



ADVANCED tvOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 9

APPLE TV

APPLE TV

SUBTITLE

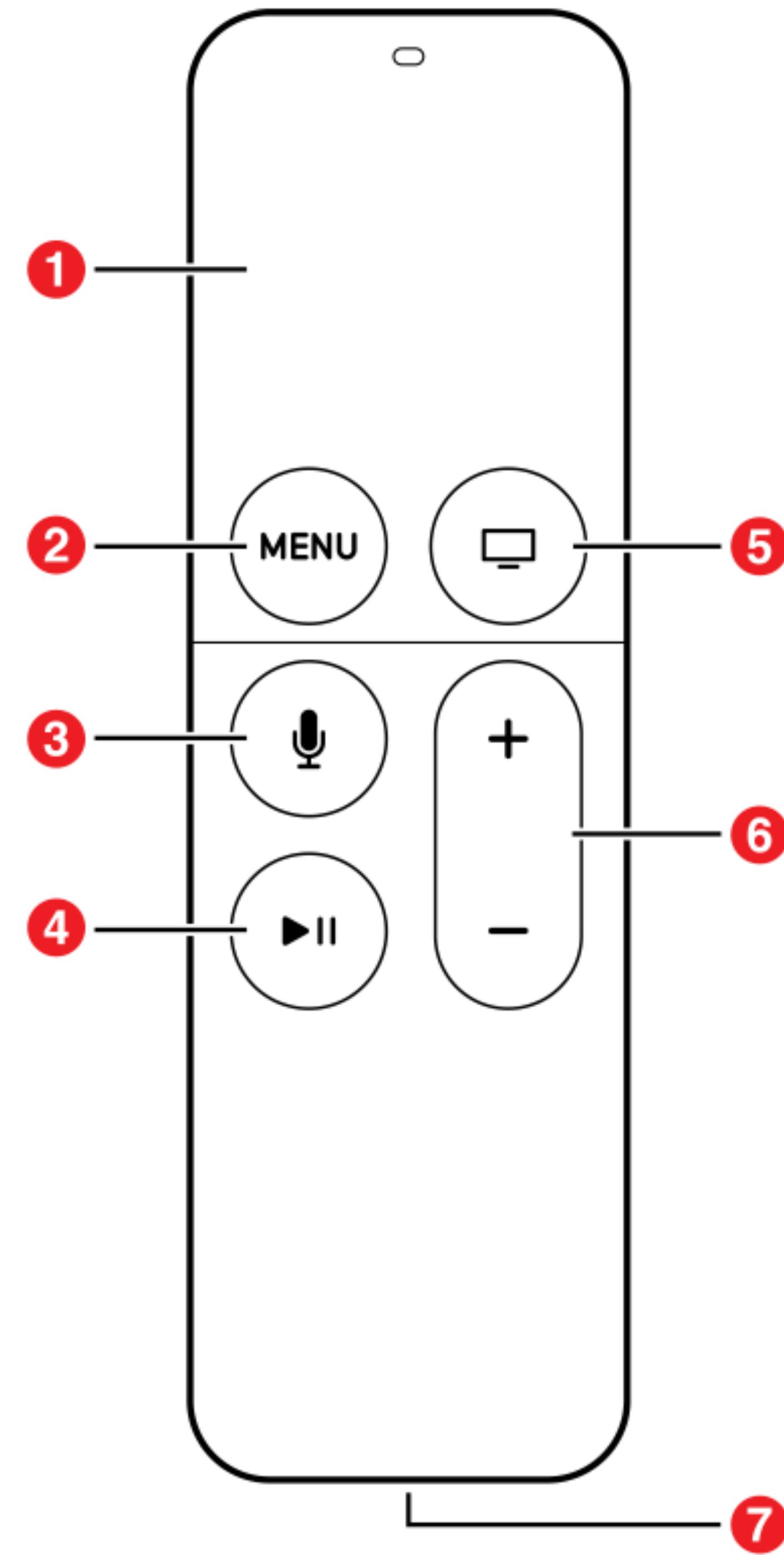
- Apple TV has the following hardware specifications
 - 64-bit A8, A10X, A12X Fusion processor
 - 32 GB, 64 GB of storage
 - 2/4 GB of RAM
 - 10/100 Mbps Ethernet
 - WiFi 802.11a/b/g/n/ac
 - 1080p/4K resolution
 - HDMI
 - New Siri Remote / Apple TV Remote



APPLE TV

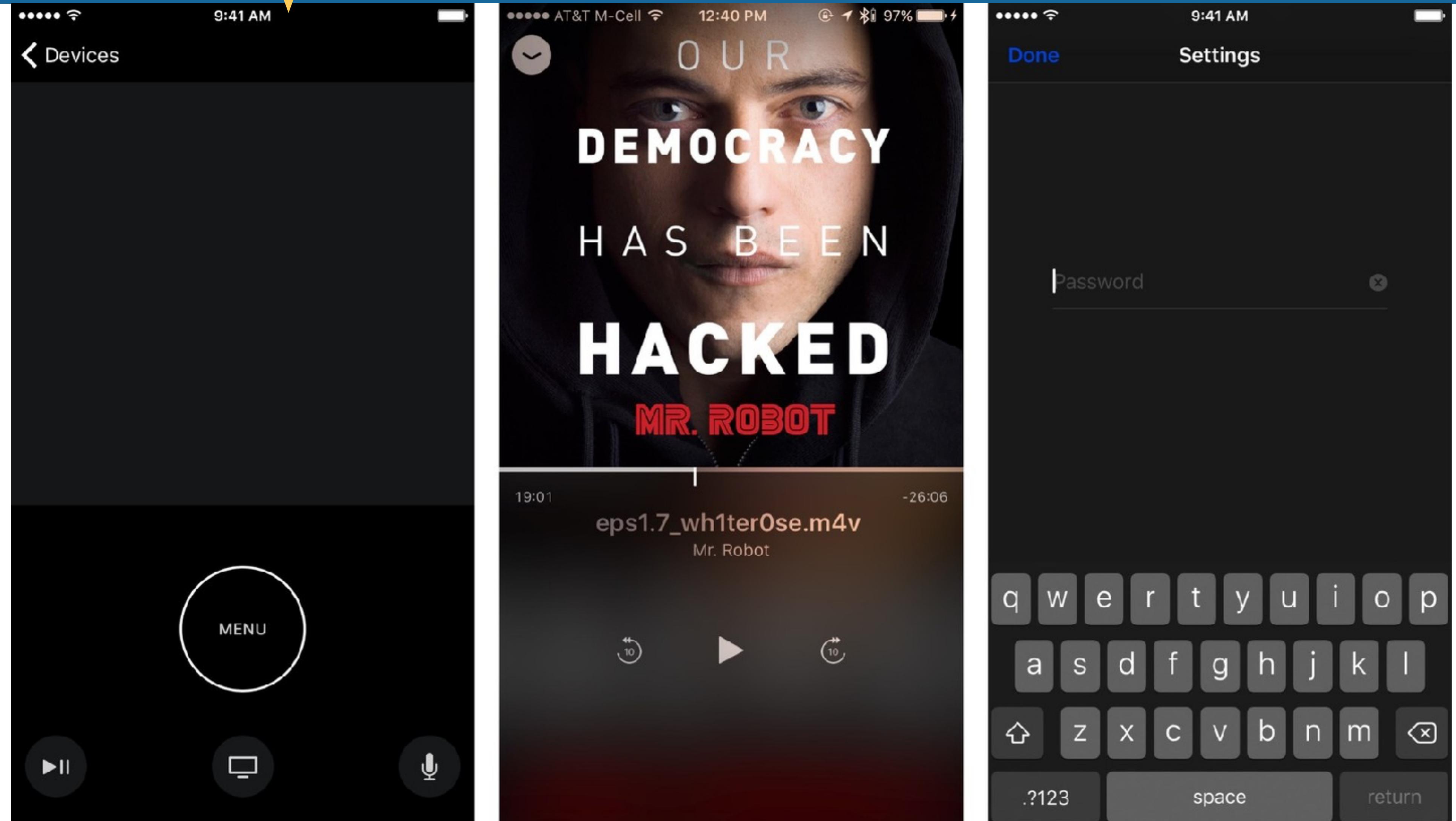
SUBTITLE

- Two remotes
 - Siri Remote
 - Apple TV remote
 - iPhone App



APPLE TV

Everything and more
but less

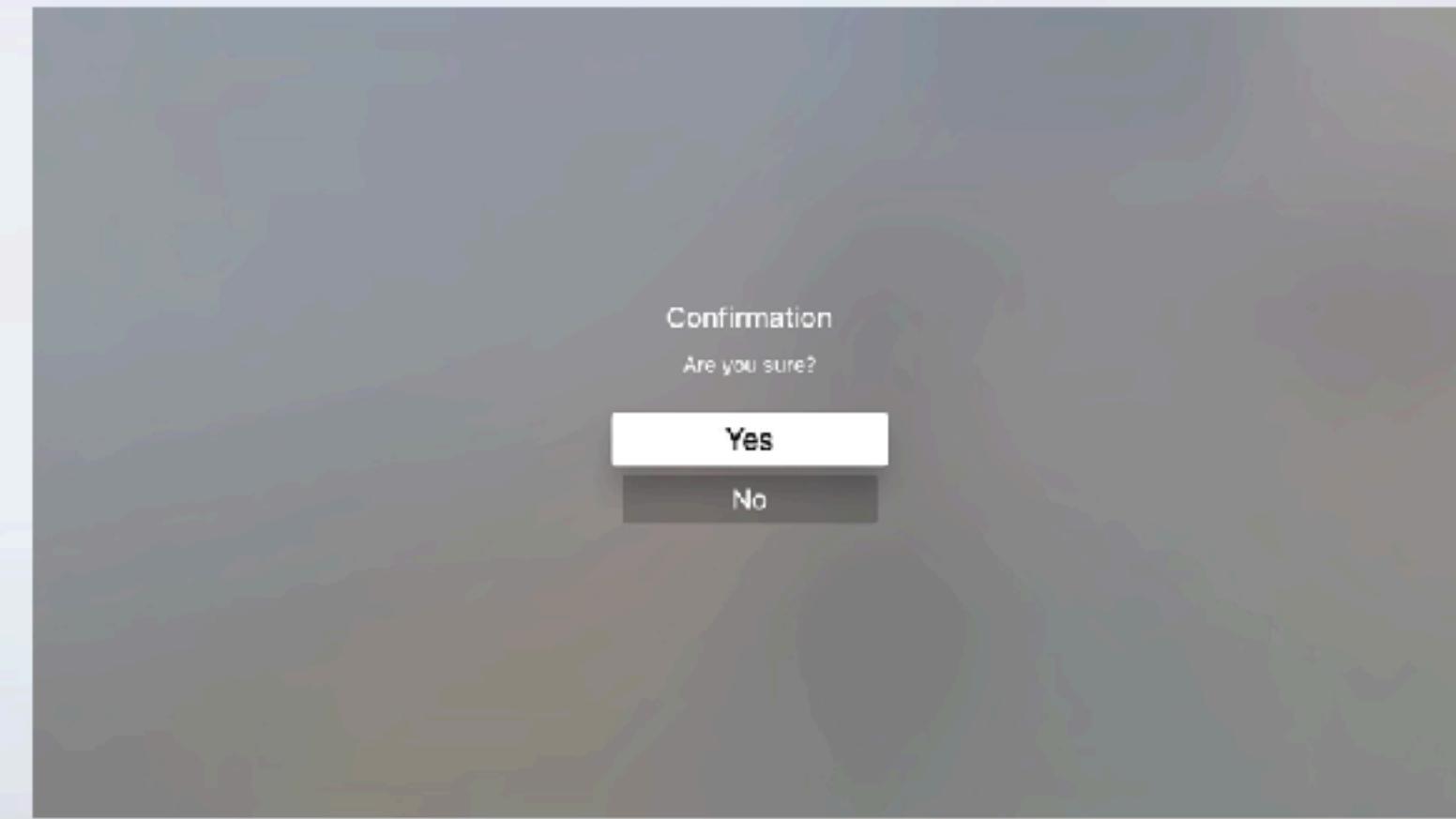


APPLE TV

- Two style of apps for Apple TV
 - UIKit
 - TVML

Apple TV 1080p - Apple TV 1080p / tvOS 9.0 (13T5365h)

Template Examples



An alert template displays a message on screen and asks the user to perform some action, such as confirm a purchase or destructive action.

Alert

- [Alert with Description](#)
- [Alert with Shelf](#)
- [Catalog](#)
- [Compilation](#)
- [Descriptive Alert](#)
- [Form](#)
- [List Item Examples](#)
- [List with Banner Image](#)
- [Loading](#)
- [Main](#)

APPLE TV

SUBTITLE

- TVML is a custom markup language used to design apps whose main purpose is to stream media
- Templates define the interface
- Javascript to interface with server providing content

Tycho

GENRE

Electronica

COUNTRY

United States

Scott Hansen (born 1976/1977), professional Tycho, is an American ambient music artist who is known as ISO50 for his photographic works. Tycho is currently signed to, and has Ghostly International, but has also released records on Gammaphone Records. His son



Play

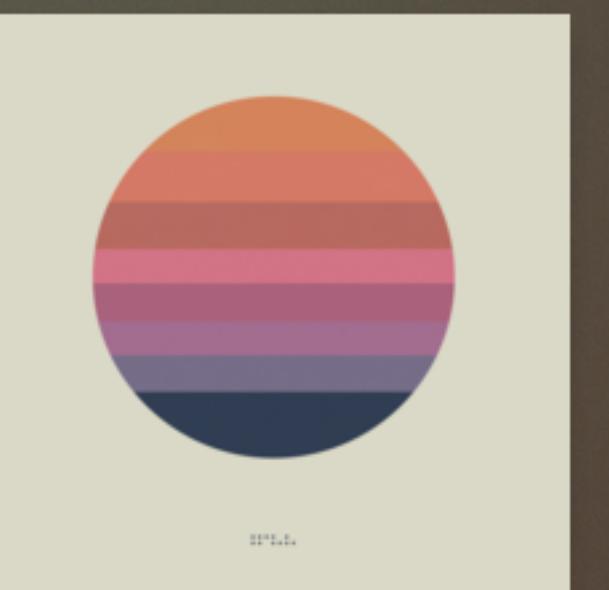


Shuffle

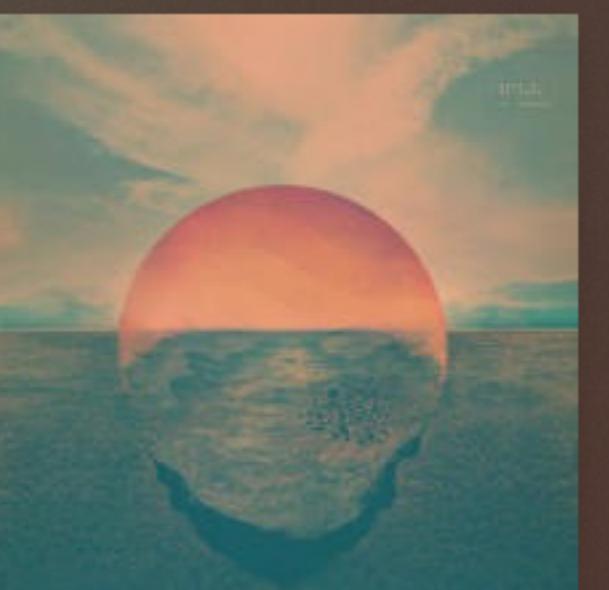


Play popular tracks

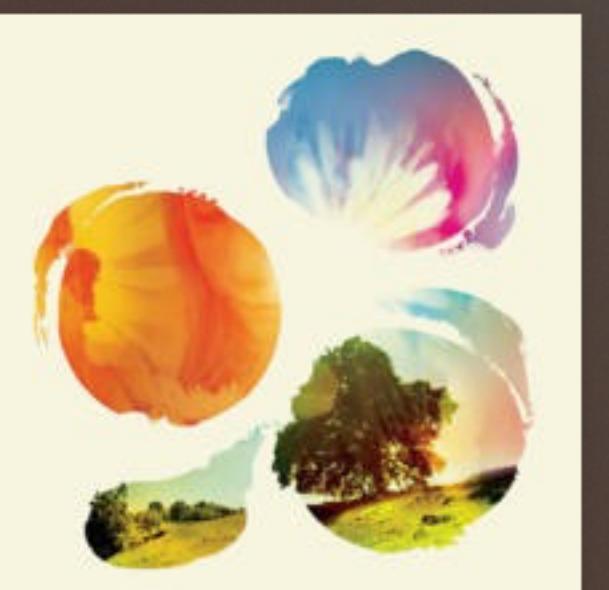
Albums



Awake
2014



Dive
2011



Past is Prologue
2006



Sunnyside
2005

APPLE TV

Tycho

GENRE

Electronica

COUNTRY

United States

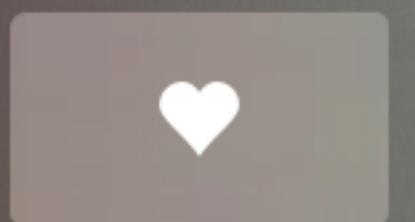
Scott Hansen (born 1976/1977), professionally known as Tycho, is an American ambient music artist and producer, who is known as ISO50 for his photographic and design works. Tycho is currently signed to, and has released under, Ghostly International, but has also released music on Merck Records and Gammaphone Records. His song, Dicta... [MORE](#)



Play



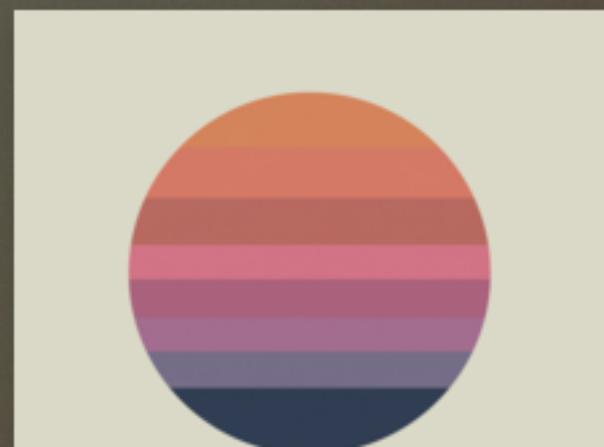
Shuffle



Play popular tracks

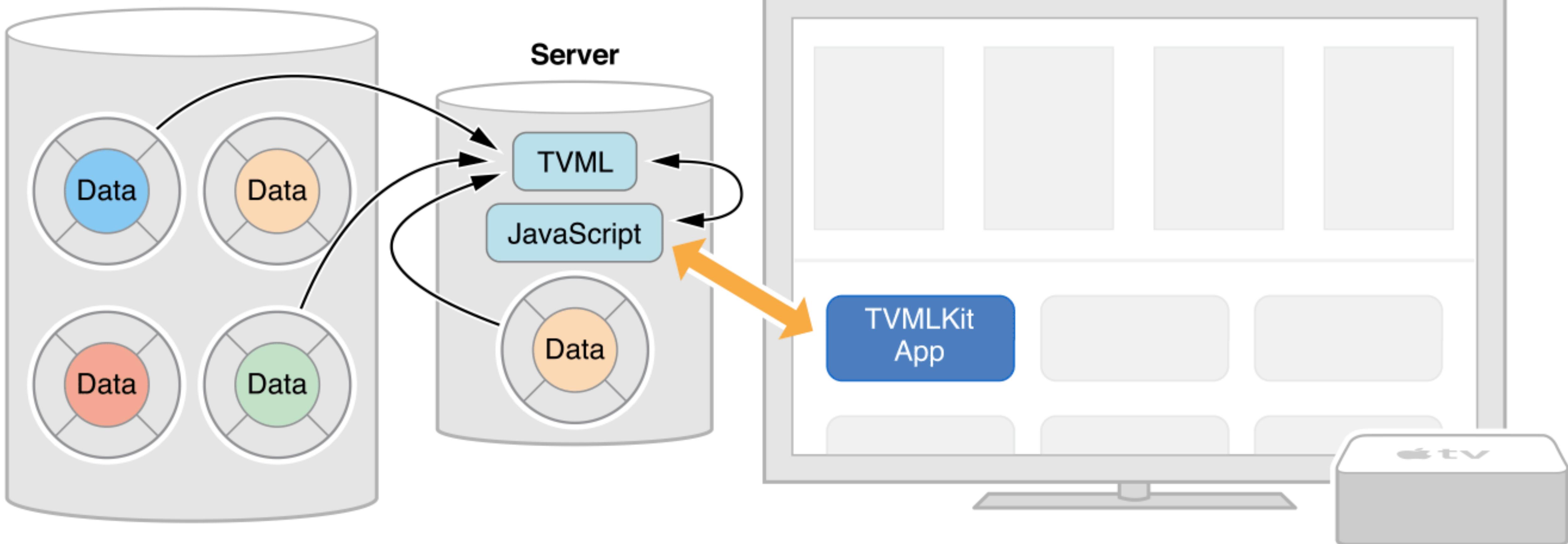


Albums



APPLE TV

**Additional Asset Server
(optional)**



APPLE TV

SUBTITLE

- Top shelf provides preferred placement in top row
- Static or dynamic content



APPLE TV

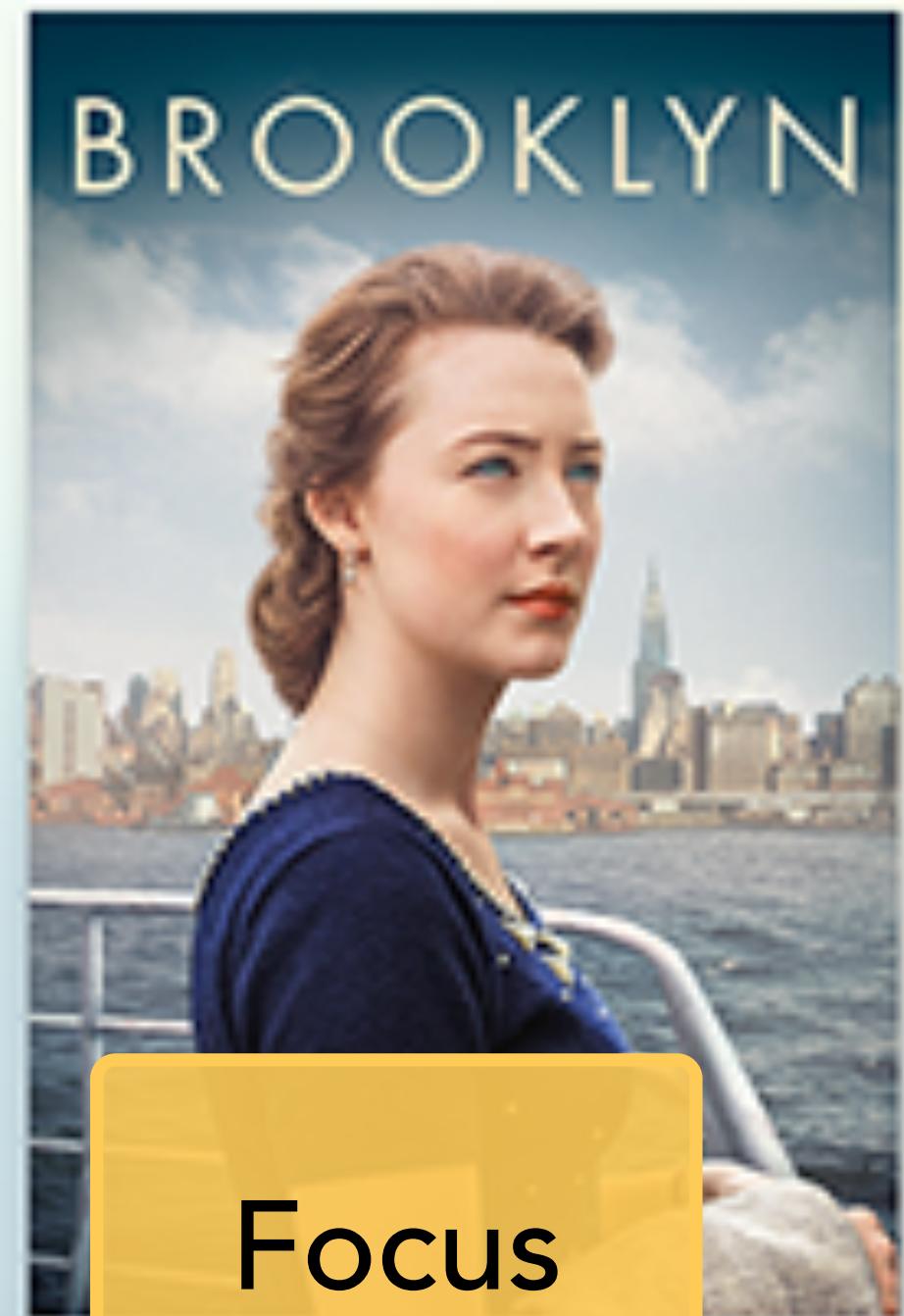


APPLE TV

SUBTITLE

- Focus
 - Navigation
 - UI element is **highlighted** by a user (not selected)

Top Movies



Focus

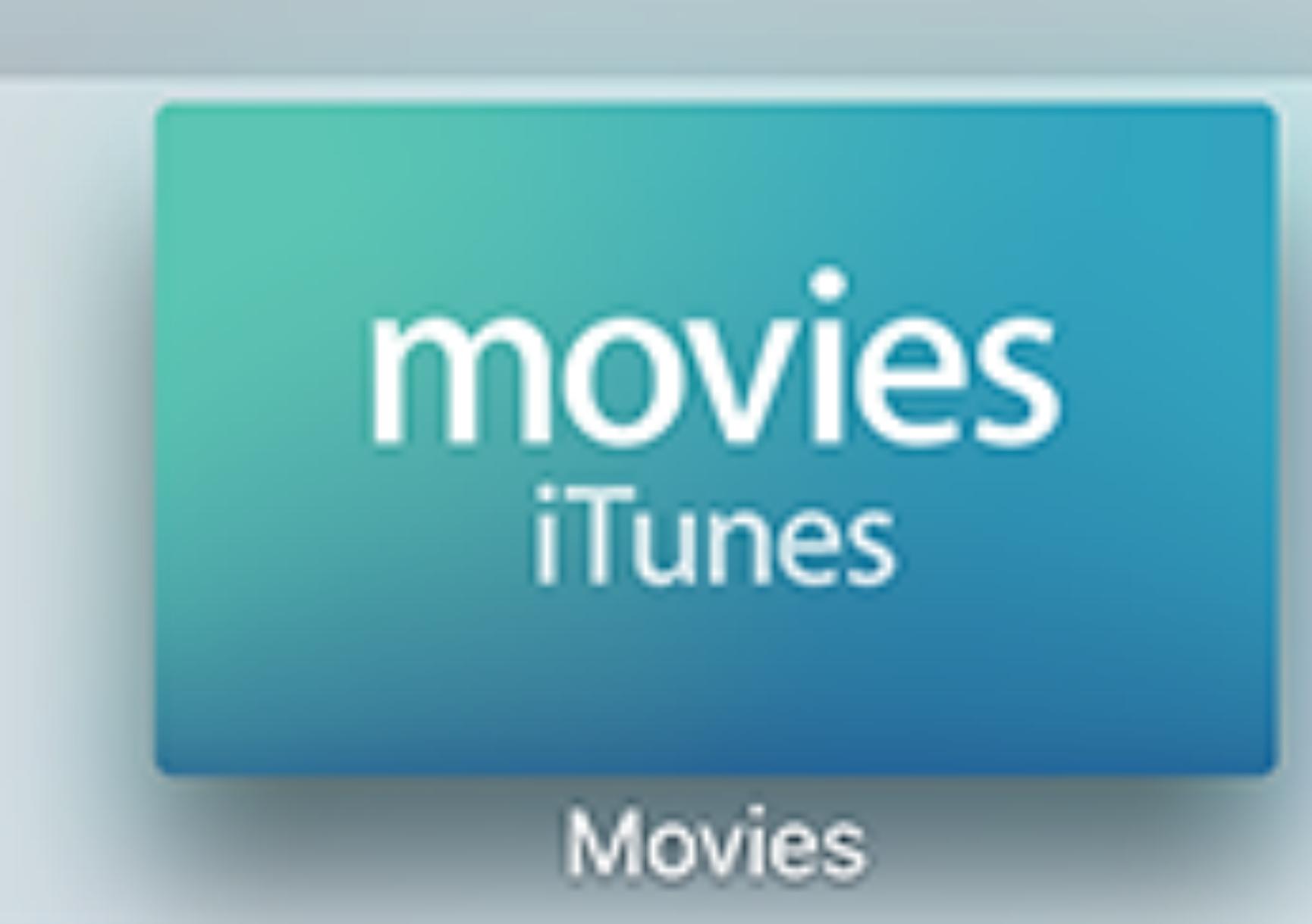


Movies

APPLE TV

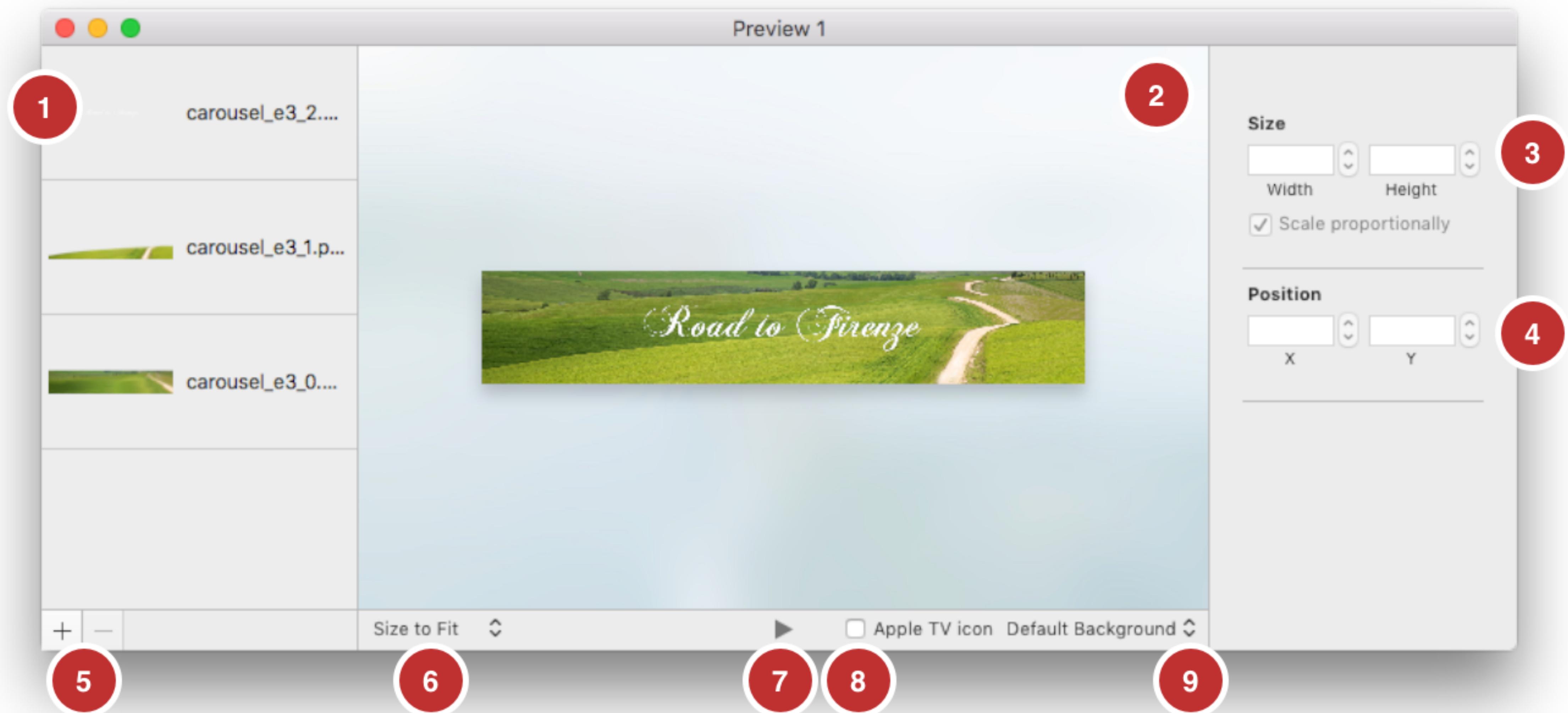
SUBTITLE

- Images can be layered for a single image view
 - UIImageView is customized for tvOS
- "Eye candy" for effect



