Stanford CS193p

Developing Applications for iOS
Fall 2011

Today

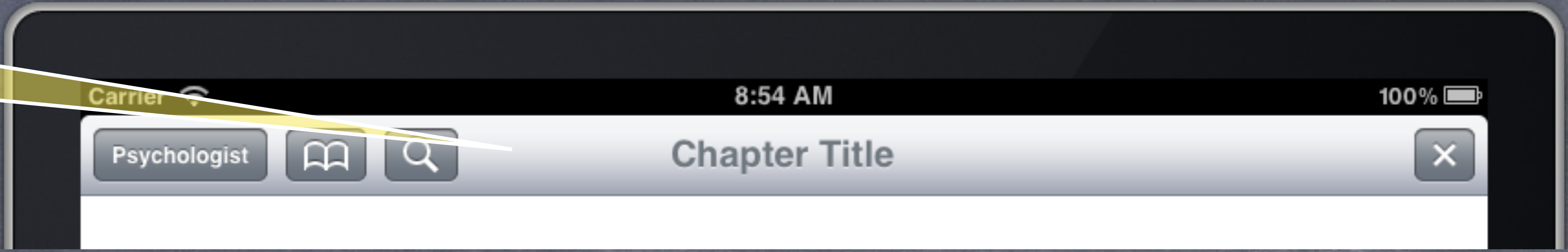
- UI Element of the Week
  - UIToolbar
- iPad
  - Split View
  - Popover
  - Universal (iPhone + iPad) Application
  - Demo
- Friday Section
  - AVFoundation framework - Capturing and manipulating images.

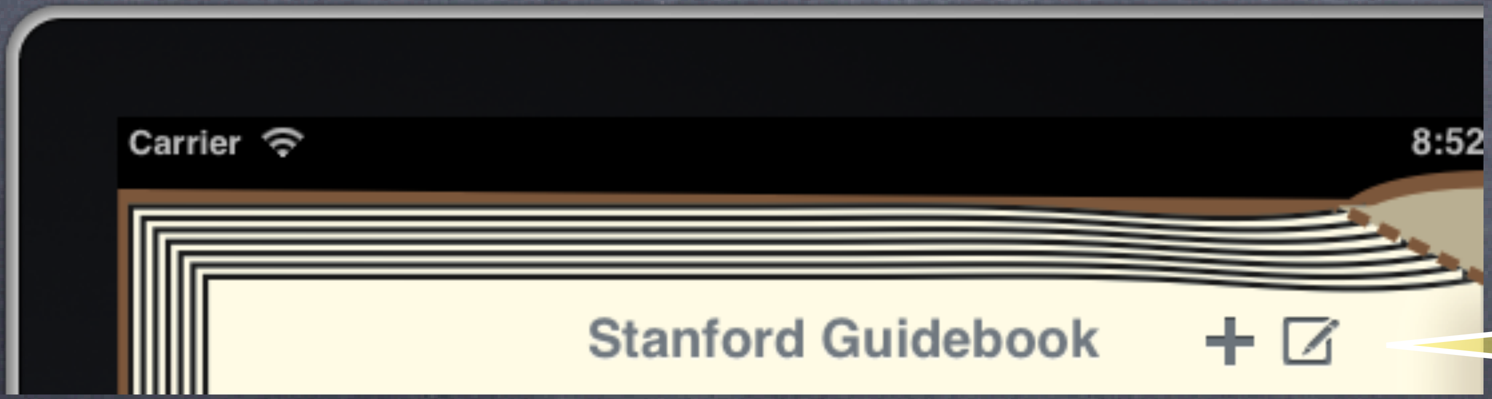
UIToolbar

- Collection of UIBarButtonItems

  Just drag a UIToolbar out into your view (usually at the top or bottom of it).
  Then drag UIBarButtonItems into the toolbar and wire up outlets/actions to/from them.
  Has a default "steel" UI, but can be customized using bar style, background image, et. al.

Standard toolbar UI.





Custom toolbar with a background image which is a .png of a yellow square.

UIToolbar

- UINavigationController's Toolbar

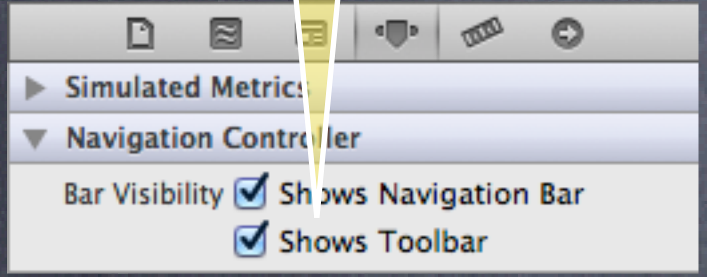
  One also appears at the bottom of a UINavigationController if its toolbarHidden @property = NO

  (then you set each UIViewController's toolbarItems @property to control which buttons

  (toolbarItems is an NSArray of UIBarButtonItems)

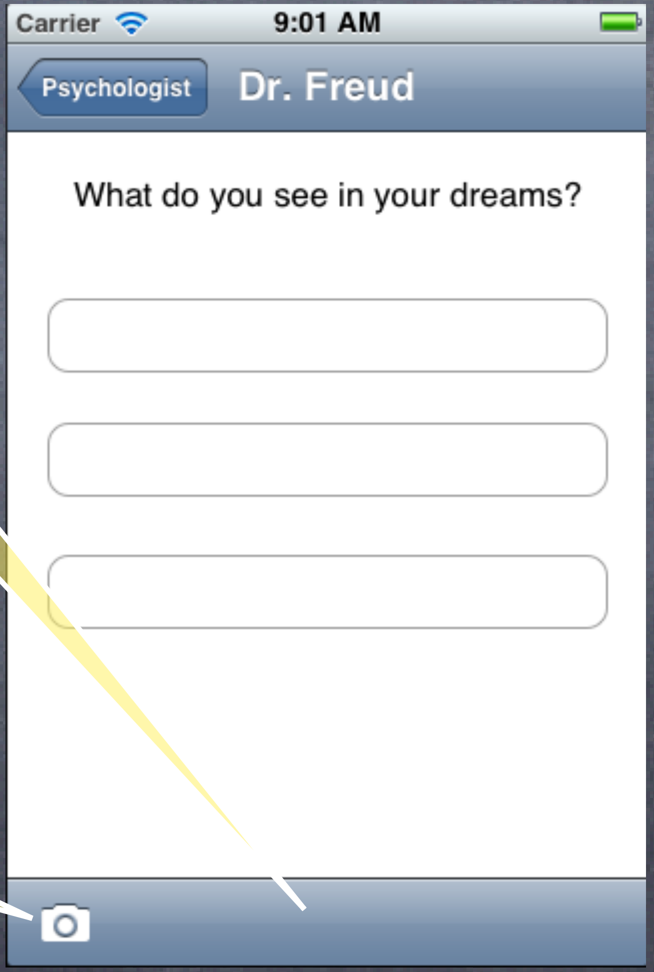
  Default value of toolbarHidden is YES.

Switch to turn on Toolbar in Navigation Controller in Xcode



Bar Visibility ☑ Shows Navigation Bar
☑ Shows Toolbar

Navigation Controller Toolbar



PsychologistViewController's toolbarItems array contains a UIBarButtonSystemItemCamera.

Stanford CS193p
Fall 2011

UIToolbar

- UINavigationController's Toolbar

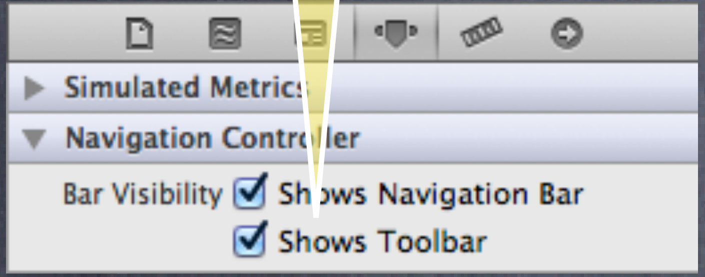
  One also appears at the bottom of a UINavigationController if its toolbarHidden @property = NO

  (then you set each UIViewController's toolbarItems @property to control which buttons

  (toolbarItems is an NSArray of UIBarButtonItems)

  Default value of toolbarHidden is YES.

