

MVC AND MULTIPLE VIEWS

CMPT 381

Overview

Review of MVC

Listener-based view updates

Example 1: direct connection

Multiple view systems

Example 2: listener, one view

Example 3: listener, two views

Example 4: listener, three views

Storing the selection in the model

Easy

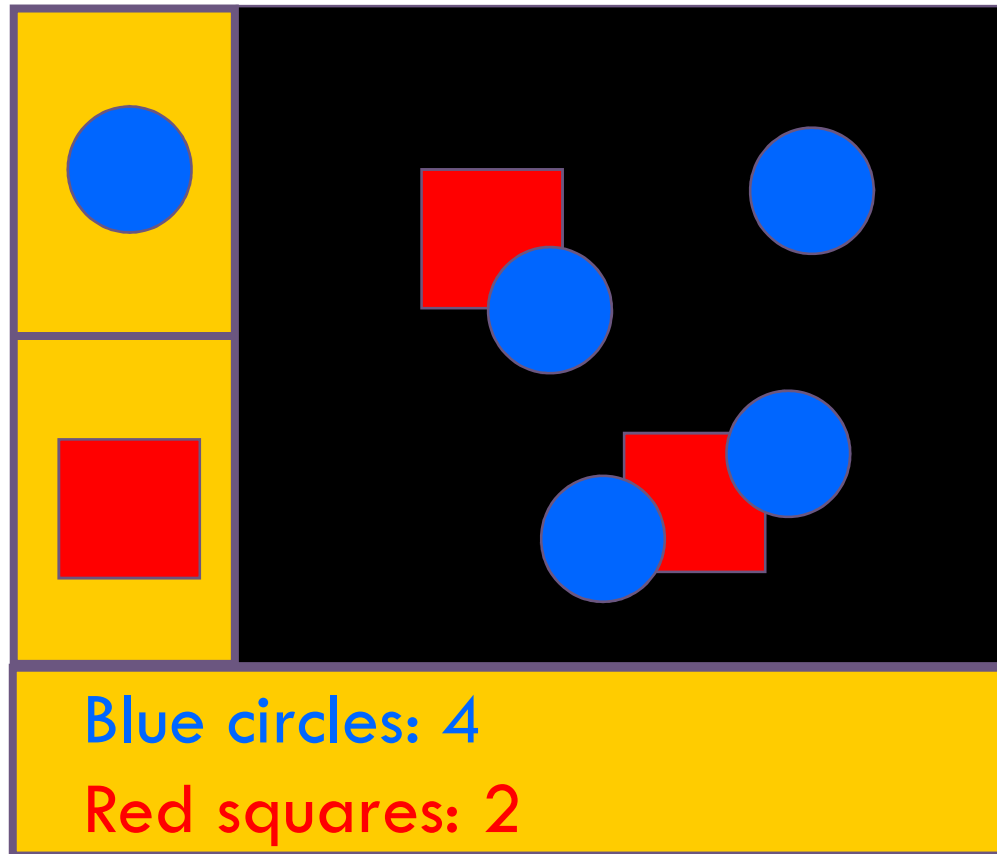
```
listbox .list  
.list insert 0 {saskatoon 36} {regina 18} \  
           {yorkton 7}   {aberdeen 40}
```



But now...

