App Frameworks #WWDC16

Mastering UIKit on tvOS

Session 210

Justin Voss UIKit Engineer

Agenda

Agenda

Event Handling

Agenda

Event Handling

Layered Images

Agenda

Event Handling

Layered Images

Scrolling

Agenda

Event Handling

Layered Images

Scrolling

Text Input

Event Handling

Event Handling Best Practices

Event Handling Best Practices

Navigation should rely on the focus engine

Event Handling Best Practices

Navigation should rely on the focus engine

Use gesture recognizers when possible

Event Handling Best Practices

Navigation should rely on the focus engine

Use gesture recognizers when possible

Avoid gestures that can’t be used on all input devices

Touches

Touches

UITouch represents contact between the user's finger and the touch surface

Touches

UITouch represents contact between the user's finger and the touch surface

- Just like the iOS version

Touches

UITouch represents contact between the user's finger and the touch surface

- Just like the iOS version
- UITouchTypeIndirect

Touches

UITouch represents contact between the user's finger and the touch surface

- Just like the iOS version
- UITouchTypeIndirect
- Begins centered in the focused view

Touches

UITouch represents contact between the user's finger and the touch surface

- Just like the iOS version
- UITouchTypeIndirect
- Begins centered in the focused view
- No absolute coordinates on the touch surface

Presses

Presses

UIPress represents the up or down state of a physical button

Presses

UIPress represents the up or down state of a physical button

- May be pressure-sensitive

Presses

UIPress represents the up or down state of a physical button

- May be pressure-sensitive
- UIGestureRecognizer

Presses

UIPress represents the up or down state of a physical button

- May be pressure-sensitive
- UIGestureRecognizer
- UIPress events mimic UITouch events

Presses

UIPress represents the up or down state of a physical button

- May be pressure-sensitive
- UIGestureRecognizer
- UIPress events mimic UITouch events

```
func pressesBegan(_ presses: Set<UIPress>, with event: UIPressesEvent?) {}
func pressesChanged(_ presses: Set<UIPress>, with event: UIPressesEvent?) {}
func pressesEnded(_ presses: Set<UIPress>, with event: UIPressesEvent?) {}
func pressesCancelled(_ presses: Set<UIPress>, with event: UIPressesEvent?) {}
```

