



iOS
DeCal

lecture 1

MVC, Swift & Xcode

cs198-001 : fall 2017

today's lecture

- logistics
- staff introduction
- more swift
- Xcode Overview
- MVC

logistic clarifications

enrollments

- everyone *should* be enrolled in cs198-001
 - let us know after class if you haven't

attendance policy

- NP if you have 4+ unexcused absences (lecture and lab)
- excused absences for midterm conflicts only
 - make a private piazza post *before* lecture with the class name and time of your midterm

lab sections

- sign up for lab sections after lecture
 - 310 Soda - Chris
 - 320 Soda - Nithi
 - 210 Wheeler - Paige (for students new to practical programming)
- 7-8:30pm on Wednesdays

staff introductions

course staff



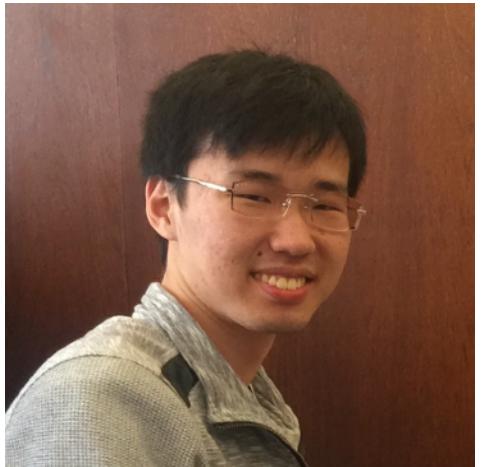
Paige Plander
instructor



Chris Zielinski
instructor



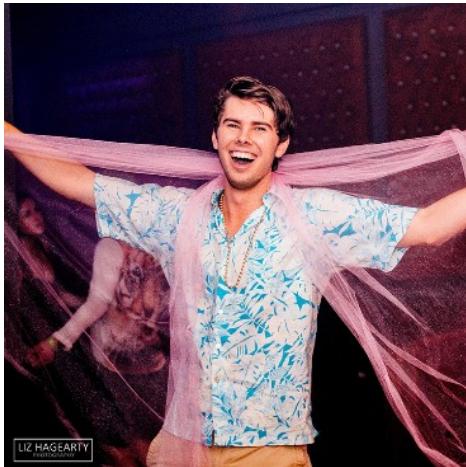
Nithi Narayanan
head TA



Jim



Sarah



Jack



Aspen



Victor



Daniel



Gokul



Marisa



Alex



Teddy

MVC

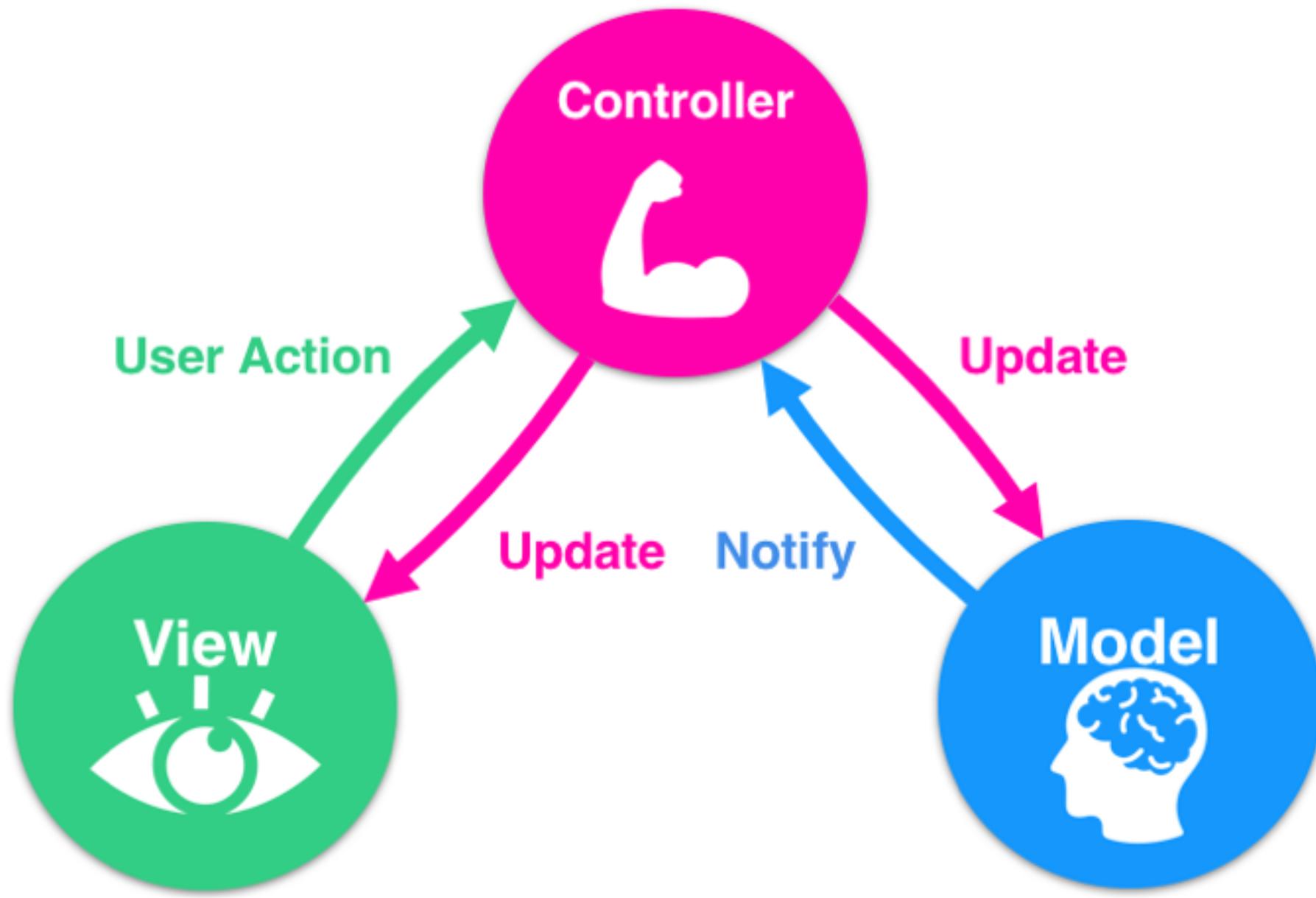
design patterns

- software design patterns are reusable solutions to common problems in software design
- **main idea:** assign objects in your application distinct roles using well-defined patterns and object relationships
 - There are many different types of patterns (not just basic MVC)!

model view controller (MVC)

assigns objects in your application one of the following distinct roles:

- **model** - encapsulates data and defines logic / computations
- **view** - what the users see and interact with
- **controller** - intermediary between models and views



Model View Controller

Example

User taps a button

Controller



View



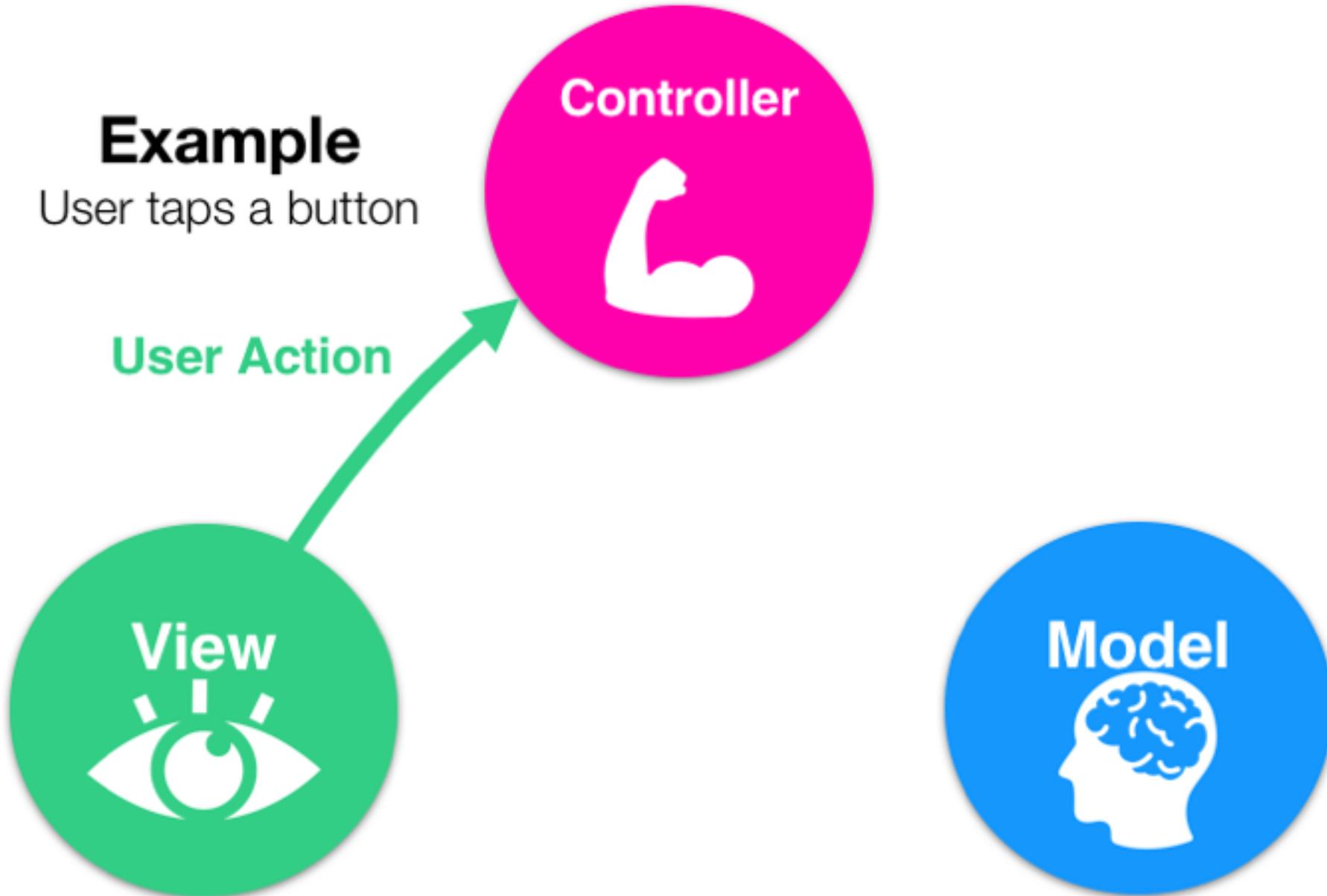
Model



User interacts with **View** (eg user taps “=” button in calculator)

