

# Stanford CS193p

Developing Applications for iOS

Fall 2011



# Today

## ⌚ Demo continuation

Delegating the View's data (its smileyness)

Adding a gesture recognizer to the View (and handled by the Controller) that modifies the Model

## ⌚ View Controllers

Multiple MVCs

Segues

UINavigationController

Demo

## ⌚ Friday

Getting your application running on a device

# View Controller

- ➊ Hopefully you've got a pretty good handle on the basics of this!

Your Controller in an MVC grouping is always a subclass of `UIViewController`.

It manages a View (made up of subviews that you usually have some outlets/actions to/from).

It is the liaison between that View and the Model (which is UI-independent).

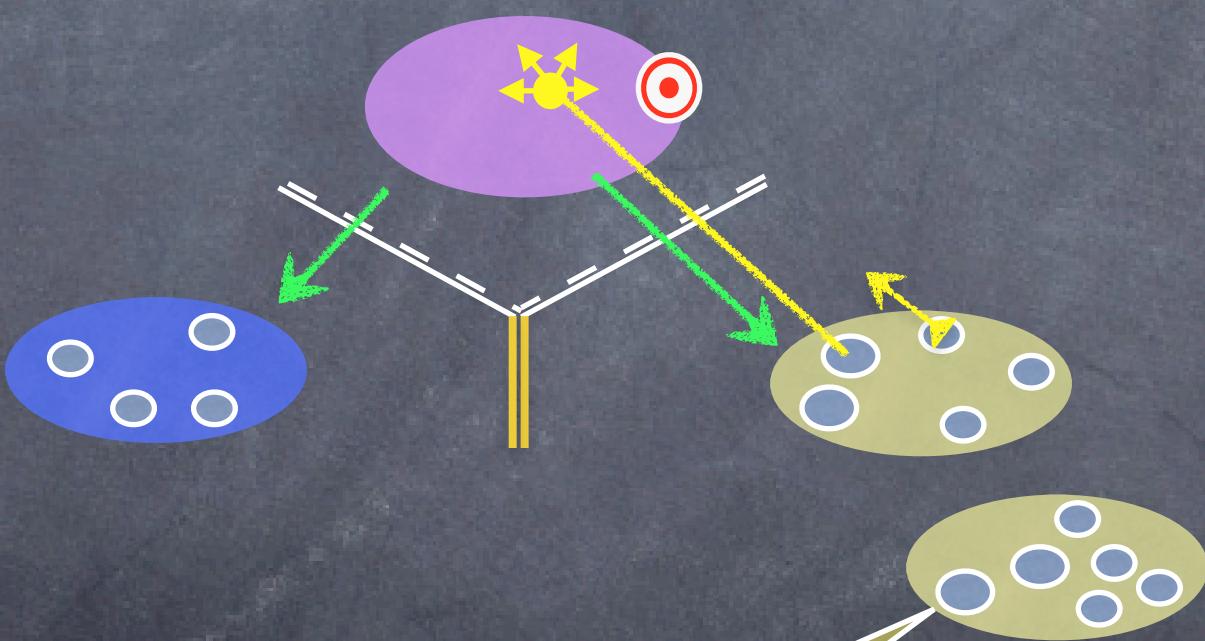
- ➋ So how do we grow our application to use multiple MVCs?

We need infrastructure to manage them all.

That's what storyboards and "controllers of controllers" are all about.

# MVCs working together

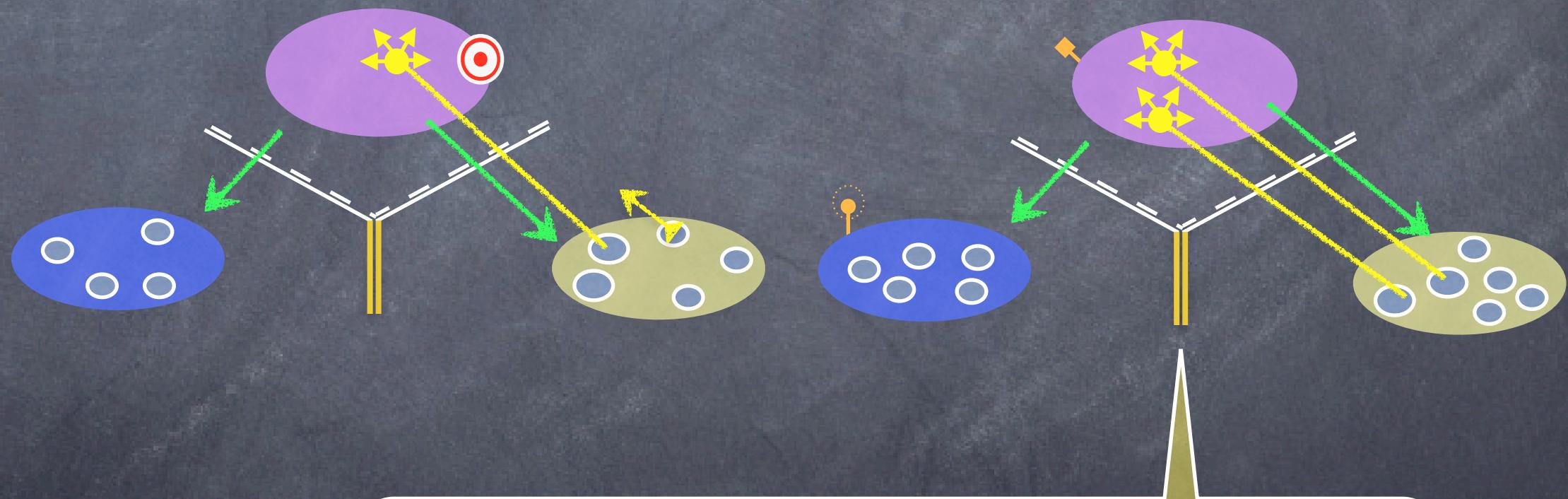
What happens when your application gets more features?



Now all of your UI can't fit in one MVC's view.

# MVCs working together

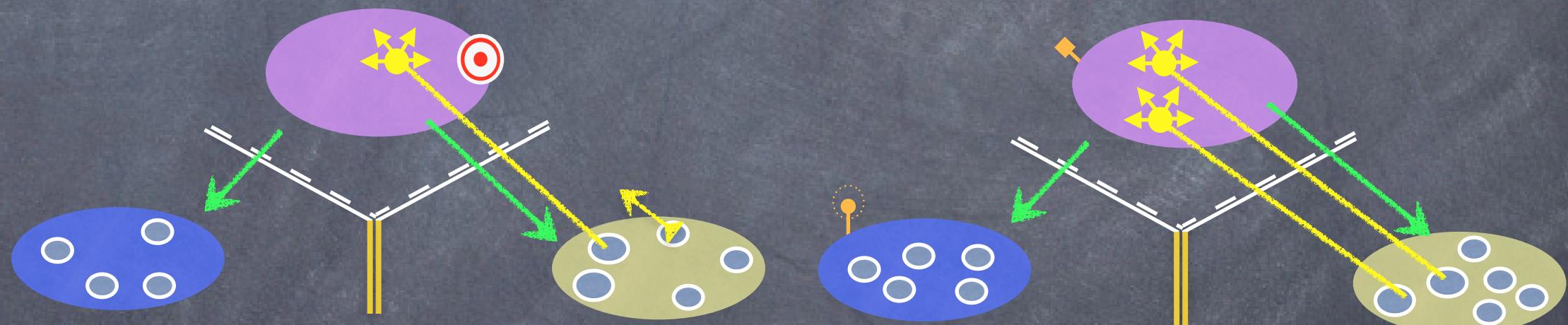
What happens when your application gets more features?



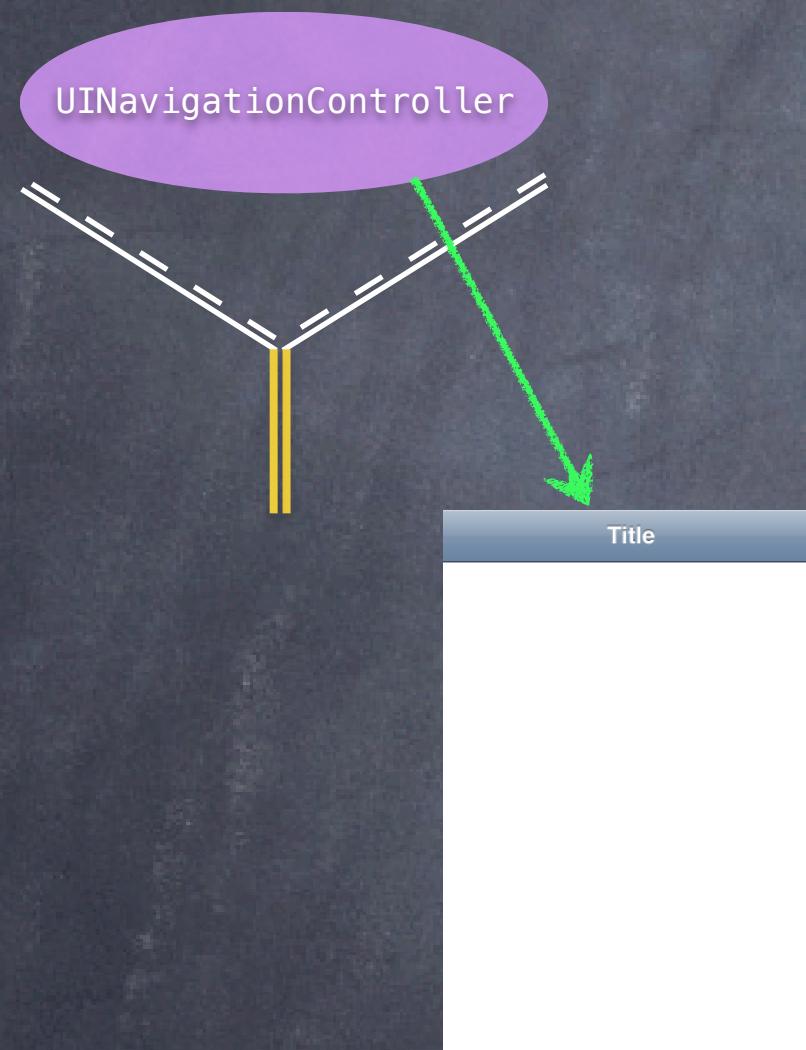
We never have an MVC's view span across screens.  
So we'll have to create a new MVC for these new features.

