Advances in UIKit Animations and Transitions

Session 216

Bruce Nilo UIKit
Michael Turner UIKit

© 2016 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

Responsive

Fluid

Smooth

Natural

Agenda

Agenda

Review

Agenda

Review

Discussion of UIViewPropertyAnimator

Agenda

Review

Discussion of UIViewPropertyAnimator

UIViewControllerAnimated Transitioning

Agenda

Review

Discussion of UIViewPropertyAnimator

UIViewControllerAnimated Transitioning

Demo of a New Photos Sample Application

Agenda

Review

Discussion of UIViewPropertyAnimator

UIViewControllerAnimated Transitioning

Demo of a New Photos Sample Application

Animation to Gesture Revisited

Agenda

Review

Discussion of UIViewPropertyAnimator

UIViewControllerAnimated Transitioning

Demo of a New Photos Sample Application

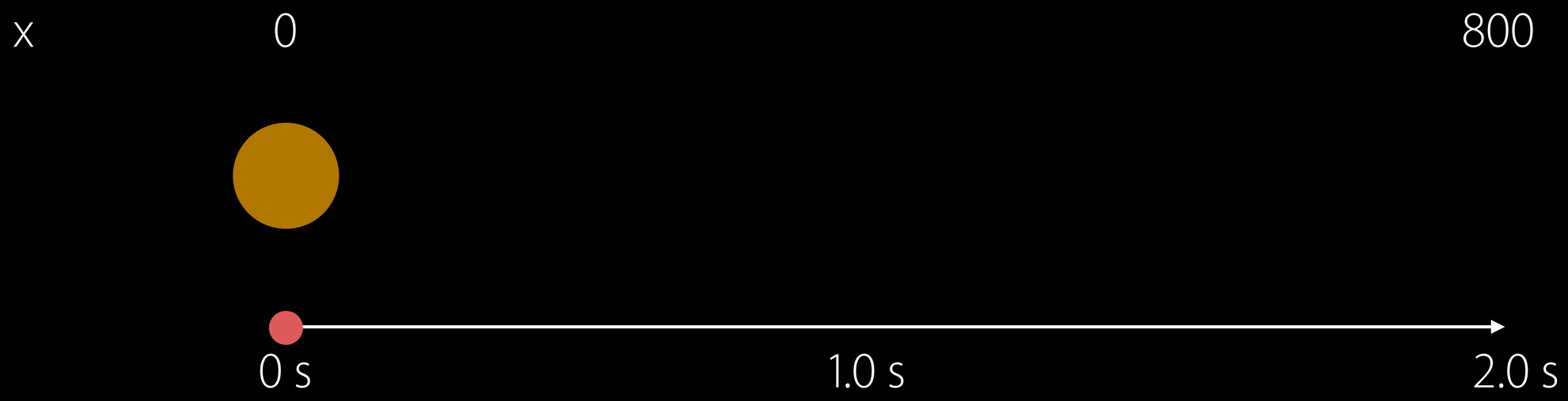
Animation to Gesture Revisited

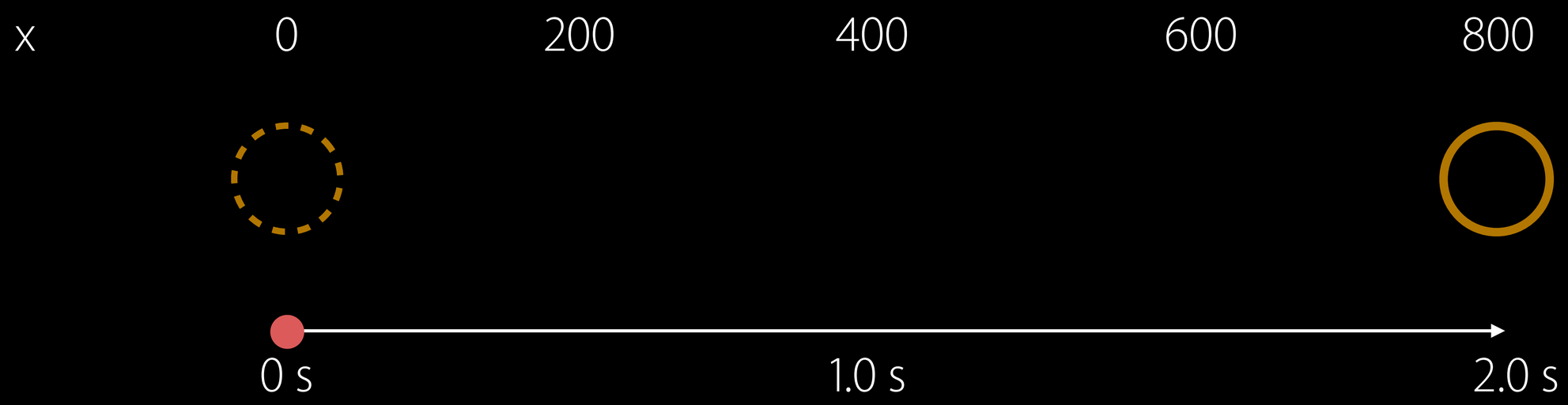
Interruptible Keyframe Animations

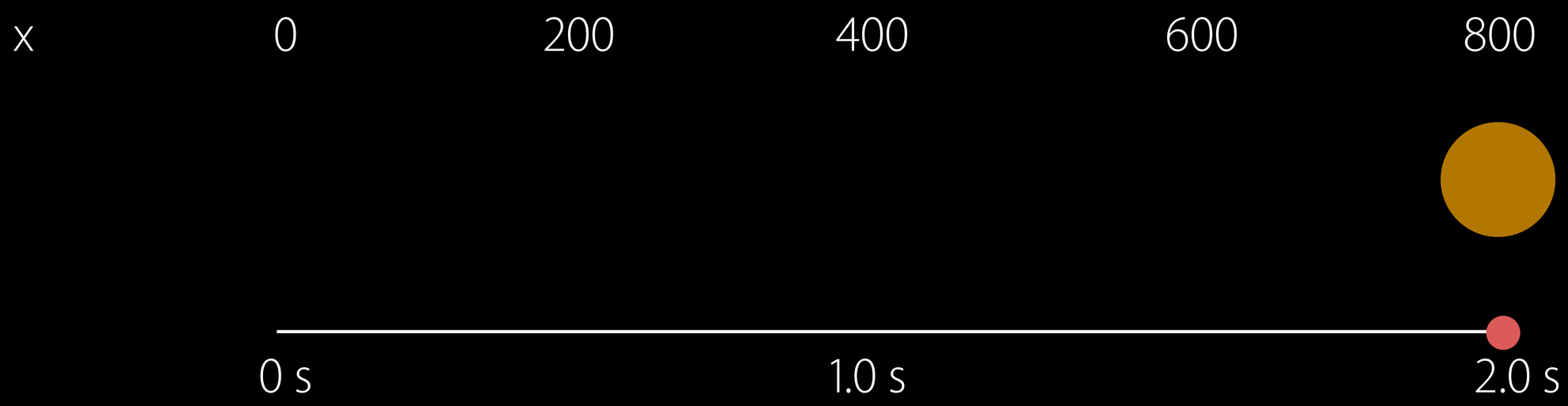
UIKit Animations

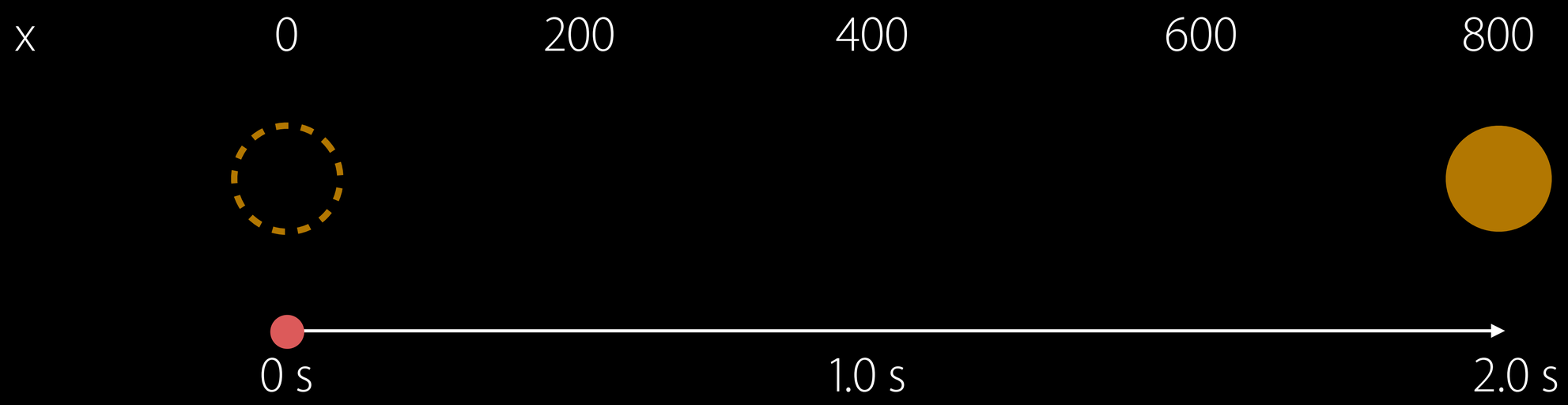
Implicit property animations

Interpolation and Pacing


x
0
800
0 s
1.0 s
2.0 s


x
0
200
400
600
800
0 s
1.0 s
2.0 s


x
0
200
400
600
800
0 s
1.0 s
2.0 s




```
UIView.animate(withDuration:2.0,
                    delay: 0.0,
                  options:[.linear]) {
   circle.center.x = 800.0
}
```

100%

50%

Normalized Value (μ)

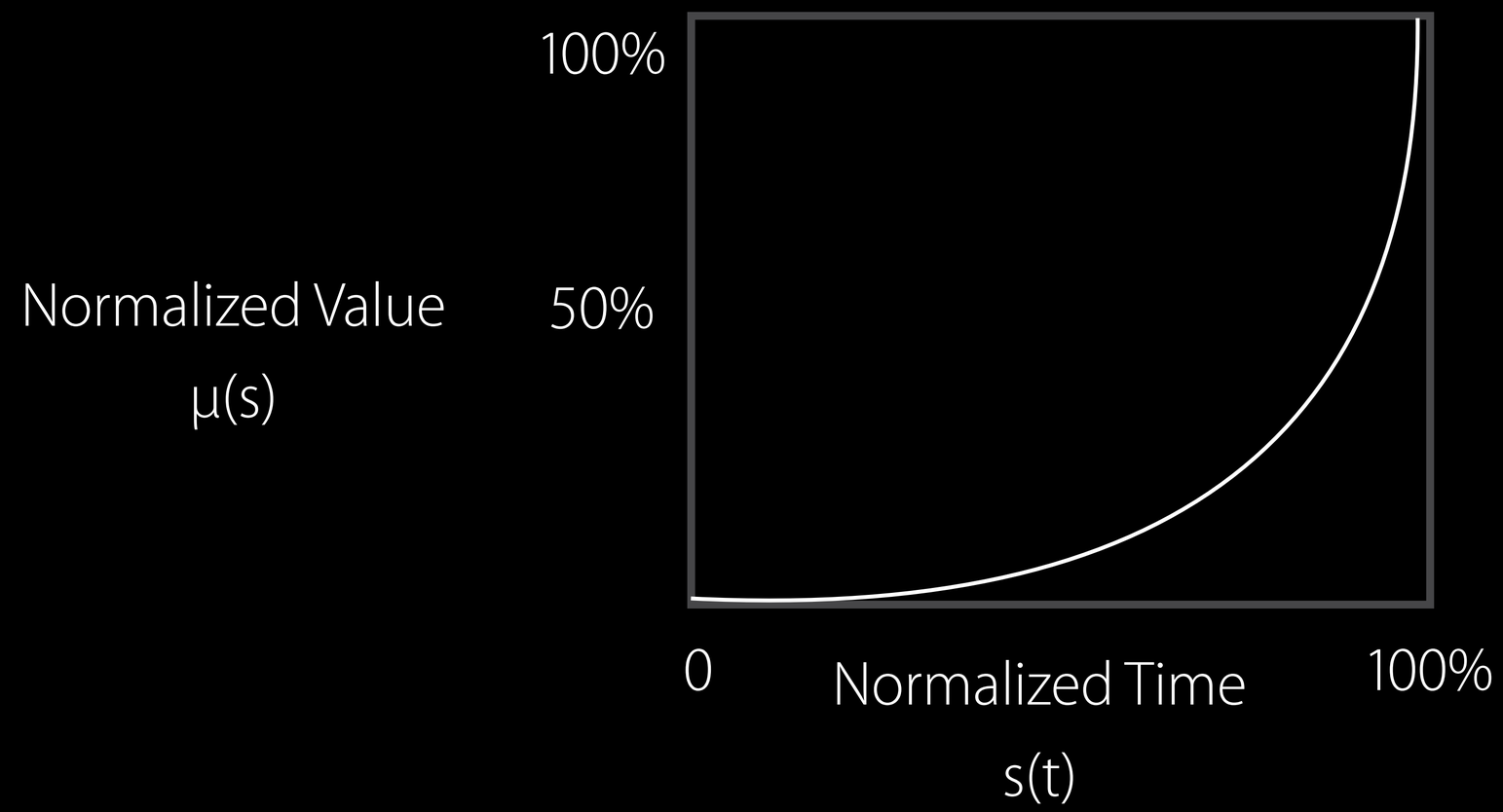
$$\frac{|x - x_0|}{|x_T - x_0|}$$

0

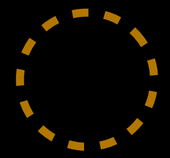
Normalized Time
s(t)

100%

x 0 200 400 600 800



x 0 200 400 600 800

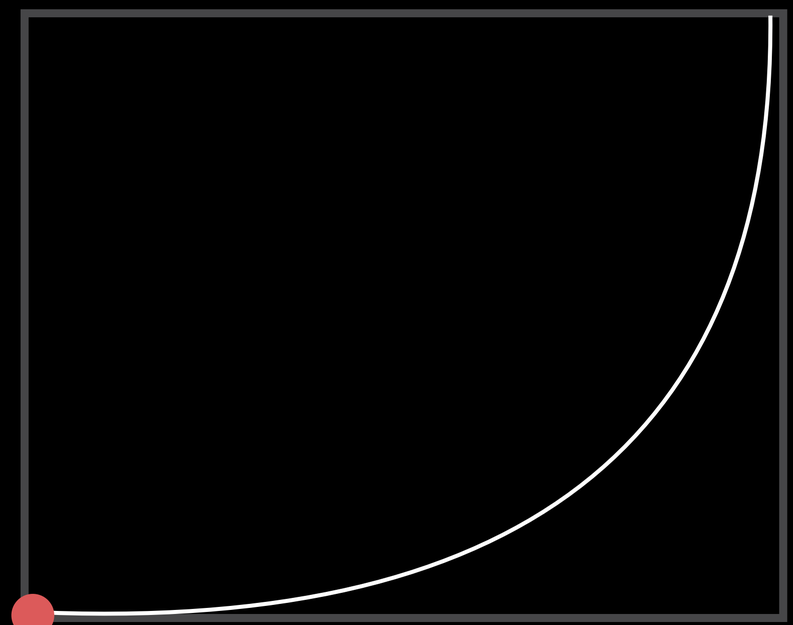


100%

Normalized Value

μ(s)

50%



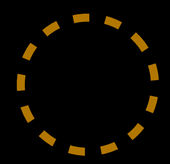
0

Normalized Time

s(t)

100%

x 0 200 400 600 800

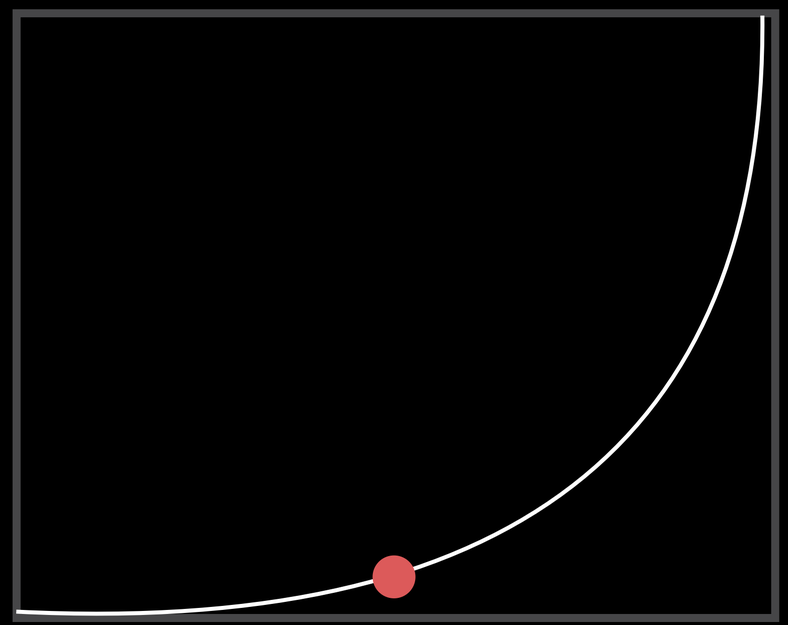


100%

Normalized Value

μ(s)

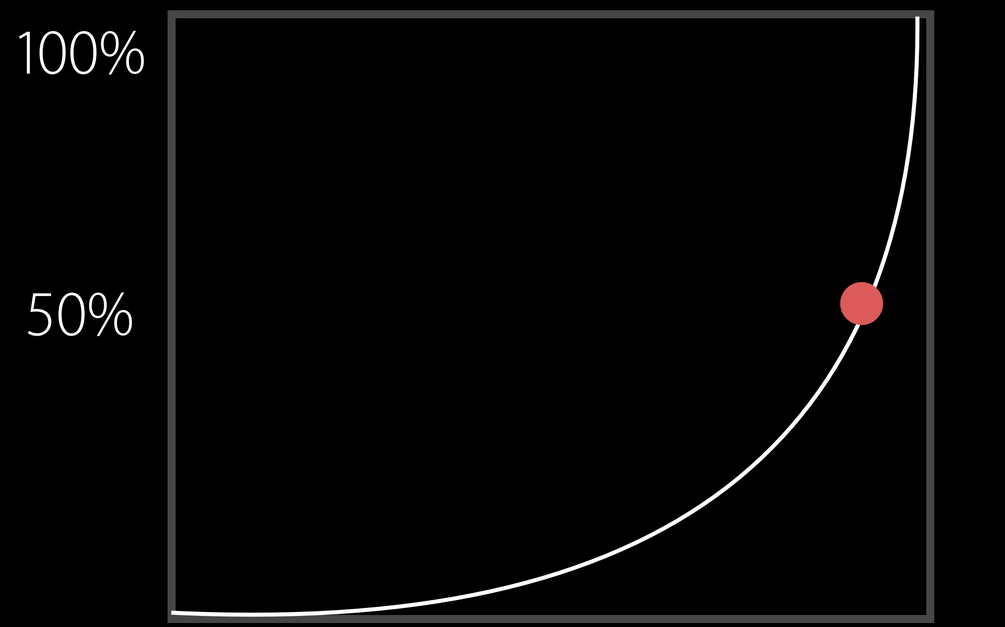
50%



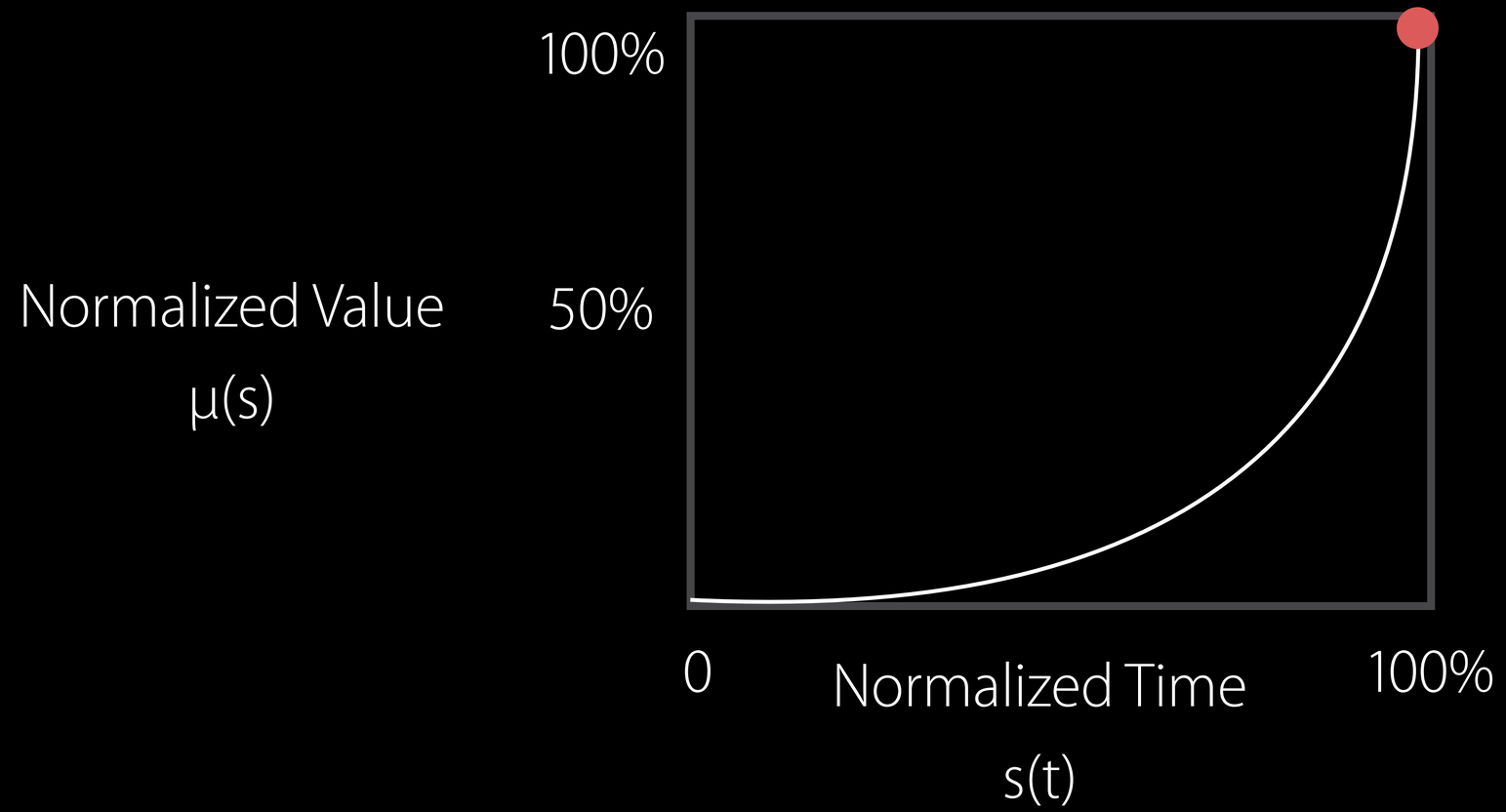
0 Normalized Time 100%

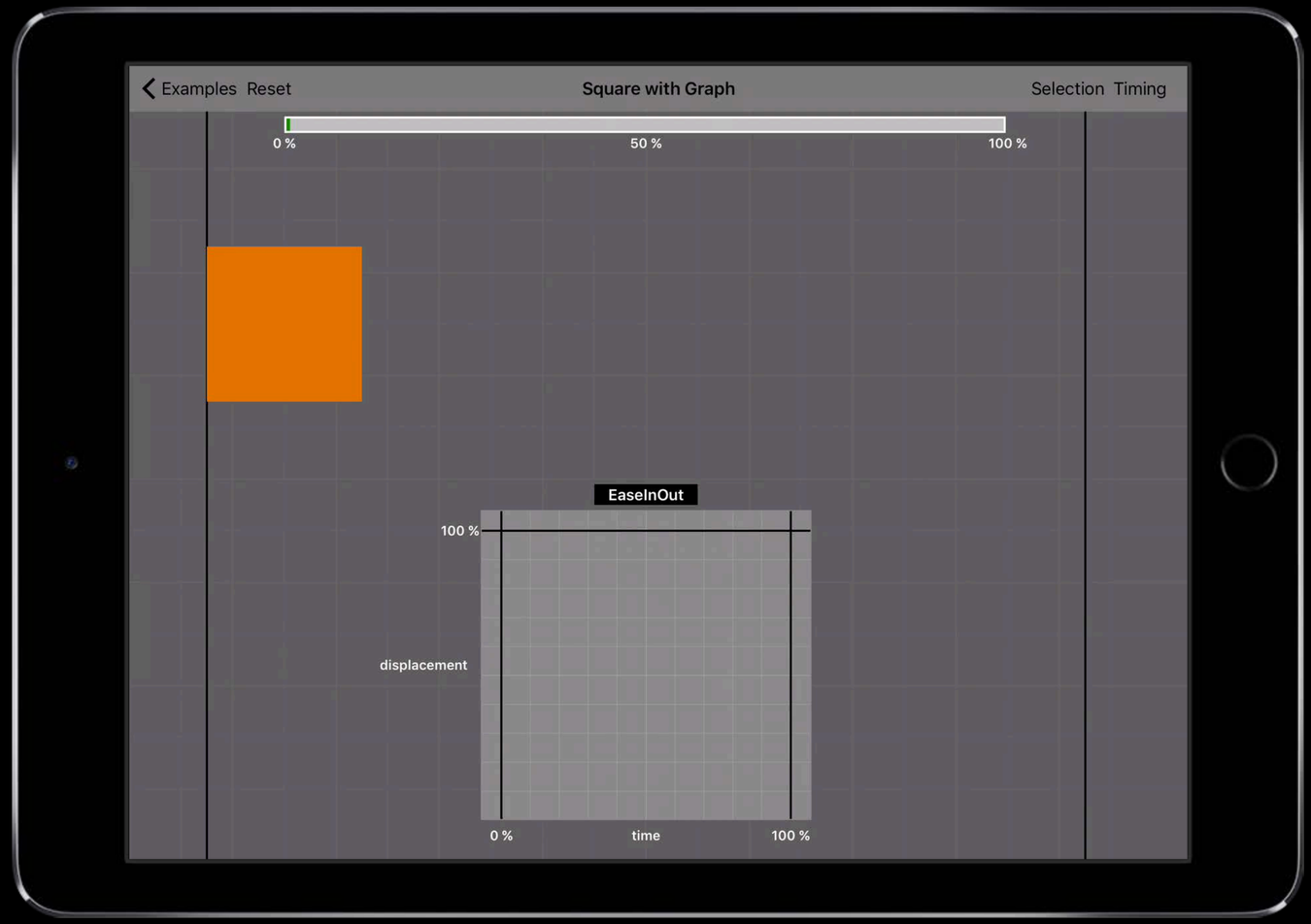
s(t)

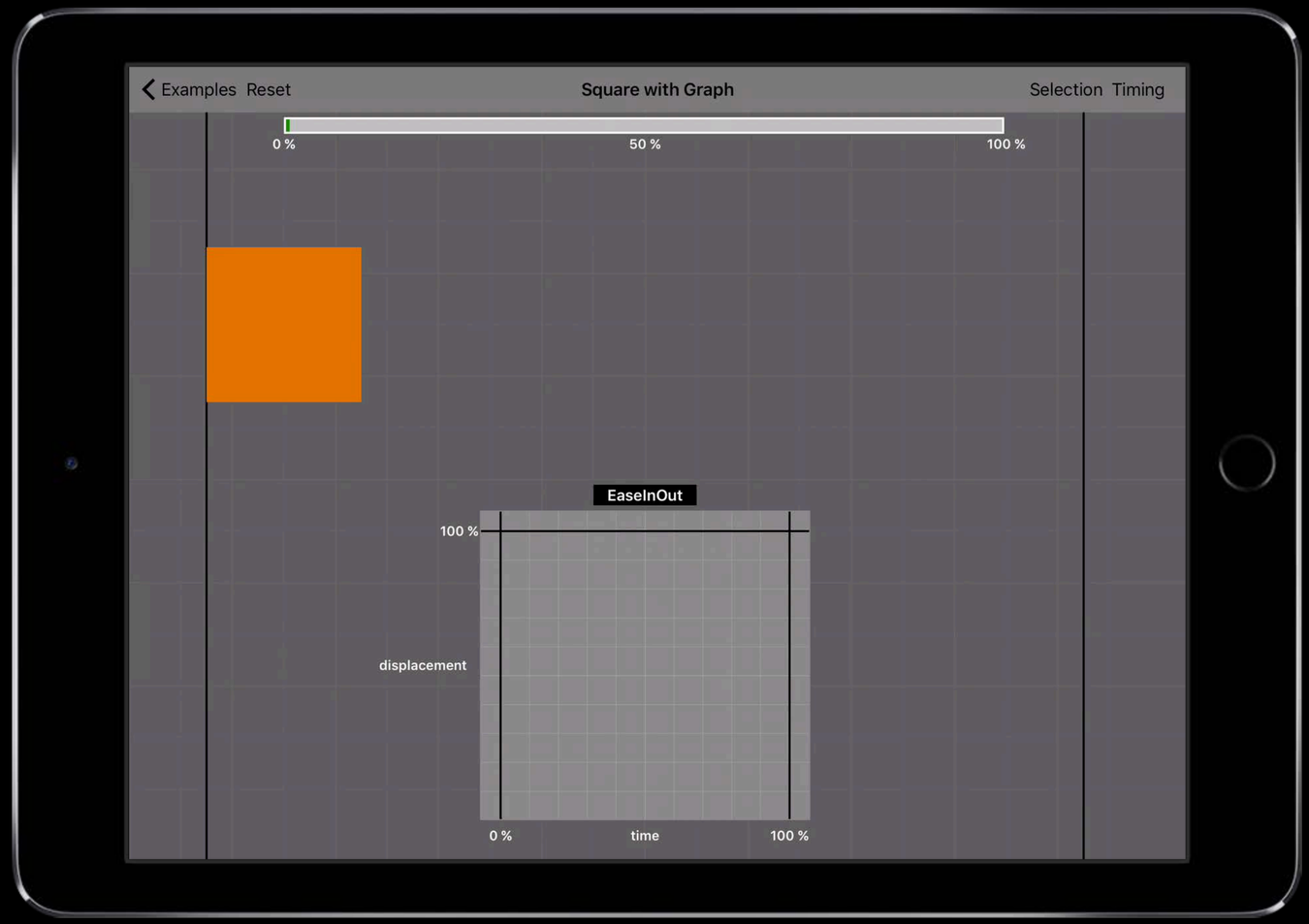
x 0 200 400 600 800



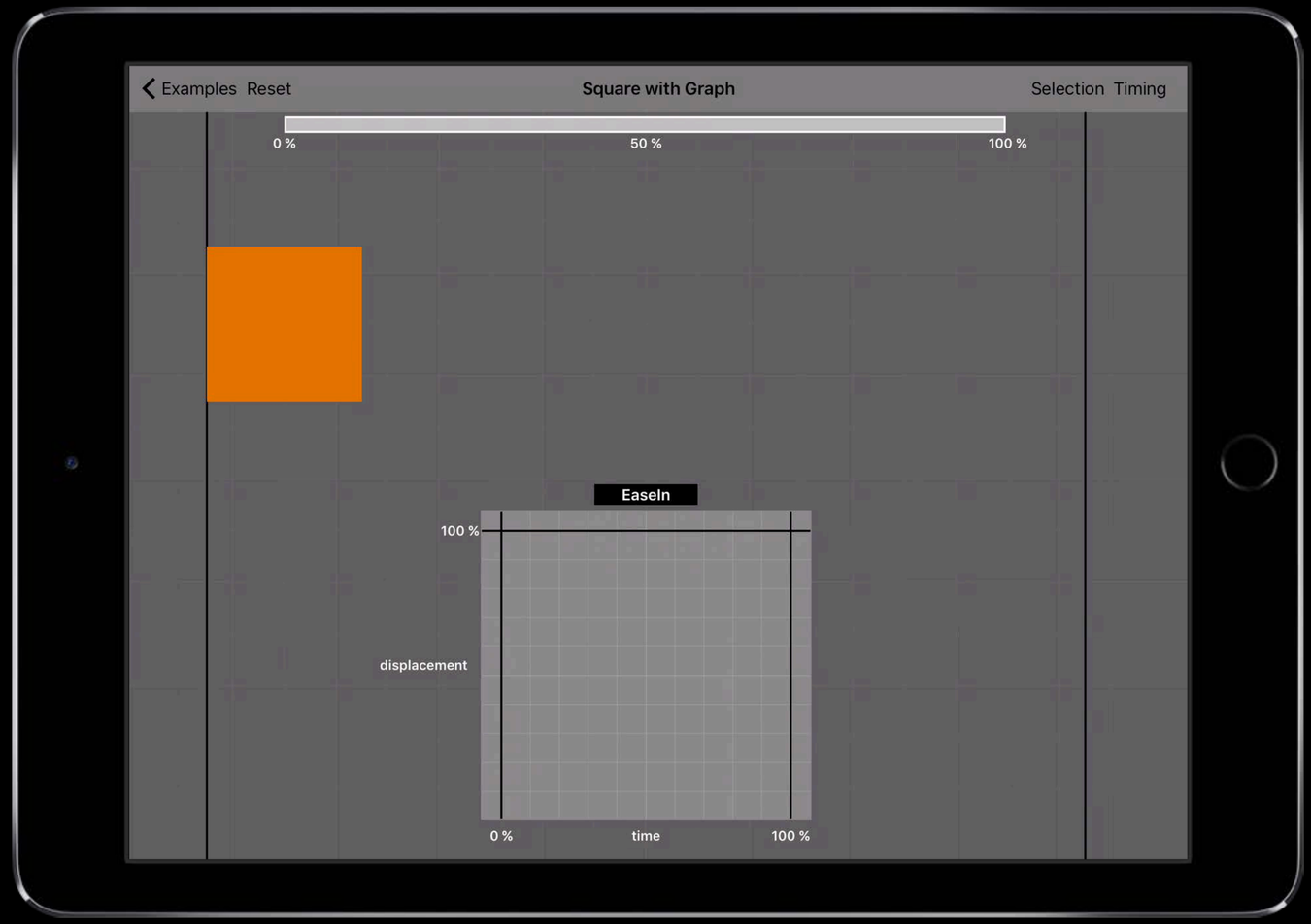
x 0 200 400 600 800



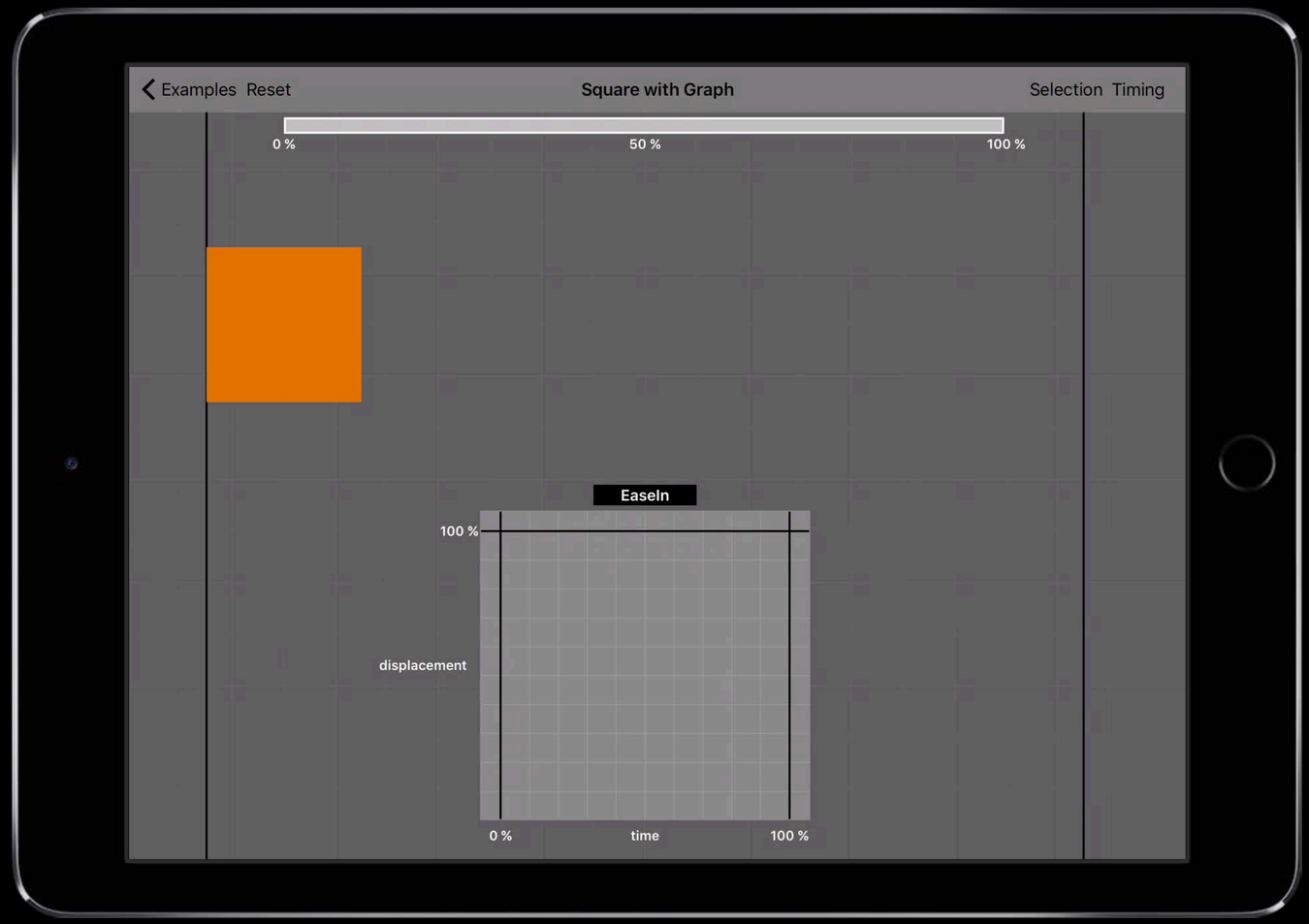

Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %


Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %

.easeInOut

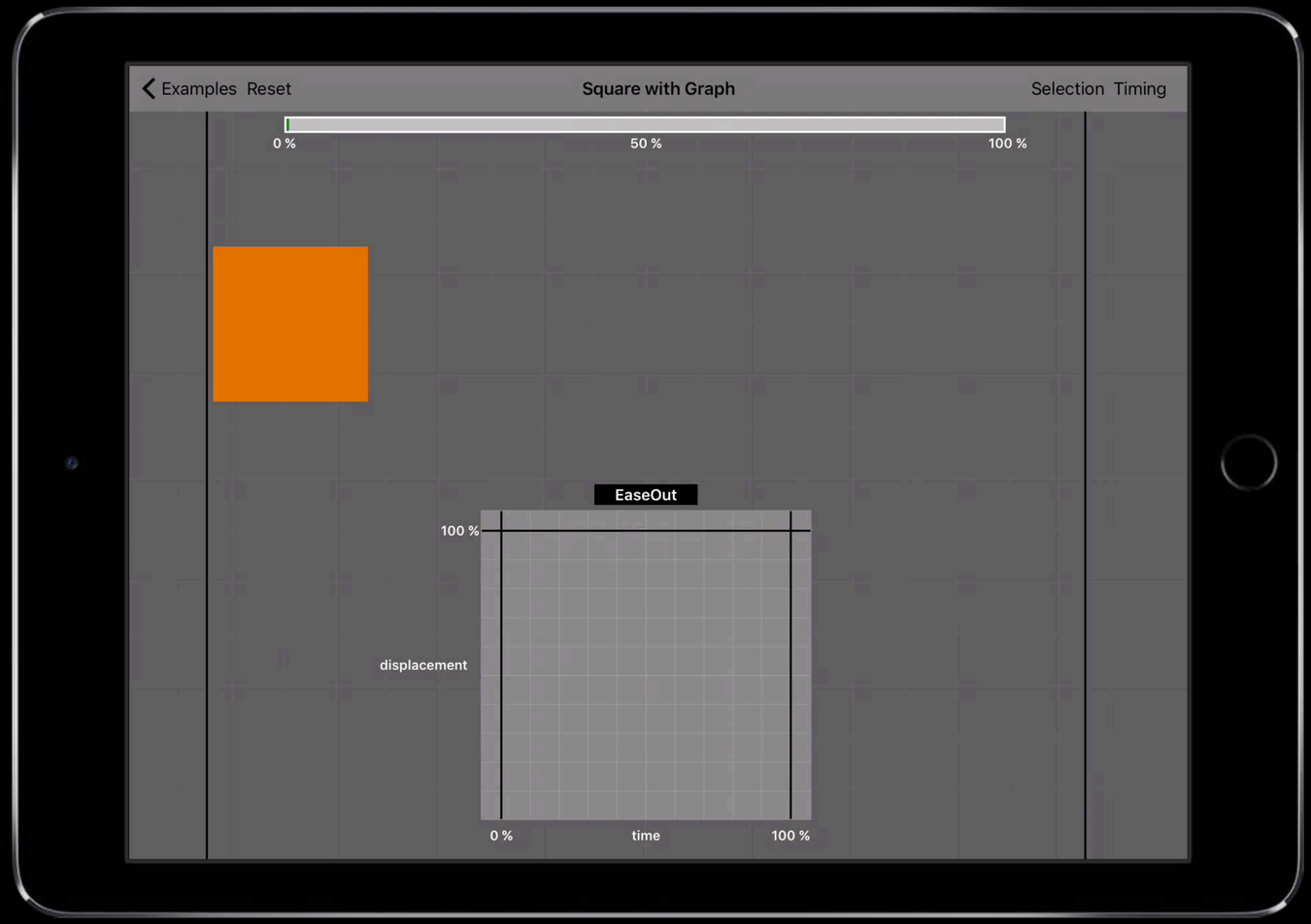

Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseIn
100 %
displacement
0 %
time
100 %

```
.easeInOut
```



