Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

- Demo

  Polymorphism with Controllers in Matchismo

  How to change the class of a Controller in a storyboard

- Multiple MVCs in an Application

  `UINavigationController`

  `UITabBarController`

- Demo

  Attributor Stats

Demo

- Making a Generic Controller in Matchismo
  Polymorphism with Controllers in Matchismo
  Get rid of `PlayingCardDeck` in `CardGameViewController`.
  How to change the class of a Controller in a storyboard

Multiple MVCs

- Why?

  When your application gets more features than can fit in one MVC.

- How to add a new MVC to your storyboard

  Drag "View Controller" from Object Palette.

  Create a subclass of `UIViewController` using New File menu item.

  Set that subclass as the class of your new Controller in the Attributes Inspector.

- How to present this new MVC to the user

  `UINavigationController`

  `UITabBarController`

  Other mechanisms we'll talk about later in the course (popover, modal, etc.).

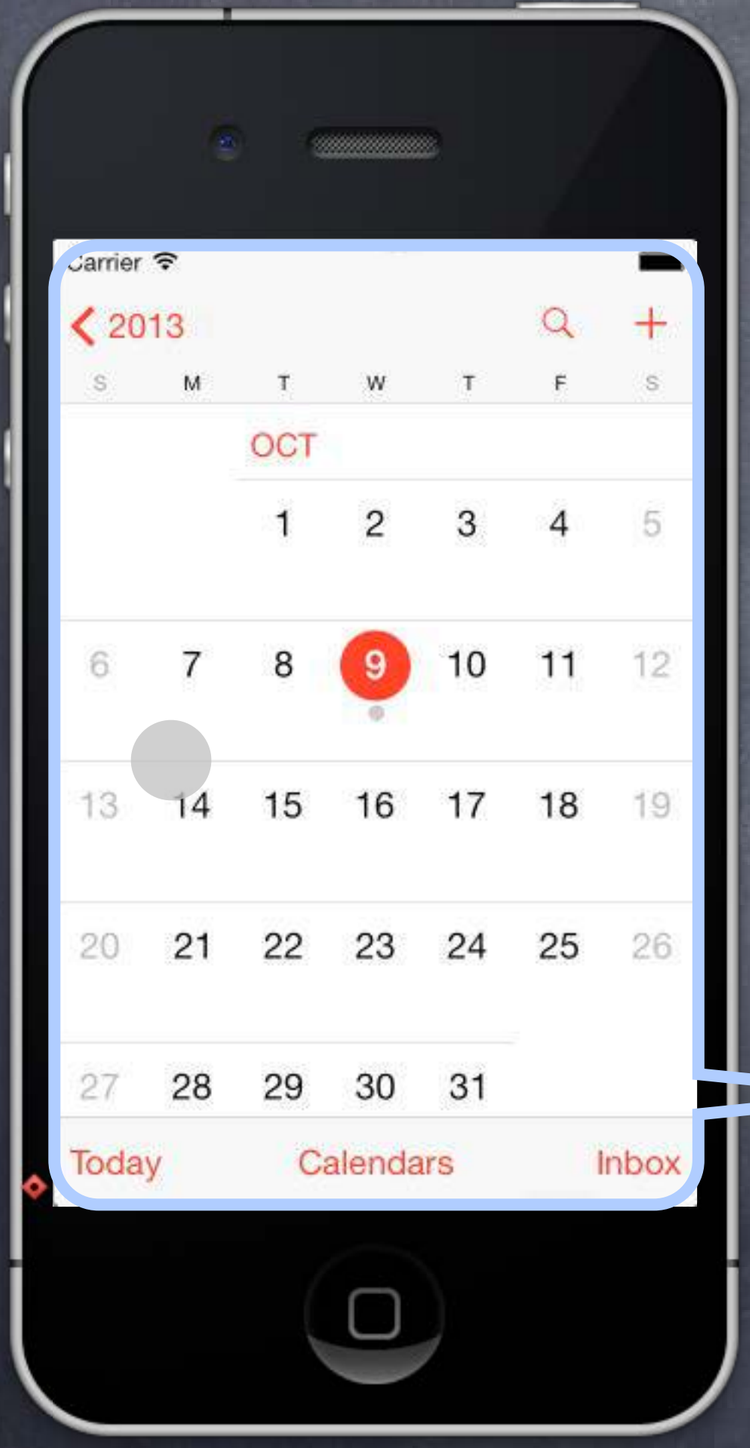
UINavigationController

- When to use it?

  When the user wants to "dive down" into more detail.

UINavigationController

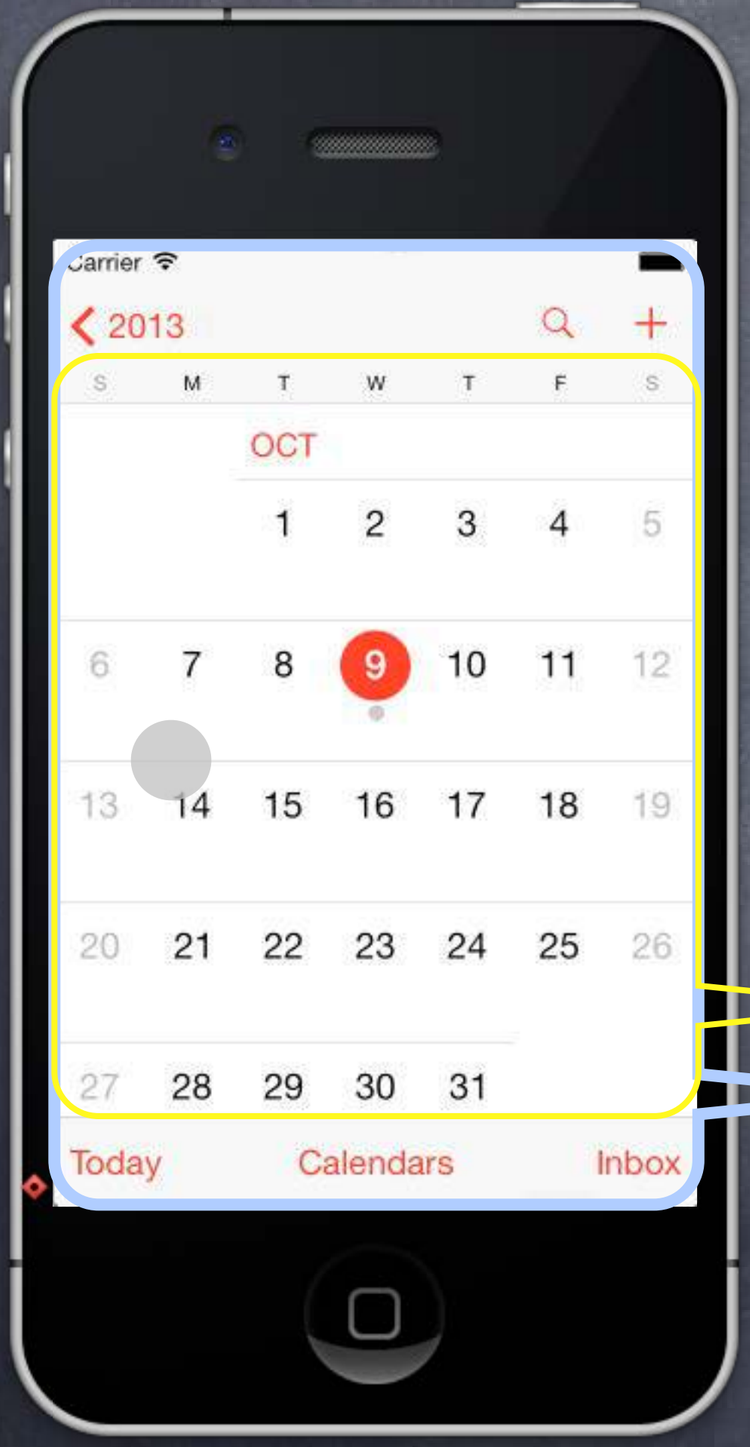
- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.



This is the `UINavigationController`'s View.

UINavigationController

- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.

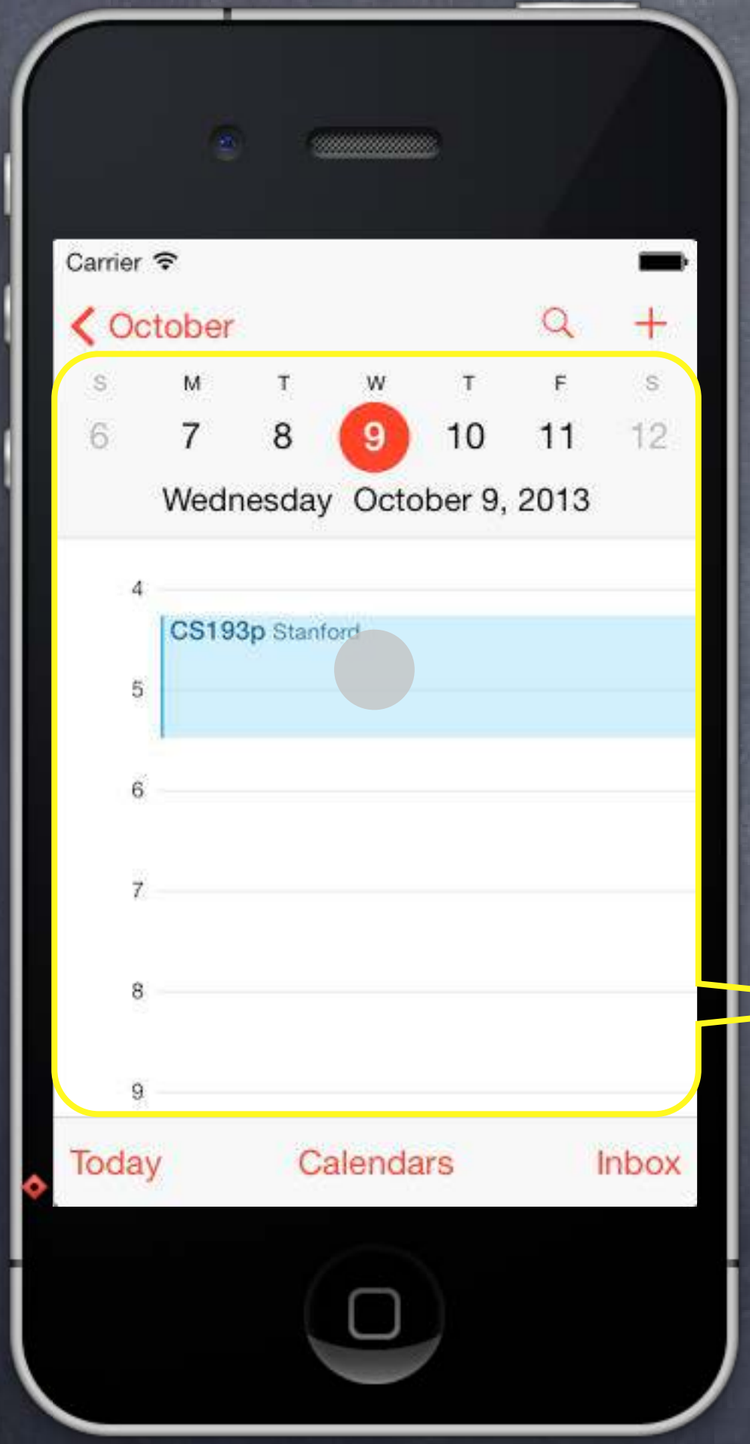


This is a Month MVC's View.

This is the UINavigationController's View.

UINavigationController

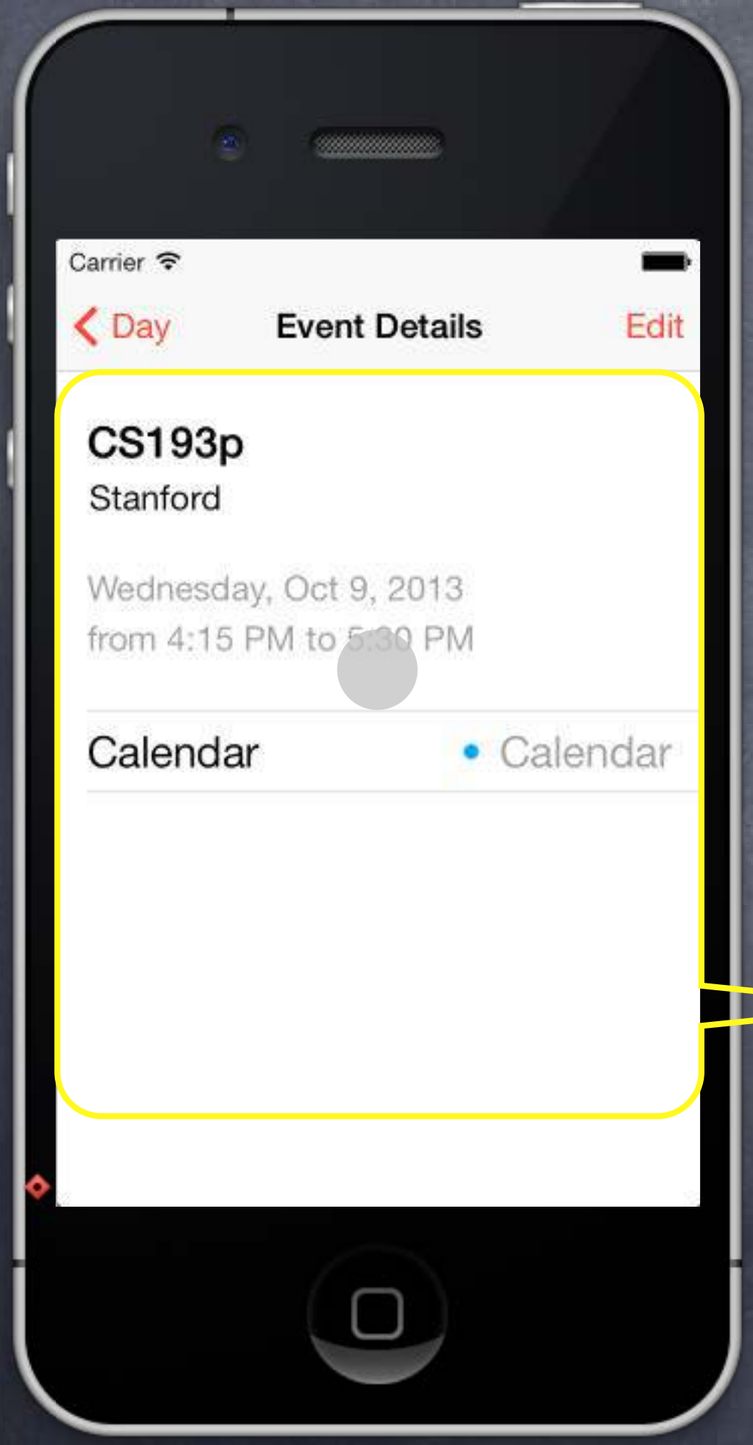
- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.



This is a Day MVC's View.

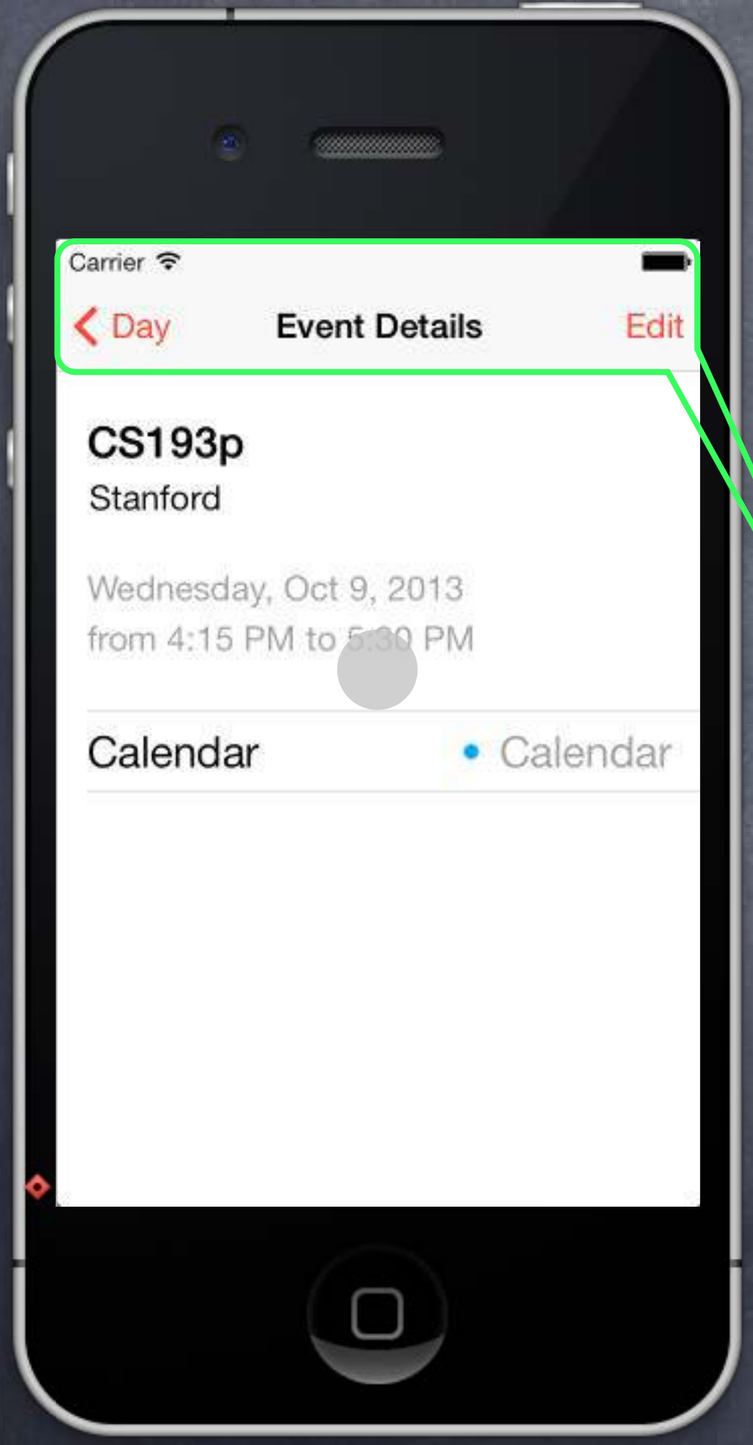
UINavigationController

- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.

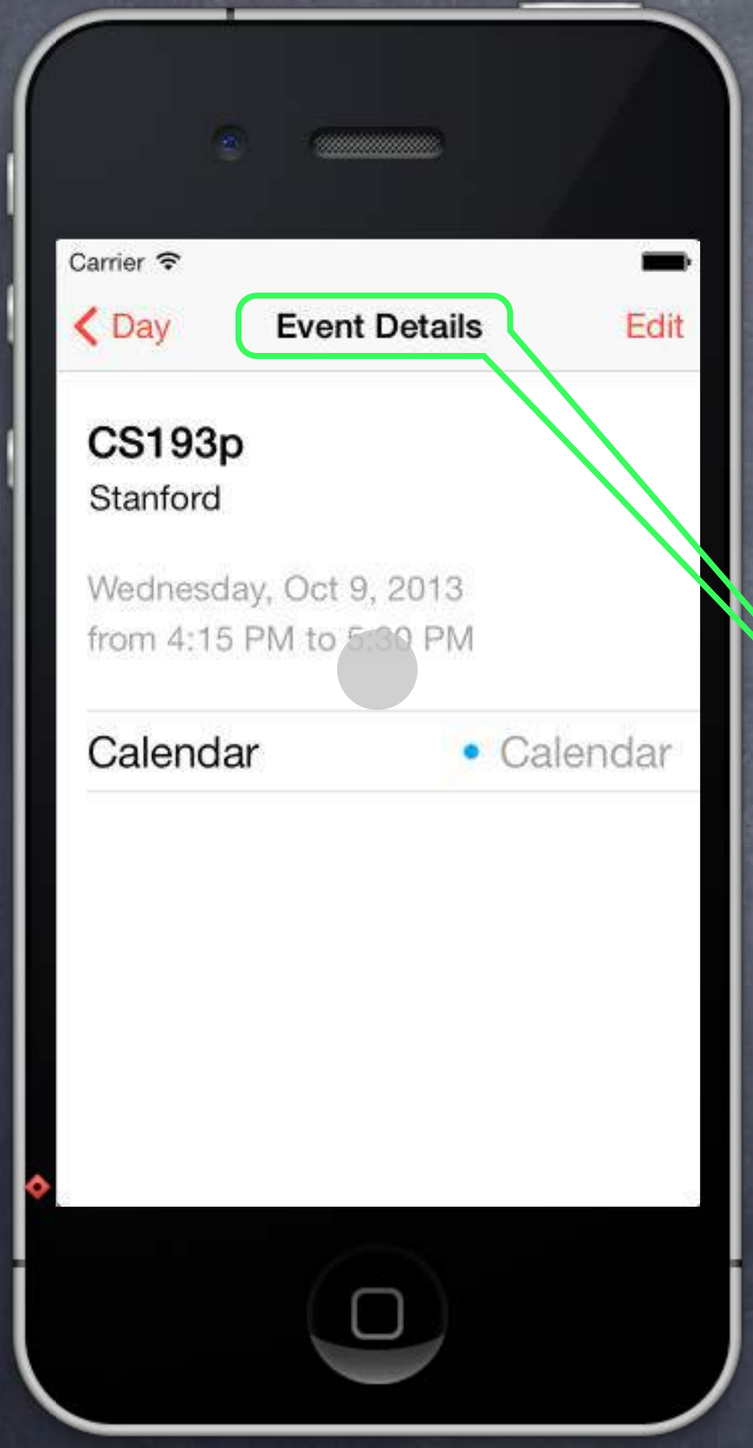


This is a Calendar Event MVC's View.

UINavigationController

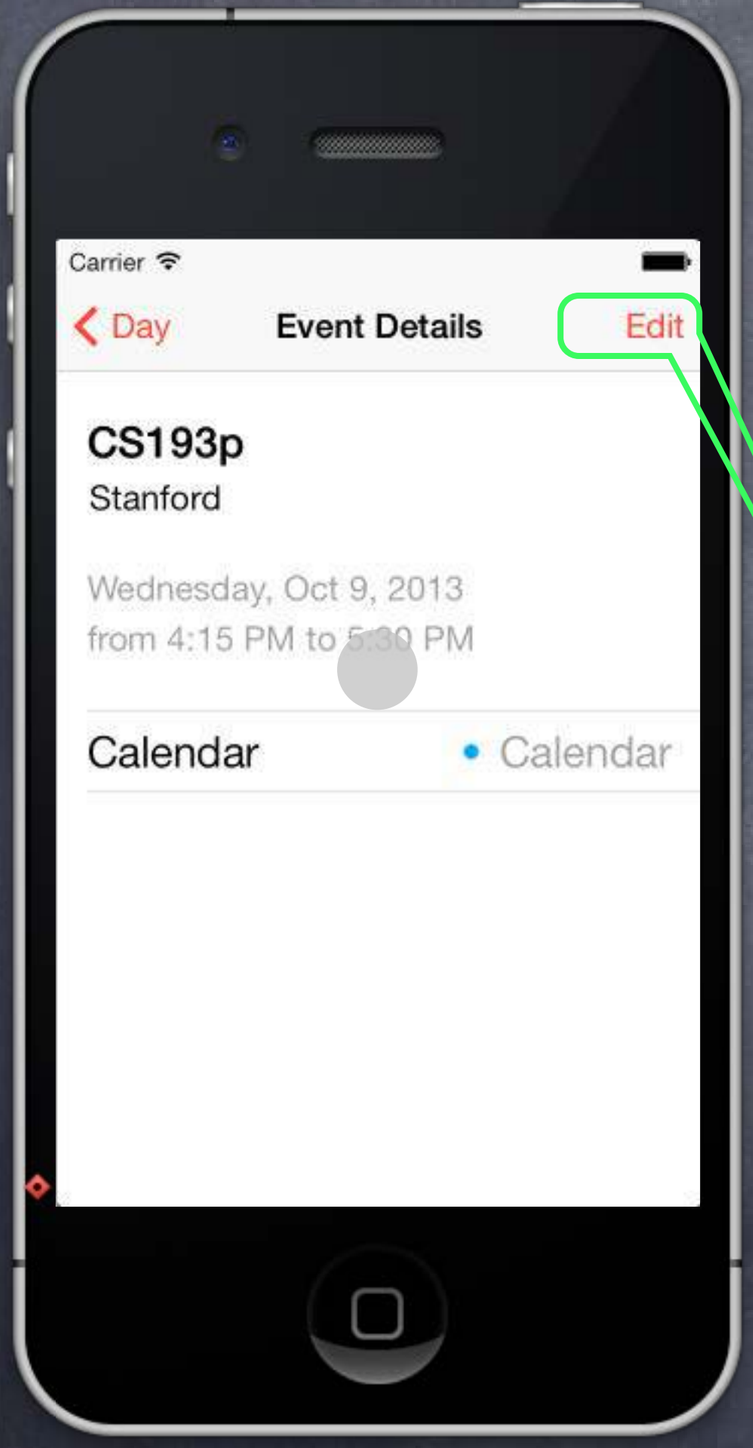


- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.
- Components of a UINavigationController
  Navigation Bar (contents determined by embedded MVC's `navigationItem`).



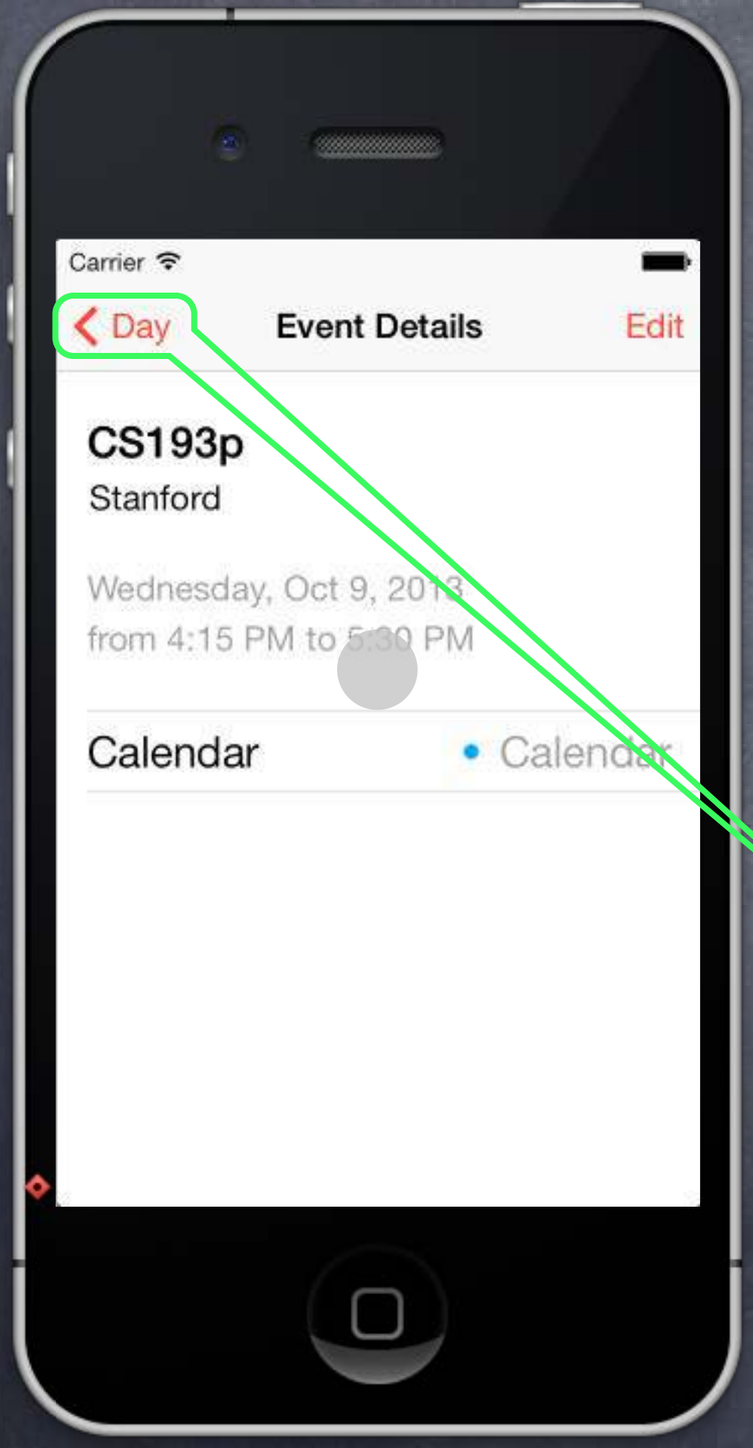
UINavigationController

- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.
- Components of a UINavigationController
  Navigation Bar (contents determined by embedded MVC's `navigationItem`).
    Title (by default is `title` property of the embedded MVC)



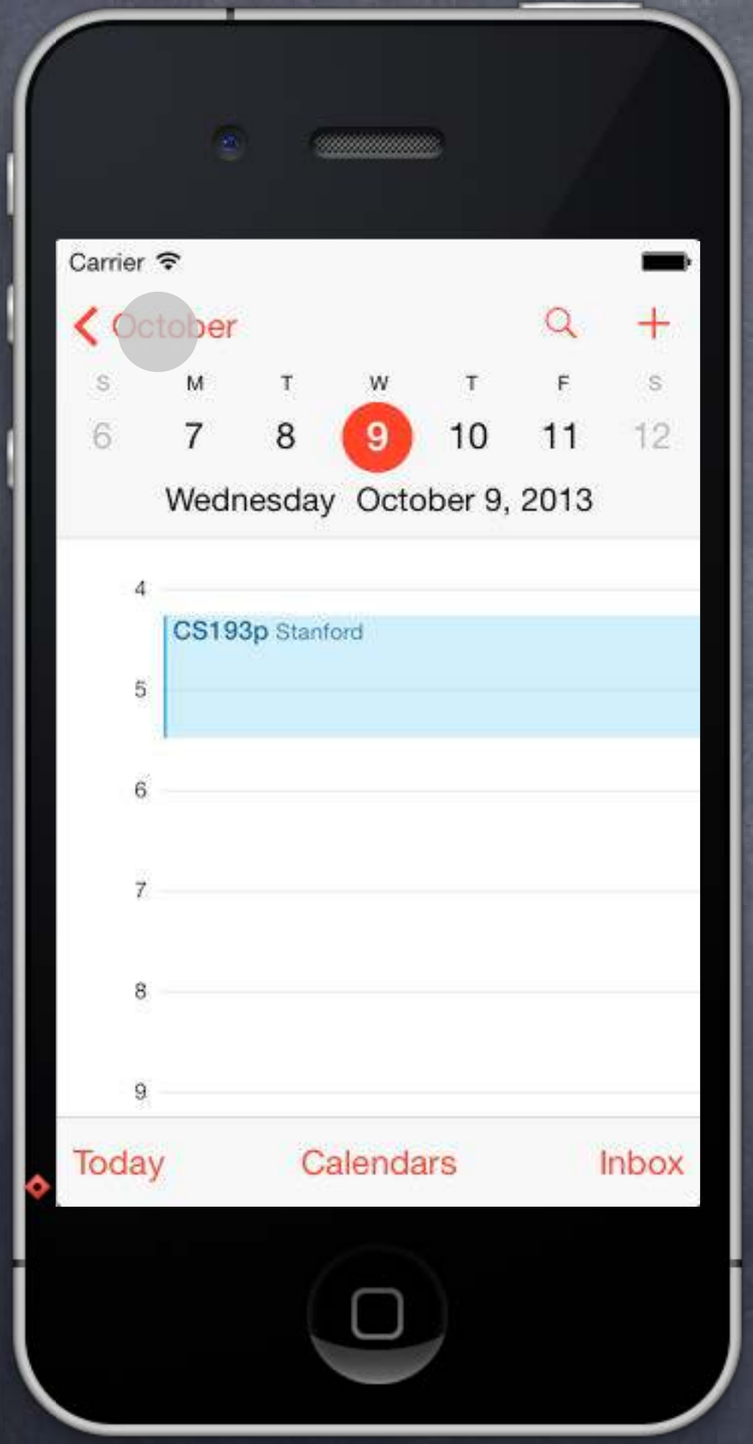
UINavigationController

- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.
- Components of a UINavigationController
  Navigation Bar (contents determined by embedded MVC's `navigationItem`).
    Title (by default is `title` property of the embedded MVC)
    Embedded MVC's `navigationItem.rightBarButtonItems`
      (an NSArray of UIBarButtonItems)



UINavigationController

- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.
- Components of a UINavigationController
  Navigation Bar (contents determined by embedded MVC's `navigationItem`).
    Title (by default is `title` property of the embedded MVC)
    Embedded MVC's `navigationItem.rightBarButtonItems`
      (an `NSArray` of `UIBarButtonItems`)
    Back Button (automatic)



UINavigationController

- When to use it?

  When the user wants to "dive down" into more detail.

- How does it work?

  Encloses other MVCs (like the Year MVC and the Month MVC).

  Touches in one MVC "segue" to the other MVCs.

- Components of a UINavigationController

  Navigation Bar (contents determined by embedded MVC's `navigationItem`).

  Title (by default is `title` property of the embedded MVC)

  Embedded MVC's `navigationItem.rightBarButtonItems`

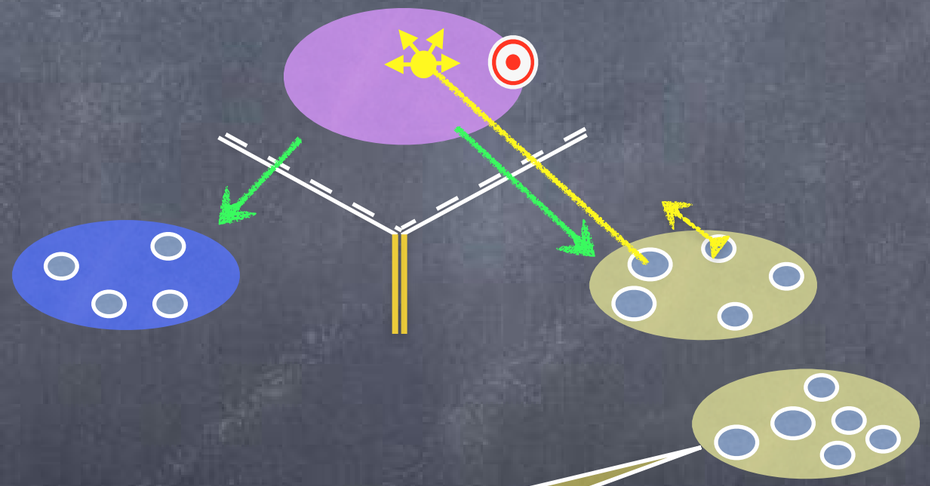
  (an `NSArray` of `UIBarButtonItems`)

  Back Button (automatic)

UINavigationController

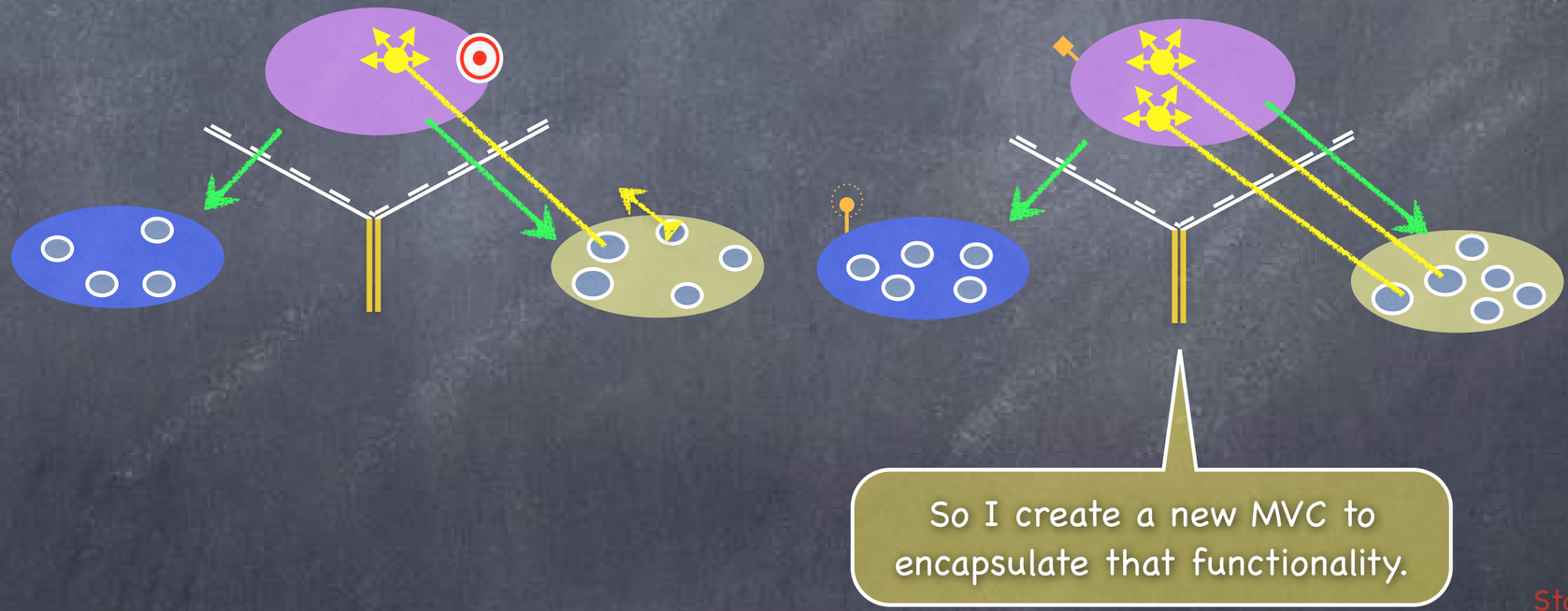
- When to use it?
  When the user wants to "dive down" into more detail.
- How does it work?
  Encloses other MVCs (like the Year MVC and the Month MVC).
  Touches in one MVC "segue" to the other MVCs.
- Components of a UINavigationController
  Navigation Bar (contents determined by embedded MVC's `navigationItem`).
    Title (by default is `title` property of the embedded MVC)
    Embedded MVC's `navigationItem.rightBarButtonItems`
      (an NSArray of UIBarButtonItems)
    Back Button (automatic)
  Embedded MVC's `toolbarItems` property
    (also an NSArray of UIBarButtonItems)

MVCs working together



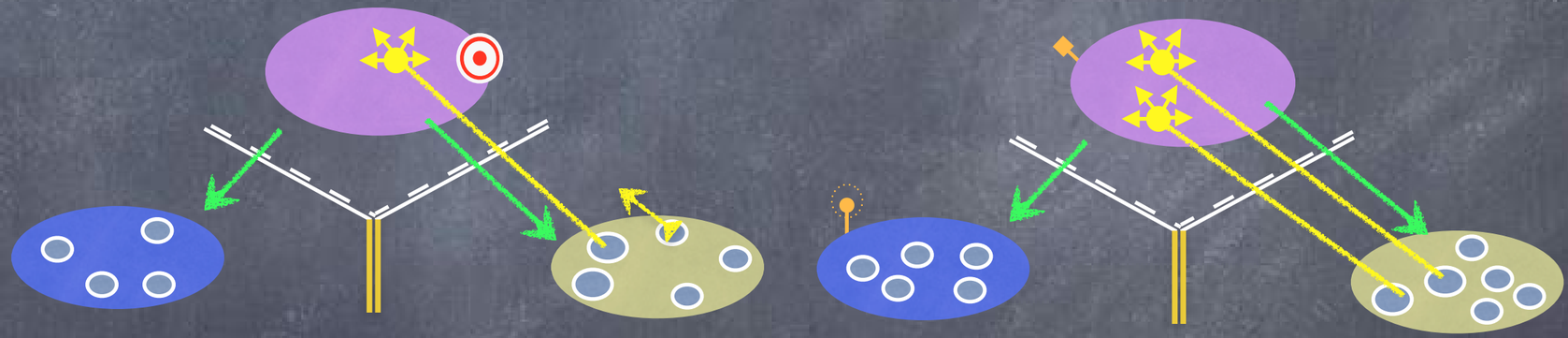
I want more features, but it doesn't make sense to put them all in one MVC!

MVCs working together

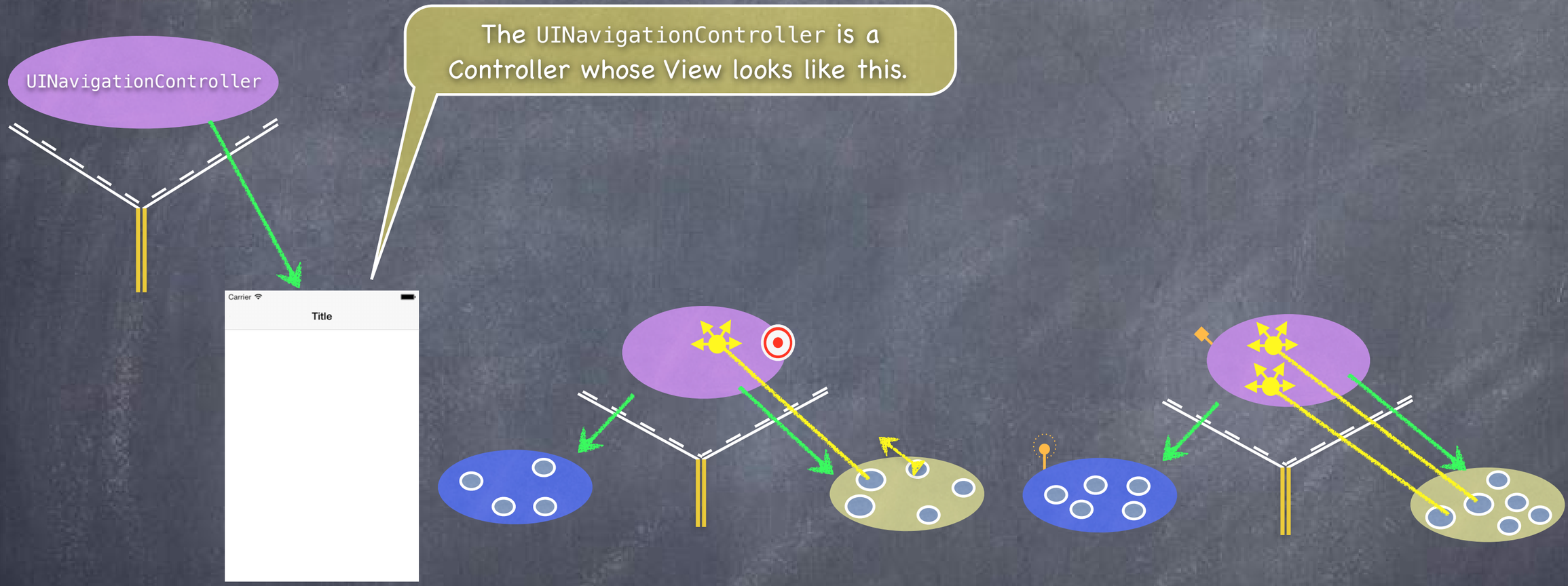


MVCs working together

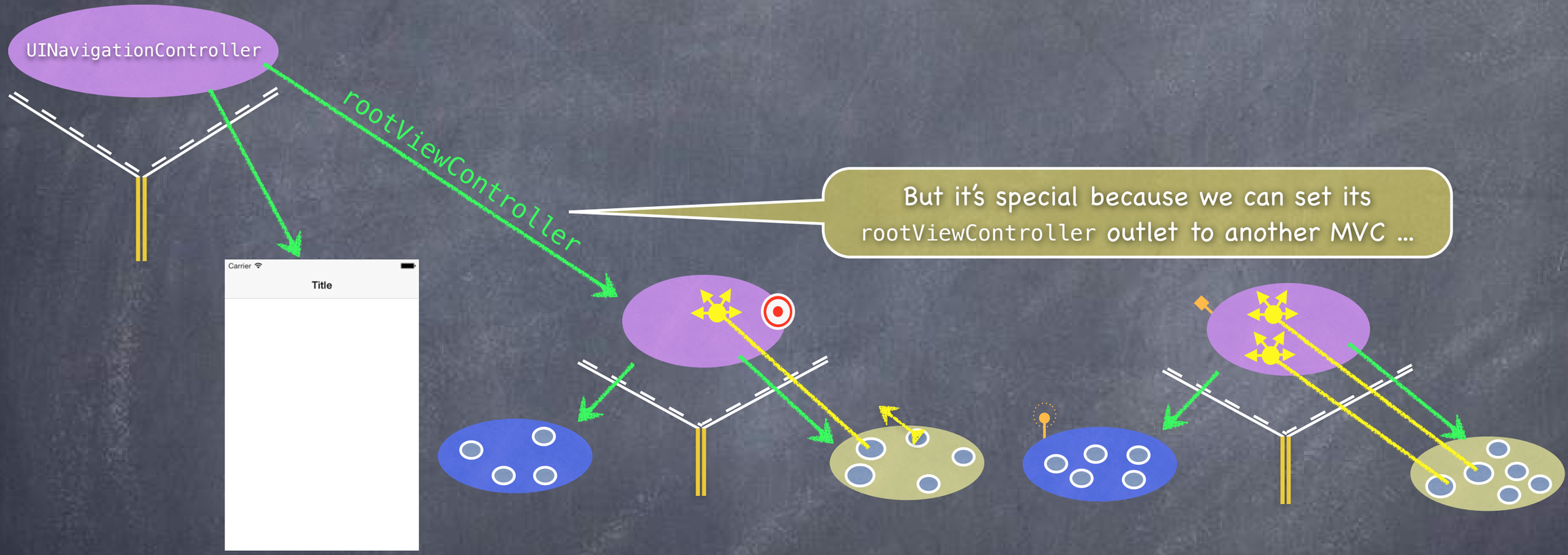
If the relationship between these two MVCs is "more detail," we use a `UINavigationController` to let them share the screen.



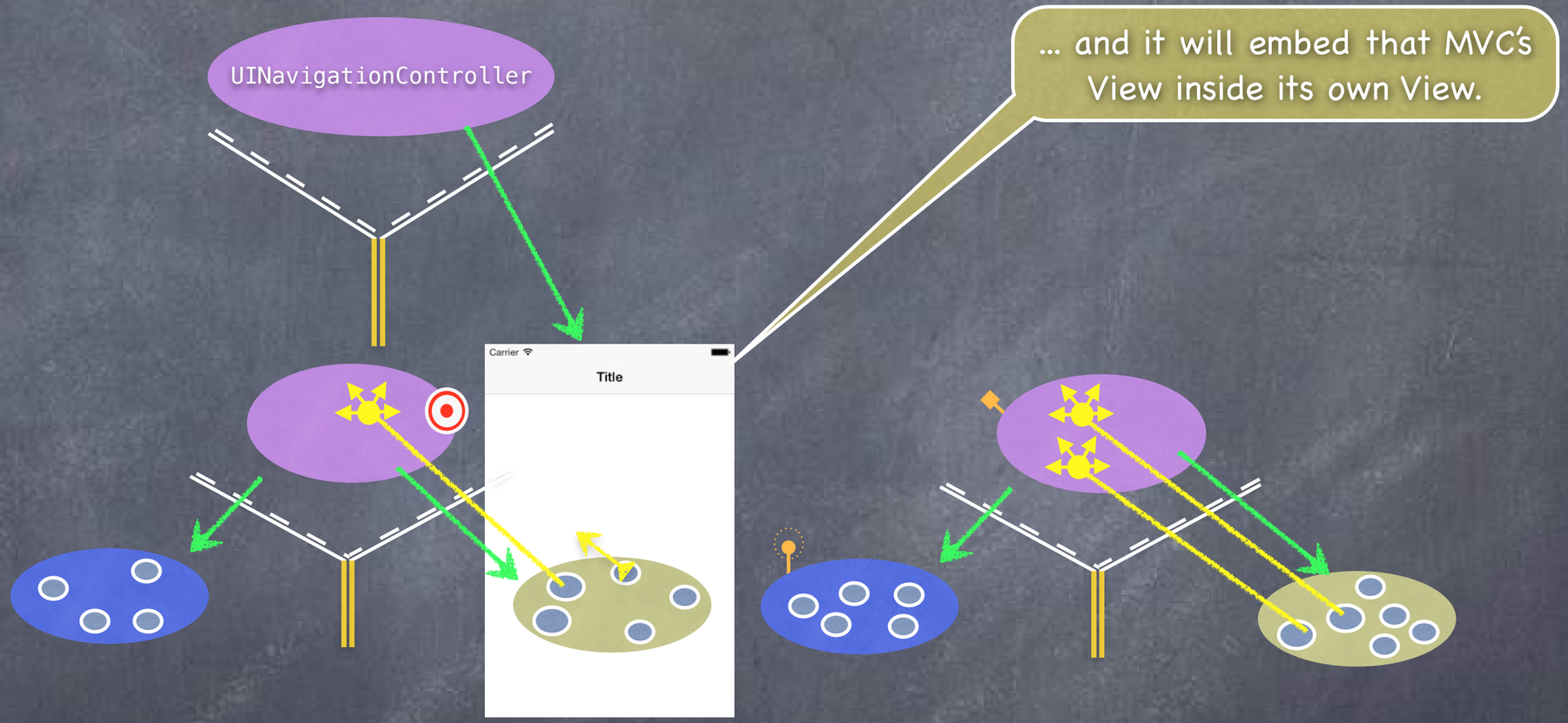
MVCs working together



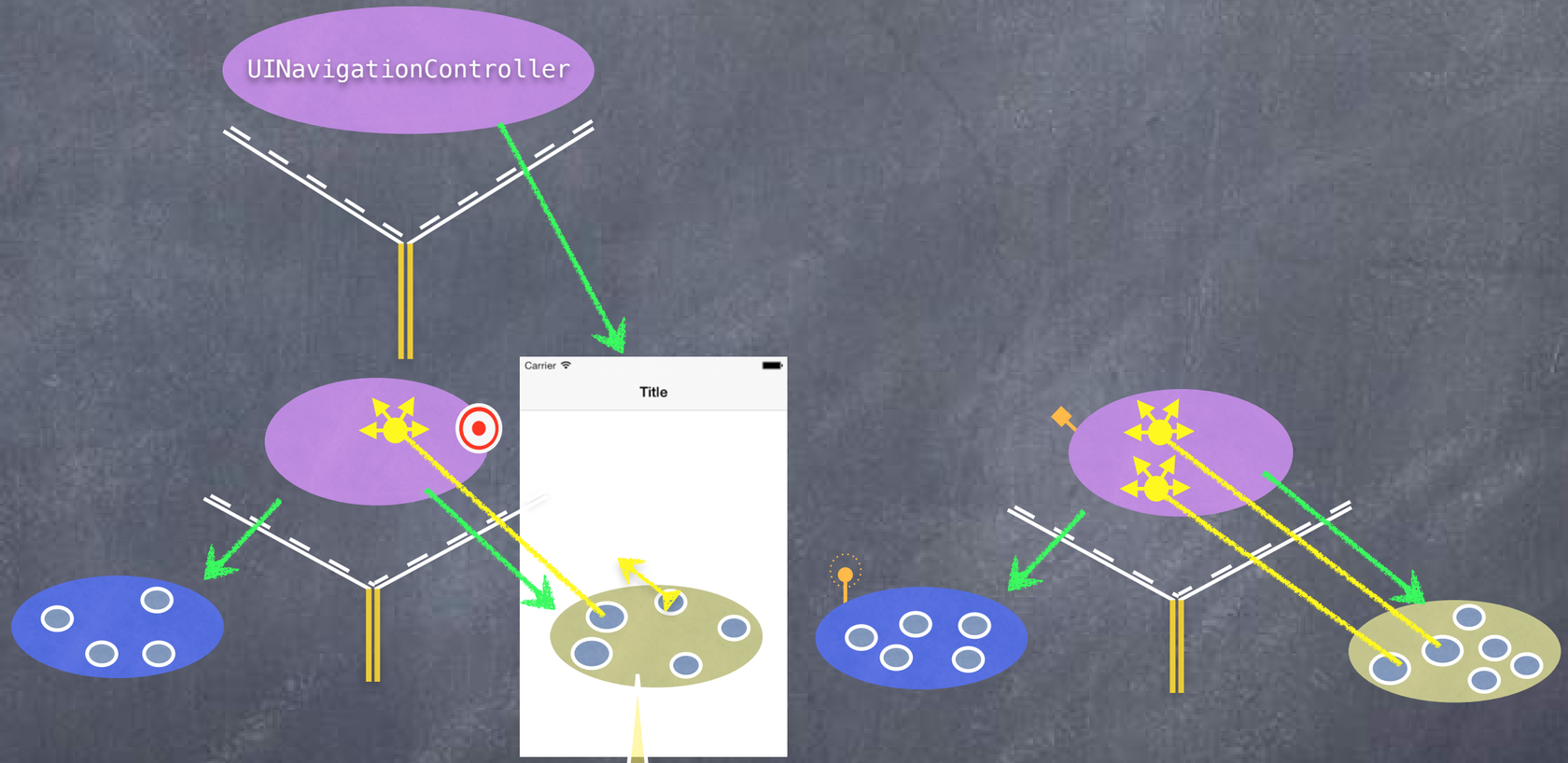
MVCs working together



MVCs working together

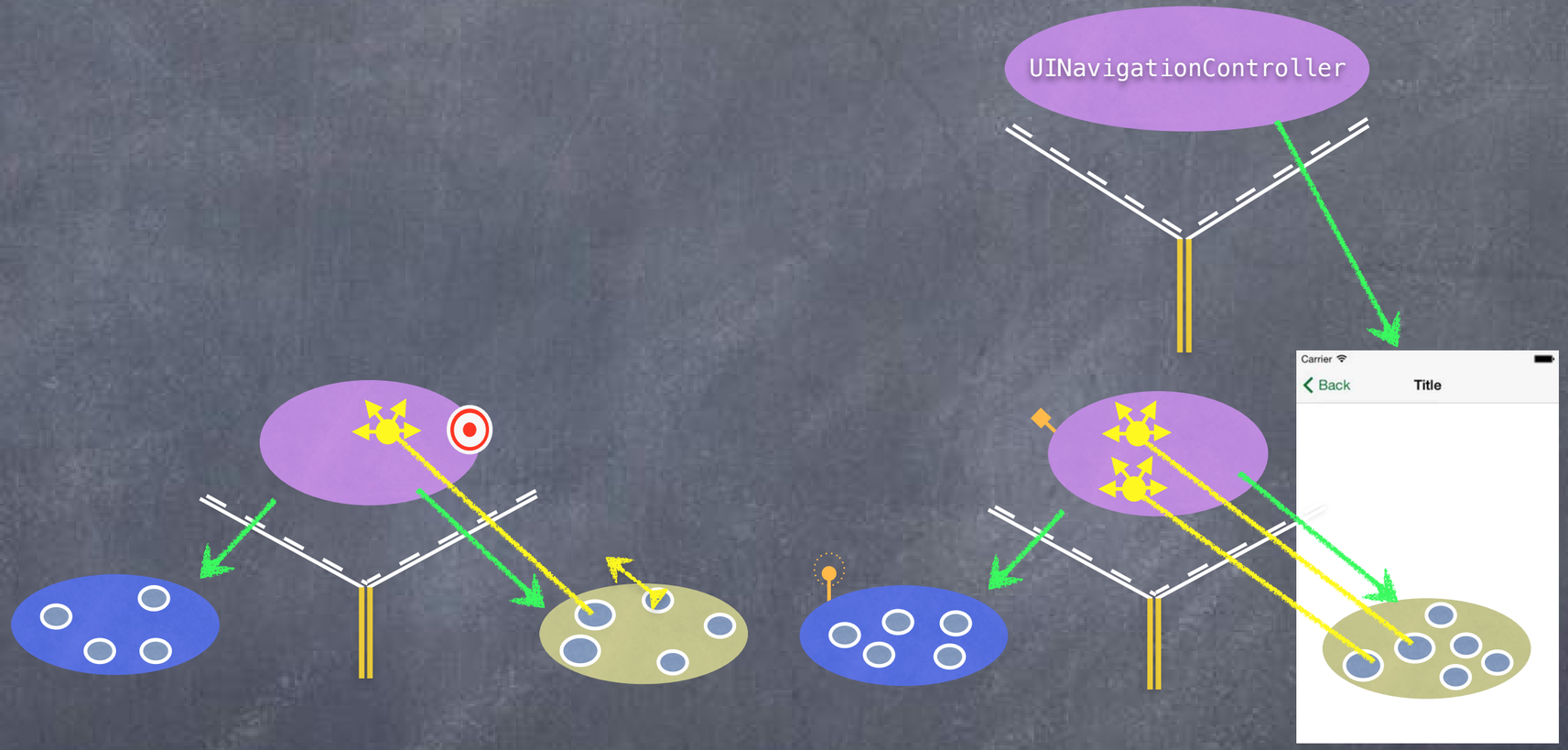


MVCs working together



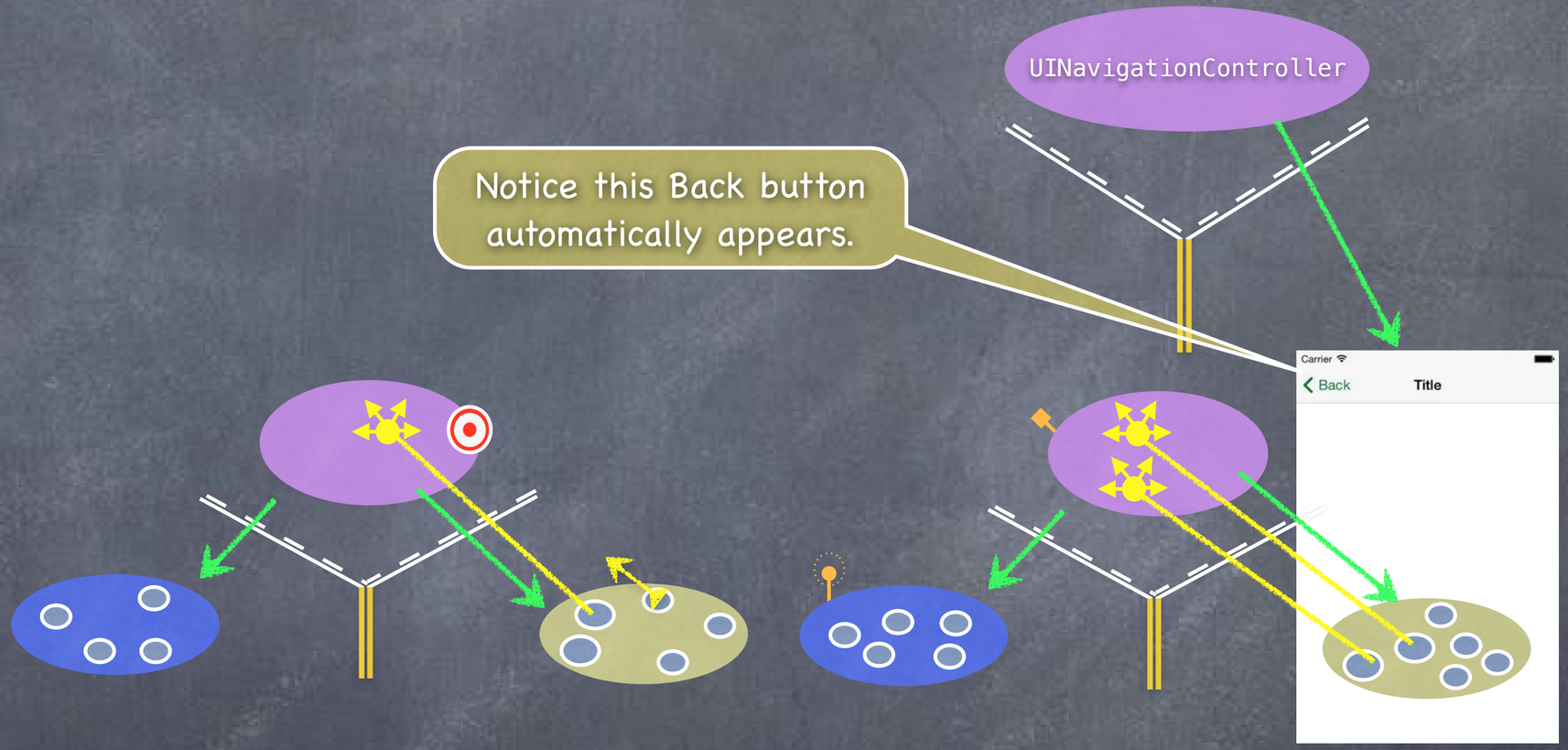
Then a UI element in this View (e.g. a `UIButton`) can segue to the other MVC and its View will now appear in the `UINavigationController` instead.

MVCs working together

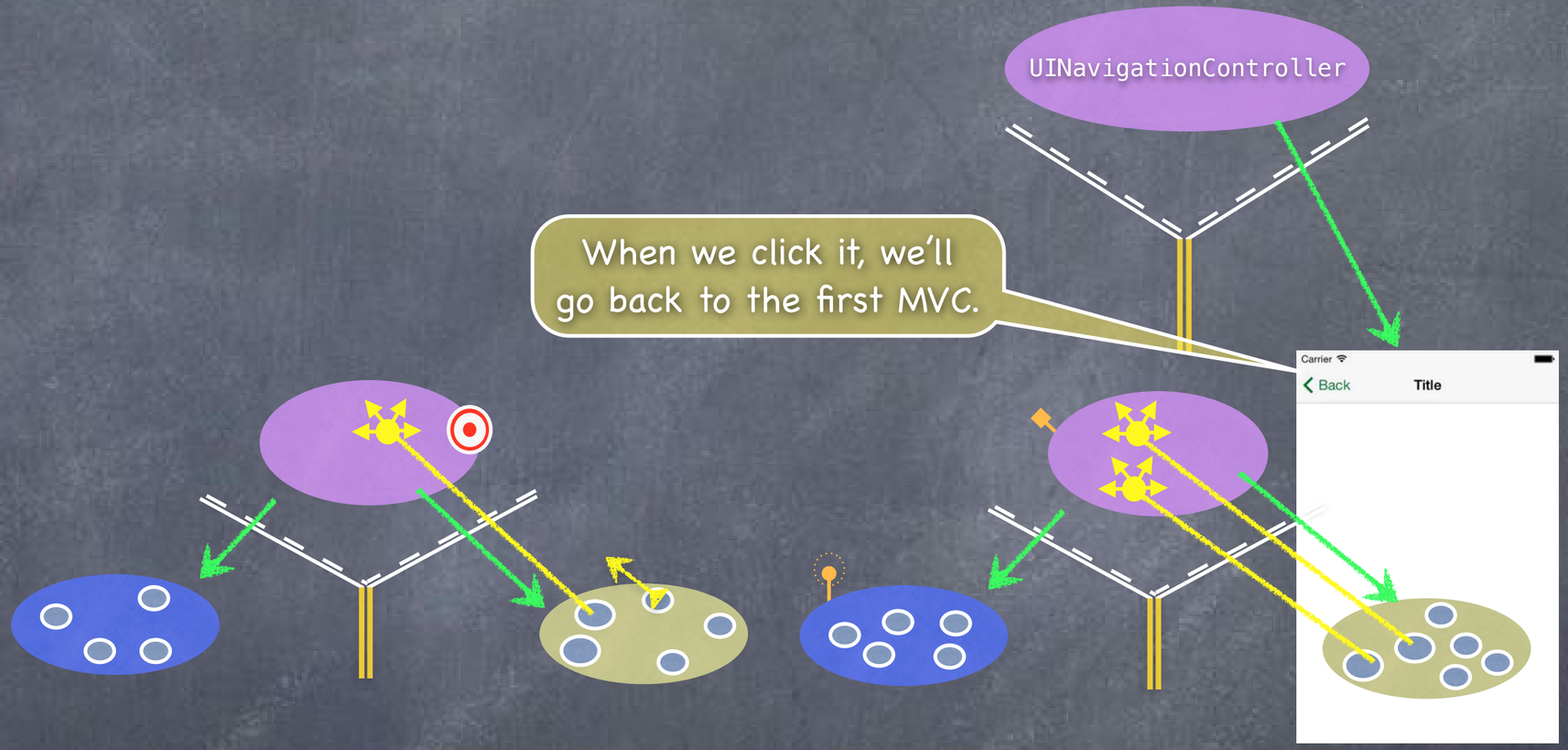


We call this kind of segue a "push segue".

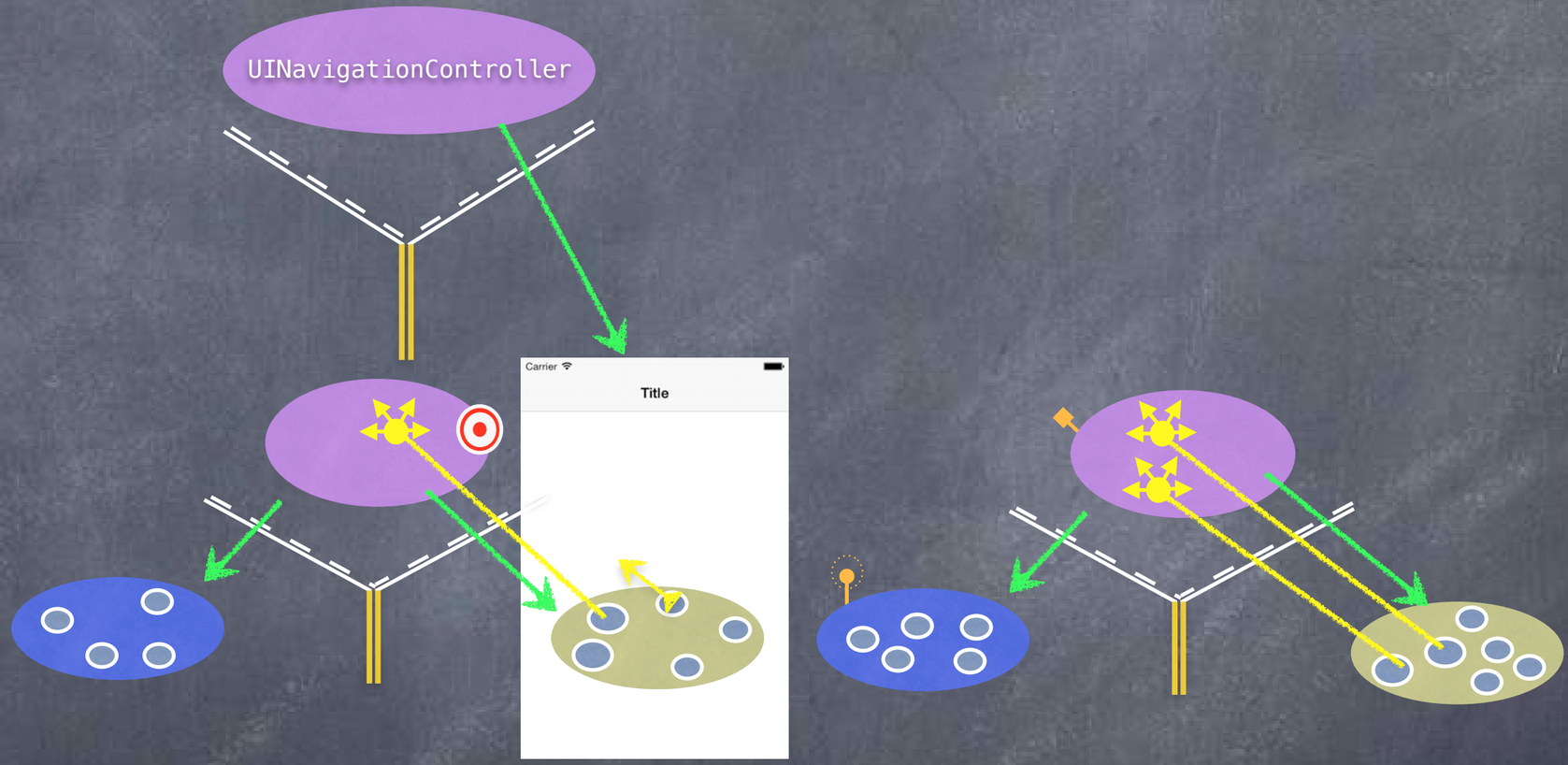
MVCs working together



MVCs working together

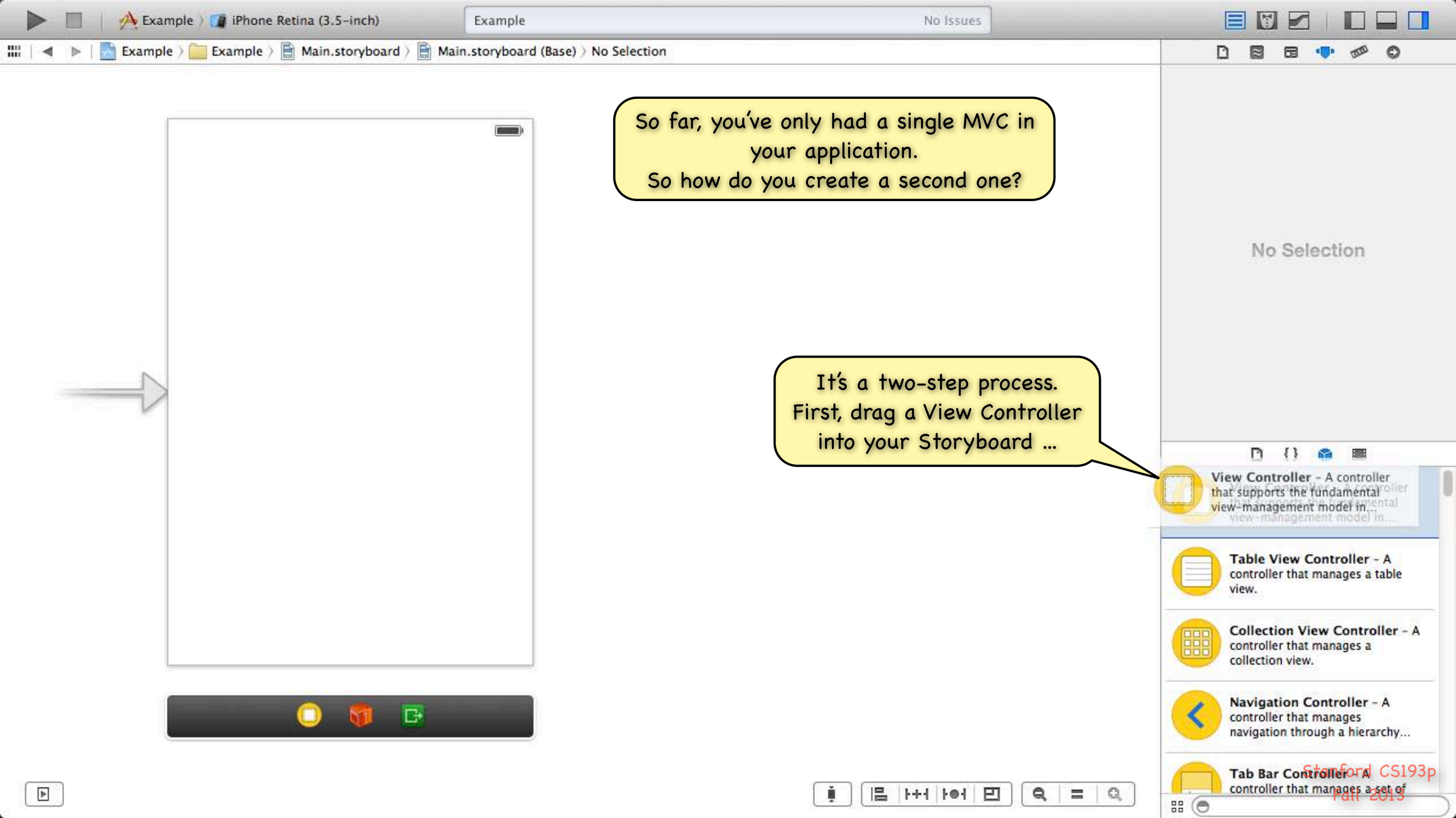


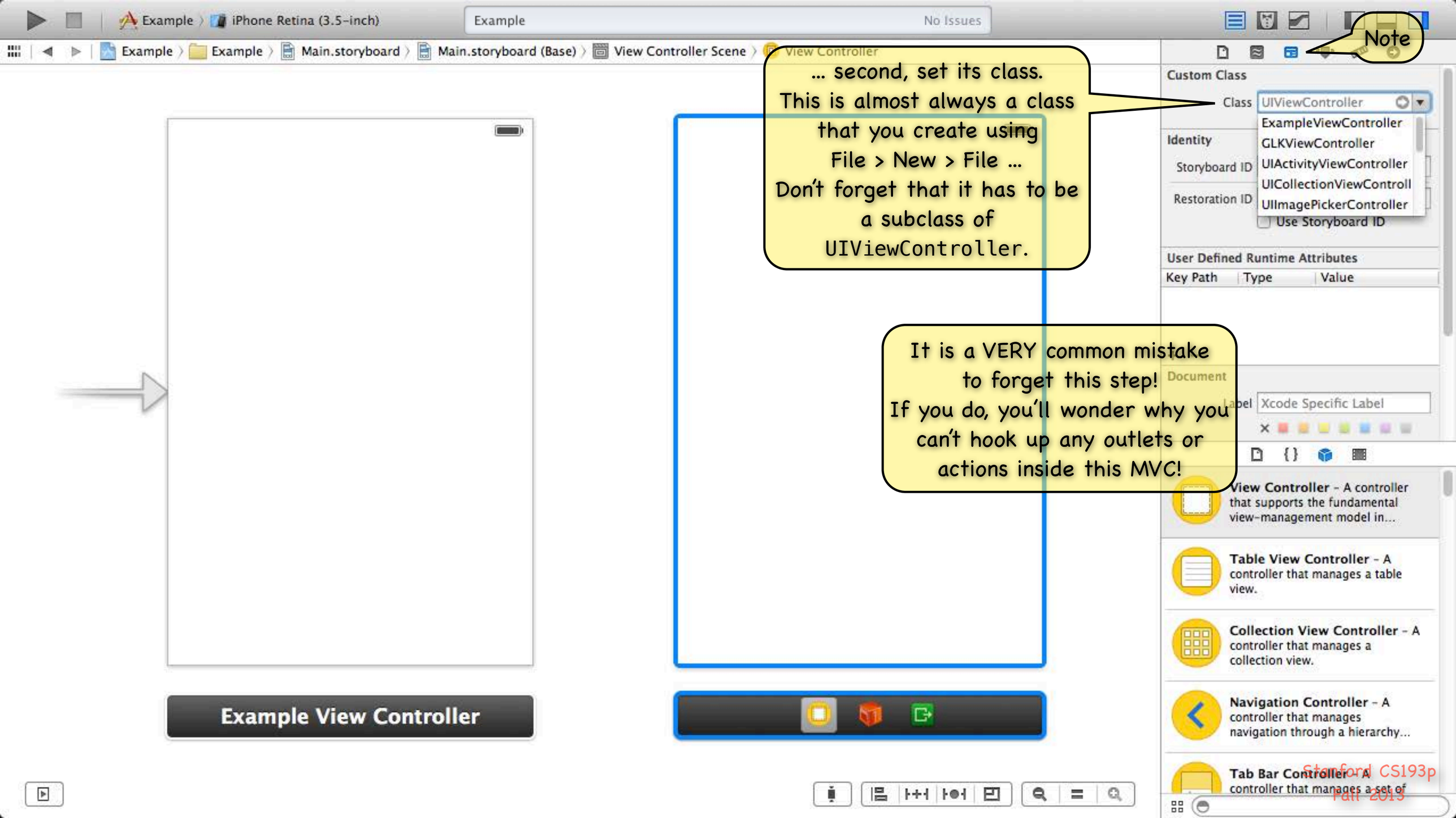
MVCs working together

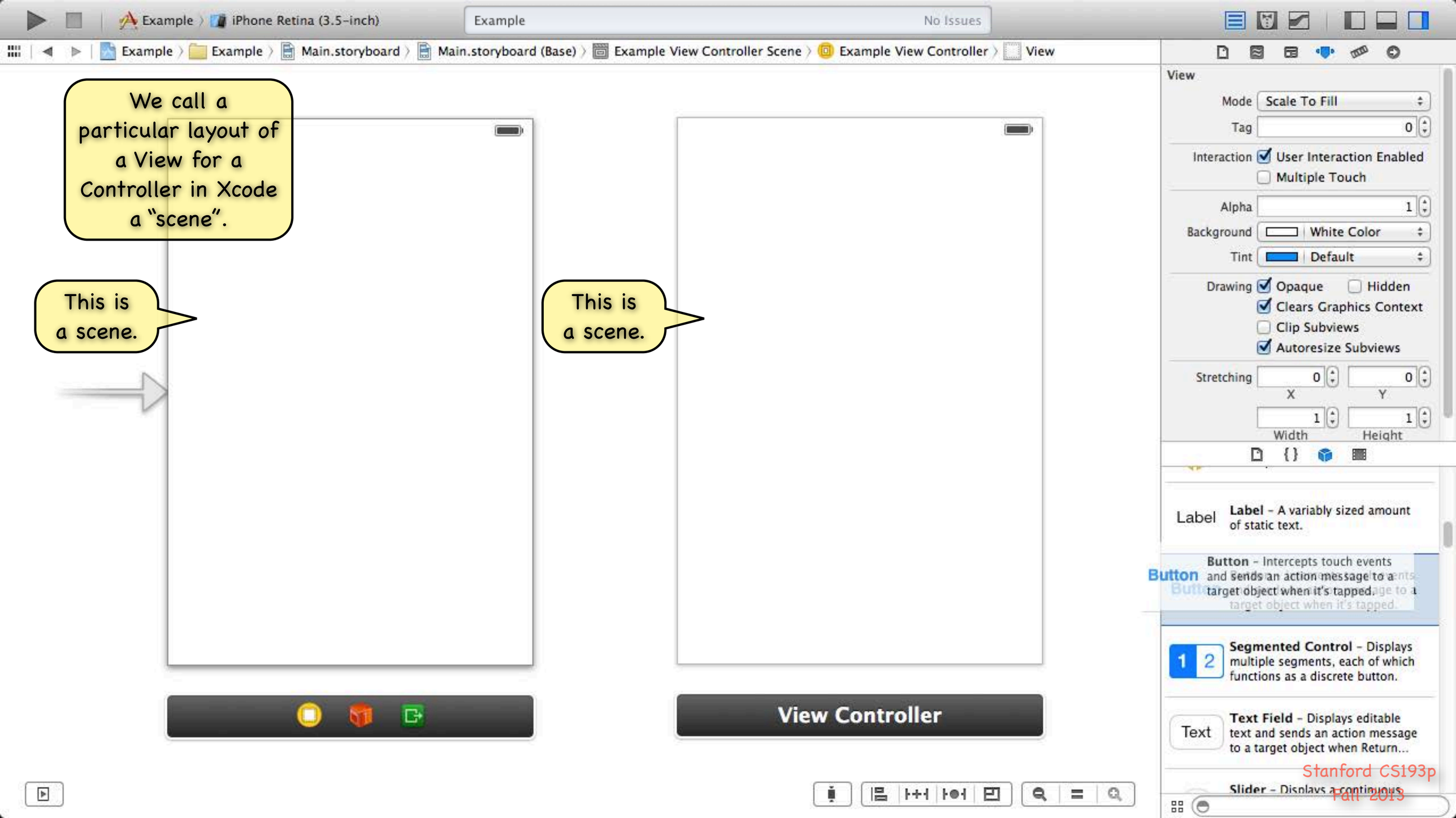


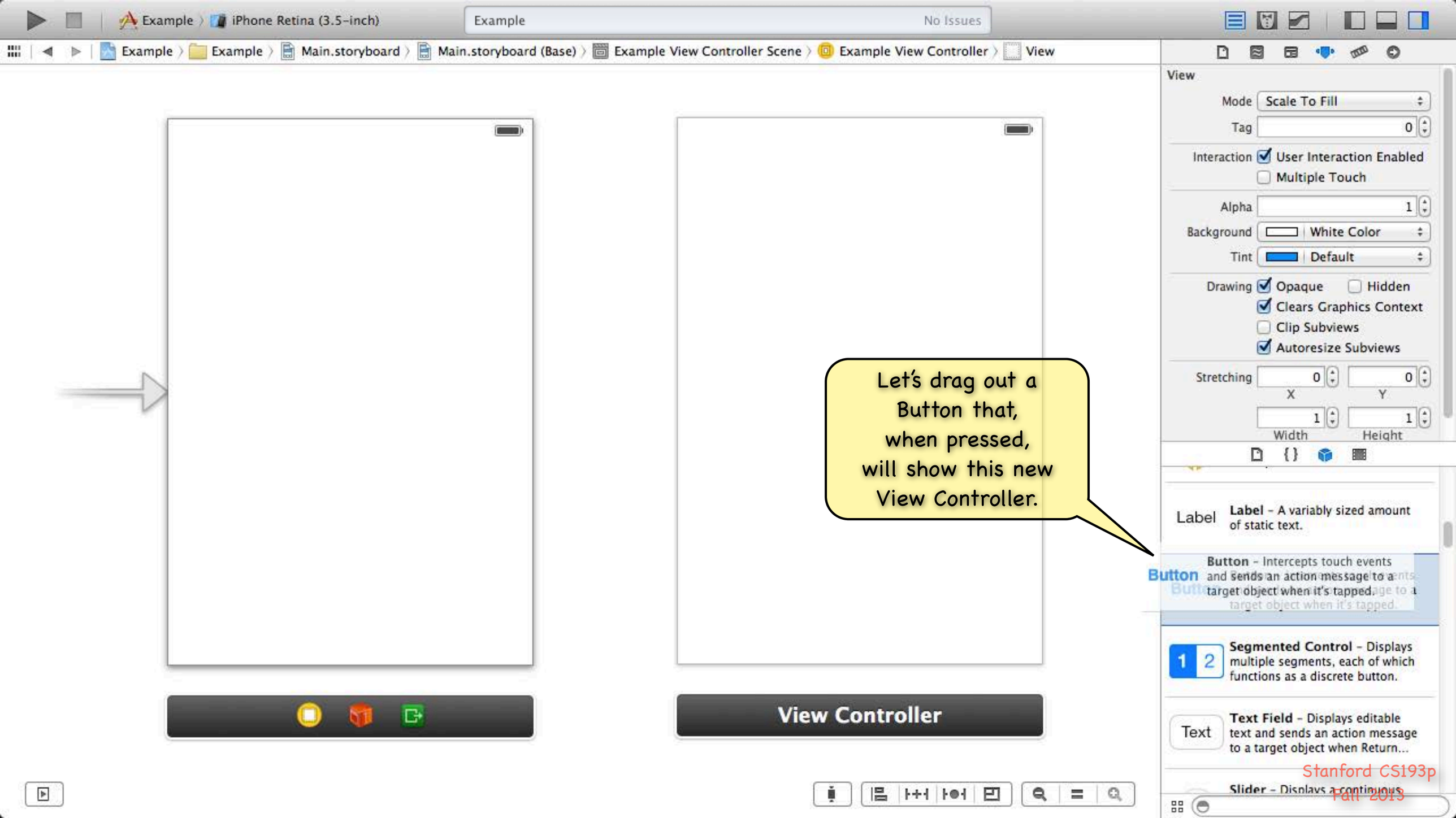
Segues

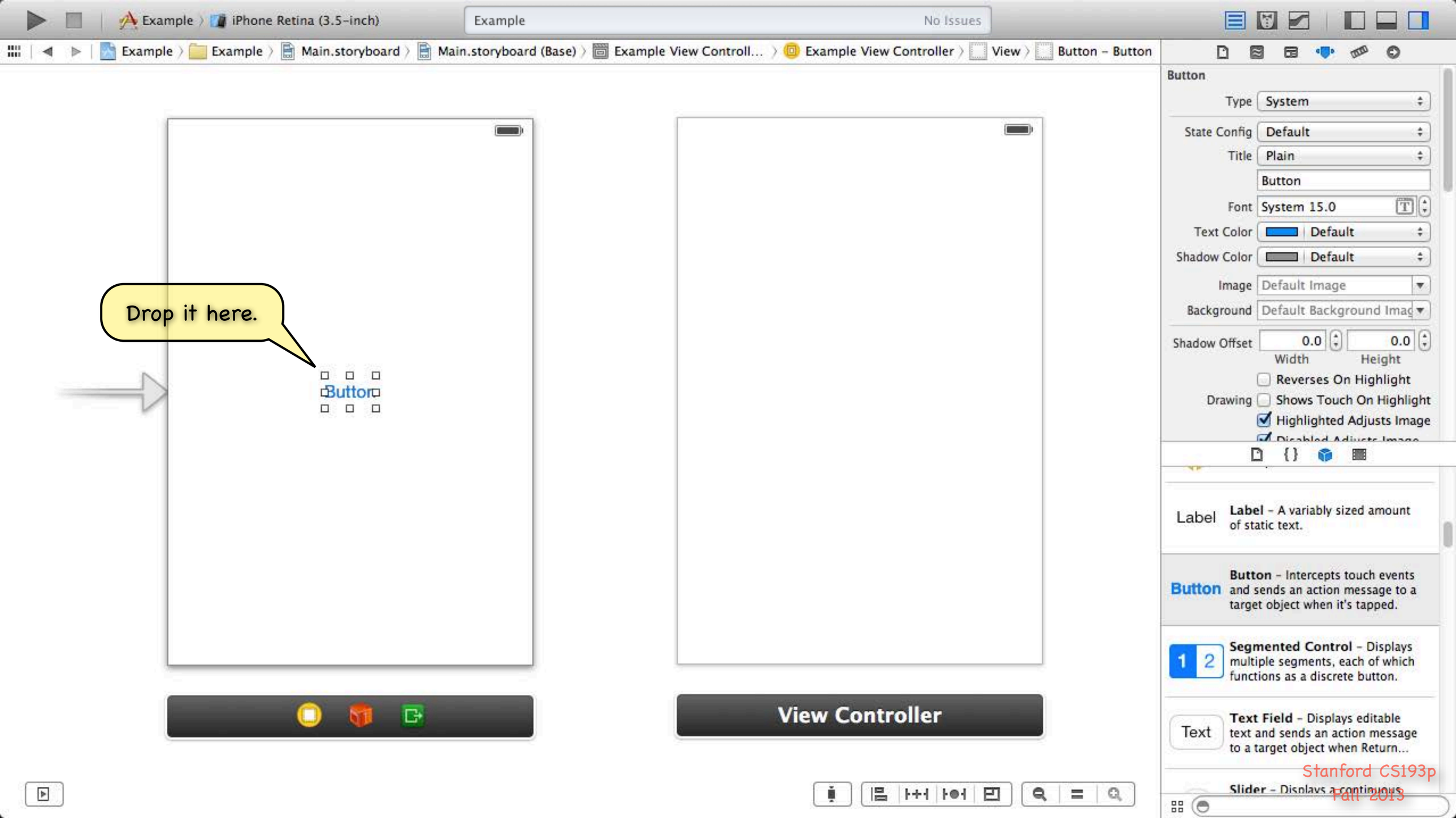
- Let's talk about how the segue gets set up first
  Then we'll look at how we create a `UINavigationController` in our storyboard.

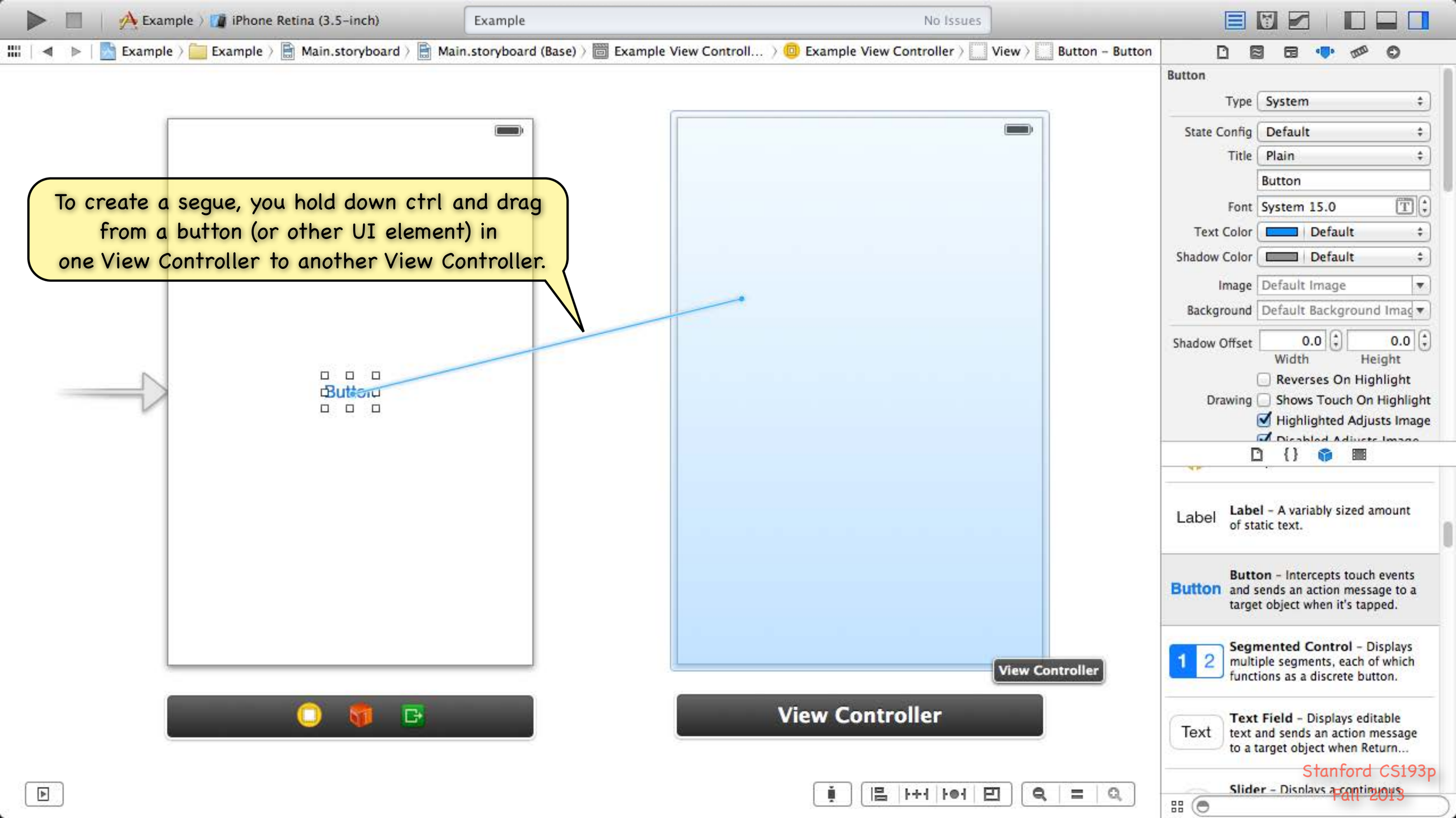

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
No Selection
So far, you've only had a single MVC in your application.
So how do you create a second one?
No Selection
It's a two-step process.
First, drag a View Controller into your Storyboard ...
View Controller – A controller that supports the fundamental view-management model in...
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy...
Tab Bar Controller – A controller that manages a set of
Stanford CS193p
Fall 2013

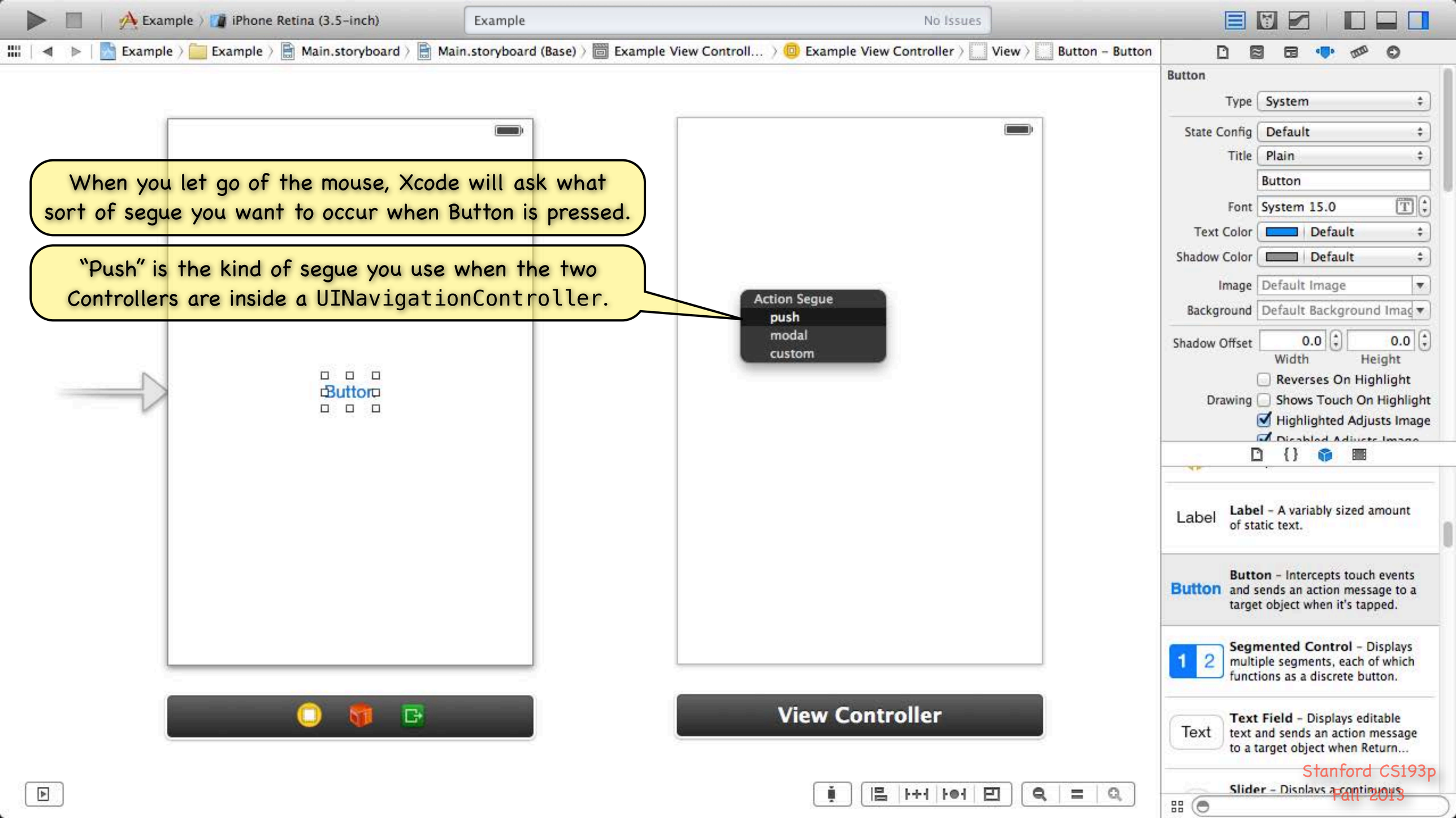

Example
iPhone Retina (3.5-inch)
No Issues
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Note
... second, set its class.
This is almost always a class that you create using File > New > File ...
Don't forget that it has to be a subclass of UIViewController.
Custom Class
Class
UIViewController
ExampleViewController
GLKViewController
UIActivityViewController
UICollectionViewControll
UIImagePickerController
Identity
Storyboard ID
Restoration ID
Use Storyboard ID
User Defined Runtime Attributes
Key Path
Type
Value
It is a VERY common mistake to forget this step!
If you do, you'll wonder why you can't hook up any outlets or actions inside this MVC!
Document
Label
Xcode Specific Label
View Controller – A controller that supports the fundamental view-management model in...
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy...
Tab Bar Controller – A controller that manages a set of
Example View Controller
Stanford CS193p
Fall 2013

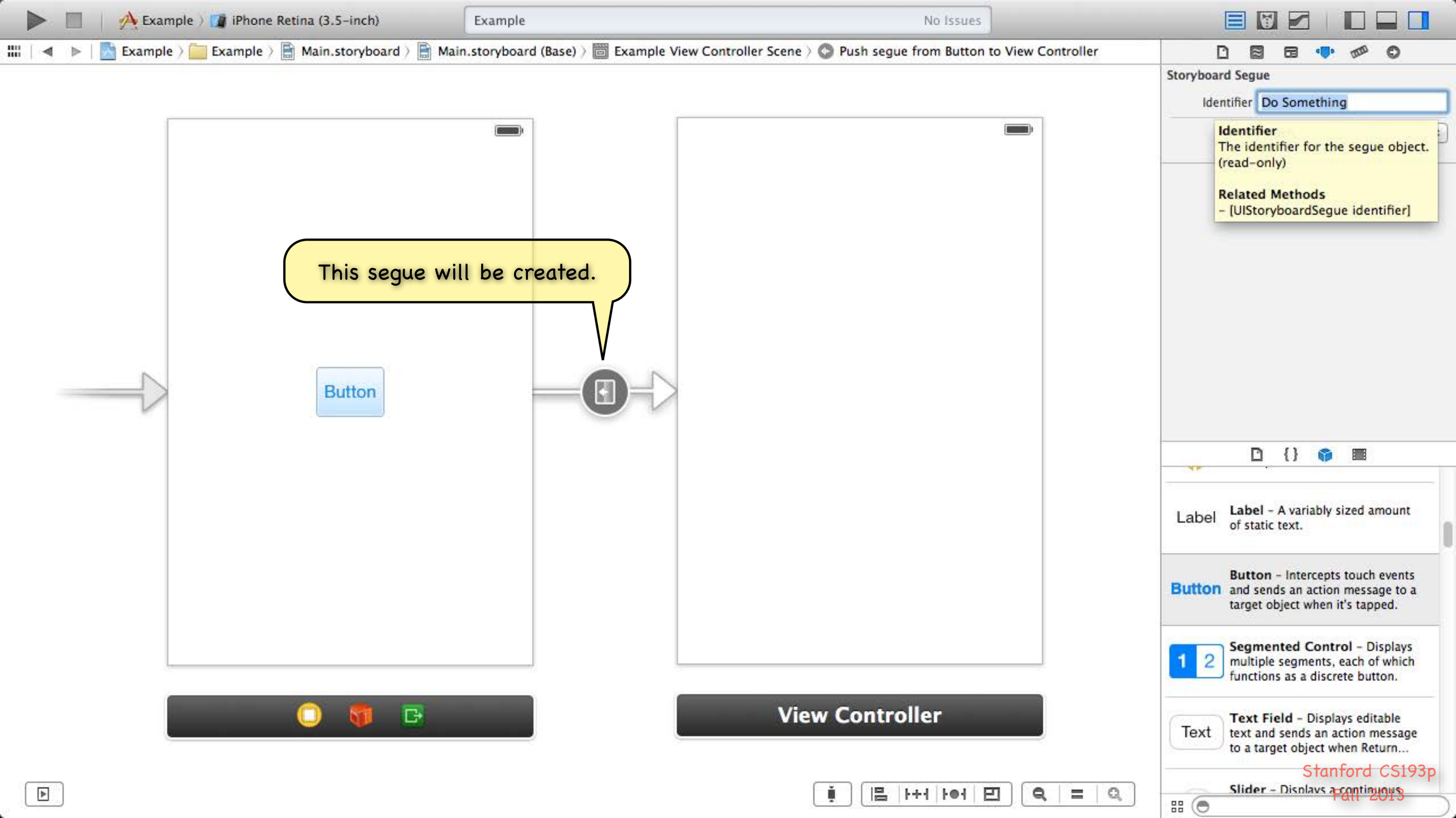

Example › iPhone Retina (3.5-inch)
Example
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › Example View Controller Scene › Example View Controller › View
We call a particular layout of a View for a Controller in Xcode a "scene".
This is a scene.
This is a scene.
View Controller
View
Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background White Color
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching 0 X 0 Y
1 Width 1 Height
Label Label – A variably sized amount of static text.
Button Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2 Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

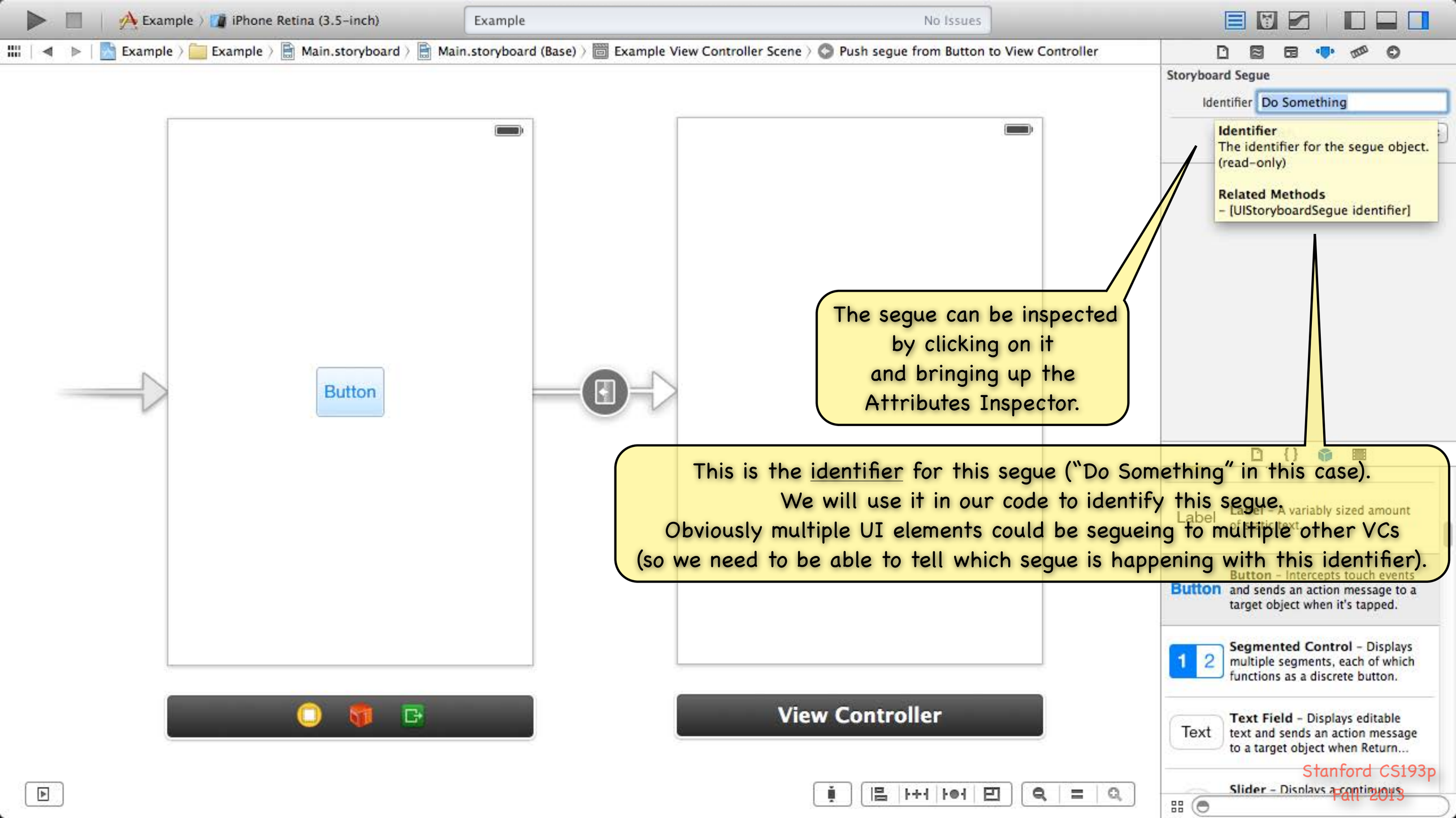

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Controller Scene
Example View Controller
View
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
White Color
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
0
X
Y
1
1
Width
Height
Let's drag out a Button that, when pressed, will show this new View Controller.
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
View Controller
Stanford CS193p
Fall 2013

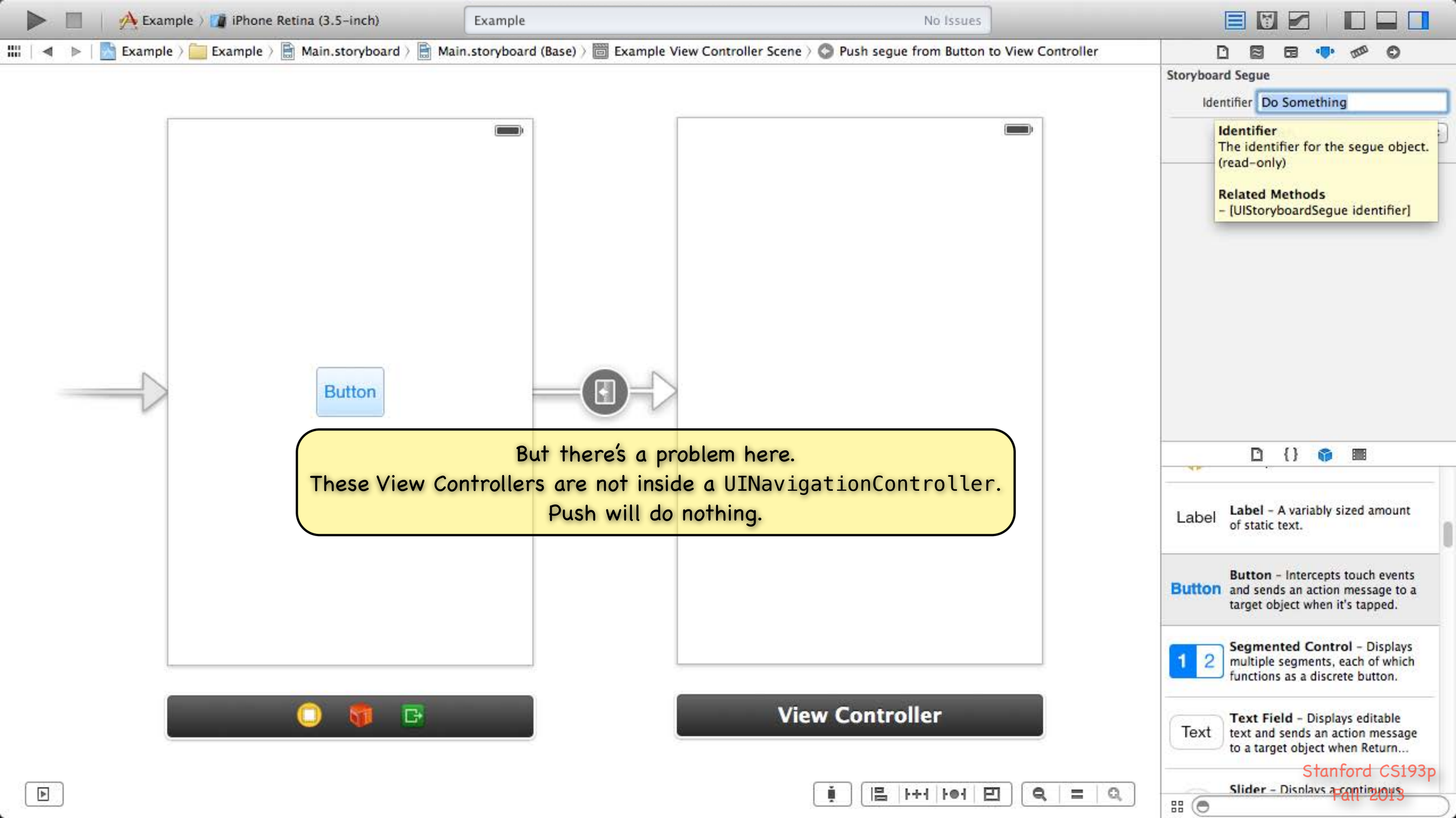

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Controll...
Example View Controller
View
Button – Button
Drop it here.
Button
View Controller
Button
Type
System
State Config
Default
Title
Plain
Button
Font
System 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Imag
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Drawing
Shows Touch On Highlight
Highlighted Adjusts Image
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

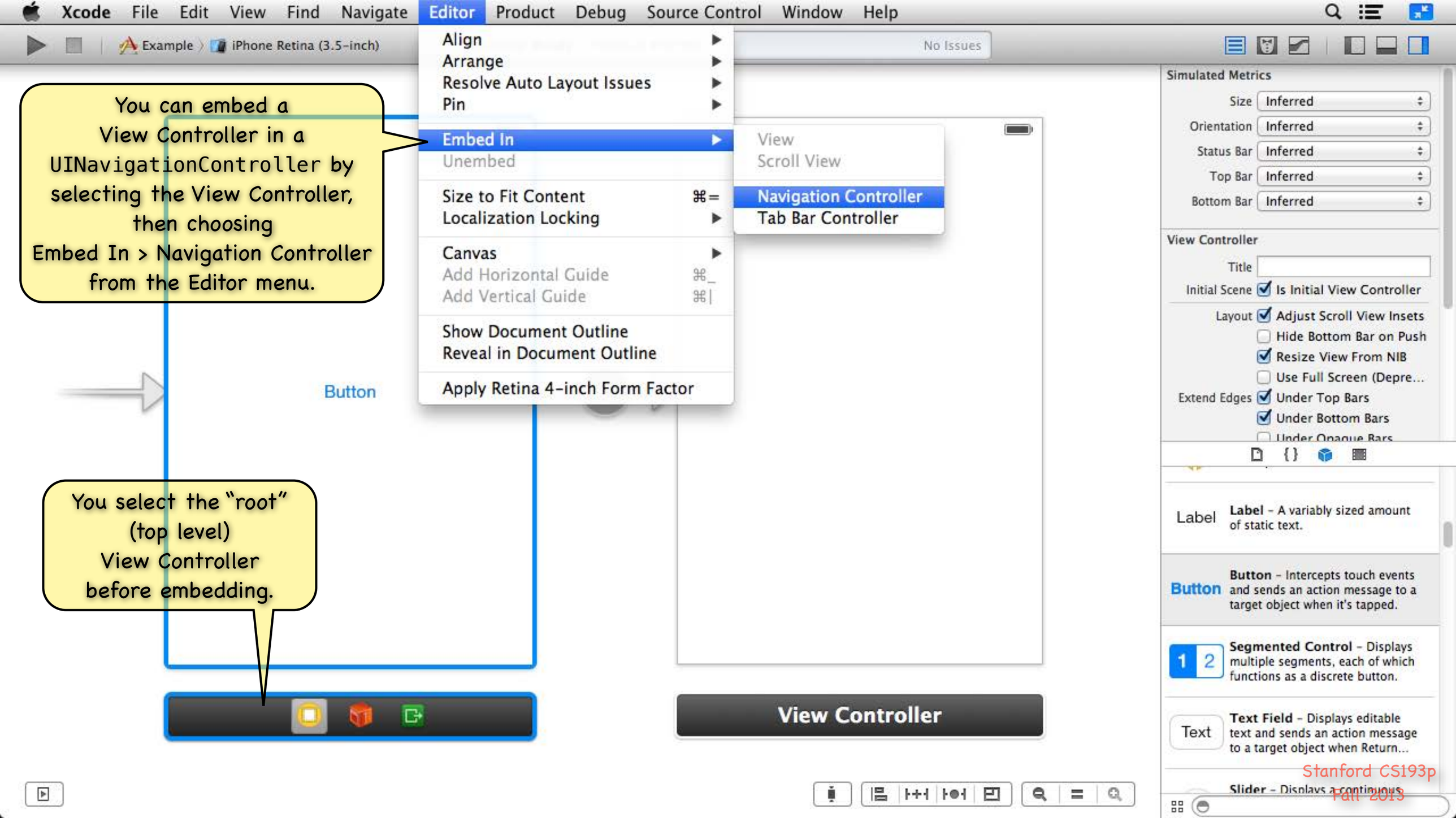

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Controll...
Example View Controller
View
Button – Button
To create a segue, you hold down ctrl and drag from a button (or other UI element) in one View Controller to another View Controller.
Button
View Controller
View Controller
Button
Type
System
State Config
Default
Title
Plain
Button
Font
System 15.0
Text Color
Default
Shadow Color
Default
Image
Default Image
Background
Default Background Imag
Shadow Offset
0.0
0.0
Width
Height
Reverses On Highlight
Drawing
Shows Touch On Highlight
Highlighted Adjusts Image
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

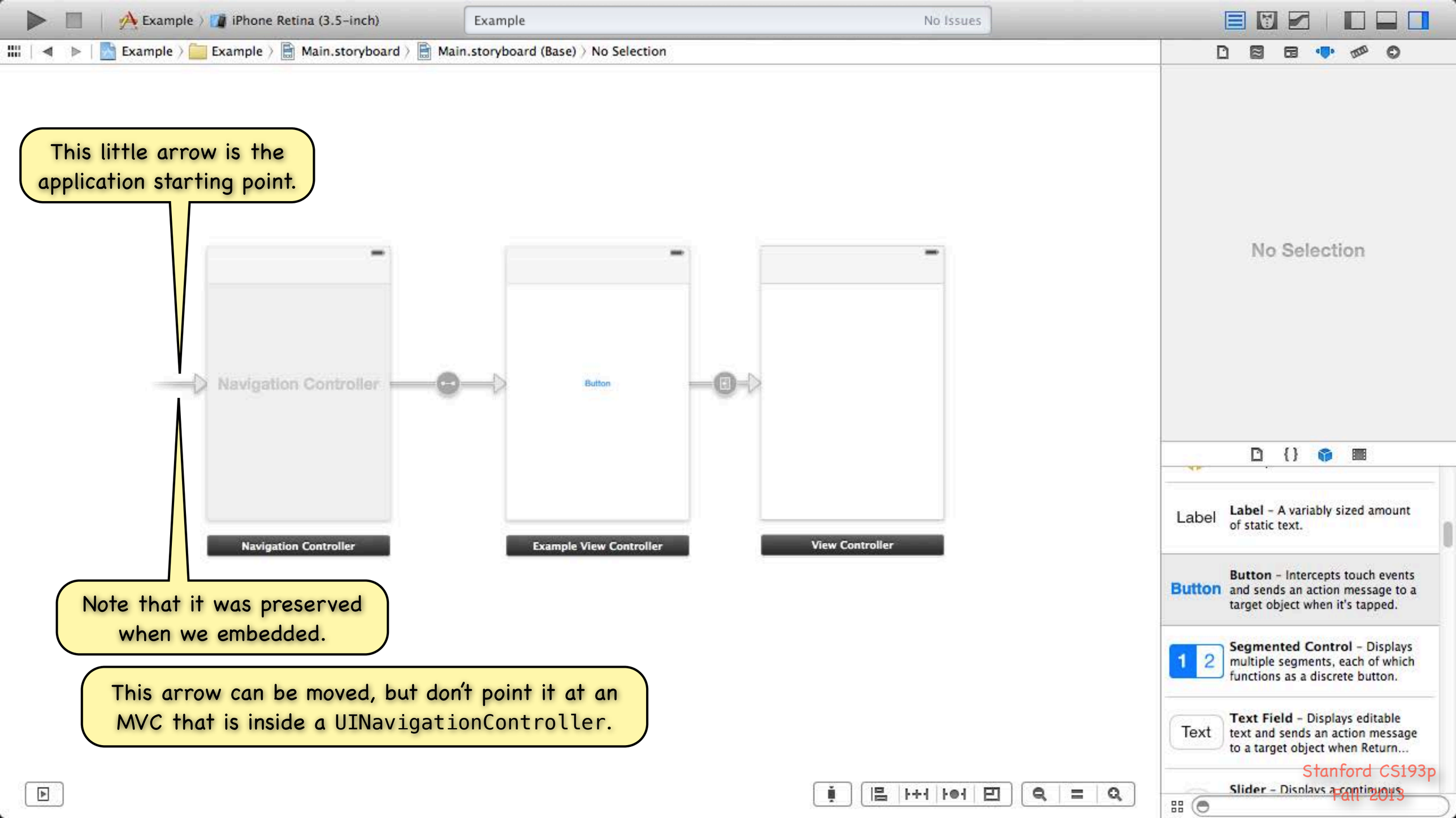

When you let go of the mouse, Xcode will ask what sort of segue you want to occur when Button is pressed.
"Push" is the kind of segue you use when the two Controllers are inside a UINavigationController.
Action Segue
push
modal
custom
Button
View Controller
Button
Type System
State Config Default
Title Plain
Button
Font System 15.0
Text Color Default
Shadow Color Default
Image Default Image
Background Default Background Imag
Shadow Offset 0.0 0.0
Width Height
Reverses On Highlight
Drawing Shows Touch On Highlight
Highlighted Adjusts Image
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

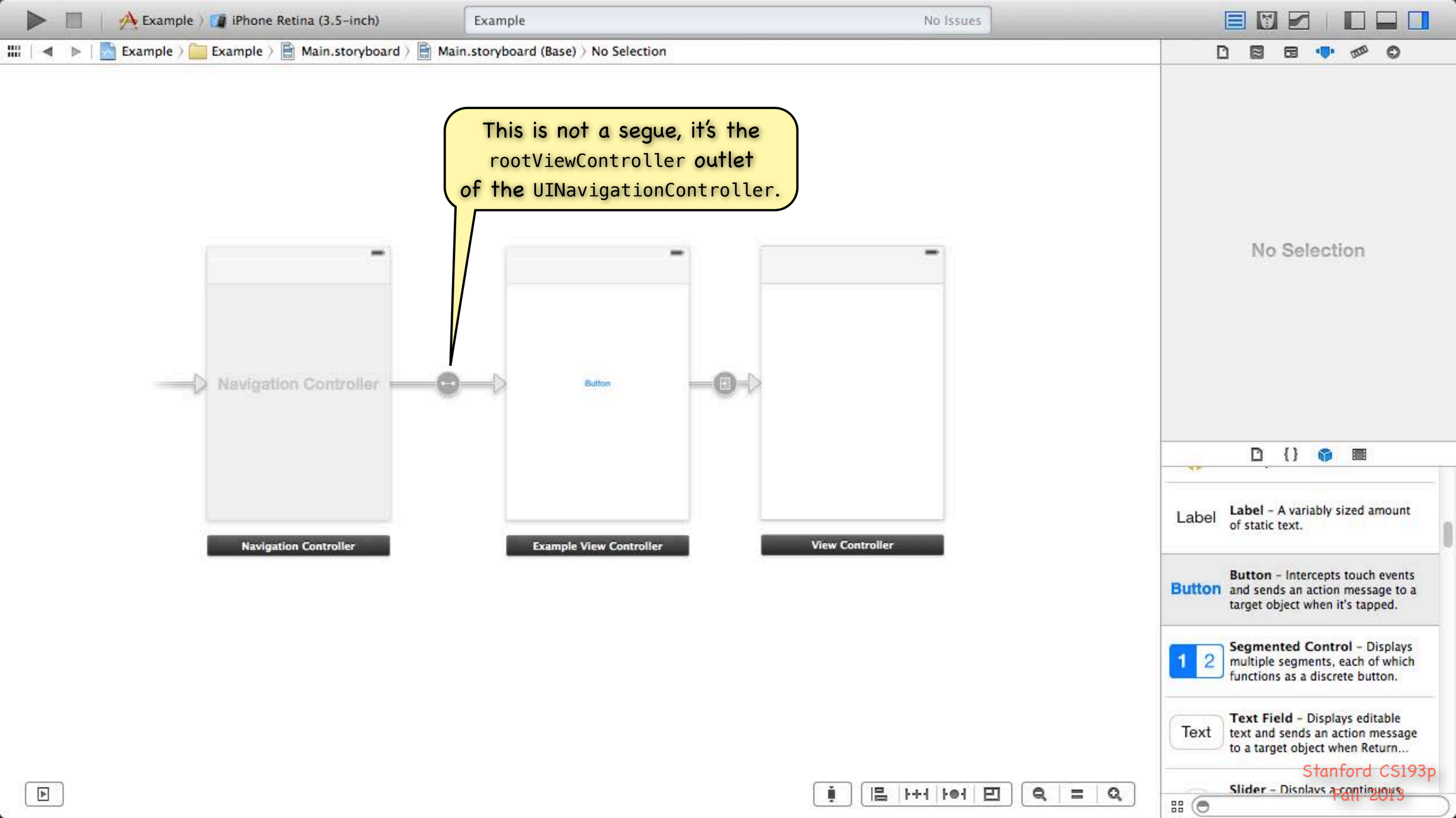


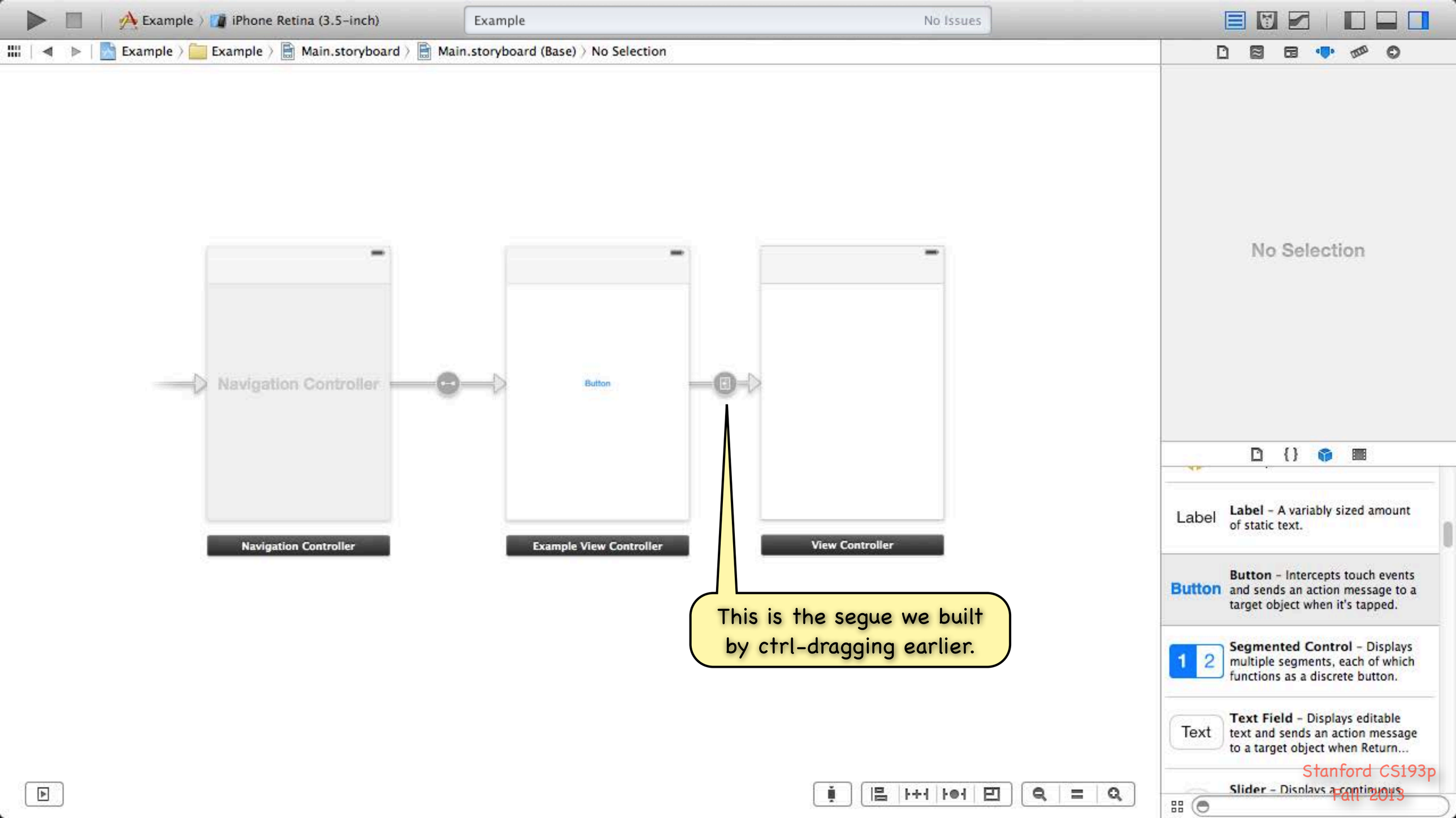

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Controller Scene
Push segue from Button to View Controller
Storyboard Segue
Identifier
Do Something
Identifier
The identifier for the segue object. (read-only)
Related Methods
- [UIStoryboardSegue identifier]
Button
View Controller
The segue can be inspected by clicking on it and bringing up the Attributes Inspector.
This is the identifier for this segue ("Do Something" in this case).
We will use it in our code to identify this segue.
Obviously multiple UI elements could be segueing to multiple other VCs
(so we need to be able to tell which segue is happening with this identifier).
Label – A variably sized amount of static text
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

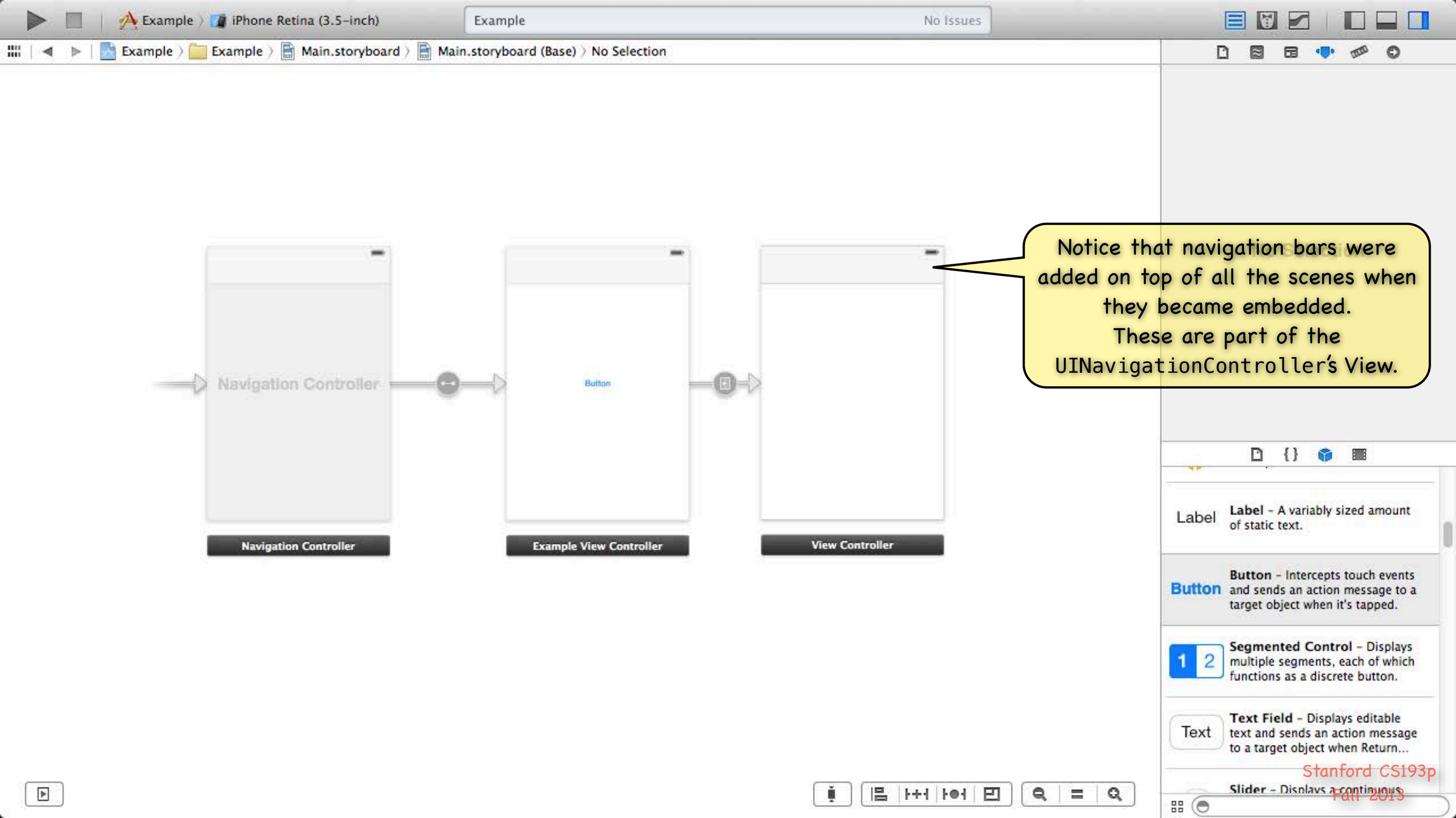

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Controller Scene
Push segue from Button to View Controller
Storyboard Segue
Identifier
Do Something
Identifier
The identifier for the segue object. (read-only)
Related Methods
- [UIStoryboardSegue identifier]
Button
But there's a problem here.
These View Controllers are not inside a UINavigationController.
Push will do nothing.
View Controller
Label
Label - A variably sized amount of static text.
Button
Button - Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control - Displays multiple segments, each of which functions as a discrete button.
Text
Text Field - Displays editable text and sends an action message to a target object when Return...
Slider - Displays a continuous
Stanford CS193p
Fall 2013

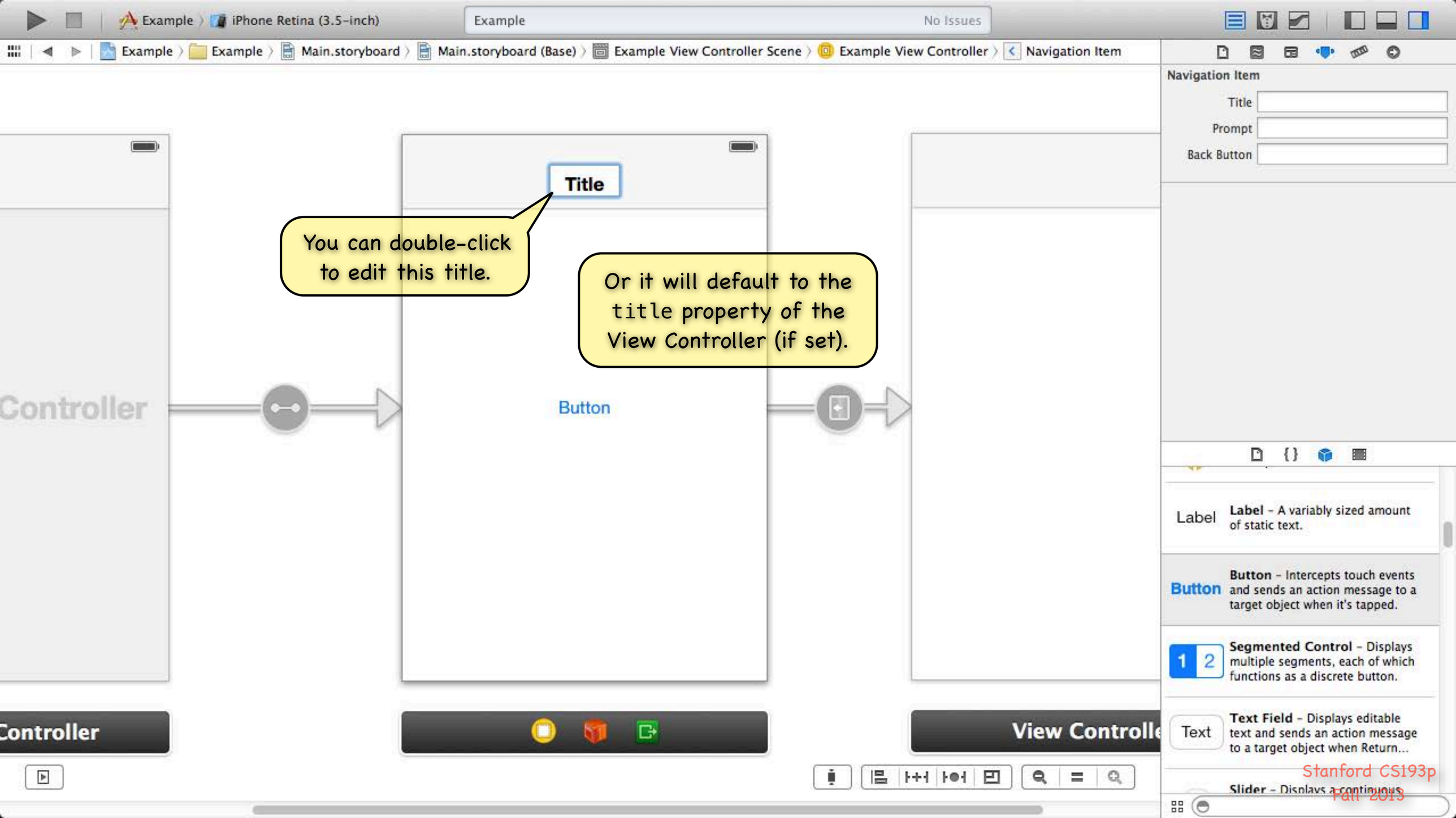

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Example iPhone Retina (3.5-inch)
No Issues
Align
Arrange
Resolve Auto Layout Issues
Pin
Embed In
Unembed
Size to Fit Content ⌘=
Localization Locking
Canvas
Add Horizontal Guide ⌘_
Add Vertical Guide ⌘|
Show Document Outline
Reveal in Document Outline
Apply Retina 4-inch Form Factor
View
Scroll View
Navigation Controller
Tab Bar Controller
You can embed a View Controller in a UINavigationController by selecting the View Controller, then choosing Embed In > Navigation Controller from the Editor menu.
Button
You select the "root" (top level) View Controller before embedding.
View Controller
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges Under Top Bars
Under Bottom Bars
Label Label – A variably sized amount of static text.
Button Button – Intercepts touch events and sends an action message to a target object when it's tapped.
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

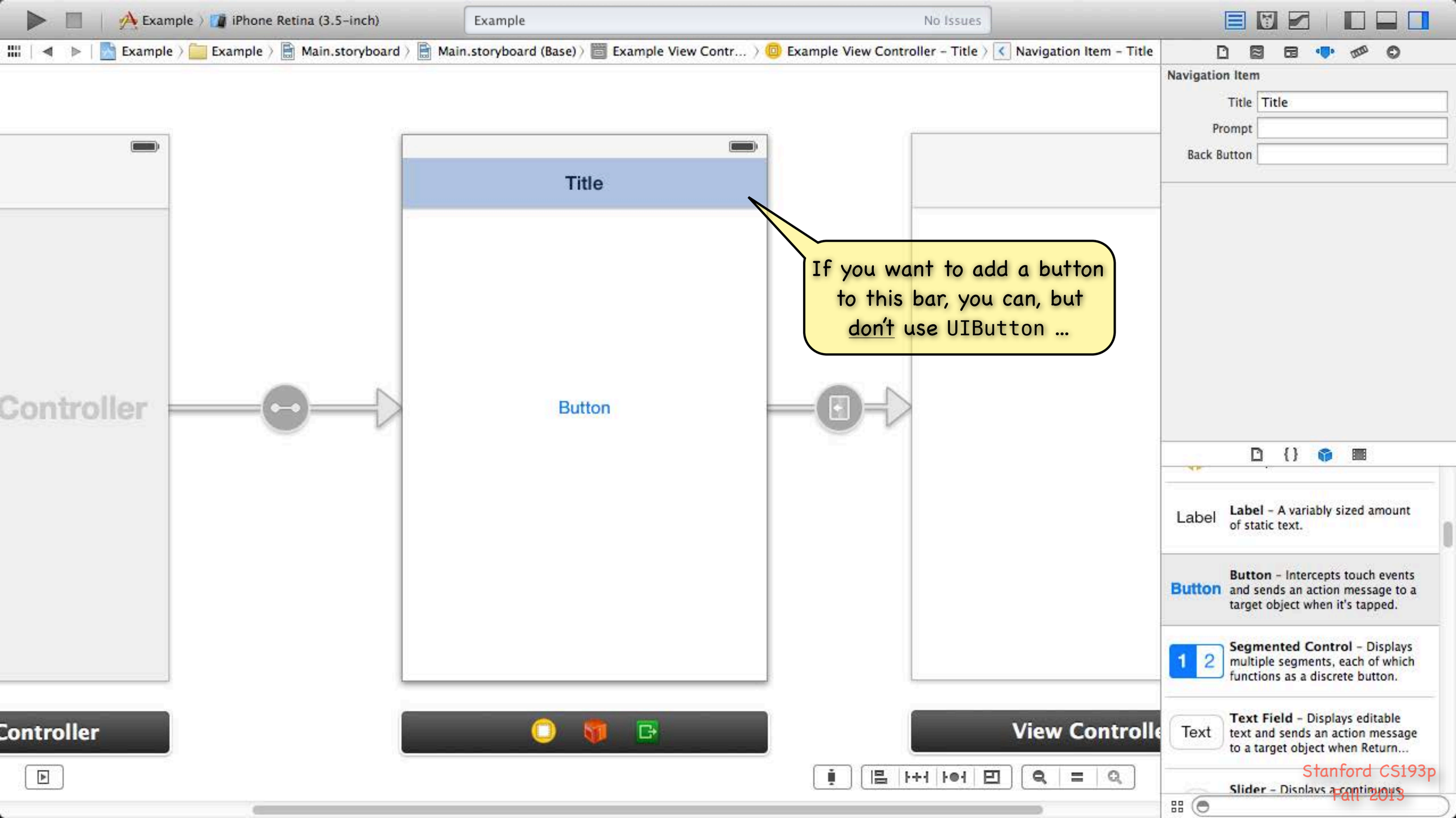

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
No Selection
This little arrow is the application starting point.
Navigation Controller
Button
Navigation Controller
Example View Controller
View Controller
Note that it was preserved when we embedded.
This arrow can be moved, but don't point it at an MVC that is inside a UINavigationController.
No Selection
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

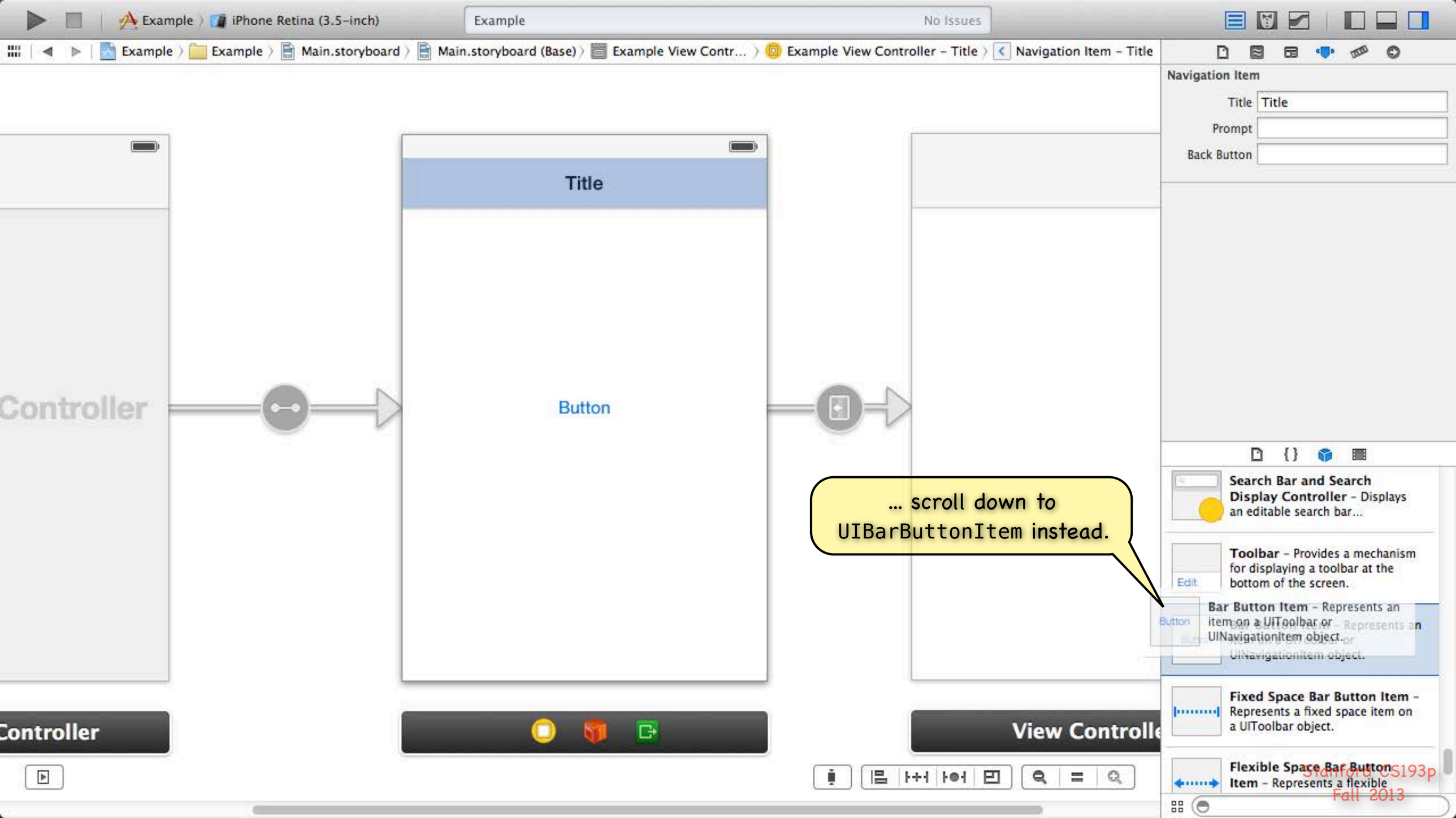

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
No Selection
This is not a segue, it's the rootViewController outlet of the UINavigationController.
Navigation Controller
Button
Navigation Controller
Example View Controller
View Controller
No Selection
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013

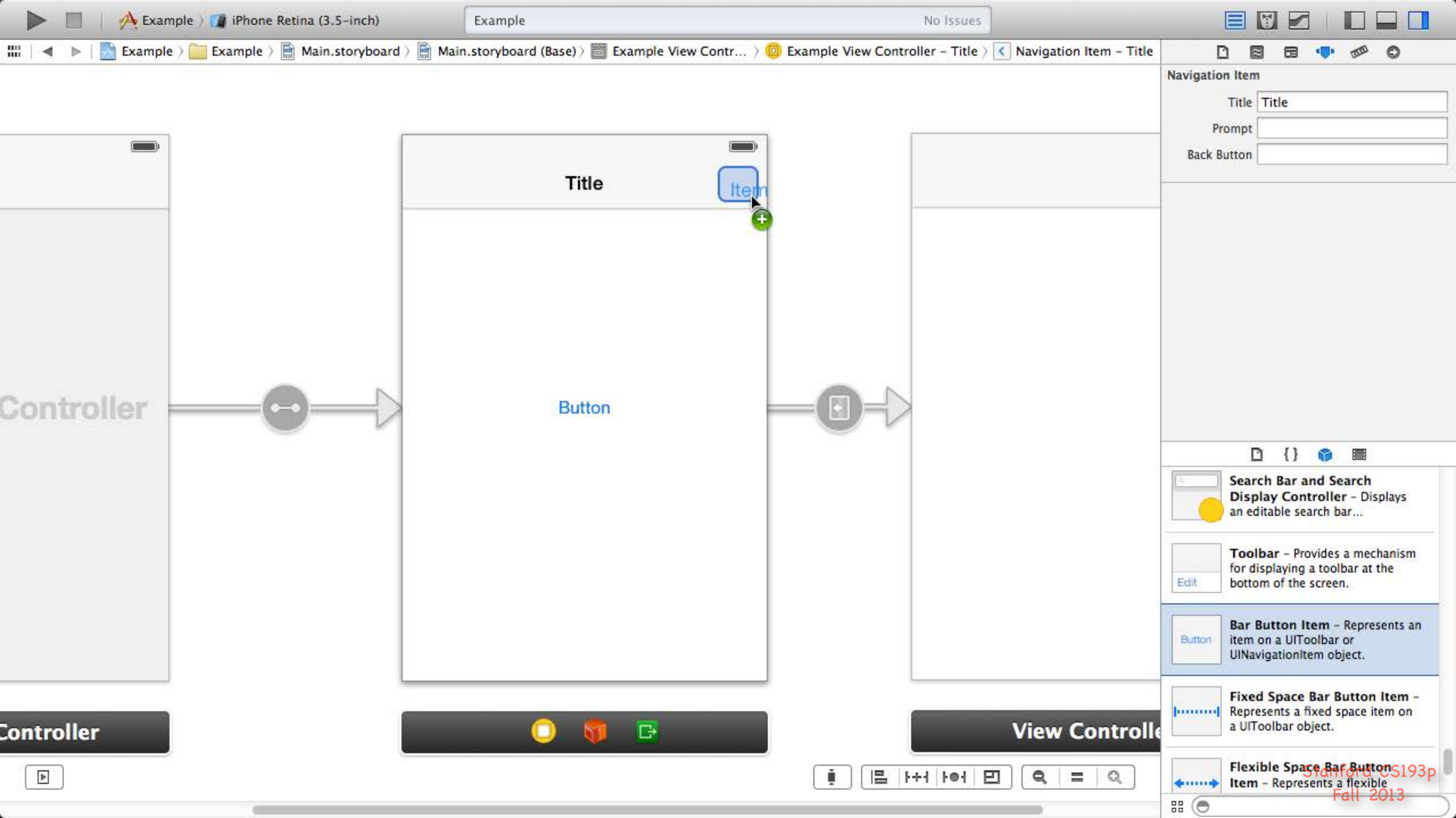

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
No Selection
Navigation Controller
Button
Navigation Controller
Example View Controller
View Controller
This is the segue we built by ctrl-dragging earlier.
No Selection
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous

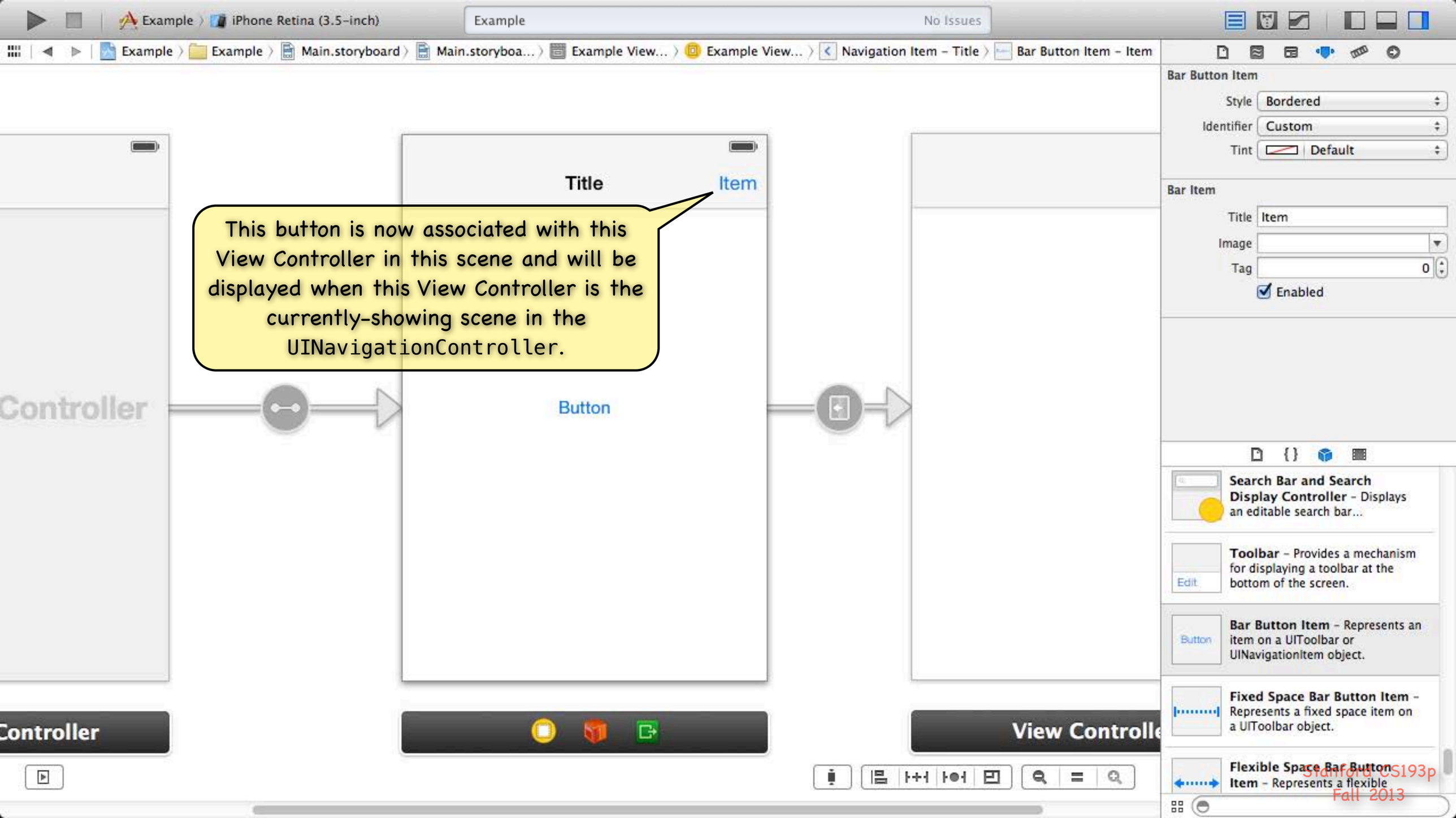

Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
No Selection
Navigation Controller
Button
Navigation Controller
Example View Controller
View Controller
Notice that navigation bars were added on top of all the scenes when they became embedded.
These are part of the UINavigationController's View.
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous


Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Controller Scene
Example View Controller
Navigation Item
Navigation Item
Title
Prompt
Back Button
Title
You can double-click to edit this title.
Or it will default to the title property of the View Controller (if set).
Button
Controller
Controller
View Controlle
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Contr...
Example View Controller – Title
Navigation Item – Title
Navigation Item
Title
Title
Prompt
Back Button
Title
Button
If you want to add a button to this bar, you can, but don't use UIButton ...
Label
Label – A variably sized amount of static text.
Button
Button – Intercepts touch events and sends an action message to a target object when it's tapped.
1 2
Segmented Control – Displays multiple segments, each of which functions as a discrete button.
Text
Text Field – Displays editable text and sends an action message to a target object when Return...
Slider – Displays a continuous
Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Contr...
Example View Controller – Title
Navigation Item – Title
Navigation Item
Title
Title
Prompt
Back Button
Title
Button
Controller
Controller
View Controlle
… scroll down to UIBarButtonItem instead.
Search Bar and Search Display Controller – Displays an editable search bar...
Toolbar – Provides a mechanism for displaying a toolbar at the bottom of the screen.
Edit
Button
Bar Button Item – Represents an item on a UIToolbar or UINavigationItem object.
Fixed Space Bar Button Item – Represents a fixed space item on a UIToolbar object.
Flexible Space Bar Button Item – Represents a flexible
Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
Example
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Example View Contr...
Example View Controller – Title
Navigation Item – Title
Navigation Item
Title
Title
Prompt
Back Button
Controller
Title
Item
Button
Controller
View Controlle
Search Bar and Search Display Controller – Displays an editable search bar...
Toolbar – Provides a mechanism for displaying a toolbar at the bottom of the screen.
Edit
Bar Button Item – Represents an item on a UIToolbar or UINavigationItem object.
Button
Fixed Space Bar Button Item – Represents a fixed space item on a UIToolbar object.
Flexible Space Bar Button Item – Represents a flexible
Stanford CS193p
Fall 2013



UINavigationController

- When does a pushed MVC pop off?

  Usually because the user presses the "back" button (shown on the previous slide).

  But it can happen programmatically as well with this `UINavigationController` instance method

  ```
  - (void)popViewControllerAnimated:(BOOL)animated;
  ```

  This does the same thing as clicking the back button.

  Somewhat rare to call this method. Usually we want the user in control of navigating the stack.

  But you might do it if some action the user takes in a view makes it irrelevant to be on screen.

- Example

  Let's say we push an MVC which displays a database record and has a delete button w/this action:

```
- (IBAction)deleteCurrentRecord:(UIButton *)sender
{
    // delete the record we are displaying
    // we just deleted the record we are displaying!
    // so it does not make sense to be on screen anymore, so pop
    [self.navigationController popViewControllerAnimated:YES];
}
```

Notice that all `UIViewControllers` know the `UINavigationController` they are in. This is `nil` if they are not in one.

View Controller

- Other kinds of segues besides Push

  Replace - Replaces the right-hand side of a `UISplitViewController` (iPad only)

  Popover - Puts the view controller on the screen in a popover (iPad only)

  Modal - Puts the view controller up in a way that blocks the app until it is dismissed

  Custom - You can create your own subclasses of `UIStoryboardSegue`

- We'll talk about iPad-related segues in future lectures

  Replace & Popover

- We'll talk about Modal segues later in the quarter too

  People often use Modal UIs as a crutch, so we don't want to go to that too early.

View Controller

- Firing off a segue from code

  Sometimes it makes sense to segue directly when a button is touched, but not always.

  For example, what if you want to conditionally segue?

  You can programmatically invoke segues using this method in `UIViewController`:

```
- (void)performSegueWithIdentifier:(NSString *)segueId sender:(id)sender;
```

  The `segueId` is set in the attributes inspector in Xcode (seen on previous slide).

  The `sender` is the initiator of the segue (a `UIButton` or yourself (`UIViewController`) usually).

```
- (IBAction)rentEquipment
{
    if (self.snowTraversingTalent == Skiing) {
        [self performSegueWithIdentifier:@“AskAboutSkis” sender:self];
    } else {
        [self performSegueWithIdentifier:@“AskAboutSnowboard” sender:self];
    }
}
```

Segues

- When a segue happens, what goes on in my code?

  The segue offers the source VC the opportunity to "prepare" the new VC to come on screen.

  This method is sent to the VC that contains the button that initiated the segue:

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([segue.identifier isEqualToString:@"DoSomething"]) {
        if ([segue.destinationViewController isKindOfClass:[DoSomethingVC class]]) {
            DoSomethingVC *doVC = (DoSomethingVC *)segue.destinationViewController;
            doVC.neededInfo = …;
        }
    }
}
```

  You should pass data the new VC needs here and "let it run."

  Think of the new VC as part of the View of the Controller that initiates the segue.

  It must play by the same rules as a View.

  For example, it should not talk back to you (except through blind communication like delegation).

Segues

- You can prevent a segue from happening

  Your Controller usually just always segues.

  But if you respond NO to this method, it would prevent the identified segue from happening.

```
- (BOOL)shouldPerformSegueWithIdentifier:(NSString *)identifier sender:(id)sender
{
    if ([segue.identifier isEqualToString:@“DoAParticularThing”]) {
        return [self canDoAParticularThing] ? YES : NO;
    }
}
```

  Do not create “dead UI” with this (e.g. buttons that do nothing).

  This is a very rare method to ever implement.

Unwinding

- There are also ways to unwind from a series of segues

  Some people think of this as "reverse segueing".

  Used if you want to dismiss the VC you are in and go back to a previous VC that segued to you.

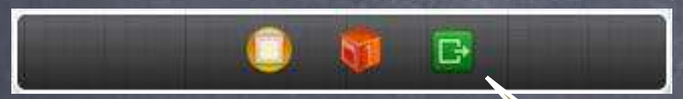
  For example, what if you wanted to pop back multiple levels in a navigation controller?

  (if you were only going back one level, you could just use `popViewControllerAnimated:`).

  The little green button in the black bar at the bottom of a scene can be used to wire that up.

  We will probably cover this when we talk about the Modal segue type (i.e. later).

  You need to master segueing forward before you start thinking about going backward!



This is the "little green button."

View Controller

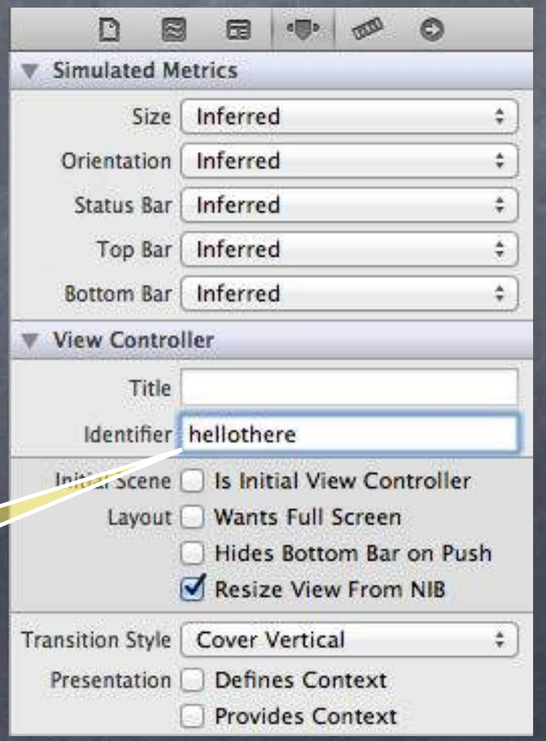
- Instantiating a `UIViewController` by name from a storyboard

  Sometimes (very rarely) you might want to put a VC on screen yourself (i.e., not use a segue).