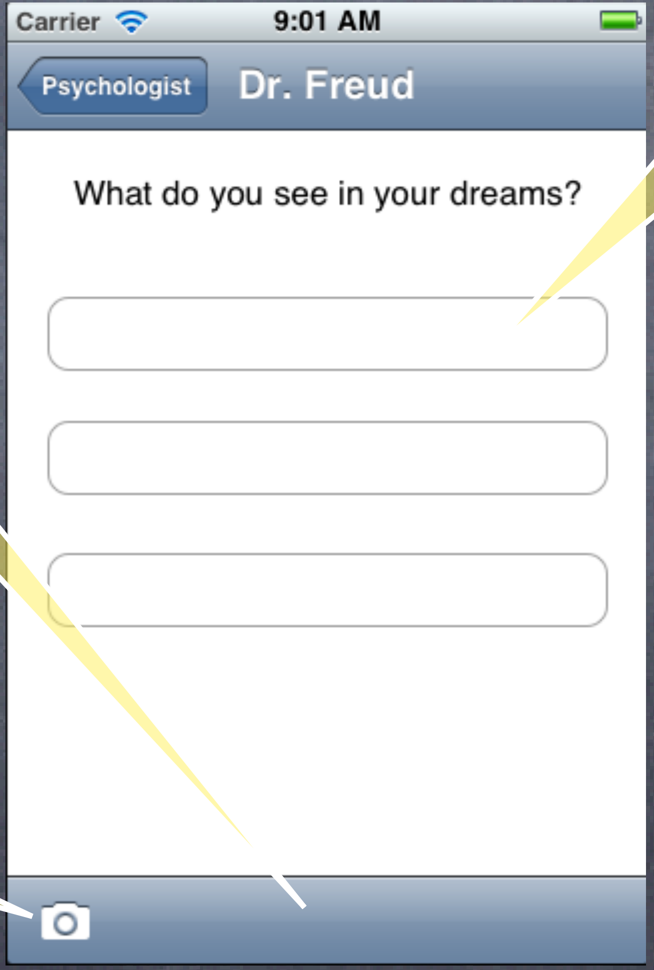
Switch to turn on Toolbar in Navigation Controller in Xcode



Bar Visibility ☑ Shows Navigation Bar
☑ Shows Toolbar

Navigation Controller Toolbar

Click Here



What do you see in your dreams?

PsychologistViewController's toolbarItems array contains a UIBarButtonSystemItemCamera.

UIToolbar

- UINavigationController's Toolbar

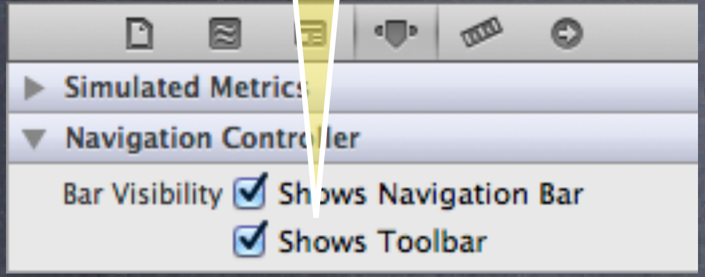
  One also appears at the bottom of a UINavigationController if its toolbarHidden @property = NO

  (then you set each UIViewController's toolbarItems @property to control which buttons

  (toolbarItems is an NSArray of UIBarButtonItems)

  Default value of toolbarHidden is YES.

Switch to turn on Toolbar in Navigation Controller in Xcode

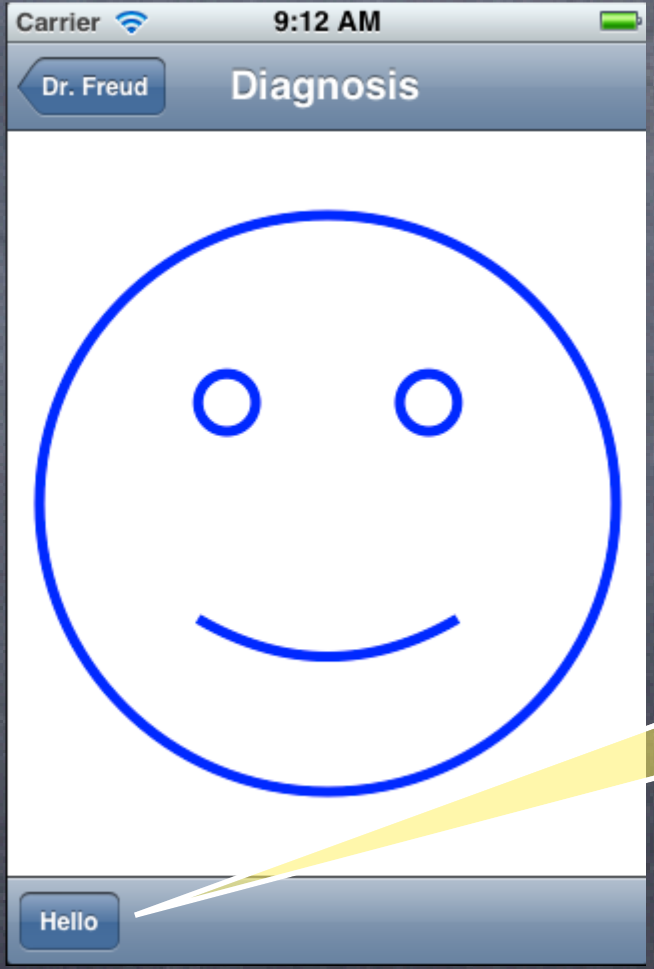


Simulated Metrics

Navigation Controller

Bar Visibility ☑ Shows Navigation Bar

☑ Shows Toolbar



HappinessViewController's toolbarItems array contains a Bordered button with the title "Hello".

UIToolbar

- UINavigationController's Toolbar

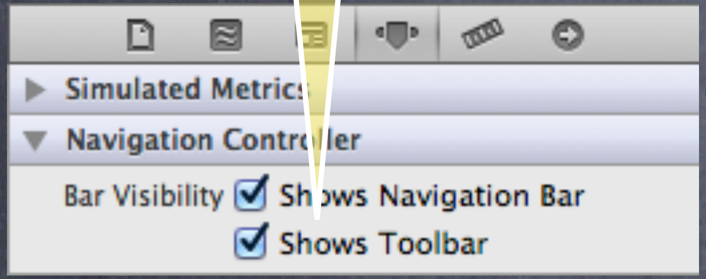
  One also appears at the bottom of a UINavigationController if its toolbarHidden @property = NO

  (then you set each UIViewController's toolbarItems @property to control which buttons

  (toolbarItems is an NSArray of UIBarButtonItems)

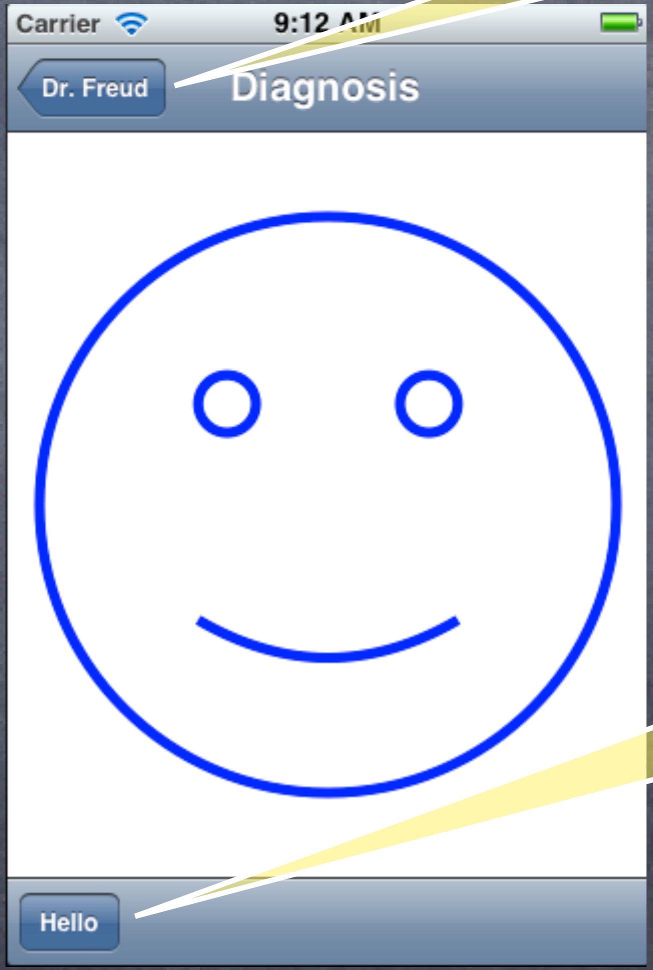
  Default value of toolbarHidden is YES.

