Chapter 4
Views and View Hierarchy

- Views and View Hierarchy
- Creating custom views

View Basics

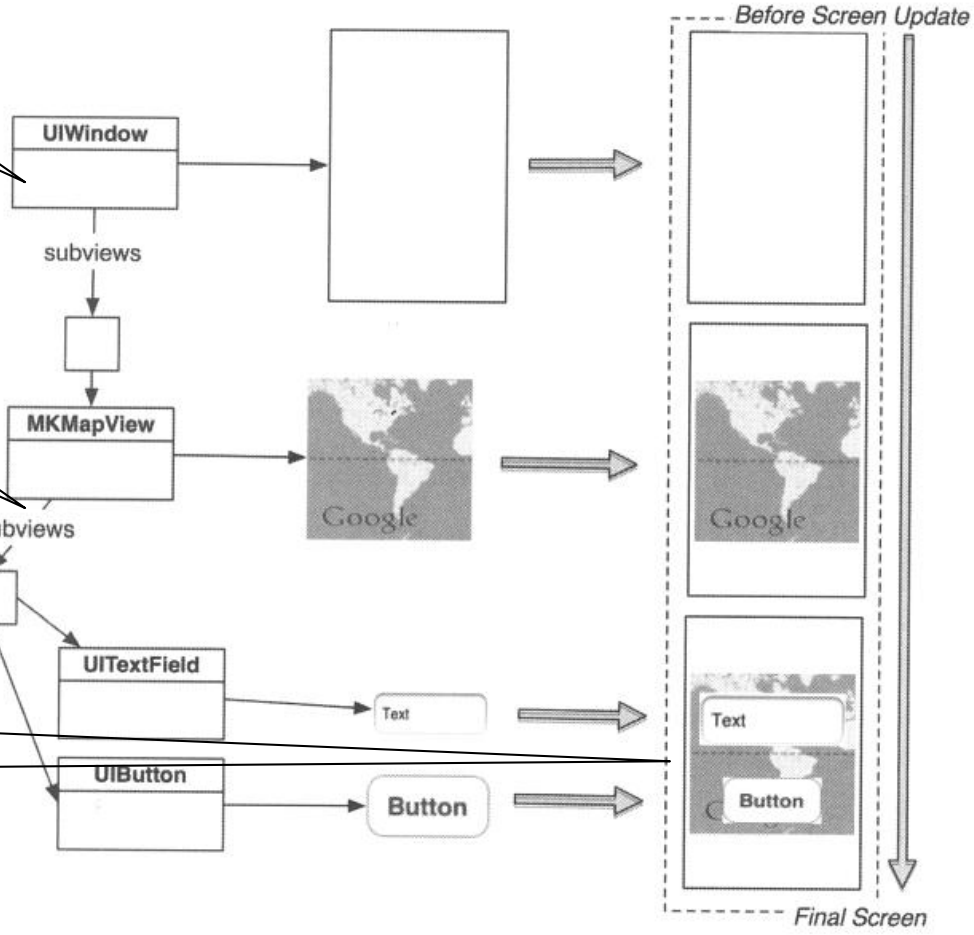
- A view is an instance of UIView or one of its subclass
- A view knows how to draw itself
- A view handles events, like touches and gestures
- A view exists within a hierarchy of views
- Views may be added to the application by the designer, similar to what we did in the Quiz application.
- Views may be generated programmatically by the application wham needed.

Views and View Hierarchy

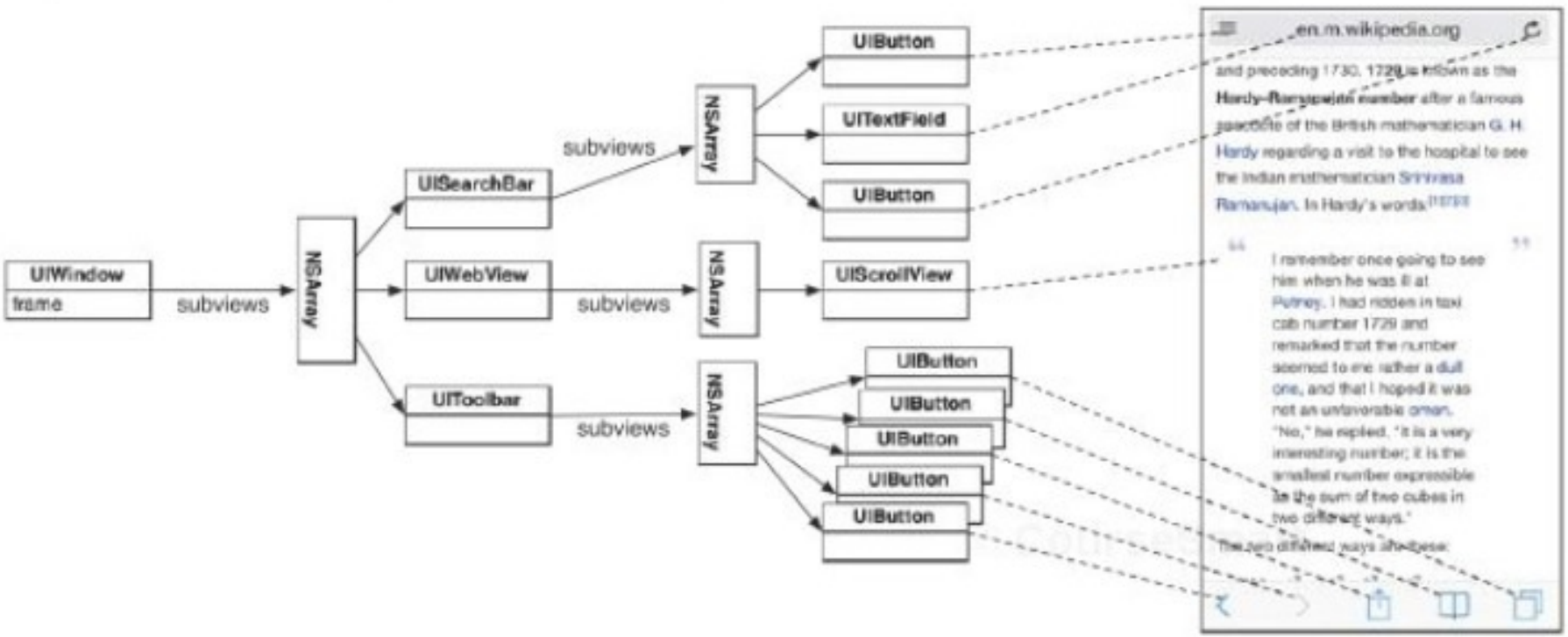
A cocoa touch application starts with a main window which is an instance of UIWindow

Subviews are layered on top of the main window to create the user interface.

Each subview will draw its own image and add it to the views hierarchy.