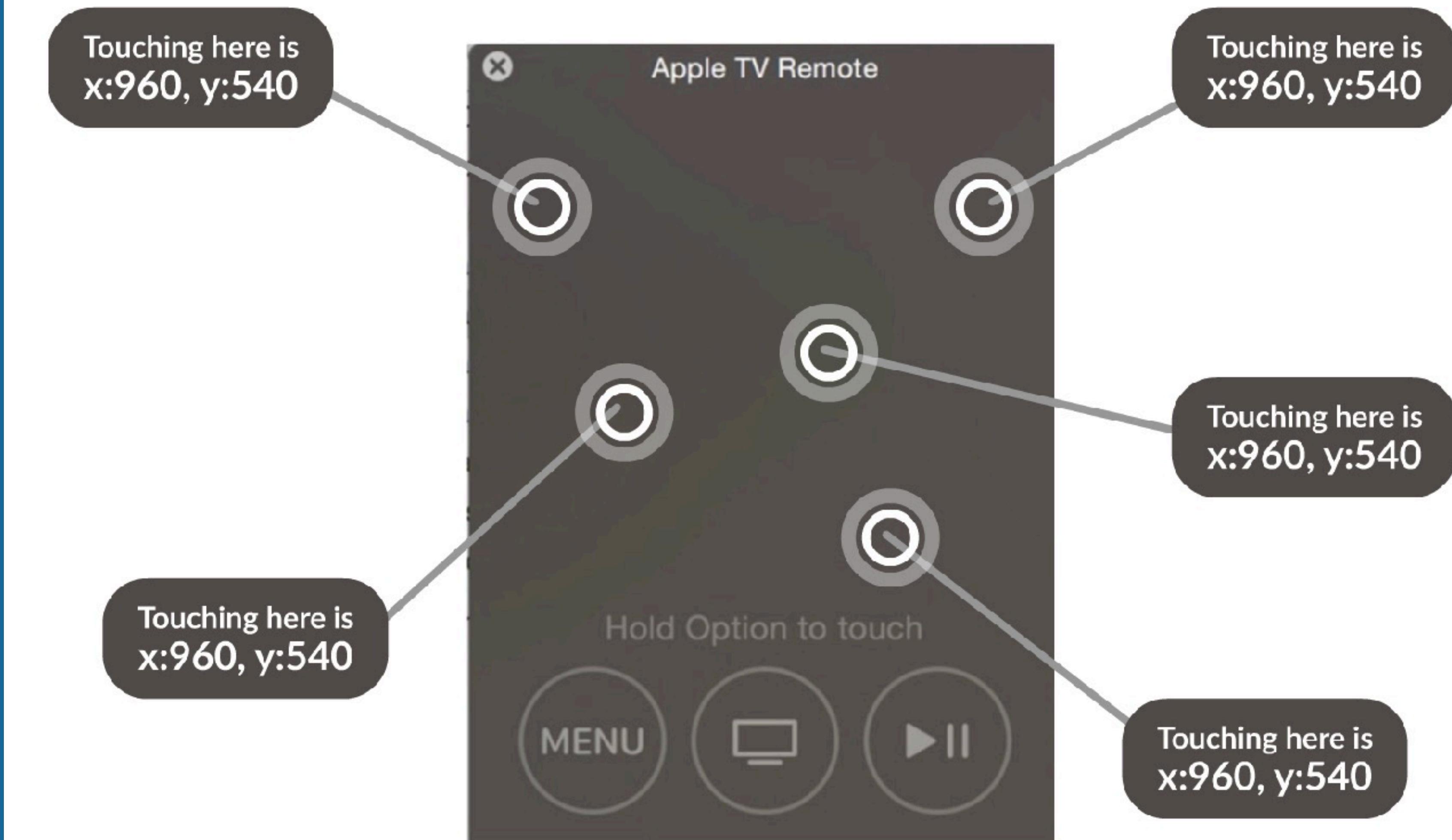
APPLE TV

Xcode tools



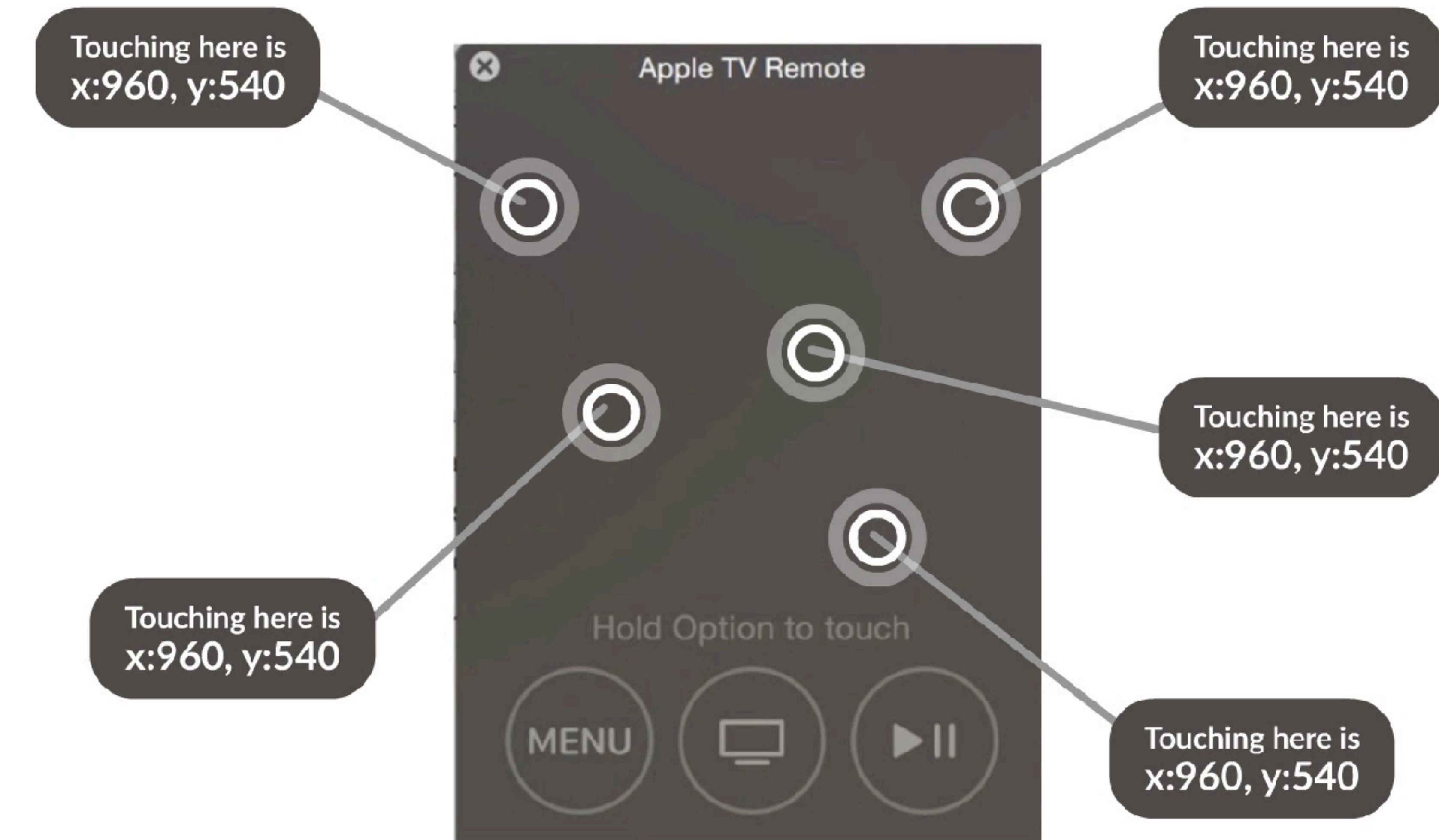
APPLE TV

- UIKit implementation is familiar
- Interaction is very different



APPLE TV

- Users can only interact in specific places accessible by remote
- Multi-users
 - Remote
 - iPhone
 - GamePad



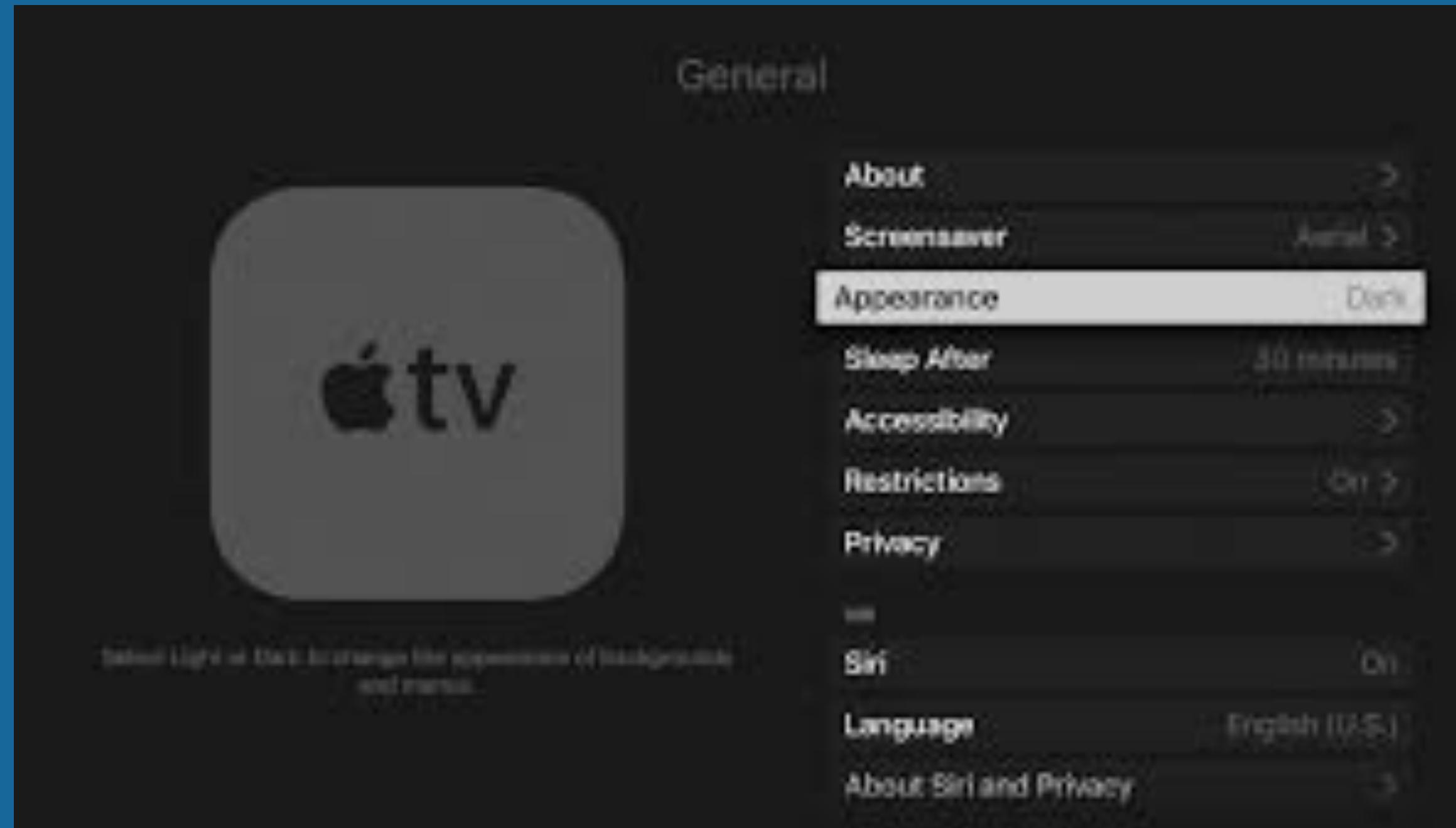
APPLE TV

- Local storage is extremely limited (500KB)
 - Highly unstable (tvOS will clear cache aggressively)
- Apps can only be 4GB
 - Majority of storage is for apps and streaming content

APPLE TV

- Strategies
 - Download and cache
 - On-demand resources (Apple provided or self-hosted)
 - iCloud (CloudKit)

APPLE TV

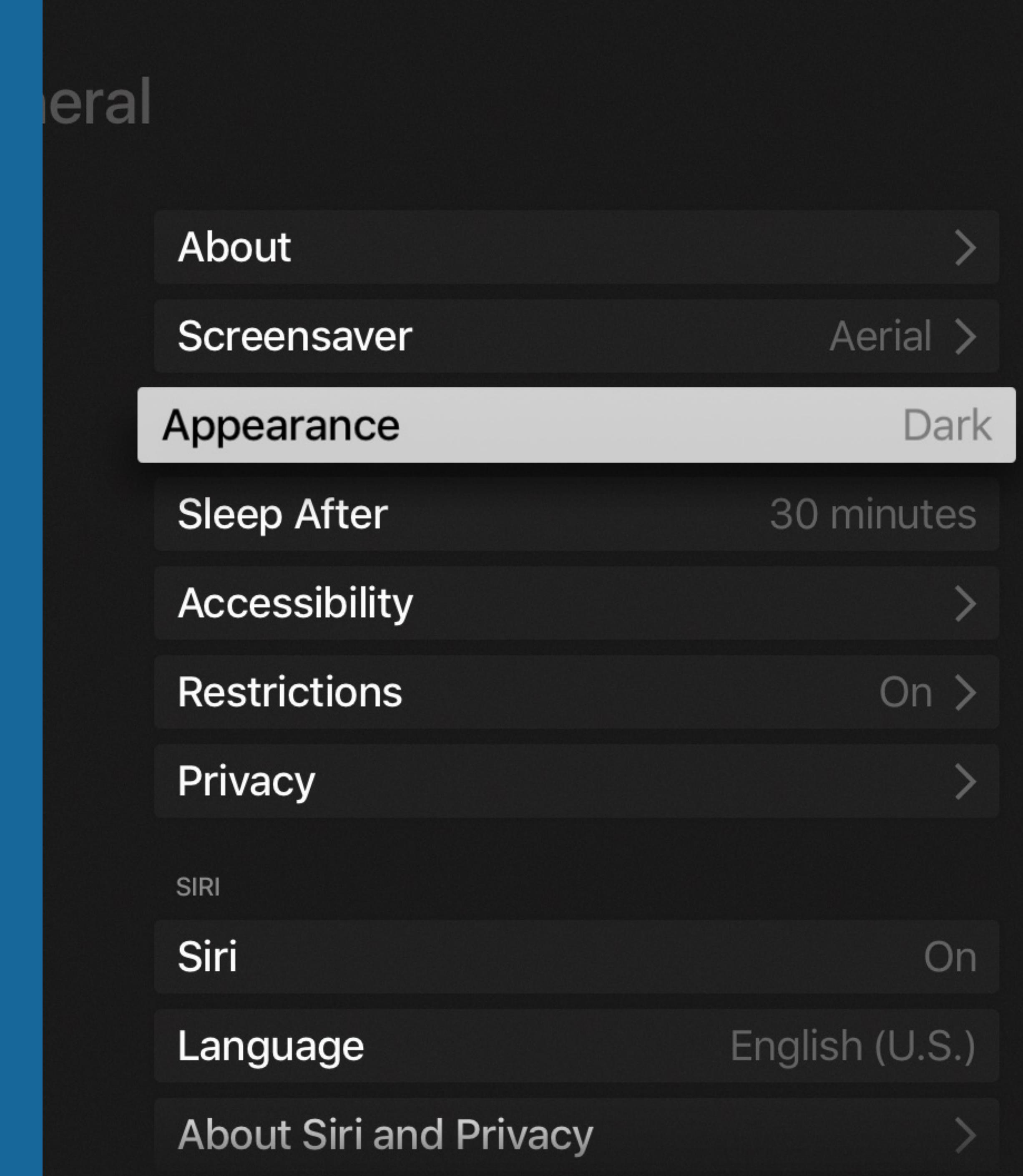


- tvOS 10.0+ has dark and light themes
 - Think about your icons and UI

APPLE TV

SUBTITLE

- Set the `UIUserInterfaceStyle` property in your app's info.plist to either Dark or Automatic



APPLE TV

```
let light = UITraitCollection(userInterfaceStyle: .light)  
let dark = UITraitCollection(userInterfaceStyle: .dark)  
  
UIButton(forTraitCollection:light).setTitleColor(.red(), for: [])  
UIButton(forTraitCollection:dark).setTitleColor(.blue(), for: [])
```

- Dynamically change your appearance

APPLE TV

- How to purchase tvOS apps
 - Directly from App Store on AppleTV
 - Universal purchasing enabled on linked iOS/tvOS app

Featured

Top Charts

Categories

Purchased



1. HBO GO



2. Zova



3. Daily Burn



4. Trunk



6. Afterpulse



7. Star Walk Kids



8. Galaxy on Fire



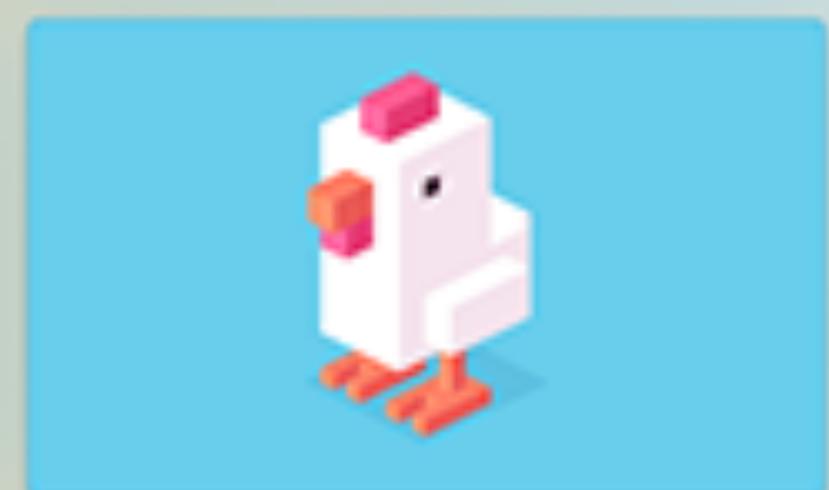
9. Perry's Peril



11. Beat Sports



12. YouTube



13. Crossy Road



14. G...

TVOS CATALOG

TVOS CATALOG

Controls

View Controllers

Text Entry

Focus

Search

UIAlertController

AlertsViewController

UICollectionViewCo...

CollectionViewContr...

UIPageViewController

PageViewController

AVPlayerViewContr...

VideoPlayerViewCo...

Show Single Option Alert

Show Multiple Option Alert

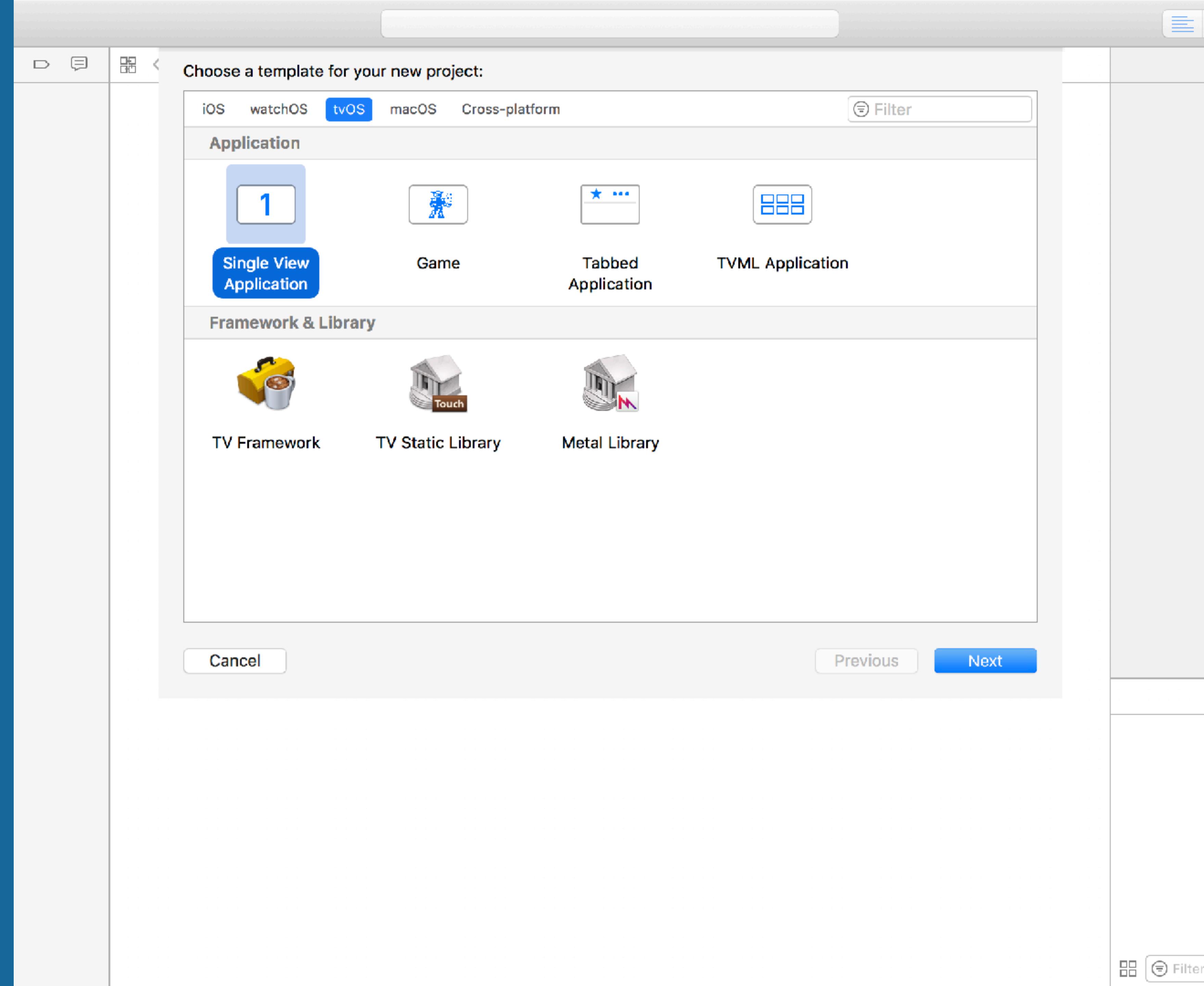
Show Destructive Option Alert



TVOS APPS

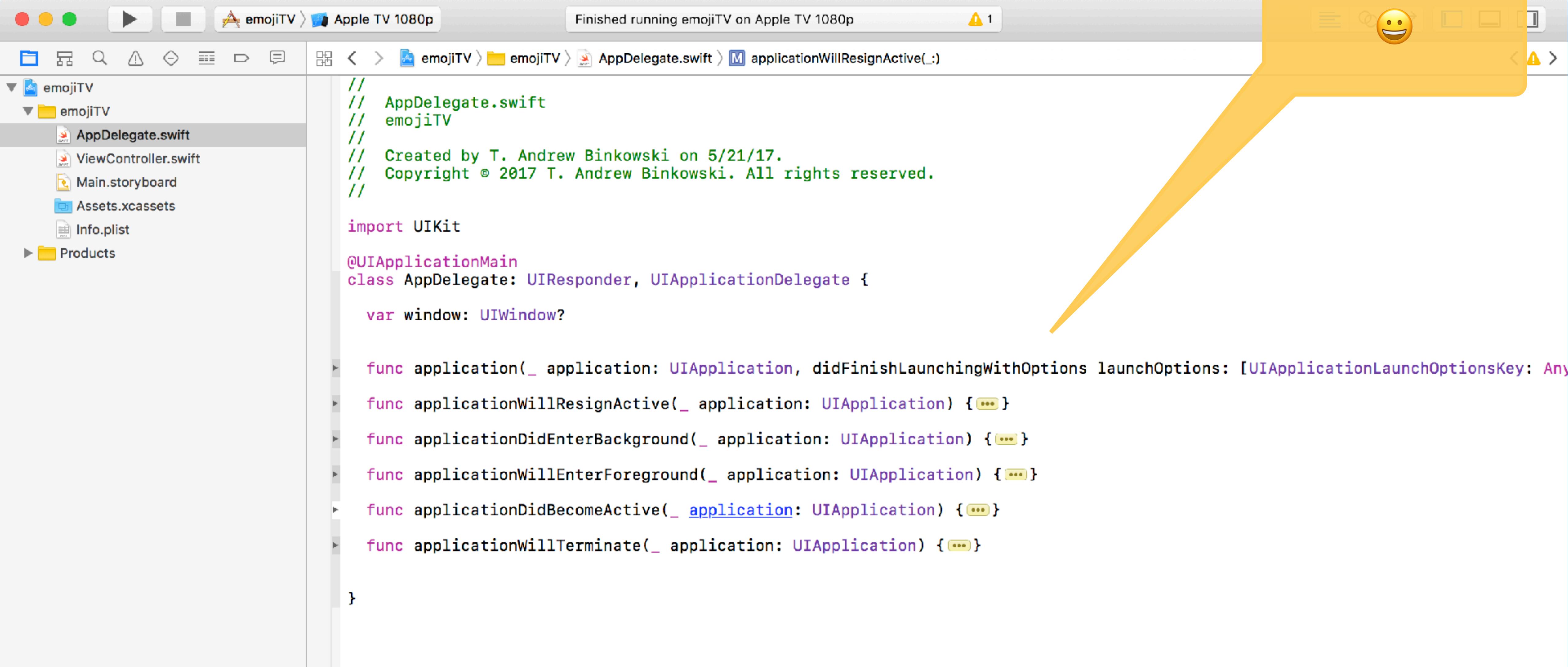
TVOS

- Xcode Templates



TVOS

Looks
familiar



A screenshot of the Xcode IDE interface. The title bar shows "emojiTV" and "Apple TV 1080p". The status bar indicates "Finished running emojiTV on Apple TV 1080p" with 1 warning. The main window displays the "AppDelegate.swift" file for the "emojiTV" project. The code is written in Swift and defines the AppDelegate class, which conforms to UIResponder and UIApplicationDelegate. It includes standard application lifecycle methods like application:didFinishLaunchingWithOptions:, applicationWillResignActive:, applicationWillEnterForeground:, applicationWillBecomeActive:, and applicationWillTerminate:. A yellow callout bubble points from the text "applicationWillResignActive(_)" towards the "Looks familiar" text in the top right corner.

```
// AppDelegate.swift
// emojiTV
//
// Created by T. Andrew Binkowski on 5/21/17.
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        // ...
        return true
    }

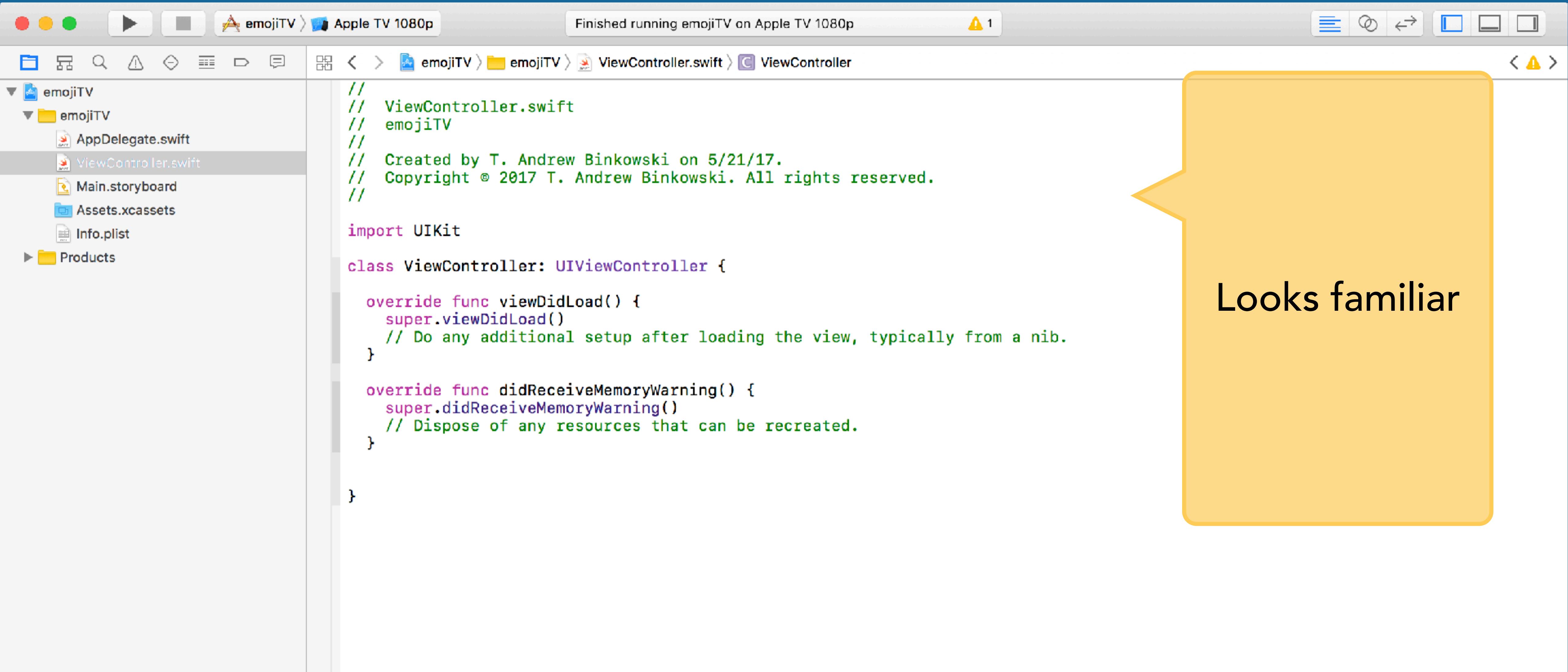
    func applicationWillResignActive(_ application: UIApplication) {
        // ...
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // ...
    }

    func applicationWillBecomeActive(_ application: UIApplication) {
        // ...
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // ...
    }
}
```

