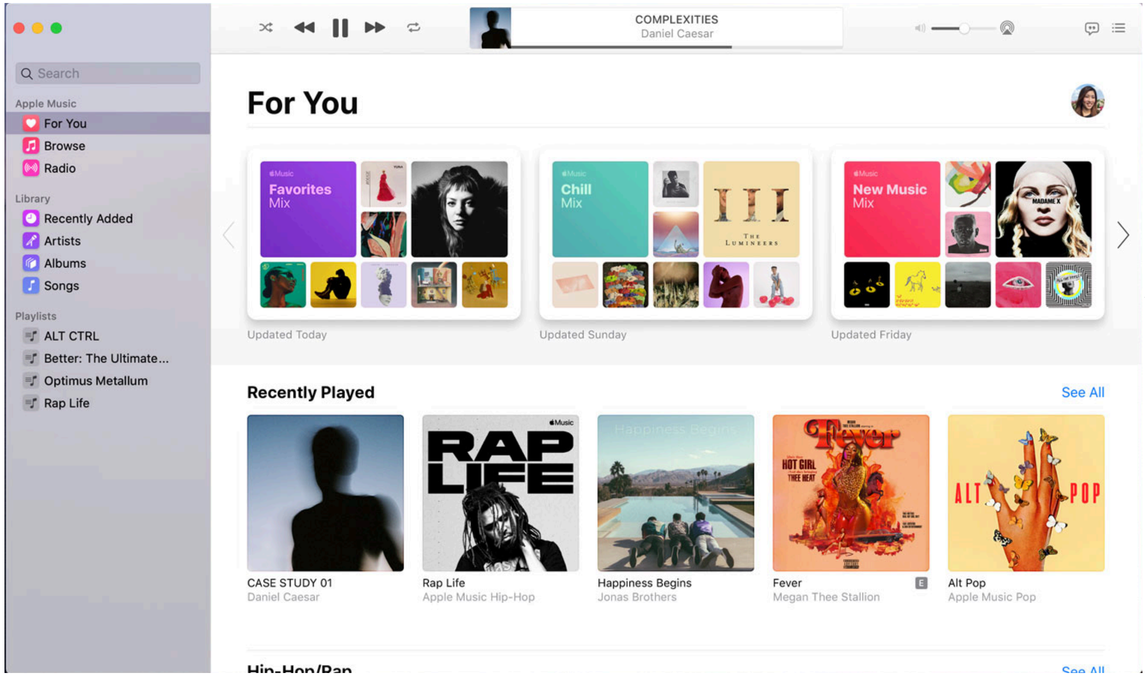
⇔

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

back end

front end

example 1b



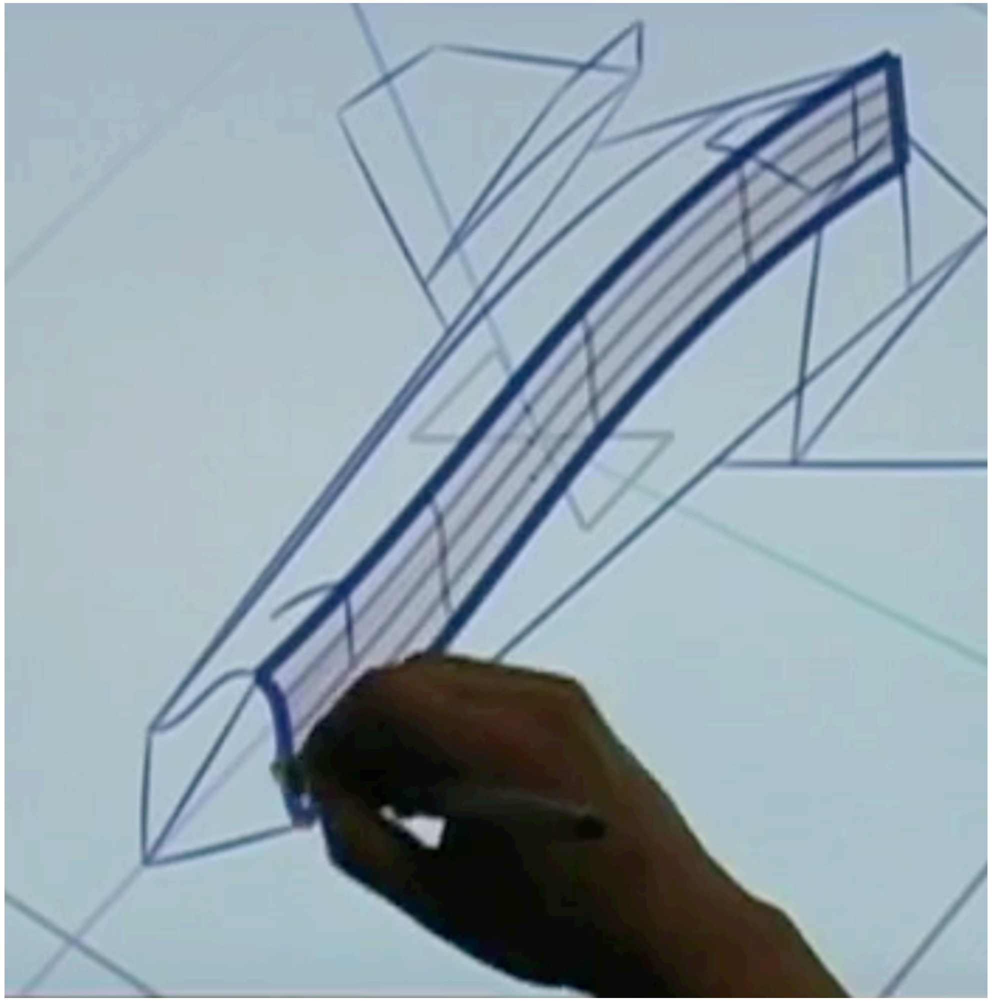
⇔

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

back end

front end

example 2

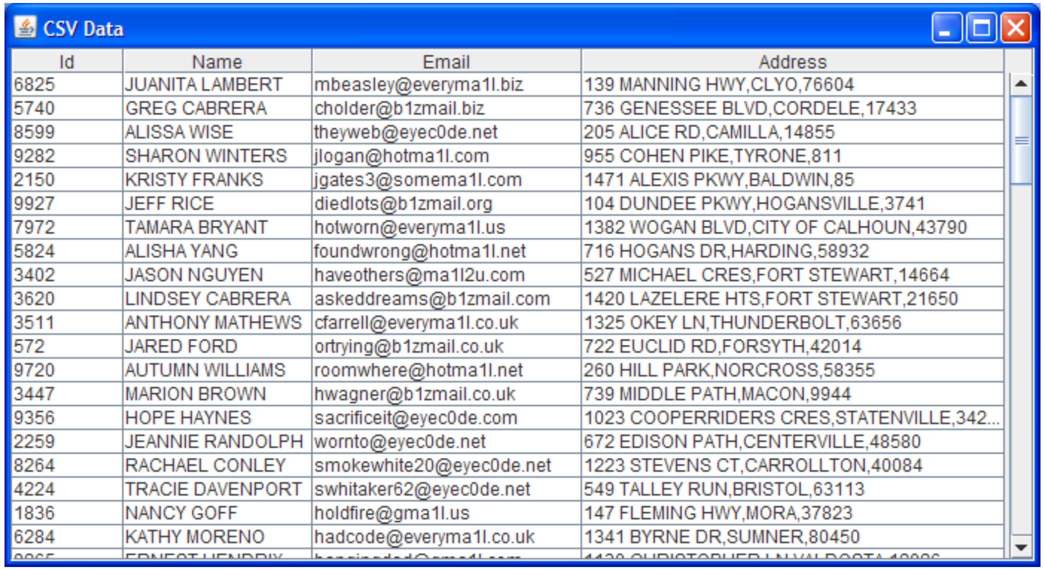


front end

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

back end

example 3



CSV Data

Id	Name	Email	Address
6825	JUANITA LAMBERT	mbeasley@everyma1l.biz	139 MANNING HWY,CLYO,76604
5740	GREG CABRERA	cholder@b1zmail.biz	736 GENESSEE BLVD,CORDELE,17433
8599	ALISSA WISE	theyweb@eyec0de.net	205 ALICE RD,CAMILLA,14855
9282	SHARON WINTERS	jlogan@hotma1l.com	955 COHEN PIKE,TYRONE,811
2150	KRISTY FRANKS	jgates3@somema1l.com	1471 ALEXIS PKWY,BALDWIN,85
9927	JEFF RICE	diedlots@b1zmail.org	104 DUNDEE PKWY,HOGANSVILLE,3741
7972	TAMARA BRYANT	hotworn@everyma1l.us	1382 WOGAN BLVD,CITY OF CALHOUN,43790
5824	ALISHA YANG	foundwrong@hotma1l.net	716 HOGANS DR,HARDING,58932
