App Frameworks

#WWDC17

Modern User Interaction on iOS

Mastering the UIKit UIGestureRecognizer System

Session 219

Dominik Wagner, UIKit Engineer
Michael Turner, UIKit Engineer
Glen Low, UIKit Engineer

© 2017 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

Multi-Touch

The UIGestureRecognizer system

System gesture interaction

Playing nice with Drag and Drop

The UIGestureRecognizer system

System gesture interaction

Playing nice with Drag and Drop

The UIGestureRecognizer system

System gesture interaction

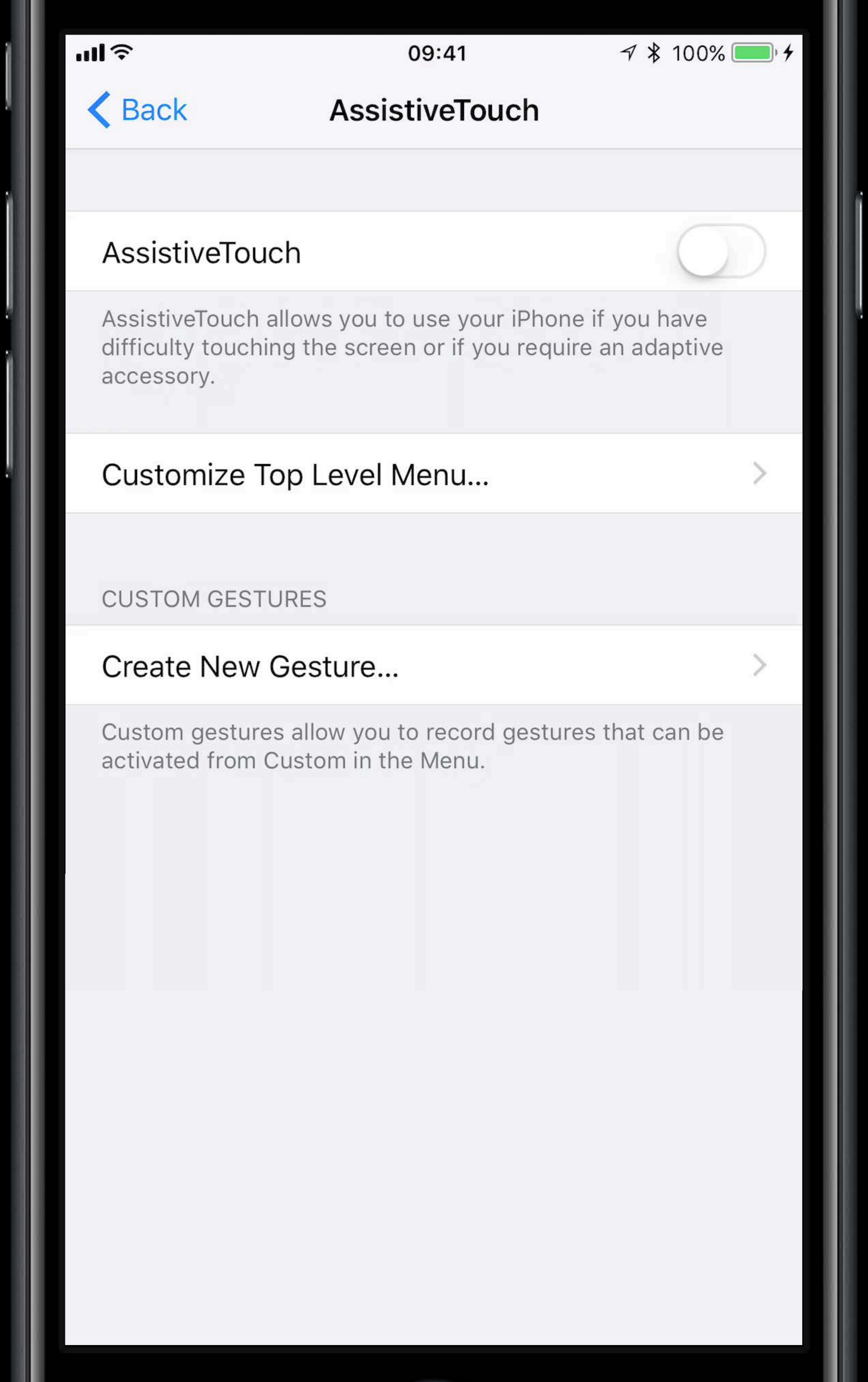
Playing nice with Drag and Drop

Basics

Basics

UITouch

UIGestureRecognizer



AssistiveTouch

AssistiveTouch ○

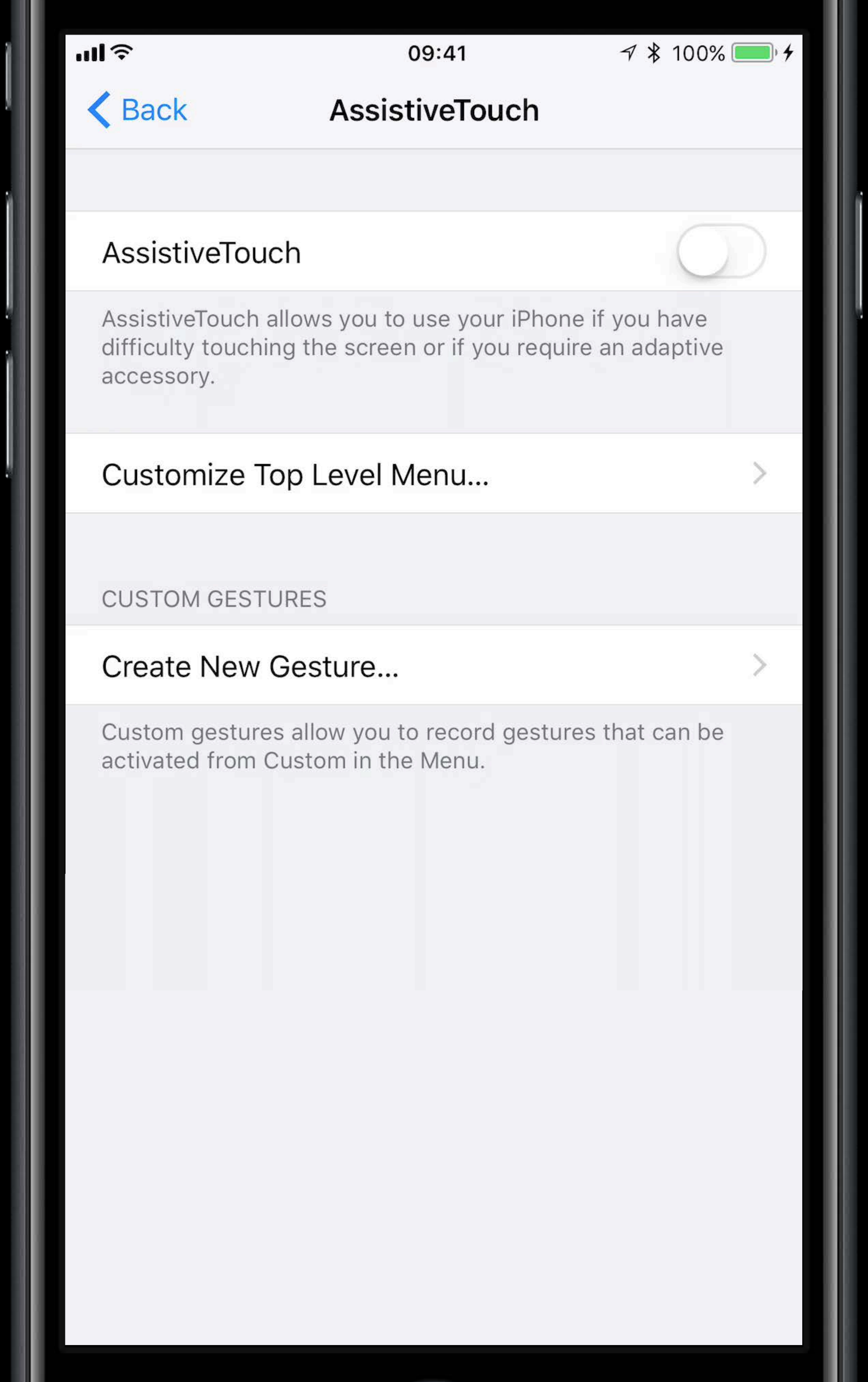
AssistiveTouch allows you to use your iPhone if you have difficulty touching the screen or if you require an adaptive accessory.

Customize Top Level Menu... >

CUSTOM GESTURES

Create New Gesture... >

Custom gestures allow you to record gestures that can be activated from Custom in the Menu.



AssistiveTouch

AssistiveTouch

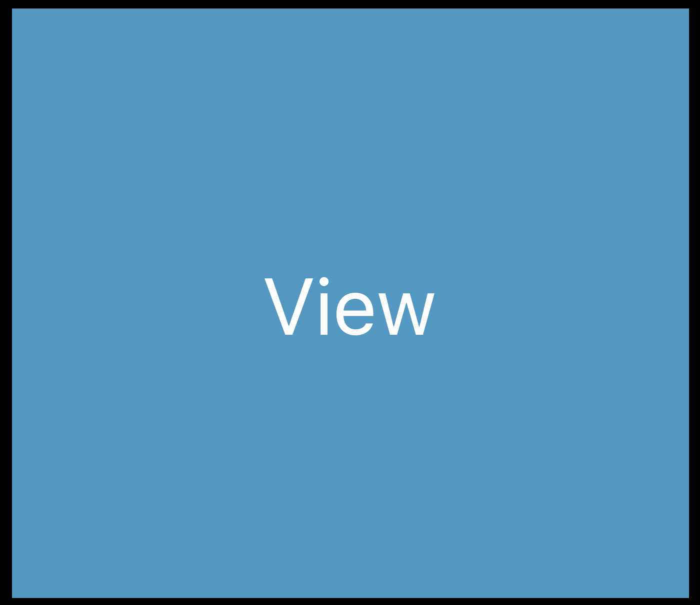
AssistiveTouch allows you to use your iPhone if you have difficulty touching the screen or if you require an adaptive accessory.

Customize Top Level Menu...

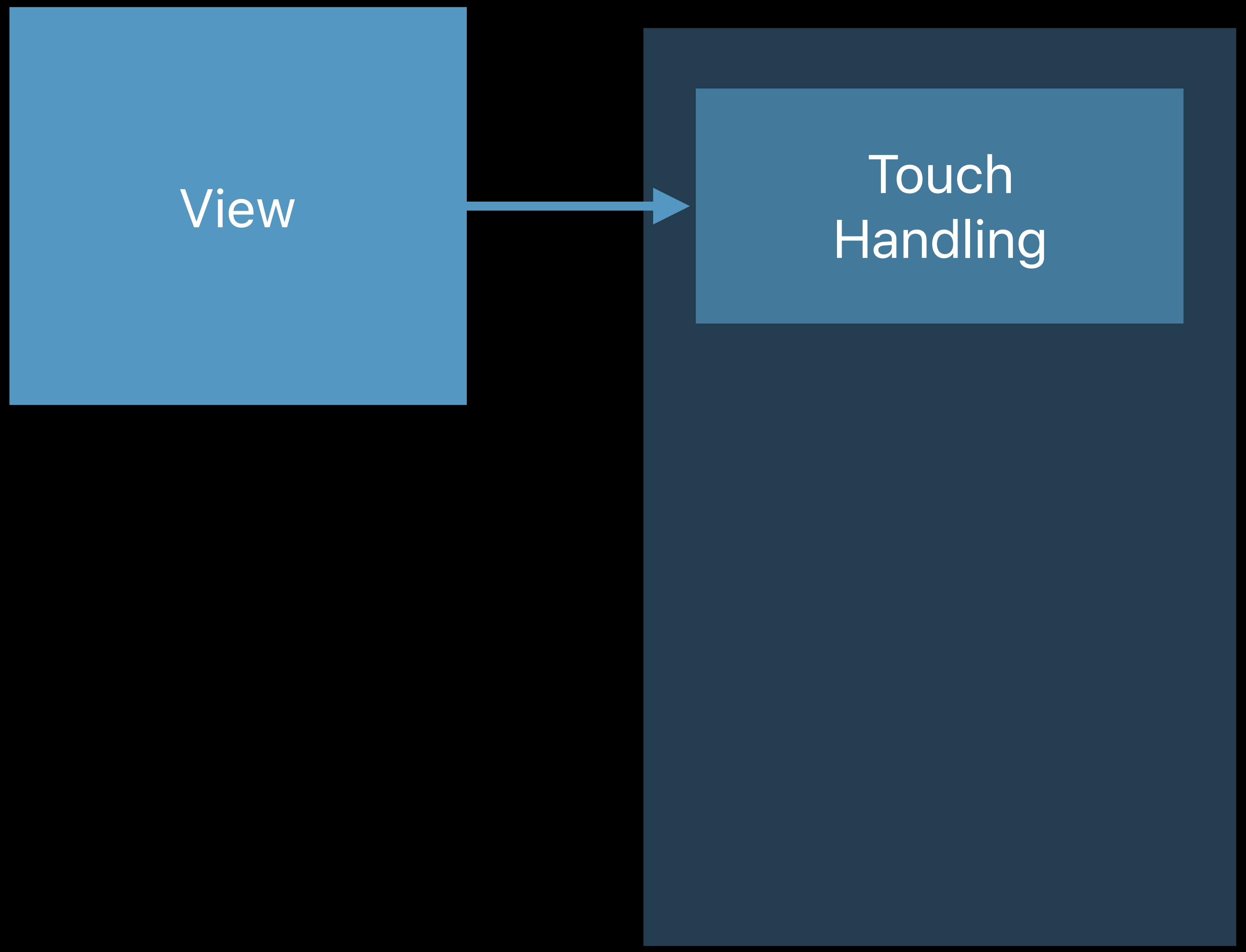
CUSTOM GESTURES

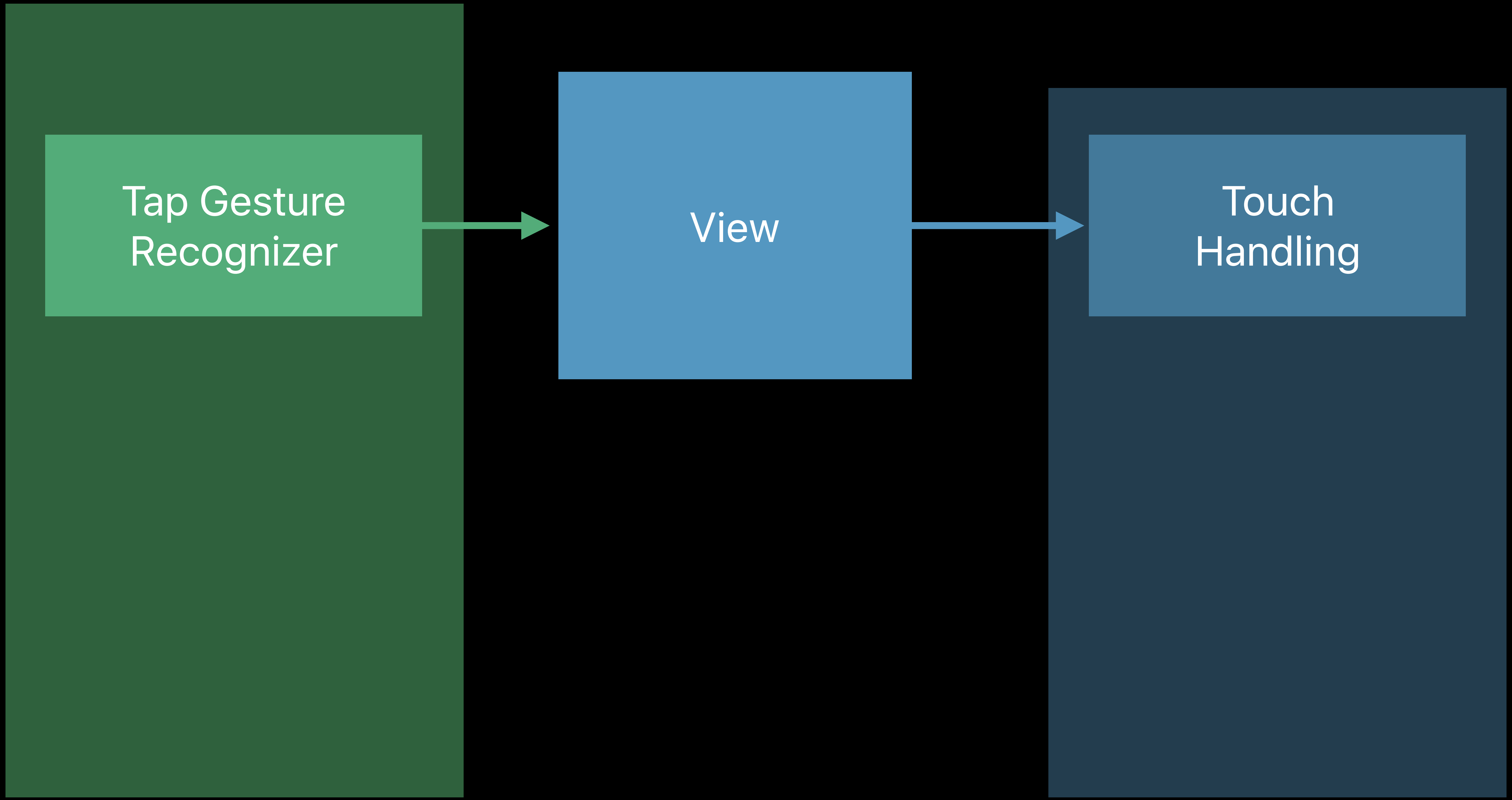
Create New Gesture...

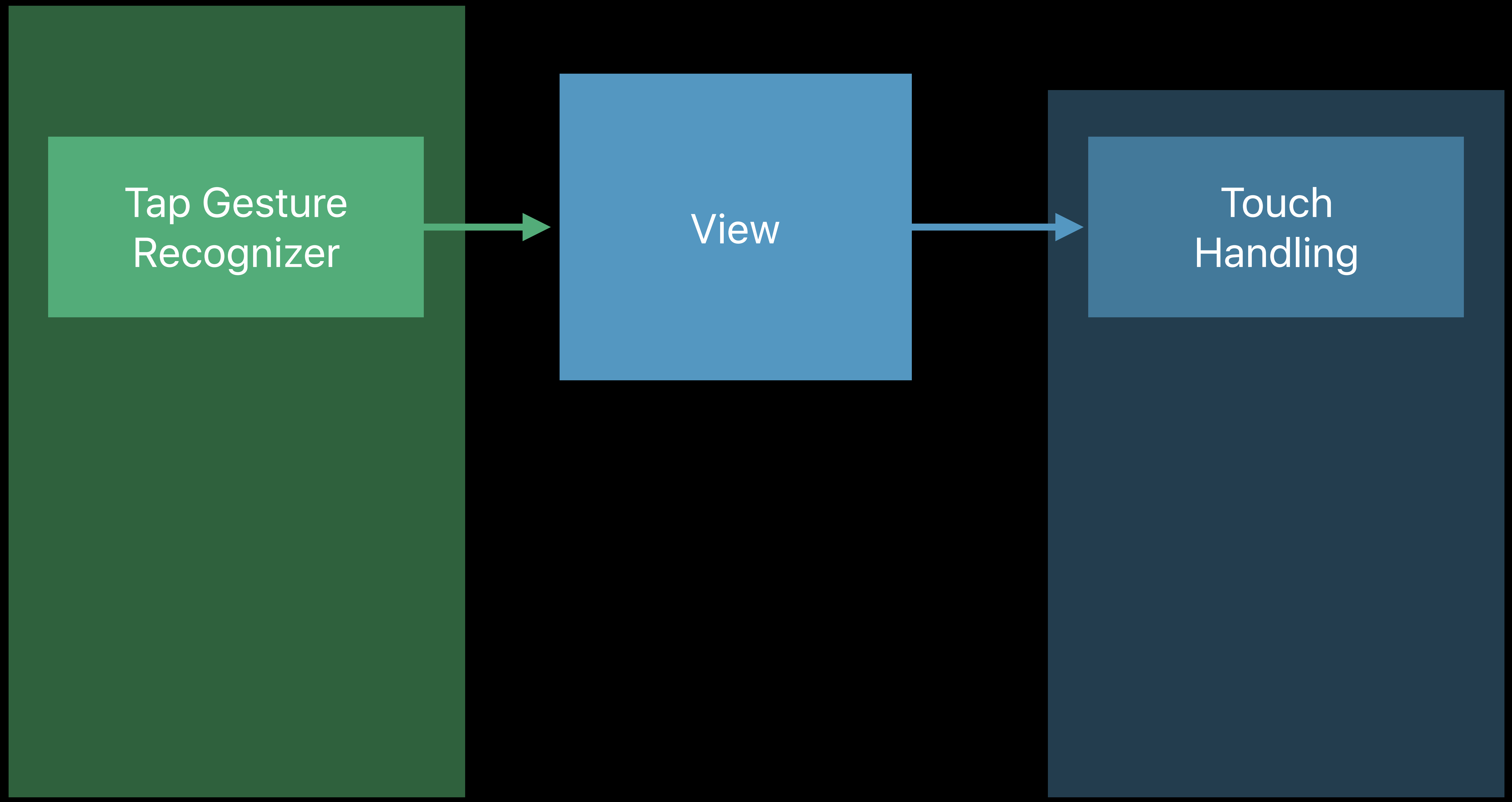
Custom gestures allow you to record gestures that can be activated from Custom in the Menu.

