Developing Applications for iOS

Lecture 1:
Mobile Applications Development

Prof. PhD. Radu Ionescu
raducu.ionescu@gmail.com
Faculty of Mathematics and Computer Science
University of Bucharest

Grading System

- Grade options (either one):

  1) 100% individual project

  2) 100% final exam (computer test)*

  (*) + 0.2p per lab attendance (up to 1p)
- In both cases, grade must be greater than 5

Content

- Key concepts of mobile applications development
- Limitations of mobile devices
- Features of mobile devices
- General advices
- Overview of the mobile environments
- Requirements
- iOS Overview
- iOS Technology Layers

Introduction

- Mobile applications development is the process of building software applications for small handheld devices such as mobile phones, personal digital assistants, tablets, etc.

Introduction

- Platforms for mobile applications:

  Android, iOS, Windows Mobile, etc.

- Mobile applications are pre-installed on phones during manufacturing, or downloaded by customers from various mobile software distribution systems:

  App Store (iOS)

  Google Play Store (Android)

  Amazon Appstore (Android)

  Microsoft Store (Windows Mobile), etc.

Key concepts

- Smartphones and tablets are becoming the computer of choice for more and more people.

- Despite the attention paid to mobile development in the last years, a lot of developers still lack the basics when it comes to building mobile applications.

- Many developers are just used to the desktop / web.

Key concepts

- Even if it may seem easy to make an application, it is hard to create a “good user experience”.
- Mobile devices have different limitations and features compared to the desktop computers.
- The emergence of mobile devices and their smaller screens means some serious adjustments in perspective.

Key concepts

- We need to make a transition to a new perspective.

Limitations of mobile devices

- Smaller screen:
- Instead of building for large PC screens (13 to 27 inches wide), developers could be dealing with a 4 to 6 inches wide Android, iPhone or BlackBerry screen.
- Because of the screen size constraint, every pixel counts to some degree.
- Even the iPad's larger screen (7.3 by 9.5 inches) needs to be considered differently because the screen resolution is still less that of most desktop monitors.

Limitations of mobile devices

- Less memory and bandwidth:
- Mobile devices really do not have a lot of memory.
- Although a typical PC can have 8-16 GB of memory, a smartphone might have just 512 MB.

  (e.g.: developers loading 100 images of 10 MB onto a phone would quickly run out of memory)
- Network connectivity for smartphones and tablets incurs limits on downloading.
- Memory, space and battery life are some of the parameters that have to be taken into account when you develop all your apps.

Limitations of mobile devices

- Different user interaction:
- Mobile devices have no mouse. The physical keyboard is much smaller or even missing.
- This means mobile applications don't respond to double clicks or keyboard shortcuts.
- Most smartphones can interact using touch screens or capacitive displays. This can also be a feature.

Features of mobile devices

- Better user interaction:
- Most smartphones can interact using touch screens or capacitive displays.
- Capacitive displays enable the use of multi-touch gestures which allow a natural interaction with the device.

  (e.g.: pinch-open to zoom in, pinch-close to zoom-out, swipe to delete, etc.)

