Praktikum Entwicklung von Mediensystemen mit iOS

WS 2011

Prof. Dr. Michael Rohs
michael.rohs@ifi.lmu.de
MHCI Lab, LMU München

Today

- Schedule
- Organization
- Introduction to iOS
- Exercise 1

Schedule

- Phase 1 – Individual Phase
  - Introduction to basics about iOS
  - Exercises 1 to 3
  - Each student works on exercises himself/herself
  - Weekly meetings
- Phase 2 – Project Phase
  - Concept and implementation of an iOS application
  - Topic: explanation tools (← proposal only!)
  - Students work in teams
  - Regular milestone meetings
- Phase 3 – Evaluation
  - Evaluate your concept
- Phase 4 – Paper Phase
  - Write up results as a paper
  - (Submit it to a relevant conference)

Topic: Explanation Tools

- Mobile learning / teaching tool
- Mobile software that explains how something works
  - How to use the copier / scanner / fax machine
  - How to repair the tire on the bike
  - How to use a complicated kitchen appliance
  - How a plant grows / develops
- Functionality
  - Showing pictures of object / device
  - Selection of from different perspectives
  - Animations that show how to move / operate the object

Timeline

#	Date	Topic
1	19.10.2011	Introduction and overview of iOS
2	26.10.2011	App architecture, touch input, saving data
3	2.11.2011	Location, networking, sensors
4	16.11.2011	Interviews, Storyboarding; Brainstorming
5	30.11.2011	Paper prototyping test, start of software prototype
6	14.12.2011	Heuristic evaluation of software prototype
7	11.1.2012	Think-aloud user study
8	25.1.2012	Completion of software prototype
9	1.2.2012	Final presentation

Organization

- 4 SWS
- (Bi-)Weekly meetings
  - Thursday 16:00 **s.t.** – 18:00
  - Room 107, Amalienstraße 17
- Homepage:
  - http://www.medien.ifi.lmu.de/lehre/ws1112/pem/

Organization

- For team work
- SVN accounts for each team
  - svn://tracsvn.medien.ifi.lmu.de/repos/pem_team[number] (e.g. svn://tracsvn.medien.ifi.lmu.de/repos/pem_team1)
- Students check in their exercises to their groups' SVN repository

Teams

- Team 5
- Team 6
- Team 7
- Team 8

Technology – SVN

SUBVERSION

Technology – SVN I

- SVN - General
  - Version control system
  - Enables collective editing of shared source code
  - Data stored in a „repository“ which is accessed over the network
  - Editing on local copies of the files
  - Old version available on the server
  - When possible, files will be merged automatically when edited by multiple users at the same time
  - Similar to CVS

Technology – SVN II

- SVN – First Steps (using Tortoise SVN)
  1. Download a SVN Client for Mac OS X
     http://gigaom.com/apple/12-subversion-apps-for-os-x/
  2. SVN command line should be already installed on your Mac
     Utilities → Terminal
  3. Checkout your team repository (creates a local copy of the repository)
     Create an empty folder, open it, right-click and choose „Checkout".
     svn://murx.medien.ifi.lmu.de/team1

Technology – SVN III

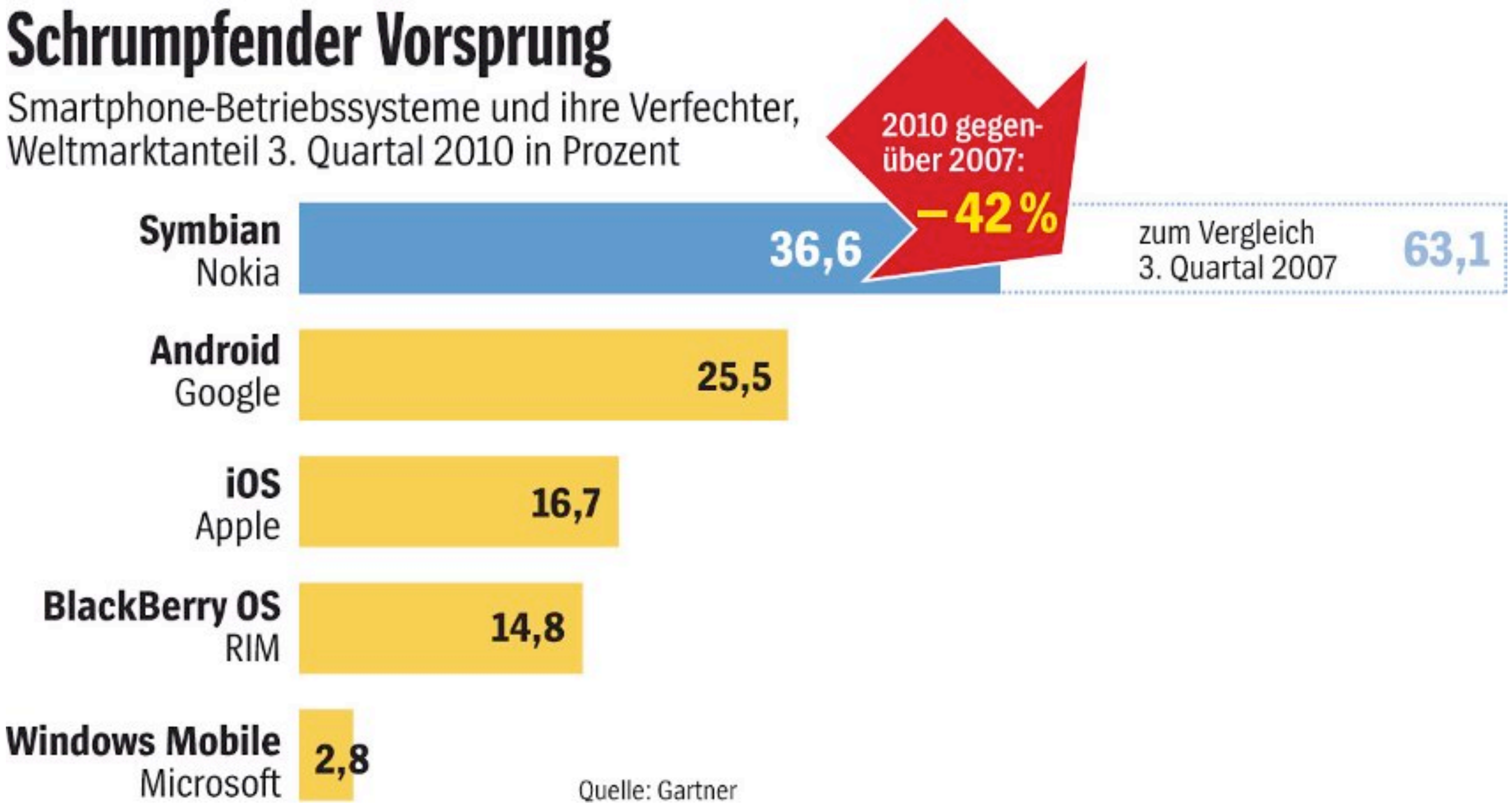
SUBVERSION

- SVN – First Steps (using Tortoise SVN)
  3. Each time you start working perform the "svn update" command
  4. Each time you are done working perform a "svn commit"
  5. Use "svn <command> help" to get help on a command
  6. Use "svn help" to discover new functionality…
  7. Attention: Do not use the OS-functionalities for "delete" or "rename". Use svn commands for this, so that svn is not confused of missing or renamed files. Never ever touch the hidden .svn-Folders.

  For further Information read the German SVN introduction by Richard Atterer, which can be found here: http://www.medien.ifi.lmu.de/fileadmin/mimuc/mmp_ss04/Projektaufgabe/mmp-subversion.pdf

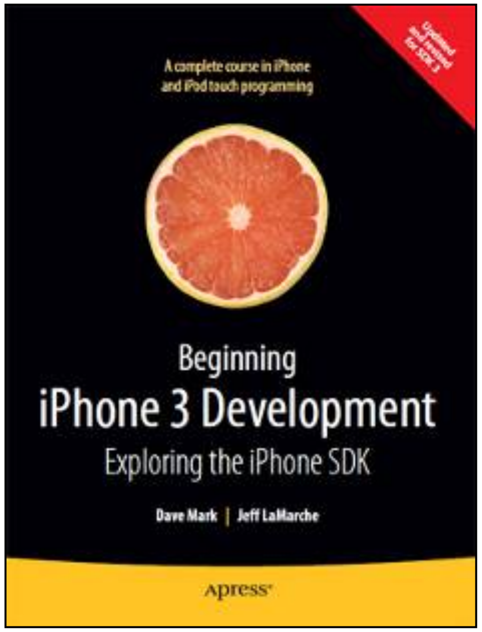
Apple iOS Overview

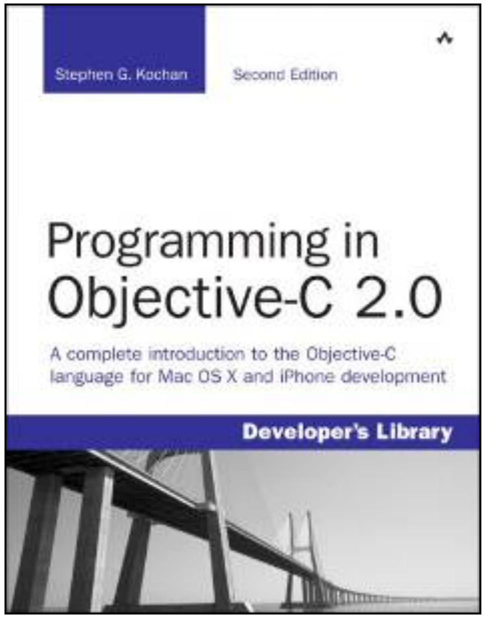
Smartphone Operating Systems



Books

- iPhone development
  - Dave Mark, Jeff LaMarche: Beginning iPhone 3 Development: Exploring the iPhone SDK. Apress, 2009.
  - http://www.amazon.com/Beginning-iPhone-Development-Exploring-SDK/dp/1430224592/
- Objective C
  - Stephen G. Kochan: Programming in Objective-C 2.0. Addison-Wesley, 2nd edition, 2009.
  - http://www.amazon.com/Programming-Objective-C-2-0-Stephen-Kochan/dp/0321566157/


