iPhone Application Programming WS 15/16–*Lecture 13*

tvOS

Lecturers:
Christian Corsten, M.Sc.
Krishna Subramanian, M.Sc.

Media Computing Group | RWTH AACHEN UNIVERSITY

Outline & Learning Objectives

- What is Apple TV?
- Human Interface Guidelines
  - Design principles, focus & parallax, UI elements, loading
- App development
  - Focus engine, Siri remote, game controllers, on-demand resources, iCloud storage, AVPlayerKit
  - TVMLKit
- Top Shelf
- Demos (in between)
- UI design for tvOS apps
- Basic native app development
  - Overview of important SDKs
  - Remote/game controller communication
  - On-Demand Resources
- Basic app development using TVML & JS
- We can't provide details–
  Further information at:
  developer.apple.com/tvos/documentation

What is Apple TV?

- Set-top box to be connected to a big screen (TV) and broadband internet
- 4th generation: tvOS (9.2), Siri remote with voice search
- Play games, use apps, watch movies, shared experiences: "*The future of television is apps.*"
- tvOS is a distinct OS for Apple TV, derived from iOS, but has exclusive frameworks
- 64 bit A8 processor, 32/64 GB storage, 2GB RAM, 1080p, WiFi 802.11ac, 10/100 Mbps ethernet, USB-C for development

Media Computing Group | RWTH AACHEN UNIVERSITY

Capacitive touch pad
(single touch gestures)

THE MARTIAN
MENU
54:24
-1:27:38

FOX
Empire
Microphone
Siri
MENU
"Show me popular TV shows"

Accelerometer & gyroscope

MFI game controllers
steelseries
MENU
Y
X
B
A

Custom game controller

372 NOTE STREAK
MAKE SOME

★★★★★

Amanda’s Entire Home in New York

$250 · 1 ROOM · 2 BEDS

Top Movies
MAD MAX FURY ROAD
FURIOUS 7
PITCH PERFECT 2
AMY
Disney PIXAR INSIDE OUT
movies iTunes
tv shows iTunes
Movies

In-Class Exercise

How does the user interface of a tvOS app differ from how users interact with an iOS app?

- In- and output are split (no direct manipulation)
- UI is at distance (what item is currently selected?)
- Single touch only, physical buttons
- Multi-user interaction
- Permanent connectivity

What Makes a Good tvOS App?

- **Uniqueness**
- Support the living room experience
- **Intuitive** use of Siri remote
- Consider **first time users**
- Maintain **performance**
- **Design specifically** for Apple TV (not just ported from iOS)
- Think carefully about user interaction (intuitive, simple, fluid UI)
- Make **focus obvious** (user should not get lost)
- Create a shared experience (at next launch of an app, the user could be different)

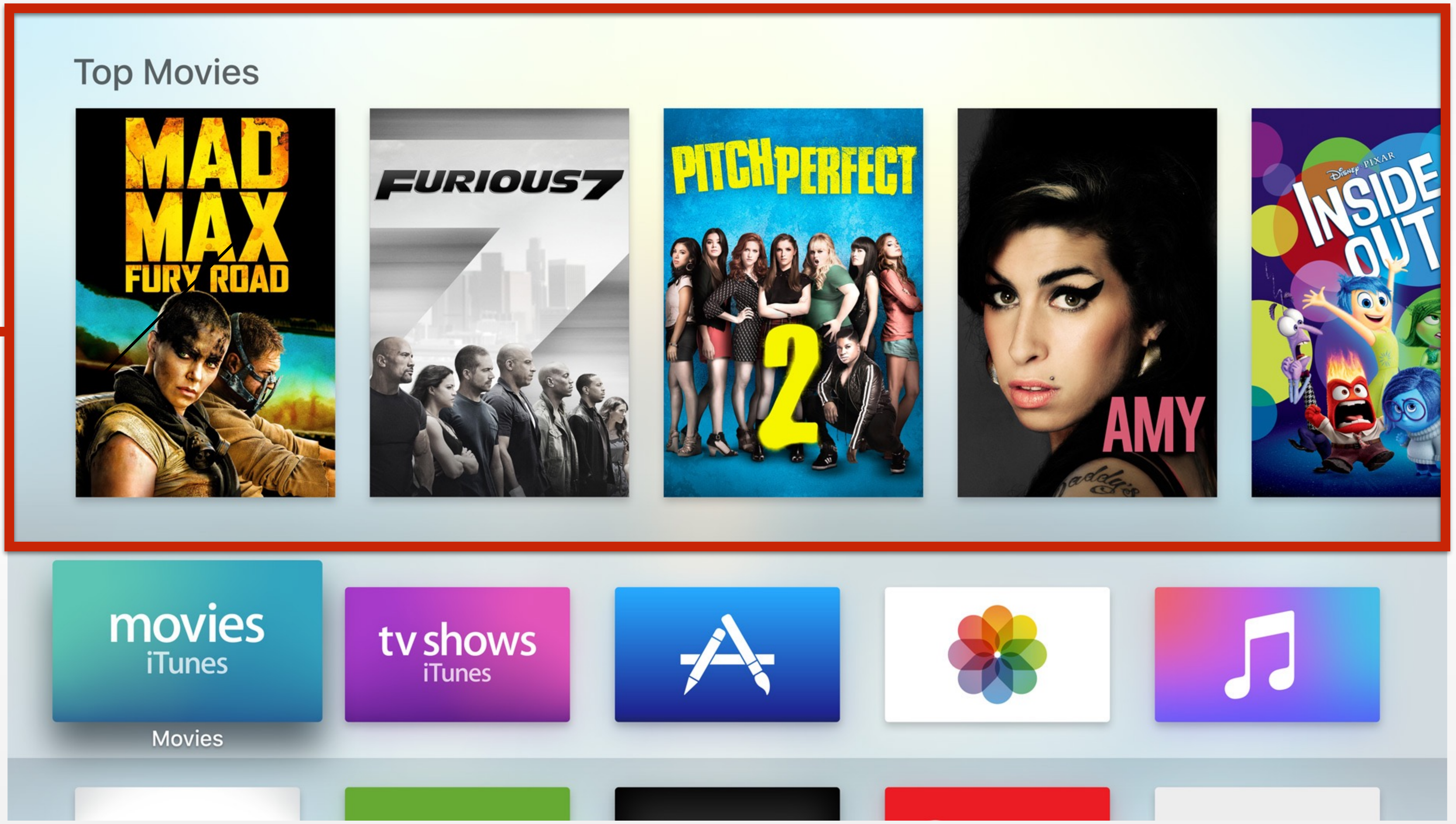
Apple TV Human Interface Guidelines

Three Key Design Principles

1. Connected
    - Interaction **across** the room
    - Siri remote **connects user with content** through gestures
2. Clear
    - Use **consistent** layouts
    - Let users know **where** they are in your app at any time
    - Element in **focus** at clearly visible from at distance
3. Immersive
    - Exploit the massive canvas
    - Use **fluid** animations