Example

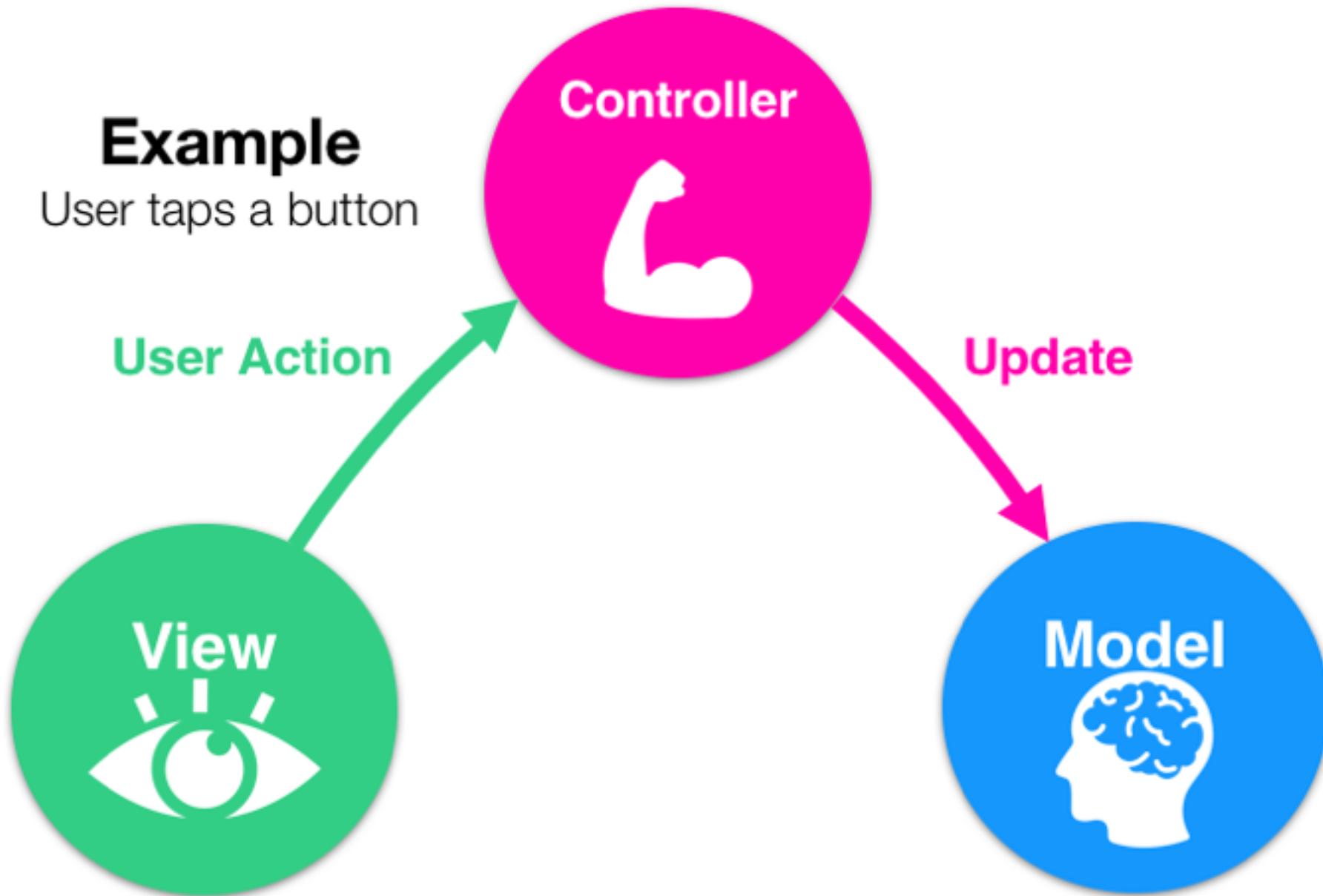
User taps a button



Controller is notified that the user has made an action

Example

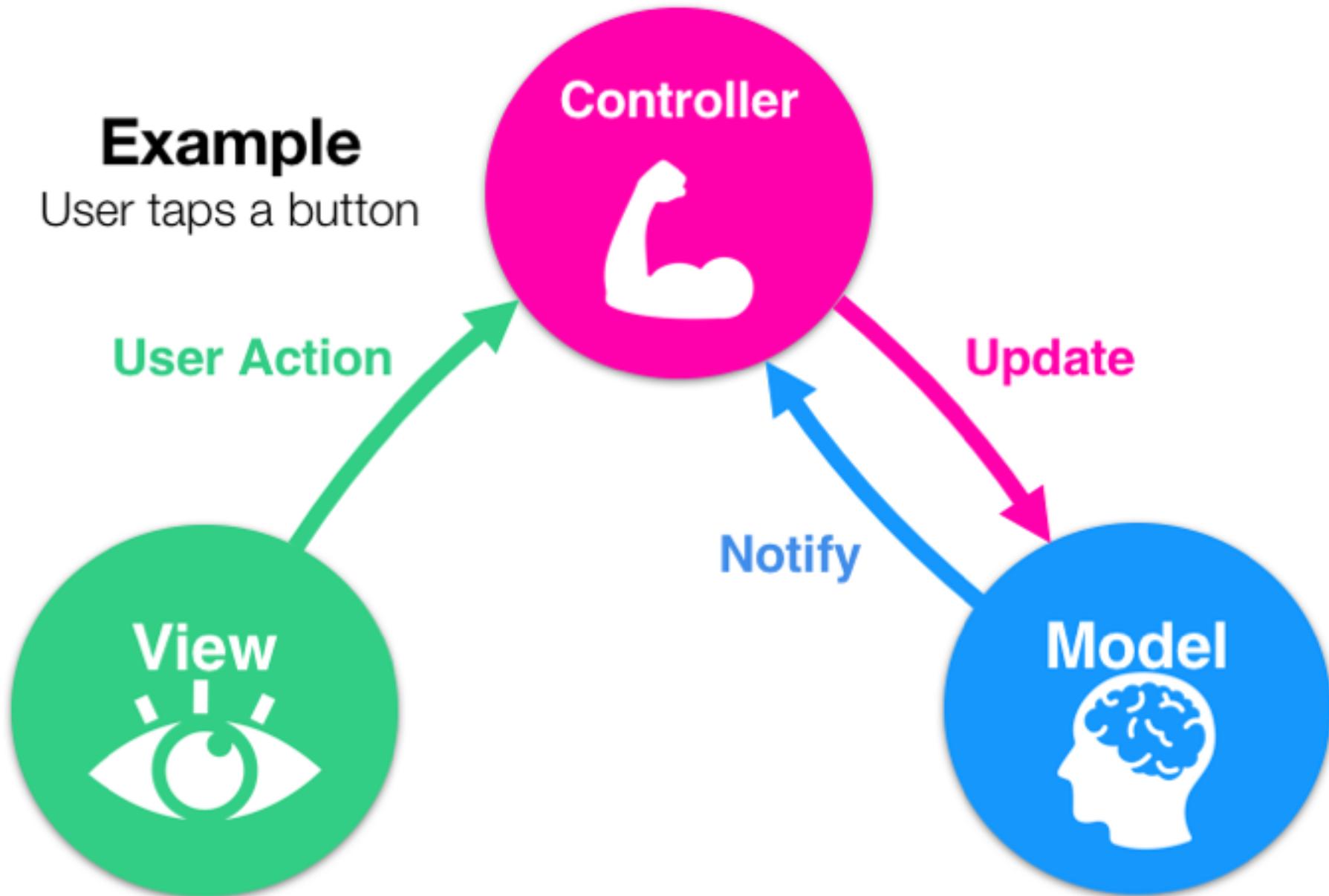
User taps a button



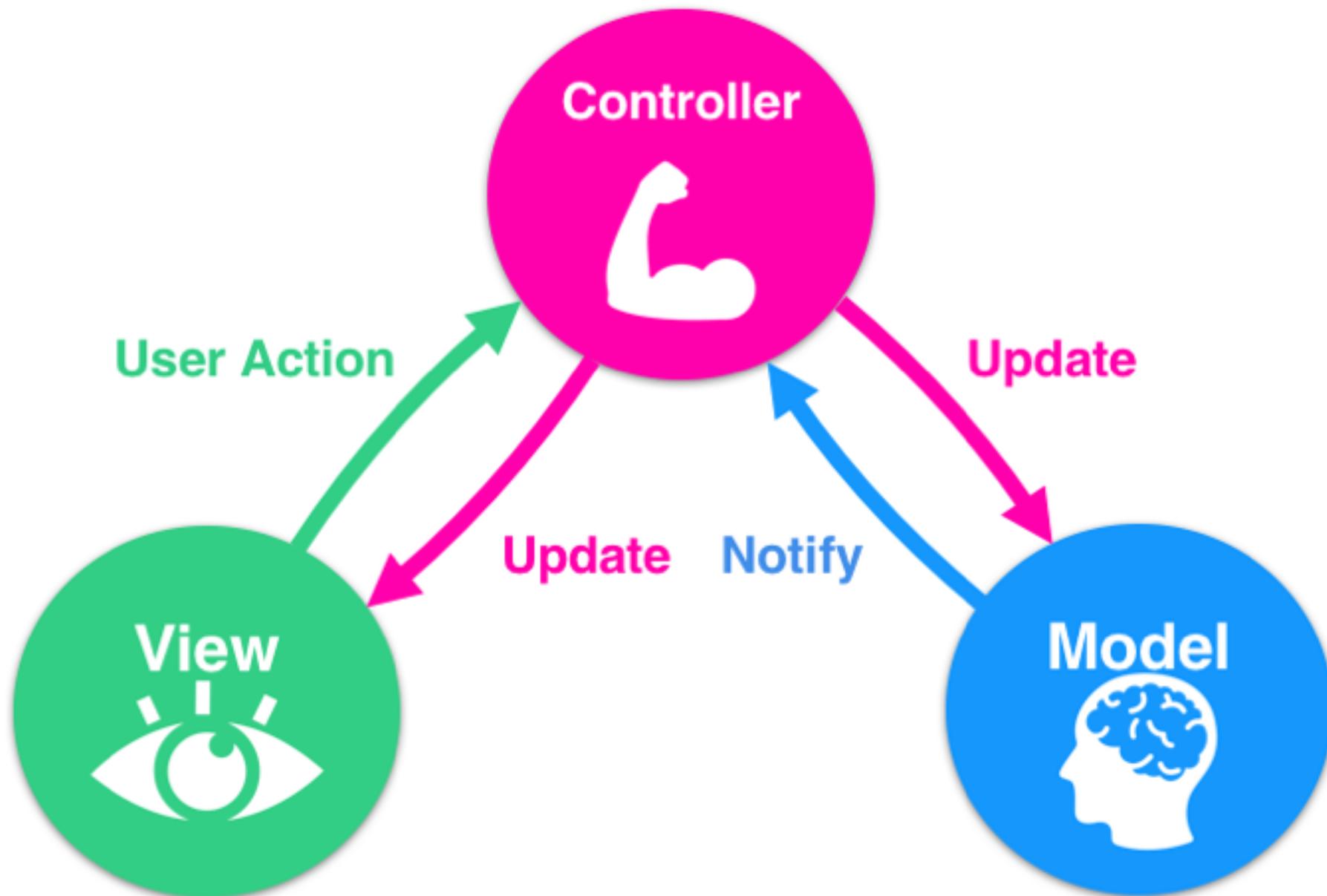
Controller updates the **Model** to reflect the users change

