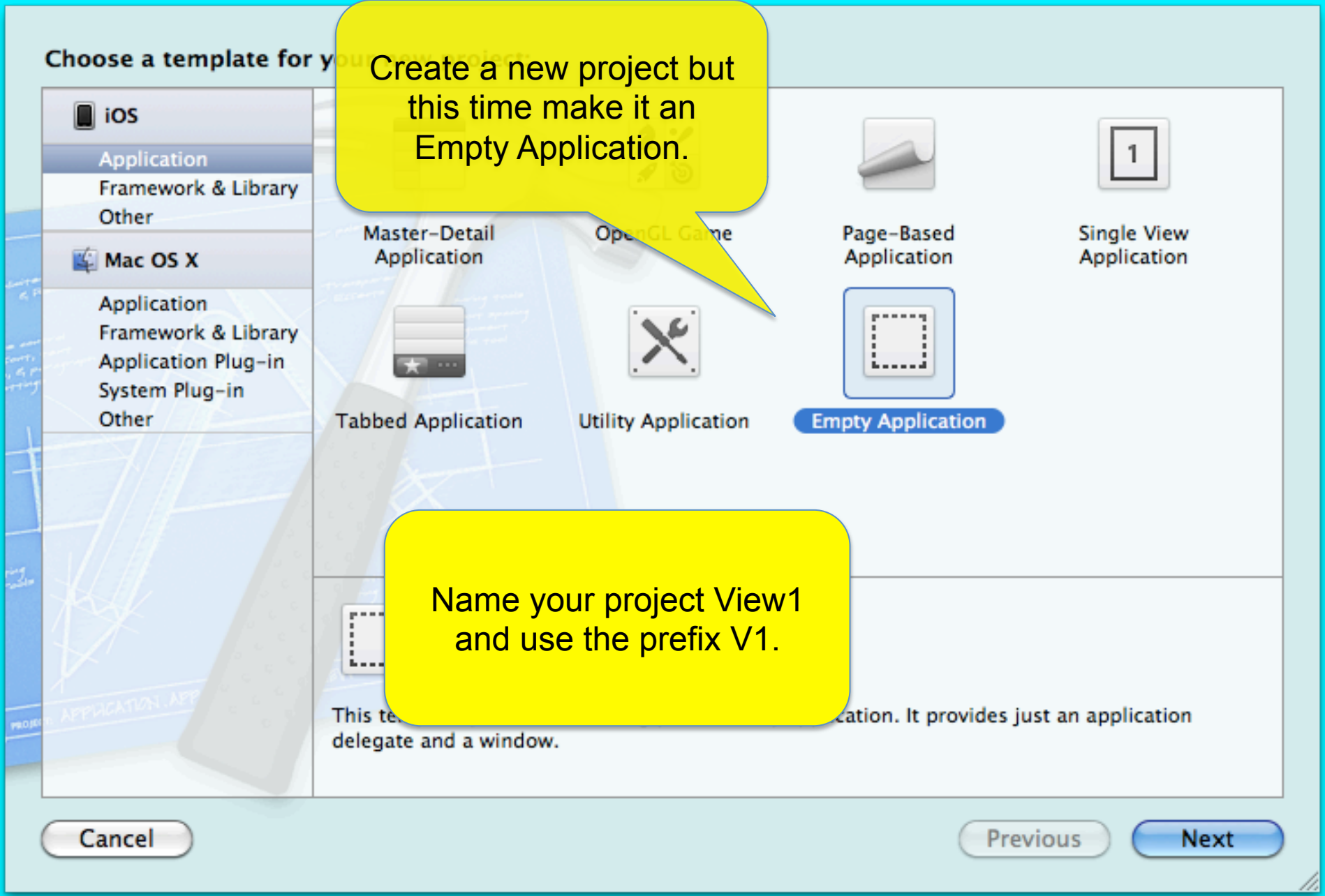
CS121 Tutorial 4

Intro to views!

Purple bubbles give you information you'll need to know.

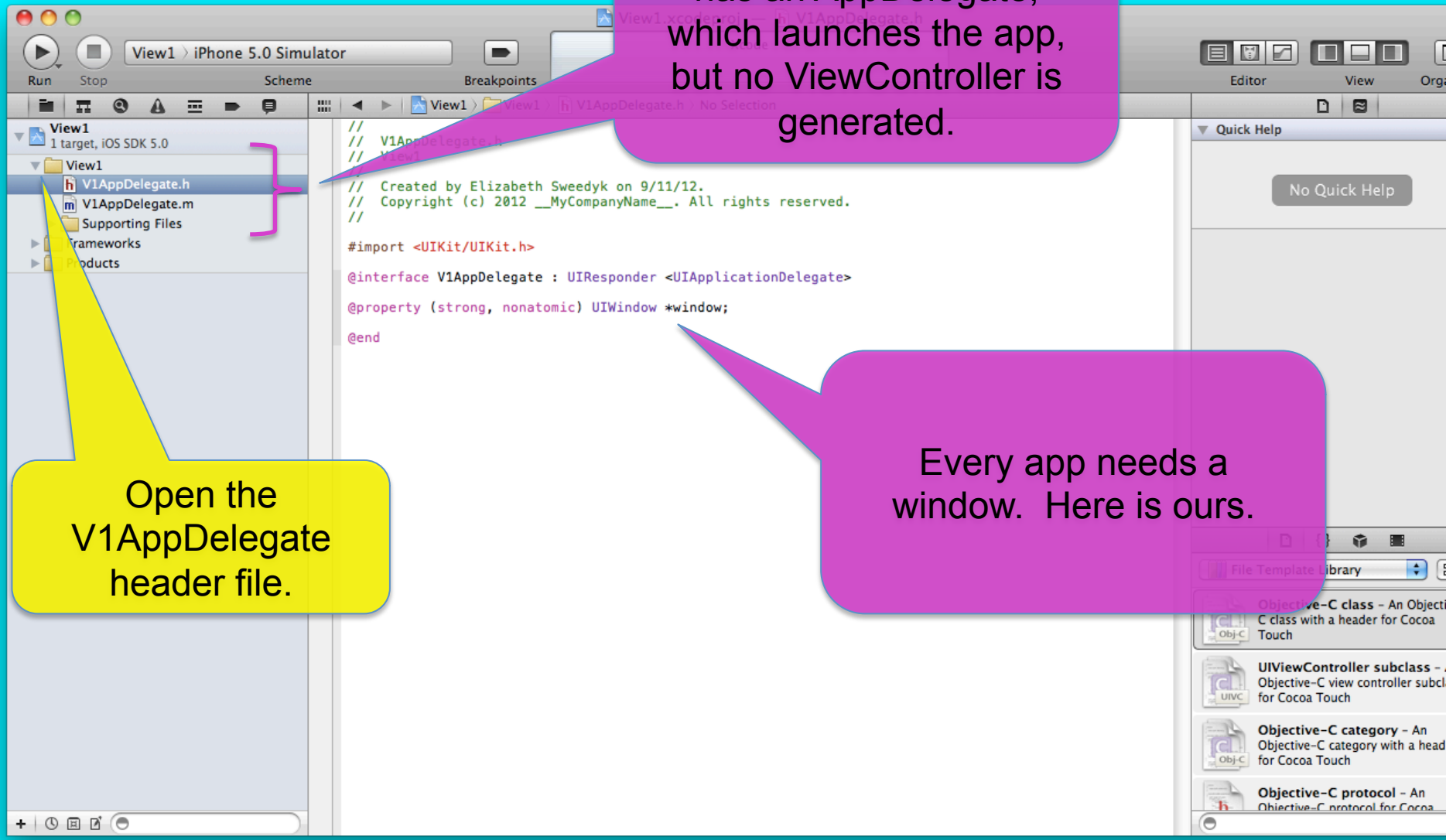
Yellow Bubbles tell you what to do.

Orange bubbles tell you what you're not expected to understand yet. ☺

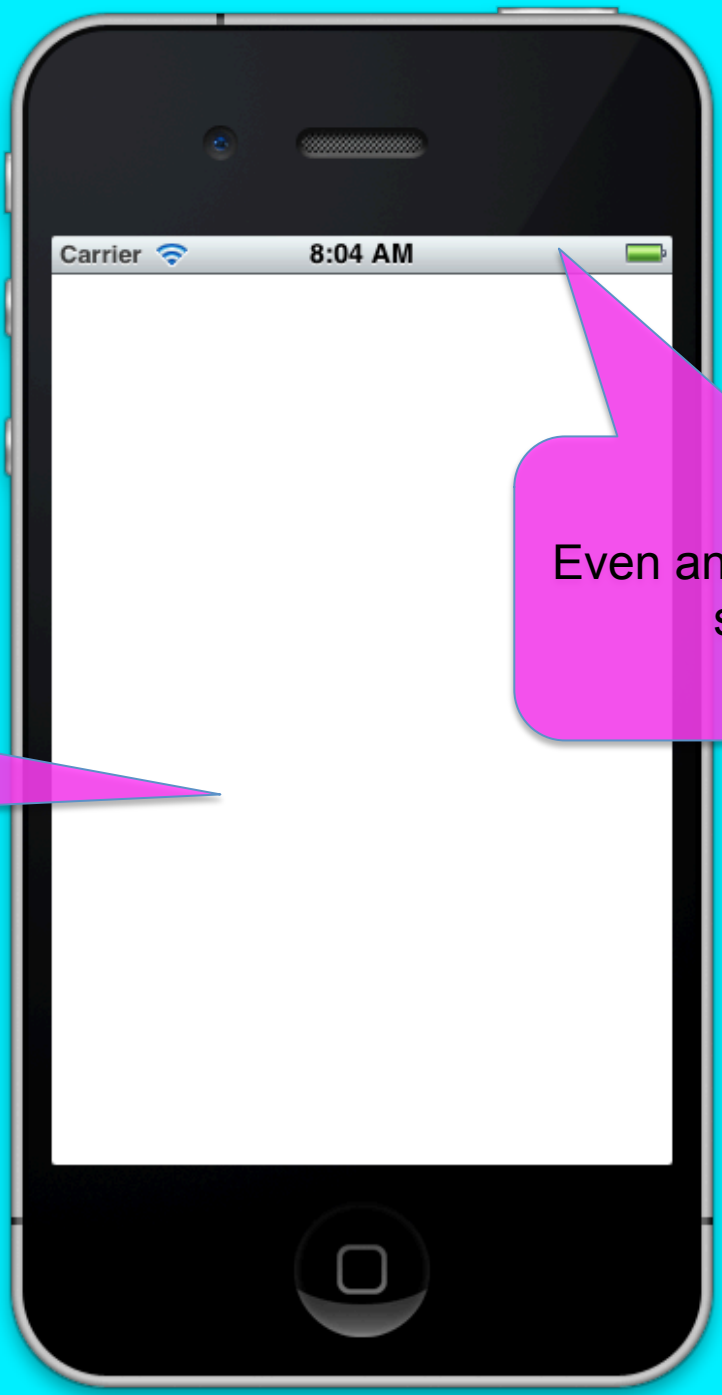


Create a new project but this time make it an Empty Application.

Name your project View1 and use the prefix V1.

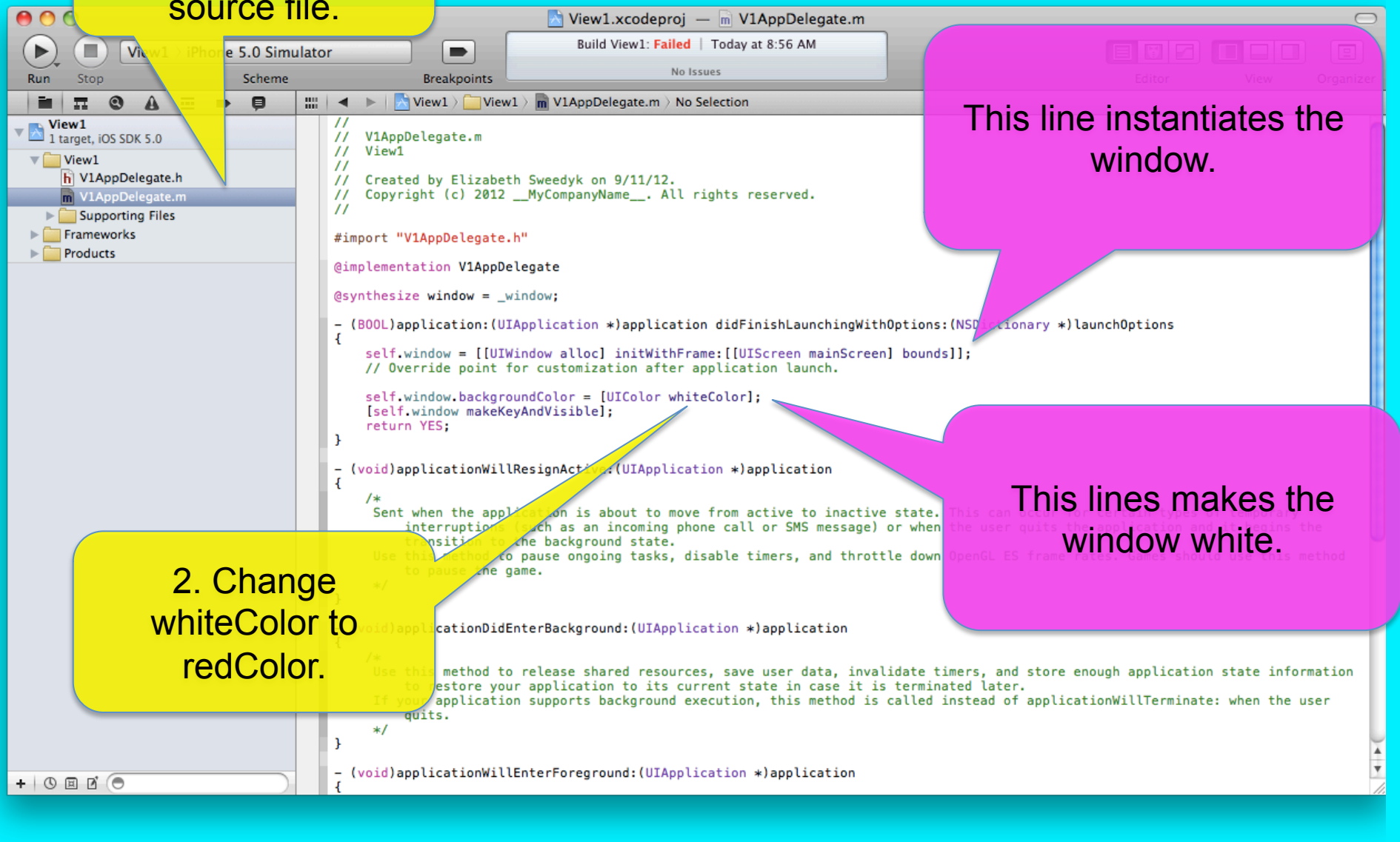

An empty application still has an AppDelegate, which launches the app, but no ViewController is generated.
Open the V1AppDelegate header file.
Every app needs a window. Here is ours.
View1 › iPhone 5.0 Simulator
Run
Stop
Scheme
Breakpoints
Editor
View
View1
1 target, iOS SDK 5.0
View1
V1AppDelegate.h
V1AppDelegate.m
Supporting Files
Quick Help
No Quick Help
// Created by Elizabeth Sweedyk on 9/11/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
#import <UIKit/UIKit.h>
@interface V1AppDelegate : UIResponder <UIApplicationDelegate>
@property (strong, nonatomic) UIWindow *window;
@end
UIViewController subclass
Objective-C category
Objective-C protocol

Run the app and check out the simulator.



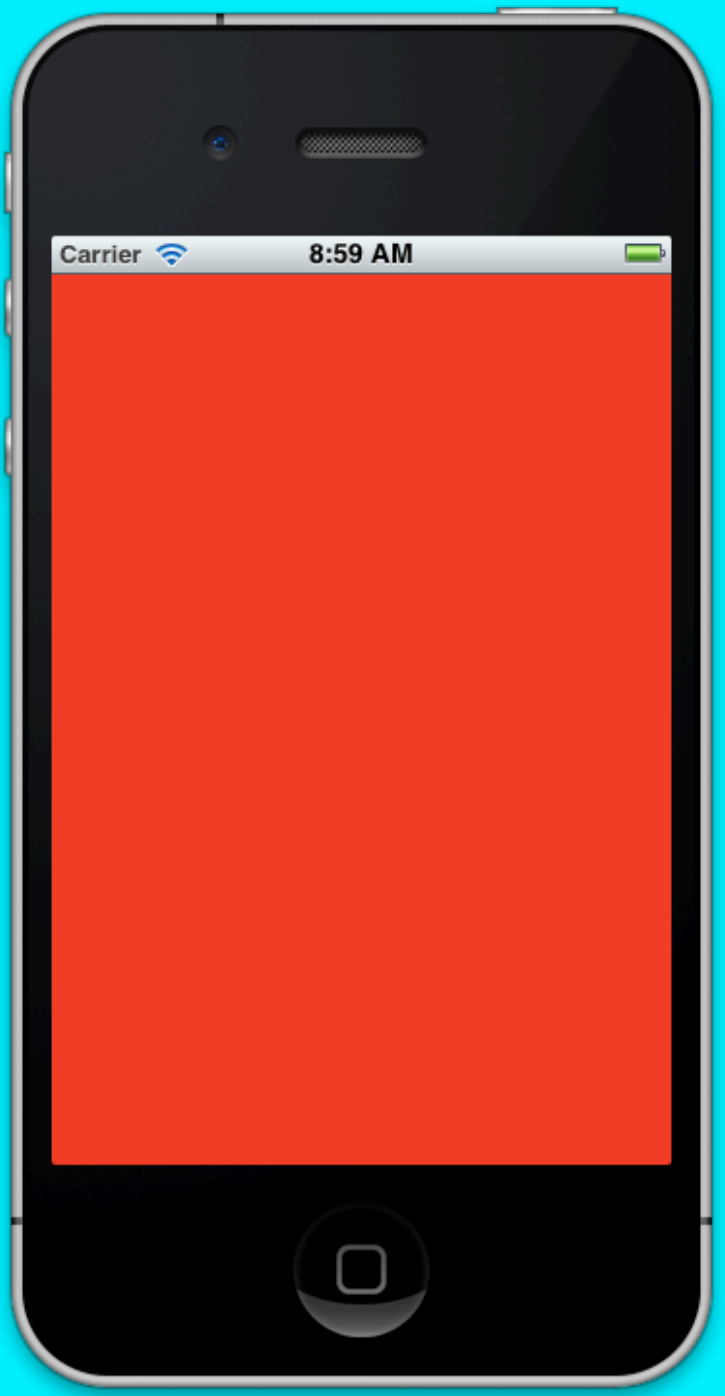
The white region is our window.

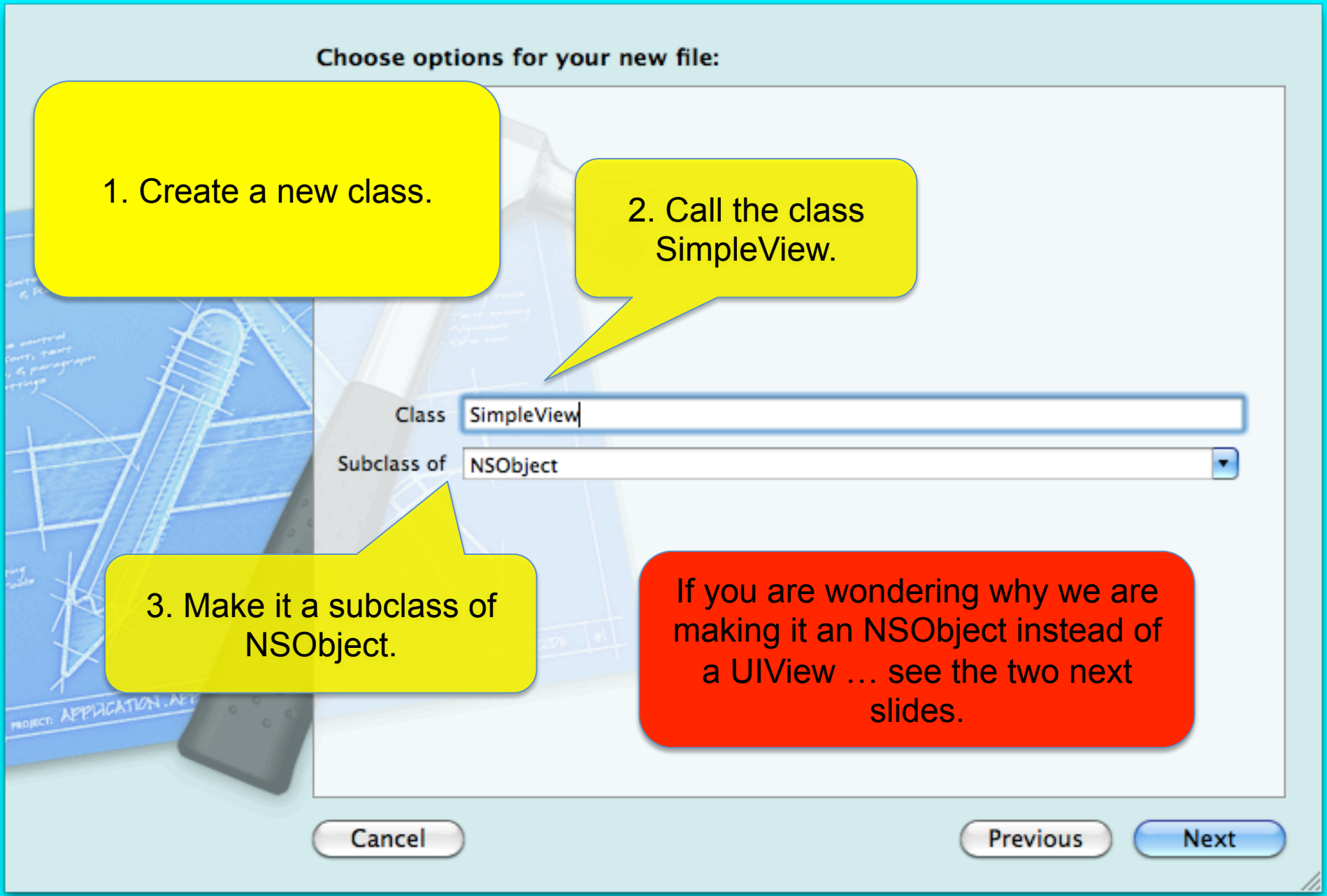
Even an empty app has a status bar.


1. Open the V1AppDelegate source file.
View1.xcodeproj — V1AppDelegate.m
Run
Stop
Scheme
Breakpoints
Build View1: Failed | Today at 8:56 AM
No Issues
View1
1 target, iOS SDK 5.0
View1
V1AppDelegate.h
V1AppDelegate.m
Supporting Files
Frameworks
Products
View1 > View1 > V1AppDelegate.m > No Selection
//
// V1AppDelegate.m
// View1
//
// Created by Elizabeth Sweedyk on 9/11/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import "V1AppDelegate.h"
@implementation V1AppDelegate
@synthesize window = _window;
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
// Override point for customization after application launch.
self.window.backgroundColor = [UIColor whiteColor];
[self.window makeKeyAndVisible];
return YES;
}
This line instantiates the window.
This lines makes the window white.
2. Change whiteColor to redColor.

