G54MDP
Mobile Device Programming

iOS, Revision

ObjC Messages

- Programming based on **message passing** between objects
  - Rather than calling methods directly
  - Send messages to an object to call a method
- Target is resolved at runtime
  - Not compile time
  - Receiving object interprets the message
- An object is not guaranteed to respond to a message
  - Raises an exception
  - Send messages to a collection of objects
    - Only some may be expected to respond
  - Objects do not have to be defined at compile time
  - Can **forward** messages to other objects
    - Delegation

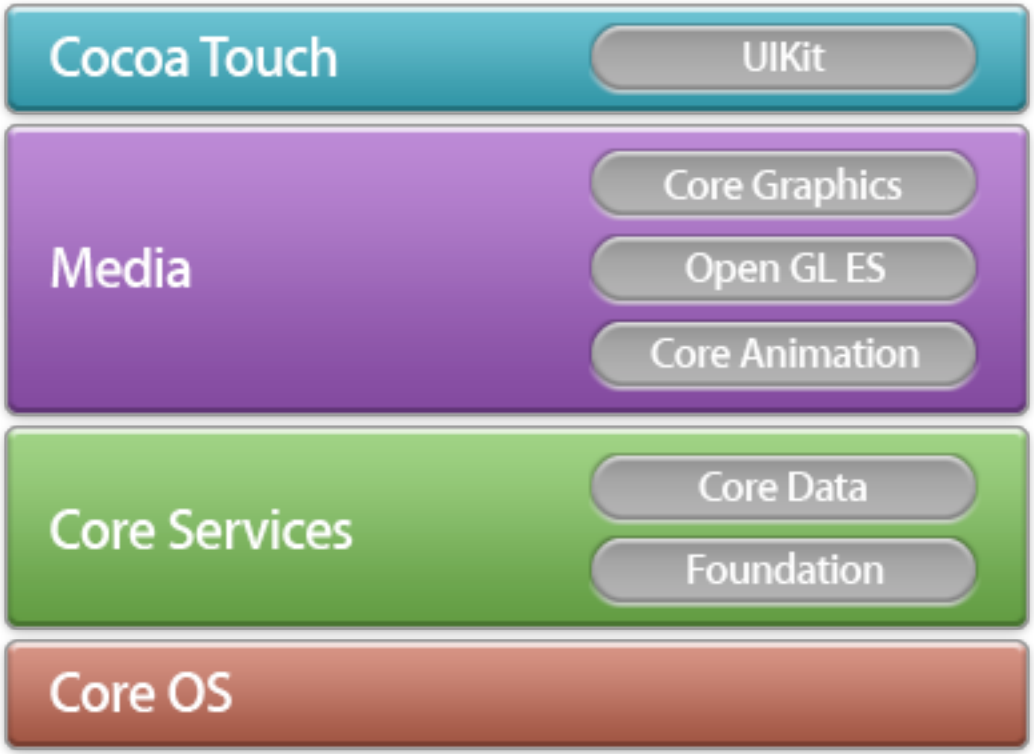
ObjC Categories

- Adding methods to a class at runtime
  - Without the need to recompile / access to source code
    - Cf "Monkey patching" in Ruby, but by design
  - Define a **category** that specifies new methods to add to an existing class
- Replace existing methods
- Add new functionality
  - E.g. add a spellchecker to a TextEdit component

iOS Frameworks

- iOS comes with several frameworks that can help us with development
  - Foundation framework provides support for strings, files, collections etc
  - Other Frameworks provide support for Audio, video, animation, location etc
  - At the top is the UI framework, CocoaTouch
    - Widgets, buttons, views
- iOS is very much an evolution of PC GUI programming into the mobile space
  - Particularly MacOS X GUI programming
  - Almost every class in CocoaTouch has an equivalent in OS X
  - Vs Android major components

