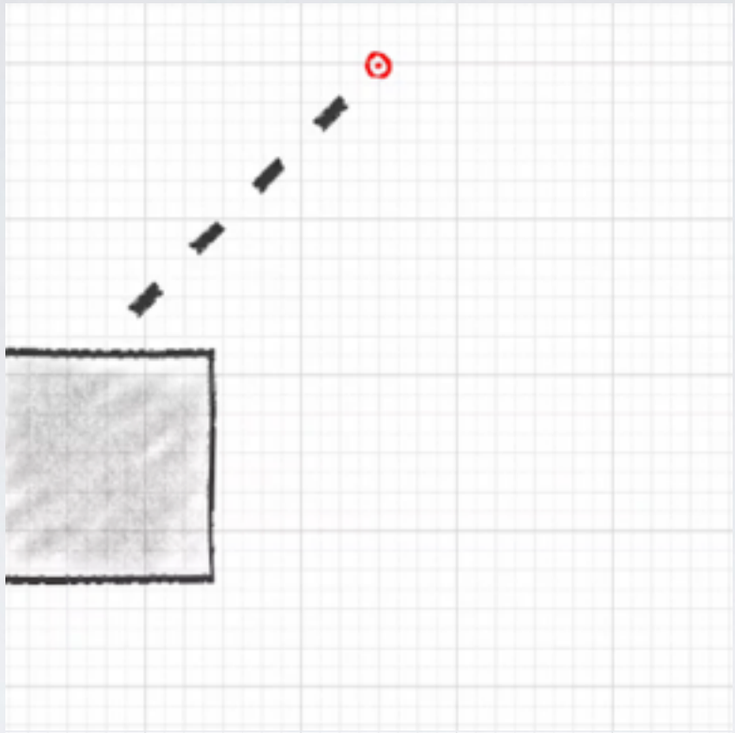
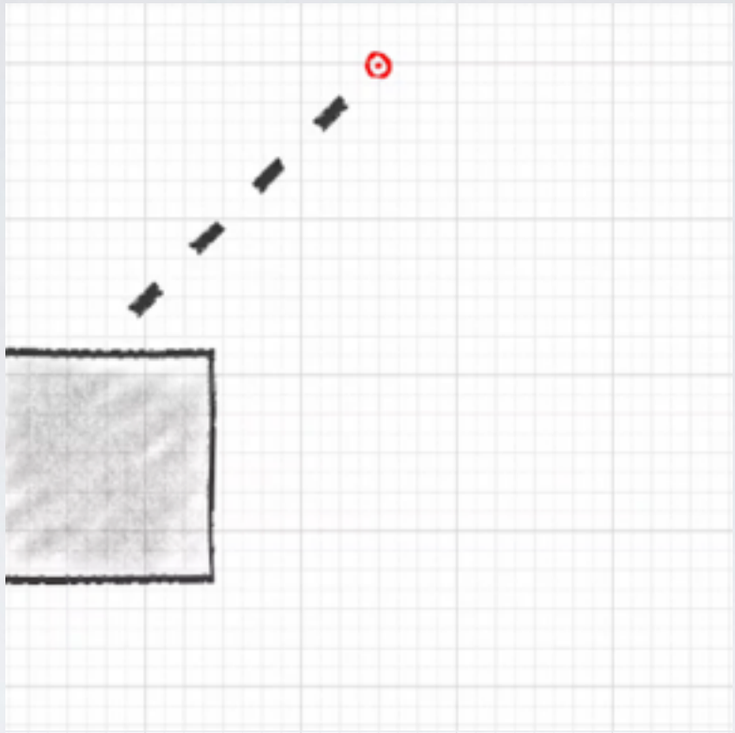


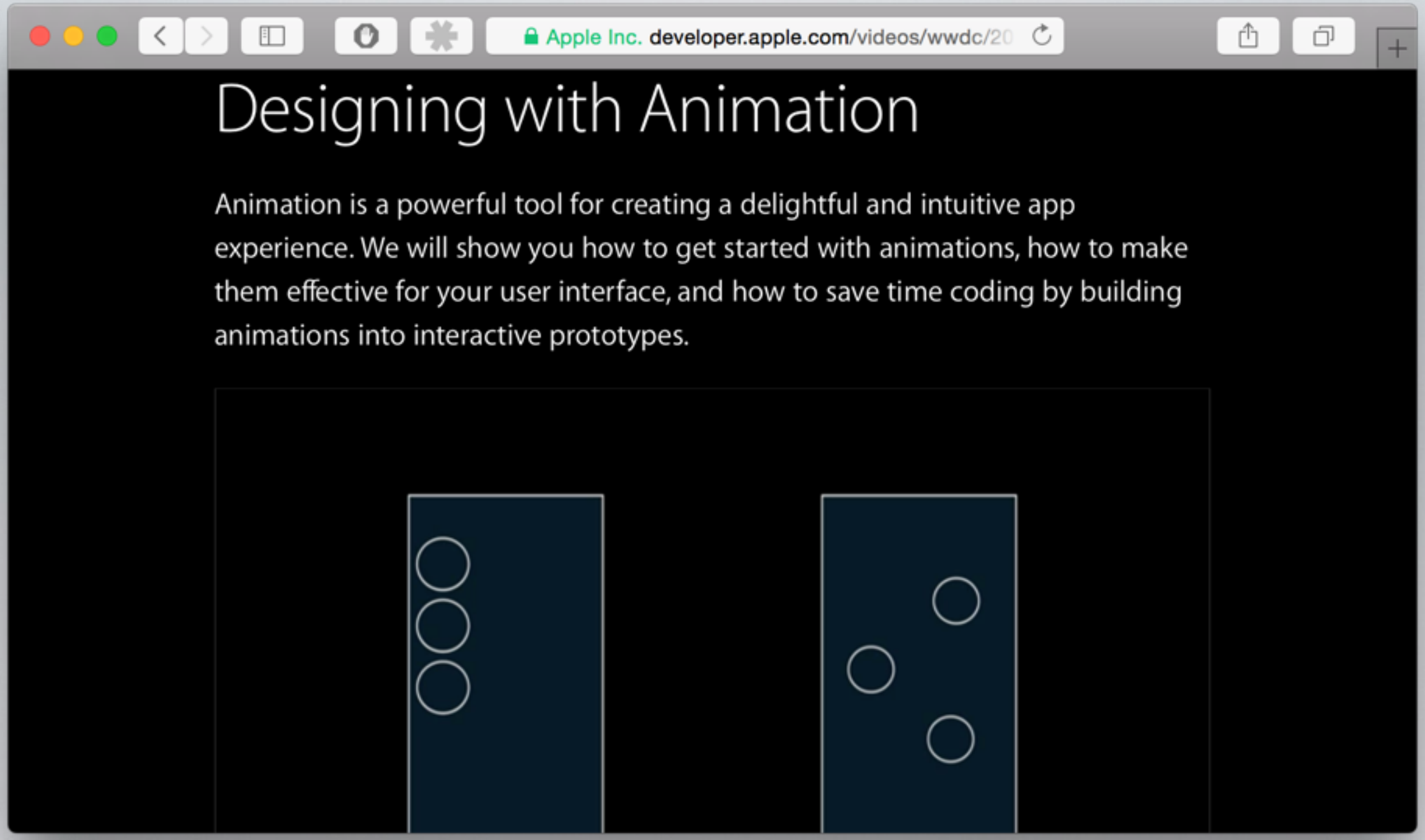
ANIMATION IN IOS BY EXAMPLES

Nick Chen
Fall 2015
talentspark.io



ANIMATION IN IOS BY EXAMPLES

Nick Chen
Fall 2015
talentspark.io



https://developer.apple.com/videos/wwdc/2015/?id=803

ANIMATIONS IN IOS

ANIMATIONS IN IOS

Animating with UIView properties

Animation of UIView transitions

Dynamic Animation

ANIMATIONS IN IOS

Animating with UIView properties

Animation of UIView transitions

Dynamic Animation

Core Animation

ANIMATIONS IN IOS

Animating with UIView properties

Animation of UIView transitions

Dynamic Animation

Core Animation

SpriteKit

ANIMATIONS IN IOS

Animating with UIView properties

Animation of UIView transitions

Dynamic Animation

Core Animation

SpriteKit

Metal

ANIMATIONS IN IOS

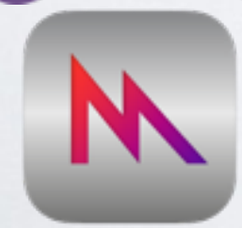
Animating with UIView properties ✓

Animation of UIView transitions

Dynamic Animation

Core Animation

SpriteKit



Metal

ANIMATIONS IN IOS

Animating with UIView properties ✓

Animation of UIView transitions

Dynamic Animation ✓

Core Animation

SpriteKit

Metal

ANIMATING UIVIEW PROPERTIES

UIVIEW PROPERTIES

Apple Inc. developer.apple.com/libra

iOS Developer Library — Prerelease

Developer

View Programming Guide for iOS

PDF

Table of Contents

Table 4-1 Animatable `UIView` properties

Property	Changes you can make
`frame`	Modify this property to change the view's size and position relative to its superview's coordinate system. (If the `transform` property does not contain the identity transform, modify the `bounds` or `center` properties instead.)
`bounds`	Modify this property to change the view's size.
`center`	Modify this property to change the view's position relative to its superview's coordinate system.
`transform`	Modify this property to scale, rotate, or translate the view relative to its center point. Transformations using this property are always performed in 2D space. (To perform 3D transformations, you must animate the view's layer object using Core Animation.)
`alpha`	Modify this property to gradually change the transparency of the view.
`backgroundColor`	Modify this property to change the view's background color.
`contentStretch`	Modify this property to change the way the view's contents are stretched to fill the available sp

Feedback

UIVIEW PROPERTIES

UIView Class Referen

animatewithduration

- animateWithDuration:anim…
- animateWithDuration(_:an…
- animateWithDuration:anim…
- animateWithDuration(_:an…
- animateWithDuration:dela…
- animateWithDuration(_:de…
- animateWithDuration:dela…
- animateWithDuration(_:de…
- animateKeyframesWithDura…
- animateKeyframesWithDura…

Managing Gesture Recognizers

- \- gestureRecognizerShouldBegi…

Animating Views with Block Objects

- + animateWithDuration(_:delay…
- + animateWithDuration(_:anima…
- + animateWithDuration(_:anima…
- + transitionWithView(_:durati…
- + transitionFromView(_:toView…
- + animateKeyframesWithDuratio…
- + addKeyframeWithRelativeStar…

UIView Class Refe… › + animateWithDurat…

animateWithDuration(_:animations:completion:)

Animate changes to one or more views using the specified duration and completion handler.

Declaration

SWIFT

```
class func animateWithDuration(_ duration: NSTimeInterval,
animations animations: () -> Void, completion completion:
((Bool) -> Void)?)
```

Parameters

duration	The total duration of the animations, measured in seconds. If you specify a negative value or `0`, the changes are made without animating them.
animations	A block object containing the changes to commit to the views. This is where you programmatically change any animatable properties of the views in your view hierarchy. This block takes no parameters and has no

UIVIEW PROPERTIES

UIViewAnimateWithDuration: **Ready** | Today at 10:09 PM

UIViewAnimateWithDuration 〉 UIViewAnimateWithDuration 〉 AlphaExampleViewController.swift 〉 No Selection

```
override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
}

@IBAction func fadeIn(sender: UIButton) {
    UIView.animateWithDuration(1.0, animations: {
        self.textField.alpha = 100
        }, completion: { finished in
            if(finished) {
                self.fadeInButton.enabled = false
                self.fadeOutButton.enabled = true
            }
    })
}

@IBAction func fadeOut(sender: UIButton) {
    UIView.animateWithDuration(1.0, animations: {
        self.textField.alpha = 0
        }, completion: { finished in
            if(finished) {
                self.fadeOutButton.enabled = false
```

UIVIEW PROPERTIES

Does the animation in 1.0 second

```
override func didReceiv
    super.didReceiveMem
}

@IBAction func fadeIn(sender: UIButton) {
    UIView.animateWithDuration(1.0, animations: {
        self.textField.alpha = 100
        }, completion: { finished in
            if(finished) {
                self.fadeInButton.enabled = false
                self.fadeOutButton.enabled = true
            }
    })
}

@IBAction func fadeOut(sender: UIButton) {
    UIView.animateWithDuration(1.0, animations: {
        self.textField.alpha = 0
        }, completion: { finished in
            if(finished) {
                self.fadeOutButton.enabled = false
```

UIVIEW PROPERTIES

UIViewAnimateWi

UIViewAnimateWithDuration 〉 UIView.

```
override func didReceiv
    super.didReceiveMem
}

@IBAction func fadeIn(sender: UIButton) {
    UIView.animateWithDuration(1.0, animations: {
        self.textField.alpha = 100
        }, completion: { finished in
            if(finished) {
                self.fadeInButton.enabled = false
                self.fadeOutButton.enabled = true
            }
    })
}

@IBAc
    U                                   {
        }, completion: { finished in
            if(finished) {
                self.fadeOutButton.enabled = false
```

Does the animation in 1.0 second

After completing, perform these actions

UIVIEW PROPERTIES

UIViewAnimateWithDuration: **Ready** | Today at 10:09 PM

UIViewAnimateWithDuration 〉 UIViewAnimateWithDuration 〉 AlphaExampleViewController.swift 〉 No Selection

```
override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
}

@IBAction func fadeIn(sender: UIButton) {
    UIView.animateWithDuration(1.0, animations: {
        self.textField.alpha = 100
        }, completion: { finished in
            if(finished) {
                self.fadeInButton.enabled = false
                self.fadeOutButton.enabled = true
            }
    })
}

@IBAction func fadeOut(sender: UIButton) {
    UIView.animateWithDuration(1.0, animations: {
        self.textField.alpha = 0
        }, completion: { finished in
            if(finished) {
                self.fadeOutButton.enabled = false
```

UIVIEW PROPERTIES

UIView Class Referen

animatewithduration

- animateWithDuration:anim…
- animateWithDuration(_:an…
- animateWithDuration:anim…
- animateWithDuration(_:an…
- animateWithDuration:dela…
- animateWithDuration(_:de…
- animateWithDuration:dela…
- animateWithDuration(_:de…
- animateKeyframesWithDura…
- animateKeyframesWithDura…

Managing Gesture Recognizers

- \- gestureRecognizerShouldBegi…

Animating Views with Block Objects

- + animateWithDuration(_:delay…
- + animateWithDuration(_:anima…
- + animateWithDuration(_:anima…
- + transitionWithView(_:durati…
- + transitionFromView(_:toView…
- + animateKeyframesWithDuratio…
- + addKeyframeWithRelativeStar…

UIView Class Refe… › + animateWithDurat…

Animating Views with Block Objects

`animateWithDuration(_:delay:options:animations:completion:)`

Animate changes to one or more views using the specified duration, delay, options, and completion handler.