Example

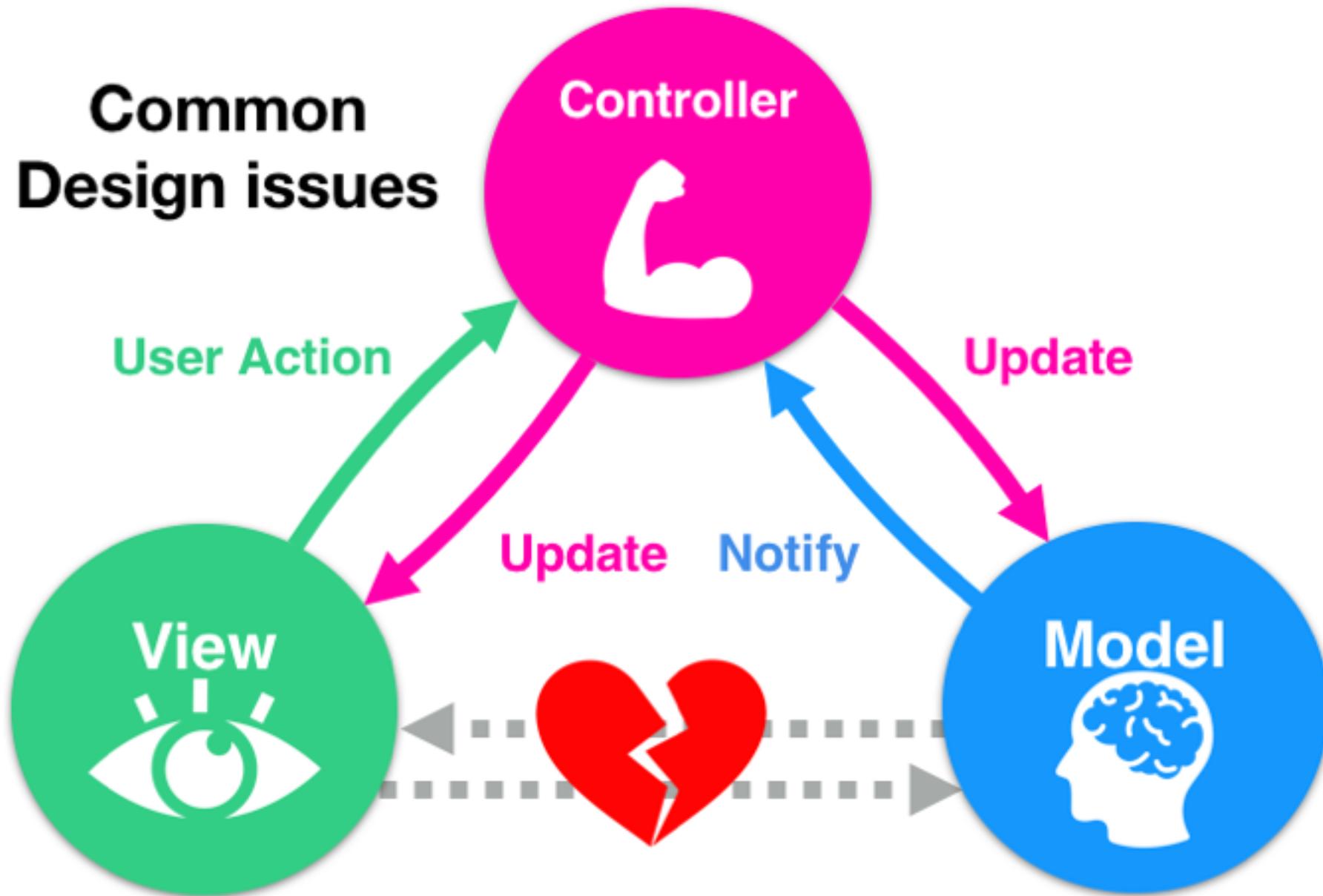
User taps a button



Model stores info / calculates then notifies controller of result

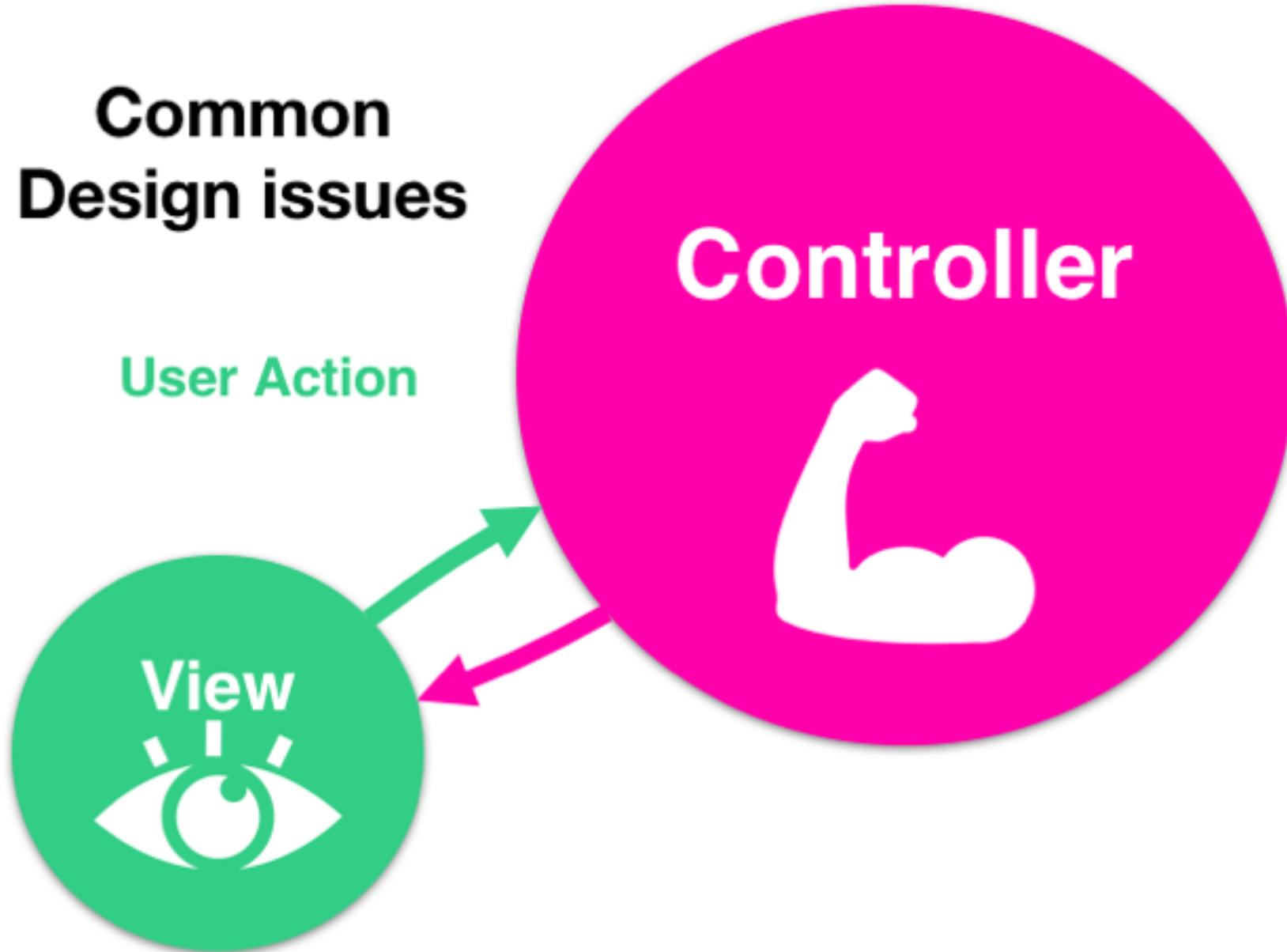


Common Design issues



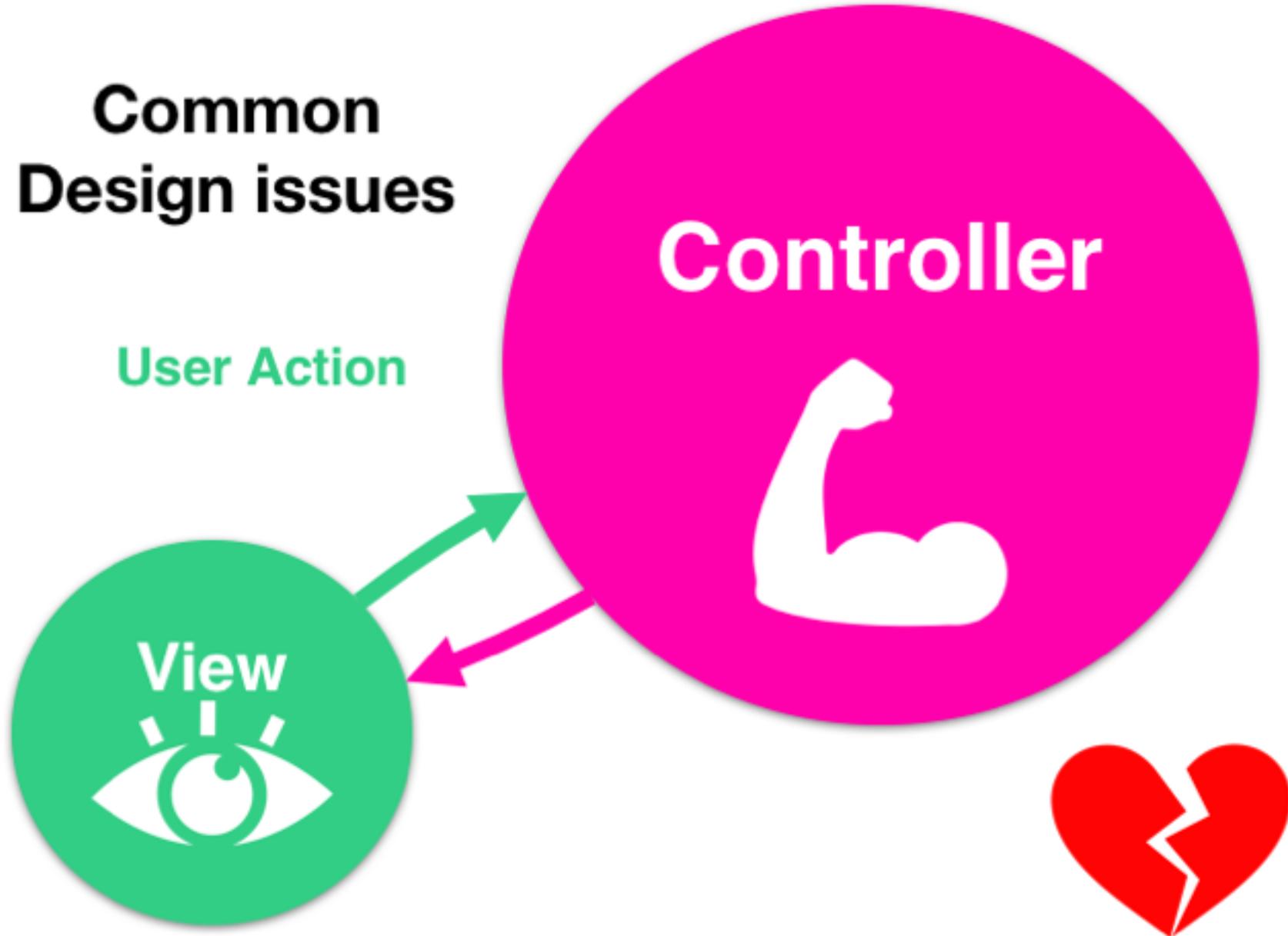
Issue #1 : Breaking Model / View Abstraction

Common Design issues



Issue #2 : Bloated Controllers

Common Design issues



Issue #2 : Bloated Controllers

MVC in Xcode

model

**data, logic,
computation
abstract classes**

To Create
**New > File > Swift
File**



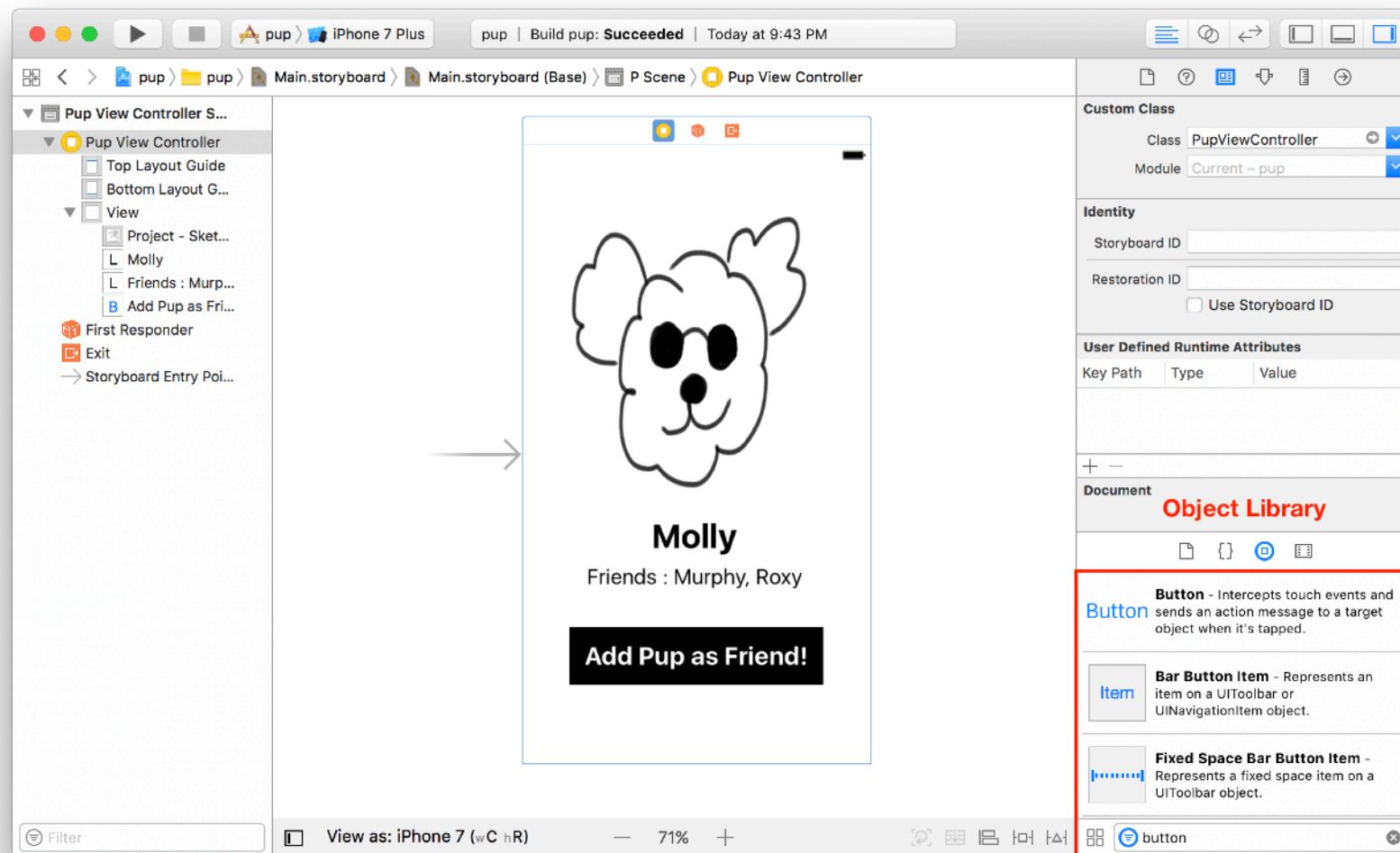
Puppy.swift — Edited

Puppy.swift > No Selection

```
1 // Puppy.swift
2
3 import Foundation
4