3402	JASON NGUYEN	haveothers@ma1l2u.com	527 MICHAEL CRES,FORT STEWART,14664
3620	LINDSEY CABRERA	askeddreams@b1zmail.com	1420 LAZELERE HTS,FORT STEWART,21650
3511	ANTHONY MATHEWS	cfarrell@everyma1l.co.uk	1325 OKEY LN,THUNDERBOLT,63656
572	JARED FORD	ortrying@b1zmail.co.uk	722 EUCLID RD,FORSYTH,42014
9720	AUTUMN WILLIAMS	roomwhere@hotma1l.net	260 HILL PARK,NORCROSS,58355
3447	MARION BROWN	hwagner@b1zmail.co.uk	739 MIDDLE PATH,MACON,9944
9356	HOPE HAYNES	sacrificeit@eyec0de.com	1023 COOPERRIDERS CRES,STATENVILLE,342...
2259	JEANNIE RANDOLPH	wornto@eyec0de.net	672 EDISON PATH,CENTERVILLE,48580
8264	RACHAEL CONLEY	smokewhite20@eyec0de.net	1223 STEVENS CT,CARROLLTON,40084
4224	TRACIE DAVENPORT	swhitaker62@eyec0de.net	549 TALLEY RUN,BRISTOL,63113
1836	NANCY GOFF	holdfire@gma1l.us	147 FLEMING HWY,MORA,37823
6284	KATHY MORENO	hadcode@everyma1l.co.uk	1341 BYRNE DR,SUMNER,80450

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)

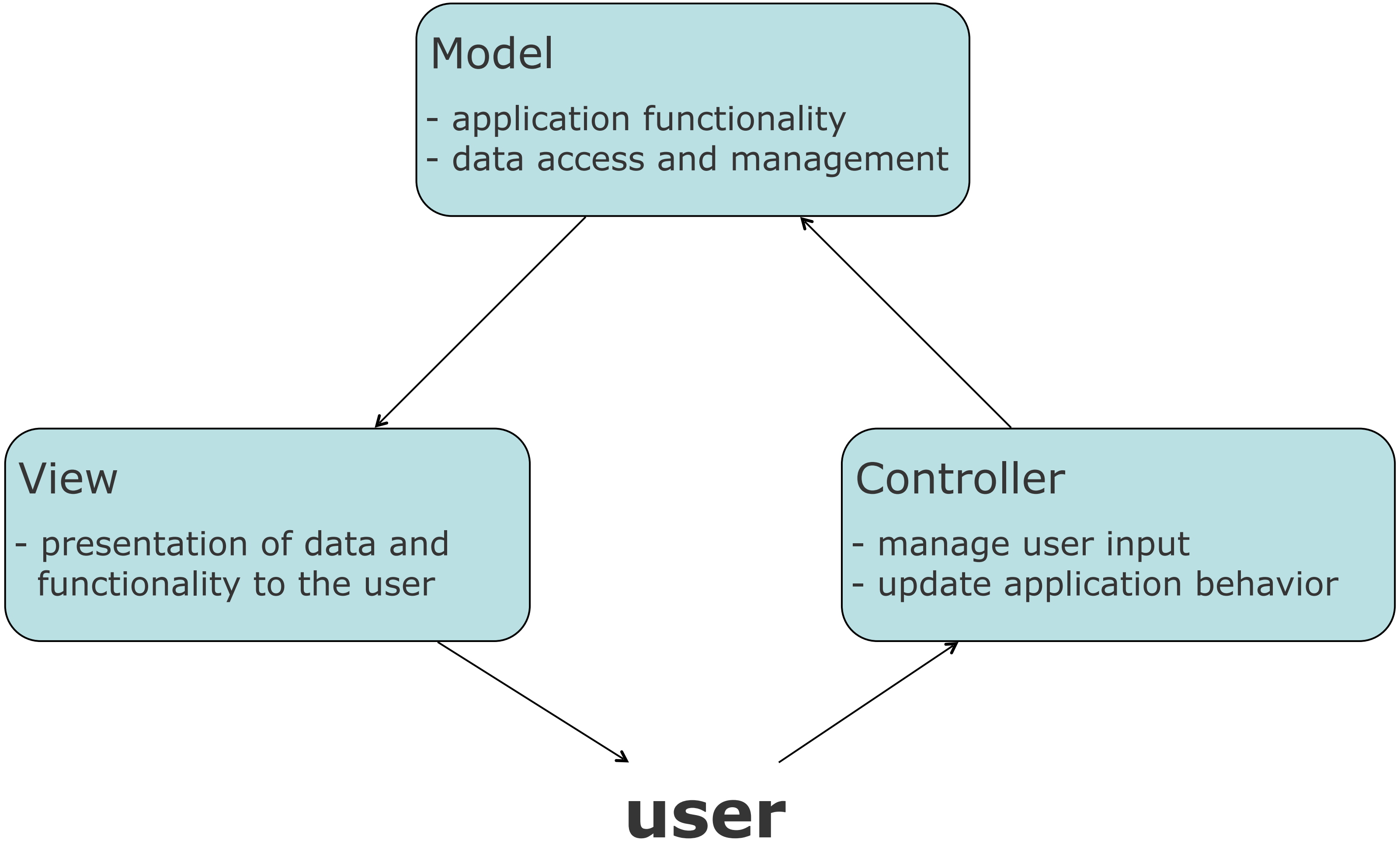
MVC is :