A complete course in iPhone and iPod touch programming
Beginning
iPhone 3 Development
Exploring the iPhone SDK
Dave Mark | Jeff LaMarche
Apress


Stephen G. Kochan
Second Edition
Programming in Objective-C 2.0
A complete introduction to the Objective-C language for Mac OS X and iPhone development
Developer's Library

User Interface Guidelines

- Concrete guidelines for look-and-feel and behavior
  - Visual appearance, e.g., icon design
  - Purpose of user interface elements
  - Layout of user interface elements
  - Behavior, conventions of system features
- iOS Human Interface Guidelines
  - http://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf
  - Aesthetic integrity, consistency, direct manipulation, feedback, metaphors, user control, …

Apple iOS

- Optimized version of Mac OS X
  - New components for handling touch
  - Memory requirement < 0.5 GB
- Hardware
  - 620 MHz ARM 1176 – 1GHz Apple A5
  - 128-512 MB DRAM
  - 4/8/16/32 GB flash RAM
  - Graphics: PowerVR OpenGL ES chip
  - Camera: 2.0-8.0 megapixels
  - Screen: 320x480 pixels, 163 ppi – 640x960 pixels, 326 ppi
  - Connectivity: GSM/UMTS, Wi-Fi (802.11b/g/n), Bluetooth
- SDK available since spring 2008

SDK Options

- Official iPhone SDK
  - Requires Mac to develop (IDE/compiler/debugger only for Mac)
  - Requires registration as developer ($99 per year)
  - Official support
  - Possibility to release on Apple App Store
  - http://developer.apple.com/devcenter/ios/

- iPhone toolchain SDK
  - Unofficial SDK
  - Available for Mac, Linux, PC (with varying comfort)
  - Command line gcc compiler (on-device compiling also possible)
  - All features of the phone actually accessible (even closed ones)
  - Requires “jailbreaking” the phone
  - May be legally questionable
  - http://code.google.com/p/iphone-dev/

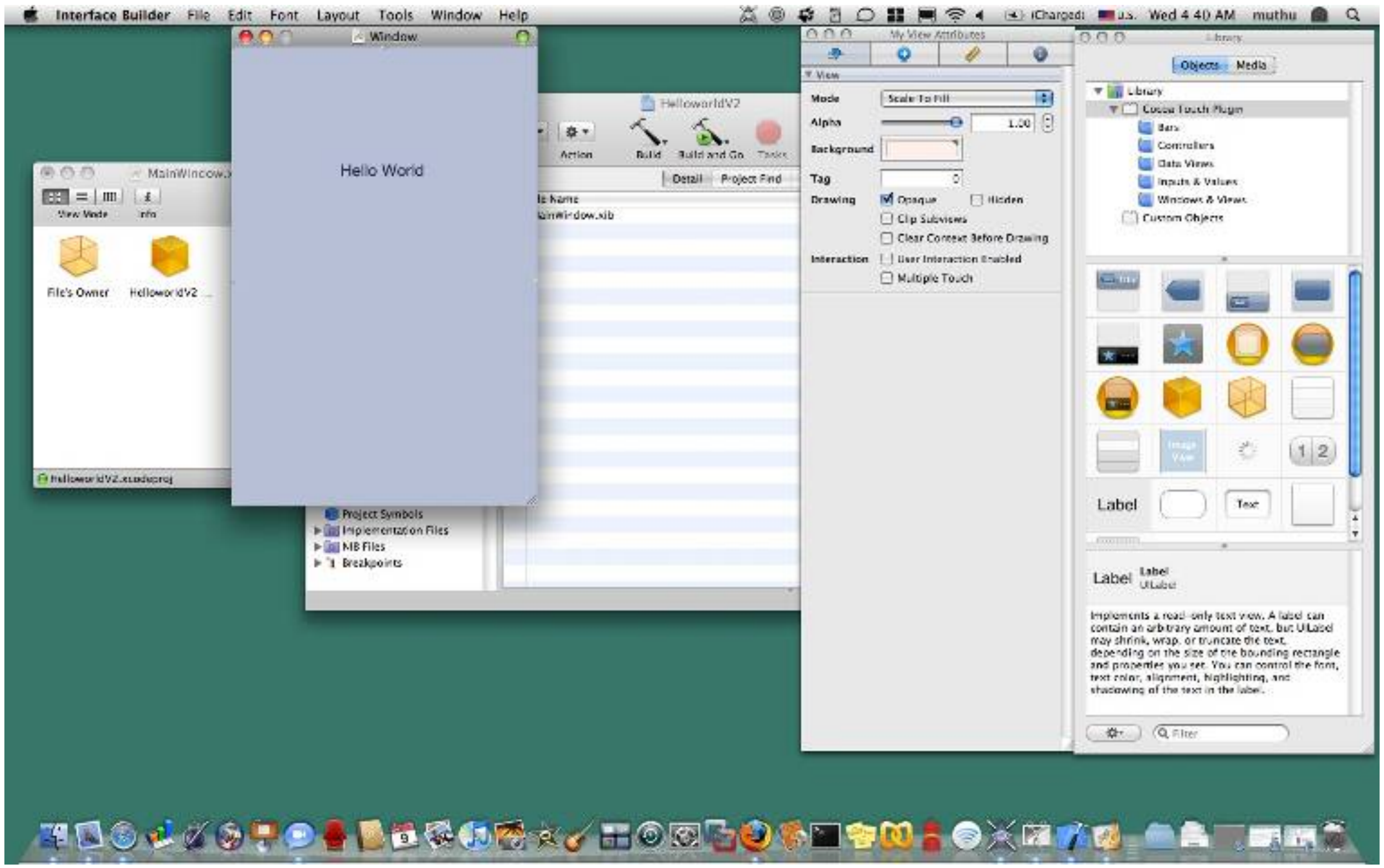
Development Environment

- Xcode: IDE + integrated compiler, run-time debugger

```
while ([[event type]!=NSLeftMouseUp) {
event = [[self window] nextEventMatchingMask:(NSLeftMouseDraggedMask | NSLeftMouseUpMask)];
[self autoscroll:event];
NSPoint currentMouseLocation = [self convertPoint:[event locationInWindow] fromView:nil];

// Figure out a new a selection rectangle based on the mouse location.
NSRect newMarqueeSelectionBounds = NSMakeRect(fmin(originalMouseLocation.x, currentMouseLocation.x), fmin(
    originalMouseLocation.y, currentMouseLocation.y), fabs(currentMouseLocation.x - originalMouseLocation
    currentMouseLocation.y - originalMouseLocation.y));
if (!NSEqualRects(newMarqueeSelectionBounds, _marqueeSelectionBounds)) {

    // Erase the old selection rectangle and draw the new one.
    [self setNeedsDisplayInRect:_marqueeSelectionBounds]
    _marqueeSelectionBounds = newMarqueeSelectionBounds;
    error: syntax error before '_marqueeSelectionBounds'
    [self setNeedsDisplayInRect:_marqueeSelectionBounds];

    // Either select or deselect all of the graphics that intersect the selection rectangle.
    NSIndexSet *indexesOfGraphicsInRubberBand = [self indexesOfGraphicsIntersectingRect:_marqueeSelection
    NSMutableIndexSet *newSelectionIndexes = [oldSelectionIndexes mutableCopy];
    for (NSUInteger index = [indexesOfGraphicsInRubberBand firstIndex]; index!=NSNotFound; index = [
        indexesOfGraphicsInRubberBand indexGreaterThanIndex:index]) {
    if ([newSelectionIndexes containsIndex:index]) {
        [newSelectionIndexes removeIndex:index];
    } else {
        [newSelectionIndexes addIndex:index];
    }
    }
    [self changeSelectionIndexes:newSelectionIndexes];
    [newSelectionIndexes release];
```