5 /// Model Class for Pups
6 class Puppy {
7
8     var name: String
9     var friendList: [Puppy]?
10    let location: Location?
11    var superPower: SuperPower?
12
13    init(pupName: String) {
14        name = pupName
15    }
16
17    func bark(direction: Direction) {
18        // ...
19    }
20
21    func makeNewFriend(withOtherPup pup: Puppy) {
22        // ...
23    }
24
25    func activateSuperPower(superPower: SuperPower?) {
26        // ...
27    }
28
29 }
```

Swift File

views: interface builder



views: programmatic



Cocoa Touch
Class

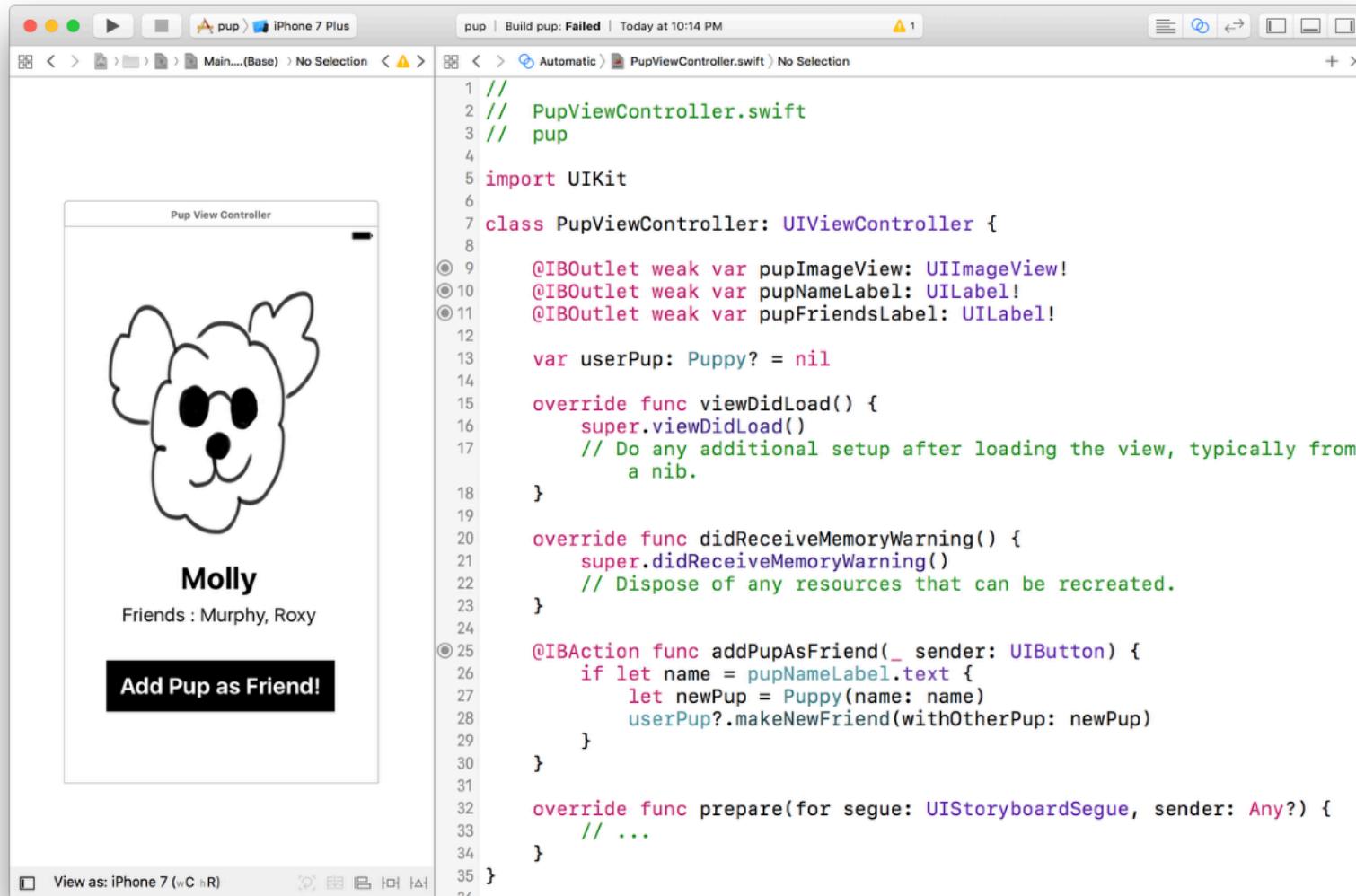
**creating views
programmatically**

New > File > Cocoa
Touch Class.