TVOS



The screenshot shows the Xcode interface with a project named "emojiTV" running on an Apple TV 1080p. The code editor displays the "ViewController.swift" file, which contains the following code:

```
// ViewController.swift
// emojiTV
//
// Created by T. Andrew Binkowski on 5/21/17.
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

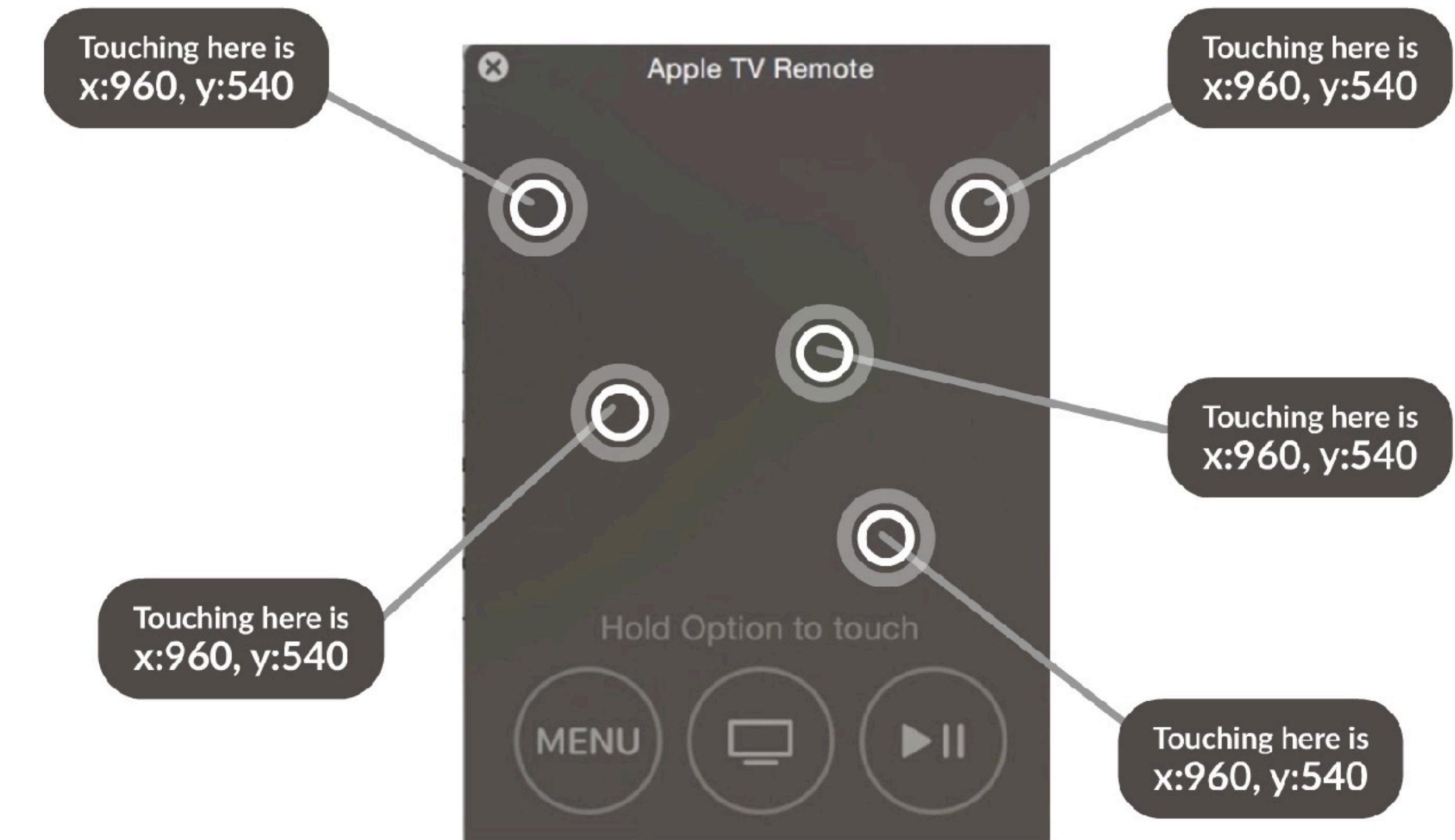
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

A yellow callout bubble with the text "Looks familiar" points to the code, indicating that the structure and syntax are similar to iOS development.

**TOUCHES AND
PRESSES**

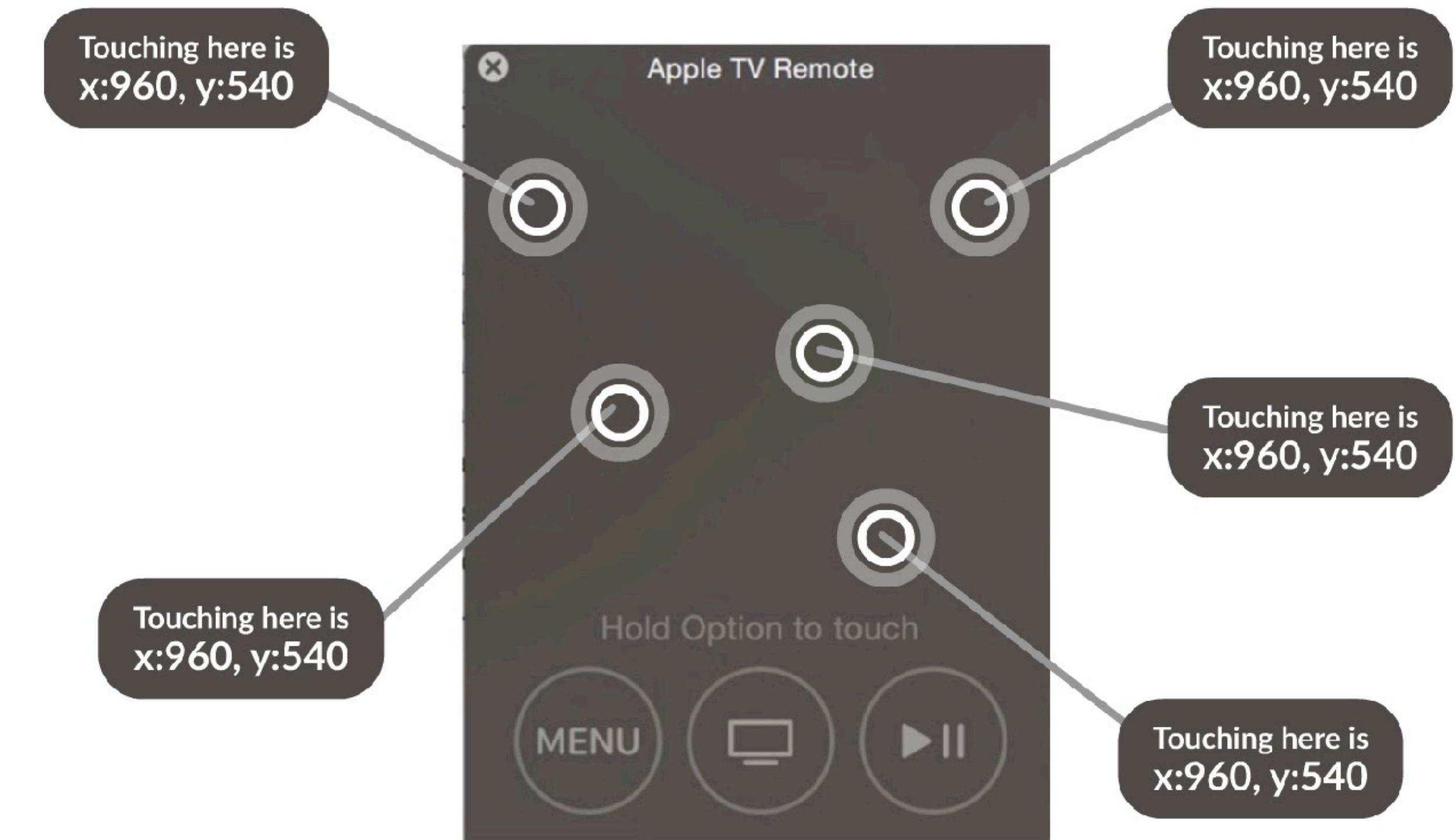
TOUCHES AND PRESSES

- `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



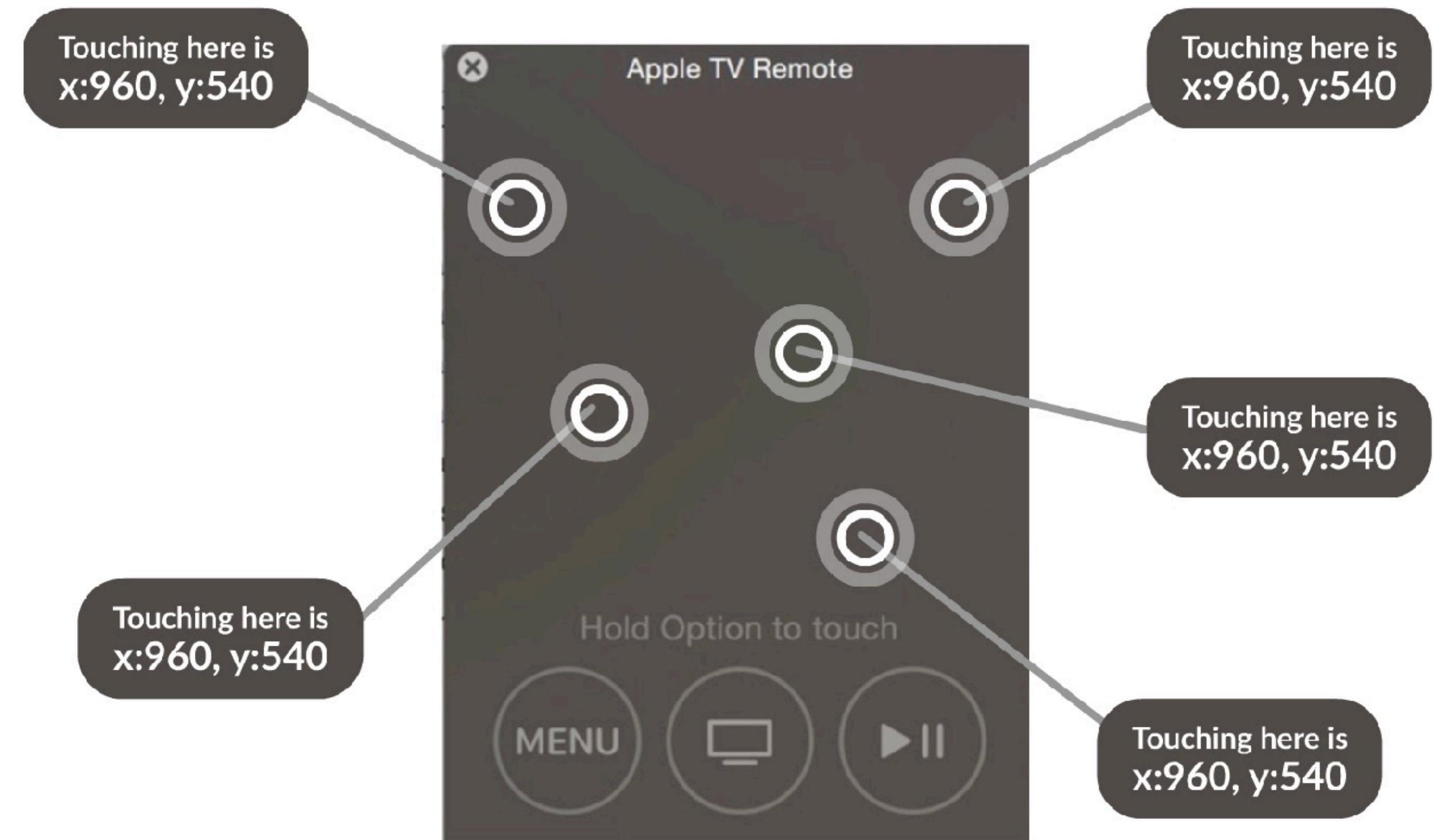
TOUCHES AND PRESSES

- Do not want a mouse/pointer type interaction



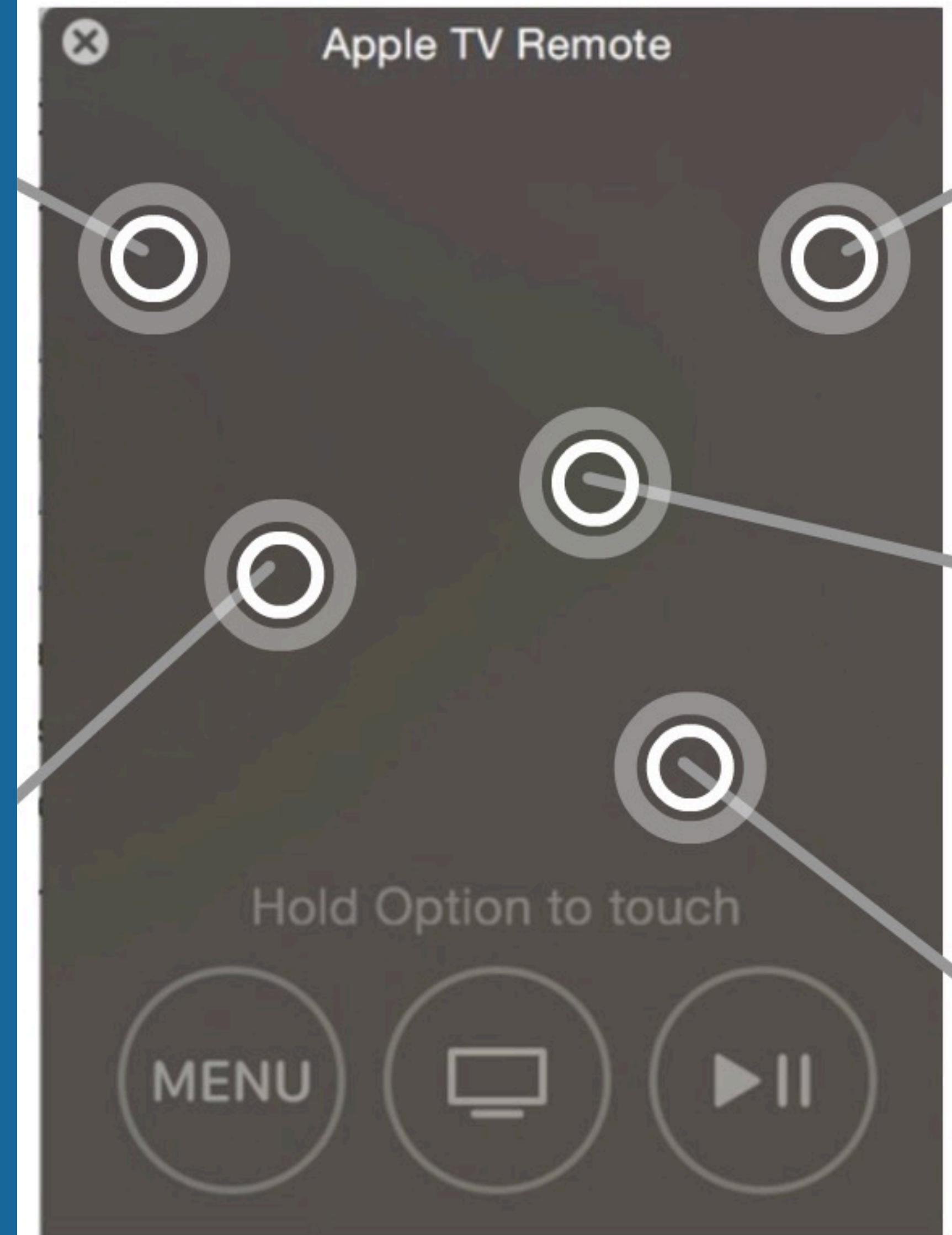
TOUCHES AND PRESSES

- All UIView methods respond
 - TouchesBegan
 - TouchesEnded
 - ...



TOUCHES AND PRESSES

- UIPress represents the up or down state of a physical button
- May be pressure-sensitive
- UIGestureRecognizer



TOUCHES AND PRESSES

```
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?) {}
```

TOUCHES AND PRESSES

Press Types An illustrated guide

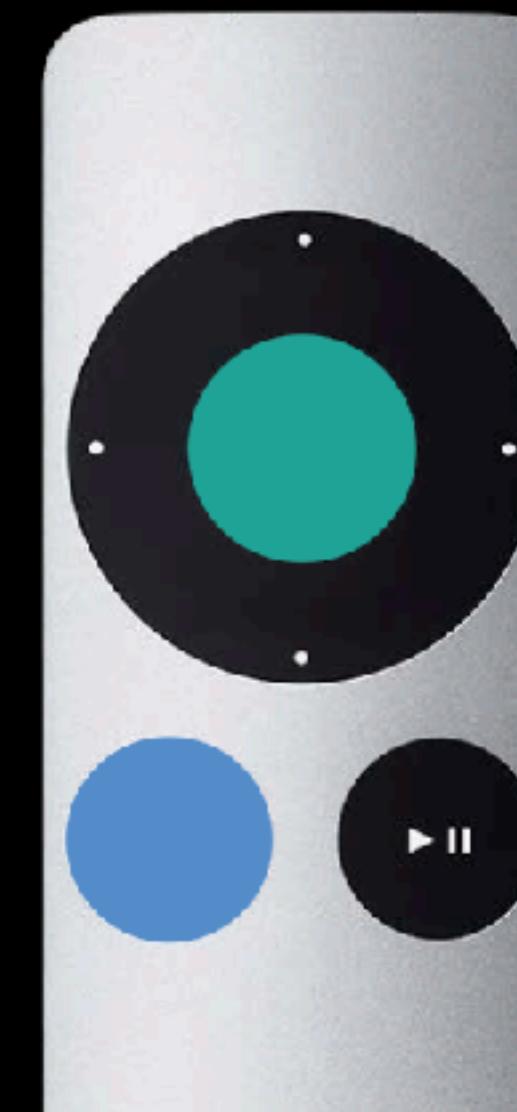
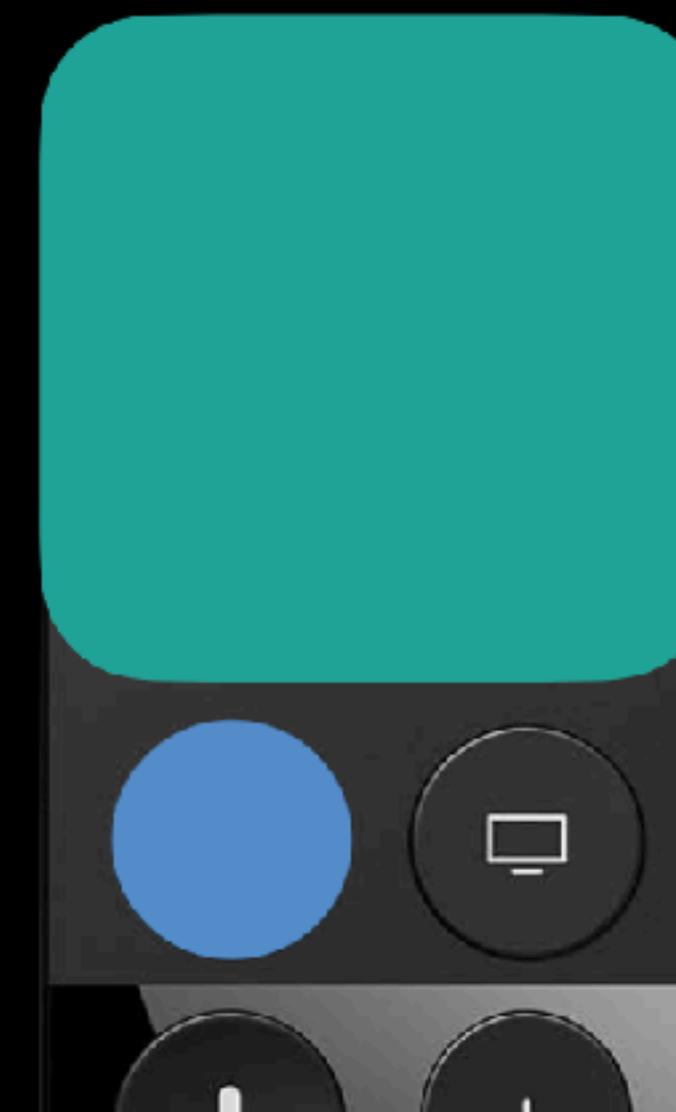
■ UIPressTypeSelect



TOUCHES AND PRESSES

Press Types An illustrated guide

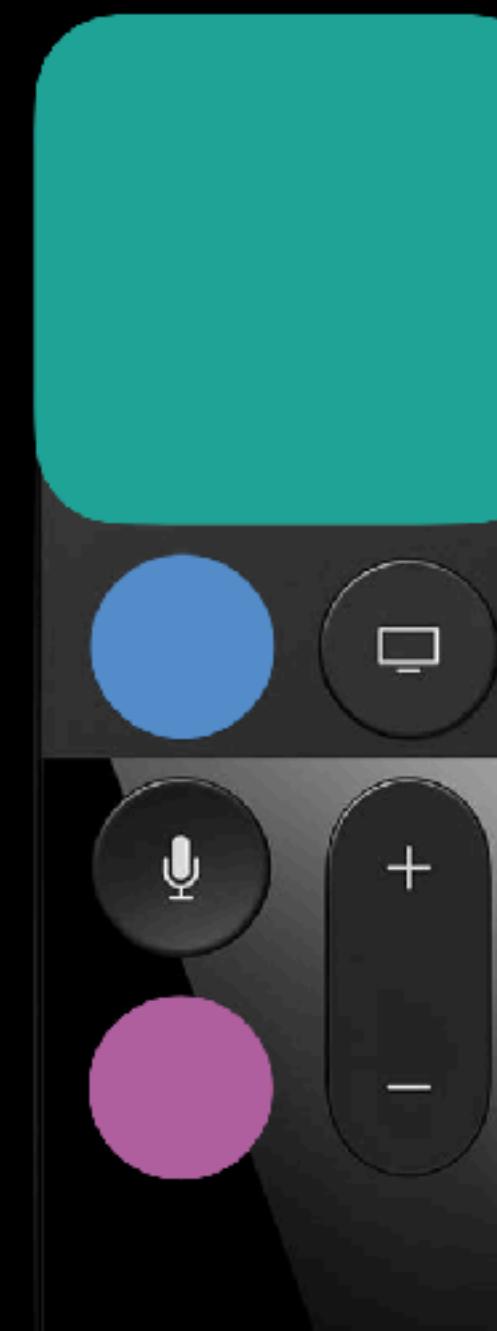
- **UIPressTypeSelect**
- **UIPressTypeMenu**



TOUCHES AND PRESSES

Press Types An illustrated guide

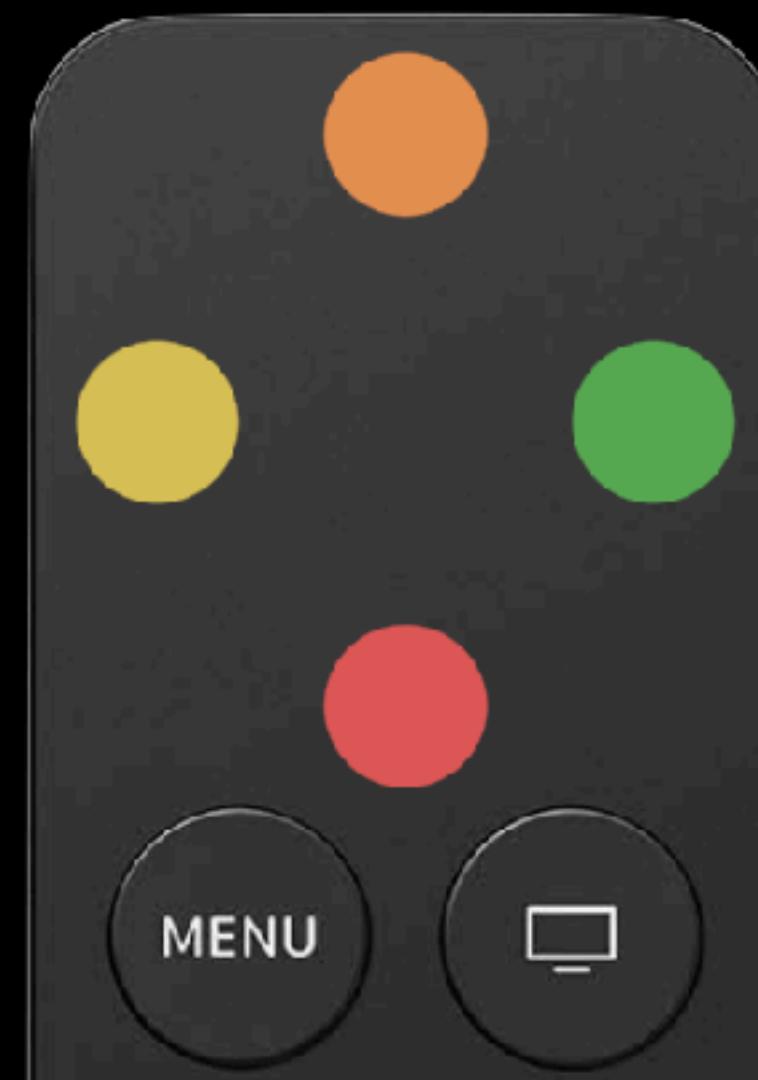
- UIPressTypeSelect
- UIPressTypeMenu
- UIPressTypePlayPause



TOUCHES AND PRESSES

Press Types An illustrated guide

- UIPressTypeUpArrow
- UIPressTypeDownArrow
- UIPressTypeLeftArrow
- UIPressTypeRightArrow



TOUCHES AND PRESSES