Switch to turn on Toolbar in Navigation Controller in Xcode



Bar Visibility ✓ Shows Navigation Bar
✓ Shows Toolbar

Click Back



HappinessViewController's toolbarItems array contains a Bordered button with the title "Hello".

UIToolbar

- UINavigationController's Toolbar

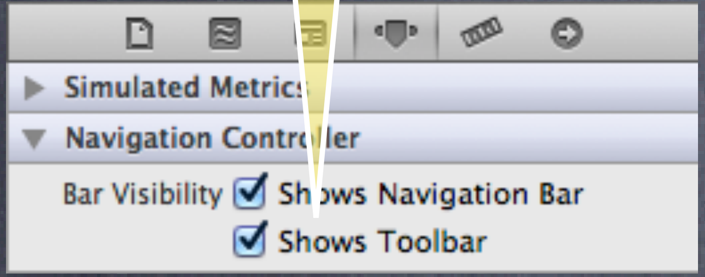
  One also appears at the bottom of a UINavigationController if its toolbarHidden @property = NO

    (then you set each UIViewController's toolbarItems @property to control which buttons

    (toolbarItems is an NSArray of UIBarButtonItems)

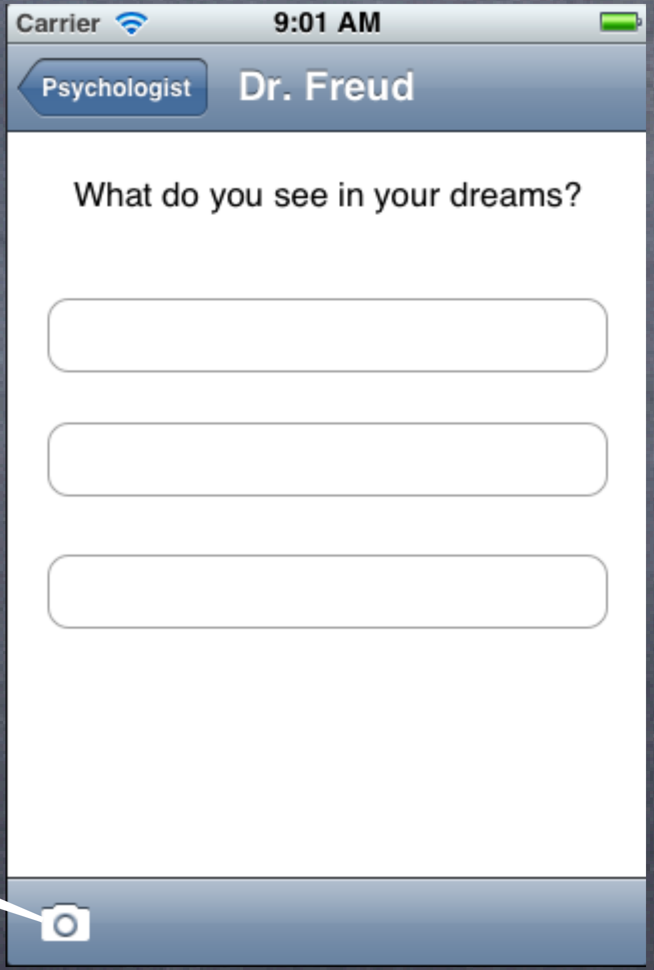
  Default value of toolbarHidden is YES.

Switch to turn on Toolbar in Navigation Controller in Xcode



Bar Visibility ☑ Shows Navigation Bar
☑ Shows Toolbar

Camera is back.



UIToolbar

- UIBarButtonItem

  Usually dragged out in Xcode, but can be created with various alloc/init methods.

  Target/Action like UIButton

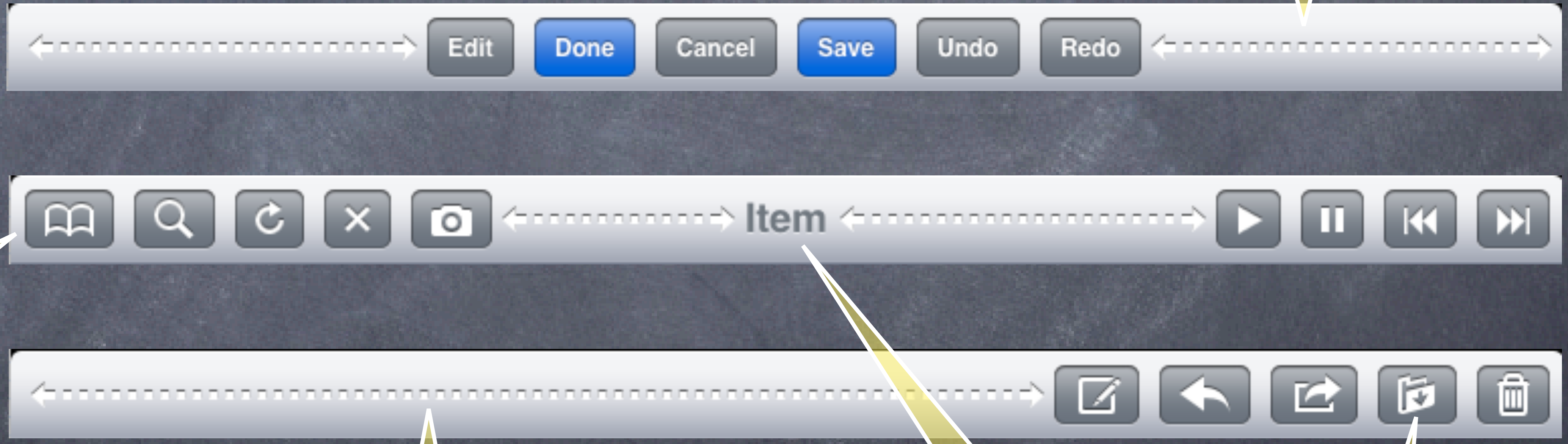
  Bordered or Plain

  Title or Image (or Custom View) or use Built-in System Items:

  Fixed and Flexible Space Items

Centering with Flexible Spaces

All these buttons are System Items except Item in the middle.