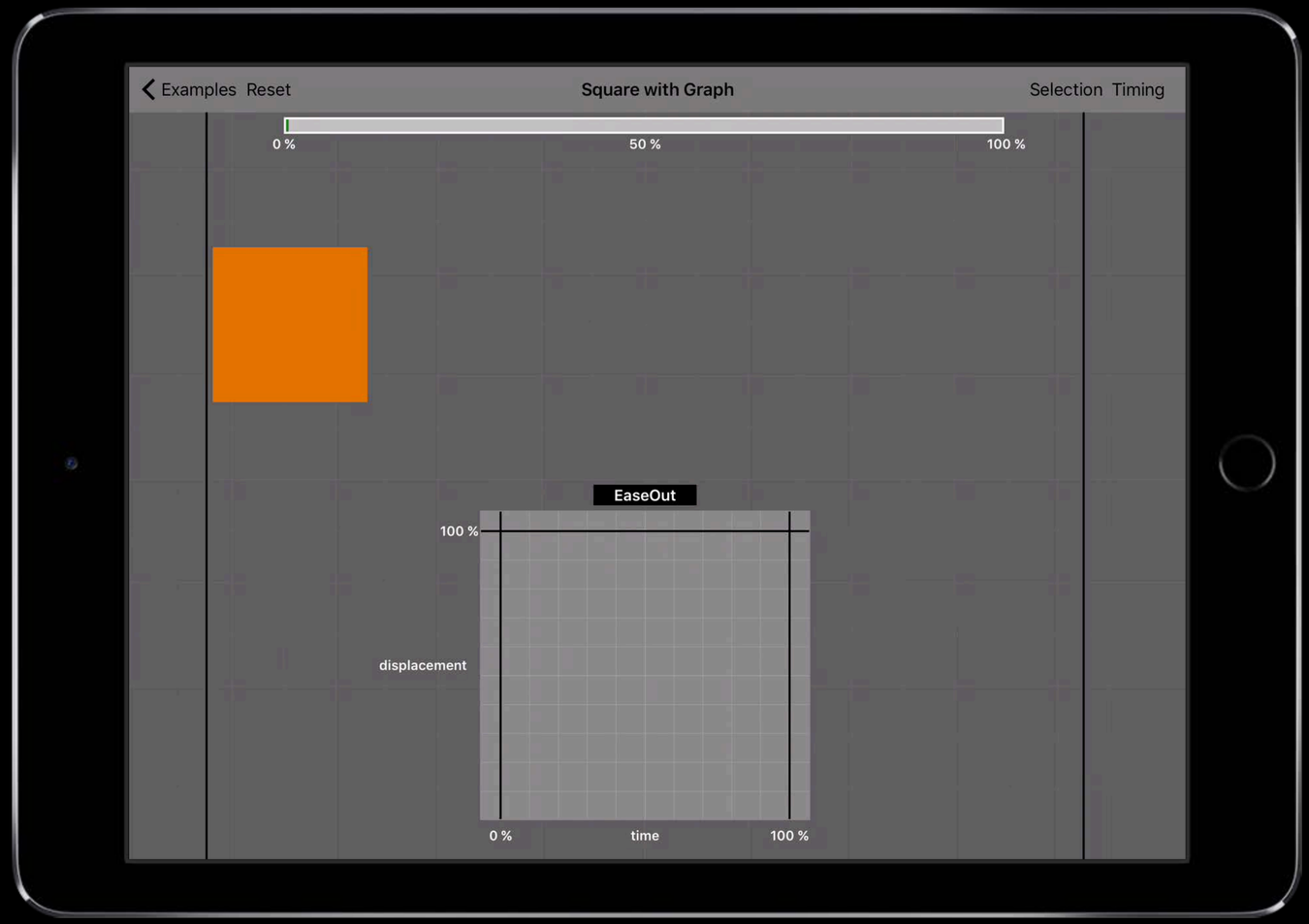
.easeInOut

.easeIn



.easeInOut

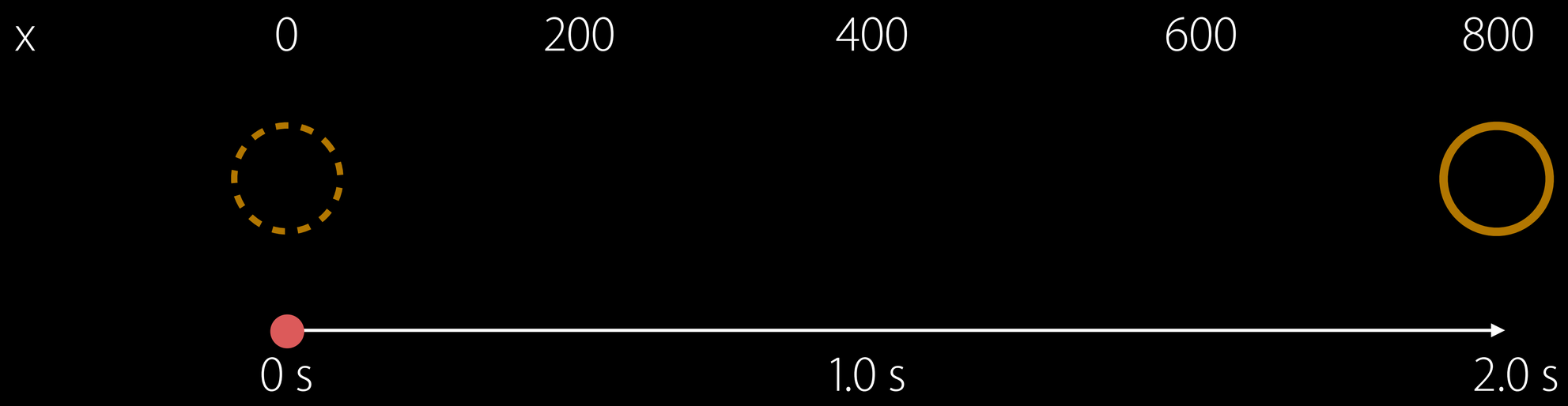
.easeIn



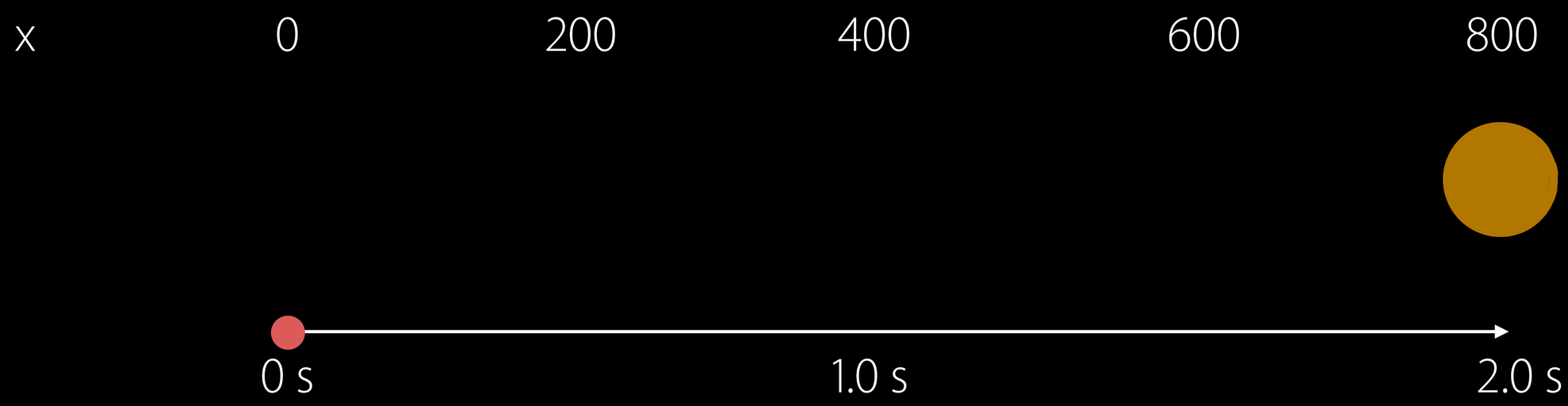
.easeInOut

.easeIn

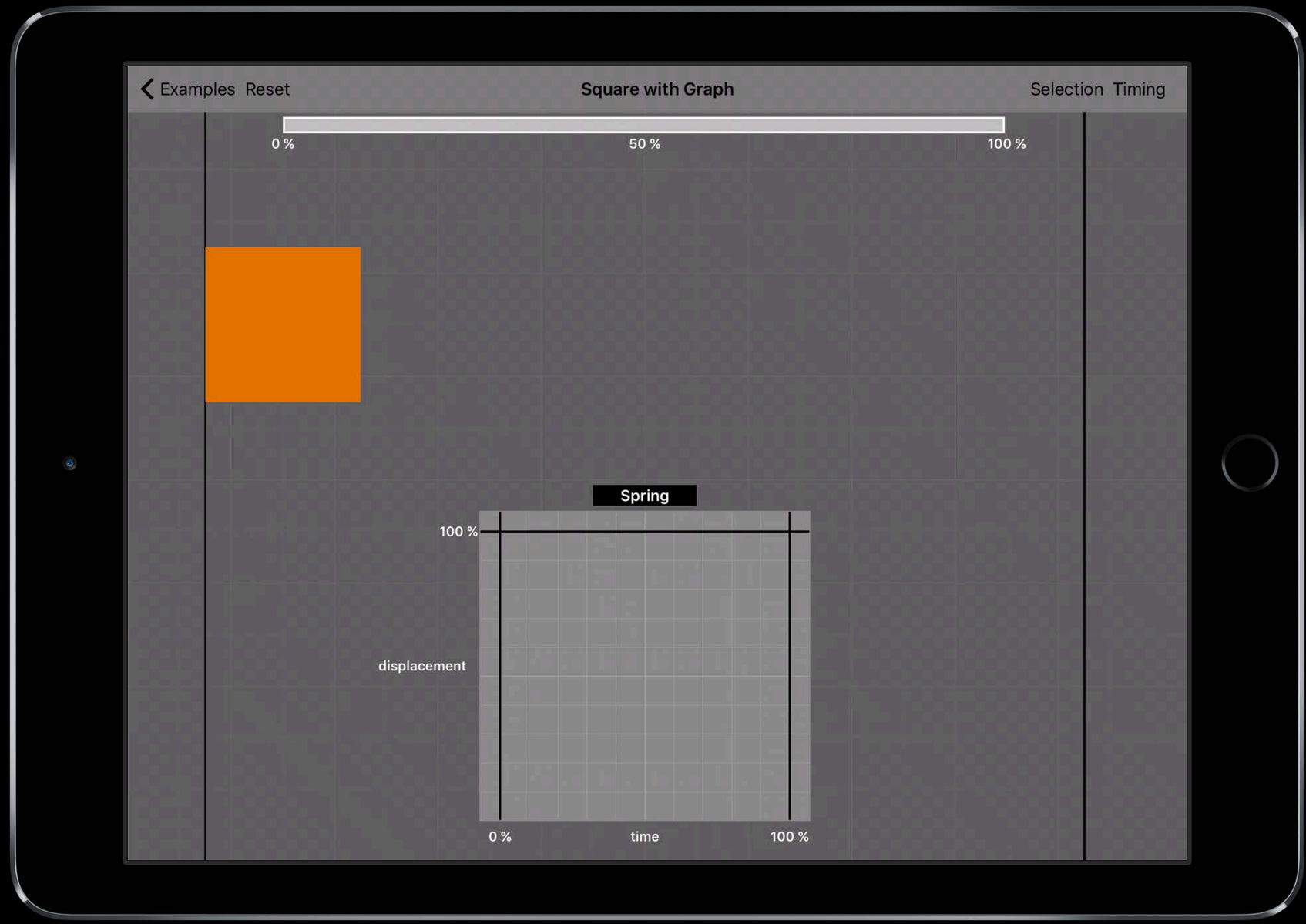
.easeOut

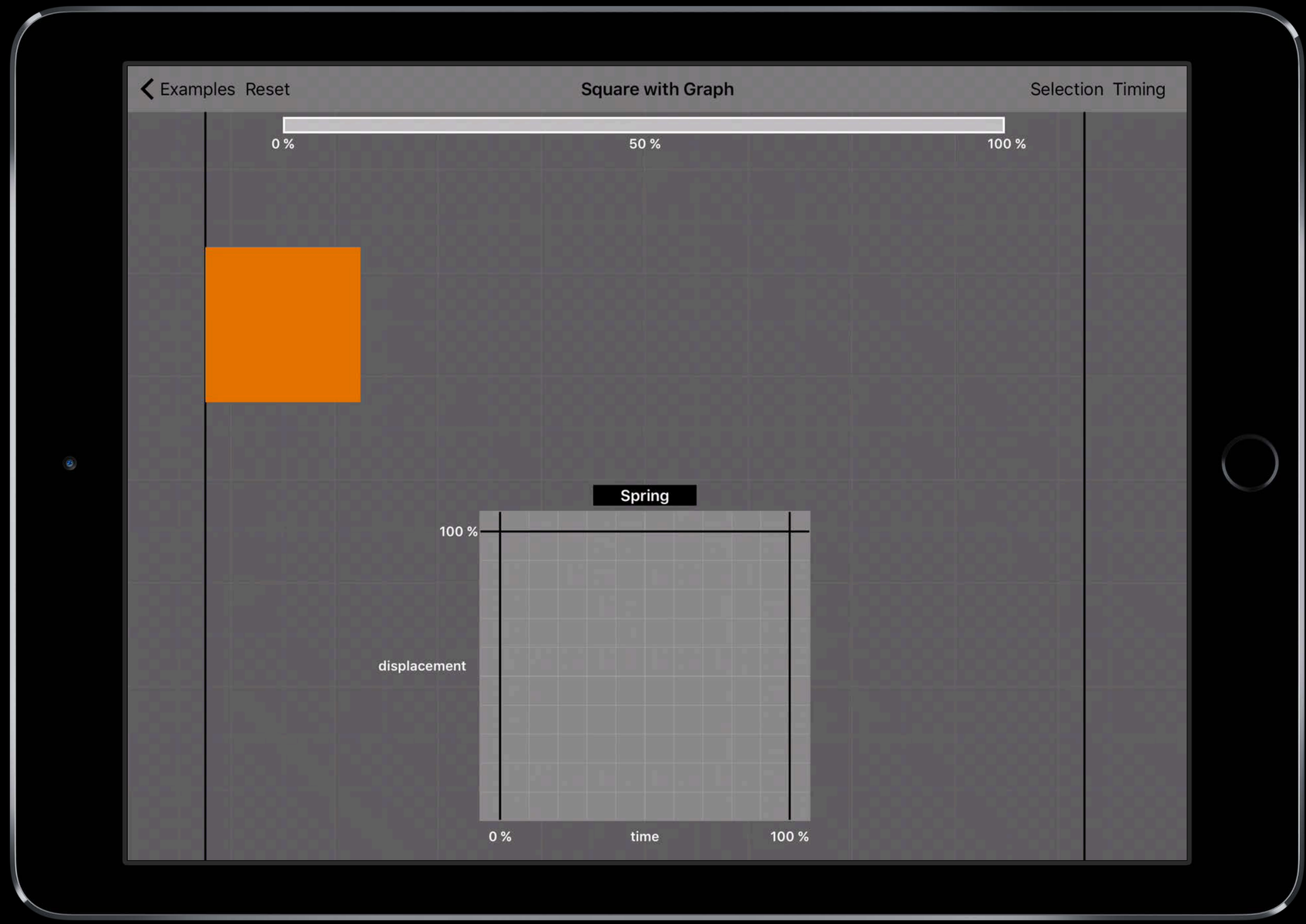



```
UIView.animate(withDuration:2.0,
                       delay: 0.0,
              usingSpringWithDamping: 0.8, initialSpringVelocity: 0.0,
                      options:[.linear]) {
   circle.center.x = 800.0
}
```

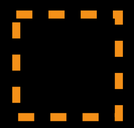



```
UIView.animate(withDuration:2.0,
                       delay: 0.0,
              usingSpringWithDamping: 0.8, initialSpringVelocity: 0.0,
                      options:[.linear]) {
   circle.center.x = 800.0
}
```


Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
Spring
100 %
displacement
0 %
time
100 %

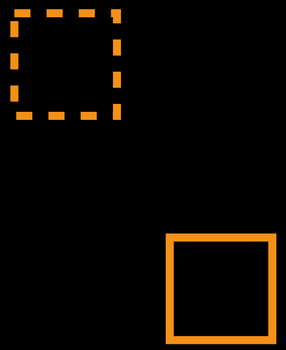

Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
Spring
100 %
displacement
0 %
time
100 %

Adding to Running Animations



Building Interruptible and Responsive Interactions

WWDC 2014

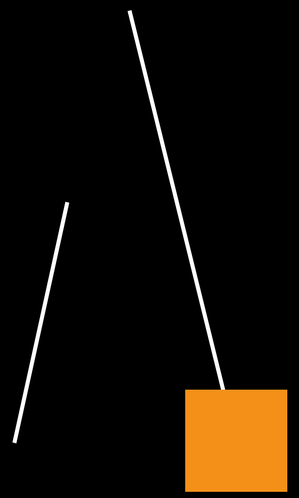


Building Interruptible and Responsive Interactions

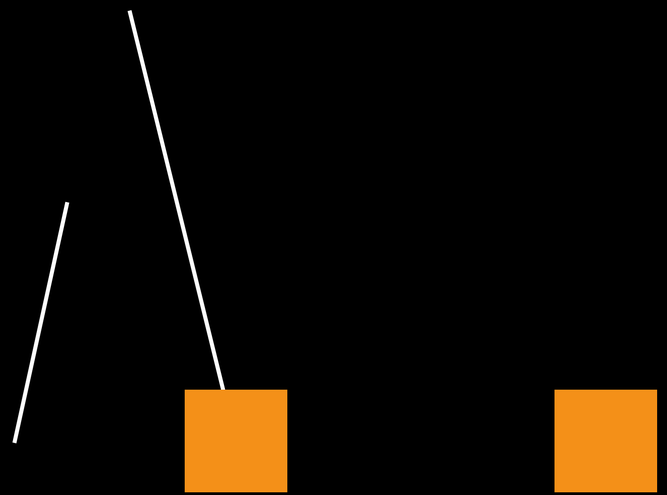
WWDC 2014

Building Interruptible and Responsive Interactions

WWDC 2014

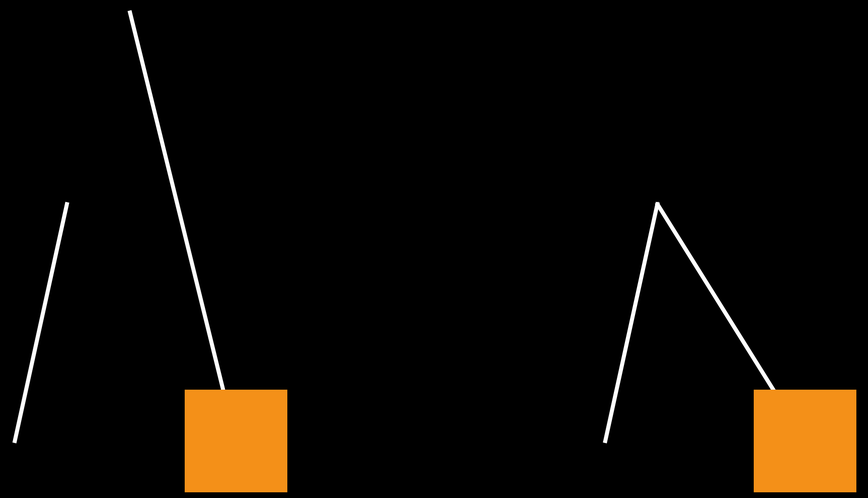


From Model



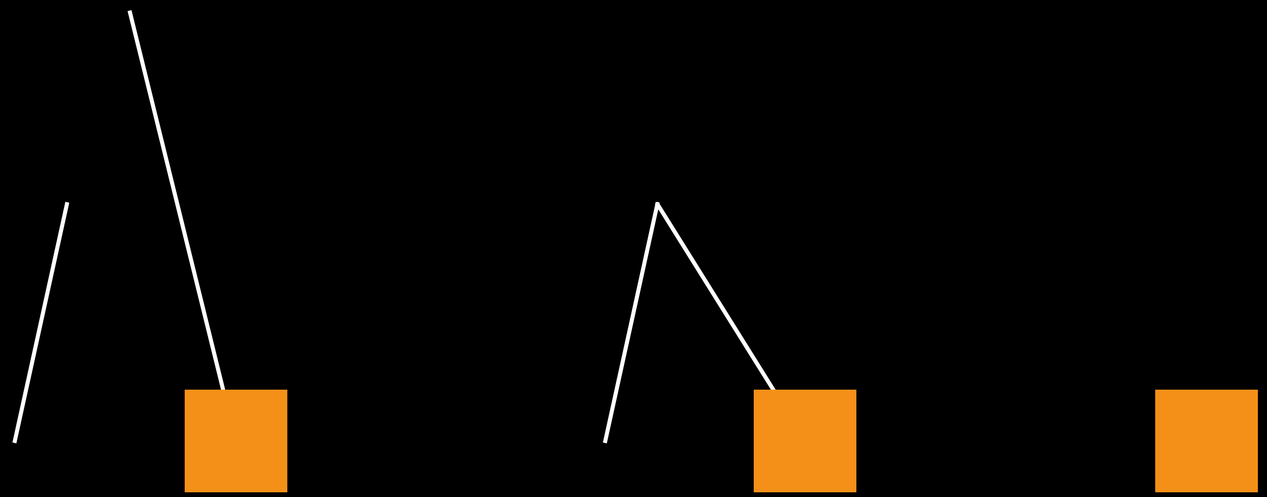
From Model

From Current



From Model

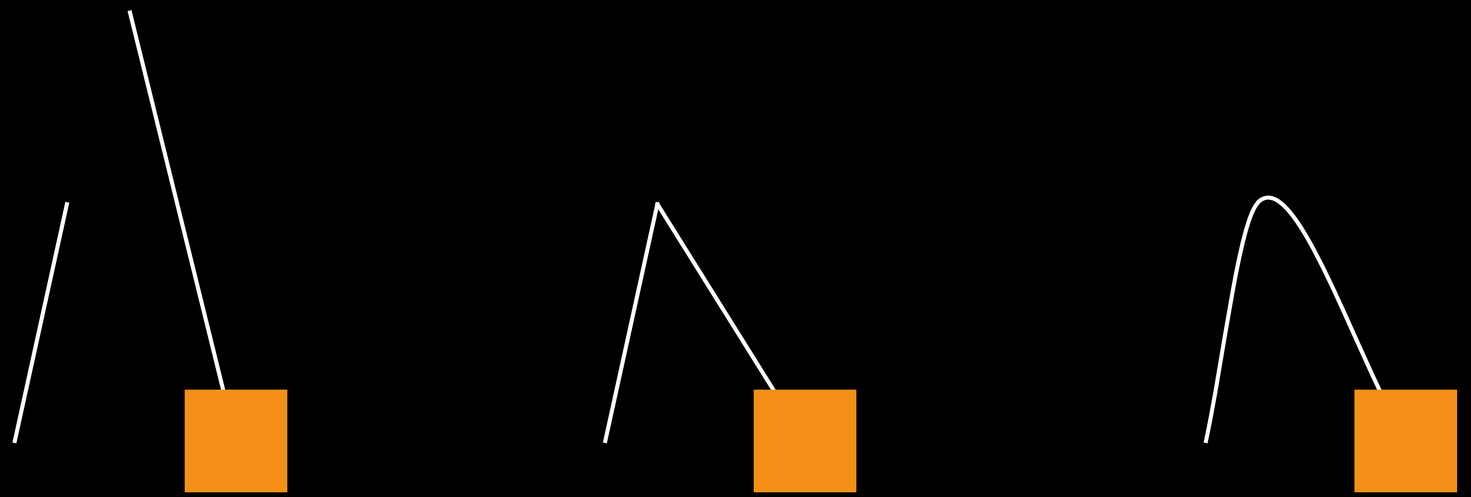
From Current



From Model

From Current

Additively



From Model

From Current

Additively

New Property Animation APIs

UIViewPropertyAnimator

UIViewPropertyAnimator

NEW

Features

Familiar

Interruptible

Scrubbable

Reversible

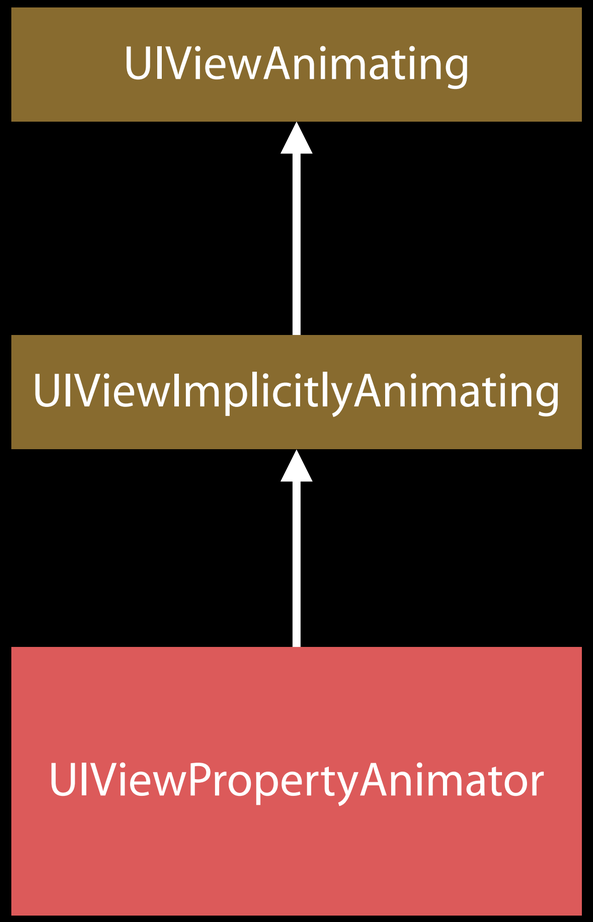
Broad availability of timing functions

Running animations can be modified

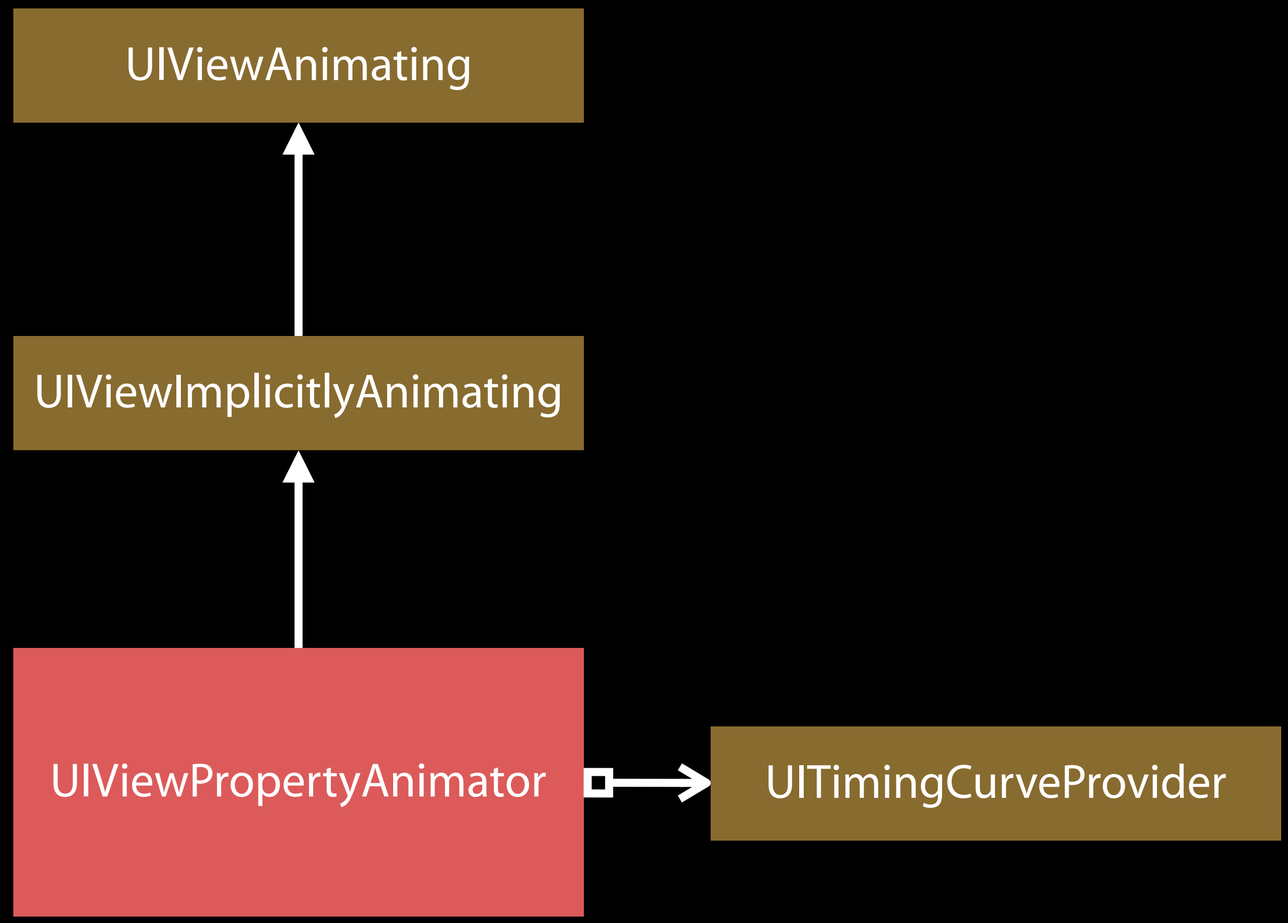
NEW

UIViewPropertyAnimator

NEW


UIViewAnimating
UIViewImplicitlyAnimating
UIViewPropertyAnimator

NEW


UIViewAnimating
UIViewImplicitlyAnimating
UIViewPropertyAnimator
UITimingCurveProvider

NEW

UIViewAnimating

UIViewImplicitlyAnimating

UIViewPropertyAnimator

UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

UIViewAnimatingState

UIViewAnimating

UIViewImplicitlyAnimating

UIViewAnimatingPosition

UIViewPropertyAnimator

UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

UIViewAnimatingState

UIViewAnimating

UIViewImplicitlyAnimating

UIViewAnimatingPosition

UIViewPropertyAnimator

UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

UIViewAnimatingState

.inactive
.active
.stopped

UIViewAnimating

UIViewImplicitlyAnimating

UIViewPropertyAnimator

UIViewAnimatingPosition

UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

UIViewAnimatingState

UIViewAnimating

UIViewImplicitlyAnimating

UIViewPropertyAnimator

UIViewAnimatingPosition

.end

.start

.current

UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

UIViewAnimatingState

UIViewAnimating

UIViewImplicitlyAnimating

UIViewAnimatingPosition

UIViewPropertyAnimator

UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

UIViewAnimatingState

UIViewAnimating

UIViewImplicitlyAnimating

UIViewAnimatingPosition

UIViewPropertyAnimator

UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

UIViewAnimatingState

UIViewAnimating

UIViewImplicitlyAnimating

UIViewAnimatingPosition

UIViewPropertyAnimator

UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

UIViewAnimatingState

UIViewAnimating

UIViewImplicitlyAnimating

UIViewAnimatingPosition

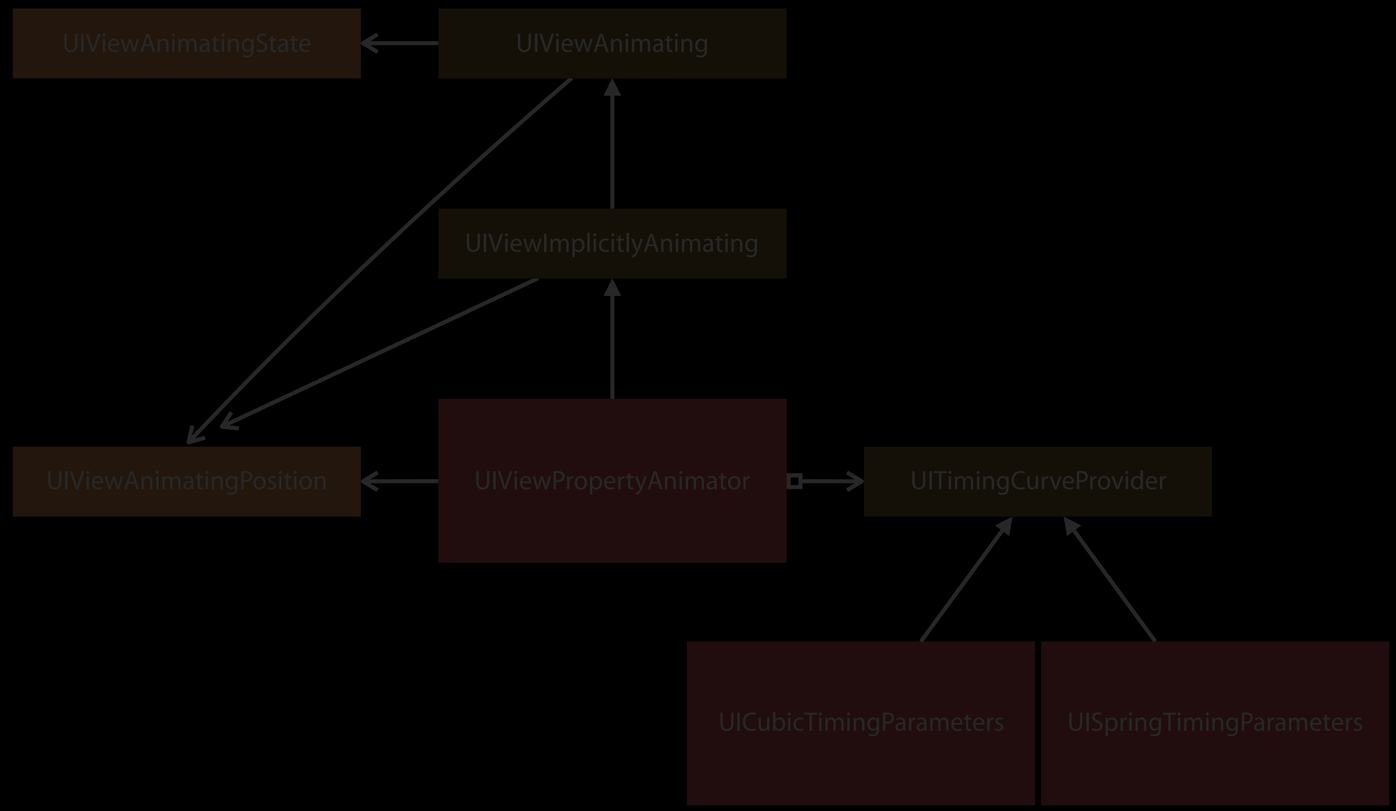
UIViewPropertyAnimator

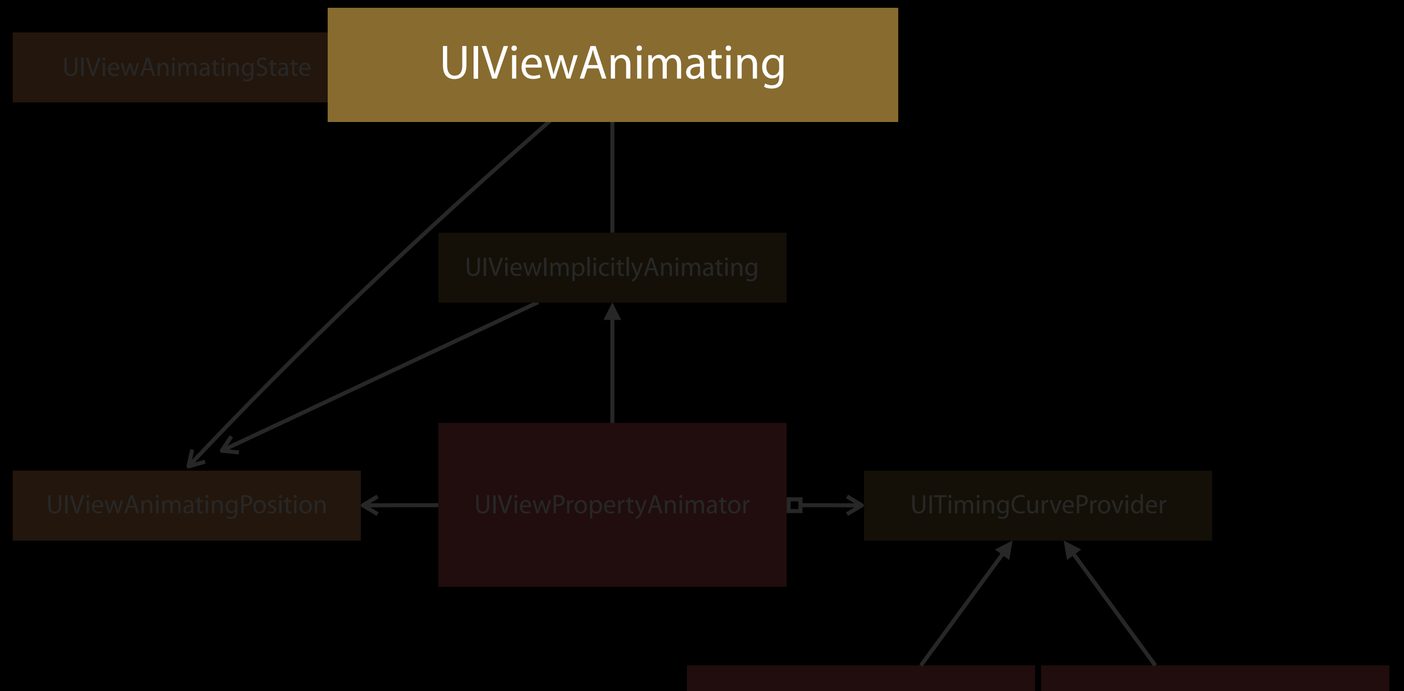
UITimingCurveProvider

UICubicTimingParameters

UISpringTimingParameters

NEW

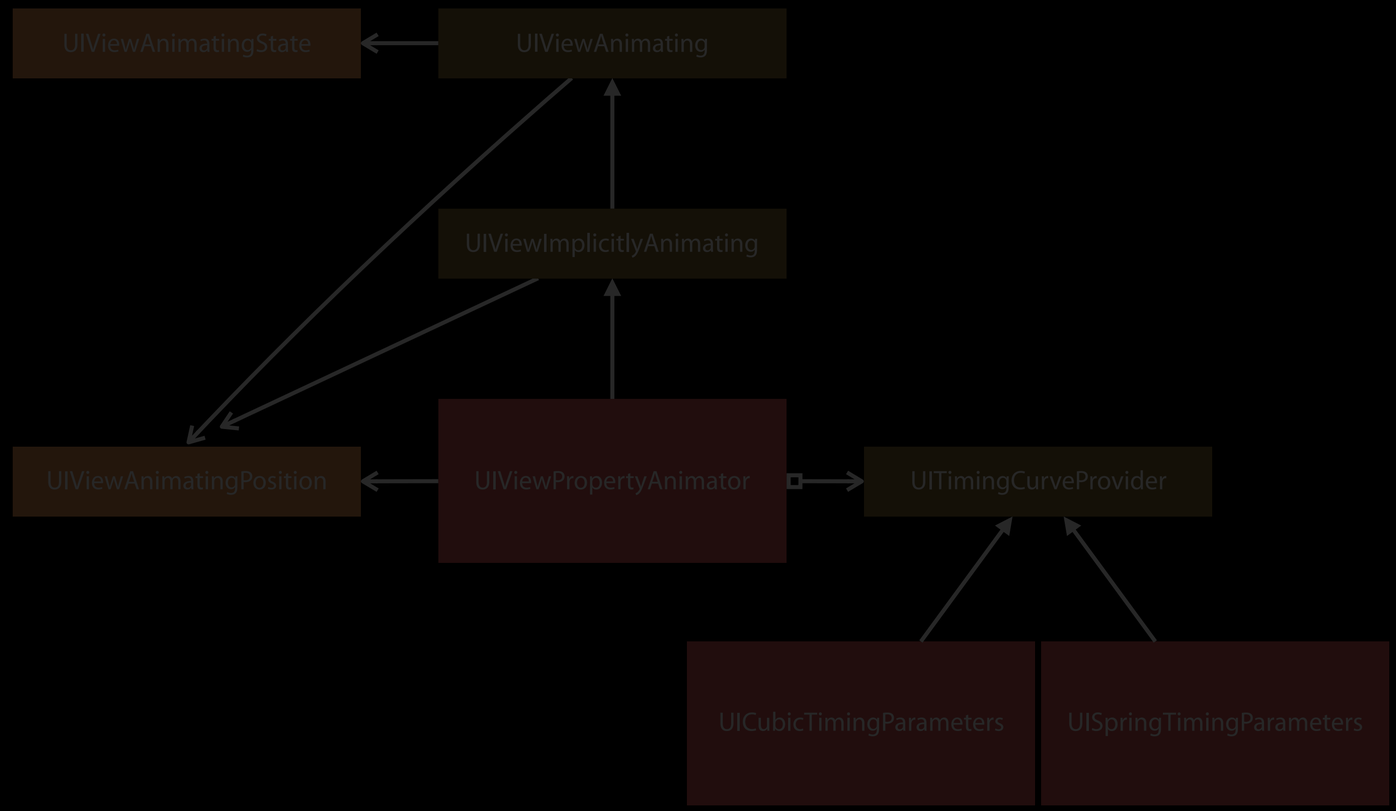

UIViewAnimatingState
UIViewAnimating
UIViewImplicitlyAnimating
UIViewAnimatingPosition
UIViewPropertyAnimator
UITimingCurveProvider
UICubicTimingParameters
UISpringTimingParameters