```
NSString *vcid = @"something";
UIViewController *controller = [storyboard instantiateViewControllerWithIdentifier:vcid];
```

  Usually you get the `storyboard` above from `self.storyboard` in an existing `UIViewController`.

  The identifier `vcid` must match a string you set in Xcode to identify a `UIViewController` there.



This `UIViewController` in the storyboard can be instantiated using the identifier "`hellothere`".

View Controller

- Instantiating a UIViewController by name from a storyboard

  Sometimes (very rarely) you might want to put a VC on screen yourself (i.e., not use a segue).

```
NSString *vcid = @"something";
UIViewController *controller = [storyboard instantiateViewControllerWithIdentifier:vcid];
```

  Usually you get the storyboard above from self.storyboard in an existing UIViewController.

  The identifier vcid must match a string you set in Xcode to identify a UIViewController there.

- Example: creating a UIViewController in a target/action method

  Lay out the View for a DoitViewController in your storyboard and name it "doit1".

```
- (IBAction)doit
{
    DoitViewController *doit =
        [self.storyboard instantiateViewControllerWithIdentifier:@"doit1"];
    doit.infoDoitNeeds = self.info;
    [self.navigationController pushViewController:doit animated:YES];
}
```

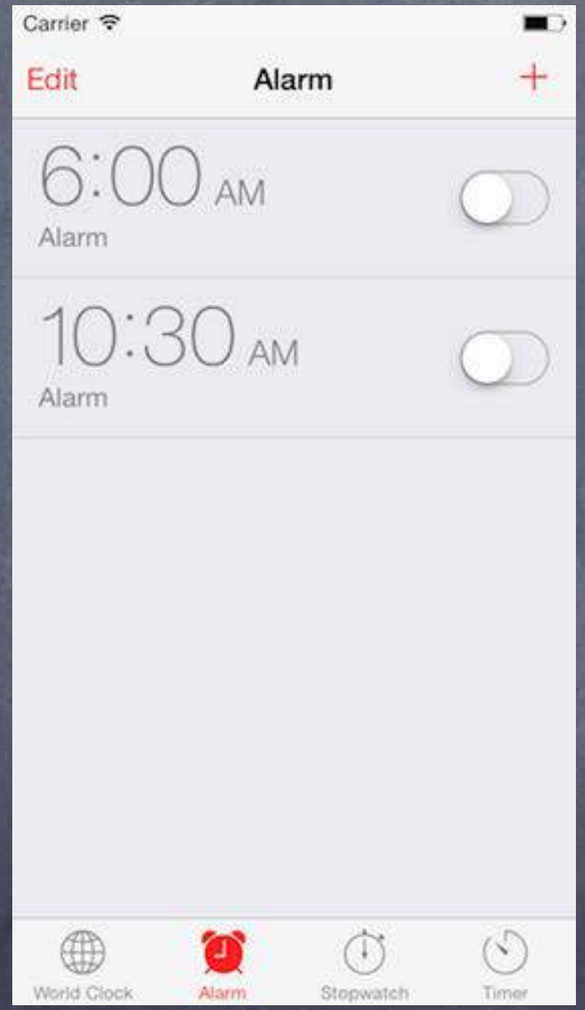
Note use of self.navigationController again.

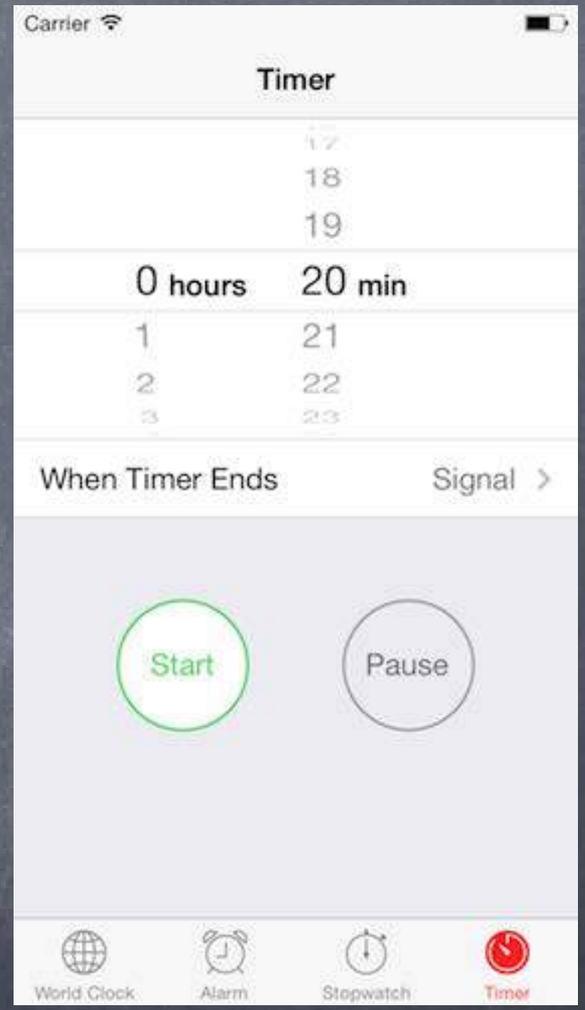
Demo

- Attributor Stats

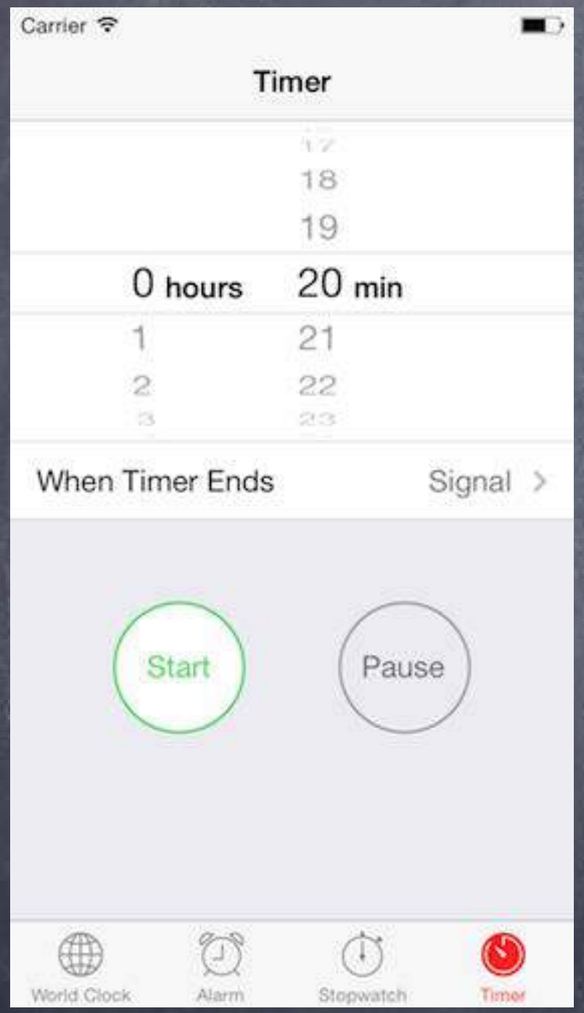
  Use a `UINavigationController` to show "statistics" on colors and outlining in Attributor.

UITabBarController





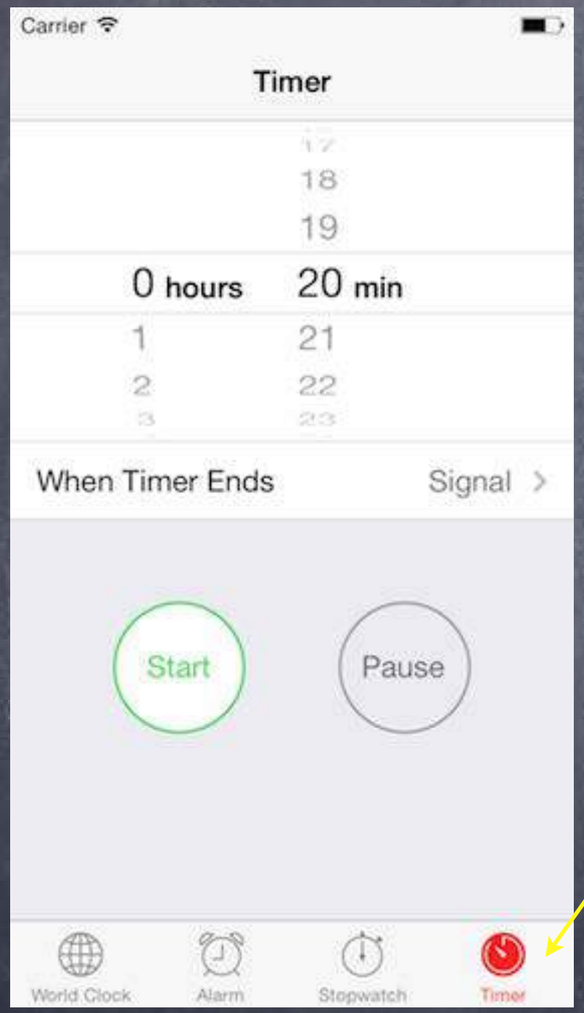
UITabBarController



You control drag to create these connections in Xcode.

Doing so is setting
`@property (nonatomic, strong) NSArray *viewControllers;`
inside your UITabBarController.

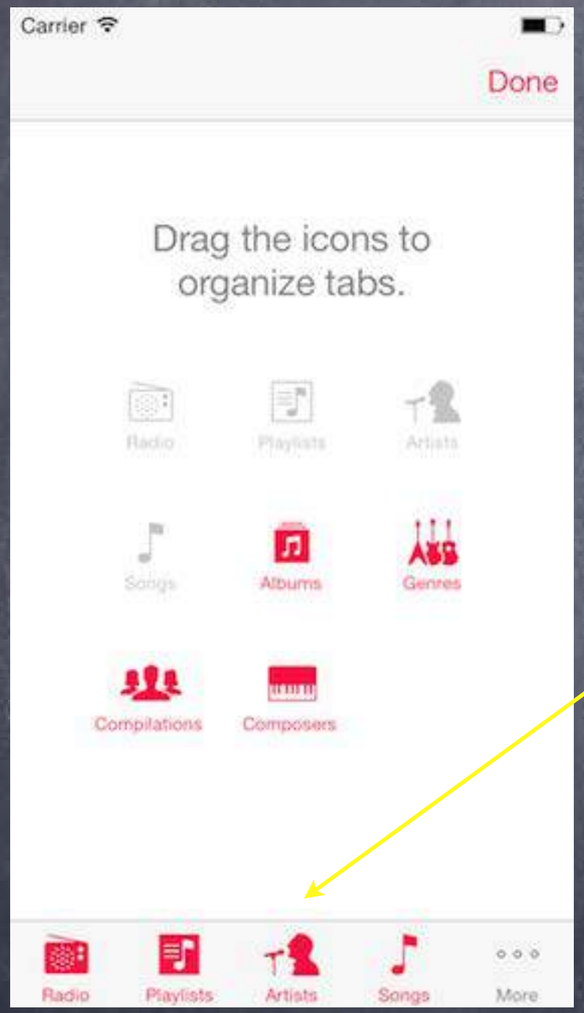
UITabBarController



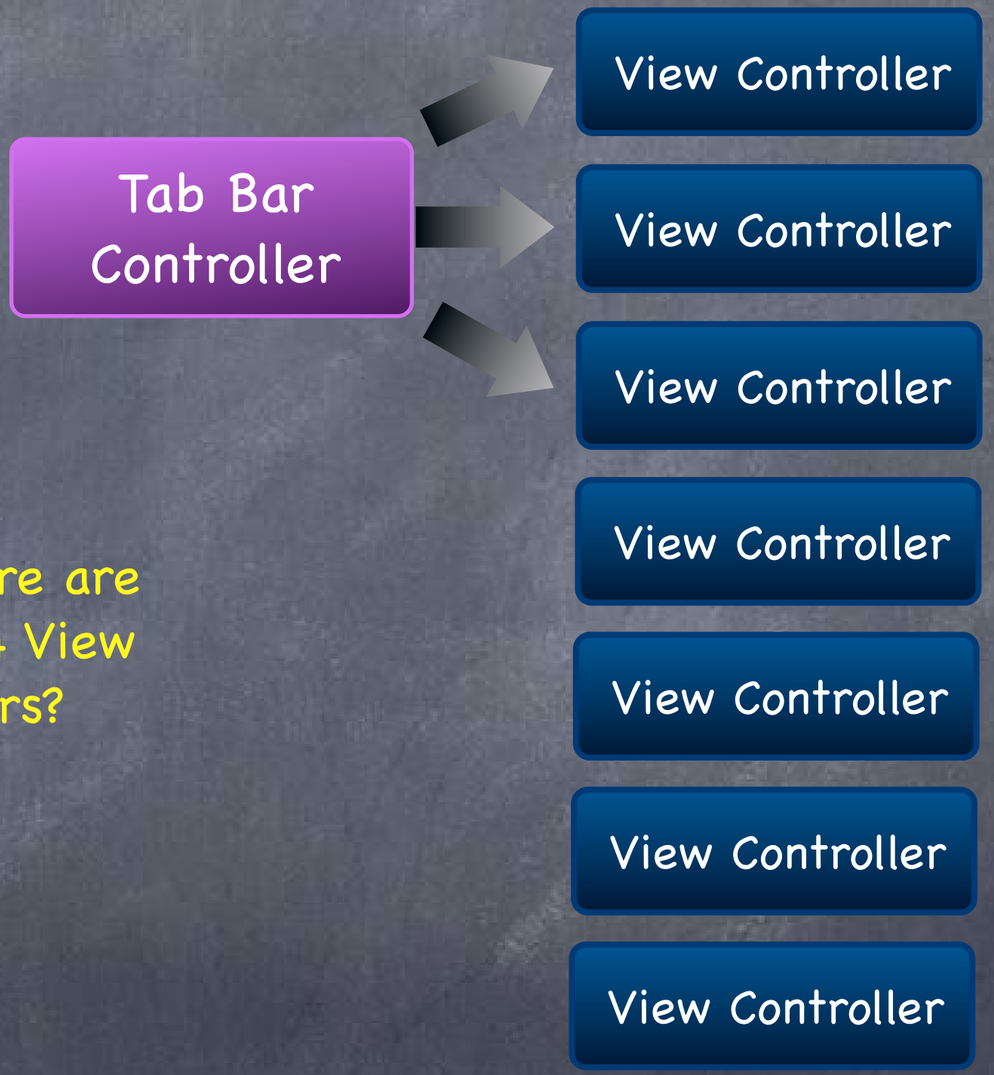
By default this is the `UIViewController`'s `title` property (and no image)

But usually you set both of these in your storyboard in Xcode.

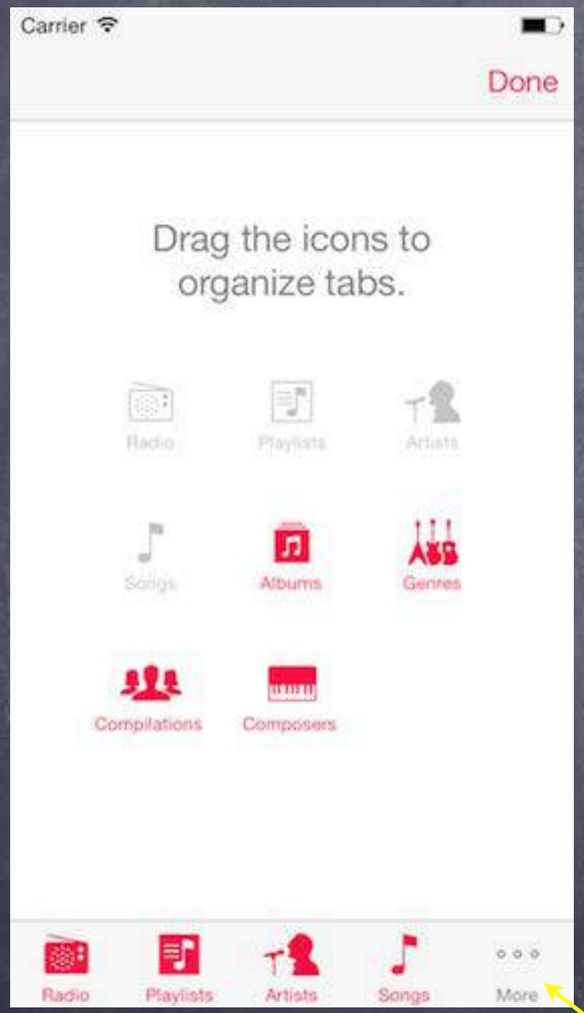
UITabBarController



What if there are more than 4 View Controllers?

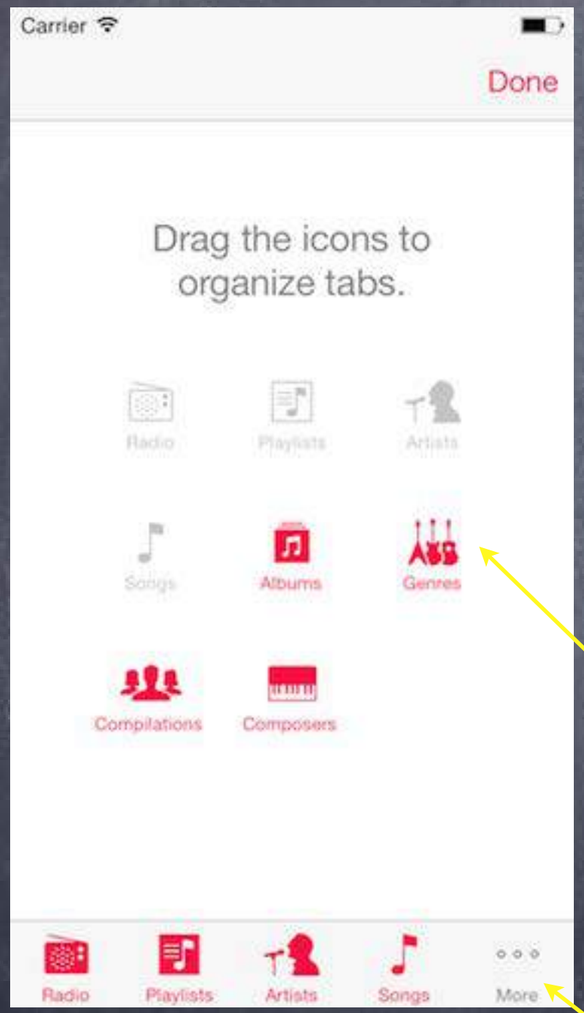


UITabBarController



A `More` button appears.

UITabBarController



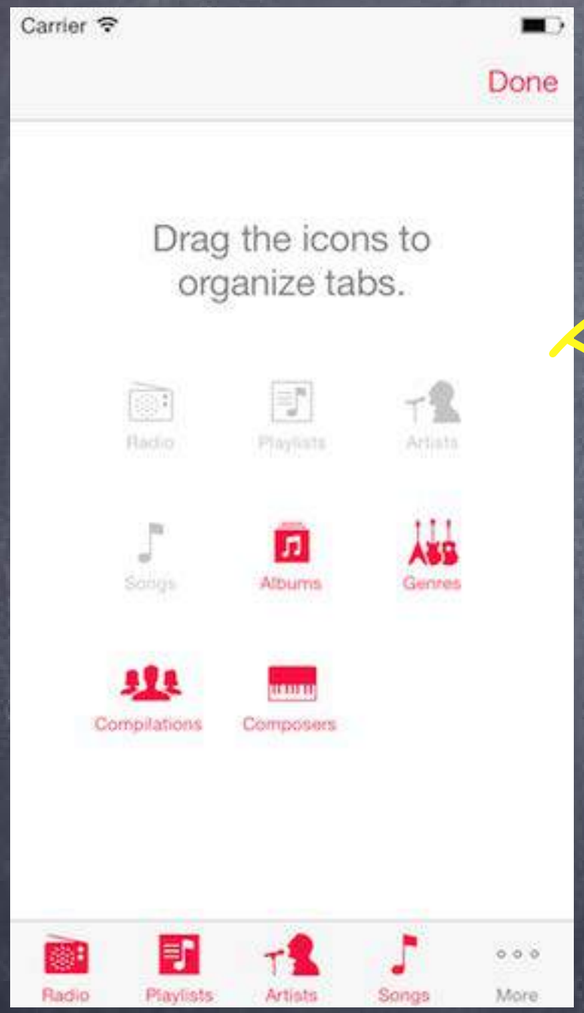
More button brings up a UI to let the user edit which buttons appear on bottom row

A More button appears.

Tab Bar Controller

- View Controller
- View Controller
- View Controller
- View Controller
- View Controller
- View Controller
- View Controller

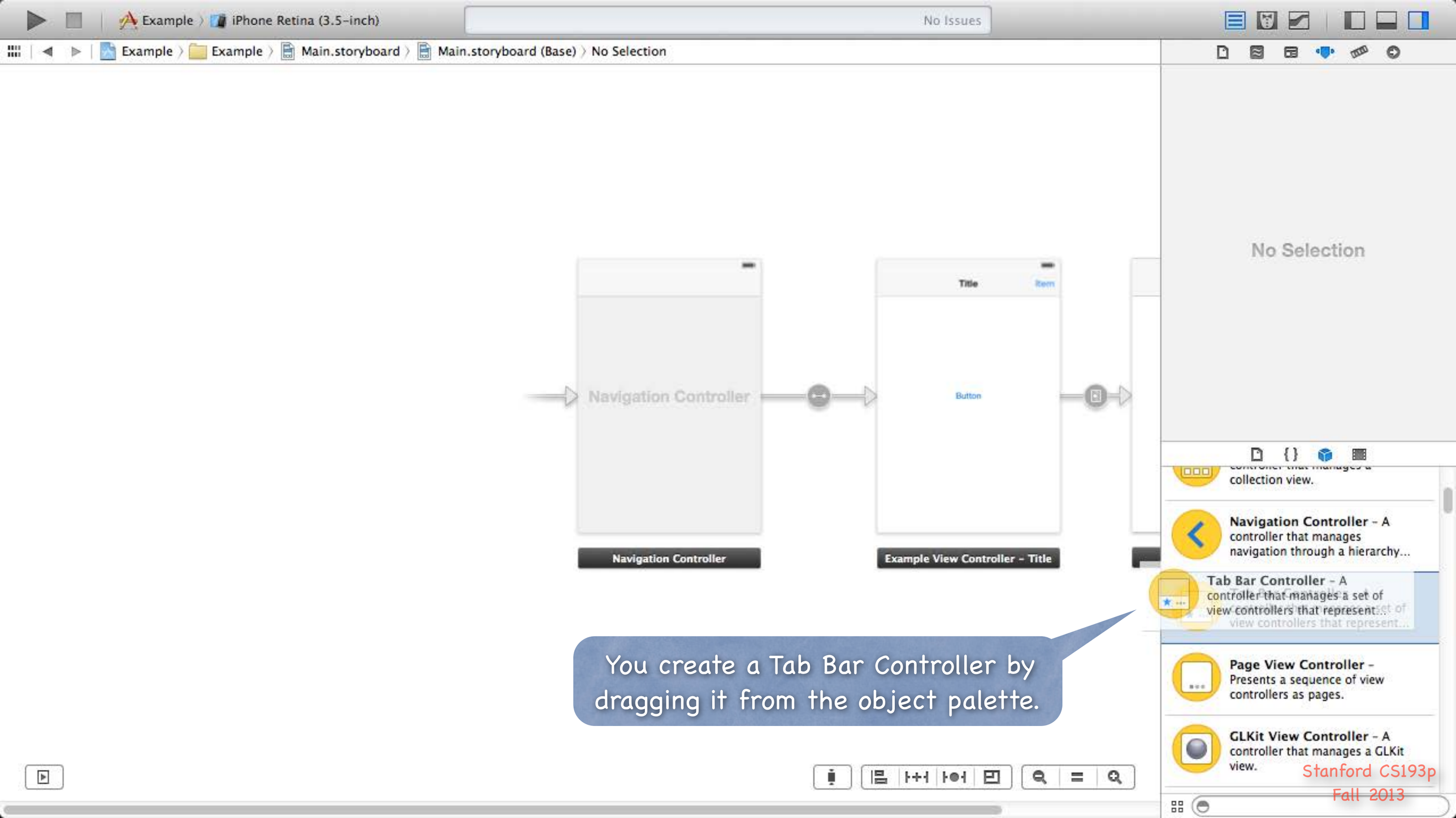
UITabBarController

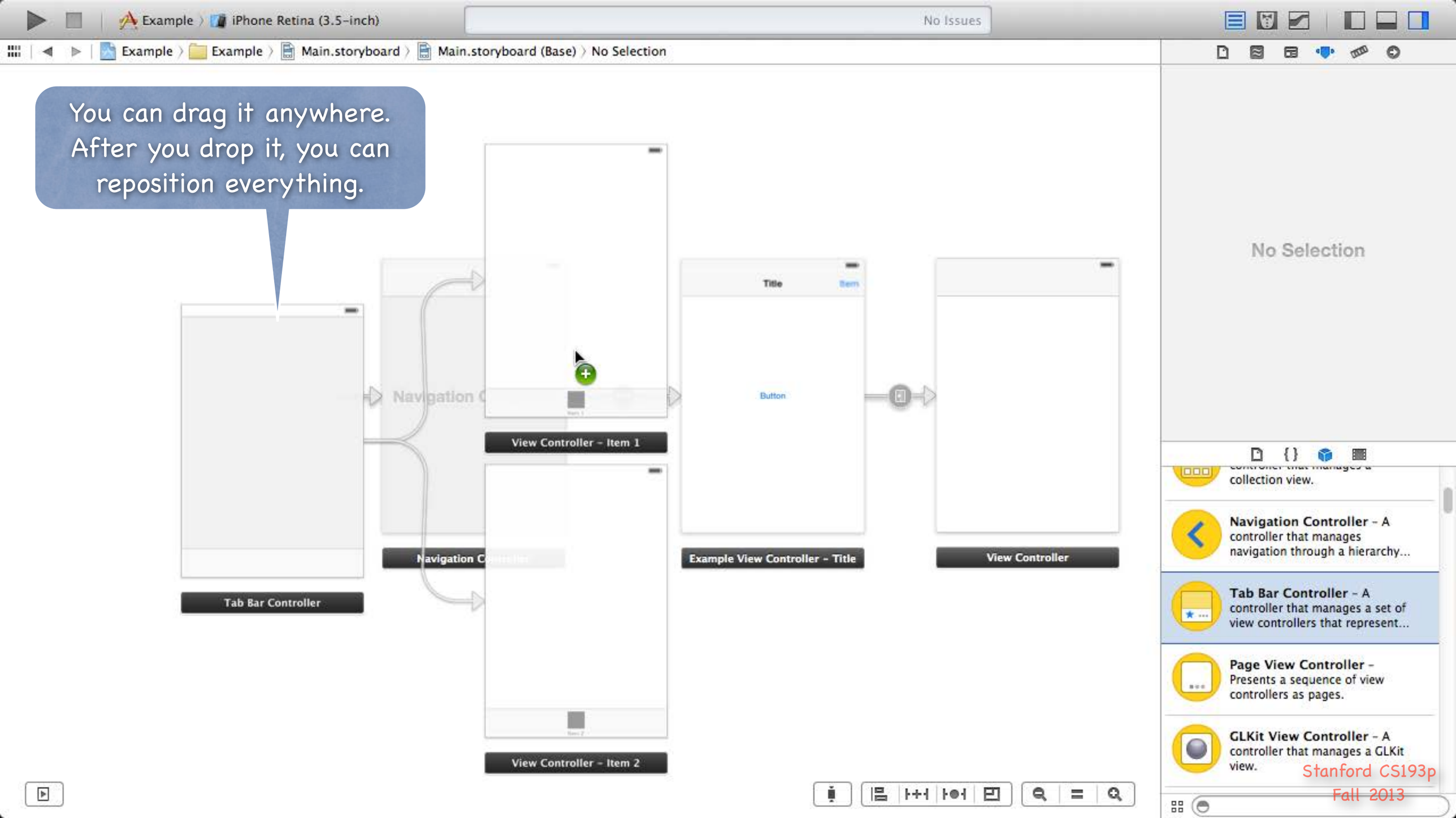


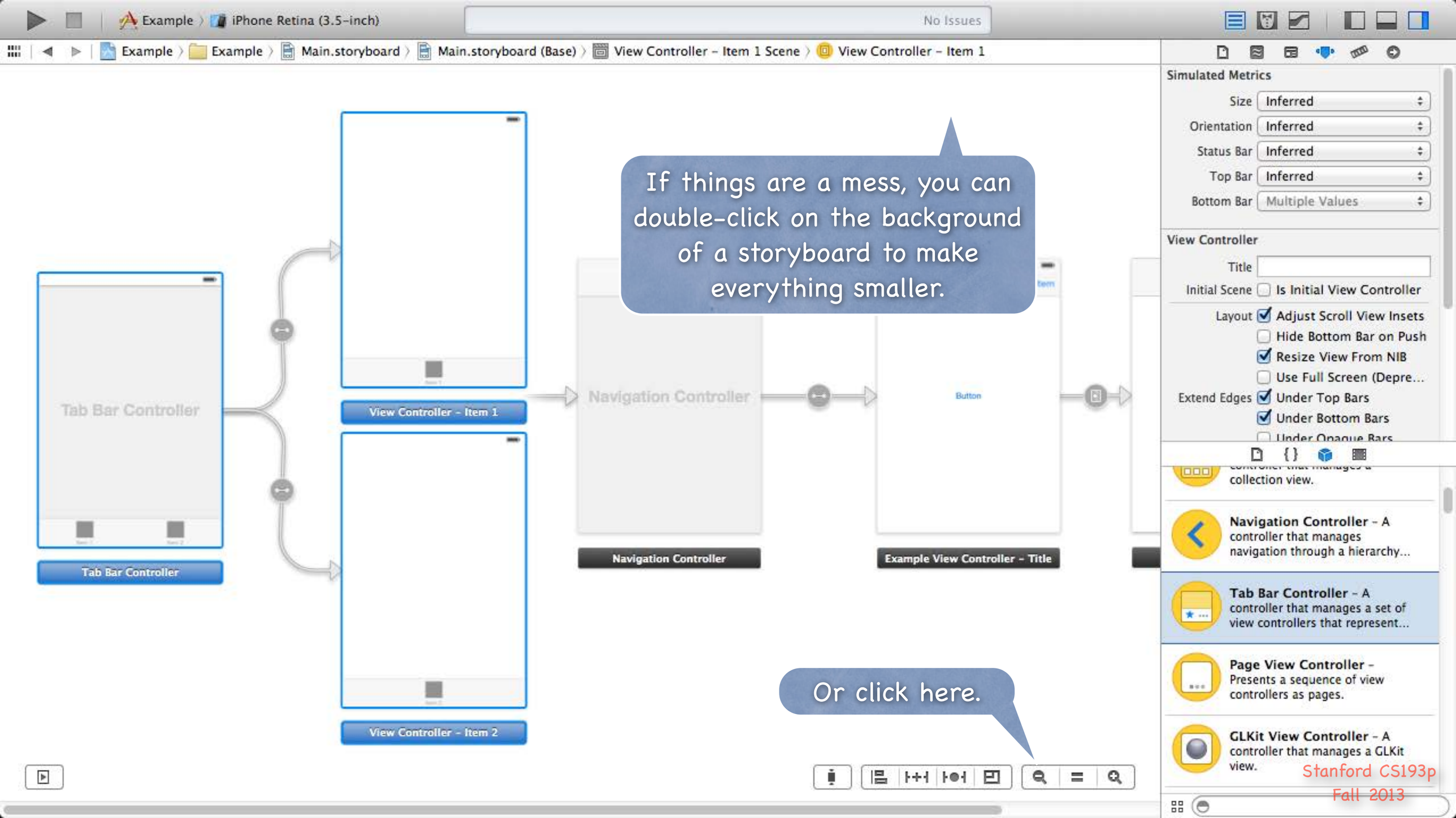
All Happens Automatically

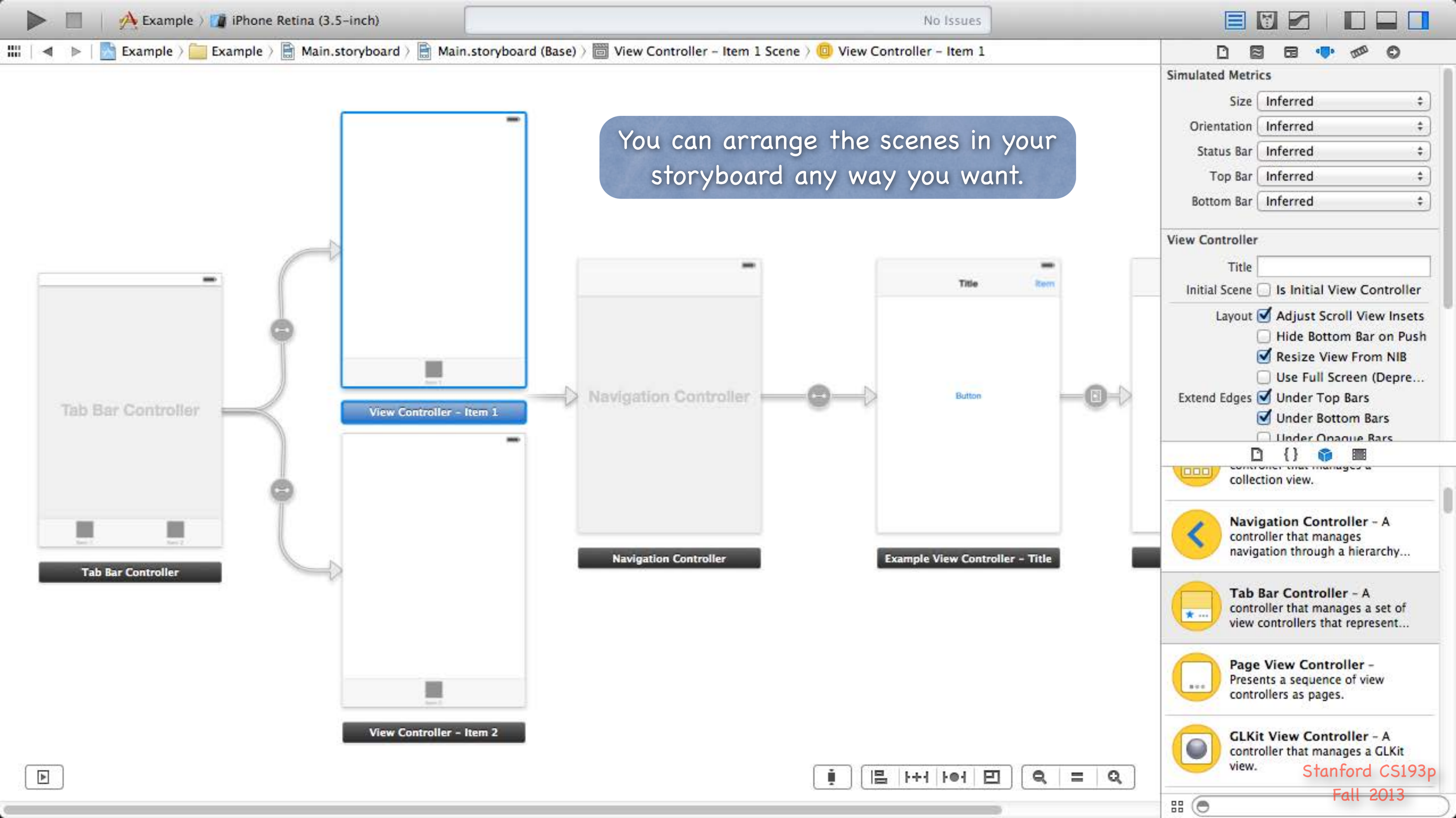
Tab Bar Controller

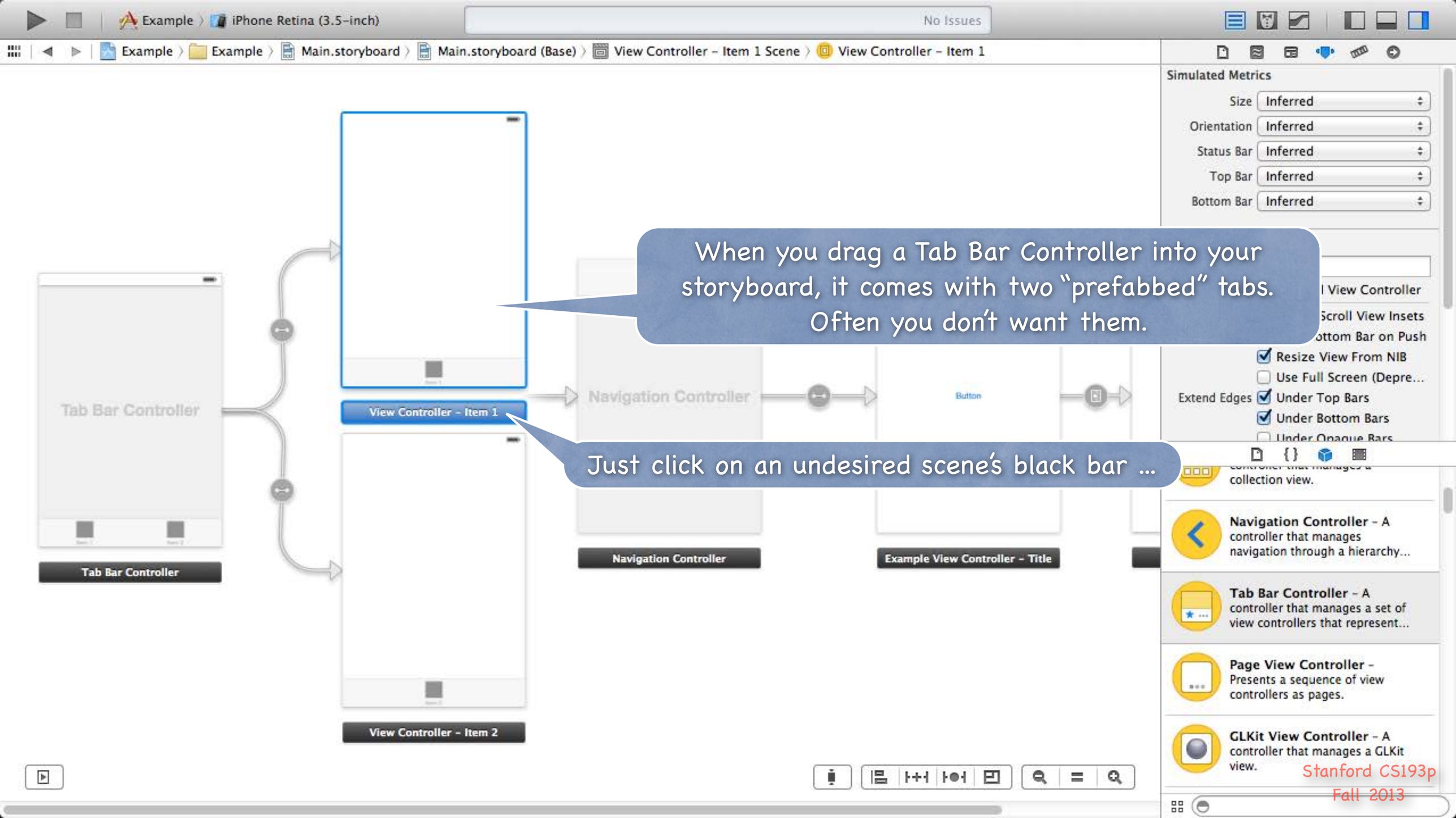
- View Controller
- View Controller
- View Controller
- View Controller
- View Controller
- View Controller
- View Controller

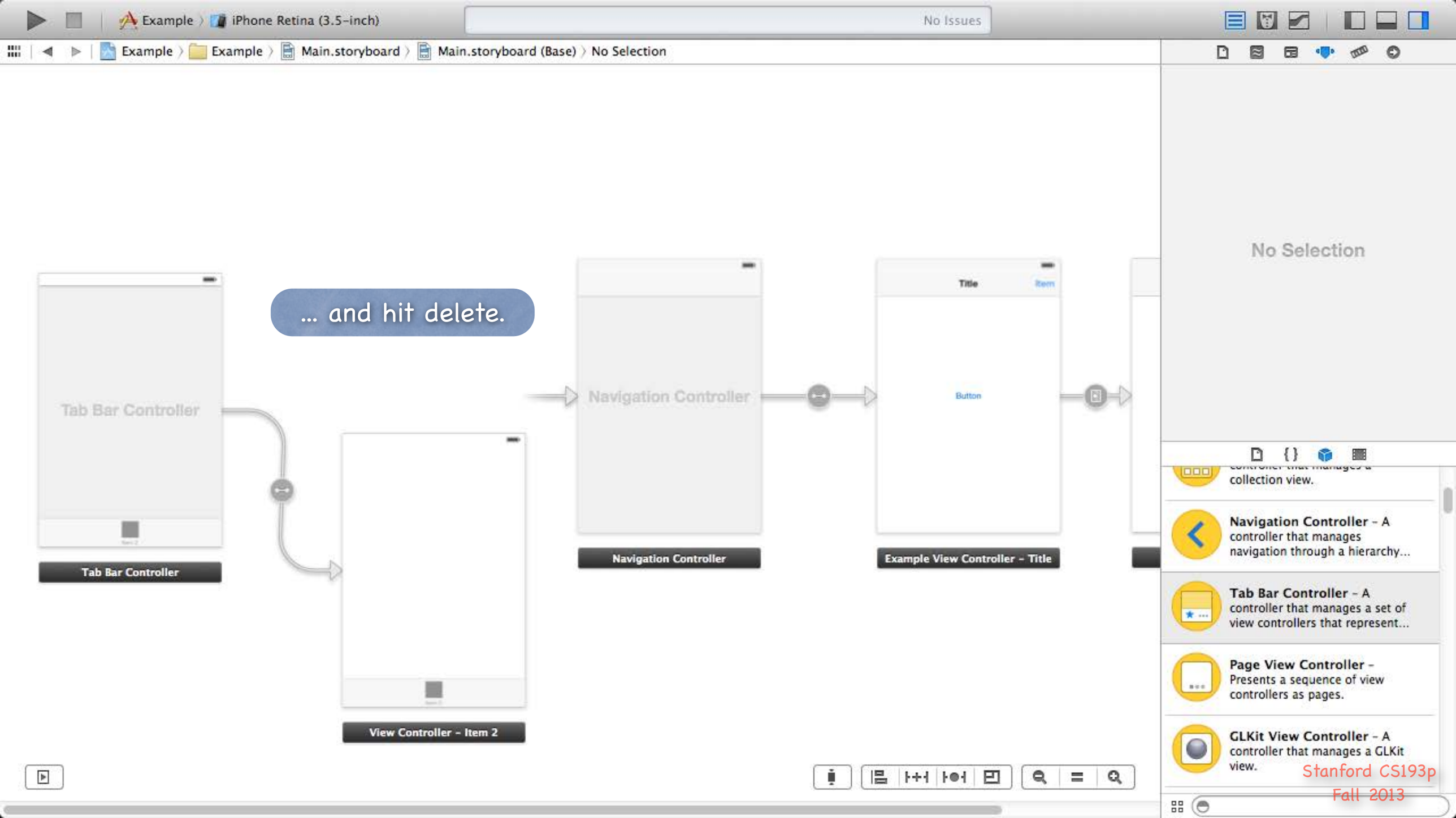

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
No Selection
Navigation Controller
Title
Item
Button
Navigation Controller
Example View Controller - Title
No Selection
collection view.
Navigation Controller - A controller that manages navigation through a hierarchy...
Tab Bar Controller - A controller that manages a set of view controllers that represent...
Page View Controller - Presents a sequence of view controllers as pages.
GLKit View Controller - A controller that manages a GLKit view.
You create a Tab Bar Controller by dragging it from the object palette.
Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
No Selection
You can drag it anywhere. After you drop it, you can reposition everything.
Tab Bar Controller
Navigation Controller
View Controller - Item 1
View Controller - Item 2
Title
Item
Button
Example View Controller - Title
View Controller
No Selection
collection view.
Navigation Controller - A controller that manages navigation through a hierarchy...
Tab Bar Controller - A controller that manages a set of view controllers that represent...
Page View Controller - Presents a sequence of view controllers as pages.
GLKit View Controller - A controller that manages a GLKit view.
Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller – Item 1 Scene
View Controller – Item 1
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Multiple Values
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Tab Bar Controller
View Controller – Item 1
View Controller – Item 2
Navigation Controller
Button
Navigation Controller
Example View Controller – Title
If things are a mess, you can double-click on the background of a storyboard to make everything smaller.
Or click here.
collection view.
Navigation Controller – A controller that manages navigation through a hierarchy...
Tab Bar Controller – A controller that manages a set of view controllers that represent...
Page View Controller – Presents a sequence of view controllers as pages.
GLKit View Controller – A controller that manages a GLKit view.
Stanford CS193p
Fall 2013

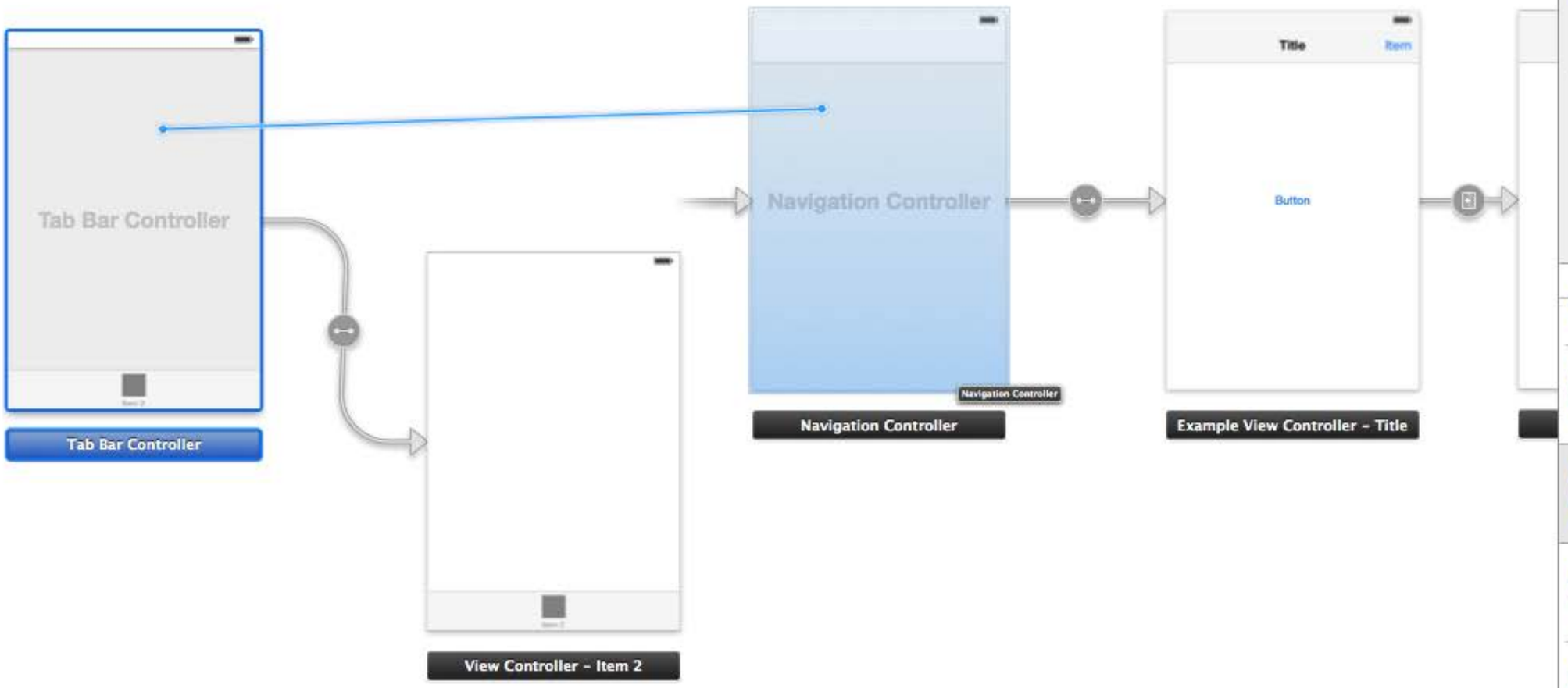

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller – Item 1 Scene
View Controller – Item 1
You can arrange the scenes in your storyboard any way you want.
Tab Bar Controller
Tab Bar Controller
View Controller – Item 1
View Controller – Item 2
Navigation Controller
Navigation Controller
Title
Item
Button
Example View Controller – Title
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
collection view.
Navigation Controller – A controller that manages navigation through a hierarchy...
Tab Bar Controller – A controller that manages a set of view controllers that represent...
Page View Controller – Presents a sequence of view controllers as pages.
GLKit View Controller – A controller that manages a GLKit view.
Stanford CS193p
Fall 2013

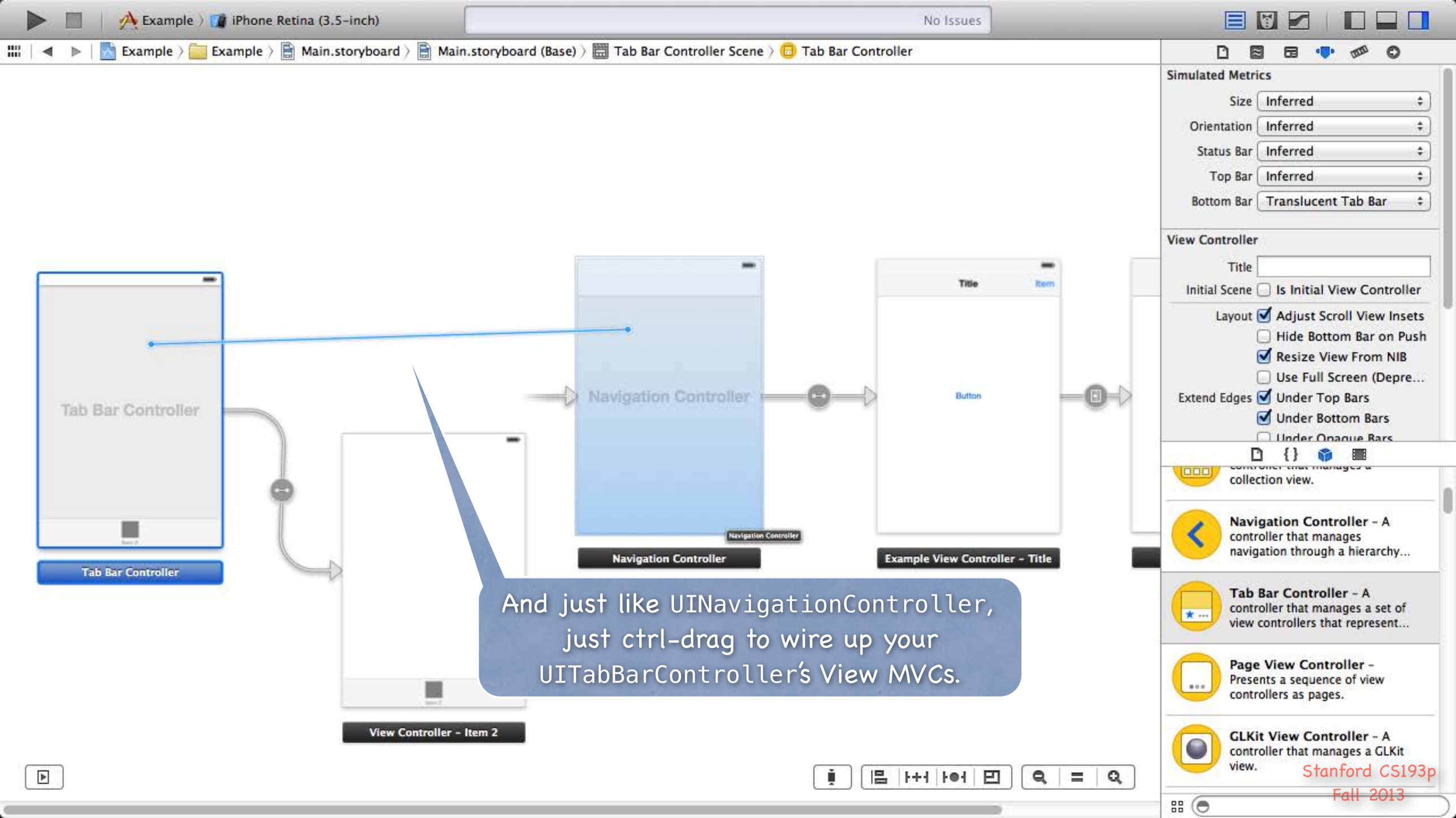

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller – Item 1 Scene
View Controller – Item 1
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
Resize View From NIB
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
When you drag a Tab Bar Controller into your storyboard, it comes with two "prefabbed" tabs. Often you don't want them.
Just click on an undesired scene's black bar ...
Tab Bar Controller
View Controller – Item 1
Navigation Controller
Button
Tab Bar Controller
Navigation Controller
Example View Controller – Title
View Controller – Item 2
collection view.
Navigation Controller – A controller that manages navigation through a hierarchy...
Tab Bar Controller – A controller that manages a set of view controllers that represent...
Page View Controller – Presents a sequence of view controllers as pages.
GLKit View Controller – A controller that manages a GLKit view.
Stanford CS193p
Fall 2013

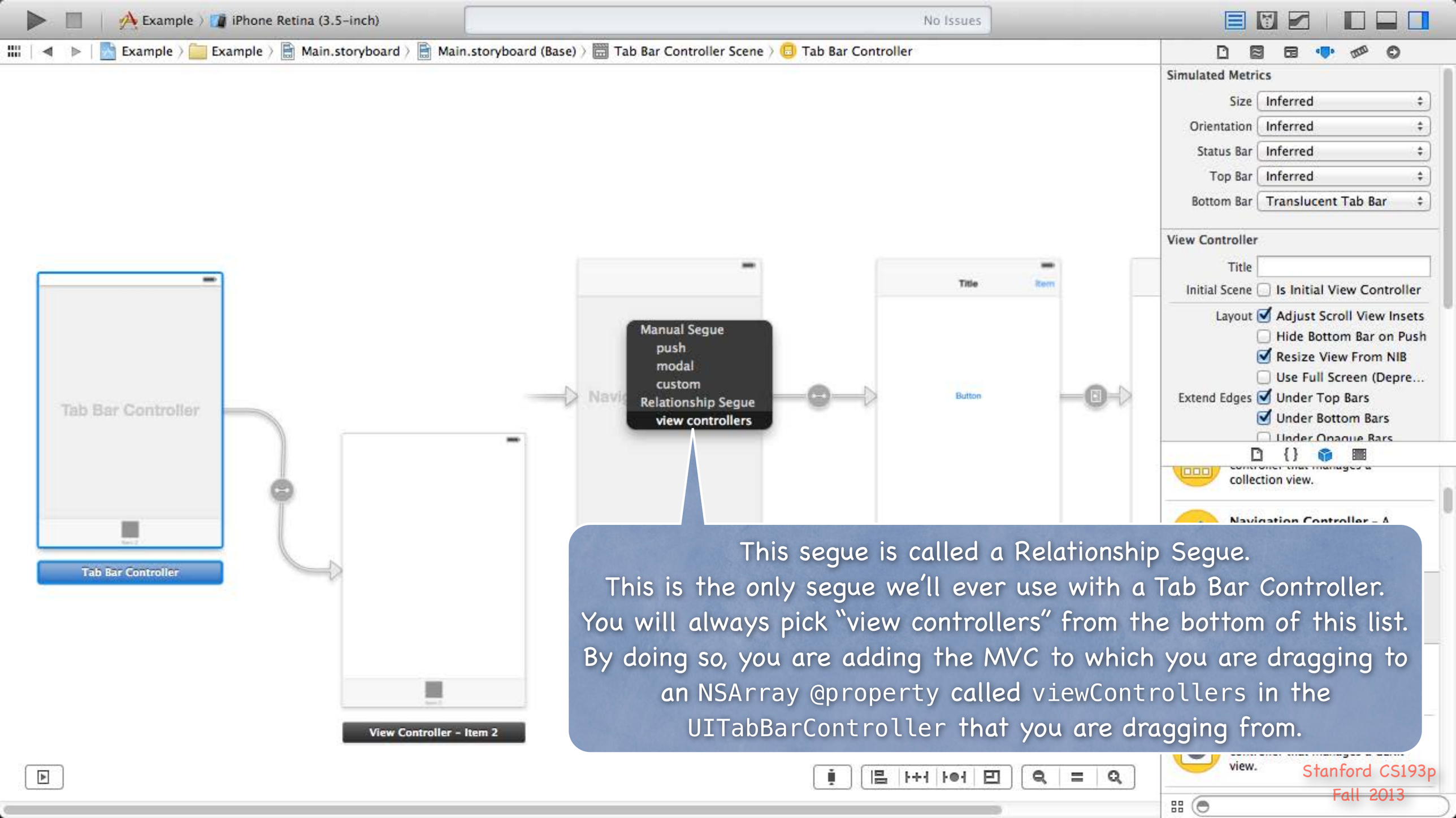

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
No Selection
... and hit delete.
Tab Bar Controller
Navigation Controller
Title
Item
Button
Tab Bar Controller
Navigation Controller
Example View Controller - Title
View Controller - Item 2
No Selection
collection view.
Navigation Controller - A controller that manages navigation through a hierarchy...
Tab Bar Controller - A controller that manages a set of view controllers that represent...
Page View Controller - Presents a sequence of view controllers as pages.
GLKit View Controller - A controller that manages a GLKit view.
Stanford CS193p
Fall 2013

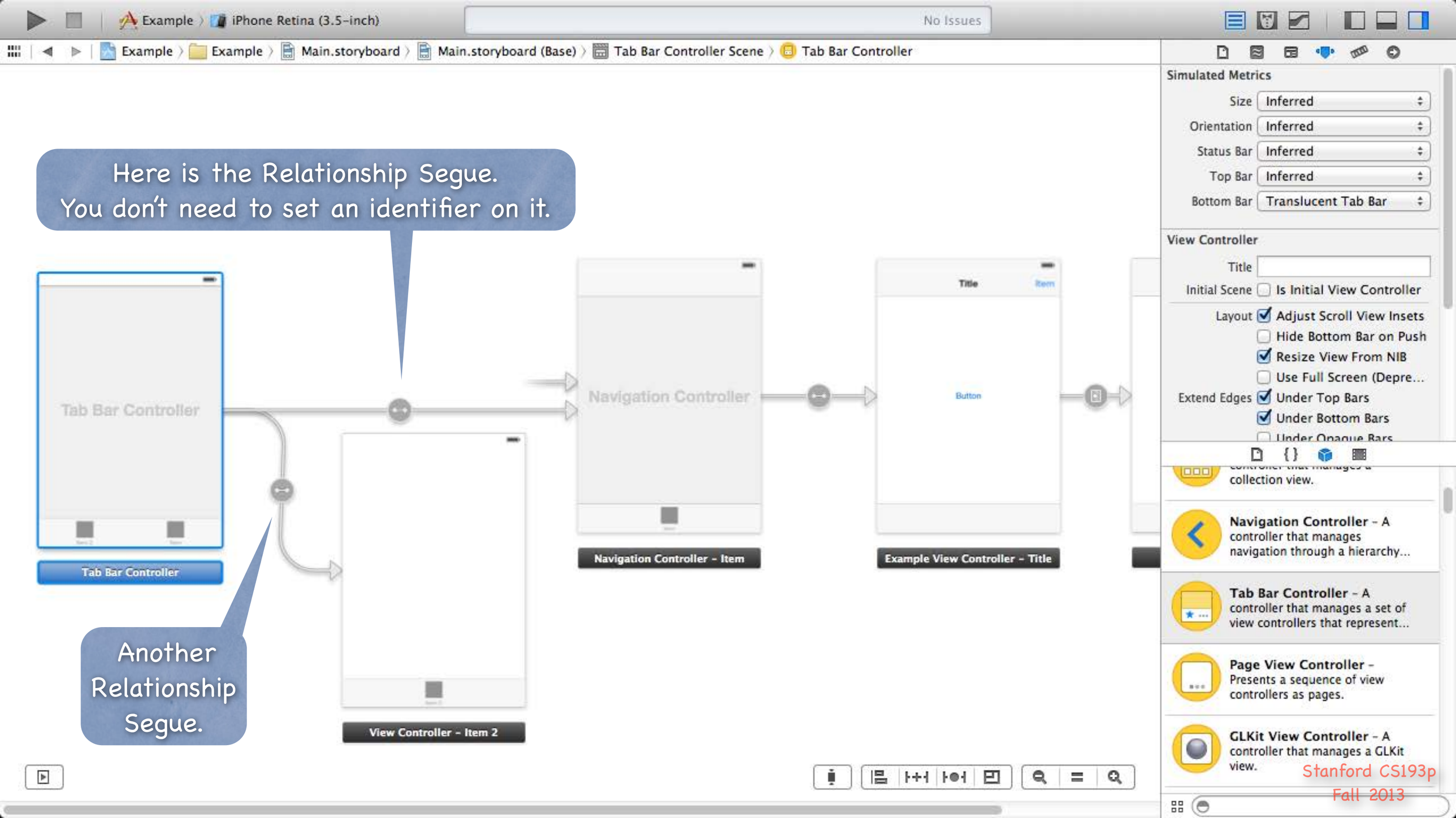
In the same way as a UINavigationController, a UITabBarController is itself the Controller of an MVC.

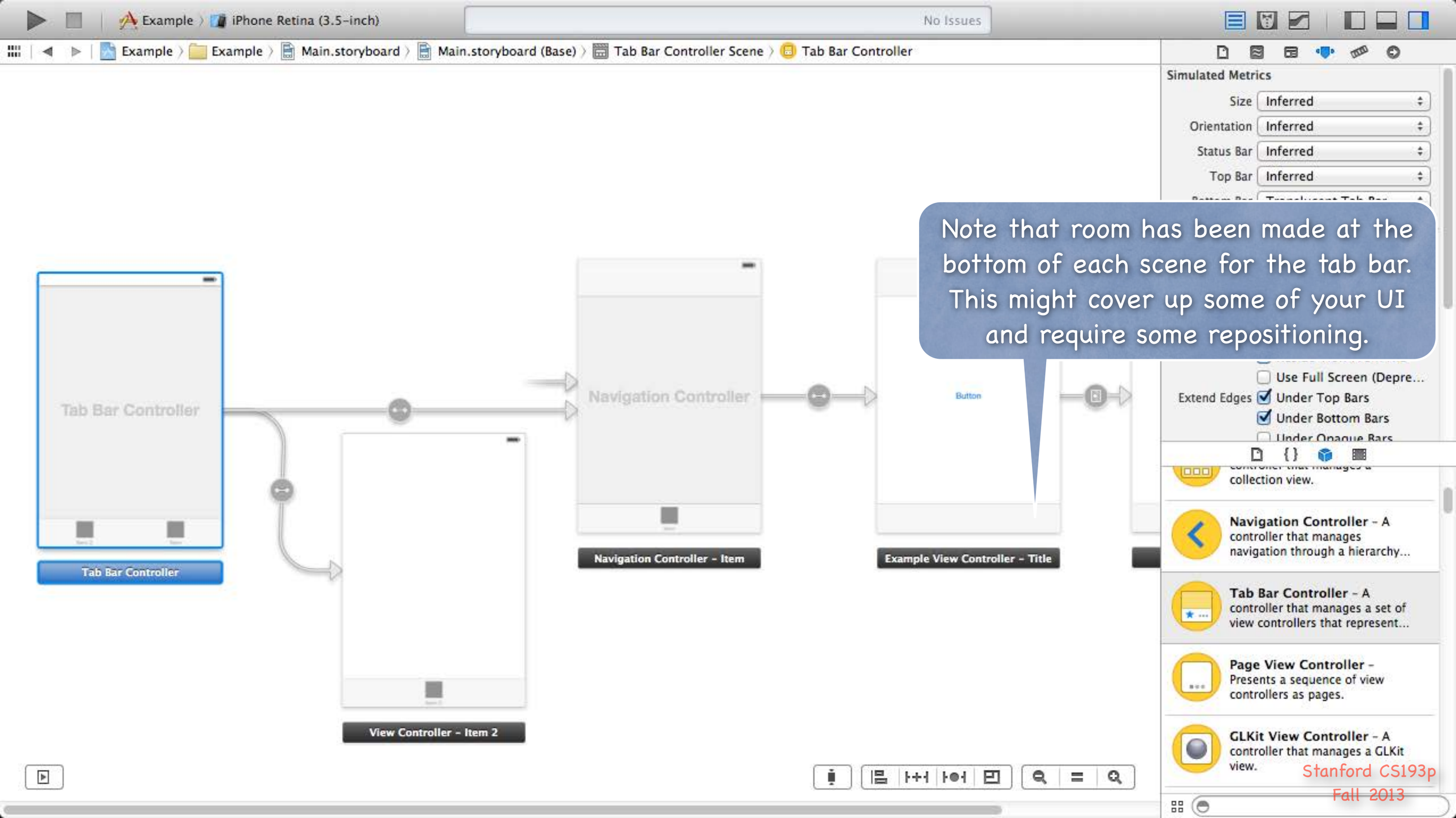
It's View consists of other MVCs.

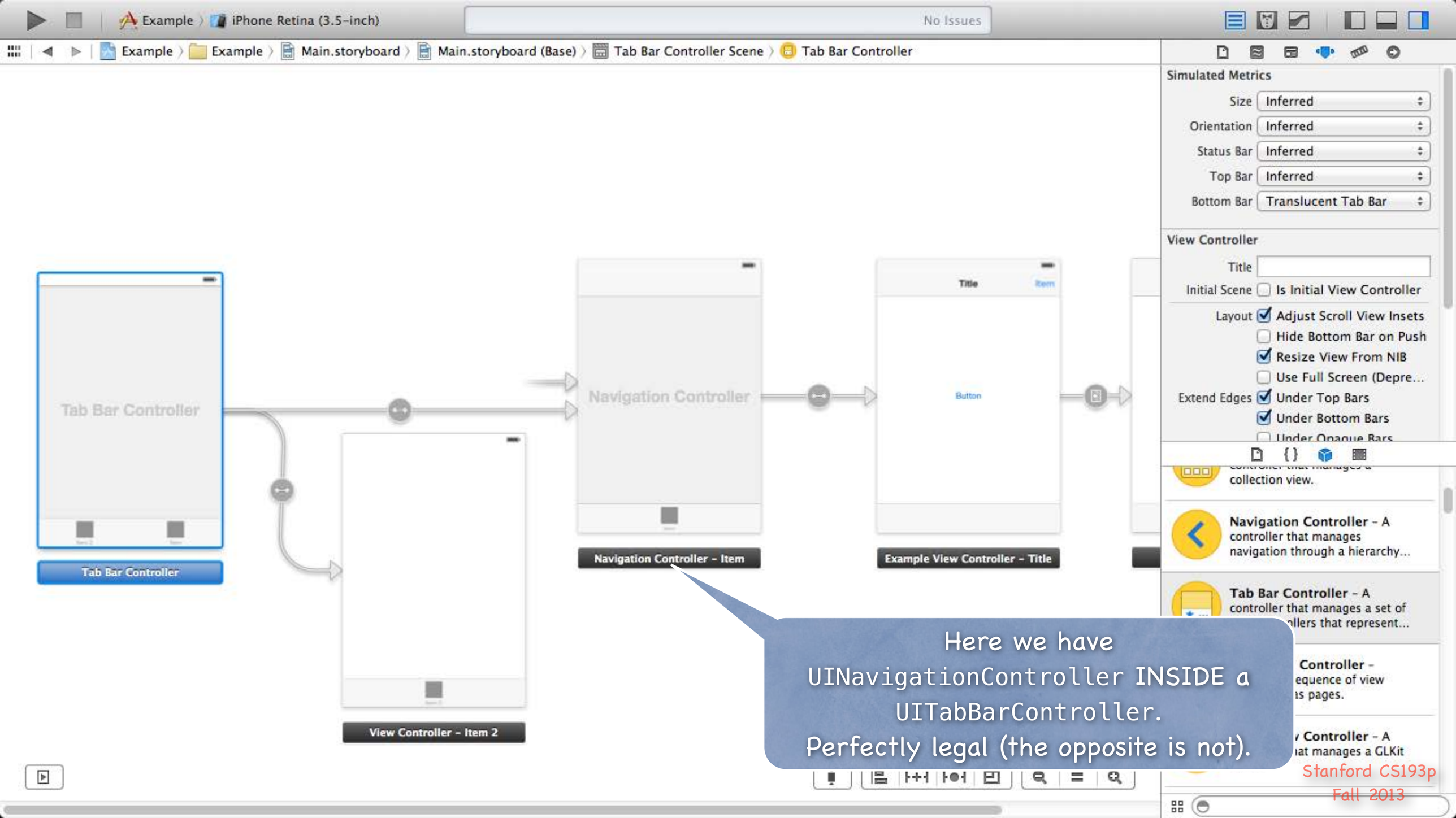


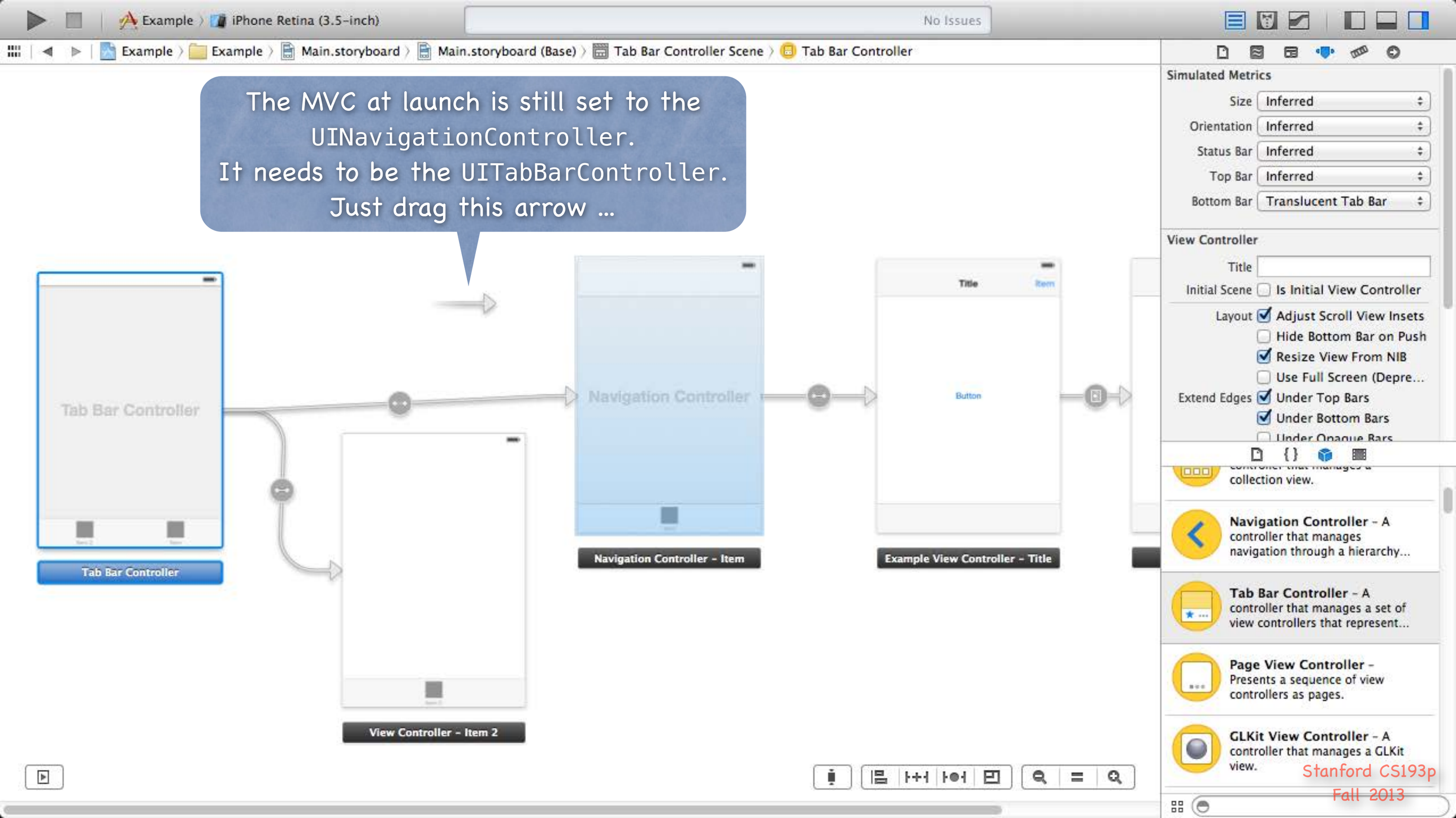

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Tab Bar Controller Scene
Tab Bar Controller
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Translucent Tab Bar
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Tab Bar Controller
Navigation Controller
Navigation Controller
Title
Item
Button
Example View Controller - Title
View Controller - Item 2
And just like UINavigationController, just ctrl-drag to wire up your UITabBarController's View MVCs.
collection view.
Navigation Controller - A controller that manages navigation through a hierarchy...
Tab Bar Controller - A controller that manages a set of view controllers that represent...
Page View Controller - Presents a sequence of view controllers as pages.
GLKit View Controller - A controller that manages a GLKit view.
Stanford CS193p
Fall 2013

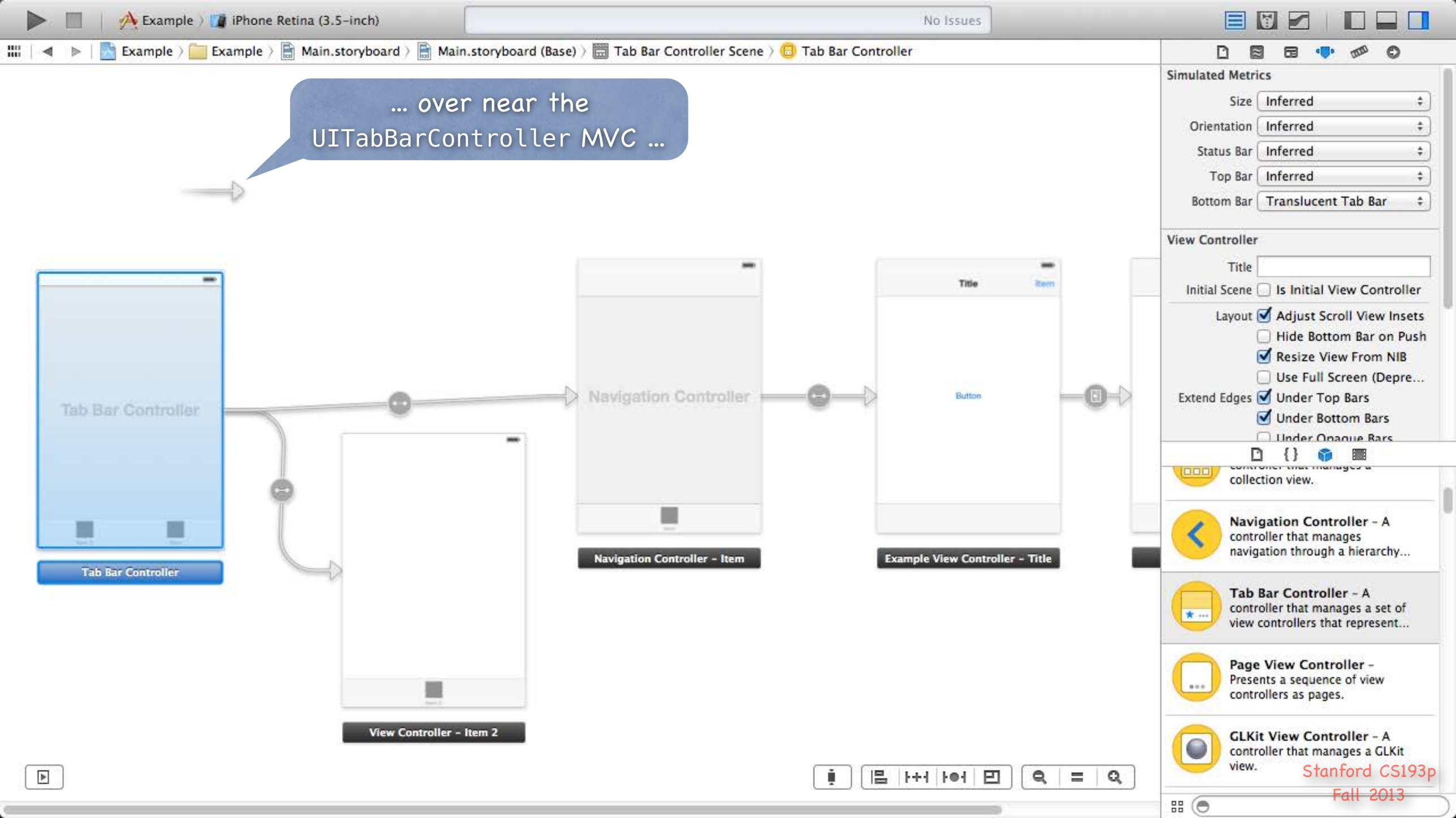

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Tab Bar Controller Scene
Tab Bar Controller
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Translucent Tab Bar
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Tab Bar Controller
Manual Segue
push
modal
custom
Relationship Segue
view controllers
Title
Item
Button
Tab Bar Controller
View Controller - Item 2
collection view.
view.
This segue is called a Relationship Segue.
This is the only segue we'll ever use with a Tab Bar Controller.
You will always pick "view controllers" from the bottom of this list.
By doing so, you are adding the MVC to which you are dragging to
an NSArray @property called viewControllers in the
UITabBarController that you are dragging from.
Stanford CS193p
Fall 2013

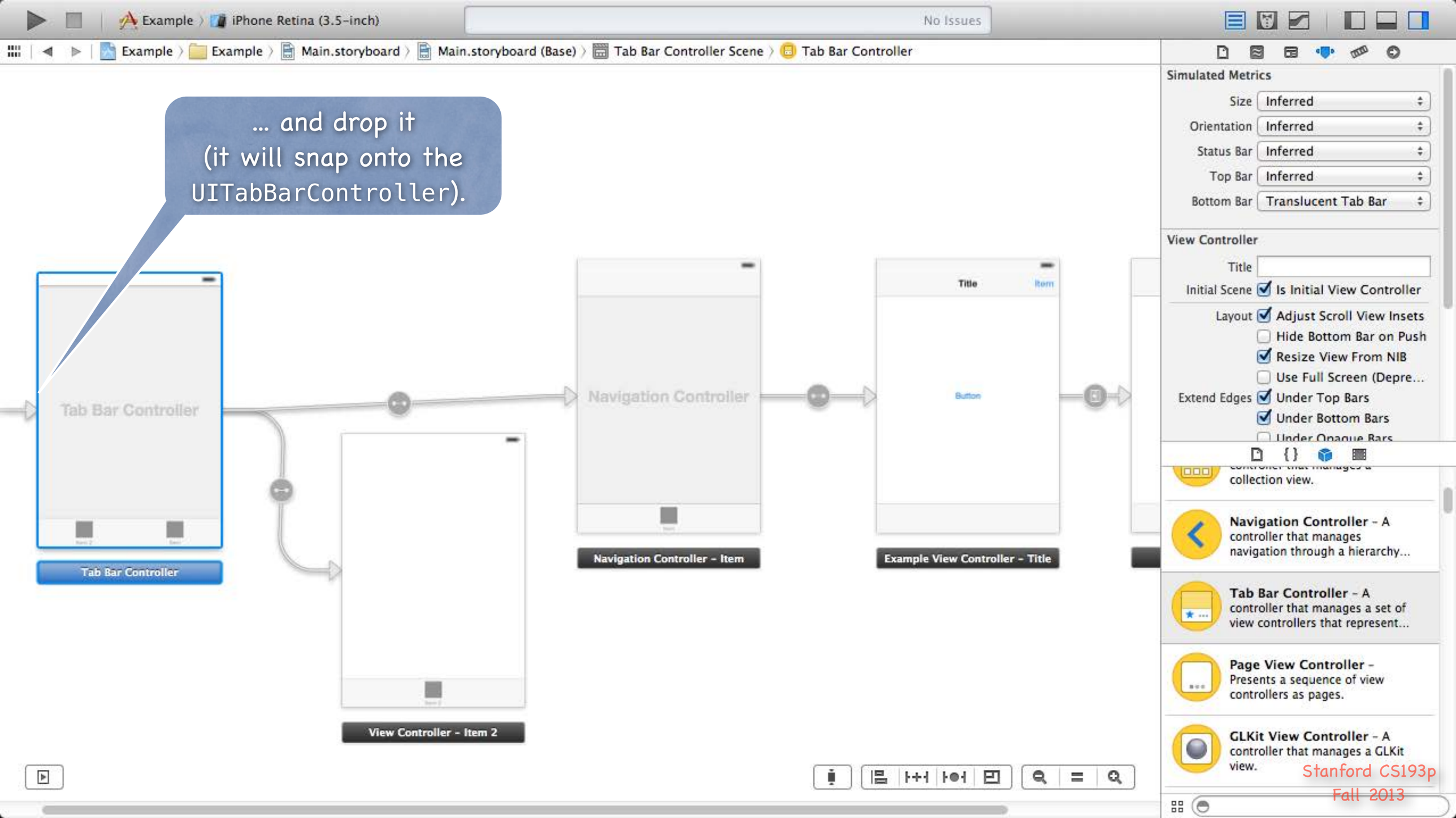

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Tab Bar Controller Scene
Tab Bar Controller
Here is the Relationship Segue.
You don't need to set an identifier on it.
Another Relationship Segue.
Tab Bar Controller
Tab Bar Controller
Navigation Controller
Navigation Controller - Item
Title
Item
Button
Example View Controller - Title
View Controller - Item 2
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Translucent Tab Bar
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
collection view.
Navigation Controller - A controller that manages navigation through a hierarchy...
Tab Bar Controller - A controller that manages a set of view controllers that represent...
Page View Controller - Presents a sequence of view controllers as pages.
GLKit View Controller - A controller that manages a GLKit view.
Stanford CS193p
Fall 2013

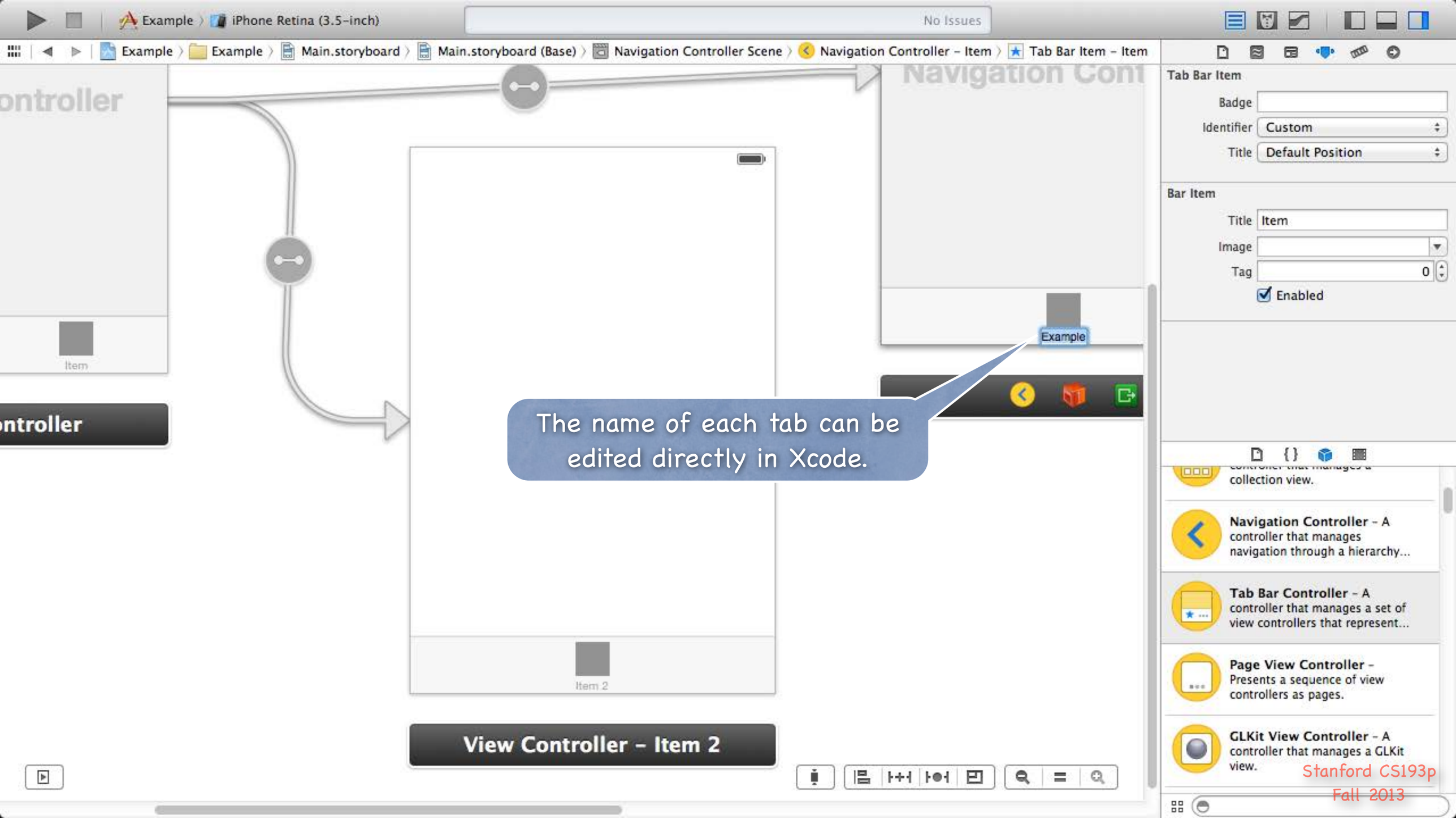

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Tab Bar Controller Scene
Tab Bar Controller
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Note that room has been made at the bottom of each scene for the tab bar. This might cover up some of your UI and require some repositioning.
Tab Bar Controller
Navigation Controller
Navigation Controller - Item
Button
Example View Controller - Title
View Controller - Item 2
collection view.
Navigation Controller - A controller that manages navigation through a hierarchy...
Tab Bar Controller - A controller that manages a set of view controllers that represent...
Page View Controller - Presents a sequence of view controllers as pages.
GLKit View Controller - A controller that manages a GLKit view.
Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Tab Bar Controller Scene
Tab Bar Controller
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Translucent Tab Bar
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Tab Bar Controller
Navigation Controller
Title
Item
Button
Tab Bar Controller
View Controller - Item 2
Navigation Controller - Item
Example View Controller - Title
collection view.
Navigation Controller - A
controller that manages
navigation through a hierarchy...
Tab Bar Controller - A
controller that manages a set of
Here we have
UINavigationController INSIDE a
UITabBarController.
Perfectly legal (the opposite is not).
Stanford CS193p
Fall 2013

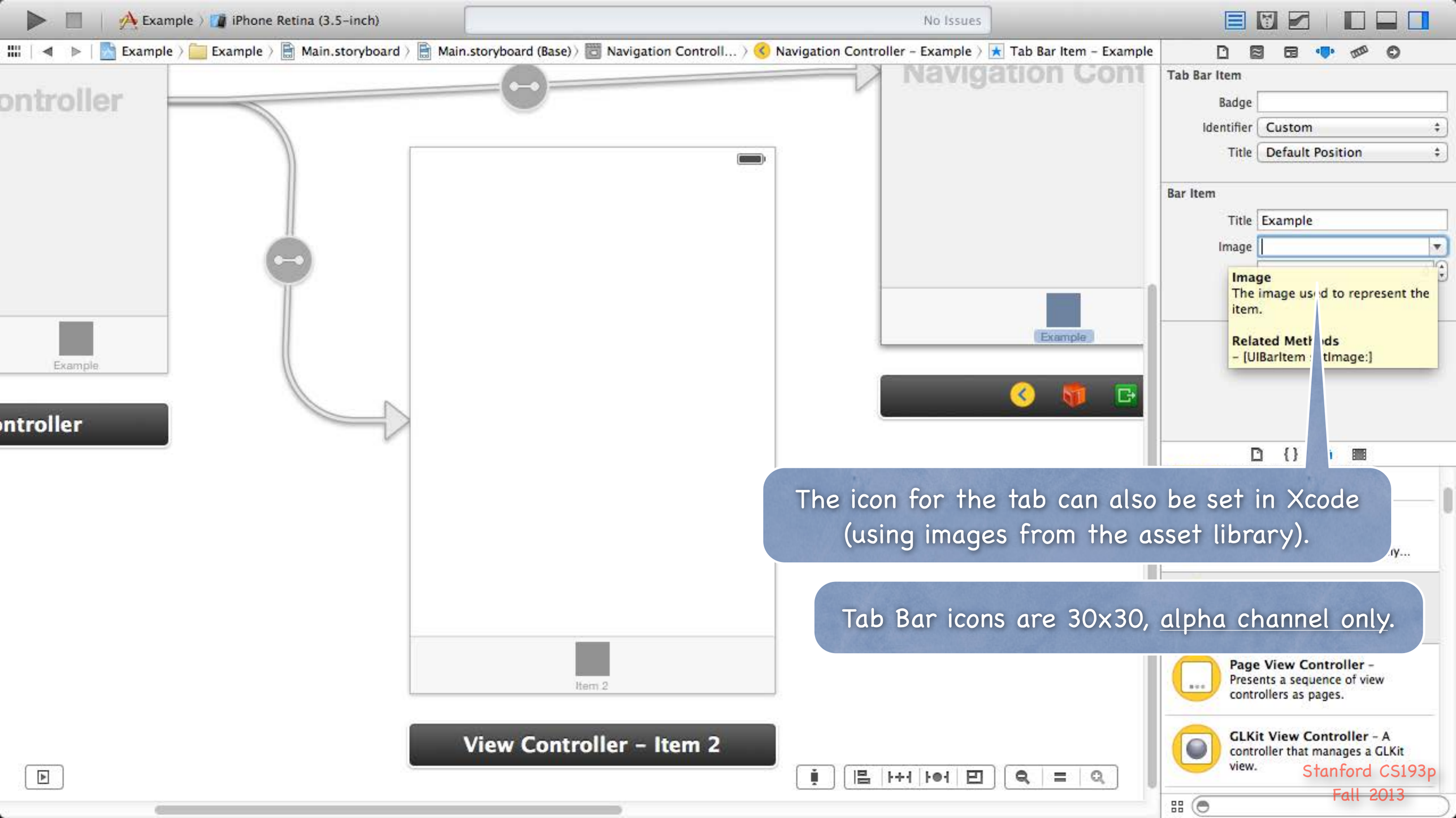

Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Tab Bar Controller Scene
Tab Bar Controller
The MVC at launch is still set to the UINavigationController.
It needs to be the UITabBarController.
Just drag this arrow ...
Tab Bar Controller
Tab Bar Controller
Navigation Controller
Navigation Controller - Item
Title
Item
Button
Example View Controller - Title
View Controller - Item 2
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Translucent Tab Bar
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges Under Top Bars
Under Bottom Bars
collection view.
Navigation Controller - A controller that manages navigation through a hierarchy...
Tab Bar Controller - A controller that manages a set of view controllers that represent...
Page View Controller - Presents a sequence of view controllers as pages.
GLKit View Controller - A controller that manages a GLKit view.
Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Tab Bar Controller Scene
Tab Bar Controller
… over near the
UITabBarController MVC …
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Translucent Tab Bar
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre…
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Tab Bar Controller
Navigation Controller
Title
Item
Button
Tab Bar Controller
Navigation Controller – Item
Example View Controller – Title
View Controller – Item 2
collection view.
Navigation Controller – A
controller that manages
navigation through a hierarchy…
Tab Bar Controller – A
controller that manages a set of
view controllers that represent…
Page View Controller –
Presents a sequence of view
controllers as pages.
GLKit View Controller – A
controller that manages a GLKit
view.
Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Tab Bar Controller Scene
Tab Bar Controller
... and drop it
(it will snap onto the
UITabBarController).
Tab Bar Controller
Navigation Controller
Title
Item
Button
Tab Bar Controller
Navigation Controller - Item
Example View Controller - Title
View Controller - Item 2
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Translucent Tab Bar
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Depre...
Extend Edges
Under Top Bars
Under Bottom Bars
collection view.
Navigation Controller - A
controller that manages
navigation through a hierarchy...
Tab Bar Controller - A
controller that manages a set of
view controllers that represent...
Page View Controller -
Presents a sequence of view
controllers as pages.
GLKit View Controller - A
controller that manages a GLKit
view.
Stanford CS193p
Fall 2013



Stanford CS193p
Fall 2013


Example
iPhone Retina (3.5-inch)
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
Navigation Controll...
Navigation Controller – Example
Tab Bar Item – Example
Tab Bar Item
Badge
Identifier
Custom
Title
Default Position
Bar Item
Title
Example
Image
Image
The image used to represent the item.
Related Methods
– [UIBarItem setImage:]
Example
Item 2
View Controller – Item 2
The icon for the tab can also be set in Xcode (using images from the asset library).
Tab Bar icons are 30x30, alpha channel only.
Page View Controller – Presents a sequence of view controllers as pages.
GLKit View Controller – A controller that manages a GLKit view.
Stanford CS193p
Fall 2013

Coming Up

- Friday
  No Section
- Next couple of weeks ...
  Drawing in your own custom View class
  Gestures
  Autolayout
  Animation

No Lecture next Monday!

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

- Views
  How to draw custom stuff on screen.
- Gestures
  How to react to user's touch gestures.
- Demo
  SuperCard

Views

- A view (i.e. `UIView` subclass) represents a rectangular area
  Defines a coordinate space
- Draws and handles events in that rectangle
- Hierarchical
  A view has only one superview `- (UIView *)superview`
  But can have many (or zero) subviews `- (NSArray *)subviews`
  Subview order (in `subviews` array) matters: those later in the array are on top of those earlier
  A view can clip its `subviews` to its `bounds` or not (switch for this in Xcode, or method in `UIView`).
- `UIWindow`
  The `UIView` at the top of the view hierarchy
  Only have one `UIWindow` (generally) in an iOS application
  It's all about views, not windows

Views

- The hierarchy is most often constructed in Xcode graphically
  Even custom views are often added to the view hierarchy using Xcode (more on this later).
- But it can be done in code as well
  ```
  - (void)addSubview:(UIView *)aView;   // sent to aView's (soon to be) superview
  - (void)removeFromSuperview;          // sent to the view that is being removed
  ```
- The top of this hierarchy for your MVC is the `@property view`!
  `UIViewController`'s `@property (strong, nonatomic) UIView *view`
  It is critical to understand what this very simple `@property` is!
  This is the view whose `bounds` will be changed when autorotation happens, for example.
  This is the view you would programmatically add `subviews` to.
  All your MVC's View's `UIView`'s eventually have this `view` as their parent (it's at the top).
  It is automatically hooked up for you when you drag out a View Controller in Xcode.

Initializing a UIView

- Yes, you might want to override UIView's designated initializer
  More common than overriding UIViewController's designated initializer (but still rare).
- But you will also want to set up stuff in awakeFromNib
  This is because initWithFrame: is NOT called for a UIView coming out of a storyboard!
  But awakeFromNib is. Same as we talked about with UIViewController.
  It's called "awakeFromNib" for historical reasons.
- Typical code …

```
- (void)setup { … }
- (void)awakeFromNib { [self setup]; }
- (id)initWithFrame:(CGRect)aRect
{
    self = [super initWithFrame:aRect];
    [self setup];
    return self;
}
```

View Coordinates

- **CGFloat**
  Just a floating point number (depends on 64-bit or not), but we always use it for graphics.
- **CGPoint**
  C struct with two CGFloats in it: x and y.

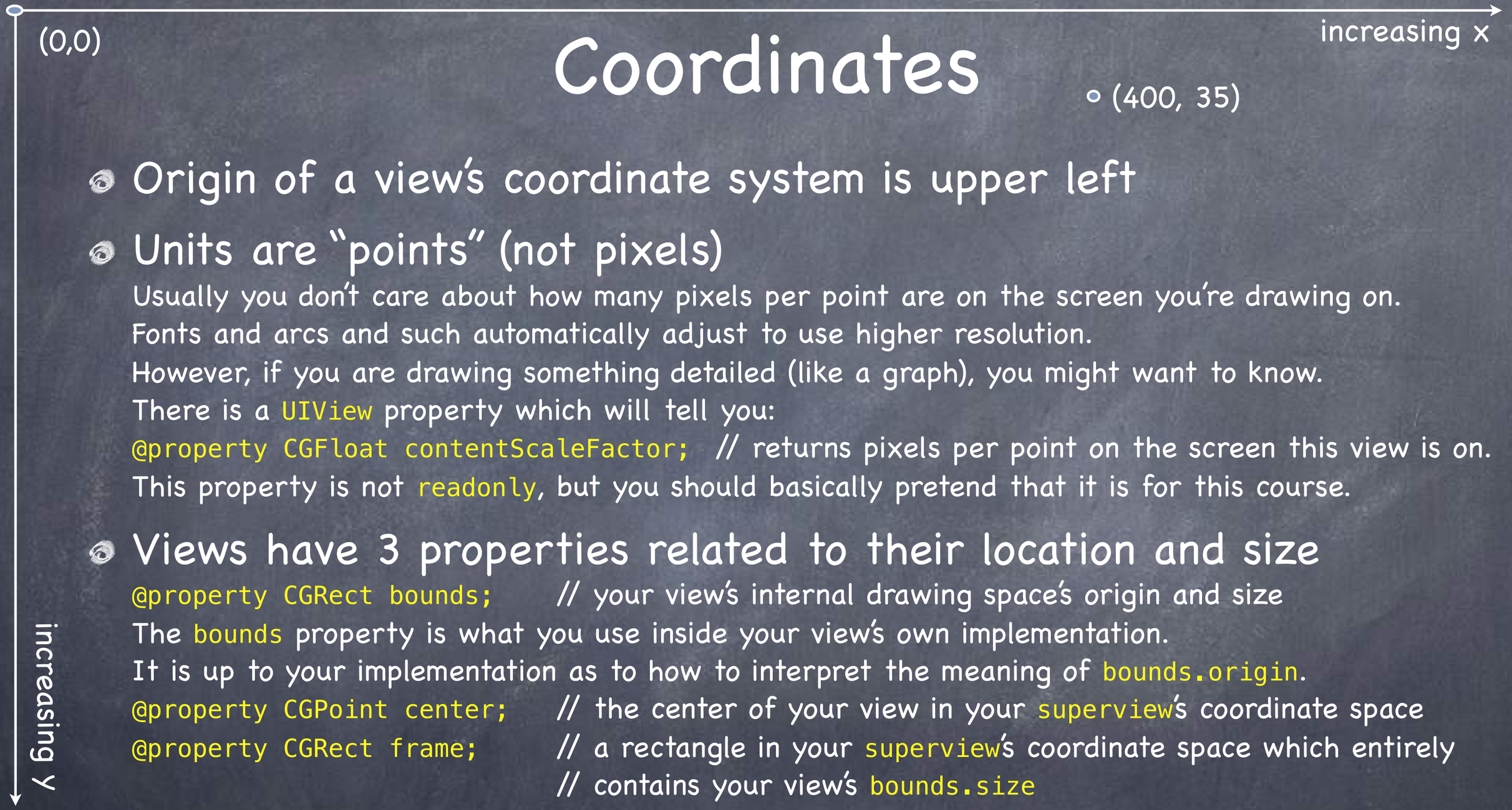
```
CGPoint p = CGPointMake(34.5, 22.0);
p.x += 20;  // move right by 20 points
```

- **CGSize**
  C struct with two CGFloats in it: width and height.

```
CGSize s = CGSizeMake(100.0, 200.0);
s.height += 50;  // make the size 50 points taller
```

- **CGRect**
  C struct with a CGPoint origin and a CGSize size.

```
CGRect aRect = CGRectMake(45.0, 75.5, 300, 500);
aRect.size.height += 45;  // make the rectangle 45 points taller
aRect.origin.x += 30;     // move the rectangle to the right 30 points
```



Coordinates

- Origin of a view's coordinate system is upper left
- Units are "points" (not pixels)

  Usually you don't care about how many pixels per point are on the screen you're drawing on.
  Fonts and arcs and such automatically adjust to use higher resolution.
  However, if you are drawing something detailed (like a graph), you might want to know.
  There is a `UIView` property which will tell you:

```
@property CGFloat contentScaleFactor;  // returns pixels per point on the screen this view is on.
```

  This property is not `readonly`, but you should basically pretend that it is for this course.

- Views have 3 properties related to their location and size

```
@property CGRect bounds;     // your view's internal drawing space's origin and size
```

  The `bounds` property is what you use inside your view's own implementation.
  It is up to your implementation as to how to interpret the meaning of `bounds.origin`.

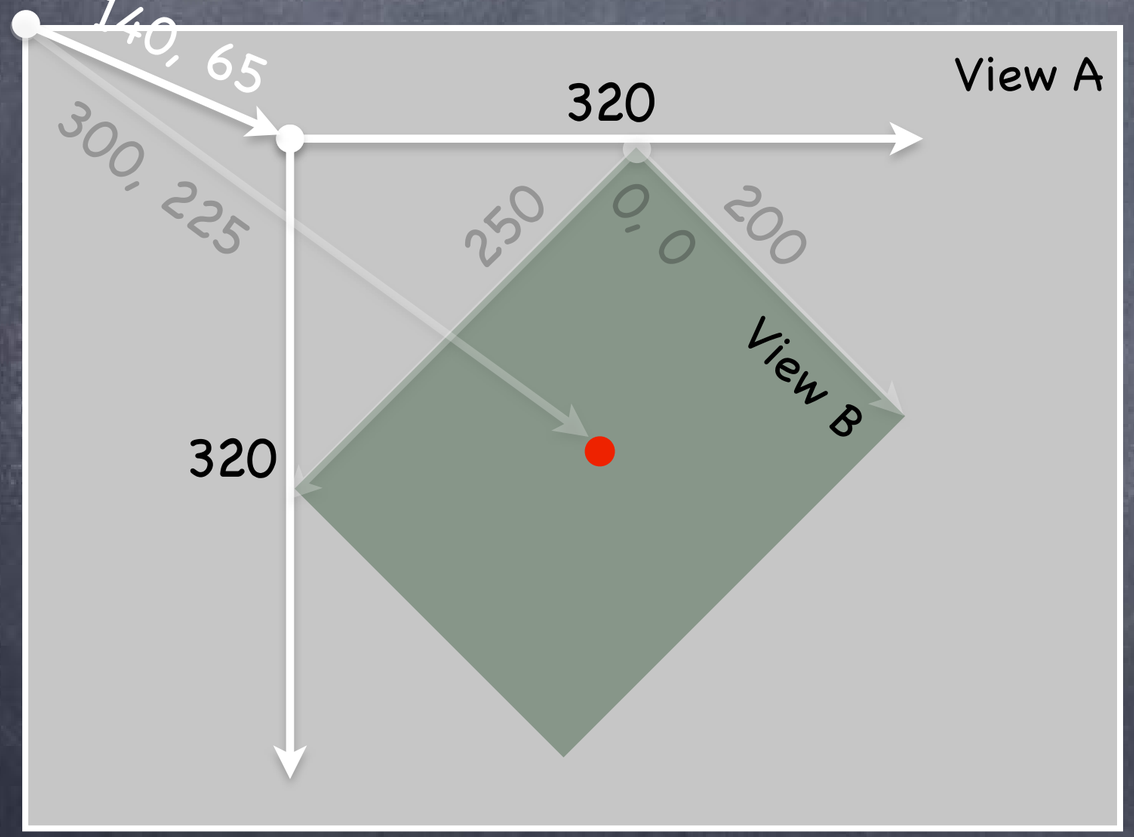
```
@property CGPoint center;    // the center of your view in your superview's coordinate space
@property CGRect frame;      // a rectangle in your superview's coordinate space which entirely
                             // contains your view's bounds.size
```

Coordinates

- Use `frame` and `center` to position the view in the hierarchy

  These are used by `superviews`, never inside your `UIView` subclass's implementation.

  You might think `frame.size` is always equal to `bounds.size`, but you'd be wrong …



Because views can be rotated
(and scaled and translated too).

View B's `bounds` = `((0,0),(200,250))`

View B's `frame` = `((140,65),(320,320))`

View B's `center` = `(300,225)`

View B's middle in its own coordinate space is
`(bound.size.width/2+bounds.origin.x,`
`bounds.size.height/2+bounds.origin.y)`
which is `(100,125)` in this case.

Views are rarely rotated, but don't
misuse `frame` or `center` by assuming that.

Creating Views

- Most often you create views in Xcode

  Of course, Xcode's palette knows nothing about a custom view class you might create.
  So you drag out a generic `UIView` from the palette and use the Identity Inspector
  to change the class of the `UIView` to your custom class (demo of this later).

- How do you create a `UIView` in code (i.e. not in Xcode)?

  Just use `alloc` and `initWithFrame:` (`UIView`'s designated initializer).
  Can also use `init` (`frame` will be `CGRectZero`).

- Example

```
CGRect labelRect = CGRectMake(20, 20, 50, 30);
UILabel *label = [[UILabel alloc] initWithFrame:labelRect];
label.text = @"Hello!";
[self.view addSubview:label];  // Note self.view!
```

Custom Views

- When would I want to create my own `UIView` subclass?
  I want to do some custom drawing on screen.
  I need to handle touch events in a special way (i.e. different than a button or slider does)
  We'll talk about handling touch events in a bit. First we'll focus on drawing.
- Drawing is easy … create a `UIView` subclass & override 1 method
  `- (void)drawRect:(CGRect)aRect;`
  You can optimize by not drawing outside of `aRect` if you want (but not required).
- NEVER call `drawRect:`!! EVER! Or else!
  Instead, let iOS know that your view's visual is out of date with one of these `UIView` methods:
  `- (void)setNeedsDisplay;`
  `- (void)setNeedsDisplayInRect:(CGRect)aRect;`
  It will then set everything up and call `drawRect:` for you at an appropriate time.
  Obviously, the second version will call your `drawRect:` with only rectangles that need updates.

Custom Views

- So how do I implement my `drawRect:`?
  - Use the Core Graphics framework directly (a C API, not object-oriented).
  - Or we can use the object-oriented `UIBezierPath` class (we'll do it this way).
- Core Graphics Concepts
  - Get a context to draw into (iOS will prepare one each time your `drawRect:` is called)
  - Create paths (out of lines, arcs, etc.)
  - Set colors, fonts, textures, linewidths, linecaps, etc.
  - Stroke or fill the above-created paths
- `UIBezierPath`
  - Do all of the above, but capture it with an object.
  - Then ask the object to stroke or fill what you've created.

Context

- The context determines where your drawing goes
  - Screen (the only one we're going to talk about today)
  - Offscreen Bitmap
  - PDF
  - Printer
- For normal drawing, `UIKit` sets up the current context for you
  - But it is only valid during that particular call to `drawRect:`.
  - A new one is set up for you each time `drawRect:` is called.
  - So never cache the current graphics context in `drawRect:` to use later!
- How to get this magic context?
  - `UIBezierPath` draws into the current context, so you don't need to get it if using that.
  - But if you're calling Core Graphics C functions directly, you'll need it (it's an argument to them).
  - Call the following C function inside your `drawRect:` method to get the current graphics context …

```
CGContextRef context = UIGraphicsGetCurrentContext();
```

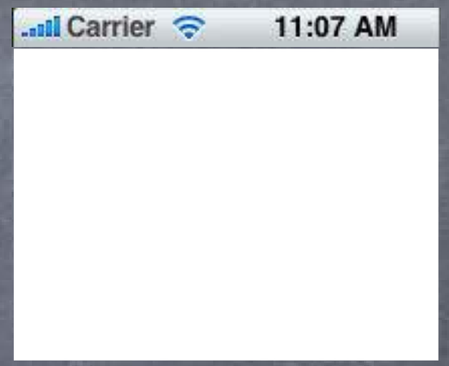
Define a Path

- Begin the path

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

- Move around, add lines or arcs to the path

```
[path moveToPoint:CGPointMake(75, 10)];
```



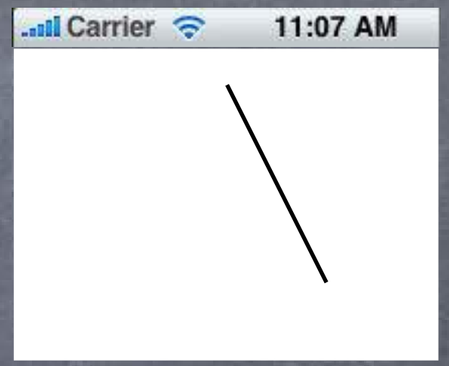
Define a Path

- Begin the path

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

- Move around, add lines or arcs to the path

```
[path moveToPoint:CGPointMake(75, 10)];
[path addLineToPoint:CGPointMake(160, 150)];
```



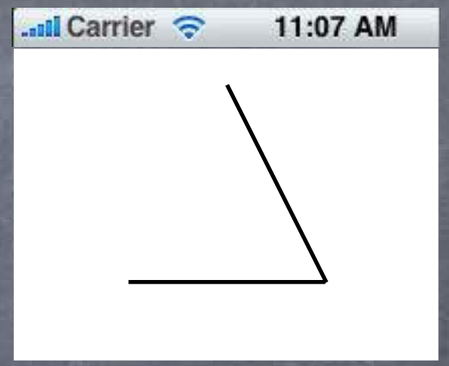
Define a Path

- Begin the path

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

- Move around, add lines or arcs to the path

```
[path moveToPoint:CGPointMake(75, 10)];
[path addLineToPoint:CGPointMake(160, 150)];
[path addLineToPoint:CGPointMake(10, 150]);
```



Define a Path

- Begin the path

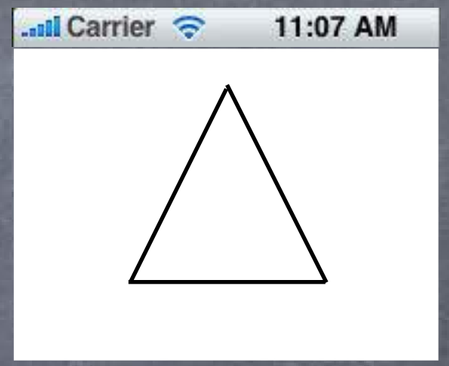
```
UIBezierPath *path = [[UIBezierPath alloc] init];
```

- Move around, add lines or arcs to the path

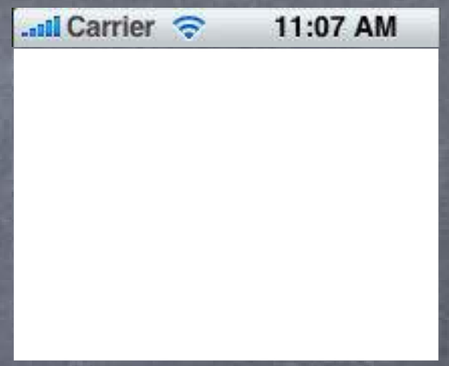
```
[path moveToPoint:CGPointMake(75, 10)];
[path addLineToPoint:CGPointMake(160, 150)];
[path addLineToPoint:CGPointMake(10, 150]);
```

- Close the path (connects the last point back to the first)

```
[path closePath];    // not strictly required but triangle won't have all 3 sides otherwise
```



Define a Path



- Begin the path

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

- Move around, add lines or arcs to the path

```
[path moveToPoint:CGPointMake(75, 10)];
[path addLineToPoint:CGPointMake(160, 150)];
[path addLineToPoint:CGPointMake(10, 150]);
```

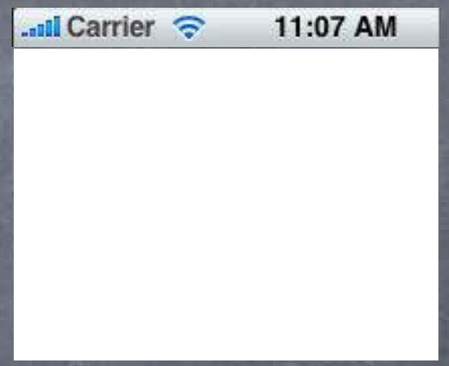
- Close the path (connects the last point back to the first)

```
[path closePath];    // not strictly required but triangle won't have all 3 sides otherwise
```

- Now that the path has been created, we can stroke/fill it

  Actually, nothing has been drawn yet, we've just created the `UIBezierPath`.

Define a Path



- Begin the path

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

- Move around, add lines or arcs to the path

```
[path moveToPoint:CGPointMake(75, 10)];
[path addLineToPoint:CGPointMake(160, 150)];
[path addLineToPoint:CGPointMake(10, 150]);
```

- Close the path (connects the last point back to the first)

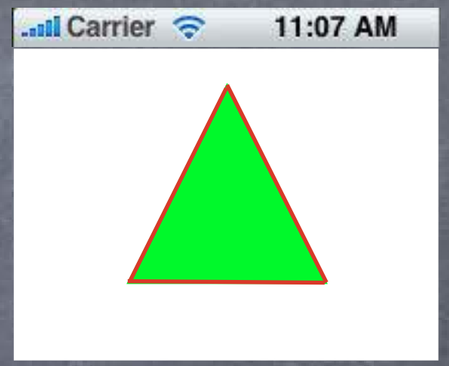
```
[path closePath];    // not strictly required but triangle won't have all 3 sides otherwise
```

- Now that the path has been created, we can stroke/fill it

Actually, nothing has been drawn yet, we've just created the UIBezierPath.

```
[[UIColor greenColor] setFill];
[[UIColor redColor] setStroke];
```

Define a Path



- Begin the path

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

- Move around, add lines or arcs to the path

```
[path moveToPoint:CGPointMake(75, 10)];
[path addLineToPoint:CGPointMake(160, 150)];
[path addLineToPoint:CGPointMake(10, 150]);
```

- Close the path (connects the last point back to the first)

```
[path closePath];    // not strictly required but triangle won't have all 3 sides otherwise
```

- Now that the path has been created, we can stroke/fill it

Actually, nothing has been drawn yet, we've just created the UIBezierPath.

```
[[UIColor greenColor] setFill];
[[UIColor redColor] setStroke];
[path fill]; [path stroke];
```

Graphics State

- Can also set graphics state

  e.g. `path.lineWidth = 2.0;` // line width in points (not pixels)

- And draw rounded rects, ovals, etc.

```
UIBezierPath *roundedRect = [UIBezierPath bezierPathWithRoundedRect:(CGRect)bounds
                                                       cornerRadius:(CGFloat)radius];
```

  Note: the "casts" in the arguments are just to let you know the types (i.e. they're not required).

```
UIBezierPath *oval = [UIBezierPath bezierPathWithOvalInRect:(CGRect)bounds];
[roundedRect stroke];
[oval fill];
```

- You can use a UIBezierPath to "clip" your drawing

```
[roundedRect addClip]; // this would clip all drawing to be inside the roundedRect
```

Graphics State

- Drawing with transparency in `UIView`

  You know that `UIColor`s can have `alpha`.
  This is how you can draw with transparency in your `drawRect:`.

  `UIView` also has a `backgroundColor` property which can be set to transparent values.

  Be sure to set `@property BOOL opaque` to `NO` in a view which is partially transparent.
  If you don't, results are unpredictable (this is a performance optimization property, by the way).

  `UIView`'s `@property CGFloat alpha` can make the entire view partially transparent.
  (you might use this to your advantage in your homework to show a "disabled" custom view)

View Transparency

- What happens when views overlap?
  As mentioned before, `subviews` list order determine's who's in front
  Lower ones (earlier in `subviews` array) can "show through" transparent views on top of them
- Default drawing is opaque
  Transparency is not cheap (performance-wise)
- Also, you can hide a view completely by setting `hidden` property

```
@property (nonatomic) BOOL hidden;
myView.hidden = YES;        // view will not be on screen and will not handle events
```

  This is not as uncommon as you might think
  On a small screen, keeping it de-cluttered by hiding currently unusable views make sense.
  Also this can be used to swap two (or more) views in and out depending on state.

Graphics State

- Special considerations for defining drawing "subroutines"

  What if you wanted to have a utility method that draws something?
  You don't want that utility method to mess up the graphics state of the calling method.
  Use save and restore context functions.

```
- (void)drawGreenCircle:(CGContextRef)ctxt {
    CGContextSaveGState(ctxt);
    [[UIColor greenColor] setFill];
    // draw my circle
    CGContextRestoreGState(ctxt);
}
- (void)drawRect:(CGRect)aRect {
    CGContextRef context = UIGraphicsGetCurrentContext();
    [[UIColor redColor] setFill];
    // do some stuff
    [self drawGreenCircle:context];
    // do more stuff and expect fill color to be red
}
```

Drawing Text

- We can use a `UILabel` as a subview to draw text in our view
  But there are certainly occasions where we want to draw text in our `drawRect:`.
- To draw in `drawRect:`, use `NSAttributedString`

```
NSAttributedString *text = …;
[text drawAtPoint:(CGPoint)p];  // NSAttributedString instance method added by UIKit
```

How much space will a piece of text will take up when drawn?

```
CGSize textSize = [text size];  // another UIKit NSAttributedString instance method
```

You might be disturbed that there are drawing methods in `Foundation` (a non-UI framework!).
These `NSAttributedString` methods are defined in `UIKit` via a mechanism called categories.
(so are the names of the attributes that define UI stuff (e.g. `NSFontAttributeName`)).
Categories are an Objective-C way to add methods to an existing class without subclassing.
We'll cover how (and when) to use this a bit later in this course.

Drawing Images

- UIImageView is like UILabel for images
  But again, occasionally you want to draw an image in your drawRect:.
- Create a UIImage object from a file in your Resources folder
  ```
  UIImage *image = [UIImage imageNamed:@"foo.jpg"]; // you did this in Matchismo
  ```
- Or create one from a named file or from raw data
  (of course, we haven't talked about the file system yet, but ...)
  ```
  UIImage *image = [[UIImage alloc] initWithContentsOfFile:(NSString *)fullPath];
  UIImage *image = [[UIImage alloc] initWithData:(NSData *)imageData];
  ```
- Or you can even create one by drawing with CGContext functions
  ```
  UIGraphicsBeginImageContext(CGSize);
  // draw with CGContext functions
  UIImage *myImage = UIGraphicsGetImageFromCurrentContext();
  UIGraphicsEndImageContext();
  ```

Drawing Images

- Now blast the UIImage's bits into the current graphics context

```
UIImage *image = …;
[image drawAtPoint:(CGPoint)p];                 // p is upper left corner of the image
[image drawInRect:(CGRect)r];                   // scales the image to fit in r
[image drawAsPatternInRect:(CGRect)patRect];    // tiles the image into patRect
```

- Aside: You can get a PNG or JPG data representation of UIImage

```
NSData *jpgData = UIImageJPEGRepresentation((UIImage *)myImage, (CGFloat)quality);
NSData *pngData = UIImagePNGRepresentation((UIImage *)myImage);
```

Redraw on bounds change?

- By default, when your UIView's bounds change, there is no redraw
  Instead, the "bits" of your view will be stretched or squished or moved.
- Often this is not what you want …
  Luckily, there is a UIView @property to control this! It can be set in Xcode.
  `@property (nonatomic) UIViewContentMode contentMode;`

  These content modes move the bits of your drawing to that location …
  `UIViewContentMode{Left,Right,Top,Right,BottomLeft,BottomRight,TopLeft,TopRight}`
  These content modes stretch the bits of your drawing …
  `UIViewContentModeScale{ToFill,AspectFill,AspectFit}` // bit stretching/shrinking
  This content mode calls drawRect: to redraw everything when the bounds changes …
  `UIViewContentModeRedraw` // it is quite often that this is what you want

  Default is `UIViewContentModeScaleToFill` (stretch the bits to fill the bounds)

UIGestureRecognizer

- We've seen how to draw in our UIView, how do we get touches?
  We can get notified of the raw touch events (touch down, moved, up).
  Or we can react to certain, predefined "gestures." This latter is the way to go.
- Gestures are recognized by the class UIGestureRecognizer
  This class is "abstract." We only actually use "concrete subclasses" of it.
- There are two sides to using a gesture recognizer
  1. Adding a gesture recognizer to a UIView to ask it to recognize that gesture.
  2. Providing the implementation of a method to "handle" that gesture when it happens.
- Usually #1 is done by a Controller
  Though occasionally a UIView will do it to itself if it just doesn't make sense without that gesture.
- Usually #2 is provided by the UIView itself
  But it would not be unreasonable for the Controller to do it.
  Or for the Controller to decide it wants to handle a gesture differently than the view does.

UIGestureRecognizer

- Adding a gesture recognizer to a UIView from a Controller

```
- (void)setPannableView:(UIView *)pannableView // maybe this is a setter in a Controller
{
    _pannableView = pannableView;
    UIPanGestureRecognizer *pangr =
        [[UIPanGestureRecognizer alloc] initWithTarget:pannableView action:@selector(pan:)];
    [pannableView addGestureRecognizer:pangr];
}
```

This is a concrete subclass of UIGestureRecognizer that recognizes "panning" (moving something around with your finger).

There are, of course, other concrete subclasses (for swipe, pinch, tap, etc.).

UIGestureRecognizer

- Adding a gesture recognizer to a UIView from a Controller

```
- (void)setPannableView:(UIView *)pannableView // maybe this is a setter in a Controller
{
    _pannableView = pannableView;
    UIPanGestureRecognizer *pangr =
        [[UIPanGestureRecognizer alloc] initWithTarget:pannableView action:@selector(pan:)];
    [pannableView addGestureRecognizer:pangr];
}
```

Note that we are specifying the view itself as the target to handle a pan gesture when it is recognized.
Thus the view will be both the recognizer and the handler of the gesture.

The UIView does not have to handle the gesture. It could be, for example, the Controller that handles it.
The View would generally handle gestures to modify how the View is drawn.
The Controller would have to handle gestures that modified the Model.

UIGestureRecognizer

- Adding a gesture recognizer to a UIView from a Controller

```
- (void)setPannableView:(UIView *)pannableView // maybe this is a setter in a Controller
{
    _pannableView = pannableView;
    UIPanGestureRecognizer *pangr =
        [[UIPanGestureRecognizer alloc] initWithTarget:pannableView action:@selector(pan:)];
    [pannableView addGestureRecognizer:pangr];
}
```

This is the action method that will be sent to the target
(the pannableView) during the handling of the recognition of this gesture.

This version of the action message takes one argument
(which is the UIGestureRecognizer that sends the action),
but there is another version that takes no arguments if you'd prefer.

We'll look at the implementation of this method in a moment.

UIGestureRecognizer

- Adding a gesture recognizer to a UIView from a Controller

```
- (void)setPannableView:(UIView *)pannableView // maybe this is a setter in a Controller
{
    _pannableView = pannableView;
    UIPanGestureRecognizer *pangr =
        [[UIPanGestureRecognizer alloc] initWithTarget:pannableView action:@selector(pan:)];
    [pannableView addGestureRecognizer:pangr];
}
```

If we don't do this, then even though the pannableView implements pan:, it would never get called because we would have never added this gesture recognizer to the view's list of gestures that it recognizes.

Think of this as "turning the handling of this gesture on."

UIGestureRecognizer

- Adding a gesture recognizer to a UIView from a Controller

```
- (void)setPannableView:(UIView *)pannableView // maybe this is a setter in a Controller
{
    _pannableView = pannableView;
    UIPanGestureRecognizer *pangr =
        [[UIPanGestureRecognizer alloc] initWithTarget:pannableView action:@selector(pan:)];
    [pannableView addGestureRecognizer:pangr];
}
```

Only UIView instances can recognize a gesture (because UIViews handle all touch input).
But any object can tell a UIView to recognize a gesture (by adding a recognizer to the UIView).
And any object can handle the recognition of a gesture (by being the target of the gesture's action).

UIGestureRecognizer

- How do we implement the target of a gesture recognizer?
  Each concrete class provides some methods to help you do that.
- For example, UIPanGestureRecognizer provides 3 methods:
  ```
  - (CGPoint)translationInView:(UIView *)aView;
  - (CGPoint)velocityInView:(UIView *)aView;
  - (void)setTranslation:(CGPoint)translation inView:(UIView *)aView;
  ```
- Also, the base class, UIGestureRecognizer provides this @property:
  ```
  @property (readonly) UIGestureRecognizerState state;
  ```
  Gesture Recognizers sit around in the state Possible until they start to be recognized
  Then they either go to Recognized (for discrete gestures like a tap)
  Or they go to Began (for continuous gestures like a pan)
  At any time, the state can change to Failed (so watch out for that)
  If the gesture is continuous, it'll move on to the Changed and eventually the Ended state
  Continuous can also go to Cancelled state (if the recognizer realizes it's not this gesture after all)

UIGestureRecognizer

- So, given these methods, what would pan: look like?

```
- (void)pan:(UIPanGestureRecognizer *)recognizer
{
    if ((recognizer.state == UIGestureRecognizerStateChanged) ||
        (recognizer.state == UIGestureRecognizerStateEnded)) {



    }
}
```

We're going to update our view every time the touch moves (and when the touch ends). This is "smooth panning."

UIGestureRecognizer

- So, given these methods, what would pan: look like?

```
- (void)pan:(UIPanGestureRecognizer *)recognizer
{
    if ((recognizer.state == UIGestureRecognizerStateChanged) ||
        (recognizer.state == UIGestureRecognizerStateEnded)) {
        CGPoint translation = [recognizer translationInView:self];


    }
}
```

This is the cumulative distance this gesture has moved.

UIGestureRecognizer

- So, given these methods, what would pan: look like?

```
- (void)pan:(UIPanGestureRecognizer *)recognizer
{
    if ((recognizer.state == UIGestureRecognizerStateChanged) ||
        (recognizer.state == UIGestureRecognizerStateEnded)) {
        CGPoint translation = [recognizer translationInView:self];
        // move something in myself (I'm a UIView) by translation.x and translation.y
        // for example, if I were a graph and my origin was set by an @property called origin
        self.origin = CGPointMake(self.origin.x+translation.x, self.origin.y+translation.y);

    }
}
```

UIGestureRecognizer

- So, given these methods, what would pan: look like?

```
- (void)pan:(UIPanGestureRecognizer *)recognizer
{
    if ((recognizer.state == UIGestureRecognizerStateChanged) ||
        (recognizer.state == UIGestureRecognizerStateEnded)) {
        CGPoint translation = [recognizer translationInView:self];
        // move something in myself (I'm a UIView) by translation.x and translation.y
        // for example, if I were a graph and my origin was set by an @property called origin
        self.origin = CGPointMake(self.origin.x+translation.x, self.origin.y+translation.y);
        [recognizer setTranslation:CGPointZero inView:self];
    }
}
```

Here we are resetting the cumulative distance to zero.

Now each time this is called, we'll get the "incremental" movement of the gesture (which is what we want). If we wanted the "cumulative" movement of the gesture, we would not include this line of code.

UIGestureRecognizer

- So, given these methods, what would pan: look like?

```
- (void)pan:(UIPanGestureRecognizer *)recognizer
{
    if ((recognizer.state == UIGestureRecognizerStateChanged) ||
        (recognizer.state == UIGestureRecognizerStateEnded)) {
        CGPoint translation = [recognizer translationInView:self];
        // move something in myself (I'm a UIView) by translation.x and translation.y
        // for example, if I were a graph and my origin was set by an @property called origin
        self.origin = CGPointMake(self.origin.x+translation.x, self.origin.y+translation.y);
        [recognizer setTranslation:CGPointZero inView:self];
    }
}
```

Other Concrete Gestures

- UIPinchGestureRecognizer

  ```
  @property CGFloat scale;  // note that this is not readonly (can reset each movement)
  @property (readonly) CGFloat velocity; // note that this is readonly; scale factor per second
  ```

- UIRotationGestureRecognizer

  ```
  @property CGFloat rotation;  // not readonly; in radians
  @property (readonly) CGFloat velocity; // readonly; radians per second
  ```

- UISwipeGestureRecognizer

  This one you "set up" (w/the following) to find certain swipe types, then look for `Recognized` state

  ```
  @property UISwipeGestureRecognizerDirection direction;  // what direction swipes you want
  @property NSUInteger numberOfTouchesRequired; // two finger swipes? or just one finger? more?
  ```

- UITapGestureRecognizer

  Set up (w/the following) then look for `Recognized` state

  ```
  @property NSUInteger numberOfTapsRequired;  // single tap or double tap or triple tap, etc.
  @property NSUInteger numberOfTouchesRequired;  // e.g., require two finger tap?
  ```

Demo

- SuperCard
  Let's make a lot better-looking playing card!
- What to watch for ...
  Custom `UIView` with its own `drawRect:`
  `setNeedsDisplay`
  `UIBezierPath`
  Clipping
  Pushing and popping graphics context
  Drawing text with `NSAttributedString` and images with `UIImage`
  Document Outline and Size Inspector in Xcode
  Gestures recognizers hooked up in Xcode vs. programmatically
  Controller vs. View as gesture handler

Coming Up

- Friday

  Running on your device with the University Developer Program (Stanford Only).

- Homework

  Due a week from Monday.

  Get started on first few Required Tasks immediately (i.e. replacing button with custom `UIView`).

- Next Week

  Animation

  Autolayout

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

- Protocols
  How to make id a little bit safer.
- Blocks
  Passing a block of code as an argument to a method.
- Animation
  Dynamic Animator
  View property animation
- Demo
  Dropit!

Protocols

- The problem with `id` …
  Obviously it's hard to communicate your intent with `id`.
  What do you want callers of this method to pass (or what are you returning) exactly?
- Introspection
  Helps occasionally, but not a "primary programming methodology."
  And it doesn't help with communicating your intent at all (it's more of a runtime thing).
- Protocols
  A syntactical modification of `id`, for example, `id <MyProtocol> obj`.
  `MyProtocol` would then be defined to be a list of methods (including `@property`s).
  The variable `obj` now can point to an object of any class, but that it implements known methods.
  Not all the methods in a protocol have to be required, but still, you'll know what's expected.
  Let's look at the syntax …

Protocols

- Declaring a @protocol

  Looks a lot like @interface (but there's no corresponding @implementation)

```
@protocol Foo
- (void)someMethod;
- (void)methodWithArgument:(BOOL)argument;
@property (readonly) int readonlyProperty; // getter (only) is part of this protocol
@property NSString *readwriteProperty;     // getter and setter are both in the protocol
- (int)methodThatReturnsSomething;
@end
```

  All of these methods are required. Anyone implementing this protocol must implement them all.

Protocols

- Declaring a @protocol

  Looks a lot like @interface (but there's no corresponding @implementation)

```
@protocol Foo
- (void)someMethod;
@optional
- (void)methodWithArgument:(BOOL)argument;
@property (readonly) int readonlyProperty; // getter (only) is part of this protocol
@property NSString *readwriteProperty;     // getter and setter are both in the protocol
- (int)methodThatReturnsSomething;
@end
```

  Now only the first one is required.

  You can still say you implement Foo even if you only implement someMethod.

Protocols

- Declaring a @protocol

  Looks a lot like @interface (but there's no corresponding @implementation)

```
@protocol Foo
- (void)someMethod;
@optional
- (void)methodWithArgument:(BOOL)argument;
@required
@property (readonly) int readonlyProperty; // getter (only) is part of this protocol
@property NSString *readwriteProperty;     // getter and setter are both in the protocol
- (int)methodThatReturnsSomething;
@end
```

  Now all of them except methodWithArgument: are required.

Protocols

- Declaring a @protocol

  Looks a lot like @interface (but there's no corresponding @implementation)

```
@protocol Foo <Xyzzy>
- (void)someMethod;
@optional
- (void)methodWithArgument:(BOOL)argument;
@required
@property (readonly) int readonlyProperty; // getter (only) is part of this protocol
@property NSString *readwriteProperty;     // getter and setter are both in the protocol
- (int)methodThatReturnsSomething;
@end
```

  Now all of them except methodWithArgument: are required.

  Now you can only say you implement Foo if you also implement the methods in Xyzzy protocol.

Protocols

- Declaring a `@protocol`

  Looks a lot like `@interface` (but there's no corresponding `@implementation`)

```
@protocol Foo <Xyzzy, NSObject>
- (void)someMethod;
@optional
- (void)methodWithArgument:(BOOL)argument;
@required
@property (readonly) int readonlyProperty; // getter (only) is part of this protocol
@property NSString *readwriteProperty;     // getter and setter are both in the protocol
- (int)methodThatReturnsSomething;
@end
```

  Now all of them except `methodWithArgument:` are required.

  Now you can only say you implement `Foo` if you also implement the methods in `Xyzzy` protocol.

  Now you would have to implement both the `Xyzzy` protocol and the `NSObject` protocol (what's that!?).

Protocols

- `@protocol NSObject`

  Has things like `class`, `isEqual:`, `isKindOfClass:`, `description`, `performSelector:`, etc.

  Not uncommon to add this requirement when declaring a protocol.

  Then you can rely on using introspection and such on the object obeying the protocol.

  Of course, the class `NSObject` implements the protocol `NSObject` (so it comes for free!).

Protocols

- Where do `@protocol` declarations go?

  In header files.

  It can go in its own, dedicated header file.

  Or it can go in the header file of the class that is going to require it's use.

  Which to do?

  If the `@protocol` is only required by a particular class's API, then put it there, else put it in its own header file.

  Example: The `UIScrollViewDelegate` protocol is defined in `UIScrollView.h`.

Protocols

- Okay, I have a `@protocol` declared, now what?

  Now classes can promise to implement the protocol in their `@interface` declarations.

  Okay to put in private `@interface` if they don't want others to know they implement it.

- Example:

```
#import "Foo.h"                    // importing the header file that declares the Foo @protocol
@interface MyClass : NSObject <Foo> // MyClass is saying it implements the Foo @protocol
   (do not have to declare Foo's methods again here, it's implicit that you implement it)
@end
... or ("or" not "and"... it's one or the other, private or public, not both) ...
@interface MyClass() <Foo>
@end

@implementation MyClass
// in either case, you had better implement Foo's @required methods here!
@end
```

Protocols

- The class must now implement all non-`@optional` methods

  Or face the wrath of the compiler if you do not (but that's the only wrath you'll face).

  No warning if you don't implement `@optional` methods.

  `@optional` is more a mechanism to say: "hey, if you implement this, I'll use it."

  (i.e. caller will likely use introspection to be sure you actually implement an `@optional` method)

  `@required` is much stronger: "if you want this to work, you must implement this."

  (very unlikely that the caller would use introspection before invoking `@required` methods)

Protocols

- Okay, so now what?

  We have protocols.

  We have classes that promise to implement them.

  Now we need variables that hold pointers to objects that make that promise.

- Examples …

```
id <Foo> obj = [[MyClass alloc] init];  // compiler will love this (due to previous slides)
id <Foo> obj = [NSArray array];         // compiler will not like this one bit!
```

- Can also declare arguments to methods to require a protocol

```
- (void)giveMeFooObject:(id <Foo>)anObjectImplementingFoo;
@property (nonatomic, weak) id <Foo> myFooProperty; // properties too!
```

  If you call these and pass an object which does not implement `Foo` … compiler warning!

Protocols

- Just like static typing, this is all just compiler-helping-you stuff
  It makes no difference at runtime.
- Think of it as documentation for your method interfaces
  It's a powerful way to leverage the id type.

Protocols

- #1 use of protocols in iOS: delegates and dataSources

  Often when an object in iOS wants something important and non-generic done, it may delegate it.
  It does this through a property on that iOS object that is specified with a certain protocol.

```
@property (nonatomic, weak) id <UISomeObjectDelegate> delegate;
@property (nonatomic, weak) id <UISomeObjectDataSource> dataSource;
```

  Note that it is a weak (or worse) @property, by the way (more on that soon).
  You may implement your own delegates too (we'll see that later in the course).
  This is an alternative to subclassing to provide non-generic behavior.
  You use delegation when you want to be "blind" to the class of the implementing object (MVC).

- dataSource and Views

  Complex UIView classes commonly have a dataSource because Views cannot own their data!

- Other uses of protocols

  Declaring what sorts of things are "animatable" (mostly UIView, but other things too).
  We'll see other uses as the quarter progresses.

Blocks

- What is a block?

  A block of code (i.e. a sequence of statements inside {}).
  Usually included "in-line" with the calling of method that is going to use the block of code.
  Very smart about local variables, referenced objects, etc.

- What does it look like?

  Here's an example of calling a method that takes a block as an argument.

```
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@"value for key %@ is %@", key, value);
    if ([@"ENOUGH" isEqualToString:key]) {
        *stop = YES;
    }
}];
```

  This NSLog()s every key and value in aDictionary (but stops if the key is "ENOUGH").

- Blocks start with the magical character caret ^

  Then (optional) return type, then (optional) arguments in parentheses, then {, then code, then }.

Blocks

- Can use local variables declared before the block inside the block

```
double stopValue = 53.5;
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@"value for key %@ is %@", key, value);
    if ([@"ENOUGH" isEqualToString:key] || ([value doubleValue] == stopValue)) {
        *stop = YES;
    }
}];
```

- But they are read only!

```
BOOL stoppedEarly = NO;
double stopValue = 53.5;
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@"value for key %@ is %@", key, value);
    if ([@"ENOUGH" isEqualToString:key] || ([value doubleValue] == stopValue)) {
        *stop = YES;
        stoppedEarly = YES;  // ILLEGAL
    }
}];
```

Blocks

- Unless you mark the local variable as `__block`

```
__block BOOL stoppedEarly = NO;
double stopValue = 53.5;
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@“value for key %@ is %@”, key, value);
    if ([@“ENOUGH” isEqualToString:key] || ([value doubleValue] == stopValue)) {
        *stop = YES;
        stoppedEarly = YES;  // this is legal now
    }
}];
if (stoppedEarly) NSLog(@“I stopped logging dictionary values early!”);
```

- Or if the “variable” is an instance variable

  But we only access instance variables (e.g. `_display`) in setters and getters.
  So this is of minimal value to us.

Blocks

- So what about objects which are messaged inside the block?

```
NSString *stopKey = [@“Enough” uppercaseString];
__block BOOL stoppedEarly = NO;
double stopValue = 53.5;
[aDictionary enumerateKeysAndObjectsUsingBlock:^(id key, id value, BOOL *stop) {
    NSLog(@“value for key %@ is %@”, key, value);
    if ([stopKey isEqualToString:key] || ([value doubleValue] == stopValue)) {
        *stop = YES;
        stoppedEarly = YES;  // this is legal now
    }
}];
if (stoppedEarly) NSLog(@“I stopped logging dictionary values early!”);
```

stopKey will automatically have a strong pointer to it until the block goes out of scope
This is obviously necessary for the block to function properly.

Blocks

- Creating a "type" for a variable that can hold a block

  Blocks are kind of like "objects" with an unusual syntax for declaring variables that hold them.

  Usually if we are going to store a block in a variable, we typedef a type for that variable, e.g.,

```
typedef double (^unary_operation_t)(double op);
```

  This declares a type called "unary_operation_t" for variables which can store a block.

  (specifically, a block which takes a double as its only argument and returns a double)

  Then we could declare a variable, square, of this type and give it a value ...

```
unary_operation_t square;
square = ^(double operand) { // the value of the square variable is a block
    return operand * operand;
}
```

  And then use the variable square like this ...

```
double squareOfFive = square(5.0);  // squareOfFive would have the value 25.0 after this
```

  (It is not mandatory to typedef, for example, the following is also a legal way to create square ...)

```
double (^square)(double op) = ^(double op) { return op * op; };
```

Blocks

- We could then use the unary_operation_t as follows …

  For example, you could have a property which is an array of blocks …

```
@property (nonatomic, strong) NSMutableDictionary *unaryOperations;
```

  Then implement a method like this …

```
typedef double (^unary_operation_t)(double op);
- (void)addUnaryOperation:(NSString *)op whichExecutesBlock:(unary_operation_t)opBlock {
    self.unaryOperations[op] = opBlock;
}
```

  Note that the block can be treated somewhat like an object (e.g., adding it to a dictionary).

  Later, we could use an operation added with the method above like this …

```
- (double)performOperation:(NSString *)operation onOperand:(double)operand
{
    unary_operation_t unaryOp = self.unaryOperations[operation];
    if (unaryOp) {
        double result = unaryOp(operand);
    }
    ...
}
```

Blocks

- We don't always `typedef`

  When a block is an argument to a method and is used immediately, often there is no `typedef`.

  Here is the declaration of the dictionary enumerating method we showed earlier …

```
- (void)enumerateKeysAndObjectsUsingBlock:(void (^)(id key, id obj, BOOL *stop))block;
```

  No "name" for the type appears here.

  The syntax is exactly the same as the `typedef` except that the name of the `typedef` is not there.

  For reference, here's what a `typedef` for this argument would look like this …

```
typedef void (^enumeratingBlock)(id key, id obj, BOOL *stop);
```

  (i.e. the underlined part is not used in the method argument)

  This ("`block`") is the keyword for the argument (e.g. the local variable name for the argument inside the method implementation).

Blocks

- Some shorthand allowed when defining a block

  If there are no arguments to the block, you do not need to have any parentheses.

  Consider this code …

```
[UIView animateWithDuration:5.0 animations:^() {
    view.opacity = 0.5;
}];
```

Blocks

- Some shorthand allowed when defining a block

  If there are no arguments to the block, you do not need to have any parentheses.

  Consider this code …

```
[UIView animateWithDuration:5.0 animations:^{
    view.opacity = 0.5;
}];
```

No arguments to this block.

No need for the () then.

Blocks

- Some shorthand allowed when defining a block

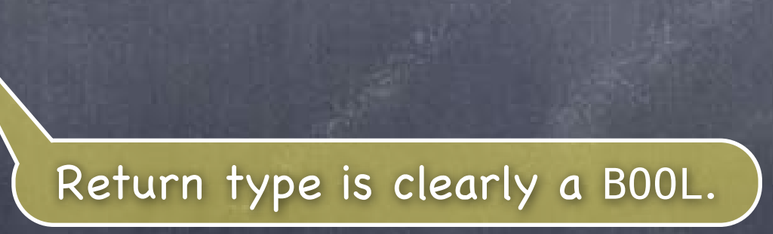
  If there are no arguments to the block, you do not need to have any parentheses.

  Consider this code …

```
[UIView animateWithDuration:5.0 animations:^{
    view.opacity = 0.5;
}];
```

  Also, return type can usually be inferred from the block, in which case it is optional.

```
NSSet *mySet = …;
NSSet *matches = [mySet objectsPassingTest:^BOOL(id obj, …) {
    return [obj isKindOfClass:[UIView class]];
}];
```



Blocks

- Some shorthand allowed when defining a block

  If there are no arguments to the block, you do not need to have any parentheses.

  Consider this code …

```
[UIView animateWithDuration:5.0 animations:^{
    view.opacity = 0.5;
}];
```

  Also, return type can usually be inferred from the block, in which case it is optional.

```
NSSet *mySet = …;
NSSet *matches = [mySet objectsPassingTest:^(id obj, …) {
    return [obj isKindOfClass:[UIView class]];
}];
```

Return type is clearly a BOOL.

So no need for the BOOL declaration here.

Blocks

- How blocks sort of act like objects

  It turns out blocks can be stored inside other objects (in properties, arrays, dictionaries, etc.).
  But they act like objects only for the purposes of storing them (their only "method" is copy).

  For example, if you had a class with the following property …

```
@property (nonatomic, strong) NSMutableArray *myBlocks; // array of blocks
```

  … you could do the following …

```
[self.myBlocks addObject:^{
    [self doSomething];
}];
```

  … neat!

  By the way, you invoke a block that is in the array like this …

```
void (^doit)(void) = self.myBlocks[0];
doit();
```

  But there is danger lurking here …

Blocks

- Memory Cycles (a bad thing)

  We said that all objects referenced inside a block will stay in the heap as long as the block does (in other words, blocks keep a strong pointer to all objects referenced inside of them).

  In the example above, self is an object reference in this block …

```
[self.myBlocks addObject:^ {
    [self doSomething];
}];
```

  Thus the block will have a strong pointer to self.

  But notice that self also has a strong pointer to the block (it's in its myBlocks array)!

  This is a serious problem.

  Neither self nor the block can ever escape the heap now.

  That's because there will always be a strong pointer to both of them (each other's pointer).

  This is called a memory "cycle."

Blocks

- ## Memory Cycles Solution

  You'll recall that local variables are always strong.
  That's fine because when they go out of scope, they disappear, so the strong pointer goes away.

  It turns out there's a way to declare that a local variable is weak. Here's how …

```
__weak MyClass *weakSelf = self; // even though self is strong, weakSelf is weak
```

  Now if we reference weakSelf inside the block, then the block will not strongly point to self …

```
[self.myBlocks addObject:^ {
    [weakSelf doSomething];
}];
```

  Now we no longer have a cycle (self still has a strong pointer to the block, but that's okay).
  As long as someone in the universe has a strong pointer to this self, the block's pointer is good.
  And since the block will not exist if self does not exist (since myBlocks won't exist), all is well!

Blocks

- When do we use blocks in iOS?
  - Enumeration (like we saw above with `NSDictionary`)
  - View Animations (we'll talk about that next)
  - Sorting (sort this thing using a block as the comparison method)
  - Notification (when something happens, execute this block)
  - Error handlers (if an error happens while doing this, execute this block)
  - Completion handlers (when you are done doing this, execute this block)
- And a super-important use: Multithreading
  - With Grand Central Dispatch (GCD) API
  - We'll talk about that later in the course
- More about blocks
  - Search "blocks" in Xcode documentation.

Animation

- Animating views

  Animating specific properties.

  Animating a group of changes to a view "all at once."

  Physics-based animation.

- Animation of View Controller transitions

  Beyond the scope of this course, but fundamental principles are the same.

- Core Animation

  Underlying powerful animation framework (also beyond the scope of this course).

Animation

- Animation of important `UIView` properties
  The changes are made immediately, but appear on-screen over time (i.e. not instantly).
  `UIView`'s class method(s) `animationWithDuration:...`
- Animation of the appearance of arbitrary changes to a `UIView`
  By flipping or dissolving or curling the entire view.
  `UIView`'s class method `transitionWithView:...`
- Dynamic Animator
  Specify the "physics" of animatable objects (usually `UIViews`).
  Gravity, pushing forces, attachments between objects, collision boundaries, etc.
  Let the physics happen!

UIView Animation

- Changes to certain UIView properties can be animated over time
  - frame
  - transform (translation, rotation and scale)
  - alpha (opacity)
- Done with UIView class method and blocks
  - The class method takes animation parameters and an animation block as arguments.
  - The animation block contains the code that makes the changes to the UIView(s).
  - Most also have a "completion block" to be executed when the animation is done.
  - The changes inside the block are made immediately (even though they will appear "over time").

UIView Animation

- Animation class method in UIView

```
+ (void)animateWithDuration:(NSTimeInterval)duration
                      delay:(NSTimeInterval)delay
                    options:(UIViewAnimationOptions)options
                 animations:(void (^)(void))animations
                 completion:(void (^)(BOOL finished))completion;
```

- Example

```
[UIView animateWithDuration:3.0
                      delay:0.0
                    options:UIViewAnimationOptionBeginFromCurrentState
                 animations:^{ myView.alpha = 0.0; }
                 completion:^(BOOL fin) { if (fin) [myView removeFromSuperview]; }];
```

This would cause myView to "fade" out over 3 seconds (starting immediately).

Then it would remove myView from the view hierarchy (but only if the fade completed).

If, within the 3 seconds, someone animated the alpha to non-zero, the removal would not happen.

UIView Animation

- Another example

```
if (myView.alpha == 1.0) {
    [UIView animateWithDuration:3.0
                          delay:2.0
                        options:UIViewAnimationOptionBeginFromCurrentState
                     animations:^{ myView.alpha = 0.0; }
                     completion:nil];
    NSLog(@"alpha is %f.", myView.alpha);
}
```

This would also cause `myView` to "fade" out over 3 seconds (starting in 2 seconds in this case).
The `NSLog()` would happen immediately (i.e. not after 3 or 5 seconds) and would print "alpha is 0."
In other words, the animation block's changes are executed immediately, but the animation itself (i.e. the visual appearance of the change to `alpha`) starts in 2 seconds and takes 3 seconds.

UIView Animation

- UIViewAnimationOptions

```
BeginFromCurrentState       // interrupt other, in-progress animations of these properties
AllowUserInteraction        // allow gestures to get processed while animation is in progress
LayoutSubviews              // animate the relayout of subviews along with a parent's animation
Repeat                      // repeat indefinitely
Autoreverse                 // play animation forwards, then backwards
OverrideInheritedDuration   // if not set, use duration of any in-progress animation
OverrideInheritedCurve      // if not set, use curve (e.g. ease-in/out) of in-progress animation
AllowAnimatedContent        // if not set, just interpolate between current and end state image
CurveEaseInEaseOut          // slower at the beginning, normal throughout, then slow at end
CurveEaseIn                 // slower at the beginning, but then constant through the rest
CurveLinear                 // same speed throughout
```

UIView Animation

- Sometimes you want to make an entire view modification at once
  By flipping view over `UIViewAnimationOptionsTransitionFlipFrom{Left,Right,Top,Bottom}`
  Dissolving from old to new state `UIViewAnimationOptionsTransitionCrossDissolve`
  Curling up or down `UIViewAnimationOptionsTransitionCurl{Up,Down}`

  Just put the changes inside the `animations` block of this `UIView` class method …

```
+ (void)transitionWithView:(UIView *)view
                  duration:(NSTimeInterval)duration
                   options:(UIViewAnimationOptions)options
                animations:(void (^)(void))animations
                completion:(void (^)(BOOL finished))completion;
```

UIView Animation

- Animating changes to the view hierarchy is slightly different

  Animate swapping the replacement of one view with another in the view hierarchy.

```
+ (void)transitionFromView:(UIView *)fromView
                    toView:(UIView *)toView
                  duration:(NSTimeInterval)duration
                   options:(UIViewAnimationOptions)options
                completion:(void (^)(BOOL finished))completion;
```

  Include `UIViewAnimationOptionShowHideTransitionViews` if you want to use the `hidden` property. Otherwise it will actually remove `fromView` from the view hierarchy and add `toView`.

Dynamic Animation

- A little different approach to animation than above
  Set up physics relating animatable objects and let them run until they resolve to stasis
  Easily possible to set it up so that stasis never occurs, but that could be performance problem
- Steps
  Create a `UIDynamicAnimator`
  Add `UIDynamicBehaviors` to it (gravity, collisions, etc.)
  Add `UIDynamicItems` (usually `UIViews`) to the `UIDynamicBehaviors`
  That's it! Things will instantly start happening.

Dynamic Animator

- Create a UIDynamicAnimator

```
UIDynamicAnimator *animator = [[UIDynamicAnimator alloc] initWithReferenceView:aView];
```

If animating views, all views must be in a view hierarchy with reference view at the top.

- Create and add UIDynamicBehaviors

e.g., `UIGravityBehavior *gravity = [[UIGravityBehavior alloc] init];`

`[animator addBehavior:gravity];`

e.g., `UICollisionBehavior *collider = [[UICollisionBehavior alloc] init];`

`[animator addBehavior:collider];`

Dynamic Animator

- Add UIDynamicItems to a UIDynamicBehavior

```
id <UIDynamicItem> item1 = …;
id <UIDynamicItem> item2 = …;
[gravity addItem:item1];
[collider addItem:item1];
[gravity addItem:item2];
```

The items have to implement the UIDynamicItem protocol …

```
@protocol UIDynamicItem
@property (readonly) CGRect bounds;
@property (readwrite) CGPoint center;
@property (readwrite) CGAffineTransform transform;
@end
```

UIView implements this @protocol.

If you change center or transform while animator is running, you must call UIDynamicAnimator's

```
- (void)updateItemUsingCurrentState:(id <UIDynamicItem>)item;
```

Behaviors

- UIGravityBehavior

```
@property CGFloat angle;
@property CGFloat magnitude; // 1.0 is 1000 points/s/s
```

- UICollisionBehavior

```
@property UICollisionBehaviorMode collisionMode; // Items, Boundaries, Everything (default)
- (void)addBoundaryWithIdentifier:(NSString *)identifier forPath:(UIBezierPath *)path;
@property BOOL translatesReferenceBoundsIntoBoundary;
```

- UIAttachmentBehavior

```
- (instancetype)initWithItem:(id <UIDynamicItem>)item attachedToAnchor:(CGPoint)anchor;
- (instancetype)initWithItem:(id <UIDynamicItem>)i1 attachedToItem:(id <UIDynamicItem>)i2;
- (instancetype)initWithItem:(id <UIDynamicItem>)item offsetFromCenter:(CGPoint)offset …
@property (readwrite) CGFloat length; // distance between attached things (settable!)
```

Can also control `damping` and `frequency` of oscillations.

```
@property (readwrite) CGPoint anchorPoint; // can be reset at any time
```

Behaviors

- UISnapBehavior

  `- (instancetype)initWithItem:(id <UIDynamicItem>)item snapToPoint:(CGPoint)point;`

  Imagine four springs at four corners around the item in the new spot.

  You can control the damping of these "four springs" with `@property CGFloat damping;`.

- UIPushBehavior

  ```
  @property UIPushBehaviorMode mode; // Continuous or Instantaneous
  @property CGVector pushDirection;
  @property CGFloat magnitude/angle; // magnitude 1.0 moves a 100x100 view at 100 pts/s/s
  ```

Behaviors

- UIDynamicItemBehavior

  Controls the behavior of items as they are affected by other behaviors.
  Any item added to this behavior (with addItem:) will be affected.

```
@property BOOL allowsRotation;
@property BOOL friction;
@property BOOL elasticity;
@property CGFloat density;
```

  Can also get information about items …

```
- (CGPoint)linearVelocityForItem:(id <UIDynamicItem>)item;
- (CGFloat)angularVelocityForItem:(id <UIDynamicItem>)item;
```

  If you have multiple UIDynamicItemBehaviors, you will have to know what you are doing.

Behaviors

- UIDynamicBehavior

  Superclass of behaviors.

  You can create your own subclass which is a combination of other behaviors.

  Usually you override `init` method(s) and `addItem(s):` and `removeItem(s):` to do …

```
- (void)addChildBehavior:(UIDynamicBehavior *)behavior;
```

  This is a good way to encapsulate a physics behavior that is a composite of other behaviors.

  You might also have some API which helps your subclass configure its children.

- All behaviors know the `UIDynamicAnimator` they are part of

  They can only be part of one at a time.