subclass an existing
type of View

```
1 //  PupView.swift
2
3 import UIKit
4
5 class PupView: UIView {
6
7     struct Constants {
8         static let marginSize: CGFloat = 8
9         static let buttonText: String = "Add Pup as Friend!"
10    }
11
12    let pupView = UIImageView(image: UIImage(named: "pup"))
13    let friendsListLabel = UILabel()
14
15    var addPupFriendButton: UIButton = {
16        var button = UIButton()
17        button.setTitle(Constants.buttonText, for: .normal)
18        return button
19    }()
20
21
22    override init(frame: CGRect) {
23        super.init(frame: CGRect.zero)
24        addSubview(pupView)
25        addSubview(friendsListLabel)
26        addSubview(addPupFriendButton)
27        setupConstraints()
28    }
29
30    func setupConstraints() {
31        pupView.topAnchor.constraint(equalTo: topAnchor,
32                                     constant: Constants.marginSize)
33        // ...
34    }
35}
```

views controllers



The image shows a screenshot of the Xcode IDE. On the left, there is a storyboard preview window titled "Pup View Controller". It displays a white dog named "Molly" with black eyes and a black nose. Below the dog, the name "Molly" is displayed in bold black text, followed by the text "Friends : Murphy, Roxy". At the bottom of the preview is a black button with the white text "Add Pup as Friend!". On the right side of the interface, there is a code editor window for "PupViewController.swift". The code is written in Swift and defines a class "PupViewController" that inherits from "UIViewController". The code includes outlets for an image view and three labels, properties for a puppy object and friends, and methods for view loading, memory warnings, and a custom action to add a friend. The code editor has syntax highlighting and line numbers.

```
// PupViewController.swift
// pup

import UIKit

class PupViewController: UIViewController {

    @IBOutlet weak var pupImageView: UIImageView!
    @IBOutlet weak var pupNameLabel: UILabel!
    @IBOutlet weak var pupFriendsLabel: UILabel!

    var userPup: Puppy? = nil

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
        // a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

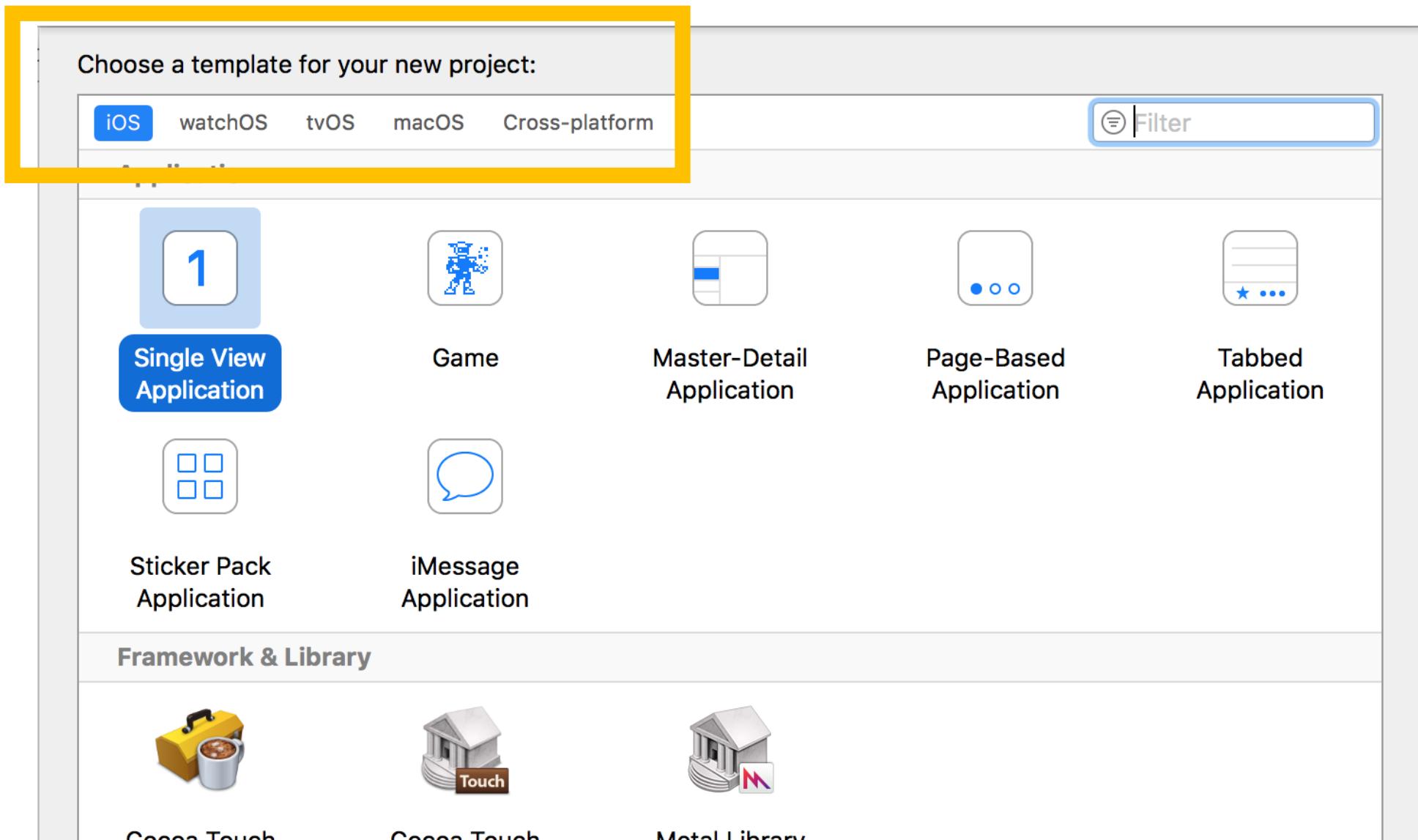
    @IBAction func addPupAsFriend(_ sender: UIButton) {
        if let name = pupNameLabel.text {
            let newPup = Puppy(name: name)
            userPup?.makeNewFriend(withOtherPup: newPup)
        }
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // ...
    }
}
```

check in

Xcode

The IDE to Rule Them All



iosDecal: Ready | Today at 11:25 AM

iosDecal.xcodeproj

iosDecal

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT TARGETS

iosDecal

Display Name: iosDecal

Bundle Identifier: org.akbapu.iosDecal

Version: 1.0

Build: 1

Identity

Signing

Automatically manage signing (checked)

Xcode will create and update profiles, app IDs, and certificates.

Team: Akilesh Bapu (Personal Team)

Provisioning Profile: Xcode Managed Profile ⓘ

Signing Certificate: iPhone Developer: akbapu-14@yahoo.com (TQ56...)

Deployment Info

Deployment Target: 10.2

Devices: Universal

Main Interface: Main

Device Orientation:

- Portrait
- Upside Down
- Landscape Left
- Landscape Right

Status Bar Style: Default

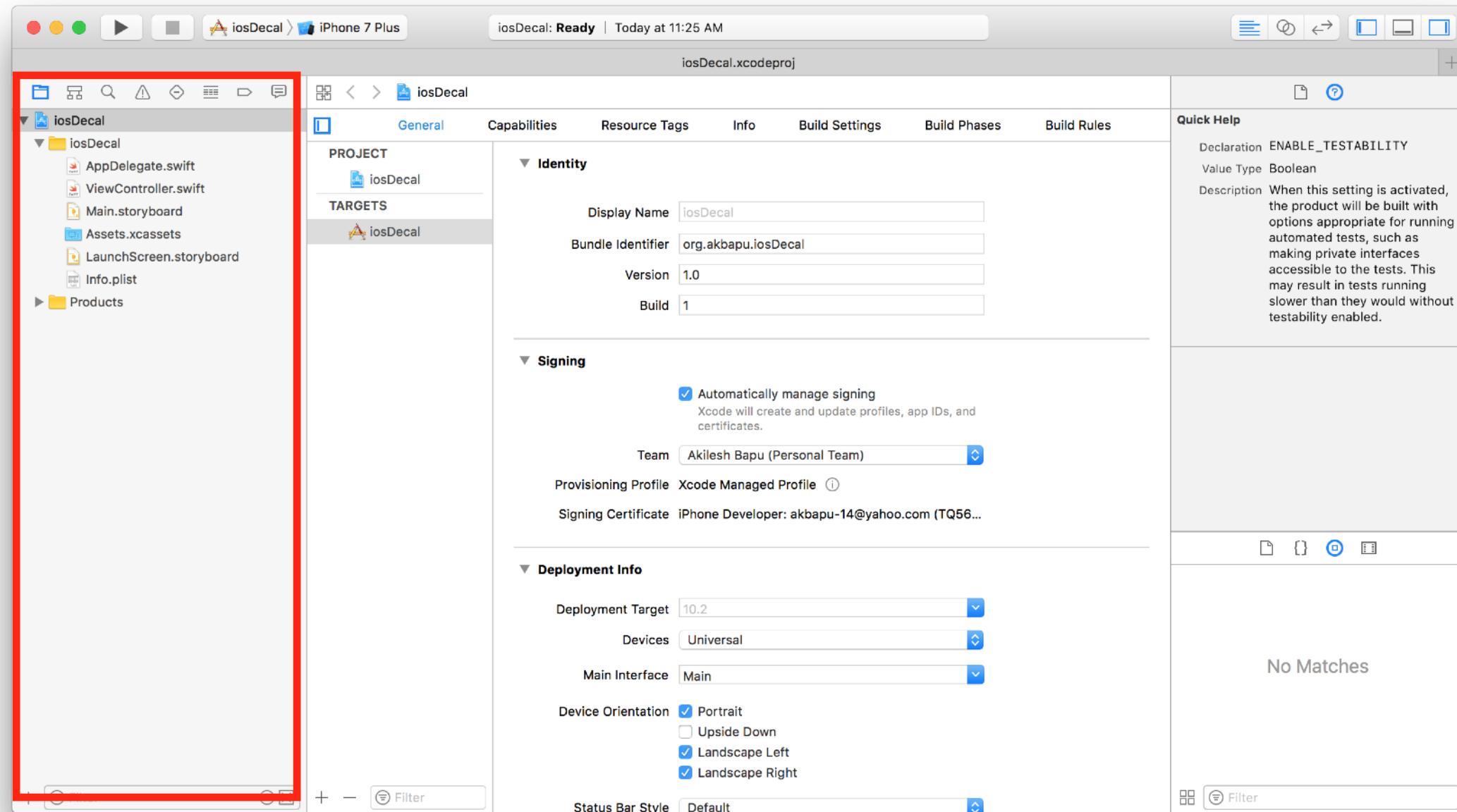
No Matches

Quick Help

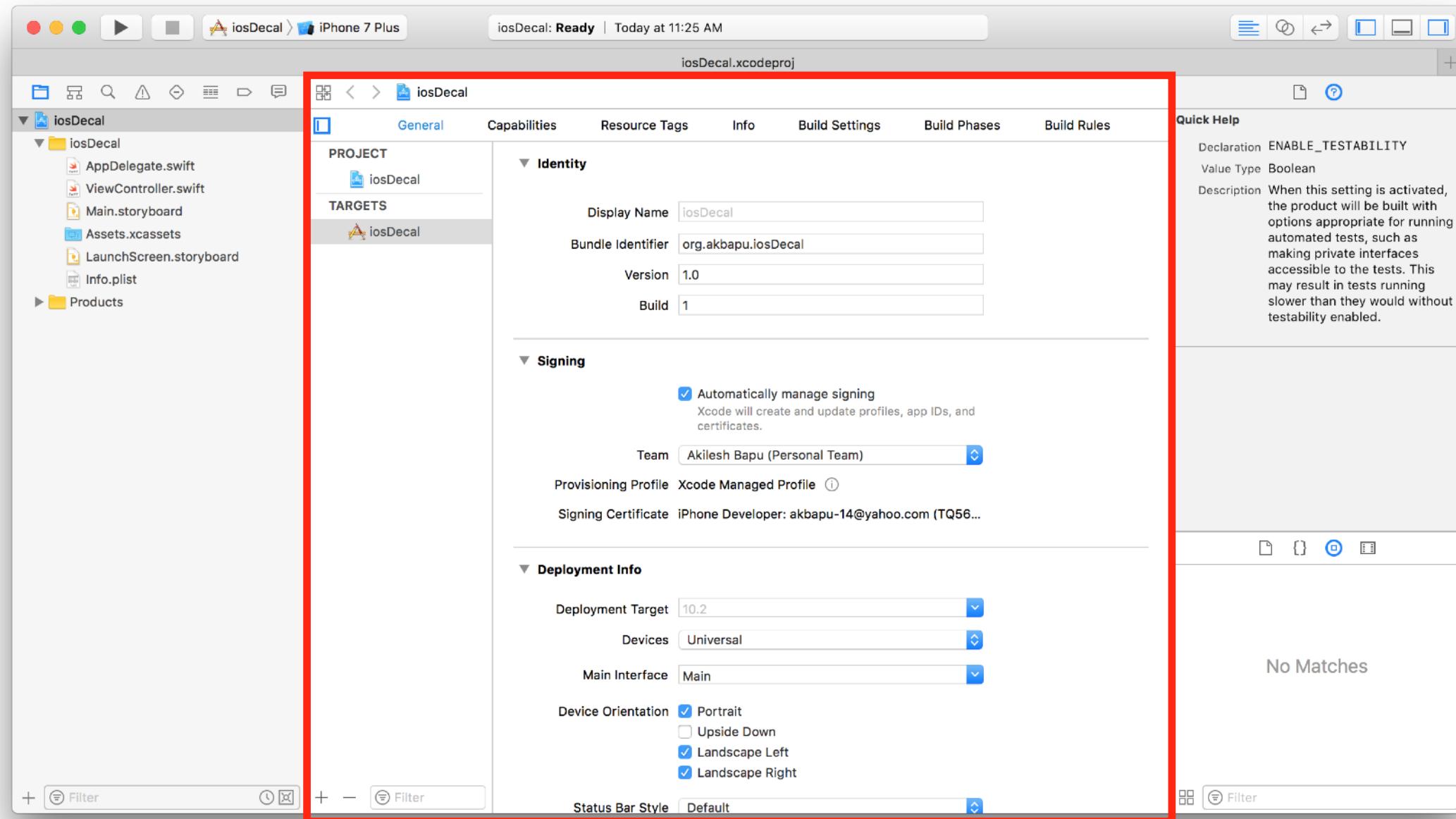
Declaration: ENABLE_TESTABILITY

Value Type: Boolean

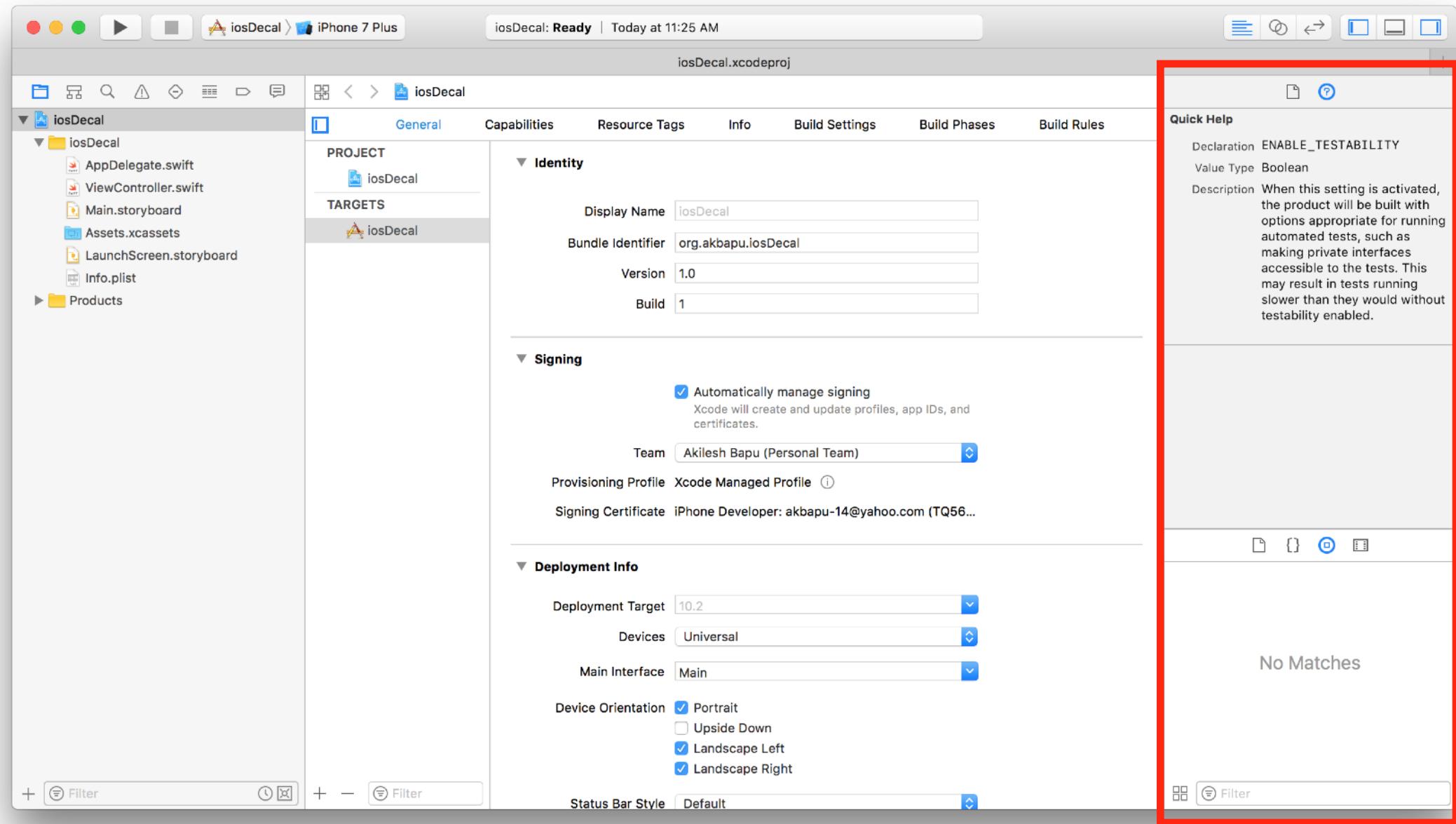
Description: When this setting is activated, the product will be built with options appropriate for running automated tests, such as making private interfaces accessible to the tests. This may result in tests running slower than they would without testability enabled.