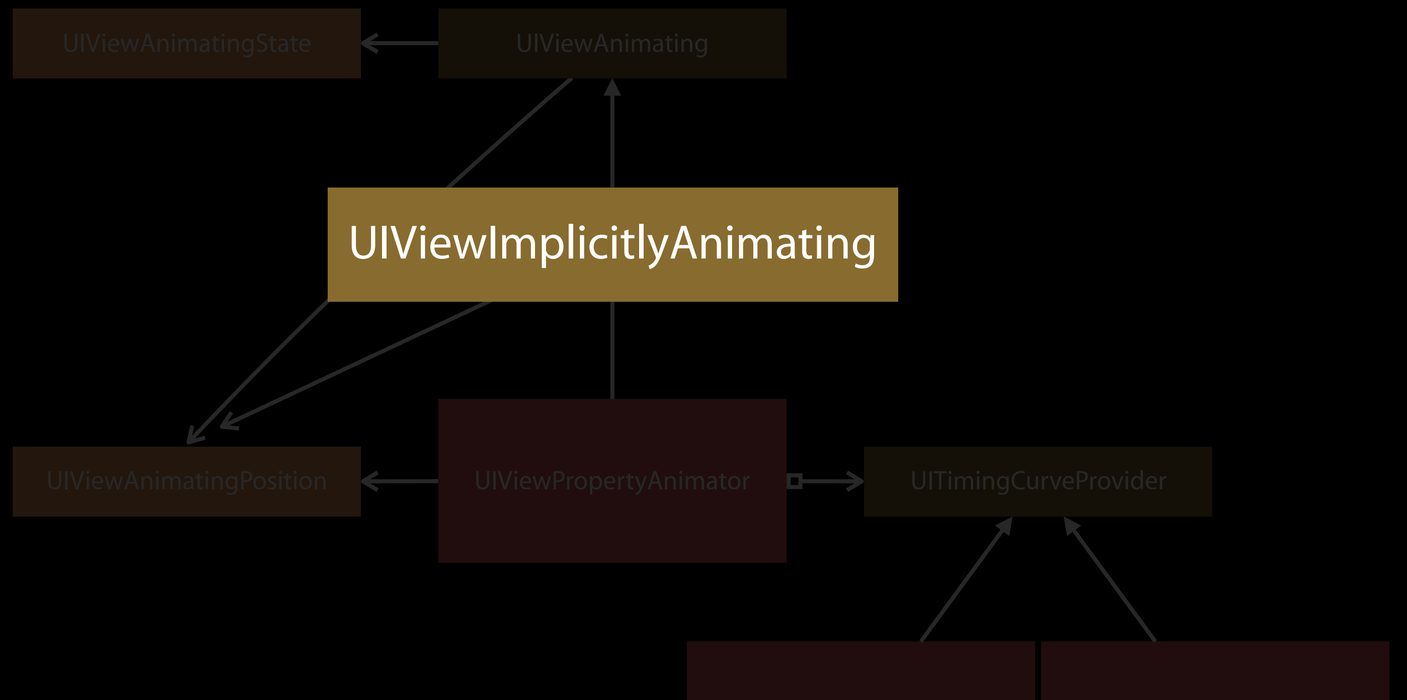


NEW

```
var state: UIViewAnimatingState { get }
var isRunning: Bool { get }
var isReversed: Bool { get set }
var fractionComplete: CGFloat { get set }

func startAnimation()
func startAnimation(afterDelay : TimInterval)

func pauseAnimation()

func stopAnimation(_ withoutFinishing: Bool)
func finishAnimation(at finalPosition: UIViewAnimatingPosition)
```

NEW

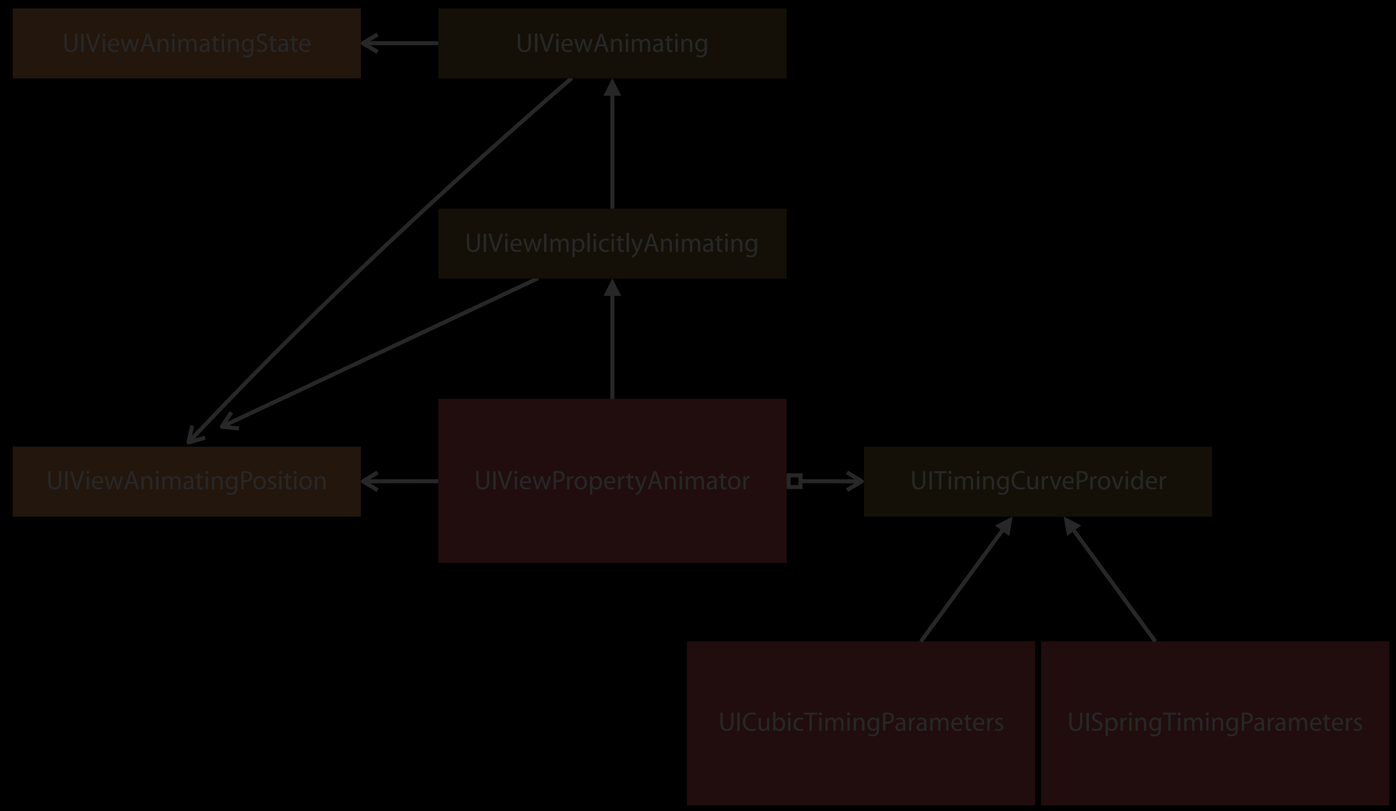

UIViewAnimatingState
UIViewAnimating
UIViewImplicitlyAnimating
UIViewAnimatingPosition
UIViewPropertyAnimator
UITimingCurveProvider
UICubicTimingParameters
UISpringTimingParameters

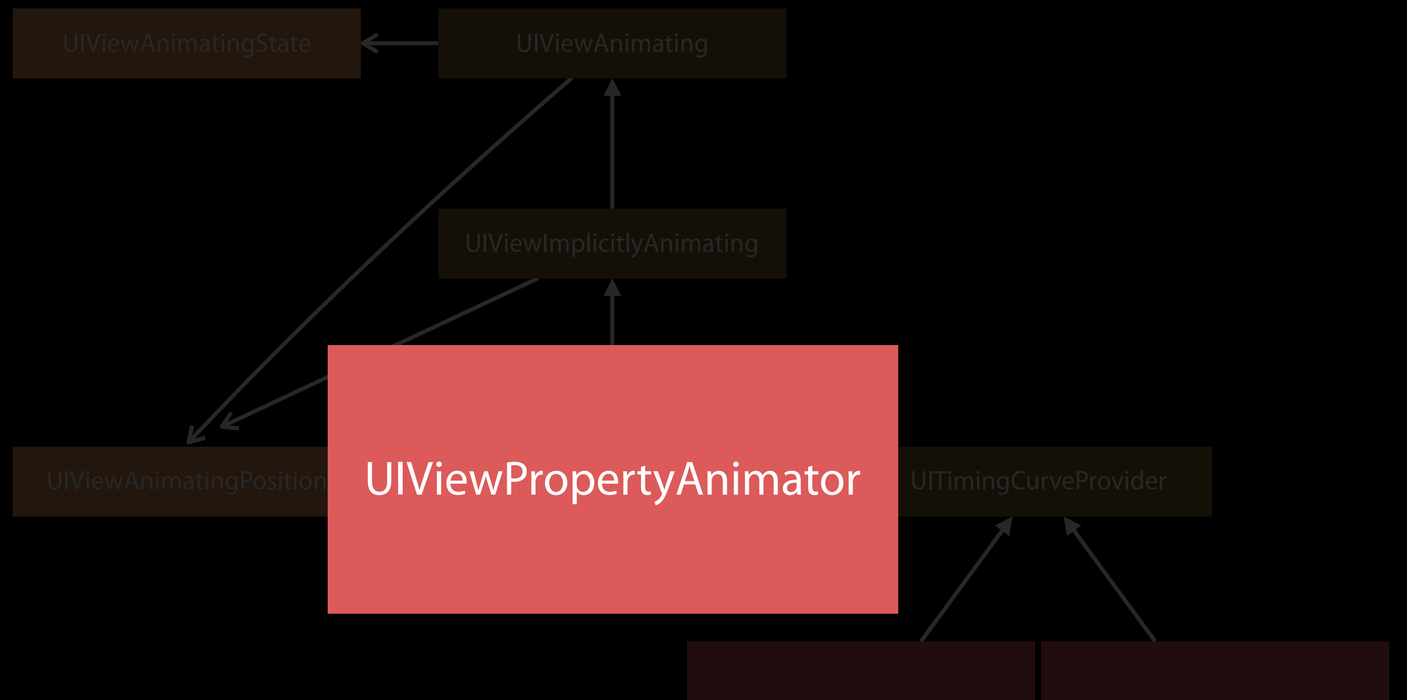


NEW

```
optional func addAnimations(_ animation: () -> Void, delayFactor: CGFloat)
optional func addAnimations(_ animation: () -> Void)


optional func addCompletion(_ completion: (UIViewAnimatingPosition) -> Void)

optional func continueAnimation(withTimingParameters parameters:
                    UITimingCurveProvider?, durationFactor: CGFloat)
```

NEW


UIViewAnimatingState
UIViewAnimating
UIViewImplicitlyAnimating
UIViewAnimatingPosition
UIViewPropertyAnimator
UITimingCurveProvider
UICubicTimingParameters
UISpringTimingParameters



NEW

```
var timingParameters: UITimingCurveProvider? { get }
var duration: TimeInterval { get }
var delay: TimeInterval { get }
var isUserInteractionEnabled: Bool { get set }
var isManualHitTestingEnabled: Bool { get set }
var isInterruptible: Bool { get set }

init(duration: TimeInterval, timingParameters parameters: UITimingCurveProvider)

class func runningPropertyAnimator(withDuration duration: TimeInterval,
    delay: TimeInterval, options: UIViewAnimationOptions = [],
    animations: (() -> Void)?,
    completion: ((UIViewAnimatingPosition) -> Void)? = nil) -> Self
```

UIViewPropertyAnimator

API discussion

UIViewPropertyAnimator

API discussion

Basic usage

UIViewPropertyAnimator

API discussion

Basic usage

Pausing and scrubbing

UIViewPropertyAnimator

API discussion

Basic usage

Pausing and scrubbing

Reversing

UIViewPropertyAnimator

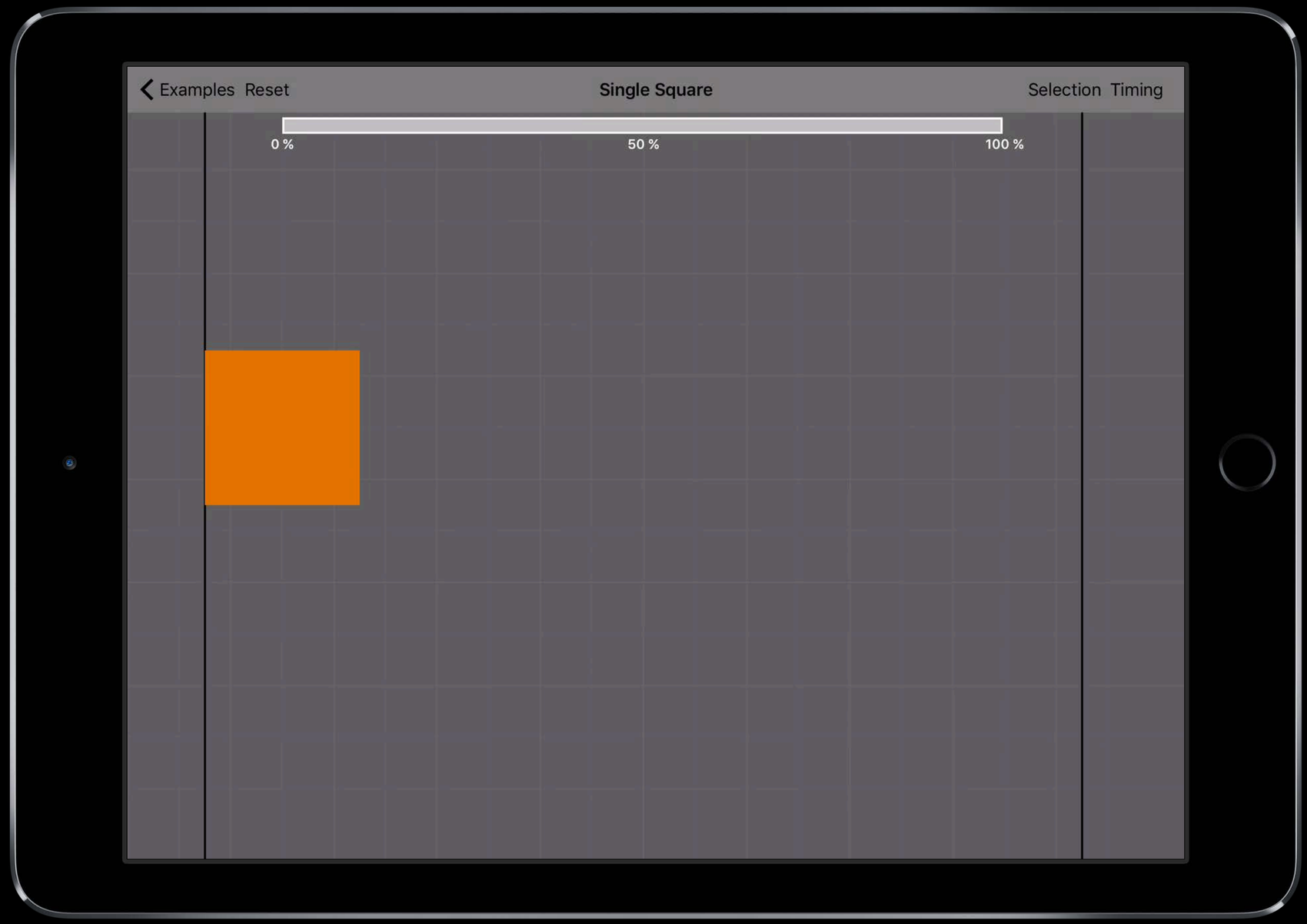
API discussion

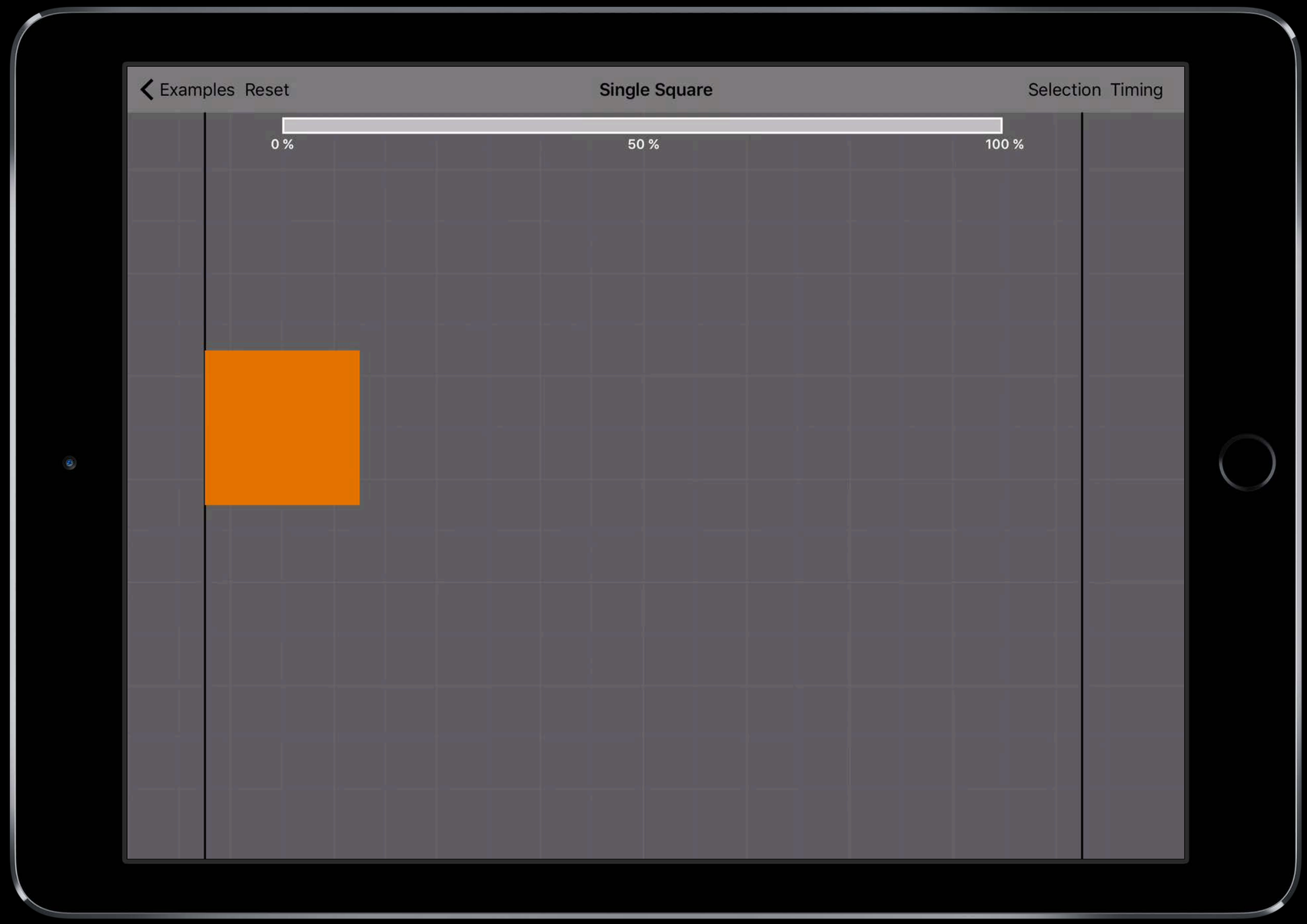
Basic usage

Pausing and scrubbing

Reversing

Timing providers


Examples
Reset
Single Square
Selection
Timing
0 %
50 %
100 %


Examples
Reset
Single Square
Selection
Timing
0 %
50 %
100 %

UIViewPropertyAnimator

Basics

```
let timing   = UICubicTimingParameters(animationCurve: .easeInOut)

let animator = UIViewPropertyAnimator(duration: 2.0, timingParameters:timing)

animator.addAnimations {
  self.squareView.center = CGPoint(x: 800.0, y: self.squareView.center,y)
  self.squareView.transform = CGAffineTransform(rotationAngle: CGFloat(M_PI_2))
}

animator.addCompletion {_ in
  self.squareView.backgroundColor = UIColor.orange()
}

animator.startAnimation()
```

UIViewPropertyAnimator

Basics

```
let timing   = UICubicTimingParameters(animationCurve: .easeInOut)

let animator = UIViewPropertyAnimator(duration: 2.0, timingParameters:timing)

animator.addAnimations {
  self.squareView.center = CGPoint(x: 800.0, y: self.squareView.center,y)
  self.squareView.transform = CGAffineTransform(rotationAngle: CGFloat(M_PI_2))
}

animator.addCompletion {_ in
  self.squareView.backgroundColor = UIColor.orange()
}

animator.startAnimation()
```

UIViewPropertyAnimator

Basics

```
let timing   = UICubicTimingParameters(animationCurve: .easeInOut)

let animator = UIViewPropertyAnimator(duration: 2.0, timingParameters:timing)

animator.addAnimations {
  self.squareView.center = CGPoint(x: 800.0, y: self.squareView.center,y)
  self.squareView.transform = CGAffineTransform(rotationAngle: CGFloat(M_PI_2))
}

animator.addCompletion {_ in
  self.squareView.backgroundColor = UIColor.orange()
}

animator.startAnimation()
```

UIViewPropertyAnimator

Basics

```
let timing   = UICubicTimingParameters(animationCurve: .easeInOut)

let animator = UIViewPropertyAnimator(duration: 2.0, timingParameters:timing)

animator.addAnimations {
  self.squareView.center = CGPoint(x: 800.0, y: self.squareView.center,y)
  self.squareView.transform = CGAffineTransform(rotationAngle: CGFloat(M_PI_2))
}

animator.addCompletion {_ in
  self.squareView.backgroundColor = UIColor.orange()
}

animator.startAnimation()
```

UIViewPropertyAnimator

Basics

```
let timing   = UICubicTimingParameters(animationCurve: .easeInOut)

let animator = UIViewPropertyAnimator(duration: 2.0, timingParameters:timing)

animator.addAnimations {
  self.squareView.center = CGPoint(x: 800.0, y: self.squareView.center,y)
  self.squareView.transform = CGAffineTransform(rotationAngle: CGFloat(M_PI_2))
}

animator.addCompletion {_ in
  self.squareView.backgroundColor = UIColor.orange()
}

animator.startAnimation()
```

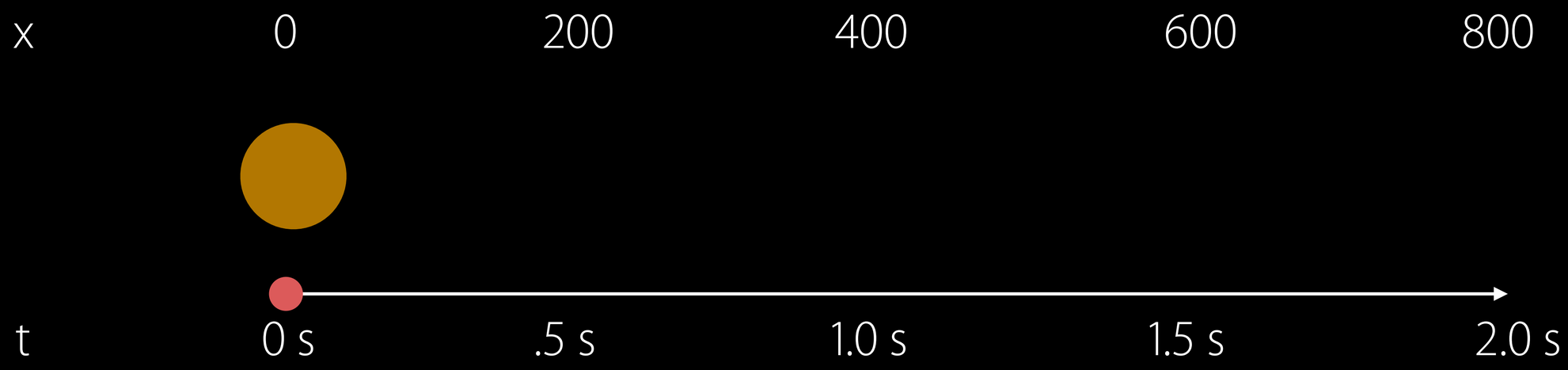
UIViewPropertyAnimator

Basics

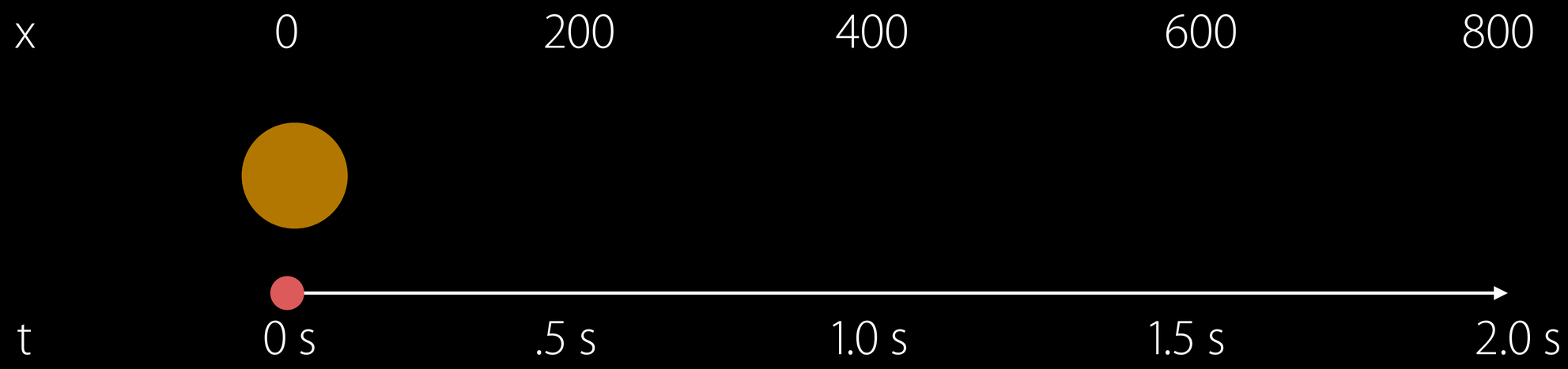
```
let timing   = UICubicTimingParameters(animationCurve: .easeInOut)

let animator = UIViewPropertyAnimator(duration: 2.0, timingParameters:timing)

animator.addAnimations {
  self.squareView.center = CGPoint(x: 800.0, y: self.squareView.center,y)
  self.squareView.transform = CGAffineTransform(rotationAngle: CGFloat(M_PI_2))
}

animator.addCompletion {_ in
  self.squareView.backgroundColor = UIColor.orange()
}

animator.startAnimation()
```

UIViewPropertyAnimator

Observable properties

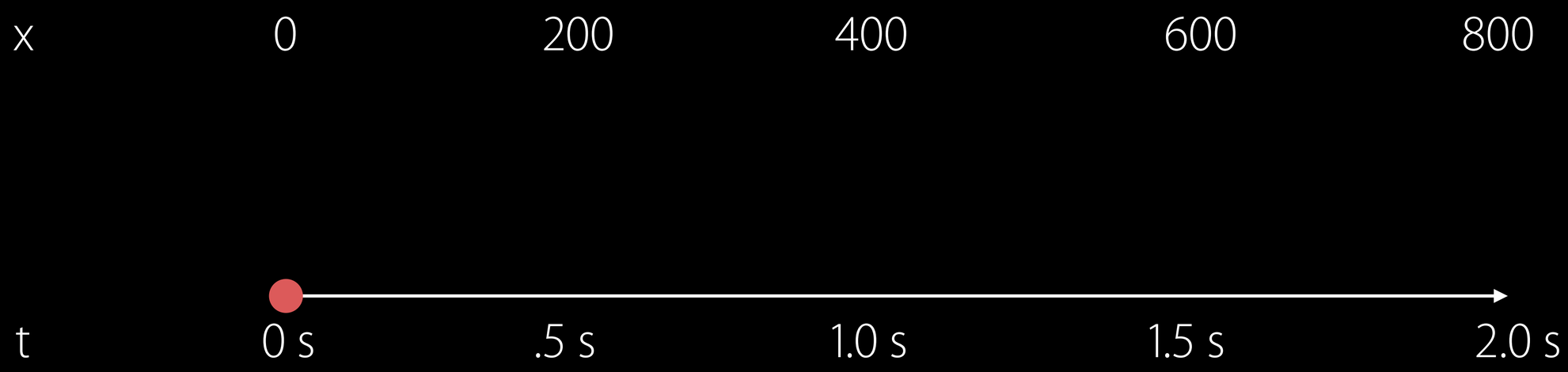


state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false



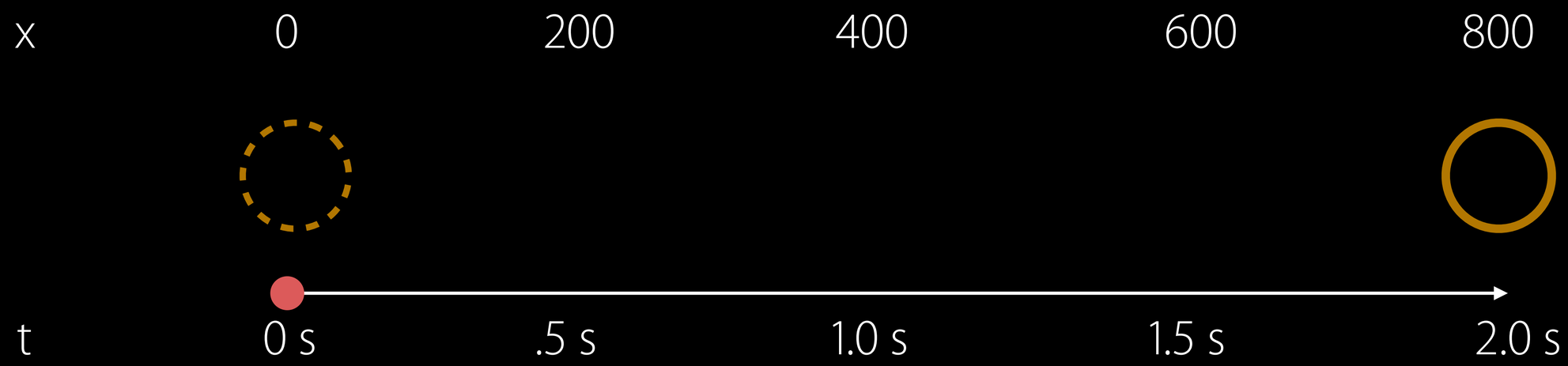
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.addAnimation {
   circle.center.x = 800.0
}
```



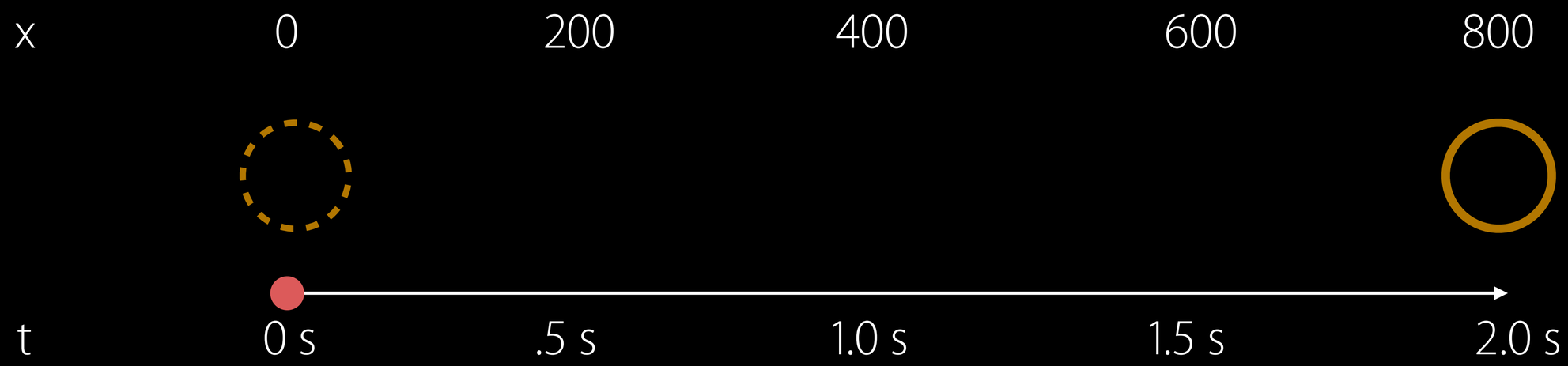
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.startAnimation ()
```

