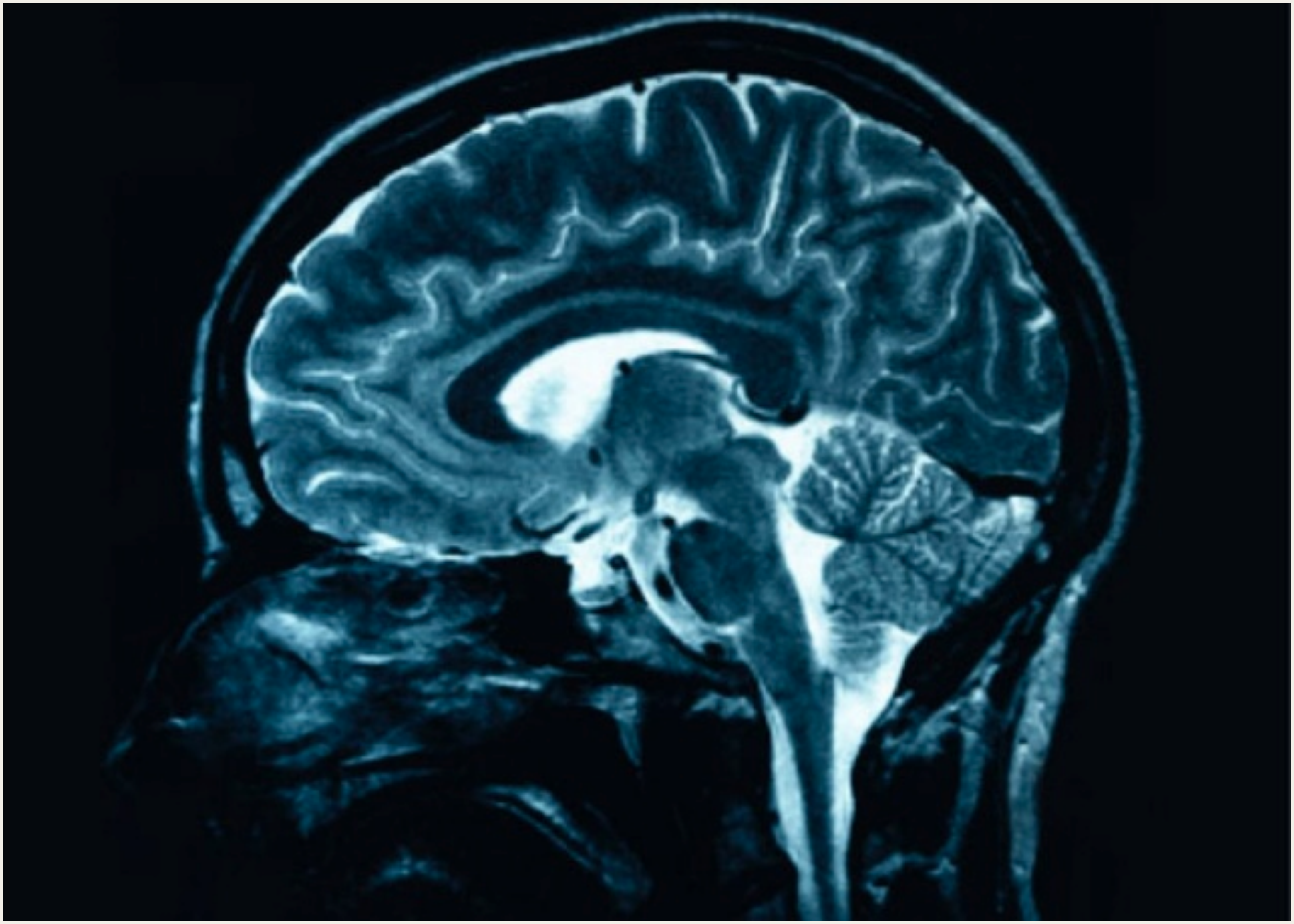
CHANGE

Intro to Iteration #3

CSCI-3081: Program Design and Development

No longer :(

Now

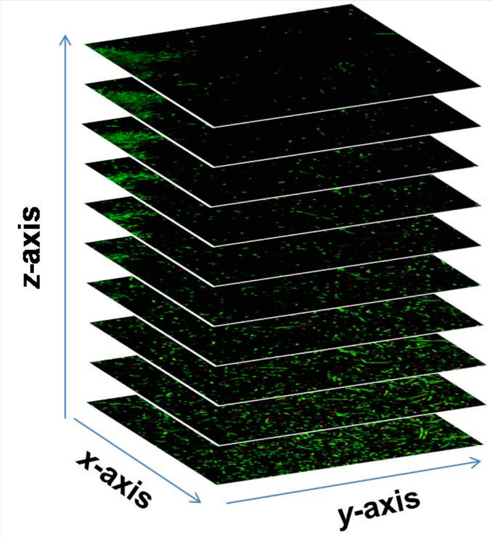


What? I can’t believe it… are these management guys complete idiots?

Why Medical Imaging (Maybe not complete idiots)

- Filters are important (e.g., quantize to segment into tissue types).
- “Painting” is important (e.g., annotate images).
- So, in theory… are code should be easy to apply to medical imaging.

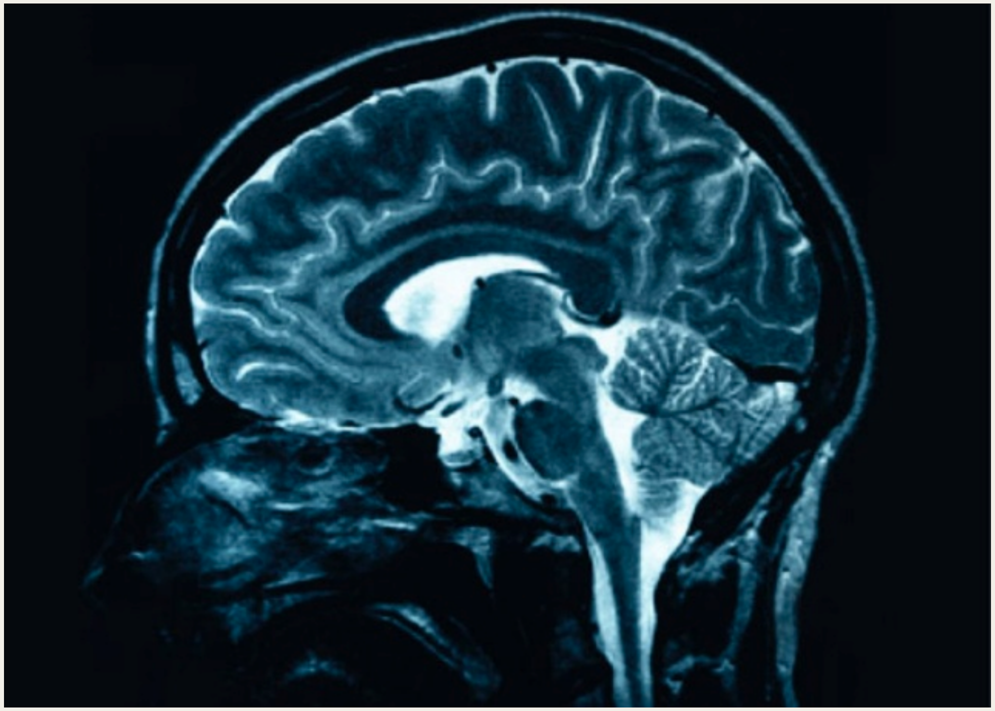
But, Medical Imaging is also different.



