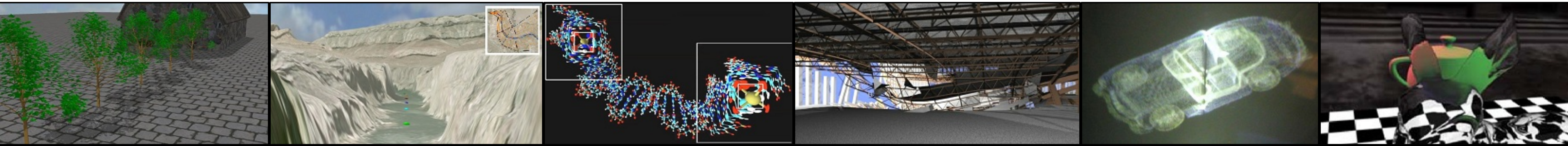
CIS 4930/6930-002
DATA VISUALIZATION

Model-View-Controller Design Pattern

Paul Rosen
Assistant Professor
University of South Florida

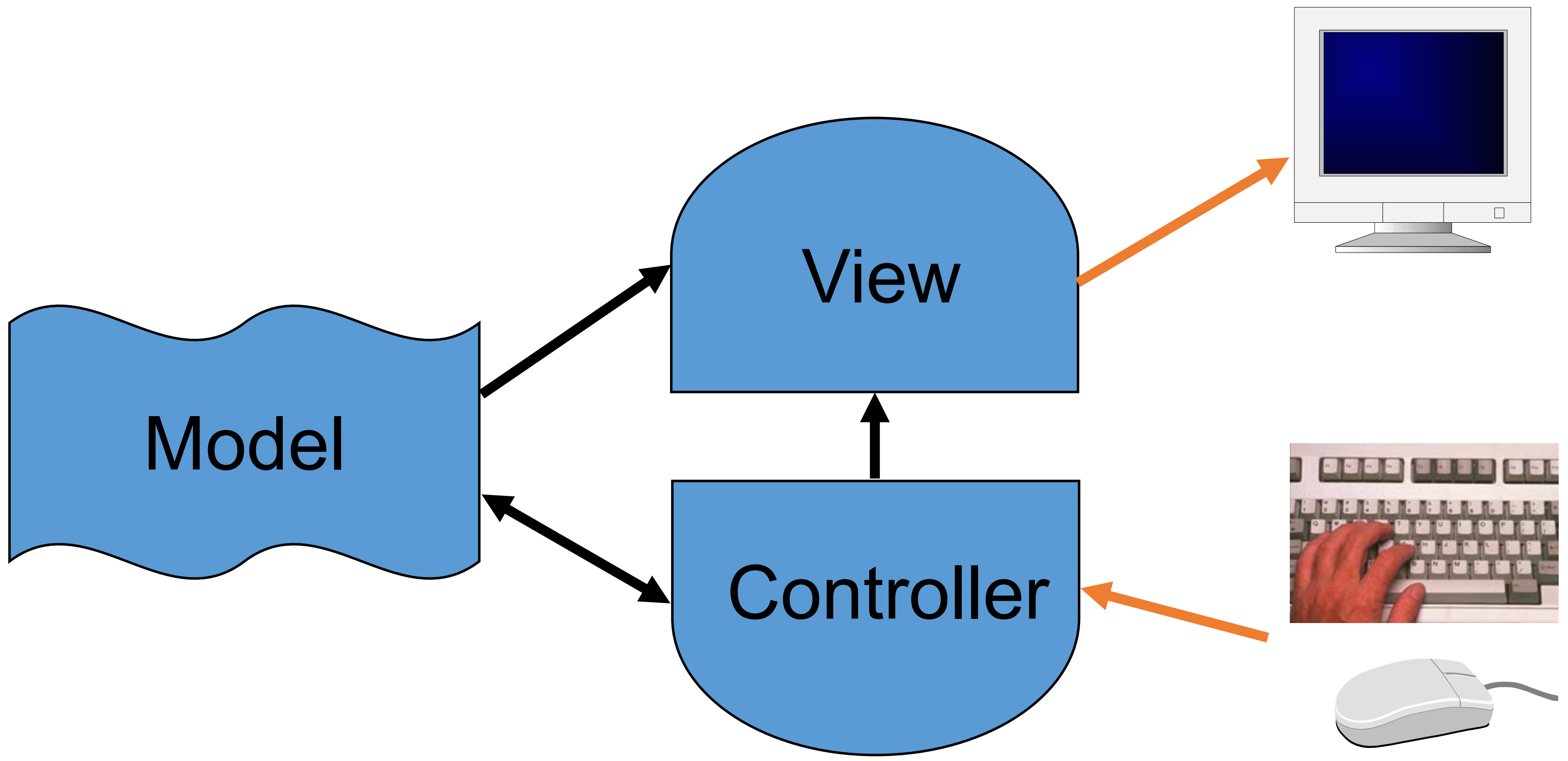


Model-View-Controller

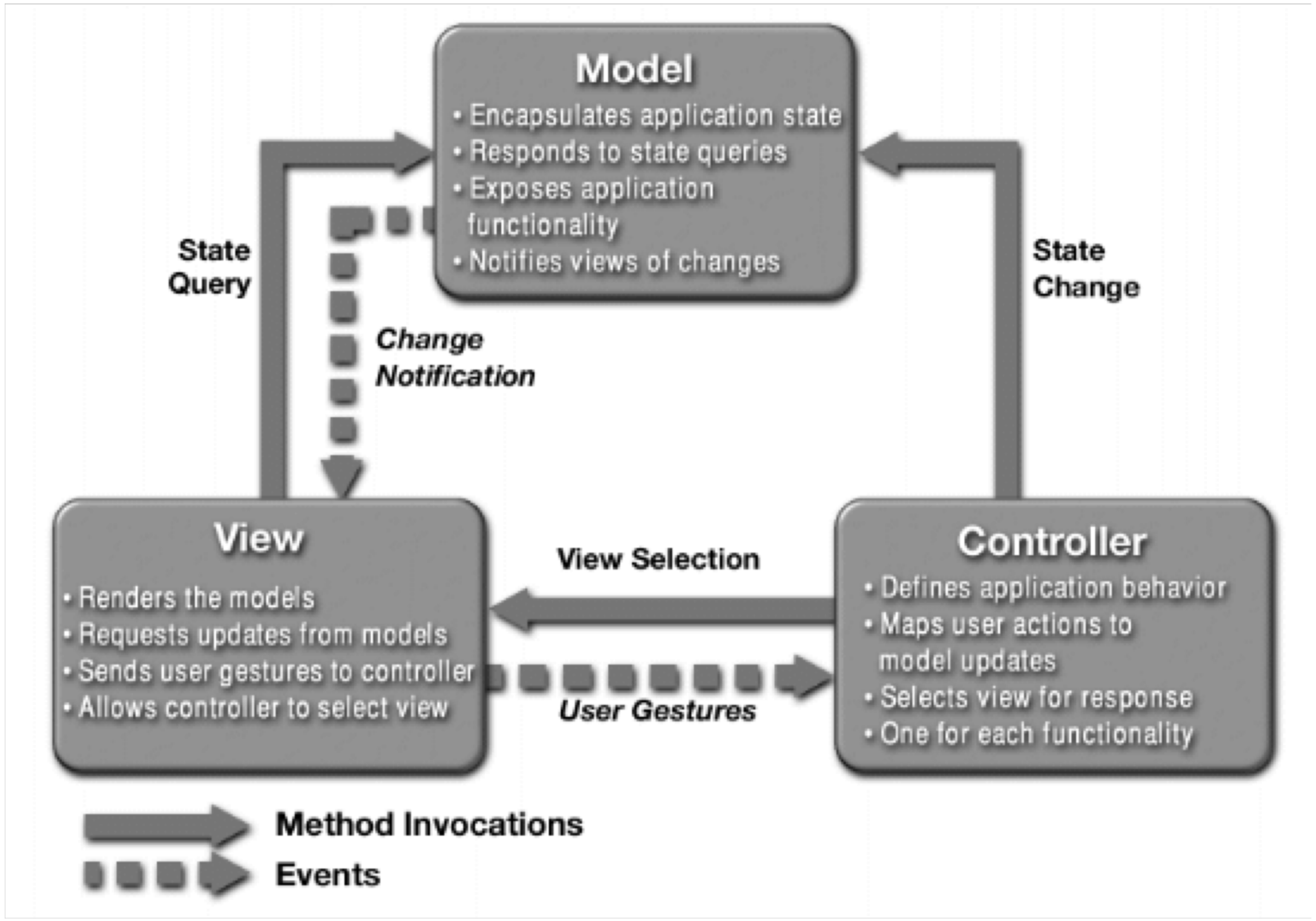
Architecture for interactive apps

