Praktikum Entwicklung von Mediensystemen mit iOS

Wintersemester 2013/2014

Christian Weiß, Dr. Alexander De Luca

Today

- Organization
- Introduction to iOS programming
- Hello World
- Assignment 1

Organization

- 6 ECTS
- Bachelor: Vertiefendes Thema
- Master: Gruppenpraktikum
- Wednesday 12 - 14, Geschwister-Scholl-Platz 1 M207
- Check your emails (cip / campus)
- http://www.medien.ifi.lmu.de/lehre/ws1314/pem/

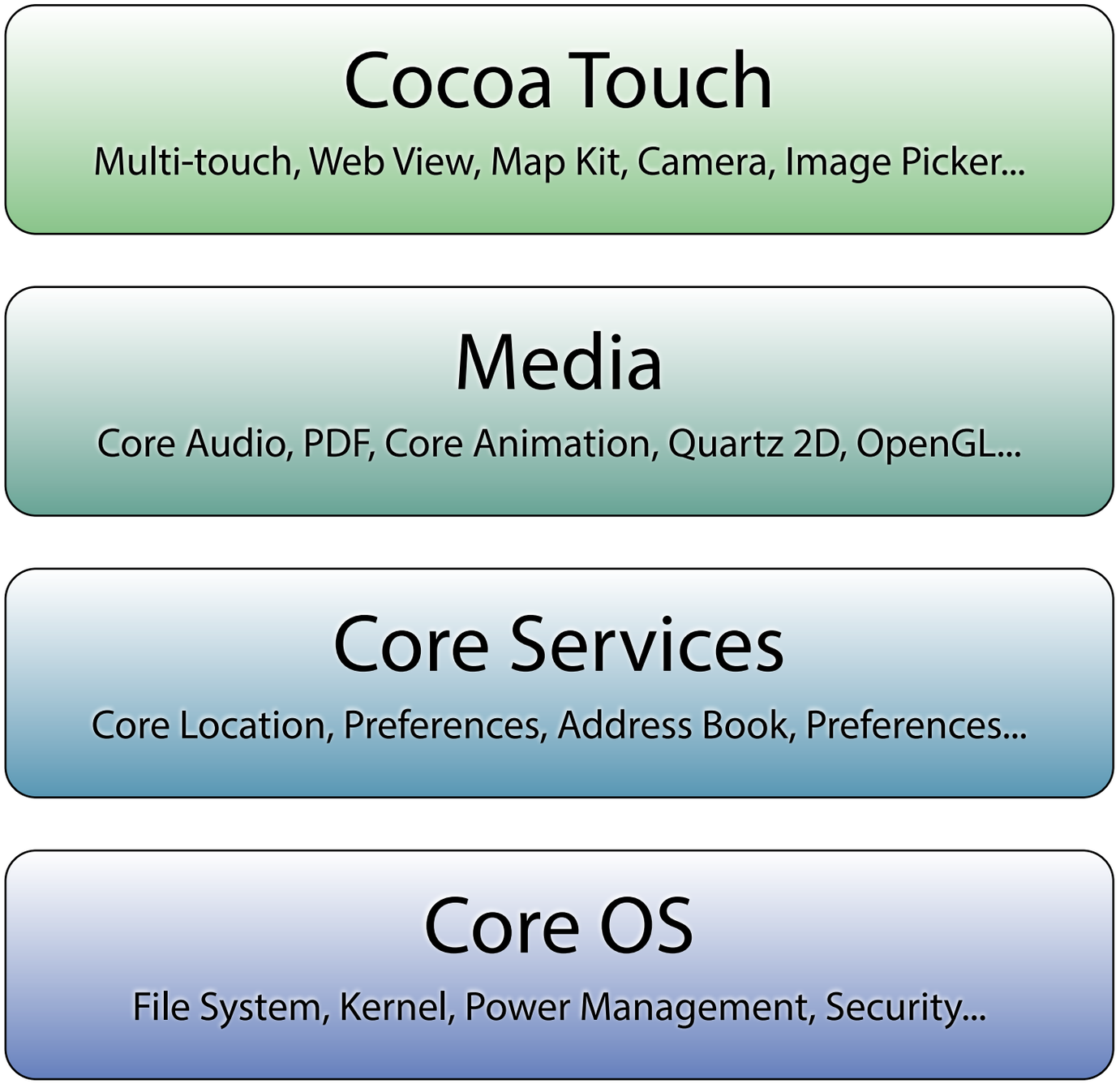
Roadmap

- October, November: weekly lectures and individual assignments
- November, December, January: app development in teams, 4 milestone presentations
- January: final presentation and closing meetings for each team