Development Environment

- Interface Builder: Graphical interface layouter

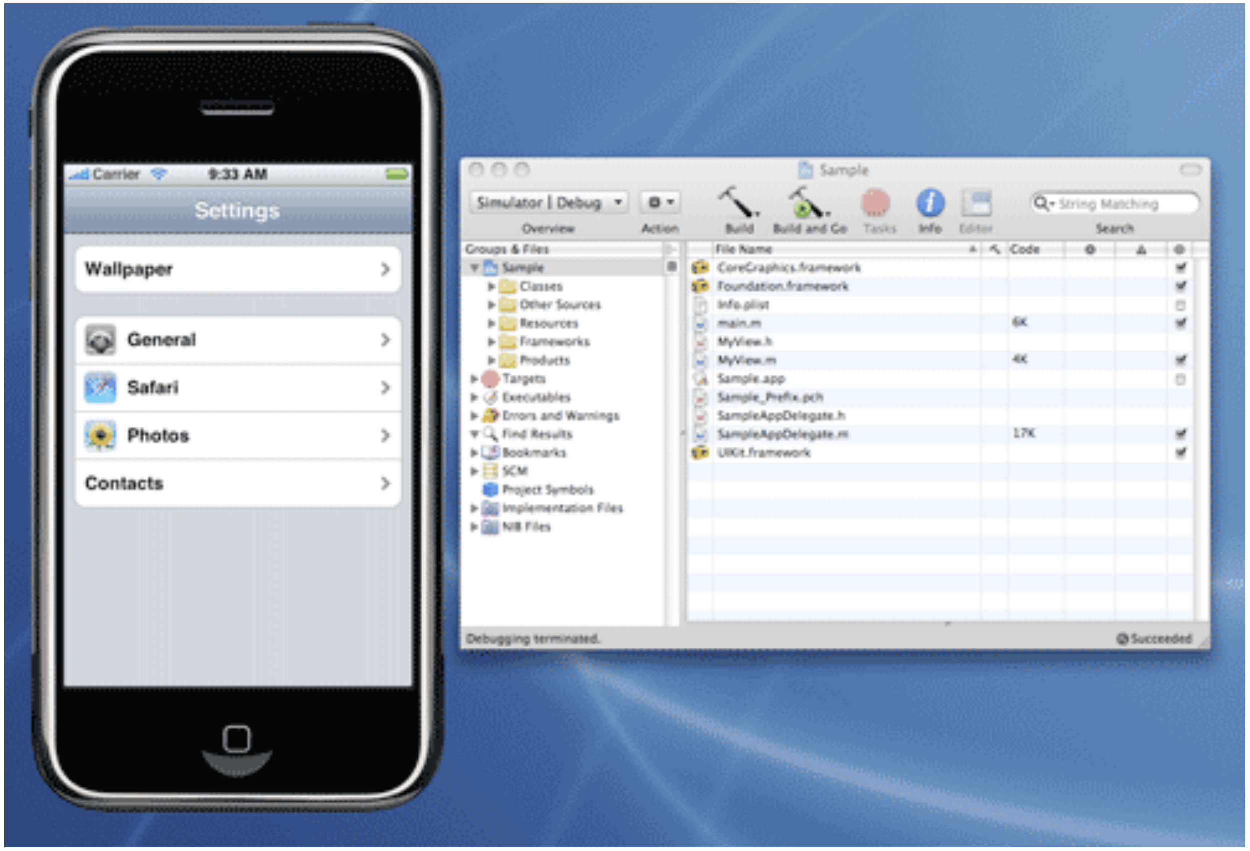


Mode: Scale To Fill
Alpha: 1.00
Background
Tag
Drawing: [✓ Opaque] Hidden
Clip Subviews
Clear Context Before Drawing
Interaction: User Interaction Enabled
Multiple Touch

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the label.

Development Environment

- iPhone Simulator: Mac simulator of iPhone
  - Most features except tilt, simulated multitouch



iOS Technical Background

Philosophy of the API

- Compatibility with Mac OS X
  - Foundation frameworks: shared, Cocoa Touch: iPhone-only
- Maintains general framework structure
- Benefit
  - Shared code development between iPhone and OS X
  - Rapid porting of applications
  - Developer familiarity (for previous Mac developers)
- Preferred language
  - Objective C (implementation language of the SDK)
  - C/C++ work
- Protective
  - Some APIs are privileged and cannot be accessed
  - Example: AudioCore, LayerKit (direct access to framebuffers)

Cocoa Touch Architecture

- Cocoa Touch
  - High level architecture for building iOS applications
- Cocoa Touch consists of:
- UIKit
  - User interface elements
  - Application runtime
  - Event handling
  - Hardware APIs
- Foundation
  - Utility classes
  - Collection classes
  - Object wrappers for system services
  - Subset of Foundation in Cocoa

Objective C

- Objective C is superset of C, with OO constructs
  - Unusual Syntax, rarely used outside Apple realm, inspired by SmallTalk
- General syntax for method calls ("messages"):

  object.method(parameter1, parameter2); becomes:

  [object method:parameter1 parameterkey:parameter2];
- Example

  employee.setSalary(100,20); // arguments base_salary, bonus

  [employee setSalary:100 withBonus:20];
- Learn more at

  developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC

Objective C - Methods

- Method declaration syntax

  ± (type) selector:(type)param paramkey:(type)param2;

  Instance methods: - (void) myInstanceMethod;

  Class methods: + (void) myClassMethod;

- Example

  - (void) setSalary:(int)income withBonus:(int)bonus;

- Basic classes, examples
  – NSObject is root class (basics of memory management)
  – NSString
    - Example: s = [NSString stringWithFormat: @"The answer is: %@", myObject];
    - Constant strings are @"this is a constant string"
  – NSLog(NSString); (NSLog is your friend…)
  – NS… also offers collections (NSArray, NSDictionary etc) and other basic language service functionality
    - Prefix "NS" is derived from OS X predecessor, NextStep

Objective C – Features and Pitfalls

- Dynamically typed objects (or hard to find bugs)
  - id someObject
  - id is generic “pointer” without type (“void*”)
  - introspection allows finding out type at runtime
- Nil object pointers (or how to make really hard to find bugs)

  object = nil;

  [object setProperty: nil];
  - Will send message to nil, hard to find if objects didn’t get proper assignment
- id, nil and dynamic typing enable message-passing paradigm

Memory Management By Hand

- Don’t create memory leaks!
- Object reference life cycle:

```
myobject = [[MyClass alloc] init];          // reference count = 1 after alloc
[myobject retain];        // increment reference count (retainCount == 2)
[myobject release];       // decrement reference count (retainCount == 1)
[myobject release];       // decrement reference count (retainCount == 0)
// at this point myobject is no longer valid, memory has been reclaimed
[myobject someMethod]; // error: this will crash!
```

- Can inspect current reference count:

```
NSLog(@"retainCount = %d", [textField retainCount]);
```

- Can autorelease (system releases at some point in future)

```
[myobject autorelease];
```

  Used when returning objects from methods.

Memory Management By Hand

- Memory rule: You are responsible for objects you allocate or copy (i.e. “allocate” or “copy” is some part of the name)!
- Not responsible:

```
NSData *data = [NSData dataWithContentsOfFile:@"file.dat"];
```

- Responsible:

```
NSData *data = [[NSData alloc] initWithContentsOfFile:@"file.dat"];
```

- Responsible:

```
NSData *data2 = [data copy];
```

- Never release objects you are not responsible for!

Objective C – Practical Aspects

- Based file extension .m
- Header file extension .h (expects Objective-C base file)
- Base file extension for Objective C++ is .mm (not .cpp)
- #import <...> (automatic redundancy check)

Objective C - Class

In .h file:

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

@interface Employee : NSObject
{ //Instance vars here
    NSString *name;
    int salary;
    int bonus;
}
// methods outside curly brackets
-   (void)setSalary:(int)cash withBonus:(int)extra
@end
```

Objective C - Class

In .m file:

```
#import ”Employee.h”

@implementation Employee
-  (void)setSalary:(int)cash withBonus:(int)extra
{
   salary = cash;
   bonus = extra;
}
@end
```

Objective C - Protocols

- Used to simulate multiple inheritance and add functionality on top of existing objects (i.e. for delegates), similar to interfaces in Java:

```
@protocol Locking
- (void)lock;
- (void)unlock;
@end
```

- Denotes that there is an abstract idea of „Locking“
- Classes can state that they implement „Locking“ by declaring the following:

```
@interface SomeClass : SomeSuperClass <Locking>
@end
```

Objective C Properties

- .h file:

```
@interface MyDetailViewController : UIViewController {
        NSString *labelText;
}
@property (nonatomic, retain) NSString *labelText;
@end
```

- .m file:

```
@synthesize labelText;
        -(void)someMethod {
        self.labelText = @"hello";
}
```

creates accessor methods: setLabelText (retains/releases) and getLabelText.

dot-syntax means: use property's setLabelText accessor method, will retain the object

equivalent to [self setLabelText:@"hello"];

Implicit Setter/Getter Accessor Methods

- .h file: @property (nonatomic, retain) NSString *labelText;
- .m file: @synthesize labelText;
- Automatic creation of accessor methods:

```
- (void) setLabelText:(NSString *)newLabelText {
    [labelText release];
    labelText = newLabelText;
    [labelText retain];
}
- (NSString*) getLabelText {
    return labelText;
}
```

decrement reference counter on old object (if any)

increment reference counter on new object (if any)

- Properties are accessible from other classes, data members only if declared @public

Property Attributes

- Writability: readwrite (default), readonly
- Setter semantics: assign, retain, copy
- Atomicity: atomic (default), nonatomic

- “readonly” means only a getter, but no setter accessor method is generated by @synthesize

Selectors

- Methods as arguments (useful for callback methods)
- Example: setting a button callback method
- .h file

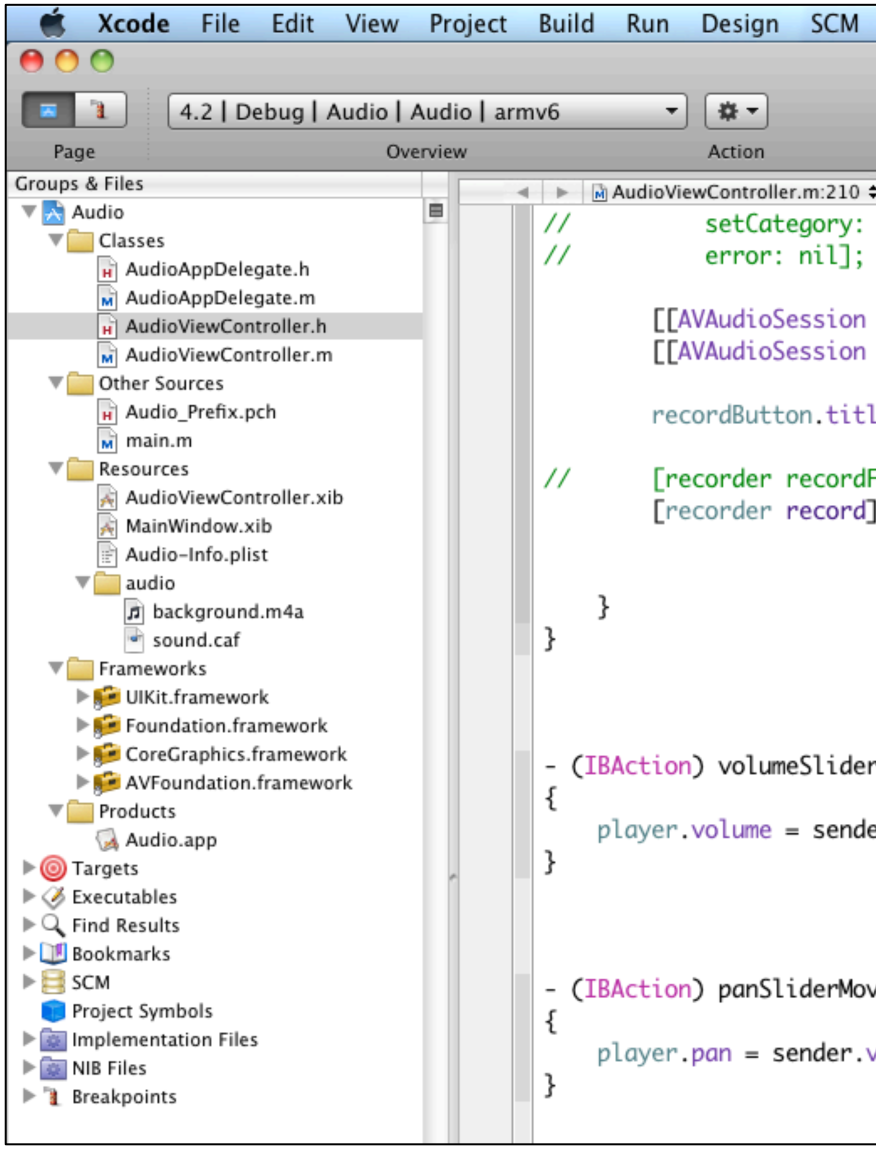
```
@interface MyDetailViewController : UIViewController {
    IBOutlet UIButton *newButton;
}
@property (nonatomic, retain) UIButton *newButton;
- (void) newButtonPressed:(id)source;
```

- .m file

