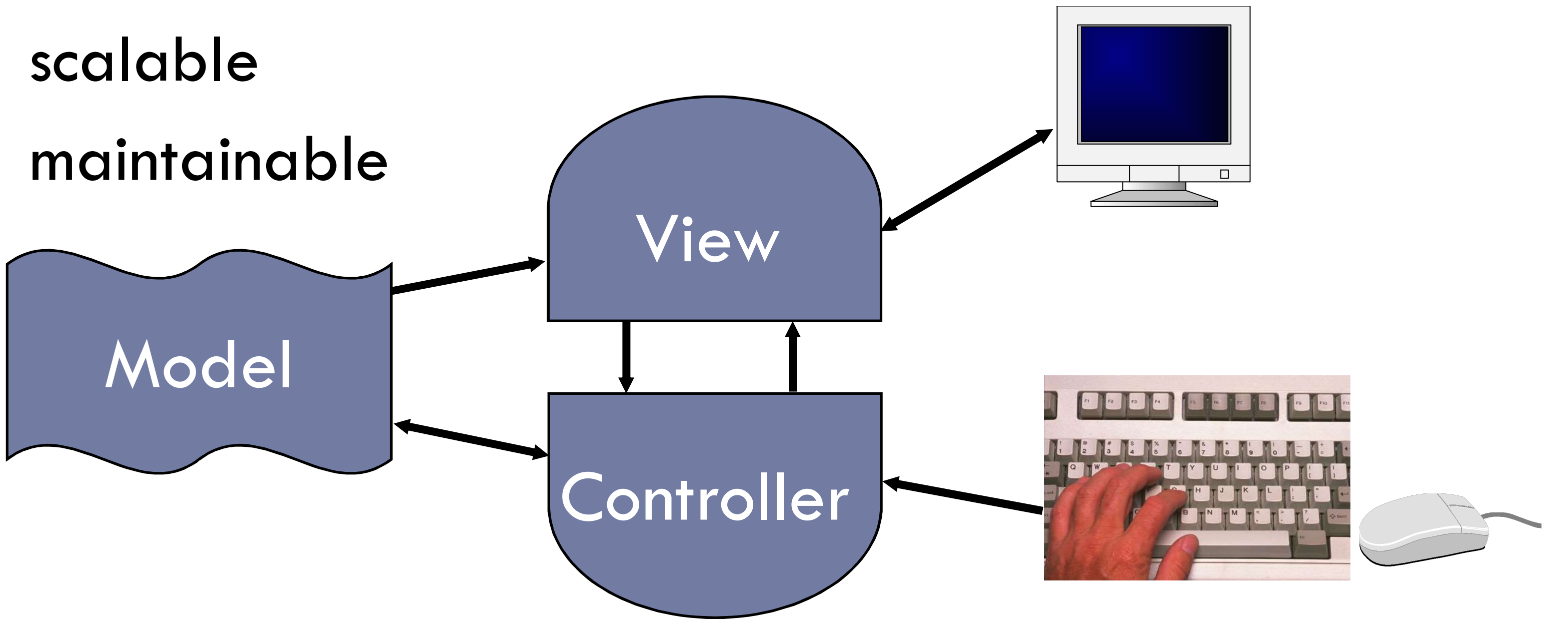
MVC ISSUES AND DETAILS

CMPT 381

Model-View-Controller

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



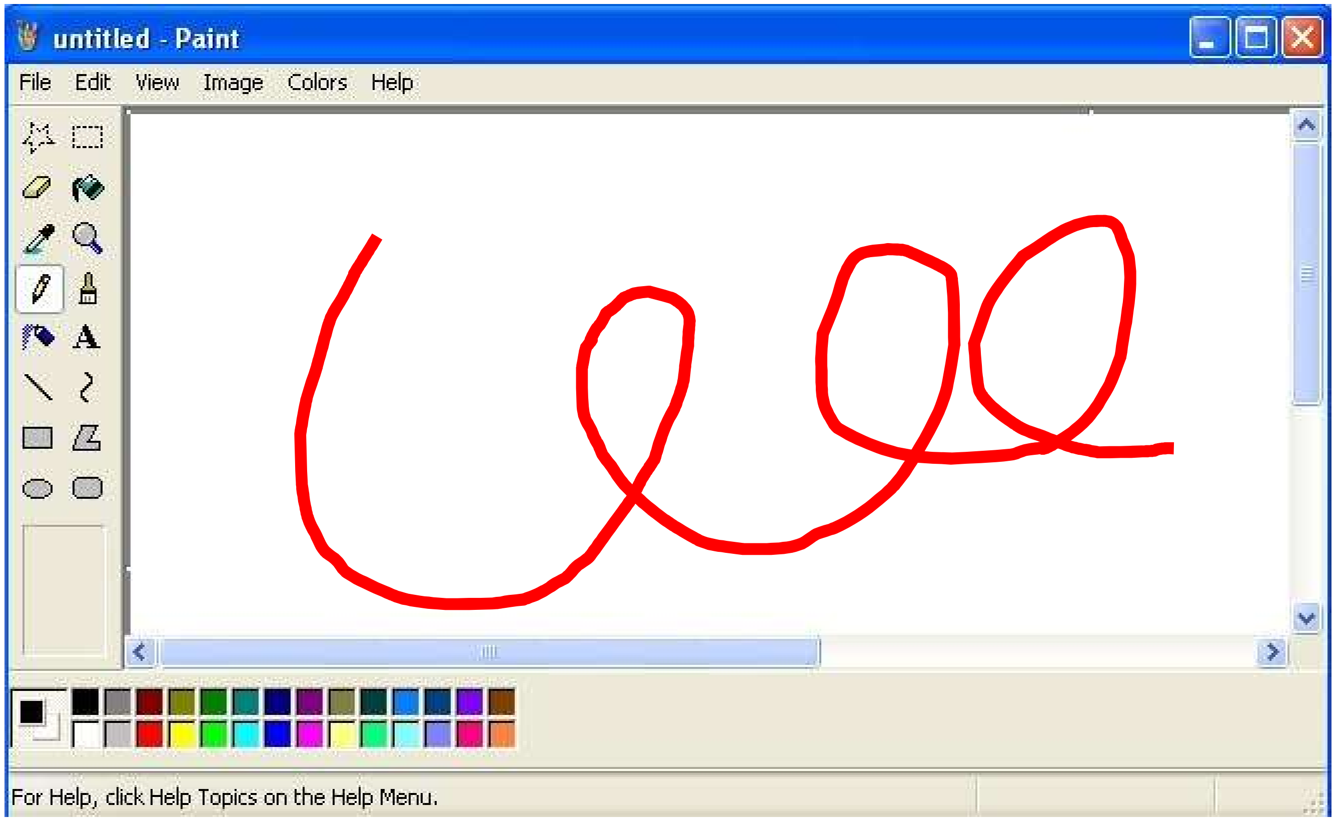
Overview

View state

Interaction models

Coordination between views

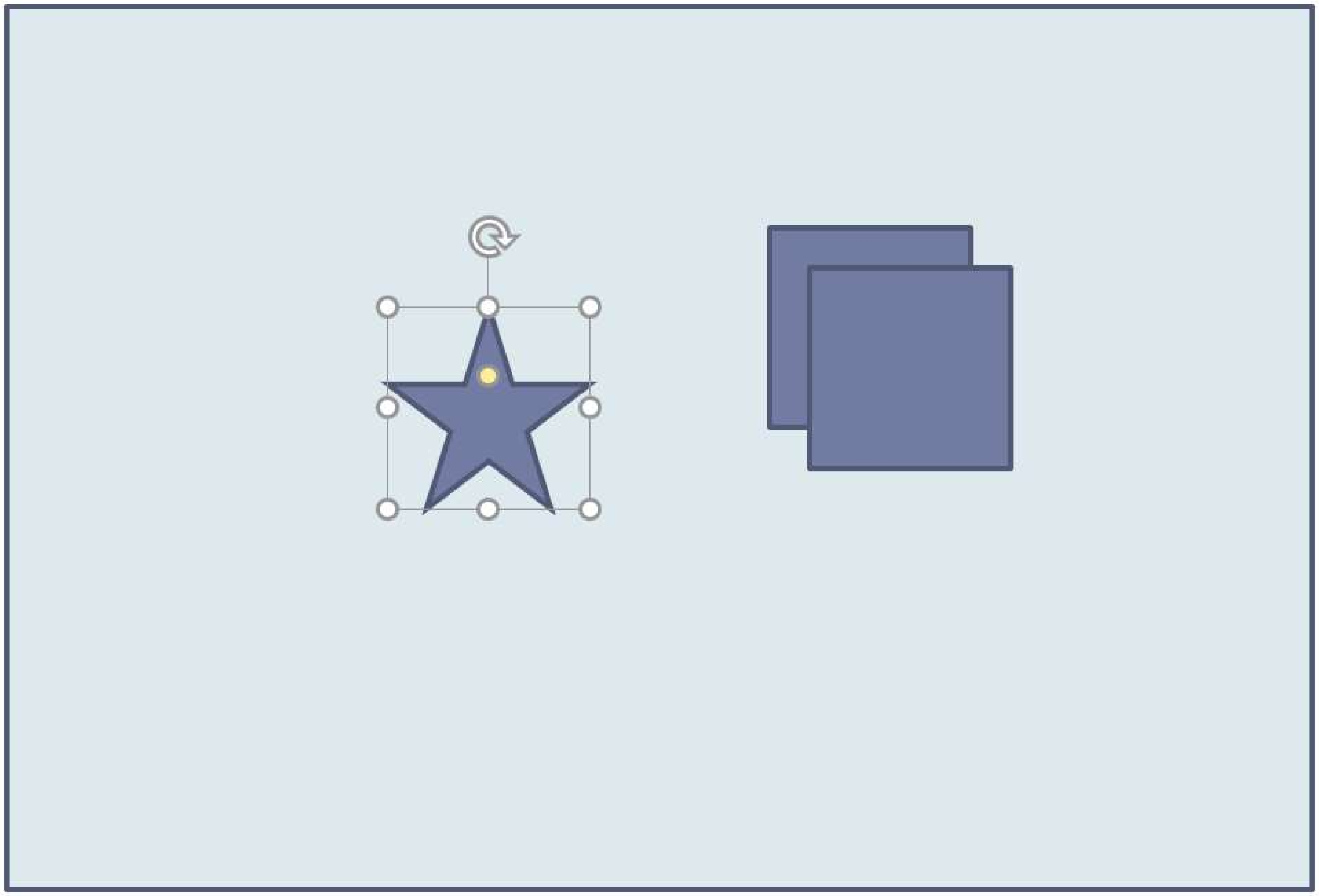
Storing view state



Storing view state

- Some UI actions are direct commands
  - Menu → "Clear All" → model.clearAll()
  - Nothing needs to be stored for these interactions
- Other actions are persistent
  - Involve changes to the state of the view
    - Selecting a current tool or colour
    - Changing the scroll region
  - Where should this information be stored?
    - Model? View? Controller?