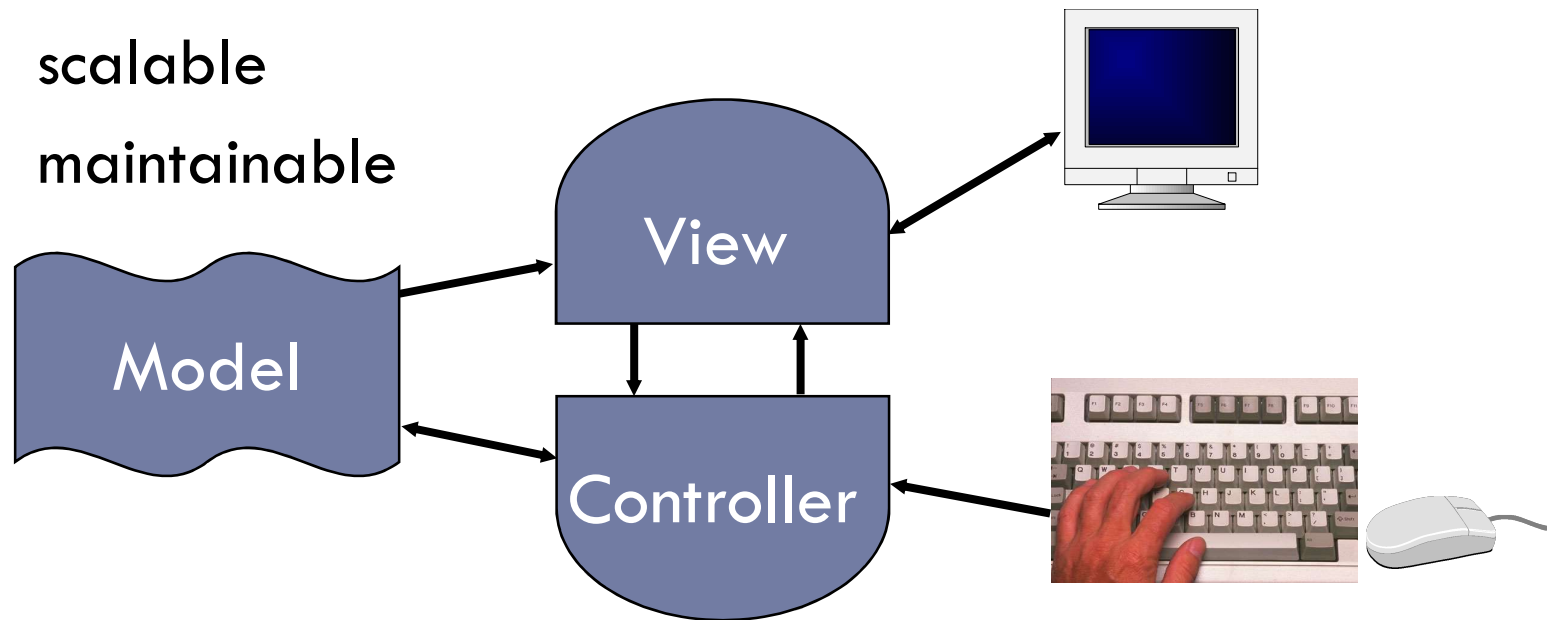
- Show the names in sorted order
- Show the names sorted by temperature
- Show a map view in addition to the list
- Show only the map view

How to store this data in the UI?



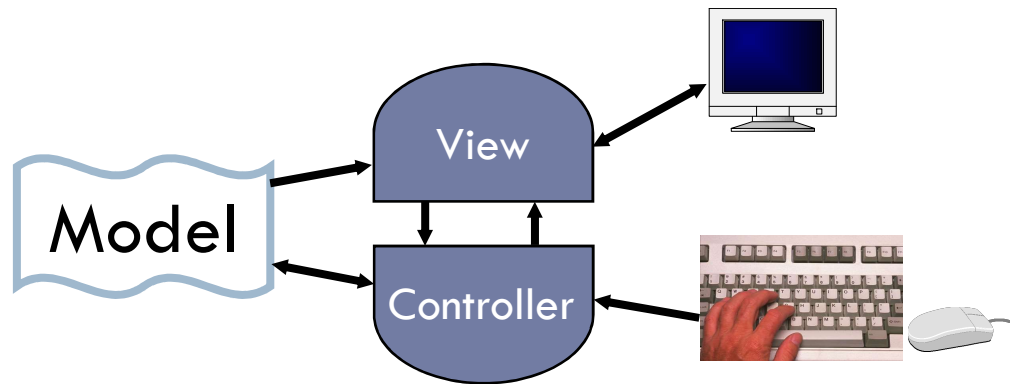
Model-View-Controller

- An architecture for interactive applications
 - introduced by Smalltalk developers at PARC
- Partitions application so that it is:
 - scalable
 - maintainable



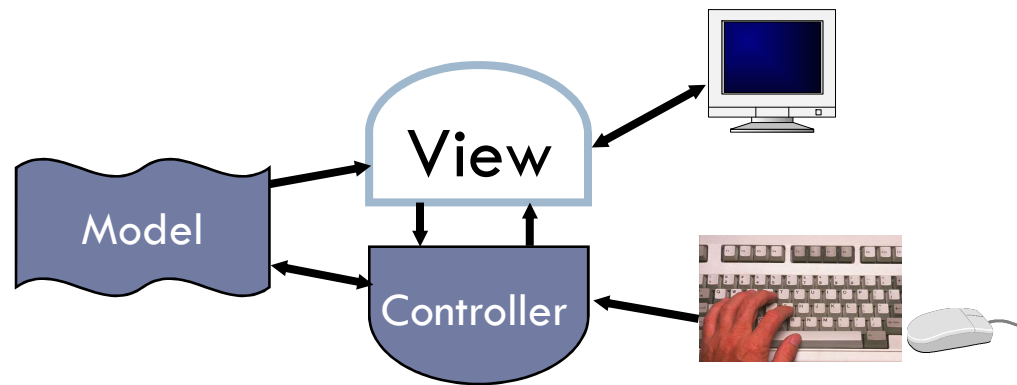
The model

- The information that the application manipulates
- “The data”
 - E.g. the circuit in a design program
 - Logic gates and wires connecting them
 - E.g. a drawing in a drawing program
 - Shapes, locations, colour