```
- (void)someInitializationMethod {
    [newButton addTarget:self action:@selector(newButtonPressed:)
                         forControlEvents:UIControlEventTouchUpInside];
}
- (void) newButtonPressed:(id)source { NSLog(@"newButtonPressed"); }
```

Contents of an Xcode iPhone Project

- Source files
- Compiled Code
- Framework code
  - E.g. UIKit.framework
- Nib file (extension .xib)
  - Contains interface builder data
- Resources
  - Media (images, icons, sound)
- Info.plist file
  - Application configuration data



```
//          setCategory:
//          error: nil];

        [[AVAudioSession
        [[AVAudioSession

        recordButton.titl

//      [recorder recordF
        [recorder record]

    }
}


- (IBAction) volumeSlider
{
    player.volume = sende
}


- (IBAction) panSliderMov
{
    player.pan = sender.v
}
```

HELLO WORLD

“Hello World” Steps

- Creating a project (“View-based Application”)
- Inspecting package contents
  - Navigator (left pane)
- Inspecting HelloWorldViewController.xib
  - Utilities (right pane)
  - Adding a label and a button
- Adding event handling method to HelloWorldViewController

  ```
  -(IBAction) buttonPressed:(id)sender;
  ```

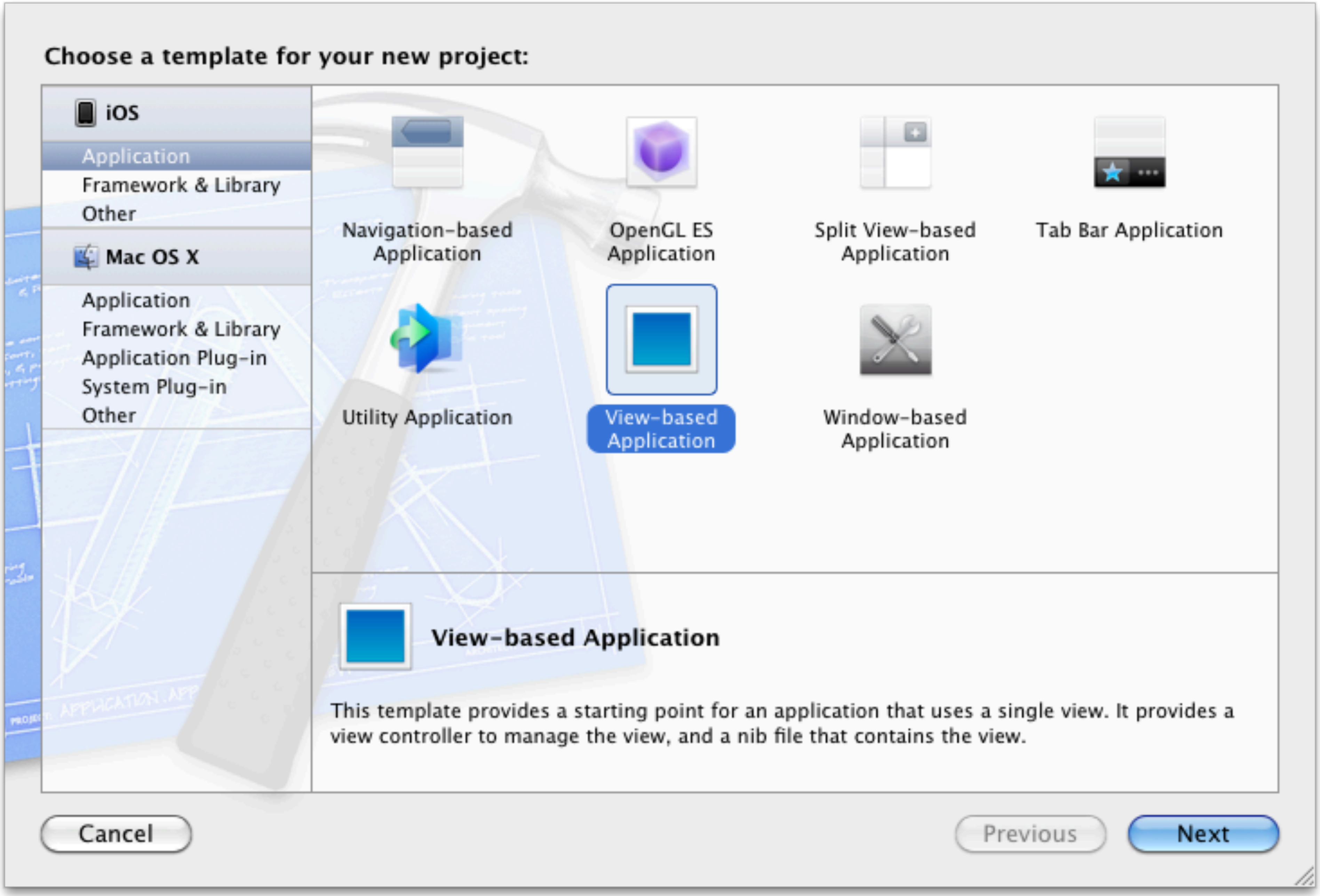
  Log output: `NSLog(@"button pressed");`
- Linking button to event handler in xib file
- Linking button to event handler using @selector

“Hello World” Steps

- Set label text when button was pressed
  - Add label outlet and property in .h file
  - Synthesize label property and set label text in .m file