Declaration

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

Parameters

duration	The total duration of the animations, measured in seconds. If you specify a negative value or `0`, the changes are made without animating them.

UIVIEW PROPERTIES

UIView Class Referen

animatewithduration

- animateWithDuration:anim…
- animateWithDuration(_:an…
- animateWithDuration:anim…
- animateWithDuration(_:an…
- animateWithDuration:dela…
- animateWithDuration(_:de…
- animateWithDuration:dela…
- animateWithDuration(_:de…
- animateKeyframesWithDura…
- animateKeyframesWithDura…

Managing Gesture Recognizers

- \- gestureRecognizerShouldBegi…

Animating Views with Block Objects

- + animateWithDuration(_:delay…
- + animateWithDuration(_:anima…
- + animateWithDuration(_:anima…
- + transitionWithView(_:durati…
- + transitionFromView(_:toView…
- + animateKeyframesWithDuratio…
- + addKeyframeWithRelativeStar…

UIView Class Refe… › + animateWithDurat…

Animating Views with Block Objects

`animateWithDuration(_:delay:options:animations:completion:)`

Animate … y, options, …

This just has more options

Declarat…

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

Parameters

duration	The total duration of the animations, measured in seconds. If you specify a negative value or 0, the changes are made without animating them.

WHAT ABOUT AUTO LAYOUT?

Finished running UIViewAnimateWithDuration on iPhone 6

UIViewAnimateWithDuration 〉 UIViewAnimateWithDuration 〉 AutoLayoutExampleViewController.swift 〉 AutoLayoutExampleViewController

```
//  AutoLayoutExampleViewController.swift
//  UIViewAnimateWithDuration
//
//  Created by Nick Chen on 9/2/15.
//  Copyright © 2015 Nick Chen. All rights reserved.
//

import UIKit

class AutoLayoutExampleViewController: UIViewController {

    @IBOutlet weak var rightViewConstraint: NSLayoutConstraint!
    @IBOutlet weak var leftRightGapConstraint: NSLayoutConstraint!

    var hasSlideYet = false

    override func viewDidLoad() {
        super.viewDidLoad()
        updateConstraints()
    }
```

WHAT ABOUT AUTO LAYOUT?

Finished running UIViewAnimateWithDuration on iPhone 6

UIViewAnimateWithDuration 〉 UIViewAnimateWithDuration 〉 AutoLayoutExampleViewController.swift 〉 AutoLayoutExampleViewController

```
//  AutoLayoutExampleViewController.swift
//  UIViewAnimateWithDuration
//
//  Created by Nick Chen on 9/2/15.
//  Copyright © 2015 Nick Chen. All rights reserved.
//

import UIKit

class AutoLayoutExampleViewController: UIViewController {

    @IBOutlet weak var rightViewConstraint: NSLayoutConstraint!
    @IBOutlet weak var leftRightGapConstraint: NSLayoutConstraint!

    var hasSlideYet

    override func v
        super.viewD
        updateConst
    }
```

Yes, you can have IBOutlets to constraints!

WHAT ABOUT AUTO LAYOUT?

Finished running UIViewAnimateWithDuration on iPhone 6

UIViewAnimateWithDuration 〉 UIViewAnimateWithDuration 〉 AutoLayoutExampleViewController.swift 〉 updateConstraints()

```
@IBAction func slideToggle(sender: UIButton) {
    hasSlideYet = !hasSlideYet
    UIView.animateWithDuration(1.0, animations: {
        self.updateConstraints()
        self.view.layoutIfNeeded()
    })
}

func updateConstraints() {
    if(hasSlideYet) {
        // slide it off
        leftRightGapConstraint.constant = 20.0
        rightViewConstraint.priority = UILayoutPriorityDefaultHigh - 1
    } else {
        // slide it back
        leftRightGapConstraint.constant = NSLayoutConstraint.
            standardConstantBetweenSiblings()
        rightViewConstraint.priority = UILayoutPriorityDefaultHigh + 1
    }
}
```

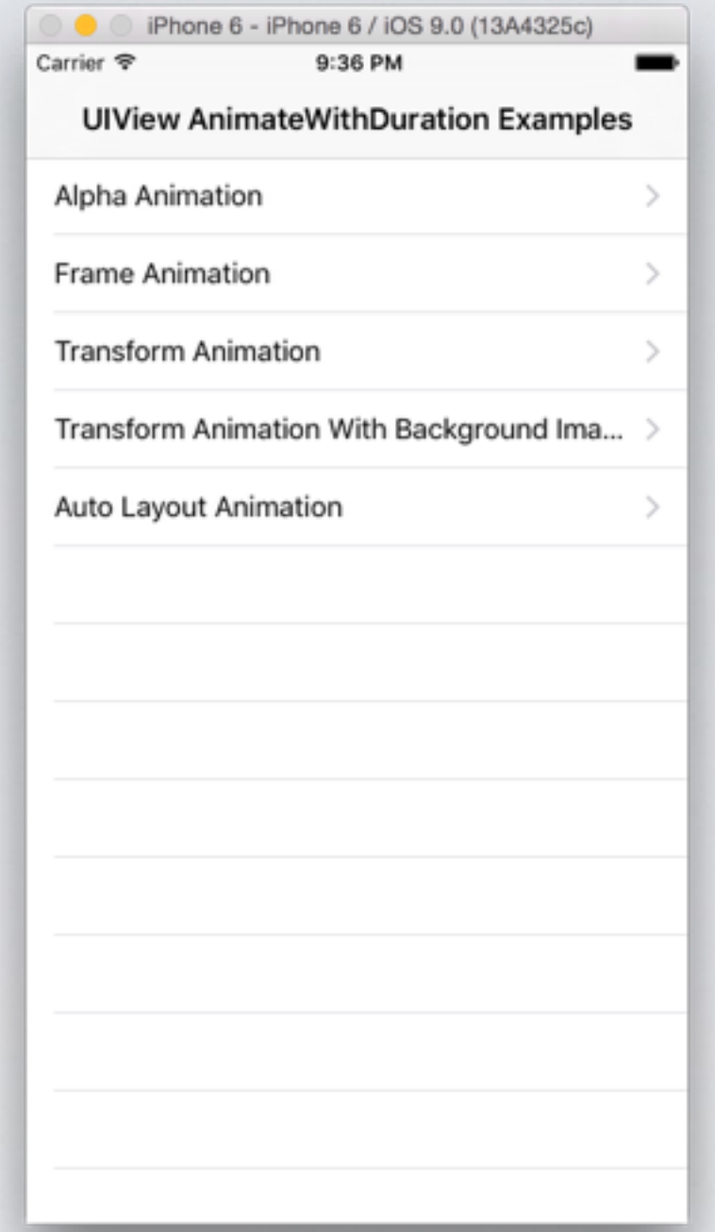
WHAT ABOUT AUTO LAYOUT?

Finished running UIViewAnimateWithDuration on iPhone 6

UIViewAnimateWithDuration 〉 UIViewAnimateWithDuration 〉 AutoLayoutExampleViewController.swift 〉 updateConstraints()

```
@IBAction func slideToggle(sender: UIButton) {
    hasSlideYet = !hasSlideYet
    UIView.animateWithDuration(1.0, animations: {
        self.updateConstraints()
        self.view.layoutIfNeeded()
    })
}

func updateConstraints() {
    if(hasSlideYet) {
        // slide it off
        leftRightGapConstraint.constant = 20.0
        rightViewConstraint.priority = UILayoutPriorityDefaultHigh - 1
    } else {
        // slide it back
        leftRightGapConstraint.co
            standardConstantBetwe
        rightViewConstraint.prior
    }
}
```

The tricks is to change the priorities on the constraints



DEMO

https://github.com/talentsparkio/UIViewAnimateWithDuration

UIKIT DYNAMICS

UIKIT DYNAMICS

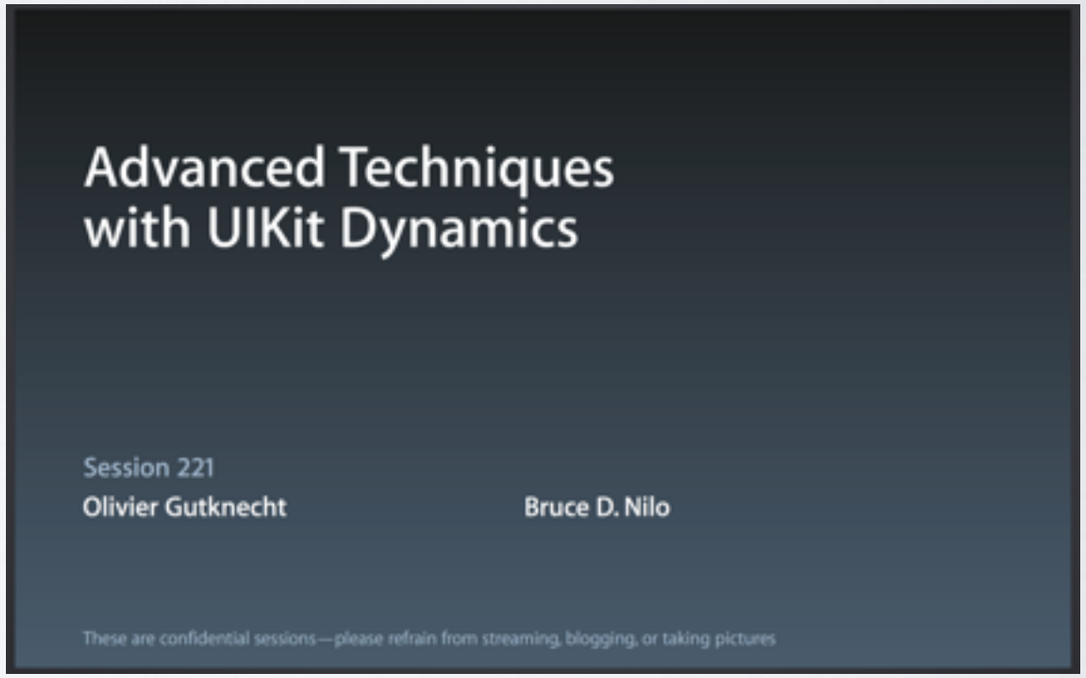
A composable, reusable, declarative, real-world inspired animation, and interaction system

UIKIT DYNAMICS

A composable, reusable, declarative, real-world inspired animation, and interaction system

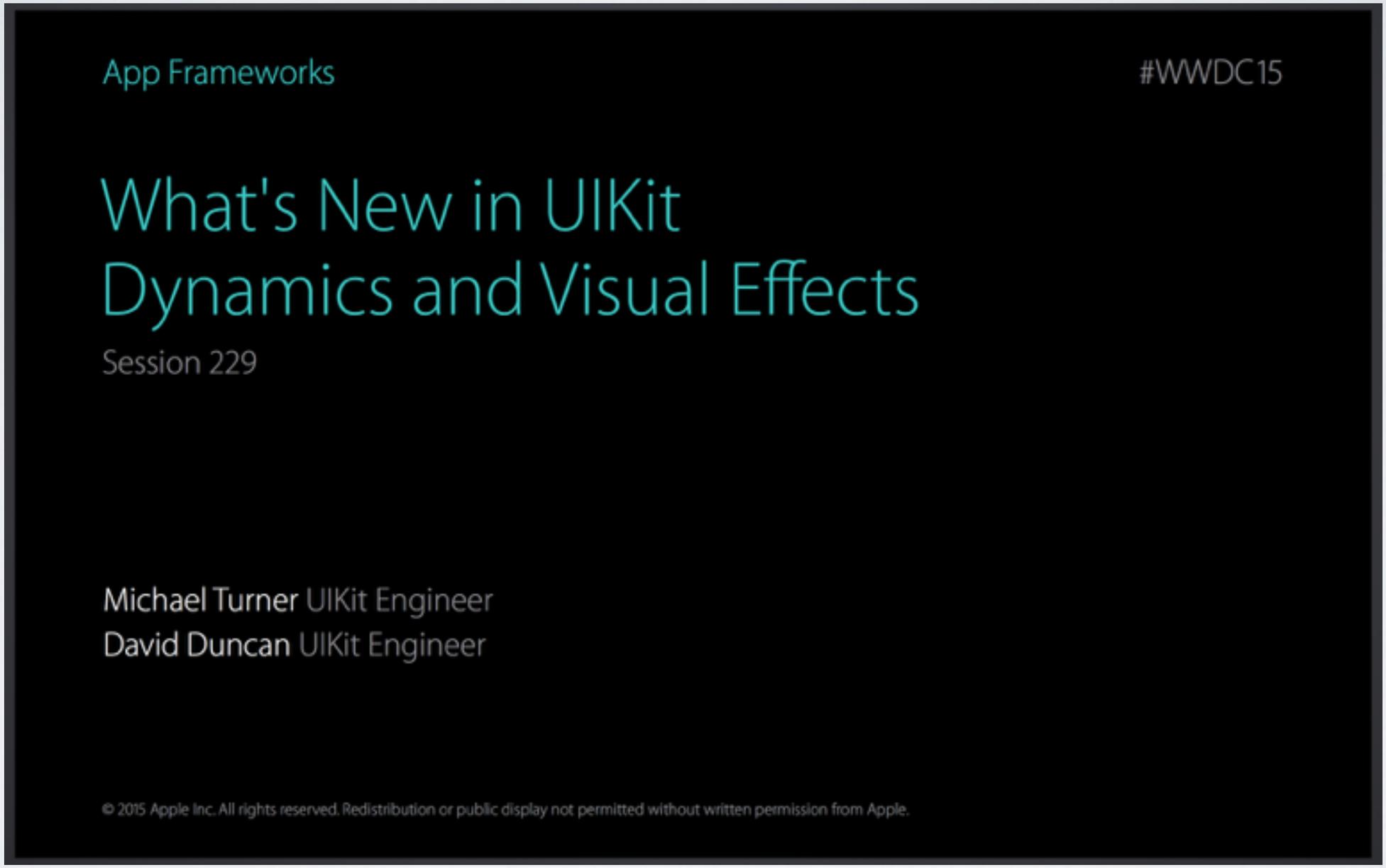
Pretty darn good fake physics that's easy to set up