Storing view state – selection



Who needs to know about view state?

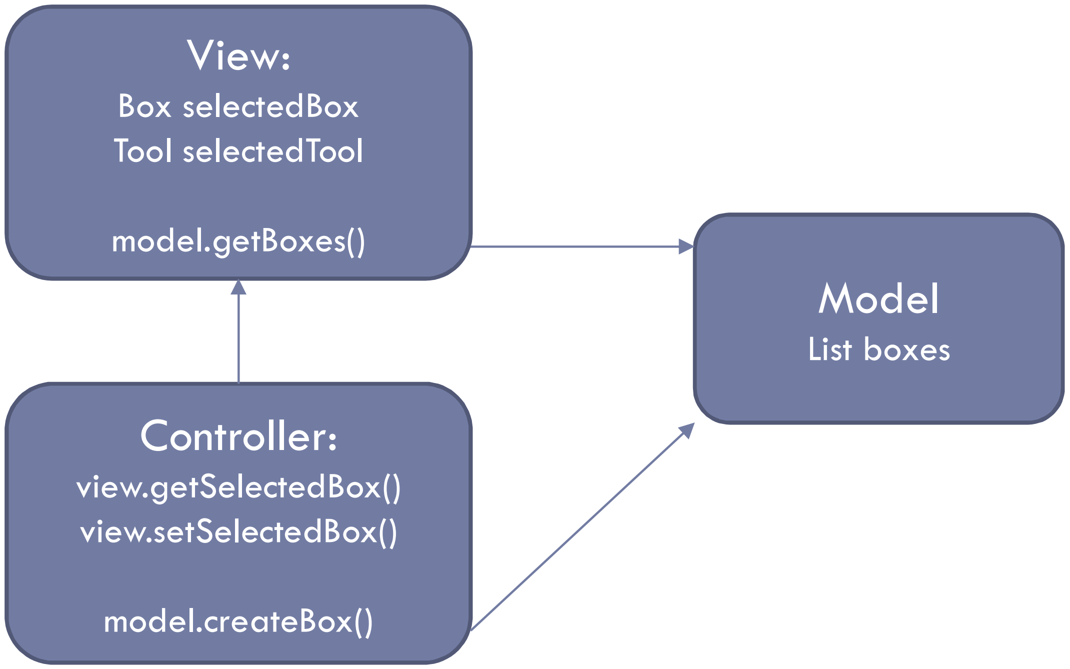
- Selection:
  - View displays selected object differently
  - Controller needs to know which object to act on
    - E.g., when delete key pressed, what gets deleted?
- Tool (mode):
  - View: specific cursor, highlighting in toolbar
  - Controller: what happens on action
    - E.g., when mouse pressed, draw or erase?
- Viewport
  - View: what region of the workspace to draw
  - Controller: handle scroll events

Where to store view state?

- If only a few pieces of information:
  - Store in View, make visible in Controller

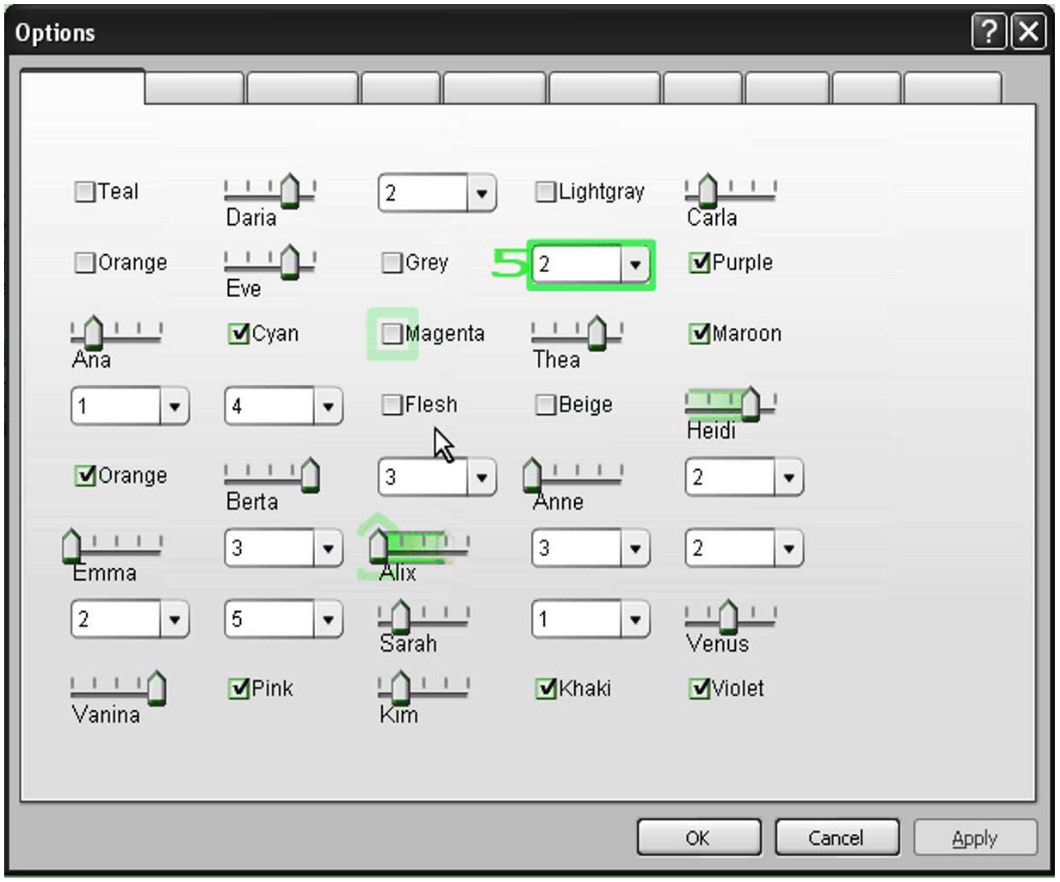
Where to store view state?