- 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



Model

- application functionality
- data access and management

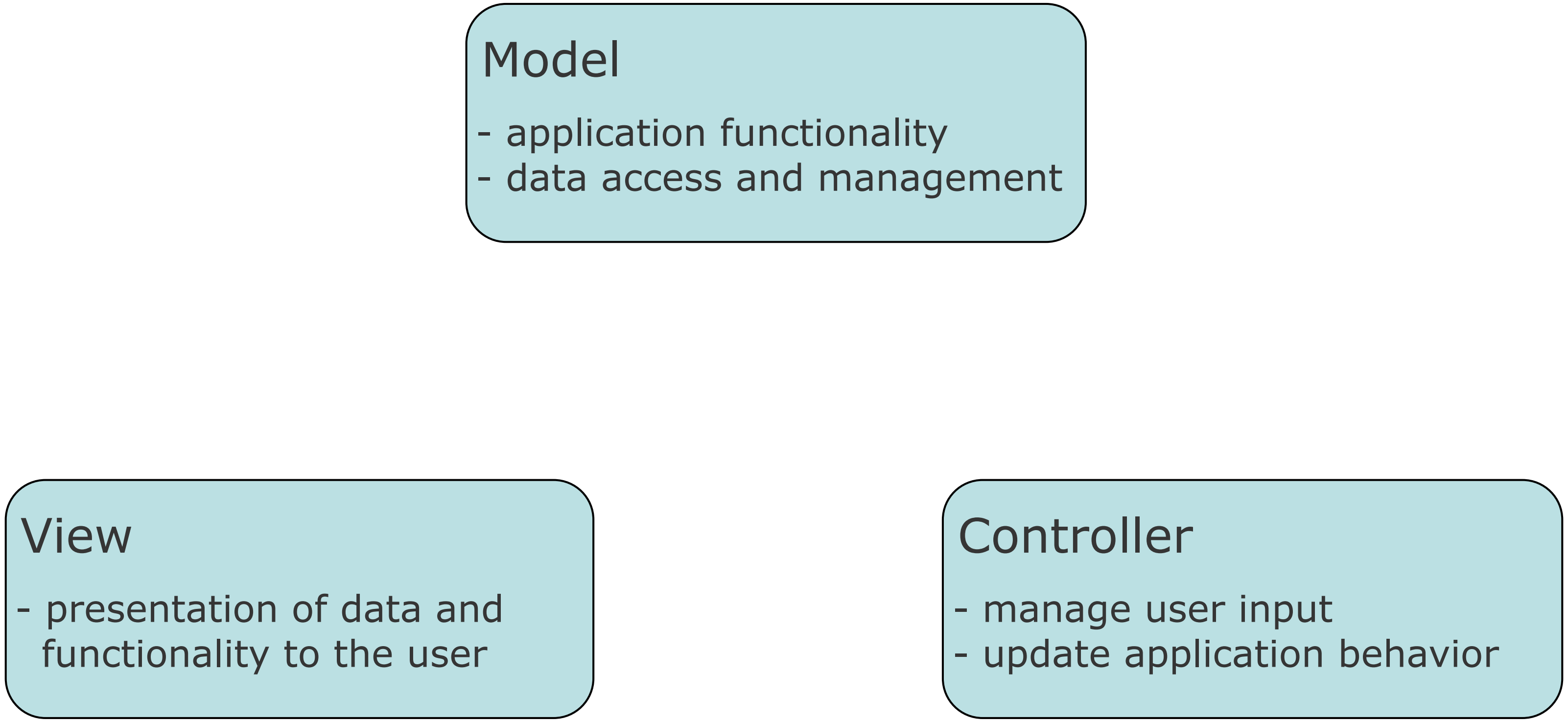
View

- presentation of data and functionality to the user

Controller

- manage user input
- update application behavior

MVC : interactions between components



Model

- application functionality
- data access and management