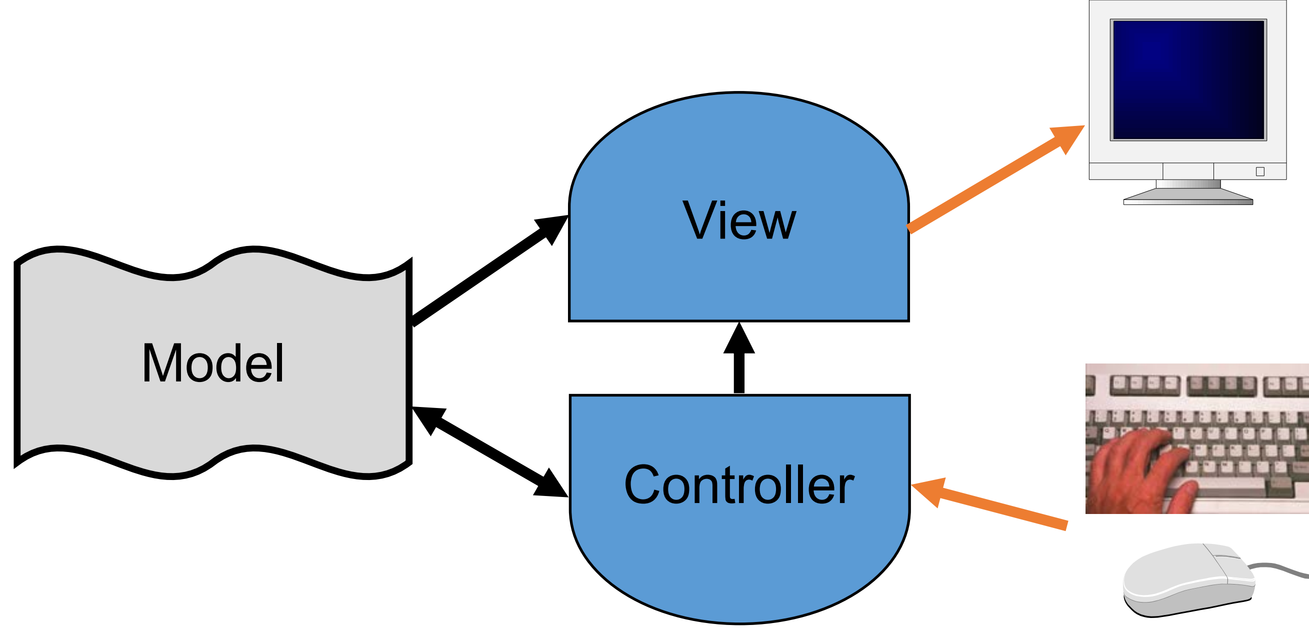
introduced by Smalltalk developers at XEROX PARC

Partitions application so that it is scalable and maintainable



What is MVC?