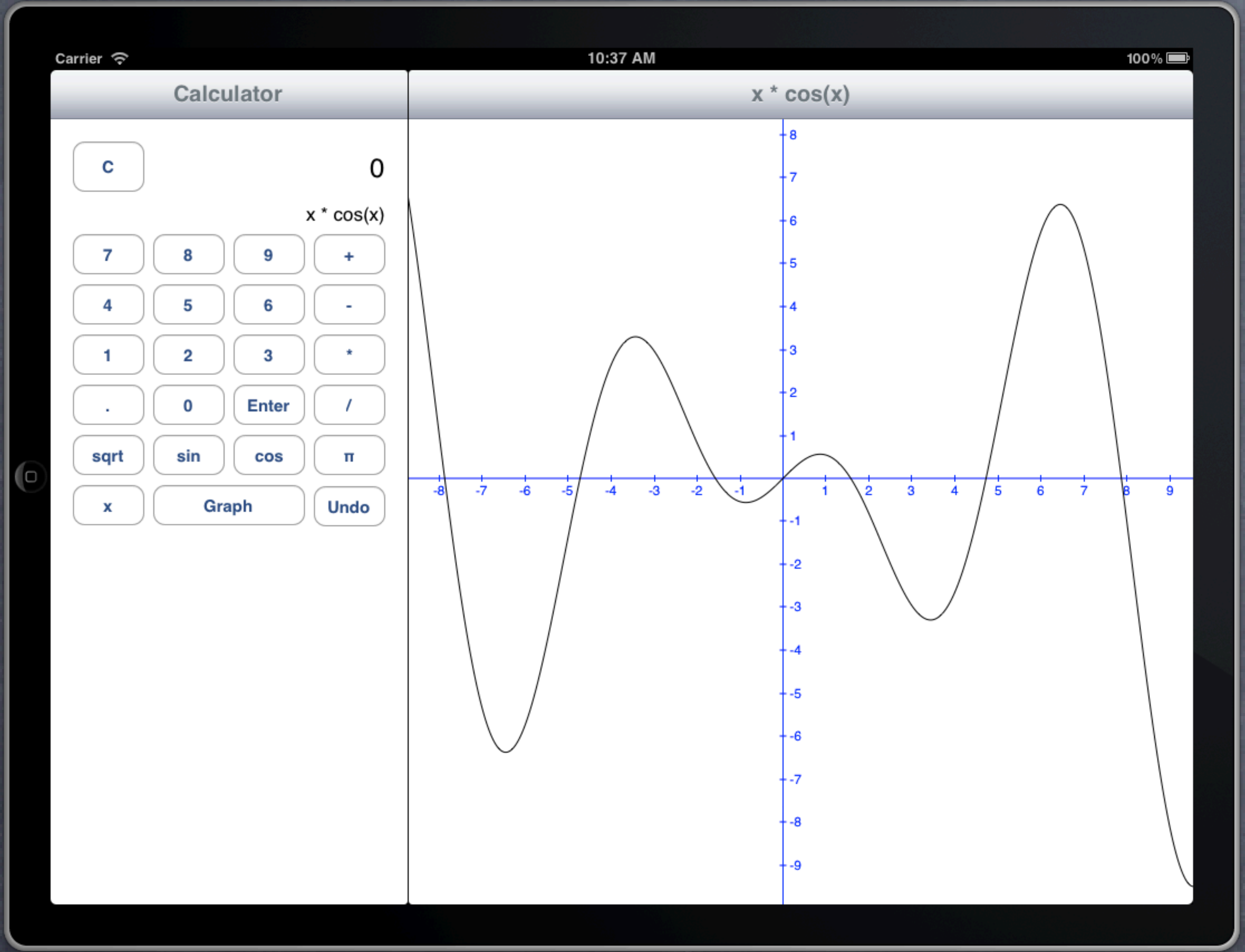
Using Flexible Space to Center an Item with other buttons left and right

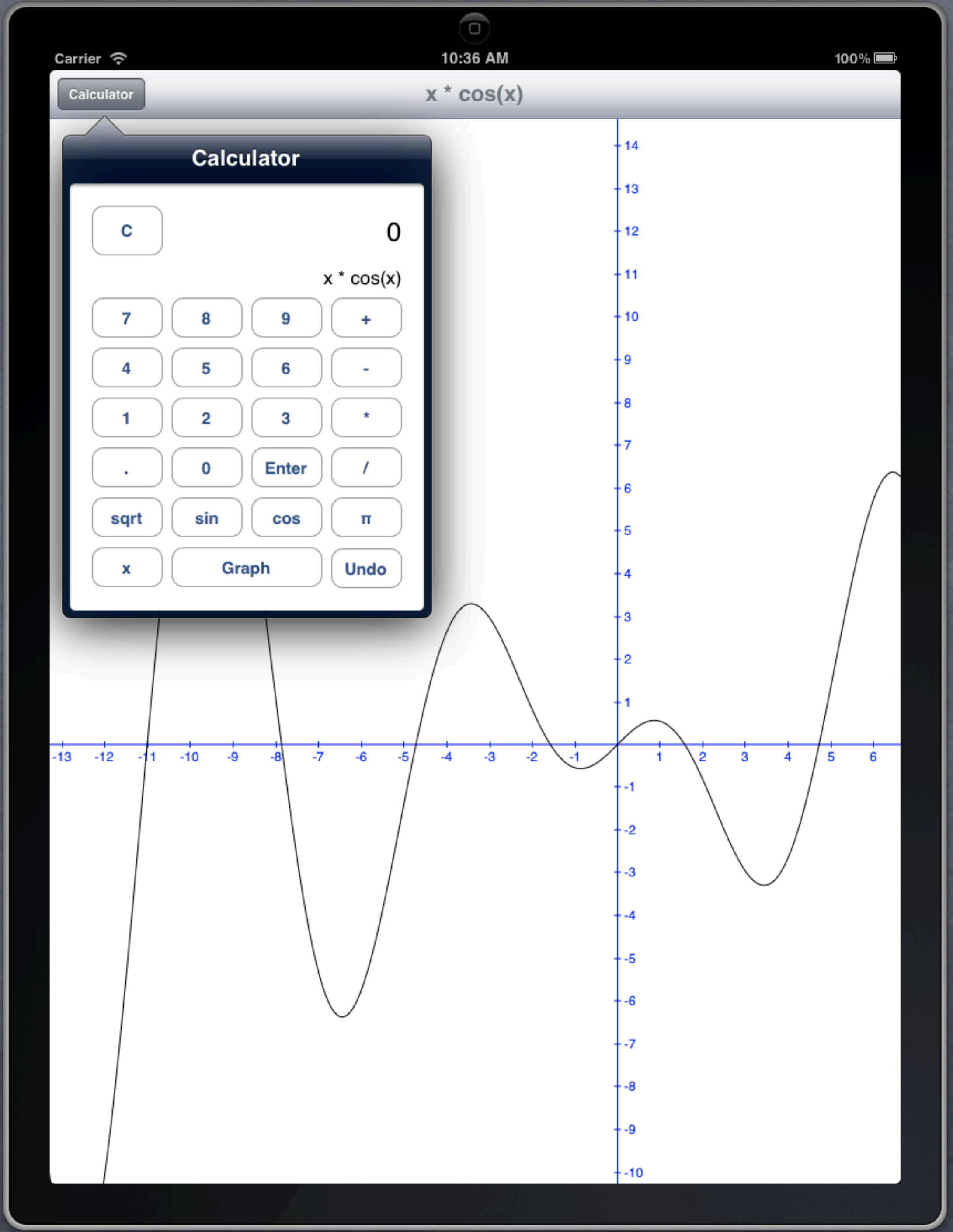
Using Flexible Space to Right Align

Plain Style

Bordered Style

UISplitViewController





UISplitViewController

- Just drag one out into an iPad Storyboard (only)

  If you don't see a Split View Controller in the Object Library, your storyboard is not iPad-style.
  You select which style of storyboard you have when you first create it (via New File ... menu item).

- Split view is a base UI element only

  It only makes sense for the Split View to be the initial view controller of your application.
  You would (probably) never embed one in another view controller.
  You would only ever have one split view controller in your application.

- Setting/Getting the two view controllers

  Both view controllers are almost always set with ctrl-drag in Xcode.
  `@property (nonatomic, copy) NSArray *viewControllers;` // 0 is left (master), 1 is right (detail)
  It doesn't really make sense for this array to have anything but 2 `UIViewControllers` in it.
  Note that this is not a `readonly @property`.
  But also note that it forces you to set/get the both at the same time.
  It is `copy` so that you don't pass a mutable array and then try to modify it afterwards.