iOS

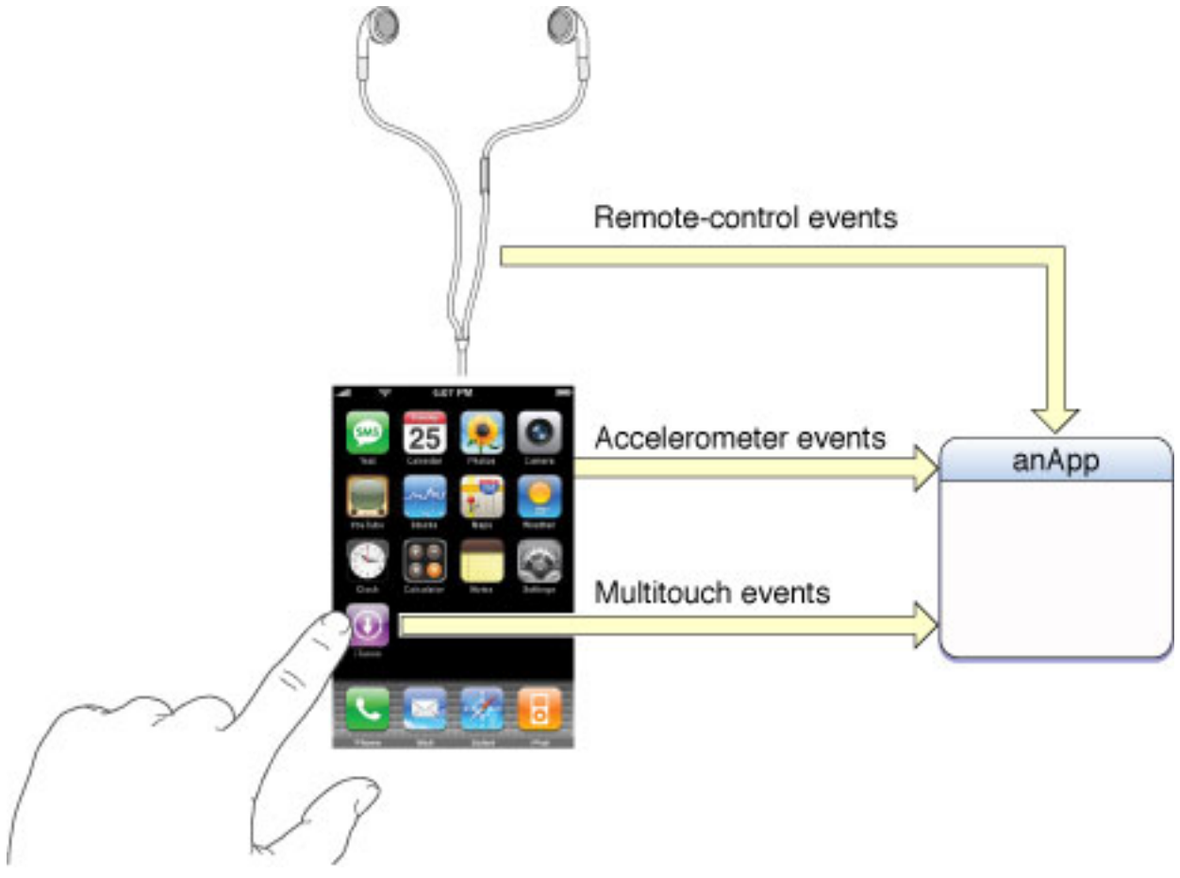
- Mobile operating system by Apple for iPhone, iPad and iPod Touch
- Based on Unix, derived from OS X
- Latest release: iOS 7 (September 2013)
- High market share, high user engagement, high willingness to pay for apps.
- Overall smartphone / tablet market is huge and still growing, and many PEM skills also apply to Android development.