- If only a few pieces of information:
  - Store in View, make visible in Controller



More complex view state

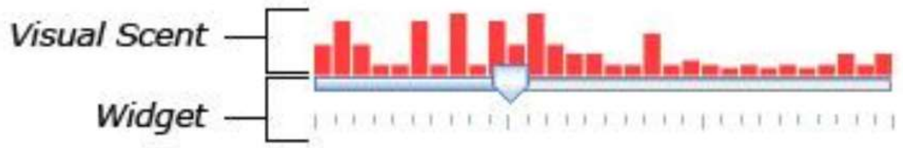
- Phosphor widgets (Baudisch)

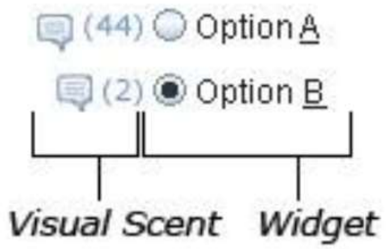


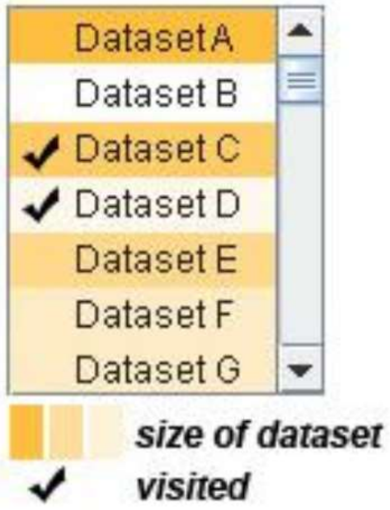
☐ Teal	Daria	2	☐ Lightgray	Carla
☐ Orange	Eve	☐ Grey	5 2	☑ Purple
Ana	☑ Cyan	☐ Magenta	Thea	☑ Maroon
1	4	☐ Flesh	☐ Beige	Heidi
☑ Orange	Berta	3	Anne	2
Emma	3	Alix	3	2
2	5	Sarah	1	Venus
Vanina	☑ Pink	Kim	☑ Khaki	☑ Violet

More complex view state

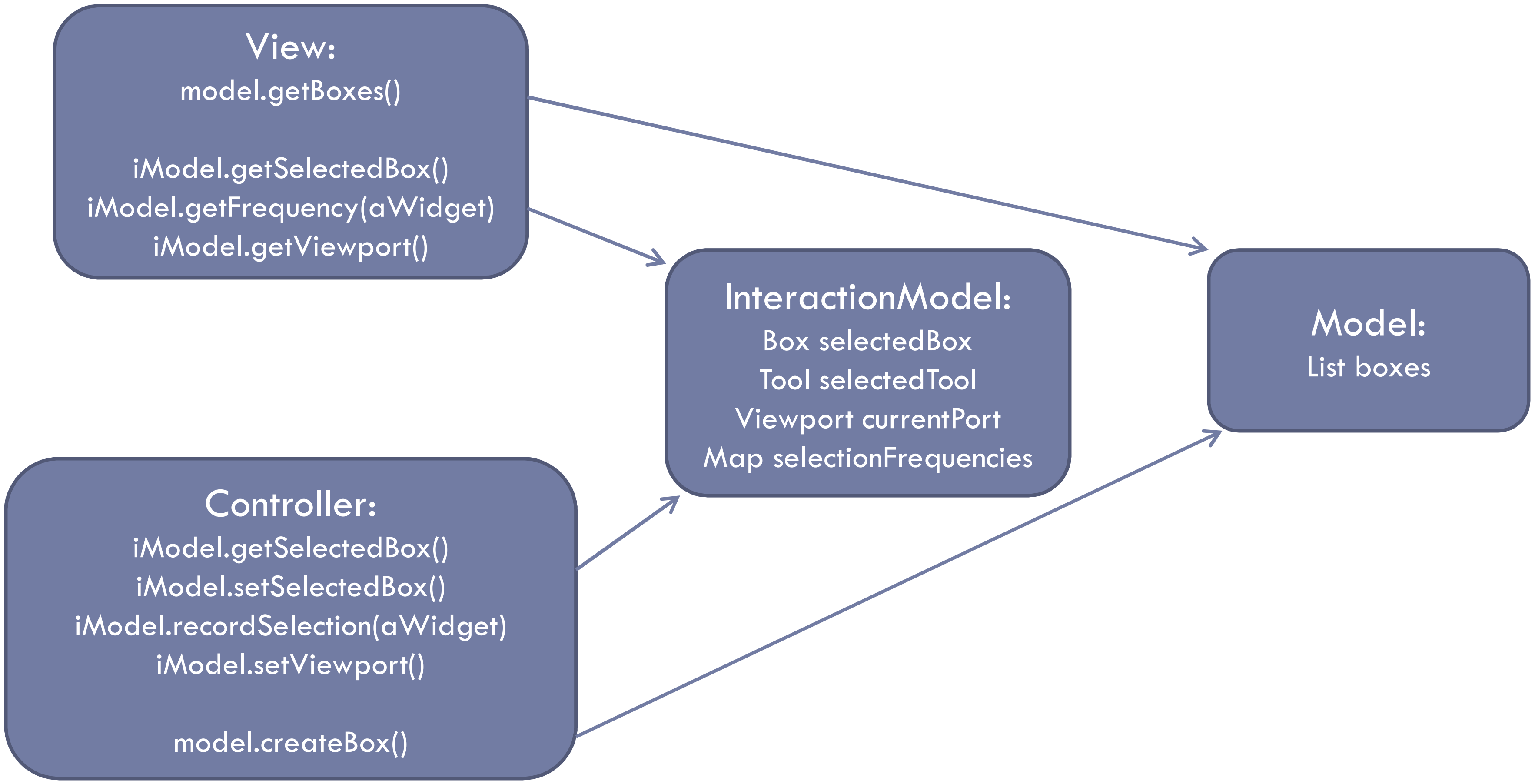
- "Scented widgets" (Willet, Heer, Agrawala)