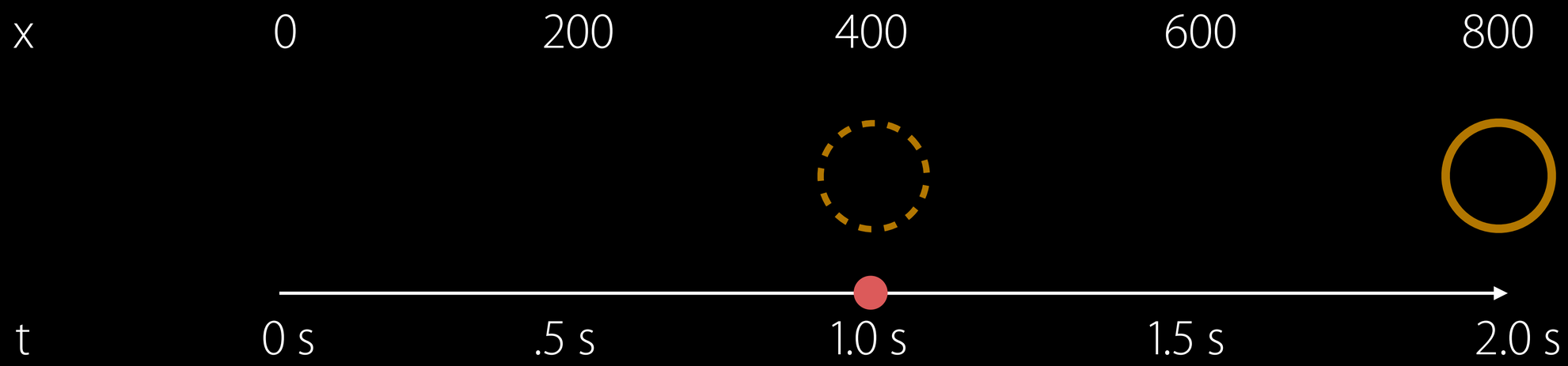


state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

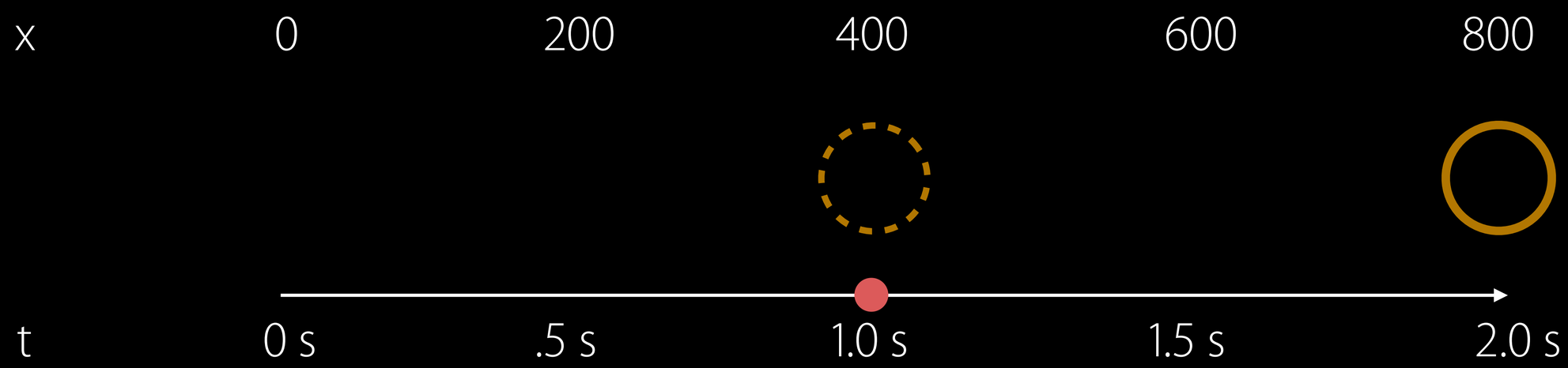
```
animator.startAnimation ()
```



state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

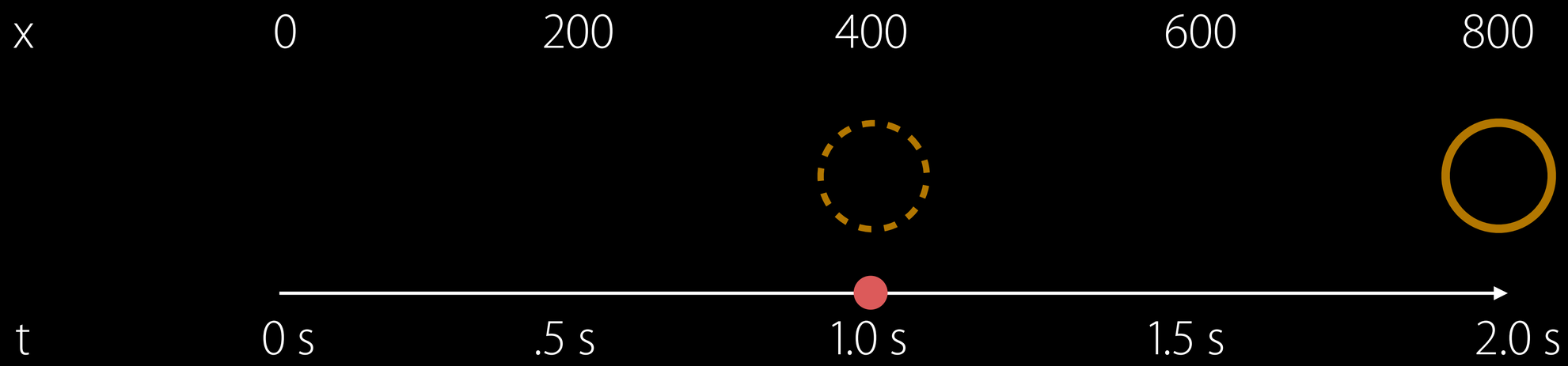


state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false



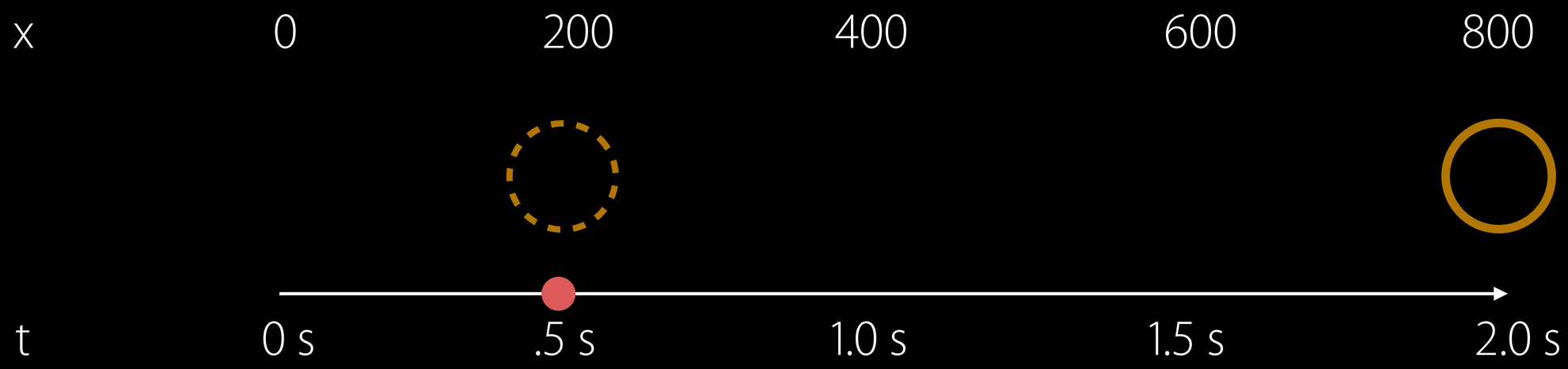
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.pauseAnimation ()
```



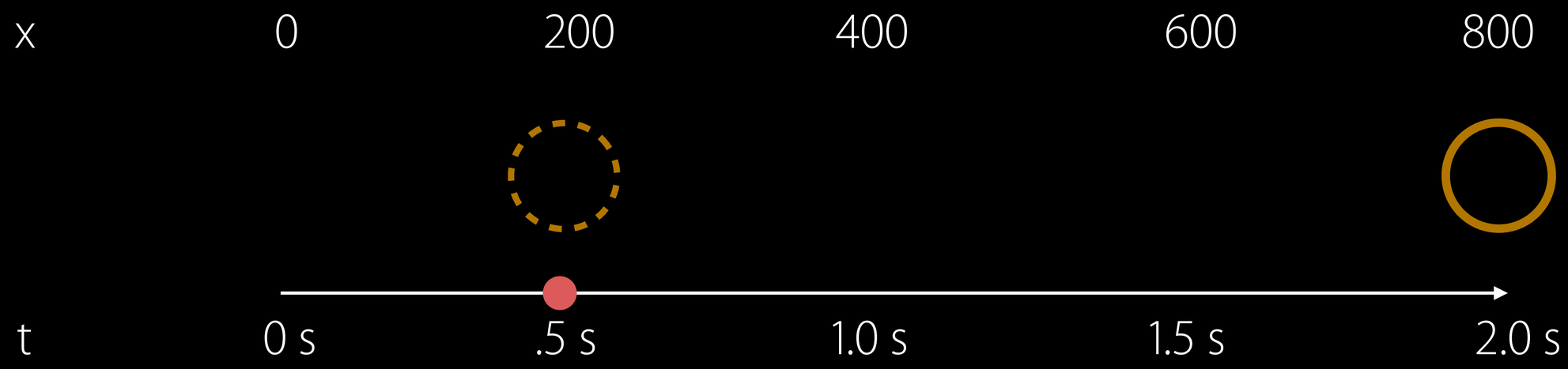
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.pauseAnimation ()

animator.isReversed = true
```



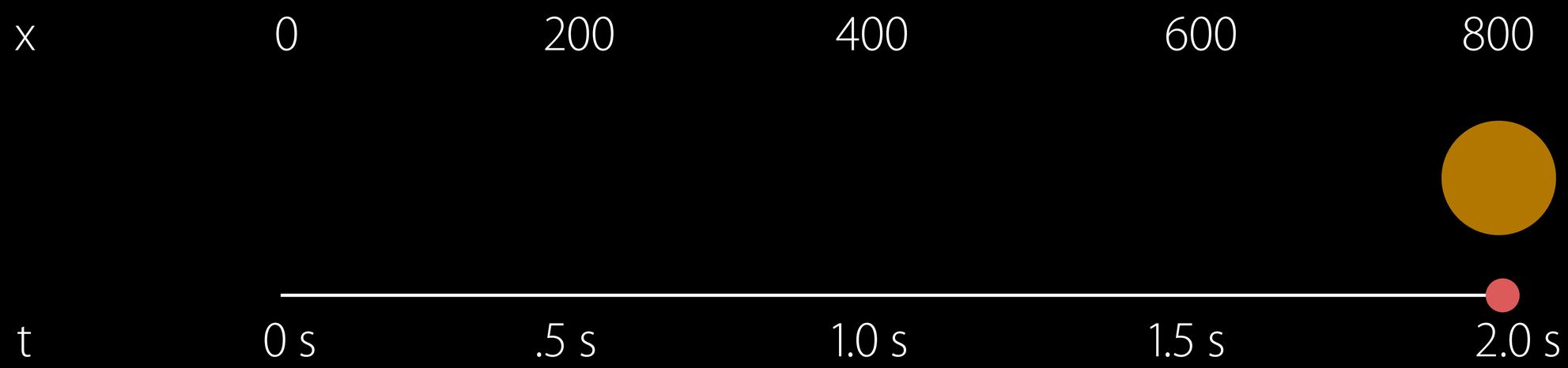
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.pauseAnimation ()
animator.isReversed = true
animator.startAnimation ()
```



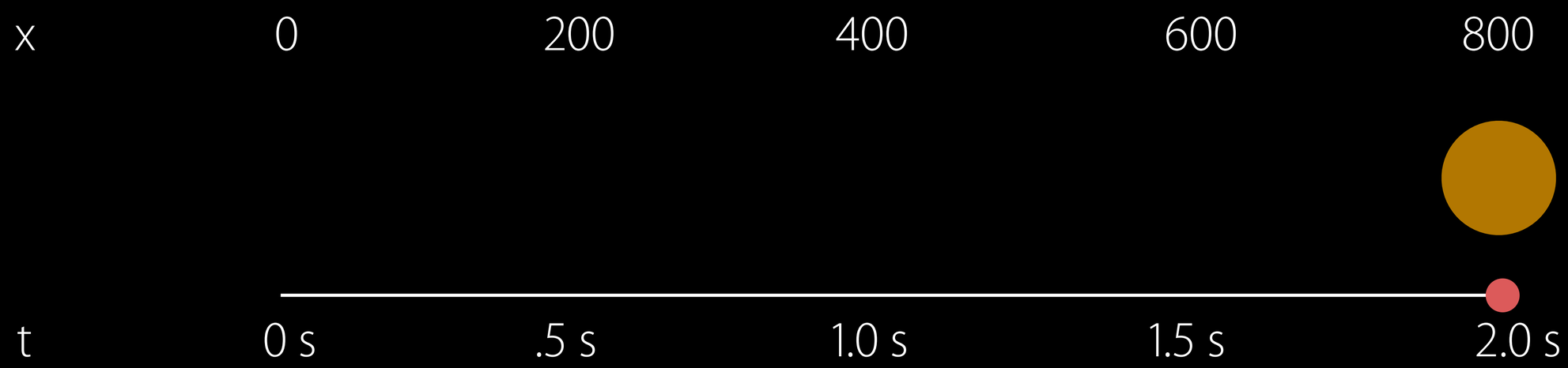
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.isReversed = false
```



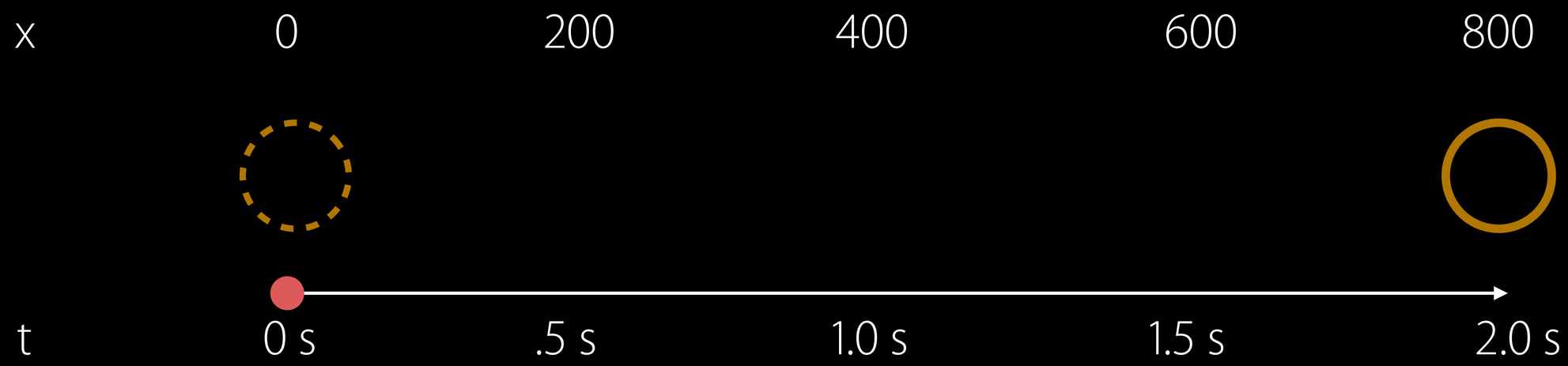
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.isReversed = false
```

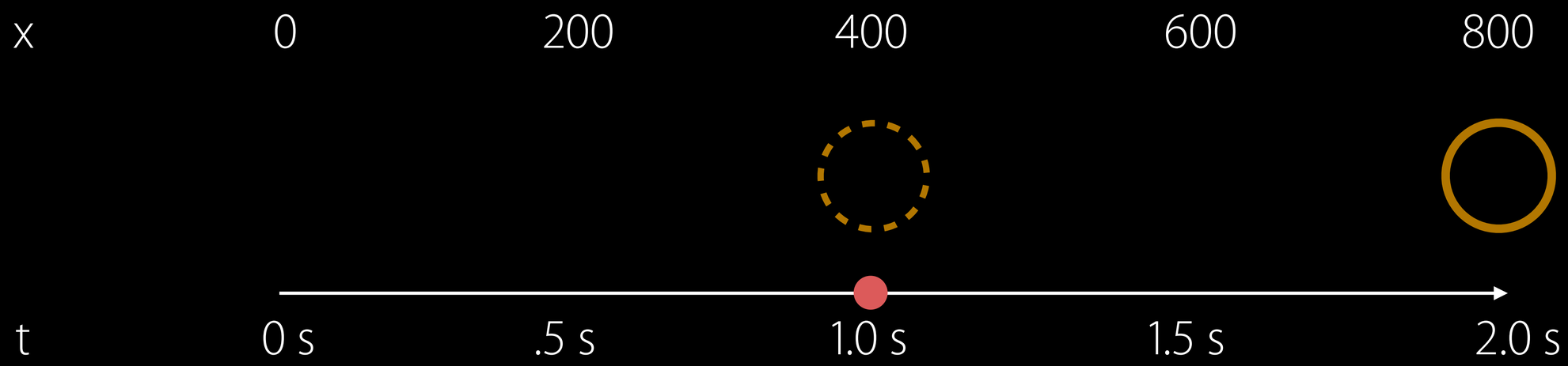


state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

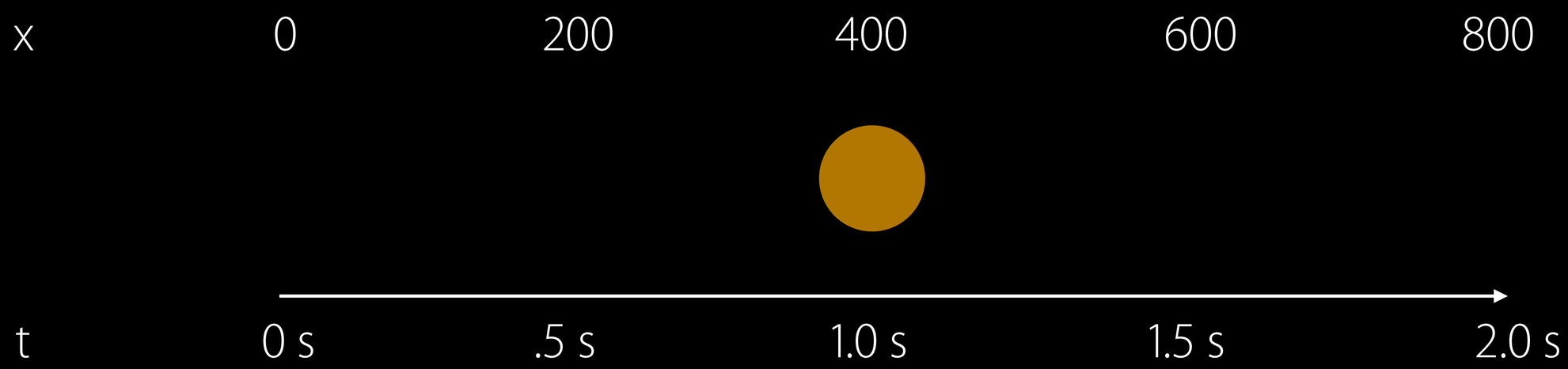
```
animator.isReversed = false

completion(.end)
```



state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

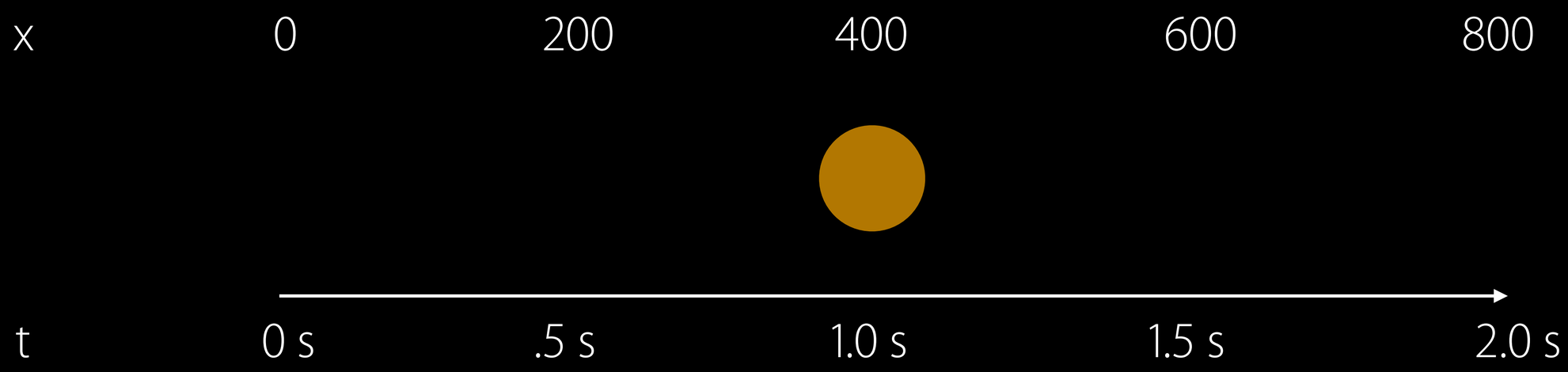


state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false



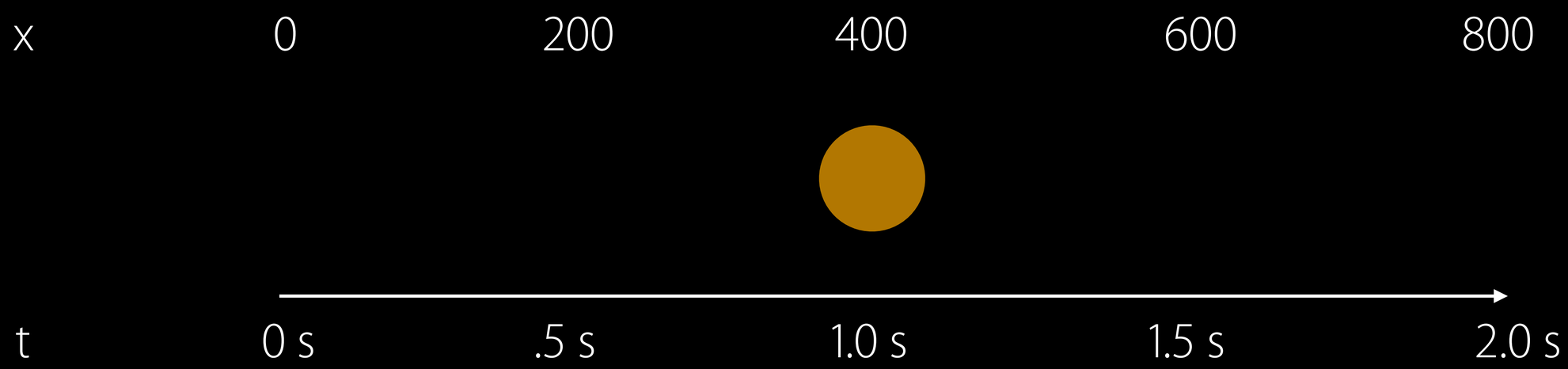
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.stopAnimation (false)
```



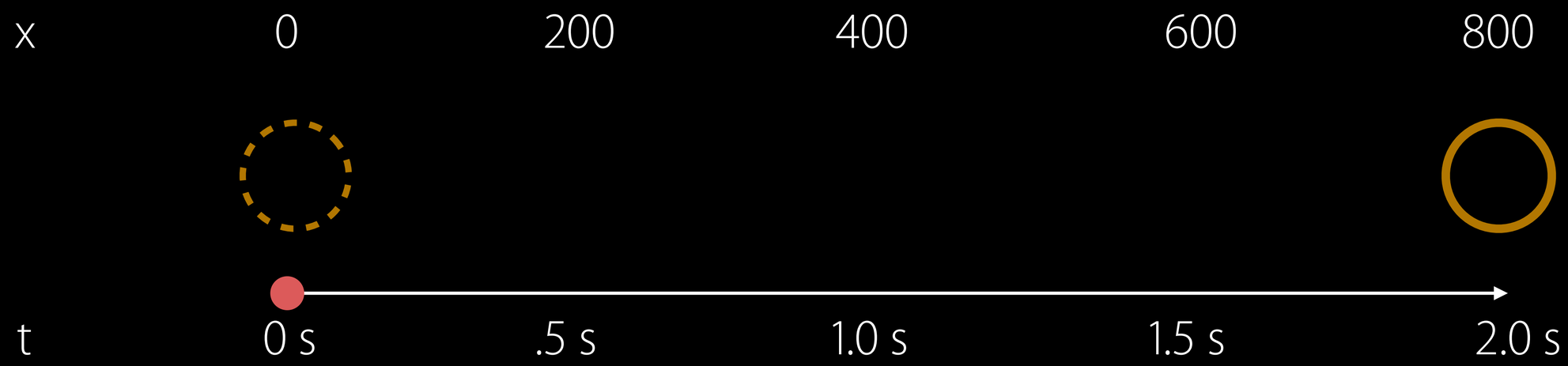
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.stopAnimation (false)

animator.finishAnimation (.current)
```

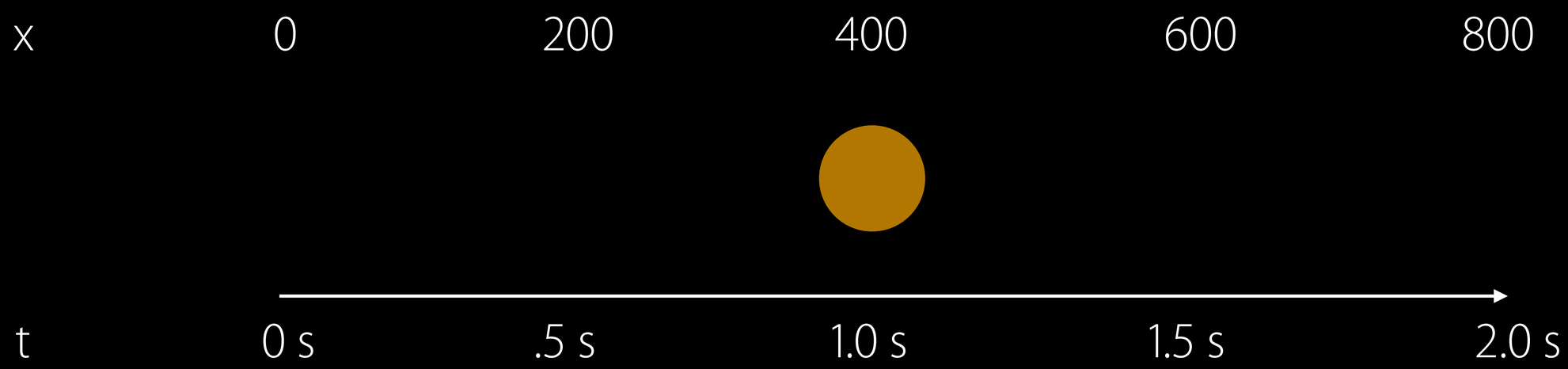


state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.stopAnimation (false)

animator.finishAnimation (.current)

completion(.current)
```

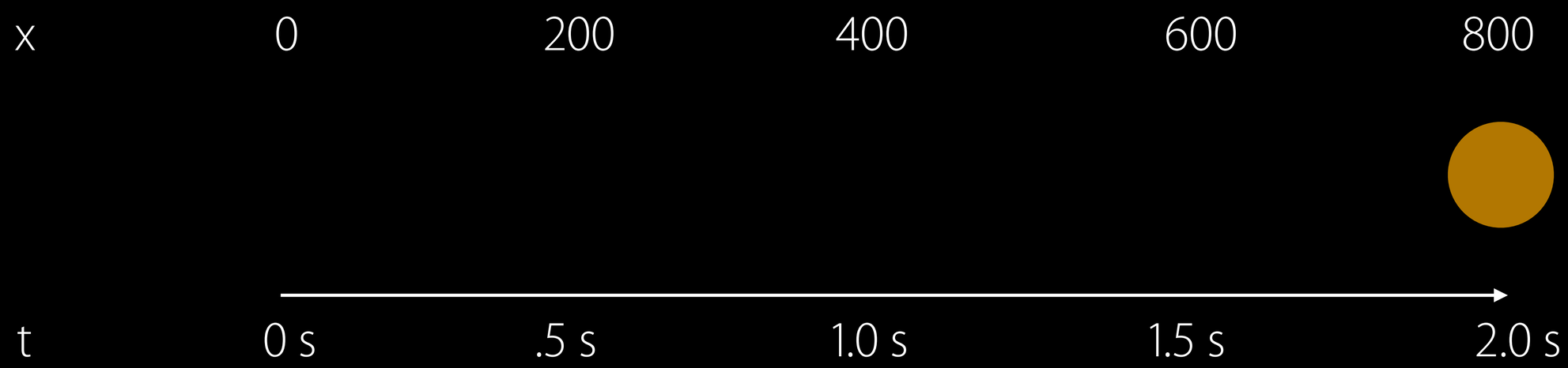


state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false



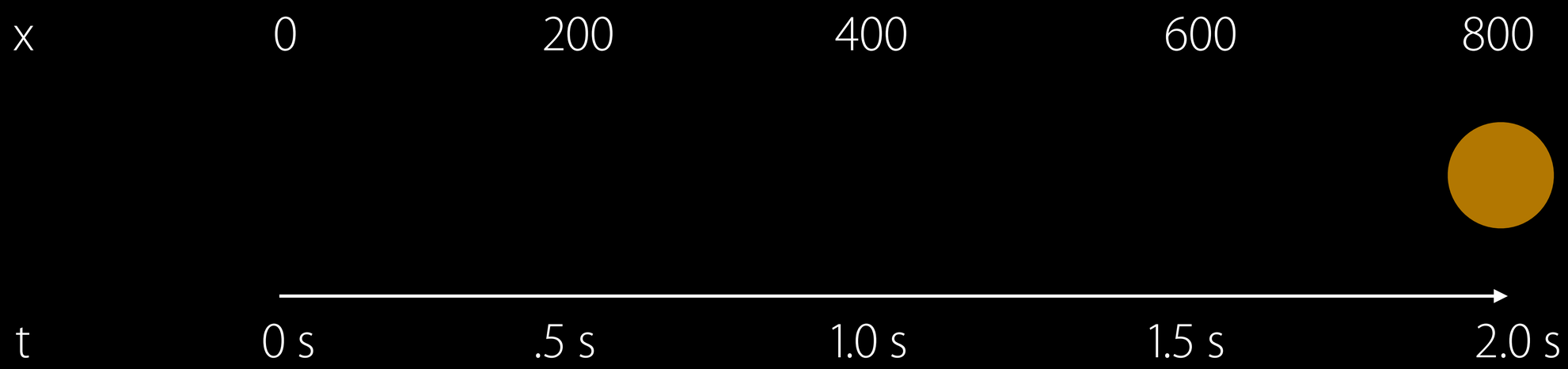
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.stopAnimation (false)
```



state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.stopAnimation (false)

animator.finishAnimation (.end)
```

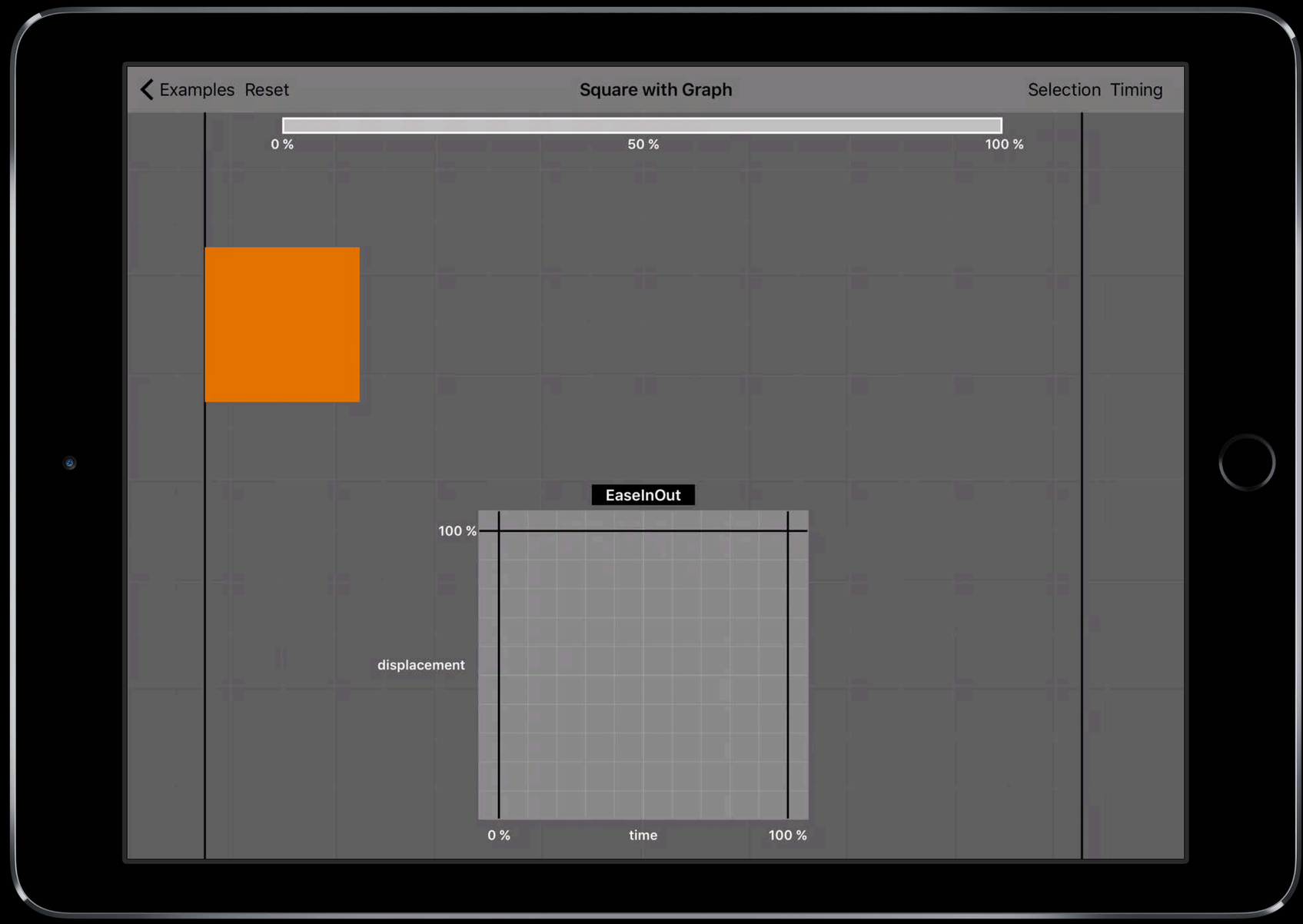


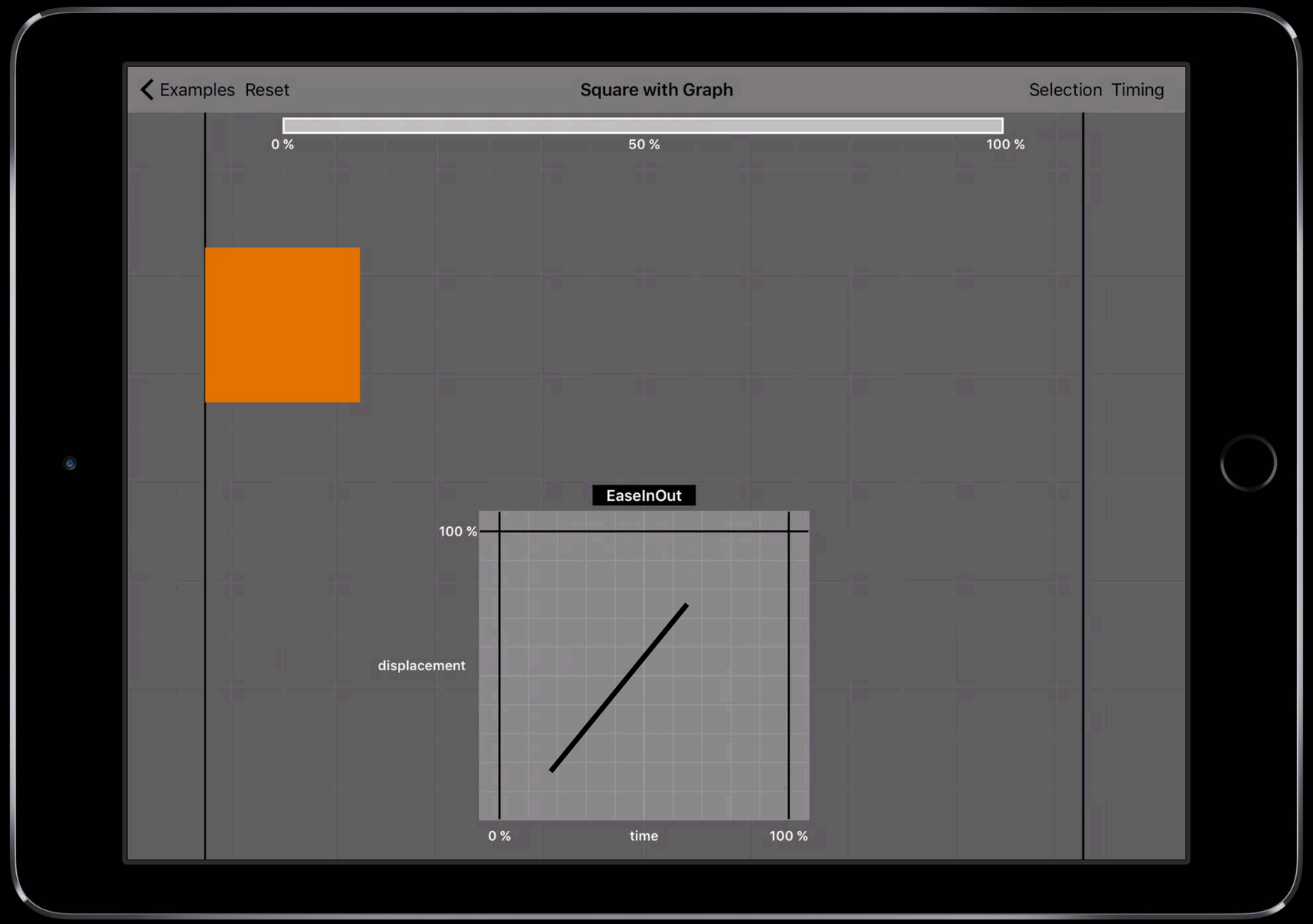
state			isRunning		isReversed	
.inactive	.active	.stopped	true	false	true	false

```
animator.stopAnimation (false)

animator.finishAnimation (.end)

completion(.end)
```

UIViewPropertyAnimator

Pausing and scrubbing


Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %


Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %

UIViewPropertyAnimator

Pausing and scrubbing

```
switch animator.state {
  case .active:
    if animator.isRunning {
      progressAnimator.pauseAnimation();
      animator.pauseAnimation();
    }
    else {
      animator.startAnimation()
      progressAnimator.startAnimation()
    }
  default:
    break
  }
}
```

UIViewPropertyAnimator

Pausing and scrubbing

```
switch animator.state {
  case .active:
    if animator.isRunning {
      progressAnimator.pauseAnimation();
      animator.pauseAnimation();
    }
    else {
      animator.startAnimation()
      progressAnimator.startAnimation()
    }
  default:
    break
  }
}
```

UIViewPropertyAnimator

Pausing and scrubbing

```
switch animator.state {
  case .active:
    if animator.isRunning {
      progressAnimator.pauseAnimation();
      animator.pauseAnimation();
     }
     else {
       animator.startAnimation()
       progressAnimator.startAnimation()
     }
   default:
     break
   }
 }
```

UIViewPropertyAnimator

Pausing and scrubbing

```
func handleProgress (_ gr : UIPanGestureRecognizer) {
  let s = gr.location(in: progress)
  let f = min(s.x / progress.bounds.size.width, 1.0)
  let fraction = max(0.0, f)
  animator.fractionComplete = fraction
  progressAnimator.fractionComplete = fraction
}
```