iOS Frameworks

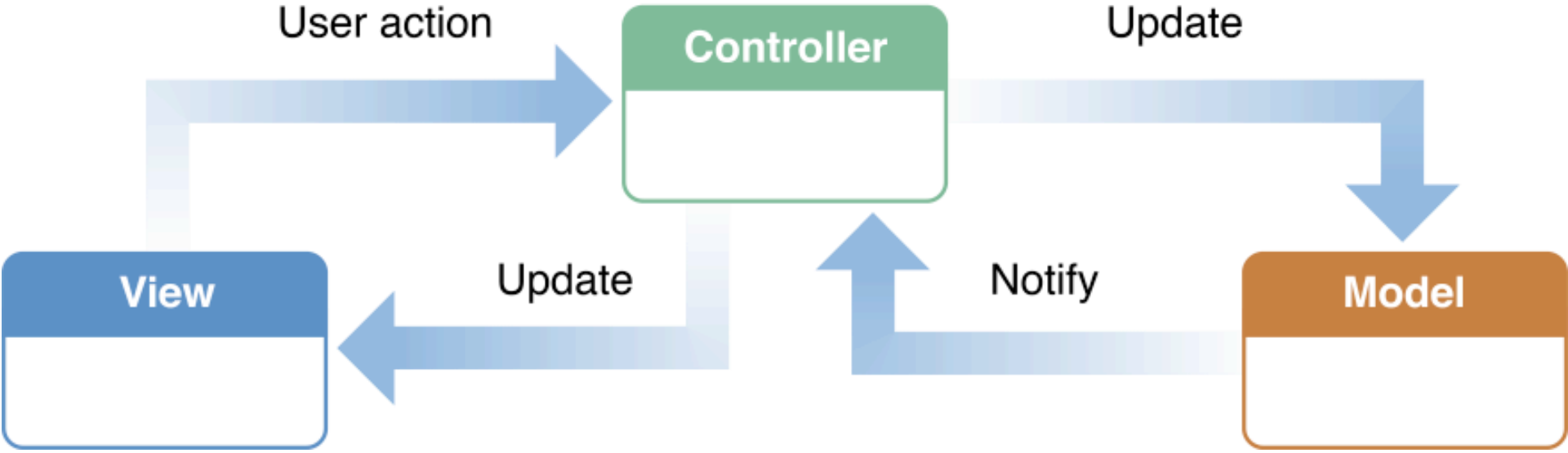


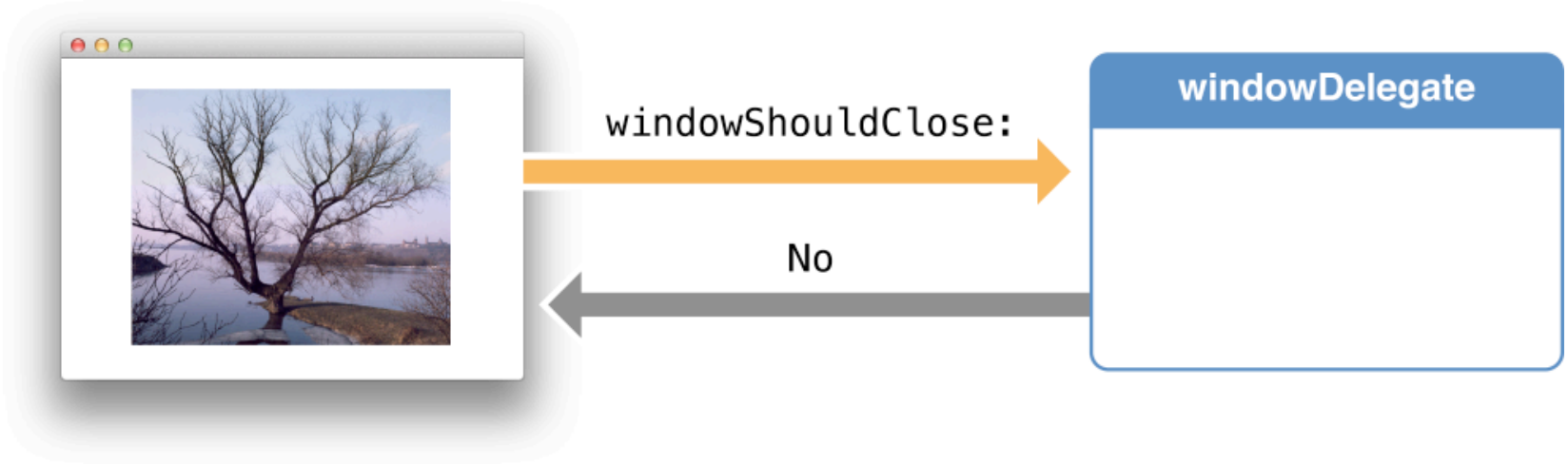
Design Patterns

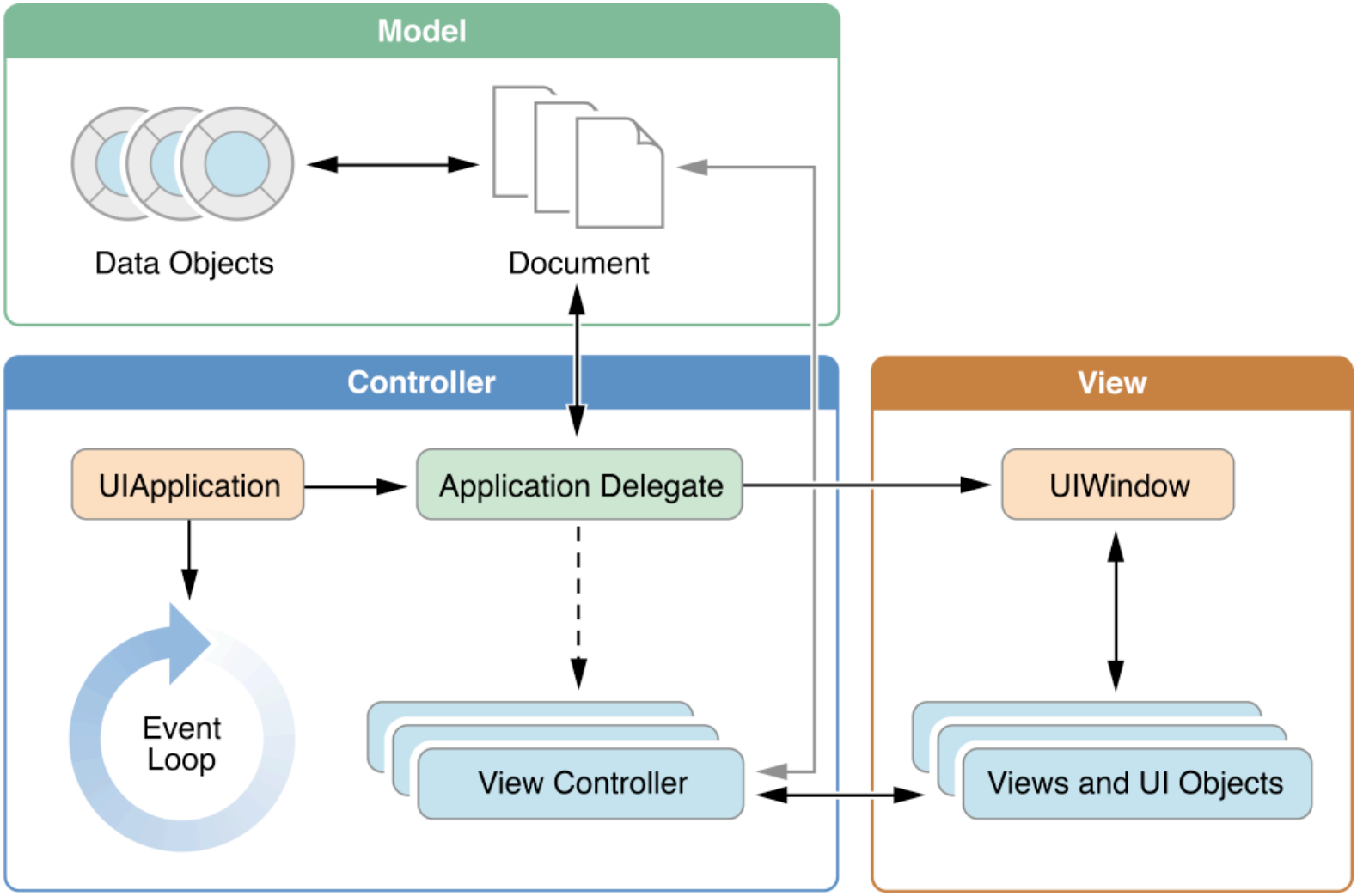
- iOS / Cocoa framework strongly suggest use of certain design patterns
  - **Model View Controller**
  - Delegation
  - Protocols
  - Notification
  - Target-Action
  - Key-Value Observation