```
@property UIDynamicAnimator *dynamicAnimator;
```

  And the behavior will be sent this message when its animator changes …

```
- (void)willMoveToAnimator:(UIDynamicAnimator *)animator;
```

Behaviors

- UIDynamicBehavior's action property

  Every time the behavior is applied, the block set with this UIDynamicBehavior property is called …

  ```
  @property (copy) void (^action)(void);
  ```

  (i.e. it's called action, it takes no arguments and returns nothing)

  You can set this to do anything you want.
  But it will be called a lot, so make it very efficient.

  If the action refers to properties in the behavior itself, watch out for memory cycles.

Demo

- Dropit

  Drop squares, collect them at the bottom of the screen, then blow them up!

- What to look for ...

  `UIDynamicAnimator` and `UIDynamicItem` `@protocol`
  `UIGravityBehavior`
  `UICollisionBehavior`
  `UIDynamicItemBehavior` (basically physics configuration)
  Composite Behaviors (`UIDynamicBehavior` subclass)
  Flying `UIViews` using `animateWithDuration:...`
  Animation `completion` blocks
  `UIDynamicAnimator`'s `delegate` (reacting to pauses in dynamic animation)
  `UIAttachmentBehavior`
  Adding an `action` block to a behavior
  Observing the behavior of items (elapsed animation time, linear velocity, etc.)
  `UICollisionBehavior`'s `collisionDelegate`

Coming Up

- Wednesday
  Continuation of demo.
  Autolayout
- Friday
  Still hoping to get University Developer Program up and running.
- Homework
  Due a week from today.
- Next Week
  Scroll View
  Table View
  Collection View

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

- Finish Animation Demo
  Less tippy, guided drops.
- Autolayout
  How to make device autorotation easy(er).
  And make your View Controller work in different environments (i.e. with different bounds).
- Autolayout Demo
  Making Attributor autorotate properly.

Demo

- More Dropit
  Less tippy!
  Guiding the fall of drops.
  If time permits, gridding using collision delegate (if not, will post code).
- What to look for today ...
  `UIDynamicItemBehavior` (basically physics configuration)
  `UIAttachmentBehavior`
  Adding an `action` block to a behavior
  Observing the behavior of items (elapsed animation time, linear velocity, etc.)
  `UICollisionBehavior`'s `collisionDelegate`

Autolayout

- Setting `UIView` `frames` using rules rather than numbers

  Why? Because many things affect the size of the area available to put views …

  Rotation

  4 inch versus 3.5 inch iPhone

  Embedding Controller's Views inside other Controllers (tab bars, navigation controllers, etc.)

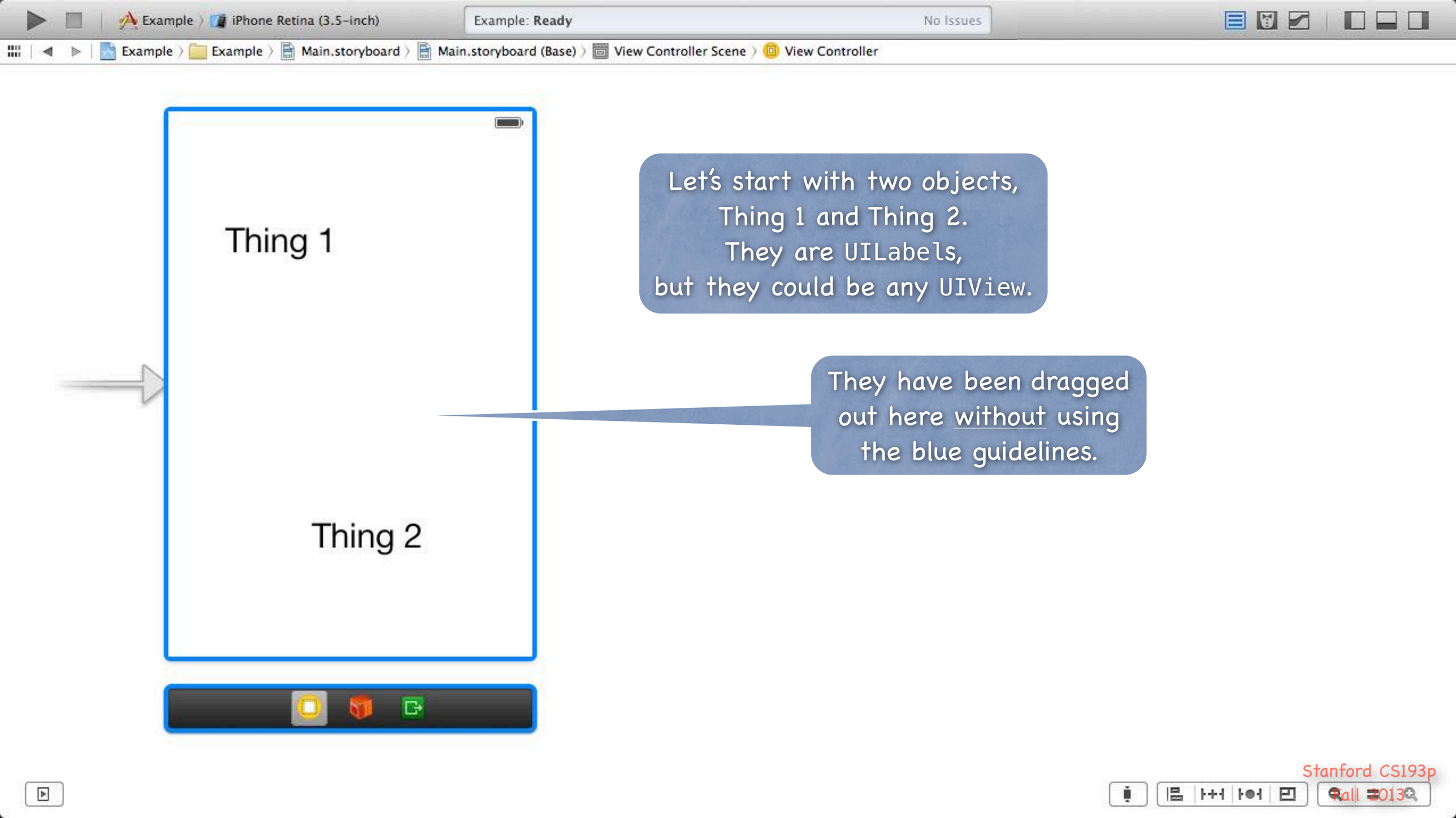
  We need these rules to put the views in their place no matter what bounds are available.

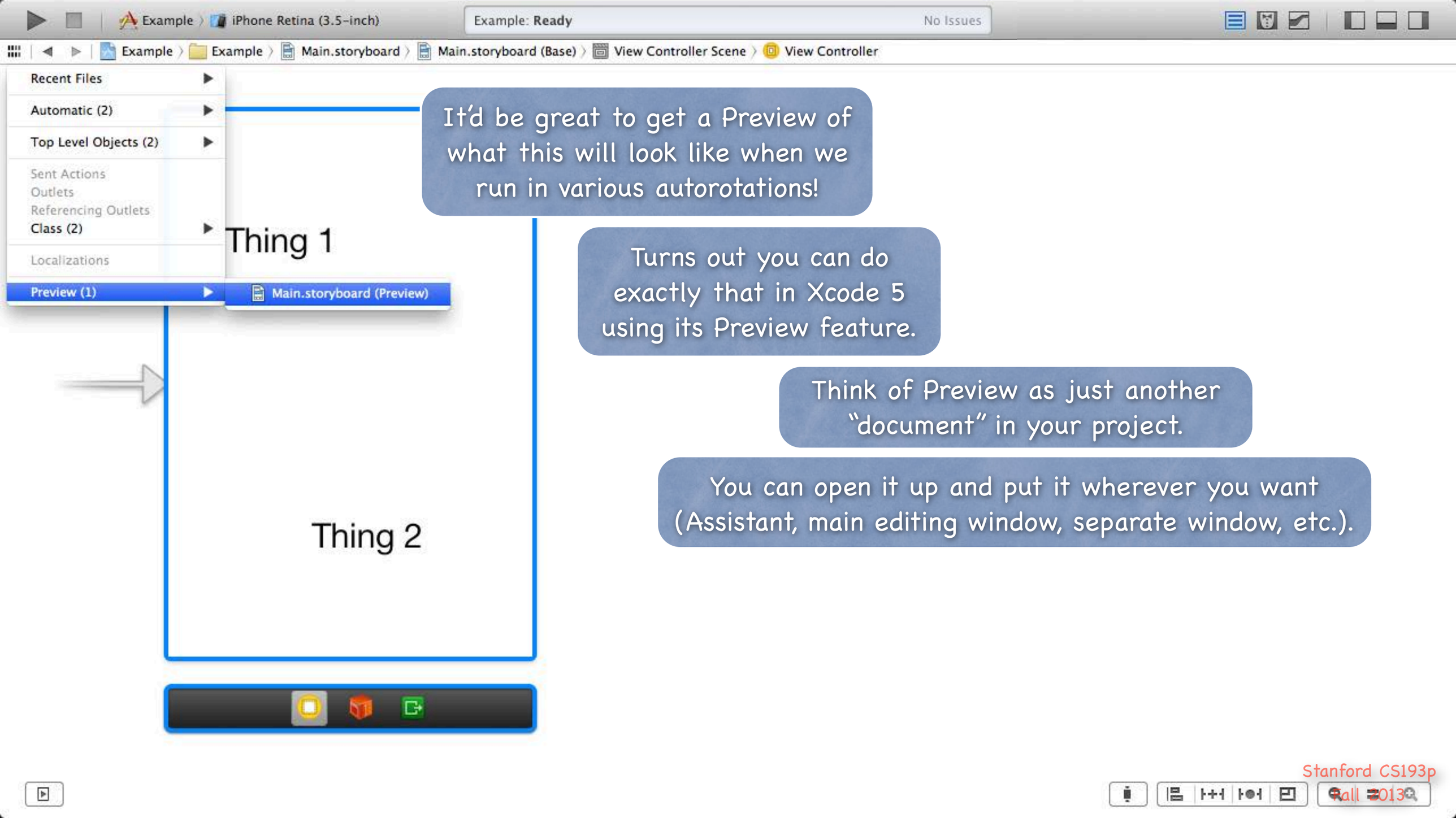
  We call these rules "constraints".

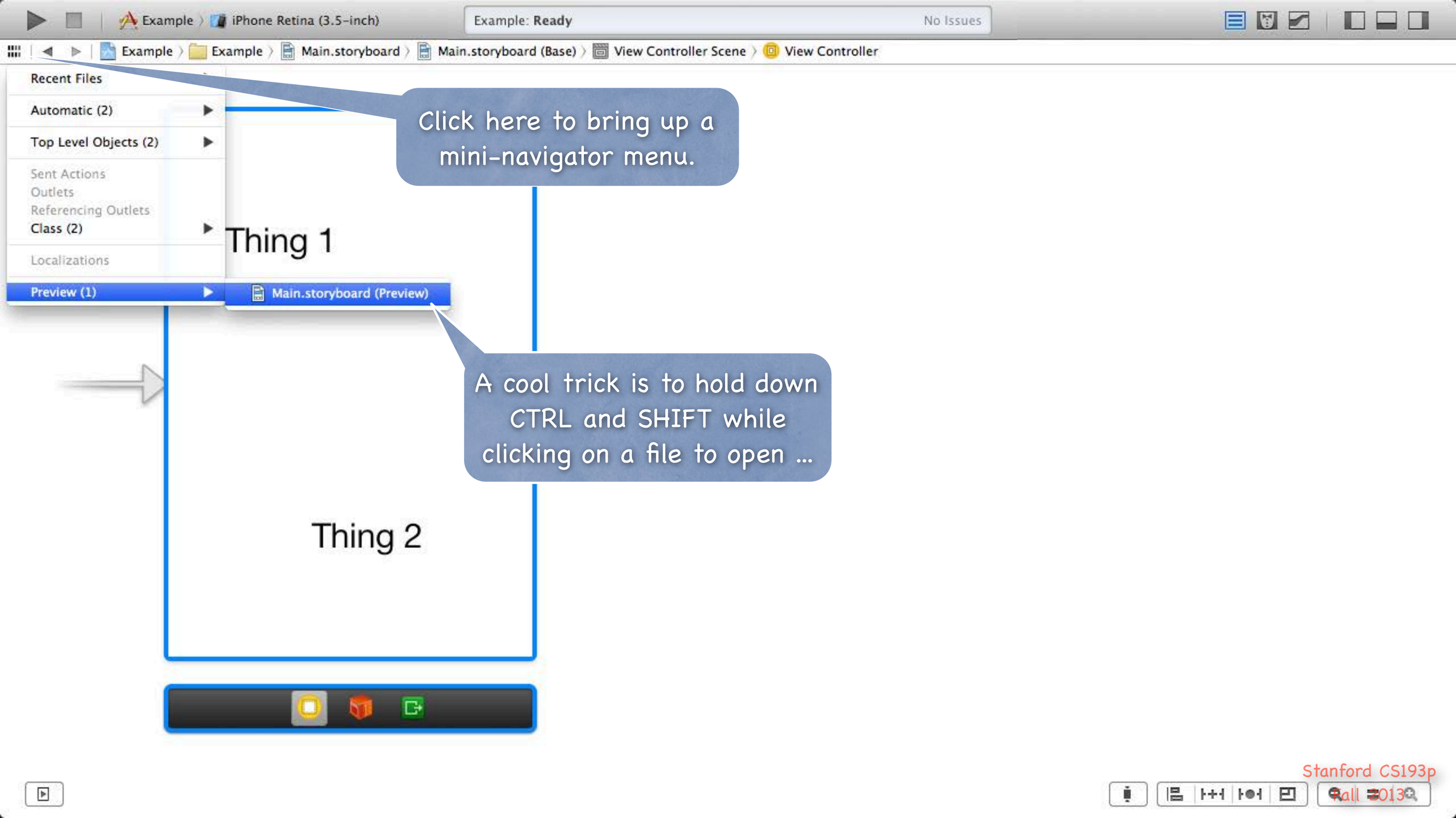
  There is a very powerful API (`NSLayoutConstraint`) for doing this, but …

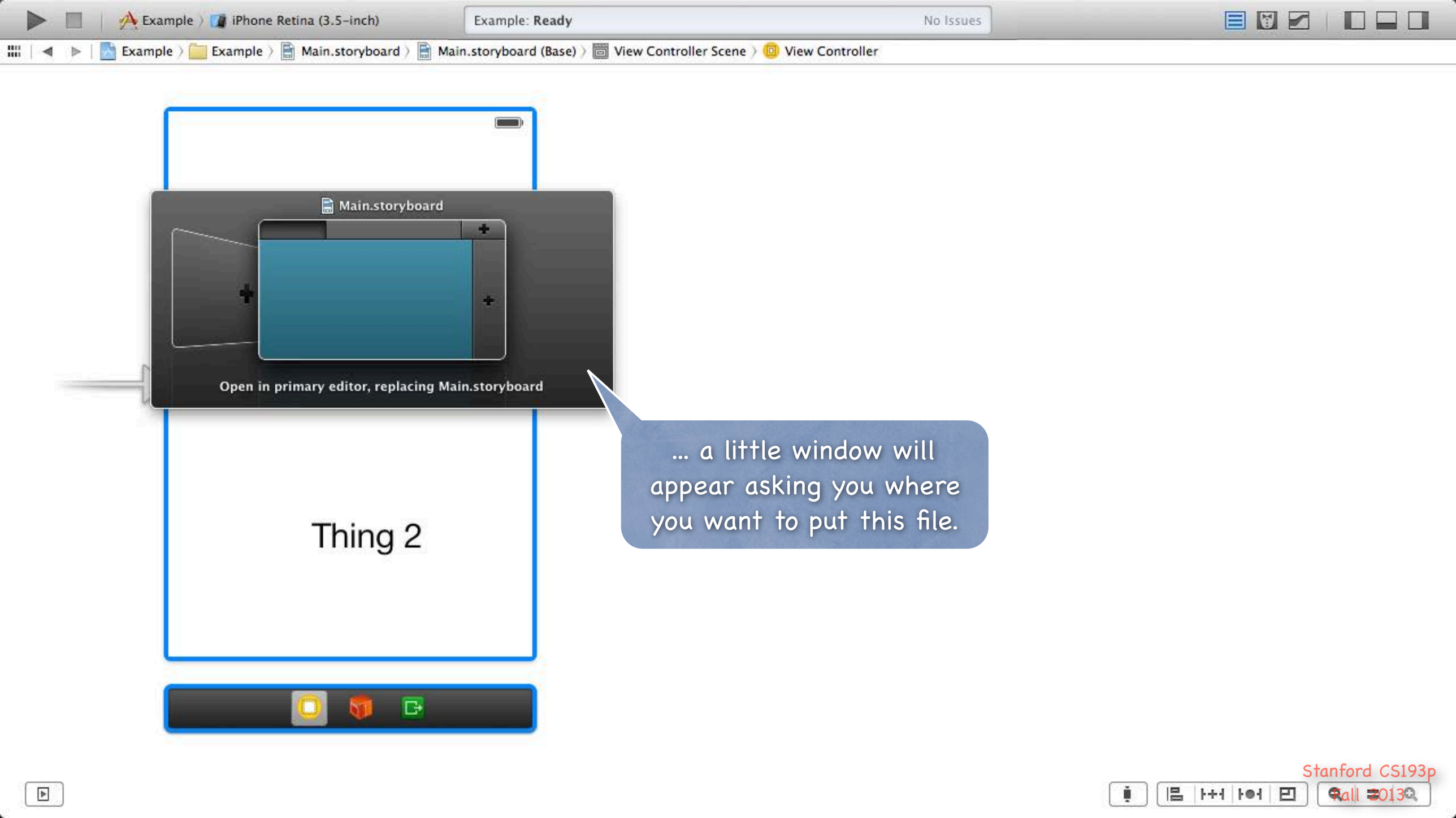
- We almost always set up these rules in Xcode 5 graphically

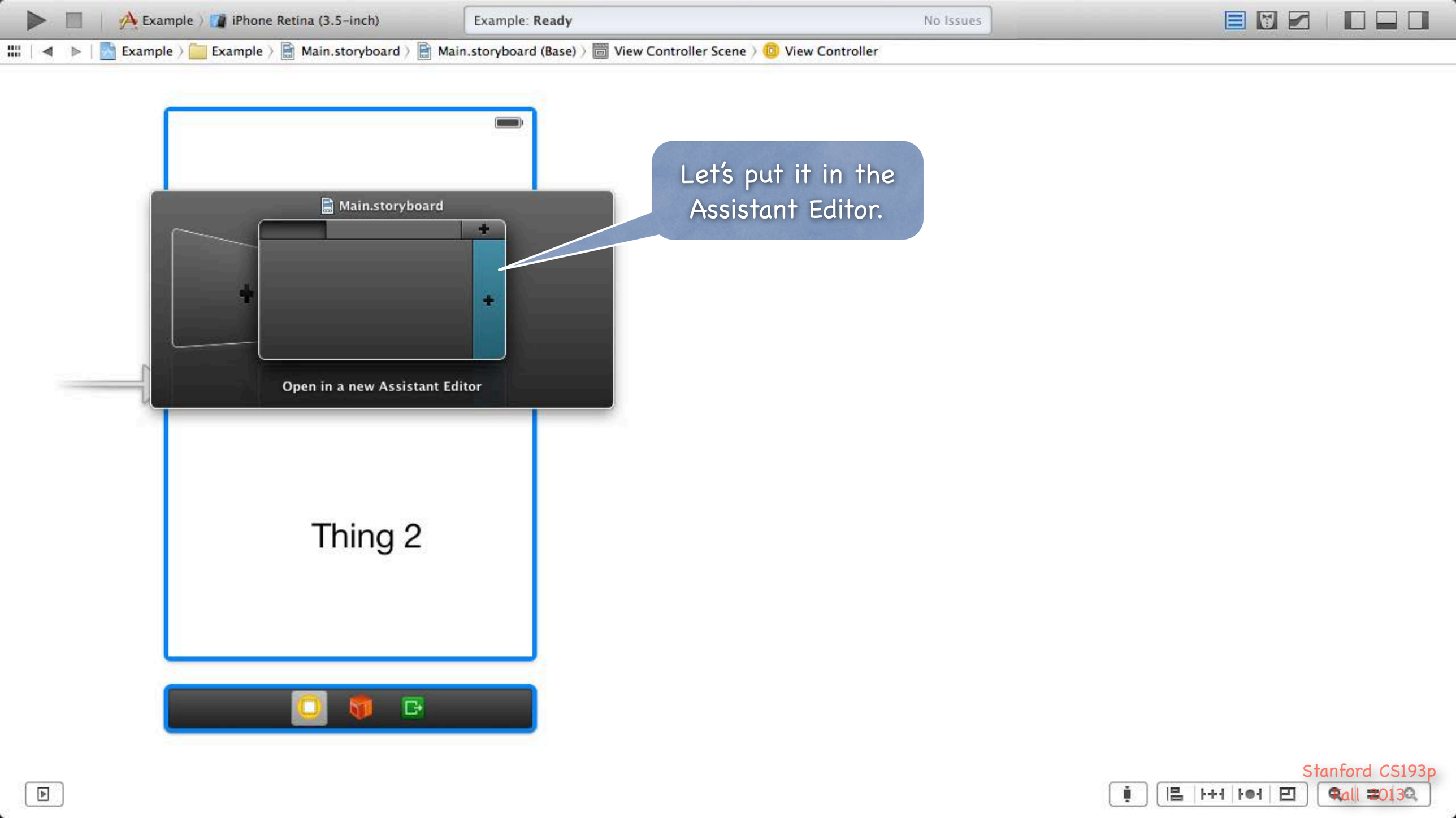
  So this is all best shown with some screen shots …

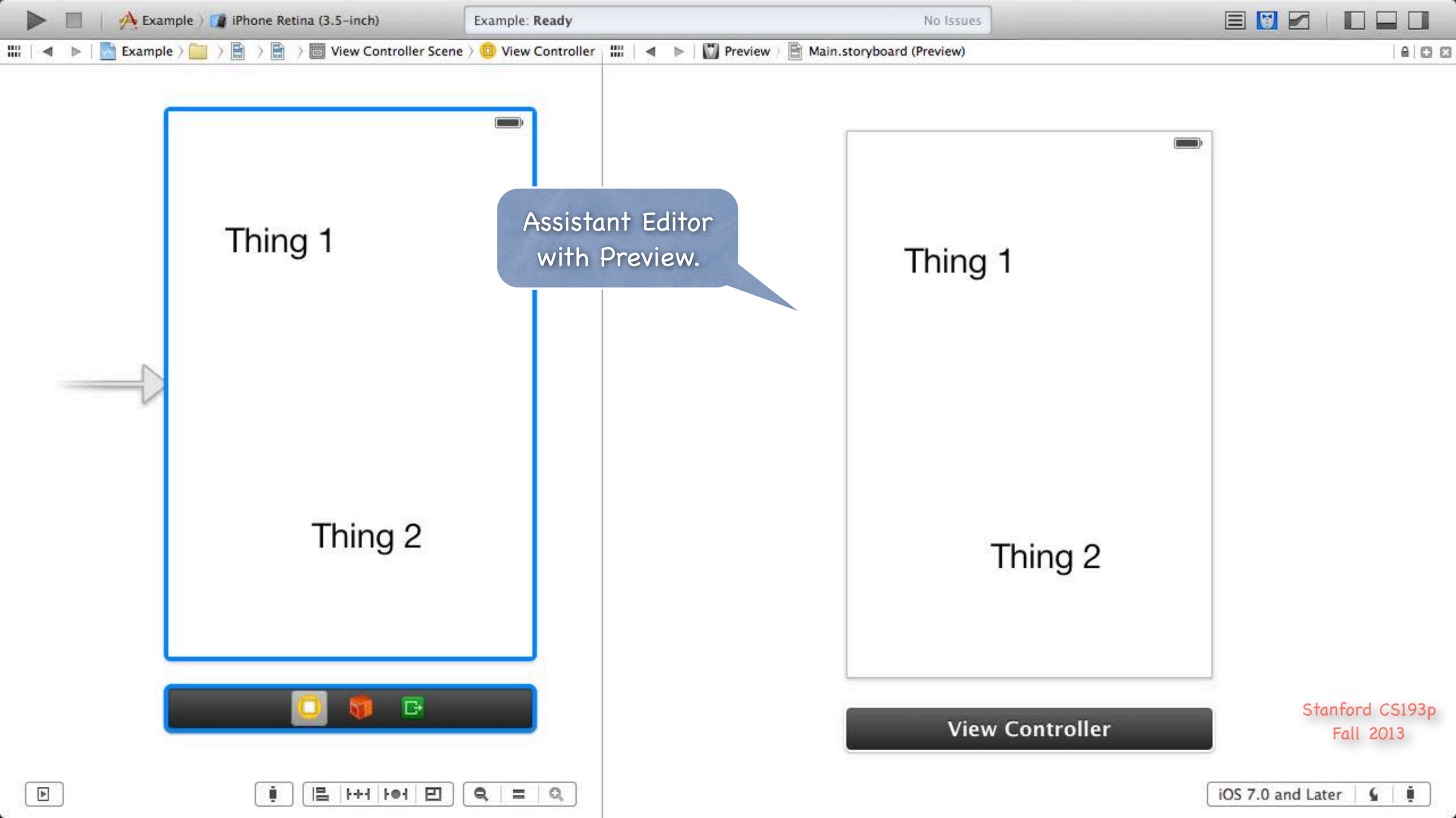

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Thing 2
Let's start with two objects,
Thing 1 and Thing 2.
They are UILabels,
but they could be any UIView.
They have been dragged
out here without using
the blue guidelines.

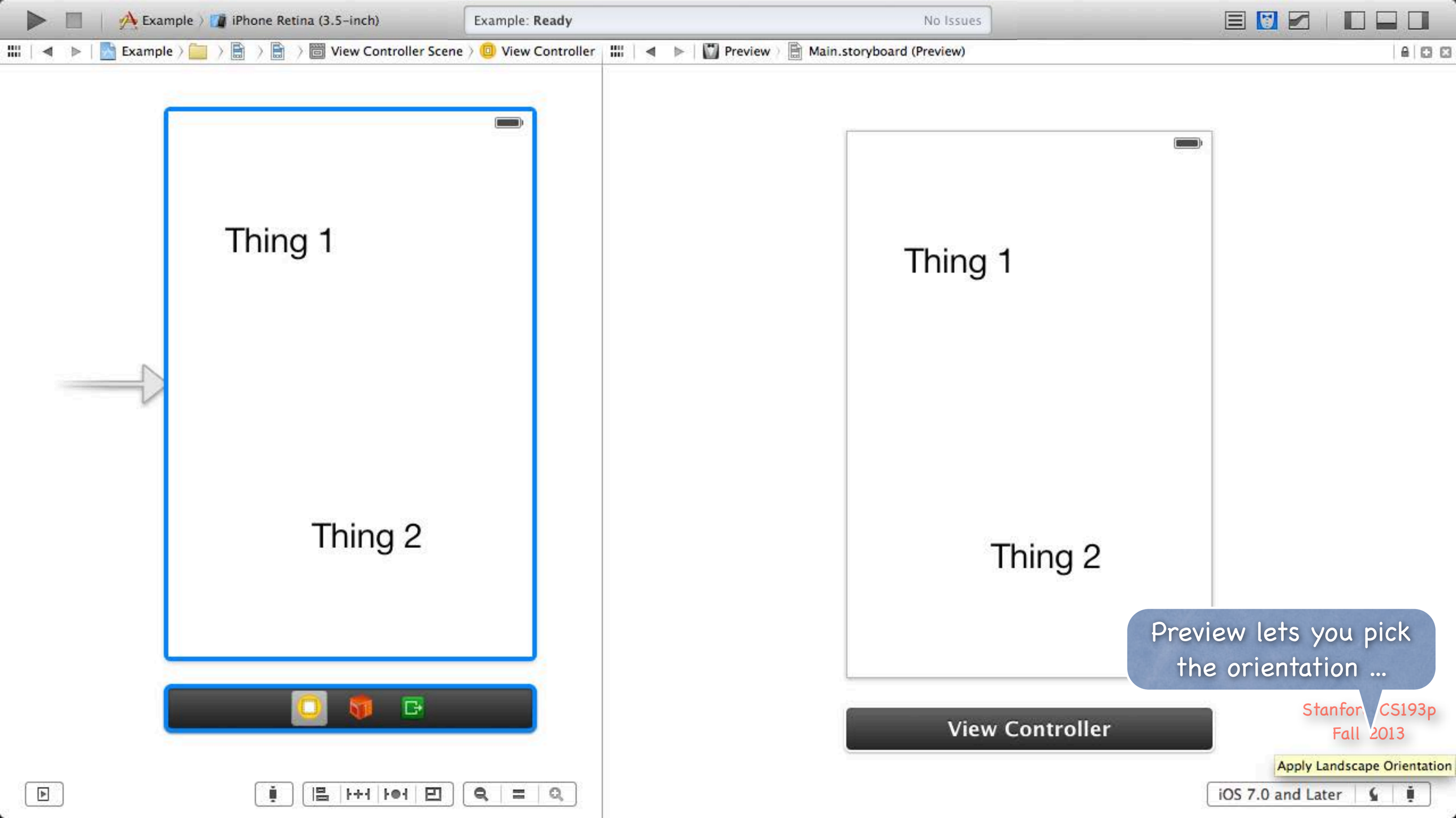

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Recent Files
Automatic (2)
Top Level Objects (2)
Sent Actions
Outlets
Referencing Outlets
Class (2)
Localizations
Preview (1)
Main.storyboard (Preview)
Thing 1
Thing 2
It'd be great to get a Preview of what this will look like when we run in various autorotations!
Turns out you can do exactly that in Xcode 5 using its Preview feature.
Think of Preview as just another "document" in your project.
You can open it up and put it wherever you want (Assistant, main editing window, separate window, etc.).

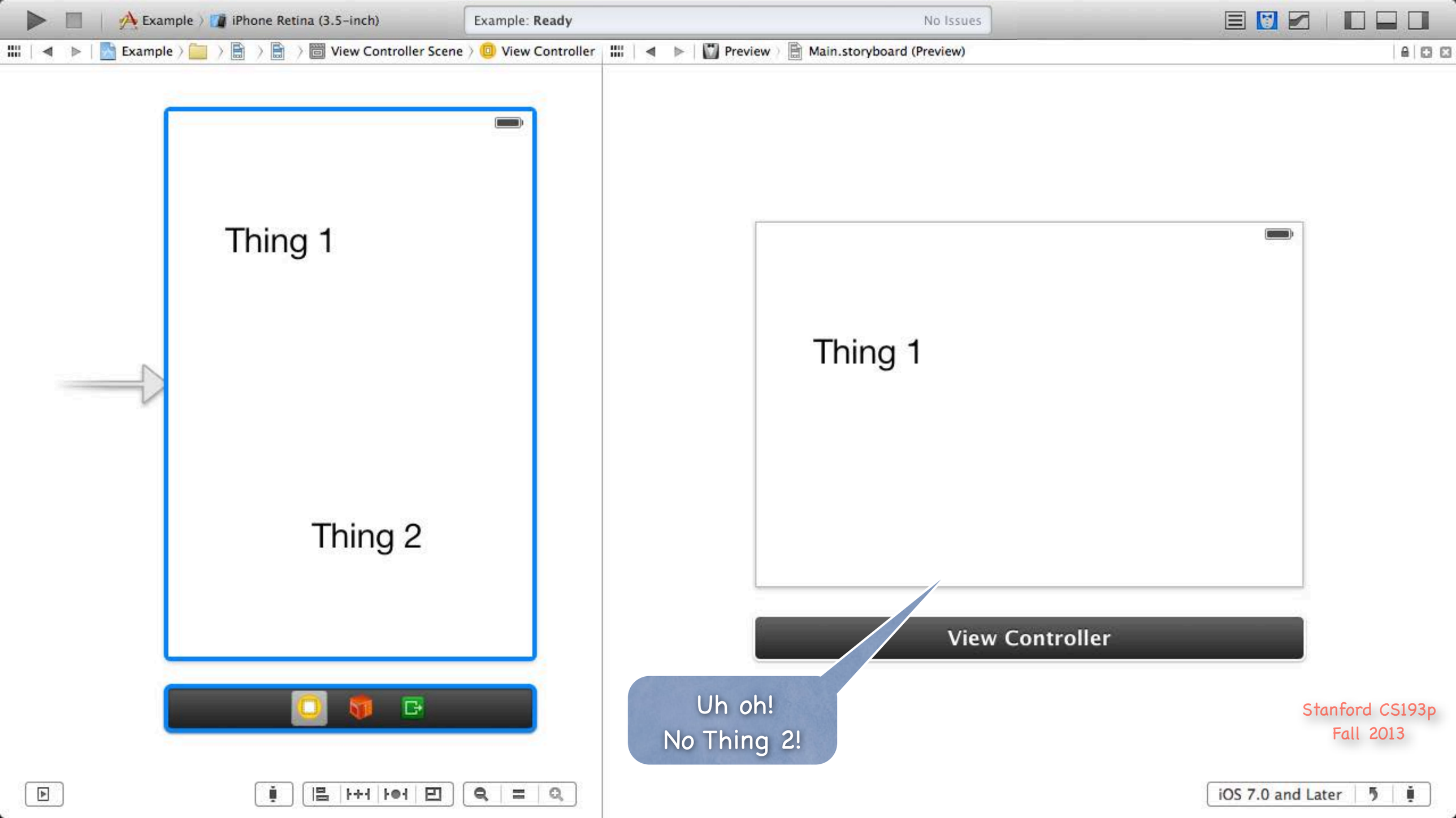

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Recent Files
Automatic (2)
Top Level Objects (2)
Sent Actions
Outlets
Referencing Outlets
Class (2)
Localizations
Preview (1)
Main.storyboard (Preview)
Click here to bring up a mini-navigator menu.
Thing 1
A cool trick is to hold down CTRL and SHIFT while clicking on a file to open ...
Thing 2

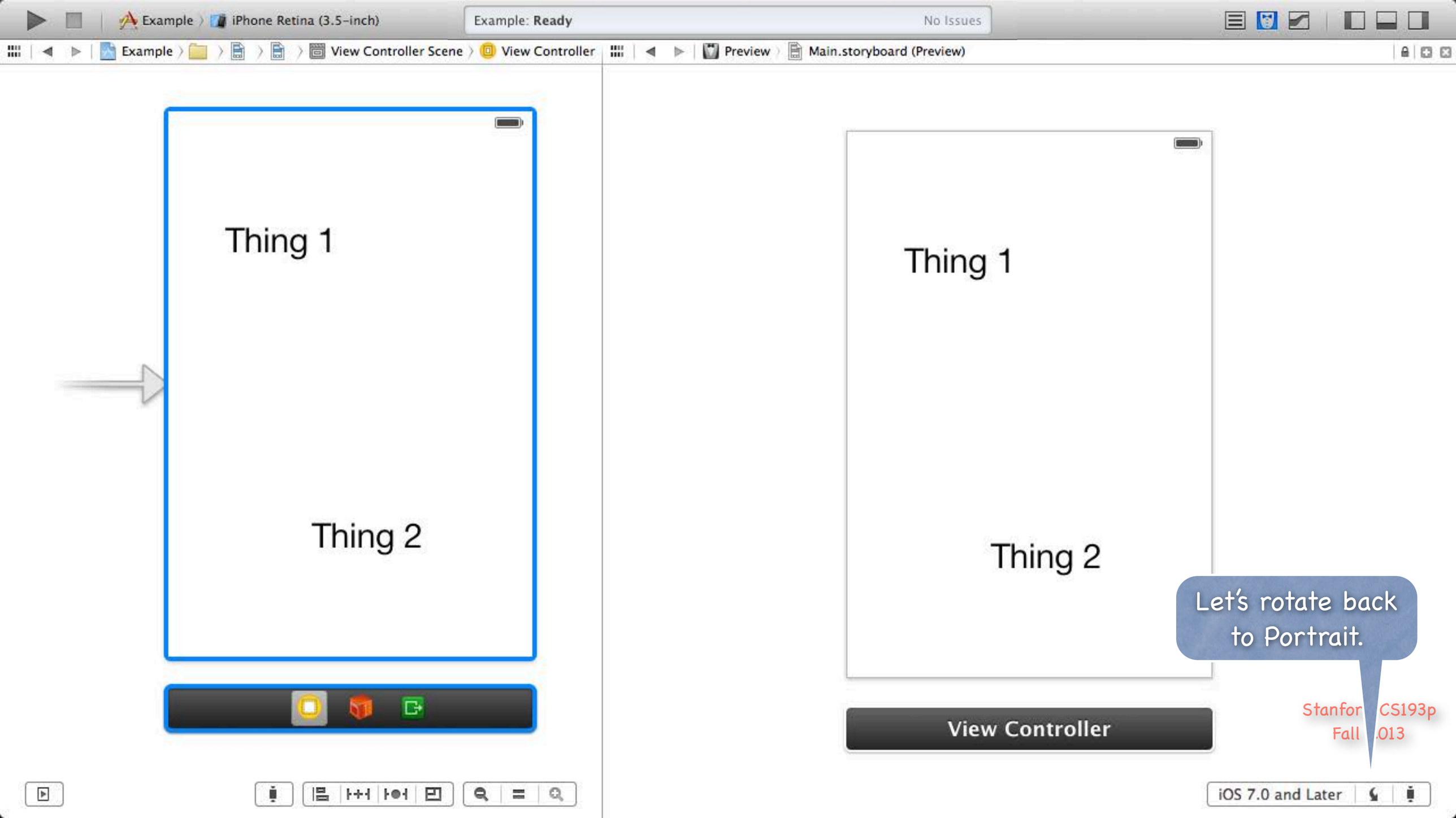

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Main.storyboard
Open in primary editor, replacing Main.storyboard
Thing 2
... a little window will appear asking you where you want to put this file.
Stanford CS193p
Fall 2013

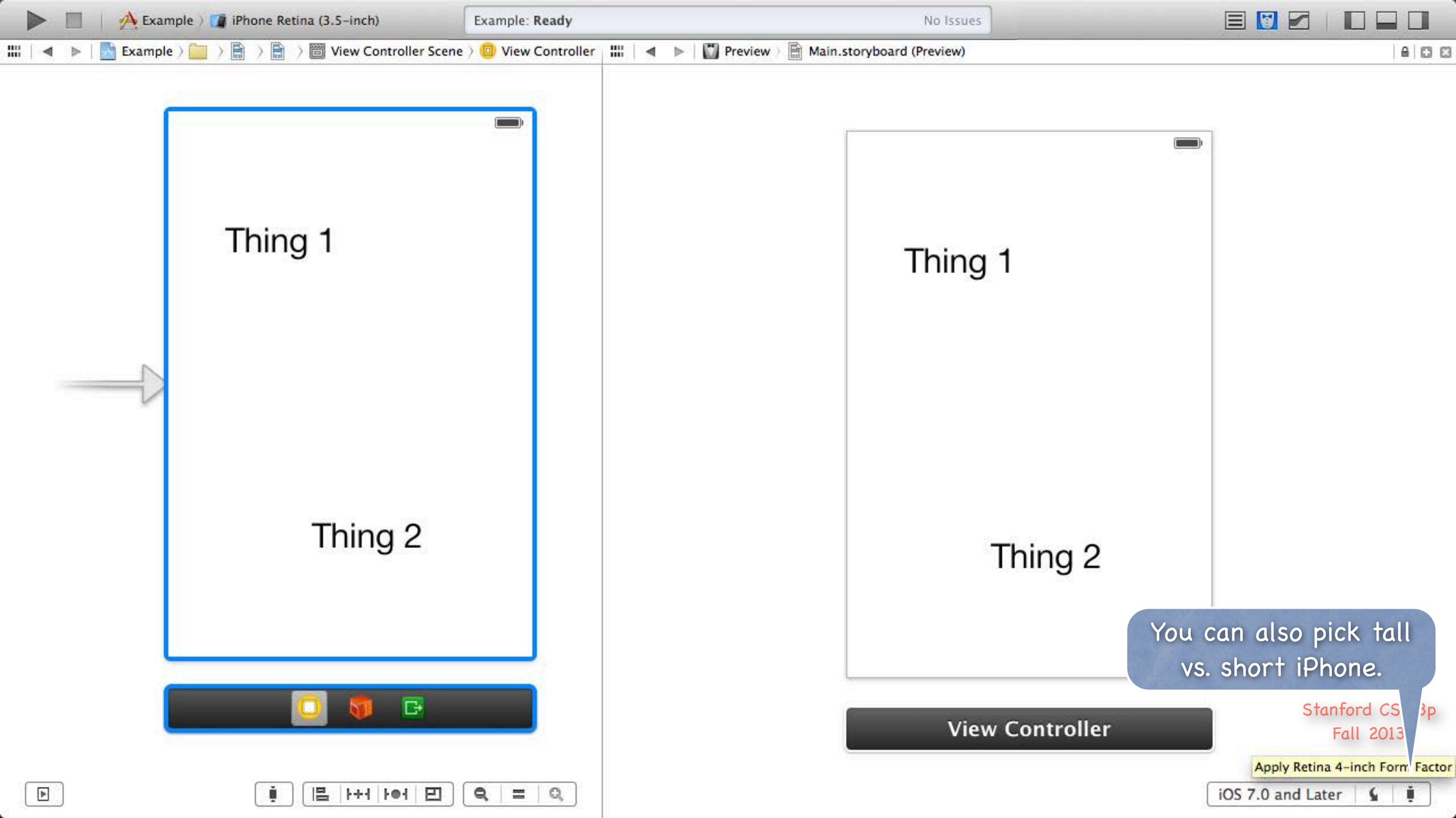

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Main.storyboard
Open in a new Assistant Editor
Let's put it in the Assistant Editor.
Thing 2
Stanford CS193p
Fall 2013

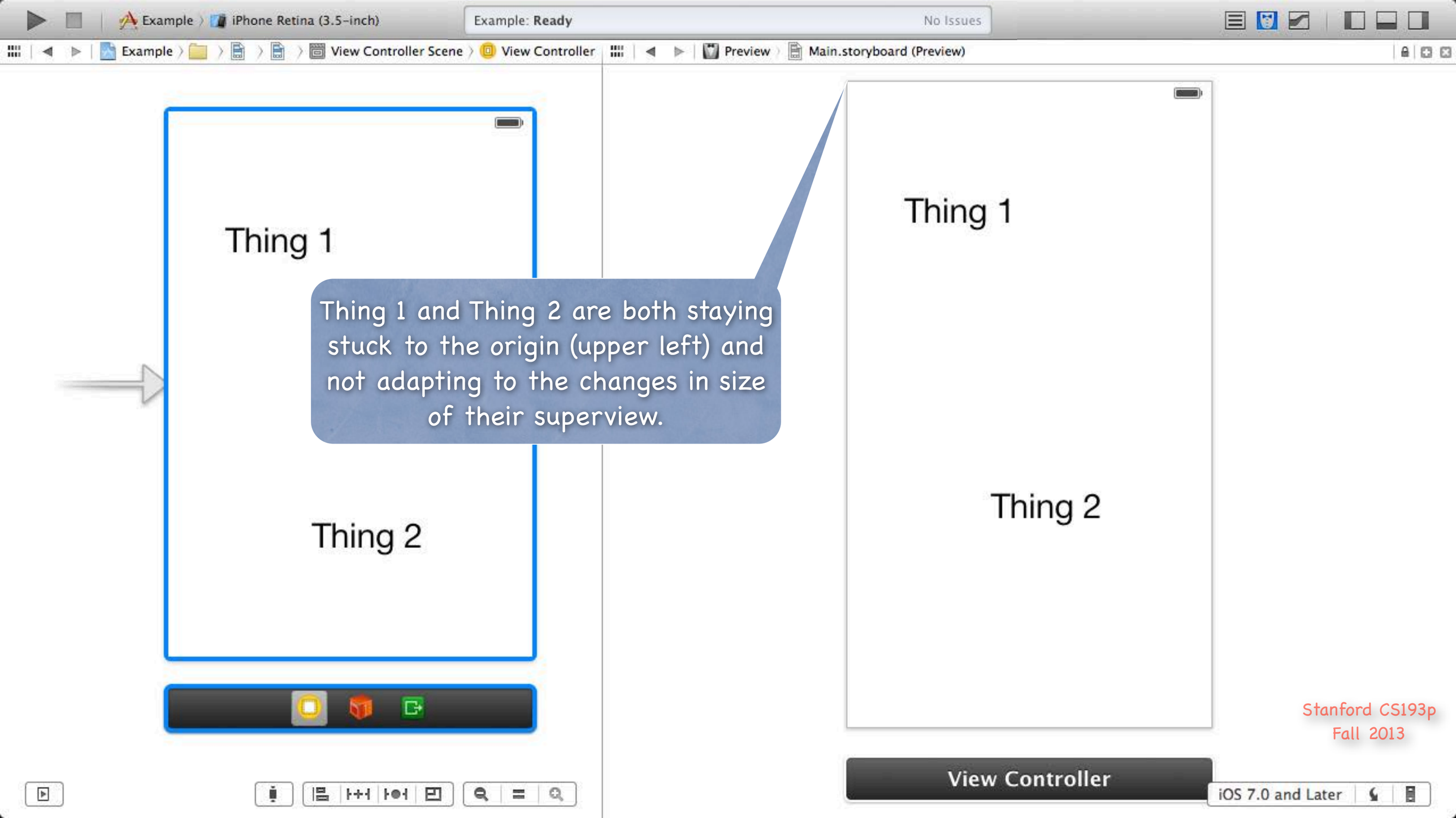

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
View Controller Scene
View Controller
Preview
Main.storyboard (Preview)
Thing 1
Thing 2
Assistant Editor with Preview.
Thing 1
Thing 2
View Controller
iOS 7.0 and Later
Stanford CS193p
Fall 2013

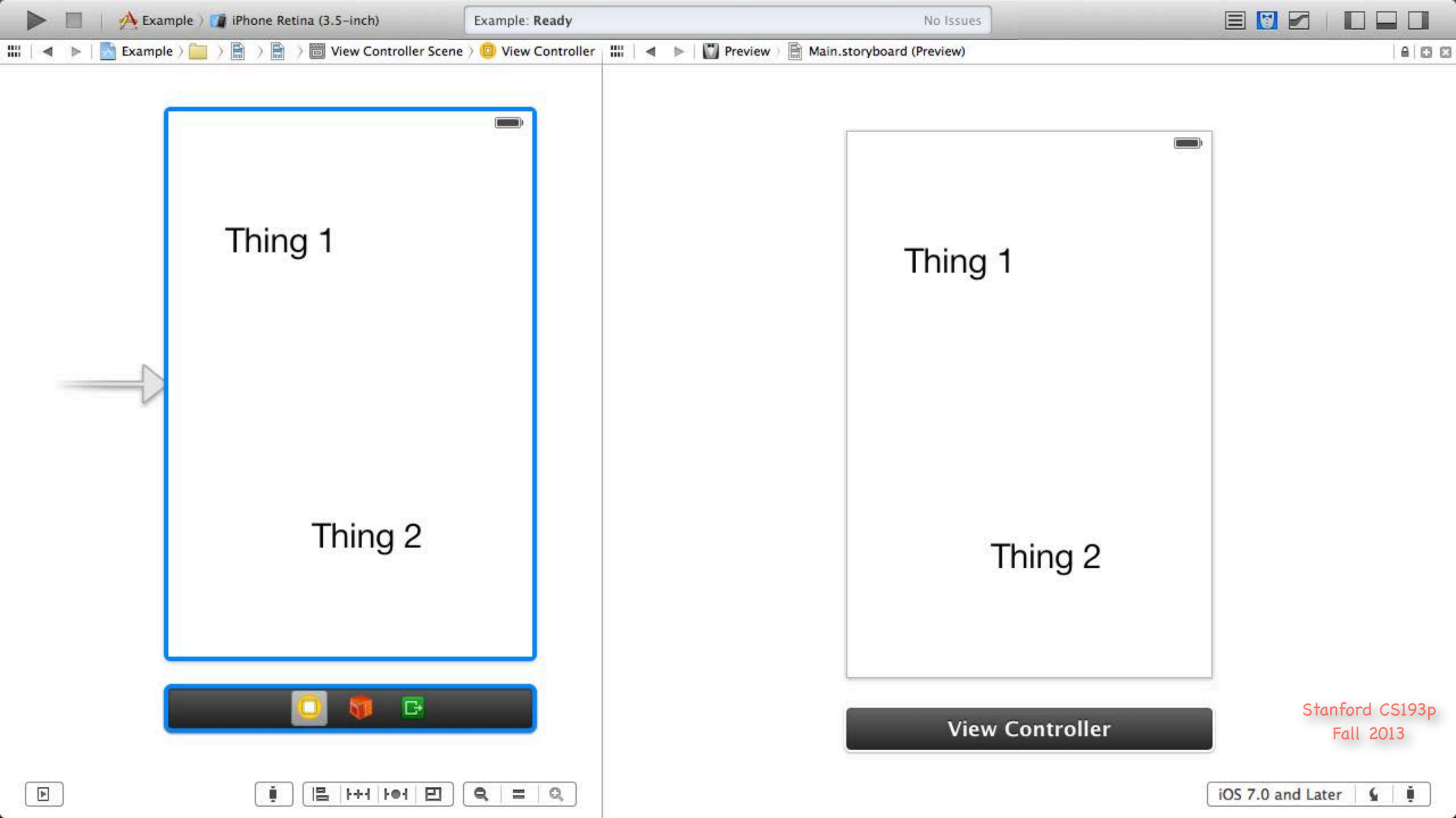

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
View Controller Scene
View Controller
Preview
Main.storyboard (Preview)
Thing 1
Thing 2
Thing 1
Thing 2
View Controller
Preview lets you pick the orientation ...
Stanfor CS193p
Fall 2013
Apply Landscape Orientation
iOS 7.0 and Later

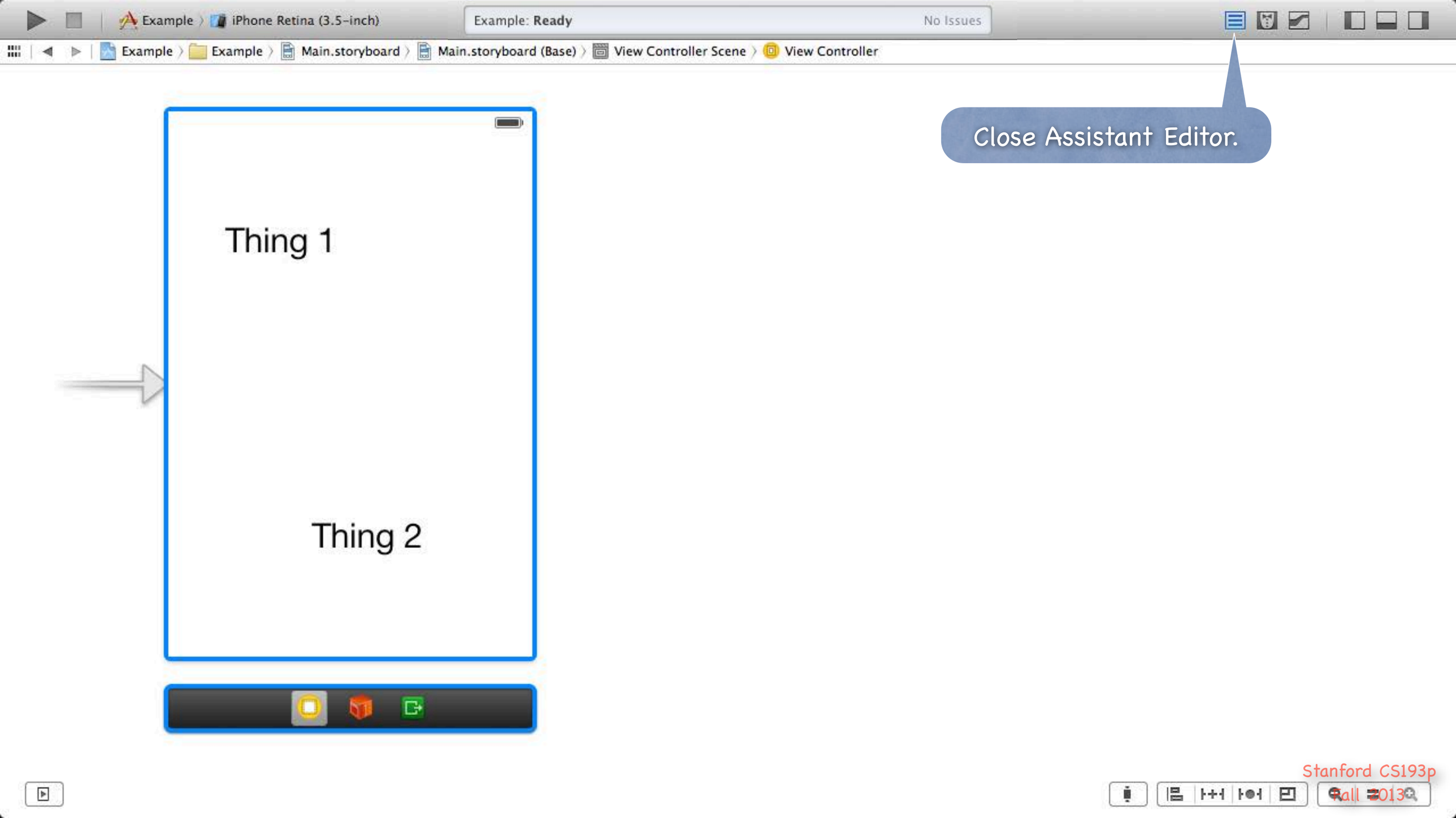

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
View Controller Scene
View Controller
Preview
Main.storyboard (Preview)
Thing 1
Thing 2
Thing 1
View Controller
Uh oh!
No Thing 2!
Stanford CS193p
Fall 2013
iOS 7.0 and Later

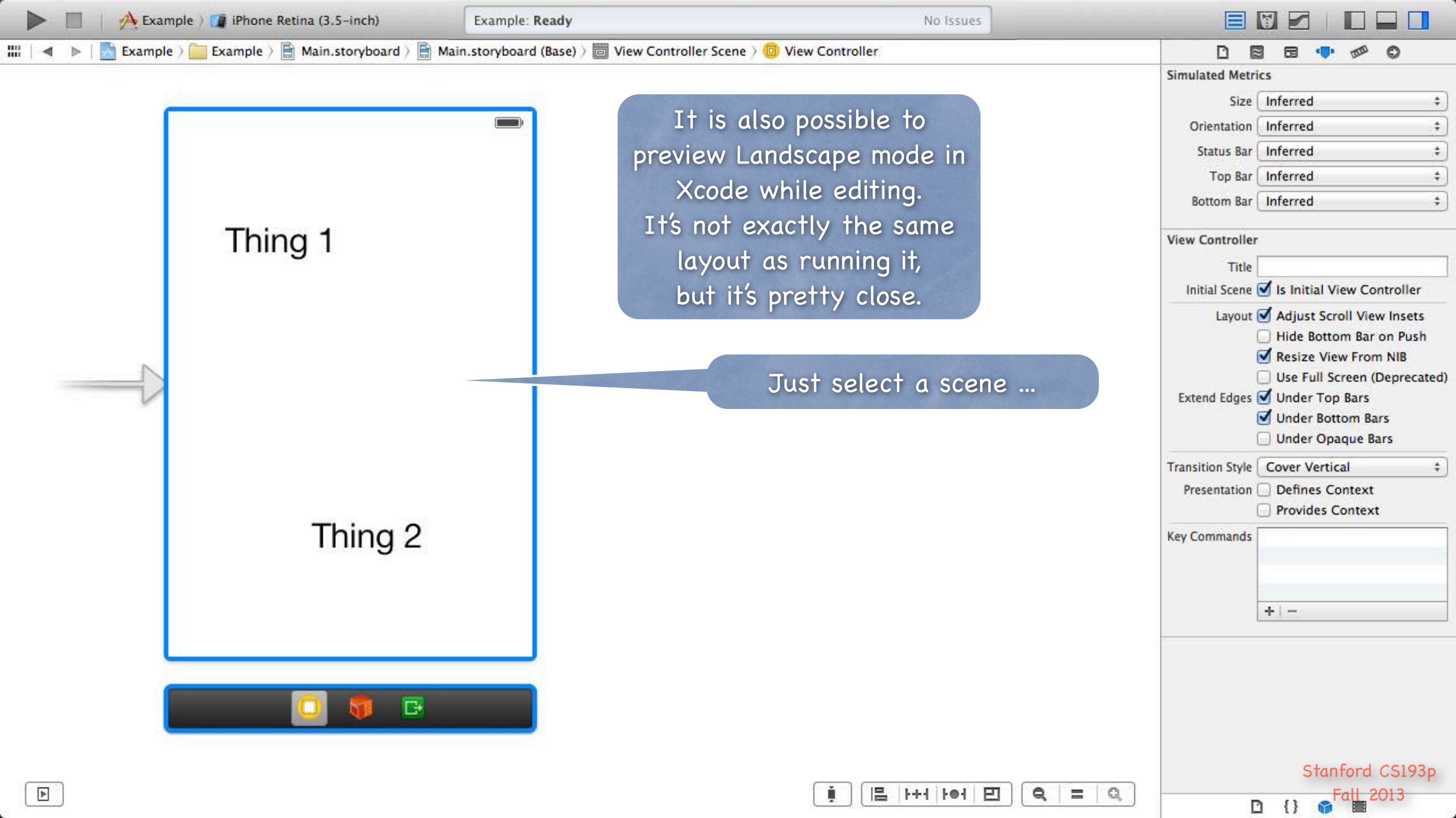

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
View Controller Scene
View Controller
Preview
Main.storyboard (Preview)
Thing 1
Thing 2
Thing 1
Thing 2
View Controller
Let's rotate back to Portrait.
Stanfor CS193p
Fall .013
iOS 7.0 and Later

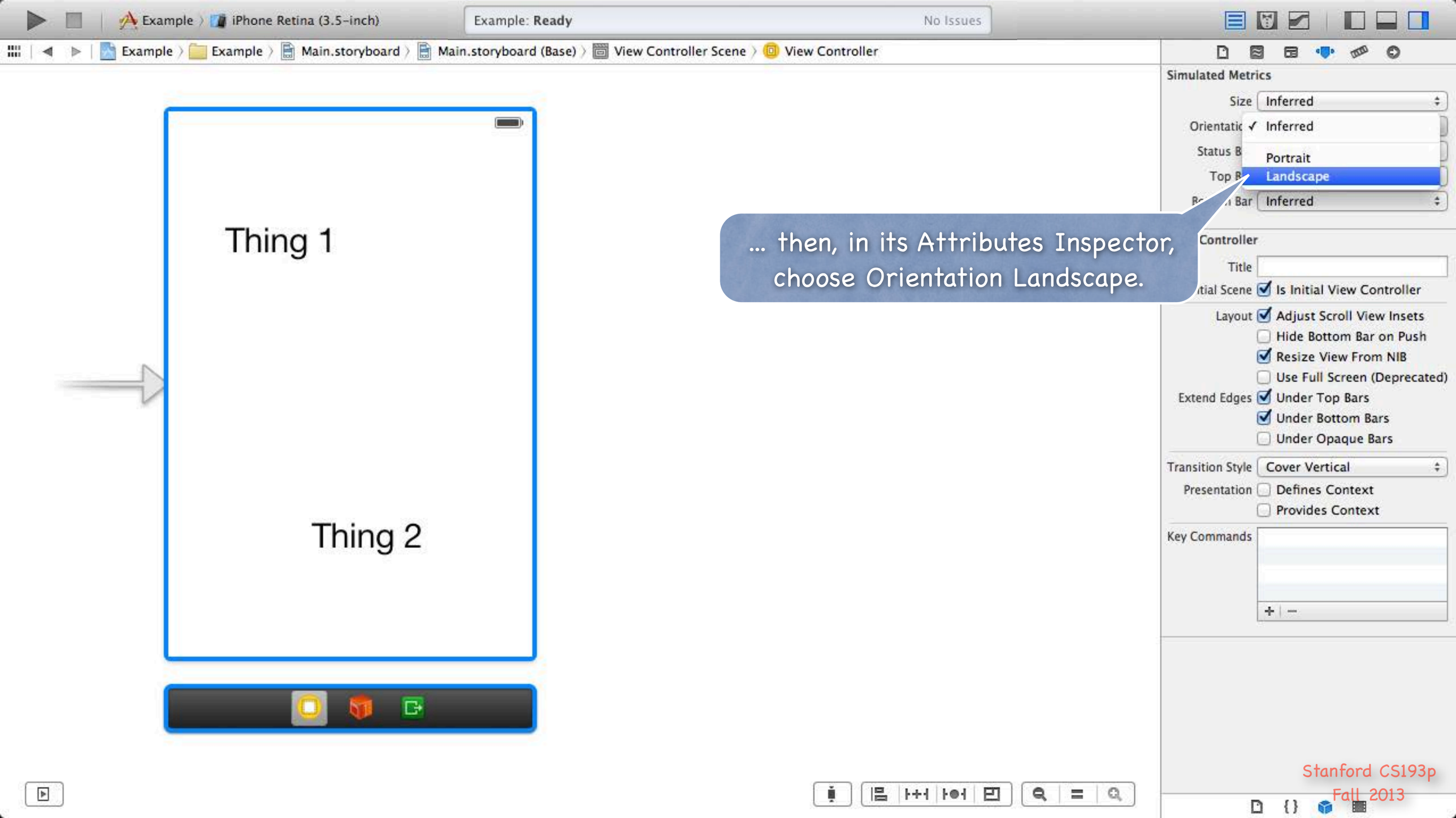

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
View Controller Scene
View Controller
Preview
Main.storyboard (Preview)
Thing 1
Thing 2
Thing 1
Thing 2
View Controller
You can also pick tall vs. short iPhone.
Stanford CS
Fall 2013
Apply Retina 4-inch Form Factor
iOS 7.0 and Later

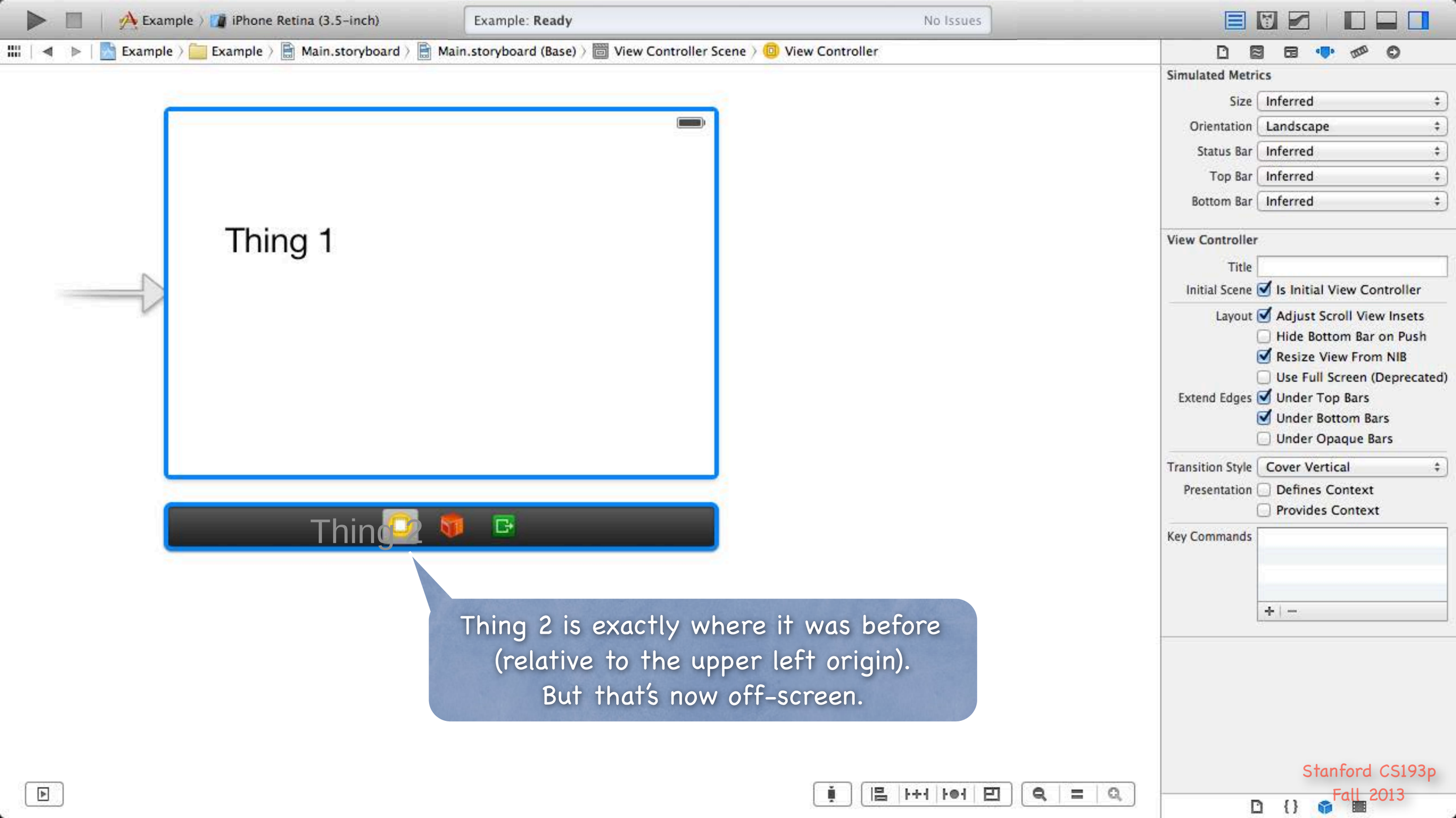

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
View Controller Scene
View Controller
Preview
Main.storyboard (Preview)
Thing 1
Thing 2
Thing 1 and Thing 2 are both staying stuck to the origin (upper left) and not adapting to the changes in size of their superview.
Thing 1
Thing 2
View Controller
iOS 7.0 and Later
Stanford CS193p
Fall 2013

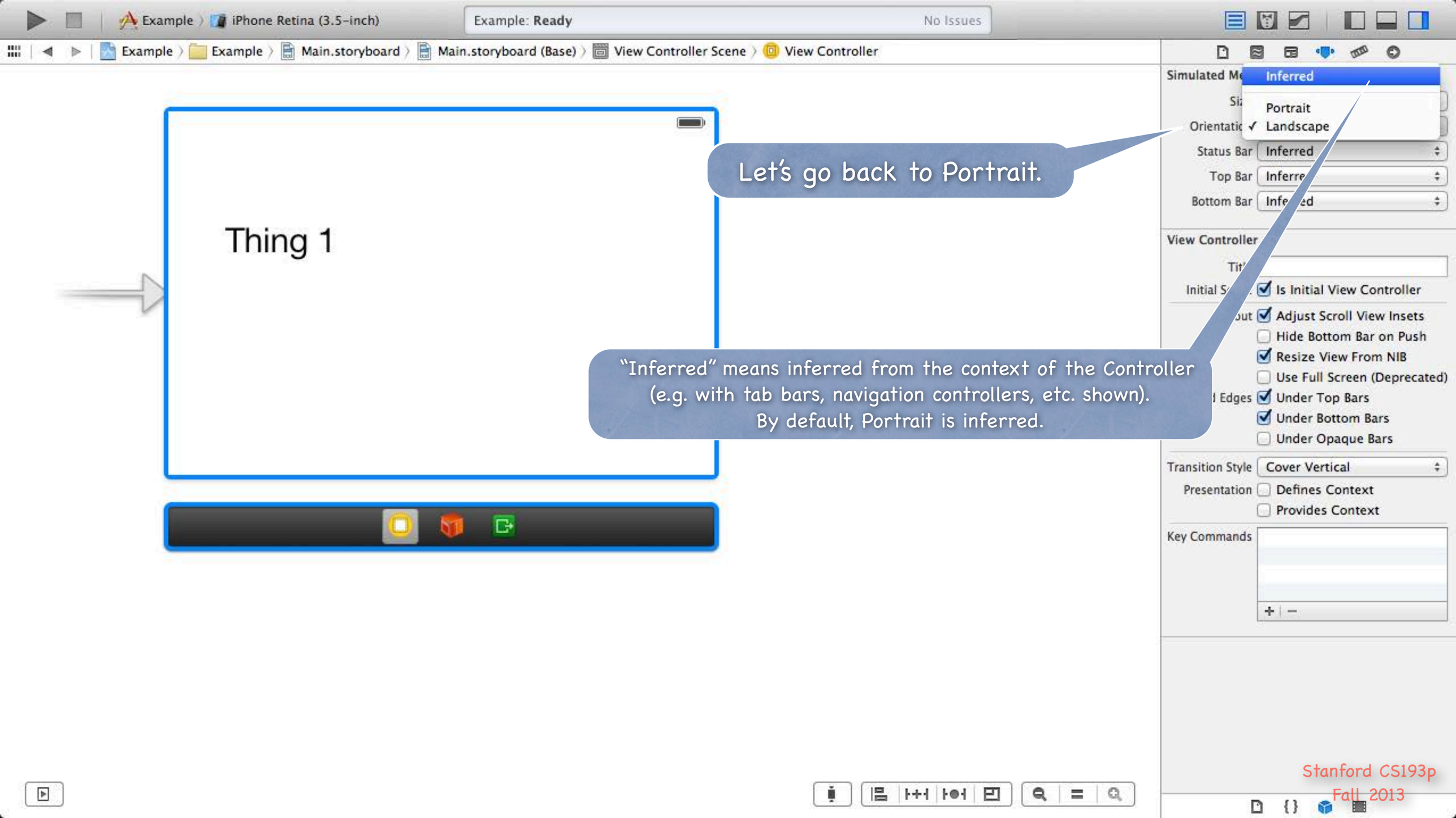

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
View Controller Scene
View Controller
Preview
Main.storyboard (Preview)
Thing 1
Thing 2
Thing 1
Thing 2
View Controller
iOS 7.0 and Later
Stanford CS193p
Fall 2013

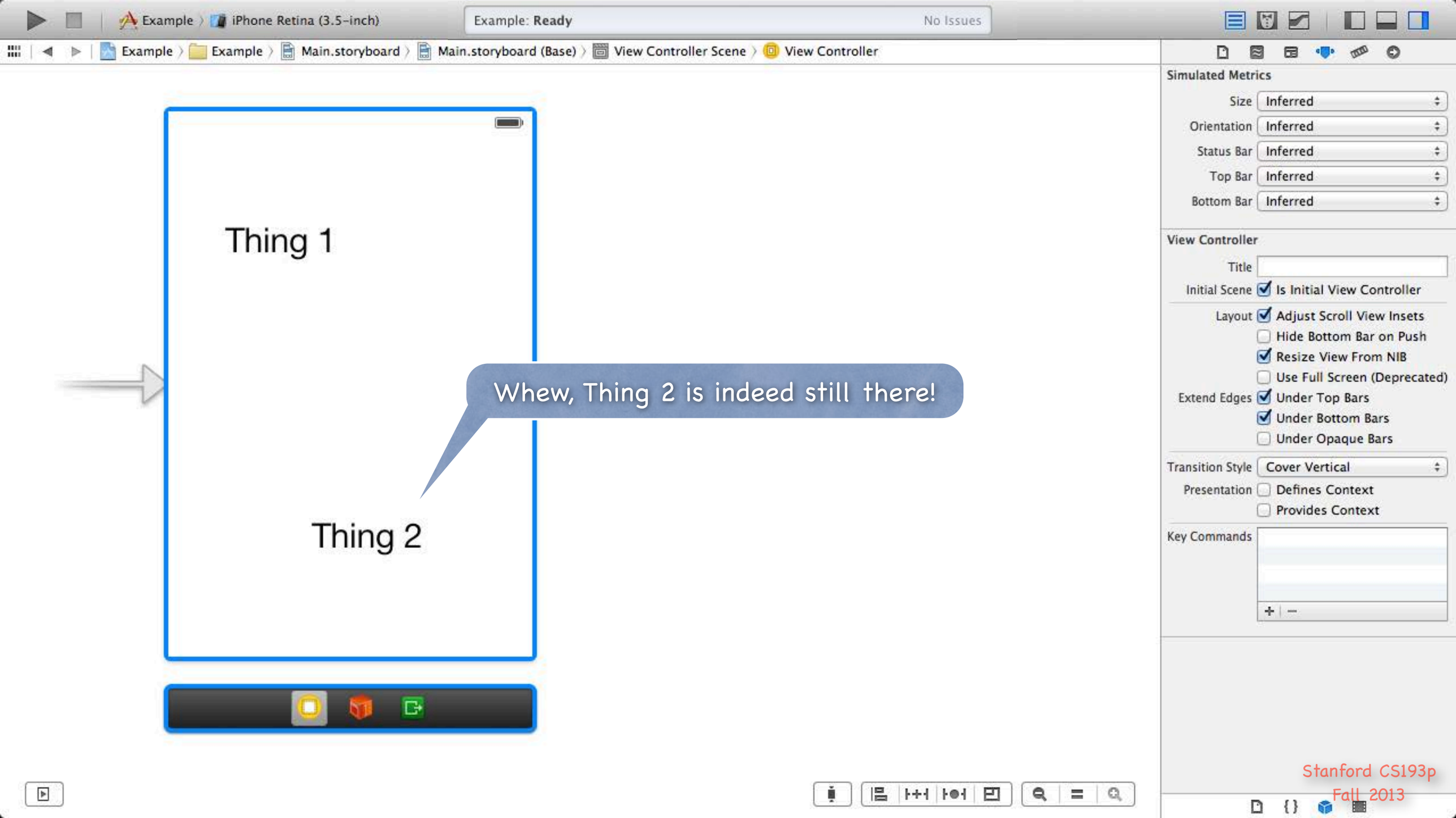

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Close Assistant Editor.
Thing 1
Thing 2
Stanford CS193p
Fall 2013

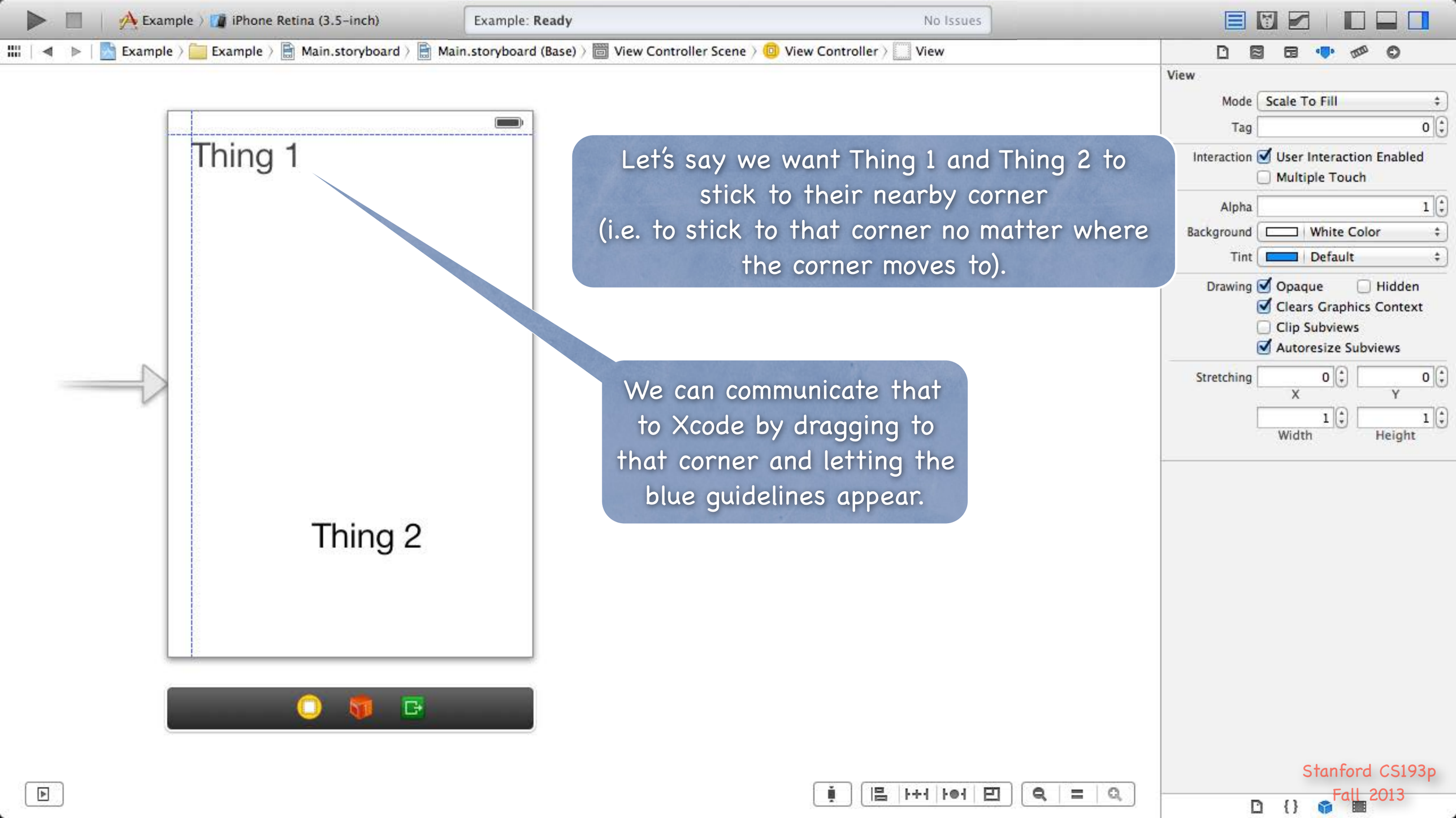

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Thing 2
It is also possible to preview Landscape mode in Xcode while editing. It's not exactly the same layout as running it, but it's pretty close.
Just select a scene ...
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands

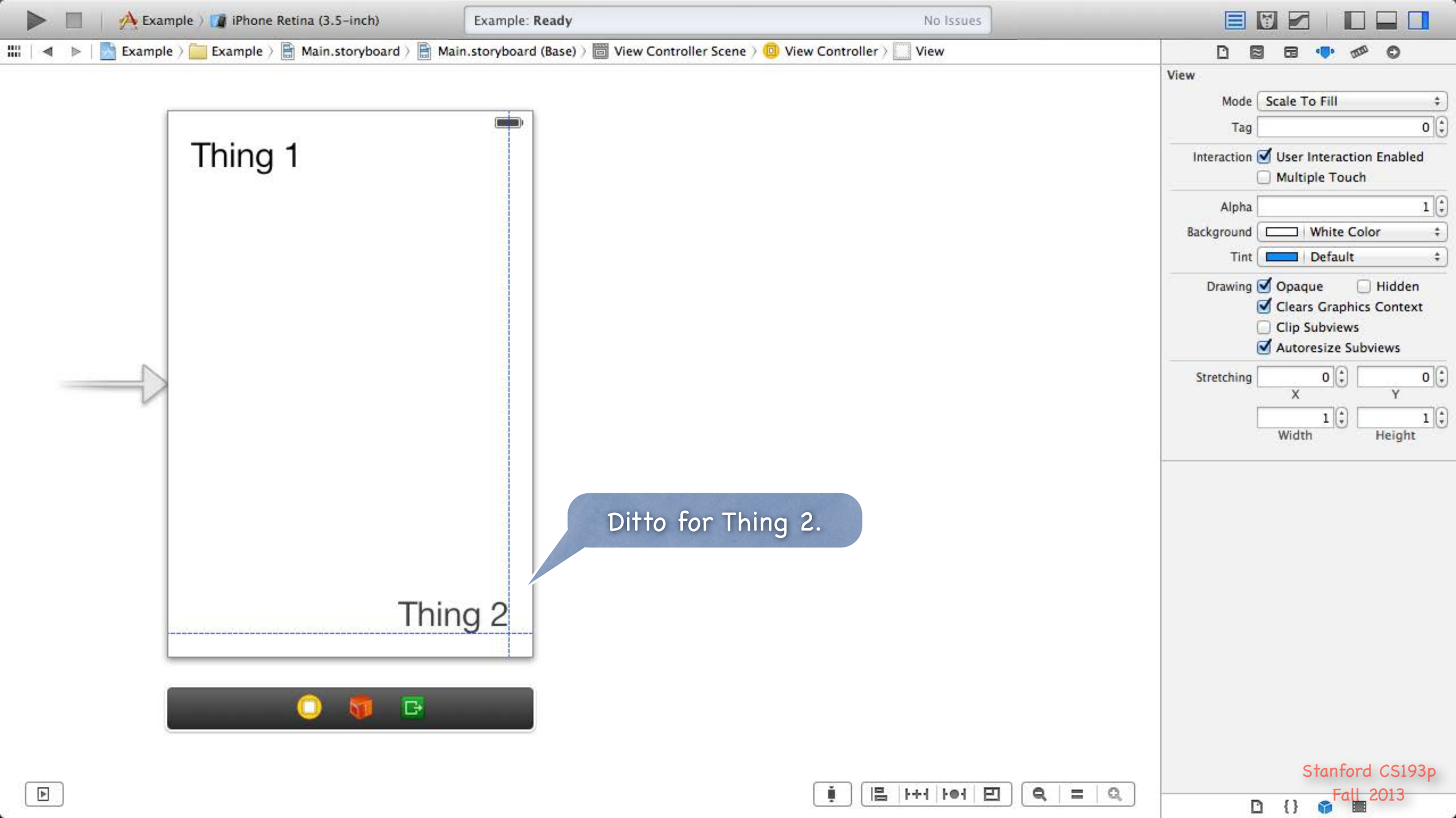

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Thing 1
Thing 2
Simulated Metrics
Size
Inferred
Orientation
Inferred
Portrait
Landscape
Status Bar
Inferred
Controller
Title
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
... then, in its Attributes Inspector, choose Orientation Landscape.
Stanford CS193p
Fall 2013

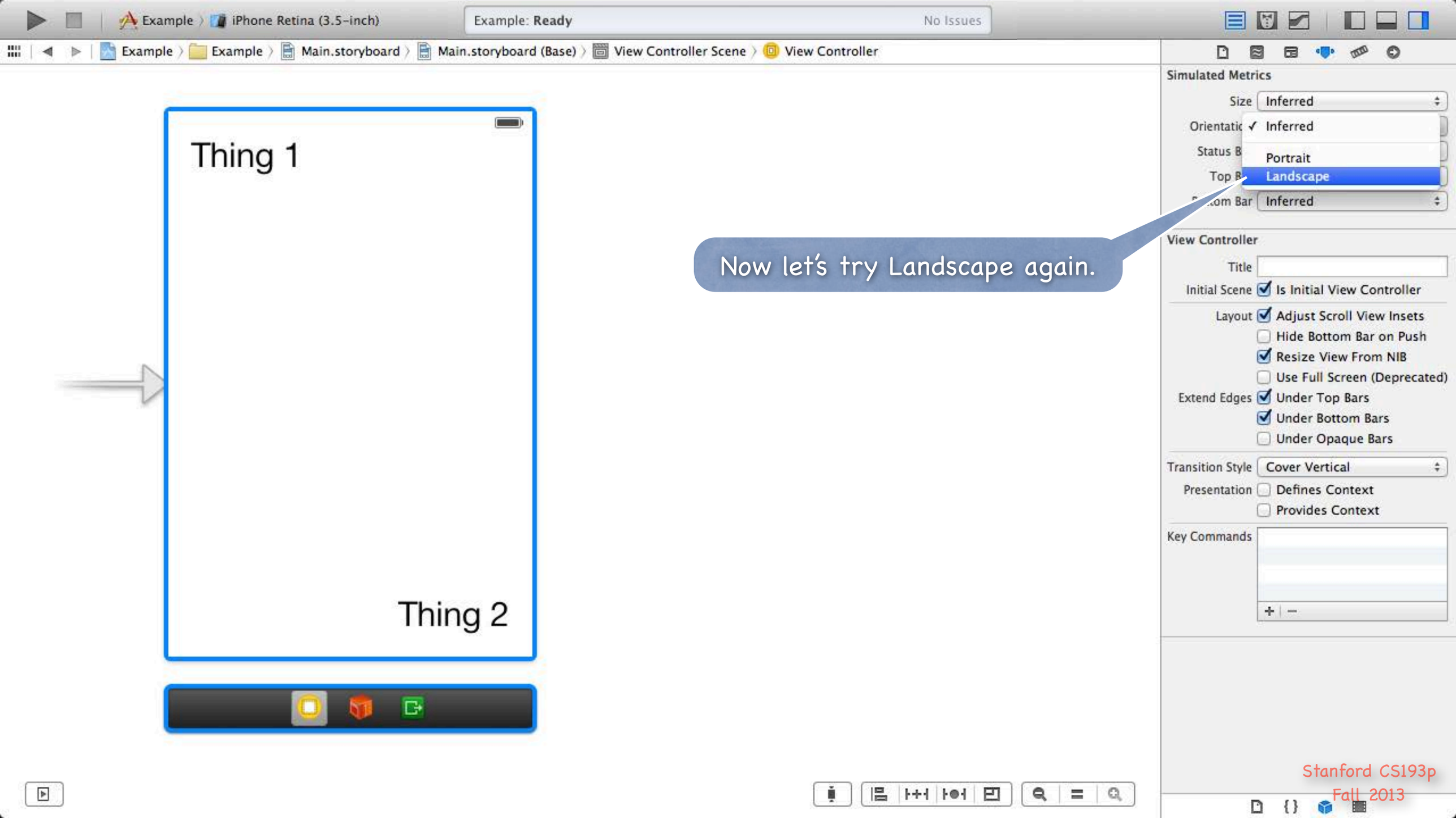

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Thing 2
Thing 2 is exactly where it was before
(relative to the upper left origin).
But that's now off-screen.
Simulated Metrics
Size Inferred
Orientation Landscape
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

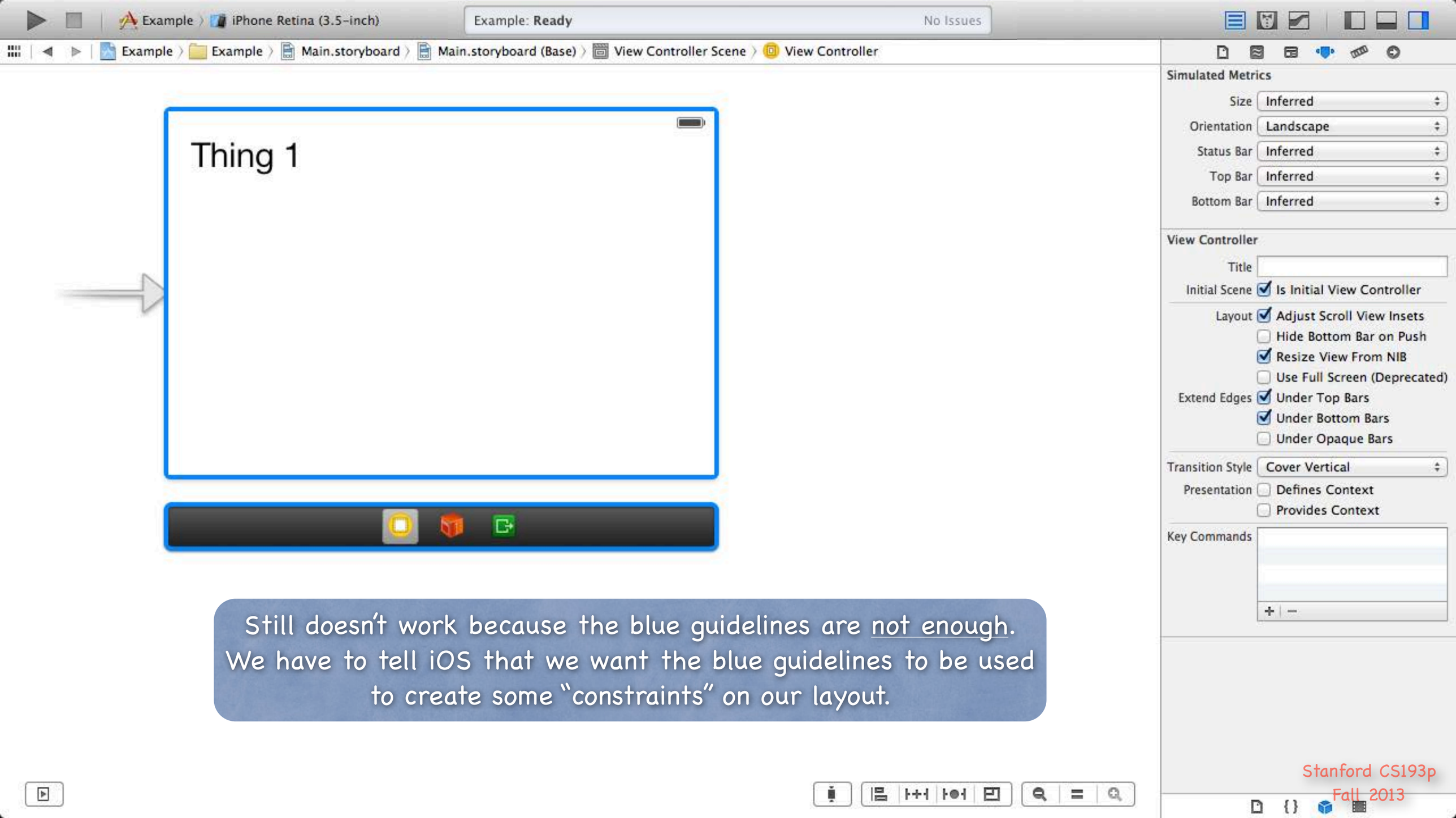

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Let's go back to Portrait.
"Inferred" means inferred from the context of the Controller
(e.g. with tab bars, navigation controllers, etc. shown).
By default, Portrait is inferred.
Simulated Me
Inferred
Portrait
Landscape
Orientatic
Status Bar
Inferred
Top Bar
Bottom Bar
View Controller
Initial S
Is Initial View Controller
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

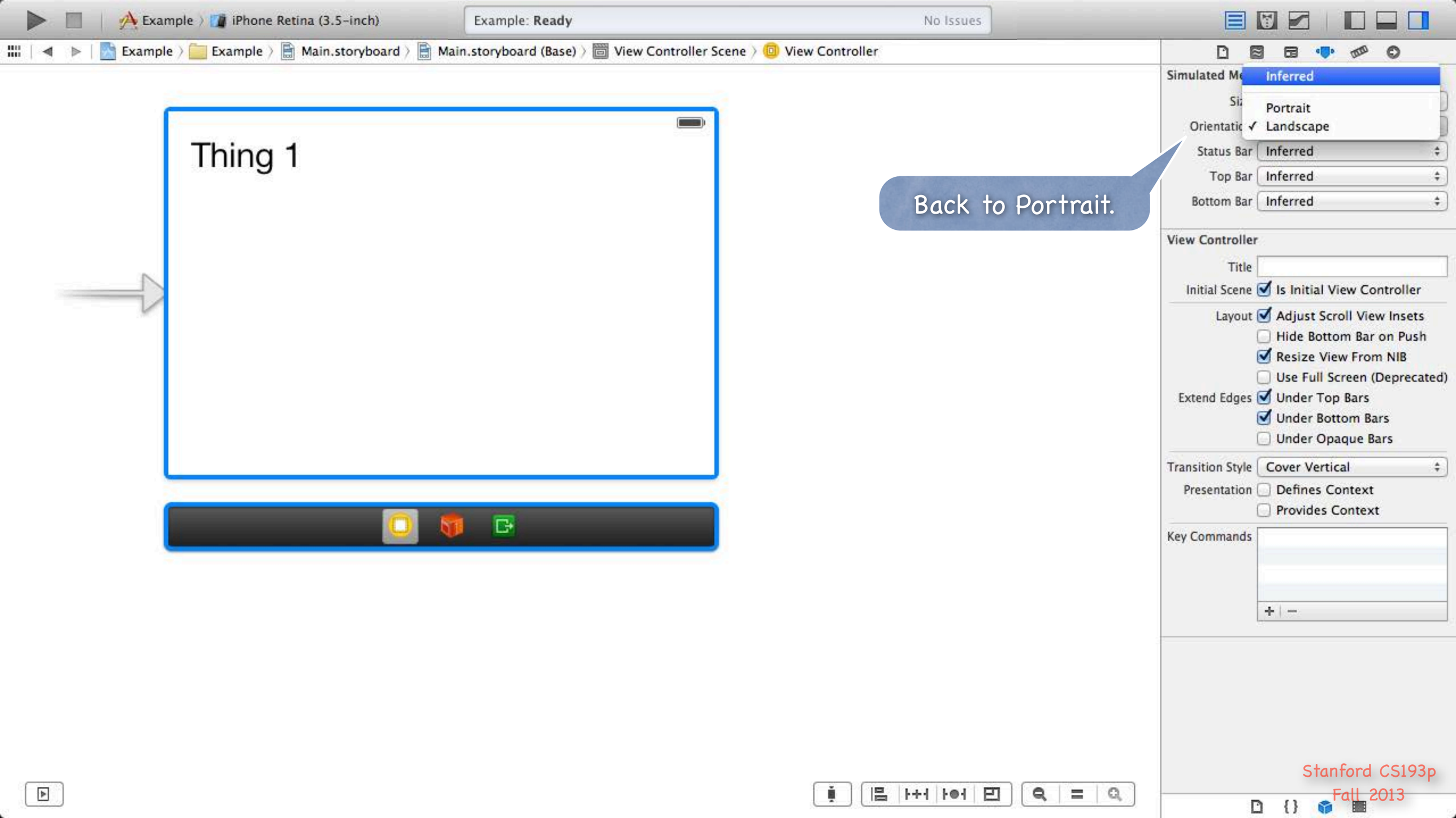

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Thing 1
Thing 2
Whew, Thing 2 is indeed still there!
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

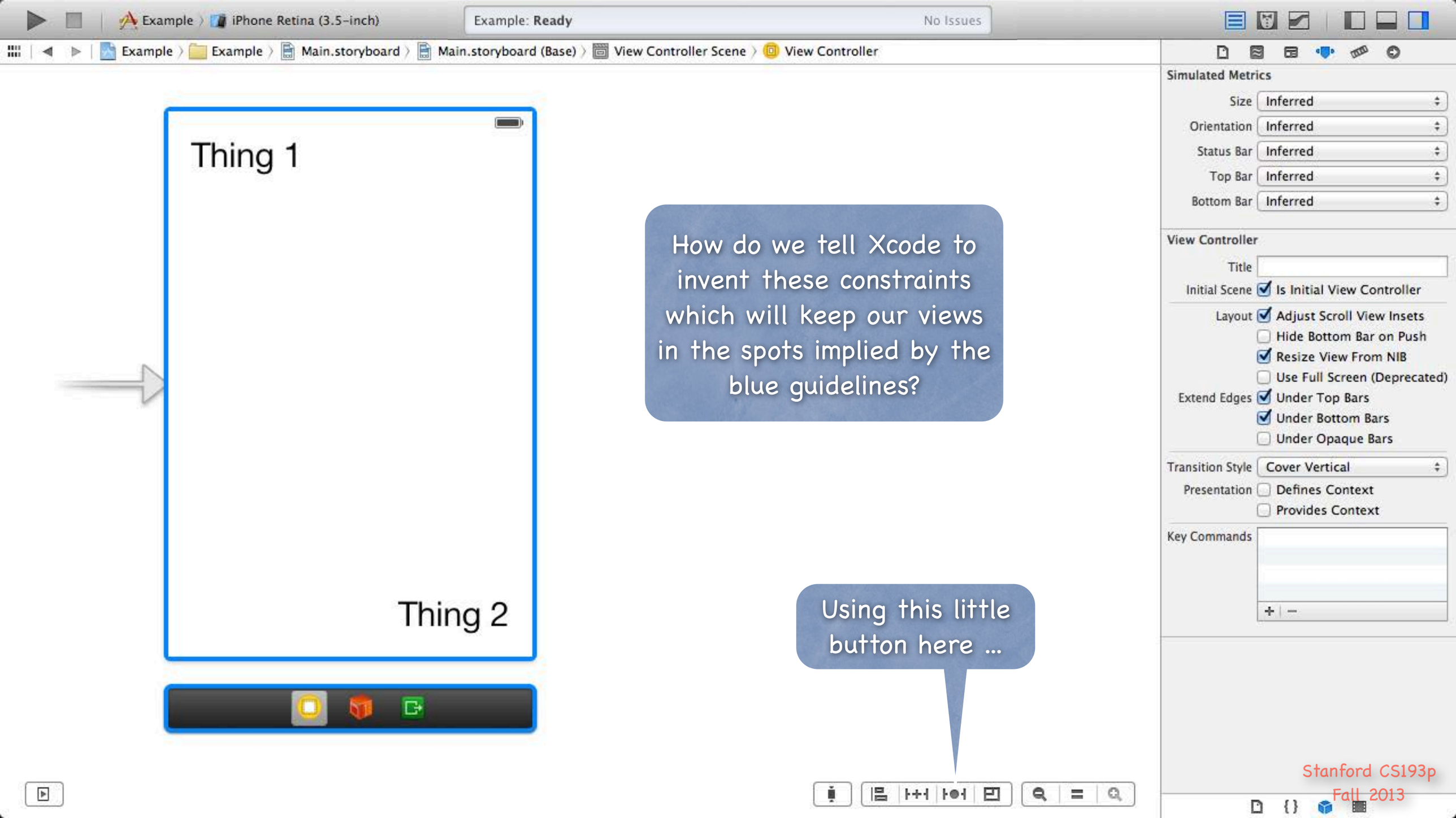

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View
Thing 1
Thing 2
Let's say we want Thing 1 and Thing 2 to stick to their nearby corner (i.e. to stick to that corner no matter where the corner moves to).
We can communicate that to Xcode by dragging to that corner and letting the blue guidelines appear.
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
White Color
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
X
0
Y
1
Width
1
Height
Stanford CS193p
Fall 2013

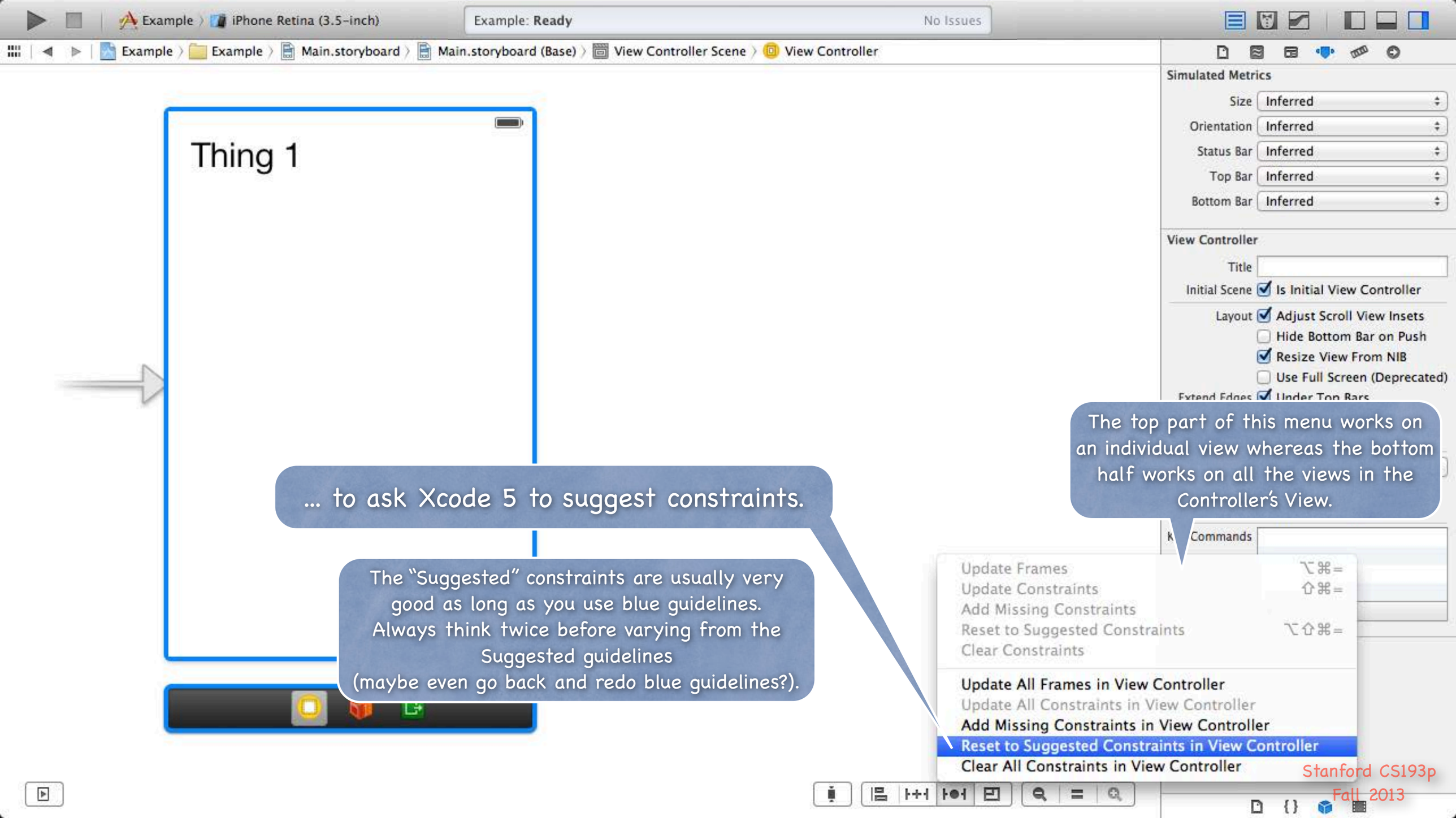

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
View
Thing 1
Thing 2
Ditto for Thing 2.
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
White Color
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
0
X
Y
1
1
Width
Height
Stanford CS193p
Fall 2013

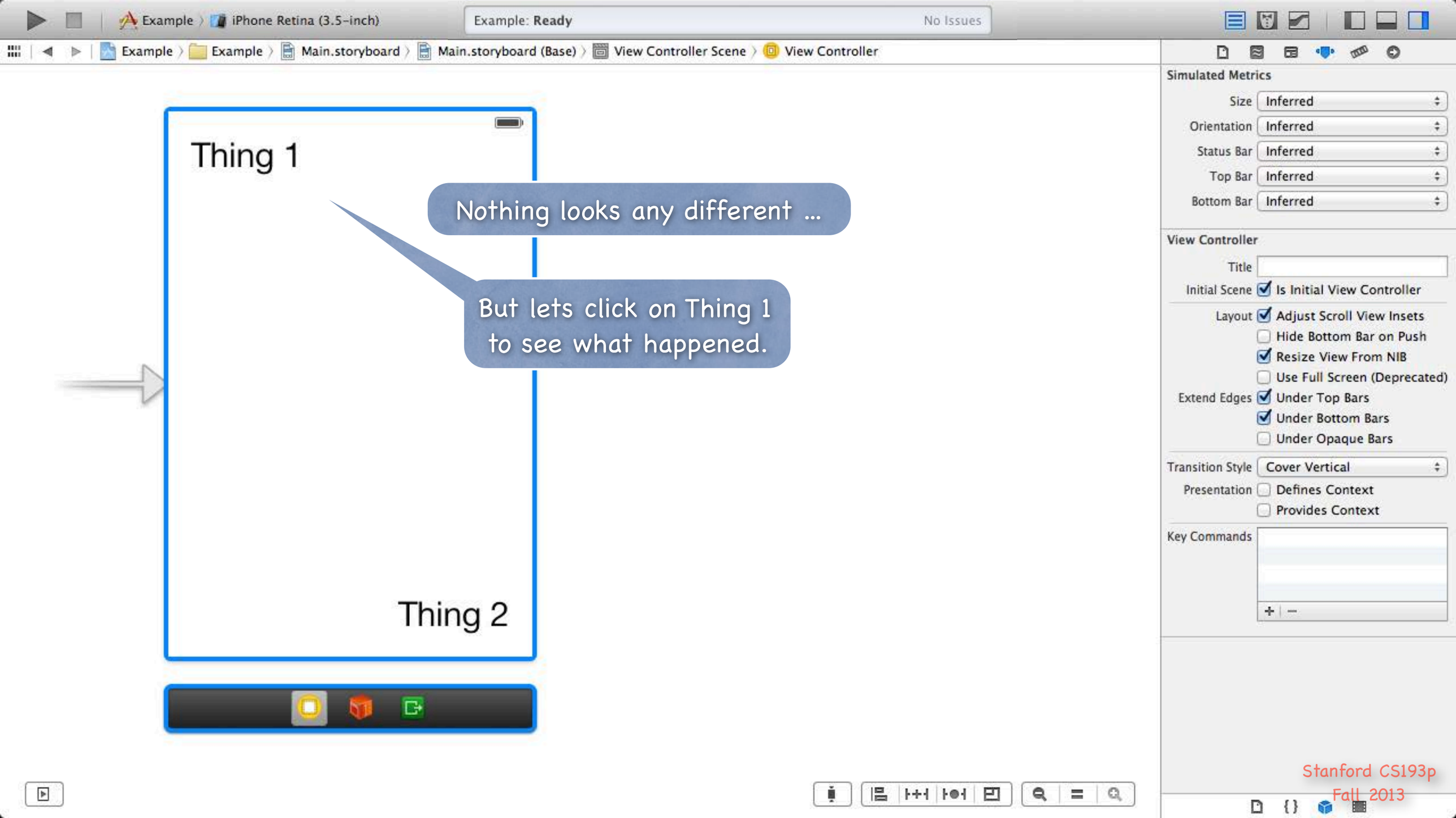

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Thing 2
Now let's try Landscape again.
Simulated Metrics
Size
Inferred
Orientation
Inferred
Portrait
Landscape
Status Bar
Top Bar
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

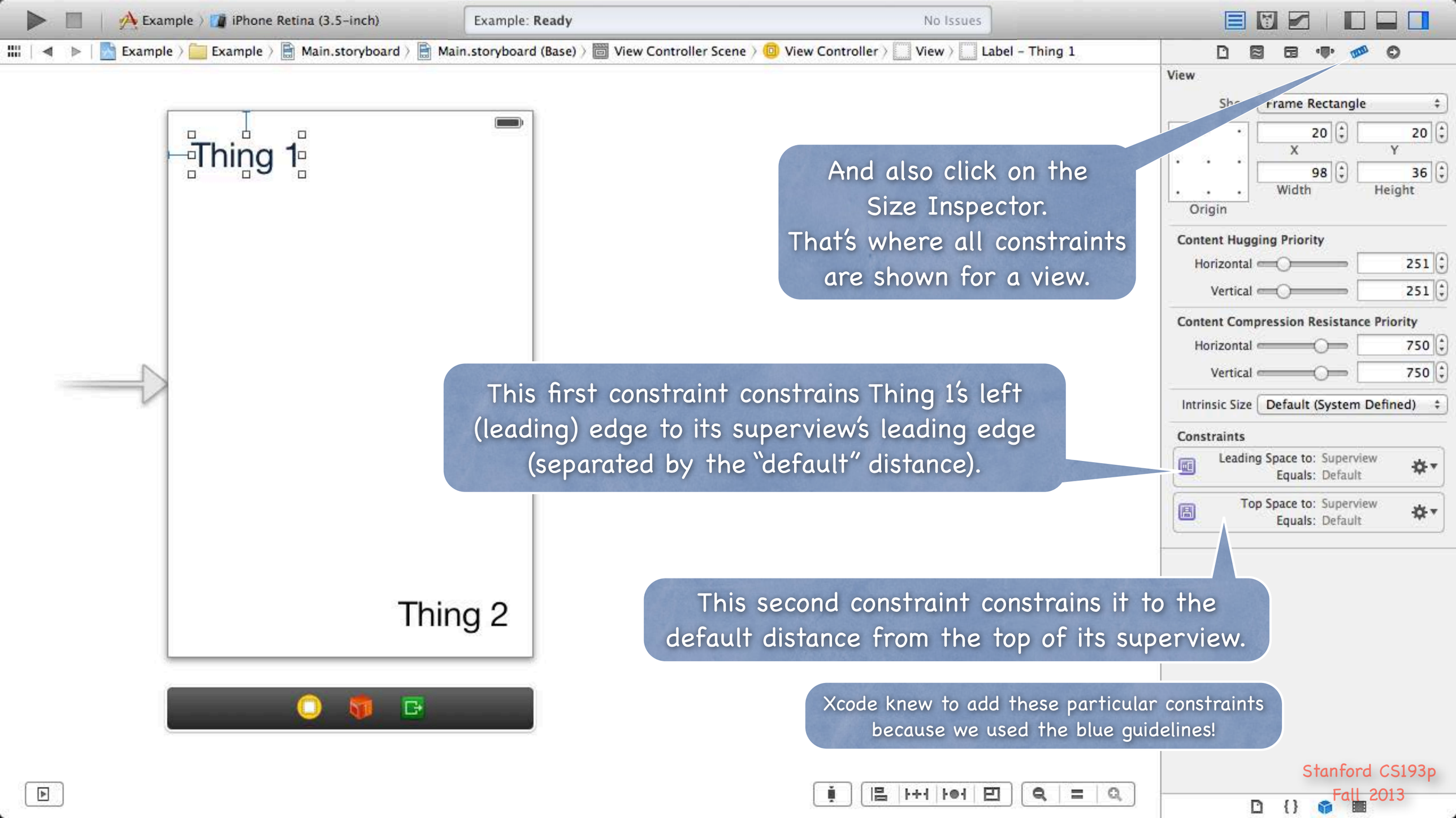

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Simulated Metrics
Size Inferred
Orientation Landscape
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands
Still doesn't work because the blue guidelines are not enough.
We have to tell iOS that we want the blue guidelines to be used
to create some "constraints" on our layout.
Stanford CS193p
Fall 2013

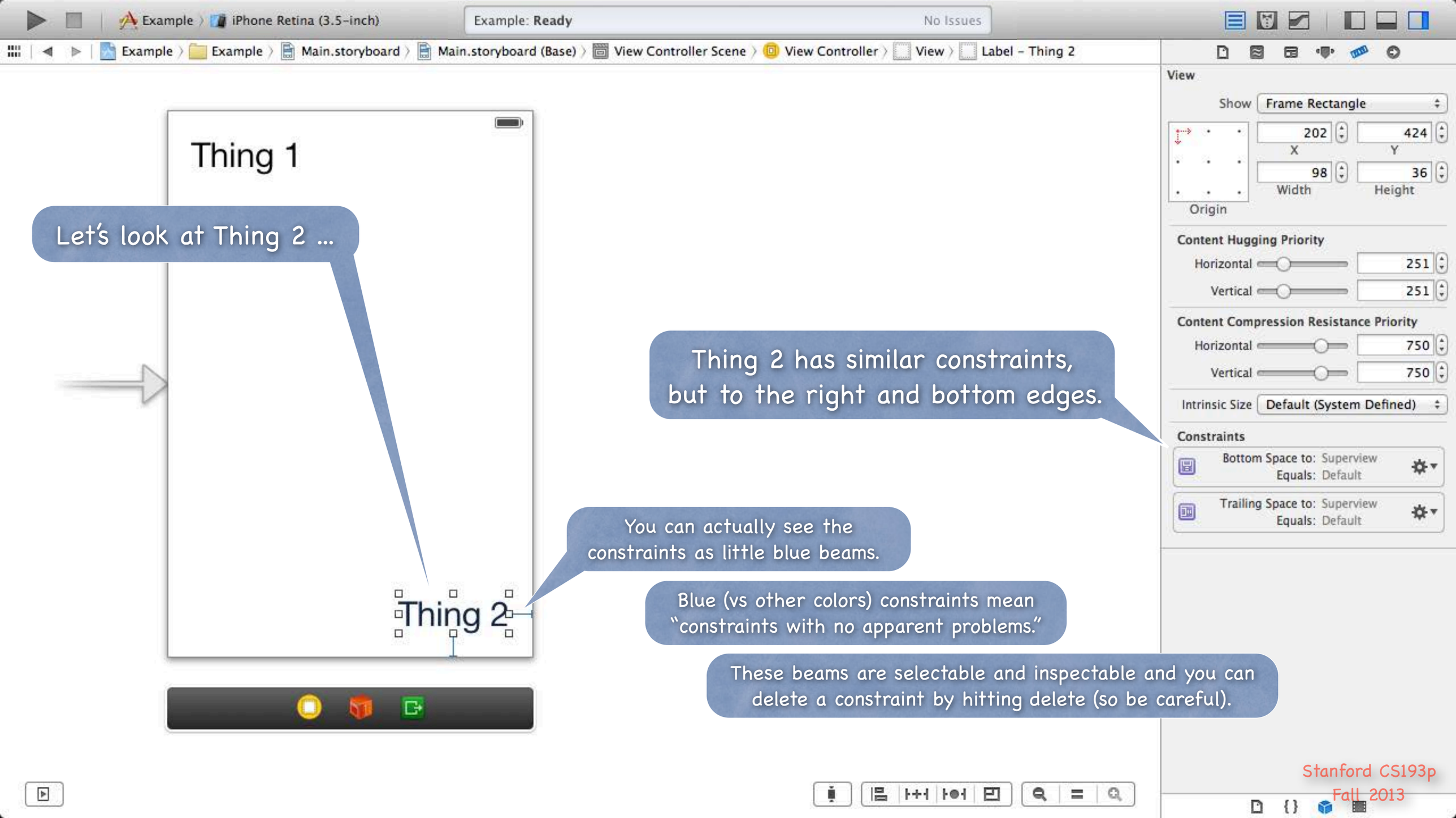

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Thing 1
Back to Portrait.
Simulated Me
Inferred
Portrait
Landscape
Orientatic
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

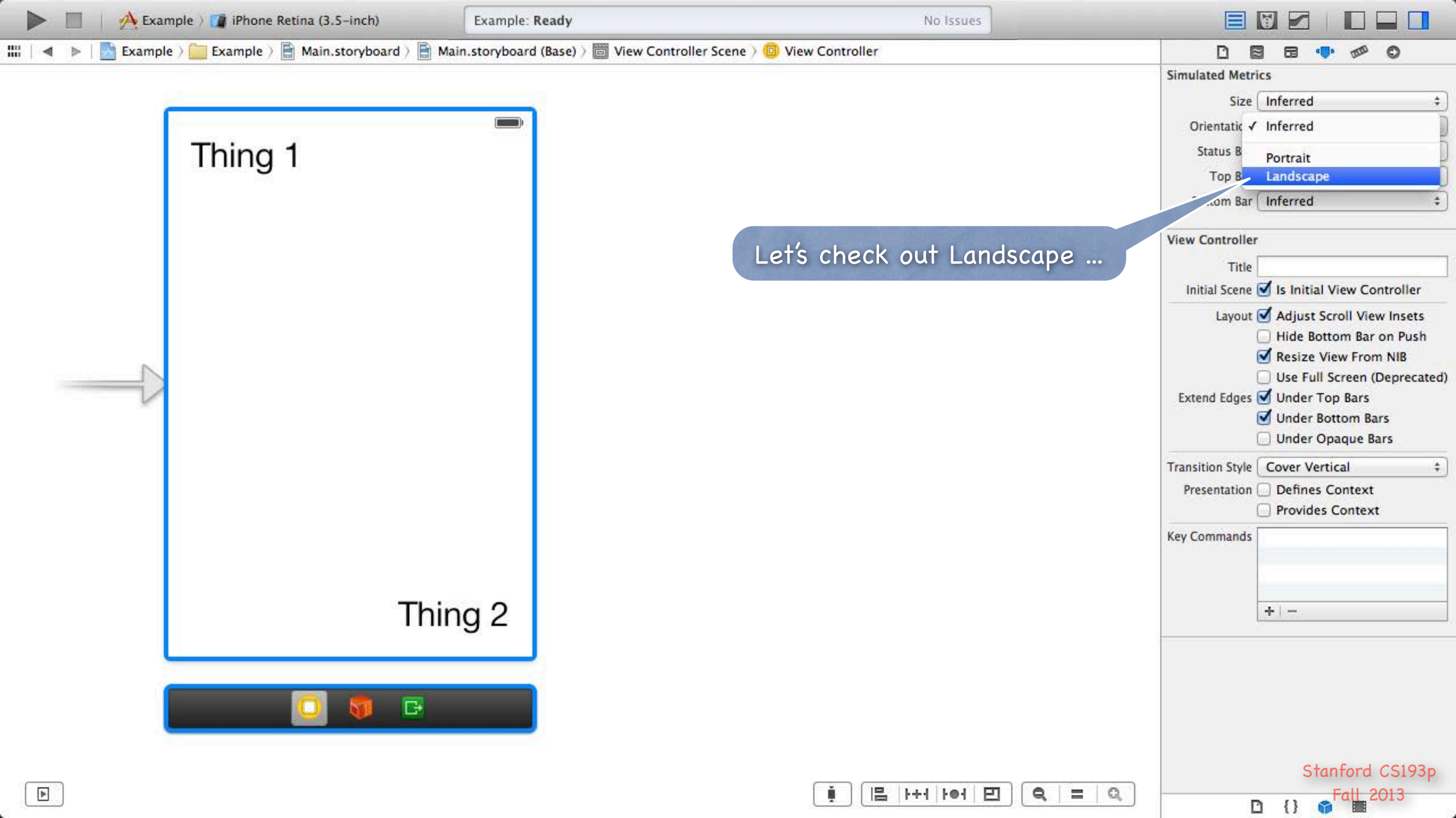

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Thing 2
How do we tell Xcode to invent these constraints which will keep our views in the spots implied by the blue guidelines?
Using this little button here ...
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

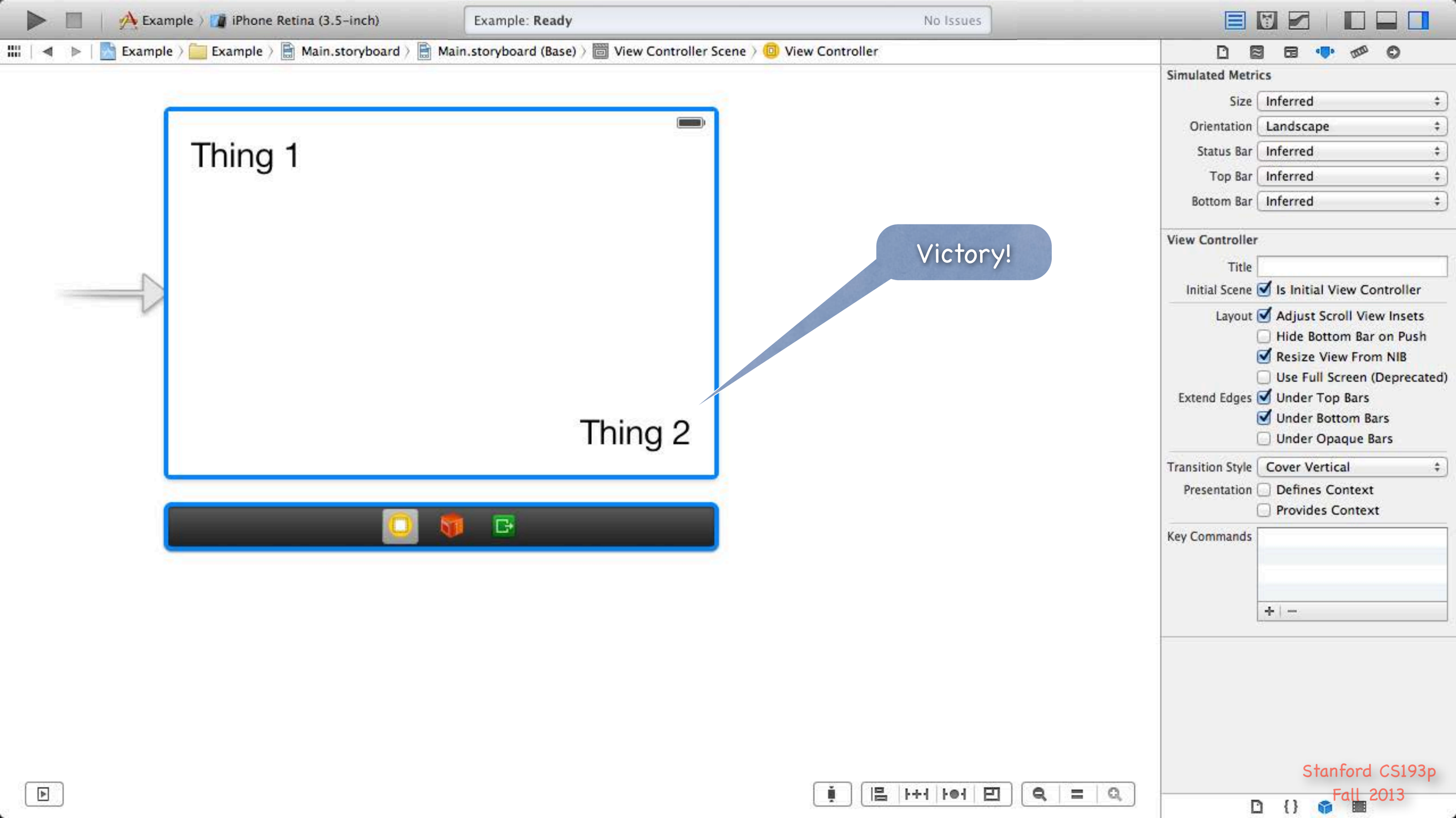

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Under Top Bars
... to ask Xcode 5 to suggest constraints.
The top part of this menu works on an individual view whereas the bottom half works on all the views in the Controller's View.
The "Suggested" constraints are usually very good as long as you use blue guidelines. Always think twice before varying from the Suggested guidelines (maybe even go back and redo blue guidelines?).
Update Frames
Update Constraints
Add Missing Constraints
Reset to Suggested Constraints
Clear Constraints
Update All Frames in View Controller
Update All Constraints in View Controller
Add Missing Constraints in View Controller
Reset to Suggested Constraints in View Controller
Clear All Constraints in View Controller
Stanford CS193p
Fall 2013

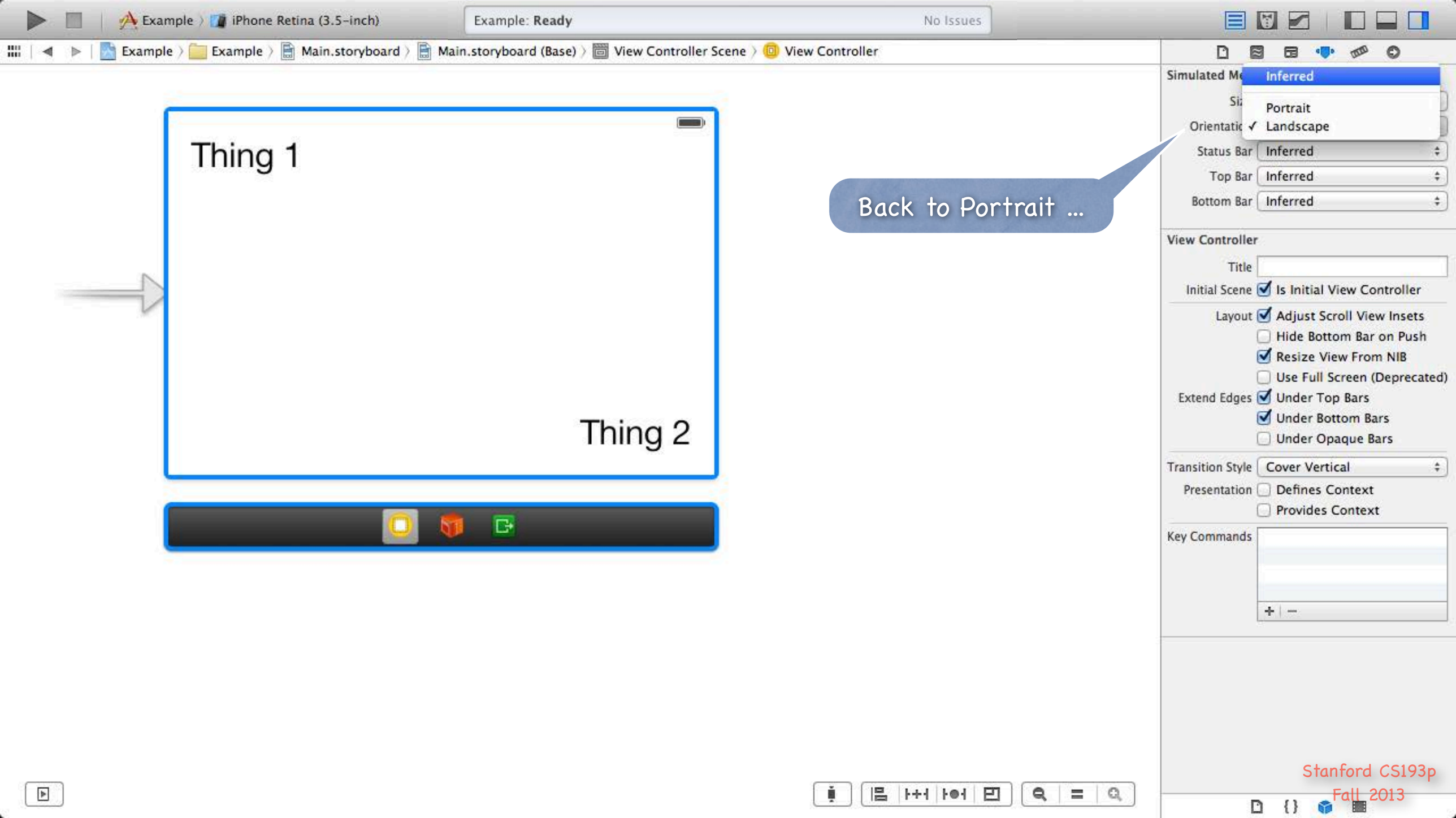

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Thing 2
Nothing looks any different ...
But lets click on Thing 1 to see what happened.
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

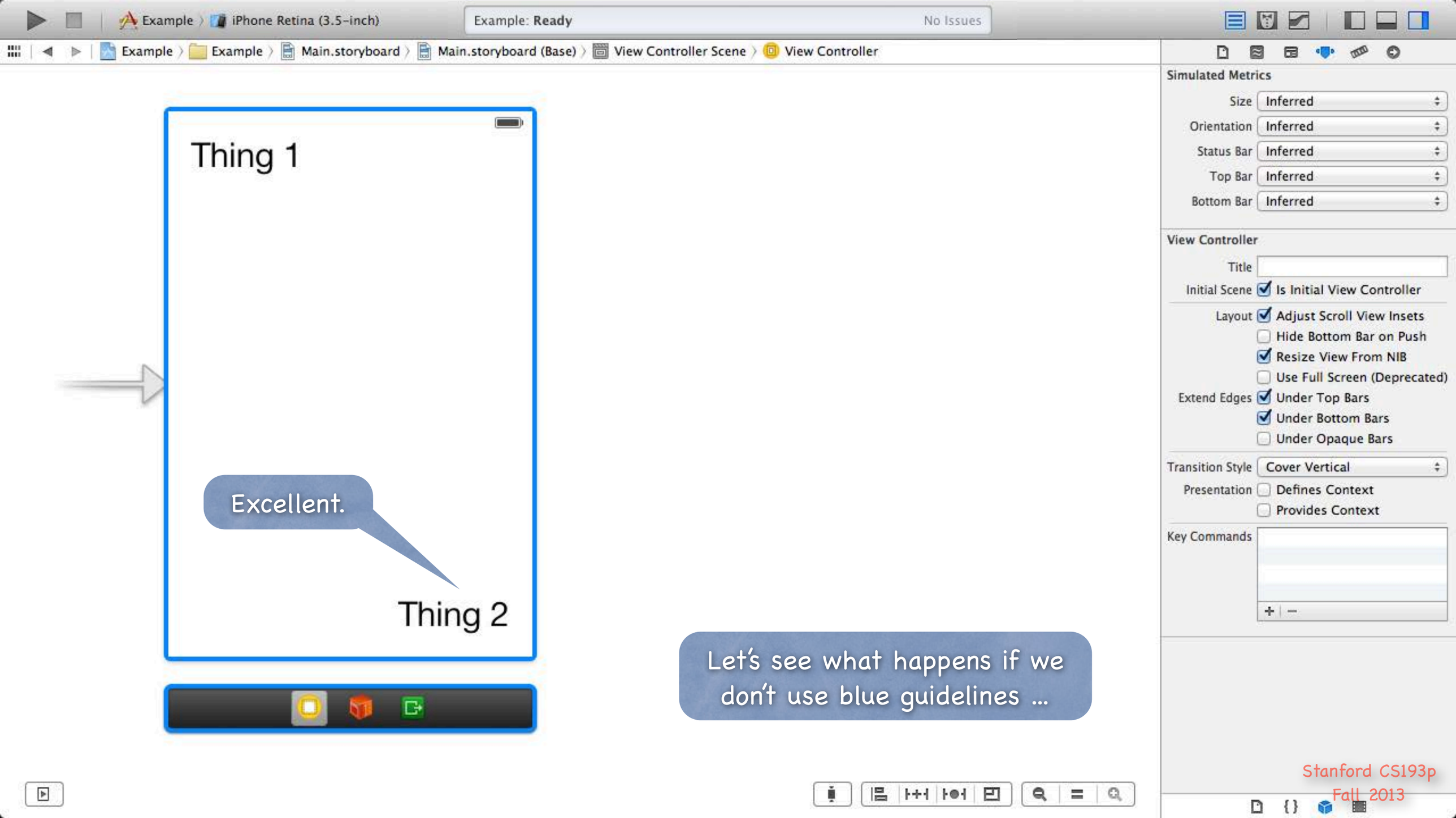

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Thing 1
Thing 1
Thing 2
View
Frame Rectangle
20
X
20
Y
98
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Leading Space to: Superview
Equals: Default
Top Space to: Superview
Equals: Default
And also click on the Size Inspector. That's where all constraints are shown for a view.
This first constraint constrains Thing 1's left (leading) edge to its superview's leading edge (separated by the "default" distance).
This second constraint constrains it to the default distance from the top of its superview.
Xcode knew to add these particular constraints because we used the blue guidelines!
Stanford CS193p
Fall 2013

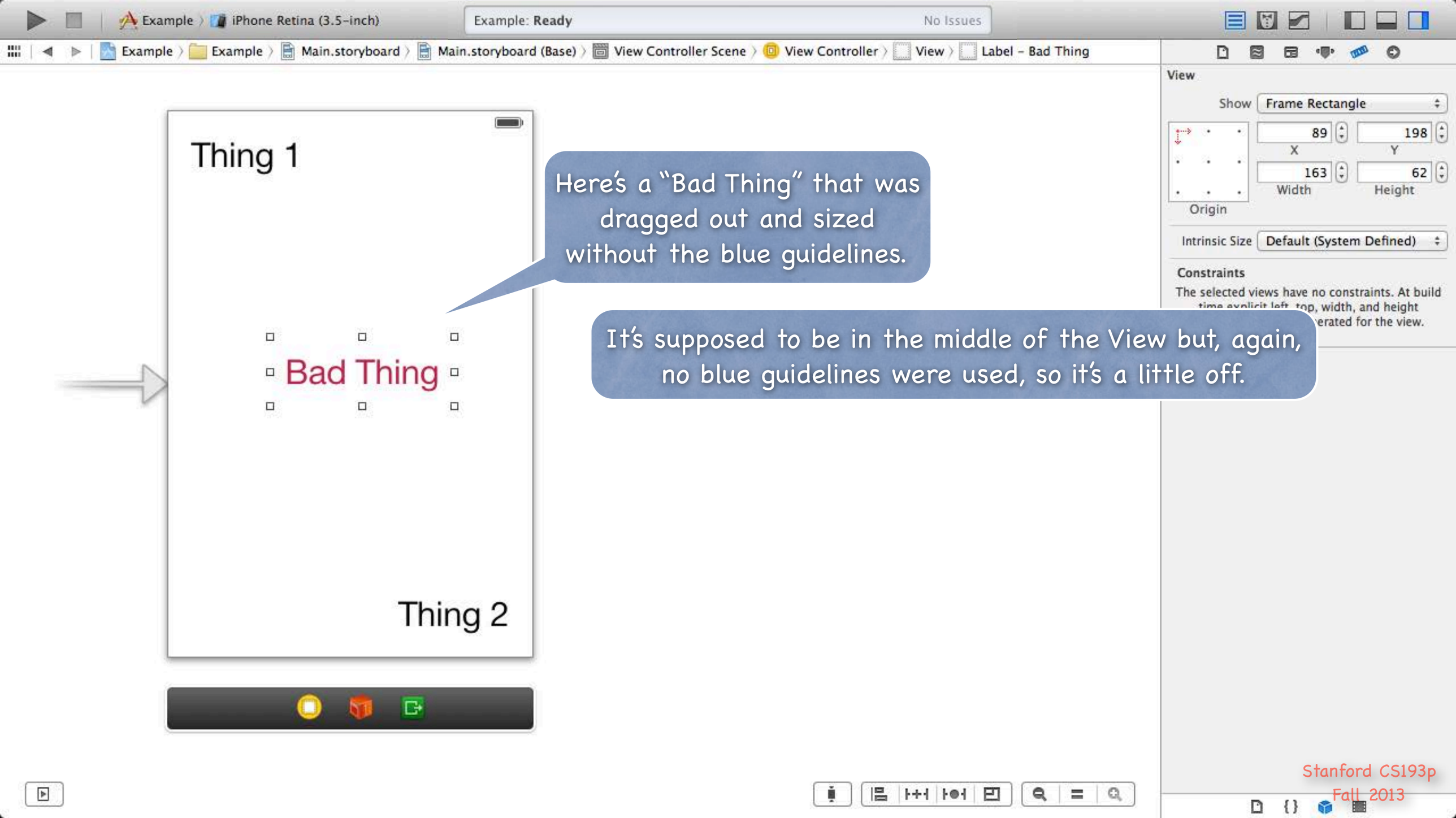

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Thing 2
Thing 1
Let's look at Thing 2 ...
Thing 2
Thing 2 has similar constraints, but to the right and bottom edges.
You can actually see the constraints as little blue beams.
Blue (vs other colors) constraints mean "constraints with no apparent problems."
These beams are selectable and inspectable and you can delete a constraint by hitting delete (so be careful).
View
Show Frame Rectangle
202 X
424 Y
98 Width
36 Height
Origin
Content Hugging Priority
Horizontal 251
Vertical 251
Content Compression Resistance Priority
Horizontal 750
Vertical 750
Intrinsic Size Default (System Defined)
Constraints
Bottom Space to: Superview
Equals: Default
Trailing Space to: Superview
Equals: Default
Stanford CS193p
Fall 2013

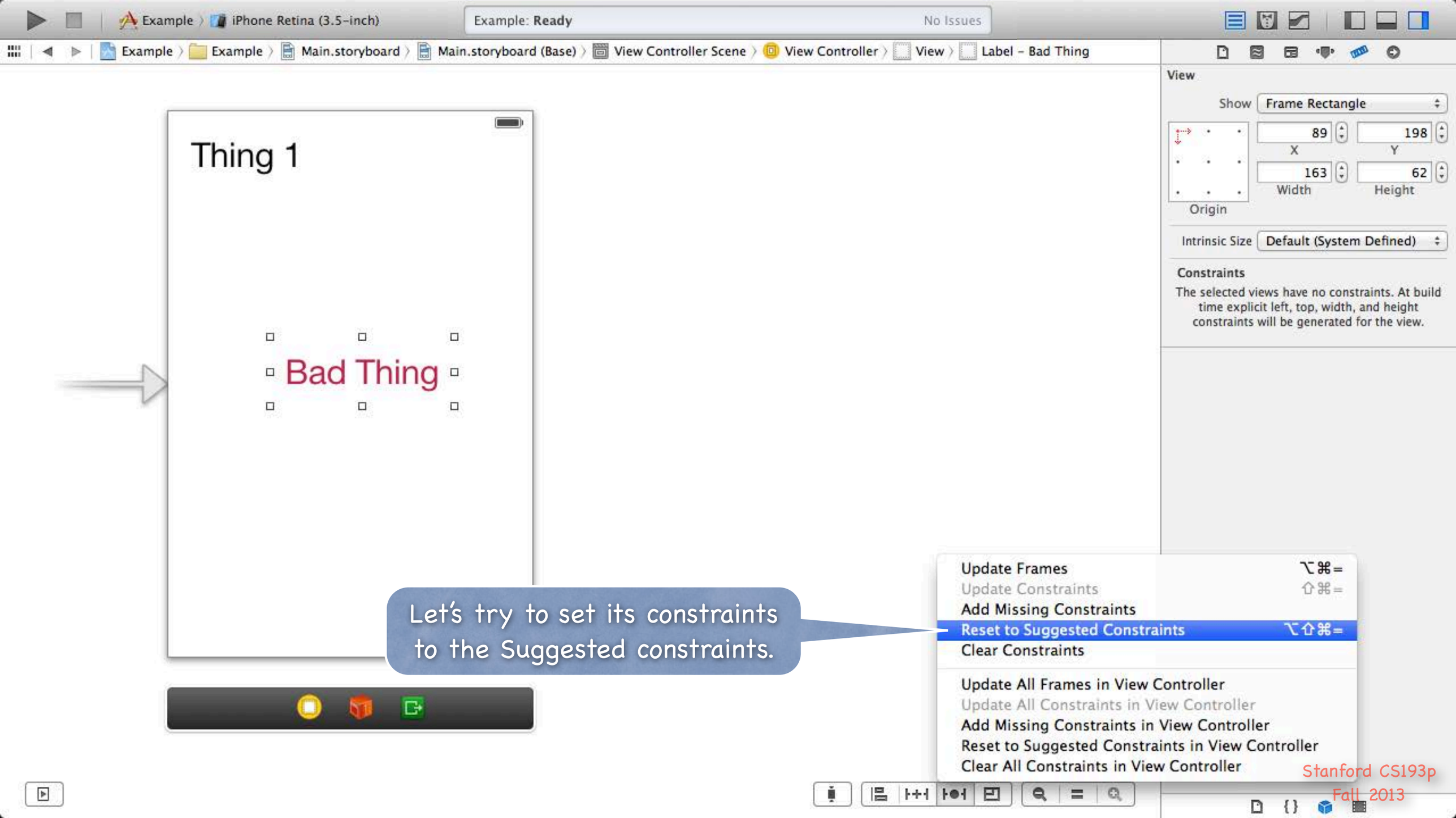

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Thing 2
Let's check out Landscape ...
Simulated Metrics
Size
Inferred
Orientatic
✓ Inferred
Portrait
Landscape
Status B
Top B
om Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

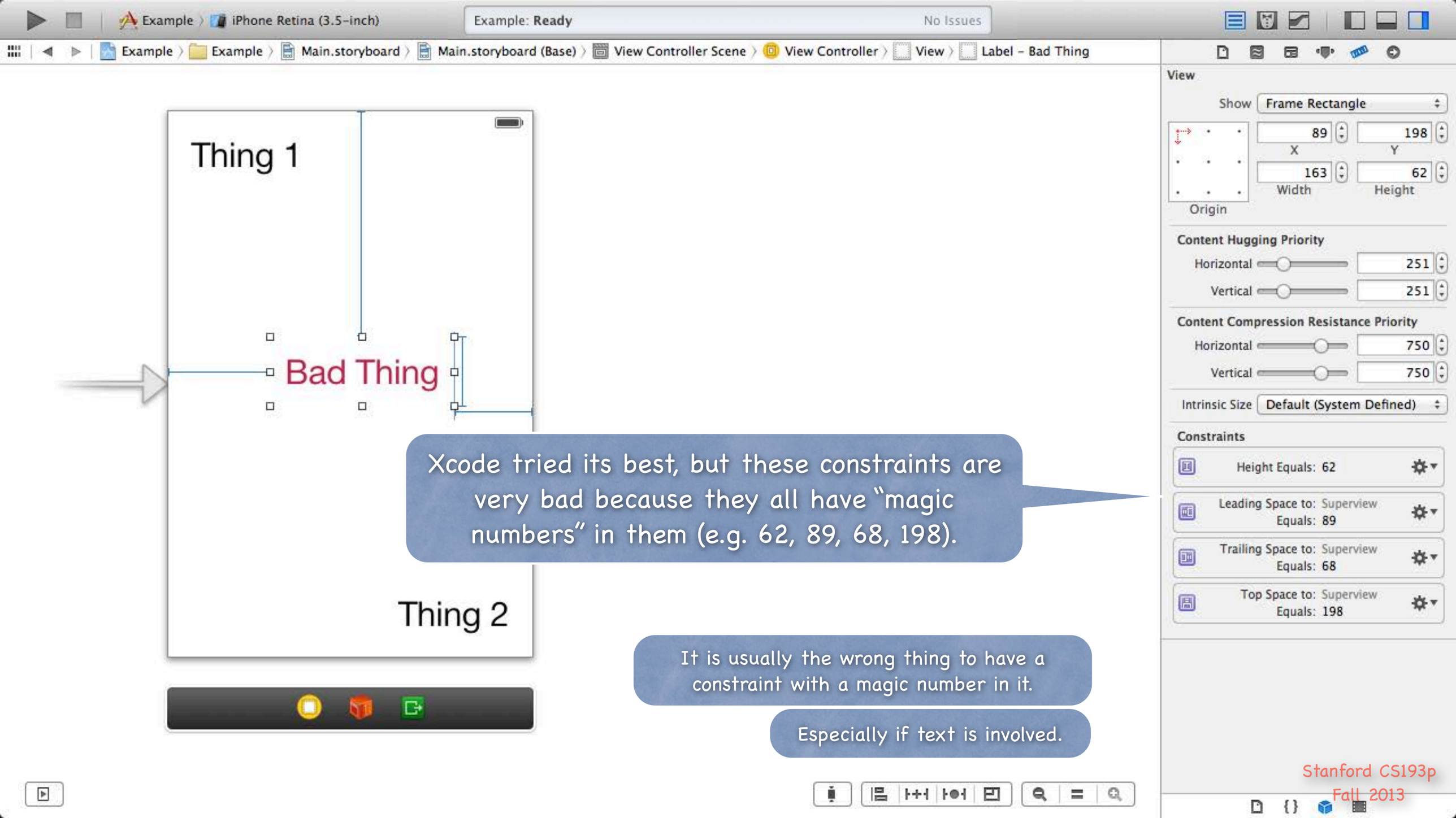

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Thing 1
Thing 2
Victory!
Simulated Metrics
Size
Inferred
Orientation
Landscape
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

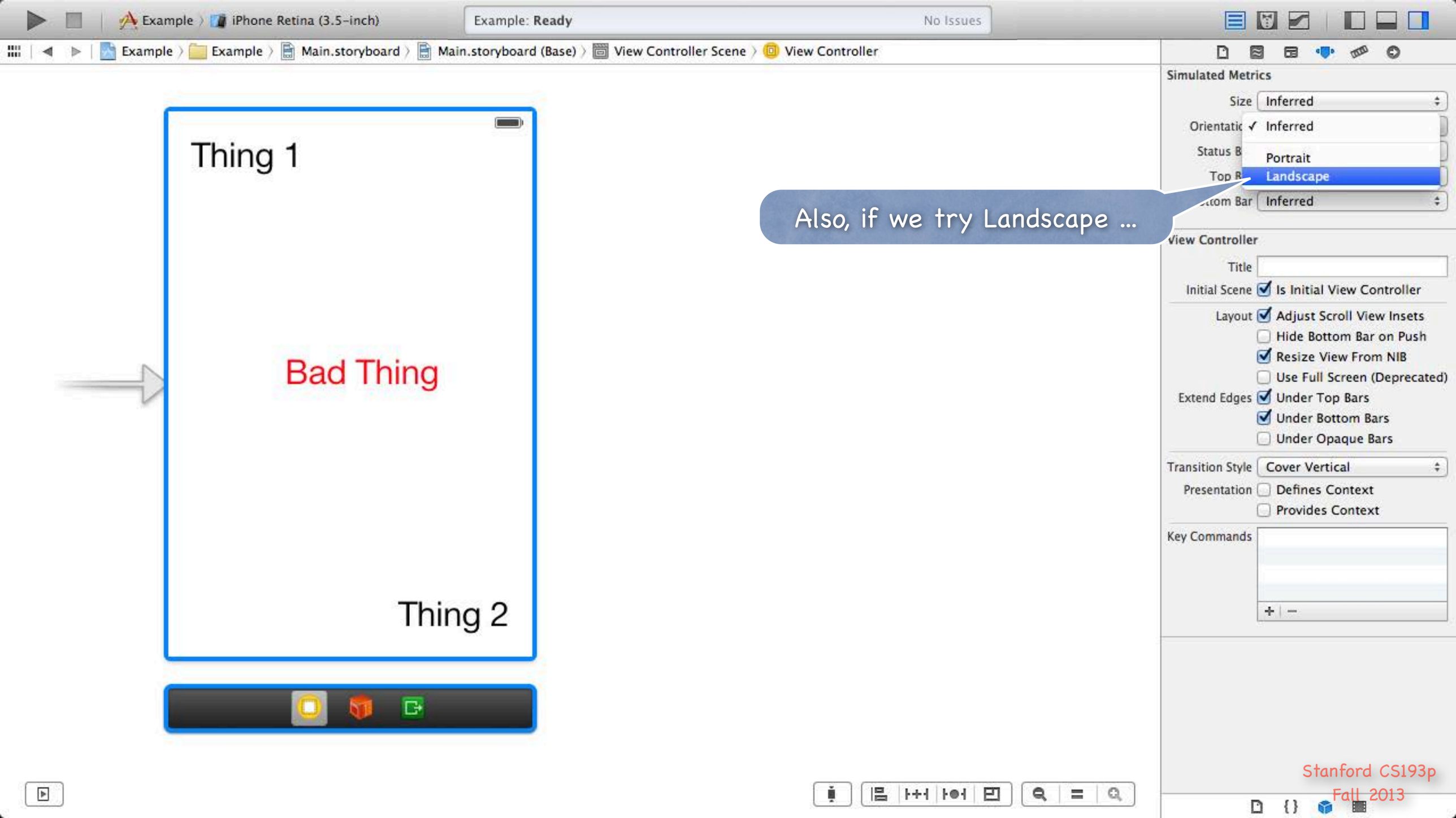

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Thing 2
Back to Portrait …
Simulated Me
Inferred
Portrait
Landscape
Orientatic
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

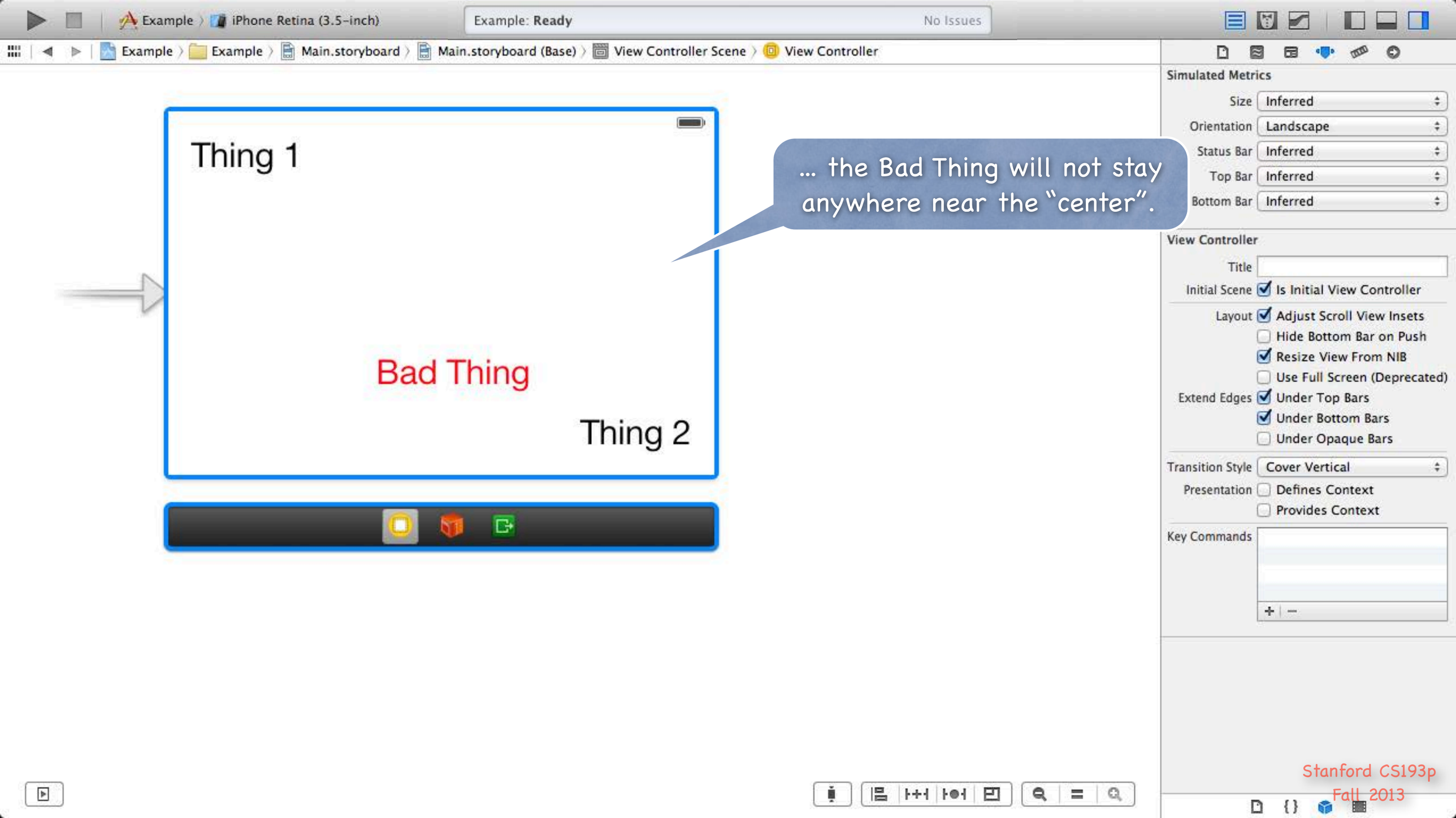

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Thing 1
Excellent.
Thing 2
Let's see what happens if we don't use blue guidelines ...
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands

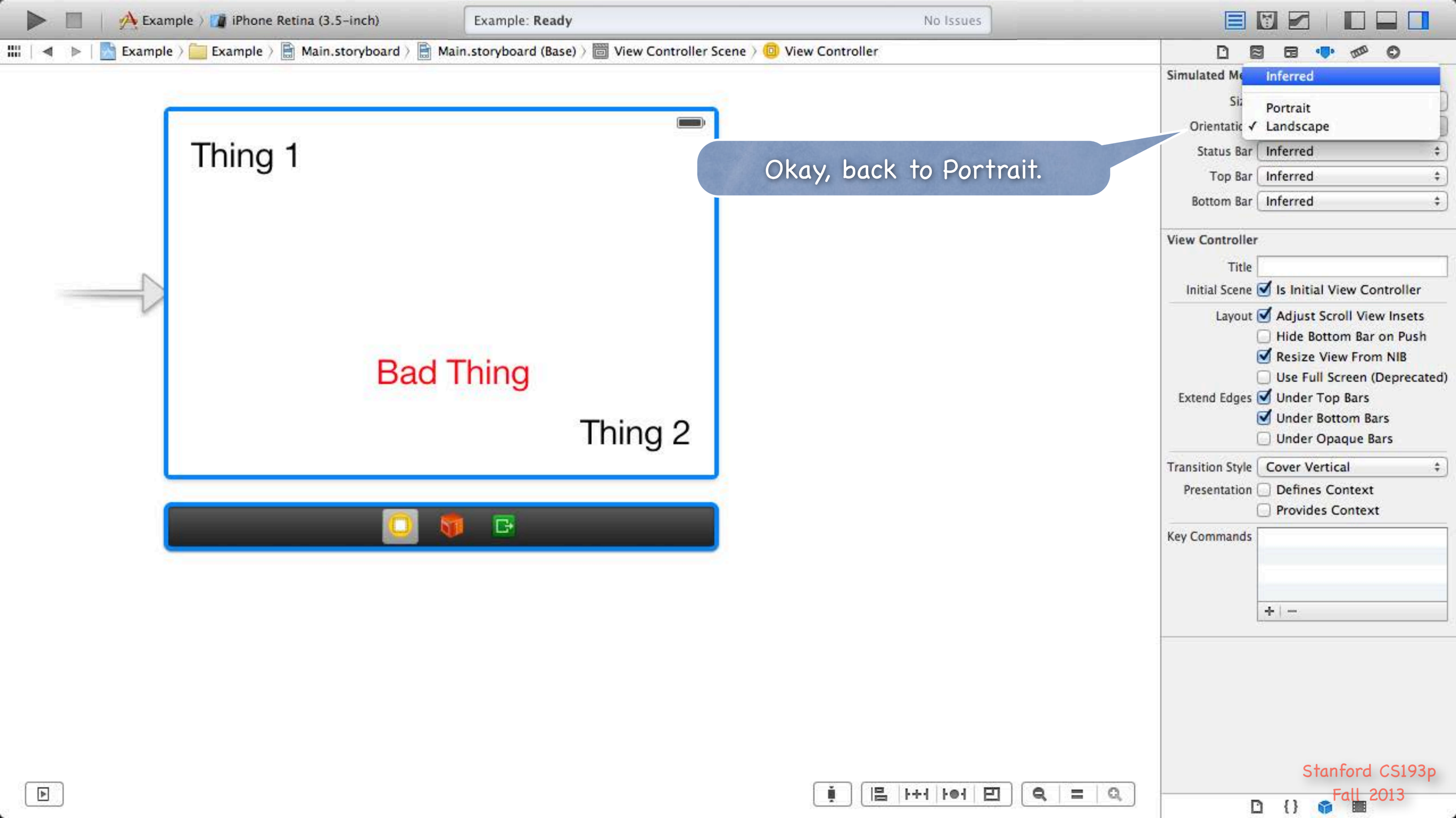

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Thing 1
Bad Thing
Thing 2
Here's a "Bad Thing" that was dragged out and sized without the blue guidelines.
It's supposed to be in the middle of the View but, again, no blue guidelines were used, so it's a little off.
View
Show
Frame Rectangle
89
198
X
Y
163
62
Width
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build
erated for the view.
Stanford CS193p
Fall 2013

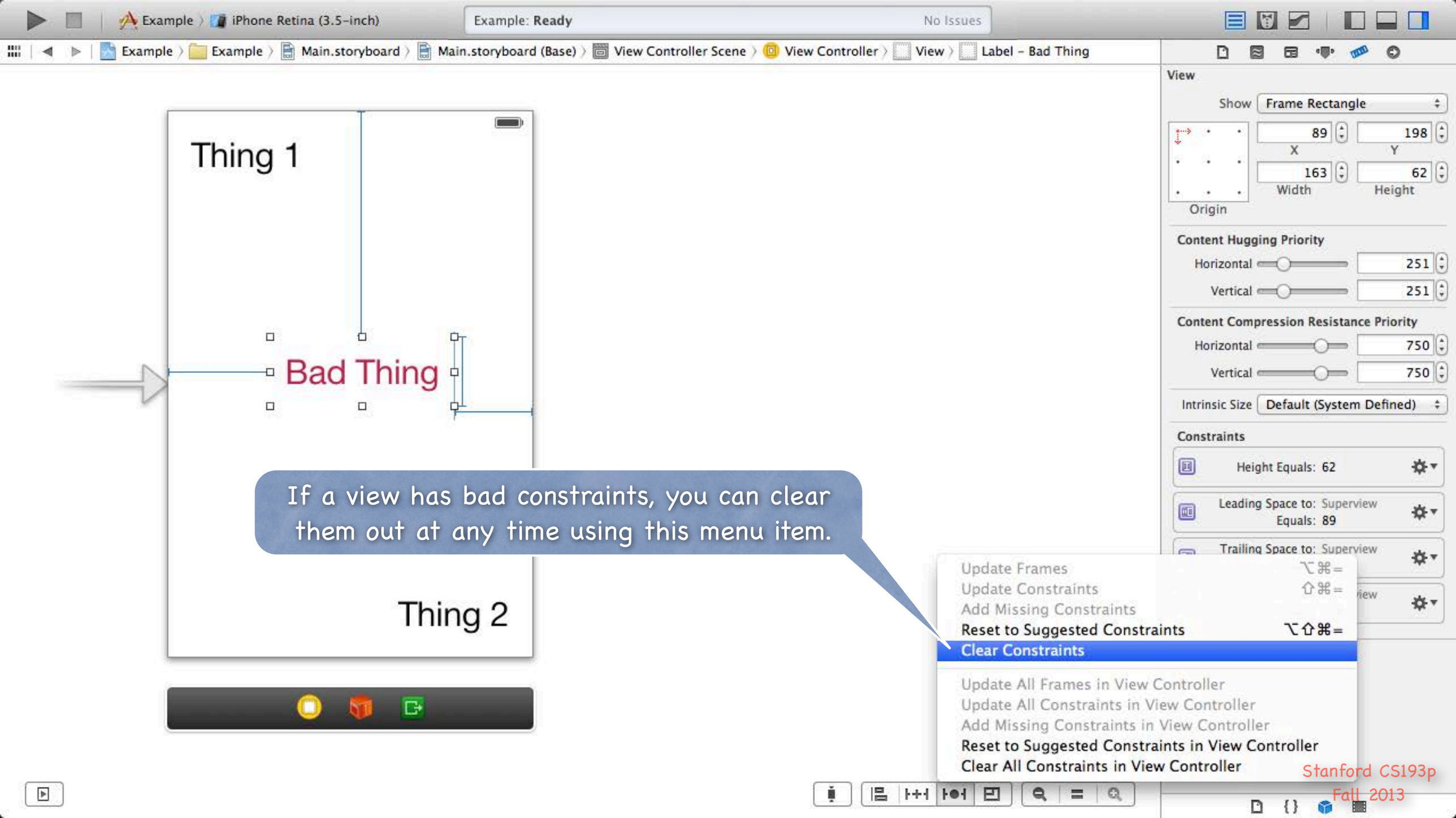

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Thing 1
Bad Thing
View
Show
Frame Rectangle
89
198
X
Y
163
62
Width
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
Let's try to set its constraints to the Suggested constraints.
Update Frames ⌥⌘=
Update Constraints ⇧⌘=
Add Missing Constraints
Reset to Suggested Constraints ⌥⇧⌘=
Clear Constraints
Update All Frames in View Controller
Update All Constraints in View Controller
Add Missing Constraints in View Controller
Reset to Suggested Constraints in View Controller
Clear All Constraints in View Controller
Stanford CS193p
Fall 2013

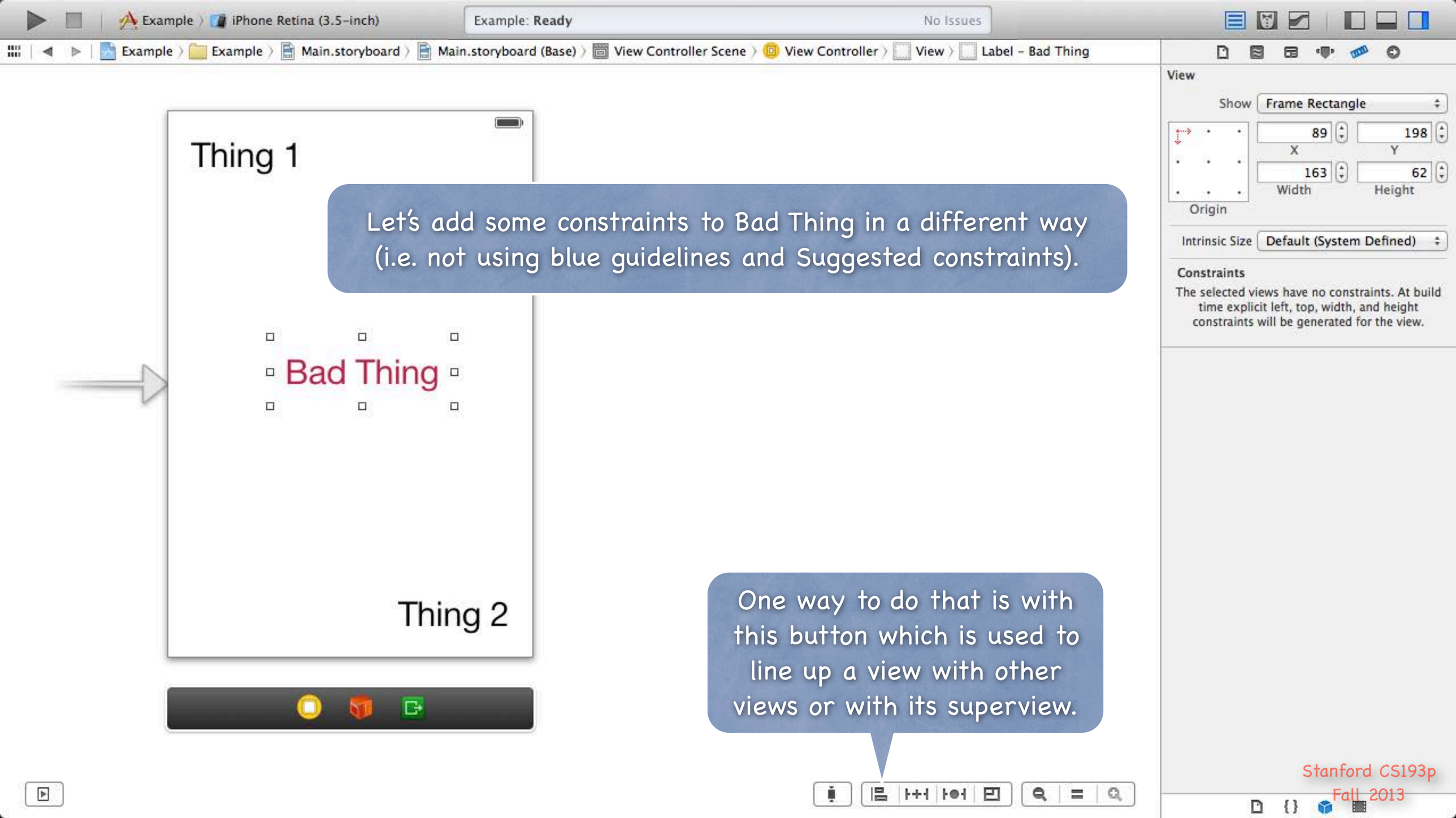

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Thing 1
Bad Thing
Thing 2
Xcode tried its best, but these constraints are very bad because they all have "magic numbers" in them (e.g. 62, 89, 68, 198).
It is usually the wrong thing to have a constraint with a magic number in it.
Especially if text is involved.
View
Show
Frame Rectangle
89
X
198
Y
163
Width
62
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Height Equals: 62
Leading Space to: Superview
Equals: 89
Trailing Space to: Superview
Equals: 68
Top Space to: Superview
Equals: 198
Stanford CS193p
Fall 2013

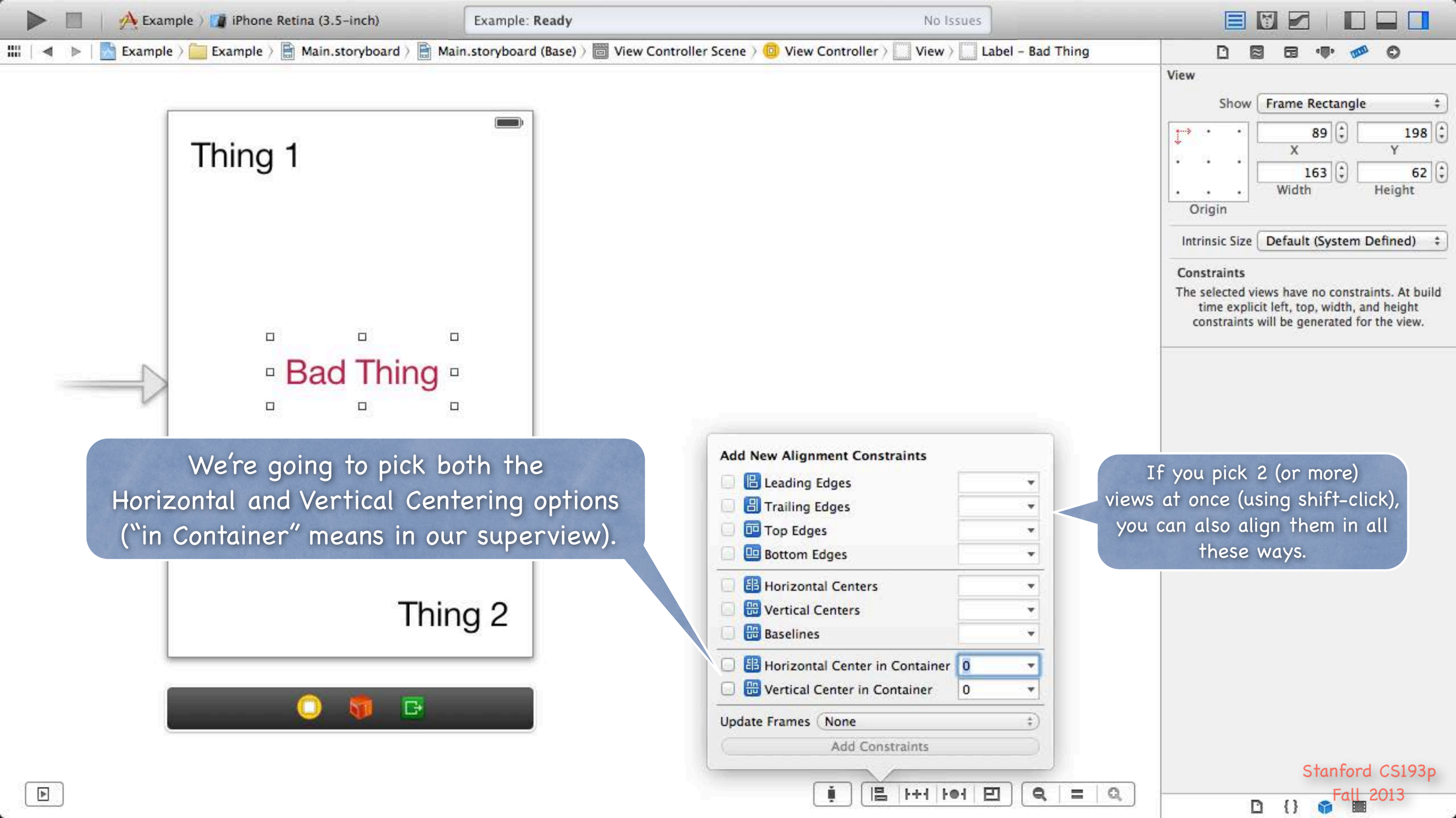

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Thing 1
Bad Thing
Thing 2
Also, if we try Landscape ...
Simulated Metrics
Size
Inferred
Orientation
Inferred
Portrait
Landscape
Status Bar
Top Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

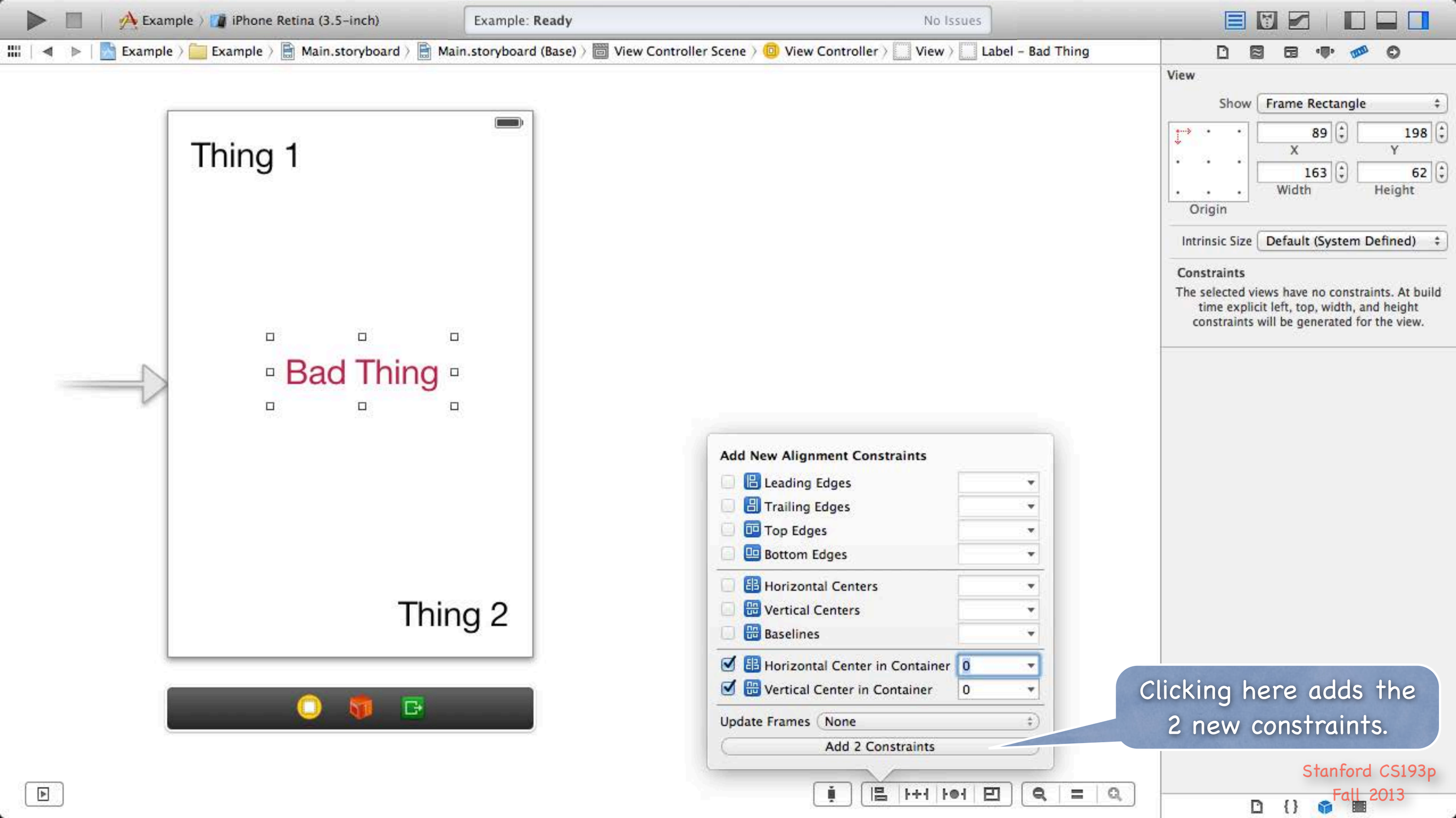

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
Thing 1
Bad Thing
Thing 2
… the Bad Thing will not stay anywhere near the "center".
Simulated Metrics
Size Inferred
Orientation Landscape
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands

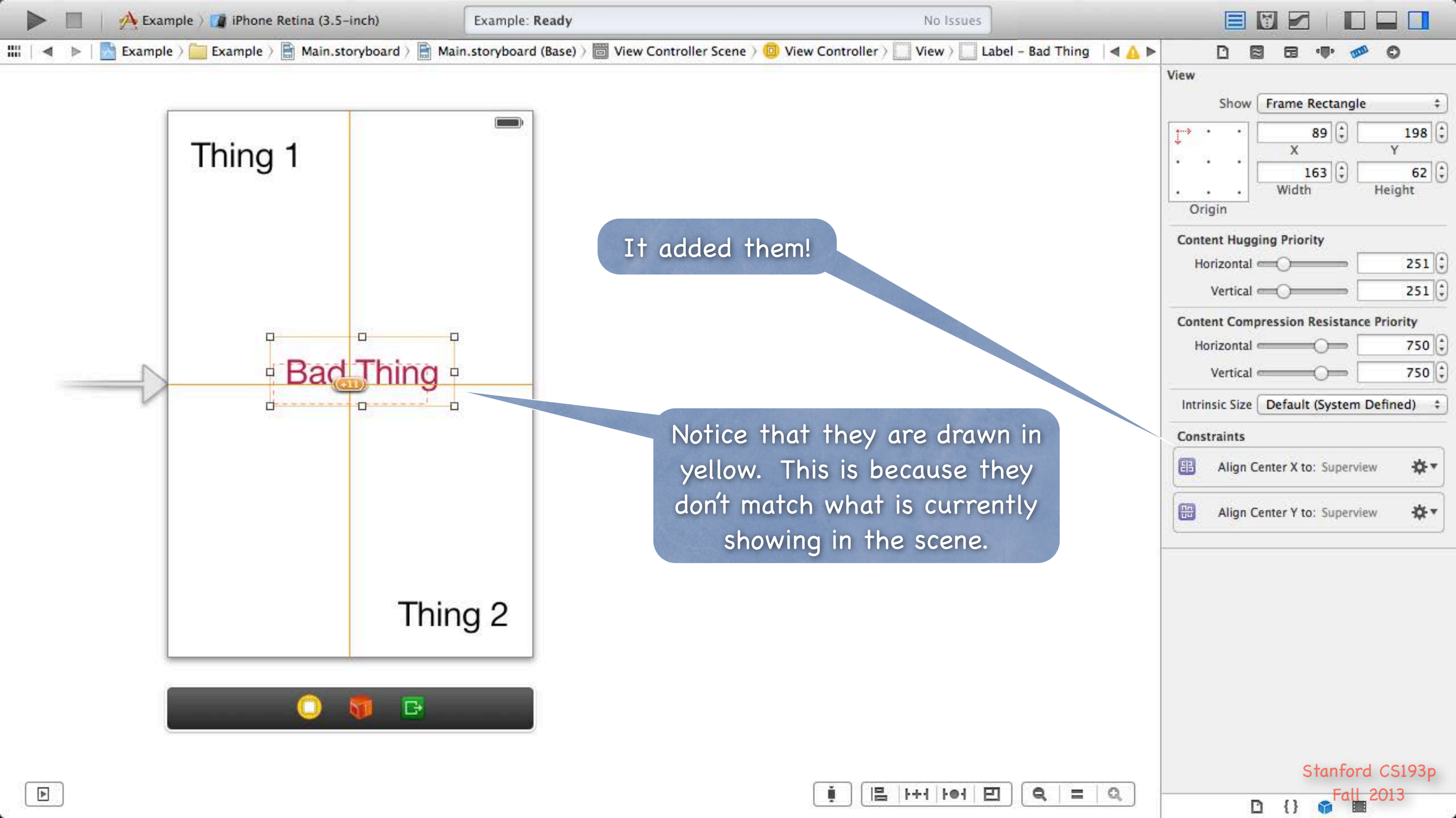

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
Thing 1
Bad Thing
Thing 2
Okay, back to Portrait.
Simulated Me
Inferred
Portrait
Landscape
Orientatic
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

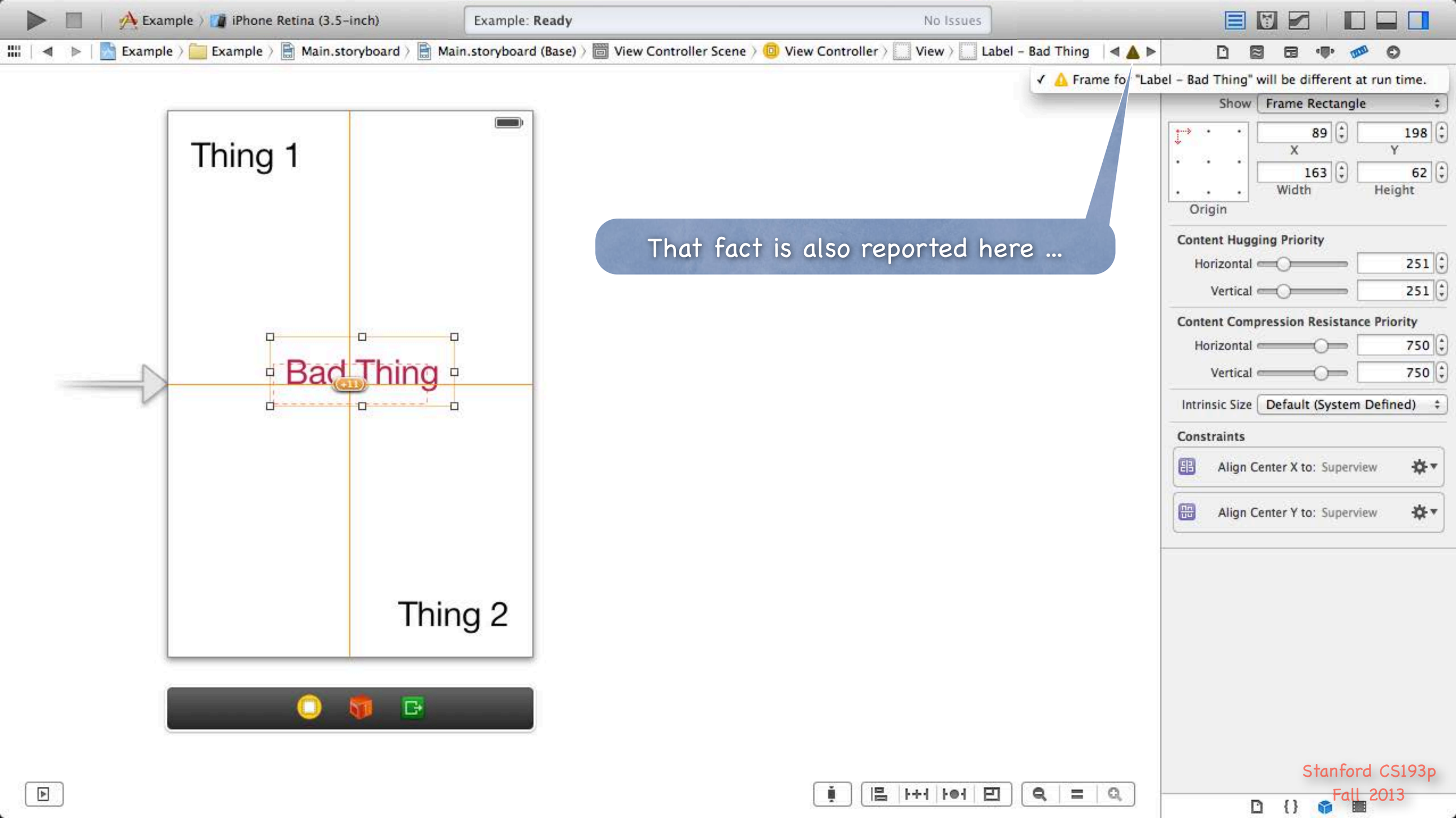

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Thing 1
Bad Thing
Thing 2
If a view has bad constraints, you can clear them out at any time using this menu item.
View
Show
Frame Rectangle
89
X
198
Y
163
Width
62
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Height Equals: 62
Leading Space to: Superview
Equals: 89
Update Frames
Update Constraints
Add Missing Constraints
Reset to Suggested Constraints
Clear Constraints
Update All Frames in View Controller
Update All Constraints in View Controller
Add Missing Constraints in View Controller
Reset to Suggested Constraints in View Controller
Clear All Constraints in View Controller
Stanford CS193p
Fall 2013

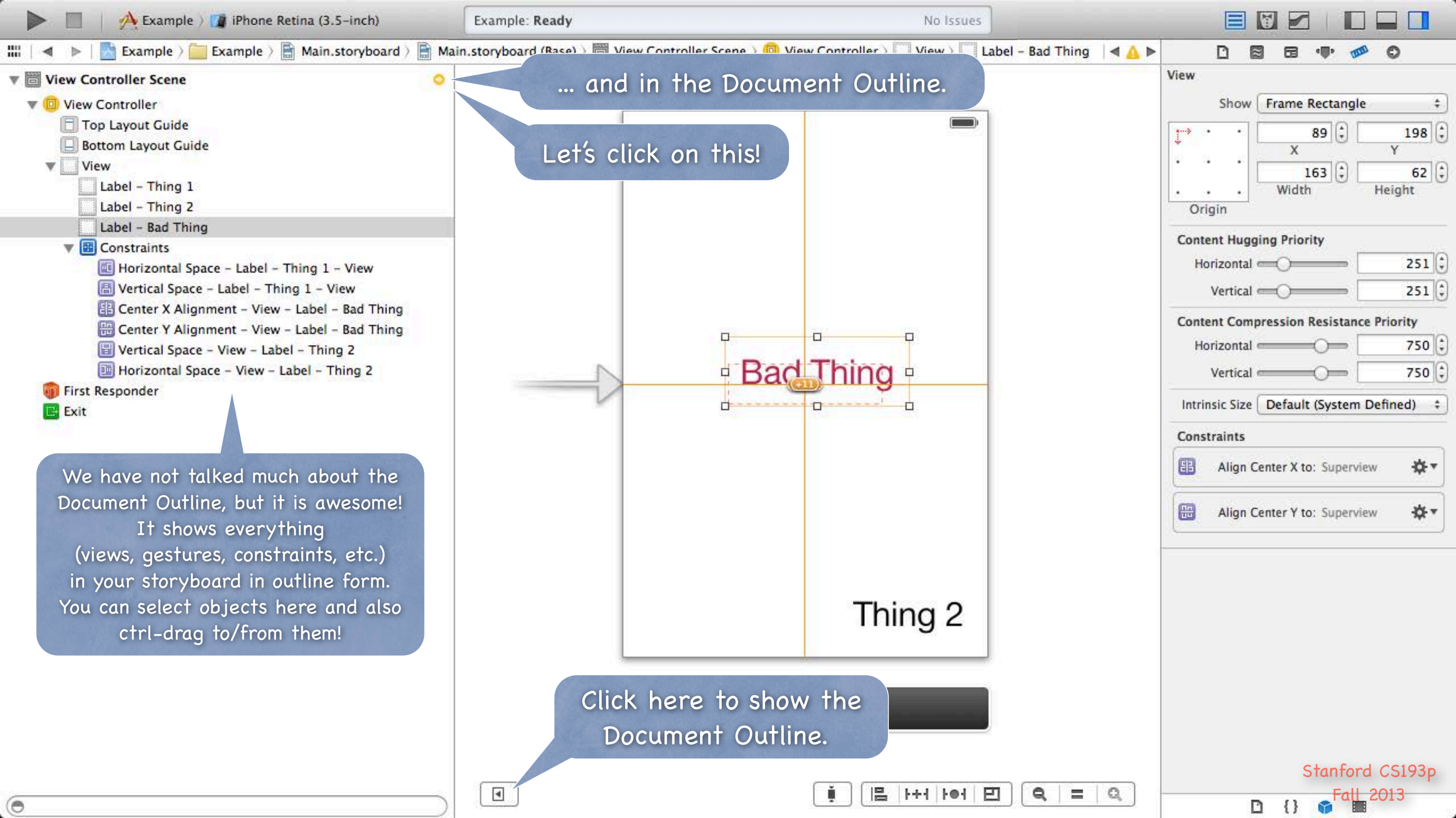

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Thing 1
Bad Thing
Thing 2
Let's add some constraints to Bad Thing in a different way (i.e. not using blue guidelines and Suggested constraints).
One way to do that is with this button which is used to line up a view with other views or with its superview.
View
Show
Frame Rectangle
89
198
X
Y
163
62
Width
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
Stanford CS193p
Fall 2013

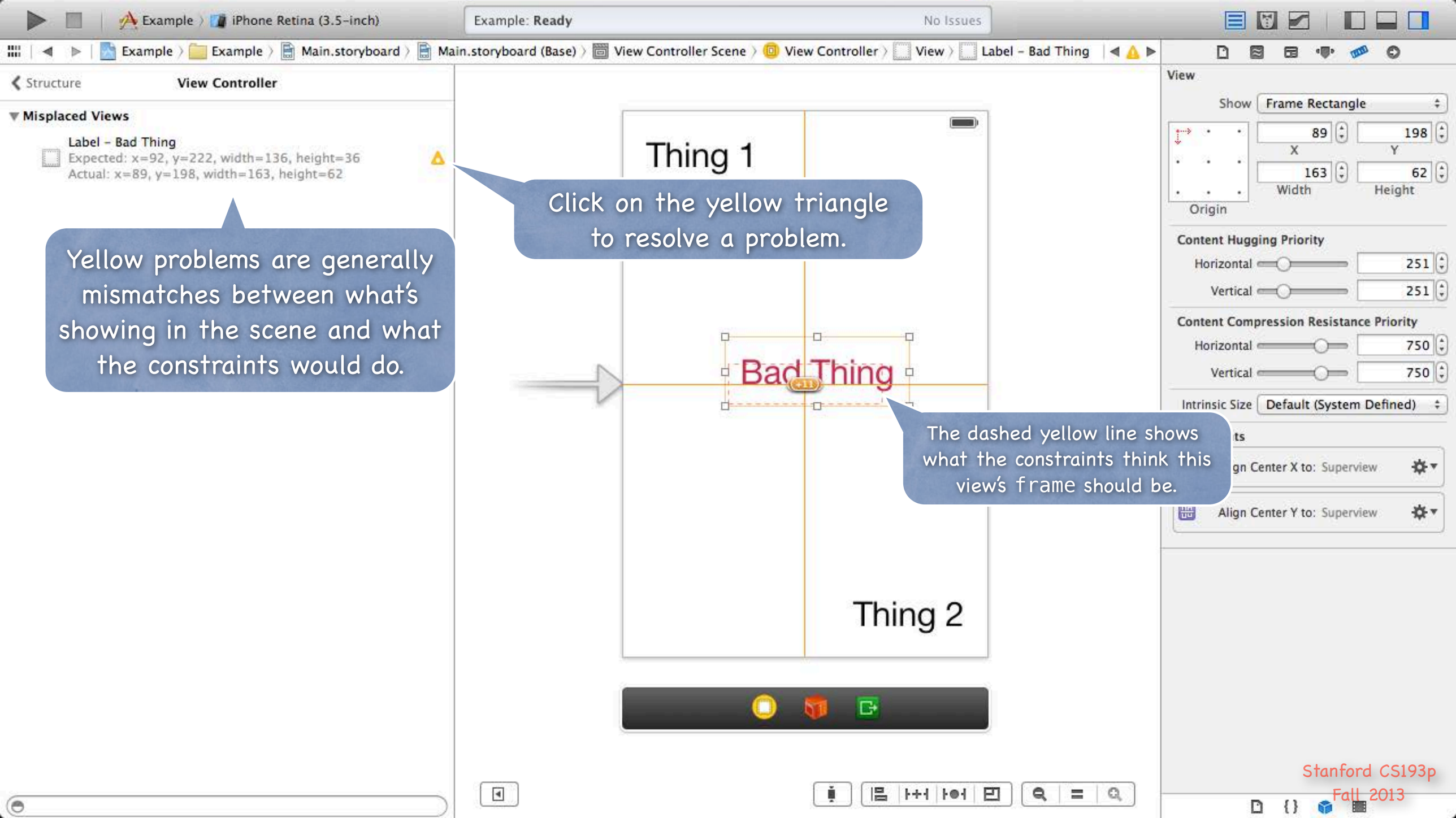

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Thing 1
Bad Thing
Thing 2
We're going to pick both the Horizontal and Vertical Centering options ("in Container" means in our superview).
Add New Alignment Constraints
Leading Edges
Trailing Edges
Top Edges
Bottom Edges
Horizontal Centers
Vertical Centers
Baselines
Horizontal Center in Container
Vertical Center in Container
Update Frames
None
Add Constraints
If you pick 2 (or more) views at once (using shift-click), you can also align them in all these ways.
View
Show
Frame Rectangle
89
198
X
Y
163
62
Width
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
Stanford CS193p
Fall 2013

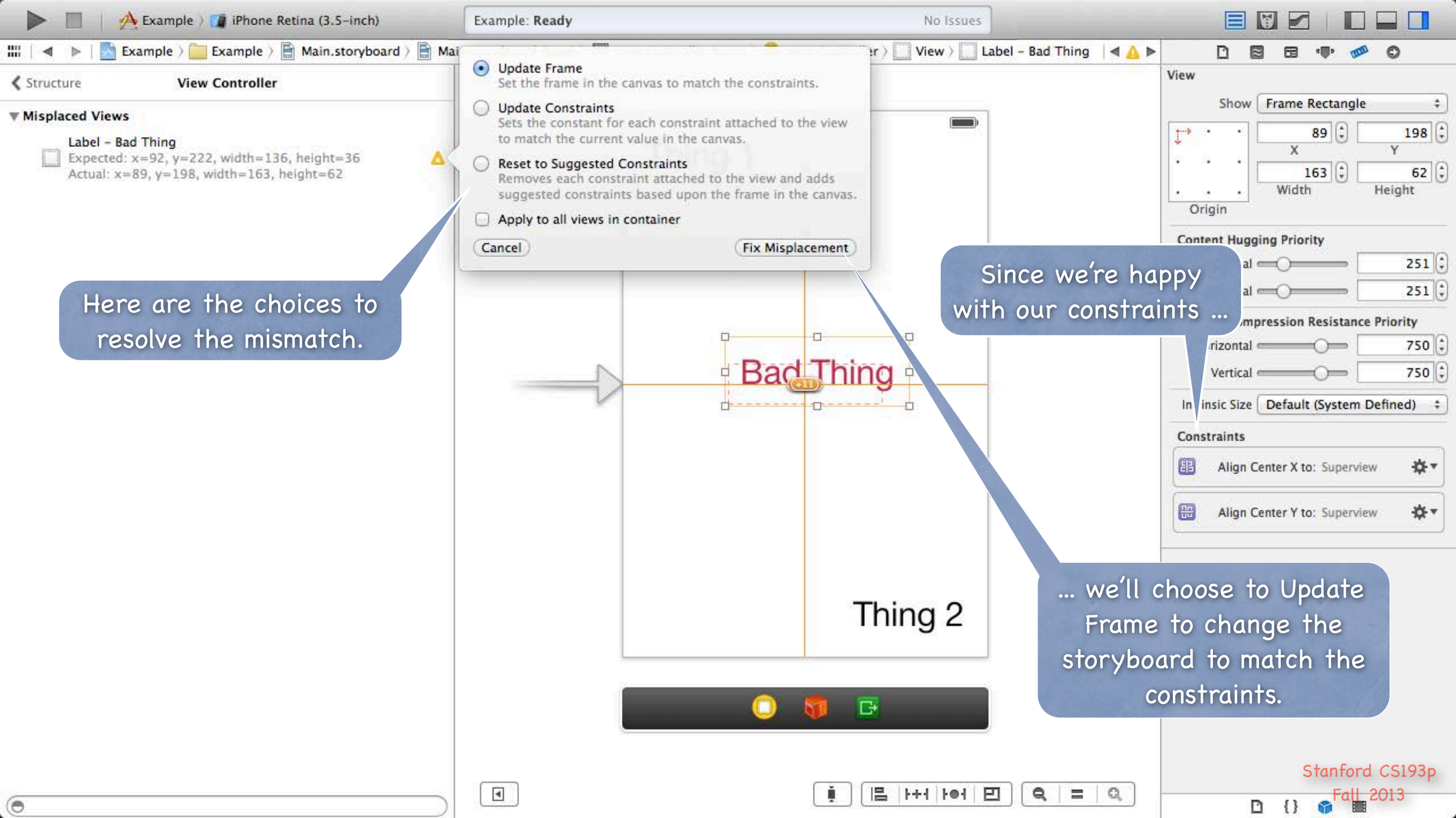

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Thing 1
Bad Thing
Thing 2
View
Show
Frame Rectangle
89
198
X
Y
163
62
Width
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
Add New Alignment Constraints
Leading Edges
Trailing Edges
Top Edges
Bottom Edges
Horizontal Centers
Vertical Centers
Baselines
Horizontal Center in Container
0
Vertical Center in Container
0
Update Frames
None
Add 2 Constraints
Clicking here adds the 2 new constraints.
Stanford CS193p
Fall 2013

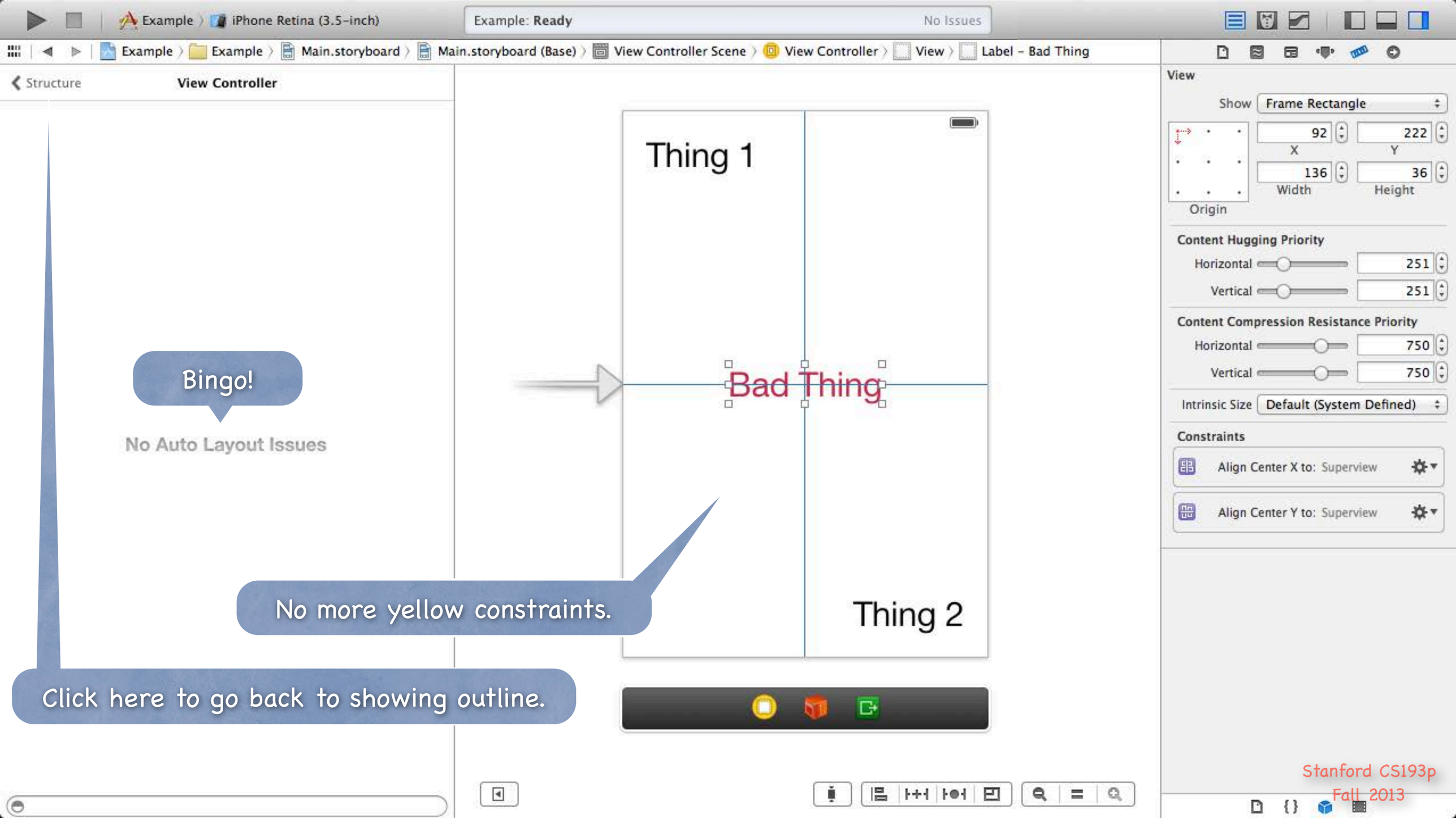

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
View
Label – Bad Thing
Thing 1
Bad Thing
Thing 2
It added them!
Notice that they are drawn in yellow. This is because they don't match what is currently showing in the scene.
View
Show
Frame Rectangle
89
X
198
Y
163
Width
62
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center X to: Superview
Align Center Y to: Superview
Stanford CS193p
Fall 2013

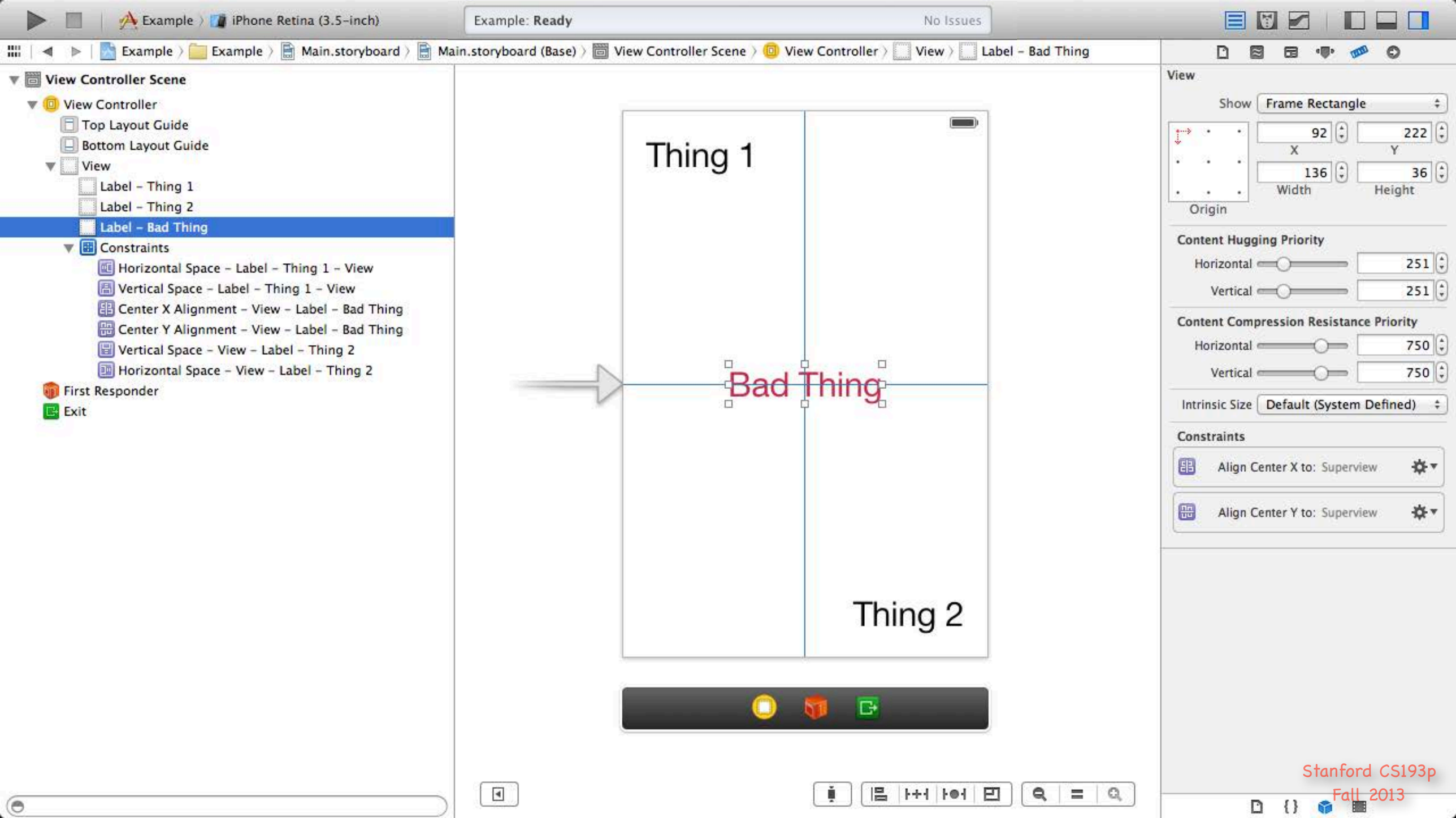

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
✓ Frame fo "Label – Bad Thing" will be different at run time.
Show Frame Rectangle
89 X
198 Y
163 Width
62 Height
Origin
Content Hugging Priority
Horizontal 251
Vertical 251
Content Compression Resistance Priority
Horizontal 750
Vertical 750
Intrinsic Size Default (System Defined)
Constraints
Align Center X to: Superview
Align Center Y to: Superview
Thing 1
Bad Thing
Thing 2
That fact is also reported here …
Stanford CS193p
Fall 2013