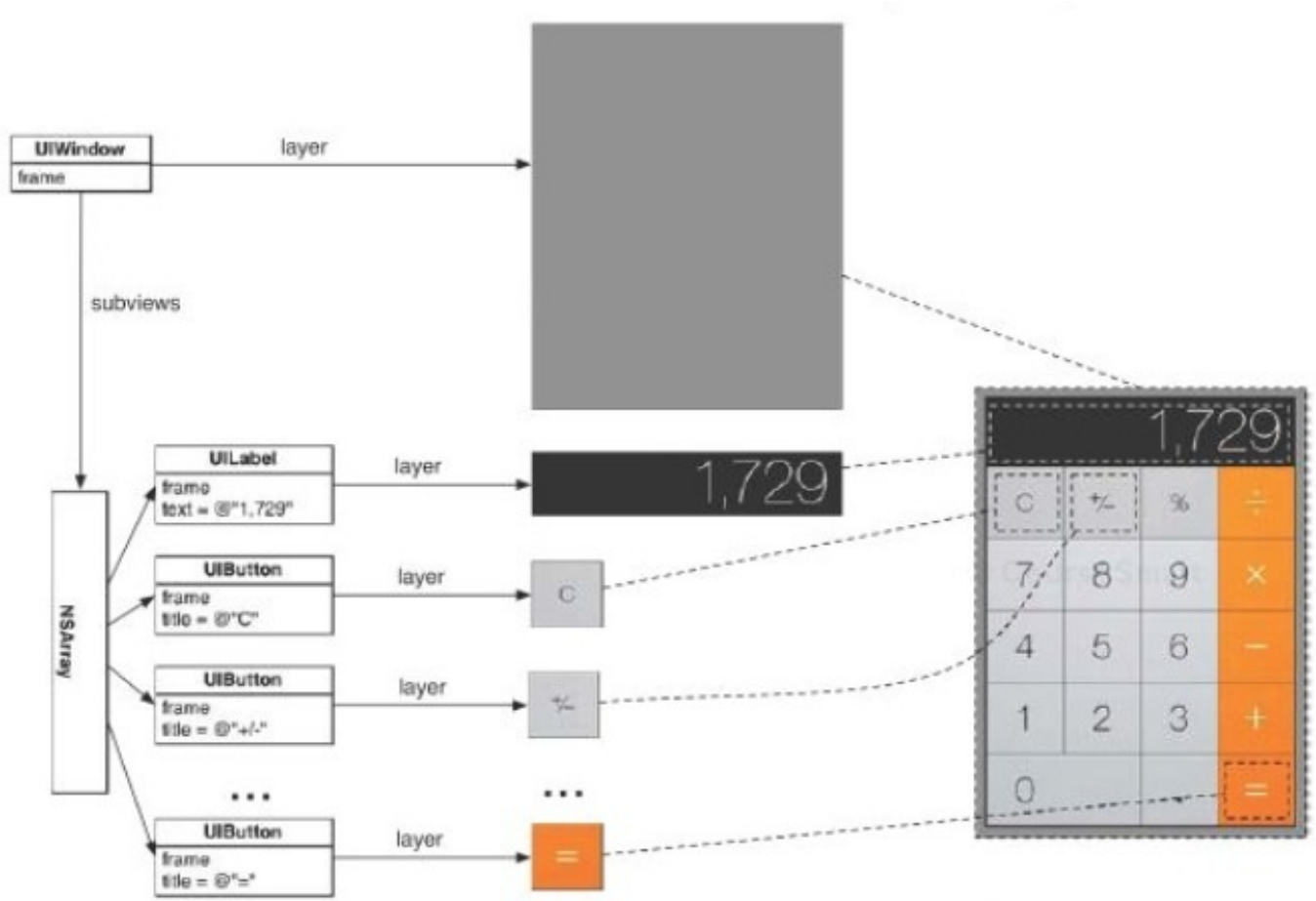


Sample View Hierarchy

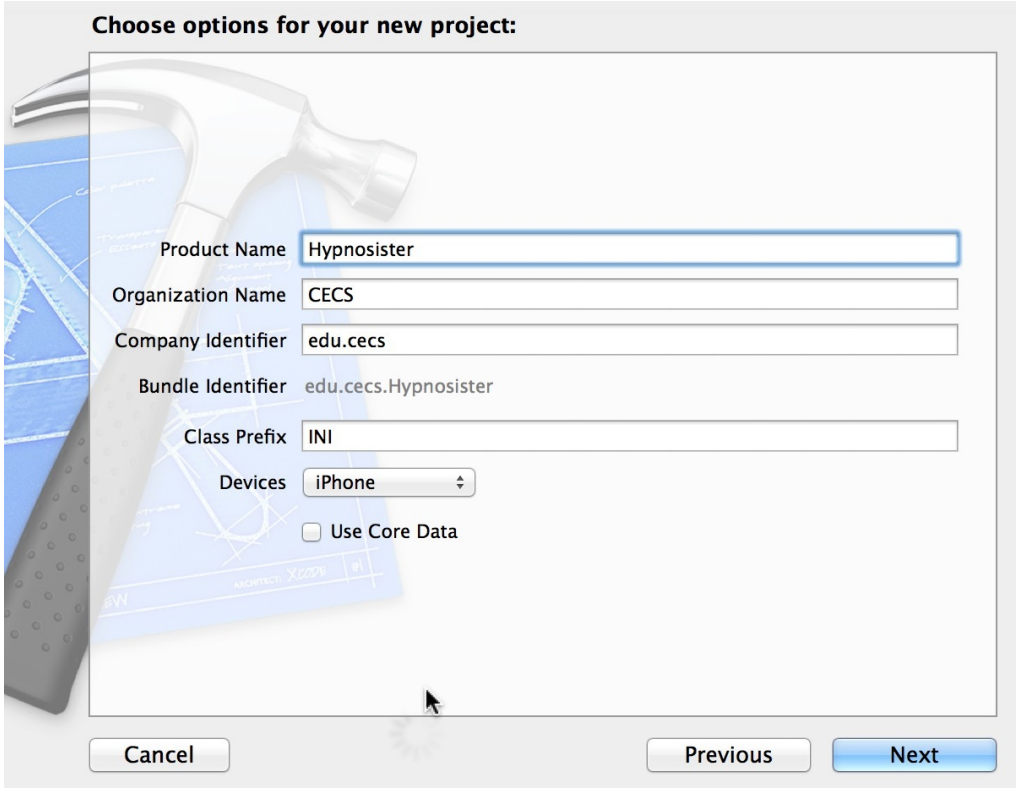


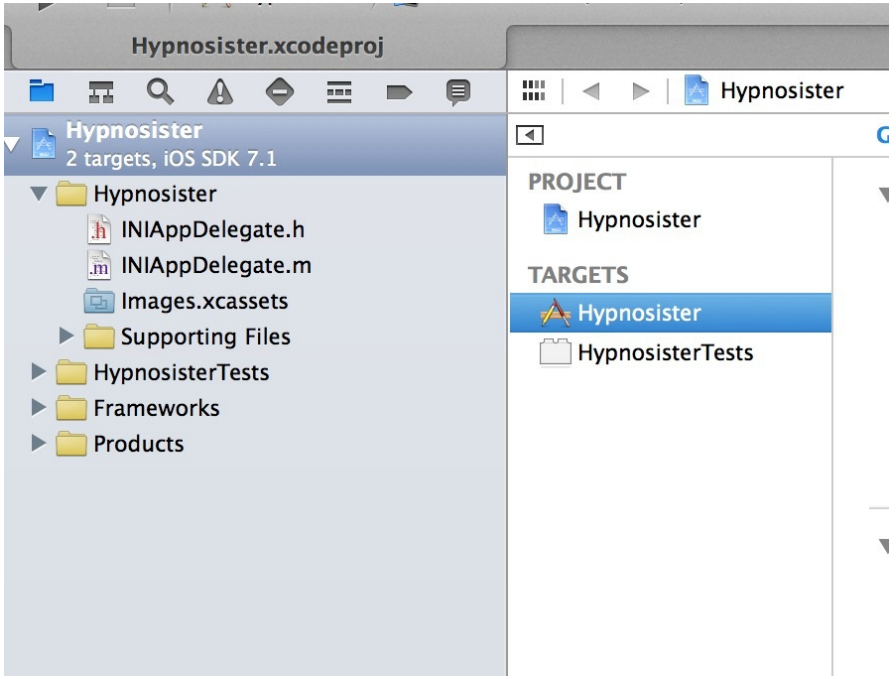
- Each view in the hierarchy, including the window, draws itself. It renders itself to its layer, which is an instance of CALayer. (You can think of a view's layer as a bitmap image.)
- The layers of all the views are composited together on the screen.