Press Types

An illustrated guide

Press Types

An illustrated guide

Press Types

An illustrated guide

Press Types

An illustrated guide

Press Types

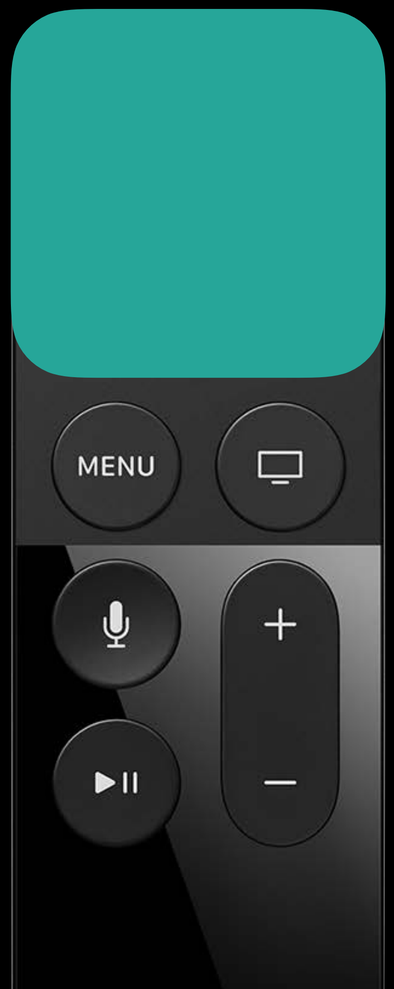
An illustrated guide

`UIPressTypeSelect`

Press Types

An illustrated guide

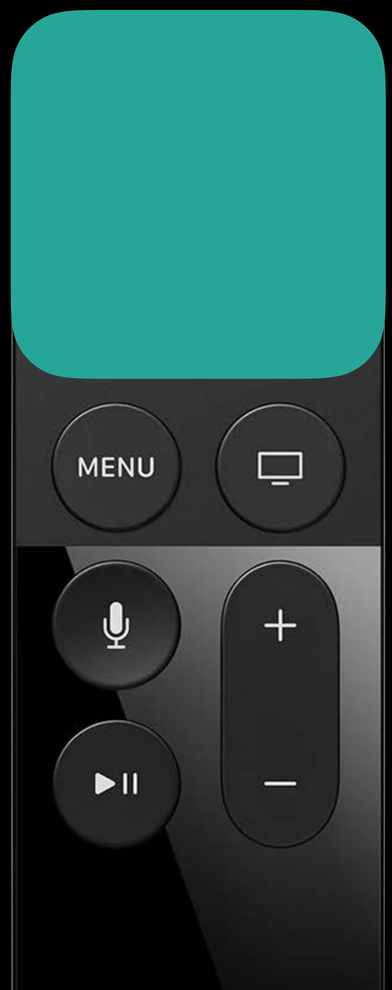
`UIPressTypeSelect`



Press Types

An illustrated guide

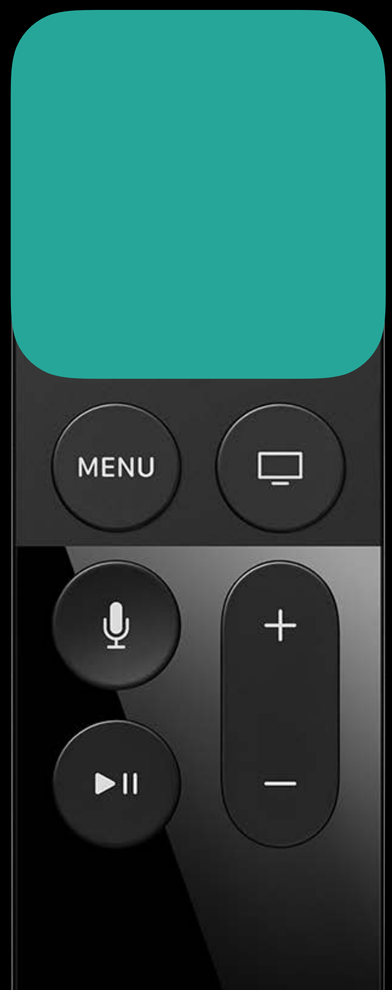
`UIPressTypeSelect`



Press Types

An illustrated guide

`UIPressTypeSelect`

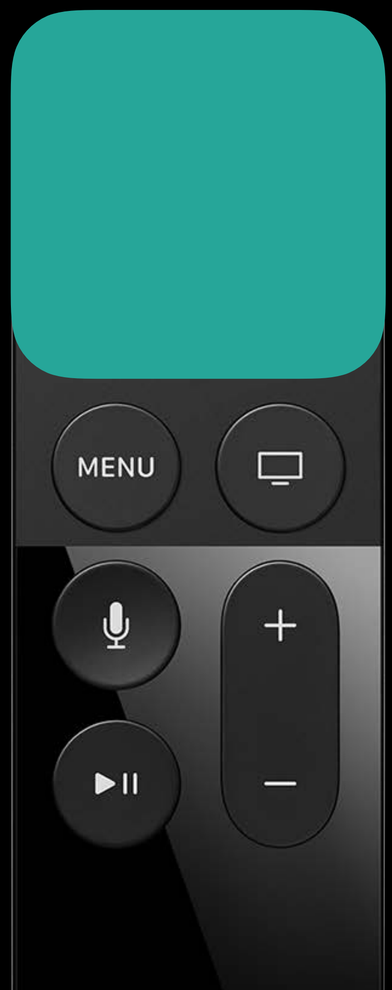


Press Types

An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`

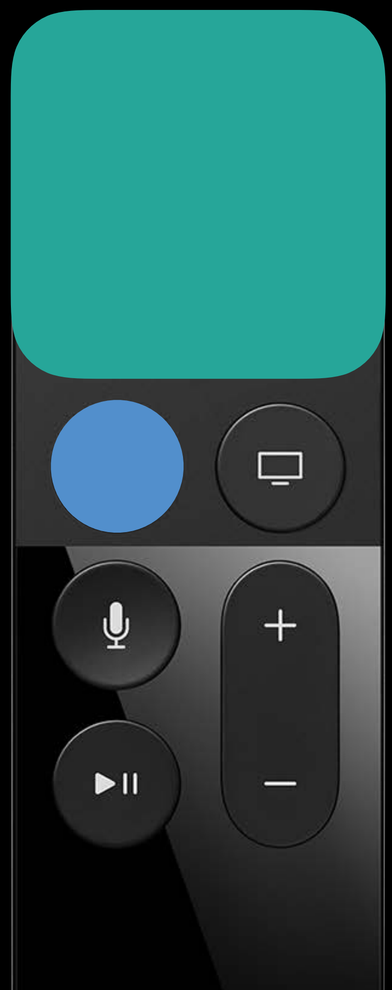


Press Types

An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`

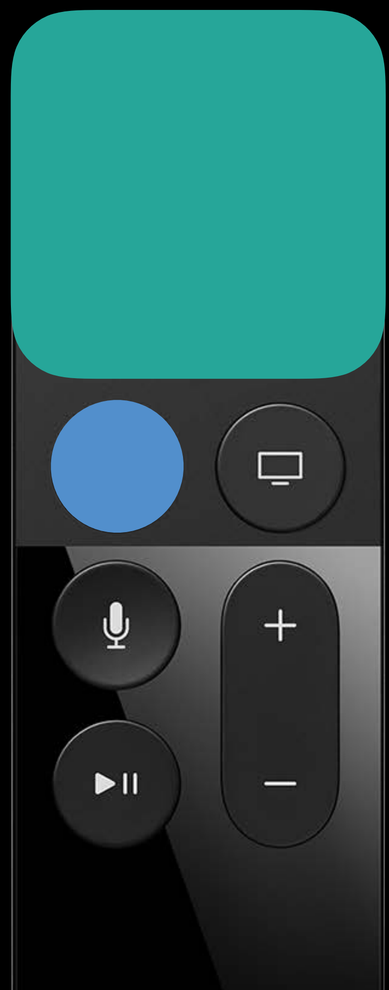


Press Types

An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`

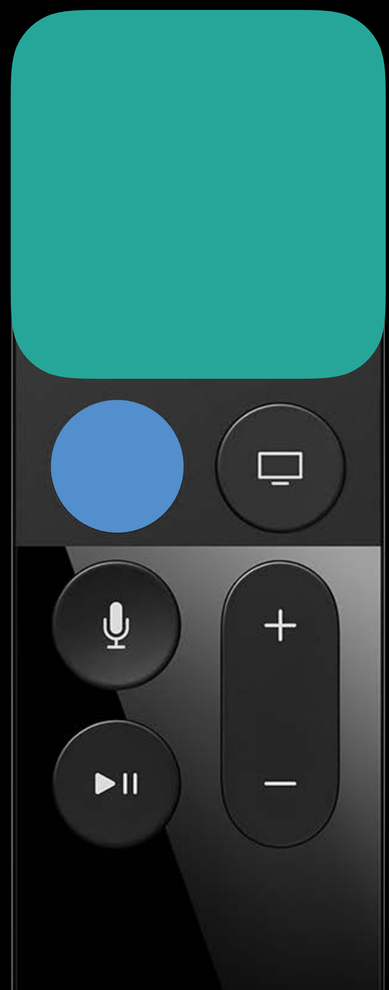


Press Types

An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`

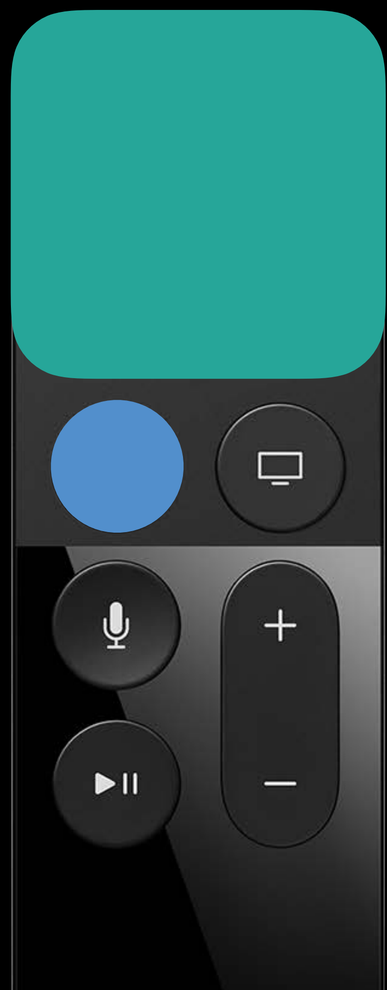


Press Types

An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`



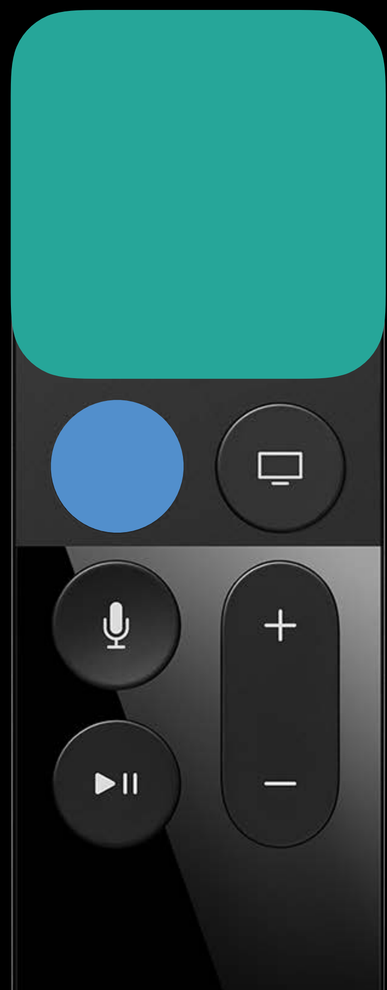
Press Types

An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`

`UIPressTypePlayPause`



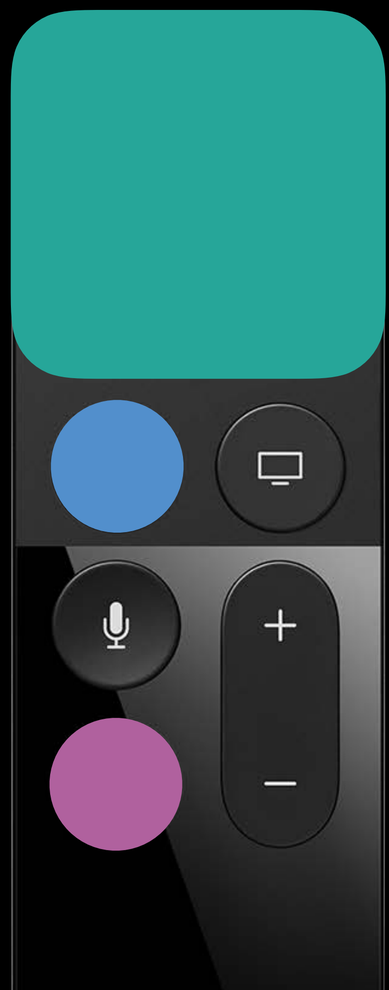
Press Types

An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`

`UIPressTypePlayPause`



Press Types

An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`

`UIPressTypePlayPause`

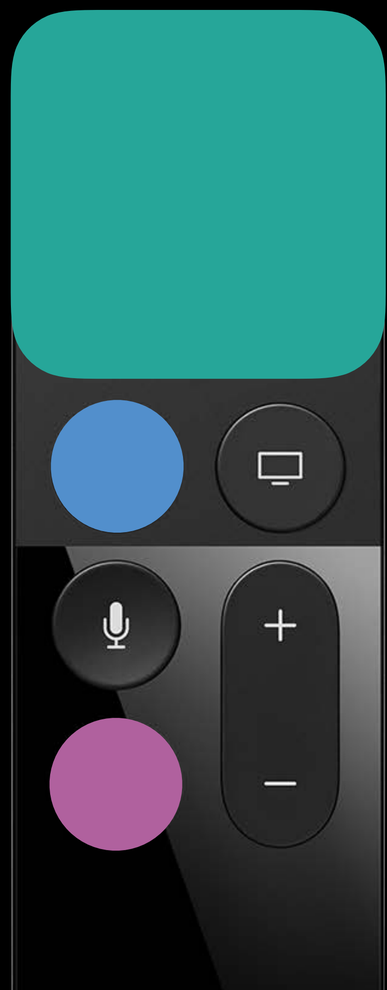
Press Types

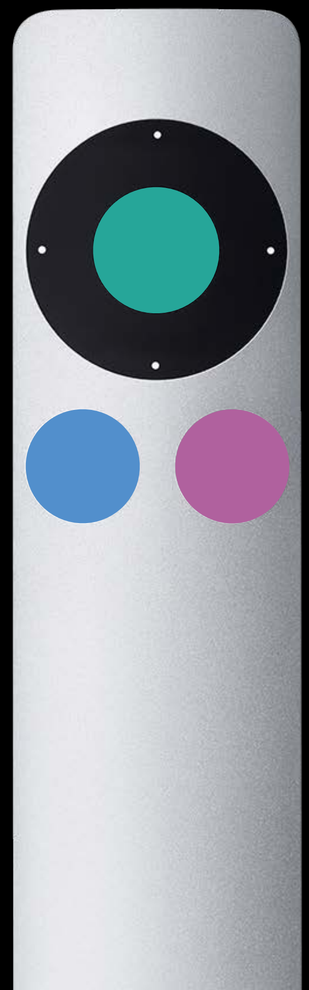
An illustrated guide

`UIPressTypeSelect`

`UIPressTypeMenu`

`UIPressTypePlayPause`





Press Types

An illustrated guide

- `UIPressTypeUpArrow`
- `UIPressTypeDownArrow`
- `UIPressTypeLeftArrow`
- `UIPressTypeRightArrow`

Press Types

An illustrated guide

- `UIPressTypeUpArrow`
- `UIPressTypeDownArrow`
- `UIPressTypeLeftArrow`
- `UIPressTypeRightArrow`

Press Types

An illustrated guide

- `UIPressTypeUpArrow`
- `UIPressTypeDownArrow`
- `UIPressTypeLeftArrow`
- `UIPressTypeRightArrow`

Press Types

An illustrated guide

- `UIPressTypeUpArrow`
- `UIPressTypeDownArrow`
- `UIPressTypeLeftArrow`
- `UIPressTypeRightArrow`

Press Types