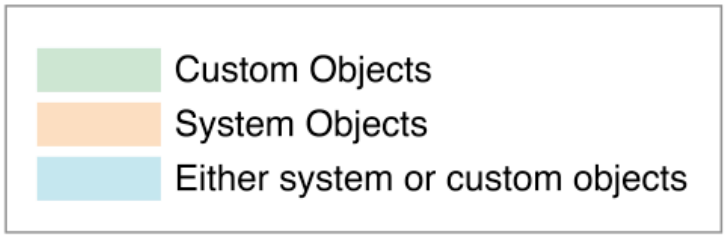
Model View Controller

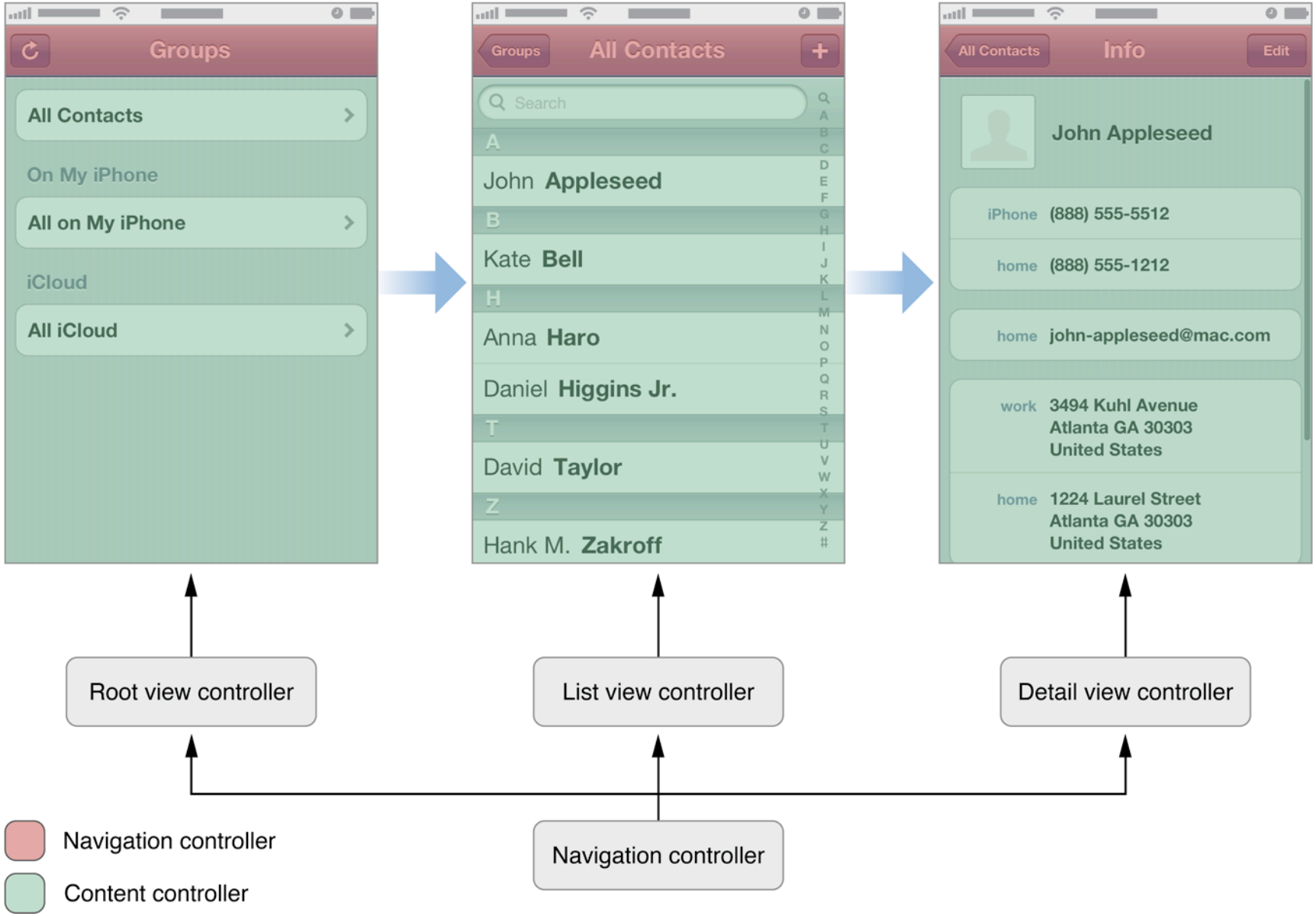
- Divide objects into three types
- Model
  - What the application **is**, but not how it is displayed
  - A contact in an address book
- Controller
  - **How** the model is presented to and manipulated by the user
  - Add / read / modify a contact
- View
  - Drawing things on the screen
  - Render a text view containing the contact
- MVC design pattern determines how these components should communicate
  - The model and view are typically decoupled