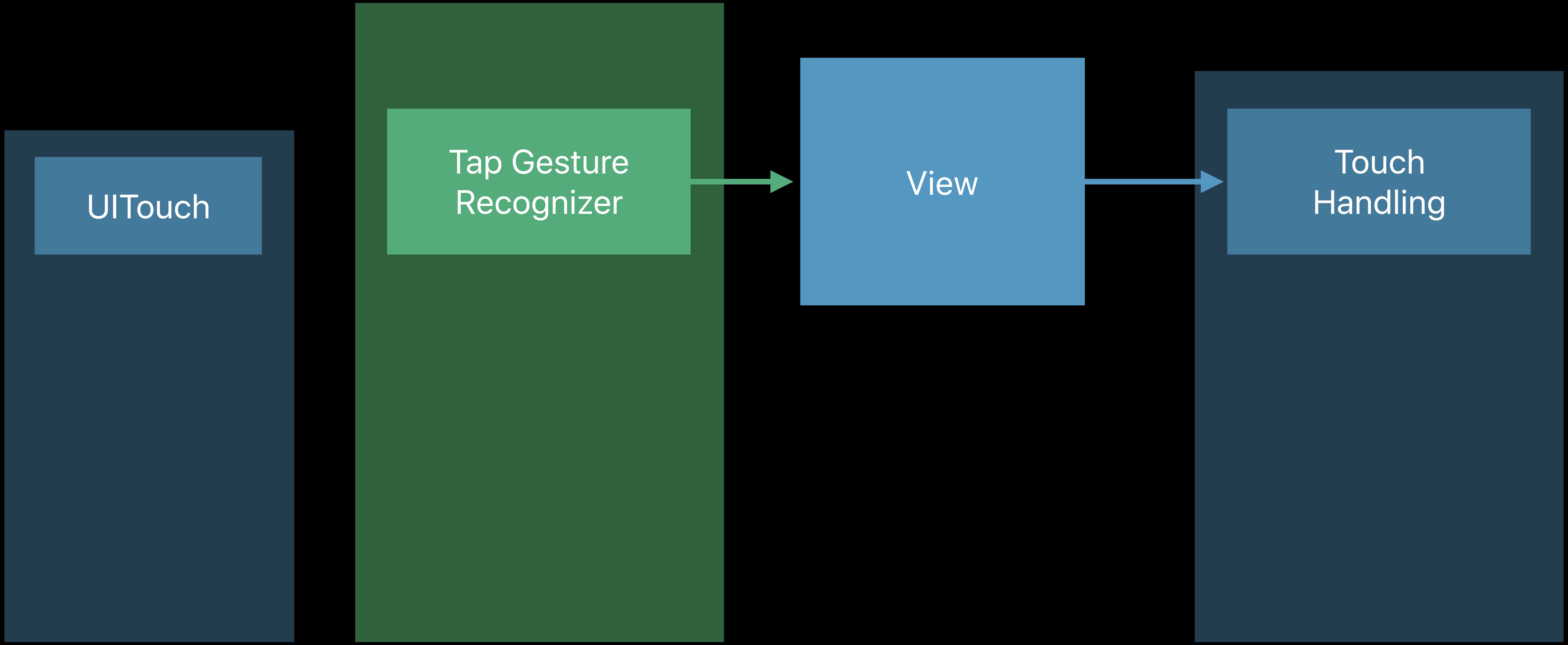


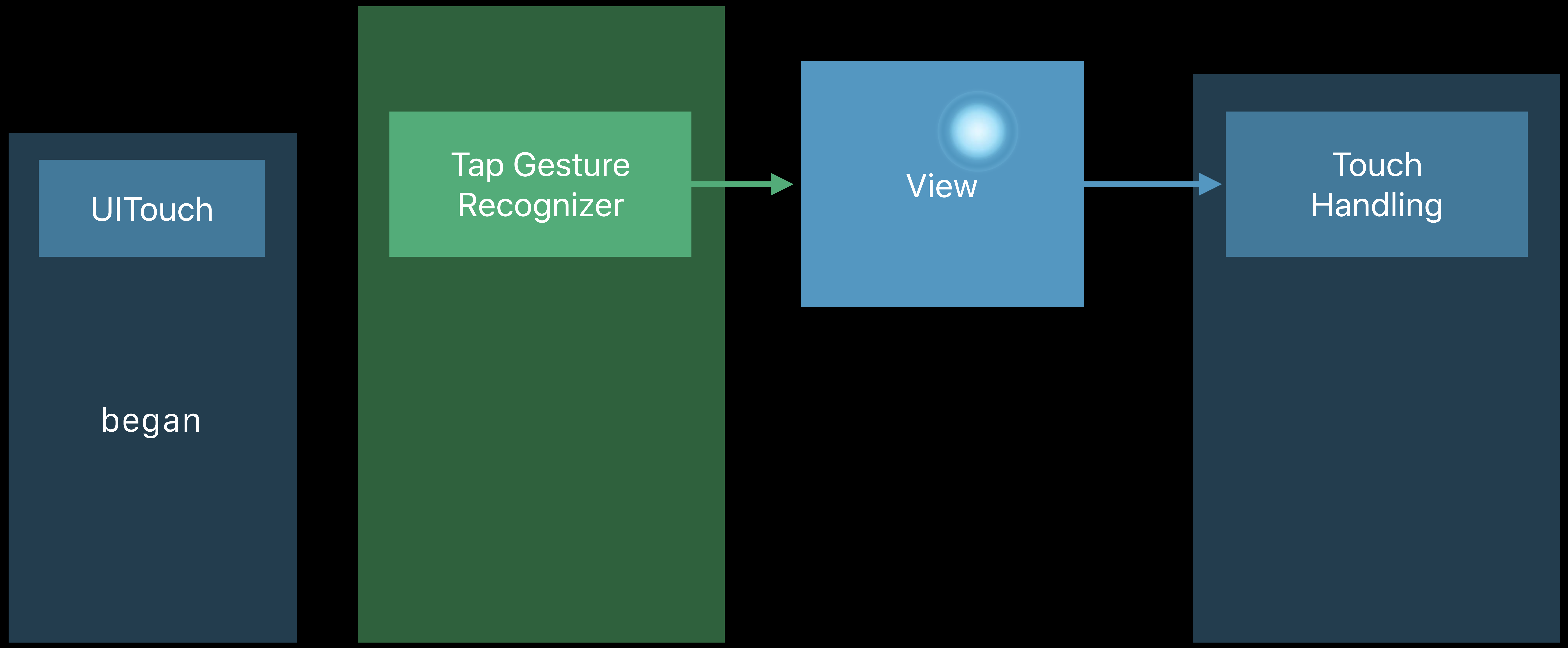
View

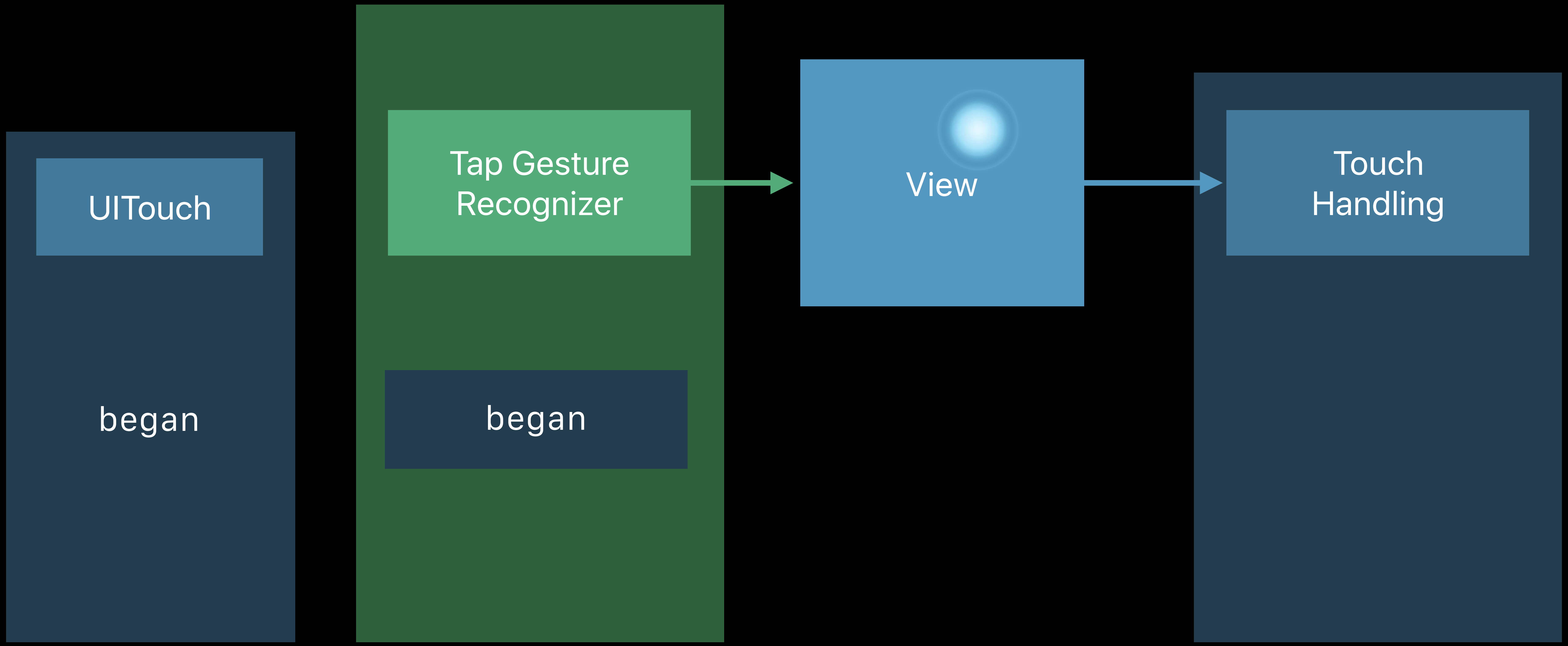

View
Touch Handling


Tap Gesture Recognizer
View
Touch Handling


Tap Gesture Recognizer
View
Touch Handling


UITouch
Tap Gesture Recognizer
View
Touch Handling


UITouch
began
Tap Gesture Recognizer
View
Touch Handling



UITouch

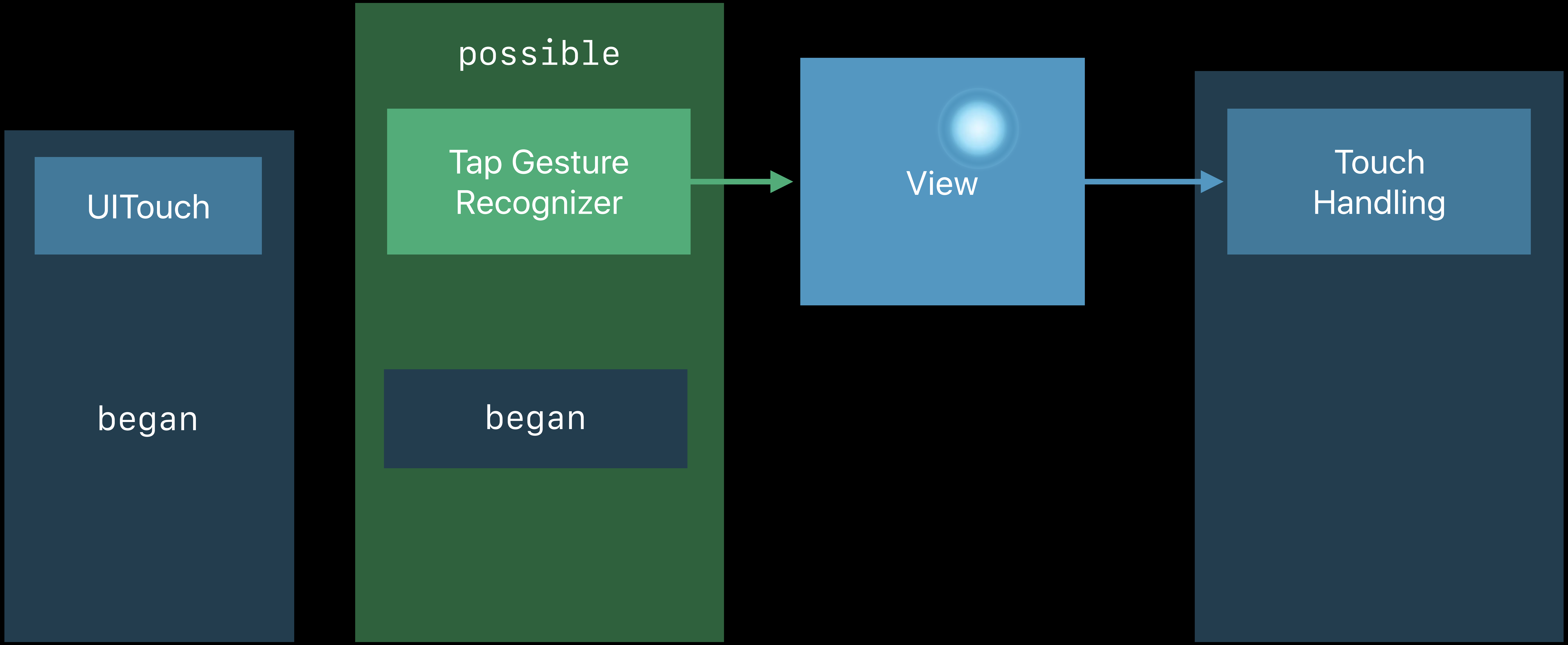
began

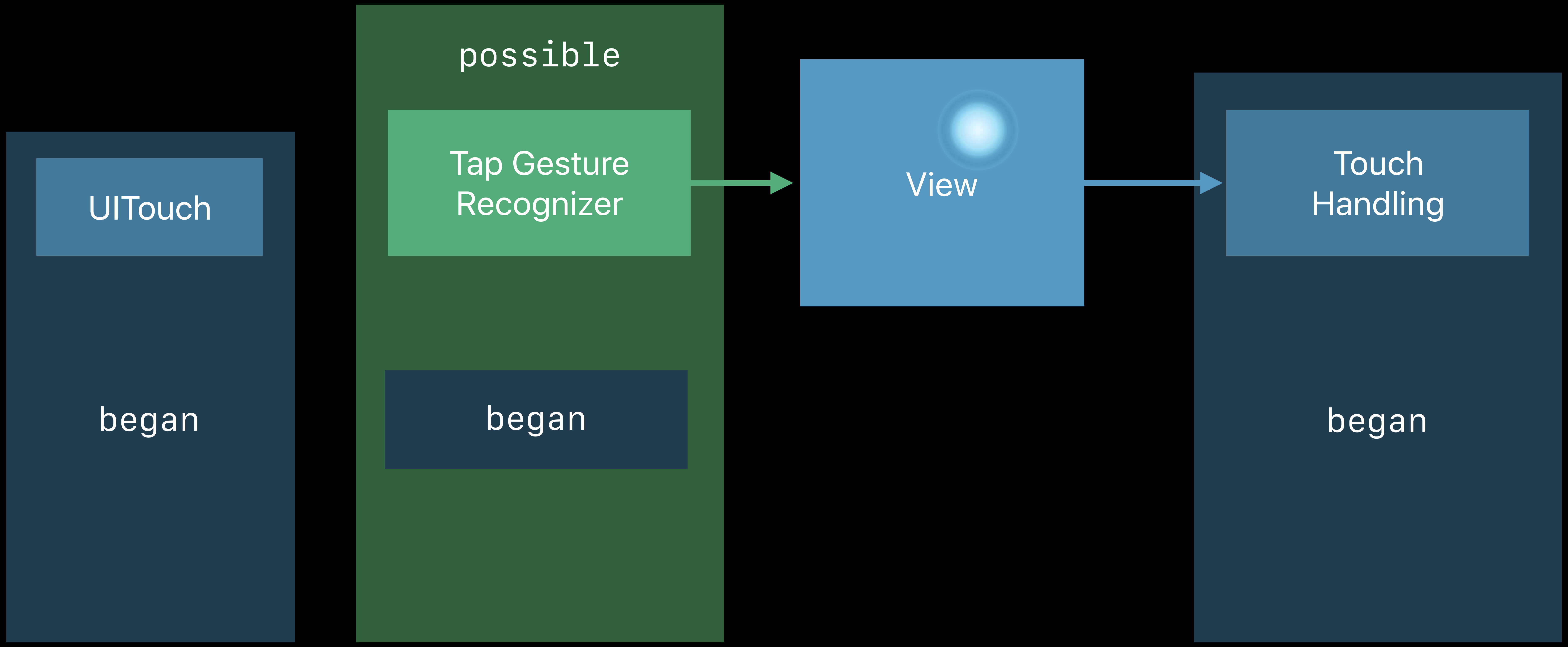
Tap Gesture Recognizer

began

View

Touch Handling


UITouch
began
possible
Tap Gesture Recognizer
began
View
Touch Handling



UITouch

began

possible

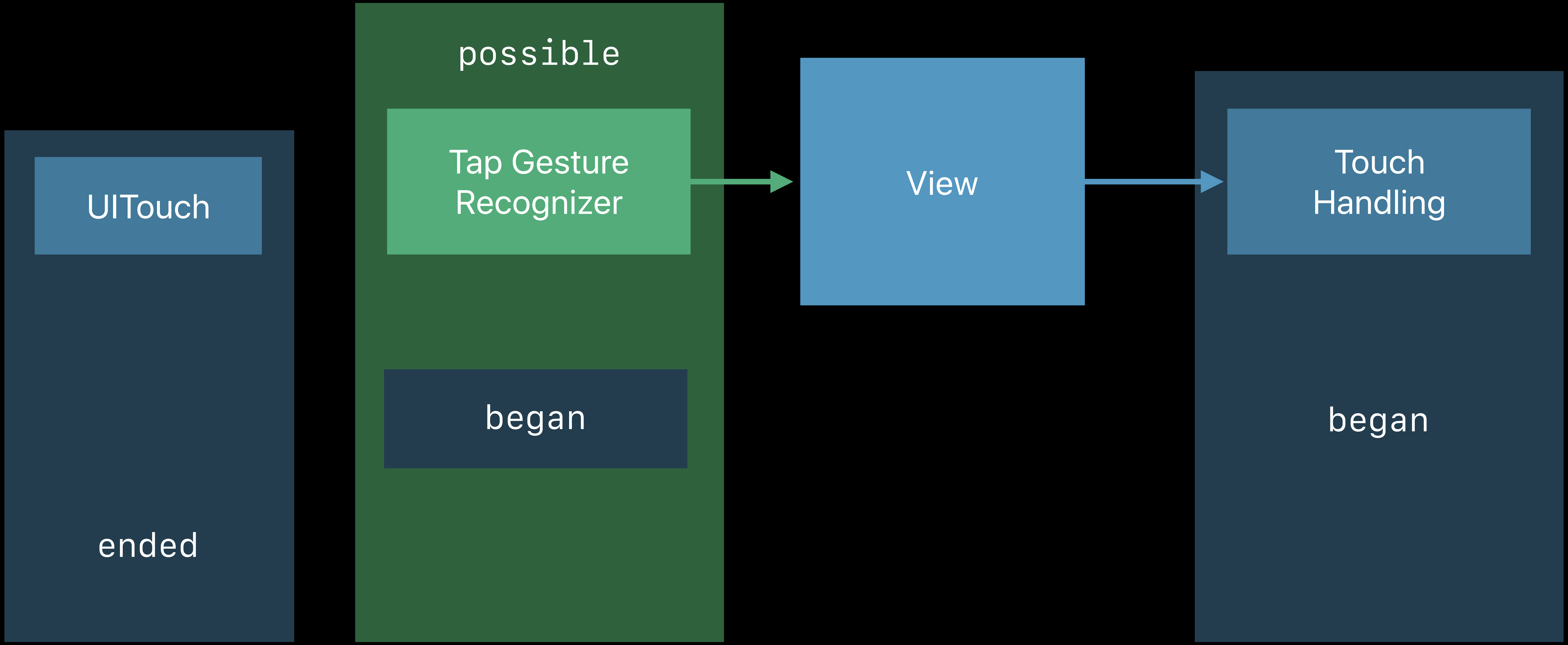
Tap Gesture Recognizer

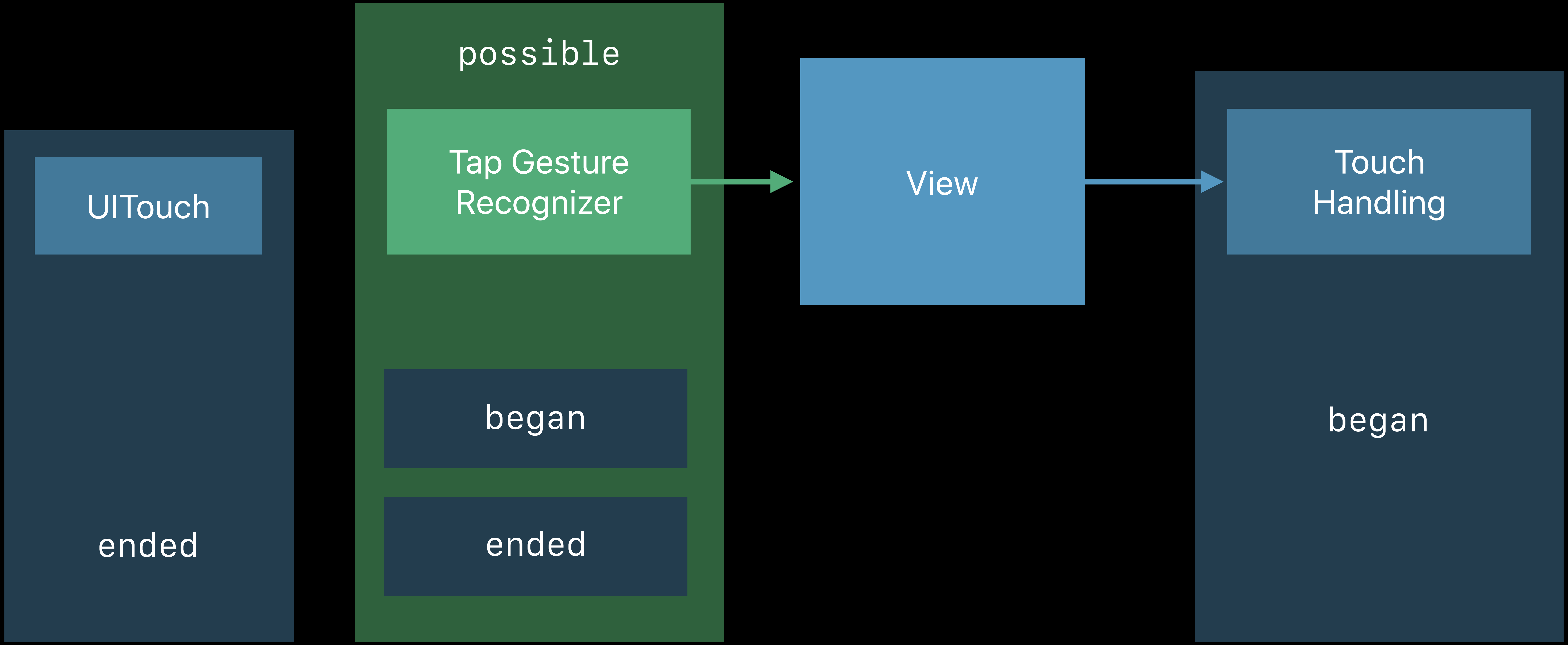
began

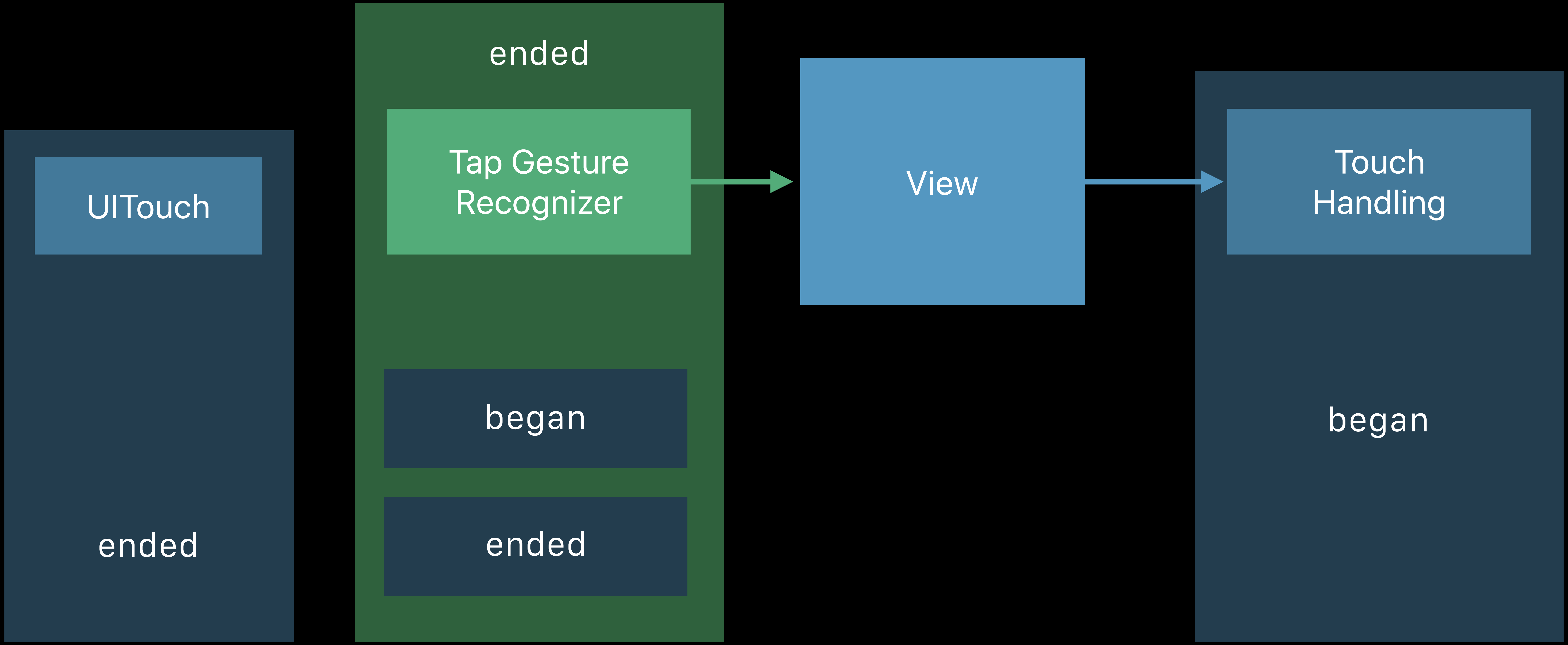
View

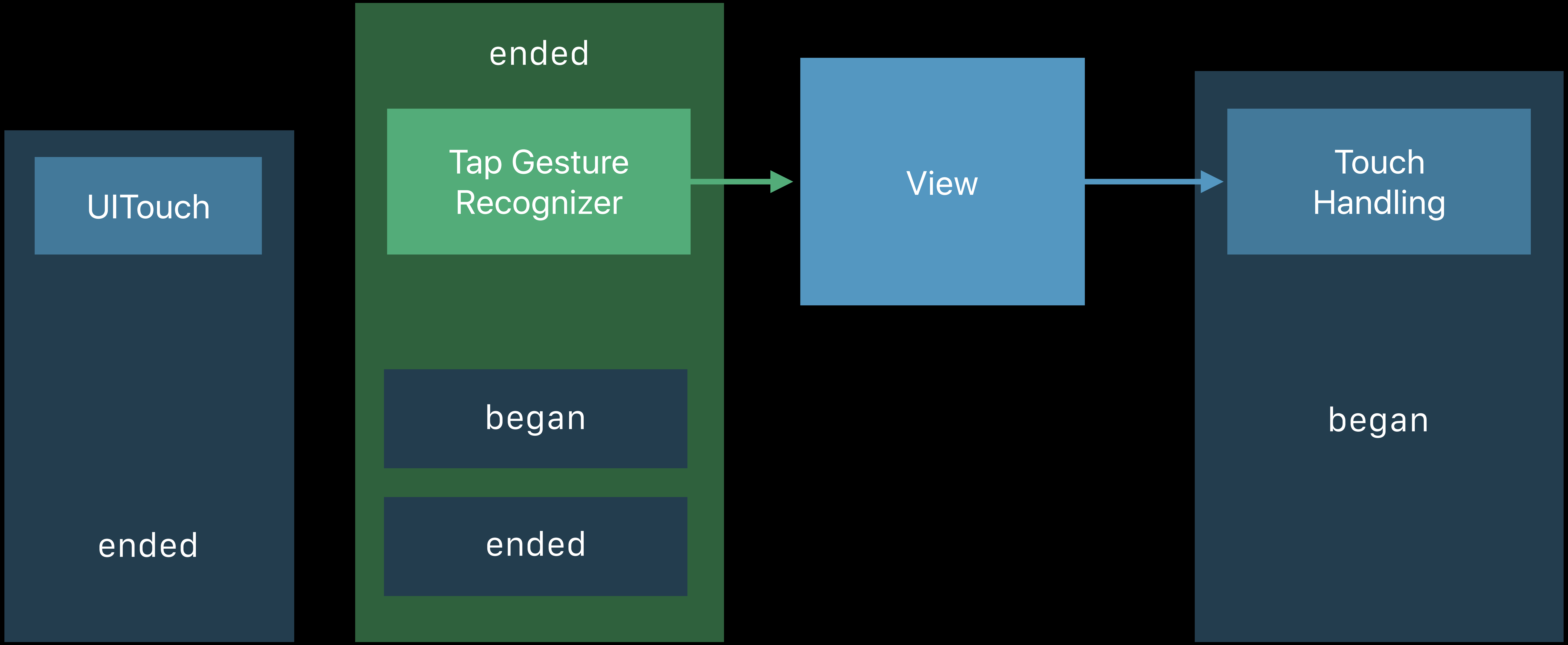
Touch Handling

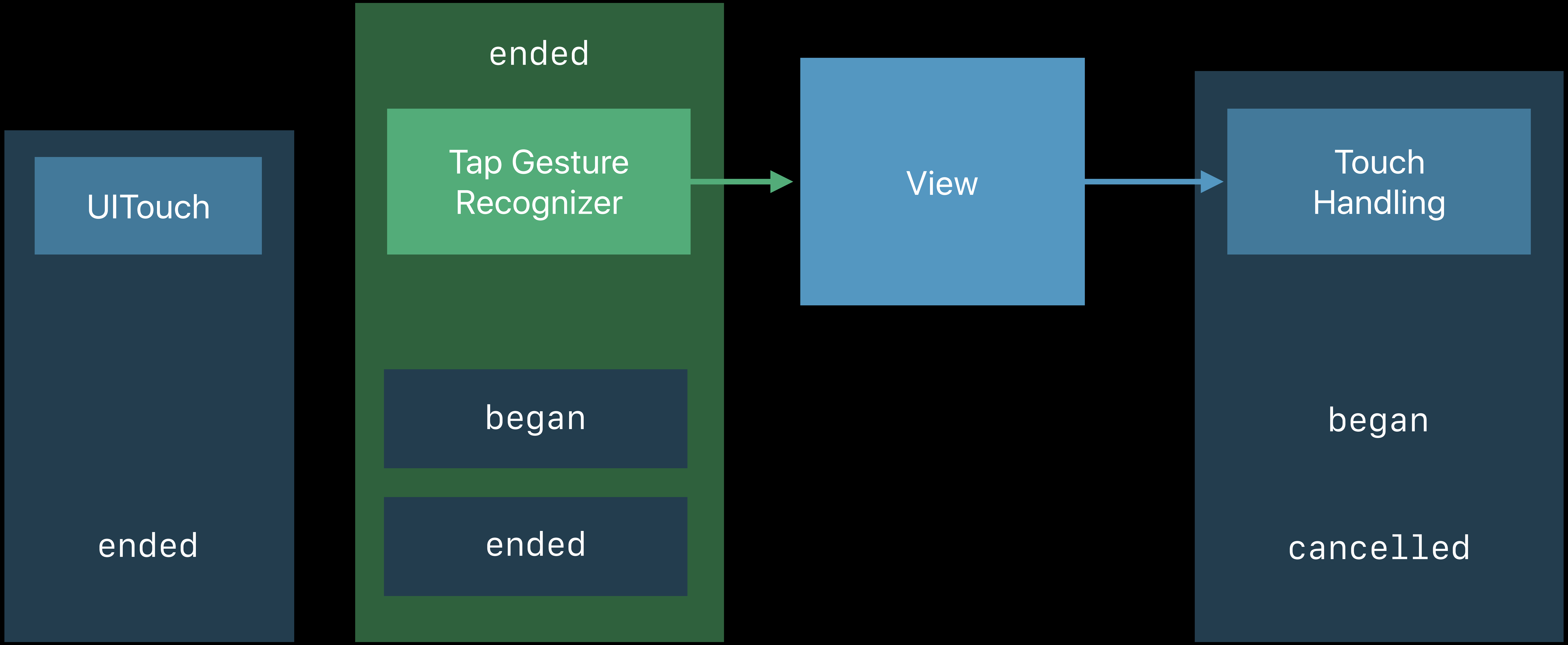
began


UITouch
ended
possible
Tap Gesture Recognizer
began
View
Touch Handling
began


UITouch
ended
possible
Tap Gesture Recognizer
began
ended
View
Touch Handling
began


UITouch
ended
ended
Tap Gesture Recognizer
began
ended
View
Touch Handling
began


UITouch
ended
ended
Tap Gesture Recognizer
began
ended
View
Touch Handling
began



UITouch

ended

ended

Tap Gesture Recognizer

began

ended

View

Touch Handling

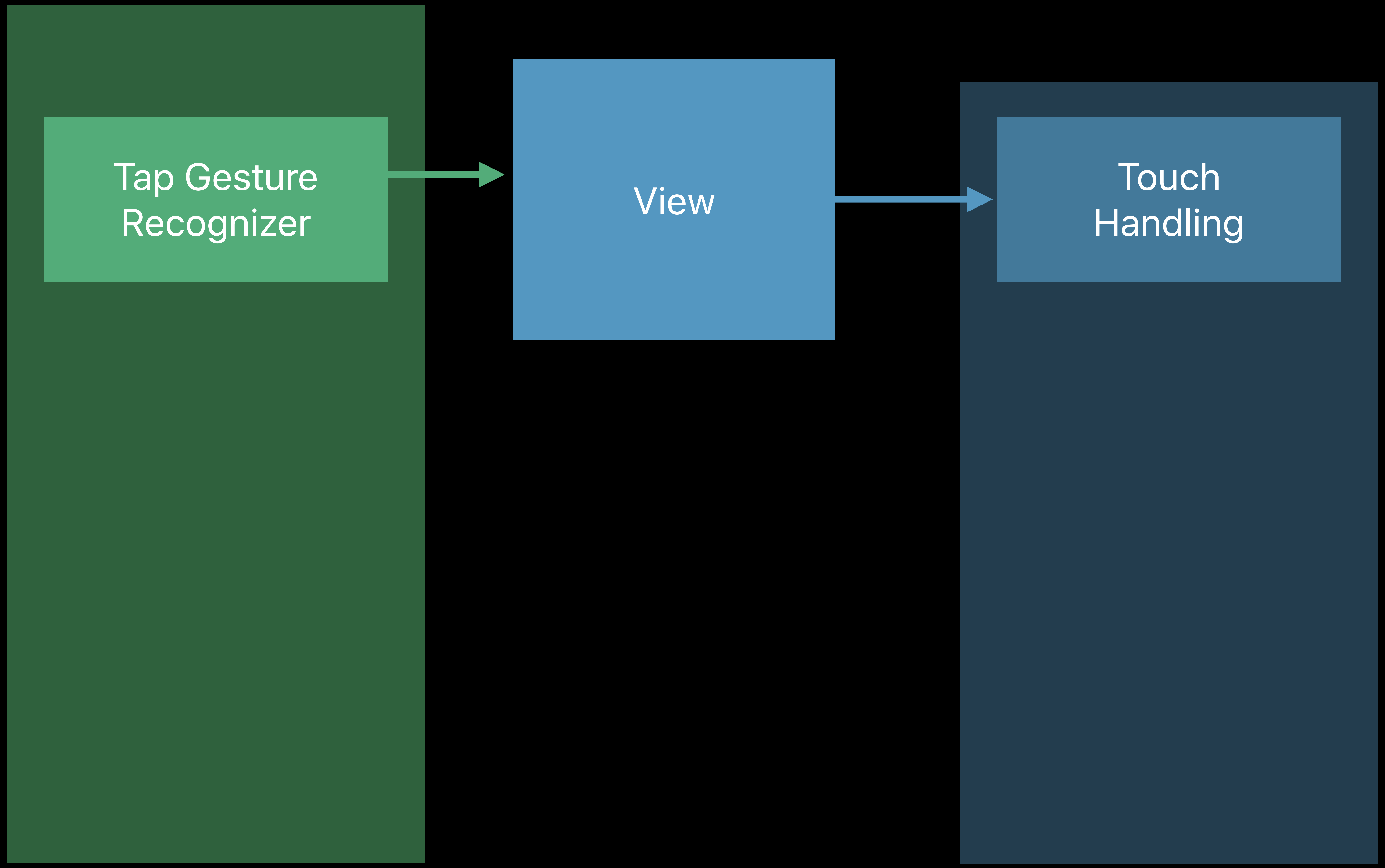
began

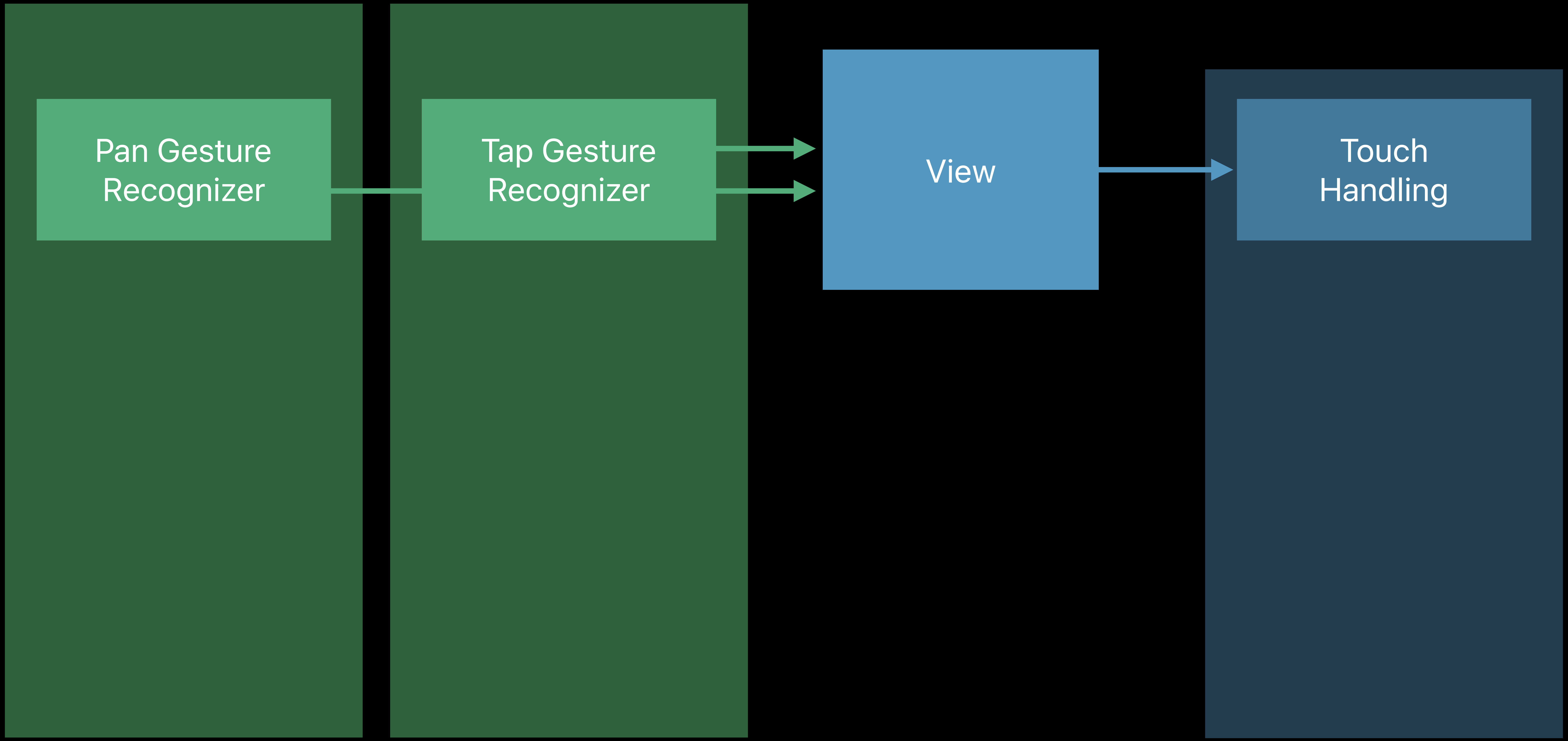
cancelled

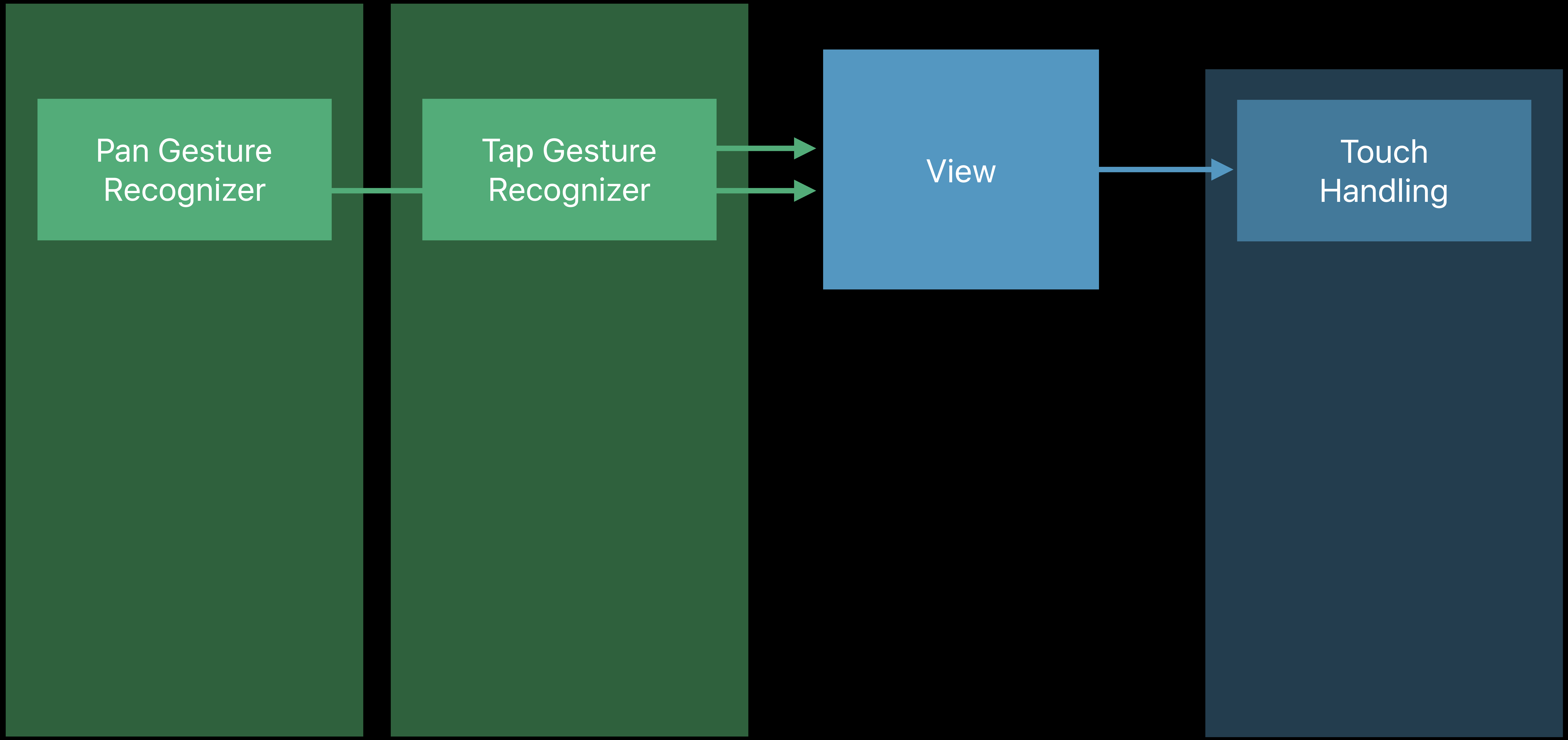
```
// Influencing responder based touch handling

class UIGestureRecognizer : NSObject {

    open var delaysTouchesEnded: Bool // default is true.
    open var cancelsTouchesInView: Bool // default is true.
    open var delaysTouchesBegan: Bool // default is false.
}
```

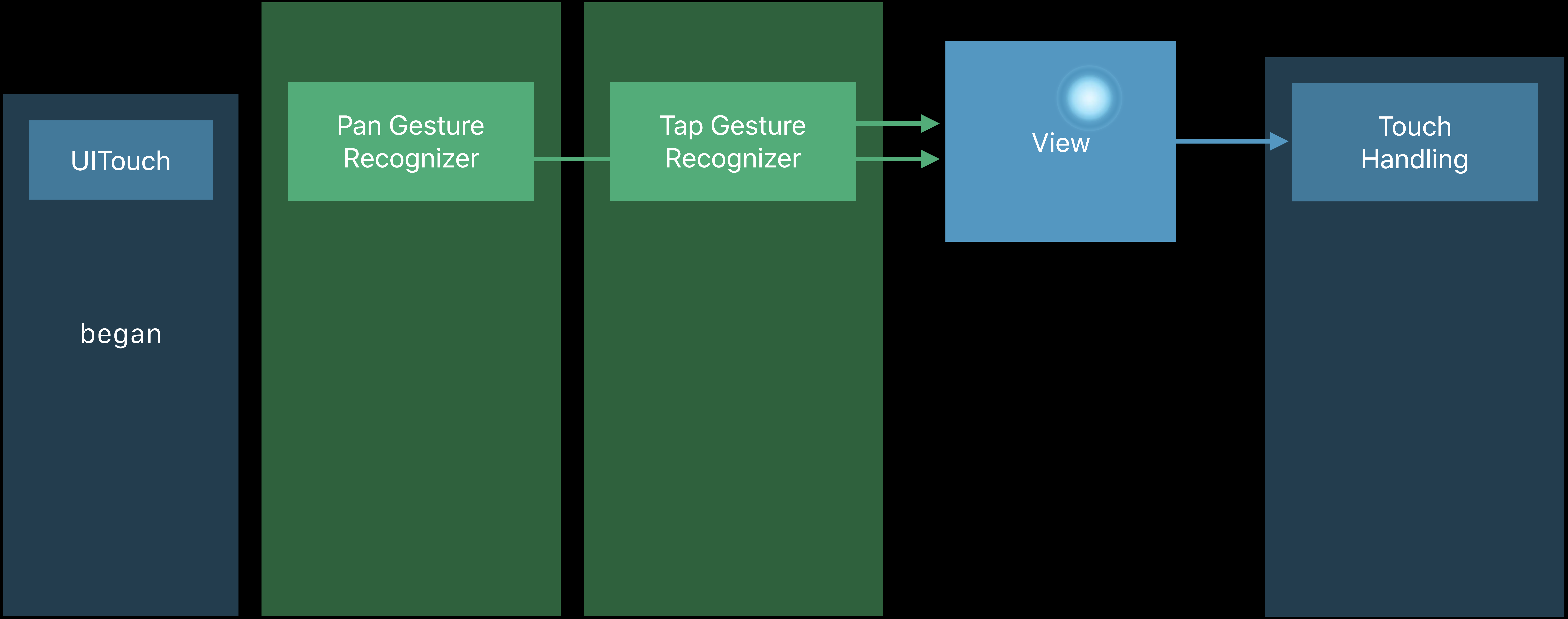
```
// Influencing responder based touch handling

class UIGestureRecognizer : NSObject {

    open var delaysTouchesEnded: Bool // default is true.
    open var cancelsTouchesInView: Bool // default is true.
    open var delaysTouchesBegan: Bool // default is false.
}
```

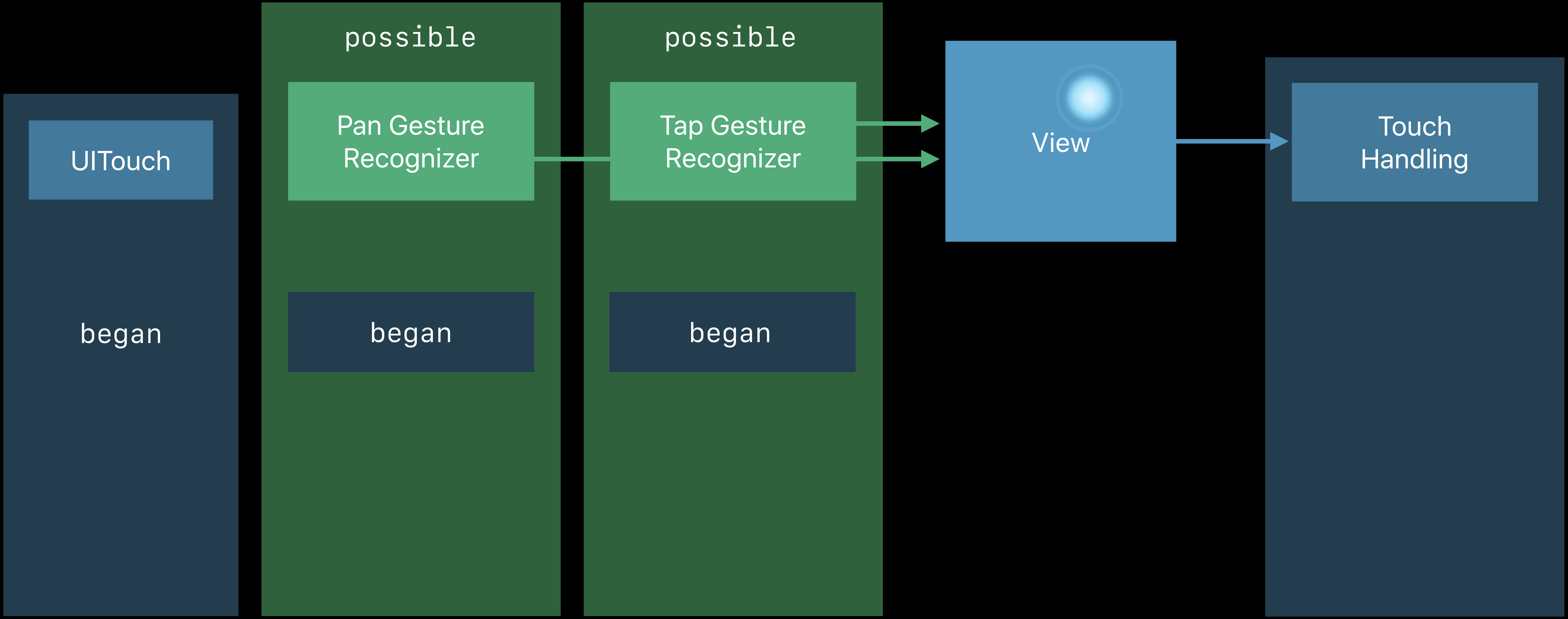
Gesture Recognizers first


Tap Gesture Recognizer
View
Touch Handling


Pan Gesture Recognizer
Tap Gesture Recognizer
View
Touch Handling


Pan Gesture Recognizer
Tap Gesture Recognizer
View
Touch Handling


UITouch
began
Pan Gesture Recognizer
Tap Gesture Recognizer
View
Touch Handling



UITouch

began

possible

Pan Gesture Recognizer

began

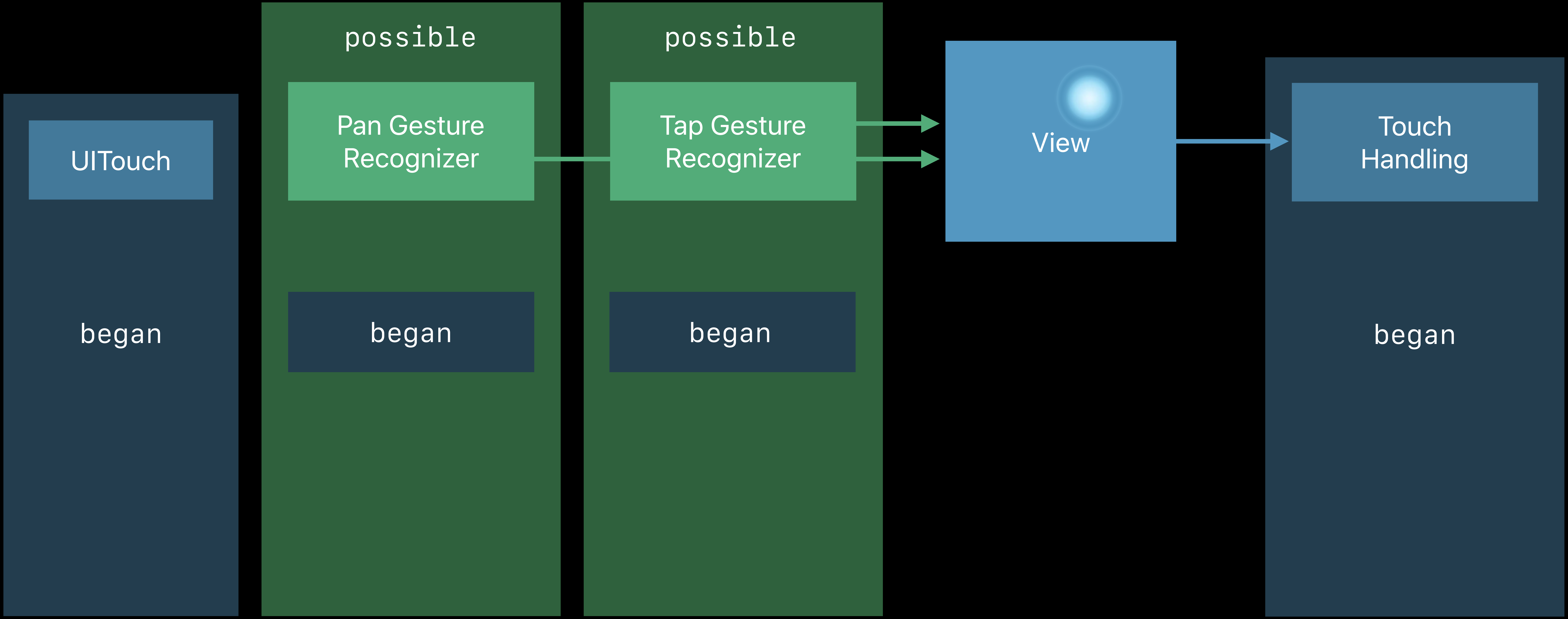
possible

Tap Gesture Recognizer

began

View

Touch Handling



UITouch

began

possible

Pan Gesture Recognizer

began

possible

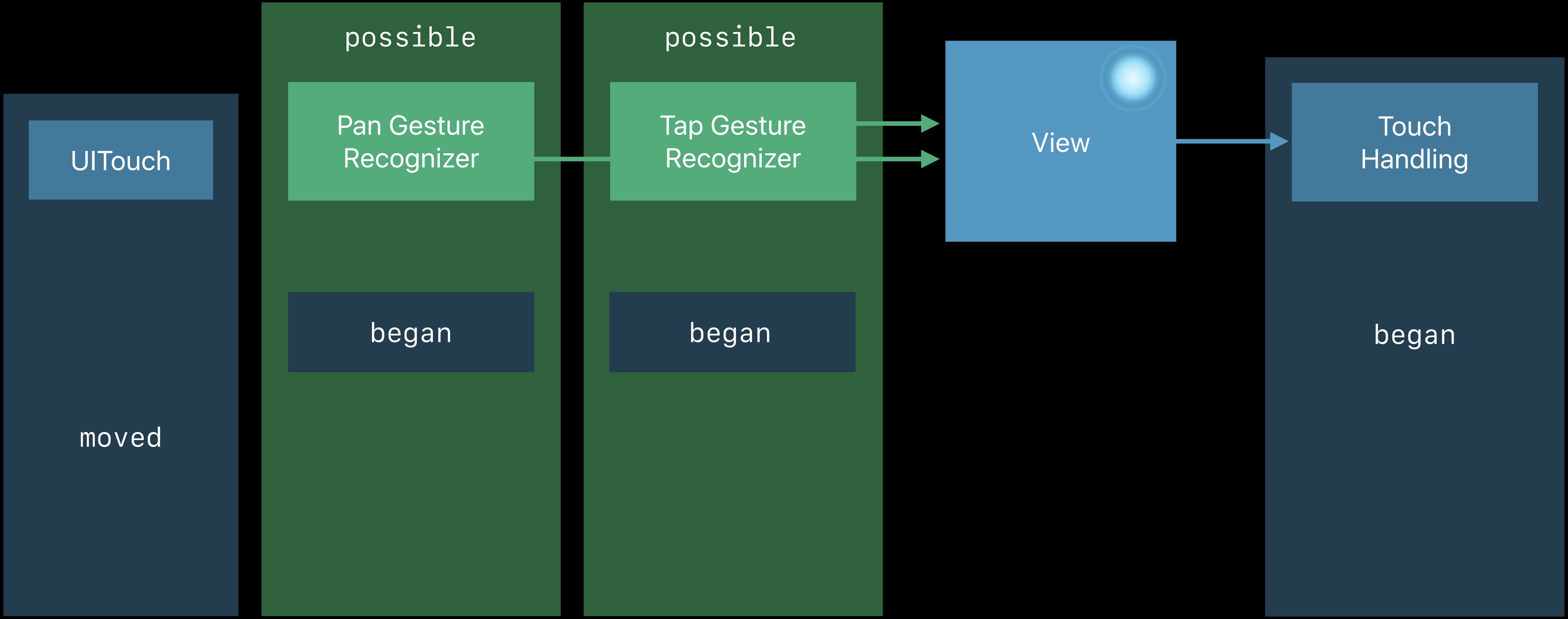
Tap Gesture Recognizer

began

View

Touch Handling

began



UITouch

moved

possible

Pan Gesture Recognizer

began

possible

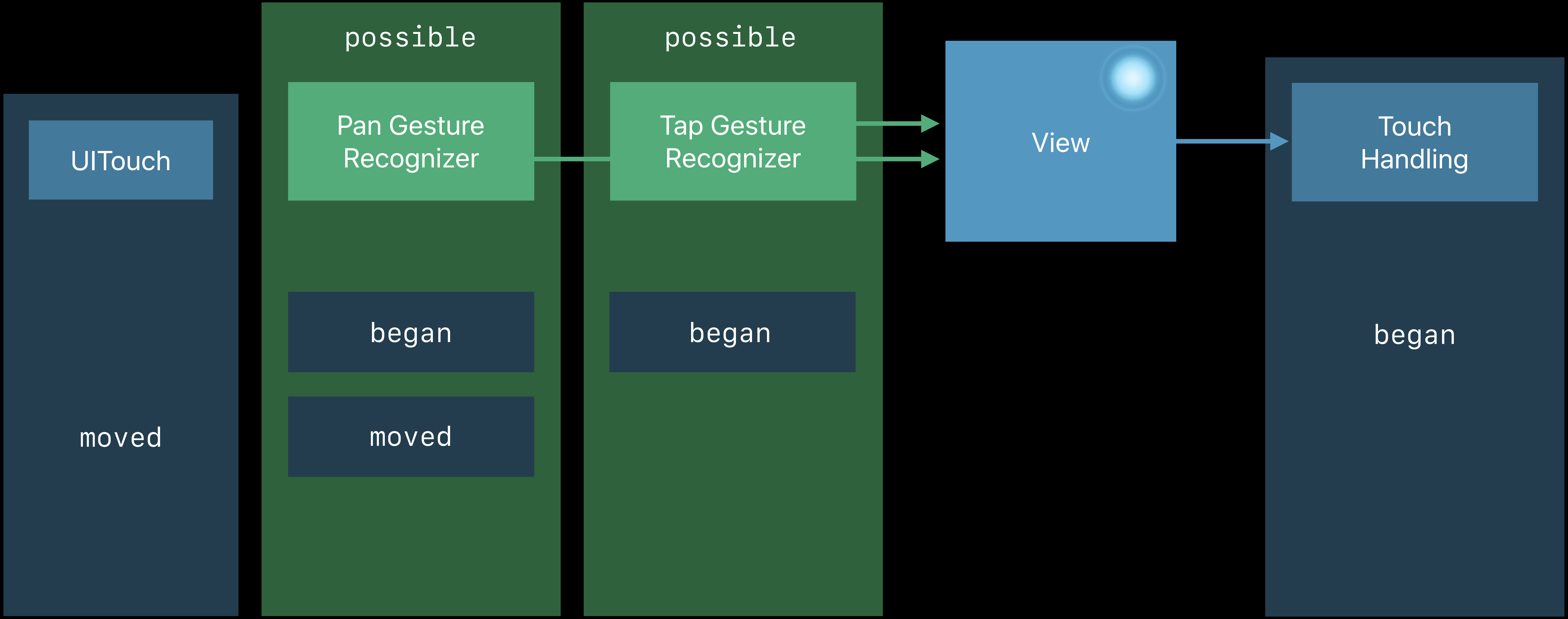
Tap Gesture Recognizer

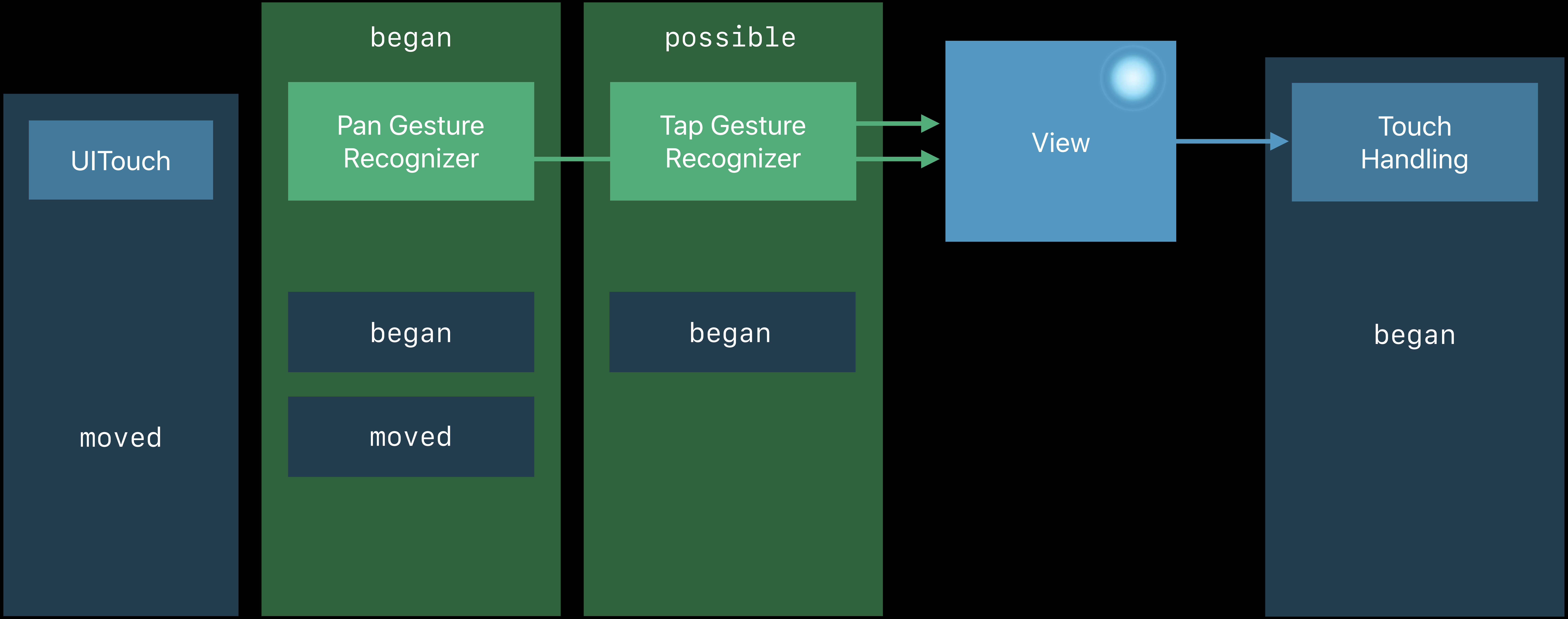
began

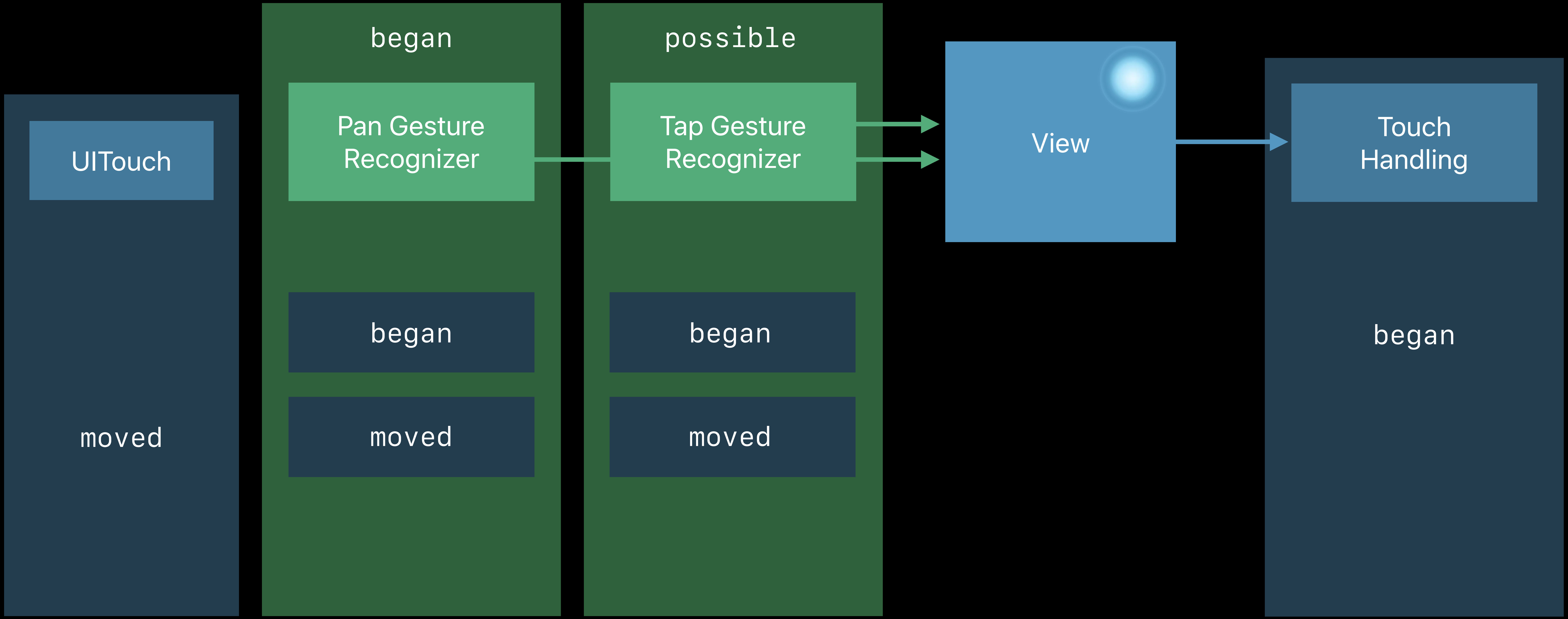
View

Touch Handling

began


possible
possible
UITouch
Pan Gesture Recognizer
Tap Gesture Recognizer
View
Touch Handling
began
began
began
moved
moved