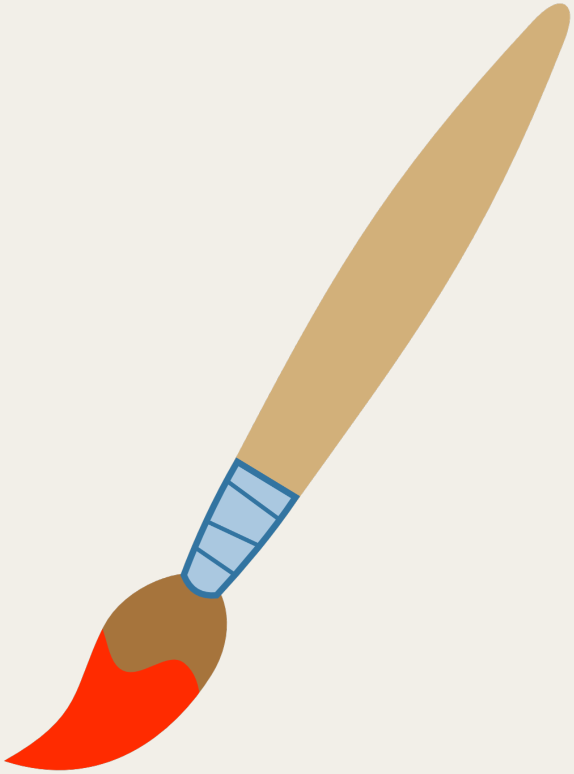
- Doctor’s don’t need / won’t put up with all of the customization provided in our artist-oriented software.
- For 3D imaging, we need to work with volume data, usually stored in stacks of images.
- Doctors/clinics need reproducible results and shortcuts (e.g., scripts) for common tasks.

Good News and Bad News…

1.



2.



This iteration is about 2 things:

1. Refactoring in response to change.

DID I REMEMBER TO TELL YOU BEFORE YOU FINISHED THE CODING THAT THE USER'S SPECIFICATIONS CHANGED?
AAAIII–YIIII–YIIII–YIIII–YIIIIII!!!!!!
SO, NO–ISH?
BAM! BAM! BAM!

Dilbert.com DilbertCartoonist@gmail.com

5-16-11 ©2011 Scott Adams, Inc. Dist. by Universal Uclick.

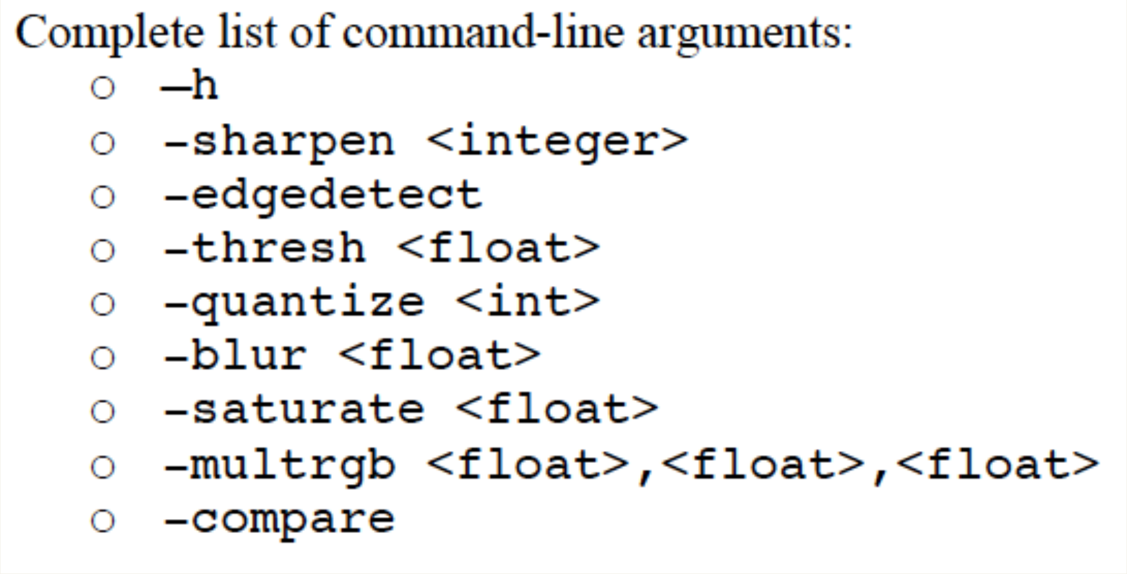
2. Polishing and releasing a project.

Let’s look at some specific requirements.

- Refactor:
  - a library + 2 applications (FlashPhoto and Mia), all in separate directories
  - no duplicate code
  - “make all” in the root directory builds the whole project
- Create the Mia application
  - A red arrow stamp and a red pen tool for annotating medical images
  - A subset of filters useful for medical imaging
  - File open and save now handle 3D “image stacks”
  - A command line mode that supports scripting.

Let's look at some specific requirements.

- Mia command line options



Complete list of command-line arguments:

- –h
- -sharpen <integer>
- -edgedetect
- -thresh <float>
- -quantize <int>
- -blur <float>
- -saturate <float>
- -multrgb <float>,<float>,<float>
- -compare

Let’s look at some specific requirements.

- Testing:
    - The command line options are also really useful for testing!
    - Can you figure out how?
    - What other forms of testing are needed?

Let’s look at some specific requirements.

- Writing: You tell me… what kind of writing is needed to “release” this project?

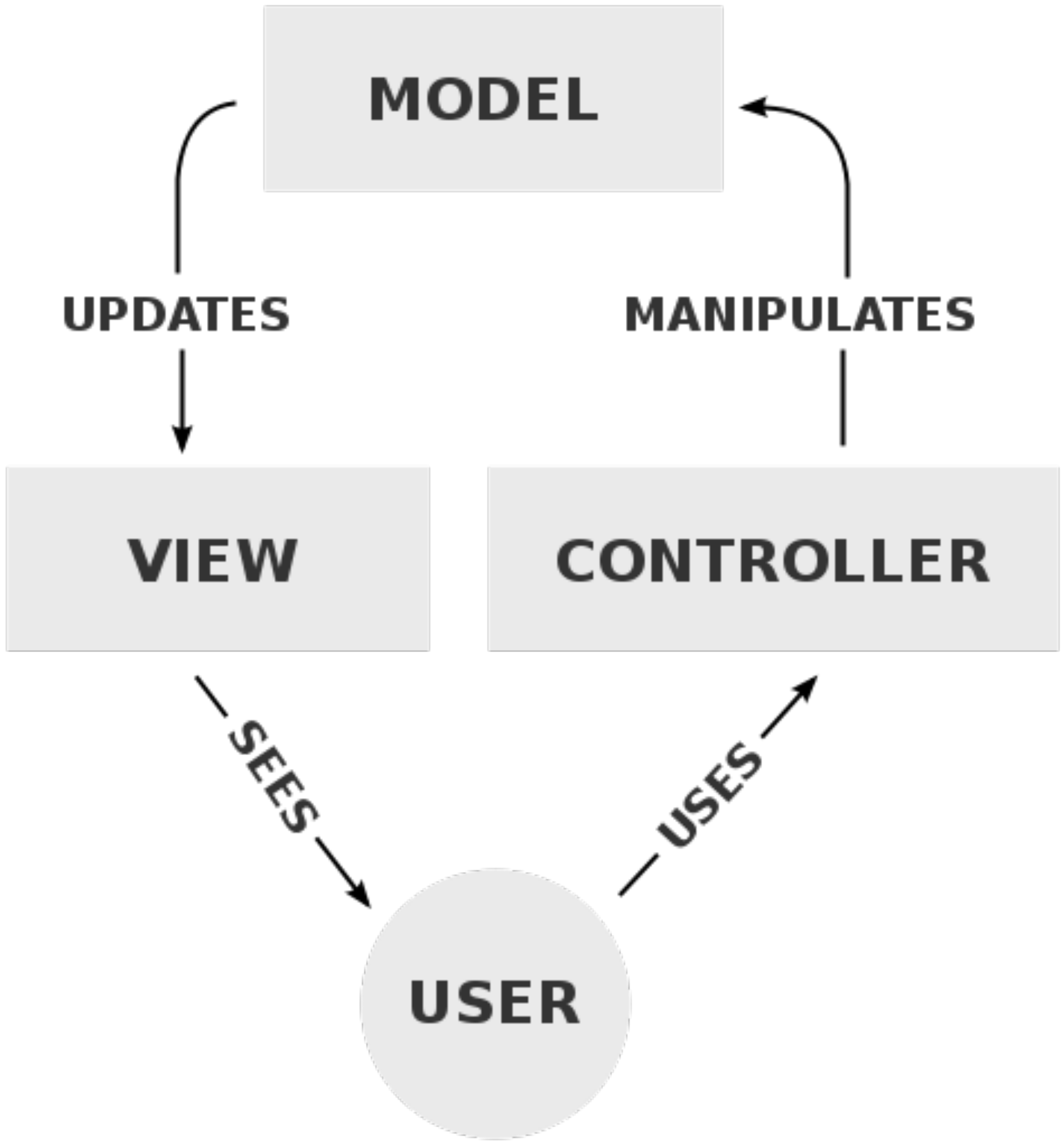
For the rest of class today, we'll discuss:

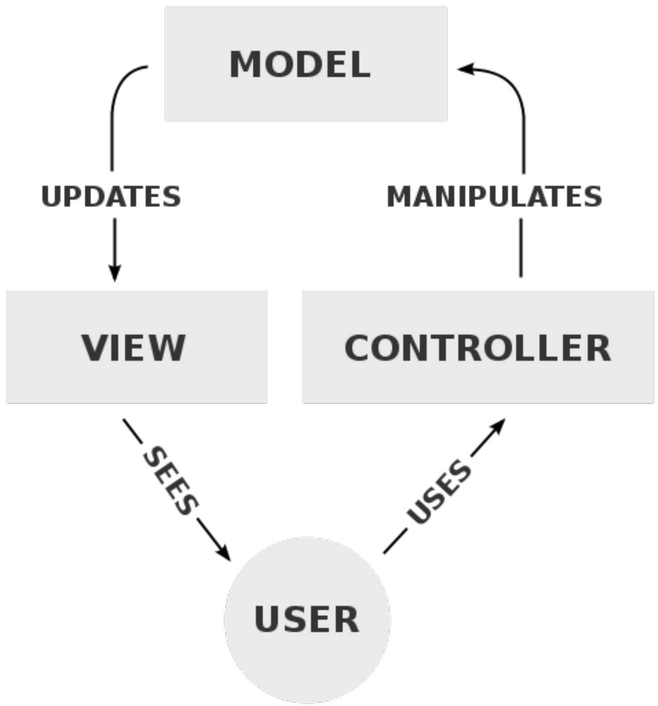
- How would you refactor / reorganize your code to support two applications?

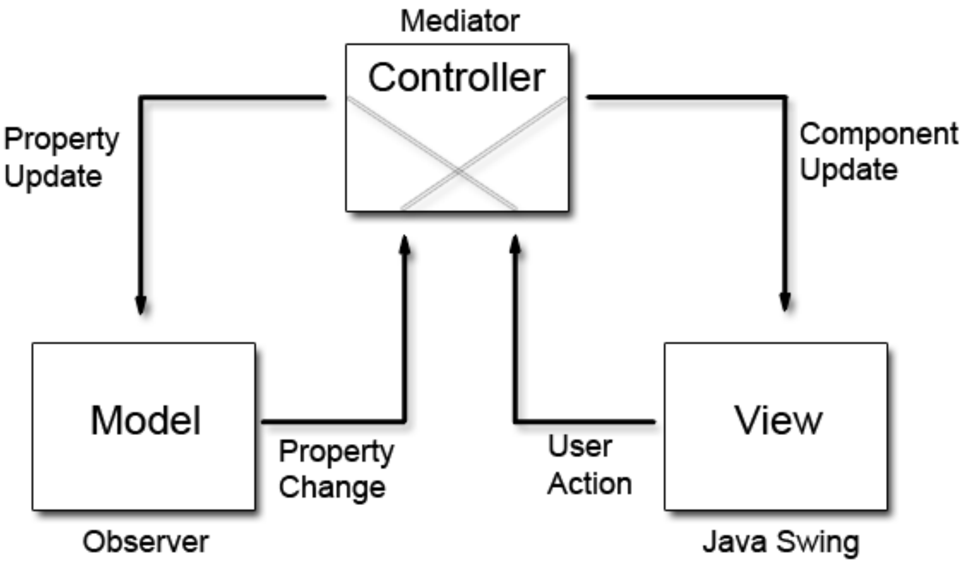
CSCI-3081: Program Design and Development

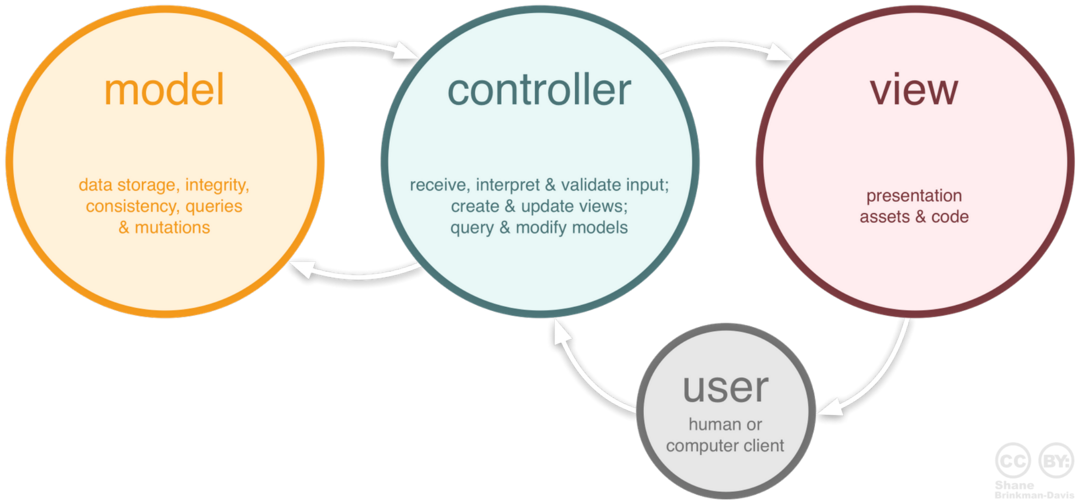
THE MODEL VIEW CONTROLLER PARADIGM

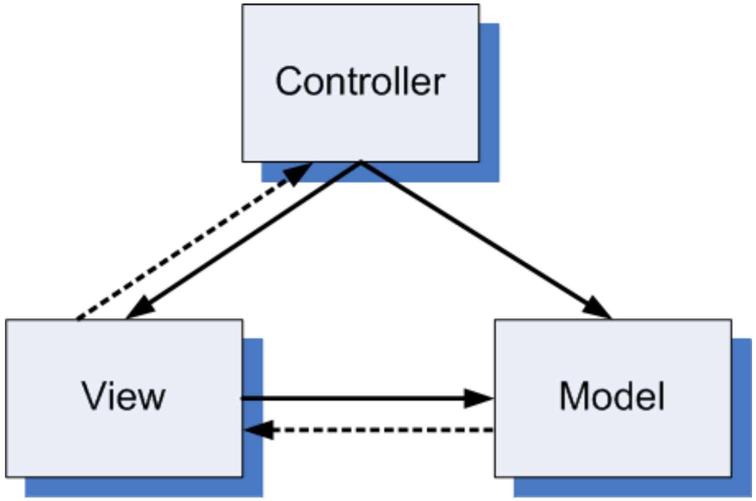
* based on slides of Jeffrey Heer, Jake Wobbrock, James Landay, and Jeffrey Nichols


MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER


MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER

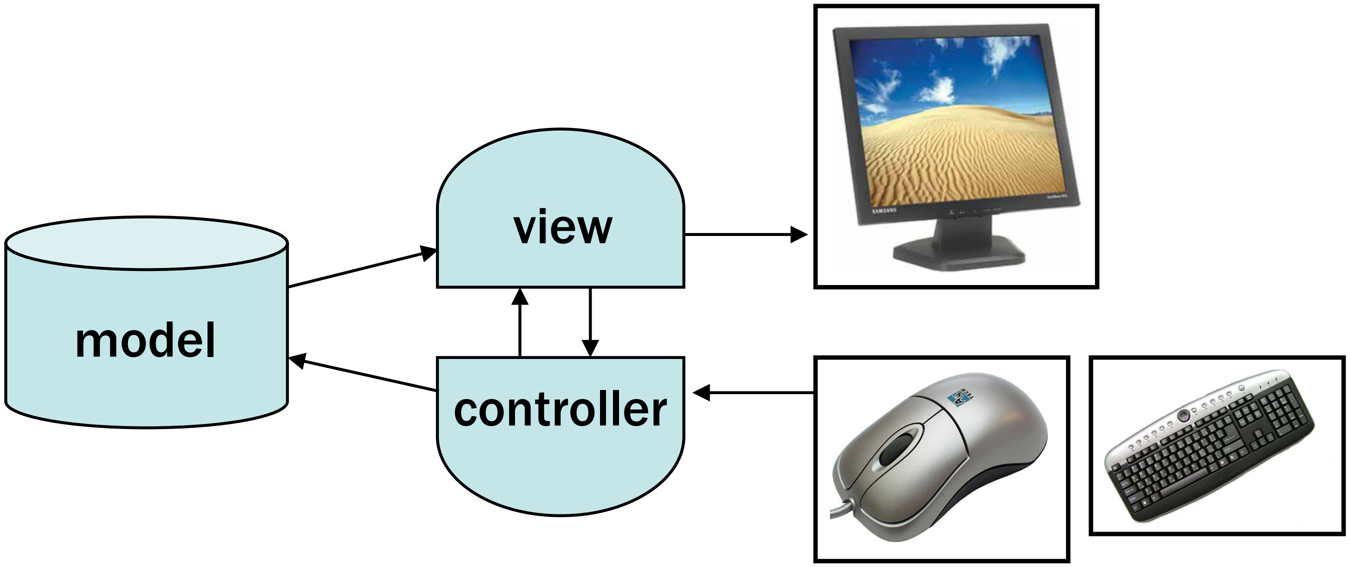

Mediator
Controller
Property Update
Component Update
Model
Property Change
User Action
View
Observer
Java Swing


model
data storage, integrity, consistency, queries & mutations
controller
receive, interpret & validate input; create & update views; query & modify models
view
presentation assets & code
user
human or computer client
Shane Brinkman-Davis


Controller
View
Model

Model--View--Controller (MVC)

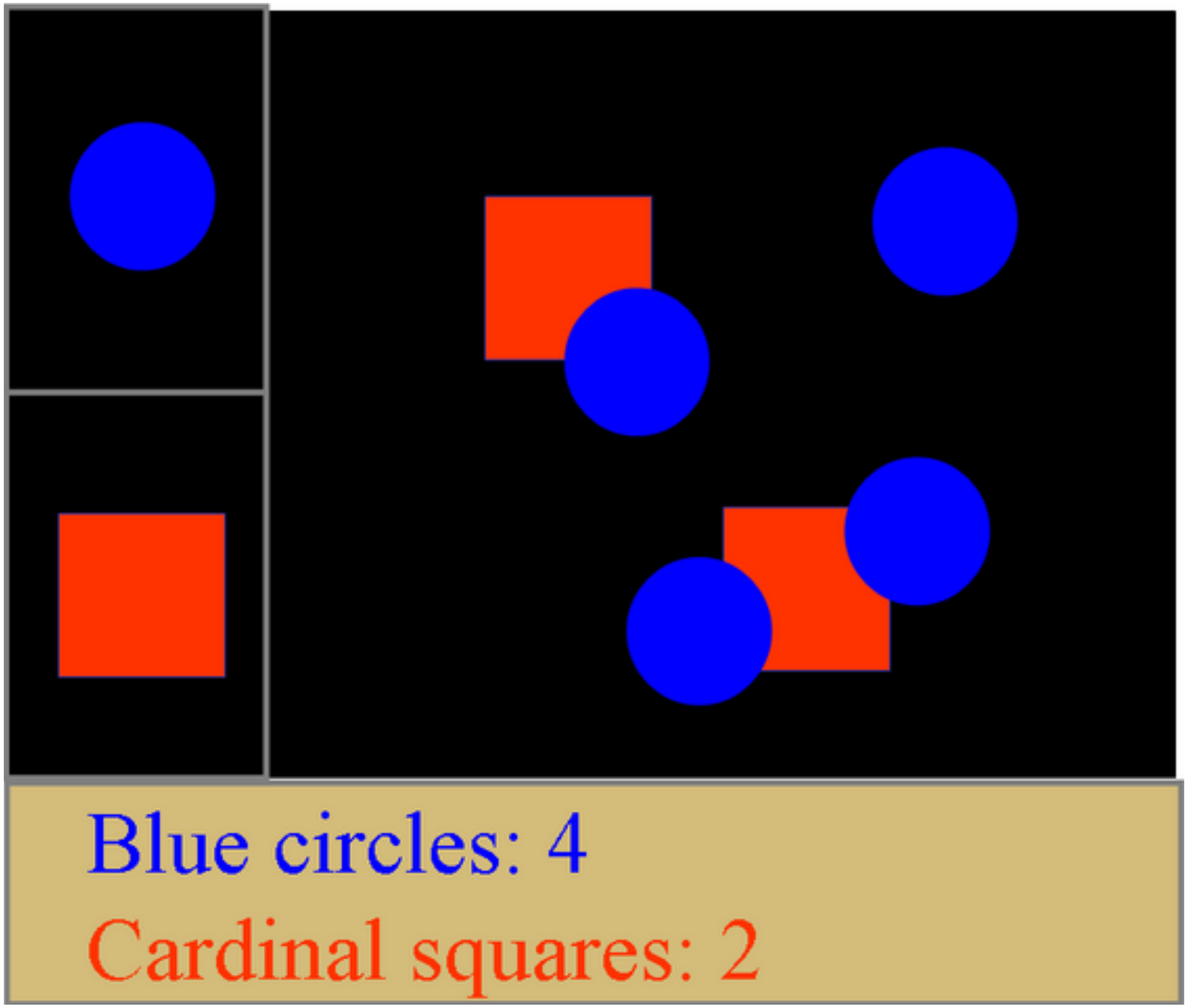
- An architecture for interactive applications
  - Introduced by Smalltalk developers at PARC
- Partitions the application in a way that is
  - Scalable
  - Maintainable



