Charting & Cover Flow

iPhone and iPod touch Development
Fall 2009 — Lecture 24

Questions?

Announcements

- No class Thursday — Happy Thanksgiving!

Today's Topics

- Core Plot
- Google Chart API
- Cover Flow

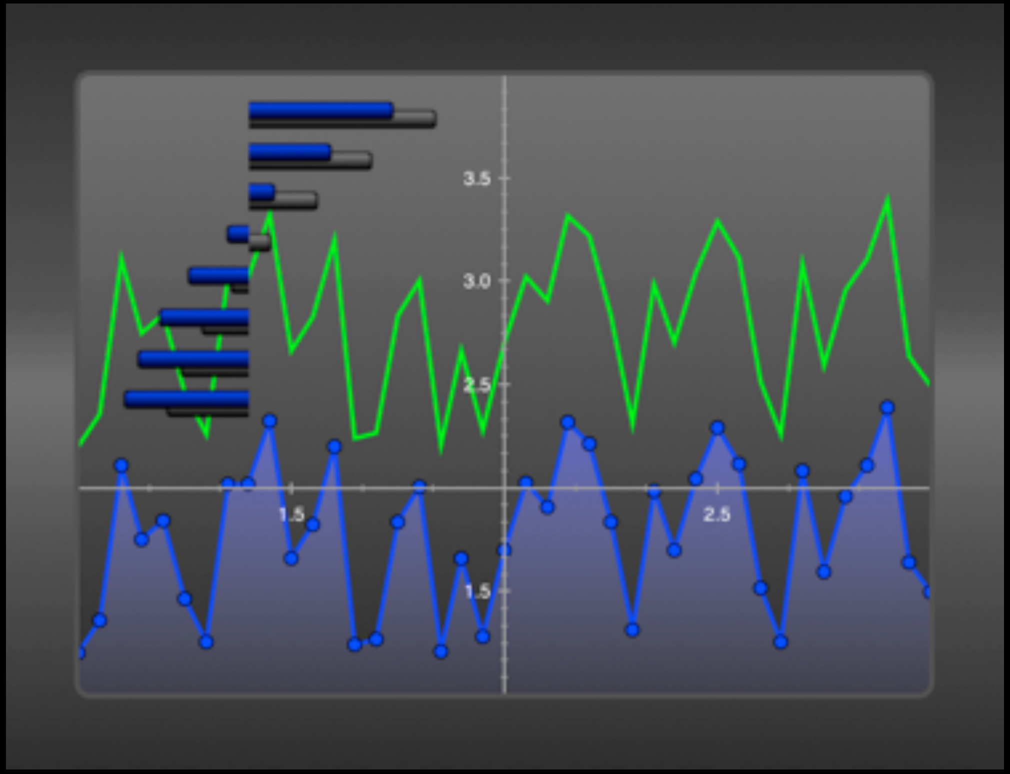
Notes

- I’m showing the relevant portions of the view controller interfaces and implementations in these notes
- Remember to release relevant memory in the -dealloc methods — they are not shown here
- You will also need to wire up outlets and actions in IB
- Where delegates are used, they too require wiring in IB

Core Plot

Core Plot

- Plotting framework for Mac OS X and the iPhone OS
- It provides 2D visualization of data, and is tightly integrated with Apple technologies like Core Animation, Core Data, and Cocoa Bindings
- http://code.google.com/p/core-plot/



About Core Plot

- Drawing is performed using the low-level Quartz 2D API, and Core Animation layers are used to build up the various different aspects of a graph
- Graphs can be animated, with transitions and 3D effects
- Mimics design patterns and technologies used in Apple's own frameworks, such as the data source pattern, delegation, and bindings, are all supported in Core Plot

Obtaining the Code

- It appears that there's no readily downloadable archive, instead you have to check out the code from the Google code repository directly
- This repository only support Mercurial and that's not part of Mac OS X or the developer tools
- Typically I'd recommend to install utilities like this via MacPorts but I had errors when trying to install it (your milage may vary)
  - http://www.macports.org/

Obtaining the Code

- Instead, I grabbed the Mercurial source and did a local install under my user account by issuing the following commands in the Terminal...

```
wget http://mercurial.selenic.com/release/mercurial-1.4.tar.gz
tar zxf mercurial-1.4.tar.gz
cd mercurial-1.4
python2.6 setup.py install --user
cd
~/.local/bin/hg clone https://core-plot.googlecode.com/hg/ core-plot
```

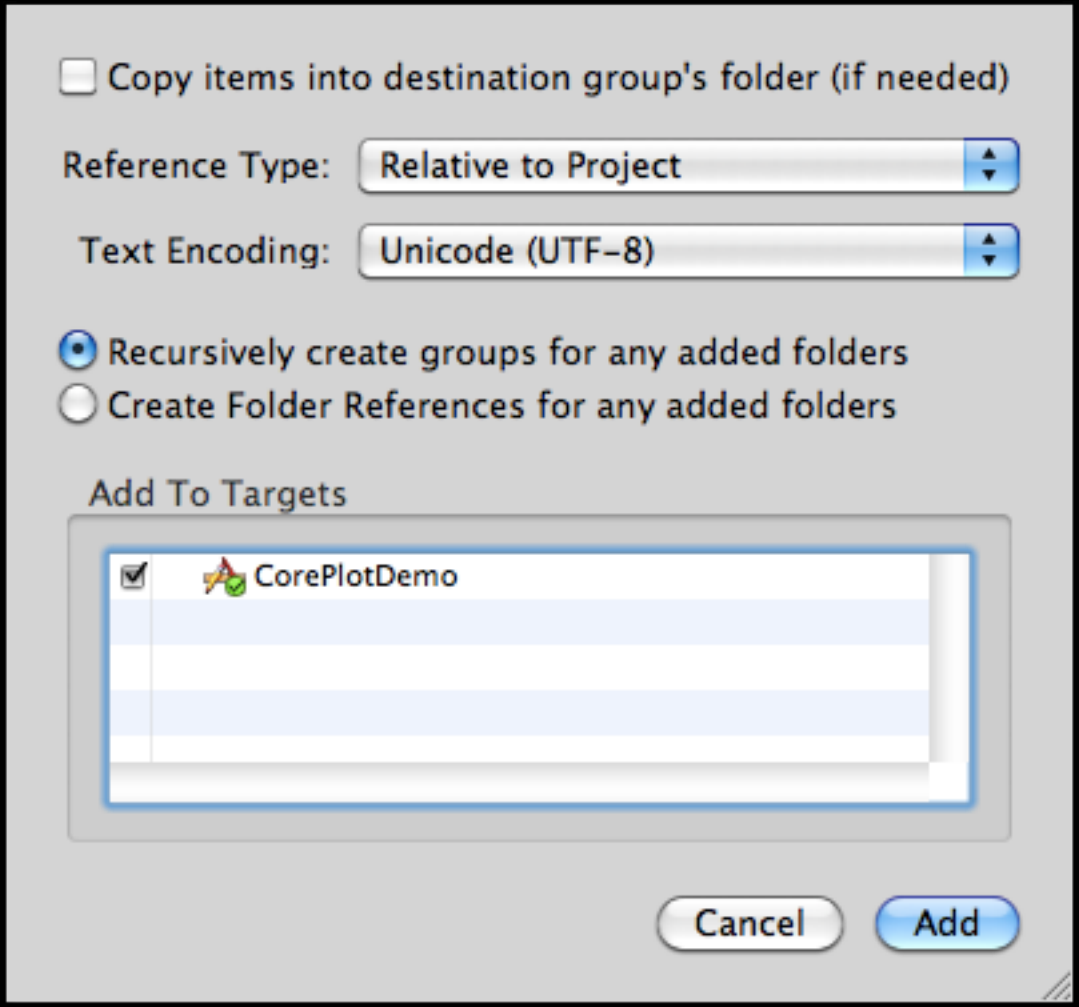
- The wget utility can be found in MacPorts, or you could simply download the tarball in your browser
- Once completed, you should have a copy of the CorePlot source tree in your home directory

Getting Started

- For this example, I’ve created a new view-based project
- First in finder browse to the core-plot directory which is the root of the checkout
- Inside the framework folder drag and drop the CorePlot-CocoaTouch.xcodeproj into your current project in Xcode

Importing Core Plot

- That should bring up the import dialog
- Uncheck the "Copy items into destination group's folder" checkbox
- Also, change the Reference Type to "Relative to Project"



☐ Copy items into destination group's folder (if needed)

Reference Type: Relative to Project

Text Encoding: Unicode (UTF-8)

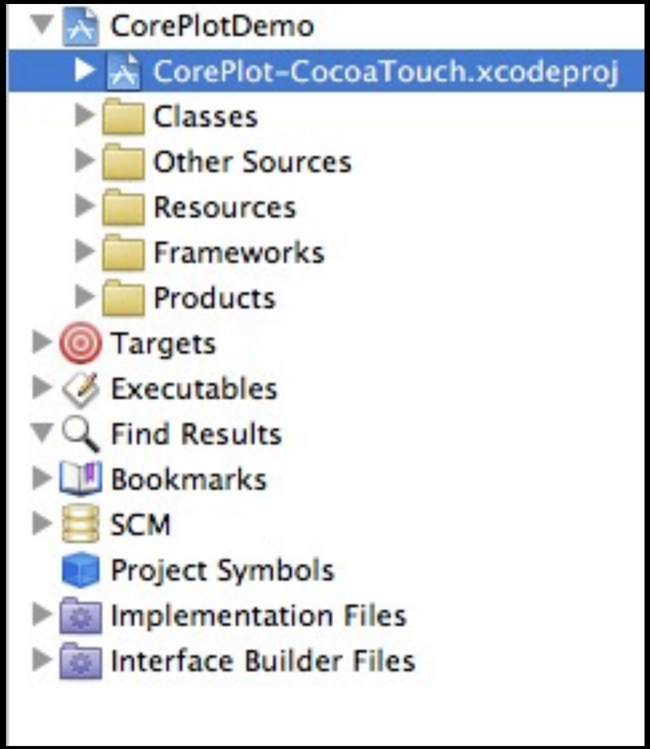
● Recursively create groups for any added folders
○ Create Folder References for any added folders

Add To Targets

☑ CorePlotDemo

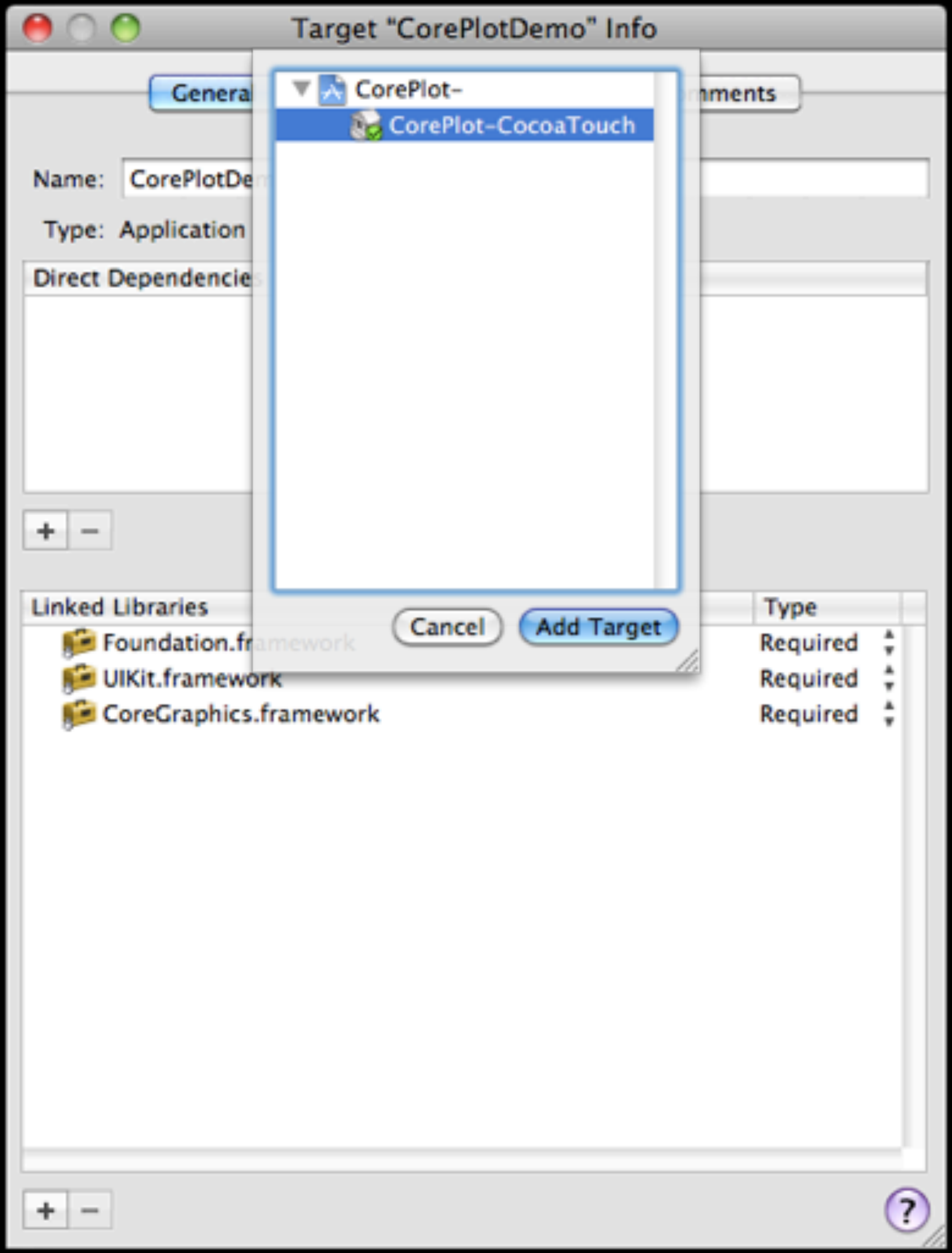
Importing Core Plot

- You should now see CorePlot-CocoaTouch.xcodeproj under your project in the Groups & Files panel...