Project
Navigator



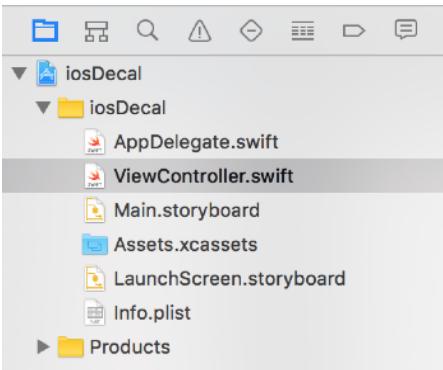
Project/Target Settings



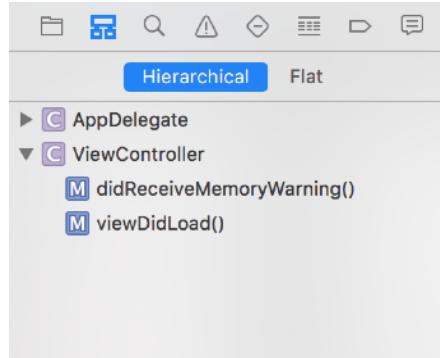
Utilities Area

Xcode structure: left panel

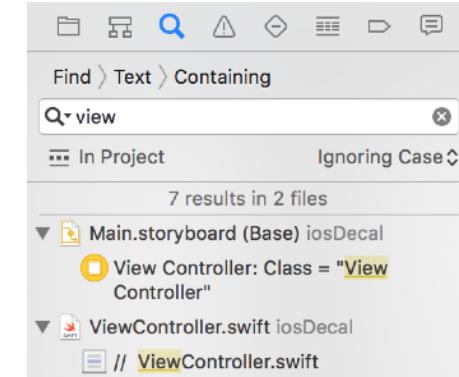
Project Navigator



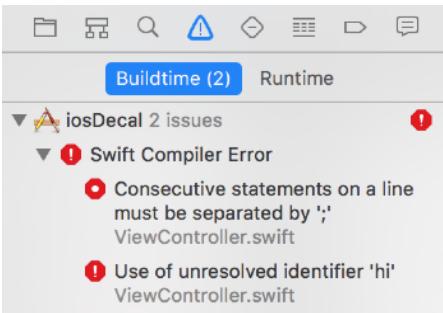
Symbol Navigator



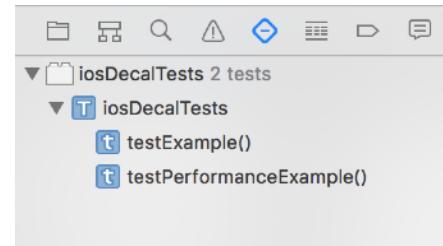
Find Navigator



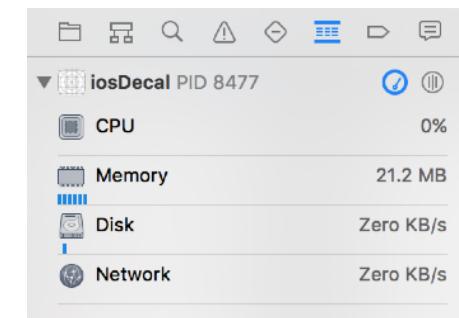
Issue Navigator



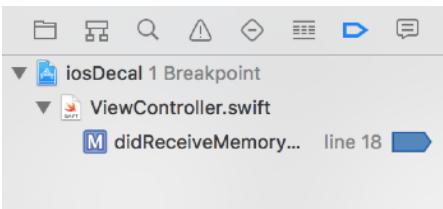
Test Navigator



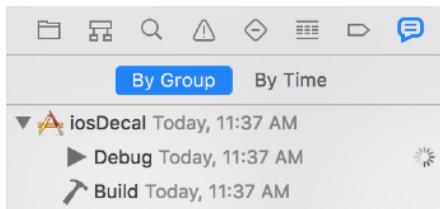
Debug Navigator



Breakpoint Navigator



Report Navigator

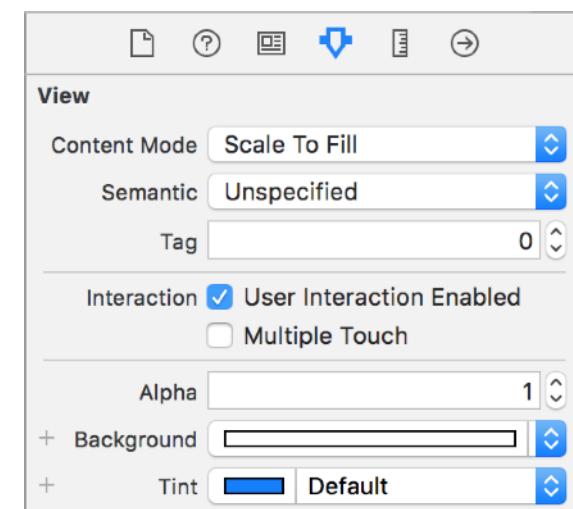
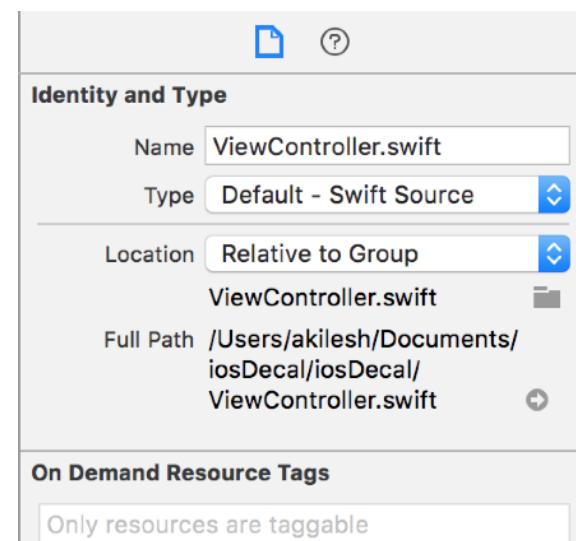


Xcode navigator

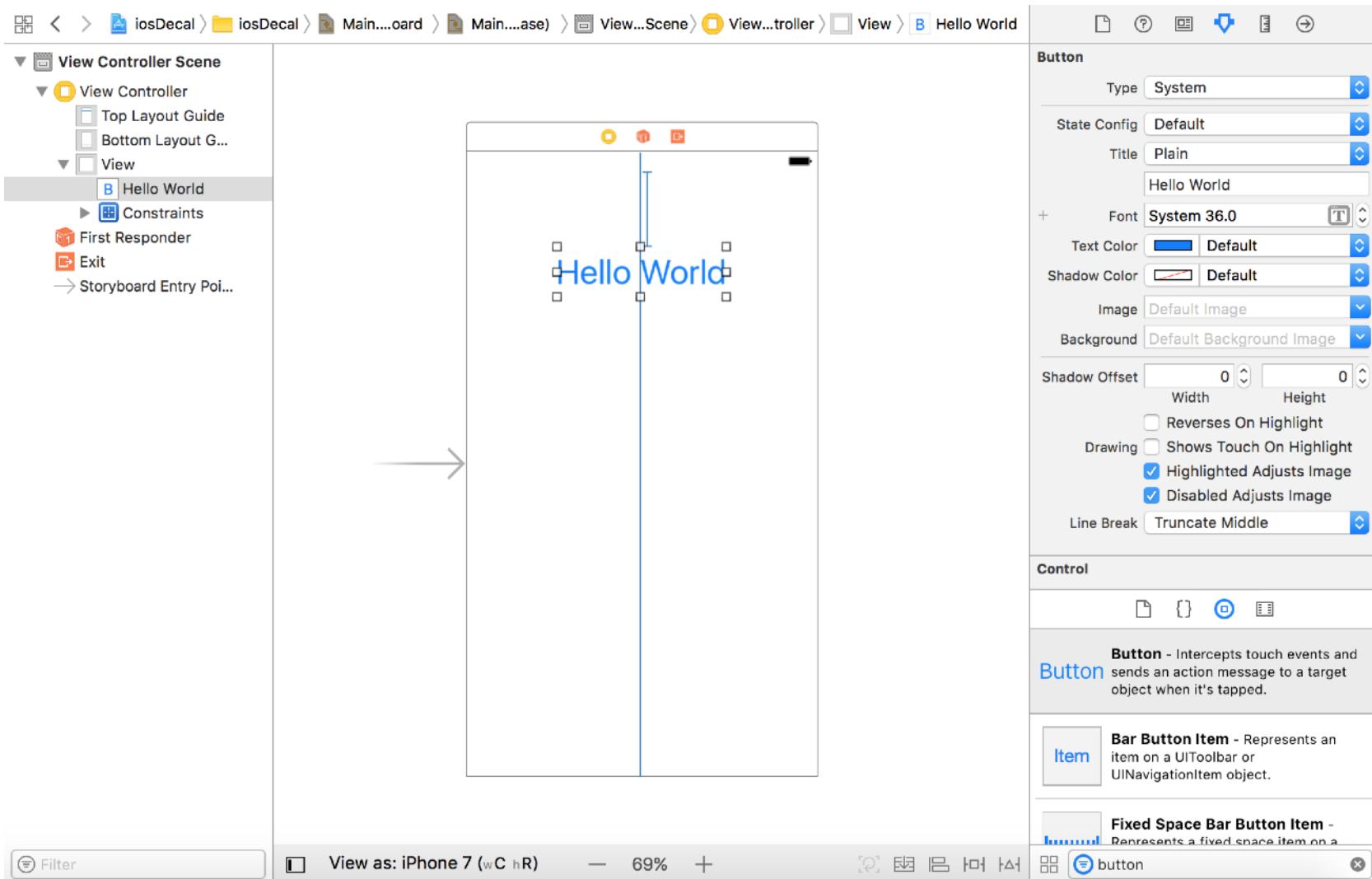
Debug Area



Utilities Area



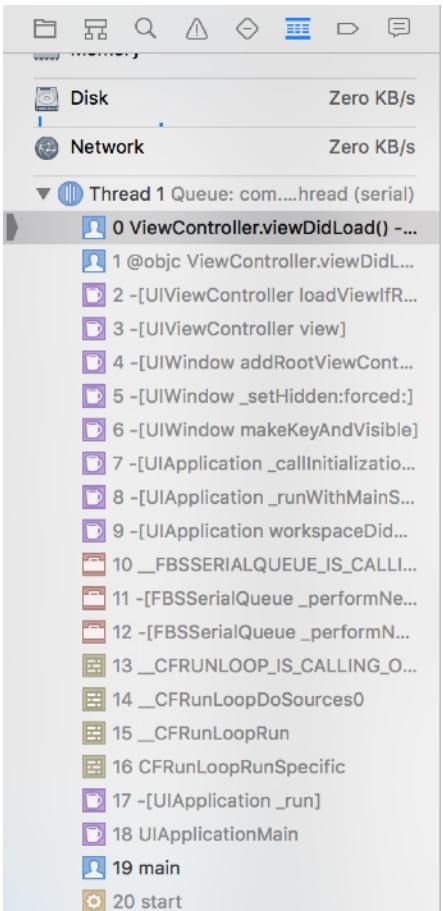
storyboards



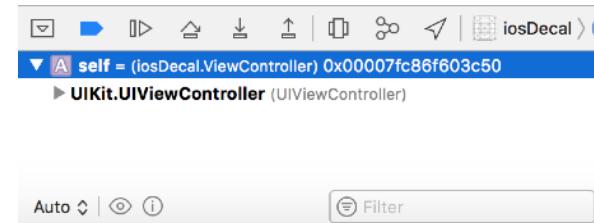
Xcode Tips and Tricks

debugging

Call Hierarchy - Great if you step through code and end up at some random file and want to see how it's relevant