Features of mobile devices

- Using multi-touch gestures

https://www.youtube.com/watch?v=TB5nnMZlZUM

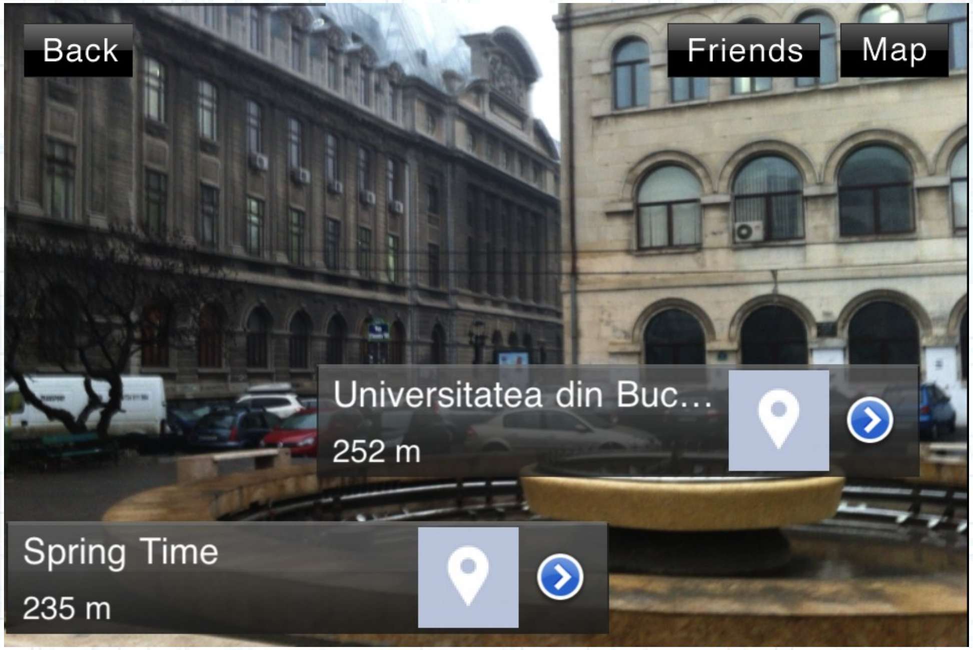
https://www.youtube.com/watch?v=flR6mz788h0

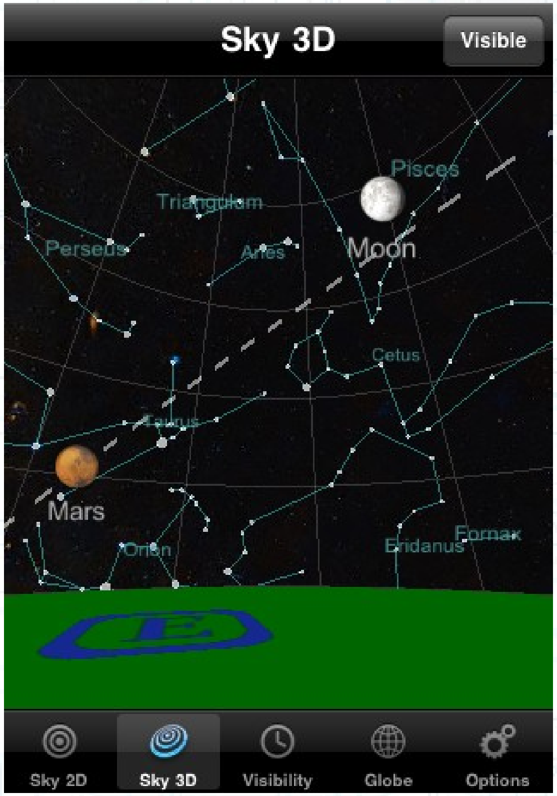
Features of mobile devices

- Using built-in devices:
- Most smartphones have built-in devices such as: camera, accelerometer, gyroscope, GPS, compass, etc.
- Mobile applications should make use of this capabilities whenever this is possible.
- E.g.: detecting the device orientation using the accelerometer (to adjust the display) can be used for creating a better user experience.
- E.g.: building augmented reality applications requires the GPS, the compass, the camera and even the accelerometer.

Features of mobile devices

- Using built-in devices for mobile applications





General Advices

- Focus on user experience: reduce navigation for users, go with defaults, remember what users did last time.
- Choose carefully between native and web development: web-based development is less expensive and not as complex, but it doesn't deliver the kind of experience the user might expect.
- Think about how to take advantage of location: location services enable developers to offer a more customized experience.

General Advices

- Design and code for touch interfaces: developers need to understand the user flows first, then translate the basis of touch interfaces into coding language.
- Expect users to make mistakes: developers should anticipate users pressing the wrong buttons.
- Smaller size of smartphones and unfamiliar users guarantee input mistakes. Mobile applications should be more tolerant and recover without extra effort.

“Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple. But it’s worth it in the end because once you get there, you can move mountains.” - Steve Jobs

Overview of the mobile environments

- Each of the platforms for mobile applications has an IDE which provides tools to allow a developer to write, test and deploy applications into the target platform environment.

- An alternative to native applications are web-based mobile applications which are less expensive to build. This alternative represents a trade-off between cost and user experience, e.g. we will not be able to use all device capabilities.

Android

- Developers can use the Android Studio IDE to build applications using the Kotlin or Java programming languages.
- Android is based on a Linux kernel with libraries and APIs written in C.
- There are more than over 1 million apps available for Android, that can be downloaded from online stores such as Google Play Store.