UIKIT DYNAMICS



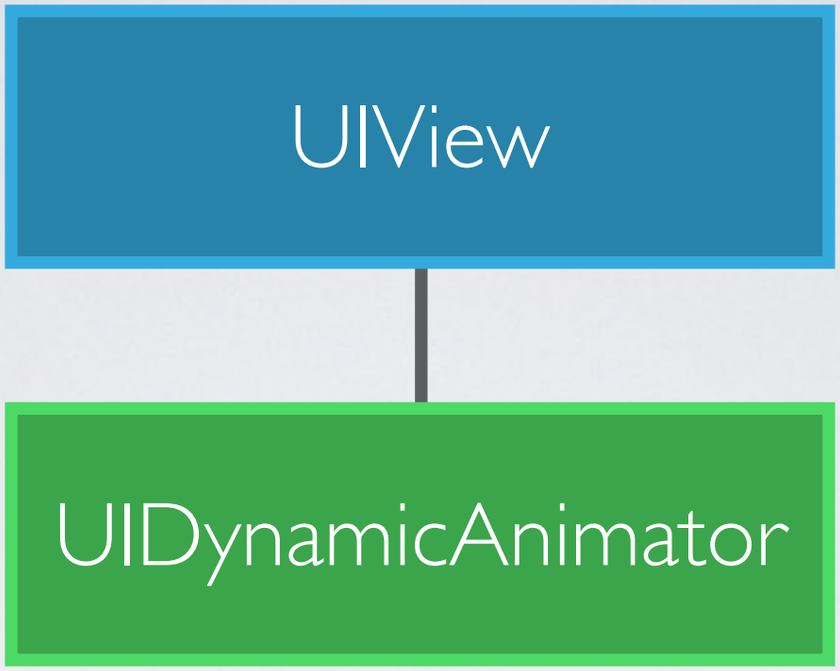
https://developer.apple.com/videos/wwdc/2013/

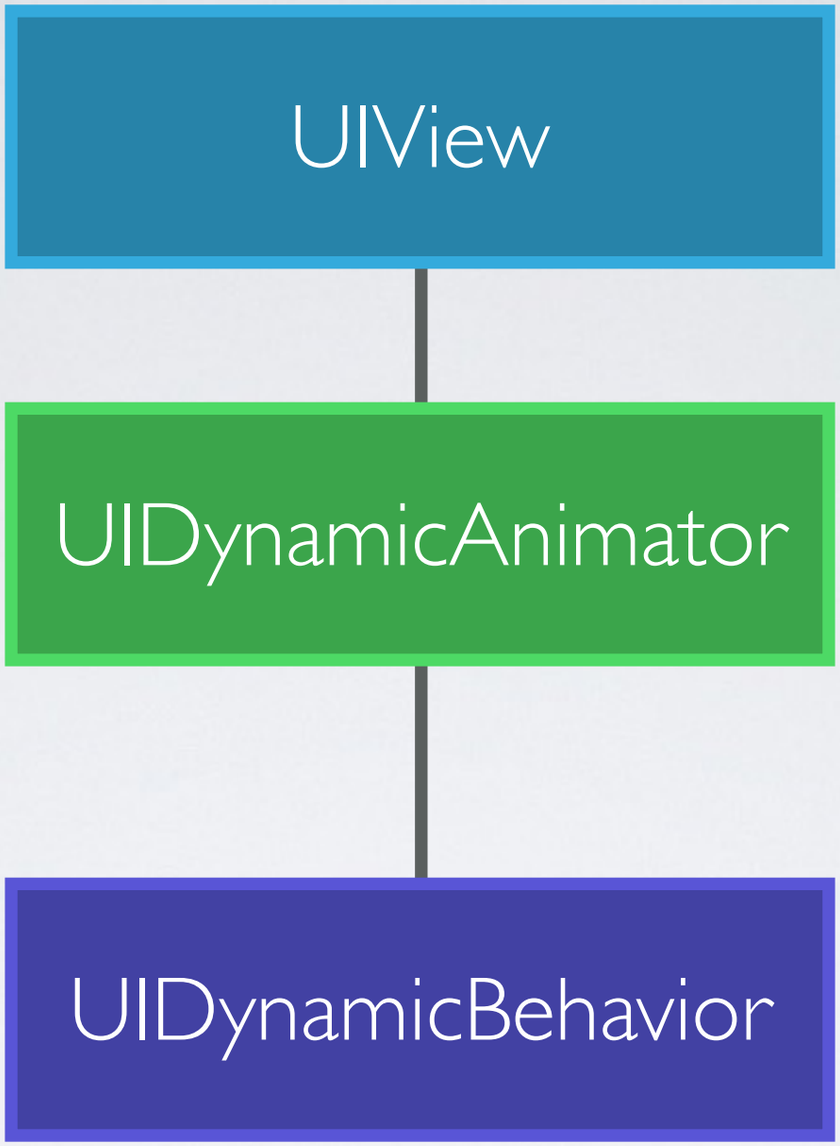
UIKIT DYNAMICS

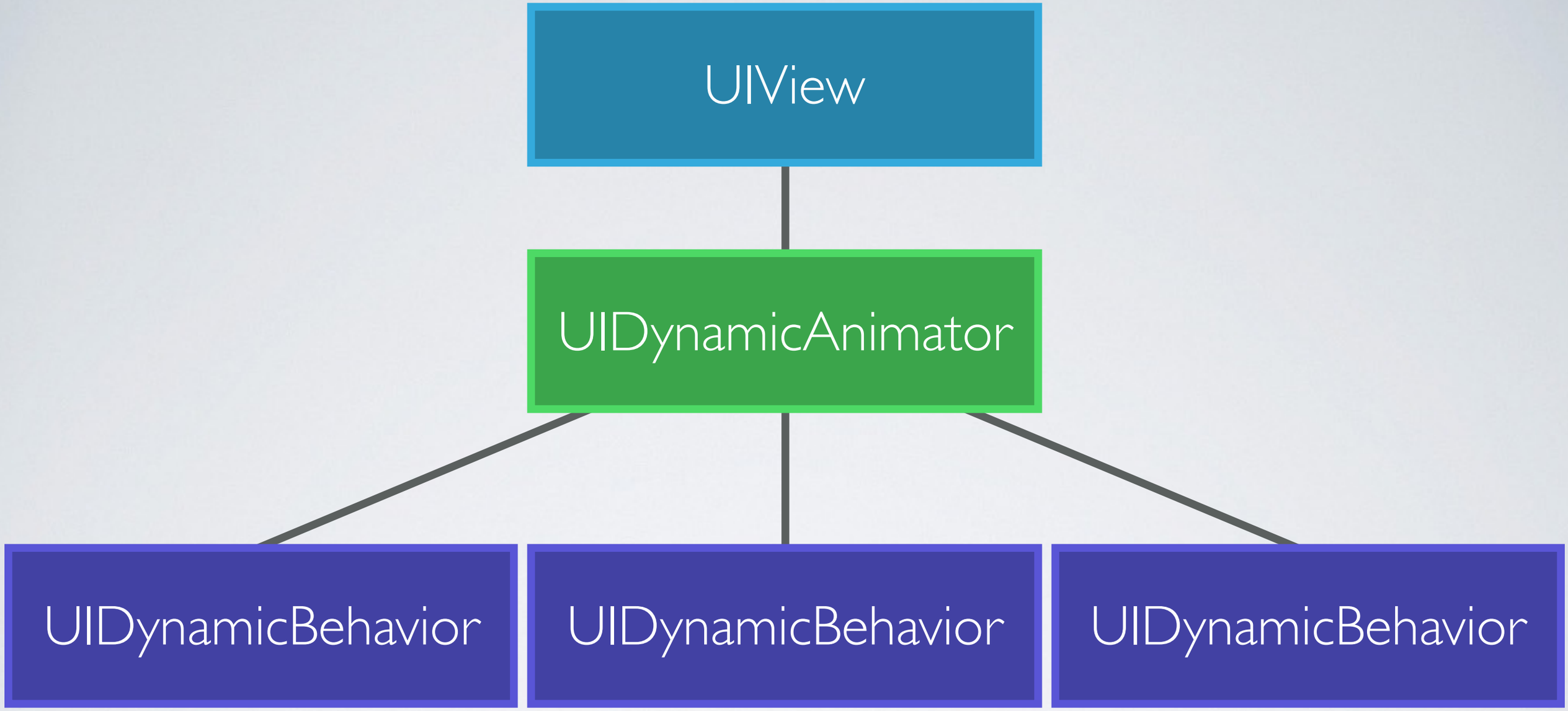


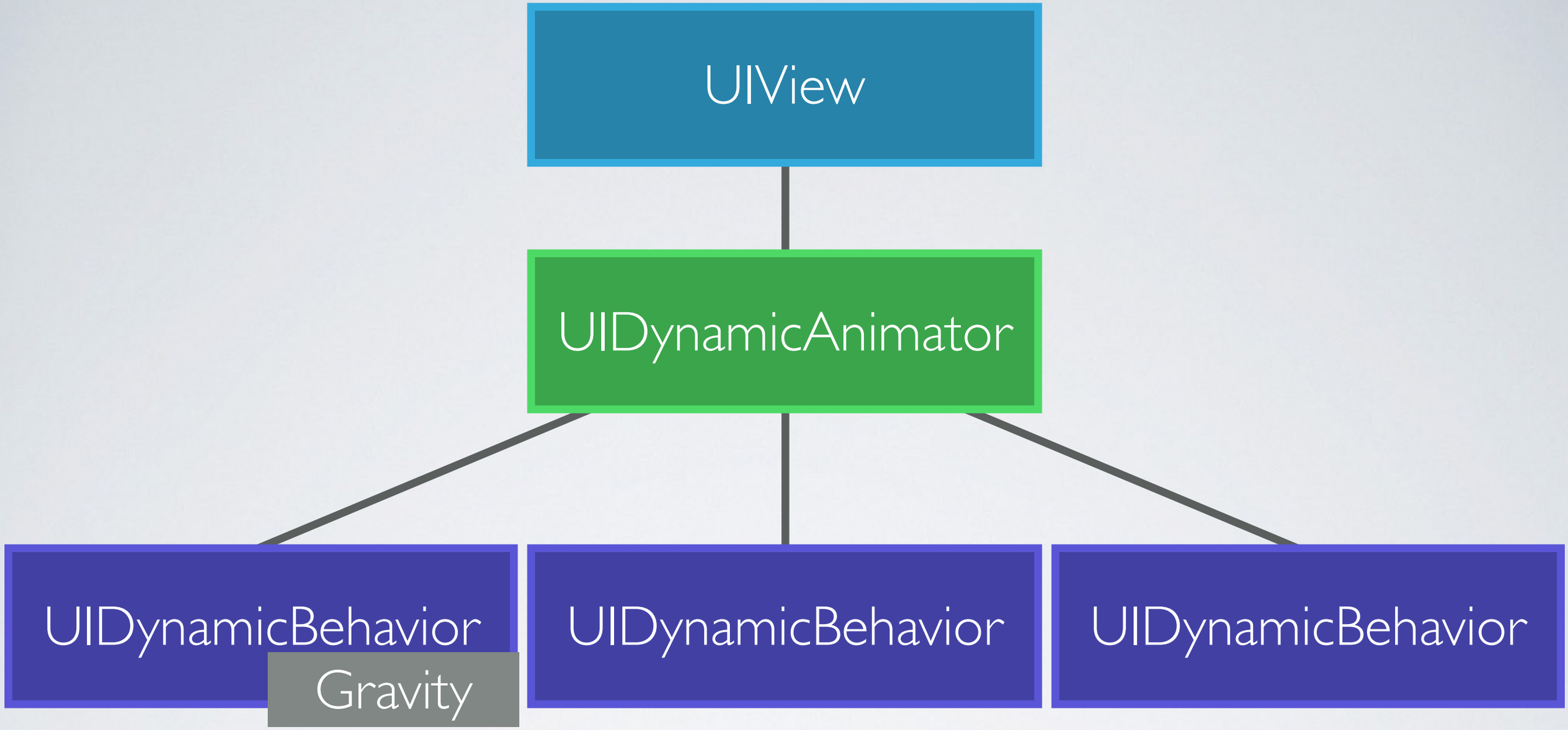
https://developer.apple.com/videos/wwdc/2015/?id=229