UITouch
moved
began
Pan Gesture Recognizer
began
moved
possible
Tap Gesture Recognizer
began
View
Touch Handling
began



UITouch

moved

began

Pan Gesture Recognizer

began

moved

possible

Tap Gesture Recognizer

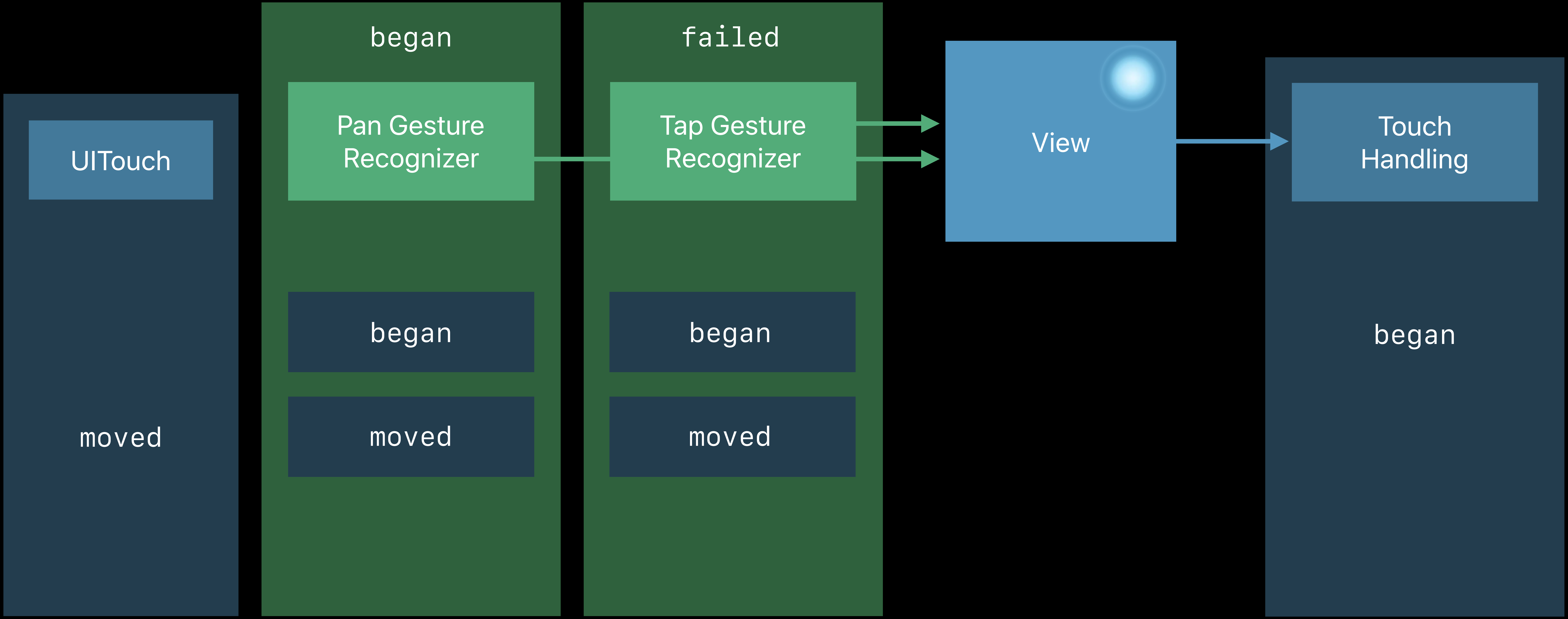
began

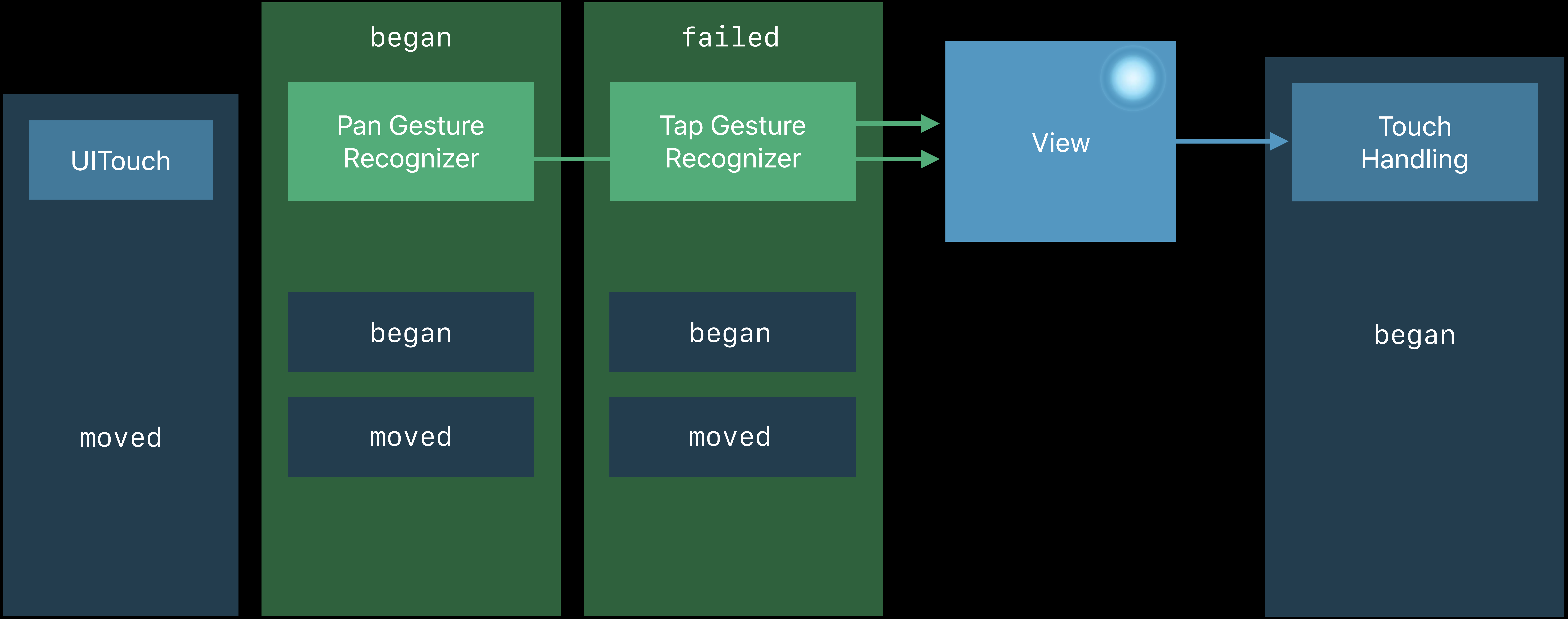
moved

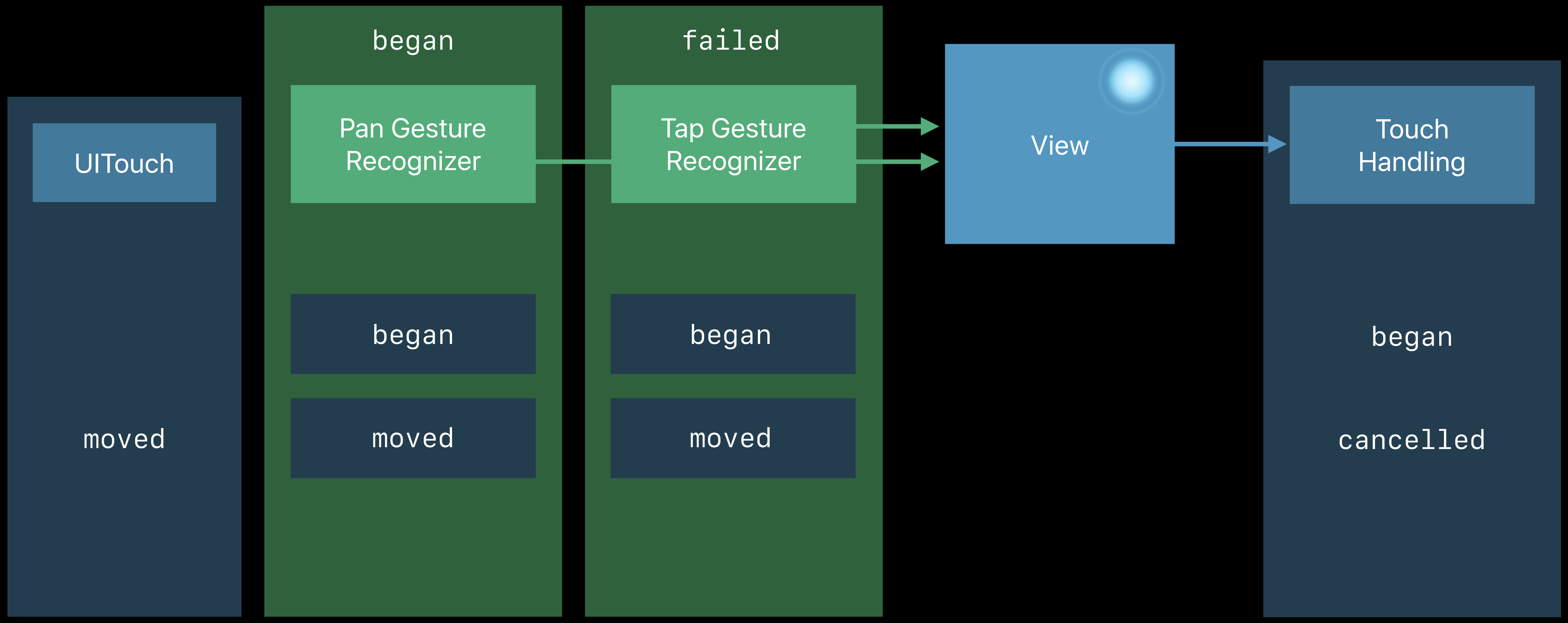
View

Touch Handling

began


UITouch
moved
began
Pan Gesture Recognizer
began
moved
failed
Tap Gesture Recognizer
began
moved
View
Touch Handling
began


UITouch
moved
began
Pan Gesture Recognizer
began
moved
failed
Tap Gesture Recognizer
began
moved
View
Touch Handling
began



UITouch

moved

began

Pan Gesture Recognizer

began

moved

failed

Tap Gesture Recognizer

began

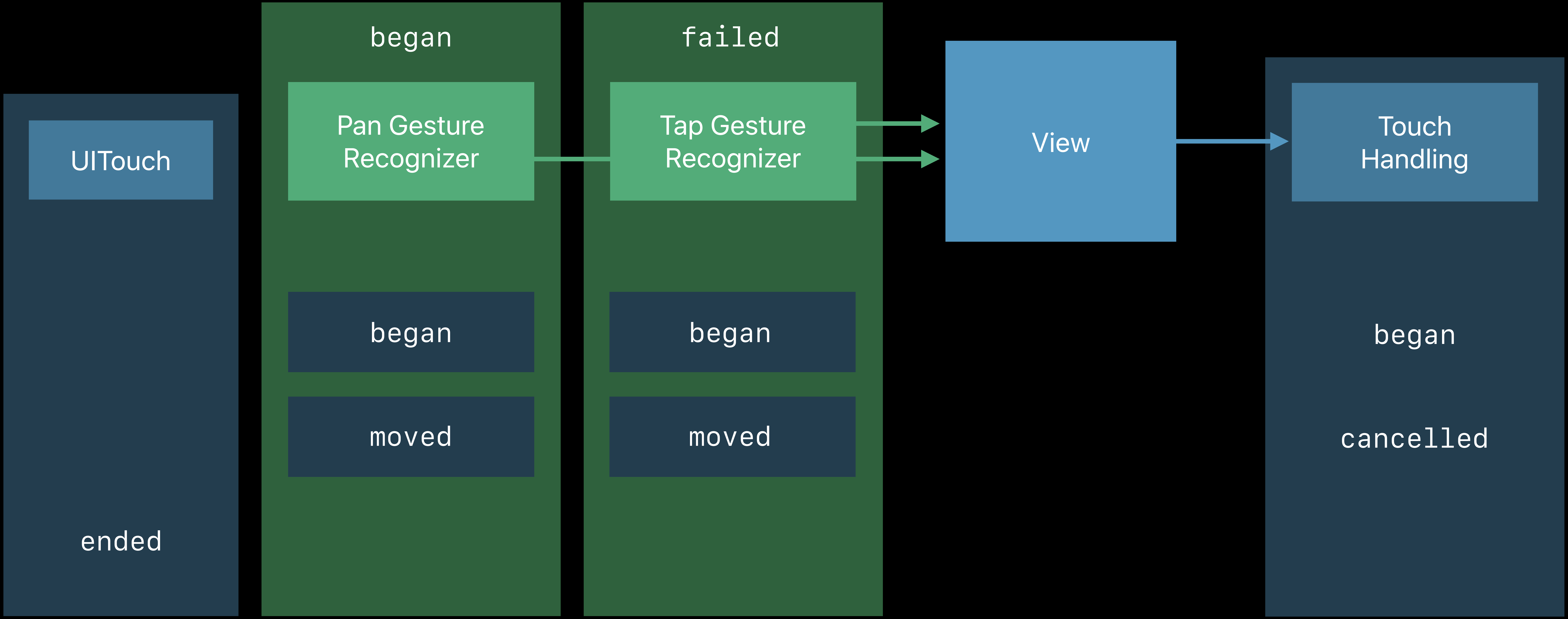
moved

View

Touch Handling

began

cancelled



UITouch

ended

began

Pan Gesture Recognizer

began

moved

failed

Tap Gesture Recognizer

began

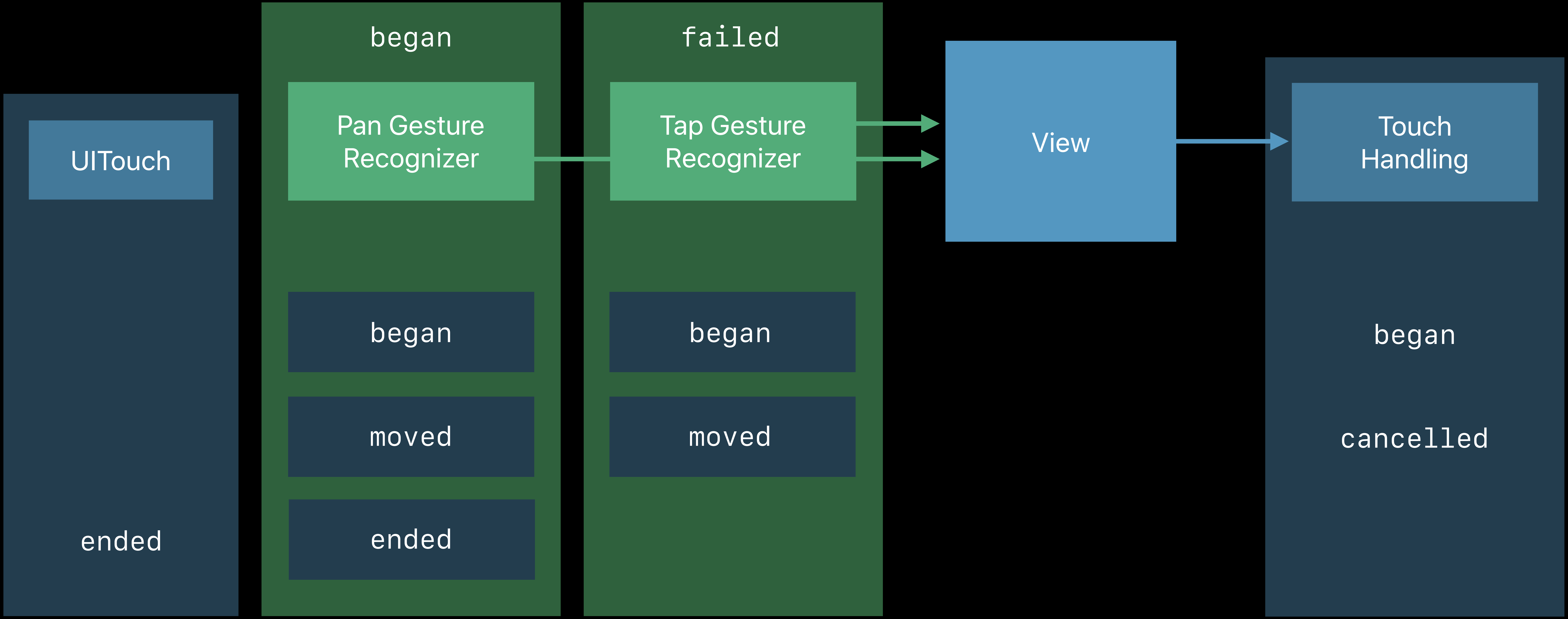
moved

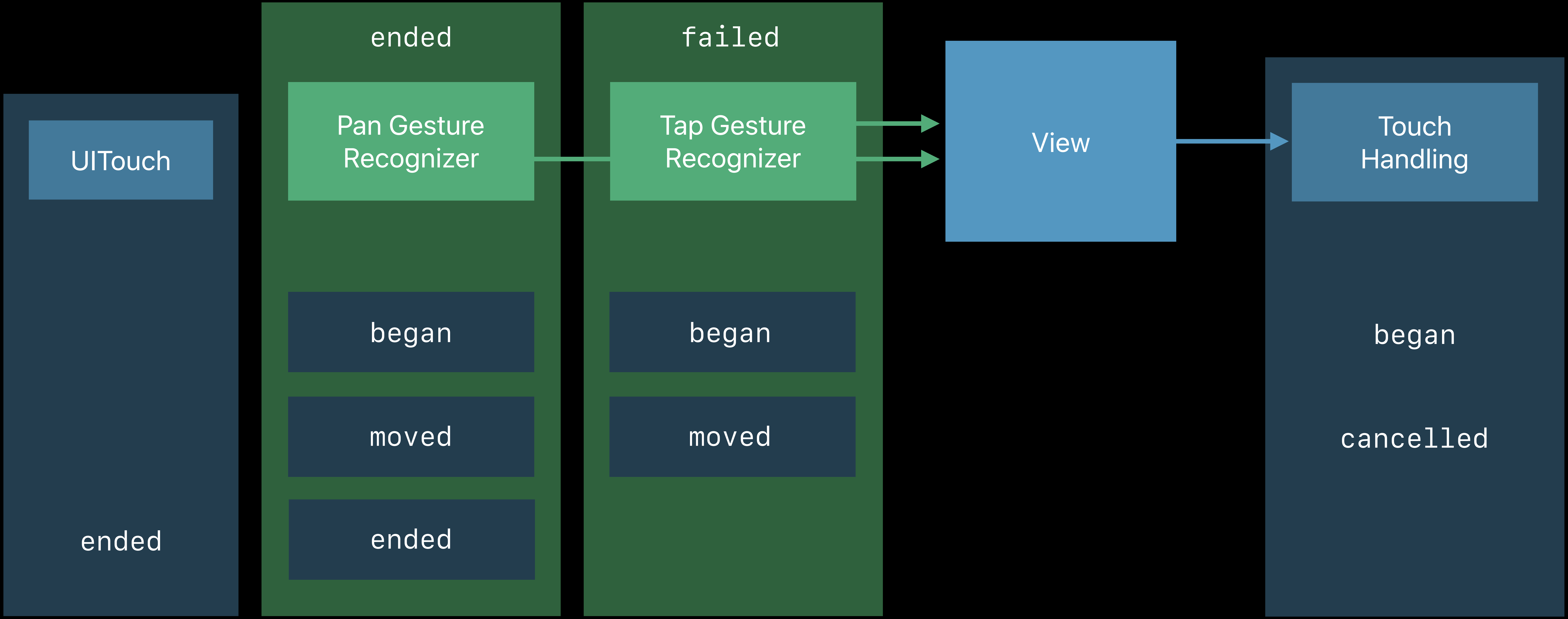
View

Touch Handling

began

cancelled


UITouch
ended
began
Pan Gesture Recognizer
began
moved
ended
failed
Tap Gesture Recognizer
began
moved
View
Touch Handling
began
cancelled



UITouch

ended

ended

Pan Gesture Recognizer

began

moved

ended

failed

Tap Gesture Recognizer

began

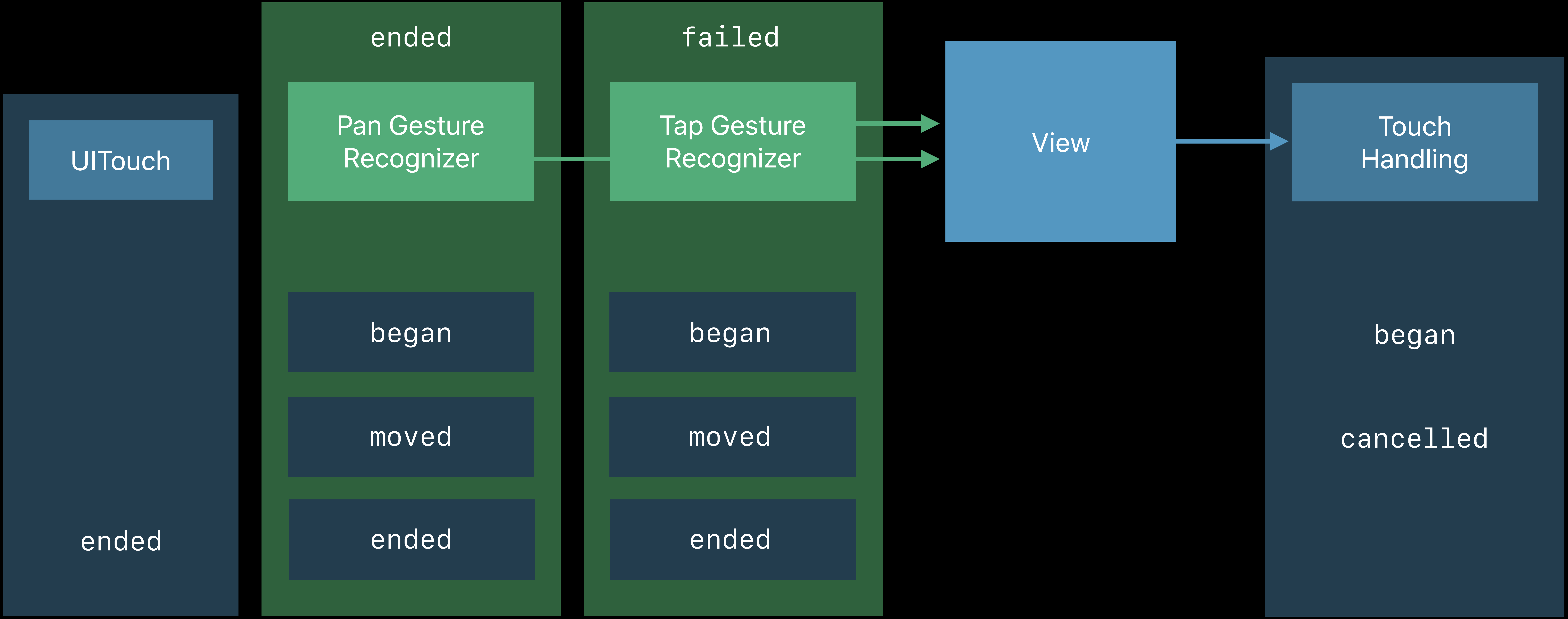
moved

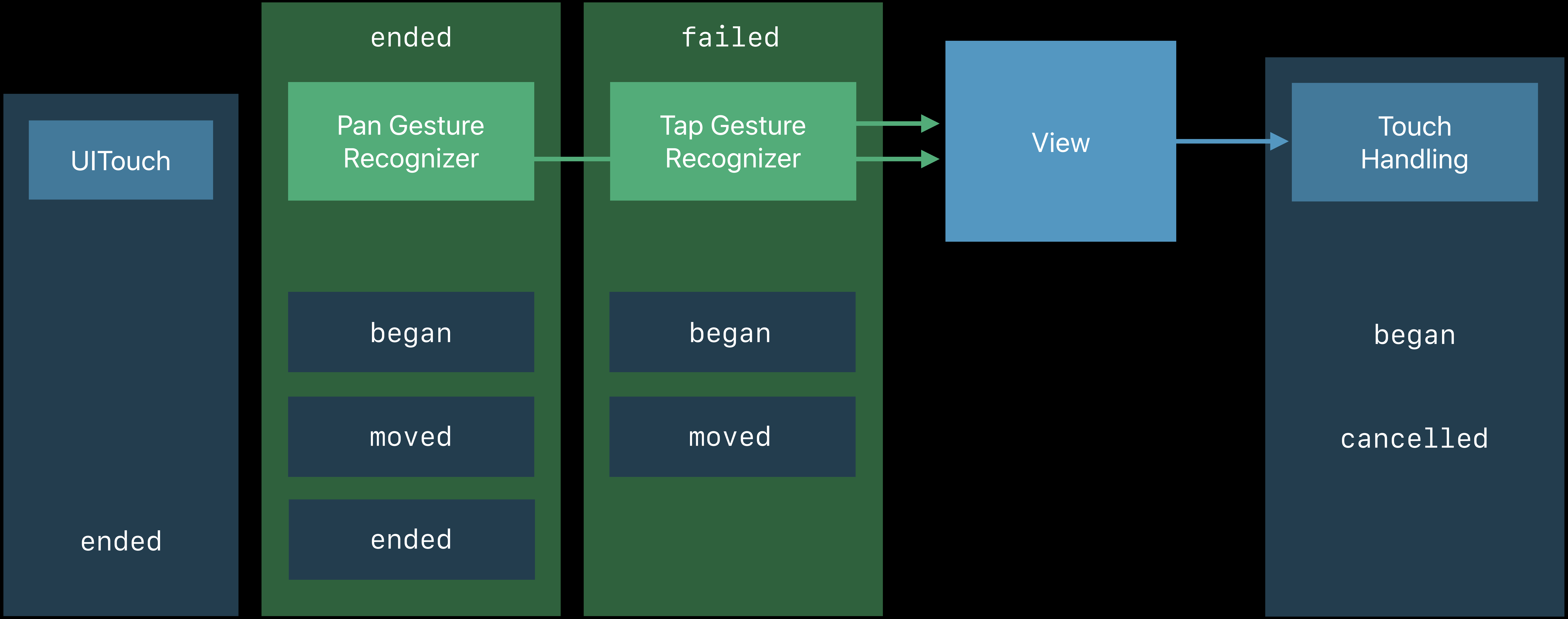
View

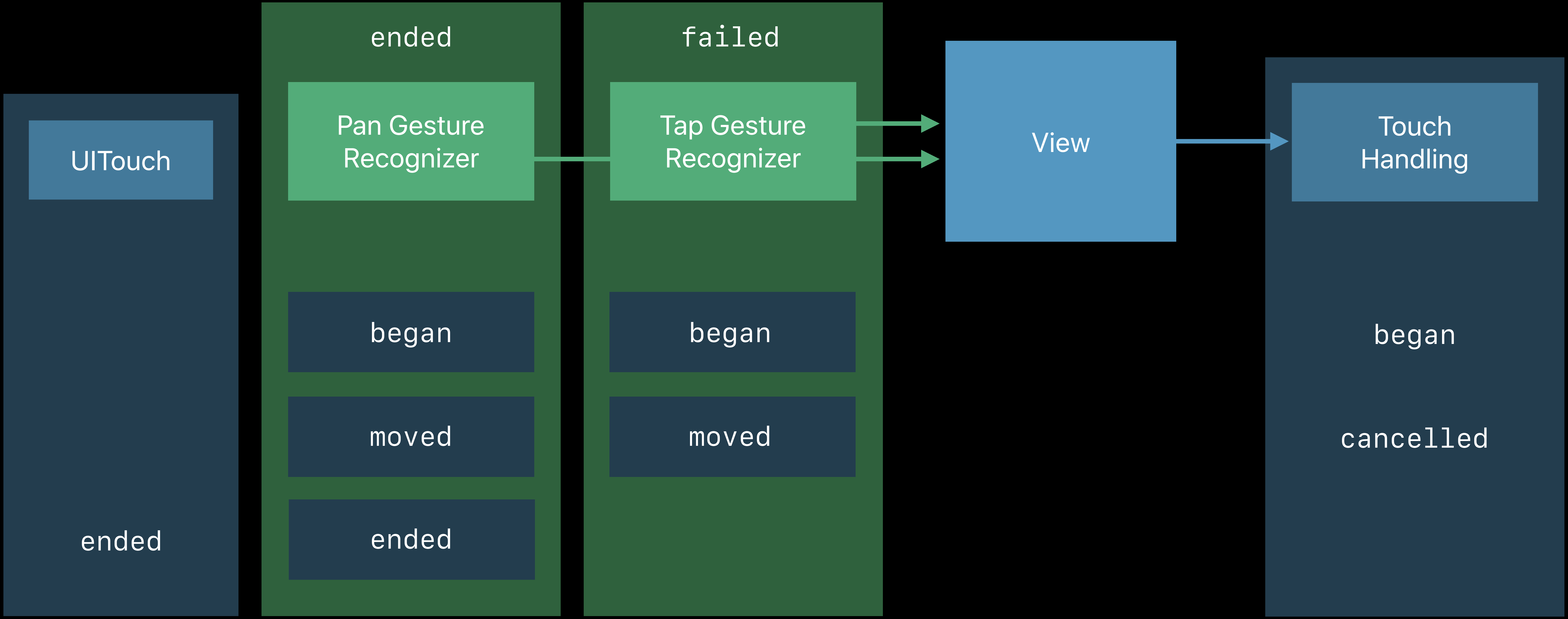
Touch Handling

began

cancelled


UITouch
ended
ended
Pan Gesture Recognizer
began
moved
ended
failed
Tap Gesture Recognizer
began
moved
ended
View
Touch Handling
began
cancelled


UITouch
ended
ended
Pan Gesture Recognizer
began
moved
ended
failed
Tap Gesture Recognizer
began
moved
View
Touch Handling
began
cancelled



UITouch

ended

ended

Pan Gesture Recognizer

began

moved

ended

failed

Tap Gesture Recognizer

began

moved

View

Touch Handling

began

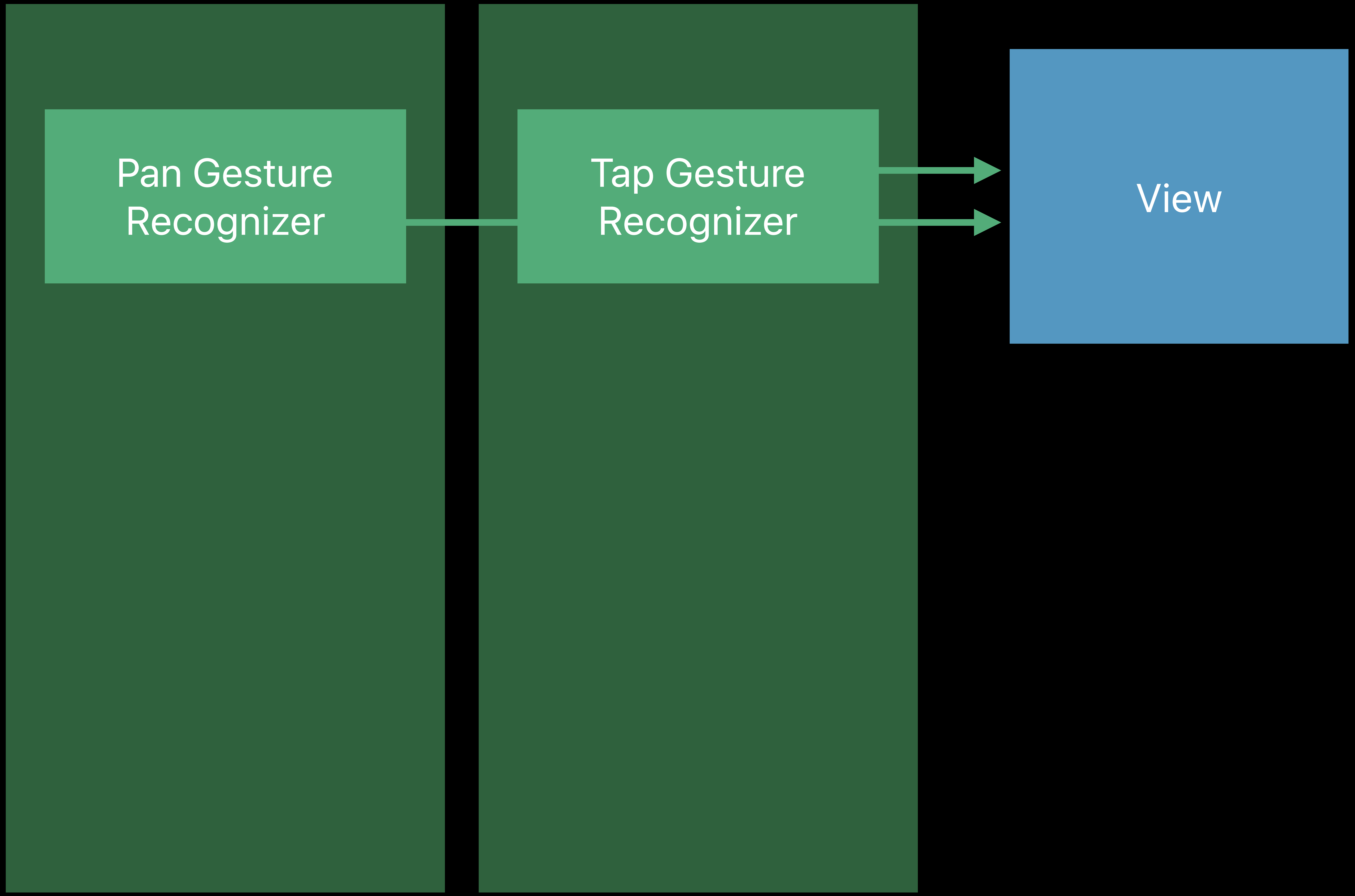
cancelled

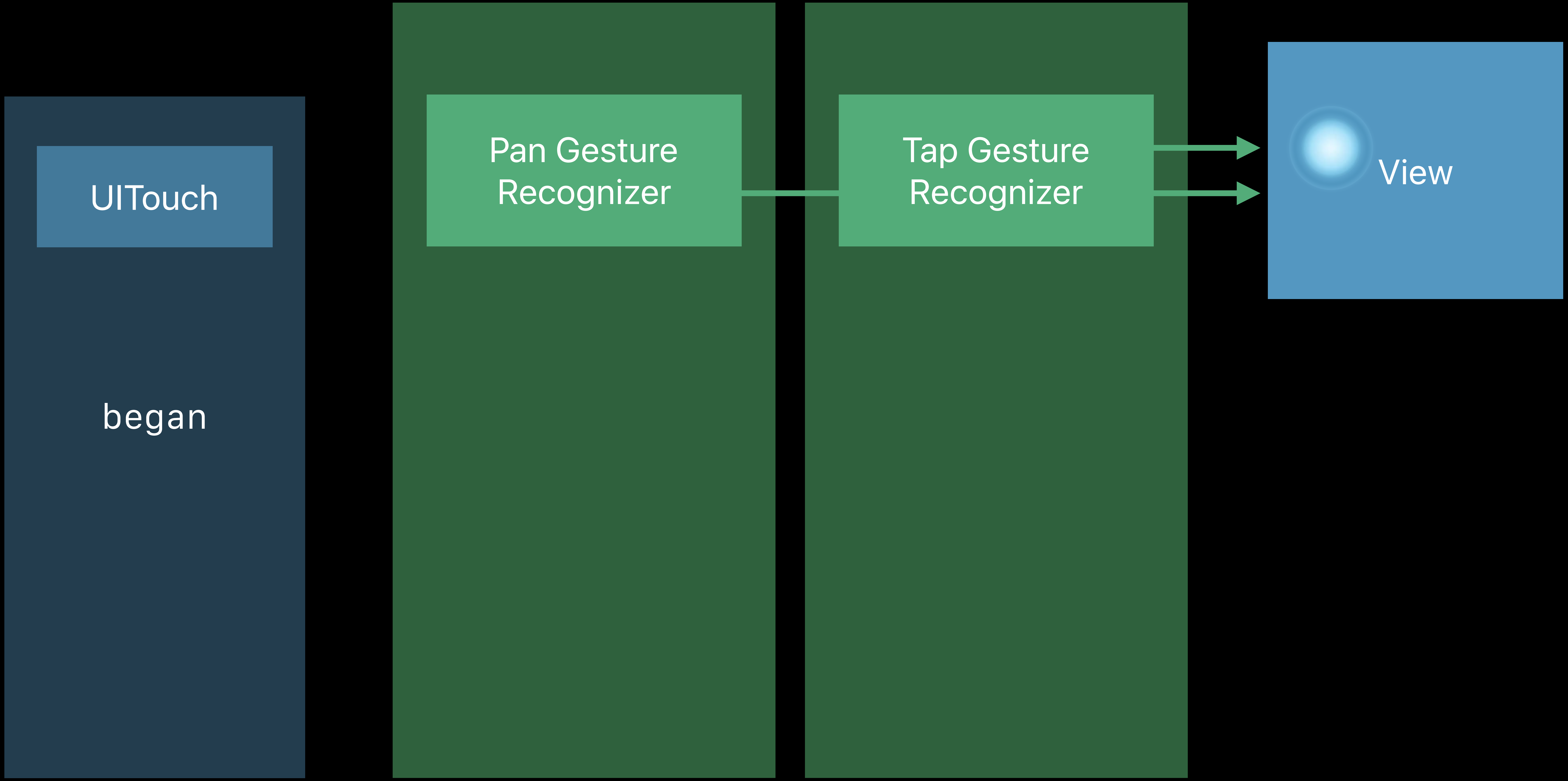
Exclusion

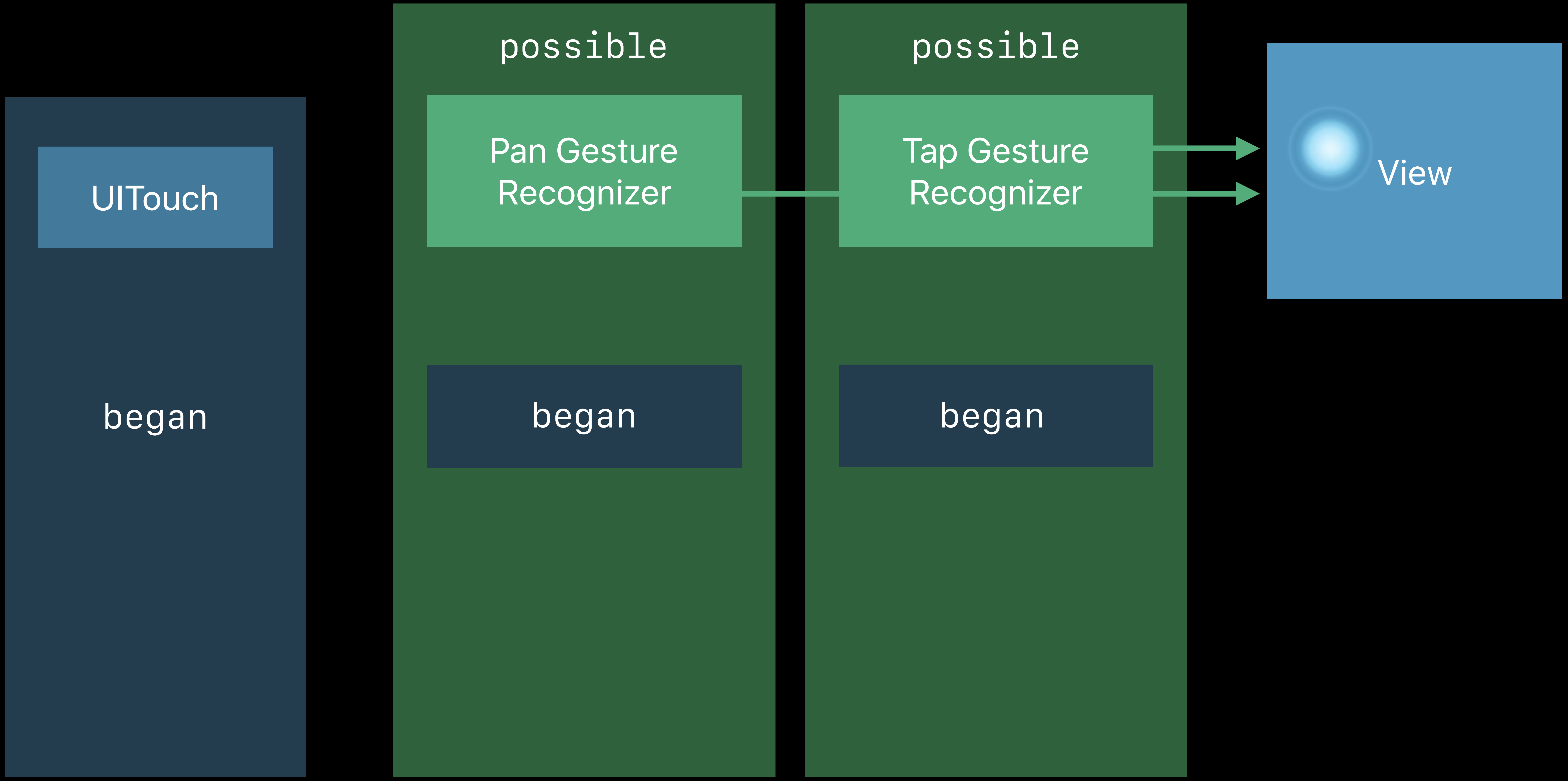
```
// Influencing exclusion

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
      shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

open class UIGestureRecognizer : NSObject {

    open func canPrevent(_ preventedGestureRecognizer: UIGestureRecognizer) -> Bool
    open func canBePrevented(by preventingGestureRecognizer: UIGestureRecognizer) -> Bool
}
```