Home Screen and Top Shelf

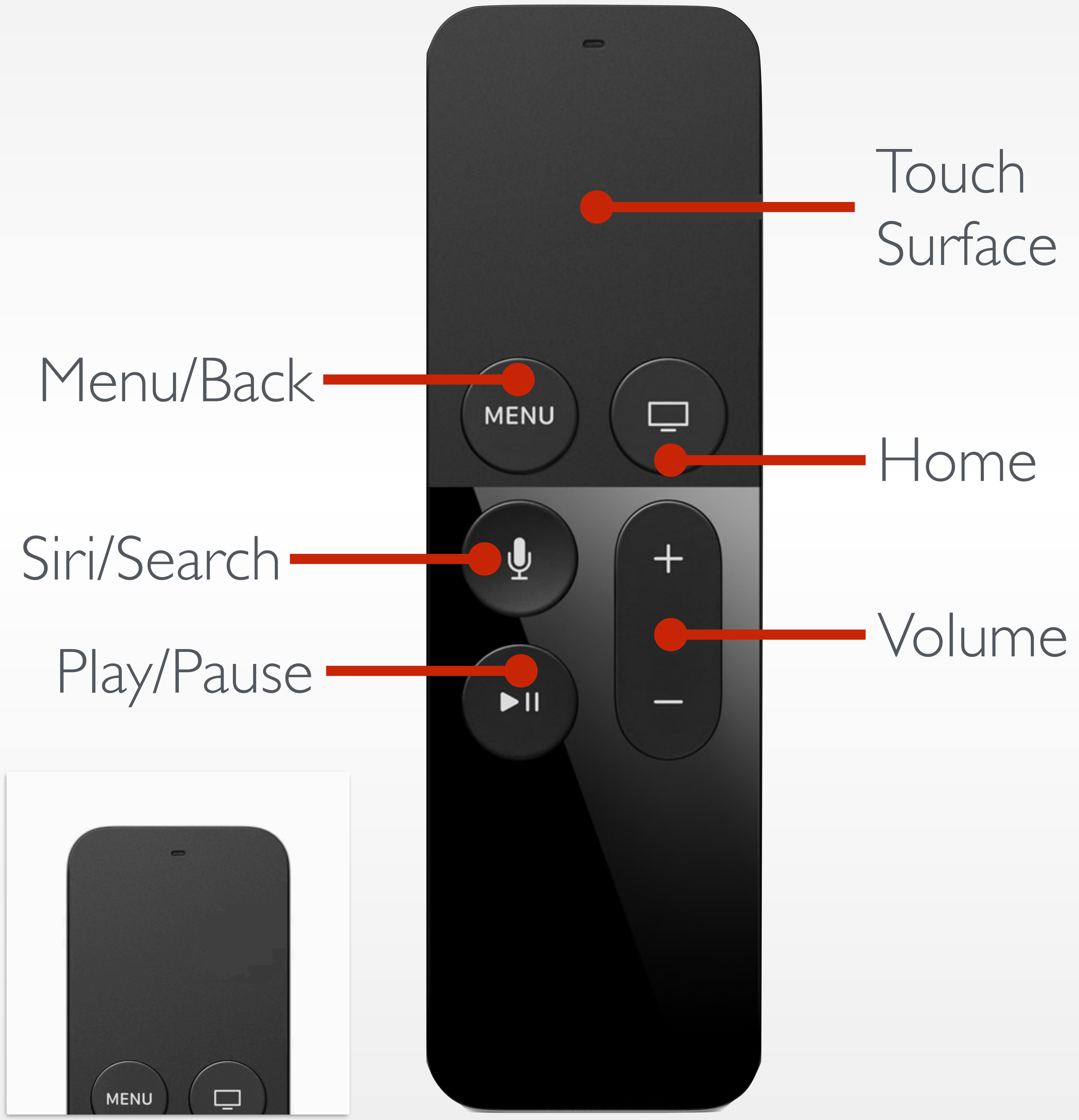
- Home Screen shows installed apps
- Grid-based navigation
  - Select, click, launch
- Top five items link to the Top Shelf
  - Showcase area that displays app-specific content
  - Can be a shortcut into your app
- App icon should build an emotional connection
  - Unique, affecting, memorable

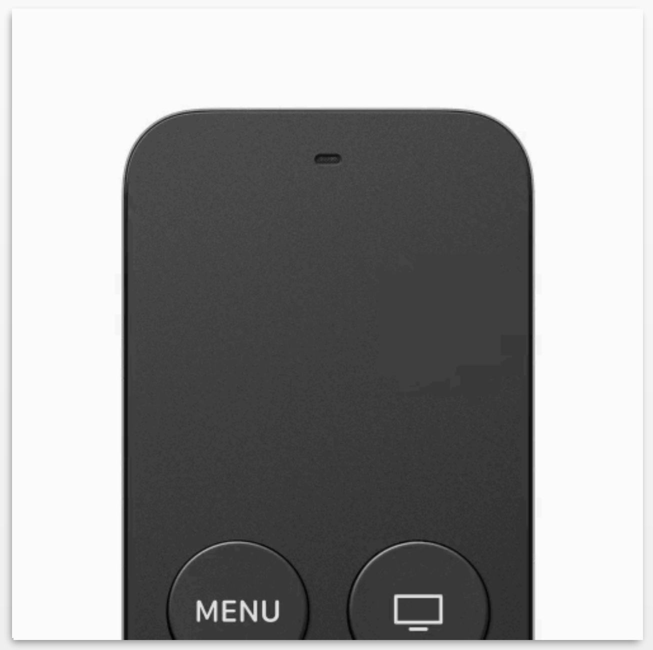


ZDF Livestreams
zdf_neo
zdf.kultur
movies iTunes
tv shows iTunes
ZDFmediathek

Remote: Primary Input Method

- 6 Physical buttons: use them consistently!
  - E.g., Play/Pause: controls media playback in apps, performs secondary button behavior/skip intro in games
- Give users a way back to the previous screen in the app
  - Games: Menu button should display an in-game pause menu
- Touch surface: single-finger gestures
  - Swipe: to move focus, scrolls with inertia
  - Click: intentional action to activate a control or select an item
  - Tap: navigates through items one-by-one (d-pad), displays hidden controls





Game Controllers

- Game controllers are optional, can also be used to navigate Apple TV
  - If a game controller is supported, the remote must must be usable as a game controller, too!
- iOS device can act as a controller as well
- Check for presence of a controller at launch of the app
  - Take most recently used device as default
  - Use device whichever presses "start"
- Inform users about the game controller's capabilities beyond those provided by the remote