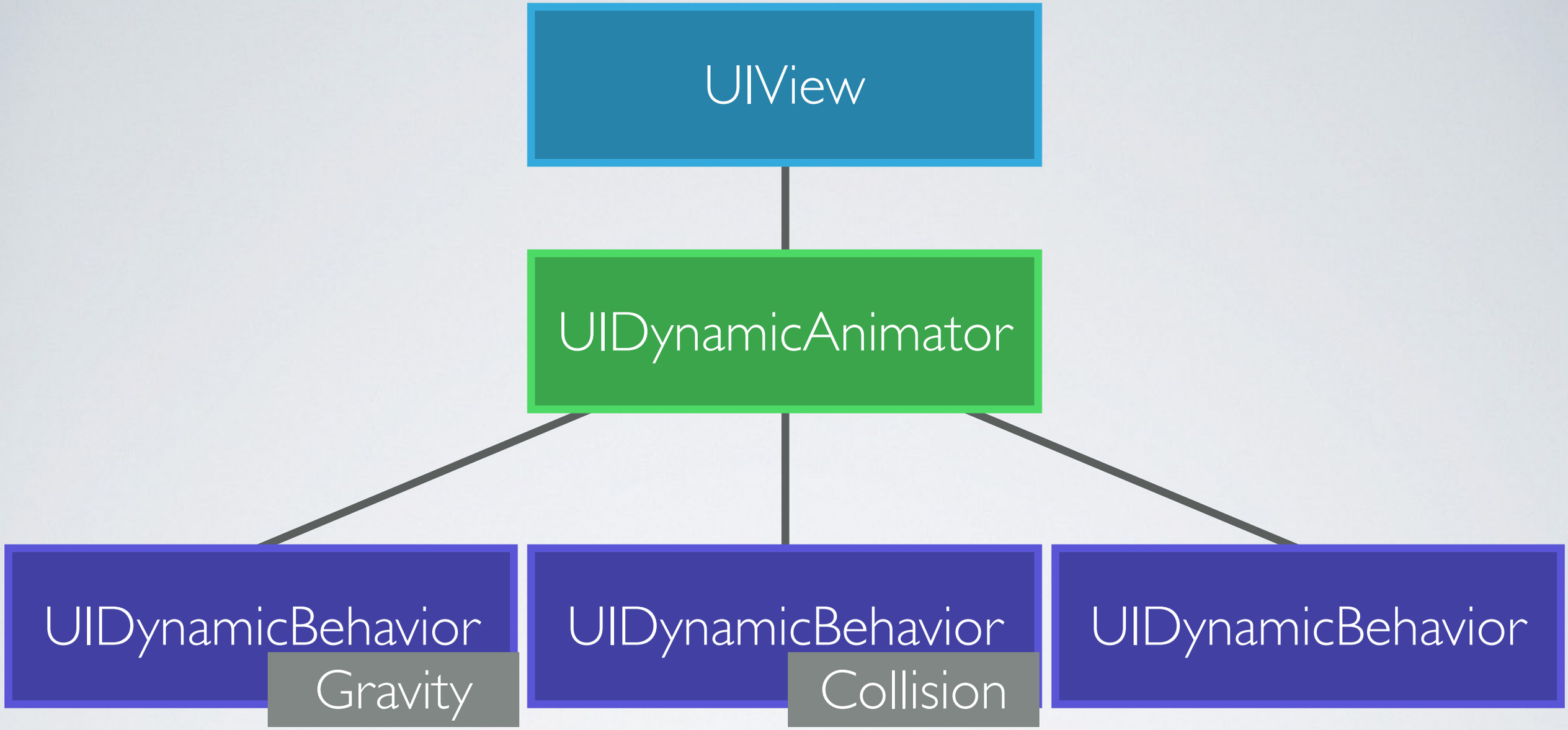
UIView

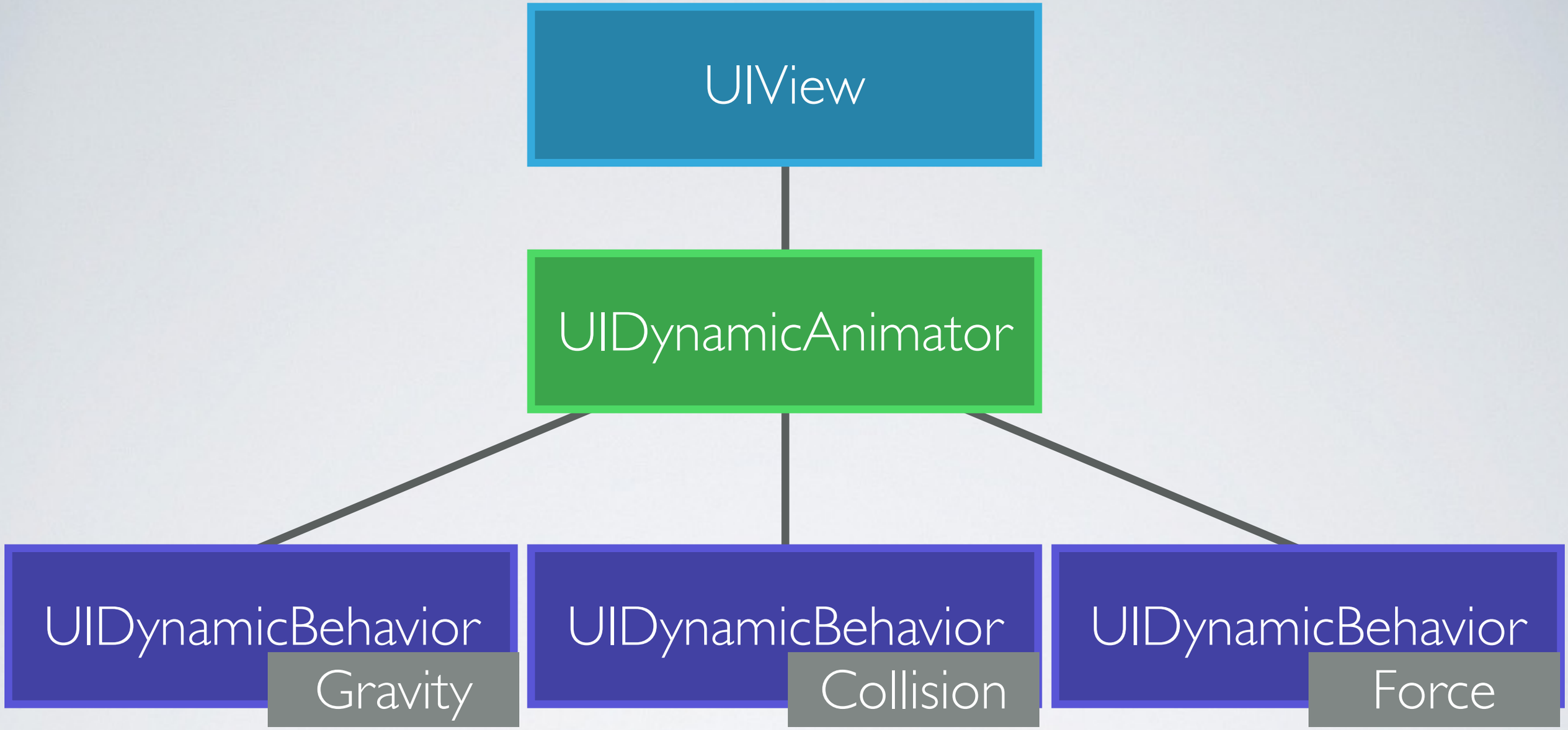

UIView
UIDynamicAnimator

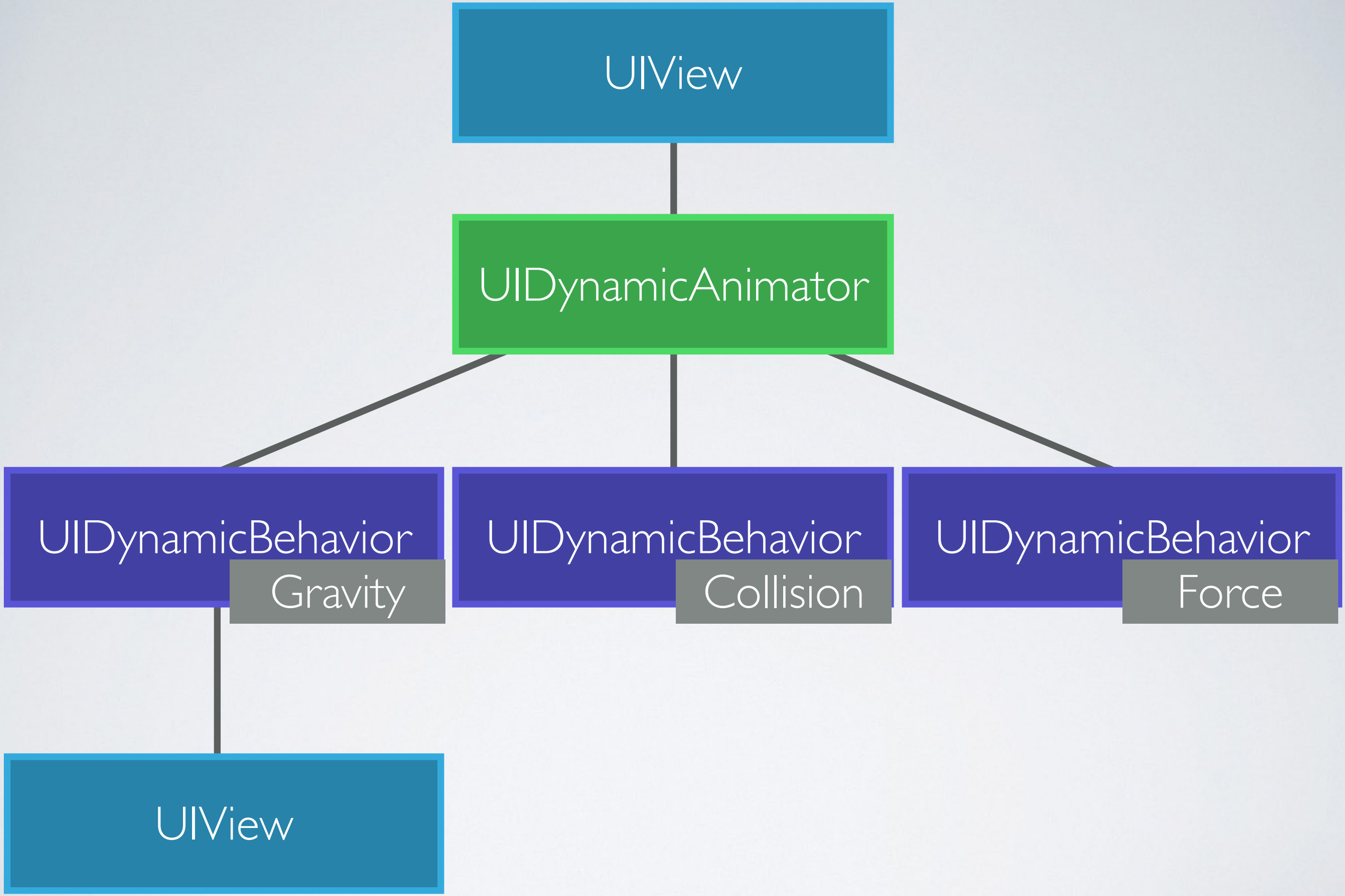

UIView
UIDynamicAnimator
UIDynamicBehavior

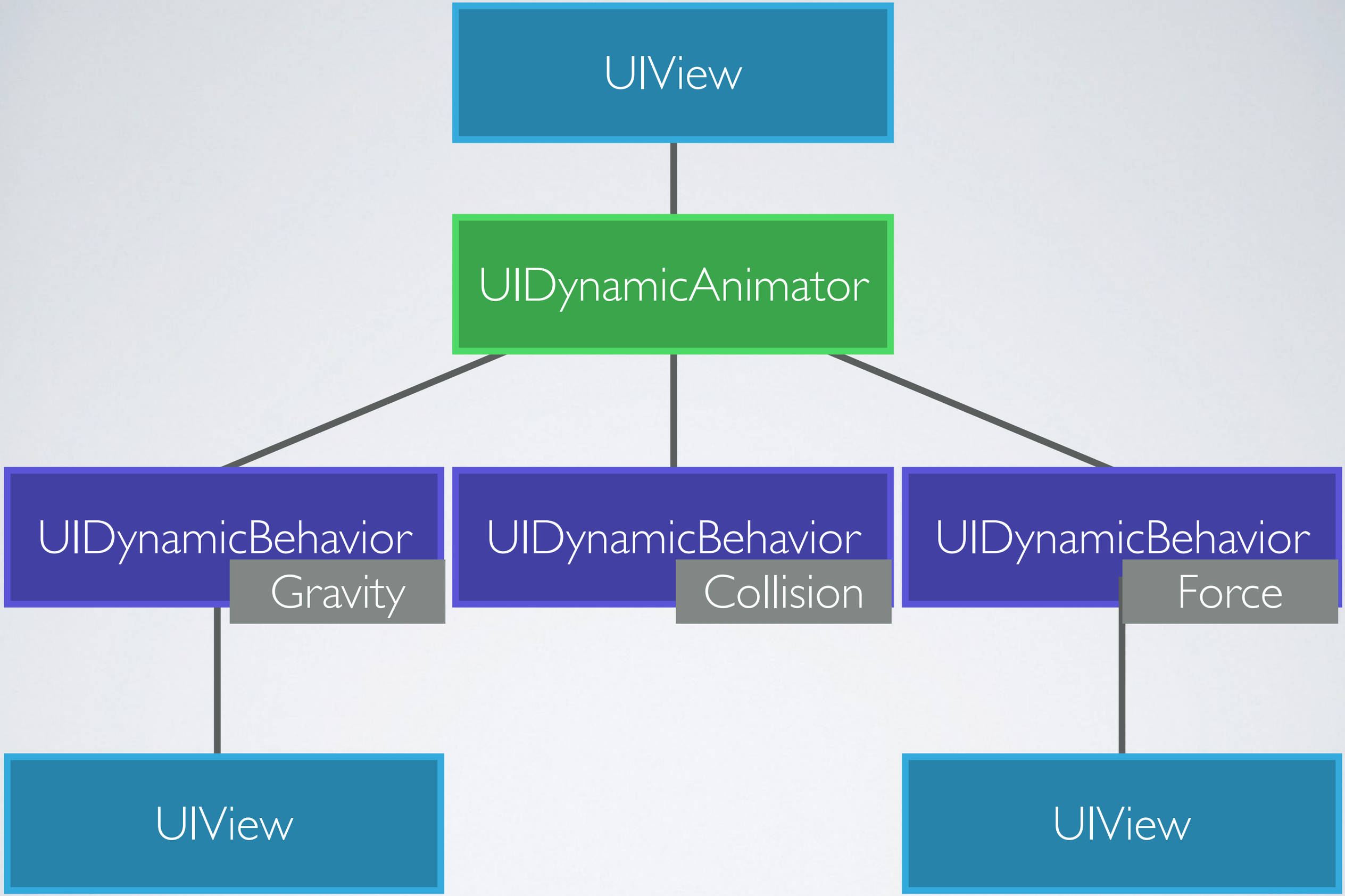

UIView
UIDynamicAnimator
UIDynamicBehavior
UIDynamicBehavior
UIDynamicBehavior

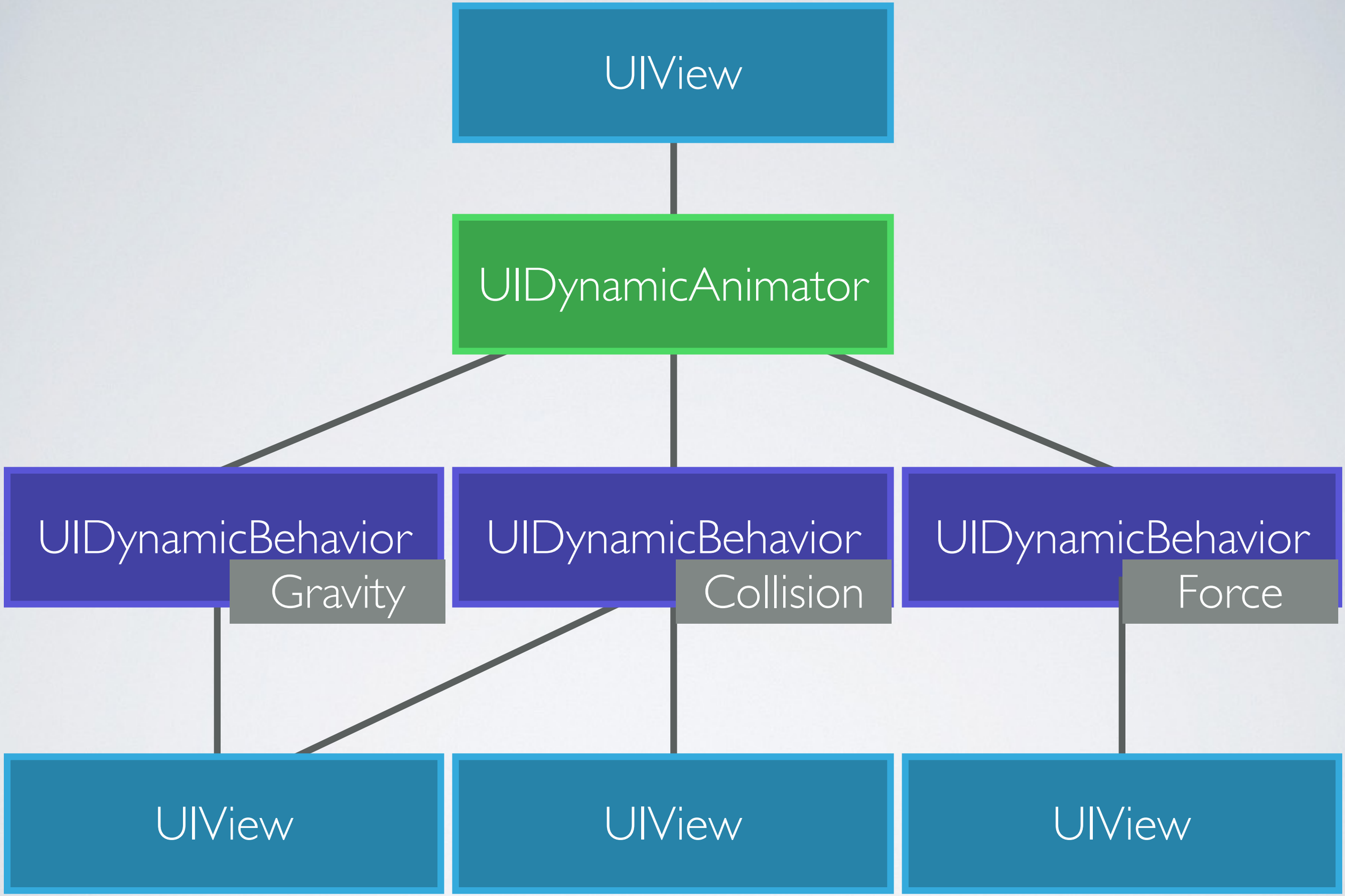

UIView
UIDynamicAnimator
UIDynamicBehavior
Gravity
UIDynamicBehavior
UIDynamicBehavior


UIView
UIDynamicAnimator
UIDynamicBehavior
Gravity
UIDynamicBehavior
Collision
UIDynamicBehavior


UIView
UIDynamicAnimator
UIDynamicBehavior
Gravity
UIDynamicBehavior
Collision
UIDynamicBehavior
Force


UIView
UIDynamicAnimator
UIDynamicBehavior
Gravity
UIDynamicBehavior
Collision
UIDynamicBehavior
Force
UIView


UIView
UIDynamicAnimator
UIDynamicBehavior
Gravity
UIDynamicBehavior
Collision
UIDynamicBehavior
Force
UIView
UIView


UIView
UIDynamicAnimator
UIDynamicBehavior
Gravity
UIDynamicBehavior
Collision
UIDynamicBehavior
Force
UIView
UIView
UIView

EXAMPLE

Finished running Pong on iPhone 6

Pong 〉 Pong 〉 ViewController.swift 〉 collisionBehavior(_:beganContactForItem:withItem:atPoint:)

```swift
func initBehaviors() {
    animator = UIDynamicAnimator(referenceView: self.view)

    // Add gravity for ball
    let gravityBehavior = UIGravityBehavior(items: [orangeBall])
    animator.addBehavior(gravityBehavior)

    // Add collision
    let collisionBoundsBehavior = UICollisionBehavior(items: [orangeBall,
        paddle])
    collisionBoundsBehavior.translatesReferenceBoundsIntoBoundary = true
    animator.addBehavior(collisionBoundsBehavior)

    // Add physical properties for ball
    let elasticityBehavior = UIDynamicItemBehavior(items: [orangeBall])
    elasticityBehavior.elasticity = 1.0
    animator.addBehavior(elasticityBehavior)

    // Add physical properties for bricks
    let brickBehavior = UIDynamicItemBehavior(items: bricks)
```

EXAMPLE

Finished running Pong on iPhone 6

Pong 〉 Pong 〉 ViewController.swift 〉 collisionBehavior(_:beganContactForItem:withItem:atPoint:)

```
func initBehaviors() {
    animator = UIDynamicAnimator(referenceView: self.view)

    // A
    let (                                                    [orangeBall])
    anima

    // A
    let collisionBoundsBehavior = UICollisionBehavior(items: [orangeBall,
        paddle])
    collisionBoundsBehavior.translatesReferenceBoundsIntoBoundary = true
    animator.addBehavior(collisionBoundsBehavior)

    // Add physical properties for ball
    let elasticityBehavior = UIDynamicItemBehavior(items: [orangeBall])
    elasticityBehavior.elasticity = 1.0
    animator.addBehavior(elasticityBehavior)

    // Add physical properties for bricks
    let brickBehavior = UIDynamicItemBehavior(items: bricks)
```

Create the UIDynamicAnimator

EXAMPLE

Finished running Pong on iPhone 6

Pong 〉 Pong 〉 ViewController.swift 〉 collisionBehavior(_:beganContactForItem:withItem:atPoint:)

```swift
func initBehaviors() {
    animator = UIDynamicAnimator(referenceView: self.view)

    // Add gravity for ball
    let gravityBehavior = UIGravityBehavior(items: [orangeBall])
    animator.addBehavior(gravityBehavior)

    // Add collision
    let collisionBoundsBehavior = UICollisionBehavior(items: [orangeBall,
        paddle])