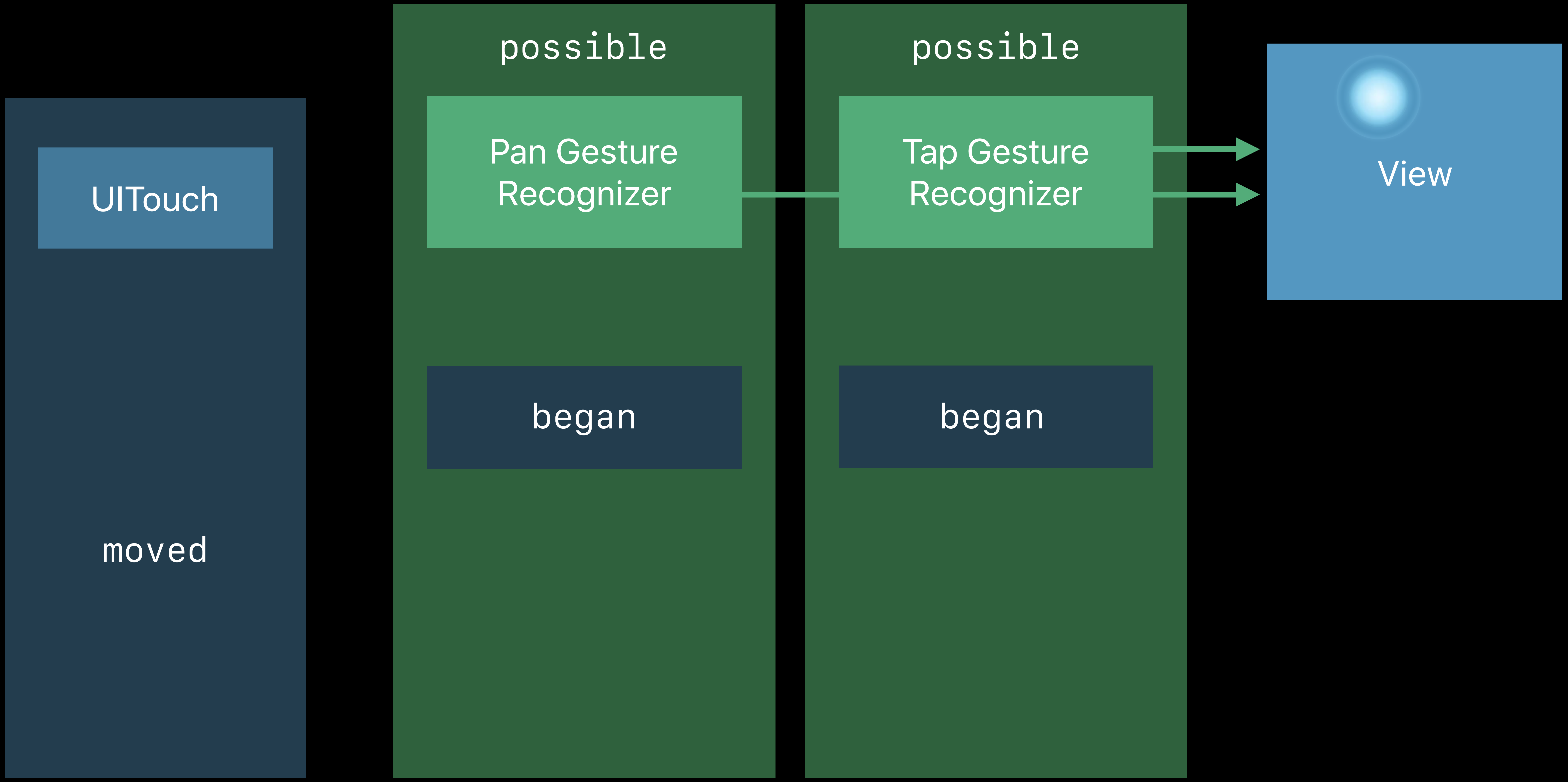
```
// Influencing exclusion

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
      shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

open class UIGestureRecognizer : NSObject {

    open func canPrevent(_ preventedGestureRecognizer: UIGestureRecognizer) -> Bool
    open func canBePrevented(by preventingGestureRecognizer: UIGestureRecognizer) -> Bool
}
```

```
// Influencing exclusion

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
      shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

open class UIGestureRecognizer : NSObject {

    open func canPrevent(_ preventedGestureRecognizer: UIGestureRecognizer) -> Bool
    open func canBePrevented(by preventingGestureRecognizer: UIGestureRecognizer) -> Bool
}
```

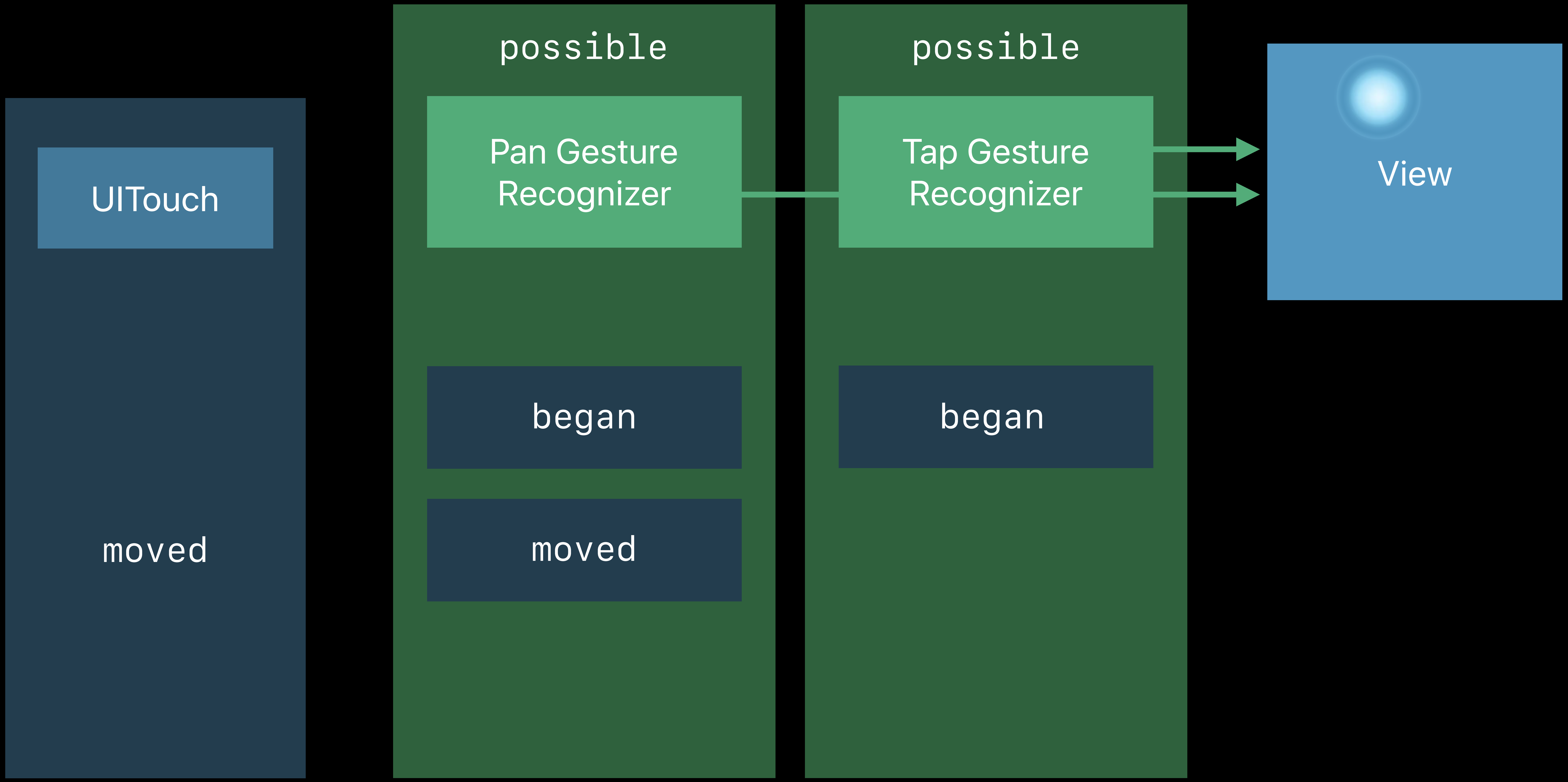

Pan Gesture Recognizer
Tap Gesture Recognizer
View

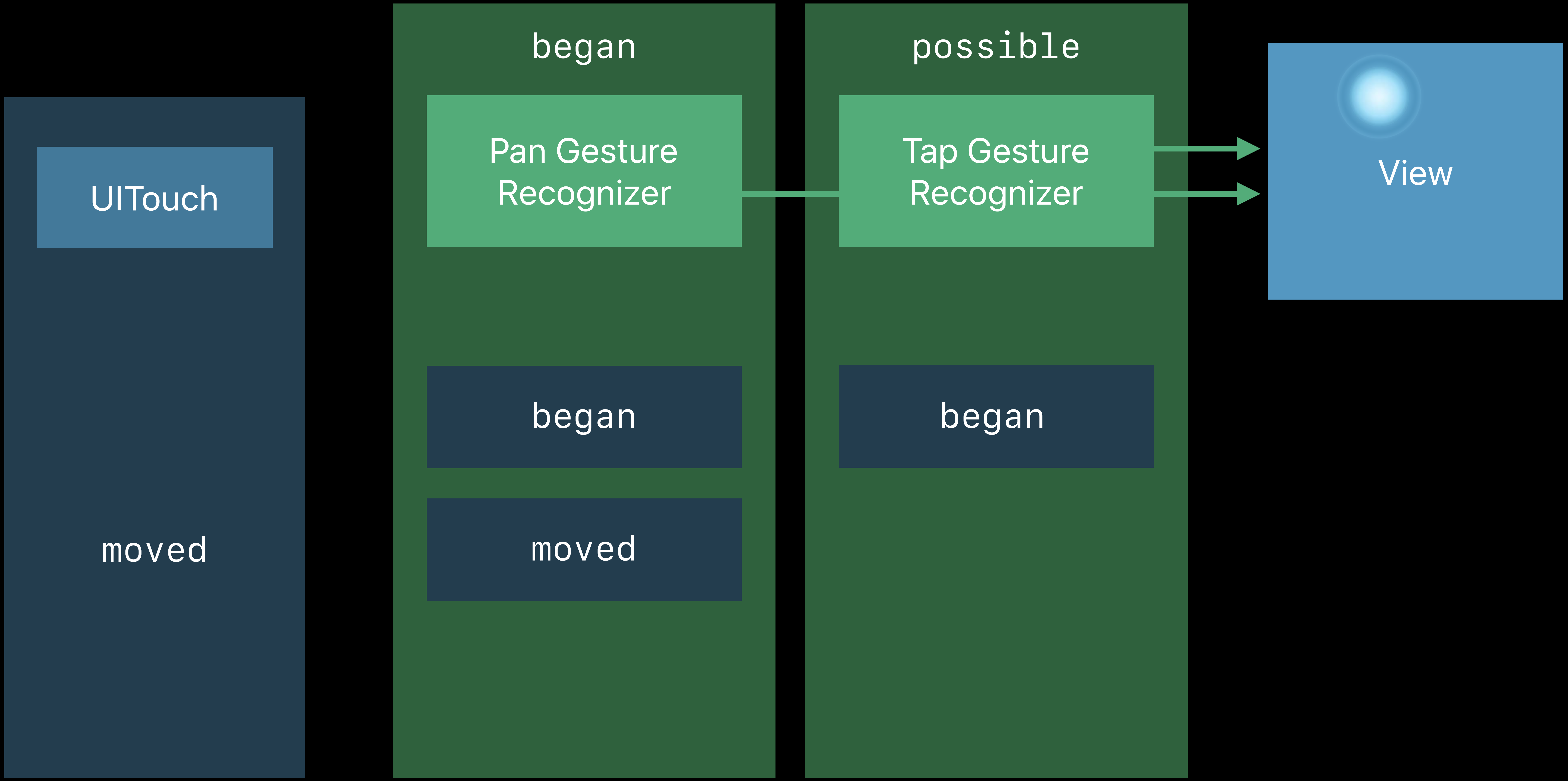

UITouch
began
Pan Gesture Recognizer
Tap Gesture Recognizer
View

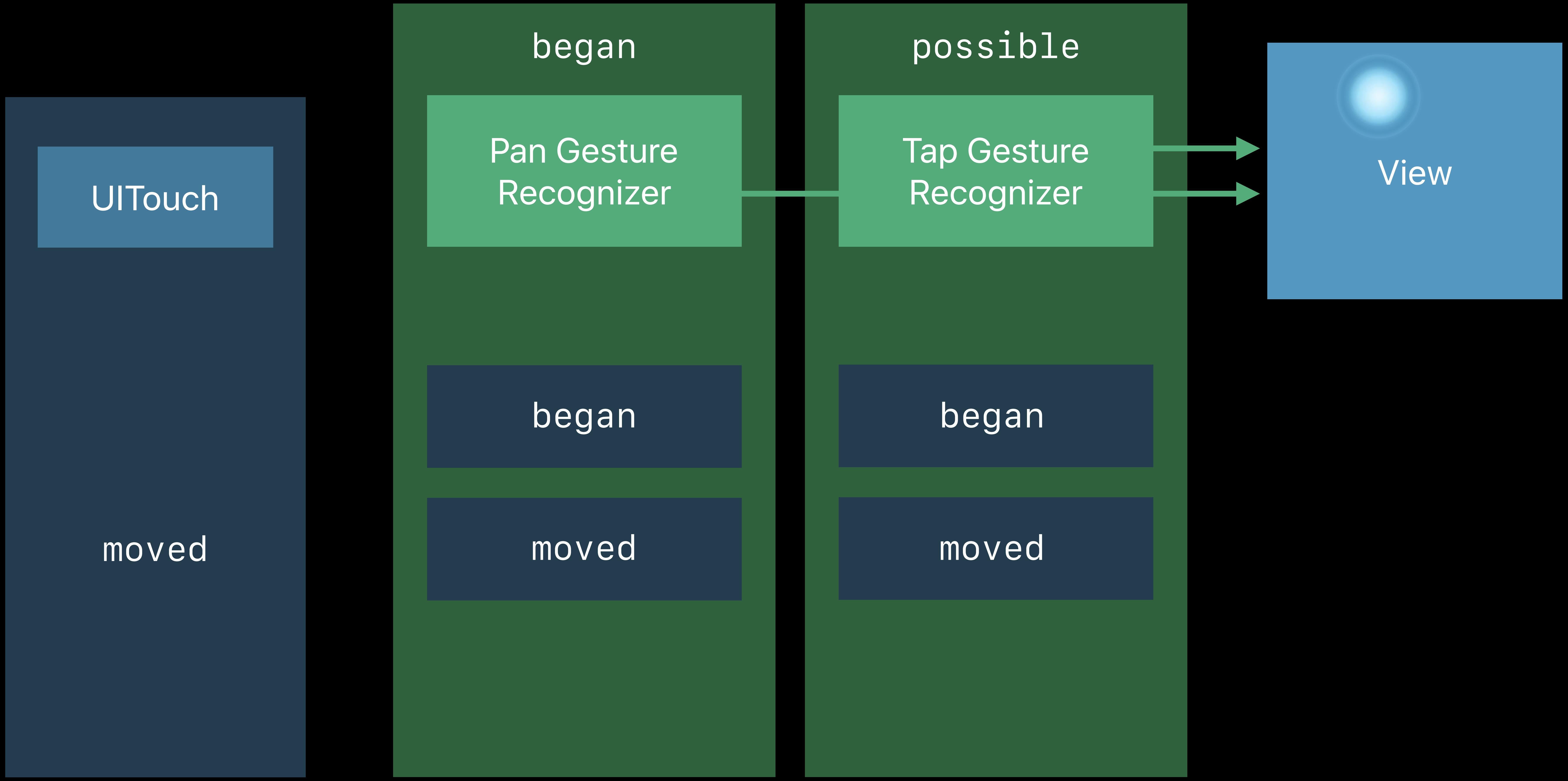

UITouch
began
possible
Pan Gesture Recognizer
began
possible
Tap Gesture Recognizer
began
View



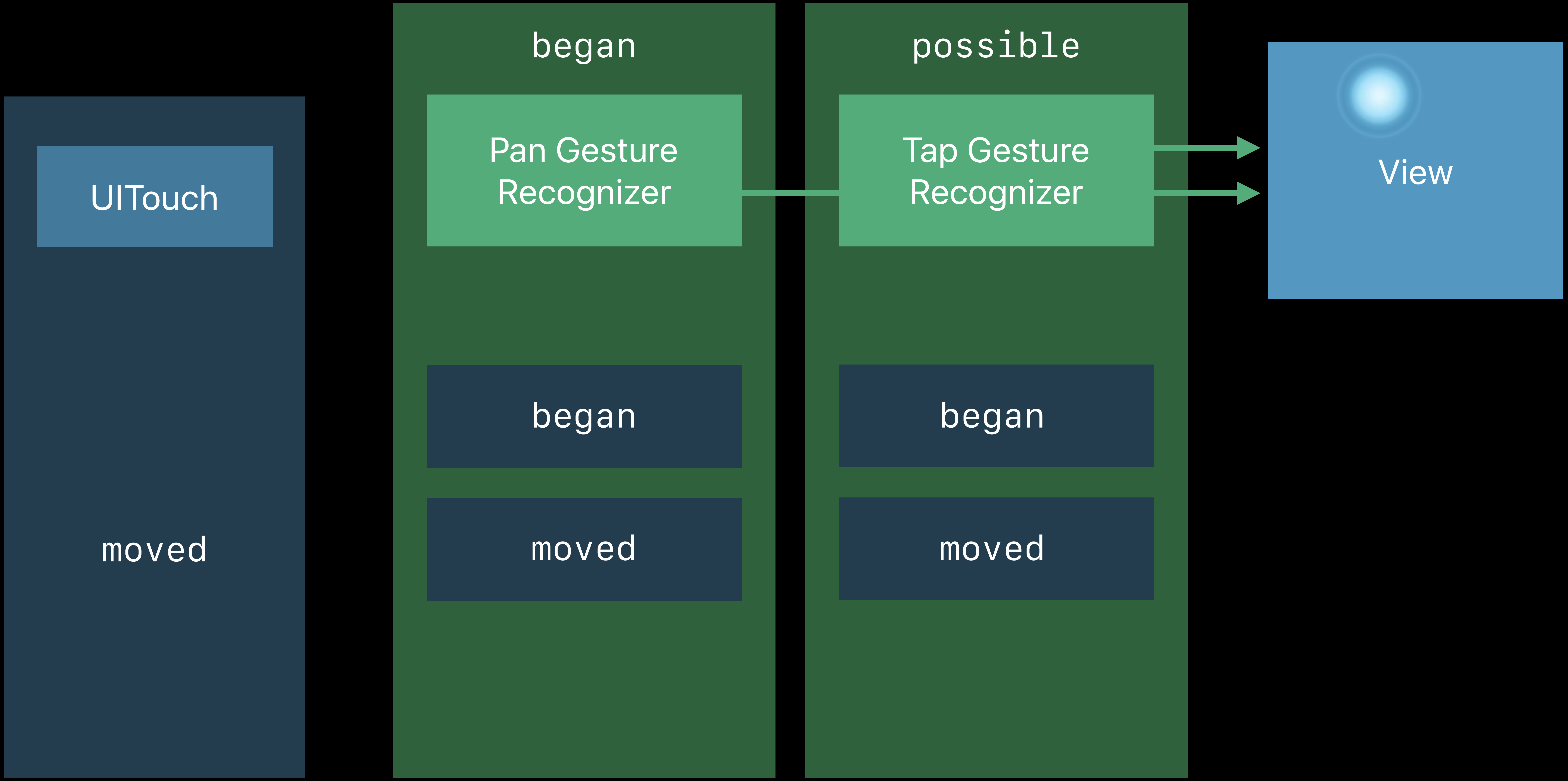
UITouch
moved
possible
Pan Gesture Recognizer
began
possible
Tap Gesture Recognizer
began
View


UITouch
moved
possible
Pan Gesture Recognizer
began
moved
possible
Tap Gesture Recognizer
began
View


began
possible
UITouch
Pan Gesture Recognizer
Tap Gesture Recognizer
View
moved
began
moved
began


UITouch
moved
began
Pan Gesture Recognizer
began
moved
possible
Tap Gesture Recognizer
began
moved
View

```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



UITouch

moved

began

Pan Gesture Recognizer

began

moved

possible

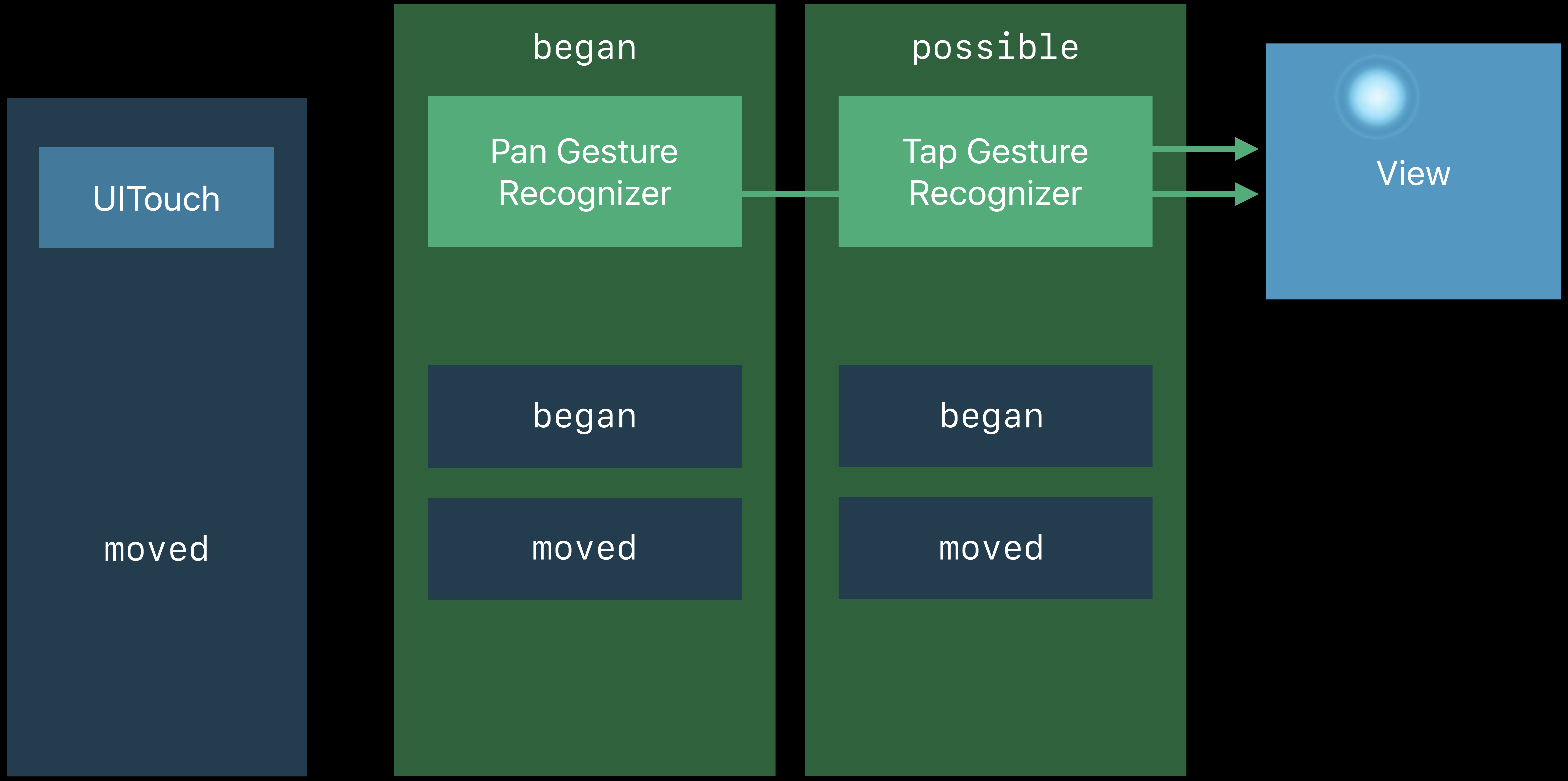
Tap Gesture Recognizer

began

moved

View

```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



UITouch

moved

began

Pan Gesture Recognizer

began

moved

possible

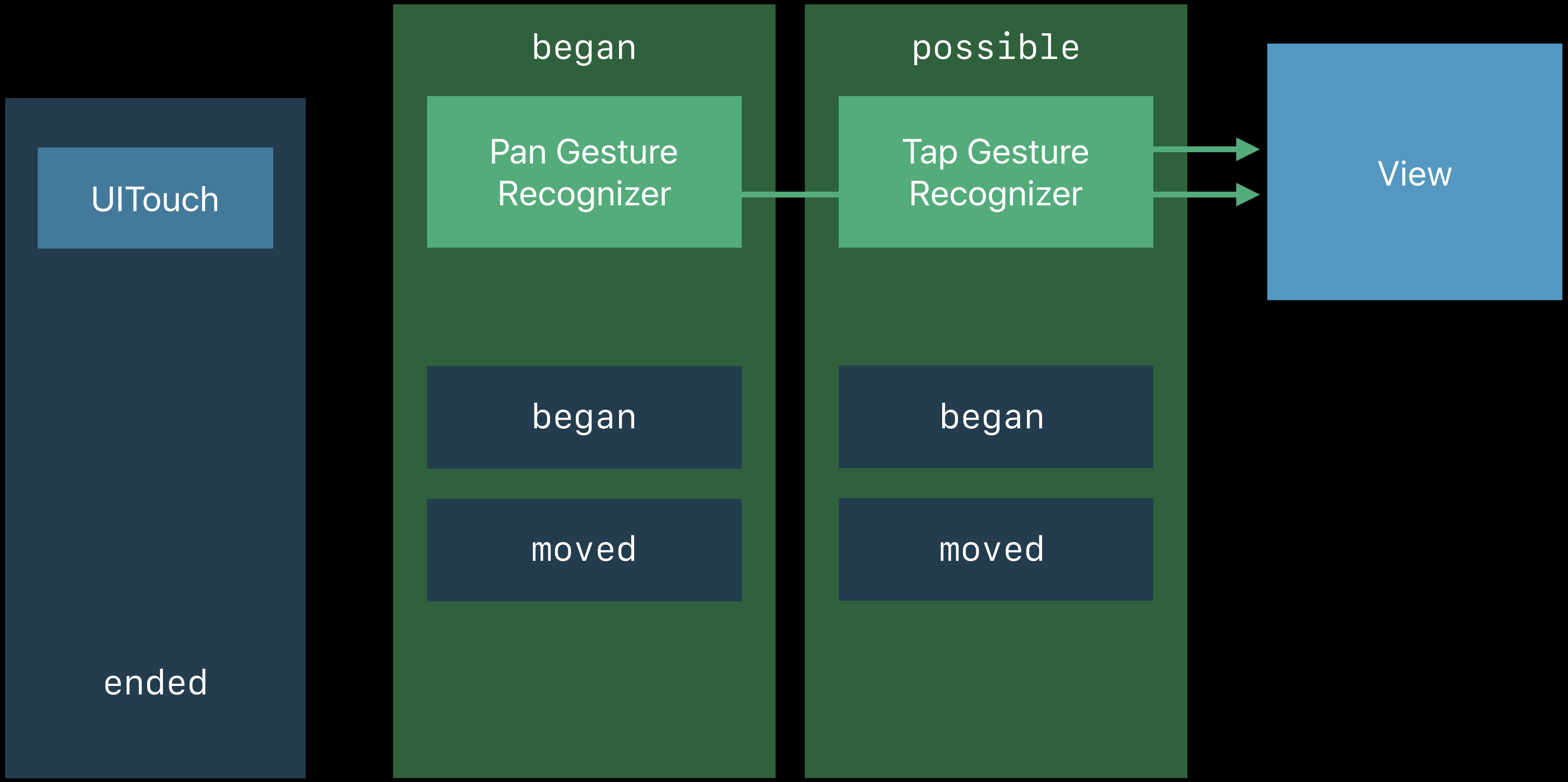
Tap Gesture Recognizer

began

moved

View

```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



UITouch

ended

began

Pan Gesture Recognizer

began

moved

possible

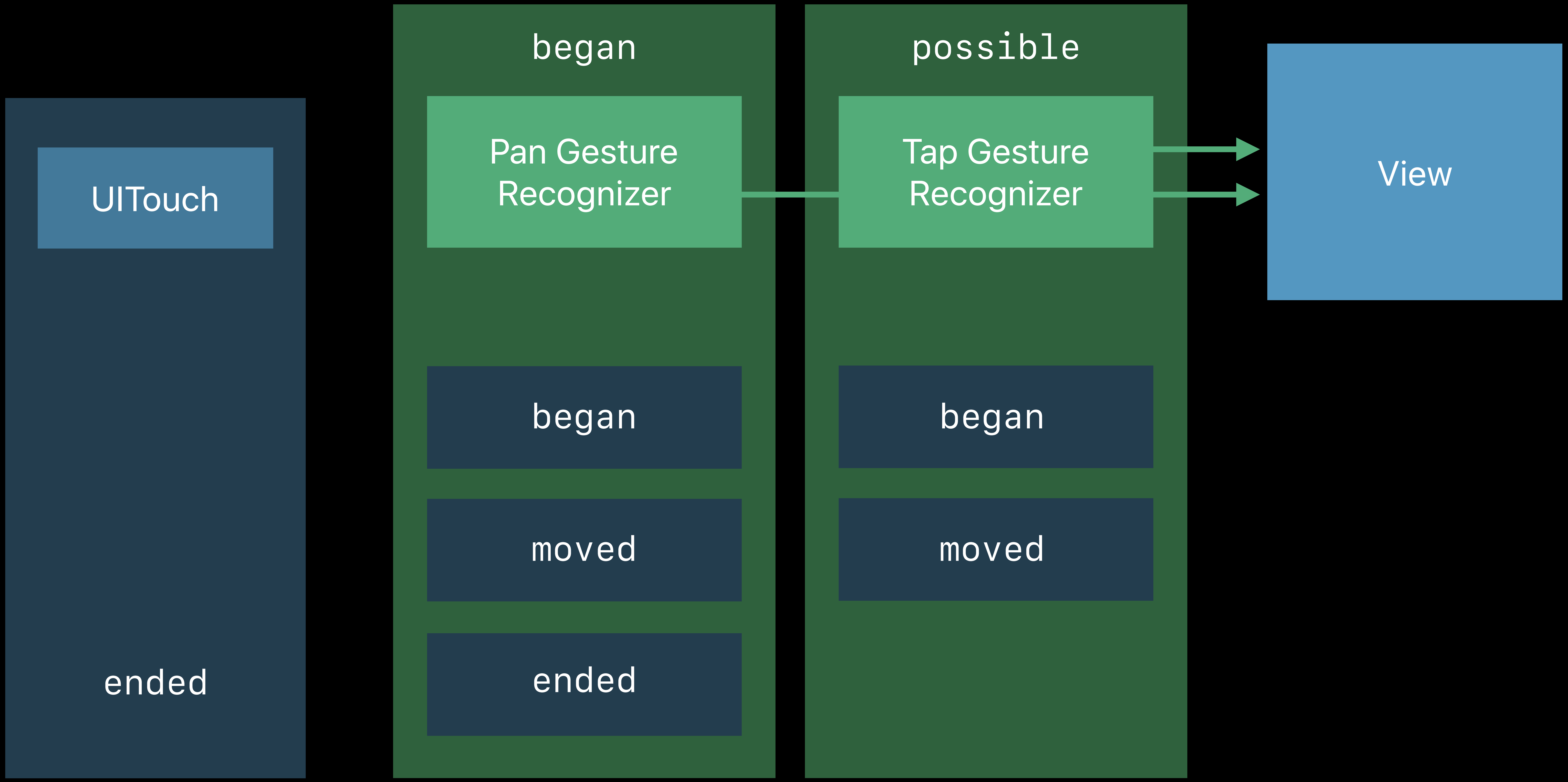
Tap Gesture Recognizer

began

moved

View

