Stanford CS193p

Developing Applications for iOS
Winter 2015

Today

- Final Project Requirements
  - What you need to do and by when
- Unwind Segues
  - The one segue that does not create a new MVC
- Alerts and Action Sheets
  - Notifying the user of exceptional events
  - Or offering the user a branching decision
- Timer
  - Calling a method periodically
- Animation
  - Time permitting

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 significant feature which was NOT taught in lecture!
  - Also, 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.
- Project scope is the same as about three weeks of homework
  - It should be at or above the level of sophistication of the last couple of homework assignments.
  - But much larger in scope obviously (3 weeks versus 1 week worth of work).
  - All students (including P/NC) must pass both homework and final project segments separately.
- Do not just regurgitate the homework assignments
  - You've already proven to us you can do what the homework assignments ask
  - Now show us that you really understood that stuff and use iOS in different contexts
  - Start from scratch (don't start with one of your homework assignments).
  - Take concepts to the next level. Be creative.

Final Project

- Examples of bad projects
  - Smashtag 2 or "thing that looks a lot like Smashtag with the names changed"
  - Lots of MVCs with only buttons, labels and text fields and nothing else (i.e. little breadth)
  - All table views and little else (i.e. little breadth)
  - An animated game (no matter how cool) with no table views or buttons or labels or … (breadth)
  - A really cool mathematical problem solving app with little UI (this is an iOS class)
- 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.
- Must work on hardware!
  - If you do a live demo on hardware, obviously you will have satisfied this.
  - Otherwise you must bring your hardware to the final exam (or before) to demo for TA.
  - iPad or iPhone or iPod Touch okay.

Final Project

- 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.)
  - This is an iOS DEVELOPMENT course! Show us how well you can develop for this platform.
  - Don't waste your time writing server-side code.
  - It's okay to "simulate" a server-side interaction to make your code demonstrable.
  - Bottom line: Only your code that uses the iOS SDK will "count".

Final Project

- Presentation Quality Matters
  - 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 90 seconds (strictly enforced), so make 'em count.
- Live demo?
  - All iOS 8 devices (iPad2+, iPhone4S, iPhone5) can mirror their screen to the projector here.
  - Live demos are perilous, as you saw all quarter :), but can be very effective!
  - You must, at worst, show screen shots of your application.
  - If you don't want to go live ...
    - 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.

  ** "blocking" is where people stand on stage.

  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 (this is the NOT COVERED LECTURE feature)
    - Play entity
    - Scene entity
    - BlockingElement entity
    - LineOfDialog entity
  - Will support printing the blocking via AirPrint (a bonus 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(s).
  - Specifies platform (iPad-only sacrifices breadth, but makes sense for this project).
  - It's creative (it's not just Calculator or Smashtag recycled).

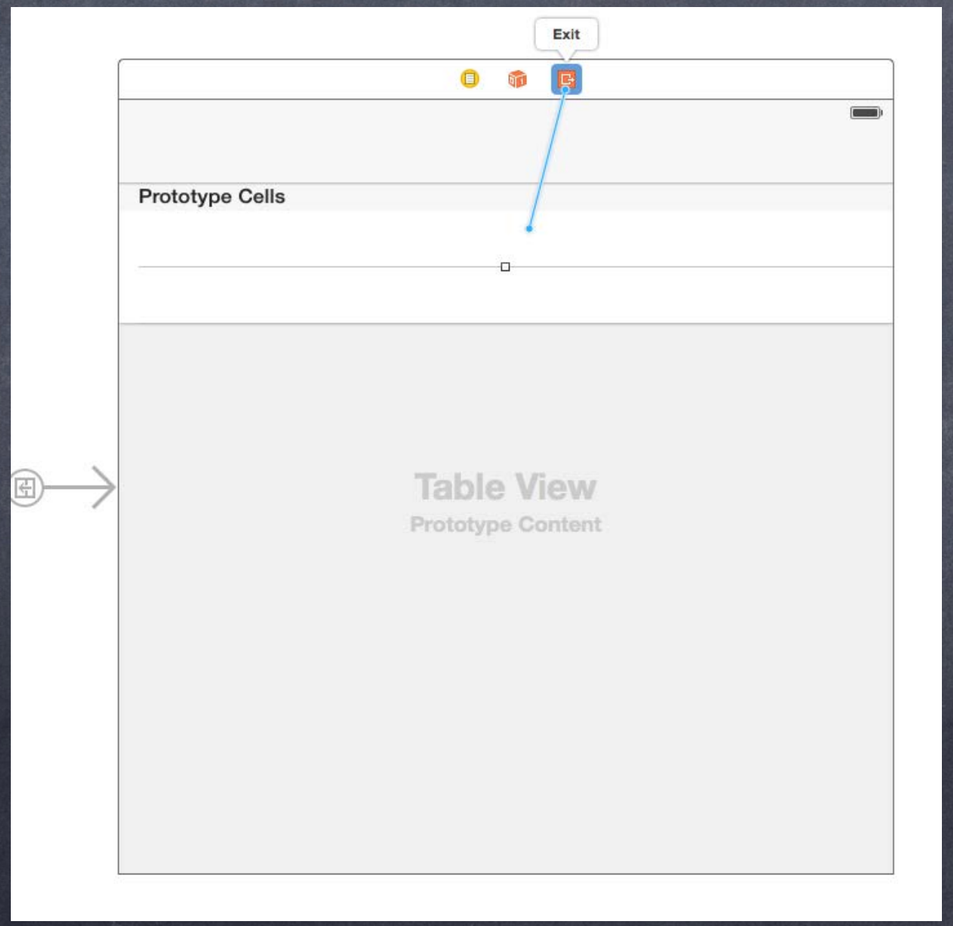
Unwind Segue

- The only segue that does NOT create a new MVC
  - It can only segue to other MVCs that (directly or indirectly) presented the current MVC
- What's it good for?
  - Jumping up the stack of cards in a navigation controller (other cards are considered presenters)
  - Dismissing a Modally segued-to MVC while reporting information back to the presenter

Unwind Segue

- How does it work?

  Instead of ctrl-dragging to another MVC, you ctrl-drag to the "Exit" button in the same MVC

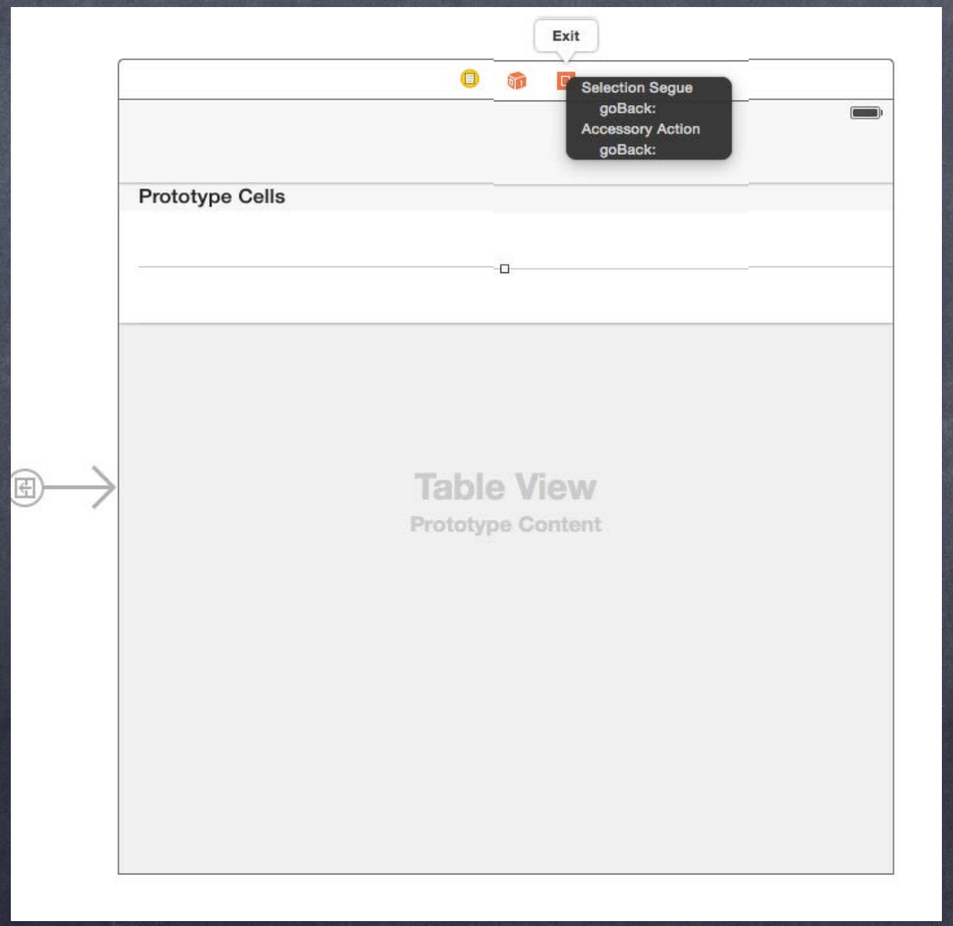


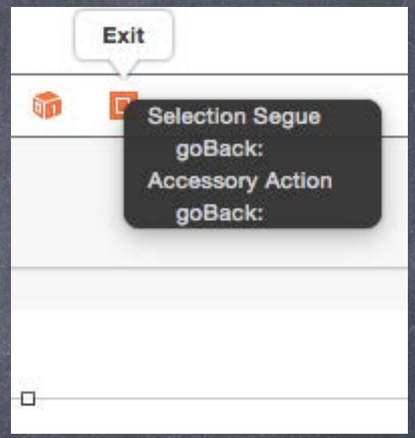
Unwind Segue

- How does it work?

  Instead of ctrl-dragging to another MVC, you ctrl-drag to the "Exit" button in the same MVC

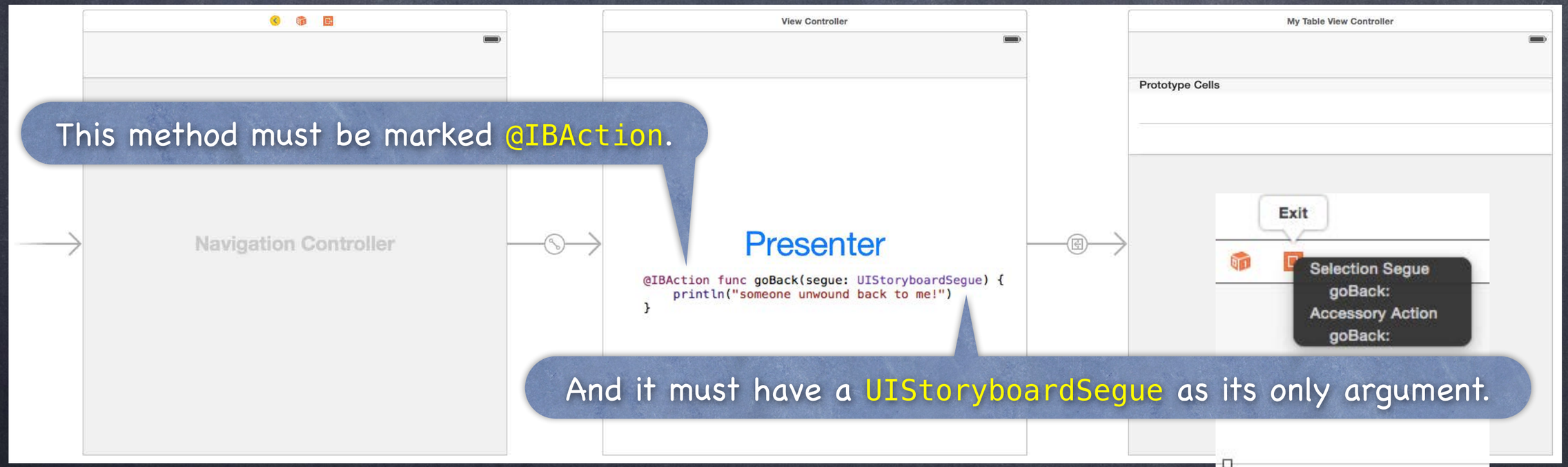
  Then you can choose a special `@IBAction` method you've created in another MVC





Unwind Segue

- How does it work?
  - Instead of ctrl-dragging to another MVC, you ctrl-drag to the "Exit" button in the same MVC
  - Then you can choose a special `@IBAction` method you've created in another MVC
  - This means "segue by exiting me and finding a presenter who implements that method"
  - If no presenter (directly or indirectly) implements that method, the segue will not happen