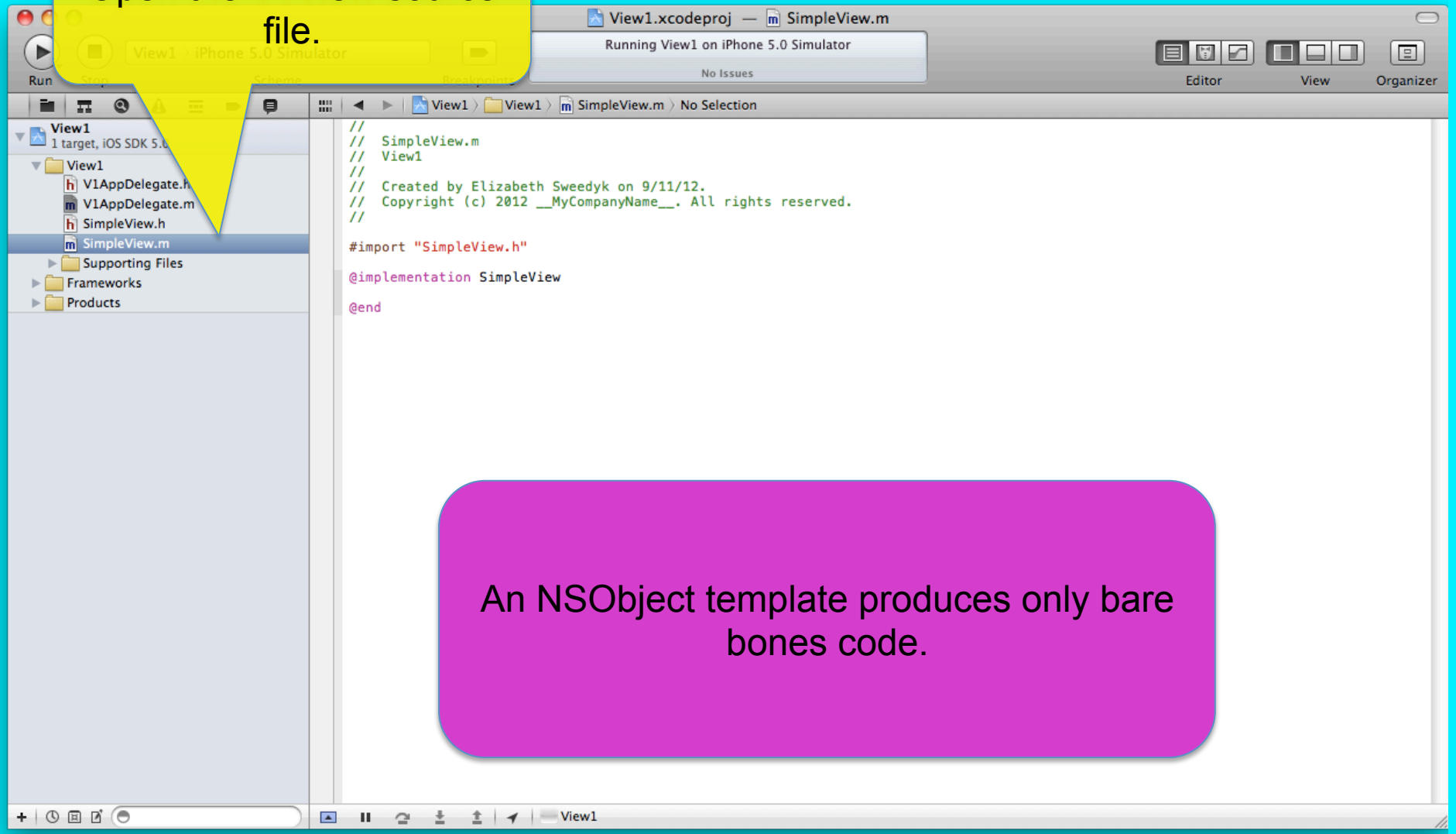
Run the app and check out the simulator.

Now our window is red.



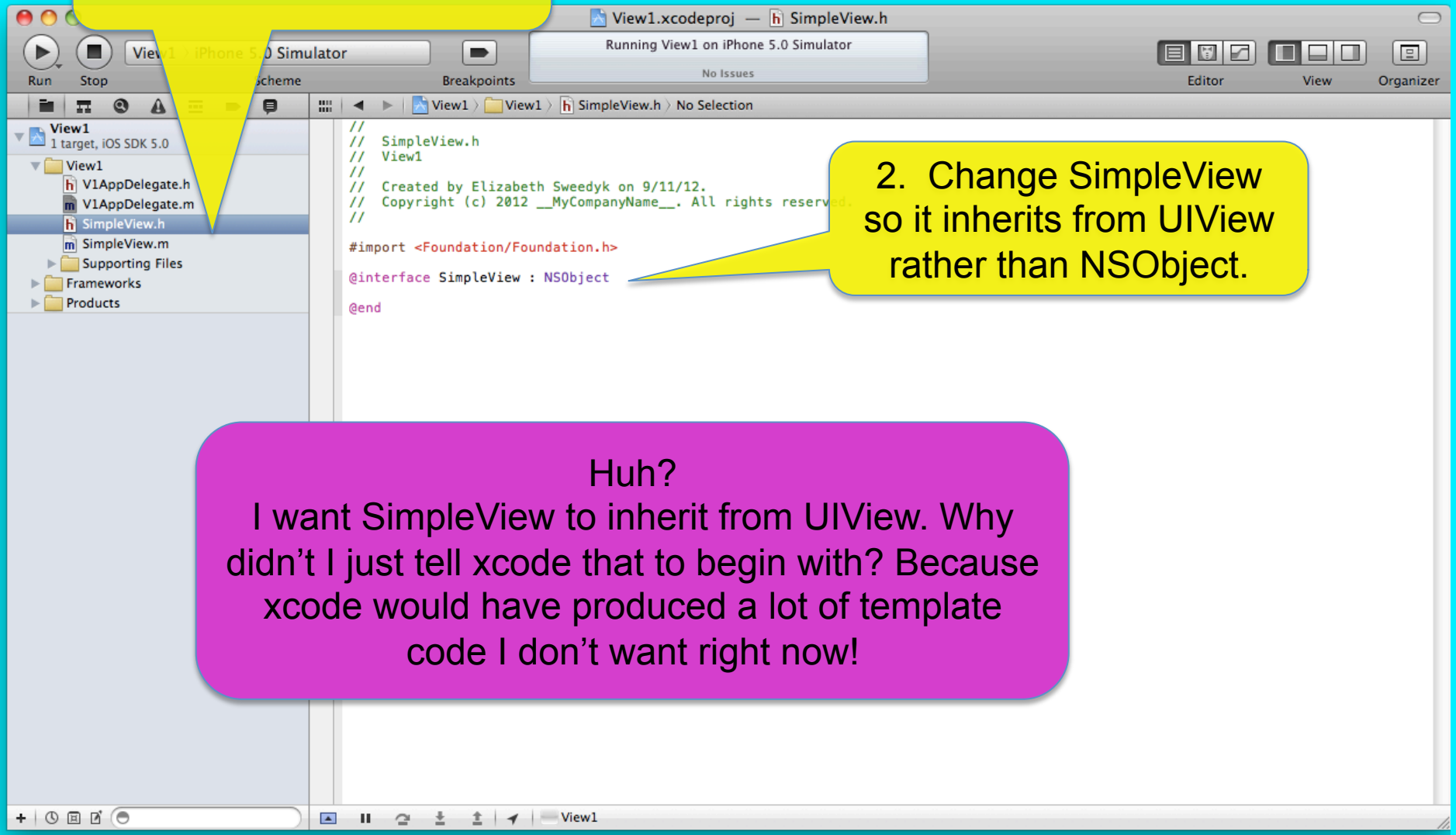

Choose options for your new file:
1. Create a new class.
2. Call the class SimpleView.
Class
SimpleView
Subclass of
NSObject
3. Make it a subclass of NSObject.
If you are wondering why we are making it an NSObject instead of a UIView … see the two next slides.
Cancel
Previous
Next

Open the V1View source file.



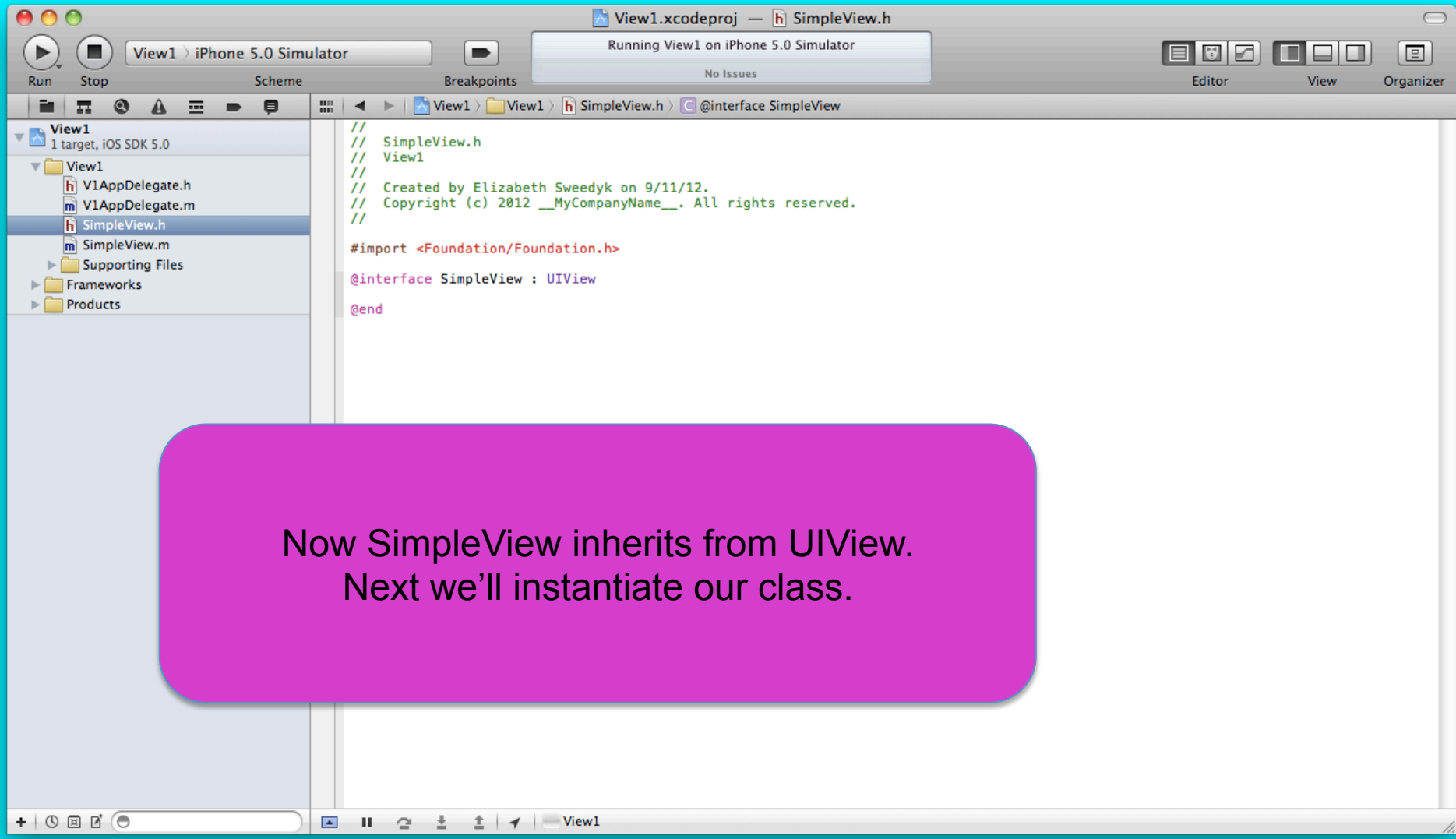
An NSObject template produces only bare bones code.

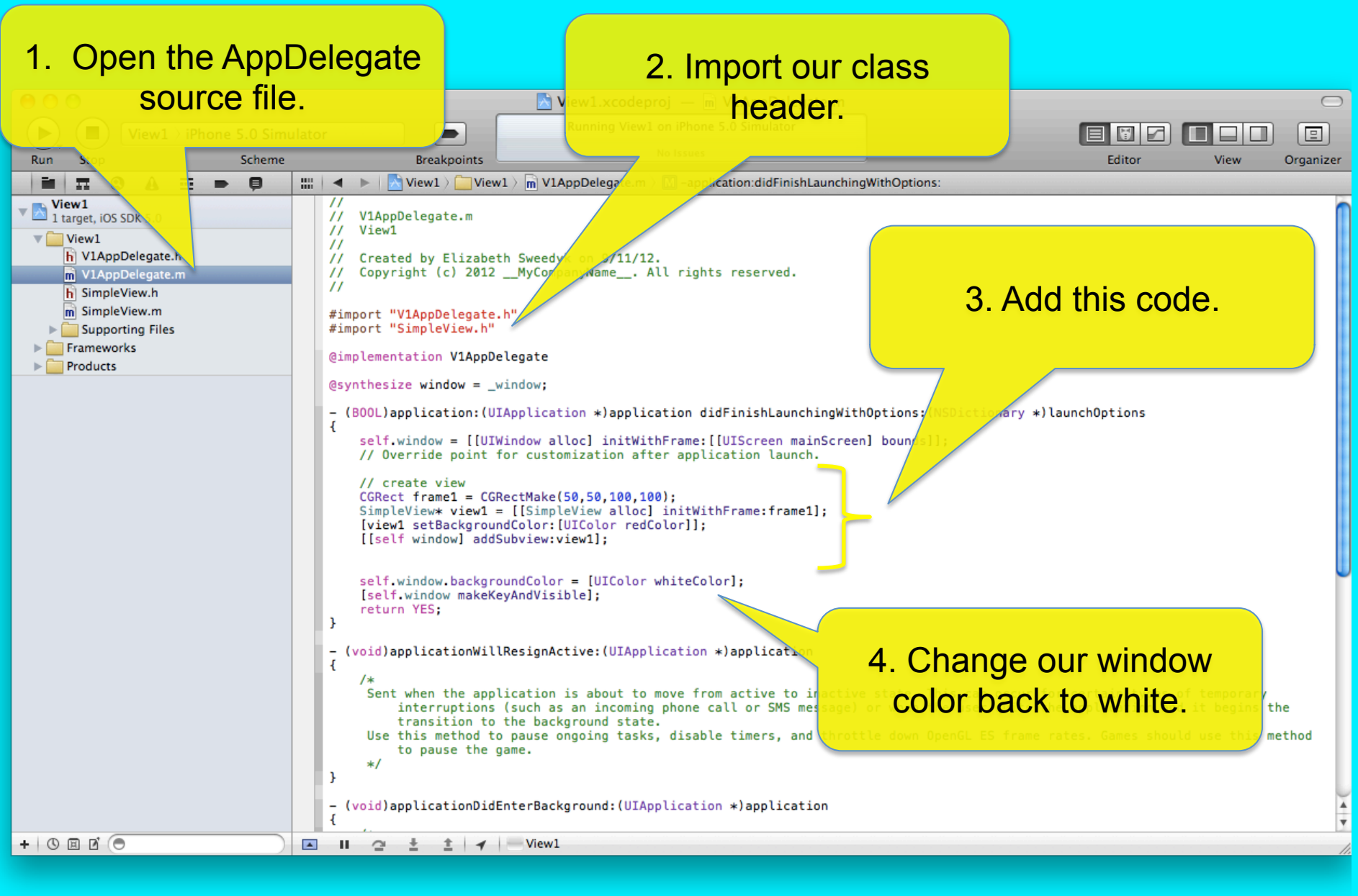
1. Open theV1View header file.



2. Change SimpleView so it inherits from UIView rather than NSObject.

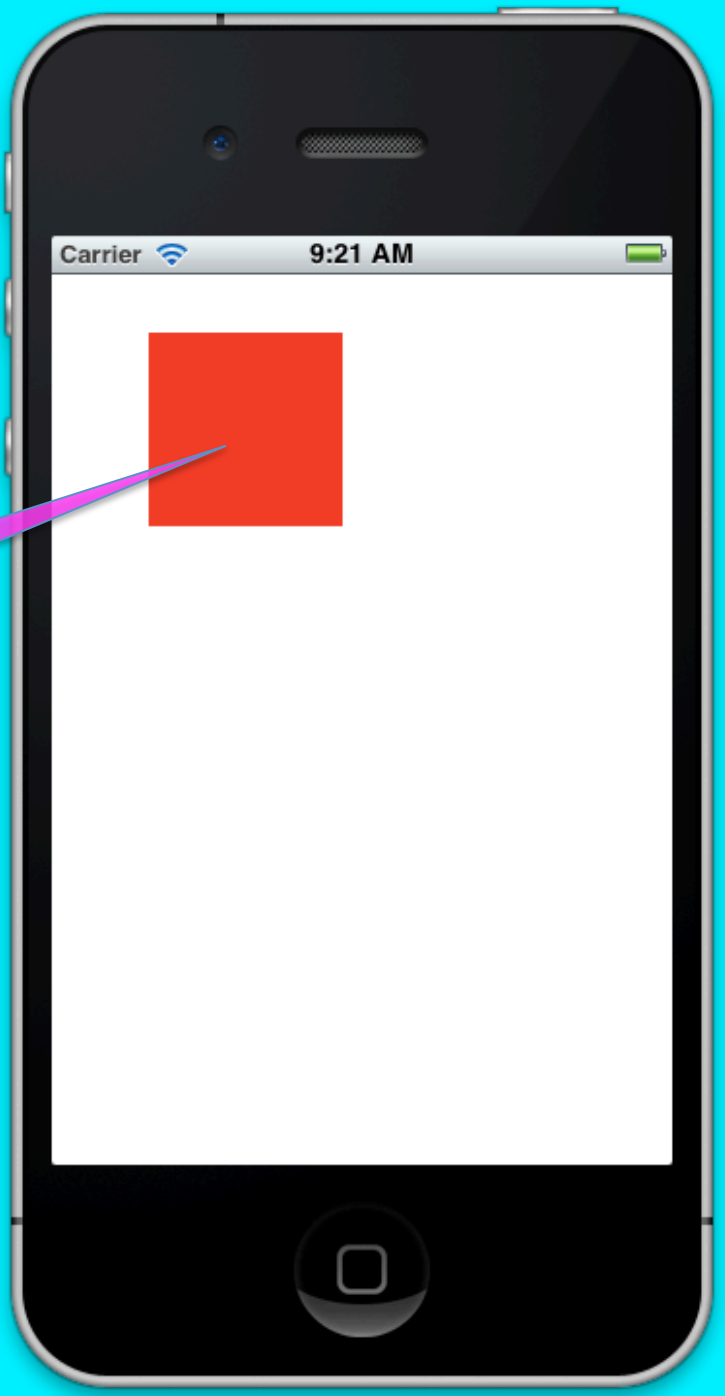
Huh?
I want SimpleView to inherit from UIView. Why didn't I just tell xcode that to begin with? Because xcode would have produced a lot of template code I don't want right now!


View1.xcodeproj — SimpleView.h
Run
Stop
View1 › iPhone 5.0 Simulator
Scheme
Breakpoints
Running View1 on iPhone 5.0 Simulator
No Issues
Editor
View
Organizer
View1
1 target, iOS SDK 5.0
View1
V1AppDelegate.h
V1AppDelegate.m
SimpleView.h
SimpleView.m
Supporting Files
Frameworks
Products
View1 › View1 › SimpleView.h › @interface SimpleView
//
// SimpleView.h
// View1
//
// Created by Elizabeth Sweedyk on 9/11/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <Foundation/Foundation.h>
@interface SimpleView : UIView
@end
Now SimpleView inherits from UIView.
Next we'll instantiate our class.
View1


1. Open the AppDelegate source file.
2. Import our class header.
3. Add this code.
4. Change our window color back to white.
View1
1 target, iOS SDK
View1
V1AppDelegate.h
V1AppDelegate.m
SimpleView.h
SimpleView.m
Supporting Files
Frameworks
Products
//
// V1AppDelegate.m
// View1
//
// Created by Elizabeth Sweedy on 6/11/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import "V1AppDelegate.h"
#import "SimpleView.h"
@implementation V1AppDelegate
@synthesize window = _window;
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
// Override point for customization after application launch.
// create view
CGRect frame1 = CGRectMake(50,50,100,100);
SimpleView* view1 = [[SimpleView alloc] initWithFrame:frame1];
[view1 setBackgroundColor:[UIColor redColor]];
[[self window] addSubview:view1];
self.window.backgroundColor = [UIColor whiteColor];
[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
{

Run the app.