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events



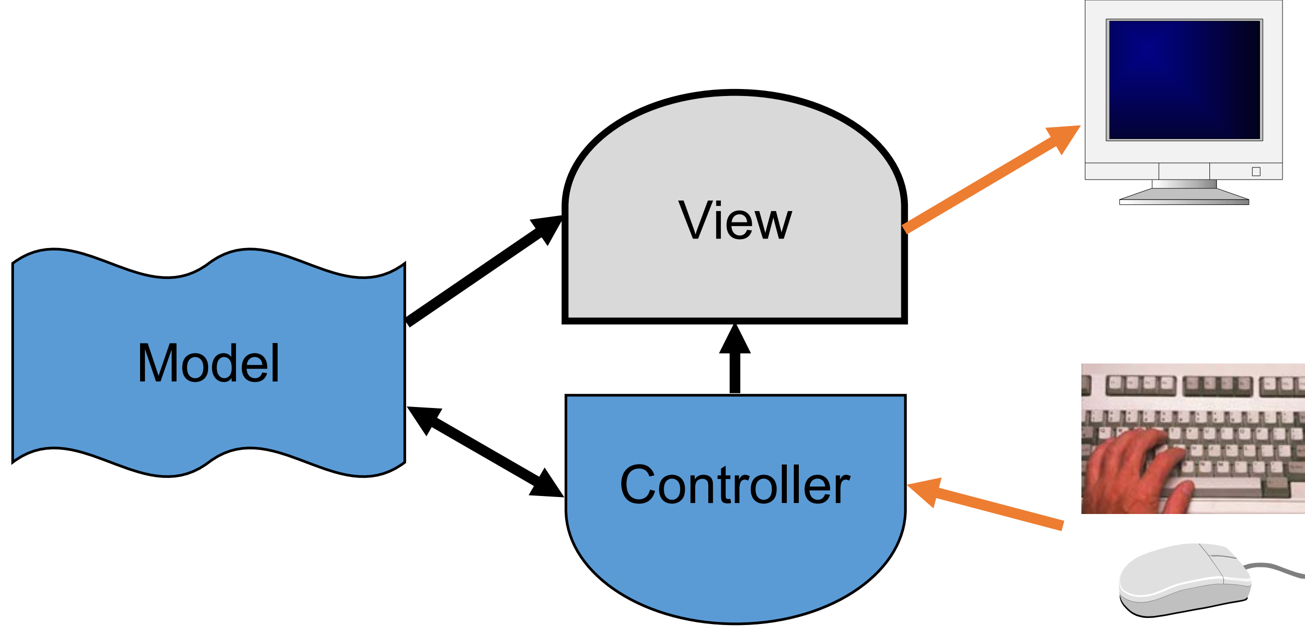
The Model

The Model is the part that does the work--it models the actual problem being solved (i.e. it stores the **data**)

