Stanford CS193p

Developing Applications for iPhone 4, iPod Touch, & iPad

Fall 2010

Today

- Some Miscellany
  Subtleties about IBOutlet Memory Management
  Initialization code options
  Memory Management Tips
- Another "Controller of Controllers"
  UITabBarController
- iPad
  UISplitViewController
  UIPopoverController
- Universal Application
  One binary runs on multiple (iPhone/iPod Touch and iPad) platforms

IBOutlet Memory Management

- There are two “subtleties” to consider regarding IBOutlets

  Do you want your outlets to be public?

  Do you really want to call your outlet’s “setter” from dealloc?

```
@interface MyVC : UIViewController {
    UILabel *outlet;
}
@property (retain) IBOutlet UILabel *outlet;
@end
```

```
@implementation MyVC

@synthesize outlet;

- (void)releaseOutlets {
    self.outlet = nil;
}
- (void)viewDidUnload {
    [self releaseOutlets];
}
- (void)dealloc {
    [self releaseOutlets];
    [super dealloc];
}

@end
```

```
@interface MyVC : UIViewController {
    IBOutlet UILabel *outlet;
}
@end
```

```
@interface MyVC()
@property (retain) IBOutlet UILabel *outlet;
@end

@implementation MyVC

@synthesize outlet;

- (void)viewDidUnload {
    self.outlet = nil;
}
- (void)dealloc {
    [outlet release];
    [super dealloc];
}

@end
```

View Controller Initialization

- Four places to initialize things in View Controller subclasses
  - (id)initWithNibName:(NSString *)nibName bundle:(NSBundle *)bundle; (i.e. override it)
  - (void)awakeFromNib
  - (void)viewDidLoad
  - (void)viewWillAppear:(BOOL)animated;

- Designated Initializer

  Usually only for things that have to be initialized for your view controller to even "make sense".
  Often thought of as the place to initialize things having to do with your Model.
  Definitely not for initializing things in your View (some "UI-related" things ok like self.title).

- awakeFromNib

  Same purpose (generally) as your designated initializer.
  This is called on every object that comes out of a .xib file (instead of its designated initializer!).
  Sometimes UIViewControllers come out of a .xib file (e.g. MainWindow.xib), but sometimes not.
  Usually you want your VC to work when it is alloc/inited or if it comes from a .xib.
  So create a method (e.g. setup) and call it from initWithNibName:bundle: and awakeFromNib.

View Controller Initialization

- `viewDidLoad`

  This is the best place to put non-geometry-related initialization code which pertains to your View.
  You might even add some more views to your hierarchy in this method (stuff you couldn't do in IB).
  Wouldn't be out of the question to put some Model initialization code here, but theoretically this method could be called multiple times, so don't re-initialize something already initialized.
  It's not totally unheard of to have Views which are loaded, but then never appear on screen.
  So consider `viewWillAppear:` for some things, but maybe check to avoid multiple initialization.

- `viewWillAppear:`

  If you have initialization that depends on your View's bounds being set, you must do it here (and <u>not</u> in `viewDidLoad`).

Memory Management

- Alarm should go off in your head when you type alloc/copy/new
  - You should immediately drop everything and go figure out where this object will be released
  - If it's an instance variable, probably you're looking at dealloc
  - If it's a local variable, release it as soon as you're done using it (at least just before return)
  - If it's a local variable that you are returning, autorelease it
- Watch out when you set a variable multiple times in your code
  - Common mistake: Assigning a new value to an instance variable without releaseing old value
  - Fix: Use an @property (retain) and dot notation to set that instance variable in your code
- Go immutable!
  - It's simpler and less error-prone than allocing a mutable object, modifying it, then releaseing it
- Try to use methods other than alloc/copy/new
  - The autorelease mechanism is mostly for objects that are being returned from methods
  - Use that to your advantage by calling methods that return autoreleased objects
- But don't abuse autorelease by creating a huge pool

Memory Management

Fine:

```
- (NSDictionary *)testVariableValues
{
    NSMutableDictionary *returnValue = [[NSMutableDictionary alloc] init];
    [returnValue setObject:[NSNumber numberWithFloat:3.5] forKey:@”x”];
    [returnValue setObject:[NSNumber numberWithFloat:23.8] forKey:@”y”];
    return [returnValue autorelease];
}
```

Better:

```
- (NSDictionary *)testVariableValues
{
    NSMutableDictionary *returnValue = [NSMutableDictionary dictionary];
    [returnValue setObject:[NSNumber numberWithFloat:3.5] forKey:@”x”];
    [returnValue setObject:[NSNumber numberWithFloat:23.8] forKey:@”y”];
    return returnValue;
}
```

Best:

```
- (NSDictionary *)testVariableValues
{
    return [NSDictionary dictionaryWithObjectsAndKeys:[NSNumber numberWithFloat:3.5], @”x”,
                                                      [NSNumber numberWithFloat:23.8], @”y”, nil];
}
```

Memory Management

Bad:

```
- (NSString *)tenThousandGs
{
    NSString *s = “”;
    for (int i = 0; i < 10000; i++) s = [s stringByAppendingString:@“G”];
    return s;
}
```

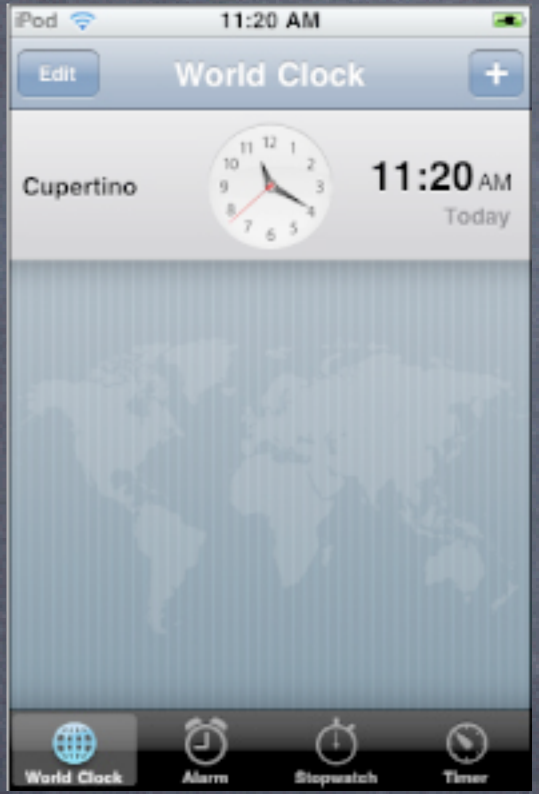
Good:

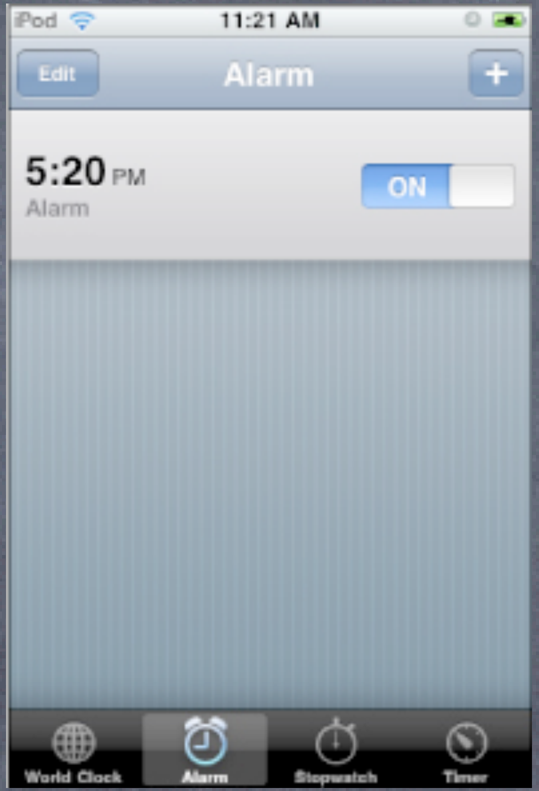
```
- (NSString *)tenThousandGs
{
    NSMutableString *s = [[NSMutableString alloc] init];
    for (int i = 0; i < 10000; i++) [s appendString:@“G”];
    return [s autorelease];
}
```

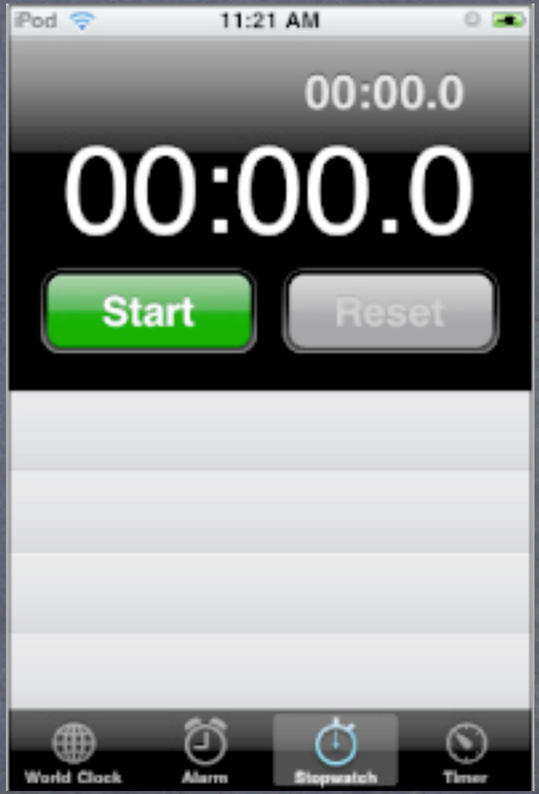
Best:

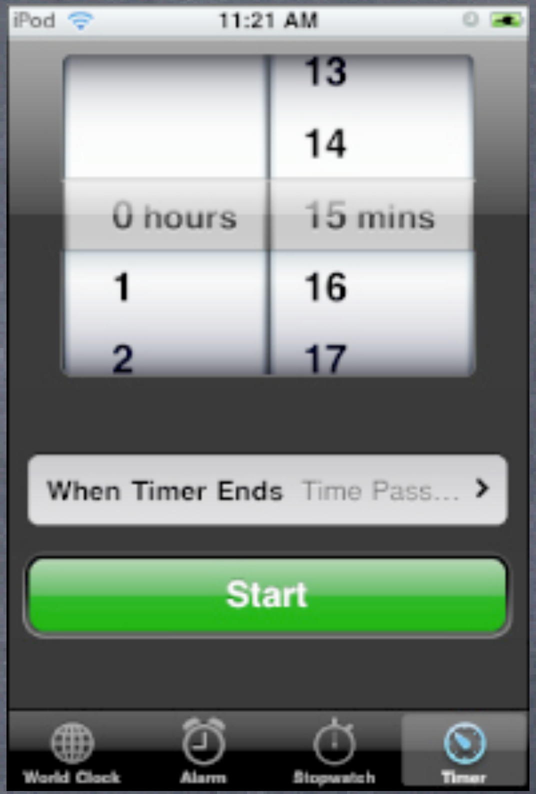
```
- (NSString *)tenThousandGs
{
    NSMutableString *s = [NSMutableString string];
    for (int i = 0; i < 10000; i++) [s appendString:@“G”];
    return s;
}
```