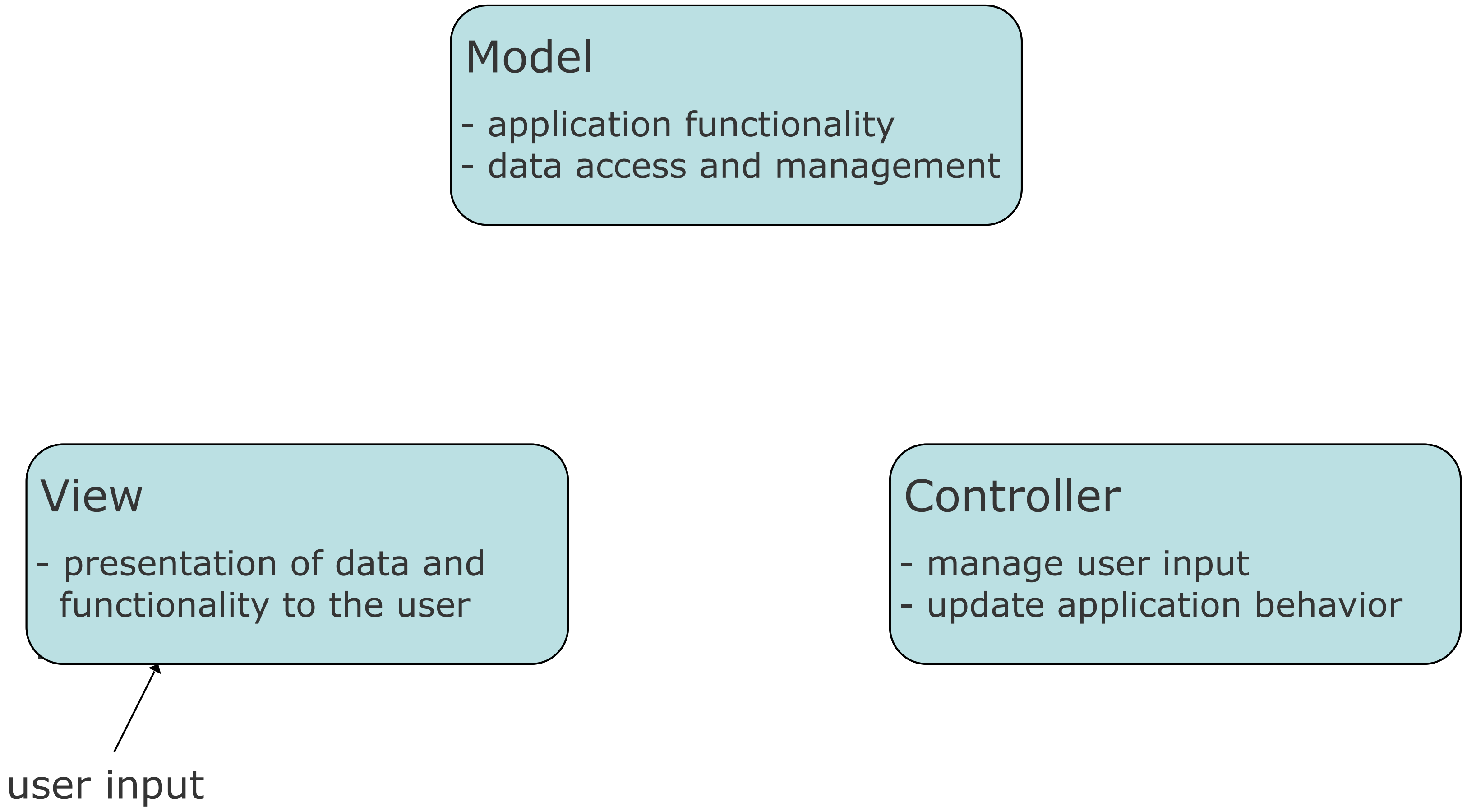
View

- presentation of data and functionality to the user

Controller

- manage user input
- update application behavior

MVC : interactions between components



Model

- application functionality
- data access and management

View

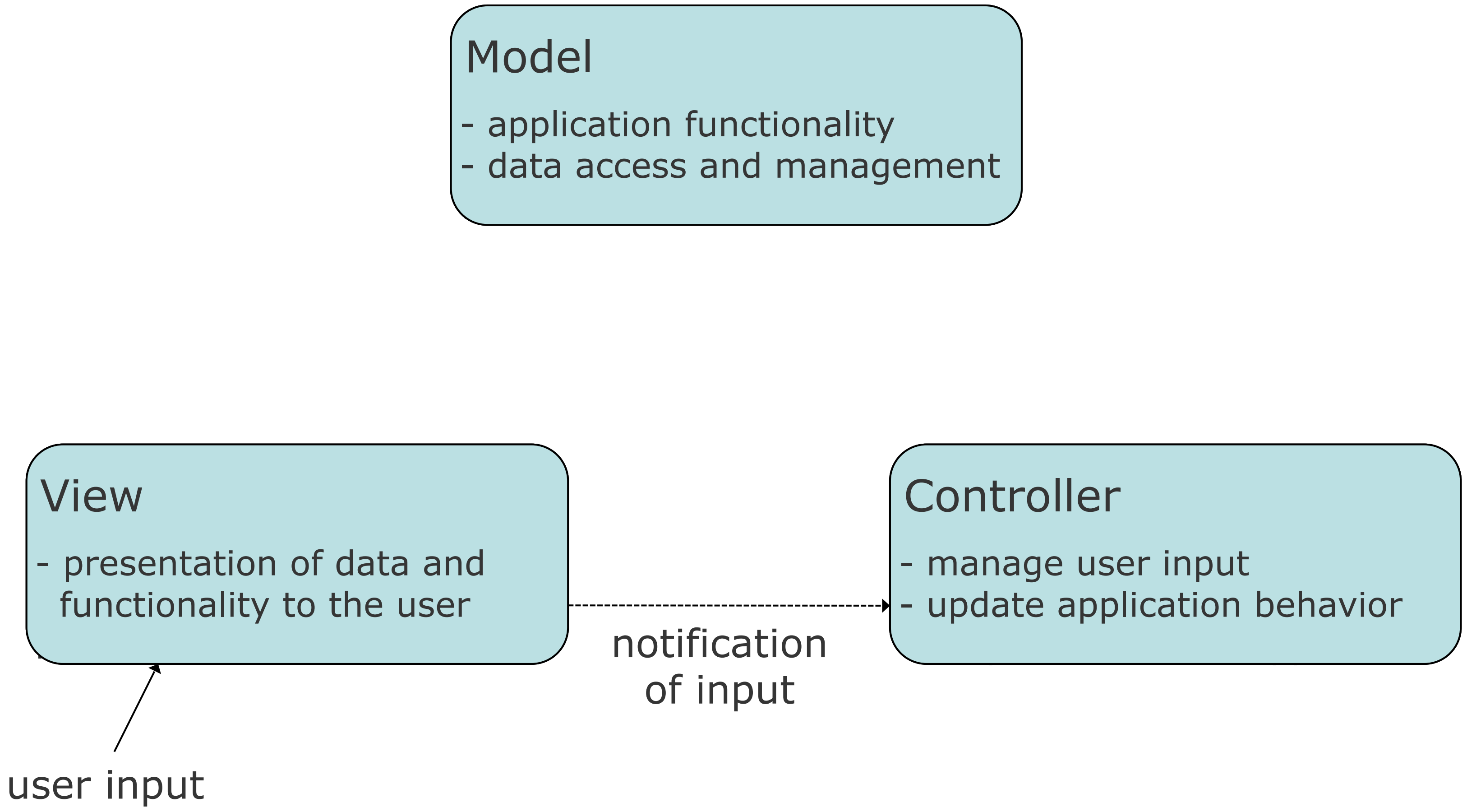
- presentation of data and functionality to the user