```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



UITouch

ended

began

Pan Gesture Recognizer

began

moved

ended

possible

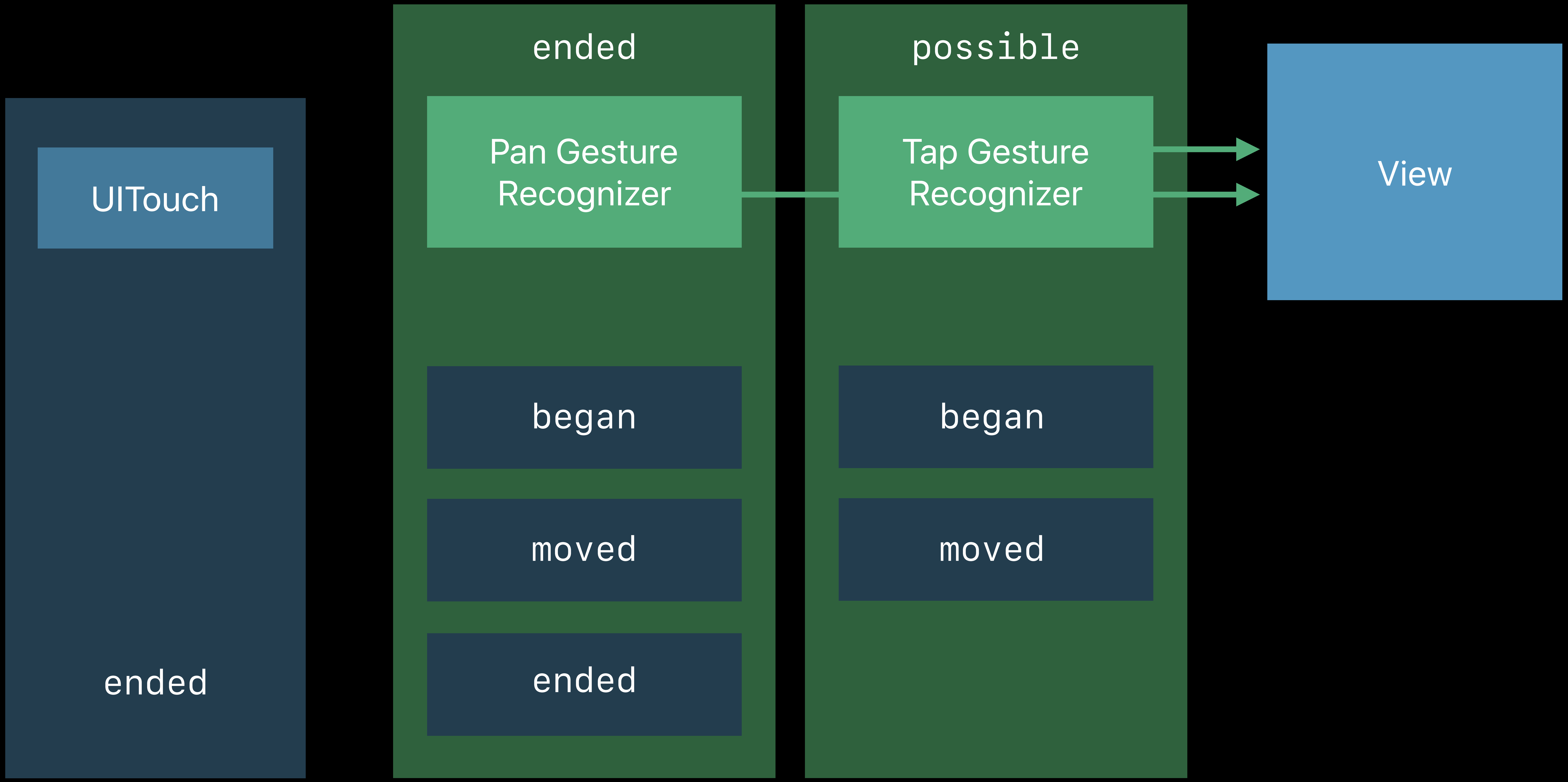
Tap Gesture Recognizer

began

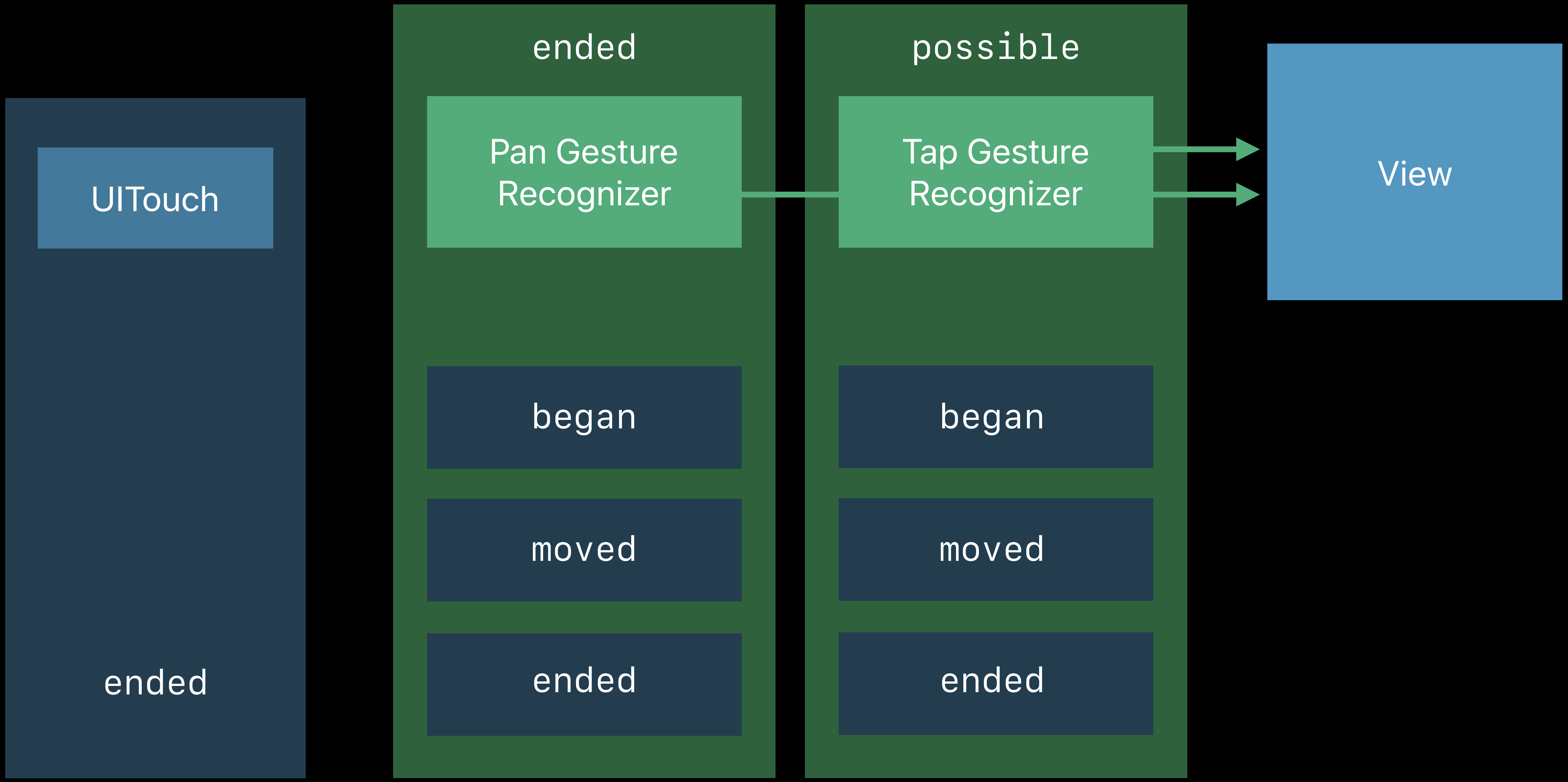
moved

View

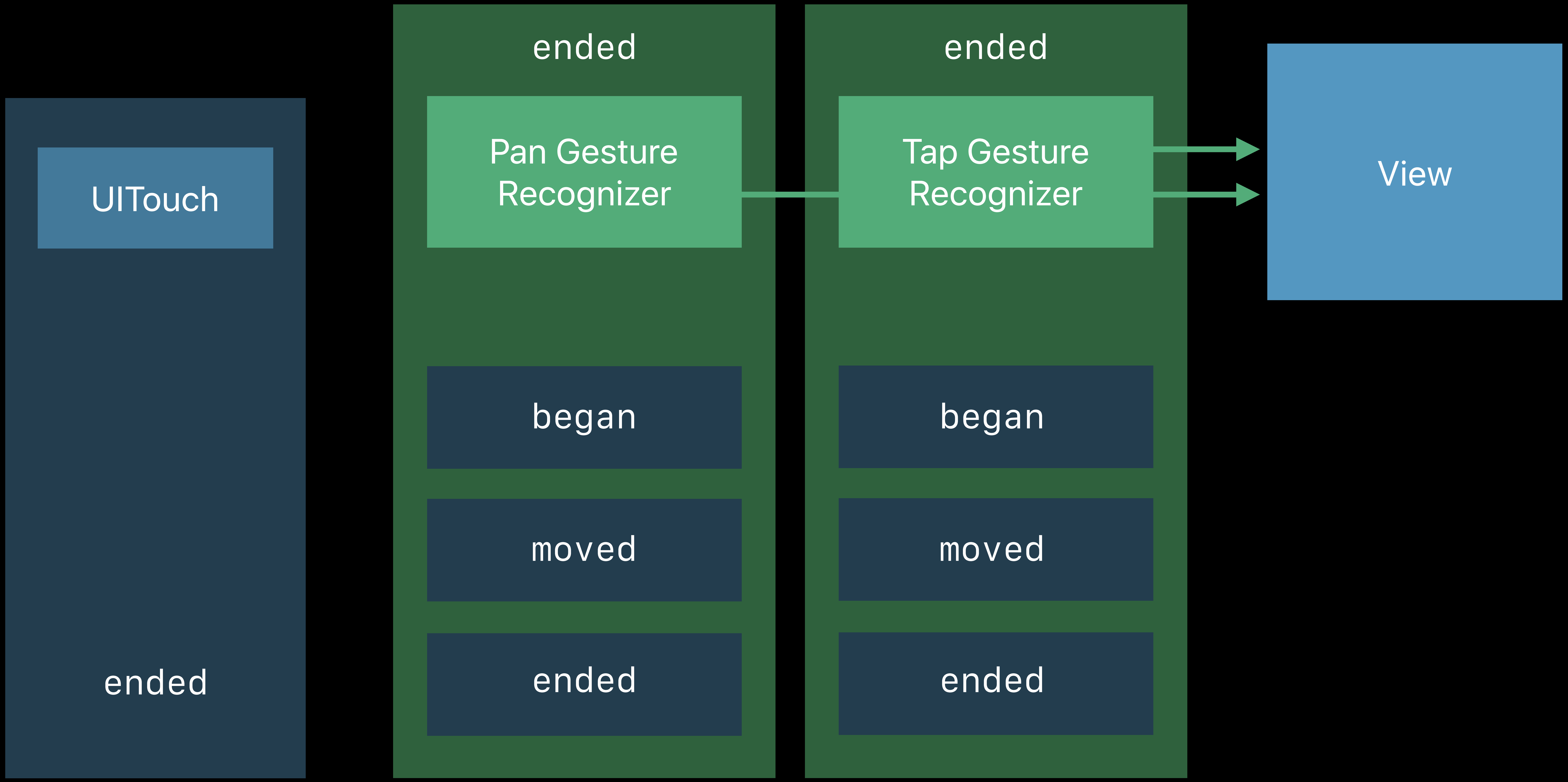
```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



UITouch

ended

ended

Pan Gesture Recognizer

began

moved

ended

ended

Tap Gesture Recognizer

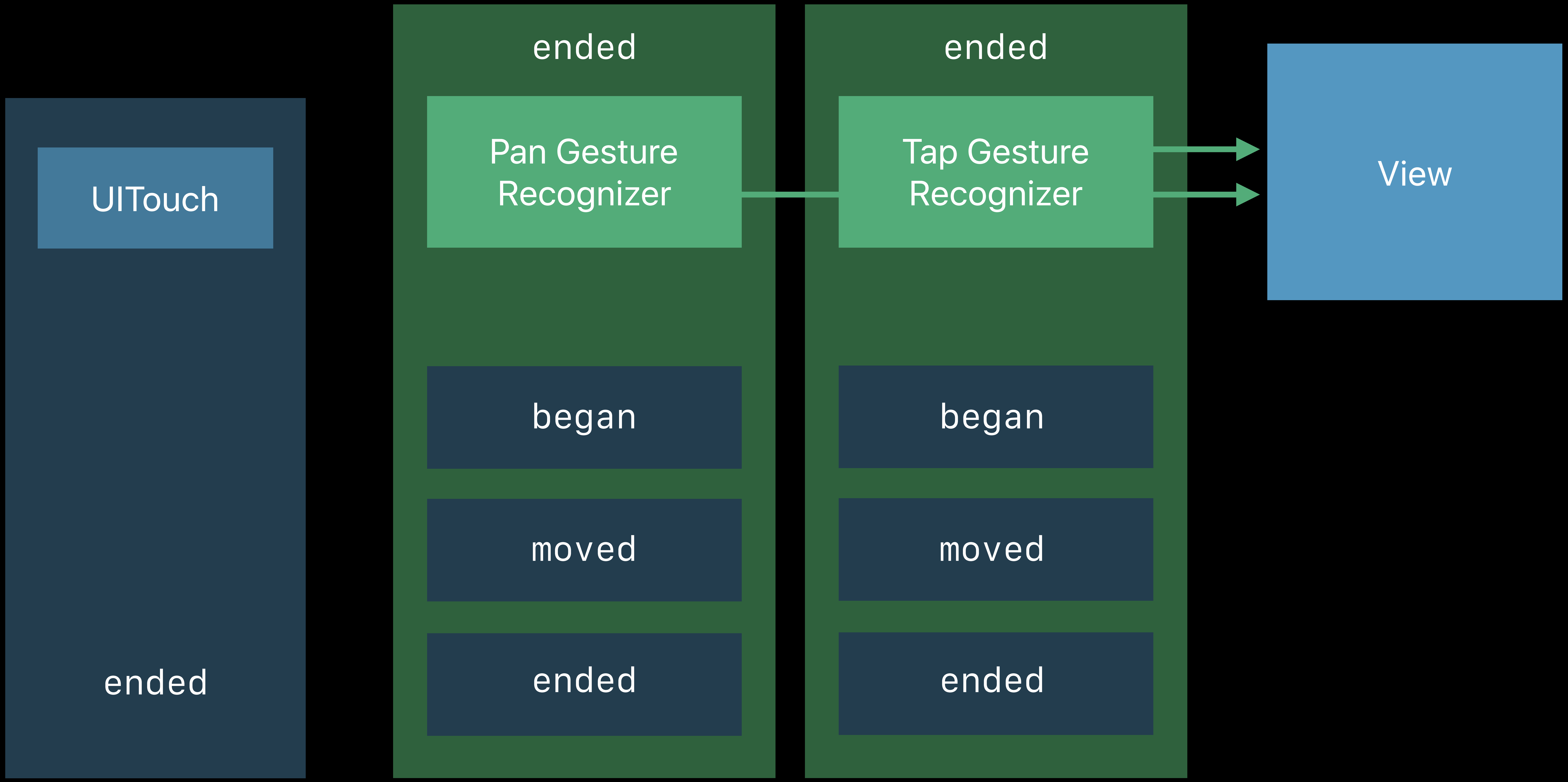
began

moved

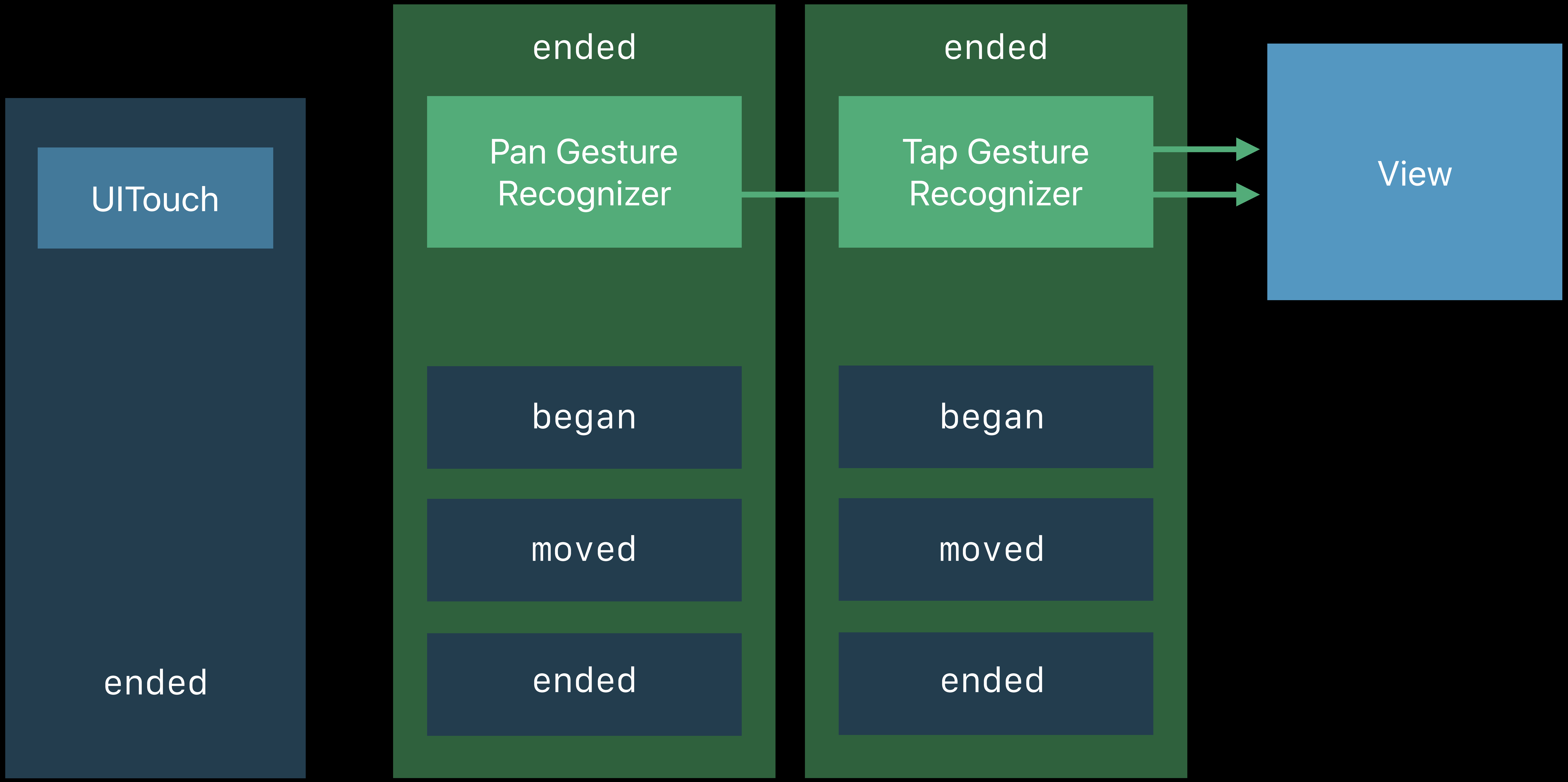
ended

View

```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



```
gestureRecognizer(_,
  shouldRecognizeSimultaneously:) -> true
```



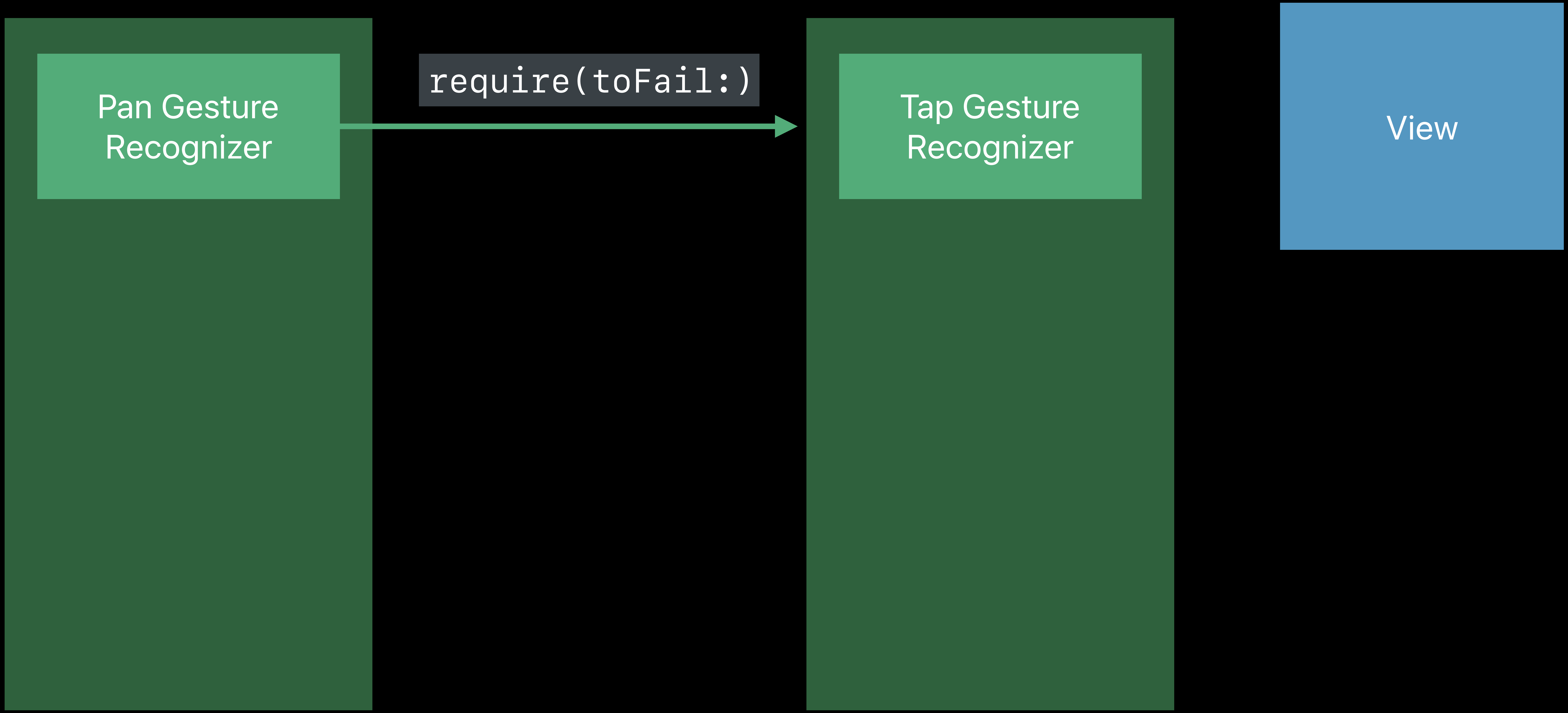
Failure Requirements

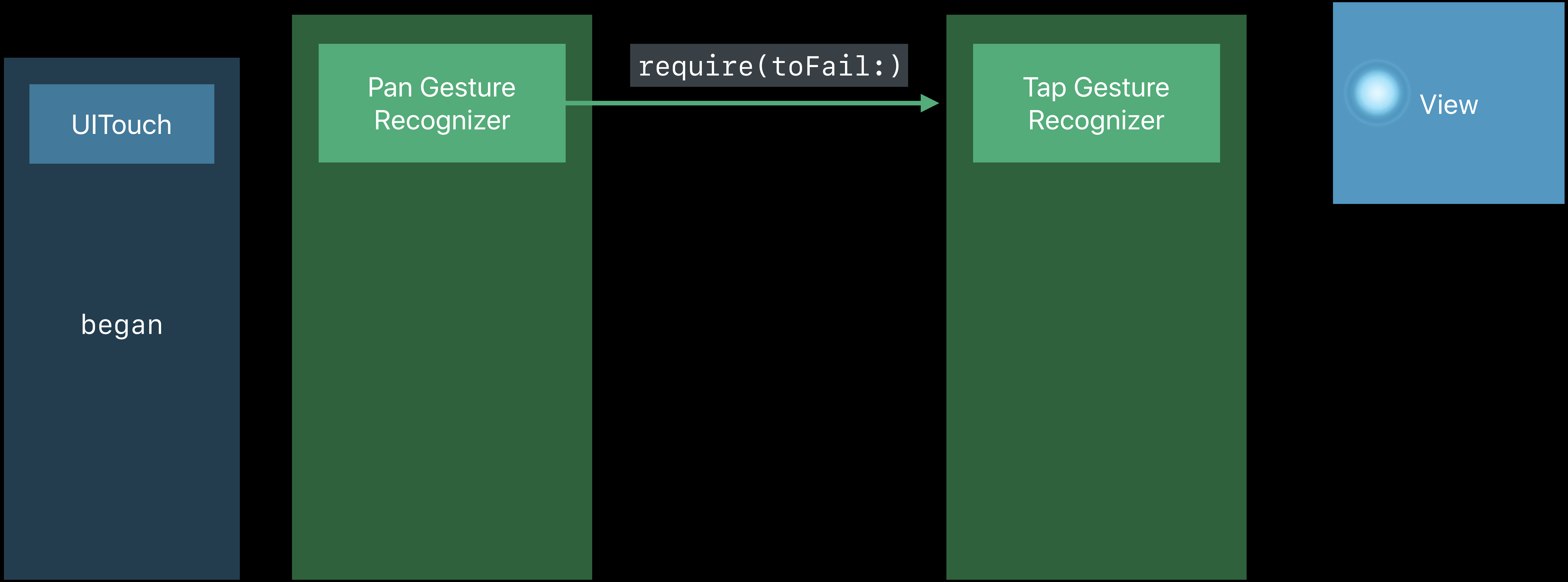
```
// Failure Requirements
class UIGestureRecognizer : NSObject {

    open func require(toFail otherGestureRecognizer: UIGestureRecognizer)
}

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                         shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                         shouldBeRequiredToFailBy otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

// Subclasses
open class UIGestureRecognizer : NSObject {

    open func shouldRequireFailure(of otherGestureRecognizer: UIGestureRecognizer) -> Bool
    open func shouldBeRequiredToFail(by otherGestureRecognizer: UIGestureRecognizer) -> Bool
}
```

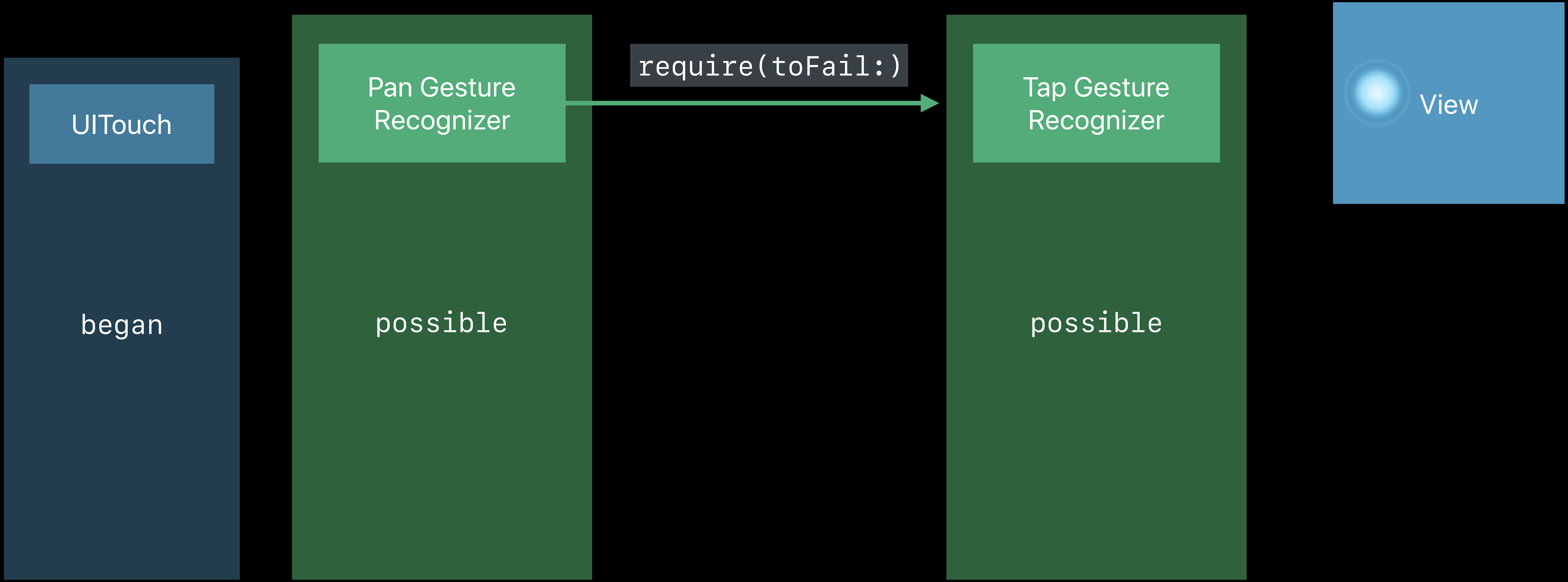
```swift
// Failure Requirements
class UIGestureRecognizer : NSObject {

    open func require(toFail otherGestureRecognizer: UIGestureRecognizer)
}

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                         shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                         shouldBeRequiredToFailBy otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

// Subclasses
open class UIGestureRecognizer : NSObject {

    open func shouldRequireFailure(of otherGestureRecognizer: UIGestureRecognizer) -> Bool
    open func shouldBeRequiredToFail(by otherGestureRecognizer: UIGestureRecognizer) -> Bool
}
```

```
// Failure Requirements
class UIGestureRecognizer : NSObject {

    open func require(toFail otherGestureRecognizer: UIGestureRecognizer)
}

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                         shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                         shouldBeRequiredToFailBy otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

// Subclasses
open class UIGestureRecognizer : NSObject {

    open func shouldRequireFailure(of otherGestureRecognizer: UIGestureRecognizer) -> Bool
    open func shouldBeRequiredToFail(by otherGestureRecognizer: UIGestureRecognizer) -> Bool
}
```

```
// Failure Requirements
class UIGestureRecognizer : NSObject {

    open func require(toFail otherGestureRecognizer: UIGestureRecognizer)
}

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                         shouldRequireFailureOf otherGestureRecognizer: UIGestureRecognizer) -> Bool

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                         shouldBeRequiredToFailBy otherGestureRecognizer: UIGestureRecognizer) -> Bool
}

// Subclasses
open class UIGestureRecognizer : NSObject {

    open func shouldRequireFailure(of otherGestureRecognizer: UIGestureRecognizer) -> Bool
    open func shouldBeRequiredToFail(by otherGestureRecognizer: UIGestureRecognizer) -> Bool
}
```


Pan Gesture Recognizer
require(toFail:)
Tap Gesture Recognizer
View


UITouch
began
Pan Gesture Recognizer
require(toFail:)
Tap Gesture Recognizer
View


UITouch

began

Pan Gesture Recognizer

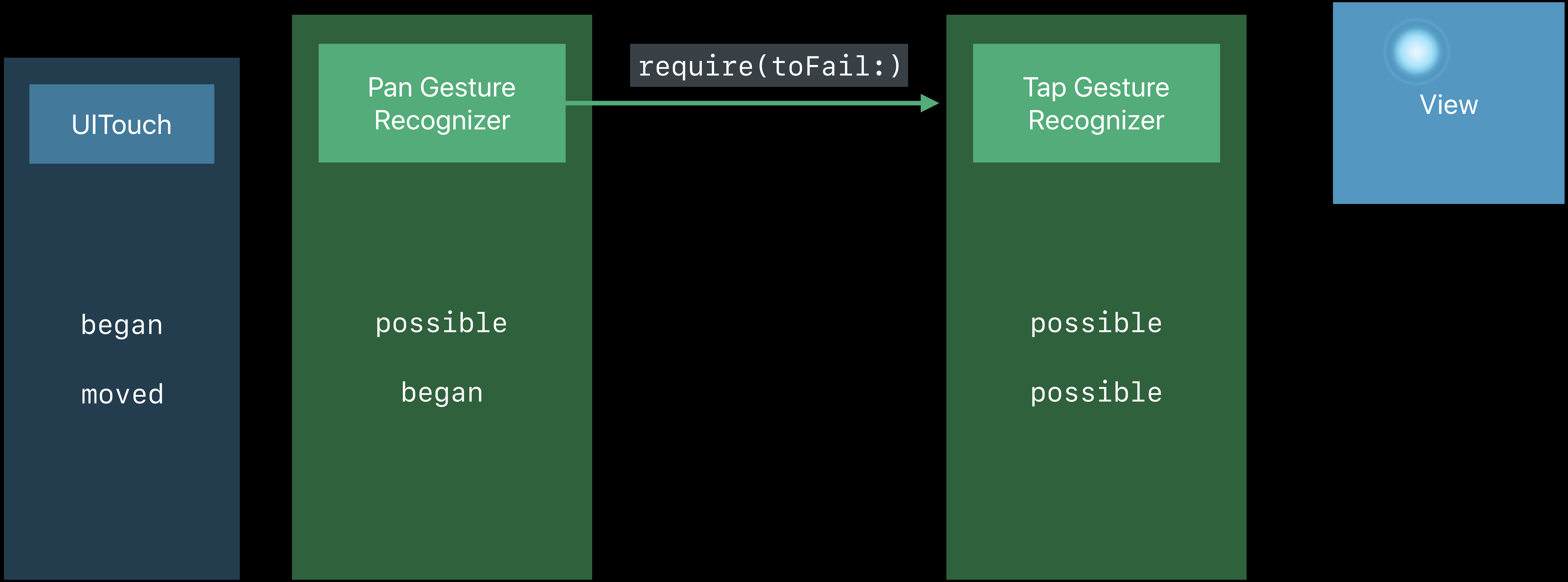
possible

`require(toFail:)`

Tap Gesture Recognizer

possible

View



UITouch

began

moved

Pan Gesture Recognizer

possible

began

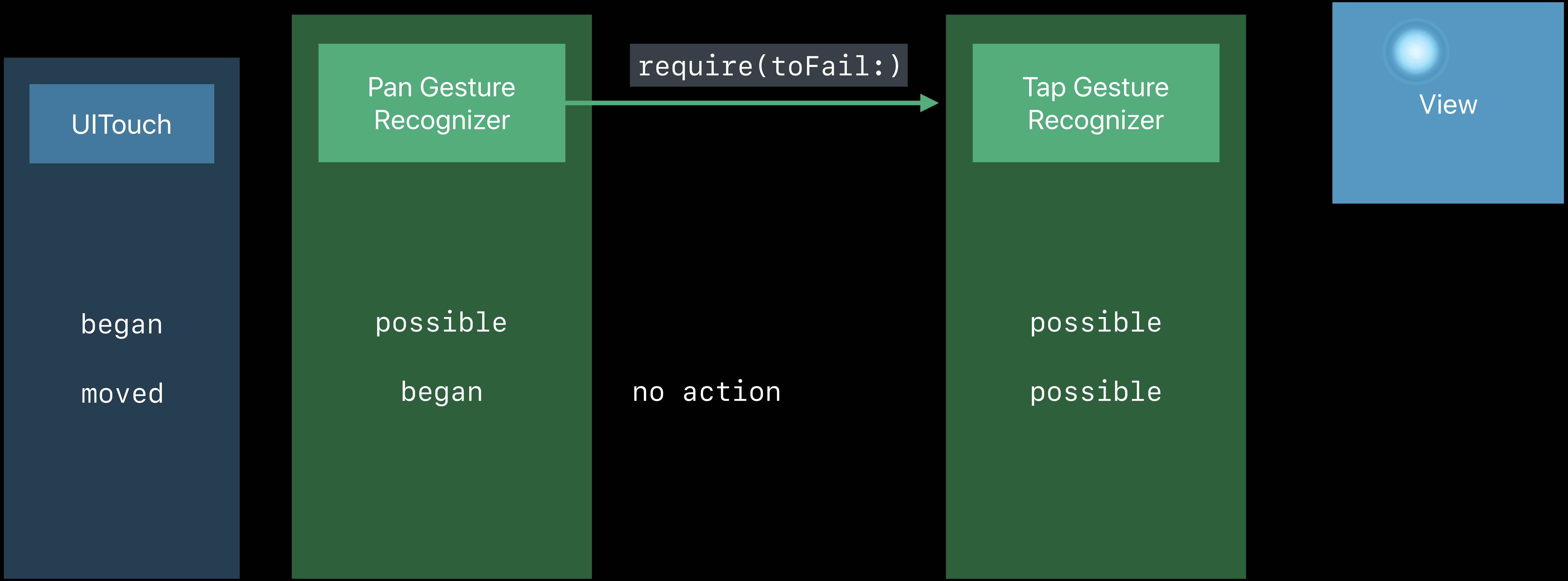
require(toFail:)

Tap Gesture Recognizer

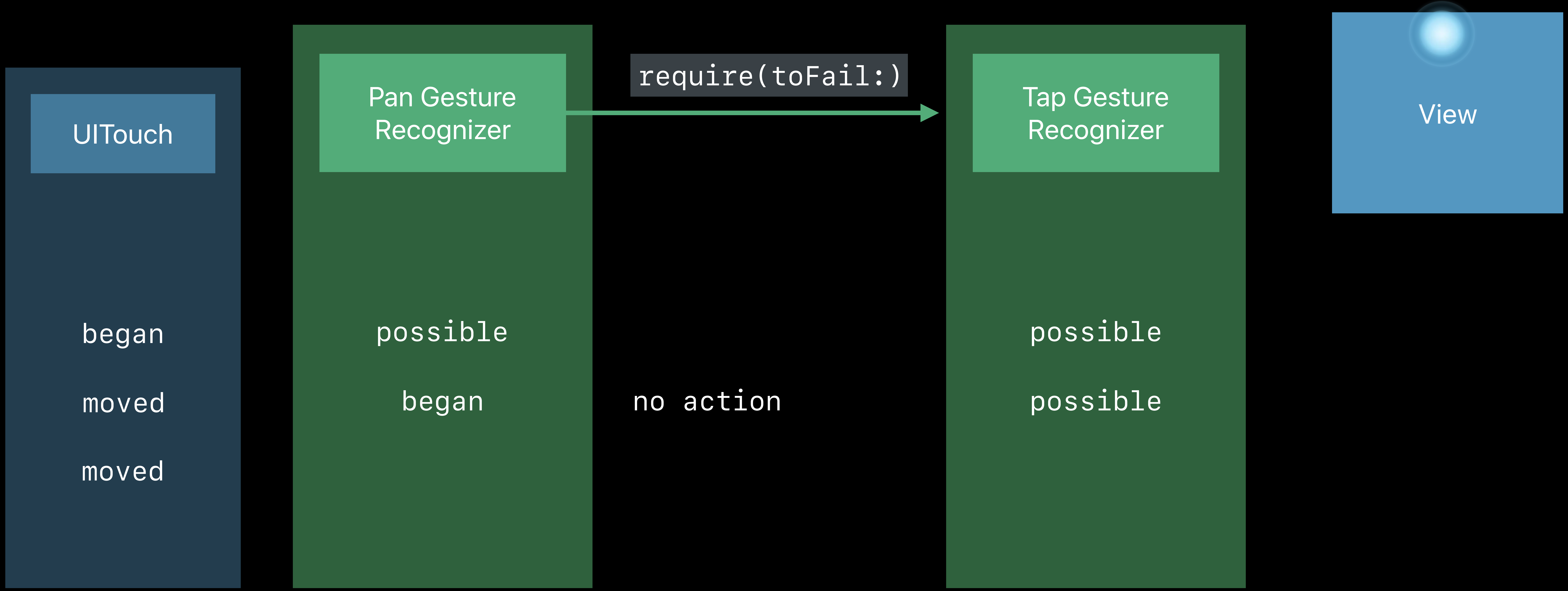
possible

possible

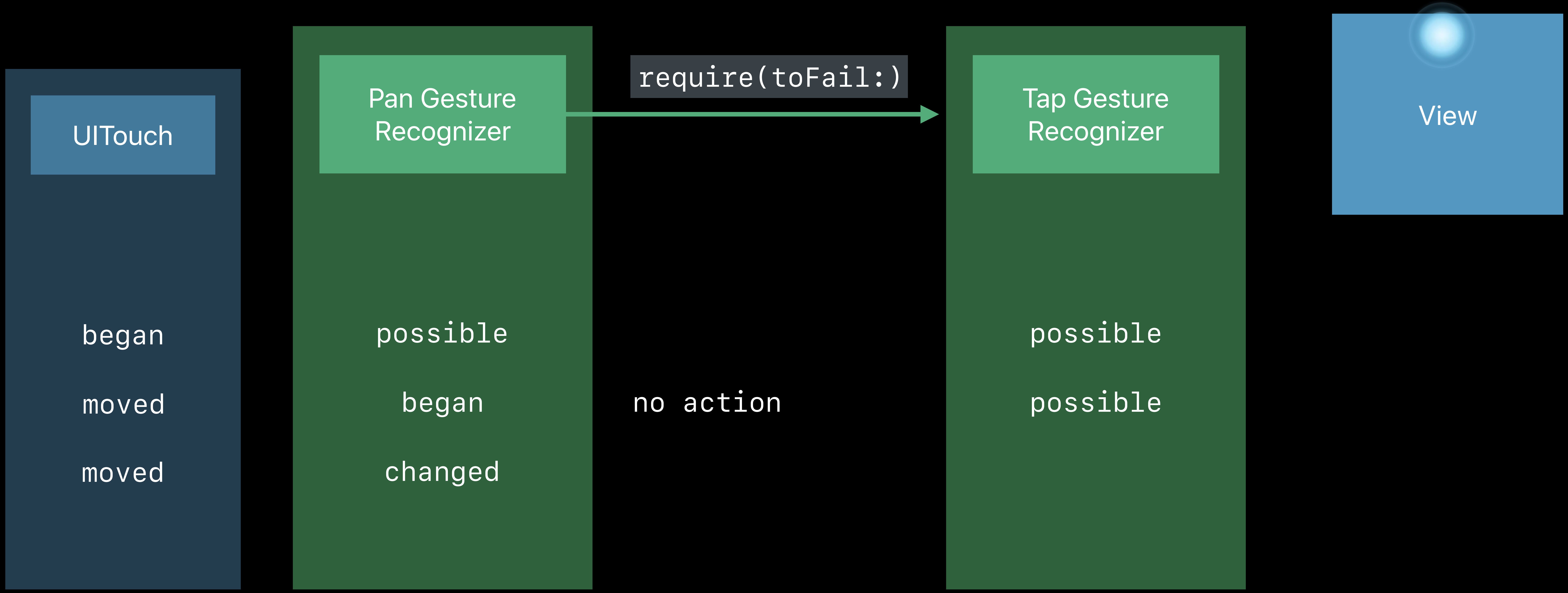
View



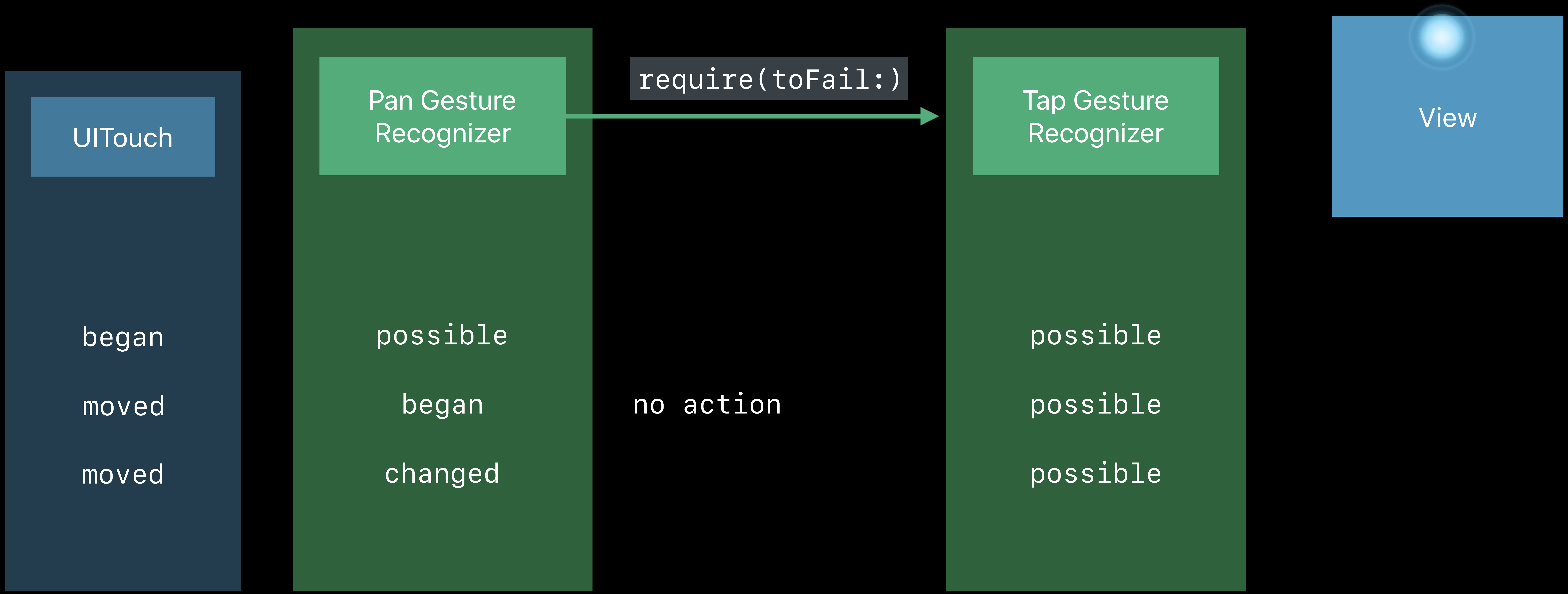
UITouch	Pan Gesture Recognizer	require(toFail:)	Tap Gesture Recognizer
began	possible		possible
moved	began	no action	possible



UITouch	Pan Gesture Recognizer	require(toFail:)	Tap Gesture Recognizer	View
began	possible		possible	
moved	began	no action	possible	
moved				



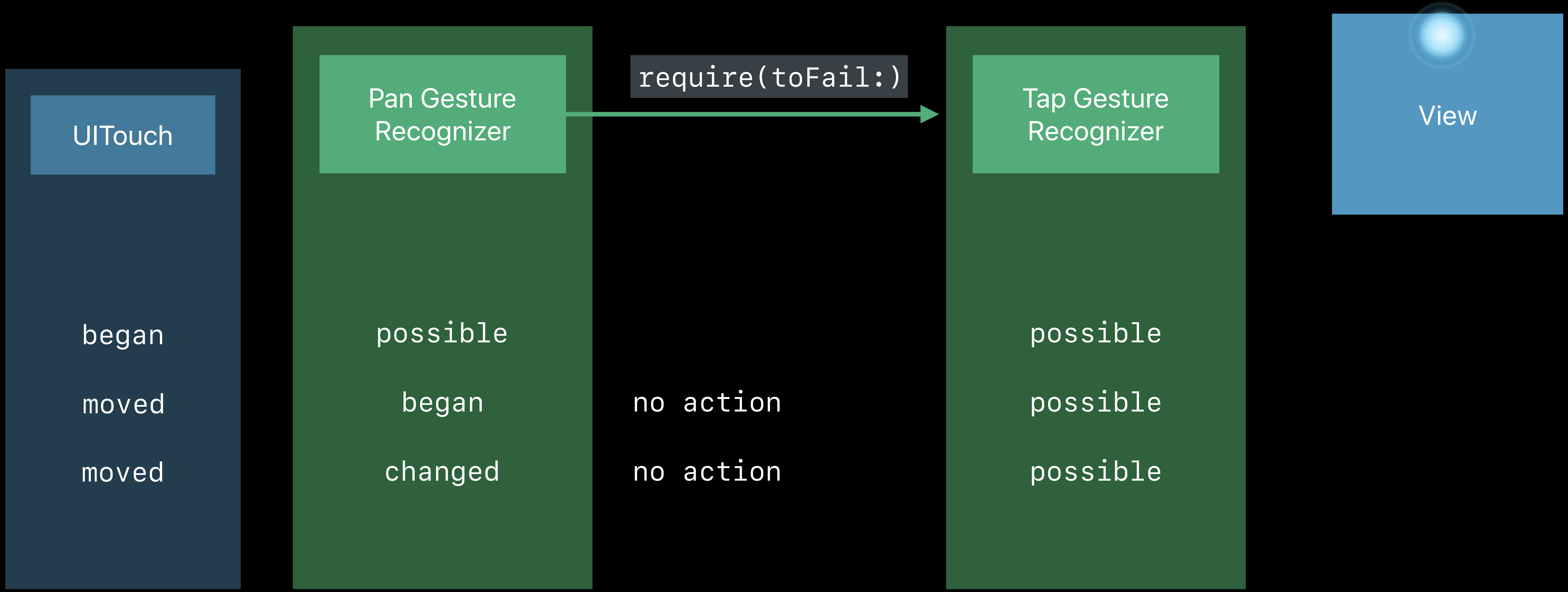
UITouch	Pan Gesture Recognizer	require(toFail:)	Tap Gesture Recognizer	View
began	possible		possible	
moved	began	no action	possible	
moved	changed			