An illustrated guide

- `UIPressTypeUpArrow`
- `UIPressTypeDownArrow`
- `UIPressTypeLeftArrow`
- `UIPressTypeRightArrow`

Press Types

An illustrated guide

- `UIPressTypeUpArrow`
- `UIPressTypeDownArrow`
- `UIPressTypeLeftArrow`
- `UIPressTypeRightArrow`

```
// Tap on Play-Pause button.
let playPauseTap = UITapGestureRecognizer(target: self, action:
   #selector(MyClass.handlePlayPause(_:)))
playPauseTap.allowedPressTypes = [ UIPressType.playPause.rawValue ]

// Long-press on Select button.
let selectLongPress = UILongPressGestureRecognizer(target: self, action:
   #selector(MyClass.handleSelectLongPress(_:)))
selectLongPress.allowedPressTypes = [ UIPressType.select.rawValue ]
```

```
// Double-tap on Select button.
let selectDoubleTap = UITapGestureRecognizer(target: self, action:
   #selector(MyClass.handleSelectDoubleTap(_:)))
selectDoubleTap.allowedPressTypes = [ UIPressType.select.rawValue ]
selectDoubleTap.numberOfTapsRequired = 2
```

The Menu Button

The Menu Button

Users must be able to exit your app using the Menu button

The Menu Button

Users must be able to exit your app using the Menu button

- This is a requirement that App Review specifically looks for

The Menu Button

Users must be able to exit your app using the Menu button

- This is a requirement that App Review specifically looks for
- The event must reach UIApplication pressesEnded:withEvent: in order for the app to exit

Custom Menu Button Handling

Custom Menu Button Handling

Remove the gesture from its view

Custom Menu Button Handling

Remove the gesture from its view

Disable the gesture

Custom Menu Button Handling

Remove the gesture from its view

Disable the gesture

Implement the gesture delegate method `gestureRecognizerShouldBegin`

Custom Menu Button Handling

Remove the gesture from its view

Disable the gesture

Implement the gesture delegate method `gestureRecognizerShouldBegin`

In `pressesEnded` call super if you're not going to handle the event

Custom Menu Button Handling

Remove the gesture from its view

Disable the gesture

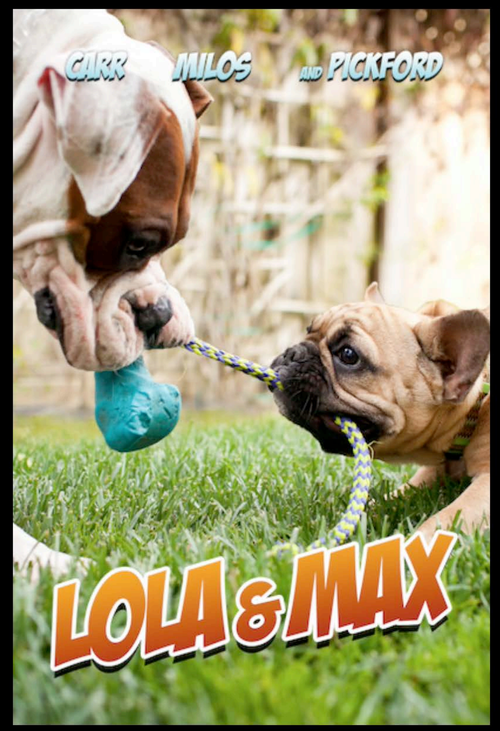
Implement the gesture delegate method `gestureRecognizerShouldBegin`