- Increment counter when button was pressed
  - Add variable in .h file
  - Use NSString stringWithFormat in .m file
- Access label view using a tag (no IBOutlet required)
  - Define tag for label in Interface Builder (e.g. Tag = 100)
  - `UILabel *label = (UILabel*)[self.view viewWithTag:100];`
- Explain what happens during instantiation
- Showing a UIAlertView
- Explain #pragma

“Hello World” Steps

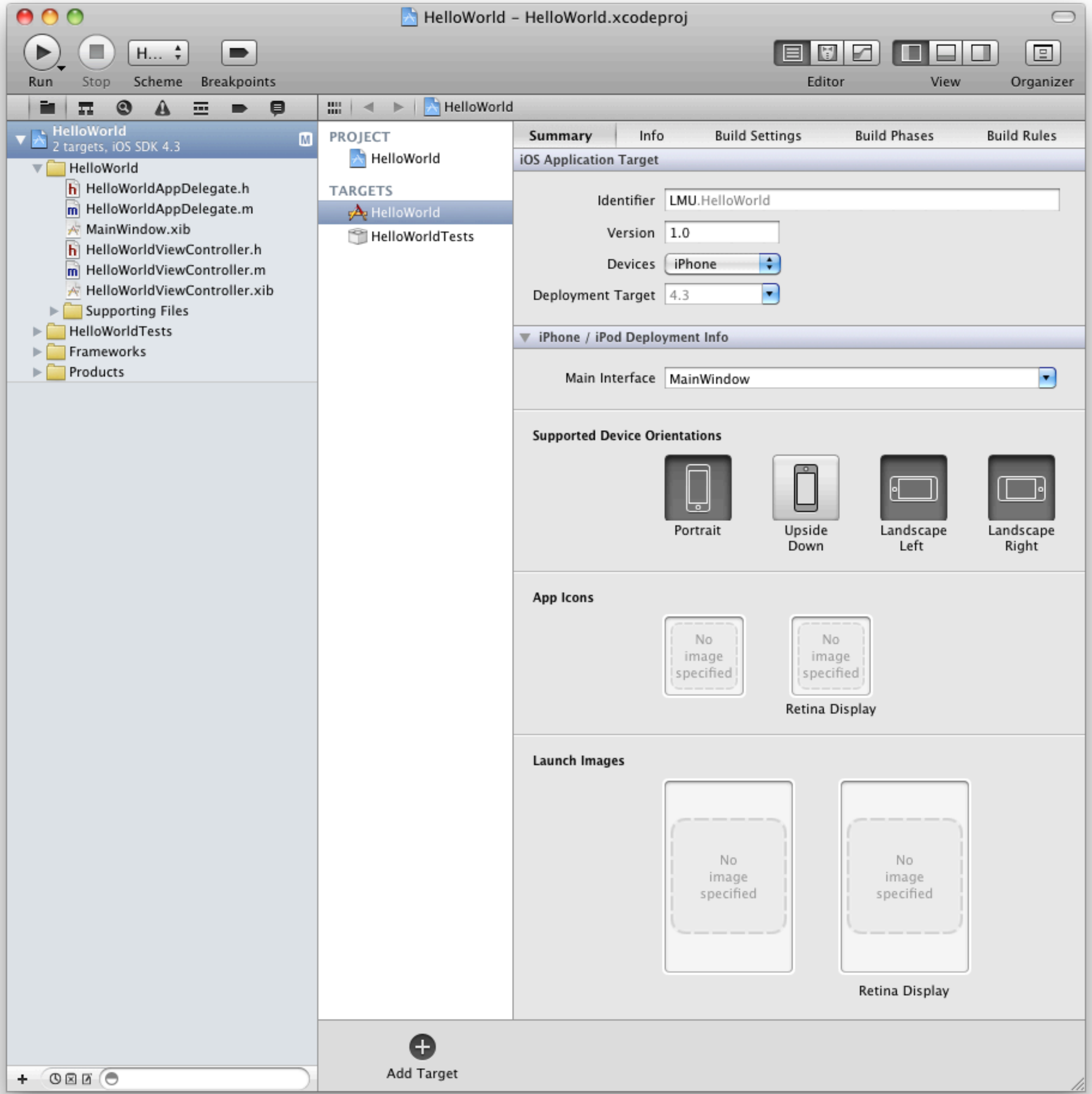
- Text input
  - Add UITextField in Interface Builder
  - Add member variable and property to .h, synthesize in .m
  - Declare UITextFieldDelegate in .h
  - Implement delegate methods in .m, set label text on end editing
  - Set delegate in viewDidLoad method
- Action sheets
  - Implement UIActionSheetDelegate in .h file
  - Construct, showInView, release
  - Implement delegate method clickedButtonAtIndex



Choose a template for your new project:

View-based Application

This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a nib file that contains the view.



iOS Application Target

Identifier: LMU.HelloWorld

Version: 1.0

Devices: iPhone

Deployment Target: 4.3

iPhone / iPod Deployment Info

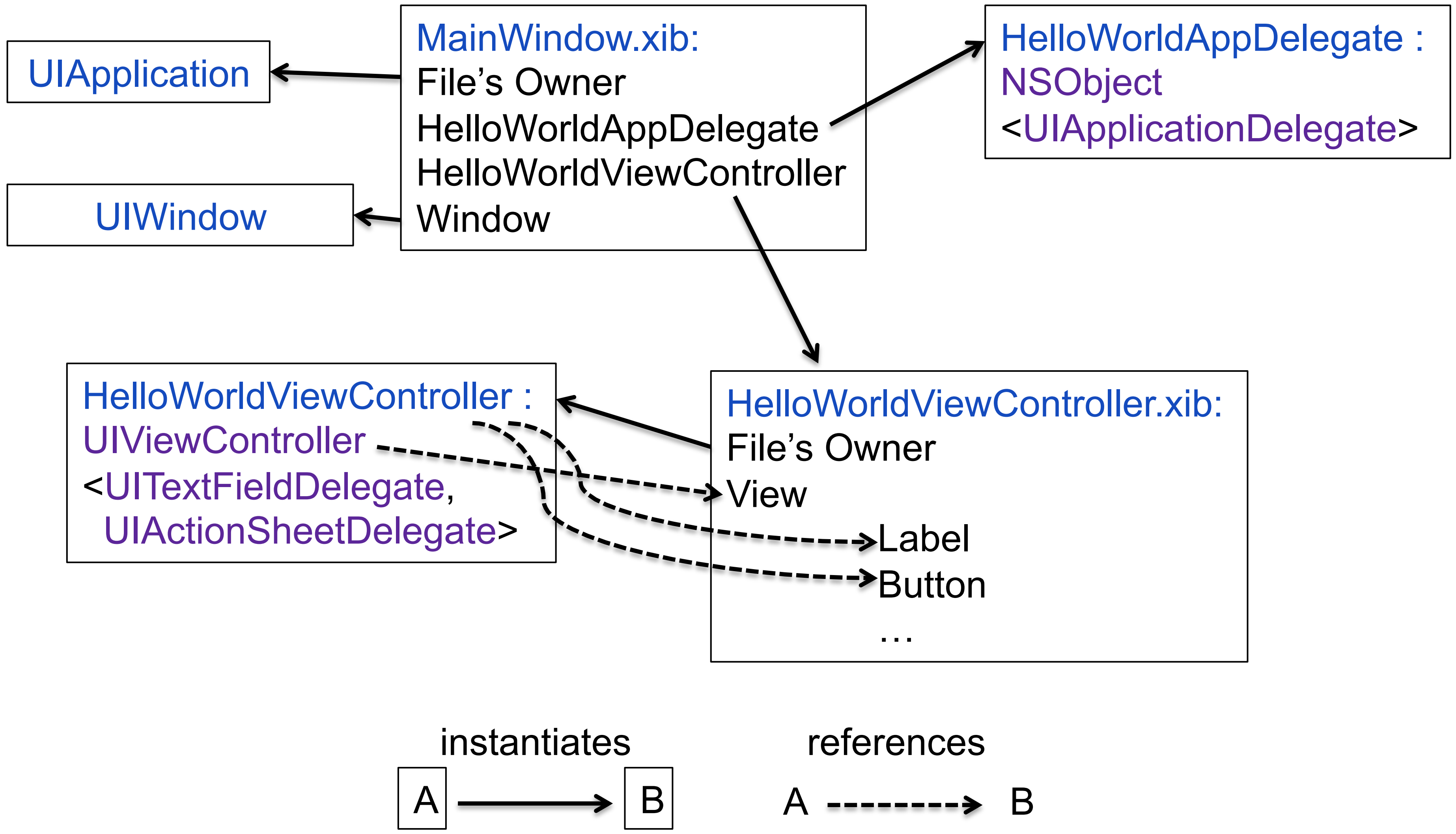
Main Interface: MainWindow

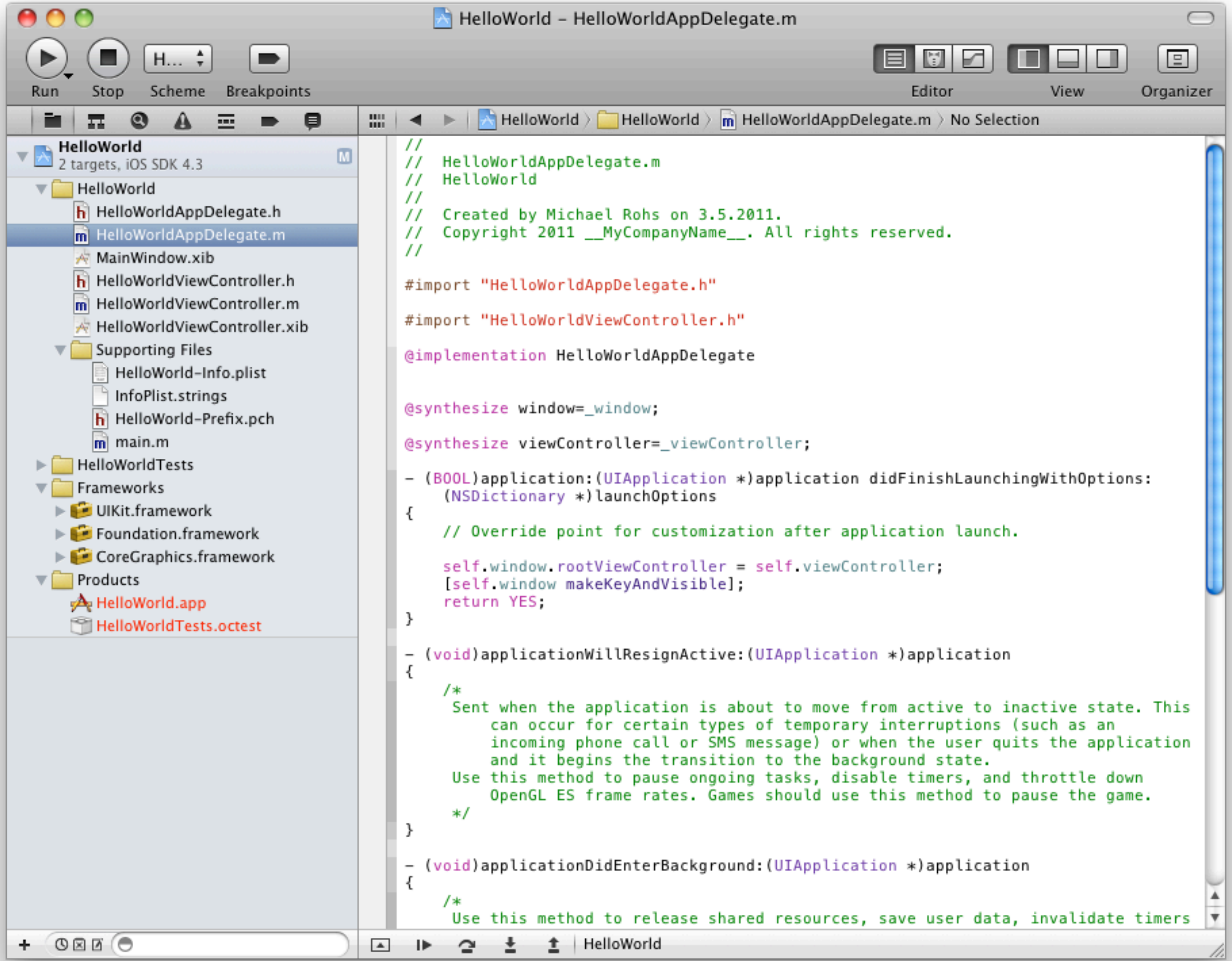
Supported Device Orientations: Portrait, Upside Down, Landscape Left, Landscape Right

App Icons: No image specified; No image specified (Retina Display)

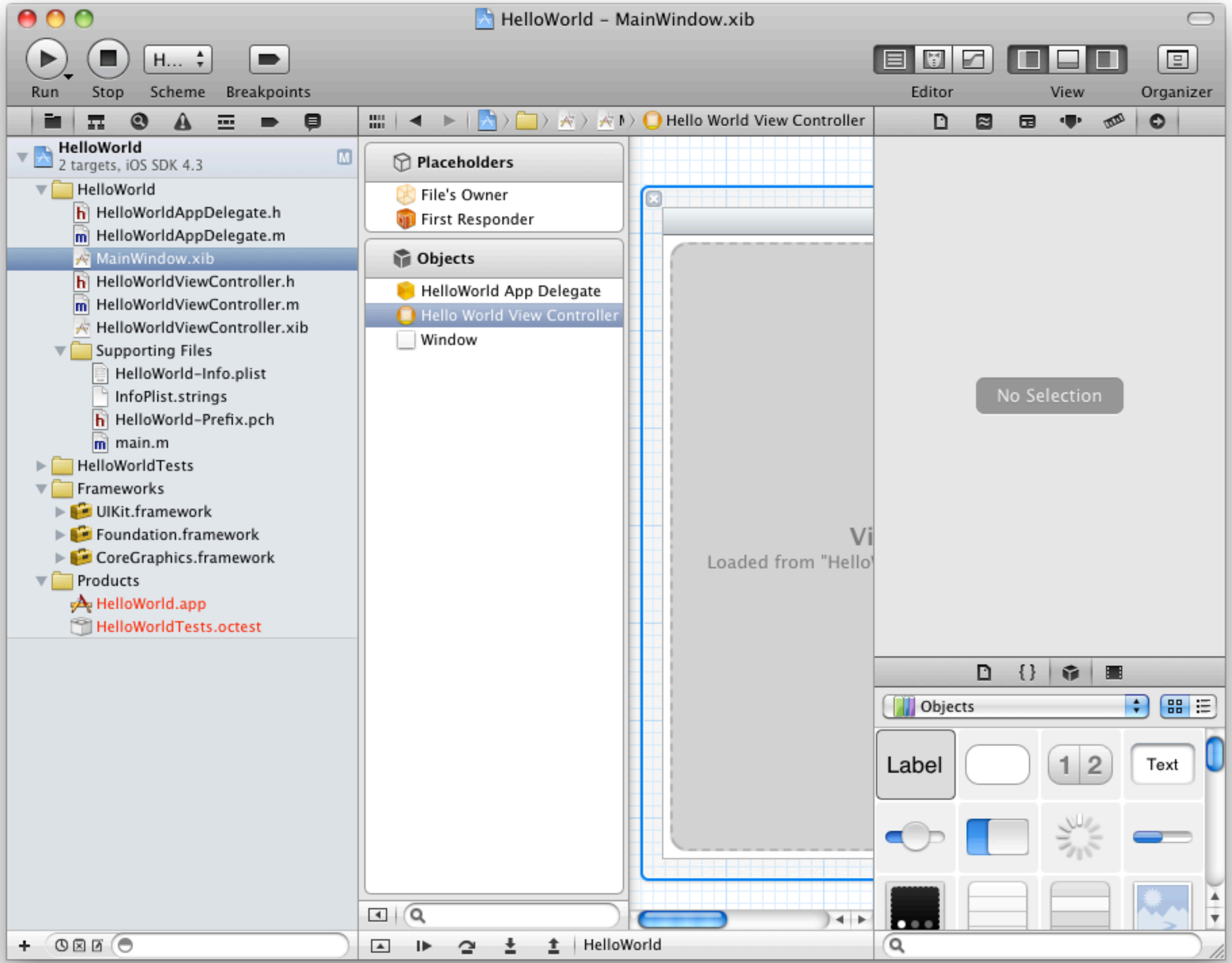
Launch Images: No image specified; No image specified (Retina Display)

Hello World Application Architecture