UIViewPropertyAnimator

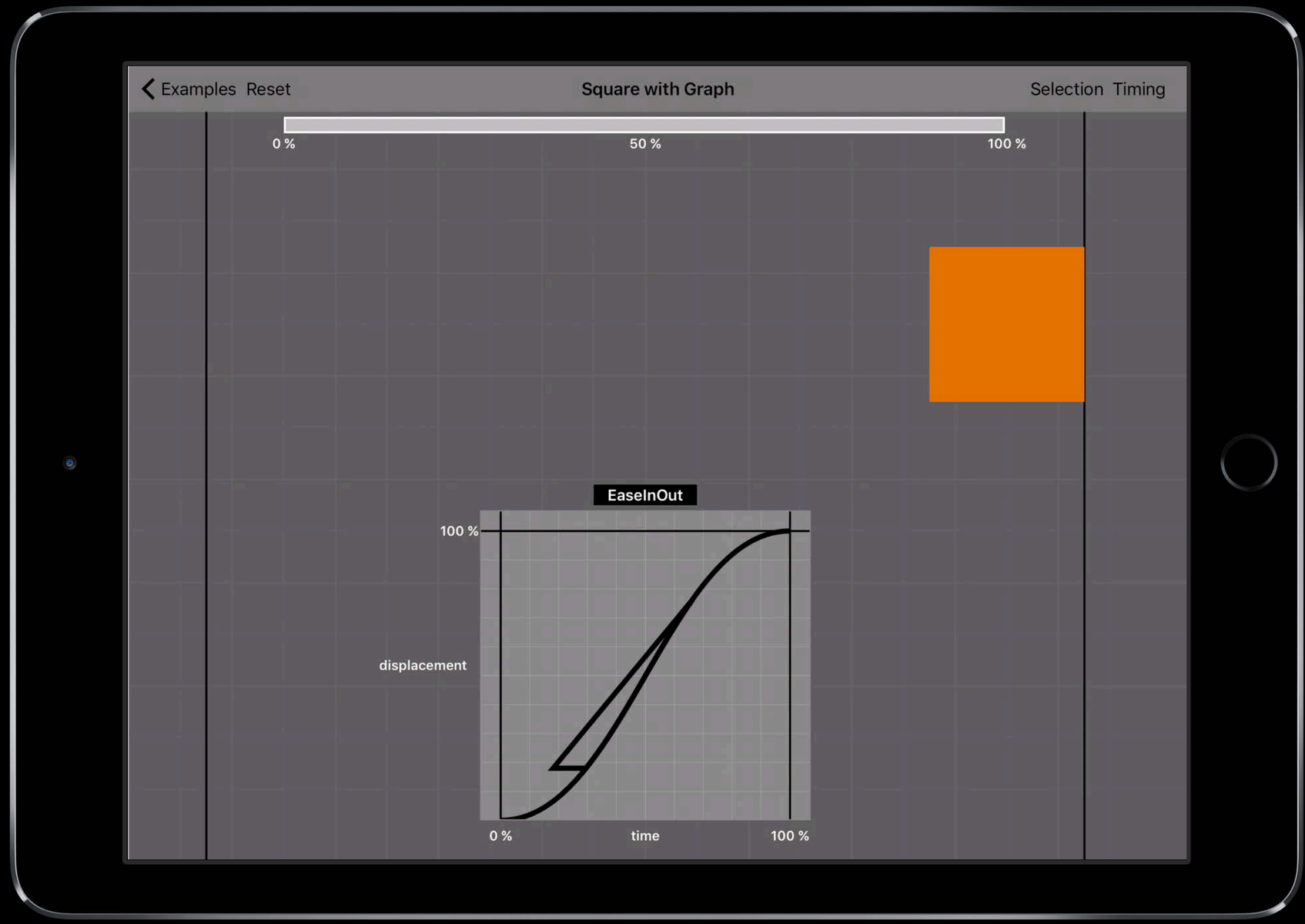
Pausing and scrubbing

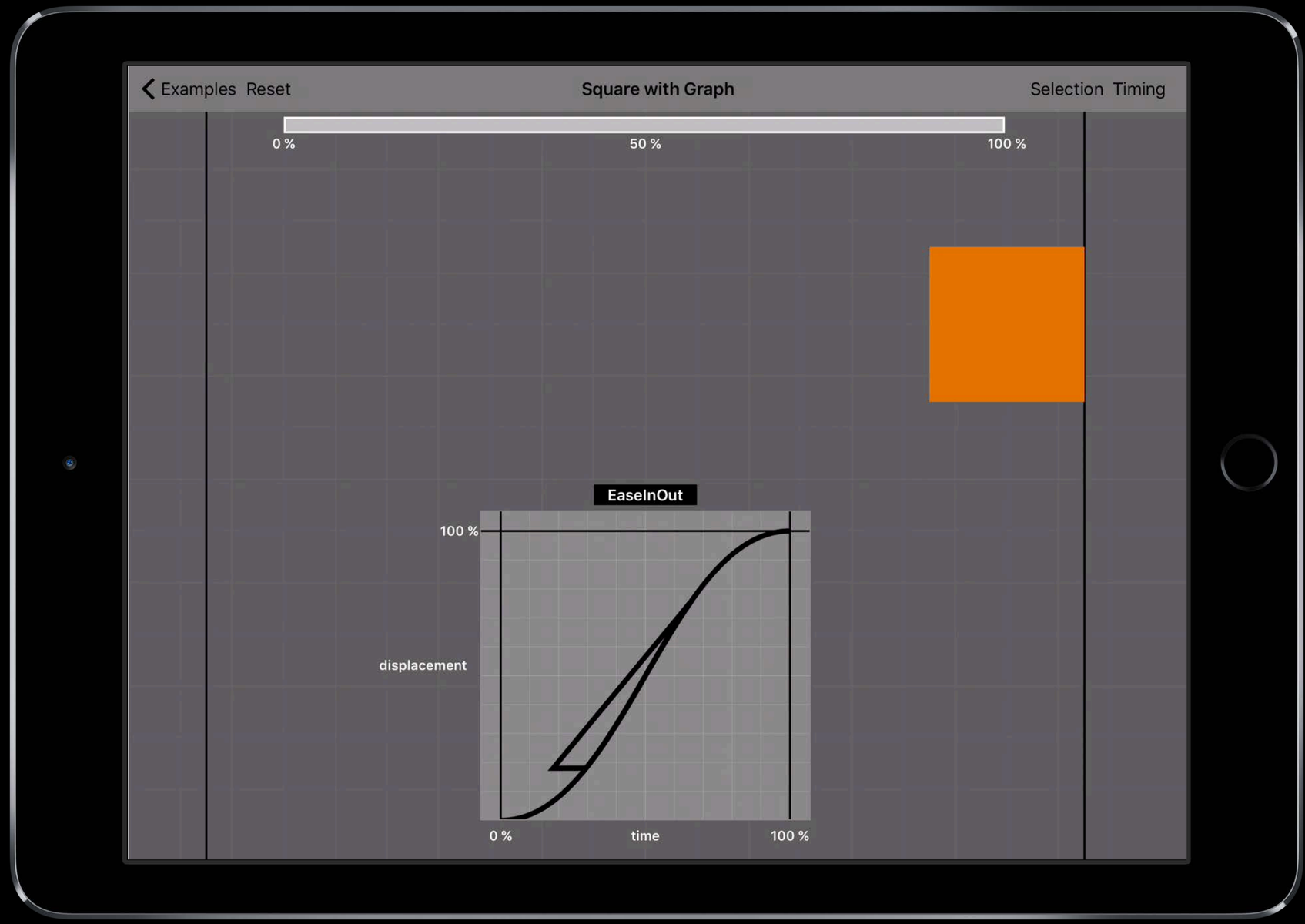
```
func handleProgress (_ gr : UIPanGestureRecognizer) {
  let s = gr.location(in: progress)
  let f = min(s.x / progress.bounds.size.width, 1.0)
  let fraction = max(0.0, f)
  animator.fractionComplete = fraction
  progressAnimator.fractionComplete = fraction
}
```

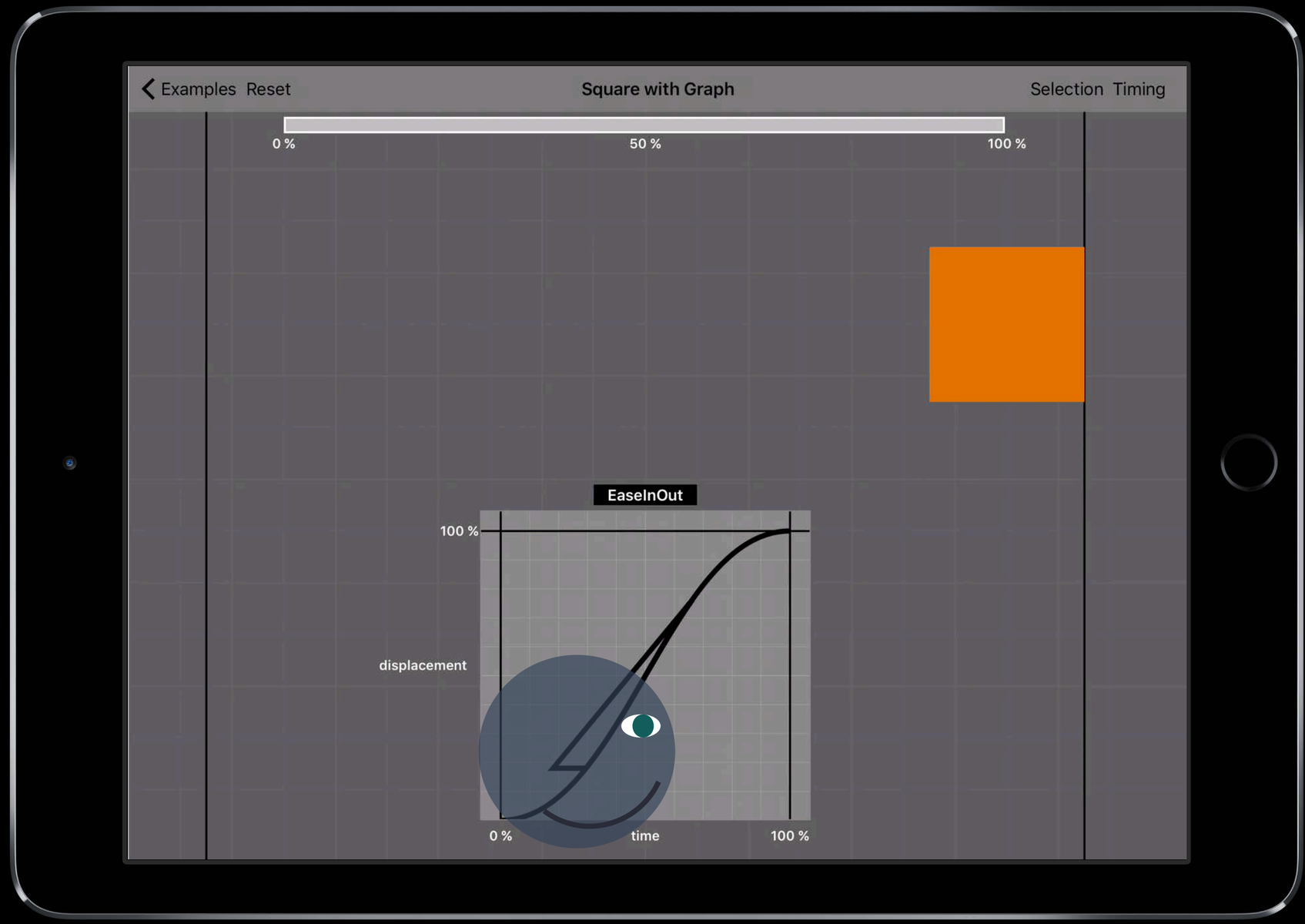
UIViewPropertyAnimator

Pausing and scrubbing

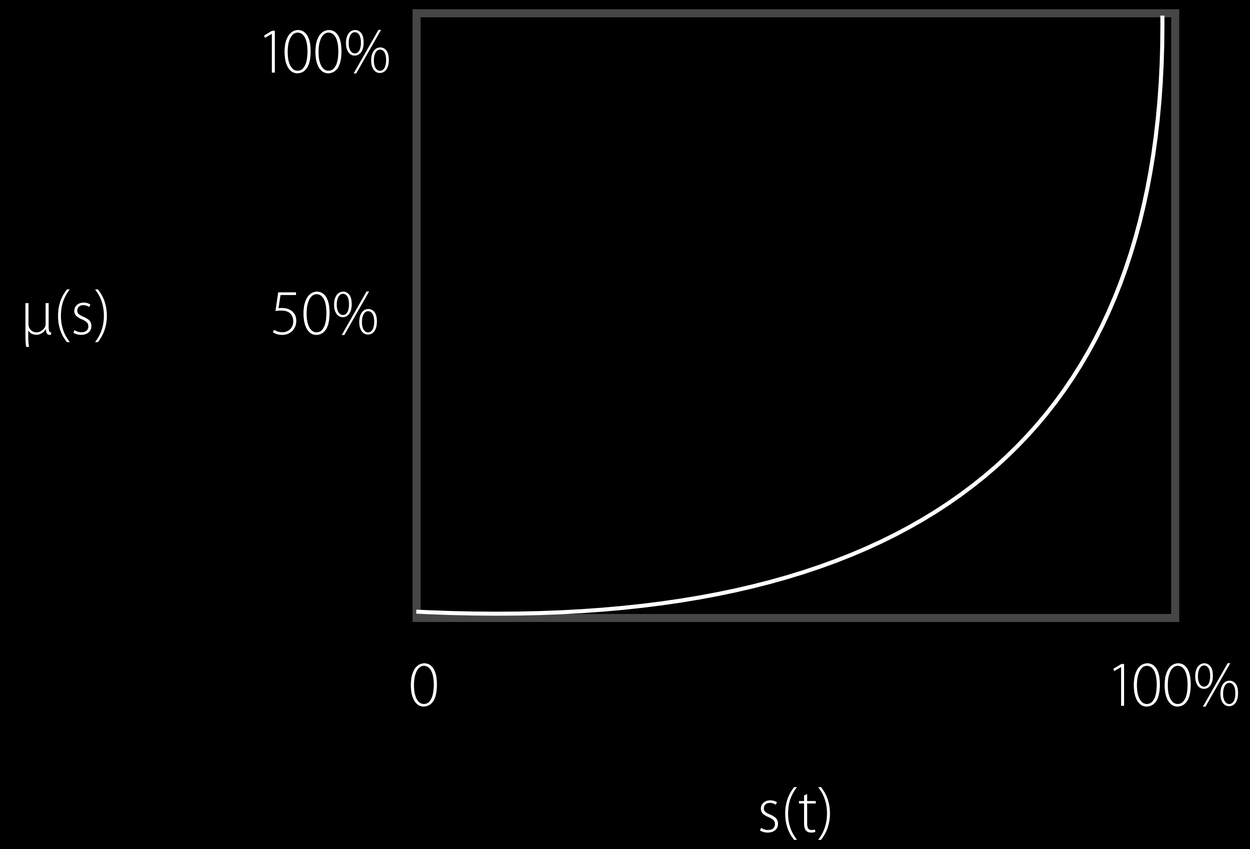
```
func handleProgress (_ gr : UIPanGestureRecognizer) {
  let s = gr.location(in: progress)
  let f = min(s.x / progress.bounds.size.width, 1.0)
  let fraction = max(0.0, f)
    animator.fractionComplete = fraction
    progressAnimator.fractionComplete = fraction
}
```


Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %


Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %


Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %

x 0 200 400 600 800



x

0

200

400

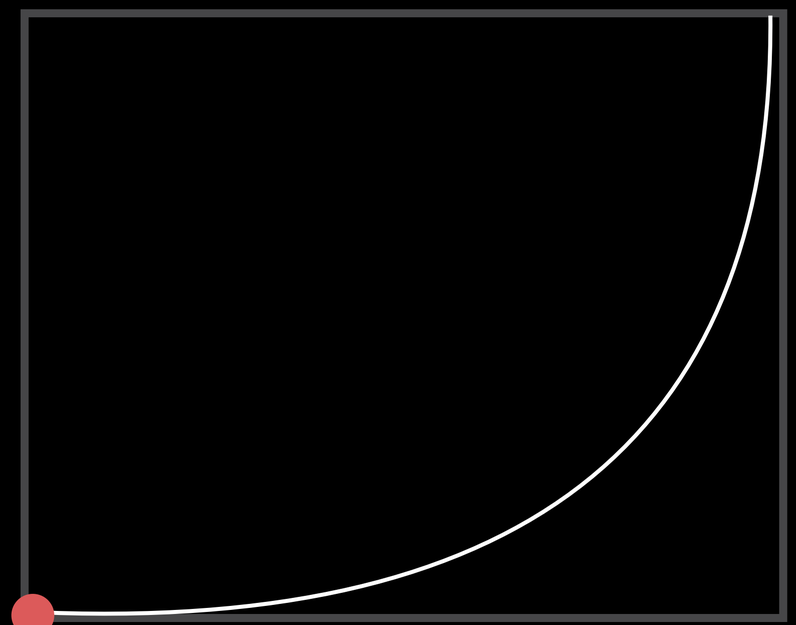
600

800

100%

μ(s)

50%



0

100%

s(t)

x

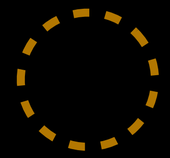
0

200

400

600

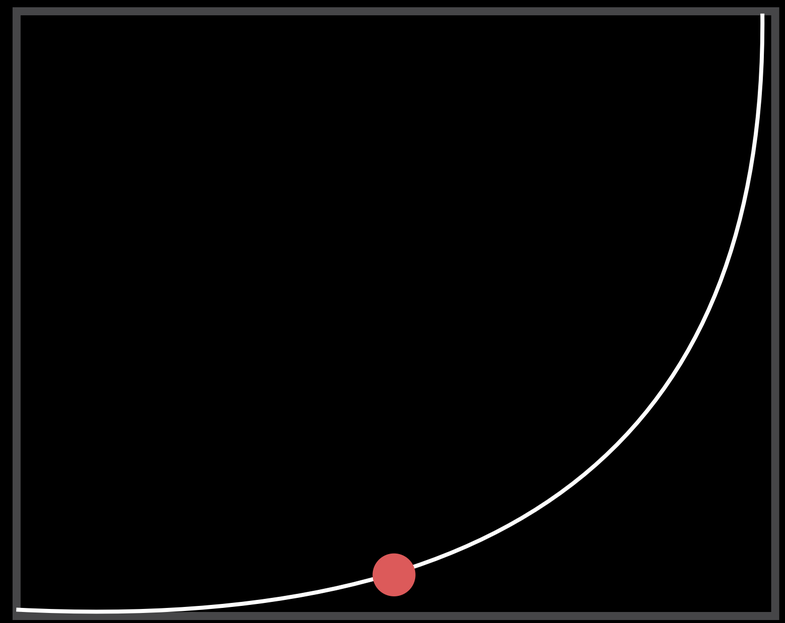
800



100%

μ(s)

50%



0

100%

s(t)

x

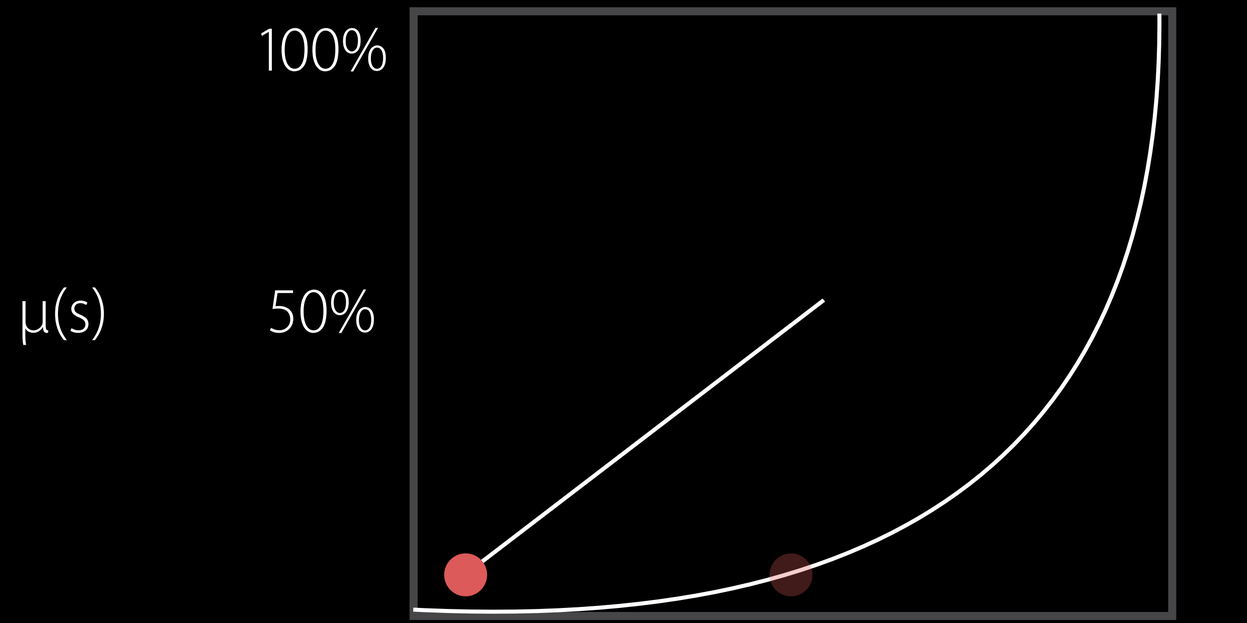
0

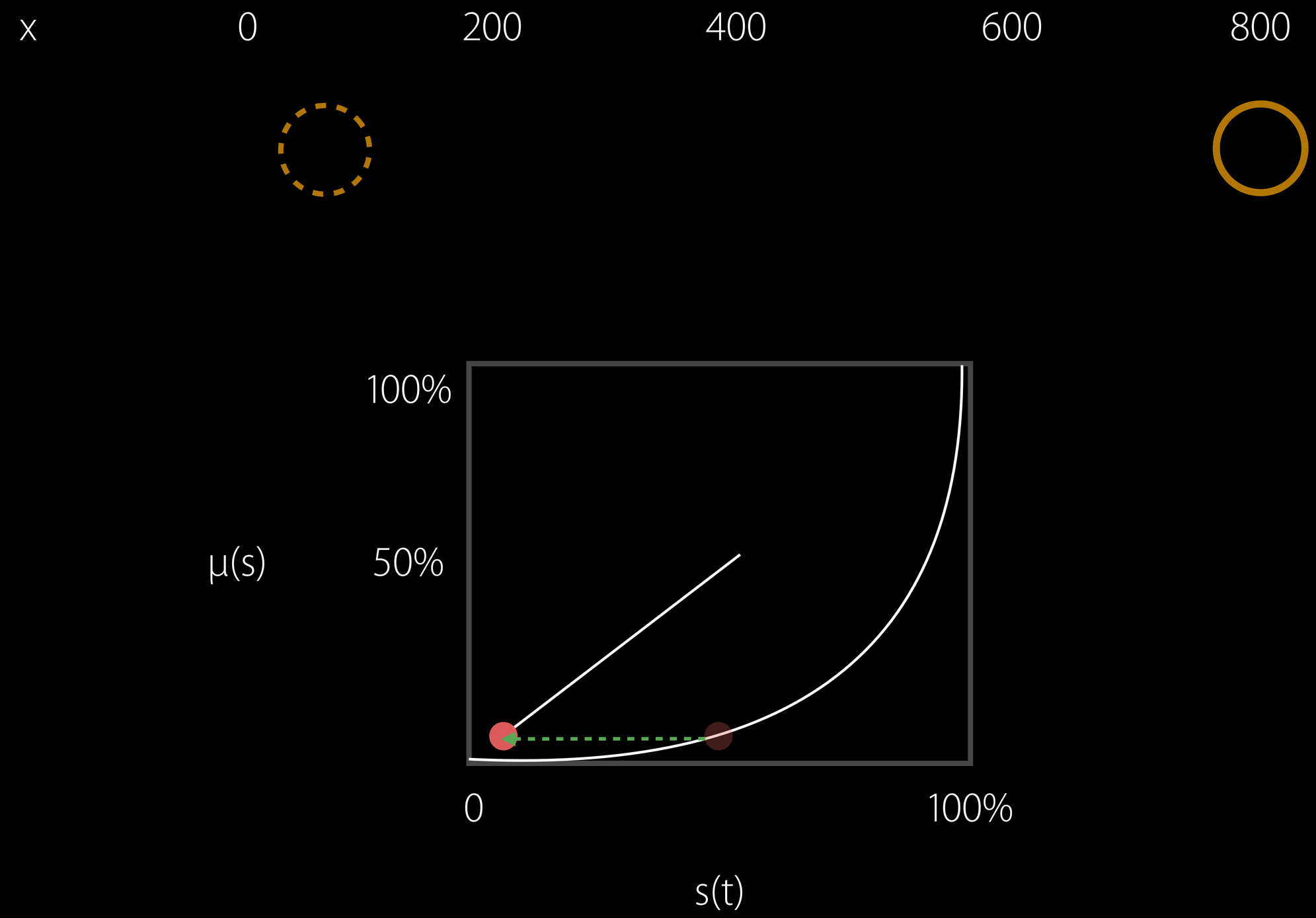
200

400

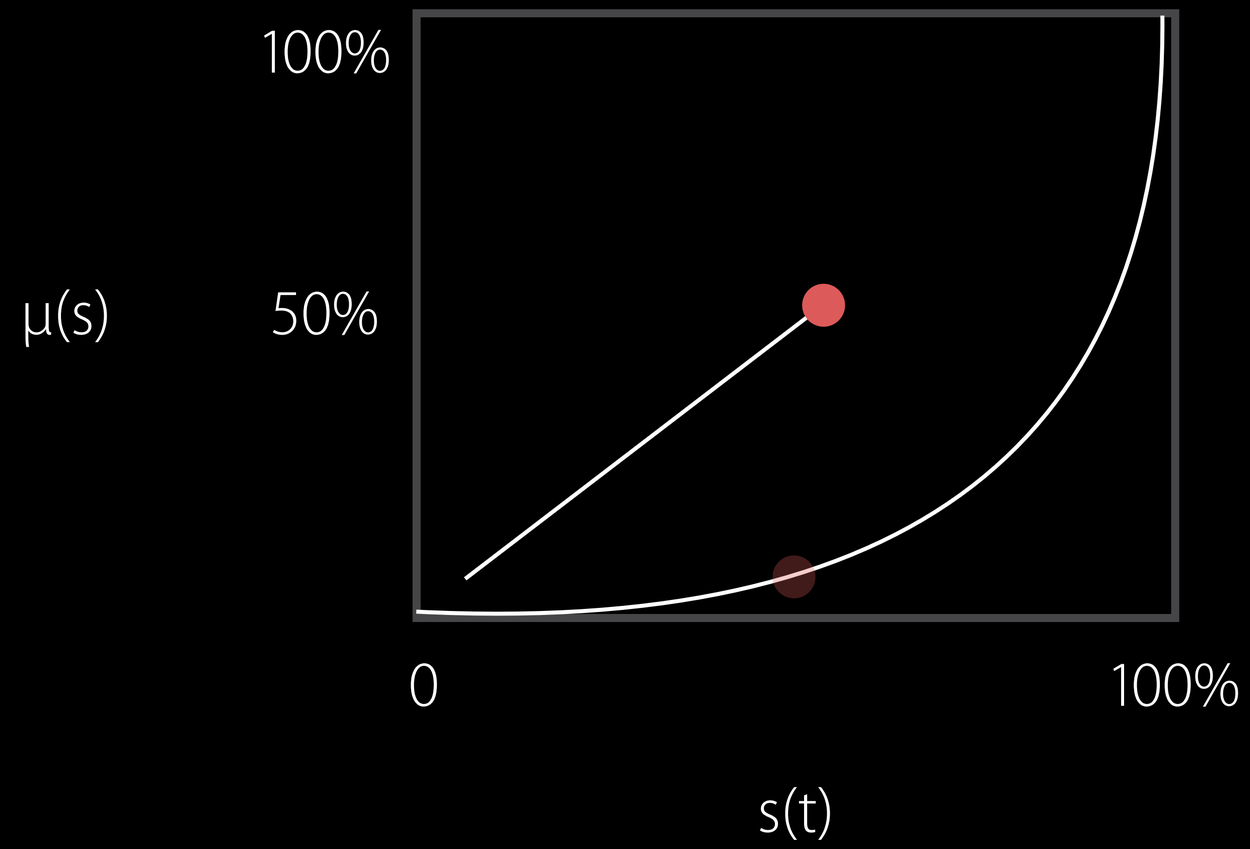
600

800

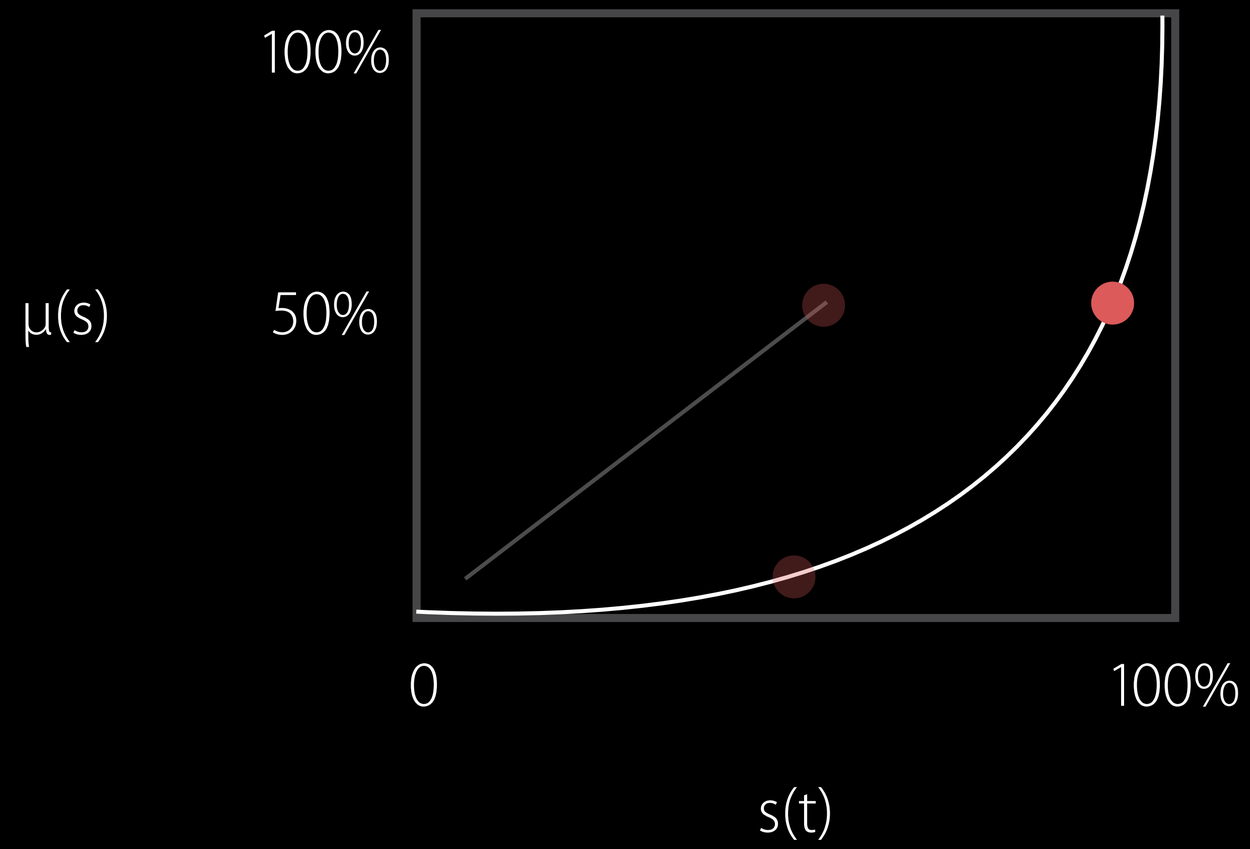



x
0
200
400
600
800
100%
μ(s)
50%
0
100%
s(t)