UITabBarController

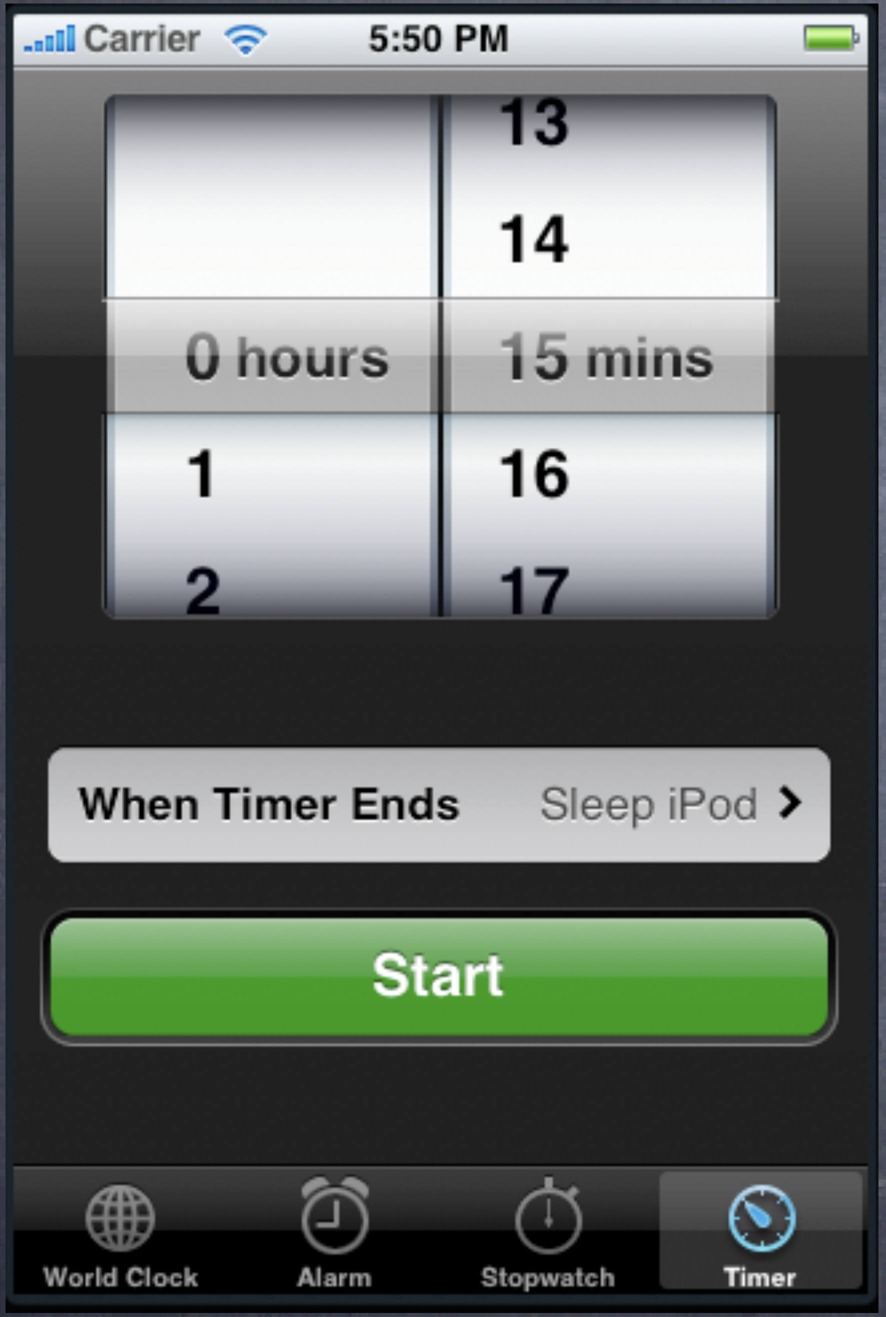


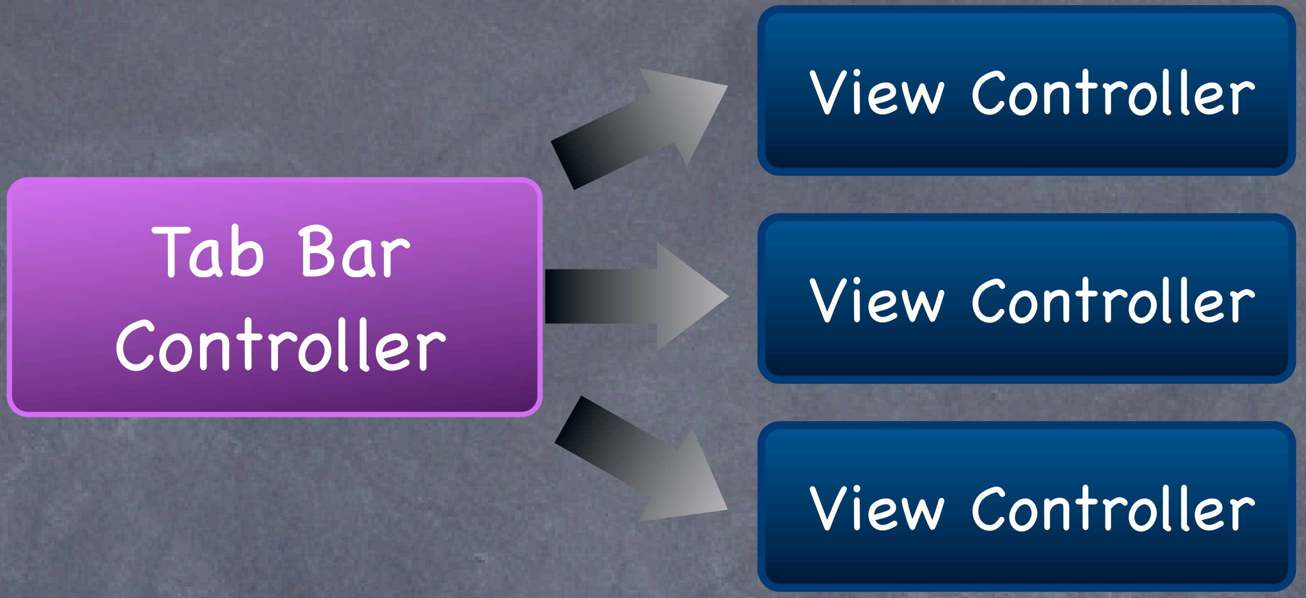






UITabBarController



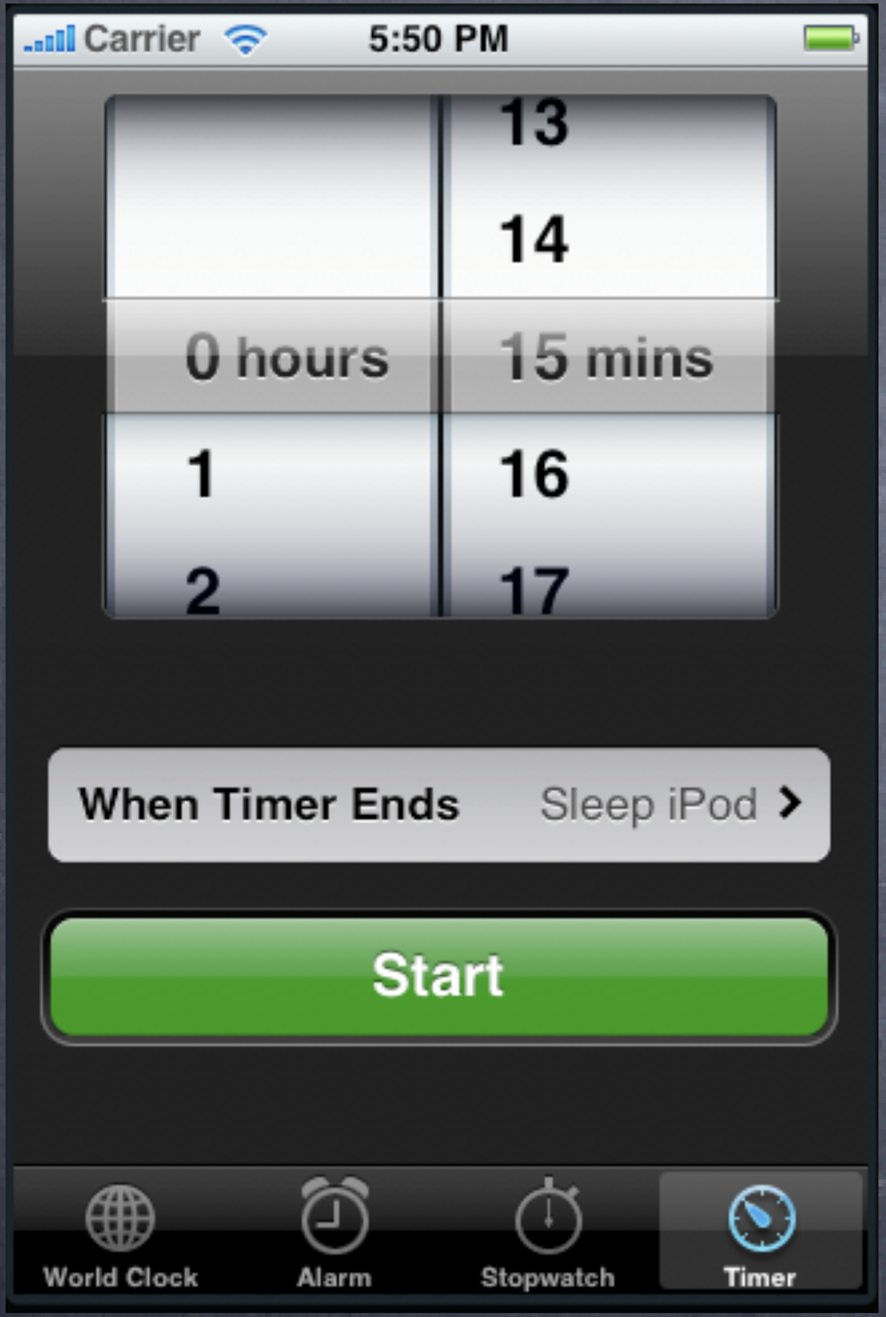

Tab Bar Controller

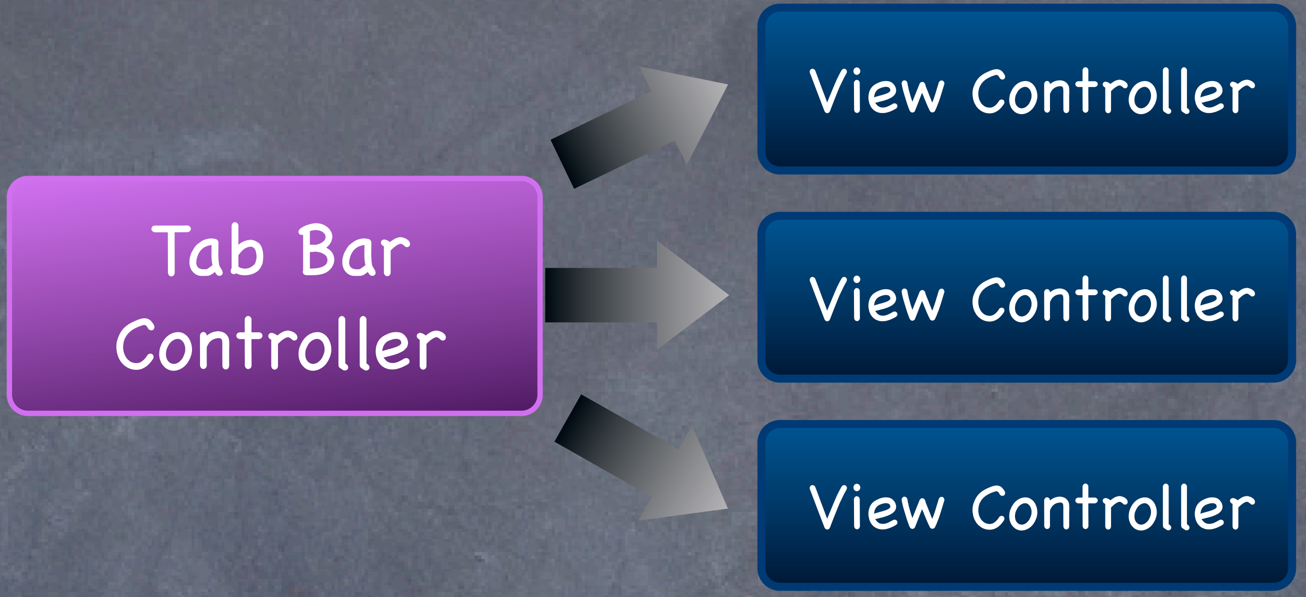
View Controller

View Controller

View Controller

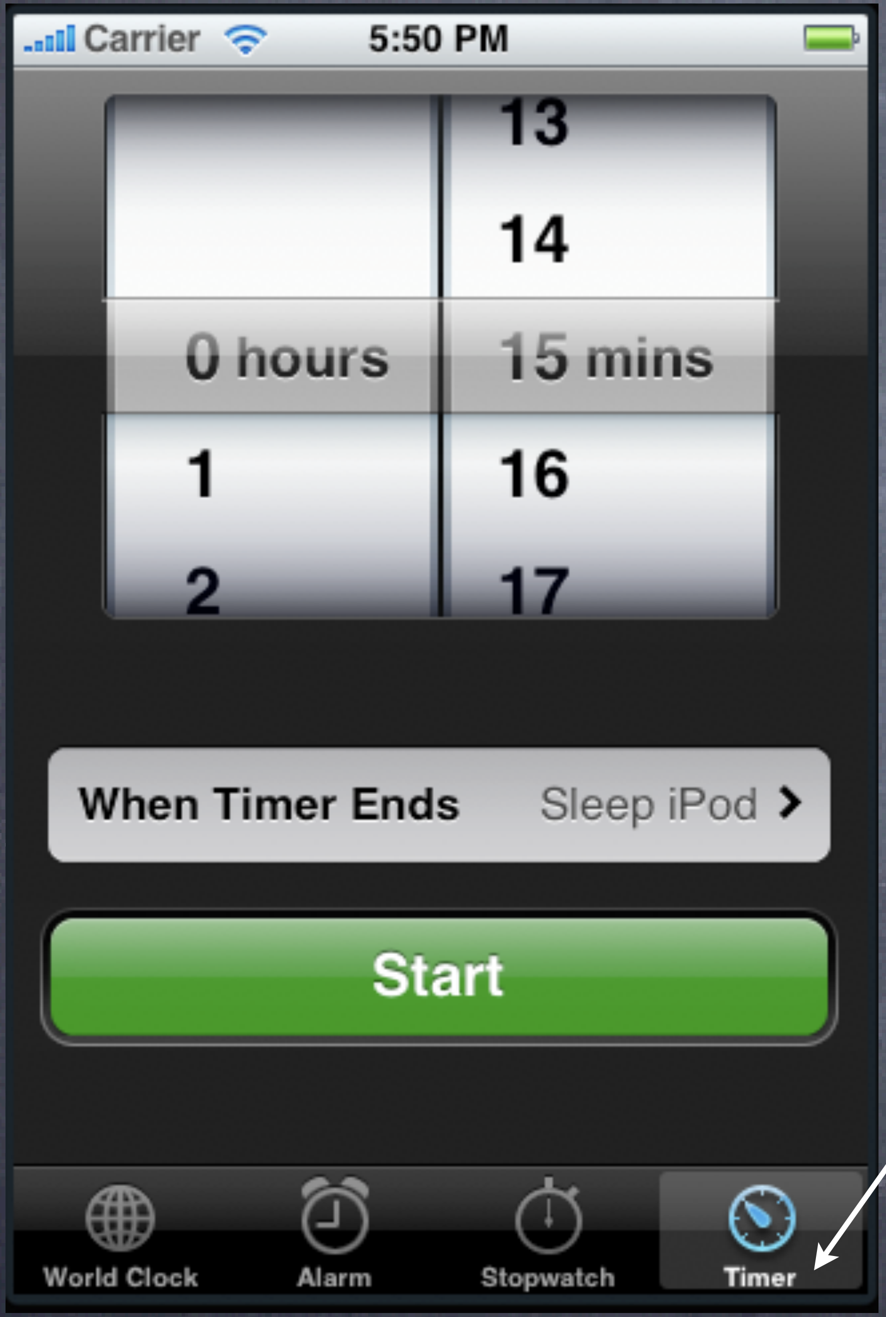
UITabBarController

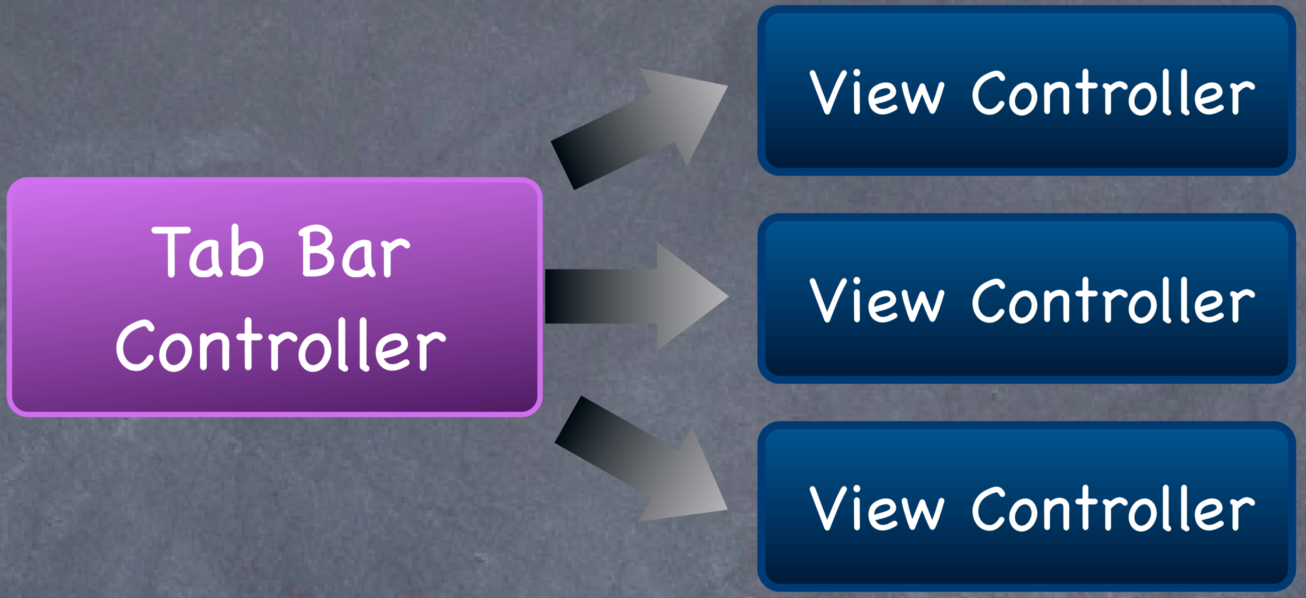




```
- (BOOL)application:(UIApplication *)
          didFinishLaunchingWithOptions:(NSDictionary *)
{
   UIViewController *vc1 = ...;
   UIViewController *vc2 = ...;

   UITabBarController *tbc = [[UITabBarController alloc] init];
   tbc.viewControllers = [NSArray arrayWithObjects: vc1, vc2, ..., nil];
   [vc1 release]; [vc2 release];

   [window addSubview:tbc.view];
   [window makeKeyAndVisible];
   return YES;
}
```