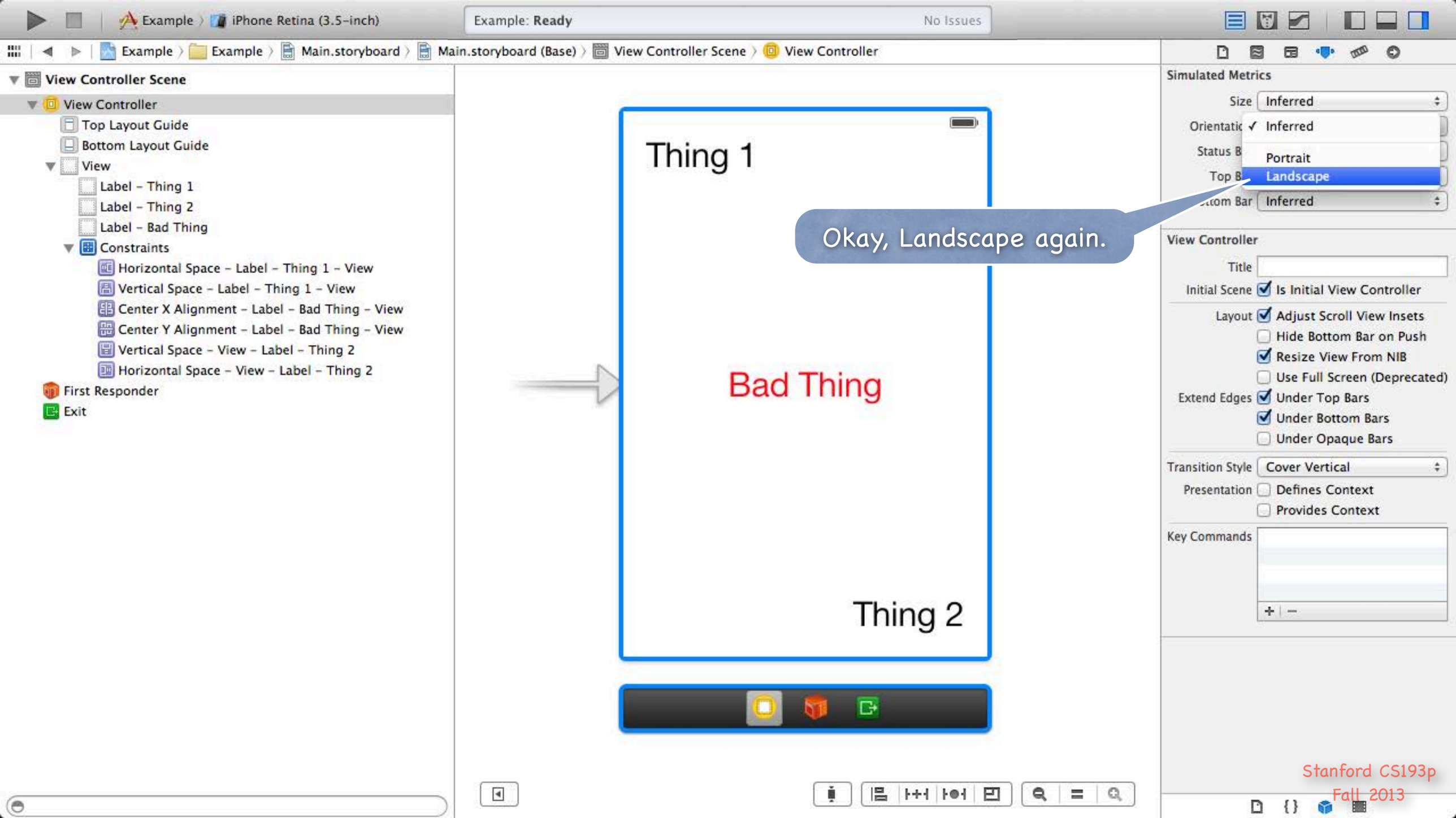

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center X Alignment – View – Label – Bad Thing
Center Y Alignment – View – Label – Bad Thing
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
We have not talked much about the Document Outline, but it is awesome! It shows everything (views, gestures, constraints, etc.) in your storyboard in outline form. You can select objects here and also ctrl-drag to/from them!
… and in the Document Outline.
Let's click on this!
Bad Thing
Thing 2
Click here to show the Document Outline.
View
Show
Frame Rectangle
89
X
198
Y
163
Width
62
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center X to: Superview
Align Center Y to: Superview
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Structure
View Controller
Misplaced Views
Label – Bad Thing
Expected: x=92, y=222, width=136, height=36
Actual: x=89, y=198, width=163, height=62
Yellow problems are generally mismatches between what's showing in the scene and what the constraints would do.
Click on the yellow triangle to resolve a problem.
Thing 1
Bad Thing
Thing 2
The dashed yellow line shows what the constraints think this view's frame should be.
View
Show
Frame Rectangle
89
198
X
Y
163
62
Width
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
gn Center X to: Superview
Align Center Y to: Superview
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Mai
View › Label – Bad Thing
Structure
View Controller
Misplaced Views
Label – Bad Thing
Expected: x=92, y=222, width=136, height=36
Actual: x=89, y=198, width=163, height=62
Update Frame
Set the frame in the canvas to match the constraints.
Update Constraints
Sets the constant for each constraint attached to the view to match the current value in the canvas.
Reset to Suggested Constraints
Removes each constraint attached to the view and adds suggested constraints based upon the frame in the canvas.
Apply to all views in container
Cancel
Fix Misplacement
Here are the choices to resolve the mismatch.
Since we're happy with our constraints ...
... we'll choose to Update Frame to change the storyboard to match the constraints.
Bad Thing
Thing 2
View
Show
Frame Rectangle
89
198
X
Y
163
62
Width
Height
Origin
Content Hugging Priority
251
251
Compression Resistance Priority
750
750
Vertical
Intrinsic Size
Default (System Defined)
Constraints
Align Center X to: Superview
Align Center Y to: Superview
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Structure
View Controller
Bingo!
No Auto Layout Issues
No more yellow constraints.
Click here to go back to showing outline.
Thing 1
Bad Thing
Thing 2
View
Show
Frame Rectangle
92
X
222
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center X to: Superview
Align Center Y to: Superview
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center X Alignment – View – Label – Bad Thing
Center Y Alignment – View – Label – Bad Thing
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
View
Show
Frame Rectangle
92
X
222
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center X to: Superview
Align Center Y to: Superview
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center X Alignment – Label – Bad Thing – View
Center Y Alignment – Label – Bad Thing – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
Okay, Landscape again.
Simulated Metrics
Size
Inferred
Orientatic
Inferred
Portrait
Landscape
Status B
Top B
tom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands

Example › iPhone Retina (3.5-inch)

Example: **Ready** No Issues

Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller

- View Controller Scene
  - View Controller
    - Top Layout Guide
    - Bottom Layout Guide
    - View
      - Label – Thing 1
      - Label – Thing 2
      - Label – Bad Thing
      - Constraints
        - Horizontal Space – Label – Thing 1 – View
        - Vertical Space – Label – Thing 1 – View
        - Center X Alignment – Label – Bad Thing – View
        - Center Y Alignment – Label – Bad Thing – View
        - Vertical Space – View – Label – Thing 2
        - Horizontal Space – View – Label – Thing 2
  - First Responder
  - Exit

Simulated Metrics

Size: Inferred
Orientation: Landscape
Status Bar: Inferred
Top Bar: Inferred
Bottom Bar: Inferred

View Controller

Title:
Initial Scene: Is Initial View Controller

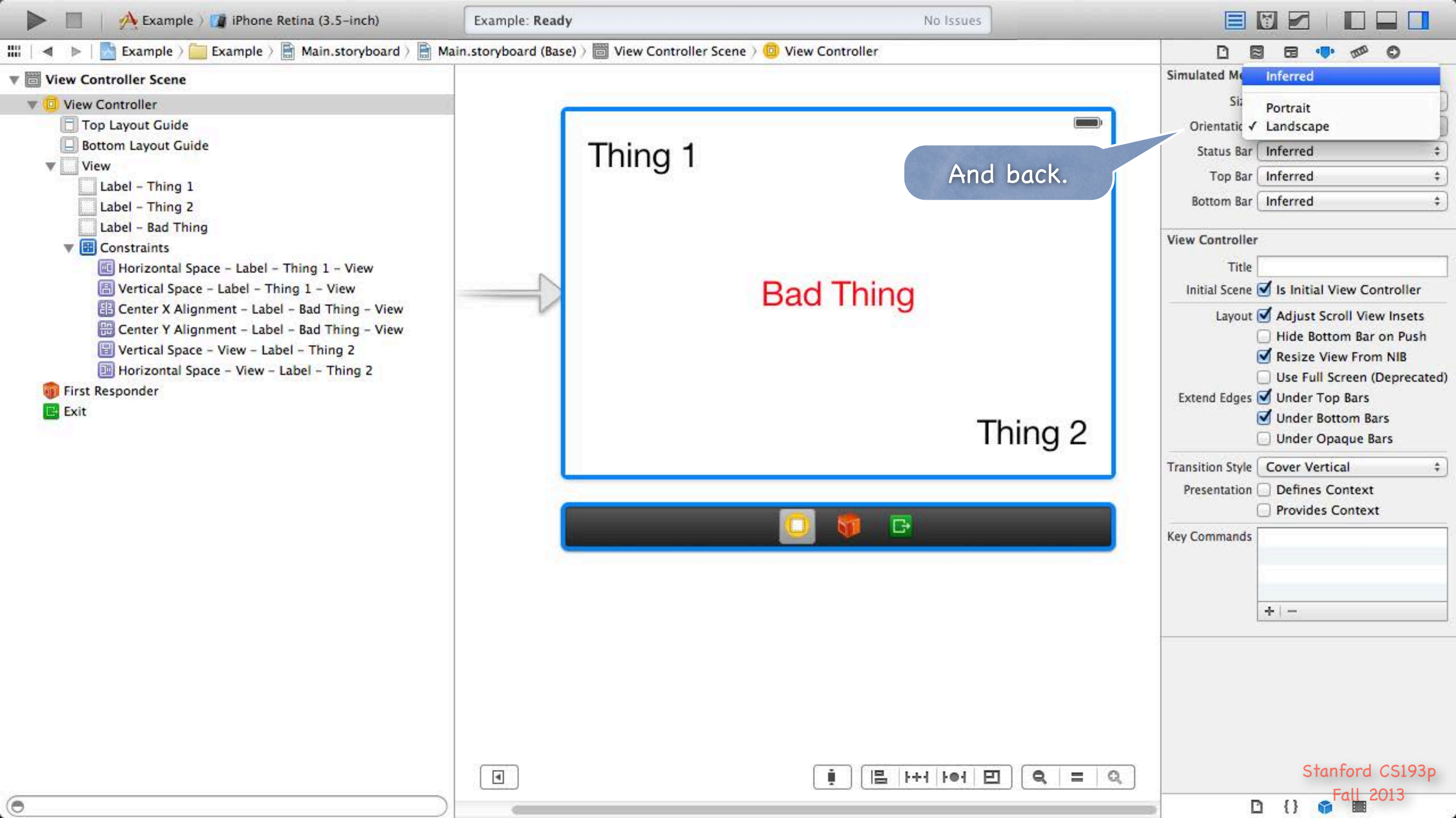
Layout: Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)

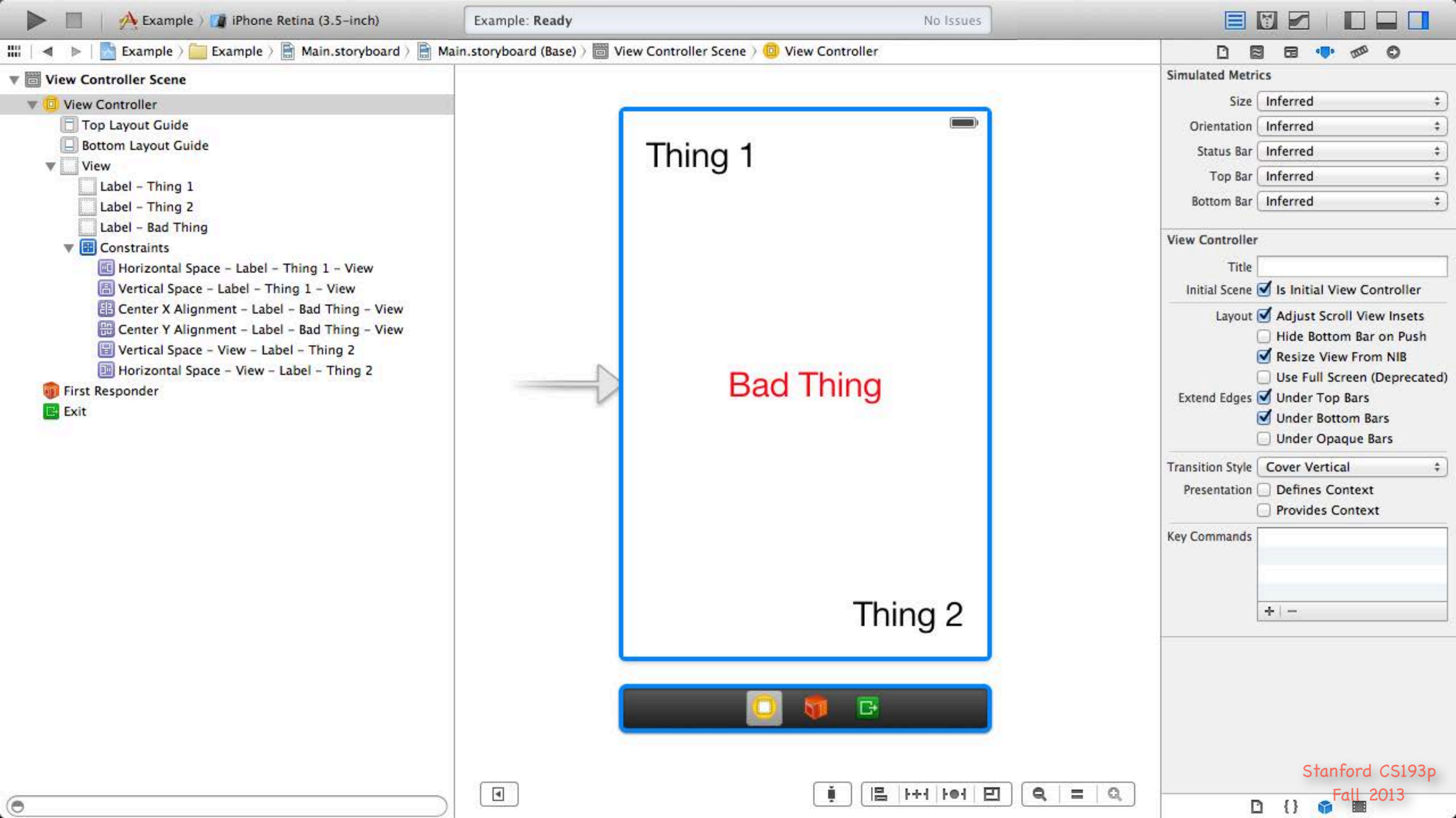
Extend Edges: Under Top Bars
Under Bottom Bars
Under Opaque Bars

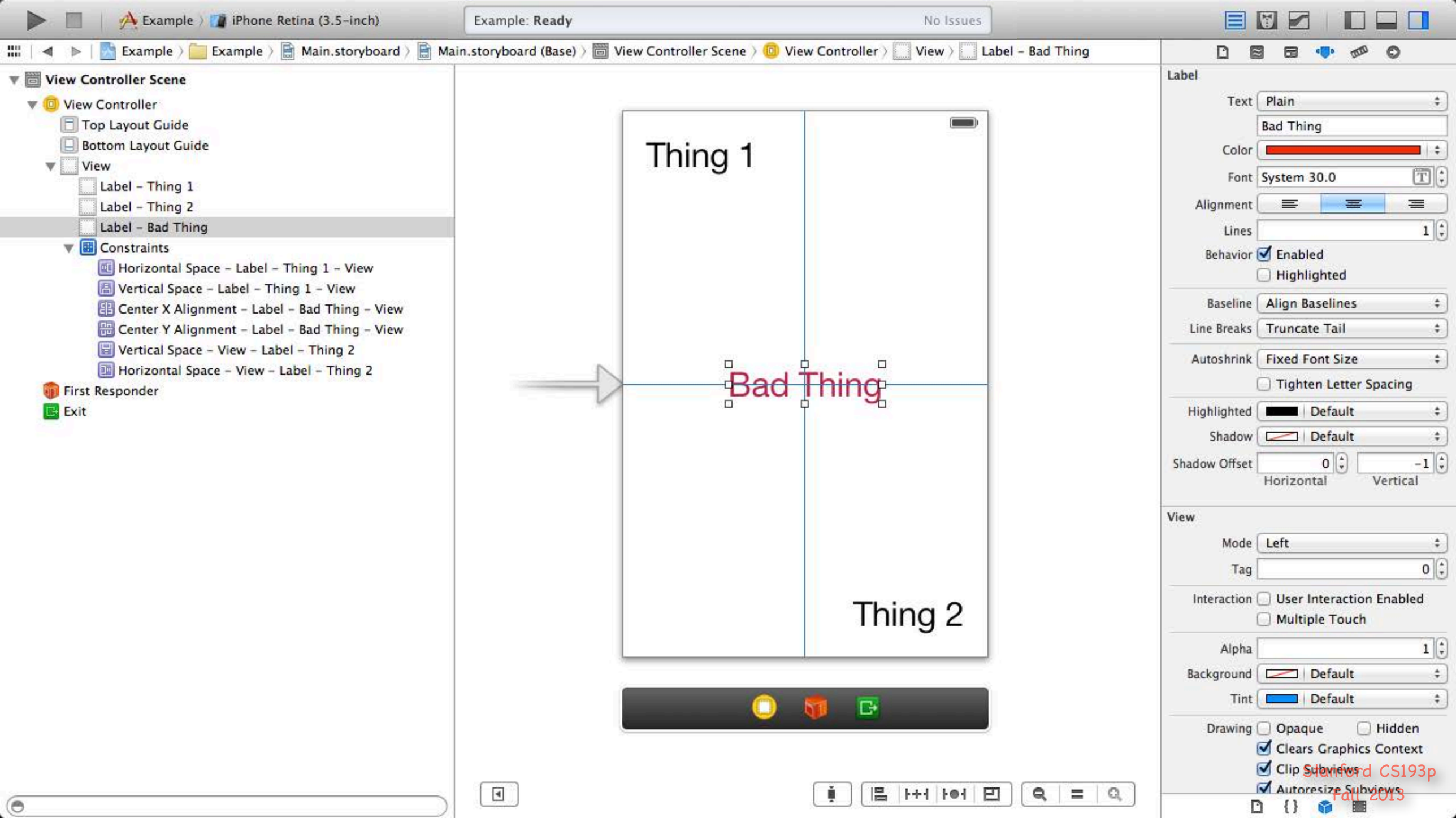
Transition Style: Cover Vertical

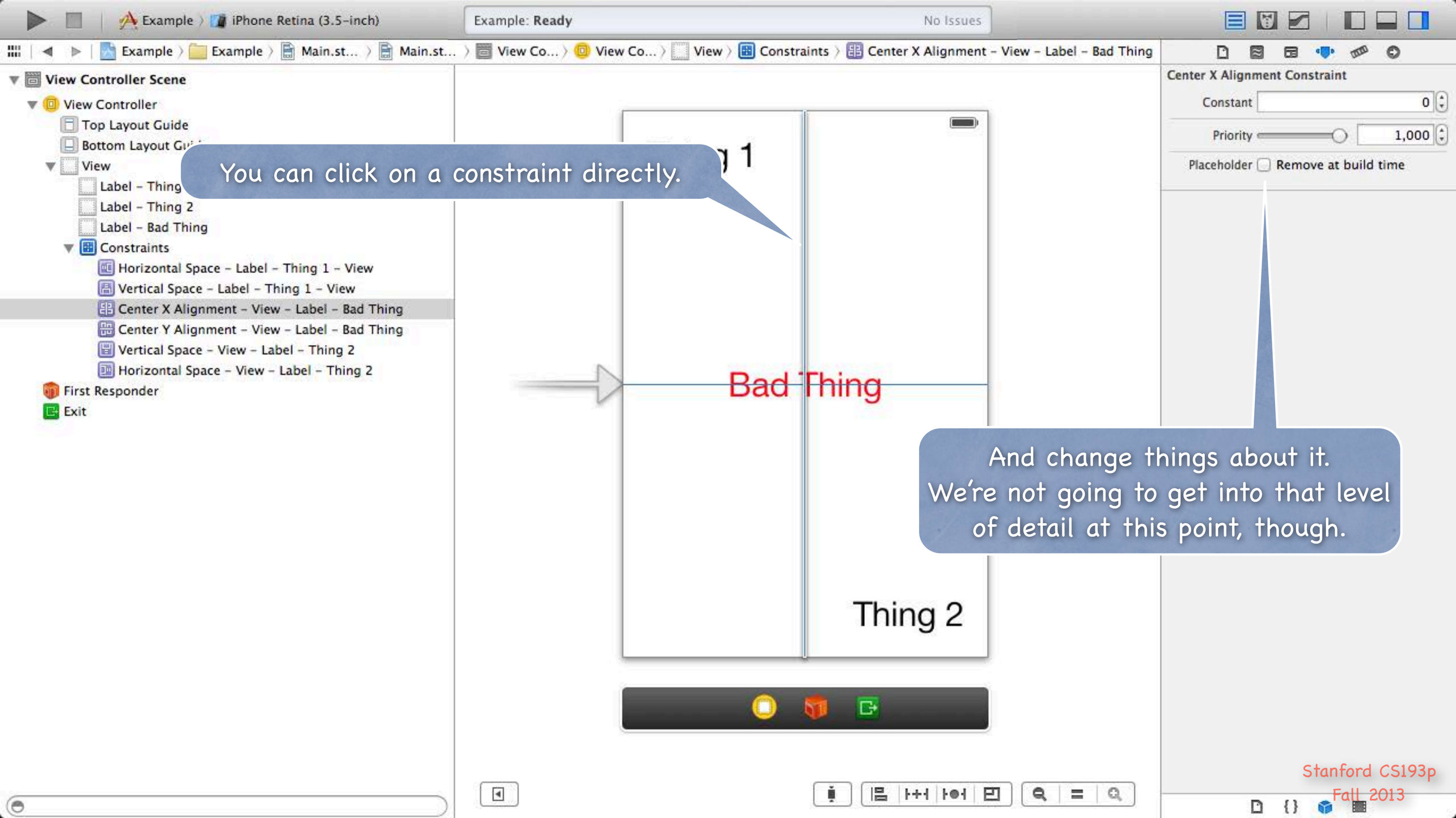
Presentation: Defines Context
Provides Context

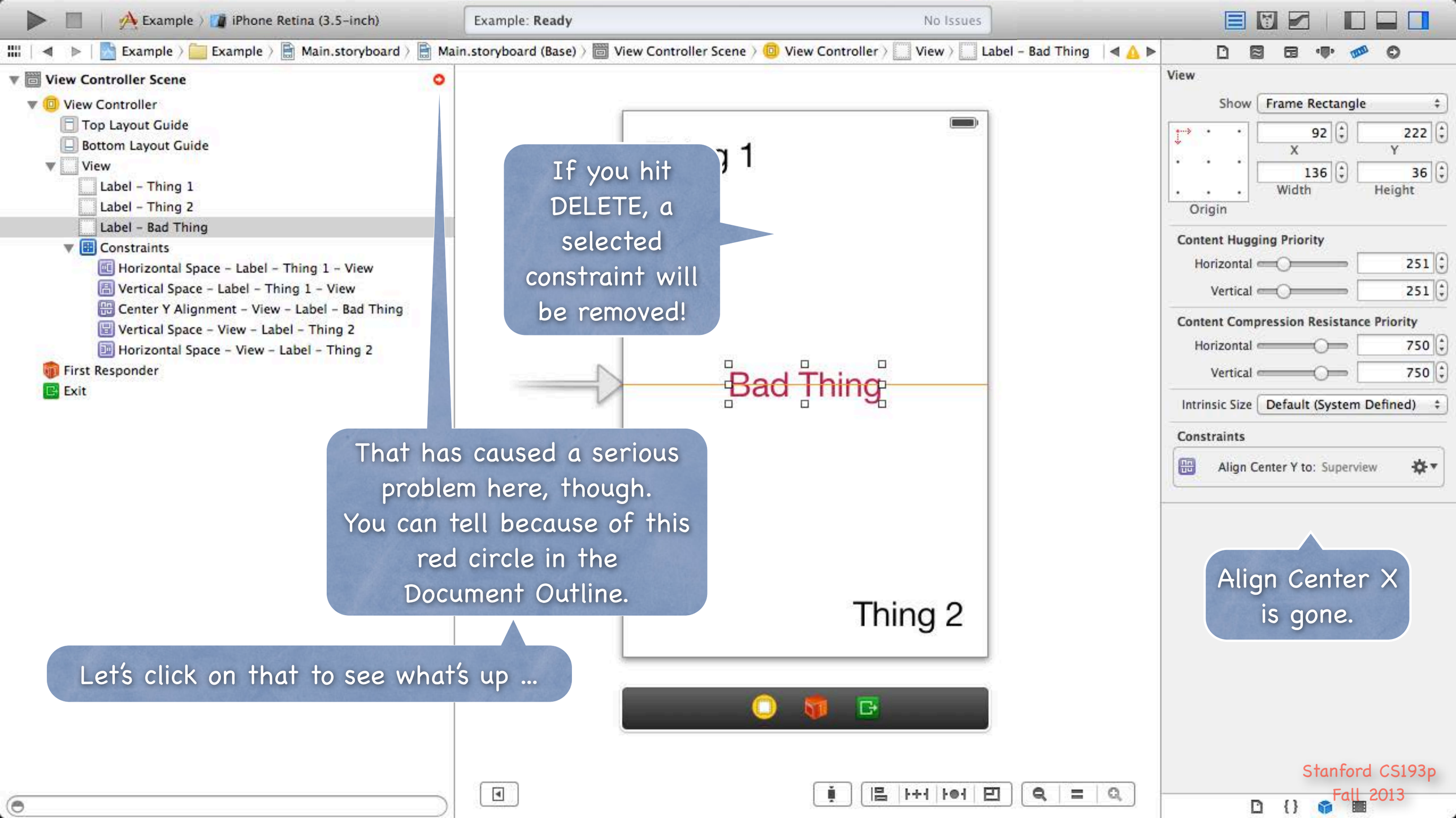
Key Commands

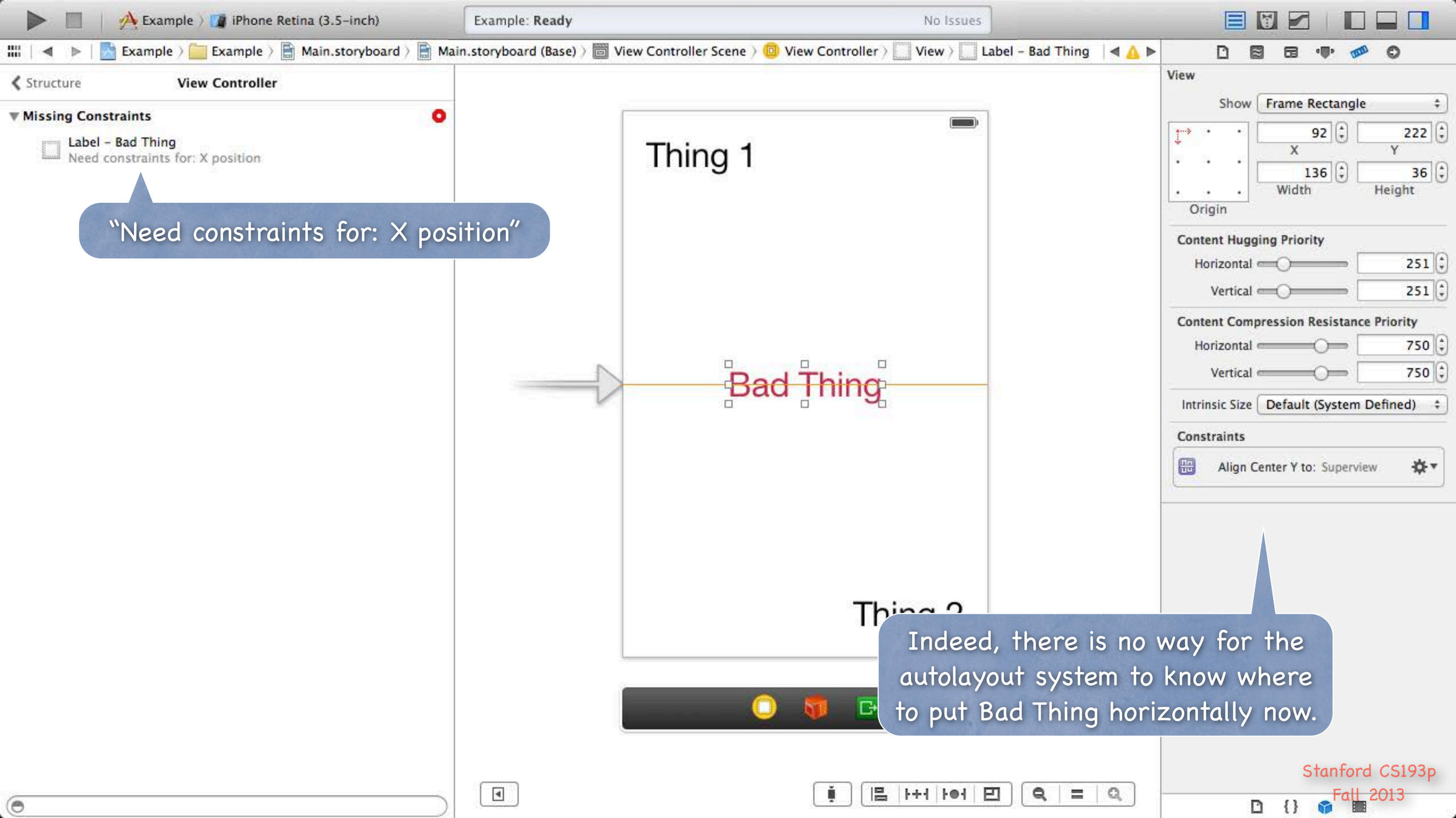

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center X Alignment – Label – Bad Thing – View
Center Y Alignment – Label – Bad Thing – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
And back.
Simulated Me
Inferred
Portrait
Landscape
Orientatic
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

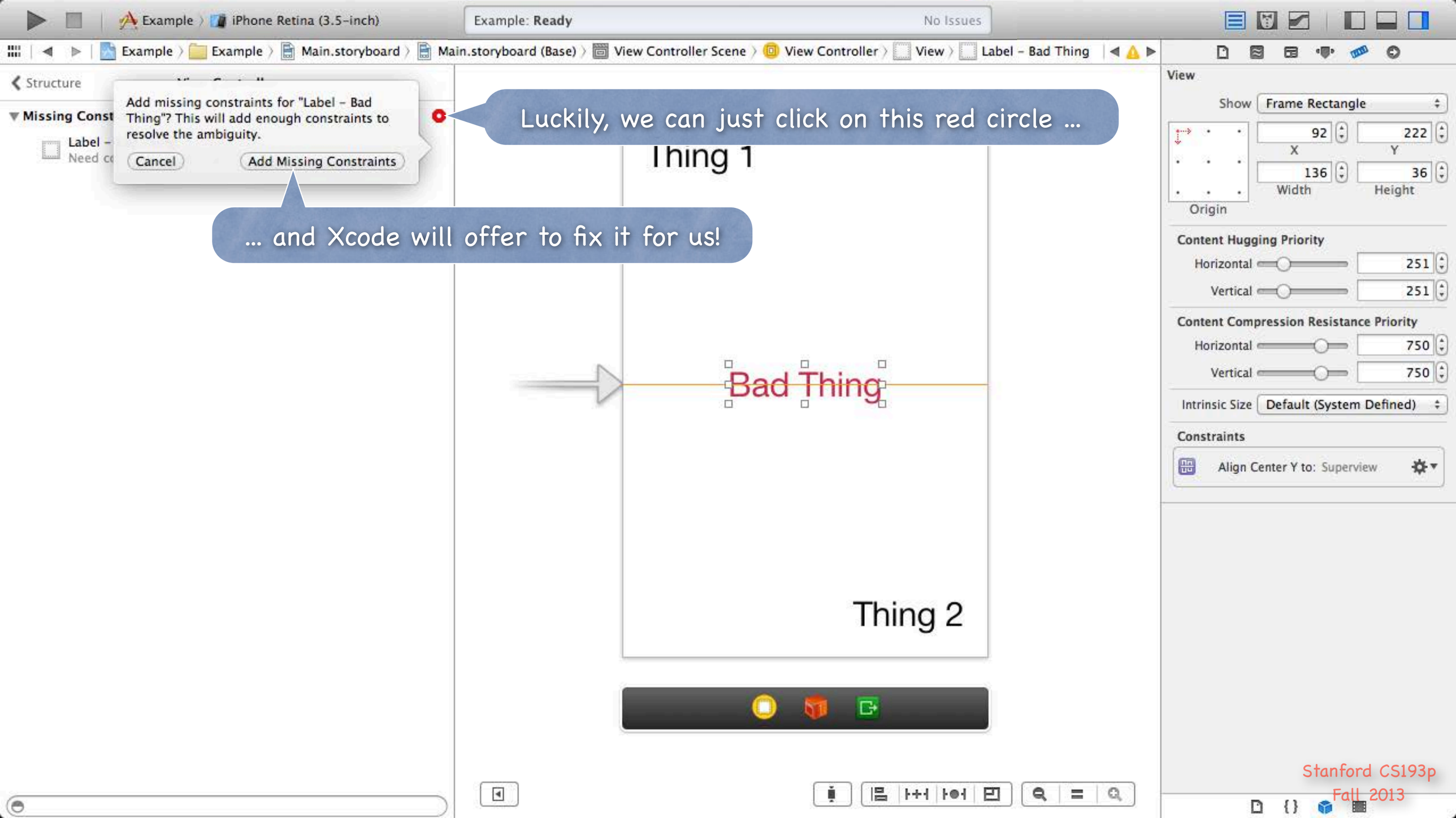

Example
iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example
Example
Main.storyboard
Main.storyboard (Base)
View Controller Scene
View Controller
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center X Alignment – Label – Bad Thing – View
Center Y Alignment – Label – Bad Thing – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
Simulated Metrics
Size
Inferred
Orientation
Inferred
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

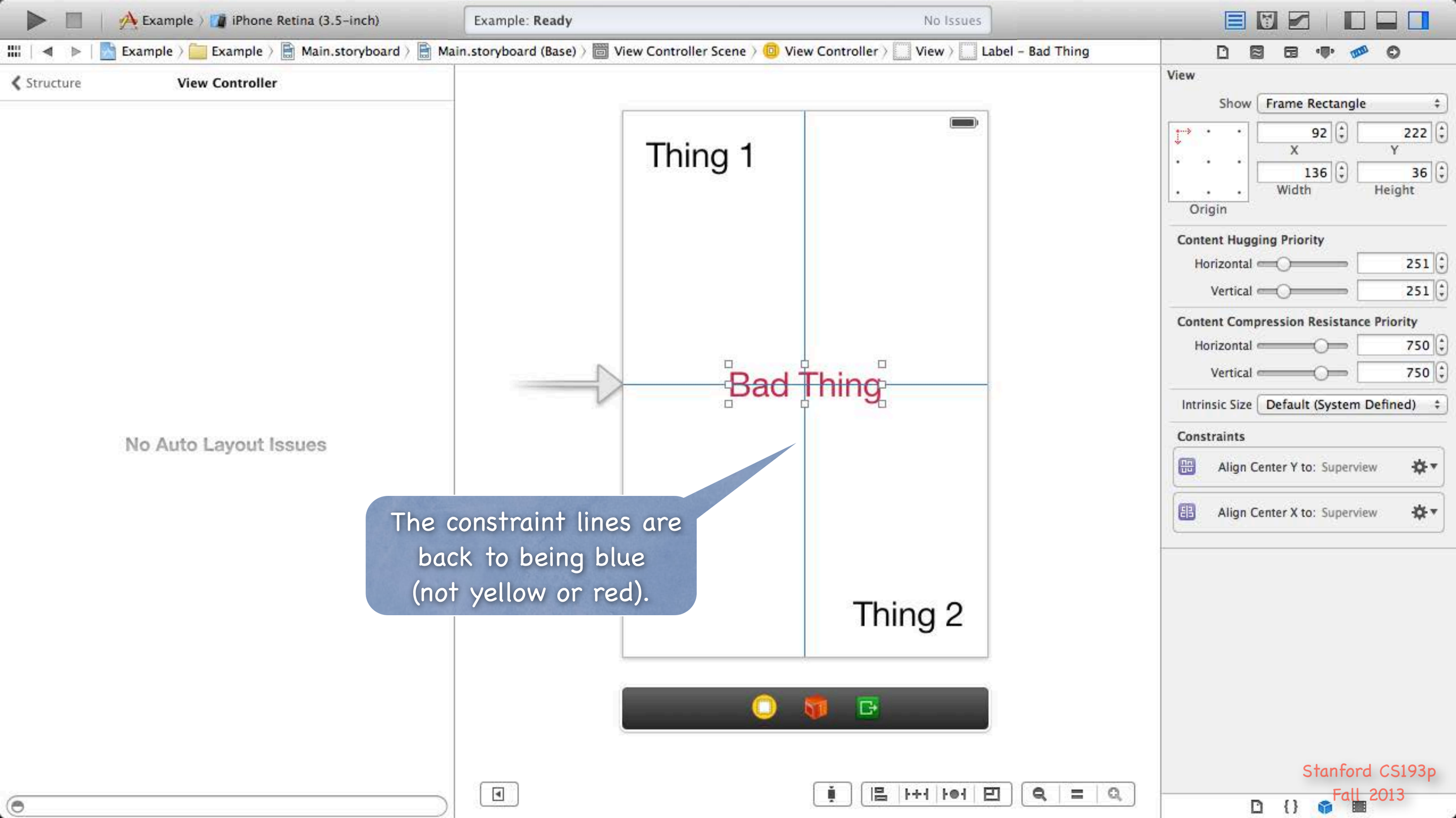

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center X Alignment – Label – Bad Thing – View
Center Y Alignment – Label – Bad Thing – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
Label
Text
Plain
Bad Thing
Color
Font
System 30.0
Alignment
Lines
1
Behavior
Enabled
Highlighted
Baseline
Align Baselines
Line Breaks
Truncate Tail
Autoshrink
Fixed Font Size
Tighten Letter Spacing
Highlighted
Default
Shadow
Default
Shadow Offset
0
-1
Horizontal
Vertical
View
Mode
Left
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.st... › Main.st... › View Co... › View Co... › View › Constraints › Center X Alignment – View – Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Gui
View
Label – Thing
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center X Alignment – View – Label – Bad Thing
Center Y Alignment – View – Label – Bad Thing
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
You can click on a constraint directly.
Bad Thing
Thing 2
Center X Alignment Constraint
Constant
0
Priority
1,000
Placeholder
Remove at build time
And change things about it.
We're not going to get into that level of detail at this point, though.
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center Y Alignment – View – Label – Bad Thing
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
If you hit DELETE, a selected constraint will be removed!
Bad Thing
Thing 2
That has caused a serious problem here, though. You can tell because of this red circle in the Document Outline.
Let's click on that to see what's up ...
View
Show
Frame Rectangle
92
X
222
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center Y to: Superview
Align Center X is gone.
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Structure
View Controller
Missing Constraints
Label – Bad Thing
Need constraints for: X position
"Need constraints for: X position"
Thing 1
Bad Thing
Thing 2
Indeed, there is no way for the autolayout system to know where to put Bad Thing horizontally now.
View
Show
Frame Rectangle
92
X
222
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center Y to: Superview
Stanford CS193p
Fall 2013

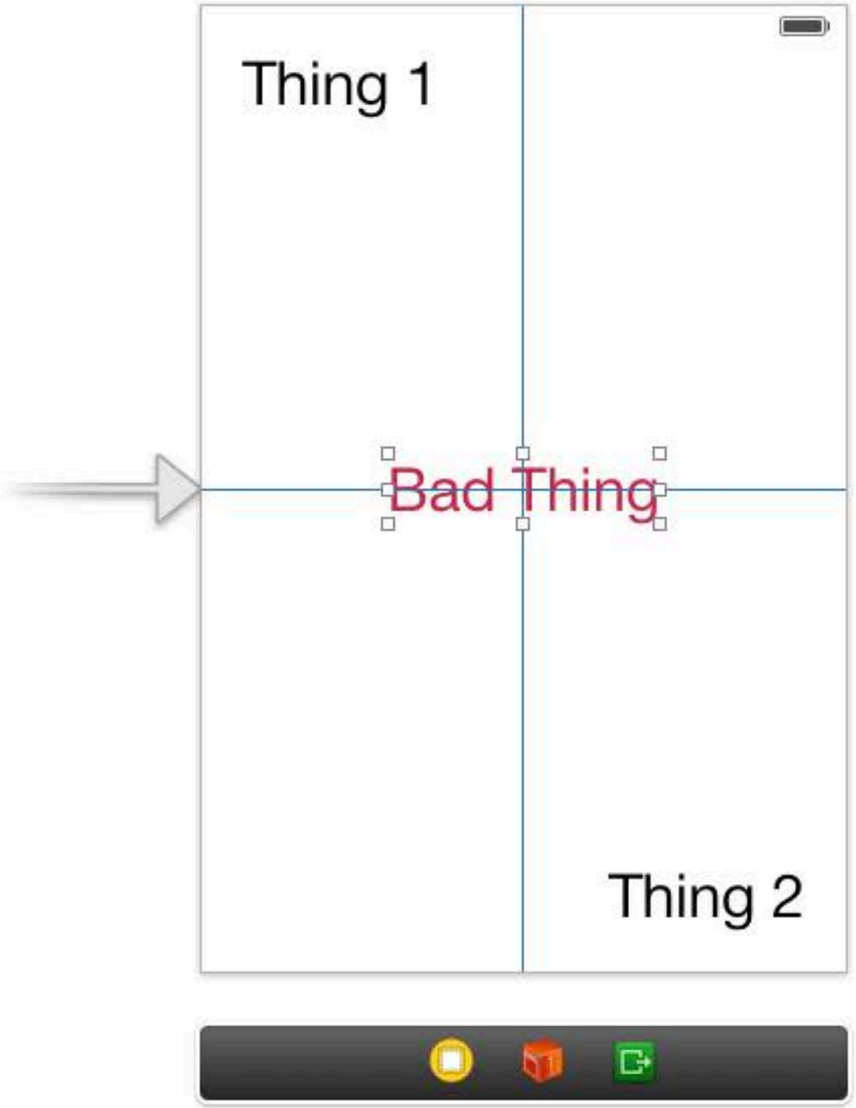

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Structure
Missing Const
Label –
Need c
Add missing constraints for "Label – Bad Thing"? This will add enough constraints to resolve the ambiguity.
Cancel
Add Missing Constraints
Luckily, we can just click on this red circle ...
... and Xcode will offer to fix it for us!
Thing 1
Bad Thing
Thing 2
View
Show
Frame Rectangle
92
X
222
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center Y to: Superview
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
Structure
View Controller
No Auto Layout Issues
Thing 1
Bad Thing
Thing 2
The constraint lines are back to being blue (not yellow or red).
View
Show
Frame Rectangle
92
X
222
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center Y to: Superview
Align Center X to: Superview
Stanford CS193p
Fall 2013

Example › iPhone Retina (3.5-inch) | Example: Ready | No Issues

Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing

- View Controller Scene
  - View Controller
    - Top Layout Guide
    - Bottom Layout Guide
    - View
      - Label – Thing 1
      - Label – Thing 2
      - Label – Bad Thing
      - Constraints
        - Horizontal Space – Label – Thing 1 – View
        - Vertical Space – Label – Thing 1 – View
        - Center Y Alignment – View – Label – Bad Thing
        - Center X Alignment – Label – Bad Thing – View
        - Vertical Space – View – Label – Thing 2
        - Horizontal Space – View – Label – Thing 2
  - First Responder
  - Exit



View

Show: Frame Rectangle

Origin: X 92, Y 222

Width 136, Height 36

Content Hugging Priority

Horizontal 251

Vertical 251

Content Compression Resistance Priority

Horizontal 750

Vertical 750

Intrinsic Size: Default (System Defined)

Constraints

Align Center Y to: Superview

Align Center X to: Superview

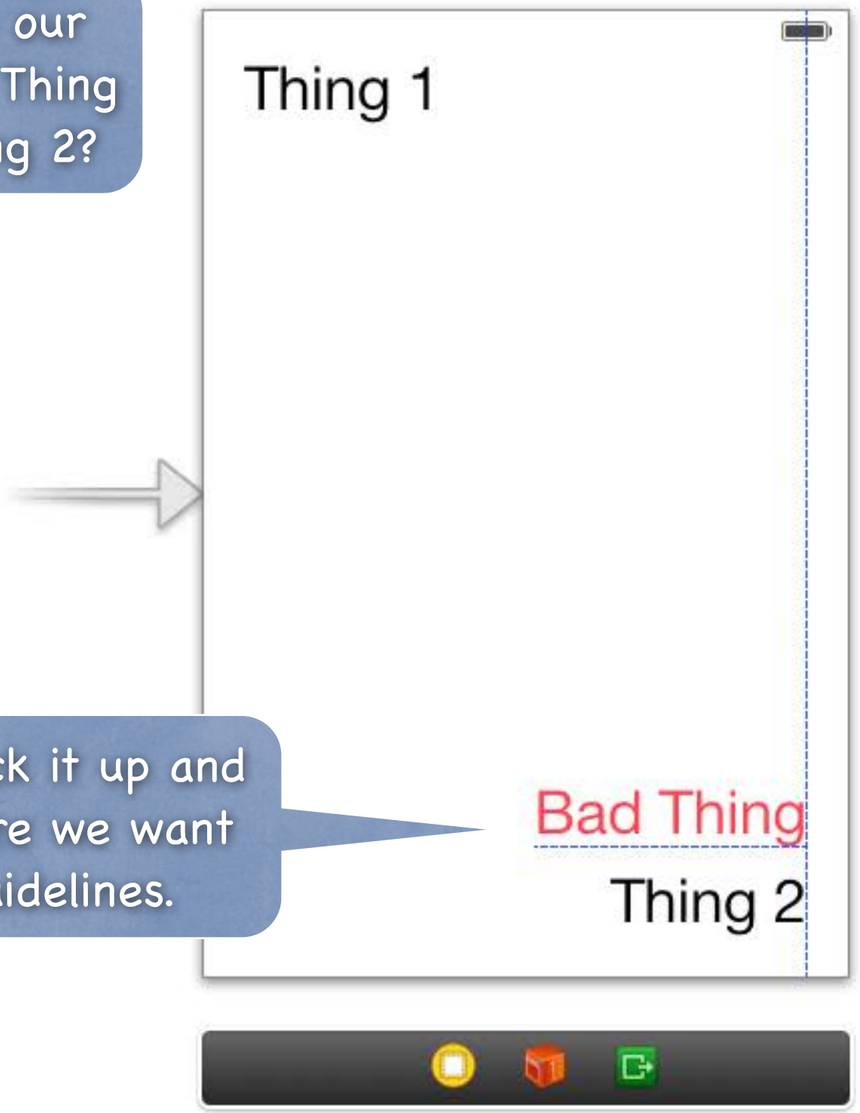
Stanford CS193p
Fall 2013

Example › iPhone Retina (3.5-inch) | Example: Ready | No Issues

Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View

- View Controller Scene
  - View Controller
    - Top Layout Guide
    - Bottom Layout Guide
    - View
      - Label – Thing 1
      - Label – Thing 2
      - Label – Bad Thing
      - Constraints
        - Horizontal Space – Label – Thing 1 – View
        - Vertical Space – Label – Thing 1 – View
        - Center Y Alignment – View – Label – Bad Thing
        - Center X Alignment – Label – Bad Thing – View
        - Vertical Space – View – Label – Thing 2
        - Horizontal Space – View – Label – Thing 2
  - First Responder
  - Exit

What if we change our minds and want Bad Thing to sit on top of Thing 2?



We can just pick it up and drag it to where we want with blue guidelines.

View

Show: Frame Rectangle

X: 0 Y: 0

Width: 320 Height: 480

Origin

Content Hugging Priority

Horizontal: 250

Vertical: 250

Content Compression Resistance Priority

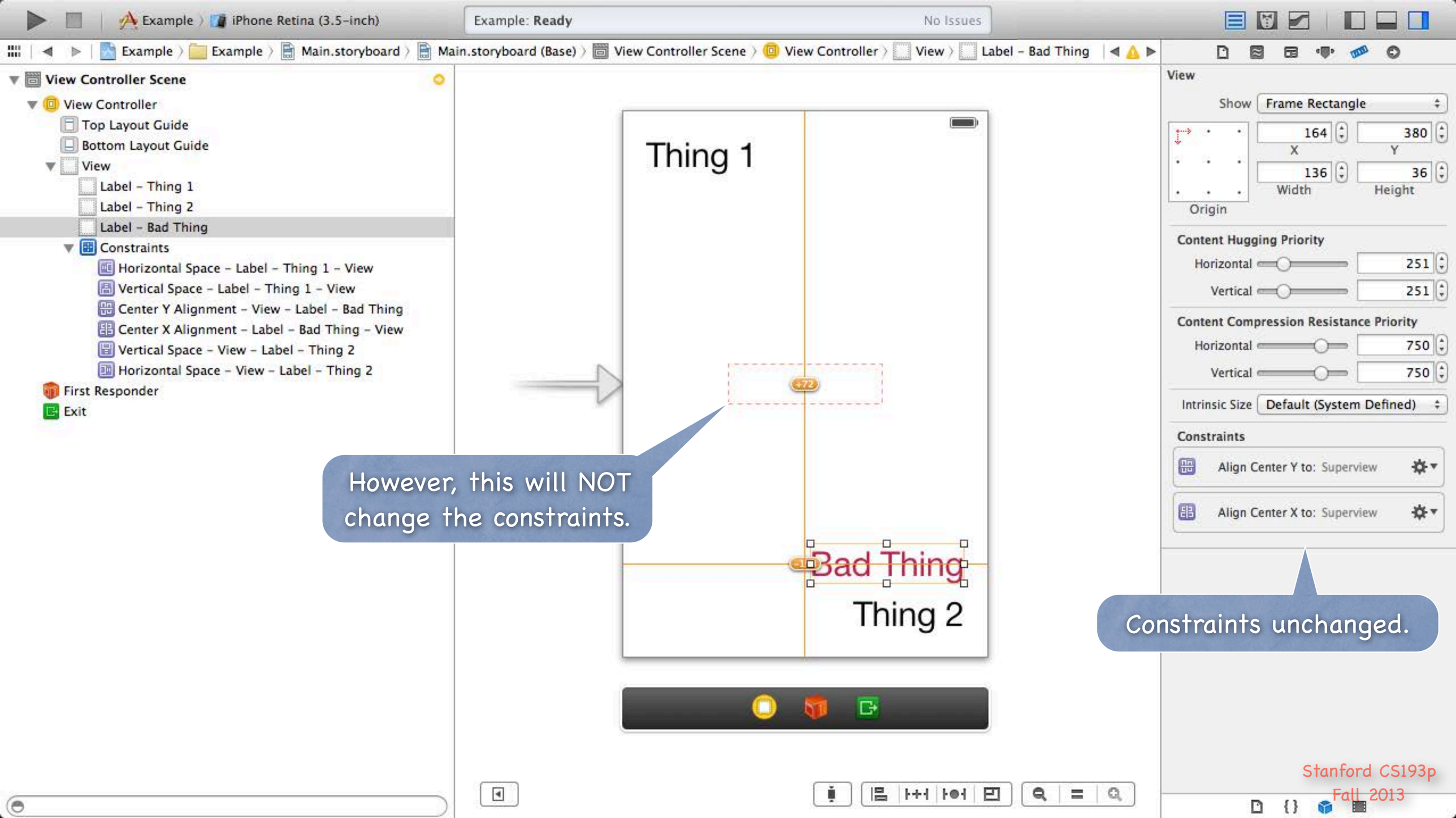
Horizontal: 750

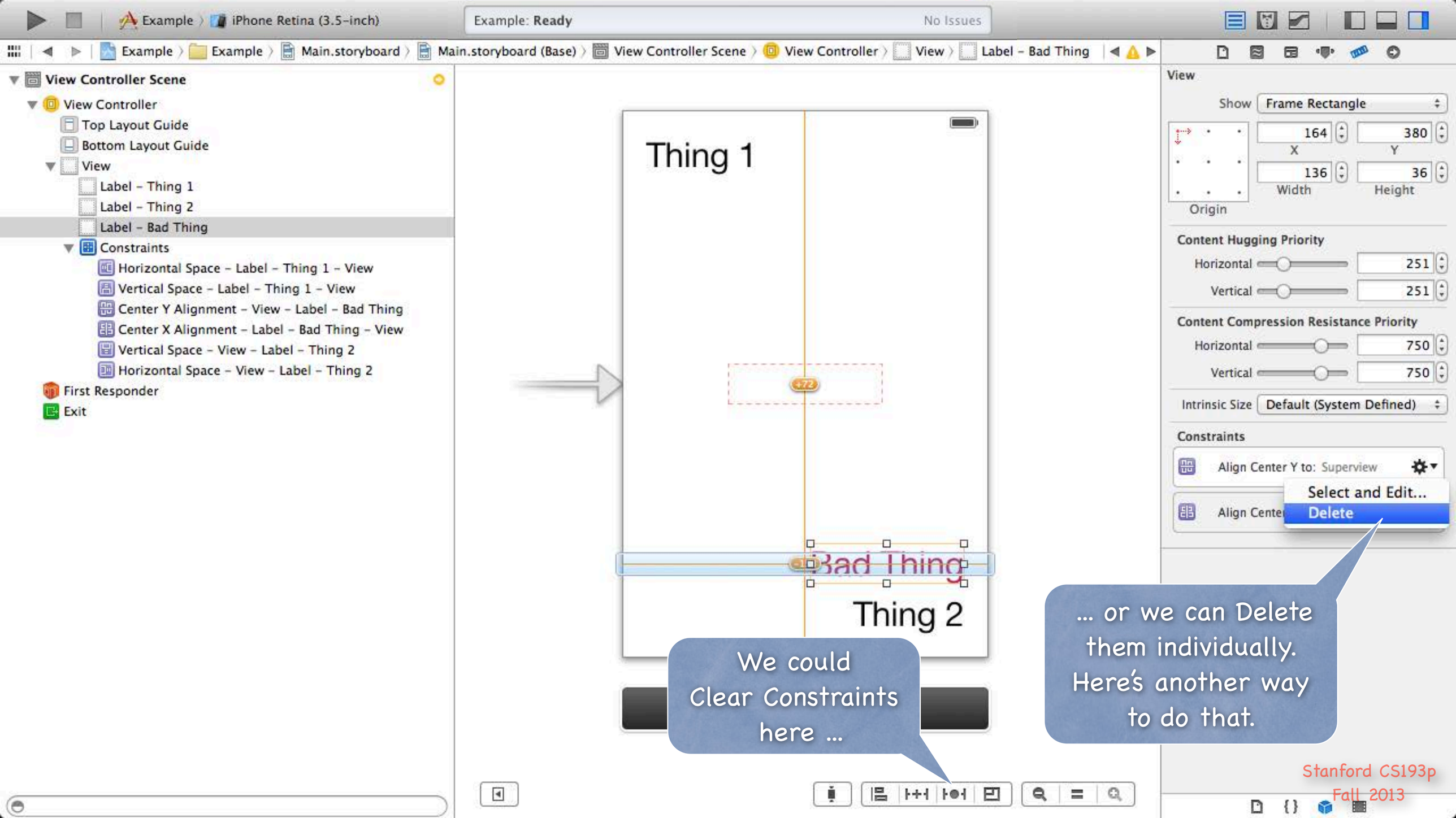
Vertical: 750

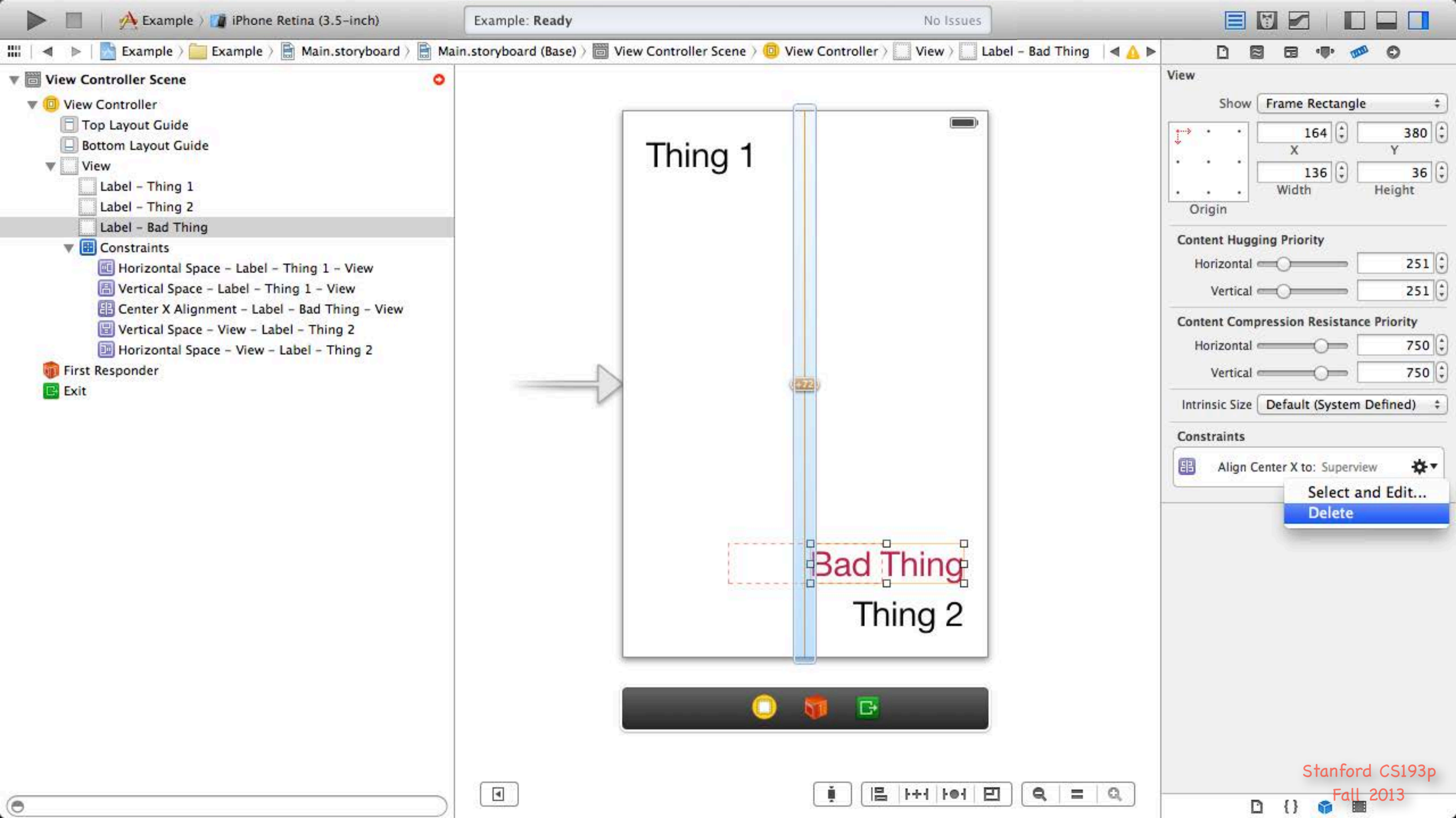
Intrinsic Size: Default (System Defined)

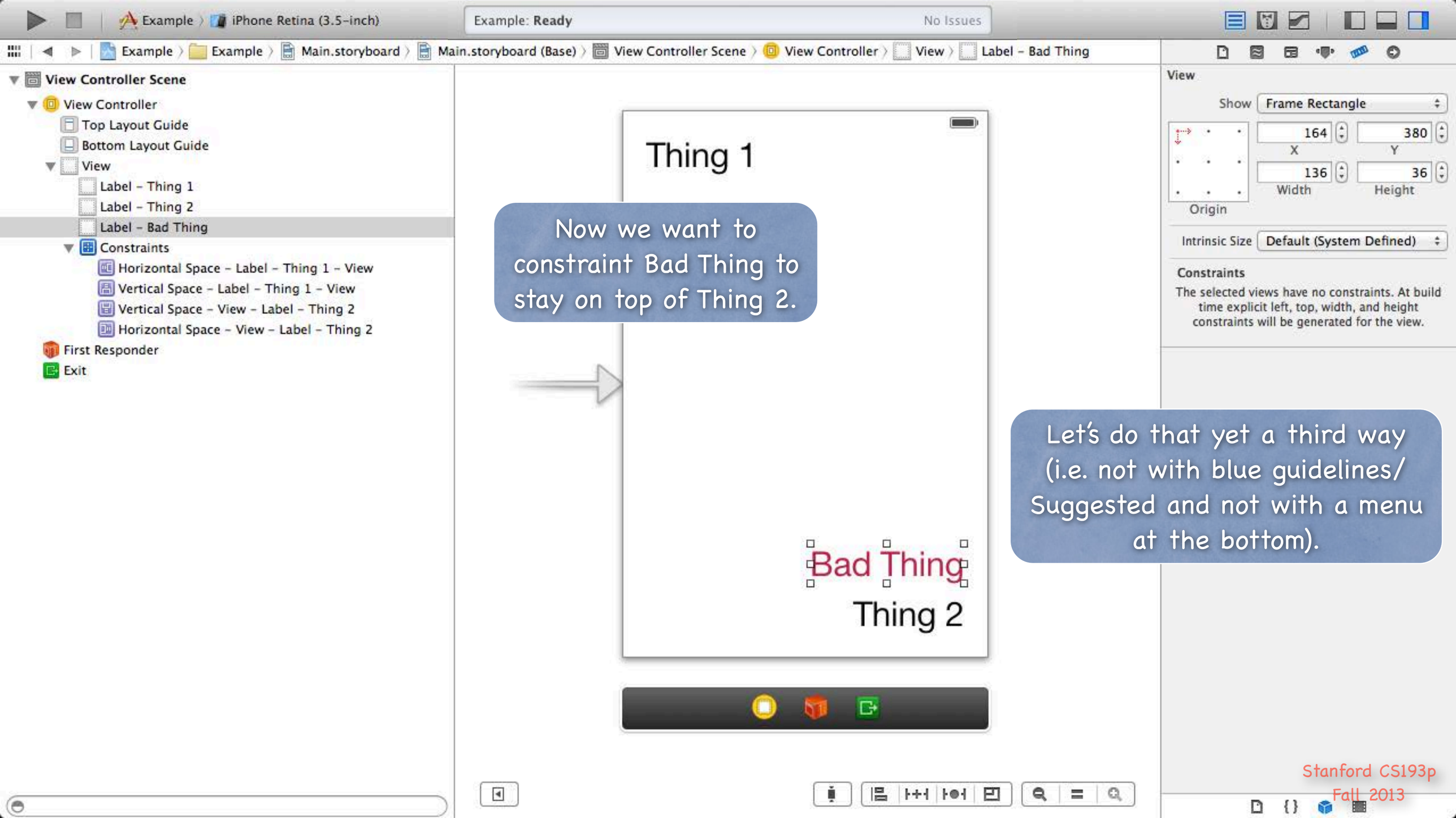
Constraints

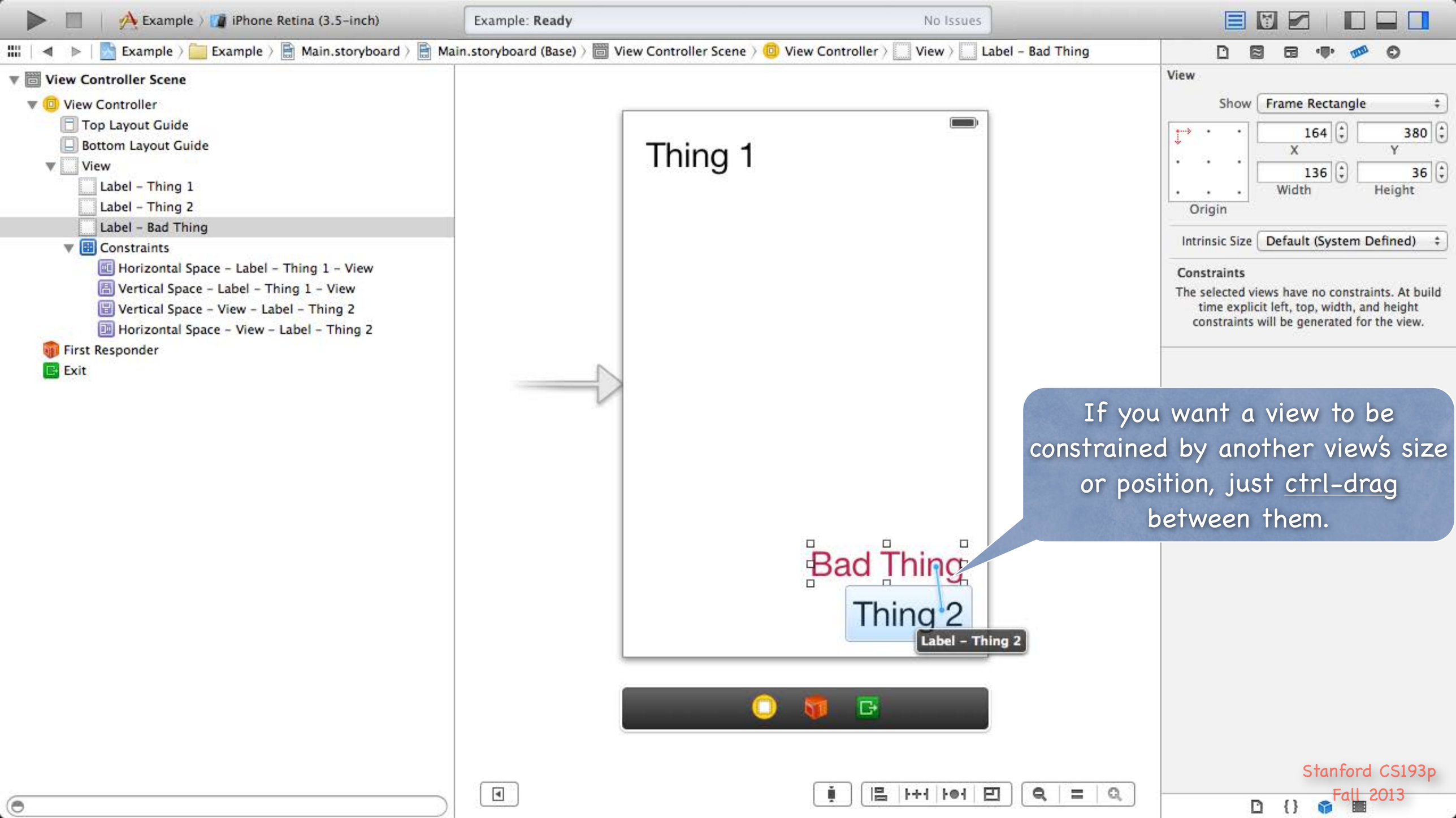
- Leading Space to: Label – Thi... Equals: Default
- Top Space to: Label – Thi... Equals: Default
- Align Center Y to: Label – Bad...
- Align Center X to: Label – Bad...
- Bottom Space to: Label – Thi... Equals: Default
- Trailing Space to: Label – Thi... Equals: Default

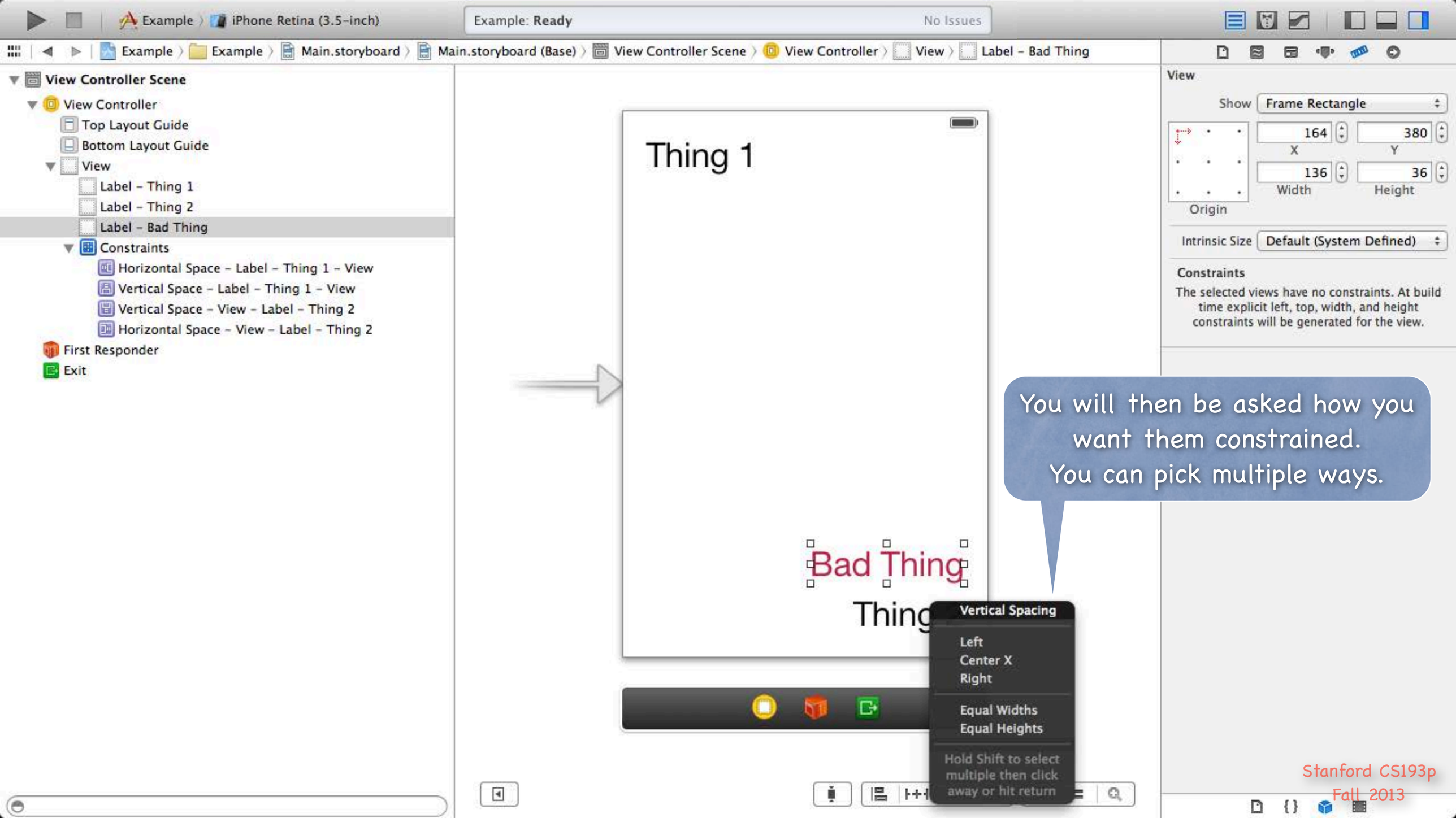

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center Y Alignment – View – Label – Bad Thing
Center X Alignment – Label – Bad Thing – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
However, this will NOT change the constraints.
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center Y to: Superview
Align Center X to: Superview
Constraints unchanged.
Stanford CS193p
Fall 2013

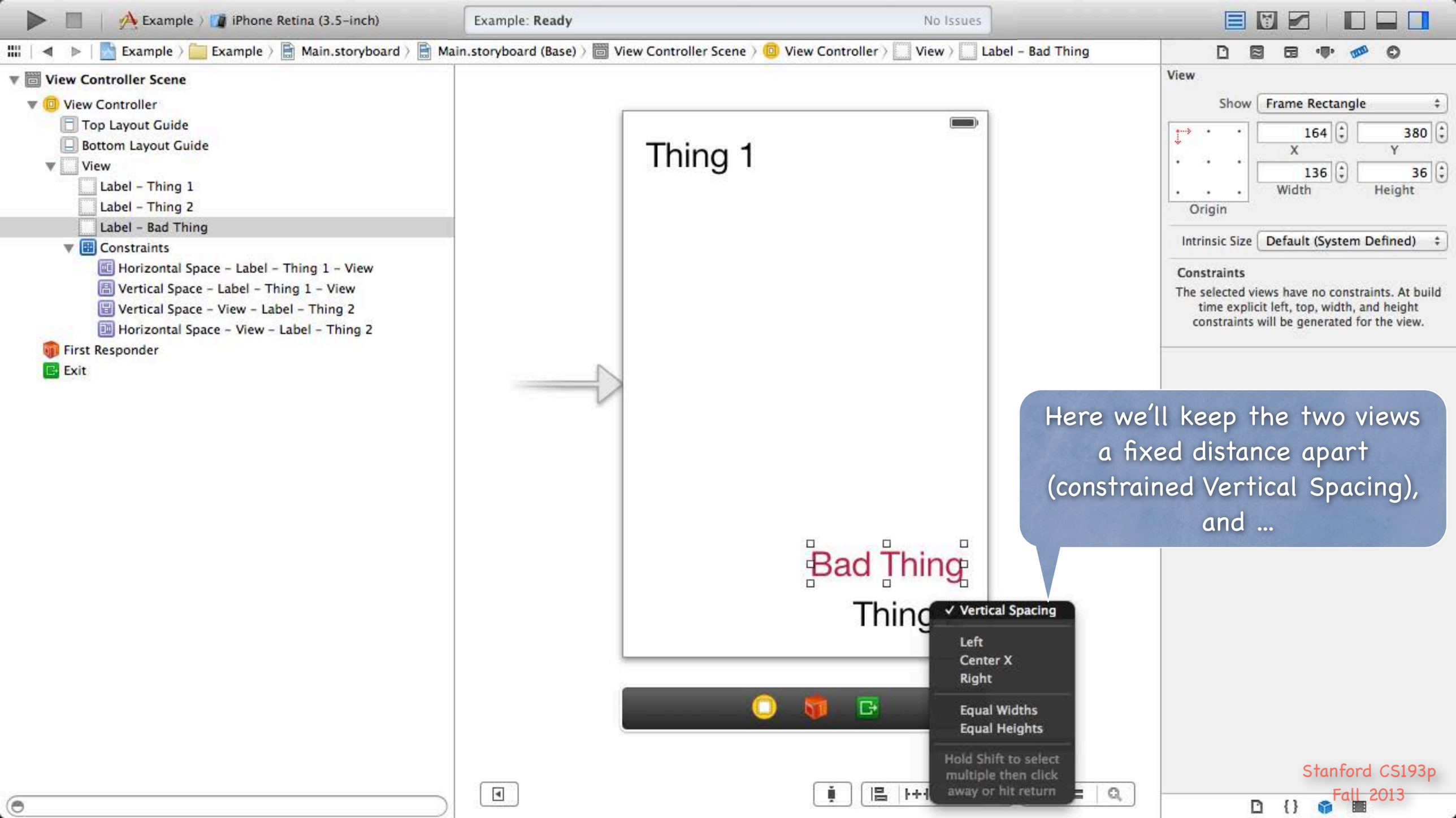

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center Y Alignment – View – Label – Bad Thing
Center X Alignment – Label – Bad Thing – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
We could Clear Constraints here …
… or we can Delete them individually. Here's another way to do that.
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center Y to: Superview
Select and Edit...
Delete
Stanford CS193p
Fall 2013

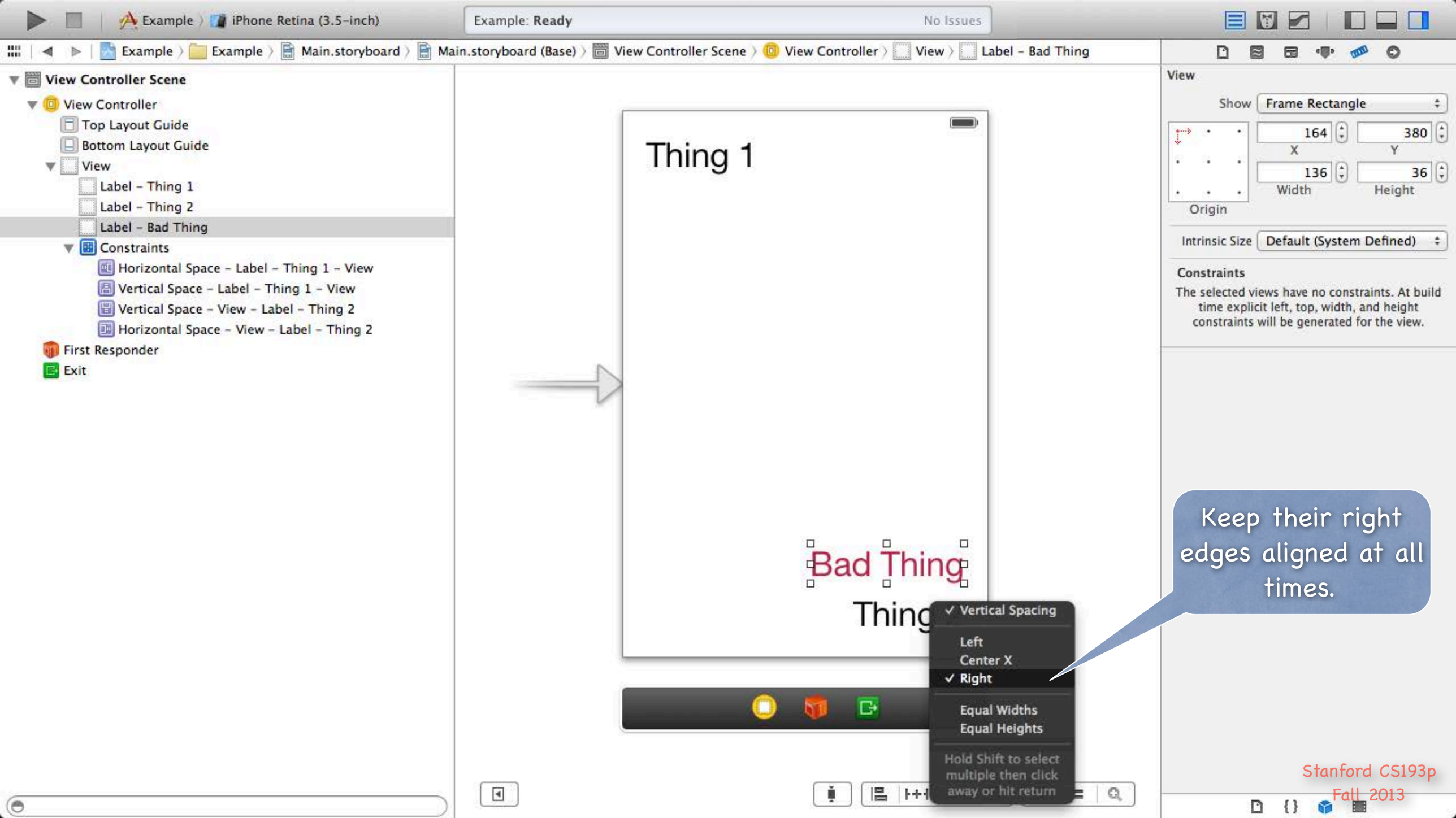

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Center X Alignment – Label – Bad Thing – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Center X to: Superview
Select and Edit...
Delete
Stanford CS193p
Fall 2013

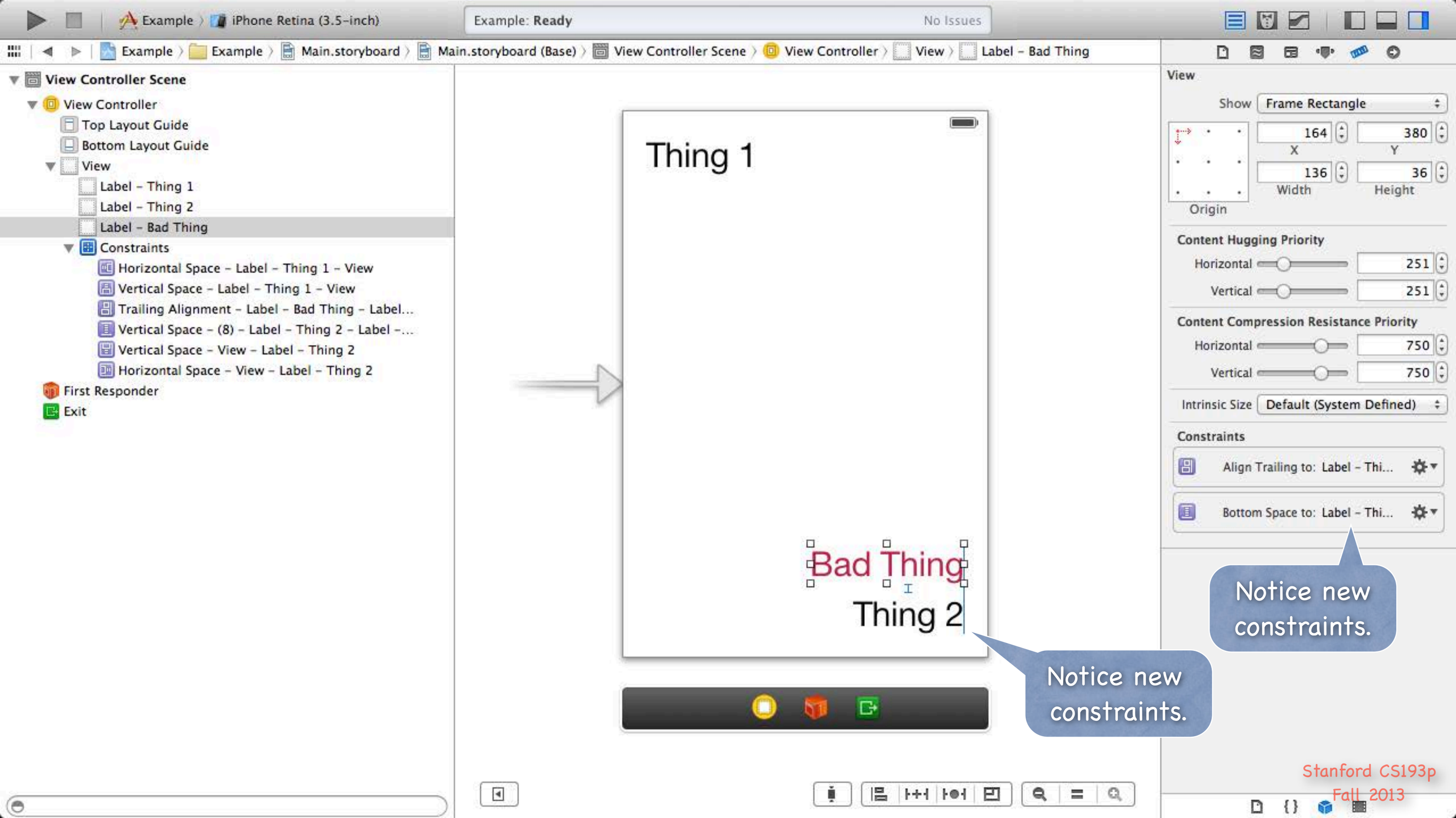

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Now we want to constraint Bad Thing to stay on top of Thing 2.
Bad Thing
Thing 2
Let's do that yet a third way (i.e. not with blue guidelines/ Suggested and not with a menu at the bottom).
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
Stanford CS193p
Fall 2013

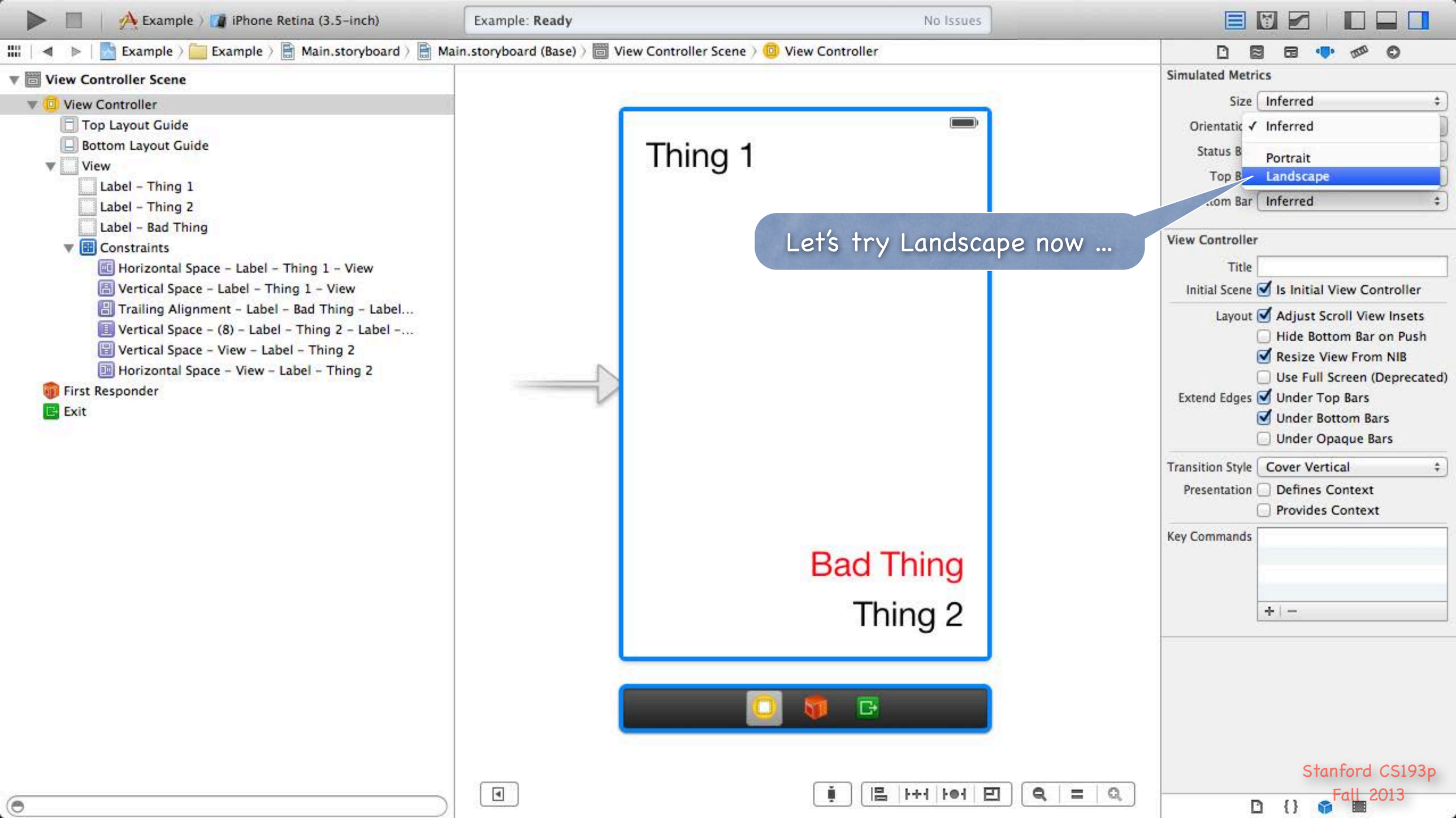

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
Label – Thing 2
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
If you want a view to be constrained by another view's size or position, just ctrl-drag between them.
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing
Vertical Spacing
Left
Center X
Right
Equal Widths
Equal Heights
Hold Shift to select multiple then click away or hit return
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
You will then be asked how you want them constrained.
You can pick multiple ways.
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing
✓ Vertical Spacing
Left
Center X
Right
Equal Widths
Equal Heights
Hold Shift to select multiple then click away or hit return
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
Here we'll keep the two views a fixed distance apart (constrained Vertical Spacing), and ...
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing
✓ Vertical Spacing
Left
Center X
✓ Right
Equal Widths
Equal Heights
Hold Shift to select multiple then click away or hit return
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Intrinsic Size
Default (System Defined)
Constraints
The selected views have no constraints. At build time explicit left, top, width, and height constraints will be generated for the view.
Keep their right edges aligned at all times.
Stanford CS193p
Fall 2013

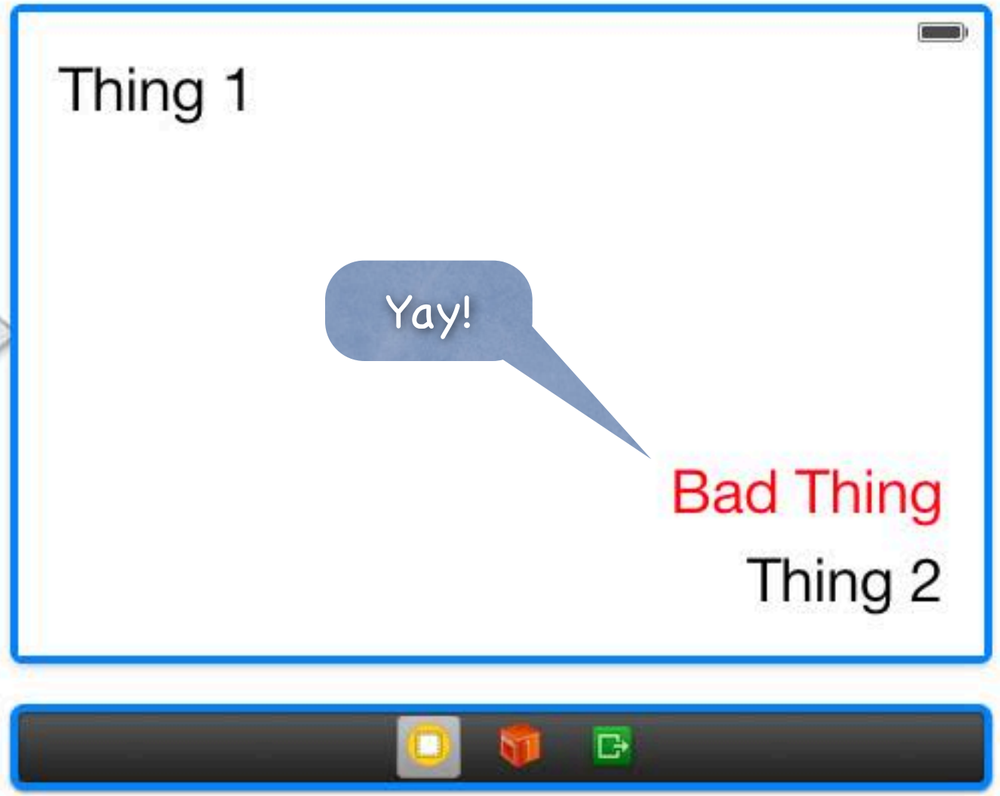

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
Notice new constraints.
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Trailing to: Label – Thi...
Bottom Space to: Label – Thi...
Notice new constraints.
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
Let's try Landscape now ...
Simulated Metrics
Size
Inferred
Orientatic
Inferred
Portrait
Landscape
Status B
Top B
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

Example › iPhone Retina (3.5-inch) | Example: Ready | No Issues

Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller

- View Controller Scene
  - View Controller
    - Top Layout Guide
    - Bottom Layout Guide
    - View
      - Label – Thing 1
      - Label – Thing 2
      - Label – Bad Thing
      - Constraints
        - Horizontal Space – Label – Thing 1 – View
        - Vertical Space – Label – Thing 1 – View
        - Trailing Alignment – Label – Bad Thing – Label…
        - Vertical Space – (8) – Label – Thing 2 – Label –…
        - Vertical Space – View – Label – Thing 2
        - Horizontal Space – View – Label – Thing 2
  - First Responder
  - Exit

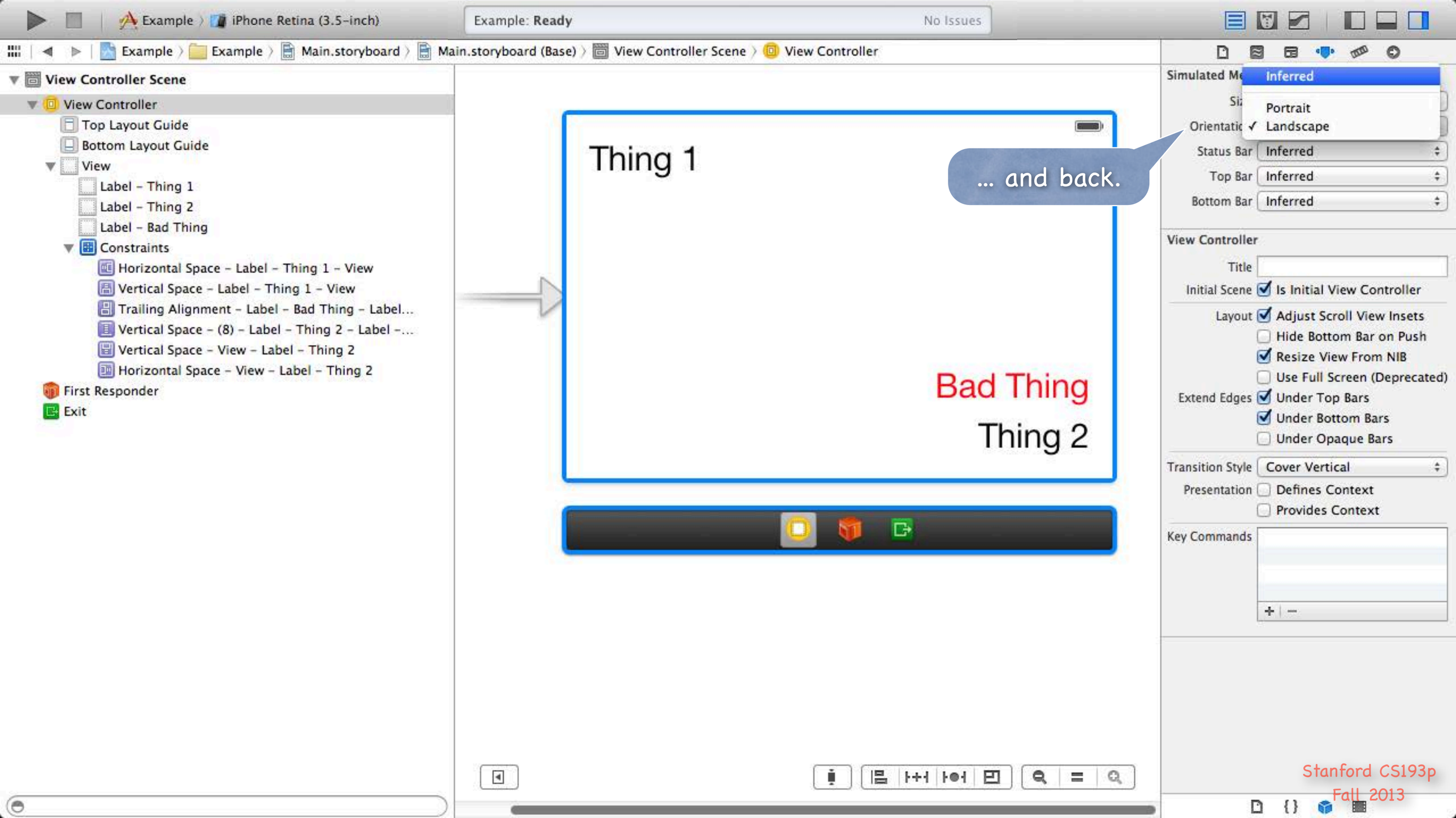


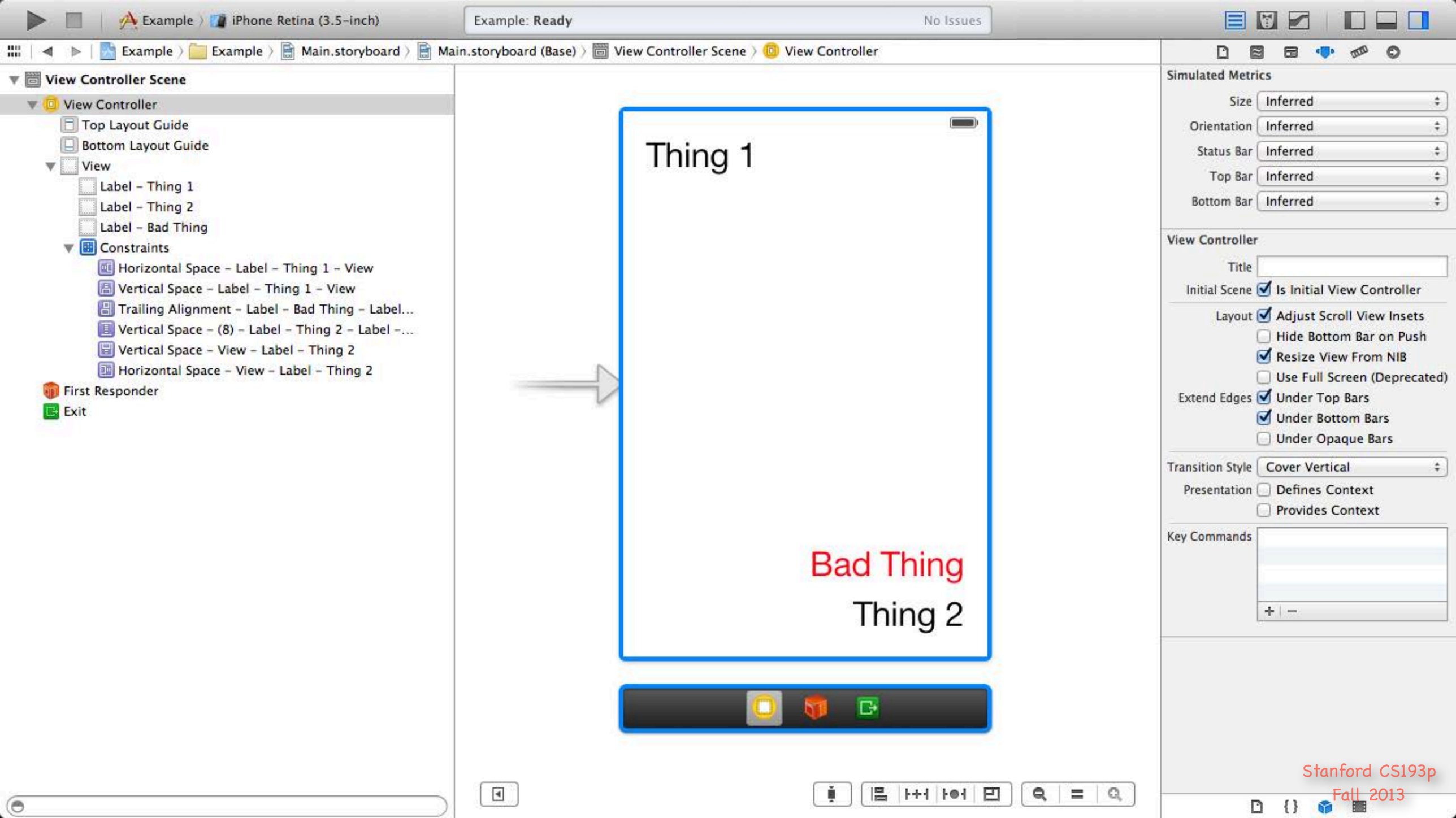
Simulated Metrics

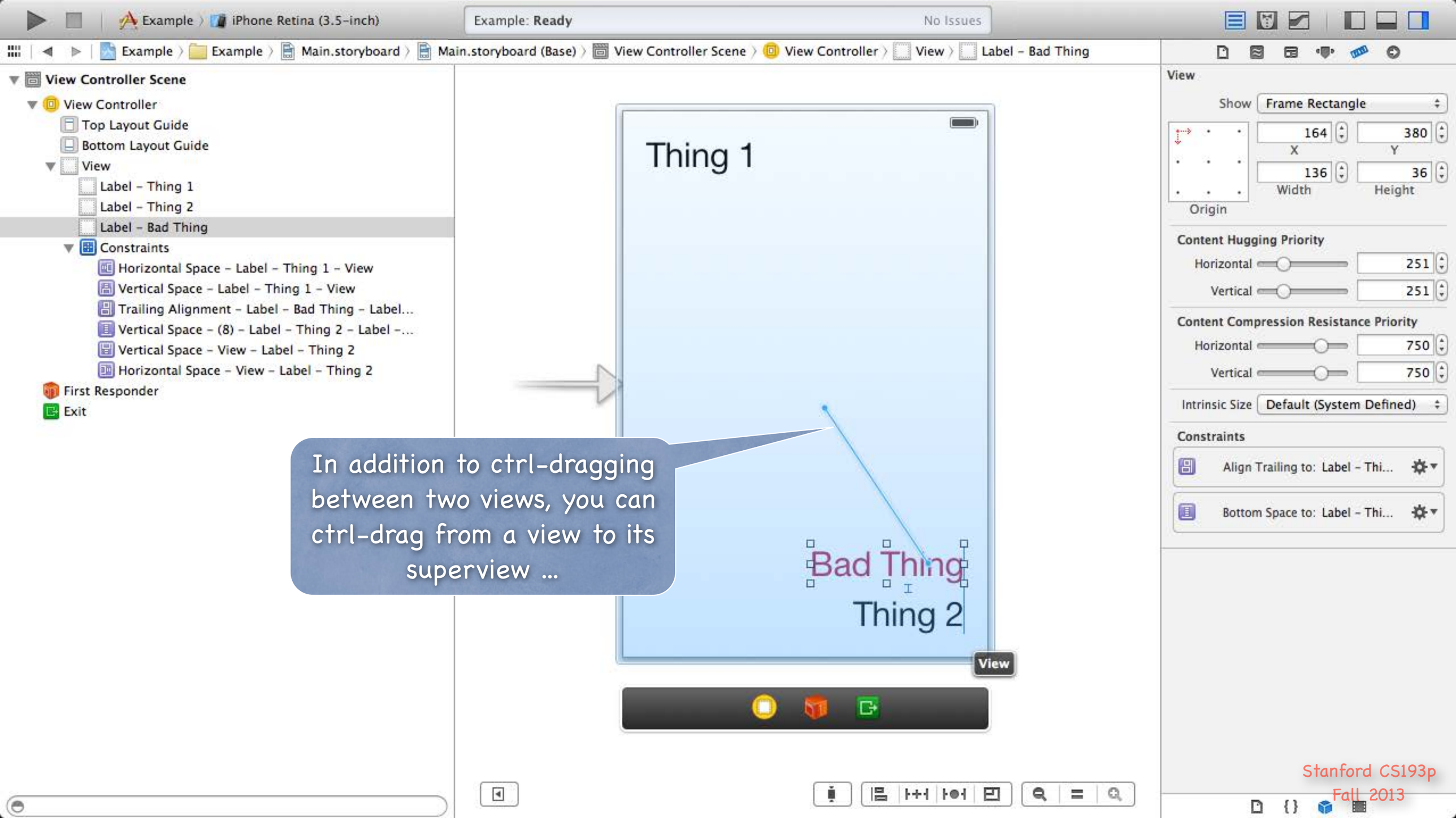
Size	Inferred
Orientation	Landscape
Status Bar	Inferred
Top Bar	Inferred
Bottom Bar	Inferred

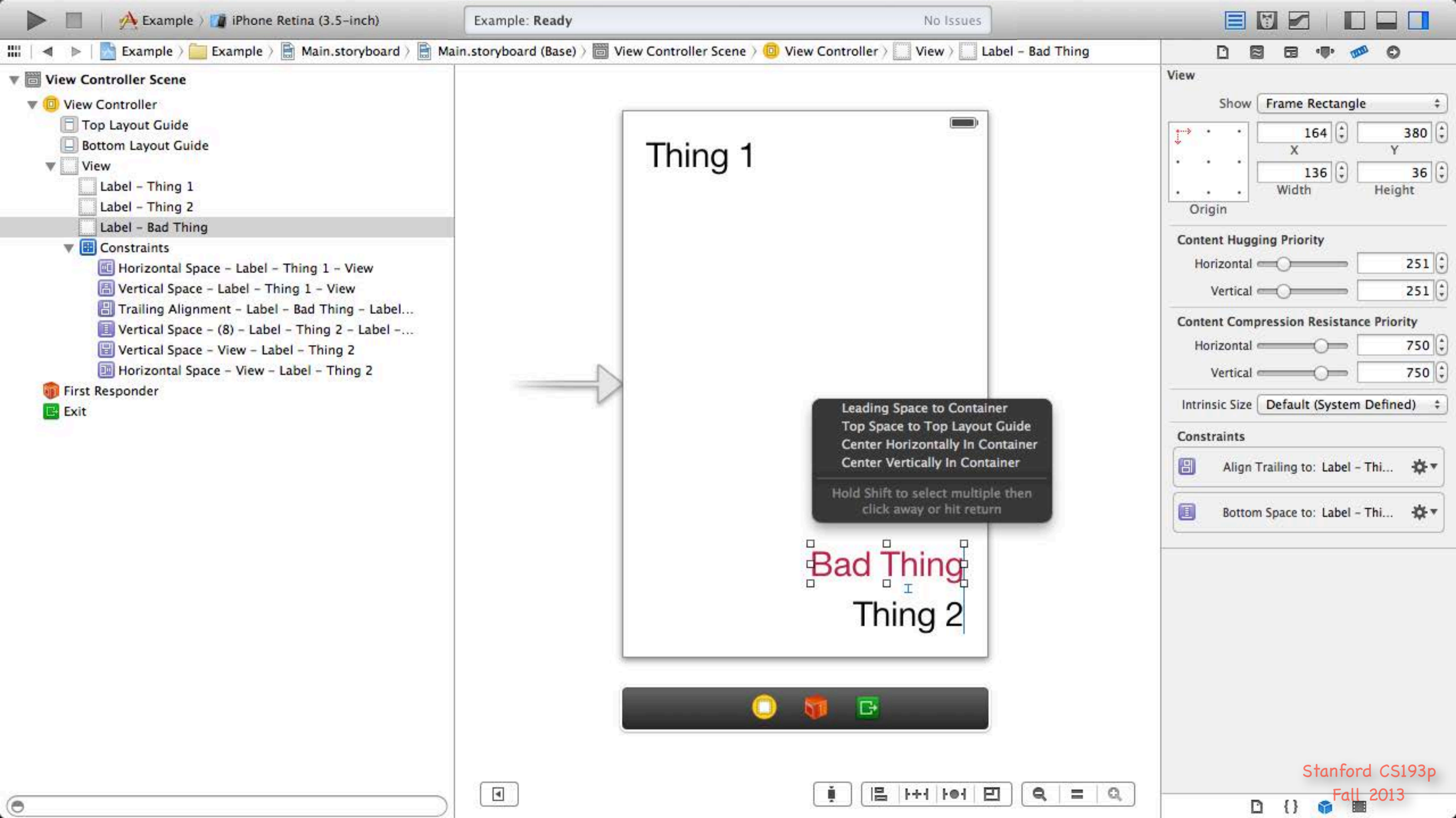
View Controller

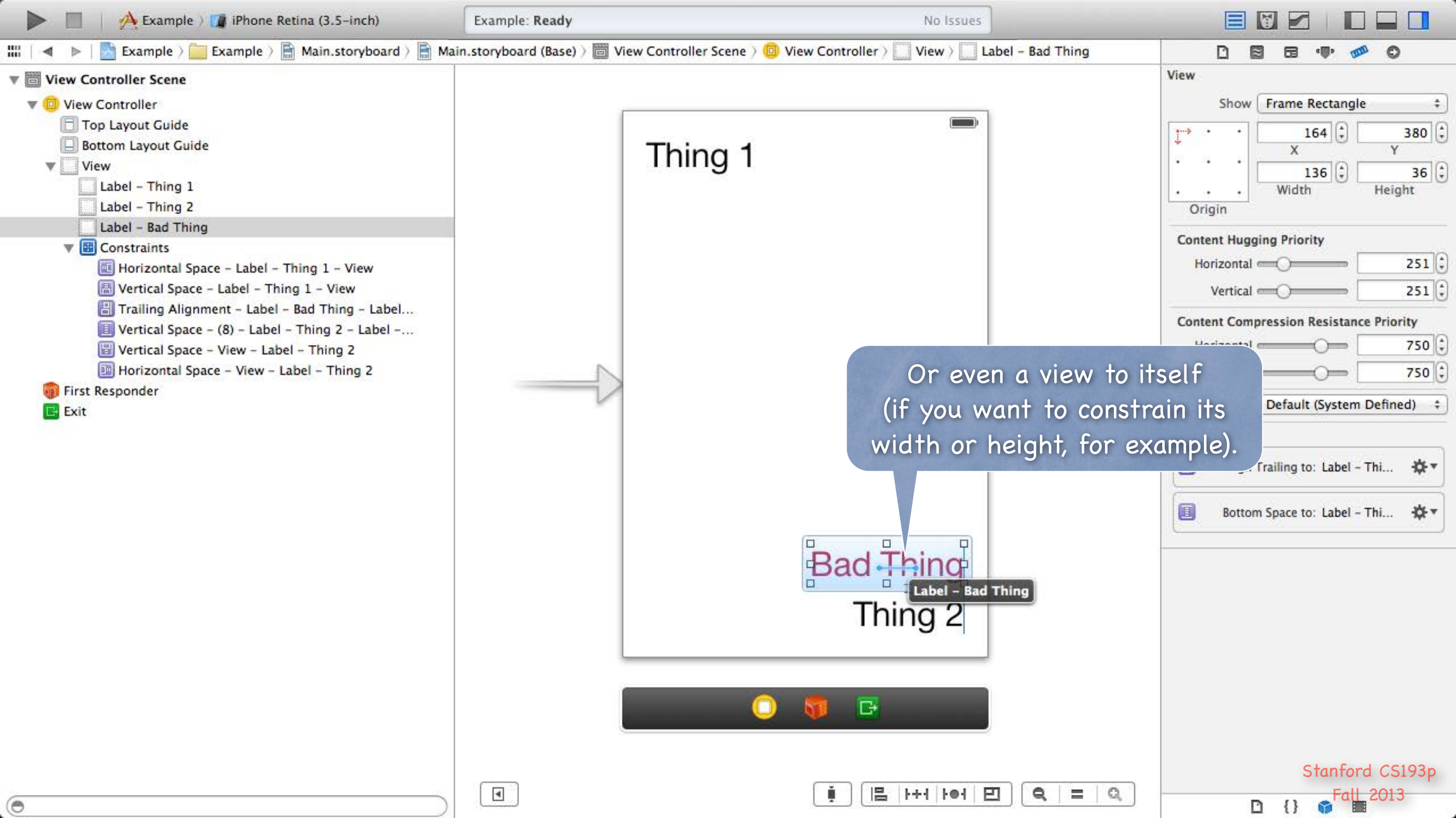
Title	
Initial Scene	☑ Is Initial View Controller
Layout	☑ Adjust Scroll View Insets
	☐ Hide Bottom Bar on Push
	☑ Resize View From NIB
	☐ Use Full Screen (Deprecated)
Extend Edges	☑ Under Top Bars
	☑ Under Bottom Bars
	☐ Under Opaque Bars
Transition Style	Cover Vertical
Presentation	☐ Defines Context
	☐ Provides Context
Key Commands	

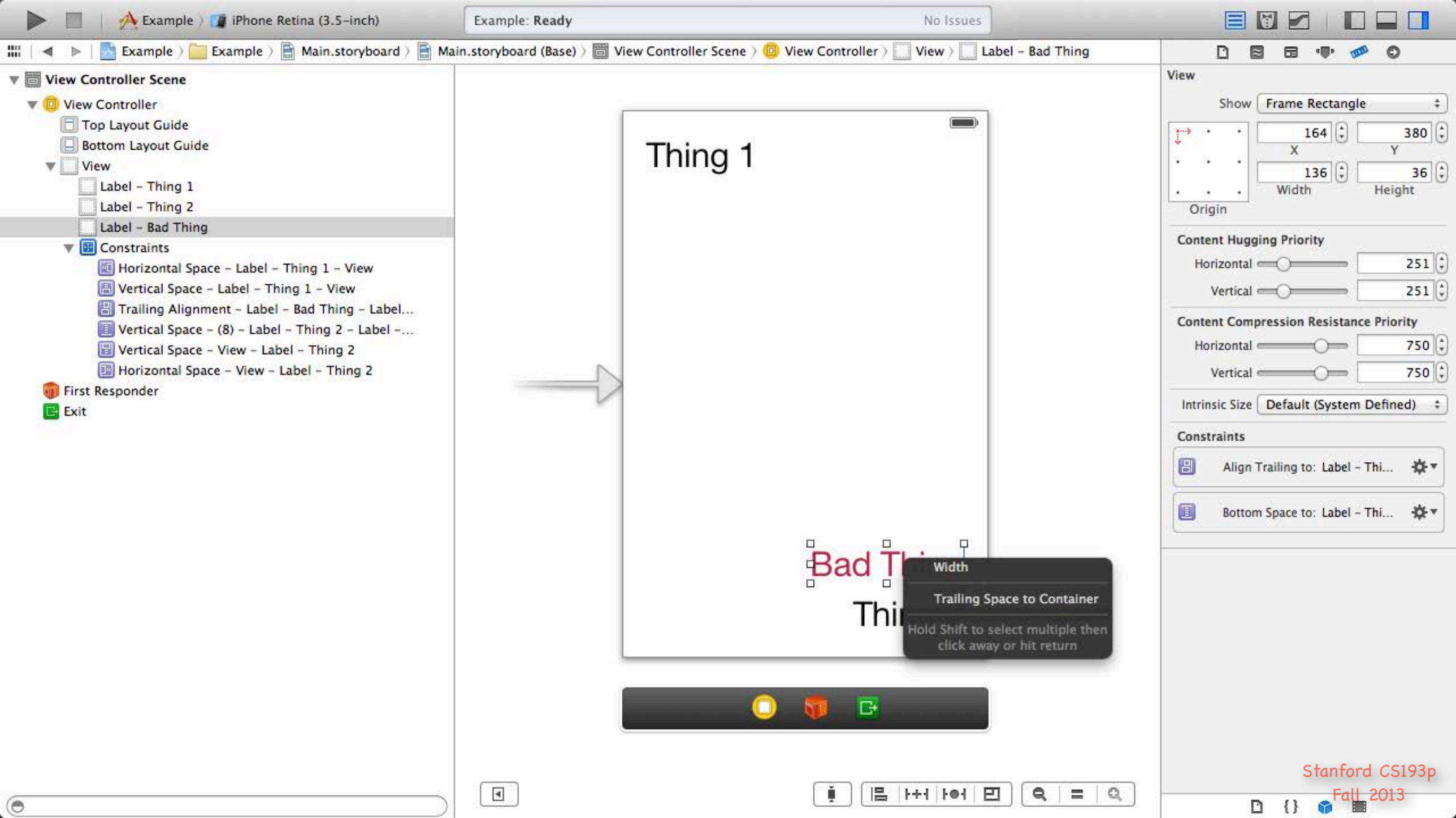

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
... and back.
Bad Thing
Thing 2
Simulated Me
Inferred
Portrait
Landscape
Orientatic
Status Bar
Inferred
Top Bar
Inferred
Bottom Bar
Inferred
View Controller
Title
Initial Scene
Is Initial View Controller
Layout
Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges
Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style
Cover Vertical
Presentation
Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013

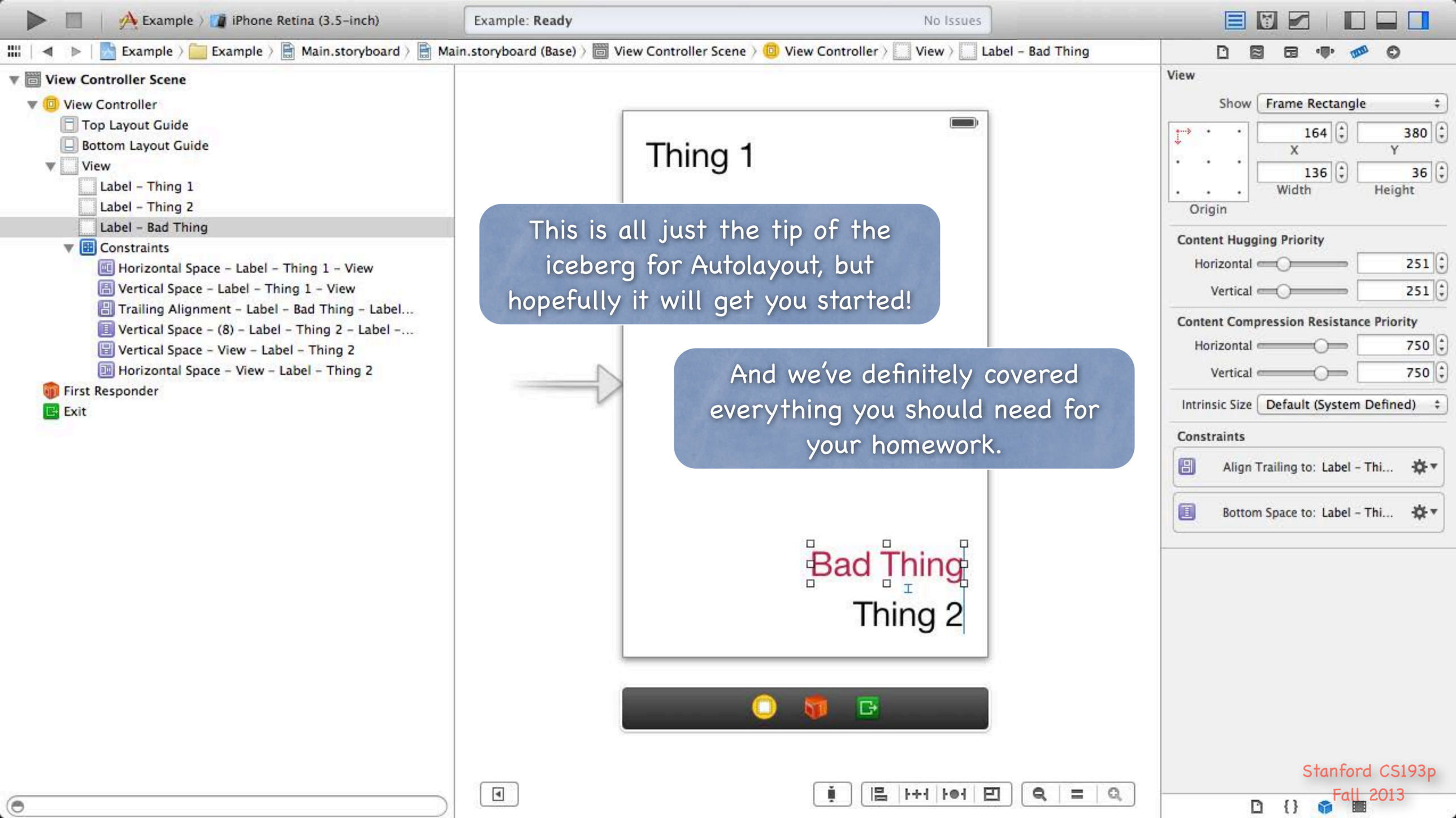

Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Thing 2
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
In addition to ctrl-dragging between two views, you can ctrl-drag from a view to its superview ...
Thing 1
Bad Thing
Thing 2
View
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Trailing to: Label – Thi...
Bottom Space to: Label – Thi...
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Leading Space to Container
Top Space to Top Layout Guide
Center Horizontally In Container
Center Vertically In Container
Hold Shift to select multiple then click away or hit return
Bad Thing
Thing 2
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Trailing to: Label – Thi...
Bottom Space to: Label – Thi...
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Bad Thing
Label – Bad Thing
Thing 2
Or even a view to itself
(if you want to constrain its
width or height, for example).
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
750
750
Default (System Defined)
Trailing to: Label – Thi...
Bottom Space to: Label – Thi...
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
Width
Trailing Space to Container
Hold Shift to select multiple then click away or hit return
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Trailing to: Label – Thi...
Bottom Space to: Label – Thi...
Stanford CS193p
Fall 2013


Example › iPhone Retina (3.5-inch)
Example: Ready
No Issues
Example › Example › Main.storyboard › Main.storyboard (Base) › View Controller Scene › View Controller › View › Label – Bad Thing
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Label – Thing 1
Label – Thing 2
Label – Bad Thing
Constraints
Horizontal Space – Label – Thing 1 – View
Vertical Space – Label – Thing 1 – View
Trailing Alignment – Label – Bad Thing – Label...
Vertical Space – (8) – Label – Thing 2 – Label –...
Vertical Space – View – Label – Thing 2
Horizontal Space – View – Label – Thing 2
First Responder
Exit
Thing 1
This is all just the tip of the iceberg for Autolayout, but hopefully it will get you started!
And we've definitely covered everything you should need for your homework.
Bad Thing
Thing 2
View
Show
Frame Rectangle
164
X
380
Y
136
Width
36
Height
Origin
Content Hugging Priority
Horizontal
251
Vertical
251
Content Compression Resistance Priority
Horizontal
750
Vertical
750
Intrinsic Size
Default (System Defined)
Constraints
Align Trailing to: Label – Thi...
Bottom Space to: Label – Thi...
Stanford CS193p
Fall 2013

Demo

- Attributor Autorotation

  Since we dragged to blue guidelines, it's mostly going to be automatic.
  But there are a couple of things to fix up.
  And we'll make a couple of changes too.

Coming Up

- **Friday**
  Still hoping to get University Developer Program up and running!
- **Homework**
  Due on Monday
- **Next Week**
  Scroll View
  Table View
  Collection View

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

- Multithreading
  Posting blocks on queues (which are then executed on other threads).
- `UIScrollView`
  A "window" on an arbitrarily large content area that can be moved and zoomed.
- Demo
  Imaginarium
- `UITableView`
  (Time Permitting)
  Data source-driven vertical list of views.

Multithreading

- Queues
  - Multithreading is mostly about "queues" in iOS.
  - Blocks are lined up in a queue (method calls can also be enqueued).
  - Then those blocks are pulled off the queue and executed on an associated thread.
- Main Queue
  - There is a very special queue called the "main queue."
  - All UI activity MUST occur on this queue and this queue only.
  - And, conversely, non-UI activity that is at all time consuming must NOT occur on that queue.
  - We want our UI to be responsive!
  - Blocks are pulled off and worked on in the main queue only when it is "quiet".
- Other Queues
  - Mostly iOS will create these for us as needed.
  - We'll give a quick overview of how to create your own (but usually not necessary).

Multithreading

- Executing a block on another queue