    collisionBoundsBehavior.translatesReferenceBoundsIntoBoundary = true
    animator.addBehavior(collisionBoundsBehavior)

    // Add physical properties for ball
    let elasticityBehavior = UIDynamicItemBehavior(items: [orangeBall])
    elasticityBehavior.elasticity = 1.0
    animator.addBehavior(elasticityBehavior)

    // Add physical properties for bricks
    let brickBehavior = UIDynamicItemBehavior(items: bricks)
```

EXAMPLE

Finished running Pong on iPhone 6

Pong 〉 Pong 〉 ViewController.swift 〉 collisionBehavior(_:beganContactForItem:withItem:atPoint:)

```
func initBehaviors() {
    animator = UIDynamicAnimator(referenceView: self.view)

    // Add gravity for ball
    let gravityBehavior = UIGravityBehavior(items: [orangeBall])
    animator.addBehavior(gravity
                                ehavior)

    // Add co
    let colli                                   items: [orangeBall,
        paddl
    collision                                   ntoBoundary = true
    animator.addBehavior(collisionBoundsBehavior)

    // Add physical properties for ball
    let elasticityBehavior = UIDynamicItemBehavior(items: [orangeBall])
    elasticityBehavior.elasticity = 1.0
    animator.addBehavior(elasticityBehavior)

    // Add physical properties for bricks
    let brickBehavior = UIDynamicItemBehavior(items: bricks)
```

Add the UIGravityBehavior

EXAMPLE

Finished running Pong on iPhone 6

Pong 〉 Pong 〉 ViewController.swift 〉 collisionBehavior(_:beganContactForItem:withItem:atPoint:)

```swift
func initBehaviors() {
    animator = UIDynamicAnimator(referenceView: self.view)

    // Add gravity for ball
    let gravityBehavior = UIGravityBehavior(items: [orangeBall])
    animator.addBehavior(gravityBehavior)

    // Add collision
    let collisionBoundsBehavior = UICollisionBehavior(items: [orangeBall,
        paddle])
    collisionBoundsBehavior.translatesReferenceBoundsIntoBoundary = true
    animator.addBehavior(collisionBoundsBehavior)

    // Add physical properties for ball
    let elasticityBehavior = UIDynamicItemBehavior(items: [orangeBall])
    elasticityBehavior.elasticity = 1.0
    animator.addBehavior(elasticityBehavior)

    // Add physical properties for bricks
    let brickBehavior = UIDynamicItemBehavior(items: bricks)
```

EXAMPLE

Finished running Pong on iPhone 6

Pong 〉 Pong 〉 ViewController.swift 〉 collisionBehavior(_:beganContactForItem:withItem:atPoint:)

```
func initBehaviors() {
    animator = UIDynamicAnimator(referenceView: self.view)

    // Add gravity for ball
    let gravityBehavior = UIGravityBehavior(items: [orangeBall])
    animator.addBehavior(gravityBehavior)

                                                sionBehavior(items: [orangeBall,

                                                erenceBoundsIntoBoundary = true
                                                ehavior)

    // Add physical properties for ball
    let elasticityBehavior = UIDynamicItemBehavior(items: [orangeBall])
    elasticityBehavior.elasticity = 1.0
    animator.addBehavior(elasticityBehavior)

    // Add physical properties for bricks
    let brickBehavior = UIDynamicItemBehavior(items: bricks)
```

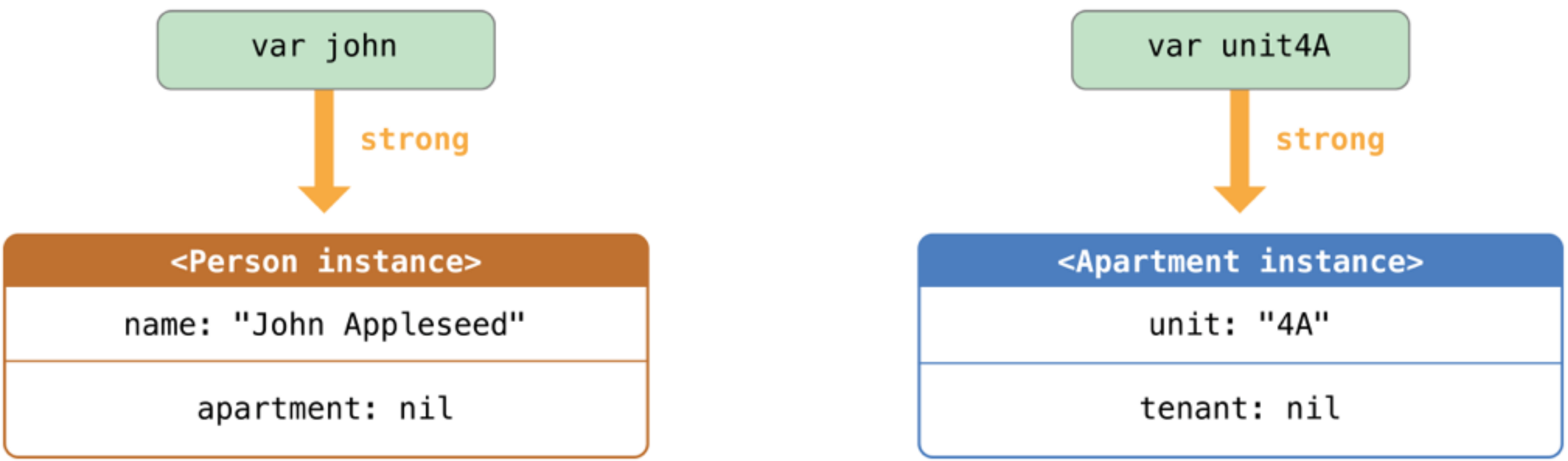
Tie it to the UIDynamicAnimator

PREVENTING MEMORY CYCLE

```
john = Person(name: "John Appleseed")
unit4A = Apartment(unit: "4A")
```