- Views render themselves and then are composited together

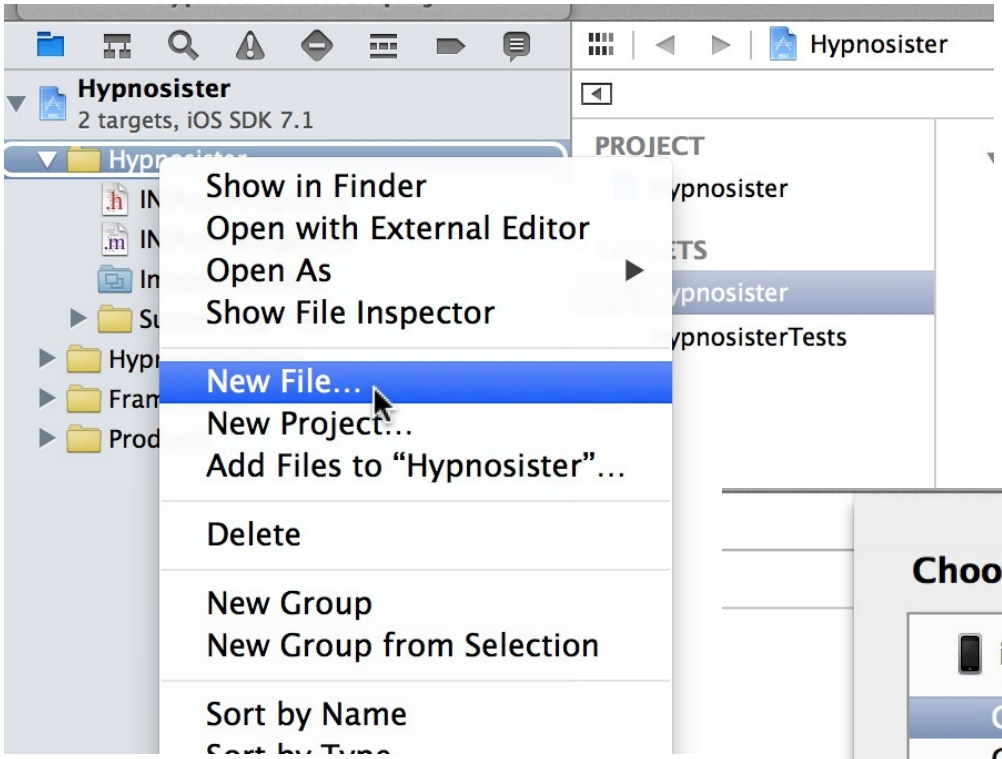


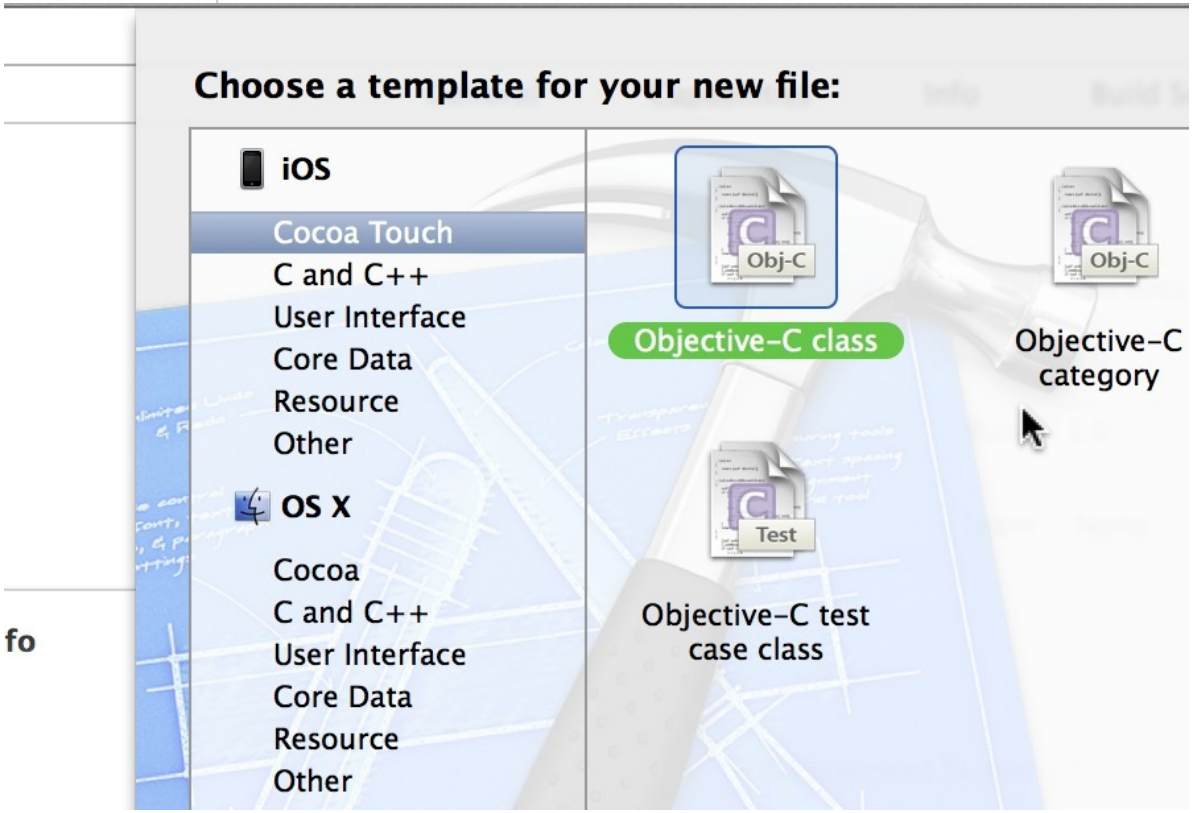
Creating Hypnosister

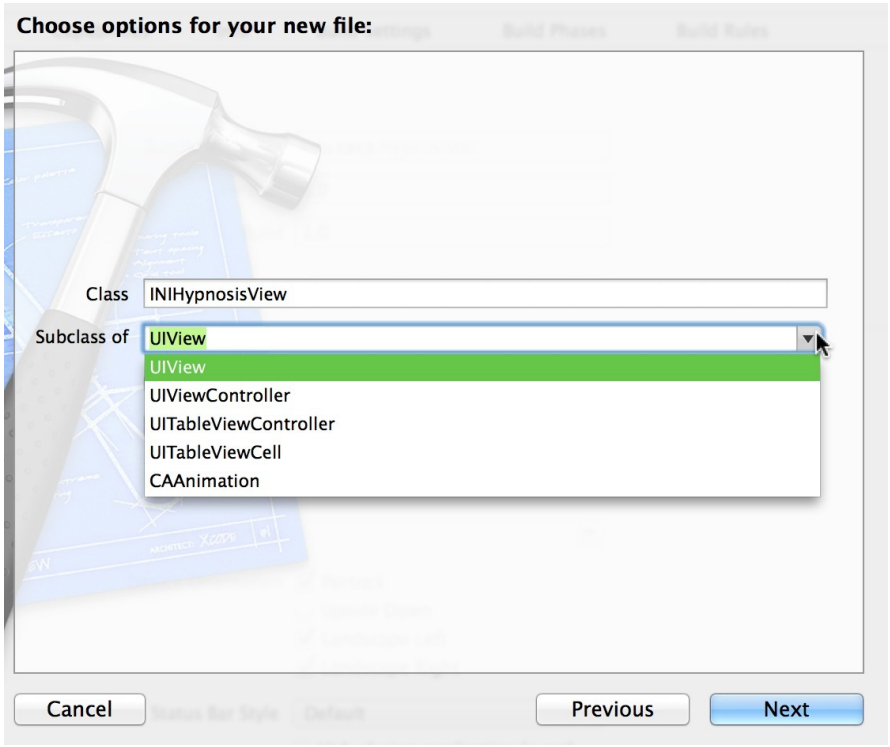


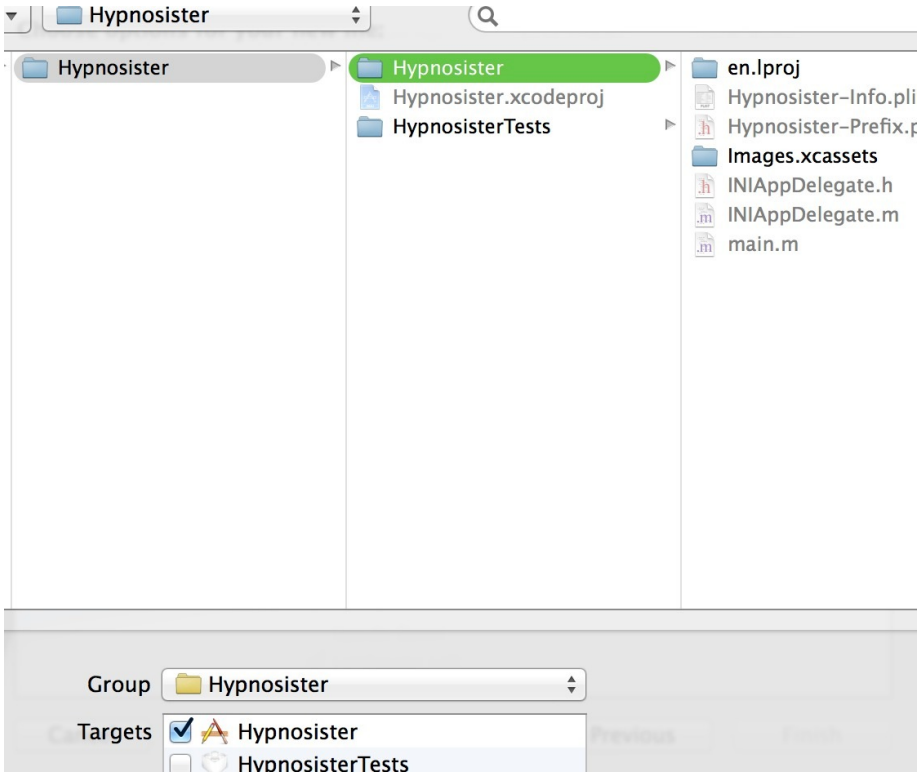


Creating INIHypnosisView






Choose options for your new file:
Class
INIHypnosisView
Subclass of
UIView
UIView
UIViewController
UITableViewController
UITableViewCell
CAAnimation
Cancel
Previous
Next


Hypnosister
Hypnosister
Hypnosister
Hypnosister.xcodeproj
HypnosisterTests
en.lproj
Hypnosister-Info.pli
Hypnosister-Prefix.p
Images.xcassets
INIAppDelegate.h
INIAppDelegate.m
main.m
Group
Hypnosister
Targets
Hypnosister
HypnosisterTests

Views and Frames

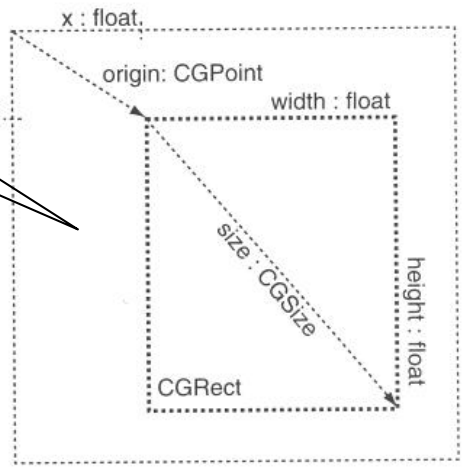
Hypnosister 〉 Hypnosister 〉 INIHypnosisView.m 〉 @implementation INIHyp

```
1  //
2  //  INIHypnosisView.m
3  //  Hypnosister
4  //
5  //  Created by Ibrahim Imam on 5/19/14.
6  //  Copyright (c) 2014 CECS. All rights reserved.
7  //
8
9  #import "INIHypnosisView.h"
10
11 @implementation INIHypnosisView
12
13 - (id)initWithFrame:(CGRect)frame
14 {
15     self = [super initWithFrame:frame];
16     if (self) {
17         // Initialization code
18     }
19     return self;
20 }
21
22 /*
23 // Only override drawRect: if you perform custom drawing.
24 // An empty implementation adversely affects performance during animation.
