Model View Controller Pattern – Behavioural

- Intent
  - » **Partition user-interactive applications into three parts**
    - > **Model**
    - > **View**
    - > **Controller**

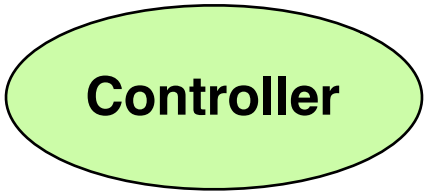
- Motivation
  - » **Use divide and conquer to simplify interactions among the parts of a program**

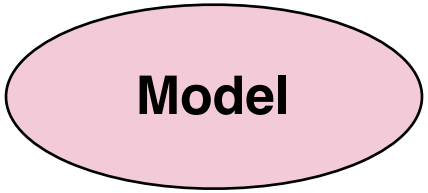
Participants – 1

- Model
  - » **What your application is**
    - > **Does computation, data manipulation and processing**
- Controller
  - » **How your model is presented to the user**
    - > **Connects the model with the view specifying what operations from the model must be executed in response to what user events occur**
- View
  - » **The controller's minions**
    - > **The graphical part of the application, presenting information visually and interacting with users.**
      - – **Notions such as buttons, other controls and events belong here**

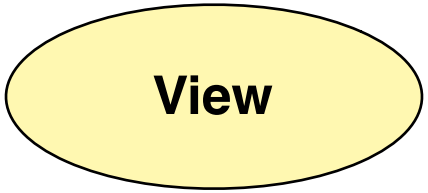
Participants – 2

How your model is presented to the user





What your application is

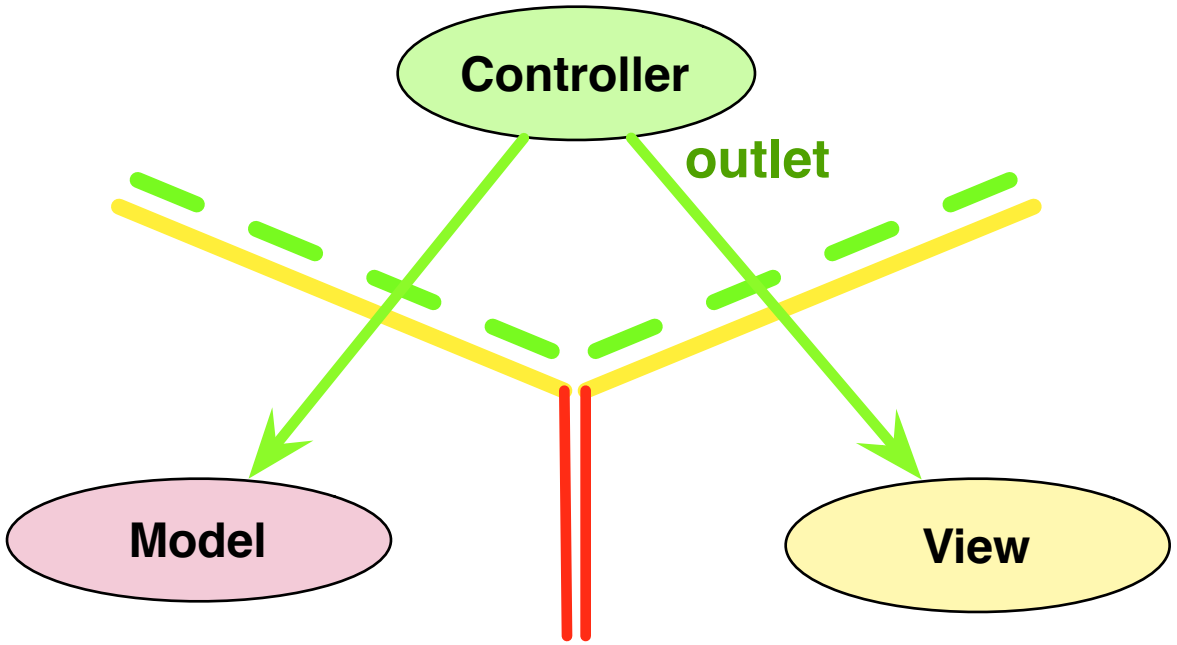


The controller's minions

Communications

It's all about managing the communications

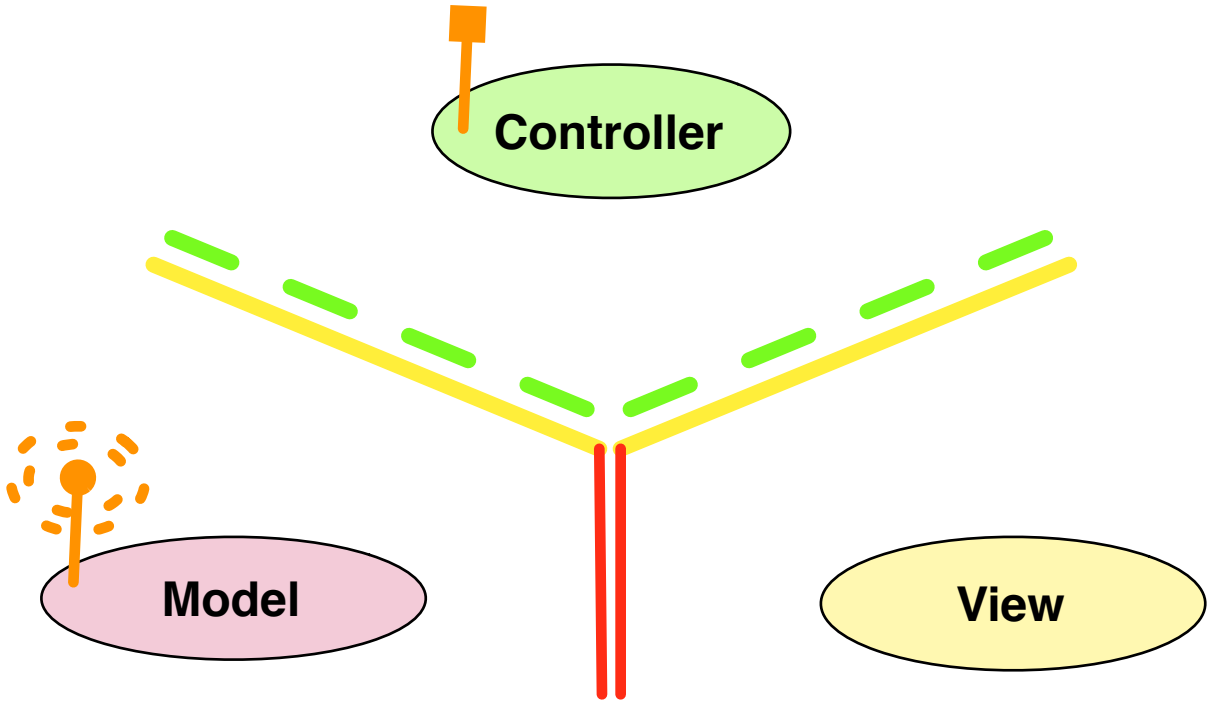
Controller can always talk to the model and view



