CS 160: Lecture 12

Professor John Canny
Spring 2003
March 12
Outline

- Model-view controller
- Why do we need it?
- Changing the display
- Event flow
- Dragging at interactive speeds
Model-View-Controller

Architecture for interactive apps
* introduced by Smalltalk developers at PARC

Partitions application in a way that is
* scalable
* maintainable
Example Application

Blue circles: 4
Cardinal squares: 2
Model

- Information the app is trying to manipulate
- Representation of real world objects
  - circuit for a CAD program
    - logic gates and wires connecting them
  - shapes in a drawing program
    - geometry and color
* Implements a visual display of the model
* May have multiple views
  * e.g., shape view and numerical view
Multiple Views

Blue circles: 4
Cardinal squares: 2
- Implements a visual display of the model
- May have multiple views
  * e.g., shape view and numerical view
- Any time the model is changed, each view must be notified so that it can change later
  * e.g., adding a new shape
Controller

- Receives all input events from the user
- Decides what they mean and what to do
  * communicates with view to determine which objects are being manipulated (e.g., selection)
  * calls model methods to make changes on objects
    + model makes change and notifies views to update
Controller

Blue circles: 4
Cardinal squares: 2
Controller

Blue circles: 4
Cardinal squares: 2
"pattern of behavior in response to user events (controller issues) is independent of visual geometry (view issues)"

Controller must contact view to interpret what user events mean (e.g., selection)
Combining View & Controller

- View and controller are tightly intertwined
  - lots of communication between the two
- Almost always occur in pairs
  - i.e., for each view, need a separate controller
- Many architectures combine into a single class
Why MVC?

- Combining MVC into one class or using global variables will not scale
  * model may have more than one view
    + each is different and needs update when model changes

- Separation eases maintenance
  * easy to add a new view later
  * new model info may be needed, but old views still work
  * can change a view later, e.g., draw shapes in 3-d (recall, view handles selection)
Adding Views Later

Blue circles: 4
Cardinal squares: 2
Changing the Display

How do we redraw when shape moves?
Moving Cardinal Square

Blue circles: 4
Cardinal squares: 2
Erase w/ Background Color and Redraw

Blue circles: 4
Cardinal squares: 2
Changing the Display

- **Erase and redraw**
  * using background color to erase fails
  * drawing shape in new position loses ordering

- **Move in model and then redraw view**
  * change position of shapes in model
  * model keeps shapes in a desired order
  * tell all views to redraw themselves in order
  * slow for large / complex drawings
    + flashing!
Damage / Redraw Method

- View informs windowing system of areas that need to be updated (i.e., damaged)
  * does not redraw them at this time...

- Windowing system
  * batches updates
  * clips them to visible portions of window

- Next time waiting for input
  * windowing system calls Redraw method for win.
    + passes region that needs to be updated
Damage old, Change position in model, Damage new

Blue circles: 4
Cardinal squares: 2
Event Flow

- Creating a new shape
Event Flow (cont.)

Assume blue circle selected
Event Flow (cont.)

Press mouse over tentative position
Windowing system identifies proper window for event
Controller for drawing area gets mouse click event
Checks mode and sees “circle”
Calls models AddCircle method with new position
Event Flow (cont.)

- **AddCircle** adds new circle to model’s list of objects
- **Model** then notifies list of views of change
  * drawing area view and text summary view
- **Views** notifies windowing system of damage
  * both views notify WS without making changes yet!
    + model may override
Views return to model, which returns to controller
Controller returns to event handler
Event handler notices damage requests pending and responds
If one of the views was obscured, it would be ignored
Event Flow (cont.)

- Event handler calls view's Redraw methods with damaged area
- Views redraw all objects in model that are in damaged area
Dragging at Interactive Speeds

- Damage old, move, damage new method may be too slow
  - must take less than 200 ms to be smooth

- Solutions
  - don’t draw object, draw an outline (cartoon)
    + use XOR to erase fast (problems w/ color)
    + save portion of frame buffer before dragging
Summary

- Model-view controller
- Motivation
- Display updates
- Event flow
- Dragging at interactive speeds