Layers of iOS



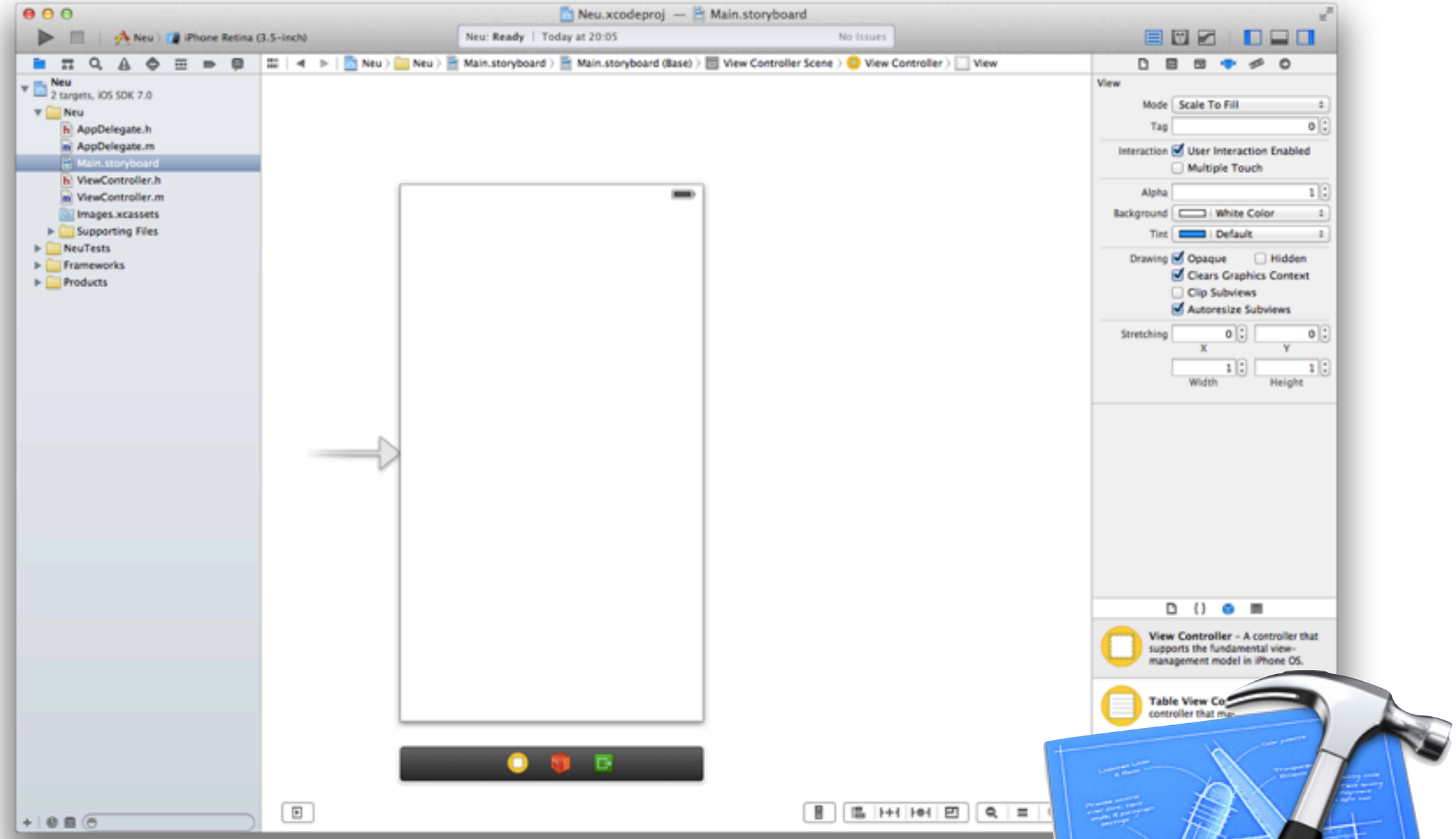
User input

- GUI controls: buttons, sliders, switches etc.
- Multi-touch gestures: tap, pinch, rotate, swipe, pan, long press
- Accelerometer: shaking, rotating



iOS Development

Development Environment



XCode

XCode

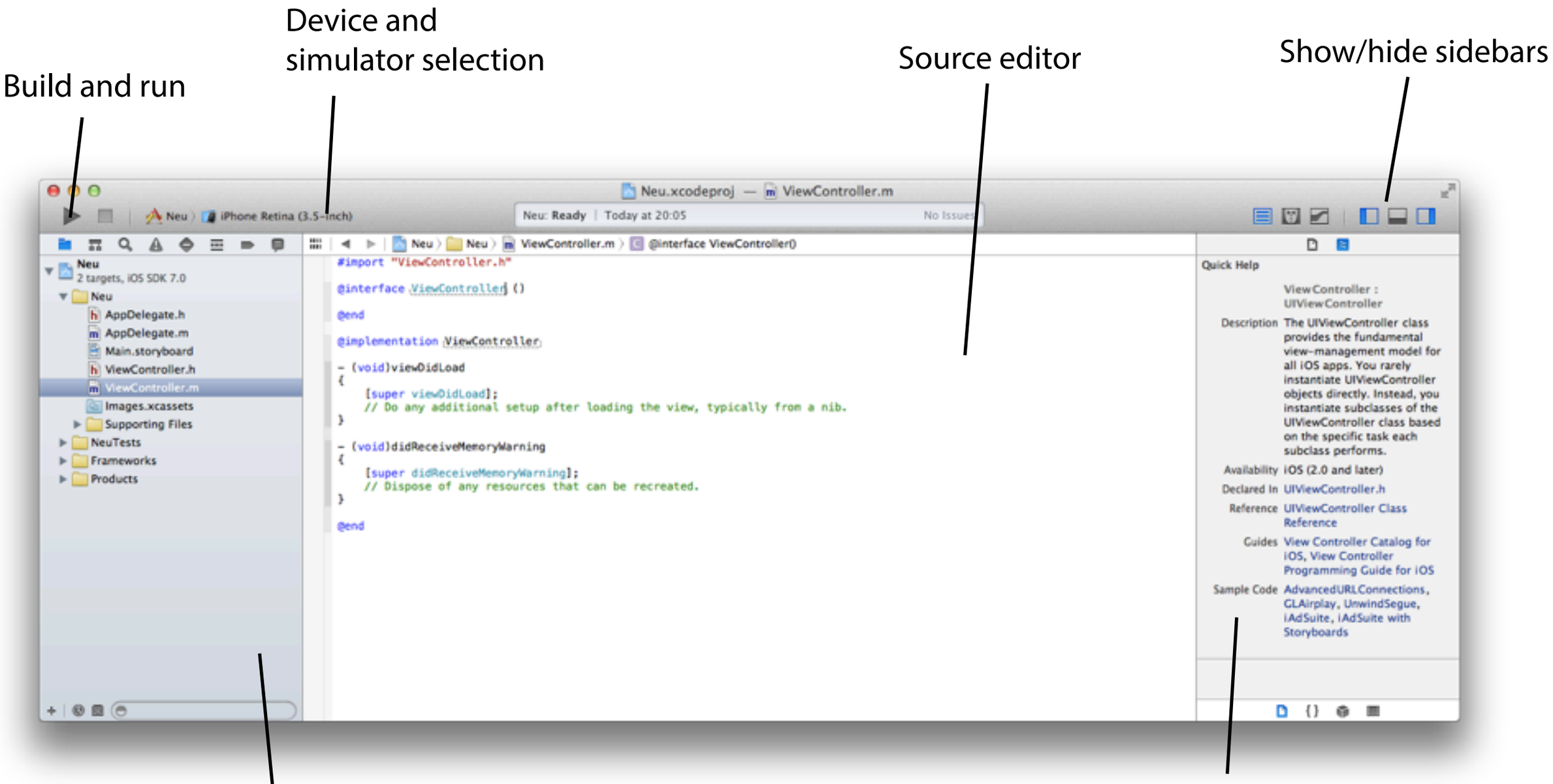
- Source editor: code completion, syntax highlighting, context-sensitive information