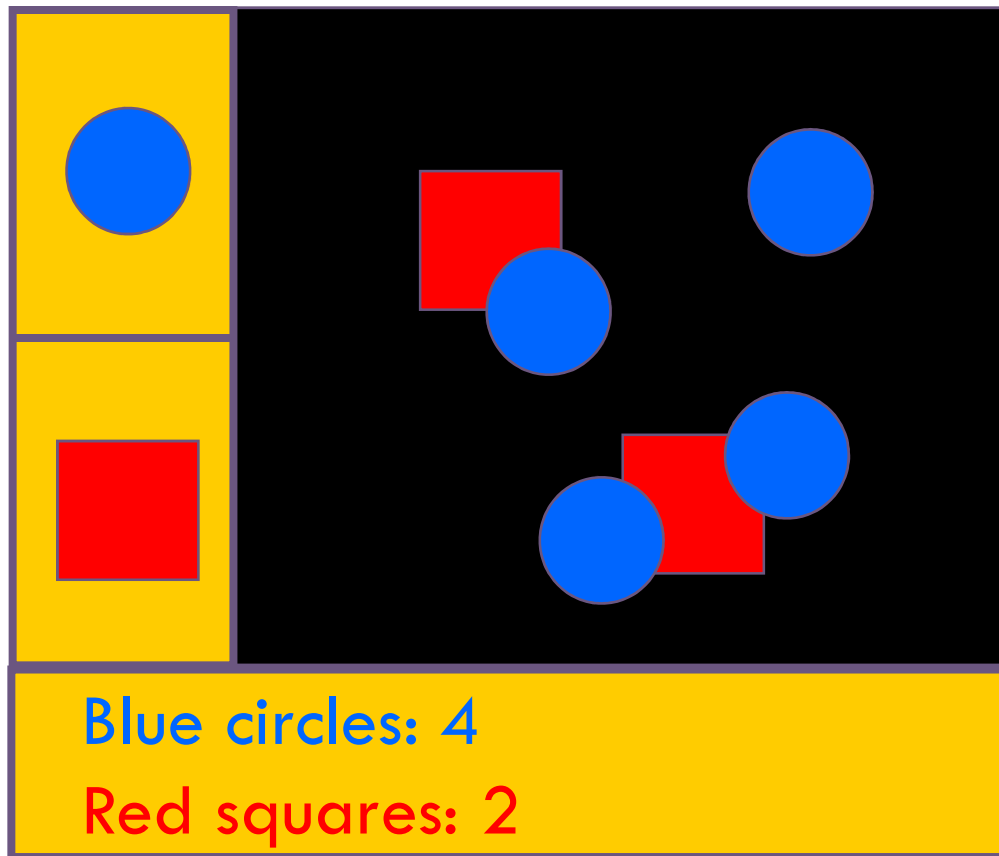


The view

- A visual display of the model
- There may be multiple views
- View is redrawn when the model changes