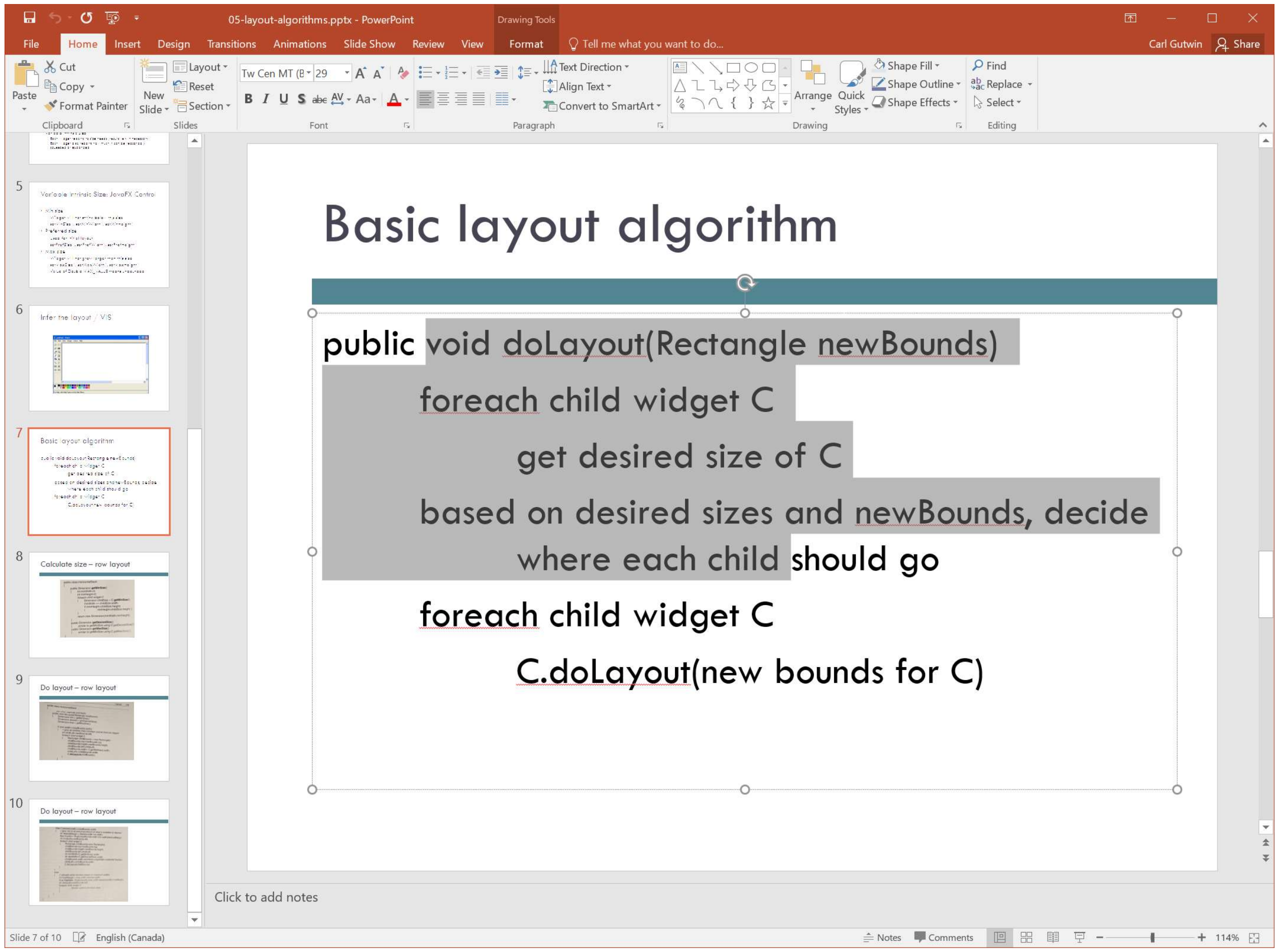




Where to store complex view state?