# MVCs working together

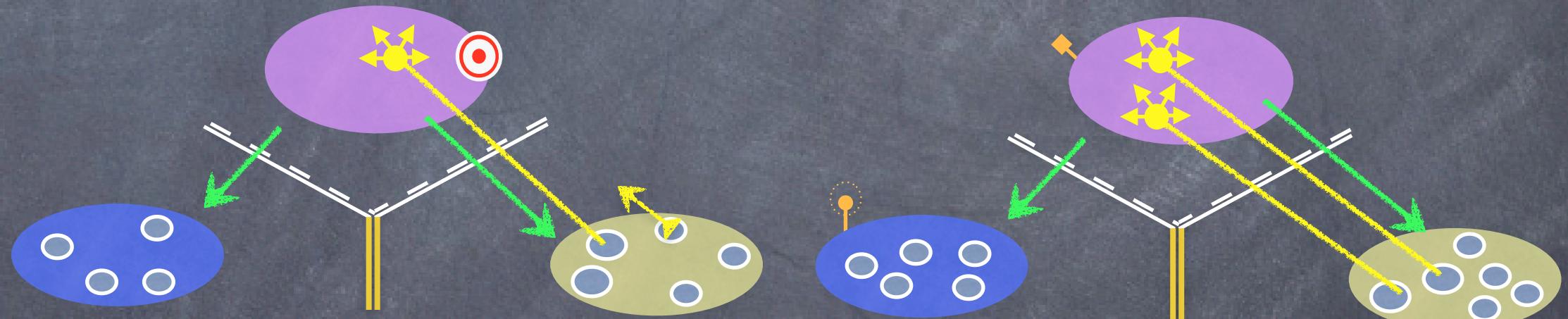
So how do we switch the screen to show this other MVC?



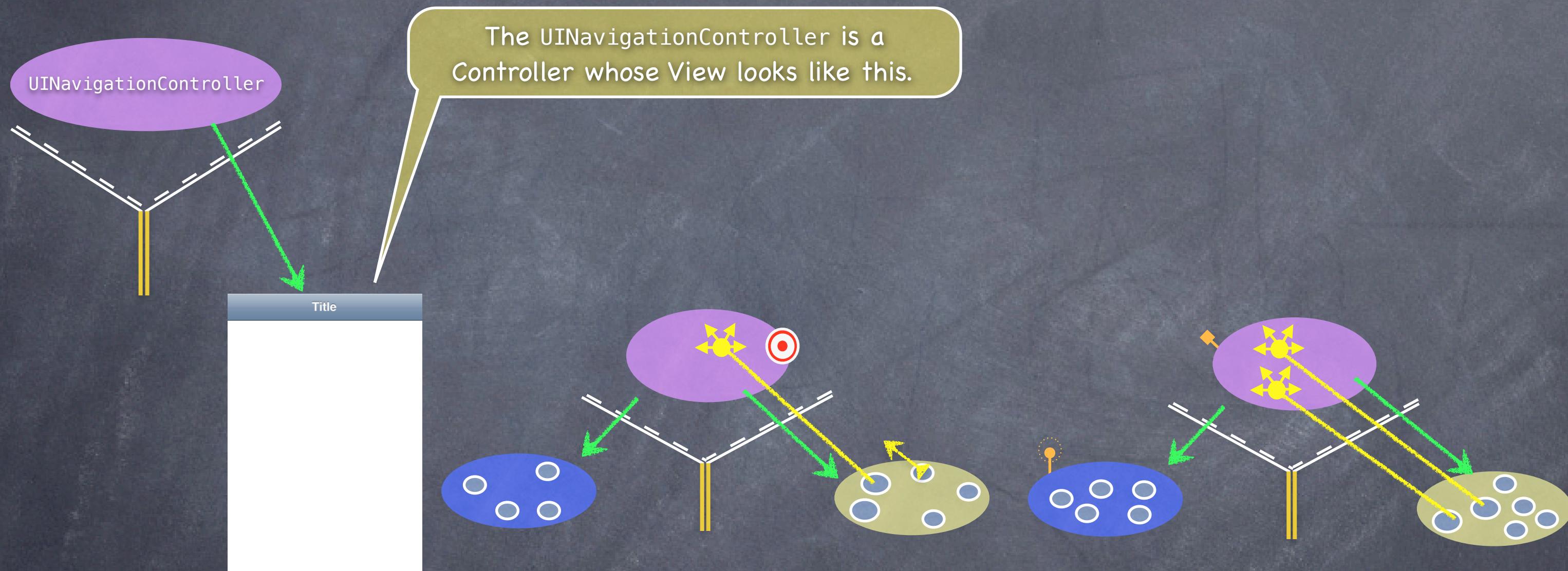
# MVCs working together



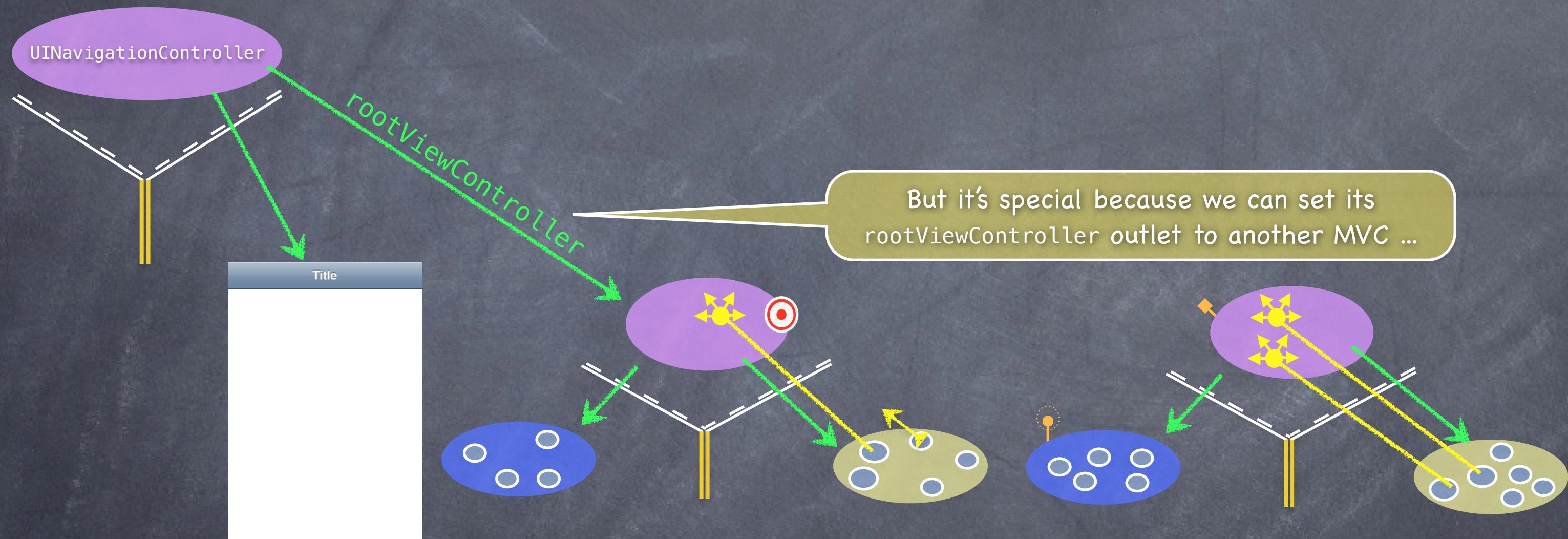
We use a “controller of controllers” to do that.  
For example, a **UINavigationController**.