- Interface builder: UI elements library and inspector, split editor to connect UI with code, Storyboards

- Compiler: C, C++, Objective-C
- iOS Simulator: run and test apps on a Mac

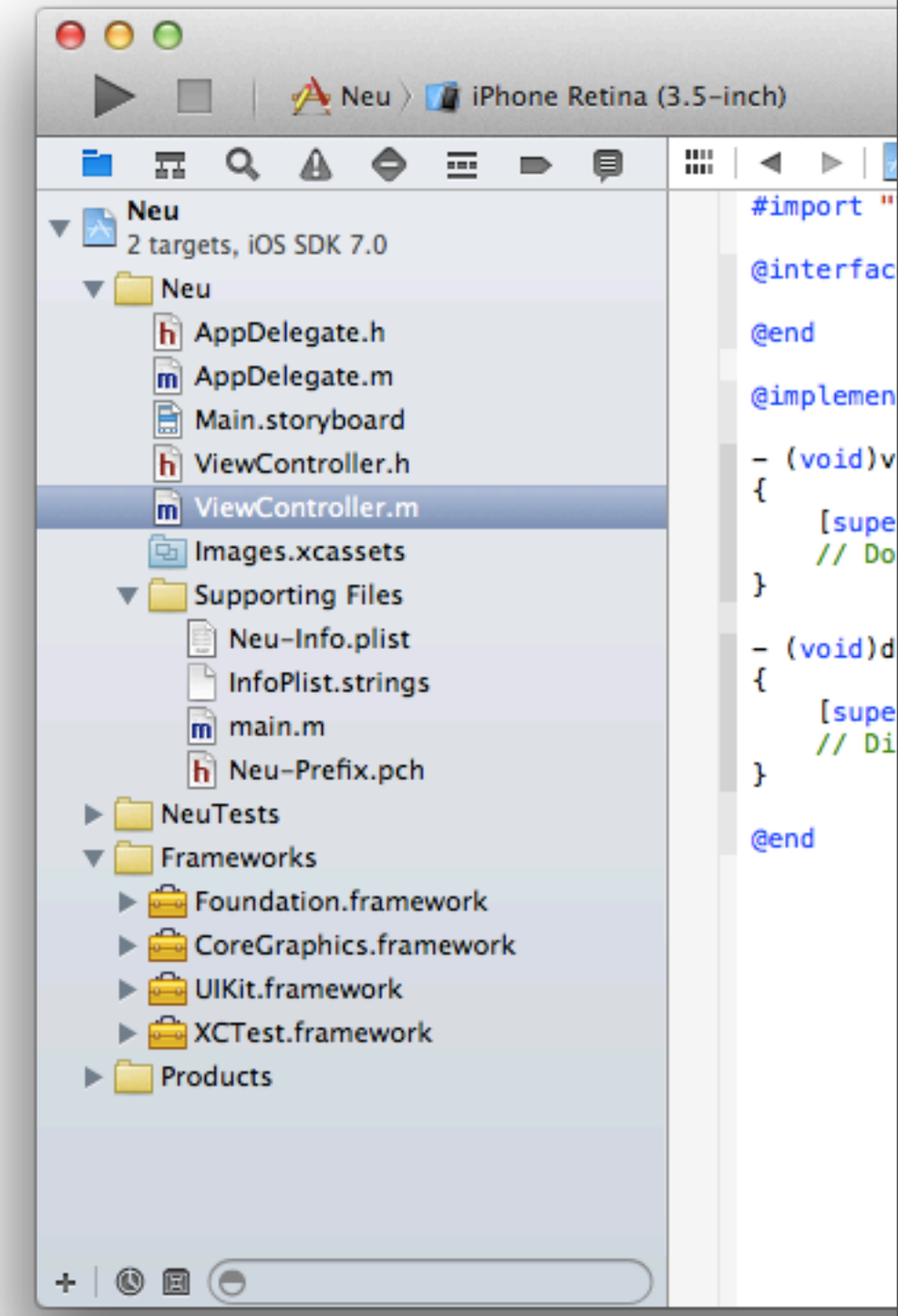
- More: refactoring, version control, debugging, analysis (https://developer.apple.com/technologies/tools/)