```
//
//  HelloWorldAppDelegate.m
//  HelloWorld
//
//  Created by Michael Rohs on 3.5.2011.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldAppDelegate.h"

#import "HelloWorldViewController.h"

@implementation HelloWorldAppDelegate


@synthesize window=_window;

@synthesize viewController=_viewController;

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
    (NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
     
    self.window.rootViewController = self.viewController;
    [self.window makeKeyAndVisible];
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    /*
     Sent when the application is about to move from active to inactive state. This
         can occur for certain types of temporary interruptions (such as an
         incoming phone call or SMS message) or when the user quits the application
         and it begins the transition to the background state.
     Use this method to pause ongoing tasks, disable timers, and throttle down
         OpenGL ES frame rates. Games should use this method to pause the game.
     */
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    /*
     Use this method to release shared resources, save user data, invalidate timers
```

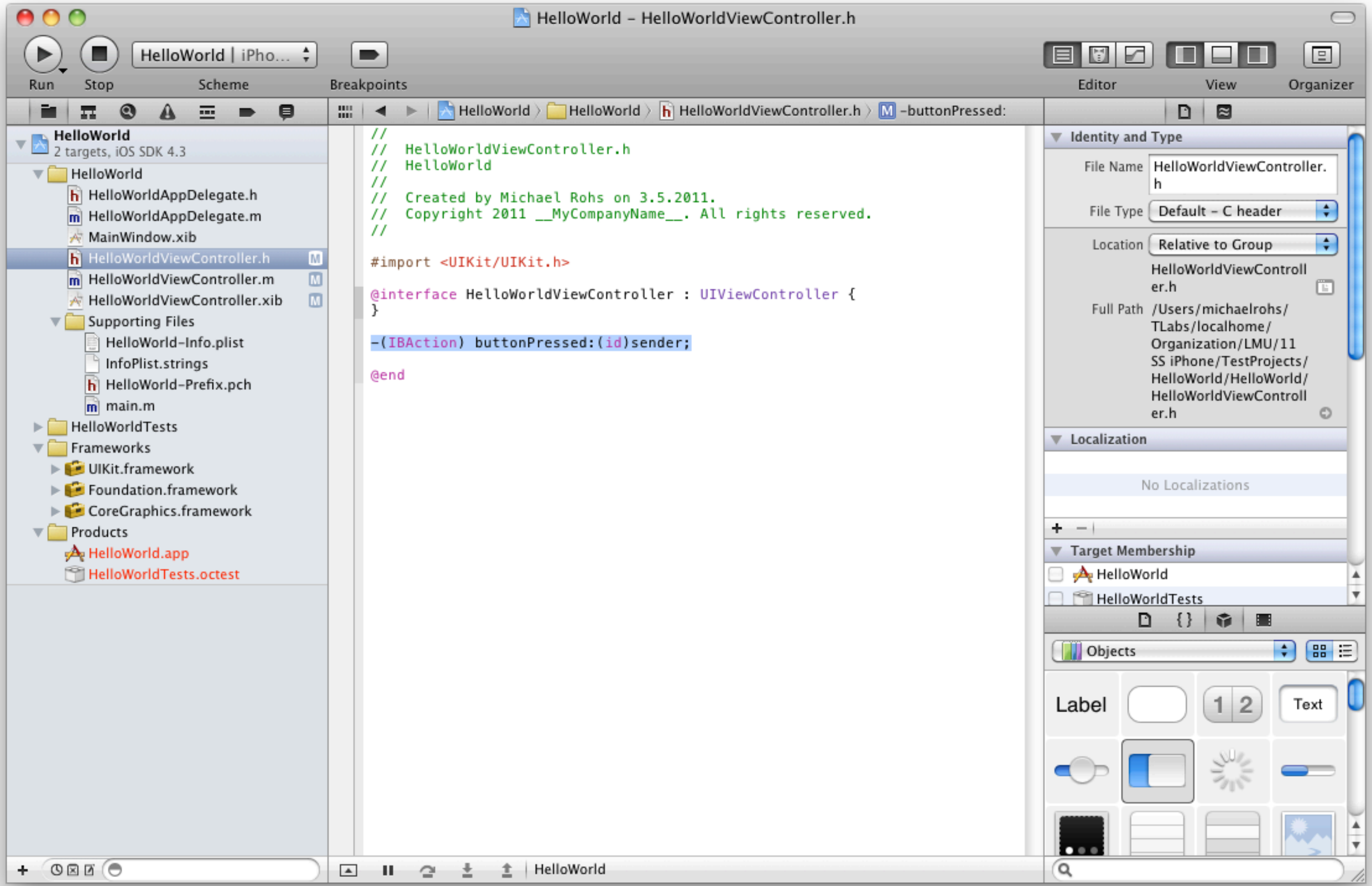

HelloWorld – MainWindow.xib
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
Hello World View Controller
HelloWorld
2 targets, iOS SDK 4.3
HelloWorld
HelloWorldAppDelegate.h
HelloWorldAppDelegate.m
MainWindow.xib
HelloWorldViewController.h
HelloWorldViewController.m
HelloWorldViewController.xib
Supporting Files
HelloWorld-Info.plist
InfoPlist.strings
HelloWorld-Prefix.pch
main.m
HelloWorldTests
Frameworks
UIKit.framework
Foundation.framework
CoreGraphics.framework
Products
HelloWorld.app
HelloWorldTests.octest
Placeholders
File's Owner
First Responder
Objects
HelloWorld App Delegate
Hello World View Controller
Window
Loaded from "Hello
No Selection
Objects
Label
1 2
Text
HelloWorld

UIViewController subclasses

- View lifecycle
  - (void)viewDidLoad
  - (void)viewDidUnload
- View events
  - (void) viewWillAppear:(BOOL)animated
  - (void) viewWillDisappear:(BOOL)animated
  - (void) viewDidAppear:(BOOL)animated
  - (void) viewDidDisappear:(BOOL)animated
- Rotation settings and events

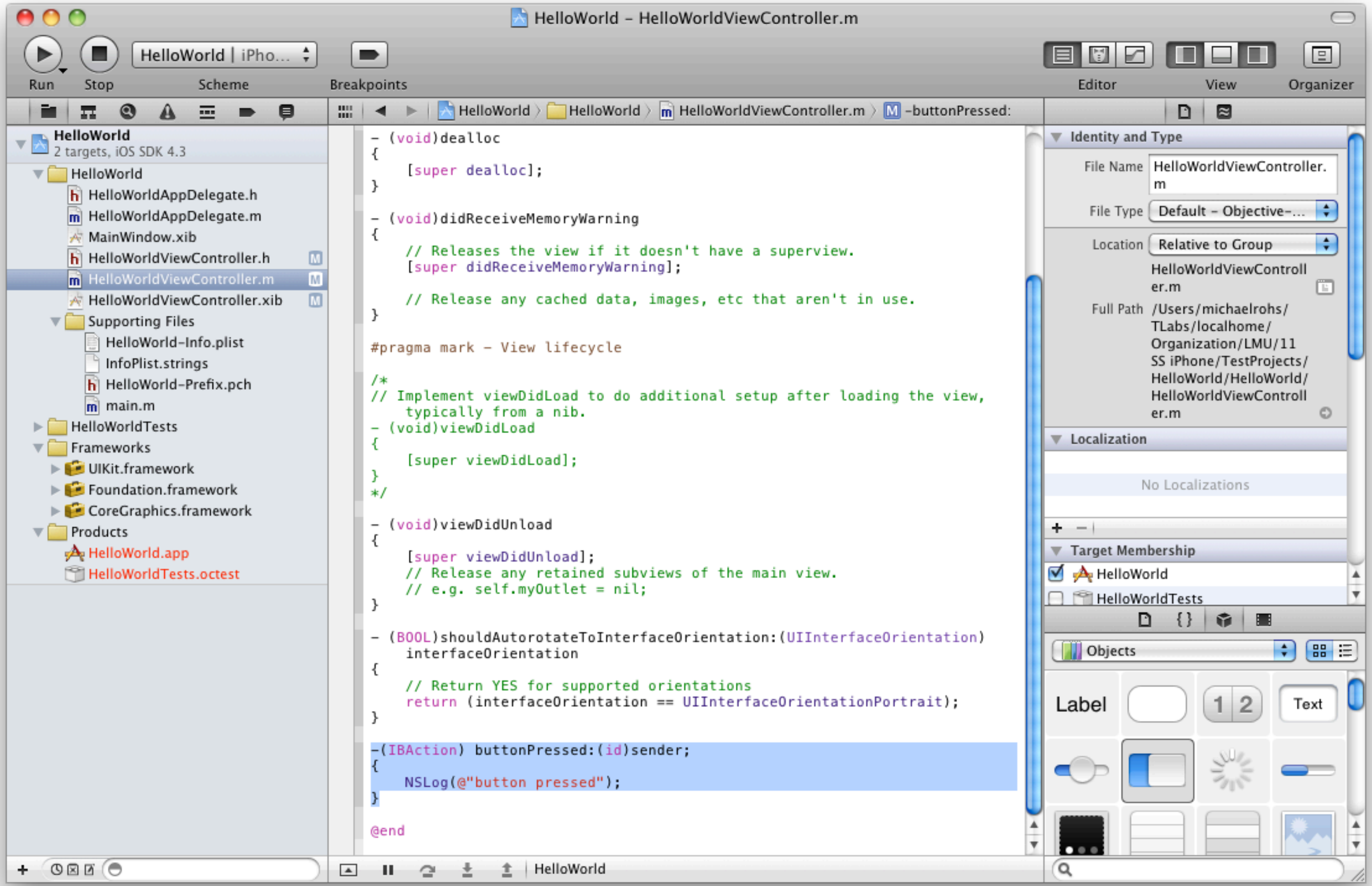
  interfaceOrientation property

  – shouldAutorotateToInterfaceOrientation:
- many more… → see documentation



