Introduction to Mobile Computing

Dr. Sarah Abraham

University of Texas at Austin
CS329e
Spring 2020

Mobile Computing

- Computers increasingly prevalent in daily life
    - Constant access to information and entertainment
    - Different types of user interfaces and displays
    - Restrictions on power usage and performance
- Mobile development requires:
    - Specific mobile programming languages
    - Database information
    - Device information
    - Novel ideas that provide customers value

Class Expectations

- Lab and project-based work
  - No exams
  - Weekly assignments/labs to build practical skills
  - Final team project to show-case understanding
- Engaged and helpful attitude
  - Ask and answer questions on Piazza
  - Academic honesty required
  - Positive teamwork and interactions
  - Ability to read syllabus and schedule on your own!

Class Format

- Lecture days provide overview of material and in-class examples
- Lab days allow students to work through tutorials and do hands-on development
- Attendance for both days are mandatory!
  - In-class/lab quizzes using Instapoll (via Canvas)
- Final project: building a complete app
  - Team-based
  - On-going reports and testable products

Class Communication

- We use Piazza for class communication
  - Announcements, issues, and questions, etc
  - You can post short (no more than 3 lines) code snippets with the class, or privately share longer code segments with the TA/professor
  - Good place to ask for help and post solutions you've discovered

Topics Covered

- iOS development framework
- Swift language
- Related programming paradigms
- Data input
- Mobile interfaces
- Common iOS frameworks
- Project development cycles and practices

What Apps Do You Use?

- What are some of the design considerations?
  - How do it utilize screen space?
  - How long does the battery last?
  - How nice are the graphics?
  - What does it require for networking functionality?

Working in iOS

- Requires ready access to Macs!
  - Macs in the PCL Media Lab
  - Mac laptop highly, highly recommended
- Use Xcode (Apple's free IDE) version 11 and Swift 5

Xcode Download

- https://developer.apple.com/xcode/downloads/

or

- https://itunes.apple.com/us/app/xcode/id497799835?mt=12

Xcode Setup

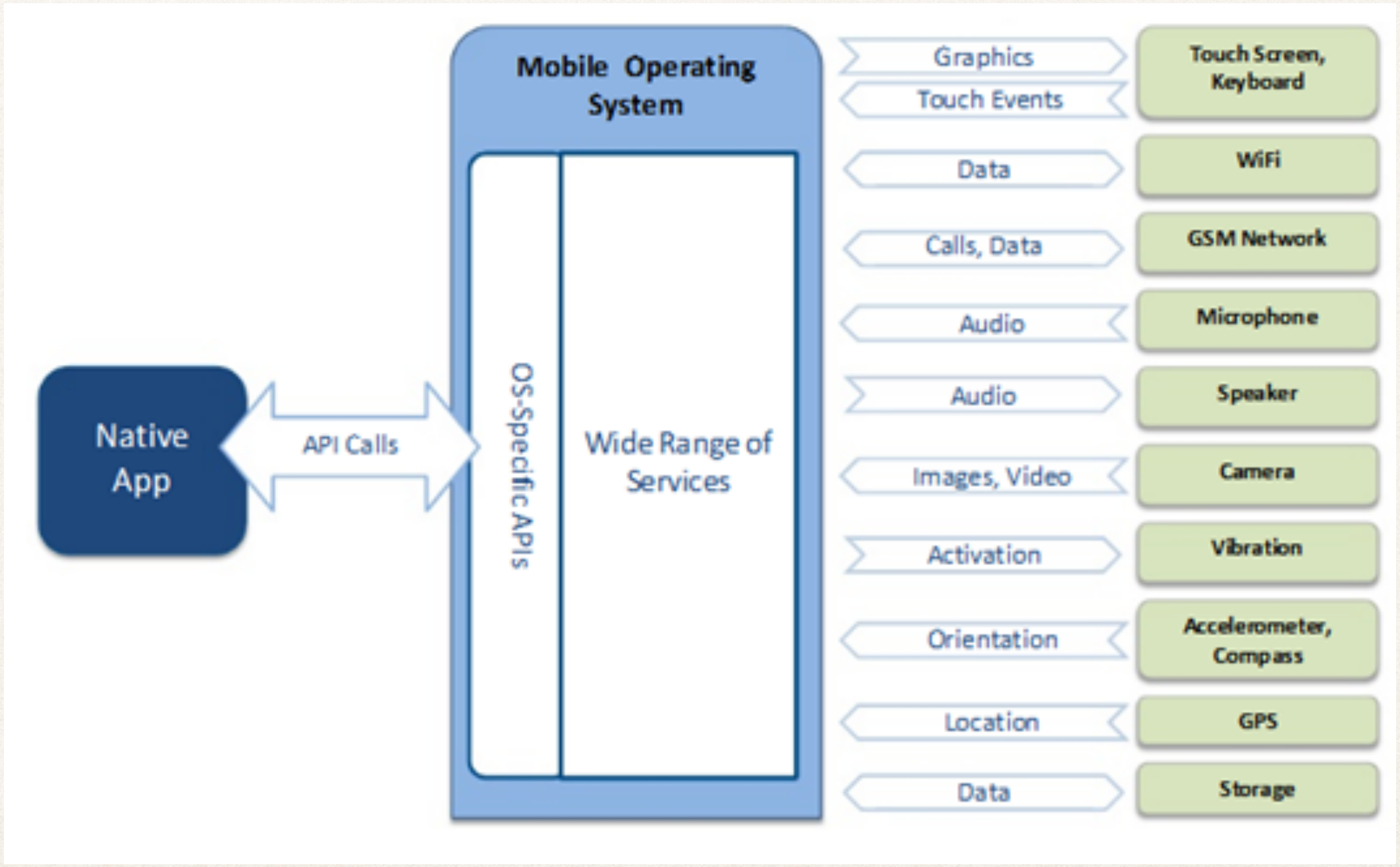
- Find Xcode after install in *Applications* folder
- Launch Xcode and keep in dock
    - Right click -> Options -> Keep in Dock