User action
Controller
Update
View
Update
Notify
Model


windowShouldClose:
No
windowDelegate

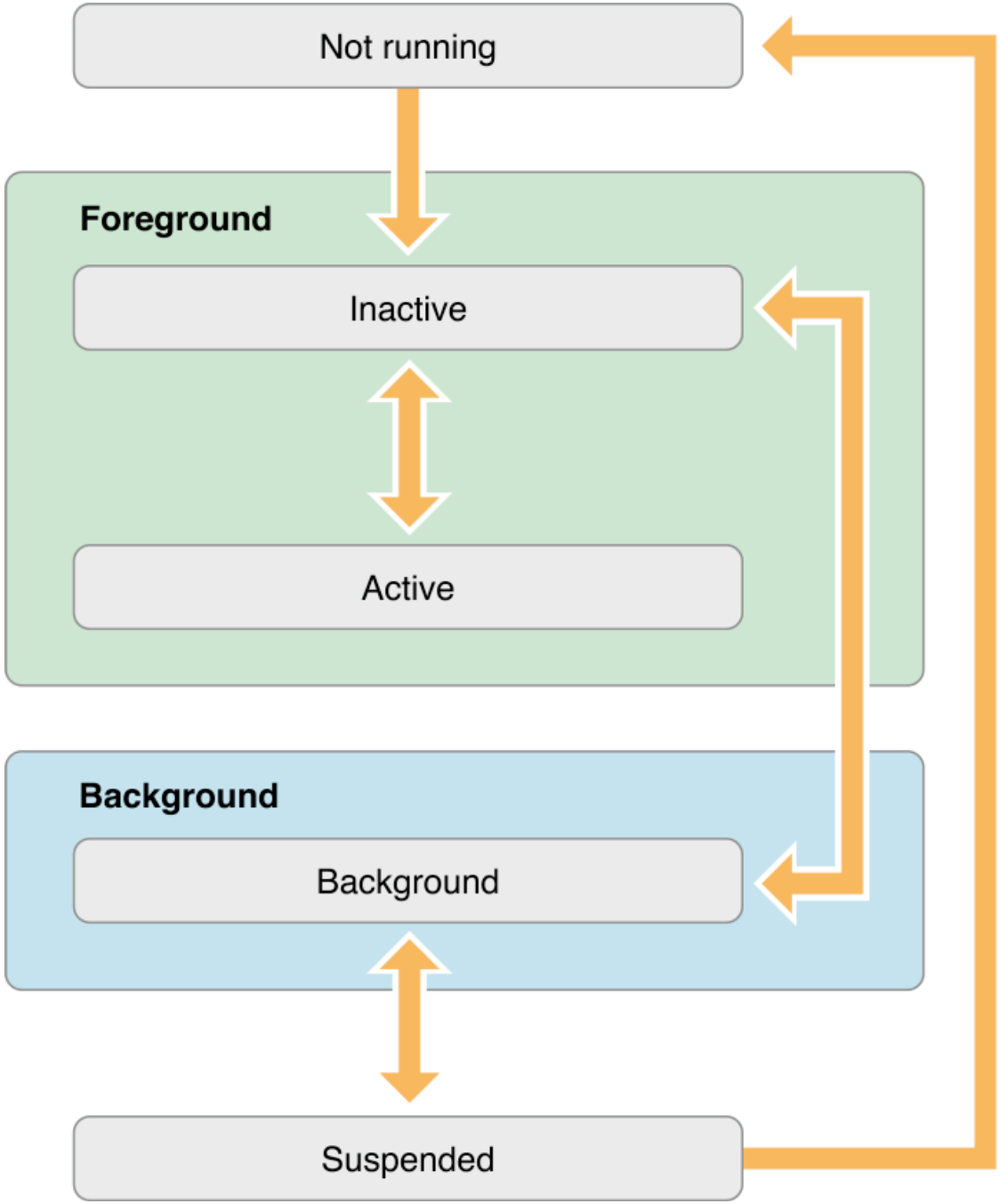

Model
Data Objects
Document
Controller
UIApplication
Application Delegate
Event Loop
View Controller
View
UIWindow
Views and UI Objects


Custom Objects
System Objects
Either system or custom objects


Groups
All Contacts
On My iPhone
All on My iPhone
iCloud
All iCloud
Groups
All Contacts
Search
A
John Appleseed
B
Kate Bell
H
Anna Haro
Daniel Higgins Jr.
T
David Taylor
Z
Hank M. Zakroff
All Contacts
Info
Edit
John Appleseed
iPhone (888) 555-5512
home (888) 555-1212
home john-appleseed@mac.com
work 3494 Kuhl Avenue
Atlanta GA 30303
United States
home 1224 Laurel Street
Atlanta GA 30303
United States
Root view controller
List view controller
Detail view controller
Navigation controller
Navigation controller
Content controller

iOS Lifecycle

- Analogous to Android lifecycle
  - Only one application in the foreground / visible at any one time
- A main loop processes events for the application
- An app can be a number of significant states
  - Active – foreground
  - Inactive – foreground but interrupted
    - By a phonecall, notification etc
  - Background – can remain in this state to perform long running tasks
    - Analogous to Services
  - Suspended
    - Main loop no longer running, potentially killed by the operating system
- iOS 3.2 and earlier
  - No support for suspended / background states
    - No long running tasks


Not running
Foreground
Inactive
Active
Background
Background
Suspended

App Store

- “We will reject Apps for any content or behavior that we believe is over the line. What line, you ask? Well, as a Supreme Court Justice once said, "I'll know it when I see it". And we think that you will also know it when you cross it.”
- Pre-moderation
  - Apple approves all applications in advance
  - Vs Android – publish then revoke
- A long list of guidelines as to what is appropriate
  - Correct use of interface components
  - Substantial content

App Store Restrictions

- **2.5** Apps that use non-public APIs will be rejected
- **2.8** Apps that install or launch other executable code will be rejected
- **2.10** iPhone Apps must also run on iPad without modification, at iPhone resolution, and at 2X iPhone 3GS resolution
- **2.16** Multitasking Apps may only use background services for their intended purposes: VoIP, audio playback, location, task completion, local notifications, etc.
- **2.17** Apps that browse the web must use the iOS WebKit framework and WebKit Javascript
- **13.2** Apps that rapidly drain the device's battery or generate excessive heat will be rejected

Cross Platform?