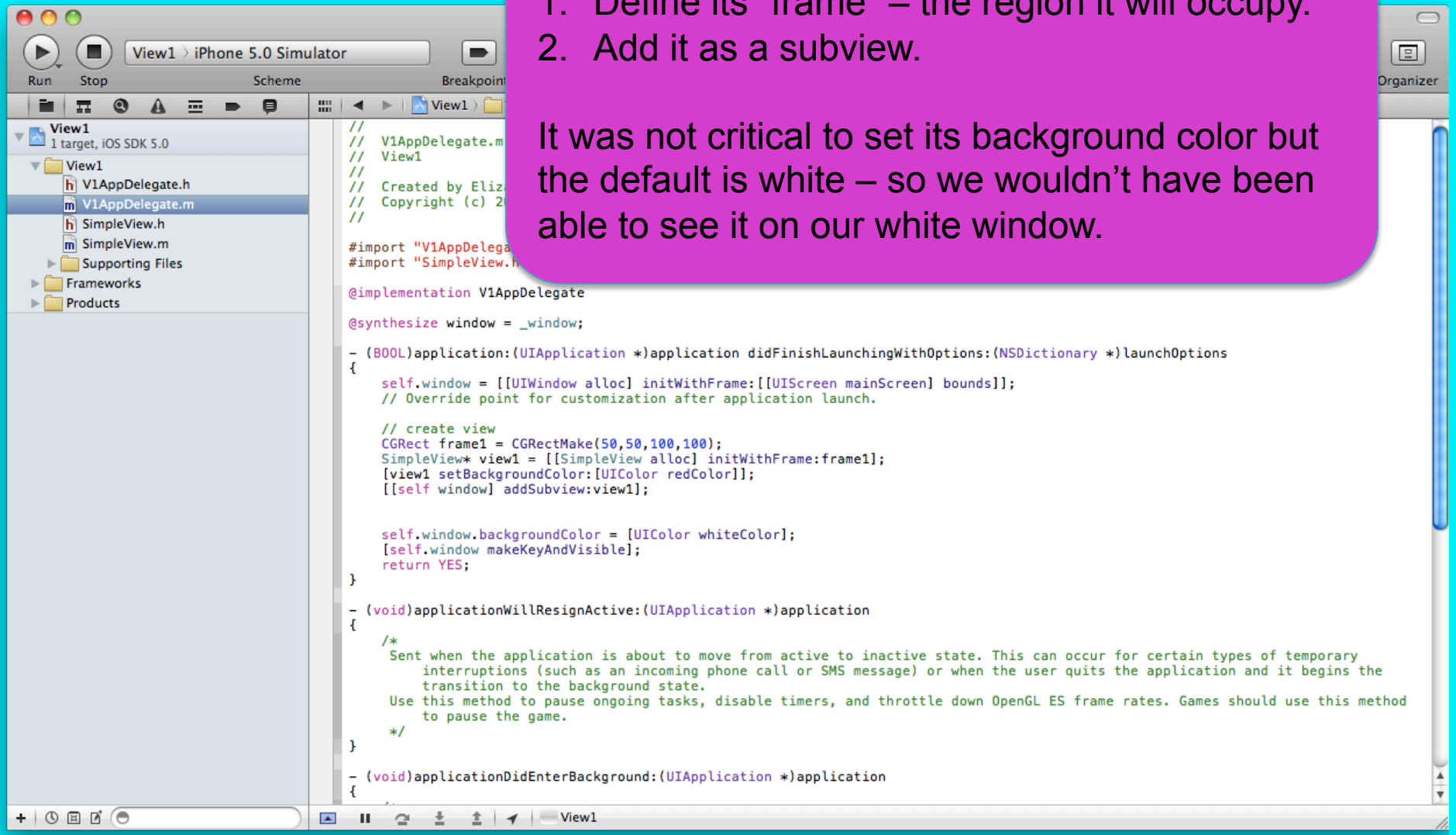
This is our new view.



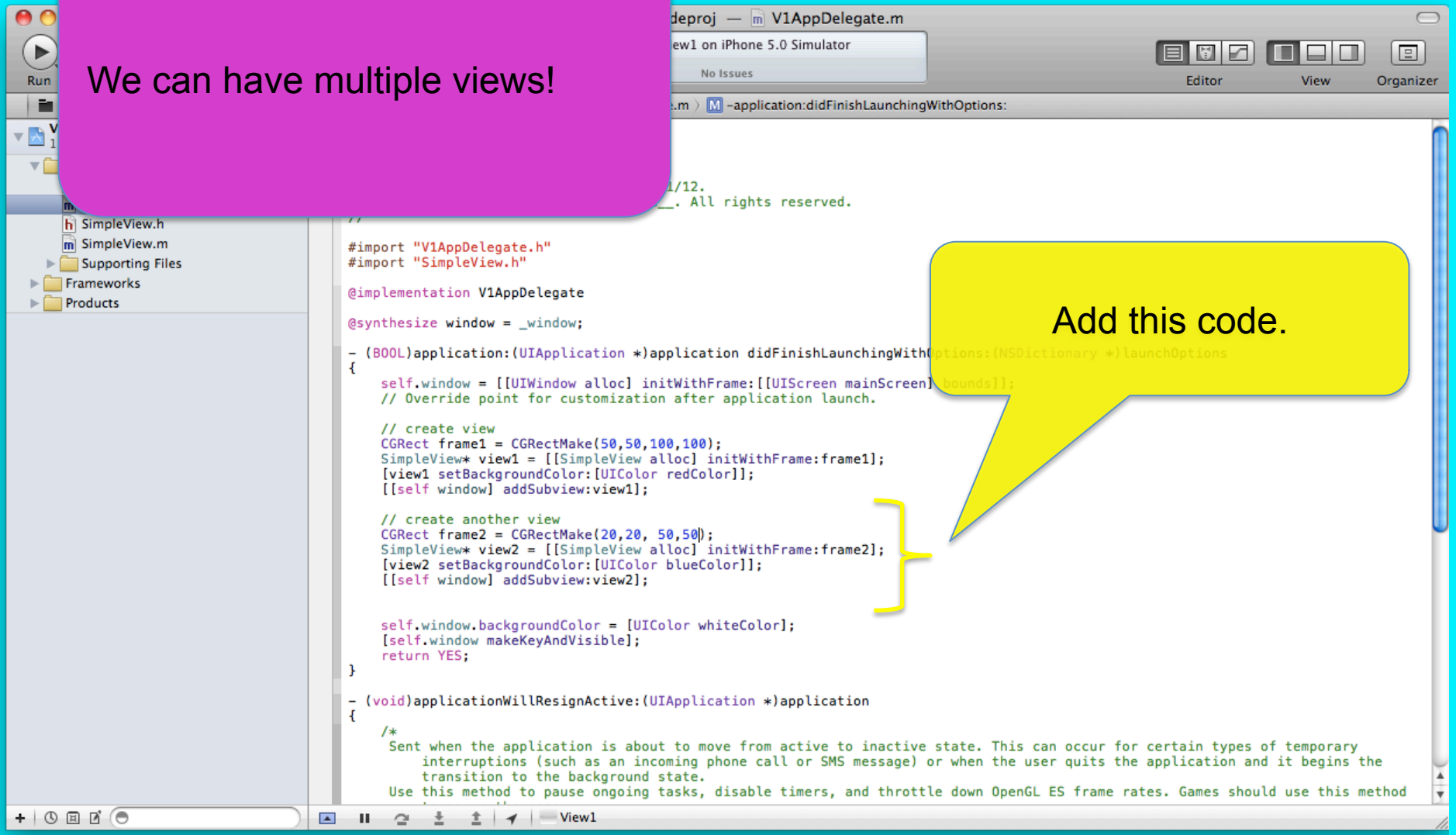
The key steps to creating a view are to:

1. Define its “frame” – the region it will occupy.
2. Add it as a subview.

It was not critical to set its background color but the default is white – so we wouldn’t have been able to see it on our white window.



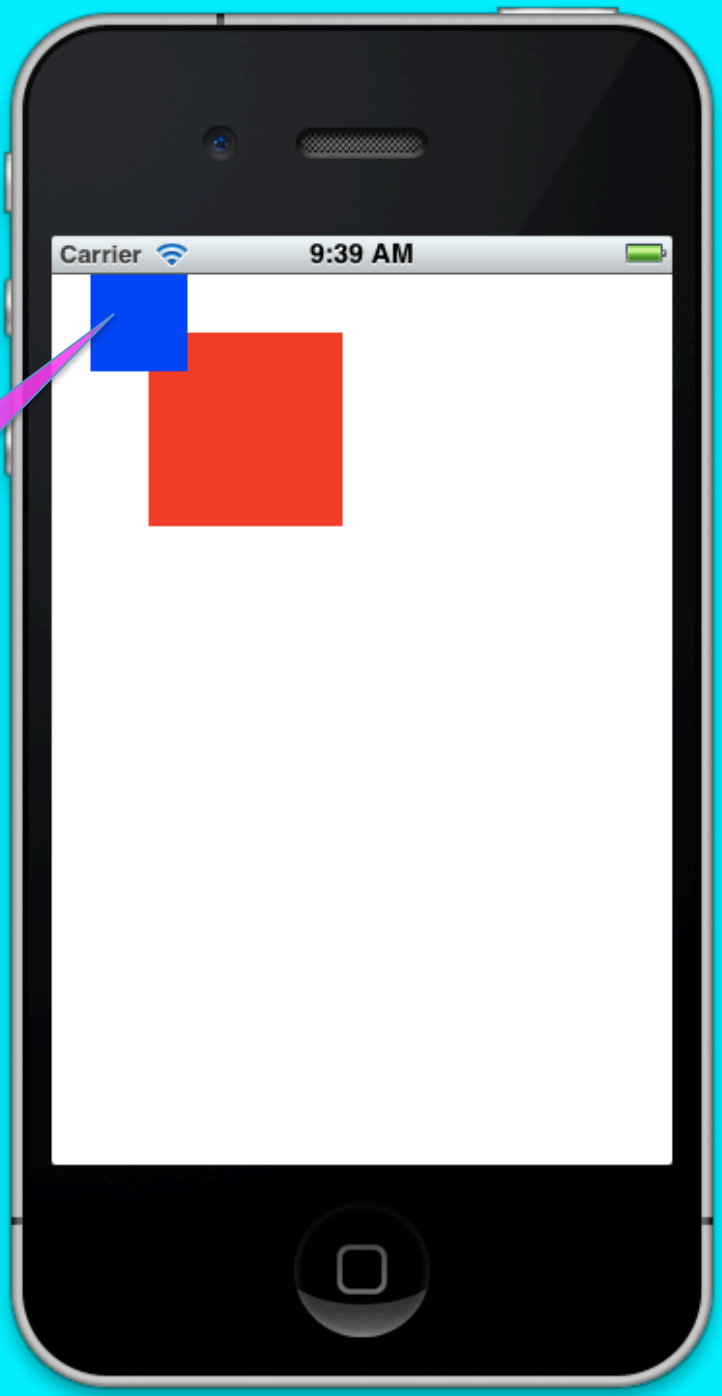
We can have multiple views!

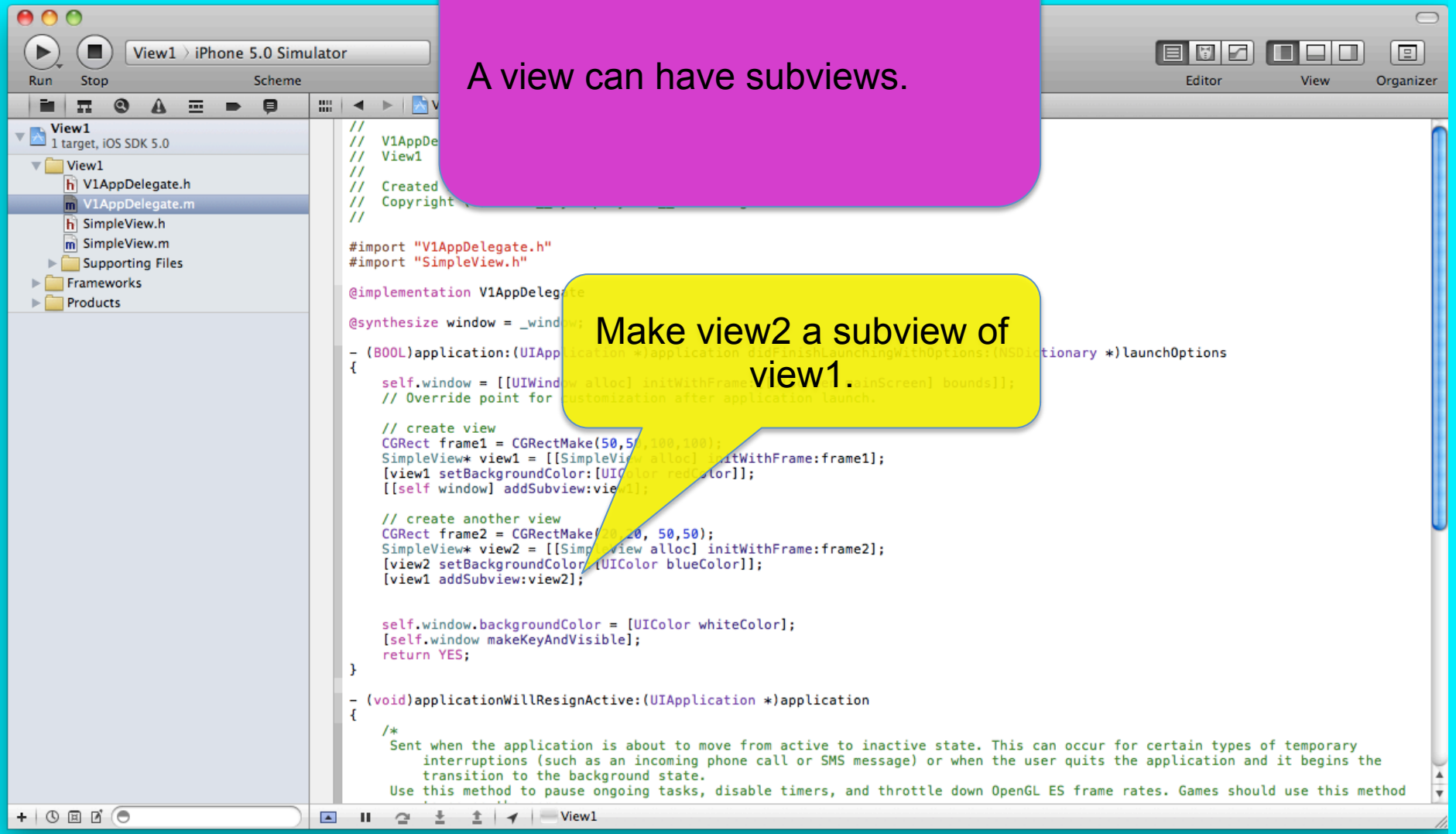


Add this code.

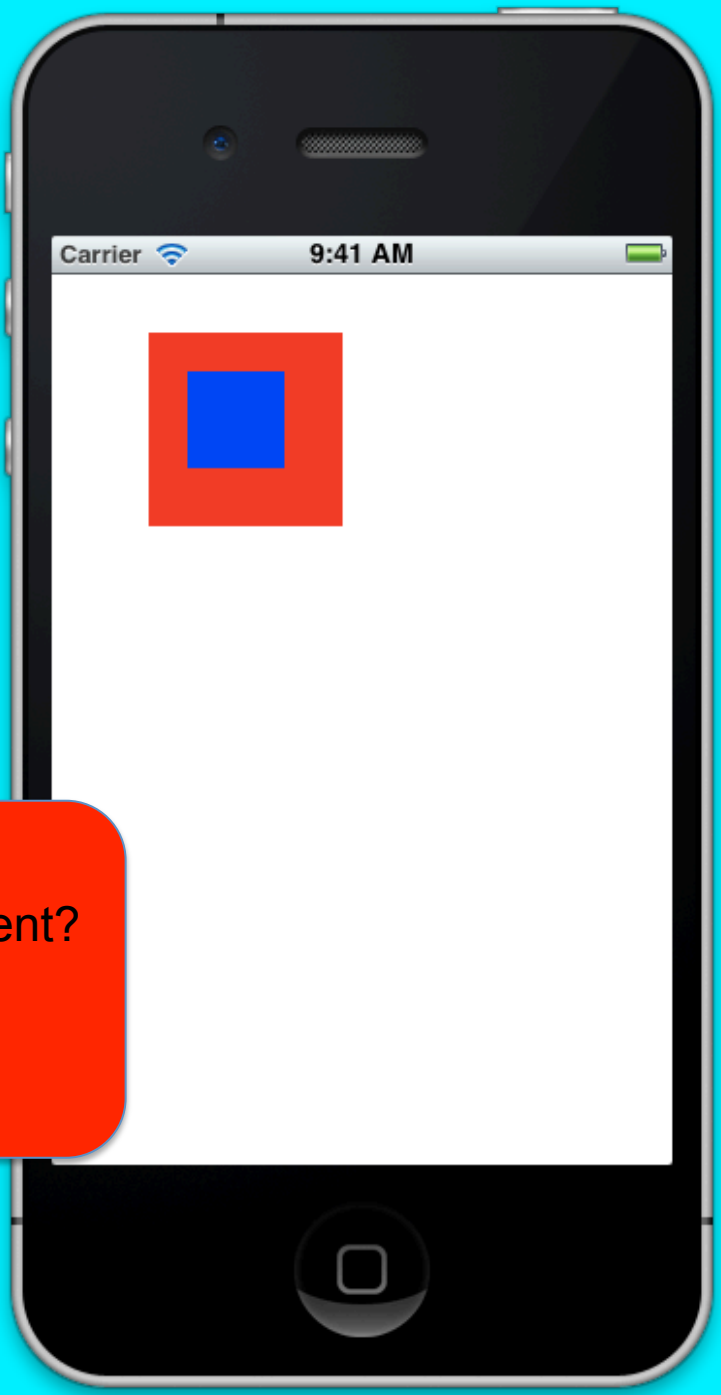
Run the app.

This is our new view. Views are drawn in the order they are added as subviews, so the blue view is drawn on top of the red one.
(There is a way to reorder the views.)



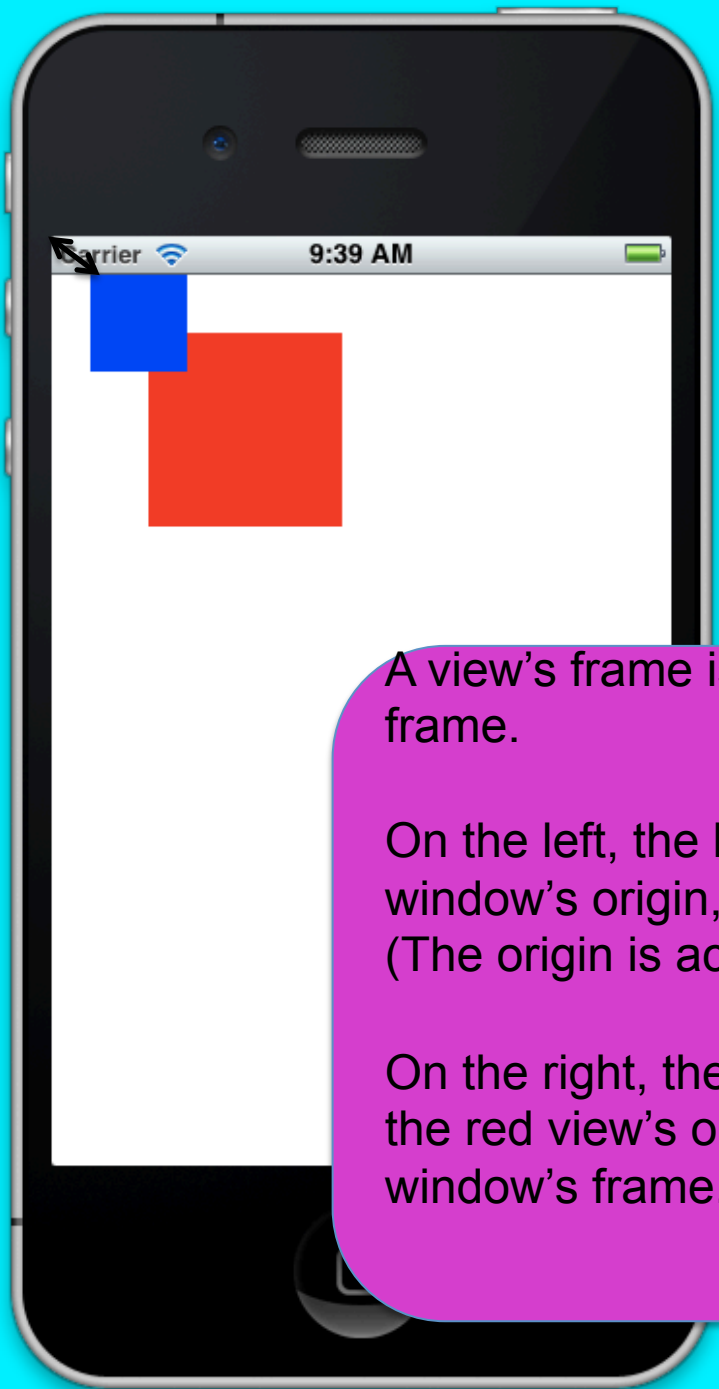

A view can have subviews.
Make view2 a subview of view1.
View1 › iPhone 5.0 Simulator
Run
Stop
Scheme
Editor
View
Organizer
View1
1 target, iOS SDK 5.0
View1
V1AppDelegate.h
V1AppDelegate.m
SimpleView.h
SimpleView.m
Supporting Files
Frameworks
Products
// create view
CGRect frame1 = CGRectMake(50,50,100,100);
SimpleView* view1 = [[SimpleView alloc] initWithFrame:frame1];
[view1 setBackgroundColor:[UIColor redColor]];
[[self window] addSubview:view1];
// create another view
CGRect frame2 = CGRectMake(20,20, 50,50);
SimpleView* view2 = [[SimpleView alloc] initWithFrame:frame2];
[view2 setBackgroundColor:[UIColor blueColor]];
[view1 addSubview:view2];
self.window.backgroundColor = [UIColor whiteColor];
[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
View1

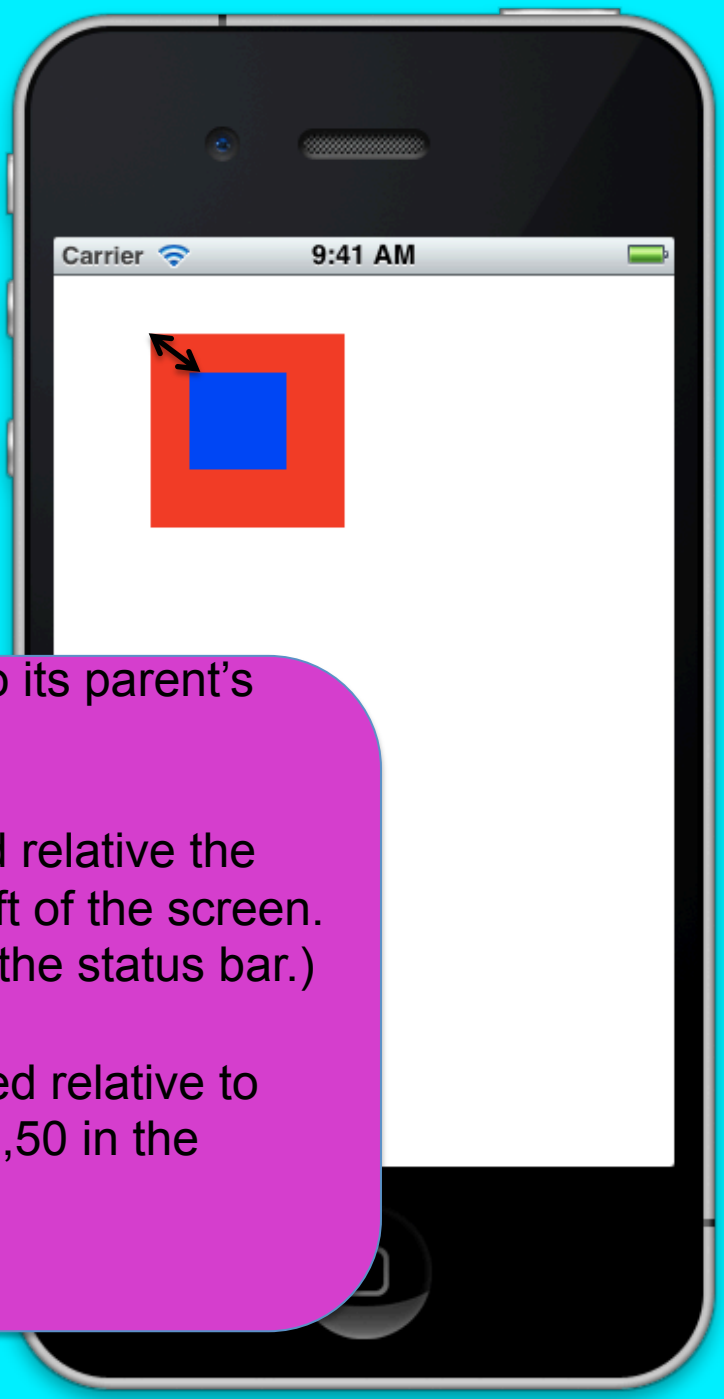
Run the app.