Playgrounds

- Interactive environment that allows developers to write Swift interactively and see results immediately
    - File -> New -> Playground
- Allows for experimentation

Xcode and Playgrounds Demo

Anatomy of a Mobile App



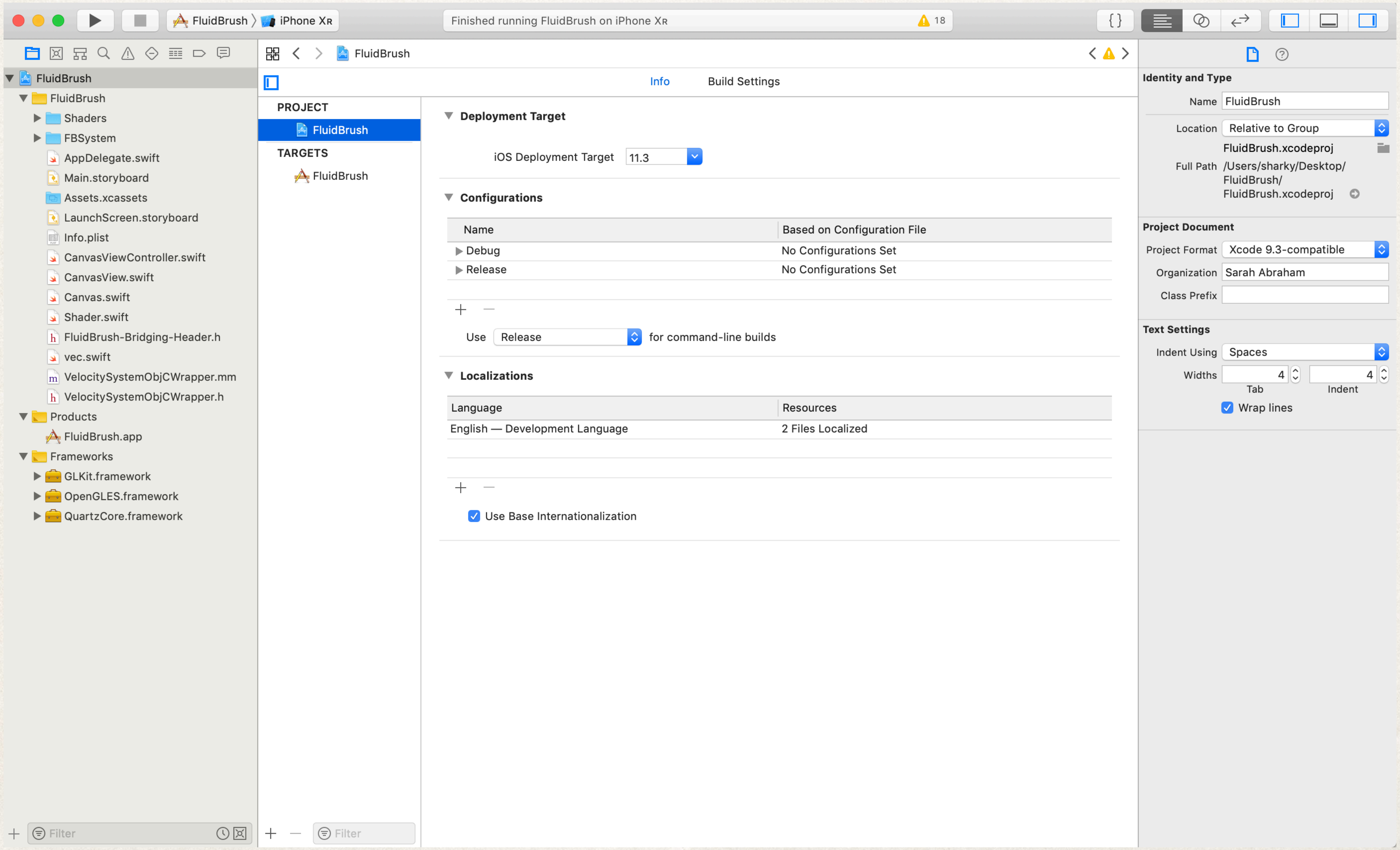
(worklight.com)

- Apps built on a common set of phone features
    - Libraries provided to use these features
    - Standardized API calls access these libraries
    - Third-party apps built upon these calls
    - Libraries optimized and reusable in memory
- Less code to write and better performance

Frameworks

- Key pieces of code that make mobile applications easy to build and stable (ideally)
- Bundle (structured directory) contains:
  - Dynamic, shared library
  - Associated resources (images, headers etc)
- Frameworks shared between applications
- Fast access, reduced memory, consistent look 'n' feel

Framework Example



Deployment Target

iOS Deployment Target: 11.3

Configurations

Name	Based on Configuration File
Debug	No Configurations Set
Release	No Configurations Set

Use Release for command-line builds

Localizations

Language	Resources
English — Development Language	2 Files Localized

☑ Use Base Internationalization

Identity and Type

Name	FluidBrush
Location	Relative to Group
	FluidBrush.xcodeproj
Full Path	/Users/sharky/Desktop/FluidBrush/FluidBrush.xcodeproj

Project Document

Project Format	Xcode 9.3-compatible
Organization	Sarah Abraham
Class Prefix	

Text Settings

Indent Using	Spaces
Widths	4 (Tab) 4 (Indent)

☑ Wrap lines

Using Frameworks

- Frameworks are designed for specific functionality
    - Native code should fit its framework (not the other way around)
    - Native code should make use of frameworks
- iOS development based on frameworks (Foundation, UIKit etc)

- Framework has default set of behaviors/functionality (i.e. methods)
- Programmer uses these methods to output desired behavior for app
- Developer code written to specialize the framework's behavior