Navigation

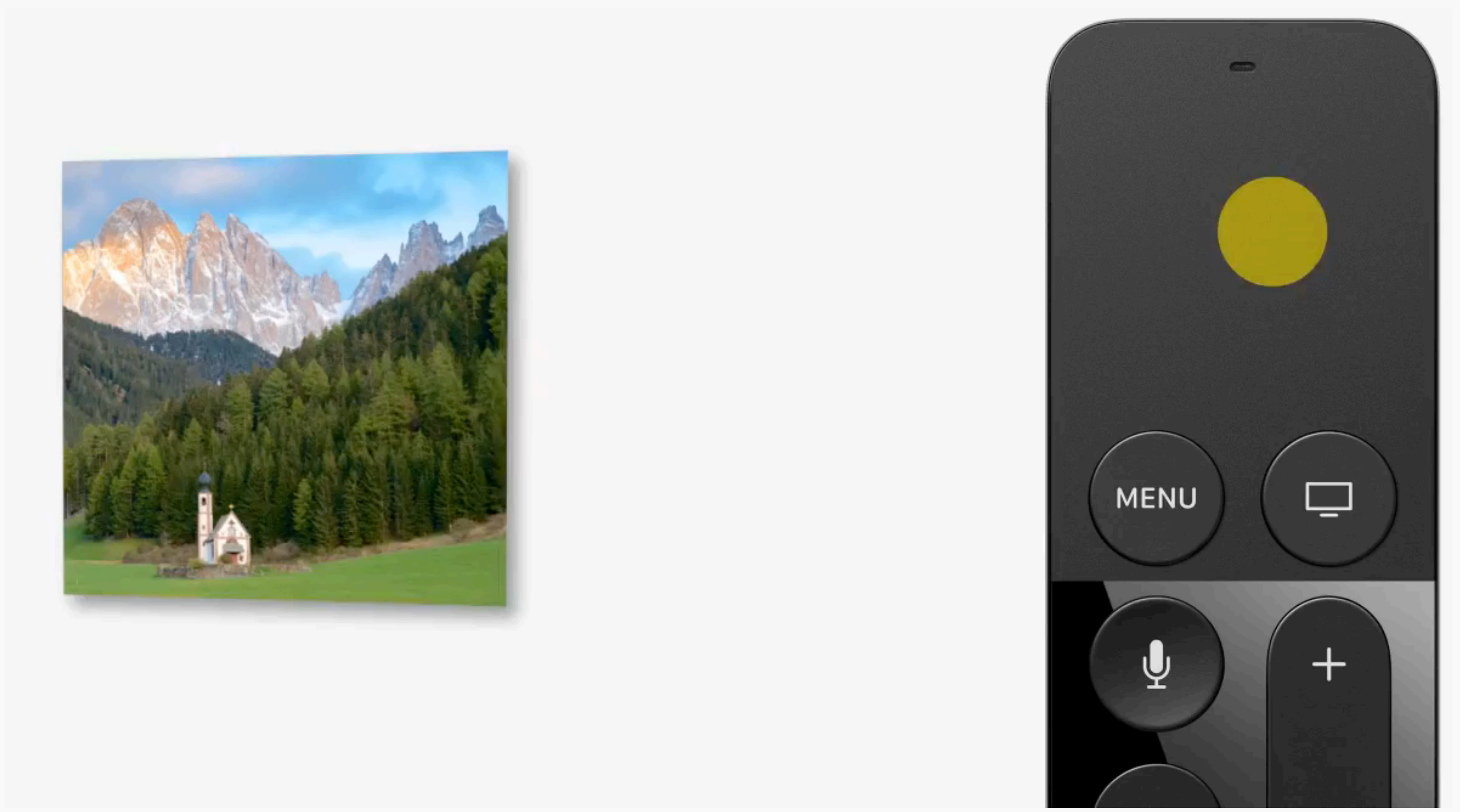
- Apps: user navigates through stacked screens of content
  - Organize UI such that it requires the user to navigate a minimum number of screens
  - Implement backward navigation by using the Menu button, but **do not display a back button** (exception: cancel for purchase/deletion)
- Show content on a single screen (scrolling) instead of splitting content up
- Favor horizontal navigation of content
  - Swiping with the thumb to the left and right is more **ergonomic and natural**
- Use standard navigation controls
  - Page controls, tab bars, segmented controls, table views, collection views, split views

Focus & Selection

- Interaction is based on a **focus model**
  - Subtle animations
  - Parallax effect
- Use UIKit and focus API to get motion and visual effects for free
- Move focus in the **expected direction**
  - Focus moves in the direction of the gesture (content may move left when focus moves to the right!)
  - However, for full-screen elements: objects should move in the direction of the gesture (swiping to the right moves a photo from left to right)
- Make the focused item **obvious**
  - Test: a "fresh" user entering the room should be able to immediately tell which item is at focus