Adding the Framework

- Expand the Targets item, select your project and bring up the info window (⌘I)
- Click on the "+" button under Direct Dependencies and add the CorePlot-CocoaTouch target


Target "CorePlotDemo" Info

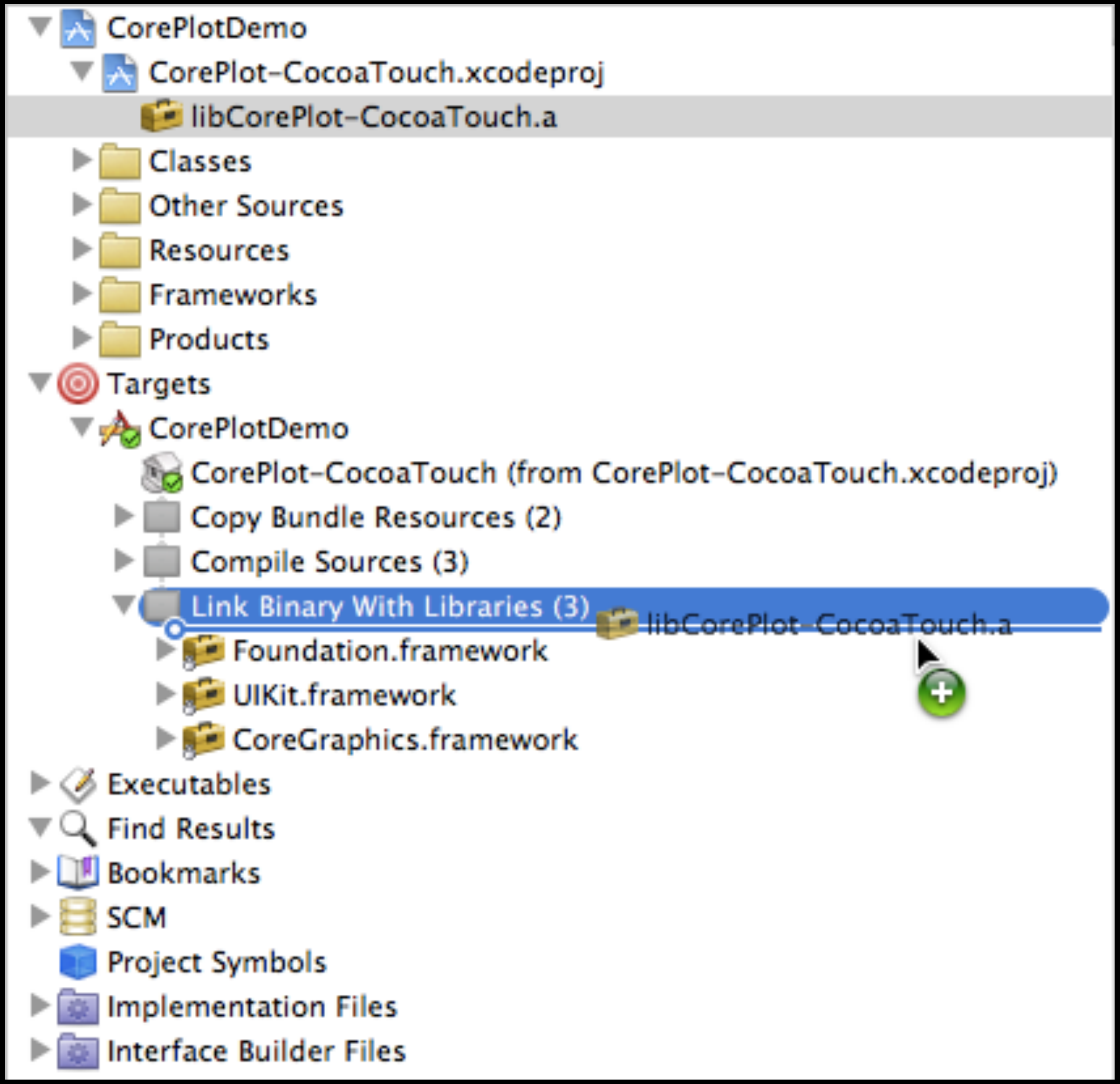
Name: CorePlotDe
Type: Application
Direct Dependencie

CorePlot-
CorePlot-CocoaTouch

Linked Libraries	Type
Foundation.framework	Required
UIKit.framework	Required
CoreGraphics.framework	Required

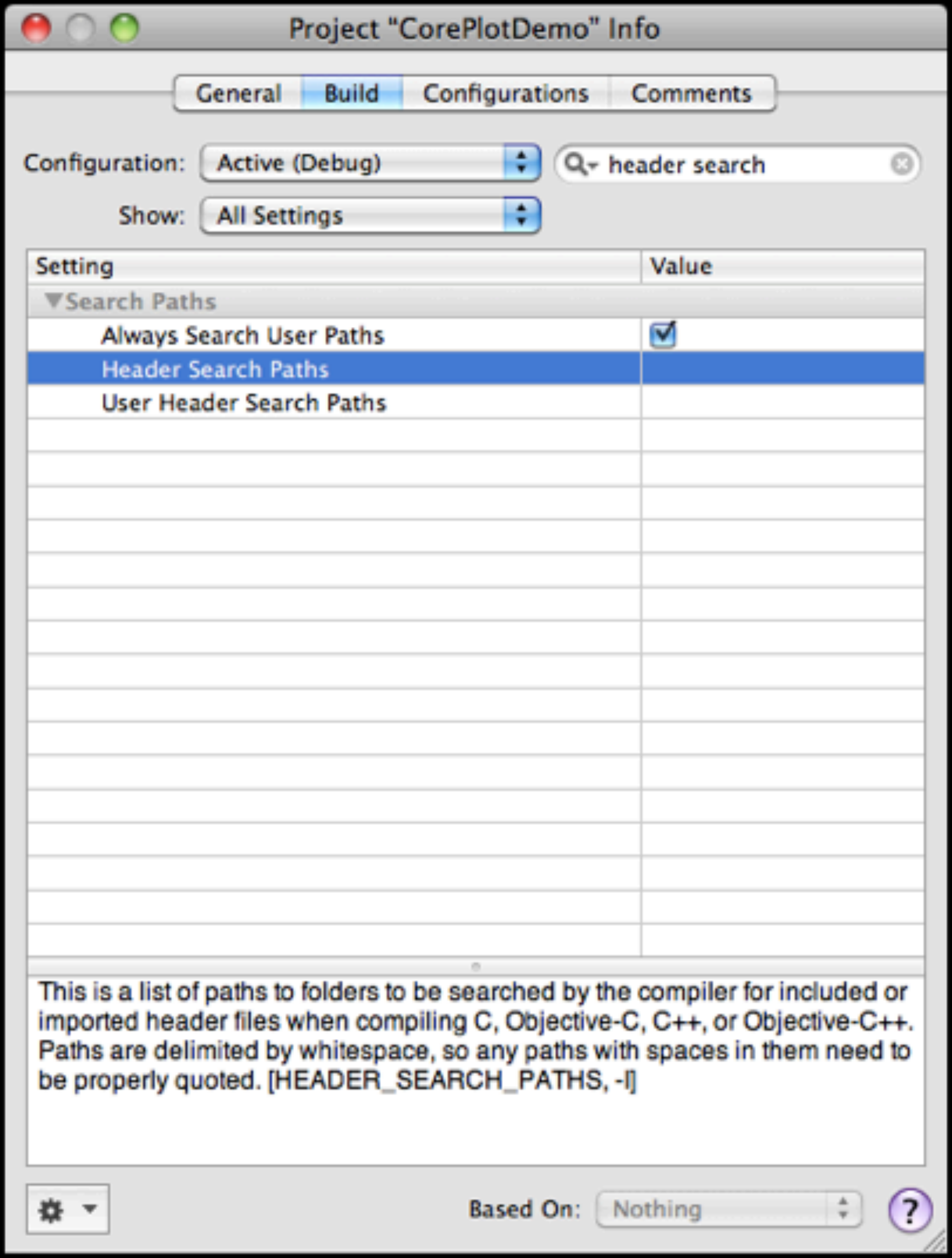
Adding the Library to the Binary

- Lastly, we need to tell our target to add the Core Plot library to our binary
- To do so, expand the embedded CorePlot project and drag and drop the libCorePlot-CocoaTouch.a library into the Target's "Link Binary With Libraries" folder



Adding the Headers

- Under your build settings (click on the project root, then ⌘I to open), switch to the build tab and filter on the string "header search"
- Double click the "Header Search Paths" entry



Configuration: Active (Debug)

Show: All Settings

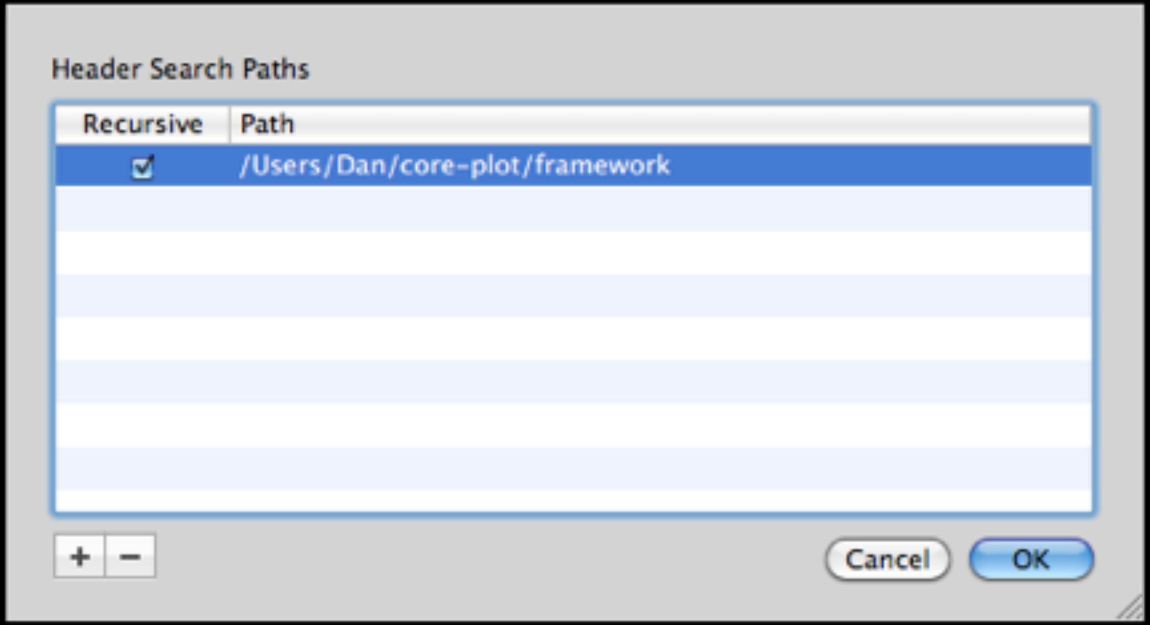
header search

Setting	Value
▼Search Paths	
Always Search User Paths	✓
Header Search Paths	
User Header Search Paths	

This is a list of paths to folders to be searched by the compiler for included or imported header files when compiling C, Objective-C, C++, or Objective-C++. Paths are delimited by whitespace, so any paths with spaces in them need to be properly quoted. [HEADER_SEARCH_PATHS, -I]

Adding the Headers

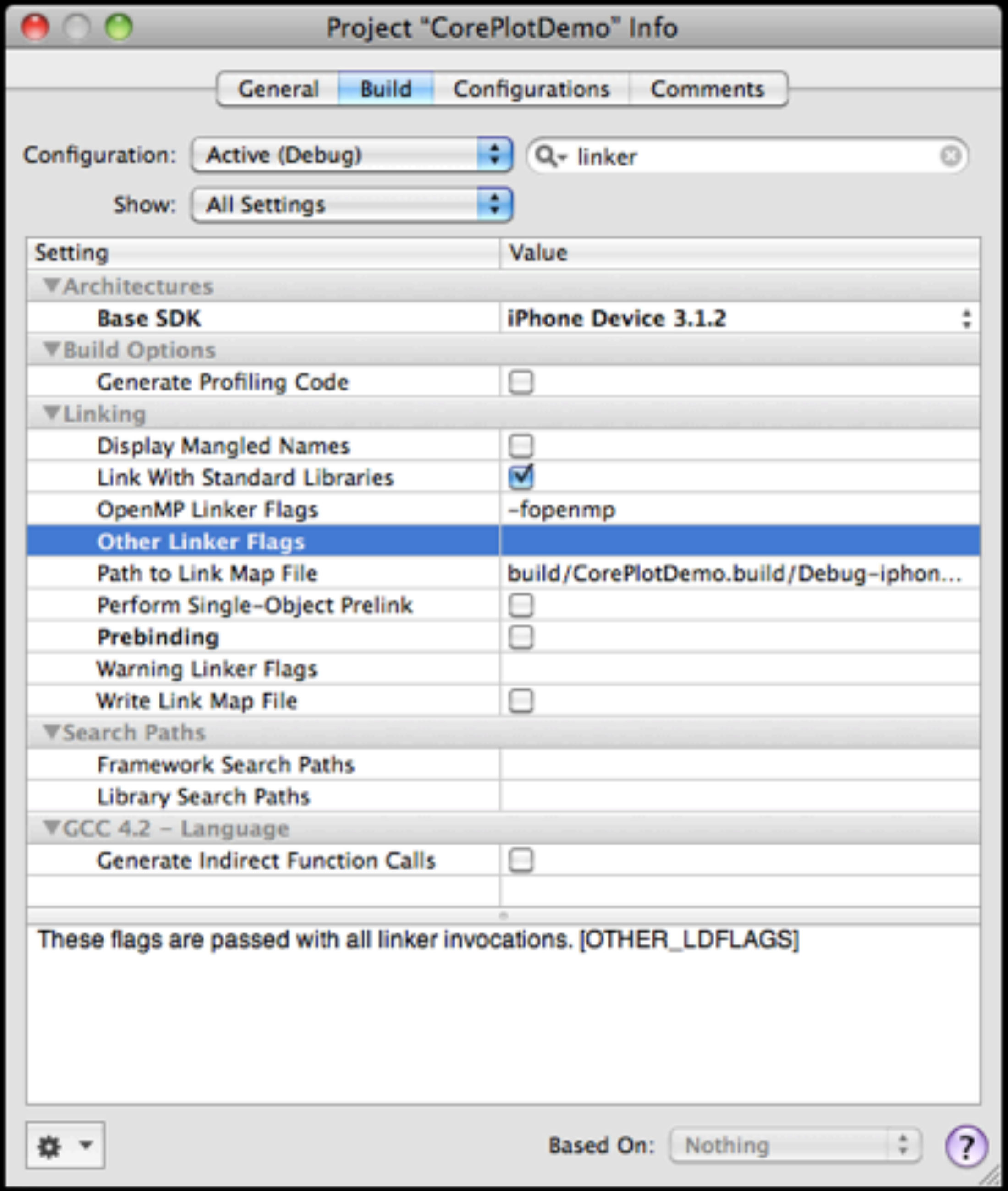
- Add the path to the framework directory under where you downloaded and unwrapped the Core Plot library...