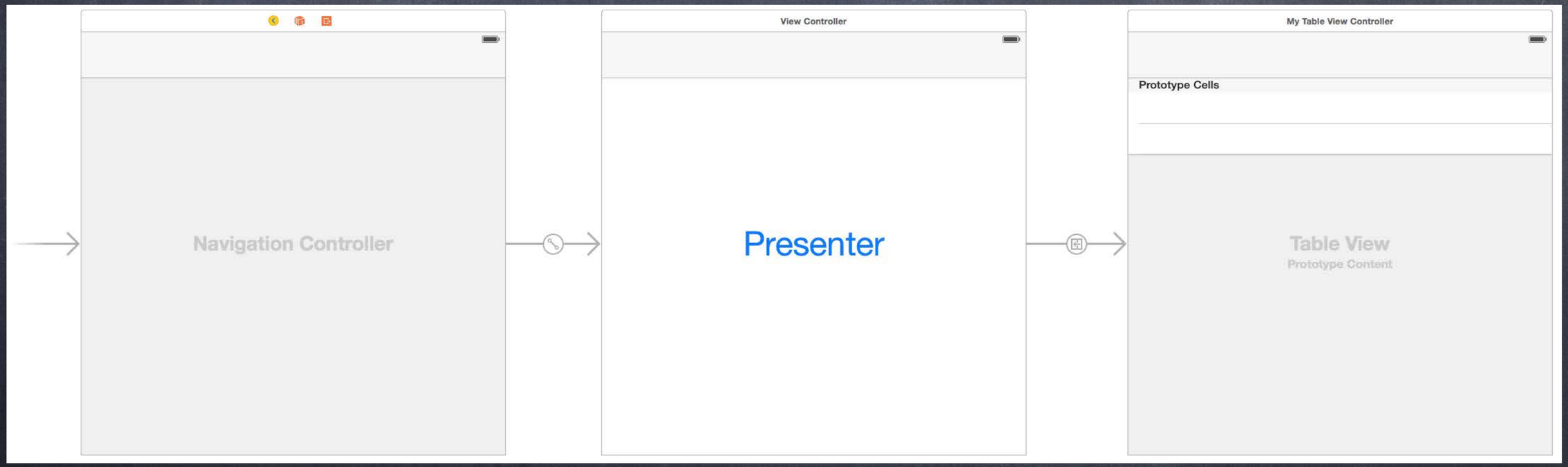


Unwind Segue

- How does it work?

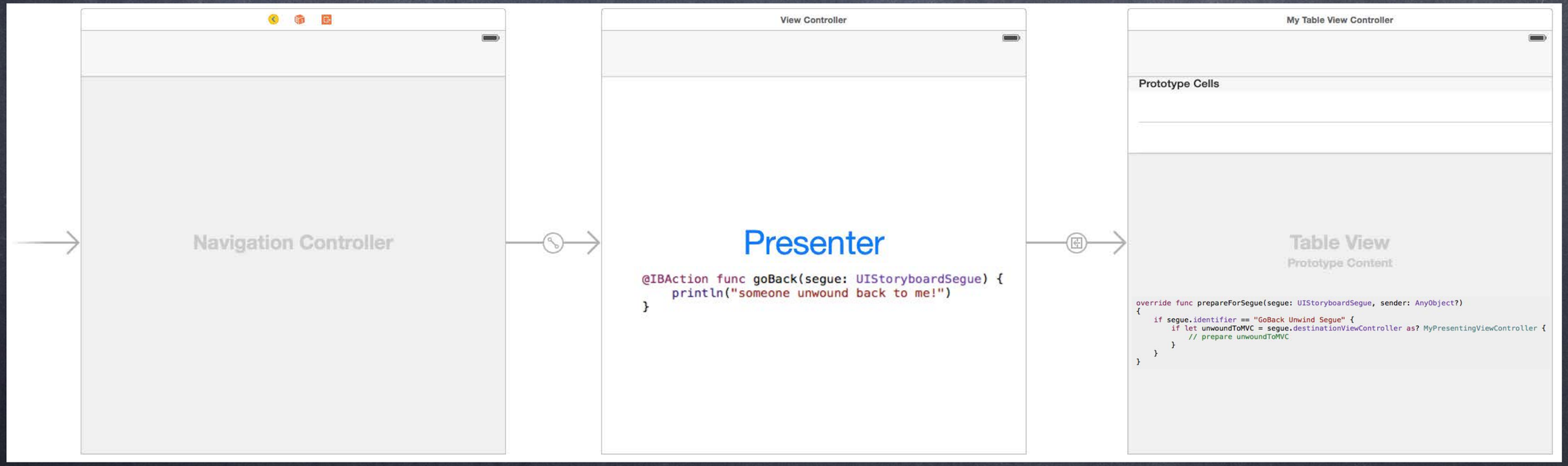
  If the `@IBAction` can be found, you (i.e. the presented MVC) will get to `prepareForSegue` as normal

