Programming: Model-View-Controller

Reading #5: "Chapter 5.1-5.8 Basic Interaction" by Dan Olsen, Developing User Interfaces, 1998, pp. 129-166.

Model-View-Controller

• A programming style for O-O programming
  – Developed in Smalltalk (circa 1980)
• Model
  – information state of the application
• View (Output)
  – visual display of model
• Controller (Input)
  – receives input events from user
• Event Propagation
  – Controller to Model, Model to View, Controller to View

Clock Example

MODEL
Time
Hrs (1-12)
Mins (0-59)
Secs (0-59)
AM (true, false)
Zone (PST, PDT)

VIEW

CONTROLLER

• Event Propagation
  – Controller to Model, Model to View, Controller to View
Model-View-Controller
Advantages

• Separation of Model from I/O allows scale up
  – May change View or Controller without changing Model
  – May have multiple models
  – May have multiple views
  – May have multiple controllers

Buttons

• Model
  – All buttons have a single model value that can be set to one or more discrete values
  • On/off (a)
  • Radio Buttons (b)
  • Checkbox (c)

Buttons cont.

• View
  – Maps mouse locations for controller
• Controller
  – Responds to single mouse click
**Scroll bars**

- **Model**
  - Manipulate continuous bounded range of values
  - Two limits and a current value between the limits
  - May include window width and page size
    - See a: left arrow moves one page left & page size may vary

**Scroll bars cont.**

- **View**
  - Maps mouse locations for controller
- **Controller**
  - Respond to single mouse clicks in left and right edges
  - Respond to mouse dragging of slider

**Menus**

- **Model**
  - Select from a potentially large number of possibilities, similar to button
  - Menu selection
    - Sets model variables to particular values
    - Issues one of model’s commands
- **View**
  - Maps mouse locations for controller
- **Controller**
  - Responds to single mouse click, drag and release
Text Entry Boxes

- Model
  - String of characters that can be typed from scratch or edited
  - Integers only, dollars, dates?
  - Error processing

MVC and Widgets

- Widgets combine View and Controller in one object
  - For most widget sets, for a given controller, there is only one view
  - Single view per controller leads to merging of view and controller
  - Exception: Product differentiation

Exception: Microsoft Toolbar

- Microsoft added this to the Mac menu bar to differentiate the Office products
- Within the toolbar, all buttons look and behave alike
- Simpler for Microsoft to implement its own button widgets rather than modify Mac code
MVC

• Object-oriented approach
  – Exploits use of abstract classes & message/method-binding mechanisms to do all event dispatching and change notification
  – Makes it easy to handle variety of messages and notifications that must pass between the objects that make up the interactive architecture

MVC

• OO Drawbacks
  – Every model must have its own abstract view class
  – Every view must inherit both the method interface necessary to receive input from the windowing systems and the method interface needed to receive change notification
  – More complicated when views contain other views and must communicate with them

MVC

• Alternative implementation to OO
  – Views have only one Event method that handles all input events, system notification events, model change notification, etc.
  – When object receives an event, its Event method looks at the type of event to sort out what should be done
  – Eliminates need for abstract view class because any can receive the Event message
  – Much more flexible than OO but programmer must do type checking and event management rather than compiler