Header Search Paths

Recursive	Path
✓	/Users/Dan/core-plot/framework

Setting the Compiler Flags

- You'll also need to set add 2 linker flags
- Filter on linker and double click on "Other Linker Flags"

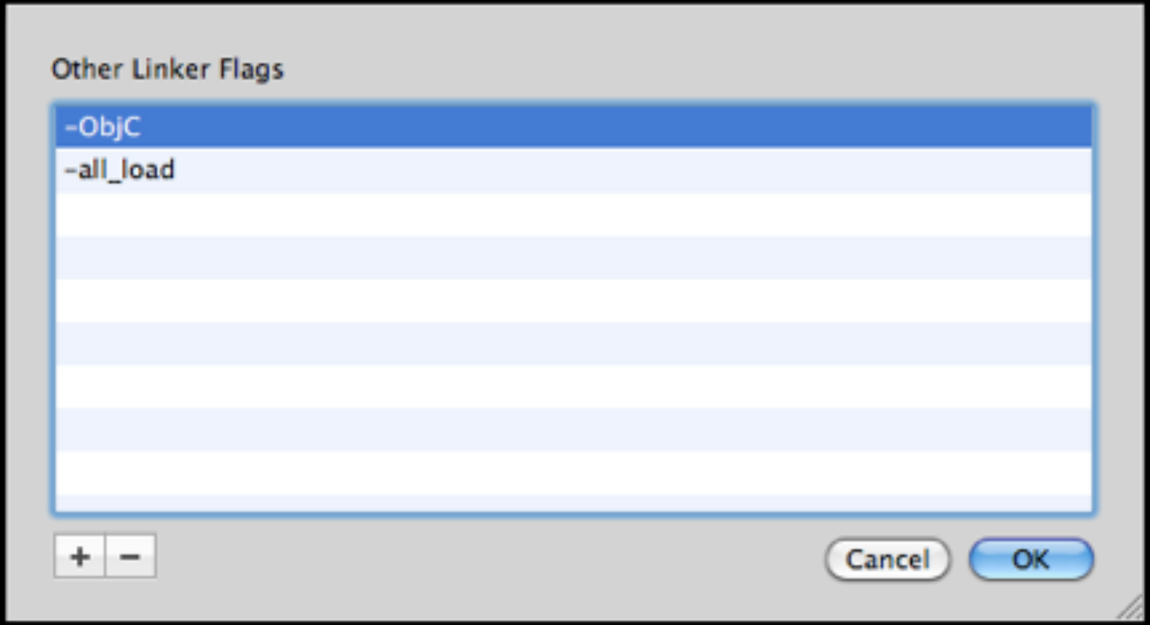


Setting	Value
▼Architectures	
Base SDK	iPhone Device 3.1.2
▼Build Options	
Generate Profiling Code	☐
▼Linking	
Display Mangled Names	☐
Link With Standard Libraries	✓
OpenMP Linker Flags	-fopenmp
Other Linker Flags	
Path to Link Map File	build/CorePlotDemo.build/Debug-iphon…
Perform Single-Object Prelink	☐
Prebinding	☐
Warning Linker Flags	
Write Link Map File	☐
▼Search Paths	
Framework Search Paths	
Library Search Paths	
▼GCC 4.2 - Language	
Generate Indirect Function Calls	☐

These flags are passed with all linker invocations. [OTHER_LDFLAGS]

Setting the Compiler Flags

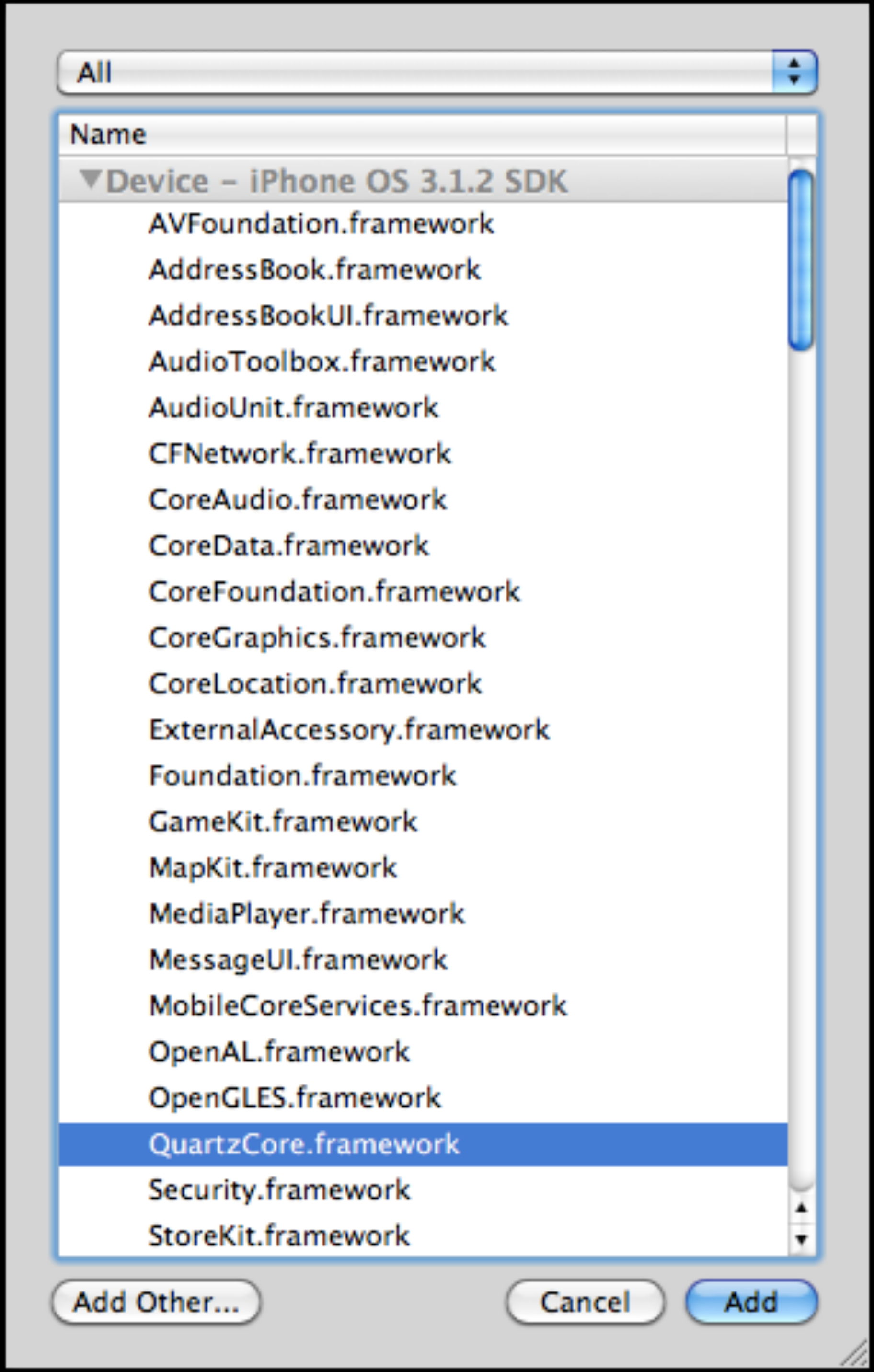
- Add the following 2 linker flags...


Other Linker Flags

```
-ObjC
-all_load
```

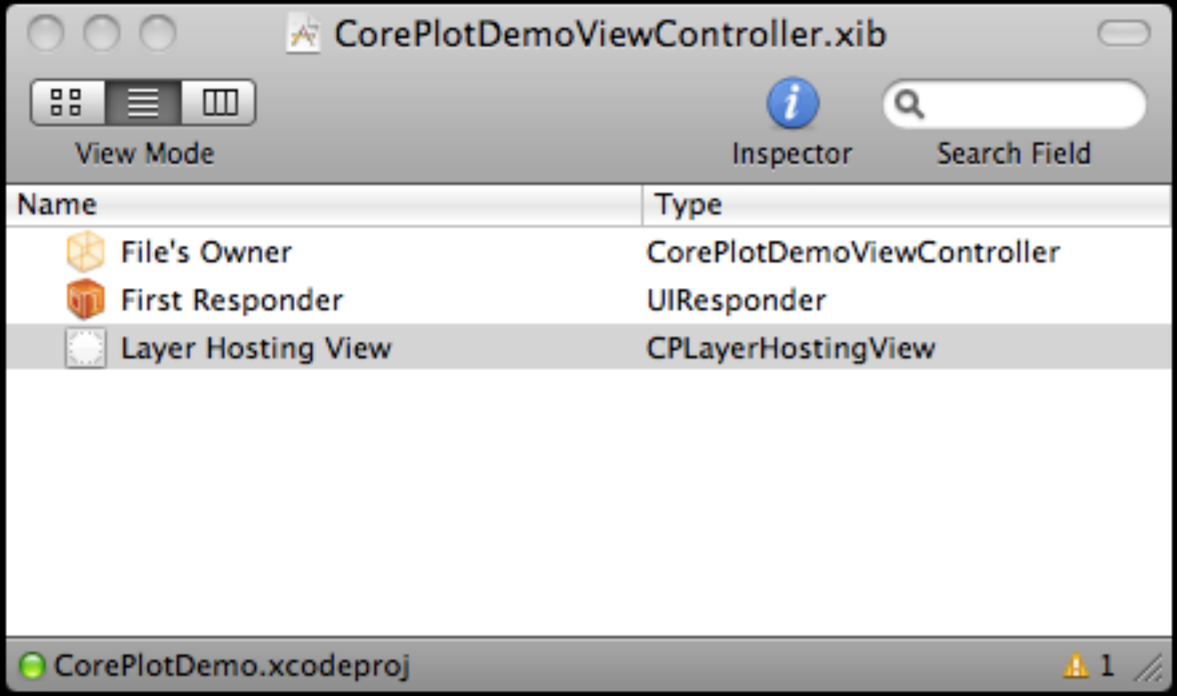
Add the Dependent Framework

- Last setup step...
- Core Plot requires the use of the QuartzCore framework, so add this to your project



CorePlotDemoViewController

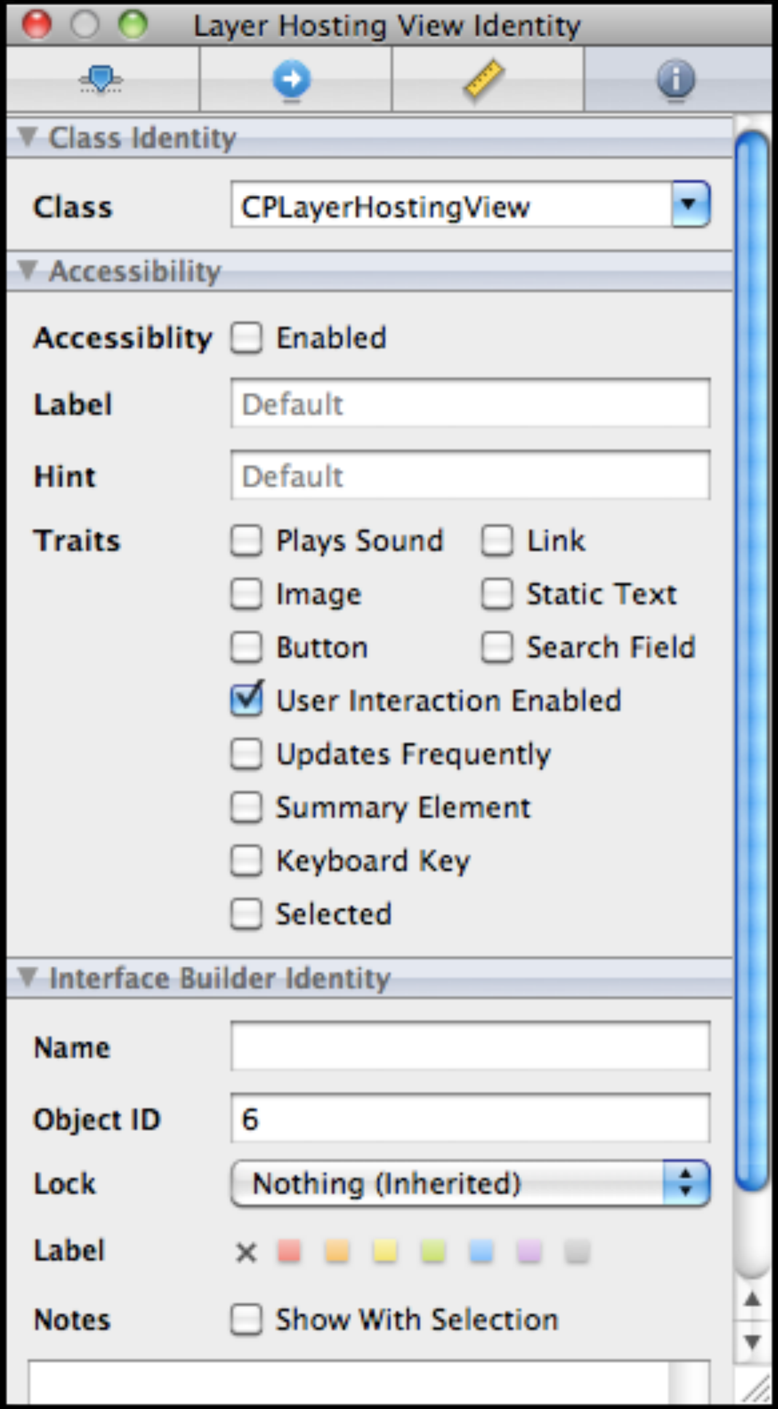
- Core Plot provides a class that is used to house the graph being drawn
- Change the view class to be of type CPLayerHostingView
- The warning is okay here



CorePlotDemoViewController.xib

Name	Type
File's Owner	CorePlotDemoViewController
First Responder	UIResponder
Layer Hosting View	CPLayerHostingView

CorePlotDemo.xcodeproj ⚠ 1



Layer Hosting View Identity

Class Identity

Class: CPLayerHostingView

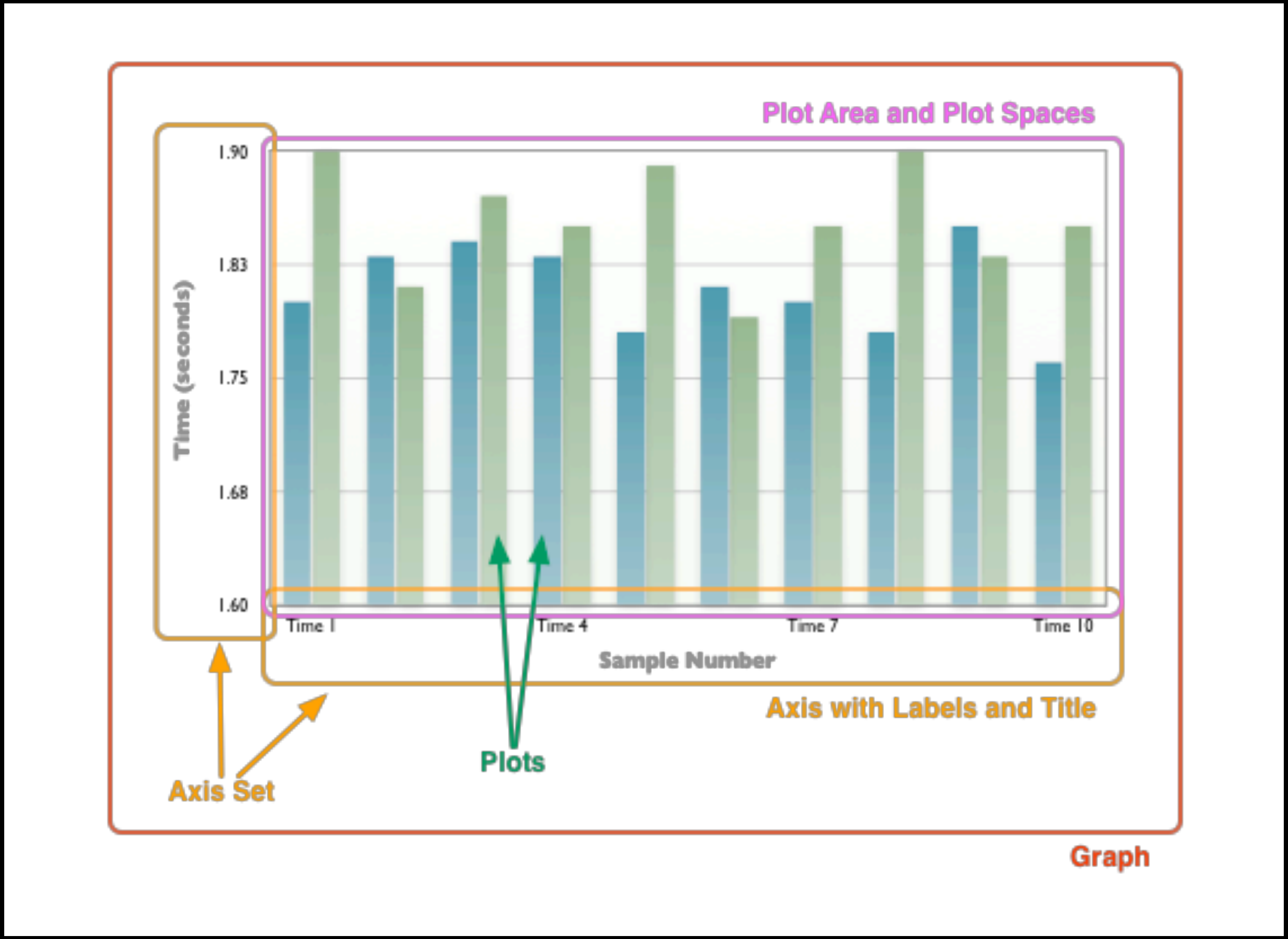
Accessibility

Accessiblity: ☐ Enabled
Label: Default
Hint: Default
Traits: ☐ Plays Sound ☐ Link ☐ Image ☐ Static Text ☐ Button ☐ Search Field ☑ User Interaction Enabled ☐ Updates Frequently ☐ Summary Element ☐ Keyboard Key ☐ Selected