Windows Phone

- Developers can build applications with Visual Studio 2010 IDE using the C# programming language.
- Windows Mobile is the successor of Windows Phone. It's a newer mobile operating system compared to Android and iOS.
- The applications are available in the Microsoft Store.

iOS

- Integrated with Xcode IDE. Developers must have Intel-based Mac computers and the latest Mac OS X installed.
- iOS applications can be developed using an open-source programming language, called Swift. This is a modern OOP language designed to be more concise than Objective-C.
- iOS is based on a UNIX kernel with libraries written in C, Objective-C and Swift.

Requirements

- Must have an Intel-based Mac with MacOS 10.11.5 or later and Xcode 11.3.1.
- Hardware:

  iPhone 4 or later, iPod Touch 4th Generation or later, iPad 2 or later
- Textbook:

  Apple online documentation

  https://developer.apple.com/develop/
- Prerequisites:

  Object-Oriented Programming Principles

Requirements

Object-Oriented Terms:

- Class (description/template for an object)
- Instance (manifestation of a class)
- Method (code invoked on an object)
- Instance Variable (object-specific storage)
- Inheritance (code-sharing mechanism)
- Superclass/Subclass (Inheritance relationships)
- Protocol (non-class-specific method declaration)

What will I learn in this course?

- How to build cool iOS apps:

  Easy to build even for very complex applications.

  Join a vibrant development community.

- Real-life Object-Oriented Programming:

  The heart of Cocoa Touch is 100% object-oriented.

  Application of MVC design model.

- Many computer science concepts applied in a commercial development platform: Databases, Graphics, Multimedia, Multithreading, Animation, Networking and much more.

- We want you to be able to go on and sell products on the AppStore.

iOS Overview

- iOS comprises the operating system and technologies that you use to run applications natively on devices, such as iPad, iPhone, and iPod Touch.
- Although it shares a common heritage and many underlying technologies with Mac OS X, iOS was designed to meet the needs of a mobile environment, where users' needs are slightly different.
- Some technologies are available only on iOS, such as the Multi-Touch interface and accelerometer support.

iOS SDK Overview

- The iOS SDK contains the code, information, and tools you need to develop, test, run, debug, and tune applications for iOS.
- Xcode provides the launching point for testing your applications on an iOS device, and in iOS Simulator.
- iOS Simulator is a platform that mimics the basic iOS environment but runs on your local Macintosh computer.