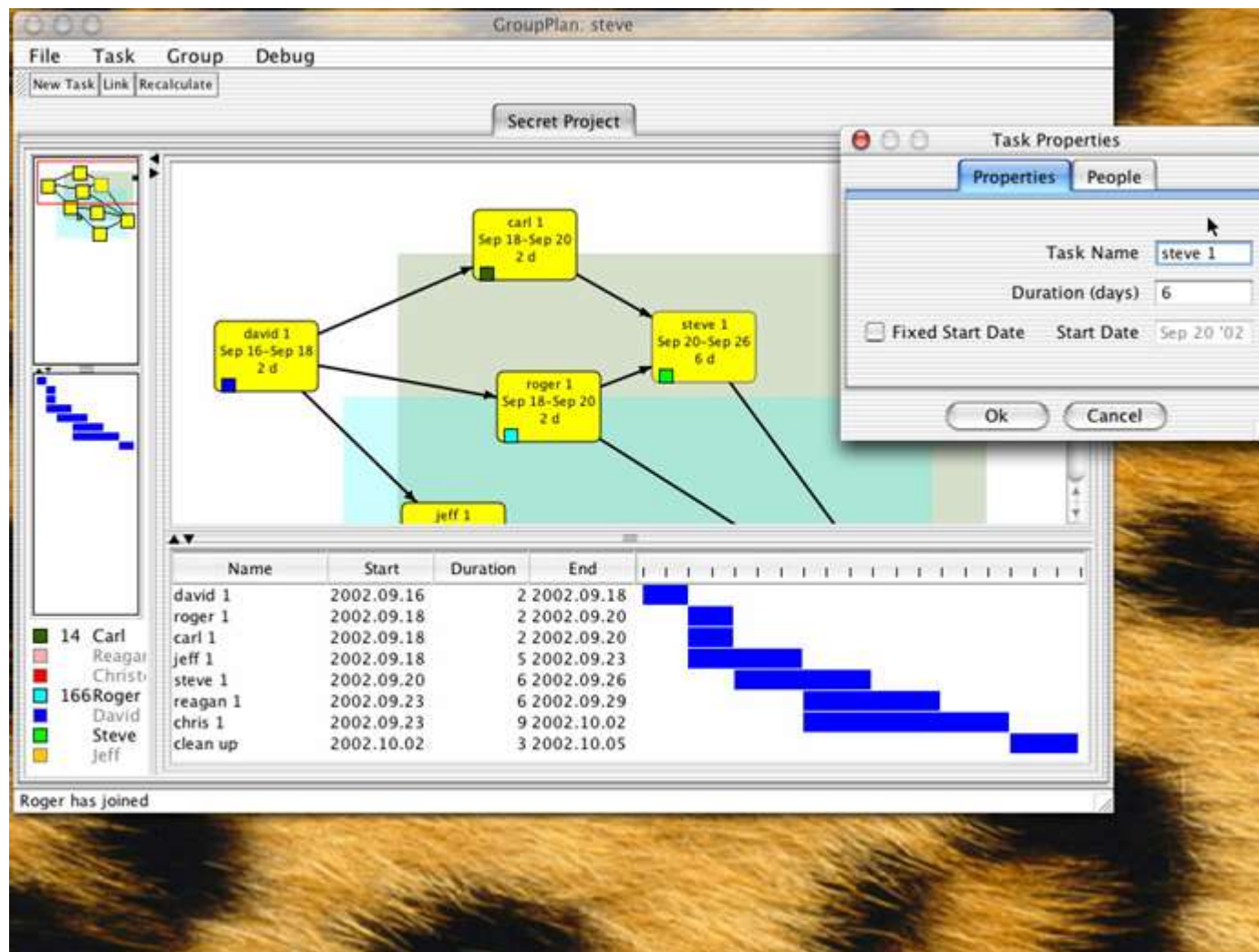
Multiple views



Shape view

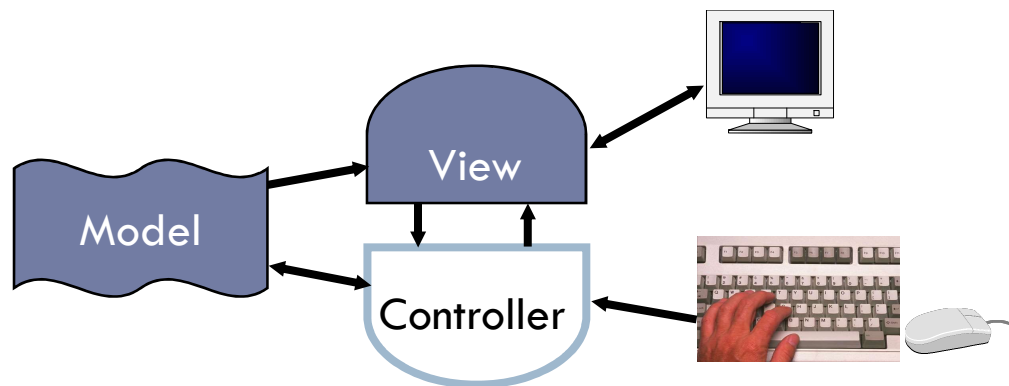
Numerical View

Multiple views



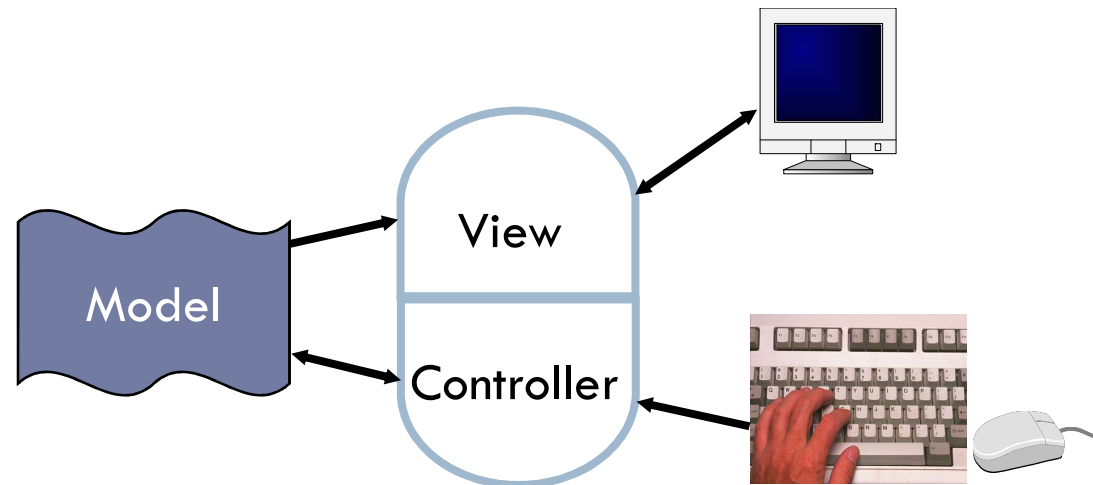
The controller

- Deals with application-level input events
- Communicates with view
 - e.g. which object was selected with mouse click