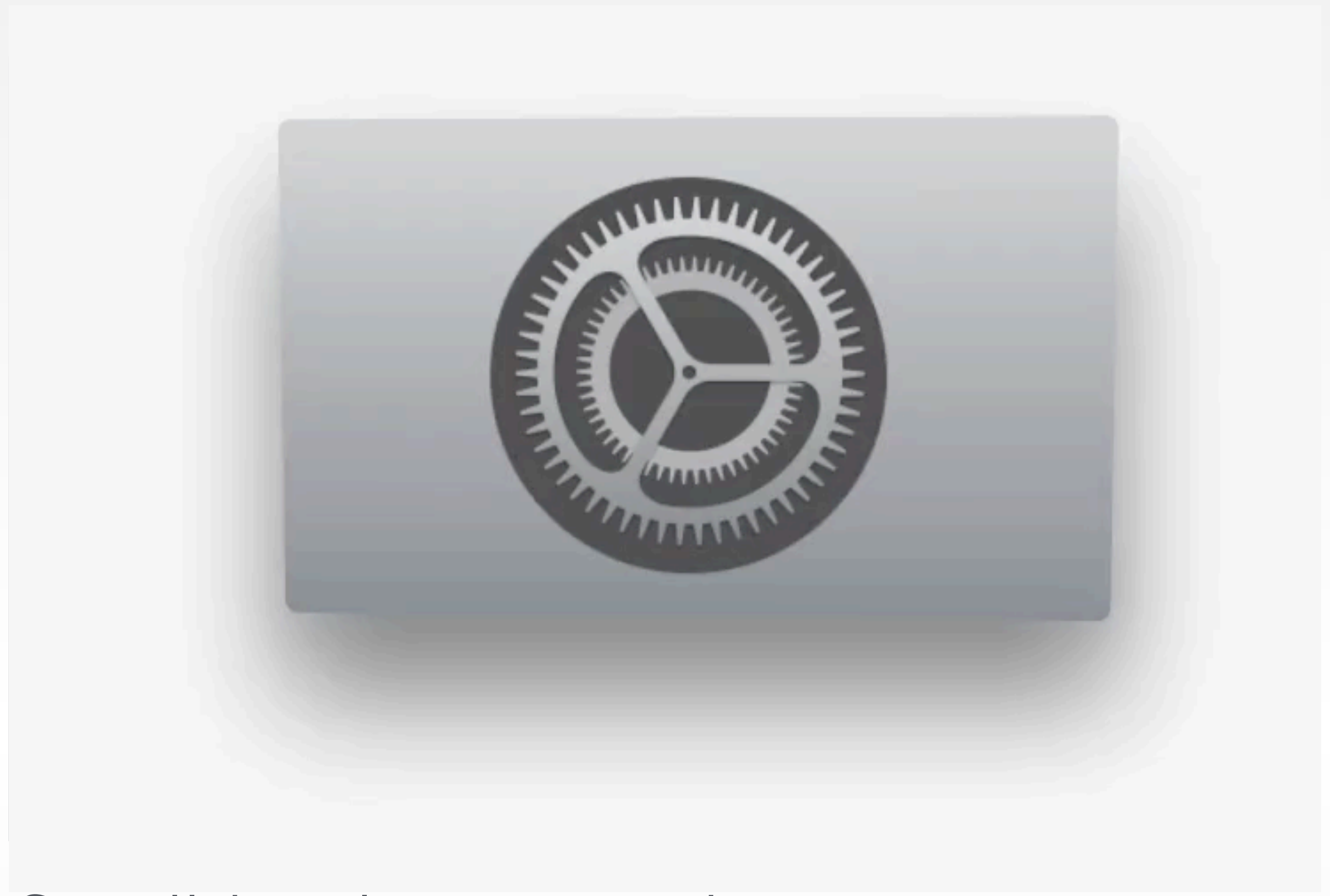
Focus & Selection

- Parallax makes focused items more responsive to user interaction
  - Effect of depth through layering, transparency, scaling, motion
  - LSR (Layer Source Representation) image has 2 to 5 layers
- Supply assets for the larger, focused size
  - Make sure images look sharp at any state
- Include appropriate padding between focusable images
- **Do not display a cursor!**
  - Use the focus model for navigation
  - Exception: games (e.g., move a crosshair)

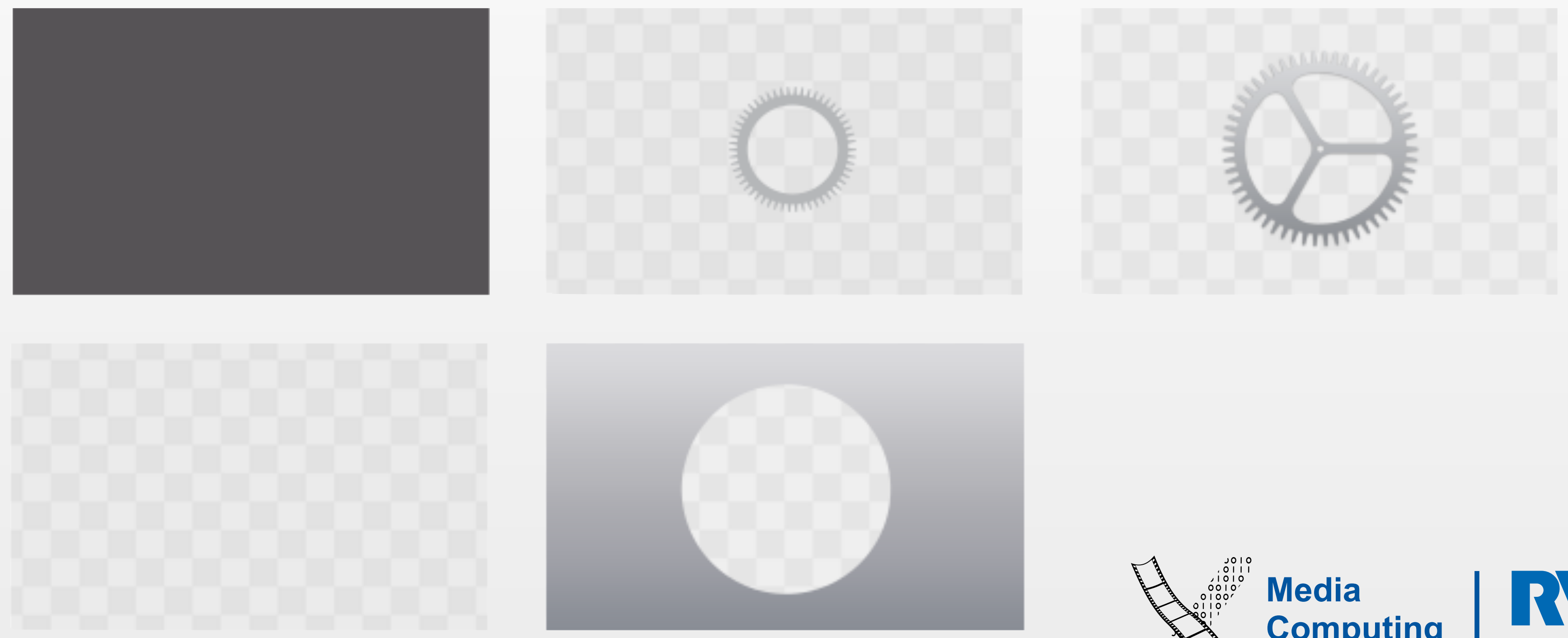


LSR App Icons

- Layered image is required for app icons
  - 2–5 layers
  - Single focus point, simple background, no screenshots, reduce text
- Gradients: Top-to-bottom, light-to-dark
- Varying opacity increases depth and liveness
- Images come at different sizes and aspect ratios
  - Small vs. large app icon, Top Shelf images, Game Center images
- LSR is now supported by UIImageView
  - To enable parallax for a UIImageView when it is a sub view, set `adjustsImageWhenAncestorFocused` to `true`