UISplitViewController

- Setting the Split View's delegate @property
  You must do this. Otherwise, the left side will be inaccessible in portrait mode.
  You usually do this in UIViewController method viewDidLoad (we'll talk about this method later).
  Either the master or the detail view controller may be the delegate depending on the situation.
  Three different options for how to deal with rotation ...

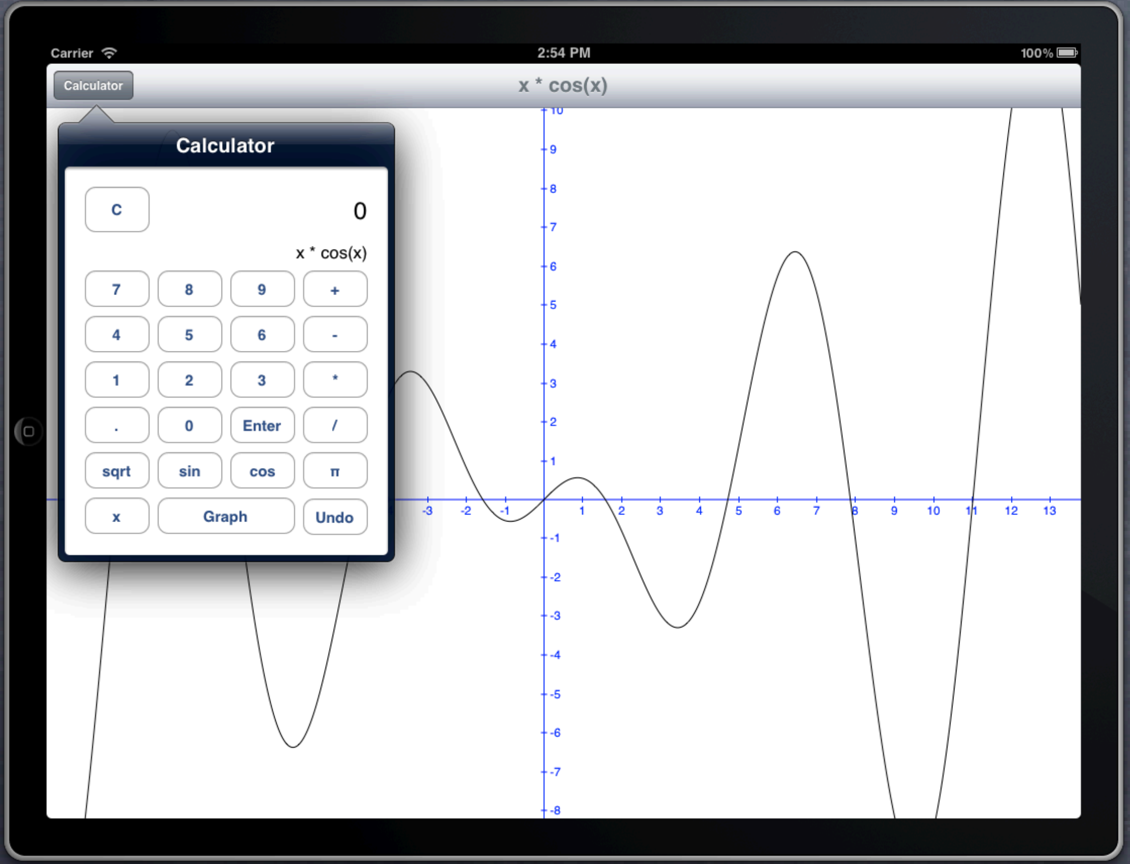
- The delegate controls the split view's visual appearance
  `@property (nonatomic, assign) id <UISplitViewControllerDelegate> delegate;`
  The right side is always visible in a split view (we call it the "detail" view of the split view).
  The split view's delegate is asked about when the left side (the "master") should be on screen.
  When the left side is not on screen, you are responsible for displaying a UIBarButtonItem to show it
  (you will be provided with a UIBarButtonItem to use in as an argument to the delegate method).

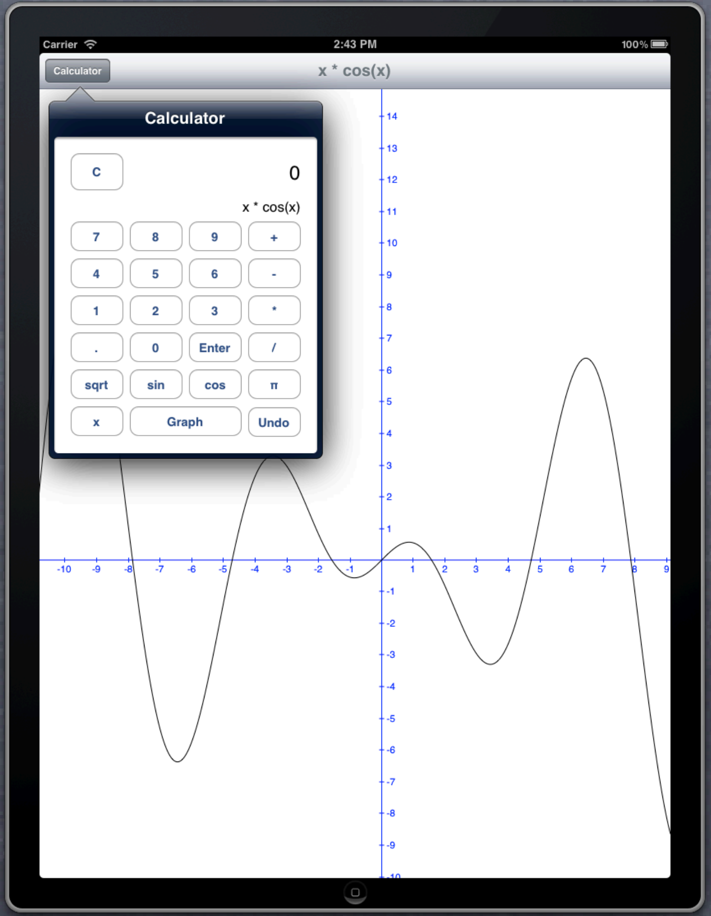
  Notice assign pointer type. Should probably be weak. assign means weak without auto-zeroing.
  This means you can get a dangling pointer (though, in practice, this is unlikely in this case).

UISplitViewController

- Always hide left side behind a bar button

```
- (BOOL)splitViewController:(UISplitViewController *)sender
    shouldHideViewController:(UIViewController *)master
               inOrientation:(UIInterfaceOrientation)orientation
{
    return YES; // always hide it
}
```