Currently Variables in Use - You can actually preview views

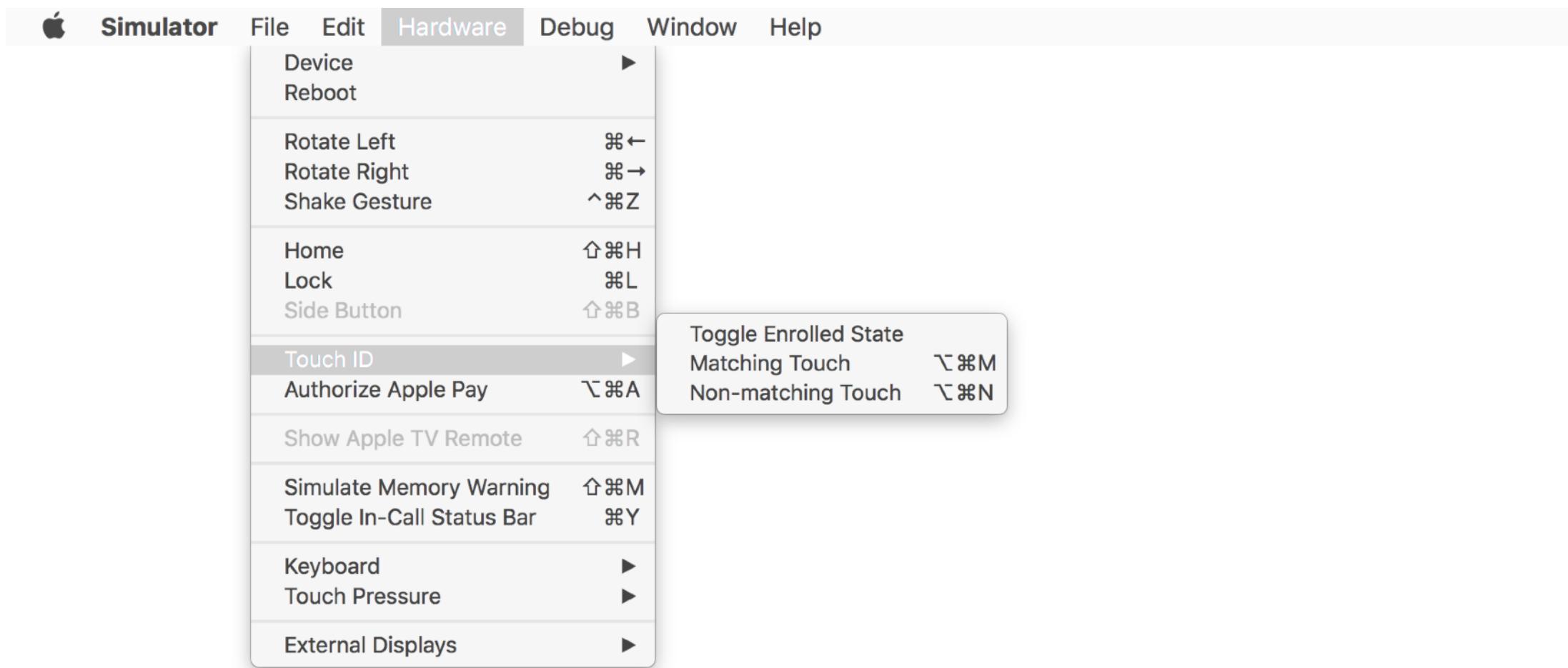


LLDB - Console Output + "po" (RUN CODE AS THE APP IS GOING)



simulator settings

Hardware and Debug Menus - Easily ignorable but amazingly helpful



Xcode IDE Demo

Let's Create an App

lab 1 – Xcode tutorial

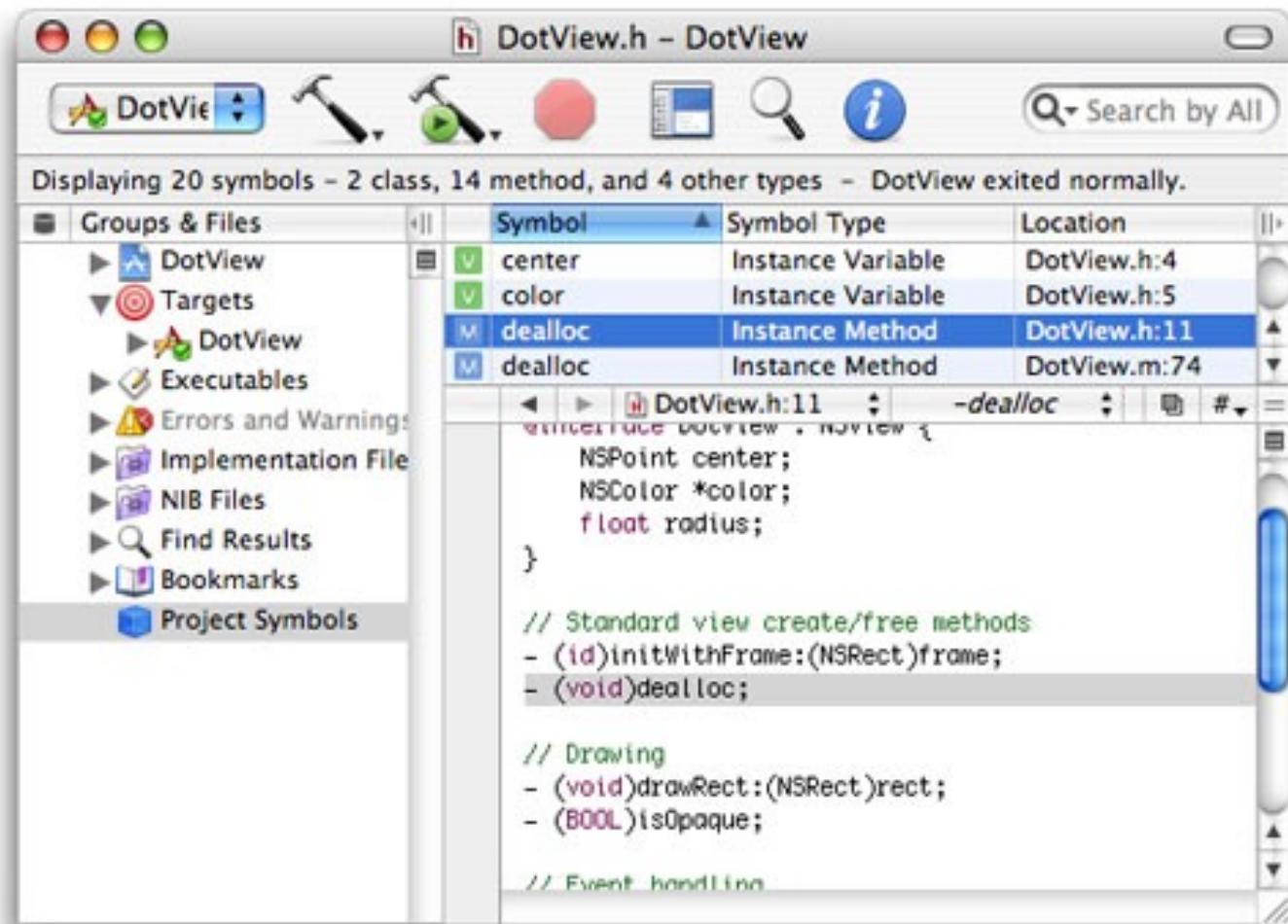
Wednesday at 7pm

remember to sign up for a lab section on piazza!

extra slides

Project Builder

- IDE for the NeXTSTEP OS (source of macOS, iOS, tvOS, etc.)
- rewritten for OS X, and rebranded as **Xcode**



The screenshot shows the Project Builder interface for the file `DotView.h`. The window title is `DotView.h - DotView`. The toolbar includes icons for file operations, a magnifying glass, and an information icon. A search bar at the top right says "Search by All". The left sidebar lists project files: `DotView`, `Targets`, `DotView`, `Executables`, `Errors and Warnings`, `Implementation File`, `NIB Files`, `Find Results`, `Bookmarks`, and `Project Symbols`. The main area displays the contents of `DotView.h`:

```
Displaying 20 symbols - 2 class, 14 method, and 4 other types - DotView exited normally.

Symbol Symbol Type Location
center Instance Variable DotView.h:4
color Instance Variable DotView.h:5
dealloc Instance Method DotView.h:11
dealloc Instance Method DotView.m:74

@interface DotView : NSView {
    NSPoint center;
    NSColor *color;
    float radius;
}

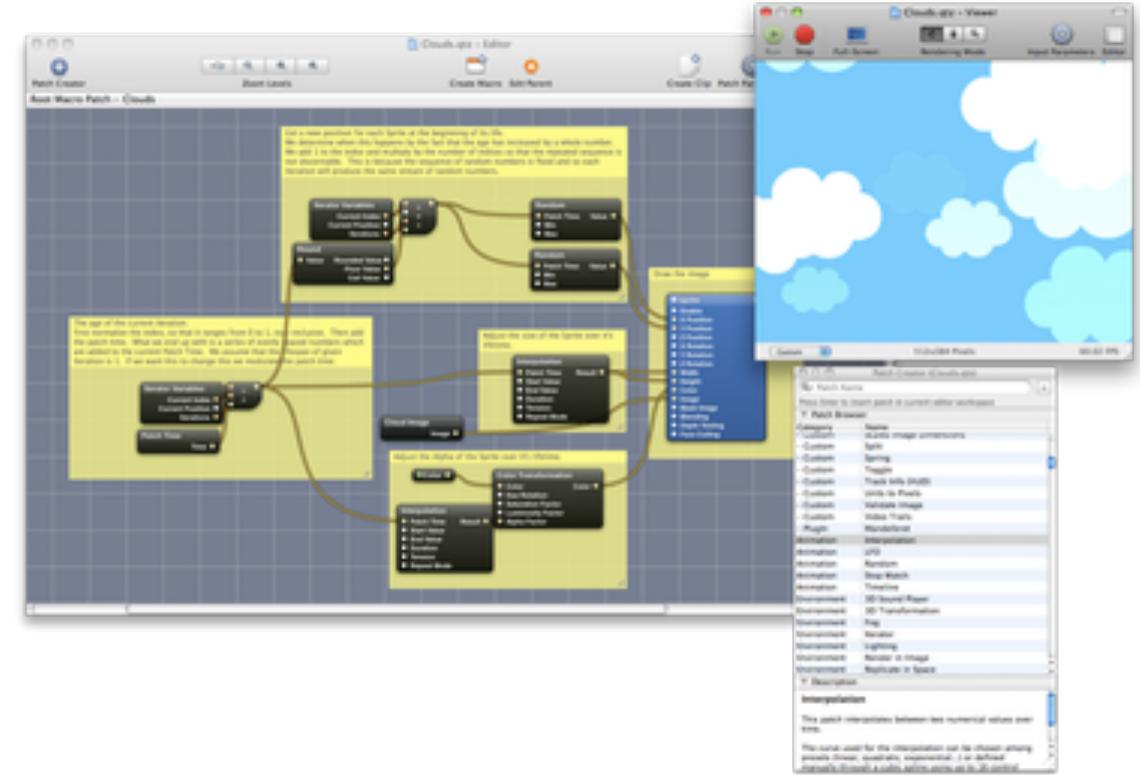
// Standard view create/free methods
- (id)initWithFrame:(NSRect)frame;
- (void)dealloc;

// Drawing
- (void)drawRect:(NSRect)rect;
- (BOOL)isOpaque;

// Event handling
- (void)mouseDown:(NSEvent *)theEvent;
```

Xcode versions

- 2003: Xcode 1.0
- 2005: Xcode 2.x
 - included a visual programming language (Quartz Composer)
 - breakpoints and watchpoints
- 2007: Xcode 3.x
 - 2008: iOS SDK released to third party developers



the Quartz Composer