Wondering why it looks different?

Go to the next slide.



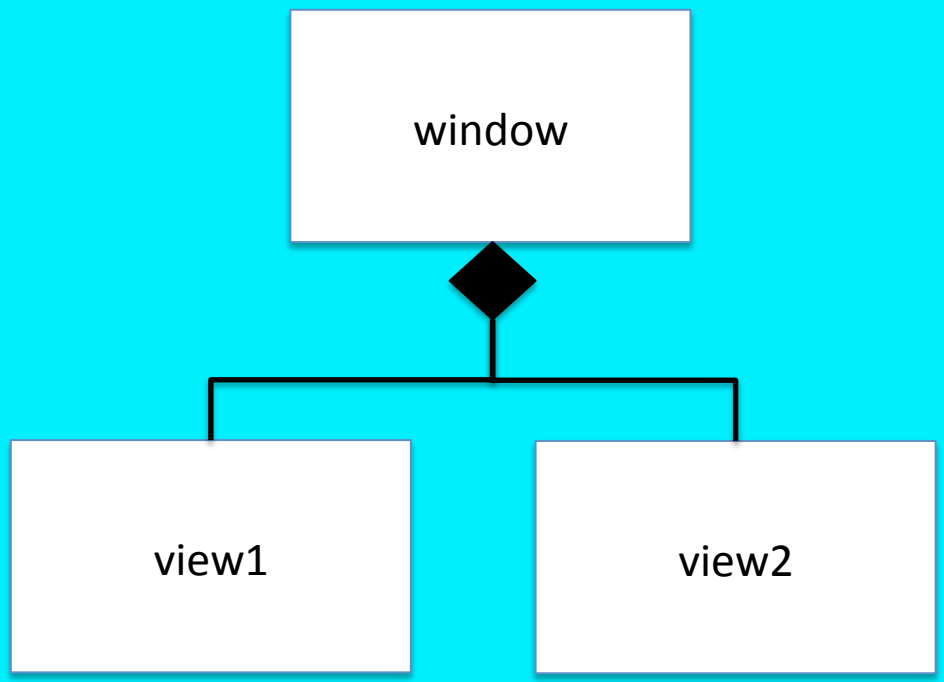


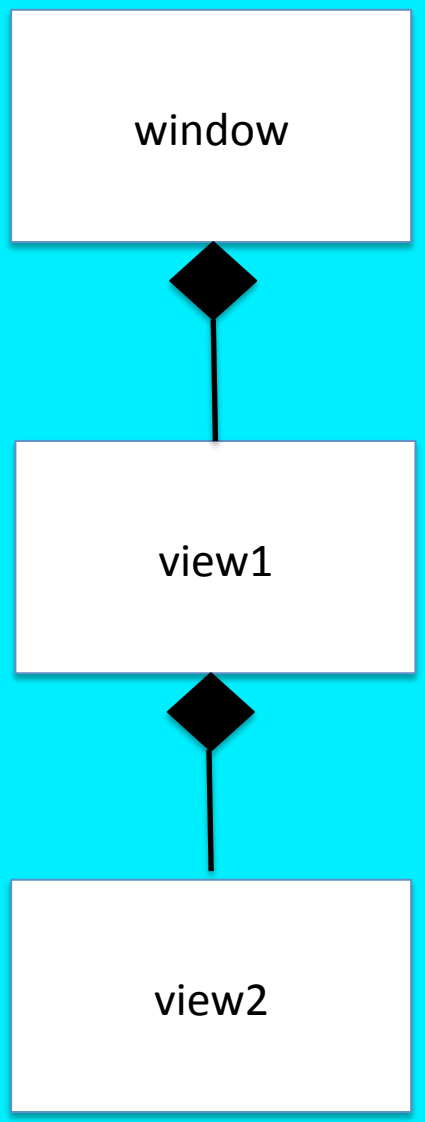
A view's frame is defined relative to its parent's frame.

On the left, the blue view is defined relative the window's origin, which is the top left of the screen. (The origin is actually occluded by the status bar.)

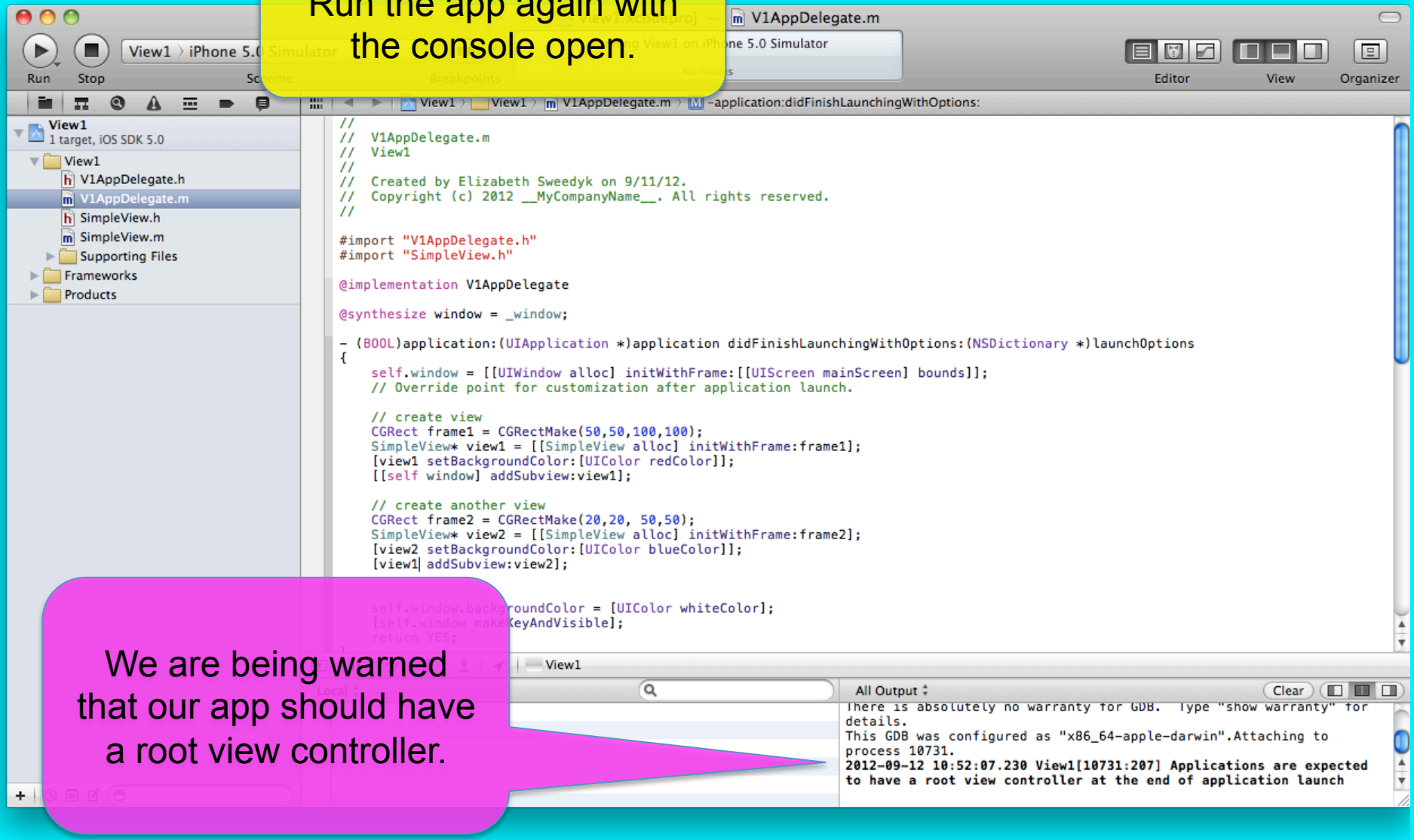
On the right, the blue view is defined relative to the red view's origin, which is at 50,50 in the window's frame.

Here are UML Object diagrams of the two approaches!





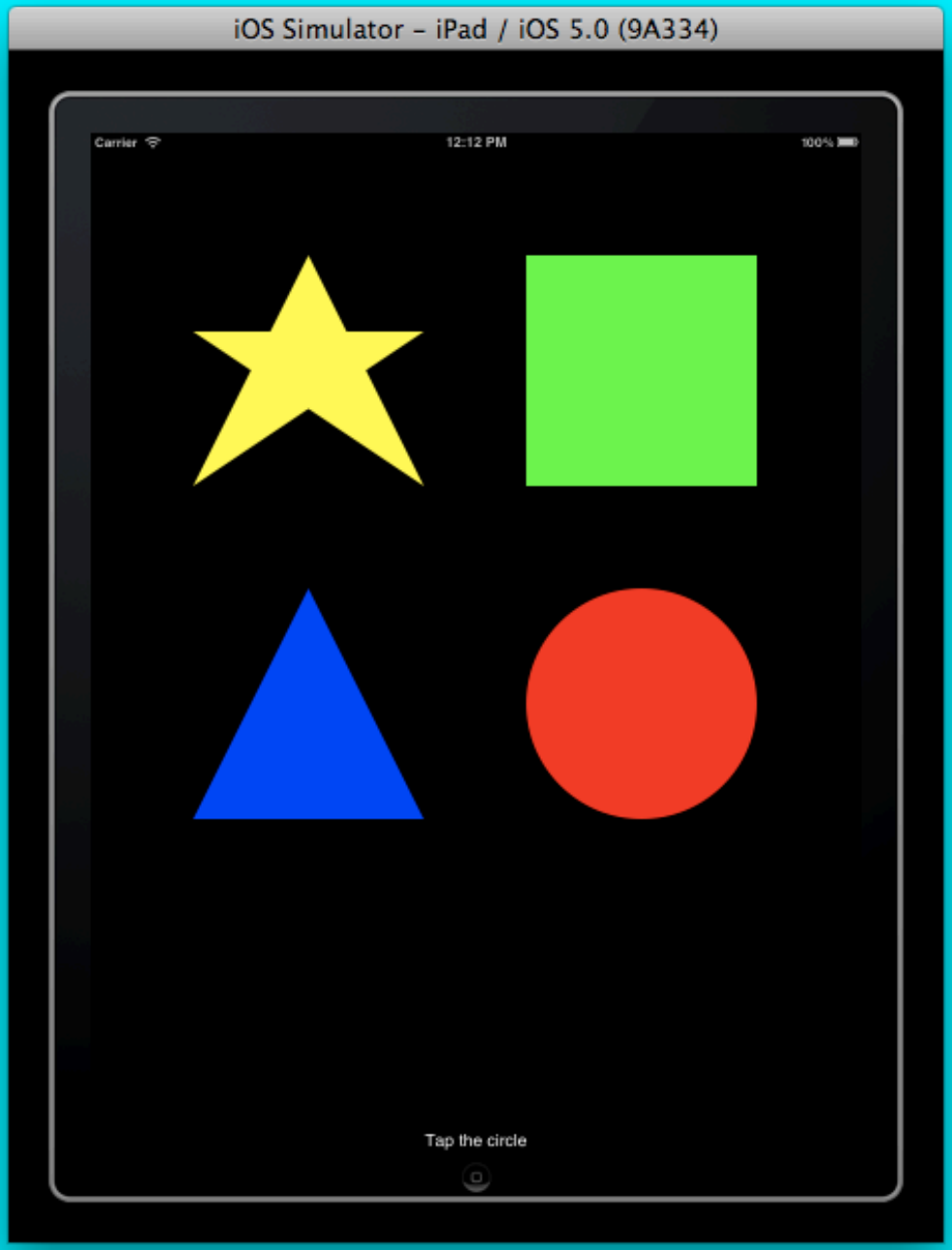
Run the app again with the console open.



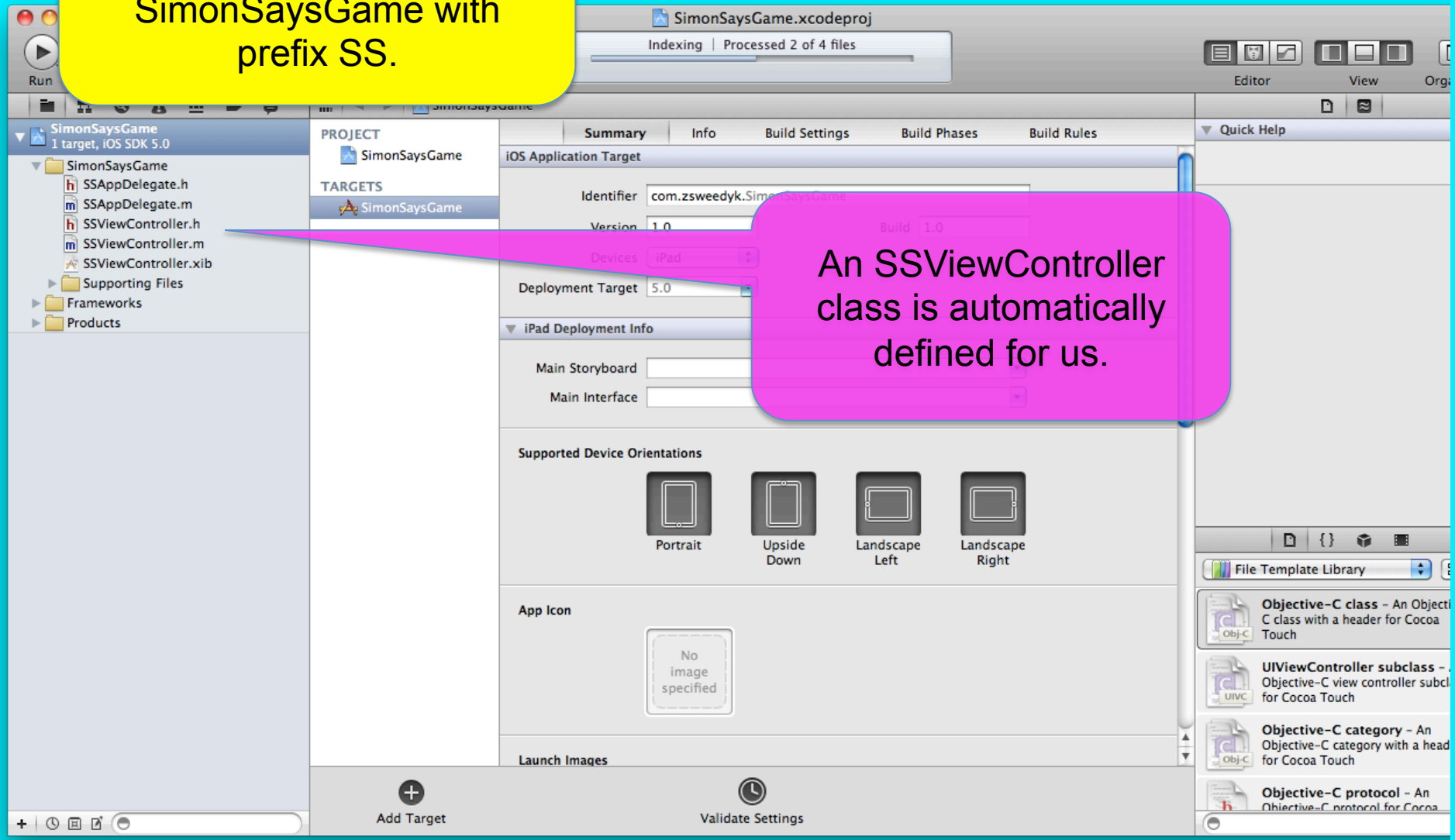
We are being warned that our app should have a root view controller.

What we did was an experiment. Until you become an iOS expert, you would be wise to use the template apps!

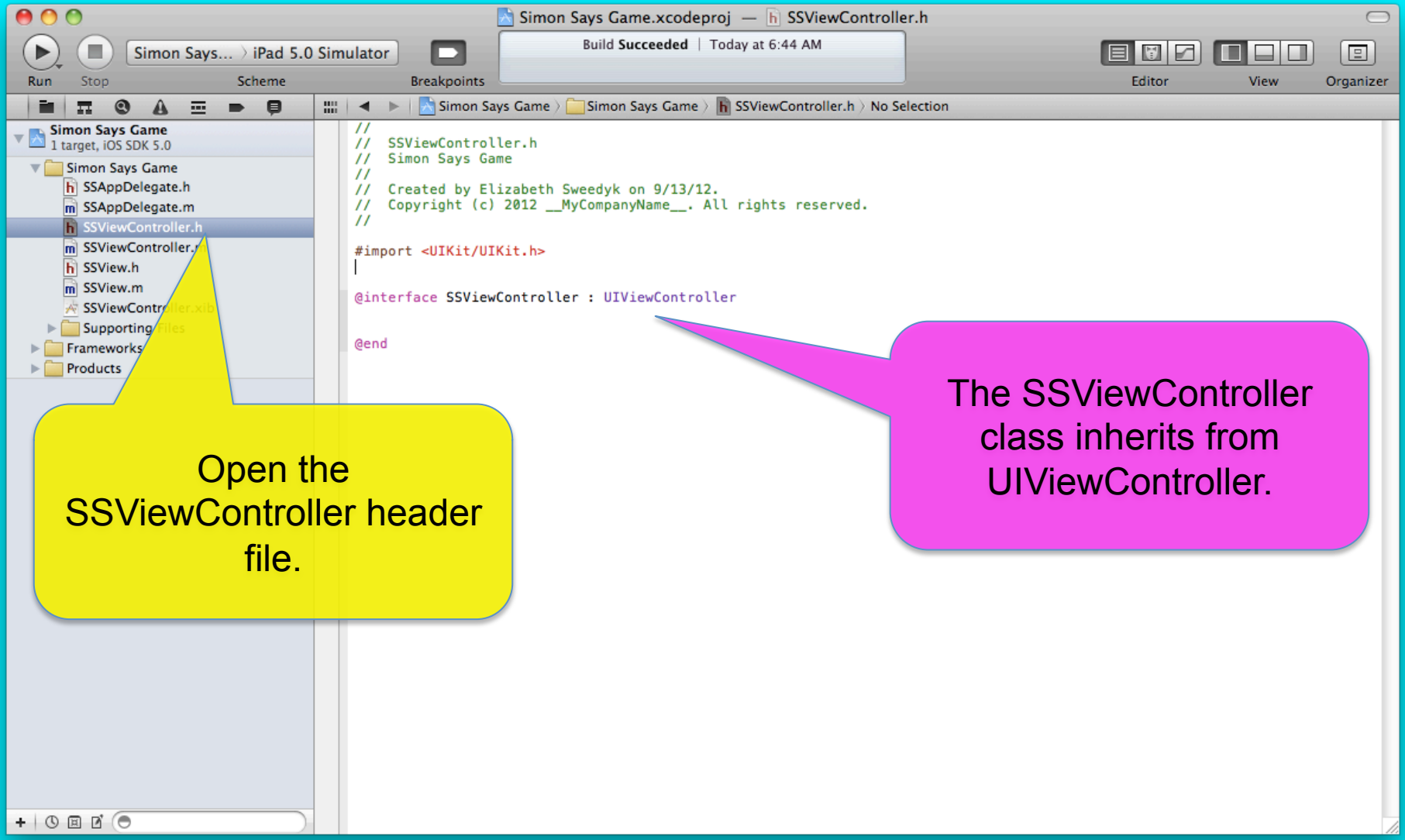
Let's start over. We are going to build a Simon Says game.



Create a new single view iPad app named SimonSaysGame with prefix SS.

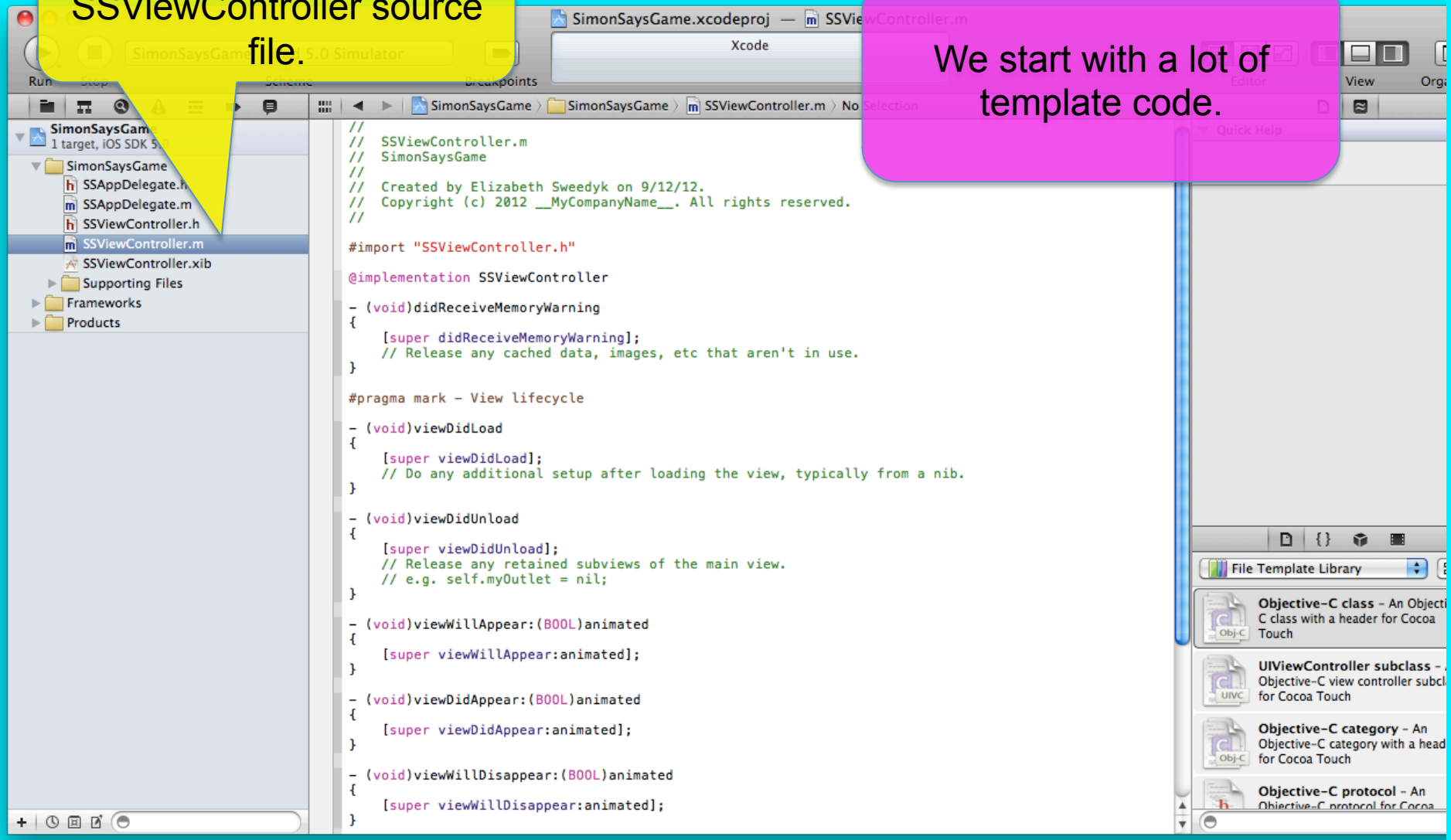


An SSViewController class is automatically defined for us.