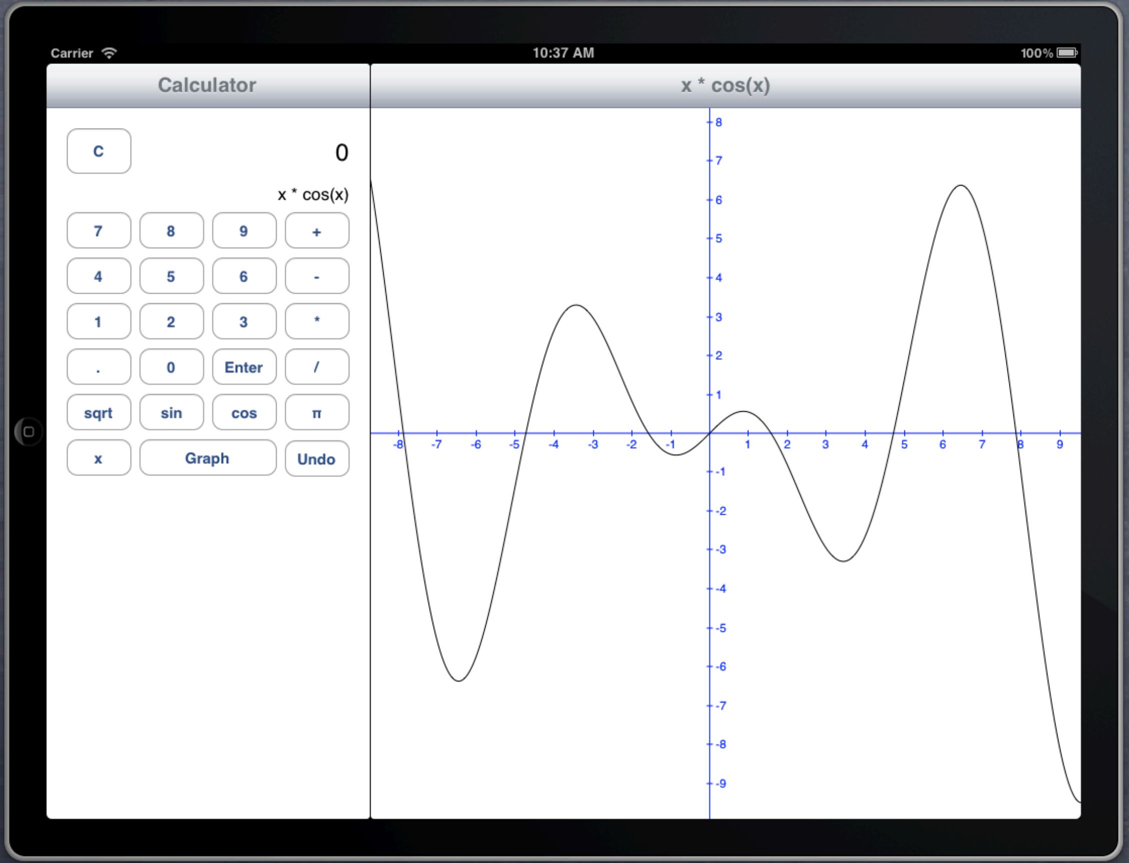


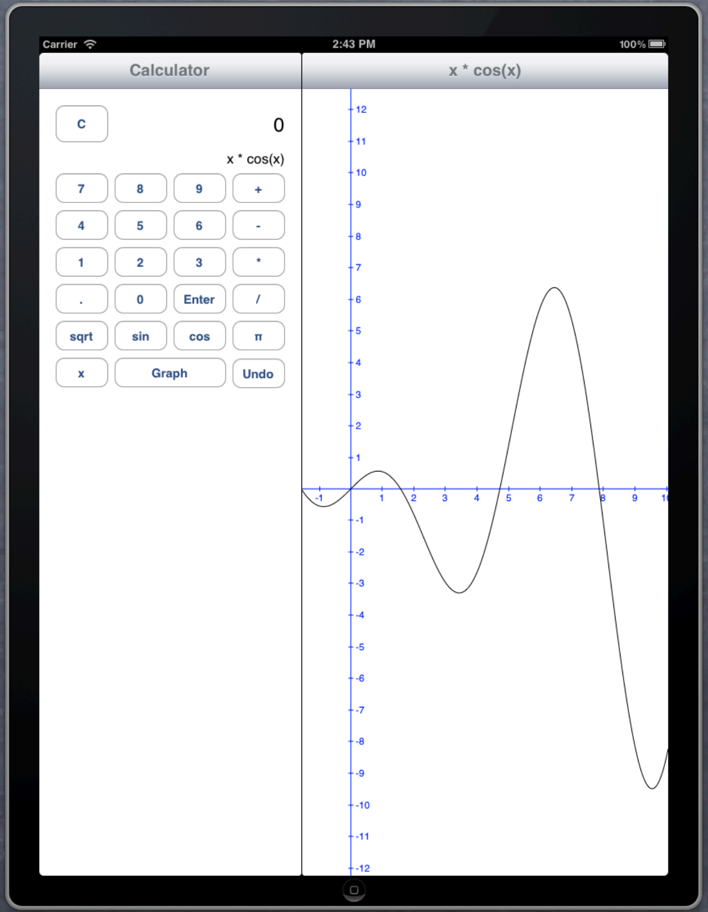


UISplitViewController

- Never hide left side behind a bar button

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

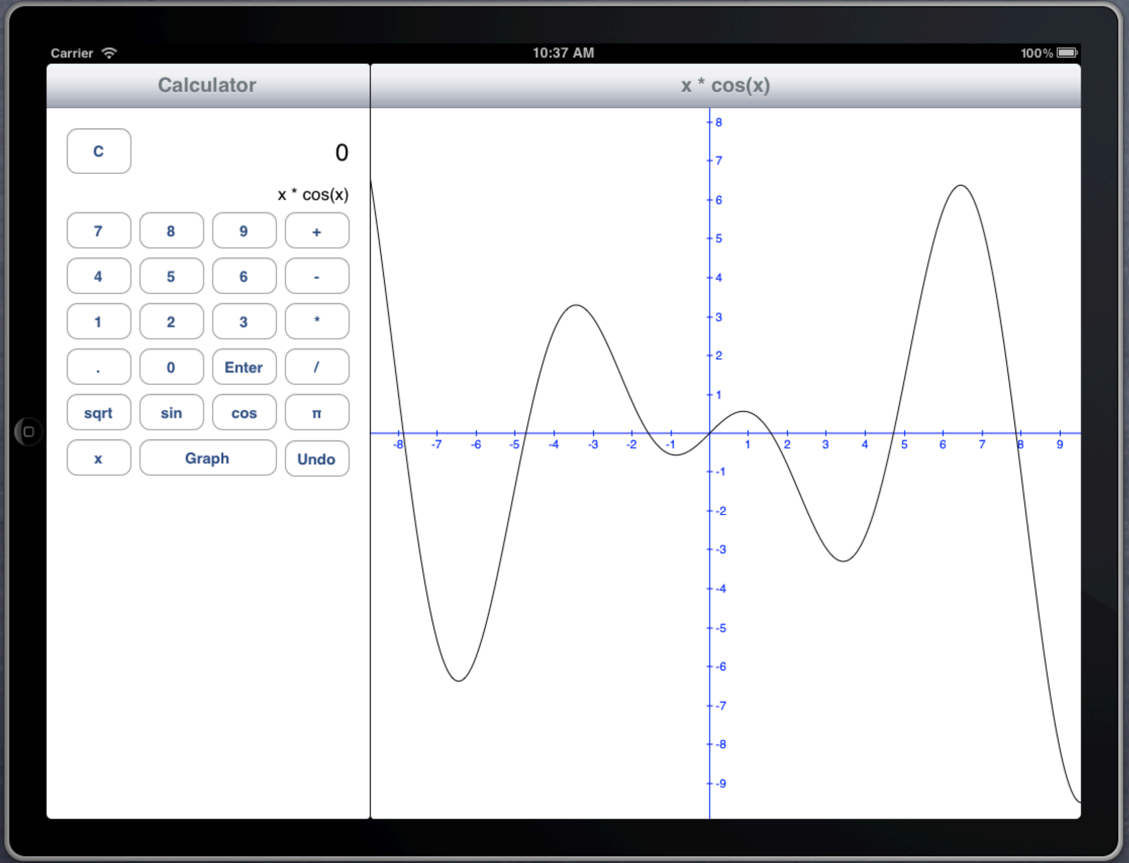


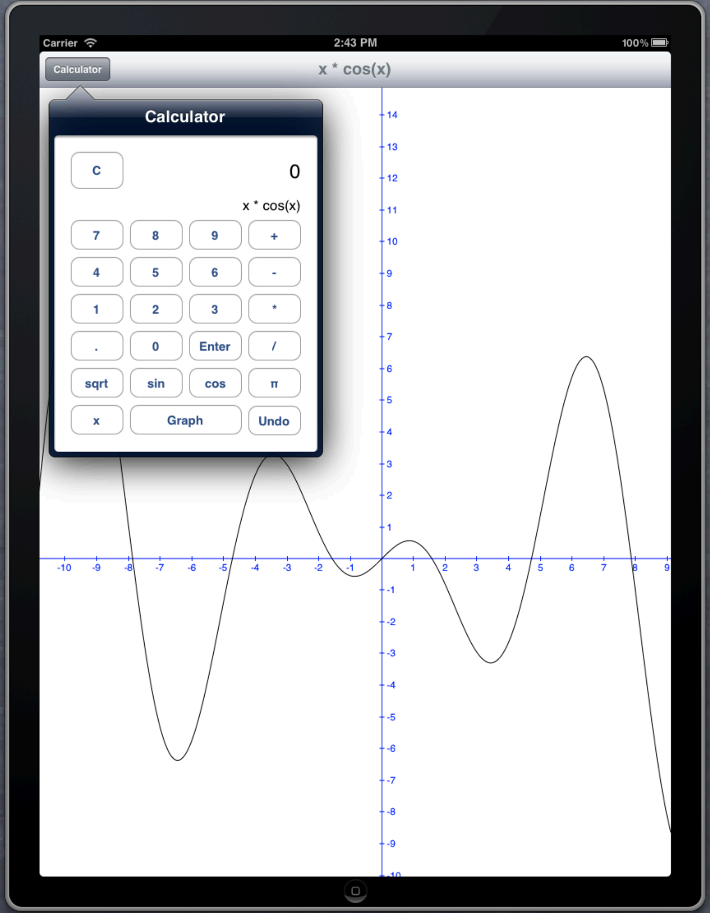


UISplitViewController

- Hide it in portrait orientation only

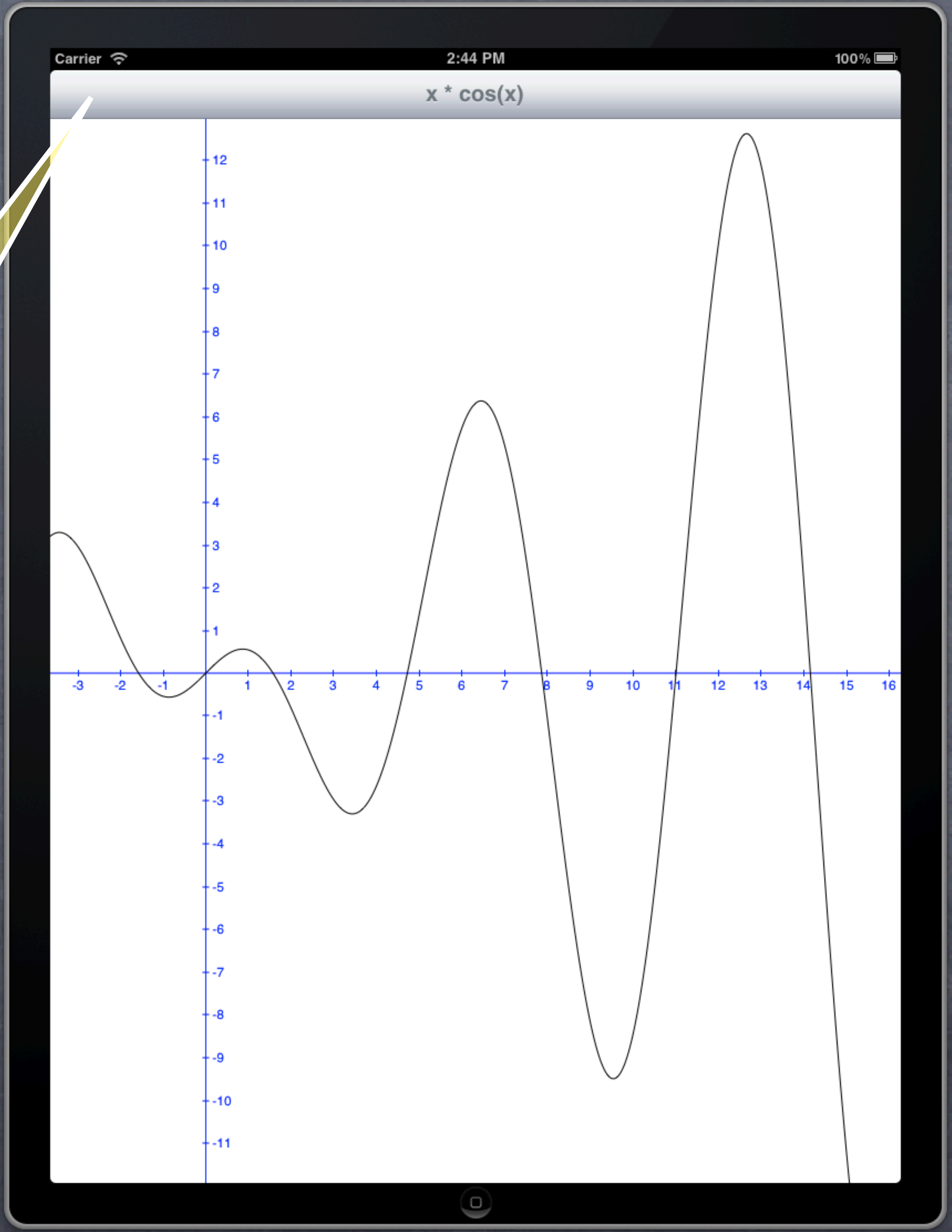
```
- (BOOL)splitViewController:(UISplitViewController *)sender
   shouldHideViewController:(UIViewController *)master
              inOrientation:(UIInterfaceOrientation)orientation
{
    return UIInterfaceOrientationIsPortrait(orientation);
}
```





UISplitViewController

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



No button to bring up the left side in a popover!

UISplitViewControllerDelegate

- Handling the bar button item ...

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"; // use a better word than "Master"!
    // setSplitViewBarButtonItem: must put the bar button somewhere on screen
    // probably in a UIToolbar or a UINavigationBar
    [detailViewController setSplitViewBarButtonItem:barButtonItem];
}
```

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

You have to implement this.

UISplitViewControllerDelegate

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

```
- (void)splitViewController:(UISplitViewController *)sender
     willShowViewController:(UIViewController *)master
  invalidatingBarButtonItem:(UIBarButtonItem *)barButtonItem
{
    // removeSplitViewBarButtonItem: must remove the bar button from its toolbar
    [detailViewController removeSplitViewBarButtonItem:nil];
}
```

UISplitViewControllerDelegate

- Typical "setSplitViewBarButton:" method