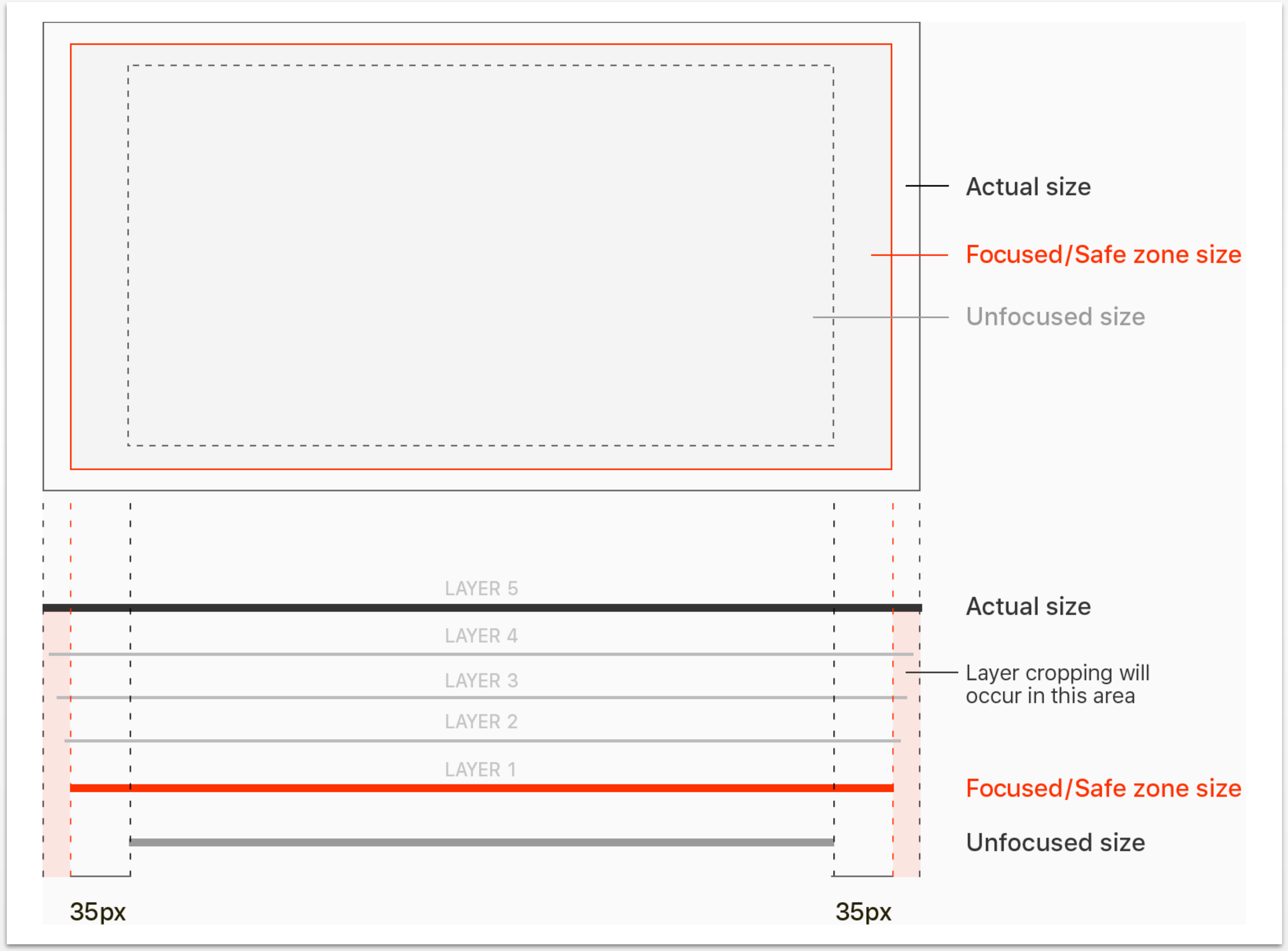


Small in-class exercise:
How would you layer this app icon?



Creating LSR Images

- Leave a **safe zone** around your content
  - Foreground layers are more cropped than background layers
- Small app icon:
  - **400 x 240px**, 370 x 222px (focused), 300 x 180px (unfocused)
- Tools for exporting .lsr images
  - Parallax Previewer for OS X (.png, .psd, .lsr)
  - Parallax Exporter plug-in for Adobe Photoshop
  - Xcode (drag .png's into app's asset catalog)
  - developer.apple.com/tvos/human-interface-guidelines/resources