Simon Says Game.xcodeproj — SSViewController.h
Build Succeeded | Today at 6:44 AM
Run
Stop
Simon Says... iPad 5.0 Simulator
Scheme
Breakpoints
Editor
View
Organizer
Simon Says Game
1 target, iOS SDK 5.0
Simon Says Game
SSAppDelegate.h
SSAppDelegate.m
SSViewController.h
SSViewController.m
SSView.h
SSView.m
SSViewController.xib
Supporting Files
Frameworks
Products
Simon Says Game > Simon Says Game > SSViewController.h > No Selection
//
// SSViewController.h
// Simon Says Game
//
// Created by Elizabeth Sweedyk on 9/13/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface SSViewController : UIViewController
@end
Open the SSViewController header file.
The SSViewController class inherits from UIViewController.

Open the SSViewController source file.

We start with a lot of template code.

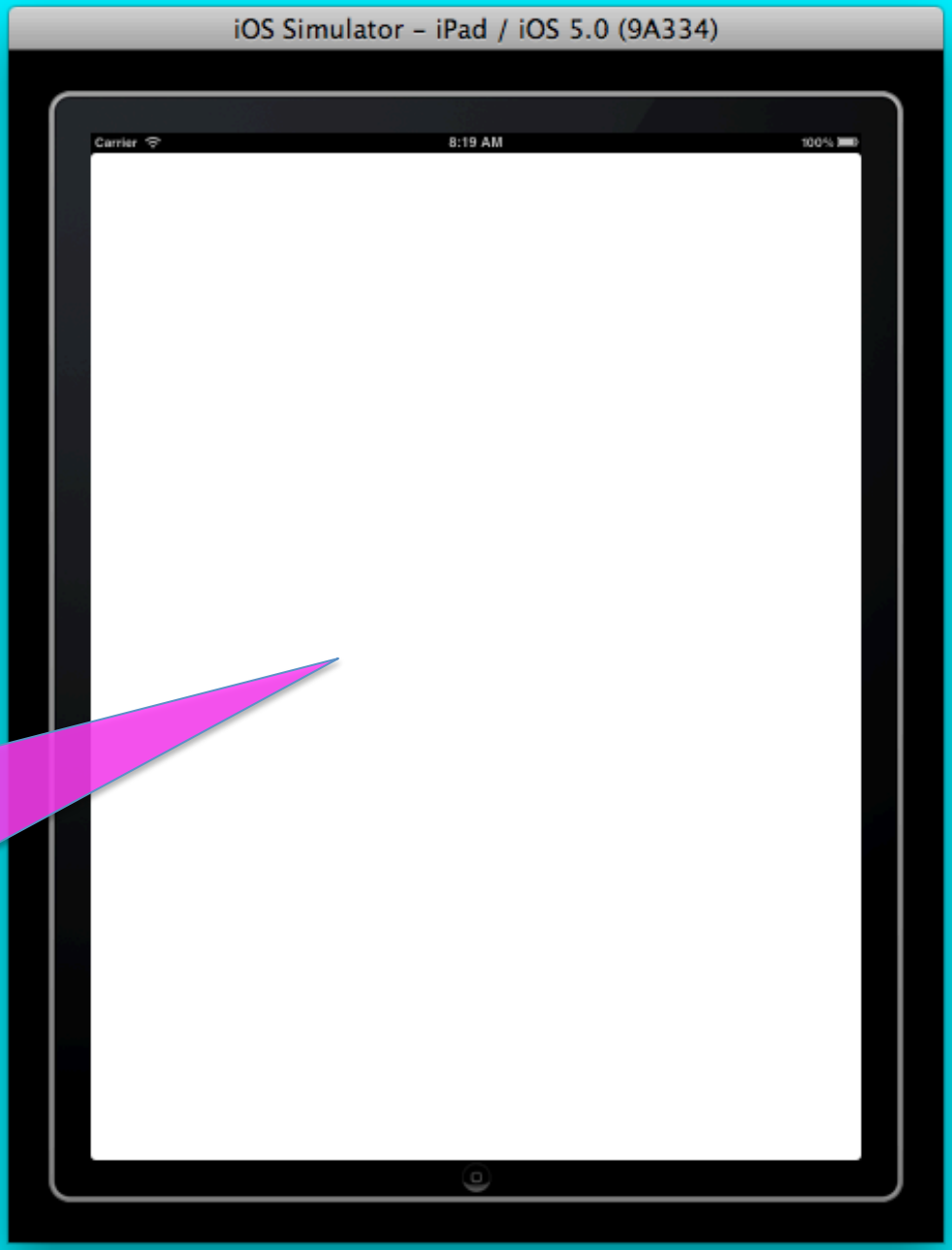


Open the AppDelegate source file.

We still have a window but this time the template code also instantiates our SSViewController class.

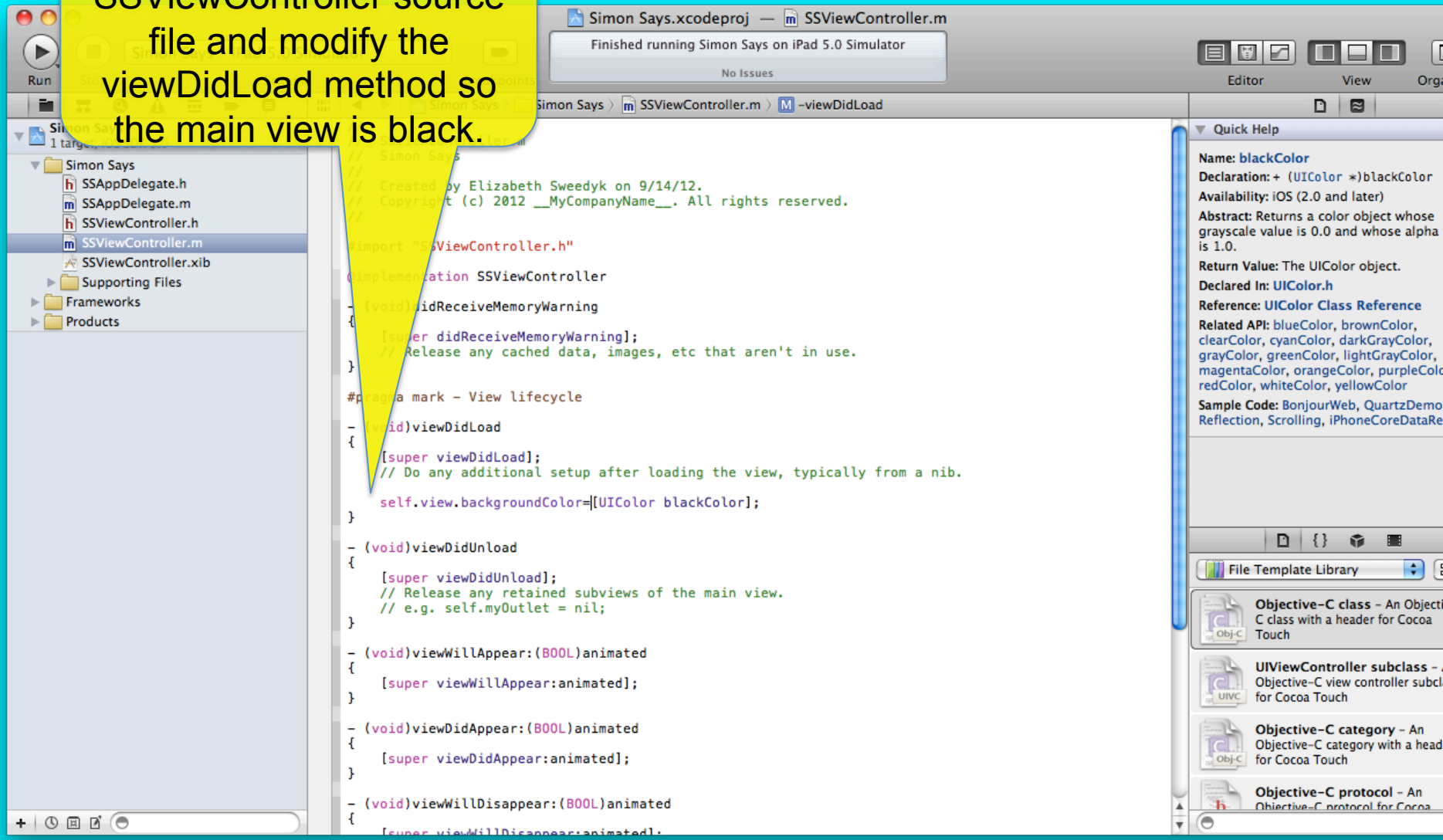
The SSViewController object will be the rootViewController for the app and the app's windows.

Run the app.



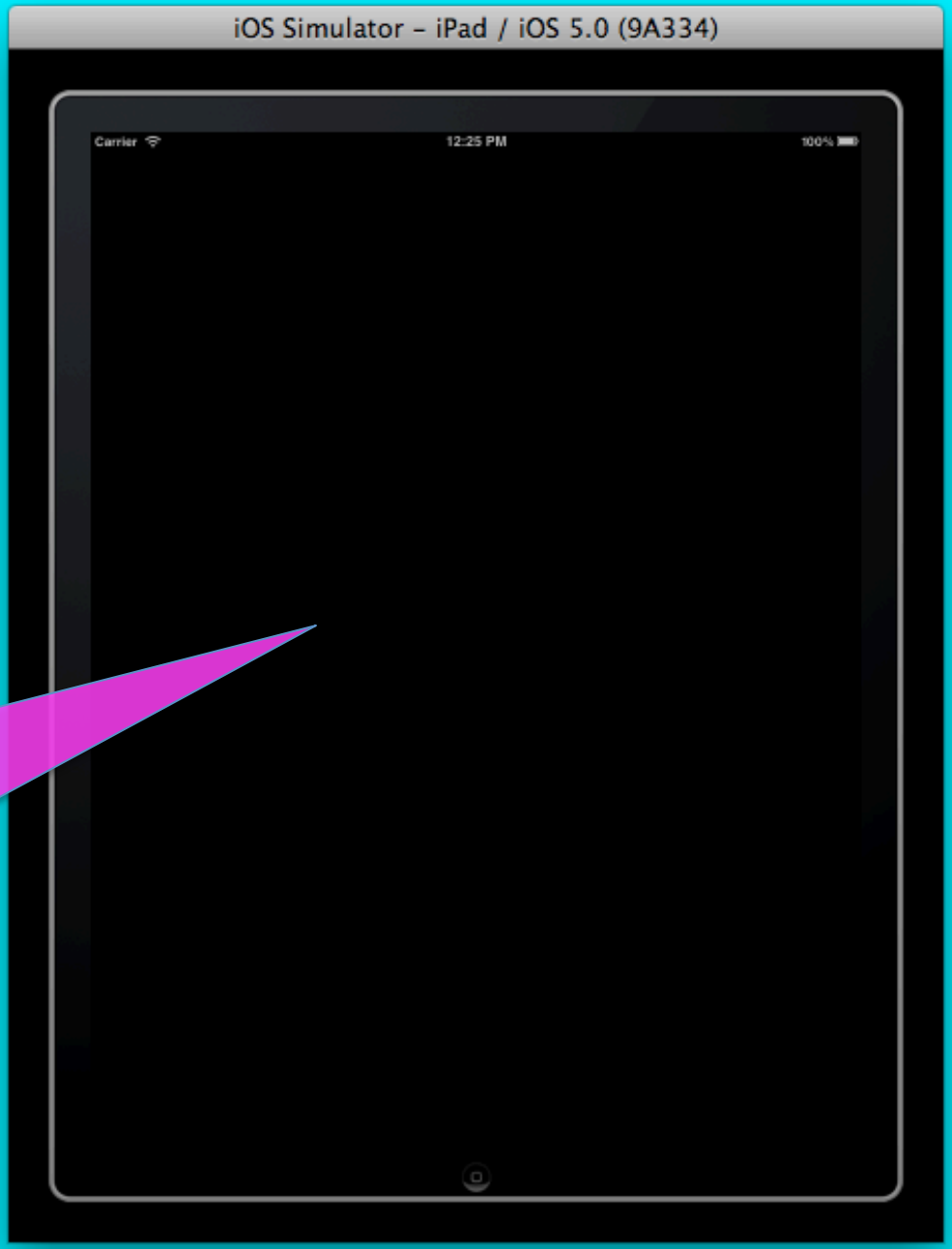
The UIViewController class has a pointer to a view. When the class is instantiated this “main view” is created as well.The default background color of the main view is white.

Open the SSViewController source file and modify the viewDidLoad method so the main view is black.



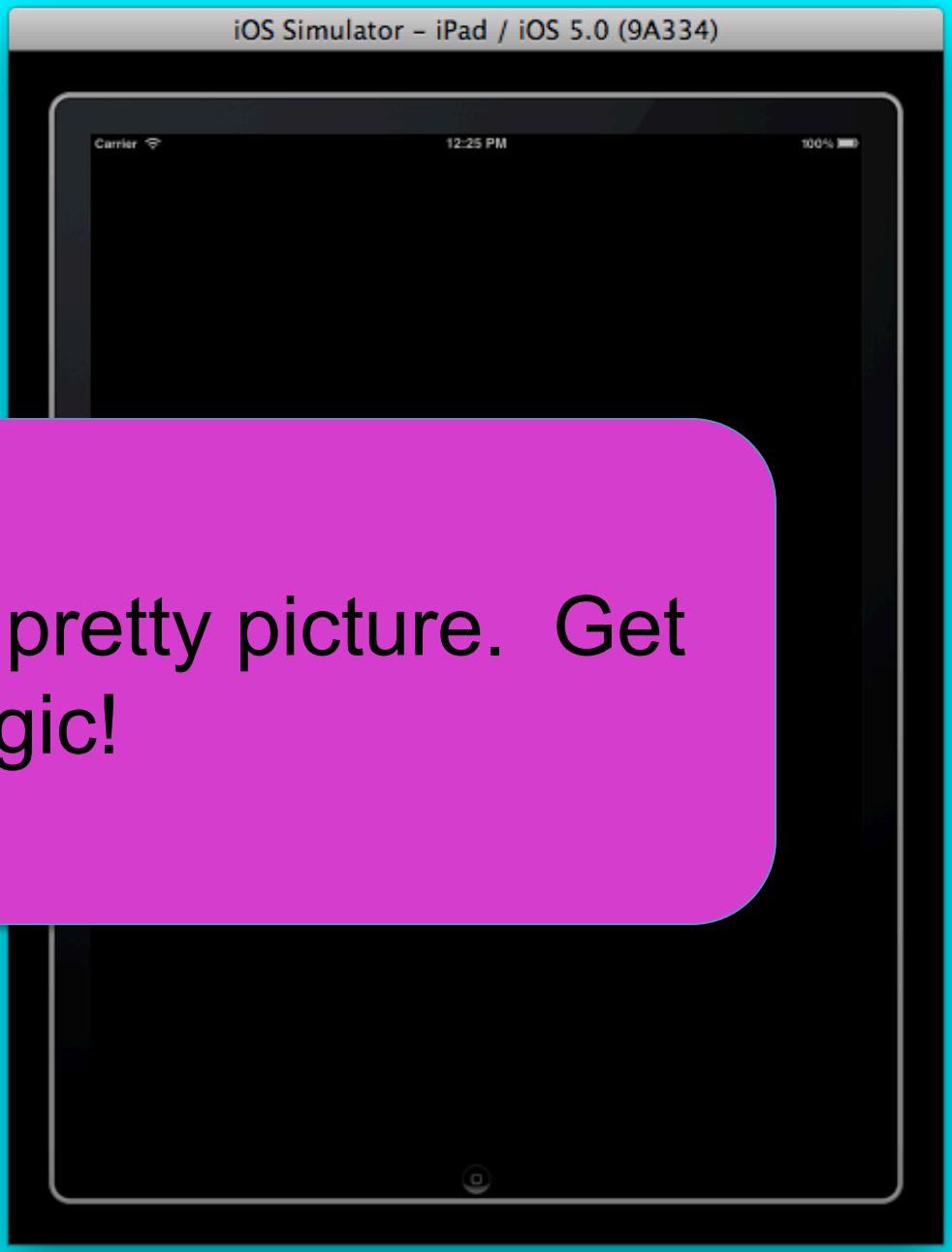
Run the app.

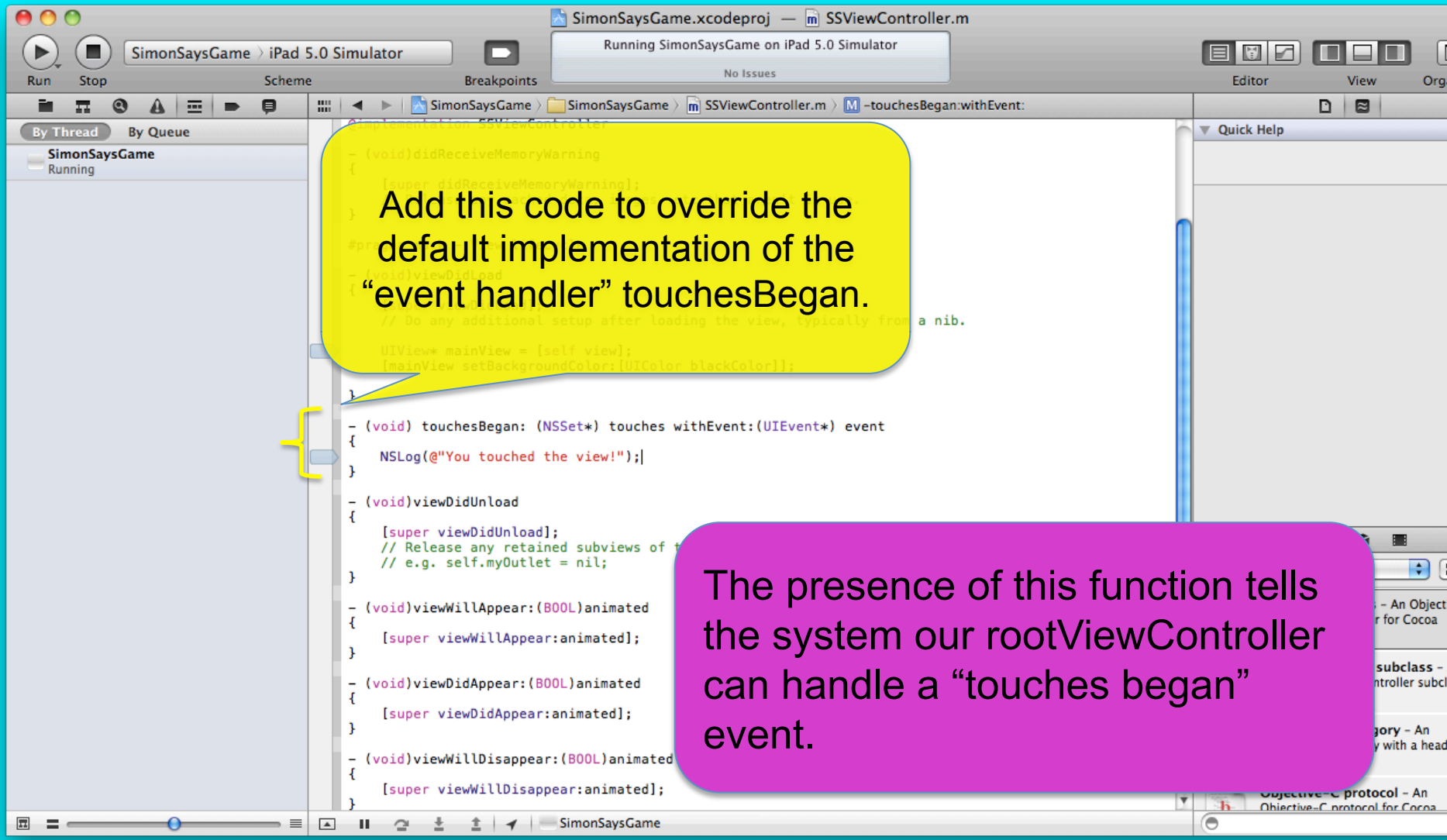
Now the rootViewController's main view is black.



Run the app again.

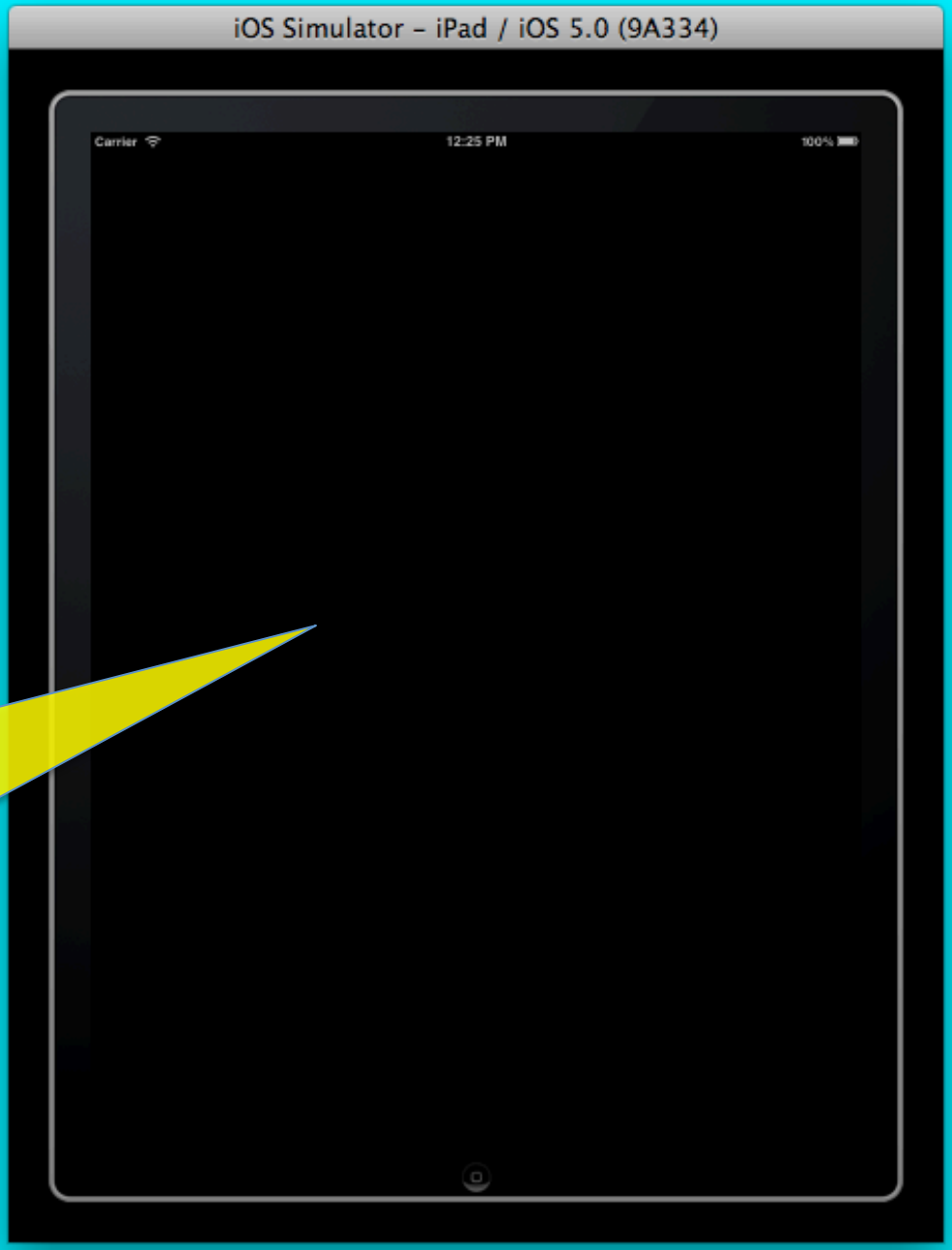
A view is not just a pretty picture. Get ready for some magic!