The Model should be independent of both the Controller and the View

But it provides services (methods) for them to use

Independence gives flexibility, robustness



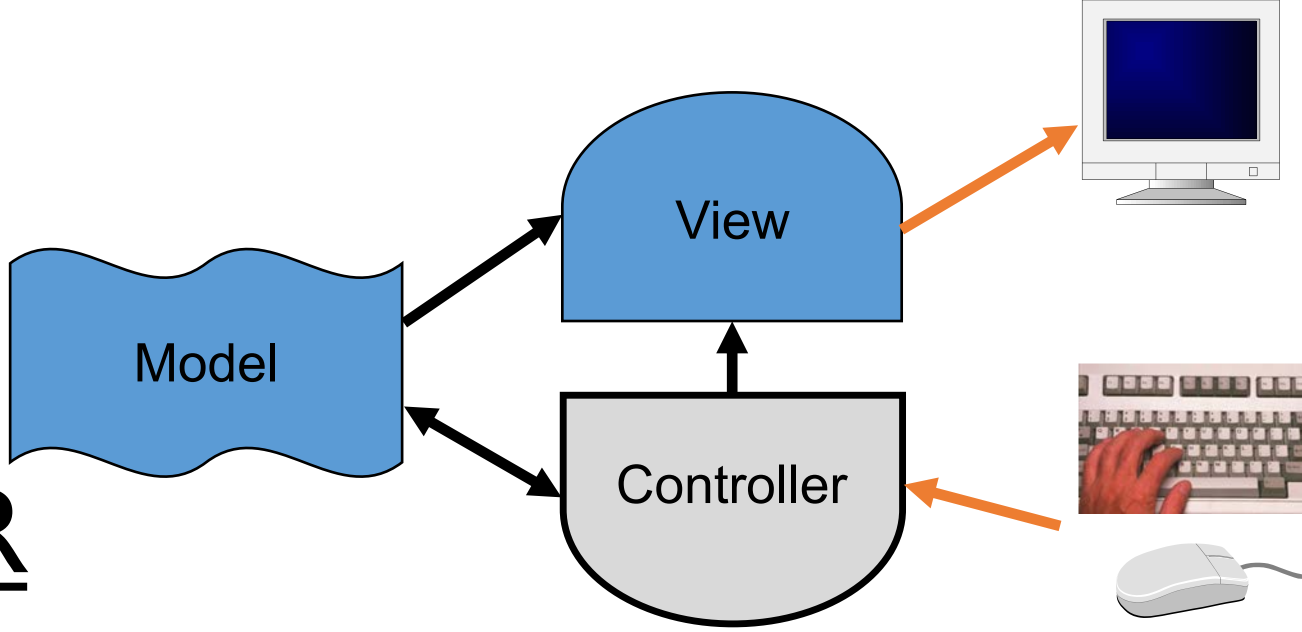
THE VIEW

The View shows what the Model is doing

The View is a passive observer; it should not affect the model

The Model should be independent of the View, but (but it can provide access methods)

The View should not display what the Controller thinks is happening



THE CONTROLLER

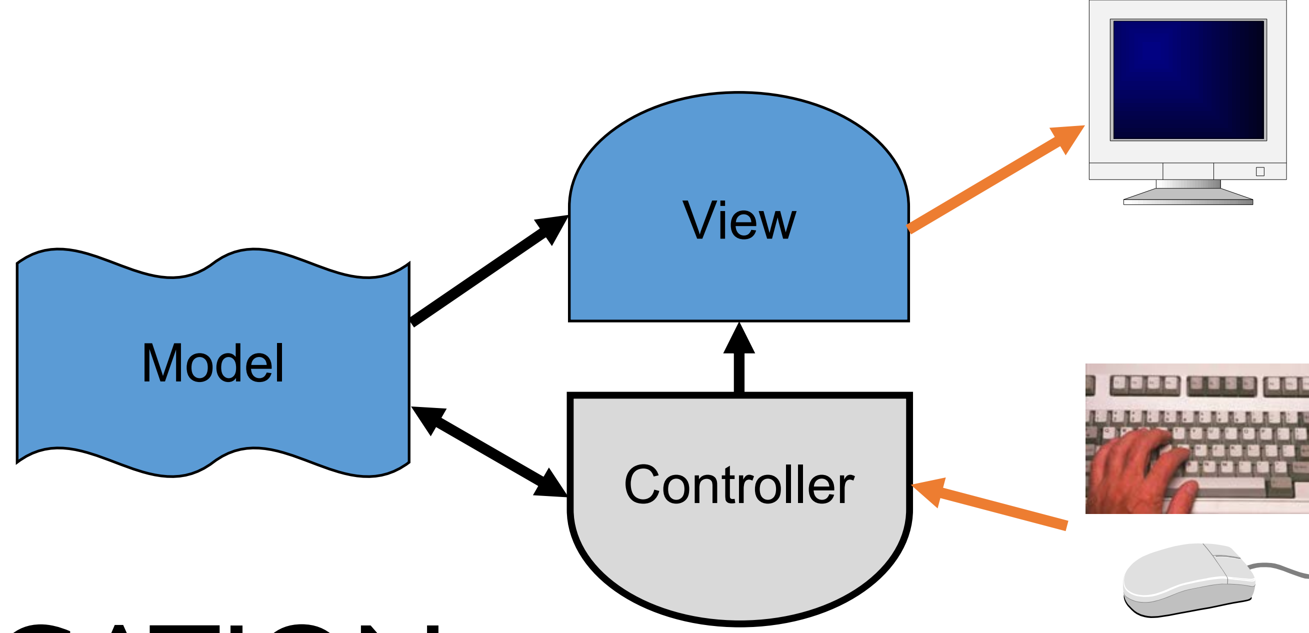
ften, the user is put in control by means of a GUI

in this case, the GUI and the Controller are often the same

The Controller and the Model can almost always be separated (what to do versus how to do it)

The design of the Controller depends on the Model

The Model should not depend on the Controller



CONTROLLER COMMUNICATION

Communicates with view

determines which objects are being manipulated (e.g., which object was selected with mouse click)