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?)
{
    if segue.identifier == "GoBack Unwind Segue" {
        if let unwoundToMVC = segue.destinationViewController as? MyPresentingViewController {
            // prepare unwoundToMVC
        }
    }
}
```



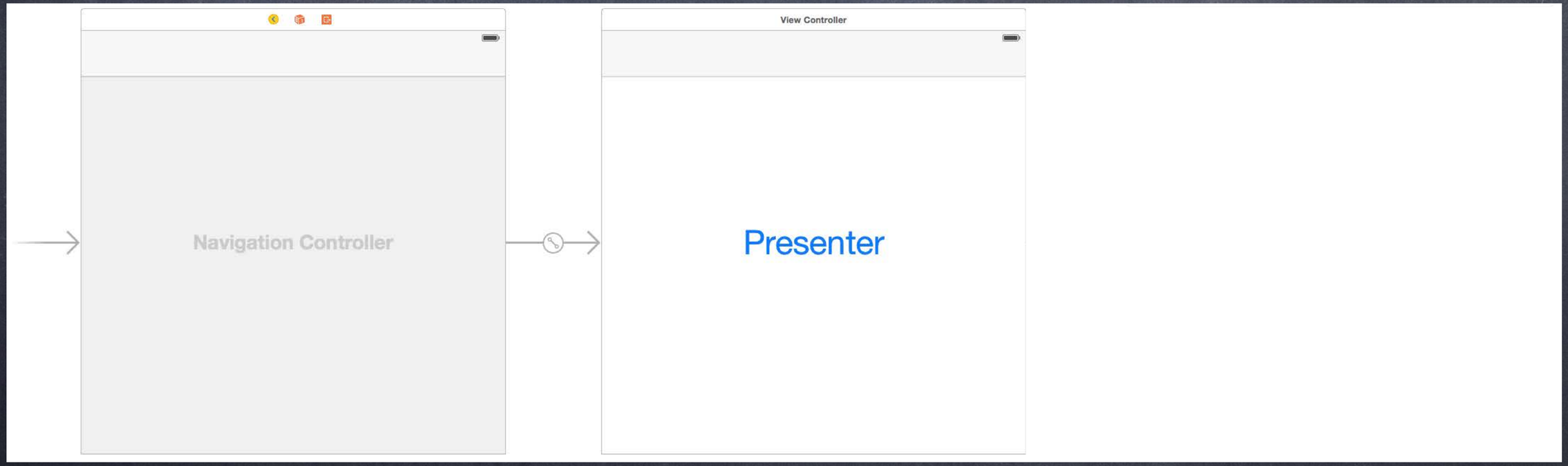
Unwind Segue

- How does it work?
  - If the `@IBAction` can be found, you (i.e. the presented MVC) will get to `prepareForSegue` as normal
  - Then the special `@IBAction` will be called in the other MVC and that MVC will be shown on screen



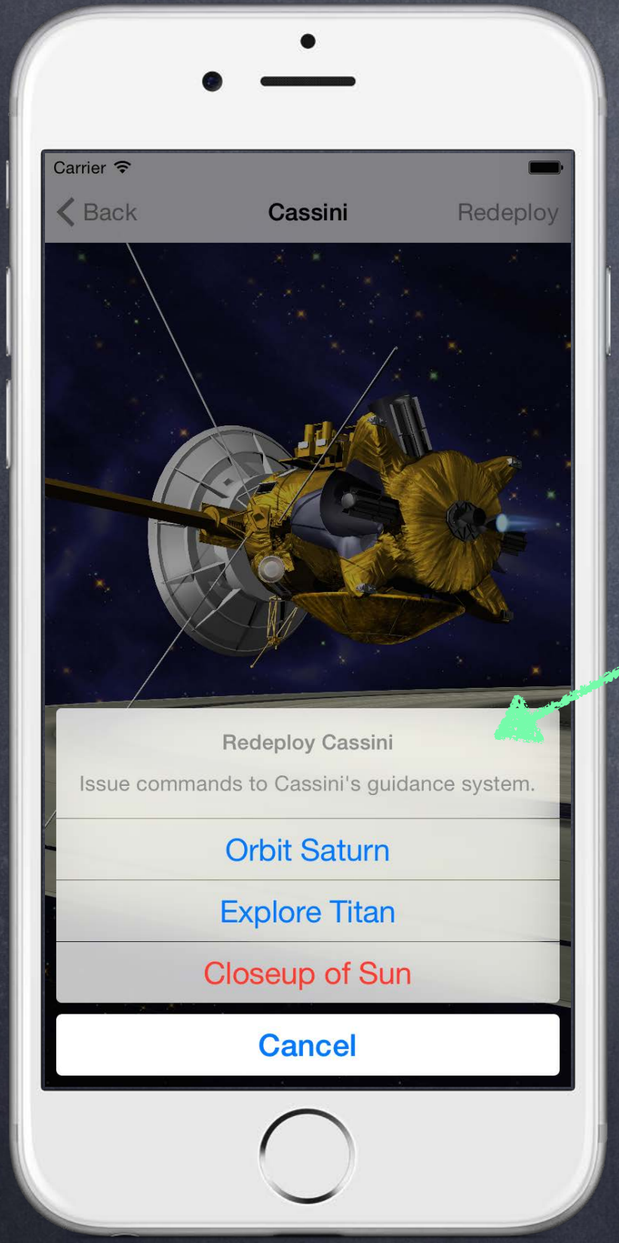
Unwind Segue

- How does it work?
  - If the `@IBAction` can be found, you (i.e. the presented MVC) will get to `prepareForSegue` as normal
  - Then the special `@IBAction` will be called in the other MVC and that MVC will be shown on screen
  - You will be dismissed in the process (i.e. you'll be "unpresented" and thrown away)
  - We haven't talked about Modal segues yet, but unwind works exactly the same for those



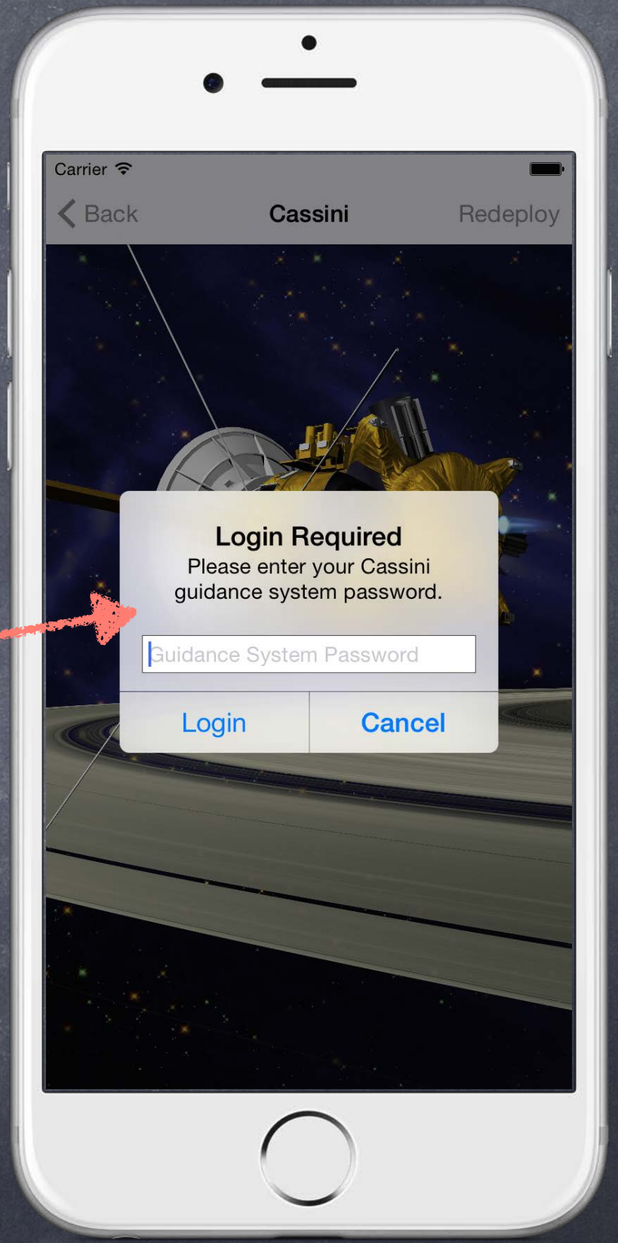
Alerts and Action Sheets

- 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").
  - Can have a text field to get a quick answer (e.g. password)
- 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 bar button item or from any rectangular area in a view.
  - Generally asks questions that have more than two answers.
  - Think of action sheets as presenting "branching decisions" to the user (i.e. what next?).

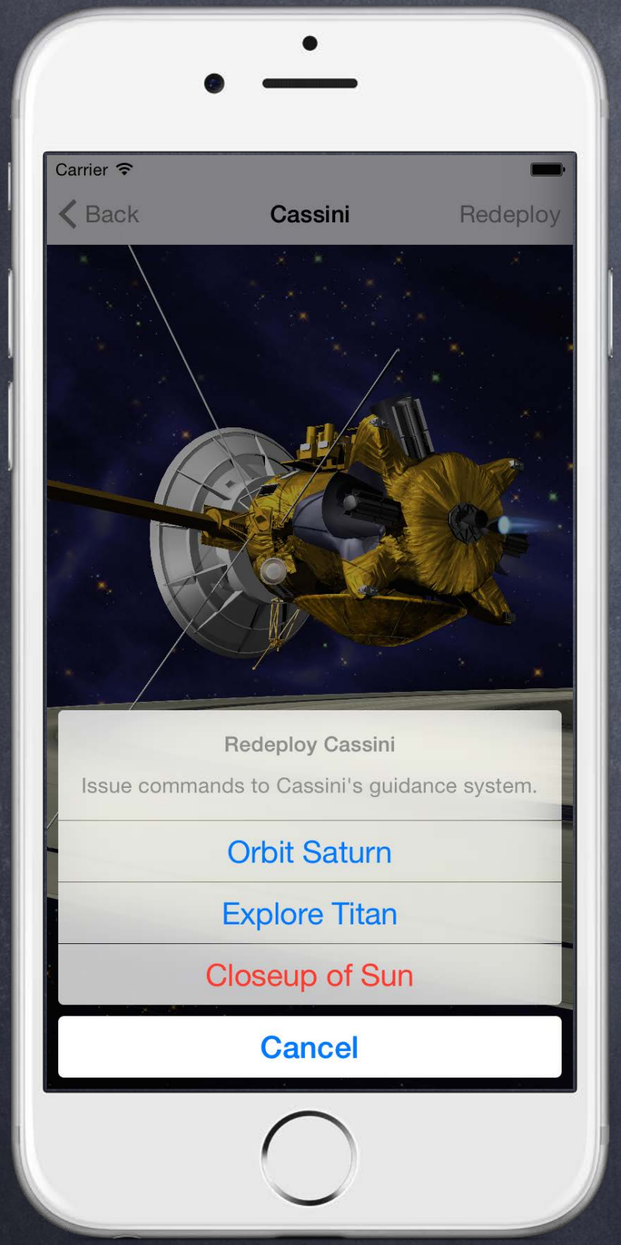


Action Sheet

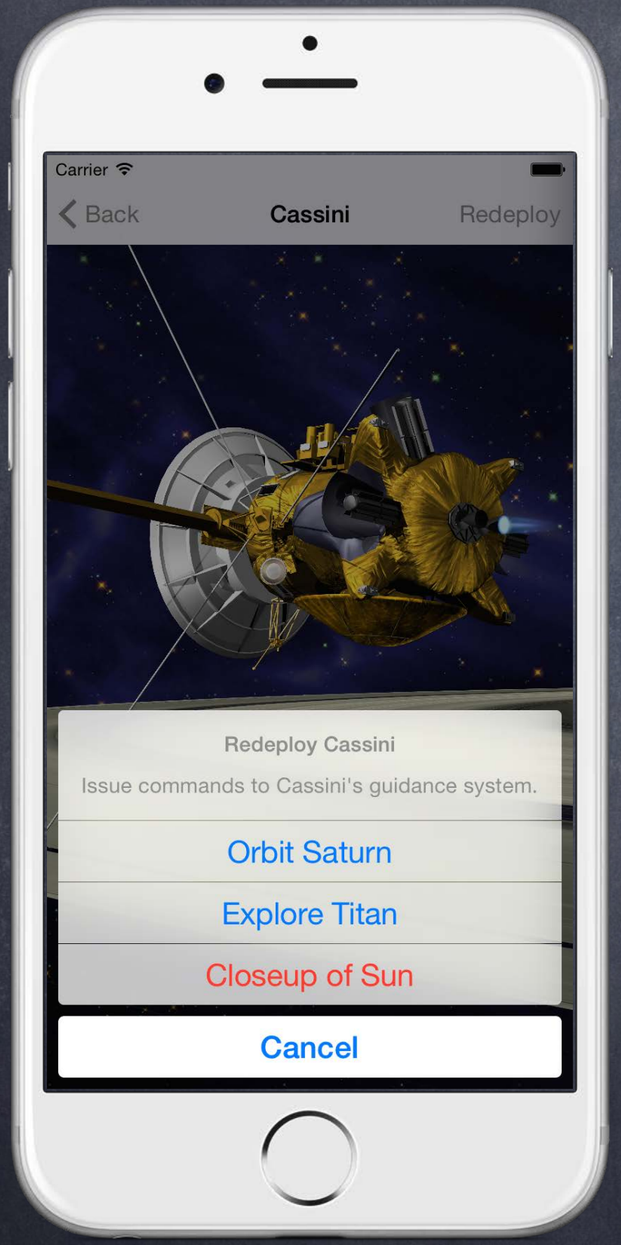
Alert



CS193p
Winter 2015

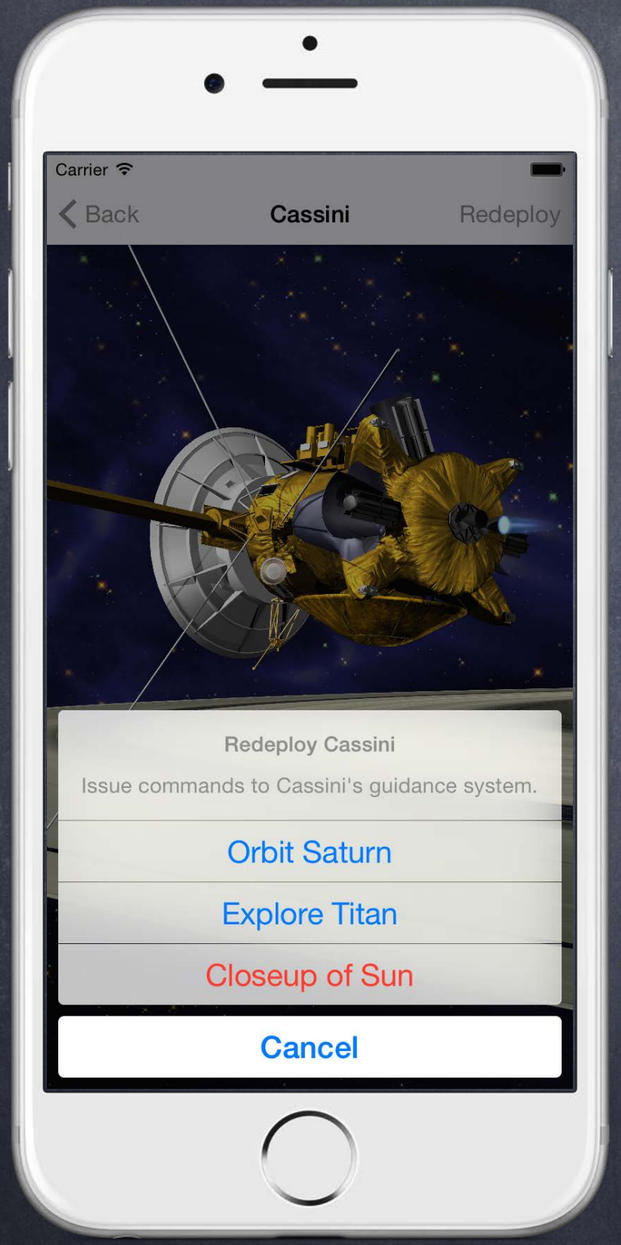



```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)
```

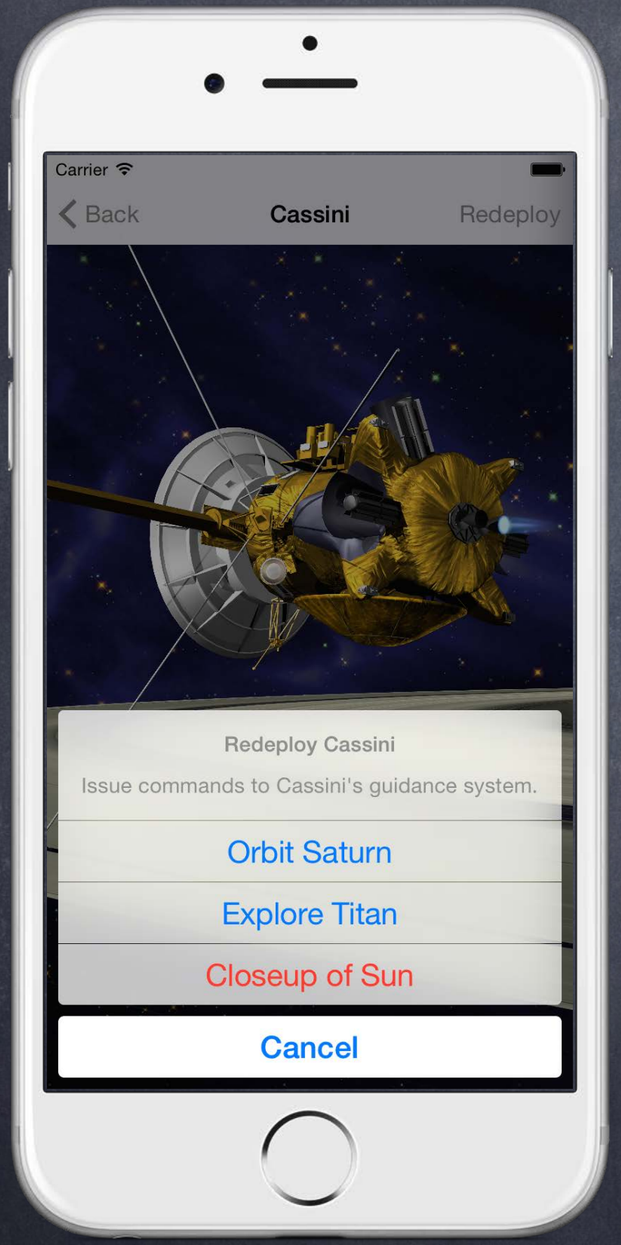