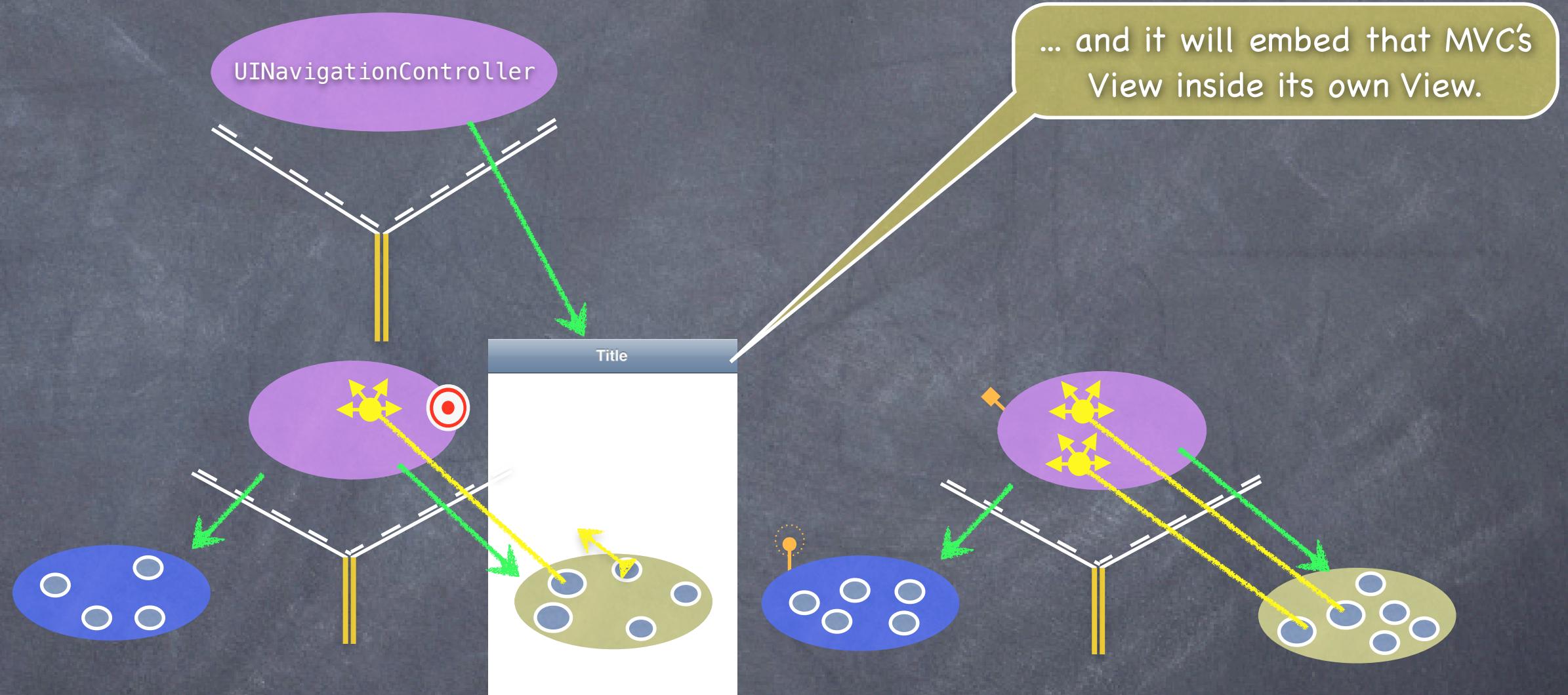
# MVCs working together



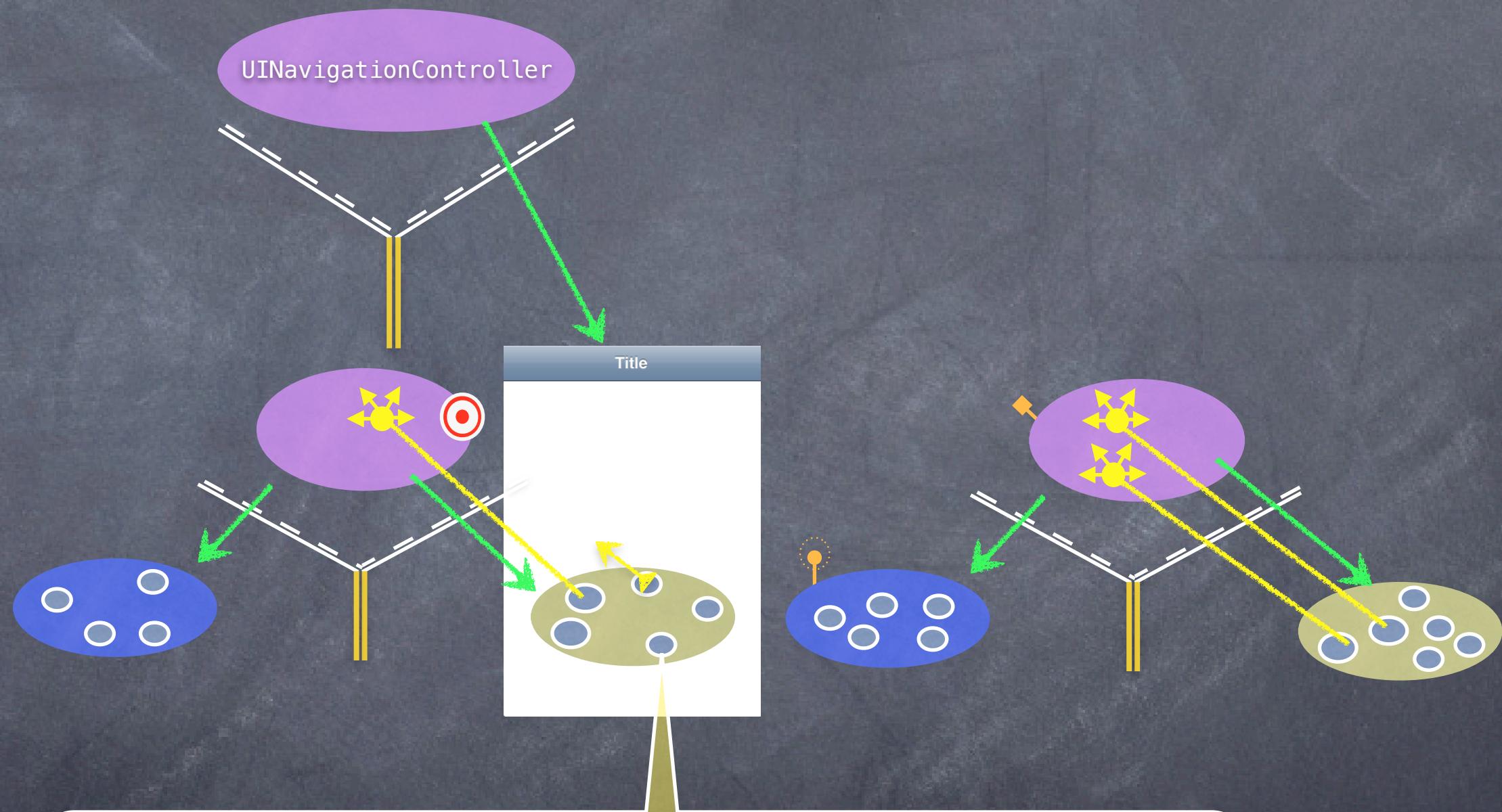
# MVCs working together



# MVCs working together

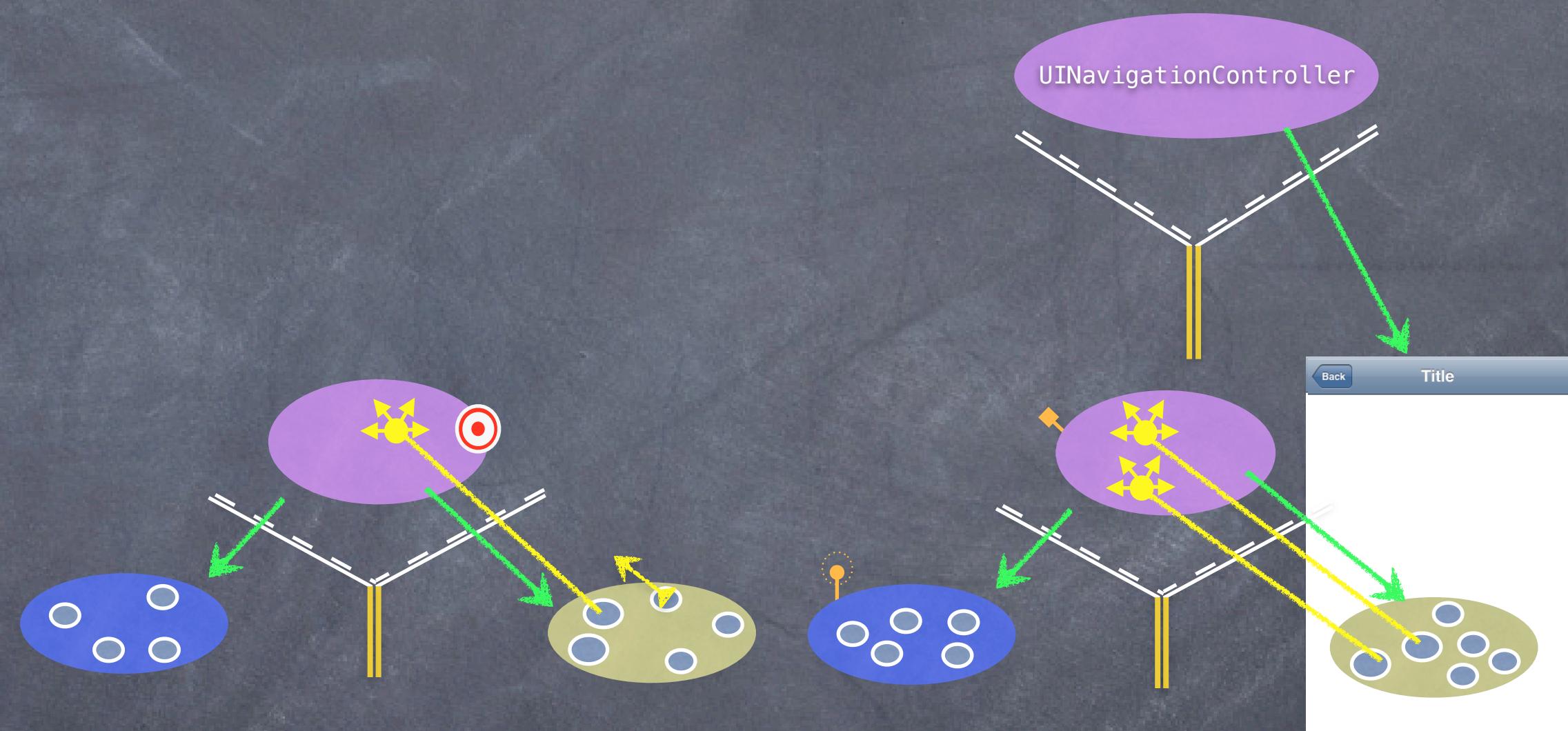


# MVCs working together

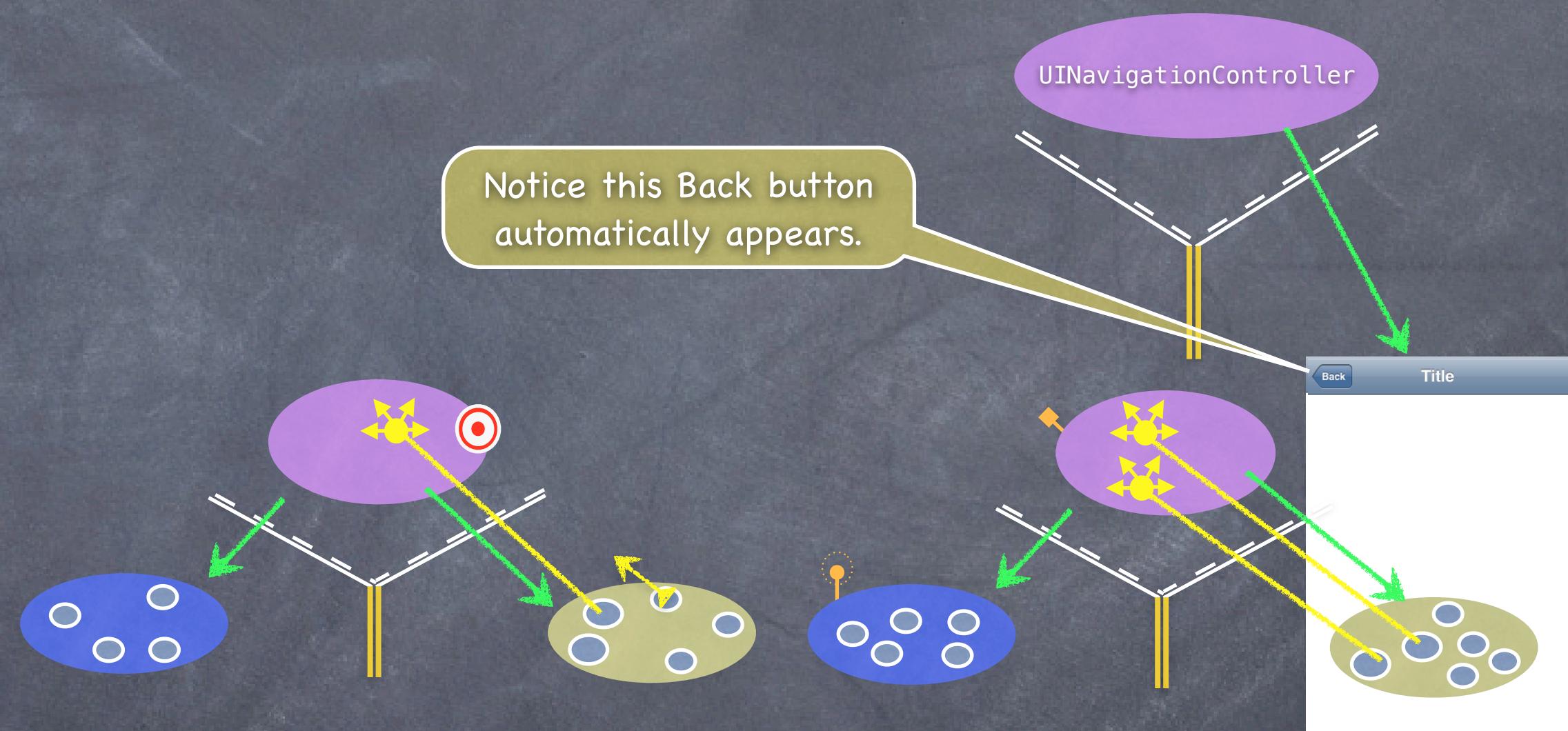


Then a UI element in this View (e.g. a UIButton) can segue to the other MVC and its View will now appear in the UINavigationController.