25 - (void)drawRect:(CGRect)rect
26 {
27     // Drawing code
28 }
29 */
30
31 @end
32
```

CGRect

CGRect

Designated Initializer



- A CGRect contains the members origin and size.
- The origin is a C structure of type CGPoint and contains two float members: x and y.
- The size is a C structure of type CGSize and has two float members: width and height

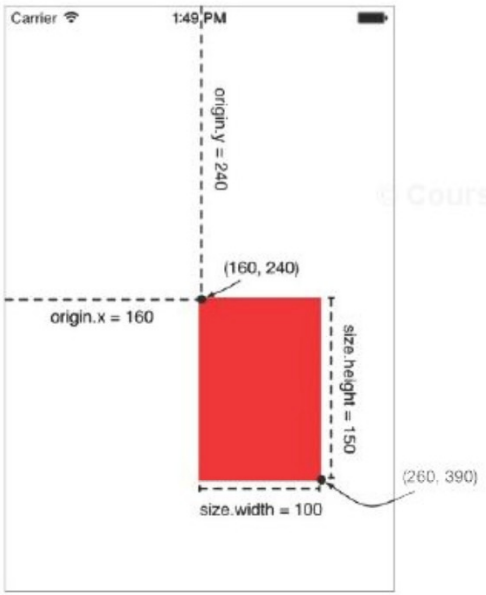
Creating the main window and CGRect

- A view's frame specifies the view's size and its position relative to its superview.
- A view's size is always specified by its frame and a view is always a rectangle.
- The AppDelegate.m usually implements the method "application:didFinishLaunchingWithOptions:" to create the main window of the application.

Making a Struct

Creating a subview and adding it to the main window

```
13
14 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
15 {
16     self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
17     CGRect firstFrame = CGRectMake( 160, 240, 100, 150);
18     INIHypnosisView *firstView = [[INIHypnosisView alloc] initWithFrame: firstFrame];
19     firstView.backgroundColor = [ UIColor redColor];
20     [self.window addSubview: firstView];
21     self.window.backgroundColor = [UIColor whiteColor];
22     [self.window makeKeyAndVisible];
23     return YES;
24 }
```

