Model-View-Controller

Dr. Sarah Abraham
MVC

- Pattern of development to modularize features and design
- Objects have one of three roles:
  - Model
  - Viewer
  - Controller
- Object types separated by abstract boundaries and communicate across these boundaries
MVC Benefits

- Objects more reusable
- Interfaces better defined
- Applications more extensible
- Common pattern for interactive applications with GUI (graphical user interface)
MVC and Cocoa

- Cocoa designed around MVC model
- Good understanding of MVC leads to good design for Cocoa applications
- Custom objects in Cocoa applications must follow one of the MVC roles
MVC Pattern Flow

How does the application receive and respond to this flow of events?
Event-driven Programming

- Events are triggered occurrences that the program receives and can respond to.
- Event-driven programming allows for efficient handling of:
  - Device input
  - Timers
  - Event loops
- Events determine flow of the program based on user input, sensor output, or messages from other programs.
Model Layer

• Defines logic and computation of the program

• Model objects encapsulate data specific to the application
  • Contains data loaded into app
  • Handles state of persistent data within the app

• Avoids explicit connection to view objects
  • No concerns about user-interface or presentation
  • Does not directly respond to user-input
Model Layer Communication

- User interfaces with view layer
  - Changes communicated via controller object to model layer
  - Based on event info, model object updates

- Backend database updates model object
  - Changes communicated via controller object to view layer
  - Based on event info, view objects update
View Layer

- Displays data from model objects to allow user to interact and modify this information
- View objects that are visible to the user
  - Draw themselves on the screen
  - Respond to user input
- UIKit and AppKit frameworks provide collections of view classes
- Interface Builder provides many view objects for building app GUI
View Layer Communication

- Controller objects notify view object about changes to model data
- User-initiated changes (buttons pressed, text-fields entered) passed from view layer to model layer via controller objects
Controller Layer

- Intermediary between one or more view objects and one or more model objects
- Conduits that allow view objects to learn about changes in model objects and vice versa
- Perform setup and coordinating tasks for an application
- Manage the life-cycles of other objects
Controller Layer Communication

- Interprets user actions made in view objects and communicates changes or new information to model layer

- Notified about changes to model objects and communicates new or updated data to the view objects for display
Using MVC with iOS

- iOS frameworks provide 2 of 3 MVC components:
  - View Controllers
  - Views
- Model component custom-defined based on application purpose
- Views customized based on desired user-interface
- View Controllers customized based on required communication between models and views
Quiz Question!

Which component of the MVC pattern is not provided via frameworks in iOS development?

A. Model layer
B. View layer
C. Controller layer
D. None are provided
Creating a View-based Application

- Select Single View iOS Application:
Enter/select project options:

Product Name: ViewAppDemo
Team: Sarah Abraham (Personal Team)
Organization Name: Sarah Abraham
Organization Identifier: cs329e
Bundle Identifier: cs329e.ViewAppDemo
Language: Swift

- Use Core Data
- Include Unit Tests
- Include UI Tests
Project is ready for prototyping!