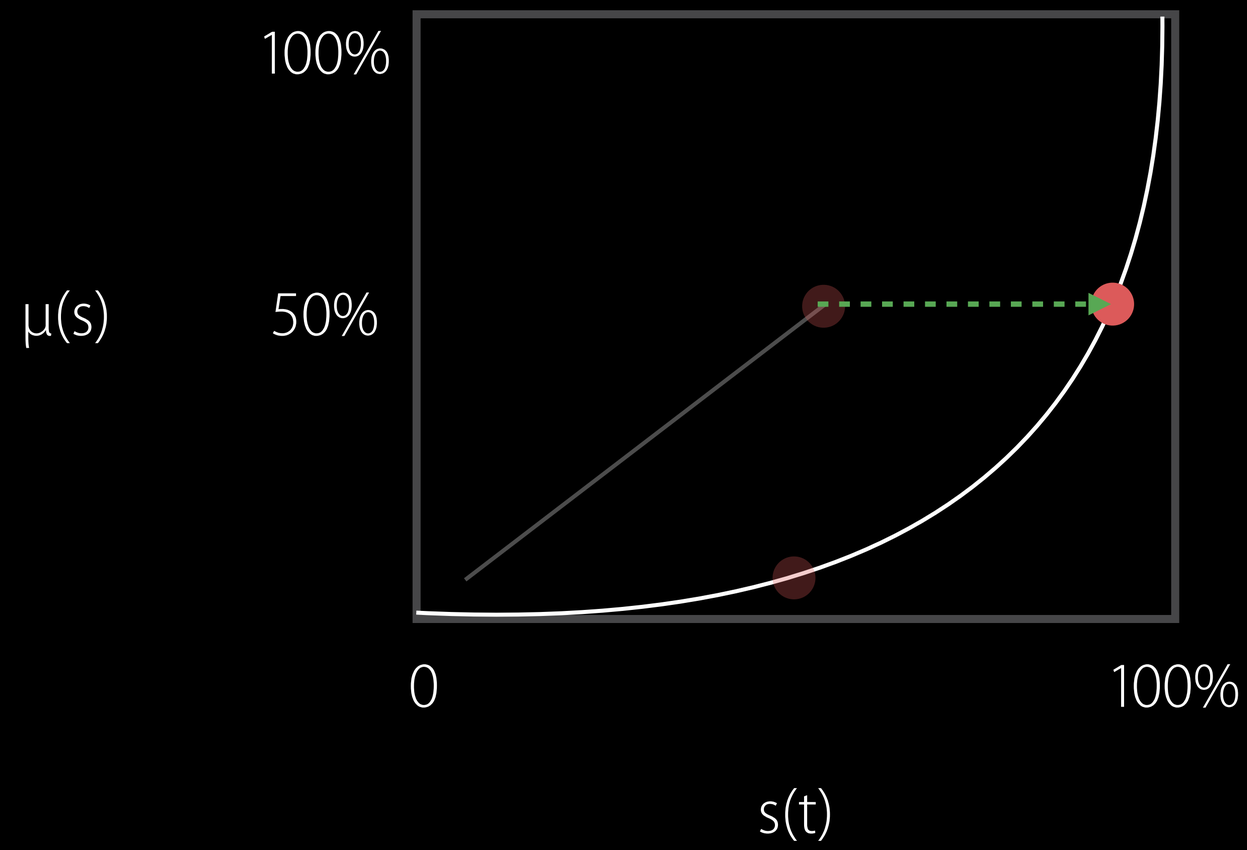
x 0 200 400 600 800



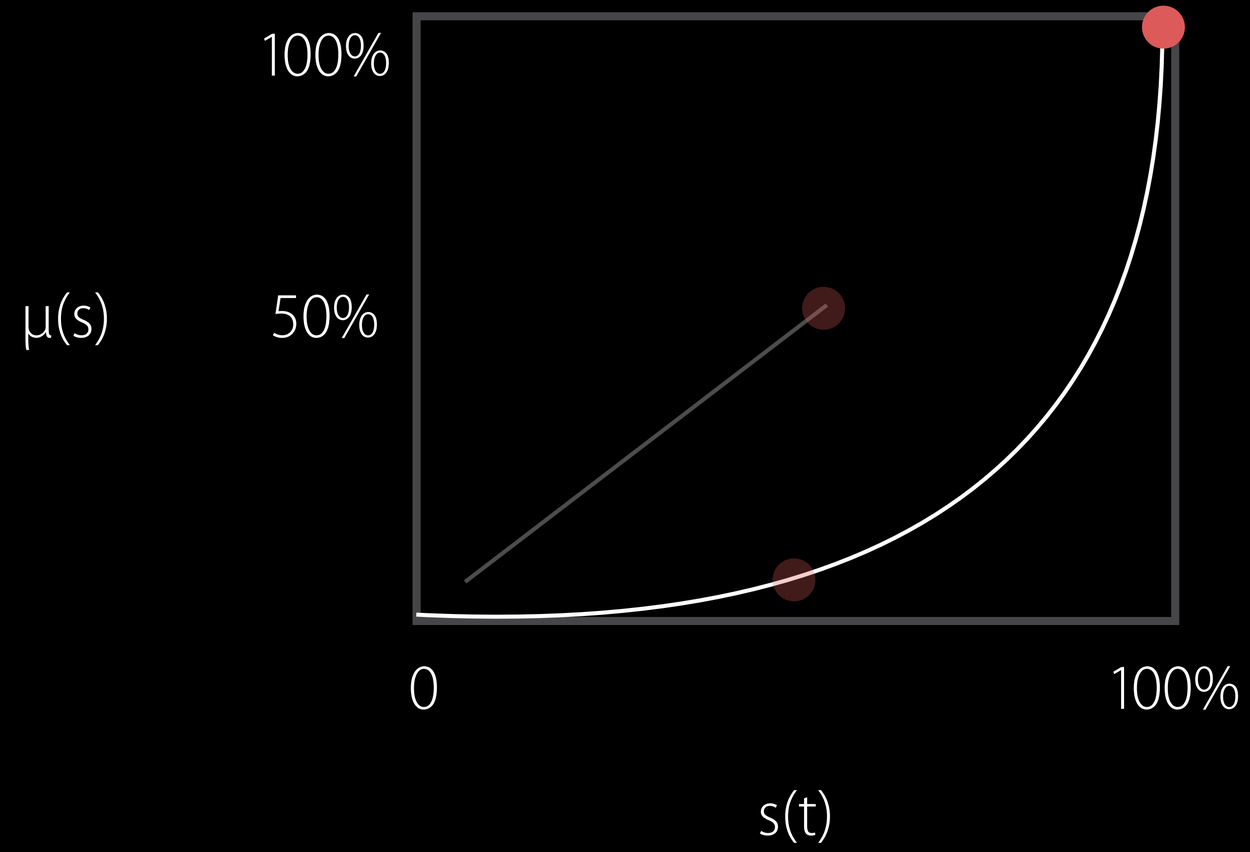
x 0 200 400 600 800

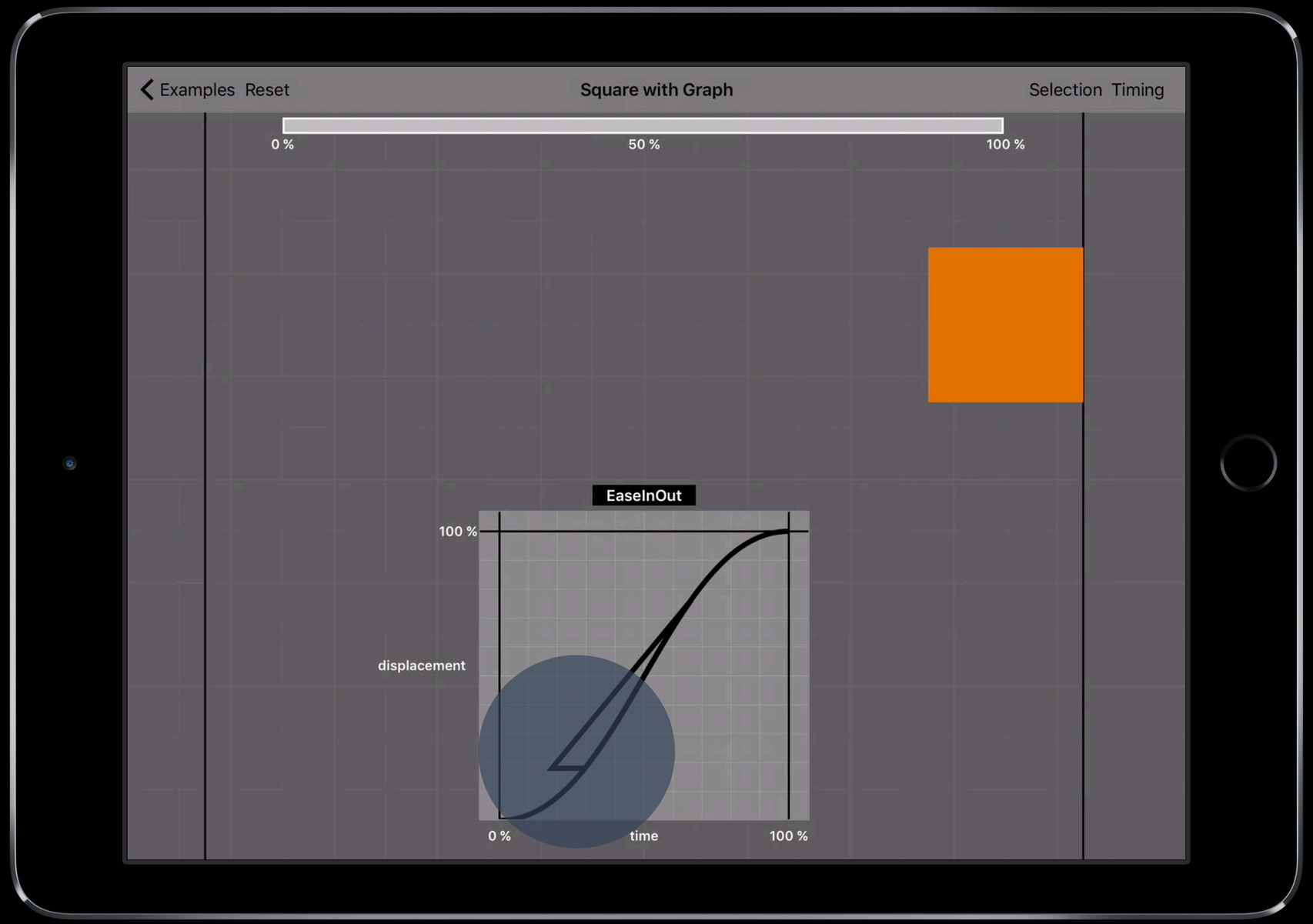


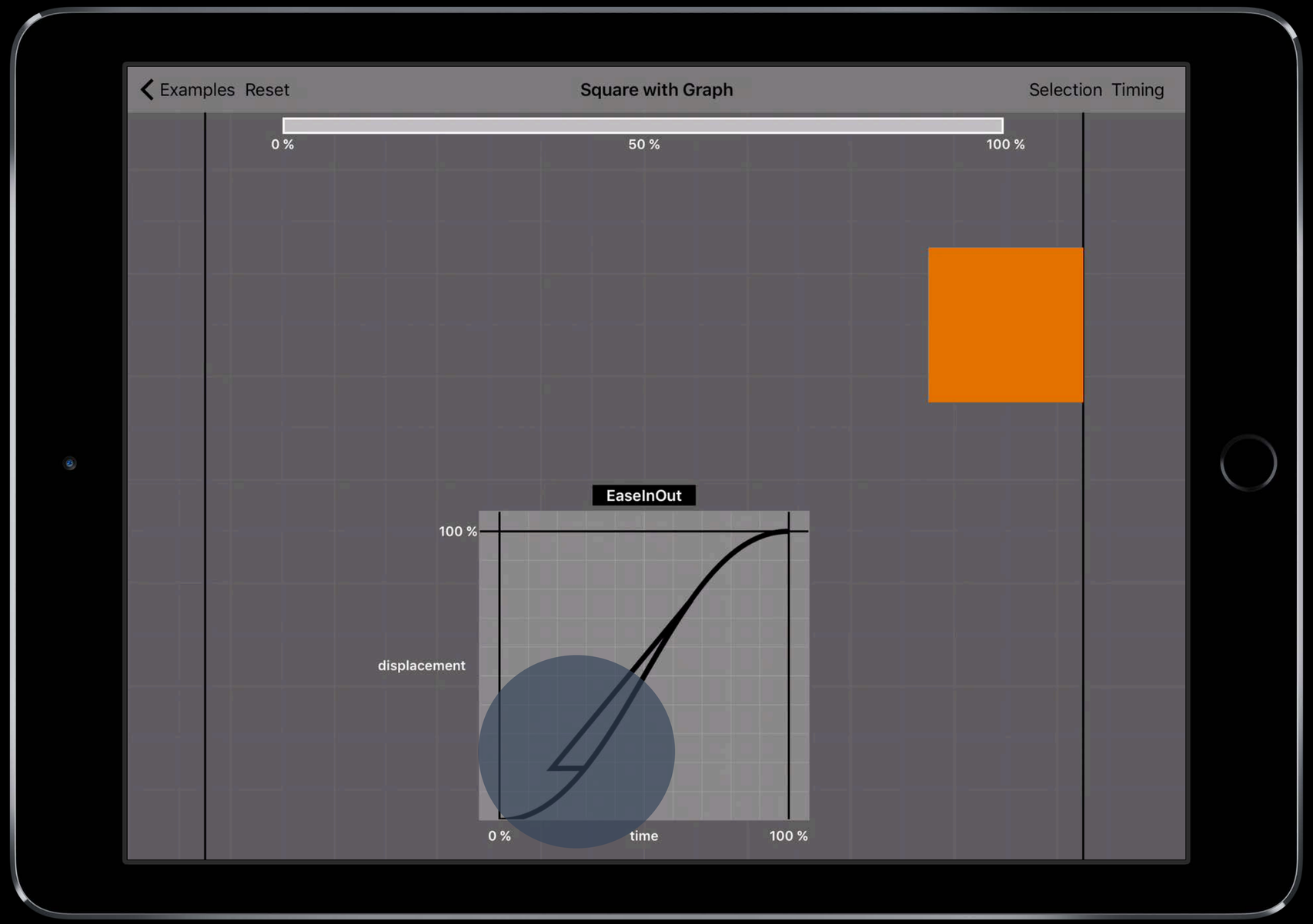
x 0 200 400 600 800



x 0 200 400 600 800



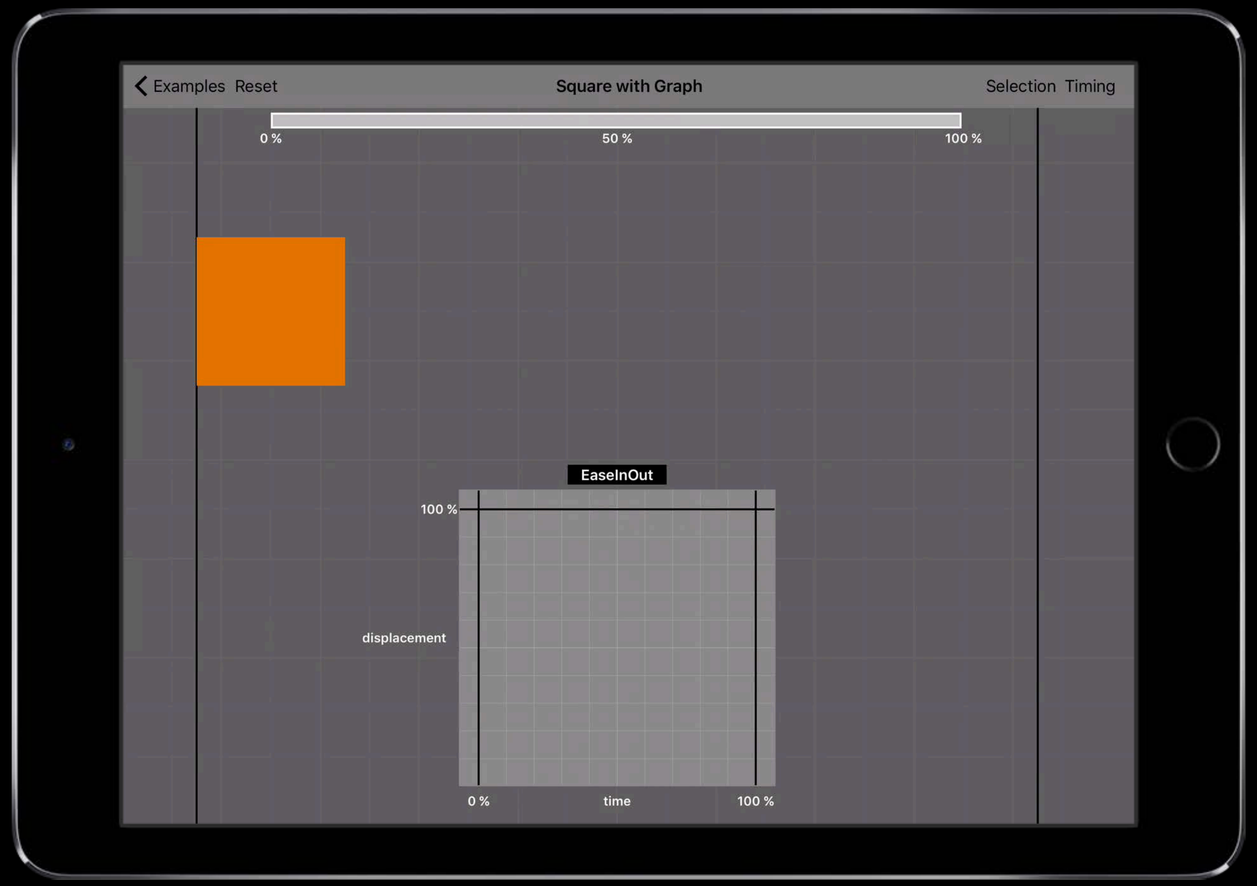

Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %


Examples
Reset
Square with Graph
Selection
Timing
0 %
50 %
100 %
EaseInOut
100 %
displacement
0 %
time
100 %

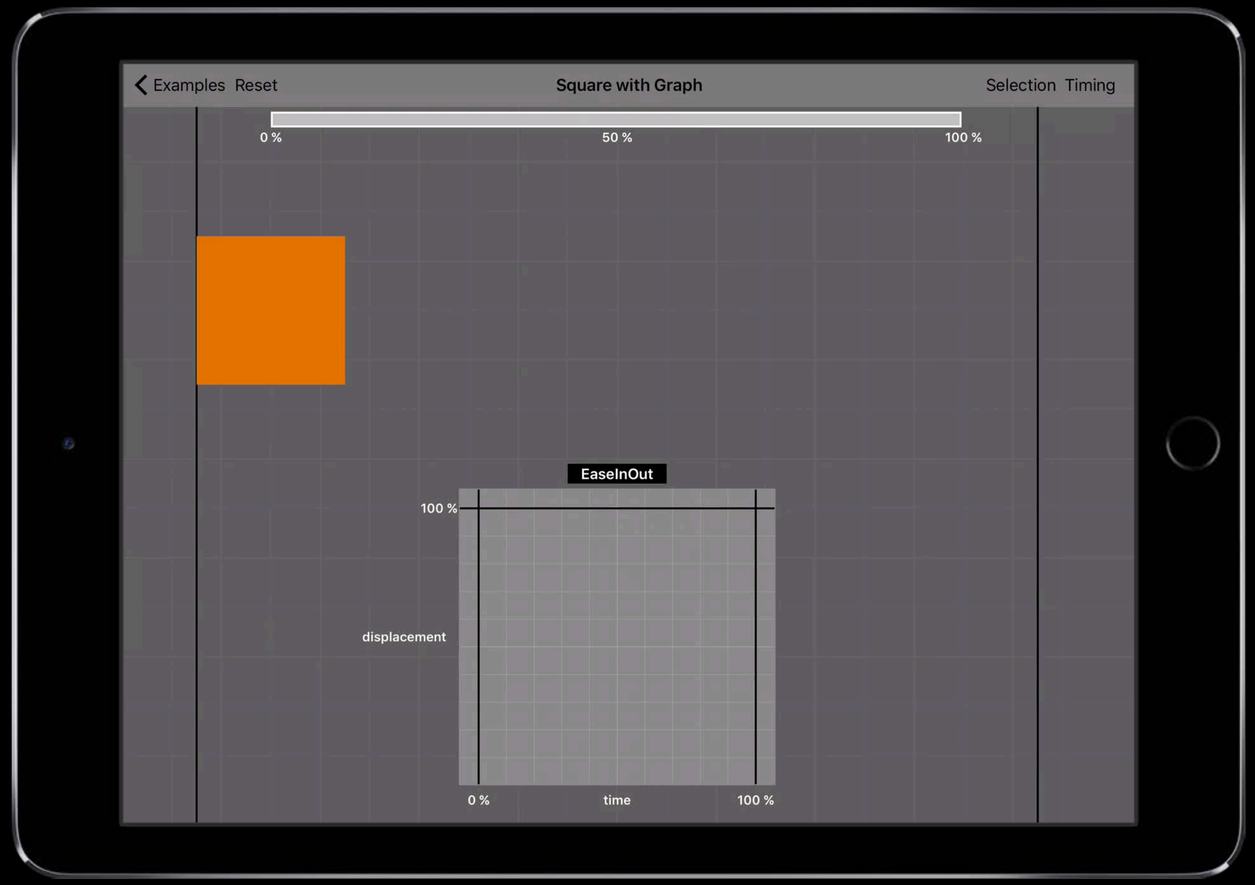
UIViewPropertyAnimator

Reversing

Pause and Reverse

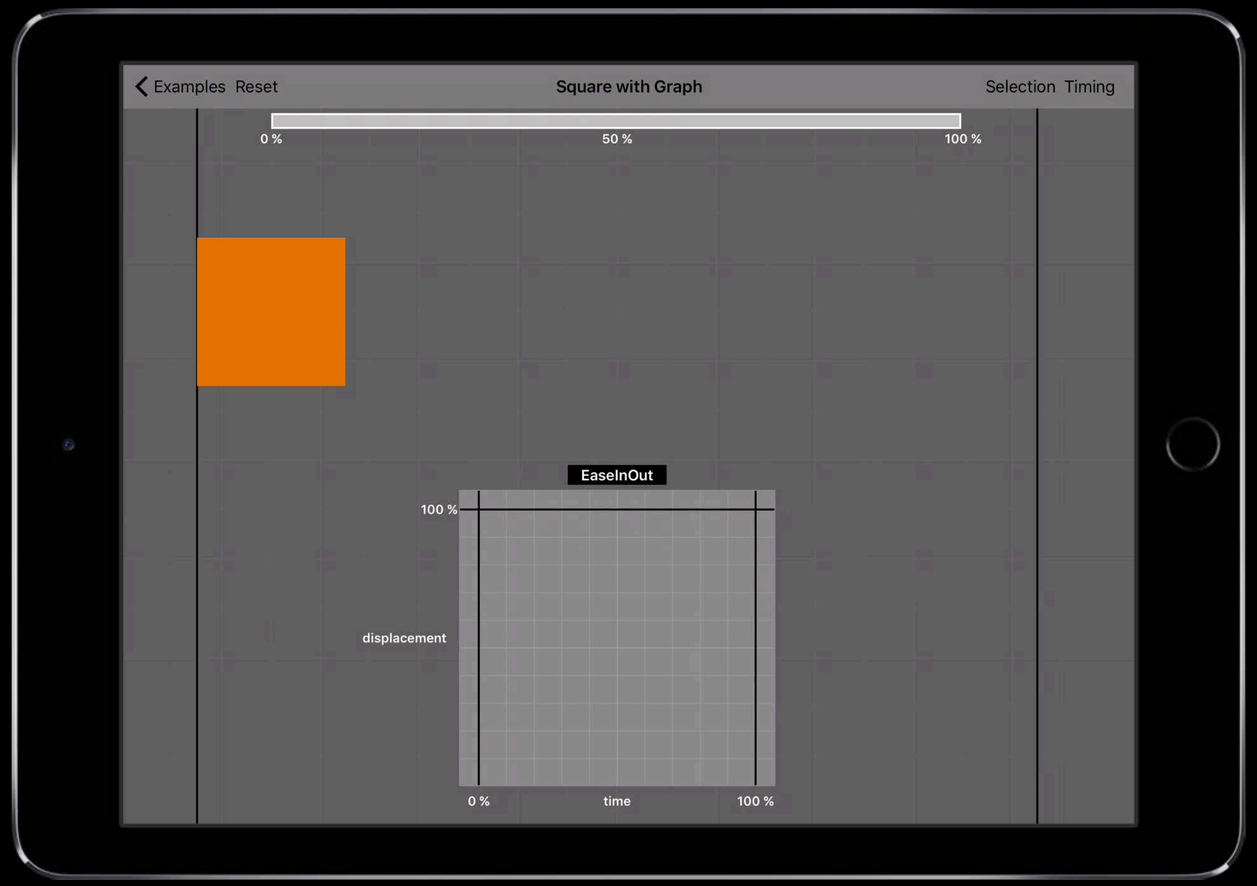


Pause and Reverse

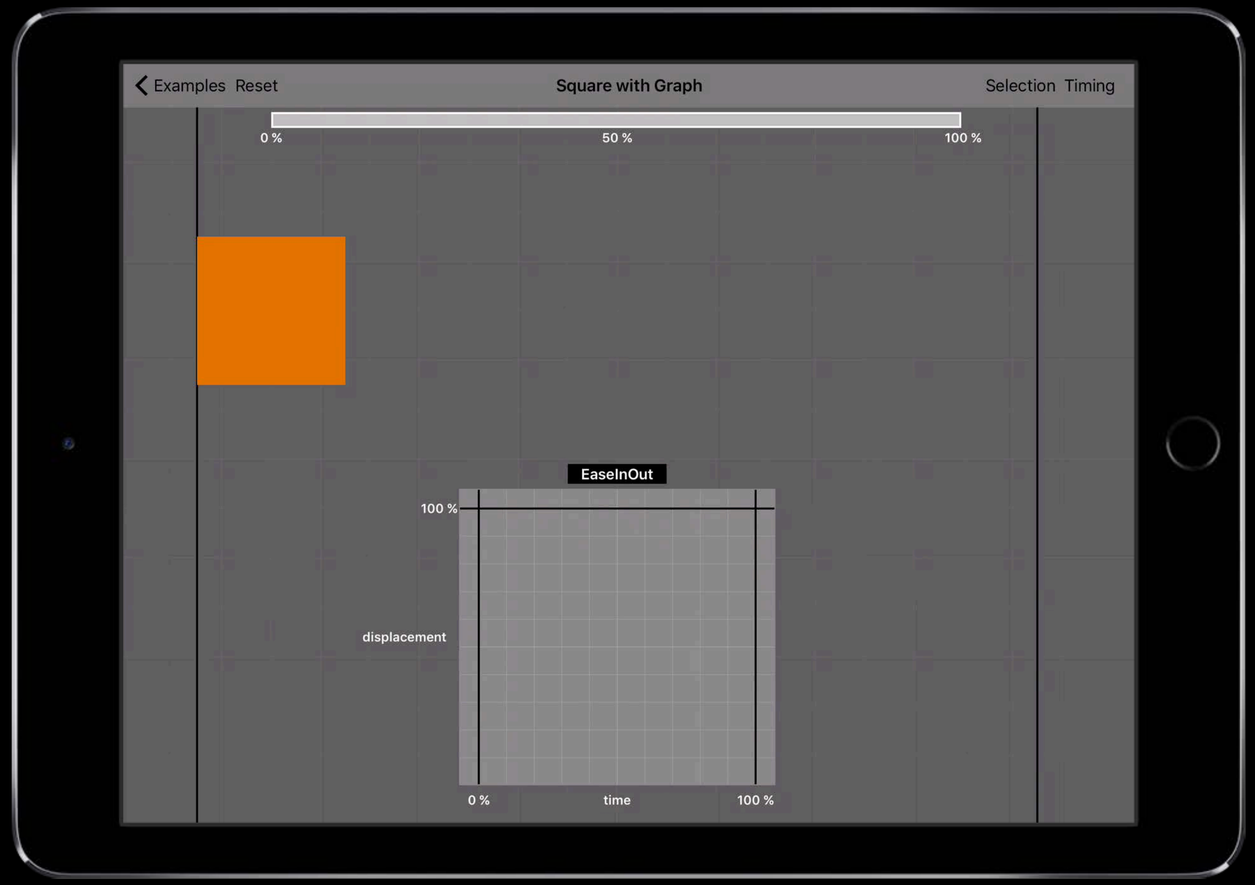



```
animator.pauseAnimation()
animator.isReversed = true
animator.startAnimation()
```

Running Reverse

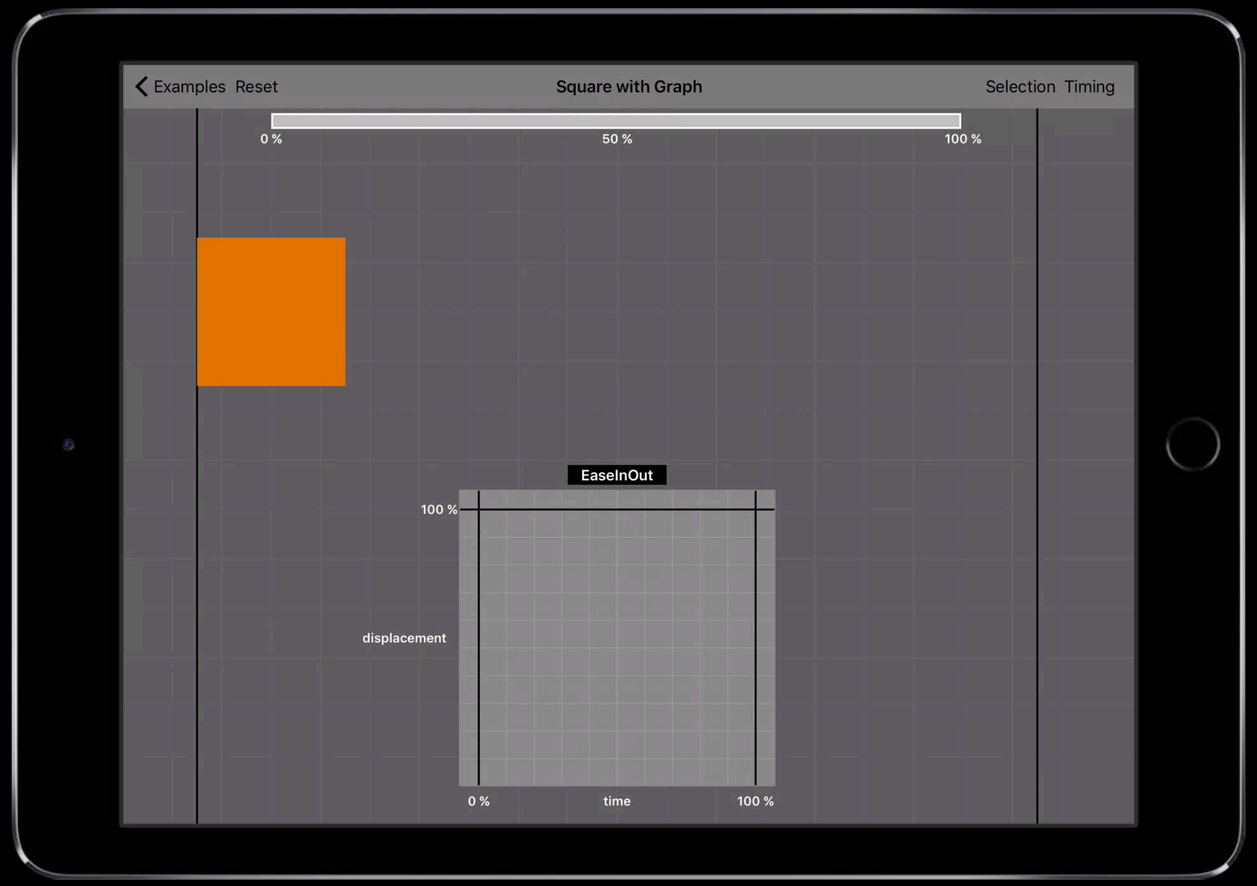


Running Reverse

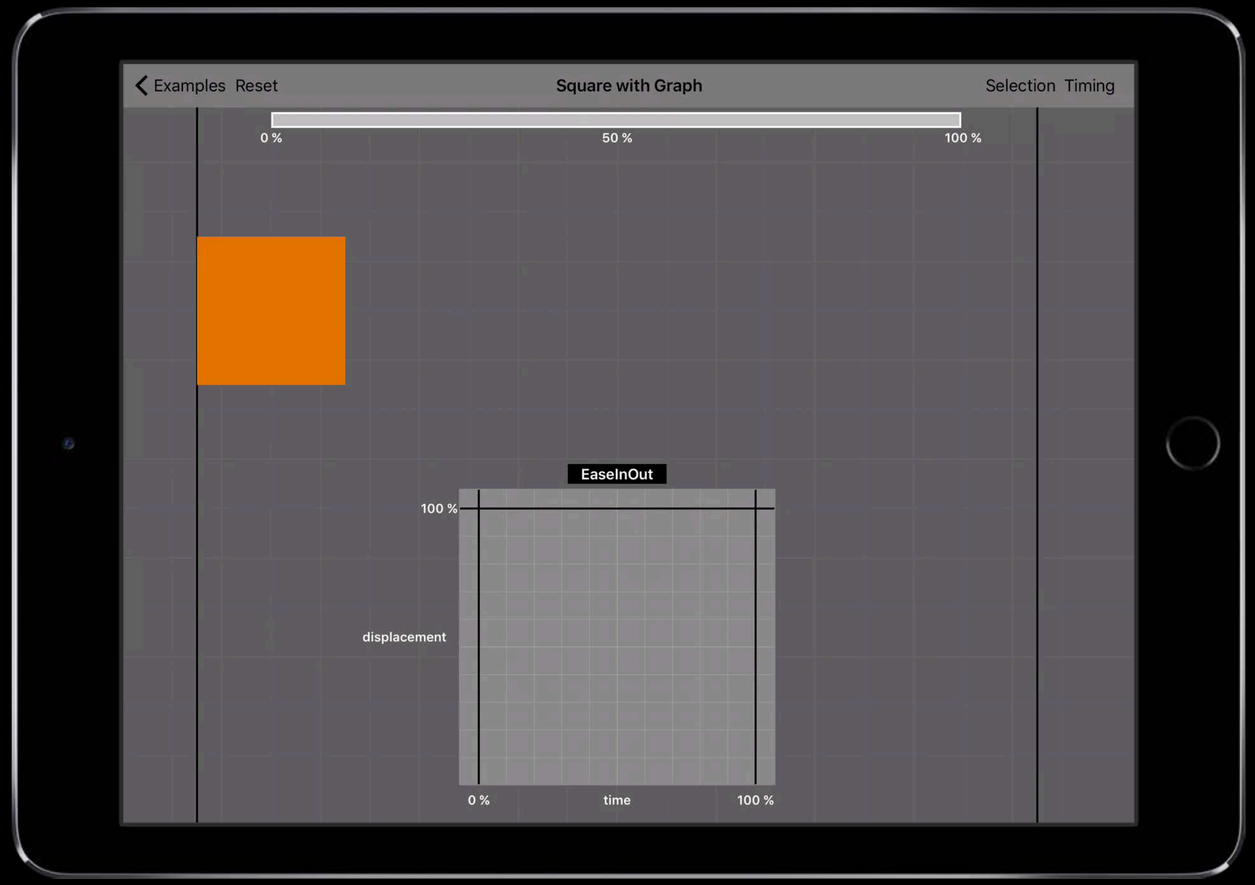



```
animator.isReversed = true
```

Animate Back



Animate Back




```
animator.addAnimations {
  view.center.x = 150.0
  view.transform = CGAffineTransform.identity
}
```

UIViewPropertyAnimator

Timing providers

NEW

UICubicTimingParameters

UIViewPropertyAnimator

Timing providers

NEW

```
UICubicTimingParameters()

UICubicTimingParameters(animationCurve: .linear)

UICubicTimingParameters(controlPoint1: CGPoint(x:0.0, y:1.0),
                        controlPoint2: CGPoint(x:1.0,y:0.0))
```

UIViewPropertyAnimator

Timing providers

NEW

```
UICubicTimingParameters()

UICubicTimingParameters(animationCurve: .linear)

UICubicTimingParameters(controlPoint1: CGPoint(x:0.0, y:1.0),
                        controlPoint2: CGPoint(x:1.0,y:0.0))
```

UIViewPropertyAnimator

Timing providers

NEW

```
UICubicTimingParameters()

UICubicTimingParameters(animationCurve: .linear)

UICubicTimingParameters(controlPoint1: CGPoint(x:0.0, y:1.0),
                        controlPoint2: CGPoint(x:1.0,y:0.0))
```

UIViewPropertyAnimator

Timing providers

NEW

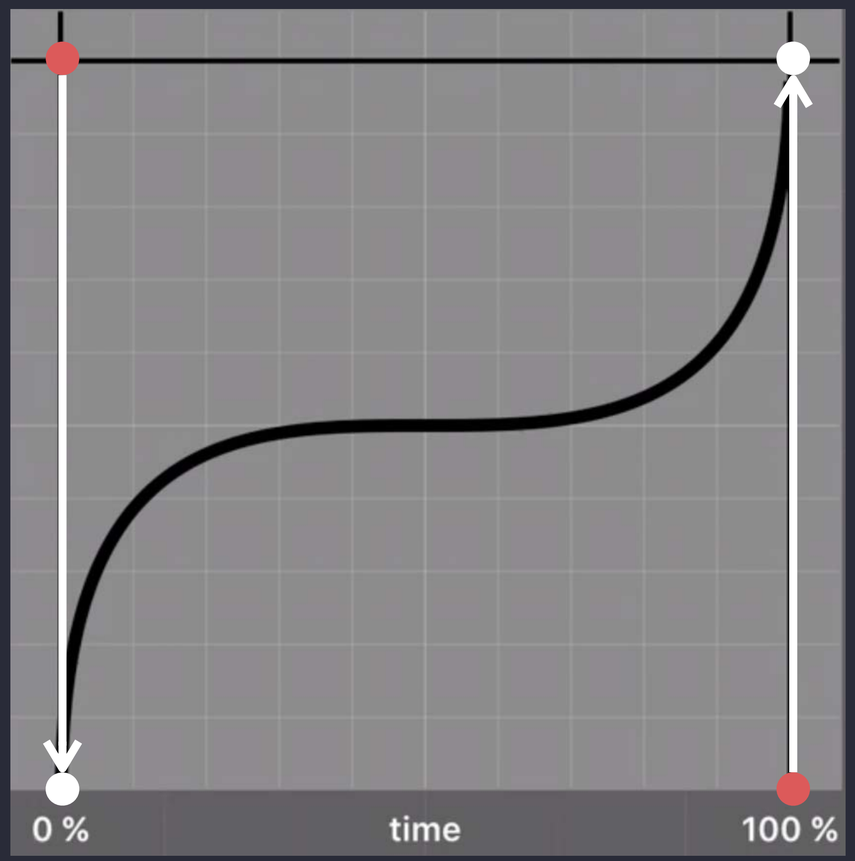
```
UICubicTimingParameters()

UICubicTimingParameters(animationCurve: .linear)

UICubicTimingParameters(controlPoint1: CGPoint(x:0.0, y:1.0),
                        controlPoint2: CGPoint(x:1.0,y:0.0))
```

UIViewPropertyAnimator

Timing providers

NEW

```
UICubicTimingParameters(controlPoint1: CGPoint(x:0.0, y:1.0),
                        controlPoint2: CGPoint(x:1.0,y:0.0))
```



NEW

UISpringTimingParameters

UIViewPropertyAnimator

Timing providers

NEW

```
UISpringTimingParameters()

UISpringTimingParameters(dampingRatio: 0.8,
                         initialVelocity: CGVector(dx:1.0, dy: 0.0))

UISpringTimingParameters(mass: CGFloat, stiffness: CGFloat,
                         damping: CGFloat, initialVelocity velocity: CGVector)
```

UIViewPropertyAnimator

Timing providers

NEW

```
UISpringTimingParameters()

UISpringTimingParameters(dampingRatio: 0.8,
                         initialVelocity: CGVector(dx:1.0, dy: 0.0))

UISpringTimingParameters(mass: CGFloat, stiffness: CGFloat,
                         damping: CGFloat, initialVelocity velocity: CGVector)
```

UIViewPropertyAnimator

Timing providers

NEW

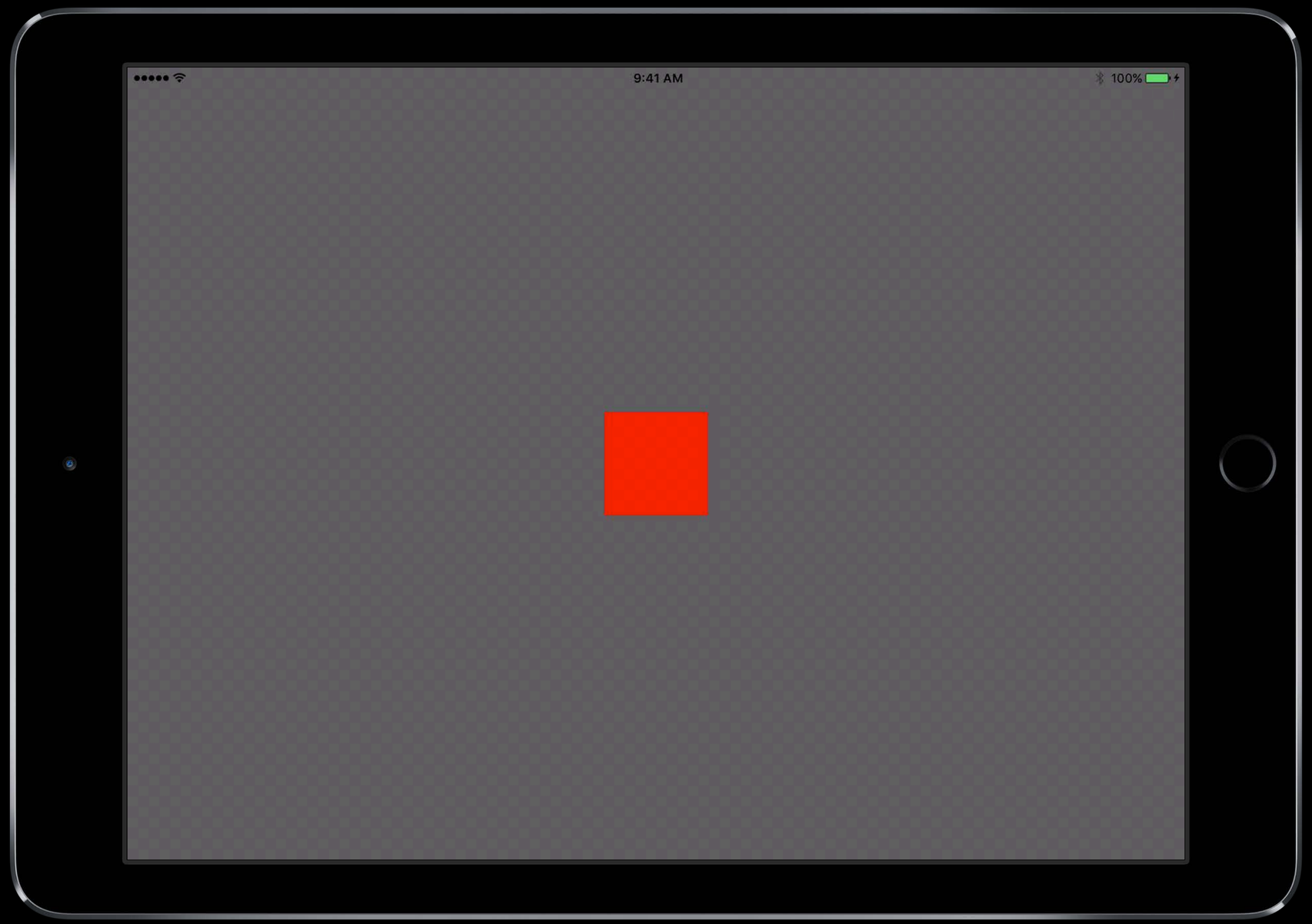
```
UISpringTimingParameters()

UISpringTimingParameters(dampingRatio: 0.8,
                         initialVelocity: CGVector(dx:1.0, dy: 0.0))

UISpringTimingParameters(mass: CGFloat, stiffness: CGFloat,
                         damping: CGFloat, initialVelocity velocity: CGVector)
```