- Calls model methods to make changes



Combined view and controller

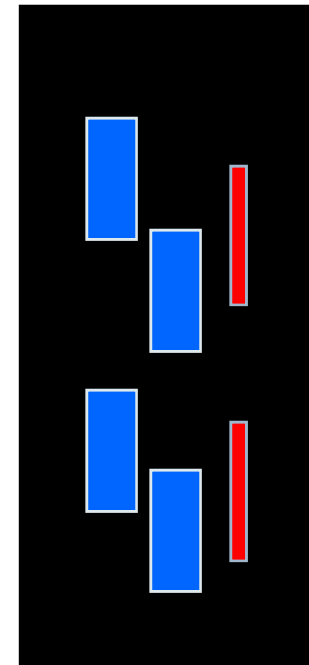
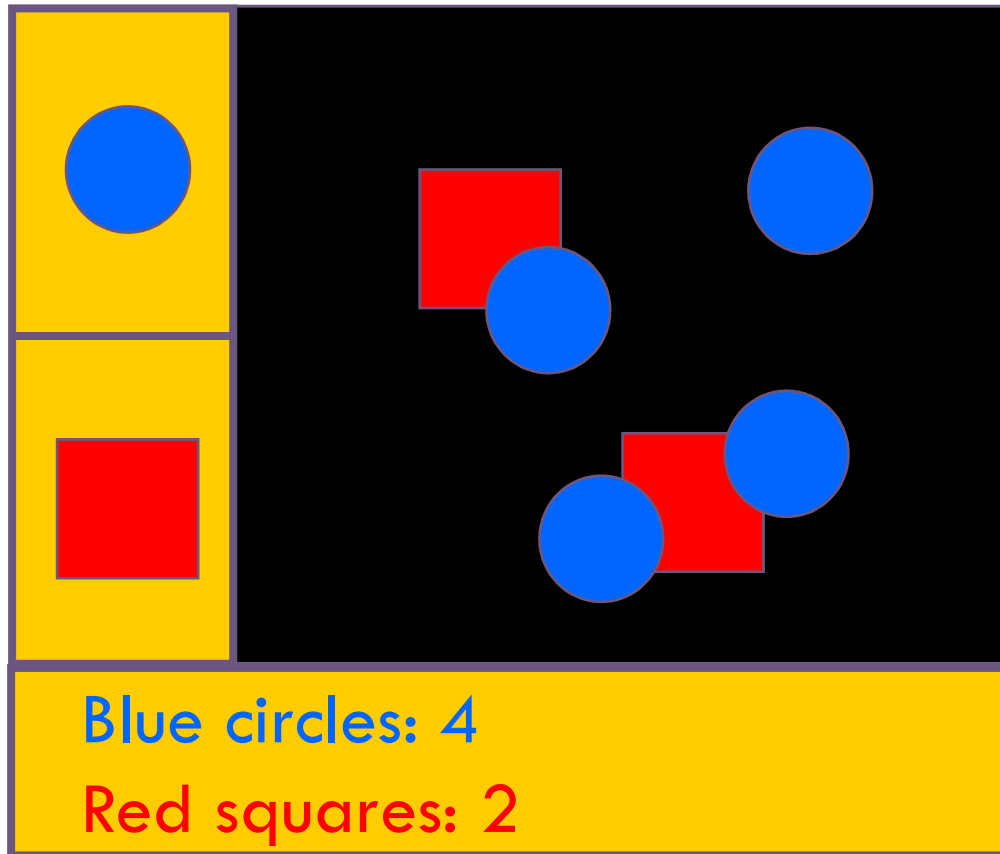
- View and controller are tightly coupled
- Almost always occur in pairs
 - each view needs a separate controller
- Some architectures combine them into one class