Example Application: A Drawing Program

Interactive Menu

2D View of the Drawing Canvas



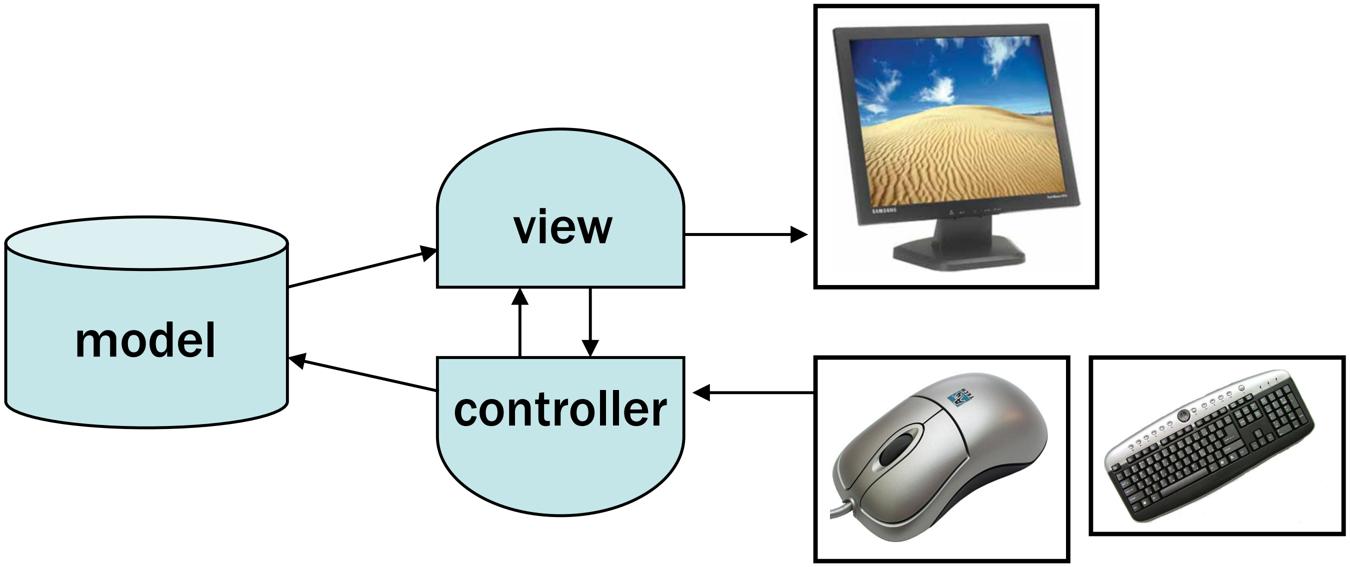
Blue circles: 4

Cardinal squares: 2

Text View of Properties of the Drawing

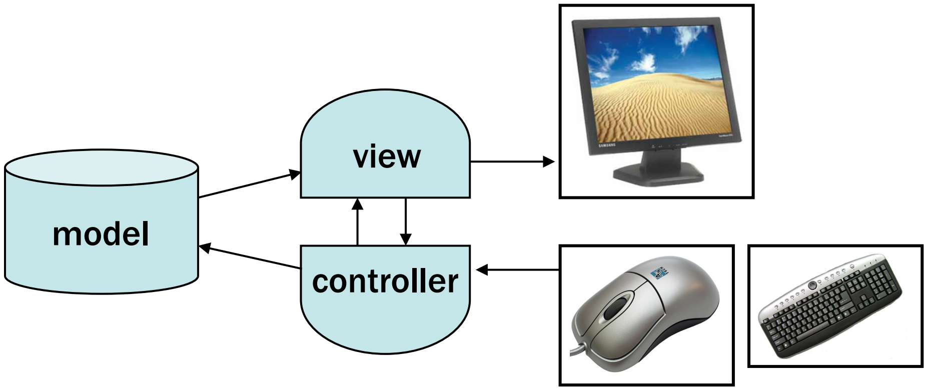
The Model

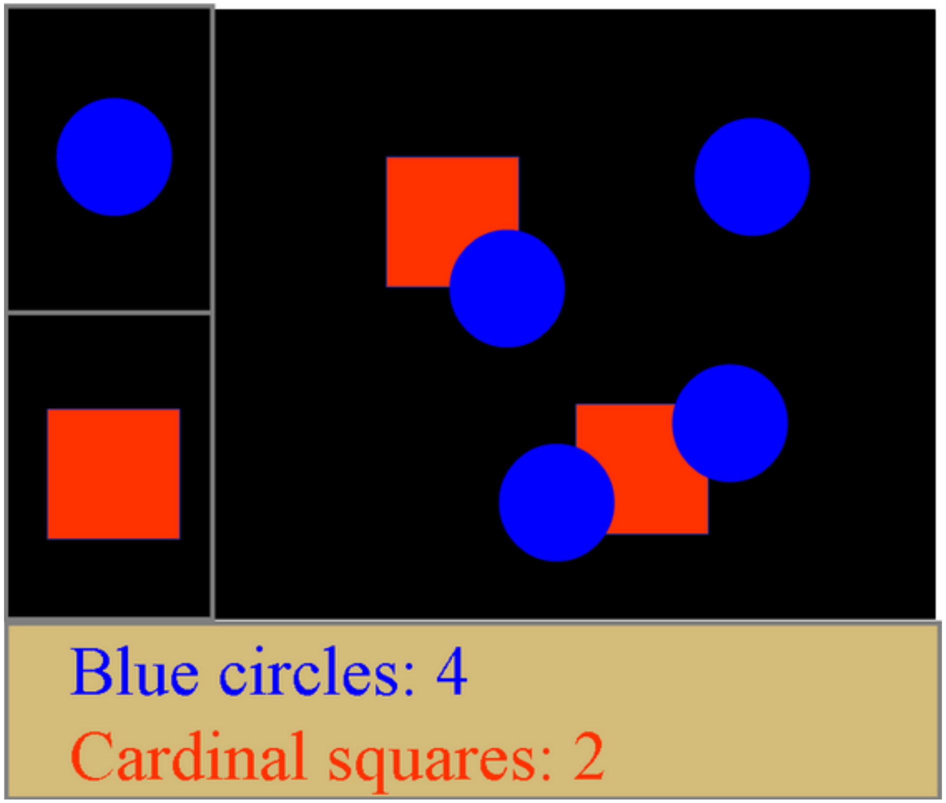
- Information the application is trying to manipulate
- Representation of real world objects
  - Circuit for a CAD program
  - Shapes in a drawing program
  - List of people in a contact management program
  - Teams and events in a basketball game



The View

- Implements a visual display of the model
- May have multiple views
  - E.g. shape view and numeric view
- Any time the model is changed, each view must be notified so that it can update