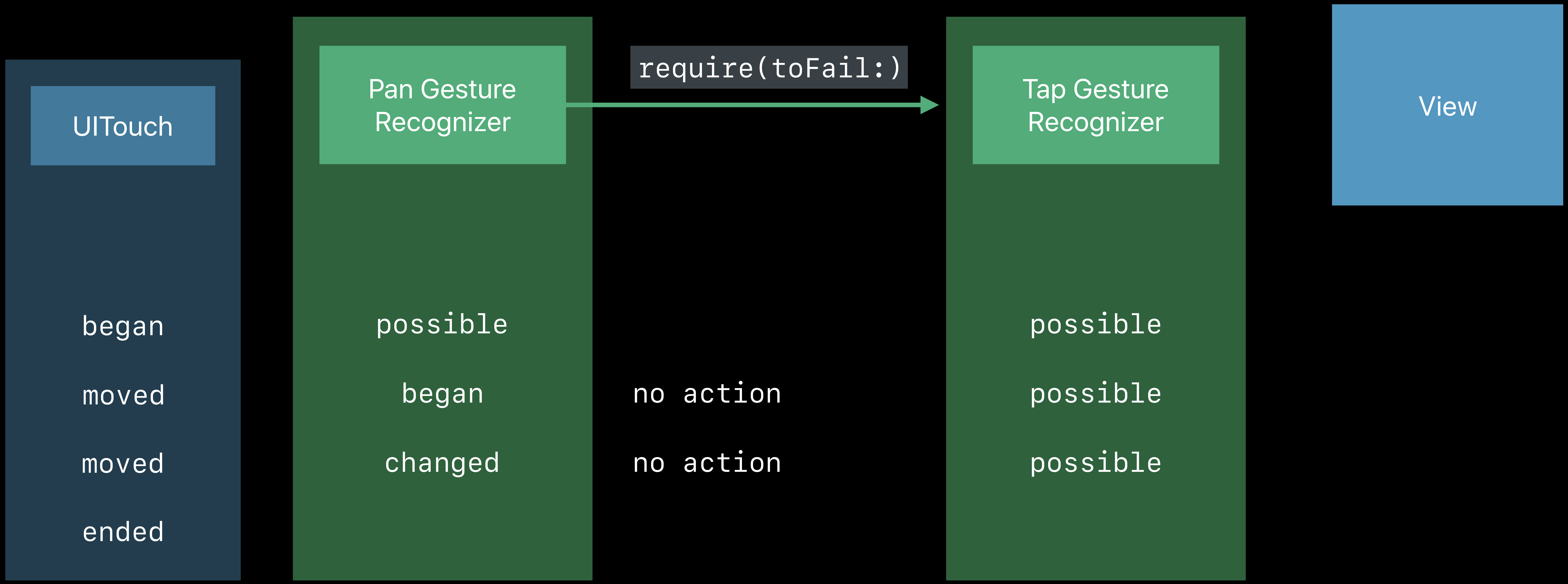
`require(toFail:)`

View

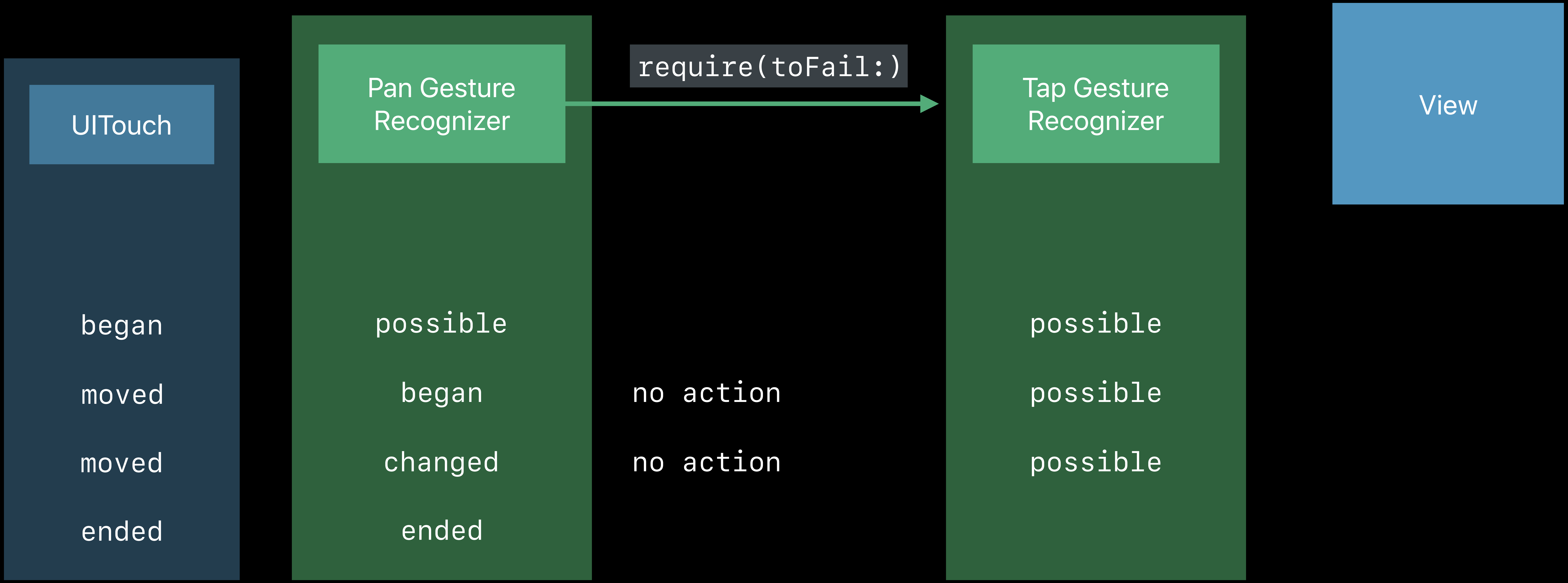
UITouch	Pan Gesture Recognizer		Tap Gesture Recognizer
`began`	`possible`		`possible`
`moved`	`began`	`no action`	`possible`
`moved`	`changed`		`possible`



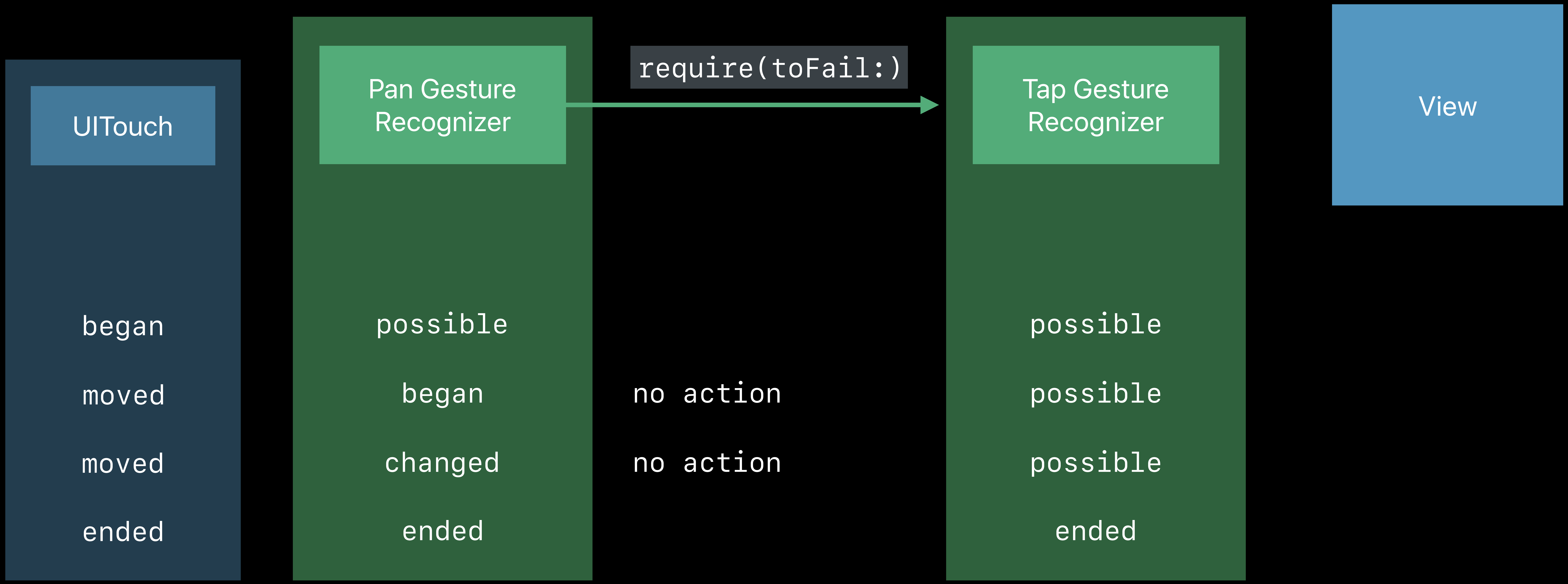
UITouch	Pan Gesture Recognizer	`require(toFail:)`	Tap Gesture Recognizer
began	possible		possible
moved	began	no action	possible
moved	changed	no action	possible



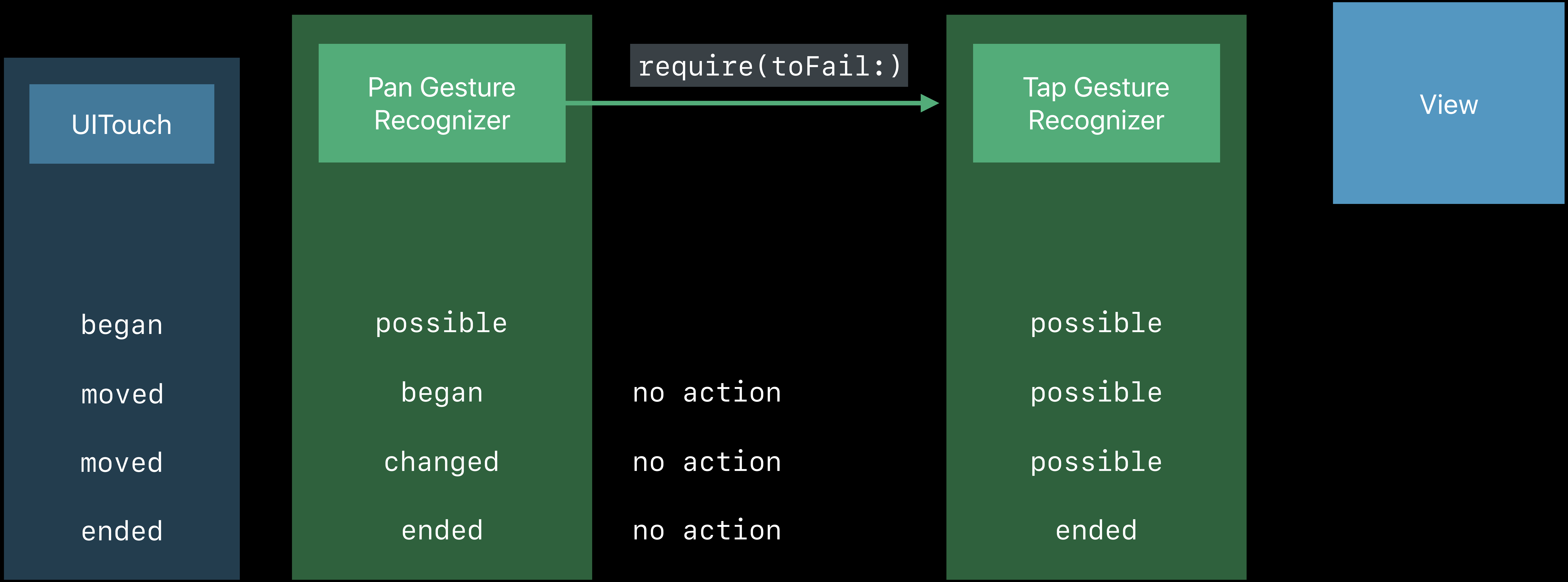
UITouch	Pan Gesture Recognizer	require(toFail:)	Tap Gesture Recognizer	View
began	possible		possible	
moved	began	no action	possible	
moved	changed	no action	possible	
ended				



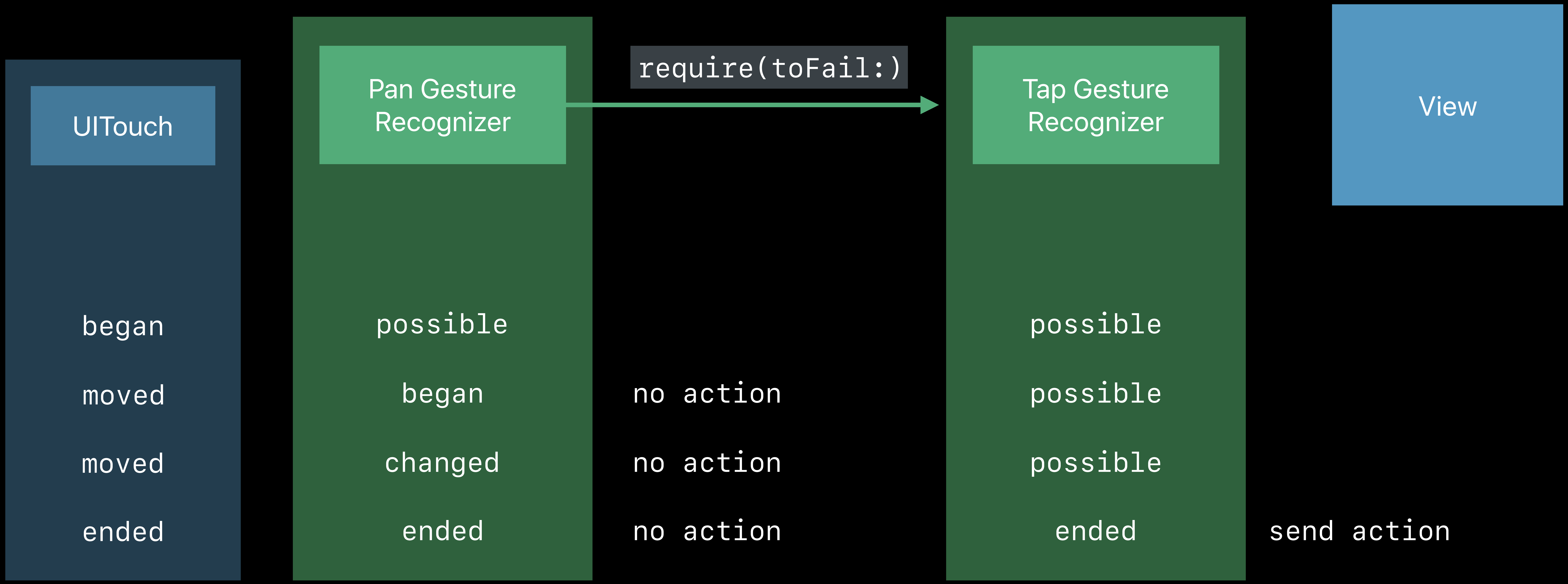
UITouch	Pan Gesture Recognizer	require(toFail:)	Tap Gesture Recognizer	View
began	possible		possible	
moved	began	no action	possible	
moved	changed	no action	possible	
ended	ended			



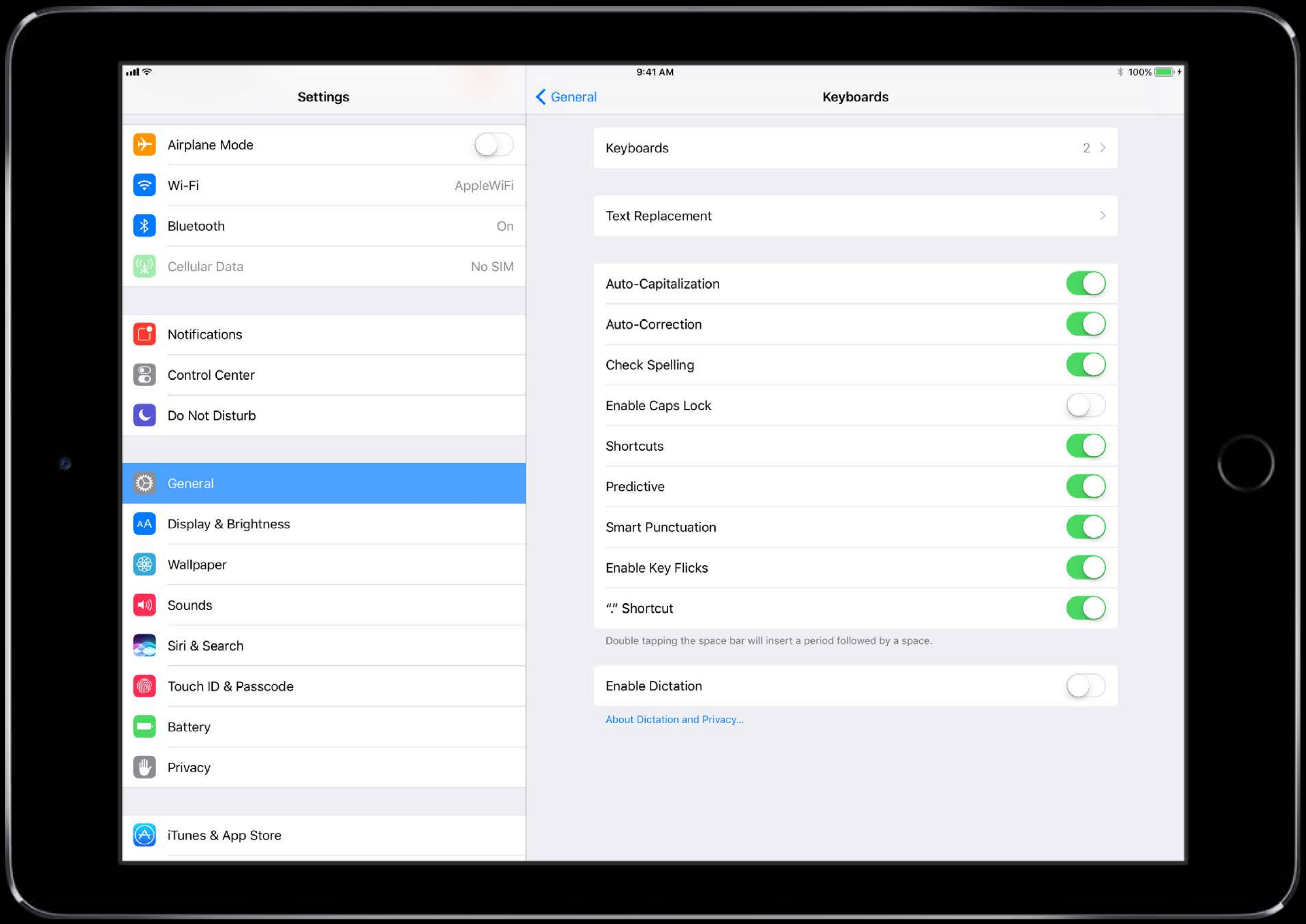
UITouch	Pan Gesture Recognizer	require(toFail:)	Tap Gesture Recognizer
began	possible		possible
moved	began	no action	possible
moved	changed	no action	possible
ended	ended		ended



UITouch	Pan Gesture Recognizer	require(toFail:)	Tap Gesture Recognizer	View
began	possible		possible	
moved	began	no action	possible	
moved	changed	no action	possible	
ended	ended	no action	ended	



UITouch	Pan Gesture Recognizer	require(toFail:)	Tap Gesture Recognizer	View
began	possible		possible	
moved	began	no action	possible	
moved	changed	no action	possible	
ended	ended	no action	ended	send action



9:41 AM

100%

Settings

Airplane Mode ○

Wi-Fi AppleWiFi

Bluetooth On

Cellular Data No SIM

Notifications

Control Center

Do Not Disturb

General

Display & Brightness

Wallpaper

Sounds

Siri & Search

Touch ID & Passcode

Battery

Privacy

iTunes & App Store

< General

Keyboards

Keyboards 2 >

Text Replacement >

Auto-Capitalization ●

Auto-Correction ●

Check Spelling ●

Enable Caps Lock ○

Shortcuts ●

Predictive ●

Smart Punctuation ●

Enable Key Flicks ●

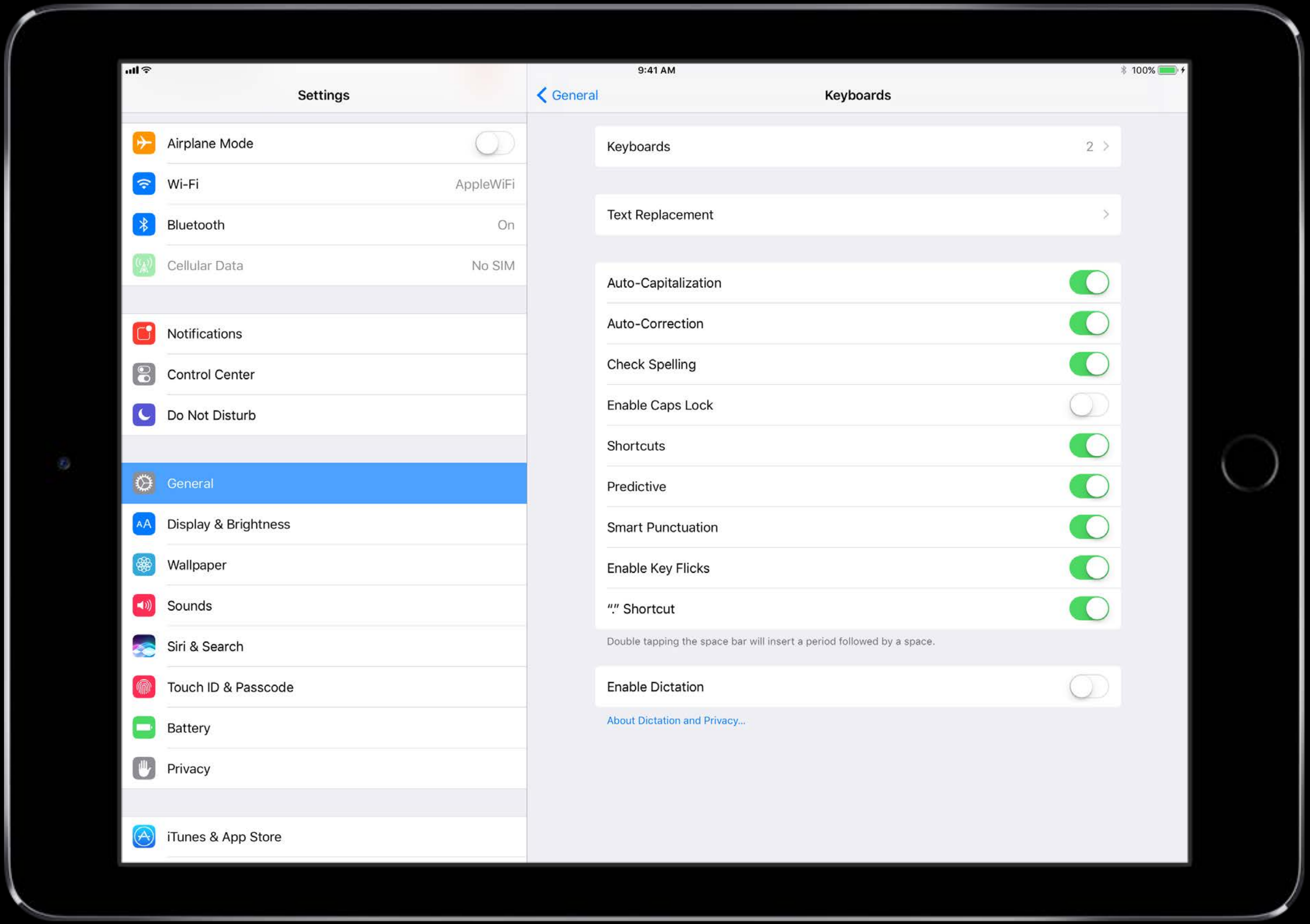
"." Shortcut ●

Double tapping the space bar will insert a period followed by a space.

Enable Dictation ○

About Dictation and Privacy...

163 UIGestureRecognizers



9:41 AM 100%

Settings

Airplane Mode ○

Wi-Fi AppleWiFi

Bluetooth On

Cellular Data No SIM

Notifications

Control Center

Do Not Disturb

General

Display & Brightness

Wallpaper

Sounds

Siri & Search

Touch ID & Passcode

Battery

Privacy

iTunes & App Store

General

Keyboards

Keyboards 2 >

Text Replacement >

Auto-Capitalization ●

Auto-Correction ●

Check Spelling ●

Enable Caps Lock ○

Shortcuts ●

Predictive ●

Smart Punctuation ●

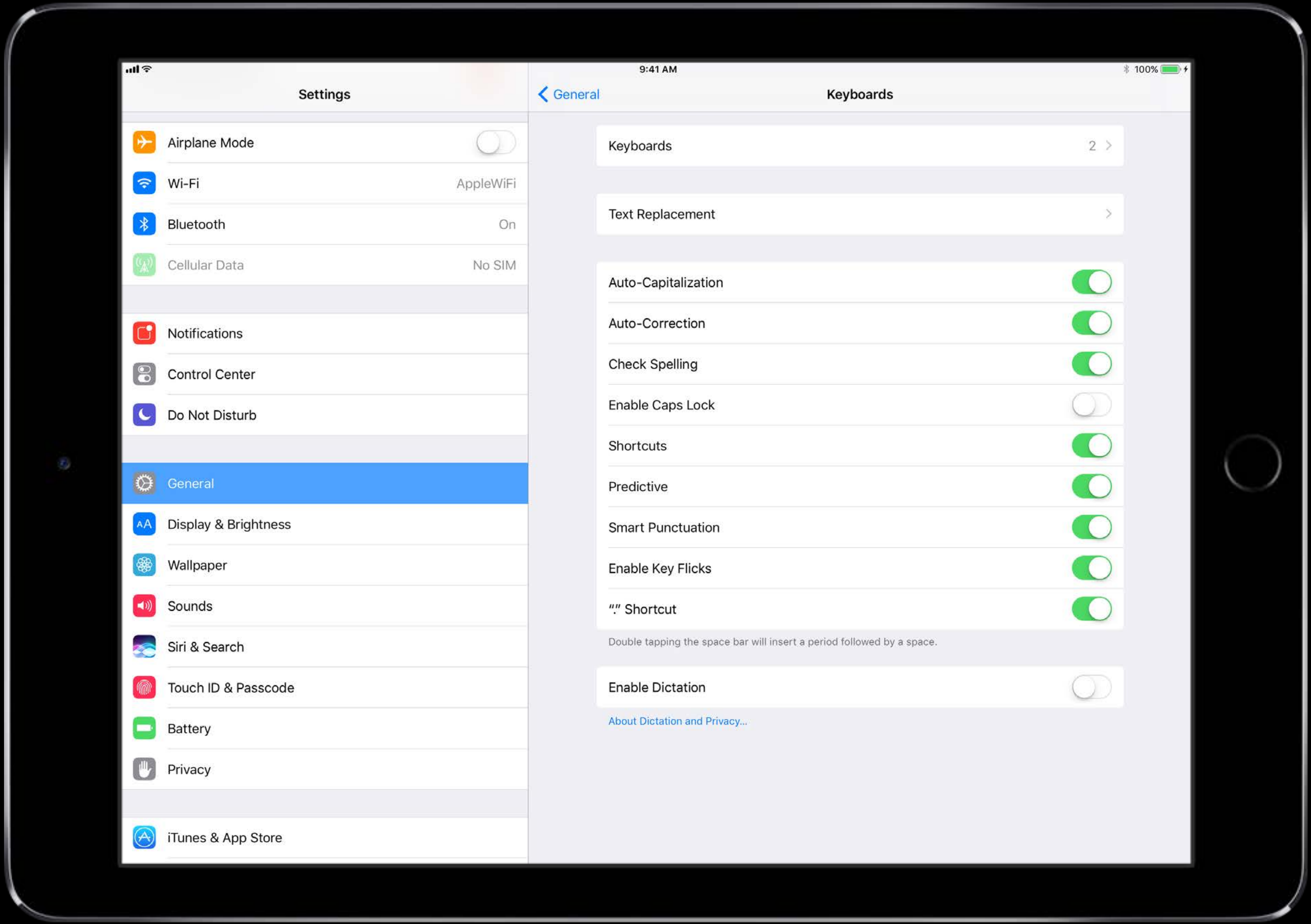
Enable Key Flicks ●

“.” Shortcut ●

Double tapping the space bar will insert a period followed by a space.

Enable Dictation ○

About Dictation and Privacy…



9:41 AM 100%

Settings

- Airplane Mode ○
- Wi-Fi AppleWiFi
- Bluetooth On
- Cellular Data No SIM

- Notifications
- Control Center
- Do Not Disturb

- General
- Display & Brightness
- Wallpaper
- Sounds
- Siri & Search
- Touch ID & Passcode
- Battery
- Privacy

- iTunes & App Store

‹ General Keyboards

- Keyboards 2 ›

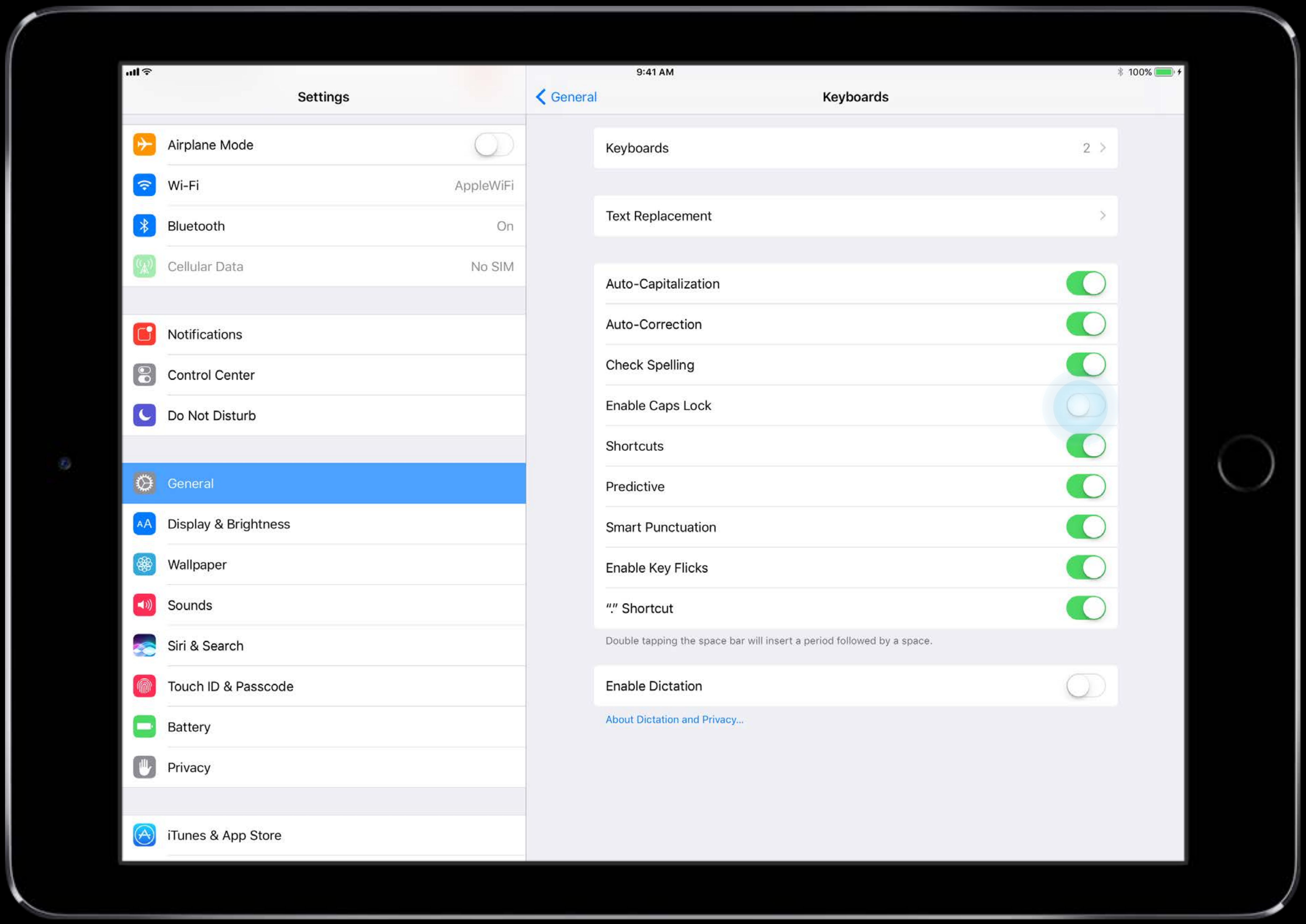
- Text Replacement ›

- Auto-Capitalization ●
- Auto-Correction ●
- Check Spelling ●
- Enable Caps Lock ○
- Shortcuts ●
- Predictive ●
- Smart Punctuation ●
- Enable Key Flicks ●
- “.” Shortcut ●

Double tapping the space bar will insert a period followed by a space.

- Enable Dictation ○

About Dictation and Privacy…



9:41 AM 100%

Settings

- Airplane Mode ○
- Wi-Fi AppleWiFi
- Bluetooth On
- Cellular Data No SIM

- Notifications
- Control Center
- Do Not Disturb

- General
- Display & Brightness
- Wallpaper
- Sounds
- Siri & Search
- Touch ID & Passcode
- Battery
- Privacy

- iTunes & App Store

General

Keyboards

- Keyboards 2 >

- Text Replacement >

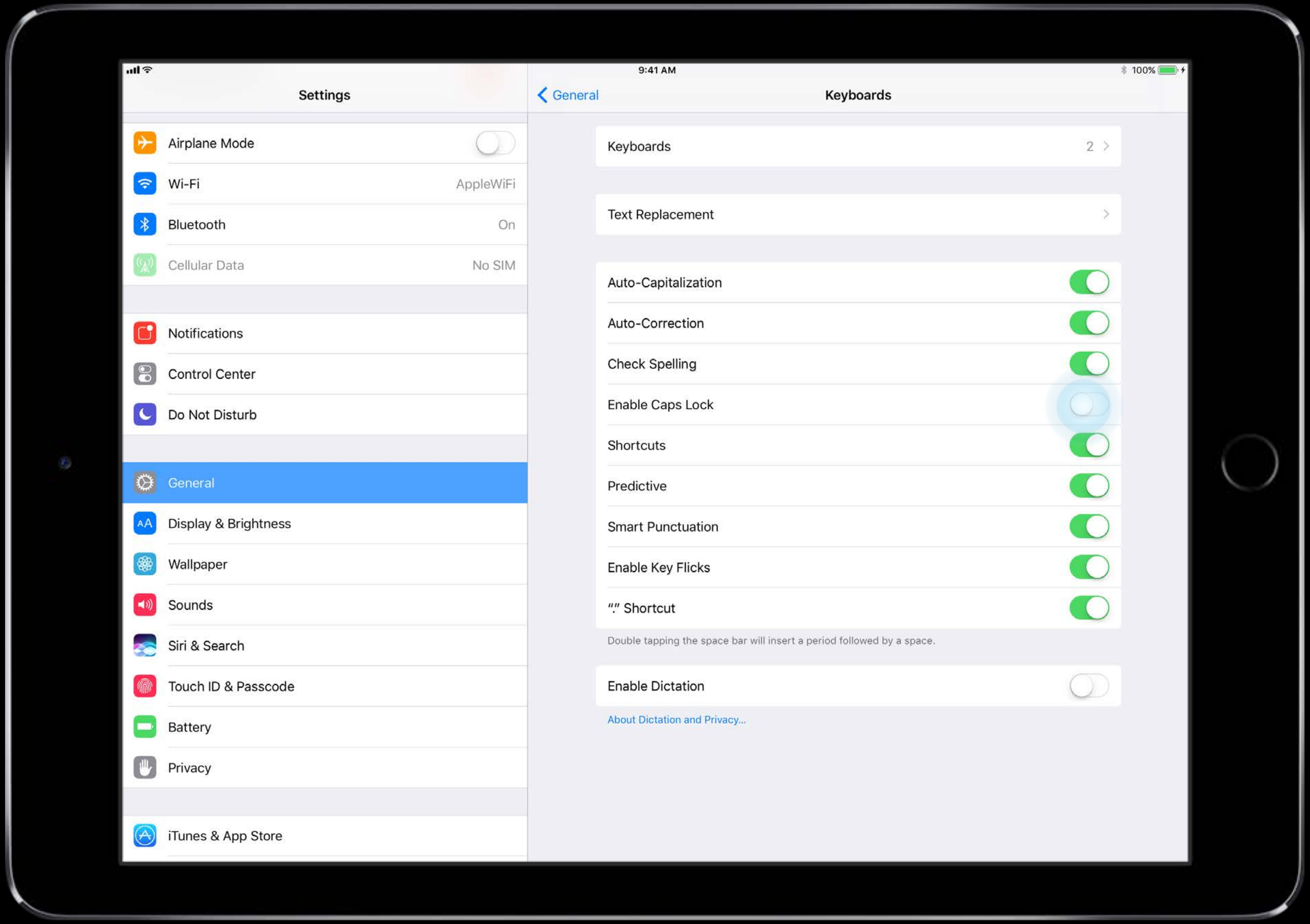
- Auto-Capitalization ●
- Auto-Correction ●
- Check Spelling ●
- Enable Caps Lock ○
- Shortcuts ●
- Predictive ●
- Smart Punctuation ●
- Enable Key Flicks ●
- “.” Shortcut ●

Double tapping the space bar will insert a period followed by a space.

- Enable Dictation ○

About Dictation and Privacy...

7 UIGestureRecognizers



9:41 AM 100%

Settings

- Airplane Mode ○
- Wi-Fi AppleWiFi
- Bluetooth On
- Cellular Data No SIM

- Notifications
- Control Center
- Do Not Disturb

- General
- Display & Brightness
- Wallpaper
- Sounds
- Siri & Search
- Touch ID & Passcode
- Battery
- Privacy

- iTunes & App Store

‹ General

Keyboards

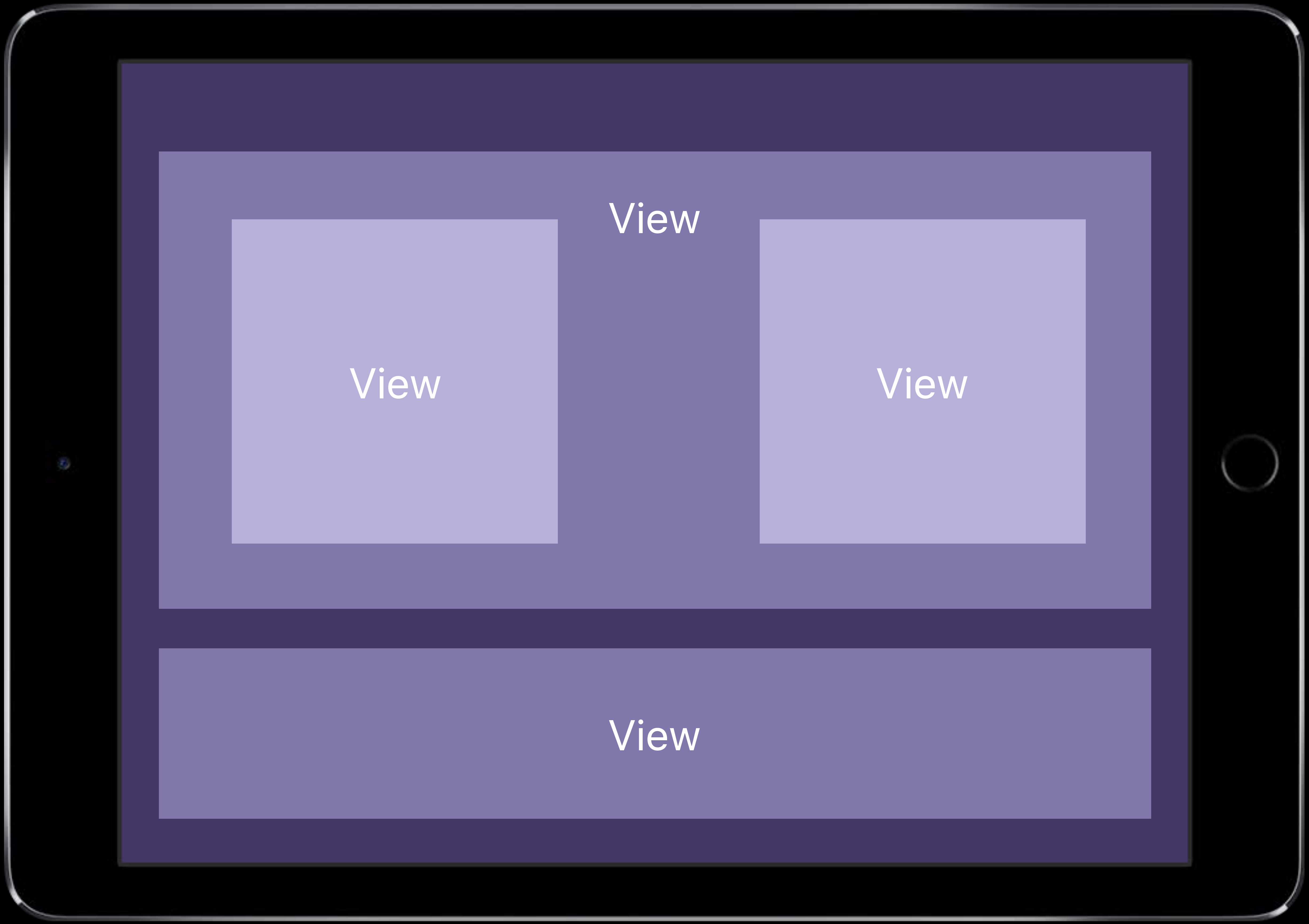
Keyboards	2 ›
Text Replacement	›
Auto-Capitalization	●
Auto-Correction	●
Check Spelling	●
Enable Caps Lock	○
Shortcuts	●
Predictive	●
Smart Punctuation	●
Enable Key Flicks	●
“.” Shortcut	●

Double tapping the space bar will insert a period followed by a space.