Common iOS Frameworks

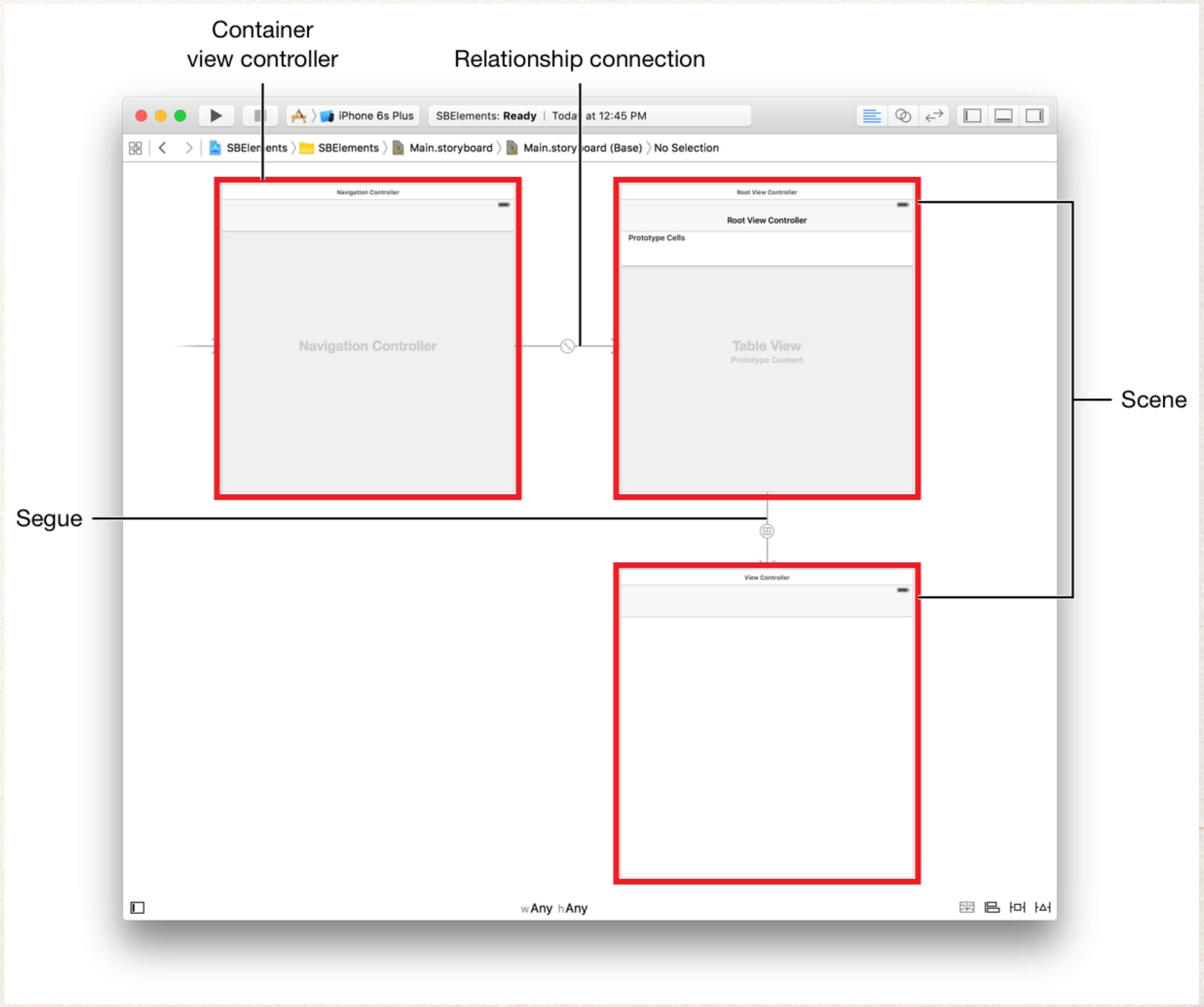
- Foundation: Low-level management of strings, collections, primitive data types, containers etc
- UIKit: Class-level management of iOS user-interface layer
- CoreData: Interfaces for managing app and user data
- CoreGraphics: Interfaces for 2D vector-based drawing engine

Other Systems in iOS

- Storyboard/SwiftUI: Defines user interface and app flow
- Delegate: Coordinates multiples pieces and systems in the app
- Views: Elements of the user interface
- View Controller: Manages user interface and display