UITabBarController




Tab Bar Controller

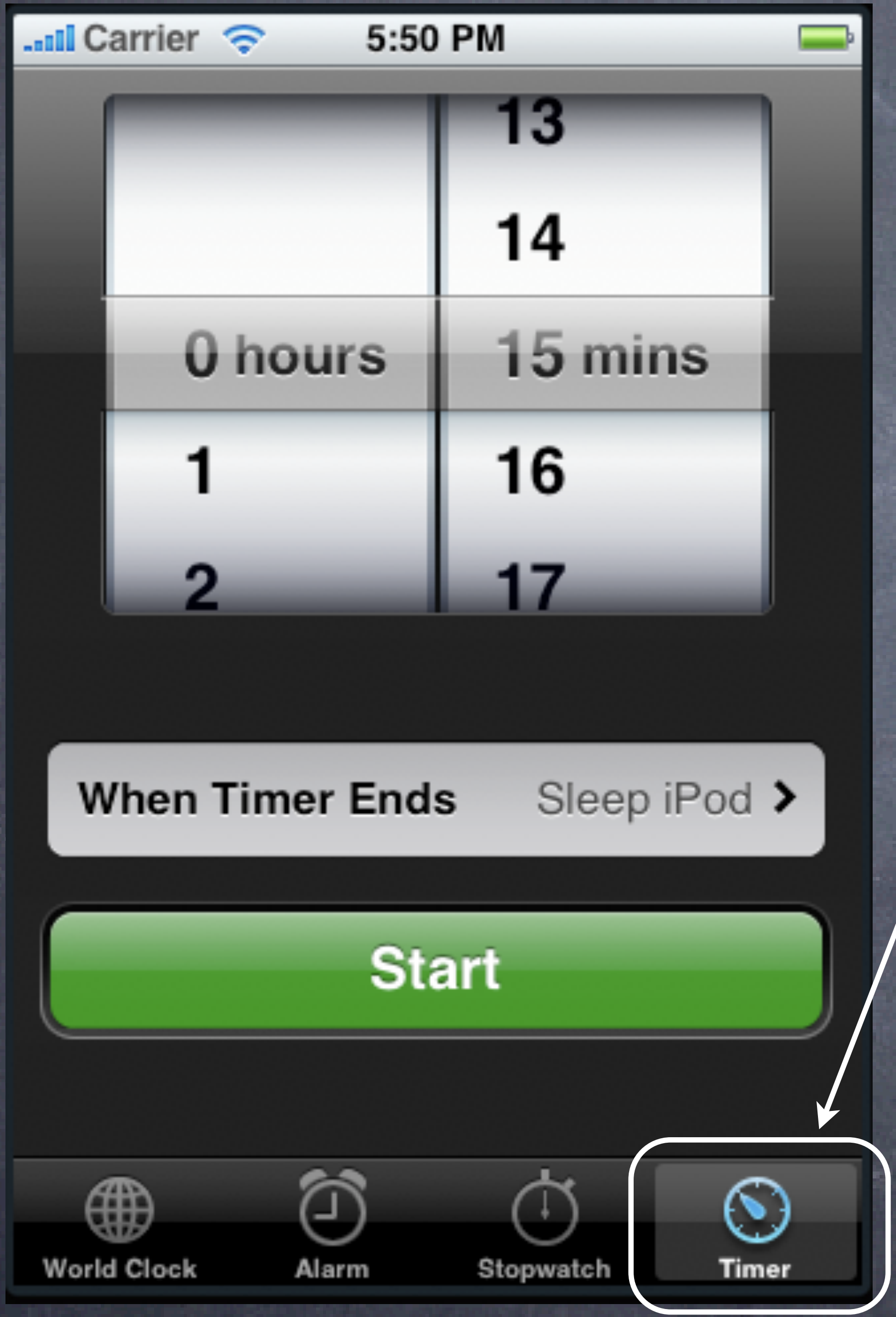
View Controller

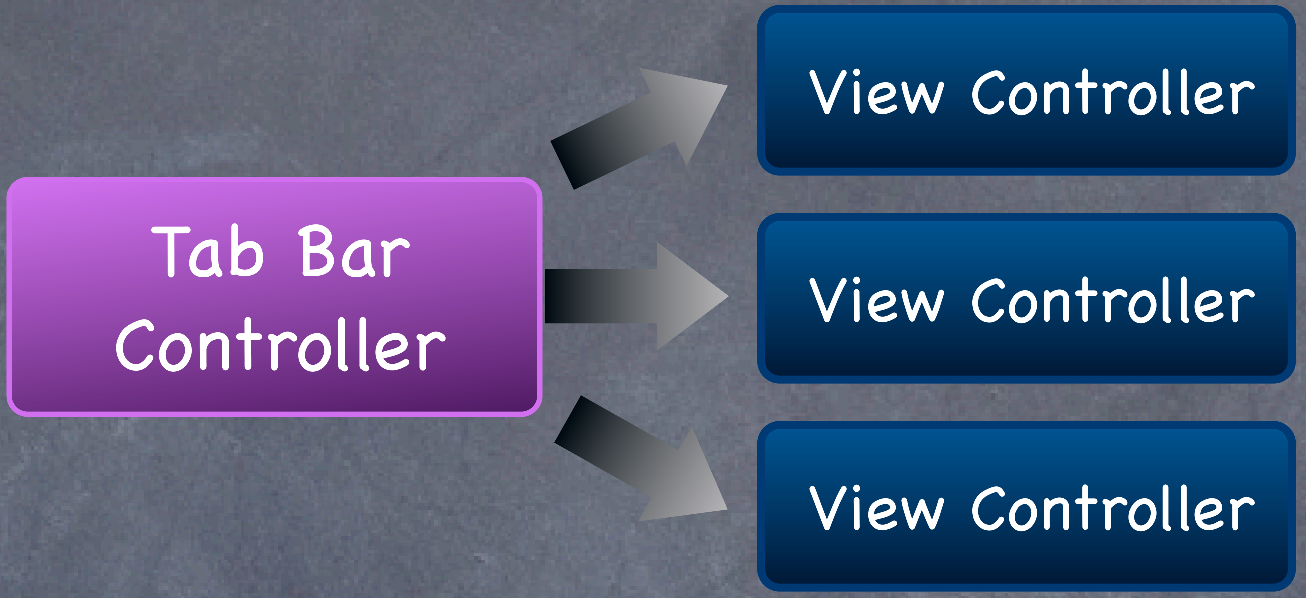
View Controller

View Controller

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

UITabBarController

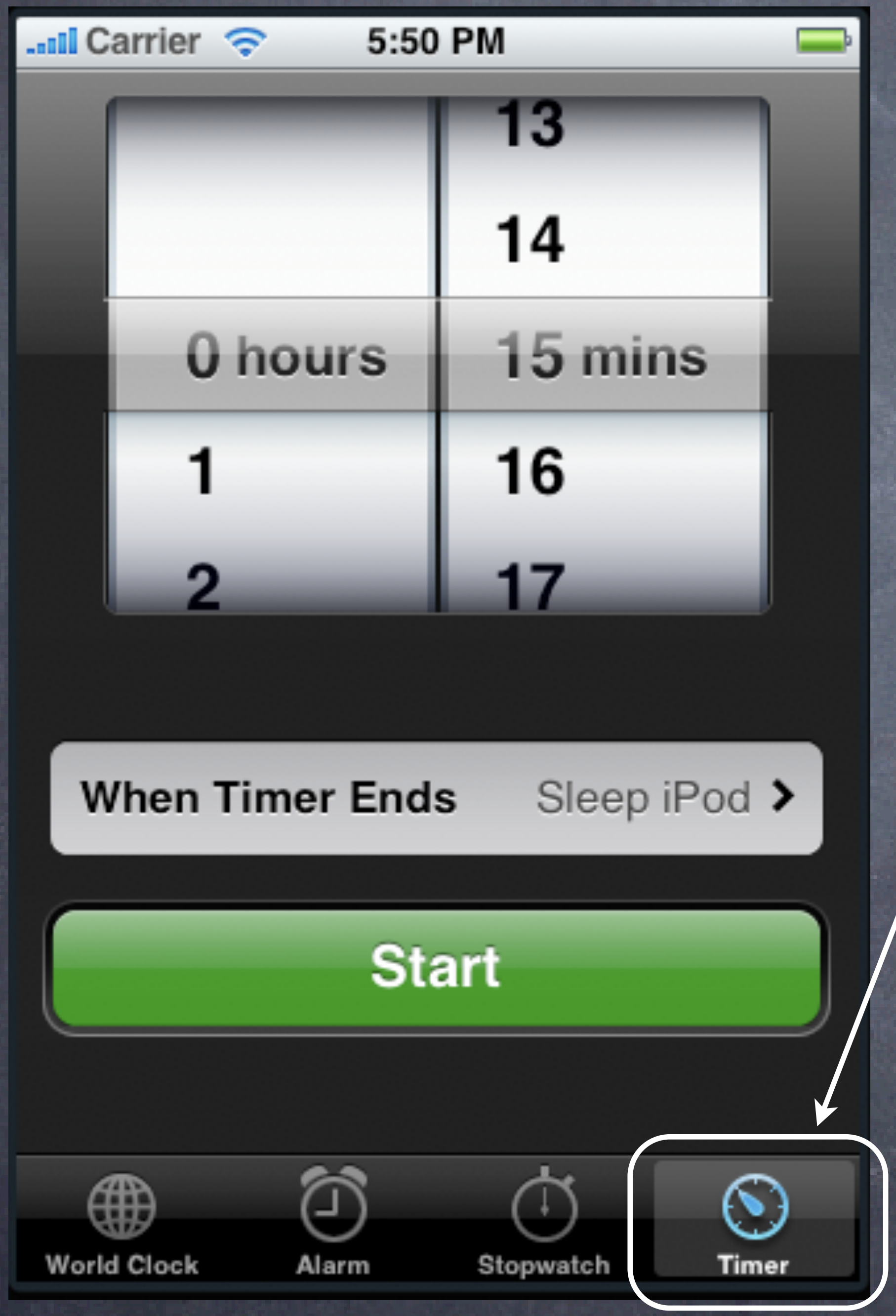


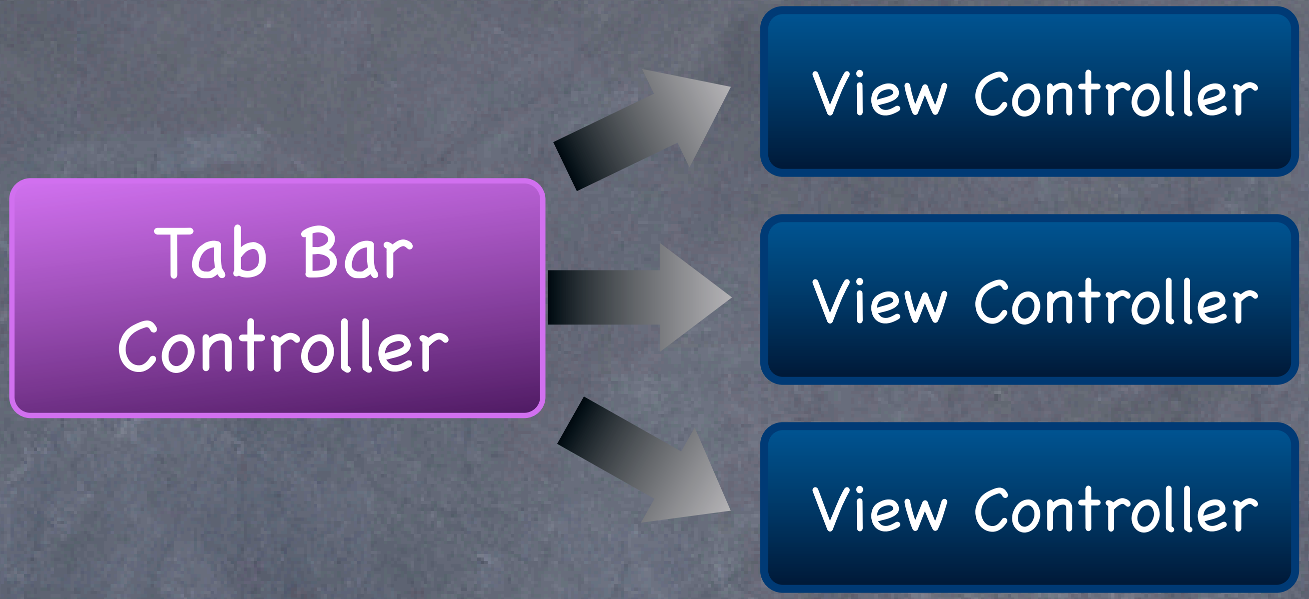


… or it can be UIViewController's
`@property UITabBarItem *tabBarItem;`

Stanford
CS193p
Fall 2010

UITabBarController




Tab Bar Controller

View Controller

View Controller

View Controller

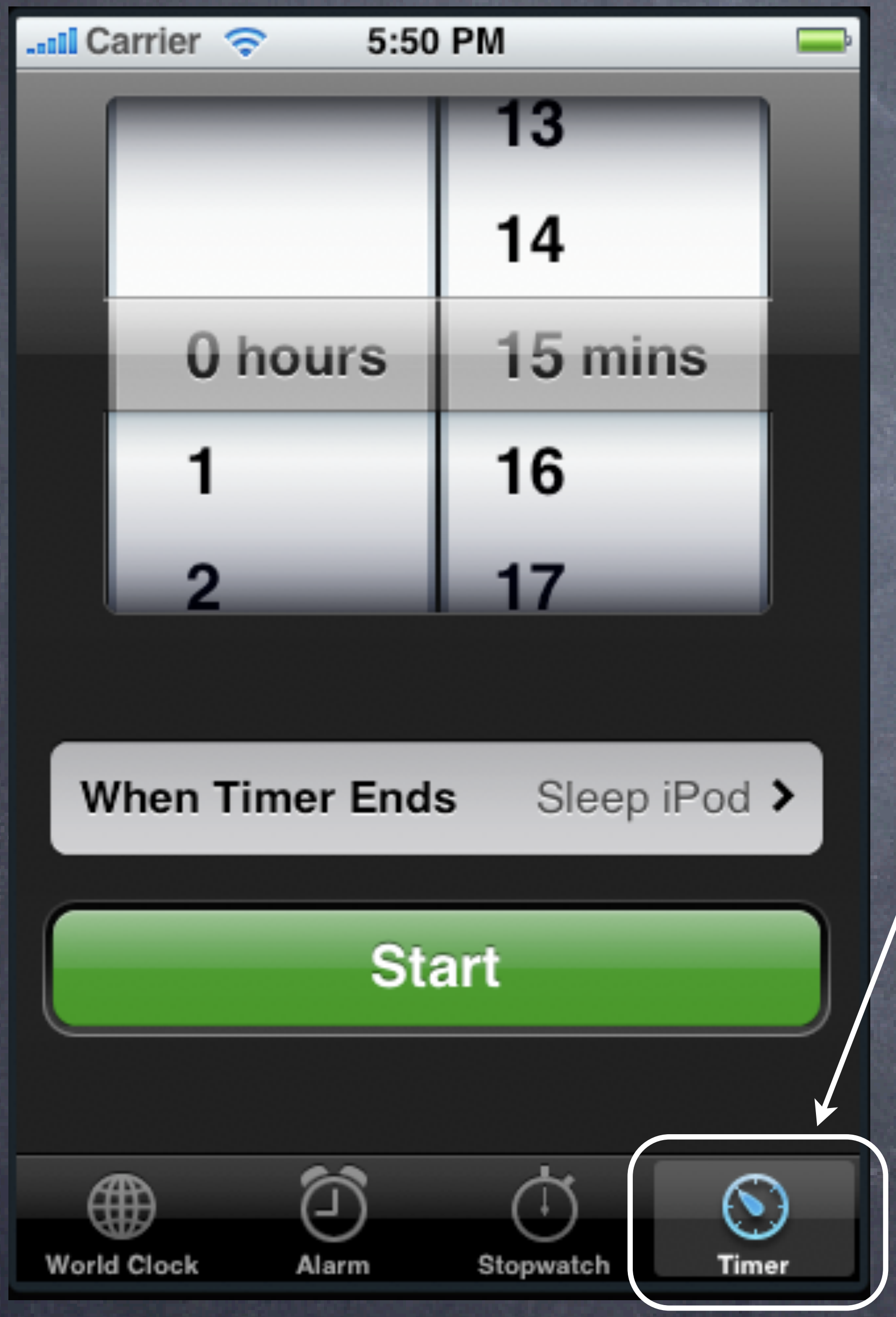
... or it can be UIViewController's

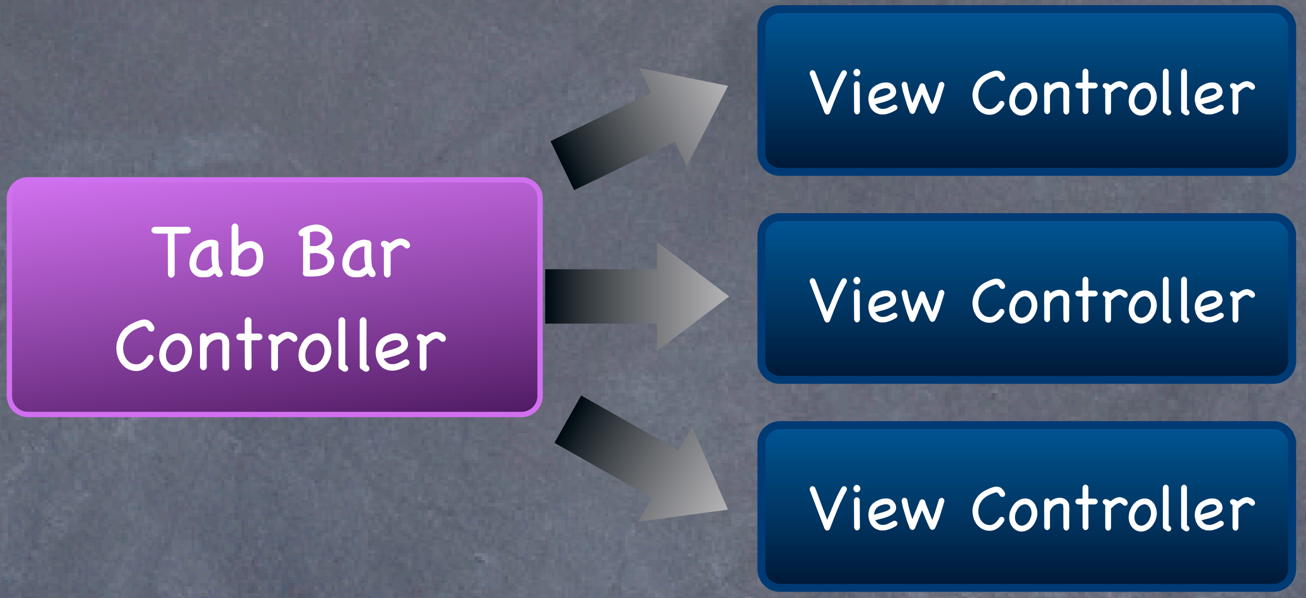
```
@property UITabBarItem *tabBarItem;
```

Can't do this in viewDidLoad because the tab is shown before view is loaded. In fact, clicking on the tab is what loads the view.

```
- (void)setup    // call from initWithNibName:bundle: and awakeFromNib