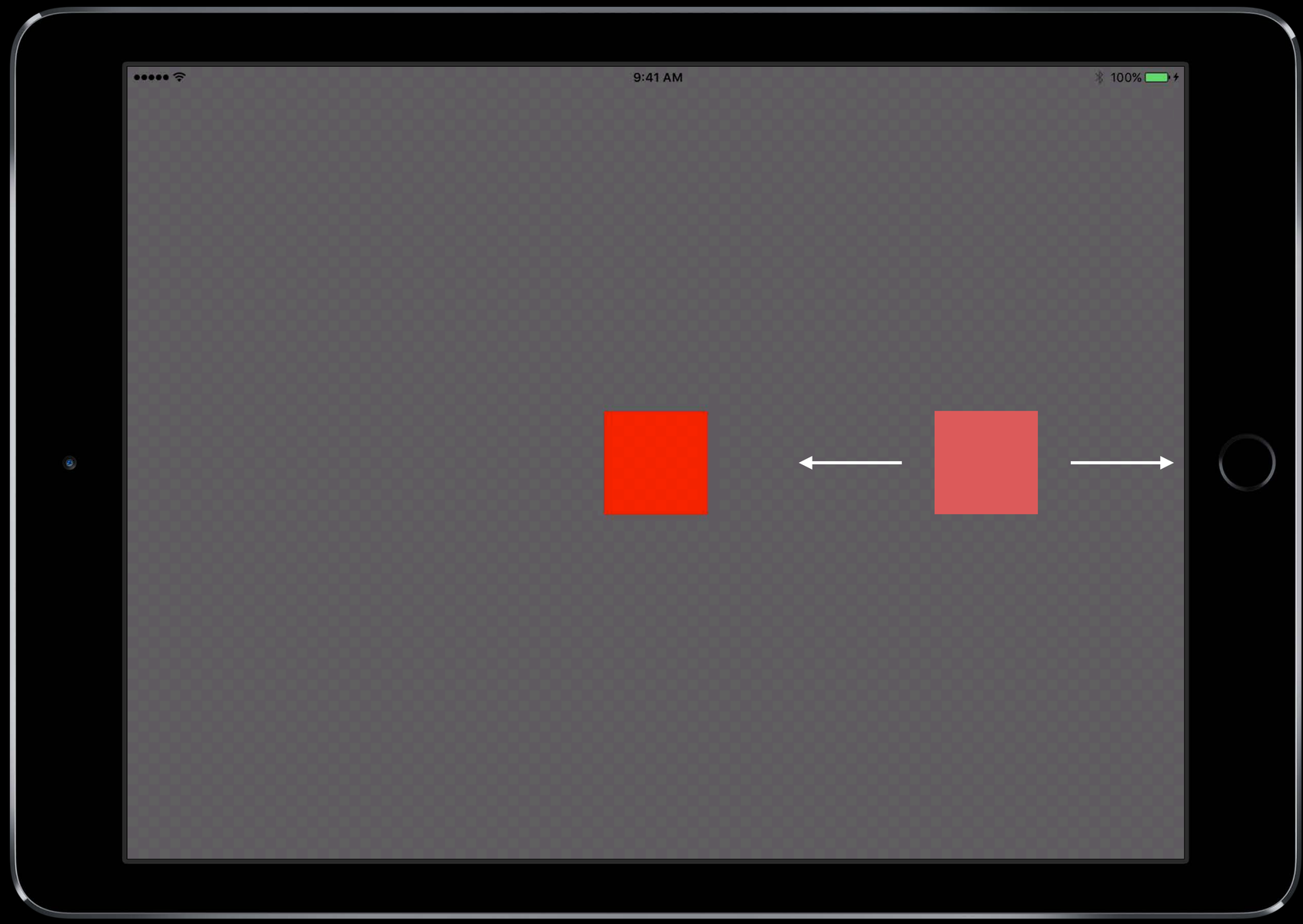
UIViewPropertyAnimator

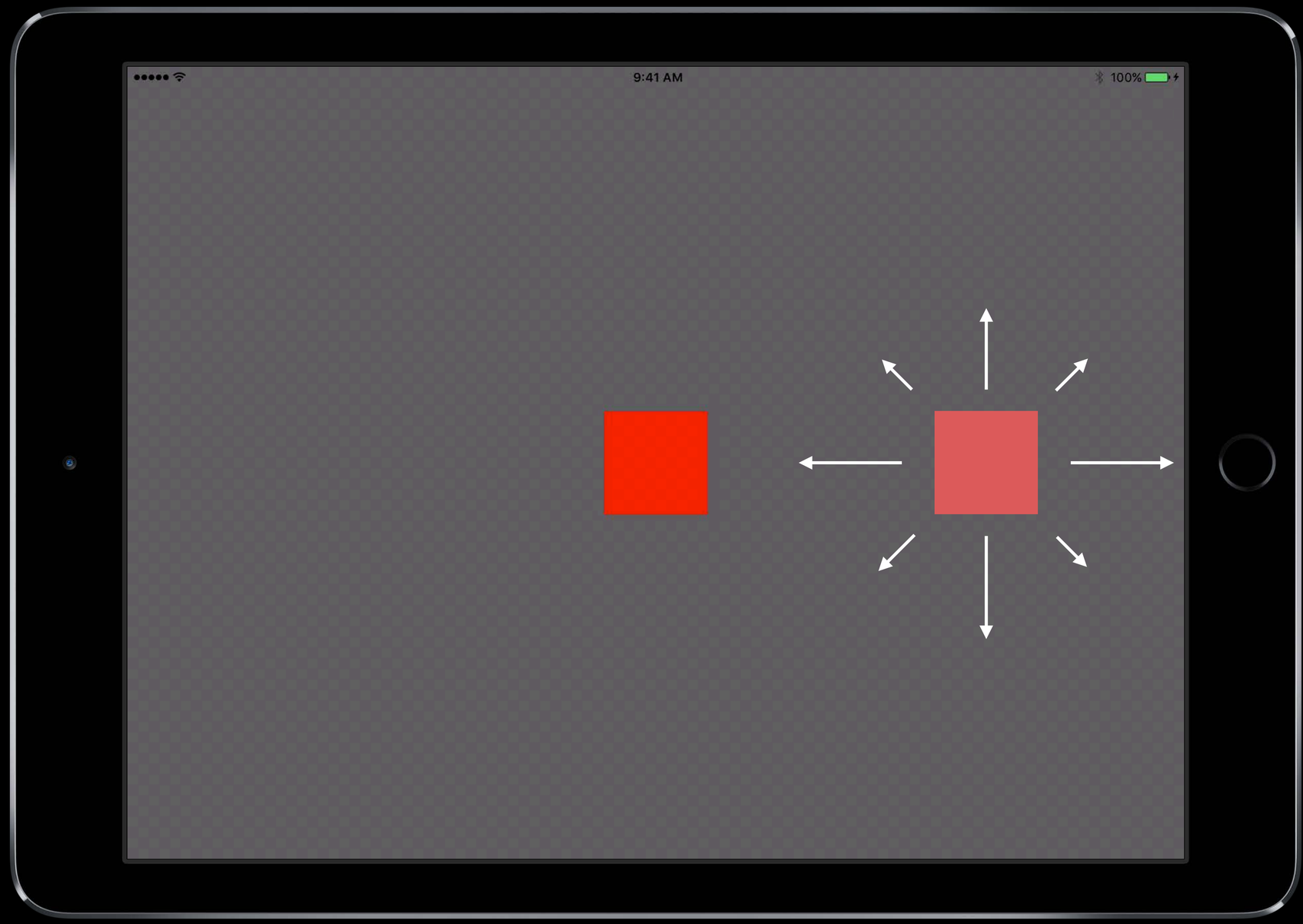
Timing providers

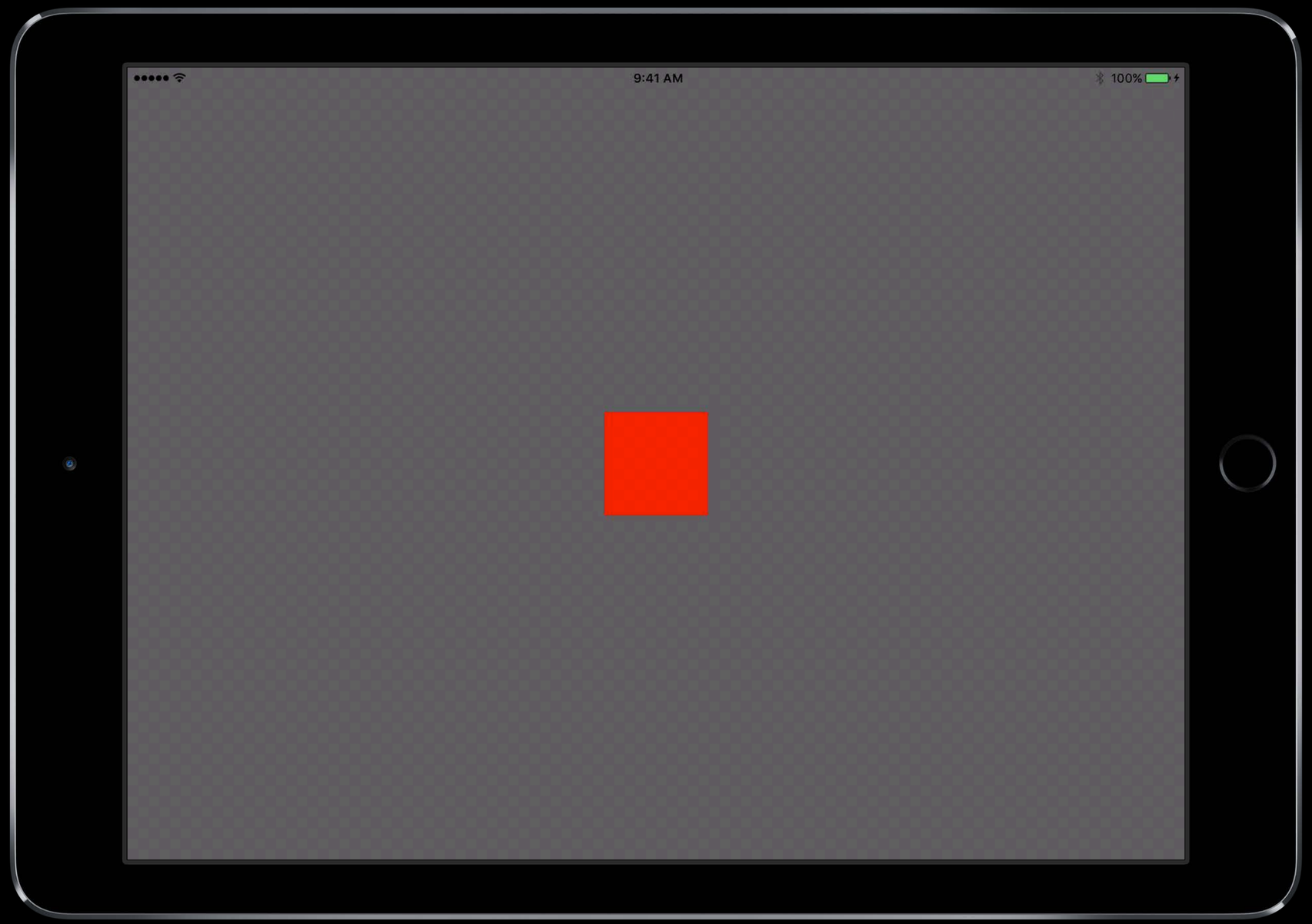
NEW

```
UISpringTimingParameters()

UISpringTimingParameters(dampingRatio: 0.8,
                         initialVelocity: CGVector(dx:1.0, dy: 0.0))

UISpringTimingParameters(mass: CGFloat, stiffness: CGFloat,
                         damping: CGFloat, initialVelocity velocity: CGVector)
```


9:41 AM
100%


9:41 AM
100%


9:41 AM
100%


9:41 AM
100%

Custom View Controller Transitions

Creating interruptible transitions

View Controller Transitions

Custom Transitions Using View Controllers

WWDC 2013

View Controller Transitions

UIViewControllerInteractiveTransitioning

UIViewControllerAnimated Transitioning

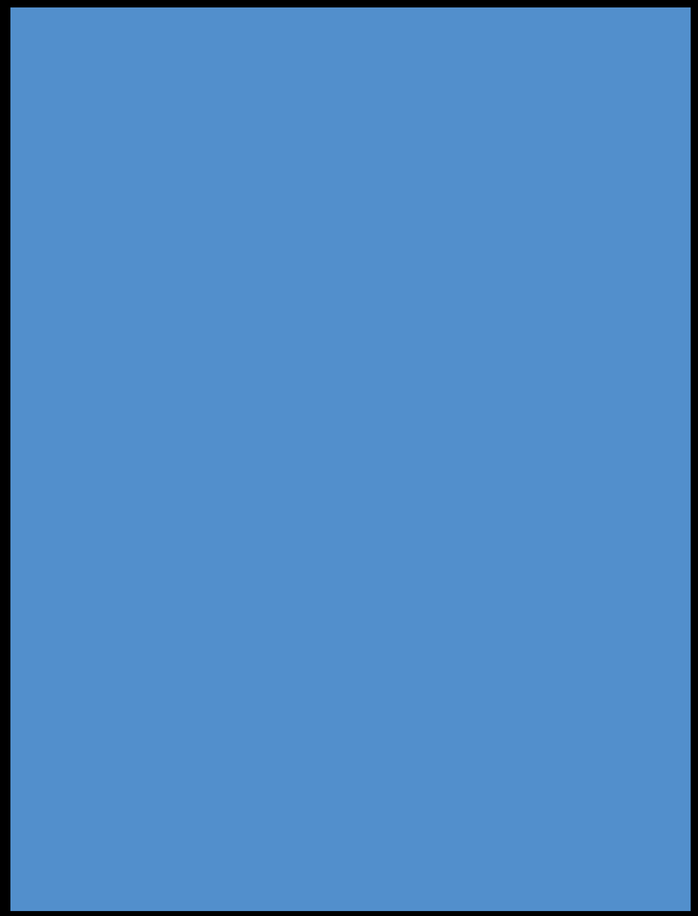
View Controller Transitions

UIViewControllerInteractiveTransitioning

UIViewControllerAnimated Transitioning

UIViewControllerContextTransitioning

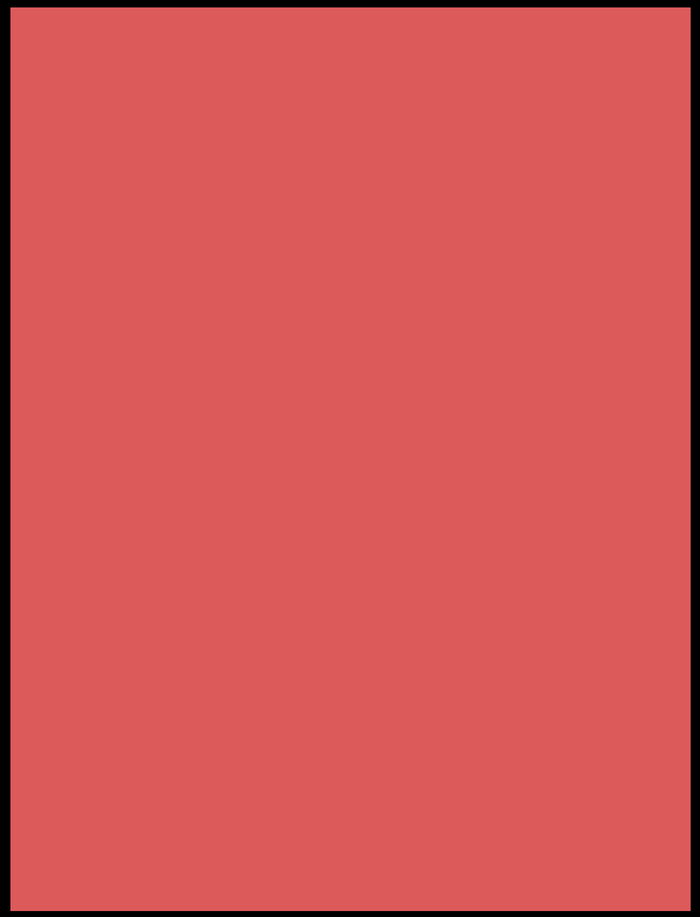
View Controller Transitions



Custom Transitions Using View Controllers

WWDC 2013

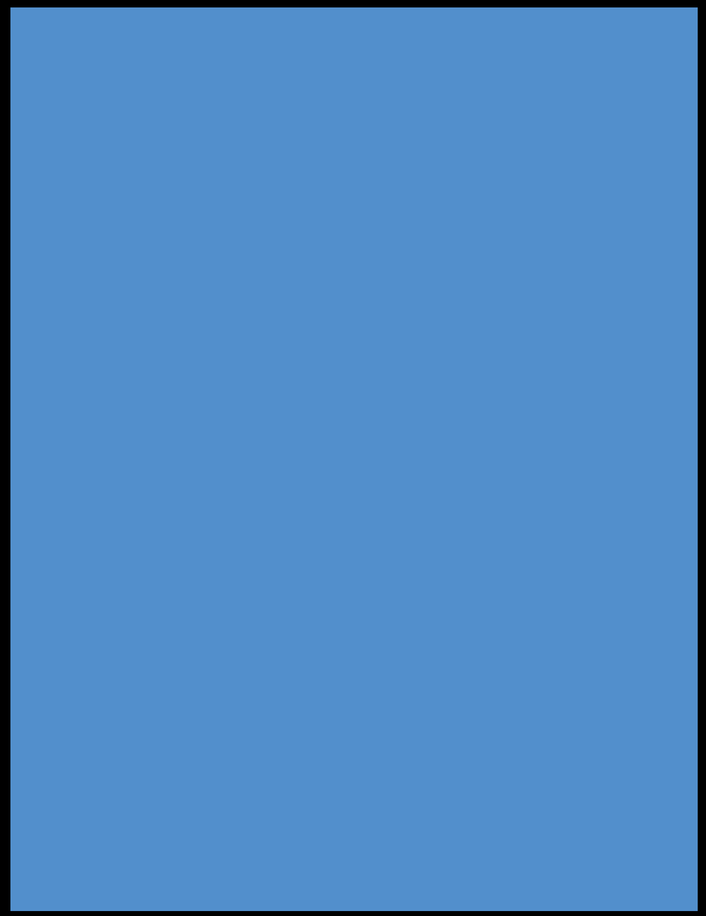
View Controller Transitions



Custom Transitions Using View Controllers

WWDC 2013

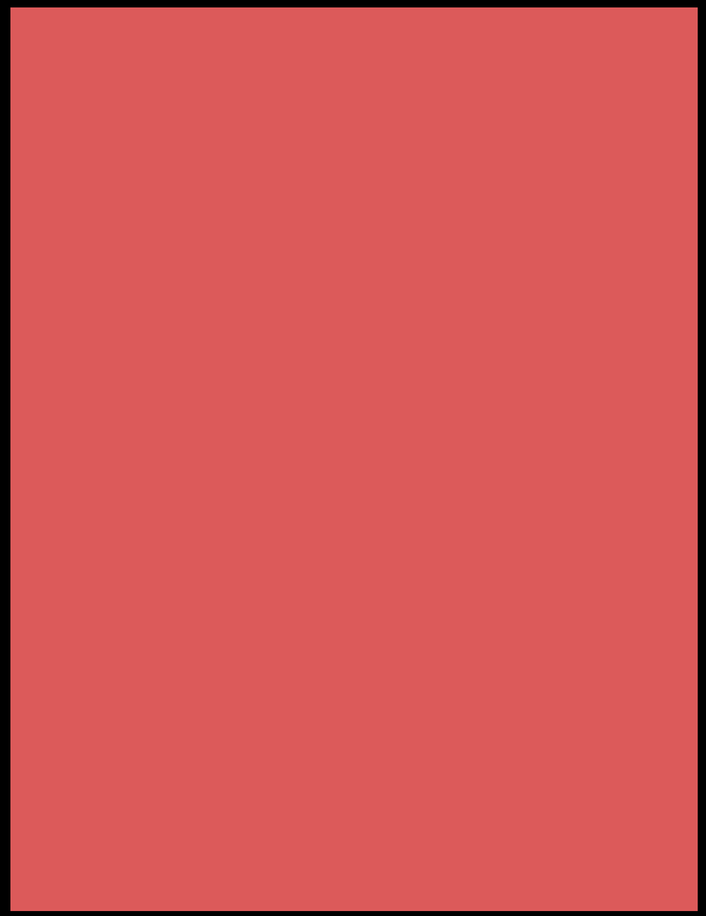
View Controller Transitions



Custom Transitions Using View Controllers

WWDC 2013

View Controller Transitions

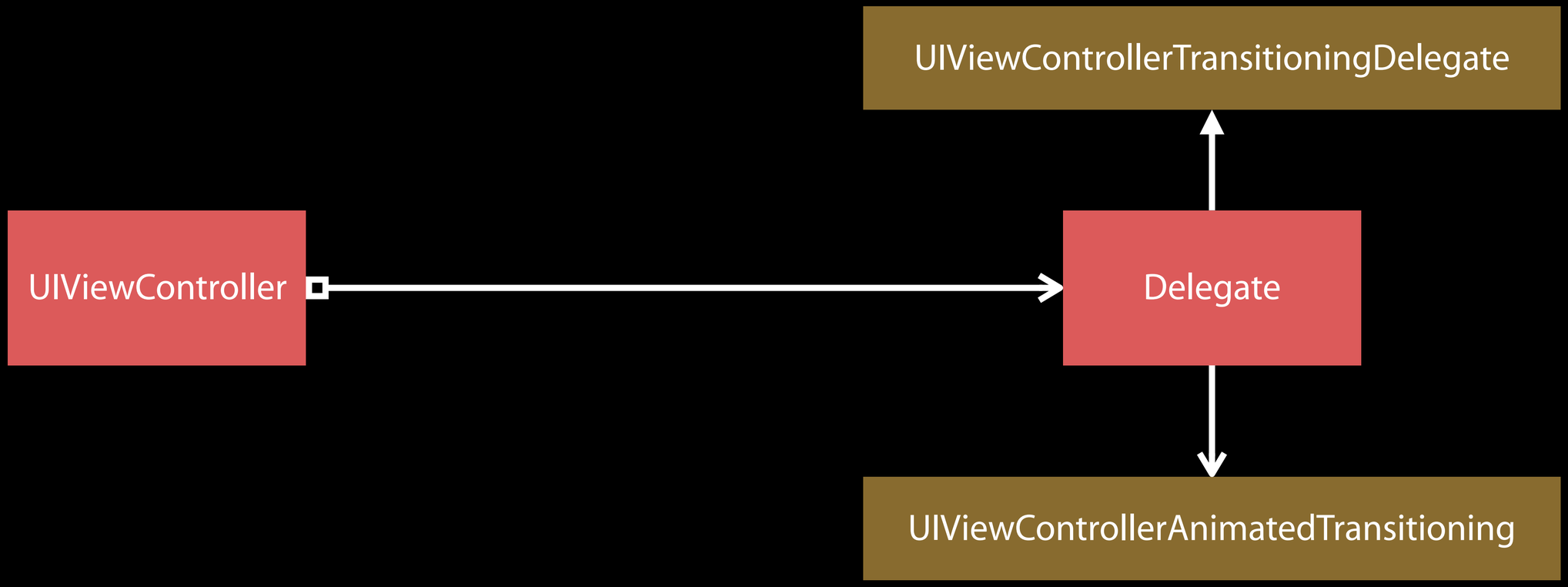


Custom Transitions Using View Controllers

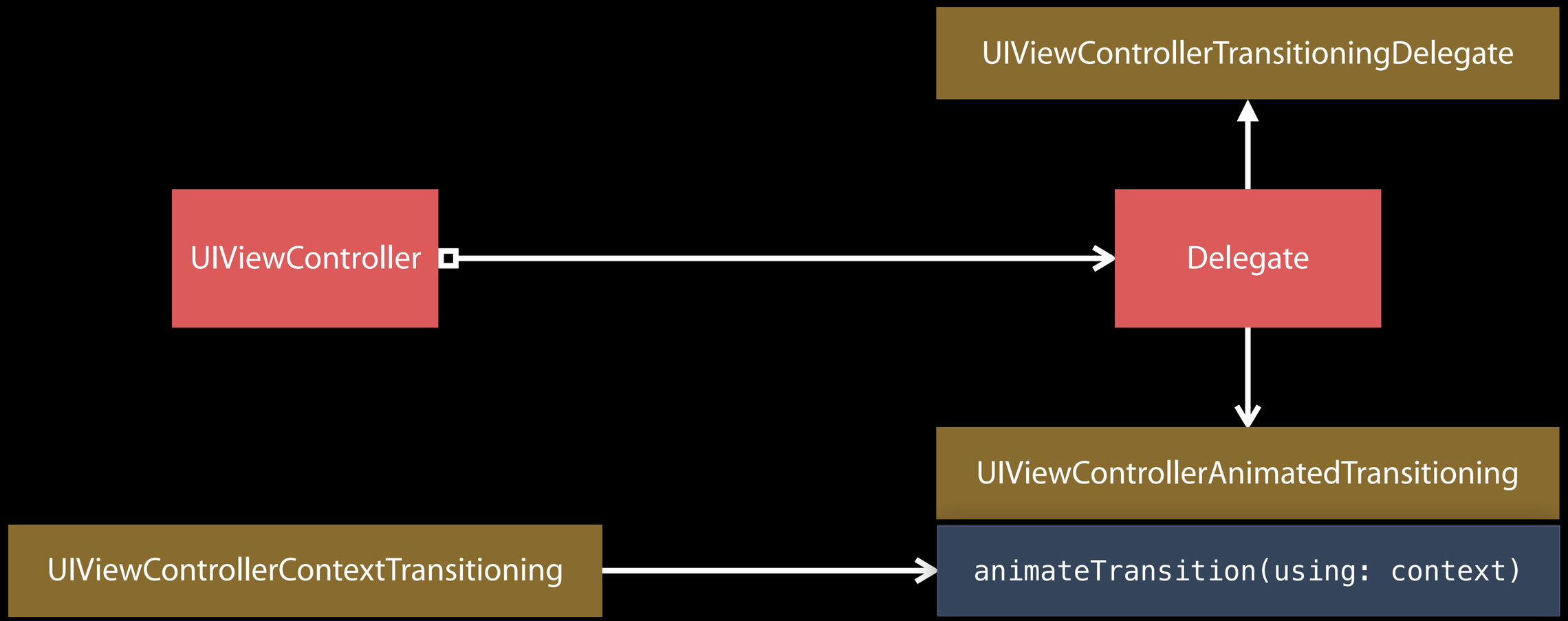
WWDC 2013

UIViewControllerAnimatedTransitioning

UIViewControllerAnimatedTransitioning

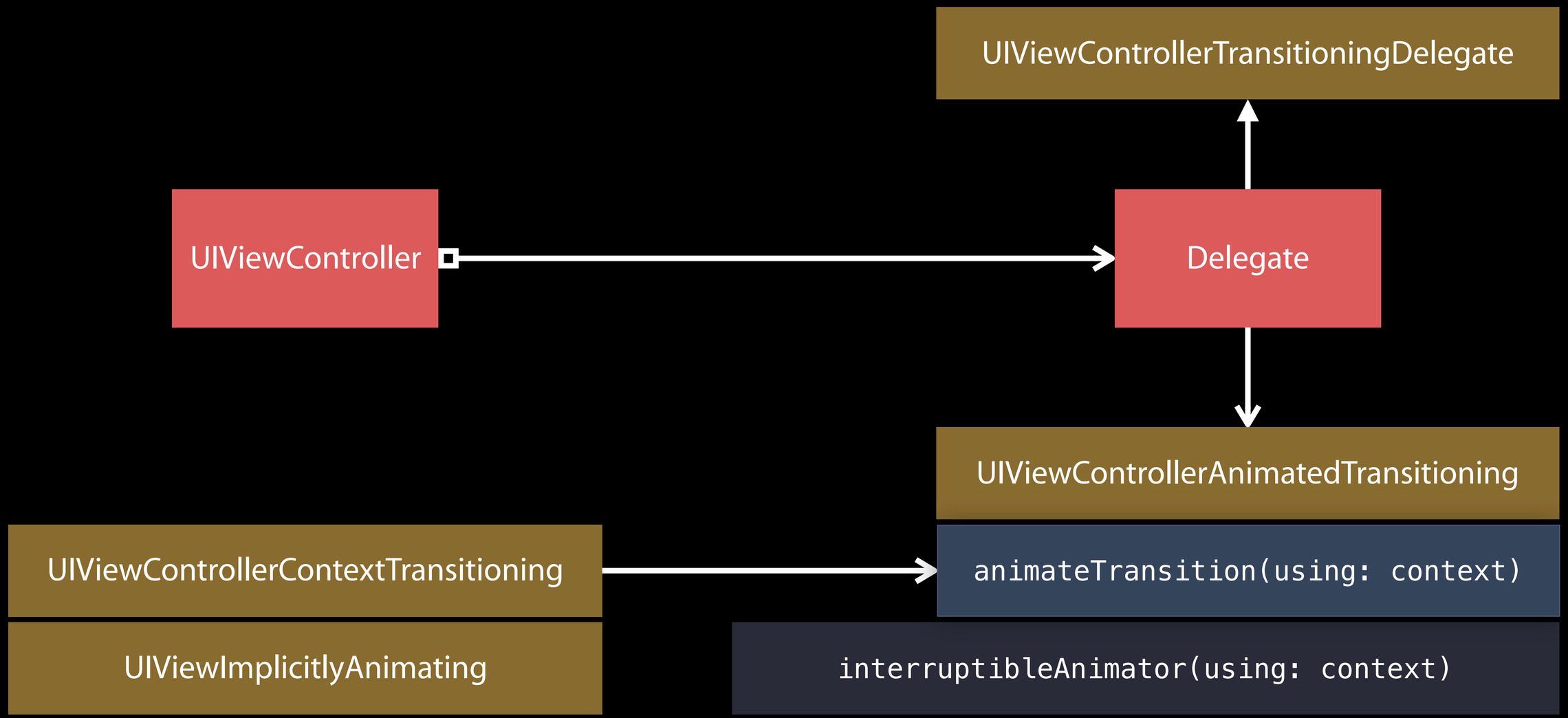


UIViewControllerAnimatedTransitioning



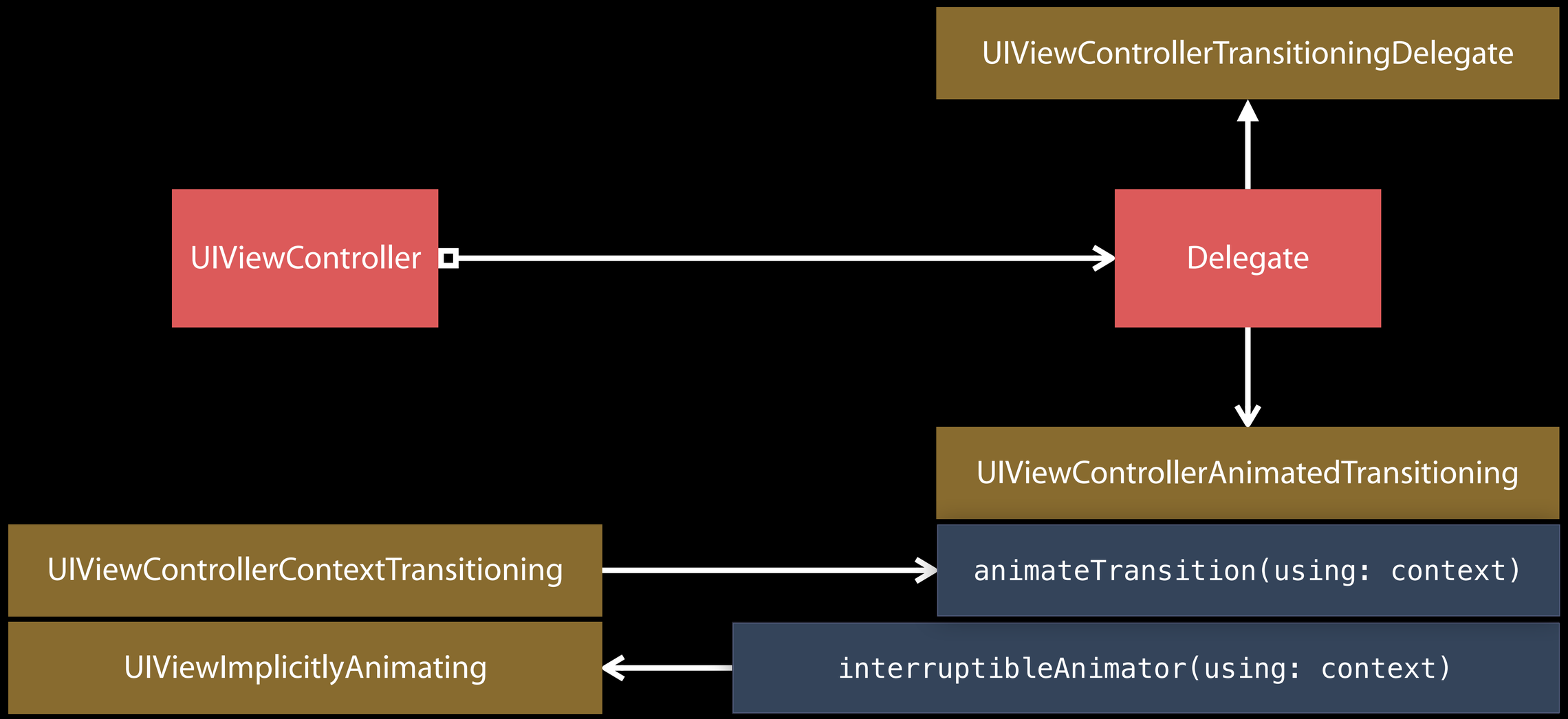
UIViewControllerAnimatedTransitioning

NEW



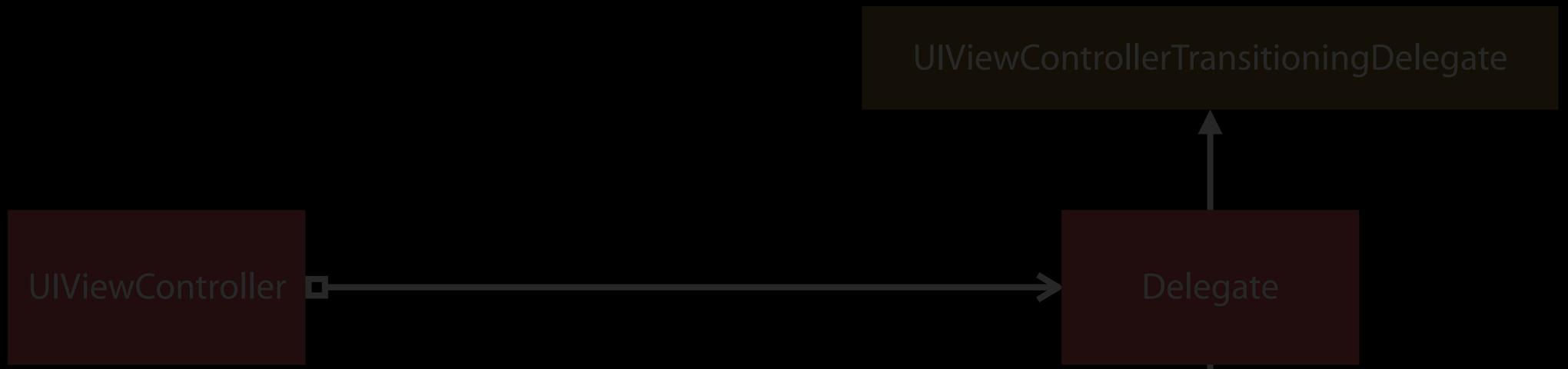
UIViewControllerAnimatedTransitioning

NEW



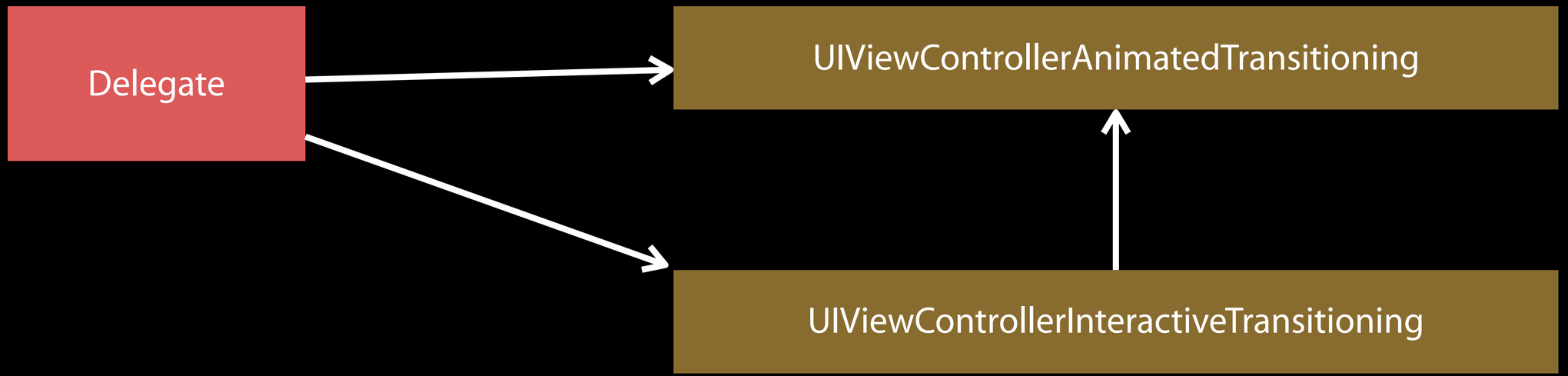
UIViewControllerAnimatedTransitioning

NEW

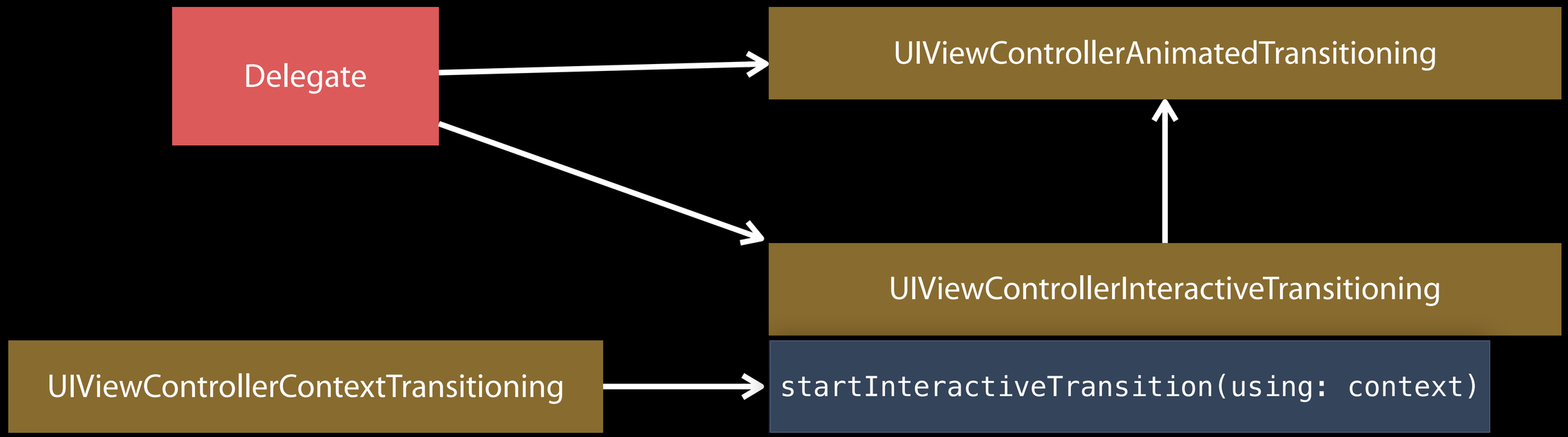



```
func animateTransition(using context:UIViewControllerTransitioningContext) -> Void {
   self.interruptibleAnimator(using: context).startTransition()
}
```