PREVENTING MEMORY CYCLE

```
john = Person(name: "John Appleseed")
unit4A = Apartment(unit: "4A")
```

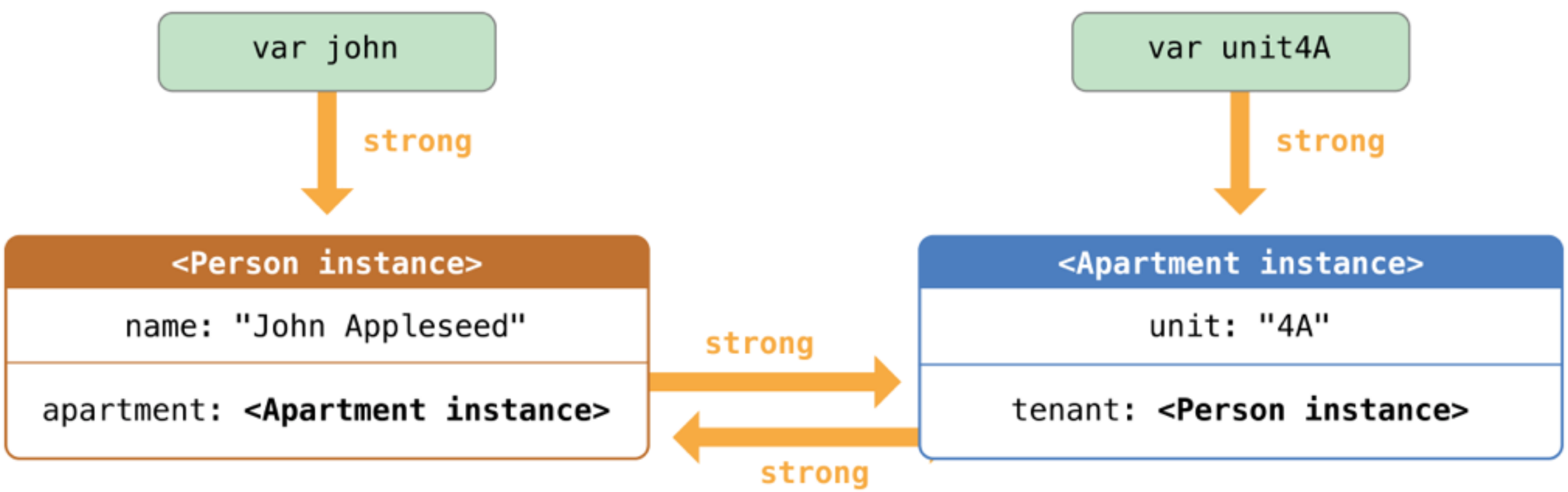


PREVENTING MEMORY CYCLE

```
john!.apartment = unit4A
unit4A!.tenant = john
```

PREVENTING MEMORY CYCLE

```
john!.apartment = unit4A
unit4A!.tenant = john
```

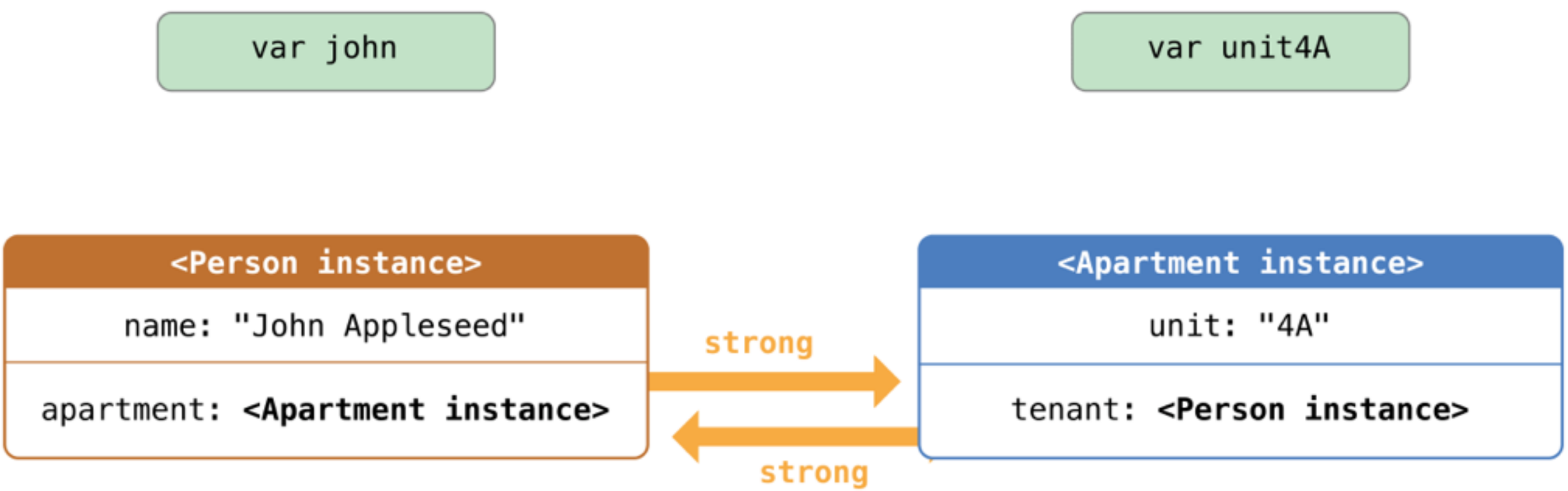


PREVENTING MEMORY CYCLE

```
john = nil
unit4A = nil
```

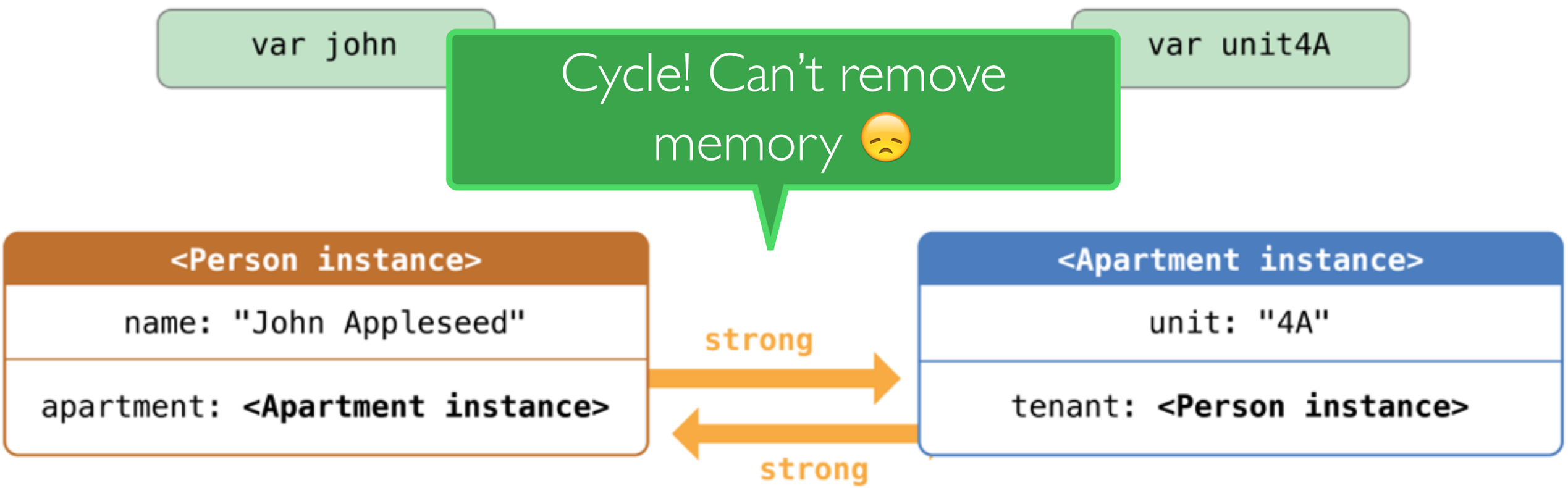
PREVENTING MEMORY CYCLE

```
john = nil
unit4A = nil
```