```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)
```

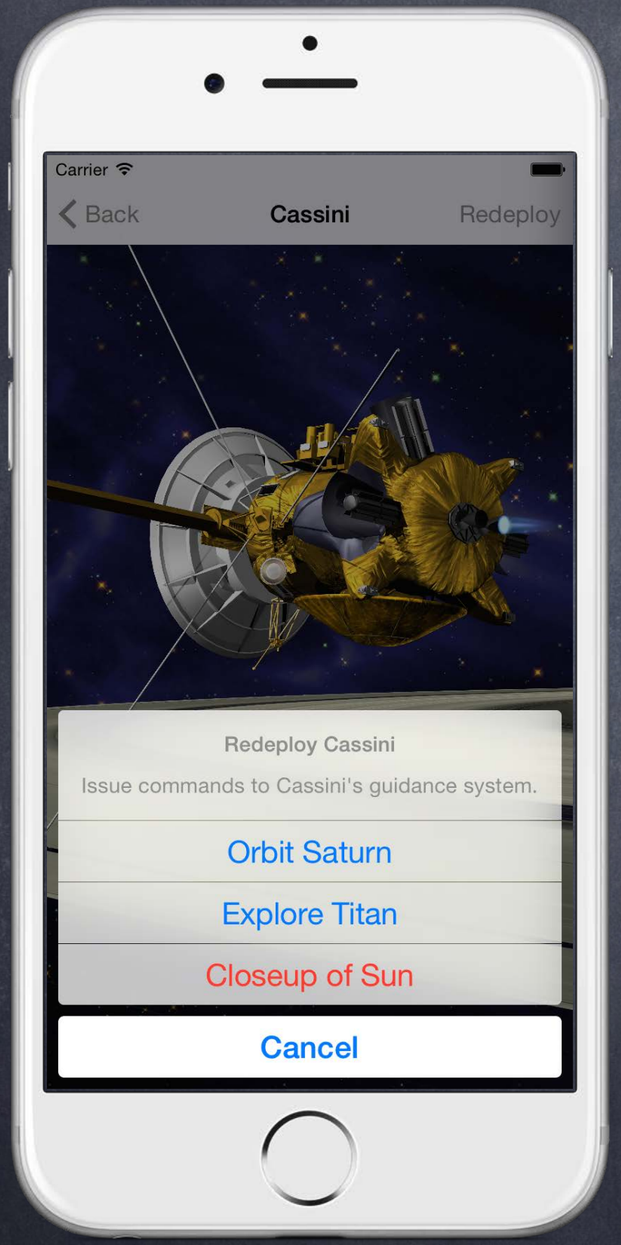
CS193p
Winter 2015



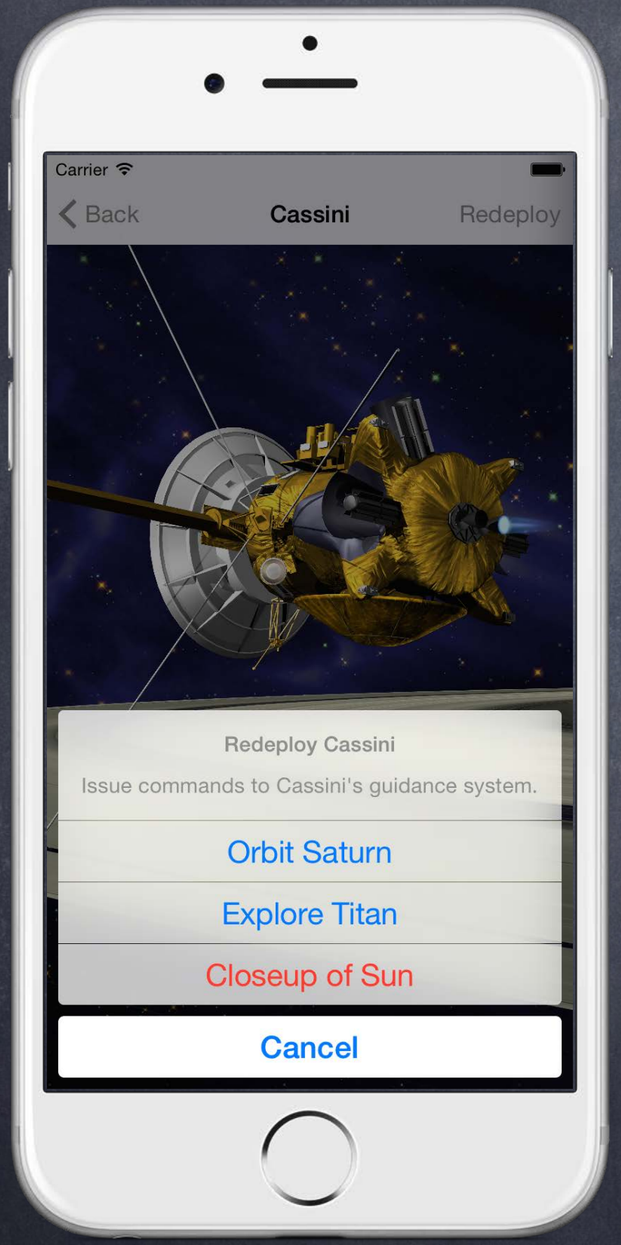

```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(...)
```



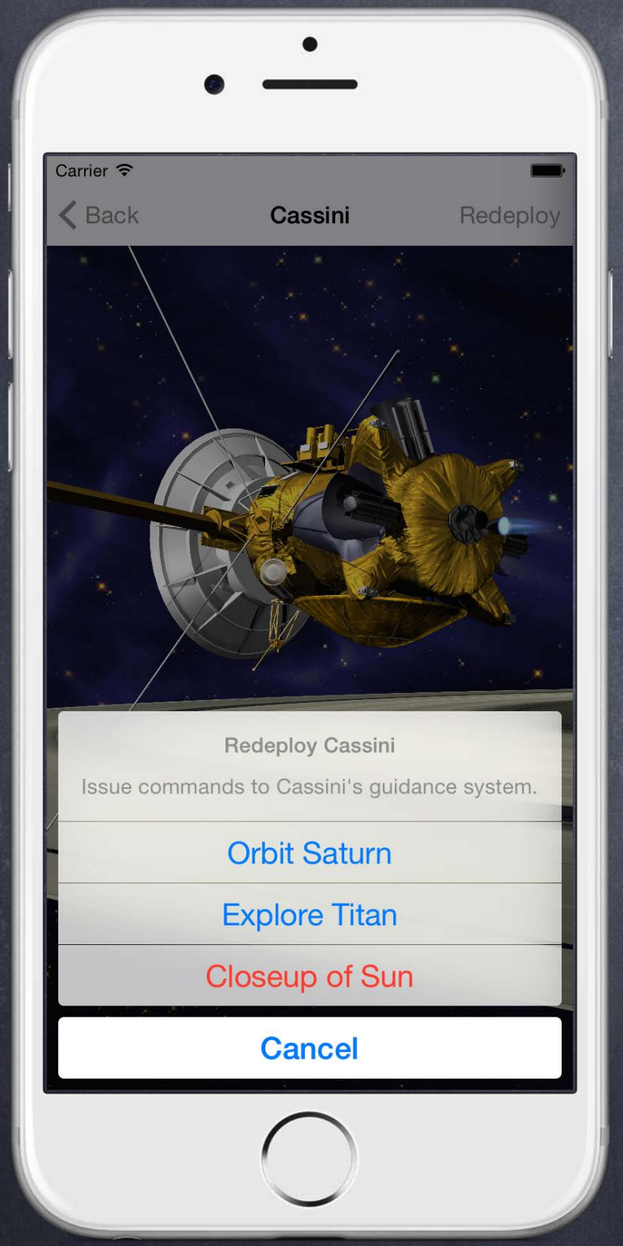

```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(UIAlertAction(...))
```



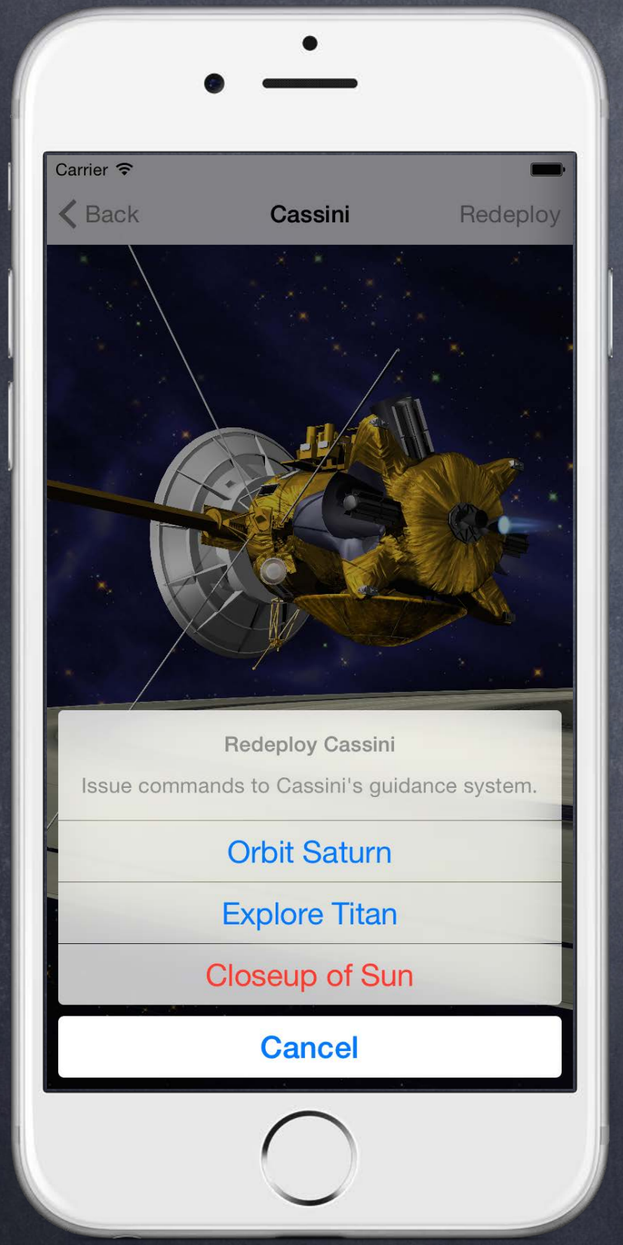

```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(UIAlertAction(
    title: String,
    style: UIAlertActionStyle,
    handler: (action: UIAlertAction) -> Void
))
```



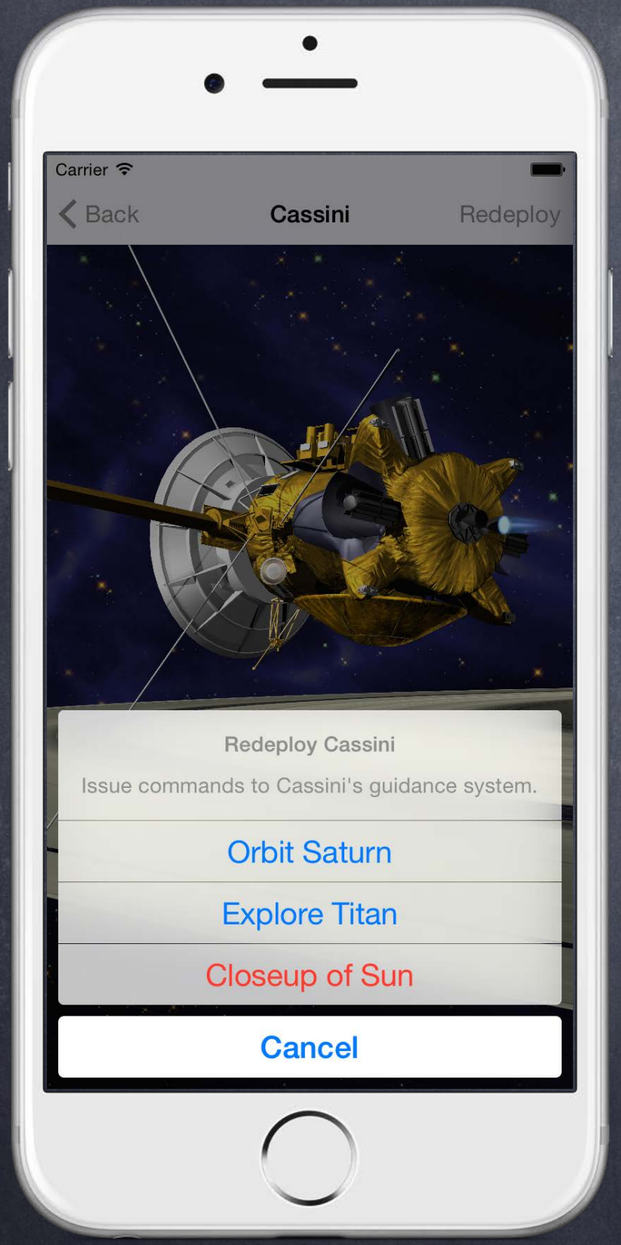

```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(UIAlertAction(
    title: “Orbit Saturn”,
    style: UIAlertActionStyle.Default)
    { (action: UIAlertAction) -> Void in
        // go into orbit around saturn
    }
)
```



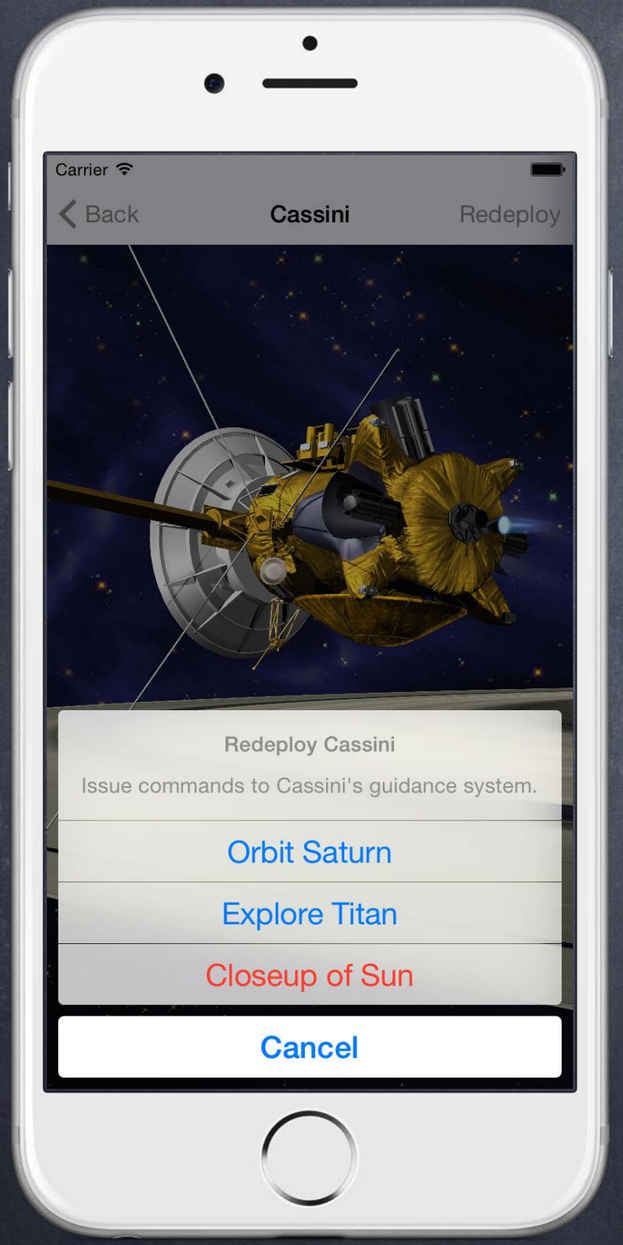