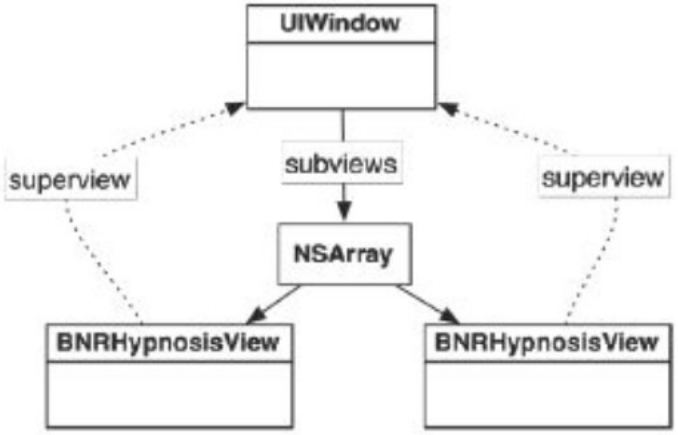


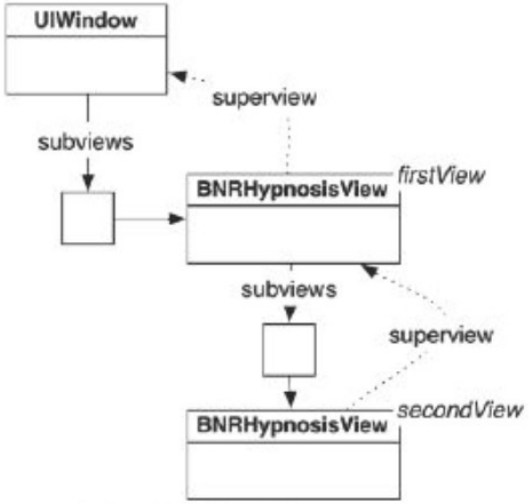
View Controllers

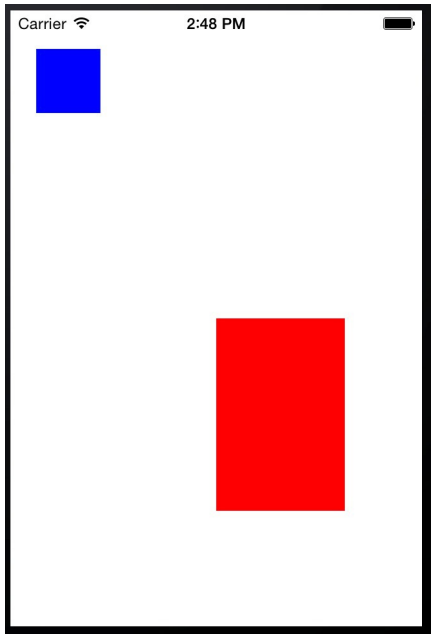
```
2014-05-25 14:18:04.688 Hypnosister[984:60b] Application windows are expected to have a root view controller at the end of application launch
```

- A view controller is an object that controls some set of an application's view hierarchy
- Most iOS apps have one or more view controllers.
- Hypnosister is simple enough that it does not need a view controller, so you can ignore this comment.

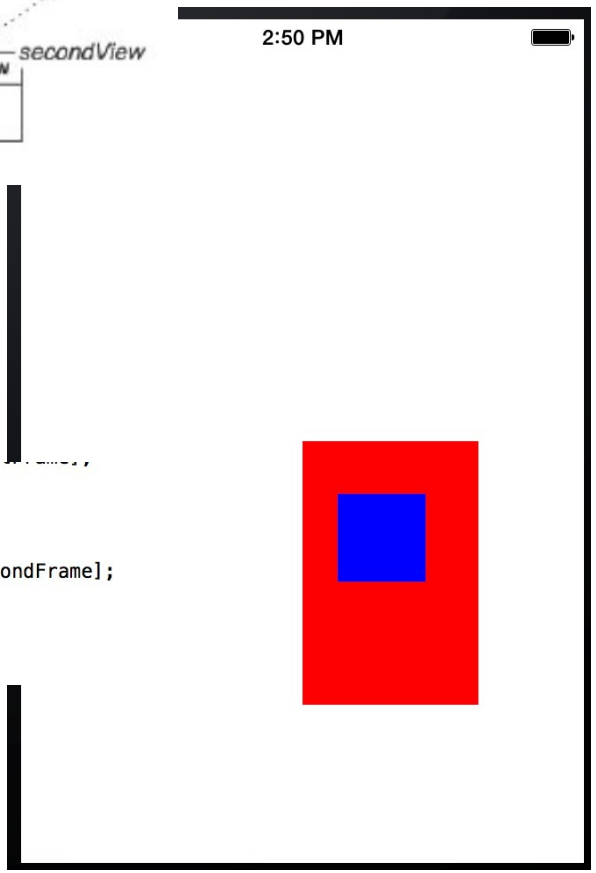
Adding More Subviews







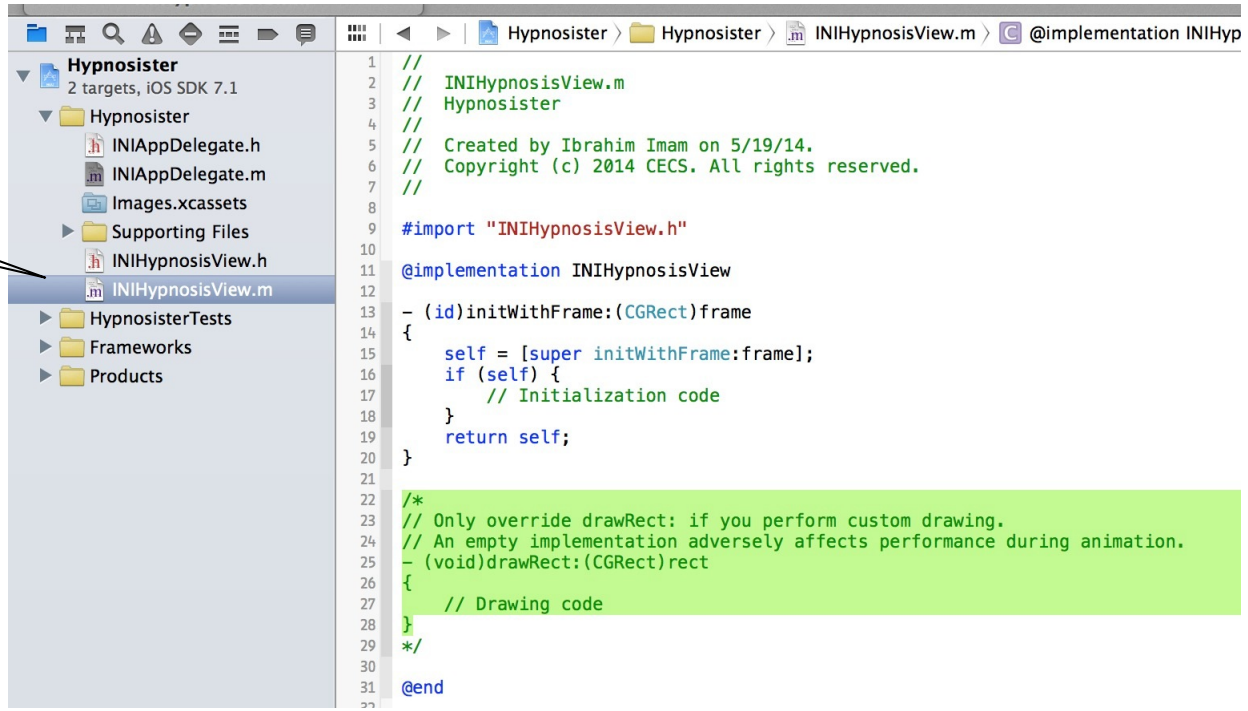

```
firstView.backgroundColor = [ UIColor redColor];
[self.window addSubview: firstView];

CGRect secondFrame = CGRectMake( 20, 30, 50, 50);
INIHypnosisView *secondView = [[INIHypnosisView alloc] initWithFrame: secondFrame];
secondView. backgroundColor = [UIColor blueColor];

// [self.window addSubview:secondView];
[firstView addSubview:secondView];
```