{
    UITabBarItem *item = [[UITabBarItem alloc]
          initWithTitle:@"Timer"
                  image:[UIImage imageNamed:@"timer.png"]  // from Resources
                    tag:0];                                // identifying tag, can ignore
    self.tabBarItem = item;
    [item release];
}
```

UITabBarController





Tab Bar Controller

View Controller

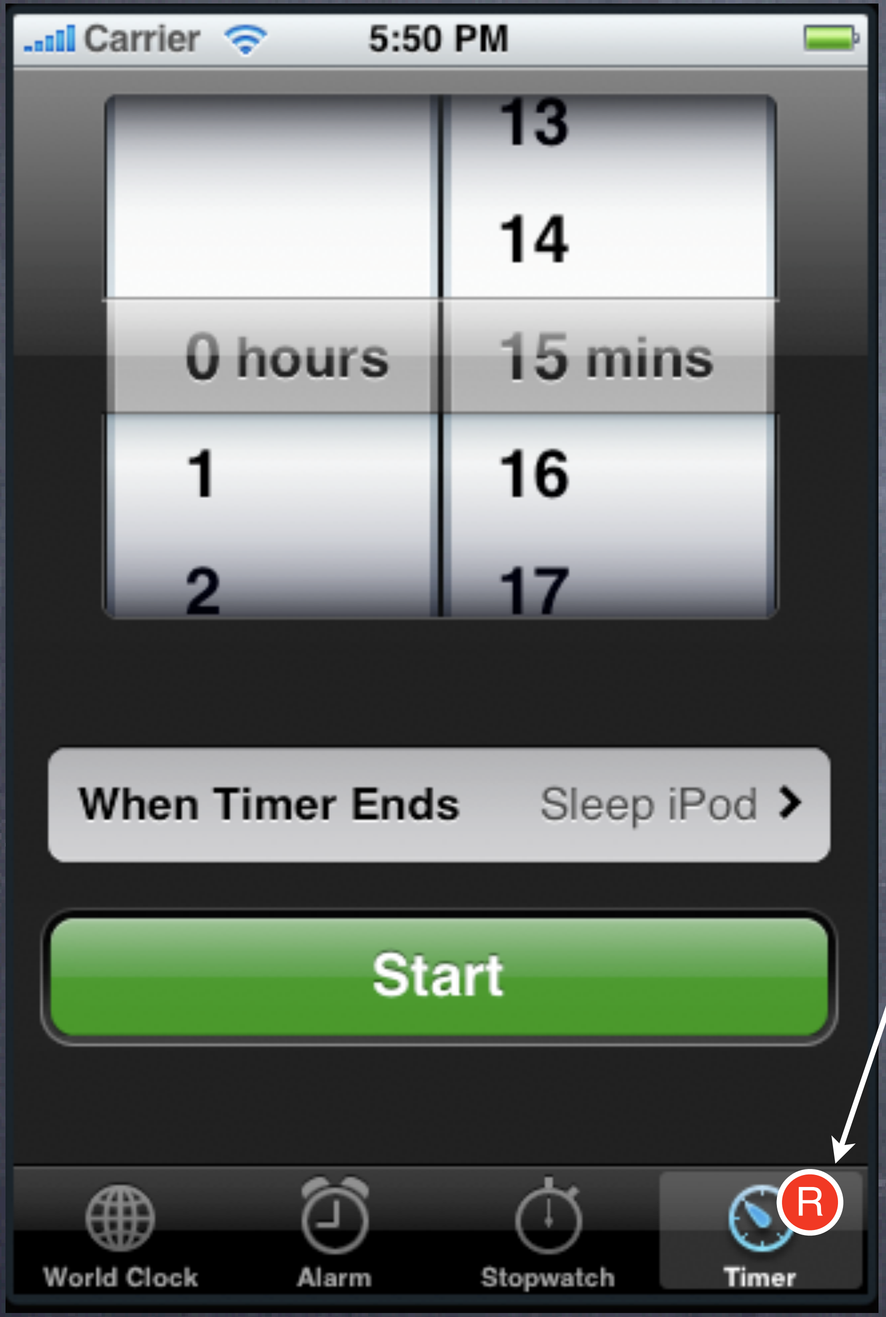
View Controller

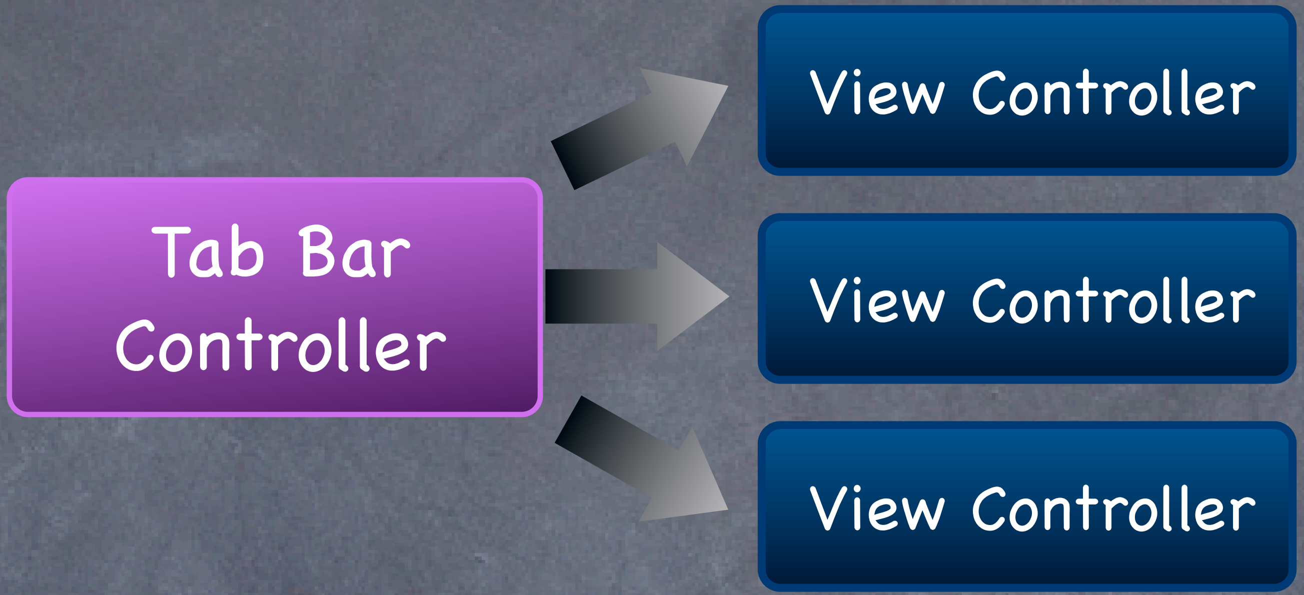
View Controller

… or it can be UIViewController's
@property UITabBarItem *tabBarItem;

```
- (void)setup    // call from initWithNibName:bundle: and awakeFromNib
{
    UITabBarItem *item = [[UITabBarItem alloc]
          initWithSystemItem:UITabBarSystemItemSearch
                     tag:0];                                  // identifying tag, can ignore
    self.tabBarItem = item;
    [item release];
}
```