Controller

- manage user input
- update application behavior

user input

MVC : interactions between components



Model

- application functionality
- data access and management

View

- presentation of data and functionality to the user

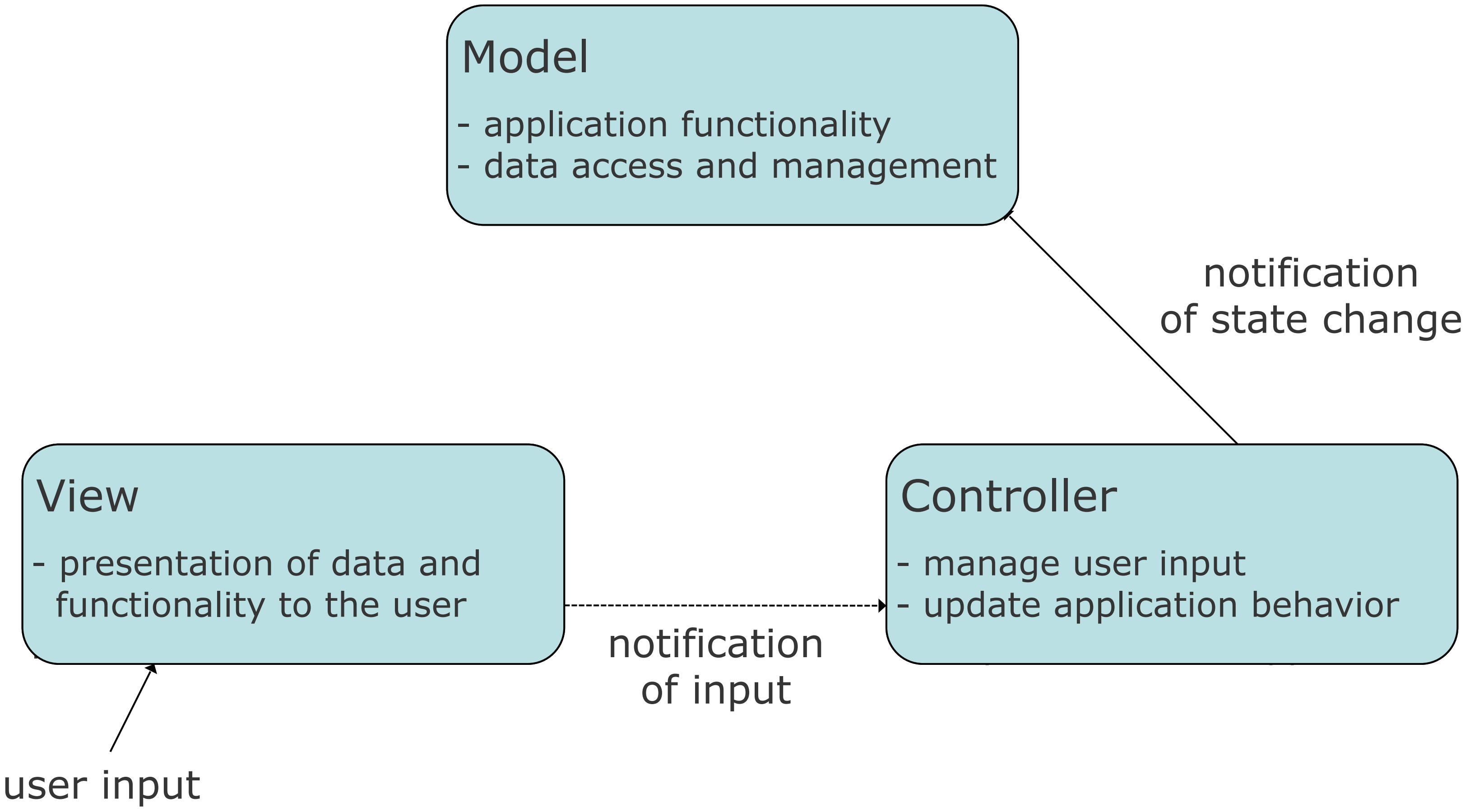
Controller

- manage user input
- update application behavior

notification of input

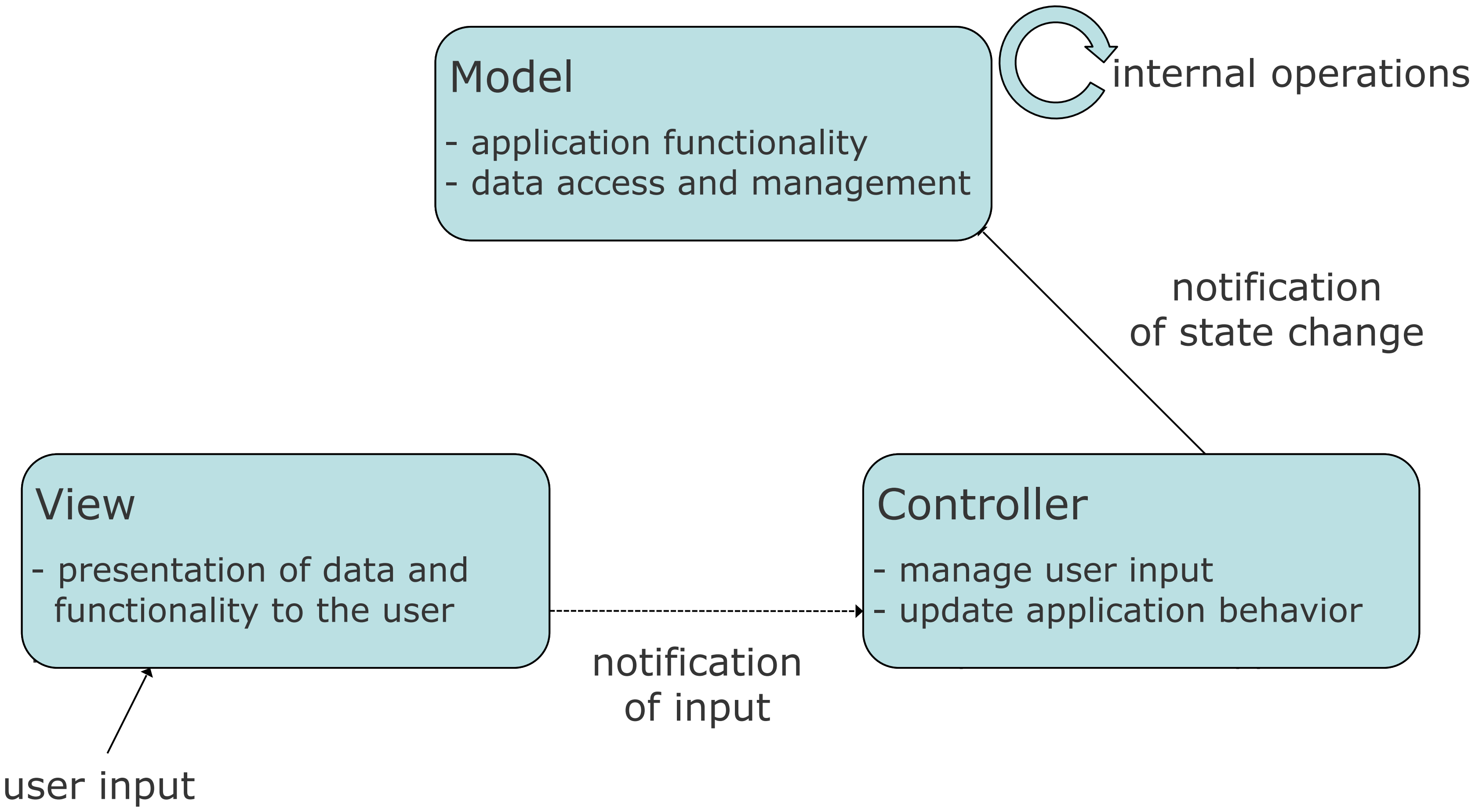
user input

MVC : interactions between components



Model
- application functionality
- data access and management
notification
of state change
View
- presentation of data and
functionality to the user
notification
of input
Controller
- manage user input
- update application behavior
user input