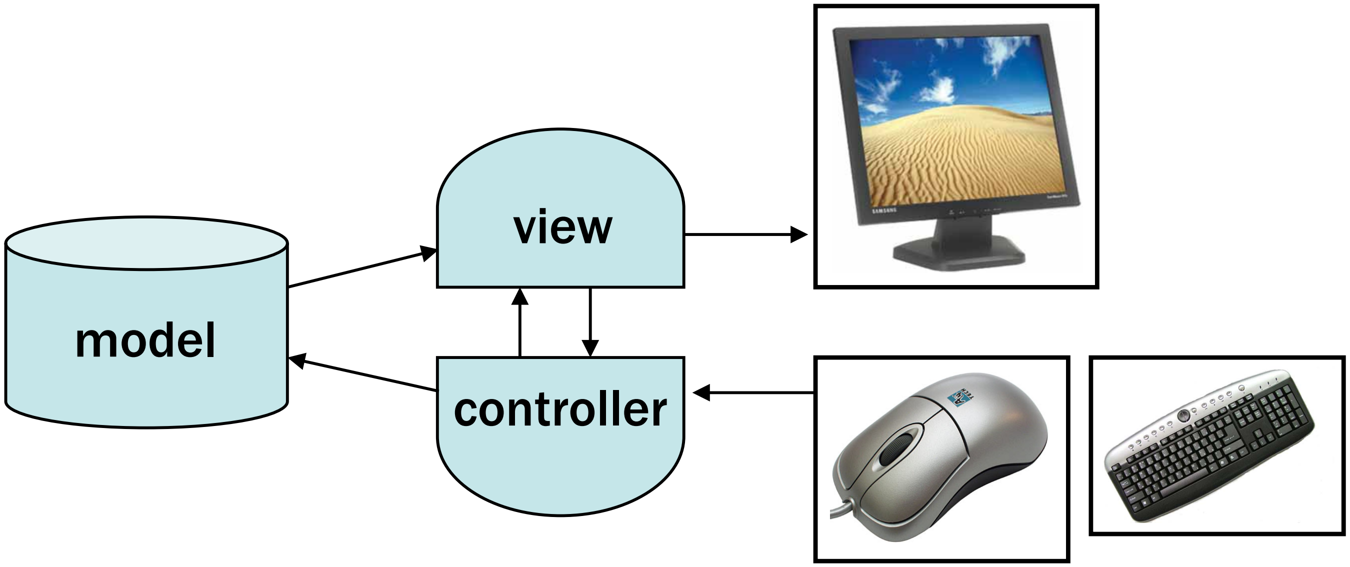


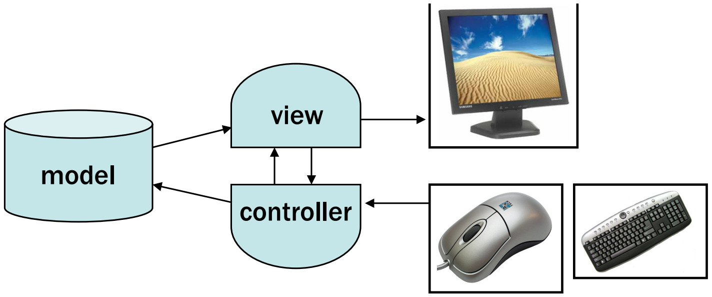

Blue circles: 4

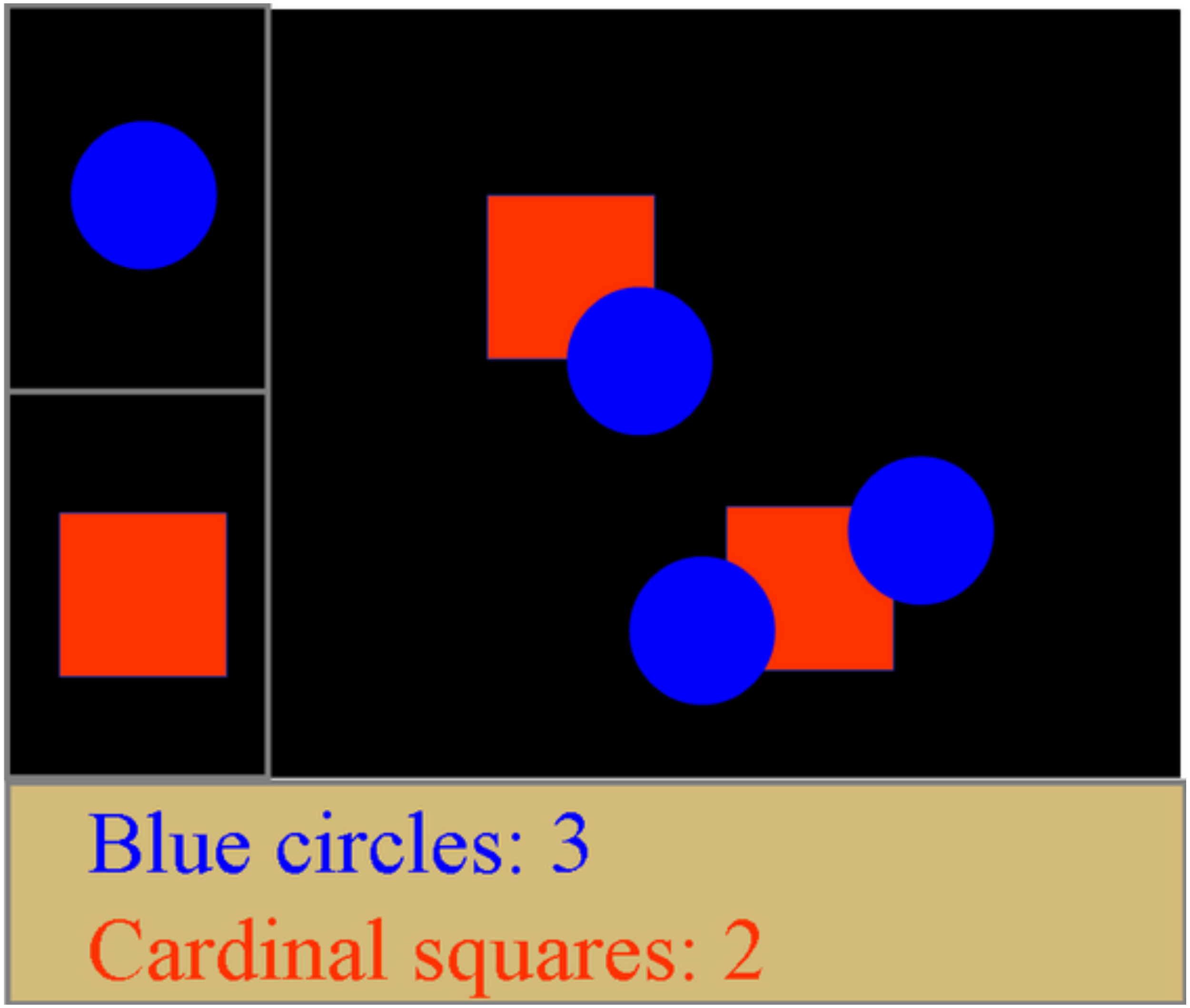
Cardinal squares: 2

The Controller

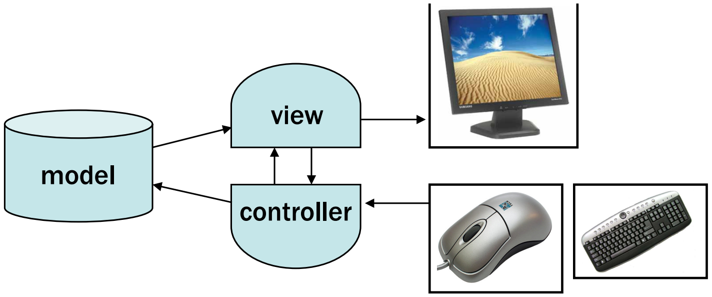
- Receives all input events from the user
- Decides what they mean and what to do
  - Communicates with the view to determine the objects being manipulated (e.g. selection)
  - Calls model methods to make changes to objects

