THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 2D

VIEW PROPERTY ANIMATOR

VIEW PROPERTY ANIMATOR

- Multiple ways to animate views
  - Begin/commit
  - Block based API (iOS4+)
  - Property animators (iOS10+)

VIEW PROPERTY ANIMATOR

UIVIEW ANIMATIONS

- Built-in animatable properties on UIView
- Different variations for customization, callbacks, timing, etc.

```
// Add a rotation transform to our red box view
UIView.animate(withDuration: 3.0, animations: { () -> Void in
  let rotationTransform = CGAffineTransform(rotationAngle: 3.14)
  redView.transform = rotationTransform
})

// Fade out the green
UIView.animate(withDuration: 2) {
  greenView.alpha = 0
}
```

VIEW PROPERTY ANIMATOR

UIVIEW ANIMATIONS

- Pros
  - Easy to use
  - Customizable to an extent
- Cons
  - Limited to predefined timing curves (fadeinout, etc.)
  - No way to cancel/interupt animations
    - Dirty solution: Create an animation to replace the current one

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

- New API that is highly extensible (iOS10+)
- Can behave the same as the old UIViewAnimations
- Provides fine grained control over the animations
  - Pause, restart, modify, cancel animations

UIViewPropertyAnimator

A class that animates changes to views and allows the dynamic modification of those animations.

Overview

A `UIViewPropertyAnimator` object lets you animate changes to views and dynamically modify your animations before they finish. With a property animator, you can run your animations from start to finish normally or you can turn them into interactive animations and control the timing yourself. The animator operates on animatable properties of views, such as the `frame`, `center`, `alpha`, and `transform` properties, creating the needed animations from the blocks you provide.

When creating a property animator object, you specify the following:

- A block containing code that modifies the properties of one or more views.
- The timing curve that defines the speed of the animation over the course of its run

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

API can be similar to UIViewAnimation

```
/// Move and Fade (similiar to UIViewAnimation)
UIViewPropertyAnimator(duration: 1, curve: .easeInOut) {
  heart.alpha = 0
  heart.center = heart.superview!.center
}.startAnimation()
```

Animations start in inactive state

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

- Property animator gives you programmatic control over the timing and execution of the animations
  - Start, pause, resume, and stop animations
  - Add animation blocks after the original animations start
  - Scrub through a paused animation by modifying the `fractionComplete` property
  - Change the animation’s direction using the `isReversed` property.
  - Modify the timing and duration of a partially complete animation

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

SUBTITLE

- Animations are additive

- Add animation blocks after the original animations start using the addAnimations(_:) and addAnimations(_:delayFactor:) methods.

```
/// Alternative call
let animator = UIViewPropertyAnimator(duration: 3,
                                      curve: .easeInOut)
// Add animation block
animator.addAnimations {
  heart.alpha = 0
}

// Now here goes our second
animator.addAnimations {
  heart.center = heart.superview!.center
}
animator.startAnimation()
```

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

- Add a completion block
- Monitor the state

```
/// Alternative call
let animator = UIViewPropertyAnimator(duration: 3,
                                      curve: .easeInOut)
// Add animation block
animator.addAnimations {
  heart.alpha = 0
}

// Now here goes our second
animator.addAnimations {
  heart.center = heart.superview!.center
}
animator.startAnimation()

/// Add completion block
animator.addCompletion { (position) in
  switch position {
  case .end: print("Completion handler called at end of animation")
  case .current: print("Completion handler called mid-way through animation")
  case .start: print("Completion handler called  at start of animation")
  }
}
```

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

- Additional properties to monitor or change the animation

fractionComplete—This property sets or returns the animation's completion percentage.

isReversed—This property sets or returns a Boolean value that determines if the animation is running in reverse.

state—This property returns a value that determines the current state of the animation. It is an enumeration of type **UIViewAnimatingState** with the values **inactive**, **active**, and **stopped**.

isRunning—This property returns a Boolean value that determines whether the animation is running or not.

startAnimation()—This method starts the animation.

pauseAnimation()—This method pauses the animation.

stopAnimation(Bool**)**—This method stops the animation. The attribute indicates whether the animator will be deactivated or it will stay active for further actions.

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

```
/// Reverse the animation
let button = UIButton(frame: CGRect(origin: .zero, size: CGSize(width: 100,
    height: 30)))
button.setTitle("Reverse", for: .normal)
button.setTitleColor(.black, for: .normal)
button.setTitleColor(.gray, for: .highlighted)
let listener = EventListener()
listener.eventFired = {
  animator.isReversed = true
}
```

Reverse the animation.
Callback at start.