```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Michael Rohs on 3.5.2011.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController {
}

-(IBAction) buttonPressed:(id)sender;

@end
```

File Name: HelloWorldViewController.h
File Type: Default – C header
Location: Relative to Group
HelloWorldViewController.h
Full Path: /Users/michaelrohs/TLabs/localhome/Organization/LMU/11 SS iPhone/TestProjects/HelloWorld/HelloWorld/HelloWorldViewController.h



```
- (void)dealloc
{
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

/*
// Implement viewDidLoad to do additional setup after loading the view,
    typically from a nib.
- (void)viewDidLoad
{
    [super viewDidLoad];
}
*/

- (void)viewDidUnload
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
    interfaceOrientation
{
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

-(IBAction) buttonPressed:(id)sender;
{
    NSLog(@"button pressed");
}

@end
```

Identity and Type

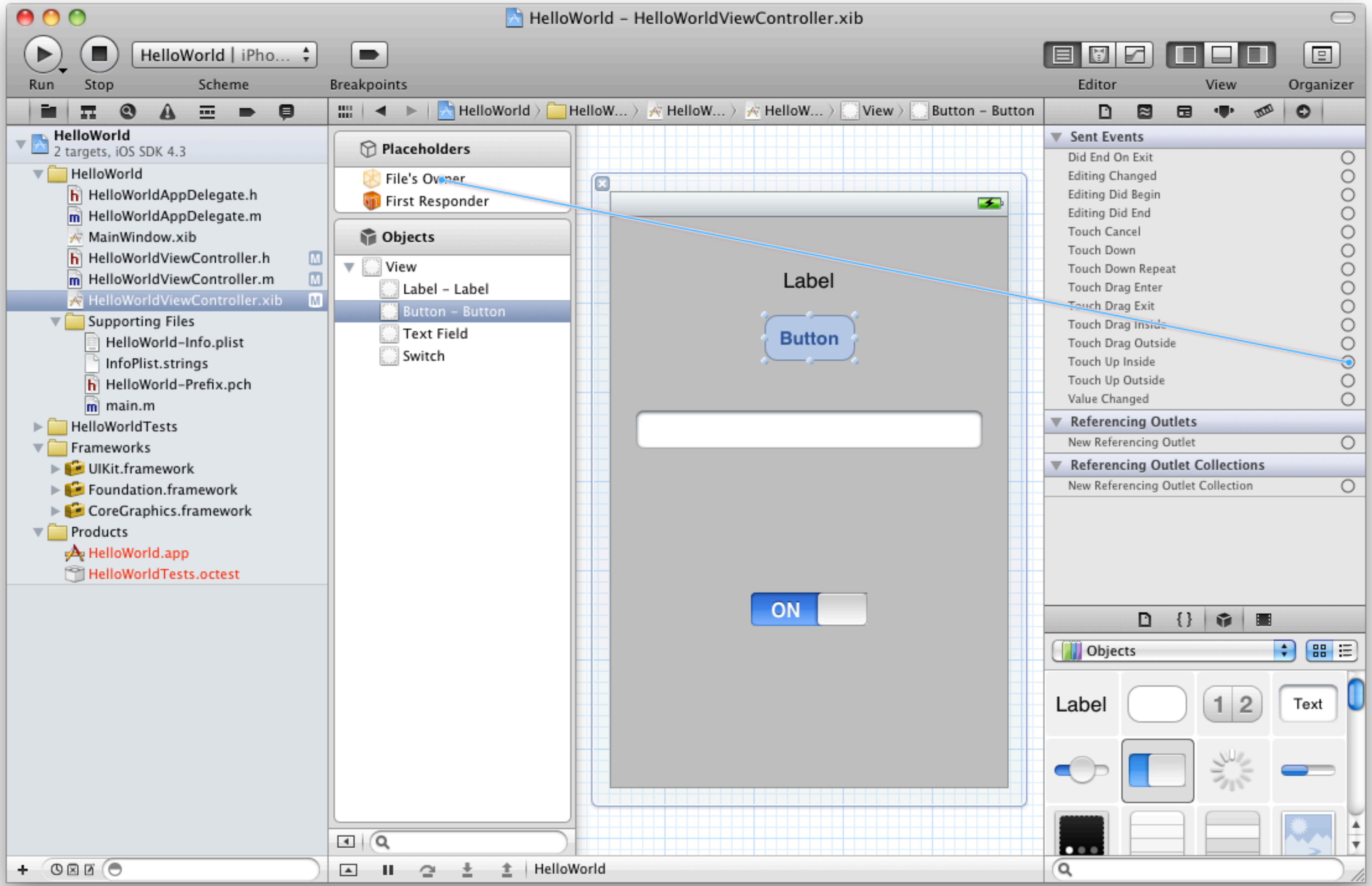
File Name	HelloWorldViewController.m
File Type	Default – Objective-...
Location	Relative to Group
	HelloWorldViewController.m
Full Path	/Users/michaelrohs/TLabs/localhome/Organization/LMU/11 SS iPhone/TestProjects/HelloWorld/HelloWorld/HelloWorldViewController.m

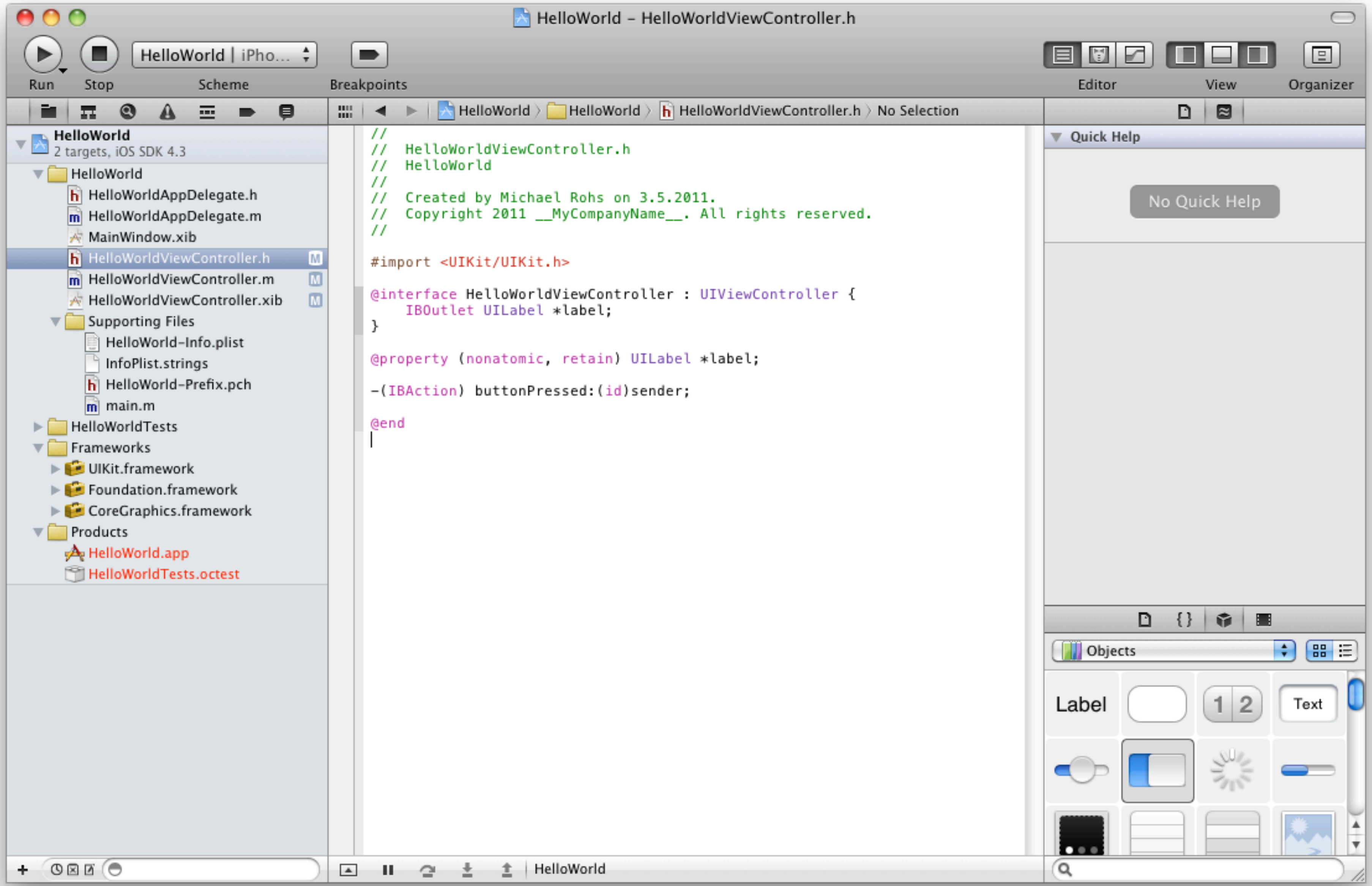
Localization

No Localizations

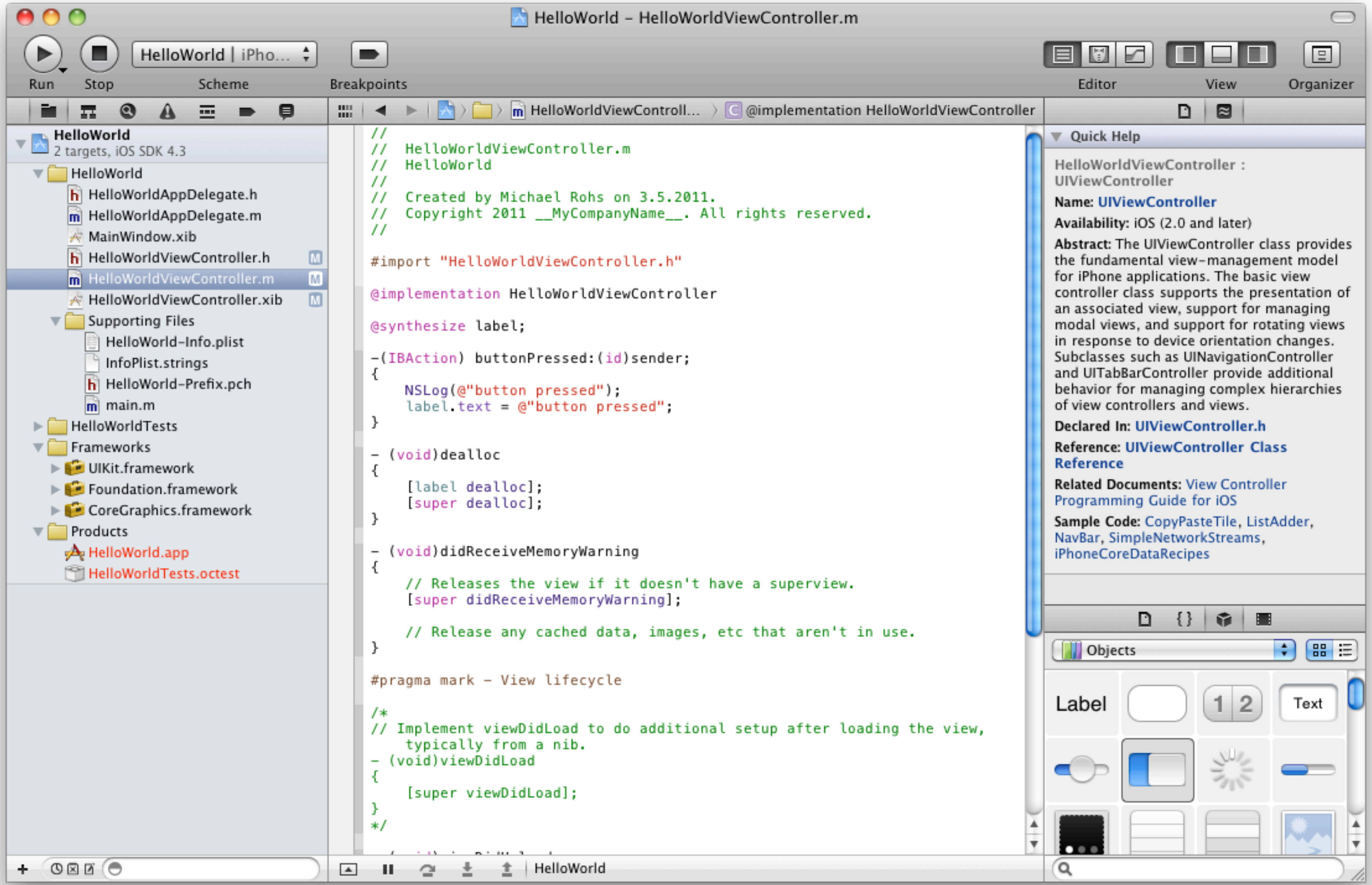
Target Membership

- [✓] HelloWorld
- [ ] HelloWorldTests


HelloWorld – HelloWorldViewController.xib
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
HelloWorld
2 targets, iOS SDK 4.3
HelloWorld
HelloWorldAppDelegate.h
HelloWorldAppDelegate.m
MainWindow.xib
HelloWorldViewController.h
HelloWorldViewController.m
HelloWorldViewController.xib
Supporting Files
HelloWorld-Info.plist
InfoPlist.strings
HelloWorld-Prefix.pch
main.m
HelloWorldTests
Frameworks
UIKit.framework
Foundation.framework
CoreGraphics.framework
Products
HelloWorld.app
HelloWorldTests.octest
Placeholders
File's Owner
First Responder
Objects
View
Label – Label
Button – Button
Text Field
Switch
Label
Button
ON
Sent Events
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Objects
Label
Text



```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Michael Rohs on 3.5.2011.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController {
    IBOutlet UILabel *label;
}

@property (nonatomic, retain) UILabel *label;

-(IBAction) buttonPressed:(id)sender;