UITabBarController

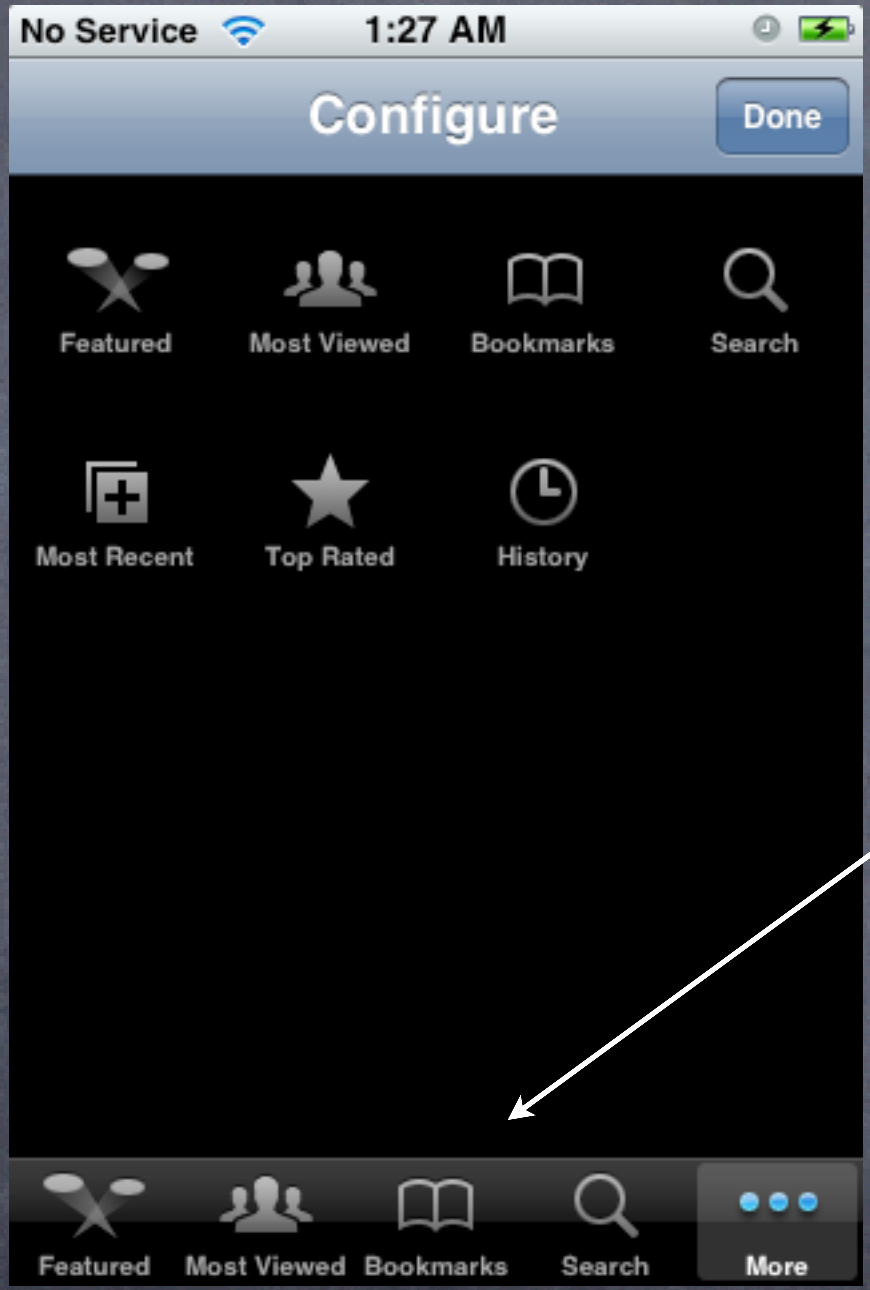


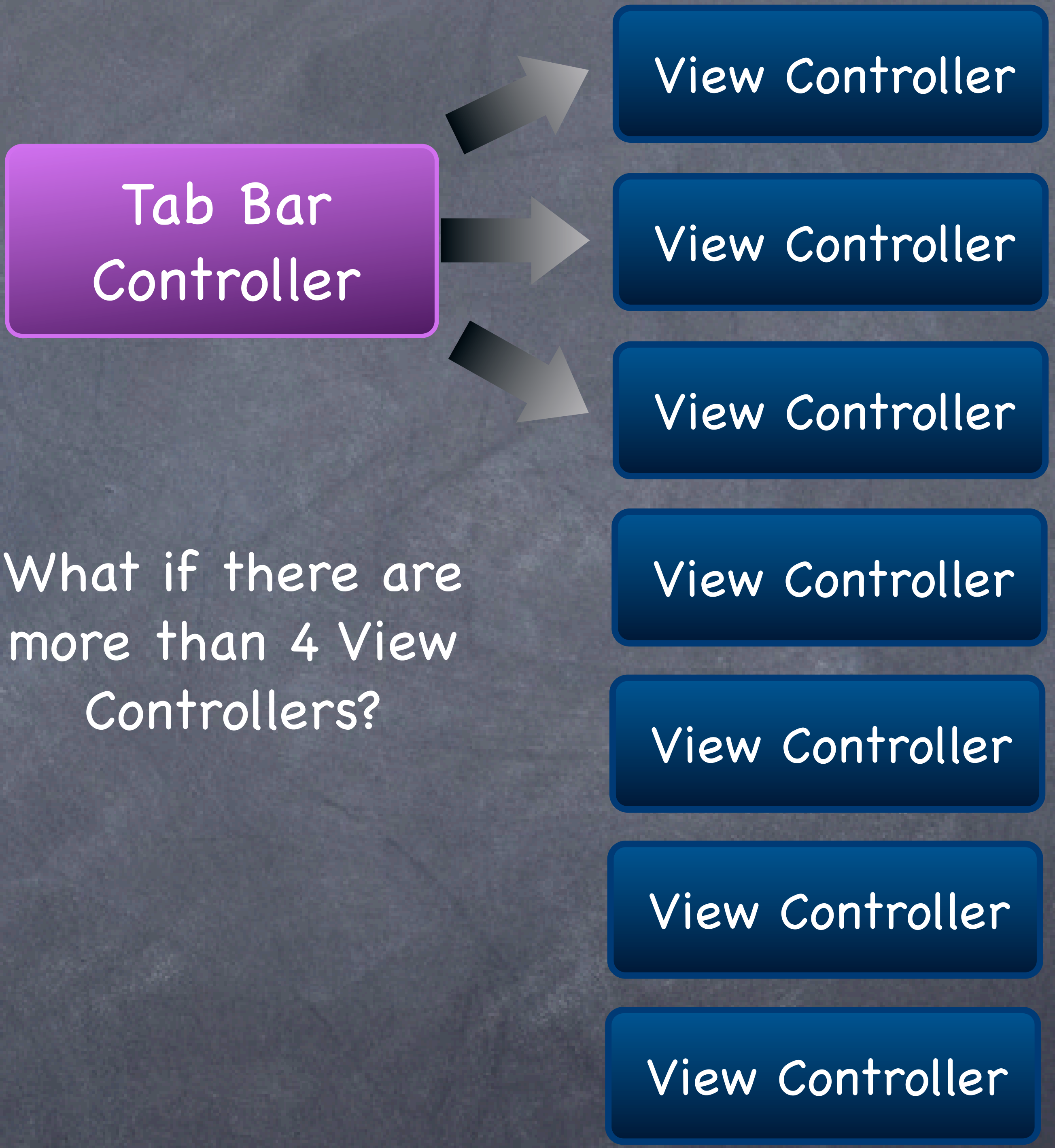


The tabBarItem can also be used to set a badge value on the tab.

```
- (void)somethingHappenedToCauseUsToNeedToShowABadgeValue
{
    self.tabBarItem.badgeValue = @“R”;
}
```

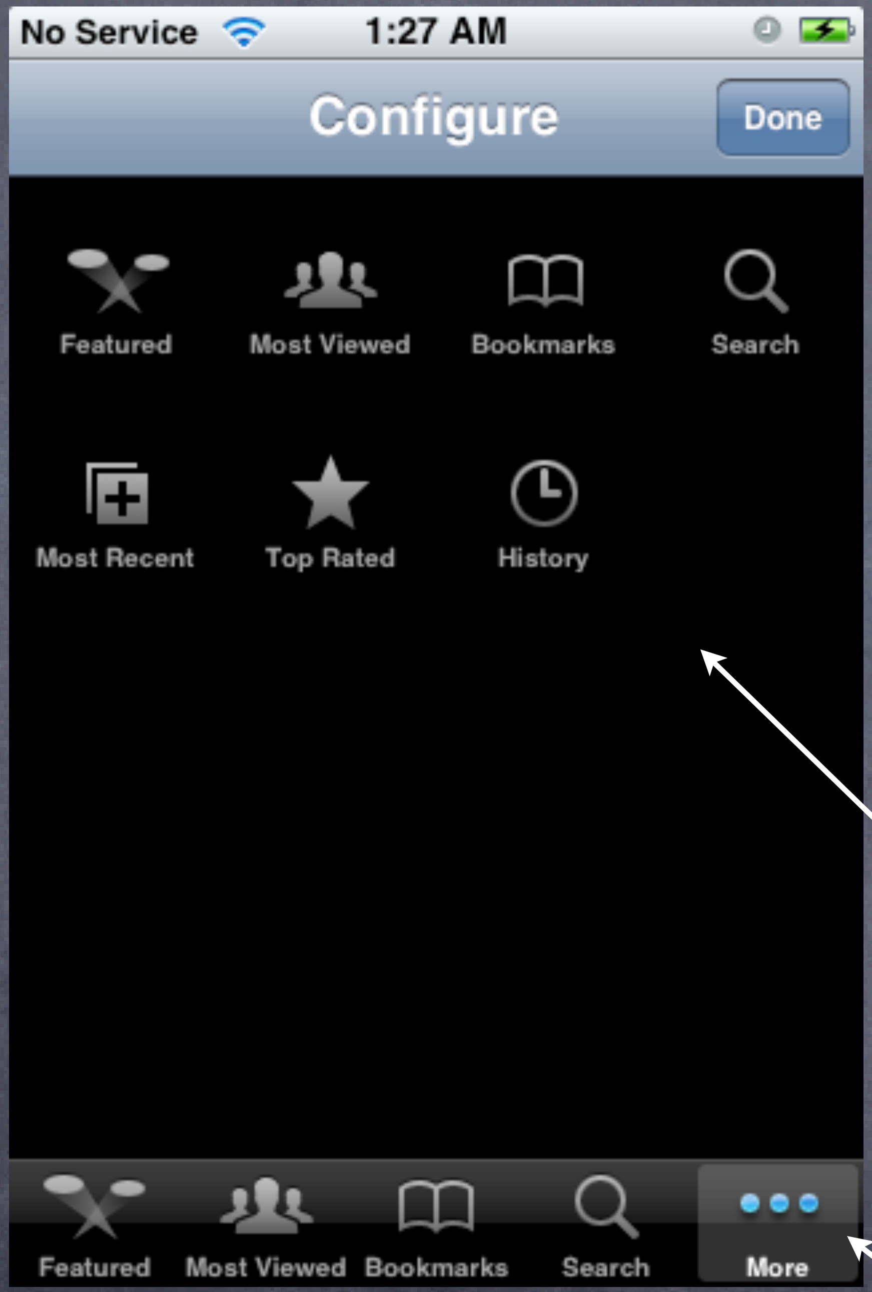
UITabBarController

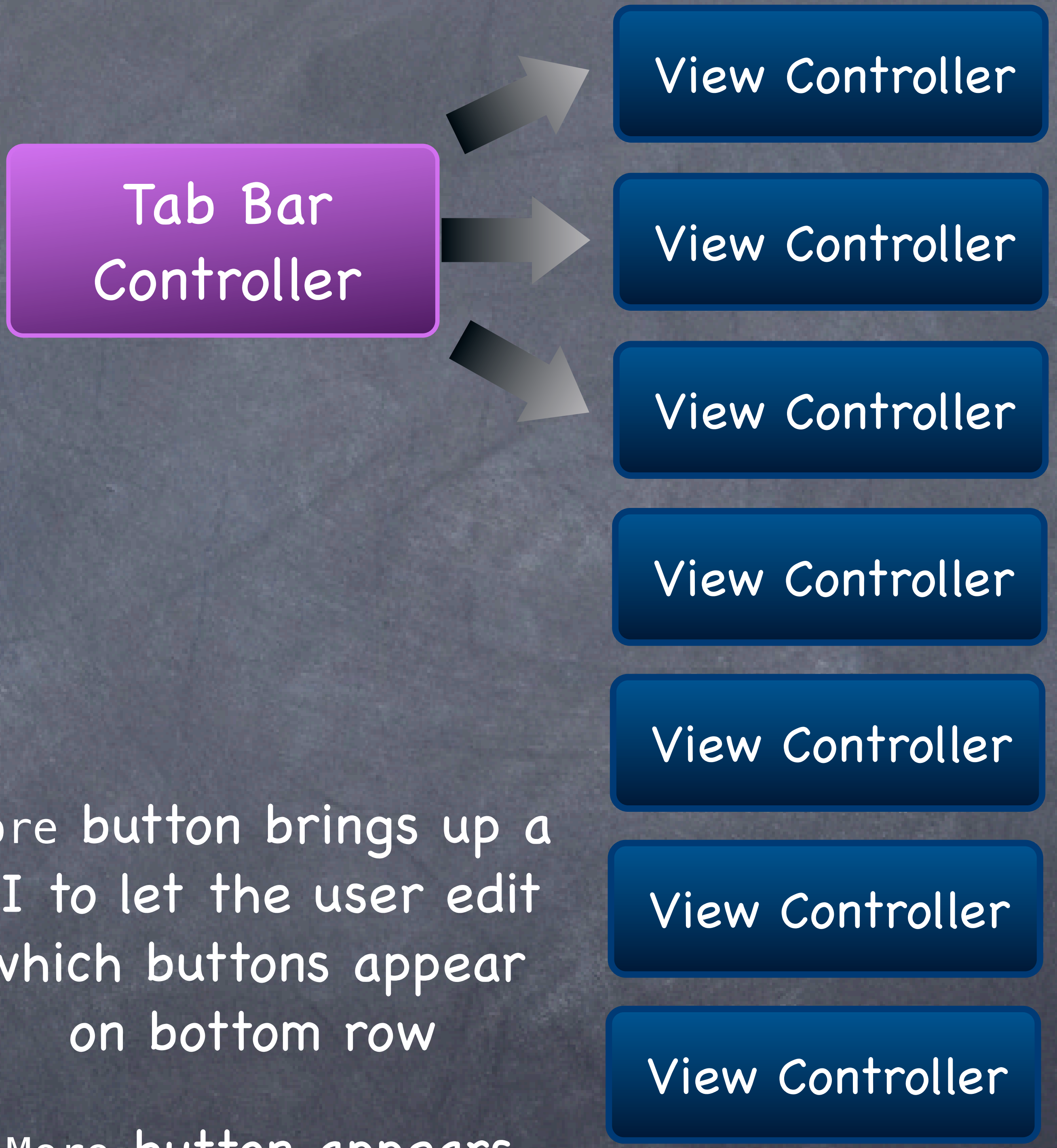




What if there are more than 4 View Controllers?