```
dispatch_queue_t queue = …;
dispatch_async(queue, ^{ });
```

- Getting the main queue

```
dispatch_queue_t mainQ = dispatch_get_main_queue();
NSOperationQueue *mainQ = [NSOperationQueue mainQueue]; // for object-oriented APIs
```

- Creating a queue (not the main queue)

```
dispatch_queue_t otherQ = dispatch_queue_create("name", NULL); // name a const char *!
```

- Easy mode … invoking a method on the main queue

```
NSObject method …
- (void)performSelectorOnMainThread:(SEL)aMethod
                         withObject:(id)obj
                      waitUntilDone:(BOOL)waitUntilDone;
dispatch_async(dispatch_get_main_queue(), ^{ /* call aMethod */ });
```

Multithreading

- Example of an iOS API that uses multithreading

```
NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL urlWithString:@“http://...”]];
NSURLConfiguration *configuration = …;
NSURLSession *session = …;
NSURLSessionDownloadTask *task;
task = [session downloadTaskWithRequest:request
                      completionHandler:^(NSURL *localfile, NSURLResponse *response, NSError *error) {
   /* want to do UI things here, can I? */
}];
[task resume];
```

`downloadTaskWithRequest:completionHandler:` will do its work (downloading that URL) NOT in the main thread (i.e. it will not block the UI while it waits on the network).

The `completionHandler` block above might execute on the main thread (or not) depending on how you create the `NSURLSession`.

Let's look at the two options (on or off the main queue) …

Multithreading

- On the main queue …

```
NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration
                                                      delegate:nil
                                                 delegateQueue:[NSOperationQueue mainQueue]];
NSURLSessionDownloadTask *task;
task = [session downloadTaskWithRequest:request
                      completionHandler:^(NSURL *localfile, NSURLResponse *response, NSErr or *error) {
    /* yes, can do UI things directly because this is called on the main queue */
}];
[task resume];
```

Since the `delegateQueue` is the main queue, our `completionHandler` will be on the main queue.
When the URL is done downloading, the block above will execute on the main queue.
Thus we can do any UI code we want there.
Of course, if you are doing non-UI things here, they'd best be quick (don't block main queue!).

Multithreading

- Off the main queue …

```
NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration]; // no delegateQueue
NSURLSessionDownloadTask *task;
task = [session downloadTaskWithRequest:request
                      completionHandler:^(NSURL *localfile, NSURLResponse *response, NSError *error) {
    dispatch_async(dispatch_get_main_queue(), ^{ /* do UI things */ });
    or [self performSelectorOnMainThread:@selector(doUIthings) withObject:nil waitUntilDone:NO];
}];
[task resume];
```

In this case, you can't do any UI stuff because the completionHandler is not on the main queue. To do UI stuff, you have to post a block (or call a method) back on the main queue (as shown).

UIScrollView

Adding subviews to a normal UIView …

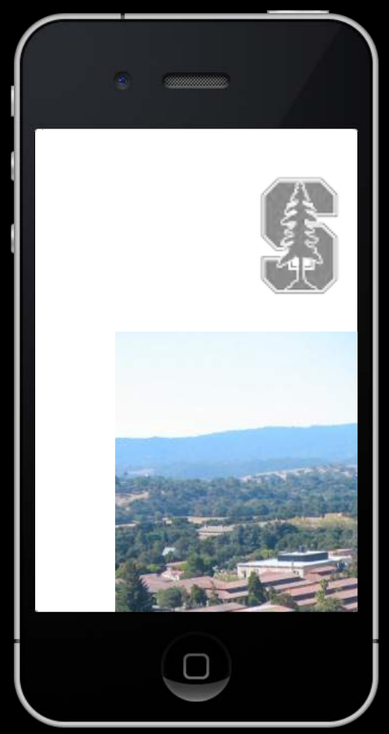
```
subview.frame = ...;
[view addSubview:subview];
```

Adding subviews to a normal UIView …

```
subview.frame = ...;
[view addSubview:subview];
```

Adding subviews to a normal UIView …

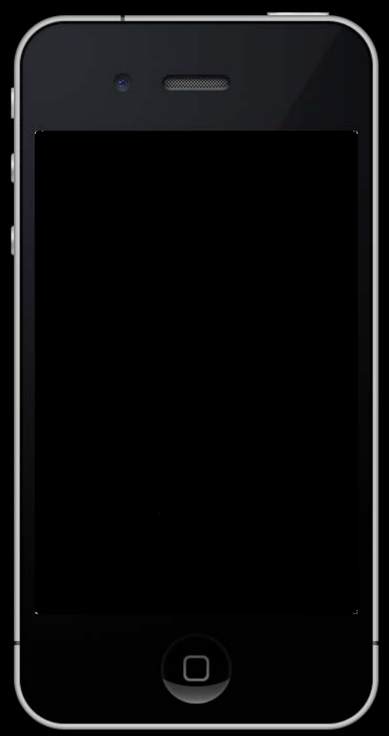
```
subview.frame = ...;
[view addSubview:subview];
```

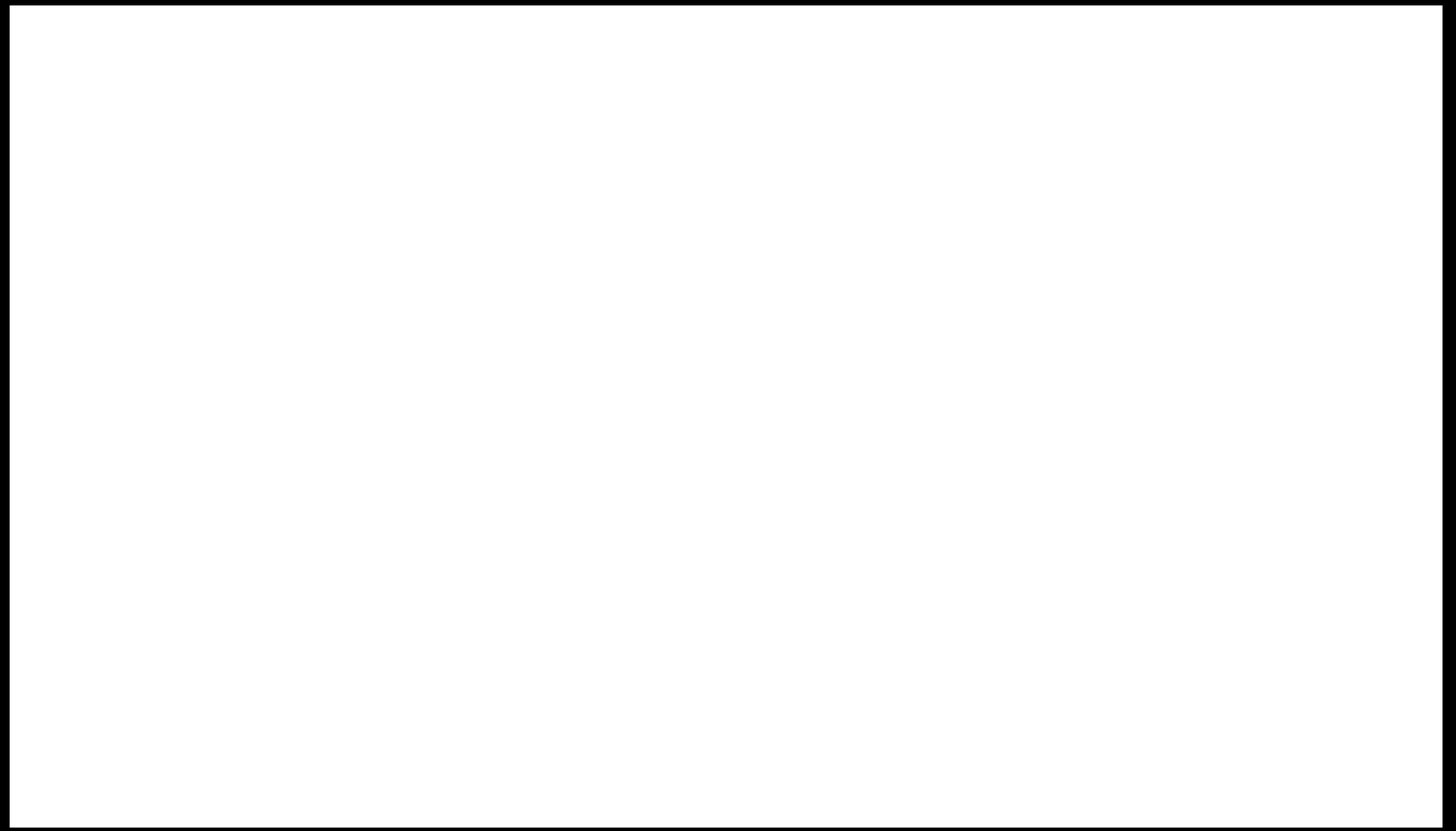


Adding subviews to a UIScrollView …

Adding subviews to a UIScrollView …

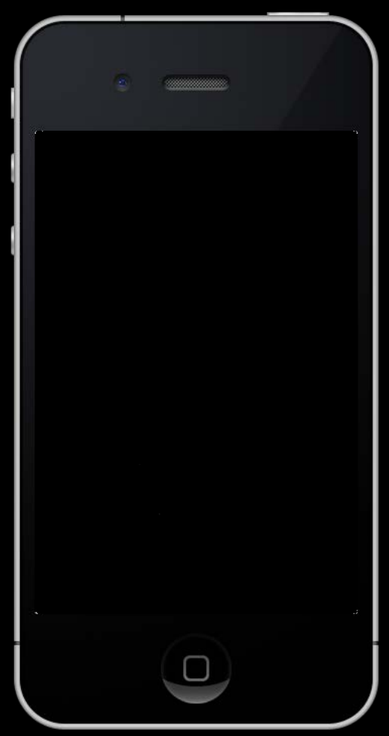
```
scrollView.contentSize = CGSizeMake(3000, 2000);
```

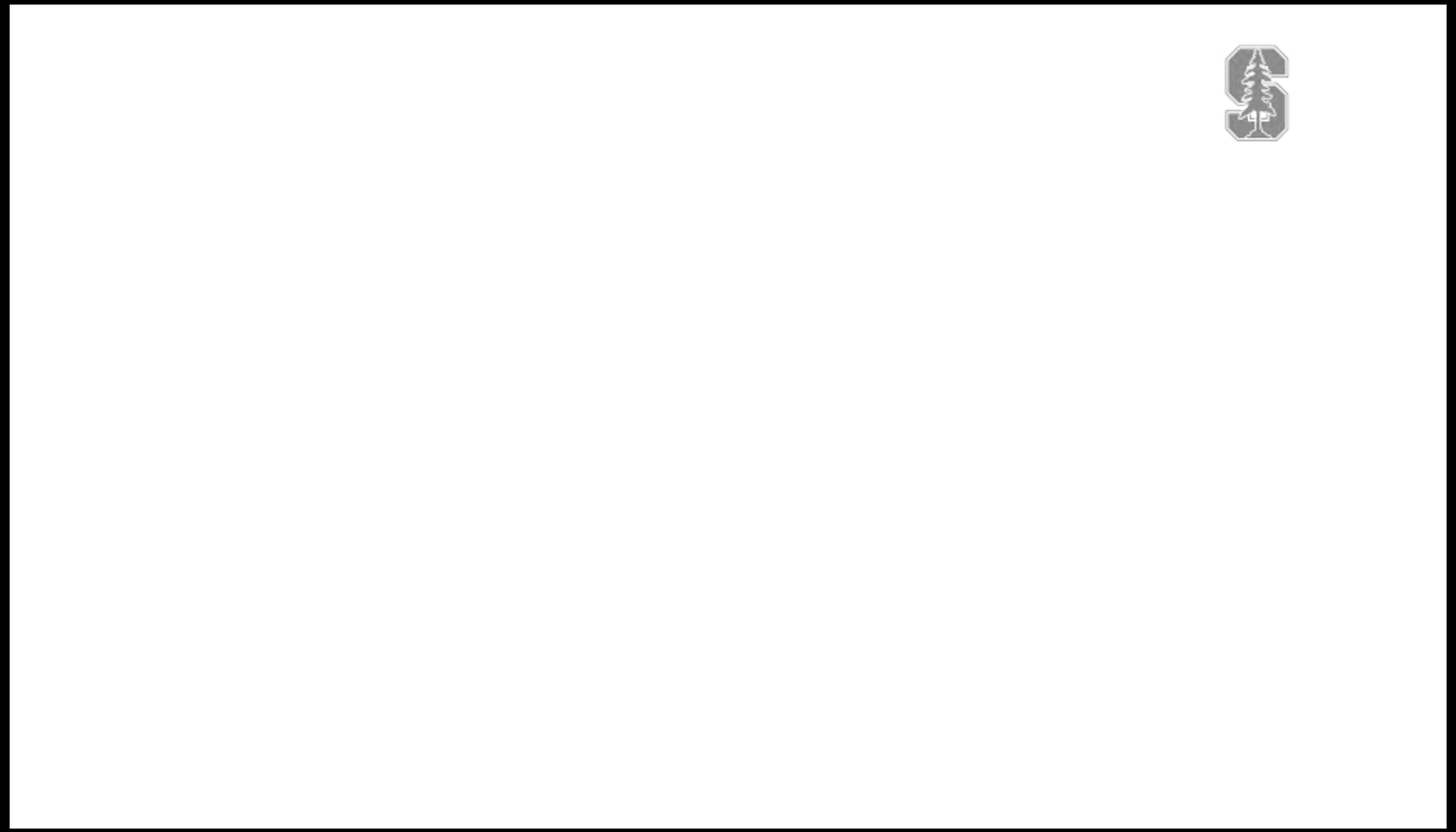




Adding subviews to a UIScrollView …

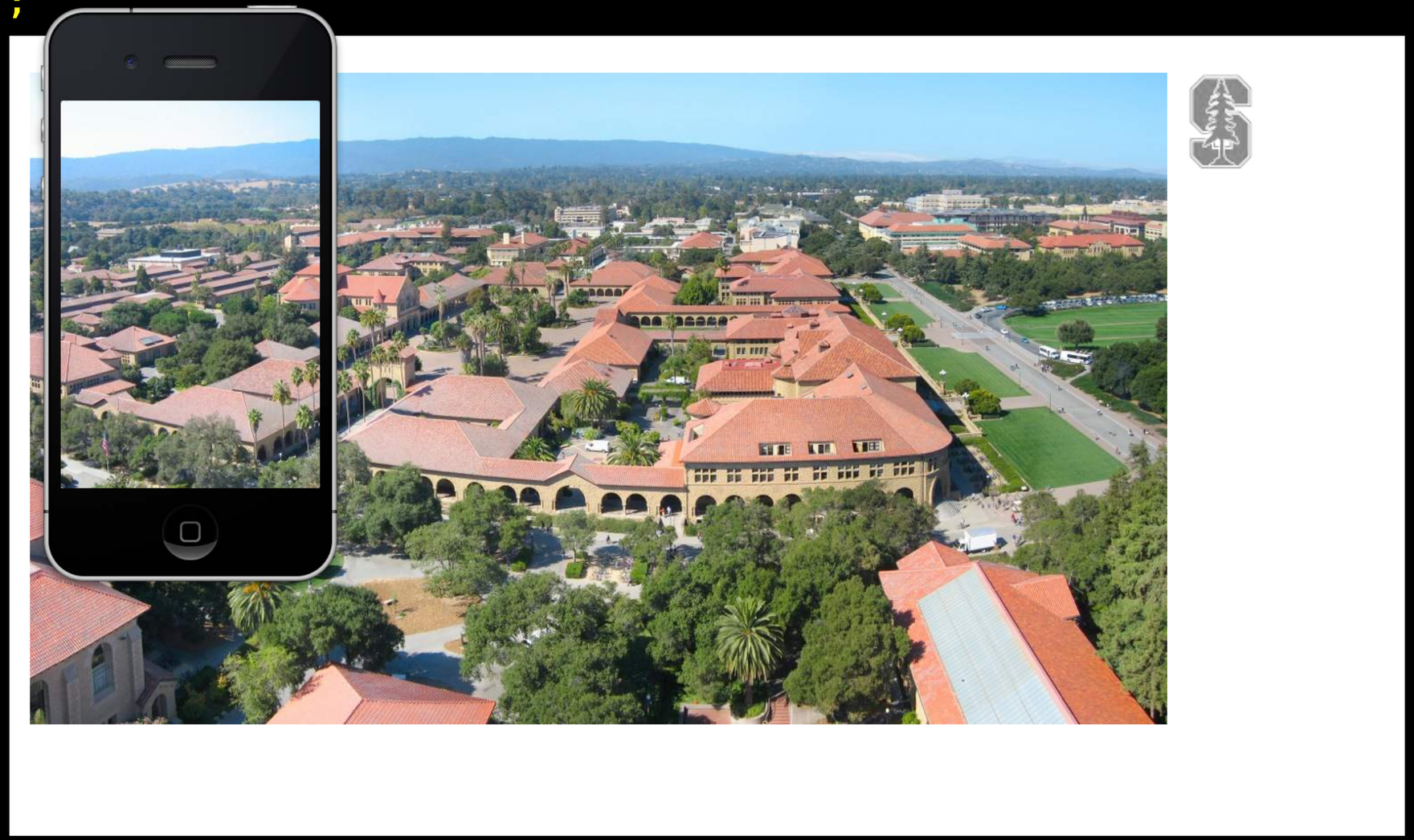
```
scrollView.contentSize = CGSizeMake(3000, 2000);
subview1.frame = CGRectMake(2700, 100, 120, 180);
[view addSubview:subview1];
```



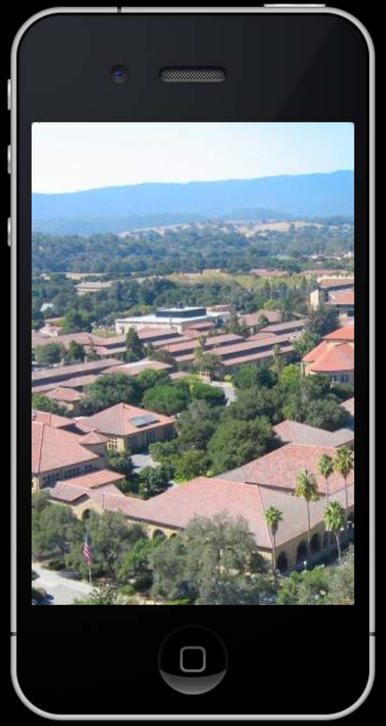


Adding subviews to a UIScrollView …

```
scrollView.contentSize = CGSizeMake(3000, 2000);
subview2.frame = CGRectMake(50, 100, 2500, 1600);
[view addSubview:subview2];
```



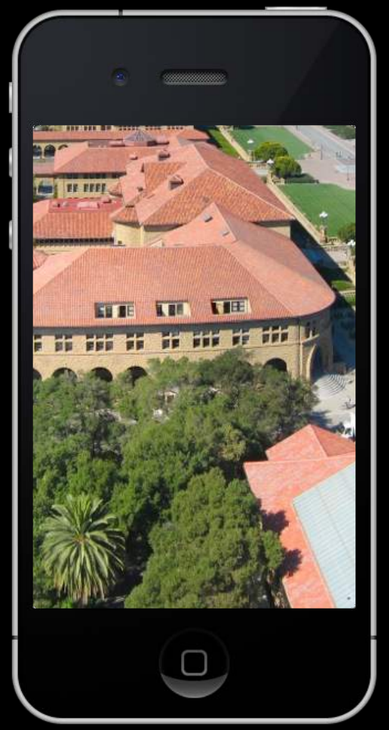
Adding subviews to a UIScrollView …



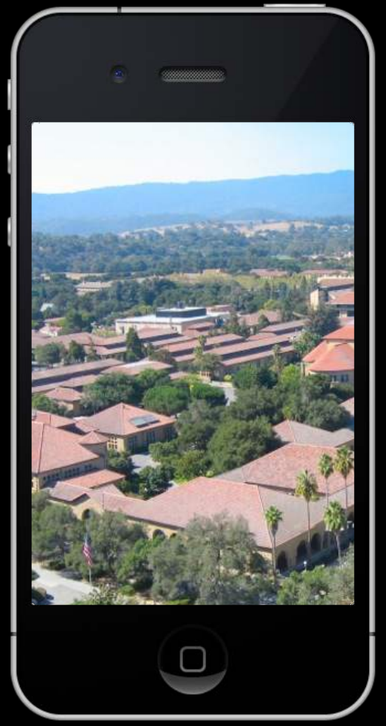
Adding subviews to a UIScrollView …

Adding subviews to a UIScrollView …

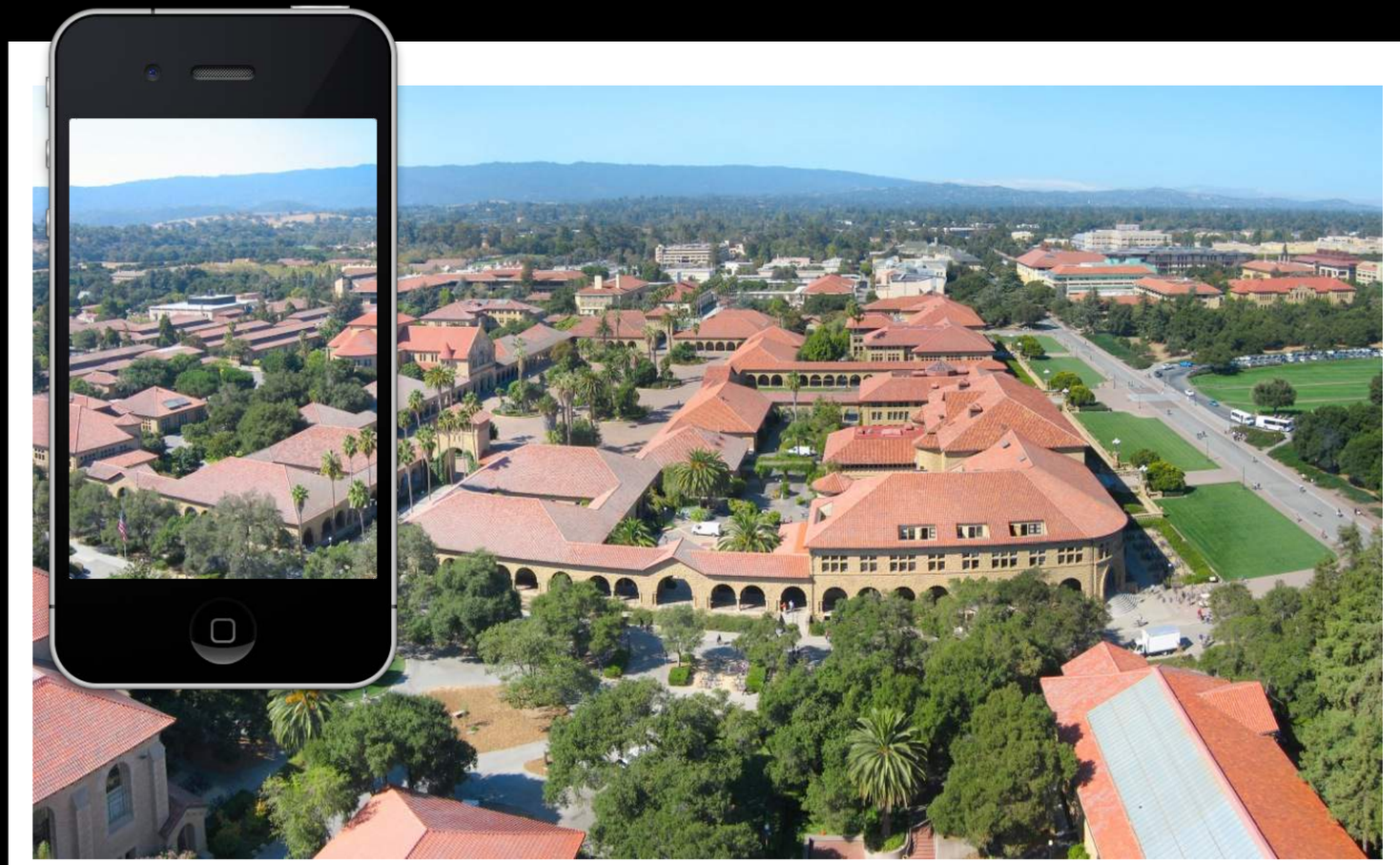
Adding subviews to a UIScrollView …



Adding subviews to a UIScrollView …

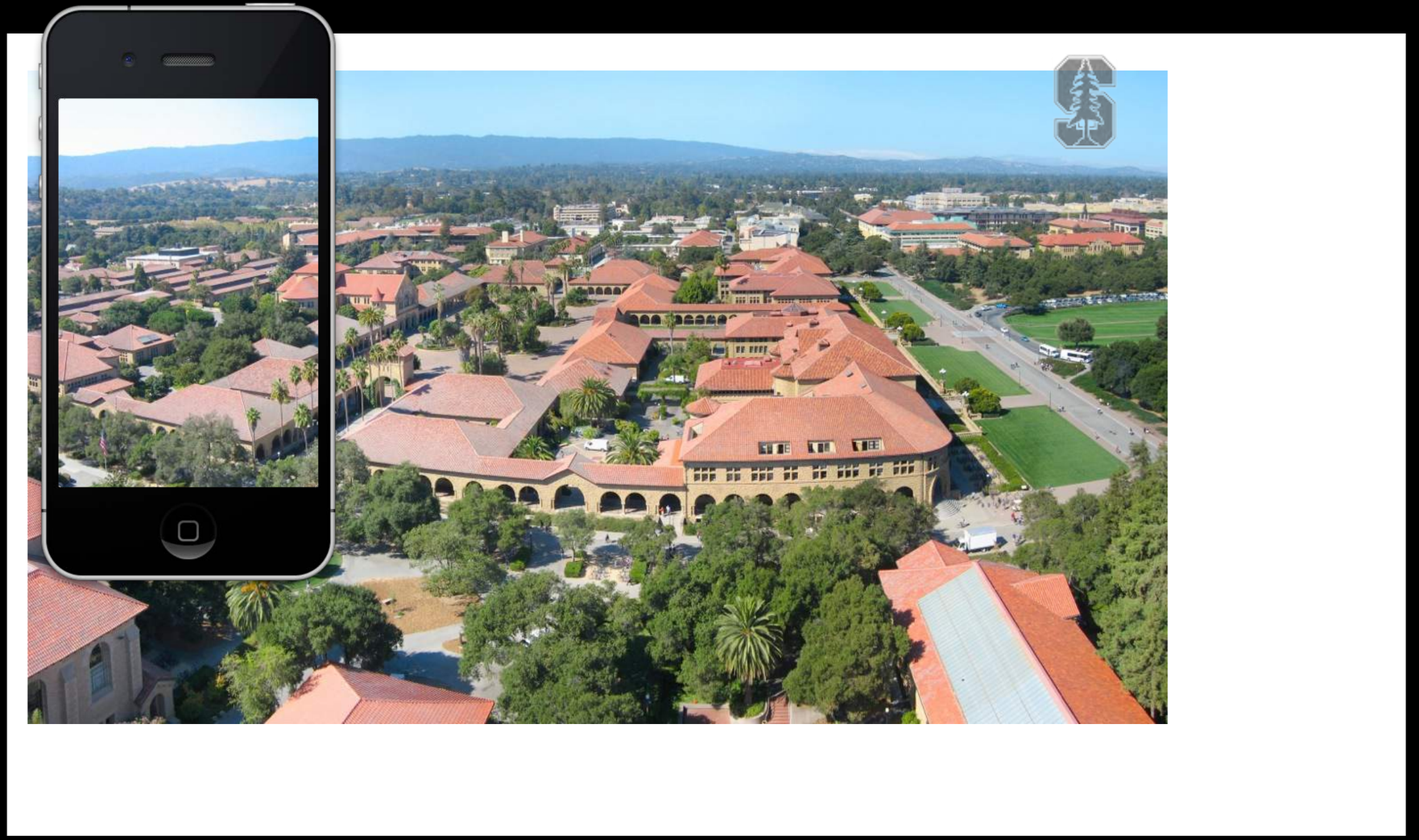


Positioning subviews in a UIScrollView …



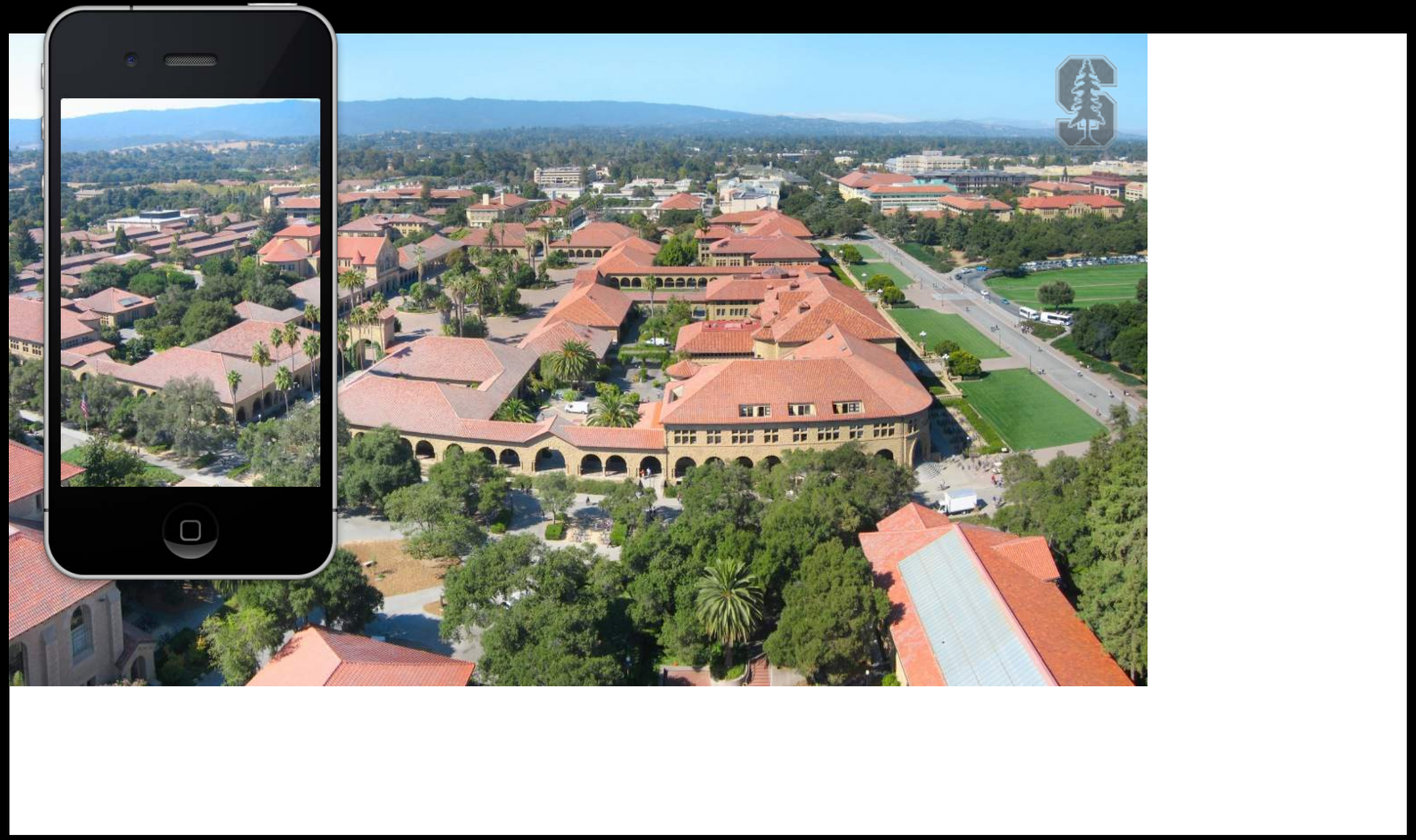
Positioning subviews in a UIScrollView …

```
subview1.frame = CGRectMake(2250, 50, 120, 180);
```



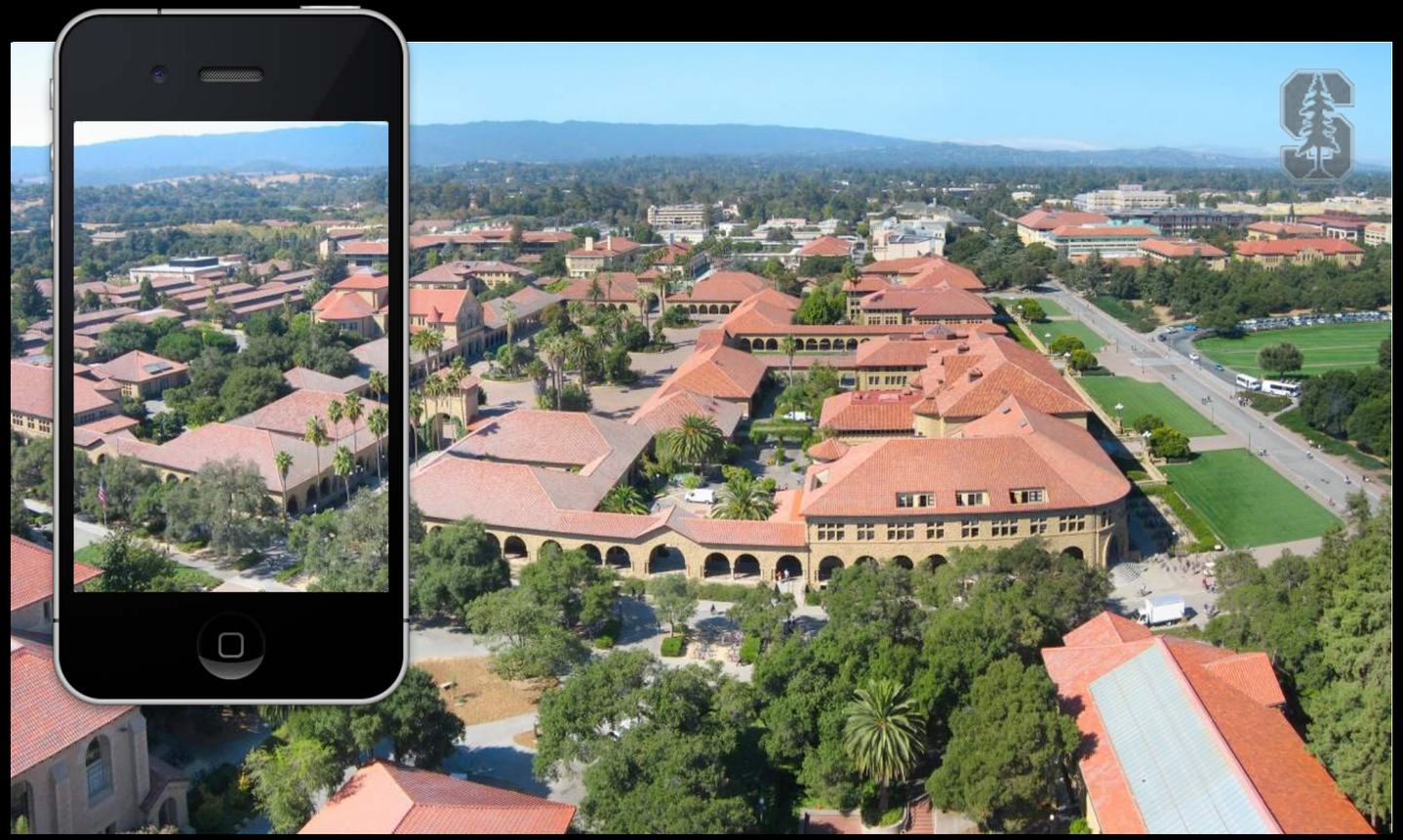
Positioning subviews in a UIScrollView …

```
subview2.frame = CGRectMake(0, 0, 2500, 1600);
```

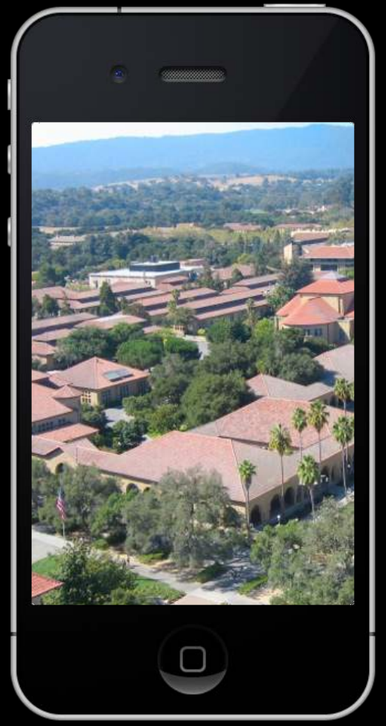


Positioning subviews in a UIScrollView …

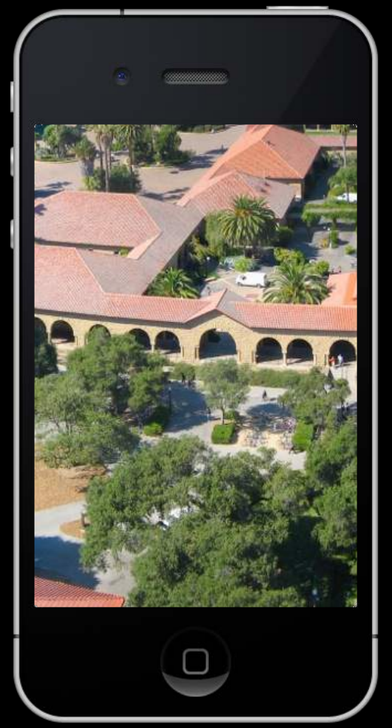
```
subview2.frame = CGRectMake(0, 0, 2500, 1600);
scrollView.contentSize = CGSizeMake(2500, 1600);
```



Voilà!

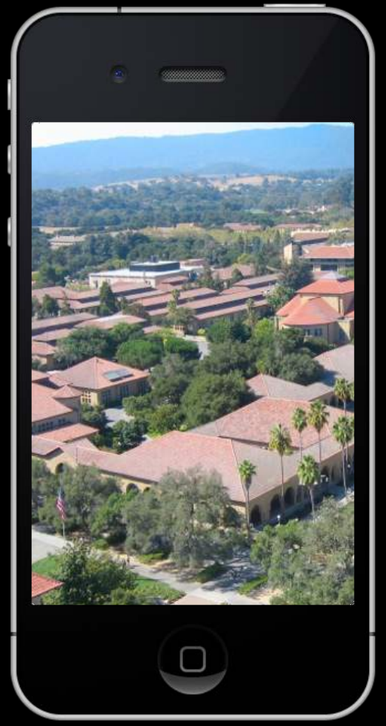


Voilà!



Voilà!

Voilà!



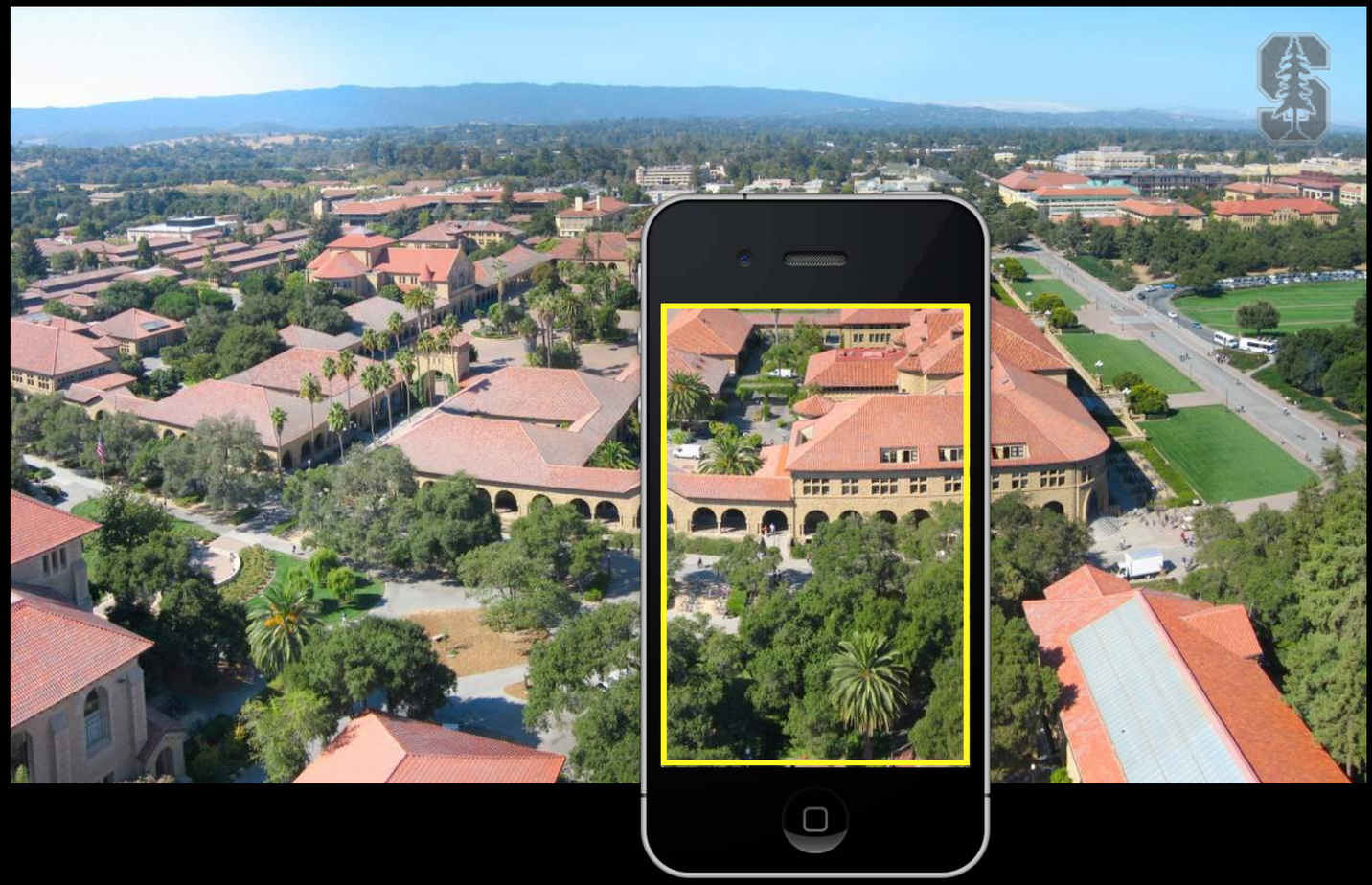
Upper left corner of currently-showing area

```
CGPoint upperLeftOfVisible = scrollView.contentOffset;
```

In content area's coordinates.

Visible area of a scroll view

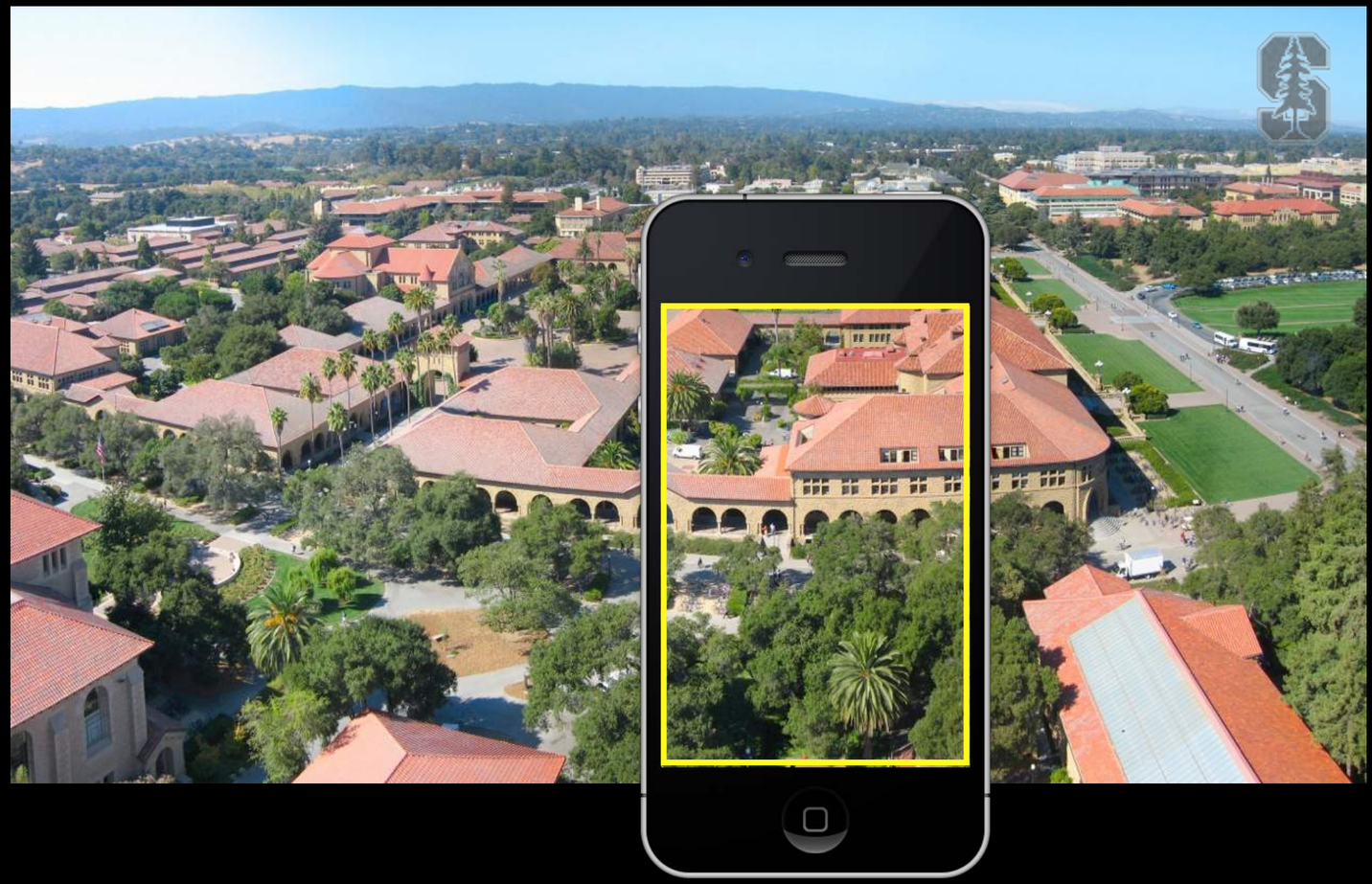
`scrollView.bounds`



Visible area of a scroll view's subview in that view's coordinates

```
CGRect visibleRect = [scrollView convertRect:scrollView.bounds toView:subview];
```

What's the difference? Might be scaled (due to zooming), for example.



UIScrollView

- How do you create one?
  - Just like any other UIView. Drag out in a storyboard or use alloc/initWithFrame:.
  - Or select a UIView in your storyboard and choose "Embed In -> Scroll View" from Editor menu.
- Or add your "too big" UIView using addSubview:

```
UIImage *image = [UIImage imageNamed:@"bigimage.jpg"];
UIImageView *iv = [[UIImageView alloc] initWithImage:image]; // frame.size = image.size
[scrollView addSubview:iv];
```

  - Add more subviews if you want.
  - All of the subviews' frames will be in the UIScrollView's content area's coordinate system (that is, (0,0) in the upper left & width and height of contentSize.width & .height).
- Don't forget to set the contentSize
  - Common bug is to do the above 3 lines of code (or embed in Xcode) and forget to say:

```
scrollView.contentSize = imageView.bounds.size
```

UIScrollView

- Scrolling programmatically
  - `- (void)scrollRectToVisible:(CGRect)aRect animated:(BOOL)animated;`
- Other things you can control in a scroll view

  Whether scrolling is enabled.

  Locking scroll direction to user's first "move".

  The style of the scroll indicators (call `flashScrollIndicators` when your scroll view appears).

  Whether the actual content is "inset" from the content area (`contentInset` property).

UIScrollView

- Zooming

  All UIView's have a property (transform) which is an affine transform (translate, scale, rotate).

  Scroll view simply modifies this transform when you zoom.

  Zooming is also going to affect the scroll view's contentSize and contentOffset.

- Will not work without minimum/maximum zoom scale being set

```
scrollView.minimumZoomScale = 0.5;   // 0.5 means half its normal size
scrollView.maximumZoomScale = 2.0;   // 2.0 means twice its normal size
```

- Will not work without delegate method to specify view to zoom

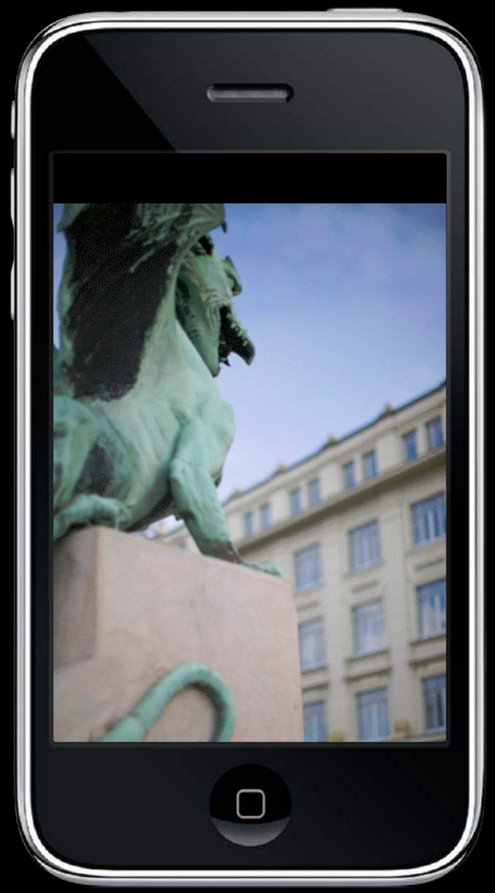
```
- (UIView *)viewForZoomingInScrollView:(UIScrollView *)sender;
```

  If your scroll view only has one subview, you return it here. More than one? Up to you.

- Zooming programatically

```
@property (nonatomic) float zoomScale;
- (void)setZoomScale:(float)scale animated:(BOOL)animated;
- (void)zoomToRect:(CGRect)zoomRect animated:(BOOL)animated;
```

```
scrollView.zoomScale = 1.2;
```



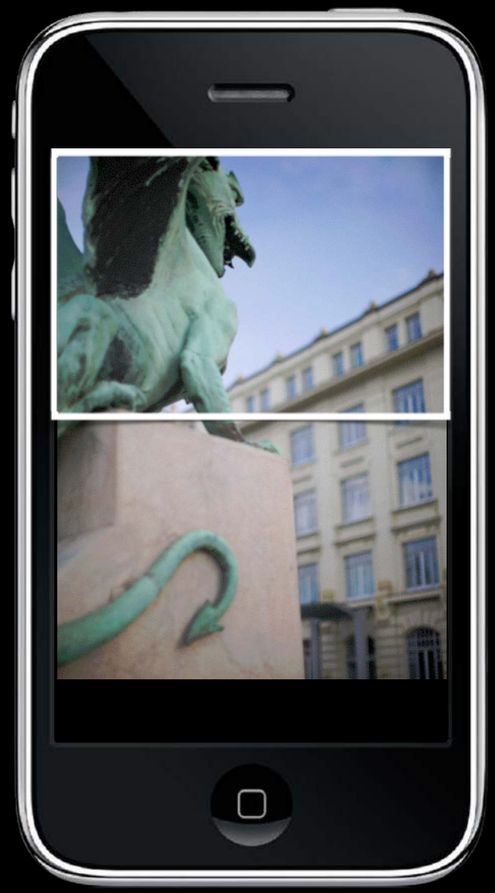
```
scrollView.zoomScale = 1.0;
```

```
scrollView.zoomScale = 1.2;
```

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;
```

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;
```

```
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;
```



```
- (void)zoomToRect:(CGRect)rect animated:(BOOL)animated;
```

UIScrollView

- Lots and lots of `delegate` methods!

  The scroll view will keep you up to date with what's going on.

- Example: `delegate` method will notify you when zooming ends

```
- (void)scrollViewDidEndZooming:(UIScrollView *)sender
                       withView:(UIView *)zoomView  // from delegate method above
                        atScale:(CGFloat)scale;
```

  If you redraw your view at the new scale, be sure to reset the transform back to identity.

Demo

- Imaginarium
  - `UIImageView` inside a `UIScrollView`
  - Multithreaded download from a URL
  - `UIActivityIndicatorView` to show user that a download is in progress

Coming Up

- Wednesday
  - More `UITableView` (with demo)
  - iPad
- Homework
  - Next Homework will be assigned on Wednesday, due the next Wednesday.
- Friday
  - Stanford Only Review Section
- Next Week
  - Core Data (Object-Oriented Database)

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

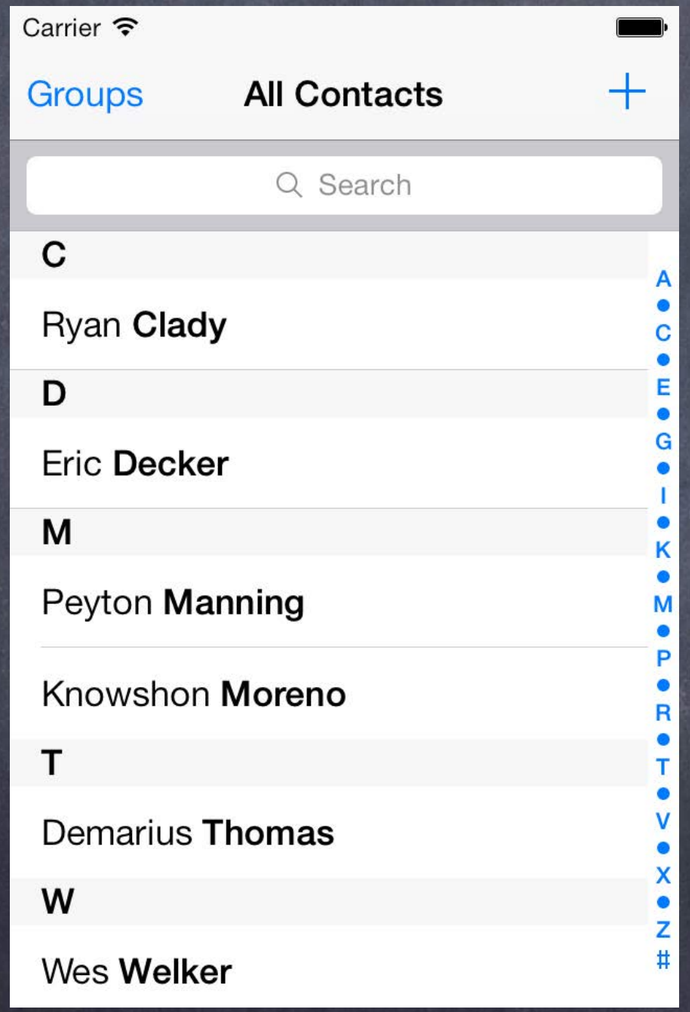
- `UITableView`
  Data source-driven vertical list of views.
- iPad
  Device-specific UI idioms.
- Demo
  Shutterbug

UITableView

- Very important class for displaying data in a table
  - One-dimensional table.
  - It's a subclass of `UIScrollView`.
  - Table can be static or dynamic (i.e. a list of items).
  - Lots and lots of customization via a `dataSource` protocol and a `delegate` protocol.
  - Very efficient even with very large sets of data.
- Displaying multi-dimensional tables ...
  - Usually done via a `UINavigationController` with multiple MVC's where View is `UITableView`
- Kinds of `UITableViews`
  - Plain or Grouped
  - Static or Dynamic
  - Divided into sections or not
  - Different formats for each row in the table (including completely customized)

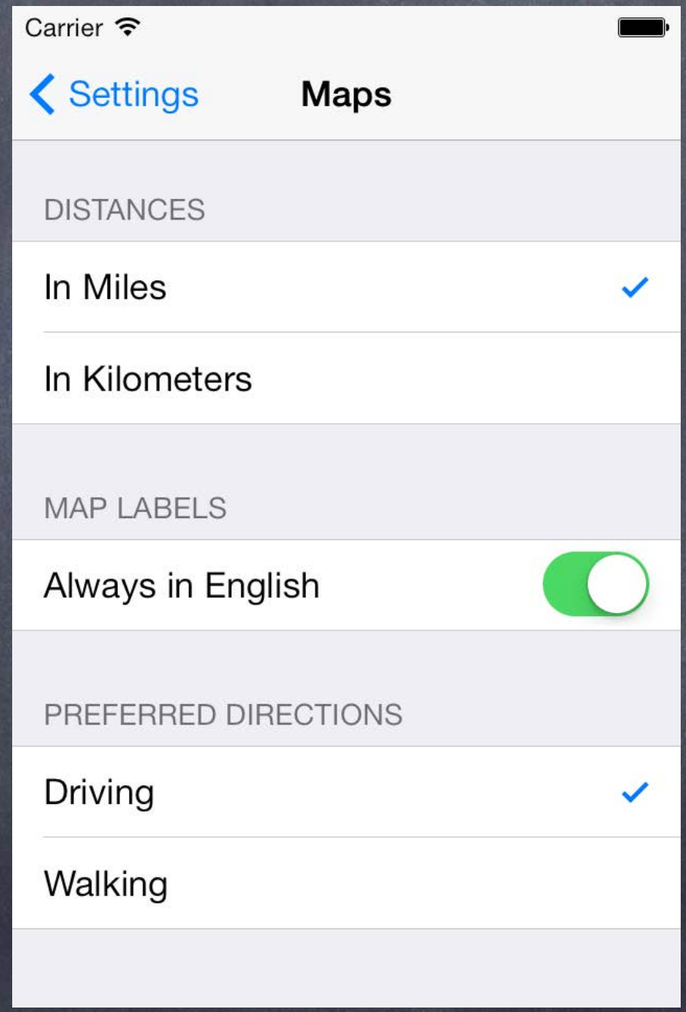
UITableView

UITableViewStylePlain



Dynamic (List) & Plain (ungrouped)

UITableViewStyleGrouped

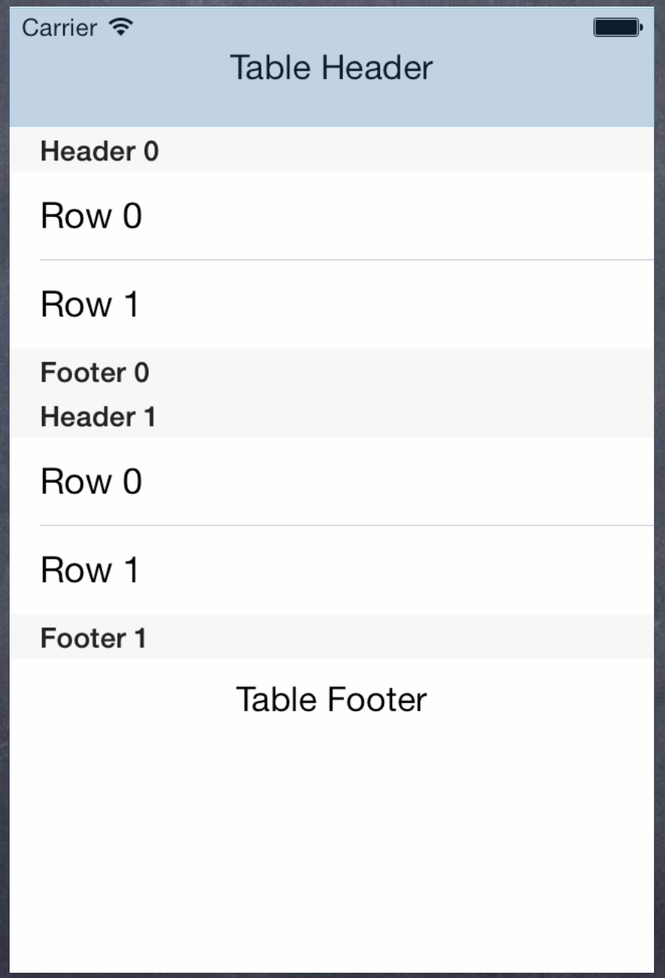


Static & Grouped

UITableView

Plain Style

Table Header




```
@property UIView *tableHeaderView;
```

UITableView

Plain Style

Table Header

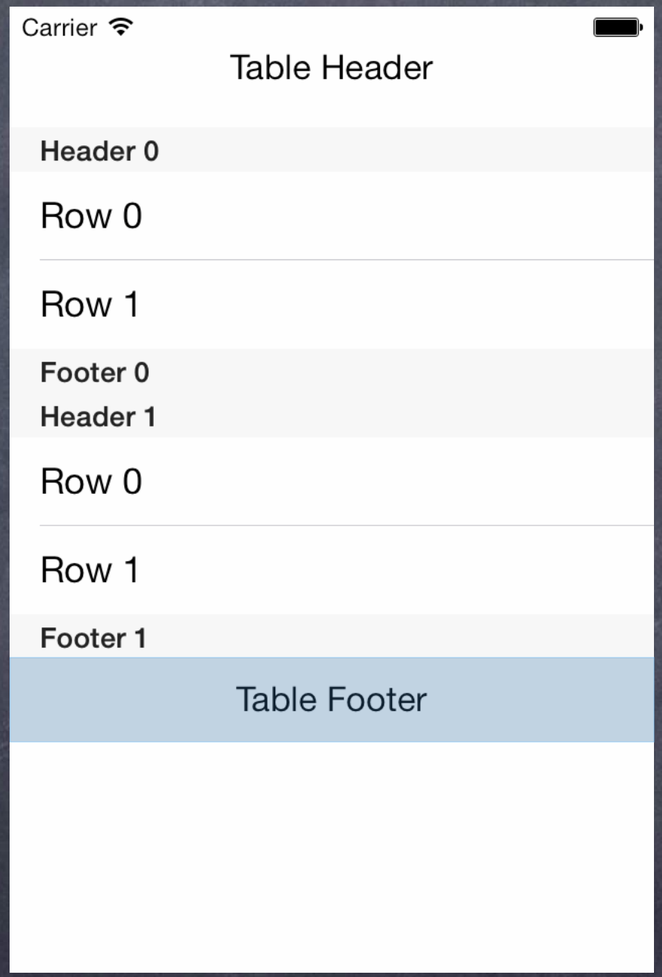
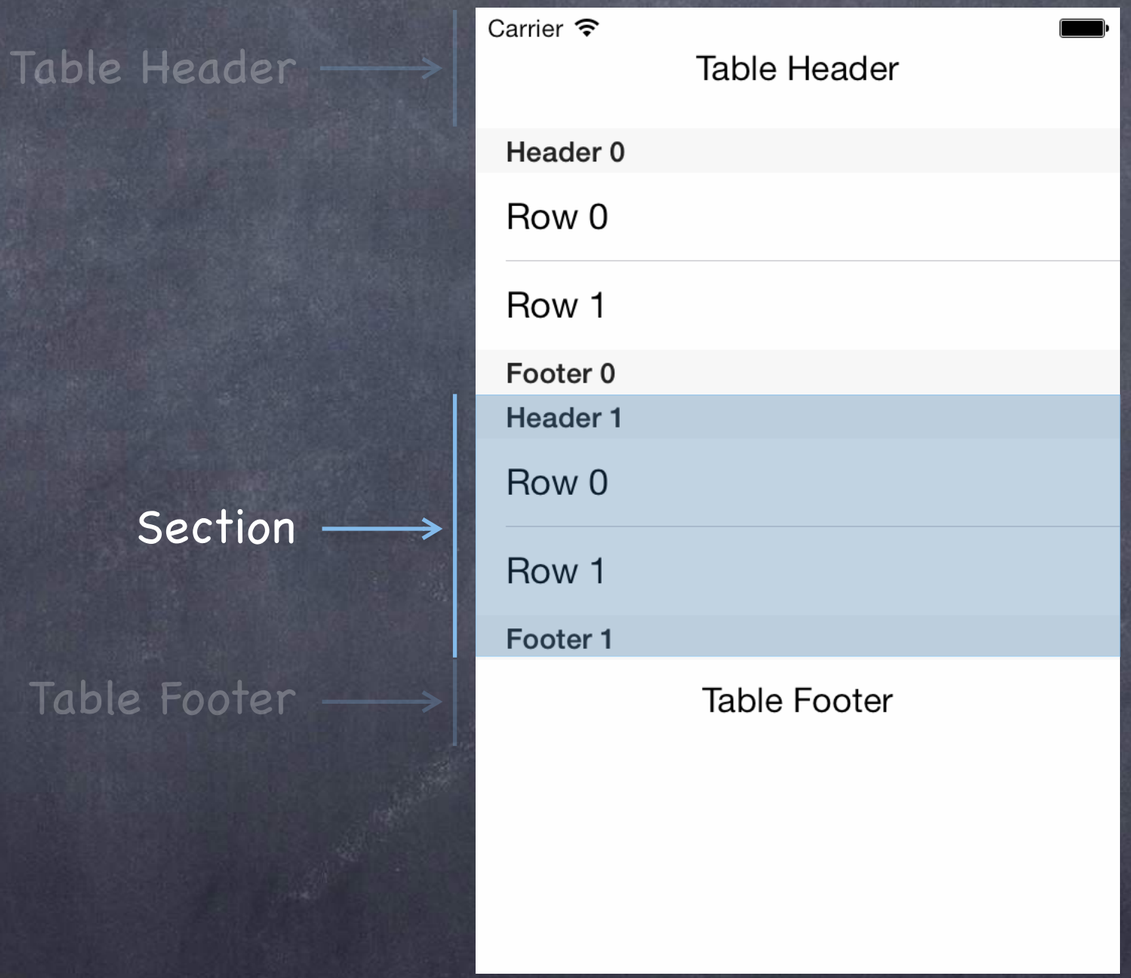


Table Footer

```
@property UIView *tableFooterView;
```

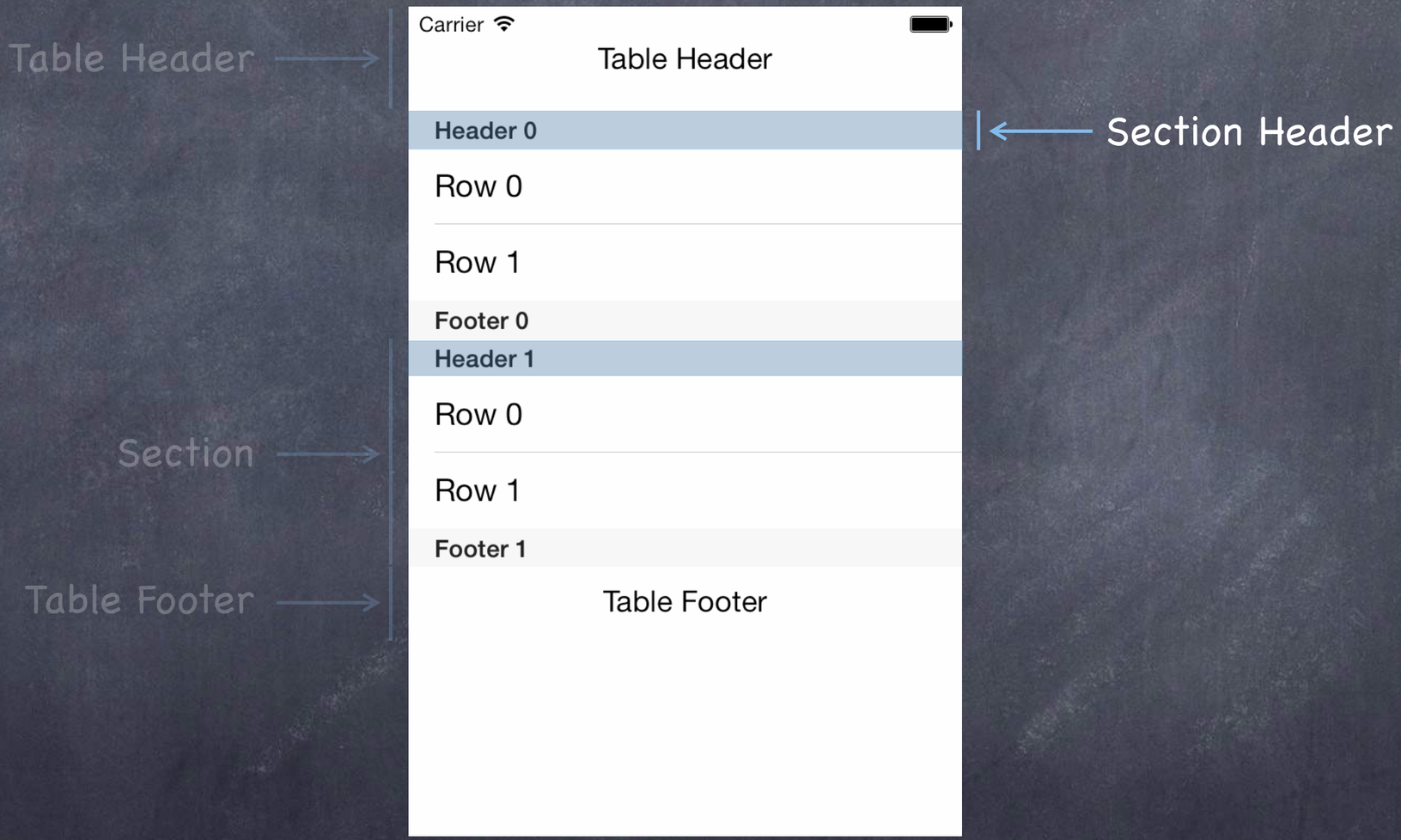
UITableView

Plain Style



UITableView

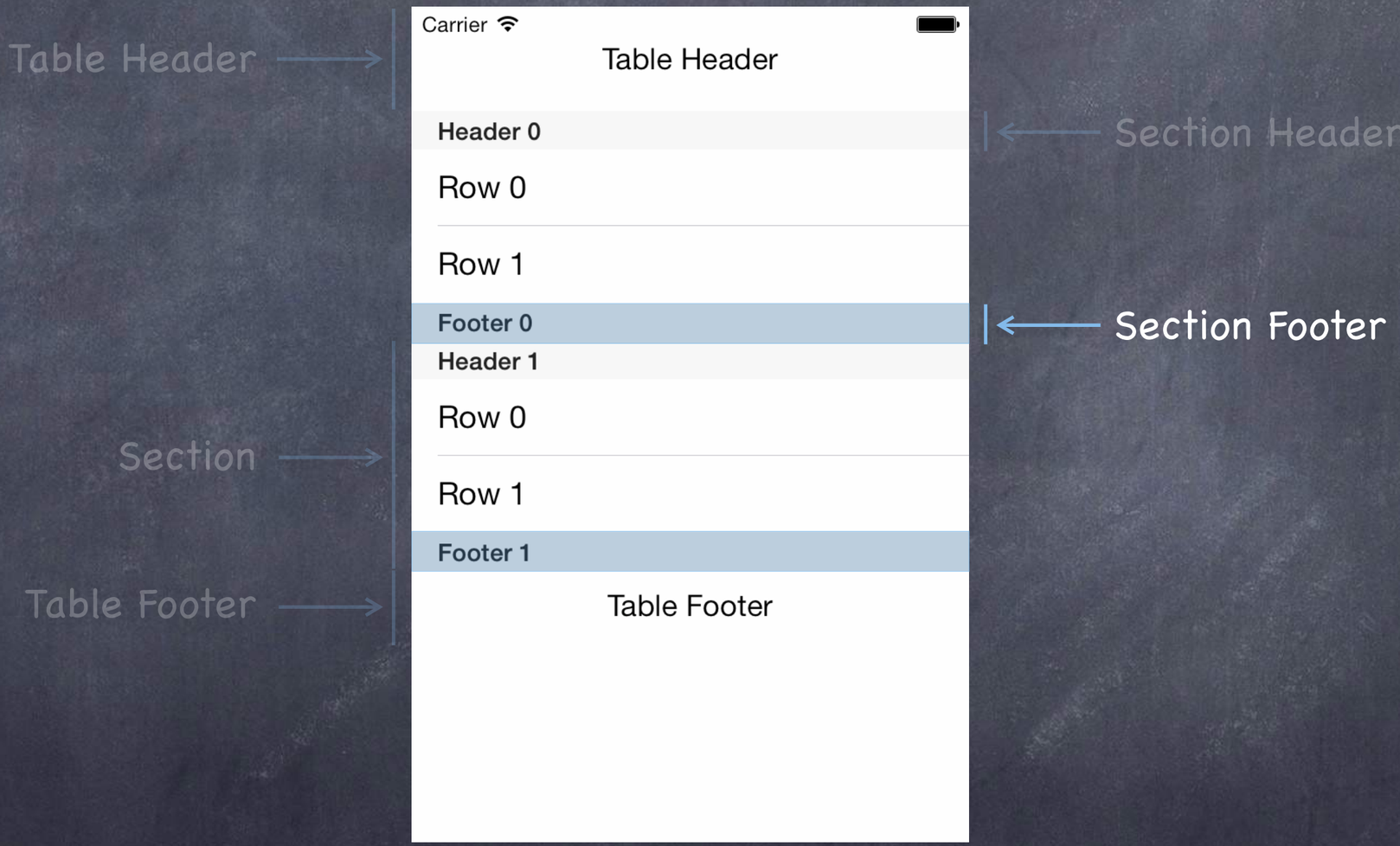
Plain Style



`UITableViewDataSource's tableView:titleForHeaderInSection:`

UITableView

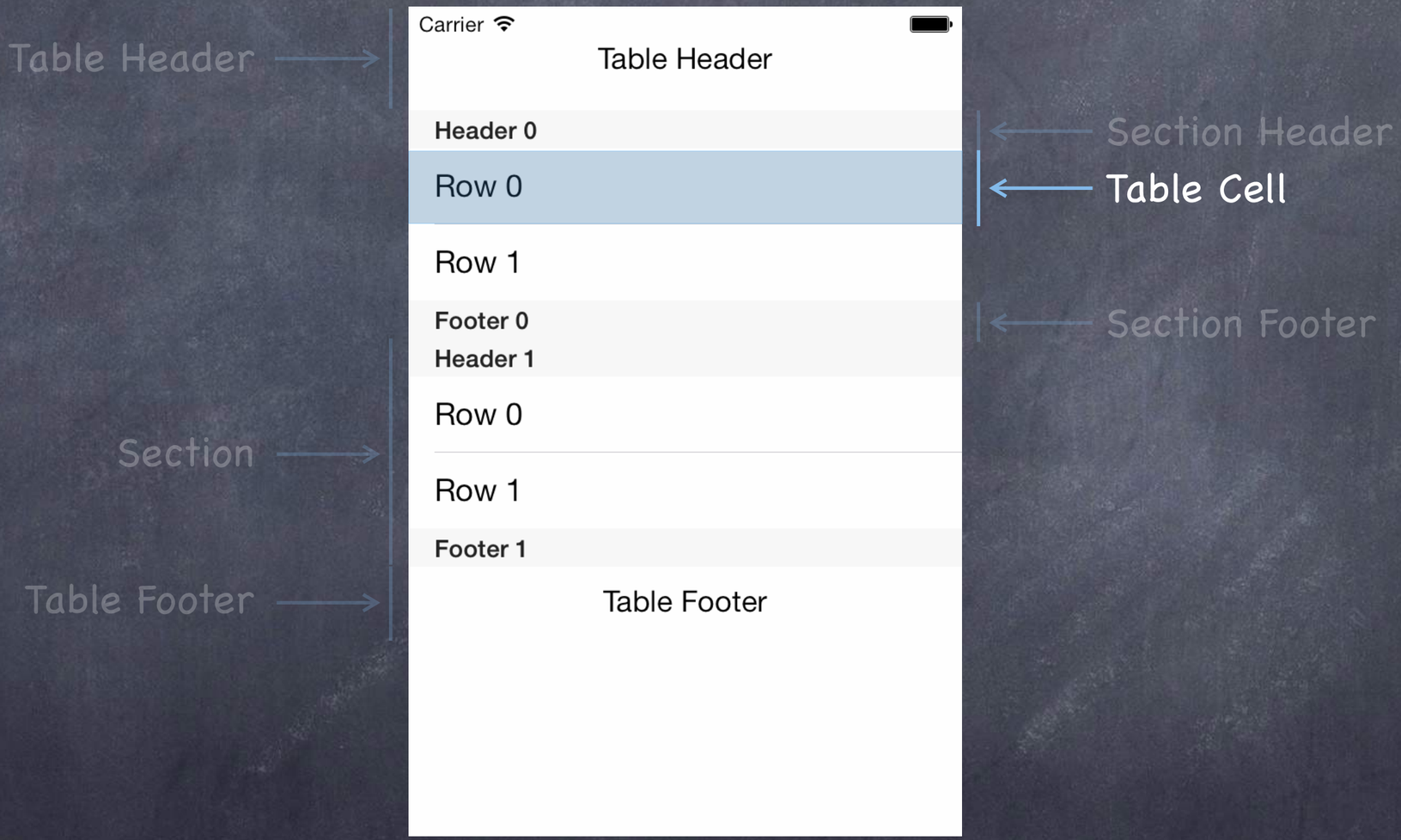
Plain Style



`UITableViewDataSource's tableView:titleForFooterInSection:`

UITableView

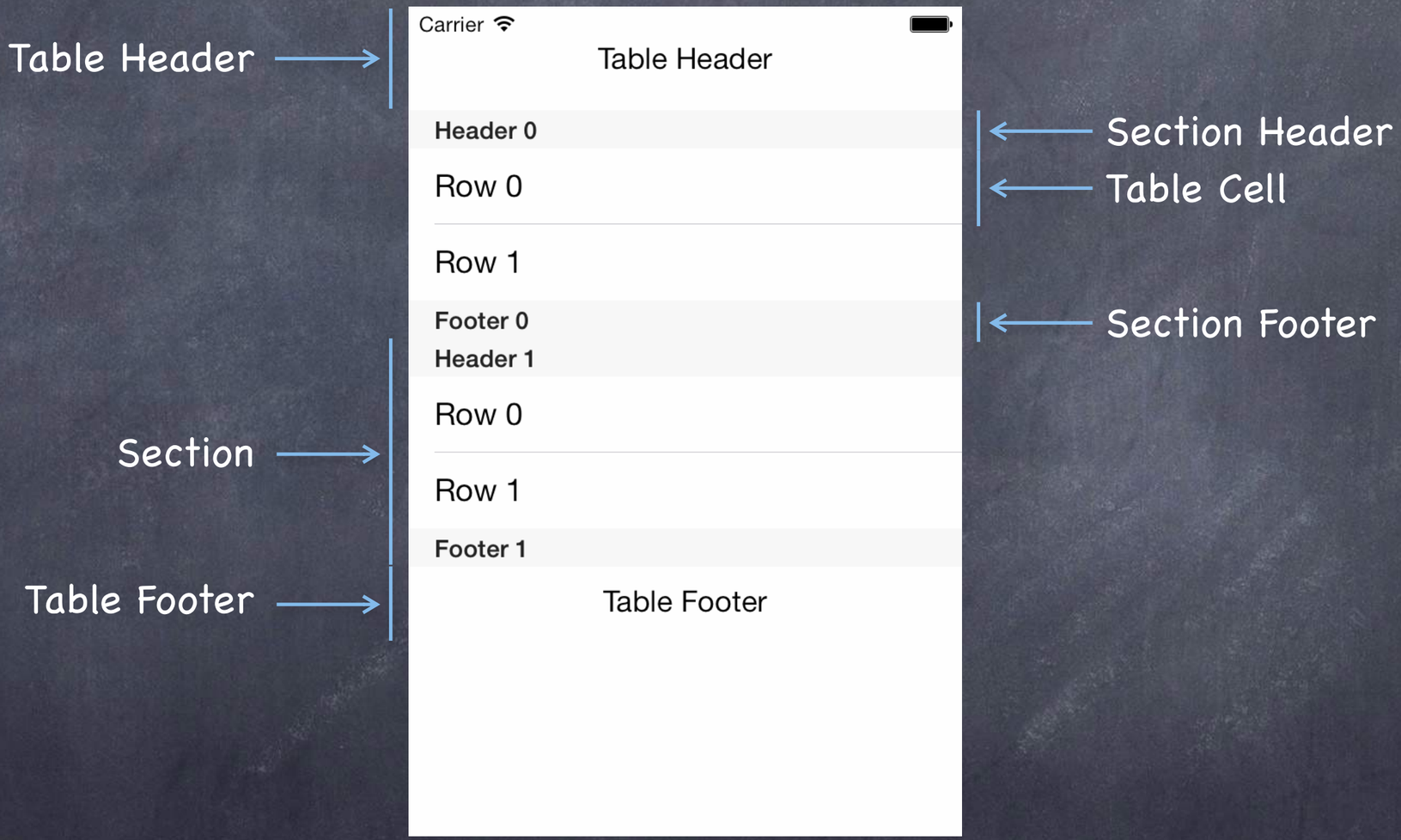
Plain Style



`UITableViewDataSource's tableView:cellForRowAtIndexPath:`

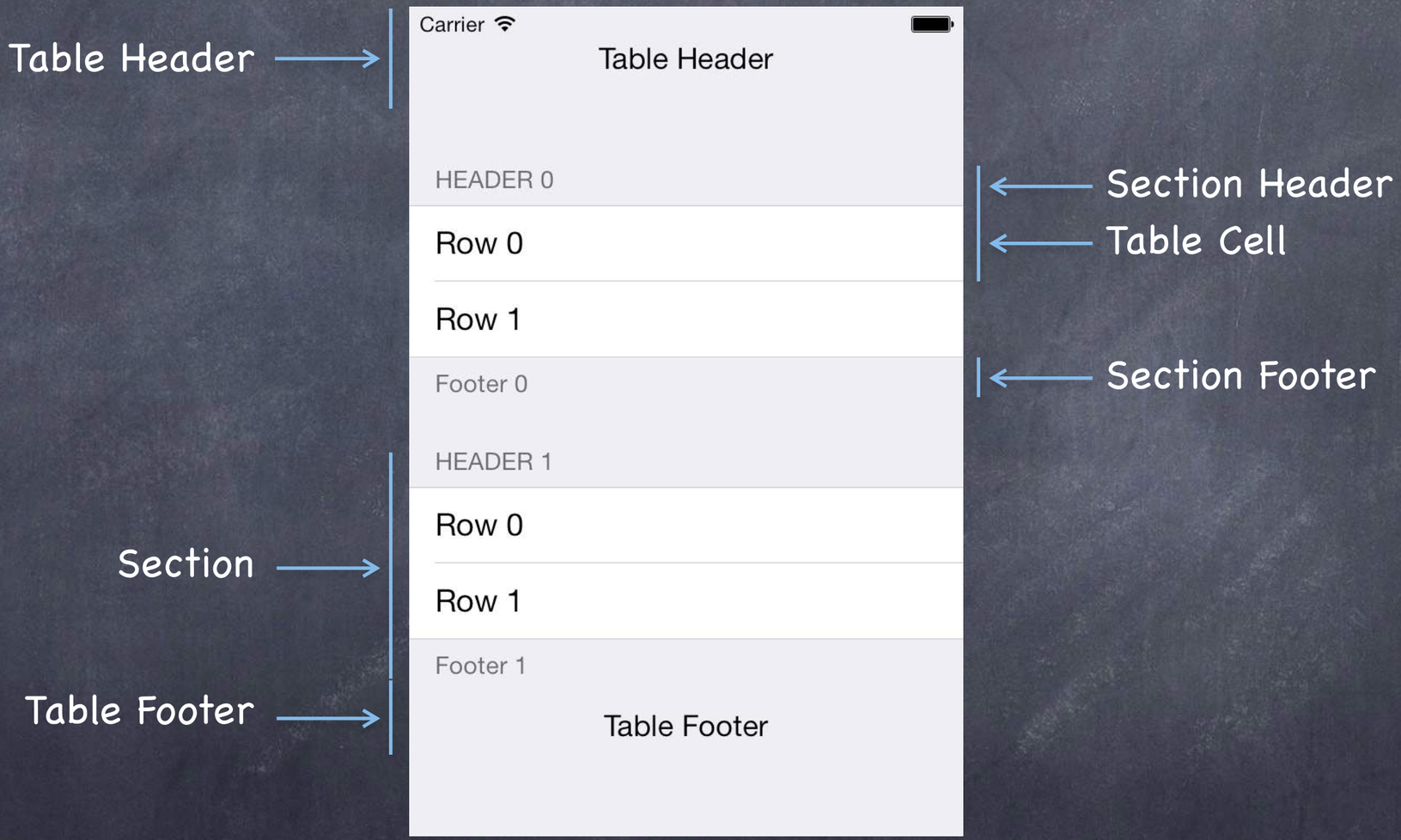
UITableView

Plain Style

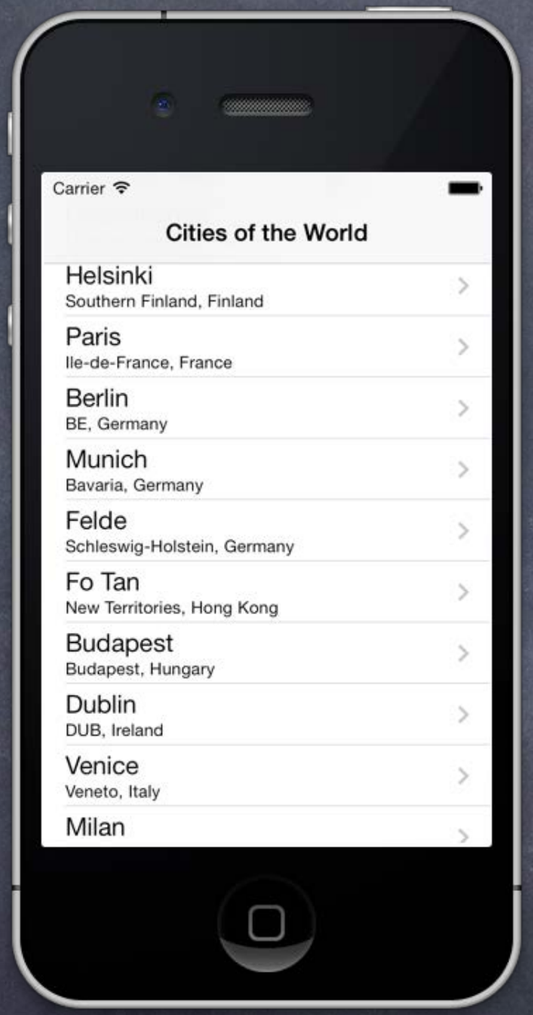


UITableView

Grouped Style



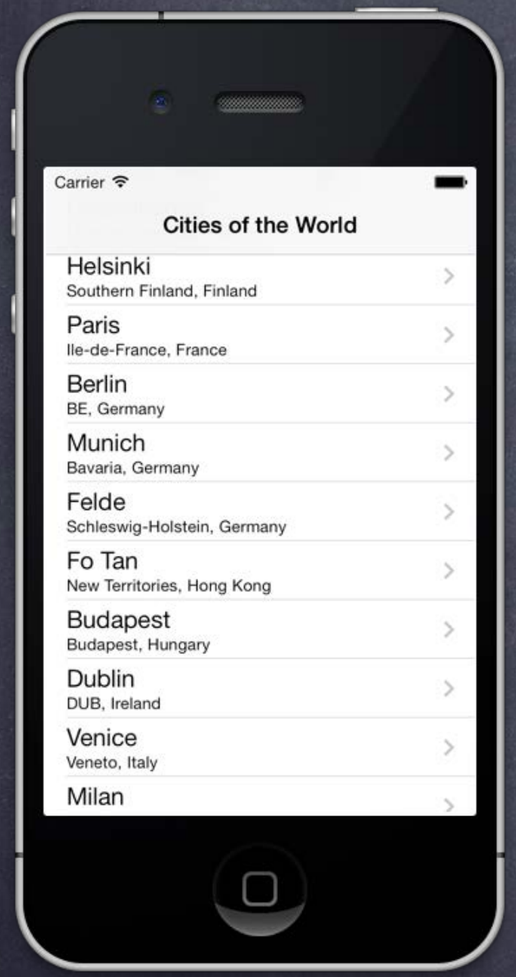
Sections or Not



No Sections

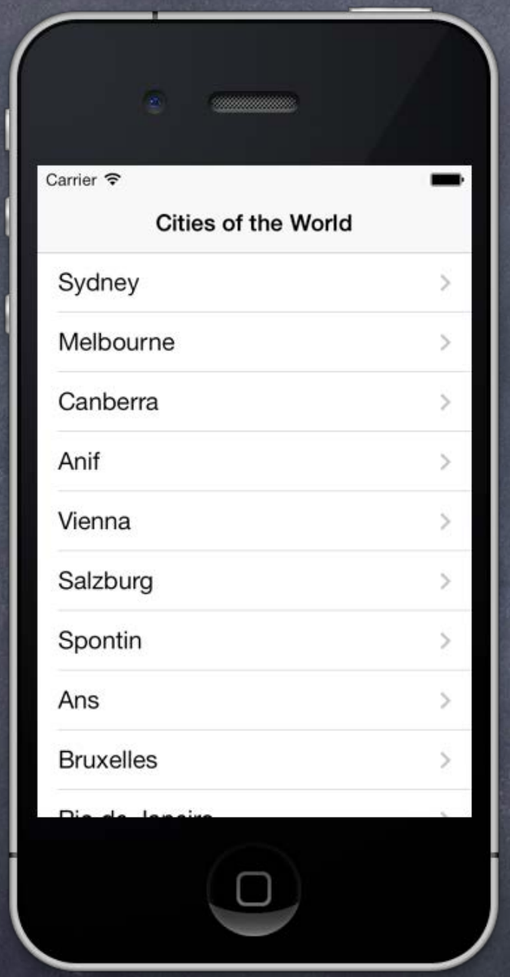
Sections

Cell Type



Subtitle

`UITableViewCellStyleSubtitle`

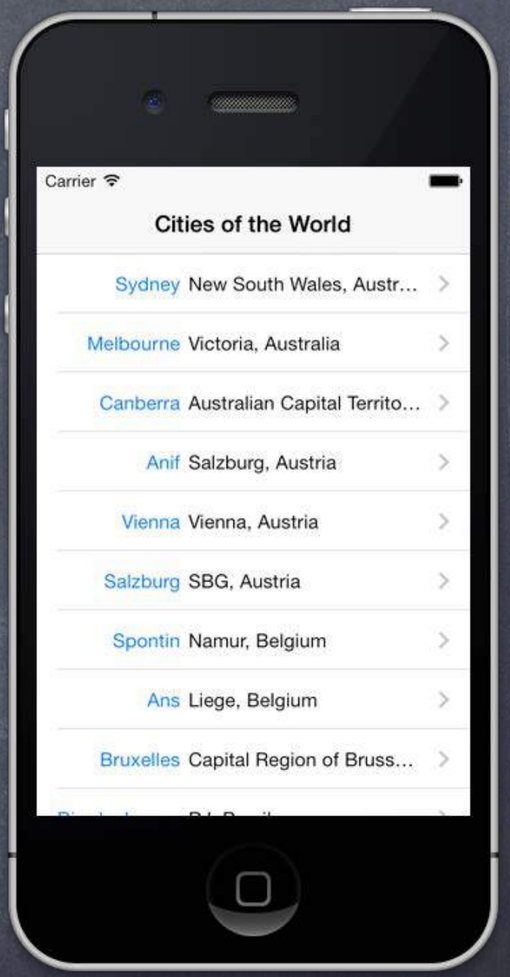


Basic

`UITableViewCellStyleDefault`

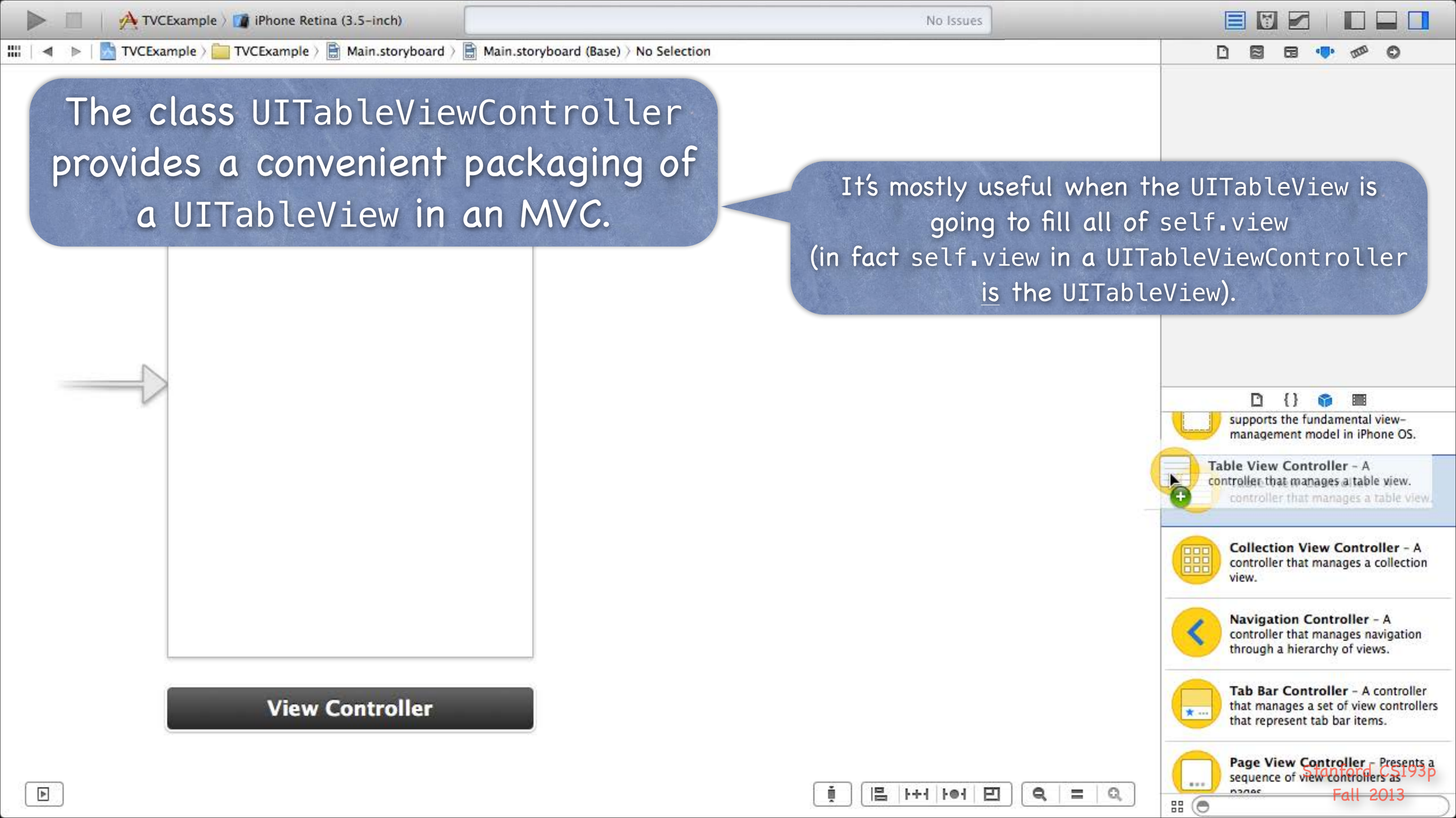
Right Detail

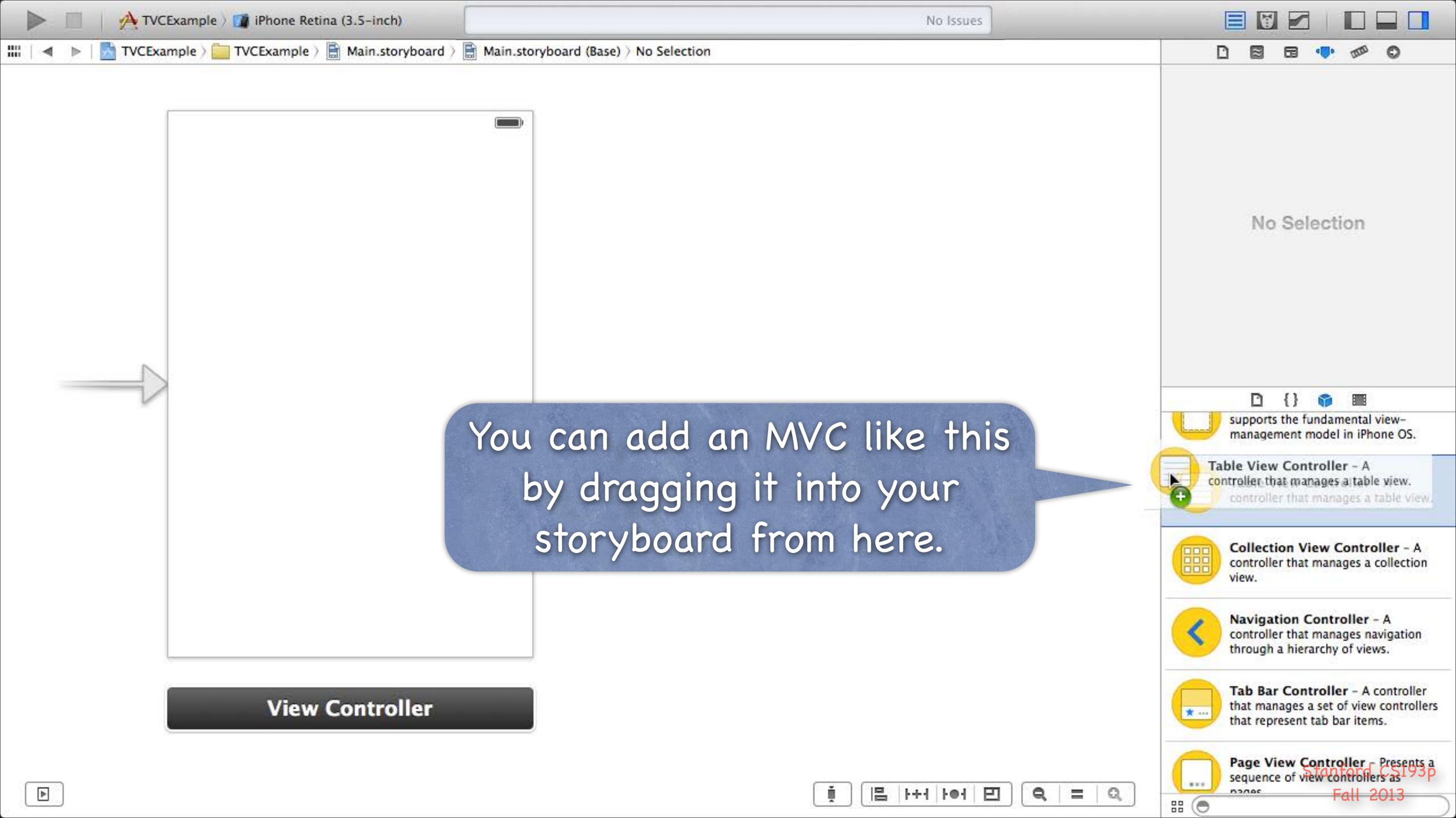
`UITableViewCellStyleValue1`

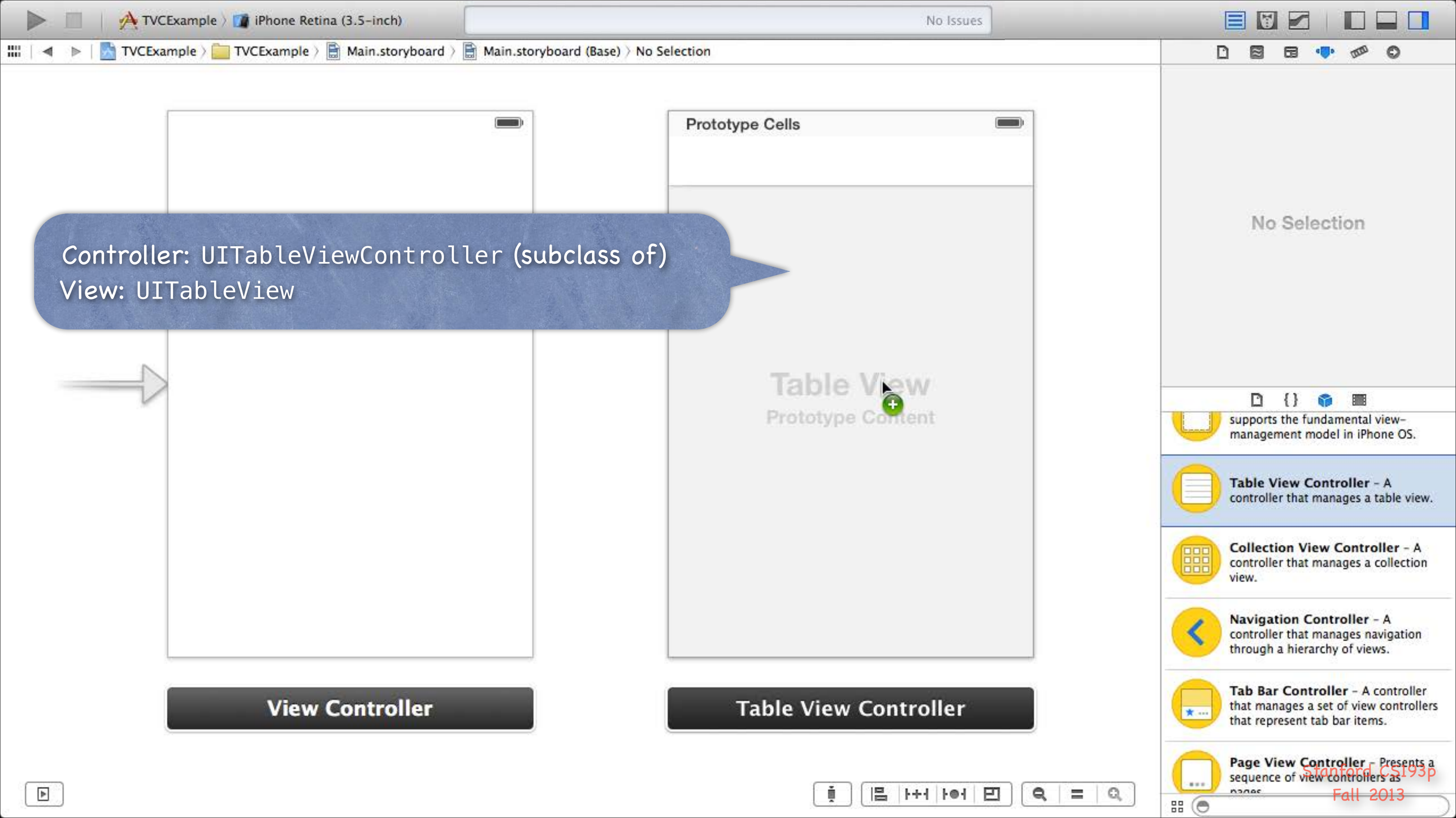


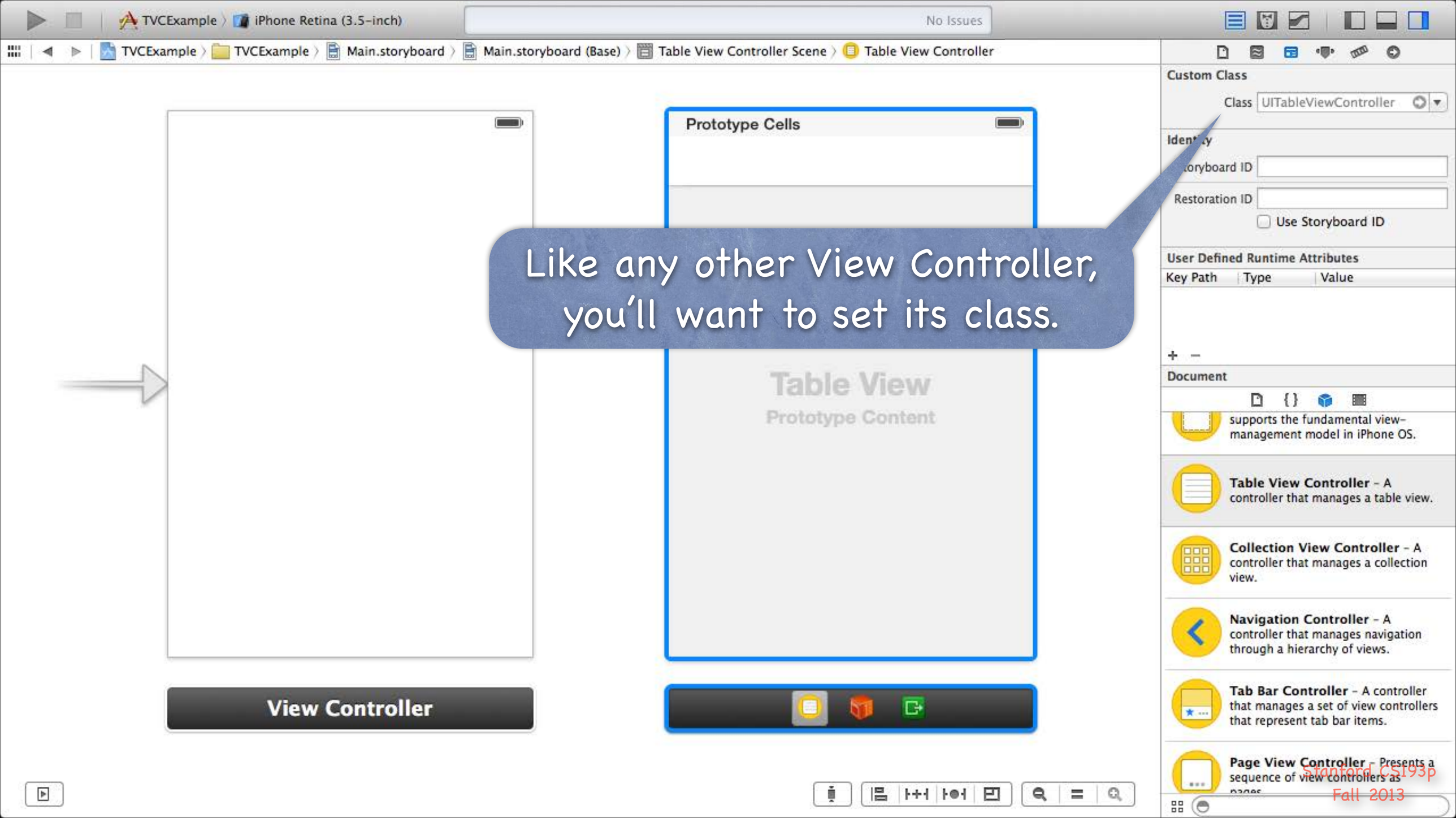
Left Detail

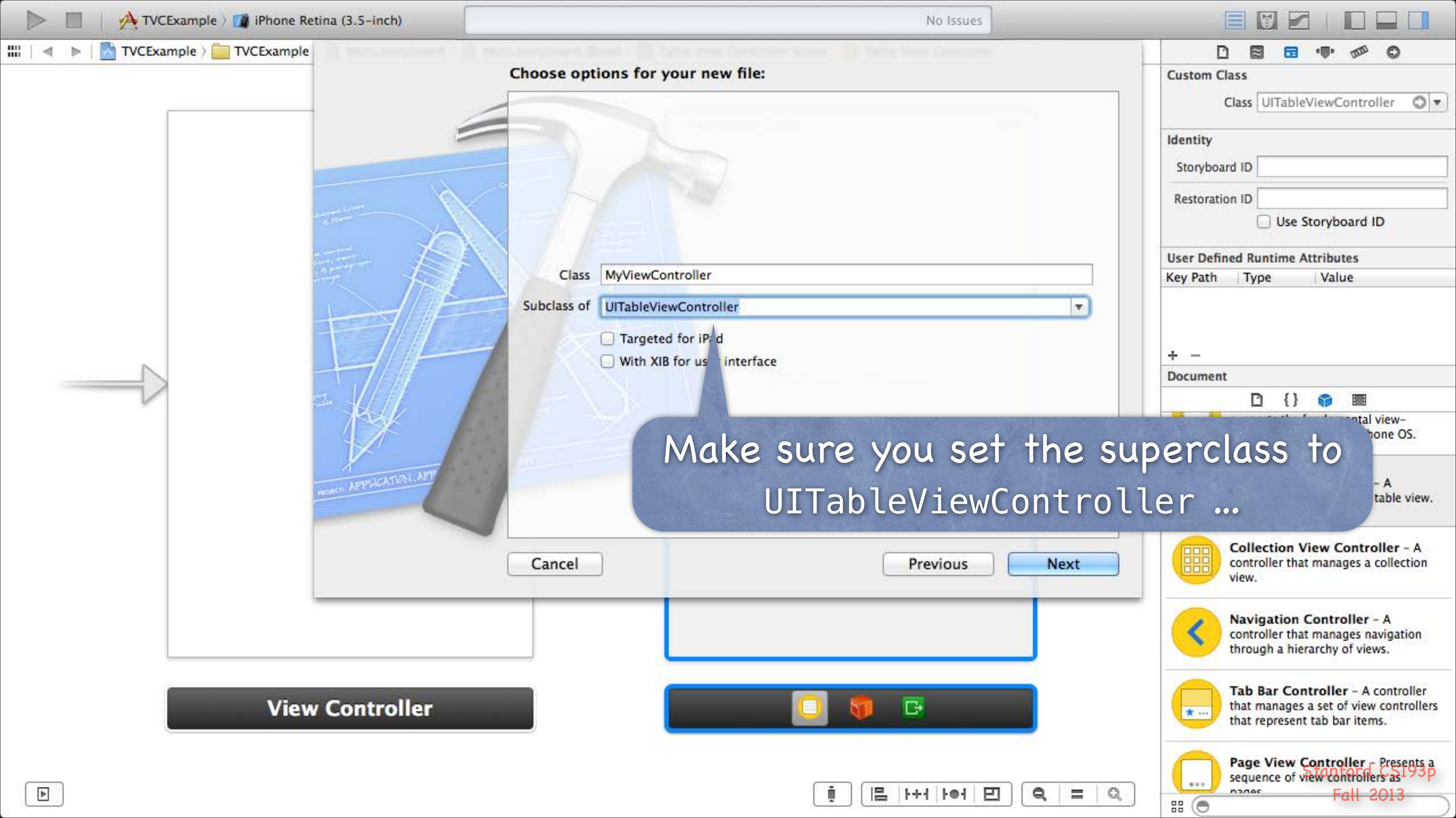
`UITableViewCellStyleValue2`

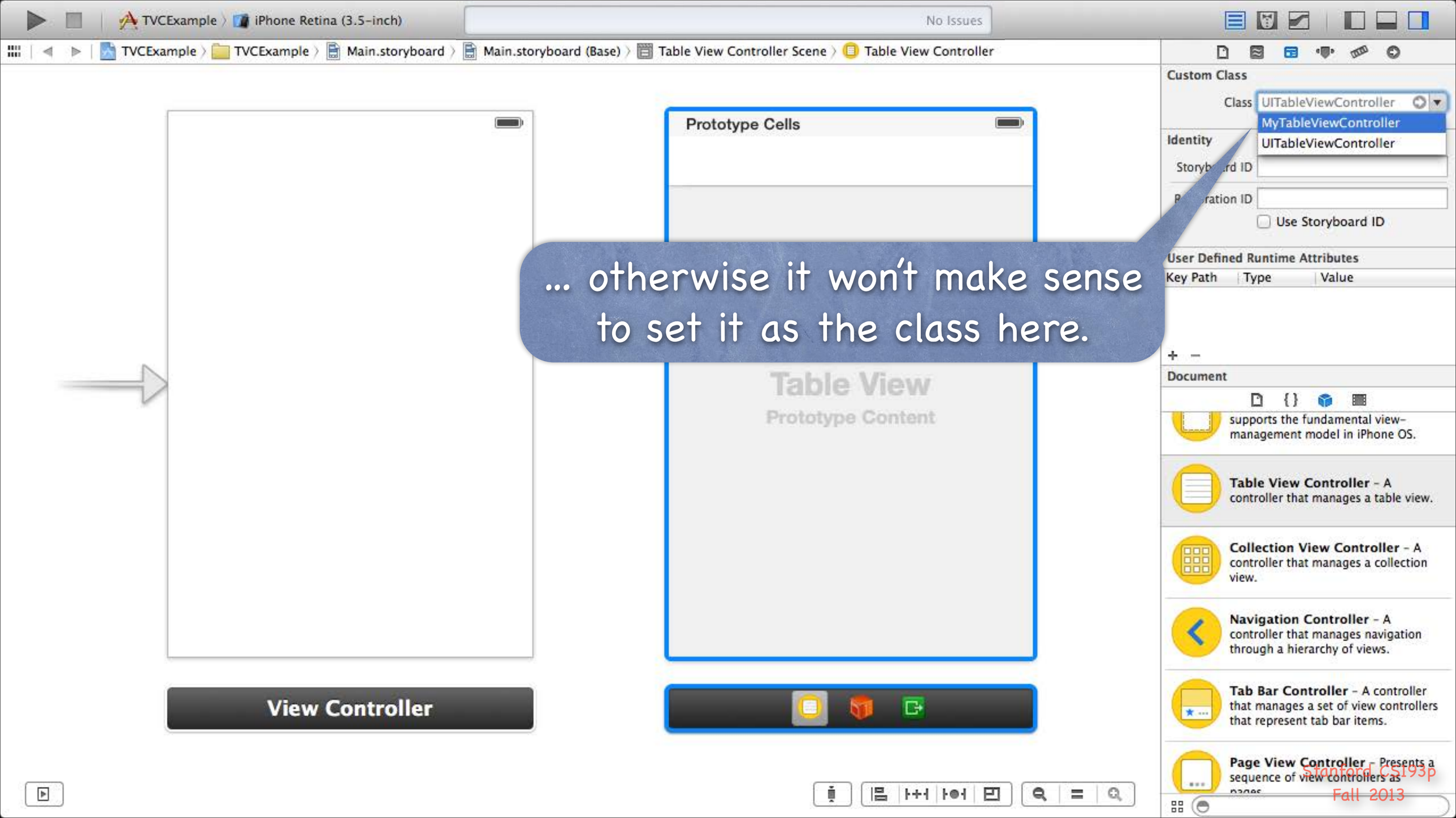

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
No Selection
The class UITableViewController provides a convenient packaging of a UITableView in an MVC.
It's mostly useful when the UITableView is going to fill all of self.view
(in fact self.view in a UITableViewController is the UITableView).
View Controller
supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar items.
Page View Controller – Presents a sequence of view controllers as
Stanford CS193p
Fall 2013

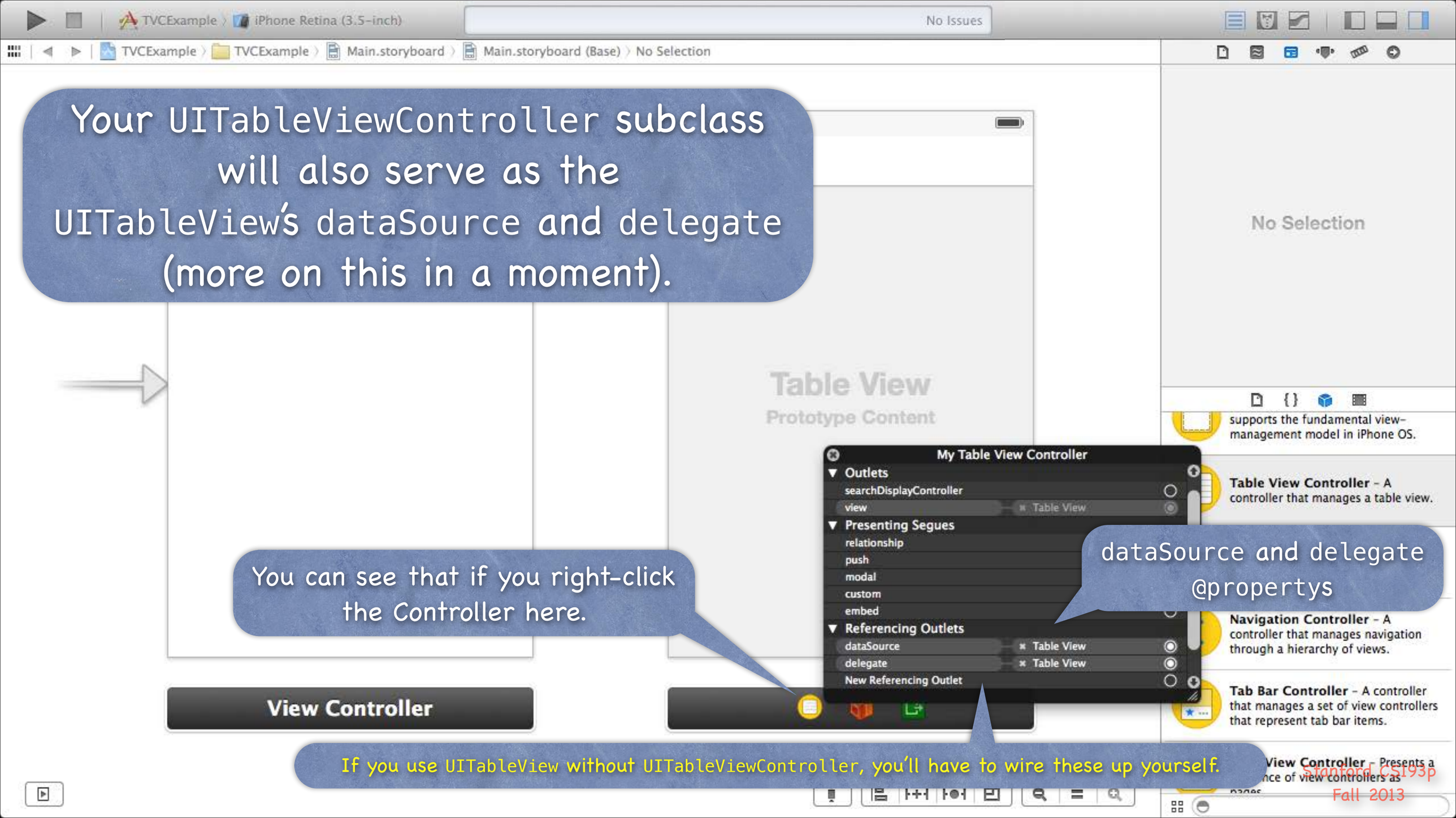

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
No Selection
No Selection
View Controller
You can add an MVC like this by dragging it into your storyboard from here.
supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar items.
Page View Controller – Presents a sequence of view controllers as
Stanford CS193p
Fall 2013

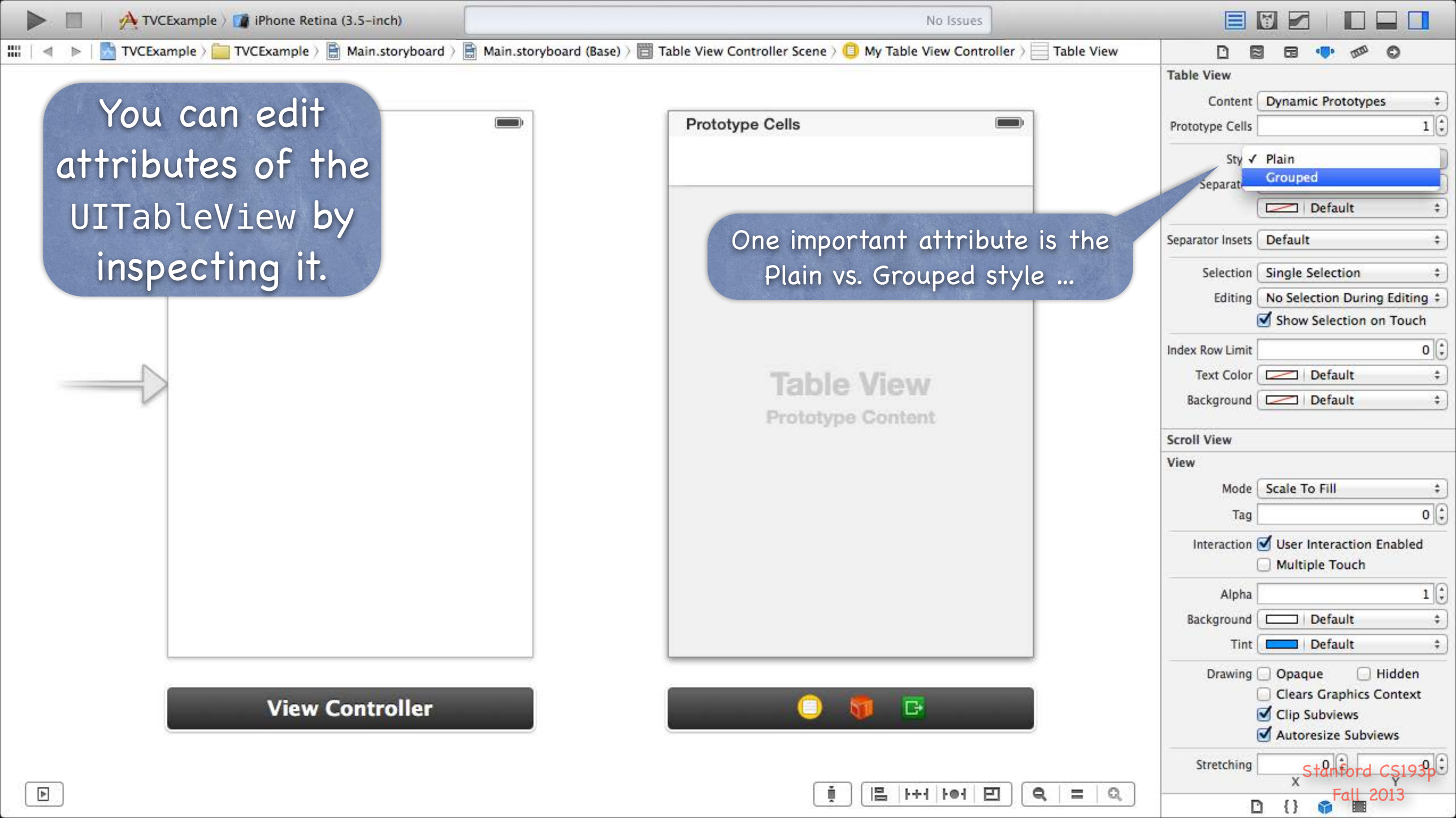

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
No Selection
Prototype Cells
Table View
Prototype Content
Controller: UITableViewController (subclass of)
View: UITableView
View Controller
Table View Controller
No Selection
supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar items.
Page View Controller – Presents a sequence of view controllers as
Stanford CS193p
Fall 2013

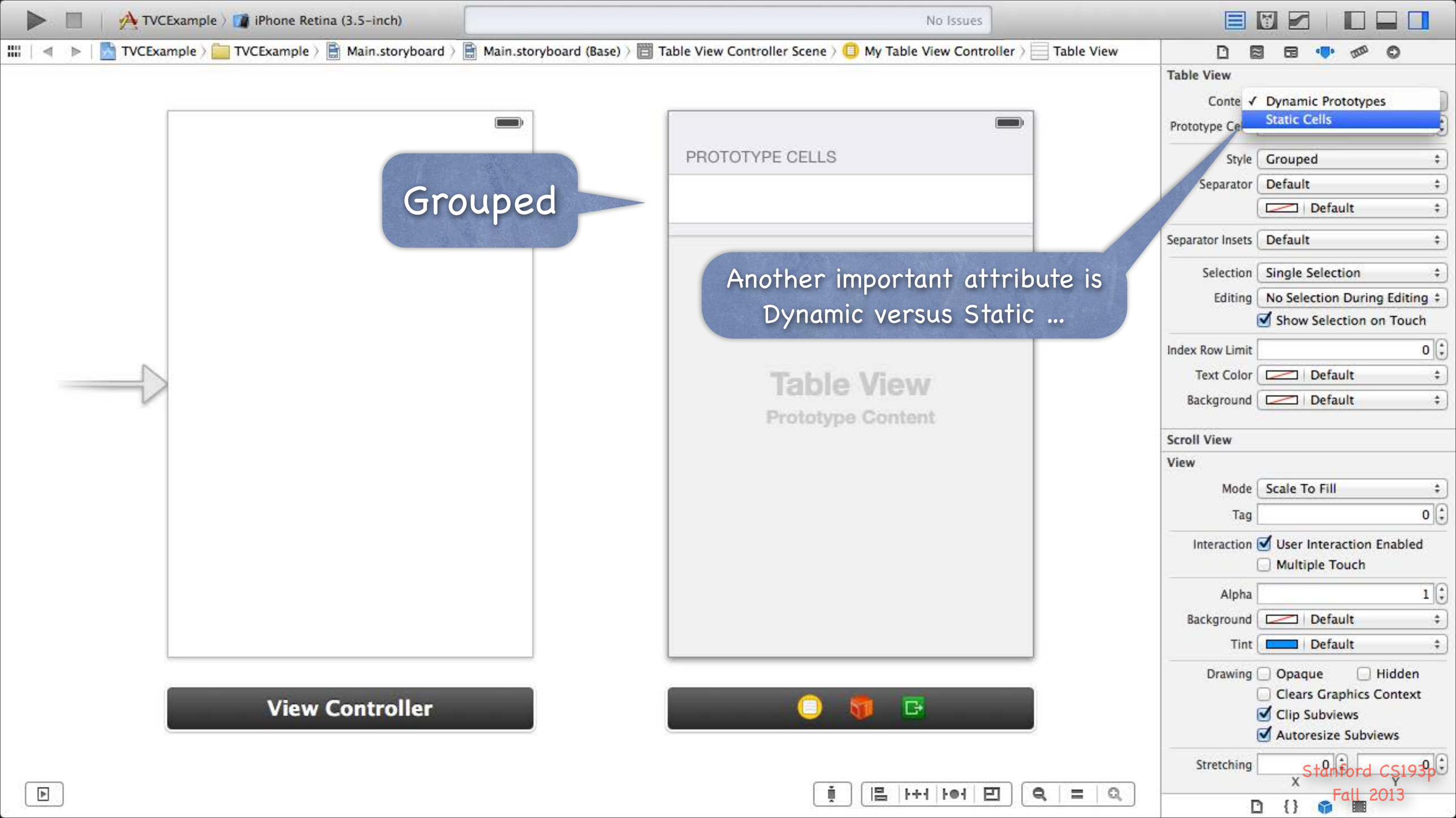

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
Table View Controller
Prototype Cells
Table View
Prototype Content
View Controller
Like any other View Controller, you'll want to set its class.
Custom Class
Class
UITableViewController
Restoration ID
Use Storyboard ID
User Defined Runtime Attributes
Key Path
Type
Value
Document
supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar items.
Page View Controller – Presents a sequence of view controllers as
Stanford CS193p
Fall 2013

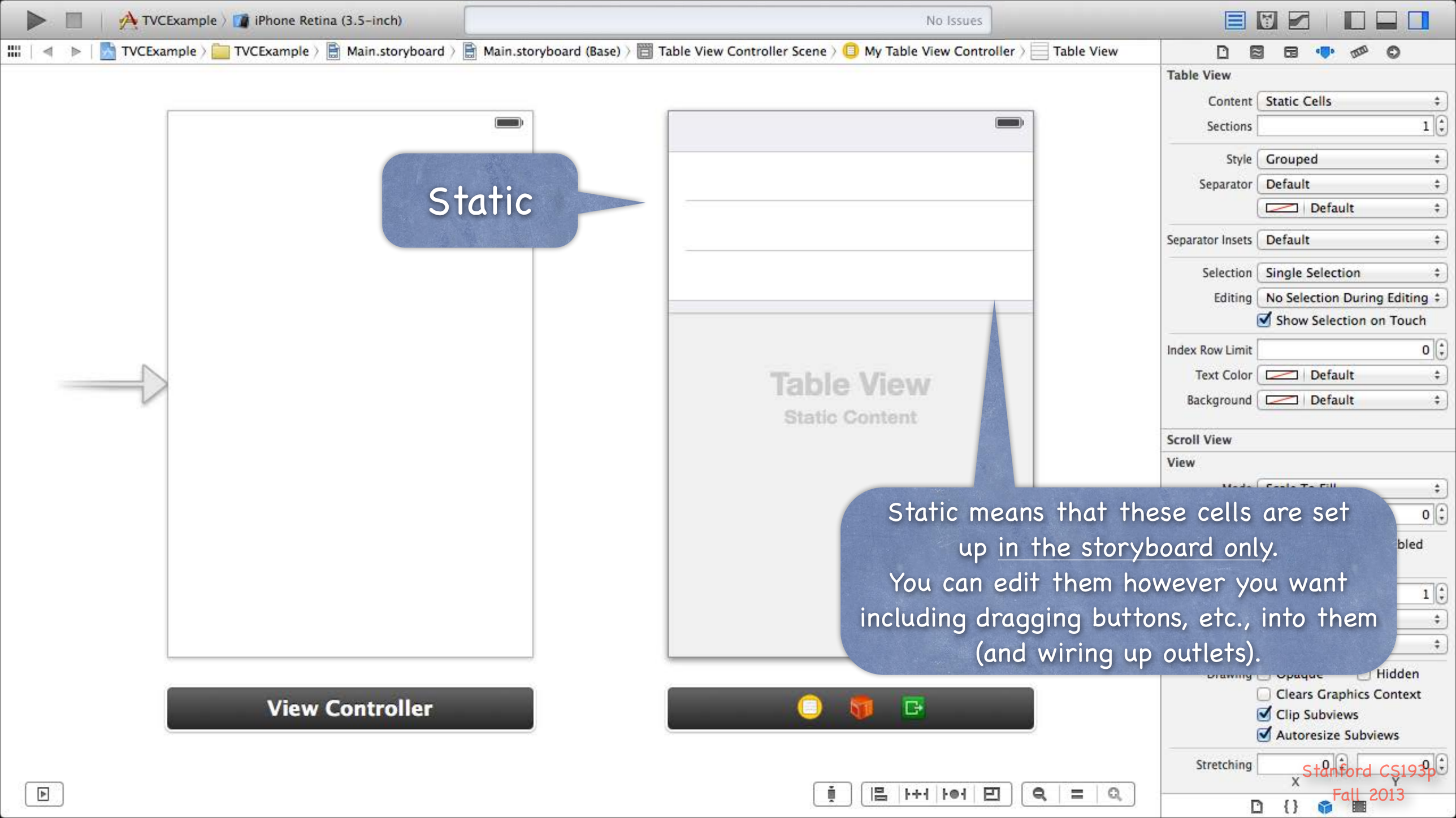

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Choose options for your new file:
Class
MyViewController
Subclass of
UITableViewController
Targeted for iPad
With XIB for user interface
Cancel
Previous
Next
Make sure you set the superclass to UITableViewController …
View Controller
Custom Class
Class
UITableViewController
Identity
Storyboard ID
Restoration ID
Use Storyboard ID
User Defined Runtime Attributes
Key Path
Type
Value
Document
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar items.
Page View Controller – Presents a sequence of view controllers as
Stanford CS193p
Fall 2013

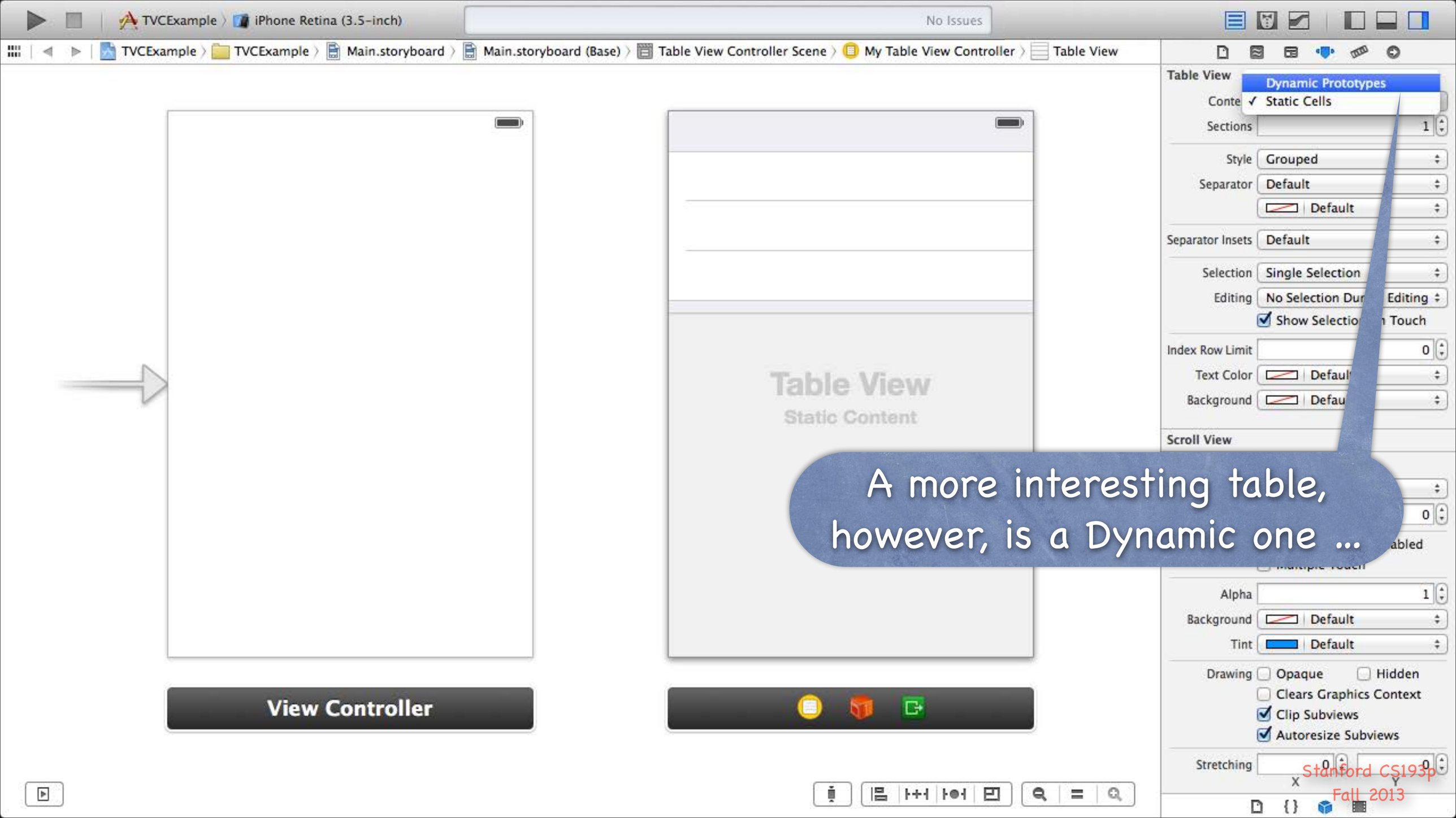

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
Table View Controller
Prototype Cells
Table View
Prototype Content
View Controller
Custom Class
Class
UITableViewController
MyTableViewController
UITableViewController
Identity
Use Storyboard ID
User Defined Runtime Attributes
Key Path
Type
Value
Document
supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar items.
Page View Controller – Presents a sequence of view controllers as
… otherwise it won't make sense to set it as the class here.
Stanford CS193p
Fall 2013

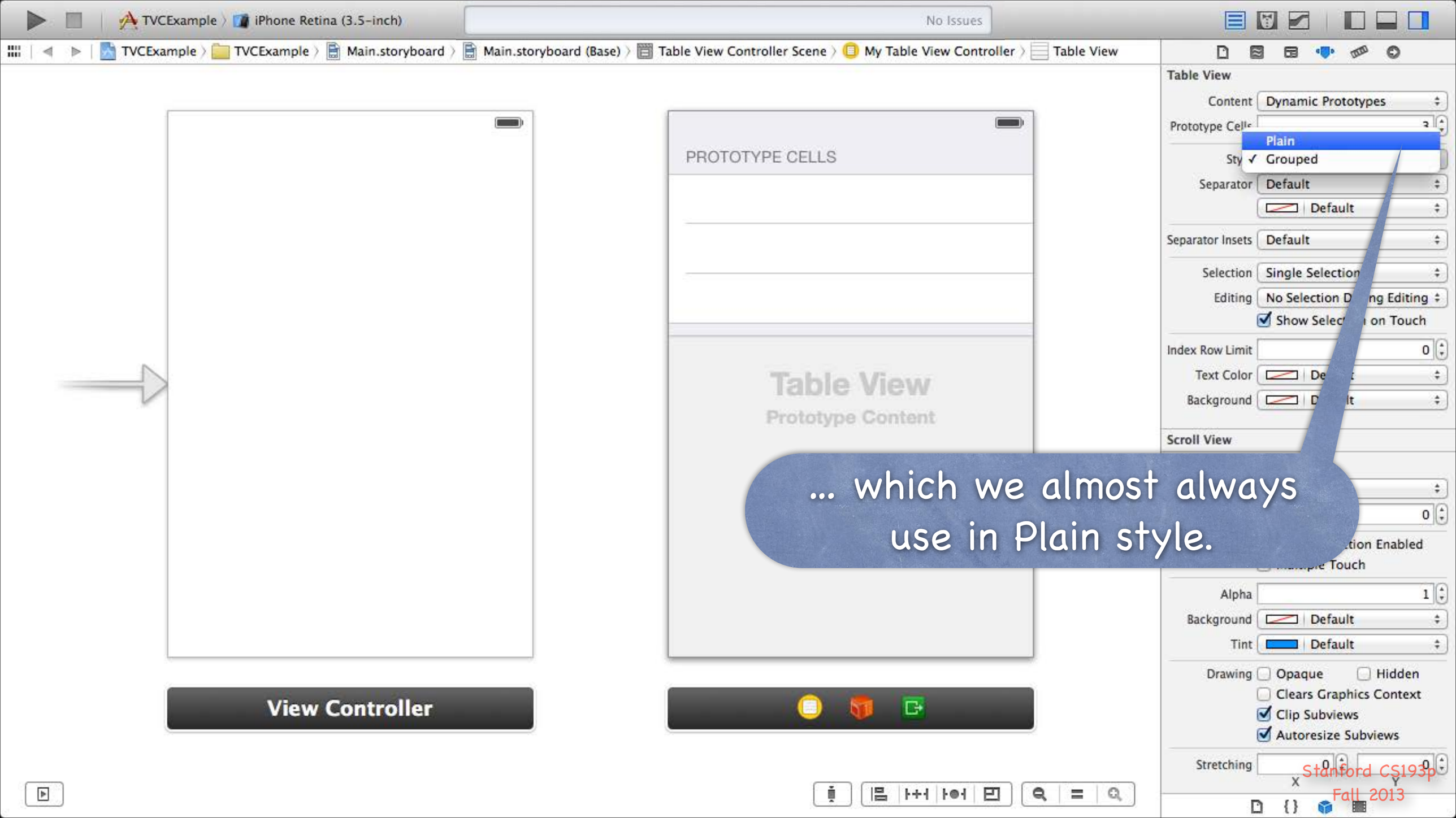

Your UITableViewController subclass will also serve as the UITableView's dataSource and delegate (more on this in a moment).
TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
No Selection
Table View
Prototype Content
View Controller
My Table View Controller
Outlets
searchDisplayController
view
Table View
Presenting Segues
relationship
push
modal
custom
embed
Referencing Outlets
dataSource
Table View
delegate
Table View
New Referencing Outlet
You can see that if you right-click the Controller here.
dataSource and delegate @propertys
If you use UITableView without UITableViewController, you'll have to wire these up yourself.
No Selection
supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar items.
Stanford CS193p
Fall 2013

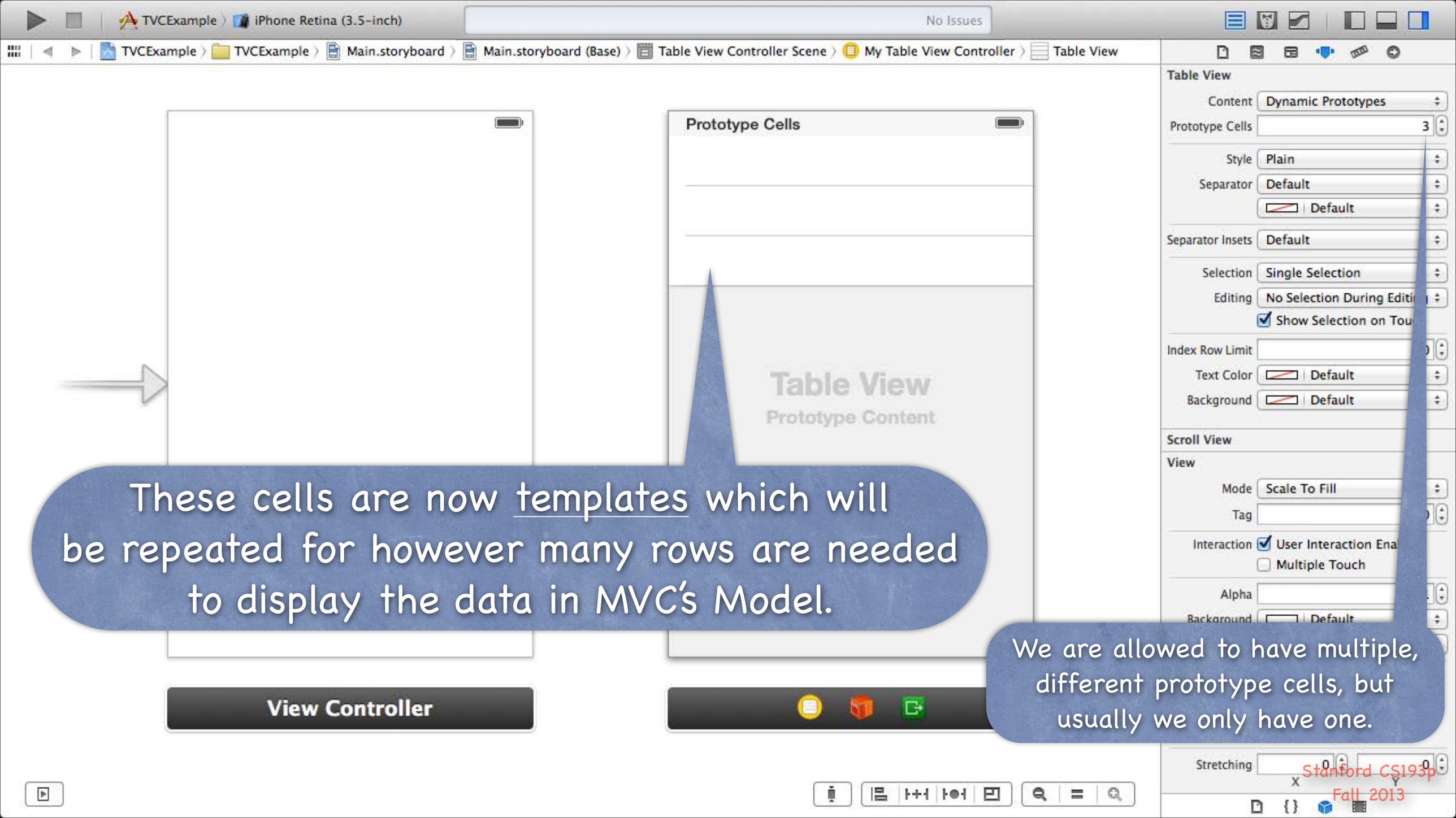

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
My Table View Controller
Table View
You can edit attributes of the UITableView by inspecting it.
Prototype Cells
One important attribute is the Plain vs. Grouped style ...
Table View
Prototype Content
View Controller
Table View
Content
Dynamic Prototypes
Prototype Cells
1
Plain
Grouped
Default
Separator Insets
Default
Selection
Single Selection
Editing
No Selection During Editing
Show Selection on Touch
Index Row Limit
0
Text Color
Default
Background
Default
Scroll View
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
X
Y
Stanford CS193p
Fall 2013

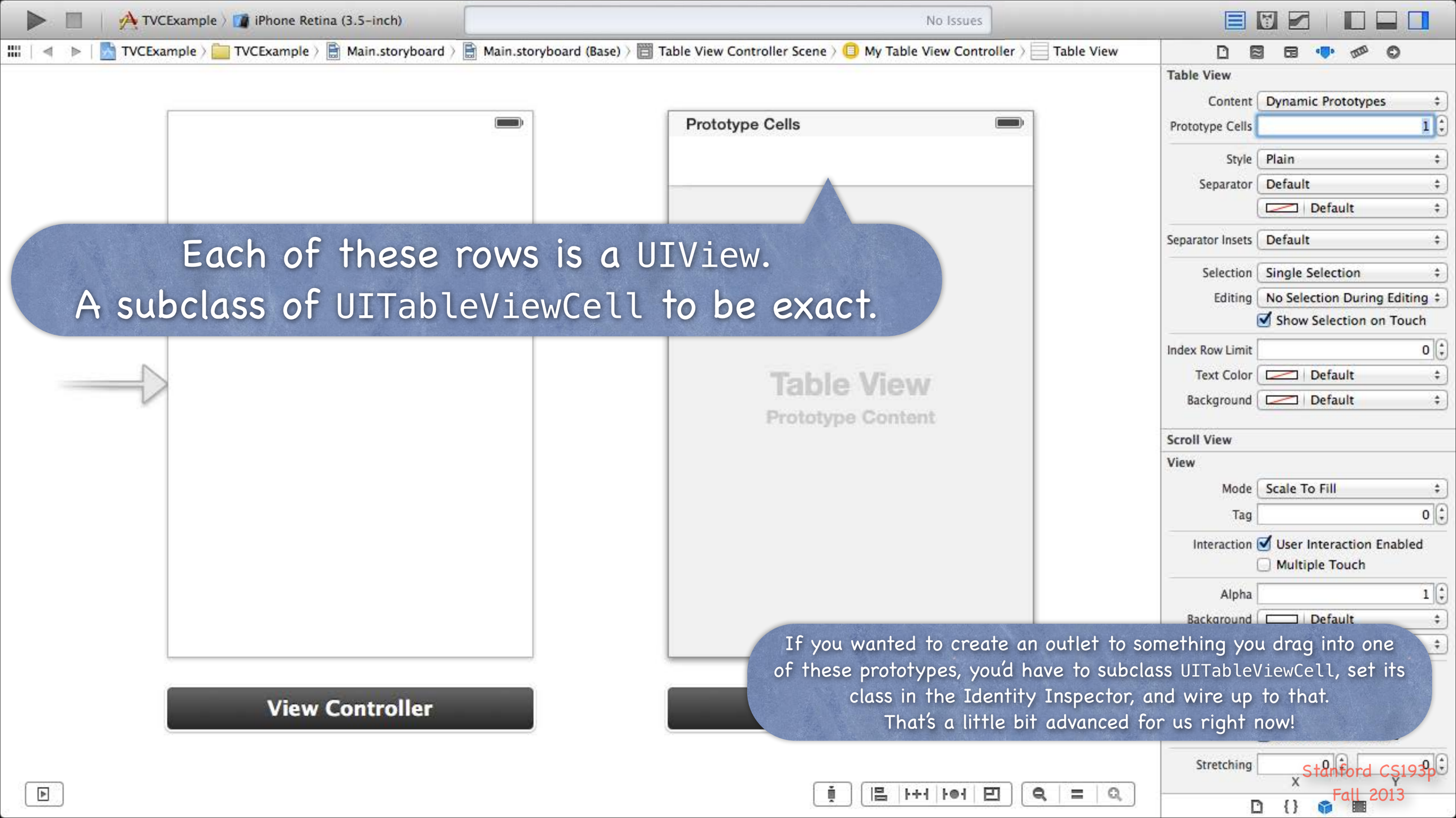

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
My Table View Controller
Table View
Grouped
PROTOTYPE CELLS
Another important attribute is Dynamic versus Static ...
Table View
Prototype Content
View Controller
Table View
Dynamic Prototypes
Static Cells
Style
Grouped
Separator
Default
Default
Separator Insets
Default
Selection
Single Selection
Editing
No Selection During Editing
Show Selection on Touch
Index Row Limit
0
Text Color
Default
Background
Default
Scroll View
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
0
X
Y
Stanford CS193p
Fall 2013

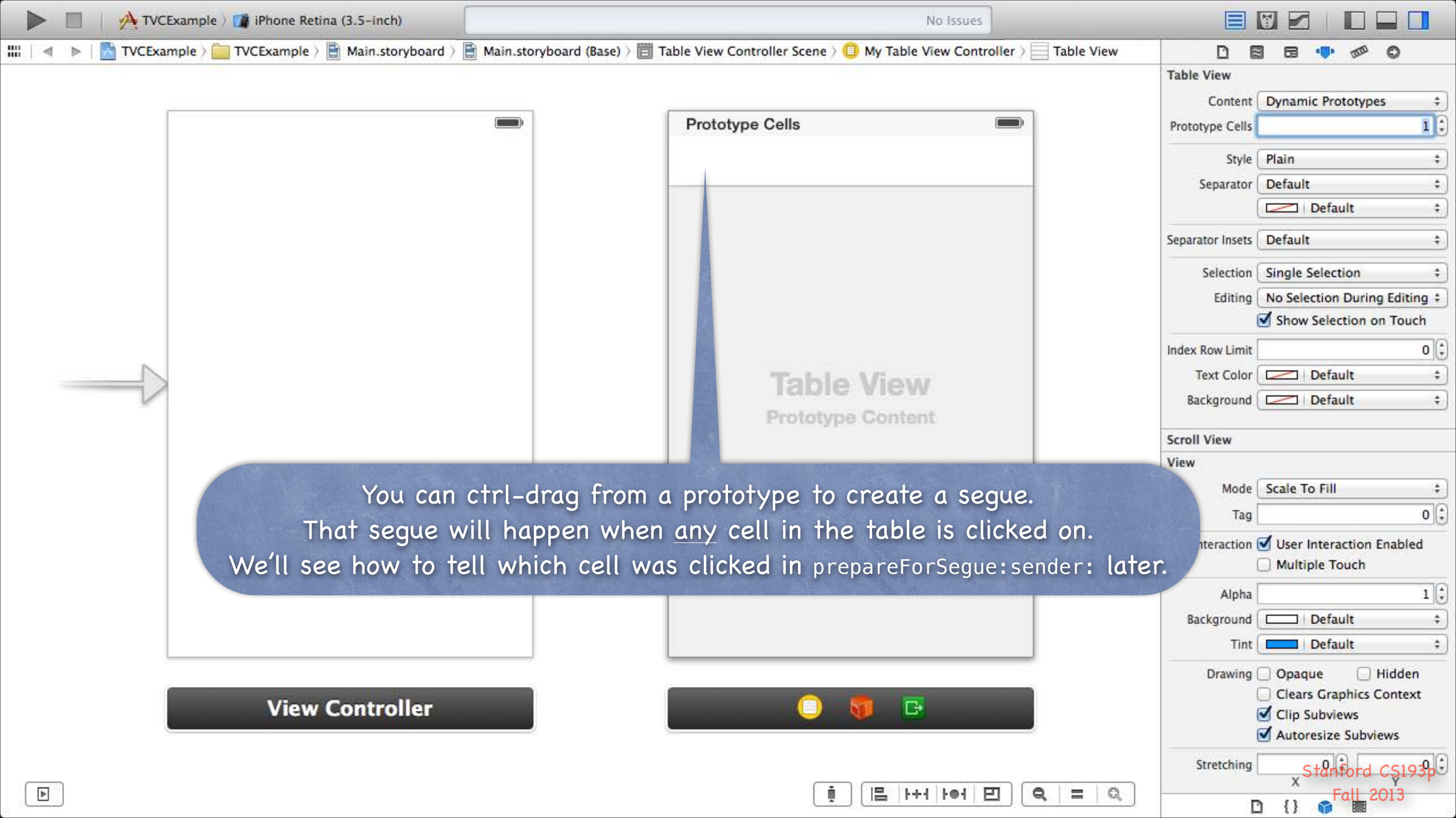

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
My Table View Controller
Table View
Static
Table View
Static Content
View Controller
Static means that these cells are set up in the storyboard only.
You can edit them however you want including dragging buttons, etc., into them (and wiring up outlets).
Table View
Content
Static Cells
Sections
1
Style
Grouped
Separator
Default
Default
Separator Insets
Default
Selection
Single Selection
Editing
No Selection During Editing
Show Selection on Touch
Index Row Limit
0
Text Color
Default
Background
Default
Scroll View
View
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
X
Y
Stanford CS193p
Fall 2013

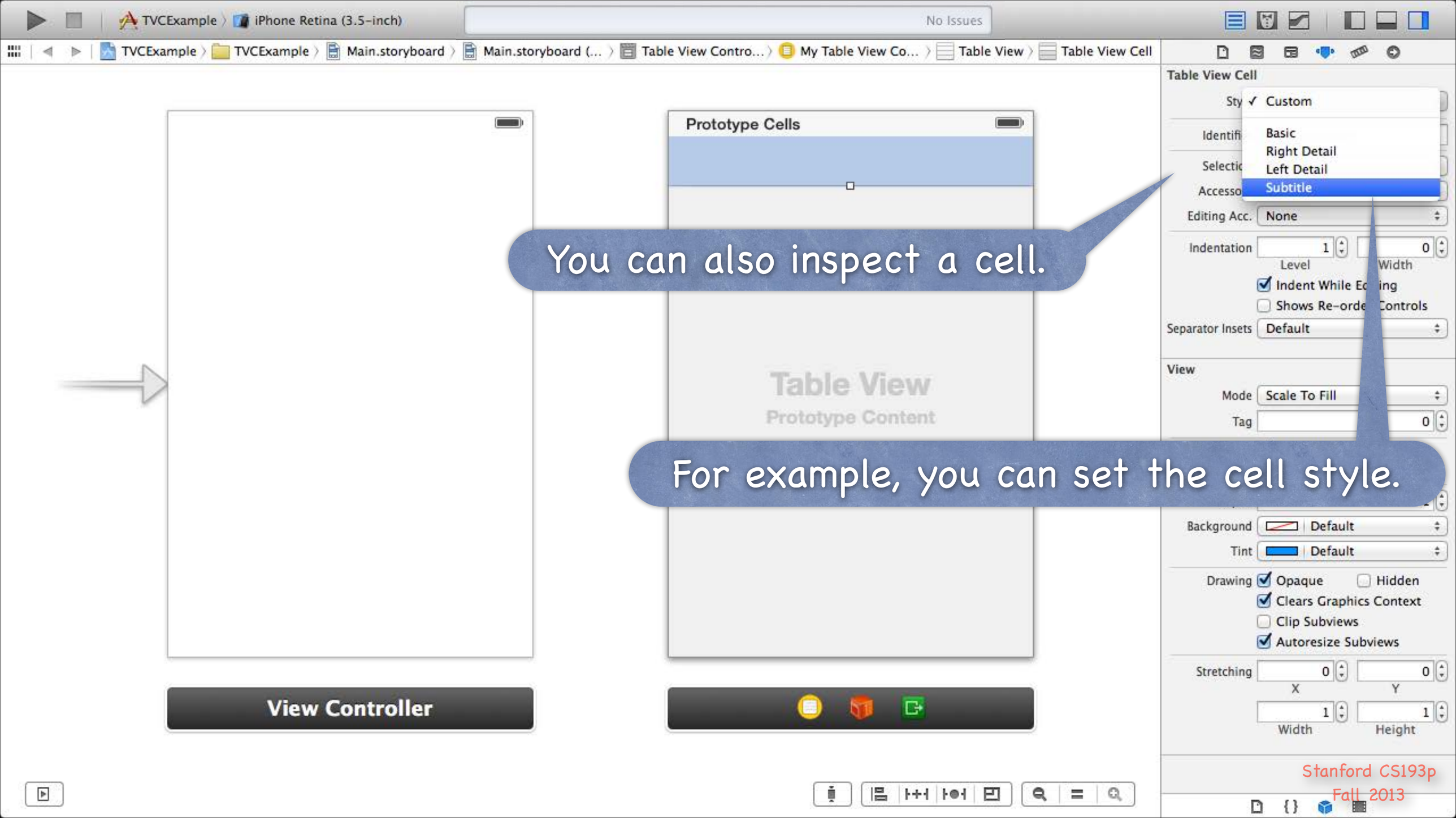

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
My Table View Controller
Table View
Table View
Dynamic Prototypes
Static Cells
Content
Sections
1
Style
Grouped
Separator
Default
Default
Separator Insets
Default
Selection
Single Selection
Editing
No Selection Dur
Editing
Show Selectio
Touch
Index Row Limit
0
Text Color
Default
Background
Defau
Scroll View
Table View
Static Content
A more interesting table,
however, is a Dynamic one ...
0
abled
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
0
X
Y
View Controller
Stanford CS193p
Fall 2013

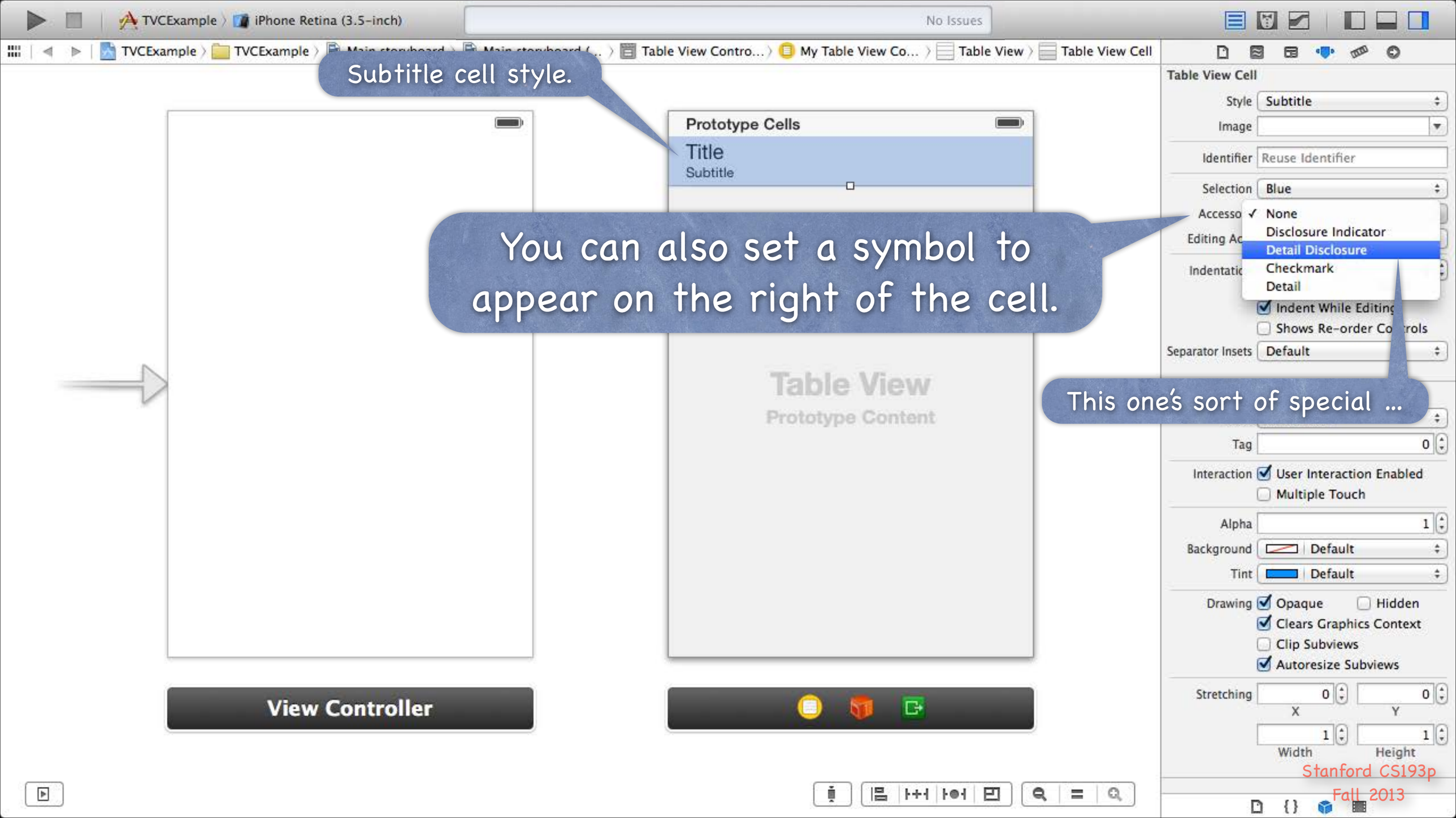

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
My Table View Controller
Table View
PROTOTYPE CELLS
Table View
Prototype Content
View Controller
Table View
Content
Dynamic Prototypes
Plain
Grouped
Separator
Default
Default
Separator Insets
Default
Selection
Single Selection
Editing
Show Selection on Touch
Index Row Limit
0
Text Color
Background
Scroll View
0
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
0
X
Y
... which we almost always use in Plain style.
Stanford CS193p
Fall 2013

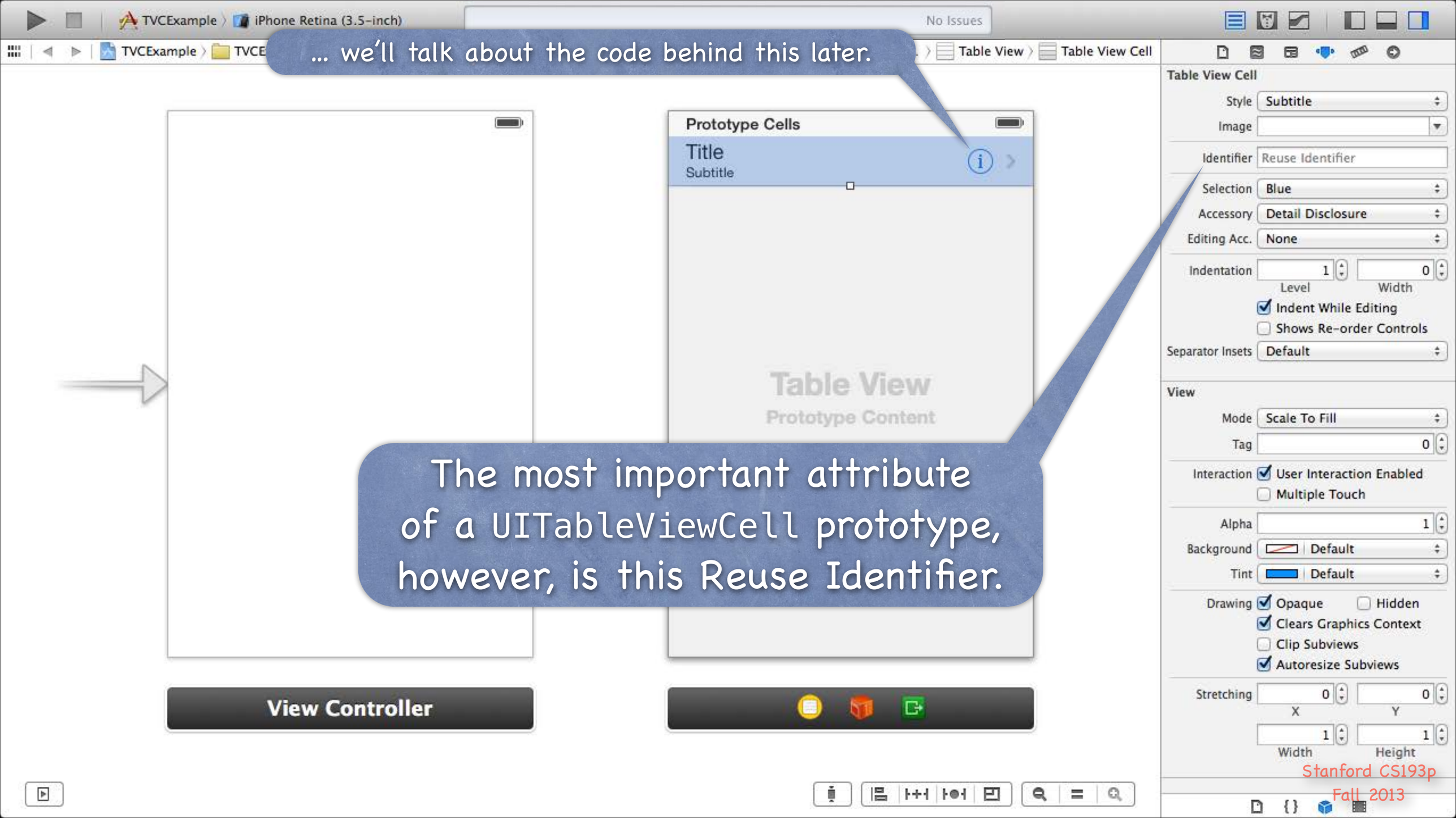

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
My Table View Controller
Table View
Prototype Cells
Table View
Prototype Content
View Controller
Table View
Content
Dynamic Prototypes
Prototype Cells
3
Style
Plain
Separator
Default
Default
Separator Insets
Default
Selection
Single Selection
Editing
No Selection During Editi
Show Selection on Tou
Index Row Limit
Text Color
Default
Background
Default
Scroll View
View
Mode
Scale To Fill
Tag
Interaction
User Interaction Ena
Multiple Touch
Alpha
Background
Default
Stretching
0
0
X
Y
These cells are now templates which will be repeated for however many rows are needed to display the data in MVC's Model.
We are allowed to have multiple, different prototype cells, but usually we only have one.
Stanford CS193p
Fall 2013

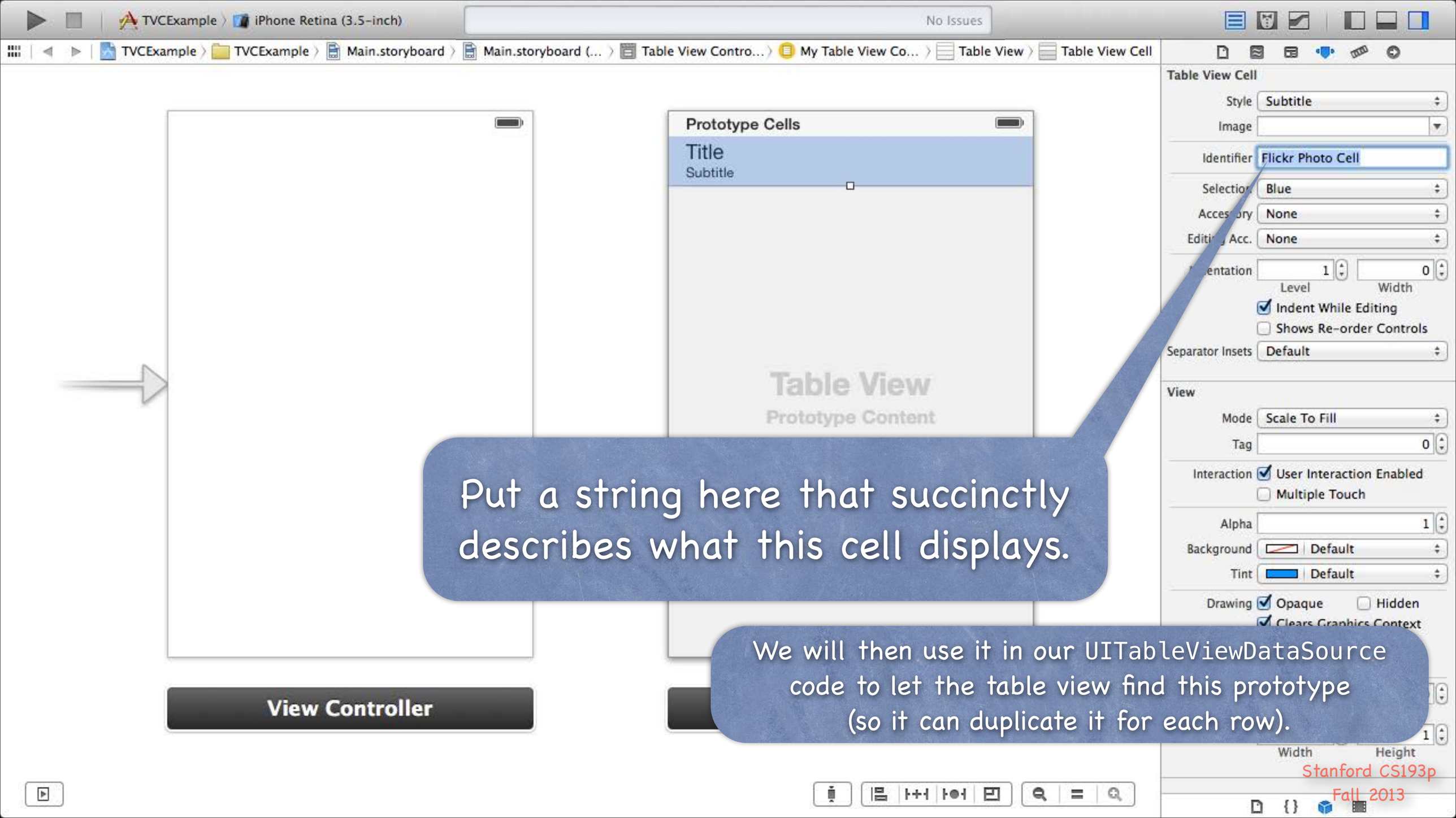

TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
My Table View Controller
Table View
Table View
Content
Dynamic Prototypes
Prototype Cells
1
Style
Plain
Separator
Default
Default
Separator Insets
Default
Selection
Single Selection
Editing
No Selection During Editing
Show Selection on Touch
Index Row Limit
0
Text Color
Default
Background
Default
Scroll View
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Stretching
0
0
X
Y
Prototype Cells
Table View
Prototype Content
View Controller
Each of these rows is a UIView.
A subclass of UITableViewCell to be exact.
If you wanted to create an outlet to something you drag into one of these prototypes, you'd have to subclass UITableViewCell, set its class in the Identity Inspector, and wire up to that.
That's a little bit advanced for us right now!
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (Base)
Table View Controller Scene
My Table View Controller
Table View
Table View
Content
Dynamic Prototypes
Prototype Cells
1
Style
Plain
Separator
Default
Default
Separator Insets
Default
Selection
Single Selection
Editing
No Selection During Editing
Show Selection on Touch
Index Row Limit
0
Text Color
Default
Background
Default
Scroll View
View
Mode
Scale To Fill
Tag
0
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
0
X
Y
Prototype Cells
Table View
Prototype Content
You can ctrl-drag from a prototype to create a segue.
That segue will happen when any cell in the table is clicked on.
We'll see how to tell which cell was clicked in prepareForSegue:sender: later.
View Controller
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (...
Table View Contro...
My Table View Co...
Table View
Table View Cell
Prototype Cells
Table View
Prototype Content
View Controller
Table View Cell
Custom
Basic
Right Detail
Left Detail
Subtitle
Editing Acc.
None
Indentation
Level
Indent While Editing
Shows Re-order Controls
Separator Insets
Default
View
Mode
Scale To Fill
Tag
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
X
Y
Width
Height
You can also inspect a cell.
For example, you can set the cell style.
Stanford CS193p
Fall 2013


Subtitle cell style.
Prototype Cells
Title
Subtitle
You can also set a symbol to appear on the right of the cell.
This one's sort of special ...
Table View
Prototype Content
View Controller
Table View Cell
Style Subtitle
Image
Identifier Reuse Identifier
Selection Blue
None
Disclosure Indicator
Detail Disclosure
Checkmark
Detail
Indent While Editing
Shows Re-order Controls
Separator Insets Default
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Tint Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching 0 0
X Y
1 1
Width Height
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
Table View
Table View Cell
... we'll talk about the code behind this later.
Prototype Cells
Title
Subtitle
Table View
Prototype Content
View Controller
Table View Cell
Style
Subtitle
Image
Identifier
Reuse Identifier
Selection
Blue
Accessory
Detail Disclosure
Editing Acc.
None
Indentation
1
Level
0
Width
Indent While Editing
Shows Re-order Controls
Separator Insets
Default
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
X
0
Y
1
Width
1
Height
The most important attribute of a UITableViewCell prototype, however, is this Reuse Identifier.
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (...
Table View Contro...
My Table View Co...
Table View
Table View Cell
Prototype Cells
Title
Subtitle
Table View
Prototype Content
View Controller
Table View Cell
Style
Subtitle
Image
Identifier
Flickr Photo Cell
Selection
Blue
Accessory
None
Editing Acc.
None
Indentation
1
0
Level
Width
Indent While Editing
Shows Re-order Controls
Separator Insets
Default
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Width
Height
Put a string here that succinctly describes what this cell displays.
We will then use it in our UITableViewDataSource code to let the table view find this prototype (so it can duplicate it for each row).
Stanford CS193p
Fall 2013

UITableView Protocols

- How do we connect to all this stuff in our code?
  - Via the UITableView's dataSource and delegate.
    - The delegate is used to control how the table is displayed.
    - The dataSource provides the data what is displayed inside the cells.
- UITableViewController
  - Automatically sets itself as its UITableView's delegate & dataSource.

  - Also has a property pointing to its UITableView:

```
@property (nonatomic, strong) UITableView *tableView;
```

    (this property is actually == self.view in UITableViewController!)

UITableViewDataSource

- Important `dataSource` methods

  We have to implement these 3 to be a "dynamic" (arbitrary number of rows) table …

  How many `sections` in the table?

  How many `rows` in each `section`?

  Give me a `UITableViewCell` to use to draw each cell at a given row in a given section.

  Let's cover the last one first (since the first two are very straightforward) …

UITableViewDataSource

- How do we control what is drawn in each cell in a dynamic table?

  Each row is drawn by its own instance of `UITableViewCell` (a `UIView` subclass).

  Here is the `UITableViewDataSource` method to get that cell for a given row in a section …

```
- (UITableViewCell *)tableView:(UITableView *)sender
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{


}
```

In a static table, you do not need to implement this method (though you can if you want to ignore what's in the storyboard).

UITableViewDataSource

- How do we control what is drawn in each cell in a dynamic table?

  Each row is drawn by its own instance of `UITableViewCell` (a `UIView` subclass).

  Here is the `UITableViewDataSource` method to get that cell for a given row in a section …

```
- (UITableViewCell *)tableView:(UITableView *)sender
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{


}
```

`NSIndexPath` is just an object with two important properties for use with `UITableView`: `row` and `section`.

UITableViewDataSource

- How do we control what is drawn in each cell in a dynamic table?

  Each row is drawn by its own instance of `UITableViewCell` (a `UIView` subclass).

  Here is the `UITableViewDataSource` method to get that cell for a given row in a section …

```
- (UITableViewCell *)tableView:(UITableView *)sender
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    // get a cell to use (instance of UITableViewCell)
    // set @propertys on the cell to prepare it to display
}
```

UITableViewDataSource

- How do we control what is drawn in each cell in a dynamic table?

  Each row is drawn by its own instance of `UITableViewCell` (a `UIView` subclass).

  Here is the `UITableViewDataSource` method to get that cell for a given row in a section …

```
- (UITableViewCell *)tableView:(UITableView *)sender
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
   UITableViewCell *cell;
   cell = [self.tableView dequeueReusableCellWithIdentifier:@“Flickr Photo Cell”
                                                forIndexPath:indexPath];
   // set @propertys on the cell to prepare it to display

}
```

This MUST match what is in your storyboard if you want to use the prototype you defined there!

UITableViewDataSource

- How do we control what is drawn in each cell in a dynamic table?

  Each row is drawn by its own instance of `UITableViewCell` (a `UIView` subclass).

  Here is the `UITableViewDataSource` method to get that cell for a given row in a section …

```
- (UITableViewCell *)tableView:(UITableView *)sender
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
   UITableViewCell *cell;
   cell = [self.tableView dequeueReusableCellWithIdentifier:@"Flickr Photo Cell"
                                                forIndexPath:indexPath];
   // set @propertys on the cell to prepare it to display

}
```

The cells in the table are actually reused.
When one goes off-screen, it gets put into a "reuse pool."
The next time a cell is needed, one is grabbed from the reuse pool if available.
If none is available, one will be put into the reuse pool if there's a prototype in the storyboard.
Otherwise this `dequeue` method will return `nil`.

UITableViewDataSource

- How do we control what is drawn in each cell in a dynamic table?

  Each row is drawn by its own instance of `UITableViewCell` (a `UIView` subclass).

  Here is the `UITableViewDataSource` method to get that cell for a given row in a section …

```
- (UITableViewCell *)tableView:(UITableView *)sender
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell;
    cell = [self.tableView dequeueReusableCellWithIdentifier:@“Flickr Photo Cell”
                                                forIndexPath:indexPath];
    cell.textLabel.text = [self getMyTitleForRow:indexPath.row inSection:indexPath.section];
    return cell;
}
```

There are obviously other things you can do in the cell besides setting its text (detail text, image, checkmark, etc.).

UITableViewDataSource

- How do we control what is drawn in each cell in a dynamic table?

  Each row is drawn by its own instance of `UITableViewCell` (a `UIView` subclass).

  Here is the `UITableViewDataSource` method to get that cell for a given row in a section …

```
- (UITableViewCell *)tableView:(UITableView *)sender
        cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell;
    cell = [self.tableView dequeueReusableCellWithIdentifier:@“Flickr Photo Cell”
                                                forIndexPath:indexPath];
    cell.textLabel.text = [self getMyTitleForRow:indexPath.row inSection:indexPath.section];
    return cell;
}
```

See how we are using `indexPath.section` and `indexPath.row` to get Model information to set up this cell.

UITableViewDataSource

- How does a dynamic table know how many rows there are?
  - And how many sections, too, of course?
  - Via these two UITableViewDataSource methods …
  - `- (NSInteger)numberOfSectionsInTableView:(UITableView *)sender;`
  - `- (NSInteger)tableView:(UITableView *)sender numberOfRowsInSection:(NSInteger)section;`
- Number of sections is 1 by default
  - In other words, if you don't implement `numberOfSectionsInTableView:`, it will be 1.
- No default for `tableView:numberOfRowsInSection:`
  - This is a required method in this protocol (as is `tableView:cellForRowAtIndexPath:`).
- What about a static table?
  - Do not implement these `dataSource` methods for a static table.
  - `UITableViewController` will take care of that for you.

UITableViewDataSource

- There are a number of other methods in this protocol
  - But we're not going to cover them today.
  - They are mostly about getting the headers and footers for sections.
  - And about keeping the Model in sync with table edits (moving/deleting/inserting rows).

UITableViewDelegate

- All of the above was the UITableView's dataSource
  - But UITableView has another protocol-driven delegate called its delegate.
- The delegate controls how the UITableView is displayed
  - Not what it displays (that's the dataSource's job).
- Common for dataSource and delegate to be the same object
  - Usually the Controller of the MVC in which the UITableView is part of the View.
  - This is the way UITableViewController sets it up for you.
- The delegate also lets you observe what the table view is doing
  - The classic "will/did" sorts of things.
  - An important one is "user did select a row."
  - Usually we don't need this because we simply segue when a row is touched.
  - But there are some occasions where it will be useful ...

UITableView "Target/Action"

- UITableViewDelegate method sent when row is selected

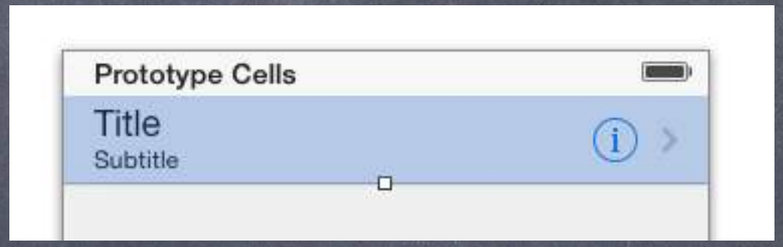
  This is sort of like "table view target/action" (only needed if you're not segueing, of course).

  On the iPad, where the table might be on screen with what it updates, you might need this.

```
- (void)tableView:(UITableView *)sender didSelectRowAtIndexPath:(NSIndexPath *)path
{
    // go do something based on information about my Model
    //   corresponding to indexPath.row in indexPath.section
}
```

UITableView Detail Disclosure

- Remember the little circled i?
  Clicking on this will not segue.

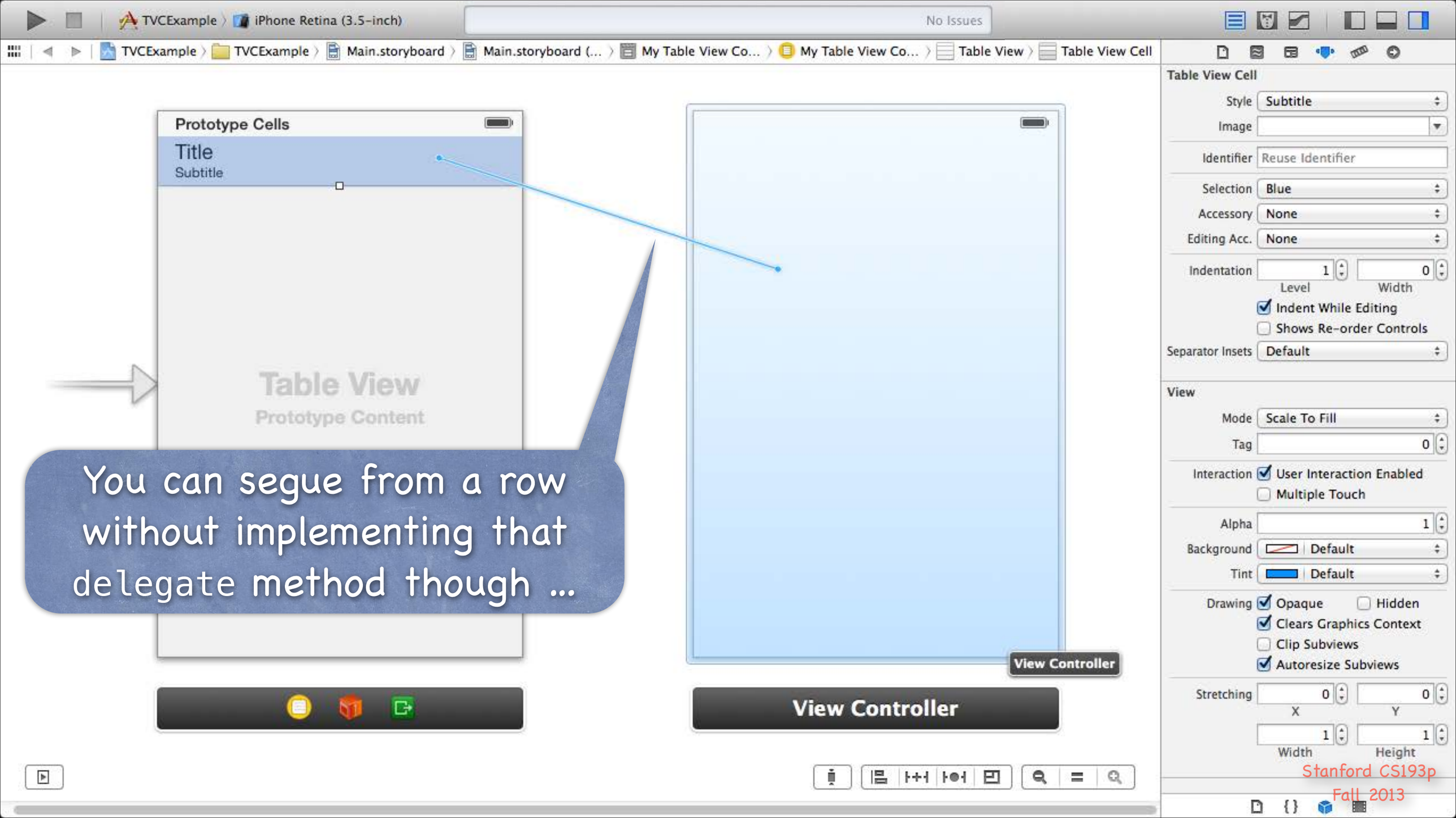


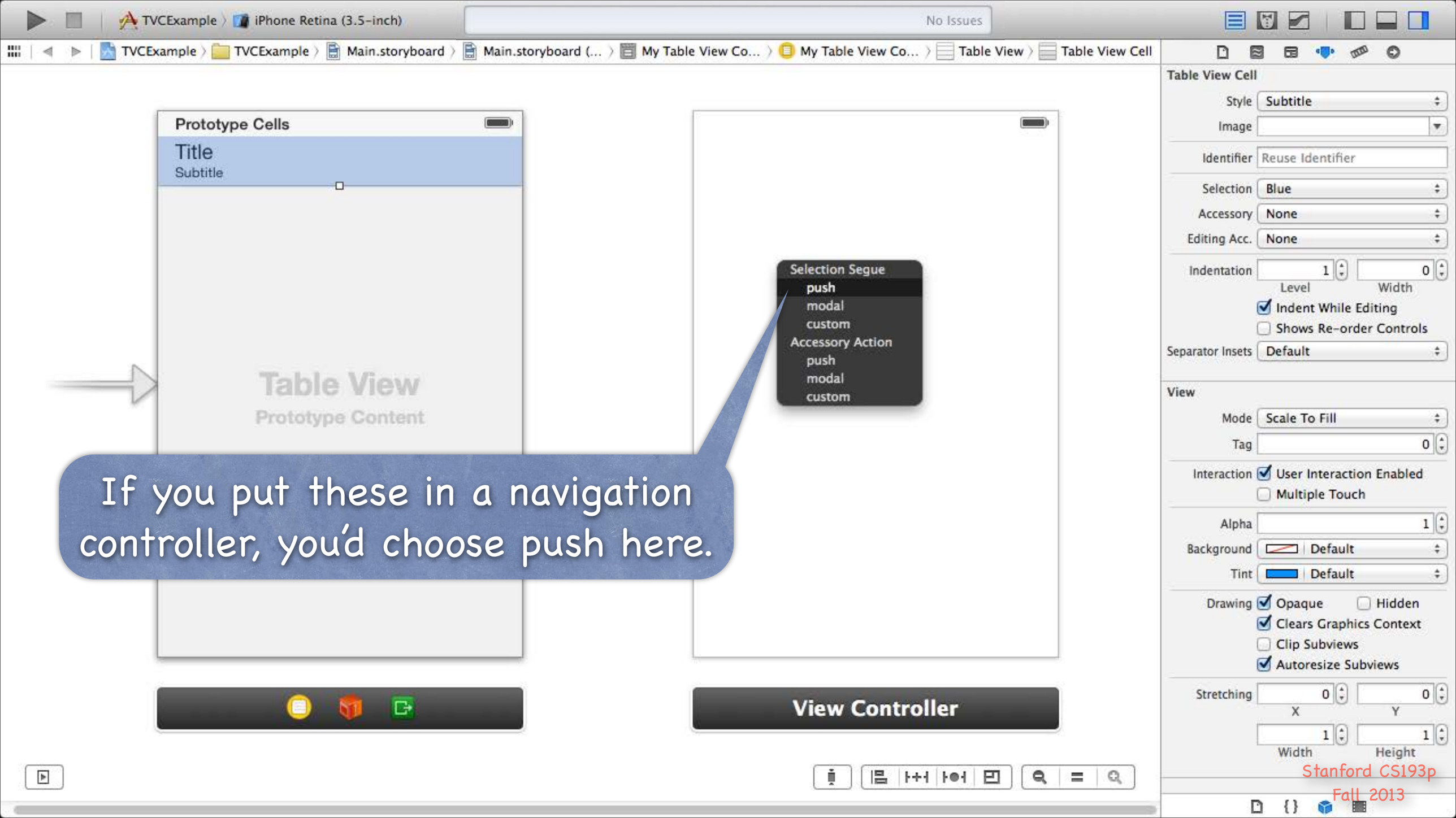
Instead it will invoke this method in the UITableViewDelegate protocol …

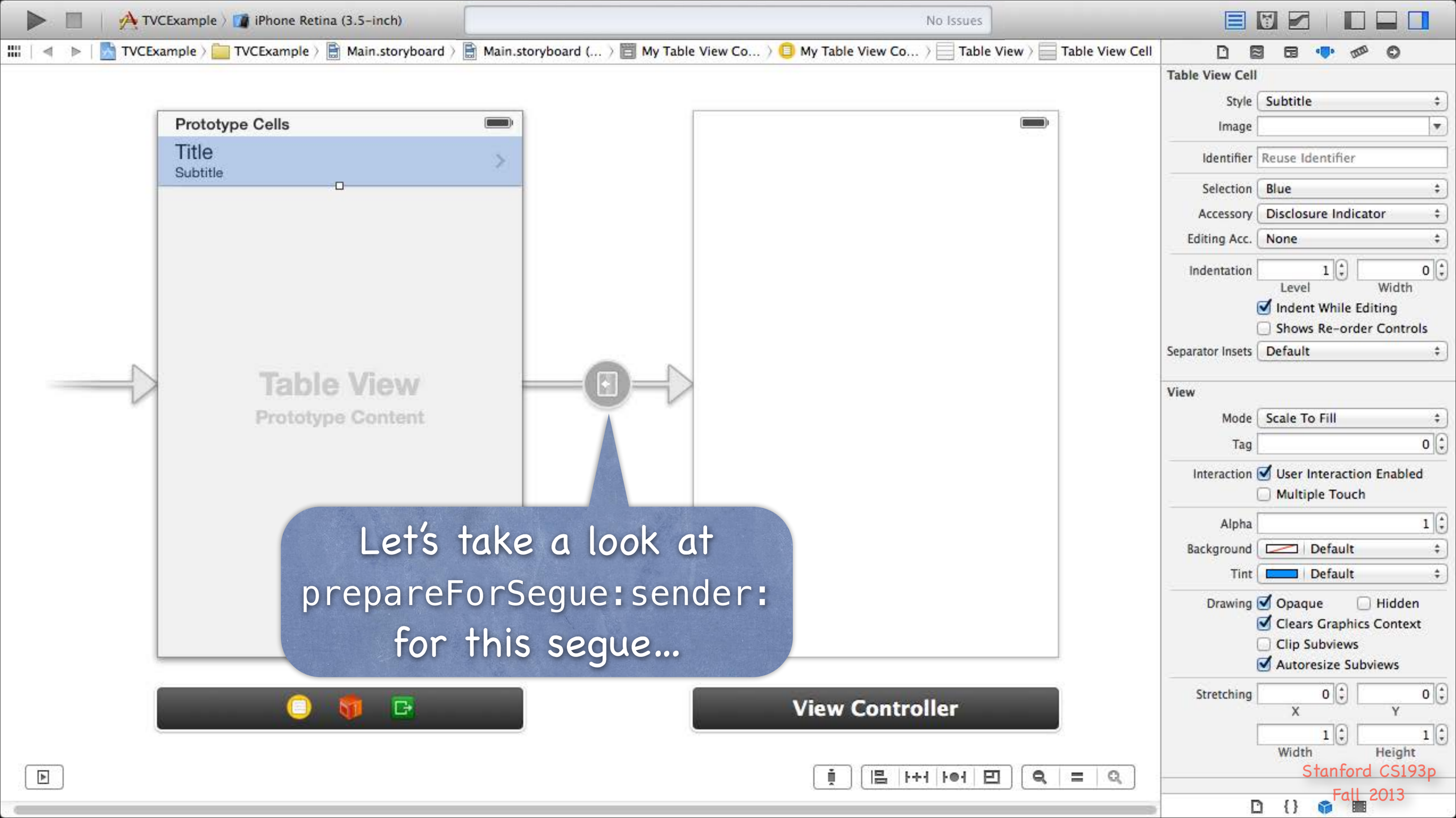
```
- (void)tableView:(UITableView *)sender
         accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath
{
    // Do something related to the row at indexPath,
    //  but not the primary action associated with touching the row
}
```

UITableViewDelegate

- Lots and lots of other `delegate` methods
  - `will`/`did` methods for both selecting and deselecting rows.
  - Providing `UIView` objects to draw section headers and footers.
  - Handling editing rows (moving them around with touch gestures).
  - `willBegin`/`didEnd` notifications for editing (i.e. removing/moving) rows.
  - Copying/pasting rows.


TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (...
My Table View Co...
My Table View Co...
Table View
Table View Cell
Prototype Cells
Title
Subtitle
Table View
Prototype Content
You can segue from a row without implementing that delegate method though ...
View Controller
View Controller
Table View Cell
Style
Subtitle
Image
Identifier
Reuse Identifier
Selection
Blue
Accessory
None
Editing Acc.
None
Indentation
1
Level
0
Width
Indent While Editing
Shows Re-order Controls
Separator Insets
Default
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
X
0
Y
1
Width
1
Height
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (...
My Table View Co...
My Table View Co...
Table View
Table View Cell
Prototype Cells
Title
Subtitle
Table View
Prototype Content
Selection Segue
push
modal
custom
Accessory Action
push
modal
custom
If you put these in a navigation controller, you'd choose push here.
View Controller
Table View Cell
Style
Subtitle
Image
Identifier
Reuse Identifier
Selection
Blue
Accessory
None
Editing Acc.
None
Indentation
1
Level
0
Width
Indent While Editing
Shows Re-order Controls
Separator Insets
Default
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
X
0
Y
1
Width
1
Height
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
TVCExample
TVCExample
Main.storyboard
Main.storyboard (...
My Table View Co...
My Table View Co...
Table View
Table View Cell
Prototype Cells
Title
Subtitle
Table View
Prototype Content
Let's take a look at prepareForSegue:sender: for this segue...
View Controller
Table View Cell
Style
Subtitle
Image
Identifier
Reuse Identifier
Selection
Blue
Accessory
Disclosure Indicator
Editing Acc.
None
Indentation
1
Level
0
Width
Indent While Editing
Shows Re-order Controls
Separator Insets
Default
View
Mode
Scale To Fill
Tag
0
Interaction
User Interaction Enabled
Multiple Touch
Alpha
1
Background
Default
Tint
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
0
X
0
Y
1
Width
1
Height
Stanford CS193p
Fall 2013

UITableView Segue

- The sender of prepareForSegue:sender: is the UITableViewCell

  Use the important method indexPathForCell: to find out the indexPath of the row that's segueing.

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    NSIndexPath *indexPath = [self.tableView indexPathForCell:sender];
    // prepare segue.destinationController to display based on information
    //   about my Model corresponding to indexPath.row in indexPath.section
}
```

UITableView Spinner

- UITableViewController has an "activity indicator" built in

  You get it via this property in UITableViewController …

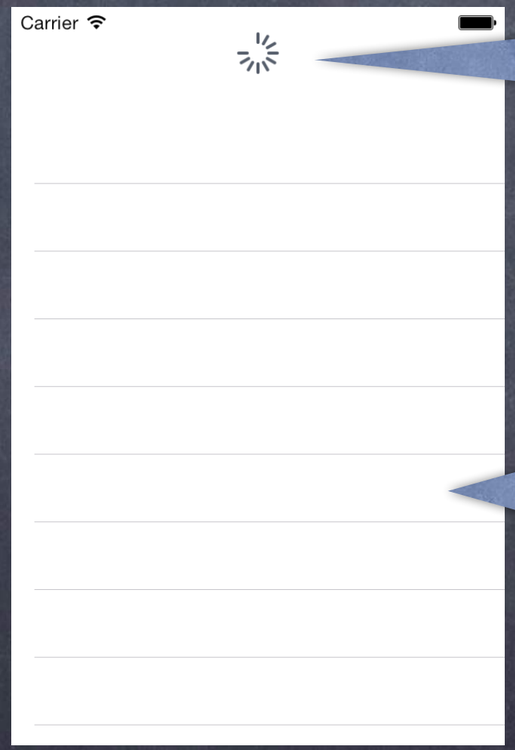
```
@property (strong) UIRefreshControl *refreshControl;
```

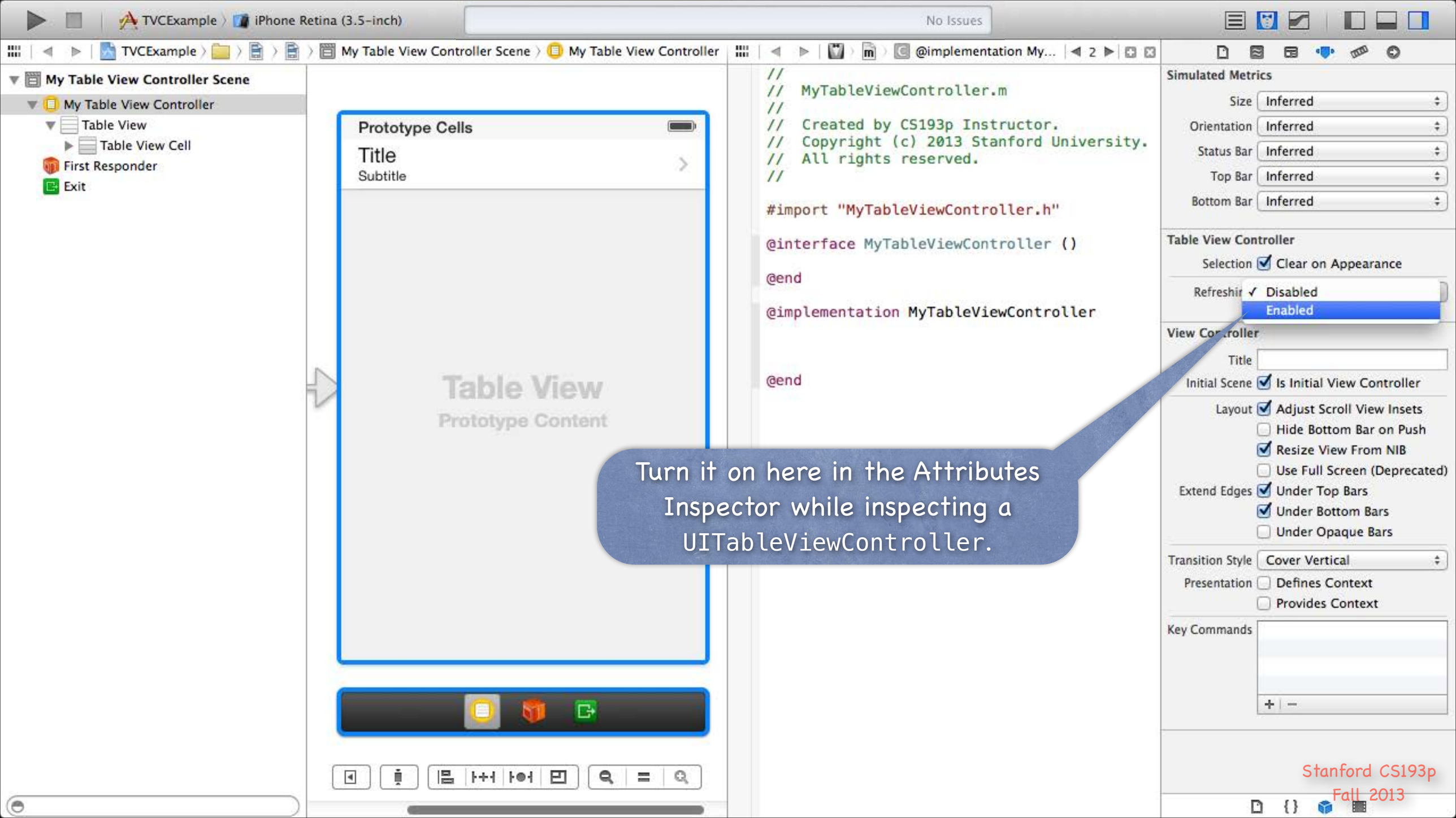
  Start it with …

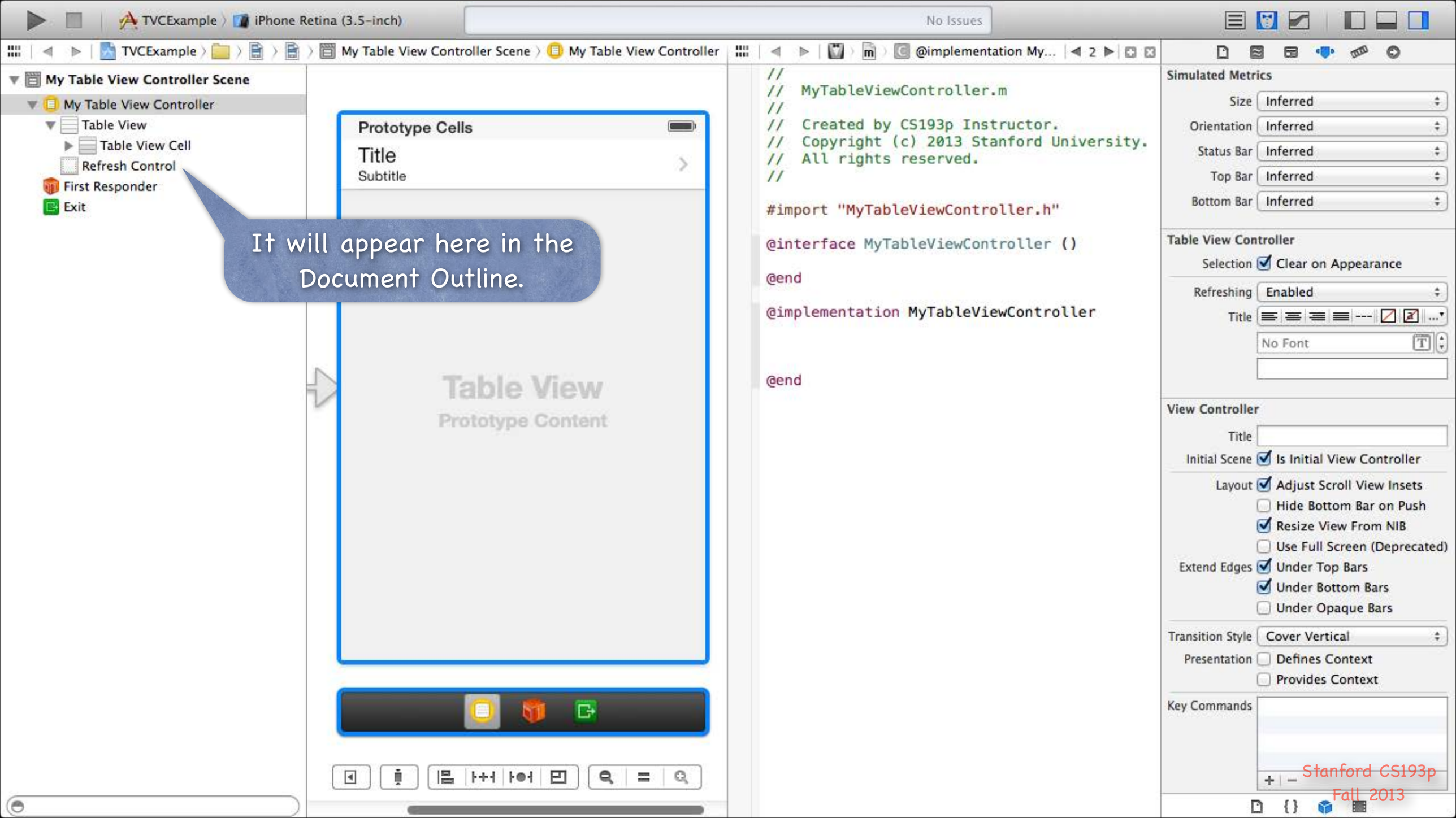
```
- (void)beginRefreshing;
```

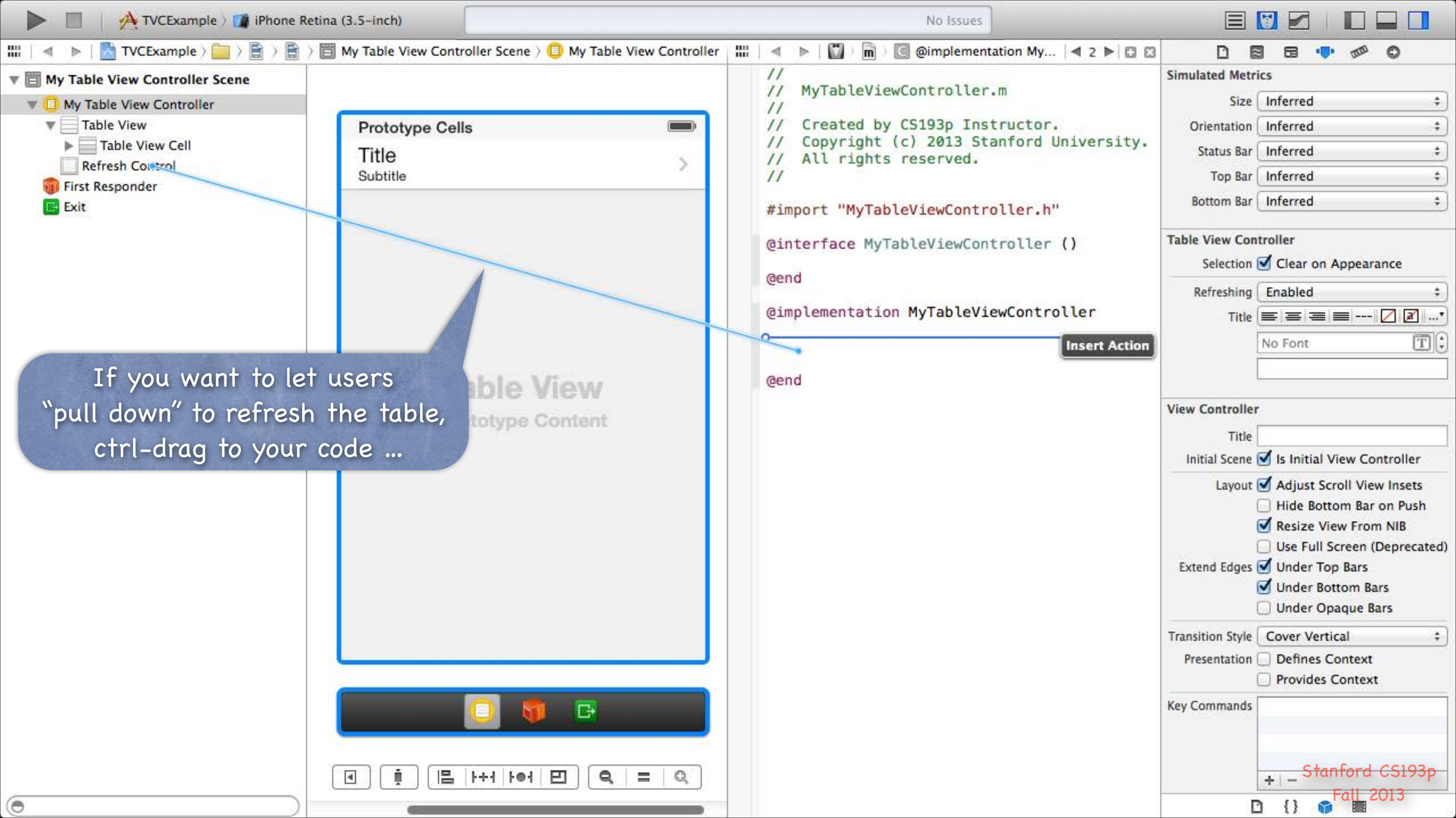
  Stop it with …

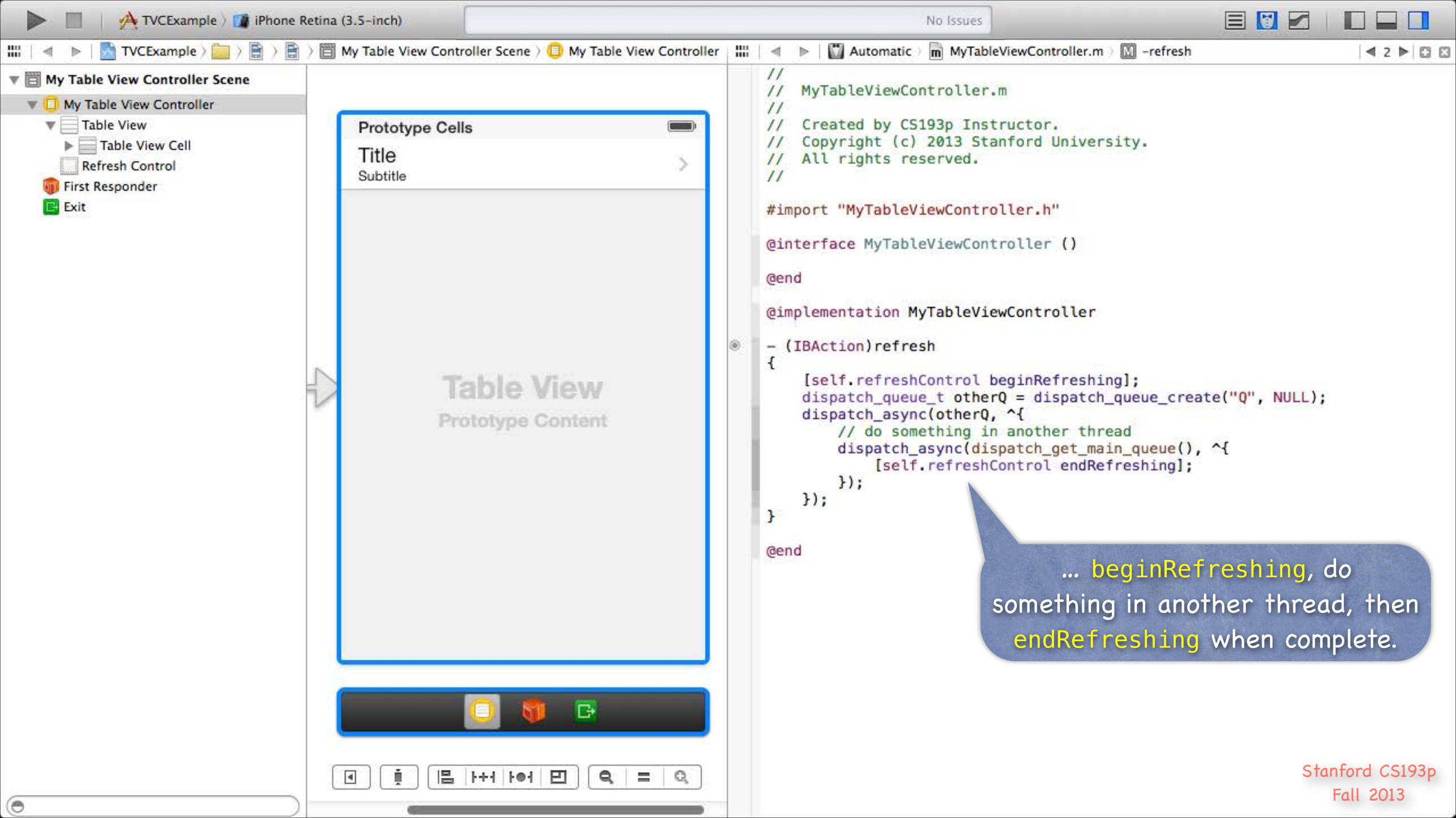
```
- (void)endRefreshing;
```




TVCExample
iPhone Retina (3.5-inch)
No Issues
My Table View Controller Scene
My Table View Controller
Table View
Table View Cell
First Responder
Exit
Prototype Cells
Title
Subtitle
Table View
Prototype Content
//
// MyTableViewController.m
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "MyTableViewController.h"
@interface MyTableViewController ()
@end
@implementation MyTableViewController
@end
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
Table View Controller
Selection Clear on Appearance
Disabled
Enabled
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands
Turn it on here in the Attributes Inspector while inspecting a UITableViewController.
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
My Table View Controller Scene
My Table View Controller
Table View
Table View Cell
Refresh Control
First Responder
Exit
It will appear here in the Document Outline.
Prototype Cells
Title
Subtitle
Table View
Prototype Content
@implementation My...
//
// MyTableViewController.m
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "MyTableViewController.h"
@interface MyTableViewController ()
@end
@implementation MyTableViewController
@end
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
Table View Controller
Selection Clear on Appearance
Refreshing Enabled
Title
No Font
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
My Table View Controller Scene
My Table View Controller
Table View
Table View Cell
Refresh Control
First Responder
Exit
Prototype Cells
Title
Subtitle
If you want to let users "pull down" to refresh the table, ctrl-drag to your code ...
//
// MyTableViewController.m
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "MyTableViewController.h"
@interface MyTableViewController ()
@end
@implementation MyTableViewController
Insert Action
@end
Simulated Metrics
Size Inferred
Orientation Inferred
Status Bar Inferred
Top Bar Inferred
Bottom Bar Inferred
Table View Controller
Selection Clear on Appearance
Refreshing Enabled
Title
No Font
View Controller
Title
Initial Scene Is Initial View Controller
Layout Adjust Scroll View Insets
Hide Bottom Bar on Push
Resize View From NIB
Use Full Screen (Deprecated)
Extend Edges Under Top Bars
Under Bottom Bars
Under Opaque Bars
Transition Style Cover Vertical
Presentation Defines Context
Provides Context
Key Commands
Stanford CS193p
Fall 2013


TVCExample
iPhone Retina (3.5-inch)
No Issues
My Table View Controller Scene
My Table View Controller
Table View
Table View Cell
Refresh Control
First Responder
Exit
Prototype Cells
Title
Subtitle
Table View
Prototype Content
Automatic
MyTableViewController.m
-refresh
//
// MyTableViewController.m
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University.
// All rights reserved.
//
#import "MyTableViewController.h"
@interface MyTableViewController ()
@end
@implementation MyTableViewController
- (IBAction)refresh
{
[self.refreshControl beginRefreshing];
dispatch_queue_t otherQ = dispatch_queue_create("Q", NULL);
dispatch_async(otherQ, ^{
// do something in another thread
dispatch_async(dispatch_get_main_queue(), ^{
[self.refreshControl endRefreshing];
});
});
}
@end
… beginRefreshing, do something in another thread, then endRefreshing when complete.
Stanford CS193p
Fall 2013

UITableView

- What if your Model changes?

  `- (void)reloadData;`

  Causes the table view to call `numberOfSectionsInTableView:` and `numberOfRowsInSection:` all over again and then `cellForRowAtIndexPath:` on each visible cell.

  Relatively heavyweight, but if your entire data structure changes, that's what you need.

  If only part of your Model changes, there are lighter-weight reloaders, for example …

```
- (void)reloadRowsAtIndexPaths:(NSArray *)indexPaths
                withRowAnimation:(UITableViewRowAnimation)animationStyle;
```

- There are dozens of other methods in `UITableView`

  Setting headers and footers for the entire table.

  Controlling the look (separator style and color, default row height, etc.).

  Getting cell information (cell for index path, index path for cell, visible cells, etc.).

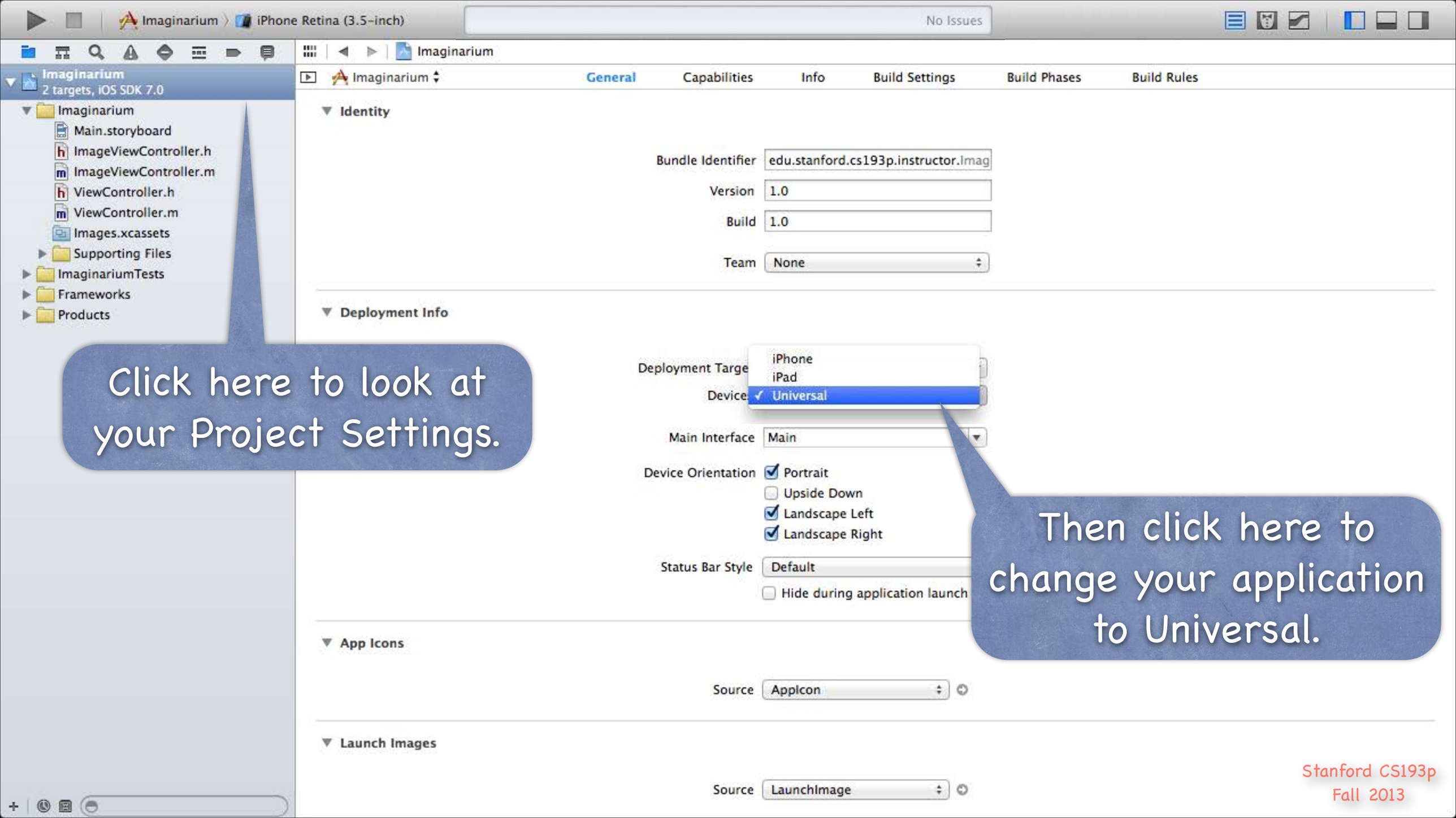
  Scrolling to a row.

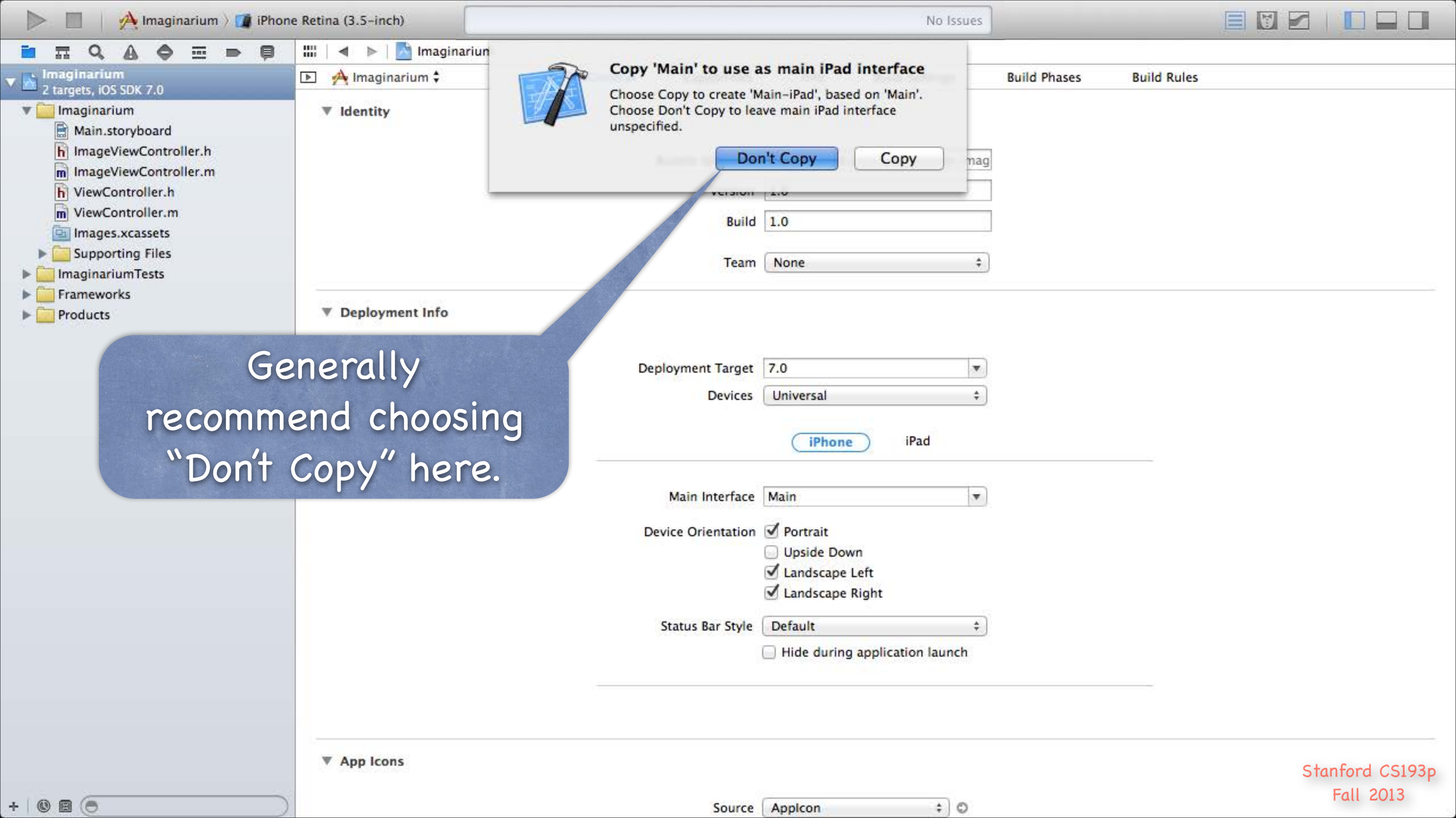
  Selection management (allows multiple selection, getting the selected row, etc.).

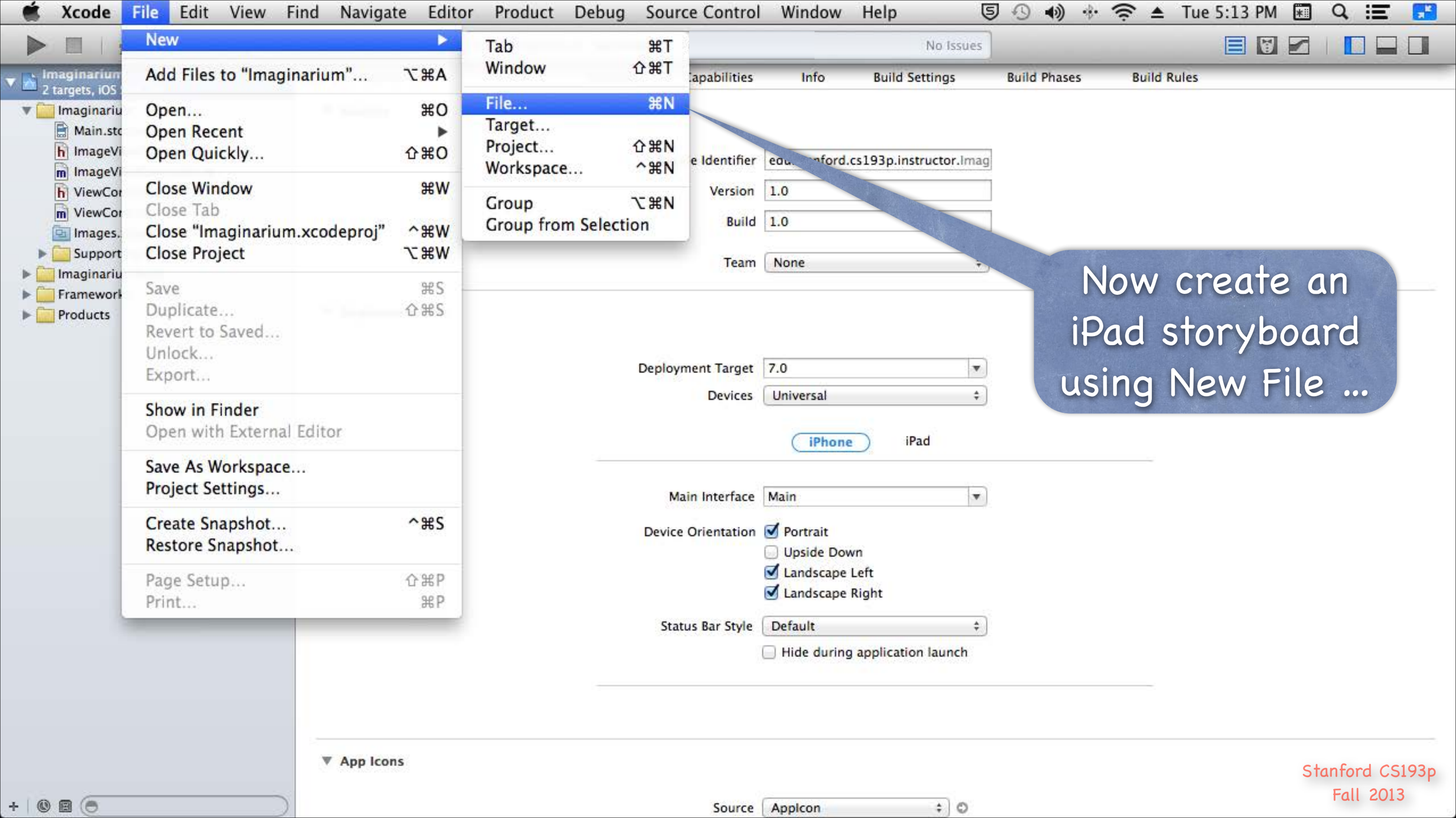
  Moving, inserting and deleting rows, etc.

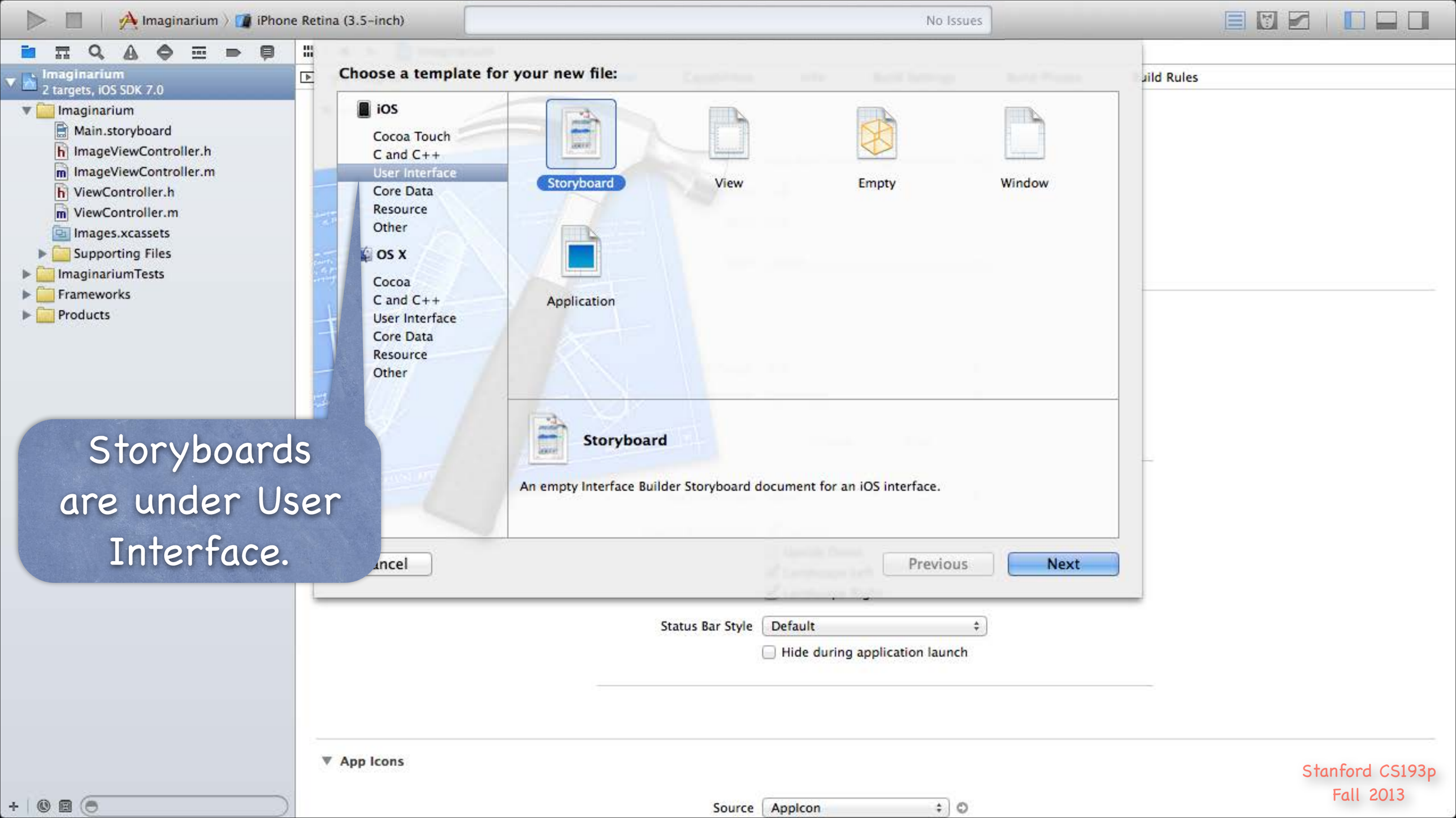
Universal Applications

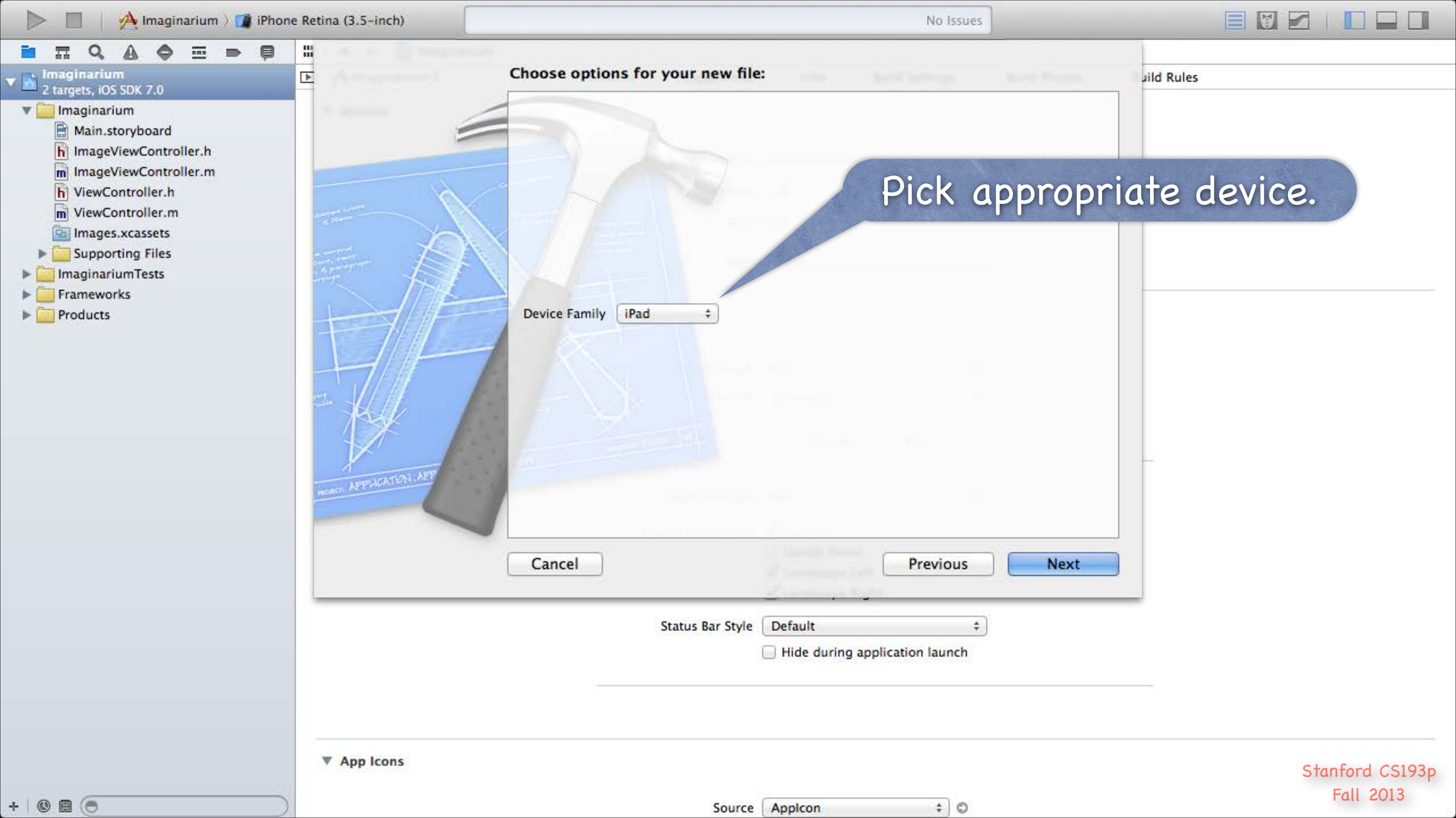
- A "Universal" Application will run on both iPhone and iPad
  - It might look different on each.
  - But it's a single binary image (i.e. it's one app, not two).
  - Two different storyboards.
- How to create one
  - When you create the project, pick Universal instead of iPhone or iPad.
  - If you have an existing iPhone- or iPad-only project, you must edit your Project Settings ...

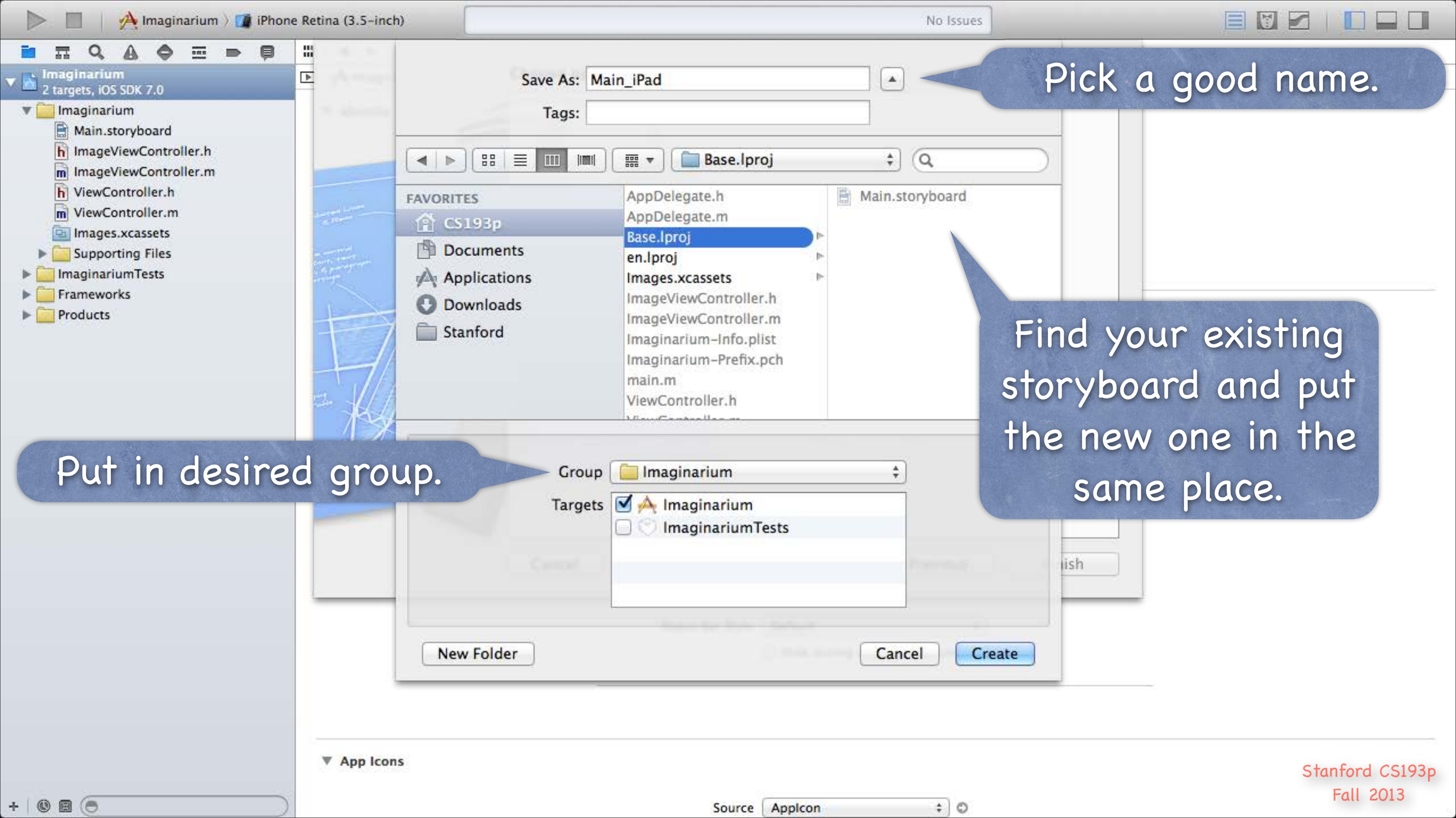

Imaginarium
iPhone Retina (3.5-inch)
No Issues
Imaginarium
2 targets, iOS SDK 7.0
Imaginarium
Main.storyboard
ImageViewController.h
ImageViewController.m
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
ImaginariumTests
Frameworks
Products
Imaginarium
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Identity
Bundle Identifier
edu.stanford.cs193p.instructor.Imag
Version
1.0
Build
1.0
Team
None
Deployment Info
Deployment Targe
Device
iPhone
iPad
Universal
Main Interface
Main
Device Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style
Default
Hide during application launch
App Icons
Source
AppIcon
Launch Images
Source
LaunchImage
Click here to look at your Project Settings.
Then click here to change your application to Universal.
Stanford CS193p
Fall 2013

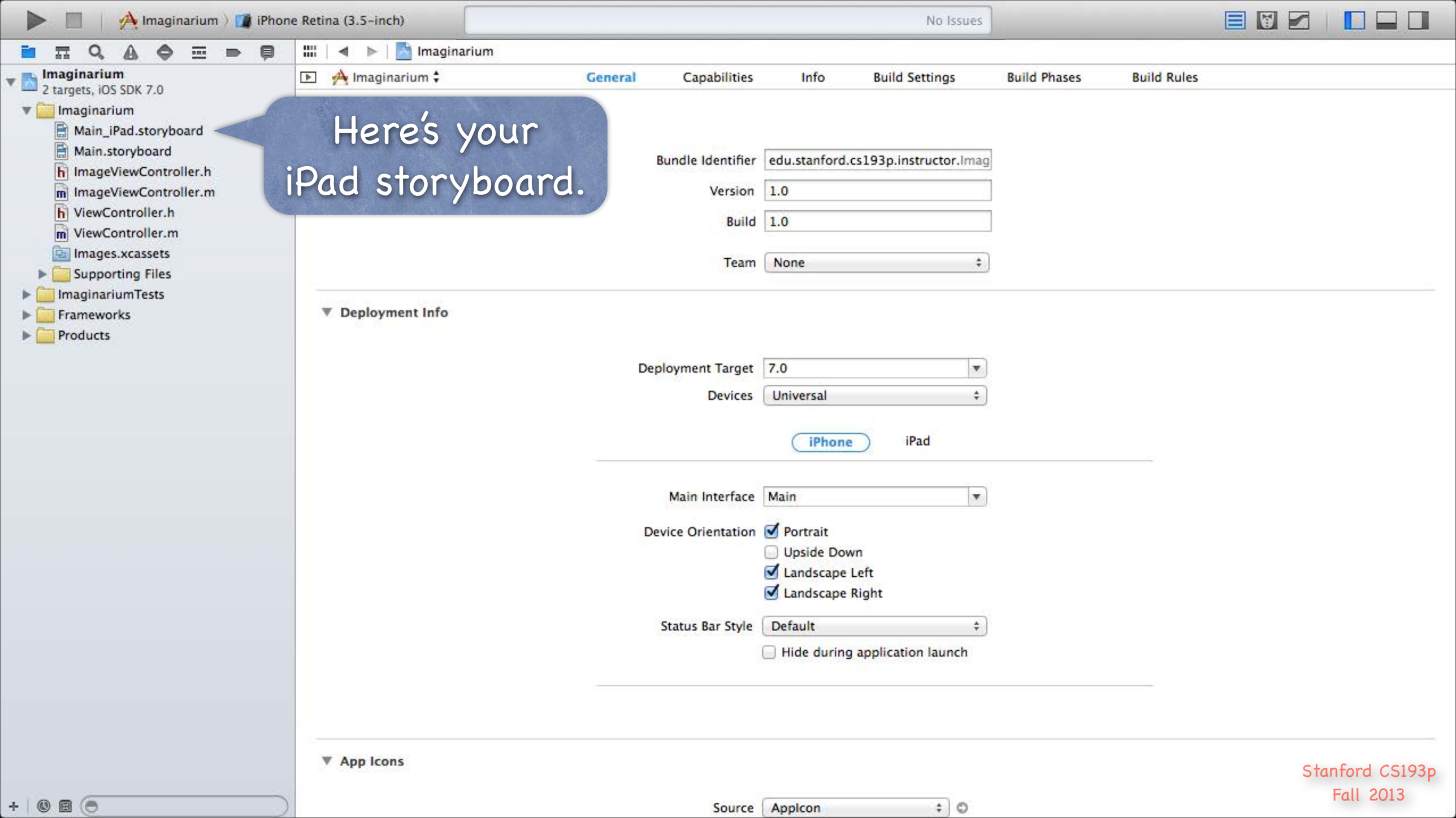

Imaginarium
iPhone Retina (3.5-inch)
No Issues
Imaginarium
2 targets, iOS SDK 7.0
Imaginarium
Main.storyboard
ImageViewController.h
ImageViewController.m
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
ImaginariumTests
Frameworks
Products
Copy 'Main' to use as main iPad interface
Choose Copy to create 'Main-iPad', based on 'Main'.
Choose Don't Copy to leave main iPad interface unspecified.
Don't Copy
Copy
Build Phases
Build Rules
Identity
Build 1.0
Team None
Deployment Info
Deployment Target 7.0
Devices Universal
iPhone
iPad
Main Interface Main
Device Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style Default
Hide during application launch
App Icons
Source AppIcon
Generally recommend choosing "Don't Copy" here.
Stanford CS193p
Fall 2013

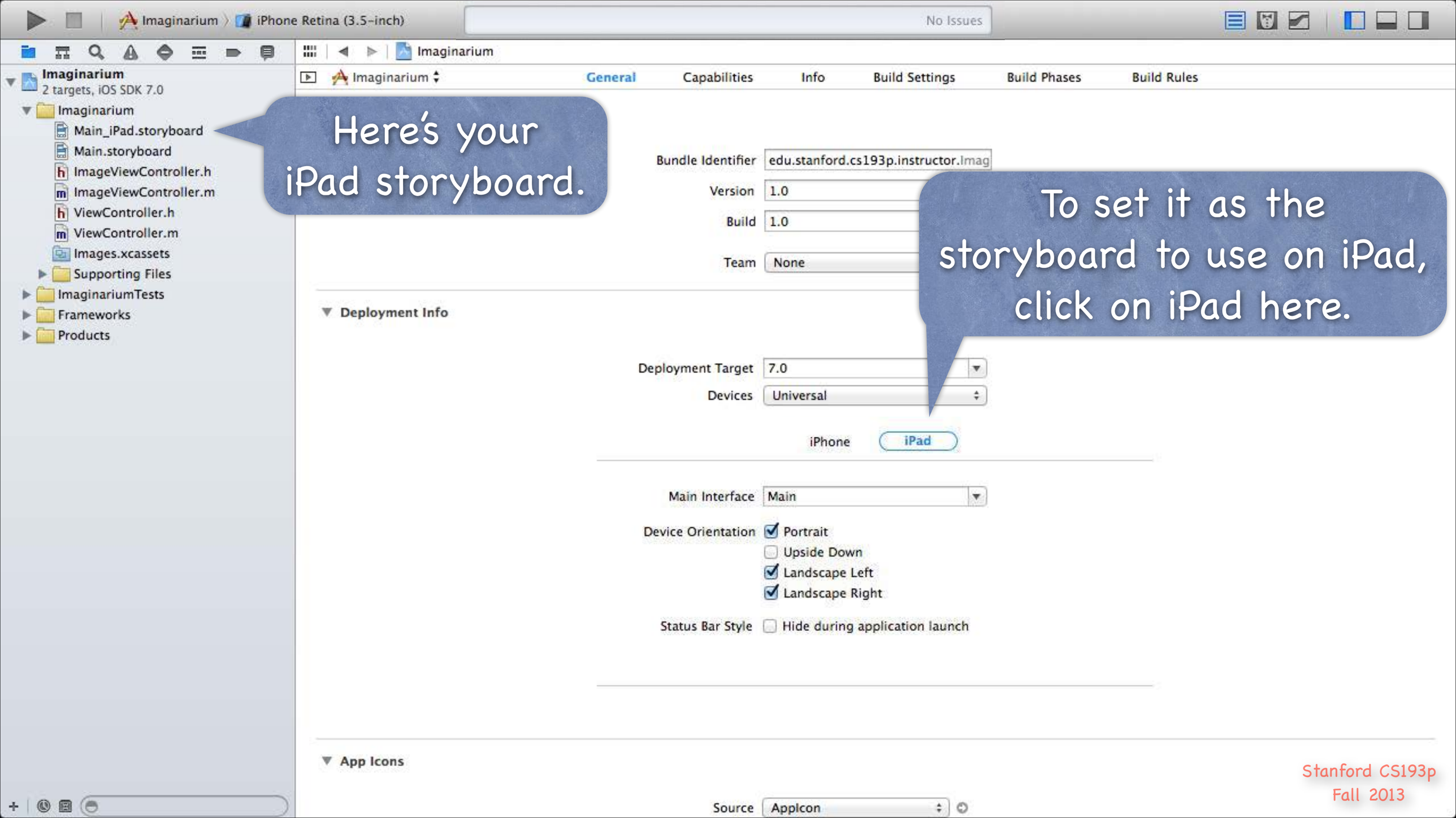

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Tue 5:13 PM
New
Add Files to "Imaginarium"... ⌥⌘A
Open... ⌘O
Open Recent
Open Quickly... ⇧⌘O
Close Window ⌘W
Close Tab
Close "Imaginarium.xcodeproj" ^⌘W
Close Project ⌥⌘W
Save ⌘S
Duplicate... ⇧⌘S
Revert to Saved...
Unlock...
Export...
Show in Finder
Open with External Editor
Save As Workspace...
Project Settings...
Create Snapshot... ^⌘S
Restore Snapshot...
Page Setup... ⇧⌘P
Print... ⌘P
Tab ⌘T
Window ⇧⌘T
File... ⌘N
Target...
Project... ⇧⌘N
Workspace... ^⌘N
Group ⌥⌘N
Group from Selection
No Issues
Capabilities
Info
Build Settings
Build Phases
Build Rules
Identifier
Version 1.0
Build 1.0
Team None
Deployment Target 7.0
Devices Universal
iPhone
iPad
Main Interface Main
Device Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style Default
Hide during application launch
App Icons
Source AppIcon
Now create an iPad storyboard using New File ...
Stanford CS193p
Fall 2013

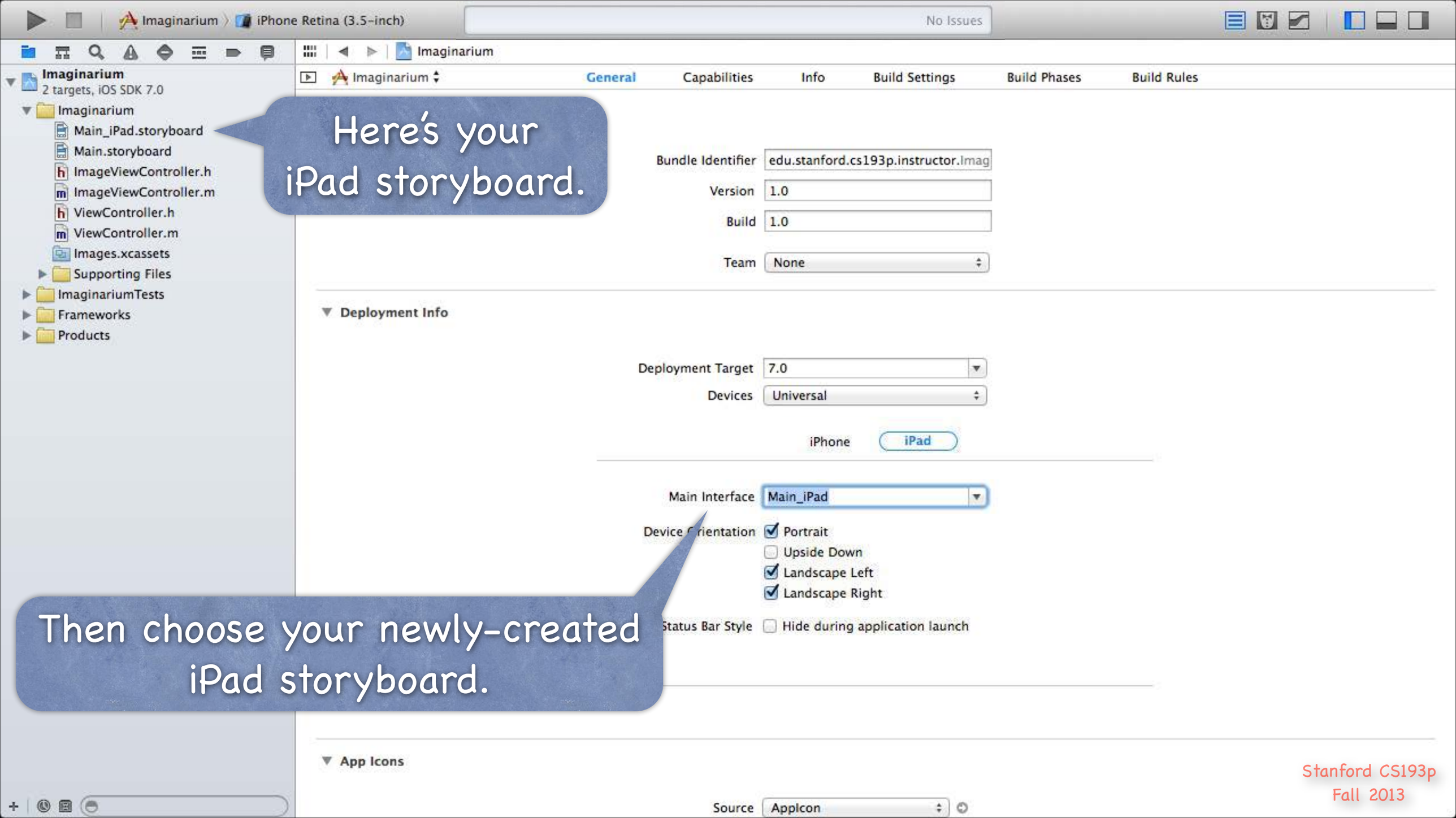

Imaginarium
iPhone Retina (3.5-inch)
No Issues
Imaginarium
2 targets, iOS SDK 7.0
Imaginarium
Main.storyboard
ImageViewController.h
ImageViewController.m
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
ImaginariumTests
Frameworks
Products
Choose a template for your new file:
iOS
Cocoa Touch
C and C++
User Interface
Core Data
Resource
Other
OS X
Cocoa
C and C++
User Interface
Core Data
Resource
Other
Storyboard
View
Empty
Window
Application
Storyboard
An empty Interface Builder Storyboard document for an iOS interface.
Cancel
Previous
Next
Storyboards are under User Interface.
Status Bar Style
Default
Hide during application launch
App Icons
Source
AppIcon
Stanford CS193p
Fall 2013


Imaginarium
iPhone Retina (3.5-inch)
No Issues
Imaginarium
2 targets, iOS SDK 7.0
Imaginarium
Main.storyboard
ImageViewController.h
ImageViewController.m
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
ImaginariumTests
Frameworks
Products
Choose options for your new file:
Device Family
iPad
Cancel
Previous
Next
Pick appropriate device.
Status Bar Style
Default
Hide during application launch
App Icons
Source
AppIcon
Stanford CS193p
Fall 2013


Imaginarium
iPhone Retina (3.5-inch)
No Issues
Imaginarium
2 targets, iOS SDK 7.0
Imaginarium
Main.storyboard
ImageViewController.h
ImageViewController.m
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
ImaginariumTests
Frameworks
Products
Save As: Main_iPad
Tags:
Base.lproj
FAVORITES
CS193p
Documents
Applications
Downloads
Stanford
AppDelegate.h
AppDelegate.m
Base.lproj
en.lproj
Images.xcassets
ImageViewController.h
ImageViewController.m
Imaginarium-Info.plist
Imaginarium-Prefix.pch
main.m
ViewController.h
Main.storyboard
Group
Imaginarium
Targets
Imaginarium
ImaginariumTests
New Folder
Cancel
Create
Pick a good name.
Find your existing storyboard and put the new one in the same place.
Put in desired group.
App Icons
Source
AppIcon
Stanford CS193p
Fall 2013


Imaginarium
iPhone Retina (3.5-inch)
No Issues
Imaginarium
Imaginarium
2 targets, iOS SDK 7.0
Imaginarium
Main_iPad.storyboard
Main.storyboard
ImageViewController.h
ImageViewController.m
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
ImaginariumTests
Frameworks
Products
Here's your iPad storyboard.
Imaginarium
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Bundle Identifier
edu.stanford.cs193p.instructor.Imag
Version
1.0
Build
1.0
Team
None
Deployment Info
Deployment Target
7.0
Devices
Universal
iPhone
iPad
Main Interface
Main
Device Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style
Default
Hide during application launch
App Icons
Source
AppIcon
Stanford CS193p
Fall 2013


Imaginarium
iPhone Retina (3.5-inch)
No Issues
Imaginarium
Imaginarium
2 targets, iOS SDK 7.0
Imaginarium
Main_iPad.storyboard
Main.storyboard
ImageViewController.h
ImageViewController.m
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
ImaginariumTests
Frameworks
Products
Here's your iPad storyboard.
Imaginarium
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Bundle Identifier
edu.stanford.cs193p.instructor.Imag
Version
1.0
Build
1.0
Team
None
To set it as the storyboard to use on iPad, click on iPad here.
Deployment Info
Deployment Target
7.0
Devices
Universal
iPhone
iPad
Main Interface
Main
Device Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style
Hide during application launch
App Icons
Source
AppIcon
Stanford CS193p
Fall 2013

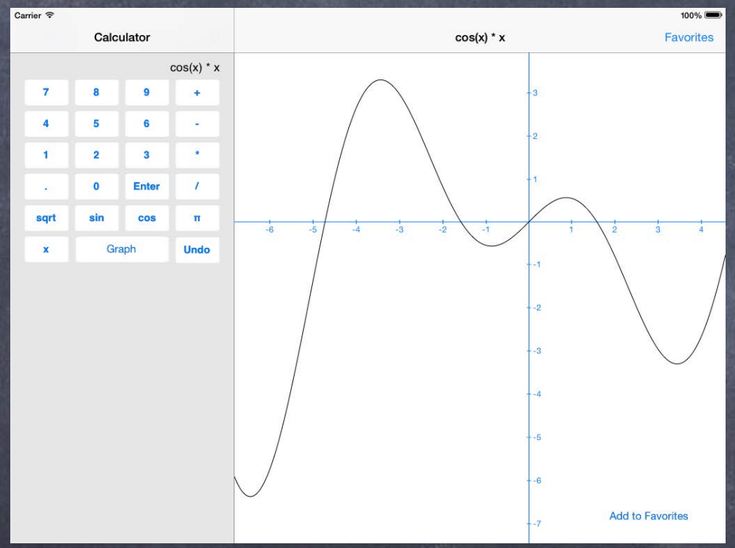

Imaginarium
iPhone Retina (3.5-inch)
No Issues
Imaginarium
Imaginarium
2 targets, iOS SDK 7.0
Imaginarium
Main_iPad.storyboard
Main.storyboard
ImageViewController.h
ImageViewController.m
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
ImaginariumTests
Frameworks
Products
Here's your iPad storyboard.
Imaginarium
General
Capabilities
Info
Build Settings
Build Phases
Build Rules
Bundle Identifier
edu.stanford.cs193p.instructor.Imag
Version
1.0
Build
1.0
Team
None
Deployment Info
Deployment Target
7.0
Devices
Universal
iPhone
iPad
Main Interface
Main_iPad
Device Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style
Hide during application launch
Then choose your newly-created iPad storyboard.
App Icons
Source
AppIcon
Stanford CS193p
Fall 2013

Universal Applications

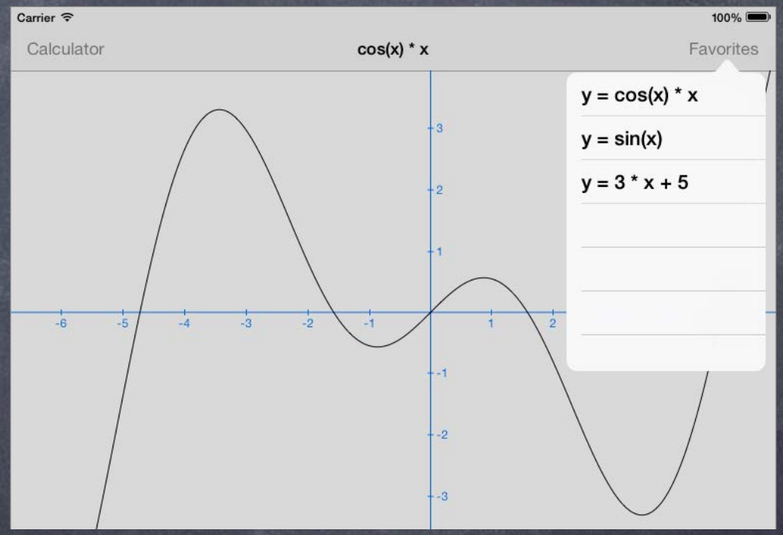
- iPad user-interface idioms

  The iPad has more screen real estate, so it can present MVCs in a couple of other ways.

Split View



Popover



Universal Applications

- How do I figure out "am I on an iPad?"

```
BOOL iPad = ([[UIDevice currentDevice] userInterfaceIdiom] == UIUserInterfaceIdiomPad);
```

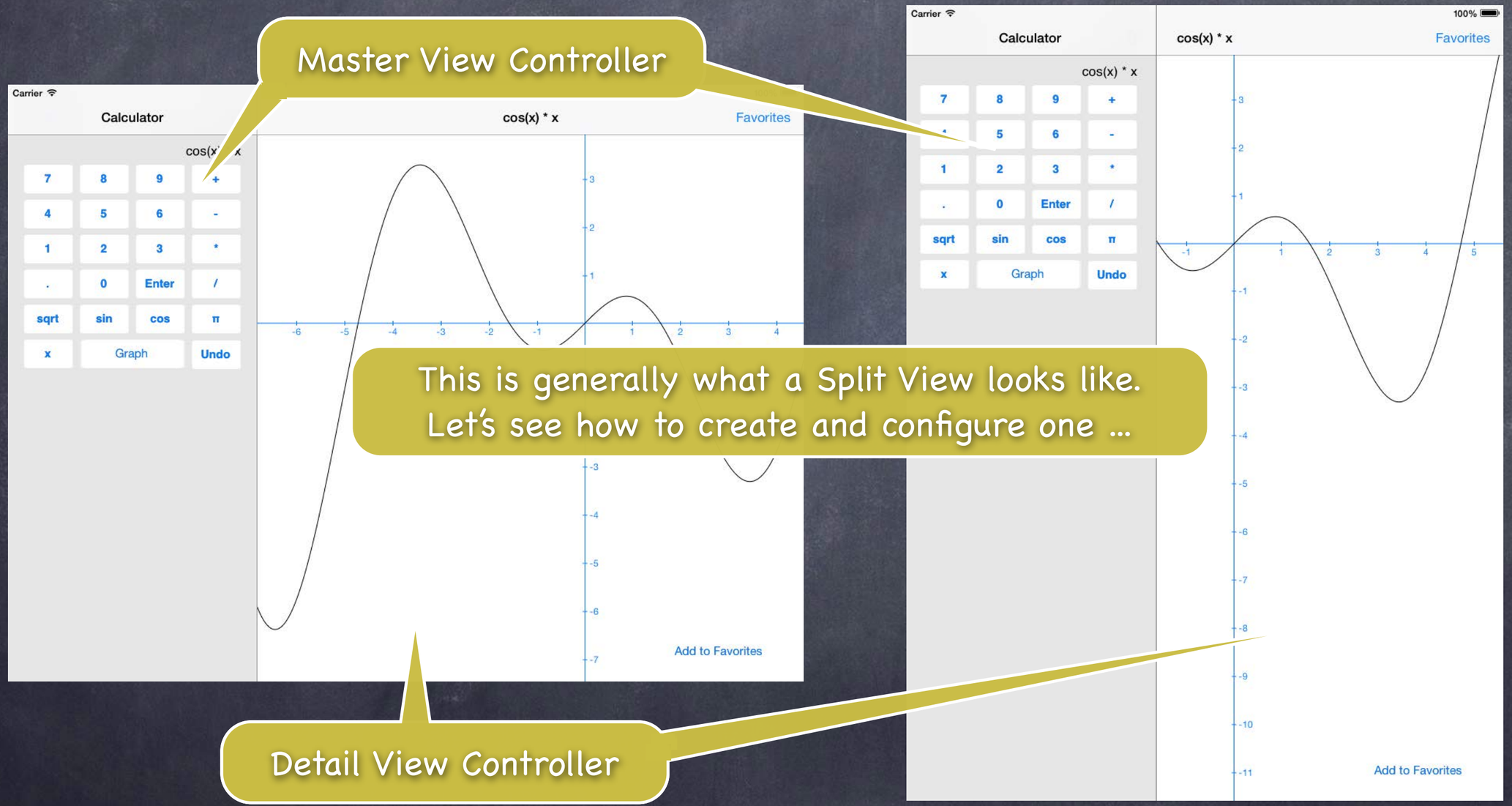
Use this sparingly!

Checking other things (like whether you are in a split view or popover) might be better design.

Or maybe check to see if your MVC or another MVC are "on screen" now
(because with more screen real estate, iPad can often have multiple MVCs showing at once).

Remember this? `if (self.view.window == nil) { /* I am not on screen right now */ }`

UISplitViewController



UISplitViewController

- Designed to be at the top-level of your storyboard
  - Don't try to put it inside a tab bar controller or navigation controller!
  - But you can put either of those inside either side of the split view.
- Simple to add to your storyboard
  - Just drag it out (and usually delete the "free" Master and Detail it gives you).
  - If you don't see a Split View in your Object Palette, then you're not editing an iPad storyboard.
  - Then ctrl-drag to each of the two sides (Master and Detail) of the split view.

UISplitViewController

- Accessing the Master and Detail MVCs from code

  All UIViewControllers know the UISplitViewController they are contained in (if in one):

  ```
  @property (strong) UISplitViewController *splitViewController;
  ```

  e.g. `if (self.splitViewController) { /* I am in a UISplitViewController */ }`

  The UISplitViewController has a property which is an array containing Master and Detail:

  ```
  @property (copy) NSArray *viewControllers; // index 0 is Master, 1 is Detail
  ```

  This property is not readonly, so you can change the Master & Detail of a Split View.
  The array is immutable though, so you must set both Master & Detail together at once.
  Usually you set this by ctrl-dragging in your storyboard though, not in code.

  e.g. A Master VC wants to get ahold of the Detail VC of the Split View both are in …

  ```
  UIViewController *detailVC = self.splitViewController.viewControllers[1];
  ```

  If the Master VC is not in a Split View, this would nicely return nil.

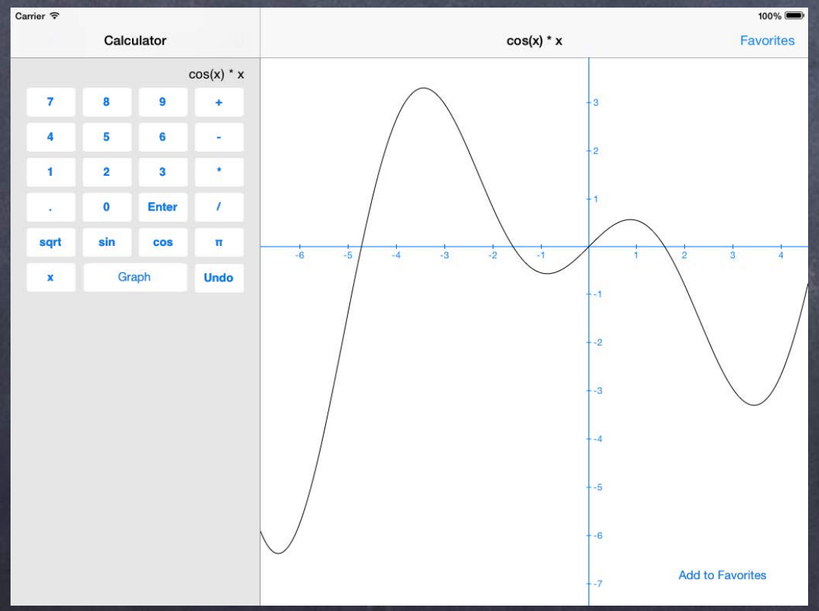
UISplitViewControllerDelegate

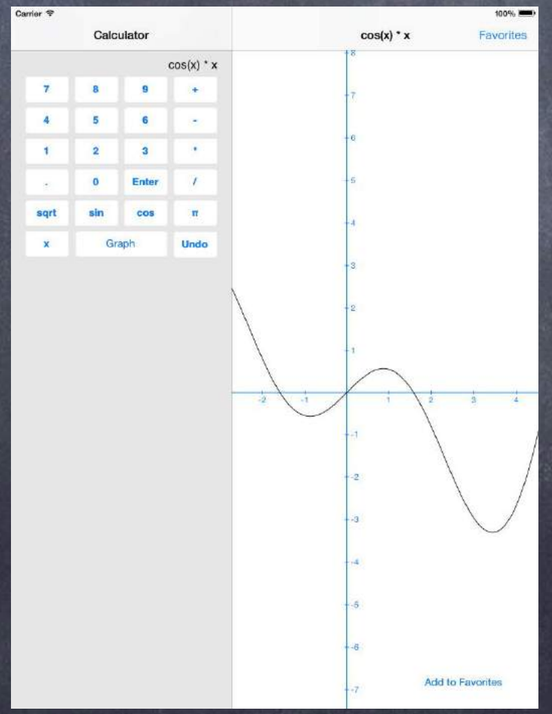
- UISplitViewController requires its delegate to be set
  - Or, at least, if you don't set it, then in portrait mode, the Master will be inaccessible.
  - `@property (assign) id <UISplitViewControllerDelegate> delegate;`
  - By the way, "assign" above is like "weak" except it doesn't set to nil when it leaves the heap!
  - Seems dangerous (and it can be), except that a Controller is almost always the delegate.
  - And a Controller is unlikely to leave the heap before elements of the View do.
- You must set this delegate very early!
  - Probably in awakeFromNib.
  - e.g., UISplitViewController starts sending its delegate methods way before viewDidLoad.
  - And then, unfortunately, when its delegate methods get sent to you, your outlets aren't set yet!
  - This can make being a UISplitViewController's delegate a real pain.
- What is the delegate's responsibility?
  - To control how the Master and Detail are presented when device rotation occurs …

UISplitViewControllerDelegate

- Never hide the left side (Master) behind a bar button

```
- (BOOL)splitViewController:(UISplitViewController *)sender
   shouldHideViewController:(UIViewController *)master
              inOrientation:(UIInterfaceOrientation)orientation
{
    return NO; // never hide it
}
```

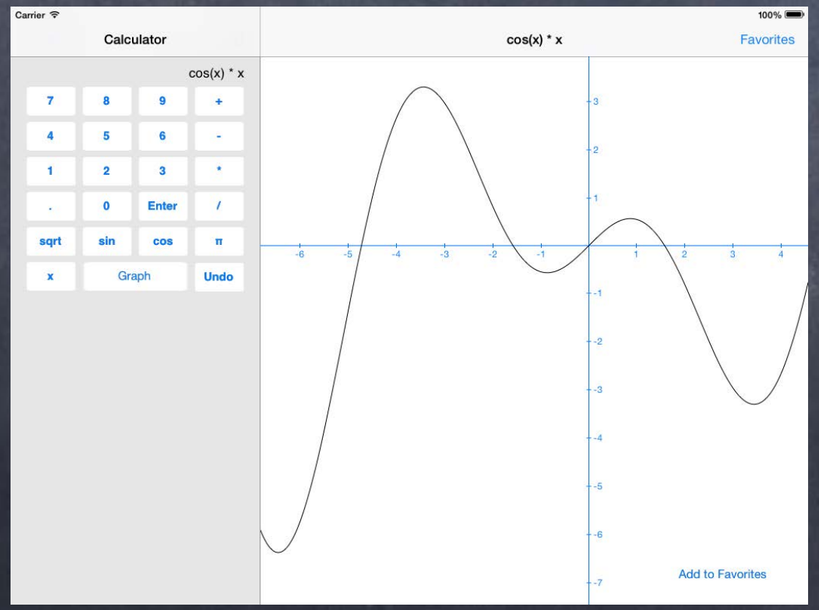


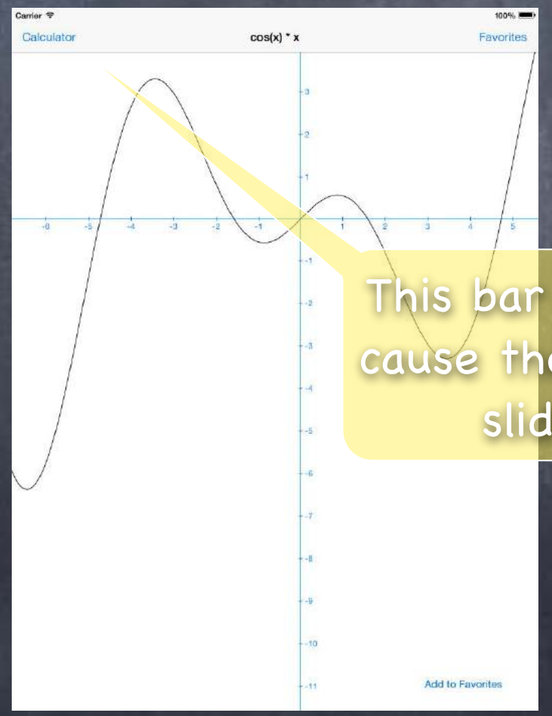


UISplitViewControllerDelegate

- Hide Master in portrait orientation only (the default)

```
- (BOOL)splitViewController:(UISplitViewController *)sender
   shouldHideViewController:(UIViewController *)master
              inOrientation:(UIInterfaceOrientation)orientation
{
    return UIInterfaceOrientationIsPortrait(orientation);
}
```



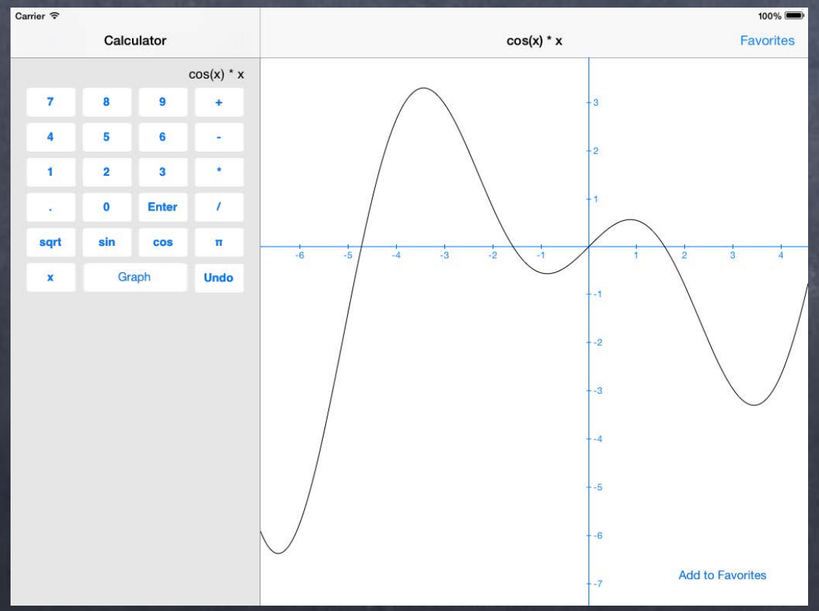


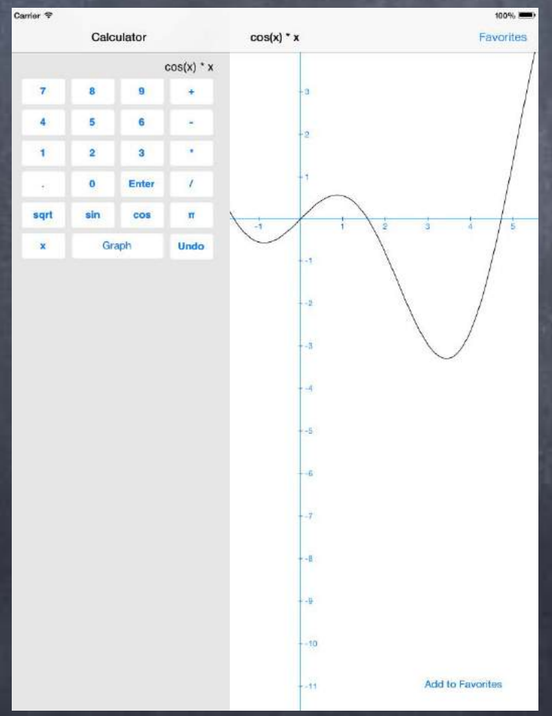
This bar button will cause the Master to slide out!

UISplitViewControllerDelegate

- Hide Master in portrait orientation only (the default)

```
- (BOOL)splitViewController:(UISplitViewController *)sender
   shouldHideViewController:(UIViewController *)master
              inOrientation:(UIInterfaceOrientation)orientation
{
    return UIInterfaceOrientationIsPortrait(orientation);
}
```

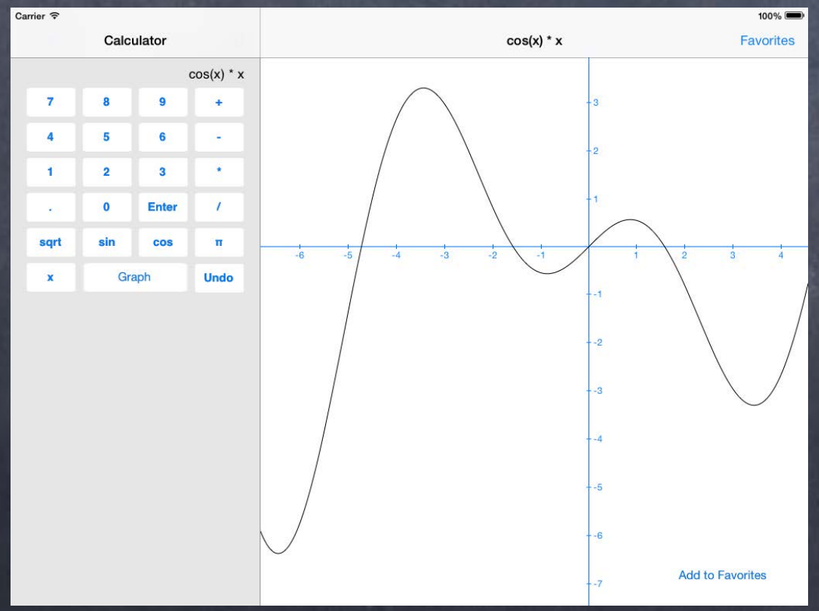


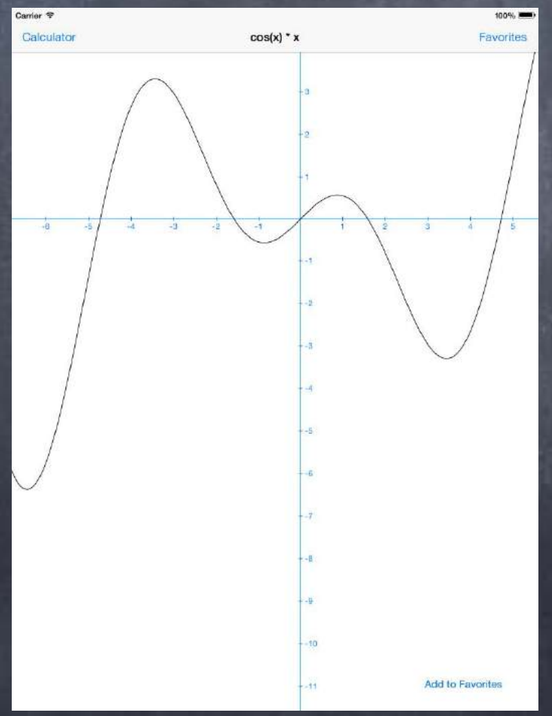


UISplitViewControllerDelegate

- Hide Master in portrait orientation only (the default)

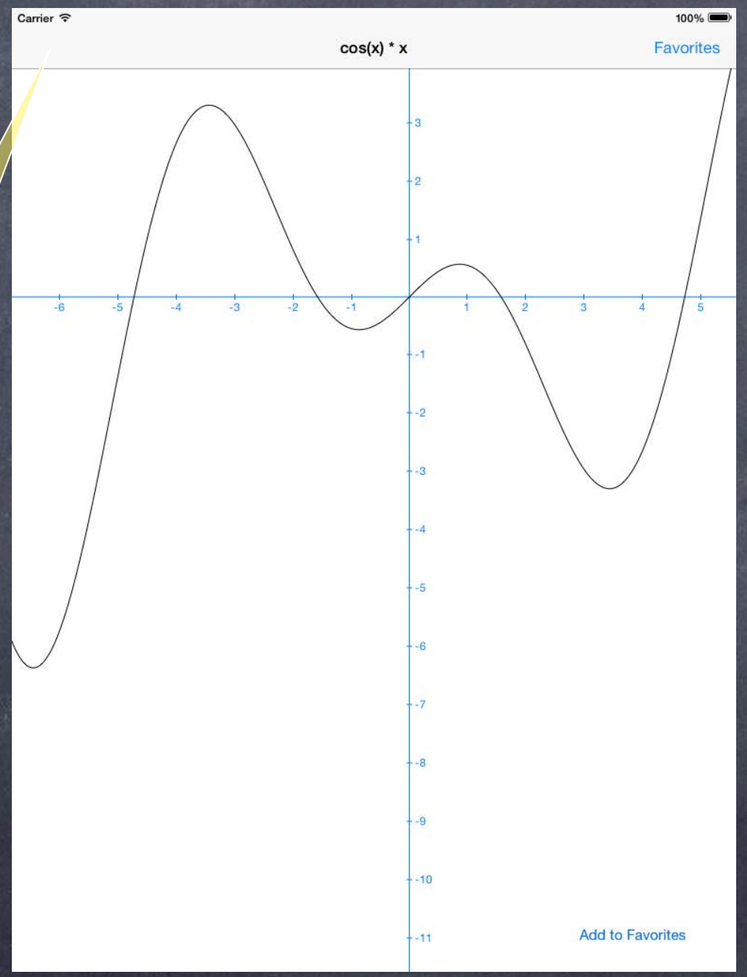
```
- (BOOL)splitViewController:(UISplitViewController *)sender
   shouldHideViewController:(UIViewController *)master
              inOrientation:(UIInterfaceOrientation)orientation
{
    return UIInterfaceOrientationIsPortrait(orientation);
}
```





UISplitViewControllerDelegate

- If you forget to set the `delegate`, you'll get this …



No button to slide the Master on screen!

UISplitViewControllerDelegate

- Split View helps you by providing that bar button

This gets called in your delegate when the master gets hidden …

```
- (void)splitViewController:(UISplitViewController *)sender
     willHideViewController:(UIViewController *)master
          withBarButtonItem:(UIBarButtonItem *)barButtonItem
       forPopoverController:(UIPopoverController *)popover
{
    barButtonItem.title = master.title;
    // this next line would only work in the Detail
    // and only if it was in a UINavigationController
    self.navigationItem.leftBarButton = barButtonItem;
}
```

See? You are being provided the bar button item. You just need to put it on screen somewhere.

UISplitViewControllerDelegate

- When it's time for the bar button to go away …

  This gets called in your delegate when the master reappears …

```
- (void)splitViewController:(UISplitViewController *)sender
     willShowViewController:(UIViewController *)master
  invalidatingBarButtonItem:(UIBarButtonItem *)barButtonItem
{
    // this next line would only work in the Detail
    // and only if it was in a UINavigationController
    self.navigationItem.leftBarButton = nil;
}
```

UISplitViewController

- Updating the Detail when the Master is touched?

  There are 2 choices for how to do this: Target/Action or Replace Segue

- Target/Action

  Example (code in the Master view controller) …

```
- (IBAction)doit
{
    id detailViewController = self.splitViewController.viewControllers[1];
    [detailViewController setSomeProperty:…]; // might want some Introspection first
}
```

- Replace Segue (entirely replaces the Detail view controller)

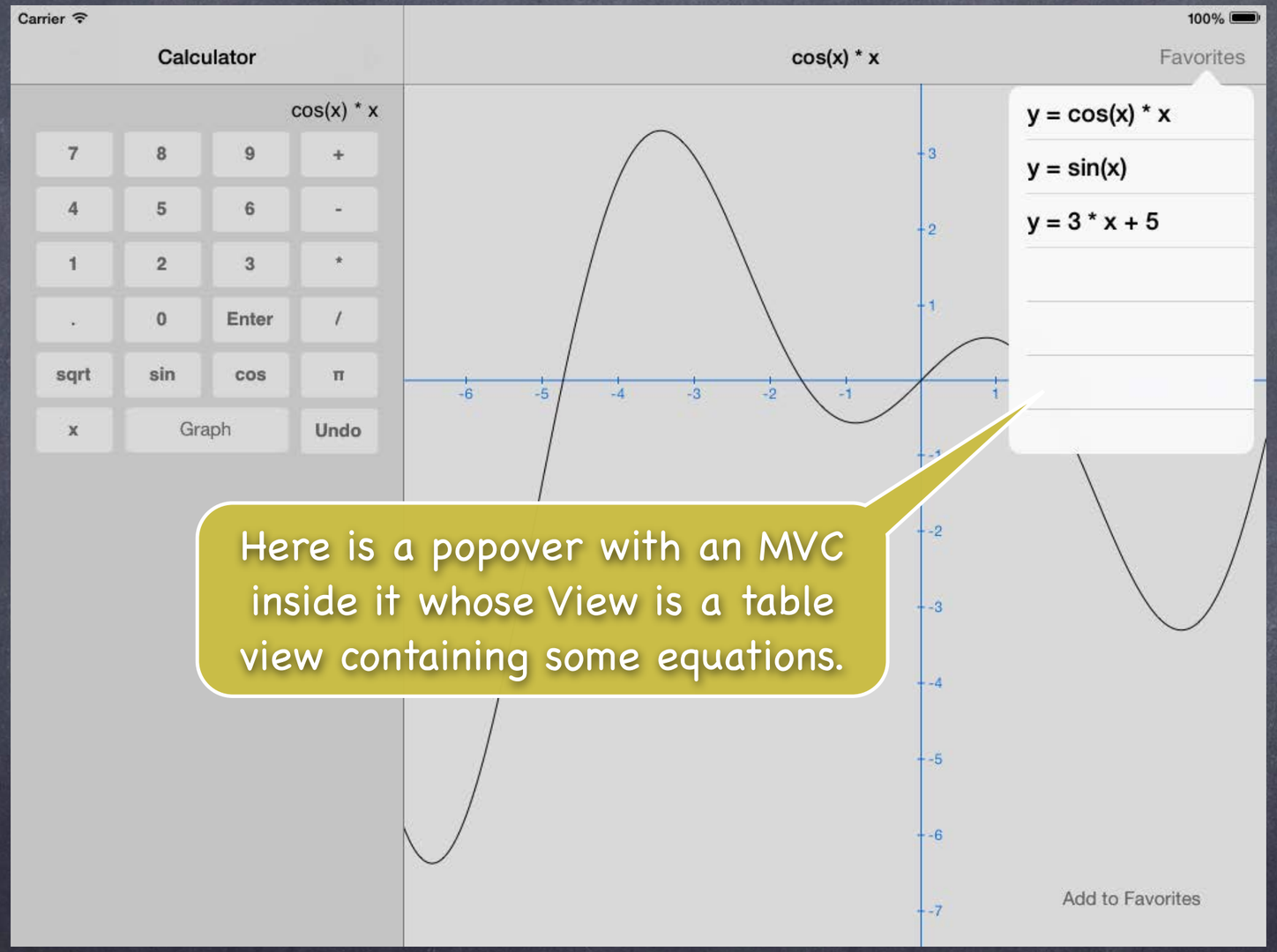
  Remember, segues always instantiate a view controller (split view stops pointing to old one).

  Can Replace either side, but much more common to replace the right side (since it's the "detail").

  Be careful! You might lose the `UIBarButtonItem` used for revealing the hidden Master!

  (you'd need to be sure to put it back into the newly instantiated view controller)

Popovers


Carrier
100%
Calculator
cos(x) * x
Favorites
cos(x) * x
7 8 9 +
4 5 6 -
1 2 3 *
. 0 Enter /
sqrt sin cos π
x Graph Undo
y = cos(x) * x
y = sin(x)
y = 3 * x + 5
Add to Favorites
Here is a popover with an MVC inside it whose View is a table view containing some equations.

Popovers

- `UIPopoverController` is not, itself, a `UIViewController`
  - Instead it has a `@property` that holds the `UIViewController` that is inside it …
  - `@property (nonatomic, strong) UIViewController *contentViewController;`
  - This is usually wired up in a storyboard …

Popovers

- Creating a Popover Segue in your Storyboard

  Just drag from the UI element you want to cause the popover to the scene you want to pop up.

  In your prepareForSegue:sender:, the argument will be isKindOf:UIStoryboardPopoverSegue.
  And UIStoryboardPopoverSegue has a @property you can use to get the UIPopoverController:

```
- (UIPopoverController *)popoverController;
```

  Example:

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([segue isKindOfClass:[UIStoryboardPopoverSegue class]]) {
        UIPopoverController *popoverController =
            ((UIStoryboardPopoverSegue *)segue).popoverController;
        . . .
    }
}
```

Popover

- You can also present a popover from code

  Popover has a little arrow that points to what (rectangle or button) brought it up.

  You can specify which directions it is valid to point (and thus where the popover will pop up).