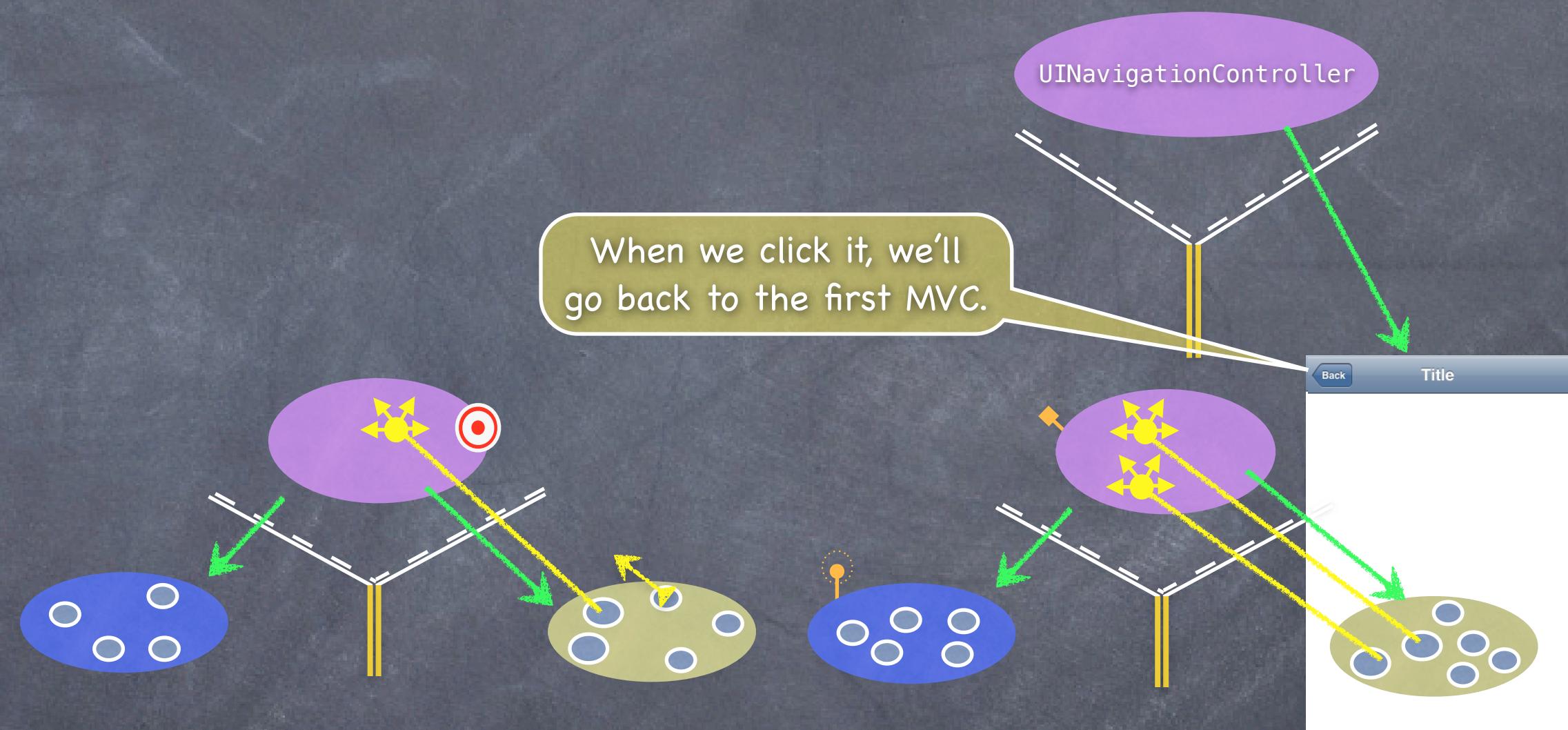
# MVCs working together



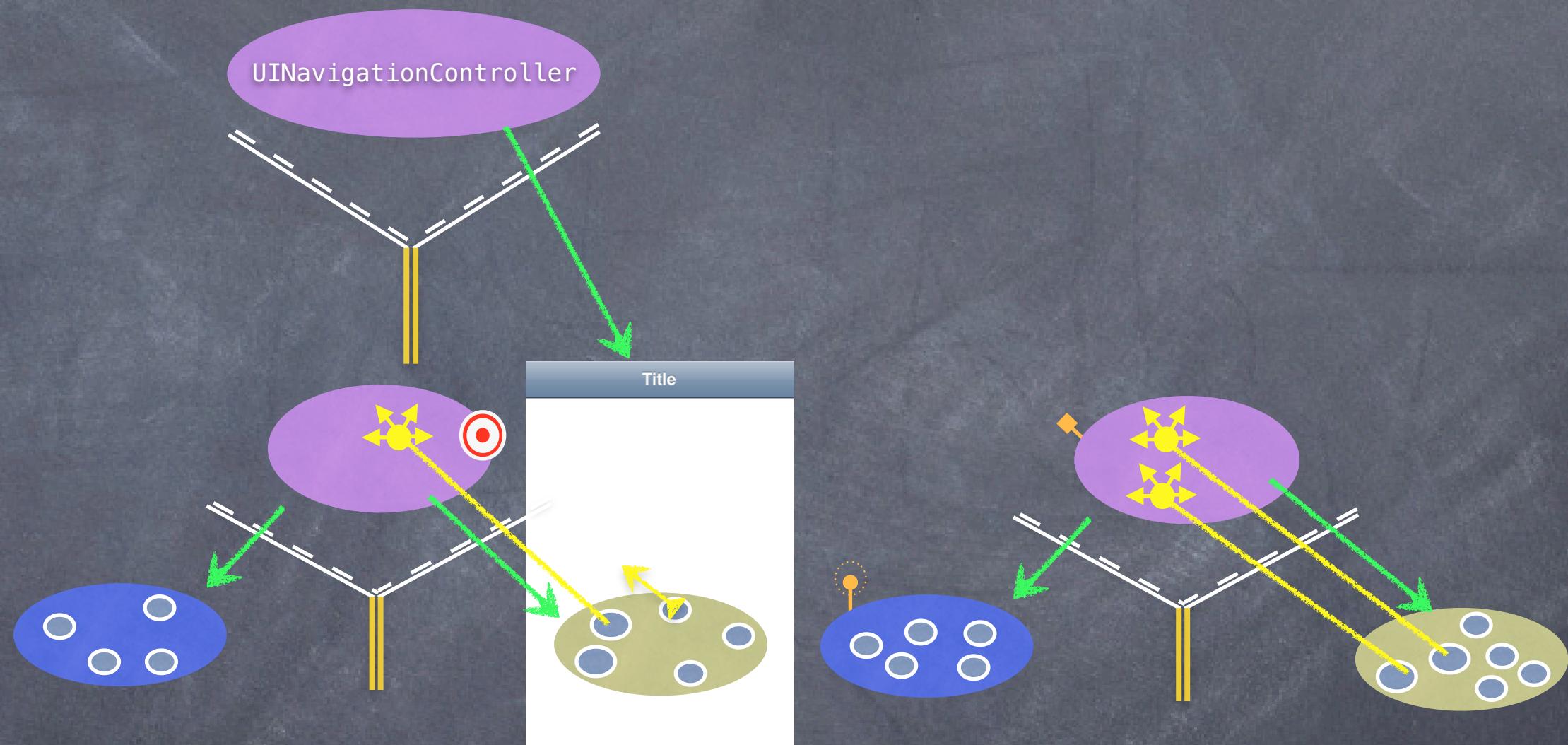
# MVCs working together



# MVCs working together



# MVCs working together



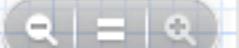
# Segues

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

To create a segue, you hold down ctrl and drag from the button to the other View Controller.

Show Other View Controller

View Controller



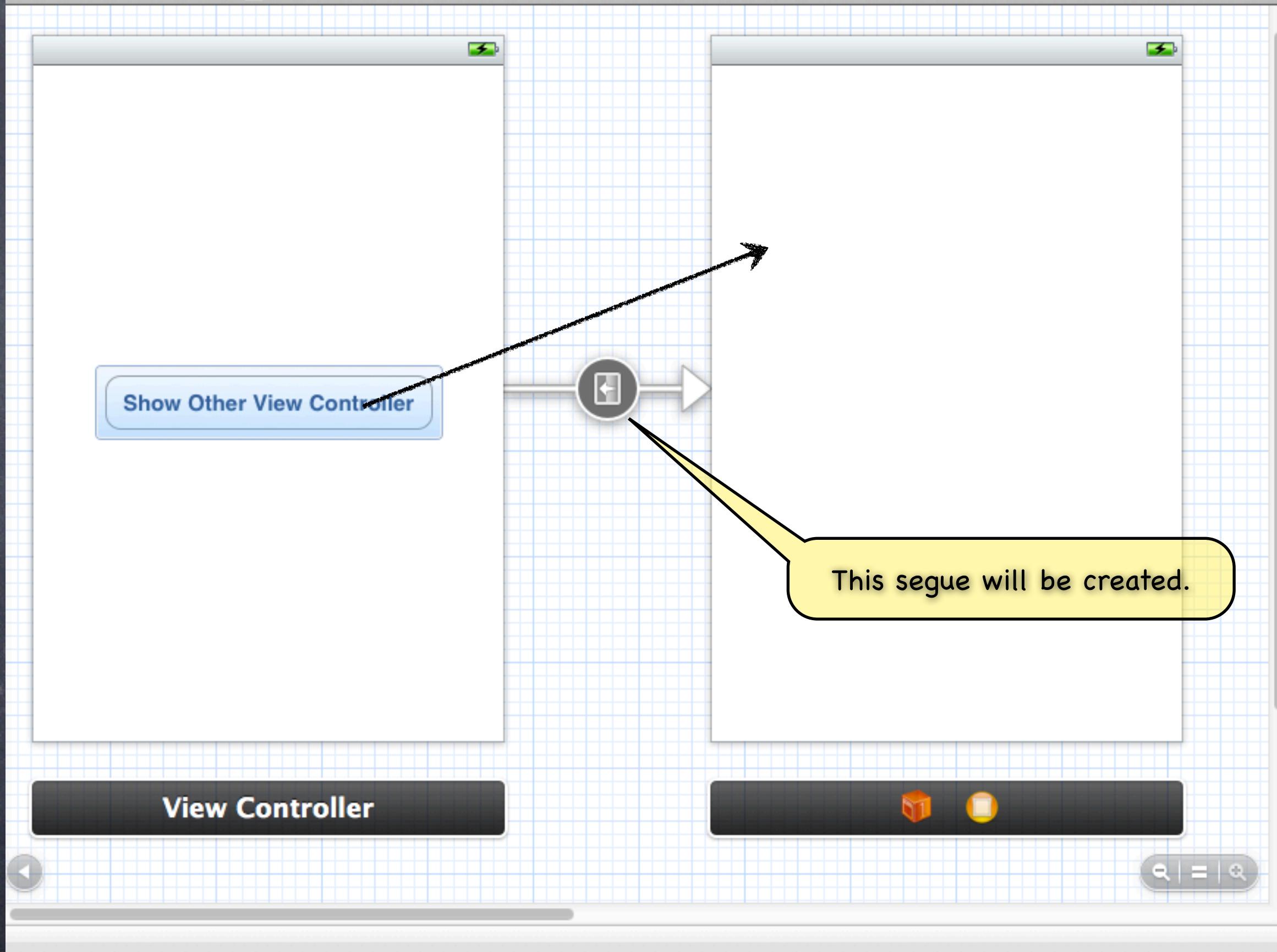
To create a segue, you hold down ctrl and drag from the button to the other View Controller.