```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(UIAlertAction(
    title: “Orbit Saturn”,
    style: UIAlertActionStyle.Default)
    { (action: UIAlertAction) -> Void in
        // go into orbit around saturn
    }
)

alert.addAction(UIAlertAction(
    title: “Explore Titan”,
    style: .Default)
    { (action: UIAlertAction) -> Void in
        if !self.loggedIn { self.login() }
        // if loggedIn go to titan
    }
)
```




```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(/* orbit saturn action */)
alert.addAction(/* explore titan action */)
```

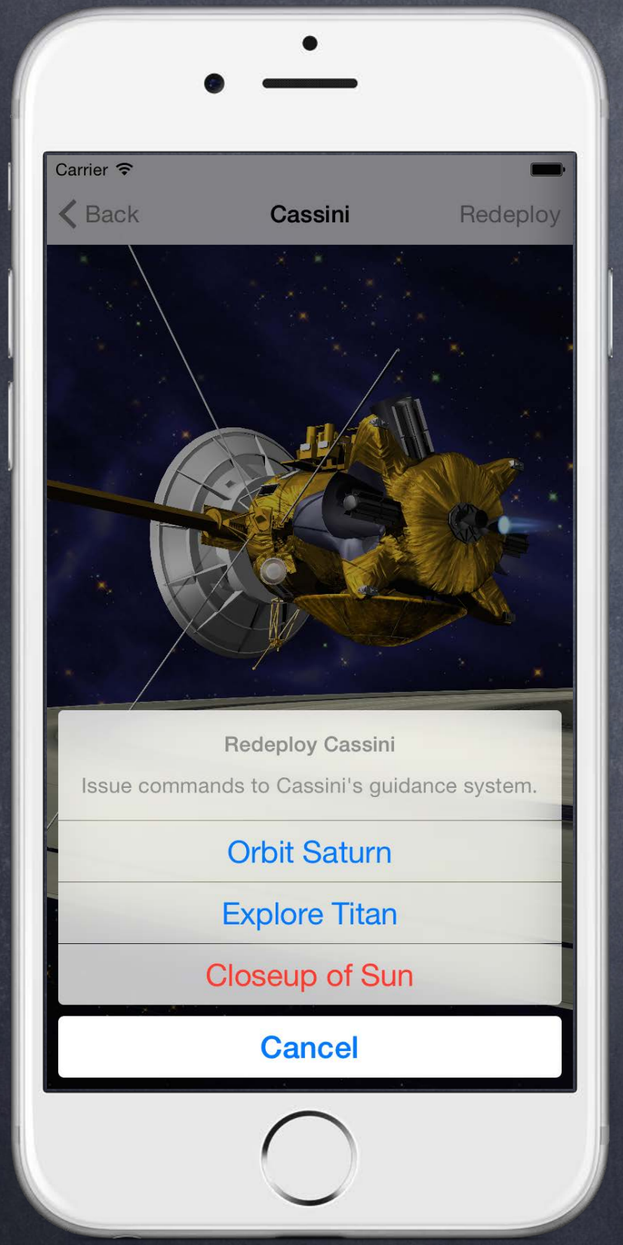



```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(/* orbit saturn action */)
alert.addAction(/* explore titan action */)

alert.addAction(UIAlertAction(
    title: “Closeup of Sun”,
    style: .Destructive)
    { (action: UIAlertAction) -> Void in
        if !loggedIn { self.login() }
        // if loggedIn destroy Cassini by going to Sun
    }
)
```

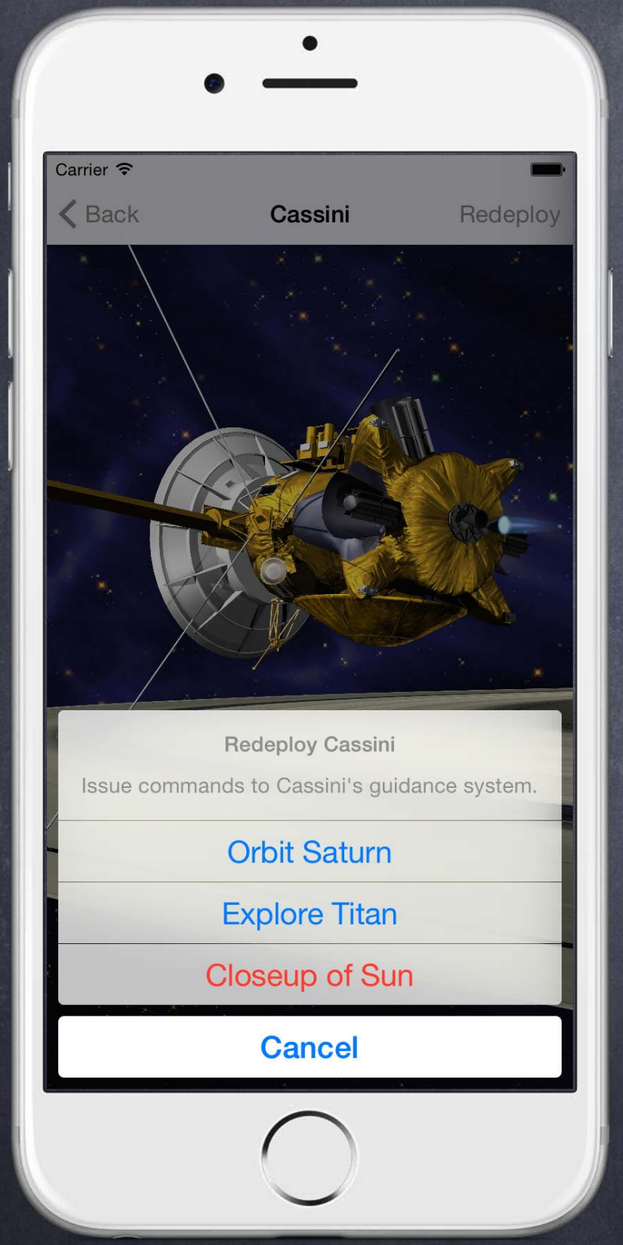



```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(/* orbit saturn action */)
alert.addAction(/* explore titan action */)

alert.addAction(UIAlertAction(
    title: “Closeup of Sun”,
    style: .Destructive)
    { (action: UIAlertAction) -> Void in
        if !loggedIn { self.login() }
        // if loggedIn destroy Cassini by going to Sun
    }
)

alert.addAction(UIAlertAction(
    title: “Cancel”,
    style: .Cancel)
    { (action: UIAlertAction) -> Void in
       // do nothing
    }
)
```

CS193p
Winter 2015

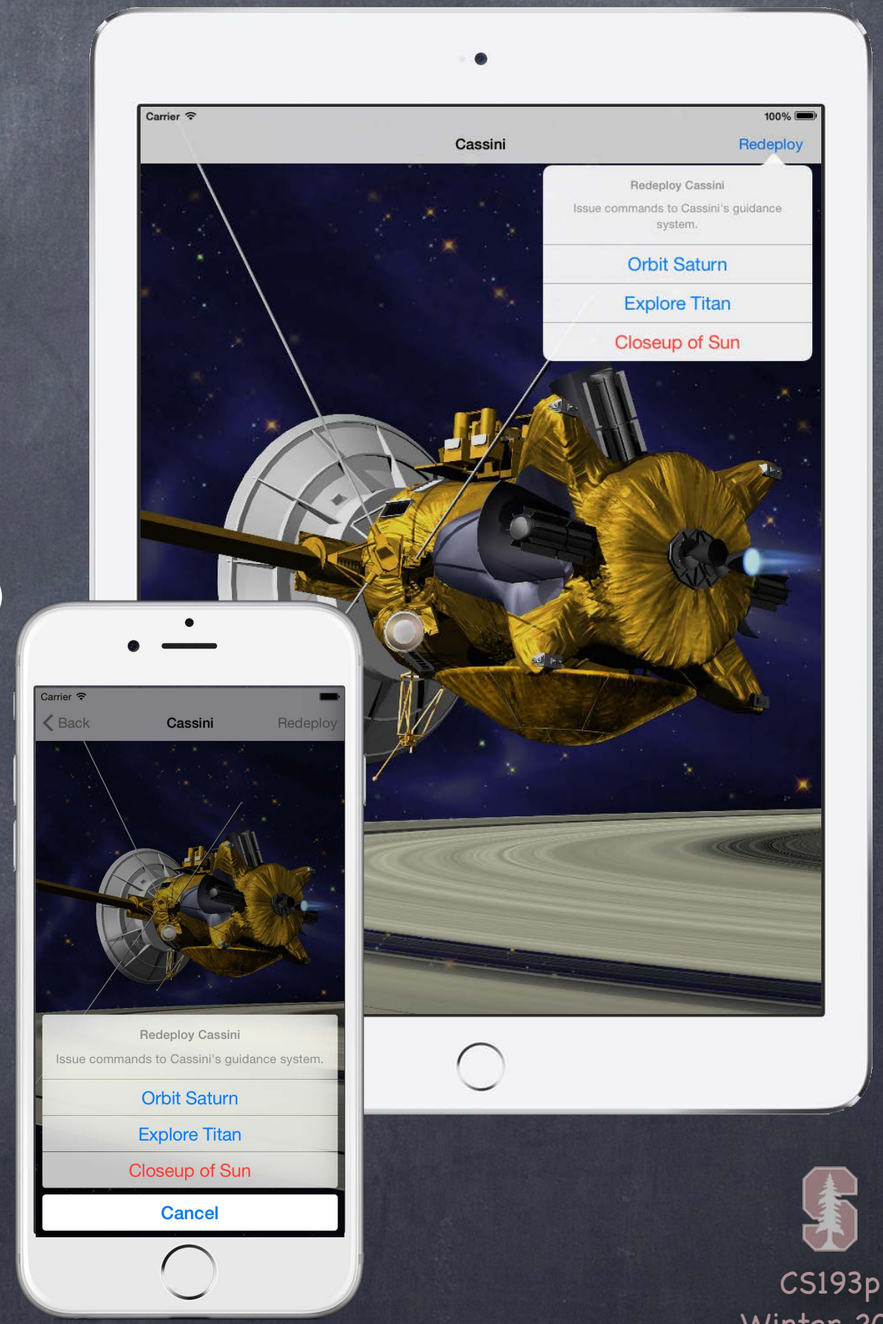