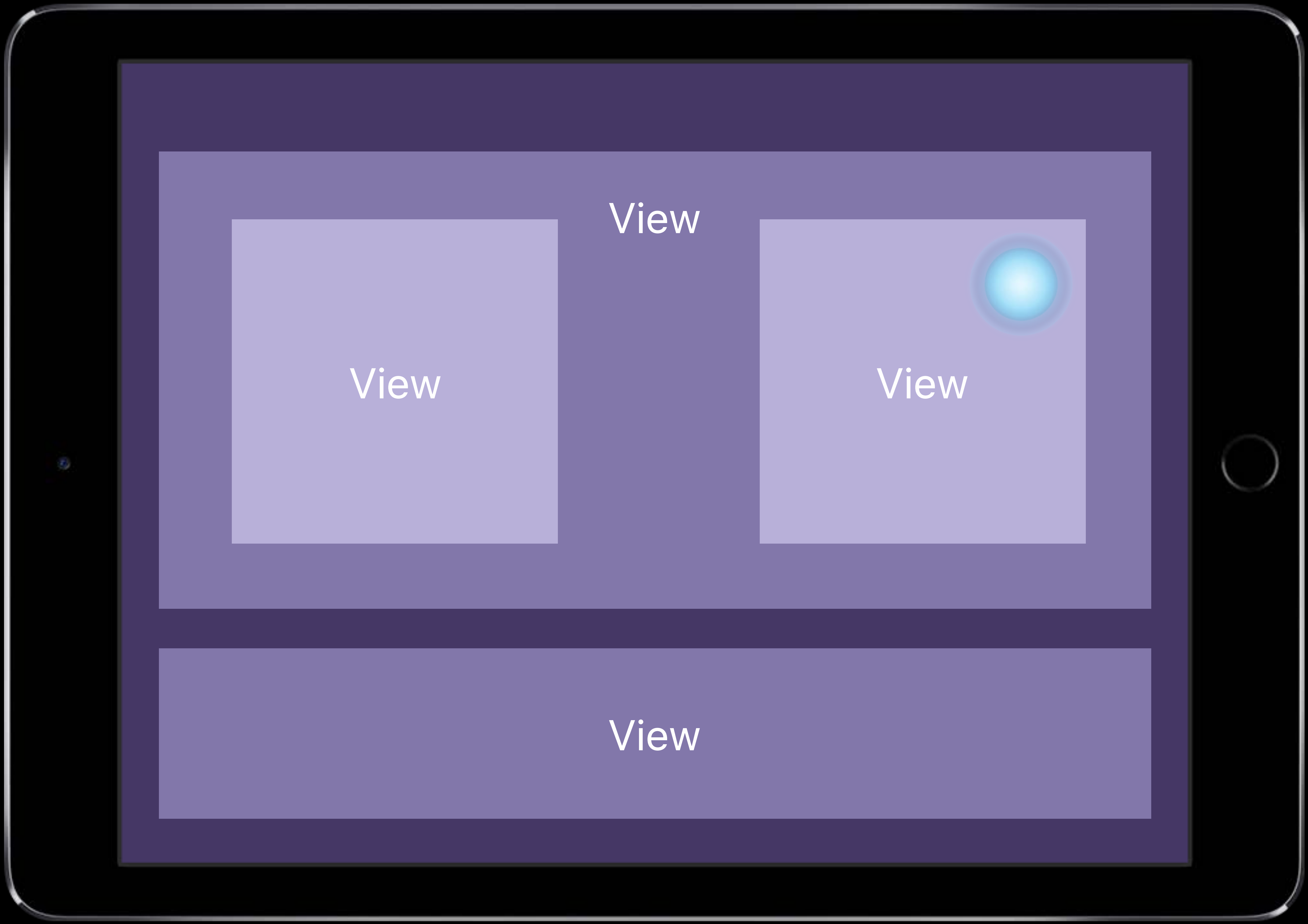
Enable Dictation	○

About Dictation and Privacy…

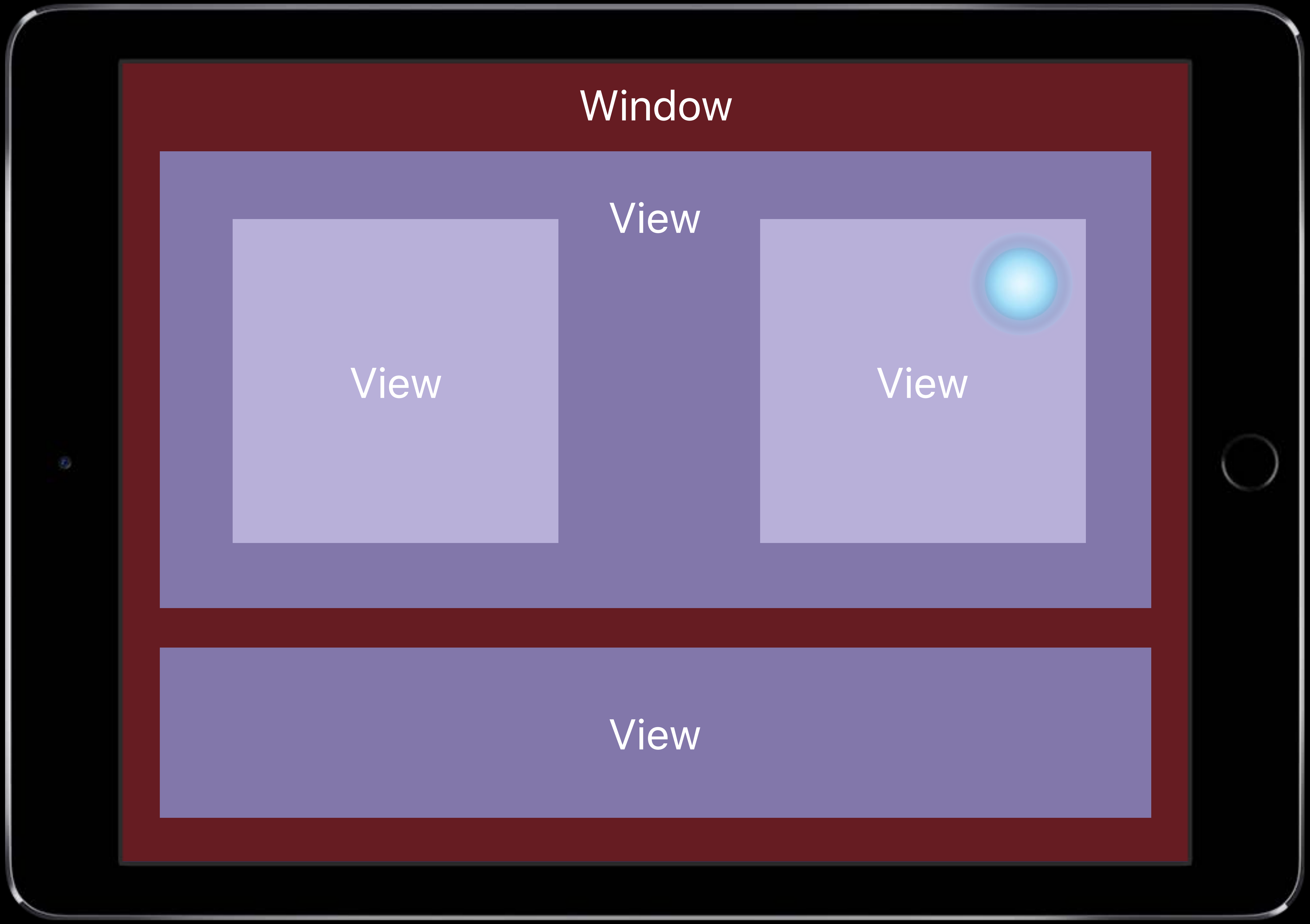
Hit Testing



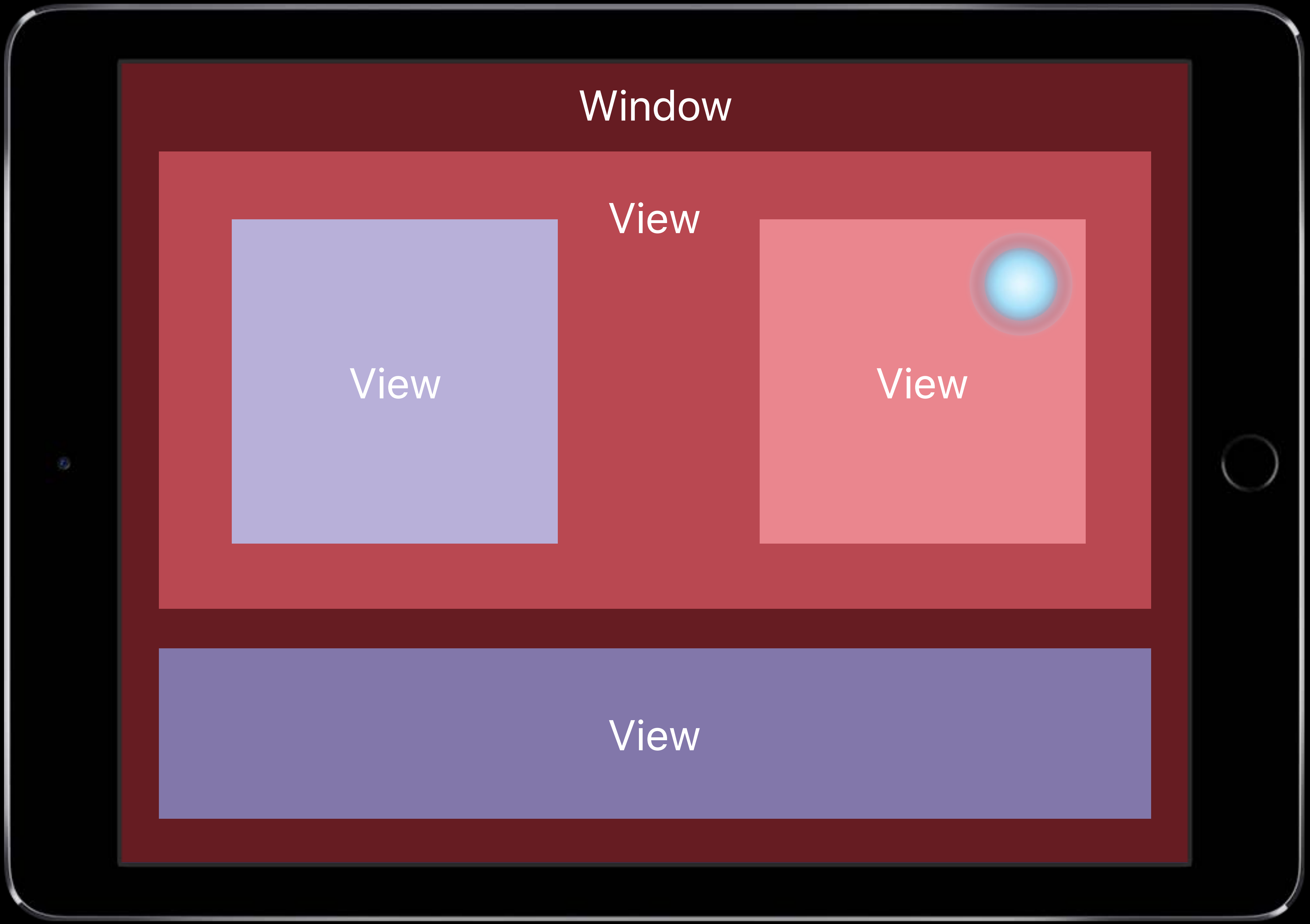
Hit Testing



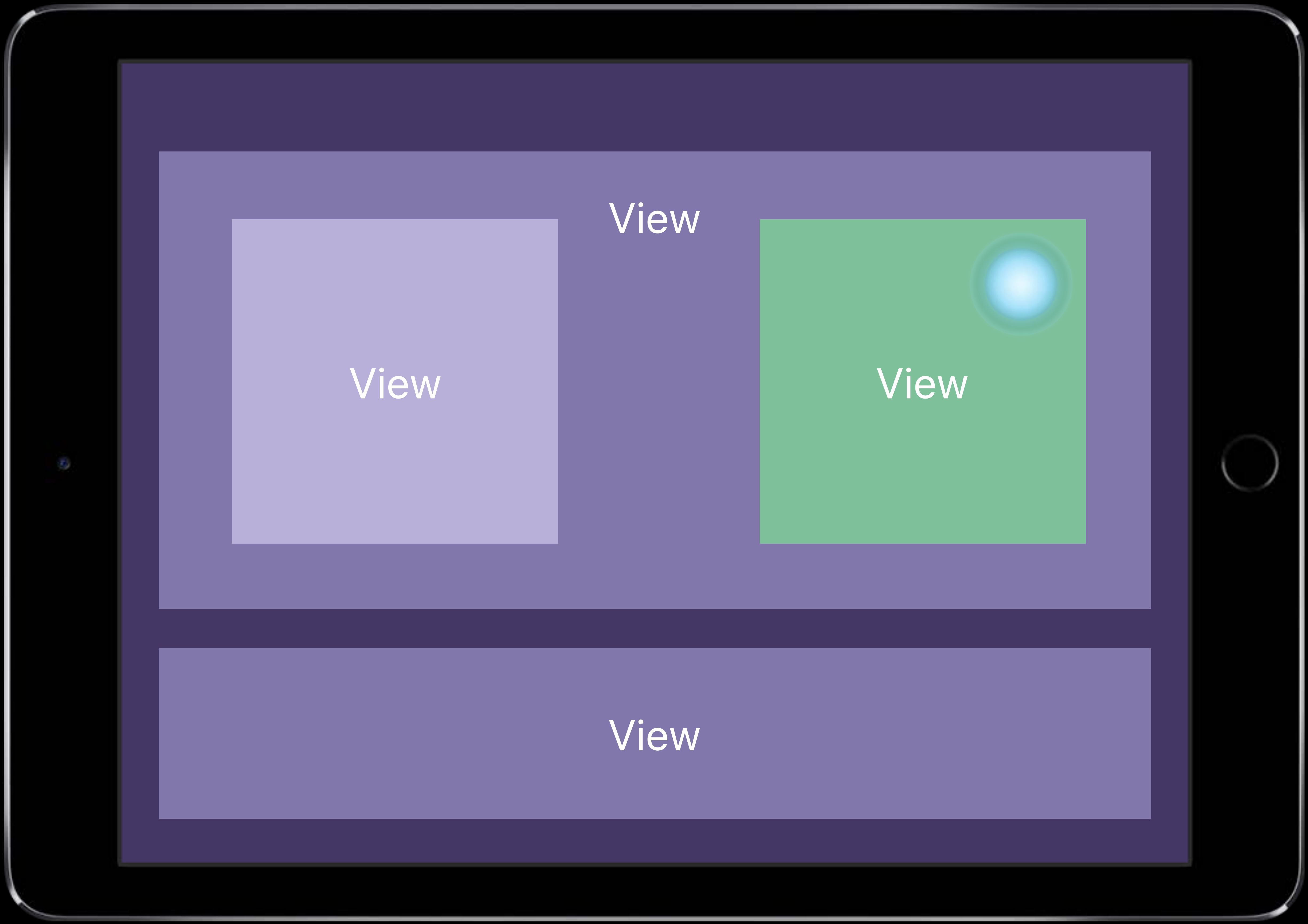
Hit Testing



Hit Testing



Hit Testing



```swift
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool

    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}
```

```
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool

    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}
```

```
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool

    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}
```

```swift
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool

    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}
```

```swift
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool

    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}
```

```
// UIView subclasses
open class UIView : NSObject {

    open func hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?
    open func point(inside point: CGPoint, with event: UIEvent?) -> Bool
}

class UIView : NSObject {

    var isUserInteractionEnabled: Bool

    var alpha: CGFloat
    var isHidden: Bool

    var isMultipleTouchEnabled: Bool
}

public struct UIViewAnimationOptions : OptionSet {

    public static var allowUserInteraction: UIViewAnimationOptions { get }
}
```

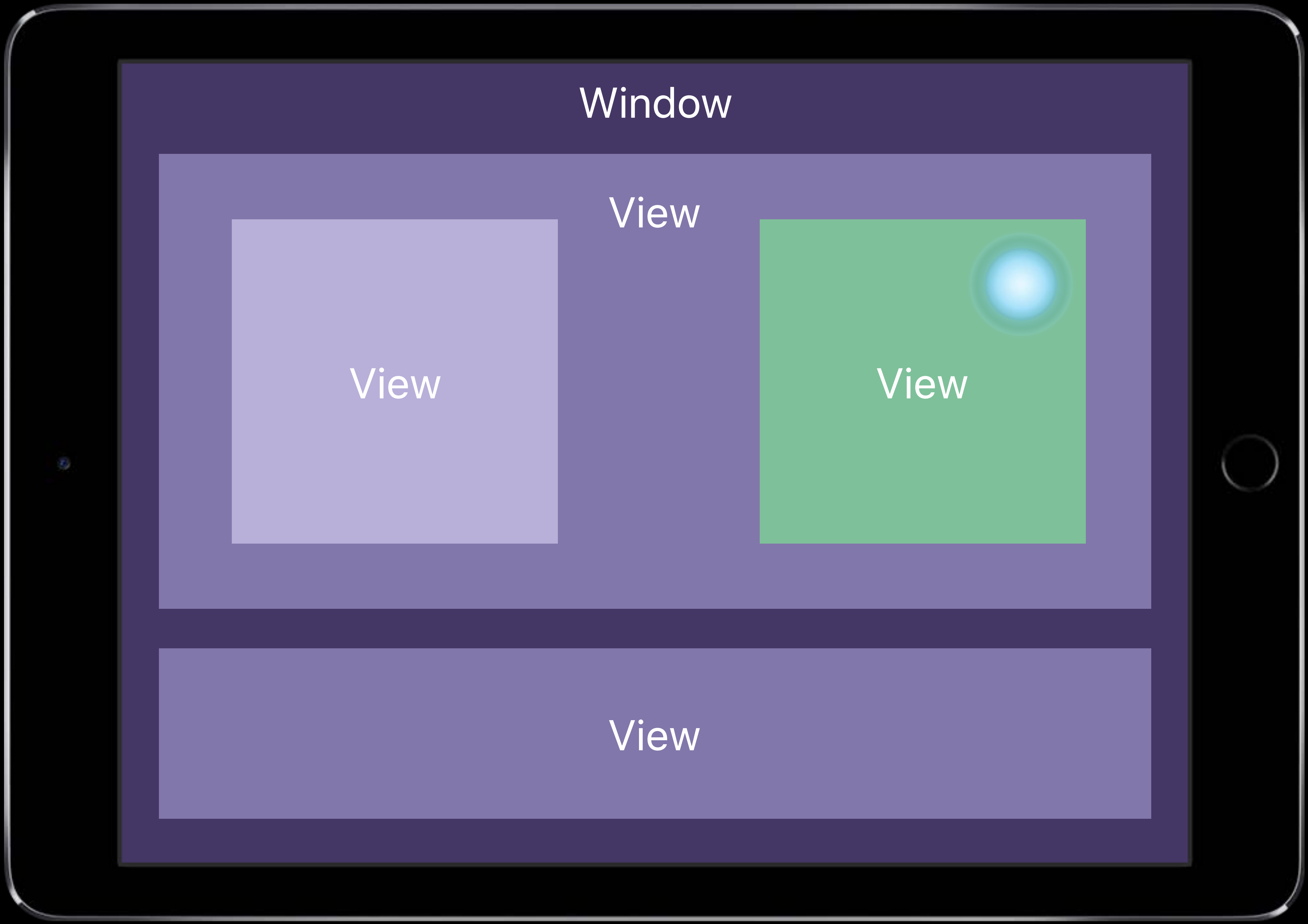
Hit Testing and Animations

Presentation layer vs model layer

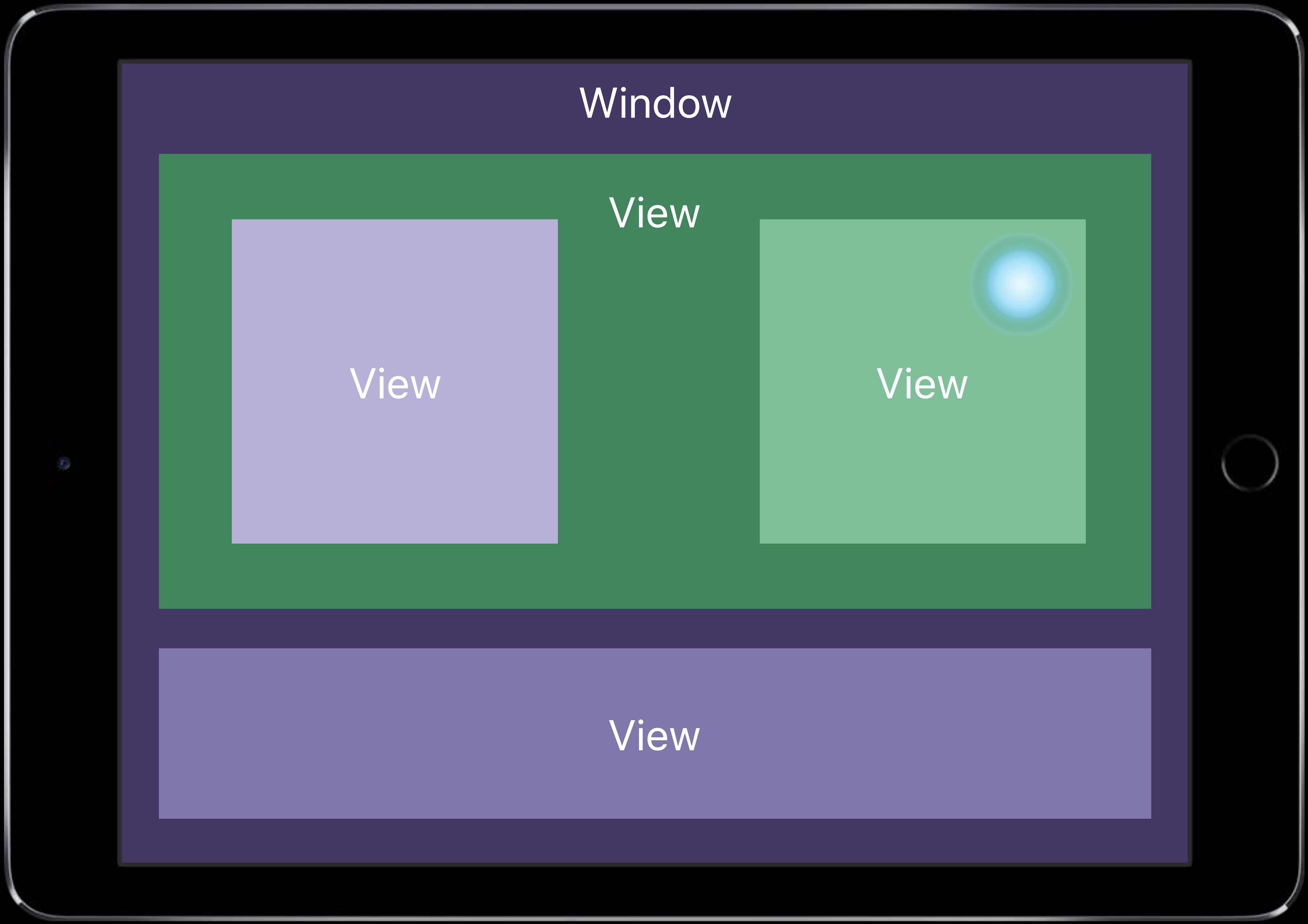
UIViewPropertyAnimator and `isManualHitTestingEnabled`

Building Interruptible and Responsive Interactions	WWDC 2014
Advances in UIKit Animations and Transitions	WWDC 2016

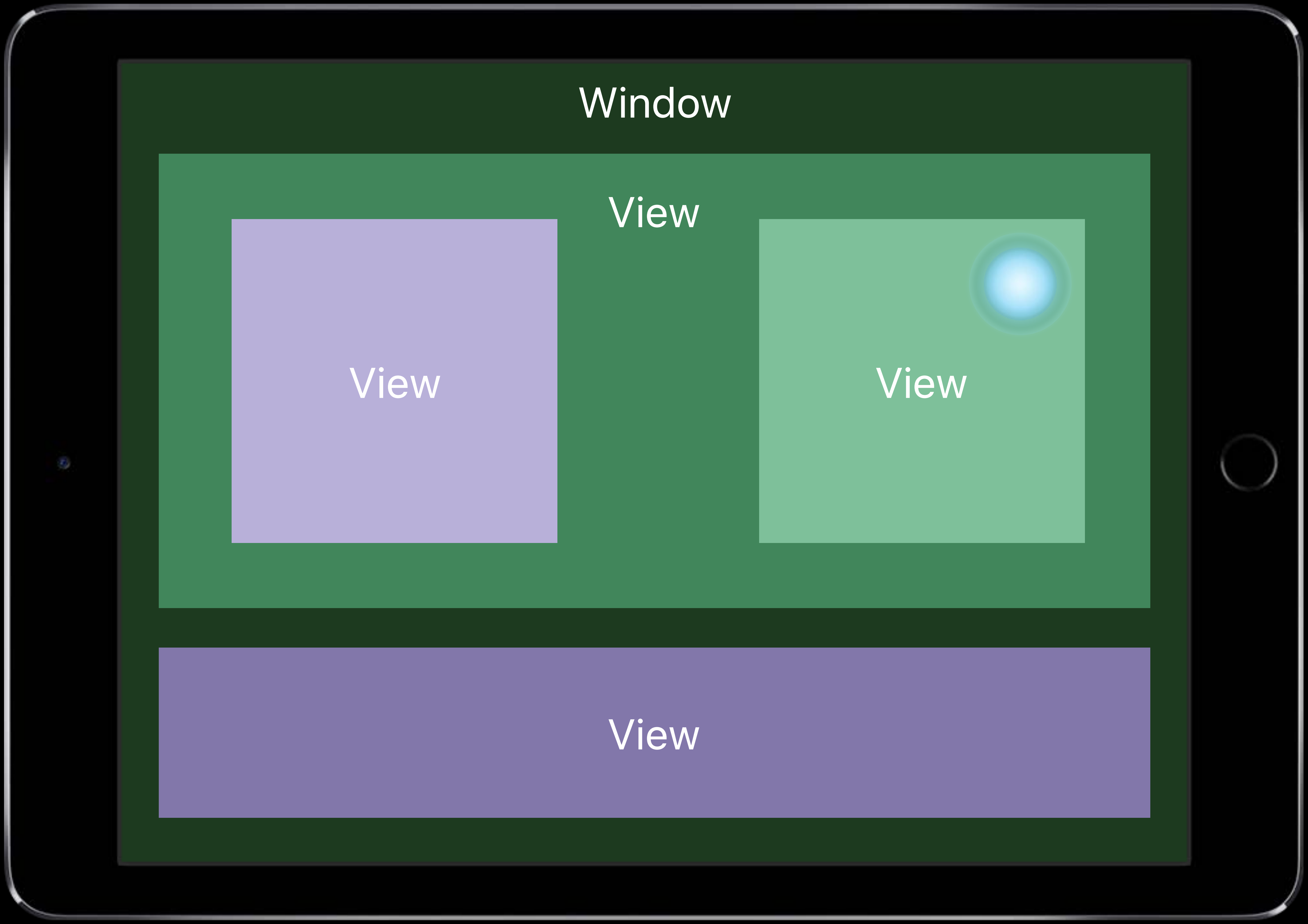
Relevant Gesture Recognizers



Relevant Gesture Recognizers



Relevant Gesture Recognizers



```
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldReceive touch: UITouch) -> Bool
    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer:
                                                      UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool
}
```

```
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldReceive touch: UITouch) -> Bool
    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer:
                                                      UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool
}
```

```
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldReceive touch: UITouch) -> Bool
    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer:
                                                      UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool
}
```

```
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldReceive touch: UITouch) -> Bool
    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer:
                                                      UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool
}
```

```
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldReceive touch: UITouch) -> Bool
    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer:
                                                      UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool
}
```

```
// Influencing participation in the interaction

public protocol UIGestureRecognizerDelegate : NSObjectProtocol {

    optional public func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                                             shouldReceive touch: UITouch) -> Bool
    optional public func gestureRecognizerShouldBegin(_ gestureRecognizer:
                                                      UIGestureRecognizer) -> Bool
}

class UIGestureRecognizer : NSObject {

    var isEnabled: Bool

    var allowedTouchTypes: [NSNumber]
    var requiresExclusiveTouchType: Bool
}
```

New on UIGestureRecognizer

NEW

Debugging only

```
@available(iOS 11.0, *)
open var name: String? // name for debugging to appear in logging
```

Debugging

Breakpoint opportunities

```
gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
                  shouldReceive touch: UITouch) -> Bool

touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)
```

Things to inspect

```
po touches.first?.gestureRecognizers

po event?.touches(for:someGestureRecognizer)

po touches.first?.view?.superview?.gestureRecognizers
```

Custom UIGestureRecognizers

Begin late and fail fast!

Ignore touches `ignore(_ touch:, for event:)`

Don’t forget `touchesCancelled(_:with:)`

Gesture Recognizer System

Gesture Recognizer System

Revisit your setups

Gesture Recognizer System

Revisit your setups

Exclusion and failure requirements

Gesture Recognizer System

Revisit your setups

Exclusion and failure requirements

Are your gesture recognizers on the right views?

System Gesture Interaction

Glen Low, UIKit Engineer

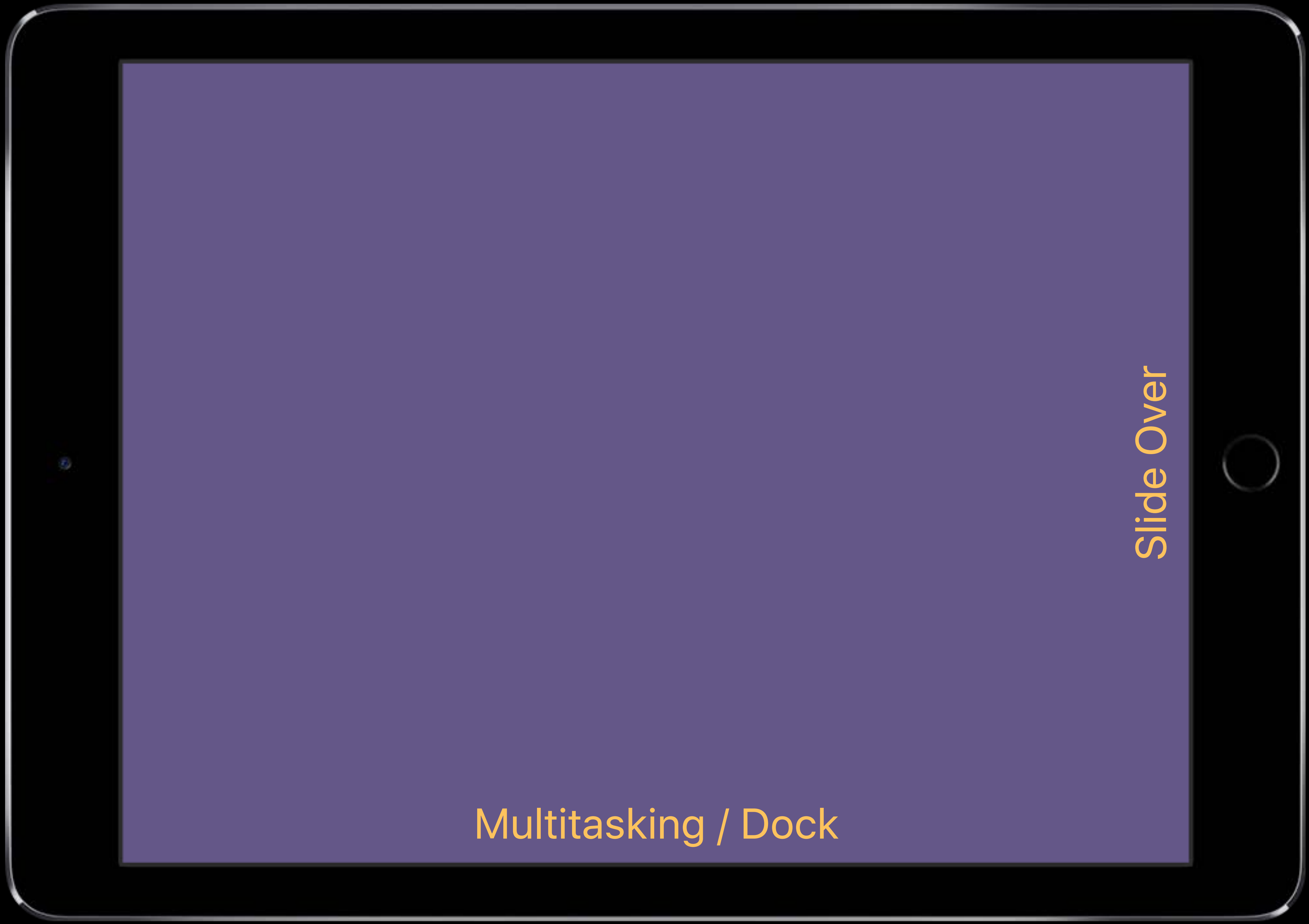
2:49

Cover Sheet or Killjoy?

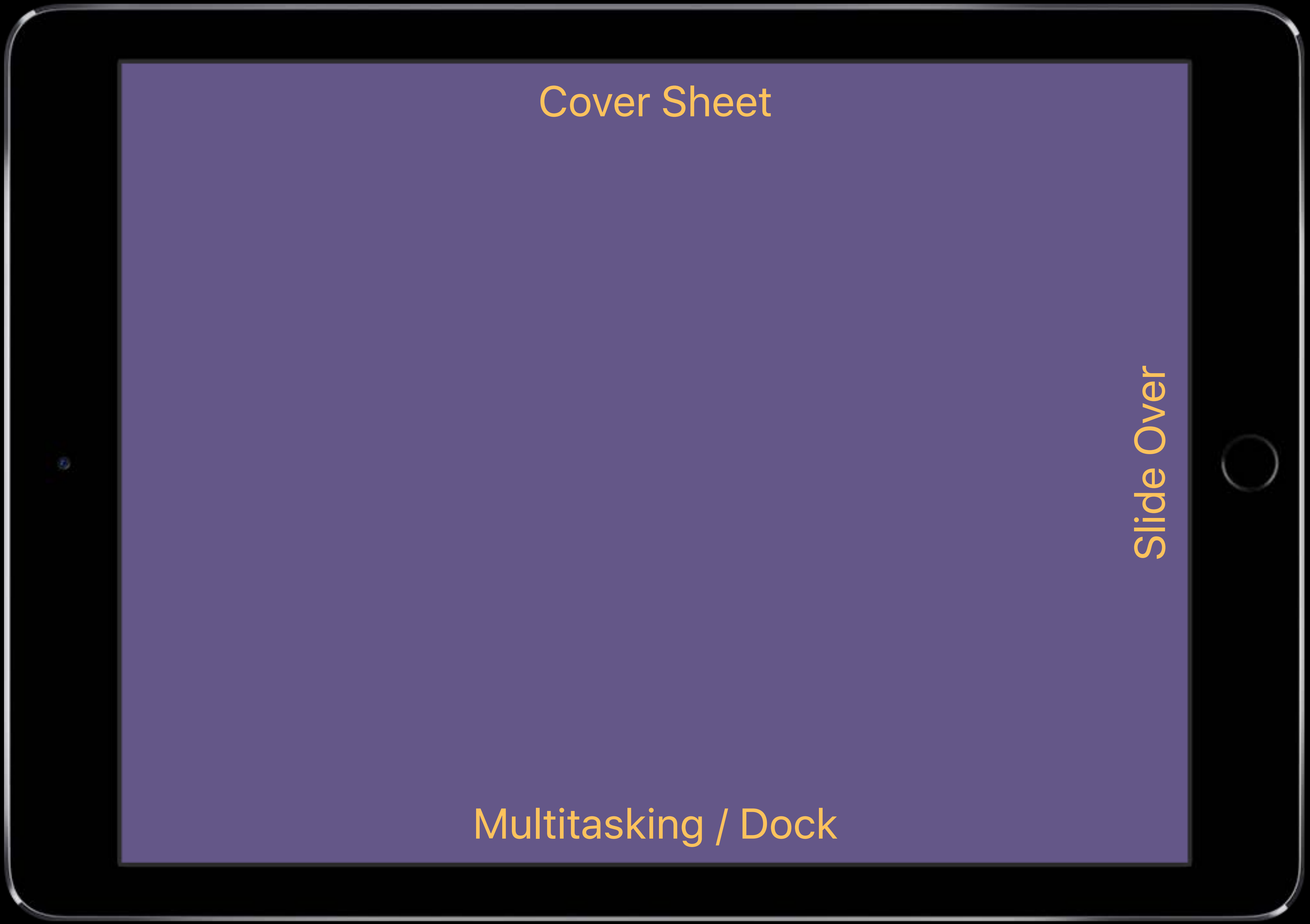
Life on the Edges

Life on the Edges

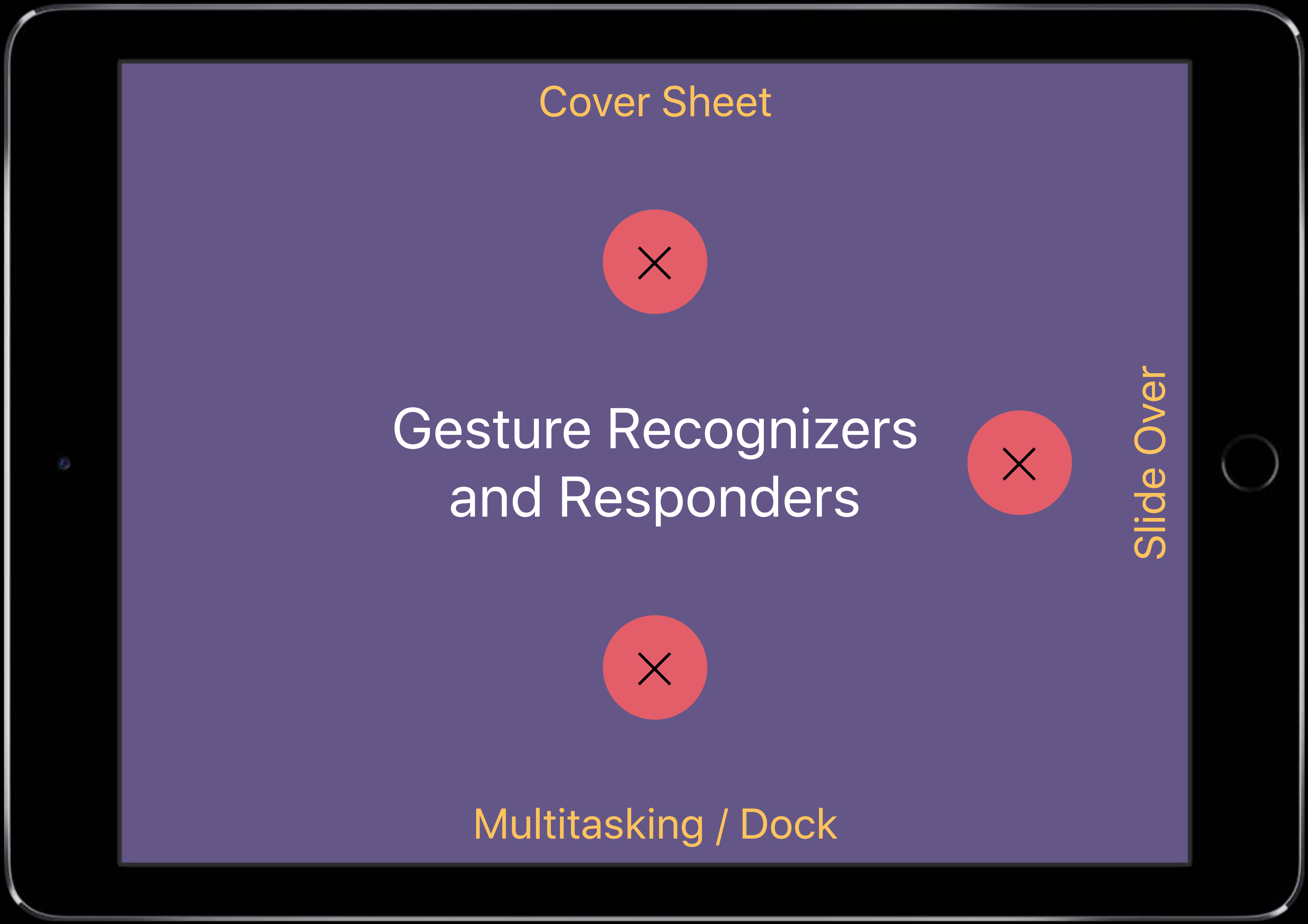
Life on the Edges



Life on the Edges



Life on the Edges



Who Gets the Touches?

System ↔ App	Tap Pinch Rotate Long Press

Who Gets the Touches?

System ↔ App	Tap Pinch Rotate Long Press
• System • App	Pan Swipe Responders

System gesture or app gesture?

We made an educated guess.

NEW

System gesture or app gesture?

You tell us.