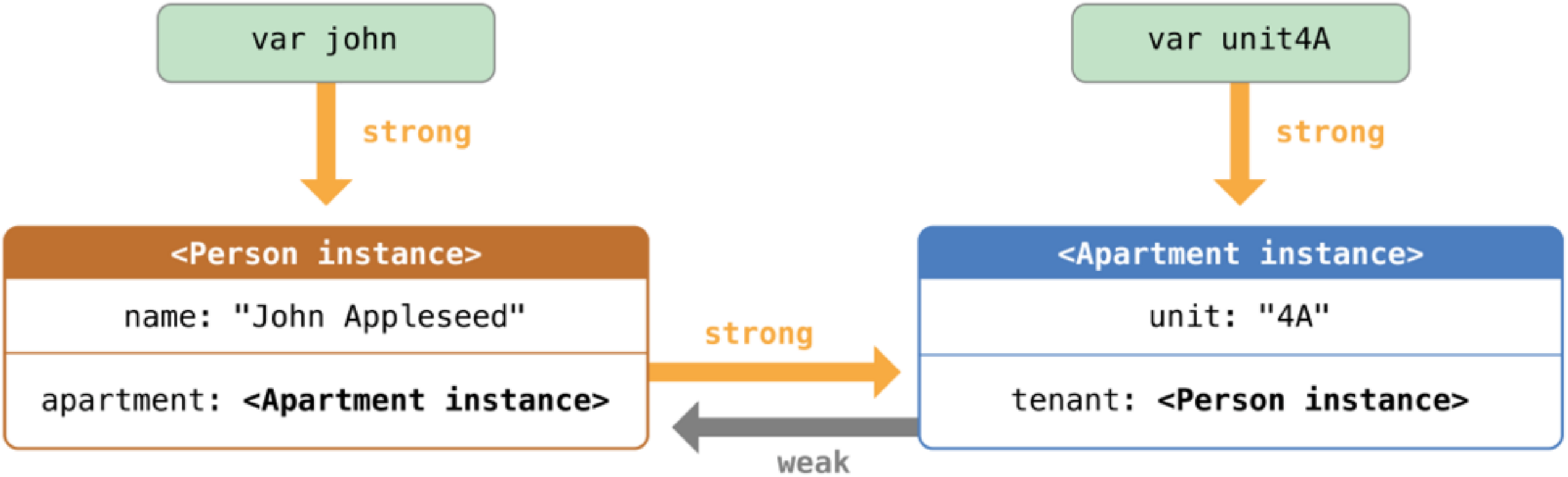


PREVENTING MEMORY CYCLE

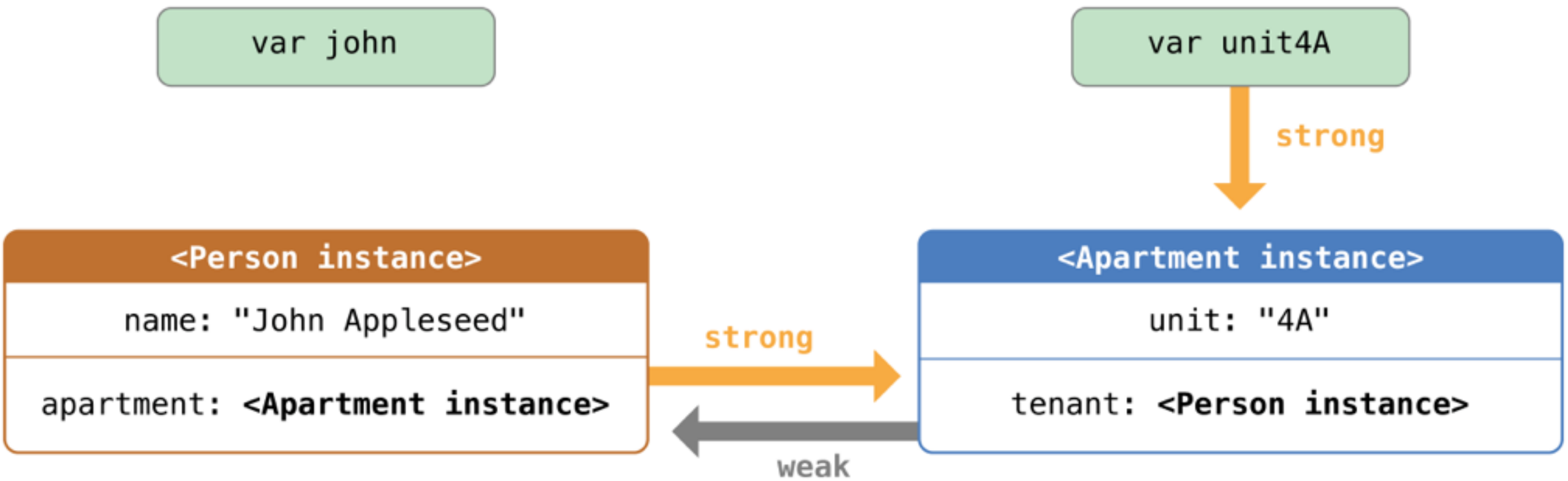
```
john = nil
unit4A = nil
```



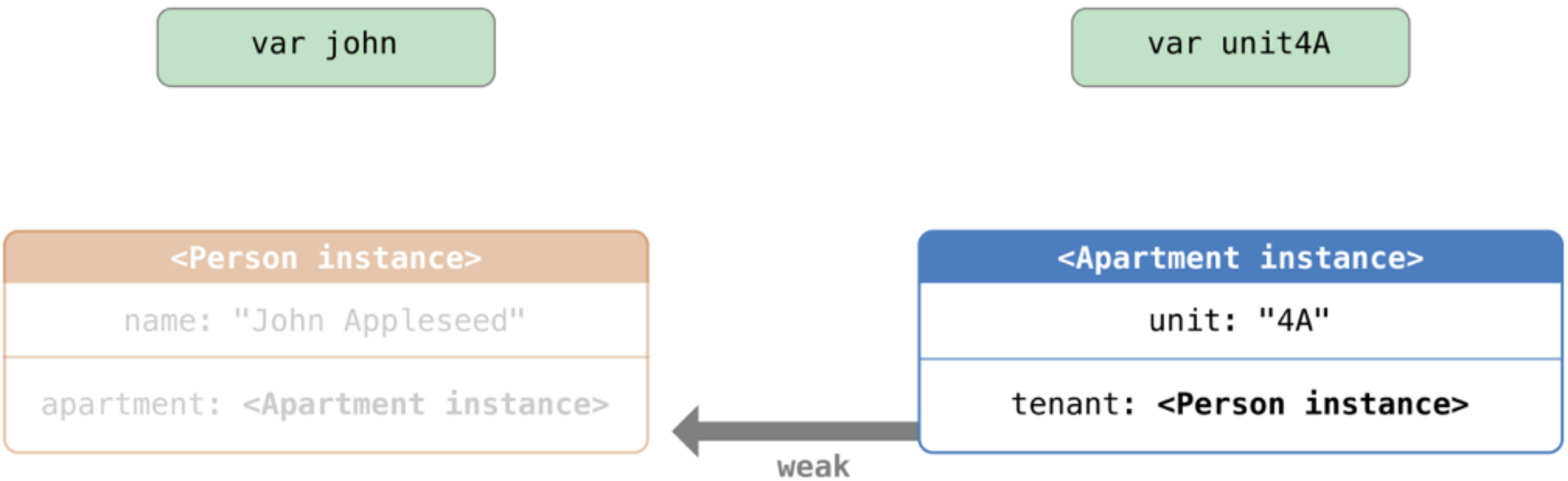
PREVENTING MEMORY CYCLE



PREVENTING MEMORY CYCLE



PREVENTING MEMORY CYCLE



PREVENTING MEMORY CYCLE

Pong | Build Pong: **Succeeded** | Today at 9:43 PM

Pong 〉 Pong 〉 ViewController.swift 〉 collisionBehavior(_:beganContactForItem:withItem:atPoint:)

```
func collisionBehavior(behavior: UICollisionBehavior, beganContactForItem item1:
    UIDynamicItem, withItem item2: UIDynamicItem, atPoint p: CGPoint) {

    // Collision between ball and paddle
    if(item1 === orangeBall && item2 === paddle) {
        let pushBehavior = UIPushBehavior(items: [orangeBall], mode: .
            Instantaneous)
        pushBehavior.pushDirection = CGVectorMake(0.0, -1.0)
        pushBehavior.magnitude = 0.75

        // Need to remove the behavior without causing circular reference
        pushBehavior.action = { [unowned pushBehavior] in
            if(!pushBehavior.active) {
                pushBehavior.dynamicAnimator?.removeBehavior(pushBehavior)
            }
        }
        animator.addBehavior(pushBehavior)
    }
}
```

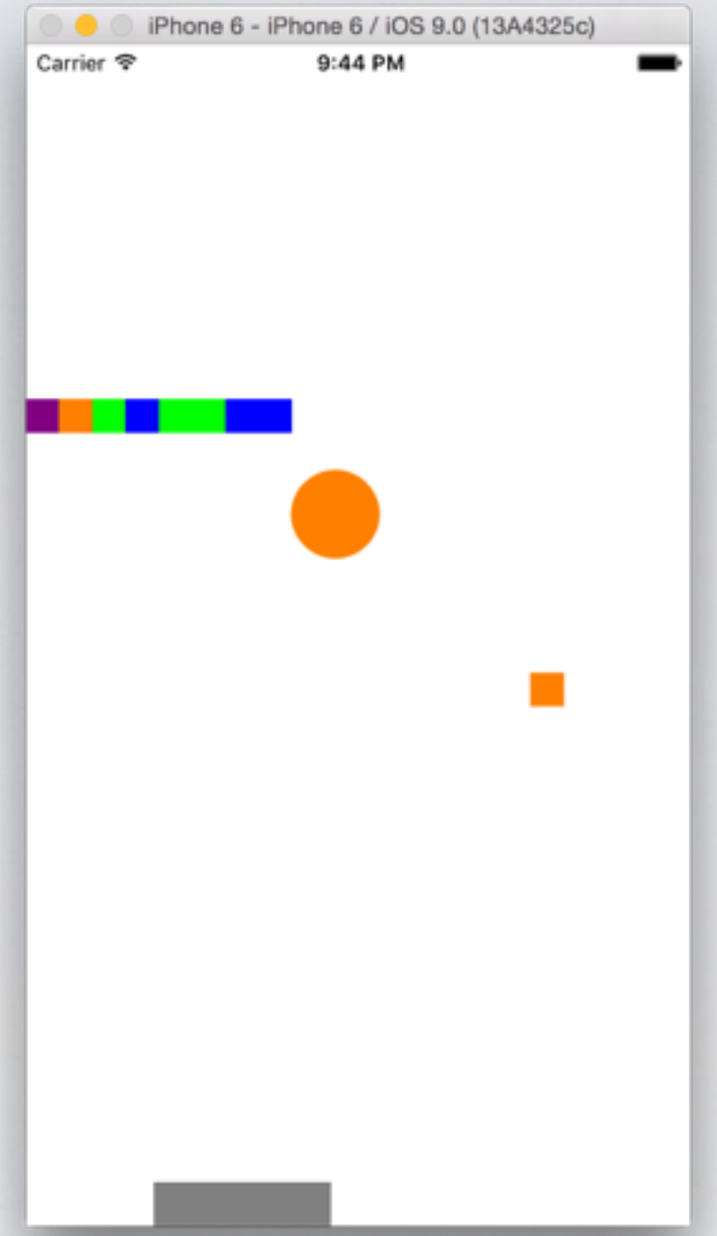
PREVENTING MEMORY CYCLE

Pong | Build Pong: **Succeeded** | Today at 9:43 PM

Pong 〉 Pong 〉 ViewController.swift 〉 collisionBehavior(_:beganContactForItem:withItem:atPoint:)

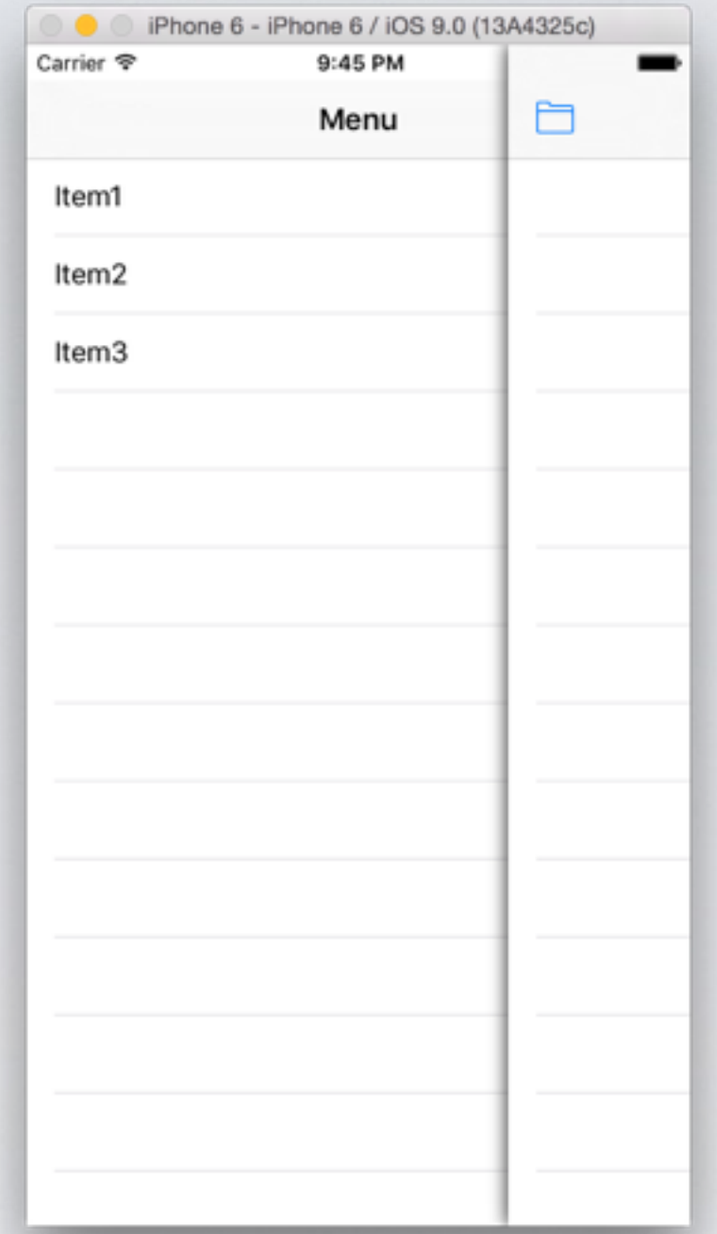
```
func collisionBehavior(behavior: UICollisionBehavior, beganContactForItem item1:
    UIDynamicItem, withItem item2: UIDynamicItem, atPoint p: CGPoint) {

    // Collision between ball and paddle
    if(item1 === orangeBall && item2 === paddle) {
        let pushBehavior = UIPushBehavior(items: [orangeBall], mode: .
            Instantaneous)
        pushBehavior.pushDirection = CGVectorMake(0.0, -1.0)
        pushBehavior.magnitude = 0.75

        // Need to remove the behavior without causing circular reference
        pushBehavior.action = { [unowned pushBehavior] in
            if(!pushBehavior.active) {
                pus                                    pushBehavior)
            }
        }
        animator.ad
    }
}
```