Interface Builder Identity

Name:
Object ID: 6
Lock: Nothing (Inherited)
Label: ×
Notes: ☐ Show With Selection

Anatomy of a Core Data Graph



CorePlotDemoViewController.h

```
#import <UIKit/UIKit.h>
#import "CorePlot-CocoaTouch.h"

@interface CorePlotDemoViewController : UIViewController <CPPlotDataSource> {

   CPXYGraph *graph;
   NSMutableArray *data;

}

@property(nonatomic, retain) CPXYGraph *graph;
@property(nonatomic, retain) NSMutableArray *data;

@end
```

CorePlotDemoViewController.m

```objc
#import "CorePlotDemoViewController.h"

// extents
#define X_MIN -1
#define X_MAX 5
#define Y_MIN -1
#define Y_MAX 9

@implementation CorePlotDemoViewController

@synthesize graph;
@synthesize data;

// setup the data we are going to graph in your app, this data
// would come from somewhere and not be randomly generated
- (void)setupData {
   self.data = [NSMutableArray arrayWithCapacity:5];
   for (int i = 0; i < 5; i++) {
       NSDictionary *point = [NSMutableDictionary dictionary];
       [point setValue:[NSNumber numberWithInt:i] forKey:@"x"];
       [point setValue:[NSNumber numberWithInt:arc4random() % 5 + 2] forKey:@"y"];
       [self.data addObject:point];
   }
}
// ...
```

CorePlotDemoViewController.m

```
// ...

- (void)setupLineGraph {

  // Create a graph and set a theme
  self.graph = [[CPXYGraph alloc] initWithFrame:CGRectZero];
  [self.graph applyTheme:[CPTheme themeNamed:kCPDarkGradientTheme]];
  ((CPLayerHostingView *)self.view).hostedLayer = self.graph;

  // narrow the padding around the graph
  self.graph.paddingLeft = 10.0;
  self.graph.paddingTop = 10.0;
  self.graph.paddingRight = 10.0;
  self.graph.paddingBottom = 10.0;

  // Setup the plot space (extents) of the graph
  CPXYPlotSpace *plotSpace = (CPXYPlotSpace *)self.graph.defaultPlotSpace;
  plotSpace.xRange = [CPPlotRange plotRangeWithLocation:CPDecimalFromFloat(X_MIN)
                                 length:CPDecimalFromFloat(X_MAX - X_MIN)];
  plotSpace.yRange = [CPPlotRange plotRangeWithLocation:CPDecimalFromFloat(Y_MIN)
                                 length:CPDecimalFromFloat(Y_MAX - Y_MIN)];

  // ...
```

CorePlotDemoViewController.m

```
// ...

// Setup axes configuration
CPXYAxis *x = ((CPXYAxisSet *)self.graph.axisSet).xAxis;
x.majorIntervalLength = CPDecimalFromInt(2);
x.constantCoordinateValue = CPDecimalFromInt(0);
x.minorTicksPerInterval = 1;
CPXYAxis *y = ((CPXYAxisSet *)self.graph.axisSet).yAxis;
y.majorIntervalLength = CPDecimalFromInt(2);
y.minorTicksPerInterval = 1;
y.constantCoordinateValue = CPDecimalFromInt(0);

// Create a blue plot area
CPScatterPlot *bluePlot = [[[CPScatterPlot alloc] init] autorelease];
bluePlot.identifier = @"Blue Plot";
bluePlot.dataLineStyle.miterLimit = 2;
bluePlot.dataLineStyle.lineWidth = 4;
bluePlot.dataLineStyle.lineColor = [CPColor blueColor];
bluePlot.dataSource = self;

// ...
```

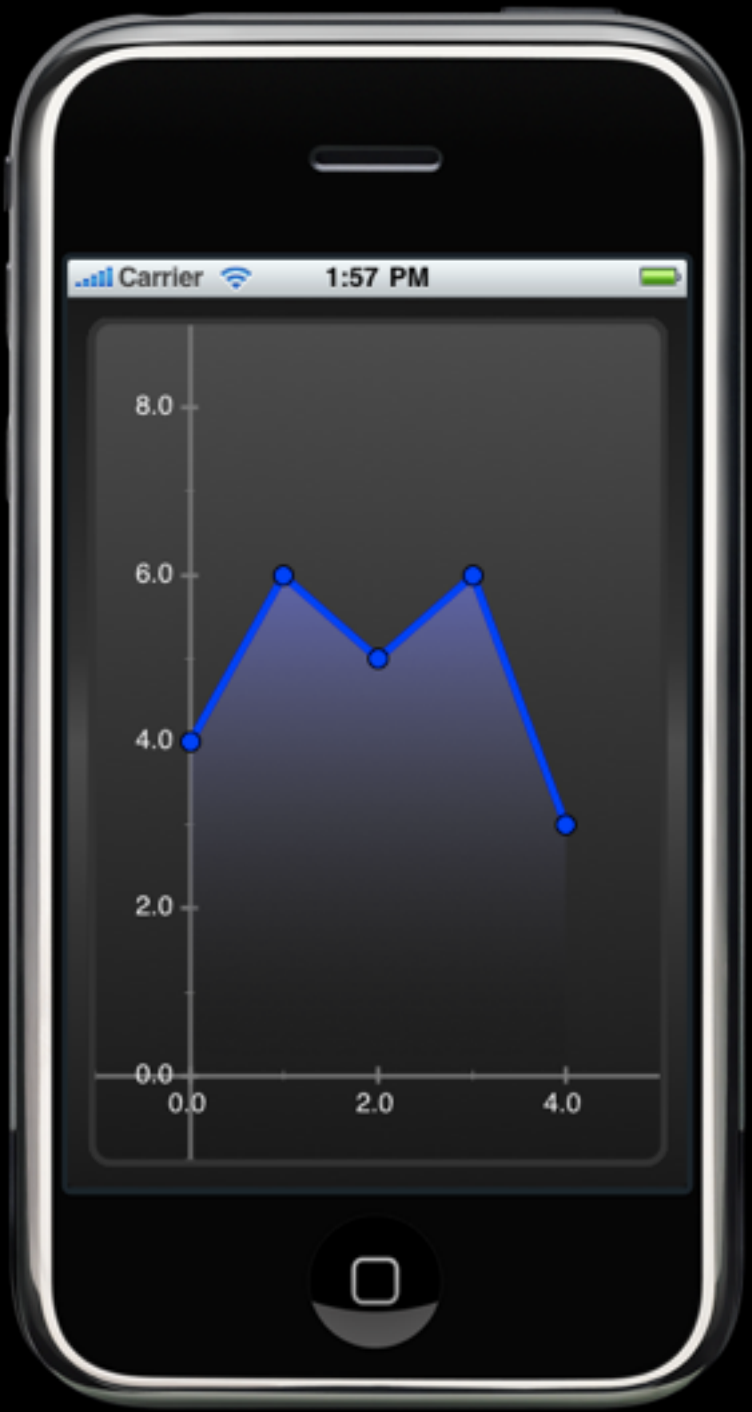
CorePlotDemoViewController.m

```
    // ...

    // Create a gradient for under the blue plot
    CPColor *startColor = [CPColor colorWithComponentRed:.5 green:.5 blue:1.0 alpha:0.8];
    CPColor *stopColor = [CPColor clearColor];
    CPGradient *gradient = [CPGradient gradientWithBeginningColor:startColor
                                                       endingColor:stopColor];
    gradient.angle = -90.0f;
    bluePlot.areaFill = [CPFill fillWithGradient:gradient];
    bluePlot.areaBaseValue = [[NSDecimalNumber zero] decimalValue];

    // Add plot symbols
    CPLineStyle *lineStyle = [CPLineStyle lineStyle];
    lineStyle.lineColor = [CPColor blackColor];
    CPPlotSymbol *lineSymbol = [CPPlotSymbol ellipsePlotSymbol];
    lineSymbol.fill = [CPFill fillWithColor:[CPColor blueColor]];
    lineSymbol.lineStyle = lineStyle;
    lineSymbol.size = CGSizeMake(10.0, 10.0);
    bluePlot.plotSymbol = lineSymbol;

    // add the plot to the graph
    [self.graph addPlot:bluePlot];
}

// ...
```

CorePlotDemoViewController.m

```
// ...
- (void)viewDidLoad {
  [super viewDidLoad];
  [self setupData];
  [self setupLineGraph];
}

- (void)dealloc {
  self.graph = nil;
  self.data = nil;
  [super dealloc];
}

// data source methods
-(NSUInteger)numberOfRecordsForPlot:(CPPlot *)plot {
  return [self.data count];
}
-(NSNumber *)numberForPlot:(CPPlot *)plot
                     field:(NSUInteger)fieldEnum
               recordIndex:(NSUInteger)index {
  NSString *key = (fieldEnum == CPScatterPlotFieldX) ? @"x" : @"y";
  return [[self.data objectAtIndex:index] valueForKey:key];
}

@end
```

The Resulting App



Multiple Series (and a Different Graph Type)

- The previous example only graphed a single series, as such a number of the data source methods were more simplified than they'd be if we had multiple series
- For this example, we'll create a bar graph using 2 different series
- We'll have to look at the CPPlot object being passed in the data source methods and return the appropriate values accordingly

CorePlotDemoViewController.h

```
#import <UIKit/UIKit.h>
#import "CorePlot-CocoaTouch.h"

@interface CorePlotDemoViewController : UIViewController <CPPlotDataSource> {

	CPXYGraph *graph;
	NSMutableArray *series1;
	NSMutableArray *series2;

}

@property(nonatomic, retain) CPXYGraph *graph;
@property(nonatomic, retain) NSMutableArray *series1;
@property(nonatomic, retain) NSMutableArray *series2;

@end
```

CorePlotDemoViewController.m

```
#import "CorePlotDemoViewController.h"

#define X_MIN -1
#define X_MAX 5
#define Y_MIN -1
#define Y_MAX 9

#define SERIES1 @"series1"
#define SERIES2 @"series2"

@implementation CorePlotDemoViewController

@synthesize graph;
@synthesize series1;
@synthesize series2;

// ...
```

CorePlotDemoViewController.m

```
// ...

- (void)setupData {
  self.series1 = [NSMutableArray arrayWithCapacity:5];
  self.series2 = [NSMutableArray arrayWithCapacity:5];
  for (int i = 0; i < 5; i++) {
    NSDictionary *point1 = [NSMutableDictionary dictionary];
    [point1 setValue:[NSNumber numberWithInt:i] forKey:@"x"];
    [point1 setValue:[NSNumber numberWithInt:arc4random() % 5 + 2] forKey:@"y"];
    [self.series1 addObject:point1];
    NSDictionary *point2 = [NSMutableDictionary dictionary];
    [point2 setValue:[NSNumber numberWithInt:i] forKey:@"x"];
    [point2 setValue:[NSNumber numberWithInt:arc4random() % 5 + 2] forKey:@"y"];
    [self.series2 addObject:point2];
  }
}

// ...
```

CorePlotDemoViewController.m

```
// ...

- (void)setupBarGraph {

  // Create self.graph from theme
  self.graph = [[CPXYGraph alloc] initWithFrame:CGRectZero];
  [self.graph applyTheme:[CPTheme themeNamed:kCPDarkGradientTheme]];
  ((CPLayerHostingView *)self.view).hostedLayer = self.graph;

  self.graph.paddingLeft = 10.0;
  self.graph.paddingTop = 10.0;
  self.graph.paddingRight = 10.0;
  self.graph.paddingBottom = 10.0;

  // Setup the plot space (extents) of the graph
  CPXYPlotSpace *plotSpace = (CPXYPlotSpace *)self.graph.defaultPlotSpace;
  plotSpace.xRange = [CPPlotRange plotRangeWithLocation:CPDecimalFromFloat(X_MIN)
                                  length:CPDecimalFromFloat(X_MAX - X_MIN)];
  plotSpace.yRange = [CPPlotRange plotRangeWithLocation:CPDecimalFromFloat(Y_MIN)
                                  length:CPDecimalFromFloat(Y_MAX - Y_MIN)];

// ...
```

CorePlotDemoViewController.m

```
// ...

// Setup axes configuration
CPXYAxis *x = ((CPXYAxisSet *)self.graph.axisSet).xAxis;
x.majorIntervalLength = CPDecimalFromInt(2);
x.constantCoordinateValue = CPDecimalFromInt(0);
x.minorTicksPerInterval = 1;
CPXYAxis *y = ((CPXYAxisSet *)self.graph.axisSet).yAxis;
y.majorIntervalLength = CPDecimalFromInt(2);
y.minorTicksPerInterval = 1;
y.constantCoordinateValue = CPDecimalFromInt(0);

// setup first bar plot
CPBarPlot *barPlot1 = [CPBarPlot tubularBarPlotWithColor:[CPColor blueColor]
                                          horizontalBars:NO];
barPlot1.baseValue = CPDecimalFromInt(0);
barPlot1.dataSource = self;
barPlot1.barOffset = -0.5;
barPlot1.identifier = SERIES1;
[self.graph addPlot:barPlot1 toPlotSpace:plotSpace];

// ...
```

CorePlotDemoViewController.m

```
  // ...

  // setup second bar plot
  CPBarPlot *barPlot2 = [CPBarPlot tubularBarPlotWithColor:[CPColor greenColor]
                                            horizontalBars:NO];
  barPlot2.baseValue = CPDecimalFromInt(0);
  barPlot2.dataSource = self;
  barPlot2.barOffset = 0.5;
  barPlot2.identifier = SERIES2;
  [self.graph addPlot:barPlot2 toPlotSpace:plotSpace];
}

- (void)viewDidLoad {
  [super viewDidLoad];
  [self setupData];
  [self setupBarGraph];
}

// ...
```