In `pressesEnded` call super if you're not going to handle the event

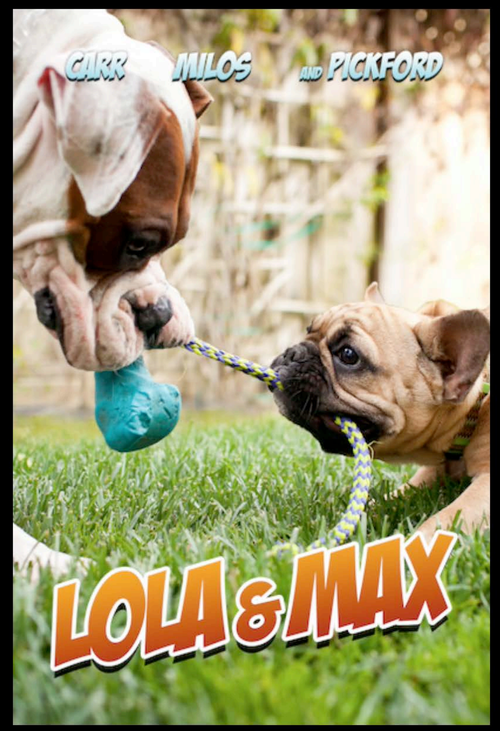
GCEventViewController and `controllerUserInteractionEnabled`