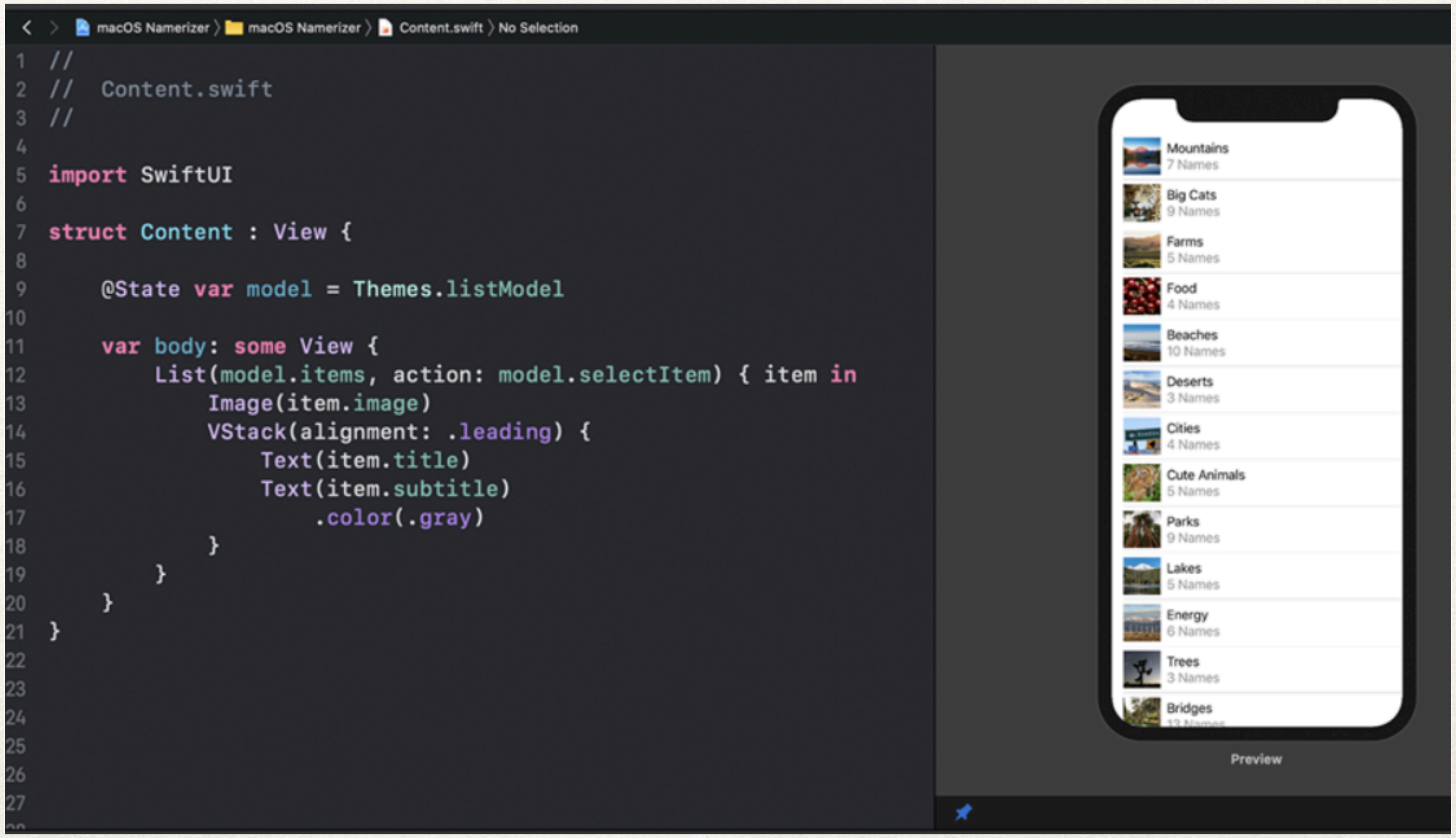
Storyboard

- Lays out user's path
- Defines UI of scenes
- Defines segues between scenes
- Uses Auto Layout for nice formatting