Custom Drawing using drawRect: Method

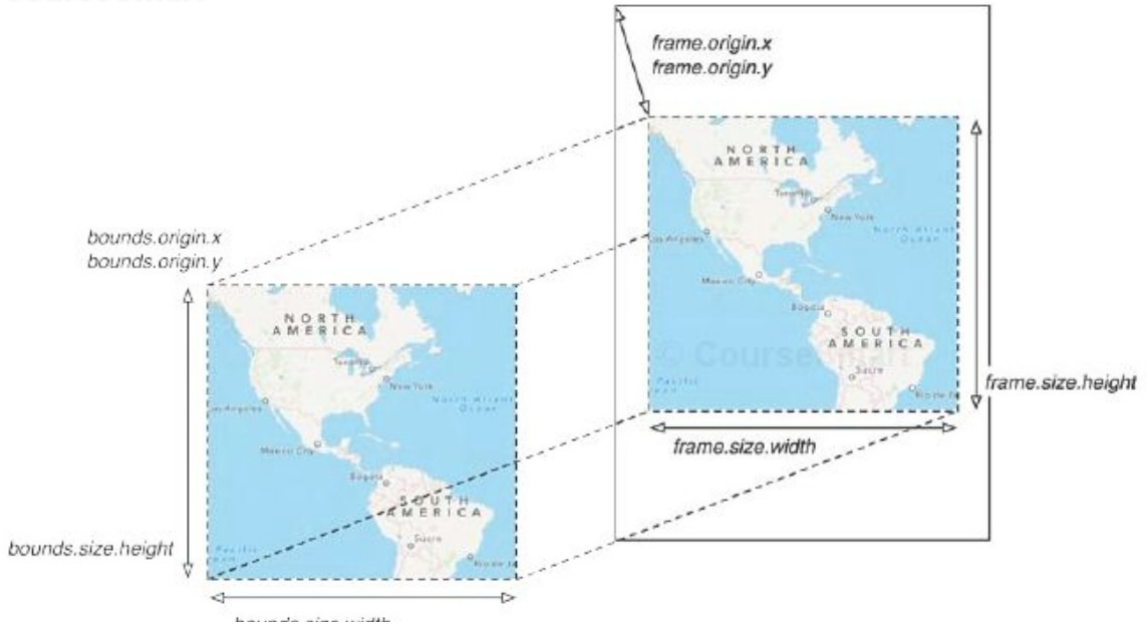
- In UIView subclasses that we create we use the method “drawRect:” to implement code needed to draw our image.
- “drawRect:” method is the rendering step a view uses to draw itself
- In the INIHypnosisView we use “drawRect:” to draw a series on concentric circles with a text image in the middle

Defining
NIHypnosisView



bounds vs. frame

- The first thing that you typically do when overriding drawRect: is get the bounds rectangle of the view.
- The frame and bounds rectangles have distinct purposes.
- A view's frame rectangle is used during compositing to lay out the view's layer relative to the rest of the view hierarchy.
- The bounds rectangle is used during the rendering step to lay out detailed drawing within the boundaries of the view's layer.

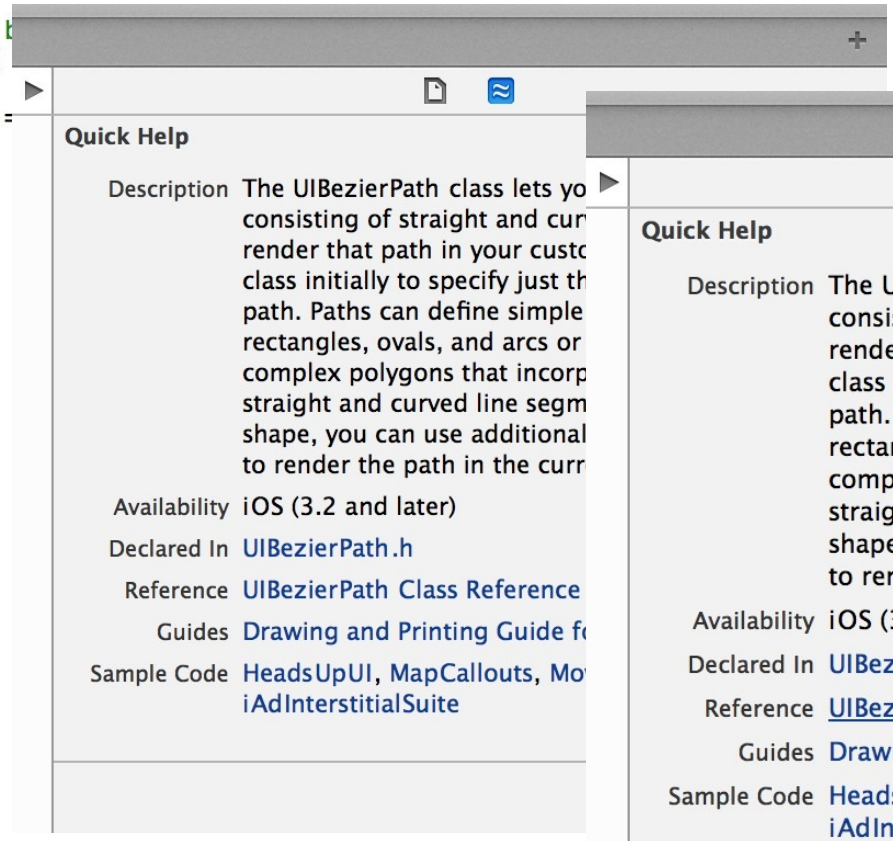


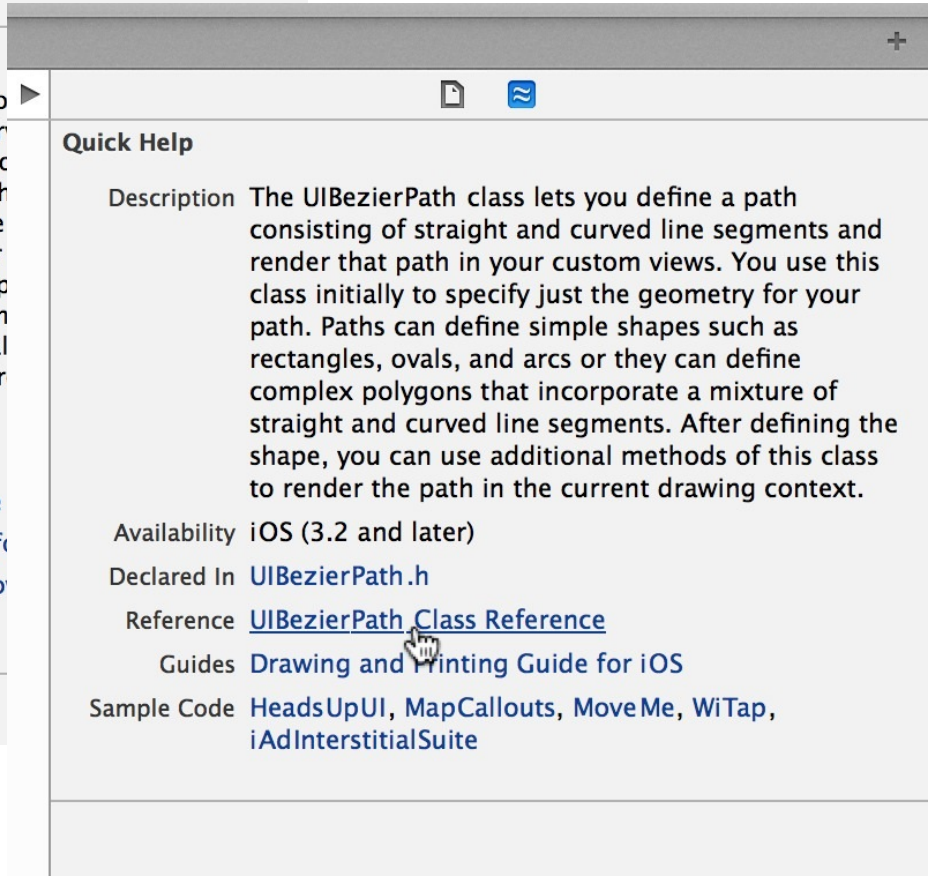
Have our view fill the screen


```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOption
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] boun
//    CGRect firstFrame = CGRectMake( 160, 240, 100, 150);
    CGRect firstFrame = self.window.bounds;
    INIHypnosisView *firstView = [[INIHypnosisView alloc] initWithFrame: firs
    firstView.backgroundColor = [ UIColor redColor];
    [self.window addSubview: firstView];

//    CGRect secondFrame = CGRectMake( 20, 30, 50, 50);
//    INIHypnosisView *secondView = [[INIHypnosisView alloc] initWithFrame: s
//    secondView. backgroundColor = [UIColor blueColor];
//
//    [self.window addSubview:secondView];
//    [firstView addSubview:secondView];

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
```