```
// 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
```

TOUCHES AND PRESSES

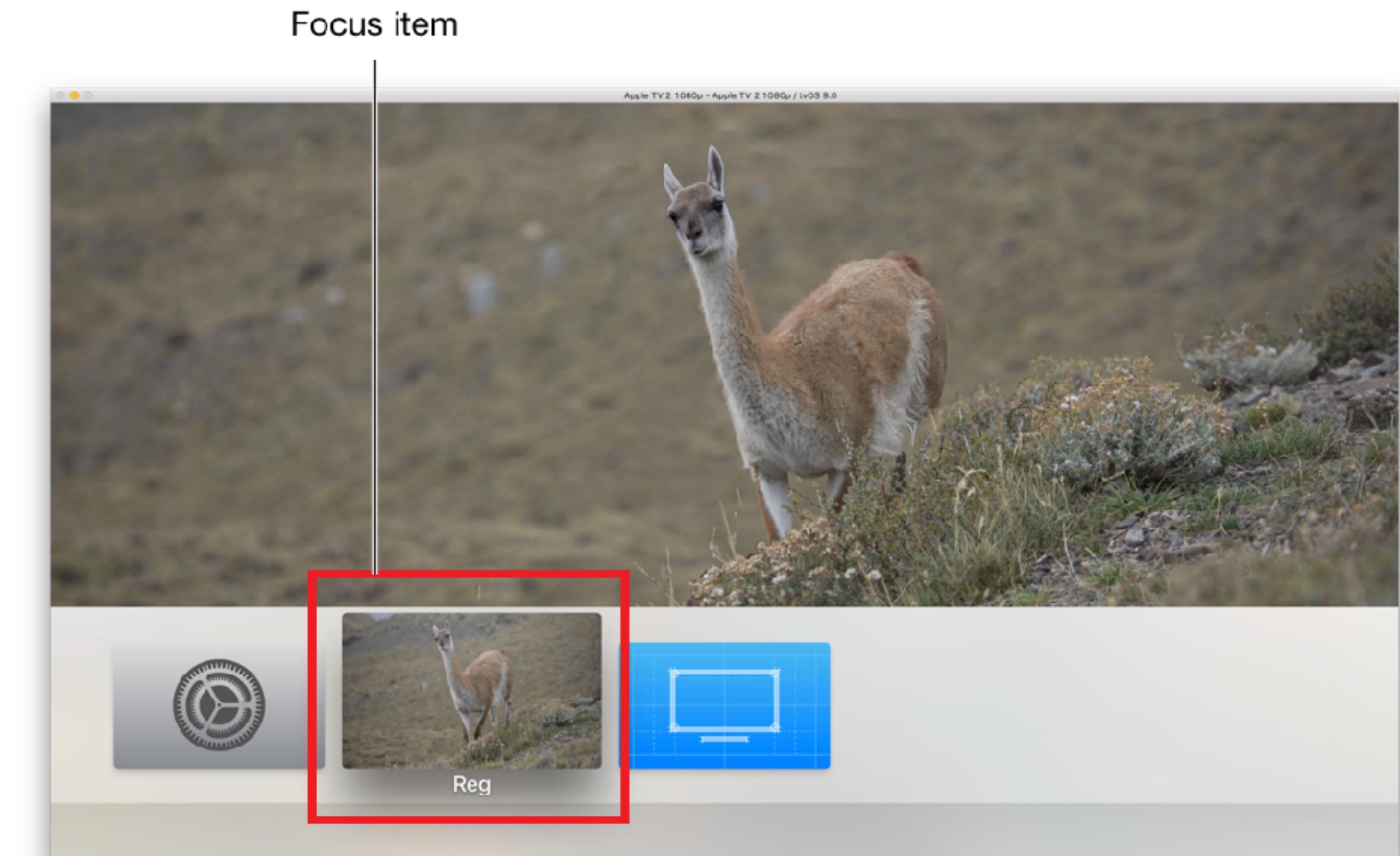
- Users must be able to exit your app using the Menu button
- This is a requirement that App Review specifically looks for
- The event must reach **UIApplication pressesEnded:withEvent:** in order for the app to exit



FOCUS

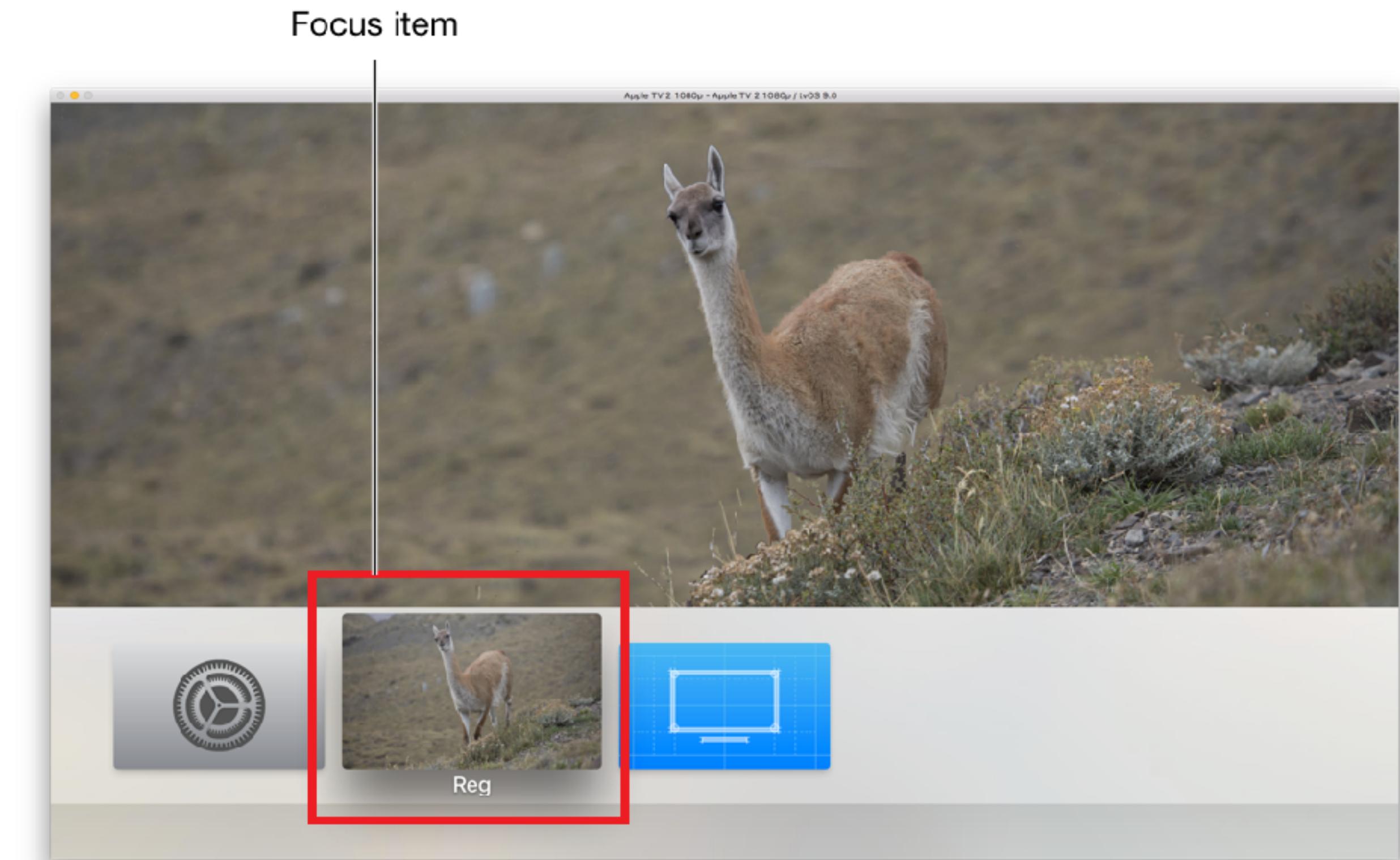
TVOS APPS

- Users interact with AppleTV using remote, game controller, iOS app
- As users navigate to items they are "focused"
 - Highlighted effects
- Focused items receive "action" when triggered



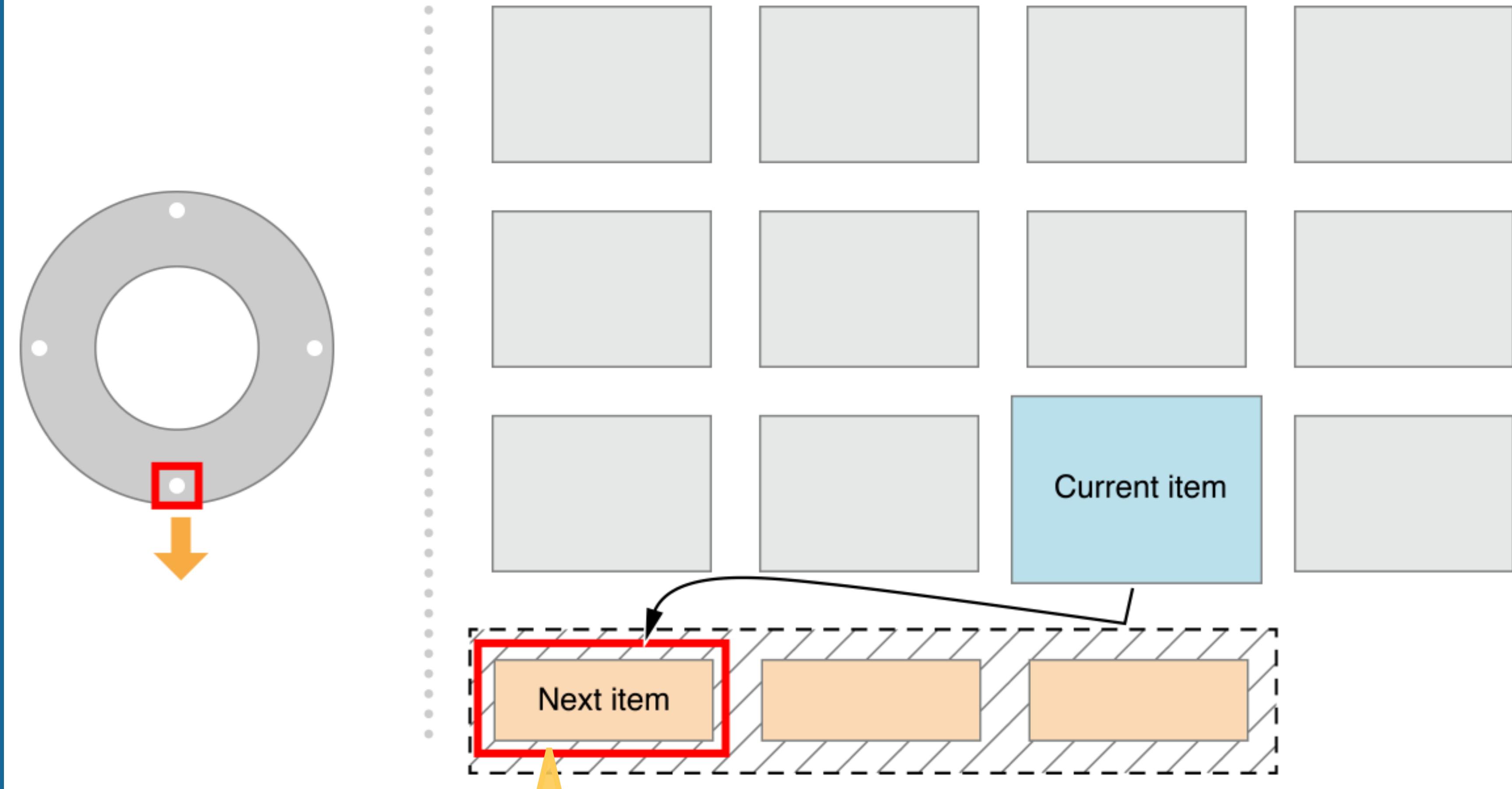
TVOS APPS

- Focus based interfaces are provided automatically (for the most part)
 - Views that make sense (eg. buttons)
- Cannot move focus programmatically
 - User in control



TVOS APPS

- The focus engine only responds to user movements
- Movements are restricted to prevent wandering around the screen



What is the first "down" item?
Not what is below item.

TVOS APPS



MORE INFO

BUY

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

A yellow curved arrow points from the "MORE INFO" button at the bottom left towards the "BUY" button at the top right.

TVOS APPS



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

BUY

MORE INFO

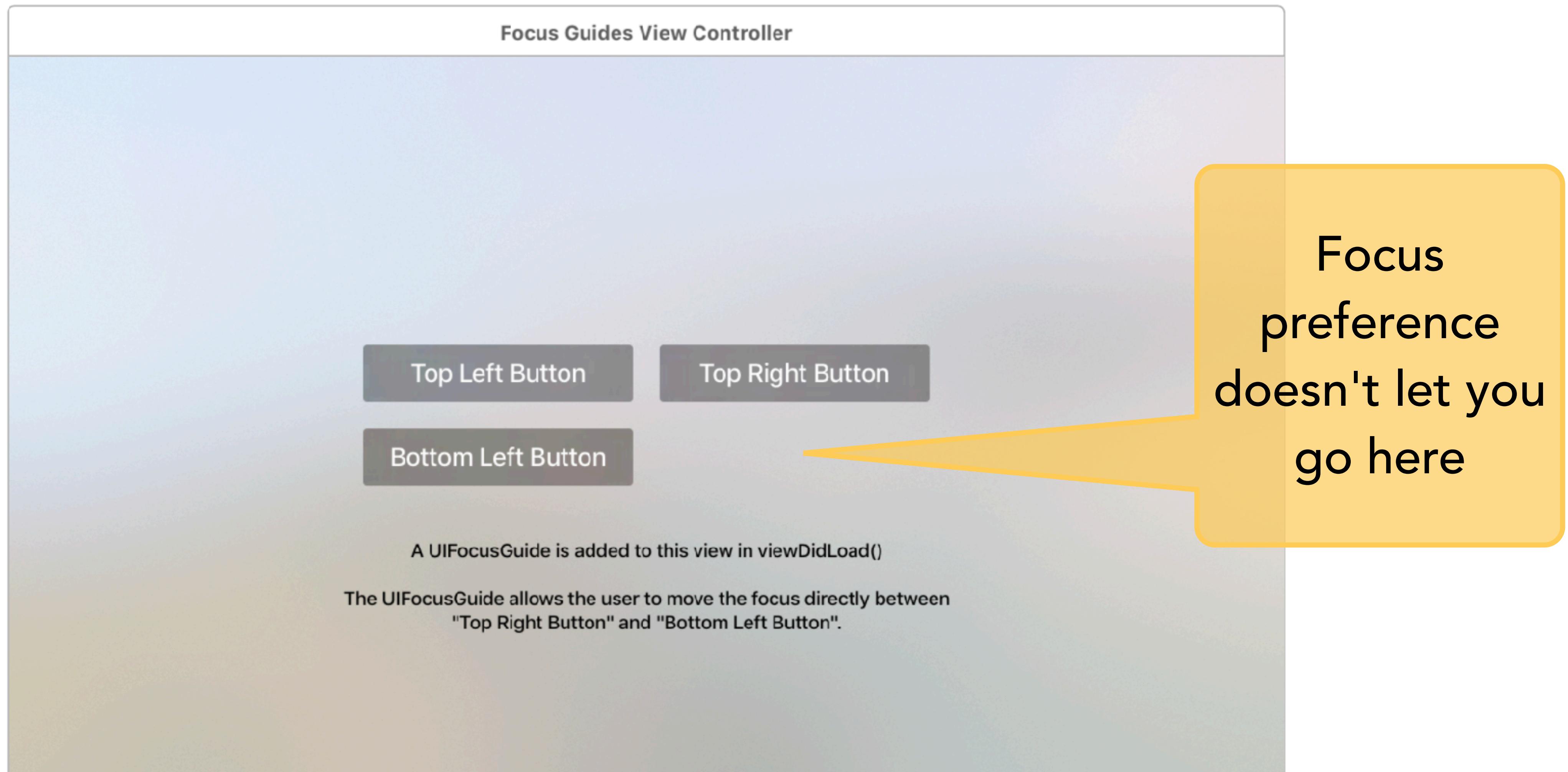


Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Our focus guide!

TVOS APPS



TVOS APPS

Focus Guides View Controller

Top Left Button Top Right Button

Bottom Left Button

A UIFocusGuide is added to this view in viewDidLoad()

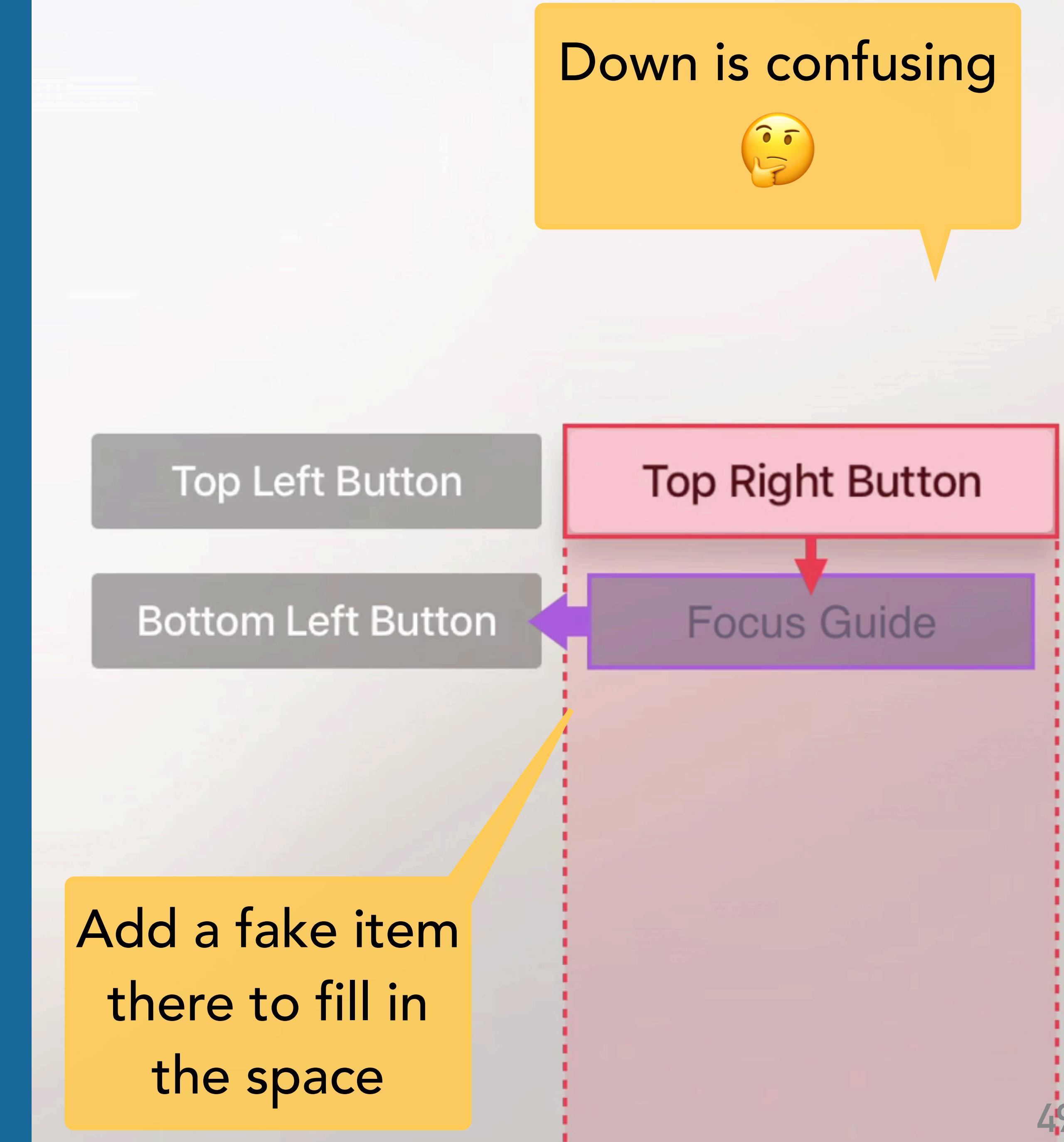
The UIFocusGuide allows the user to move the focus directly between "Top Right Button" and "Bottom Left Button".

We need to create a focus guide for continuity

TVOS APPS

SUBTITLE

- `Focus guide` allows you to create alignments to conform to the focus preferences
 - Typically not visually aligned

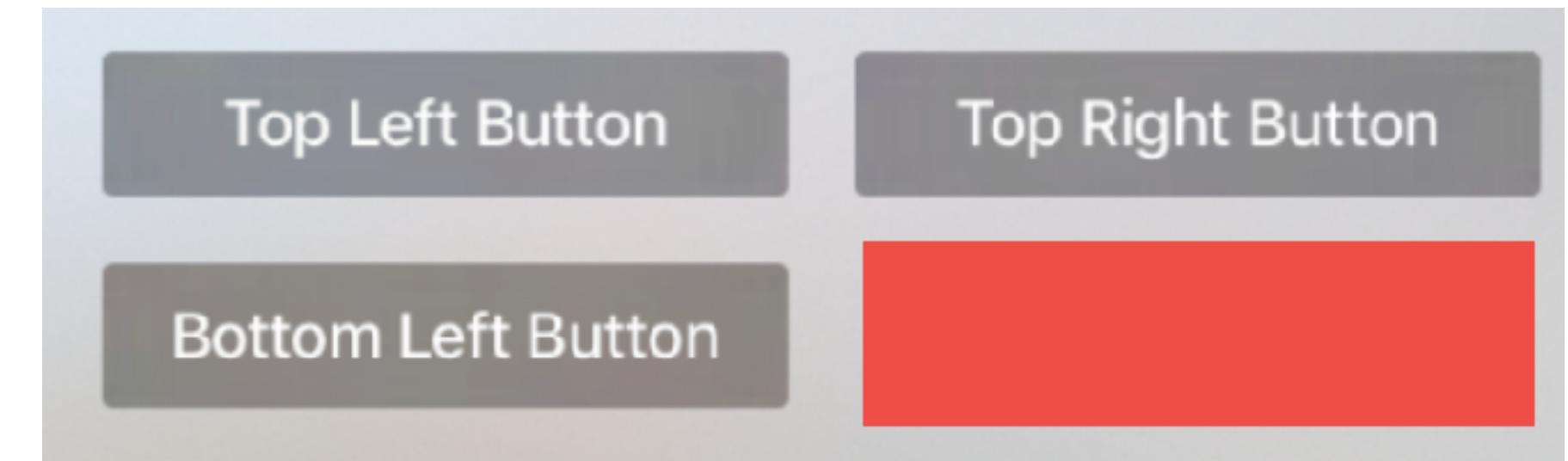


FOCUS

```
/*
    Create a focus guide to fill the gap below button 2 and to the right
    of button 3.
*/
focusGuide = UIFocusGuide()
view.addLayoutGuide(focusGuide)

// Anchor the top left of the focus guide.
focusGuide.leftAnchor.constraint(equalTo: topRightButton.leftAnchor).isActive = true
focusGuide.topAnchor.constraint(equalTo: bottomLeftButton.topAnchor).isActive = true