SwiftUI

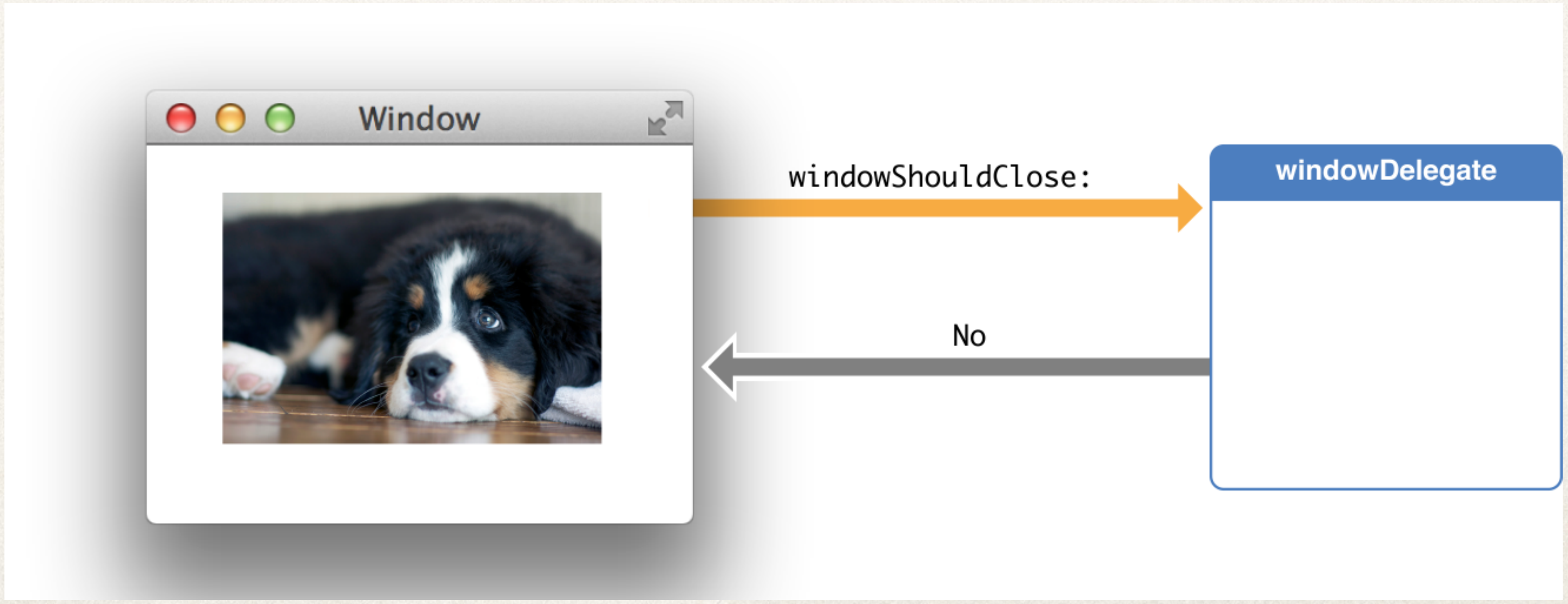
- Programmatic way of building out UI code



```swift
//
//  Content.swift
//

import SwiftUI

struct Content : View {

    @State var model = Themes.listModel

    var body: some View {
        List(model.items, action: model.selectItem) { item in
            Image(item.image)
            VStack(alignment: .leading) {
                Text(item.title)
                Text(item.subtitle)
                    .color(.gray)
            }
        }
    }
}
```

Delegate

- Pattern of development where one object in a program acts on another object's behalf
- Coordinates between objects by passing messages
- Can return values to determine how to handle event