```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(/* orbit saturn action */)
alert.addAction(/* explore titan action */)
alert.addAction(/* destroy with closeup of sun action */
alert.addAction(/* do nothing cancel action */
```



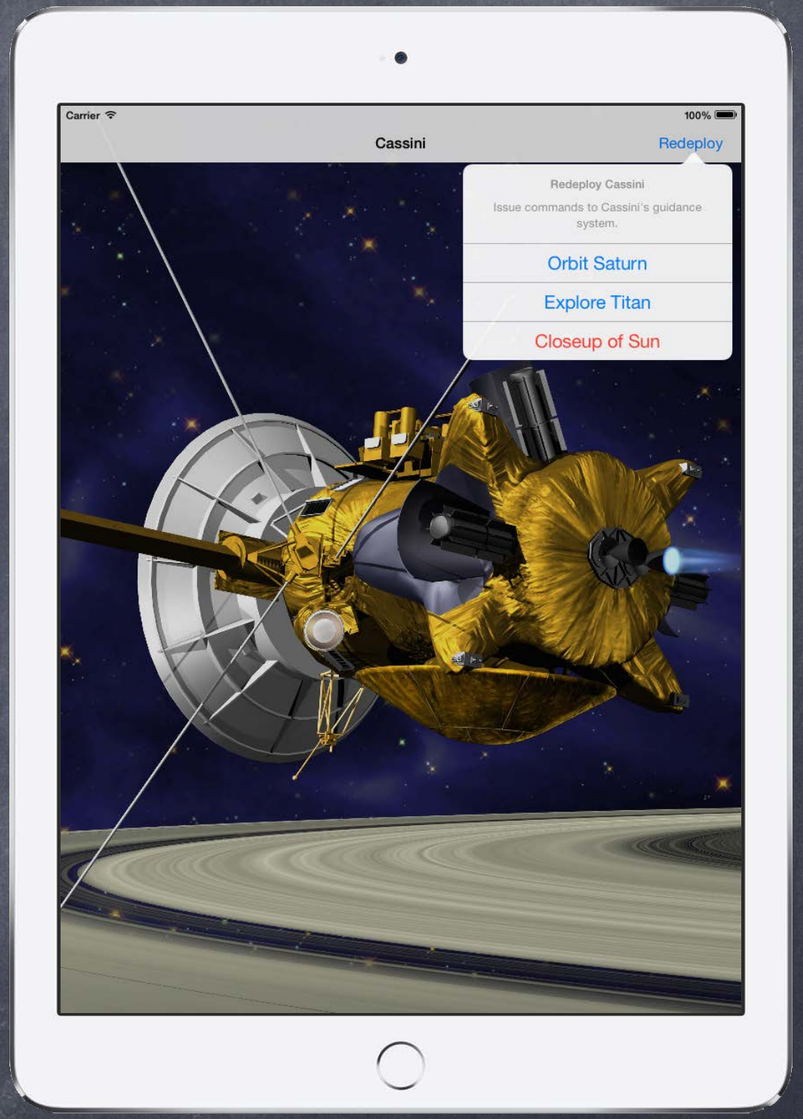

```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(/* orbit saturn action */)
alert.addAction(/* explore titan action */)
alert.addAction(/* destroy with closeup of sun action */
alert.addAction(/* do nothing cancel action */

presentViewController(alert, animated: true, completion: nil)
```

```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(/* orbit saturn action */)
alert.addAction(/* explore titan action */)
alert.addAction(/* destroy with closeup of sun action */
alert.addAction(/* do nothing cancel action */

presentViewController(alert, animated: true, completion: nil)
```

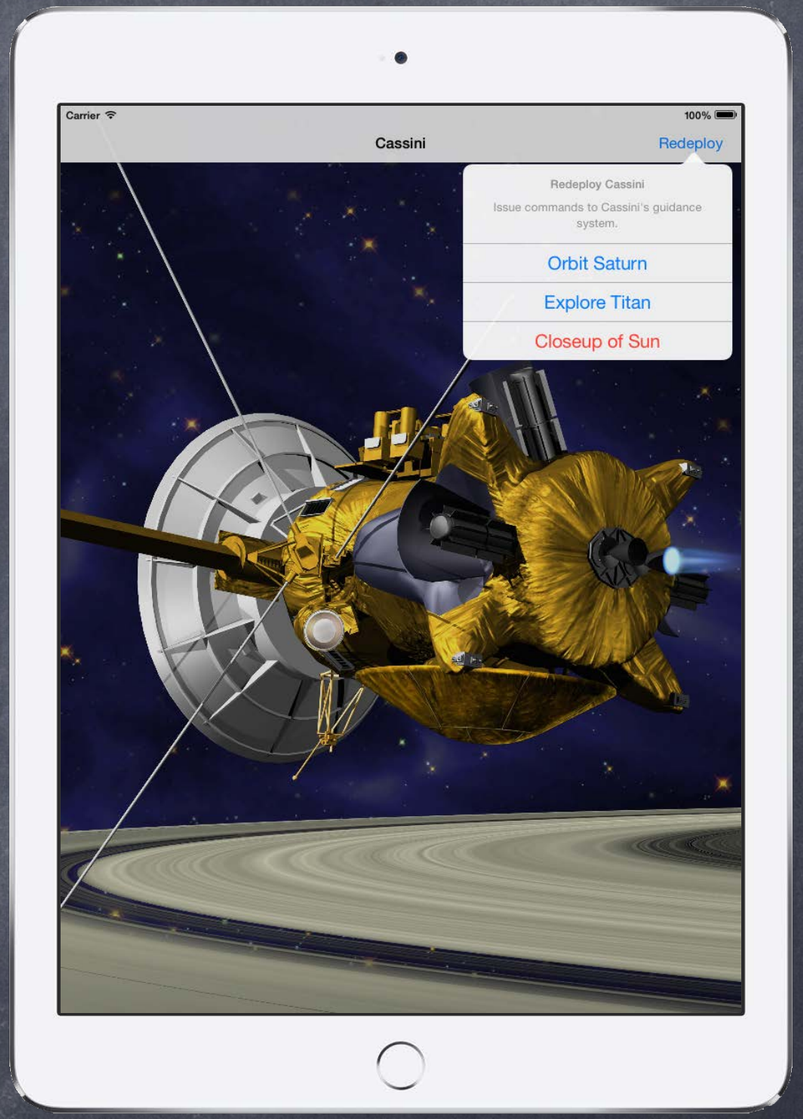


CS193p
Winter 2015

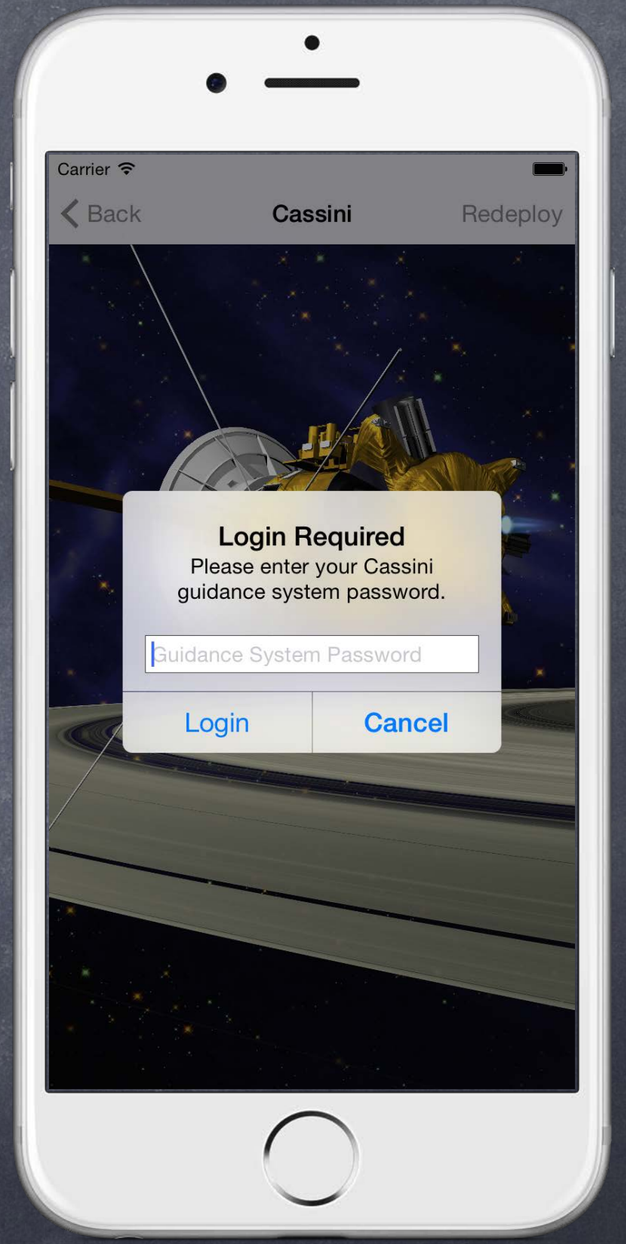
```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(/* orbit saturn action */)
alert.addAction(/* explore titan action */)
alert.addAction(/* destroy with closeup of sun action */
alert.addAction(/* do nothing cancel action */

alert.modalPresentationStyle = .Popover


presentViewController(alert, animated: true, completion: nil)
```



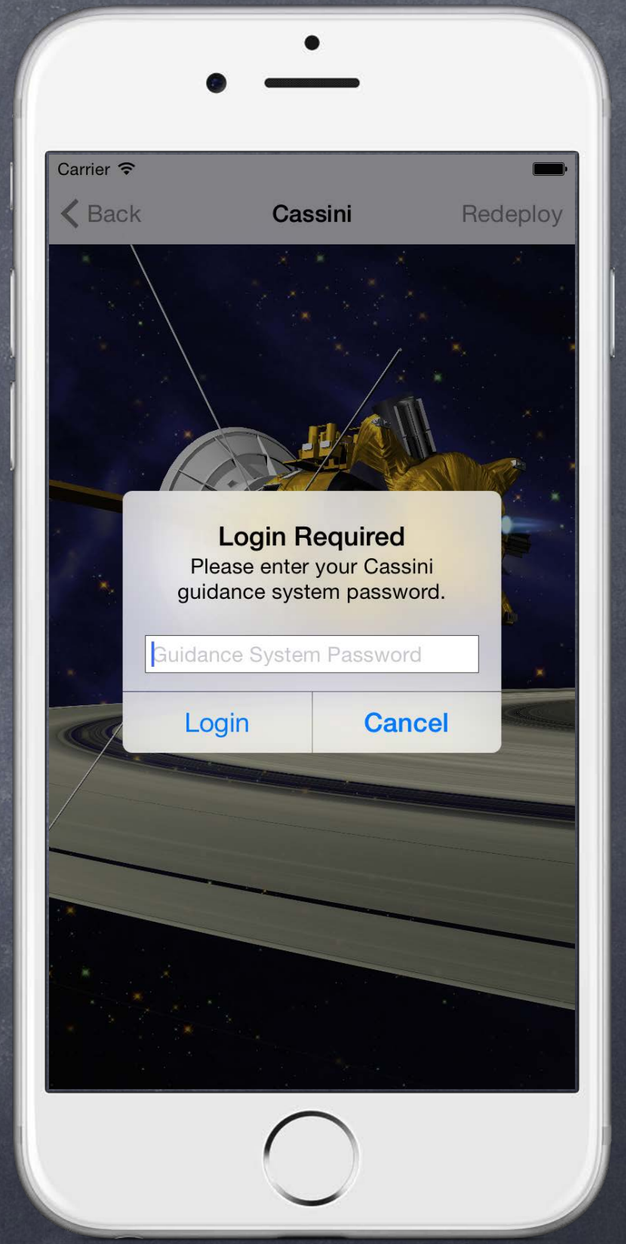
```
var alert = UIAlertController(
    title: “Redeploy Cassini”,
    message: “Issue commands to Cassini’s guidance system.”,
    preferredStyle: UIAlertControllerStyle.ActionSheet
)

alert.addAction(/* orbit saturn action */)
alert.addAction(/* explore titan action */)
alert.addAction(/* destroy with closeup of sun action */
alert.addAction(/* do nothing cancel action */

alert.modalPresentationStyle = .Popover
let ppc = alert.popoverPresentationController
ppc?.barButtonItem = redeployBarButtonItem