Example of using a UIToolbar to show the bar button item.
This both adds and removes because it stores the old bar button in a property
(so setSplitViewBarButtonItem:nil removes the existing one).

```
- (void)setSplitViewBarButtonItem:(UIBarButtonItem *)barButtonItem
{
    UIToolbar *toolbar = [self toolbar]; // might be outlet or calculated
    NSMutableArray *toolbarItems = [toolbar.items mutableCopy];
    if (_splitViewBarButtonItem) [toolbarItems removeObject:_splitViewBarButtonItem];
    // put the bar button on the left of our existing toolbar
    if (barButtonItem) [toolbarItems insertObject:barButtonItem atIndex:0];
    toolbar.items = toolbarItems;
    _splitViewBarButtonItem = barButtonItem;
}
```

UISplitViewController

- So how does the detail get updated when master changes?
  Target/Action or Segue
- Target/Action
  Example (code in the master view controller):

```
- (IBAction)doit
{
    id detailViewController = [[self.splitViewController viewControllers] lastObject];
    [detailViewController setSomeProperty:...];
}
```

- Segue for split view entirely replaces the detail view controller
  Remember, segues always instantiate a view controller (split view stops strongly 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 Replace the toolbar that you put the split view bar button into!
  Need to be sure to put it back into the newly instantiated view controller before it goes on screen.

UISplitViewController

- (Somewhat) general bar button replacement on Replace segue

  Implement delegate in the master: call `setSplitViewBarButtonItem:` in detail from delegate methods.
  Then transfer the bar button in `prepareForSegue:sender:` (along with updating the new detail VC).