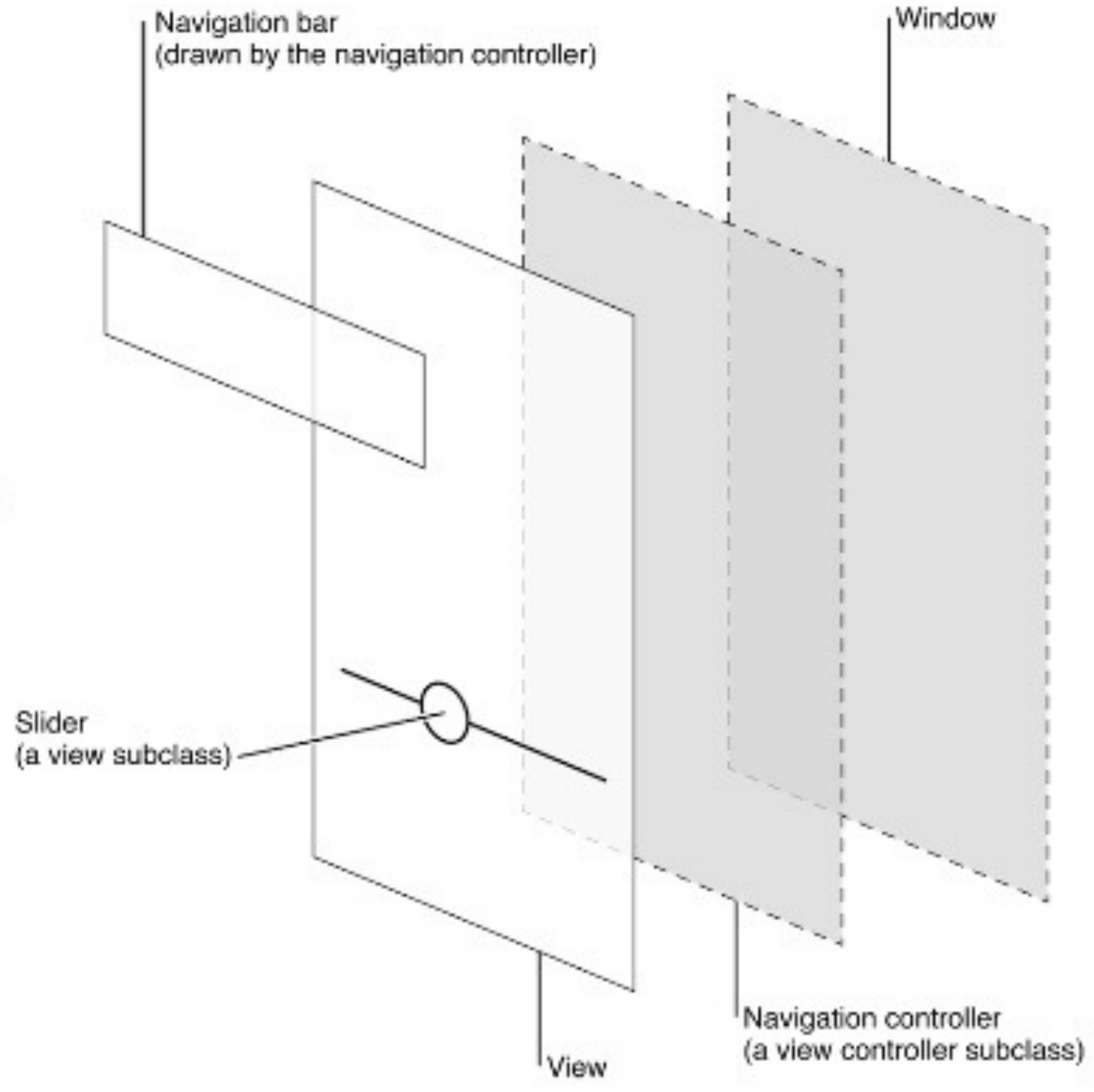


Views

- Display presented to user
- Controls layout and subviews
- Handles drawing and animation
- Responds to events
- Created programmatically or through Storyboards