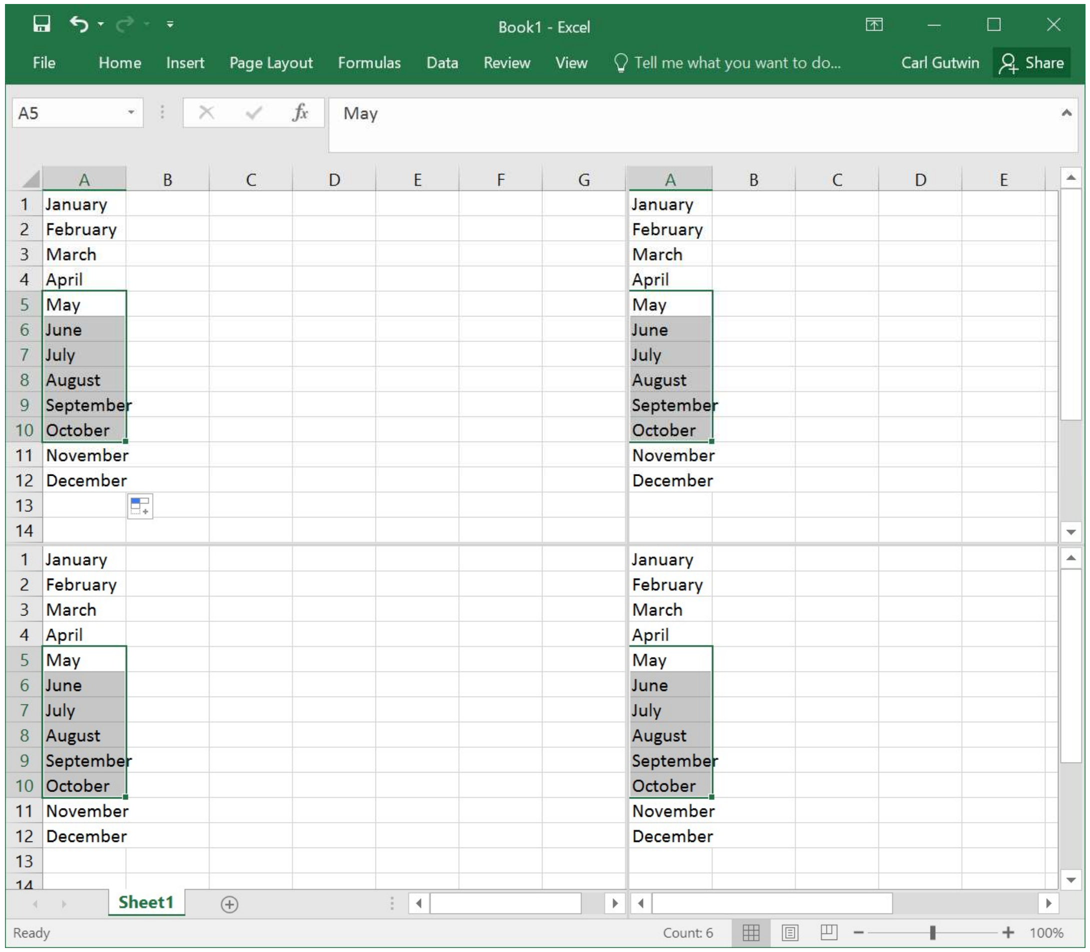
Synchronization between views



Basic layout algorithm

```
public void doLayout(Rectangle newBounds)
    foreach child widget C
        get desired size of C
    based on desired sizes and newBounds, decide
        where each child should go
    foreach child widget C
        C.doLayout(new bounds for C)
```