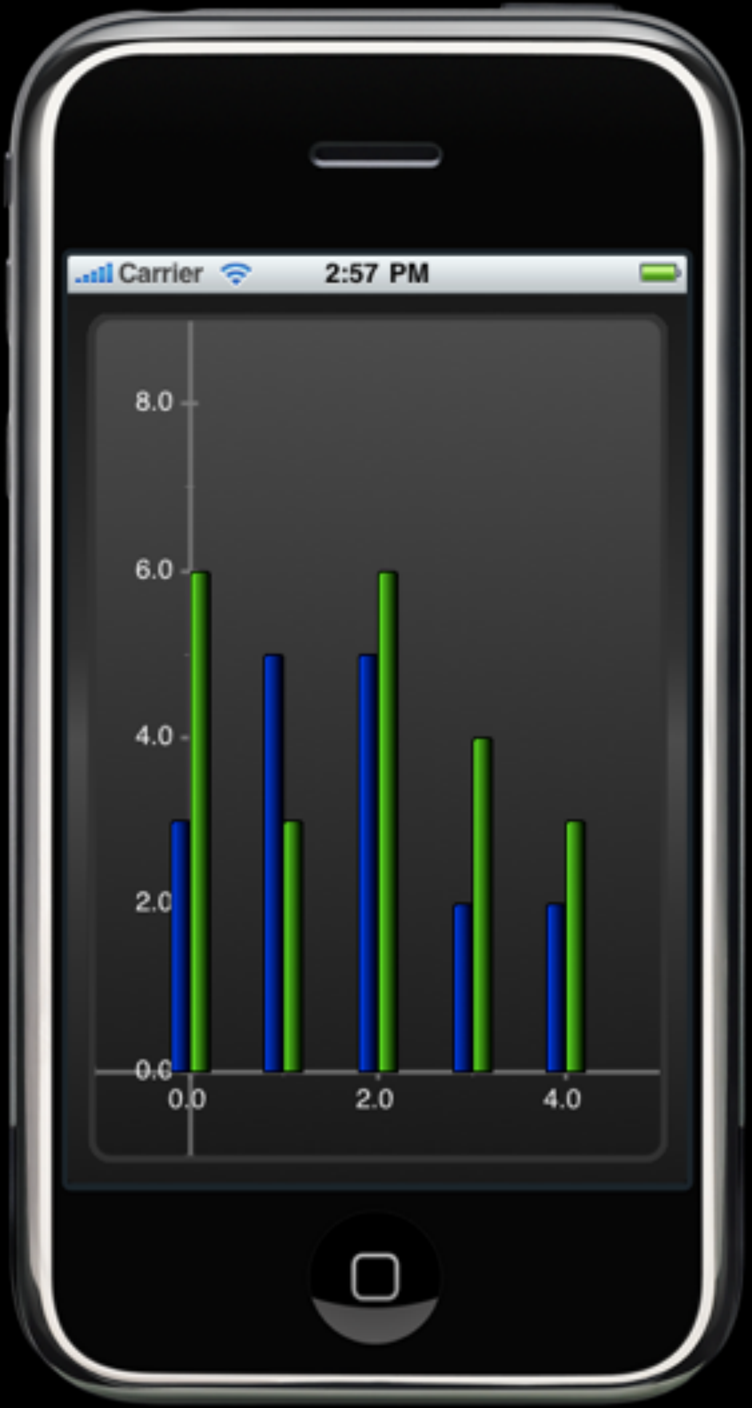
CorePlotDemoViewController.m

```
// ...

// data source methods
-(NSUInteger)numberOfRecordsForPlot:(CPPlot *)plot {
  if ([(NSString *)plot.identifier isEqualToString:SERIES1]) {
    return [self.series1 count];
  } else {
    return [self.series2 count];
  }
}
-(NSNumber *)numberForPlot:(CPPlot *)plot
                     field:(NSUInteger)fieldEnum
               recordIndex:(NSUInteger)index {
  NSString *key = (fieldEnum == CPScatterPlotFieldX) ? @"x" : @"y";
  if ([(NSString *)plot.identifier isEqualToString:SERIES1]) {
    return [[self.series1 objectAtIndex:index] valueForKey:key];
  } else {
    return [[self.series2 objectAtIndex:index] valueForKey:key];
  }
}

// ...
```

CorePlotDemoViewController.m

```
// ...

- (void)dealloc {
   self.graph = nil;
   self.series1 = nil;
   self.series2 = nil;
   [super dealloc];
}

@end
```

The Resulting App

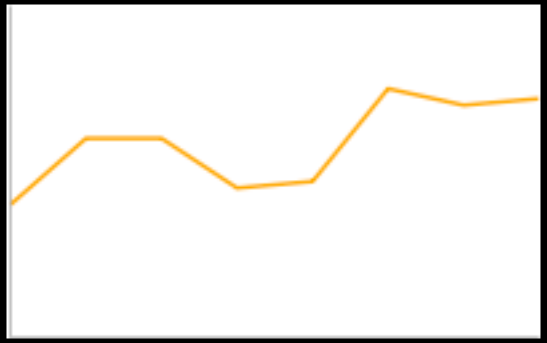
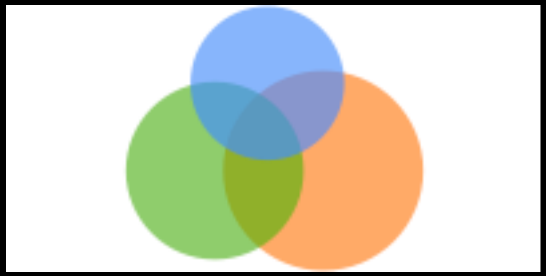


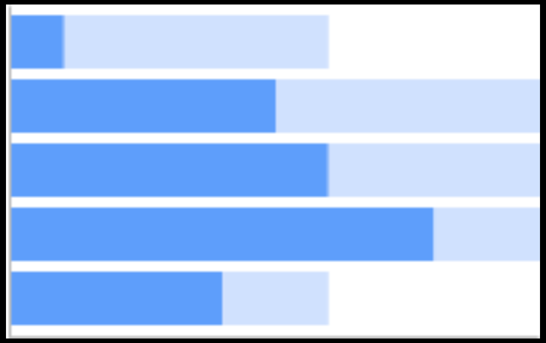
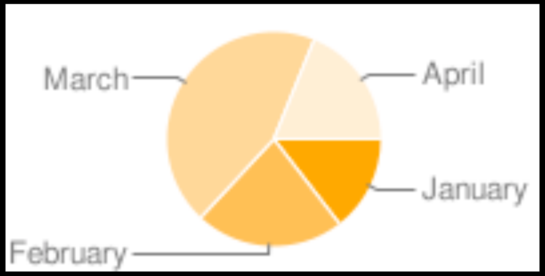
Google Chart API

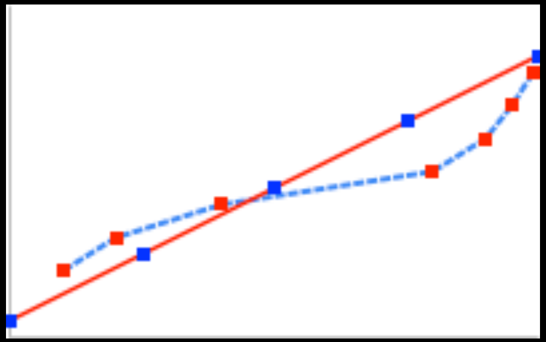
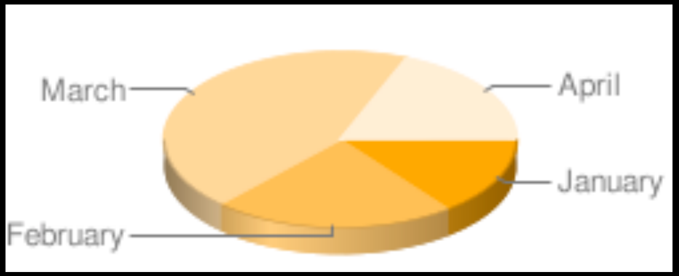
Google Chart API

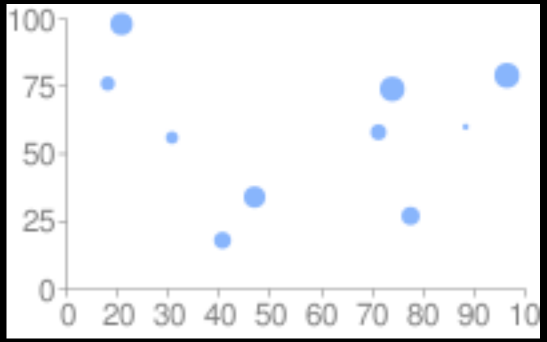
- The Google Chart API lets you dynamically generate charts of many different varieties
- To use the API, all you need to do is to simply encode your data into a specific URL scheme and perform an HTTP request
- Google will generate an image based on your data and return it as the HTTP response
- Understand that since these charts are being created on Google's end over the network, there may be latency or lack of connectivity considerations when using this approach

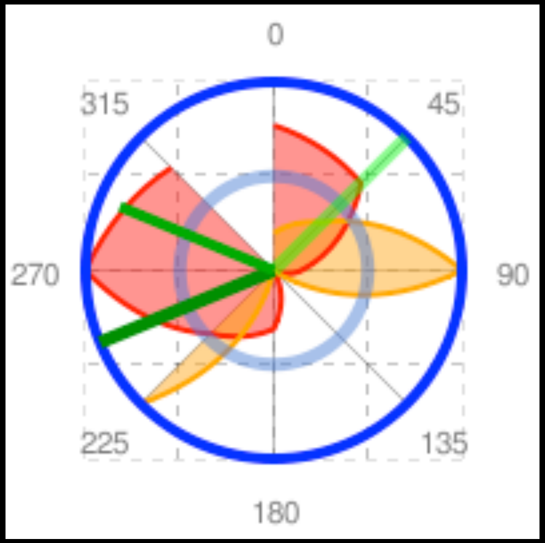
Some Google Chart Samples

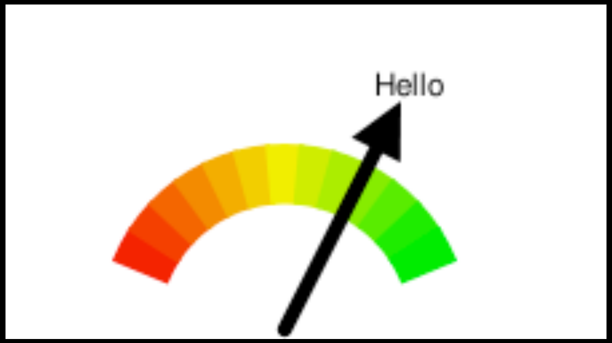



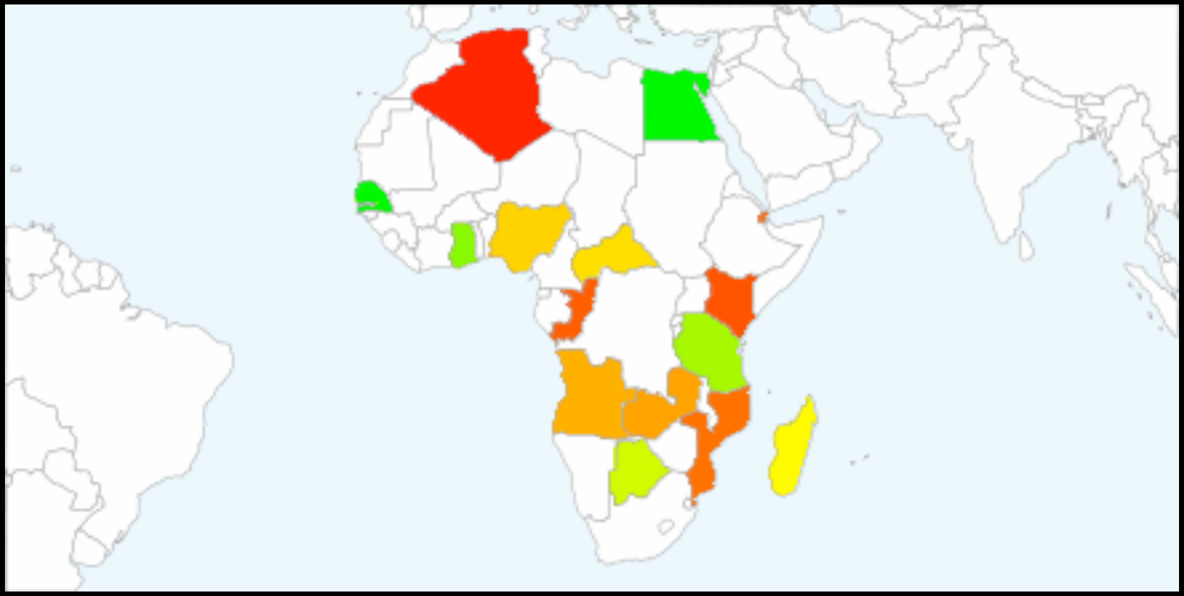











Google Chart API URLs

- All URLs requests must be in the following format...

```
http://chart.apis.google.com/chart?<parameter 1>&<parameter 2>&<parameter n>
```

- All chart requests must provide at least the following 3 pieces of information...
  - chs — the size of the chart
  - chd — the data to be rendered in the chart
  - cht — the type of chart

Documentation

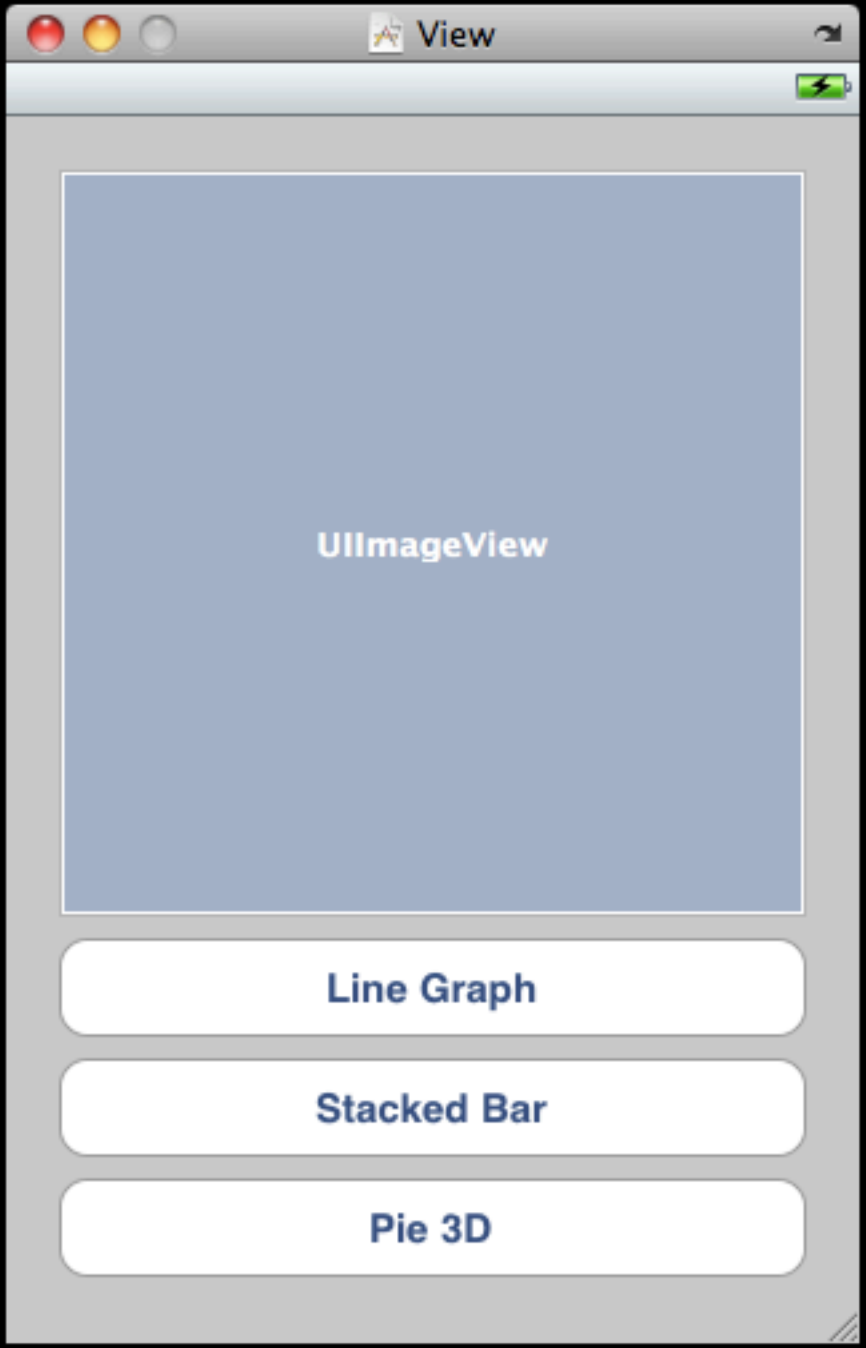
- Chart Basics
  - http://code.google.com/apis/chart/basics.html
- Chart Types
  - http://code.google.com/apis/chart/types.html
- Data Formats
  - http://code.google.com/apis/chart/formats.html

Example Request

- For example...
  - http://chart.apis.google.com/chart?chs=250x100&chd=t:60,40&cht=p3&chl=Hello|World
- Translates to...
  - http://chart.apis.google.com/chart? — chart API location
  - & — separates parameters
  - chs=250x100 — the chart's size in pixels
  - chd=t:60,40 — the chart's data
  - cht=p3 — the chart's type
  - chl=Hello|World — the chart's label

GoogleChartsDemoViewController.xib

- In the NIB, I've added an image view where we'll display the image
- 3 buttons, each calling a different action method



GoogleChartsDemoViewController.h

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

@interface GoogleChartsDemoViewController : UIViewController {

  UIImageView *imageView;

}

@property(nonatomic, retain) IBOutlet UIImageView *imageView;

- (IBAction)showLineGraph;
- (IBAction)showStackedBar;
- (IBAction)showPie3D;

@end
```

GoogleChartsDemoViewController.m

```
#import "GoogleChartsDemoViewController.h"

#define CHART_API_ROOT @"http://chart.apis.google.com/chart?"