// Anchor the width and height of the focus guide.
focusGuide.widthAnchor.constraint(equalTo: topRightButton.widthAnchor).isActive = true
focusGuide.heightAnchor.constraint(equalTo: bottomLeftButton.heightAnchor).isActive = true
```

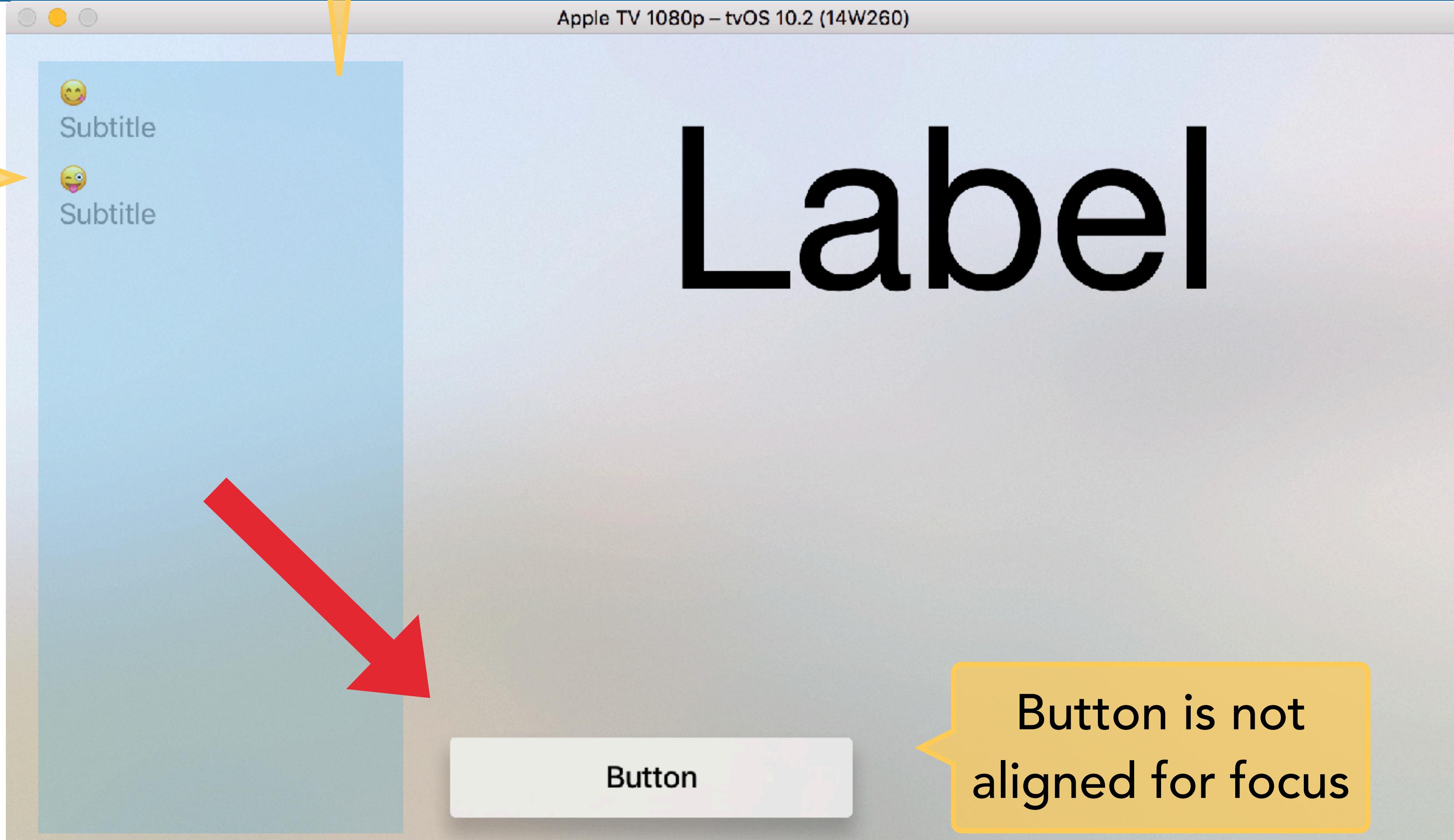


- Add a custom focus guide

TVOS APPS

Table

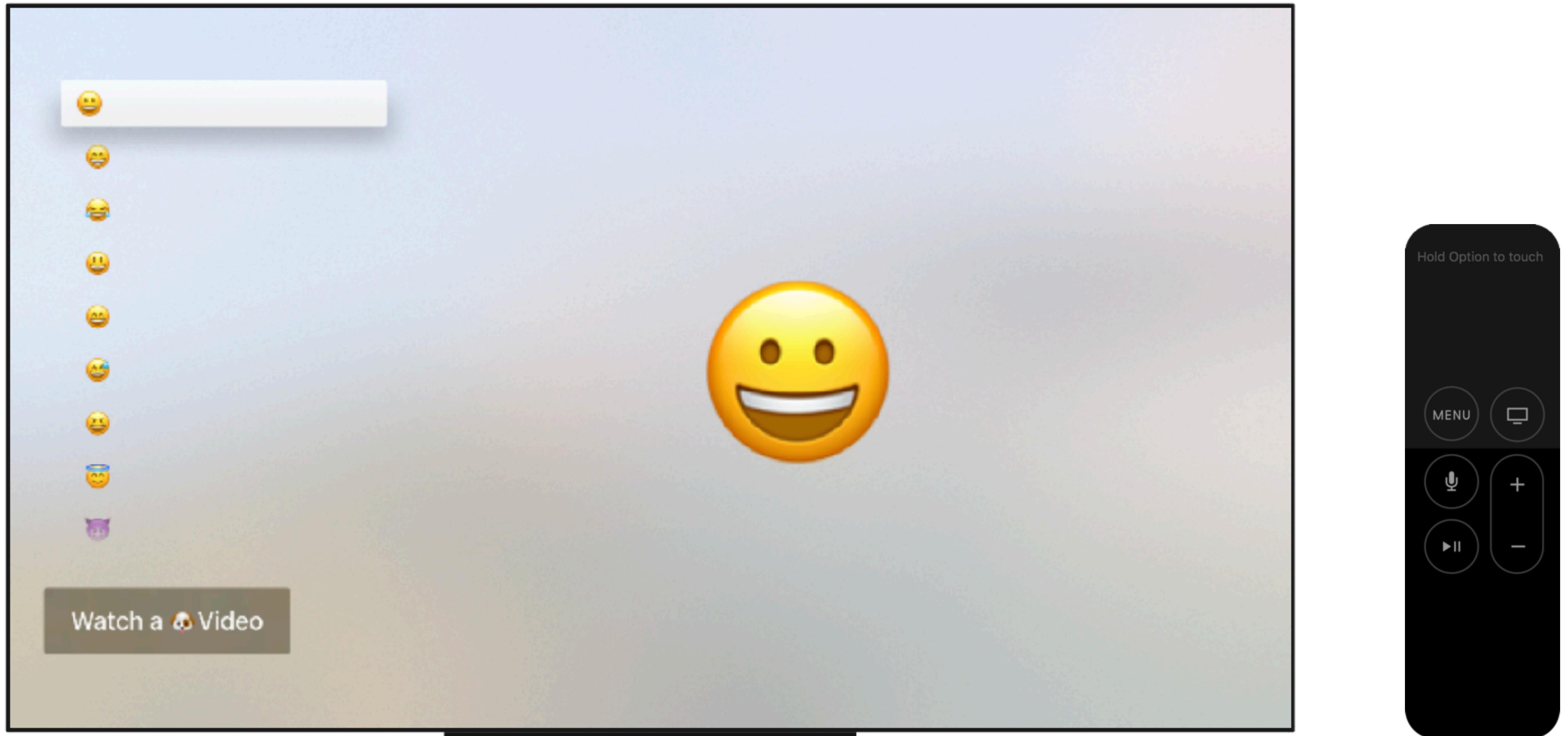
Aligned
for
focus



Button is not
aligned for focus

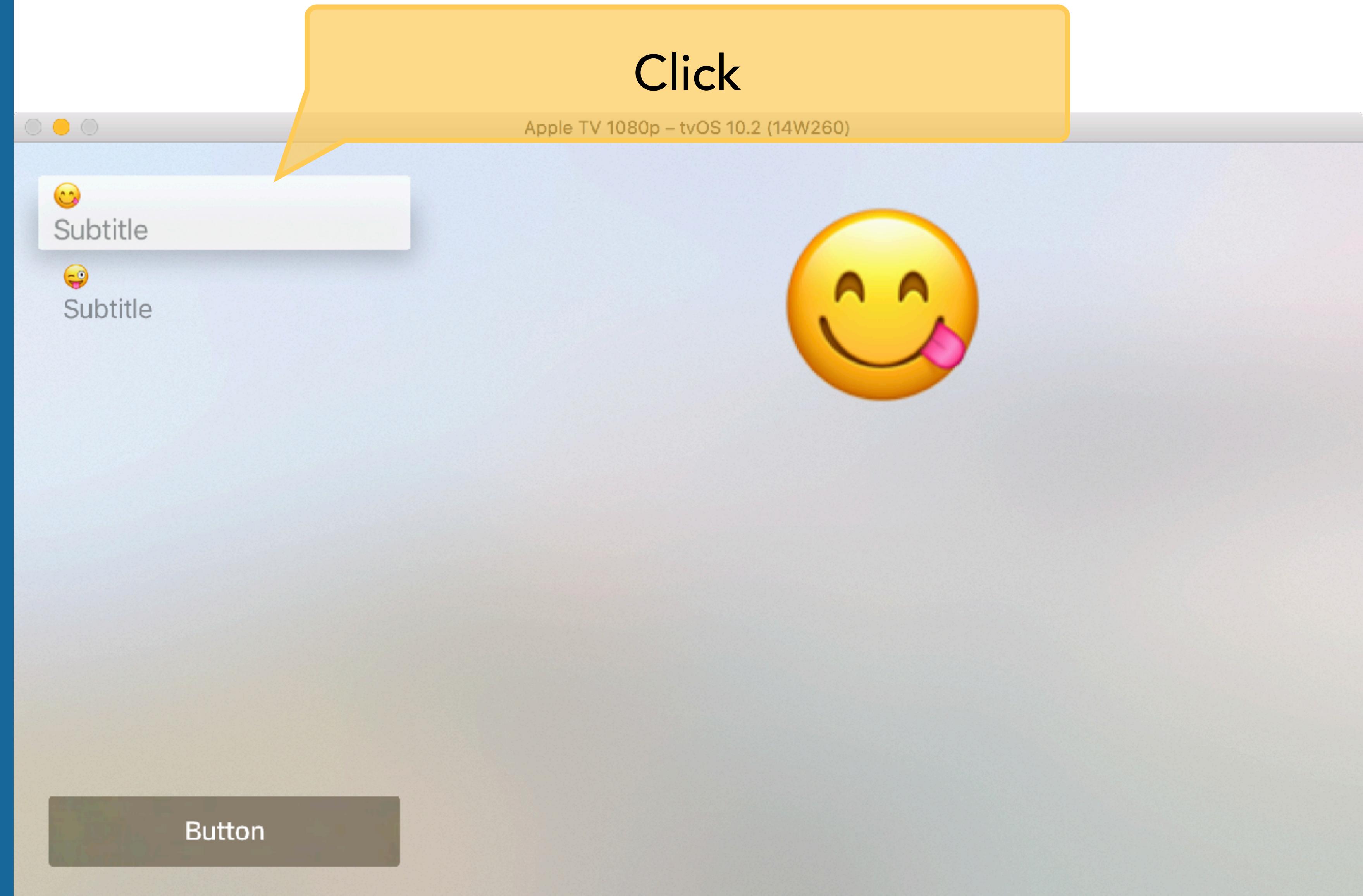
EMOJI TV

EMOJI TV



TVOS APPS

- Focus changes on a cell, update the label



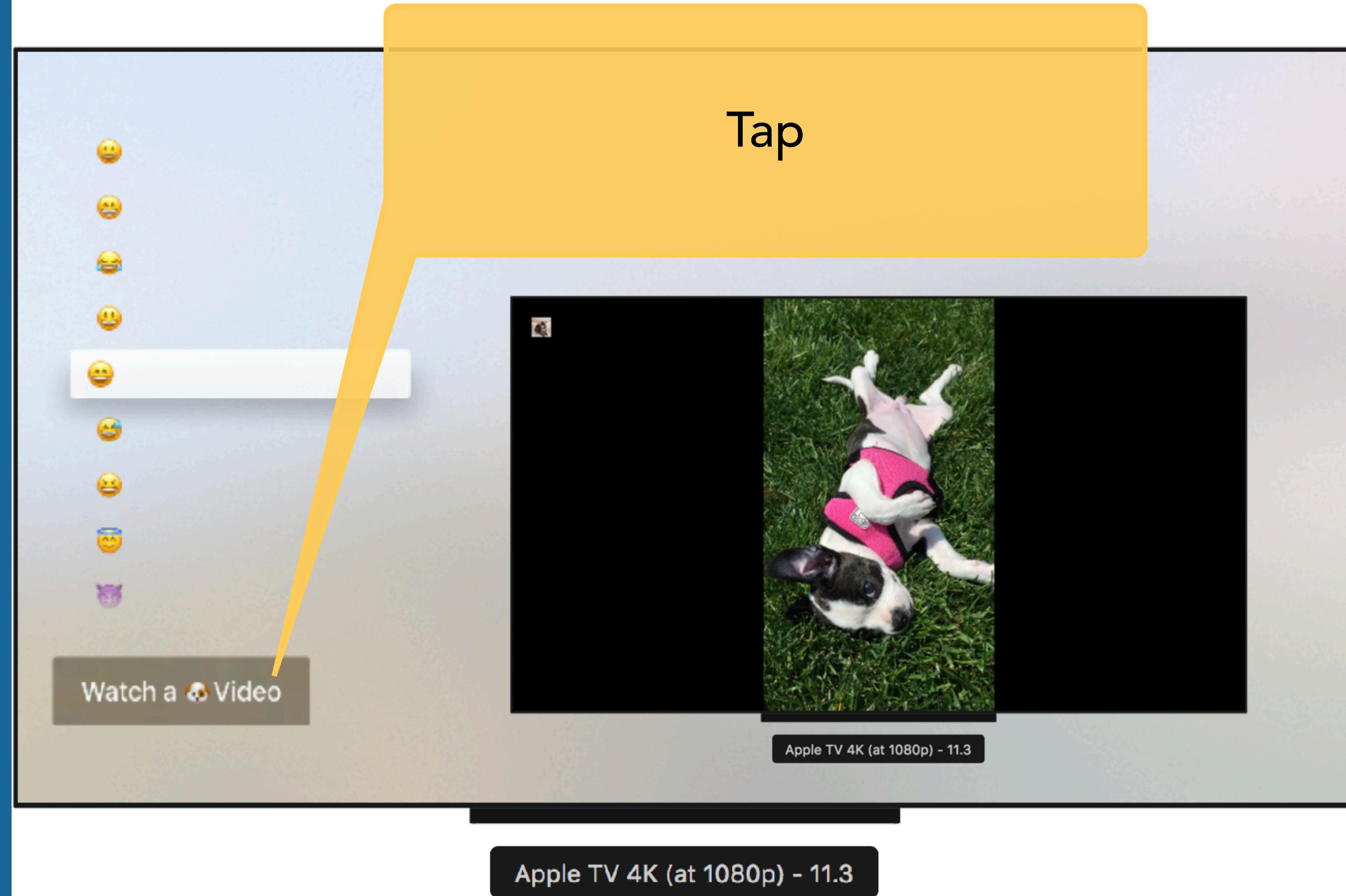
TVOS APPS

- Click on table cell show the row



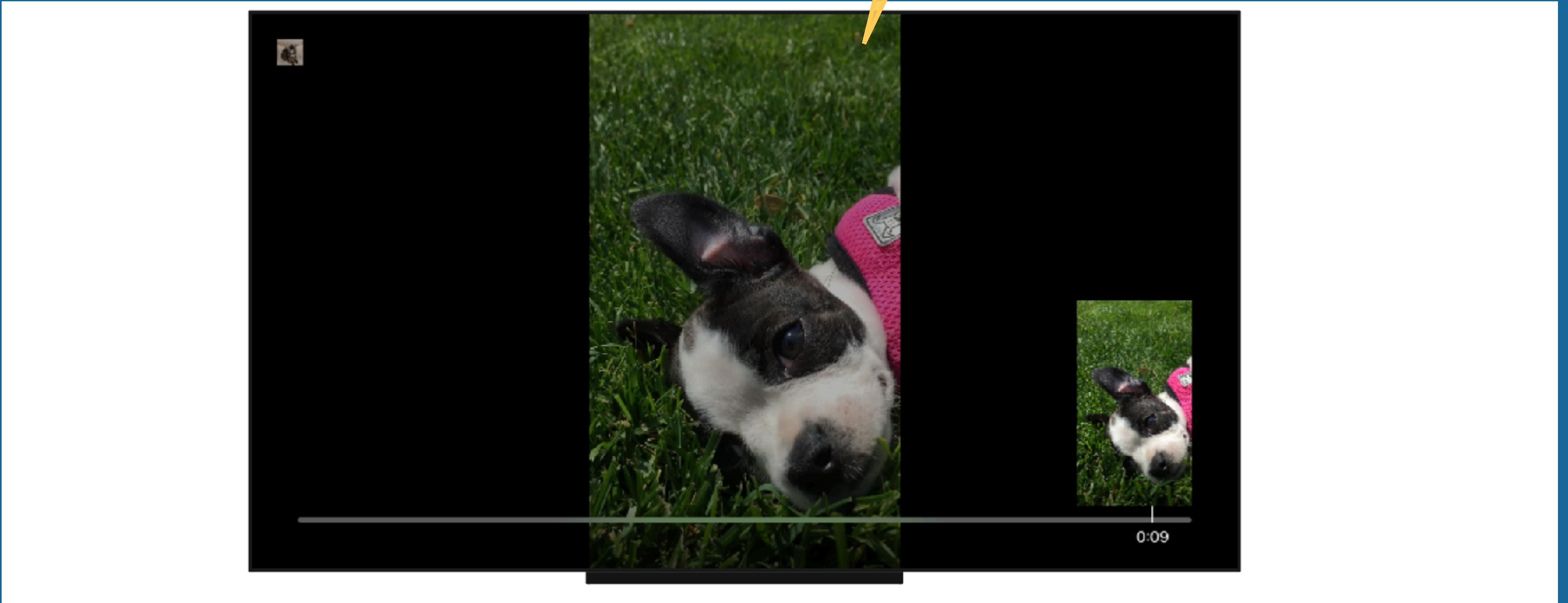
TVOS APPS

- Segue to View Controller to show a video



TVOS APPS

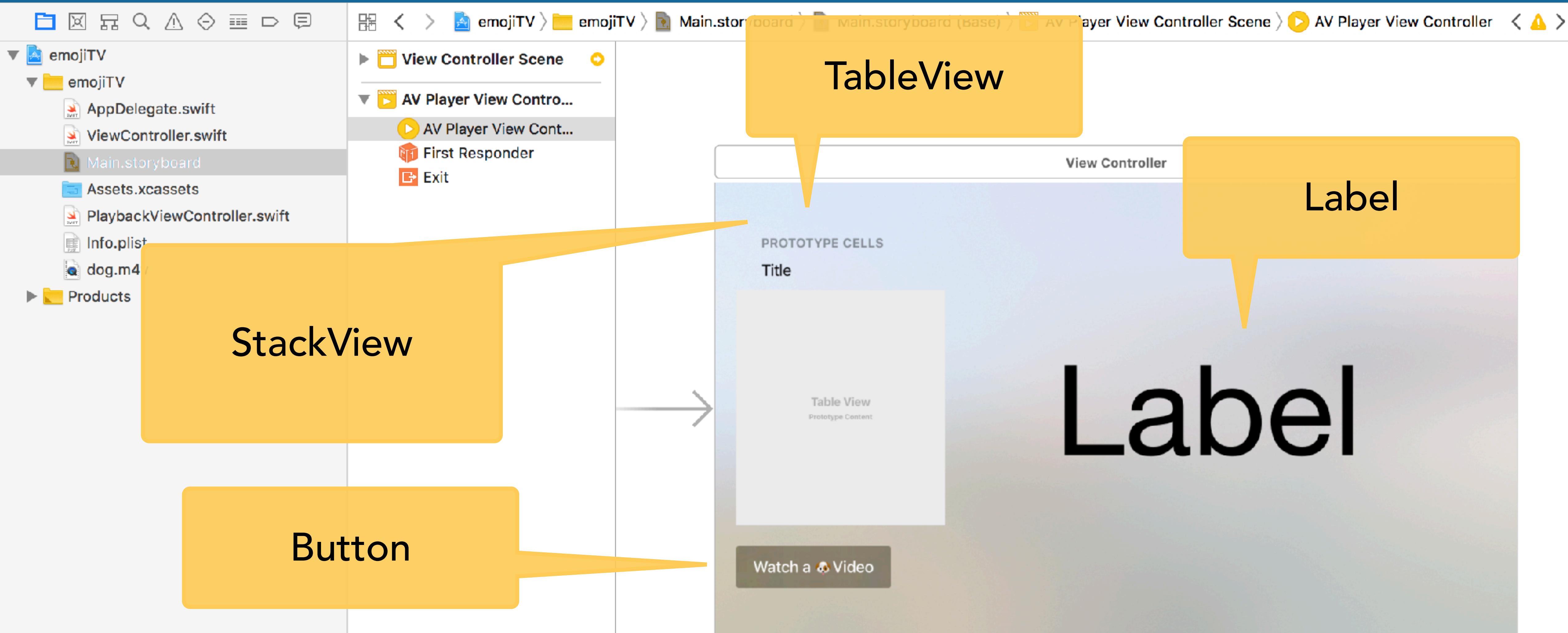
Overlay view



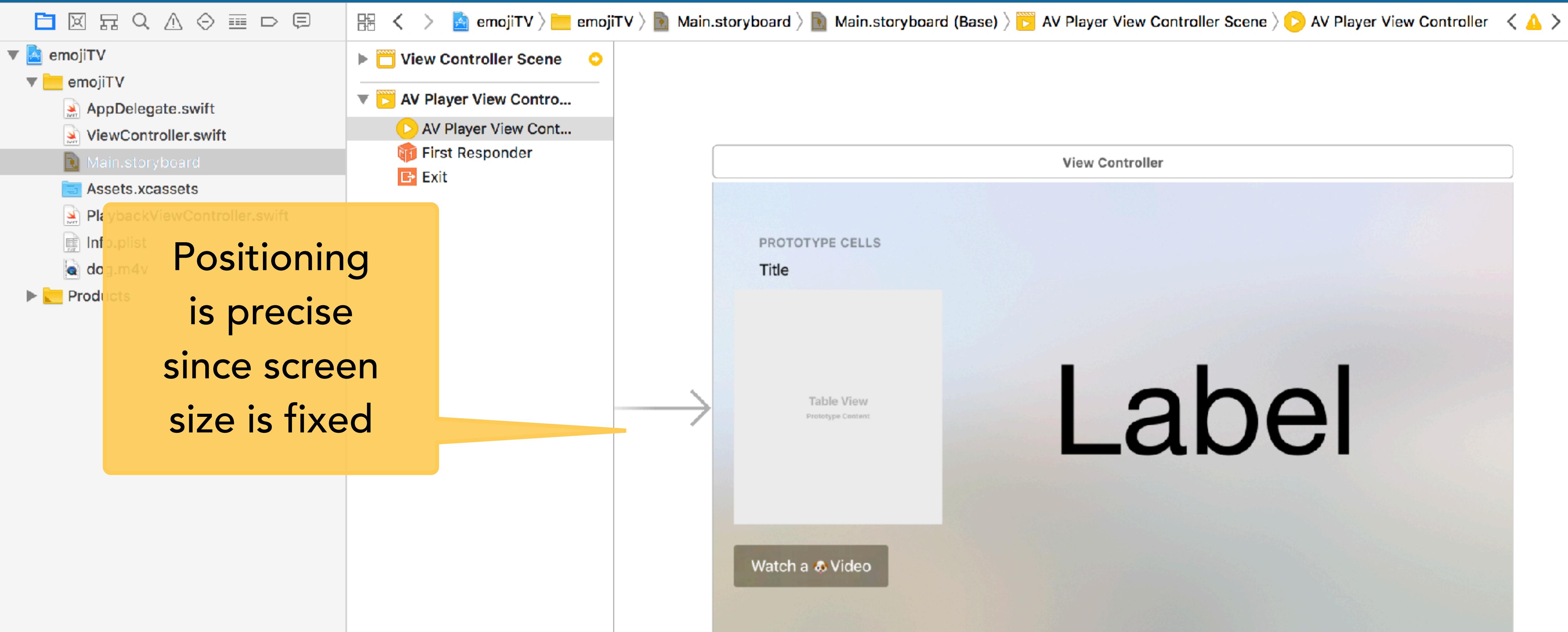
- AVPlayerViewController

STORYBOARD

TVOS APPS



TVOS APPS



TVOS APPS

Nothing new

```
// MARK: - Actions
@IBAction func tapButton(_ sender: UIButton) {
    print("Tapped: \(String(describing: sender.titleLabel?.description))")
}

// MARK: - Outlets
@IBOutlet weak var bigLabel:UILabel!
@IBOutlet weak var button: UIButton!
@IBOutlet weak var tableView: UITableView! {
    didSet {
        self.tableView.delegate = self
        self.tableView.dataSource = self
    }
}
```

APP DELEGATE

TVOS APPS

- AppDelegate

The screenshot shows the Xcode interface with the following details:

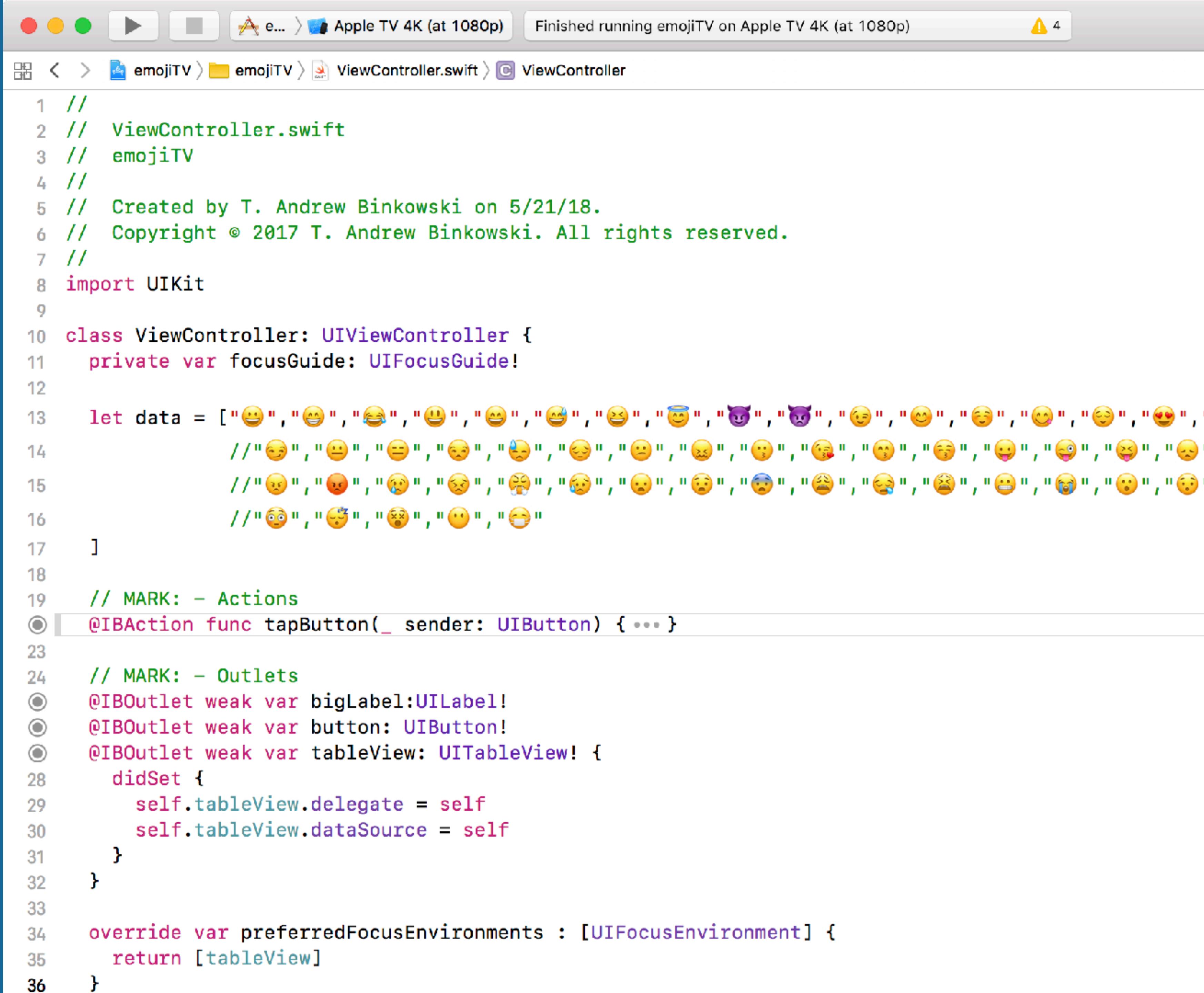
- Title Bar:** Shows the project name "emojiTV" and the target "Apple TV 4K (at 1080p)". It also displays the message "Finished running emojiTV on Apple TV 4K (at 1080p)".
- File Navigator:** Shows the project structure:
 - emojiTV (group)
 - emojiTV (group)
 - AppDelegate.swift (selected)
 - ViewController.swift
 - Main.storyboard
 - Assets.xcassets
 - PlaybackViewController.swift
 - Info.plist
 - dog.m4v
 - Products
- Editor Area:** Displays the content of the selected `AppDelegate.swift` file. The code is as follows:

```
1 //  
2 // AppDelegate.swift  
3 // emojiTV  
4 //  
5 // Created by T. Andrew Binkowski on 5/21/17.  
6 // Copyright © 2017 T. Andrew Binkowski. All rights reserved.  
7 //  
8  
9 import UIKit  
10  
11 @UIApplicationMain  
12 class AppDelegate: UIResponder, UIApplicationDelegate {  
13  
14     var window: UIWindow?  
15  
16  
17     func application(_ application: UIApplication, didFinishLaunchingWith launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool  
18         // Override point for customization after application launch.  
19         return true  
20     }  
21  
22  
23 }  
24  
25
```

VIEW CONTROLLER

TVOS APPS

- ViewController implements a UITableView



The screenshot shows the Xcode interface with the title bar "Apple TV 4K (at 1080p)" and status bar "Finished running emojiTV on Apple TV 4K (at 1080p) 4". The project navigation bar shows "emojiTV > emojiTV > ViewController.swift > ViewController". The code editor displays the following Swift code:

```
1 //  
2 // ViewController.swift  
3 // emojiTV  
4 //  
5 // Created by T. Andrew Binkowski on 5/21/18.  
6 // Copyright © 2017 T. Andrew Binkowski. All rights reserved.  
7 //  
8 import UIKit  
9  
10 class ViewController: UIViewController {  
11     private var focusGuide: UIFocusGuide!  
12  
13     let data = ["😊", "😁", "😂", "😃", "😄", "😅", "😆", "😇", "😈", "👿", "😉", "😊", "😊", "😊", "😊", "😊",  
14         // "😏", "😐", "😑", "Ҫ",  
15         // "Ҫ",  
16         // "Ҫ",  
17     ]  
18  
19     // MARK: - Actions  
20     @IBAction func tapButton(_ sender: UIButton) { ... }  
21  
22     // MARK: - Outlets  
23     @IBOutlet weak var bigLabel: UILabel!  
24     @IBOutlet weak var button: UIButton!  
25     @IBOutlet weak var tableView: UITableView! {  
26         didSet {  
27             self.tableView.delegate = self  
28             self.tableView.dataSource = self  
29         }  
30     }  
31  
32     override var preferredFocusEnvironments : [UIFocusEnvironment] {  
33         return [tableView]  
34     }  
35 }
```

The code defines a ViewController class that implements the UITableViewDataSource and UITableViewDelegate protocols. It contains a private var focusGuide and a let data array filled with various emojis. It also includes an @IBAction for a tap event on a button, and outlets for a bigLabel, a button, and a tableView.

TVOS APPS

```
func numberOfSections(in tableView: UITableView) -> Int {  
    return 1  
}  
  
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
    return self.data.count  
}  
}  
  
extension ViewController: UITableViewDelegate {  
  
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)  
        cell.textLabel?.text = data[indexPath.row]  
  
        return cell  
    }  
  
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {  
        print("tap: \(indexPath.row)")  
        self.bigLabel?.text = "Tap \(indexPath.row)"  
    }  
}
```

Nothing new

TVOS APPS

Tap on cell
triggers update

```
func numberOfSections(in tableView: UITableView) -> Int {  
    return 1  
}  
  
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
    return self.data.count  
}  
}  
  
extension ViewController: UITableViewDelegate {  
  
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)  
        cell.textLabel?.text = data[indexPath.row]  
  
        return cell  
    }  
  
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {  
        print("tap: \(indexPath.row)")  
        self.bigLabel?.text = "Tap \(indexPath.row)"  
    }  
}
```

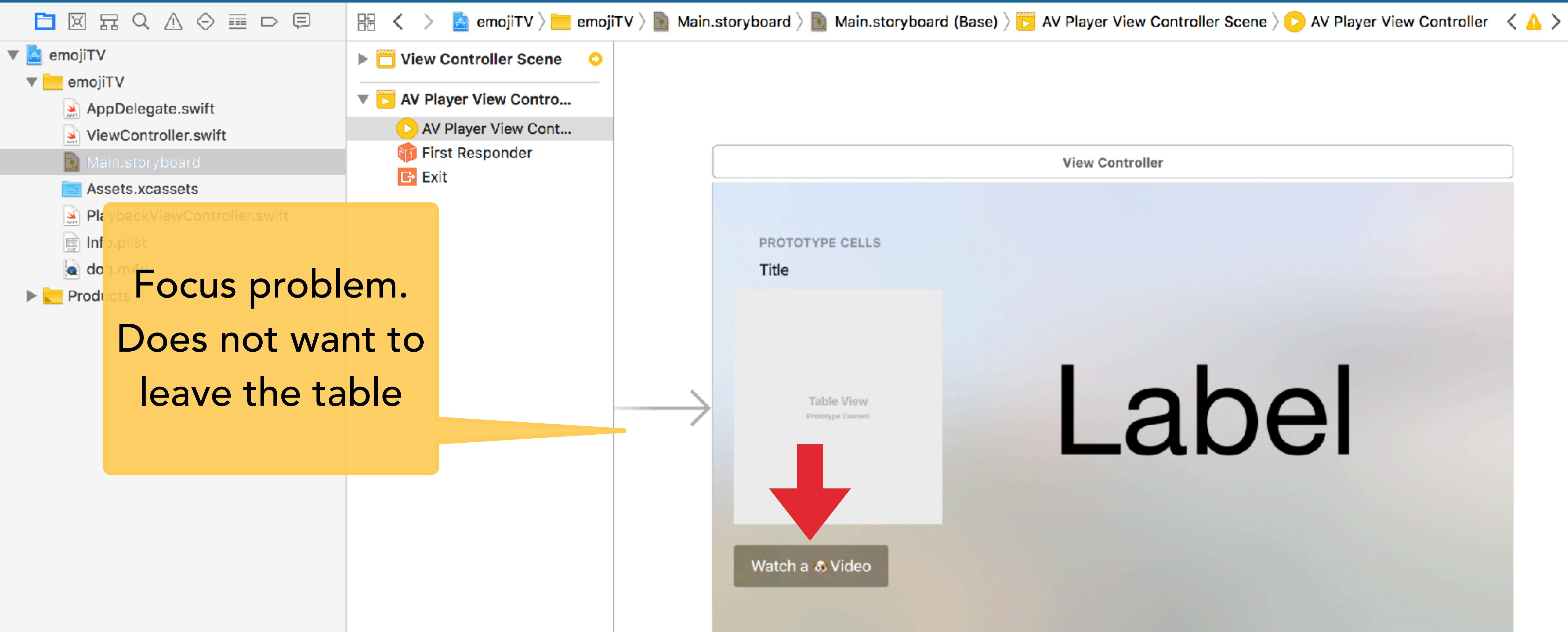
TVOS APPS

Focus triggers
update