Show Other View Controller



View Controller





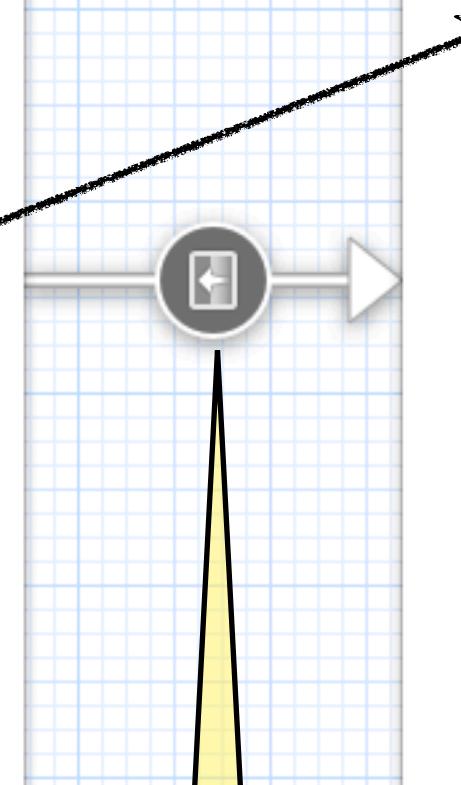
## Storyboard Segue

Identifier ShowOther

Style Push

This is the identifier for this segue ("ShowOther" in this case). You use it in `prepareForSegue:sender:` to figure out which segue is happening. Or you can use it to programmatically force a segue with `performSegueWithIdentifier:sender:`.

Show Other View Controller

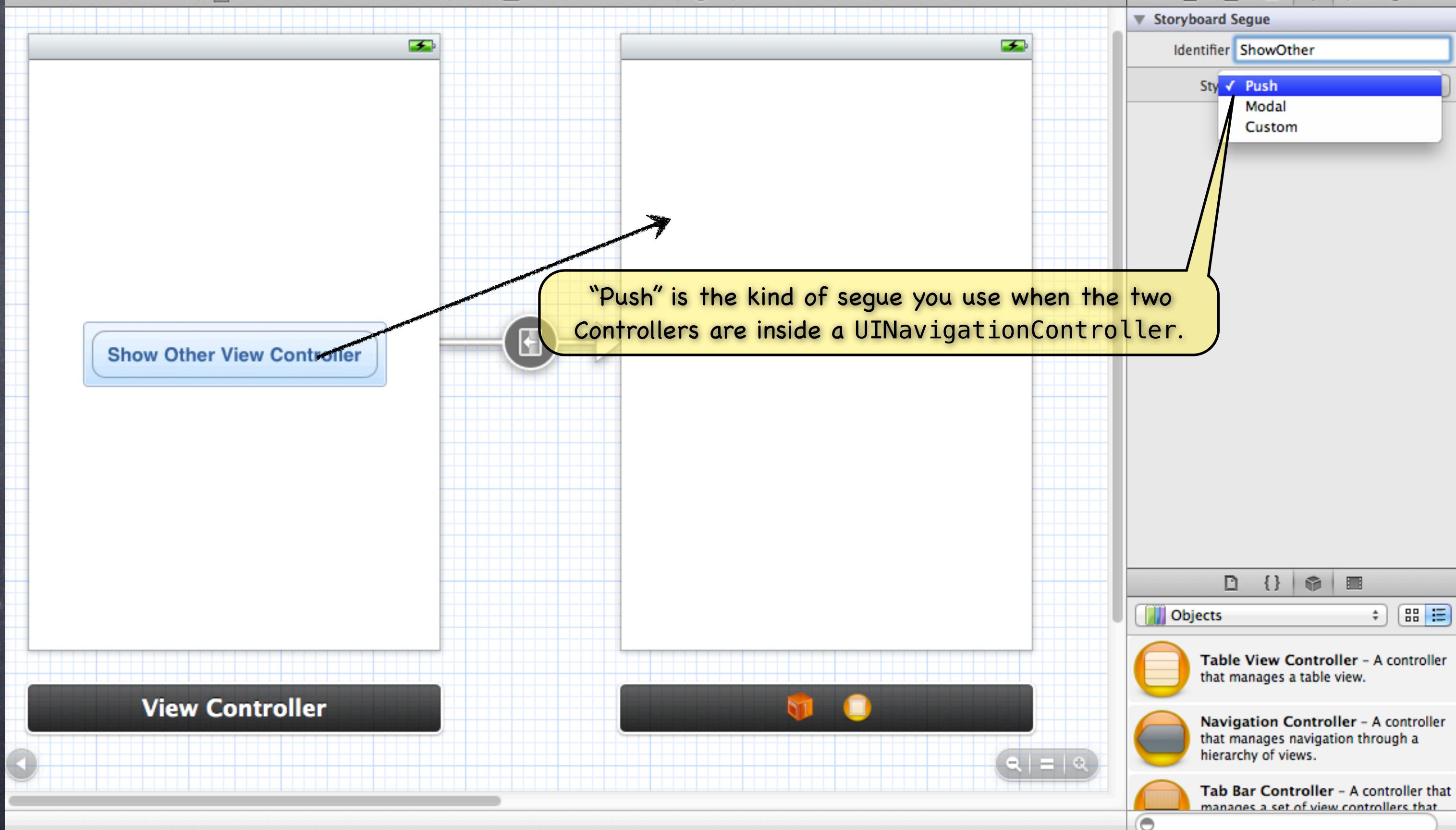


You can change the segue type by clicking on the segue and looking in the attributes inspector.

View Controller



Objects	
	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



The screenshot shows the Xcode storyboard editor. A segue is being drawn from a blue rounded rectangular button labeled "Show Other View Controller" to a white view controller. The segue is currently set to "Push" in the storyboard segue inspector, which is indicated by a blue selection bar. A callout bubble with a yellow background and black border appears over the segue, containing the text: "But there's a problem here. These View Controllers are not inside a UINavigationController. Push will do nothing." The storyboard also includes a "View Controller" object at the bottom left and a "Table View Controller" object at the bottom right. The right side of the interface shows the storyboard sidebar with sections for "Storyboard Segue", "Objects", and "Library".

Identifier ShowOther

Style  Push

Modal

Custom

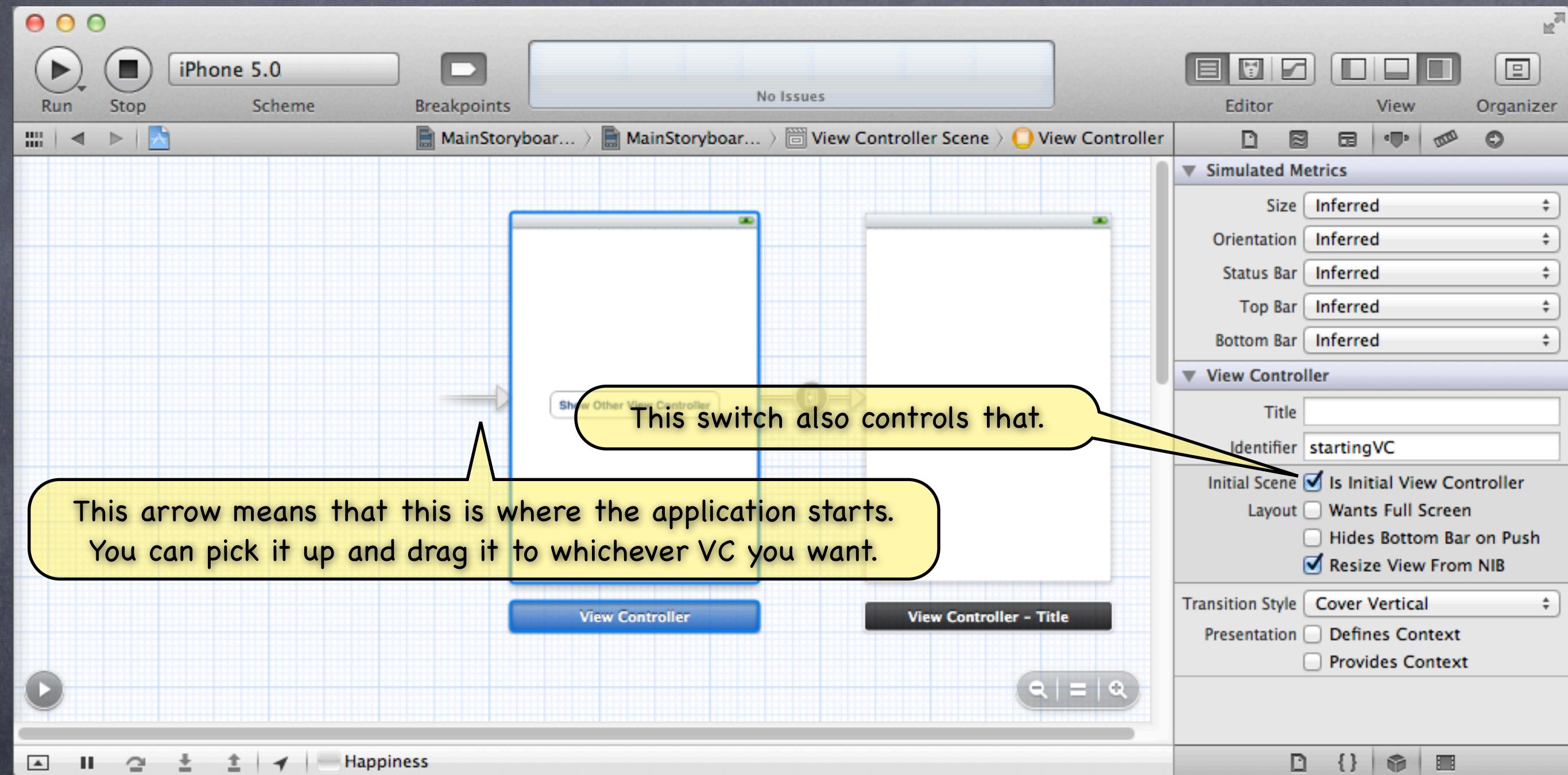
But there's a problem here.  
These View Controllers are not inside a UINavigationController.  
Push will do nothing.