- Apps developed for one system won't work on another
- Would need to port it over
  - This can actually be desirable
    - Can tailor our app to the look and feel of the target device
    - Apple encourage the use of iOS "metaphors"
      - Sliding on/off switches, spinning picker wheels
  - Significant coding effort
- However, there are times when it is desirable to target multiple platforms
  - In-house apps
  - Games (Platform chrome usually irrelevant)
- What are the issues behind cross-platform support?

Language

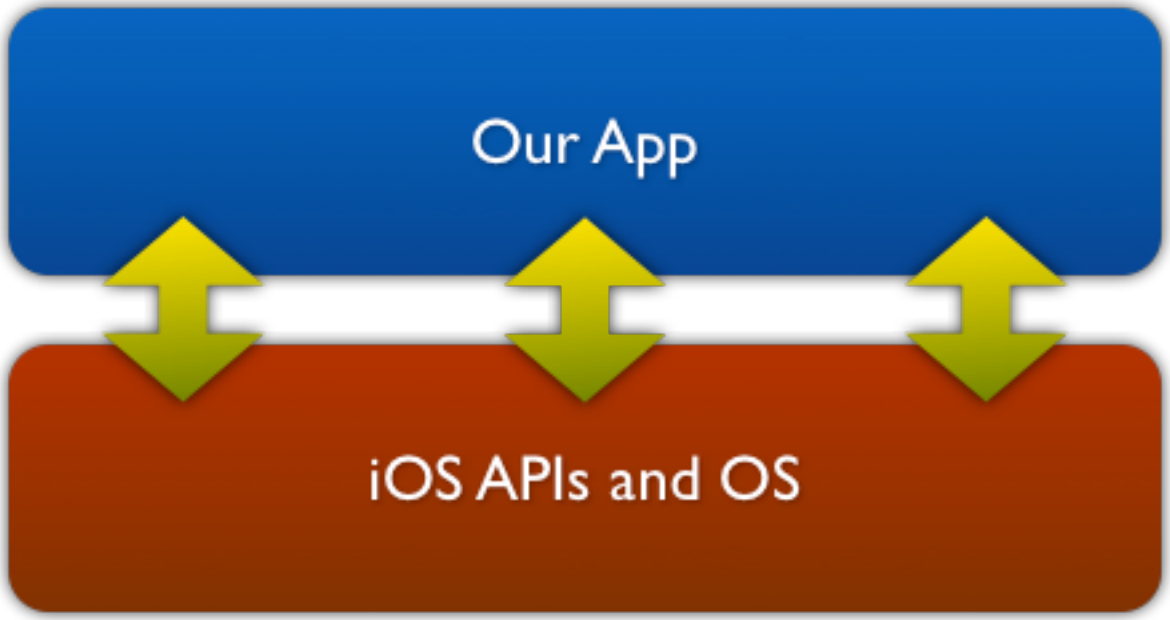
Platform	Language
Android	Java / C++
iOS	Objective C
Blackberry	Java, some Android support
Windows Phone	C#
webOS	C/C++ or HTML/ Javascript