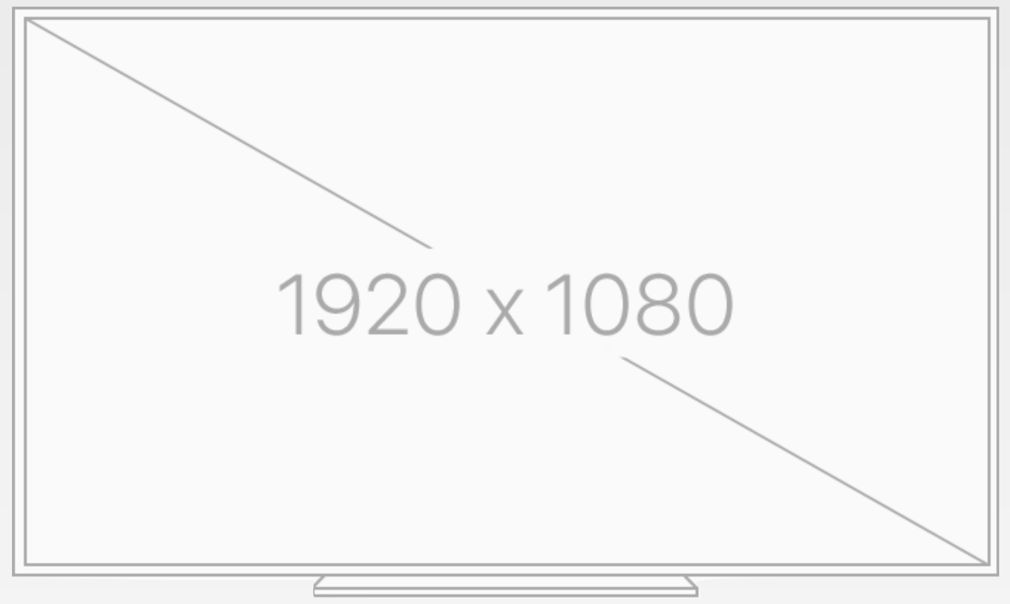


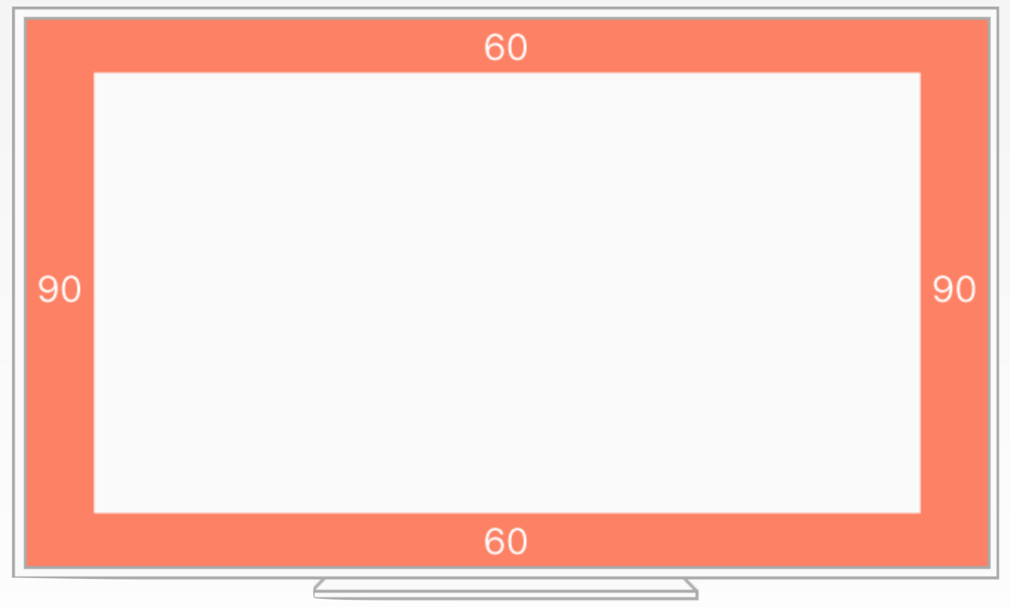
Demo

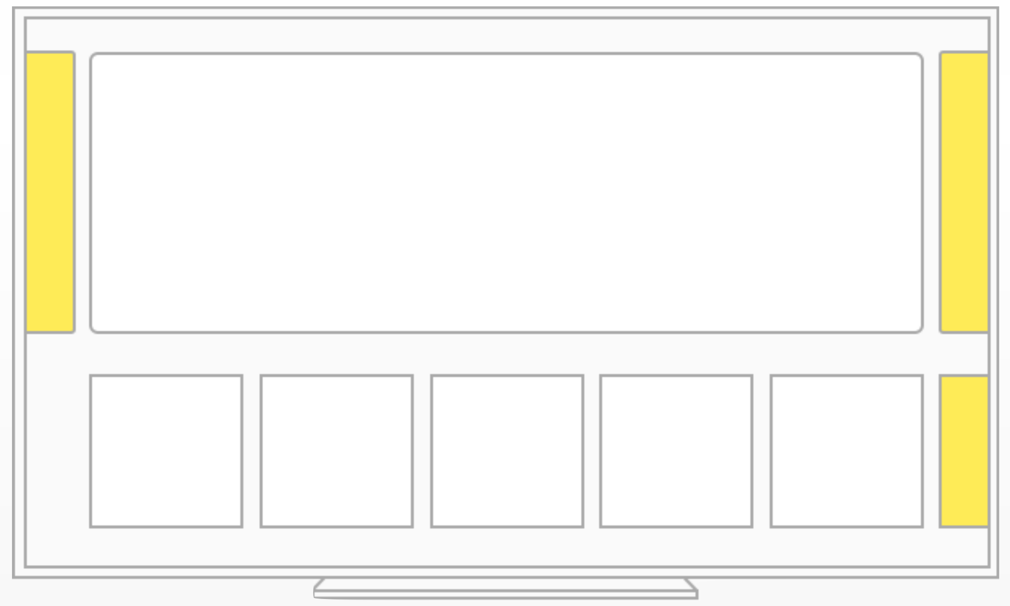
Creating an LSR App Icon with Photoshop and Parallax Previewer

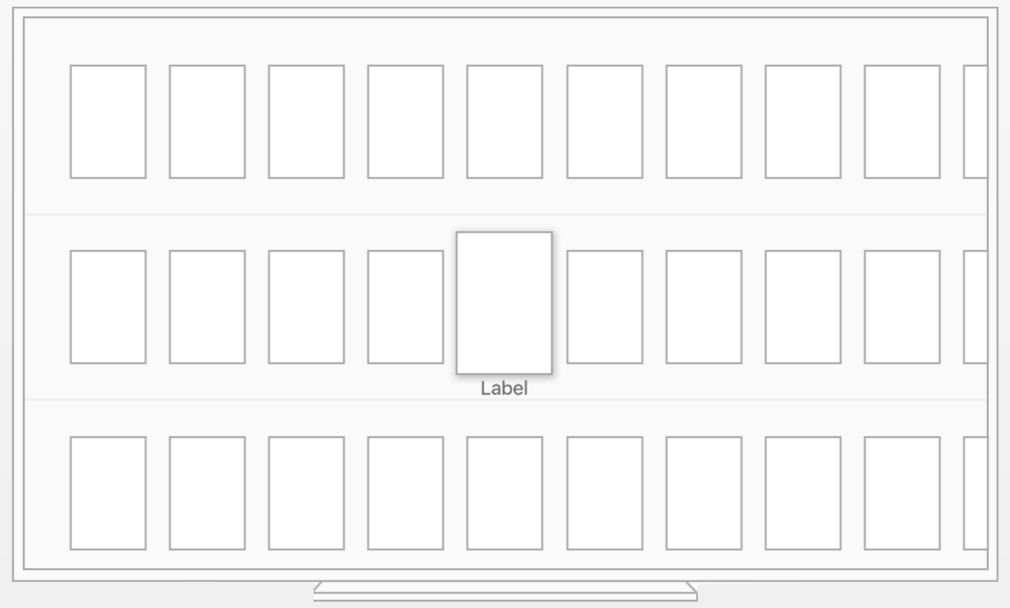
Visual Design

- TVs **vary widely** in size, but the UI does not adapt to the screen size
  - 1920 x 1080px, 16:9, @1x resolution
- Keep primary content away from the edges of the screen
  - 90px left/right, 60px top/bottom – mitigates copping issues due to overscanning
- Hint at additional content by showing parts of the **offscreen items**
- Use **grids** for collections of content
  - 3-/5-/6-9-column grids provide optimal viewing experience
  - See `UICollectionViewFlowLayout` class reference
- Use layout templates for media-centric apps (TVML)









Typography & Accessibility

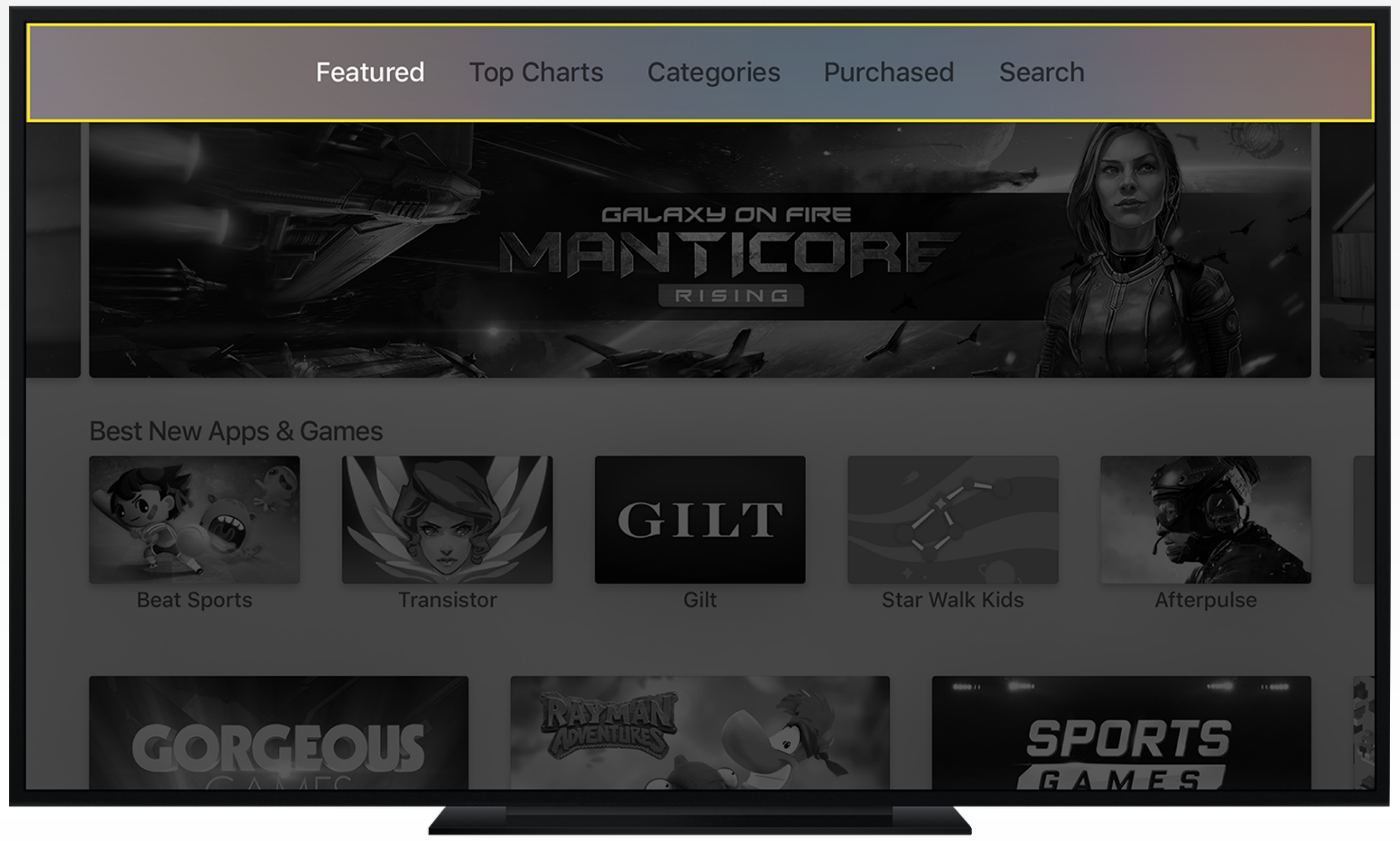
- San Francisco system font
  - San Francisco Text for texts with font size <39pt
  - San Francisco Display for texts with larger font size
- Body text style for primary content, footnote and caption style for labels and secondary content
- Yet, **reduce** the amount of **text**: "*Show, don't tell.*"
- System fonts automatically react to **accessibility** features
  - E.g., enable **bold** text
- Provide alternative text labels for images and interface elements
  - Invisible, only used for VoiceOver
- Include closed captions and audio descriptions in videos