XCode


Build and run
Device and simulator selection
Source editor
Show/hide sidebars
File navigator
Utilities sidebar (API info)
Neu.xcodeproj — ViewController.m
Neu: Ready | Today at 20:05
No Issues
Neu 2 targets, iOS SDK 7.0
Neu
AppDelegate.h
AppDelegate.m
Main.storyboard
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
NeuTests
Frameworks
Products
#import "ViewController.h"
@interface ViewController ()
@end
@implementation ViewController
- (void)viewDidLoad
{
[super viewDidLoad];
// Do any additional setup after loading the view, typically from a nib.
}
- (void)didReceiveMemoryWarning
{
[super didReceiveMemoryWarning];
// Dispose of any resources that can be recreated.
}
@end
Quick Help
ViewController : UIViewController
Description The UIViewController class provides the fundamental view-management model for all iOS apps. You rarely instantiate UIViewController objects directly. Instead, you instantiate subclasses of the UIViewController class based on the specific task each subclass performs.
Availability iOS (2.0 and later)
Declared In UIViewController.h
Reference UIViewController Class Reference
Guides View Controller Catalog for iOS, View Controller Programming Guide for iOS
Sample Code AdvancedURLConnections, GLAirplay, UnwindSegue, iAdSuite, iAdSuite with Storyboards

Contents of an XCode project

- Source code files (.h and .m)
- User interface files (.storyboard and .xib)
- Libraries (.framework)
- Resources, e.g. images (.png)
- App configuration file (Info.plist)



Objective-C

- Language for programming iOS and Mac apps, also used by Apple to create much of OS X, iOS, APIs
- Strict superset of C, adds syntax for classes, methods, etc.
- Object-orientated

Short introduction: https://developer.apple.com/library/mac/referencelibrary/GettingStarted/Learning_Objective-C_A_Primer/index.html

Detailed introduction: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

Elements of Objective-C

Java	Objective-C	
MyClass.java	Header.h Implementation.m	
Methods and method calls	Methods and messages	*
Attributes, setters, getters	Properties, instance variables	
Constructor	Initializer	*
Interface	Protocol	*
Garbage Collection	Automatic Reference Counting (ARC)	*

* Different terminology, but for us very similar to writing Java code

Methods

- Definition (in .h):

```
- (void) doSomething;
```

```
- (void) doSomethingWithA: (NSString *) a
andB: (NSString *) b;
```

- Implementation (in .m):

```
- (void) doSomething {
  // do something
}
```

```
- (void) doSomethingWithA: (NSString *) a
andB: (NSString *) b {
    // do something with a and b
}
```

- Method call (“message”) (in .m):


```
[self doSomething];
```

```
NSString* a = @"a";
NSString* b = @"b";
[self doSomethingWithA:a andB:b];
```

Properties

- Auto-creation instance variable, getter and setter
- The getter has the name of the property (”myProperty“)
- The name of the setter is ”get“ + property name (”getMyProperty“)
- Definition (in .h):

```
@property(strong, nonatomic) NSString *name;
```

`strong/weak:` refers to ownership. Always use strong except for properties that point to a parent.

`nonatomic/atomic:` use nonatomic to avoid multi-threading issues.

- Using getters (in .m):

```
NSString *labelText = self.name;
labelText = [self name];
```

- Using setters (in .m):

```
[self setName:@"Max"];
self.name = @"Max";
```

`self.name:` this syntax does NOT access the variable itself. It’s a getter/setter, just like the other syntax.

- Using the instance variable (in .m):

```
_name = @"Max";
```

`_name:` Use this instance variable in custom setters/getters and in init-methods only. In any other case, use the getter/setter.