```
UIPopoverController *popover =
    [[UIPopoverController alloc] initWithContentViewController:myPoppedUpVC];
[popover presentPopoverFromRect:(CGRect)aRect // little arrow points to aRect in view's coords
                         inView:(UIView *)view
       permittedArrowDirections:(UIPopoverArrowDirection)direction
                       animated:(BOOL)flag];
```

  … or (points to a bar button item) …

```
[popover presentPopoverFromBarButtonItem:(UIBarButtonItem *)barButtonItem
                permittedArrowDirections:(UIPopoverArrowDirection)direction
                                animated:(BOOL)flag;
```

  * the "casts" on the arguments above are only for educational purposes, they are not required

Popover

- But don't forget to keep a `strong` pointer to the popover controller!

  Example: a target/action method attached to a `UIBarButtonItem` that presents a popover …

```
- (IBAction)presentPopover:(UIBarButtonItem *)item
{
    UIPopoverController *pop = [[UIPopoverController alloc] initWithViewController:vc];
    [pop presentPopoverFromBarButtonItem:item …];
}
```

  The above is bad because there is no strong pointer kept to the `UIPopoverController`!

Popover

- But don't forget to keep a `strong` pointer to the popover controller!

  Example: a target/action method attached to a `UIBarButtonItem` that presents a popover …

```
- (IBAction)presentPopover:(UIBarButtonItem *)item
{
    if (!self.popover) {
        self.popover = [[UIPopoverController alloc] initWithViewController:vc];
        [self.popover presentPopoverFromBarButtonItem:item …];
    }
}
// then set self.popover to nil when the popover is dismissed at a later time
```

  Speaking of which … how do we dismiss a popover (or find out that the user has dismissed it)?

Popover

- The user dismisses a popover by touching outside of it

  Unless the user touches in one of the views in this array property in UIPopoverController:

  ```
  @property (copy) NSArray *passthroughViews;
  ```

- Dismissing a popover from code

  UIPopoverController method:

  ```
  - (void)dismissPopoverAnimated:(BOOL)animated;
  ```

- Finding out that the user dismissed the popover

  UIPopoverController has a delegate too and it will be sent this message:

  ```
  - (void)popoverControllerDidDismissPopover:(UIPopoverController *)sender;
  ```

  This is only sent if the user dismisses the popover.

Demo

- Shutterbug
  - `UITableView`
  - Flickr
  - Universal Application
  - `UISplitViewController`
  - `UIRefreshControl`
  - GCD
  - No Popover, sorry, but you will not be asked to do that in your homework assignment.

Coming Up

- Homework
  - Due next Wednesday.
- Friday
  - Stanford Only Review Section
- Next Week
  - Core Data (Object-Oriented Database)
  - Maybe some Multitasking API

Stanford CS193p

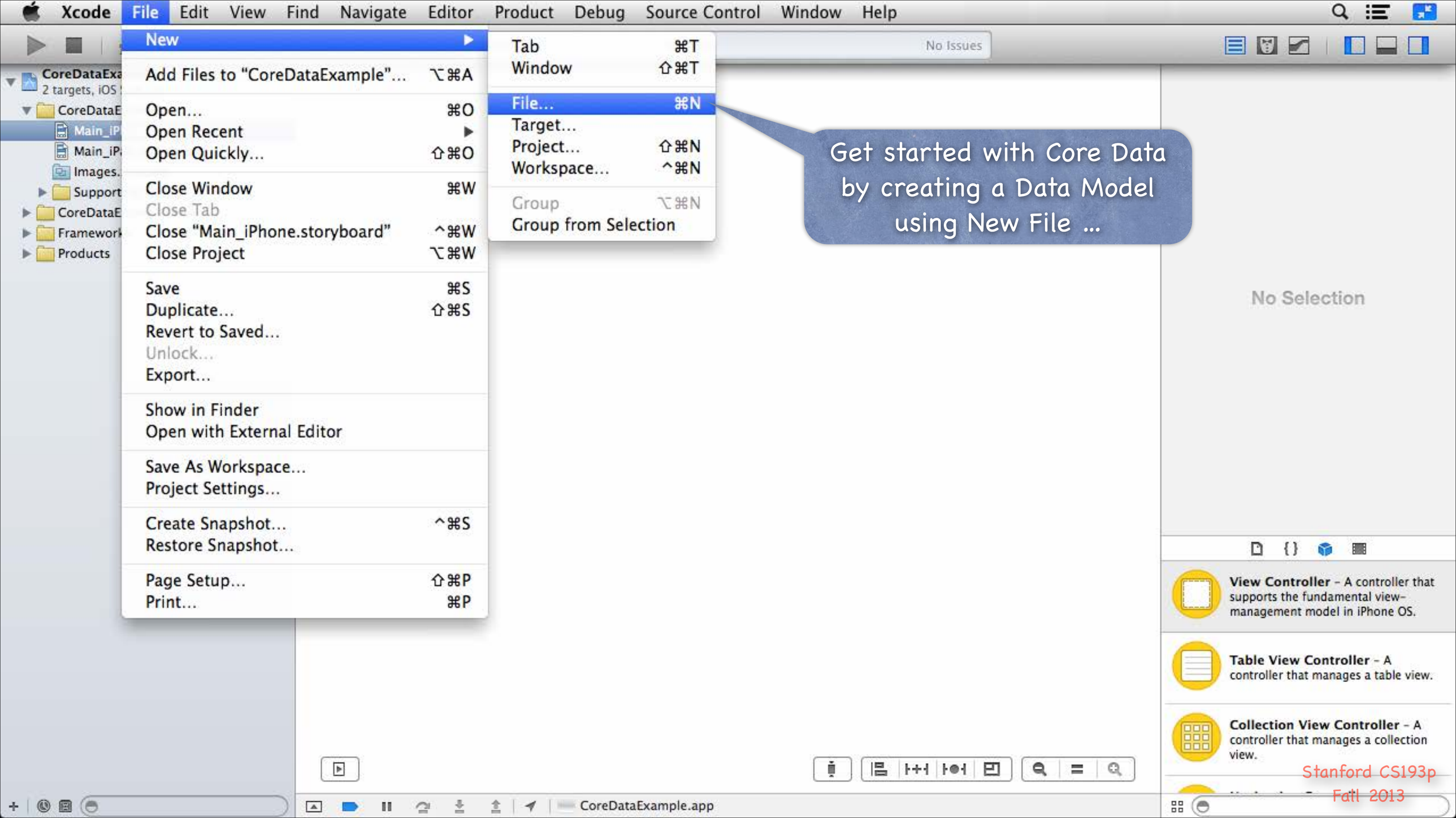
Developing Applications for iOS
Fall 2013-14

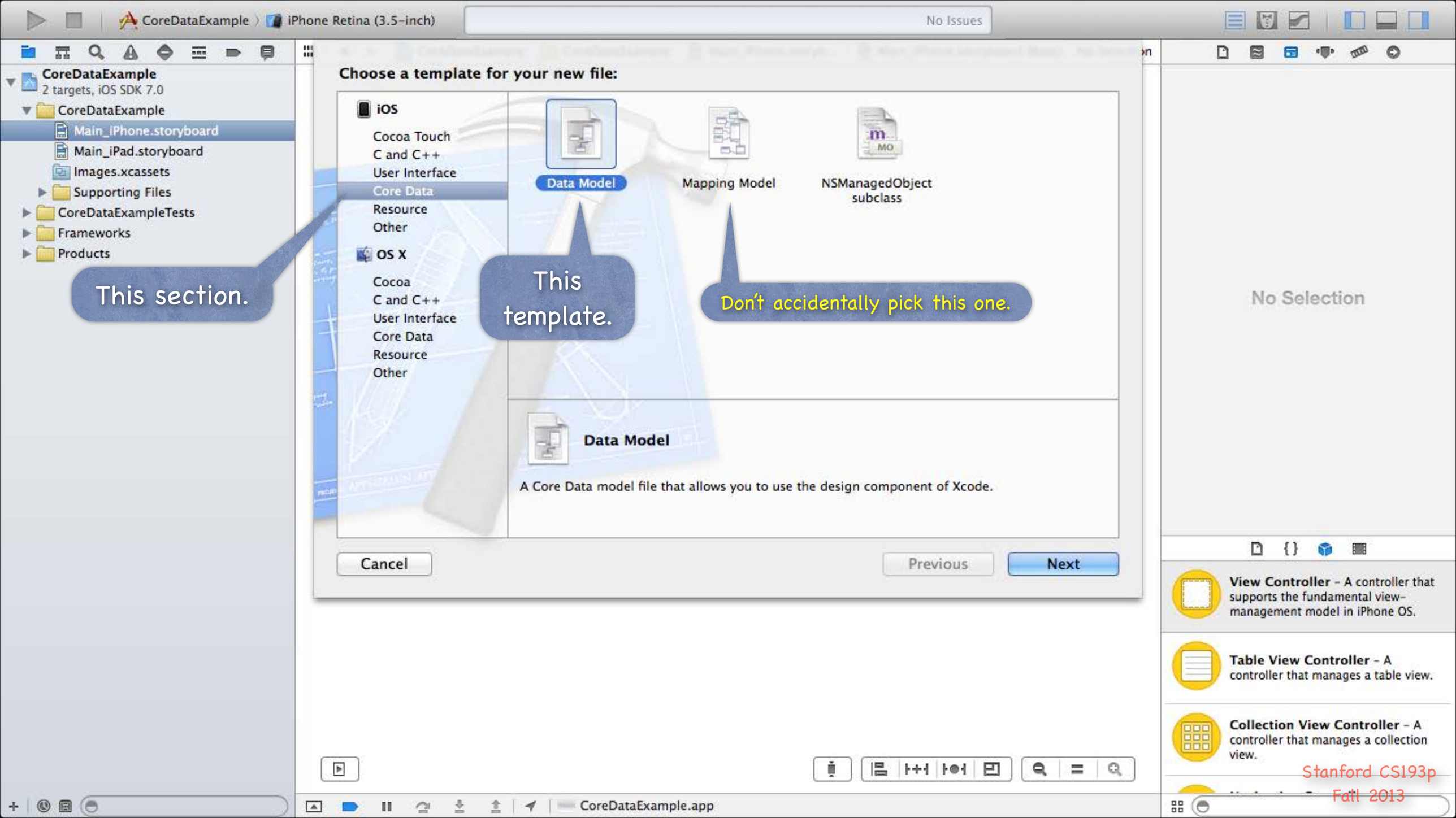
Agenda

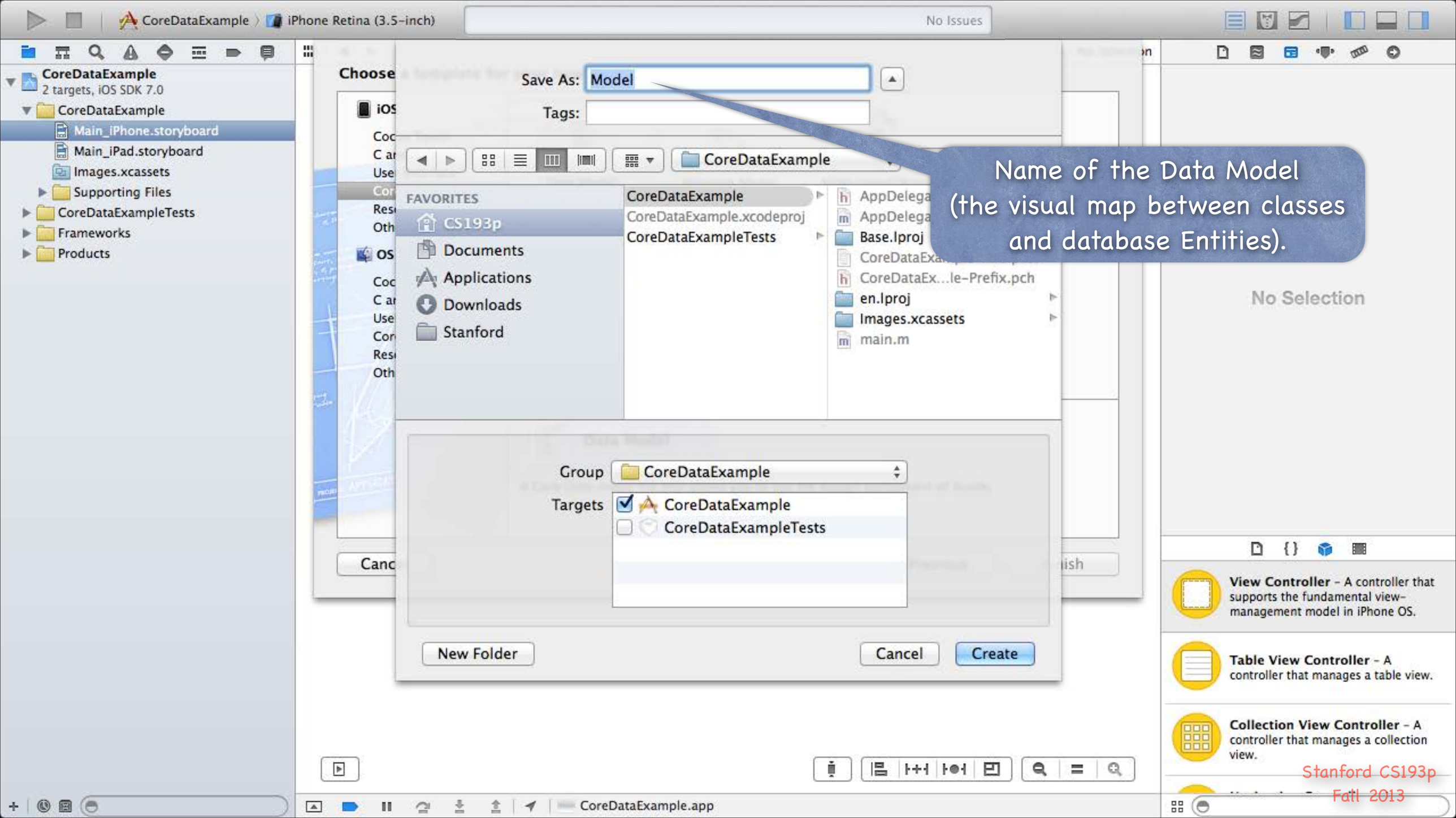
- Core Data
  - Storing your Model permanently in an object-oriented database.
- Homework
  - Assignment 5 due Wednesday.
  - Final homework (Assignment 6) will be assigned Wednesday, due the next Wednesday.
- Wednesday
  - Final Project Requirements
  - Core Data and `UITableView`
  - Core Data Demo
- Next Week
  - Multitasking
  - Advanced Segueing
  - Map Kit?

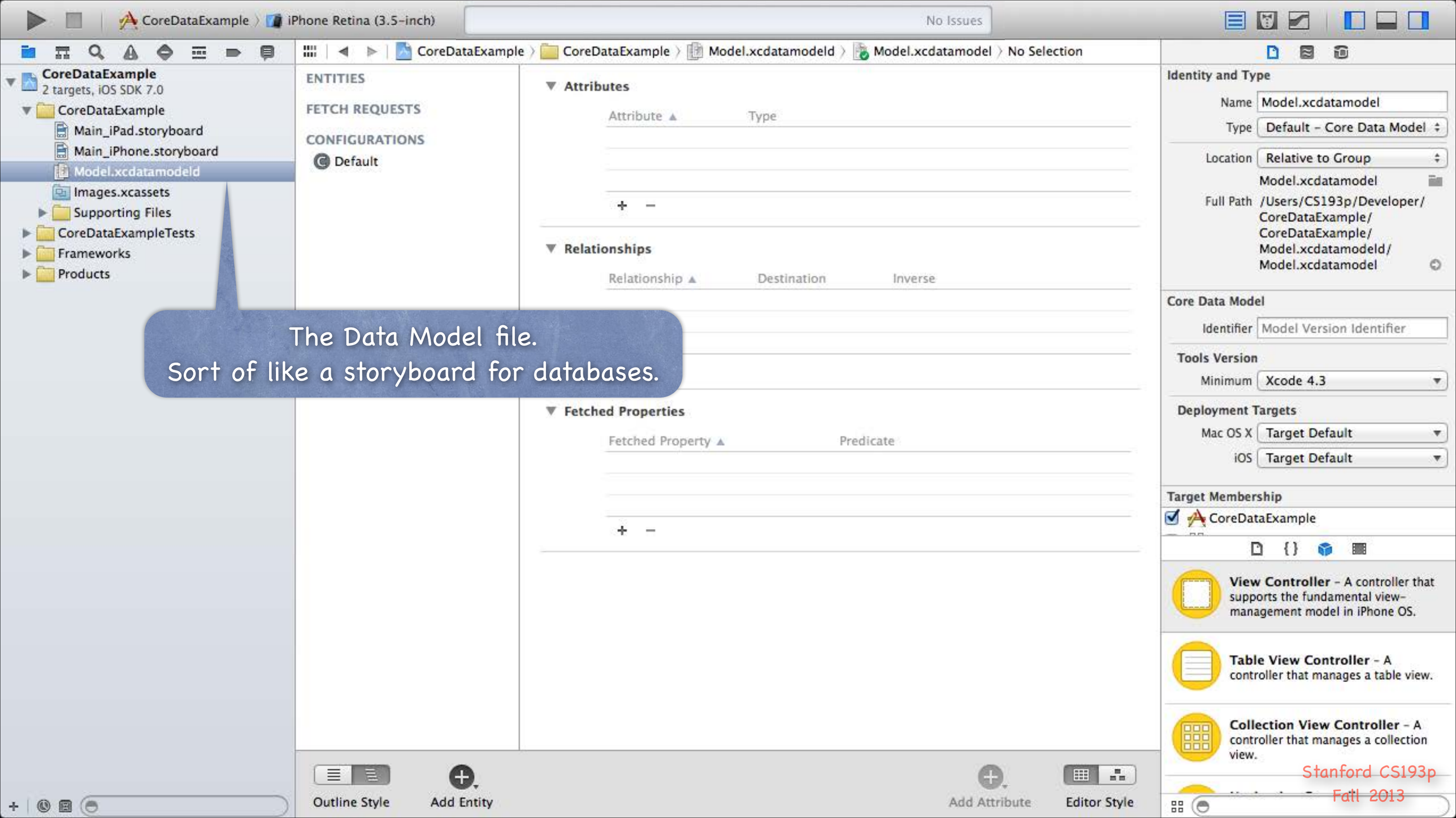
Core Data

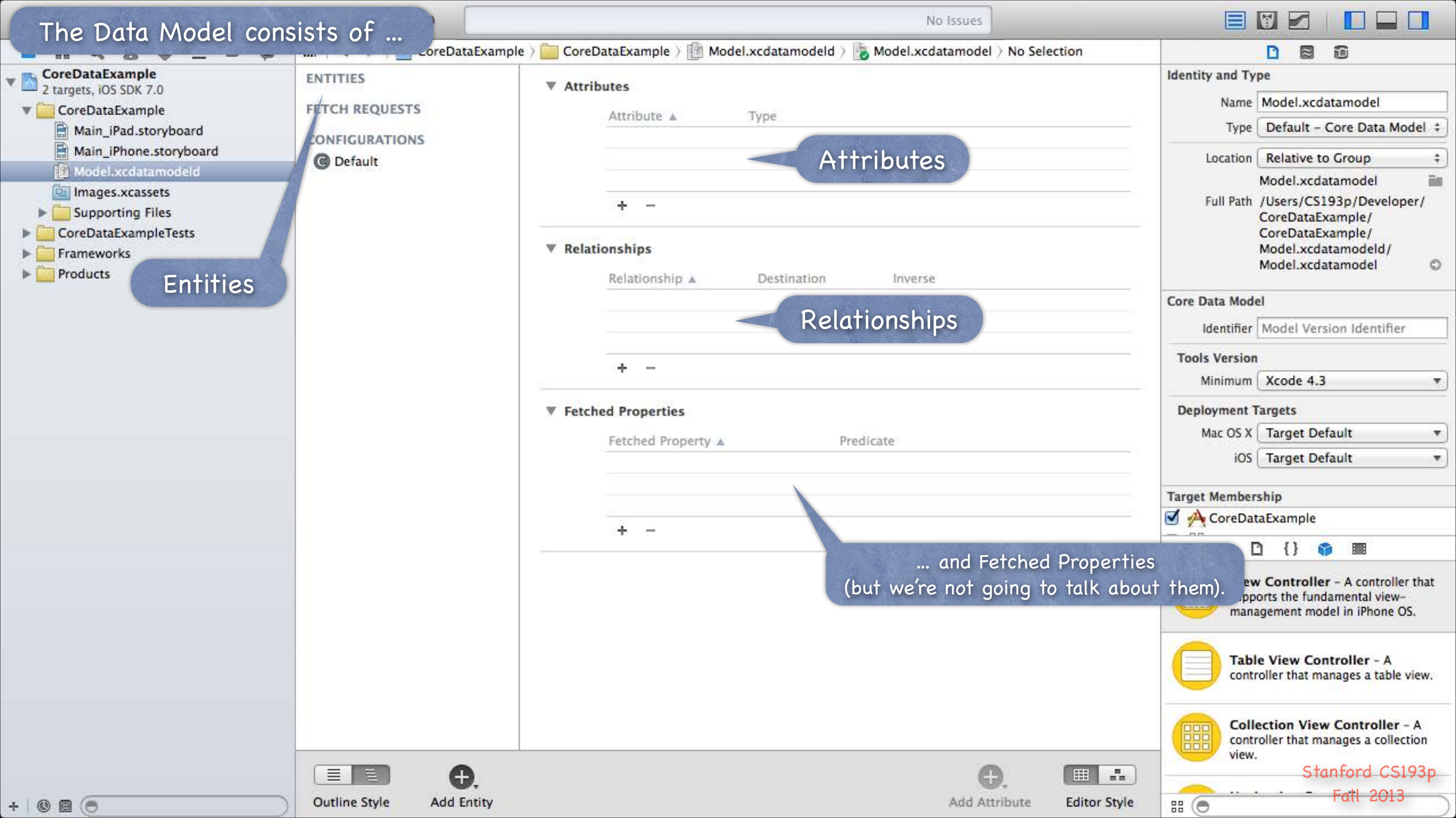
- Database
  - Sometimes you need to store large amounts of data or query it in a sophisticated manner.
  - But we still want it to be object-oriented objects!
- Enter Core Data
  - Object-oriented database.
  - Very, very powerful framework in iOS (we will only be covering the absolute basics).
- It's a way of creating an object graph backed by a database
  - Usually backed by SQL (but also can do XML or just in memory).
- How does it work?
  - Create a visual mapping (using Xcode tool) between database and objects.
  - Create and query for objects using object-oriented API.
  - Access the "columns in the database table" using `@propertys` on those objects.
  - Let's get started by creating that visual map ...

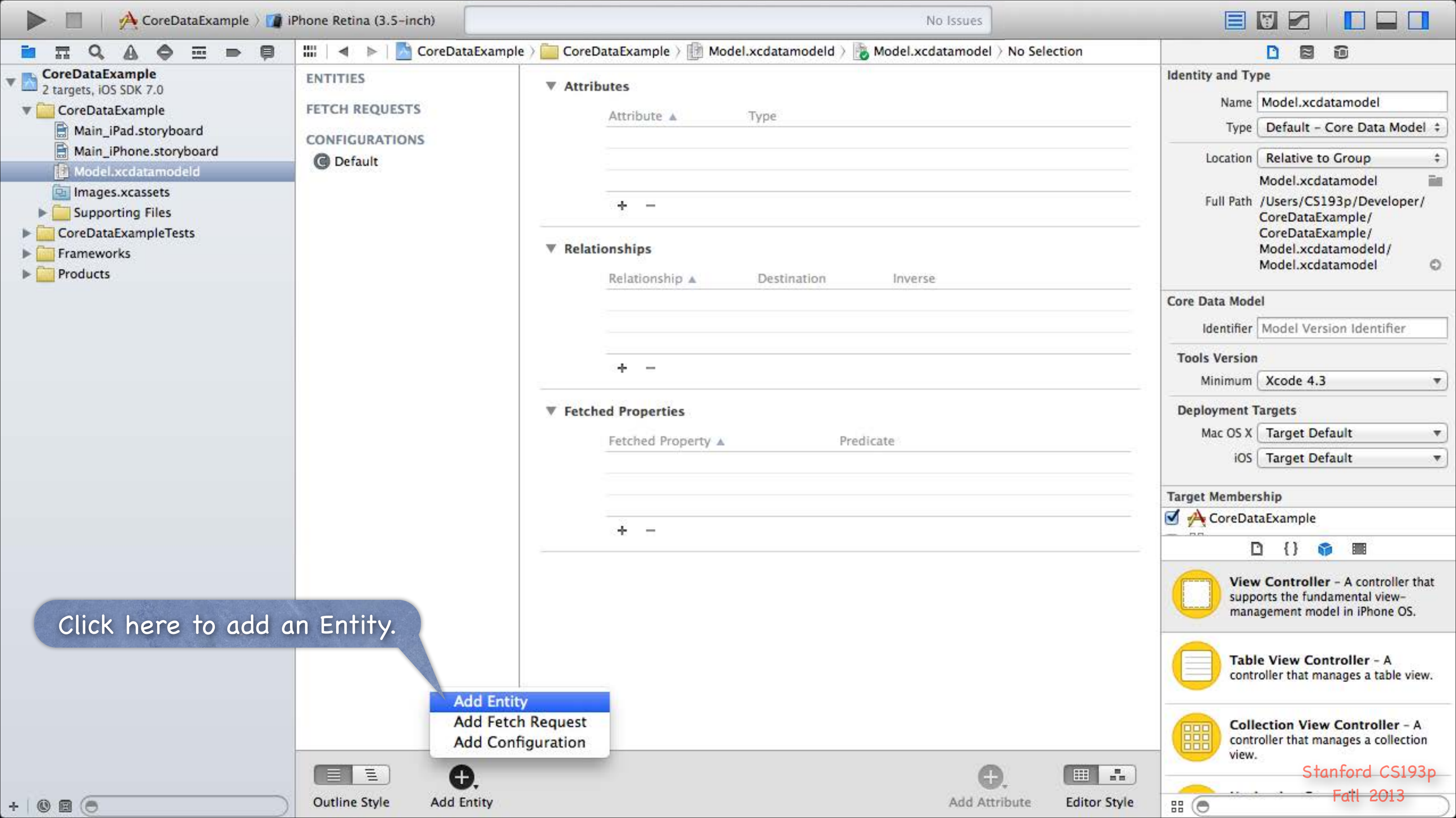

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
New
Add Files to "CoreDataExample"... ⌥⌘A
Open... ⌘O
Open Recent
Open Quickly... ⇧⌘O
Close Window ⌘W
Close Tab
Close "Main_iPhone.storyboard" ^⌘W
Close Project ⌥⌘W
Save ⌘S
Duplicate... ⇧⌘S
Revert to Saved...
Unlock...
Export...
Show in Finder
Open with External Editor
Save As Workspace...
Project Settings...
Create Snapshot... ^⌘S
Restore Snapshot...
Page Setup... ⇧⌘P
Print... ⌘P
Tab ⌘T
Window ⇧⌘T
File... ⌘N
Target...
Project... ⇧⌘N
Workspace... ^⌘N
Group ⌥⌘N
Group from Selection
Get started with Core Data by creating a Data Model using New File ...
No Selection
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
CoreDataExample.app
Stanford CS193p
Fall 2013

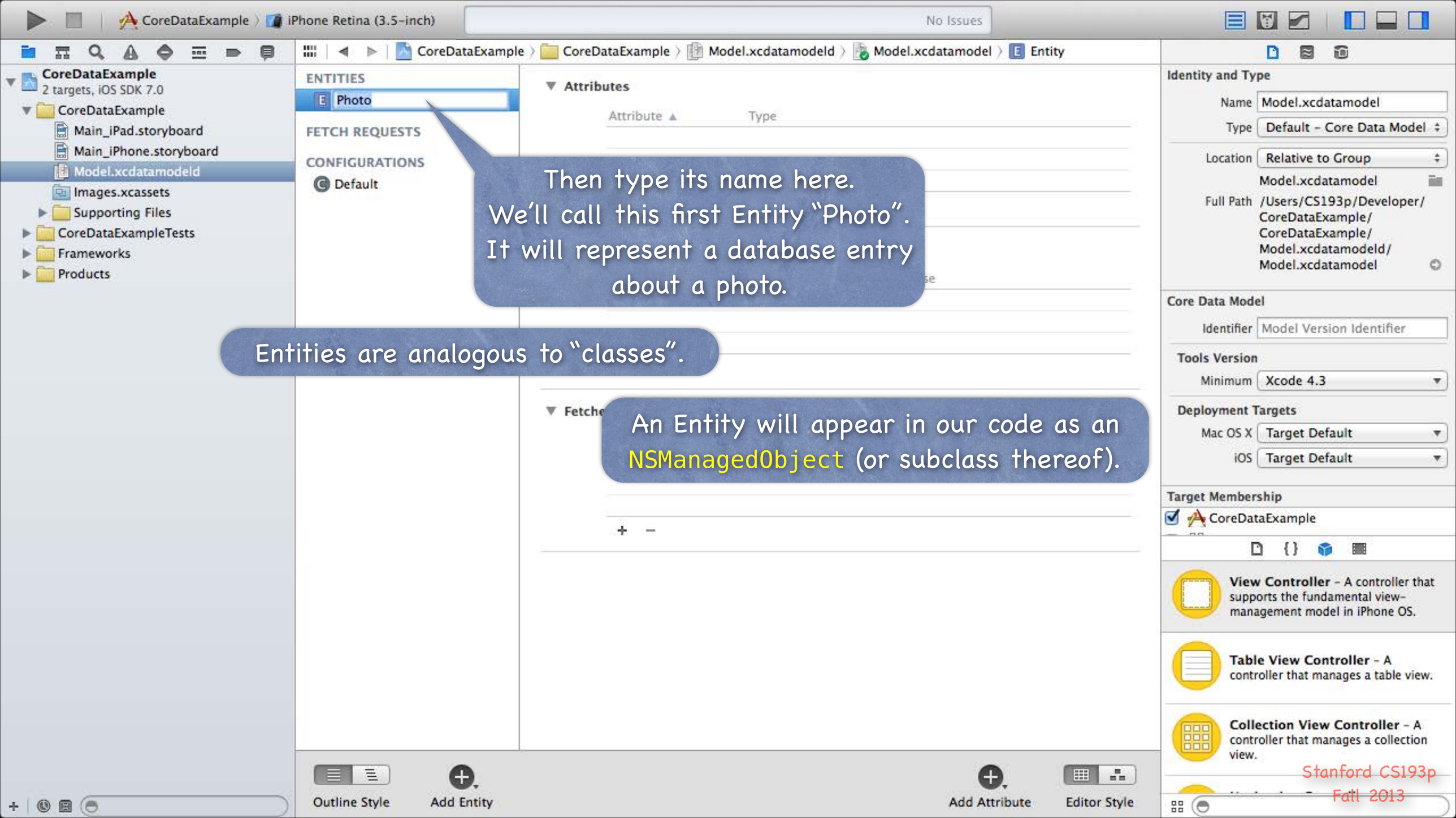

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPhone.storyboard
Main_iPad.storyboard
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
This section.
Choose a template for your new file:
iOS
Cocoa Touch
C and C++
User Interface
Core Data
Resource
Other
OS X
Cocoa
C and C++
User Interface
Core Data
Resource
Other
Data Model
Mapping Model
NSManagedObject subclass
This template.
Don't accidentally pick this one.
Data Model
A Core Data model file that allows you to use the design component of Xcode.
Cancel
Previous
Next
No Selection
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
CoreDataExample.app
Stanford CS193p
Fall 2013

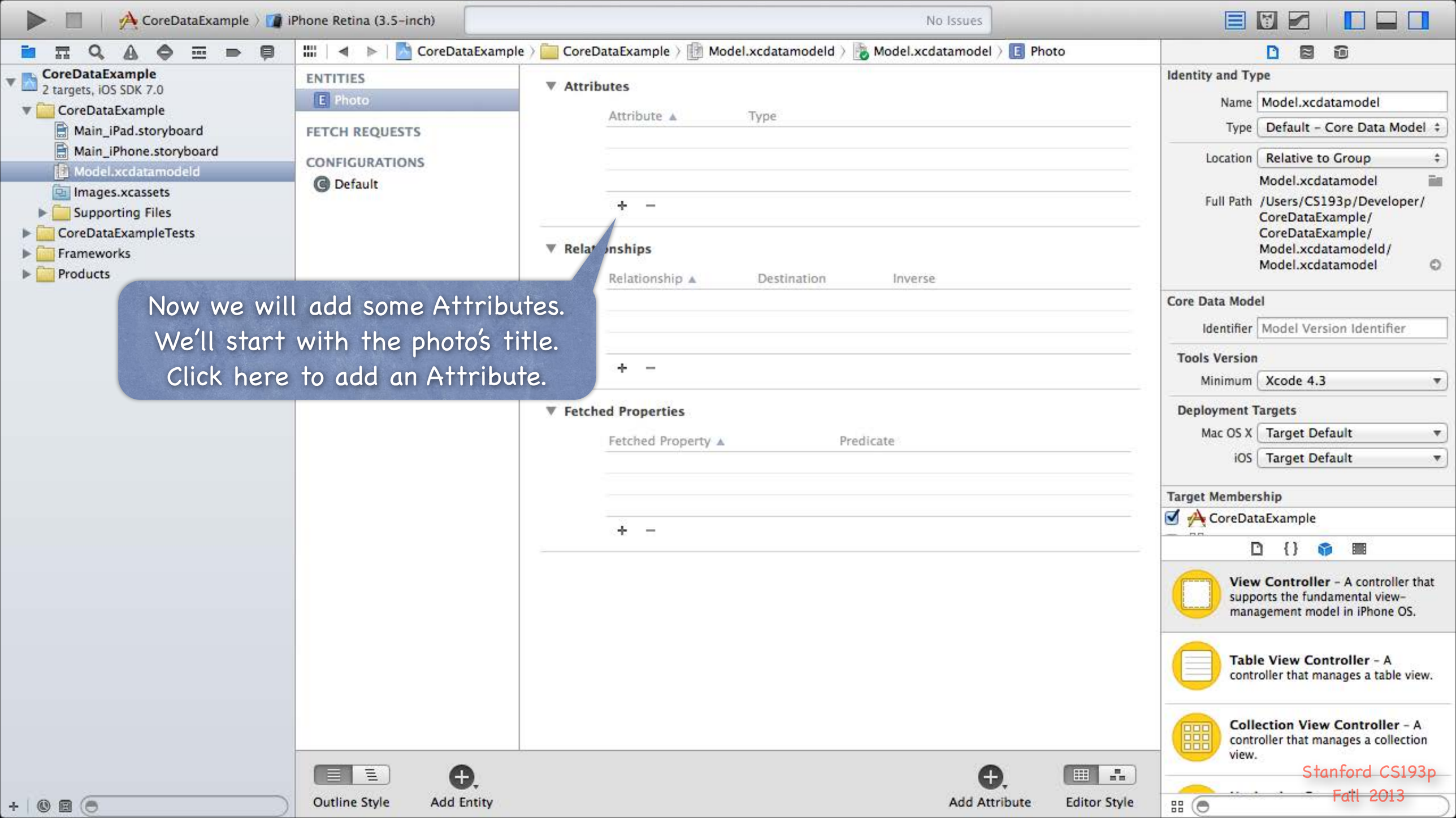

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPhone.storyboard
Main_iPad.storyboard
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
Save As: Model
Tags:
CoreDataExample
FAVORITES
CS193p
Documents
Applications
Downloads
Stanford
CoreDataExample
CoreDataExample.xcodeproj
CoreDataExampleTests
AppDelega
AppDelega
Base.lproj
CoreDataEx...le-Prefix.pch
en.lproj
Images.xcassets
main.m
Name of the Data Model
(the visual map between classes
and database Entities).
Group
CoreDataExample
Targets
CoreDataExample
CoreDataExampleTests
New Folder
Cancel
Create
No Selection
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
CoreDataExample.app
Stanford CS193p
Fall 2013

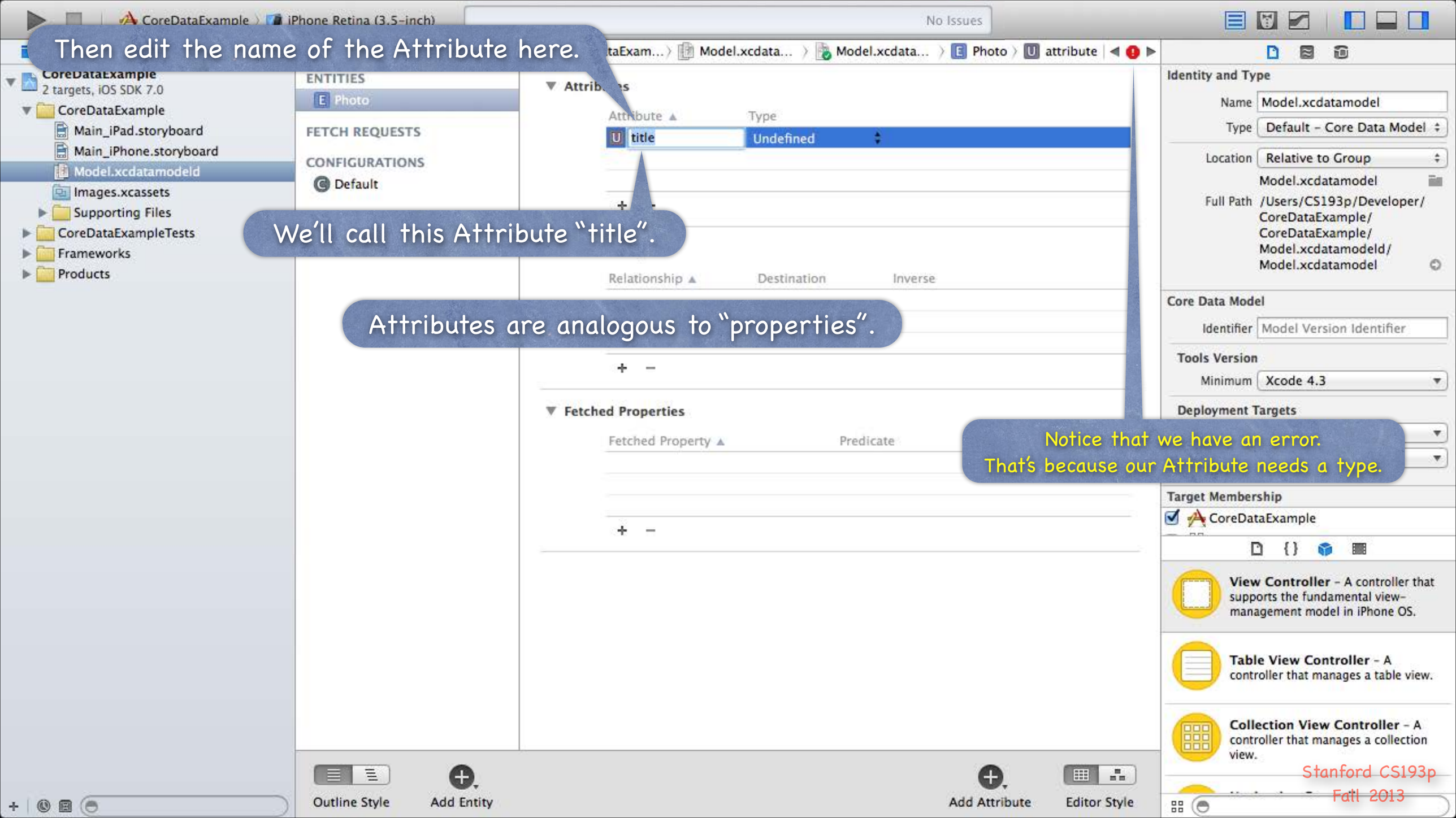

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
No Selection
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
FETCH REQUESTS
CONFIGURATIONS
Default
Attributes
Attribute
Type
Relationships
Relationship
Destination
Inverse
Fetched Properties
Fetched Property
Predicate
The Data Model file.
Sort of like a storyboard for databases.
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/
CoreDataExample/
CoreDataExample/
Model.xcdatamodeld/
Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Mac OS X
Target Default
iOS
Target Default
Target Membership
CoreDataExample
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

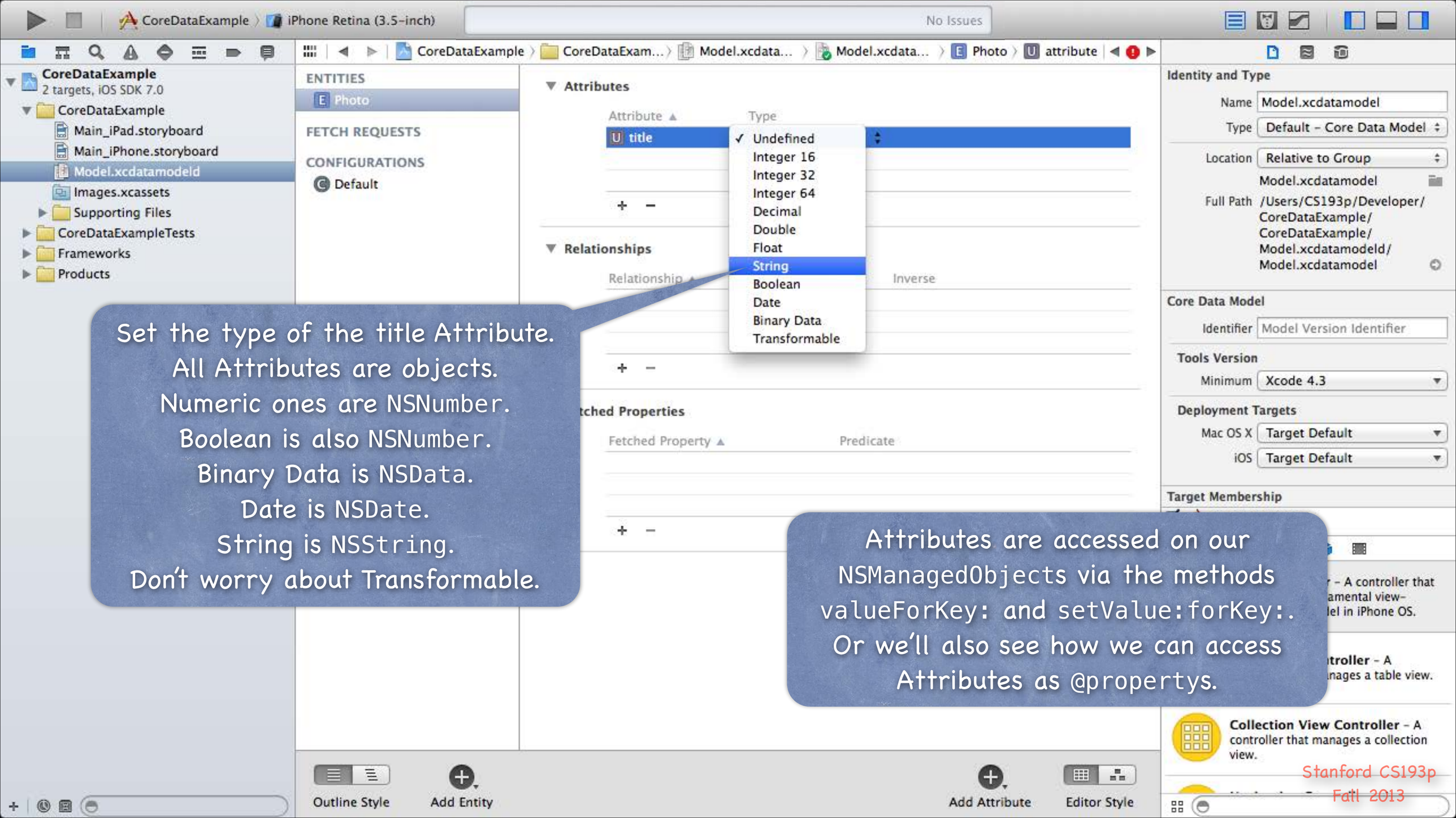

The Data Model consists of …
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
Entities
CoreDataExample › CoreDataExample › Model.xcdatamodeld › Model.xcdatamodel › No Selection
ENTITIES
FETCH REQUESTS
CONFIGURATIONS
Default
Attributes
Attribute
Type
Attributes
Relationships
Relationship
Destination
Inverse
Relationships
Fetched Properties
Fetched Property
Predicate
… and Fetched Properties
(but we're not going to talk about them).
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/CoreDataExample/CoreDataExample/Model.xcdatamodeld/Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Mac OS X
Target Default
iOS
Target Default
Target Membership
CoreDataExample
ew Controller - A controller that
pports the fundamental view-
management model in iPhone OS.
Table View Controller - A
controller that manages a table view.
Collection View Controller - A
controller that manages a collection
view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
No Selection
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
FETCH REQUESTS
CONFIGURATIONS
Default
Attributes
Attribute
Type
Relationships
Relationship
Destination
Inverse
Fetched Properties
Fetched Property
Predicate
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/CoreDataExample/CoreDataExample/Model.xcdatamodeld/Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Mac OS X
Target Default
iOS
Target Default
Target Membership
CoreDataExample
Click here to add an Entity.
Add Entity
Add Fetch Request
Add Configuration
Outline Style
Add Entity
Add Attribute
Editor Style
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
CoreDataExample › CoreDataExample › Model.xcdatamodeld › Model.xcdatamodel › Entity
ENTITIES
Photo
FETCH REQUESTS
CONFIGURATIONS
Default
Attributes
Attribute
Type
Then type its name here.
We'll call this first Entity "Photo".
It will represent a database entry
about a photo.
Entities are analogous to "classes".
An Entity will appear in our code as an
NSManagedObject (or subclass thereof).
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/
CoreDataExample/
CoreDataExample/
Model.xcdatamodeld/
Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Mac OS X
Target Default
iOS
Target Default
Target Membership
CoreDataExample
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

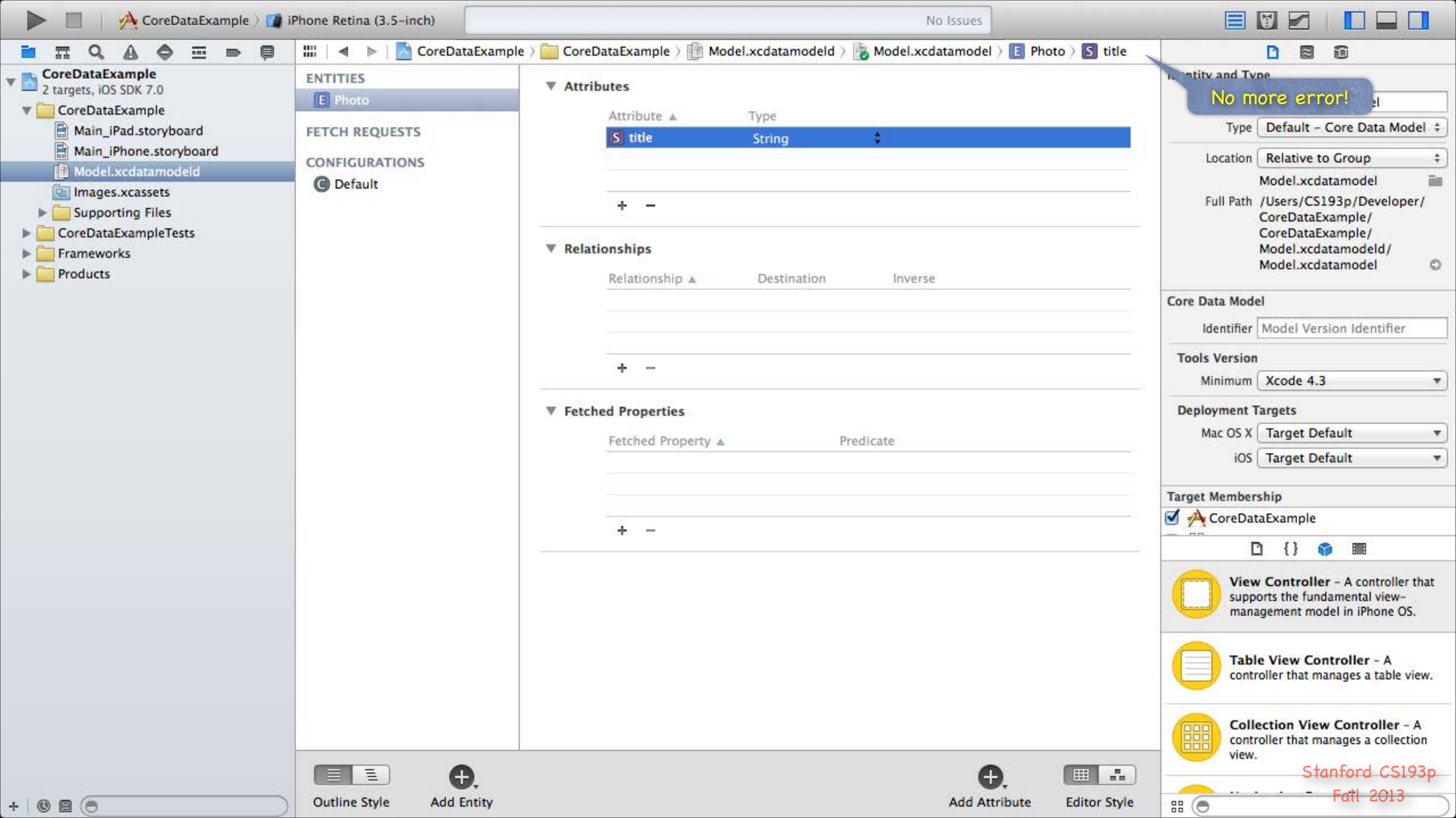

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
FETCH REQUESTS
CONFIGURATIONS
Default
Attributes
Attribute
Type
Relationships
Relationship
Destination
Inverse
Fetched Properties
Fetched Property
Predicate
Now we will add some Attributes.
We'll start with the photo's title.
Click here to add an Attribute.
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/
CoreDataExample/
CoreDataExample/
Model.xcdatamodeld/
Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Mac OS X
Target Default
iOS
Target Default
Target Membership
CoreDataExample
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

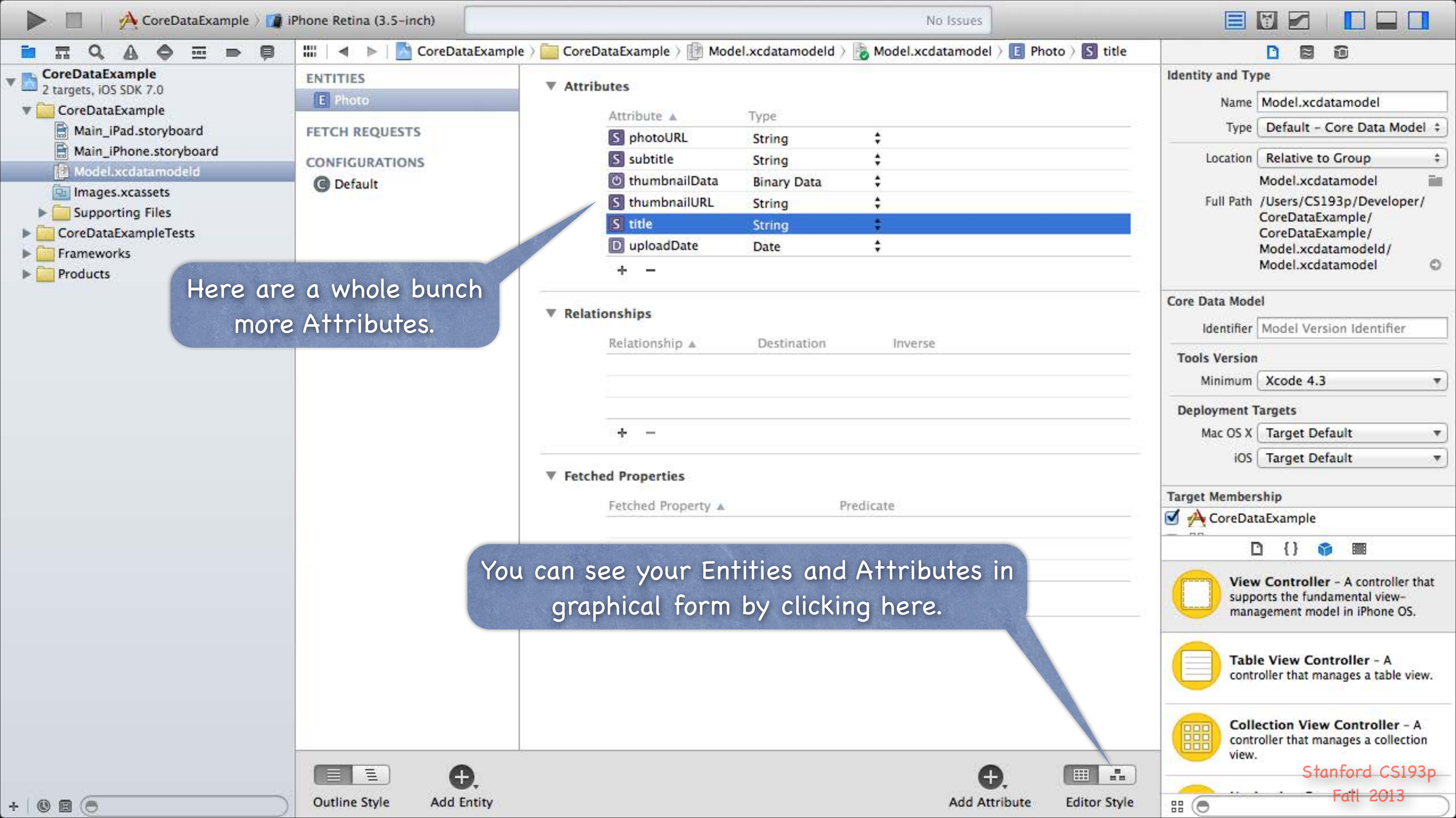

Then edit the name of the Attribute here.
We'll call this Attribute "title".
Attributes are analogous to "properties".
Notice that we have an error.
That's because our Attribute needs a type.
CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
FETCH REQUESTS
CONFIGURATIONS
Default
Model.xcdata...
Photo
attribute
Attributes
Attribute
Type
title
Undefined
Relationship
Destination
Inverse
Fetched Properties
Fetched Property
Predicate
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/CoreDataExample/CoreDataExample/Model.xcdatamodeld/Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Target Membership
CoreDataExample
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

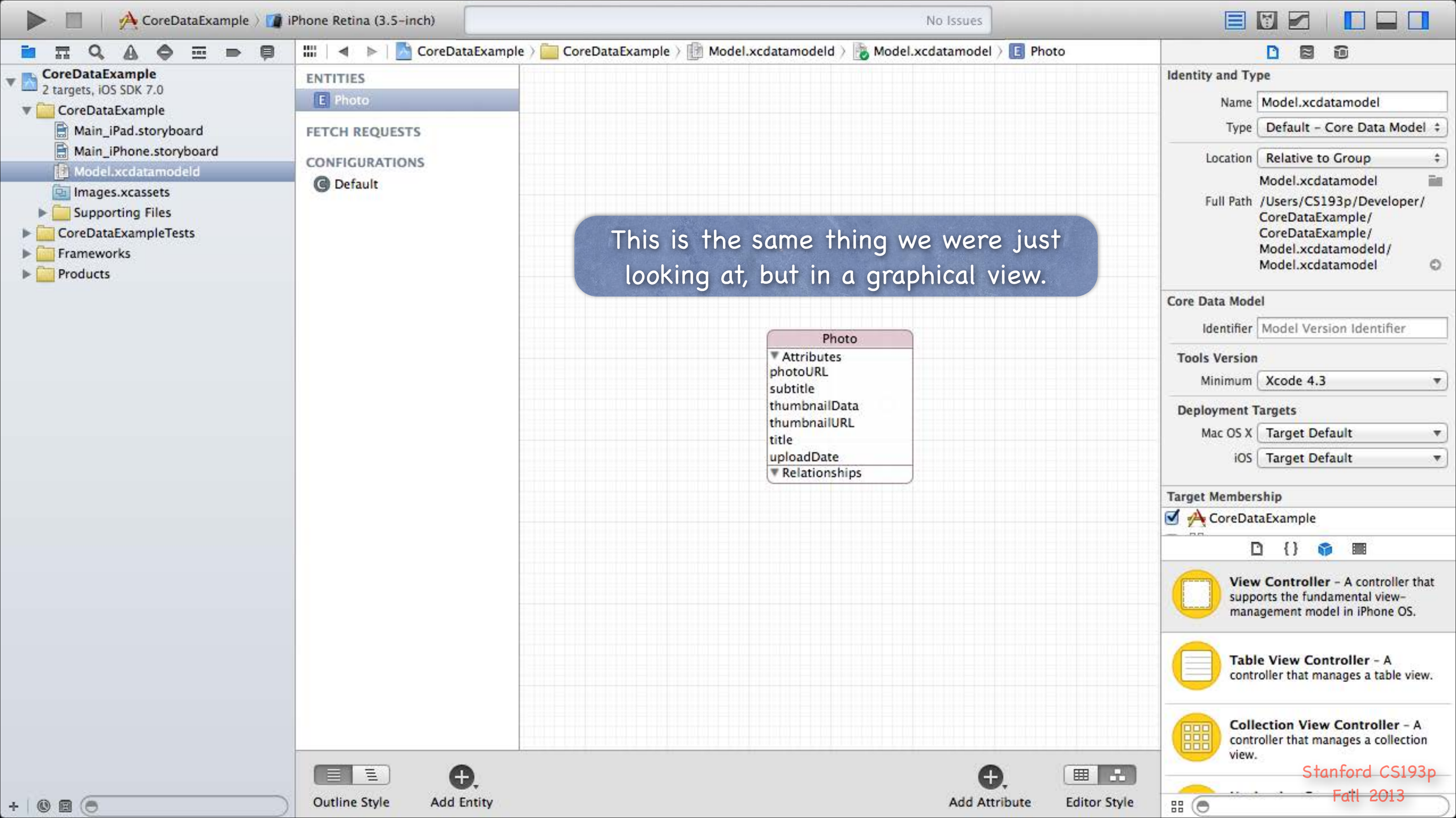


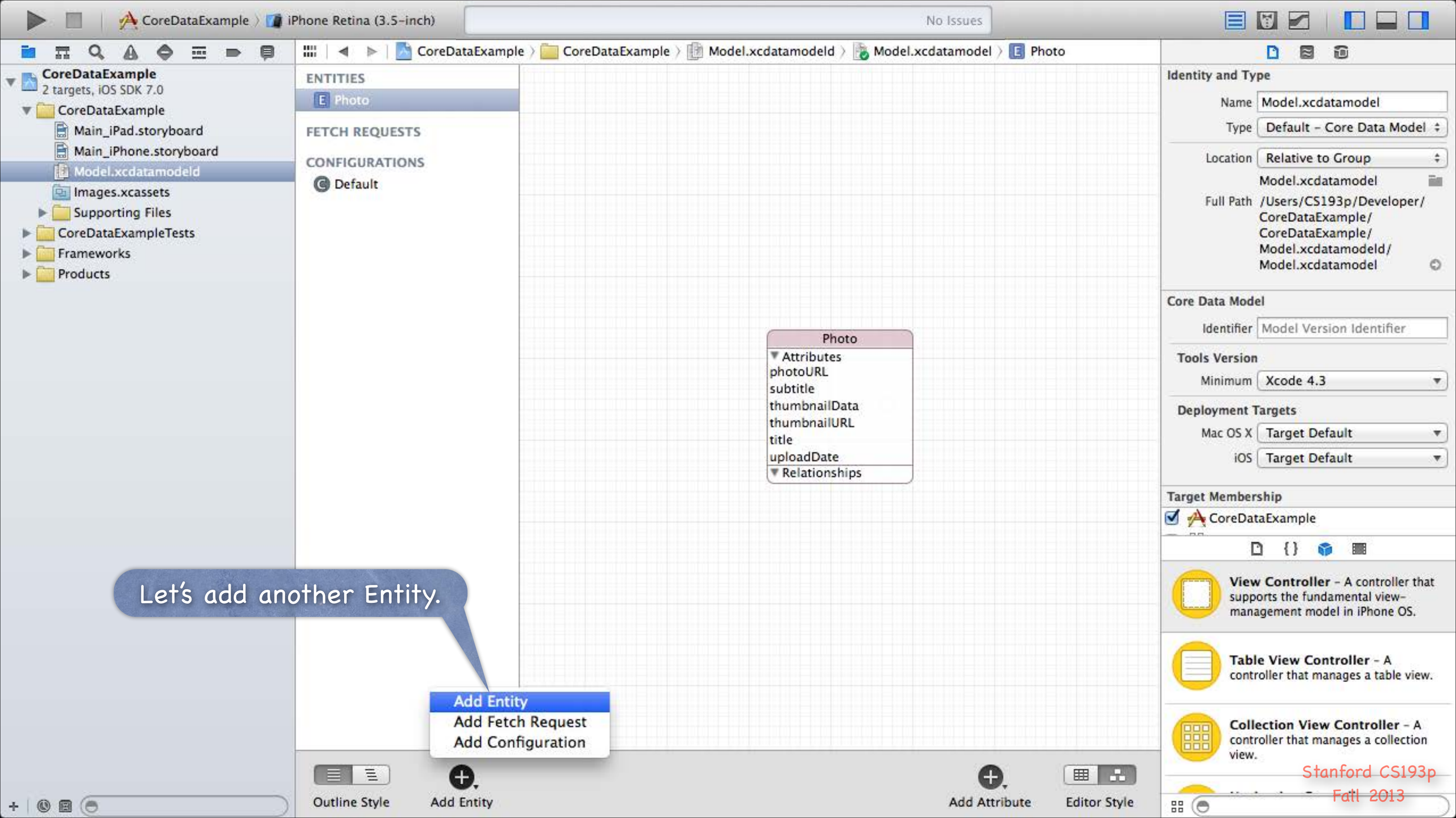
Set the type of the title Attribute.
All Attributes are objects.
Numeric ones are NSNumber.
Boolean is also NSNumber.
Binary Data is NSData.
Date is NSDate.
String is NSString.
Don't worry about Transformable.

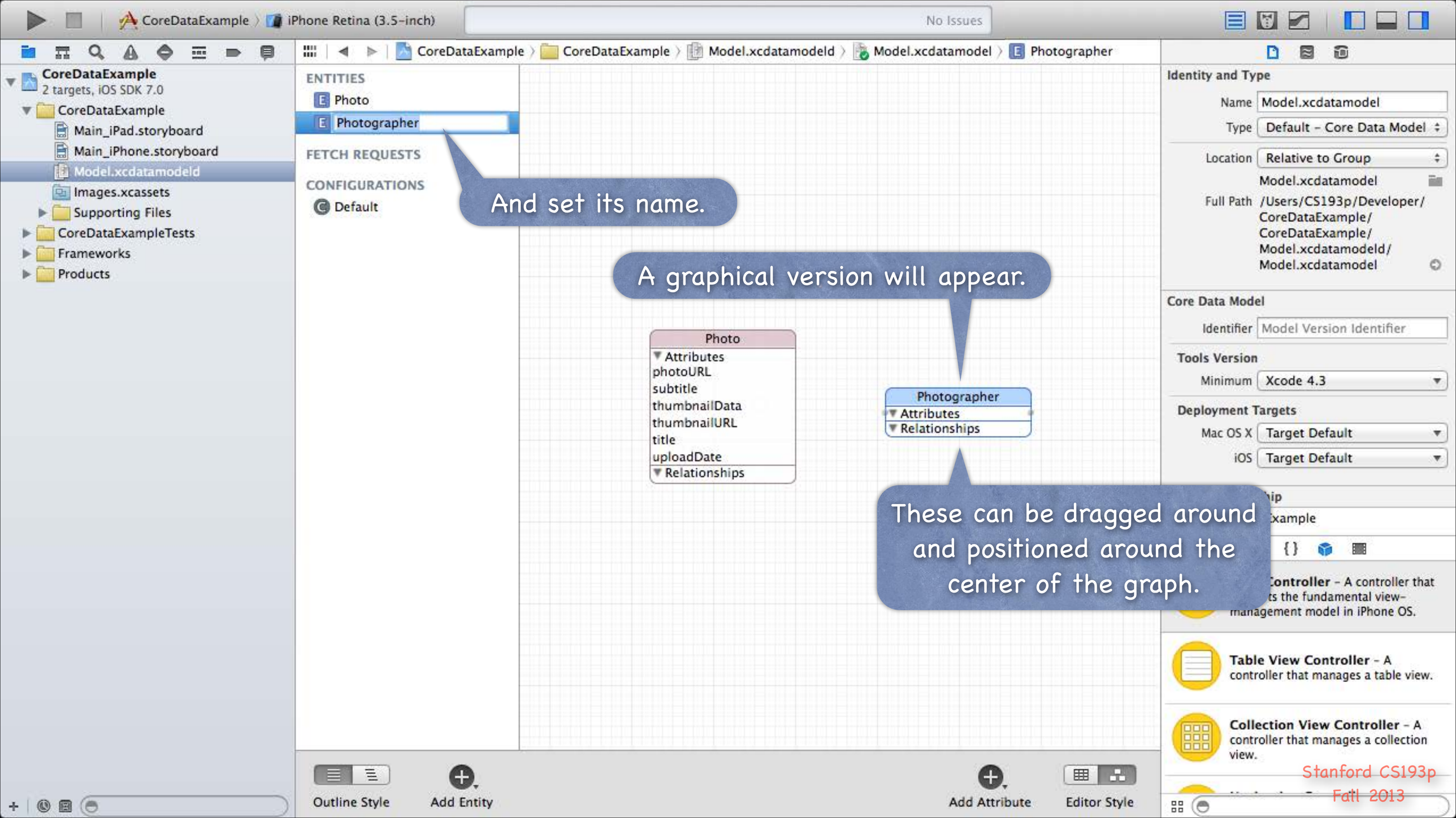
Attributes are accessed on our NSManagedObjects via the methods valueForKey: and setValue:forKey:.
Or we'll also see how we can access Attributes as @propertys.

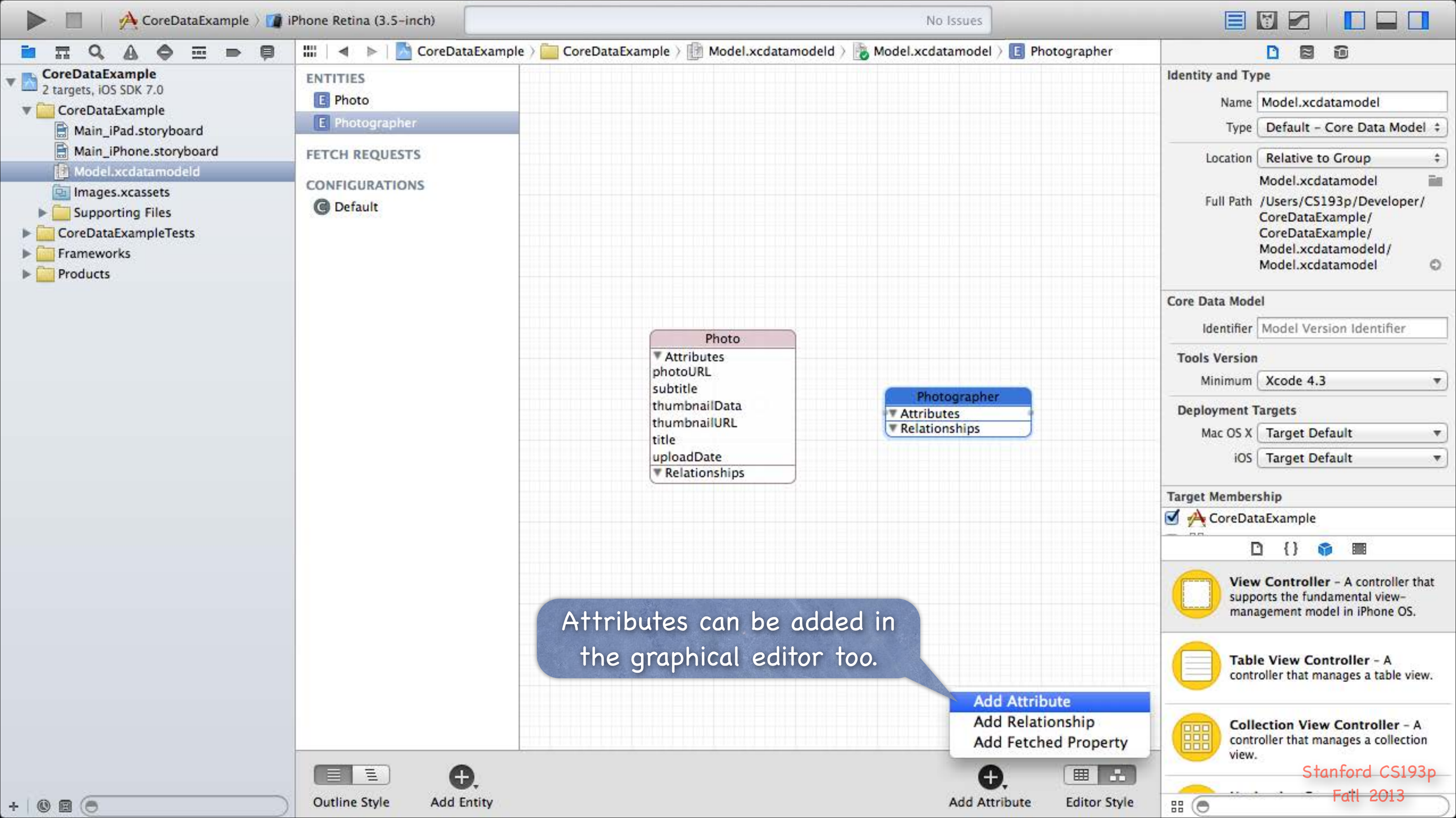

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
Photo
title
No more error!
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
FETCH REQUESTS
CONFIGURATIONS
Default
Attributes
Attribute
Type
title
String
Relationships
Relationship
Destination
Inverse
Fetched Properties
Fetched Property
Predicate
Type
Default – Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/
CoreDataExample/
CoreDataExample/
Model.xcdatamodeld/
Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Mac OS X
Target Default
iOS
Target Default
Target Membership
CoreDataExample
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

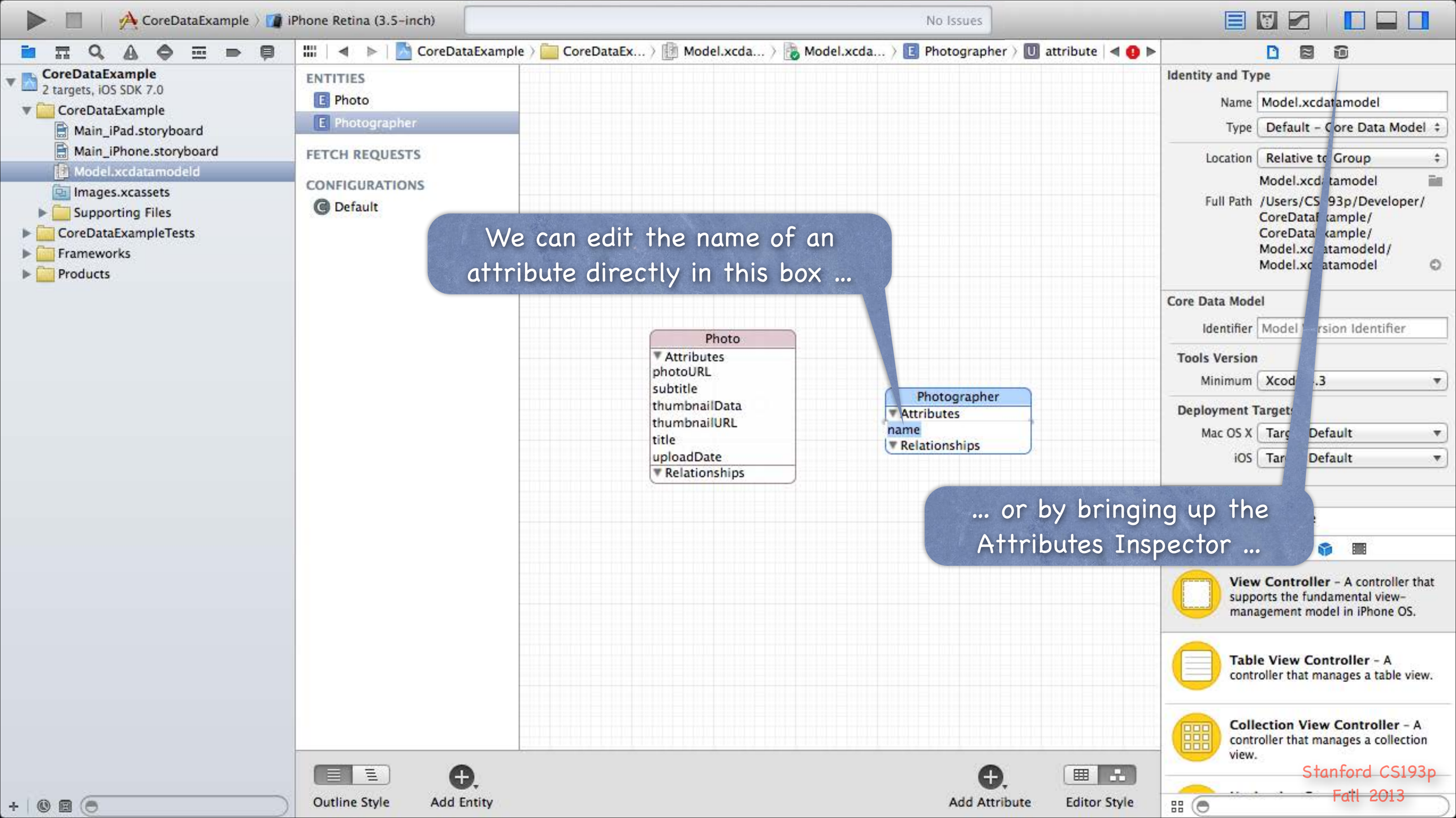

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
FETCH REQUESTS
CONFIGURATIONS
Default
Attributes
Attribute
Type
photoURL String
subtitle String
thumbnailData Binary Data
thumbnailURL String
title String
uploadDate Date
Relationships
Relationship
Destination
Inverse
Fetched Properties
Fetched Property
Predicate
Here are a whole bunch more Attributes.
You can see your Entities and Attributes in graphical form by clicking here.
Outline Style
Add Entity
Add Attribute
Editor Style
Identity and Type
Name Model.xcdatamodel
Type Default - Core Data Model
Location Relative to Group
Model.xcdatamodel
Full Path /Users/CS193p/Developer/CoreDataExample/CoreDataExample/Model.xcdatamodeld/Model.xcdatamodel
Core Data Model
Identifier Model Version Identifier
Tools Version
Minimum Xcode 4.3
Deployment Targets
Mac OS X Target Default
iOS Target Default
Target Membership
CoreDataExample
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Stanford CS193p
Fall 2013

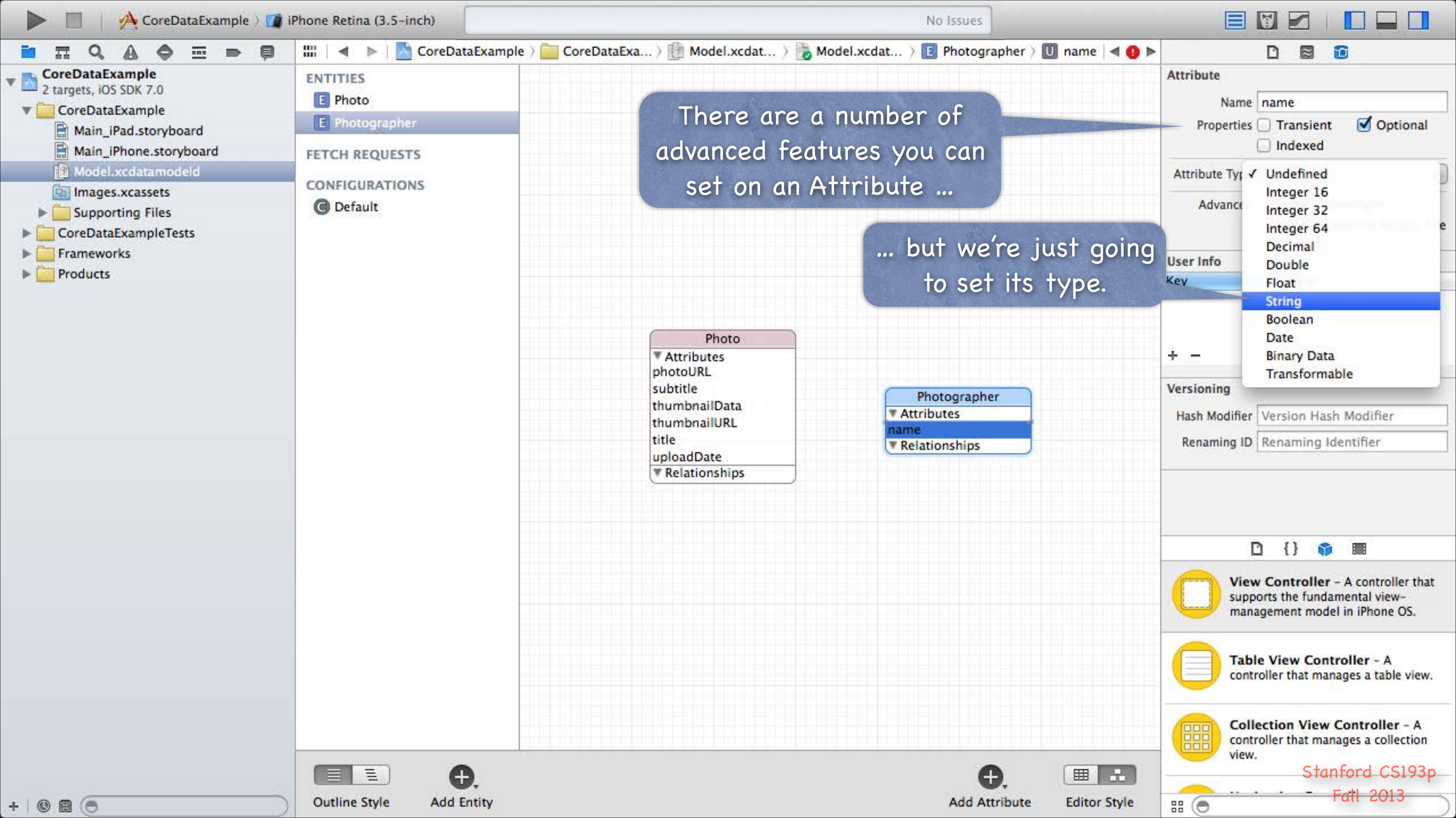

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
Photo
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
FETCH REQUESTS
CONFIGURATIONS
Default
This is the same thing we were just looking at, but in a graphical view.
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/CoreDataExample/CoreDataExample/Model.xcdatamodeld/Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Mac OS X
Target Default
iOS
Target Default
Target Membership
CoreDataExample
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

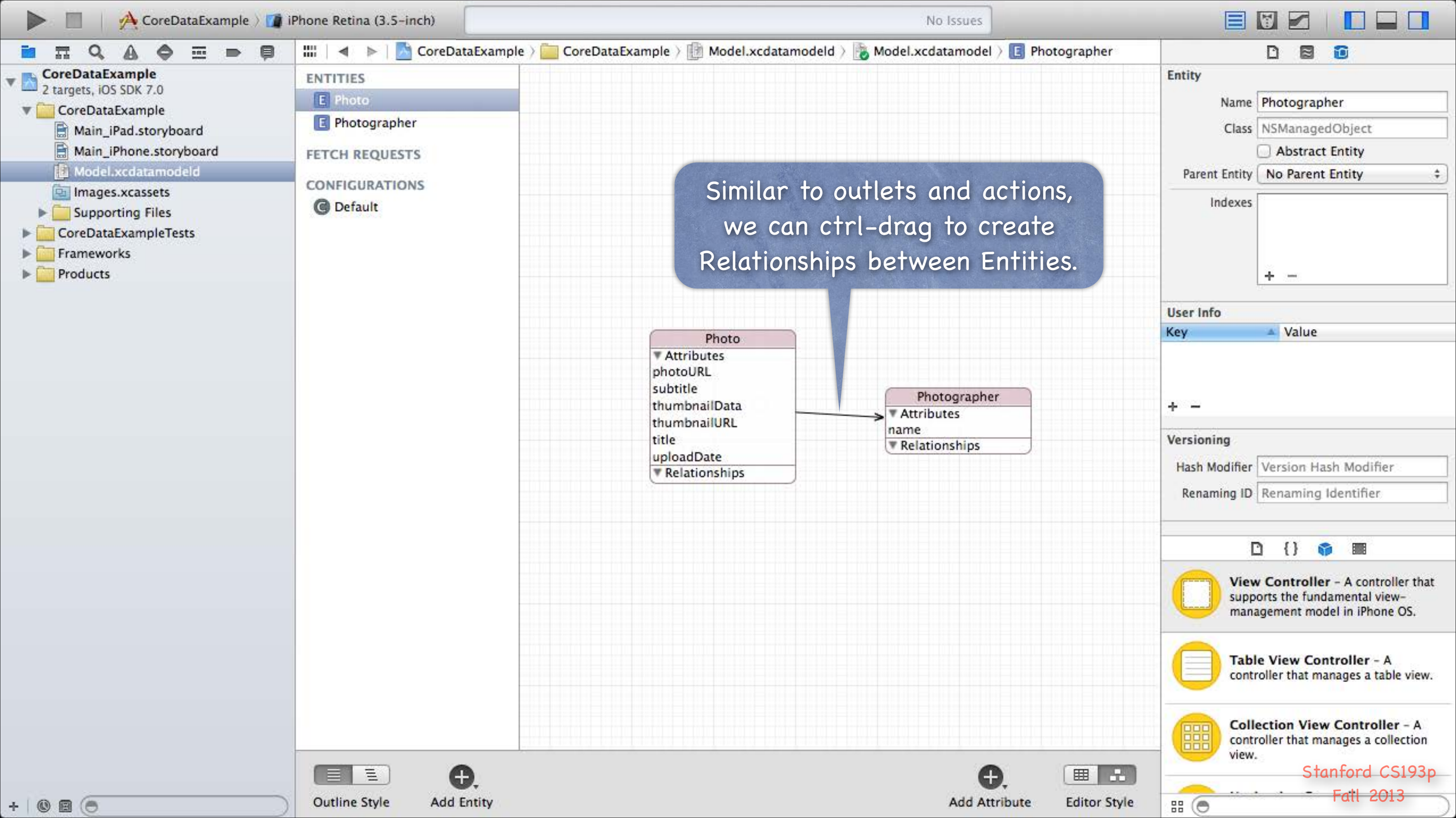

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
CoreDataExample › CoreDataExample › Model.xcdatamodeld › Model.xcdatamodel › Photo
ENTITIES
Photo
FETCH REQUESTS
CONFIGURATIONS
Default
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
Identity and Type
Name Model.xcdatamodel
Type Default - Core Data Model
Location Relative to Group
Model.xcdatamodel
Full Path /Users/CS193p/Developer/CoreDataExample/CoreDataExample/Model.xcdatamodeld/Model.xcdatamodel
Core Data Model
Identifier Model Version Identifier
Tools Version
Minimum Xcode 4.3
Deployment Targets
Mac OS X Target Default
iOS Target Default
Target Membership
CoreDataExample
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Let's add another Entity.
Add Entity
Add Fetch Request
Add Configuration
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

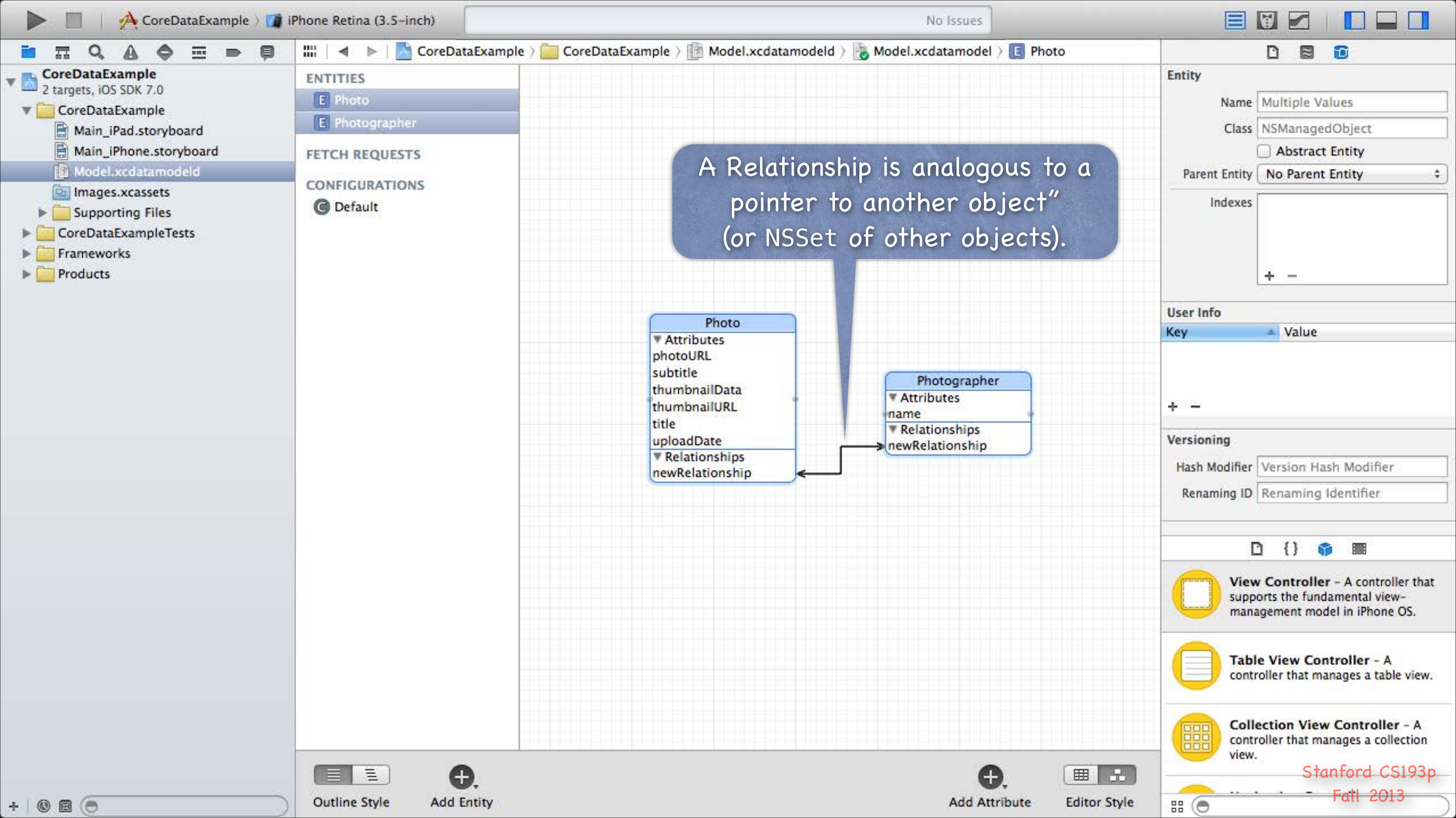

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
Photographer
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
And set its name.
A graphical version will appear.
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
Photographer
Attributes
Relationships
These can be dragged around and positioned around the center of the graph.
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Model.xcdatamodel
Full Path
/Users/CS193p/Developer/CoreDataExample/CoreDataExample/Model.xcdatamodeld/Model.xcdatamodel
Core Data Model
Identifier
Model Version Identifier
Tools Version
Minimum
Xcode 4.3
Deployment Targets
Mac OS X
Target Default
iOS
Target Default
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

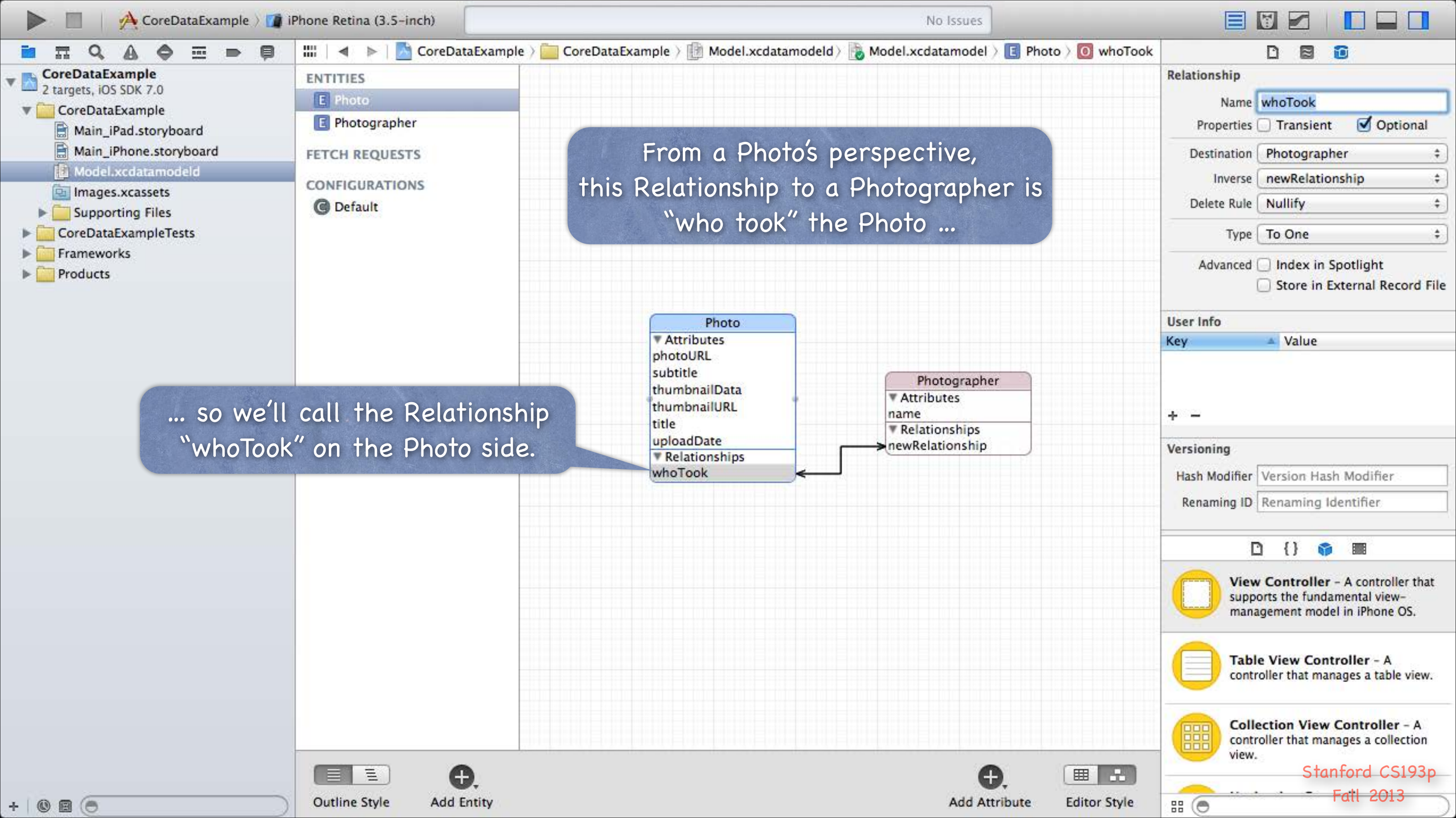

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
CoreDataExample › CoreDataExample › Model.xcdatamodeld › Model.xcdatamodel › Photographer
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
Photographer
Attributes
Relationships
Attributes can be added in the graphical editor too.
Add Attribute
Add Relationship
Add Fetched Property
Outline Style
Add Entity
Add Attribute
Editor Style
Identity and Type
Name Model.xcdatamodel
Type Default - Core Data Model
Location Relative to Group
Model.xcdatamodel
Full Path /Users/CS193p/Developer/CoreDataExample/CoreDataExample/Model.xcdatamodeld/Model.xcdatamodel
Core Data Model
Identifier Model Version Identifier
Tools Version
Minimum Xcode 4.3
Deployment Targets
Mac OS X Target Default
iOS Target Default
Target Membership
CoreDataExample
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Stanford CS193p
Fall 2013

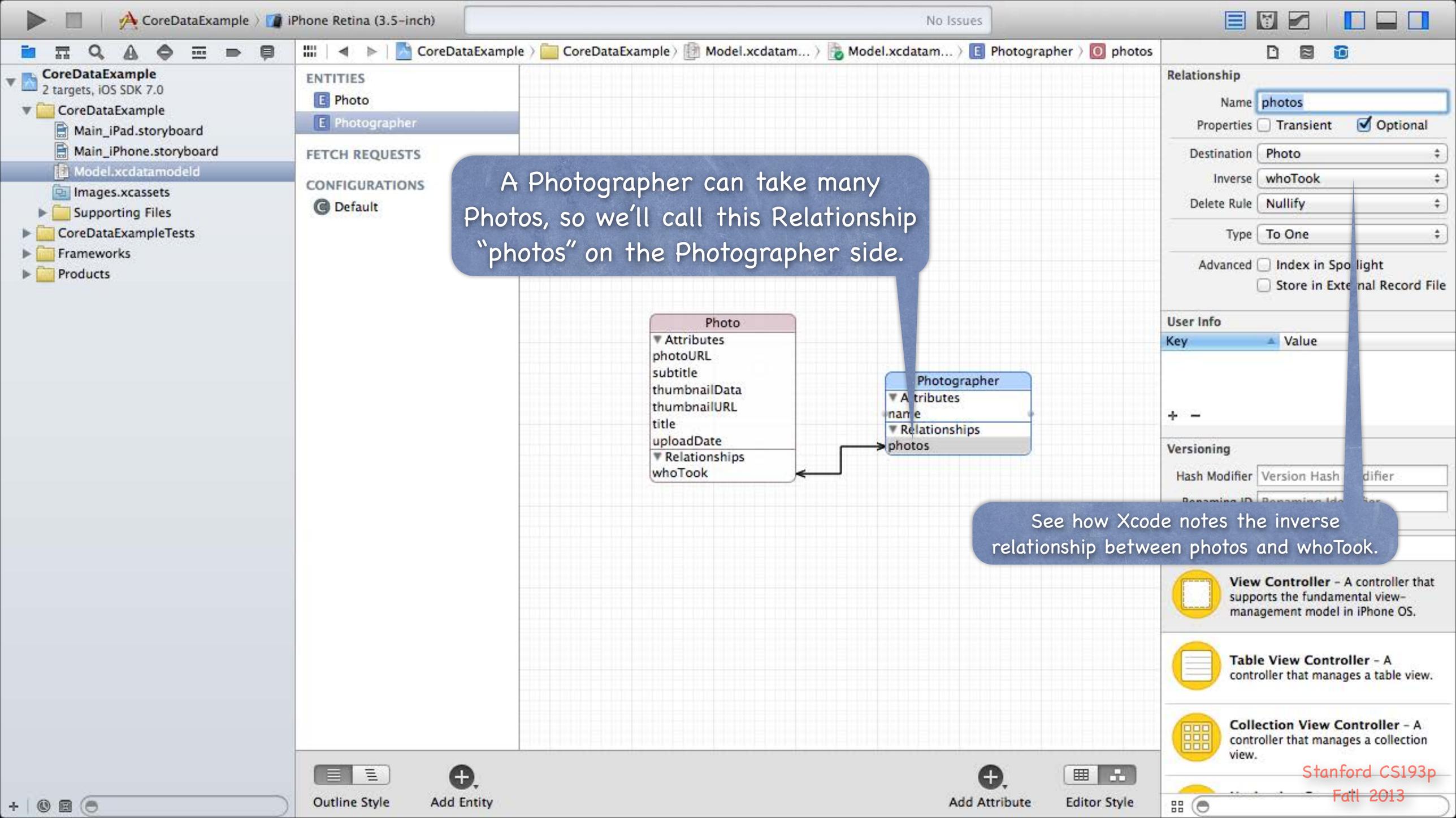

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
We can edit the name of an attribute directly in this box ...
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
Photographer
Attributes
name
Relationships
... or by bringing up the Attributes Inspector ...
Identity and Type
Name
Model.xcdatamodel
Type
Default - Core Data Model
Location
Relative to Group
Full Path
Core Data Model
Identifier
Tools Version
Minimum
Deployment Targets
Mac OS X
iOS
View Controller - A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller - A controller that manages a table view.
Collection View Controller - A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

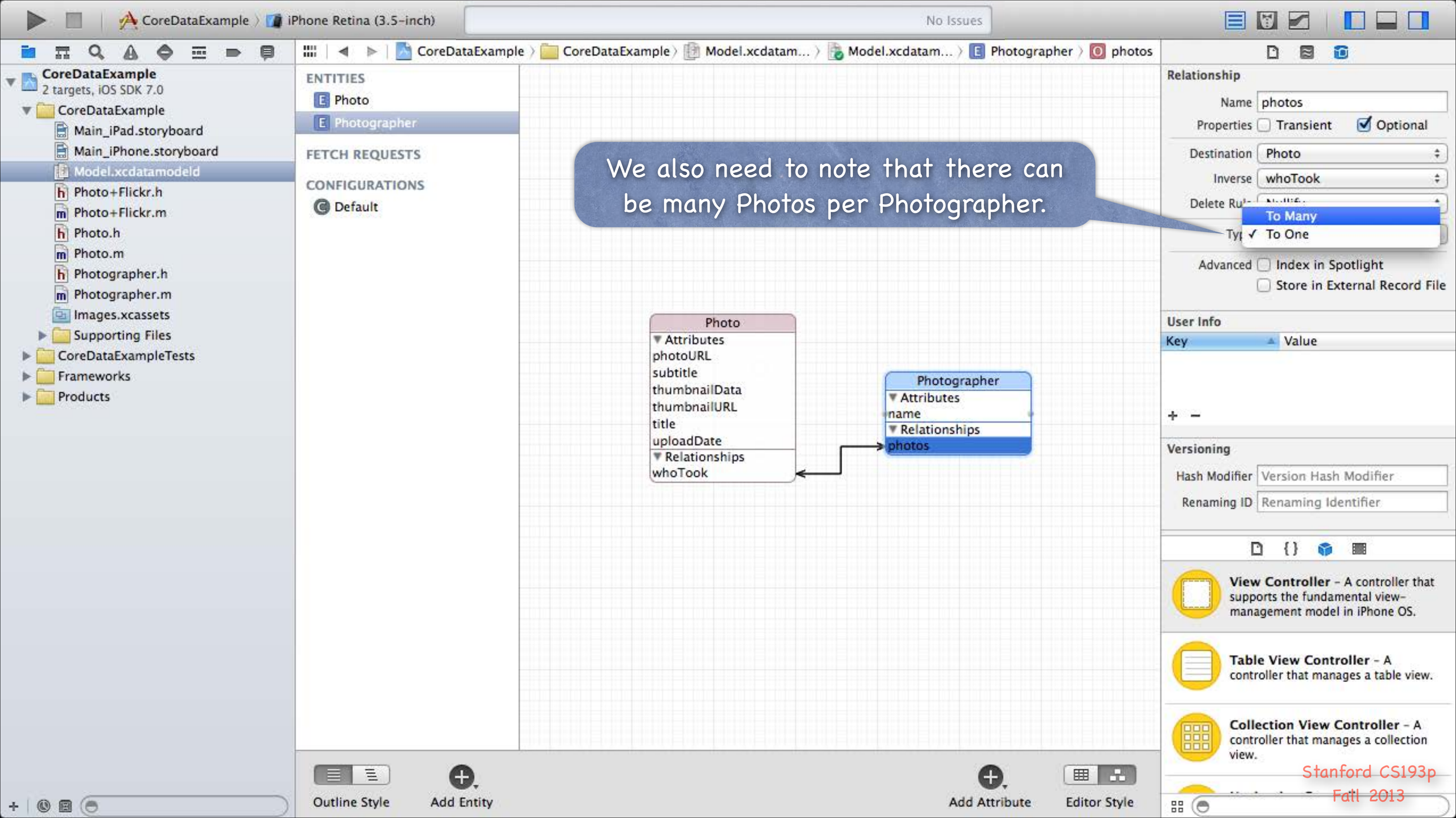

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
There are a number of advanced features you can set on an Attribute ...
... but we're just going to set its type.
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
Photographer
Attributes
name
Relationships
Attribute
Name
name
Properties
Transient
Optional
Indexed
Attribute Type
Undefined
Integer 16
Integer 32
Integer 64
Decimal
Double
Float
String
Boolean
Date
Binary Data
Transformable
User Info
Key
Versioning
Hash Modifier
Version Hash Modifier
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

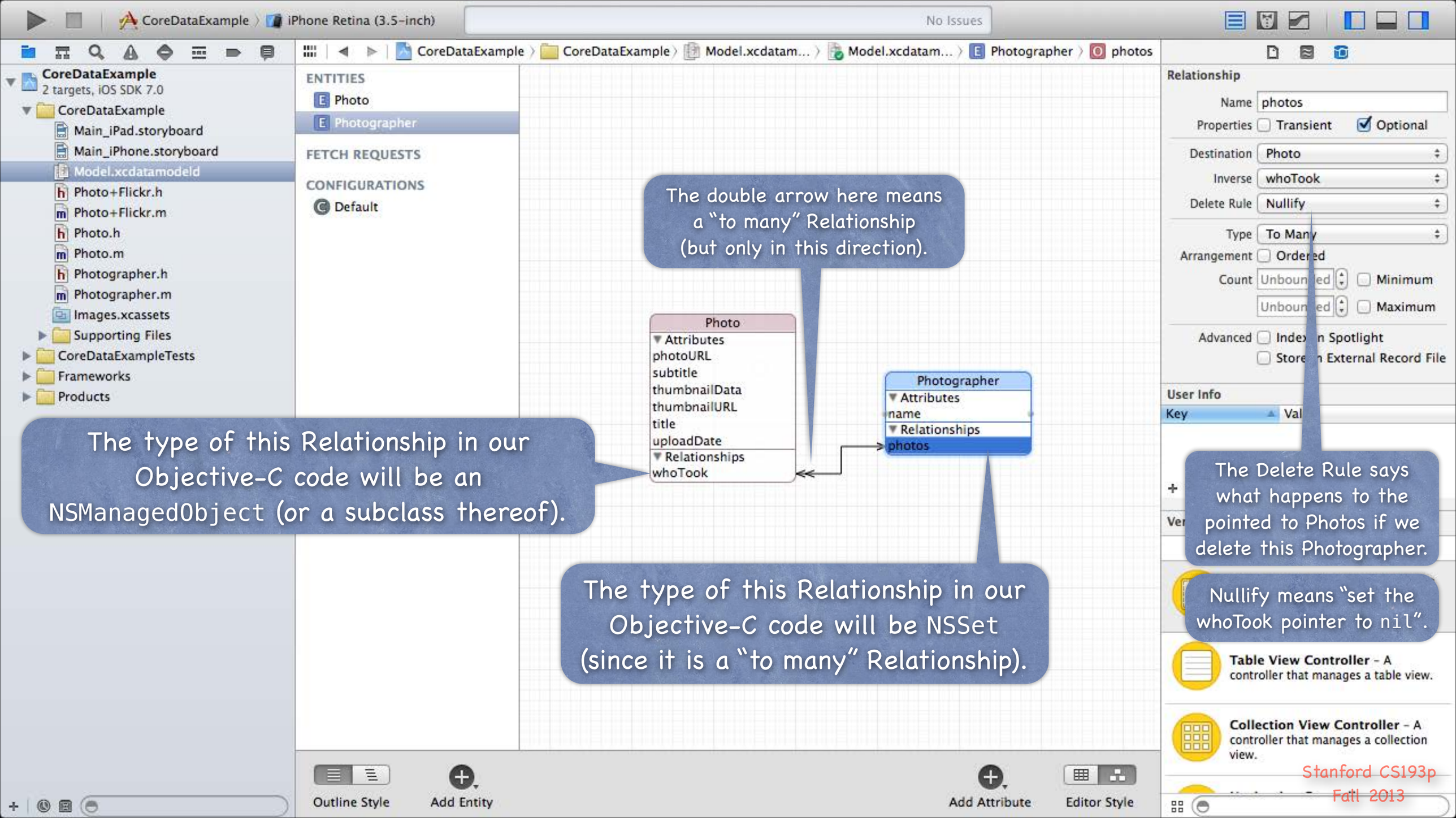

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
Photographer
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
Similar to outlets and actions, we can ctrl-drag to create Relationships between Entities.
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
Photographer
Attributes
name
Relationships
Entity
Name
Photographer
Class
NSManagedObject
Abstract Entity
Parent Entity
No Parent Entity
Indexes
User Info
Key
Value
Versioning
Hash Modifier
Version Hash Modifier
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
Photo
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
A Relationship is analogous to a pointer to another object" (or NSSet of other objects).
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
newRelationship
Photographer
Attributes
name
Relationships
newRelationship
Entity
Name
Multiple Values
Class
NSManagedObject
Abstract Entity
Parent Entity
No Parent Entity
Indexes
User Info
Key
Value
Versioning
Hash Modifier
Version Hash Modifier
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
From a Photo's perspective, this Relationship to a Photographer is "who took" the Photo ...
... so we'll call the Relationship "whoTook" on the Photo side.
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
whoTook
Photographer
Attributes
name
Relationships
newRelationship
Relationship
Name whoTook
Properties Transient Optional
Destination Photographer
Inverse newRelationship
Delete Rule Nullify
Type To One
Advanced Index in Spotlight
Store in External Record File
User Info
Key
Value
Versioning
Hash Modifier Version Hash Modifier
Renaming ID Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
A Photographer can take many Photos, so we'll call this Relationship "photos" on the Photographer side.
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
whoTook
Photographer
Attributes
name
Relationships
photos
Relationship
Name photos
Properties Transient Optional
Destination Photo
Inverse whoTook
Delete Rule Nullify
Type To One
Advanced Index in Spotlight
Store in External Record File
User Info
Key
Value
Versioning
Hash Modifier
See how Xcode notes the inverse relationship between photos and whoTook.
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo+Flickr.h
Photo+Flickr.m
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
CoreDataExample
CoreDataExample
Model.xcdatam...
Model.xcdatam...
Photographer
photos
We also need to note that there can be many Photos per Photographer.
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
whoTook
Photographer
Attributes
name
Relationships
photos
Relationship
Name photos
Properties Transient Optional
Destination Photo
Inverse whoTook
To Many
To One
Advanced Index in Spotlight
Store in External Record File
User Info
Key
Value
Versioning
Hash Modifier Version Hash Modifier
Renaming ID Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo+Flickr.h
Photo+Flickr.m
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
The double arrow here means a "to many" Relationship (but only in this direction).
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
whoTook
Photographer
Attributes
name
Relationships
photos
The type of this Relationship in our Objective-C code will be an NSManagedObject (or a subclass thereof).
The type of this Relationship in our Objective-C code will be NSSet (since it is a "to many" Relationship).
Relationship
Name photos
Properties Transient Optional
Destination Photo
Inverse whoTook
Delete Rule Nullify
Type To Many
Arrangement Ordered
Count Minimum
Maximum
Advanced
User Info
Key
The Delete Rule says what happens to the pointed to Photos if we delete this Photographer.
Nullify means "set the whoTook pointer to nil".
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

Core Data

- There are lots of other things you can do
  - But we are going to focus on creating Entities, Attributes and Relationships.
- So how do you access all of this stuff in your code?
  - You need an `NSManagedObjectContext`.
  - It is the hub around which all Core Data activity turns.
- How do I get one?
  - There are two ways …
  1. Create a `UIManagedDocument` and ask for its `managedObjectContext` (a `@property`).
  2. Click the "Use Core Data" button when you create a project (only works with certain templates) (then your `AppDelegate` will have a `managedObjectContext` `@property`).
  - If you study what the template (e.g. Master-Detail template) does, you'll get an idea how it works.
  - We're going to focus on doing the first one.

UIManagedDocument

- UIManagedDocument

  It inherits from UIDocument which provides a lot of mechanism for the management of storage.
  If you use UIManagedDocument, you'll be on the fast-track to iCloud support.
  Think of a UIManagedDocument as simply a container for your Core Data database.

- Creating a UIManagedDocument

```
NSFileManager *fileManager = [NSFileManager defaultManager];
NSURL *documentsDirectory = [[fileManager URLsForDirectory:NSDocumentDirectory
                                                 inDomains:NSUserDomainMask] firstObject];
NSString *documentName = @"MyDocument";
NSURL *url = [documentsDirectory URLByAppendingPathComponent:documentName];
UIManagedDocument *document = [[UIManagedDocument alloc] initWithFileURL:url];
```

  This creates the UIManagedDocument instance, but does not open nor create the underlying file.

UIManagedDocument

- How to open or create a UIManagedDocument

  First, check to see if the UIManagedDocument's underlying file exists on disk …

```
BOOL fileExists = [[NSFileManager defaultManager] fileExistsAtPath:[url path]];
```

  … if it does, open the document using …

```
[document openWithCompletionHandler:^(BOOL success) { /* block to execute when open */ }];
```

  … if it does not, create the document using …

```
[document saveToURL:url                // could (should?) use document.fileURL property here
   forSaveOperation:UIDocumentSaveForCreating
   competionHandler:^(BOOL success) { /* block to execute when create is done */ }];
```

- What is that completionHander?

  Just a block of code to execute when the open/save completes.

  That's needed because the open/save is asynchronous (i.e. happens on its own queue).

  Do not ignore this fact!

UIManagedDocument

- Example

```
self.document = [[UIManagedDocument alloc] initWithFileURL:(URL *)url];
if ([[NSFileManager defaultManager] fileExistsAtPath:[url path]]) {
    [document openWithCompletionHandler:^(BOOL success) {
        if (success) [self documentIsReady];
        if (!success) NSLog(@“couldn’t open document at %@”, url);
    }];
} else {
    [document saveToURL:url forSaveOperation:UIDocumentSaveForCreating
      completionHandler:^(BOOL success) {
        if (success) [self documentIsReady];
        if (!success) NSLog(@“couldn’t create document at %@”, url);
    }];
}
// can’t do anything with the document yet (do it in documentIsReady).
```

UIManagedDocument

- Once document is open/created, you can start using it

  But you might want to check the documentState when you do …

```
- (void)documentIsReady
{
    if (self.document.documentState == UIDocumentStateNormal) {
        // start using document
    }
}
```

- Other documentStates

  UIDocumentStateClosed (you haven't done the open or create yet)

  UIDocumentStateSavingError (success will be NO in completion handler)

  UIDocumentStateEditingDisabled (temporary situation, try again)

  UIDocumentStateInConflict (e.g., because some other device changed it via iCloud)

  We don't have time to address these (you can ignore in homework), but know that they exist.

UIManagedDocument

- Okay, document is ready to use, now what?

  Now you can get a managedObjectContext from it and use it to do Core Data stuff!

```
- (void)documentIsReady
{
    if (self.document.documentState == UIDocumentStateNormal) {
        NSManagedObjectContext *context = self.document.managedObjectContext;
        // start doing Core Data stuff with context
    }
}
```

  Okay, just a couple of more UIManagedDocument things before we start using that context …

UIManagedDocument

- Saving the document

  UIManagedDocuments AUTOSAVE themselves!

  However, if, for some reason you wanted to manually save (asynchronous!) …

```
[document saveToURL:document.fileURL
   forSaveOperation:UIDocumentSaveForOverwriting
   competionHandler:^(BOOL success) { /* block to execute when save is done */ }];
```

  Note that this is almost identical to creation (just UIDocumentSaveForOverwriting is different).

  This is a UIKit class and so this method must be called on the main queue.

- Closing the document

  Will automatically close if there are no strong pointers left to it.

  But you can explicitly close with (asynchronous!) …

```
[self.document closeWithCompletionHandler:^(BOOL success) {
    if (!success) NSLog(@“failed to close document %@”, self.document.localizedName);
}];
```

  UIManagedDocument's localizedName method …

```
@property (strong) NSString *localizedName; // suitable for UI (but only valid once saved)
```

UIManagedDocument

- Multiple instances of `UIManagedDocument` on the same document

  This is perfectly legal, but understand that they will not share an `NSManagedObjectContext`.
  Thus, changes in one will not automatically be reflected in the other.

  You'll have to refetch in other `UIManagedDocuments` after you make a change in one.

  Conflicting changes in two different `UIManagedDocuments` would have to be resolved by you!
  It's exceedingly rare to have two "writing" instances of `UIManagedDocument` on the same file.
  But a single writer and multiple readers? Less rare. But you need to know when to refetch.

  You can watch (via "radio station") other documents' `managedObjectContexts` (then refetch).
  Or you can use a single `UIManagedDocument` instance (per actually document) throughout.

NSNotification

- How would you watch a document's managedObjectContext?

```
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [center addObserver:self
               selector:@selector(contextChanged:)
                   name:NSManagedObjectContextDidSaveNotification
                 object:document.managedObjectContext];  // don't pass nil here!
}
- (void)viewWillDisappear:(BOOL)animated
{
    [center removeObserver:self
                      name:NSManagedObjectContextDidSaveNotification
                    object:document.managedObjectContext];
    [super viewWillDisappear:animated];
}
```

NSNotification

- NSManagedObjectContextDidSaveNotification

```
- (void)contextChanged:(NSNotification *)notification
{
    // The notification.userInfo object is an NSDictionary with the following keys:
    NSInsertedObjectsKey // an array of objects which were inserted
    NSUpdatedObjectsKey  // an array of objects whose attributes changed
    NSDeletedObjectsKey  // an array of objects which were deleted
}
```

- Merging changes

If you get notified that another NSManagedObjectContext has changed your database …

… you can just refetch (if you haven't changed anything in your NSMOC, for example).

… or you can use the NSManagedObjectContext method:

```
- (void)mergeChangesFromContextDidSaveNotification:(NSNotification *)notification;
```

Core Data

- Okay, we have an `NSManagedObjectContext`, now what?
  - We grabbed it from an open `UIManagedDocument`'s `managedObjectContext` `@property`.
  - Now we use it to insert/delete objects in the database and query for objects in the database.

Core Data

- Inserting objects into the database

```
NSManagedObjectContext *context = aDocument.managedObjectContext;
NSManagedObject *photo =
    [NSEntityDescription insertNewObjectForEntityForName:@“Photo”
                                  inManagedObjectContext:context];
```

Note that this `NSEntityDescription` class method returns an `NSManagedObject` instance.
All objects in the database are represented by `NSManagedObjects` or subclasses thereof.

An instance of `NSManagedObject` is a manifestation of an Entity in our Core Data Model*.
Attributes of a newly-inserted object will start out `nil` (unless you specify a default in Xcode).

* i.e., the Data Model that we just graphically built in Xcode!

Core Data

- How to access Attributes in an NSManagedObject instance
  - You can access them using the following two NSKeyValueCoding protocol methods …
  - `- (id)valueForKey:(NSString *)key;`
  - `- (void)setValue:(id)value forKey:(NSString *)key;`
  - You can also use valueForKeyPath:/setValue:forKeyPath: and it will follow your Relationships!
- The key is an Attribute name in your data mapping
  - For example, @"thumbnailURL" or @"title".
- The value is whatever is stored (or to be stored) in the database
  - It'll be nil if nothing has been stored yet (unless Attribute has a default value in Xcode).
  - Note that all values are objects (numbers and booleans are NSNumber objects).
  - Binary data values are NSData objects.
  - Date values are NSDate objects.
  - "To-many" mapped relationships are NSSet objects (or NSOrderedSet if ordered).
  - Non-"to-many" relationships are NSManagedObjects.

Core Data

- Changes (writes) only happen in memory, until you save

  Remember, `UIManagedDocument` autosaves.

  When the document is saved, the context is saved and your changes get written to the database.

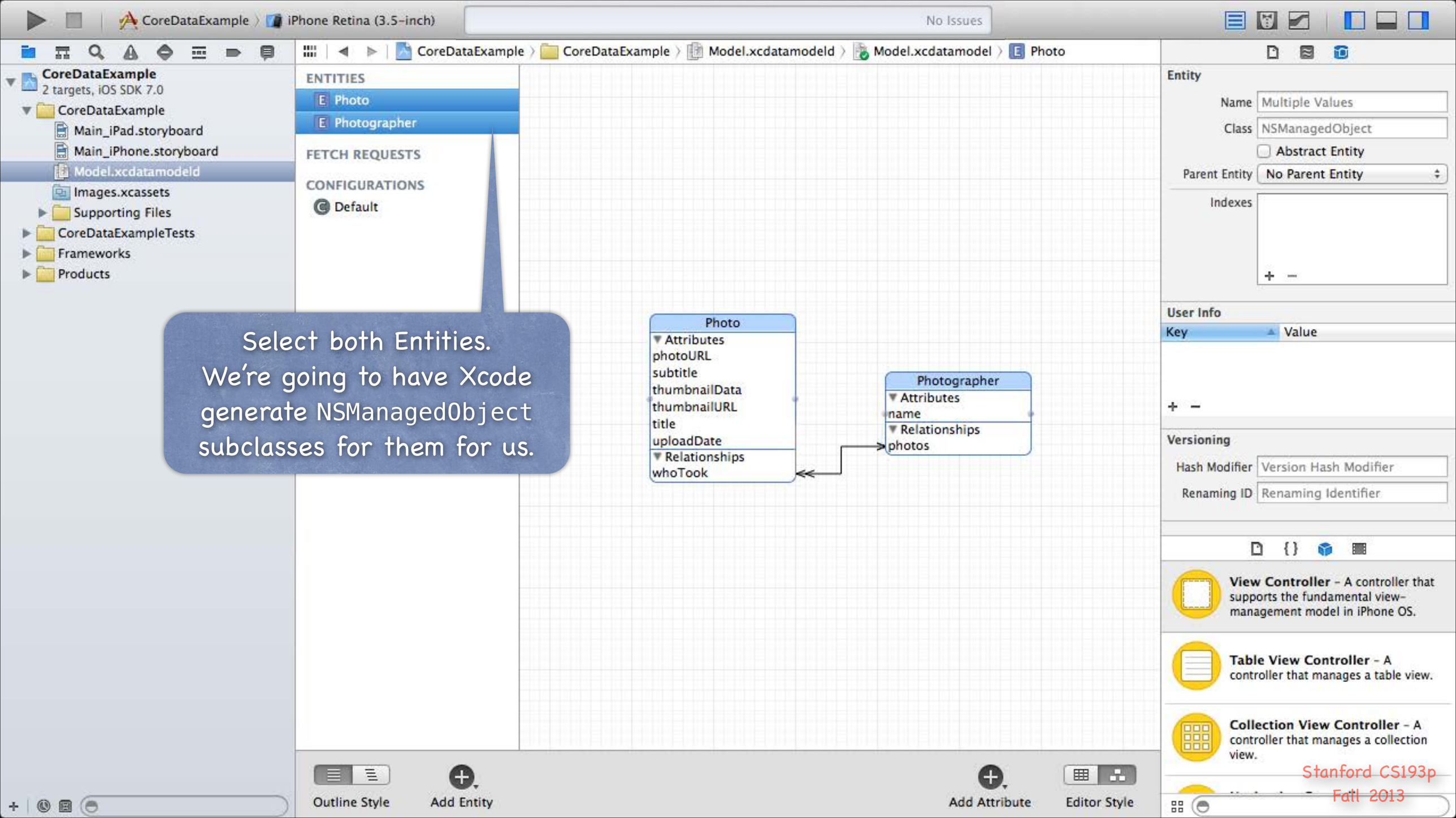
  `UIManagedDocumentDidSaveNotification` will be "broadcast" at that point.

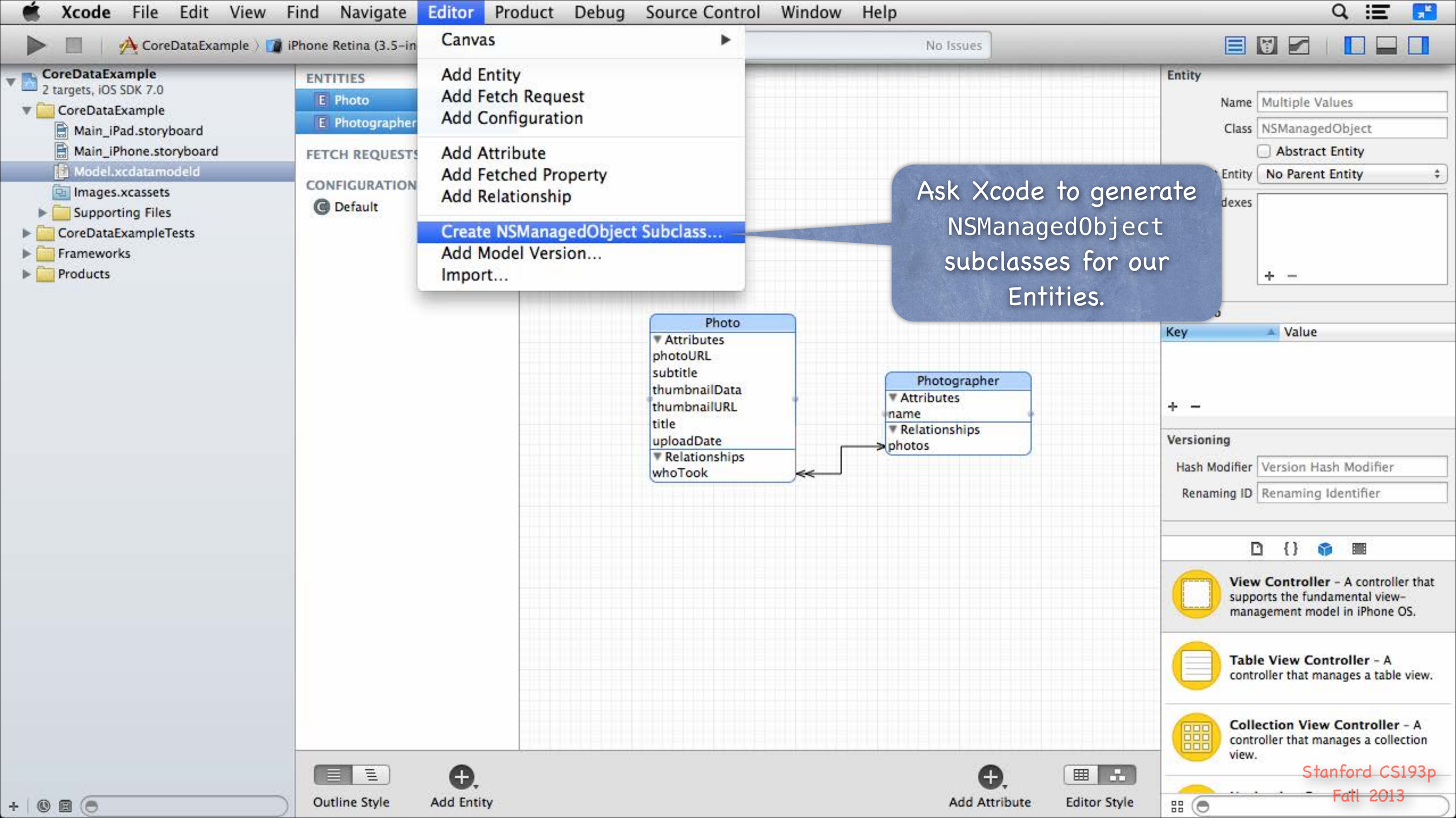
  Be careful during development where you press "Stop" in Xcode (sometimes autosave is pending).

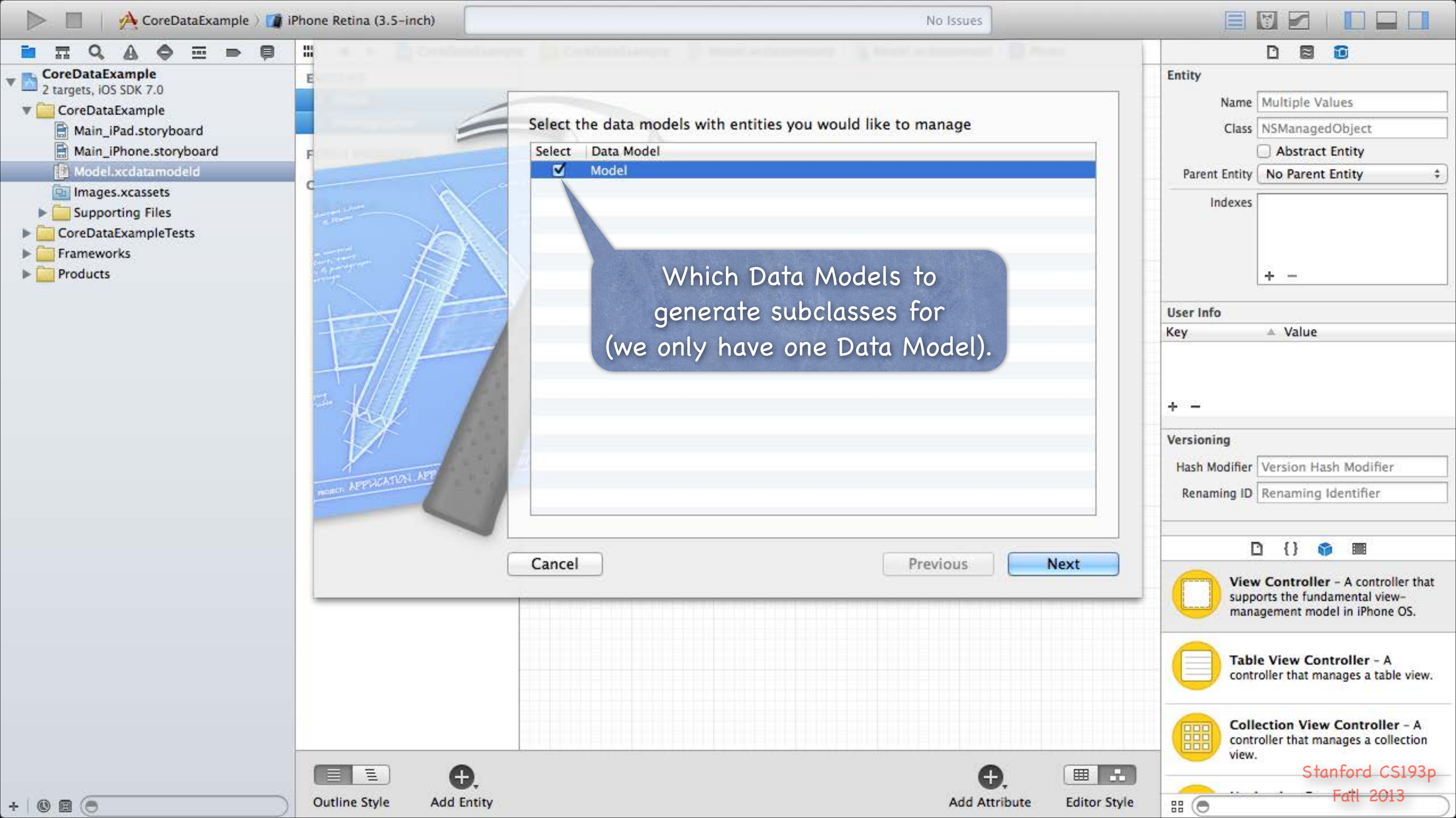
Core Data

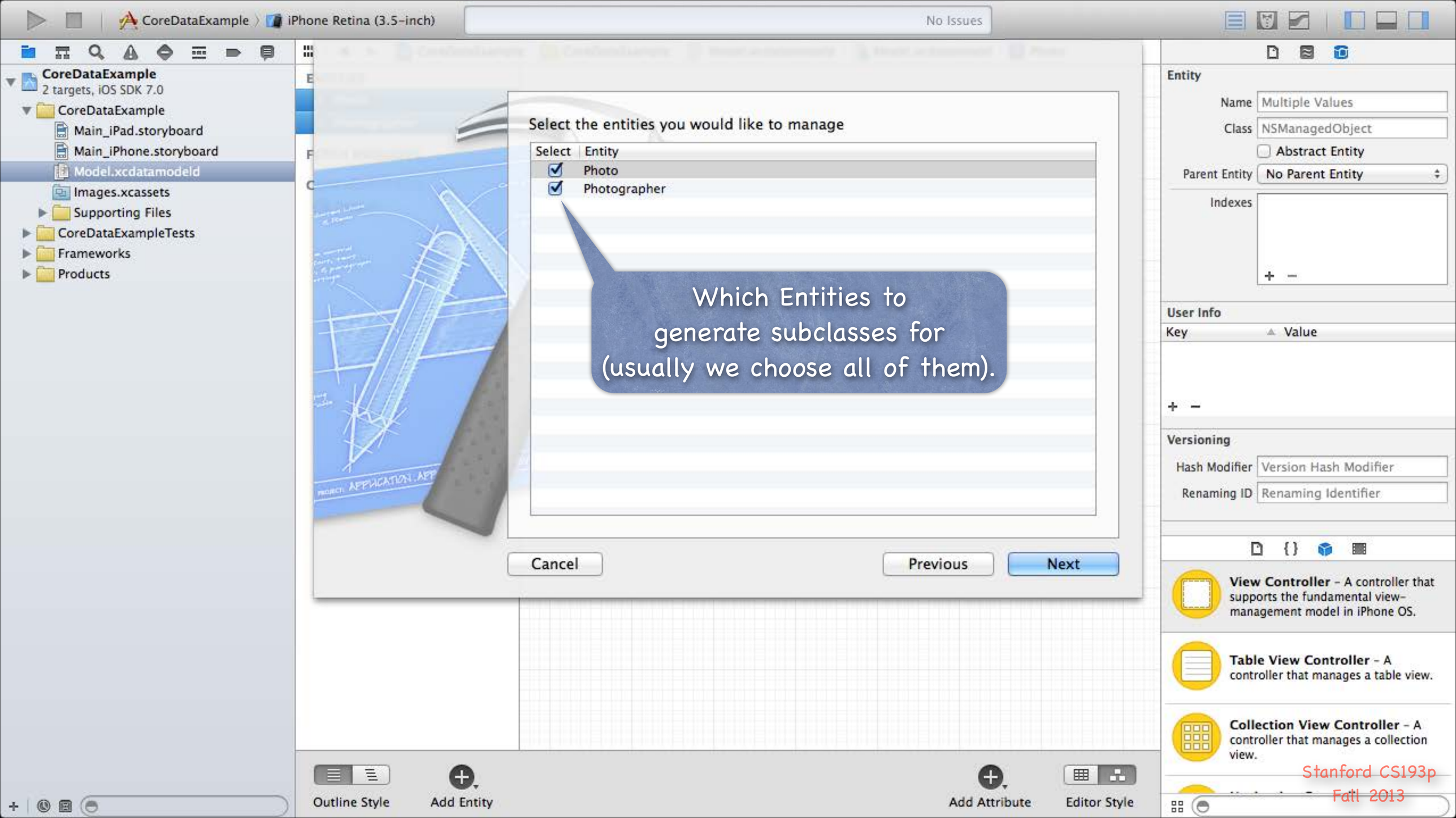
- But calling `valueForKey:`/`setValue:forKey:` is pretty ugly
  - There's no type-checking.
  - And you have a lot of literal strings in your code (e.g. `@"thumbnailURL"`)
- What we really want is to set/get using `@property`s!
- No problem … we just create a subclass of `NSManagedObject`
  - The subclass will have `@property`s for each attribute in the database.
  - We name our subclass the same name as the Entity it matches (not strictly required, but do it).

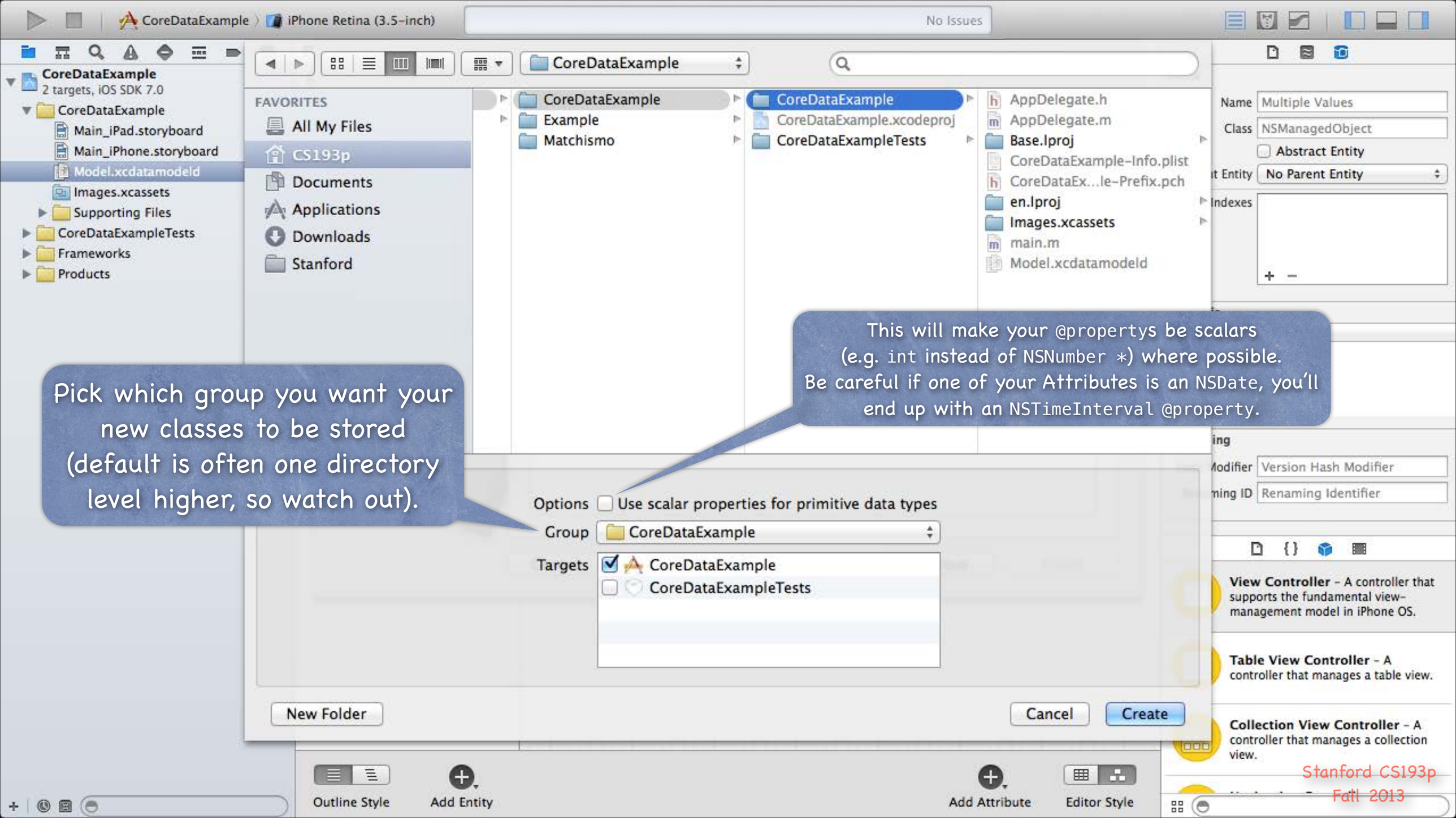
  And, as you might imagine, we can get Xcode to generate both the header file `@property` entries, and the corresponding implementation code (which is not `@synthesize`, so watch out!).

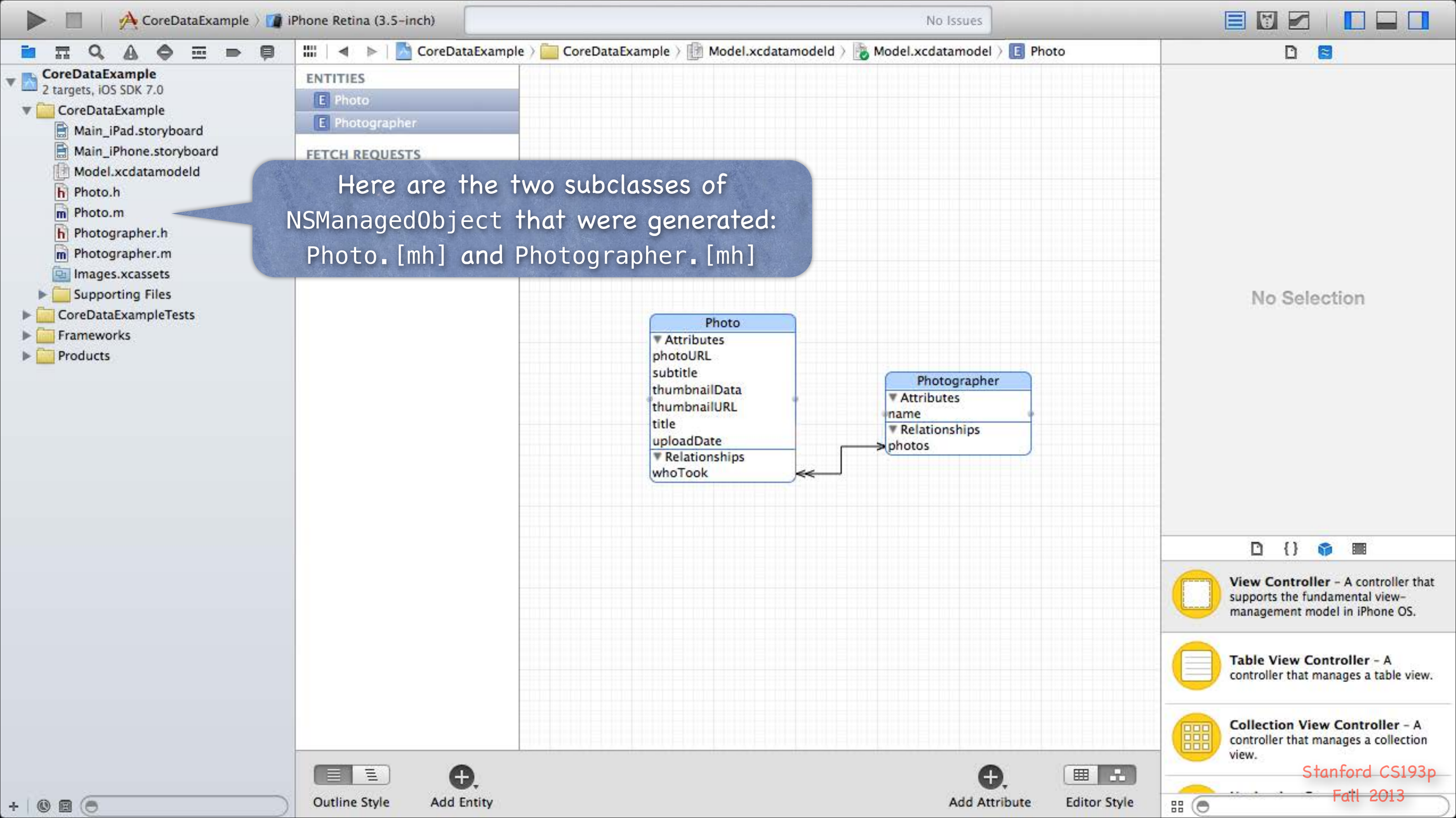

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
Photo
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATIONS
Default
Select both Entities.
We're going to have Xcode
generate NSManagedObject
subclasses for them for us.
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
whoTook
Photographer
Attributes
name
Relationships
photos
Entity
Name
Multiple Values
Class
NSManagedObject
Abstract Entity
Parent Entity
No Parent Entity
Indexes
User Info
Key
Value
Versioning
Hash Modifier
Version Hash Modifier
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

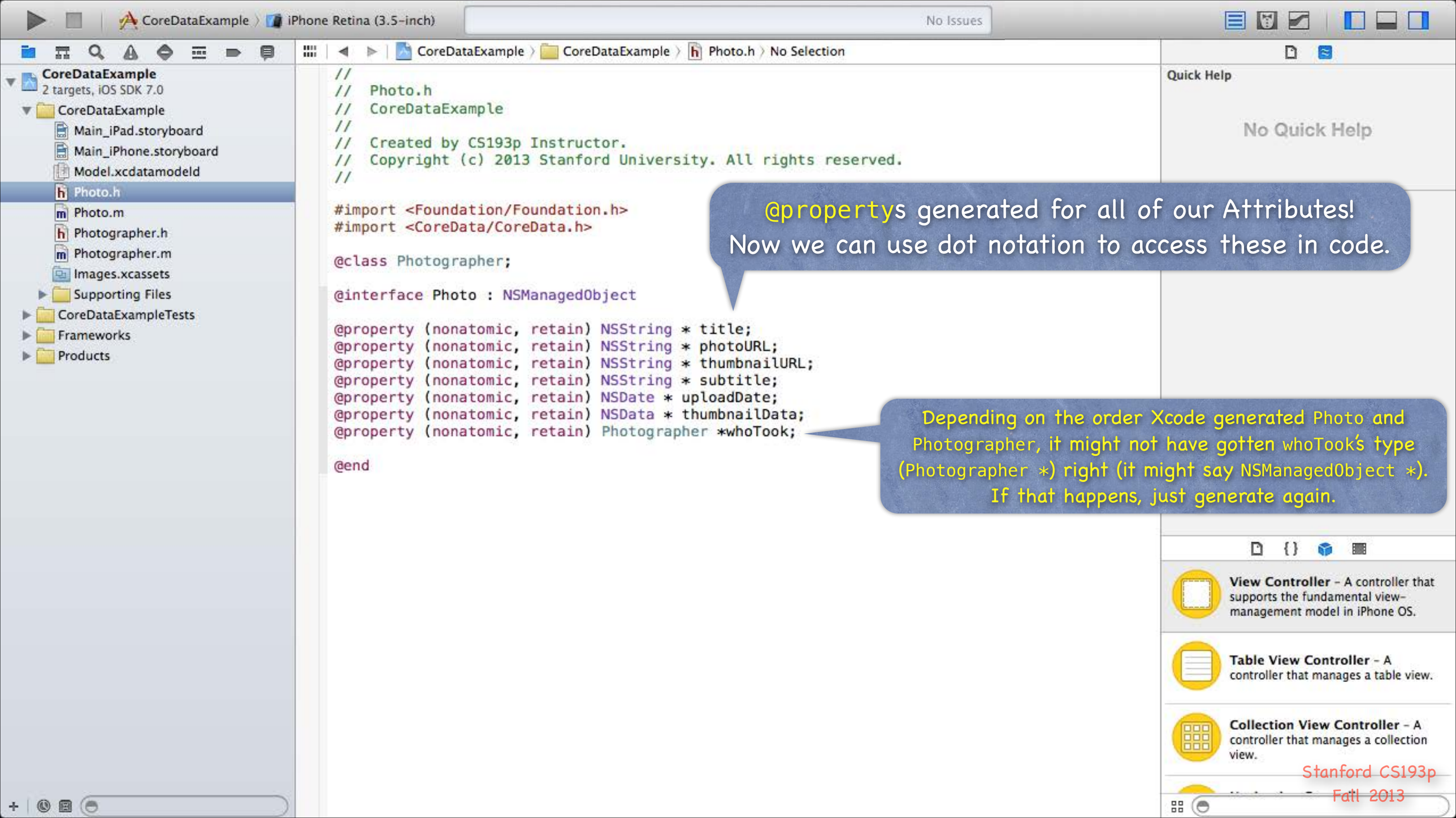

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Canvas
Add Entity
Add Fetch Request
Add Configuration
Add Attribute
Add Fetched Property
Add Relationship
Create NSManagedObject Subclass...
Add Model Version...
Import...
Ask Xcode to generate NSManagedObject subclasses for our Entities.
CoreDataExample
2 targets, iOS SDK 7.0
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
ENTITIES
Photo
Photographer
FETCH REQUESTS
CONFIGURATION
Default
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
whoTook
Photographer
Attributes
name
Relationships
photos
Entity
Name Multiple Values
Class NSManagedObject
Abstract Entity
Entity No Parent Entity
Key Value
Versioning
Hash Modifier Version Hash Modifier
Renaming ID Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

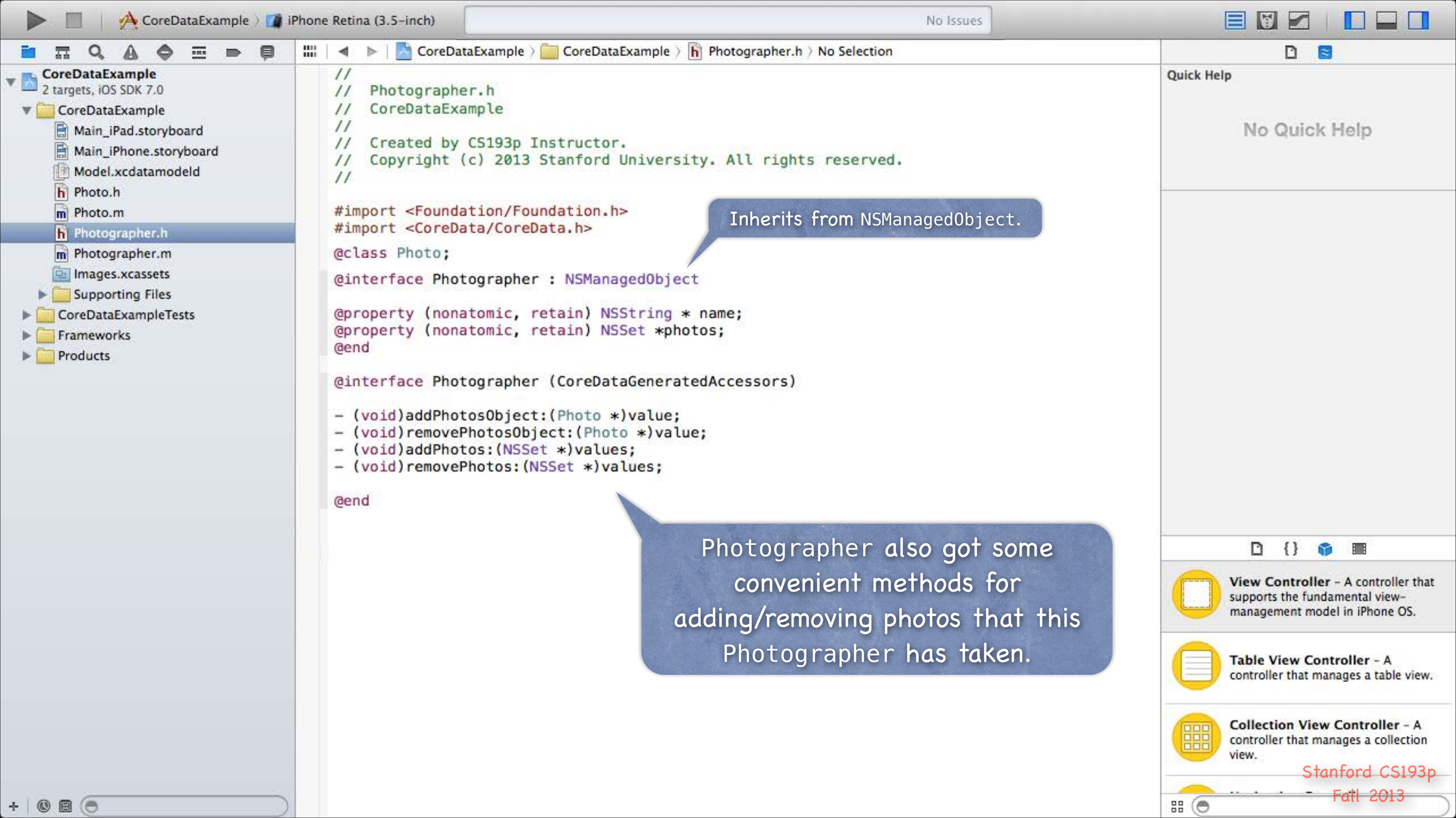

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
Select the data models with entities you would like to manage
Select
Data Model
Model
Which Data Models to generate subclasses for (we only have one Data Model).
Cancel
Previous
Next
Entity
Name
Multiple Values
Class
NSManagedObject
Abstract Entity
Parent Entity
No Parent Entity
Indexes
User Info
Key
Value
Versioning
Hash Modifier
Version Hash Modifier
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013

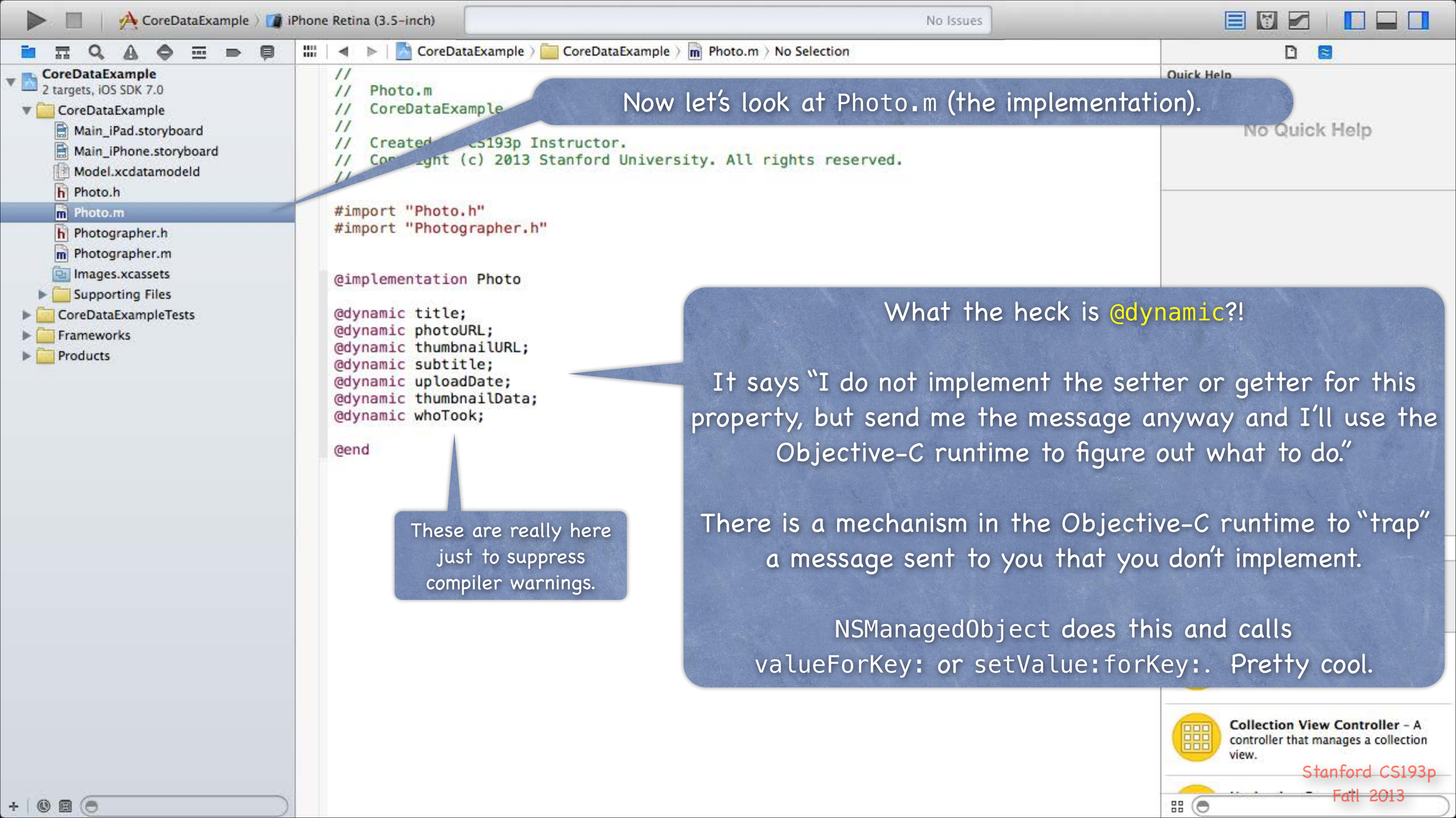

CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
Select the entities you would like to manage
Select
Entity
Photo
Photographer
Which Entities to generate subclasses for (usually we choose all of them).
Cancel
Previous
Next
Entity
Name
Multiple Values
Class
NSManagedObject
Abstract Entity
Parent Entity
No Parent Entity
Indexes
User Info
Key
Value
Versioning
Hash Modifier
Version Hash Modifier
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
FAVORITES
All My Files
CS193p
Documents
Applications
Downloads
Stanford
CoreDataExample
Example
Matchismo
CoreDataExample
CoreDataExample.xcodeproj
CoreDataExampleTests
AppDelegate.h
AppDelegate.m
Base.lproj
CoreDataExample-Info.plist
CoreDataEx...le-Prefix.pch
en.lproj
Images.xcassets
main.m
Model.xcdatamodeld
Name
Multiple Values
Class
NSManagedObject
Abstract Entity
Entity
No Parent Entity
Indexes
Pick which group you want your new classes to be stored (default is often one directory level higher, so watch out).
This will make your @propertys be scalars (e.g. int instead of NSNumber *) where possible. Be careful if one of your Attributes is an NSDate, you'll end up with an NSTimeInterval @property.
Options
Use scalar properties for primitive data types
Group
CoreDataExample
Targets
CoreDataExample
CoreDataExampleTests
New Folder
Cancel
Create
Modifier
Version Hash Modifier
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
Here are the two subclasses of NSManagedObject that were generated: Photo.[mh] and Photographer.[mh]
ENTITIES
Photo
Photographer
FETCH REQUESTS
CoreDataExample
CoreDataExample
Model.xcdatamodeld
Model.xcdatamodel
Photo
Photo
Attributes
photoURL
subtitle
thumbnailData
thumbnailURL
title
uploadDate
Relationships
whoTook
Photographer
Attributes
name
Relationships
photos
No Selection
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
CoreDataExample
Photo.h
No Selection
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
//
// Photo.h
// CoreDataExample
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
@class Photographer;
@interface Photo : NSManagedObject
@property (nonatomic, retain) NSString * title;
@property (nonatomic, retain) NSString * photoURL;
@property (nonatomic, retain) NSString * thumbnailURL;
@property (nonatomic, retain) NSString * subtitle;
@property (nonatomic, retain) NSDate * uploadDate;
@property (nonatomic, retain) NSData * thumbnailData;
@property (nonatomic, retain) Photographer *whoTook;
@end
@propertys generated for all of our Attributes!
Now we can use dot notation to access these in code.
Depending on the order Xcode generated Photo and Photographer, it might not have gotten whoTook's type (Photographer *) right (it might say NSManagedObject *). If that happens, just generate again.
Quick Help
No Quick Help
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
CoreDataExample
Photographer.h
No Selection
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
//
// Photographer.h
// CoreDataExample
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>
@class Photo;
@interface Photographer : NSManagedObject
@property (nonatomic, retain) NSString * name;
@property (nonatomic, retain) NSSet *photos;
@end
@interface Photographer (CoreDataGeneratedAccessors)
- (void)addPhotosObject:(Photo *)value;
- (void)removePhotosObject:(Photo *)value;
- (void)addPhotos:(NSSet *)values;
- (void)removePhotos:(NSSet *)values;
@end
Inherits from NSManagedObject.
Photographer also got some convenient methods for adding/removing photos that this Photographer has taken.
Quick Help
No Quick Help
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Stanford CS193p
Fall 2013


CoreDataExample 〉 iPhone Retina (3.5-inch)
No Issues
CoreDataExample 〉 CoreDataExample 〉 Photo.m 〉 No Selection
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
Now let's look at Photo.m (the implementation).
//
// Photo.m
// CoreDataExample
//
// Created by CS193p Instructor.
// Copyright (c) 2013 Stanford University. All rights reserved.
//
#import "Photo.h"
#import "Photographer.h"
@implementation Photo
@dynamic title;
@dynamic photoURL;
@dynamic thumbnailURL;
@dynamic subtitle;
@dynamic uploadDate;
@dynamic thumbnailData;
@dynamic whoTook;
@end
These are really here just to suppress compiler warnings.
What the heck is @dynamic?!
It says "I do not implement the setter or getter for this property, but send me the message anyway and I'll use the Objective-C runtime to figure out what to do."
There is a mechanism in the Objective-C runtime to "trap" a message sent to you that you don't implement.
NSManagedObject does this and calls valueForKey: or setValue:forKey:. Pretty cool.
Quick Help
No Quick Help
Collection View Controller – A controller that manages a collection view.
Stanford CS193p
Fall 2013

Core Data

- So how do I access my Entities' Attributes with dot notation?

```
// let's create an instance of the Photo Entity in the database …
NSManagedObjectContext *context = document.managedObjectContext;
Photo *photo = [NSEntityDescription insertNewObjectForEntityForName:@"Photo"
                                             inManagedObjectContext:context];

// then set the attributes in our Photo using, say, an NSDictionary we got from Flickr …
e.g. photo.title = [flickrData objectForKey:FLICKR_PHOTO_TITLE];
// the information will automatically be saved (i.e. autosaved) into our document by Core Data

// now here's some other things we could do too …
NSString *myThumbnail = photo.thumbnailURL;
photo.lastViewedDate = [NSDate date];
photo.whoTook = …; // a Photographer object we created or got by querying
photo.whoTook.name = @"CS193p Instructor"; // yes, multiple dots will follow relationships!
```

Core Data

- What if I want to add code to my `NSManagedObject` subclass?

  For example, we might want to add a method or two (to the `@property`s added by Xcode).

  It would be especially nice to add class methods to create and set up an object in the database
  (e.g. set all the properties of a `Photo` or `Photographer` using an `NSDictionary` from Flickr).
  Or maybe to derive new `@property`s based on ones in the database
  (e.g. a `UIImage` based on a URL in the database).

  But that could be a problem if we edited `Photo.m` or `Photographer.m` …
  Because you might want to modify your schema and re-generate those `.h` and `.m` files from Xcode!

  To get around this, we need to use an Objective-C feature called "categories".
  So let's take a moment to learn about that …

Categories

- Categories are an Objective-C syntax for adding to a class …
  - Without subclassing it.
  - Without even having to have access to the code of the class (e.g. you don't need its `.m`).
- Examples
  - `NSAttributedString`'s `drawAtPoint:` method.
    - Added by UIKit (since it's a UI method) even though `NSAttributedString` is in Foundation.
  - `NSIndexPath`'s `row` and `section` properties (used in `UITableView`-related code).
    - Added by UIKit too, even though `NSIndexPath` is also in Foundation.
- Syntax

```
@interface Photo (AddOn)
- (UIImage *)image;
@property (readonly) BOOL isOld;
@end
```

  - Categories have their own `.h` and `.m` files (usually `ClassName+PurposeOfExtension.[mh]`).
  - Categories cannot have instance variables!

Categories

- Implementation

```
@implementation Photo (AddOn)
- (UIImage *)image  // image is not an attribute in the database, but photoURL is
{
    NSURL *imageURL = [NSURL URLWithString:self.photoURL];
    NSData *imageData = [NSData dataWithContentsOfURL:imageURL];
    return [UIImage imageWithData:imageData];
}
- (BOOL)isOld  // whether this Photo was uploaded more than a day ago
{
    return [self.uploadDate timeIntervalSinceNow] > -24*60*60;
}
@end
```

Other examples … sometimes we add `@property`s to an `NSManagedObject` subclass via categories to make accessing `BOOL` attributes (which are `NSNumber`s) more cleanly.

Or we add `@property`s to convert `NSData`s to whatever the bits represent.

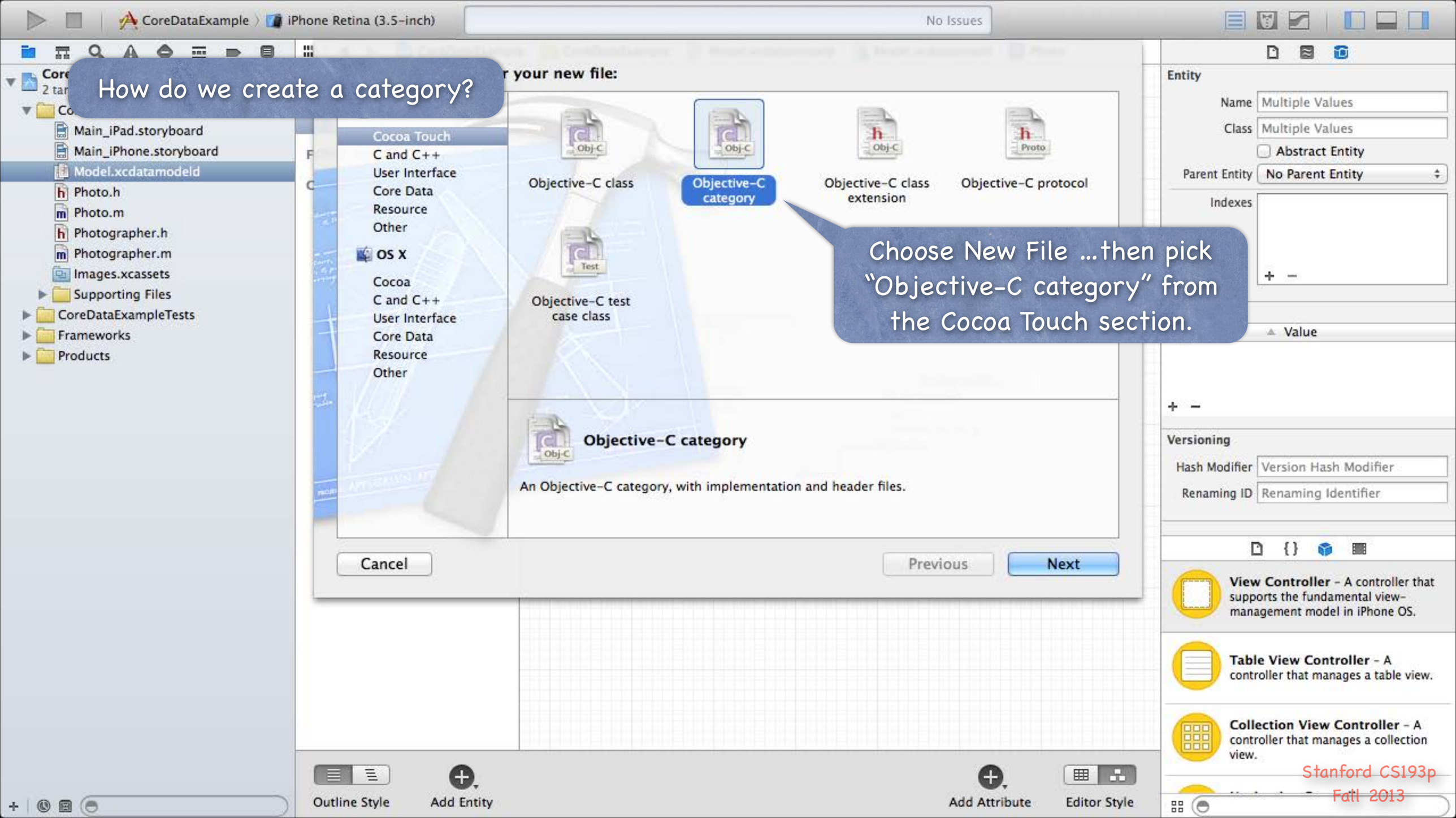
Any class can have a category added to it, but don't overuse/abuse this mechanism.

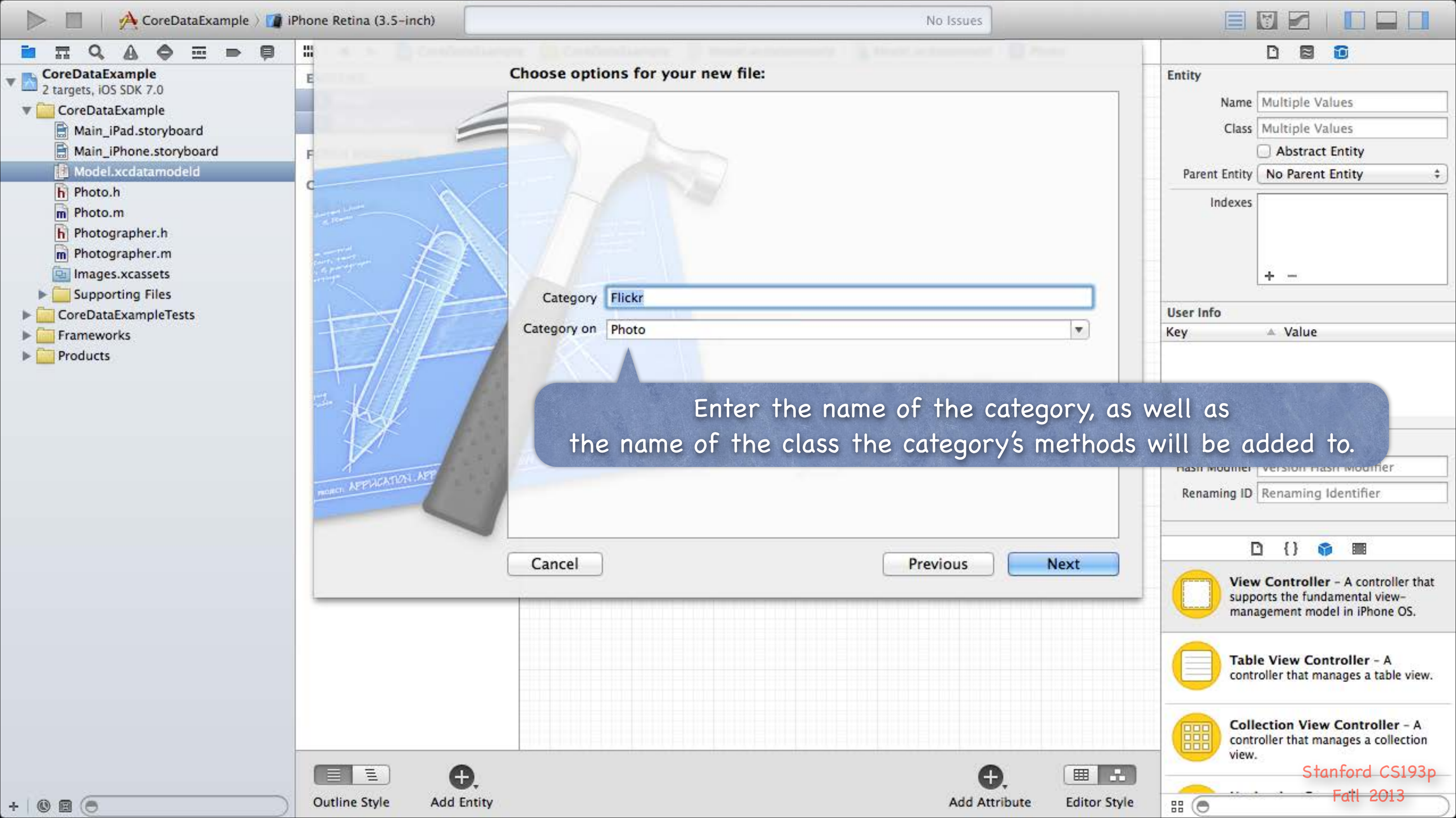
Categories

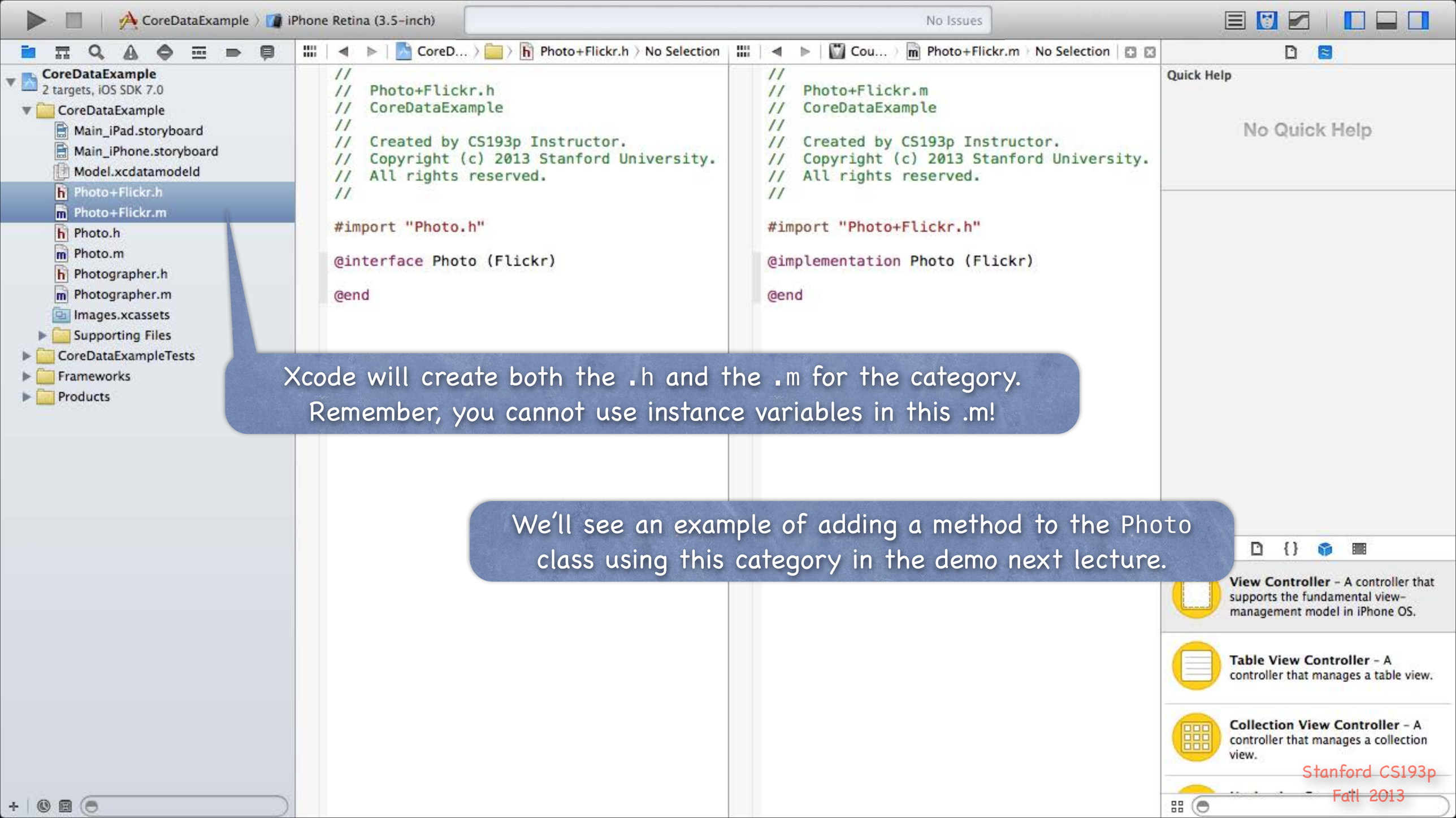
- Most common category on an `NSManagedObject` subclass?

Creation …

```
@implementation Photo (Create)
+ (Photo *)photoWithFlickrData:(NSDictionary *)flickrData
        inManagedObjectContext:(NSManagedObjectContext *)context
{
    Photo *photo = ...; // see if a Photo for that Flickr data is already in the database
    if (!photo) {
        photo = [NSEntityDescription insertNewObjectForEntityForName:@“Photo”
                                              inManagedObjectContext:context];
        // initialize the photo from the Flickr data
        // perhaps even create other database objects (like the Photographer)
    }
    return photo;
}
@end
```


How do we create a category?
CoreDataExample
iPhone Retina (3.5-inch)
No Issues
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
your new file:
Cocoa Touch
C and C++
User Interface
Core Data
Resource
Other
OS X
Cocoa
C and C++
User Interface
Core Data
Resource
Other
Objective-C class
Objective-C category
Objective-C class extension
Objective-C protocol
Objective-C test case class
Objective-C category
An Objective-C category, with implementation and header files.
Cancel
Previous
Next
Choose New File ...then pick "Objective-C category" from the Cocoa Touch section.
Entity
Name
Multiple Values
Class
Multiple Values
Abstract Entity
Parent Entity
No Parent Entity
Indexes
Value
Versioning
Hash Modifier
Version Hash Modifier
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
Choose options for your new file:
Category
Flickr
Category on
Photo
Cancel
Previous
Next
Enter the name of the category, as well as
the name of the class the category's methods will be added to.
Entity
Name
Multiple Values
Class
Multiple Values
Abstract Entity
Parent Entity
No Parent Entity
Indexes
User Info
Key
Value
Renaming ID
Renaming Identifier
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Outline Style
Add Entity
Add Attribute
Editor Style
Stanford CS193p
Fall 2013


CoreDataExample
iPhone Retina (3.5-inch)
No Issues
CoreDataExample
2 targets, iOS SDK 7.0
CoreDataExample
Main_iPad.storyboard
Main_iPhone.storyboard
Model.xcdatamodeld
Photo+Flickr.h
Photo+Flickr.m
Photo.h
Photo.m
Photographer.h
Photographer.m
Images.xcassets
Supporting Files
CoreDataExampleTests
Frameworks
Products
Photo+Flickr.h
No Selection
// Photo+Flickr.h // CoreDataExample // // Created by CS193p Instructor. // Copyright (c) 2013 Stanford University. // All rights reserved. //
#import "Photo.h"
@interface Photo (Flickr)
@end
Photo+Flickr.m
No Selection
// Photo+Flickr.m // CoreDataExample // // Created by CS193p Instructor. // Copyright (c) 2013 Stanford University. // All rights reserved. //
#import "Photo+Flickr.h"
@implementation Photo (Flickr)
@end
Quick Help
No Quick Help
Xcode will create both the .h and the .m for the category.
Remember, you cannot use instance variables in this .m!
We'll see an example of adding a method to the Photo class using this category in the demo next lecture.
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Collection View Controller – A controller that manages a collection view.
Stanford CS193p
Fall 2013

Deletion

- Deletion

  Deleting objects from the database is easy (sometimes too easy!)

```
[aDocument.managedObjectContext deleteObject:photo];
```

  Make sure that the rest of your objects in the database are in a sensible state after this.

  Relationships will be updated for you (if you set Delete Rule for relationship attributes properly).

  And don't keep any `strong` pointers to `photo` after you delete it!

- `prepareForDeletion`

  This is another method we sometimes put in a category of an `NSManagedObject` subclass …

```
@implementation Photo (Deletion)
- (void)prepareForDeletion
{
    // we don't need to set our whoTook to nil or anything here (that will happen automatically)
    // but if Photographer had, for example, a "number of photos taken" attribute,
    //    we might adjust it down by one here (e.g. self.whoTook.photoCount--).
}
@end
```

Querying

- So far you can …
  - Create objects in the database with `insertNewObjectForEntityForName:inManagedObjectContext:`.
  - Get/set properties with `valueForKey:`/`setValue:forKey:` or `@property`s in a custom subclass.
  - Delete objects using the `NSManagedObjectContext` `deleteObject:` method.
- One very important thing left to know how to do: QUERY
  - Basically you need to be able to retrieve objects from the database, not just create new ones
  - You do this by executing an `NSFetchRequest` in your `NSManagedObjectContext`
- Four important things involved in creating an `NSFetchRequest`
  1. `Entity` to fetch (required)
  2. How many objects to fetch at a time and/or maximum to fetch (optional, default: all)
  3. `NSSortDescriptor`s to specify the order in which the array of fetched objects are returned
  4. `NSPredicate` specifying which of those Entities to fetch (optional, default is all of them)

Querying

- Creating an `NSFetchRequest`

  We'll consider each of these lines of code one by one …

```
NSFetchRequest *request = [NSFetchRequest fetchRequestWithEntityName:@"Photo"];
request.fetchBatchSize = 20;
request.fetchLimit = 100;
request.sortDescriptors = @[sortDescriptor];
request.predicate = ...;
```

- Specifying the kind of Entity we want to fetch

  A given fetch returns objects all of the same Entity.

  You can't have a fetch that returns some `Photos` and some `Photographers` (it's one or the other).

- Setting fetch sizes/limits

  If you created a fetch that would match 1000 objects, the request above faults 20 at a time.

  And it would stop fetching after it had fetched 100 of the 1000.

Querying

- NSSortDescriptor

  When we execute a fetch request, it's going to return an NSArray of NSManagedObjects.
  NSArrays are "ordered," so we should specify the order when we fetch.

  We do that by giving the fetch request a list of "sort descriptors" that describe what to sort by.