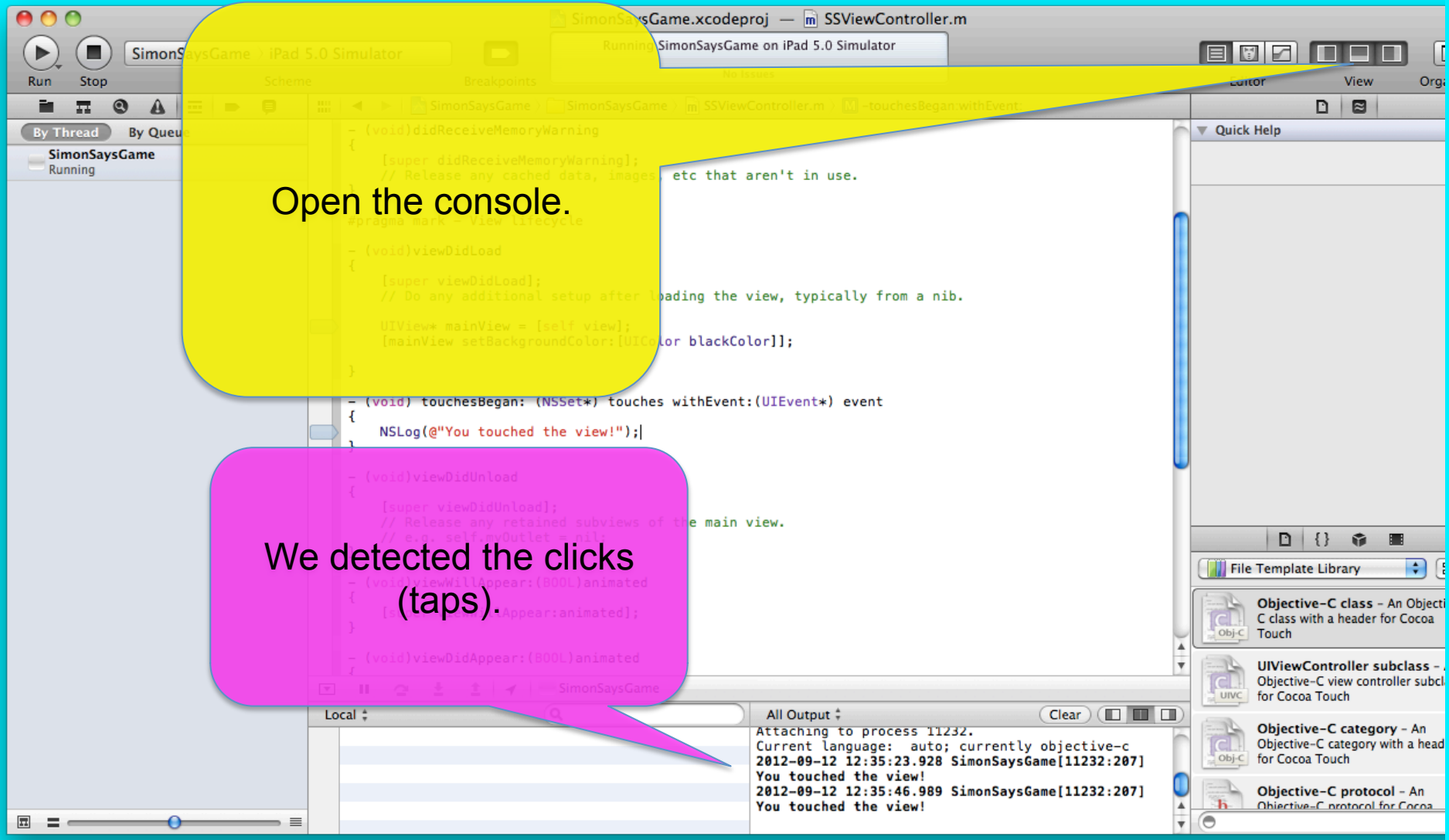



SimonSaysGame.xcodeproj — SSViewController.m
Run
Stop
SimonSaysGame › iPad 5.0 Simulator
Scheme
Breakpoints
Running SimonSaysGame on iPad 5.0 Simulator
No Issues
Editor
View
By Thread
By Queue
SimonSaysGame
Running
SimonSaysGame › SimonSaysGame › SSViewController.m › -touchesBegan:withEvent:
Quick Help
Add this code to override the default implementation of the "event handler" touchesBegan.
- (void) touchesBegan: (NSSet*) touches withEvent:(UIEvent*) event
{
NSLog(@"You touched the view!");
}
- (void)viewDidUnload
{
[super viewDidUnload];
// Release any retained subviews of t
// e.g. self.myOutlet = nil;
}
- (void)viewWillAppear:(BOOL)animated
{
[super viewWillAppear:animated];
}
- (void)viewDidAppear:(BOOL)animated
{
[super viewDidAppear:animated];
}
- (void)viewWillDisappear:(BOOL)animated
{
[super viewWillDisappear:animated];
}
The presence of this function tells the system our rootViewController can handle a "touches began" event.
SimonSaysGame

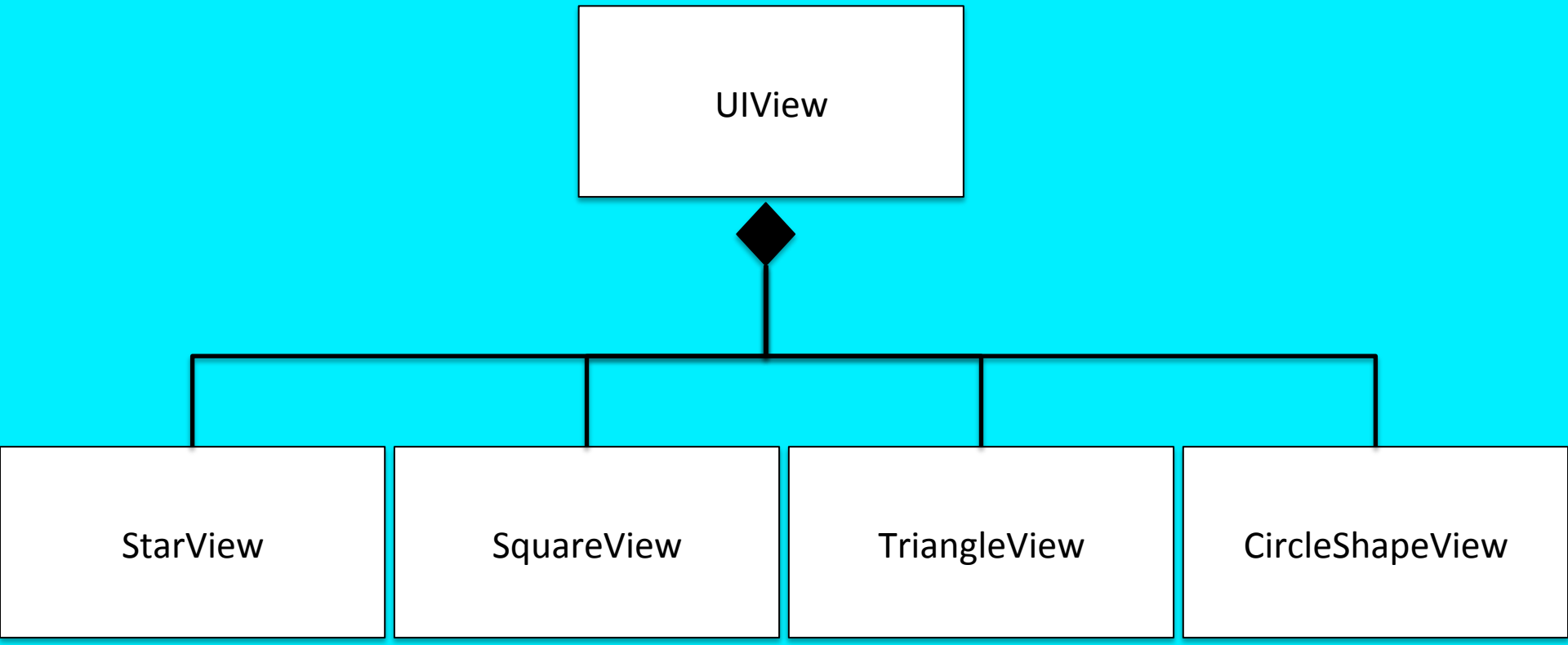
Run the app again.

Click on the screen a couple of times.

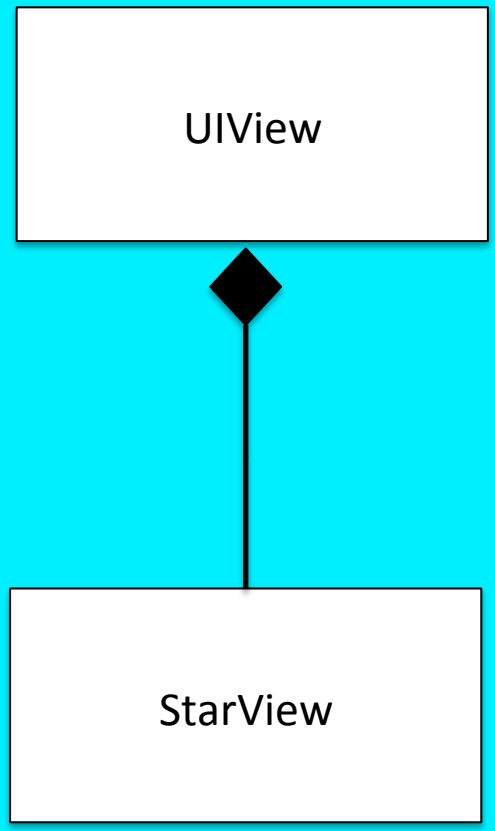



Open the console.
We detected the clicks (taps).
SSViewController.m
SimonSaysGame on iPad 5.0 Simulator
Run
Stop
Quick Help
File Template Library
Objective-C class
UIViewController subclass
Objective-C category
Objective-C protocol
- (void) touchesBegan: (NSSet*) touches withEvent:(UIEvent*) event
{
NSLog(@"You touched the view!");
}
Attaching to process 11232.
Current language: auto; currently objective-c
2012-09-12 12:35:23.928 SimonSaysGame[11232:207]
You touched the view!
2012-09-12 12:35:46.989 SimonSaysGame[11232:207]
You touched the view!

We'll create several custom views for our Simon Says game.

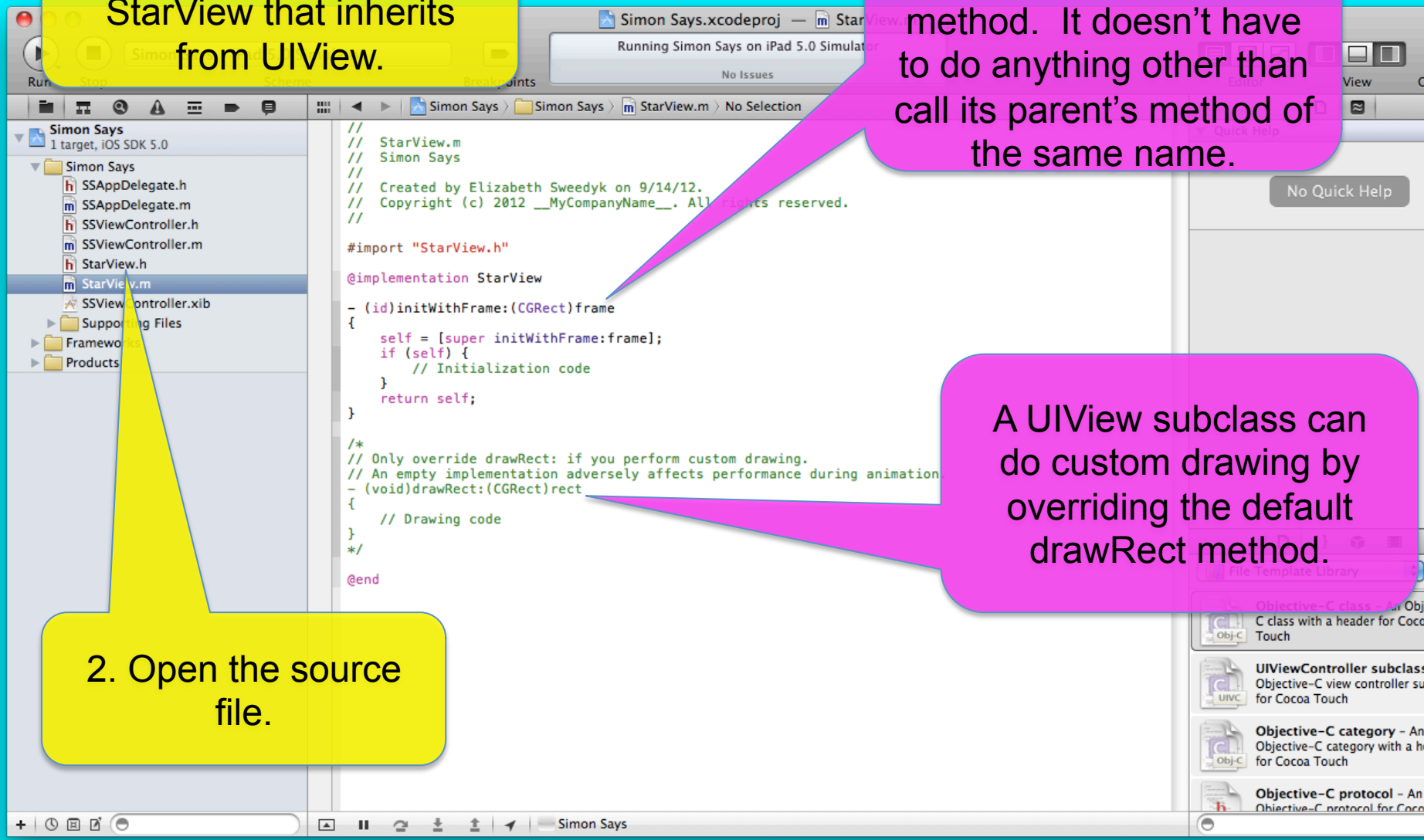


But let’s start simple.



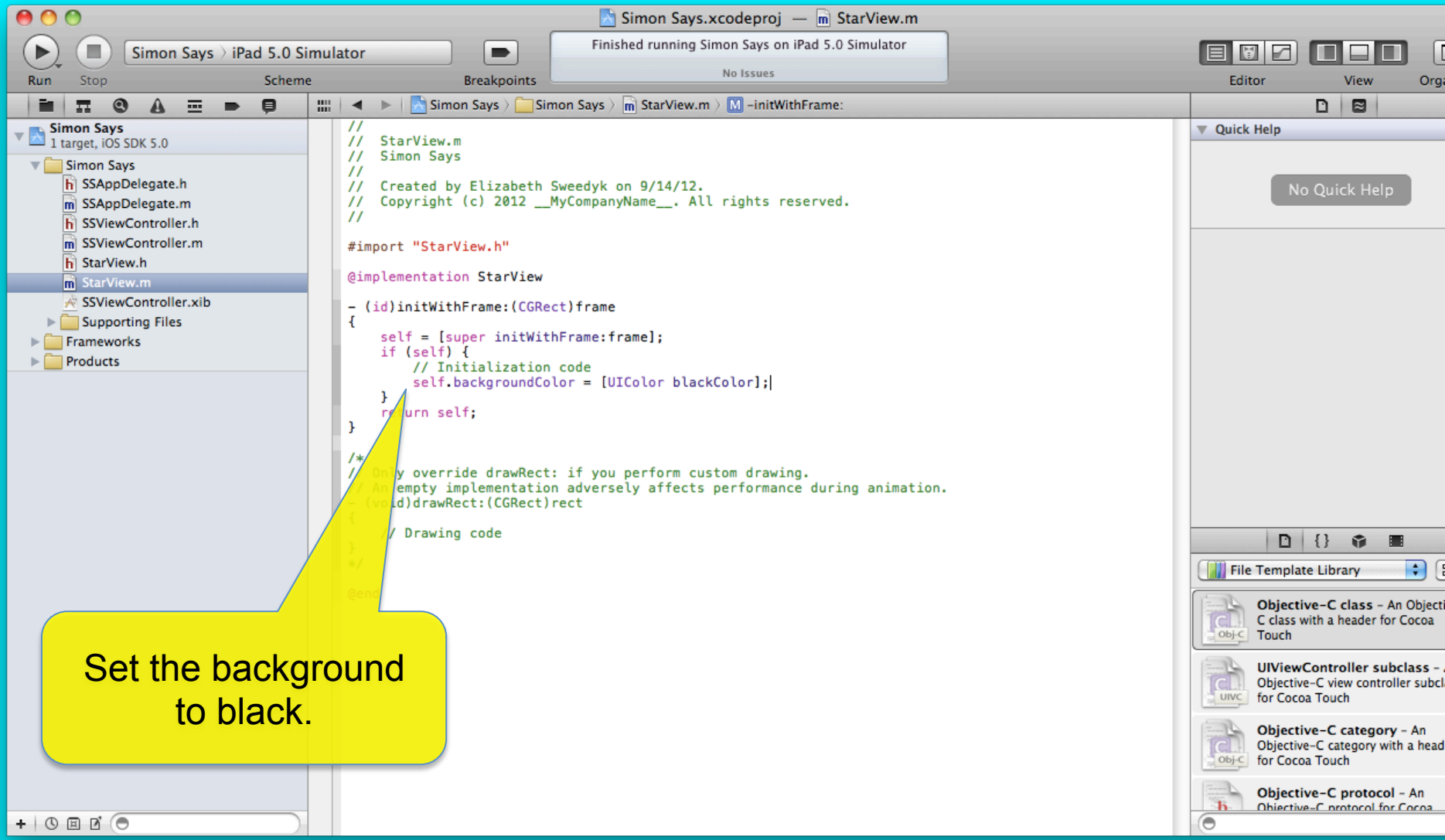
1. Create a new class StarView that inherits from UIView.

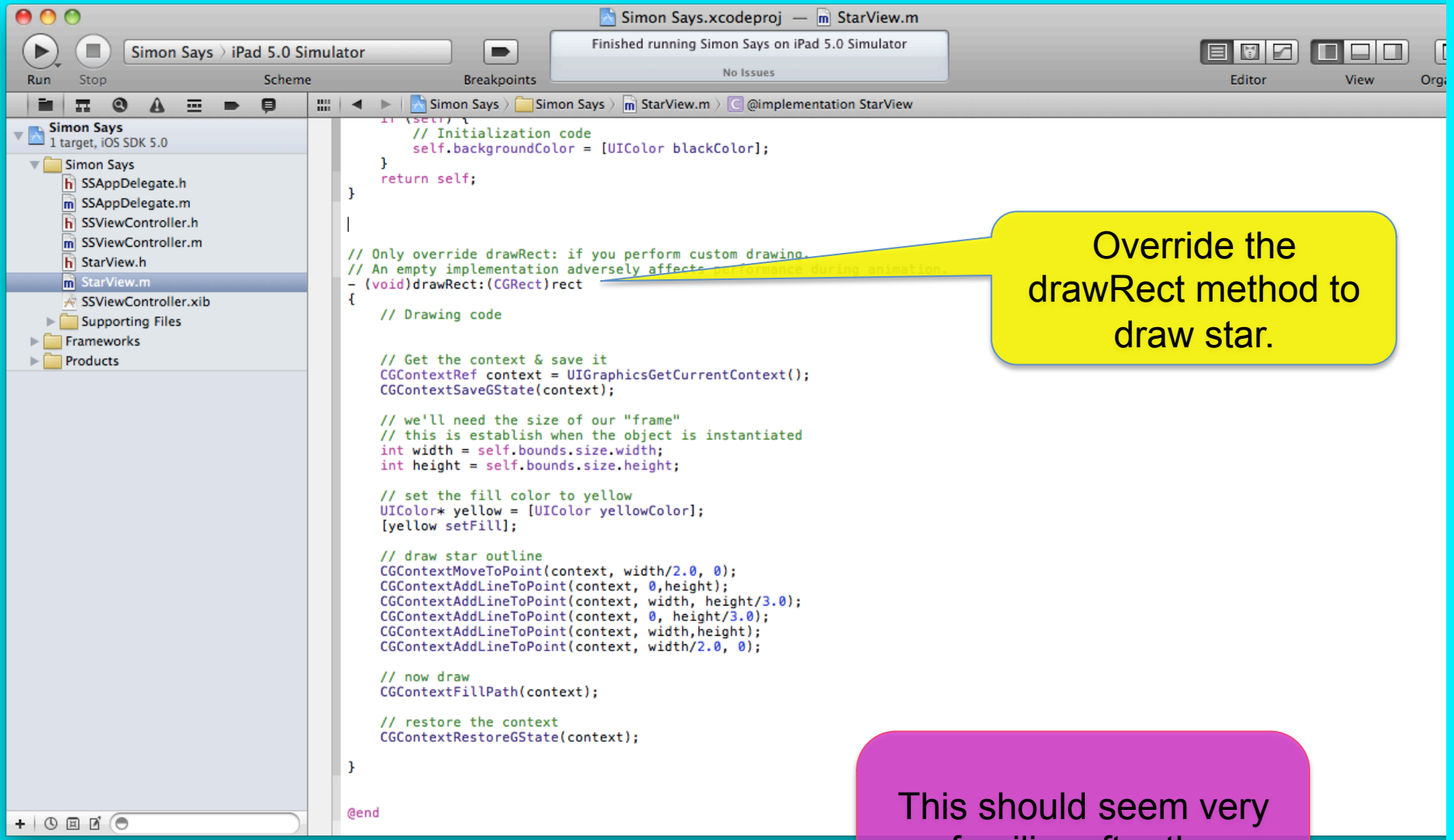
A UIView subclass should implement this method. It doesn't have to do anything other than call its parent's method of the same name.