Unowned reference



DEMO

https://github.com/talentsparkio/Pong



DEMO

https://github.com/talentsparkio/SlidingMenu

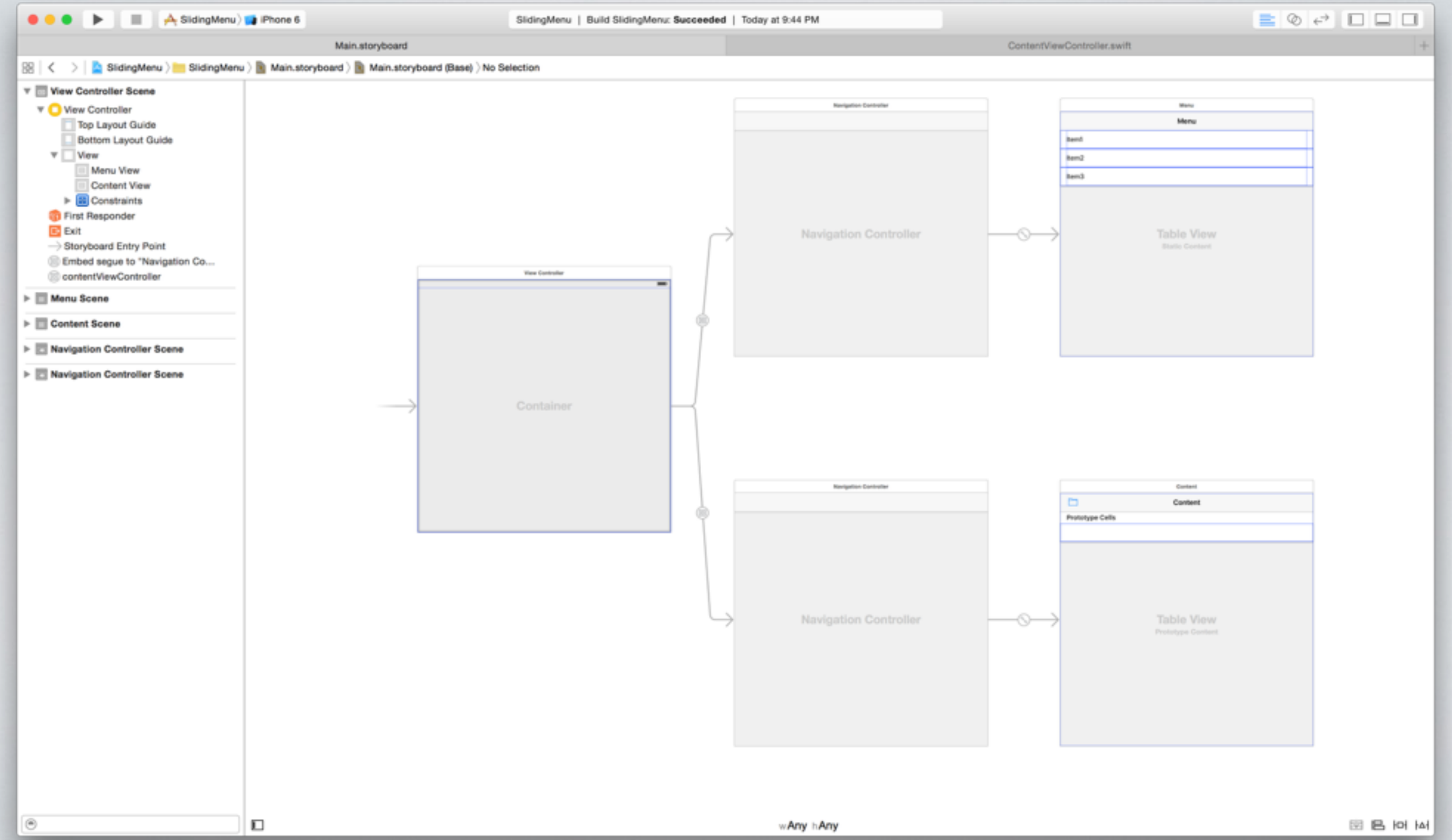
www.teehanlax.com/blog/

Introduction to UIKit Dynamics

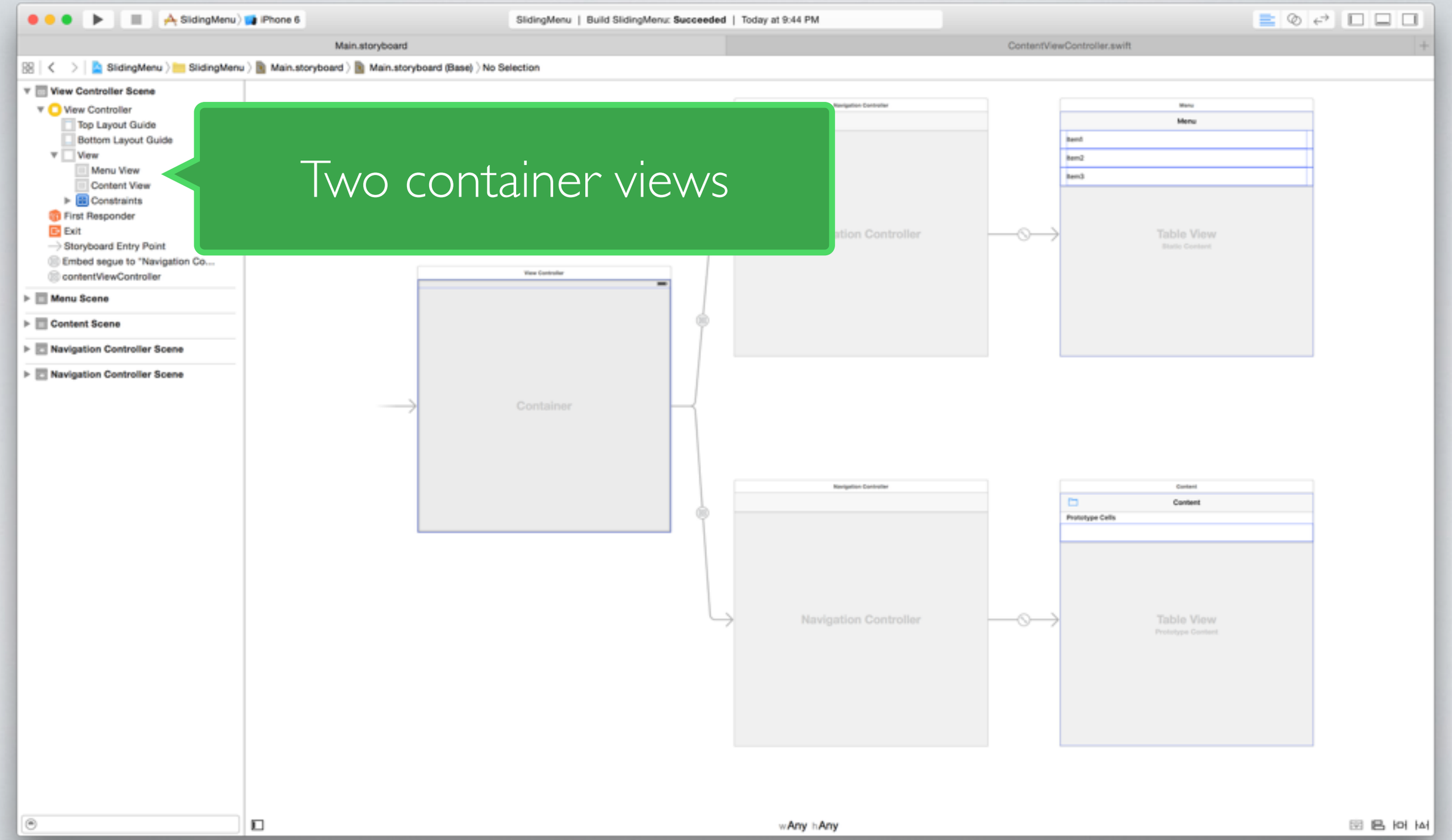
Written by Ash Furrow on September 20,2013 in Development

iOS 7 is a real conundrum. It juxtaposes its smooth, platonic interface elements with the physical realism of making those elements respond realistically to user interaction. We already covered UIMotionEffects, which adjust the appearance of an interface to the way the user is holding a device. Today, we're going to cover realistic animations using UIKit Dynamics.

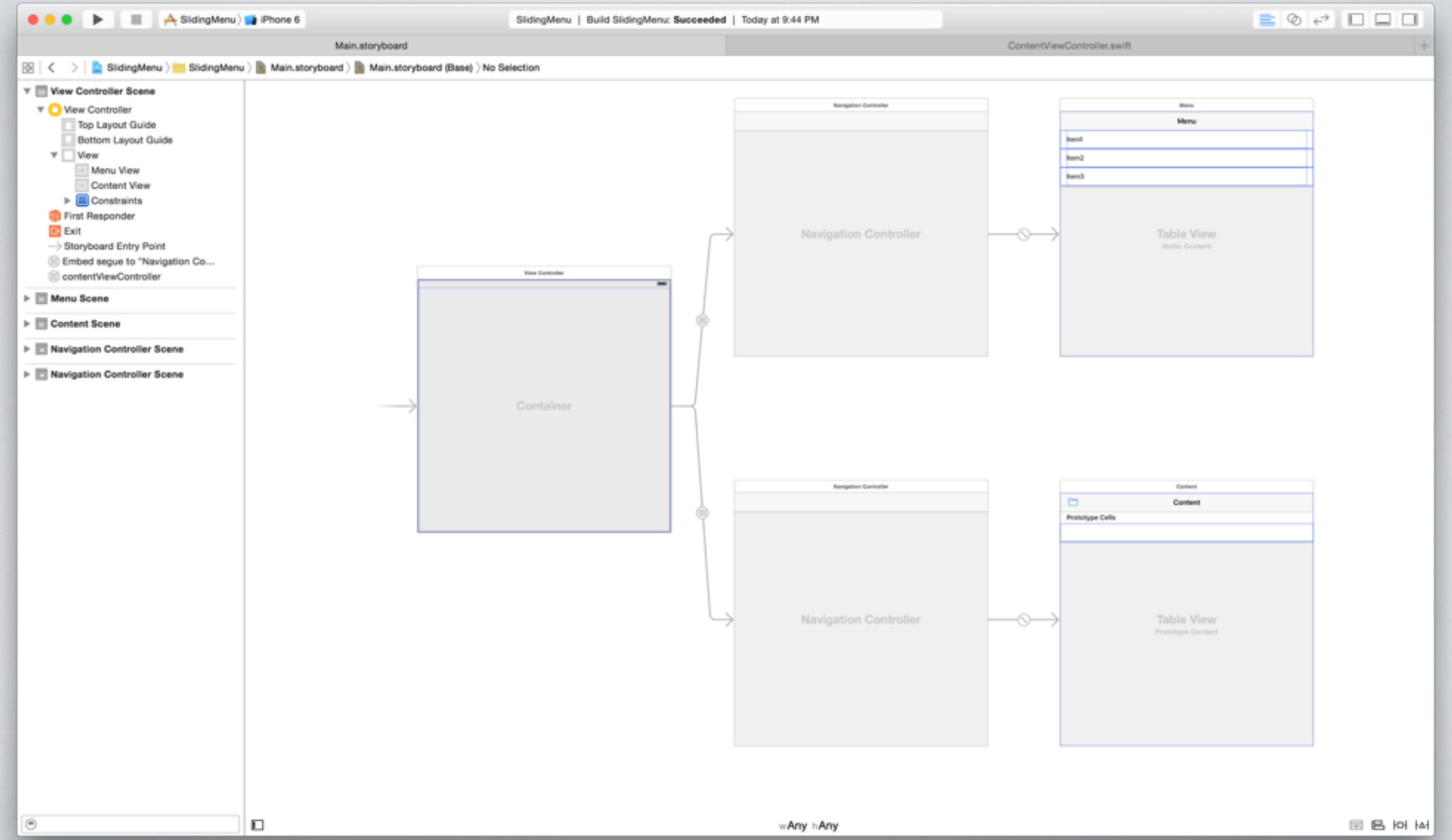
KEY IDEAS


SlidingMenu
iPhone 6
SlidingMenu | Build SlidingMenu: Succeeded | Today at 9:44 PM
Main.storyboard
ContentViewController.swift
SlidingMenu
SlidingMenu
Main.storyboard
Main.storyboard (Base)
No Selection
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Menu View
Content View
Constraints
First Responder
Exit
Storyboard Entry Point
Embed segue to "Navigation Co...
contentViewController
Menu Scene
Content Scene
Navigation Controller Scene
Navigation Controller Scene
Container
Navigation Controller
Navigation Controller
Menu
Table View
Navigation Controller
Navigation Controller
Content
Content
Prototype Cells
Table View
wAny hAny

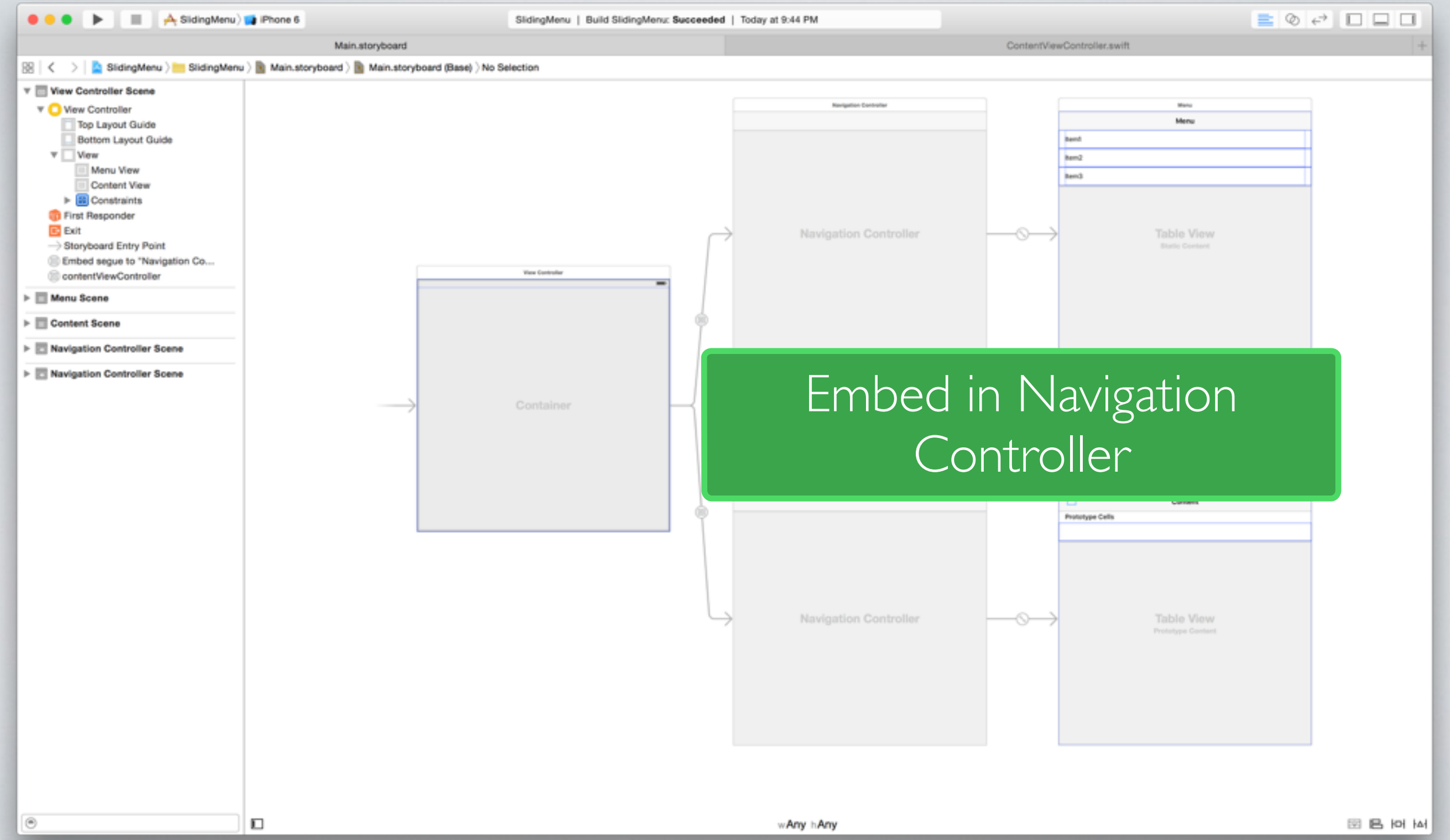
KEY IDEAS


Two container views
SlidingMenu
iPhone 6
SlidingMenu | Build SlidingMenu: Succeeded | Today at 9:44 PM
Main.storyboard
ContentViewController.swift
SlidingMenu
SlidingMenu
Main.storyboard
Main.storyboard (Base)
No Selection
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Menu View
Content View
Constraints
First Responder
Exit
Storyboard Entry Point
Embed segue to "Navigation Co...
contentViewController
Menu Scene
Content Scene
Navigation Controller Scene
Navigation Controller Scene
View Controller
Container
Navigation Controller
Menu
Table View
Navigation Controller
Content
Content
Prototype Cells
Table View
Prototype Content
wAny hAny

KEY IDEAS


SlidingMenu
iPhone 6
SlidingMenu | Build SlidingMenu: Succeeded | Today at 9:44 PM
Main.storyboard
ContentViewController.swift
SlidingMenu
SlidingMenu
Main.storyboard
Main.storyboard (Base)
No Selection
View Controller Scene
View Controller
Top Layout Guide
Bottom Layout Guide
View
Menu View
Content View
Constraints
First Responder
Exit
Storyboard Entry Point
Embed segue to "Navigation Co...
contentViewController
Menu Scene
Content Scene
Navigation Controller Scene
Navigation Controller Scene
Container
Navigation Controller
Navigation Controller
Menu
Table View
Navigation Controller
Navigation Controller
Content
Content
Prototype Cells
Table View
wAny hAny

KEY IDEAS



Embed in Navigation Controller

WHAT'S NEXT?

- Go through the WWDC videos
- Check out http://pttrns.com
- Check out https://www.cocoacontrols.com