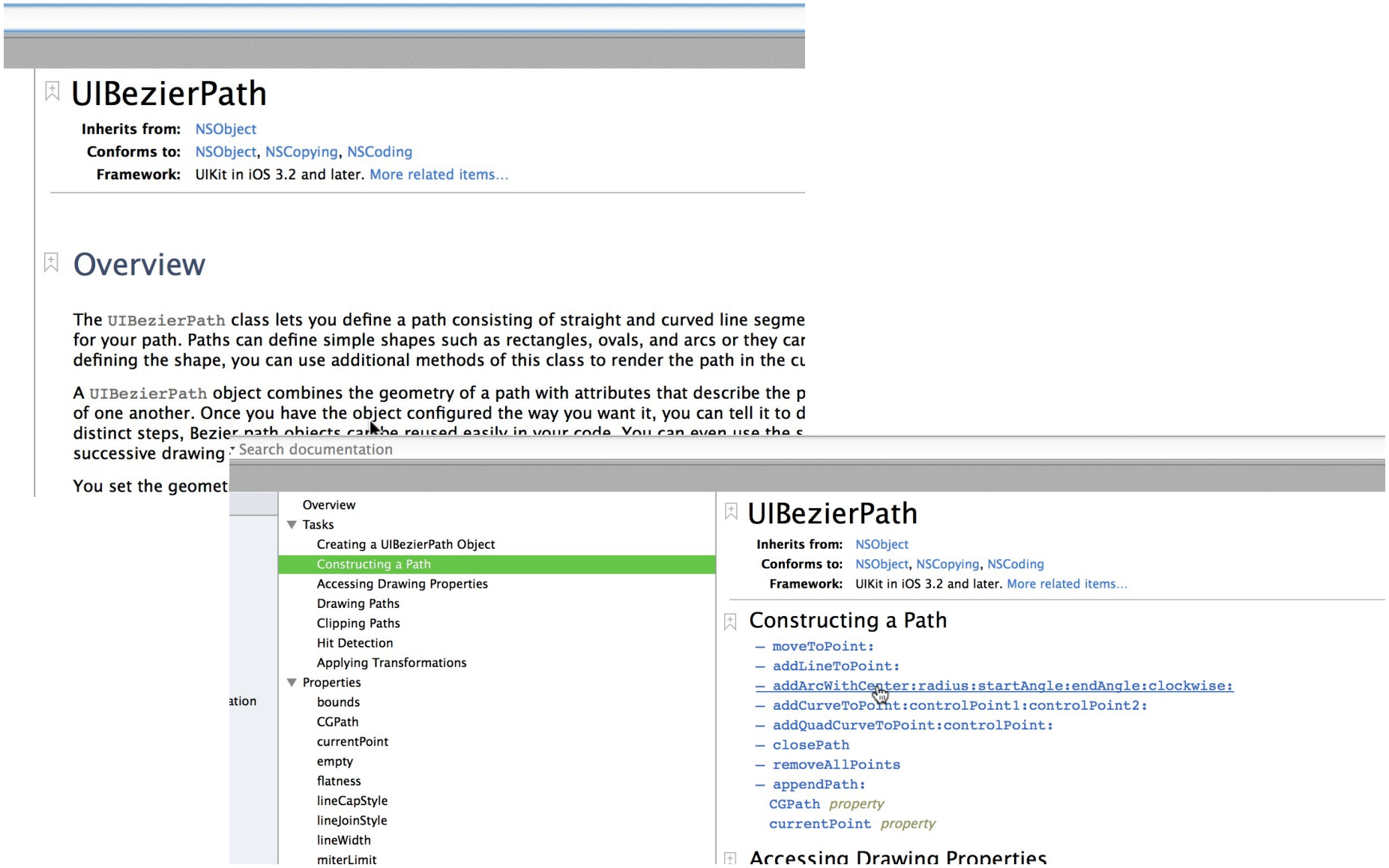
Getting Help

```
23  // Only override drawRect: if you perform custom drawing.
24  // An empty implementation adversely affects performance during animation.
25
26  - (void)drawRect:(CGRect)rect
27  {
28      CGRect bounds = self.bounds;
29      //Figureoutthecenteroftheboundsrectangle
30
31      CGPoint center;
32      center.x = bounds.origin.x + bounds.size.width / 2.0;
33      center.y = bounds.origin.y + bounds.size.height / 2.0;
34
35      // The circle will b
36      float radius = MIN(
37
38      UIBezierPath  path =
39  }
40
```