@implementation GoogleChartsDemoViewController

@synthesize imageView;

- (UIImage *)imageFromURL:(NSString *)path {
  NSURL *url = [NSURL URLWithString:path];
  NSData *data = [NSData dataWithContentsOfURL:url];
  return [UIImage imageWithData:data];
}

- (NSString *)urlFromDictionary:(NSDictionary *)dict {
  NSMutableString *str = [NSMutableString stringWithString:CHART_API_ROOT];
  for (NSString *key in [dict allKeys]) {
    [str appendFormat:@"%@=%@&", key, [dict objectForKey:key]];
  }
  return str;
}

// ...
```

GoogleChartsDemoViewController.m

```
// ...

- (NSString *)imageViewSizeAsString {
  return [NSString stringWithFormat:@"%dx%d",
          (int)self.imageView.bounds.size.width,
          (int)self.imageView.bounds.size.height];
}



// ...
```

GoogleChartsDemoViewController.m

```
// ...

- (IBAction)showLineGraph {
    // setup data
    NSMutableArray *values = [NSMutableArray array];
    for (int i = 0; i < 10; i++) {
       [values addObject:[NSNumber numberWithInt:50 + arc4random() % 10]];
    }
    NSString *data = [values componentsJoinedByString:@","];

    // setup graph options
    NSMutableDictionary *options = [NSMutableDictionary dictionary];
    [options setValue:@"ls" forKey:@"cht"];
    [options setValue:[self imageViewSizeAsString] forKey:@"chs"];
    [options setValue:[NSString stringWithFormat:@"t:%@", data] forKey:@"chd"];

    // make request
    self.imageView.image = [self imageFromURL:[self urlFromDictionary:options]];
}

// ...
```

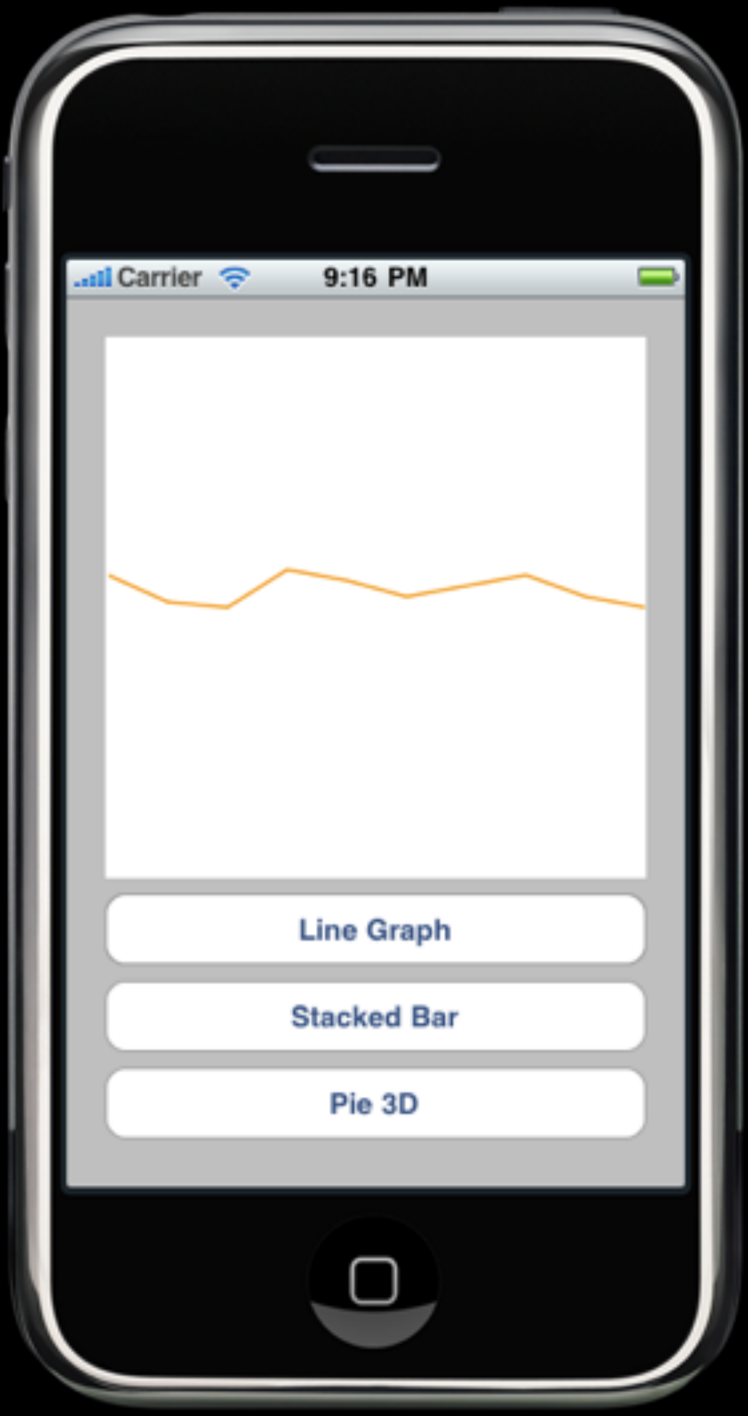
GoogleChartsDemoViewController.m

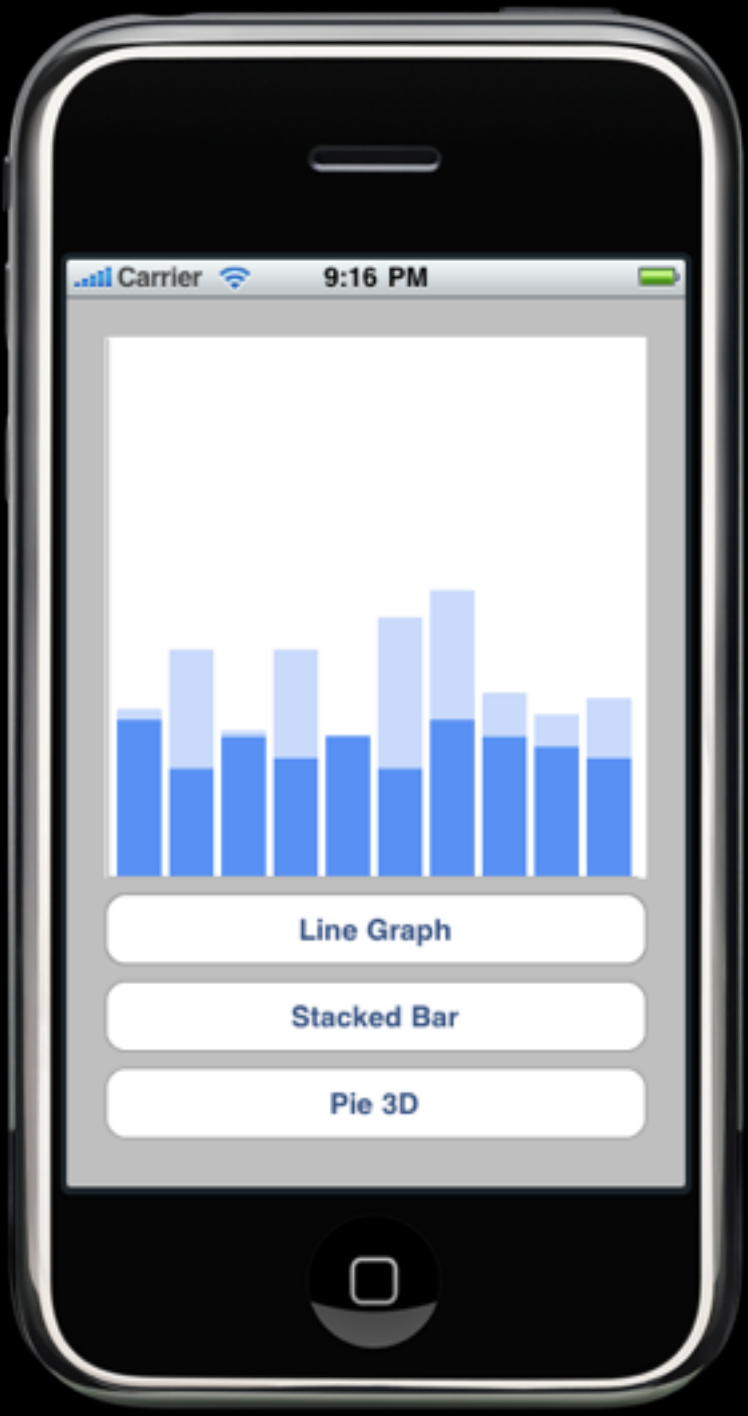
```
// ...

- (IBAction)showStackedBar {
  // setup data
  NSMutableArray *series1 = [NSMutableArray array];
  NSMutableArray *series2 = [NSMutableArray array];
  for (int i = 0; i < 10; i++) {
    [series1 addObject:[NSNumber numberWithInt:20 + arc4random() % 10]];
    [series2 addObject:[NSNumber numberWithInt:arc4random() % 30]];
  }
  NSString *data1 = [series1 componentsJoinedByString:@","];
  NSString *data2 = [series2 componentsJoinedByString:@","];

  // setup graph options
  NSMutableDictionary *options = [NSMutableDictionary dictionary];
  [options setValue:@"bvs" forKey:@"cht"];
  [options setValue:[self imageViewSizeAsString] forKey:@"chs"];
  [options setValue:@"4D89F9,C6D9FD" forKey:@"chco"];
  [options setValue:[NSString stringWithFormat:@"t:%@%%7c%@", data1, data2]
           forKey:@"chd"];

    // make request
    self.imageView.image = [self imageFromURL:[self urlFromDictionary:options]];
}

// ...
```

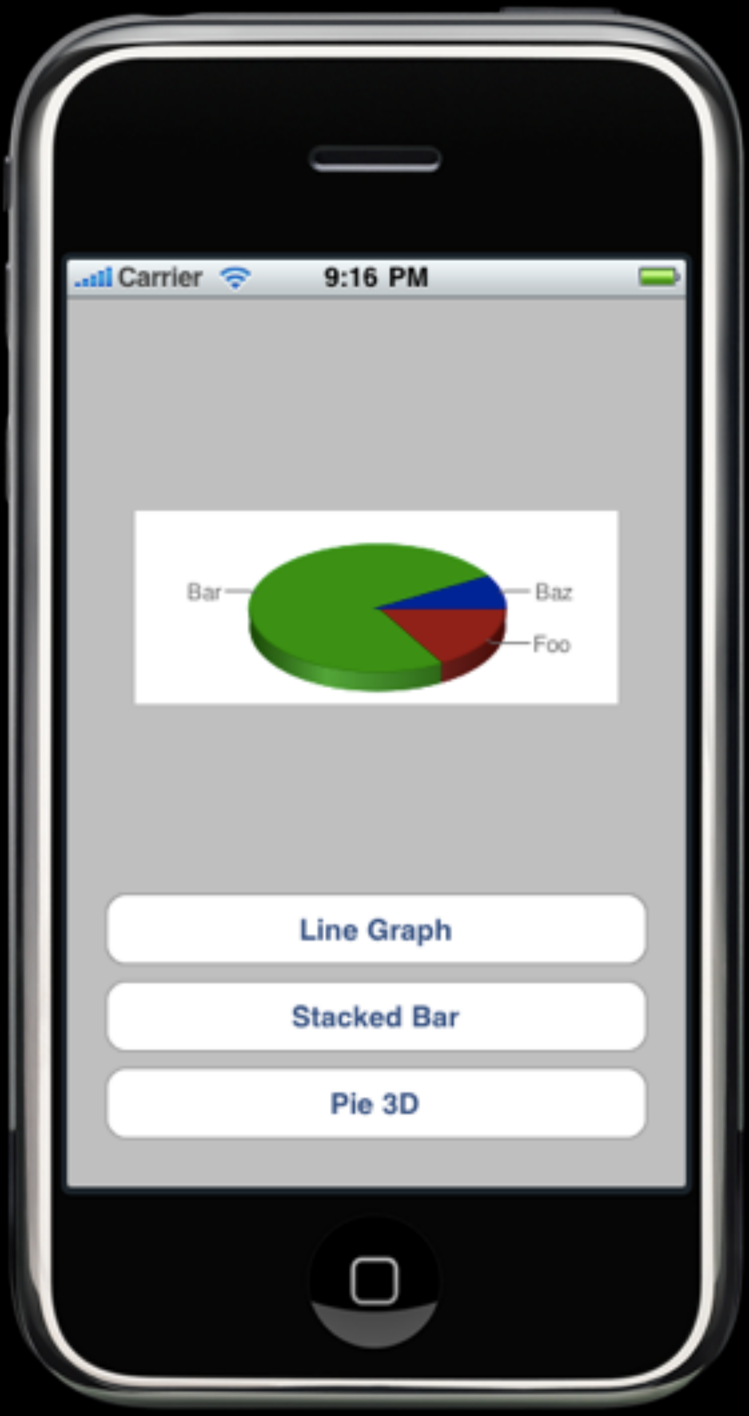
GoogleChartsDemoViewController.m

```objc
// ...