Why MVC?

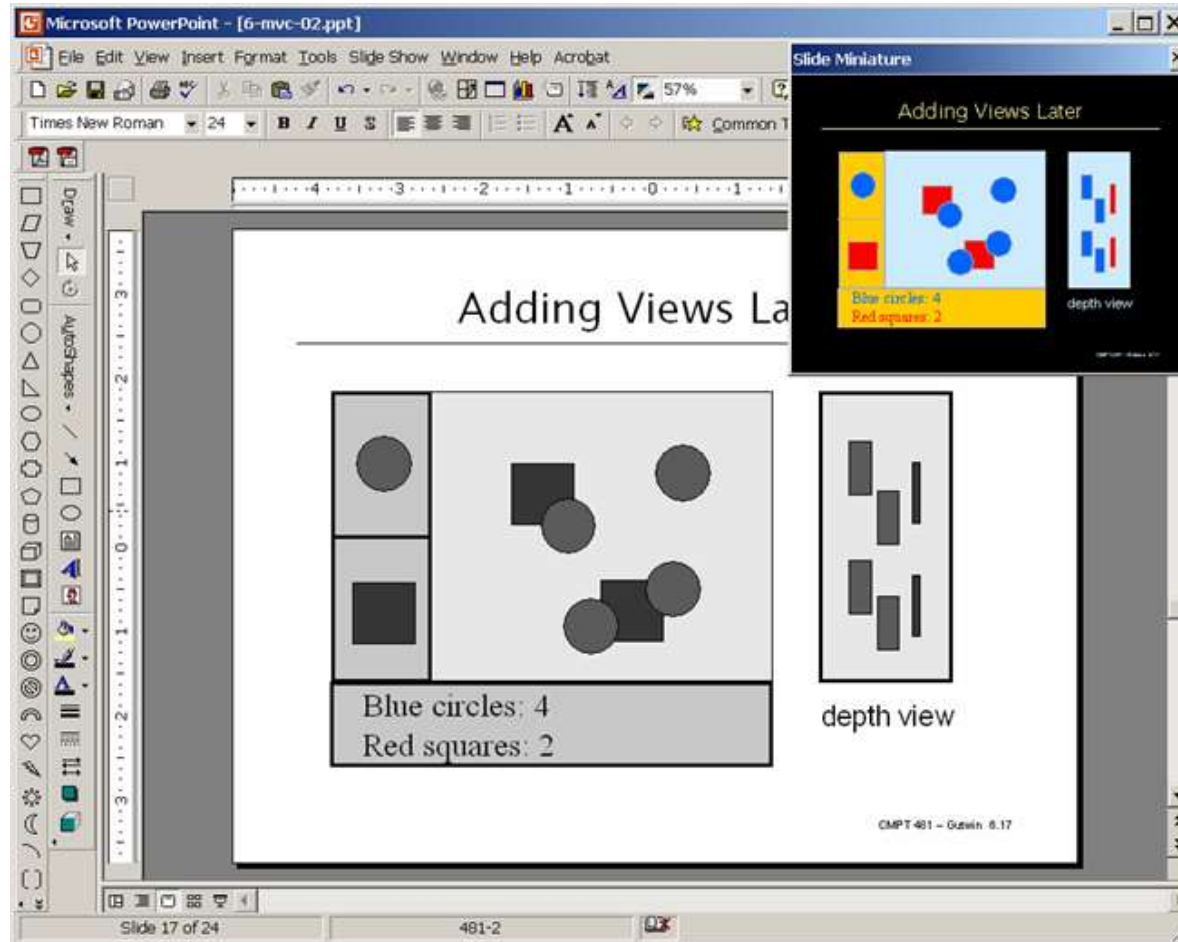
- Alternate (bad) approaches
 - Storing your data in your UI
 - Combining all of MVC into one class
 - Using global variables
- Separation simplifies scalability
 - A model may need/want more than one view
- Separation simplifies maintenance
 - Adding new views or changing views
 - Adding to or changing the model