Calls model methods to make changes

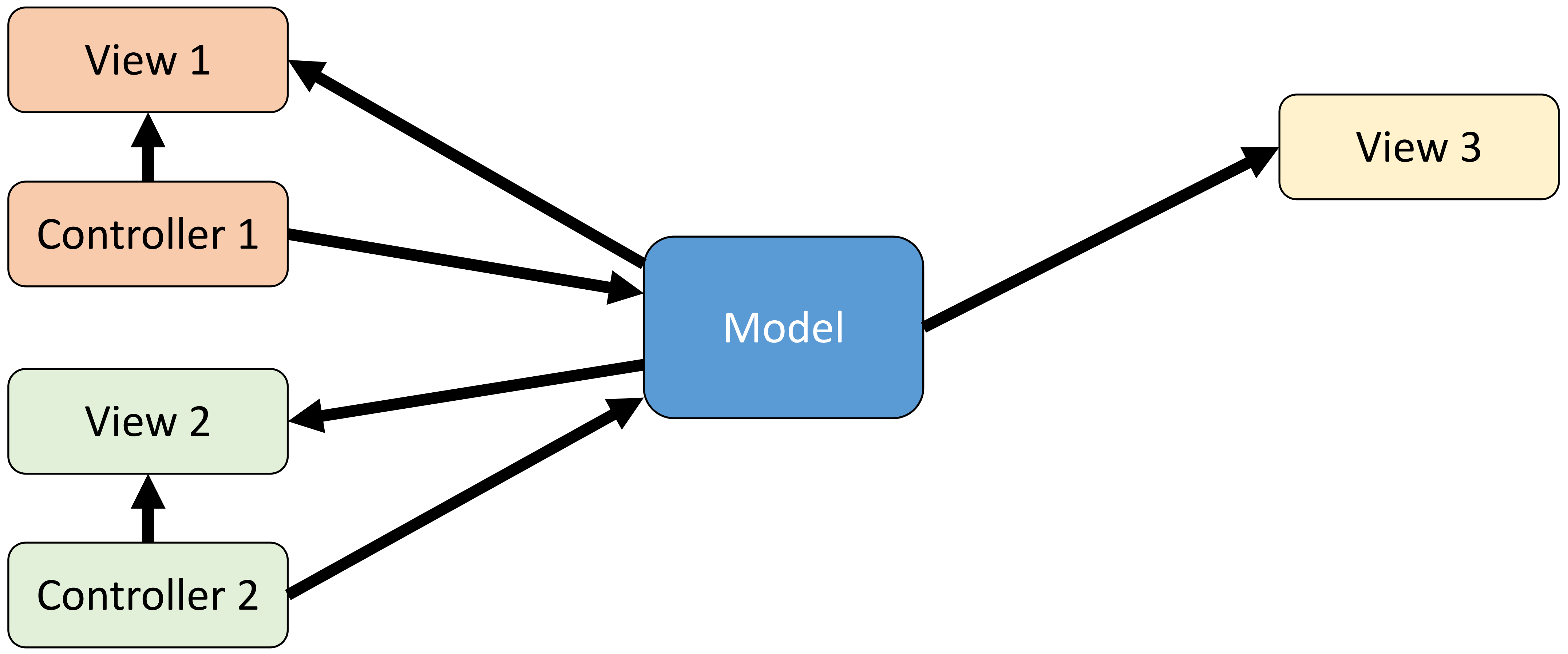
model makes change and notifies views to update

Advantages

Input processing is separated from output processing.

Controllers can be interchanged, allowing different user interaction modes.

Multiple views of the model can be supported easily.



Why MVC?

Provides a logical structure for heavily interactive system

Adheres to good engineering design principles and practices

Information hiding, less coupling, simplicity, etc.

Delegated control style

WHY MVC?

Combining MVC into one class or using global variables will not scale. Why?

model may have more than one view

each different & needing update on model changes

Separation eases maintenance. Why?

easy to add a new view later

may need new model info, but old views still work

can change a view later

e.g., draw shapes in 3-d

recall that the view handles selection

WHAT'S THE POINT?

It's just plain easier

(even if it doesn't look it)!

Studies show that most introductory CS college students tend to clump a UI program into one big monolithic main class

Unfamiliarity stems from student tendency to program according to styles presented in textbooks and not being introduced to design principles early