- (IBAction)showPie3D {
    // setup data
    NSArray *labels = [NSArray arrayWithObjects:@"Foo", @"Bar", @"Baz", nil];
    NSMutableArray *values = [NSMutableArray array];
    for (int i = 0; i < 3; i++) {
       [values addObject:[NSNumber numberWithInt:arc4random() % 10]];
    }
    NSString *data = [values componentsJoinedByString:@","];

    // setup graph options
    NSMutableDictionary *options = [NSMutableDictionary dictionary];
    [options setValue:@"p3" forKey:@"cht"];
    [options setValue:@"250x100" forKey:@"chs"];
    [options setValue:@"990000,009900,000099" forKey:@"chco"];
    [options setValue:[labels componentsJoinedByString:@"%7c"] forKey:@"chl"];
    [options setValue:[NSString stringWithFormat:@"t:%@", data] forKey:@"chd"];

    // make request
    self.imageView.image = [self imageFromURL:[self urlFromDictionary:options]];
}

@end
```

The Resulting App







Cover Flow

Apple’s Cover Flow APIs

- The cover flow APIs used by the iPod app are unfortunately private APIs, as such use is prohibited

Open Cover Flow Implementations

- That said, there are cover flow re-implementations that only use the public APIs which are safe to use...
  - Flow Cover
    - http://www.chaosinmotion.com/flowcover.m
  - Open Flow
    - http://apparentlogic.com/openflow/
  - Others?

Open Flow

- Open Flow is licensed under the liberal MIT open source license (free to use and modify)
- You can currently get the code from github...
  - http://github.com/thefaj/OpenFlow
- The git repository contains both the library and a sample application as well

The Library

- The library consists of the following files...
  - AFItemView.[mh] — view representing a given item in the flow collection
  - AFOpenFlowConstants.h — constant declarations, duh!
  - AFOpenFlowView.[mh] — the UIView that wraps up the flow logic
  - AFUIImageReflection.[mh] — the code that handles generating the reflection for an item

AFOpenFlowView

- AFOpenFlowView must have an associated data source
- There's a protocol in the AFOpenFlowView.h which defines the data source methods...

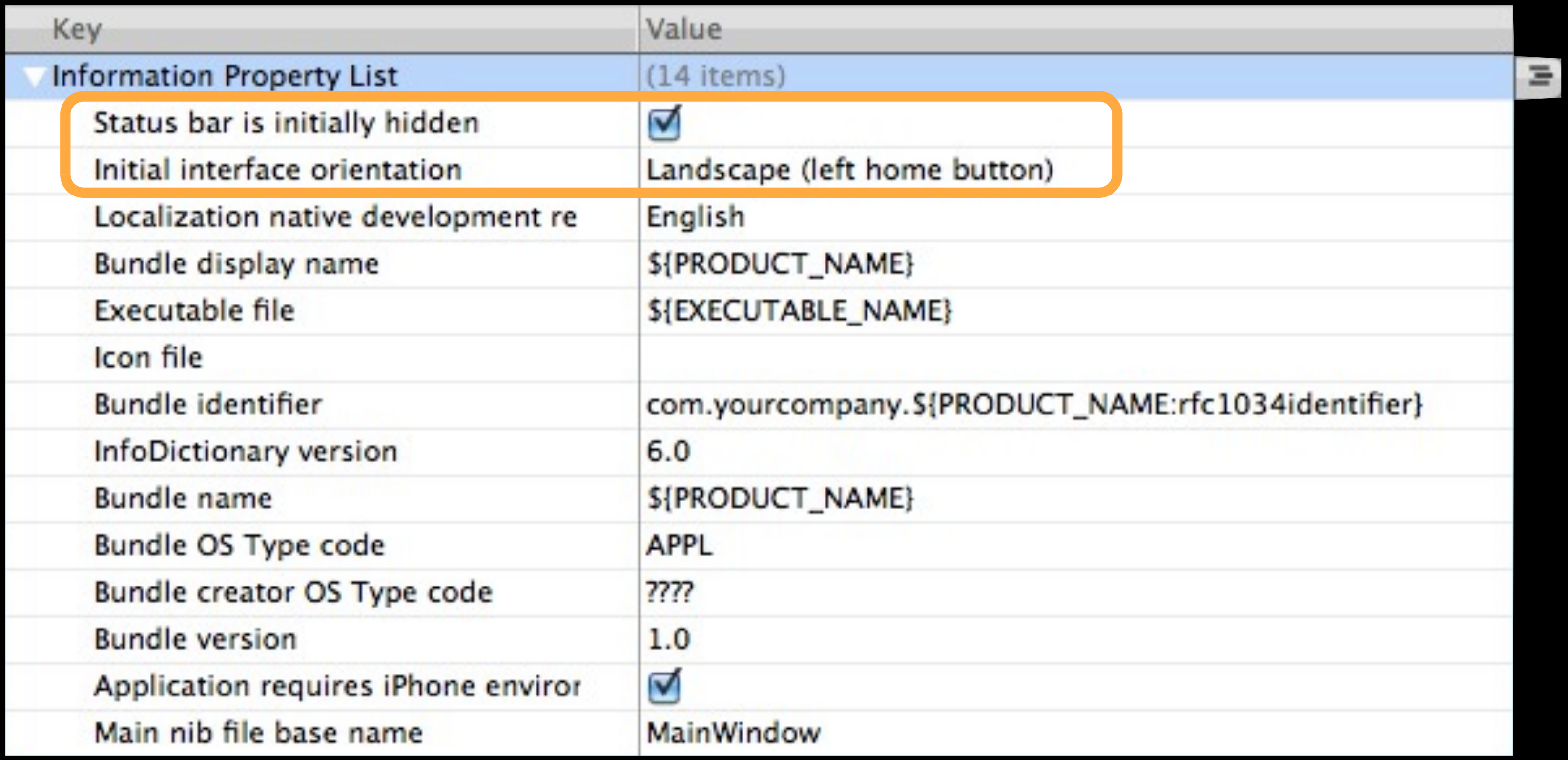
```
@protocol AFOpenFlowViewDataSource <NSObject>
- (void)openFlowView:(AFOpenFlowView *)openFlowView
requestImageForIndex:(int)index;
- (UIImage *)defaultImage;
@end
```

- There's also a delegate provided that allows you to hook into when an item is selected (centered)...