Model and view never talk directly to each other

Communications – Model

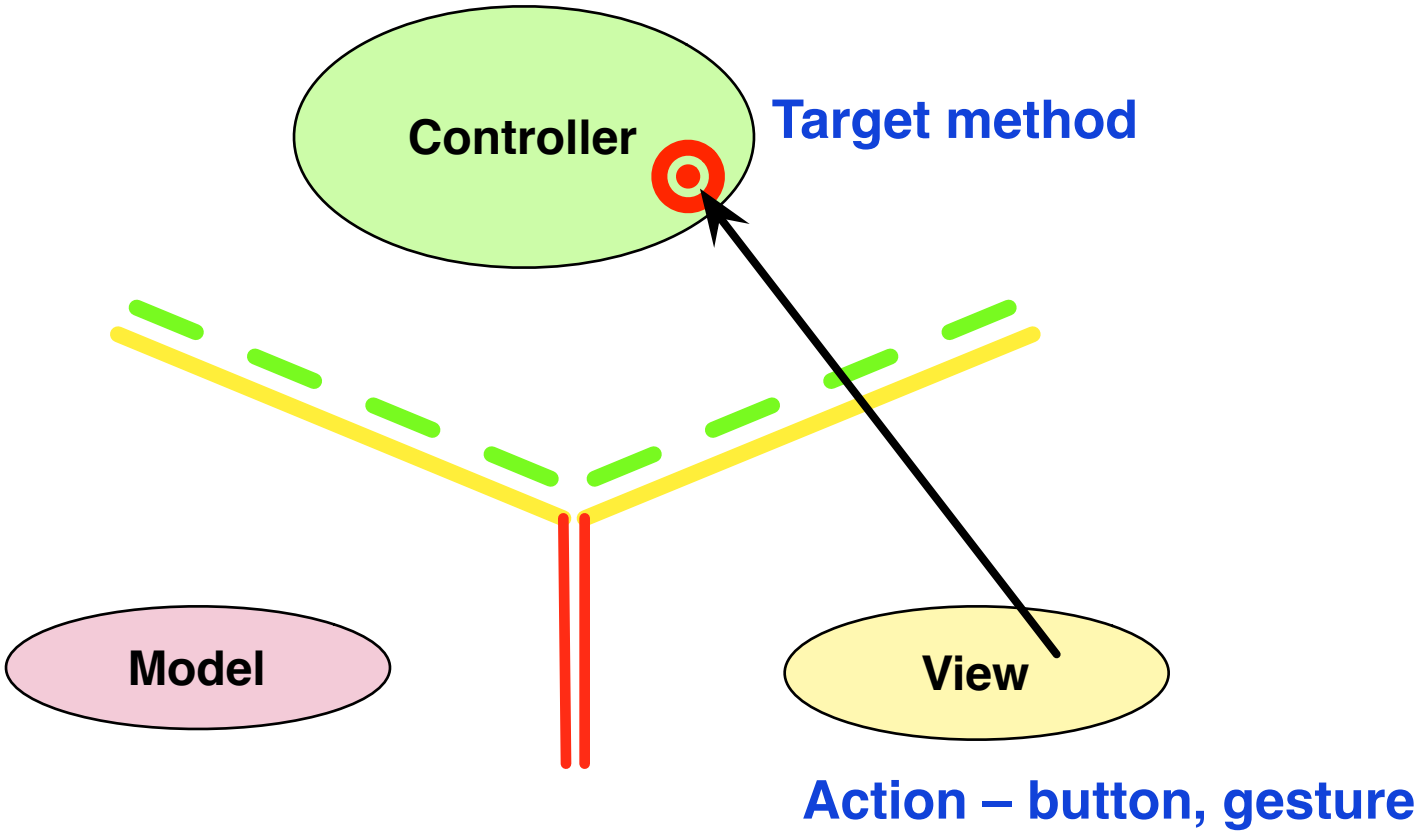
Controllers (and other models) tune into stations in which they are interested



Model has a radio station to broadcast change notifications (subject in observer pattern)