@end
```



```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Michael Rohs on 3.5.2011.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize label;

-(IBAction) buttonPressed:(id)sender;
{
    NSLog(@"button pressed");
    label.text = @"button pressed";
}

- (void)dealloc
{
    [label dealloc];
    [super dealloc];
}

- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

/*
// Implement viewDidLoad to do additional setup after loading the view,
    typically from a nib.
- (void)viewDidLoad
{
    [super viewDidLoad];
}
*/
```

HelloWorldViewController : UIViewController

Name: UIViewController

Availability: iOS (2.0 and later)

Abstract: The UIViewController class provides the fundamental view-management model for iPhone applications. The basic view controller class supports the presentation of an associated view, support for managing modal views, and support for rotating views in response to device orientation changes. Subclasses such as UINavigationController and UITabBarController provide additional behavior for managing complex hierarchies of view controllers and views.

Declared In: UIViewController.h

Reference: UIViewController Class Reference

Related Documents: View Controller Programming Guide for iOS

Sample Code: CopyPasteTile, ListAdder, NavBar, SimpleNetworkStreams, iPhoneCoreDataRecipes

HELLO TABLEVIEW

UITableView Example

- Create new project (“View-based application”)
- Change controller base class to UITableViewController
- Declare UITableViewDataSource, UITableViewDelegate
- Add data array to header file, release data in dealloc
- Change view in nib file to UITableView, connect File’s Owner (view, data source, delegate)
- Create arrayWithObjects in onViewDidLoad
- Implement table data source and delegate methods

HELLO MULTIVIEW

View Navigation Example

- Create a “Navigation-Based Application”
- Add NSArray *data to RootViewController
  - Add some data in onViewLoad, retain!
- New File… → UIViewController subclass (with nib file) → “MyDetailViewController”
  - Add UILabel to nib file and to .h file (IBOutlet, @property) and to .m file (@synthesize)
- #import "MyDetailViewController.h”
- Implement didSelectRowAtIndexPath, set selected item
- Show that it does not work ☺ → Debugger
- Show that label is still nil → use member variable, set label in viewDidLoad

View Navigation Example

- Add back button: `self.navigationItem.title = @"List";`