Stage 6: video 2

Scene 1: Screencast

The first place a Cocoa programmer encounters MVC is in the structure of the application when a new project is created in XCode. Let's do that now.

- We'll create a new XCode iOS project and select the "single-view" template
- It will be called "MVCexample" and we'll check the box to use core-data
- With the exception of the AppDelegate modules, all the project components are already members of the Model, the View or the Controller.
- MVCexample.xcdatamodeld is, as the name suggests, is a data model
- The storyboard is part of the View
- and the ViewController modules are part of the controller
- The AppDelegate modules also arguably fit into the controller sub-system.

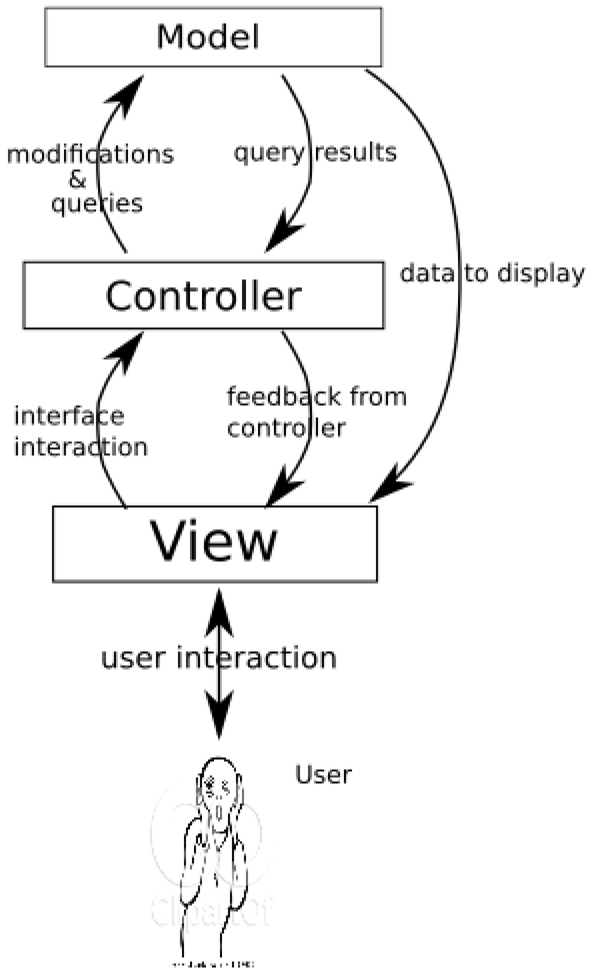
Going into each of these subsystems in a little more detail…
For the model - XCode knows nothing about the core functionality of your app, so unless the core-data box is checked, a new project will have no model components. Persistence is a common application need, and core-data is a general enough solution that Xcode makes it an option for new projects.

As for the view: The evolution of interface-builder(or IB) in Apple's development environment has been a process of solidifying the limits of how much “smarts” goes into the view. The xml representation specified in IB for either a nib or a storyboard is highly constrained. While the nib and storyboard objects are instantiated at runtime, Xcode assumes no need for logic beyond low-level event handling that is routed directly to the controller subsystem.

The skeleton of the controller is laid out in the form of the UIViewController subclass modules. Looking through the methods provided by the UIViewController class creates a clear picture of it’s expected functionality relative to the view. See the teacher’s notes for more details.

Scene 2: On-Set

Now we’ll look at MVC in a bit more detail.
<MOTION>
The relationships shown are not always present and certainly not required, but they are the most common ones between the three subsystems.



The majority of the interaction is between the model and the controller, or between the controller and the view. Since the controller is described as the “glue” or “insulation” between the other two components, this makes sense.

One relationship that might seem surprising is the “data to display” from the model directly to the view; in some cases it’s useful to provide model state directly to the view so that it can be displayed

verbatim. We’ll see an example of this when we look at UITableView. Note that this relationship is directional; user interaction generally should *not* be affecting model state without some processing by the controller.
</MOTION>

A common example of the MVC system at work within the Cocoa-Touch frameworks is UITableView. By setting the “delegate” to an class instance that conforms to the *UITableViewDelegate* protocol and “dataSource” to one that conforms to *UITableViewDataSource*, the responsibilities for supporting the tableview are neatly divided into *model, view* and *controller* functions.
<SLIDE>
The data source takes over responding to methods such as:

- -cellForRowAtIndexPath
- -numberOfSections
- -numberOfRowsInSection

While the delegate assumes responsibilities for interactions, such:

- -didSelectRowAtIndexPath
- -willBeginEditingRowAtIndexPath

And the UITableView instance retains responsibilities related to the appearance of the table.

Unsurprisingly, the UITableViewController class conforms to the *UITableViewDelegate* protocol automatically.

Questions:

1. In what format are Interface Builder files (e.g. nibs, storyboards) stored? (ans. XML)
2. When Core Data is included in an application, in which MVC component does it usually reside? (ans. Model)
3. When the Model subsystem communications with the View subsystem, the communication should be only one way, which is? (ans. Model->View)
    a. View -> Model
    b. Model -> View
4. The most autonomous one of the three subsystems should be? (ans. Model)
5. True or False: Model-View-Controller is only a useful design pattern in Objective-C or Swift? (ans. false)