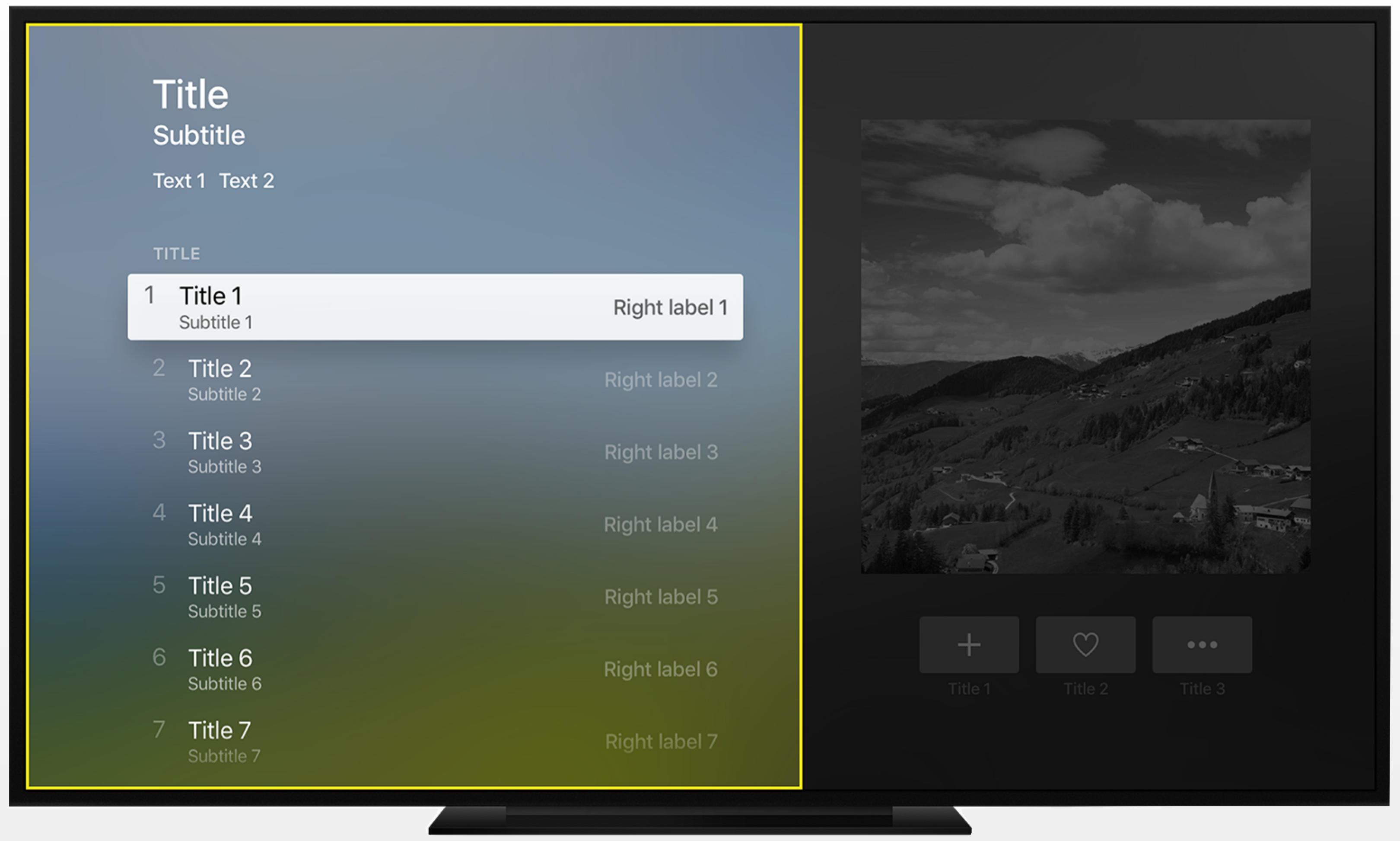
Video

- Use the system video player
  - `AVKit` framework (you'll see a demo later 😃)
  - Provides **consistent user experience** with the remote for video playback
- Show interactive elements on top of videos
  - Implement a **delay of 0.5 seconds** to pause media before displaying an interactive overlay
- For streaming, refer to HTTP Live Streaming (HLS) specification