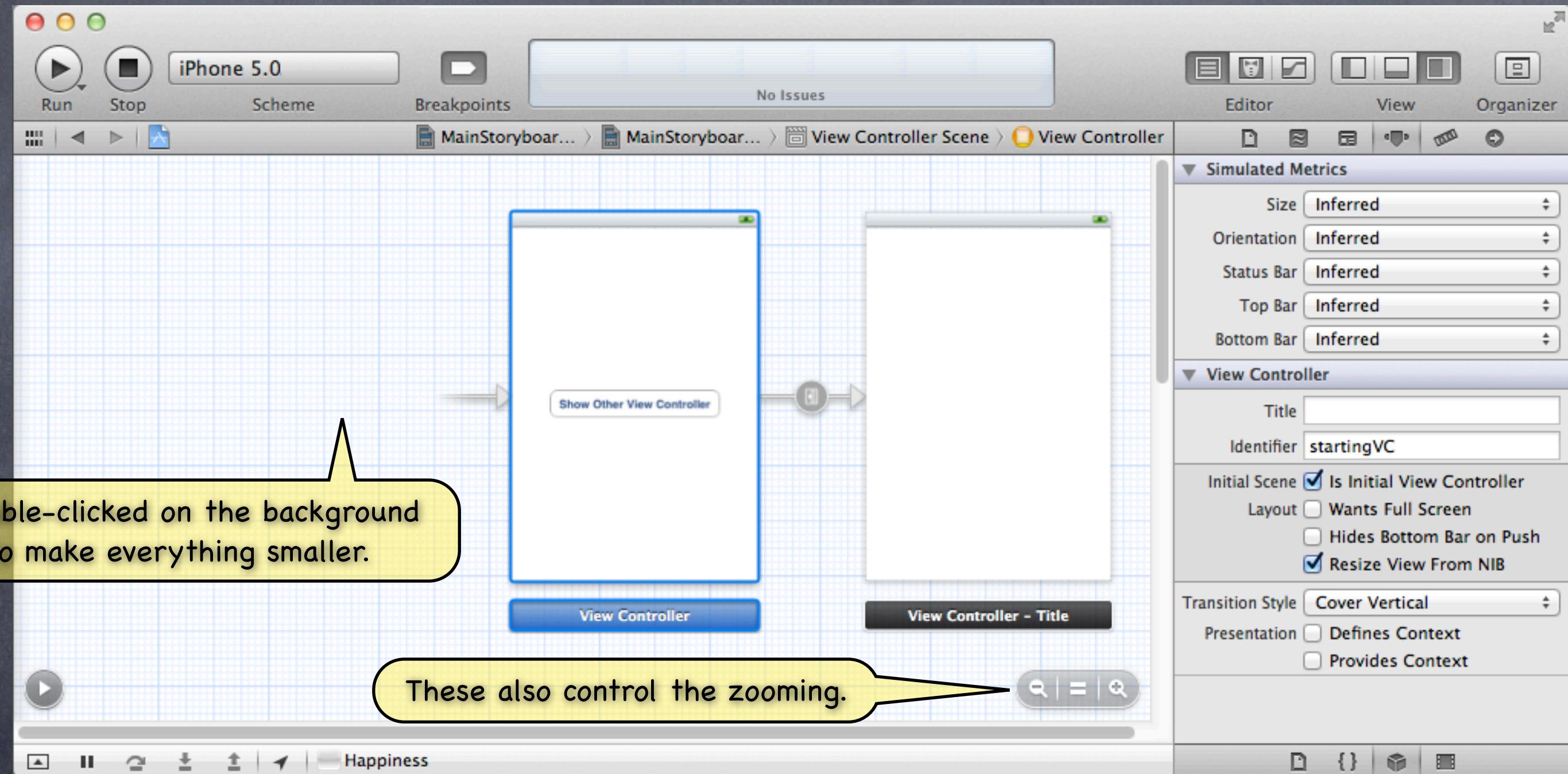
View Controller

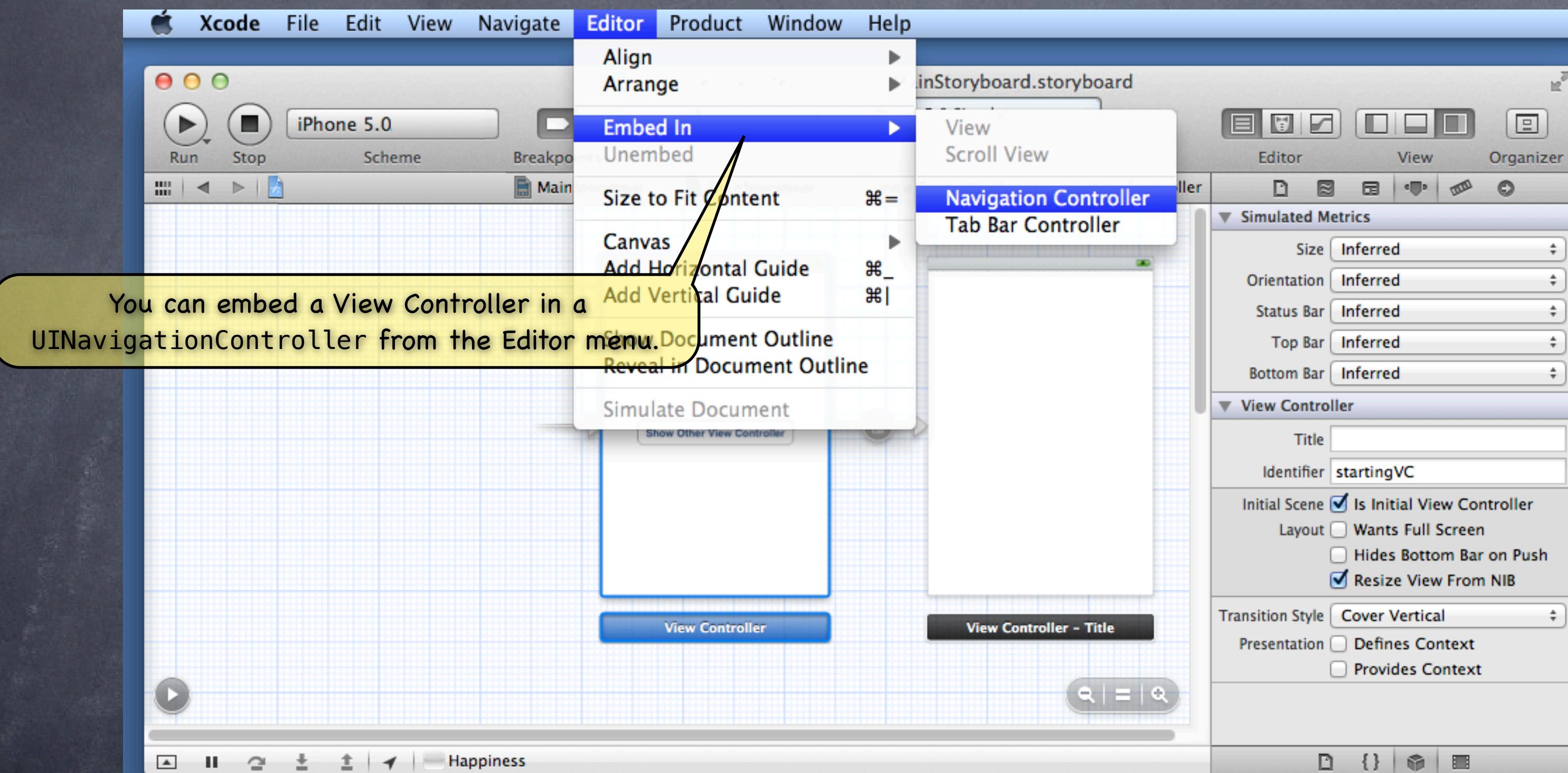
Table View Controller - A controller that manages a table view.

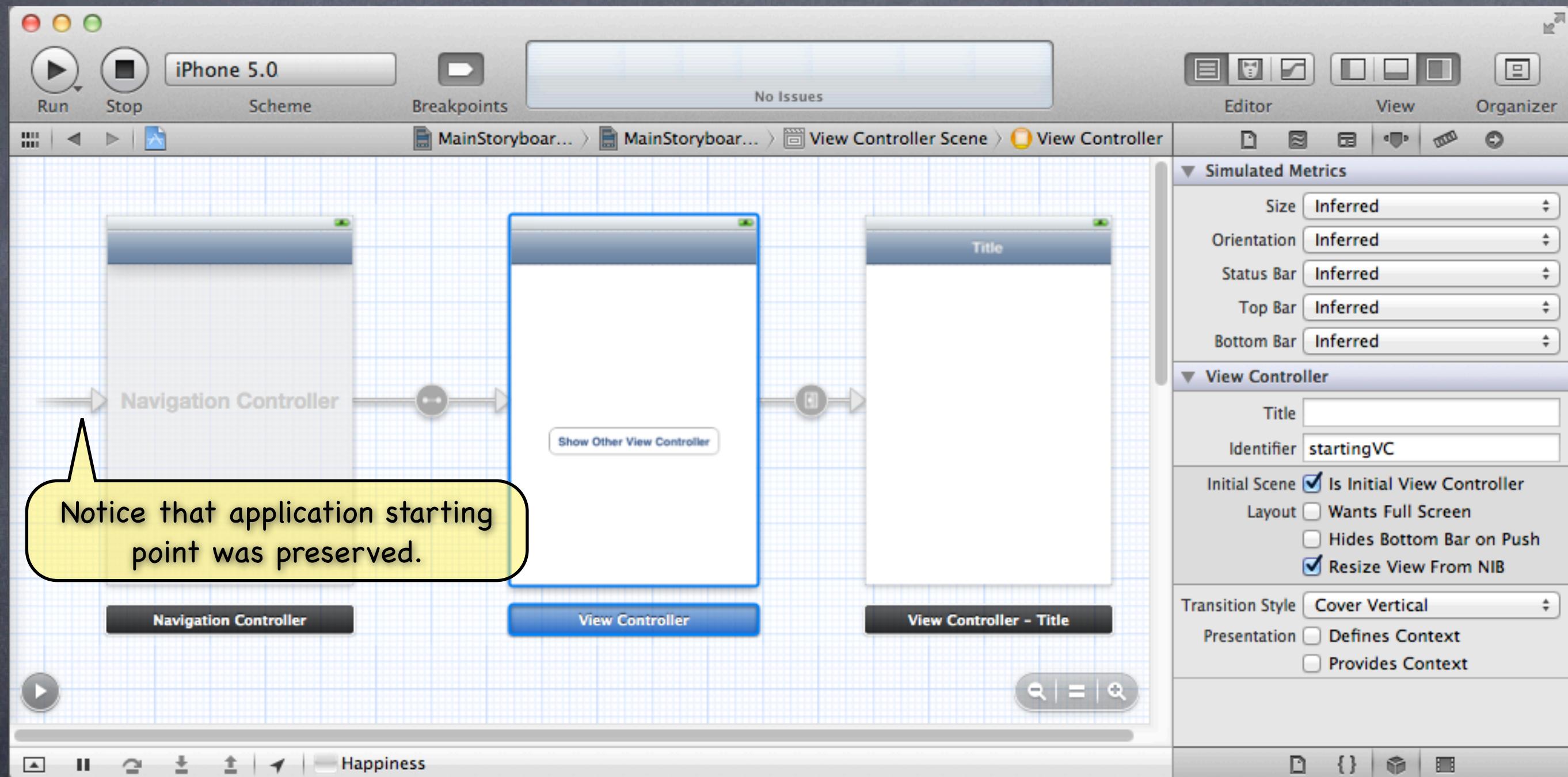
Navigation Controller - A controller that manages navigation through a hierarchy of views.

