

Announcements

- **Lab 1 due on Wednesday by 11:59 PM**
 - Email it to cse438ta@gmail.com
 - All labs must run on the iPhone X_R
 - Previously I mistakenly mentioned they needed to run on the iPhone X_S
- **We will hold Studio 2 on Wednesday**
 - Along with additional labs on Thursday

Today's Topics

- **Views Introduction**
- **Auto Layout Demo**
- **MVC**
- **Lab 2**
- **MVC Demo**

Views

View Fundamentals

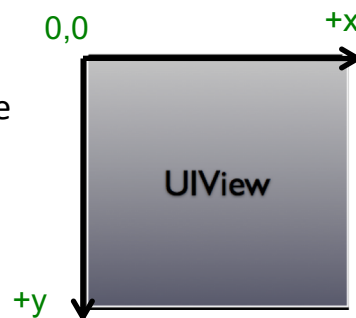
- Rectangular area on screen
- Draws content
- Handles events
- Subclass of UIResponder (event handling class)
- Views arranged hierarchically
 - every view has one **superview**
 - every view has zero or more **subviews**

View Hierarchy - UIWindow

- Views live inside of a window
- UIWindow is actually just a view
 - adds some additional functionality specific to top level view
- One UIWindow for an iOS app
 - Contains the entire view hierarchy
 - Set up by default in Xcode template project

UIView Coordinate System

- Origin in upper left corner
- y axis grows downwards
- Units are points, not pixels
 - Points are units of coordinate system
 - Pixels are min size unit of drawing
 - Typically 2 pixels per point
 - `var ContentScaleFactor`



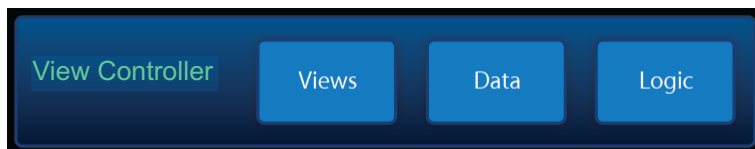
Lab 2 Preview

Auto Layout Demo

View Controllers

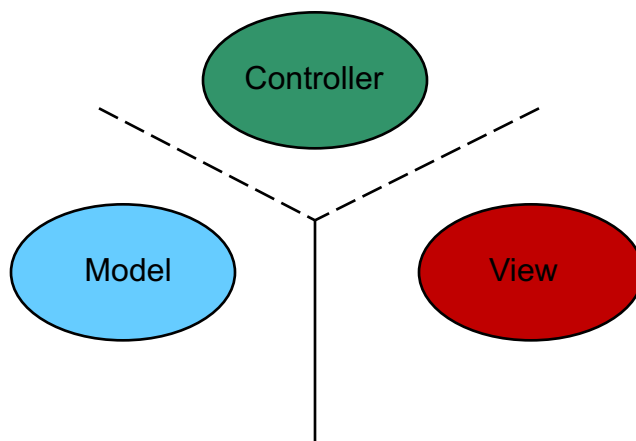
UIViewController

- **Basic building block**
- **Manages a screenful of content**
- **Subclass to add your application logic**



Model, View, Controller

Model, View, Controller



Model

- **Manages the app data and state**
- **Not concerned with UI or presentation**
- **Often persists somewhere**
- **Same model should be reusable, unchanged in different interfaces**

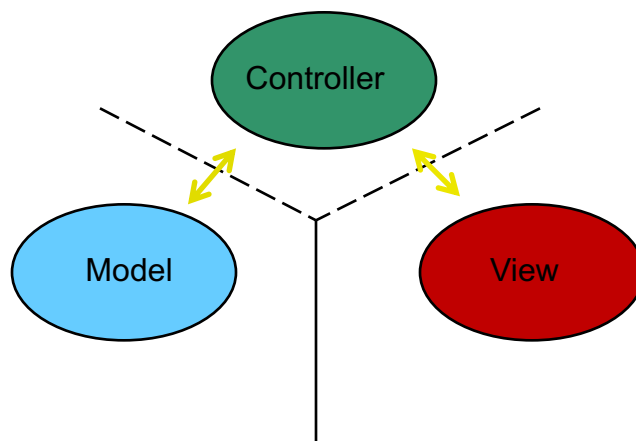
View

- **Present the Model to the user in an appropriate interface**
- **Allows user to manipulate data**
- **Does not store any data**
 - (except to cache state)
- **Easily reusable & configurable to display different data**

Controller

- Intermediary between Model & View
- Updates the view when the model changes
- Updates the model when the user manipulates the view
- Typically where the app logic lives

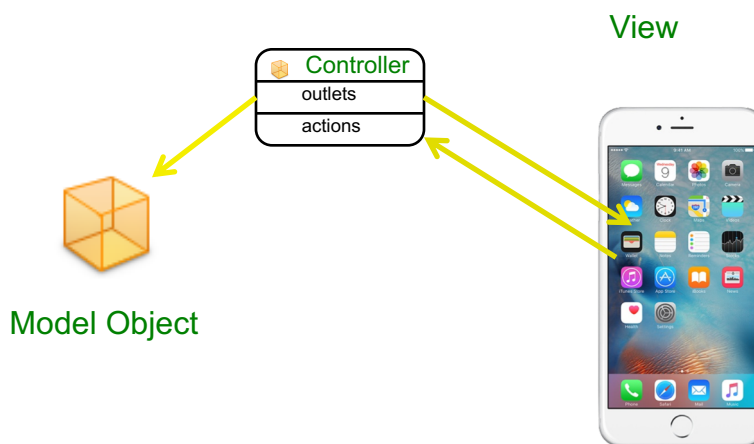
Model, View, Controller



Why Model-View-Controller?

- Separating responsibilities also leads to reusability
- By minimizing dependencies, you can take a model or view class you've already written and use it elsewhere
- Think of ways to write fewer lines of code

Model, View, Controller



MVC Demo