Layered Images

Layered Images

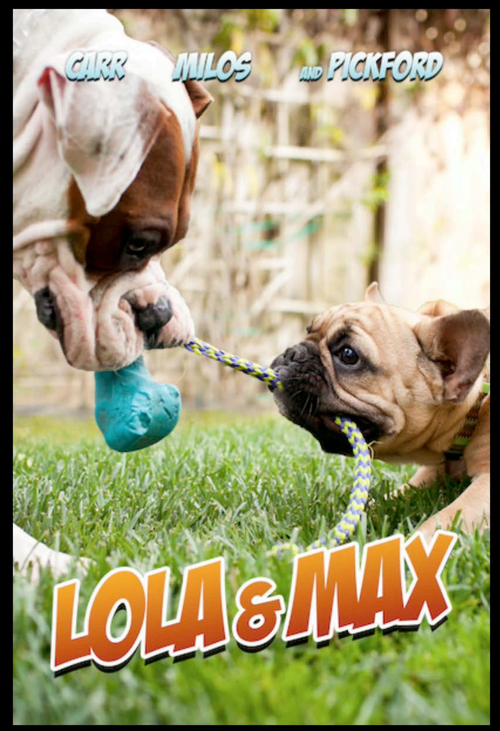


Layered Images



Layered Images

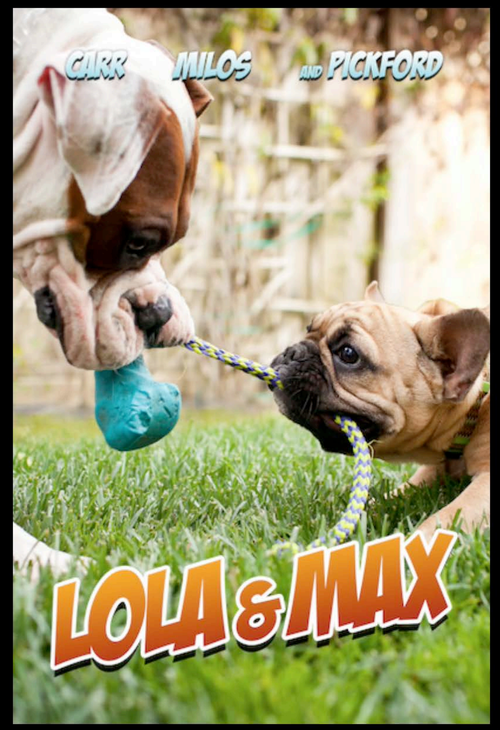
Specific to tvOS



Layered Images

Specific to tvOS

Can have up to five layers

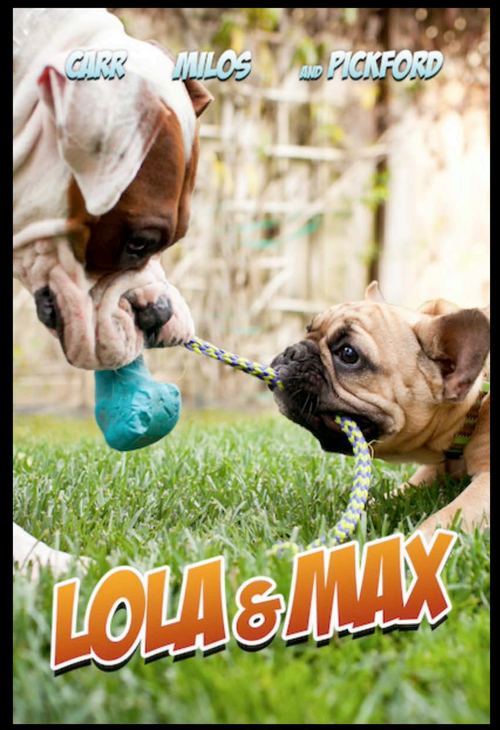


Layered Images

Specific to tvOS

Can have up to five layers

Required for app icons



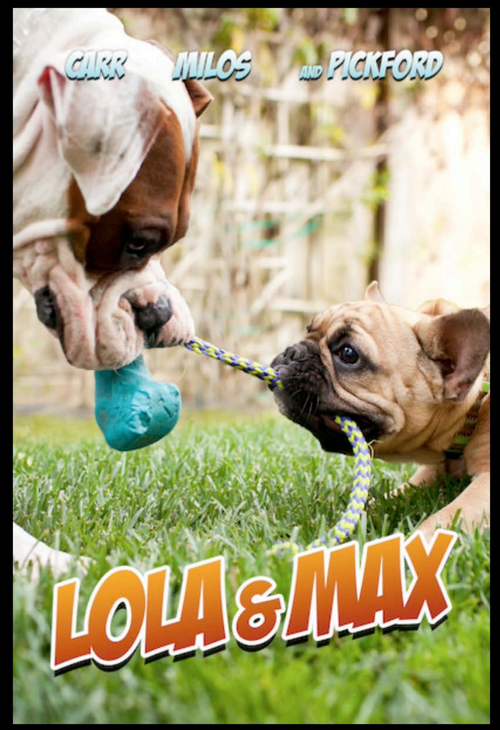
Layered Images

Specific to tvOS

Can have up to five layers

Required for app icons

Interactive



Layered Images

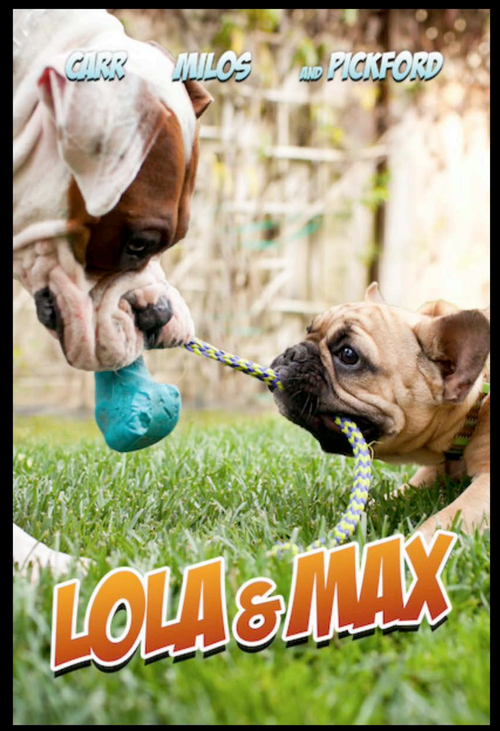
Specific to tvOS

Can have up to five layers

Required for app icons

Interactive

Animated



Interactivity with Layered Images

Interactivity with Layered Images

Images are usually not interactive controls themselves, but components of larger controls

Interactivity with Layered Images

Images are usually not interactive controls themselves, but components of larger controls

The image view itself doesn't need to be focused to get the floating appearance

Interactivity with Layered Images

Images are usually not interactive controls themselves, but components of larger controls

The image view itself doesn't need to be focused to get the floating appearance

```
imageView.adjustsImageWhenAncestorFocused = true
```

Interactivity with Layered Images

Images are usually not interactive controls themselves, but components of larger controls

The image view itself doesn't need to be focused to get the floating appearance

```
imageView.adjustsImageWhenAncestorFocused = true
```

The image view can show a "pressed-in" state

Interactivity with Layered Images

Images are usually not interactive controls themselves, but components of larger controls

The image view itself doesn't need to be focused to get the floating appearance

```
imageView.adjustsImageWhenAncestorFocused = true
```

The image view can show a "pressed-in" state

```
imageView.isHighlighted = pressed ? true : false
```

Interactivity with Layered Images

Images are usually not interactive controls themselves, but components of larger controls

The image view itself doesn't need to be focused to get the floating appearance

```
imageView.adjustsImageWhenAncestorFocused = true
```

The image view can show a "pressed-in" state

```
imageView.isHighlighted = pressed ? true : false
```

Two common use cases that are covered for you:

Interactivity with Layered Images

Images are usually not interactive controls themselves, but components of larger controls

The image view itself doesn't need to be focused to get the floating appearance

```
imageView.adjustsImageWhenAncestorFocused = true
```

The image view can show a "pressed-in" state

```
imageView.isHighlighted = pressed ? true : false
```

Two common use cases that are covered for you:

- Images inside UICollectionViewCells

Interactivity with Layered Images

Images are usually not interactive controls themselves, but components of larger controls

The image view itself doesn't need to be focused to get the floating appearance

```
imageView.adjustsImageWhenAncestorFocused = true
```

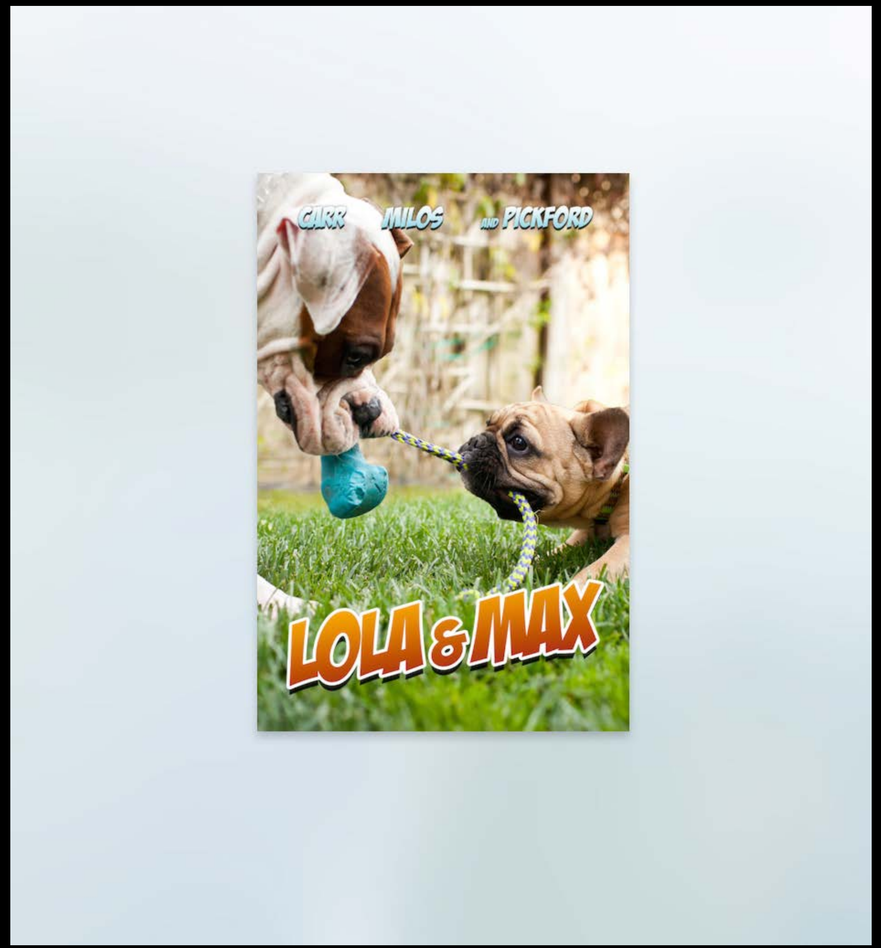
The image view can show a "pressed-in" state

```
imageView.isHighlighted = pressed ? true : false
```

Two common use cases that are covered for you:

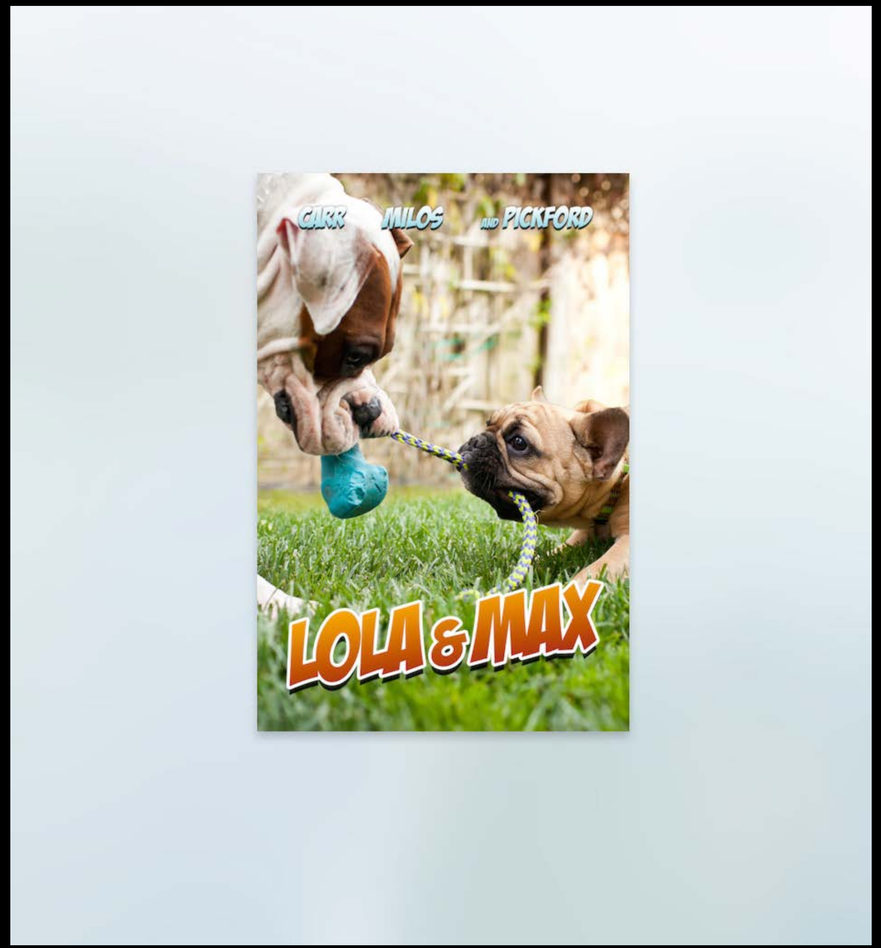
- Images inside UICollectionViewCells
- The imageView within a custom UIButton

Animations with Layered Images



Animations with Layered Images

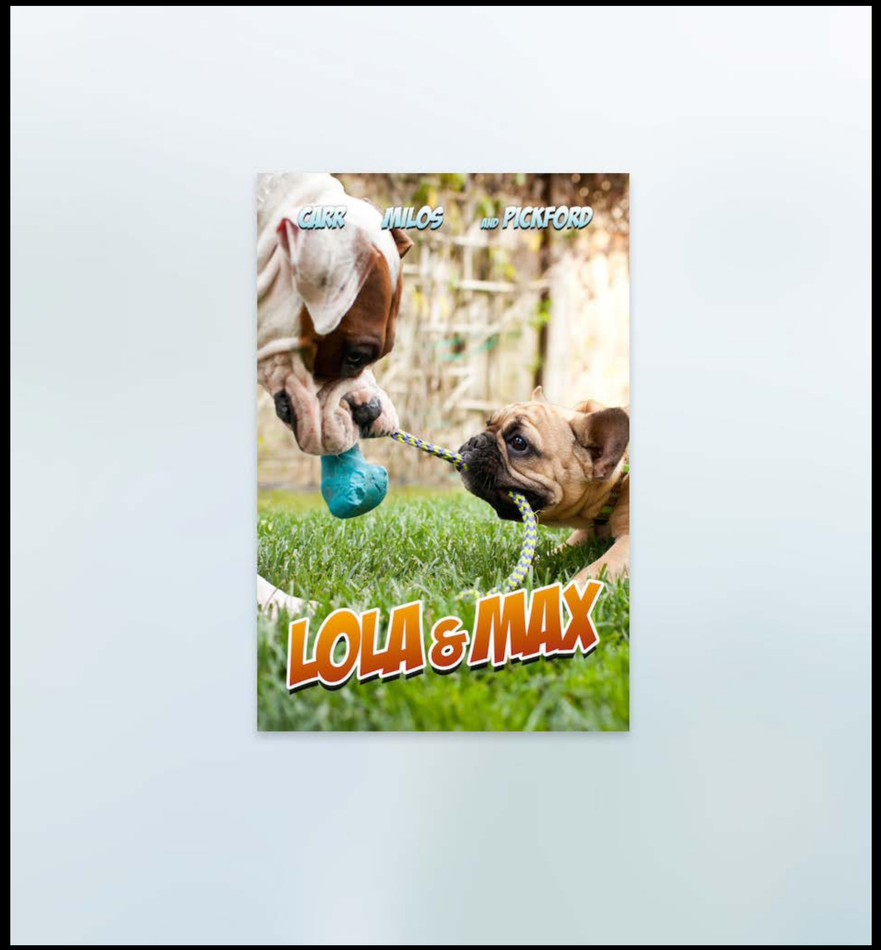
When an image enlarges, you can rearrange nearby views using Auto Layout



Animations with Layered Images

When an image enlarges, you can rearrange nearby views using Auto Layout

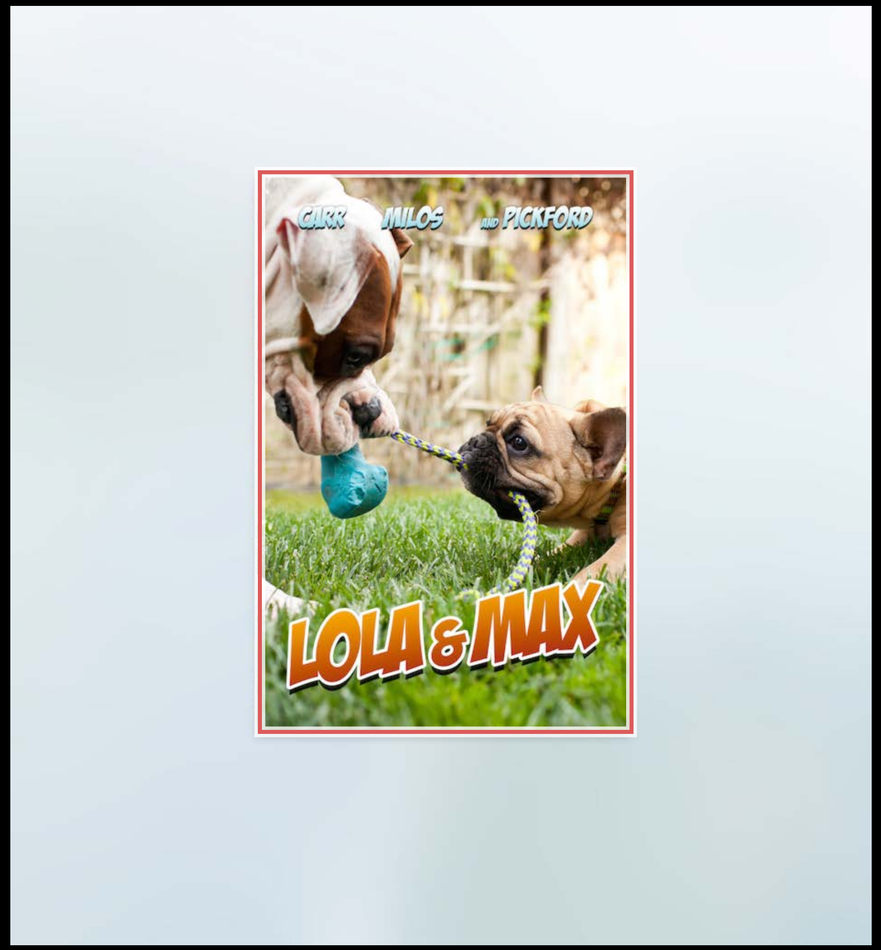
- Use the layout guide `focusedFrameGuide` to create a constraint