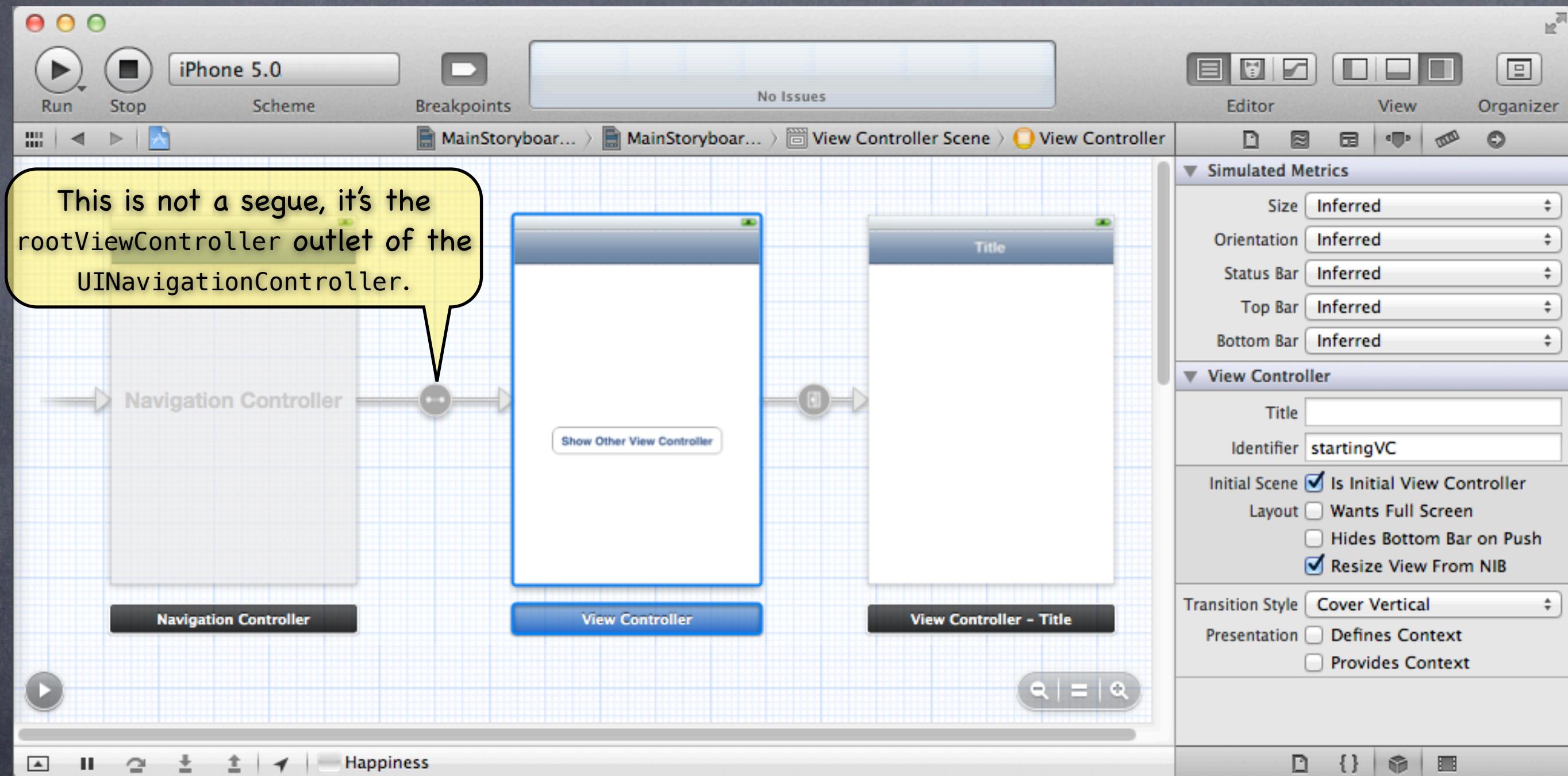
Tab Bar Controller - A controller that manages a tab bar and a set of view controllers that are presented simultaneously.