A UIView subclass can do custom drawing by overriding the default drawRect method.

2. Open the source file.

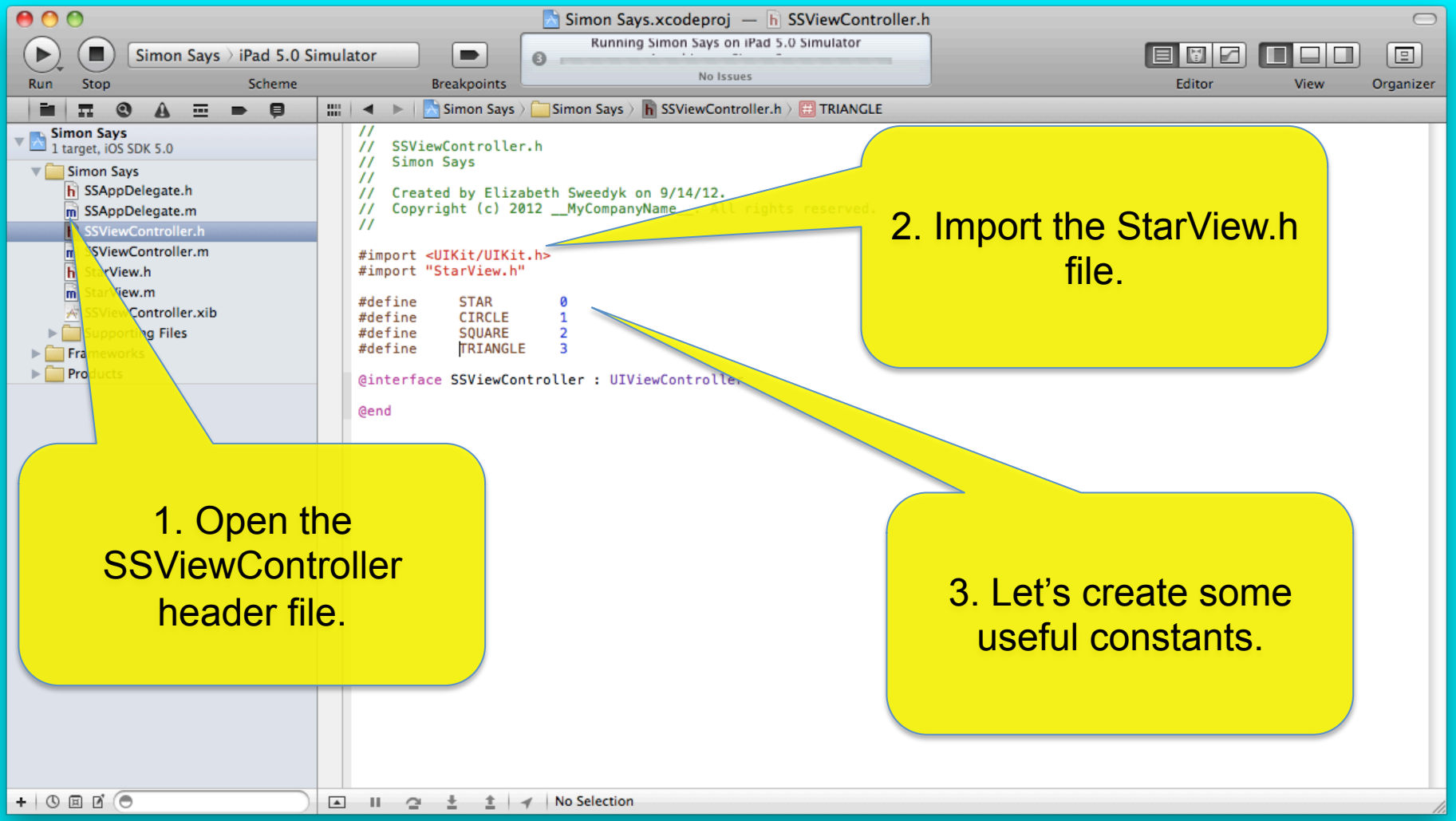

Simon Says.xcodeproj — StarView.m
Simon Says › iPad 5.0 Simulator
Finished running Simon Says on iPad 5.0 Simulator
No Issues
Run
Stop
Scheme
Breakpoints
Editor
View
Simon Says › Simon Says › StarView.m › -initWithFrame:
Simon Says
1 target, iOS SDK 5.0
Simon Says
SSAppDelegate.h
SSAppDelegate.m
SSViewController.h
SSViewController.m
StarView.h
StarView.m
SSViewController.xib
Supporting Files
Frameworks
Products
//
// StarView.m
// Simon Says
//
// Created by Elizabeth Sweedyk on 9/14/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import "StarView.h"
@implementation StarView
- (id)initWithFrame:(CGRect)frame
{
self = [super initWithFrame:frame];
if (self) {
// Initialization code
self.backgroundColor = [UIColor blackColor];
}
return self;
}
/*
// Only override drawRect: if you perform custom drawing.
// An empty implementation adversely affects performance during animation.
- (void)drawRect:(CGRect)rect
{
// Drawing code
}
*/
@end
Quick Help
No Quick Help
File Template Library
Objective-C class – An Objecti
C class with a header for Cocoa Touch
UIViewController subclass –
Objective-C view controller subcl
for Cocoa Touch
Objective-C category – An
Objective-C category with a head
for Cocoa Touch
Objective-C protocol – An
Set the background to black.

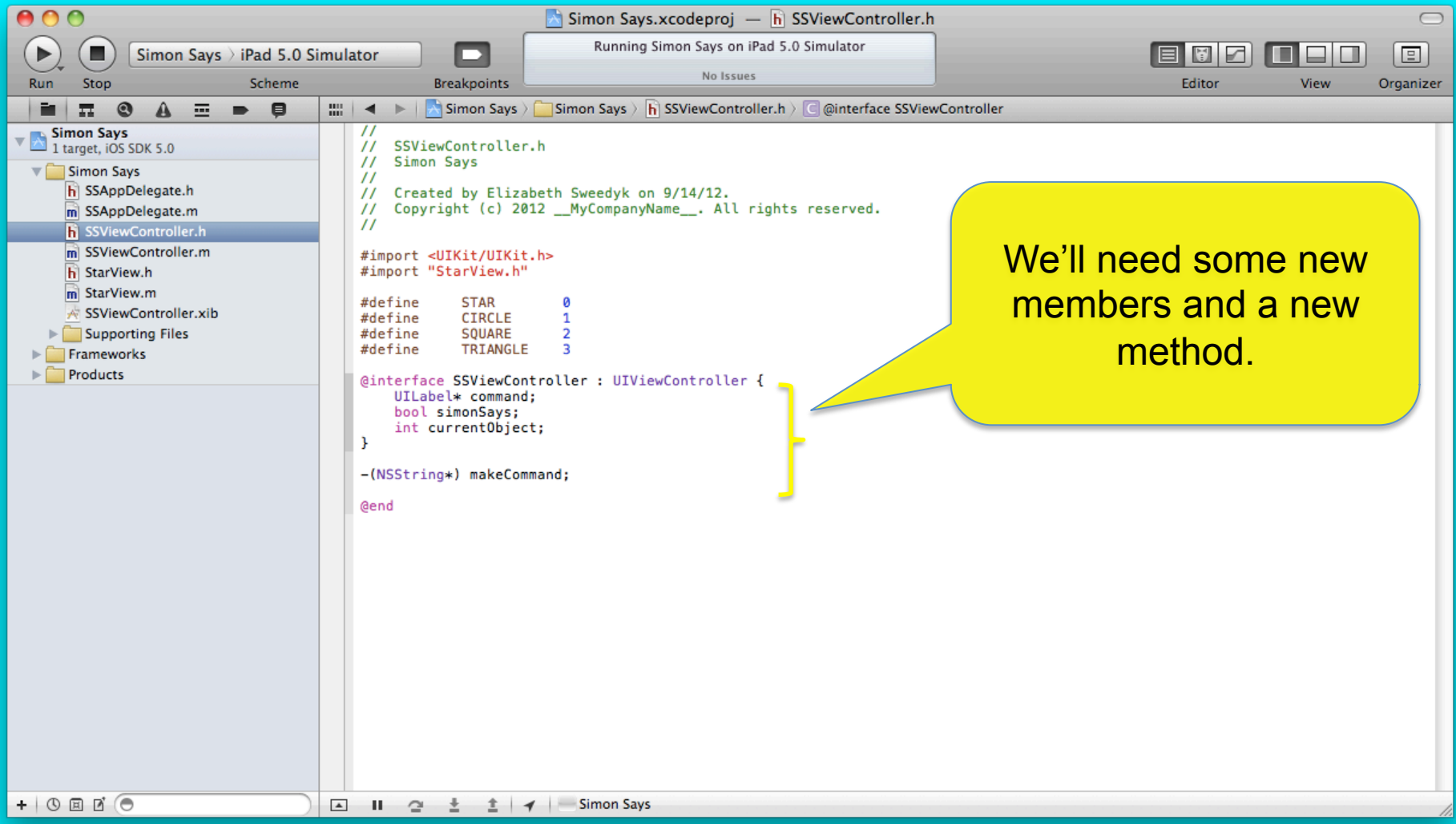


Override the drawRect method to draw star.

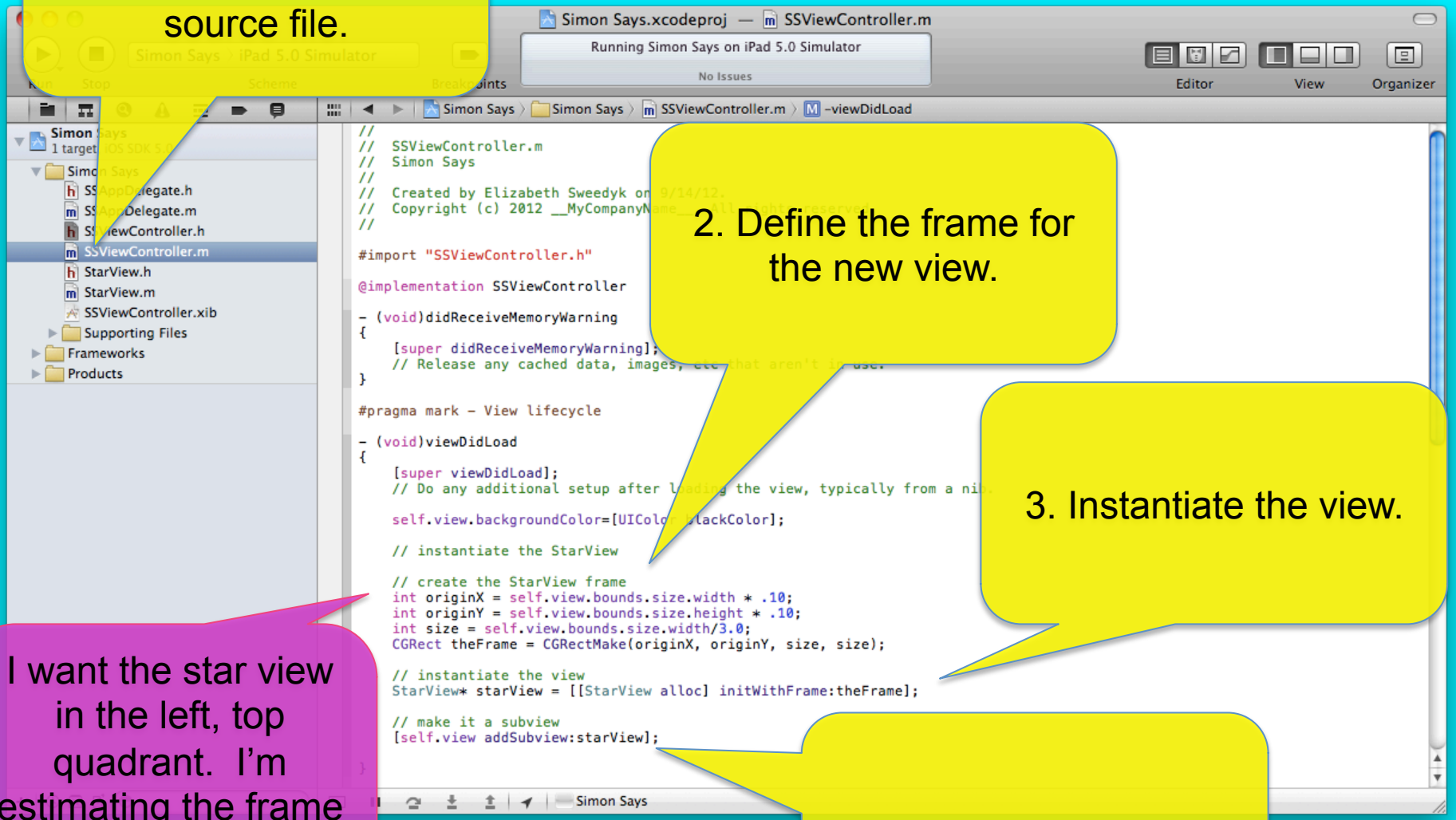
This should seem very familiar after the graphics tutorial.

Our StarView class will be instantiated by our rootViewController in the viewDidLoad method.


Simon Says.xcodeproj — SSViewController.h
Run
Stop
Simon Says › iPad 5.0 Simulator
Scheme
Breakpoints
Running Simon Says on iPad 5.0 Simulator
No Issues
Editor
View
Organizer
Simon Says › Simon Says › SSViewController.h › TRIANGLE
Simon Says
1 target, iOS SDK 5.0
Simon Says
SSAppDelegate.h
SSAppDelegate.m
SSViewController.h
SSViewController.m
SSViewController.xib
Supporting Files
Frameworks
Products
//
// SSViewController.h
// Simon Says
//
// Created by Elizabeth Sweedyk on 9/14/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
#import "StarView.h"
#define STAR 0
#define CIRCLE 1
#define SQUARE 2
#define TRIANGLE 3
@interface SSViewController : UIViewController
@end
No Selection
1. Open the SSViewController header file.
2. Import the StarView.h file.
3. Let's create some useful constants.


Simon Says.xcodeproj — SSViewController.h
Run
Stop
Simon Says › iPad 5.0 Simulator
Scheme
Breakpoints
Running Simon Says on iPad 5.0 Simulator
No Issues
Editor
View
Organizer
Simon Says
1 target, iOS SDK 5.0
Simon Says
SSAppDelegate.h
SSAppDelegate.m
SSViewController.h
SSViewController.m
StarView.h
StarView.m
SSViewController.xib
Supporting Files
Frameworks
Products
Simon Says › Simon Says › SSViewController.h › @interface SSViewController
//
// SSViewController.h
// Simon Says
//
// Created by Elizabeth Sweedyk on 9/14/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
#import "StarView.h"
#define STAR 0
#define CIRCLE 1
#define SQUARE 2
#define TRIANGLE 3
@interface SSViewController : UIViewController {
UILabel* command;
bool simonSays;
int currentObject;
}
-(NSString*) makeCommand;
@end
We'll need some new members and a new method.
Simon Says

1. Open the SSViewController source file.



2. Define the frame for the new view.

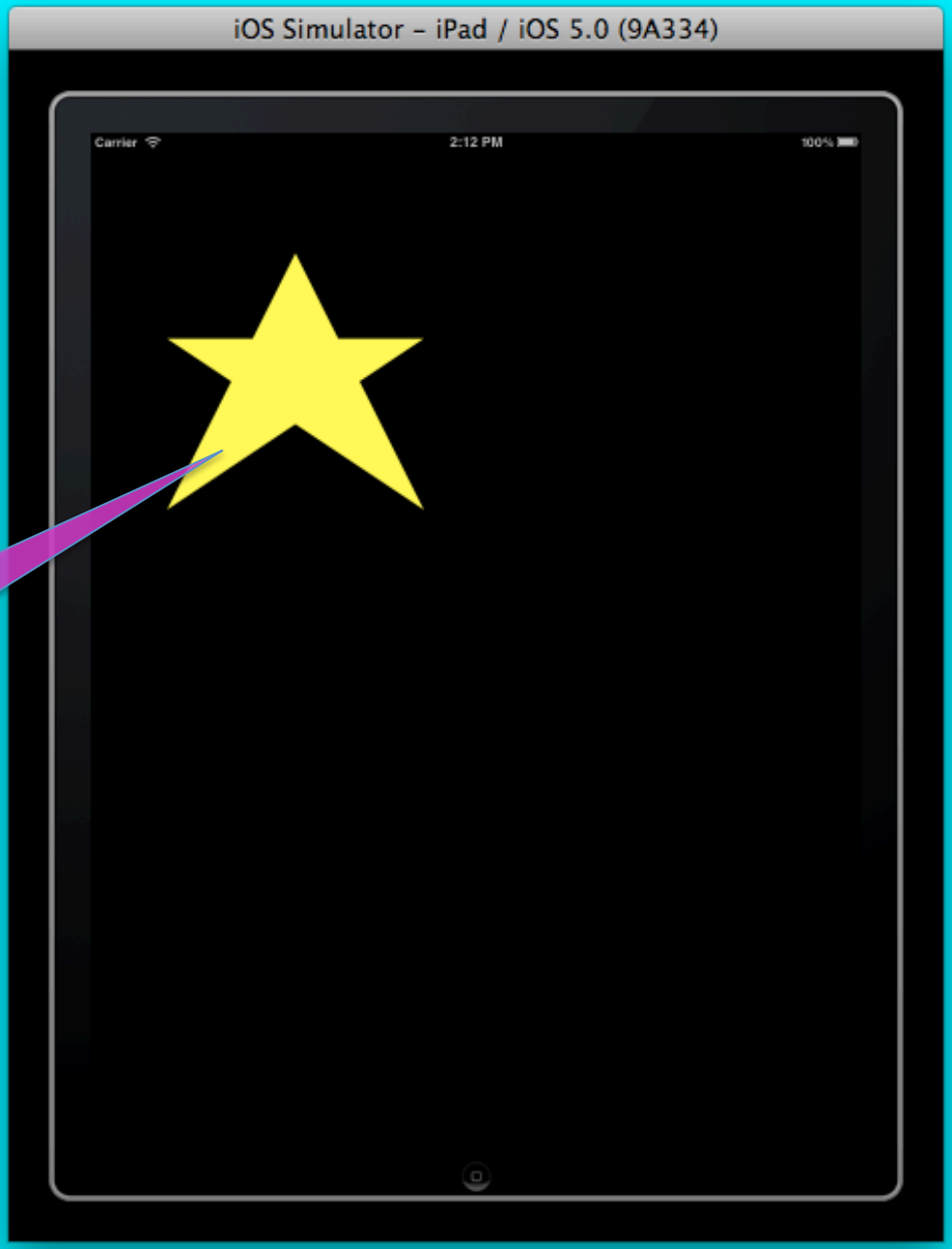
3. Instantiate the view.

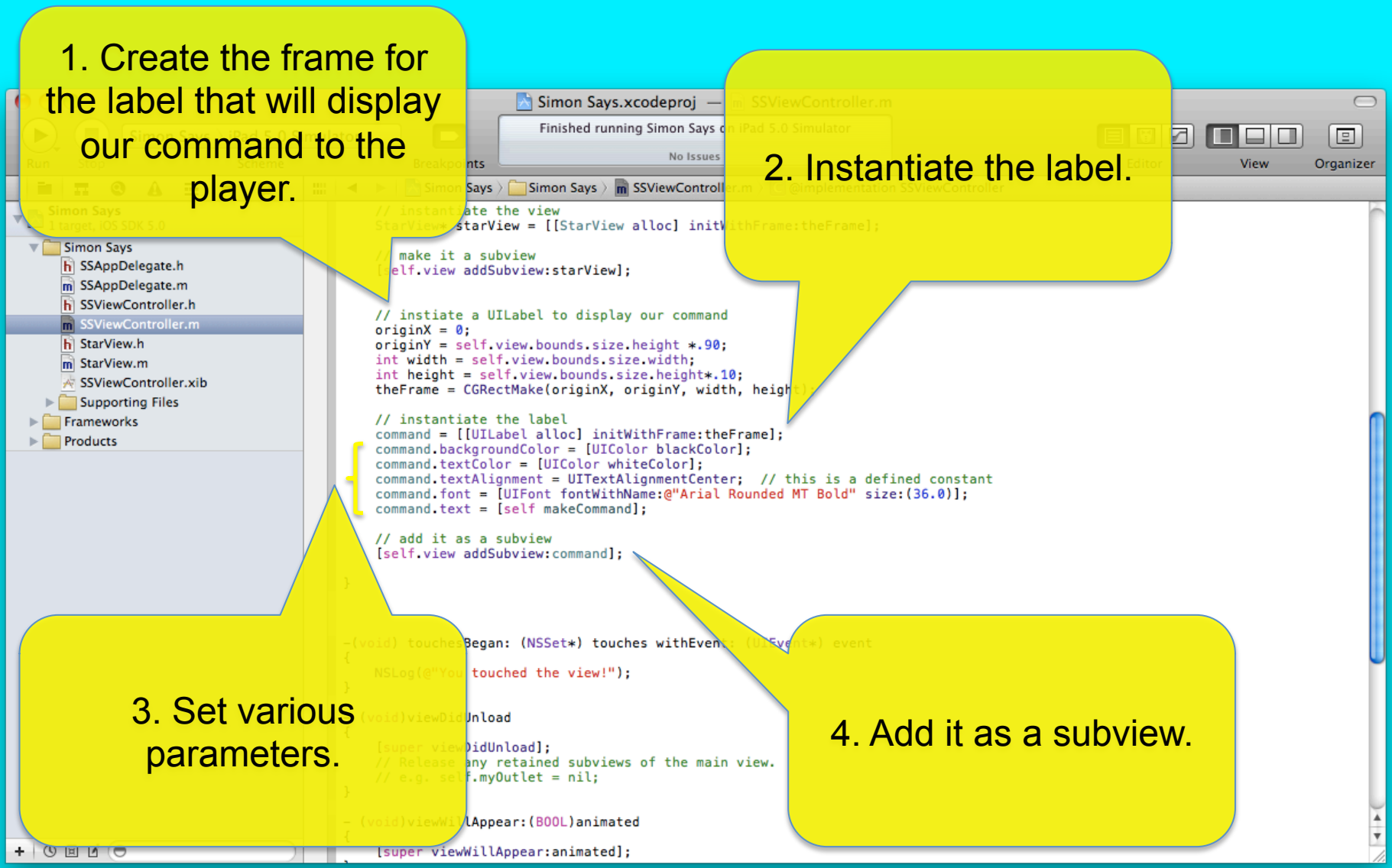
I want the star view in the left, top quadrant. I'm estimating the frame based on the size of the screen.