MVC : interactions between components



Model

- application functionality
- data access and management

internal operations

notification of state change

View

- presentation of data and functionality to the user

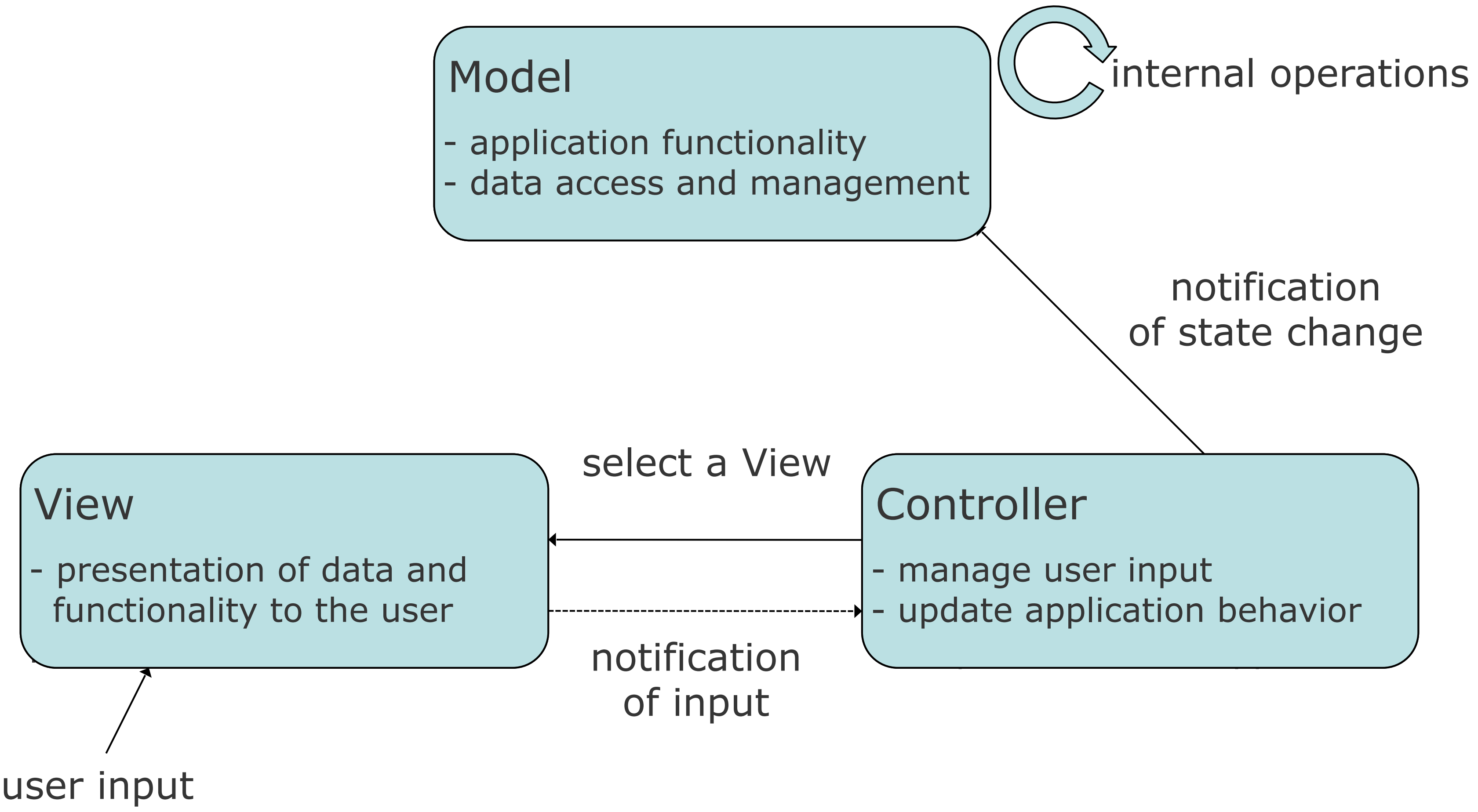
Controller

- manage user input
- update application behavior

notification of input

user input

MVC : interactions between components



Model

- application functionality
- data access and management

internal operations

notification of state change

select a View

View

- presentation of data and functionality to the user

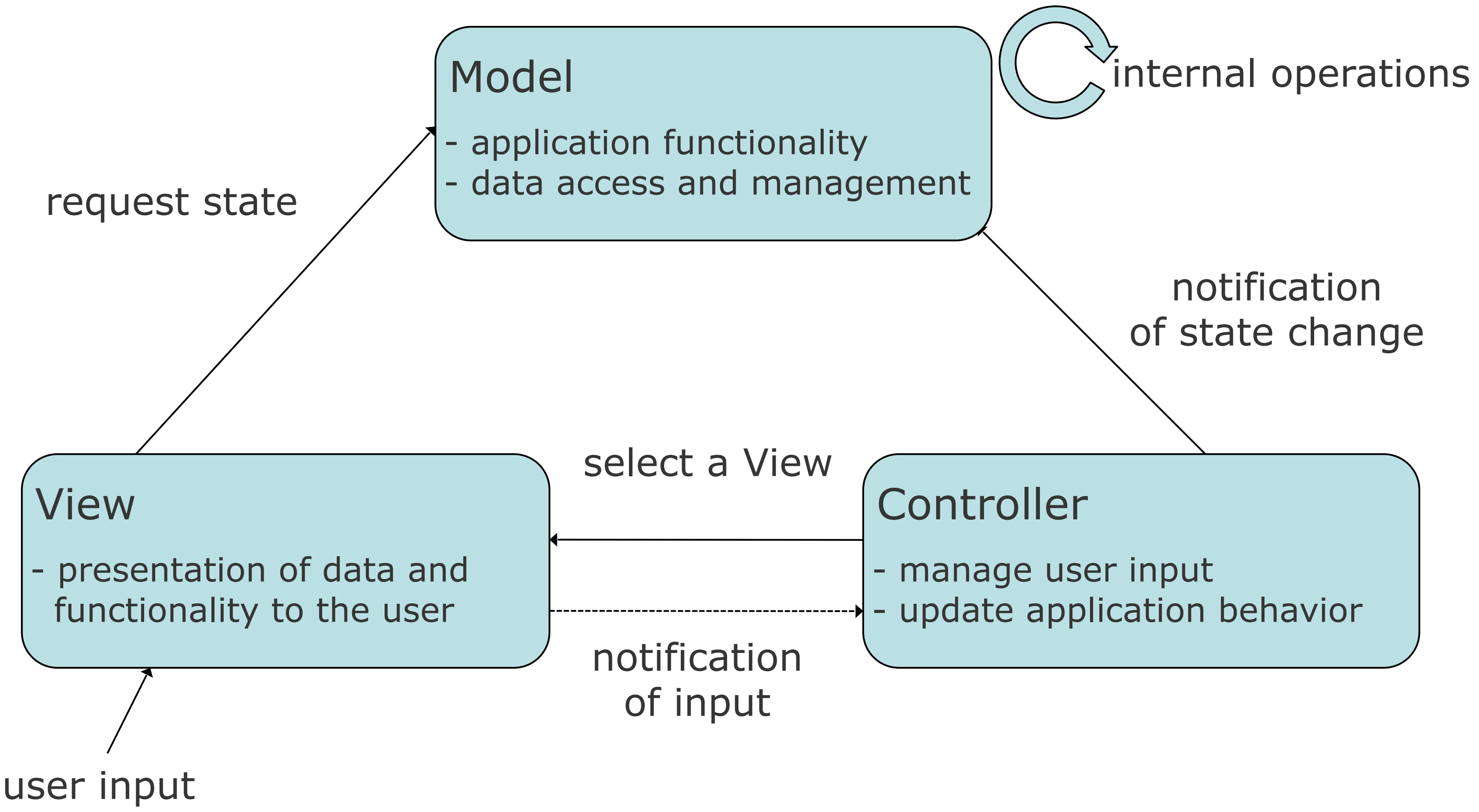
Controller

- manage user input
- update application behavior

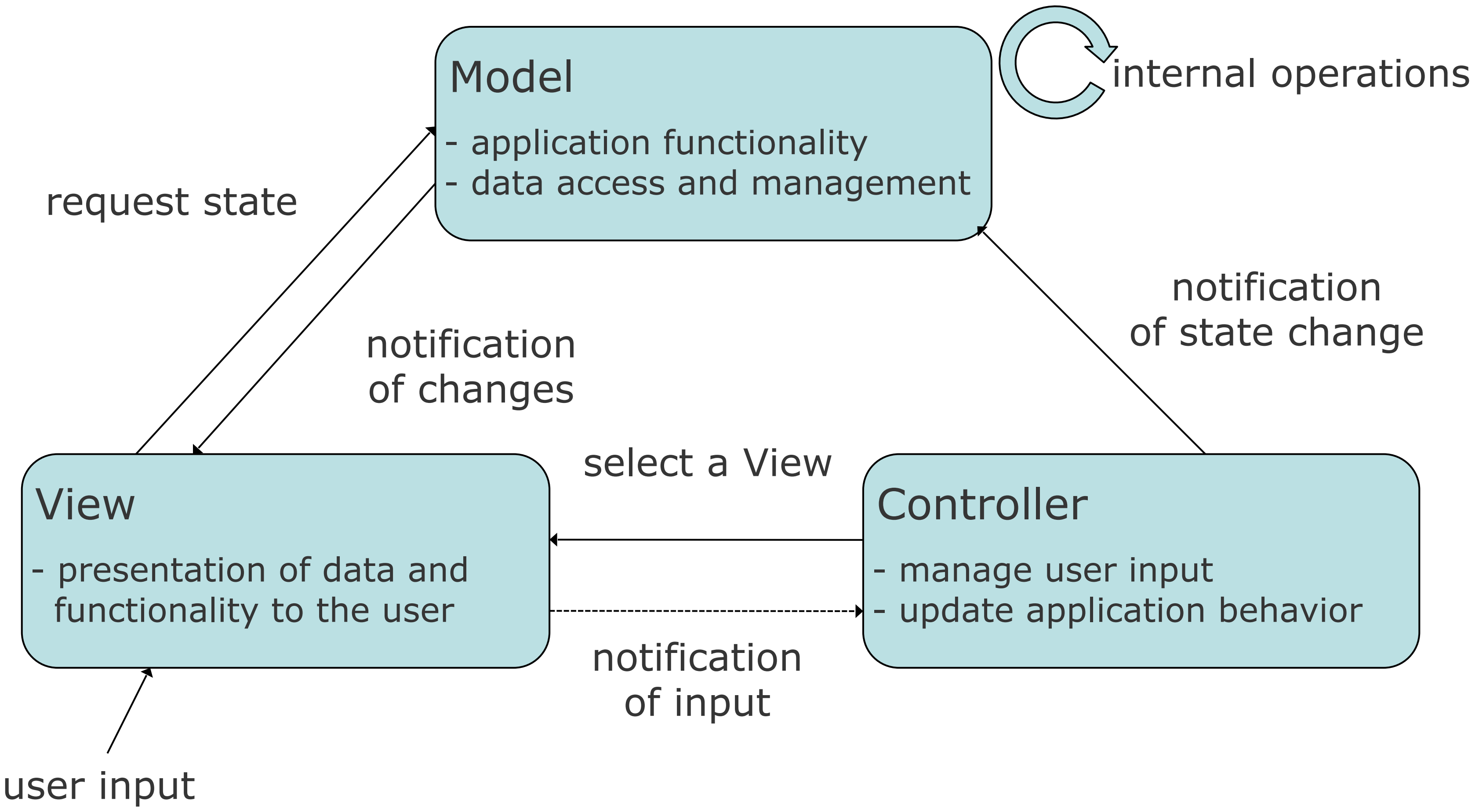
notification of input

user input

MVC : interactions between components



MVC : interactions between components