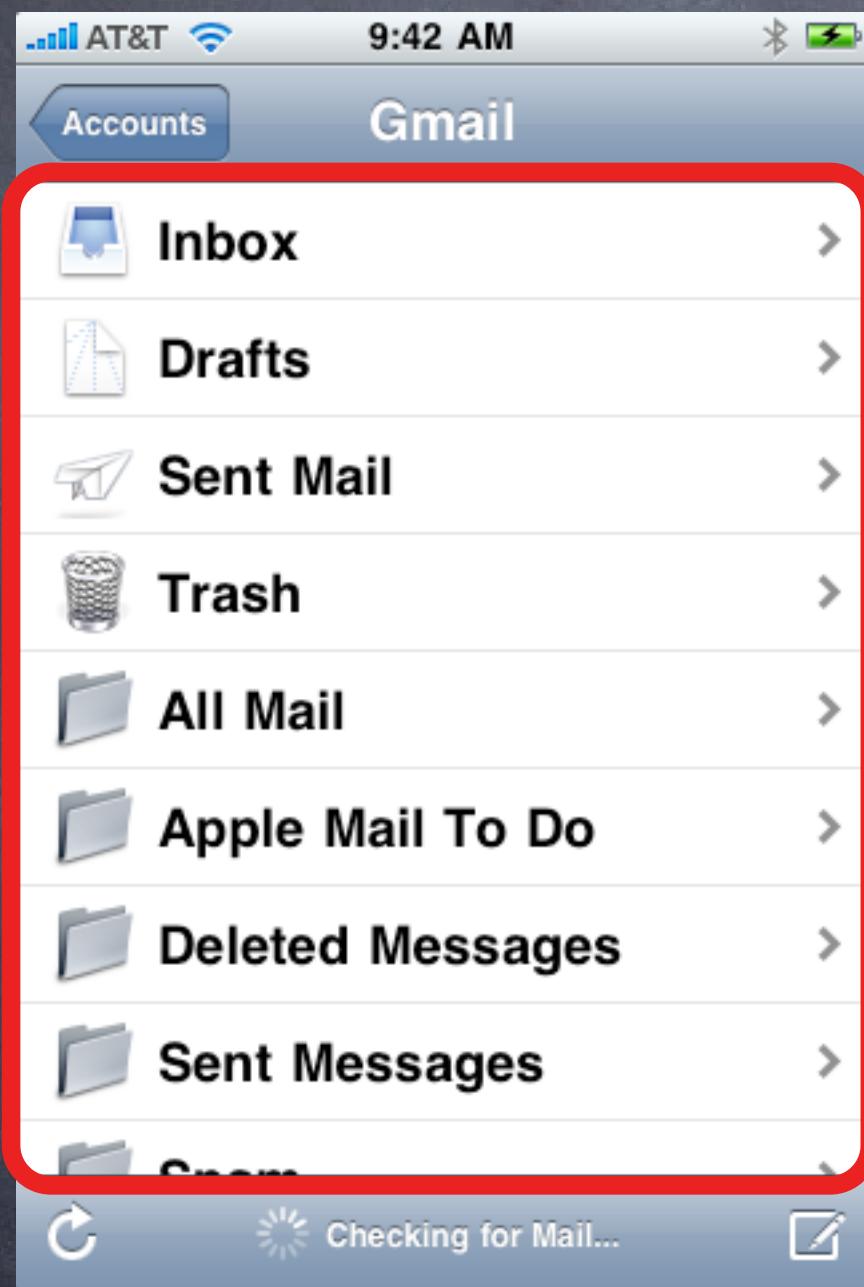






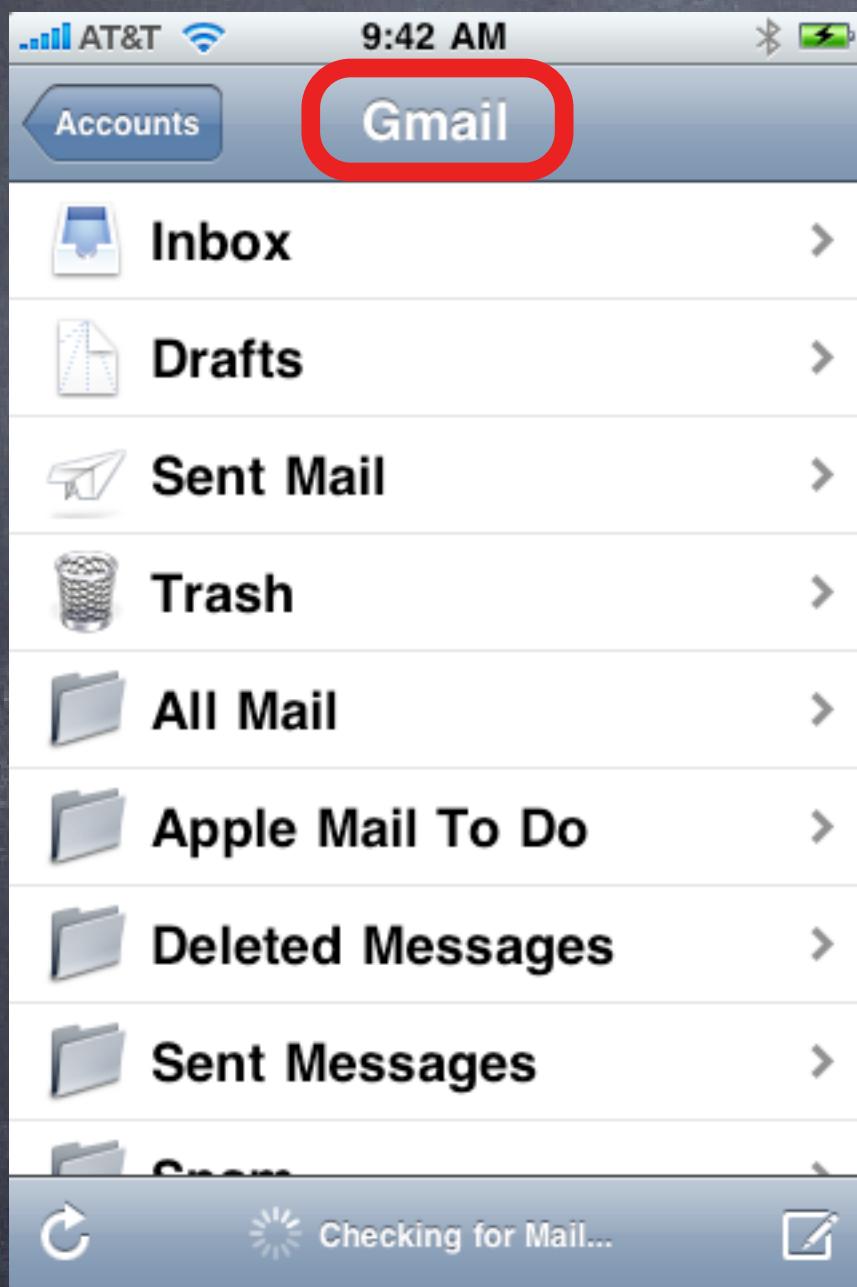


# UINavigationController



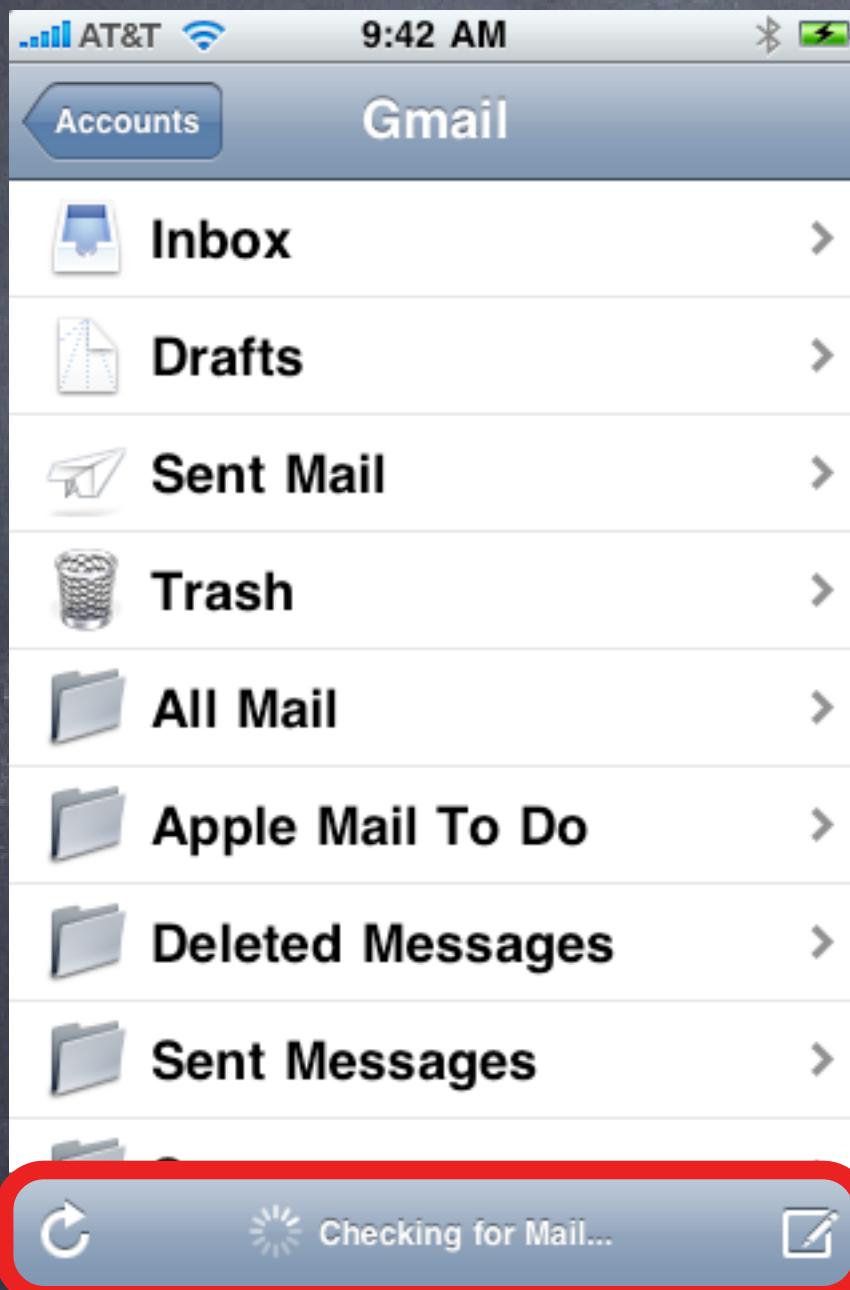
- **UIView** obtained from the **view** property of the **UIViewController** most recently pushed (or root)

# UINavigationController



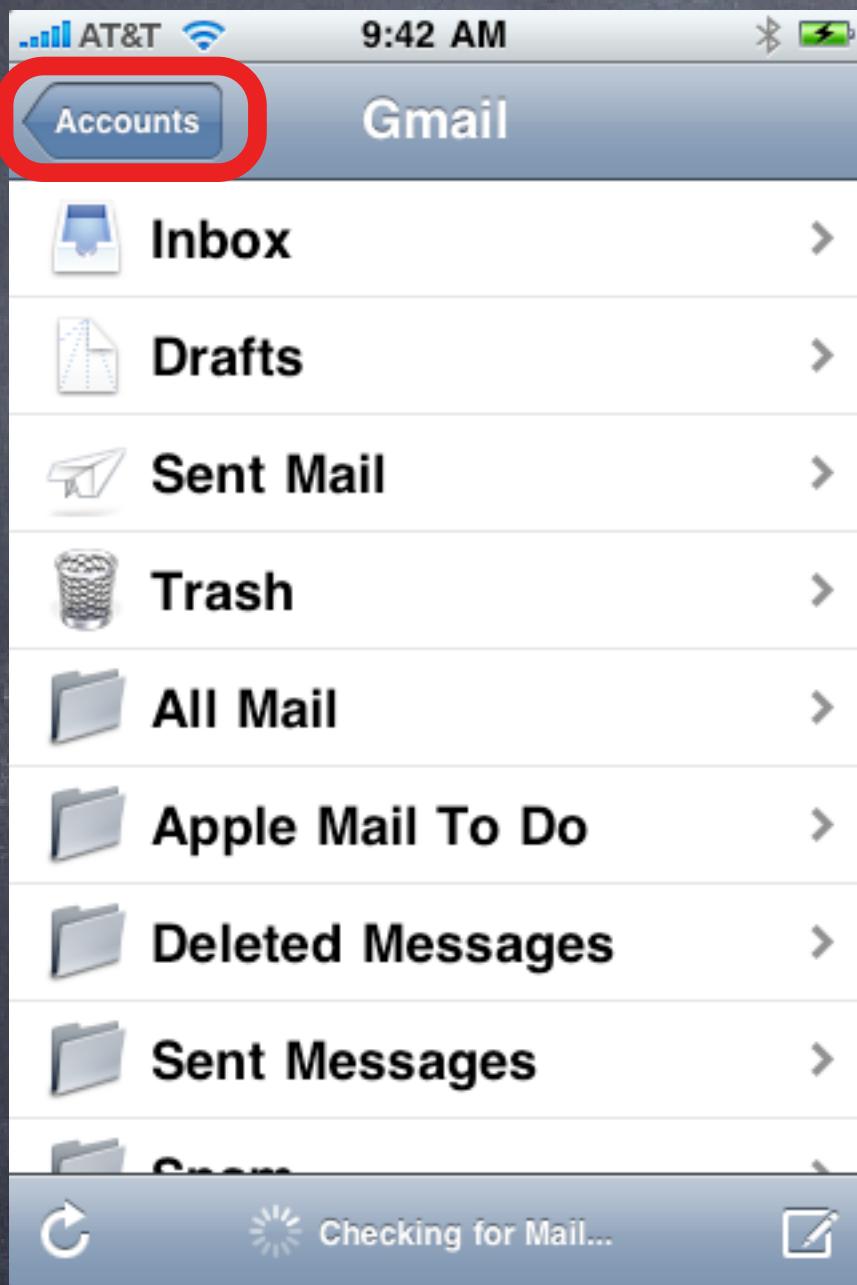
- `UIView` obtained from the `view` property of the `UIViewController` most recently pushed (or root)
- `NSString` obtained from the `title` property of the `UIViewController` most recently pushed (or root)

# UINavigationController



- `UIView` obtained from the `view` property of the `UIViewController` most recently pushed (or root)
- `NSString` obtained from the `title` property of the `UIViewController` most recently pushed (or root)
- An `NSArray` of `UIBarButtonItem`s obtained from the `toolbarItems` property of the `UIViewController` most recently pushed (or root)

# UINavigationController



- `UIView` obtained from the `view` property of the `UIViewController` most recently pushed (or root)
- `NSString` obtained from the `title` property of the `UIViewController` most recently pushed (or root)
- An `NSArray` of `UIBarButtonItem`s obtained from the `toolbarItems` property of the `UIViewController` most recently pushed (or root)
- A `UIBarButtonItem` item whose `title` is an `NSString` obtained from the `title` property of the previous `UIViewController` that was pushed. It is being displayed on a button provided by the navigation controller which, when touched, will cause the previous `UIViewController` to reappear. This is a "back" button.

# 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 on Tuesday

Replace & Popover

- ⦿ We'll talk about Modal later in the quarter

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:@"DoAParticularThing"]) {
        UIViewController *newController = segue.destinationViewController;
        // send messages to newController to prepare it to appear on screen
        // the segue will do the work of putting the new controller on screen
    }
}
```

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 delegation.

So, for complicated MVC relationships, you might well set the new VC’s delegate to self here.

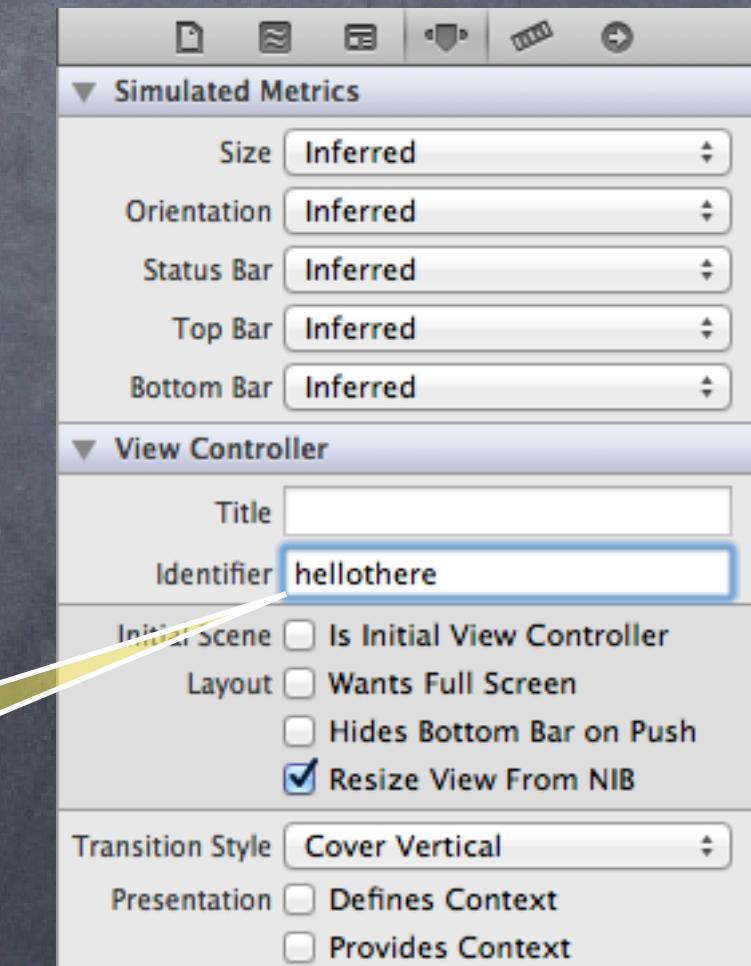
# View Controller

## • 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

- ➊ New application: Psychologist

Our psychologist will make a diagnosis and use the Happiness MVC to communicate it.

Psychologist really has no Model (perhaps its “diagnosis”, but it doesn’t store it anywhere).

Some MVCs are just for presenting user-interface (e.g. UINavigationController).

- ➋ Watch for ...

Reusing the Happiness MVC.

Creating a segue between our new MVC and a Happiness MVC.

Embedding our view controllers in a UINavigationController.

# Coming Up

- ⦿ Next Lecture

- iPad

- ⦿ Friday

- Getting your application running on a device