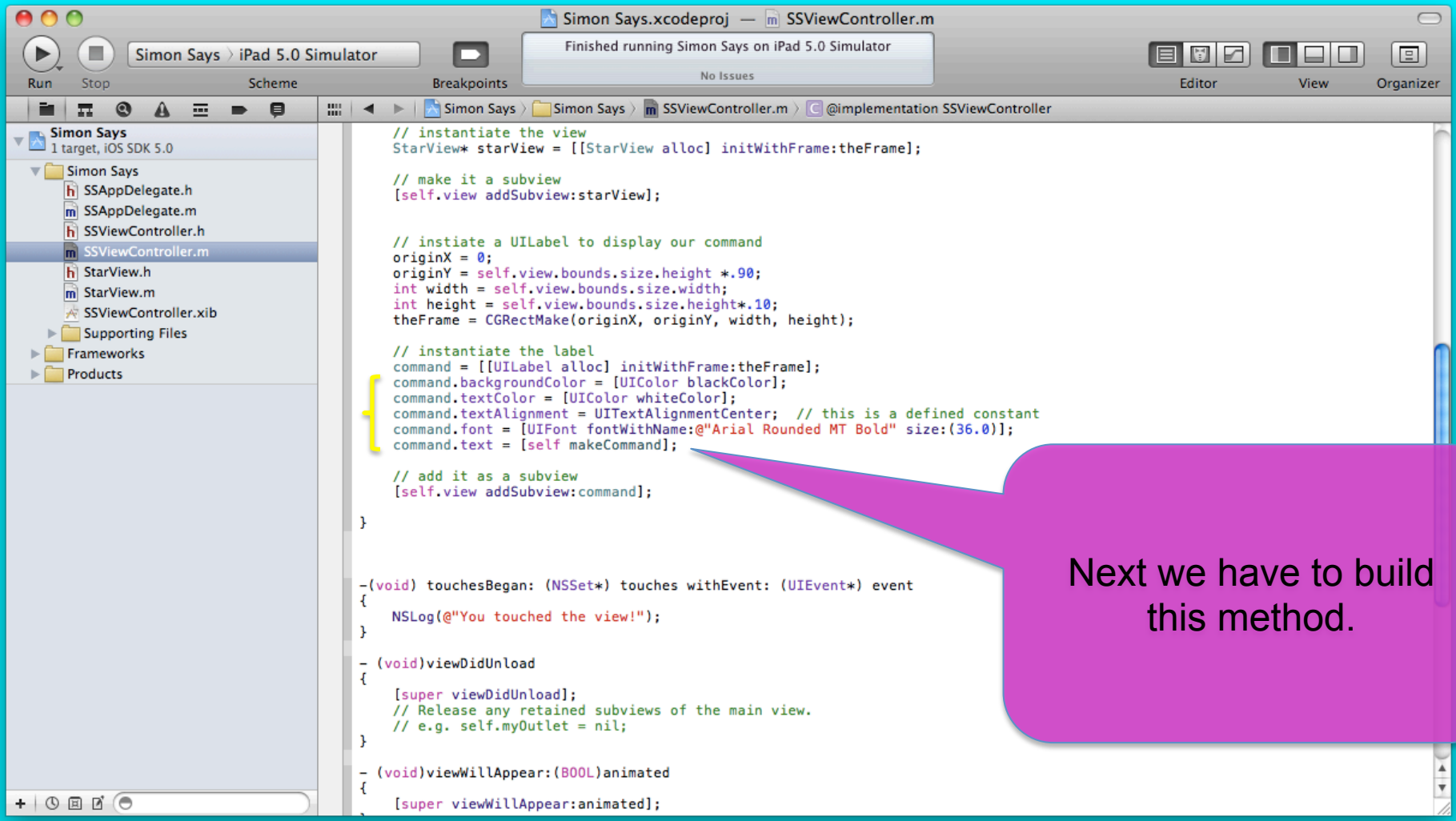
4. Add it as a subview.

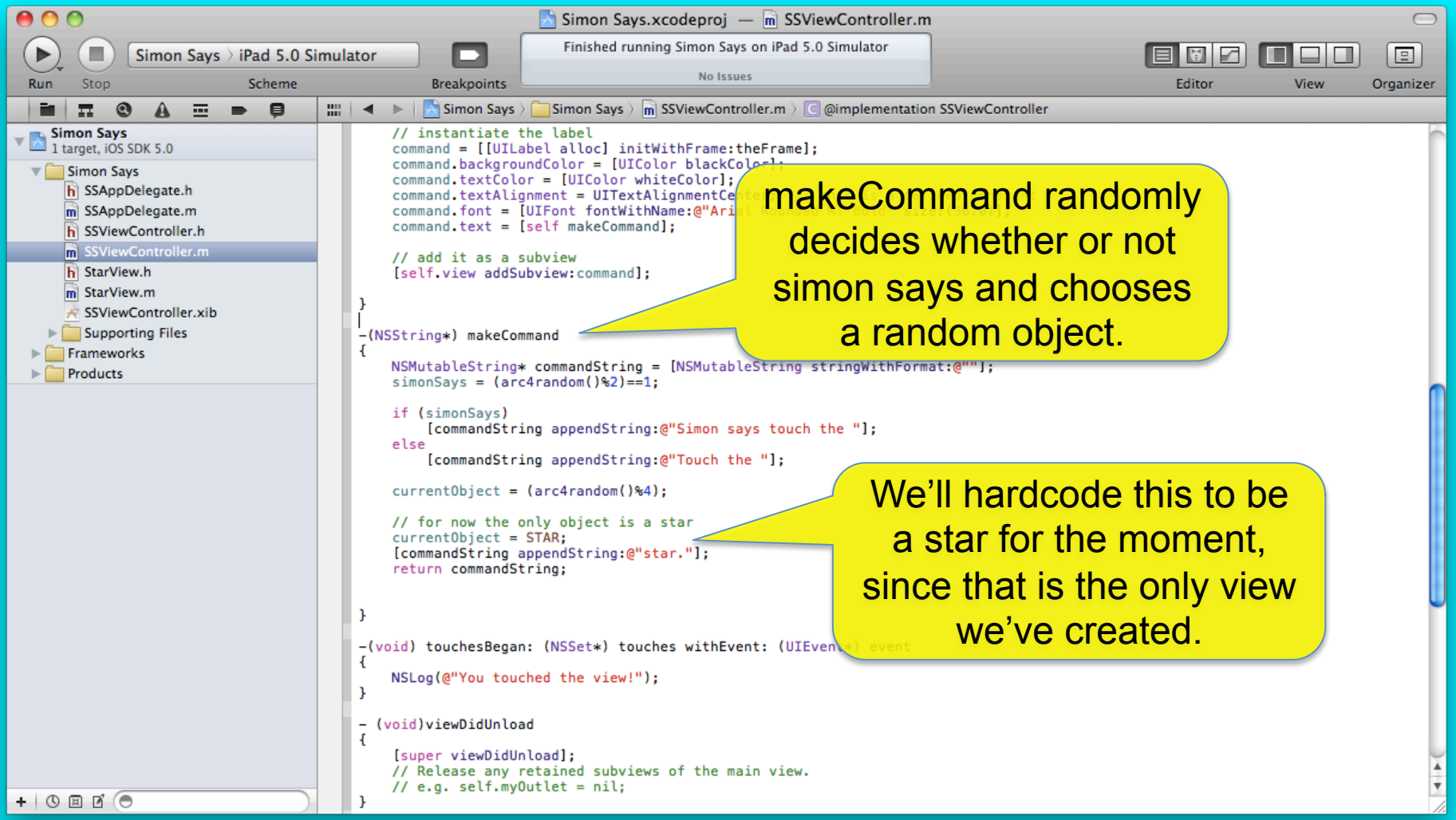
Run the app.

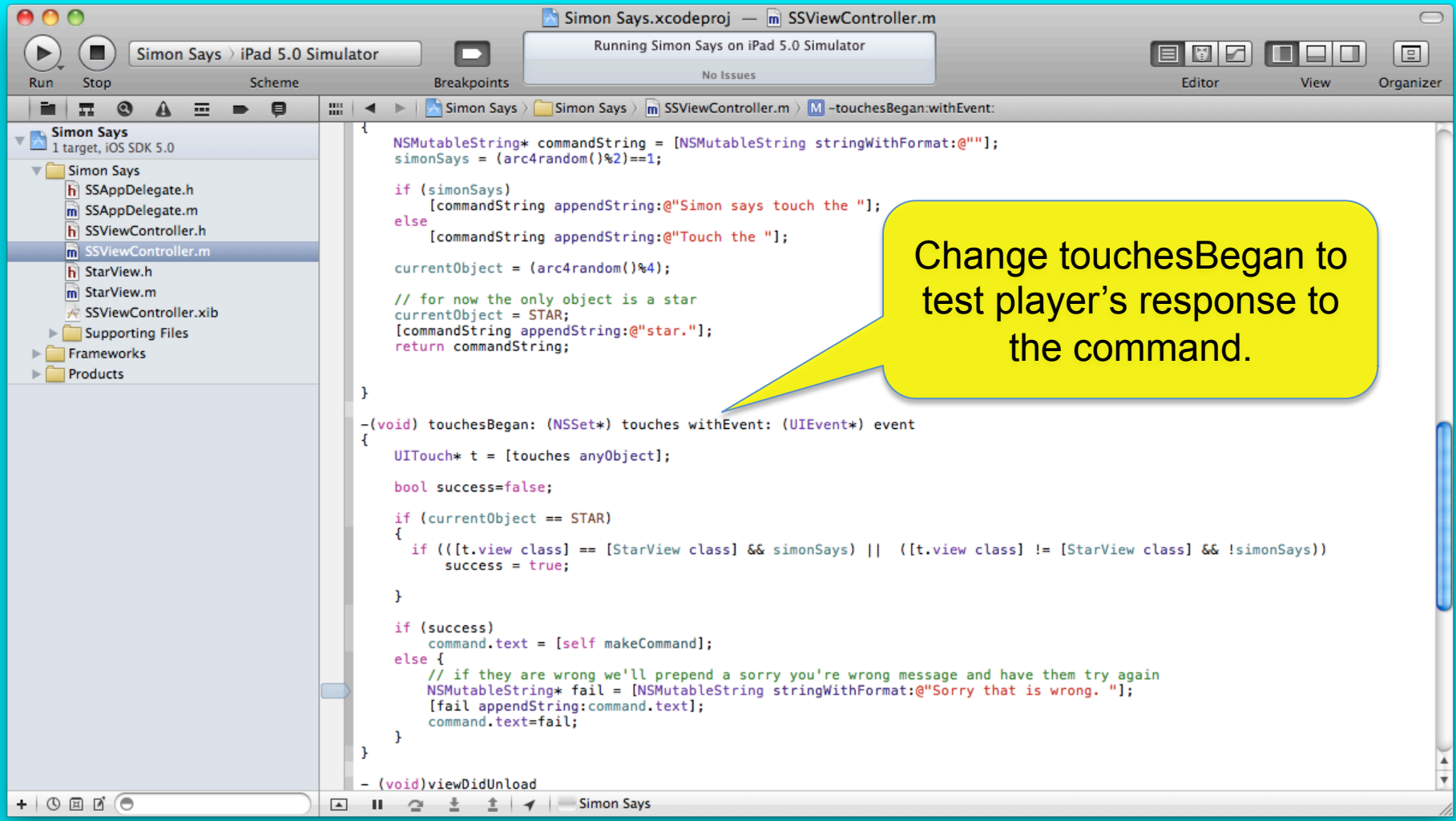
Here is our view!

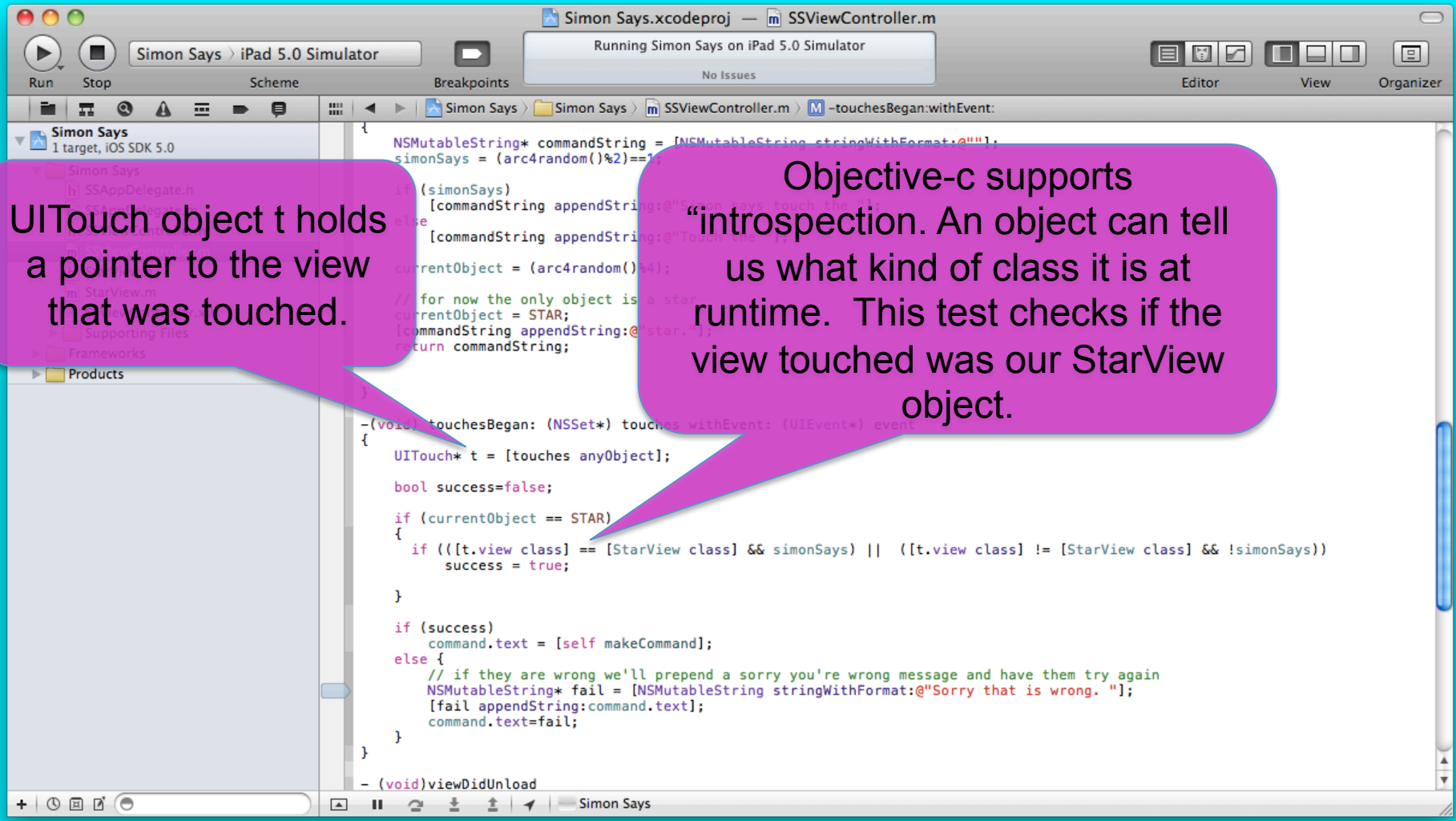



1. Create the frame for the label that will display our command to the player.
2. Instantiate the label.
3. Set various parameters.
4. Add it as a subview.
Simon Says.xcodeproj
Finished running Simon Says on iPad 5.0 Simulator
No Issues
SSViewController.m
View
Organizer
Simon Says
Simon Says
SSAppDelegate.h
SSAppDelegate.m
SSViewController.h
SSViewController.m
StarView.h
StarView.m
SSViewController.xib
Supporting Files
Frameworks
Products
// instantiate the view
StarView *starView = [[StarView alloc] initWithFrame:theFrame];
// make it a subview
[self.view addSubview:starView];
// instiate a UILabel to display our command
originX = 0;
originY = self.view.bounds.size.height *.90;
int width = self.view.bounds.size.width;
int height = self.view.bounds.size.height*.10;
theFrame = CGRectMake(originX, originY, width, height);
// instantiate the label
command = [[UILabel alloc] initWithFrame:theFrame];
command.backgroundColor = [UIColor blackColor];
command.textColor = [UIColor whiteColor];
command.textAlignment = UITextAlignmentCenter; // this is a defined constant
command.font = [UIFont fontWithName:@"Arial Rounded MT Bold" size:(36.0)];
command.text = [self makeCommand];
// add it as a subview
[self.view addSubview:command];
touchesBegan: (NSSet*) touches withEvent: (UIEvent*) event
NSLog(@"You touched the view!");
viewDidUnload
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
viewWillAppear:(BOOL)animated
[super viewWillAppear:animated];


Simon Says.xcodeproj — SSViewController.m
Run
Stop
Simon Says › iPad 5.0 Simulator
Scheme
Breakpoints
Finished running Simon Says on iPad 5.0 Simulator
No Issues
Editor
View
Organizer
Simon Says
1 target, iOS SDK 5.0
Simon Says
SSAppDelegate.h
SSAppDelegate.m
SSViewController.h
SSViewController.m
StarView.h
StarView.m
SSViewController.xib
Supporting Files
Frameworks
Products
Simon Says › Simon Says › SSViewController.m › @implementation SSViewController
// instantiate the view
StarView* starView = [[StarView alloc] initWithFrame:theFrame];
// make it a subview
[self.view addSubview:starView];
// instiate a UILabel to display our command
originX = 0;
originY = self.view.bounds.size.height *.90;
int width = self.view.bounds.size.width;
int height = self.view.bounds.size.height*.10;
theFrame = CGRectMake(originX, originY, width, height);
// instantiate the label
command = [[UILabel alloc] initWithFrame:theFrame];
command.backgroundColor = [UIColor blackColor];
command.textColor = [UIColor whiteColor];
command.textAlignment = UITextAlignmentCenter; // this is a defined constant
command.font = [UIFont fontWithName:@"Arial Rounded MT Bold" size:(36.0)];
command.text = [self makeCommand];
// add it as a subview
[self.view addSubview:command];
}
-(void) touchesBegan: (NSSet*) touches withEvent: (UIEvent*) event
{
NSLog(@"You touched the view!");
}
- (void)viewDidUnload
{
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
- (void)viewWillAppear:(BOOL)animated
{
[super viewWillAppear:animated];
Next we have to build this method.


Simon Says.xcodeproj — SSViewController.m
Simon Says › iPad 5.0 Simulator
Finished running Simon Says on iPad 5.0 Simulator
No Issues
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
Simon Says › Simon Says › SSViewController.m › @implementation SSViewController
Simon Says
1 target, iOS SDK 5.0
Simon Says
SSAppDelegate.h
SSAppDelegate.m
SSViewController.h
SSViewController.m
StarView.h
StarView.m
SSViewController.xib
Supporting Files
Frameworks
Products
// instantiate the label
command = [[UILabel alloc] initWithFrame:theFrame];
command.backgroundColor = [UIColor blackColor];
command.textColor = [UIColor whiteColor];
command.font = [UIFont fontWithName:@"Ari
command.text = [self makeCommand];
// add it as a subview
[self.view addSubview:command];
}
-(NSString*) makeCommand
{
NSMutableString* commandString = [NSMutableString stringWithFormat:@""];
simonSays = (arc4random()%2)==1;
if (simonSays)
[commandString appendString:@"Simon says touch the "];
else
[commandString appendString:@"Touch the "];
currentObject = (arc4random()%4);
// for now the only object is a star
currentObject = STAR;
[commandString appendString:@"star."];
return commandString;
}
-(void) touchesBegan: (NSSet*) touches withEvent: (UIEven
{
NSLog(@"You touched the view!");
}
- (void)viewDidUnload
{
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
makeCommand randomly decides whether or not simon says and chooses a random object.
We'll hardcode this to be a star for the moment, since that is the only view we've created.


Simon Says.xcodeproj — SSViewController.m
Run
Stop
Simon Says › iPad 5.0 Simulator
Scheme
Breakpoints
Running Simon Says on iPad 5.0 Simulator
No Issues
Editor
View
Organizer
Simon Says
1 target, iOS SDK 5.0
Simon Says
SSAppDelegate.h
SSAppDelegate.m
SSViewController.h
SSViewController.m
StarView.h
StarView.m
SSViewController.xib
Supporting Files
Frameworks
Products
Simon Says › Simon Says › SSViewController.m › -touchesBegan:withEvent:
{
NSMutableString* commandString = [NSMutableString stringWithFormat:@""];
simonSays = (arc4random()%2)==1;
if (simonSays)
[commandString appendString:@"Simon says touch the "];
else
[commandString appendString:@"Touch the "];
currentObject = (arc4random()%4);
// for now the only object is a star
currentObject = STAR;
[commandString appendString:@"star."];
return commandString;
}
-(void) touchesBegan: (NSSet*) touches withEvent: (UIEvent*) event
{
UITouch* t = [touches anyObject];
bool success=false;
if (currentObject == STAR)
{
if (([t.view class] == [StarView class] && simonSays) || ([t.view class] != [StarView class] && !simonSays))
success = true;
}
if (success)
command.text = [self makeCommand];
else {
// if they are wrong we'll prepend a sorry you're wrong message and have them try again
NSMutableString* fail = [NSMutableString stringWithFormat:@"Sorry that is wrong. "];
[fail appendString:command.text];
command.text=fail;
}
}
- (void)viewDidUnload
Simon Says
Change touchesBegan to test player's response to the command.


Simon Says.xcodeproj — SSViewController.m
Run
Stop
Simon Says › iPad 5.0 Simulator
Scheme
Breakpoints
Running Simon Says on iPad 5.0 Simulator
No Issues
Editor
View
Organizer
Simon Says
1 target, iOS SDK 5.0
Products
UITouch object t holds a pointer to the view that was touched.
Objective-c supports "introspection. An object can tell us what kind of class it is at runtime. This test checks if the view touched was our StarView object.
-touchesBegan:withEvent:
UITouch* t = [touches anyObject];
bool success=false;
if (currentObject == STAR)
if (([t.view class] == [StarView class] && simonSays) || ([t.view class] != [StarView class] && !simonSays))
success = true;
if (success)
command.text = [self makeCommand];
else {
// if they are wrong we'll prepend a sorry you're wrong message and have them try again
NSMutableString* fail = [NSMutableString stringWithFormat:@"Sorry that is wrong. "];
[fail appendString:command.text];
command.text=fail;
}
- (void)viewDidUnload
Simon Says

It is up to you to finish Simon Says. Add the remaining views, and fix makeCommand and touchesBegan to handle them.