UITabBarController

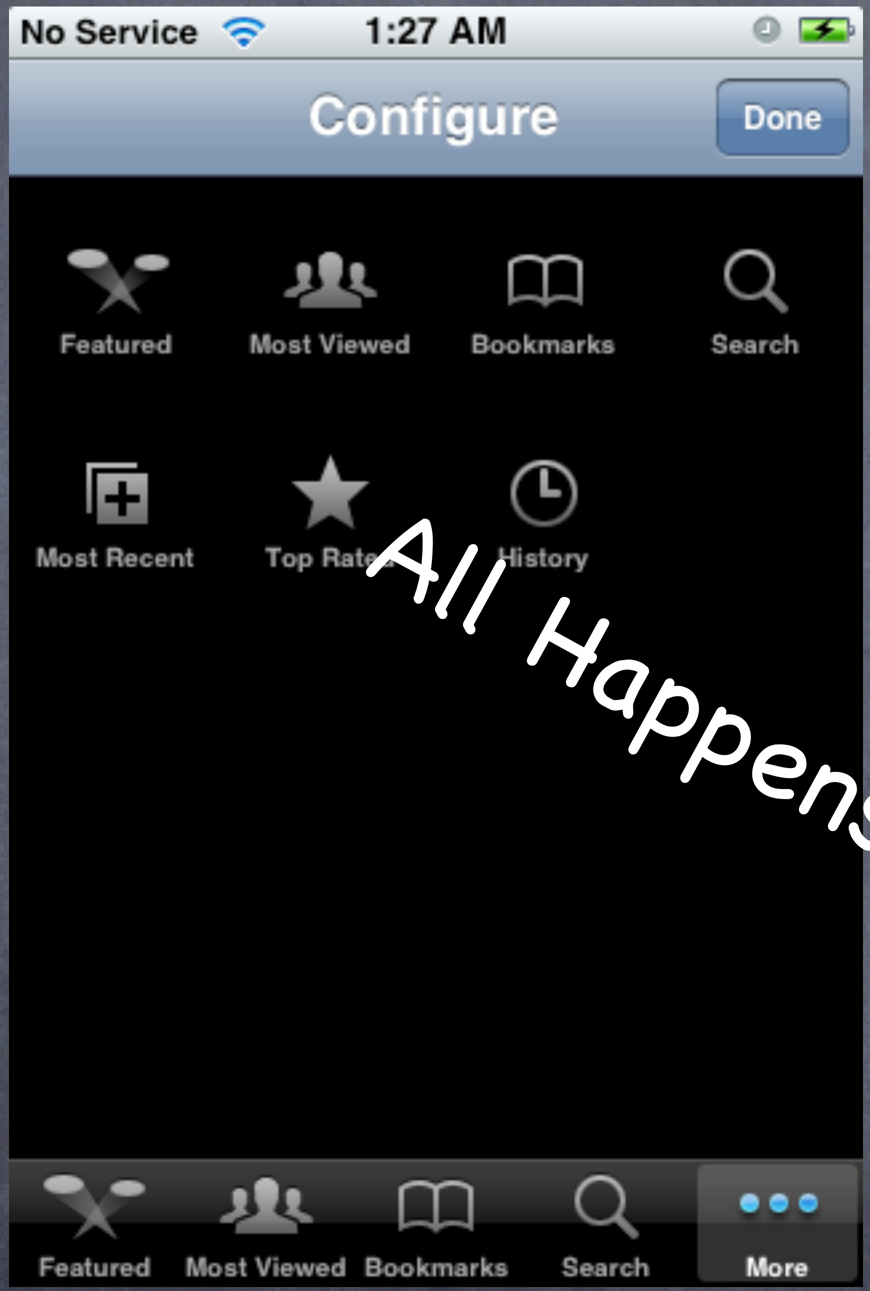


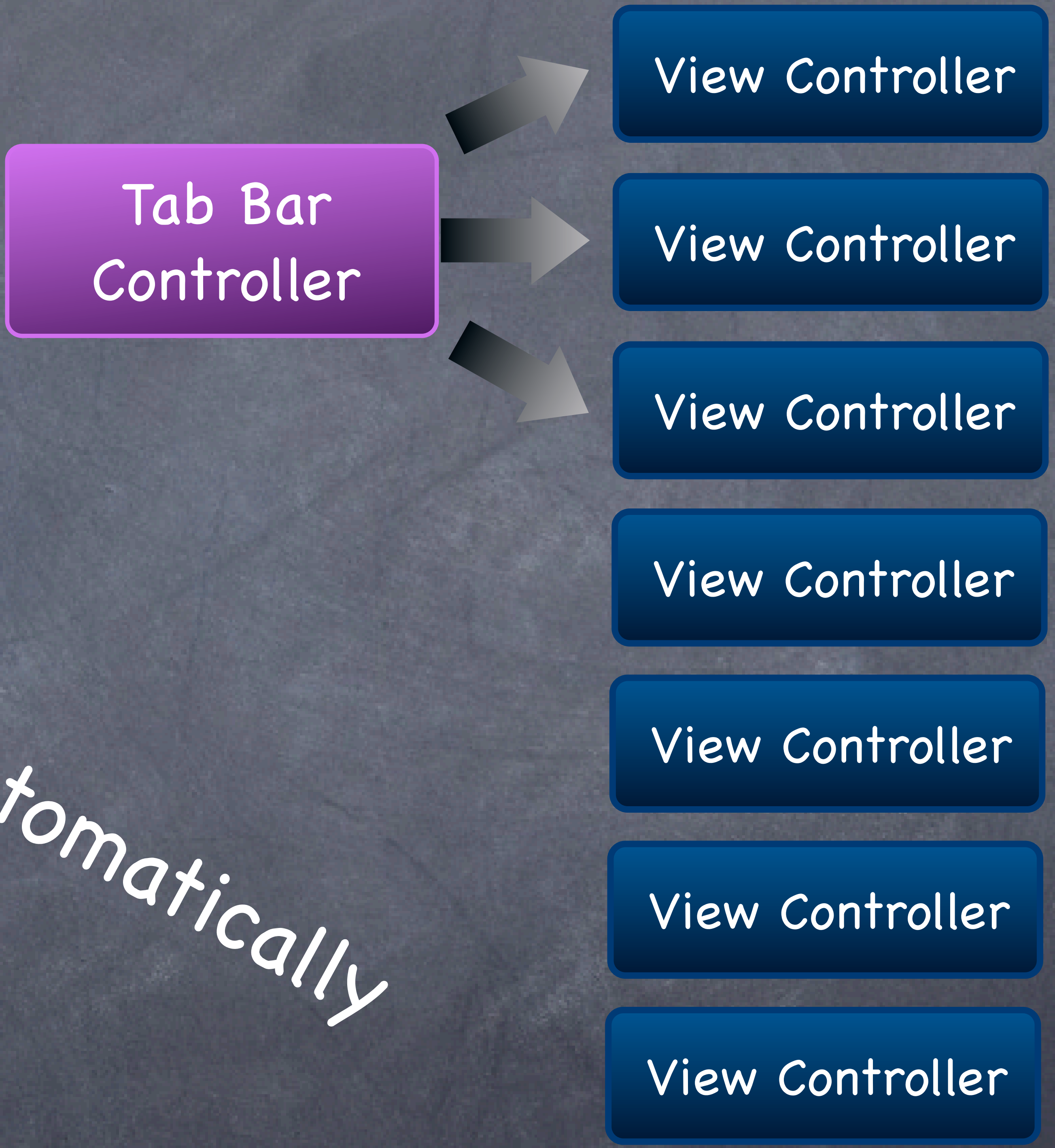


More button brings up a UI to let the user edit which buttons appear on bottom row

A More button appears.

UITabBarController





All Happens Automatically

Combine?

- Can you combine UINavigationController & UITabBarController?

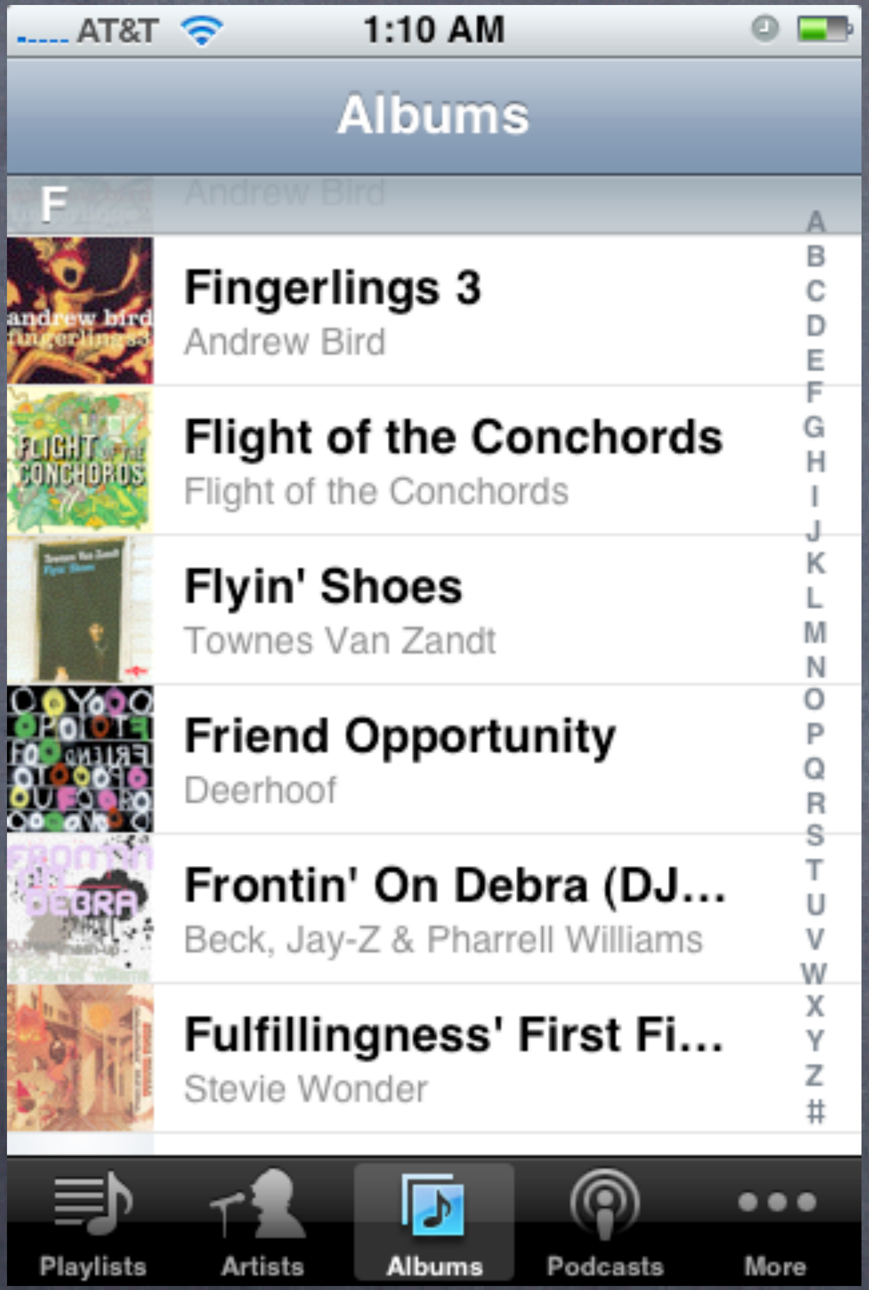
  Certainly. Quite common.

- UINavigationController goes "inside" the UITabBarController

  Never the other way around.

```
- (BOOL)application:(UIApplication *)app didFinishLaunchingWithOptions:(NSDictionary *)options
{
    UINavigationController *nav1 = [[UINavigationController alloc] init];
    UINavigationController *nav2 = [[UINavigationController alloc] init];
    [nav1 pushViewController:...];
    [nav2 pushViewController:...];
    // here we would likely to want to release the view controllers pushed onto the navigation controllers

    UITabBarController *tbc = [[UITabBarController alloc] init];   // should be ivar released in dealloc
    tbc.viewControllers = [NSArray arrayWithObjects:nav1, nav2, nil];
    [nav1 release]; [nav2 release];

    [window addSubview:tbc.view];
    [window makeKeyAndVisible];
    return YES;
}
```

Combine



Andrew Bird

Fingerlings 3
Andrew Bird

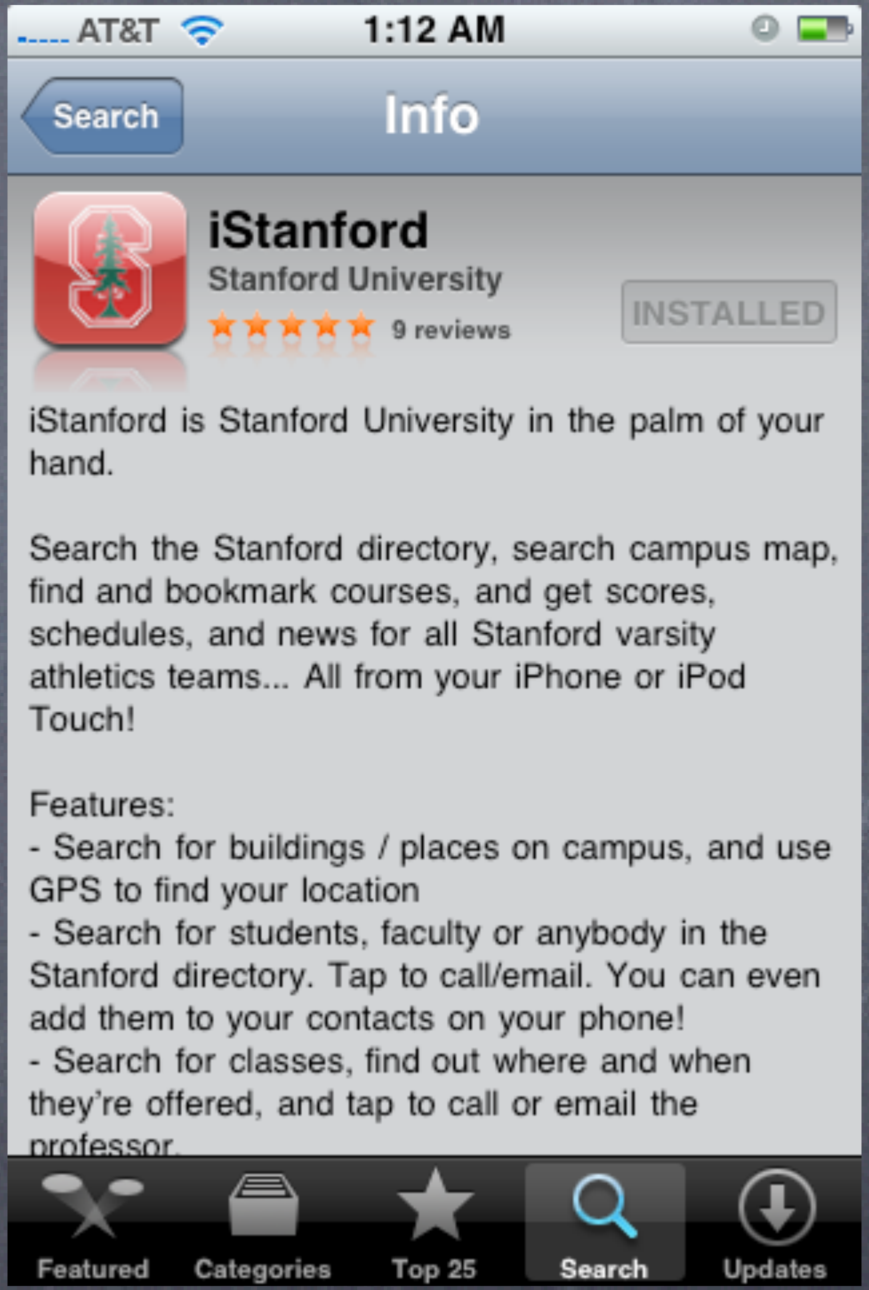
Flight of the Conchords
Flight of the Conchords

Flyin' Shoes
Townes Van Zandt

Friend Opportunity
Deerhoof

Frontin' On Debra (DJ...
Beck, Jay-Z & Pharrell Williams

Fulfillingness' First Fi...
Stevie Wonder



iStanford
Stanford University
★★★★★ 9 reviews

iStanford is Stanford University in the palm of your hand.

Search the Stanford directory, search campus map, find and bookmark courses, and get scores, schedules, and news for all Stanford varsity athletics teams... All from your iPhone or iPod Touch!

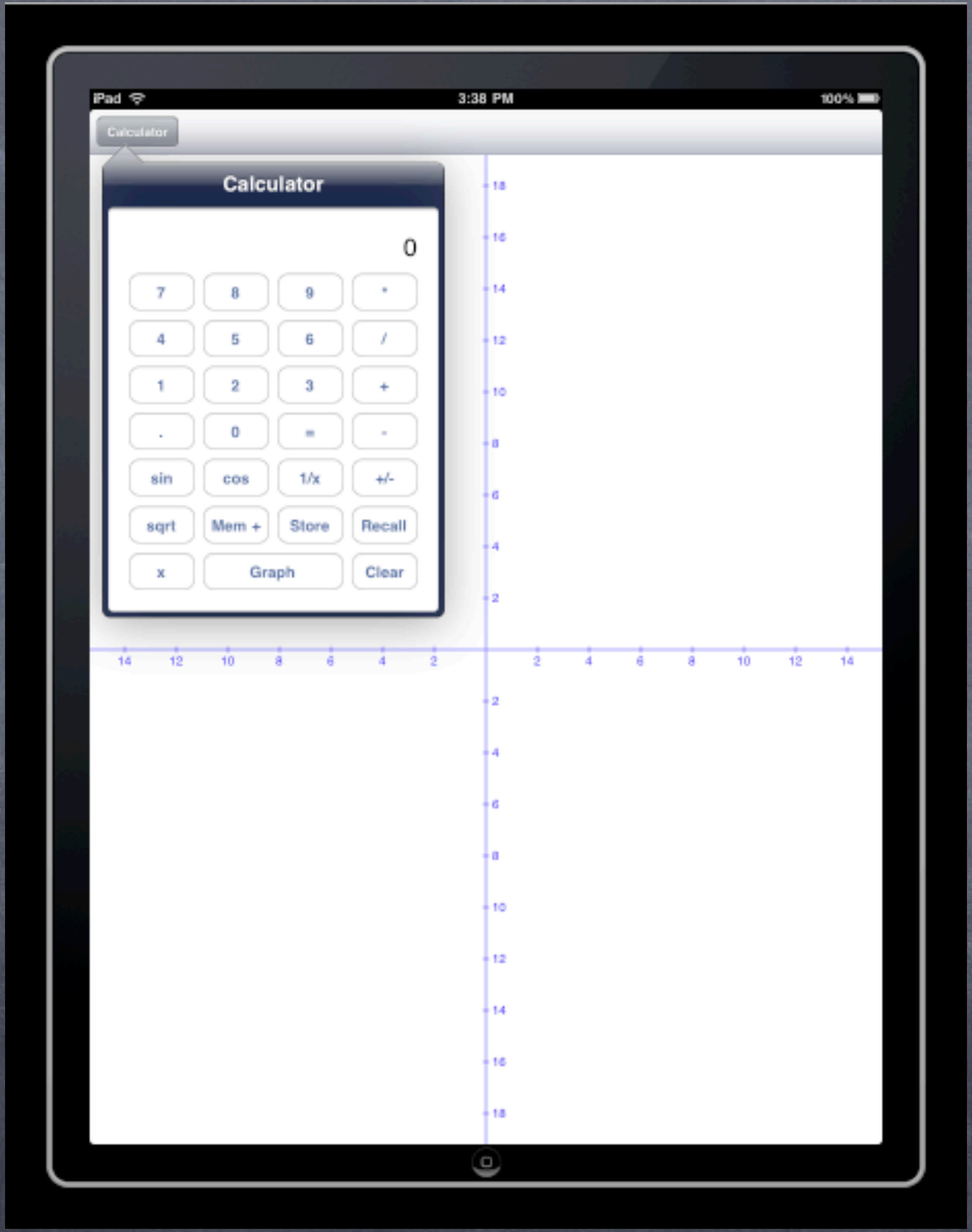
Features:
- Search for buildings / places on campus, and use GPS to find your location
- Search for students, faculty or anybody in the Stanford directory. Tap to call/email. You can even add them to your contacts on your phone!
- Search for classes, find out where and when they're offered, and tap to call or email the professor.