Let’s Talk Terms

NEW

```
class MyViewController: UIViewController {

    // override to return which screen edges to defer system gestures
    override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
        return deferControlCenter ? .bottom : UIRectEdge()
    }

    // call whenever your method would return a different screen edge
    var deferControlCenter : Bool {
        didSet { setNeedsUpdateOfScreenEdgesDeferringSystemGestures() }
    }

}
```

Let’s Talk Terms

NEW

```
class MyViewController: UIViewController {

    // override to return which screen edges to defer system gestures
    override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
        return deferControlCenter ? .bottom : UIRectEdge()
    }

    // call whenever your method would return a different screen edge
    var deferControlCenter : Bool {
        didSet { setNeedsUpdateOfScreenEdgesDeferringSystemGestures() }
    }

}
```

Let’s Talk Terms

NEW

```
class MyViewController: UIViewController {

    // override to return which screen edges to defer system gestures
    override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
        return deferControlCenter ? .bottom : UIRectEdge()
    }

    // call whenever your method would return a different screen edge
    var deferControlCenter : Bool {
        didSet { setNeedsUpdateOfScreenEdgesDeferringSystemGestures() }
    }

}
```

Speaking in Tongues

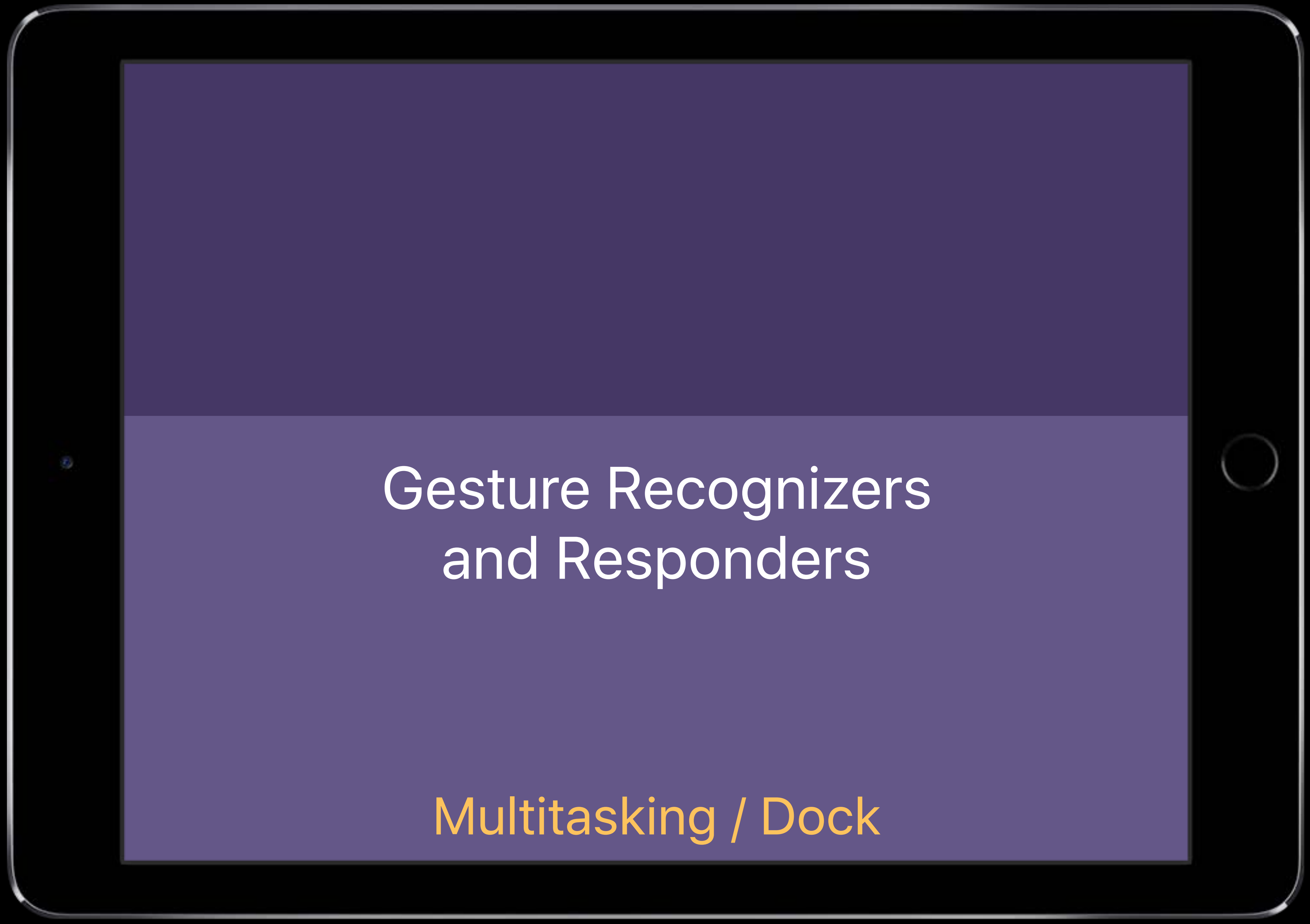
Gesture Recognizers
and Responders

```
override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
    return deferControlCenter ? .bottom : UIRectEdge()
}
```

Speaking in Tongues

```
override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
    return deferControlCenter ? .bottom : UIRectEdge()
}
```

Speaking in Tongues



```
override func preferredScreenEdgesDeferringSystemGestures() -> UIRectEdge {
    return deferControlCenter ? .bottom : UIRectEdge()
}
```

For Containers Only

NEW

```swift
class MyContainerViewController: UIViewController {

    // override to return which child view controller to consult
    override func childViewControllerForScreenEdgesDeferringSystemGestures()
        -> UIViewController? {
        return mySelectedChildViewController
    }

}
```

Don’t Do It, Because…

Don’t mess with the familiar

Your recognizers may already get the touches

Telling us ≠ us doing it

Don’t Do It, Because…

Don’t mess with the familiar

Your recognizers may already get the touches

Telling us ≠ us doing it

Your app is only used casually

Playing Nice With Drag and Drop

Michael Turner, UIKit Engineer

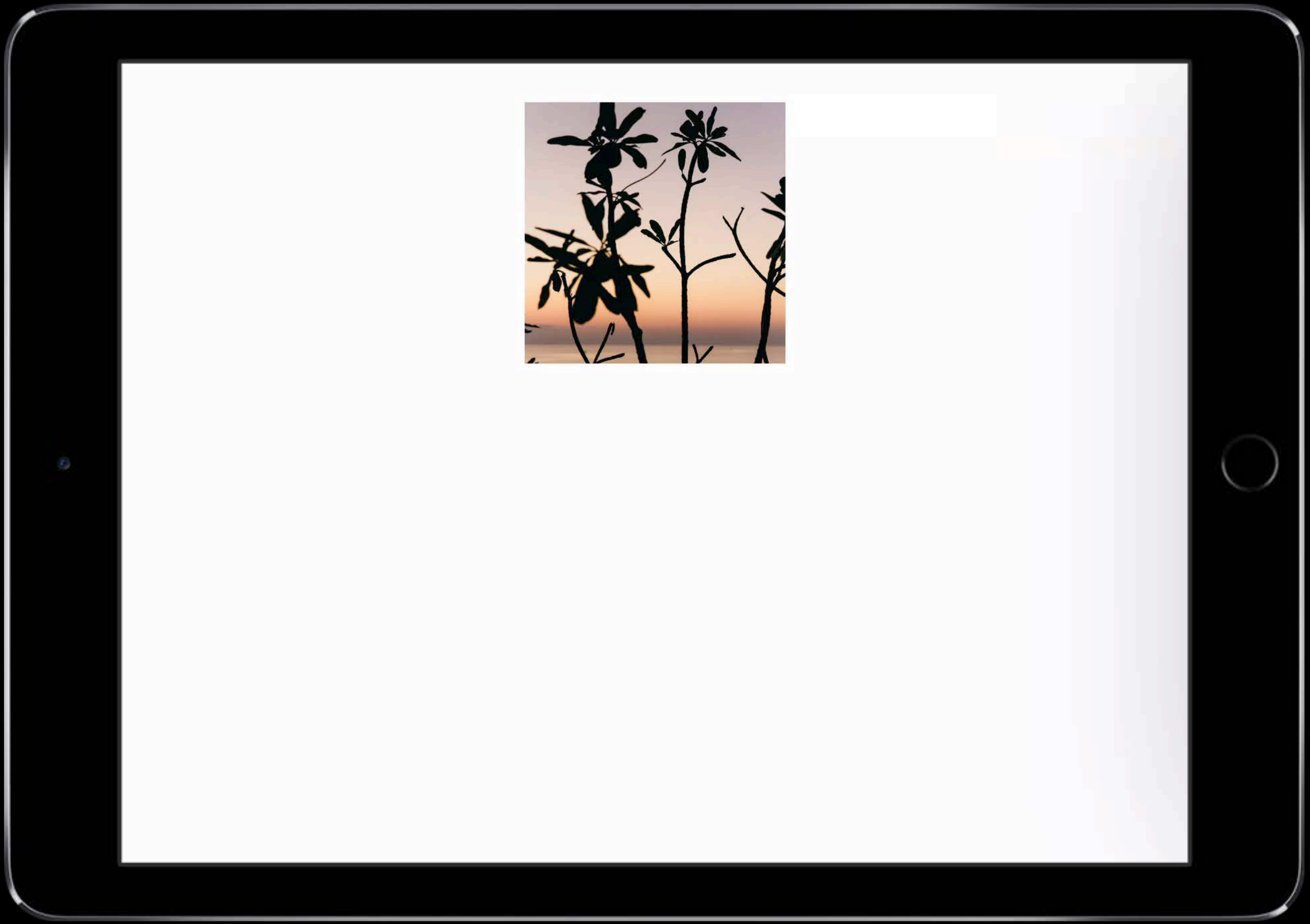
UIDragInteraction

NEW

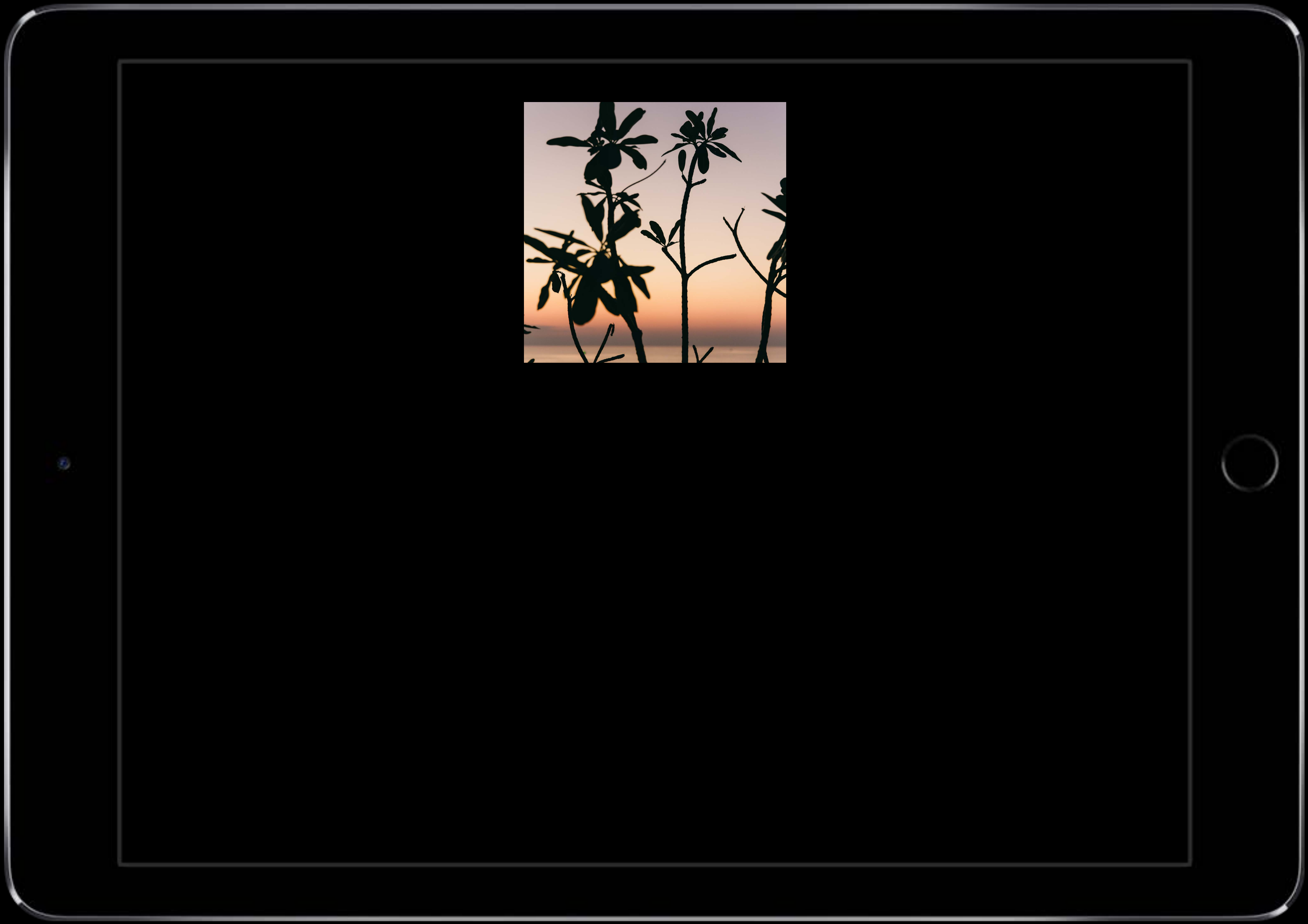
Adding UIDragInteraction to a UIView within your app is easy

```
let dragInteraction = UIDragInteraction(delegate: myDelegate)
myView.addInteraction(dragInteraction)
```

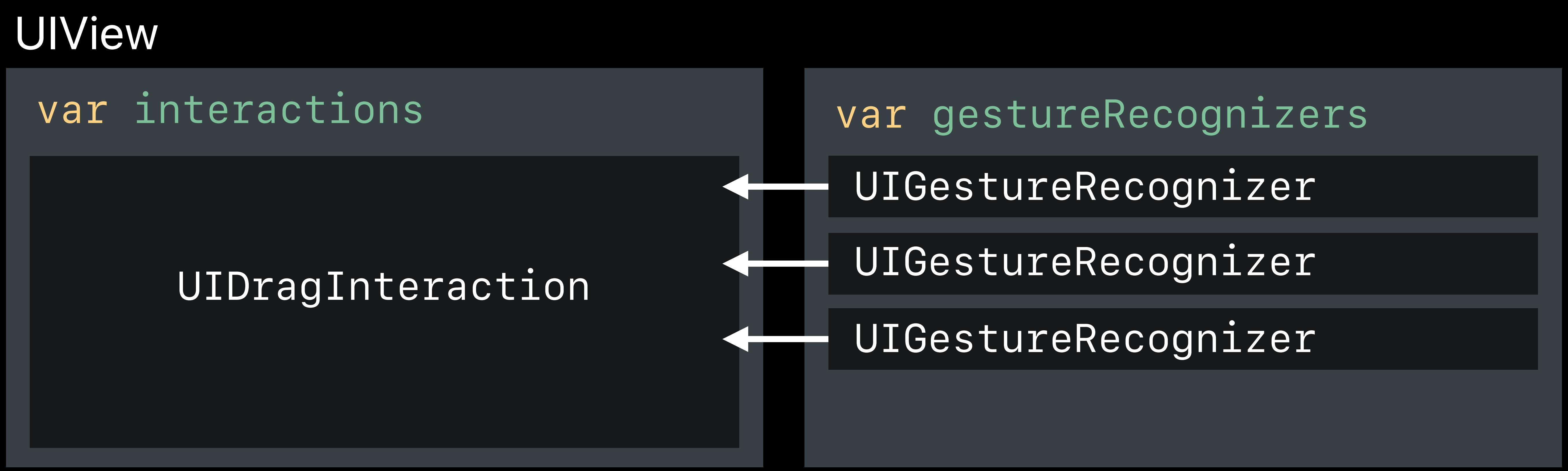
Initiating a Drag



Initiating a Drag



UIDragInteraction Example



UIView

var interactions

UIDragInteraction

var gestureRecognizers

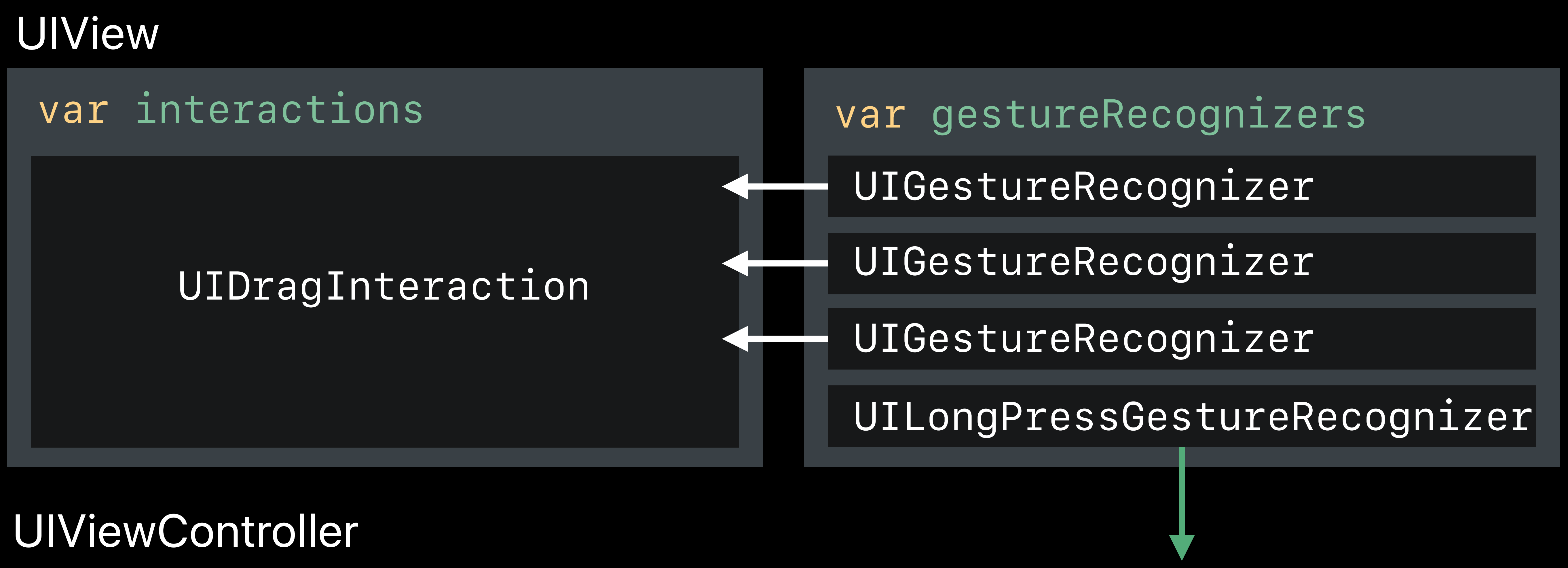
UIGestureRecognizer

UIGestureRecognizer

UIGestureRecognizer

UIView

UIDragInteraction Example



UIView

var interactions

UIDragInteraction

var gestureRecognizers

UIGestureRecognizer

UIGestureRecognizer

UIGestureRecognizer

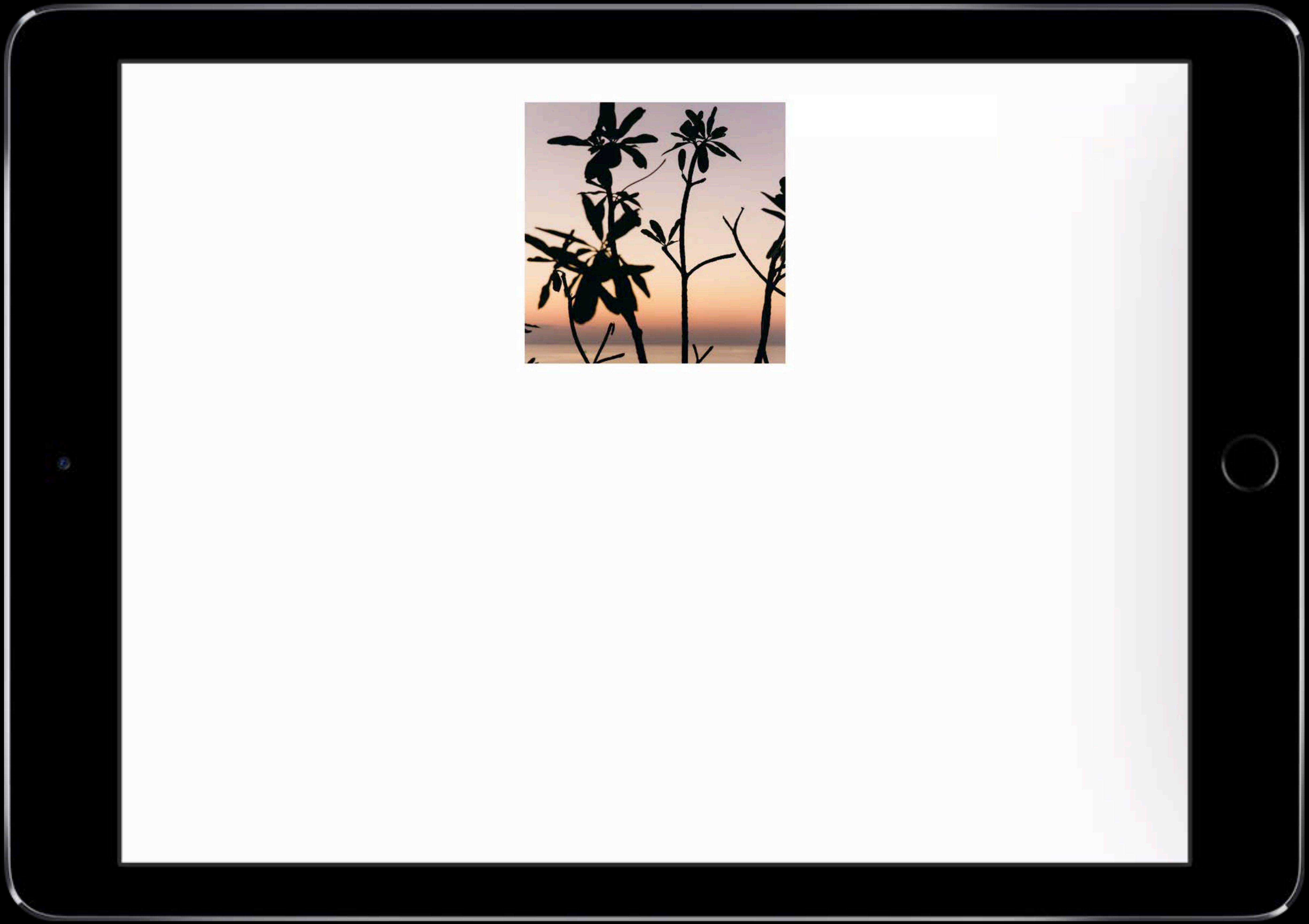
UILongPressGestureRecognizer

UIViewController

```
func longPress(gesture: UIGestureRecognizer) {
     switch gesture.state {
     case .began:
         presentActivityViewController()
     case .cancelled:
         dismissActivityViewController()
     default: break
     }
}
```

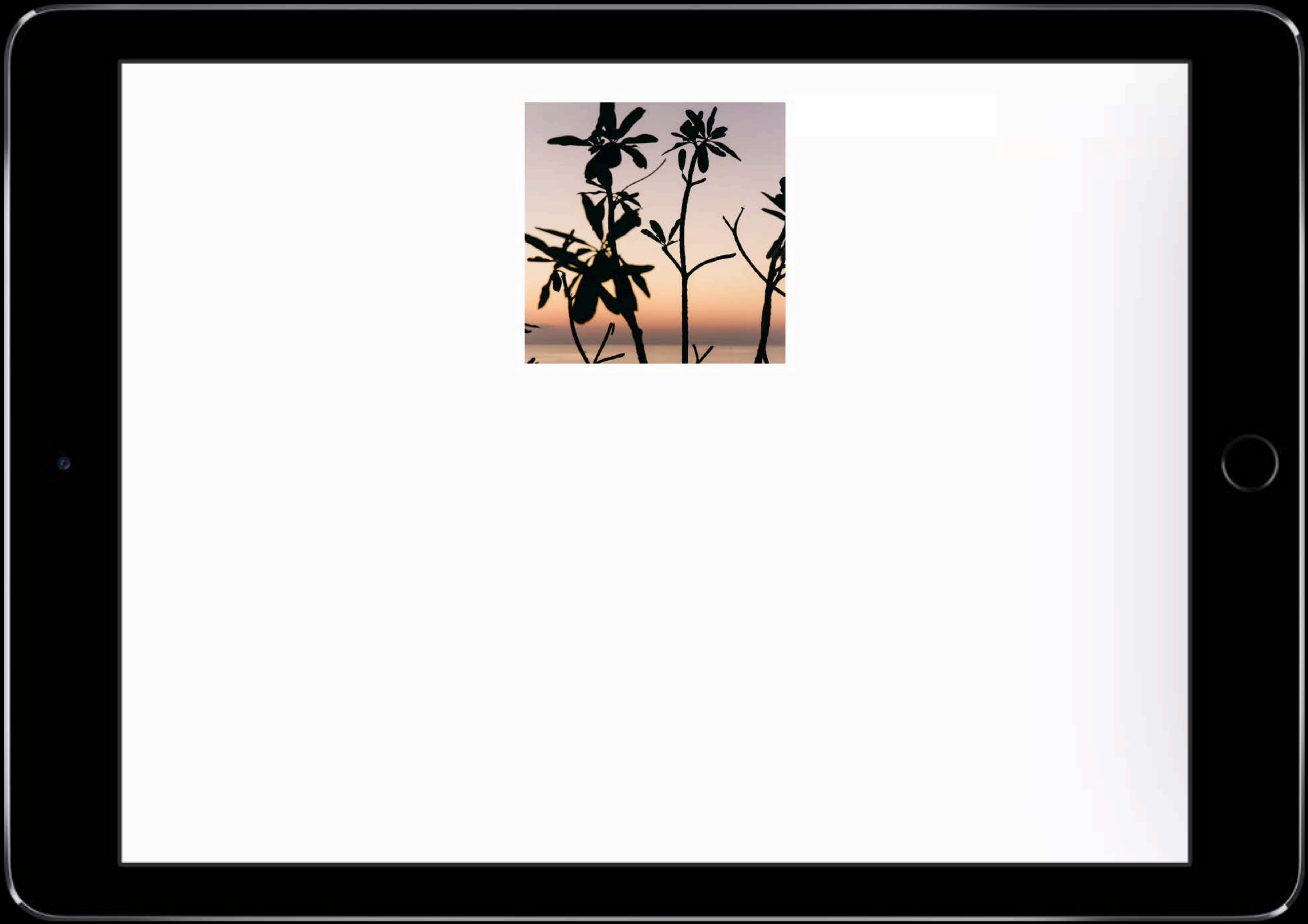
UIView

Long Press and Move



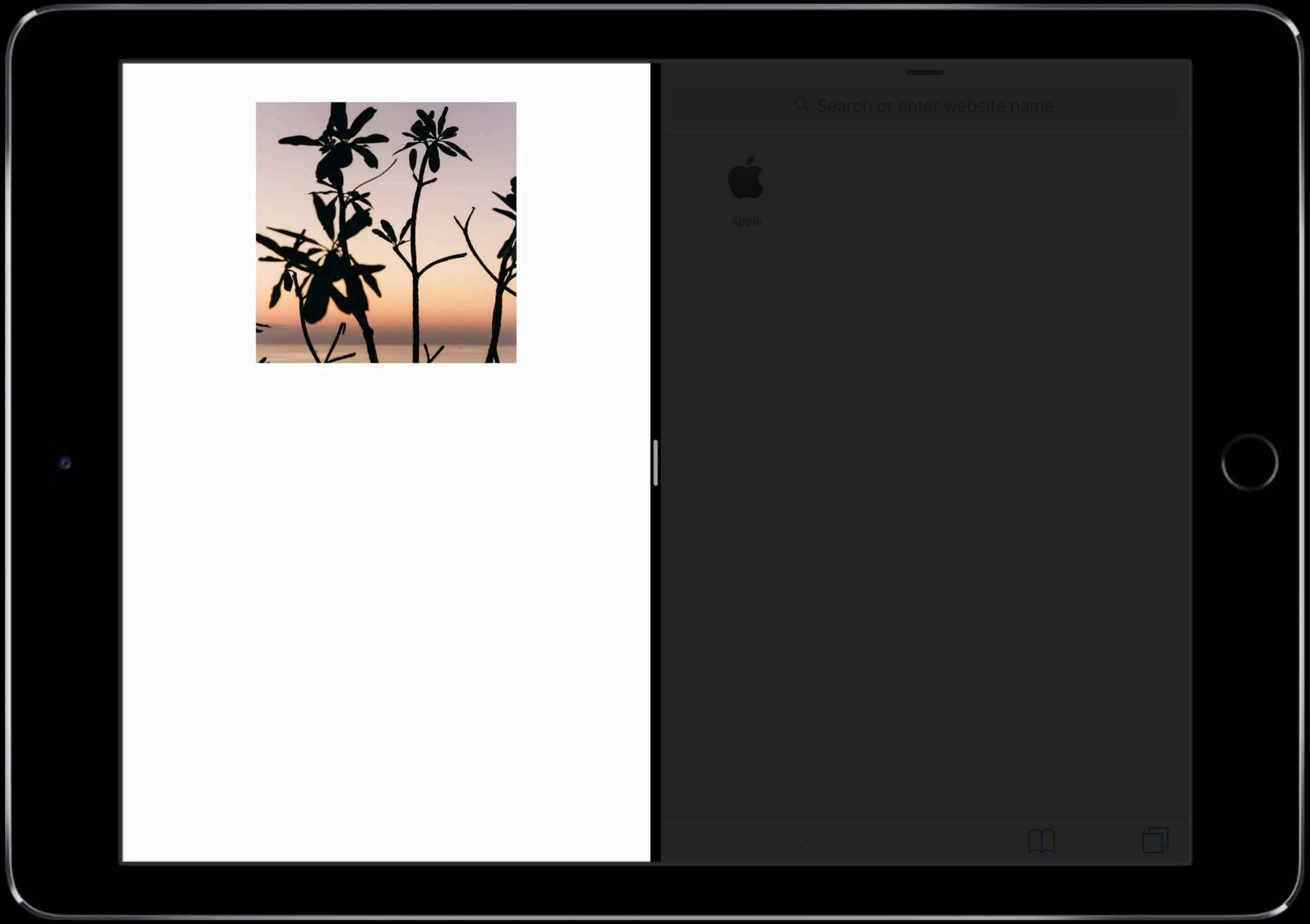
UILongPressGestureRecognizers are delayed

Long Press, Hold, and Move

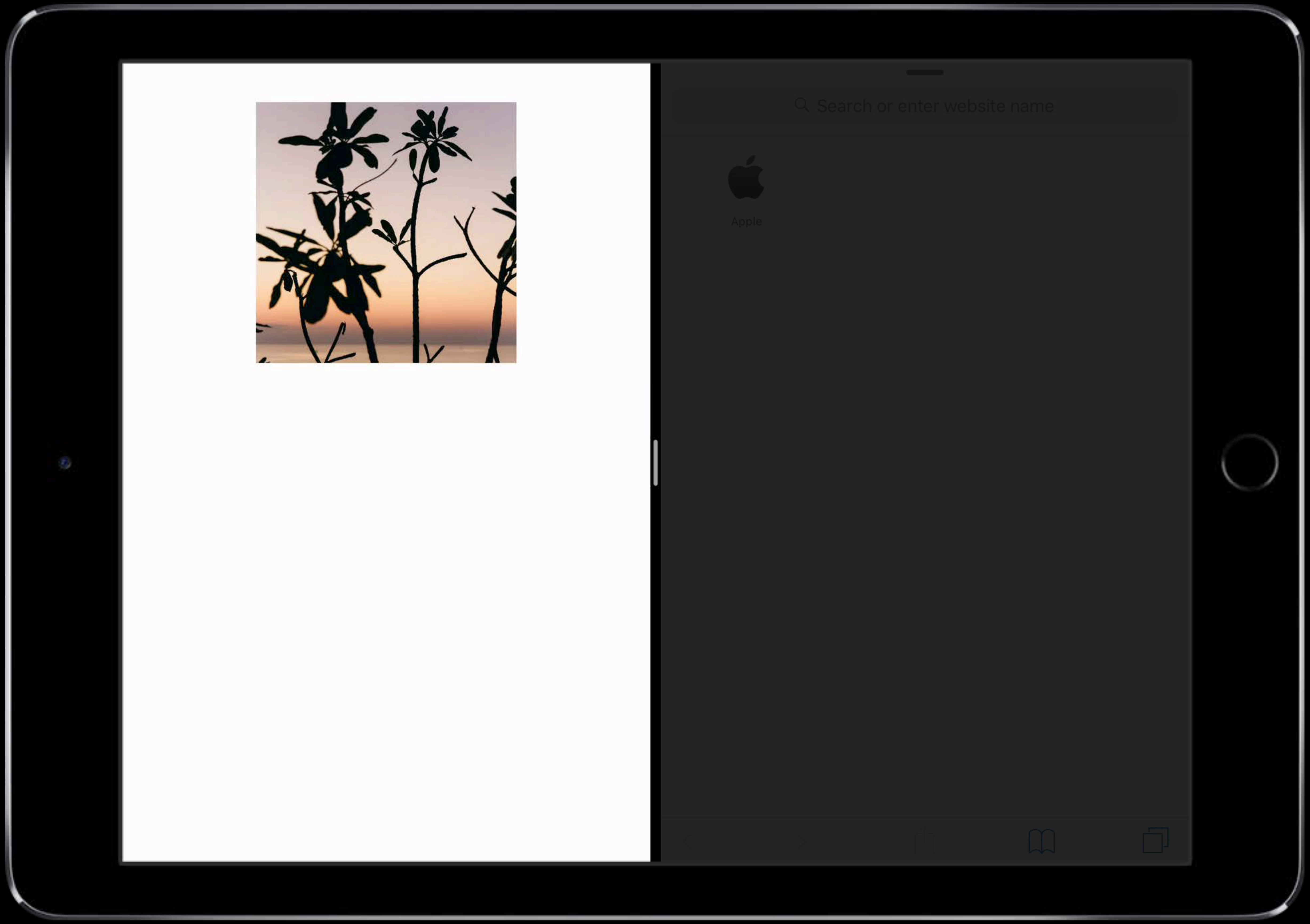


A beginning drag will
cancel the touch

Compact Trait Environment

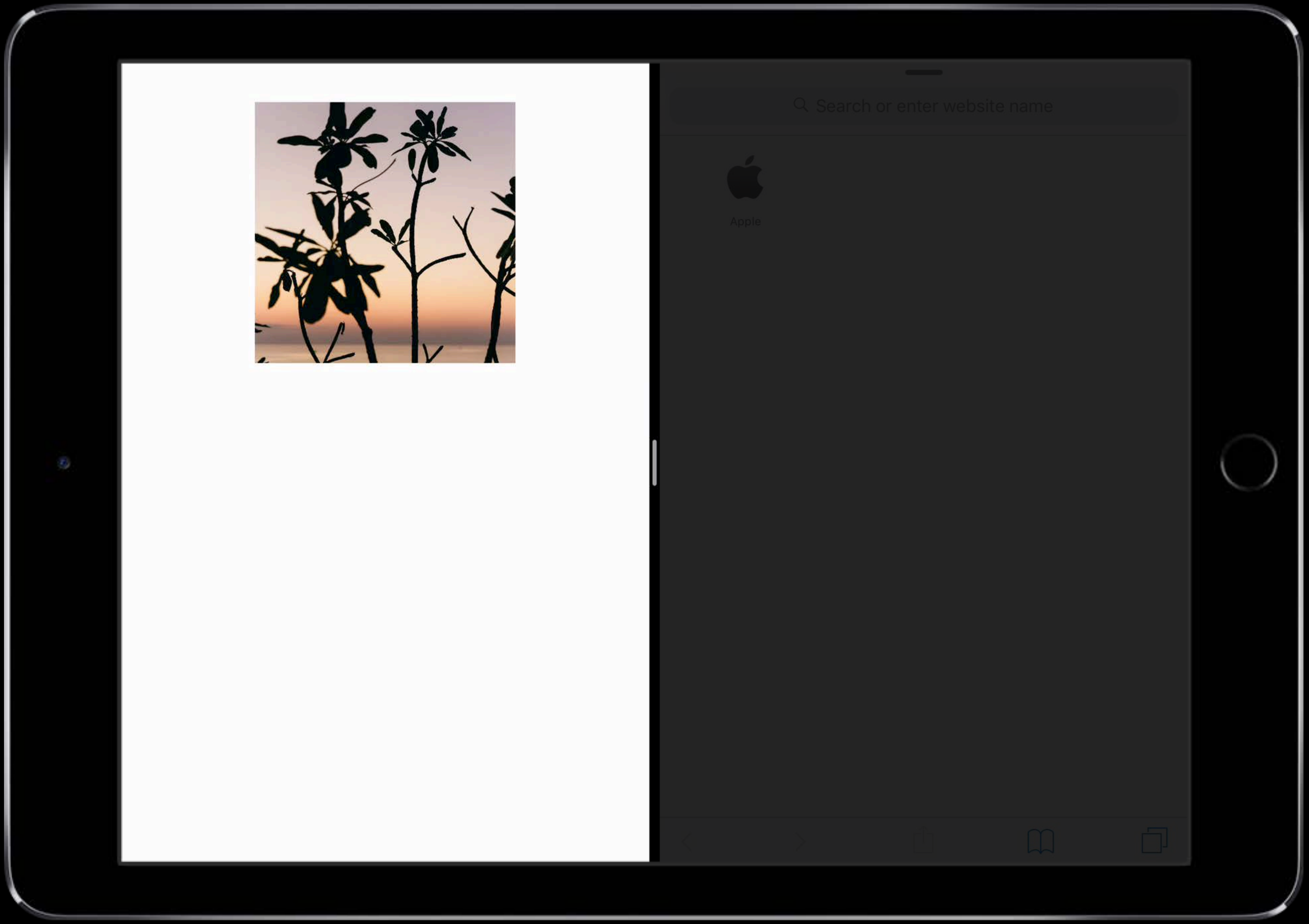


Long Press, Hold, and Move

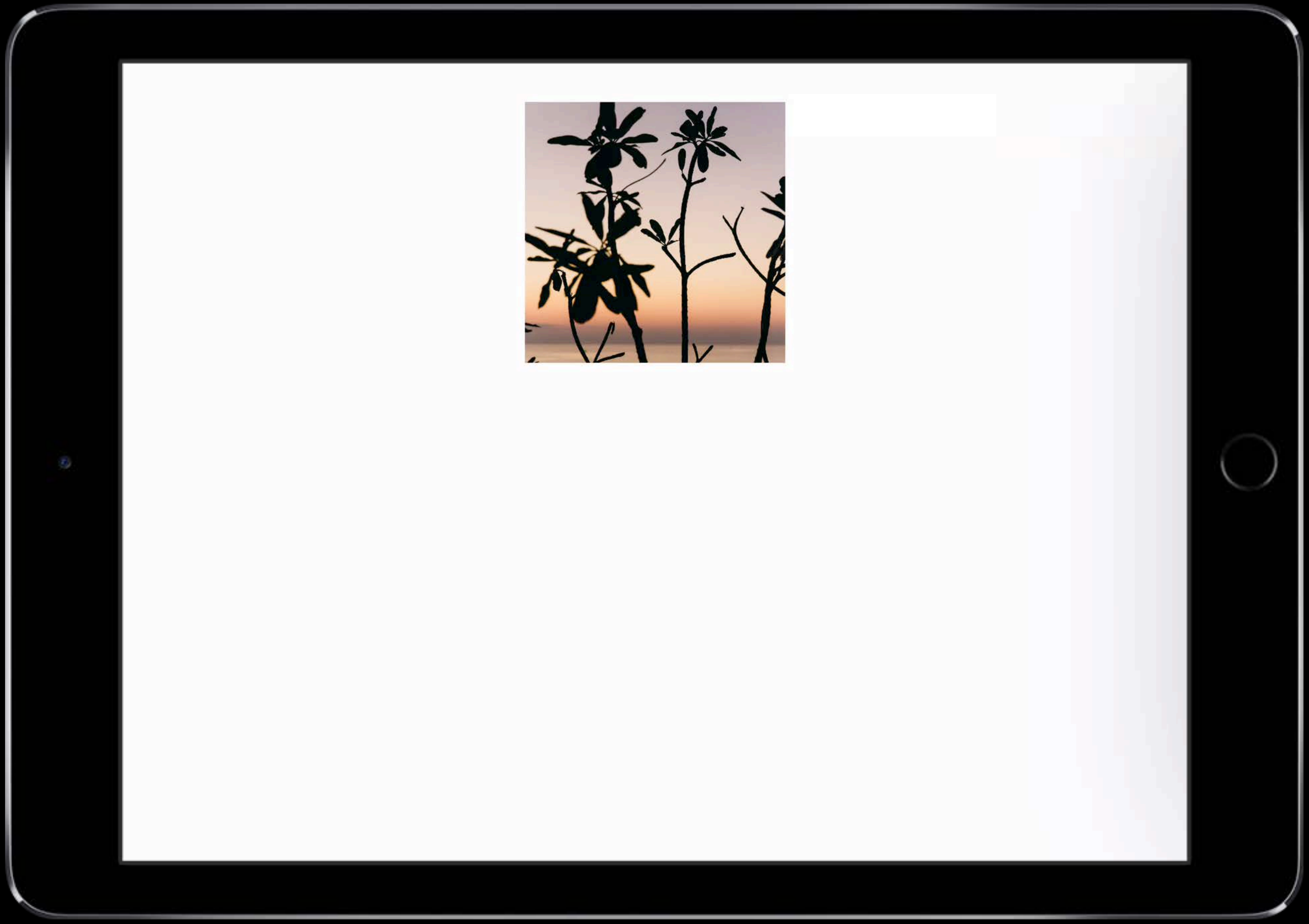


Long presses are delayed
until the touch ends

Long Press and Lift



Adding to a Drag



Adding to a Drag

Control this behavior using UIDragInteractionDelegate

```
func dragInteraction(_ interaction: UIDragInteraction, itemsForAddingTo session:
UIDragSession, withTouchAt point: CGPoint) -> [UIDragItem] {
    // Returning 0 items allows normal touch processing to continue
    return []
}
```

How to Adapt for UIDragInteraction

Examine your existing actions

Present modal UI carefully

Handle the `.cancelled` state

Your app is interactive during a drag!

Summary

Leveraging the gesture system

Use the new deferred system gesture APIs

Working alongside UIDragInteraction

Summary

Leveraging the gesture system

Use the new deferred system gesture APIs

Working alongside UIDragInteraction

Summary

Leveraging the gesture system

Use the new deferred system gesture APIs

Working alongside UIDragInteraction

More Information

https://developer.apple.com/wwdc17/219

Past Sessions

Building Advanced Gesture Recognizers	WWDC 2012
Making the Most of Multi-Touch on iOS	WWDC 2011
Leveraging Touch Input on iOS	WWDC 2016
Building Interruptible and Responsive Interactions	WWDC 2014
Advanced Scrollviews and Touch Handling Techniques	WWDC 2014

Related Sessions

Introducing Drag and Drop	Hall 3	Tuesday 11:20AM
What’s New in Cocoa Touch	Hall 3	Tuesday 10:20AM
Updating Your App for iOS 11	Hall 3	Tuesday 4:10PM
Mastering Drag and Drop	Grand Ballroom B	Wednesday 11:00AM
Advanced Animations with UIKit	Hall 3	Thursday 3:10PM

Labs

Cocoa Touch and Haptics Lab	Technology Lab C	Fri 12:00PM
UIKit and Collection View Lab	Technology Lab B	Thu 10:00AM

WWDC17