Animations with Layered Images

When an image enlarges, you can rearrange nearby views using Auto Layout

- Use the layout guide `focusedFrameGuide` to create a constraint



Animations with Layered Images

When an image enlarges, you can rearrange nearby views using Auto Layout

- Use the layout guide `focusedFrameGuide` to create a constraint

Animations with Layered Images

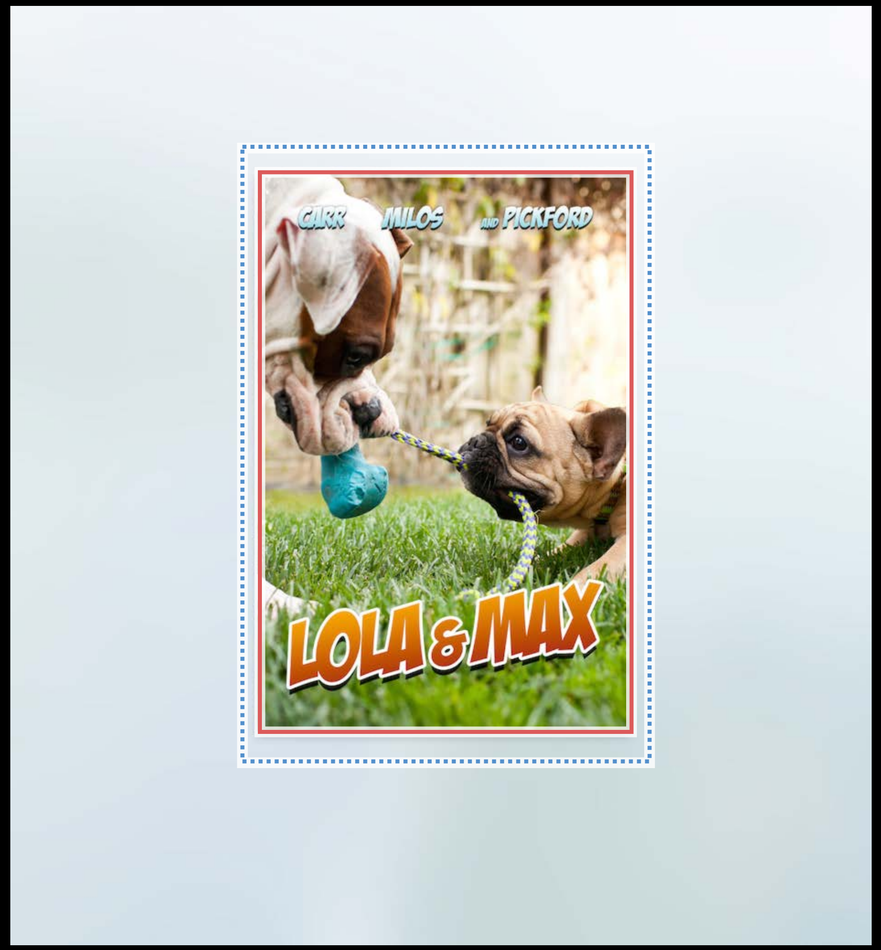
When an image enlarges, you can rearrange nearby views using Auto Layout

- Use the layout guide `focusedFrameGuide` to create a constraint

Animations with Layered Images

When an image enlarges, you can rearrange nearby views using Auto Layout

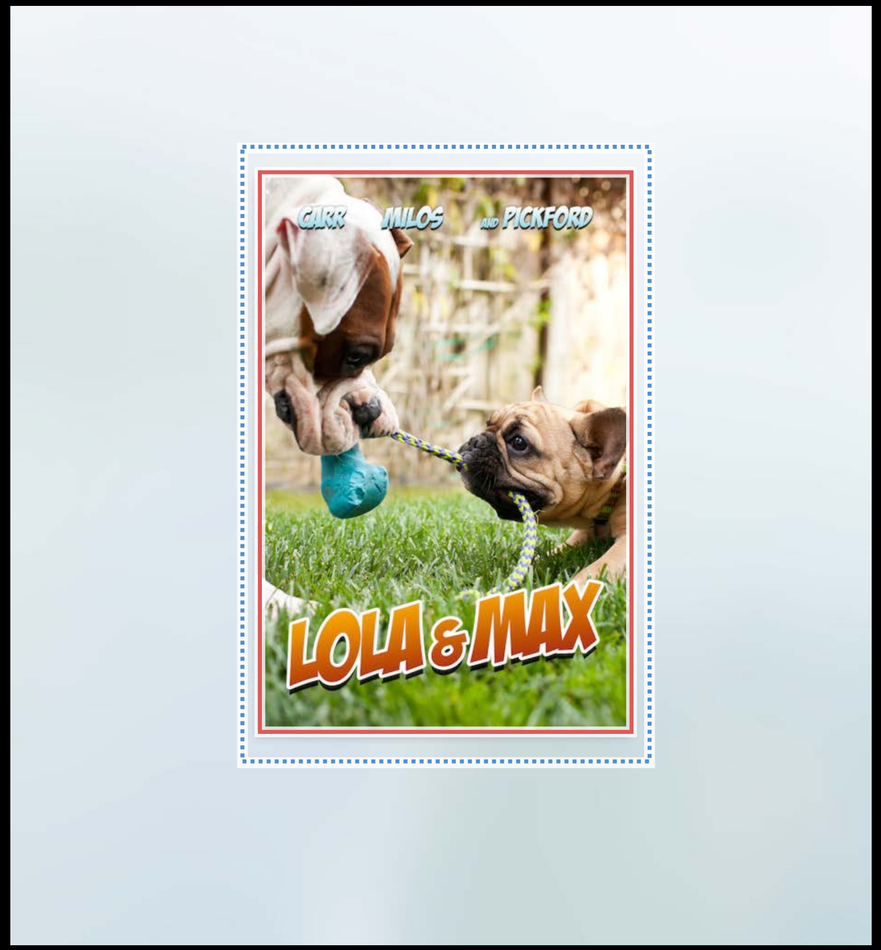
- Use the layout guide `focusedFrameGuide` to create a constraint



Animations with Layered Images

When an image enlarges, you can rearrange nearby views using Auto Layout

- Use the layout guide `focusedFrameGuide` to create a constraint
- Use focus context methods to coordinate animations correctly



Demo

Interactivity and animation for layered images

Randy Becker

Scrolling

Scroll Offset Selection

Scroll Offset Selection

Usually not doing direct manipulation

Scroll Offset Selection

Usually not doing direct manipulation

- The user manipulates focus, and the focus engine chooses a scroll offset

Scroll Offset Selection

Usually not doing direct manipulation

- The user manipulates focus, and the focus engine chooses a scroll offset
- UIScrollViewDelegate can override the automatic scroll offset

Scroll Offset Selection

Usually not doing direct manipulation

- The user manipulates focus, and the focus engine chooses a scroll offset
- UIScrollViewDelegate can override the automatic scroll offset

```
func scrollViewWillEndDragging(_ scrollView: UIScrollView, withVelocity velocity: CGPoint,
   targetContentOffset: UnsafeMutablePointer<CGPoint>) {
      let myOffset = CGPoint(x: 42.0, y: 0) // Do your own calculations here.
      targetContentOffset.initialize(with: myOffset)
}
```