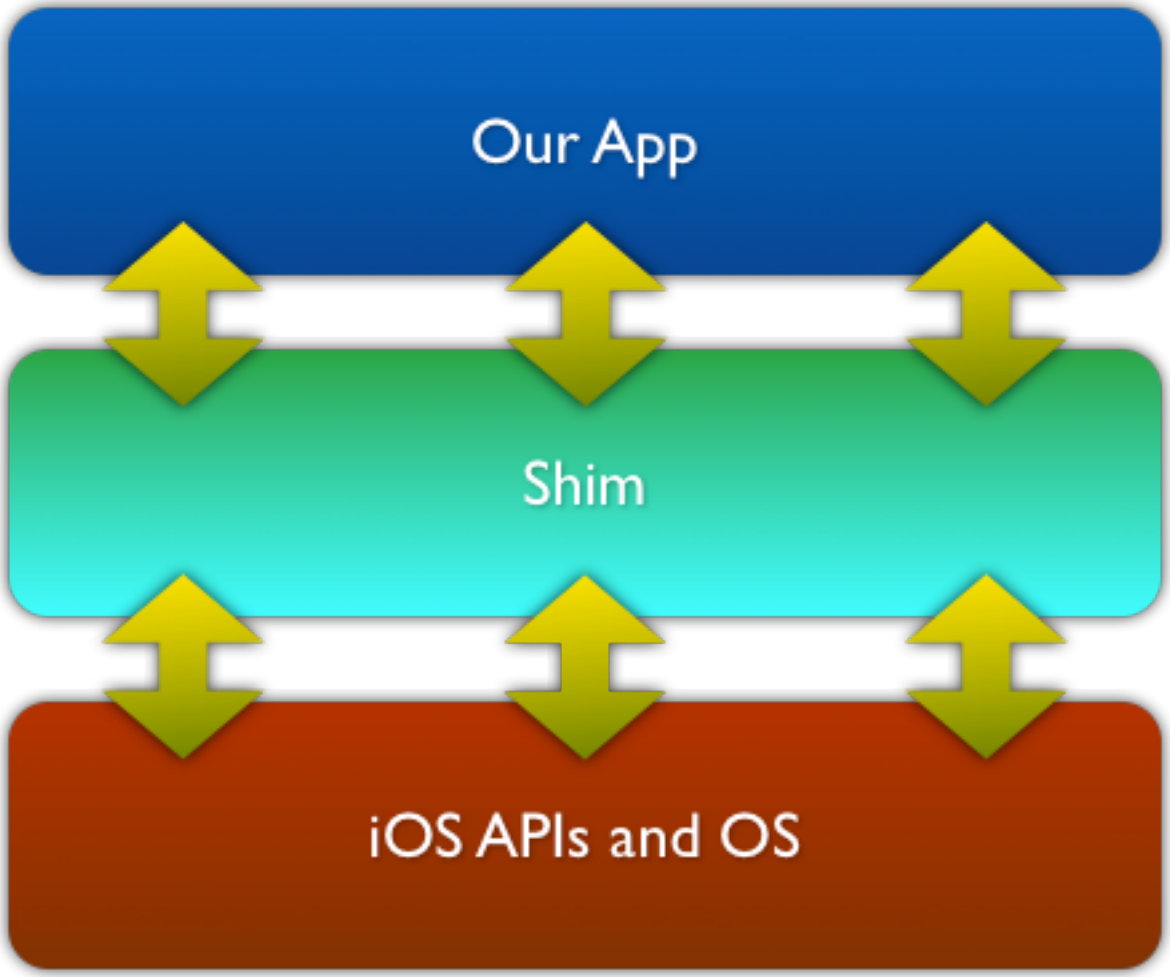
Language

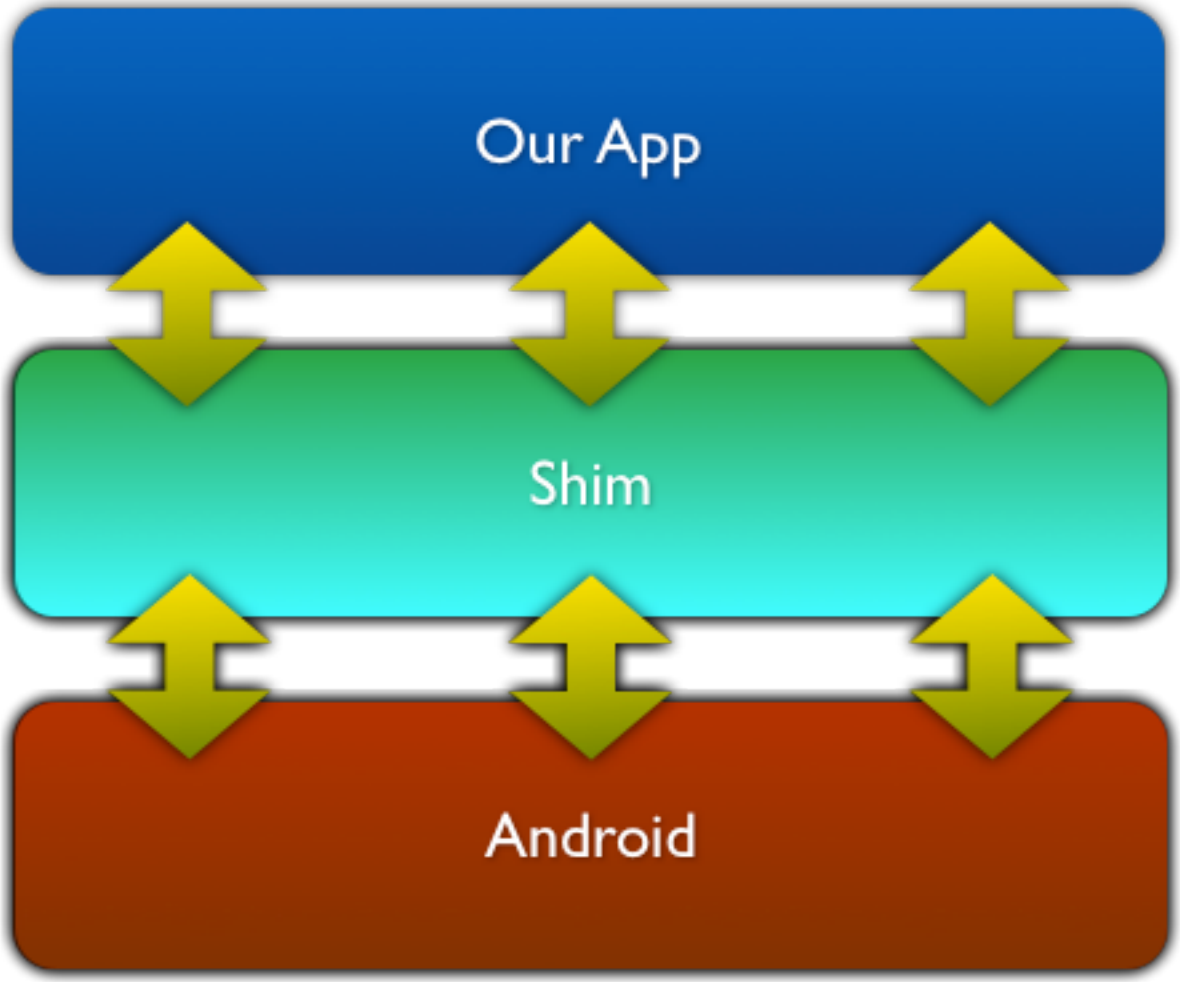
- Compile code for multiple platforms?
  - Can compile Java into native code (gcj)
    - Android supports native code libraries
  - C++ can be compiled to CLR (for WinPhone 7)
- This would work at a technical level
- The code would execute
  - But the app wouldn't run
  - Each platform has different APIs…
    - Android — Activitys, Intents, Services…
    - iOS — Views, ViewControllers,
- Could port the app logic relatively easil;y
  - But would still need to rewrite the UI
  - This maybe enough for games
    - OpenGL ES supported across several platforms
  - End up rewriting boilerplate UI setup stuff