presentViewController(alert, animated: true, completion: nil)
```



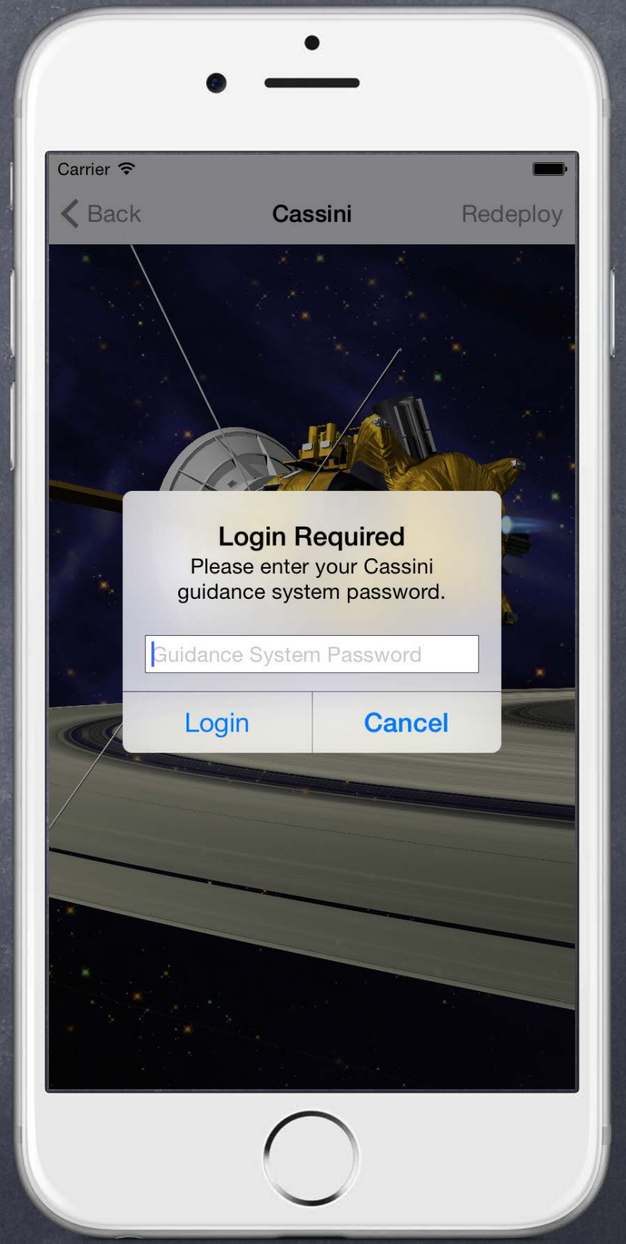
```
var alert = UIAlertController(
    title: “Login Required”,
    message: “Please enter your Cassini guidance system...”,
    preferredStyle: UIAlertControllerStyle.Alert
)
```



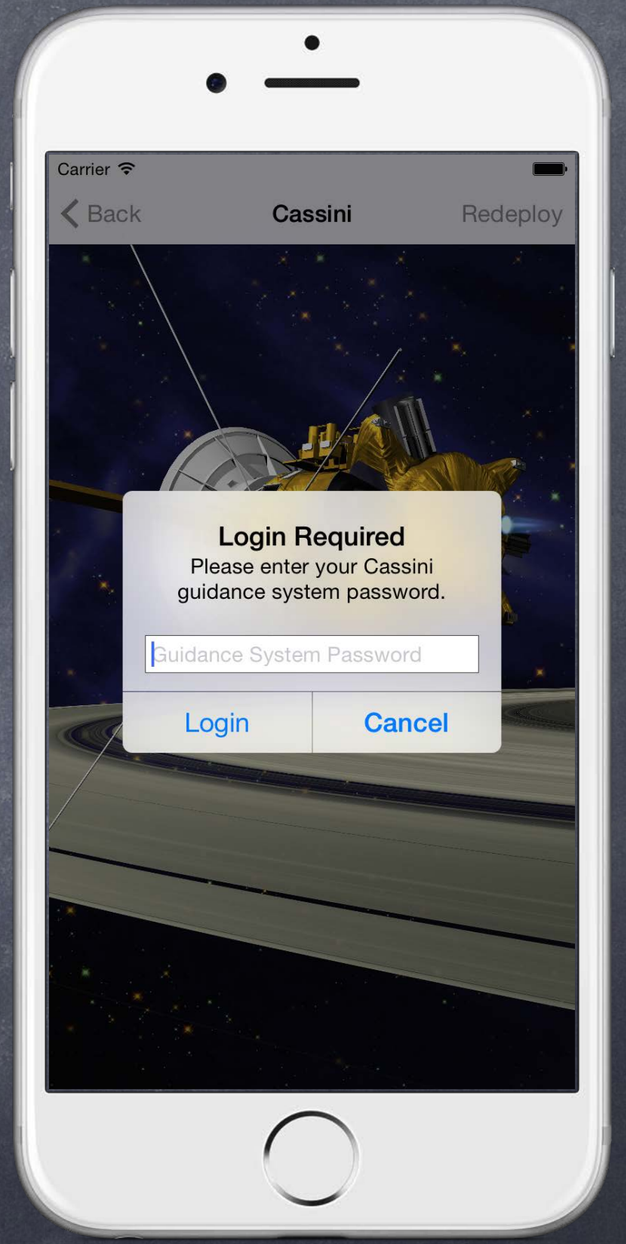
```
var alert = UIAlertController(
    title: “Login Required”,
    message: “Please enter your Cassini guidance system...”,
    preferredStyle: UIAlertControllerStyle.Alert
)

alert.addAction(UIAlertAction(
    title: “Cancel”,
    style: .Cancel)
    { (action: UIAlertAction) -> Void in
        // do nothing
    }
)
```



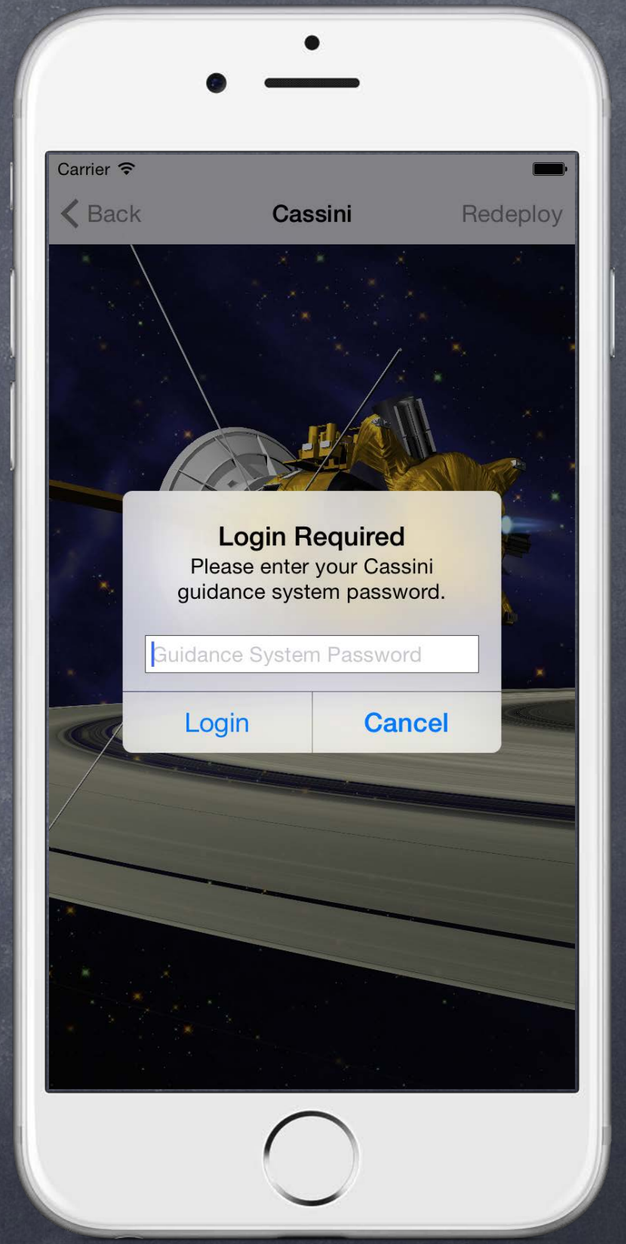
```
var alert = UIAlertController(
    title: “Login Required”,
    message: “Please enter your Cassini guidance system...”,
    preferredStyle: UIAlertControllerStyle.Alert
)

alert.addAction(/* cancel button action */)
```



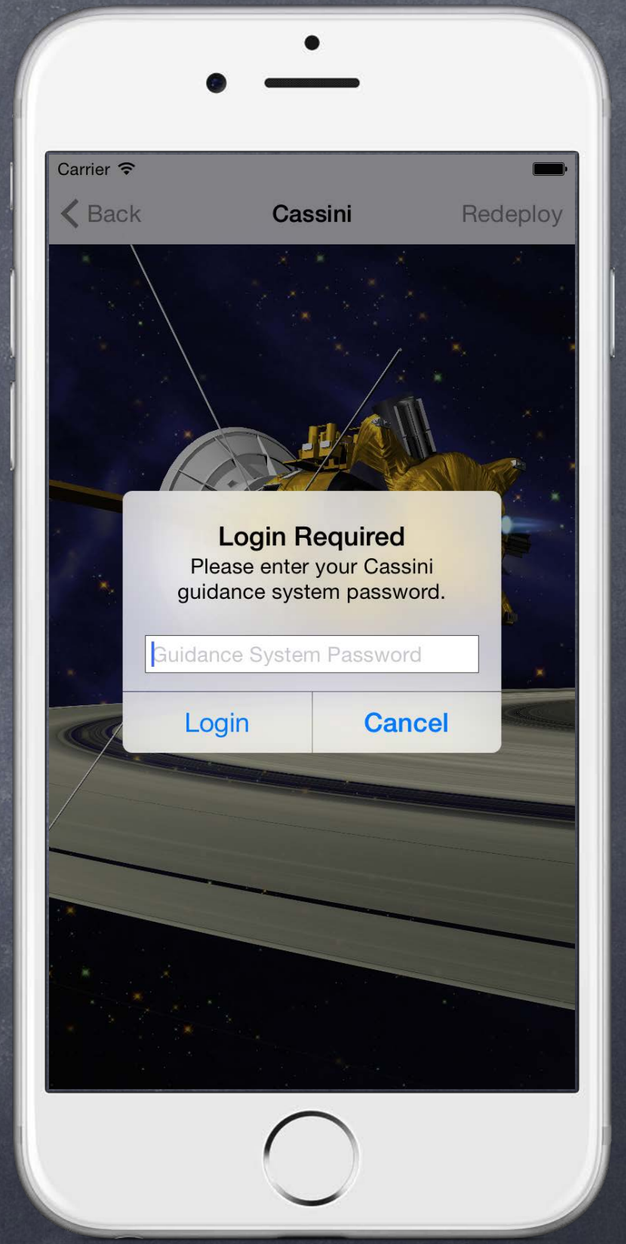
```
var alert = UIAlertController(
    title: “Login Required”,
    message: “Please enter your Cassini guidance system...”,
    preferredStyle: UIAlertControllerStyle.Alert
)

alert.addAction(/* cancel button action */)

alert.addAction(UIAlertAction(
    title: “Login”,
    style: .Default)
    { (action: UIAlertAction) -> Void in
        // get password and log in


    }
)
```



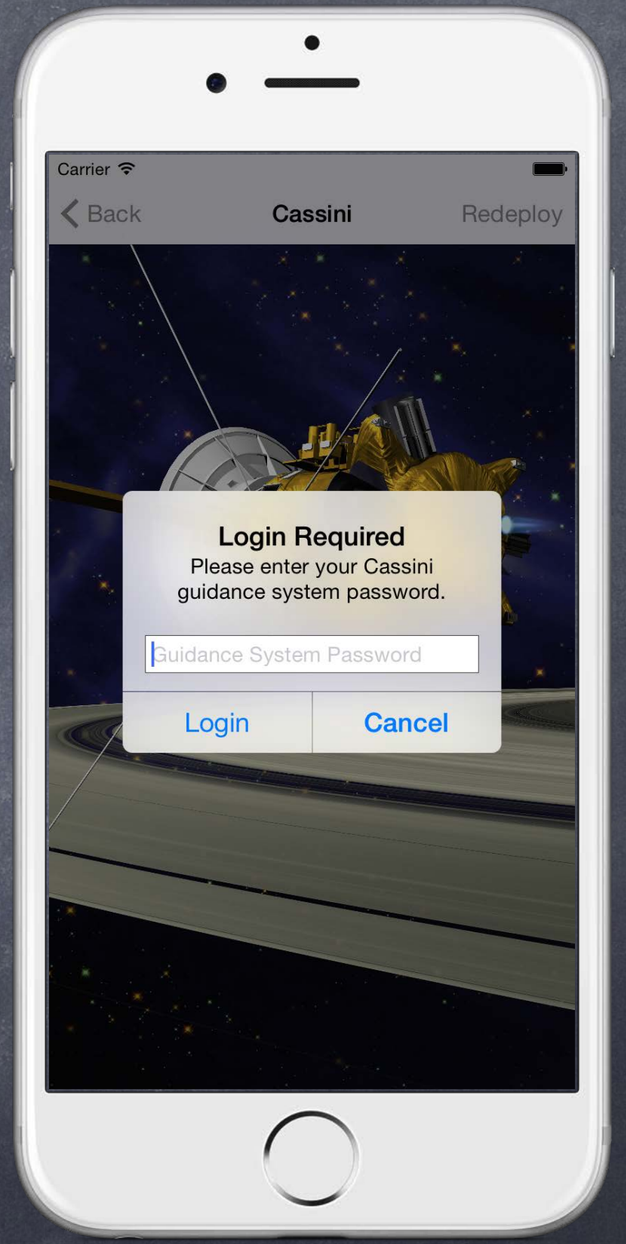
```
var alert = UIAlertController(
    title: “Login Required”,
    message: “Please enter your Cassini guidance system...”,
    preferredStyle: UIAlertControllerStyle.Alert
)

alert.addAction(/* cancel button action */)

alert.addAction(UIAlertAction(
    title: “Login”,
    style: .Default)
    { (action: UIAlertAction) -> Void in
        // get password and log in