Model

- application functionality
- data access and management

internal operations

request state

notification of changes

notification of state change

select a View

View

- presentation of data and functionality to the user

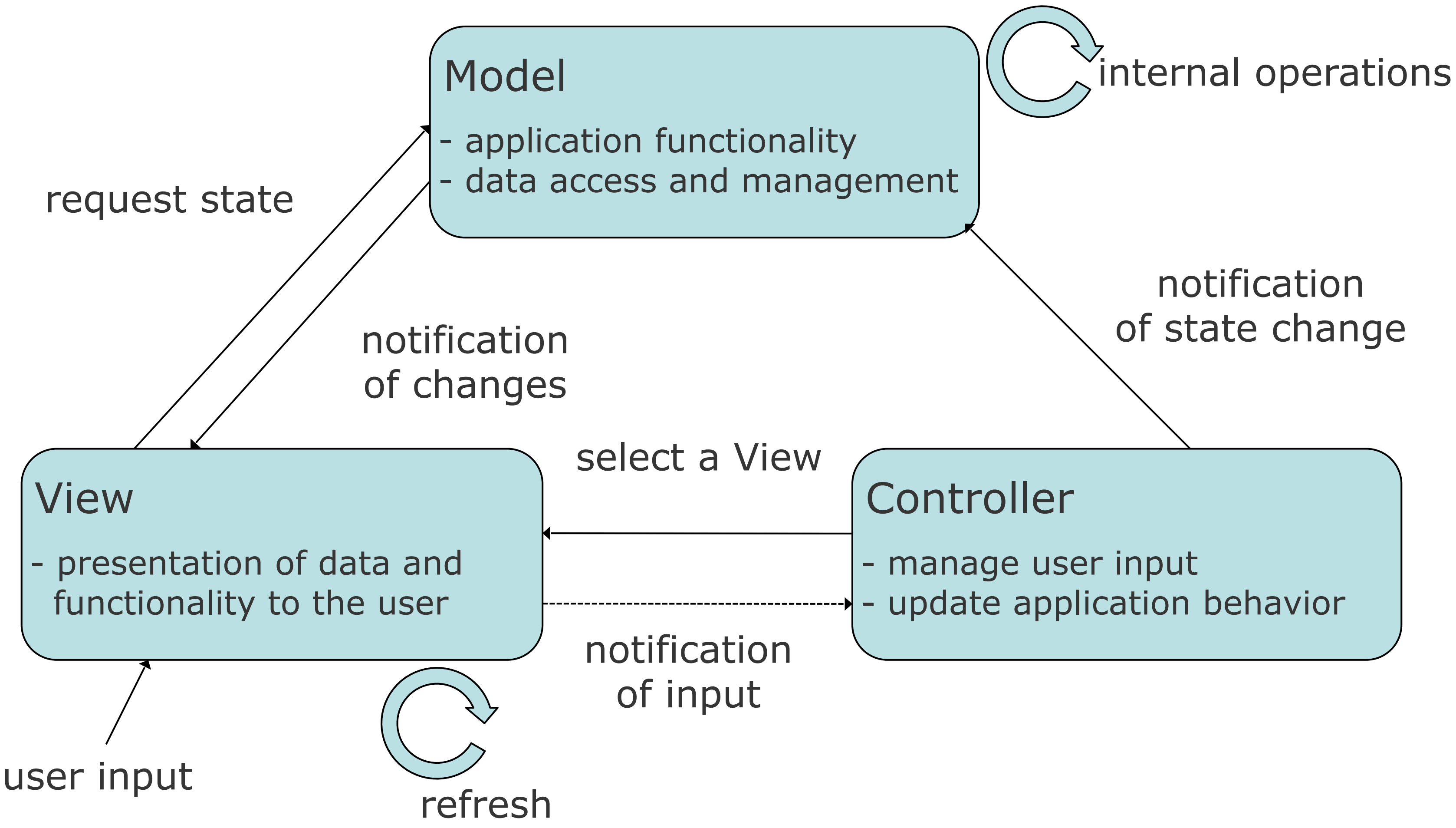
Controller

- manage user input
- update application behavior

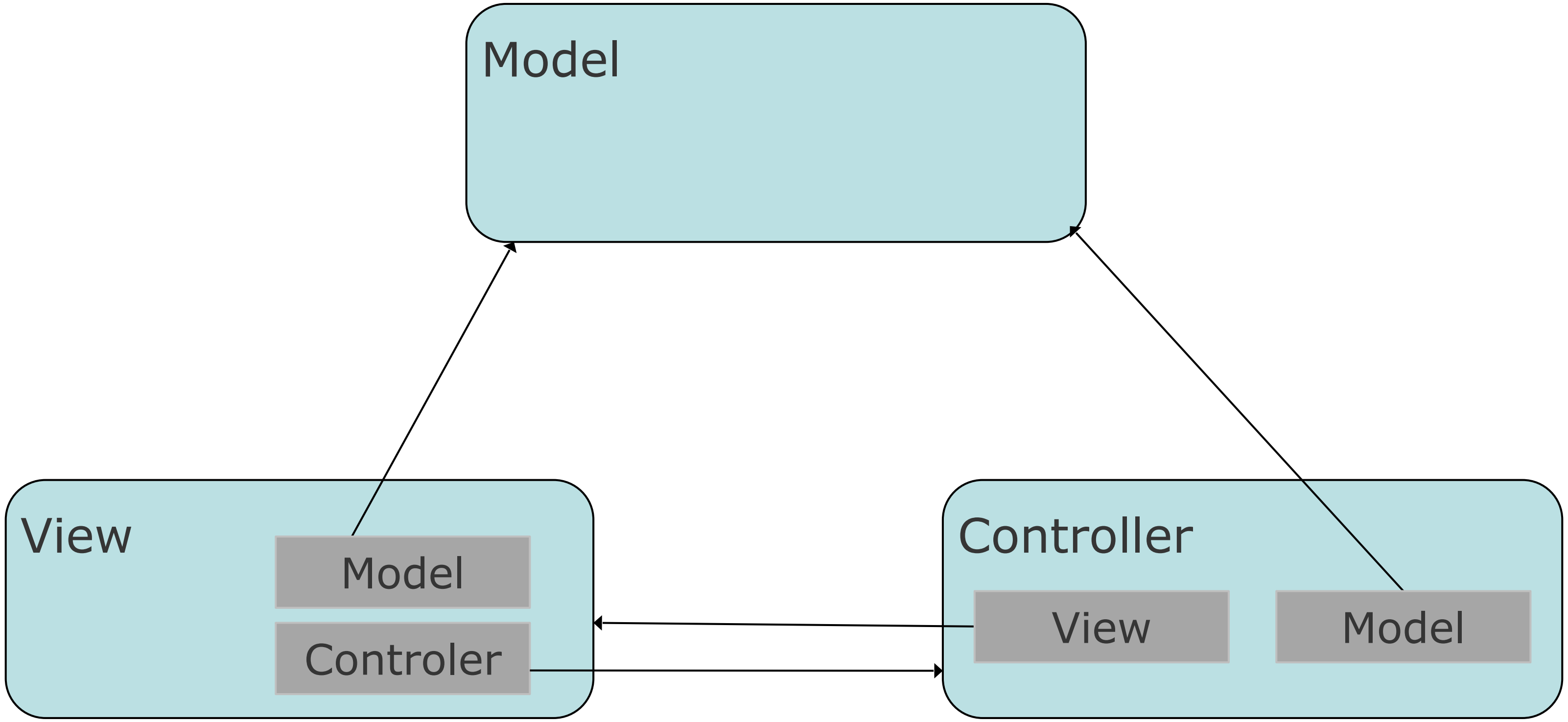
notification of input

user input

MVC : interactions between components


Model
- application functionality
- data access and management
internal operations
request state
notification of changes
notification of state change
select a View
View
- presentation of data and functionality to the user
Controller
- manage user input
- update application behavior
notification of input
user input
refresh

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