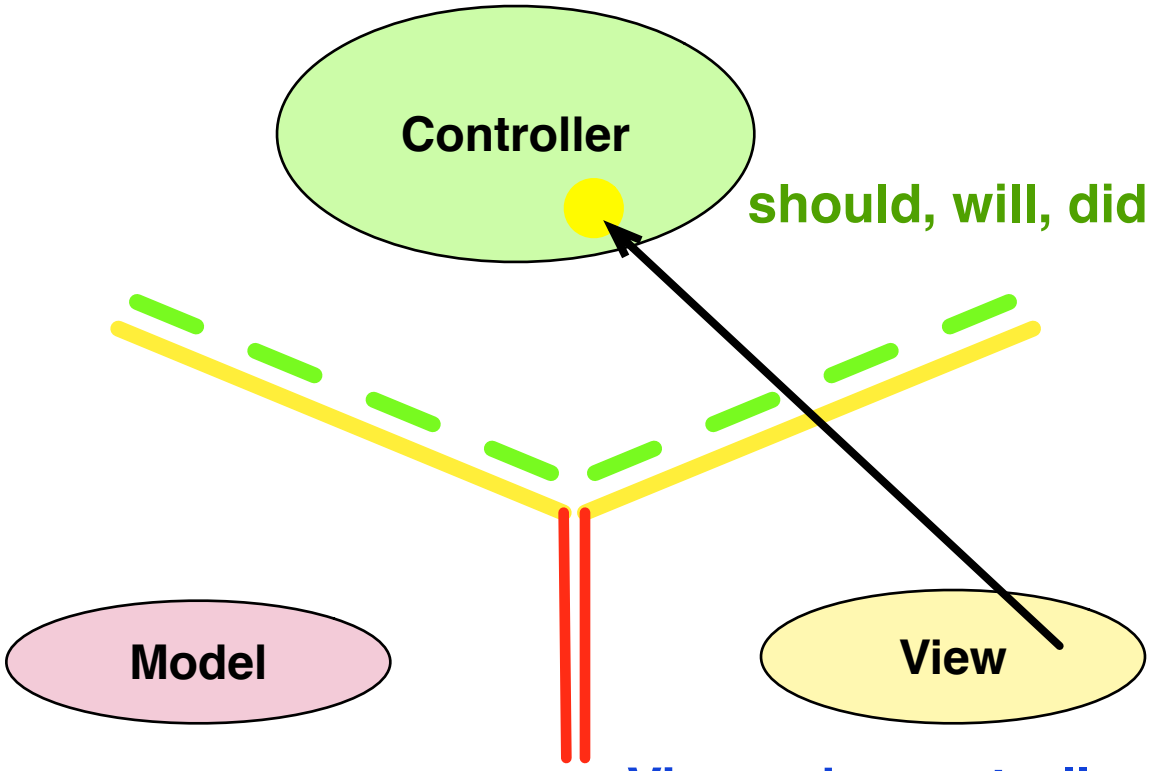
Models can only communicate directly with other models

Communications – View – 1 of 3



Communications – View – 2 of 3

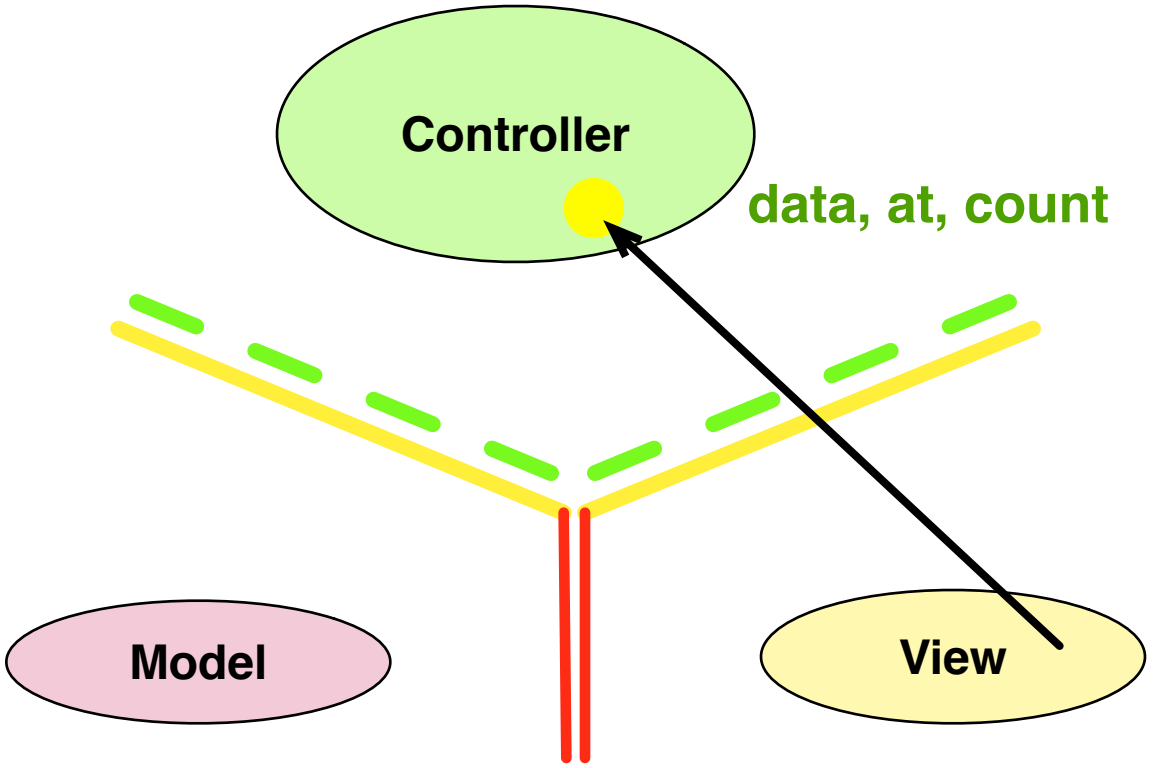
Controller sets itself as view's delegate