```
@protocol AFOpenFlowViewDelegate <NSObject>
@optional
- (void)openFlowView:(AFOpenFlowView *)openFlowView
  selectionDidChange:(int)index;
@end
```

Getting Started

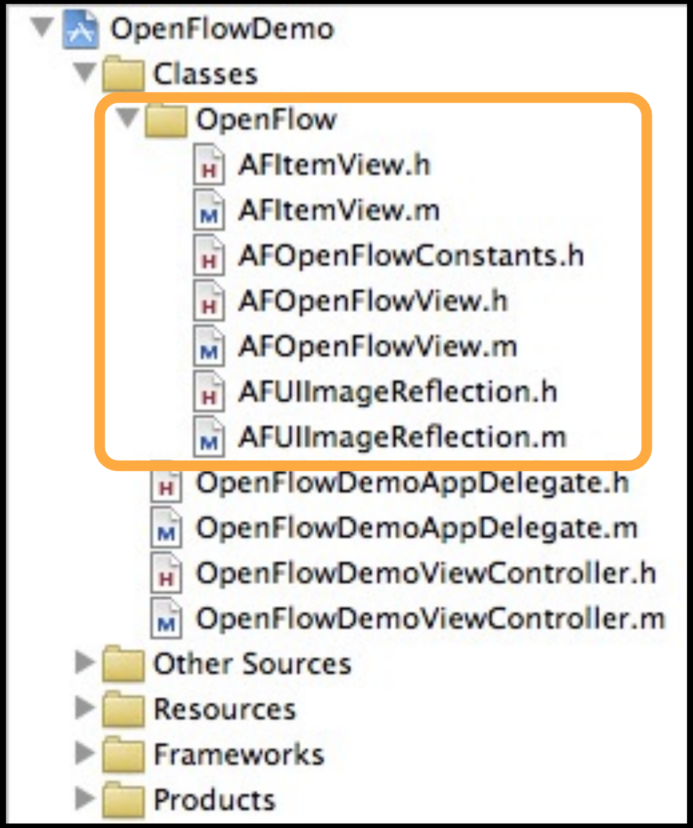
- For this example, I've created a new view-based project
- This flow library supports both landscape and portrait views
- We'll go ahead and change our app to be landscape and remove the status bar
  - Remember, we do this by editing the *-Info.plist...



Key	Value
Information Property List	(14 items)
Status bar is initially hidden	✓
Initial interface orientation	Landscape (left home button)
Localization native development re	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone enviror	✓
Main nib file base name	MainWindow

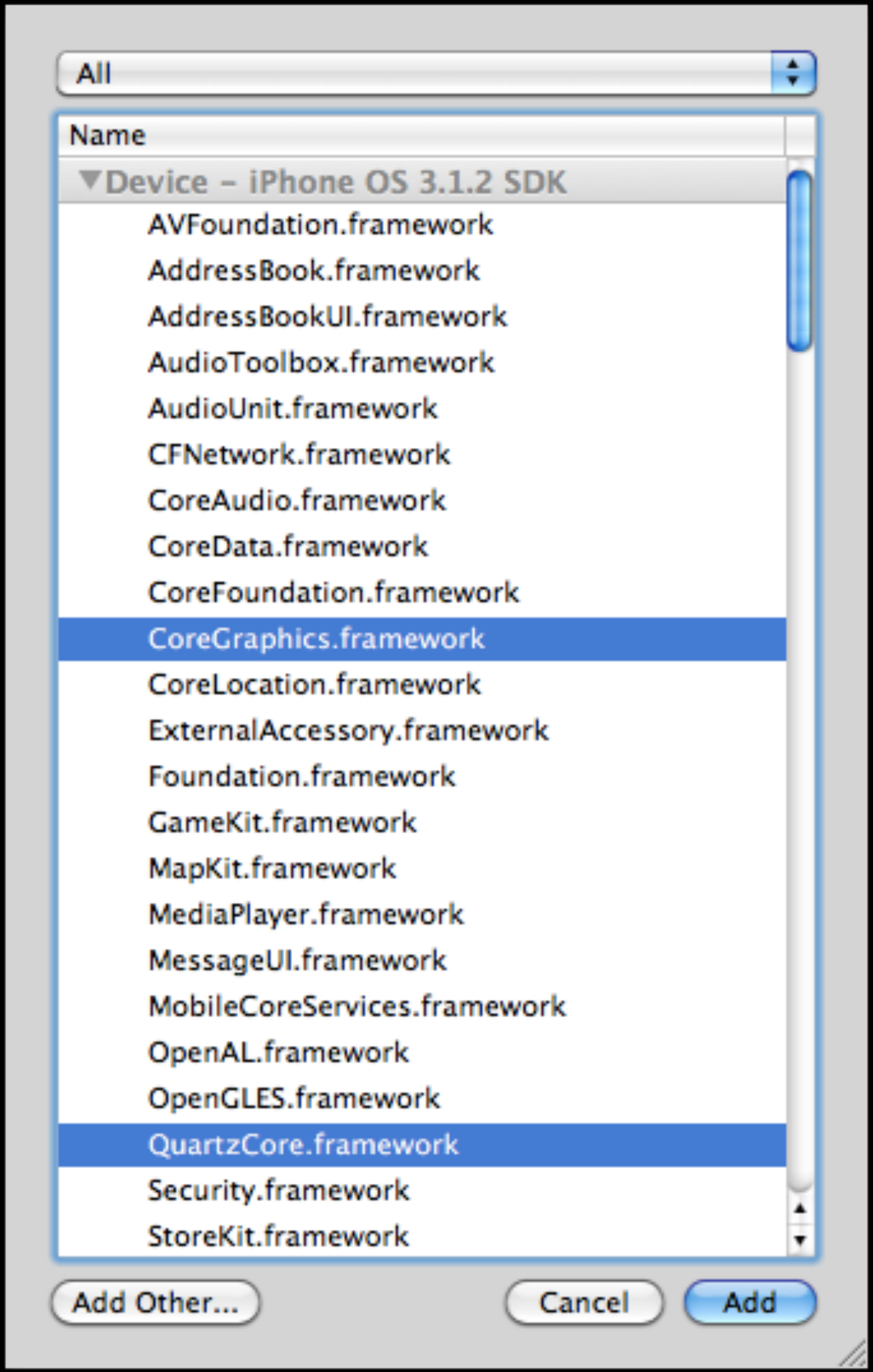
Including the Library

- Before we start coding, we need to first include the library into our app
- To do so, grab the OpenFlow directory from github and copy it into your classes directory



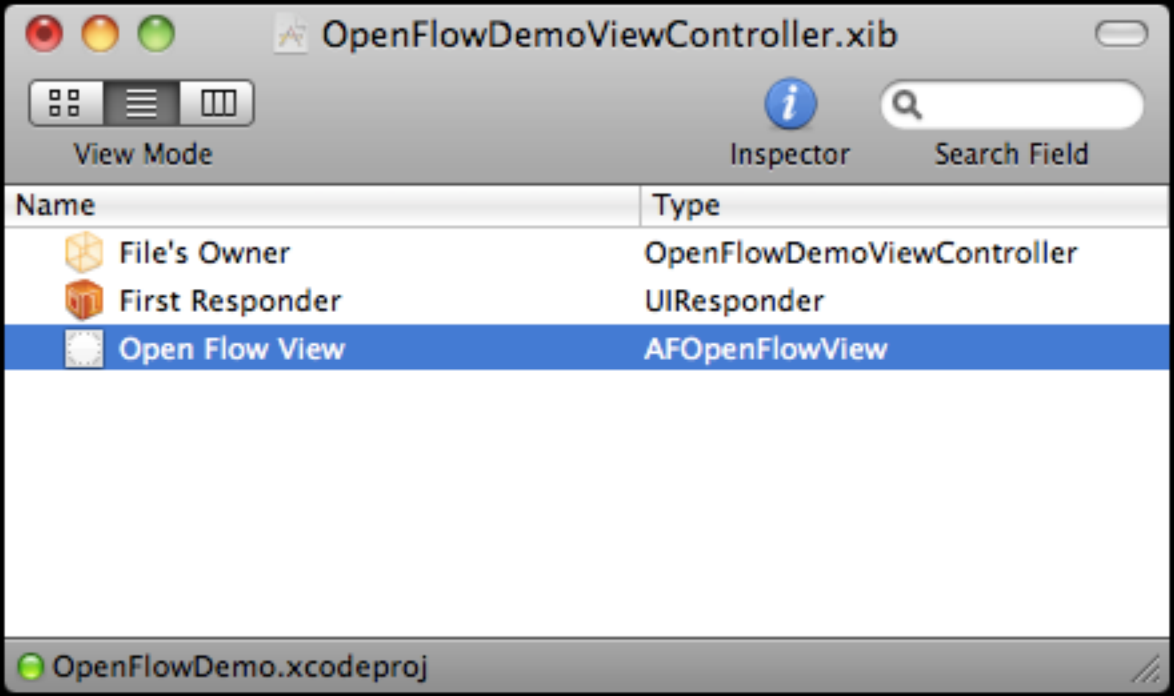
Including Dependent Frameworks

- The library is dependent upon both CoreGraphics and QuartzCore frameworks
- Include these frameworks if not already included in your app



OpenFlowDemoViewController.xib

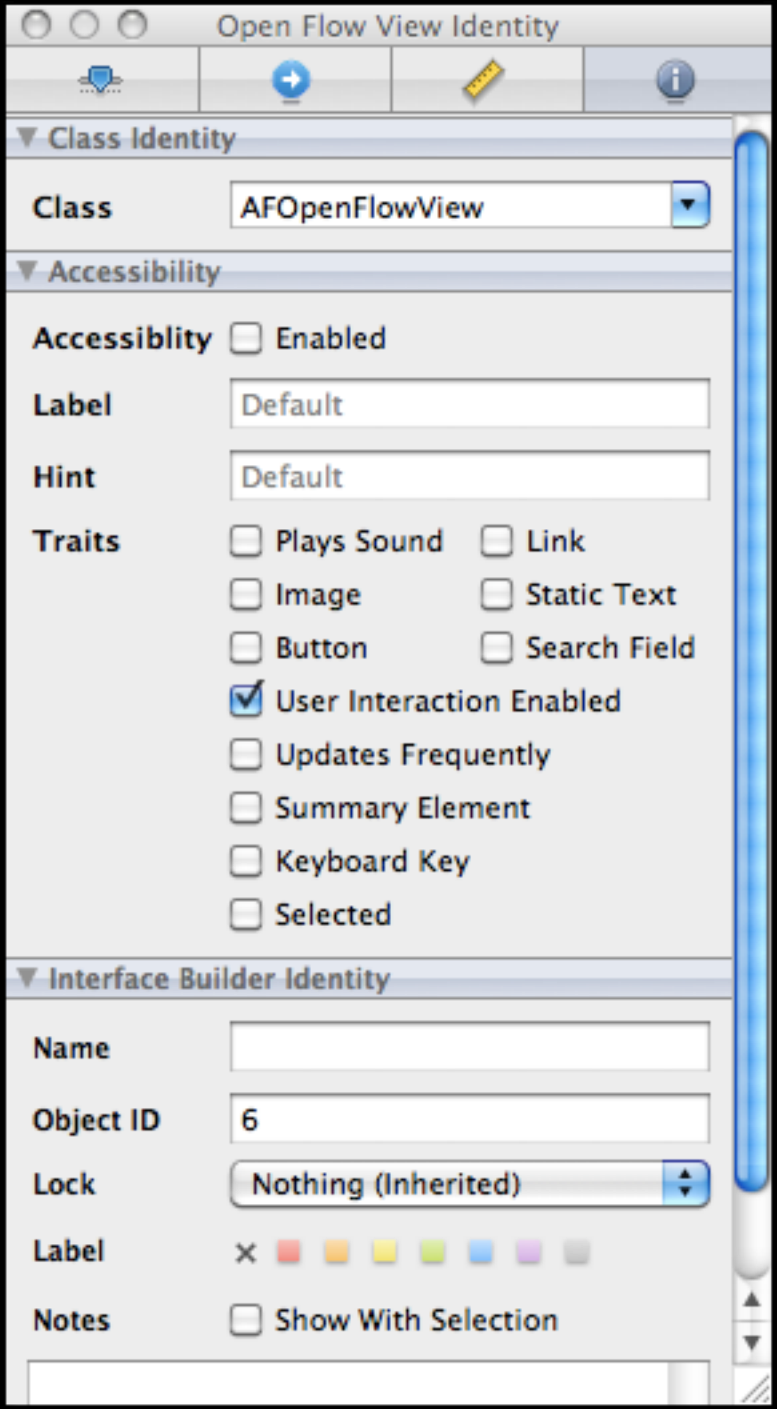
- Your controller's view must be changes to be an instance of AFOpenFlowView



OpenFlowDemoViewController.xib

Name	Type
File's Owner	OpenFlowDemoViewController
First Responder	UIResponder
Open Flow View	AFOpenFlowView

OpenFlowDemo.xcodeproj



Open Flow View Identity

Class Identity

Class: AFOpenFlowView

Accessibility

Accessiblity: ☐ Enabled

Label: Default

Hint: Default

Traits: ☐ Plays Sound ☐ Link ☐ Image ☐ Static Text ☐ Button ☐ Search Field [✓ User Interaction Enabled] ☐ Updates Frequently ☐ Summary Element ☐ Keyboard Key ☐ Selected

Interface Builder Identity

Name:

Object ID: 6

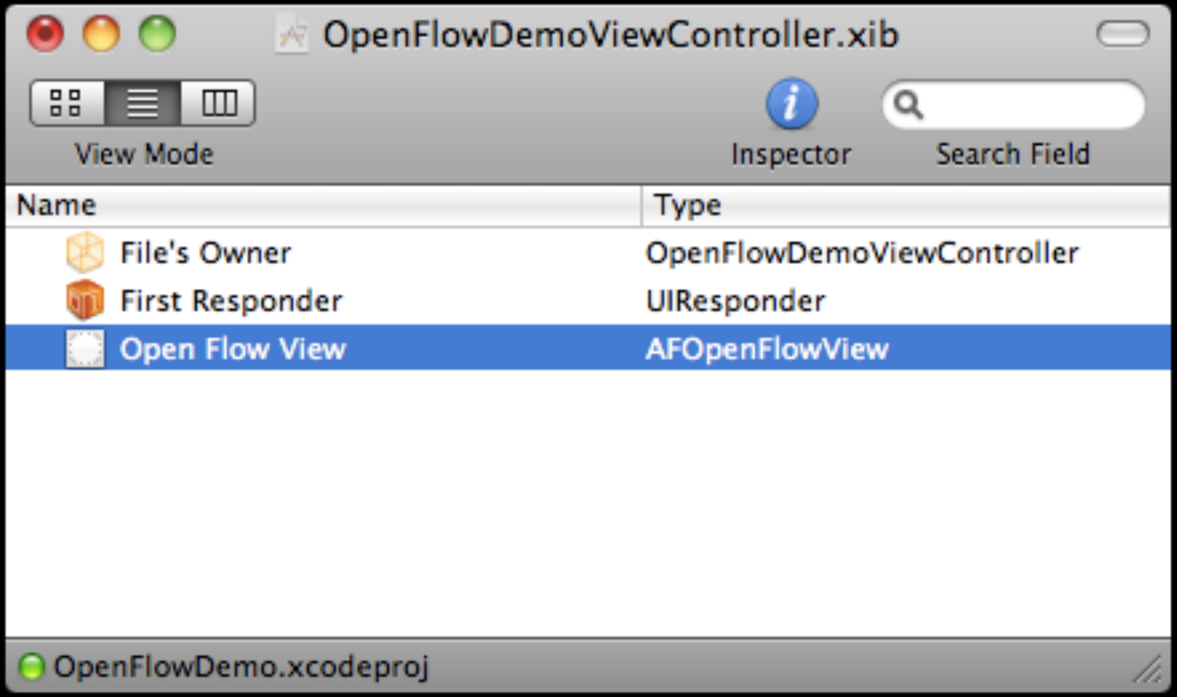
Lock: Nothing (Inherited)

Label: ✕

Notes: ☐ Show With Selection

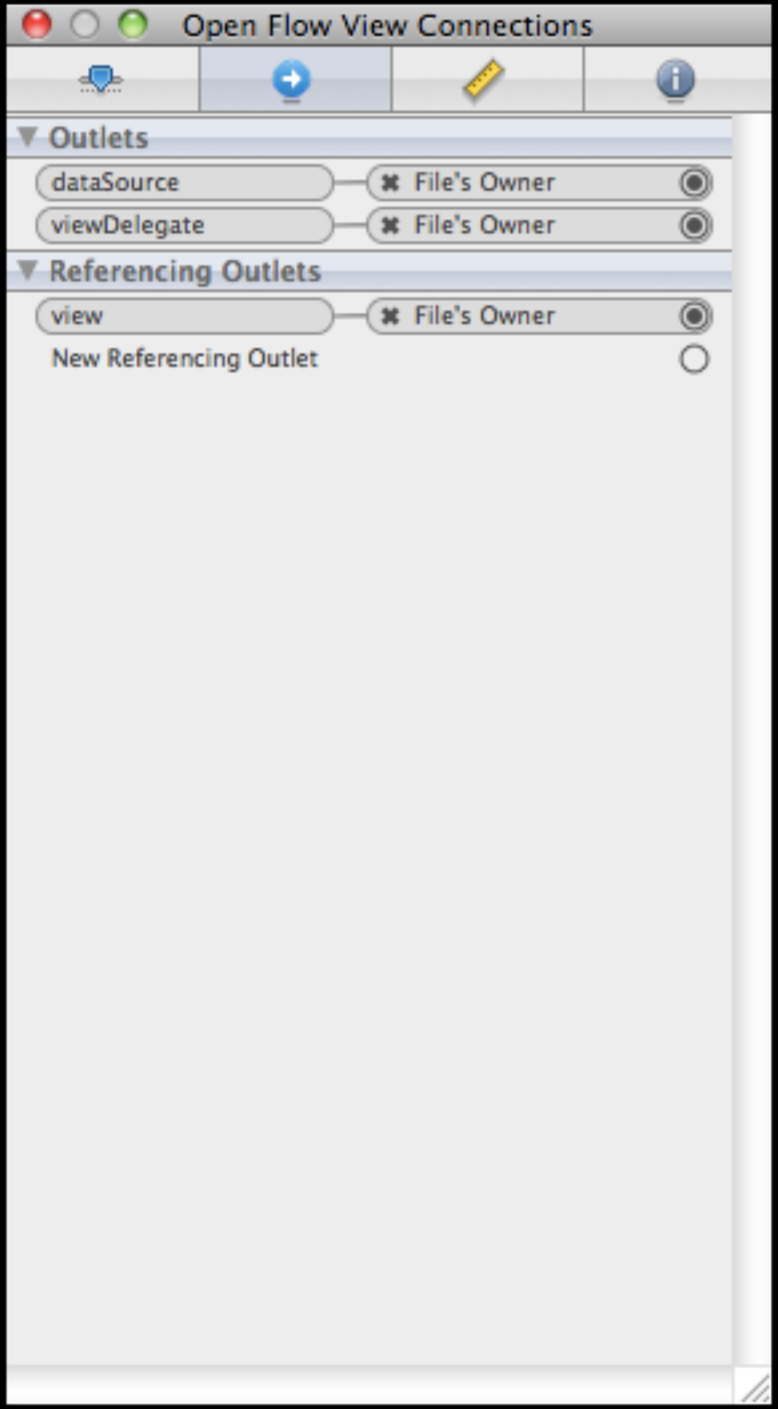
OpenFlowDemoViewController.xib

- While in IB, set the view's delegate and data sources to the view controller
- I changed the view's background color to black to match Apple's cover flow color scheme



Name	Type
File's Owner	OpenFlowDemoViewController
First Responder	UIResponder
Open Flow View	AFOpenFlowView

OpenFlowDemo.xcodeproj



Outlets

dataSource	File's Owner
viewDelegate	File's Owner

Referencing Outlets

view	File's Owner

New Referencing Outlet

OpenFlowDemoViewController.h

```objc
#import <UIKit/UIKit.h>
#import "AFOpenFlowView.h"

@interface OpenFlowDemoViewController : UIViewController <AFOpenFlowViewDataSource,
                                                           AFOpenFlowViewDelegate> {

}

@end
```

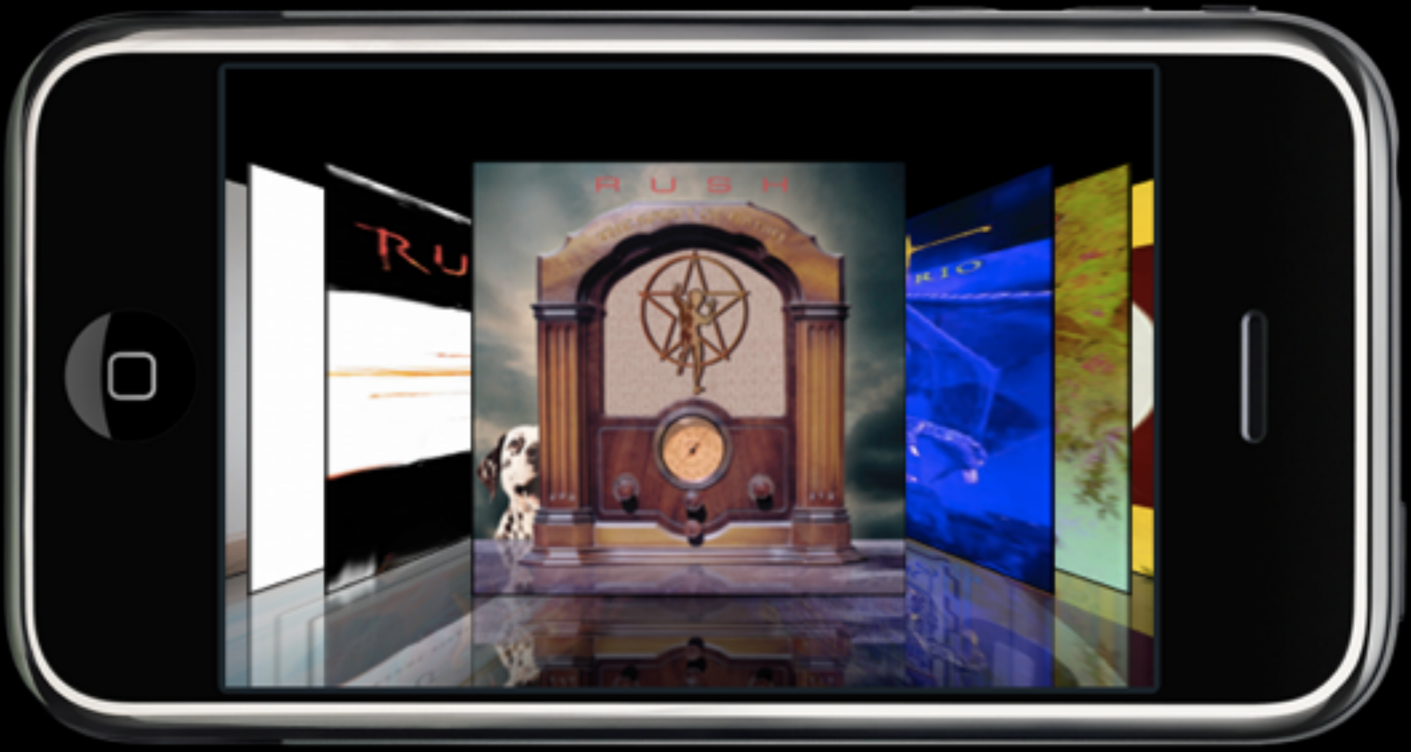
OpenFlowDemoViewController.m

```
#import "OpenFlowDemoViewController.h"

@implementation OpenFlowDemoViewController

- (void)openFlowView:(AFOpenFlowView *)openFlowView selectionDidChange:(int)index {
  NSLog(@"Selected item %d", index);
}

- (void)openFlowView:(AFOpenFlowView *)openFlowView
requestImageForIndex:(int)index {
  // Called if images not already set, so you could fetch
  // and set them on the view here if doing it on demand.
  //
  // We will just set them all ahead of time, thus avoid
  // implementing this method altogether.
}

- (UIImage *)defaultImage {
  return [UIImage imageNamed:@"default.png"];
}

// ...
```

OpenFlowDemoViewController.m

```
// ...

// preload all images
- (void)viewDidLoad {
  [super viewDidLoad];
  for (int i=0; i < 30; i++) {
    NSString *name = [NSString stringWithFormat:@"%d.png", i];
    [(AFOpenFlowView *)self.view setImage:[UIImage imageNamed:name] forIndex:i];
  }
  [(AFOpenFlowView *)self.view setNumberOfImages:30];
}

// need to override since we are in landscape mode
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
                                                               interfaceOrientation {
  // Return YES for supported orientations
  return (interfaceOrientation == UIInterfaceOrientationLandscapeLeft);
}

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

@end
```

The Resulting App



Additional Resources

- Core Plot
  - http://code.google.com/p/core-plot/wiki/UsingCorePlotInApplications
  - http://code.google.com/p/core-plot/wiki/HighLevelDesignOverview
- Google Charts
  - http://code.google.com/apis/chart/
- Open Flow
  - http://apparentlogic.com/openflow/
  - http://fajkowski.com/blog/2009/08/02/openflow-a-coverflow-api-replacement-for-the-iphone/