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

```
/// Start/stop the animation
let button = UIButton(frame: CGRect(origin: .zero, size: CGSize(width: 100, height: 30)))
button.setTitle("Pause", for: .normal)
button.setTitleColor(.black, for: .normal)
button.setTitleColor(.gray, for: .highlighted)

let listener = EventListener()
listener.eventFired = {
  if animator.isRunning {
    animator.pauseAnimation()
  } else {
    animator.startAnimation()
  }
}
```

Check state

Pause/restart

VIEW PROPERTY ANIMATOR

UIVIEWPROPERTYANIMATOR

- Additional properties to monitor or change the animation

fractionComplete—This property sets or returns the animation's completion percentage.

isReversed—This property sets or returns a Boolean value that determines if the animation is running in reverse.

state—This property returns a value that determines the current state of the animation. It is an enumeration of type **UIViewAnimatingState** with the values **inactive**, **active**, and **stopped**.

isRunning—This property returns a Boolean value that determines whether the animation is running or not.

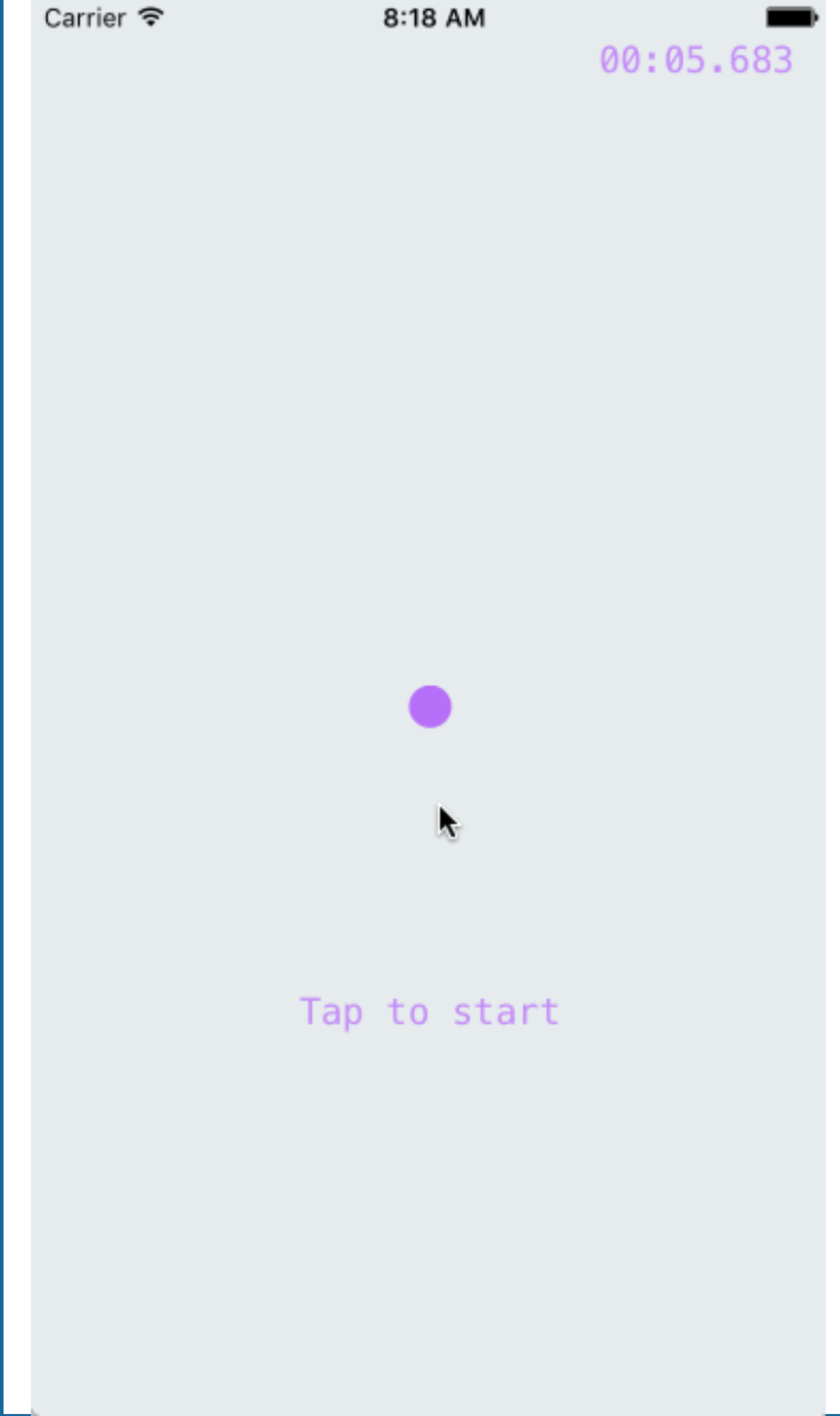
startAnimation()—This method starts the animation.

pauseAnimation()—This method pauses the animation.

stopAnimation(Bool**)**—This method stops the animation. The attribute indicates whether the animator will be deactivated or it will stay active for further actions.

VIEW PROPERTY ANIMATOR

- Nothing wrong with UIView animations
- In our book
  - Might pause/cancel animations when the page is being turned
  - Mini-game



THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 2D