```
extension ViewController: UITableViewDelegate {  
  
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell { ... }  
  
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) { ... }  
  
    // Update the big emoji label when we change focus on the table (no click required)  
    func tableView(_ tableView:  
                  didUpdateFocusIn context: UITableViewFocusUpdateContext,  
                  with coordinator: UIFocusAnimationCoordinator) {  
        // this gives you the indexPath of the focused cell  
        if let indexPath = context.nextFocusedIndexPath {  
            self.bigLabel?.text = data[indexPath.row]  
        }  
    }  
}
```

FOCUS

TVOS APPS



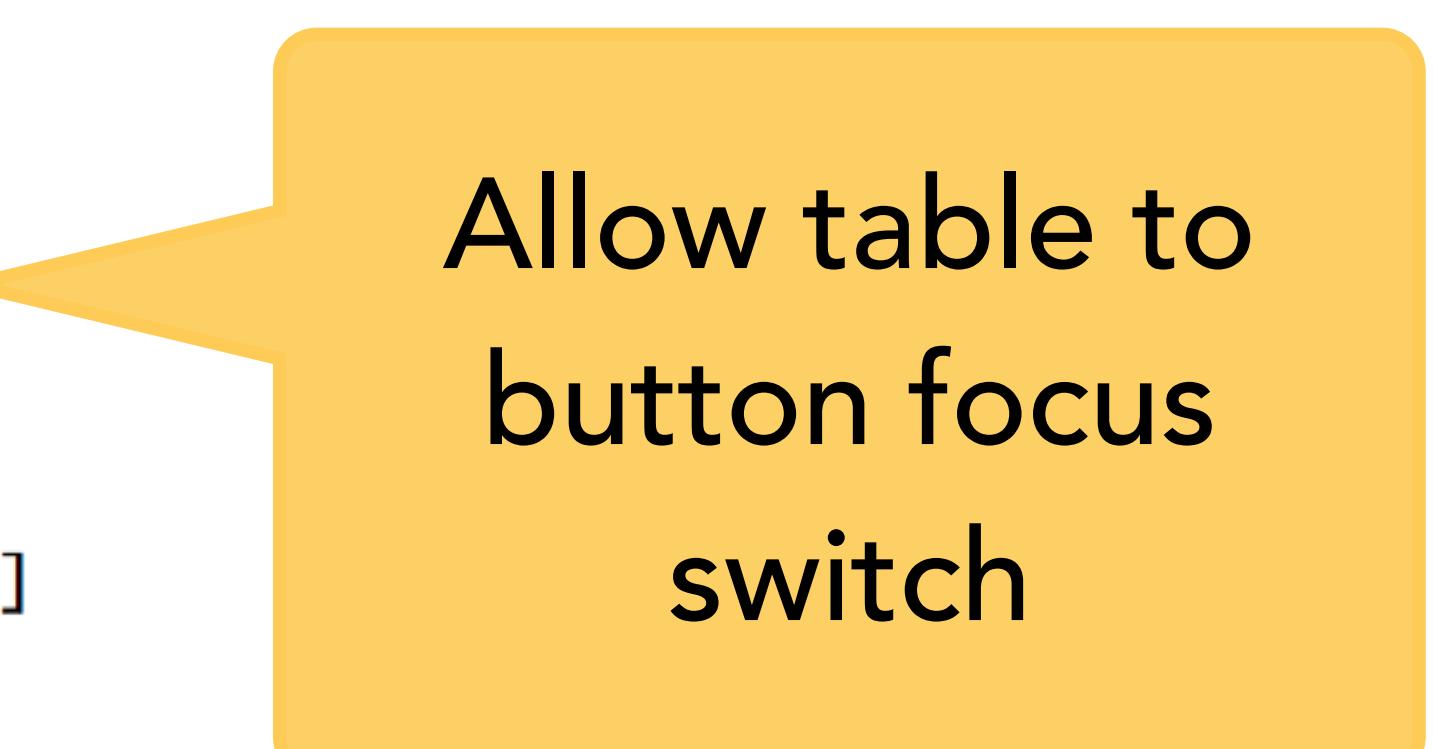
TVOS APPS

```
// MARK: - UIFocusEnvironment

override func didUpdateFocus(in context: UIFocusUpdateContext,
                           with coordinator: UIFocusAnimationCoordinator) {
    super.didUpdateFocus(in: context, with: coordinator)

    // Update the focus guide's `preferredFocusedView` depending on which
    // button has the focus
    guard let nextFocusedView = context.nextFocusedView else { return }

    switch nextFocusedView {
        case tableView:
            focusGuide.preferredFocusEnvironments = [button]
        case button:
            focusGuide.preferredFocusEnvironments = [tableView]
        default:
            focusGuide.preferredFocusEnvironments = []
    }
}
```



Allow table to
button focus
switch

TVOS APPS

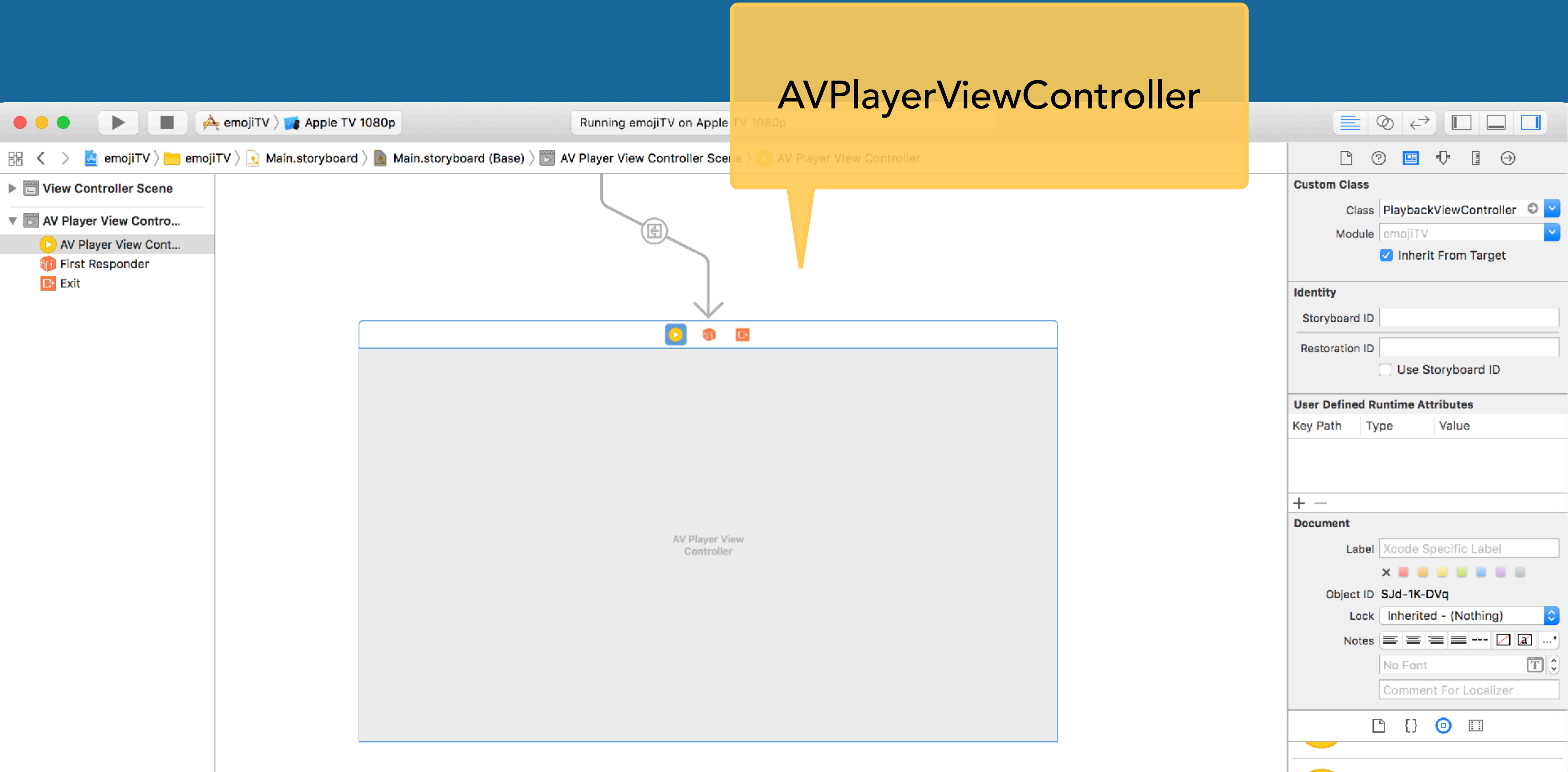
Set the table to have initial focus

```
/// MARK: - Focus
private var focusGuide: UIFocusGuide!

override var preferredFocusEnvironments : [UIFocusEnvironment] {
    return [tableView]
}
```

PLAYING VIDEO

PLAYING VIDEO



PLAYING VIDEO

- All-in-one for video playback
 - Airplay
 - Picture-in-Picture
 - Video controls
 - Chapter markers
 - Interstitial (ad) content
 - Overlays
 -and more...

AVPlayerViewController

An object that displays the video content from a player object along with system-supplied playback controls.

SDKs

iOS 8.0+

tvOS 9.0+

Framework

AVKit

On This Page

[Overview](#) ⓘ

[Topics](#) ⓘ

[Relationships](#) ⓘ

[See Also](#) ⓘ

Overview

Using `AVPlayerViewController` makes it easy for you to add media playback capabilities to your application matching the styling and features of the native system players. Since `AVPlayerViewController` is a system framework class, your playback applications automatically adopt the new aesthetics and features of future operating system updates without any additional work from you.

Important

Do not subclass `AVPlayerViewController`. Overriding this class's methods is unsupported and results in undefined behavior.

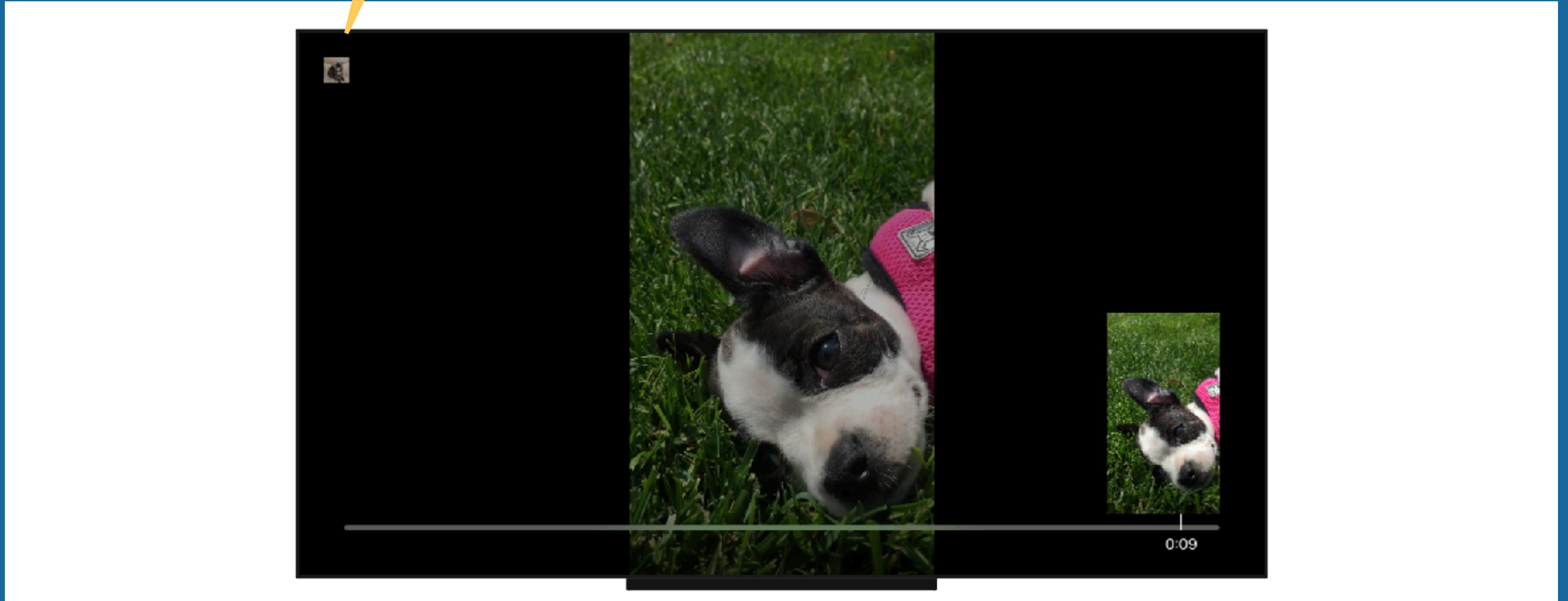
Supporting AirPlay

AirPlay lets users play your video content on an Apple TV and stream audio content to third-party AirPlay speakers and receivers. `AVPlayerViewController` automatically supports AirPlay, but you need to perform some project and audio session configuration before it can be enabled in your application. See [Configuring Audio Settings for iOS and tvOS](#) in [Media Playback Programming Guide](#) for more on how to perform this configuration.

See [AirPlay Overview](#) for more information about the features and capabilities of AirPlay.

TVOS APPS

Overlay view



- AVPlayerViewController

PLAYING VIDEO

```
< > emojiTV > emojiTV > PlaybackViewController.swift > M viewDidLoad()
// Copyright © 2017 T. Andrew Binkowski. All rights reserved.
//

import UIKit
import AVKit

class PlaybackViewController: AVPlayerViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        /* ...
        guard let path = Bundle.main.path(forResource: "dog", ofType:"m4v") else {
            return
        }

        // Set up the player
        self.player = AVPlayer(url: URL(fileURLWithPath: path))
        self.player?.play()
        NotificationCenter.default.addObserver(self,
                                               selector: #selector(PlaybackViewController.itemDidFinishPlaying(notification:)),
                                               name: NSNotification.Name.AVPlayerItemDidPlayToEndTime,
                                               object: player?.currentItem)
    }
}
```



Load up and play

PLAYING VIDEO

```
// Set up the player
self.player = AVPlayer(url: URL(fileURLWithPath: path))
self.player?.play()
NotificationCenter.default.addObserver(self,
                                         selector: #selector(PlaybackViewController.itemDidFinishPlaying(notification:)),
                                         name: NSNotification.Name.AVPlayerItemDidPlayToEndTime,
                                         object: player?.currentItem)

}

deinit {
    NotificationCenter.default.removeObserver(self)
}

/// Automatically dismiss the player when done
@objc func itemDidFinishPlaying(notification: NSNotification) {
    print ("Notification sent with done playing")
    self.dismiss(animated: true) {
        print("Dismissed player view controller")
    }
}
```

Notification to
dismiss viewer
when done



ADVANCED tvOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 9



ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 9

LOCALIZATION

LOCALIZATION

- 155 App Stores worldwide
 - Adding new ones all the time
- Your apps can be for sale globally with minimal effort
 - Toggle switch in iTunes Connect
- Making your app make sense in International App Stores is a completely different thing



LOCALIZATION

- Top 5 countries
 - China
 - US
 - Japan
 - UK
 - Australia



LOCALIZATION



Translation is not as
simple as using
Google Translate



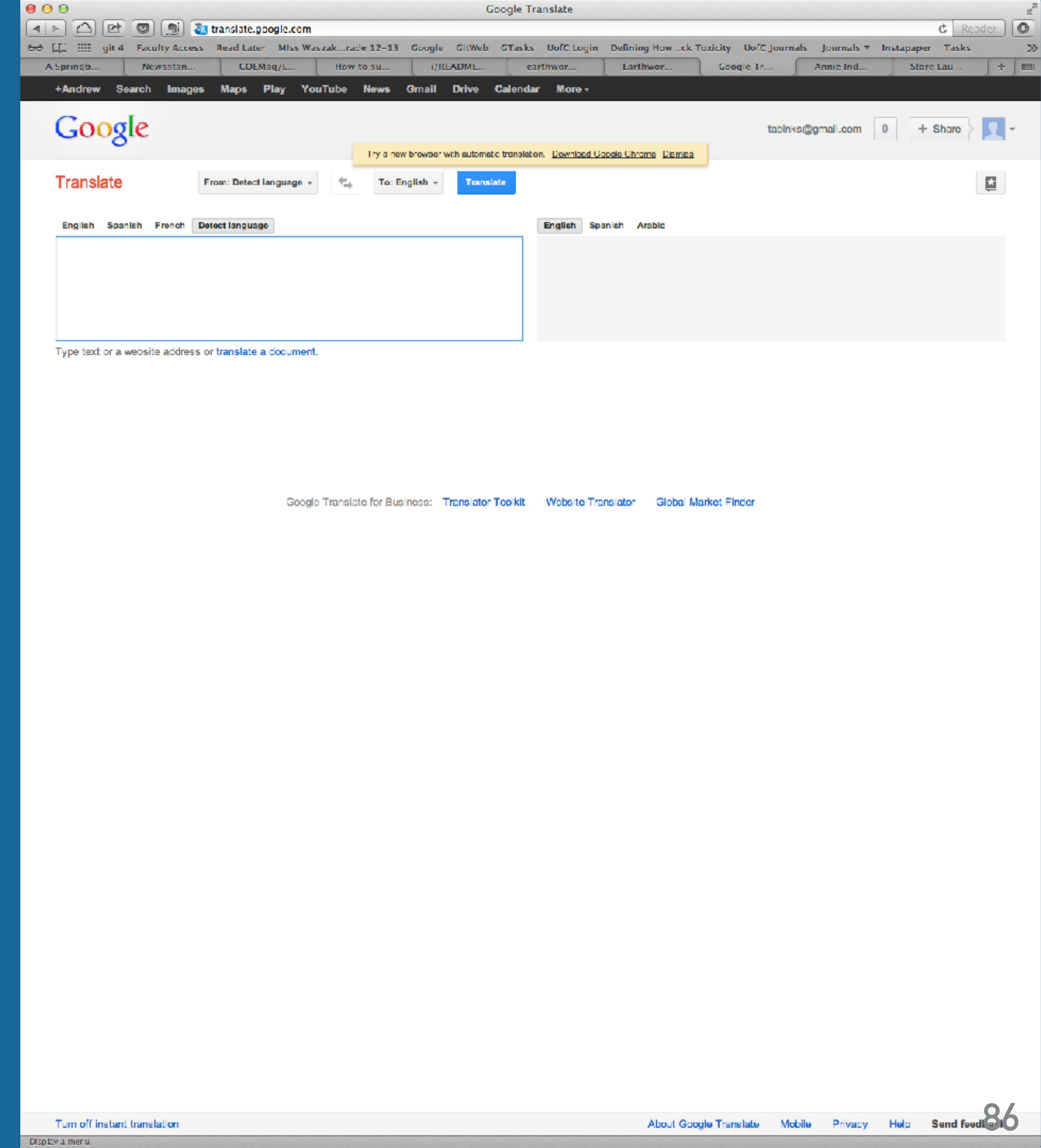
LOCALIZATION



Bad for in-app purchase

LOCALIZATION

- Apple makes localization straightforward through APIs and built-in Xcode support
 - Localize strings, numbers, images
- Managing the localization (and accuracy) can be tedious
 - Preparing assets for each language support



LOCALIZATION

- Google translation service (for Google Play only)
 - Service available directly to app developers in the Google Play Developer Console
 - Third-party translation services managed by Google Play
 - Pay per word



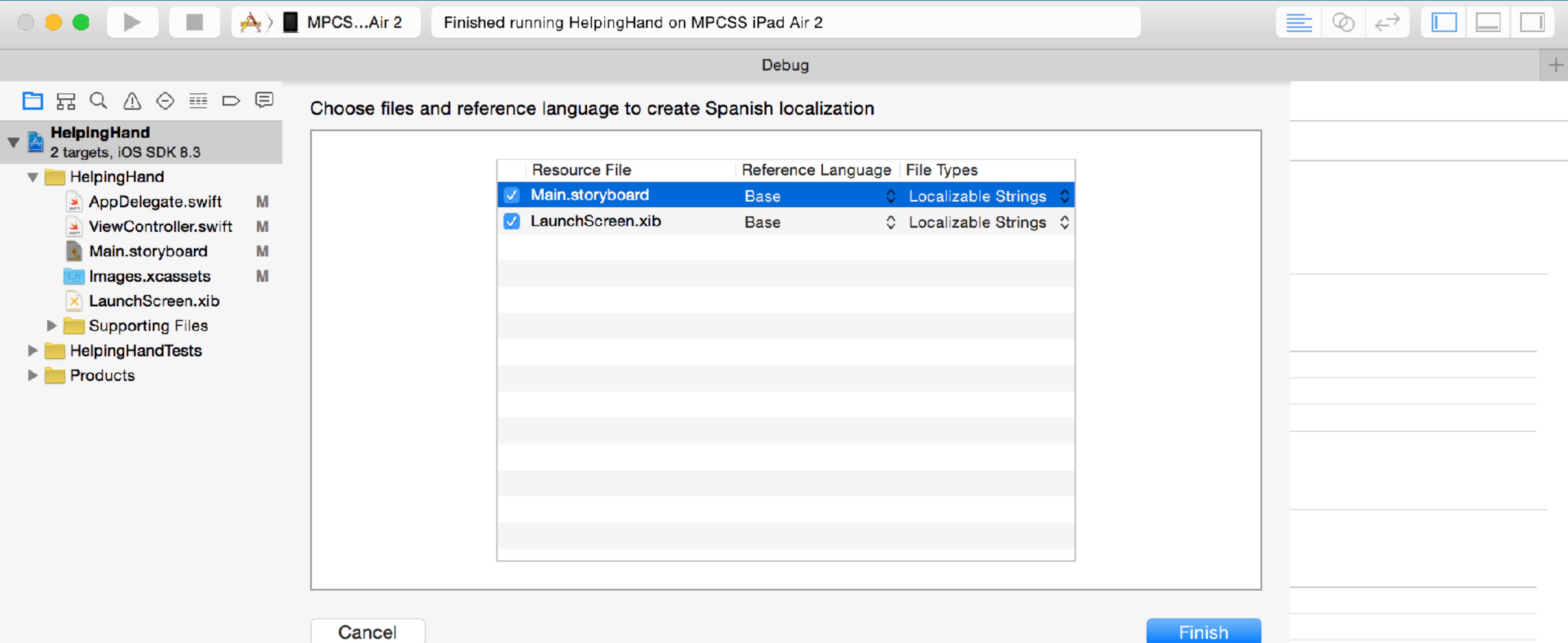
ADDING LOCALIZATION TO YOUR APP

LOCALIZATION

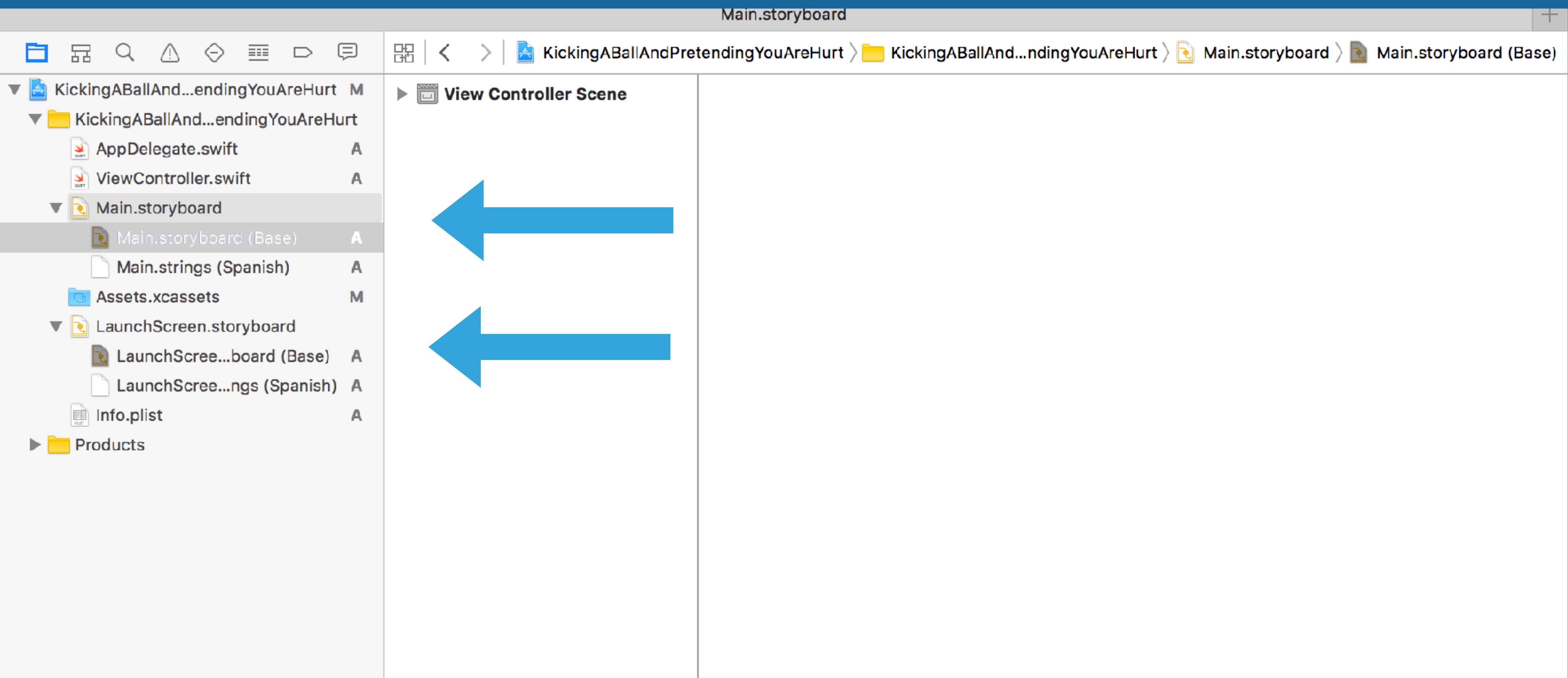
The screenshot shows the Xcode interface with the following details:

- Project Navigator (Left):** Shows the project structure for "HelpingHand". It includes files like AppDelegate.swift, ViewController.swift, Main.storyboard, Images.xcassets, LaunchScreen.xib, Supporting Files, HelpingsHandTests, and Products.
- Deployment Target (Top Right):** Set to iOS Deployment Target 8.3.
- Configurations (Middle Right):** Shows configurations for Debug and Release. Both are set to "Based on Configuration File" and "No Configurations Set".
- Localizations (Bottom Right):** Shows the "Localizations" section. It lists supported languages: French (fr), German (de), Chinese (Simplified) (zh-Hans), Chinese (Traditional) (zh-Hant), Japanese (ja), Spanish (es), Spanish (Mexico) (es-MX), Italian (it), Dutch (nl), Korean (ko), Portuguese (pt), Portuguese (Portugal) (pt-PT), Danish (da), Finnish (fi), and Norwegian Bokmål (nb).
 - A yellow arrow points from the "Localizations" section towards the "Use Base Internationalization" checkbox.
 - The "Use Base Internationalization" checkbox is checked (indicated by a blue checkmark).
 - The "Resources" column shows "2 Files Localized".

LOCALIZATION



LOCALIZATION



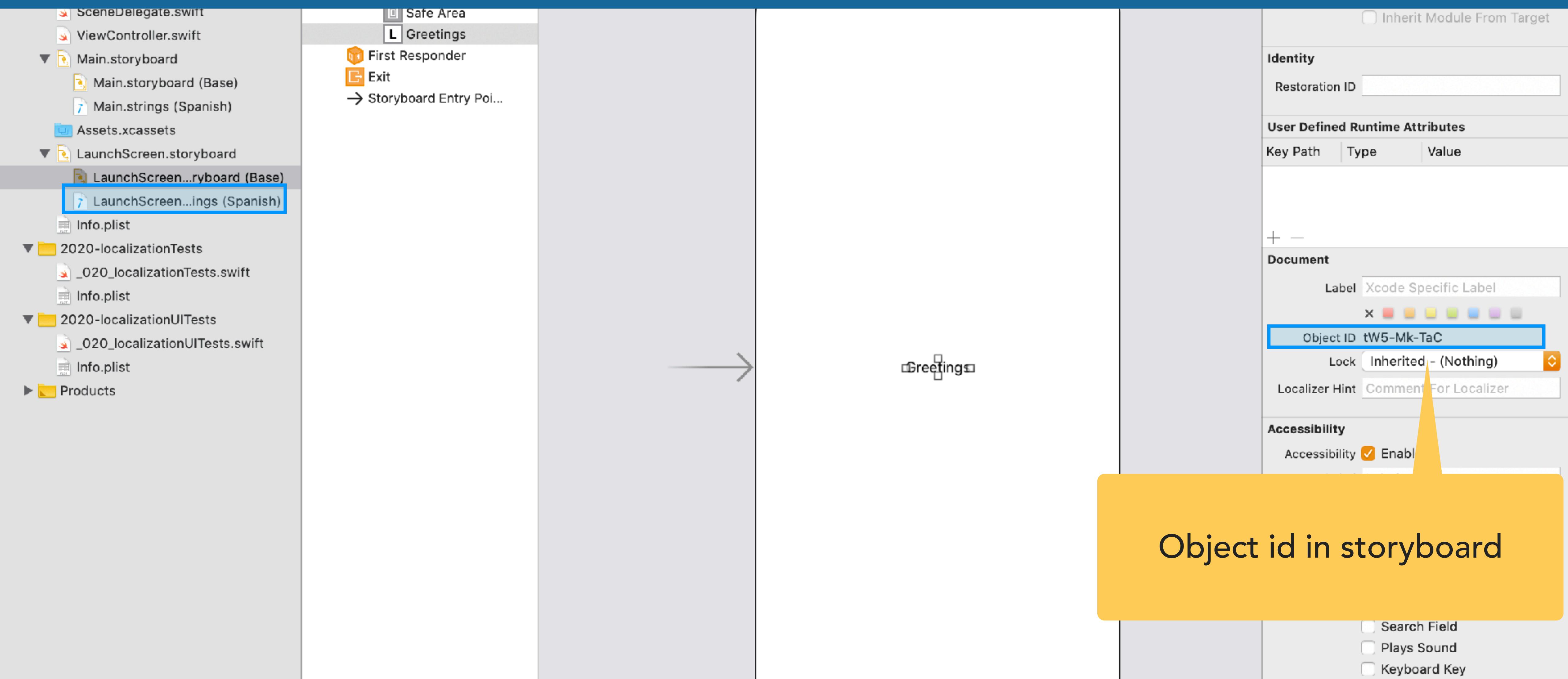
LOCALIZATION

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure with files like AppDelegate.swift, ViewController.swift, Main.storyboard, Assets.xcassets, LaunchScreen.storyboard, Info.plist, and Products.
- Main.storyboard:** The file is selected and highlighted with a blue border. A yellow callout box points to it with the text "Object id in storyboard".
- Editor:** Displays the storyboard code. Lines 2 and 3 show localization for a UILabel object:

```
/* Class = "UILabel"; text = "Label"; ObjectID = "JVV-sm-Em4"; */  
"JVV-sm-Em4.text" = "Spanish";
```
- Utilities:** The right panel shows the Localization editor for Main.strings.
 - Localization:** Shows checkboxes for Base (checked), English (unchecked), and Spanish (checked). Both English and Spanish are listed under "Localizable Strings".
 - Target Membership:** Shows the target KickingABallAndPretendingYouAreHurt selected.
 - Text Settings:** Includes options for Text Encoding (Default - Unicode (UTF-8)), Line Endings (Default - OS X / Unix (LF)), Indent Using (Spaces), Widths (Tab width 2, Indent width 2), and Wrap lines (checked).
 - Icons:** Shows icons for File, Group, Delete, and Edit.
 - Description:** A horizontal stack view icon with the text "Horizontal Stack View - Arranges views linearly."

LOCALIZATION



LOCALIZATION

The screenshot shows the Xcode interface with the following details:

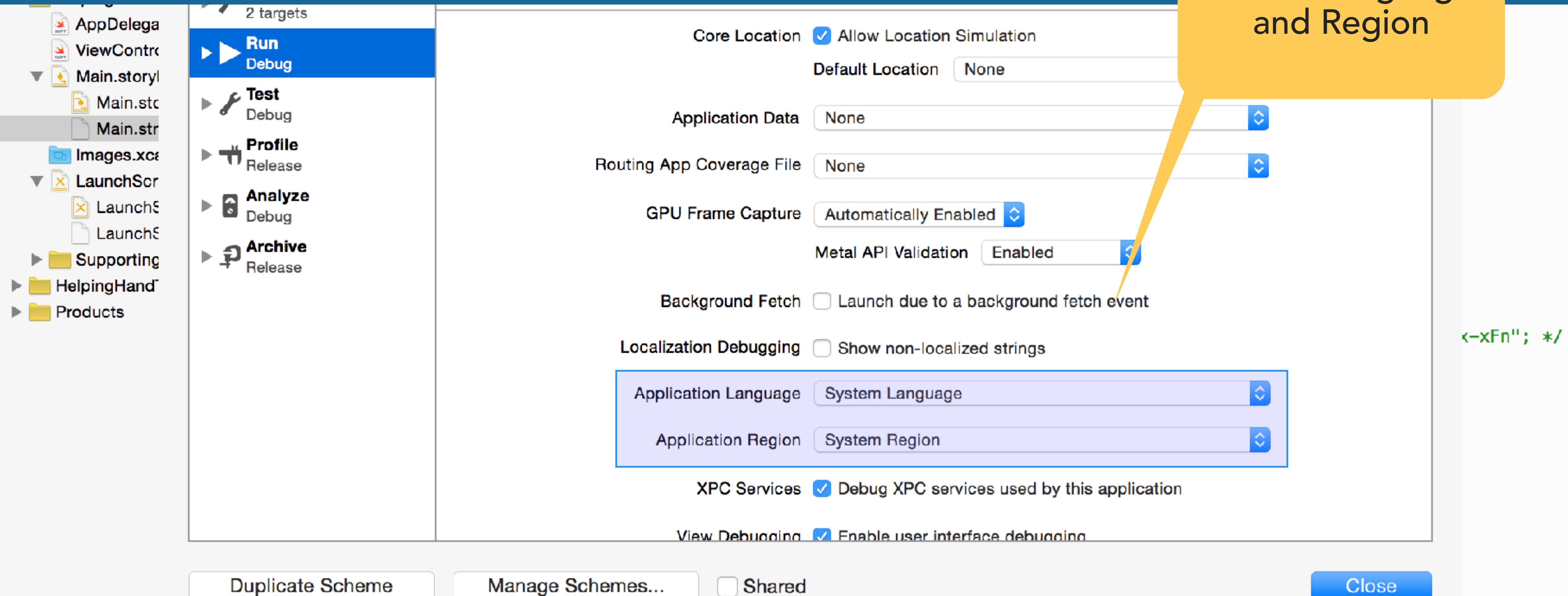
- Project Navigator:** Shows the project structure under "2020-localization".
 - 2020-localization:** Contains "2020-localization" (a folder), "AppDelegate.swift", "SceneDelegate.swift", "ViewController.swift", "Main.storyboard" (selected), "Assets.xcassets", and "LaunchScreen.storyboard".
 - LaunchScreen.storyboard:** Contains "LaunchScreen.storyboard (Base)" and "LaunchScreen.storyboard (Spanish)" (selected).
- File Navigator:** Shows the file path: 2020-localization > 2020-localization > LaunchSc...storyboard > LaunchScreen.strings (Spanish). The status bar indicates "No Selection".
- Editor:** Displays the contents of "LaunchScreen.strings (Spanish)".