Instance Variables (“ivars”)

- Like private/protected attributes in Java
- Definition (in .h): `NSString* _name;`
- Use (in .m):

```
_name = @"Max";
labelText = _name;
```

- You don’t have to use the underscore (_), but it’s good practice. Otherwise you accidentally mix up ivars and properties (see next slide).
- Most of the time it is better to use properties instead

Object Initialization

- Object: `MyClass *myObject = [[MyClass alloc] init];`
- Object with parameter: `MyClass *myObject = [[MyClass alloc] initWithParameter: parameter];`
- String:
```
NSString *hello = @"Hello";
NSString *helloWorld = [NSString stringWithFormat:@"%@ World", hello];
```
- Array:
```
NSArray *colors = @[@"Green", @"Red", @"Yellow"];
NSMutableArray *mutableColors = [@[@"Green", @"Red", @"Yellow"] mutableCopy];
```

If your app doesn't work properly, make sure your objects aren't `nil`. THERE ARE NO NULL POINTER EXCEPTIONS - Less crashes, more confusion.

Objective-C - Example

Student.h

```
#import <Foundation/Foundation.h>



@interface Student : NSObject



@end
```

Student.m

```
#import "Student.h"



@implementation Student



@end
```

Objective-C - Example

Student.h

```
#import <Foundation/Foundation.h>



@interface Student : NSObject



@end
```

Student.m

```
#import "Student.h"

@interface student()

@end

@implementation Student



@end
```

Objective-C - Example

Student.h

```objc
#import <Foundation/Foundation.h>



@interface Student : NSObject

@property (strong, nonatomic) NSString *fullName;
@property (nonatomic) NSUInteger number;




@end
```

Student.m

```objc
#import "Student.h"

@interface student()

@end

@implementation Student




@end
```

Objective-C - Example

Student.h

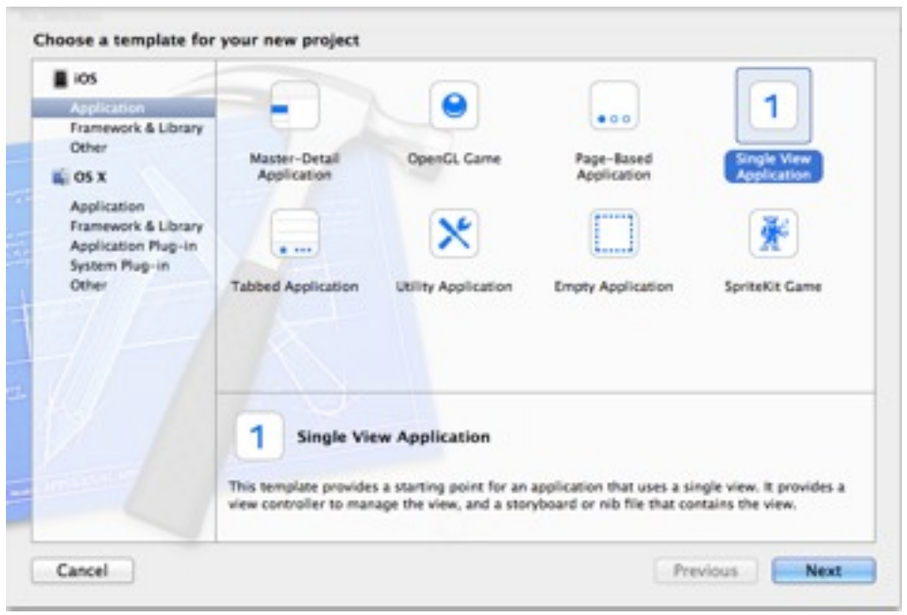
```objc
#import <Foundation/Foundation.h>



@interface Student : NSObject

@property (strong, nonatomic) NSString *fullName;
@property (nonatomic) NSUInteger number;




@end
```