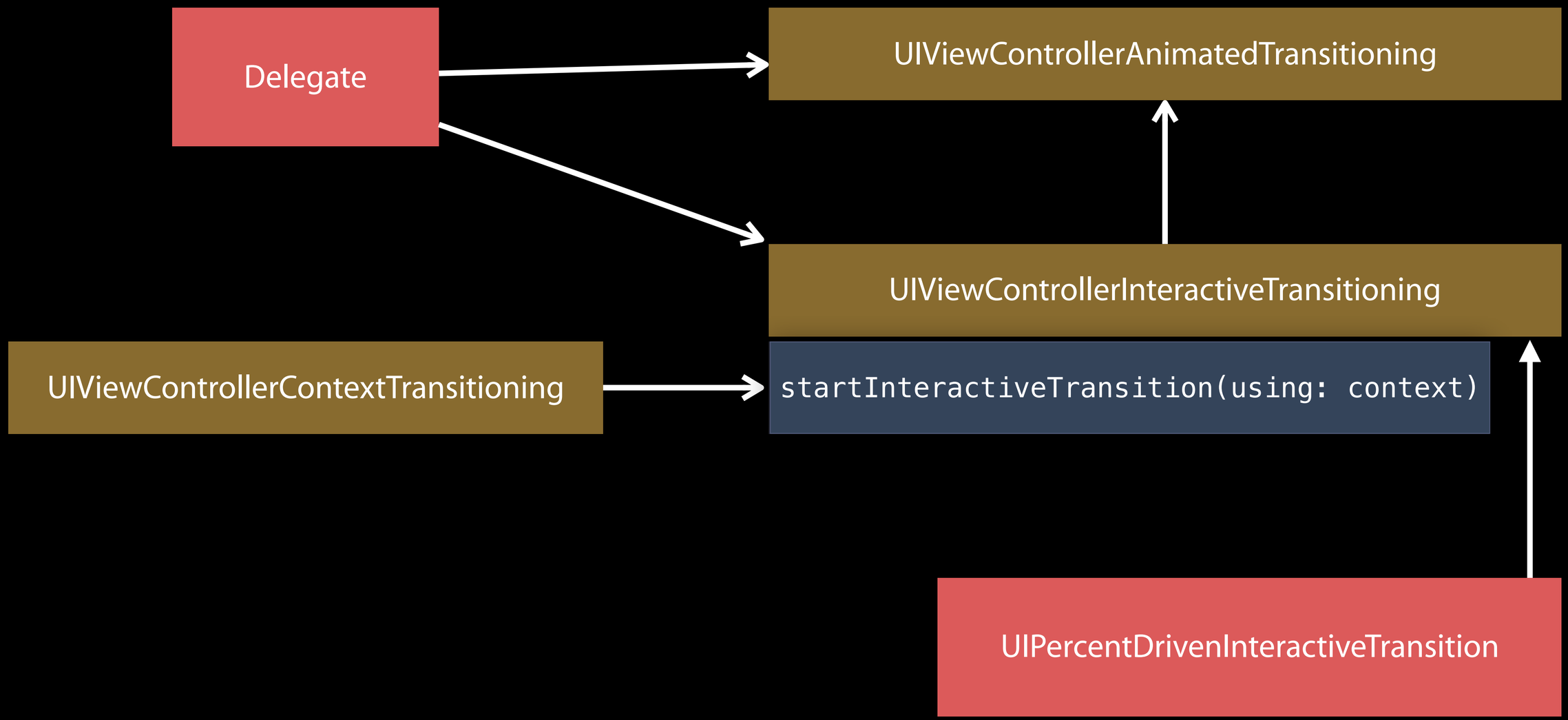
UIViewControllerInteractiveTransitioning



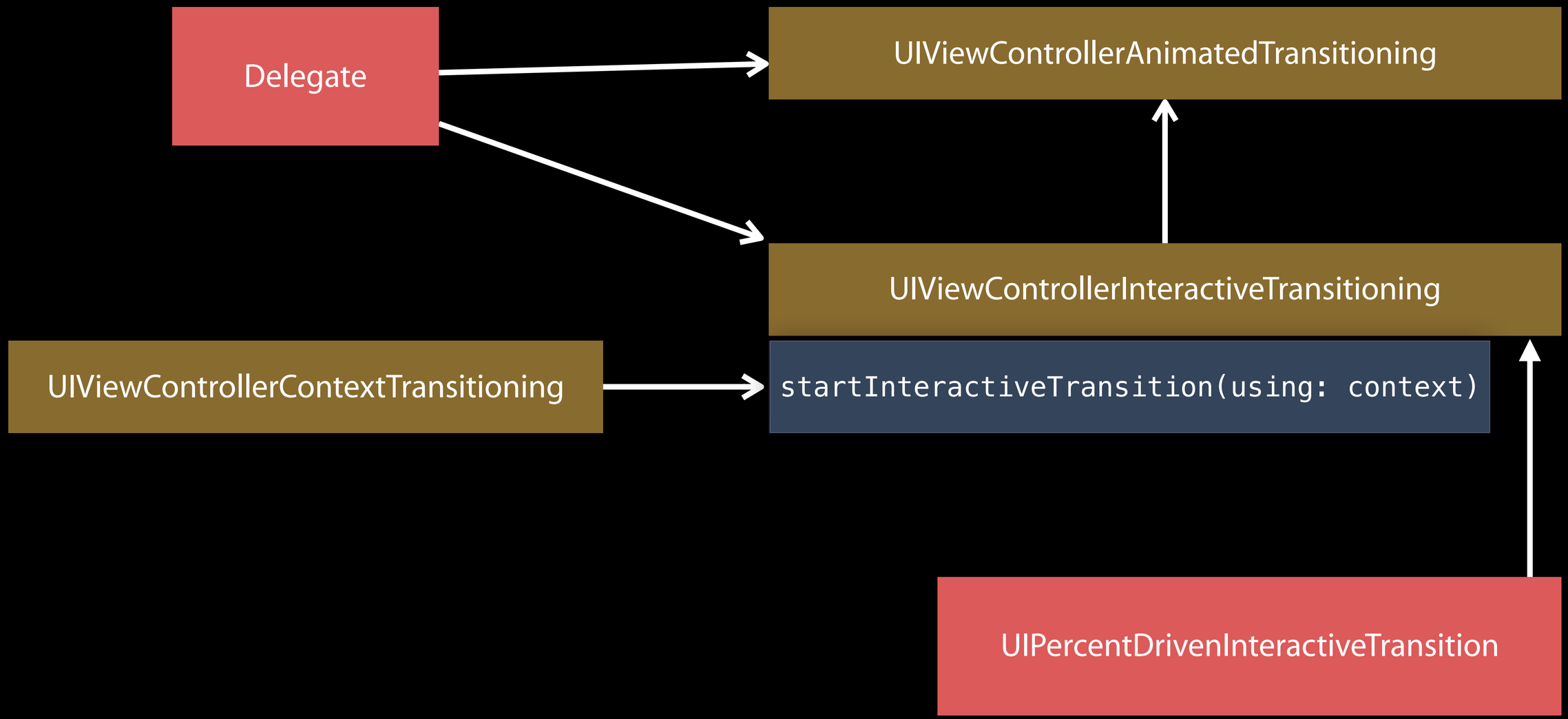
UIViewControllerInteractiveTransitioning



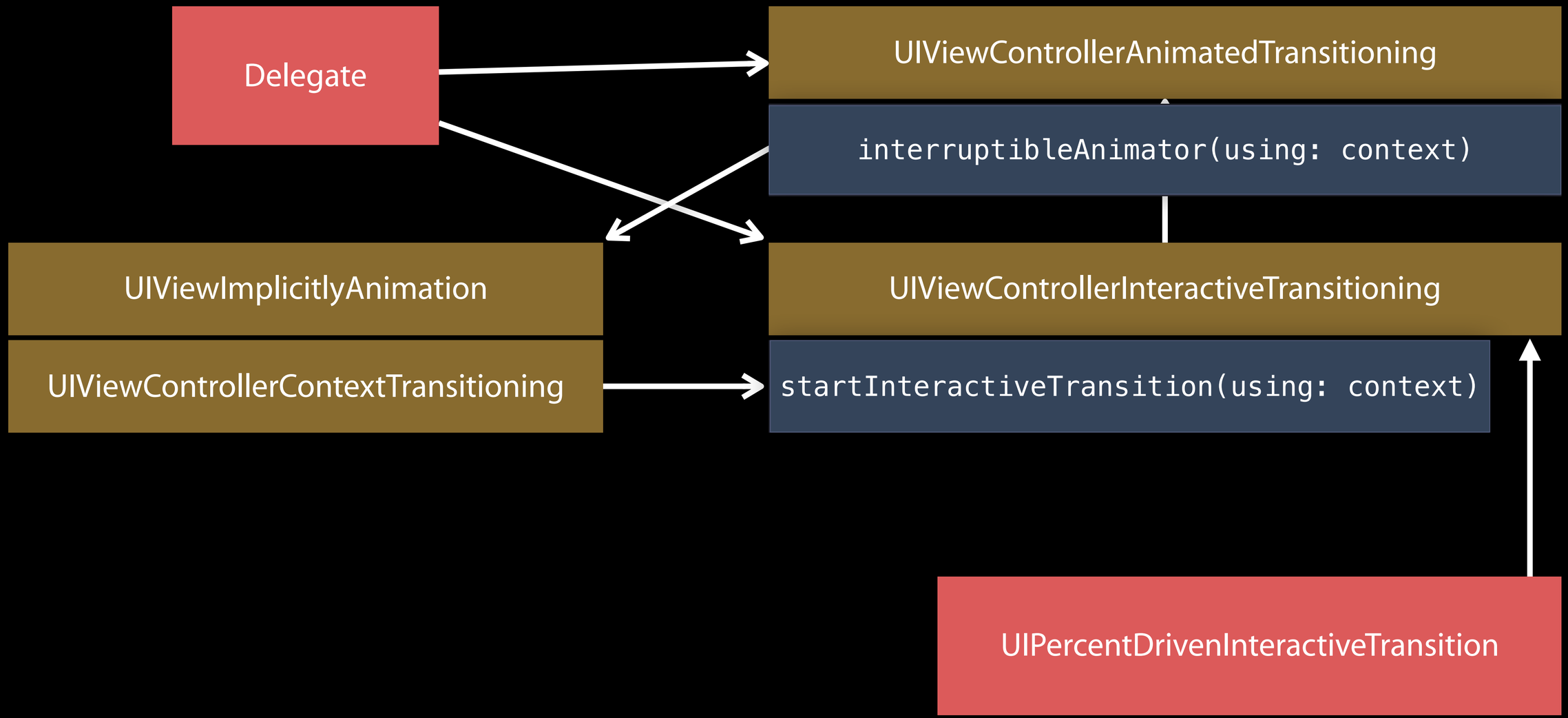
UIViewControllerInteractiveTransitioning



UIViewControllerInteractiveTransitioning



UIViewControllerInteractiveTransitioning



View Controller Transitions

Migrating

```
func interruptibleAnimator(using context:
                           UIViewControllerContextTransitioning) -> UIViewImplicitlyAnimating {
  let timing = UICubicTimingParameters(animationCurve: .easeInOut)
  let animator = UIViewPropertyAnimator(duration:self.duration, timingParameters:timing)
  animator.addAnimations {
    self.myAnimateTransition(context)
  }
  return animator
}
```

View Controller Transitions

Migrating

```
func interruptibleAnimator(using context:
                           UIViewControllerContextTransitioning) -> UIViewImplicitlyAnimating {
  let timing = UICubicTimingParameters(animationCurve: .easeInOut)
  let animator = UIViewPropertyAnimator(duration:self.duration, timingParameters:timing)
  animator.addAnimations {
    self.myAnimateTransition(context)
  }
  return animator
}
```

View Controller Transitions

UIViewControllerContextTransitioning

NEW

```
public protocol UIViewControllerContextTransitioning : NSObjectProtocol {
...
    // This should be called if the transition animation is interruptible and it
    // is being paused.
    @available(iOS 10.0, *)
    public func pauseInteractiveTransition()

    // The next two values can change if the animating transition is interruptible.
    public var isInteractive : Bool { get }// This indicates whether the transition is
    currently interactive.
    public var transitionWasCancelled : Bool { get }

    public func updateInteractiveTransition(_ percentComplete: CGFloat)
    public func finishInteractiveTransition()
    public func cancelInteractiveTransition()
...
}
```

View Controller Transitions

UIViewControllerContextTransitioning

NEW

```
public protocol UIViewControllerContextTransitioning : NSObjectProtocol {
...
    // This should be called if the transition animation is interruptible and it
    // is being paused.
    @available(iOS 10.0, *)
    public func pauseInteractiveTransition()

    // The next two values can change if the animating transition is interruptible.
    public var isInteractive : Bool { get }// This indicates whether the transition is
    currently interactive.
    public var transitionWasCancelled : Bool { get }

    public func updateInteractiveTransition(_ percentComplete: CGFloat)
    public func finishInteractiveTransition()
    public func cancelInteractiveTransition()
...
}
```

View Controller Transitions

UIViewControllerContextTransitioning

NEW

```
public protocol UIViewControllerContextTransitioning : NSObjectProtocol {
...
    // This should be called if the transition animation is interruptible and it
    // is being paused.
    @available(iOS 10.0, *)
    public func pauseInteractiveTransition()

    // The next two values can change if the animating transition is interruptible.
    public var isInteractive : Bool { get }// This indicates whether the transition is
    currently interactive.
    public var transitionWasCancelled : Bool { get }

    public func updateInteractiveTransition(_ percentComplete: CGFloat)
    public func finishInteractiveTransition()
    public func cancelInteractiveTransition()
...
}
```

View Controller Transitions

UIViewControllerInteractiveTransitioning

NEW

```
public protocol UIViewControllerInteractiveTransitioning : NSObjectProtocol {
   optional public var wantsInteractiveStart : Bool { get }
}
```

View Controller Transitions

UIPercentDrivenInteractiveTransition

NEW

```
public class UIPercentDrivenInteractiveTransition : NSObject,
UIViewControllerInteractiveTransitioning {


    public var timingCurve: UITimingCurveProvider?

    public var wantsInteractiveStart: Bool


    public func pause()
}
```

View Controller Transitions

Rules

View Controller Transitions

Rules

Implementation of `interruptibleAnimator(using: context)` implies adoption

View Controller Transitions

Rules

Implementation of `interruptibleAnimator(using: context)` implies adoption

`animateTransition(using: context)` or
`startInteractiveTransition(using: context)` are called first

View Controller Transitions

Rules

Implementation of `interruptibleAnimator(using: context)` implies adoption

`animateTransition(using: context)` or
`startInteractiveTransition(using: context)` are called first

`interruptibleAnimator(using: context)` returns the same instance

View Controller Transitions

Rules

Implementation of `interruptibleAnimator(using: context)` implies adoption

`animateTransition(using: context)` or
`startInteractiveTransition(using: context)` are called first

`interruptibleAnimator(using: context)` returns the same instance

The animator survives the life of the transition

Demo

An improved sample Photos app

Michael Turner UIKit

Two More Topics

Agenda

Review

Discussion of UIViewPropertyAnimator

UIViewControllerAnimated Transitioning

Demo of a New Photos Sample Application

Animation to Gesture Revisited (Hit Testing)

Interruptible Keyframe Animations

Agenda

Review

Discussion of UIViewPropertyAnimator

UIViewControllerAnimated Transitioning

Demo of a New Photos Sample Application

Animation to Gesture Revisited (Hit Testing)

Interruptible Keyframe Animations

UIViewPropertyAnimator

Hit testing

```
var isUserInteractionEnabled: Bool


var isManualHitTestingEnabled: Bool
```

UIViewPropertyAnimator

Hit testing

```
var isUserInteractionEnabled: Bool

var isManualHitTestingEnabled: Bool
```

Defaults to true

UIViewPropertyAnimator

Hit testing

```
var isUserInteractionEnabled: Bool
```

Defaults to true

```
var isManualHitTestingEnabled: Bool
```

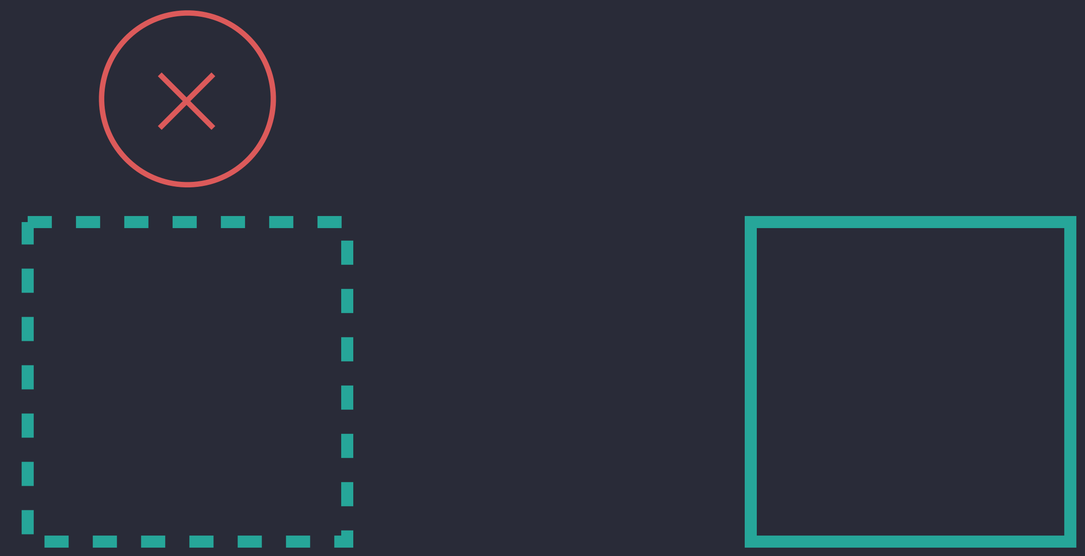
Defaults to false

Hit testing moving views

```
animator.isManualHitTestingEnabled = true
```

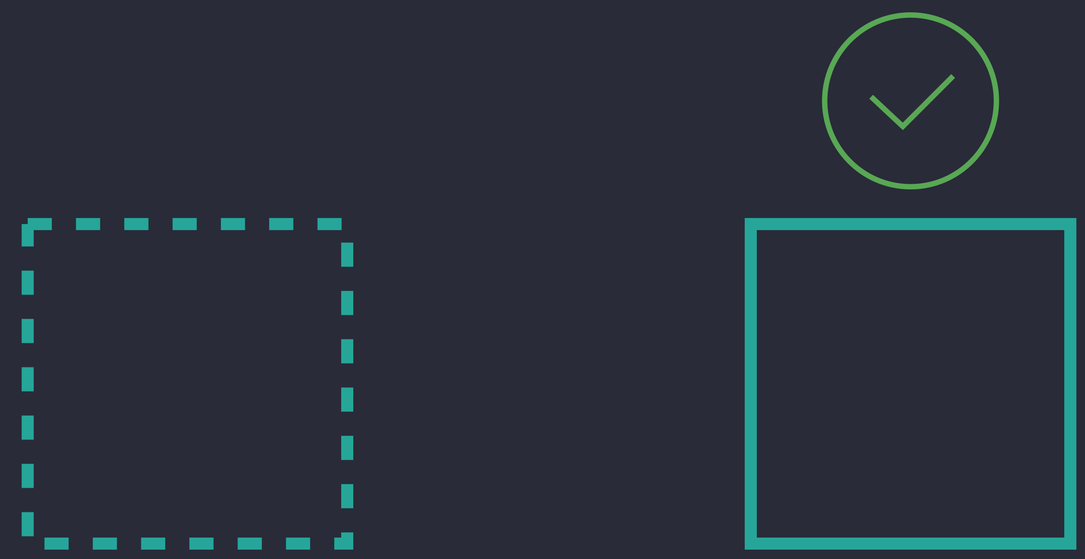
Hit testing moving views

```
animator.isManualHitTestingEnabled = true
```



Hit testing moving views

```
animator.isManualHitTestingEnabled = true
```



Hit testing

```
animator.isManualHitTestingEnabled = true
```

Hit testing

```
animator.isManualHitTestingEnabled = true
```

```
  override func hitTest(_ point: CGPoint, with event: UIEvent!) -> UIView? {
    let superPoint = self.convert(point, to: superview)
    let pt = layer.presentation()?.convert(superPoint, from: superview!.layer)
    return super.hitTest(pt!, with: event)
  }
```

Hit testing

```
animator.isManualHitTestingEnabled = false
```

Hit testing

```
animator.isManualHitTestingEnabled = false
```

Hit testing

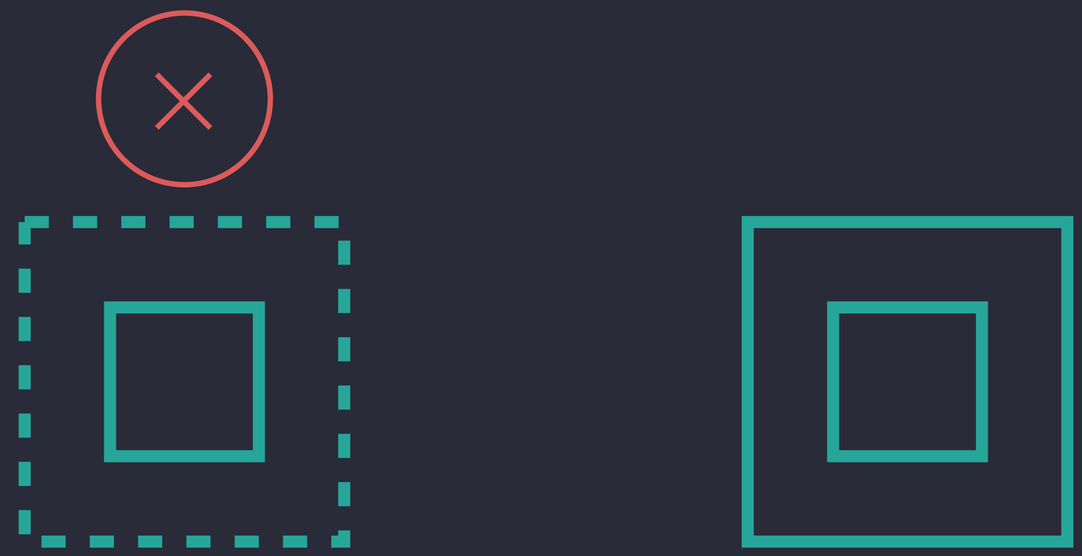
```
animator.isManualHitTestingEnabled = false
```

Hit testing

```
animator.isManualHitTestingEnabled = false
```

Hit testing

```
animator.isManualHitTestingEnabled = false
```



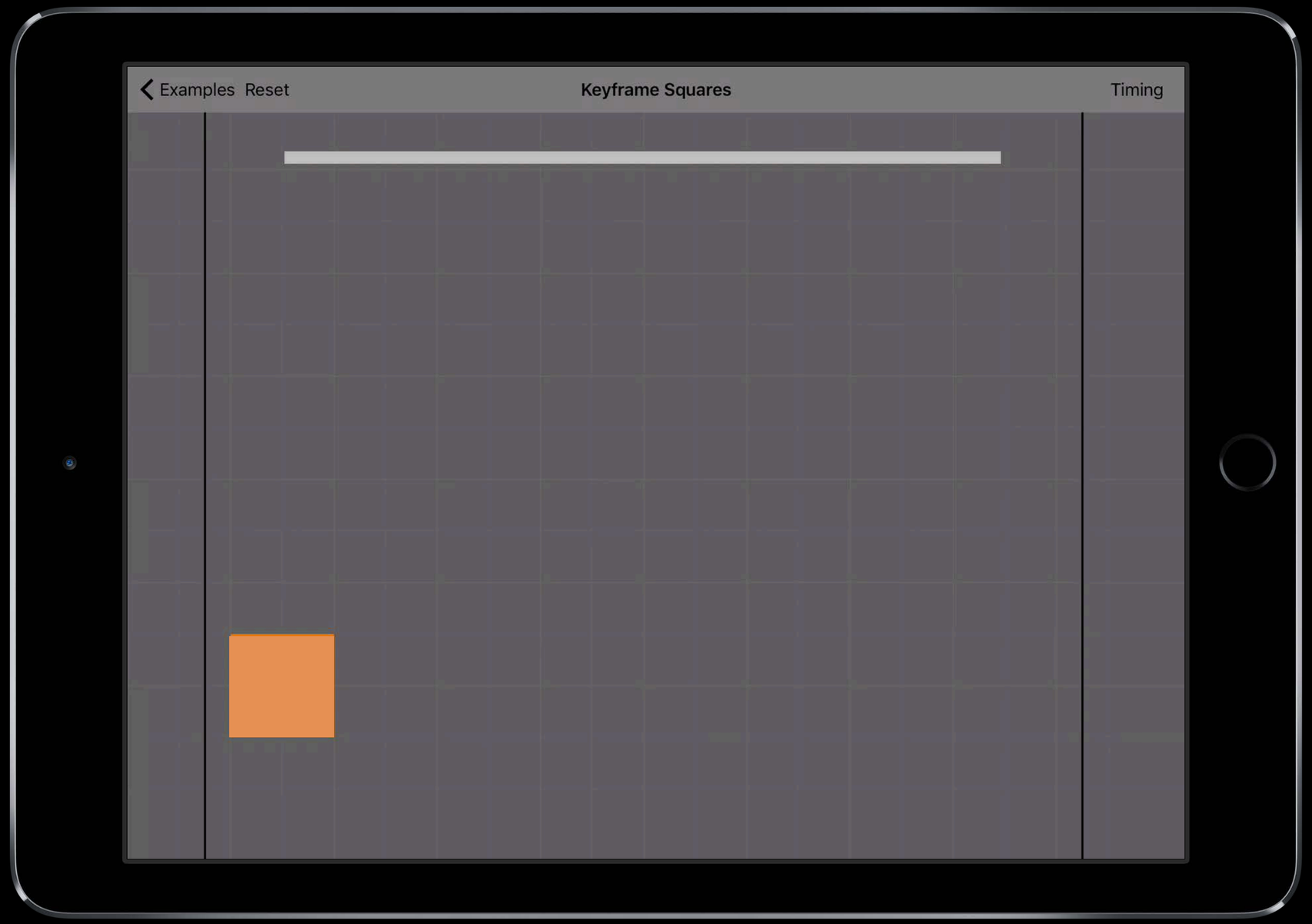
Hit testing

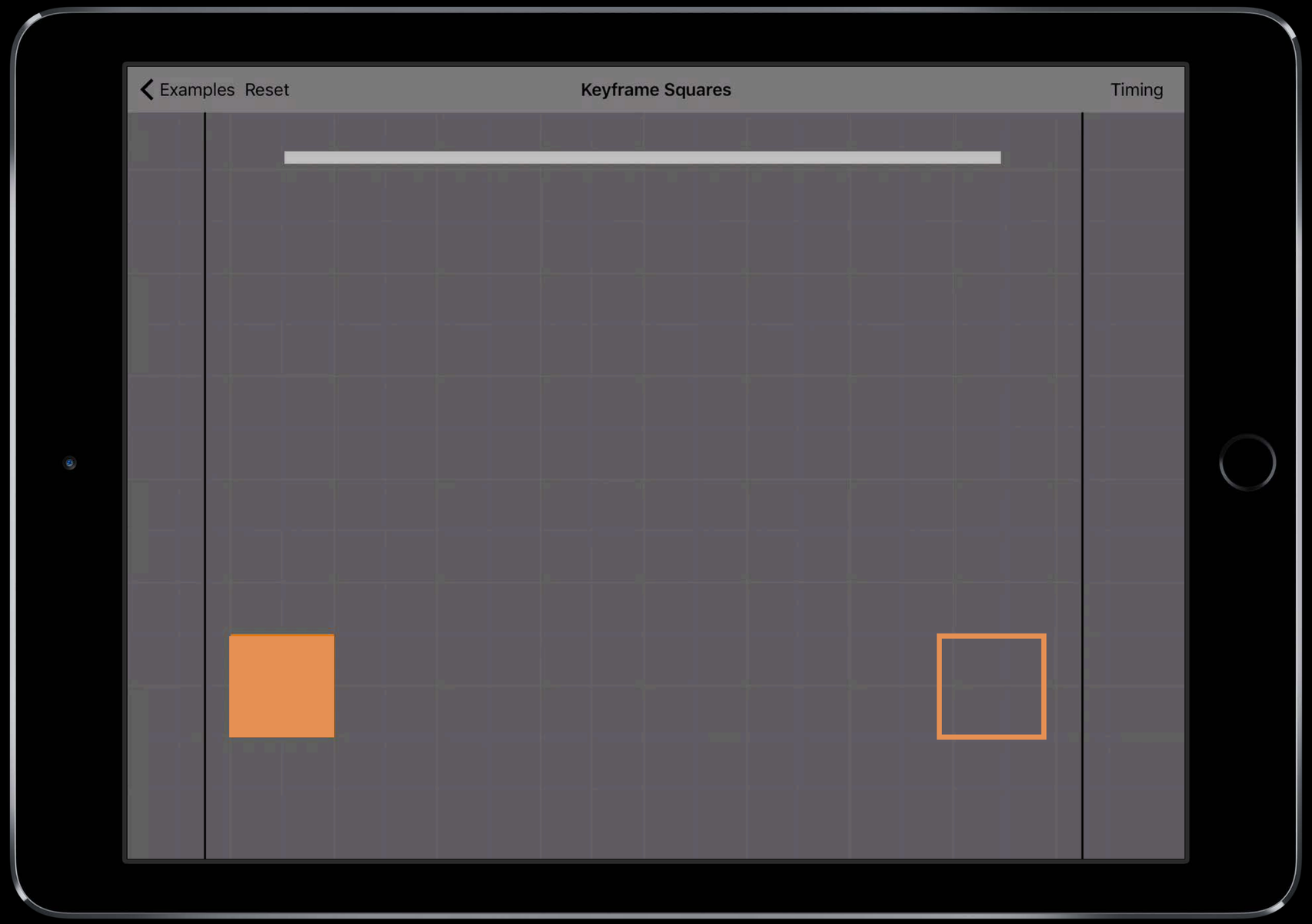
```
animator.isManualHitTestingEnabled = false
```

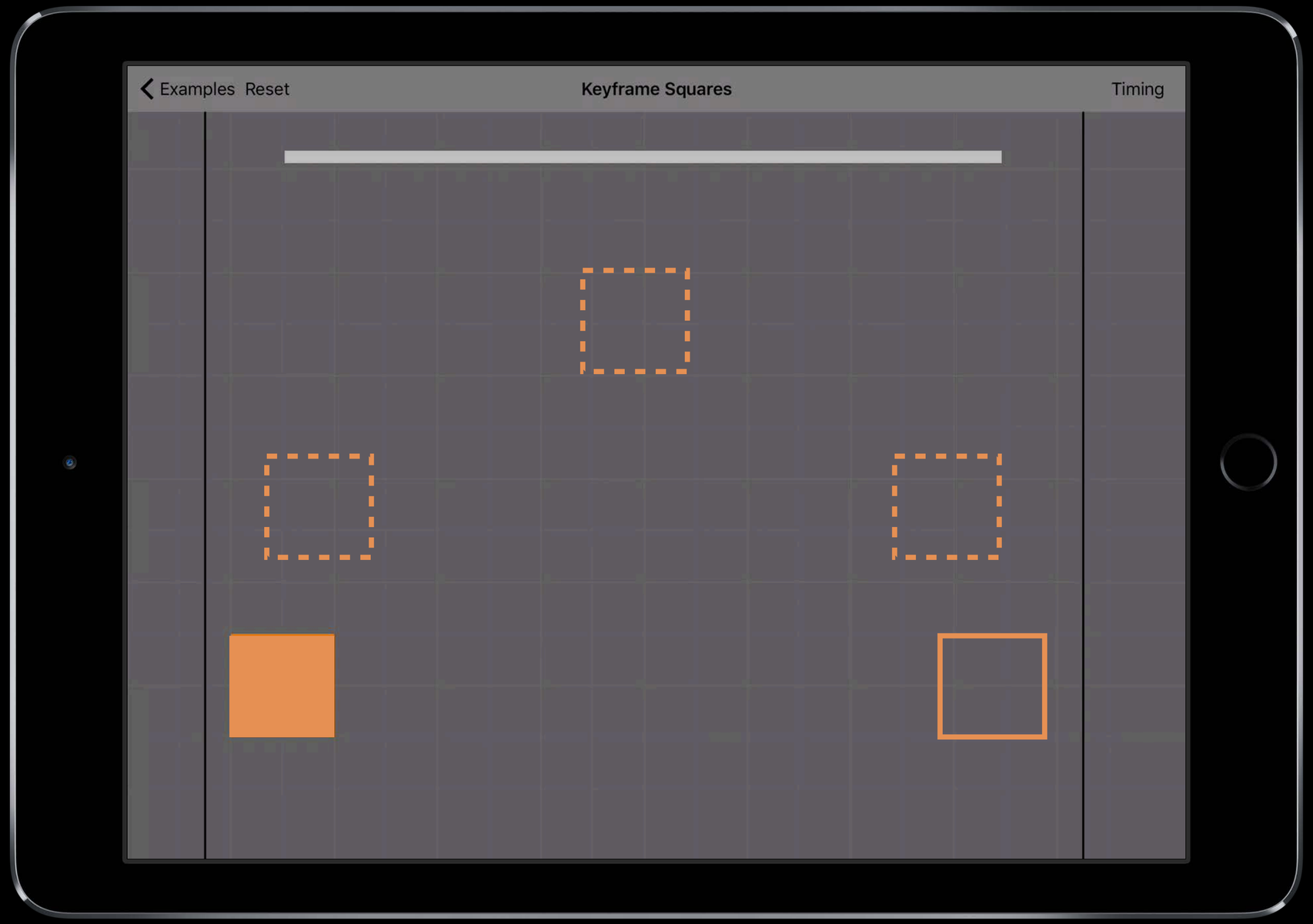
Hit testing

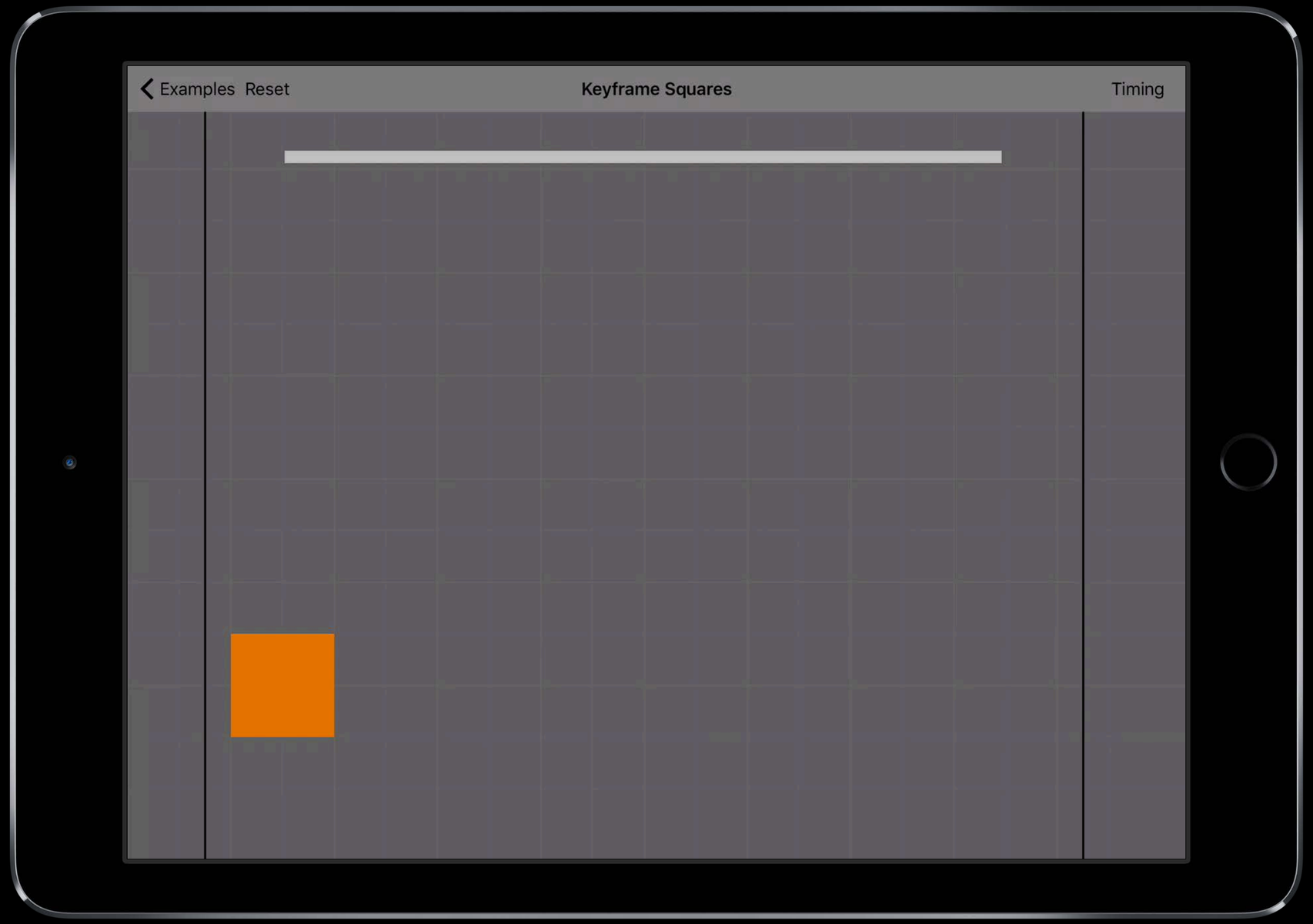
```
animator.isManualHitTestingEnabled = true
```

Keyframe Animations


Examples
Reset
Keyframe Squares
Timing


Examples
Reset
Keyframe Squares
Timing


Examples
Reset
Keyframe Squares
Timing


Examples
Reset
Keyframe Squares
Timing

UIViewPropertyAnimator

Keyframe animations

```
animator.addAnimations { _ in
     UIView.animateKeyframes(withDuration: self.duration, delay: 0.0, options:[.calculationModeCubic]
     {_ in
        UIView.addKeyframe(withRelativeStartTime: 0.0, relativeDuration: 0.25) {
          self.squareView1.center = CGPoint(x: 200.0, y: 450.0)
        }
        UIView.addKeyframe(withRelativeStartTime: 0.25, relativeDuration: 0.25) {
          self.squareView1.center = CGPoint(x: 500.0, y: 250.0)
        }
        UIView.addKeyframe(withRelativeStartTime: 0.5, relativeDuration: 0.25) {
          self.squareView1.center = CGPoint(x: 800.0, y: 450.0)
        }
        UIView.addKeyframe(withRelativeStartTime: 0.75, relativeDuration: 0.25) {
          self.squareView1.center = CGPoint(x: 850.0, y: 600.0)
        }
     }
}
```

interactivePopGestureRecognizer

interactivePopGestureRecognizer

```
let popGesture = navController.interactivePopGestureRecognizer!
```

interactivePopGestureRecognizer

```
let popGesture = navController.interactivePopGestureRecognizer!

myInteractiveGesture.require(toFail: popGesture)
```

Summary

Summary

Create interruptible animations with a UIViewPropertyAnimator

Summary

Create interruptible animations with a UIViewPropertyAnimator

UIViewPropertyAnimators support a wide new range of pacing options

Summary

Create interruptible animations with a UIViewPropertyAnimator

UIViewPropertyAnimators support a wide new range of pacing options

Use UIViewPropertyAnimator to create interruptible view controller transitions

More Information

https://developer.apple.com/wwdc16/216

Related Sessions

What's New in UICollectionView in iOS10	Presidio	Thursday 9:00 AM
A Peek at 3D Touch	Presidio	Thursday 4:00 PM
Custom Transitions Using View Controllers		WWDC 2013
Advanced Techniques with UIKit Dynamics		WWDC 2013
Building Interruptible and Responsive Interactions		WWDC 2014
What's New in UIKit Dynamics and Visual Effects		WWDC 2015

Labs

UIKit and UIKit Animations Lab	Frameworks Lab C	Thursday 1:00 PM

WWDC16