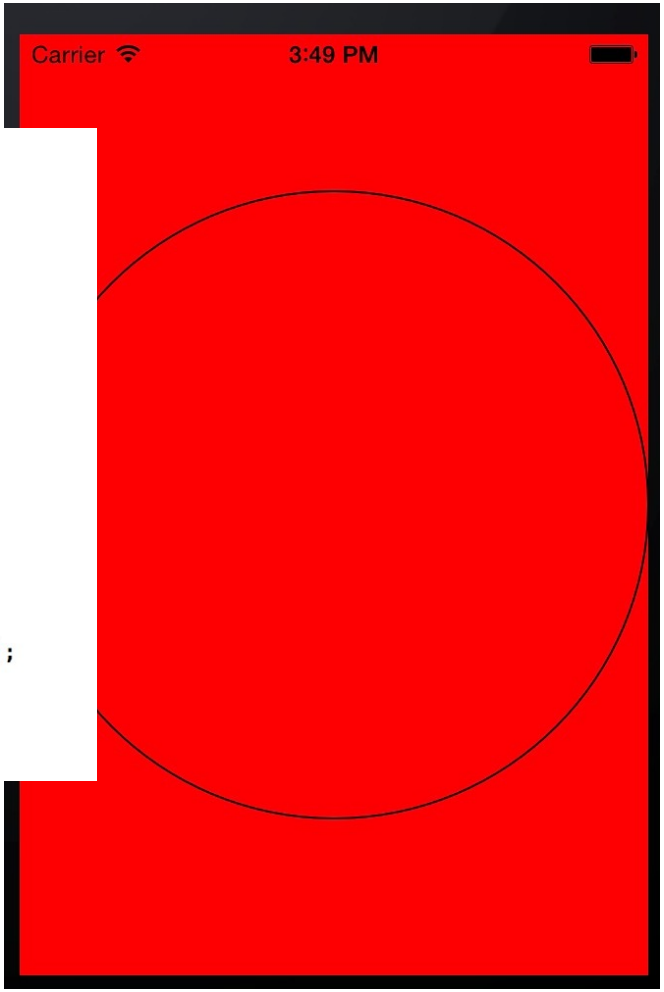
Classes and API Documentation



Overriding the drawRect: Method

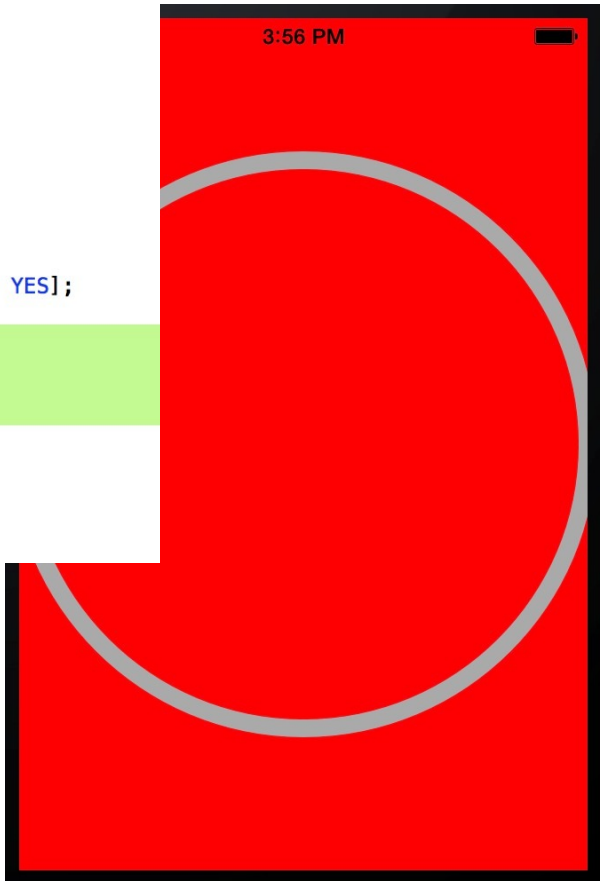
Using the drawRect: method to draw our circles

```
// Only override drawRect: if you perform custom drawing.
// An empty implementation adversely affects performance during animation.

- (void)drawRect:(CGRect)rect
{
    CGRect bounds = self.bounds;
    //Figureoutthecenteroftheboundsrectangle

    CGPoint center;
    center.x = bounds.origin.x + bounds.size.width / 2.0;
    center.y = bounds.origin.y + bounds.size.height / 2.0;

    // The circle will be the largest that will fit in the view
    float radius = MIN( bounds.size.width, bounds.size.height) / 2.0;

    UIBezierPath *path = [[ UIBezierPath alloc] init];

    // Add an arc to the path at center, with radius of radius,
    // from 0 to 2* PI radians ( a circle)

    [path addArcWithCenter: center radius: radius startAngle: 0.0 endAngle: M_PI * 2.0 clockwise: YES];

    // Draw the line!
    [path stroke];
}
```



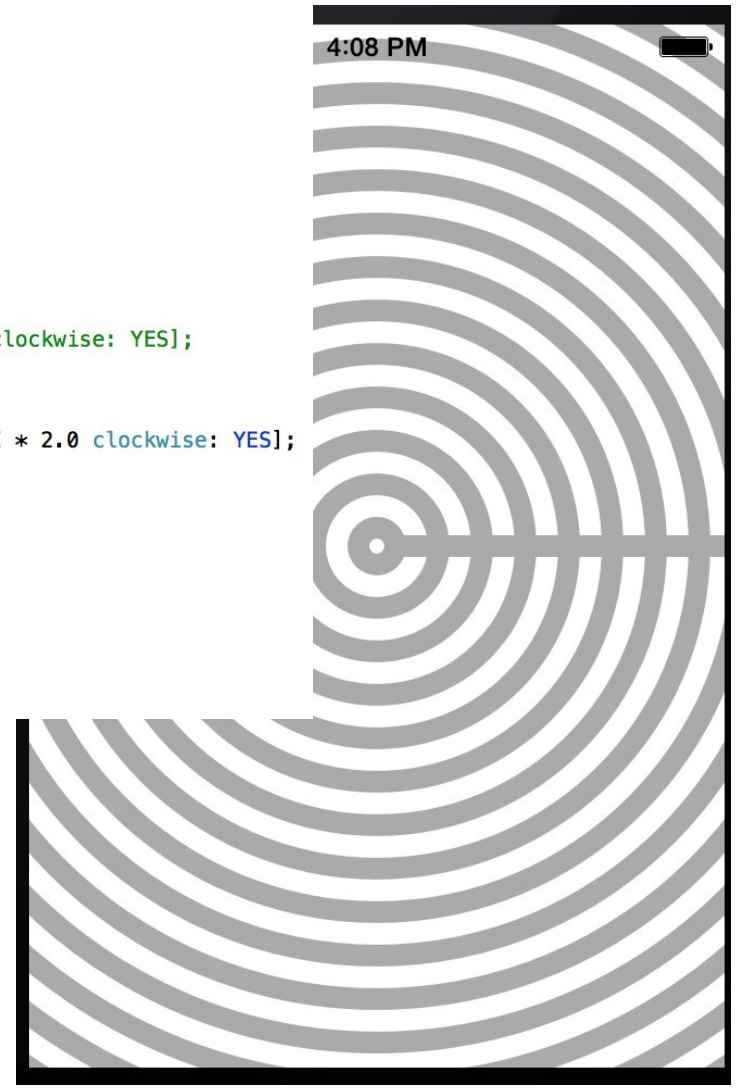
Set Stroke Color and Thickness

```
// Only override drawRect: if you perform custom drawing.
// An empty implementation adversely affects performance during animation.

- (void)drawRect:(CGRect)rect
{
    CGRect bounds = self.bounds;
    //Figureoutthecenteroftheboundsrectangle

    CGPoint center;
    center.x = bounds.origin.x + bounds.size.width / 2.0;
    center.y = bounds.origin.y + bounds.size.height / 2.0;

    // The circle will be the largest that will fit in the view
    float radius = MIN( bounds.size.width, bounds.size.height) / 2.0;

    UIBezierPath *path = [[ UIBezierPath alloc] init];

    // Add an arc to the path at center, with radius of radius,
    // from 0 to 2* PI radians ( a circle)

    [path addArcWithCenter: center radius: radius startAngle: 0.0 endAngle: M_PI * 2.0 clockwise: YES];

    // Configure line width to 10 points
    path.lineWidth = 10;

    // Configure the drawing color to light gray
    [[UIColor lightGrayColor] setStroke];

    // Draw the line!
    [path stroke];
}
```



Drawing Concentric Circles

```
- (void)drawRect:(CGRect)rect
{
    CGRect bounds = self.bounds;
    //Figureoutthecenteroftheboundsrectangle

    CGPoint center;
    center.x = bounds.origin.x + bounds.size.width / 2.0;
    center.y = bounds.origin.y + bounds.size.height / 2.0;

    // The circle will be the largest that will fit in the view
    // float radius = MIN( bounds.size.width, bounds.size.height) / 2.0;

    float maxRadius = hypot( bounds. size. width, bounds. size. height) / 2.0;

    UIBezierPath *path = [[ UIBezierPath alloc] init];

    // Add an arc to the path at center, with radius of radius,
    // from 0 to 2* PI radians ( a circle)
    // [path addArcWithCenter: center radius: radius startAngle: 0.0 endAngle: M_PI * 2.0 clockwise: YES];

    for ( float currentRadius = maxRadius; currentRadius > 0; currentRadius -= 20)
    {
        [path addArcWithCenter: center radius: currentRadius startAngle: 0.0 endAngle: M_PI * 2.0 clockwise: YES];
    }

    // Configure line width to 10 points
    path.lineWidth = 10;
    // Configure the drawing color to light gray
    [[UIColor lightGrayColor] setStroke];

    // Draw the line!
    [path stroke];
}
```



Cleaning Up

```
- (void)drawRect:(CGRect)rect
{
    CGRect bounds = self.bounds;
    //Figureoutthecenteroftheboundsrectangle

    CGPoint center;
    center.x = bounds.origin.x + bounds.size.width / 2.0;
    center.y = bounds.origin.y + bounds.size.height / 2.0;

    // The circle will be the largest that will fit in the view
    // float radius = MIN( bounds.size.width, bounds.size.height) / 2.0;

    float maxRadius = hypot( bounds. size. width, bounds. size. height) / 2.0;

    UIBezierPath *path = [[ UIBezierPath alloc] init];

    // Add an arc to the path at center, with radius of radius,
    // from 0 to 2* PI radians ( a circle)
    // [path addArcWithCenter: center radius: radius startAngle: 0.0 endAngle: M_PI * 2.0 clockwise: YES];

    for ( float currentRadius = maxRadius; currentRadius > 0; currentRadius -= 20)
    {
        [path moveToPoint: CGPointMake( center.x + currentRadius, center.y)];
        [path addArcWithCenter: center radius: currentRadius startAngle: 0.0 endAngle: M_PI * 2.0 clockwise: YES];
    }

    // Configure line width to 10 points
    path.lineWidth = 10;
    // Configure the drawing color to light gray
    [[UIColor lightGrayColor] setStroke];

    // Draw the line!
    [path stroke];
}
```