Student.m

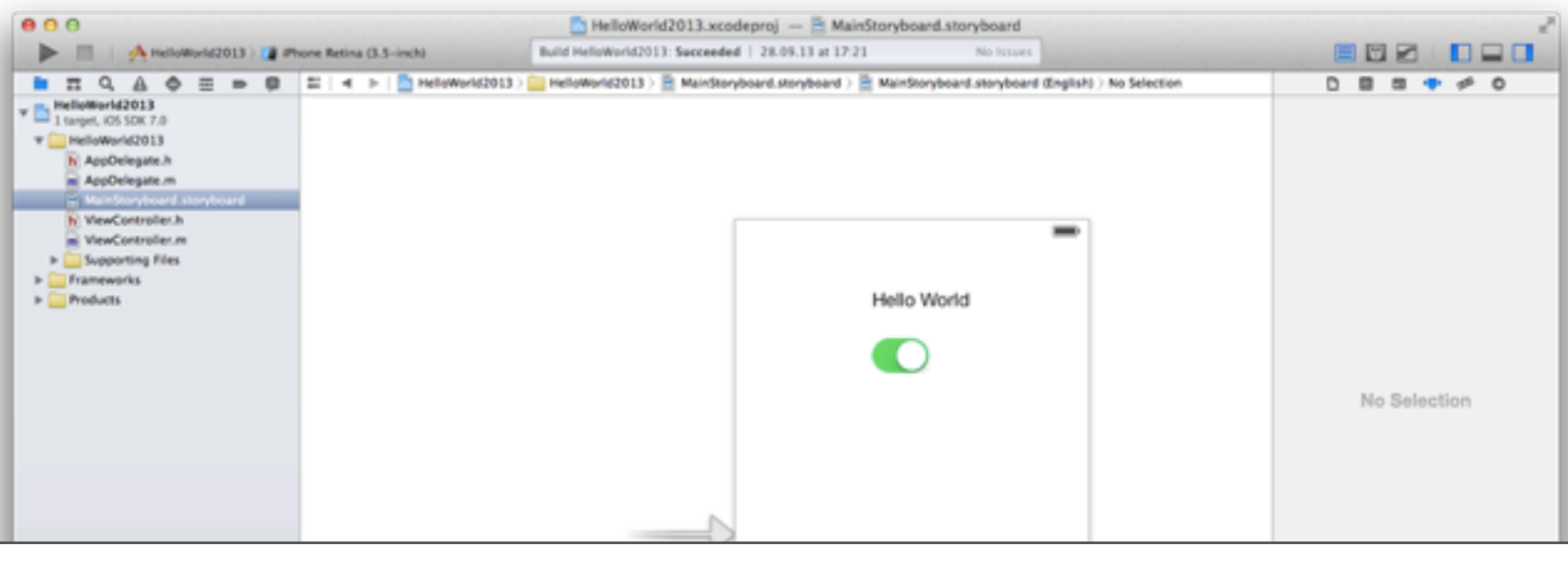
```objc
#import "Student.h"

@interface student()

@end

@implementation Student

- (void) setFullName:(NSString *)fullName
{
    NSLog(@"%@", fullName);
    _fullName = fullName;
}


@end
```

Hello World

- New XCode Project: Single View Application

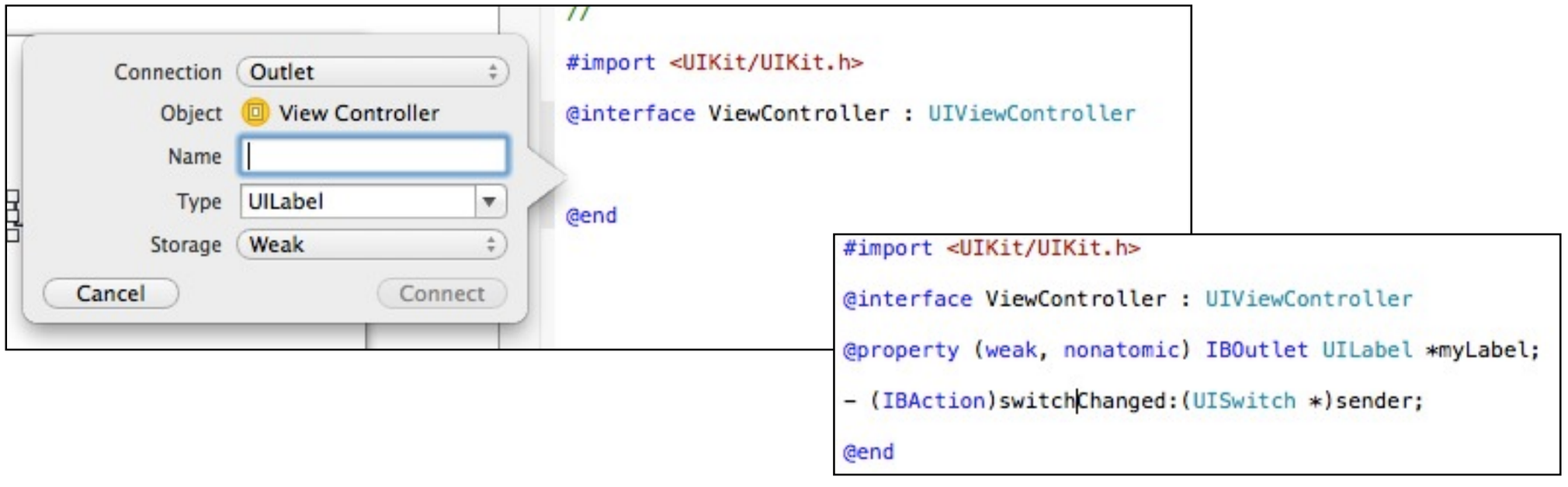


- In the storyboard, drag a text label and a switch onto the screen



Hello World

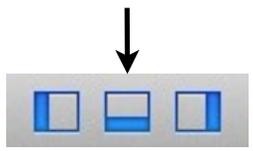
- Open the assistant editor  and ctrl-drag the text label into ViewController.h. Enter a name and click Connect. You now have access to the UI element in your code.
- Again, ctrl-drag the switch into the code. This time, select Action instead of Outlet. Change the type from id to UISwitch. Enter a name and click Connect. You now have a listener method that is called by the OS when the user changes the value of our switch.