Truly Cross Platform

- Assuming that we can compile code for each device
- To be completely cross platform we can insert a **shim** between our code and the APIs
  - Effectively abstracting our code from the original APIs
  - Our code calls our abstraction
- To port to another device, change the shim and recompile with the appropriate tool-chain
  - New shim provides the same interface to our app
  - But implements it using the native APIs of the new platform


Our App
iOS APIs and OS


Our App
Shim
iOS APIs and OS


Our App
Shim
Android

Adobe AIR

- Developed to let web developers leverage their existing skills to develop desktop apps
- AIR apps can be written in either Flash or HTML+JScript
- Additional libraries allow support for native APIs
  - Windows, widgets, sensors
- App packaged up using Adobe tools
  - Executed by the Adobe AIR runtime
  - Multi-platform support
  - One app — multiple runtimes
- Used for the BBC iPlayer Desktop app and TweetDeck

AIR on mobile

- Android
  - Apps are written using ActionScript
  - Using standard Flash/ActionScript libraries
  - Compiled into a SWF file as normal
    - Then packaged as an AIR app
  - Packaged into a .apk file
    - Need the AIR runtime installed
- iOS
  - Apple forbid the use of VMs on their platform
    - **2.8** Apps that install or launch other executable code will be rejected
  - .swf -> llvm -> ARM native code
  - Linked with Flash runtime written on top of iOS APIs
  - Packaged as a static app executable

Other cross-platform offerings

- Unity
  - Primarily for game development
- Phonegap, Appcelerator, App Furnace
  - Develop application as HTML / Javascript pages
    - Primarily hosted in native web component
  - Integration with native components via a shim

References

- http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html
- http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/DesignPatterns.html
- https://developer.apple.com/appstore/resources/approval/guidelines.html
- http://phonegap.com/
- http://unity3d.com/
- http://appfurnace.com/

Exam

- 1 hr
- Answer 2 of 3 questions
- Worth 50% of your module mark
- 15 marks per question
- “State…”
- “Describe…”
- “Giving an example…”

- Topics we have covered
  - Mobile Device Characteristics
  - Device Architecture
  - Android
    - Application Lifecycle
    - Activities
    - Layout and Widgets
    - Threads and Services / IPC
    - Data Storage and Content Providers
    - Broadcast Receivers
  - Touch
  - Power Management
  - iOS / Cross-platform

Example Question 1

- This question concerns the application components used in an Android application.
  - Name four most significant application components used in an Android application. For each component, describe its role and responsibilities, and give an example of how it might be used in a typical Android application. [12 marks]
  - Android applications are single-threaded by default. Describe the implications of this model when developing an application that must appear to be responsive to the user. [3 marks]

Example Question 2

- This question concerns the Activity Lifecycle.
  - The Android system maintains a stack of Activity objects. Outline the lifecycle of an Android Activity and the methods that are called to inform it of its current state. Describe under what conditions each of these methods would be called and the different stages of the lifecycle. [10 marks]
  - Describe the role of Intent objects in an Android application, and state how one would be used to start a new Activity or a Service. [5 marks]

Example Question 3

- This question concerns data storage in Android.
- The Android SDK provides several mechanisms for an app to store data, either as files or as structured data in an SQLite database. Additionally, an app can share its data with other apps through the use of a ContentProvider.
  - Explain how an app would use the SQLite database library to store structured data. You should outline any classes or objects used, including any that the programmer would need to create, and what the purpose of those classes and objects is. [10 marks]
  - Explain, in general terms, how an app can access data provided by another app within a ContentProvider. [5 marks]