Synchronization between views

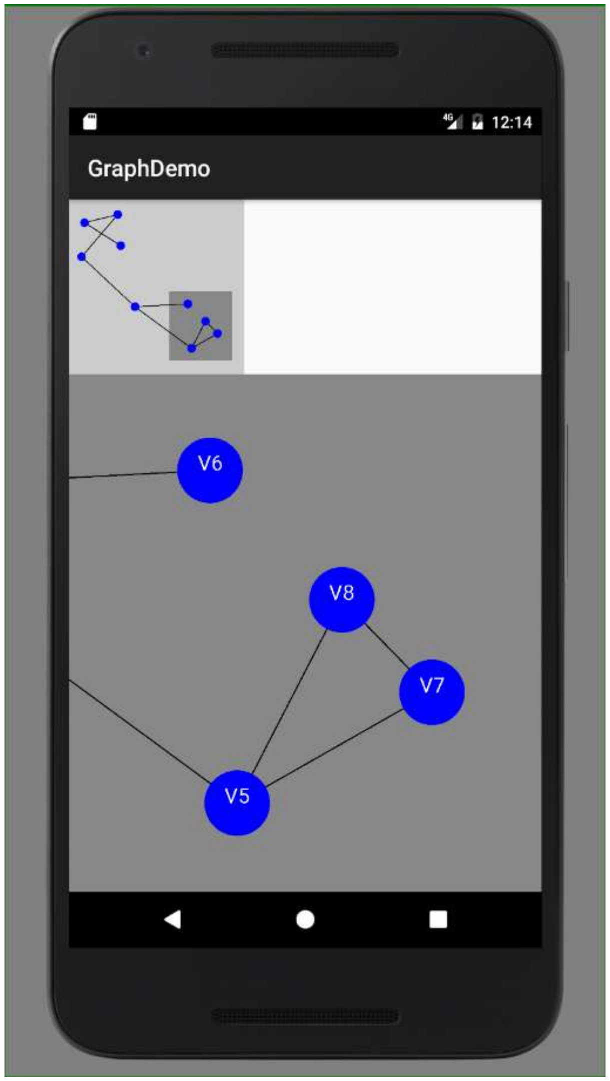


A5 | May

	A	B	C	D	E	F	G	A	B	C	D	E
1	January							January				
2	February							February				
3	March							March				
4	April							April				
5	May							May				
6	June							June				
7	July							July				
8	August							August				
9	September							September				
10	October							October				
11	November							November				
12	December							December				
13												
14												

1	January							January				
2	February							February				
3	March							March				
4	April							April				
5	May							May				
6	June							June				
7	July							July				
8	August							August				
9	September							September				
10	October							October				
11	November							November				
12	December							December				
13												
14												

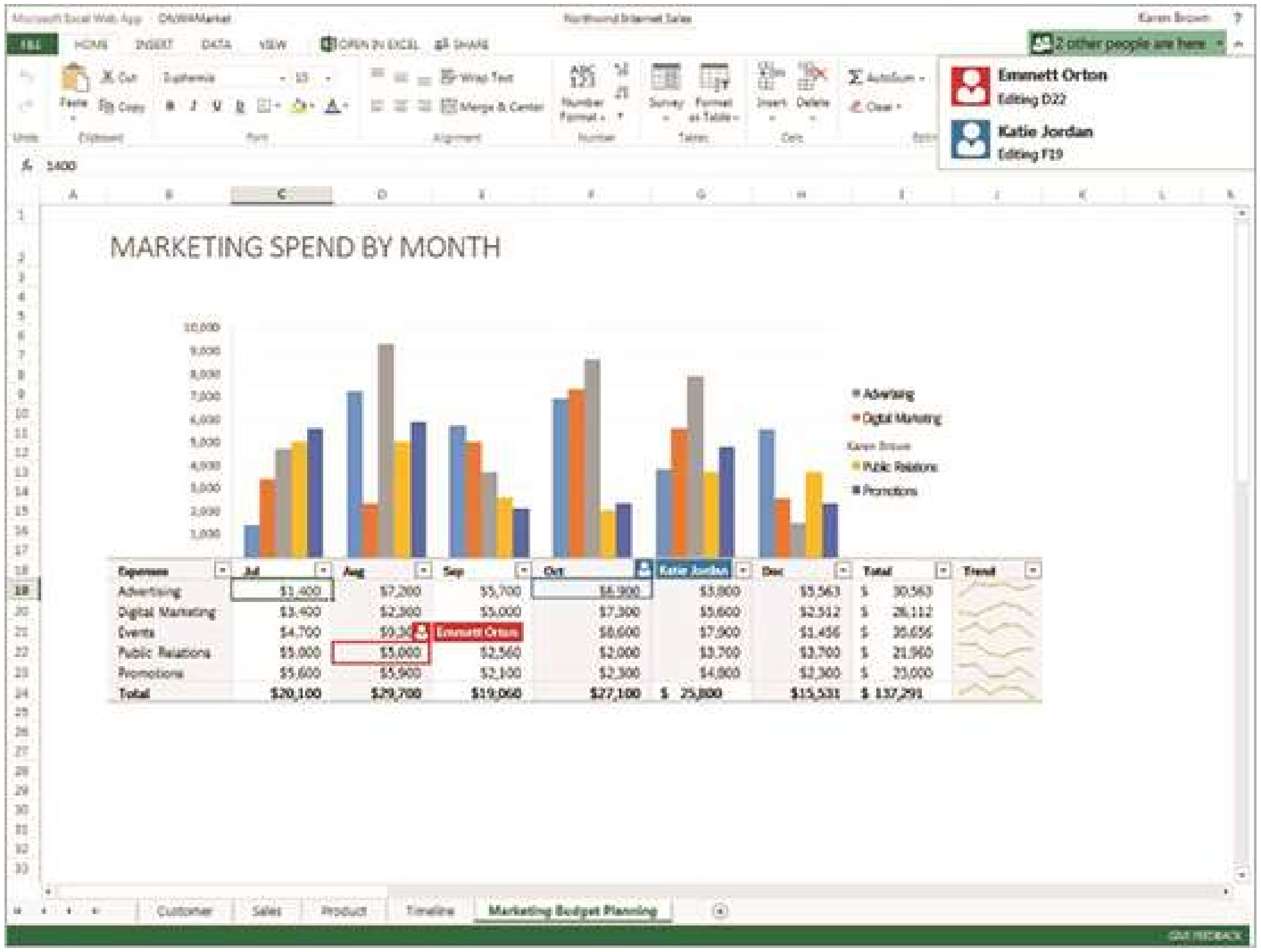
Synchronization between views



How to handle this?