    }
)

alert.addTextFieldWithConfigurationHandler { (textField) in
    textField.placeholder = “Guidance System Password”
}
```

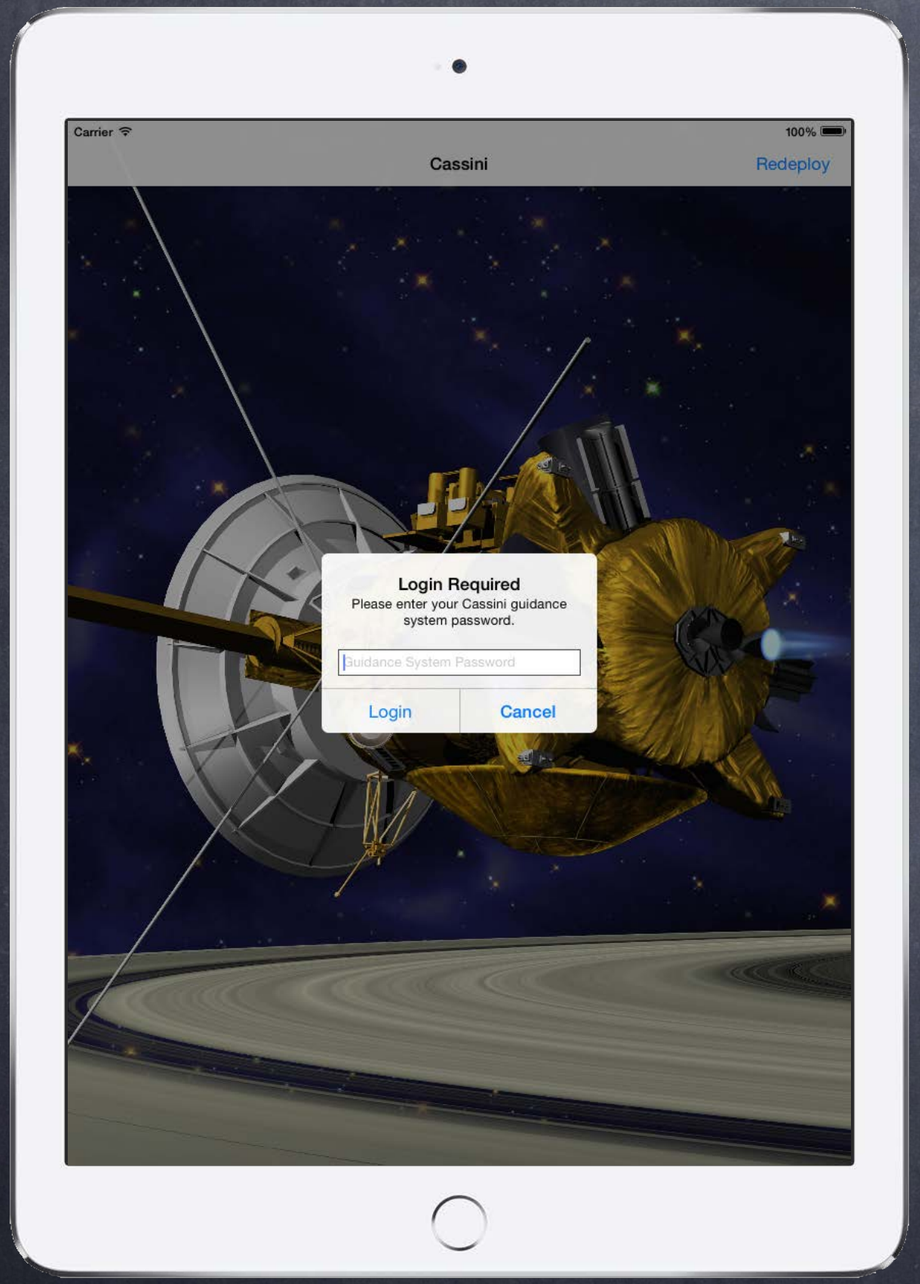


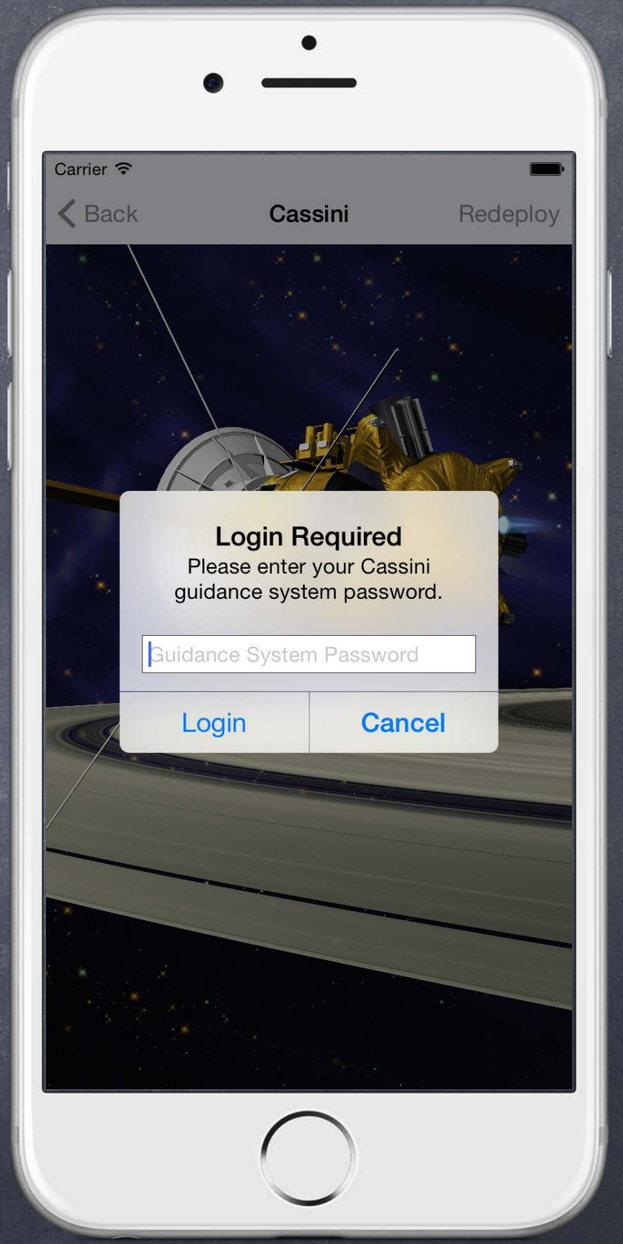
```
var alert = UIAlertController(
    title: “Login Required”,
    message: “Please enter your Cassini guidance system...”,
    preferredStyle: UIAlertControllerStyle.Alert
)

alert.addAction(/* cancel button action */)

alert.addAction(UIAlertAction(
    title: “Login”,
    style: .Default)
    { (action: UIAlertAction) -> Void in
        // get password and log in
        let tf = self.alert.textFields?.first as? UITextField
        if tf != nil { self.loginWithPassword(tf.text) }
    }
)

alert.addTextFieldWithConfigurationHandler { (textField) in
    textField.placeholder = “Guidance System Password”
}
```



```
var alert = UIAlertController(
    title: “Login Required”,
    message: “Please enter your Cassini guidance system...”,
    preferredStyle: UIAlertControllerStyle.Alert
)

alert.addAction(/* cancel button action */)

alert.addAction(UIAlertAction(
    title: “Login”,
    style: .Default)
    { (action: UIAlertAction) -> Void in
        // get password and log in
        let tf = self.alert.textFields?.first as? UITextField
        if tf != nil { self.loginWithPassword(tf.text) }
    }
)

alert.addTextFieldWithConfigurationHandler { (textField) in
    textField.placeholder = “Guidance System Password”
}

presentViewController(alert, animated: true, completion: nil)
```







CS193p
Winter 2015

NSTimer

- Setting up a timer to call a method periodically
  - You can set it up to go off once at at some time in the future, or to repeatedly go off
  - If repeatedly, the system will not guarantee exactly when it goes off, so this is not "real-time"
  - But for most UI "order of magnitude" activities, it's perfectly fine
  - We don't use it for "animation" (more on that later)
  - It's more for larger-grained activities
- Run loops
  - Timers work with run loops (which we have not and will not talk about)
  - So for your purposes, you can only use `NSTimer` on the main queue
  - Check out the documentation if you want to learn about run loops and timers on other queue

NSTimer

- Firing one off …

```
class func scheduledTimerWithTimeInterval(
    _ seconds: NSTimeInterval,
    target: AnyObject,
    selector: Selector (String),
    userInfo: AnyObject?,
    repeats: Bool
)
```

NSTimer

- Example

```
let timer = NSTimer.scheduledTimerWithTimeInterval(2.0
    target: self, selector: “fire:”, // you must have a colon on the end of the selector!
    userInfo: nil,
    repeats: true
)
```

Every 2 seconds, the method `fire(NSTimer)` will be invoked in `self`.

- What does that fire method look like?

```
func fire(timer: NSTimer) {
    // do whatever you want to do every 2 seconds
    // don’t take too long in here, remember you are on the main queue
    let theTimersUserInfo = timer.userInfo // feeds back the userInfo you set above
}
```

NSTimer

- Stopping a repeating timer

  Just call `invalidate()` on a timer to stop it …

```
func fire(timer: NSTimer) {
    if imDoneWithThisTimer {
        timer.invalidate()
    }
}
```

- Tolerance

  It might help system performance to set a tolerance for "late firing"

  For example, if you have timer that goes off once a minute, a tolerance of 10s might be fine

```
myOneMinuteTimer.tolerance = 10 // in seconds
```

  The firing time is relative to the start of the timer (not the last time it fired), i.e. no "drift"

Kinds of Animation

- Animating UIView properties
  That's what we're going to talk about today.
- Animation of View Controller transitions (like UINC's)
  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).
- Dynamic Animation
  "Physics"-based animation (we'll talk about that next week)

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(s) using closures
  - The class methods takes animation parameters and an animation block as arguments.
  - The animation block contains the code that makes the changes to the UIView(s).
  - The changes inside the block are made immediately (even though they will appear "over time").
  - Most also have another "completion block" to be executed when the animation is done.

UIView Animation

- Animation class method in UIView

```
class func animateWithDuration(duration: NSTimeInterval,
                                 delay: NSTimeInterval,
                               options: UIViewAnimationOptions,
                            animations: () -> Void,
                            completion: ((finished: Bool) -> Void)?)
```

UIView Animation

- Example

```
if myView.alpha = 1.0 {
    UIView.animateWithDuration(3.0
                         delay: 2.0
                       options: UIViewAnimationOptions.CurveEaseInEaseOut
                    animations: { myView.alpha = 0.0 }
                    completion: { if $0 { myView.removeFromSuperview() } })
    println("myView.alpha = \(myView.alpha)")
}
```

This would cause `myView` to "fade" out over 3 seconds (starting 2s from now).
Then it would remove `myView` from the view hierarchy (but only if the fade completed).
If, within the 5s, someone animated the `alpha` to non-zero, the removal would not happen.
The output on the console would be …

```
myView.alpha = 0.0
```

… even though the `alpha` on the screen won't be zero for 5 more 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 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 "bits"
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
  - In this case you are not limited to special properties like `alpha`, `frame` and `transform`
  - Flip the entire view over `UIViewAnimationOptionsTransitionFlipFrom{Left,Right,Top,Bottom}`
  - Dissolve from old to new state `UIViewAnimationOptionsTransitionCrossDissolve`
  - Curling up or down `UIViewAnimationOptionsTransitionCurl{Up,Down}`
- Use closures again with this UIView class method

```
UIView.transitionWithView(view: UIView,
                         duration: NSTimeInterval,
                          options: UIViewAnimationOptions,
                       animations: () -> Void,
                       completion: ((finished: Bool) -> Void)?)
```

UIView Animation

- Example

  Flipping a playing card over …

```
UIView.transitionWithView(view: myPlayingCardView,
                         duration: 0.75,
                          options: UIViewAnimationOptions.TransitionFlipFromLeft,
                       animations: { cardIsFaceUp = !cardIsFaceUp }
                       completion: nil)
```

  Presuming `myPlayingCardView` draws itself face up or down depending on `cardIsFaceUp`

  This will cause the card to flip over (from the left edge of the card)

UIView Animation

- Animating changes to the view hierarchy is slightly different

  In other words, you want to animate the adding/removing of subviews (or (un)hiding them)

```
UIView.transitionFromView(fromView: UIView,
                            toView: UIView,
                          duration: NSTimeInterval,
                           options: UIViewAnimationOptions,
                        completion: ((finished: Bool) -> Void)?)
```

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