View asks controller to do work, indirectly as its delegate, and notifies the controller when the view changes

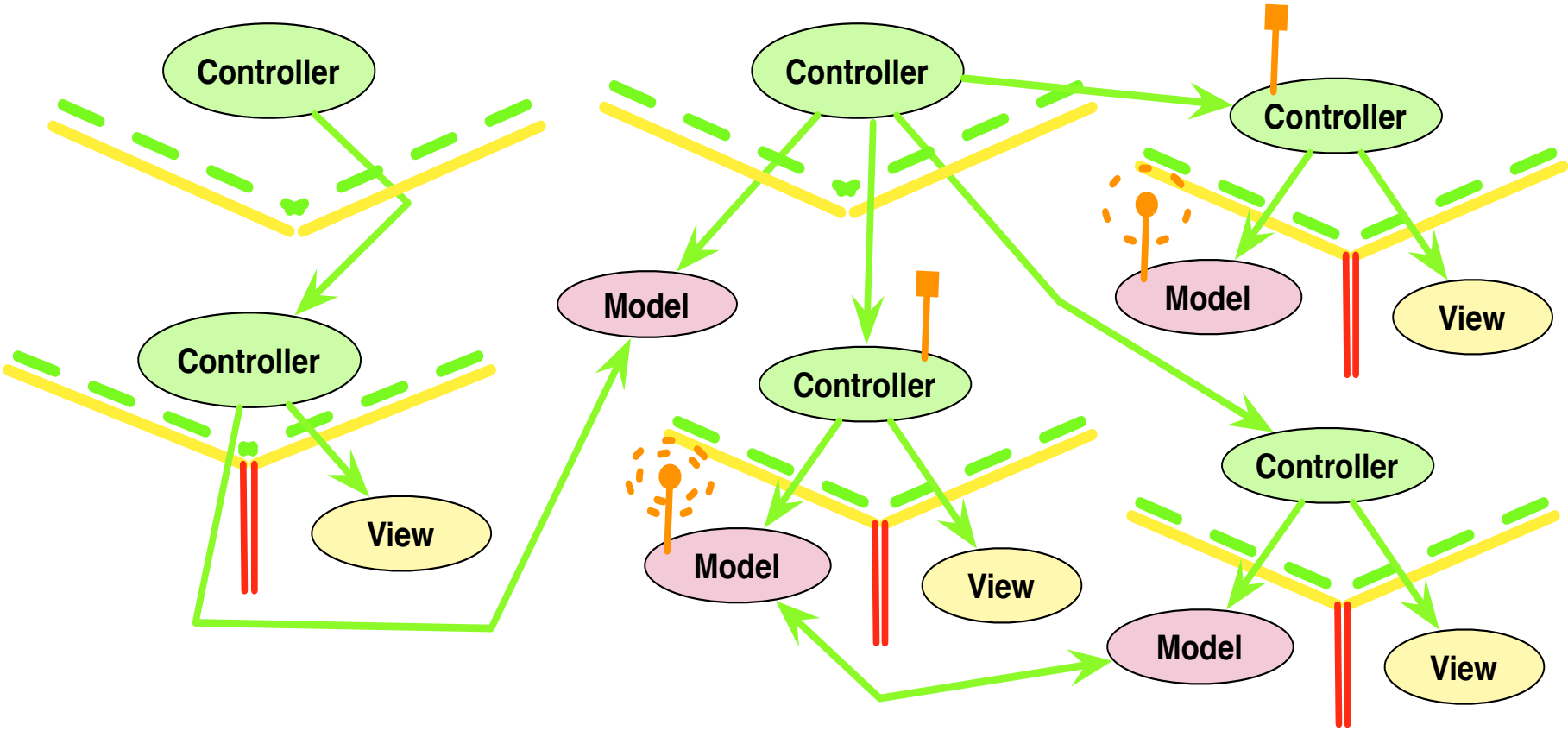
Communications – View – 3 of 3

Controller sets itself as view's data source

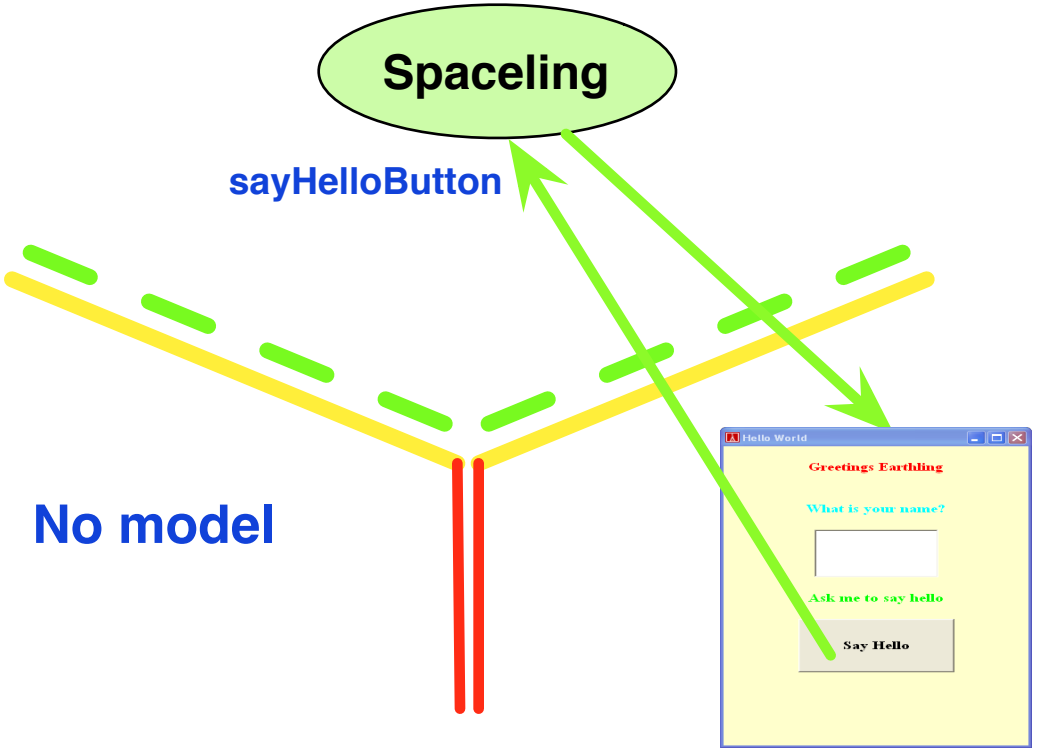


View asks controller for the data it needs

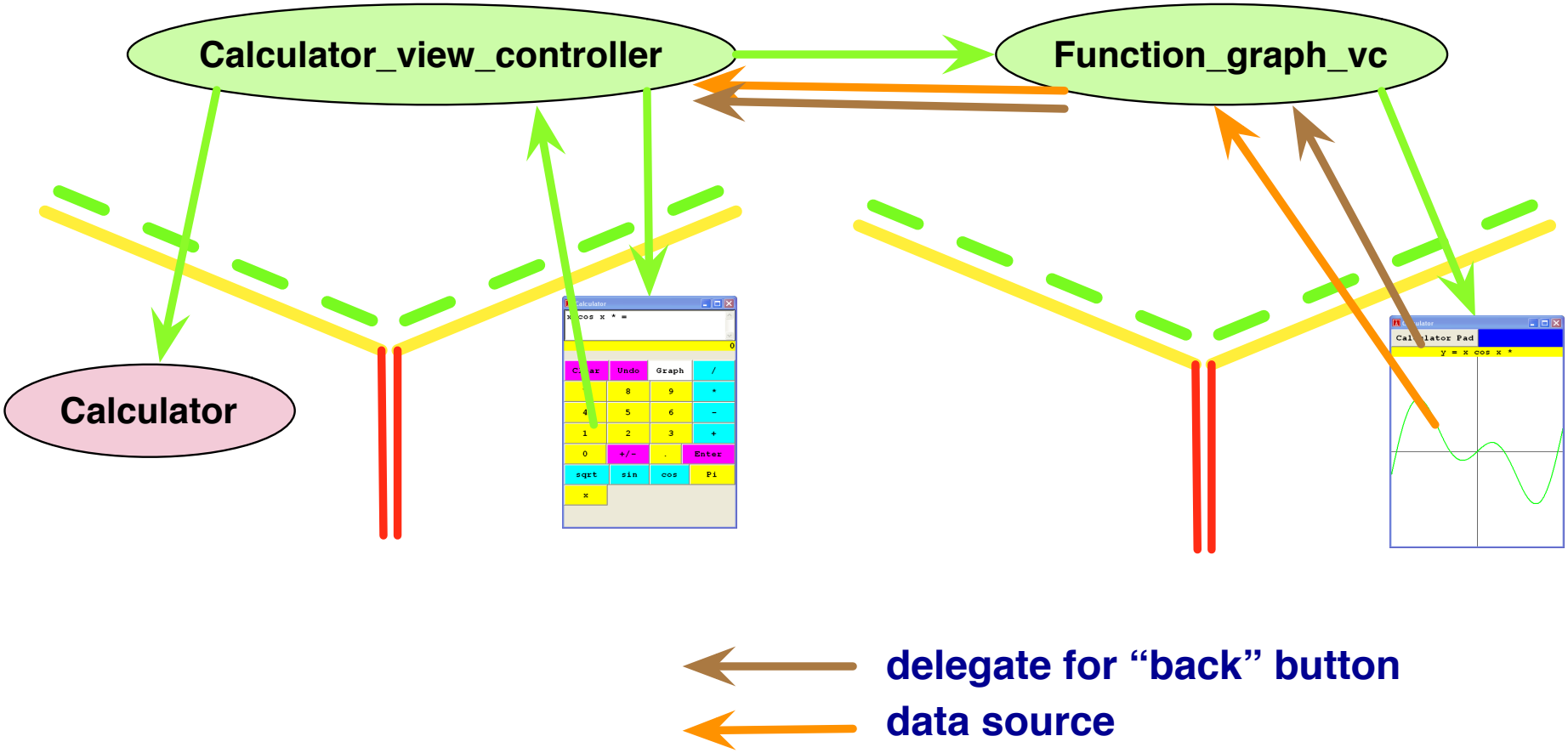
Multiple MVC patterns



Hello World Example



Calculator Example



delegate for “back” button

data source

Participants

- Model

  Has all the data and methods for maintaining the data

- View

  Displays relevant data, accepts user actions and notifies controller

- Controller

  - **Reacts to changes in the model to instruct how the view is to change**
  - **Reacts to view "actions" to instruct the view how to change and to instruct how the model is to change**

Applicability

- Use MVC in all interactive applications

Consequences

- The accepted way to deal with interactive applications

Related Patterns

- MVC is a special case of the Mediator pattern
  - » **The controller is the mediator between the model and the view**
- MVC has similarities to the Observer pattern.