iOS App Anatomy

Windows layers of an iOS app



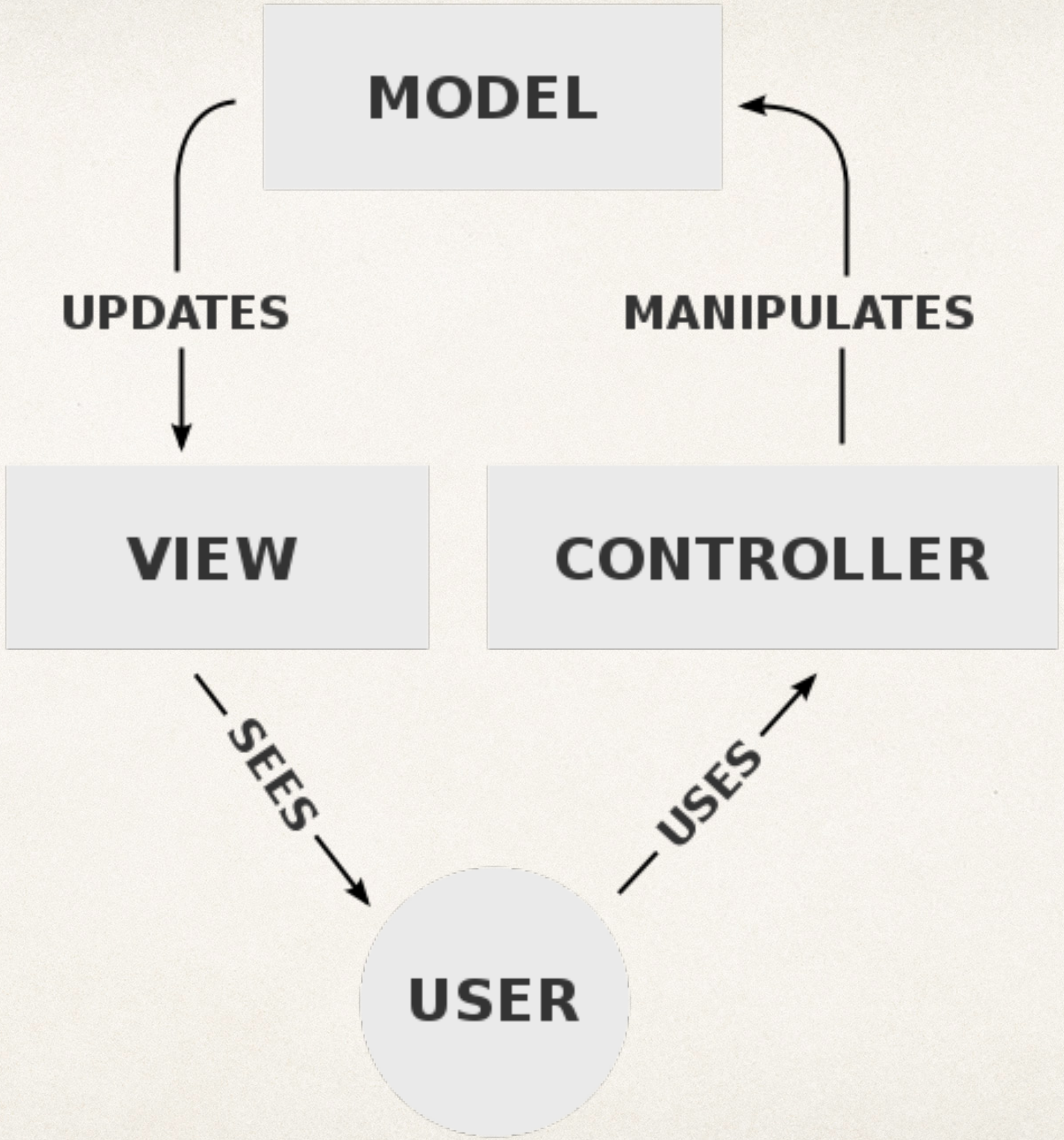
(UI/UX design for iOS 7, Vu Tran Lam)

View Controller

- At least one per app
- Manages defined part of user interface
- Handles interactions between interface and underlying data
- Central to app development

MVC Pattern

- Pattern guiding all iOS development
- Model-View-Controller
  - Model includes app-specific data, classes etc
  - View includes interface and screens presented to user
  - Controller dictates how model and view should change based on user input



(Wikipedia)

Working in Xcode on Campus

- PCL Media Lab has 44 iMacs with Xcode installed
  - http://www.lib.utexas.edu/services/media-labs/
  - Xcode may not be on latest version though
- To use Xcode:
  1. Open Xcode (use Spotlight Search or go through Applications folder)
  2. Check "Don't Enable" when pop asks if you want to enable developer mode
  3. Enter your EID and password when it provides a prompt
  4. Xcode should now run

Xcode Layout Demo