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**

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 JavaFX widgets

Model-View-Controller separation not strict

Even for simple widgets

- **Model** : abstract behavior of widget
- **View & Controller** : Look & Feel + Handlers

Examples : Button, Label, Panel, etc.

Most often we do not touch the model of widgets

- JavaFX uses a model by default for each widget

JavaFX : types of models

All nodes have a view and a model (not always visible, but accessible through methods)

Some give us access to both view and model, ex

- ListView, TableView, TreeView
- SelectionModel, FocusModel, MultipleSelectionModel …

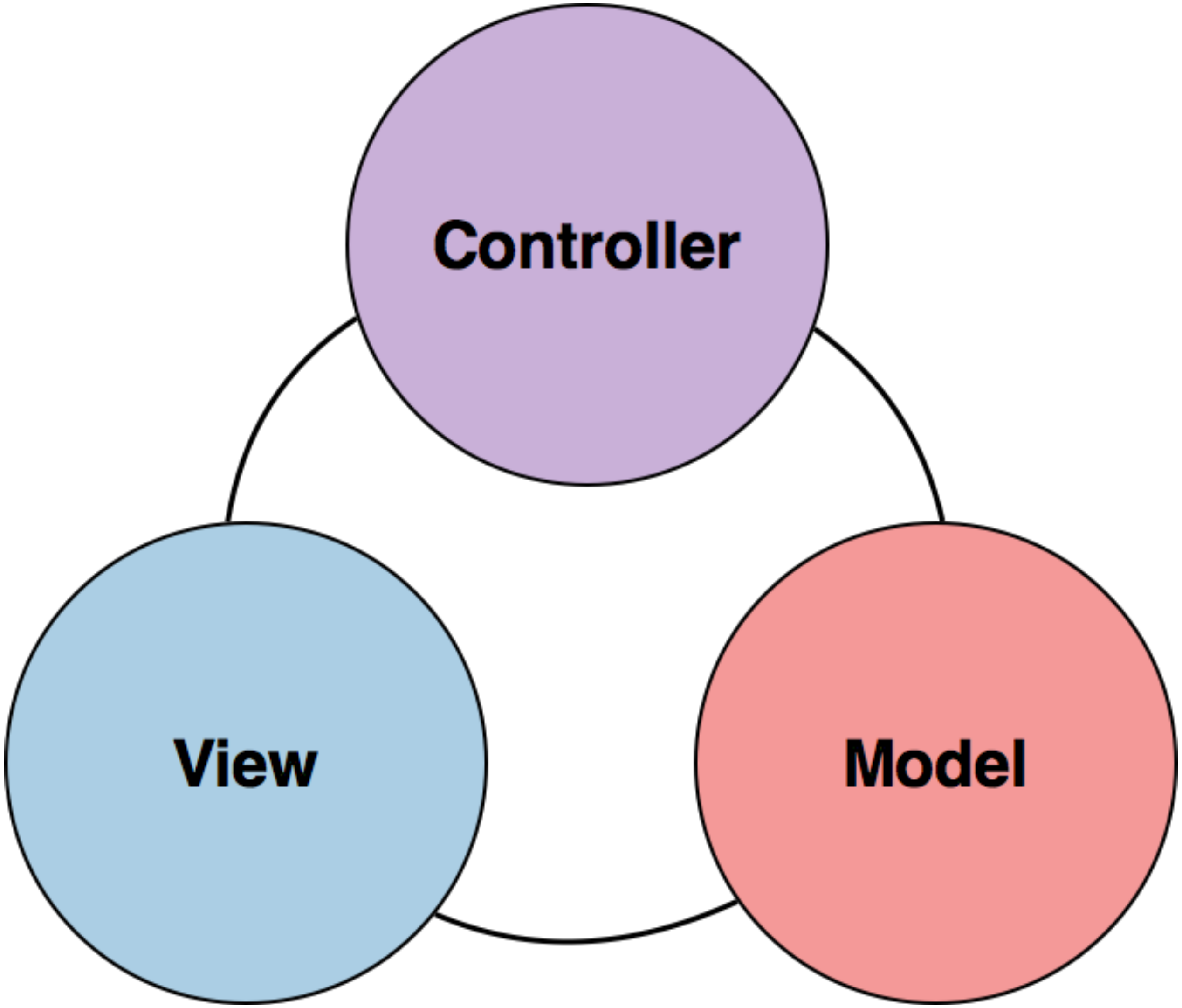
example: revisit our temp calculator (java)

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 FX

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 …