Platform Components

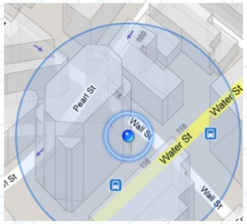
- Tools

- Language

```
label.textColor = UIColor.blueColor()
```

- Frameworks

Foundation



Map Kit

Core Data

Core Motion

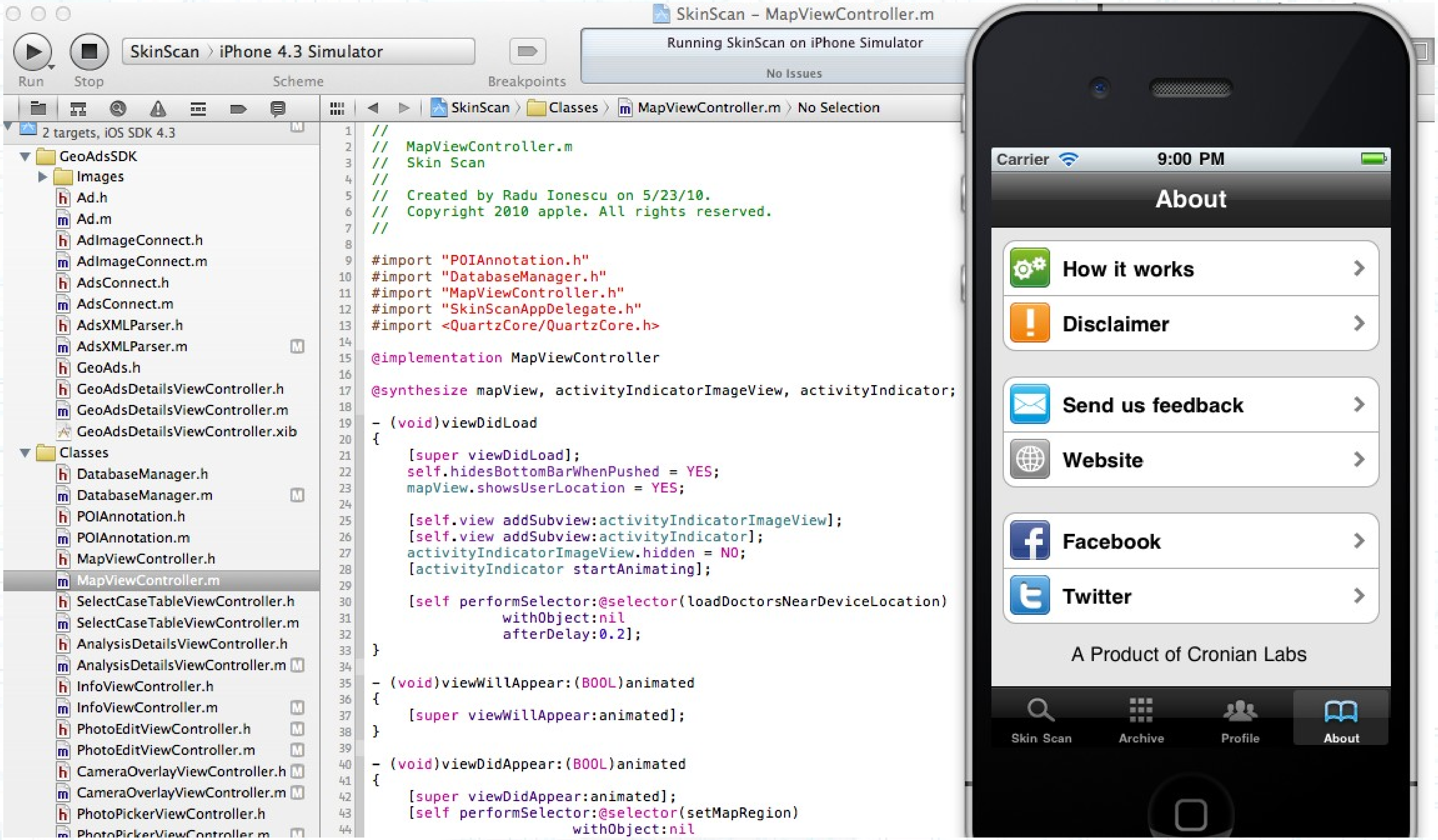
UI Kit

- Design Strategies

MVC

iOS SDK Overview

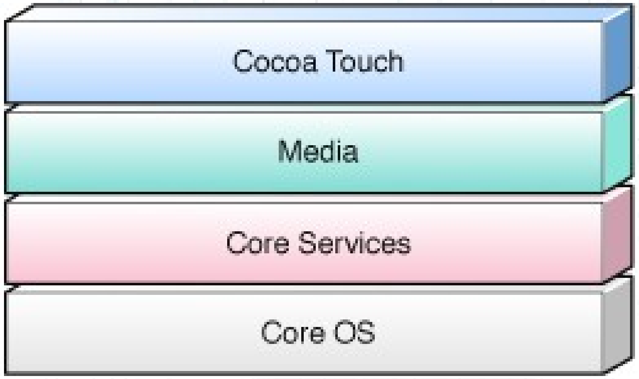
- Xcode and iOS Simulator:


SkinScan – MapViewController.m
Run
Stop
SkinScan 〉 iPhone 4.3 Simulator
Scheme
Breakpoints
Running SkinScan on iPhone Simulator
No Issues
SkinScan 〉 Classes 〉 MapViewController.m 〉 No Selection
2 targets, iOS SDK 4.3
GeoAdsSDK
Images
Ad.h
Ad.m
AdImageConnect.h
AdImageConnect.m
AdsConnect.h
AdsConnect.m
AdsXMLParser.h
AdsXMLParser.m
GeoAds.h
GeoAdsDetailsViewController.h
GeoAdsDetailsViewController.m
GeoAdsDetailsViewController.xib
Classes
DatabaseManager.h
DatabaseManager.m
POIAnnotation.h
POIAnnotation.m
MapViewController.h
MapViewController.m
SelectCaseTableViewController.h
SelectCaseTableViewController.m
AnalysisDetailsViewController.h
AnalysisDetailsViewController.m
InfoViewController.h
InfoViewController.m
PhotoEditViewController.h
PhotoEditViewController.m
CameraOverlayViewController.h
CameraOverlayViewController.m
PhotoPickerViewController.h
PhotoPickerViewController.m
Carrier
9:00 PM
About
How it works
Disclaimer
Send us feedback
Website
Facebook
Twitter
A Product of Cronian Labs
Skin Scan
Archive
Profile
About

```
//
//  MapViewController.m
//  Skin Scan
//
//  Created by Radu Ionescu on 5/23/10.
//  Copyright 2010 apple. All rights reserved.
//

#import "POIAnnotation.h"
#import "DatabaseManager.h"
#import "MapViewController.h"
#import "SkinScanAppDelegate.h"
#import <QuartzCore/QuartzCore.h>

@implementation MapViewController

@synthesize mapView, activityIndicatorImageView, activityIndicator;

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.hidesBottomBarWhenPushed = YES;
    mapView.showsUserLocation = YES;

    [self.view addSubview:activityIndicatorImageView];
    [self.view addSubview:activityIndicator];
    activityIndicatorImageView.hidden = NO;
    [activityIndicator startAnimating];

    [self performSelector:@selector(loadDoctorsNearDeviceLocation)
                withObject:nil
                afterDelay:0.2];
}

- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [self performSelector:@selector(setMapRegion)
                withObject:nil
```

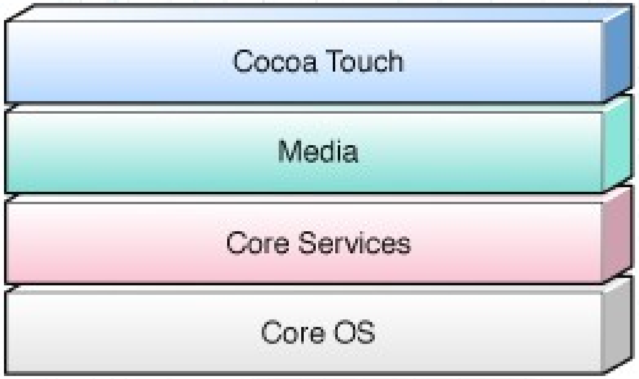
iOS Technology Layers

- The kernel in iOS is based on a variant of the same basic Mach kernel that is found in Mac OS X.
- On top of this UNIX kernel are the layers of services that are used to implement applications on the platform.
- This layering gives you choices when it comes to implementing your code.



iOS Technology Layers

- The Core OS and Core Services layers contain the fundamental interfaces for iOS, including those used for accessing low-level data types, network sockets, and so on.
- On the upper layers you find more advanced technologies. For example, the Media layer contains the fundamental technologies used to support 2D and 3D drawing, audio, and video.



iOS Technology Layers

- Core OS:

OSX Kernel

Mach 3.0

BSD Sockets

POSIX Threads

Security

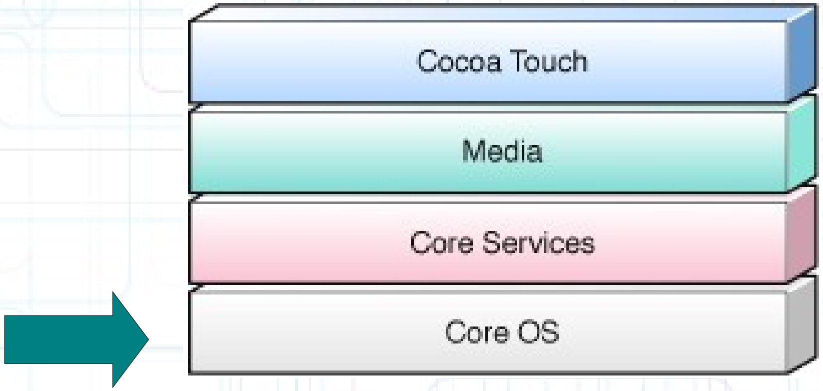
Power Management

Keychain Access

Certificates

File System

Bonjour and DNS Services



Cocoa Touch

Media

Core Services

Core OS

iOS Technology Layers

- Core Services:

Collections

Address Book

Networking

File Access

SQLite

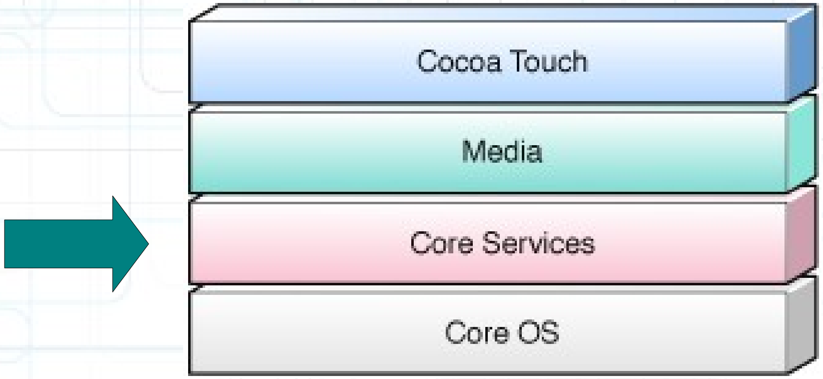
Core Location

Net Services

Threading

Preferences

URL Utilities



Cocoa Touch

Media

Core Services

Core OS

iOS Technology Layers

- Media:

Core Audio

OpenAL

Audio Mixing

Audio Recording

Video Playback

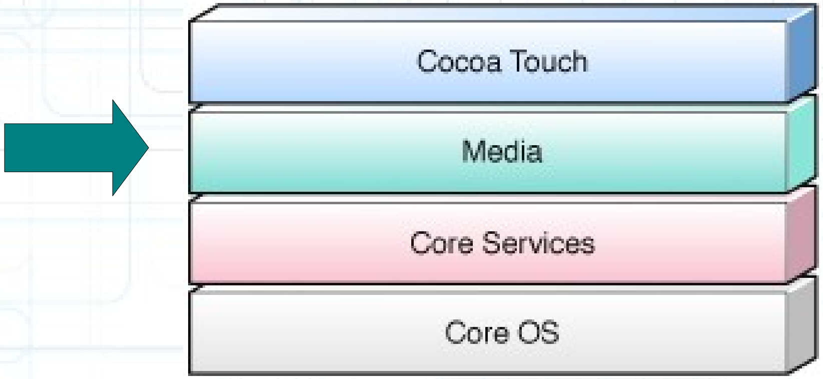
JPEG, PNG, TIFF

PDF

Quartz 2D

Core Animation

OpenGL ES



iOS Technology Layers

- Cocoa Touch:

Multi-Touch

Core Motion

View Hierarchy

Localization

Controls

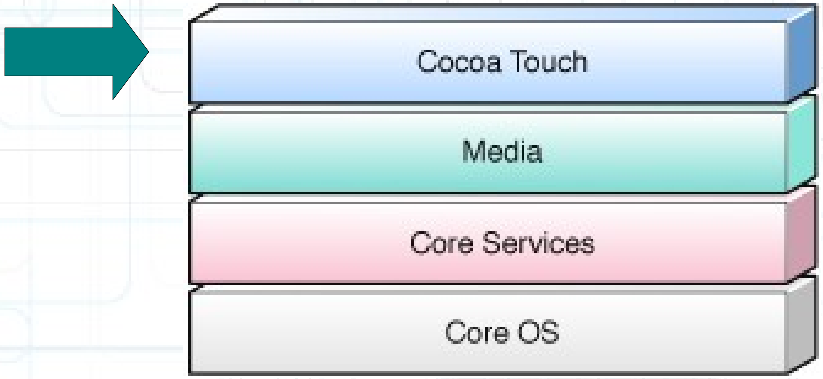
Alerts

Web View

Map Kit

Image Picker

Camera



Cocoa Touch

Media

Core Services

Core OS

Practical Advice

- The starting point for any new project is the Cocoa Touch layer, and the UIKit framework in particular.
- When deciding what additional technologies to use, you should start with frameworks in the higher-level layers.
- The higher-level frameworks make it easy to support standard system behaviors with the least amount of effort on your part.
- You should fall back to the lower-level frameworks only if you want to implement custom behavior that is not provided at a higher level.

Next Time

- MVC Design Concept
- Introduction to Swift