- Store in View
  - Views need to inform other Views of selection changes
- Store in InteractionModel
  - Works well for selection
    - (assuming one selection across all views)
- What about View-specific information?
  - E.g., viewport, cursor location
  - Can register View with InteractionModel
  - Valuable for groupware applications

Multiple distributed views in groupware

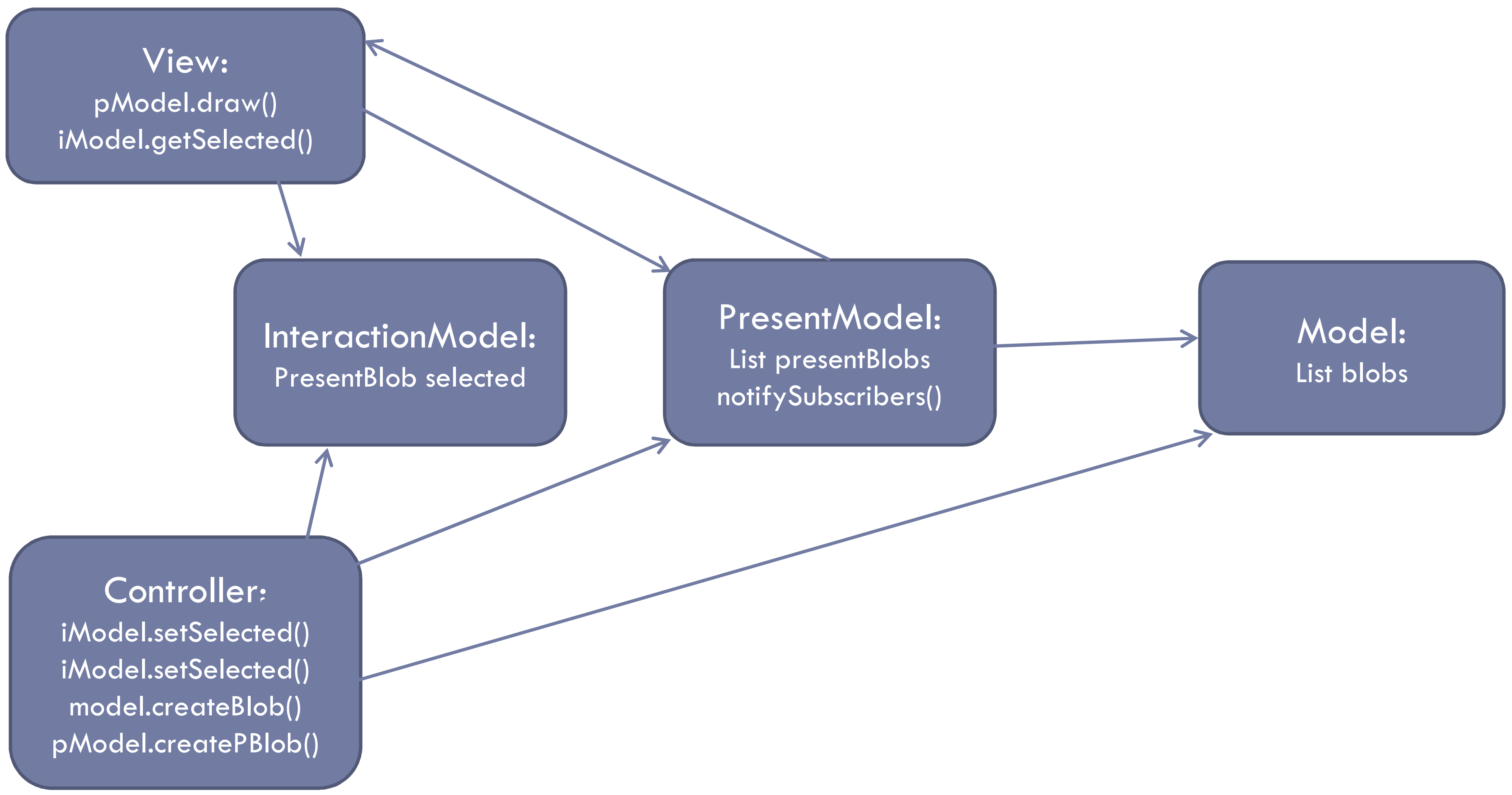


Expenses	Jul	Aug	Sep	Oct	Katie Jordan	Dec	Total	Trend
Advertising	$1,400	$7,200	$5,700	$6,900	$3,800	$5,563	$ 30,563	
Digital Marketing	$3,400	$2,300	$5,000	$7,300	$5,600	$2,512	$ 26,112	
Events	$4,700	$9,3		$8,600	$7,900	$1,456	$ 35,656	
Public Relations	$5,000	$5,000	$2,560	$2,000	$3,700	$3,700	$ 21,960	
Promotions	$5,600	$5,900	$2,100	$2,300	$4,800	$2,300	$ 23,000	
Total	$20,100	$29,700	$19,060	$27,100	$ 25,800	$15,531	$ 137,291	

Presentation Models

- In many apps we include some presentation information in the view
  - E.g., coordinates and radius for a vertex
- We may not want to (or be able to) do this
  - E.g., if we are using an external library for the model
  - E.g., if we want to maximize code separation
- Where to store the information we need to present the model in the View?

PresentationModel Architecture



Presentation Model

- The PM mirrors the model, and adds information needed to display the model
- Model (fully separated):
  - Contains a list of Vertex and a list of Edge
  - Class Vertex: contains a list of Edges
  - Class Edge: contains a start and end Vertex
- Presentation Model
  - Contains a list of PVertex and of PEdge
  - Class PVertex: contains a Vertex, x, y, radius, colour
  - Class PEdge: contains an Edge, width, colour