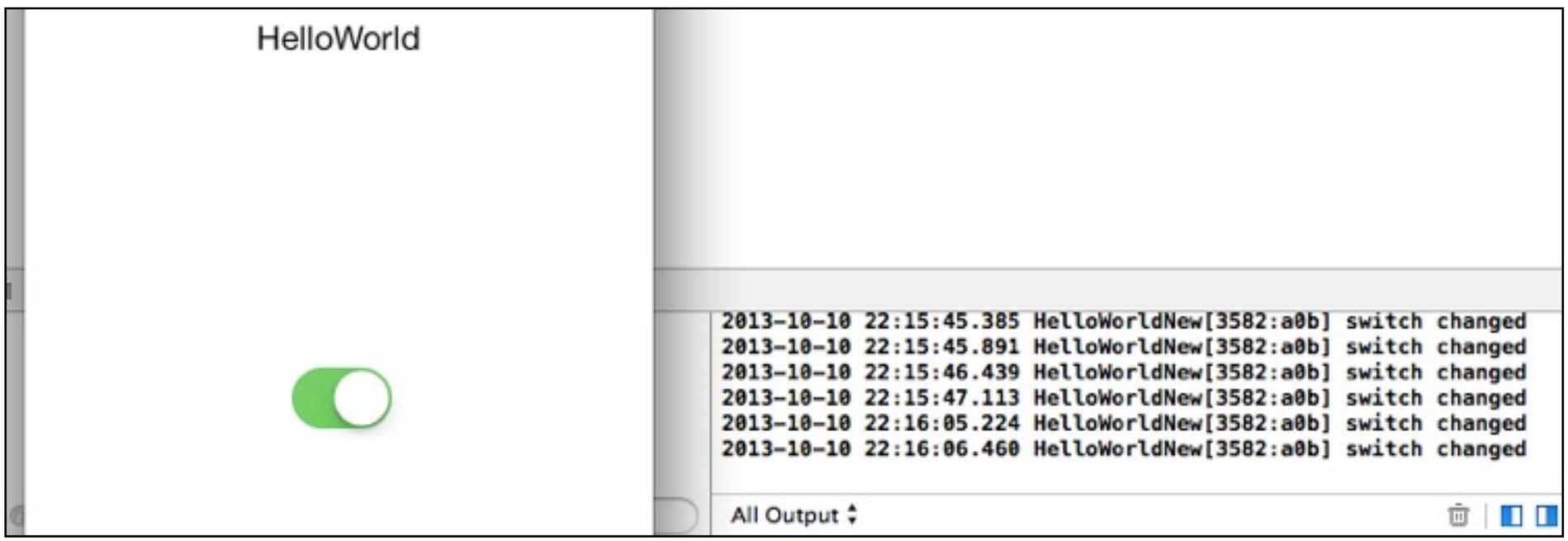


Hello World

- Close the assistant editor and go to ViewController.m. Complete the IBAction method:

```
- (IBAction)switchChanged:(UISwitch *)sender {
    NSLog(@"switch changed");
    if (sender.on) {
        self.myLabel.text = @"HelloWorld";
    } else {
        self.myLabel.text = @"";
    }
}
```

- Open the debug area and run the code. 



UIViewController

- One of the most important classes in iOS programming
- You have to subclass UIViewController when creating a new screen
- Provides methods for managing the view hierarchy throughout its life cycle and for reacting to events (also great for debugging), e.g.

```
– viewDidLoad:
– viewWillAppear:
– viewDidAppear:
– viewWillDisappear:
– viewDidDisappear:
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
duration:(NSTimeInterval)duration;
```

- For more see http://developer.apple.com/library/ios/#documentation/uikit/reference/UIViewController_Class/Reference/Reference.html

App Delegate

- Every app must have an App Delegate.
- Provides methods for managing the app throughout its life cycle (also great for debugging), e.g.

```
- application:didFinishLaunchingWithOptions:
– applicationDidBecomeActive:
– applicationDidEnterBackground:
– applicationWillEnterForeground:
– applicationWillTerminate:
```

- For more see: http://developer.apple.com/library/ios/#documentation/uikit/reference/UIApplicationDelegate_Protocol/Reference/Reference.html
- There are lots of protocols (often named Delegate), e.g. for managing the keyboard, table views, date pickers.

Resources

- Stanford CS 193P iPhone Application Development: https://itunes.apple.com/us/course/coding-together-developing/id593208016
- Official documentation: https://developer.apple.com/library/ios
- Tutorials: http://www.raywenderlich.com/tutorials
- Solutions to specific problems: Google + Stackoverflow
- Book: "iOS Programming: The Big Nerd Ranch Guide" by Joe Conway and Aaron Hillegass
- Developer videos: https://developer.apple.com/videos/

Assignment 1

- Individual assignment
- Get to know XCode and Objective-C
- Due next Wednesday 10:00, upload to Uniworx

- Questions?