```
1 "tw5-Mk-TaC" = "Your Localized String"
2
```

A yellow callout box with a black border and white text points to the first line of the string:

Object id in storyboard
- Toolbar:** Standard Xcode toolbar with icons for file operations.
- Top Bar:** Shows the project name "2020-localization", device "iPhone 11 Pro Max", and message "Finished running 2020-localization on iPhone SE (2nd generation)".

LOCALIZATION



```
35 /* Class = "UISlider"; accessibilityHint = "Satisfaction with accessibility implementation."; ObjectID =
36   "vqZ-1V-NaA"; */
37 "vqZ-1V-NaA.accessibilityHint" = "Satisfaction with accessibility implementation.;"
```

LOCALIZATION

The screenshot shows the Xcode interface for managing localization schemes. At the top, there are two targets: 'Run' and 'Debug'. Below them are settings for 'Core Location' and 'Allow Location Simulation'. A 'Default Location' dropdown is set to 'None'. On the left, a sidebar lists project files: AppDelegate.swift, ViewController.swift, Main.storyboard, Main.story, Main.storyboard, Images.xcassets, LaunchScreen.storyboard, LaunchScreen.storyboard, LaunchScreen.storyboard, Supporting, HelpHand, and Products.

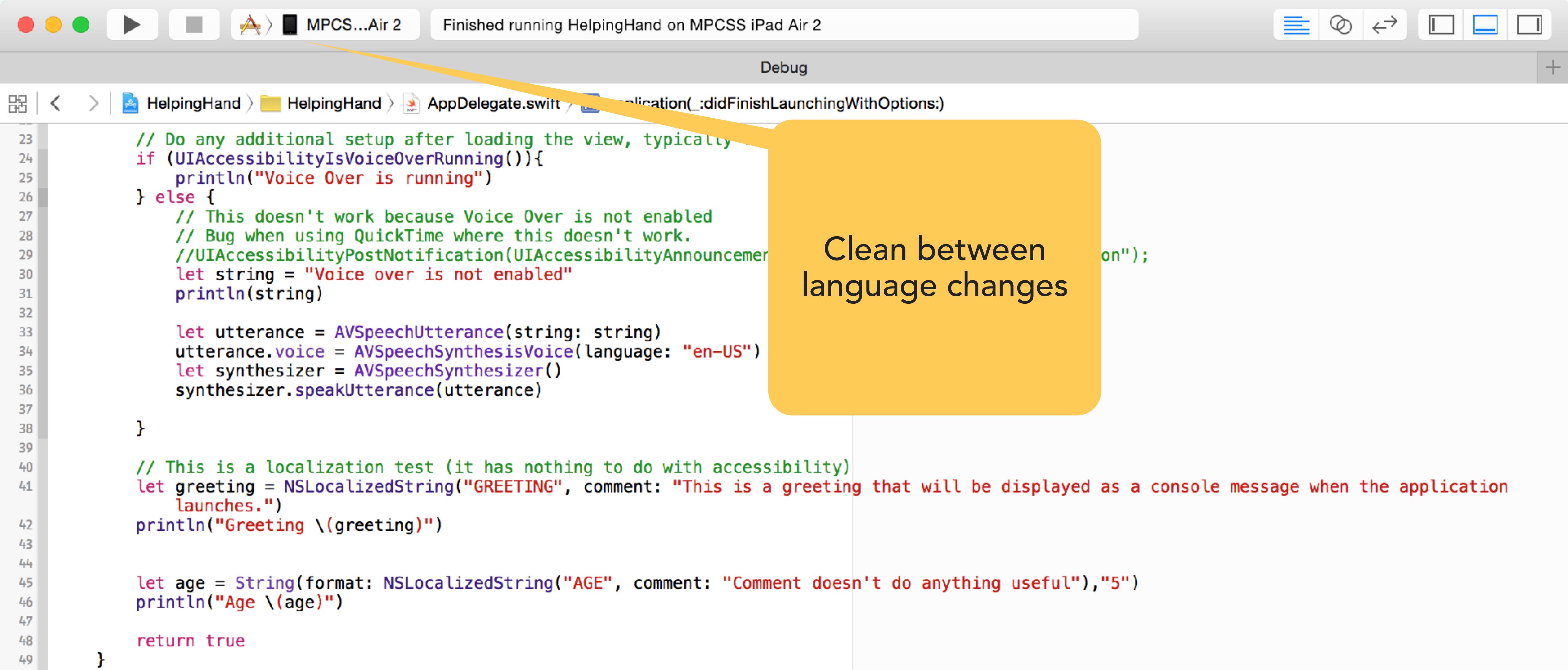
The main area displays two localization schemes:

- KickingABallAndPretendingYouAreHurt**: Indicated by a checkmark and a pencil icon.
- Run (Spanish)**: Indicated by a pencil icon.

At the bottom, there are three buttons: 'Edit Scheme...', 'New Scheme...', and 'Manage Schemes...'. A yellow callout bubble points to the 'New Scheme...' button with the text 'Add a new language specific scheme'.

```
35 /* Class = "UISlider"; accessibilityHint = "Satisfaction with accessibility implementation."; ObjectID =
      "vqZ-1V-NaA"; */
36 "vqZ-1V-NaA.accessibilityHint" = "Satisfaction with accessibility implementation.";
```

LOCALIZATION

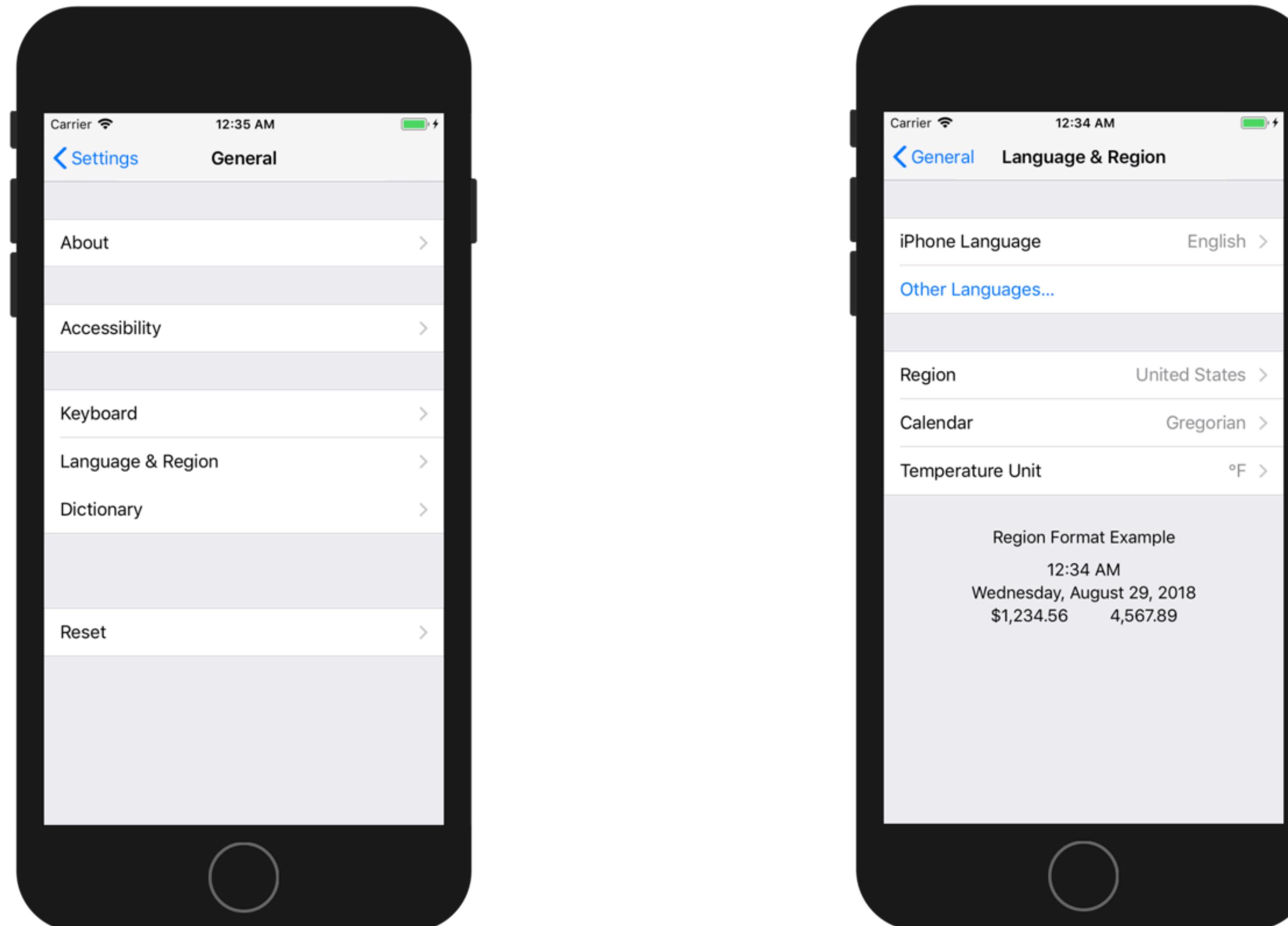


A screenshot of the Xcode IDE showing the AppDelegate.swift file. A yellow callout bubble points from the text "Clean between language changes" to the NSLocalizedString function at line 41.

```
23 // Do any additional setup after loading the view, typically
24 if (UIAccessibilityIsVoiceOverRunning()){
25     println("Voice Over is running")
26 } else {
27     // This doesn't work because Voice Over is not enabled
28     // Bug when using QuickTime where this doesn't work.
29     //UIAccessibilityPostNotification(UIAccessibilityAnnouncement, "on");
30     let string = "Voice over is not enabled"
31     println(string)
32
33     let utterance = AVSpeechUtterance(string: string)
34     utterance.voice = AVSpeechSynthesisVoice(language: "en-US")
35     let synthesizer = AVSpeechSynthesizer()
36     synthesizer.speakUtterance(utterance)
37 }
38
39 // This is a localization test (it has nothing to do with accessibility)
40 let greeting = NSLocalizedString("GREETING", comment: "This is a greeting that will be displayed as a console message when the application launches.")
41 println("Greeting \(greeting)")
42
43
44 let age = String(format: NSLocalizedString("AGE", comment: "Comment doesn't do anything useful"), "5")
45 println("Age \(age)")
46
47
48 return true
49 }
```

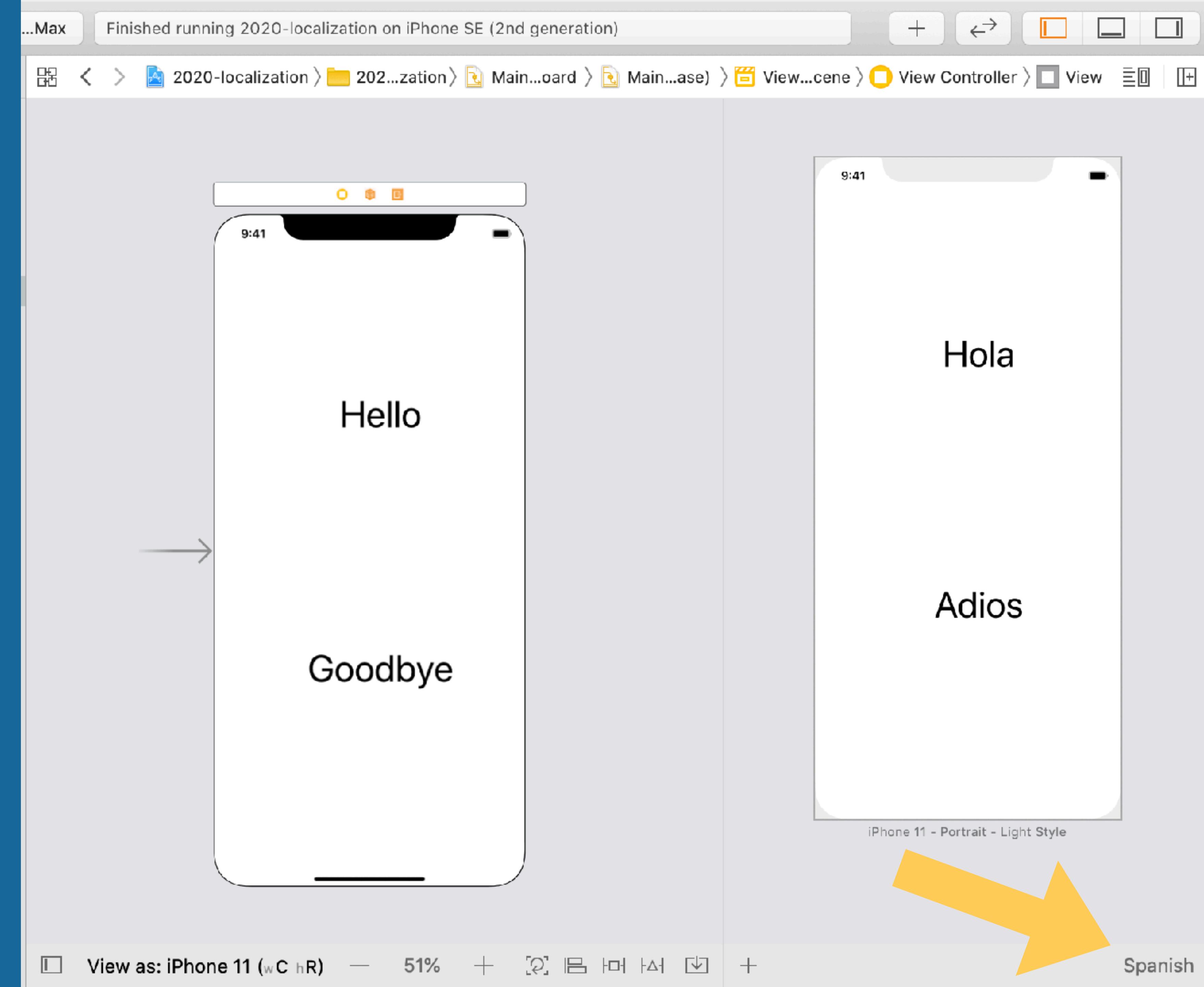
Clean between language changes

LOCALIZATION



LOCALIZATION

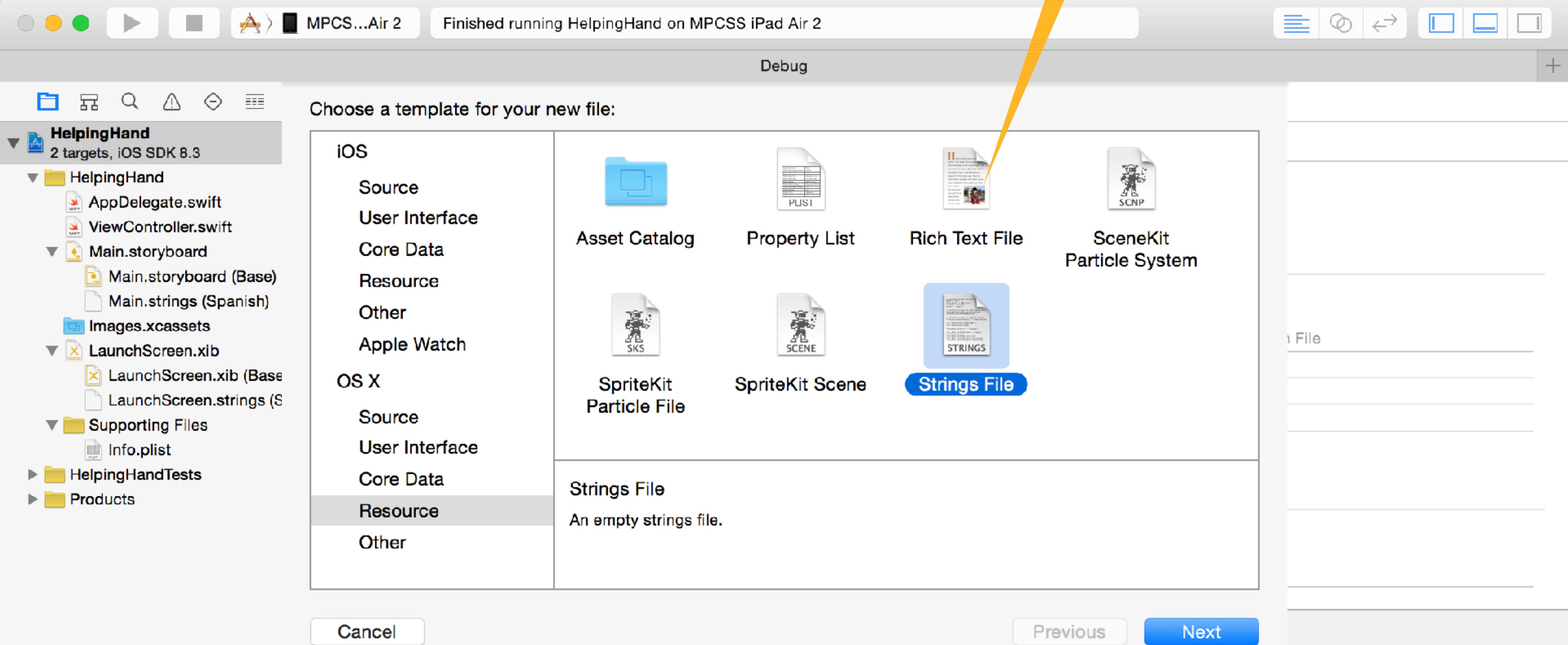
- Preview mode



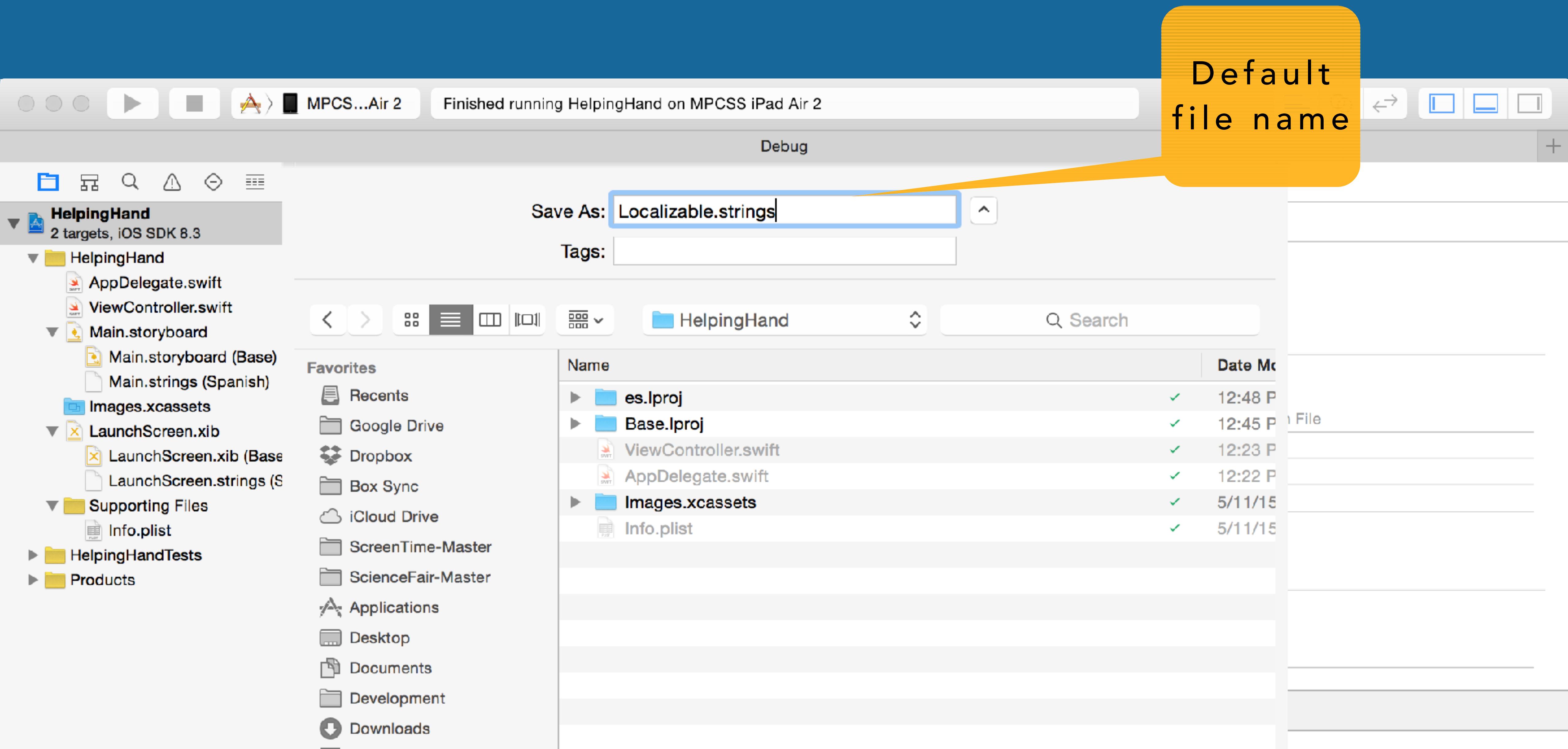
LOCALIZING TEXT STRINGS

LOCALIZATION

Add new file



LOCALIZATION



LOCALIZATION

The screenshot shows the Xcode interface with a project named "KickingABallAnd...endingYouAreHurt". The left sidebar displays the project structure, including files like AppDelegate.swift, ViewController.swift, Main.storyboard, Assets.xcassets, and Localizable.strings. A yellow callout bubble points from the "Localizable.strings" file in the project navigator towards the code editor. The code editor shows the contents of Localizable.strings:

```
/* Localizable.strings
KickingABallAndPretendingYouAreHurt
Created by T. Andrew Binkowski on 5/16/16.
Copyright © 2016 The University of Chicago, Department of Computer Science. All rights reserved.
*/
```

A large yellow arrow points from the code editor towards the right-hand utilities pane. The utilities pane contains several sections:

- MPC51032-2016-Spring-Playgrounds/**: A list of files including KickingABallAndPretendingYouAreHurt/, KickingABallAndPretendingYouAreHurt/, and Localizable.strings.
- On Demand Resource Tags**: A table with a single row labeled "Tags".
- Localization**: A section with a "Localize..." button.
- Target Membership**: A checkbox for "KickingABallAndPretendingYouAreHurt" which is checked.
- Text Settings**: Options for Text Encoding (Unicode (UTF-8)), Line Endings (Default - OS X / Unix (LF)), Indent Using (Spaces), and Widths (Tab and Indent).
- Label**: A description of the Label control: "Label - A variably sized amount of static text."
- Button**: A description of the Button control: "Button - Intercepts touch events and...".

LOCALIZATION

The screenshot shows a localization dialog in Xcode. On the left, the project navigator displays a file structure for a project named "HelpingHand". The "Main.storyboard" file is selected, showing its base file ("Main.storyboard (Base)") and a Spanish localization file ("Main.strings (Spanish)"). The storyboard icon features a blue blueprint and a hammer.

The main area of the dialog contains the following text:

Do you want to localize this file?
The file will be moved into the lproj folder for the following language.

A dropdown menu shows "Spanish" as the selected language. Below the dropdown are two buttons: "Cancel" and "Localize".

The right side of the screen shows the Xcode interface with various panels:

- Type**: Default - Localizable Strings
- Group**: I/Localizable.strings
- Path**: /Localizable.strings
owksi/Google Drive/g-Teaching/-2015-Spring/MyDemos/
I/HelpingHand/Localizable.strings
- Localization**: Localize...
- Text Encoding**: UTF-8
- Line Endings**: OS X / Unix (LF)
- Formatting**: Indent Using: Spaces, Widths: Tab: 4, Indent: 4, Wrap lines: checked
- Source Control**: Repository: HelpingHand, Type: Git, Current Branch: master

LOCALIZATION

The screenshot shows the Xcode interface with a focus on localization. On the left, the Project Navigator displays the project structure:

- KickingABallAndPretendingYouAreHurt**:
 - KickingABallAndPretendingYouAreHurt**:
 - AppDelegate.swift
 - ViewController.swift
 - Main.storyboard**:
 - Main.storyboard (Base)
 - Main.strings (Spanish)
 - Assets.xcassets**
 - Localizable.strings**:
 - Localizable.strings (Spanish)
 - Localizable.strings (Base)
 - LaunchScreen.storyboard**:
 - LaunchScreen.storyboard (Base)
 - LaunchScreen.strings (Spanish)
 - Info.plist
- Products**

The **Localizable.strings** file is selected and highlighted with a blue border. The code editor on the right shows the contents of the file:

```
/* Localizable.strings
KickingABallAndPretendingYouAreHurt
Created by T. Andrew Binkowski on 5/16/16.
Copyright © 2016 The University of Chicago, Department of Computer Science. All rights reserved.
*/
```

The right side of the screen contains the **File Inspector**, which provides details about the selected file:

- Development/GitHub/MPCS51032-2016-Spring-Playgrounds/KickingABallAndPretendingYouAreHurt/KickingABallAndPretendingYouAreHurt/Base.lproj/Localizable.strings**
- On Demand Resource Tags**: Tags
- Localization**:
 - Base (checked)
 - English
 - Spanish (checked)
- Target Membership**: KickingABallAndPretendingYouAreHurt
- Text Settings**:
 - Text Encoding: Default - Unicode (UTF-8)
 - Line Endings: Default - OS X / Unix (LF)
 - Indent Using: Spaces
- Label**: Label - A variably sized amount of static text.
- Button**: Button - Intercepts touch events and

LOCALIZATION

The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations.
- WindowTitle:** "Running HelpingHand on MPCSS iPad Air 2".
- DocumentTitle:** "Debug" (selected tab).
- File Navigator:** Shows the project structure:
 - HelpingHand**: 2 targets, iOS SDK 8.3.
 - Localizable.strings**:
 - Localizable.strings (Spanish)** (selected, highlighted in grey).
 - Localizable.strings (English)**.
 - HelpingHand**:
 - AppDelegate.swift**
 - ViewController.swift**
 - Main.storyboard**:
 - Main.storyboard (Base)**
 - Main.strings (Spanish)**
 - Images.xcassets**
 - LaunchScreen.xib**:
 - LaunchScreen.xib (Base)**
 - LaunchScreen.strings (Spanish)** (highlighted in grey).
 - Supporting Files**:
 - Info.plist**
 - HelpingHandTests**
 - Products**
- Editor Area:** Displays the content of the selected Localizable.strings (Spanish) file:

```
/*  
 * Localizable.strings  
 * HelpingHand  
 *  
 * Created by T. Andrew Binkowski on 6/1/15.  
 * Copyright (c) 2015 The University of Chicago. All rights reserved.  
 */  
GREETING="Hola (Spanish)";
```

LOCALIZATION

The screenshot shows the Xcode interface with two main panes. The left pane is a file browser displaying the project structure:

- Localizable.strings (Spanish) A
- Localizable.strings (English) A
- HelpingHand folder:
 - AppDelegate.swift M
 - ViewController.swift M
- Main.storyboard folder:
 - Main.storyboard (Base) M
 - Main.strings (Spanish) A
- Images.xcassets M
- LaunchScreen.xib folder:
 - LaunchScreen.xib (Base)
 - LaunchScreen.strings (Spanish) A
- Supporting Files folder:
 - Info.plist
- HelpingHandTests folder
- Products folder

The right pane shows a text editor with the following content:

```
4
5 Created by T. Andrew Binkowski on 6/1/15.
6 Copyright (c) 2015 The University of Chicago. All rights reserved.
7 */
8
9 GREETING="Hello (English)";
10
```

LOCALIZATION

```
// This is a localization test (it has nothing to do with accessibility)
let greeting = NSLocalizedString("GREETING", comment: "This is a greeting.")
print("Greeting \(greeting)")
```

```
/*
 Localizable.strings
 KickingABallAndPretendingYouAreHurt

 Created by T. Andrew Binkowski on 5/16/16.
 Copyright © 2016 The University of Chicago, Department of
 Computer Science. All rights reserved.
 */

GREETING = "Hello";
```

```
/*
 Localizable.strings
 KickingABallAndPretendingYouAreHurt

 Created by T. Andrew Binkowski on 5/16/16.
 Copyright © 2016 The University of Chicago, Department o
 reserved.
 */

GREETING = "Hola";
```

LOCALIZATION

```
#define NSLocalizedString(key, comment)
[ [NSBundle mainBundle] localizedStringForKey:(key) value:@""  
    table:nil]
```

- Uses the `localizedStringForKey:` method to look up the string for a given key
 - nil is the default table name and uses “Localized.strings”

`localizedStringForKey:value:table:`

Returns a localized version of the string designated by the specified key and residing in the specified table.

- `(NSString *)localizedStringForKey:(NSString *)key value:(NSString *)value table:(NSString *)tableName`

Parameters

`key`

The key for a string in the table identified by `tableName`.

`value`

The value to return if `key` is `nil` or if a localized string for `key` can't be found in the table.

`tableName`

The receiver's string table to search. If `tableName` is `nil` or is an empty string, the method attempts to use the table in `Localizable.strings`.

LOCALIZATION

```
// This is a localization test (it has nothing to do with accessibility)
let greeting = NSLocalizedString("GREETING", comment: "This is a greeting.")
print("Greeting \(greeting)")
```

- Comment is used for “genstrings” tool
- Appears next to default string as aid for translator
 - Useful for large projects to separate the code from the strings

LOCALIZATION

genstrings(1)

BSD General Commands Manual

genstrings(1)

NAME

genstrings -- generate string table from source code

SYNOPSIS

```
genstrings [-a] [-s <routine>] [-skipTable <Table>] [-noPositionalParameters]
           [-bigEndian | -littleEndian] [-u] [-macRoman] [-q] [-o <outputDir>] file
```

...

DESCRIPTION

The **genstrings** utility generates a `.strings` file(s) from the C or Objective-C (`.c` or `.m`) source code file(s) given as the argument(s). A `.strings` file is used for localizing an application for different languages, as described under "Internationalization" in the Cocoa Developer Documentation.

* C and Objective-C:

Source lines containing text of the form `NSLocalizedString("key", comment)` or `CFCopyLocalizedString("key", comment)` will generate an appropriate string table entry to a file named `Localizable.strings`.

Source lines containing `NSLocalizedStringFromTable("key", Table, comment)` or `CFCopyLocalizedStringFromTable("key", Table, comment)` will generate an appropriate string table entry in a file named `Table.strings`.

Source lines with `NSLocalizedStringFromTableInBundle("key", Table, bundle, comment)` or `CFCopyLocalizedStringFromTableInBundle("key", Table, bundle, comment)` will generate an appropriate string table entry

LOCALIZATION

```
26 } else {
27     // This doesn't work because Voice Over
28     // Bug when using QuickTime where the
29     //UIAccessibilityPostNotification(UIAccessibilityPostNotification(UI
30 let string = "Voice over is not enabled"
31 println(string)
32
33 let utterance = AVSpeechUtterance(string)
34 utterance.voice = AVSpeechSynthesisVoice.defaultVoice()
35 let synthesizer = AVSpeechSynthesizer()
36 synthesizer.speakUtterance(utterance)
37
38 }
39
40 // This is a localization test (it has nothing to do with accessibility)
41 let greeting = NSLocalizedString("GREETING", comment: "This is a greeting that will be displayed as a console message when the application launches.")
42 println("Greeting \(greeting)")
43
44 let age = String(format: NSLocalizedString("AGE", comment: "Comment doesn't do anything useful"), "5")
45 println("Age \(age)")
46
47
48 return true
49
50
51 func applicationWillResignActive(application: UIApplication) {
52     // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as a phone call or SMS) or when the user quits the application and it begins the transition to the background state.
53 }
```

Parameters

```
1  /*
2   * Localizable.strings
3   * HelpingHand
4   *
5   * Created by T. Andrew Binkowski on 6/1/15.
6   * Copyright (c) 2015 The University of Chicago. All rights reserved.
7   */
8
9 GREETING="Hello (English)";
10 AGE="I am %d years old";
11 |
```

Parameters

Voice over is not enabled
Greeting Hello (English)
Age I am 5 years old

LOCALIZING ASSETS

LOCALIZING ASSETS



<https://www.youtube.com/watch?v=xSCOjXwXzX0>

LOCALIZING ASSETS

The screenshot shows the Xcode project navigator on the left, displaying files like `KickingABall...ngYouAreHurt`, `KickingABall...urt.xcodeproj`, `AppDelegate.swift`, `Assets.xcassets`, `Base.lproj`, `en.lproj`, `es.lproj`, `Info.plist`, and `ViewController.swift`. The `es.lproj` folder is selected. In the center, the `Assets.xcassets` library is open, showing a folder structure for the `Football.png` asset. The `Football.png` file is selected, highlighted with a blue bar. To the right, the file's details are shown in a sidebar:

Football.png

PNG image - 85...

Created Today, 12:39 PM
Modified Today, 12:39 PM
Last opened Today, 12:39 PM
Dimensions 400 × 300
[Add Tags...](#)

This only applies to images stored directly in main bundle. There is no way to do this for XCAsset catalog.

A yellow callout bubble points from the text in the bottom-left towards the `Football.png` file in the assets library.

LOCALIZING ASSETS

The screenshot shows the Xcode interface with a project named "Local". On the left, the Project Navigator displays files like AppDelegate.h, AppDelegate.m, MainStoryboard.storyboard, Localizable.strings, and Supporting Files (including football1.jpg). In the center, a modal dialog titled "Do you want to localize this file?" asks if the file will be moved into the lproj folder for the English language. The "Localize" button is highlighted. On the right, the Utilities panel shows the properties for "football1.jpg", including dimensions (500 x 500 pixels), resolution (72 pixels/inch), color space (RGB), and alpha channel (No). A yellow callout bubble points to the "Localize" button in the modal, with the word "Localize" written inside it.

Local
1 target, iOS SDK 6.1

Local

- AppDelegate.h
- AppDelegate.m
- MainStoryboard.storyboard
 - MainStor...(English) M
 - MainStor...(Spanish) A
- Localizable.strings
 - Localizab...(English) A
 - Localizab...(Spanish) A
- ViewController.h
- ViewController.m

Supporting Files

- football1.jpg A
- Local-Info.plist
- InfoPlist.strings
 - InfoPlist....s (English)
 - InfoPlist...panish) A
- main.m
- Local-Prefix.pch
- Default.png
- Default@2x.png
- Default-568h@2x.png

Frameworks

Products

Do you want to localize this file?

The file will be moved into the lproj folder for the following language.

English

Cancel Localize

football1.jpg

Full Path /Users/tbinkowski/Dropbox/Teaching/2013-Spring/Demonstrations/Local/Local/football1.jpg

Image Properties

- Dimensions 500 x 500 pixels
- Resolution 72 pixels/inch
- Color Space RGB
- Alpha Channel No

Localization

Localize...

Target Membership

Local

Source Control

Version: Not yet committed

Status A Added

Media Library

Default-568h@2x.png

LOCALIZING ASSETS

The screenshot shows the Xcode interface with the Project Navigator on the left, the Preview pane in the center displaying a brown Nike football, and the Utilities panel on the right.

Project Navigator:

- AppDelegate.swift
- ViewController.swift
- Main.storyboard
- Assets.xcassets
- Football.png** (selected)
- Localizable.strings
- LaunchScreen.storyboard
- Info.plist
- Products

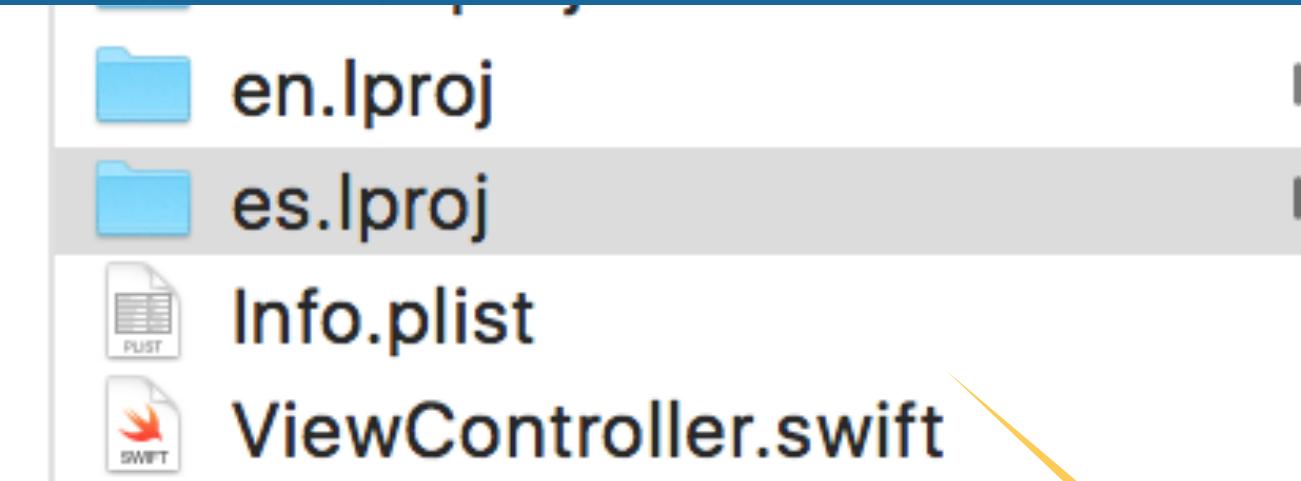
A yellow callout bubble with the word "Localize" points to the "Football.png" entry in the Project Navigator.

Utilities Panel (Preview View):

- Type: Default - PNG Image
- Location: Relative to Group
- Full Path: /Users/tbinkowski/Development/GitHub/MPCS51032-2016-Spring-Playgrounds/KickingABallAndPretendingYouAreHurt/Base.ipproj/Football.png
- On Demand Resource Tags: Tags
- Image Properties:
 - Dimensions: 400 × 300 pixels
 - Resolution: 72 pixels/inch
 - Color Space: RGB
 - Alpha Channel: Yes
- Localization:
 - Base
 - English
 - Spanish
 - KickingABallAndPretendingYouAreHurt
- Source Control:
 - Repository: MPCS51032-2016-Spring-Playgrounds
 - Type: Git
 - Current Branch: master

A yellow callout bubble with the word "Localize" points to the localization checkboxes in the Utilities panel.

LOCALIZING ASSETS



Main.strings



Directory structure for
localized application

Football.png

PNG image - 85...

Created Today, 12:39 PM

Modified Today, 12:39 PM

Last opened Today, 12:39 PM

Dimensions 100 × 200

LOCALIZING ASSETS



A yellow speech bubble points from the text "Football?" to a soccer ball.

The Xcode interface shows the project structure and asset details:

- Project Navigator:** AppDelegate.swift, ViewController.swift, Main.storyboard, Assets.xcassets, Football.png (selected), Localizable.strings, LaunchScreen.storyboard, Info.plist, Products.
- Assets Inspector (for Football.png):**
 - Type: Default - PNG Image
 - Location: Relative to Group es.iproj/Football.png
 - Full Path: /Users/tbinkowski/Development/GitHub/MPCS51032-2016-Spring-Playgrounds/KickingABallAndPretendingYouAreHurt/es.iproj/Football.png
 - On Demand Resource Tags: Tags
 - Image Properties: Dimensions 445 x 355 pixels, Resolution 72 pixels/inch, Color Space RGB, Alpha Channel Yes
 - Localization: Base (checked), English, Spanish (checked)
 - Target Membership: KickingABallAndPretendingYouAreHurt (checked)
 - Source Control: Repository MPC51032-2016-Spring-Playgrounds, Type Git, Current Branch master

LOCALIZING ASSETS

- Alternative ways
 - Load all images into a XCAsset catalog with unique names
 - Football-en
 - Footabll-es
 - At runtime extract the regionalized name
 - func imageNameWithRegion(name: String) ->String

Or Apple will improve support
in XCAsset Catalog

LOCALIZING APP NAME

LOCALIZING APP NAME

The screenshot shows the Xcode interface with the project navigation bar on the left and the main editor area on the right.

Project Navigator:

- 2 targets, iOS SDK 8.3
- Localizable.strings
 - Localizable.strings (Spanish) A
 - Localizable.strings (English) A
- HelpingHand
 - AppDelegate.swift M
 - ViewController.swift M
- Main.storyboard
 - Main.storyboard (Base) M
 - Main.strings (Spanish) A
- Images.xcassets M
- LaunchScreen.xib
 - LaunchScreen.xib (Base)
 - LaunchScreen.strings (Spanish) A
- Supporting Files
 - Info.plist
- HelpingHandTests
- Products

Editor Area (Main View):

Information Property List (Dictionary, 15 items):

- Localization native development r... String en
- Executable file String \$(EXECUTABLE_NAME)
- Bundle identifier String mobi.uchicago.\$(PRODUCT_NAME:rfc1034identifier)
- InfoDictionary version String 6.0
- Bundle name** String \$(PRODUCT_NAME) (Selected)
- Bundle OS Type code String APPL
- Bundle versions string, short String 1.0
- Bundle creator OS Type code String ????
- Bundle version String 1
- Application requires iPhone envir... Boolean YES
- Launch screen interface file base... String LaunchScreen
- Main storyboard file base name String Main
- Required device capabilities Array (1 item)
- Supported interface orientations Array (3 items)
- Supported interface orientations (...) Array (4 items)

File Inspector (Right Panel):

- Name: Info.plist
- Type: Default - Property List XML
- Location: Relative to Group
- Full Path: /Users/tbinkowski/Google Drive/g-Teaching/MPCS51032-2015-Spring/MyDemos/HelpingHand/HelpingHand/Info.plist
- Localization: Localize...
- Target Membership:
 - HelpingHand
 - HelpingHandTests
- Source Control:
 - Repository: HelpingHand
 - Type: Git
 - Current Branch: master
 - Version: --
 - Status: No changes
 - Location: /Users/tbinkowski/Google Drive/g-Teaching/MPCS51032-2015-Spring/MyDemos/HelpingHand/HelpingHand/Info.plist

Callout: A yellow callout points from the text "Localize .plist" to the "Localize..." button in the File Inspector.

Localize .plist

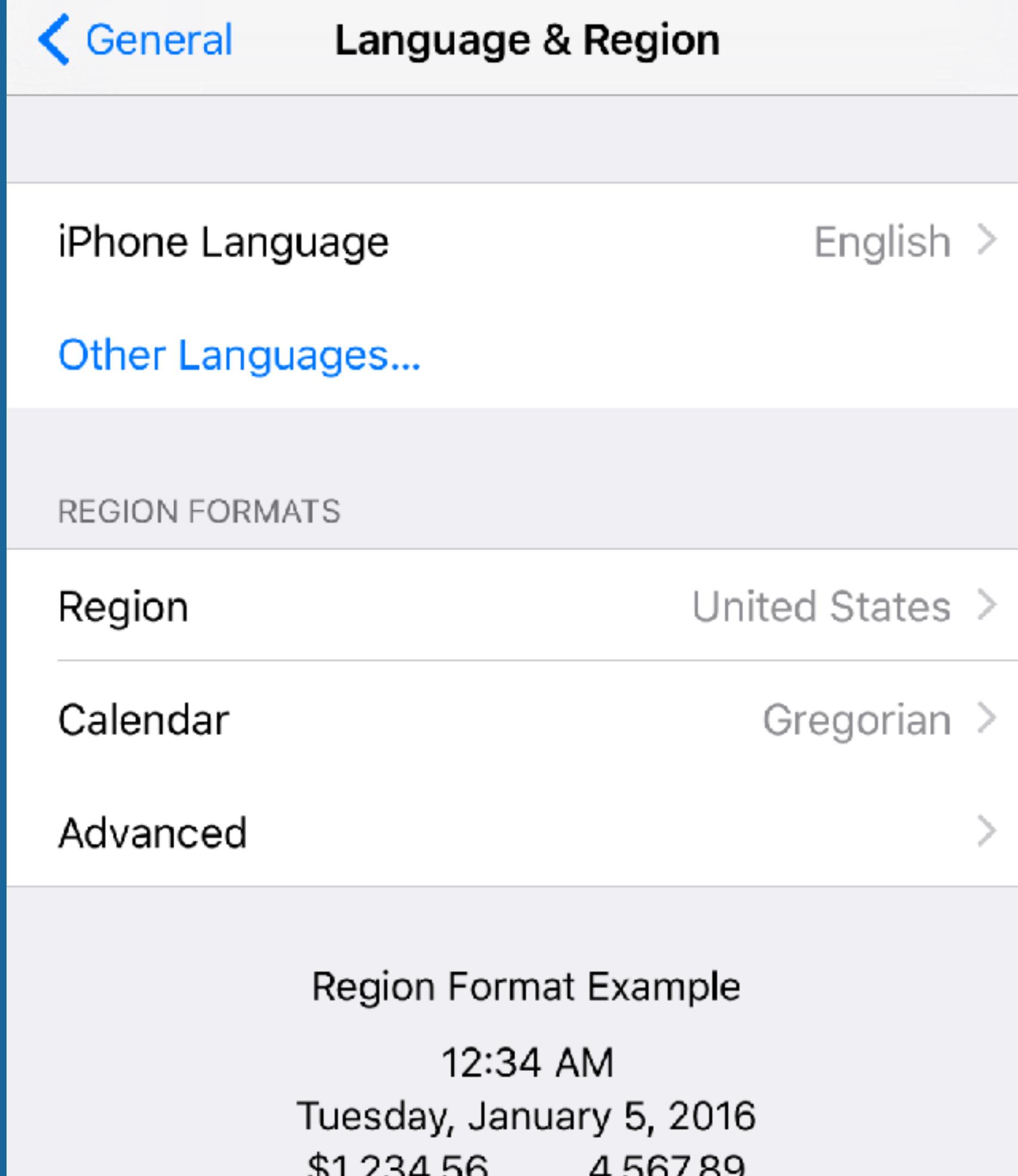
LOCALIZING APP NAME

- There are many different opportunities to localize by tweaking values in the .plist
 - Different launch options
 - Different servers to hit
 - Different Storyboard files to load
 - ...
- You can put any values in .plist and read them at runtime

**REGIONALIZE YOUR
APP**

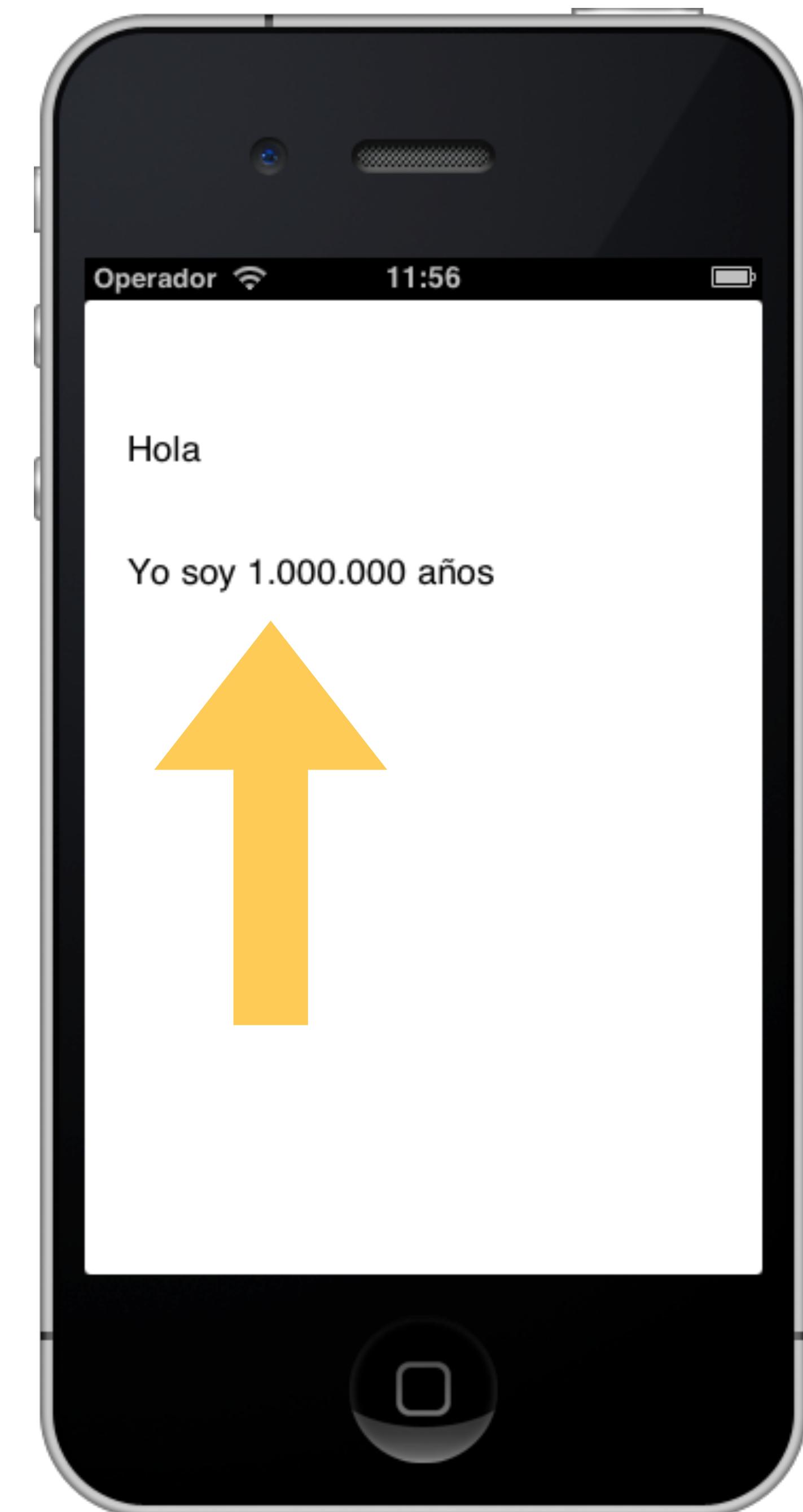
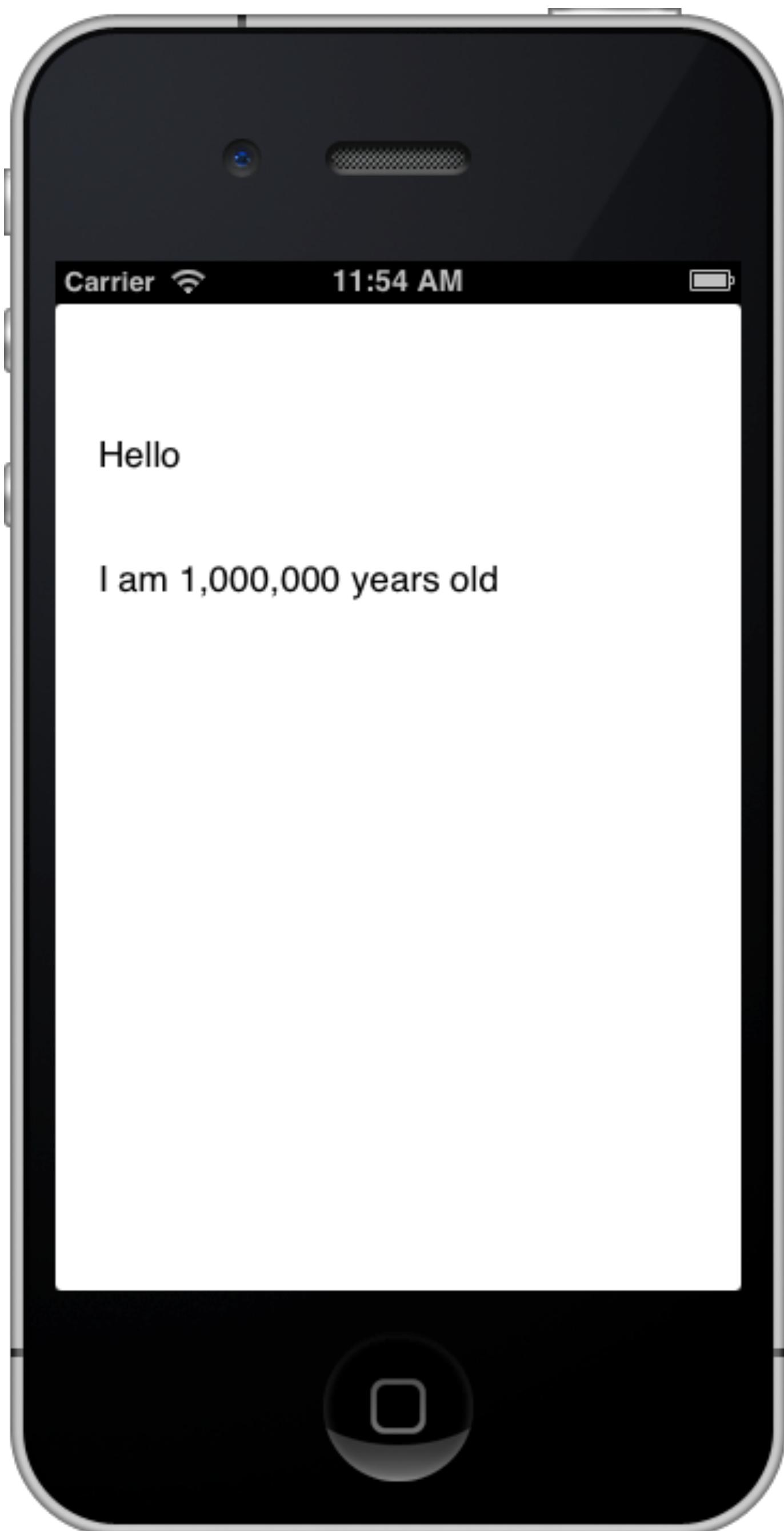
REGIONALIZE

- The region is not the same as the language
 - Different formatting based on the region
- Examples
 - Color vs. colour



REGIONALIZE

- Example
 - 1,000,000 US
 - 1.000.000 Spain



REGIONALIZE

- Simply use 'formatting' APIs and you get all the behavior for free
- NSNumberFormatter will format based on the region

```
NSNumberFormatter *numberFormatter = [[NSNumberFormatter alloc] init];
[numberFormatter setNumberStyle:NSNumberFormatterDecimalStyle];
NSString *numberString = [numberFormatter stringFromNumber:@(1000000)];

self.ageLabel.text = [NSString stringWithFormat:NSLocalizedString(@"I am %@ years old", nil), numberString];
```



ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 9

FINAL PROJECTS

FINAL PROJECTS

FINAL PROJECT REQUIREMENTS MPCS51032 - SPRING 2020

Application

All applications will be graded on implementing and complying with the following specifications, as well as the implementation of their functionality. The completeness of the submitted app is the most important objective, not necessarily a full realization.

FINAL PROJECTS

- Due date
 - Graduating students need projects submitted by June 5th at 5:00pm
 - Non-graduating by June 9th at 5:29pm

CLASS IN REVIEW

CLASS IN REVIEW

- Children's Book App
 - PageViewController
 - UIKitDynamics, Property Animators
 - AVSpeechSynthesis, AVAudioSessions
 - Objective-C
 - NIB base view controllers
 - Shaq and the Beanstalk...

CLASS IN REVIEW

- Journal
 - CloudKit
 - Push Notifications
 - Sync engine with Codable Protocols
 - App Groups
 - Extensions (Today, Share)
 - MLKit Vision Framework
 - MapKit (clustering)
 - Local Authentication
 - Notification Center
 - UISearchController

CLASS IN REVIEW

- Game Playground
 - SpriteKit

CLASS IN REVIEW

- Pop Quiz
 - watchOS using UIKit
 - watchOS using SwiftUI
 - CoreData
 - Cross platform development

CLASS IN REVIEW

- Pop Quiz
 - watchOS using UIKit
 - watchOS using SwiftUI
 - CoreData
 - Cross platform development
 - Notifications
 - Complications



THE UNIVERSITY OF
CHICAGO

THANK YOU



SPRING



Binkowski