UISplitViewController

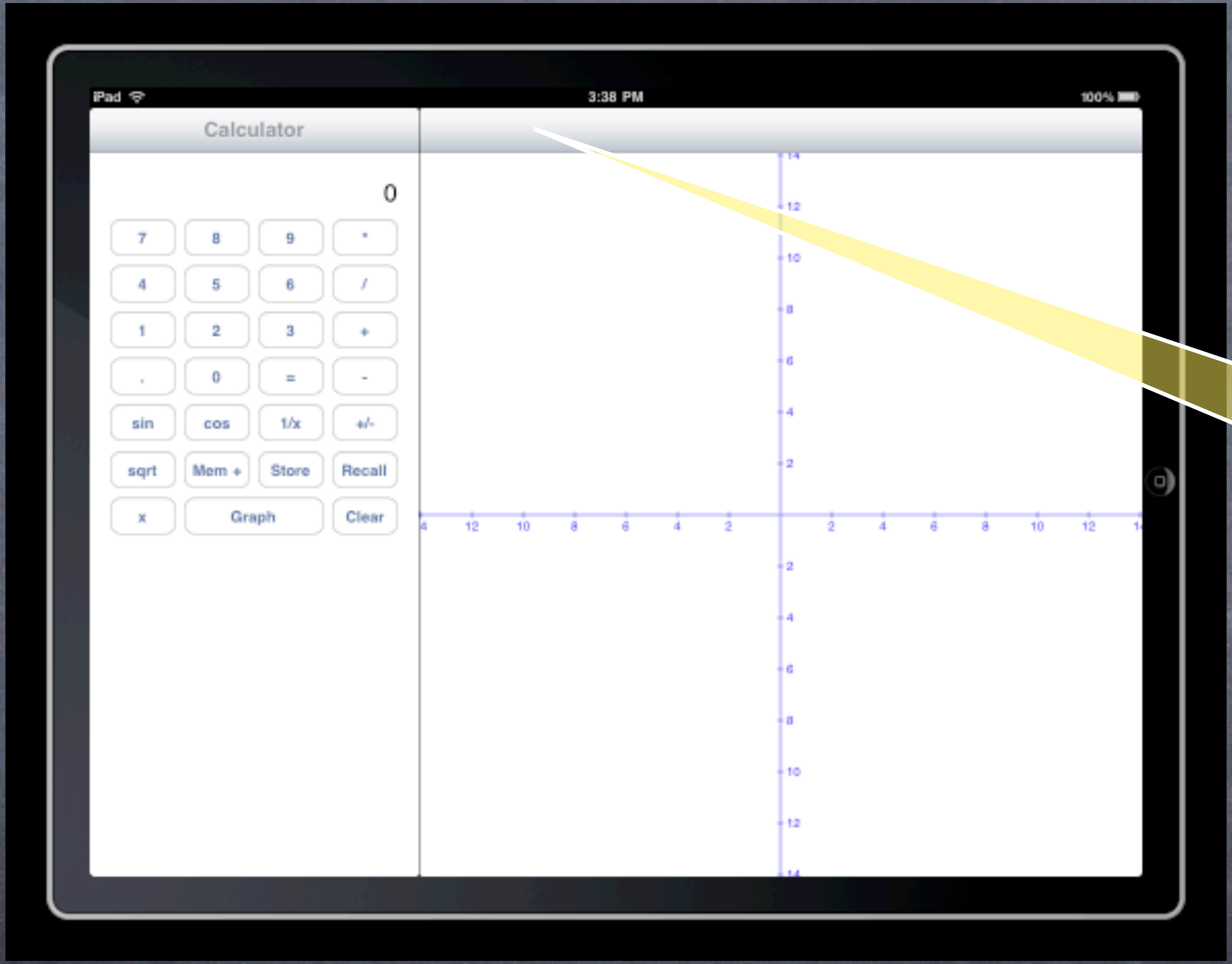
- Only makes sense on large screens
- Side-by-side views in Landscape orientation
- Popover from a bar button in Portrait orientation
- API almost identical to Tab Bar, but only two VC's allowed

```
- (BOOL)application:(UIApplication *)app didFinishLaunchingWithOptions:(NSDictionary *)options
{
    UIViewController *vc1 = [[CalculatorViewController alloc] init];
    UIViewController *vc2 = [[GraphViewController alloc] init];

    UISplitViewController *svc = [[UISplitViewController alloc] init];  // should be ivar released in dealloc
    svc.viewControllers = [NSArray arrayWithObjects:vc1, vc2, nil];
    [vc1 release]; [vc2 release];

    [window addSubview:svc.view];
    [window makeKeyAndVisible];
    return YES;
}
```

UISplitViewController



UISplitViewController



Notice these nice toolbars at the top of both the left and right views

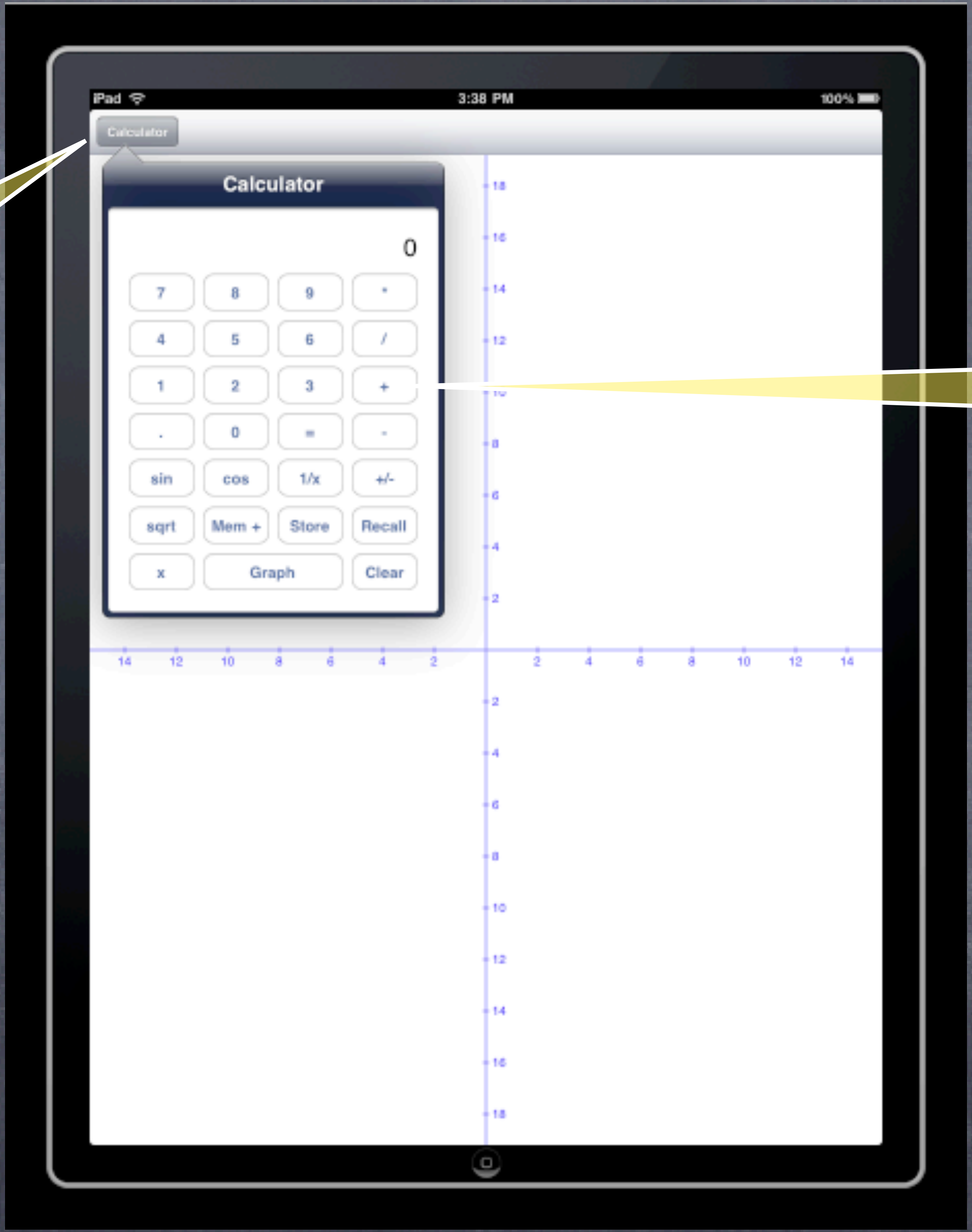
UISplitViewController

- Common to put both view controllers inside navigation controllers

  Makes it easy to add titles, bar buttons, etc.

```
- (BOOL)application:(UIApplication *)app didFinishLaunchingWithOptions:(NSDictionary *)options
{
    UIViewController *left = ...;
    UIViewController *right = ...;

    UINavigationController *leftNav = [[UINavigationController alloc] init];
    UINavigationController *rightNav = [[UINavigationController alloc] init];
    [leftNav pushViewController:left animated:NO];
    [rightNav pushViewController:right animated:NO];
    [left release]; [right release];   // if they were alloc'ed above, that is

    UISplitViewController *svc = [[UISplitViewController alloc] init];          // “leak”
    svc.viewControllers = [NSArray arrayWithObjects:leftNav, rightNav, nil];
    [leftNav release]; [rightNav release];

    [window addSubview:svc.view];
    [window makeKeyAndVisible];
    return YES;
}
```

UISplitViewController



Who puts this button here?

And makes this popover appear?

UISplitViewController

- You are responsible for putting up a bar button
- You do it in an implementation of a split view delegate method

  It gets called when device rotation causes the left-hand view to be hidden

```
- (void)splitViewController:(UISplitViewController*)sv
       willHideViewController:(UIViewController *)aViewController
            withBarButtonItem:(UIBarButtonItem*)barButtonItem
         forPopoverController:(UIPopoverController*)pc;
```

- Notice that the delegate method gives a bar button to use

  So you don't have to create it, nor `release` it (`retain` it if you want to keep a pointer to it though).

  Just add it to the UI (easy if you're in a navigation controller).

- It's cool to have any "potential" right-hand view implement this

  It's so easy to implement, you might as well.

  Even if you're not thinking of using it in a split view in your current UI.

UISplitViewController

```
- (void)splitViewController:(UISplitViewController*)sv
     willHideViewController:(UIViewController *)aViewController
          withBarButtonItem:(UIBarButtonItem*)barButtonItem
       forPopoverController:(UIPopoverController*)popover
{
    barButtonItem.title = aViewController.title;   // this is the title of the left-hand vc
    self.navigationItem.rightBarButtonItem = barButtonItem;
    // if we are not in a UINavigationController this method (appropriately) does nothing
}
```

UISplitViewController

```
- (void)splitViewController:(UISplitViewController*)sv
     willHideViewController:(UIViewController *)aViewController
          withBarButtonItem:(UIBarButtonItem*)barButtonItem
       forPopoverController:(UIPopoverController*)popover
{
    barButtonItem.title = aViewController.title;   // this is the title of the left-hand vc
    self.navigationItem.rightBarButtonItem = barButtonItem;
    // if we are not in a UINavigationController this method (appropriately) does nothing
}
```

UISplitViewController

```
- (void)splitViewController:(UISplitViewController*)sv
      willHideViewController:(UIViewController *)aViewController
           withBarButtonItem:(UIBarButtonItem*)barButtonItem
        forPopoverController:(UIPopoverController*)popover
{
    barButtonItem.title = aViewController.title;   // this is the title of the left-hand vc
    self.navigationItem.rightBarButtonItem = barButtonItem;
    // if we are not in a UINavigationController this method (appropriately) does nothing
}
```

This @property on UIViewController
sets attributes of the UINavigationController UI
that will appear when our UIViewController
is the top "card on the stack."

In this case, we're setting the button that is
on the right on the navigation controller's bar.

UISplitViewController

```
- (void)splitViewController:(UISplitViewController*)sv
      willHideViewController:(UIViewController *)aViewController
           withBarButtonItem:(UIBarButtonItem*)barButtonItem
        forPopoverController:(UIPopoverController*)popover
{
    barButtonItem.title = aViewController.title;   // this is the title of the left-hand vc
    self.navigationItem.rightBarButtonItem = barButtonItem;
    // if we are not in a UINavigationController this method (appropriately) does nothing
}
```

Usually we don't want to futz with this because the left view is in it and we don't know anything about that view.

UISplitViewController

```
- (void)splitViewController:(UISplitViewController*)sv
     willHideViewController:(UIViewController *)aViewController
          withBarButtonItem:(UIBarButtonItem*)barButtonItem
       forPopoverController:(UIPopoverController*)popover
{
    barButtonItem.title = aViewController.title;  // this is the title of the left-hand vc
    self.navigationItem.rightBarButtonItem = barButtonItem;
    // if we are not in a UINavigationController this method (appropriately) does nothing
}
- (void)splitViewController:(UISplitViewController*)sv
     willShowViewController:(UIViewController *)aViewController
  invalidatingBarButtonItem:(UIBarButtonItem*)barButtonItem
{
    self.navigationItem.rightBarButtonItem = nil;
}
```

Here we are going back to landscape.
So we don't want that bar button anymore.