```
NSSortDescriptor *sortDescriptor =
    [NSSortDescriptor sortDescriptorWithKey:@"title"
                                  ascending:YES
                                   selector:@selector(localizedStandardCompare:)];
```

  The selector: argument is just a method (conceptually) sent to each object to compare it to others.
  Some of these "methods" might be smart (i.e. they can happen on the database side).
  localizedStandardCompare: is for ordering strings like the Finder on the Mac does (very common).

  We give an array of these NSSortDescriptors to the NSFetchRequest because sometimes
  we want to sort first by one key (e.g. last name), then, within that sort, by another (e.g. first name).
  Examples: @[sortDescriptor] or @[lastNameSortDescriptor, firstNameSortDescriptor]

Querying

- NSPredicate

  This is the guts of how we specify exactly which objects we want from the database.

- Predicate formats

  Creating one looks a lot like creating an NSString, but the contents have semantic meaning.

```
NSString *serverName = @“flickr-5”;
NSPredicate *predicate =
    [NSPredicate predicateWithFormat:@“thumbnailURL contains %@”, serverName];
```

- Examples

```
@“uniqueId = %@”, [flickrInfo objectForKey:@“id”] // unique a photo in the database
@“name contains[c] %@”, (NSString *) // matches name case insensitively
@“viewed > %@”, (NSDate *) // viewed is a Date attribute in the data mapping
@“whoTook.name = %@”, (NSString *) // Photo search (by photographer’s name)
@“any photos.title contains %@”, (NSString *) // Photographer search (not a Photo search)
```

  Many more options. Look at the class documentation for NSPredicate.

Querying

- NSCompoundPredicate

  You can use AND and OR inside a predicate string, e.g. `@"(name = %@) OR (title = %@)"`

  Or you can combine NSPredicate objects with special NSCompoundPredicates.

```
NSArray *array = @[predicate1, predicate2];
NSPredicate *predicate = [NSCompoundPredicate andPredicateWithSubpredicates:array];
```

  This predicate is "predicate1 AND predicate2". Or available too, of course.

Advanced Querying

- Key Value Coding

  Can actually do predicates like `@"photos.@count > 5"` (Photographers with more than 5 photos).

  `@count` is a function (there are others) executed in the database itself.

  https://developer.apple.com/library/ios/documentation/cocoa/conceptual/KeyValueCoding/Articles/CollectionOperators.html.

  By the way, all this stuff (and more) works on dictionaries, arrays and sets too …

  e.g. `[propertyListResults valueForKeyPath:@"photos.photo.@avg.latitude"]` on Flickr results

  returns the average latitude of all of the photos in the results (yes, really)

  e.g. `@"photos.photo.title.length"` would return an array of the lengths of the titles of the photos

- NSExpression

  Advanced topic. Can do sophisticated data gathering from the database.

  No time to cover it now, unfortunately.

  If interested, for both NSExpression and Key Value Coding queries, investigate …

```
NSFetchRequest *request = [NSFetchRequest fetchRequestWithEntityName:@"…"];
[request setResultType:NSDictionaryResultType]; // fetch returns array of dicts instead of NSMO's
[request setPropertiesToFetch:@[@"name", expression, etc.]];
```

Querying

- Putting it all together

  Let's say we want to query for all Photographers …

```
NSFetchRequest *request = [NSFetchRequest fetchRequestWithEntityName:@“Photographer”];
```

  … who have taken a photo in the last 24 hours …

```
NSDate *yesterday = [NSDate dateWithTimeIntervalSinceNow:-24*60*60];
request.predicate = [NSPredicate predicateWithFormat:@“any photos.uploadDate > %@”, yesterday];
```

  … sorted by the Photographer's name …

```
request.sortDescriptors = @[[NSSortDescriptor sortDescriptorWithKey:@“name” ascending:YES]];
```

Querying

- Executing the fetch

```
NSManagedObjectContext *context = aDocument.managedObjectContext;
NSError *error;
NSArray *photographers = [context executeFetchRequest:request error:&error];
```

Returns `nil` if there is an error (check the `NSError` for details).
Returns an empty array (not `nil`) if there are no matches in the database.
Returns an `NSArray` of `NSManagedObjects` (or subclasses thereof) if there were any matches.
You can pass `NULL` for `error:` if you don't care why it fails.

That's it. Very simple really.

Query Results

- Faulting

  The above fetch does not necessarily fetch any actual data.

  It could be an array of "as yet unfaulted" objects, waiting for you to access their attributes.

  Core Data is very smart about "faulting" the data in as it is actually accessed.

  For example, if you did something like this …

```
for (Photographer *photographer in photographers) {
    NSLog(@"fetched photographer %@", photographer);
}
```

  You may or may not see the names of the photographers in the output

  (you might just see "unfaulted object", depending on whether it prefetched them)

  But if you did this …

```
for (Photographer *photographer in photographers) {
    NSLog(@"fetched photographer named %@", photographer.name);
}
```

  … then you would definitely fault all the Photographers in from the database.

  That's because in the second case, you actually access the `NSManagedObject`'s data.

Core Data Thread Safety

- NSManagedObjectContext is not thread safe
  - Luckily, Core Data access is usually very fast, so multithreading is only rarely needed.
  - Usually we create NSManagedObjectContext using a queue-based concurrency model.
  - This means that you can only touch a context and its NSMO's in the queue it was created on.
- Thread-Safe Access to an NSManagedObjectContext

```
[context performBlock:^{  // or performBlockAndWait:
    // do stuff with context in its safe queue (the queue it was created on)
}];
```

  - Note that the Q might well be the main Q, so you're not necessarily getting "multithreaded."
- Parent Context (advanced)
  - Some contexts (including UIManagedDocument ones) have a parentContext (a @property on NSMOC).
  - This parentContext will almost always be on a separate queue, but access the same database.
  - This means you can performBlock: on it to access the database off the main queue (e.g.).
  - But it is still a different context, so you'll have to refetch in the child context to see any changes.

Core Data

- There is so much more (that we don't have time to talk about)!
  - Optimistic locking (`deleteConflictsForObject:`)
  - Rolling back unsaved changes
  - Undo/Redo
  - Staleness (how long after a fetch until a refetch of an object is required?)

Coming Up

- Homework
  - Assignment 5 due Wednesday.
  - Final homework (Assignment 6) will be assigned Wednesday, due the next Wednesday.
- Wednesday
  - Final Project Requirements
  - Core Data and `UITableView`
  - Core Data Demo
- Next Week
  - Multitasking
  - Advanced Segueing
  - Map Kit?

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

- Final Project Requirements
- Core Data and `UITableView`
- Core Data Demo

  Photomania

Final Project

- Proposal due immediately!
  - And must be received no later than next Wednesday.
  - Send PDF of your proposal to your CA (the one who has graded your latest assignment).
  - Proposal must say not only what you are doing, but also what parts of SDK will be featured.
- Project (including Keynote) due on Friday, December 6th.
  - Use normal submission process (put the keynote file at the top level where README is).
  - NO LATE DAYS (last two assignments are the last opportunity to use free late days).
- Required presentation during final exam period
  - Thursday, December 12th at 12:15pm in this room.
  - 2-minute Keynote (not PowerPoint) presentation (more on this in a moment).
  - 1280x720 aspect ratio (not 1024x768 or 800x600).

  - Alternate presentation time on Thursday, December 5th (w/Keynote due by Tuesday, December 3rd).
  - If you need/want the alternate presentation time, let us know immediately (via class staff e-mail).

Final Project

- Scope is the same as about three weeks of homework
  - Luckily, you'll have about three weeks to do it (counts as approximately 35% of your overall grade).
  - P/NC students must pass both homework and final project segments separately.
- Must work on hardware!
  - Bring your hardware to final exam to demo to TA (if not used during your presentation).
  - iPad or iPhone or iPod Touch okay.
- Only iOS SDK code "counts"
  - Don't waste your time writing server-side code
  - Okay to "simulate" a server-side interaction to make your code demonstrable.

Final Project

- You'll be graded on proper use of SDK
  - Hackery will count against you. Use good object-oriented programming technique.
  - Must have at least one feature which was NOT taught in lecture/demo/homework assignment.
  - Breadth is VERY important. Don't get stuck down a rathole.
  - Only need to show depth in one or two areas. Breadth is more important.
- Aesthetics of your user-interface matter
  - (although we do not expect professional graphic designer quality graphics)
  - Sloppy layouts will be graded down.
  - Lots of places to get graphics from on the internet.
- Be careful not to get side-tracked on non-iOS-code
  - Some students in the past have spent 80% of their time working on stuff that didn't demonstrate their mastery of the class material.
  - (e.g. preparing some large database or working on graphics too much, etc.)
  - In the end, this is an iOS PROGRAMMING course, so we want to see how well you can program on this platform.

Final Project

- Presentation Quality Matters
  - A (tiny) portion of your grade will be related to the quality of your presentation.
  - Not okay to just put up a recording of you or of your application and say nothing.
  - Being able to make a live presentation is a valuable skill.
  - Practice your presentation before you show up.
  - You only get 2 minutes (strictly enforced), so make 'em count.
- Live demo?
  - All iOS 7 devices (iPad2+, iPhone4S, iPhone5) can mirror their screen to the projector here.
  - Live demos are perilous, as you saw all quarter :), but effective!
  - You must, at worst, show screen shots of your application.
  - Keynote/Quicktime has some tools to "animate" screen shots (better than static).
  - Video (screen capture) of your app in action can be good also.

Sample Proposal

- Section 1: What am I doing?

  I will be building a "Shakespeare Director" application.

  It will have the following features:

  - A table for choosing a Shakespearean play from a list downloaded from Folio*.
  - A custom view for laying out the blocking of a chosen Shakespearean play.
  - A dialogue-learning mode.

  * Folio is an on-line database of all of Shakespeare's works.

  The custom view will be simple (only rectangles and circles with colors for stroke/fill, and text).

  Photos (from Camera or Library) can be put in rectangles in the blocking view.

  The blocking can change from line to line in the dialog (but no more often than that).

  Blocking can be stepped through, line by line, or played back in "time lapse" mode.

  The dialogue-learning mode will step through all the dialog line by line.

  Users can record the dialog for other parts (as prompts for them to learn their own part).

  iPad only.

Sample Proposal

- Section 2: What parts of iOS will it use?
  - `UITableView` for choosing plays and stepping through dialog
  - Custom `UITableViewCell` prototypes (for dialog, including speaker, blocking instructions)
  - Custom `UIView` with `drawRect:` for scene-setting
  - Camera/Photo Library for putting images in blocking rectangles
  - `UITextField` in a `UIPopoverController` for text labels in the scene-setting view
  - `UIPopoverController` for choosing stroke and fill color and shape in scene-setting mode
  - Scroll view to zoom in/pan around in blocking view
  - `AVFoundation` for record/playback of dialog
  - `NSTimer` for "time lapse playback" of entire play with dialog/blocking linked
  - Core Data to store the scene-setting and dialog
    - Play entity
    - Scene entity
    - BlockingElement entity
    - LineOfDialog entity
  - Printing of blocking to AirPrint printers (this is the NOT COVERED IN LECTURE feature)

Sample Proposal

- What to notice about this sample proposal?
  - Clear description of what the application will do (section 1).
  - Clear list of the iOS features that will be used (section 2).
  - Lots of breadth (not necessarily that much depth in any one area).
  - Clearly delineates the NOT COVERED IN LECTURE feature.
  - Specifies platform (iPad only sacrifices breadth, but makes sense for this project).
  - It's creative (it's not just Matchismo or Top Places recycled).

Core Data and UITableView

- How to hook these up
  - As you can imagine, they were (probably literally) made for each other!
  - The magic to doing this? `NSFetchedResultsController` …

Core Data and UITableView

- NSFetchedResultsController

  Simply hooks an NSFetchRequest up to a UITableViewController

  Usually you'll have an NSFetchedResultsController @property in your UITableViewController.

  It will be hooked up to an NSFetchRequest that returns the data you want to show in your table.

  Then use it to answer all your UITableViewDataSource protocol's questions!

- For example ...

```
- (NSUInteger)numberOfSectionsInTableView:(UITableView *)sender
{
    return [[self.fetchedResultsController sections] count];
}

- (NSUInteger)tableView:(UITableView *)sender numberOfRowsInSection:(NSUInteger)section
{
    return [[[self.fetchedResultsController sections] objectAtIndex:section] numberOfObjects];
}
```

NSFetchedResultsController

- Very important method … objectAtIndexPath:

NSFetchedResultsController method …

```
- (NSManagedObject *)objectAtIndexPath:(NSIndexPath *)indexPath;
```

Here's how you would use it in, for example, tableView:cellForRowAtIndexPath: …

```
- (UITableViewCell *)tableView:(UITableView *)sender
         cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = ...;
    NSManagedObject *managedObject =  // or, e.g., Photo *photo = (Photo *) …
        [self.fetchedResultsController objectAtIndexPath:indexPath];
   // load up the cell based on the properties of the managedObject
   // of course, if you had a custom subclass, you'd be using dot notation to get them
   return cell;
}
```

NSFetchedResultsController

- How do you create an NSFetchedResultsController?

  Just need the NSFetchRequest to drive it (and a NSManagedObjectContext to fetch from).
  Let's say we want to show all photos taken by someone with the name photogName in our table:

```
NSFetchRequest *request = [NSFetchRequest fetchRequestWithEntityName:@“Photo”];
request.sortDescriptors = @[[NSSortDescriptor sortDescriptorWithKey:@“title” ...]];
request.predicate = [NSPredicate predicateWithFormat:@“whoTook.name = %@”, photogName];

NSFetchedResultsController *frc = [[NSFetchedResultsController alloc]
    initWithFetchRequest:(NSFetchRequest *)request
    managedObjectContext:(NSManagedObjectContext *)context
      sectionNameKeyPath:(NSString *)keyThatSaysWhichSectionEachManagedObjectIsIn
               cacheName:@“MyPhotoCache”];  // careful!
```

  Be sure that any cacheName you use is always associated with exactly the same request.
  It's okay to specify nil for the cacheName (no cacheing of fetch results in that case).

  It is critical that the sortDescriptor matches up with the keyThatSaysWhichSection...
  The results must sort such that all objects in the first section come first, second second, etc.

NSFetchedResultsController

- NSFRC also "watches" changes in Core Data and auto-updates table

  Uses a key-value observing mechanism.

  When it notices a change, it sends message like this to its `delegate` …

```
- (void)controller:(NSFetchedResultsController *)controller
   didChangeObject:(id)anObject
       atIndexPath:(NSIndexPath *)indexPath
     forChangeType:(NSFetchedResultsChangeType)type
      newIndexPath:(NSIndexPath *)newIndexPath
{
   // here you are supposed call appropriate UITableView methods to update rows
   // but don't worry, we're going to make it easy on you …
}
```

CoreDataTableViewController

- NSFetchedResultsController's doc shows how to do all this
  - In fact, you're supposed to copy/paste the code from the doc into your table view subclass.
  - But that's all a bit of a pain, so …
- Enter CoreDataTableViewController!
  - We've copy/pasted the code from NSFetchedResultsController into a subclass of UITVC for you!
- How does CoreDataTableViewController work?
  - It's just a UITableViewController that adds an NSFetchedResultsController as a @property.
  - Whenever you set it, it will immediately start using it to fill the contents of its UITableView.
- Easy to use
  - Download it along with your homework assignment.
  - Just subclass it and override the methods that load up cells and/or react to rows being selected (you'll use the NSFetchedResultsController method objectAtIndexPath: mentioned earlier).
  - Then just set the fetchedResultsController @property and watch it go!

Demo

- Photomania
  - Gets recent photos from Flickr.
  - Shows a list of photographers who took all the photos.
  - Select a photographer -> shows a list of all the photos that photographer took.
  - Core Data Entities: Photographer and Photo.
- Watch for ...
  - How we define our database schema graphically in Xcode.
  - How we create `NSManagedObject` subclasses and then add categories to them.
  - Especially how we use categories to create "factory" methods to create/initialize database objects.
  - The Application Delegate (finally!)
  - `NSManagedObjectContext`
  - Background Fetching
  - Background URL Sessions
  - `NSNotification` posting and listening
  - How we use `CoreDataTableViewController` to hook the table views up to the database.

Coming Up

- Homework
  - Last one!
  - Due next Wednesday.
- Friday
  - Instruments (performance monitoring in Xcode).
  - (This is actually at risk. Watch Piazza for whether it's going to come together.)
- Next Week
  - More Multitasking
  - Advanced Segueing
  - Map Kit?

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

- UIApplication
  - What's that?
- Network Activity Indicator
  - An application wide activity spinner for network activity only
- Demo Followup
  - A couple of things to note about last week's demo
- Demo
  - More Photomania (iPad version with popover)
- Maps
  - Showing whether things are on earth
  - We'll get as far as we can, then continue on Wednesday (along with a demo)

UIApplication

- UIApplication

  There is a shared instance of a `UIApplication` object in your application.

  This is different from your Application Delegate (the thing that handles all those message from iOS).

  You almost never need it, but it can give you some interesting (very global) information.

  ```
  UIApplication *myApplication = [UIApplication sharedApplication];
  ```

  Check out its documentation.

Network Activity Indicator

- Network Activity Indicator
  - This property in UIApplication is interesting …
  - `@property (nonatomic, getter=is…) networkActivityIndicatorVisible;`
  - When this is set to YES, a little spinner will appear in the status bar. NO means turn it off.
  - This spinner is ONLY for network activity (but you should spin it for ALL network activity you do).
- It can be somewhat difficult to use this property correctly
  - Because it is global and is a boolean.
  - What if you have multiple, overlapping threads using the network at the same time?
  - You are required to layer mechanism for that on top of this property yourself.

Demo Followup

- We forgot to set our minimum background fetch interval

```
[[UIApplication sharedApplication] setMinimumBackgroundFetchInterval:(NSTimeInterval)interval];
```

  The default is `UIApplicationBackgroundFetchIntervalNever`, so set it or you get none!
  Minimum you can set it to is `UIApplicationBackgroundFetchIntervalMinimum` (often want this).
  Usually you would set this in `application:didFinishLaunchingWithOptions:`.

  Also, the user can turn off your application's ability to run in the background entirely!

```
@property UIBackgroundRefreshStatus backgroundRefreshStatus;
```

- Fetching when given the opportunity

  When we are given the opportunity to fetch in the background, we should do a normal fetch.
  In other words, do a normal, ephemeral URL session fetch, not a background session URL fetch.
  Background session URL fetches are discretionary (meaning iOS can refuse if in background).
  The posted code from last week does this.
  Doing a normal fetch also makes it easier to call the completion handler with the `NewData` option!

Demo

- More Photomania!

  Flesh out Photomania on iPad & add the table of photos by the photographer and an image VC.
  Then we'll add a popover to show the URL of the photo we're looking at.

Core Location

- Framework for managing location and heading

  No user-interface.

- Basic object is `CLLocation`

  `@propertys: coordinate, altitude, horizontal/verticalAccuracy, timestamp, speed, course`

- Where (approximately) is this location?

```
@property (readonly) CLLocationCoordinate2D coordinate;
typedef {
    CLLocationDegrees latitude;    // a double
    CLLocationDegrees longitude;   // a double
} CLLocationCoordinate2D;

@property (readonly) CLLocationDistance altitude;  // meters
```

A negative value means "below sea level."

Core Location

- How close to that latitude/longitude is the actual location?

```
@property (readonly) CLLocationAccuracy horizontalAccuracy;   // in meters
@property (readonly) CLLocationAccuracy verticalAccuracy;     // in meters
```

A negative value means the coordinate or altitude (respectively) is invalid.

```
kCLLocationAccuracyBestForNavigation  // phone should be plugged in to power source
kCLLocationAccuracyBest
kCLLocationAccuracyNearestTenMeters
kCLLocationAccuracyHundredMeters
kCLLocationAccuracyKilometer
kCLLocationAccuracyThreeKilometers
```

- The more accuracy you request, the more battery will be used

Device "does its best" given a specified accuracy request
Cellular tower triangulation (not very accurate, but low power)
WiFi node database lookup (more accurate, more power)
GPS (very accurate, lots of power)

Core Location

- Speed

```
@property (readonly) CLLocationSpeed speed;          // in meters/second
```

Note that the speed is instantaneous (not average speed).
Generally it's useful as "advisory information" when you are in a vehicle.
A negative value means "speed is invalid."

- Course

```
@property (readonly) CLLocationDirection course;     // in degrees, 0 is north, clockwise
```

Not all devices can deliver this information.
A negative value means "course is invalid."

- Time stamp

```
@property (readonly) NSDate *timestamp;
```

Pay attention to these since locations will be delivered on an inconsistent time basis.

- Distance between CLLocations

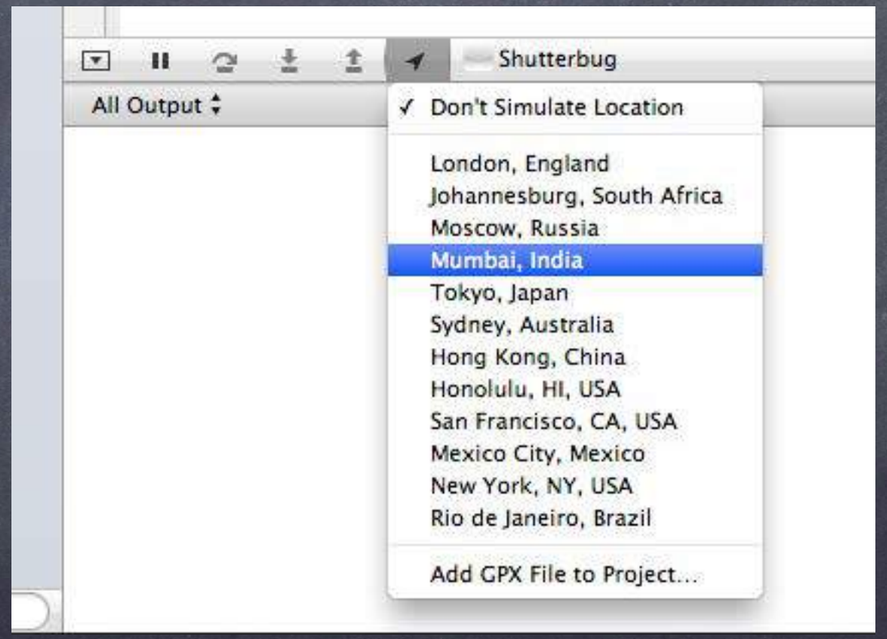
```
- (CLLocationDistance)distanceFromLocation:(CLLocation *)otherLocation; // in meters
```

Core Location

- How do you get a `CLLocation`?

  Almost always from a `CLLocationManager` (sent to you via its `delegate`).

  Can be tested in the simulator from Xcode.



Core Location

- How do you get a `CLLocation`?

  Almost always from a `CLLocationManager` (sent to you via its `delegate`).

  Can be tested in the simulator from Xcode.

- `CLLocationManager`

  General approach to using it:

  1. Check to see if the hardware you are on/user supports the kind of location updating you want.
  2. Create a `CLLocationManager` instance and set the delegate to receive updates.
  3. Configure the manager according to what kind of location updating you want.
  4. Start the manager monitoring for location changes.

Core Location

- Kinds of location monitoring
  - Accuracy-based continual updates.
  - Updates only when "significant" changes in location occur.
  - Region-based updates.
  - Heading monitoring.

Core Location

- Checking to see what your hardware can do

```
+ (CLAuthorizationStatus)authorizationStatus; // Authorized, Denied or Restricted (parental, enterprise)
+ (BOOL)locationServicesEnabled; // user has enabled (or not) location services for your application
+ (BOOL)significantLocationChangeMonitoringAvailable;
+ (BOOL)isMonitoringAvailableForClass:(Class)regionClass;  // [CLBeacon/CLCircularRegion class]
+ (BOOL)isRangingAvailable;   // device can tell how far it is from beacons
```

  Other tests for other location capabilities too.

- Getting the information from the `CLLocationManager`

  You can just ask (poll) the `CLLocationManager` for the location or heading, but usually we don't.
  Instead, we let it update us when the location changes (enough) via its `delegate` …

Core Location

- Error reporting to the delegate

```
- (void)locationManager:(CLLocationManager *)manager
       didFailWithError:(NSError *)error;
```

Not always a fatal thing, so pay attention to this delegate method. Some examples …

```
kCLErrorLocationUnknown   // likely temporary, keep waiting (for a while at least)
kCLErrorDenied            // user refused to allow your application to receive updates
kCLErrorHeadingFailure    // too much local magnetic interference, keep waiting
```

Core Location

- Accuracy-based continuous location monitoring

```
@property CLLocationAccuracy desiredAccuracy;  // always set this as low as will work for you
```

  You can also limit updates to only occurring if the change in location exceeds a certain distance …

```
@property CLLocationDistance distanceFilter;
```

- Starting and stopping normal position monitoring

```
- (void)startUpdatingLocation;
- (void)stopUpdatingLocation;
```

  Be sure to turn updating off when your application is not going to consume the changes!

- Get notified via the CLLocationManager's delegate

```
- (void)locationManager:(CLLocationManager *)manager
    didUpdateLocations:(NSArray *)locations; // of CLLocation
```

- Similar API for heading (CLHeading, et. al.)

Core Location

- Background

  It is possible to receive these kinds of updates in the background.
  Apps that do this have to be very careful (because these updates can be power hungry).
  There are very cool ways to, for example, coalesce and defer location update reporting.
  Have to enable backgrounding (in the same area of your project settings as background fetch).

  But there are 2 ways to get location notifications (on a coarser scale) without doing that ...

Core Location

- Significant location change monitoring in CLLocationManager
  - "Significant" is not strictly defined. Think vehicles, not walking. Likely uses cell towers.
  - `- (void)startMonitoringSignificantLocationChanges;`
  - `- (void)stopMonitoringSignificantLocationChanges;`
  - Be sure to turn updating off when your application is not going to consume the changes!
- Get notified via the CLLocationManager's delegate
  - Same as for accuracy-based updating if your application is running.
- And this works even if your application is not running!
  - (Or is in the background.)
  - You will get launched and your Application Delegate will receive the message `application:didFinishLaunchingWithOptions:` with an options dictionary that will contain `UIApplicationLaunchOptionsLocationKey`
  - Create a `CLLocationManager` (if you don't have one), then get the latest location via `@property (readonly) CLLocation *location;`
  - If you are running in the background, don't take too long (a few seconds)!

Core Location

- Region-based location monitoring in CLLocationManager
  - `- (void)startMonitoringForRegion:(CLRegion *)region; // CLCircularRegion/CLBeaconRegion`
  - `- (void)stopMonitoringForRegion:(CLRegion *)region;`
  - Alloc and initWithCenter:radius:identifier: a CLCircularRegion to monitor an area.
  - Beacons are for detecting when you are near another device. New in iOS 7.
- Get notified via the CLLocationManager's delegate
  - `- (void)locationManager:(CLLocationManager *)manager didEnterRegion:(CLRegion *)region;`
  - `- (void)locationManager:(CLLocationManager *)manager didExitRegion:(CLRegion *)region;`
  - `- (void)locationManager:(CLLocationManager *)manager monitoringDidFailForRegion:(CLRegion *)region withError:(NSError *)error;`
- Works even if your application is not running!
  - In exactly the same way as "significant location change" monitoring.
  - The set of monitored regions persists across application termination/launch.
  - `@property (readonly) NSSet *monitoredRegions;` // property on CLLocationManager

Core Location

- CLRegions are tracked by name
  Because they survive application termination/relaunch.
- Circular region monitoring size limit
  `@property (readonly) CLLocationDistance maximumRegionMonitoringDistance;`
  Attempting to monitor a region larger than this (radius in meters) will generate an error
  (which will be sent via the `delegate` method mentioned on previous slide).
  If this property returns a negative value, then region monitoring is not working.
- Beacon regions can also detect range from a beacon
  `- (void)startRangingBeaconsInRegion:(CLBeaconRegion *)beaconRegion;`
  Delegate method `locationManager:didRangeBeacons:inRegion:` gives you `CLBeacon` objects.
  `CLBeacon` objects will tell you proximity (e.g. `CLProximityImmediate`/`Near`/`Far`).
- To be a beacon is a bit more involved
  Beacons are identified by a globally unique UUID (that you generate).
  Check out `CBPeripheralManager` (Core Bluetooth Framework).

Coming Up

- Homework
  - Due Friday
- Wednesday
  - MapKit
  - Photomania Map (and Embed Segue) Demo
- Friday
  - Core Image
- Next Week
  - Miscellaneous Topics

Stanford CS193p

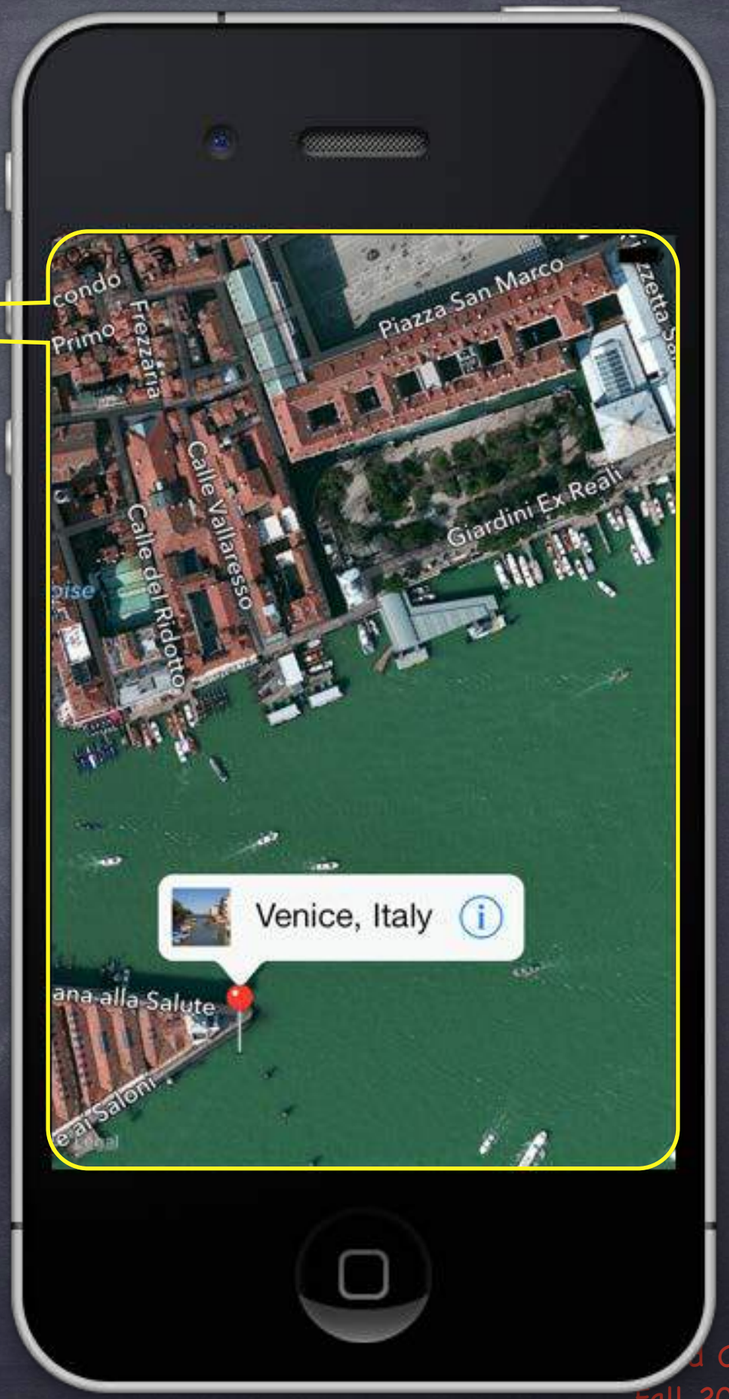
Developing Applications for iOS
Fall 2013-14

Today

- MapKit
  User interface for dealing with locations.
- Embed Segue
  Putting one VC's `self.view` inside another VC's View
- Photomania Map Demo
  Embedding a Map View Controller into our View Controller that displays a Photo

Map Kit

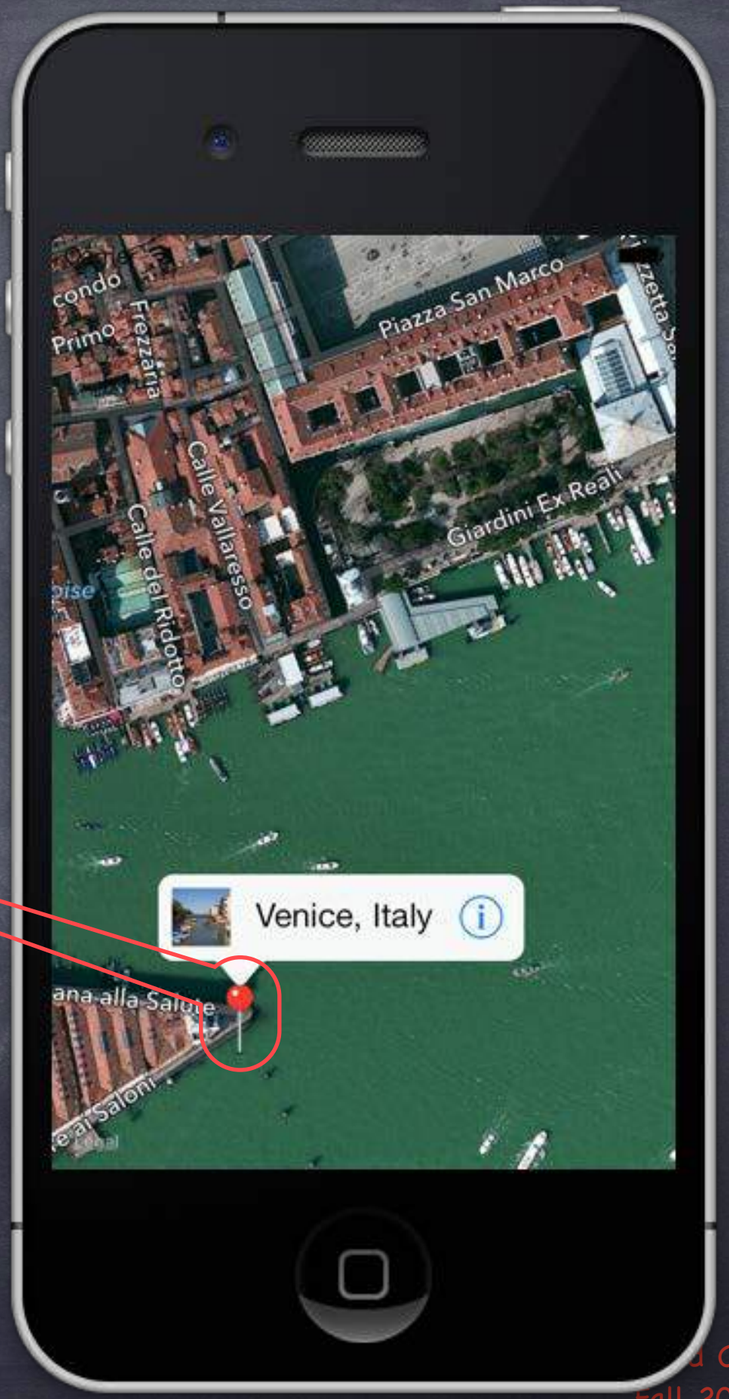
- `MKMapView` displays a map



Map Kit

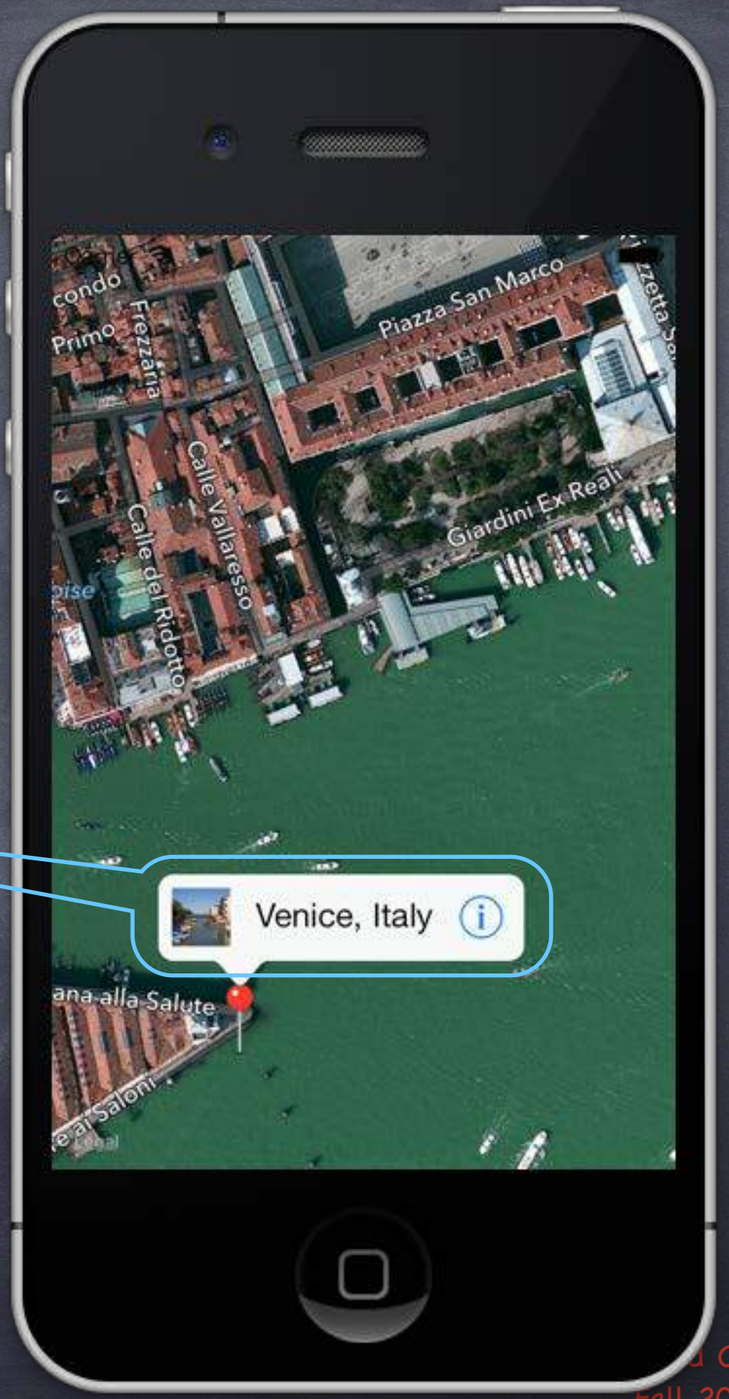
- MKMapView displays a map
- The map can have annotations on it

  Each annotation is simply a coordinate, a title and a subtitle. They are displayed using an MKAnnotationView (MKPinAnnotationView shown here).



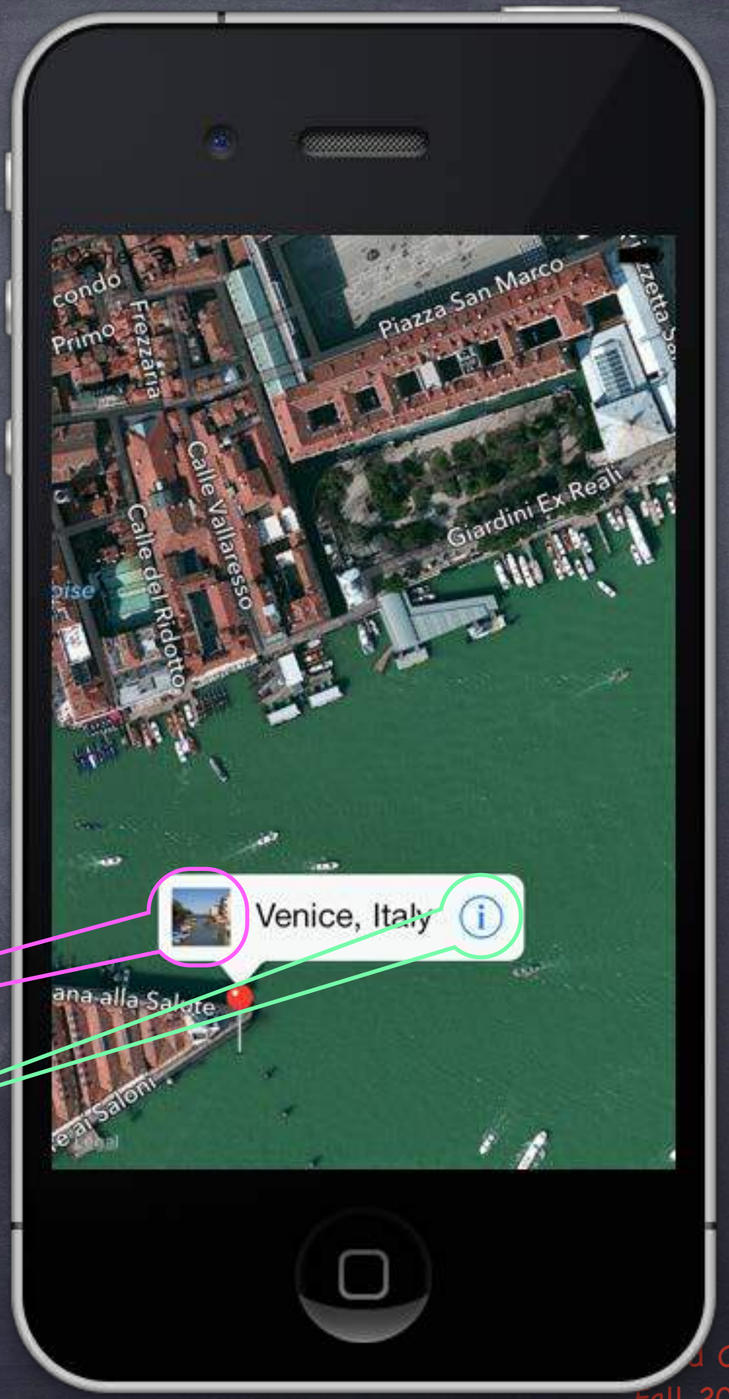
Map Kit

- MKMapView displays a map
- The map can have annotations on it
  Each annotation is simply a coordinate, a title and a subtitle.
  They are displayed using an MKAnnotationView
  (MKPinAnnotationView shown here).
- Annotations can have a callout
  It appears when the annotation view is clicked.
  By default just shows the title and subtitle.



Map Kit

- MKMapView displays a map
- The map can have annotations on it
  - Each annotation is simply a coordinate, a title and a subtitle.
  - They are displayed using an MKAnnotationView (MKPinAnnotationView shown here).
- Annotations can have a callout
  - It appears when the annotation view is clicked.
  - By default just shows the title and subtitle.
- But callout can also have accessory views
  - In this example, the left is a UIImageView,
  - the right is a UIButton (UIButtonTypeDetailDisclosure)



MKMapView

- Create with alloc/init or drag from object palette in Xcode
- Displays an array of objects which implement MKAnnotation

```
@property (readonly) NSArray *annotations; // contains id <MKAnnotation> objects
```

- MKAnnotation protocol

```
@protocol MKAnnotation <NSObject>
@property (readonly) CLLocationCoordinate2D coordinate;
@optional
@property (readonly) NSString *title;
@property (readonly) NSString *subtitle;
@end

typedef {
    CLLocationDegrees latitude;
    CLLocationDegrees longitude;
} CLLocationCoordinate2D;
```

MKAnnotation

- Note that the annotations property is readonly, so …

```
@property (readonly) NSArray *annotations; // contains id <MKAnnotation> objects
```

Must add/remove annotations explicitly

```
- (void)addAnnotation:(id <MKAnnotation>)annotation;
- (void)addAnnotations:(NSArray *)annotations;
- (void)removeAnnotation:(id <MKAnnotation>)annotation;
- (void)removeAnnotations:(NSArray *)annotations;
```

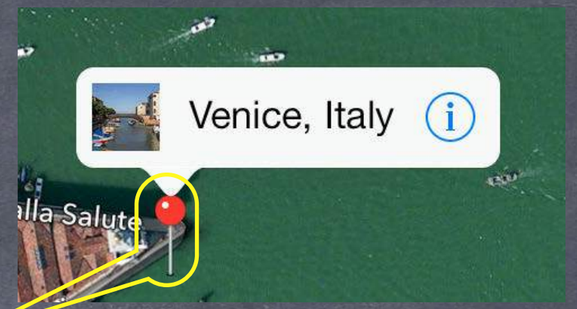
- Generally a good idea to add all your annotations up-front

Allows the MKMapView to be efficient about how it displays them

Annotations are light-weight, but annotation views are not.

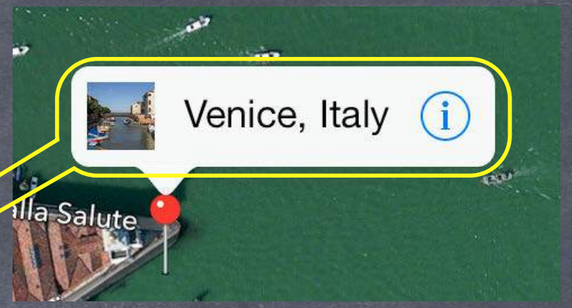
Luckily MKMapView reuses annotation views similar to how UITableView reuses cells.

MKAnnotation



- What do annotations look like on the map?
  - Annotations are drawn using an `MKAnnotationView` subclass.
  - The default one is `MKPinAnnotationView` (which is why they look like pins by default).
  - You can subclass or set properties on existing `MKAnnotationViews` to modify the look.

MKAnnotation



- What do annotations look like on the map?
  - Annotations are drawn using an `MKAnnotationView` subclass.
  - The default one is `MKPinAnnotationView` (which is why they look like pins by default).
  - You can subclass or set properties on existing `MKAnnotationViews` to modify the look.
- What happens when you touch on an annotation (e.g. the pin)?
  - Depends on the `MKAnnotationView` that is associated with the annotation (more on this later).
  - By default, nothing happens, but if `canShowCallout` is `YES` in the `MKAnnotationView`, then a little box will appear showing the annotation's title and subtitle.
  - And this little box (the callout) can be enhanced with `left`/`rightCalloutAccessoryViews`.

  The following `delegate` method is also called…

```
- (void)mapView:(MKMapView *)sender didSelectAnnotationView:(MKAnnotationView *)aView;
```

  This is a great place to set up the `MKAnnotationView`'s callout accessory views lazily.
  For example, you might want to wait until this method is called to download an image to show.

MKAnnotationView

- How are MKAnnotationViews created & associated w/annotations?

  Very similar to UITableViewCells in a UITableView.

  Implement the following MKMapViewDelegate method (if not implemented, returns a pin view).

```
- (MKAnnotationView *)mapView:(MKMapView *)sender
            viewForAnnotation:(id <MKAnnotation>)annotation
{
    MKAnnotationView *aView = [sender dequeueReusableAnnotationViewWithIdentifier:IDENT];
    if (!aView) {
        aView = [[MKPinAnnotationView alloc] initWithAnnotation:annotation
                                                 reuseIdentifier:IDENT];
        // set canShowCallout to YES and build aView's callout accessory views here
    }
    aView.annotation = annotation; // yes, this happens twice if no dequeue
    // maybe load up accessory views here (if not too expensive)?
    // or reset them and wait until mapView:didSelectAnnotationView: to load actual data
    return aView;
}
```

  You can see why you might want to only show visible annotations (to keep view count low)

MKAnnotationView

- MKAnnotationView

  Interesting properties (all nonatomic, strong if a pointer) …

```
@property id <MKAnnotation> annotation;  // the annotation; treat as if readonly
@property UIImage *image;  // instead of the pin, for example
@property UIView *leftCalloutAccessoryView;  // maybe a UIImageView
@property UIView *rightCalloutAccessoryView;  // maybe a "disclosure" UIButton
@property BOOL enabled;  // NO means it ignores touch events, no delegate method, no callout
@property CGPoint centerOffset;  // where the "head of the pin" is relative to the image
@property BOOL draggable;  // only works if the annotation implements setCoordinate:
```

- If you set one of the callout accessory views to a UIControl

  e.g. `aView.rightCalloutAccessoryView = [UIButton buttonWithType:UIButtonTypeDetailDisclosure];`

  The following MKMapViewDelegate method will get called when the accessory view is touched …

```
- (void)mapView:(MKMapView *)sender
 annotationView:(MKAnnotationView *)aView
calloutAccessoryControlTapped:(UIControl *)control;
```

MKAnnotationView

- Using `didSelectAnnotationView:` to load up callout accessories

  Example … downloaded thumbnail image in `leftCalloutAccessoryView`.

  Create the `UIImageView` and assign it to `leftCalloutAccessoryView` in `mapView:viewForAnnotation:`.

  Reset the `UIImageView`'s image to `nil` there as well.

  Then load the image on demand in `mapView:didSelectAnnotationView:` …

```
- (void)mapView:(MKMapView *)sender didSelectAnnotationView:(MKAnnotationView *)aView
{
    if ([aView.leftCalloutAccessoryView isKindOfClass:[UIImageView class]]) {
        UIImageView *imageView = (UIImageView *)aView.leftCalloutAccessoryView;
        imageView.image = ...;  // if you do this in a GCD queue, be careful, views are reused!
    }
}
```

MKMapView

- Configuring the map view's display type

```
@property MKMapType mapType;
MKMapTypeStandard, MKMapTypeSatellite, MKMapTypeHybrid;
```

- Showing the user's current location

```
@property BOOL showsUserLocation;
@property (readonly) BOOL isUserLocationVisible;
@property (readonly) MKUserLocation *userLocation;
```

MKUserLocation is an object which conforms to MKAnnotation which holds the user's location.

- Restricting the user's interaction with the map

```
@property BOOL zoomEnabled;
@property BOOL scrollEnabled;
@property BOOL pitchEnabled;   // 3D
@property BOOL rotateEnabled;
```

MKMapCamera

- Setting where the user is seeing the map from (in 3D)

```
MKMapView @property (copy) MKMapCamera *camera;
```

- MKMapCamera

  Specify `centerCoordinate`, `heading`, `pitch` and `altitude` of the camera.

  Or use convenient initializer …

```
+ (MKMapCamera *)cameraLookingAtCenterCoordinate:(CLLocationCoordinate2D)coord
                                   fromEyeCoordinate:(CLLocationCoordinate2D)cameraPosition
                                         eyeAltitude:(CLLocationDistance)eyeAltitude;
```

MKMapView

- Controlling the region (part of the world) the map is displaying

```
@property MKCoordinateRegion region;
typedef struct {
    CLLocationCoordinate2D center;
    MKCoordinateSpan span;
} MKCoordinateRegion;
typedef struct {
    CLLocationDegrees latitudeDelta;
    CLLocationDegrees longitudeDelta;
}
- (void)setRegion:(MKCoordinateRegion)region animated:(BOOL)animated; // animate
```

- Can also set the center point only or set to show annotations

```
@property CLLocationCoordinate2D centerCoordinate;
- (void)setCenterCoordinate:(CLLocationCoordinate2D)center animated:(BOOL)animated;
- (void)showAnnotations:(NSArray *)someAnnotations animated:(BOOL)animated;
```

MKMapView

- Lots of C functions to convert points, regions, rects, etc.

  See documentation, e.g. `MKMapRectContainsPoint`, `MKMapPointForCoordinate`, etc.

- Converting to/from map points/rects from/to view coordinates

  ```
  - (MKMapPoint)mapPointForPoint:(CGPoint)point;
  - (MKMapRect)mapRectForRect:(CGRect)rect;
  - (CGPoint)pointForMapPoint:(MKMapPoint)mapPoint;
  - (CGRect)rectForMapRect:(MKMapRect)mapRect;
  ```

  Etc.

MKMapView

- Another MKMapViewDelegate method …

```
- (void)mapView:(MKMapView *)mapView didChangeRegionAnimated:(BOOL)animated;
```

This is a good place to "chain" animations to the map.

When you display somewhere new in the map that is far away, zoom out, then back in.

This method will let you know when it's finished zooming out, so you can then zoom in.

MKLocalSearch

- Searching for places in the world

  Can search by "natural language" strings asynchronously (uses the network) …

```
MKLocalSearchRequest *request = [[MKLocalSearchRequest alloc] init];
request.naturalLanguageQuery = @"Ike's";
request.region = …; // e.g., Stanford campus
MKLocalSearch *search = [[MKLocalSearch alloc] initWithRequest:request];
[search startWithCompletionHandler:^(MKLocalSearchResponse *response, NSError *error) {
    // response contains an array of MKMapItem which contains MKPlacemark
}];
```

- MKMapItem

  You can open one of these in the Maps app!

```
- (BOOL)openInMapsWithLaunchOptions:(NSDictionary *)options; // options like region, show traffic
```

- MKPlacemark

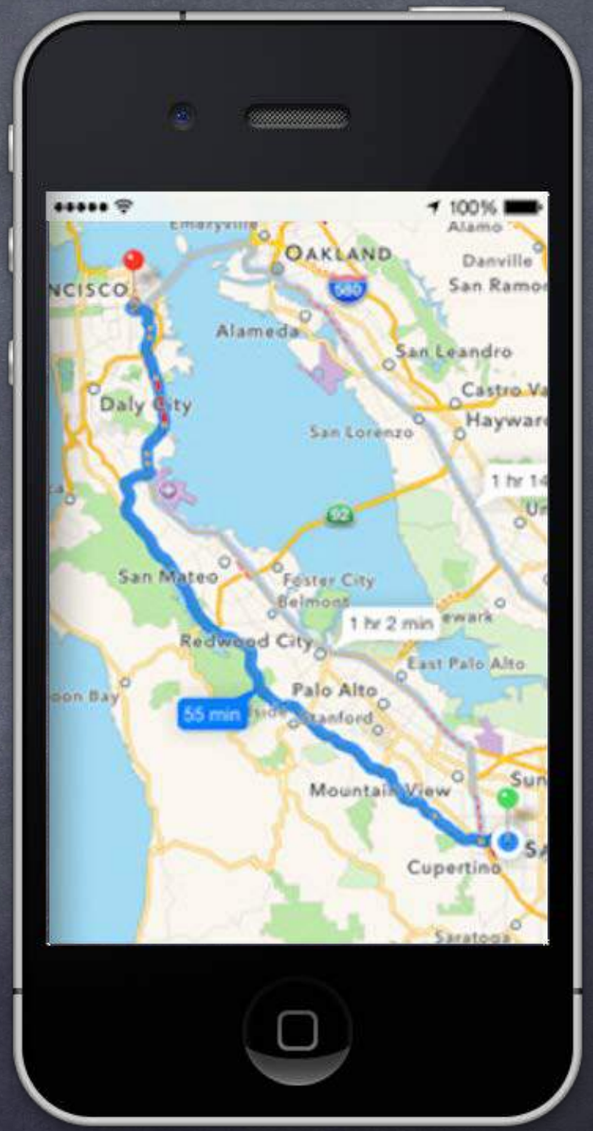
  Contains location, name of `location`, `postalCode`, `region`, etc.

MKDirections

- Getting directions from one place to another
  - Very similar API to searching.
  - Specify source and destination `MKMapItem`.
  - Asynchronous API to get a bunch of `MKRoute`s.

  `MKRoute` includes a name for the route, turn-by-turn directions, expected travel time, etc.

  Also come with `MKPolyline` descriptions of the routes which can be overlaid on the map …



Overlays

- Overlays

  Add overlays to the MKMapView and it will later ask you for a renderer to draw the overlay.

  ```
  - (void)addOverlay:(id <MKOverlay>)overlay level:(MKOverlayLevel)level;
  ```

  Level is (currently) either `AboveRoads` or `AboveLabels` (over everything but annotation views).

  ```
  - (void)removeOverlay:(id <MKOverlay>)overlay;
  ```

- MKOverlay protocol

  Protocol which includes MKAnnotation plus …

  ```
  @property (readonly) MKMapRect boundingMapRect;
  - (BOOL)intersectsMapRect:(MKMapRect)mapRect; // optional, uses boundingMapRect otherwise
  ```

- Overlays are associated with MKOverlayRenderers via delegate

  Just like annotations are associated with MKAnnotationViews, so are renderers with overlays …

  ```
  - (MKOverlayRenderer *)mapView:(MKMapView *)sender
                rendererForOverlay:(id <MKOverlay>)overlay;
  ```

MKOverlayView

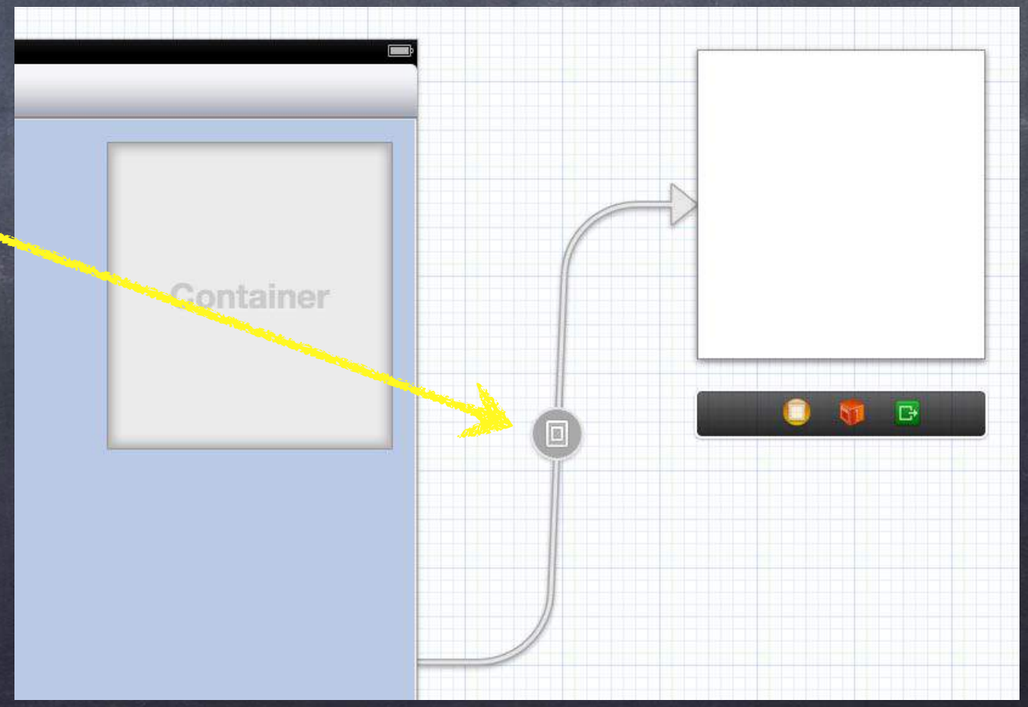
- Built-in Overlays and Renderers for numerous shapes …

  `MKCircleRenderer`
  `MKPolylineRenderer`
  `MKPolygonRenderer`
  `MKTileOverlayRenderer` // can also be used to replace the map data from Apple

  There's a whole set of `MKShape` and subclasses thereof for you to explore.

Embed Segues

- Putting a VC's `self.view` in another VC's view hierarchy!
  - This can be a very powerful encapsulation technique.
- Xcode makes this easy
  - Drag out a Container View from the object palette into the scene you want to embed it in.
  - Automatically sets up an "Embed Segue" from container VC to the contained VC.
- Embed Segue
  - Works just like other segues.
  - `prepareForSegue:sender:`, et. al.



Embed Segues

- Putting a VC's `self.view` in another VC's view hierarchy!
  - This can be a very powerful encapsulation technique.
- Xcode makes this easy
  - Drag out a Container View from the object palette into the scene you want to embed it in.
  - Automatically sets up an "Embed Segue" from container VC to the contained VC.
- Embed Segue
  - Works just like other segues.
  - `prepareForSegue:sender:`, et. al.
- View Loading Timing
  - Don't forget, though, that just like other segued-to VCs the embedded VC's outlets are not set at the time `prepareForSegue:sender:` is called.

Demo

- Photomania Maps
  - Instead of showing a table of photos, show a map of them.
  - Maps show `id <MKAnnotation>`s, so we'll turn a `Photo` object into an `MKAnnotation`!
  - Show thumbnails when users click on photo pins in the map.
  - Allow user to segue to a full view of the photo from the callout.
  - On iPad embed the map inside a `ImageViewController`.

Coming Up

- Homework
  Due Friday
- Friday
  Core Image
- Next Week
  Miscellaneous Topics

Stanford CS193p

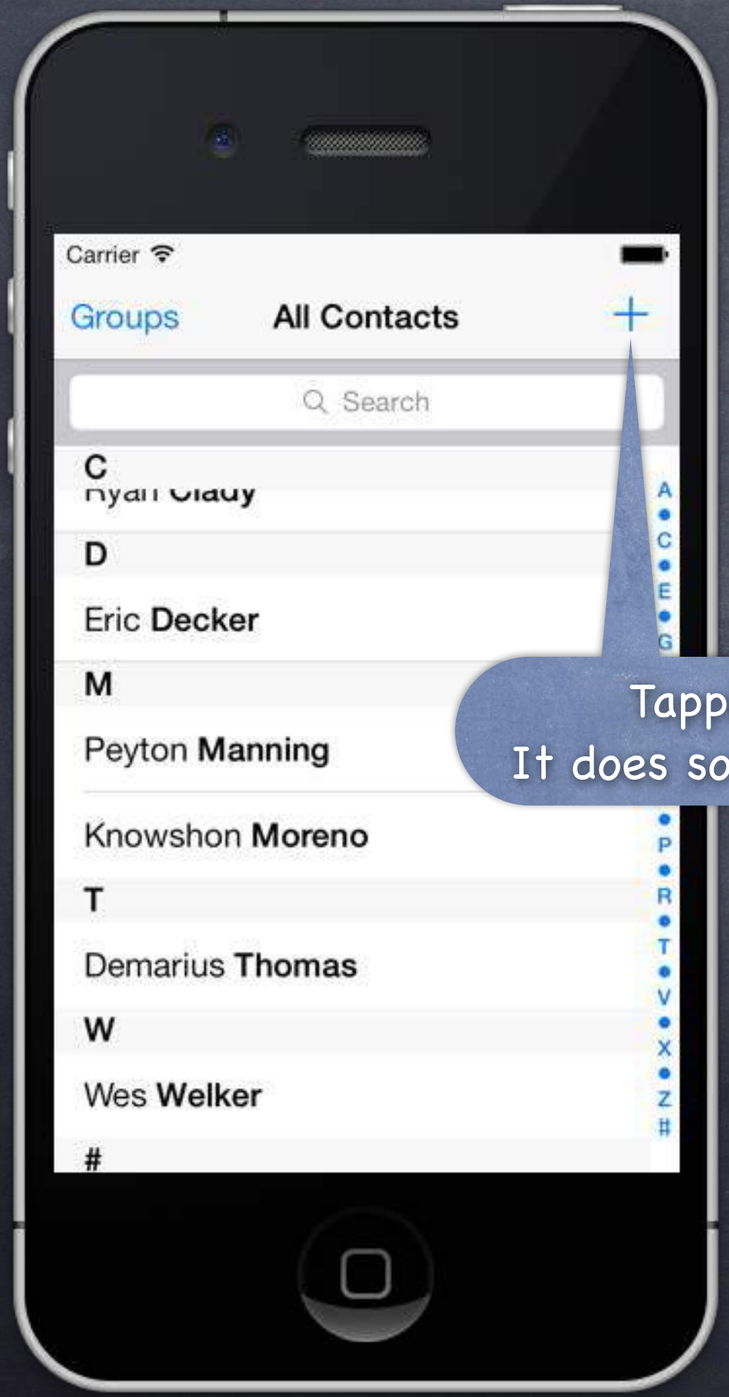
Developing Applications for iOS
Fall 2013-14

Today

- Modal Segues
  Transitioning to a Controller which "takes over your UI" until it's done with the user.
- Text Fields
  How to get text input from the user.
- Alerts and Action Sheets
  Notifying the user and getting "branching decisions" from the user.
- Demo
  Adding a photo taken by the user to Photomania.
- Camera (time permitting)
  And Photo Library.

Modal View Controllers

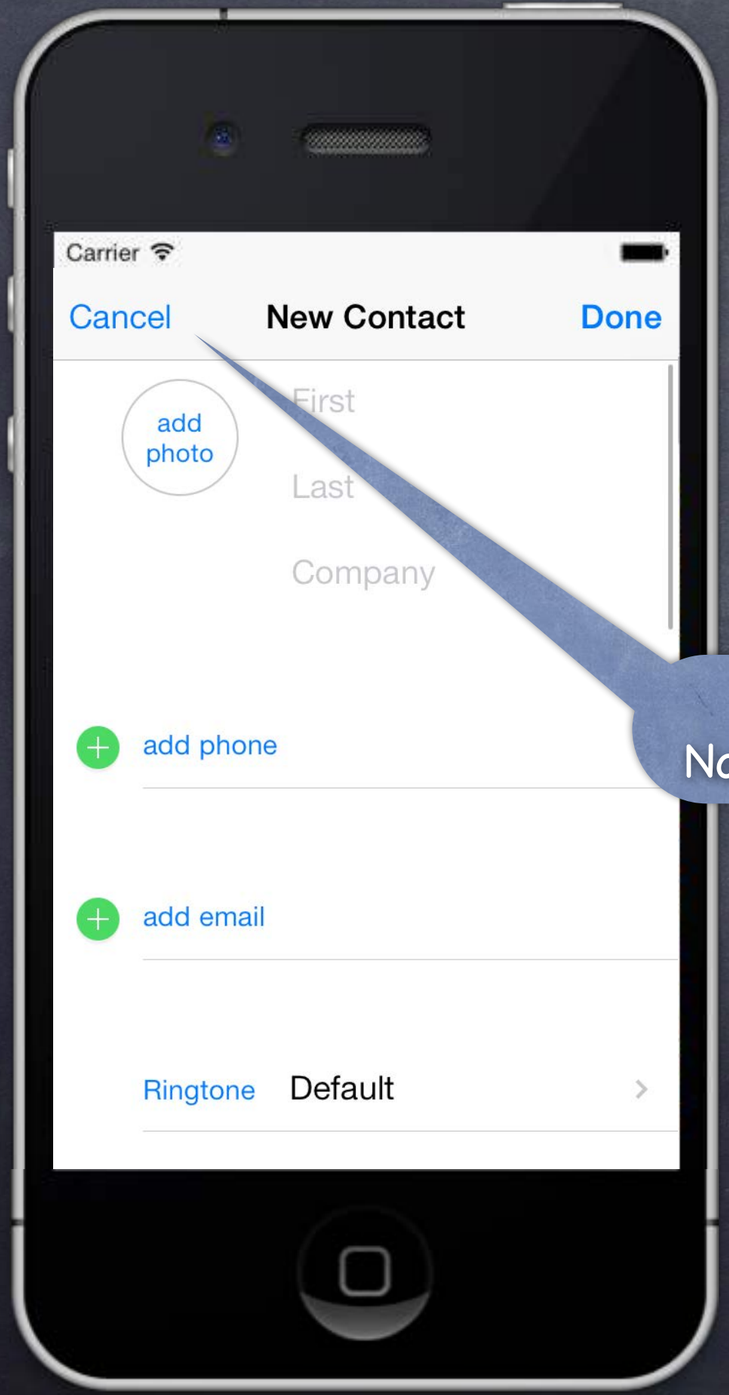
- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



Tapping here adds a new contact.
It does so by taking over the entire screen.

Modal View Controllers

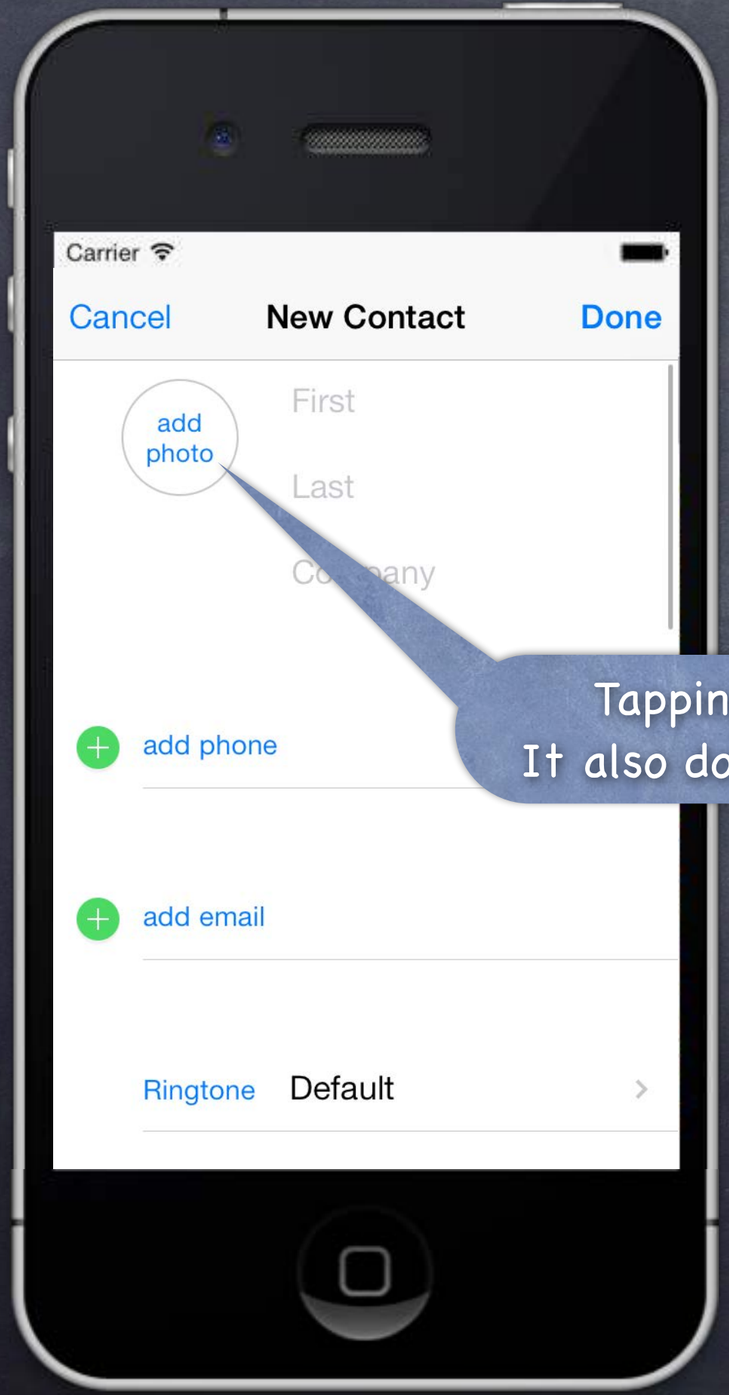
- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



This is not a push.
Notice, no back button (only Cancel).

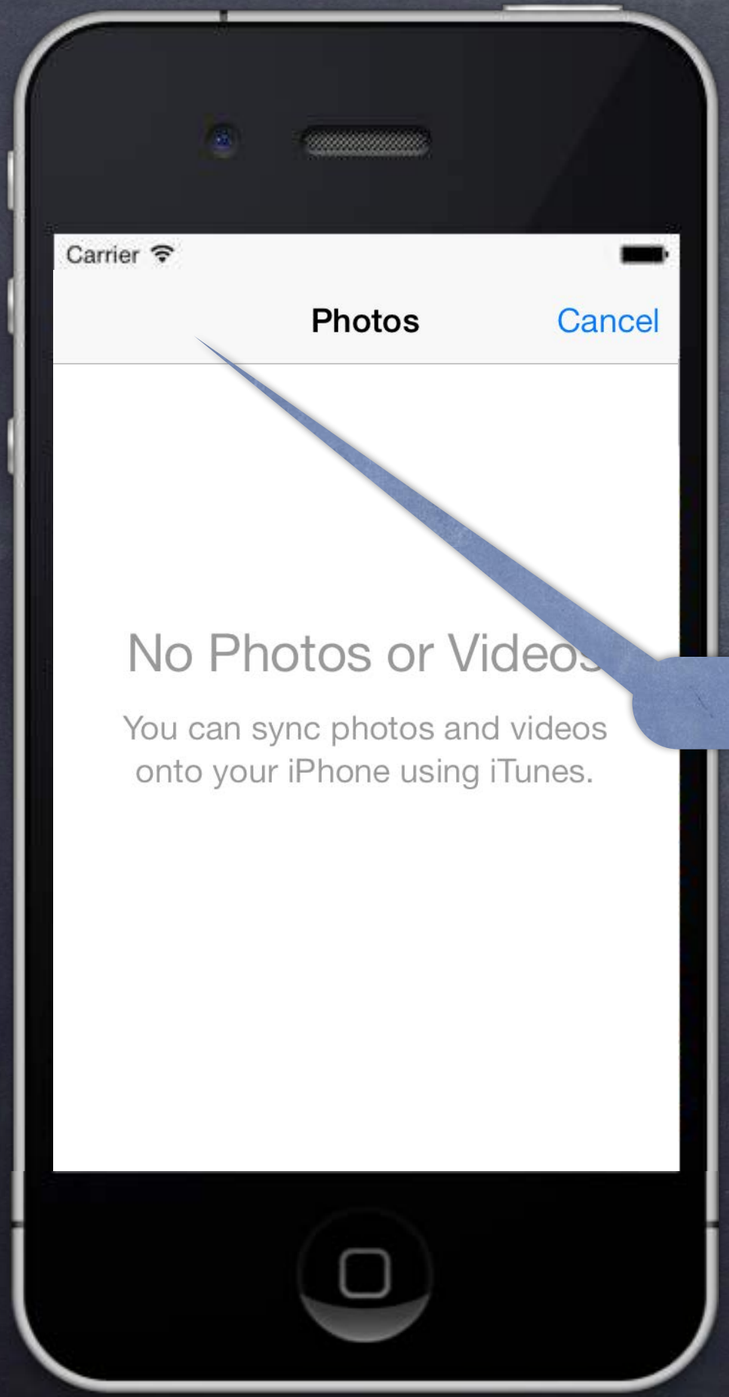
Modal View Controllers

- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



Tapping here adds a photo to this contact.
It also does so by taking over the entire screen.

Modal View Controllers



- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.

Again, no back button.

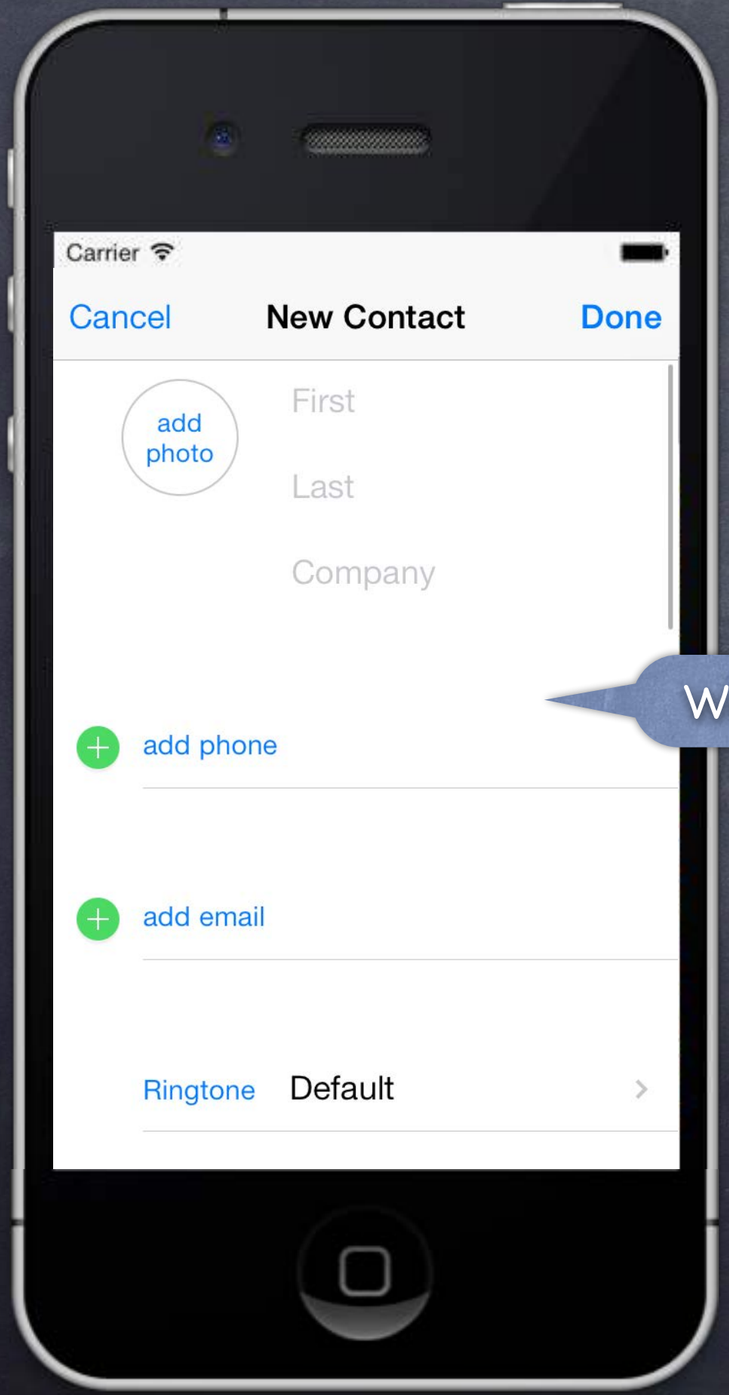
Modal View Controllers

- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.

Let's Cancel and see what happens.

Modal View Controllers

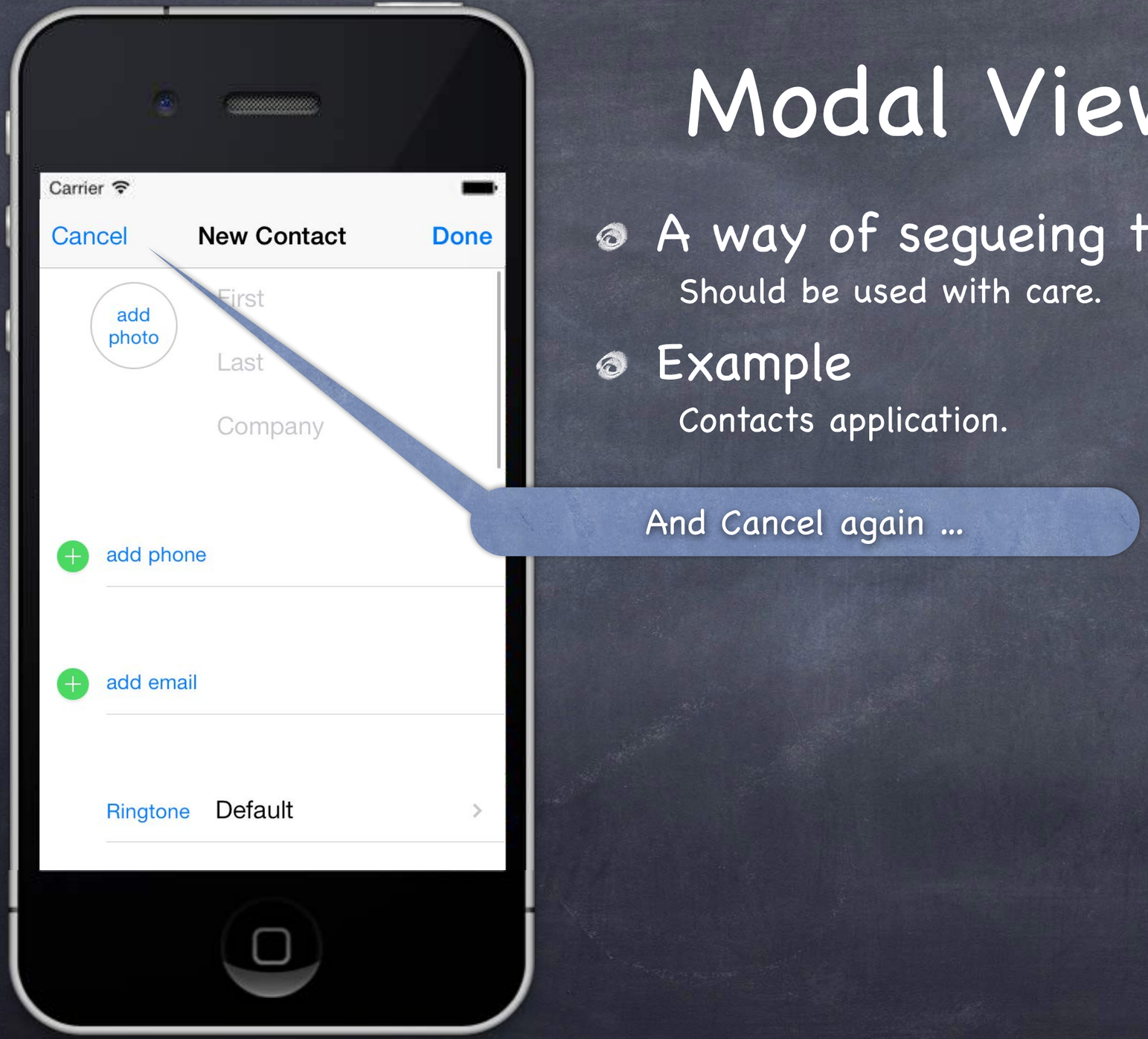
- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



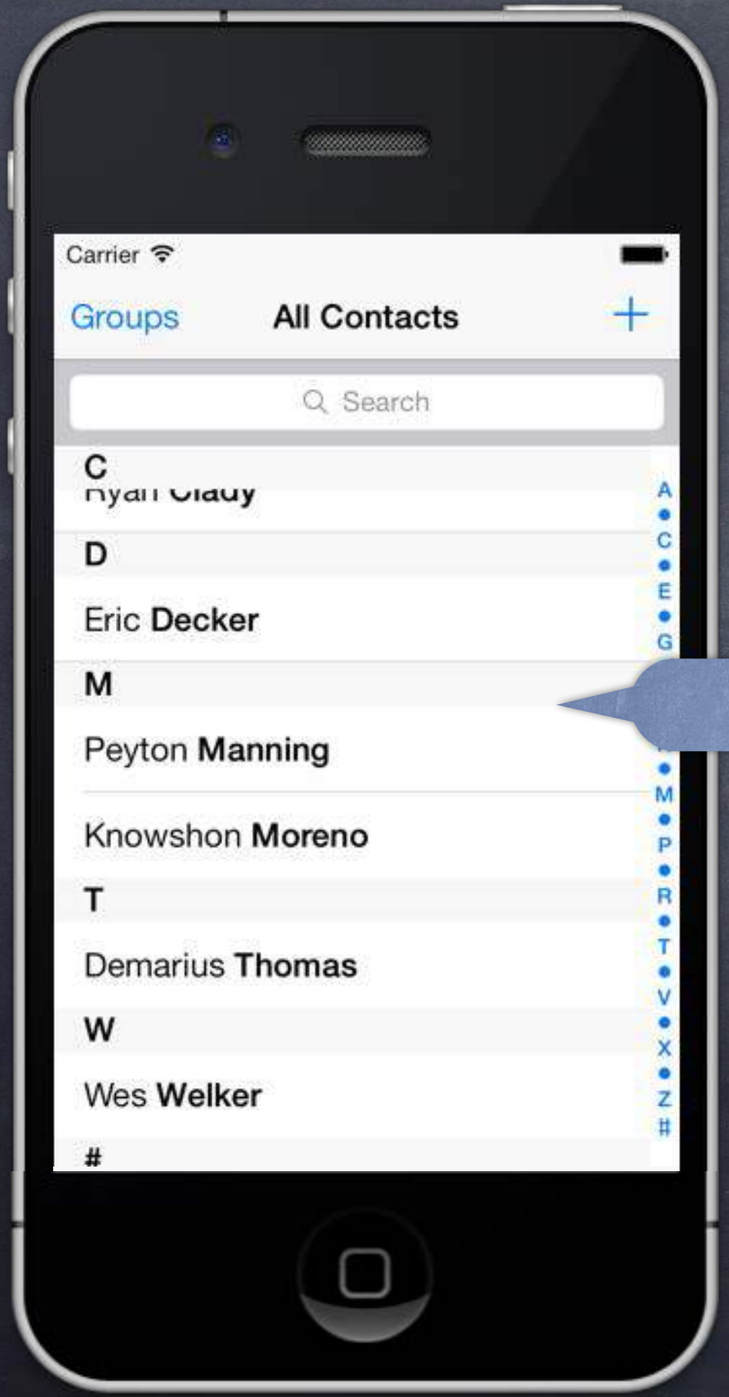
We're back to the last Modal View Controller.

Modal View Controllers

- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.



Modal View Controllers



- A way of segueing that takes over the screen
  - Should be used with care.
- Example
  - Contacts application.

Back to where we started.

Modal View Controllers

- Considerations

  The view controller we segue to using a Modal segue will take over the entire screen.
  This can be rather disconcerting to the user, so use this carefully.

- How do we set a Modal segue up?

  Just ctrl-drag from, for example, a button to another View Controller & pick segue type "Modal".
  Inspect the segue to set the style of presentation (more on this later).

  If you need to present a Modal VC not from a button, use a manual segue (last lecture).
  Or it can be done in code (not via segue) with `presentViewController:animated:completion:` method
  (that's kind of "old style" way to do it, though, pretty rare).

Modal View Controllers

- Preparing for a Modal segue

  You prepare for a Modal segue just like any other segue …

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
    if ([segue.identifier isEqualToString:@“GoToMyModalVC”]) {
        MyModalVC *vc = segue.destinationViewController;
        // set up the vc to run here
    }
}
```

- Hearing back from a Modally segue-to View Controller

  When the Modal View Controller is “done”, how does it communicate results back to presenter?
  You do this by having the segued-to View Controller “segue back” using an “unwind segue.”
  “Unwind segues” are special because they are the only segues that do not instantiate a new VC!
  Instead, they segue to an already existing VC.
  But they are limited to VC’s that “presented” the VC that is segueing back.
  This can be this Modal mechanism, but could also be, e.g., “pushing” in a navigation controller.

Modal View Controllers

- Setting up an Unwind Segue

In the presenting view controller (the one to which you want to "segue back" or "unwind"),
you implement an `IBAction` with any name, but with a `UIStoryBoardSegue` as its argument.

For example …

```
- (IBAction)done:(UIStoryboardSegue *)segue
    MyModalVC *vc = (MyModalVC *)segue.sourceViewController;
    // get results out of vc, which I presented
}
```

Then, ctrl-drag from some UI (button?) in the presented view controller's scene to this icon
in the presented view controller's scene (not in the presenter's scene).

Select the method (e.g. `done:` above) you want to use to unwind.

Now the method above will be called in the presenting view controller when that UI is activated.

When this happens, a modally presented view controller will also automatically dismiss.

The presented view controller will also be sent `prepareForSegue:sender:` before `done:` gets called.
(You can set an unwind segue's `identifier` using the Document Outline.)

Modal View Controllers

- Can you dismiss a view controller from code?

  Yes, but it is generally not the preferred way to do it (unwind instead) …

```
- (void)dismissViewControllerAnimated:(BOOL)animated
                           completion:(void (^)(void))block;
```

  You do NOT send this to the modal VC! You send it to the view controller that presented it.

- Modal view controllers dismissing themselves

  This is usually frowned upon.

  However, it sometimes happens on cancel (i.e. the user did nothing in the modal view controller).

  But you still do it by sending `dismissModalViewController:` to the presenting view controller:

```
[self.presentingViewController dismissViewControllerAnimated:YES …];
```

Modal View Controllers

- How is the modal view controller animated onto the screen?

  Depends on this property in the view controller that is being put up modally …

```
@property UIModalTransitionStyle modalTransitionStyle;
UIModalTransitionStyleCoverVertical  // slides up and down from bottom of screen
UIModalTransitionStyleFlipHorizontal // flips the current view controller view over to modal
UIModalTransitionStyleCrossDissolve  // old fades out as new fades in
UIModalTransitionStylePartialCurl    // only if presenter is full screen (and no more modal)
```

- What about iPad?

  Sometimes it might not look good for a presented view to take up the entire screen.

```
@property UIModalPresentationStyle modalPresentationStyle;  // in the modal VC
UIModalPresentationFullScreen     // full screen anyway (always on iPhone/iPod Touch)
UIModalPresentationPageSheet      // full screen height, but portrait width even if landscape
UIModalPresentationFormSheet      // centered on the screen (all else dimmed)
UIModalPresentationCurrentContext // parent's context (e.g. in a popover)
```

  Also possible for the presenting VC to control these things (see `definesPresentationContext`).

UITextField

- Like UILabel, but editable
  - Typing things in on an iPhone is secondary UI (keyboard is tiny).
  - More of a mainstream UI element on iPad.
  - Don't be fooled by your UI in the simulator (because you can use physical keyboard!).
  - You can set attributed text, text color, alignment, font, etc., just like a UILabel.
- Keyboard appears when UITextField becomes "first responder"
  - It will do this automatically when the user taps on it.
  - Or you can make it the first responder by sending it the becomeFirstResponder message.
  - To make the keyboard go away, send resignFirstResponder to the UITextField.
- Delegate can get involved with Return key, etc.
  - `- (BOOL)textFieldShouldReturn:(UITextField *)sender;` // sent when Return key is pressed
  - Oftentimes, you will `[sender resignFirstResponder]` in this method.
  - Returns whether to do normal processing when Return key is pressed (e.g. target/action).

UITextField

- Finding out when editing has ended
  - Another delegate method …
  - `- (void)textFieldDidEndEditing:(UITextField *)sender;`
  - Sent when the text field resigns being first responder.
- Finding out when the text changes
  - `UITextFieldTextDidChangeNotification`
  - You can sign up for this `NSNotification` to find out when the user changes the text.
- `UITextField` is a `UIControl`
  - So you can also set up target/action to notify you when things change.
  - Just like with a button, there are different `UIControlEvents` which can kick off an action.
  - Right-click on a `UITextField` in a storyboard to see the options available.

Keyboard

- Controlling the appearance of the keyboard

  Set the properties defined in the `UITextInputTraits` protocol (which `UITextField` implements).

```
@property UITextAutocapitalizationType autocapitalizationType; // words, sentences, etc.
@property UITextAutocorrectionType autocorrectionType;         // UITextAutocorrectionTypeYES/NO
@property UIReturnKeyType returnKeyType;                       // Go, Search, Google, Done, etc.
@property BOOL secureTextEntry;                                // for passwords, for example
@property UIKeyboardType keyboardType;                         // ASCII, URL, PhonePad, etc.
```

- The keyboard comes up over other views

  So you may need to adjust your view positioning (especially to keep the text field itself visible).

  You do this by reacting to the `UIKeyboard{Will,Did}{Show,Hide}Notifications` sent by `UIWindow`.

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(theKeyboardAppeared:)
                                             name:UIKeyboardDidShowNotification
                                           object:self.view.window];
```

  The `userInfo` in the `NSNotification` will have details about the appearance.

  `UITableViewController` listens for this and scrolls table automatically if a row has a `UITextField`.

UITextField

- Other UITextField properties

```
@property BOOL clearsOnBeginEditing;
@property BOOL adjustsFontSizeToFitWidth;
@property CGFloat minimumFontSize;  // always set this if you set adjustsFontSizeToFitWidth
@property NSString *placeholder;    // drawn in gray when text field is empty
@property UIImage *background/disabledBackground;
@property NSDictionary *defaultTextAttributes; // applies to entire text
```

- Other UITextField functionality

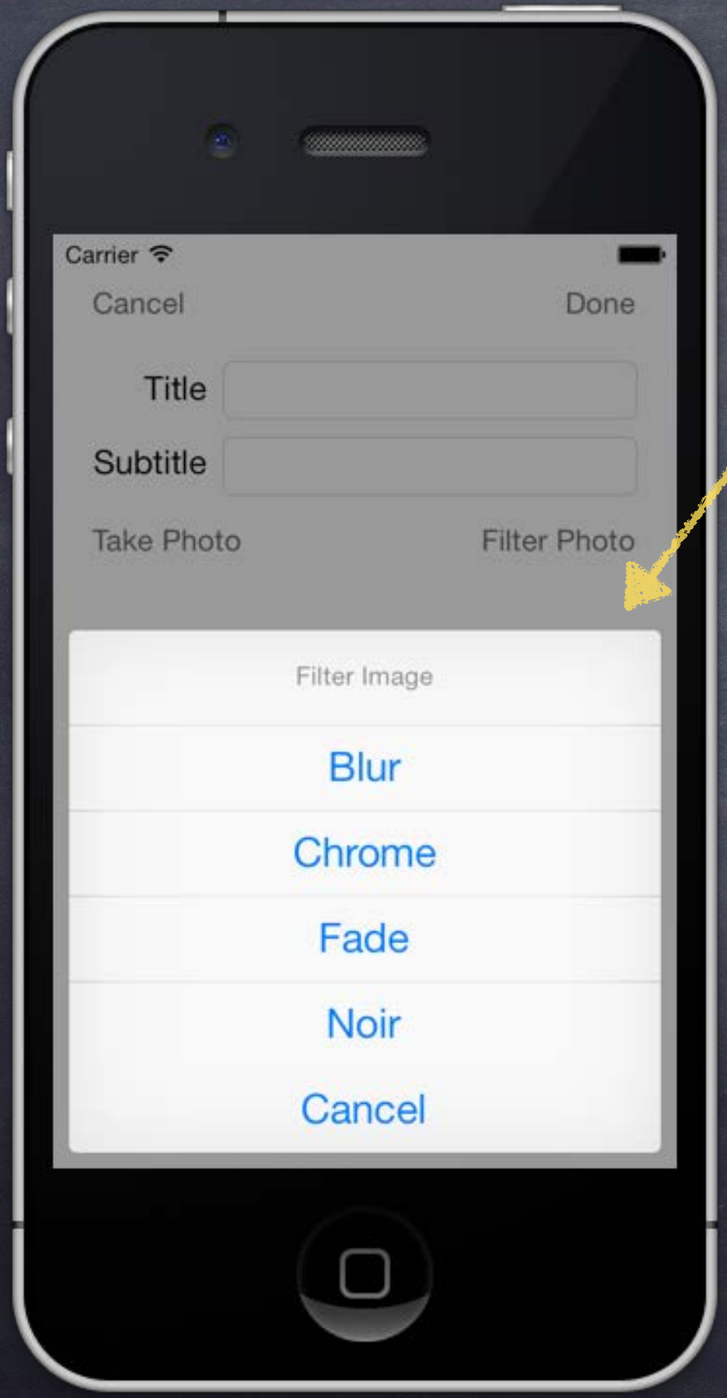
  UITextFields have a "left" and "right" overlays (similar to accessory views in MKAnnotationView).
  You can control in detail the layout of the text field (border, left/right view, clear button).

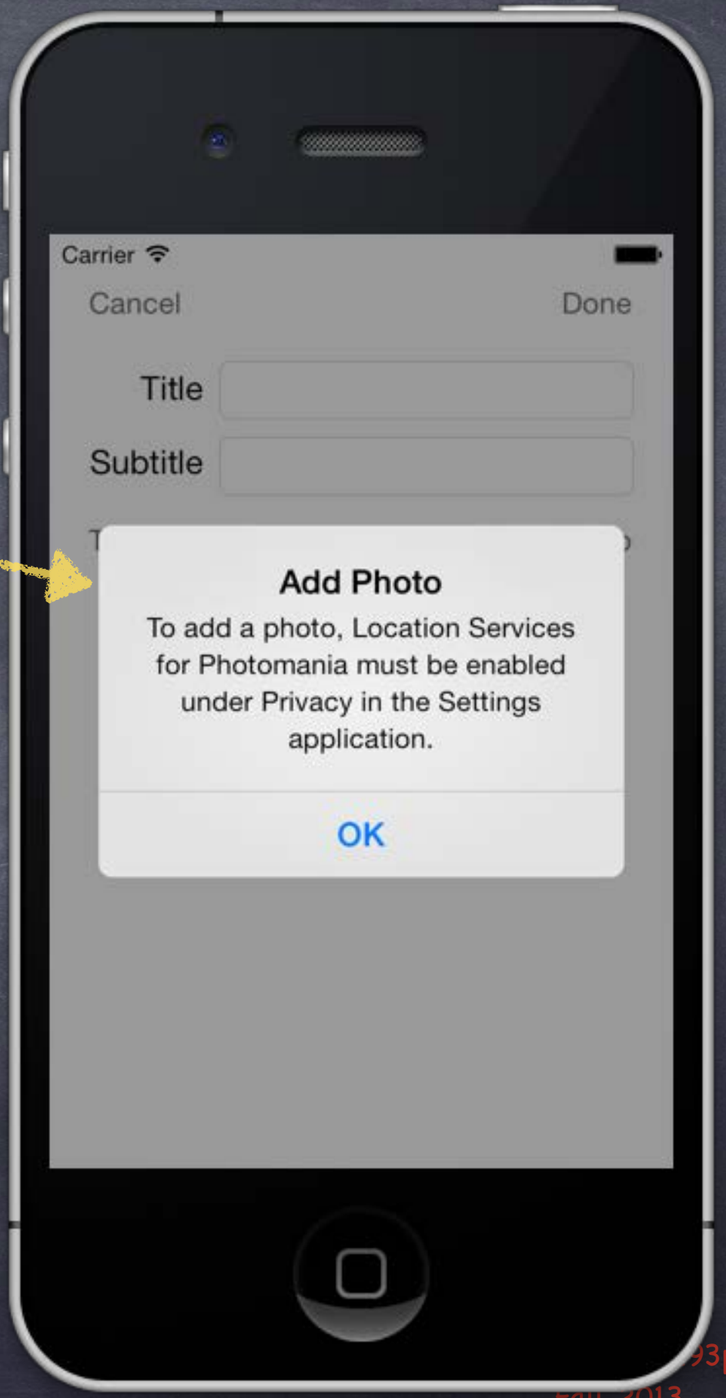
- Other Keyboard functionality

  Keyboards can have accessory views that appear above the keyboard (custom toolbar, etc.).

```
@property (retain) UIView *inputAccessoryView;  // UITextField method
```

Action Sheet & Alert





Alerts and Action Sheets

- Two kinds of "pop up and ask the user something" mechanisms
  - Alerts
  - Action Sheets
- Alerts
  - Pop up in the middle of the screen.
  - Usually ask questions with only two (or one) answers (e.g. OK/Cancel, Yes/No, etc.).
  - Can be disruptive to your user-interface, so use carefully.
  - Often used for "asynchronous" problems ("connection reset" or "network fetch failed").
- Action Sheets
  - Usually slides in from the bottom of the screen on iPhone/iPod Touch, and in a popover on iPad.
  - Can be displayed from a tab bar, toolbar, bar button item or from a rectangular area in a view.
  - Usually asks questions that have more than two answers.
  - Think of action sheets as presenting "branching decisions" to the user (i.e. what next?).

UIActionSheet

- Initializer

```
-(id)initWithTitle:(NSString *)title
          delegate:(id <UIActionSheetDelegate>)delegate
 cancelButtonTitle:(NSString *)cancelButtonTitle
destructiveButtonTitle:(NSString *)destructiveButtonTitle
 otherButtonTitles:(NSString *)otherButtonTitles, ...;
```

- And you can add more buttons programmatically

```
- (void)addButtonWithTitle:(NSString *)buttonTitle;
```

- Displaying the Action Sheet

```
UIActionSheet *actionSheet = [[UIActionSheet alloc] initWithTitle:…];
[actionSheet showInView:(UIView *)];  // centers the view on iPad (don't use this on iPad)
[actionSheet showFromRect:(CGRect) inView:(UIView *) animated:(BOOL)];   // good on iPad
[actionSheet showFromBarButtonItem:(UIBarButtonItem *) animated:(BOOL)]; // good on iPad
```

Universal apps require care here (though some can work on both platforms, e.g., `showFromRect:`).

UIActionSheet

- Finding out what the user has chosen via the delegate

```
- (void)actionSheet:(UIActionSheet *)sender didDismissWithButtonIndex:(NSInteger)index;
```

- Remember from initializer that Cancel/Destructive are special

```
@property NSInteger cancelButtonIndex;      // don't set this if you set it in initializer
@property NSInteger destructiveButtonIndex; // don't set this if you set it in initializer
```

- Other indexes

```
@property (readonly) NSInteger firstOtherButtonIndex;
@property (readonly) NSInteger numberOfButtons;
- (NSString *)buttonTitleAtIndex:(NSInteger)index;
```

The "other button" indexes are in the order you specified them in initializer and/or added them.

- You can programmatically dismiss the action sheet as well

```
- (void)dismissWithClickedButtonIndex:(NSInteger)index animated:(BOOL)animated;
```

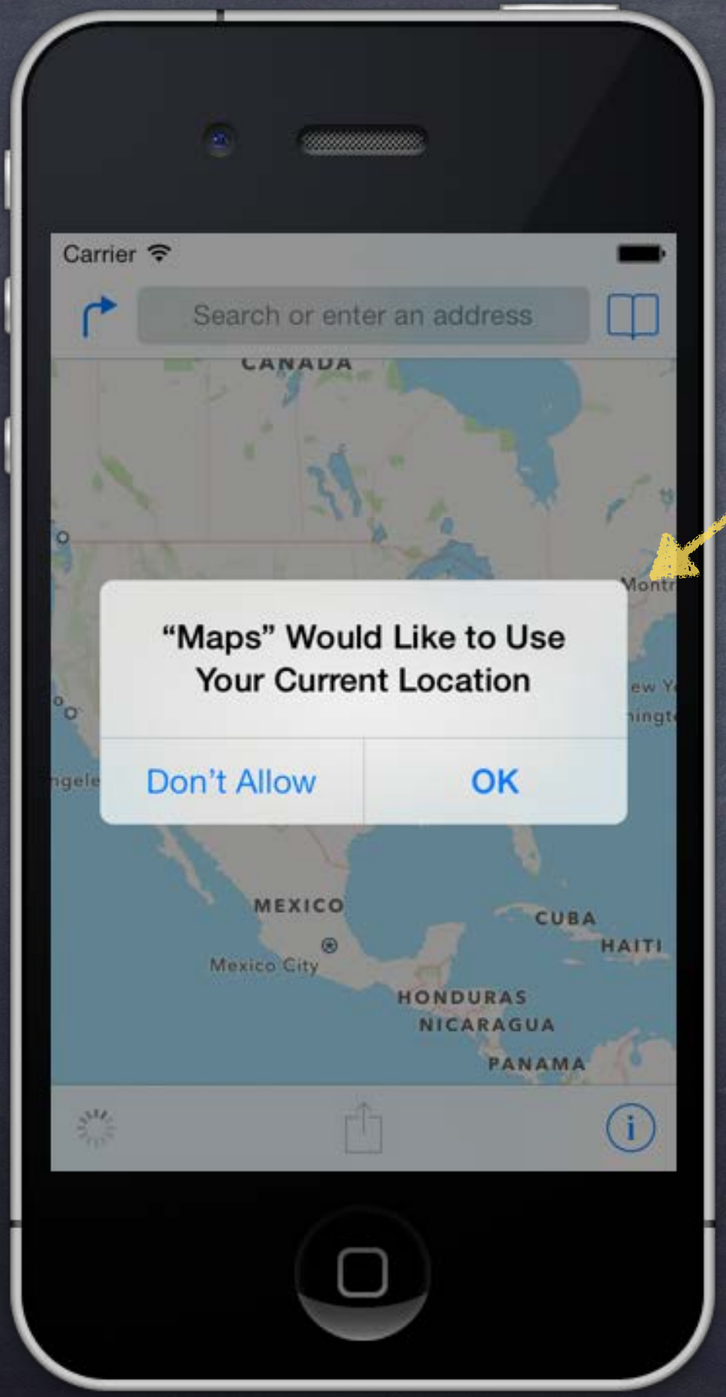
It is generally recommended to call this on `UIApplicationDidEnterBackgroundNotification`.
Remember also that you might be terminated while you are in the background, so be ready.

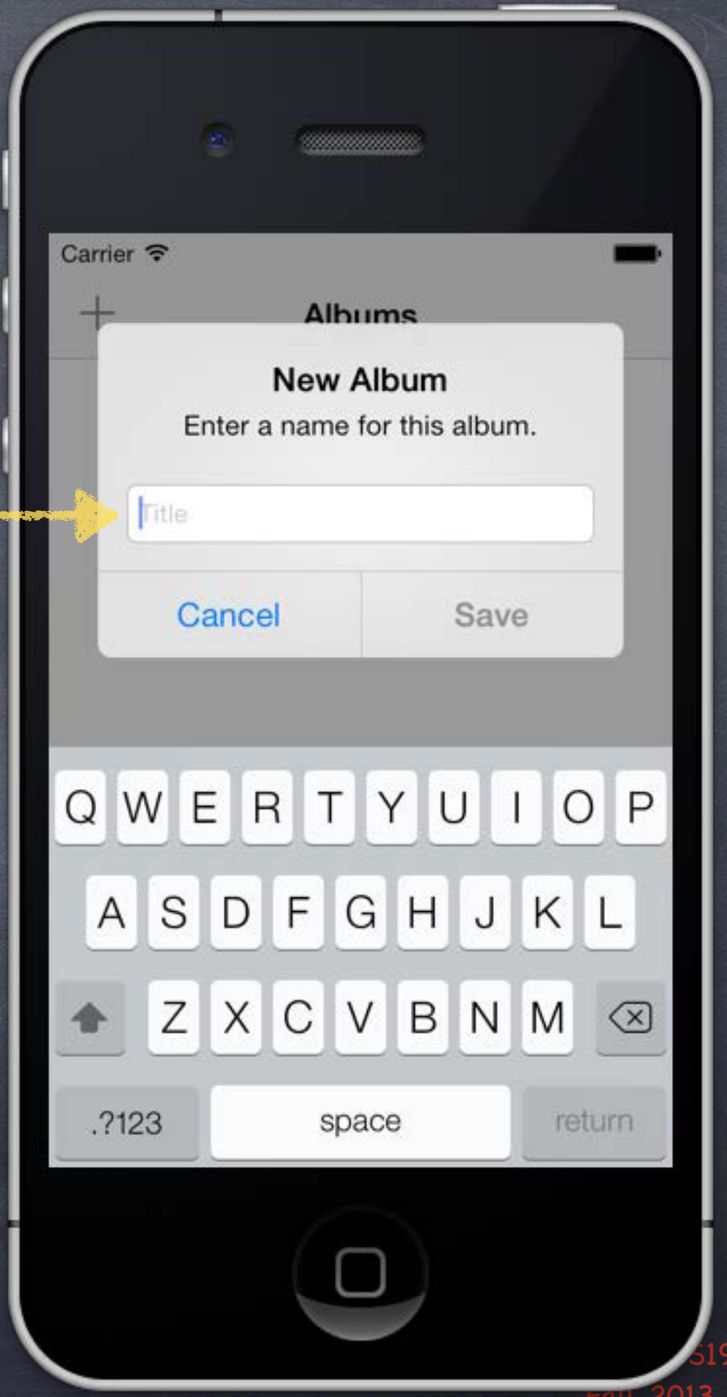
UIActionSheet

- Special popover considerations: no Cancel button
  - An action sheet in a popover (that is not inside a popover) does not show the cancel button.
  - It does not need one because clicking outside the popover dismisses it.
  - It will automatically not show the Cancel button (just don't be surprised that it's not there).
- Special popover considerations: the popover's `passthroughViews`
  - If you `showFromBarButtonItem:animated:`, it adds the toolbar to popover's `passthroughViews`.
  - This is annoying because repeated touches on the bar button item give multiple action sheets!
  - Also, other buttons in your toolbar will work (which might or might not make sense).
  - Unfortunately, you just have to handle this in all of your bar buttons, including the action sheet's.
- Special popover considerations: bar button item handling
  - Have a `weak @property` in your class that points to the `UIActionSheet`.
  - Set it right after you show the action sheet.
  - Check that `@property` at the start of your bar button item's action method.
  - If it is not-`nil` (since it is `weak`, it will only be non-`nil` if it's still on-screen), just dismiss it.
  - If it is `nil`, prepare and show your action sheet.

UIAlertView

Multiple Buttons
&
Embedded Views





UIAlertView

- Very similar to Action Sheet …

```
-(id)initWithTitle:(NSString *)title
           message:(NSString *)message // different from UIActionSheet
          delegate:(id <UIActionSheetDelegate>)delegate
 cancelButtonTitle:(NSString *)cancelButtonTitle
 otherButtonTitles:(NSString *)otherButtonTitles, ...;
```

- And you can add more buttons programmatically

```
- (void)addButtonWithTitle:(NSString *)buttonTitle;
```

- Displaying the Action Sheet

```
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:…];
[alert show];    // different from UIActionSheet, always appears in center of screen
```

- You can even have a UITextField in your Alert

```
alert.alertViewStyle = UIAlertViewStyle{SecureText,PlainText,LoginAndPassword}Input;
```

`[alertView textFieldAtIndex:0]` gives you the `UITextField` (1 is password in `LoginAndPassword`)

Demo

- Photomania Add Photo
  - Let user add a photo to our Photomania database using the camera.
  - We probably won't actually get to the "camera" part today!
  - But we'll set up for that by creating a Modally-segued-to View Controller to do it.
  - Watch for ... Modal Segue, Unwinding Segue, Text Field, Alert

Coming Up

- Wednesday
  - Demo Continued
  - `UIImagePickerController` (Camera)
  - Core Motion
- Friday
  - Sprite Kit
- Next Week
  - Thanksgiving

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Today

- Demo Odds & Ends
  - Cleaning up unused image URLs.
  - A bit more Core Location error checking.
- Camera
  - Actually taking the photo.
  - Finish off Photomania Demo (also includes Action Sheet).
- Core Motion
  - Tracking the device's movement in space.
- Demo
  - Simple game based on Core Motion.
- Application Lifecycle
  - Application Delegate Methods and NSNotifications.

Demo

- Photomania Add Photo (continued)
  - Cleaning up unused image URLs.
  - A bit more Core Location error checking.

UIImagePickerController

- Modal view to get media from camera or photo library
  - Modal means you put it up with `presentViewController:animated:completion:`.
  - On iPad, you might also put it up in a `UIPopoverController`.
- Usage
  1. Create it with `alloc/init` and set `delegate`.
  2. Configure it (source, kind of media, user editability).
  3. Present it.
  4. Respond to delegate method when user is done picking the media.
- What the user can do depends on the platform
  - Some devices have cameras, some do not, some can record video, some can not.
  - Also, you can only offer camera OR photo library on iPad (not both together at the same time).
  - As with all device-dependent API, we want to start by check what's available.
  - `+ (BOOL)isSourceTypeAvailable:(UIImagePickerControllerSourceType)sourceType;`
  - Source type is `UIImagePickerControllerSourceTypePhotoLibrary/Camera/SavedPhotosAlbum`

UIImagePickerController

- But don't forget that not every source type can give video
  - So, you then want to check …
  - `+ (NSArray *)availableMediaTypesForSourceType:(UIImagePickerControllerSourceType)sourceType;`
  - Returns an array of strings you check against constants.
  - Check documentation for all possible, but there are two key ones …
  - `kUTTypeImage` // pretty much all sources provide this
  - `kUTTypeMovie` // audio and video together, only some sources provide this

UIImagePickerController

- But don't forget that not every source type can give video

  So, you then want to check …

  `+ (NSArray *)availableMediaTy`

  Returns an array of strings yo

  Check documentation for … possible, but there are two key ones …

  `kUTTypeImage` // pretty much all sources provide this

  `kUTTypeMovie` // audio and video together, only some sources provide this

  These are declared in the MobileCoreServices framework.
  `#import <MobileCoreServices/MobileCoreServices.h>`
  and add MobileCoreServices to your list of linked frameworks.

- You can get even more specific about front/rear cameras

  (Though usually this is not necessary.)

  `+ (BOOL)isCameraDeviceAvailable:(UIImagePickerControllerCameraDevice)cameraDevice;`

  Either `UIImagePickerControllerCameraDeviceFront` or `UIImagePickerControllerCameraDeviceRear`.

  Then check out more about each available camera:

  `+ (BOOL)isFlashAvailableForCameraDevice:(UIImagePickerControllerCameraDevice);`

  `+ (NSArray *)availableCaptureModesForCameraDevice:(UIImagePickerControllerCameraDevice);`

  This array contains NSNumber objects with constants `UIImagePic…lerCaptureModePhoto/Video`.

UIImagePickerController

- Set the source and media type you want in the picker

(From here out, UIImagePickerController will be abbreviated UIIPC for space reasons.)

```
UIIPC *picker = [[UIIPC alloc] init];
picker.delegate = self;  // self has to say it implements UINavigationControllerDelegate too
if ([UIIPC isSourceTypeAvailable:UIIPCSourceTypeCamera]) {
    picker.sourceType = UIIPCSourceTypeCamera;
} // else we'll take what we can get (photo library by default)
NSString *desired = (NSString *)kUTTypeMovie;  // e.g., could be kUTTypeImage
if ([[UIIPC availableMediaTypesForSourceType:picker.sourceType] containsObject:desired]) {
    picker.mediaTypes = @[desired];
    // proceed to put the picker up
} else {
    // fail, we can't get the type of media we want from the source we want
}
```

Notice the cast to NSString here.
kUTTypeMovie (and kUTTypeImage) are CFStrings (Core Foundation strings).
Unfortunately, the cast is required to avoid a warning here.

UIImagePickerController

- Editability

  ```
  @property BOOL allowsEditing;
  ```

  If YES, then the user will have opportunity to edit the image/video inside the picker.

  When your delegate is notified that the user is done, you'll get both raw and edited versions.

- Limiting Video Capture

  ```
  @property UIIPCQualityType videoQuality;
  UIIPCQualityTypeMedium    // default
  UIIPCQualityTypeHigh
  UIIPCQualityType640x480
  UIIPCQualityTypeLow
  UIPCQualityTypeIFrame1280x720     // native on some devices
  UIPCQualityTypeIFrame960x540      // native on some devices
  @property NSTimeInterval videoMaximumDuration;
  ```

- Other

  You can control which camera is used, how flash is used, etc., as well (or user can choose).

UIImagePickerController

- Present the picker

  Note that on iPad, if you are not offering Camera, you must present with popover.
  If you are offering the Camera on iPad, then full-screen is preferred.
  Remember: on iPad, it's Camera OR Photo Library (not both at the same time).

- Delegate will be notified when user is done

```
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info
{
    // extract image/movie data/metadata here, more on the next slide
    [self dismissViewControllerAnimated:YES completion:...]; // or popover dismissal
}
```

- Also dismiss it when cancel happens

```
- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker
{
    [self dismissViewControllerAnimated:YES completion:…]; // or popover dismissal
}
```

  If on iPad, you'll want to implement popover's `didDismissPopover…` delegate method too.

UIImagePickerController

- What is in that info dictionary?

```
UIImagePickerControllerMediaType          // kUTTypeImage or kUTTypeMovie
UIImagePickerControllerOriginalImage      // UIImage
UIImagePickerControllerEditedImage        // UIImage
UIImagePickerControllerCropRect           // CGRect (in an NSValue)
UIImagePickerControllerMediaMetadata      // NSDictionary info about the image
UIImagePickerControllerMediaURL           // NSURL edited video
UIImagePickerControllerReferenceURL       // NSURL original (unedited) video
```

- Saving taken images or video into the device's photo library

  Check out `ALAssetsLibrary`.

UIImagePickerController

- Overlay View
  - `@property UIView *cameraOverlayView;`
  - Be sure to set this view's `frame` properly.
  - Camera is always full screen (on iPhone/iPod Touch anyway): `UIScreen`'s `bounds` property.
  - But if you use the built-in controls at the bottom, you might want your view to be smaller.
- Hiding the normal camera controls (at the bottom)
  - `@property BOOL showsCameraControls;`
  - Will leave a blank area at the bottom of the screen (camera's aspect 4:3, not same as screen's).
  - With no controls, you'll need an overlay view with a "take picture" (at least) button.
  - That button should send `- (void)takePicture` to the picker.
  - Don't forget to `dismissModalViewController:` when you are done taking pictures.
- You can zoom or translate the image while capturing
  - `@property CGAffineTransform cameraViewTransform;`
  - For example, you might want to scale the image up to full screen (some of it will get clipped).

Demo

- Photomania Add Photo (continued)
  - Photo taking.
  - Filtering via Action Sheet.

Core Motion

- API to access motion sensing hardware on your device
- Primary inputs: Accelerometer, Gyro, Magnetometer

  Not all devices have all inputs (e.g. only iPhone4-5 and 4th G iPod Touch and iPad 2 have a gyro).
- Class used to get this input is `CMMotionManager`

  Create with `alloc/init`, but use only one instance per application (else performance hit).

  It is a "global resource," so getting one via a class method somewhere is okay.
- Usage

  1. Check to see what hardware is available.
  2. Start the sampling going and poll the motion manager for the latest sample it has.

  ... or ...

  1. Check to see what hardware is available.
  2. Set the rate at which you want data to be reported from the hardware,
  3. Register a block (and a dispatch queue to run it on) each time a sample is taken.

Core Motion

- Checking availability of hardware sensors

  ```
  @property (readonly) BOOL {accelerometer,gyro,magnetometer,deviceMotion}Available;
  ```

  The "device motion" is a combination of all available (accelerometer, magnetometer, gyro).

  We'll talk more about that in a couple of slides.

- Starting the hardware sensors collecting data

  You only need to do this if you are going to poll for data.

  ```
  - (void)start{Accelerometer,Gyro,Magnetometer,DeviceMotion}Updates;
  ```

- Is the hardware currently collecting data?

  ```
  @property (readonly) BOOL {accelerometer,gyro,magnetometer,deviceMotion}Active;
  ```

- Stop the hardware collecting data

  It is a performance hit to be collecting data, so stop during times you don't need the data.

  ```
  - (void)stop{Accelerometer,Gyro,Magnetometer,DeviceMotion}Updates;
  ```

Core Motion

- Checking the data (polling not recommended, more later)

```
@property (readonly) CMAccelerometerData *accelerometerData;
```

CMAccelerometerData object provides `@property (readonly) CMAcceleration acceleration;`

```
typedef struct { double x; double y; double z; } CMAcceleration; // x, y, z in "g"
```

This raw data includes acceleration due to gravity.

```
@property (readonly) CMGyroData *gyroData;
```

CMGyroData object has one `@property (readonly) CMRotationRate rotationRate;`

```
typedef struct { double x; double y; double z; } CMRotationRate; // x, y, z in rads/sec
```

Sign of rotation rate follows right hand rule. This raw data will be biased.

```
@property (readonly) CMMagnetometerData *magnetometerData;
```

CMMagnetometerData object has one `@property (readonly) CMMagneticField magneticField;`

```
typedef struct { double x; double y; double z; } CMMagneticField; // x, y, z in microteslas
```

This raw data will be biased.

```
@property (readonly) CMDeviceMotion *deviceMotion;
```

CMDeviceMotion is an intelligent combination of gyro and acceleration.

If you have multiple detection hardware, you can report better information about each.

CMDeviceMotion

- Acceleration Data in CMDeviceMotion

```
@property (readonly) CMAcceleration gravity;
@property (readonly) CMAcceleration userAcceleration;  // gravity factored out using gyro
typedef struct { double x; double y; double z; } CMAcceleration; // x, y, z in "g"
```

- Rotation Data in CMDeviceMotion

```
@property CMRotationRate rotationRate;  // bias removed from raw data using accelerometer
typedef struct { double x; double y; double z; } CMRotationRate; // x, y, z in rads/sec

@property CMAttitude *attitude;         // device's attitude (orientation) in 3D space

@interface CMAttitude : NSObject        // roll, pitch and yaw are in radians
@property (readonly) double roll;       // around longitudinal axis passing through top/bottom
@property (readonly) double pitch;      // around lateral axis passing through sides
@property (readonly) double yaw;        // around axis with origin at center of gravity and
                                        // perpendicular to screen directed down
   // other mathematical representations of the device's attitude also available
@end
```

CMDeviceMotion

- Magnetic Field Data in CMDeviceMotion

```
@property (readonly) CMCalibratedMagneticField magneticField;
struct {
    CMMagneticField field;
    CMMagneticFieldCalibrationAccuracy accuracy;
} CMCalibratedMagneticField;
enum {
    CMMagneticFieldCalibrationAccuracyUncalibrated,
                                      Low,
                                      Medium,
                                      High
} CMMagneticFieldCalibrationAccuracy;
```

Core Motion

- Registering a block to receive Accelerometer data

```
- (void)startAccelerometerUpdatesToQueue:(NSOperationQueue *)queue
                             withHandler:(CMAccelerometerHandler)handler;
typedef void (^CMAccelerationHandler)(CMAccelerometerData *data, NSError *error);
```

queue == [[NSOperationQueue alloc] init] or [NSOperation mainQueue (or currentQueue)].

- Registering a block to receive Gyro data

```
- (void)startGyroUpdatesToQueue:(NSOperationQueue *)queue
                    withHandler:(CMGyroHandler)handler;
typedef void (^CMGyroHandler)(CMGyroData *data, NSError *error)
```

- Registering a block to receive Magnetometer data

```
- (void)startMagnetometerUpdatesToQueue:(NSOperationQueue *)queue
                             withHandler:(CMMagnetometerHandler)handler;
typedef void (^CMMagnetometerHandler)(CMMagnetometerData *data, NSError *error)
```

Core Motion

- Registering a block to receive (intelligently) combined data

```
- (void)startDeviceMotionUpdatesToQueue:(NSOperationQueue *)queue
                            withHandler:(CMDeviceMotionHandler)handler;
typedef void (^CMDeviceMotionHandler)(CMDeviceMotion *motion, NSError *error);
Interesting NSError types: CMErrorDeviceRequiresMovement/CMErrorTrueNorthNotAvailable

- (void)startDeviceMotionUpdatesUsingReferenceFrame:(CMAttitudeReferenceFrame)frame
                                            toQueue:(NSOperationQueue *)queue
                                        withHandler:(CMDeviceMotionHandler)handler;
enum {
    CMAttitudeReferenceFrameXArbitraryZVertical,
                            XArbitraryCorrectedZVertical, // needs magnetometer; ++CPU
                            XMagneticZVertical,           // above + device movement
                            XTrueNorthZVertical           // requires GPS + magnetometer
}
@property (nonatomic) BOOL showsDeviceMovementDisplay; // whether to put up UI if required
```

Core Motion

- Setting the rate at which your block gets executed

```
@property NSTimeInterval accelerometerUpdateInterval;
@property NSTimeInterval gyroUpdateInterval;
@property NSTimeInterval magnetometerUpdateInterval;
@property NSTimeInterval deviceMotionUpdateInterval;
```

- It is okay to add multiple handler blocks

  Even though you are only allowed one `CMMotionManager`.
  However, each of the blocks will receive the data at the same rate (as set above).
  (Multiple objects are allowed to poll at the same time as well, of course.)

Demo

- Bouncer

  Using Accelerometer information to drive our user-interface.

Application State

- When your application's UI starts/stops receiving events ...
  - Your Application Delegate gets ...
  - `- (void)applicationDidBecomeActive:(UIApplication *)sender;`
  - `- (void)applicationWillResignActive:(UIApplication *)sender;`
  - Everyone gets these radio station broadcasts ...
  - `UIApplicationDidBecomeActiveNotification`
  - `UIApplicationWillResignActiveNotification`
  - These might happen because user switched to another app or maybe a phone call come in.
  - Use these notifications to pause doing stuff in your UI and then restart it later.

Application State

- When you enter the background …
  - You only get a few seconds to respond to this.
  - `- (void)applicationDidEnterBackground:(UIApplication *)sender;`
  - and `UIApplicationDidEnterBackgroundNotification`
  - If you need more time, it is possible (see `beginBackgroundTaskWithExpirationHandler:`).
  - This is a notification for you to clean up any significant resource usage, etc.
- You find out when you get back to the foreground too …
  - Your Application Delegate gets …
  - `- (void)applicationWillEnterForeground:(UIApplication *)sender;`
  - and `UIApplicationWillEnterForegroundNotification`
  - Generally you undo whatever you did in `DidEnterBackground`.
  - You'll get `applicationDidBecomeActive:` soon after receiving the above.

Application State

- Other Application Delegate items of interest …
  - Local Notifications (set timers to go off at certain times … will wake your application if needed).
  - State Restoration (saving the state of your UI so that you can restore it even if you are killed).
  - Data Protection (files can be set to be protected when a user's device's screen is locked).
  - Open URL (in Xcode's Info tab of Project Settings, you can register for certain URLs).

Coming Up

- Friday
  Sprite Kit
- Next Week
  Thanksgiving
- Final Project
  Do Not Procrastinate

Stanford CS193p

Developing Applications for iOS
Fall 2013-14

Coming Up

- Wednesday
  - Alternate Final Presentation.
  - If you are using Alternate Presentation time, submit your Keynote by noon tomorrow (Tuesday).
  - Submit the slides using the normal submit script (submit again with code by Sunday).
  - We will have a "live demo testing" opportunity on Wednesday as well, so bring your demo device.
- Friday
  - No Section.
- Sunday
  - Final Project Due (by midnight).
  - Don't forget to submit your Keynote slides along with!
- Final
  - A week from Thursday at 12:15pm to 3:15pm in this room.
  - Presentation is required.
  - Presentation time limit is 2.5 minutes (150 seconds) and must be 1280x720 aspect ratio.
  - Presentation order is random (no exceptions).

Today

- Localization
  Internationalization really.
- Settings
  Adding UI to the Settings application.
- Demo
  Internationalizing Photomania.
  Adding a Bouncer setting.

Internationalization

- Two steps to making international versions of your application
  - Internationalization (i18n)
  - Localization (l10n)
- Internationalization
  - This is a process of making strings externally editable (from storyboard or code).
  - It also involves using certain "formatting" classes for things like dates, numbers, etc.
  - You (the developer) get to do this work.
- Localization
  - A process of editing those externalized strings (and then QA'ing the result) for a given language.
  - You usually hire a localization company to do this work.

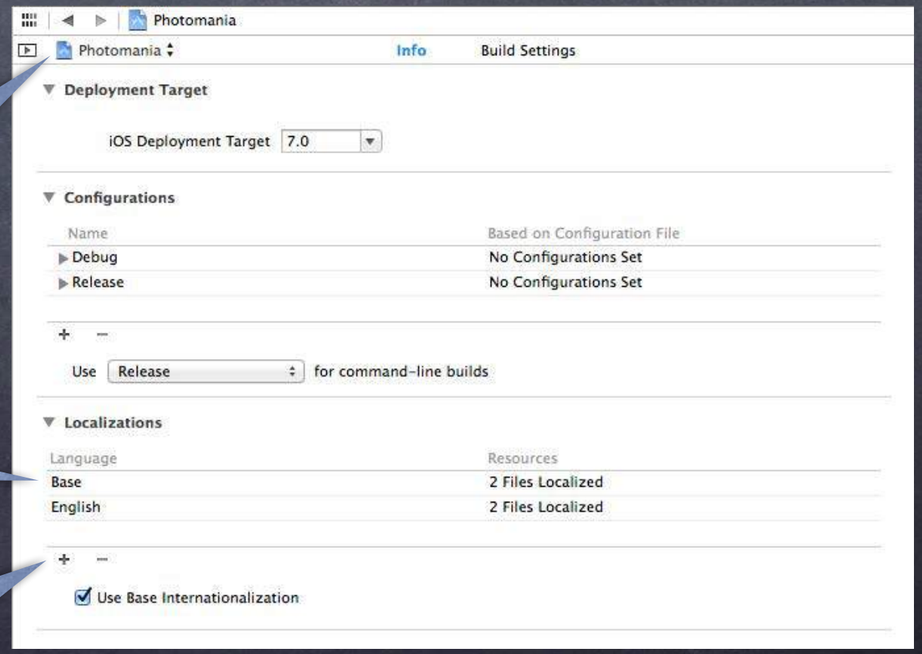
Internationalization

- Storyboards are localized by changing its strings only
  And we rely on Autolayout to make it all look nice.
- First step though: Registering Localizable Languages
  Go to the Project pane in Xcode (top in Navigator), then Info tab to add Localizations.
  If you click "Use Base Internationalization" the strings in your storyboards will be extracted into editable `.strings` files (one for each language).

You must inspect the project itself here, not the Target you build.

"Base" is the "localization" where storyboards live that are localizable using only `.strings` files (hopefully this is all storyboards).

Click this + to add more languages that you intend to support.

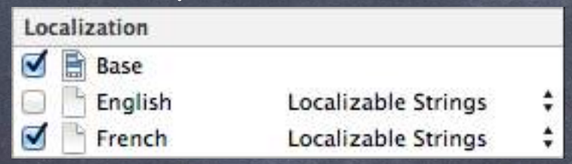


Localizing Storyboards

- Storyboards in Navigator will now have localizations
  - Send the `.strings` files out to localizers to translate the strings.
  - Localizers appreciate a demo of your application in your Base language.
  - Or at least send them the storyboards so they can get context.

Navigator

File Inspector



Internationalization

- What about strings not in storyboards?

  i.e., literal strings @"string"

  Replace them with a variant of NSLocalizedString …

```
NSString *NSLocalizedStringWithDefaultValue(NSString *key, NSString *table,
                                            NSString *bundle, NSString *defaultValue,
                                            NSString *comment); // comment is for localizers
```

  Also NSLocalizedStringFromTableInBundle() (defaultValue is the key)

  and NSLocalizedStringFromTable() (defaultValue is the key and uses mainBundle)

  and NSLocalizedString() (defaultValue is key; mainBundle; table Localizable.strings)

  Example: Change @"hello" to NSLocalizedString(@"hello", @"Greeting at start of application.")

- What these macros do …

  They send this method to [NSBundle mainBundle] (or the specified bundle if macro takes one) …

```
- (NSString *)localizedStringForKey:(NSString *)key
                              value:(NSString *)defaultValue // if nil, will be key
                              table:(NSString *)tableName;   // if nil: Localizable.strings
```

Localization

- Generating `.strings` files with `genstrings`

  Once you have used `NSLocalizedString` and its variants to eliminate literal strings …

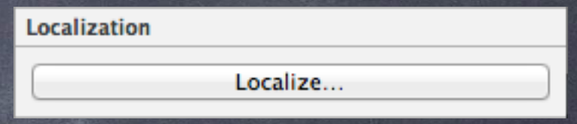
  You can use the command line utility `genstrings` to generate `.strings` files from `.m` files.

  ```
  > cd <directory where all your .m files are>
  > genstrings *.m
  ```

  Example: `NSLocalizedString(@"hello", @"Greeting at start of application.")`

  … would generate an entry in `Localizable.strings` which looks like this …

  ```
  /* Greeting at start of application. */
  "hello" = "hello";
  ```



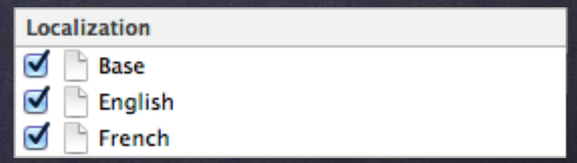
- Drag the `.strings` into Xcode and then inspect to Localize

  Hit the button "Localize" in the File Inspector on the strings file or storyboard.

  You can then pick languages for which there is a localization set up for your application.

  (As per the first slide on this topic.)

  E.g., French localizers would change entry to `"hello" = "bonjour"`.



Bundles

- Resources are drawn from a "bundle" using the user's locale
  - Inside a bundle, there will be "`.lproj`" directories (e.g. `en.lproj`, `fr.lproj`, etc.).
  - Inside these `.lproj` directories, there will be `.strings` files, images, sounds, etc.
  - When you get a path to a file from a bundle, it tries top-level first, then searches `.lprojs` (depending on the language the user has chosen for his system in Settings app).
- Bundles can be associated with a framework or an application
- Using `NSBundle` API to get a resource (e.g. an image or sound)

```
NSBundle *bundle = [NSBundle bundleForClass:[self class]];
NSString *path = [bundle pathForResource:@"speedlimit" ofType:@"jpg"];
```

`bundleForClass:` knows whether that class came from a framework or just with the application.

Localization

- Debugging

  Set the `NSUserDefault` `NSShowNonLocalizedStrings` to `YES` and a message will be logged to the console whenever these `NSLocalizedString` methods cannot find a string.

- Build Clean

  If changes you make to `.strings` files don't seem to be appearing when you run … try Build Clean. Usually this is not necessary, but it's something to try if things get out of sync.

Locales

- Formats

  Dates and numbers are written in different formats in different locales.

- Locale

  Locale is different from language.

  The `NSLocale` class encapsulates the locale the user has chosen in Settings.

  It knows all about date and number formats (independent of the language that is currently set).

```
+ (NSLocale *)currentLocale;
+ (NSLocale *)autoupdatingCurrentLocale; // watch NSCurrentLocaleDidChangeNotification
```

  Usually you don't need to access this directly because you'll use a formatter which is looking at it.

NSNumberFormatter

- Lots going on here. Check out the documentation.
  But we'll look at two simple cases …
- Displaying numbers
  Shouldn't really use `[NSString stringWithFormat:@"%g"]` for user-visible floats.
  Instead use this NSNumberFormatter class method …

```
+ (NSString *)localizedStringFromNumber:(NSNumber *)number
                            numberStyle:(NSNumberFormatterStyle)style
```

  Example styles: NSNumberFormatter`DecimalStyle` or `CurrencyStyle` or even `SpellOutStyle`
- Parsing numbers
  Don't use `intValue` to parse a number typed in by the user, use …

```
NSNumberFormatter *formatter = [[NSNumberFormatter alloc] init];
[formatter setNumberStyle:NSNumberFormatterDecimalStyle];
NSNumber *parsedNumber = [formatter numberFromString:userInputtedString];
```

  Note that this will return `nil` if a number of the proper format is not found.
  That can be valuable to differentiate from the user entering "zero" for example.

NSDateFormatter

- Dates are rather complicated to display properly

  If you are presenting dates to the user, familiarize yourself with these concepts …

  Calendars. Not all locales use the Gregorian calendar that we do. `NSCalendar`.

  Date Components, e.g., what is a "month" (calendar dependent)? `NSDateComponents`.

  And if you have in mind something like MM/DD/YYYY, check out this method first …

```
+ (NSString *)dateFormatFromTemplate:(NSString *)template
                              options:(NSUInteger)options
                               locale:(NSLocale *)locale;
```

- Simple date formatting

  At least use this `NSDateFormatter` class method …

```
+ (NSString *)localizedStringFromDate:(NSDate *)date
                            dateStyle:(NSDateFormatterStyle)dateStyle
                            timeStyle:(NSDateFormatterStyle)timeStyle;
```

  Example styles: `NSDateFormatterShortStyle` or `MediumStyle` or `LongStyle` or `FullStyle`

NSString

- Searching in strings

  Do not use plain `rangeOfString:` if you are looking around in user-inputted strings.

  Instead, use this …

```
+ (NSRange)rangeOfString:(NSString *)useEnteredSubstring
                 options:(NSStringCompareOptions)options // e.g. case-insensitively
                   range:(NSRange)rangeToSearchIn
                  locale:(NSLocale *)locale;
```

  … especially if you are searching case-insensitively, since this concept is locale-specific.

UIImage

- The method `imageNamed:` does the right thing!
  - It searches inside the `.lproj`'s to find images.

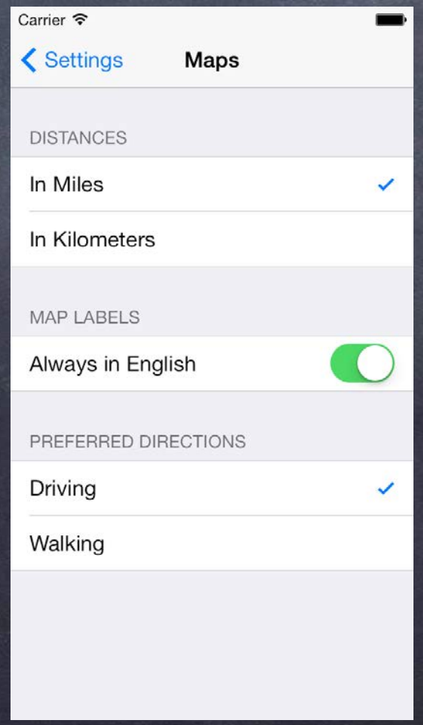
Demo

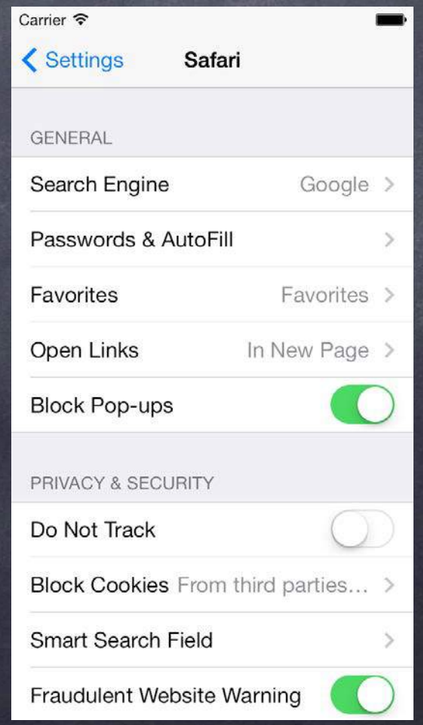
- Photomania

  Let's internationalize it.

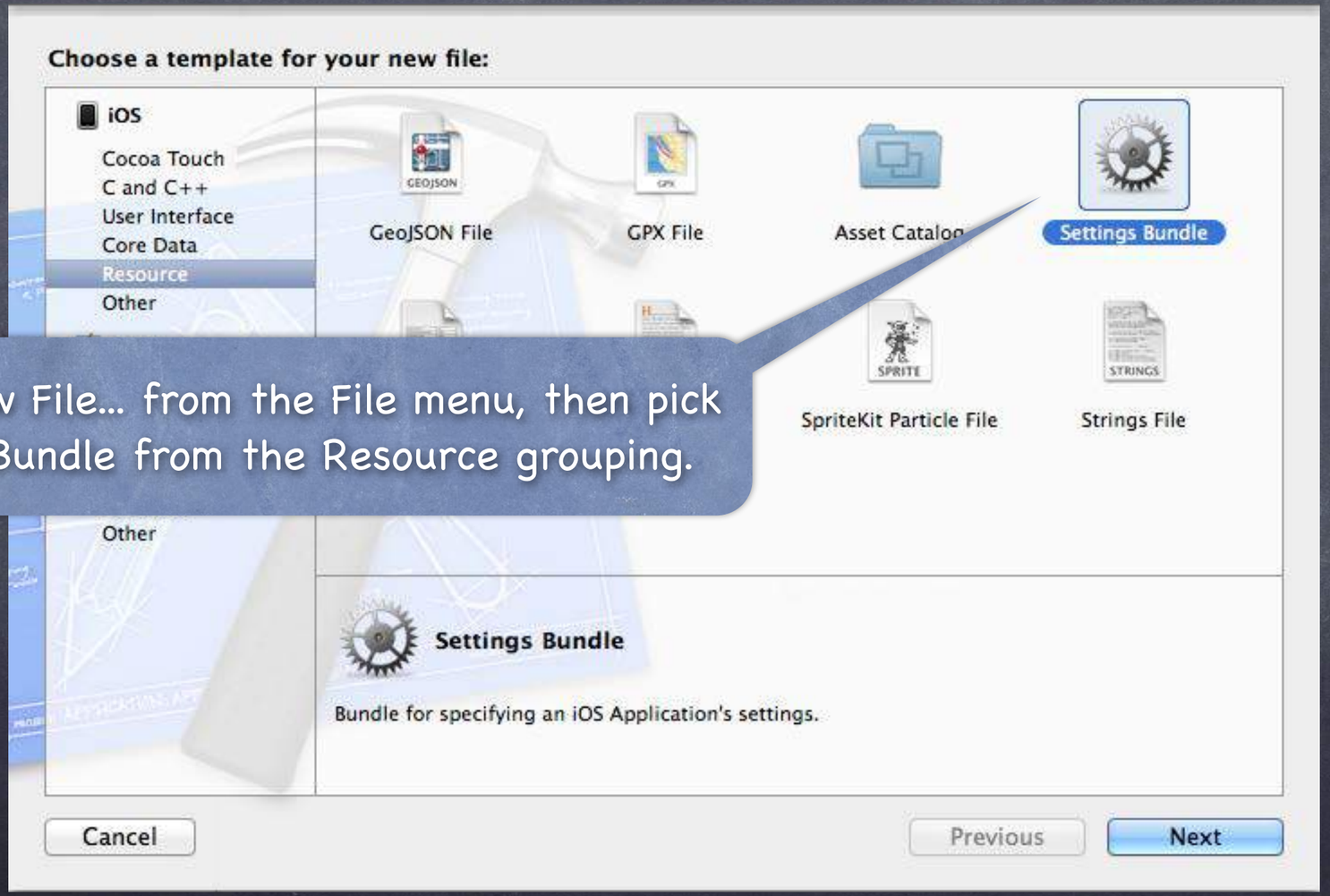
Settings

- A little bit of UI for your application in the Settings application
  - You should use this sparingly (if at all).
  - It's appropriate only for very rarely used settings or default behavior.
  - You don't want to make your users ever have to go here for normal use of your application.
  - The settings appear in your application via `NSUserDefaults`.
  - You specify the UI and the associated defaults in a property list file.





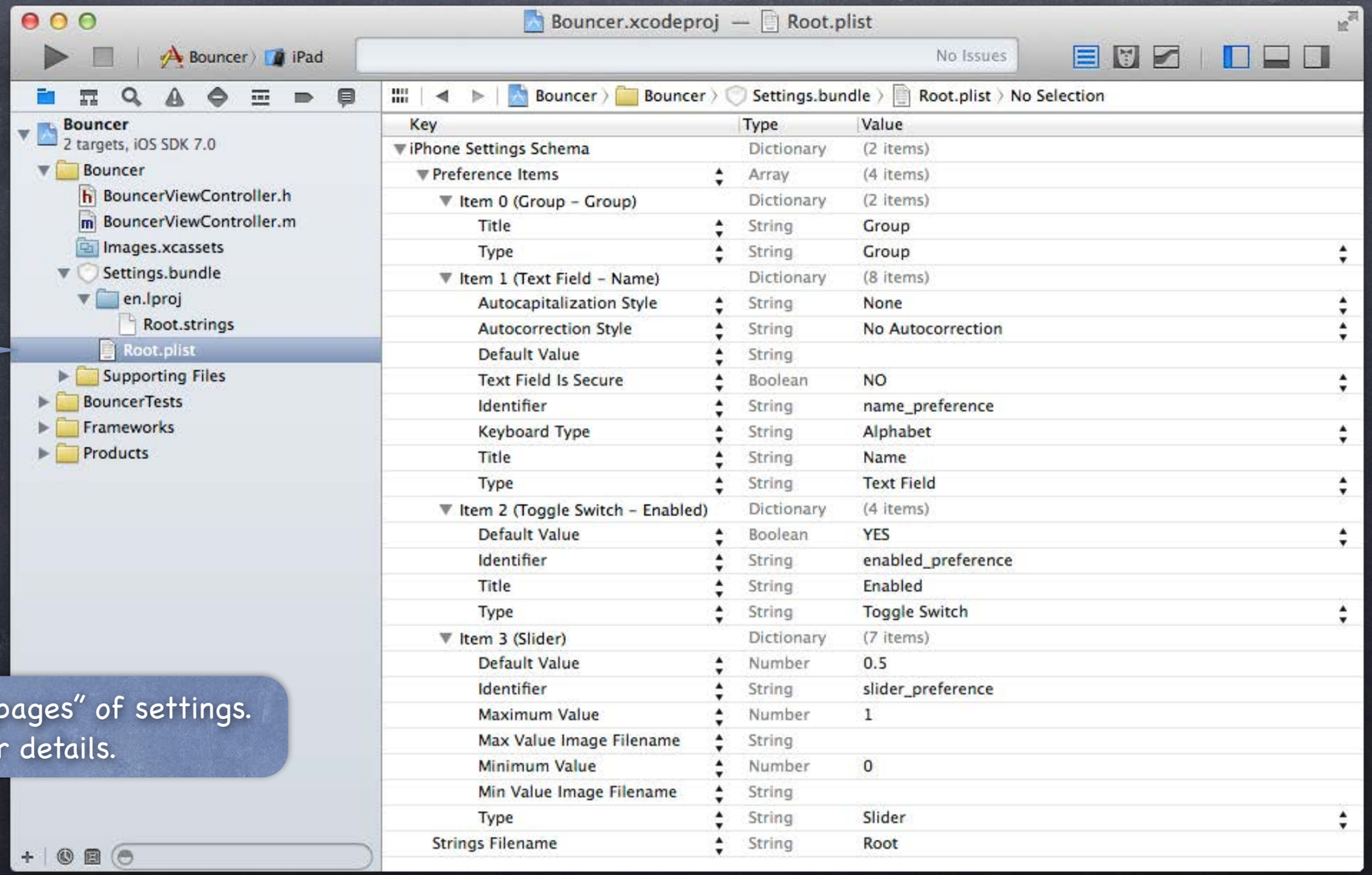
Settings



Choose New File... from the File menu, then pick Settings Bundle from the Resource grouping.

Settings

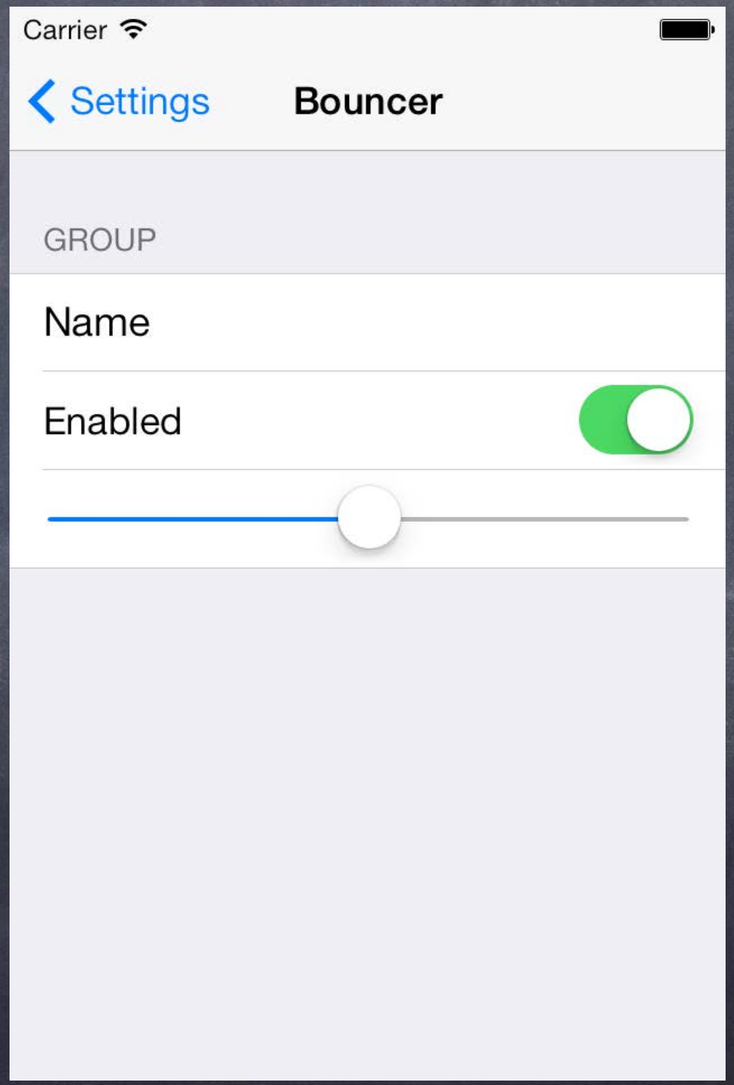
A sort of "example" settings bundle will be created for you. You can edit it by clicking here. Check the documentation for all the possibilities.



It is possible to have multiple "pages" of settings. See documentation for details.

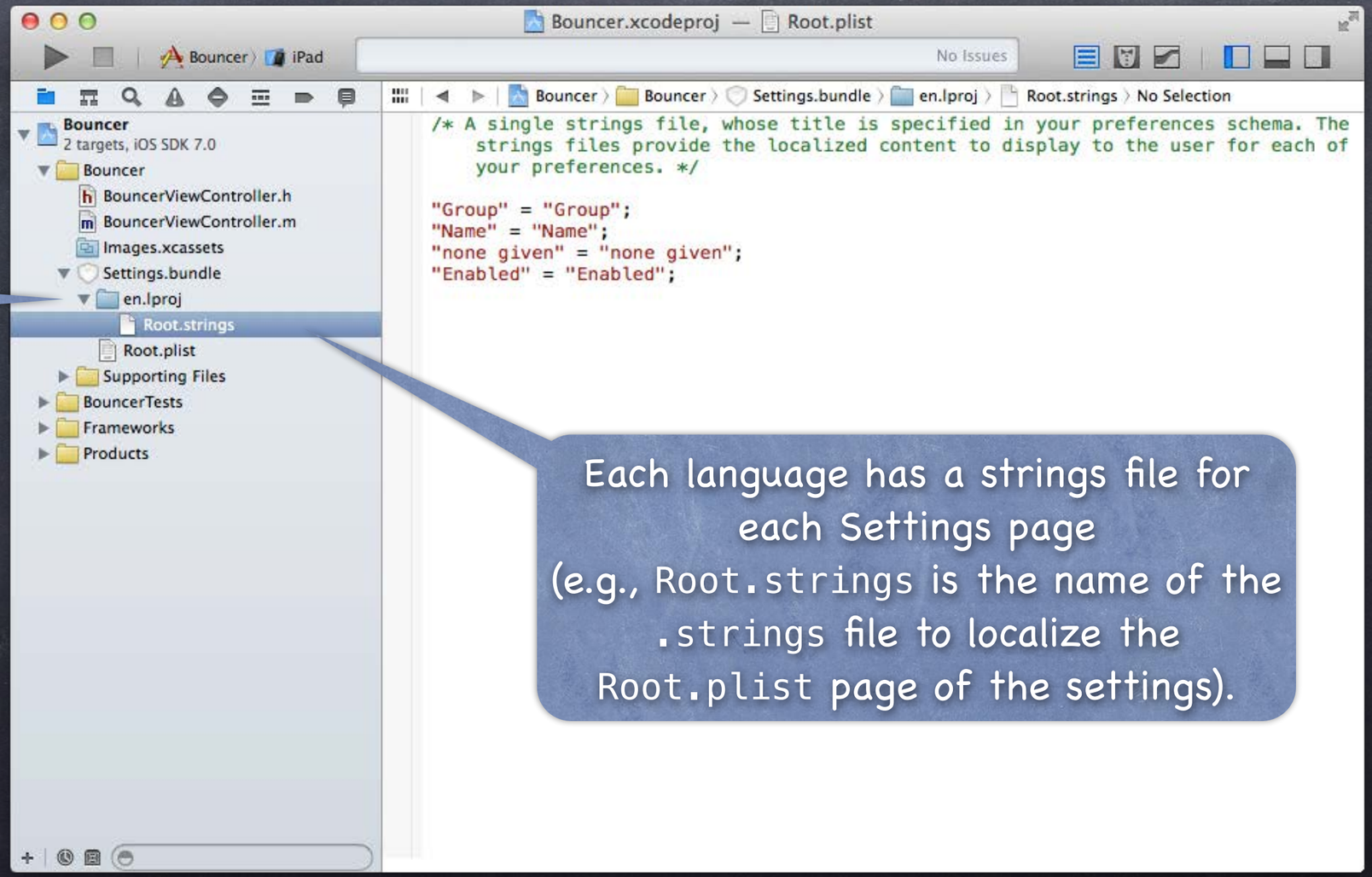
Settings

The sample from the previous slide would result in a Settings UI like this.



Settings

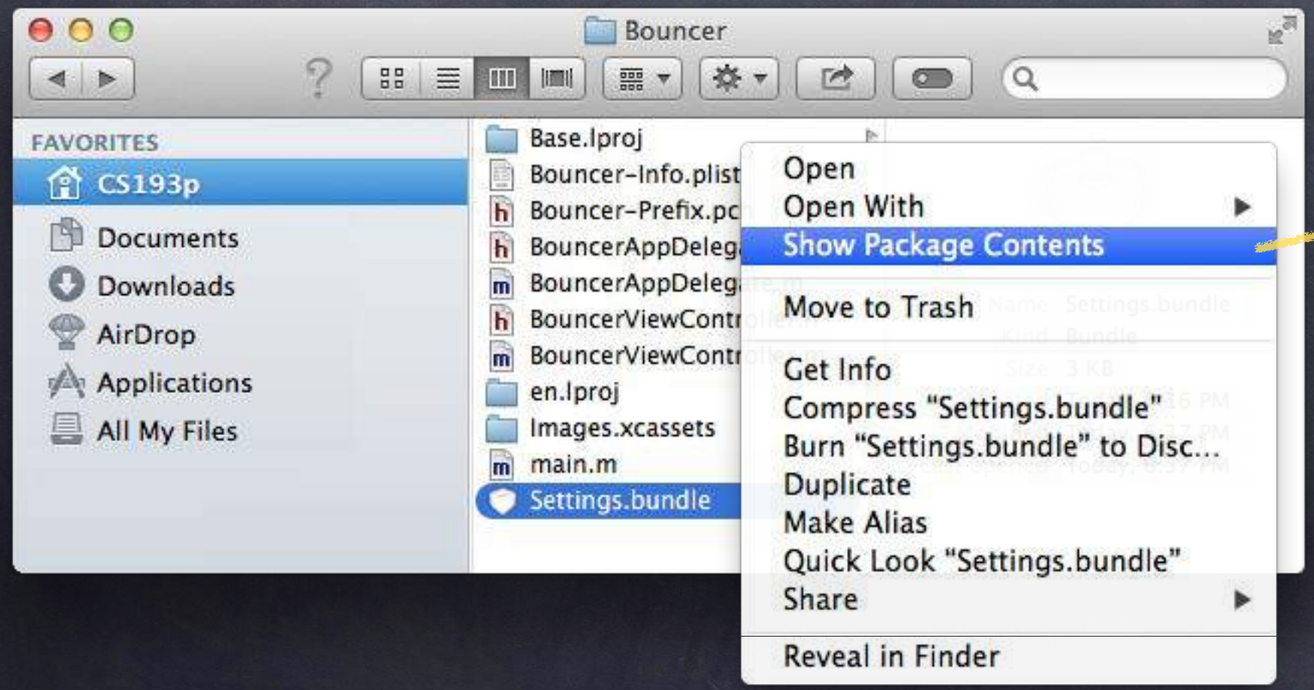
Note the `en.lproj`. Yes, settings are localizable, but it's not very well supported in Xcode.

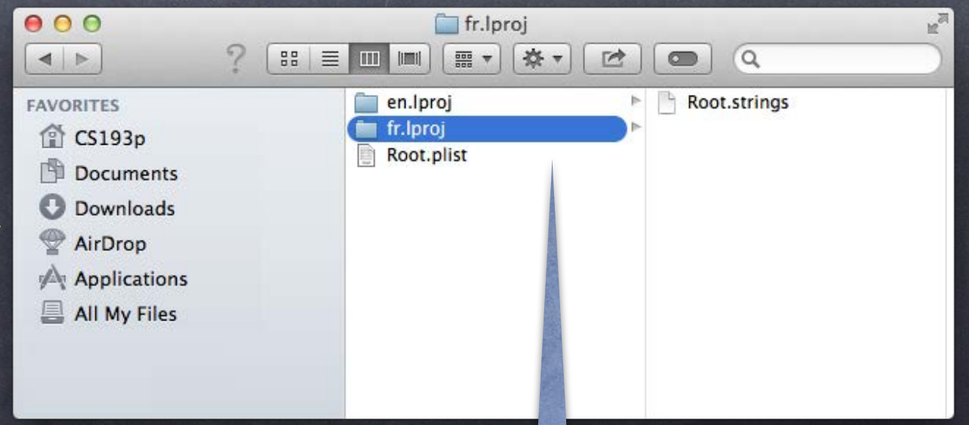


Each language has a strings file for each Settings page (e.g., `Root.strings` is the name of the `.strings` file to localize the `Root.plist` page of the settings).

Settings

- Unfortunately, localization of settings is a bit of a pain
  - You have to find the `Settings.bundle` in your Finder and create `.lproj` directories yourself.
  - Each `.lproj` directory should contain a `.strings` file for each screen in your settings.





Copy and paste `en.lproj` to other languages (like `fr.lproj`), then edit the `Root.strings` (or other `.strings` files) inside for each language.

Demo

- Bouncer

  Allow setting the Elasticity from Settings.

Coming Up

- Wednesday
  - Alternate Final Presentation.
  - If you are using Alternate Presentation time, submit your Keynote by noon tomorrow (Tuesday).
  - Submit the slides using the normal submit script (submit again with code by Sunday).
  - We will have a "live demo testing" opportunity on Wednesday as well, so bring your demo device.
- Friday
  - No Section.
- Sunday
  - Final Project Due (by midnight).
  - Don't forget to submit your Keynote slides along with!
- Final
  - A week from Thursday at 12:15pm to 3:15pm in this room.
  - Presentation is required.
  - Presentation time limit is 2.5 minutes (150 seconds) and must be 1280x720 aspect ratio.
  - Presentation order is random (no exceptions).