Direct Manipulation

Direct Manipulation

It is possible to do direct manipulation if the situation calls for it

Direct Manipulation

It is possible to do direct manipulation if the situation calls for it

- Reconfigure pan gesture recognizer on scroll view to recognize indirect touches

Direct Manipulation

It is possible to do direct manipulation if the situation calls for it

- Reconfigure pan gesture recognizer on scroll view to recognize indirect touches
- Enable the directional press gesture

Direct Manipulation

It is possible to do direct manipulation if the situation calls for it

- Reconfigure pan gesture recognizer on scroll view to recognize indirect touches
- Enable the directional press gesture

```
scrollView.panGestureRecognizer.allowedTouchTypes = [ UITouchType.indirect.rawValue ]
scrollView.directionalPressGestureRecognizer.isEnabled = true
```

Demo

Focus and direction manipulation

Kevin Hiscott

Text Input

The System Keyboard

The System Keyboard

The system standard keyboard is the only way to get certain text input features

The System Keyboard

The system standard keyboard is the only way to get certain text input features

- Dictation

The System Keyboard

The system standard keyboard is the only way to get certain text input features

- Dictation
- Bluetooth keyboards

The System Keyboard

The system standard keyboard is the only way to get certain text input features

- Dictation
- Bluetooth keyboards
- Apple TV Remote app

The System Keyboard

The system standard keyboard is the only way to get certain text input features

- Dictation
- Bluetooth keyboards
- Apple TV Remote app
- Localization

The System Keyboard

The system standard keyboard is the only way to get certain text input features

- Dictation
- Bluetooth keyboards
- Apple TV Remote app
- Localization
- Automatic grid or linear layout based on input device

The System Keyboard

The system standard keyboard is the only way to get certain text input features

- Dictation
- Bluetooth keyboards
- Apple TV Remote app
- Localization
- Automatic grid or linear layout based on input device

Don’t try to create your own keyboard, or your users will miss out on these features

Keyboard Appearance

Keyboard Appearance

Adding custom UI to the keyboard is possible

Keyboard Appearance

Adding custom UI to the keyboard is possible

- Use inputAccessoryView or inputAccessoryViewController

Keyboard Appearance

Adding custom UI to the keyboard is possible

- Use inputAccessoryView or inputAccessoryViewController

UITextField appearance no longer dictated by keyboardAppearance

Accessory View

Everything in the red box is part of the text field's inputAccessoryView property.

Hold to dictate

SPACE a b c d e f g h i j k l m n o p q r s t u v w x y z

1 2 3 4 5 6 7 8 9 0 . - _ ! @ # $ % ^ & *

ABC abc #+-

Done

Search Controller

Search Controller

UISearchController

Search Controller

UISearchController

Keyboard and search results visible at the same time

Search Controller

UISearchController

Keyboard and search results visible at the same time

Automatically adapts to linear and grid keyboards

Search Controller

UISearchController

Keyboard and search results visible at the same time

Automatically adapts to linear and grid keyboards

Can be embedded within container view controllers

Search Controller

UISearchController

Keyboard and search results visible at the same time

Automatically adapts to linear and grid keyboards

Can be embedded within container view controllers

- UISearchContainerViewController

Search Controller

UISearchController

Keyboard and search results visible at the same time

Automatically adapts to linear and grid keyboards

Can be embedded within container view controllers

- UISearchContainerViewController

Custom view controller for search results

```swift
// Embedding UISearchController
class MySearchController: UIViewController, UISearchResultsUpdating
{
    var searchController: UISearchController!
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        if (searchController == nil) {
            let results = UIViewController()
            searchController = UISearchController(searchResultsController: results)
            searchController.searchResultsUpdater = self
            let container = UISearchContainerViewController
               (searchController: searchController)
            self.addChildViewController(container)
            self.view.addSubview(container.view)
            container.didMove(toParentViewController: self)
        }
    }
}
```

```swift
// Embedding UISearchController
class MySearchController: UIViewController, UISearchResultsUpdating
{
    var searchController: UISearchController!
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        if (searchController == nil) {
            let results = UIViewController()
            searchController = UISearchController(searchResultsController: results)
            searchController.searchResultsUpdater = self
            let container = UISearchContainerViewController
               (searchController: searchController)
            self.addChildViewController(container)
            self.view.addSubview(container.view)
            container.didMove(toParentViewController: self)
        }
    }
}
```

```swift
// Embedding UISearchController
class MySearchController: UIViewController, UISearchResultsUpdating
{
    var searchController: UISearchController!
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        if (searchController == nil) {
            let results = UIViewController()
            searchController = UISearchController(searchResultsController: results)
            searchController.searchResultsUpdater = self
            let container = UISearchContainerViewController
               (searchController: searchController)
            self.addChildViewController(container)
            self.view.addSubview(container.view)
            container.didMove(toParentViewController: self)
        }
    }
}
```

```swift
// Embedding UISearchController
class MySearchController: UIViewController, UISearchResultsUpdating
{
    var searchController: UISearchController!
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        if (searchController == nil) {
            let results = UIViewController()
            searchController = UISearchController(searchResultsController: results)
            searchController.searchResultsUpdater = self
            let container = UISearchContainerViewController
               (searchController: searchController)
            self.addChildViewController(container)
            self.view.addSubview(container.view)
            container.didMove(toParentViewController: self)
        }
    }
}
```

```swift
// Embedding UISearchController
class MySearchController: UIViewController, UISearchResultsUpdating
{
    var searchController: UISearchController!
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        if (searchController == nil) {
            let results = UIViewController()
            searchController = UISearchController(searchResultsController: results)
            searchController.searchResultsUpdater = self
            let container = UISearchContainerViewController
               (searchController: searchController)
            self.addChildViewController(container)
            self.view.addSubview(container.view)
            container.didMove(toParentViewController: self)
        }
    }
}
```

```swift
// Embedding UISearchController
class MySearchController: UIViewController, UISearchResultsUpdating
{
    var searchController: UISearchController!
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        if (searchController == nil) {
            let results = UIViewController()
            searchController = UISearchController(searchResultsController: results)
            searchController.searchResultsUpdater = self
            let container = UISearchContainerViewController
               (searchController: searchController)
            self.addChildViewController(container)
            self.view.addSubview(container.view)
            container.didMove(toParentViewController: self)
        }
    }
}
```

Summary

Summary

Be careful with custom Menu button handling

Summary

Be careful with custom Menu button handling

Use layout guides and coordinated animations for great layered images

Summary

Be careful with custom Menu button handling

Use layout guides and coordinated animations for great layered images

If you need direct manipulation, use the existing UIScrollView gestures

Summary

Be careful with custom Menu button handling

Use layout guides and coordinated animations for great layered images

If you need direct manipulation, use the existing UIScrollView gestures

Use the system standard keyboard for text input

More Information

developer.apple.com/wwdc16/210

Related Sessions

What's New in tvOS	Presidio	Tuesday 3:00PM
Focus Interaction on tvOS	Mission	Wednesday 4:00PM

Labs

tvOS Lab	Frameworks Lab D	Wednesday 2:00PM
tvOS Lab	Frameworks Lab D	Thursday 9:00AM

WWDC16