```
- (id)splitViewDetailWithBarButtonItem
{
    id detail = [self.splitViewController.viewControllers lastObject];
    if ([detail respondsToSelector:@selector(setSplitViewBarButtonItem:)] &&
        [detail respondsToSelector:@selector(splitViewBarButtonItem)]) {
    }
    return nil;
}

- (void)transferSplitViewBarButtonItemToViewController:(id)destinationViewController
{
    UIBarButtonItem *splitViewBarButtonItem = [[self splitViewDetailWithBarButtonItem] splitViewBarButtonItem];
    [[self splitViewDetailWithBarButtonItem] setSplitViewBarButtonItem:nil];
    if (splitViewBarButtonItem) [destinationViewController setSplitViewBarButtonItem:splitViewBarButtonItem];
}

- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([segue.identifier isEqualToString:@"MyReplaceSegue"]) {
        [self transferSplitViewBarButtonItemToViewController:segue.destinationViewController];
        [segue.destinationViewController setSomeProperty:self.dataToTransferToDetail];
    }
}
```

Popover

- Not a `UIViewController`, but displays one inside a rectangle

  `@property (nonatomic, strong) UIViewController *contentViewController;`

  Also you can animate the changing of a popover that's already on screen with:

  `- (void)setContentViewController:(UIViewController *)vc animated:(BOOL)animated;`

- Usually the above property is ctrl-dragged in Xcode

  This creates a Popover segue.

  In your `prepareForSegue:sender:` method, the arg will be `isKindOf:UIStoryboardPopoverSegue`.

  The `UIStoryboardPopoverSegue` has a `- (UIPopoverController *)popoverController @property`.

- You can `alloc/initWithContentViewController:` one too

  Not very common, but not a bad thing.

  Segues are nicer because segues are essentially "documentation" of your UI in storyboards.

- Visible?

  You can tell whether a popover is currently visible with `- (BOOL)popoverVisible;`

Popover

- But you can also present a popover from code

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

```
- (void)presentPopoverFromRect:(CGRect)aRect or
                        inView:(UIView *)view
      permittedArrowDirections:(UIPopoverArrowDirection)direction
                      animated:(BOOL)flag;
```

or

```
- (void)presentPopoverFromBarButtonItem:(UIBarButtonItem *)barButtonItem
               permittedArrowDirections:(UIPopoverArrowDirection)direction
                               animated:(BOOL)flag;
```

- Don't forget to keep a strong pointer to the popover controller!

```
- (IBAction)presentPopover {
    UIPopoverController *pop = [[UIPopoverController alloc] initWithViewController:vc];
    [pop presentPopoverFromBarButtonItem:item permittedArrowDirections:…];
} // BAD! no strong pointer to the popover controller! presenting it is not enough!
```

Popover

- Dismissing a popover automatically

  Dismissed automatically if the user clicks outside of the popover.
  Unless the user clicks in one of the views set by this `UIPopoverController` property:
  `@property (nonatomic, copy) NSArray *passthroughViews;`

- Dismissing a popover from your code

  `- (void)dismissPopoverAnimated:(BOOL)animated;`
  You would call this when some interaction inside your popover's view controller results in dismissal.
  For example, the user presses "OK" or otherwise chooses something.
  Usually the object that put the popover up (the one that segued to it) dismisses it.
  How? Delegation. The view controller inside the popover sends a delegate method when it's done.
  So very often view controllers inside a popover have a settable delegate method in their API.

- Popover has delegate which gets notified on dismissal

  `- (void)popoverControllerDidDismissPopover:(UIPopoverController *)sender;`
  Only sent if the user dismisses the popover by clicking elsewhere (not programmatic dismissals).

Popover

- Size

  Three ways for a popover's size to be determined ...

- Setting the size in Xcode

  Inspect a view controller.

  Now whenever this view controller appears in a popover, it'll be that size.

- Set `contentSizeForViewInPopover` in the content VC controller

  ```
  @property (nonatomic) CGSize contentSizeForViewInPopover;
  ```

  Or you could override it to calculate the size on the fly.

- Set the size by sending a message to the popover controller

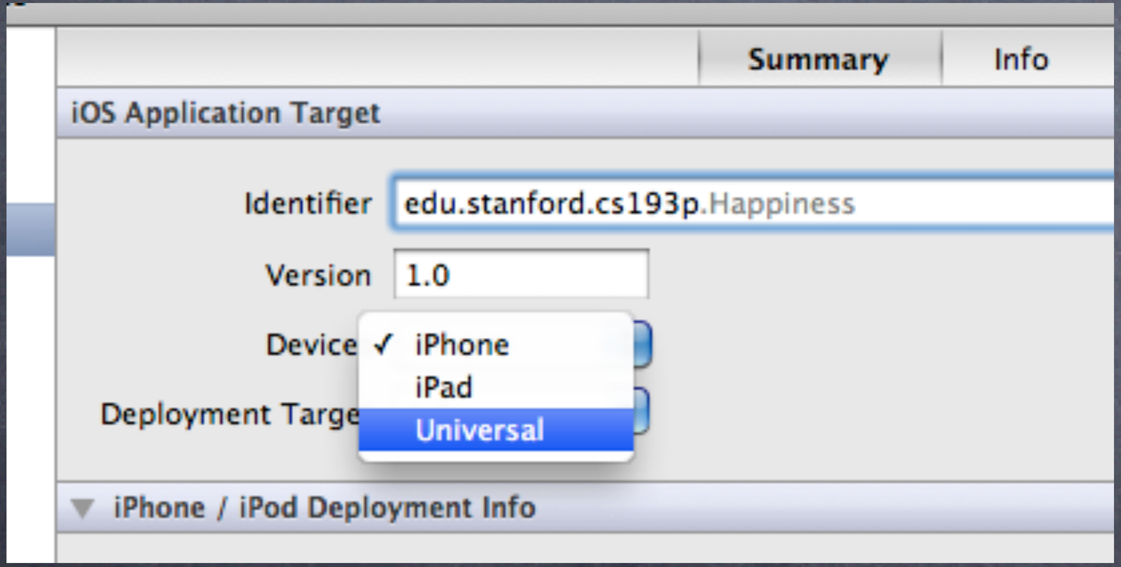
  ```
  - (void)setPopoverContentSize:(CGSize)size animated:(BOOL)animated;
  ```

  This last one is pretty non-object-oriented.

  Presumably a view controller is more likely to know it's "natural size" than some other object.

Universal Applications

- A "Universal" application that will run on iPhone or iPad
  Single binary image.
- How to create one
  In an existing iOS 5 iPhone project, select the project in the Navigator (on the left in Xcode)
  Change the Devices pull down under the Summary tab to be Universal.
  Then, lower down in that window, choose the storyboard for each platform.
  An iPad storyboard must be created as an iPad storyboard from the start (no conversion).


Summary | Info
iOS Application Target
Identifier: edu.stanford.cs193p.Happiness
Version: 1.0
Devices: ✓ iPhone / iPad / [✓ Universal]
Deployment Target
iPhone / iPod Deployment Info

Universal Applications

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

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

  This is not a first-resort. You might well want to use other conditionals before using this.
  E.g., checking `self.splitViewController` in your Controller to see if you're in a split view.
  We'll see some examples of "other things to check" on the next slide.

Universal Applications

- Code conditional on whether you're on screen

  Because 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.
  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?

- What is the resolution of the view I'm drawing in?

  Use UIView's `@property (CGFloat) contentScaleFactor`.

Demo

- iPad Psychologist
  Two storyboards: one for iPhone/iPod Touch, one for iPad.
  Share the MVC code (but different segues and layout in the two storyboards).
  `UISplitViewController`
- Watch for ...
  A little bit more examples of struts and springs at the start (before we do iPad work).
  How we create a new storyboard and update our project settings to use both storyboards.
  Two different ways to update the right-hand side of a split view (target/action and Replace segue).
  How to control when split view hides the left-hand side of a split view behind a bar button.

Coming Up

- Homework
  You know everything you need to know now to do assignment 3!
- Next Lecture
  View Controller Lifecycle
  Scroll View, Image View, Web View
  Perhaps Tab Bar Controller, etc.
- Friday
  AVFoundation framework - Capturing and manipulating images.