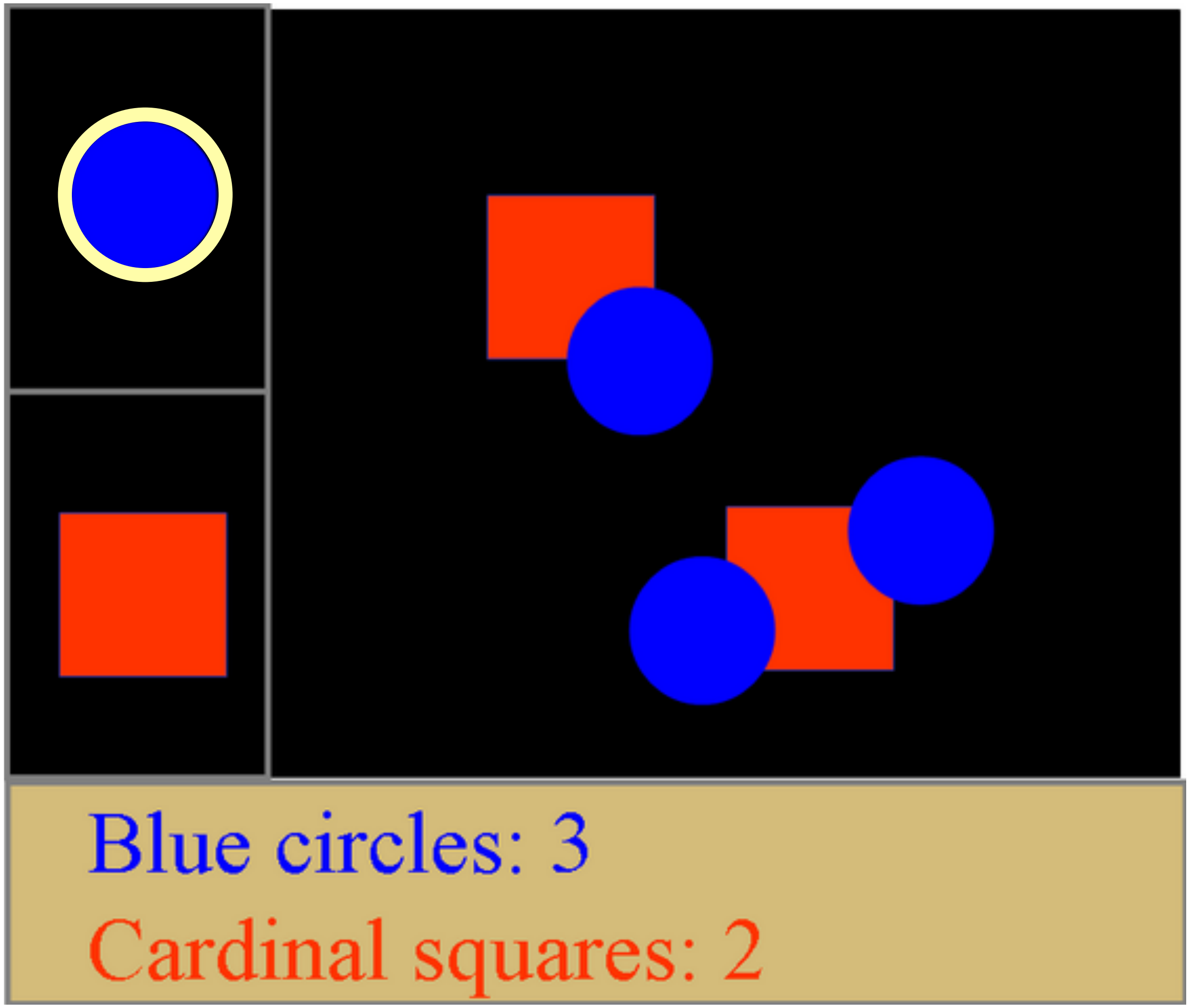




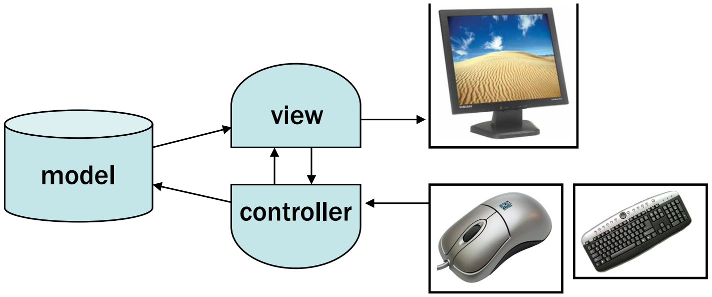

Blue circles: 3

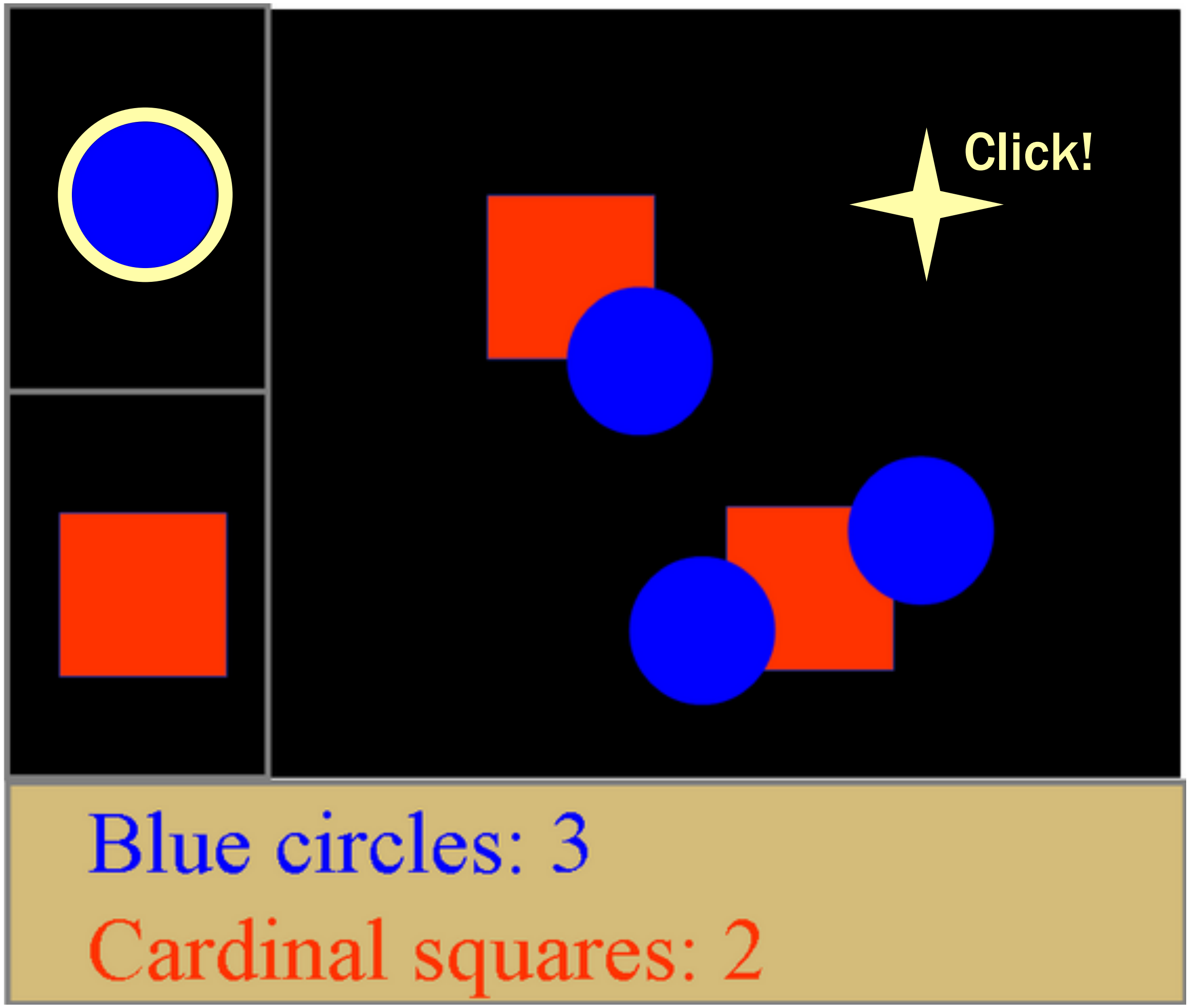
Cardinal squares: 2




Blue circles: 3

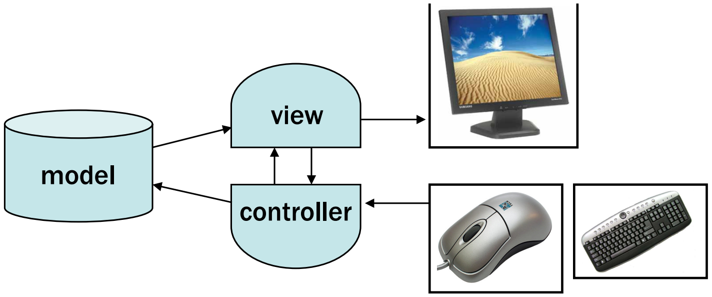
Cardinal squares: 2

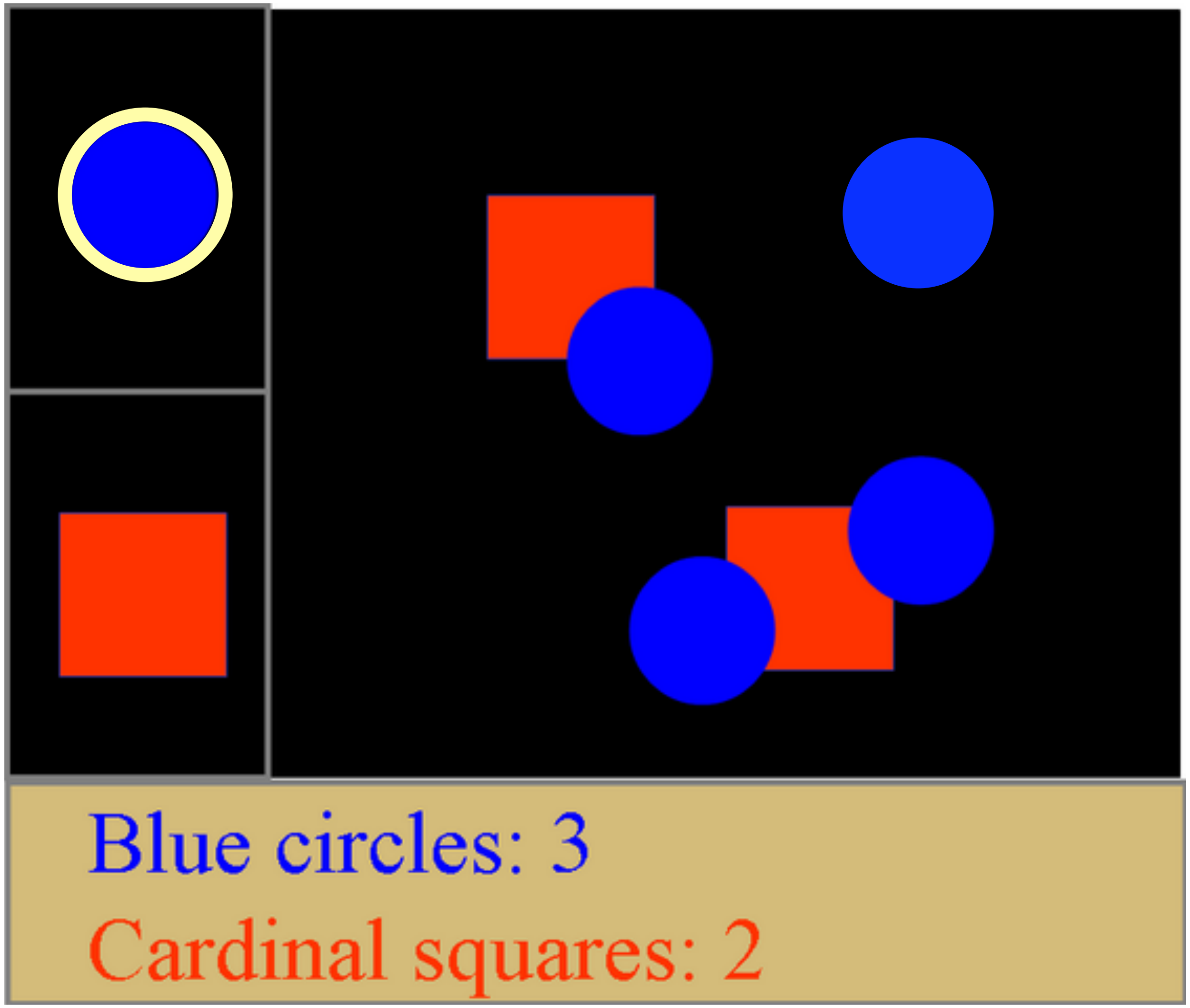




Blue circles: 3

Cardinal squares: 2

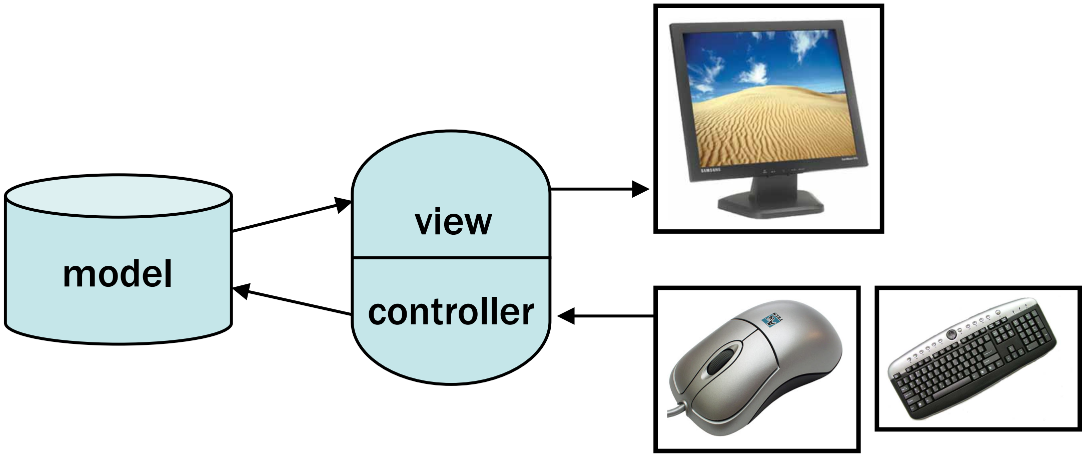



Blue circles: 3

Cardinal squares: 2

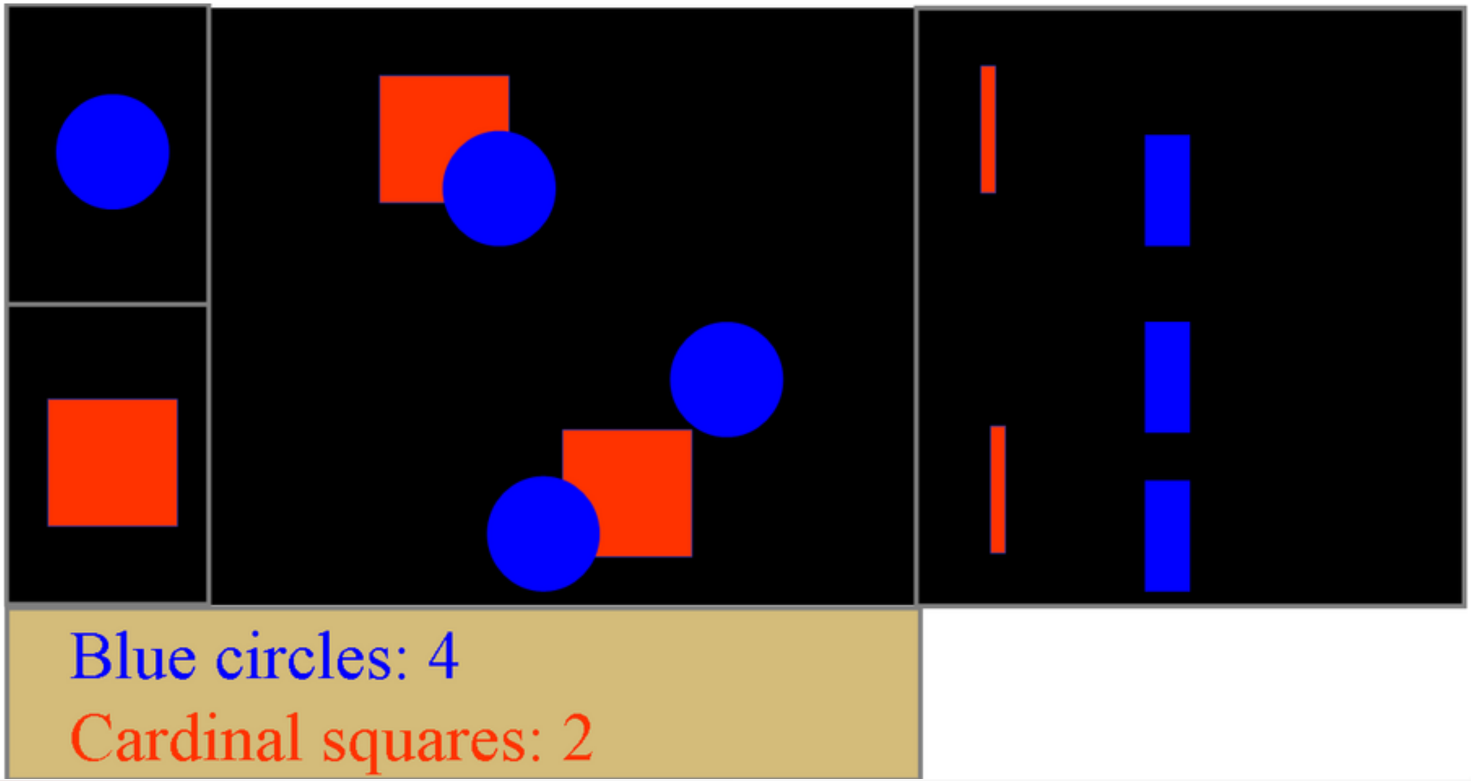
Combining the View and the Controller

- The View and Controller are tightly integrated
  - E.g. it requires communication to determine what was clicked
- Almost always occur in pairs
  - Need a separate controller for each view
- Many architectures combine View and Controller into a single unit.



Why MVC?

- Mixing all pieces in one place does not scale
- Separation eases maintenance and extensibility
  - Easy to add a new view later
  - Model can be extended, but old views still work
  - Views can be changed later (e.g. add 3D)



Blue circles: 4

Cardinal squares: 2

Nesting MVC

- MVC is useful on both large and small scales
- Can be used for a whole application
- Or, can be used within a complex widget
  - Complex components need to store internal state (a model) that affects their drawing and event handling
  - Simplifies internal implementation
  - Example: Many Java Swing components have an internal MVC architecture

For graphical applications, we often have:

- Some underlying model (e.g. the application your working with, like a basketball game, a payroll, a painting canvas, or a 3D model).
- Some display of interface and maybe the underlying data in the program -- might change based on the state of the underlying model.
- Some ability to control the program through user interaction -- generating input events with the mouse and keyboard, etc.

- How do the classes in your project fit into these categories?
- To what extent can we (or do we already) employ MVC?