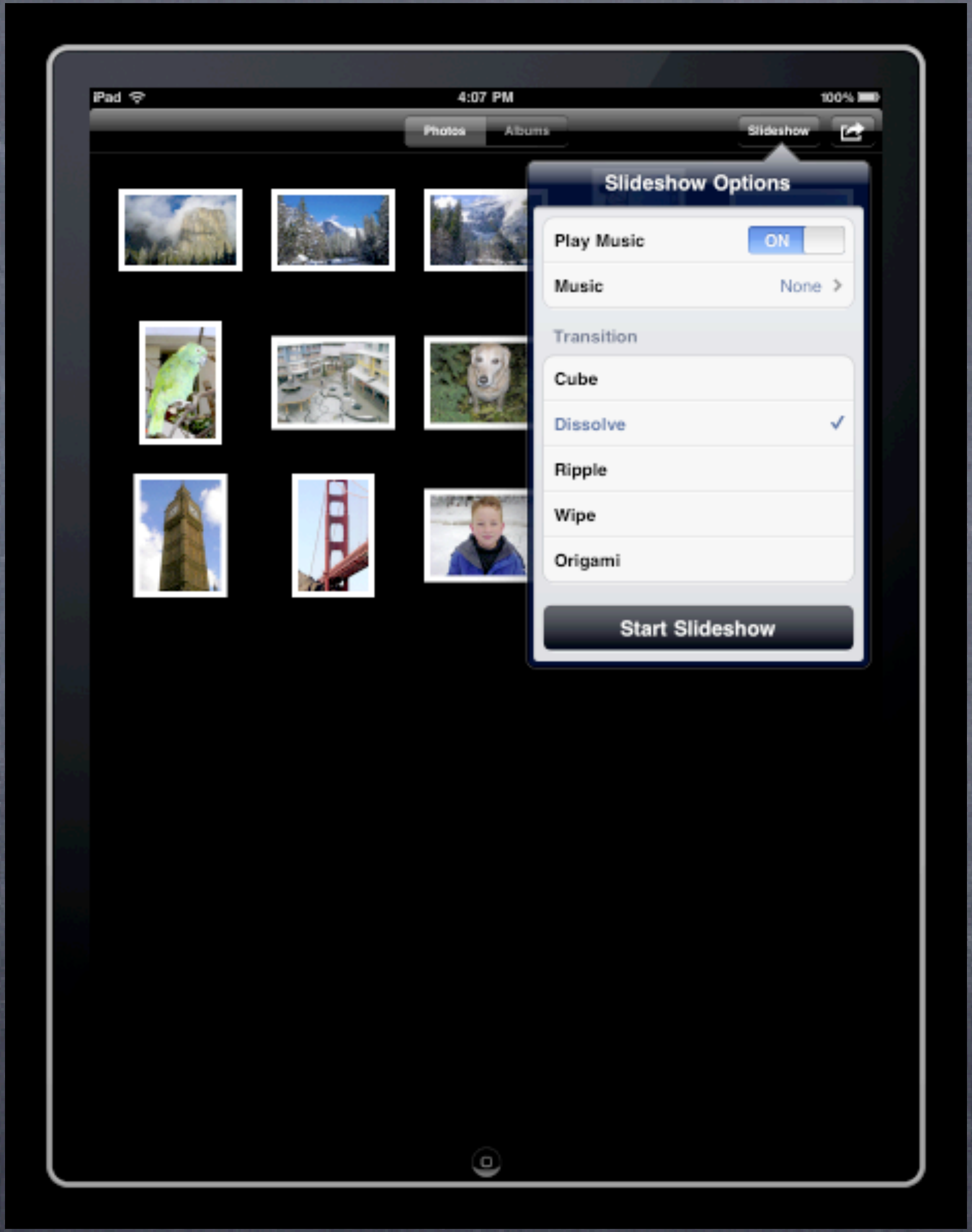
UISplitViewController

```
- (void)splitViewController:(UISplitViewController*)sv
     willHideViewController:(UIViewController *)aViewController
          withBarButtonItem:(UIBarButtonItem*)barButtonItem
       forPopoverController:(UIPopoverController*)popover
{
    barButtonItem.title = aViewController.title;   // this is the title of the left-hand vc
    self.navigationItem.rightBarButtonItem = barButtonItem;
    // if we are not in a UINavigationController this method (appropriately) does nothing
}
- (void)splitViewController:(UISplitViewController*)sv
     willShowViewController:(UIViewController *)aViewController
  invalidatingBarButtonItem:(UIBarButtonItem*)barButtonItem
{
    self.navigationItem.rightBarButtonItem = nil;
}
- (void)splitViewController:(UISplitViewController*)sv
     popoverController:(UIPopoverController *)popover
  willPresentViewController:(UIViewController *)leftViewController
{
    // do nothing here because we don't "know about" the left view controller
}
```

UIPopoverController

- UIPopoverController is NOT a UIViewController subclass
  Don't be confused by its name.
- Pops up on screen from "active areas"
  Buttons in tool bars (like the UISplitViewController case we just went through).
  Text selections (like cut/copy/paste or other text-related operations).
  Other buttons or active areas on screen.
- Sometimes you might want the popover in portrait <u>&</u> landscape
  In this case, you would NOT use a UISplitViewController, just a UIPopoverController

UIPopoverController



Slideshow Options

Play Music ON

Music None

Transition

Cube

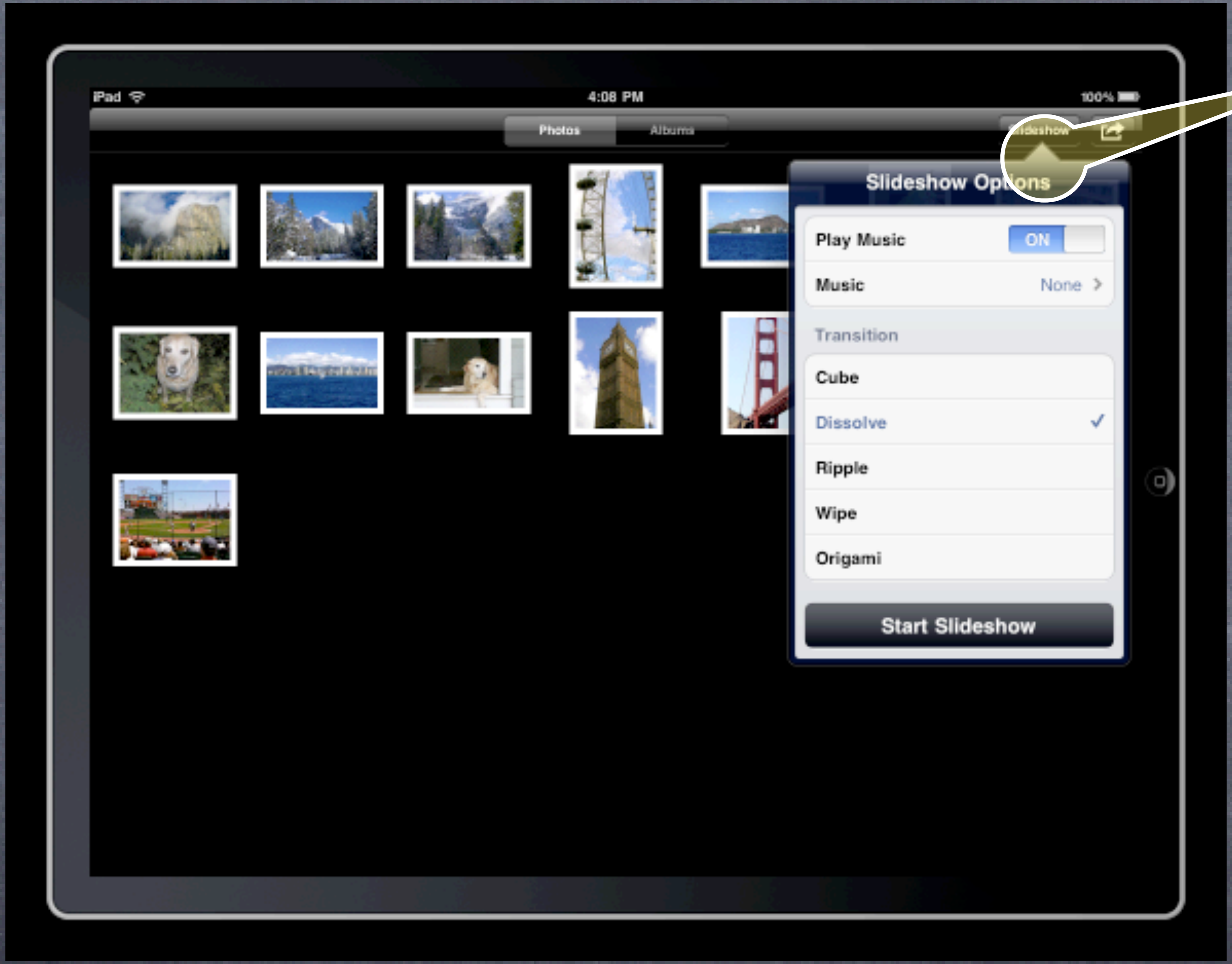
Dissolve ✓

Ripple

Wipe

Origami

UIPopoverController



Slideshow Options

Play Music ON

Music None >

Transition

Cube

Dissolve ✓

Ripple

Wipe

Origami

Start Slideshow

This is the popover's "arrow."

This popover is popping with an arrow direction `UIPopoverArrowDirectionUp`

UIPopoverController

- Designated initializer takes the UIViewController to display in it

```
UIViewController *vc = ...;
UIPopoverController *popover = [[UIPopoverController] alloc] initWithContentViewController:vc];
```

The argument (vc) cannot be nil.
You must specify a UIViewController for the popover to contain at initialization time.

You can change it with this property later (we rarely do that).

```
@property (retain) UIViewController *contentViewController;
```

- Then just send this message to get the popover to appear

Specify the rectangle where the popover's little "arrow" should point to.
Also restrict which ways that little "arrow" can point (and thus where the popover can appear).

```
[popover presentPopoverFromRect:(CGRect)aRect        // e.g. the rectangle containing selected text
                         inView:(UIView *)aView      // e.g. the text view the selected text is in
       permittedArrowDirections:(UIPopoverArrowDirection)direction
                       animated:(BOOL)animated];
```

The direction can be UIPopoverArrowDirectionUp/Down/Left/Right/Any.

UIPopoverController

- Or, if it's a bar button that's popping it up, use this method ...

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

UIPopoverController

- How to set the size of the popover

  Either send a message to the popover setting the size (will animate if on screen) …

  ```
  [popover setPopoverContentSize:(CGSize)aSize animated:(BOOL)animated];
  ```

  Or implement this method in the popover view's `UIViewController` to return the size …

  ```
  - (CGSize)contentSizeForViewInPopover;
  ```

  The latter is recommended (more object-oriented).

UIPopoverController

- How to dismiss the popover from the screen

  User clicks anywhere outside the popover.

  Or you can dismiss it programmatically by sending the popover this method …

  ```
  [popover dismissPopoverAnimated:(BOOL)animated];
  ```

- How to tell if the popover is currently on the screen

  ```
  if ([popover.isPopoverVisible]) ...
  ```

- Popover has a delegate to prevent (and get notified of) dismissal

  ```
  - (BOOL)popoverControllerShouldDismissPopover:(UIPopoverController *)popover;
  - (void)popoverControllerDidDismissPopover:(UIPopoverController *)popover;
  ```

  These are only sent to the delegate if the user initiates the dismissal (i.e. not programmatically).

Universal Applications

- A "Universal" application will run on iPhone or iPad

  Single binary image built using the iOS 4 SDK.

  Even though iPad is running iOS 3.2, this can work if you take care.

- Requires the Build setting "iOS Deployment Target"

  Find your Application executable in the Targets folder of the Groups & Files section of Xcode.

  Right click on it and choose Get Info.

  In the Build tab, scroll down to the iOS Deployment Target property and set it to iOS 3.2.

  Setting it to iOS 3.2 means "this application is allowed to run on iOS 3.2 or later."

- How to create one

  In an existing iOS 4 project, select your Application (in the Targets folder described above).

  Choose "Upgrade Current Target for iPad ..." from the Project menu.

  The name is misleading ... choose "One Universal application" from the dialog that comes up.

Universal Applications

- There is still work to do to make an application "Universal"
- If you call API that is new in iOS 4, it has to be conditional

  For example, you might access the property contentScaleFactor (new for retina display in iOS4).

  You would want to use introspection to be sure UIView responds to that first.

```
- (void)drawRect:(CGRect)aRect
{
    CGFloat scaleFactor = 1.0;
    if ([self respondsToSelector:@selector(contentScaleFactor)]) {
        scaleFactor = self.contentScaleFactor;
    }
    // use scaleFactor here
}
```

- Can make conditional choices depending on what CofC you're in

  (CofC means "Controller of Controllers")

  `if (self.splitViewController != nil) ...`

Universal Applications

- Or whether you are on screen

  There is room for more view controllers to be on screen at the same time on the iPad.
  E.g. left and right split view versus views that appear one at a time via navigation controller.
  So the setup in `application:didFinishLaunchingWithOptions:` will be different.
  Then later on, you can make decisions based on whether a view is on screen.
  A simple way to do that is to check a `UIView`'s `window` property. If `nil`, then it's off screen.
  For example, this code snippet might appear in a `UIViewController` somewhere ...

  ```
  if (self.view.window) ...
  ```

- How big is the current screen?

  ```
  CGRect screenBounds = [[UIScreen mainScreen] bounds];  // in points
  ```

  Probably wouldn't want to check an exact size here, but maybe a threshold?

Universal Applications

- If the above don’t work, it is possible to ask “am I on an iPad?”

  Here’s how ...

  ```
  BOOL iPad = (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad);
  ```

  This is not a first-resort. Use the “right” conditional before using this.

- Conditionally loading a .xib

  Remember that UIViewController’s designated initializer lets you specify the name of its .xib.

  So you could pick a different .xib depending on which device you are on.

  ```
  NSString *xibName = iPad ? @“MyNib-iPad” : @“MyNib”;
  UIViewController *vc = [[UIViewController alloc] initWithNibName:xibName bundle:nil];
  ```

  ... instead of ...

  ```
  UIViewController *vc = [[UIViewController alloc] init];
  ```

- application:didFinishLaunchingWithOptions:

  You will very likely be building a different base user-interface here.

Universal Applications

```
- (void)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)options
{
   UIViewController *master = ...;
   UIViewController *detail = ...;
   master.subservientController = detail; // master pushes detail on iPhone only
```

We've made master have a property to hold an instance of the view controller it pushes (detail).

Universal Applications

```
- (void)application:(UIApplication *)application
        didFinishLaunchingWithOptions:(NSDictionary *)options
{
   UIViewController *master = ...;
   UIViewController *detail = ...;
   master.subservientController = detail; // master pushes detail on iPhone only
   UINavigationController *masterNav = [[UINavigationController alloc] init];
   [masterNav pushViewController:master animated:NO];

   if (iPad) {
      UINavigationController *detailNav = [[UINavigationController alloc] init];
      [detailNav pushViewController:detail animated:NO];
      UISplitViewController *splitVC = [[UISplitViewController alloc] init];
      splitVC.delegate = detail;  // note: detail implements UISplitViewControllerDelegate
      splitVC.viewControllers = [NSArray arrayWithObjects:masterNav, detailNav, nil];
      [masterNav release]; [detailNav release];
```

On the iPad, the detail view controller will already be on screen in the right hand side of the split view so we'll have to make sure not to push it in our master code by checking whether detail is on screen at push time.

Universal Applications

```
- (void)application:(UIApplication *)application
         didFinishLaunchingWithOptions:(NSDictionary *)options
{
   UIViewController *master = ...;
   UIViewController *detail = ...;
   master.subservientController = detail; // master pushes detail on iPhone only
   UINavigationController *masterNav = [[UINavigationController alloc] init];
   [masterNav pushViewController:master animated:NO];

   if (iPad) {
      UINavigationController *detailNav = [[UINavigationController alloc] init];
      [detailNav pushViewController:detail animated:NO];
      UISplitViewController *splitVC = [[UISplitViewController alloc] init];
      splitVC.delegate = detail;  // note: detail implements UISplitViewControllerDelegate
      splitVC.viewControllers = [NSArray arrayWithObjects:masterNav, detailNav, nil];
      [masterNav release]; [detailNav release];
      rootVC = splitVC;      // rootVC is an instance variable
   } else {
      rootVC = masterNav;      // master will push detail on iPhone when appropriate
   }

   [window addSubview:rootVC.view];
}
```

Universal Applications

- How to run a Universal Application in the iPad Simulator 3.2

  There is a pull-down in the upper left corner of Xcode's main window.

  From there you can switch to iPad Simulator 3.2 (from iPhone Simulator 4.0).

  If that option is not there, you probably forgot to set iOS Deployment Target to iOS 3.2.

Next Time

- Gesture Recognizers

  How your view can handle touches

- Demo

  Universal Application

  `UISplitViewController`

  Handling device rotation (`shouldAutorotate...` and springs and struts)