Adding views later

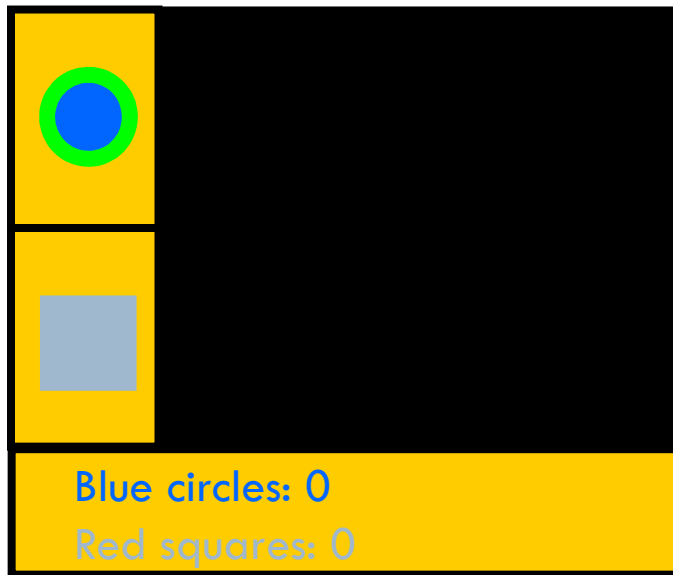


depth view

Adding views later

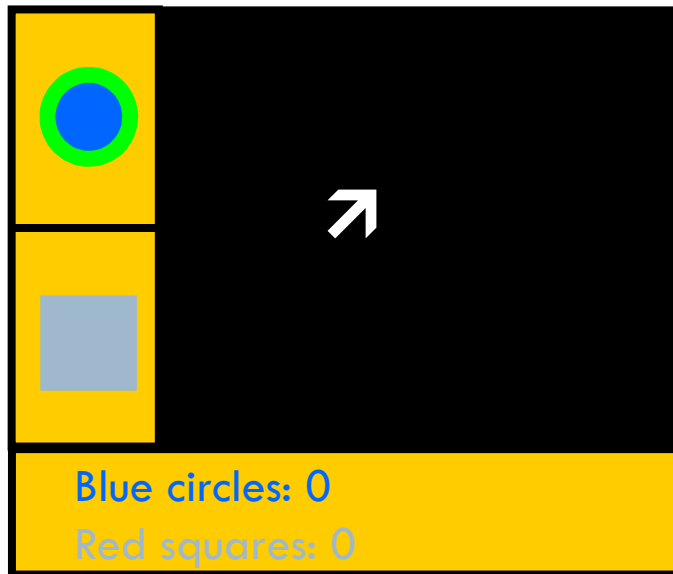


MVC event flow



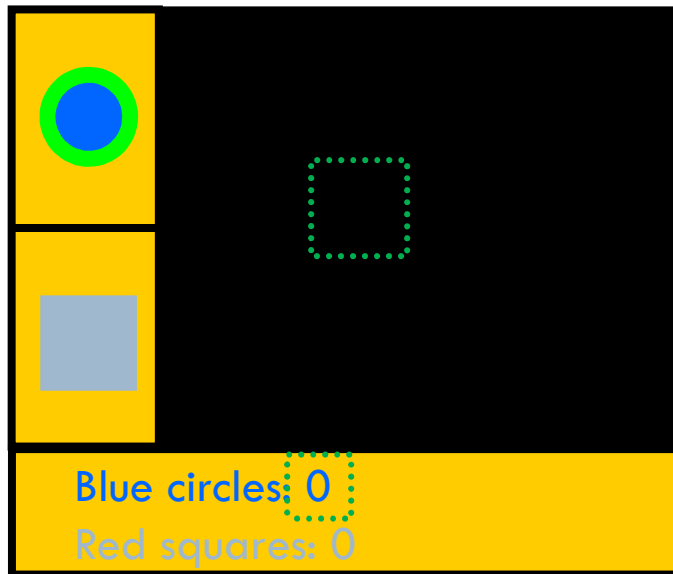
- Creating a new shape
- Assume “circle tool” selected
- User:
- Controller:
- View:
- Model:
- Toolkit:

MVC event flow



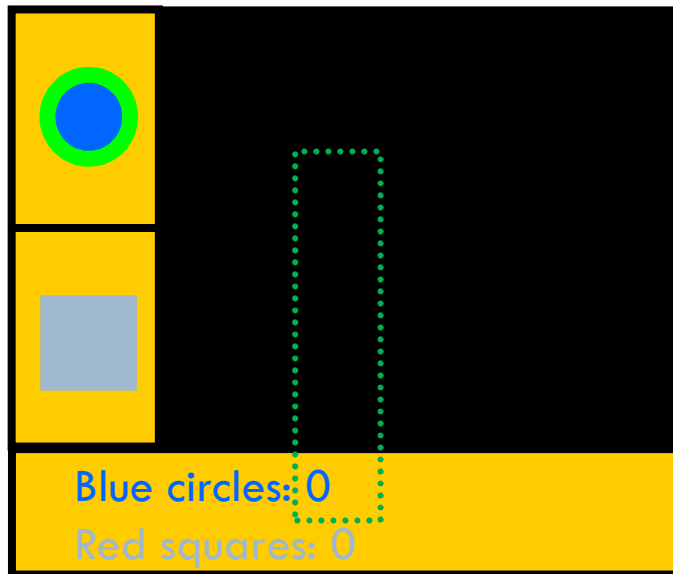
- User:
 - click mouse at location
- Toolkit:
 - forward event to handler
- Controller:
 - receive mouse click event
 - check mode → "circle"
 - call `model.addCircle(x,y)`

MVC event flow



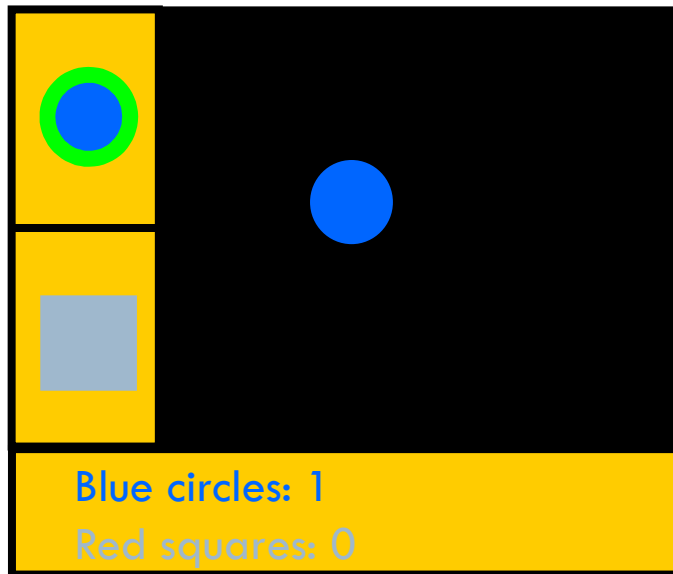
- **Model:**
 - `model.addCircle` adds the shape to a data structure
 - model notifies its list of views that a change has occurred
 - drawing area view
 - text summary view
- **View:**
 - call `toolkit.repaint()` or `toolkit.repaint (rectangle)`

MVC event flow



- View:
 - returns to model
- Model:
 - returns to controller
- Controller:
 - returns to toolkit
- Toolkit:
 - notices repaint request
 - is the area visible?
 - schedules call to view's paint method

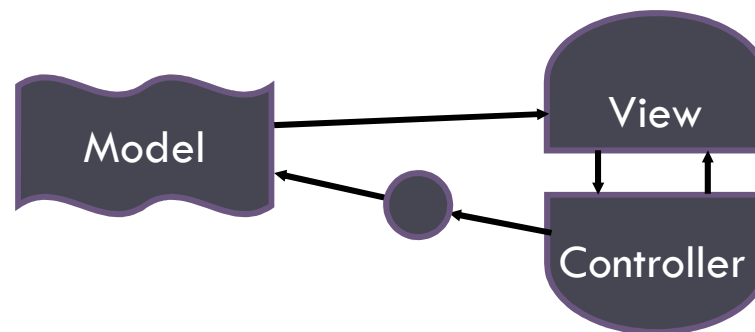
MVC event flow



- View:
 - request needed information from model
 - redraw view
- Toolkit:
 - bitblt rectangle onto screen

Controller-Model communication

- Direct connection
 - UI controller has visibility to model
- Do we want real model-view separation?
 - Changing the model
- Controller pattern
 - A gatekeeper for talking to the model
 - UI controller can only see the gatekeeper



Model-View communication

- Direct connection
 - Model or controller calls view methods directly
- Publish-subscribe
 - Views register for notifications from model objects
- Event approach
 - Use the actual event system to do the communication