Interface Elements: Tab Bars & Tables

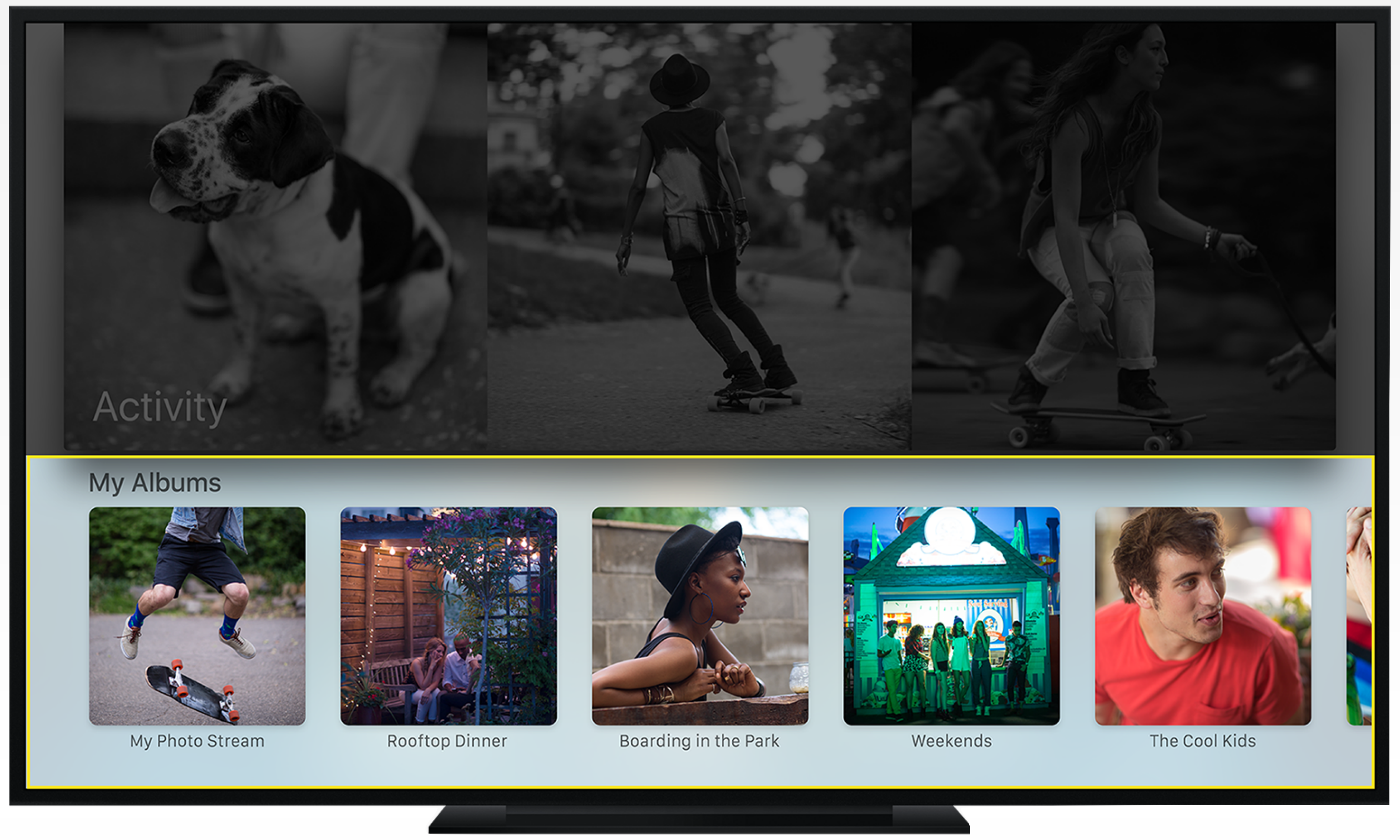
- Tab Bars (UITabBarController)
  - At the **top of the screen**
  - Tells users where they are in the app
  - **Slides away** when not in focus
  - Flattens information hierarchy and provides quick access
- Tables (UITableViewController) and Cells (UITableViewCellStyle)
  - **Scrollable**
  - Show text before loading images (increased responsiveness)
  - Deleting and reordering table rows takes longer on Apple TV than on iOS!



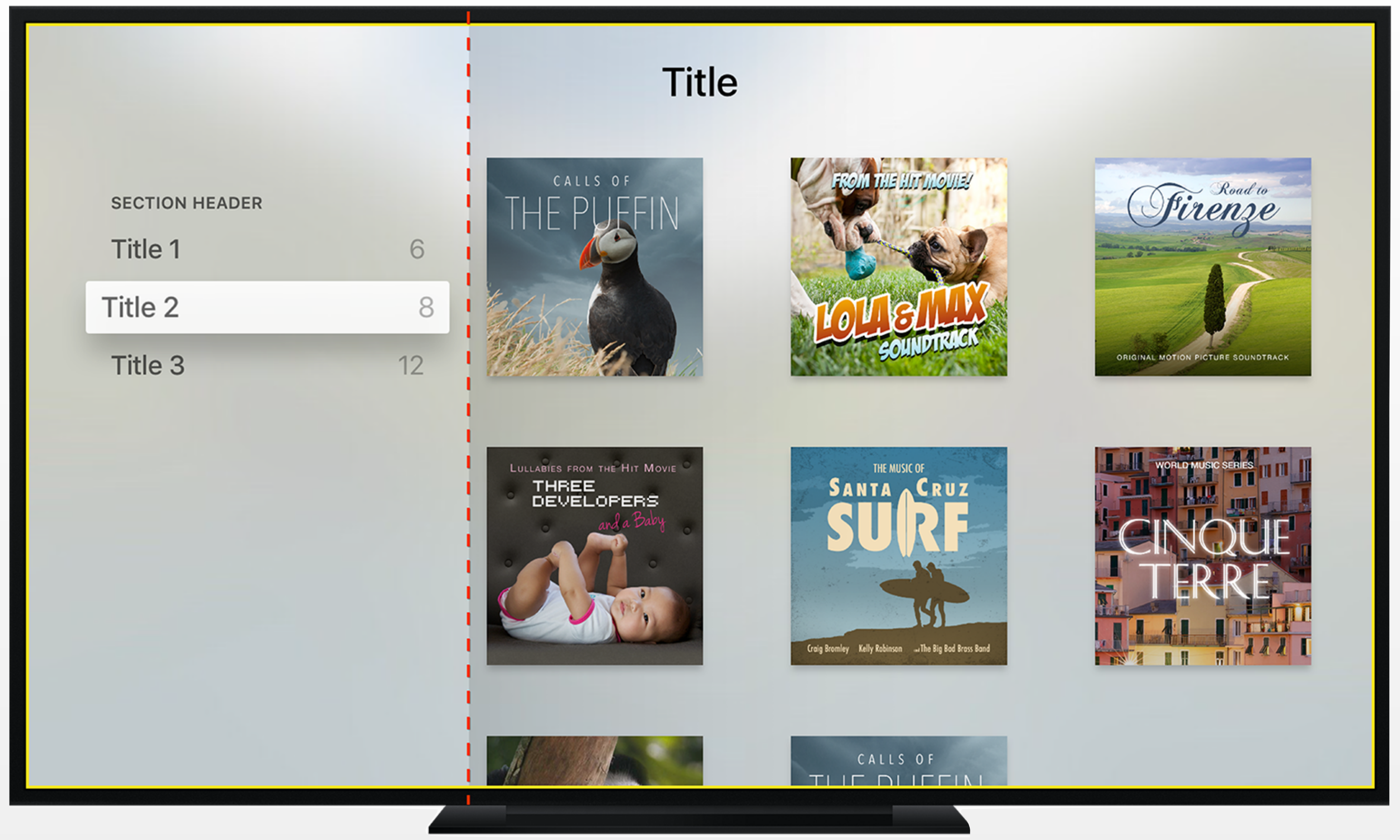

Title
Subtitle
Text 1 Text 2
TITLE
1 Title 1 Subtitle 1 Right label 1
2 Title 2 Subtitle 2 Right label 2
3 Title 3 Subtitle 3 Right label 3
4 Title 4 Subtitle 4 Right label 4
5 Title 5 Subtitle 5 Right label 5
6 Title 6 Subtitle 6 Right label 6
7 Title 7 Subtitle 7 Right label 7
Title 1
Title 2
Title 3

Collections & Split Views

- Collections (UICollectionViewController)
  - Typically used to **display and browse images** of varying size
  - Use table views for a collection of text!

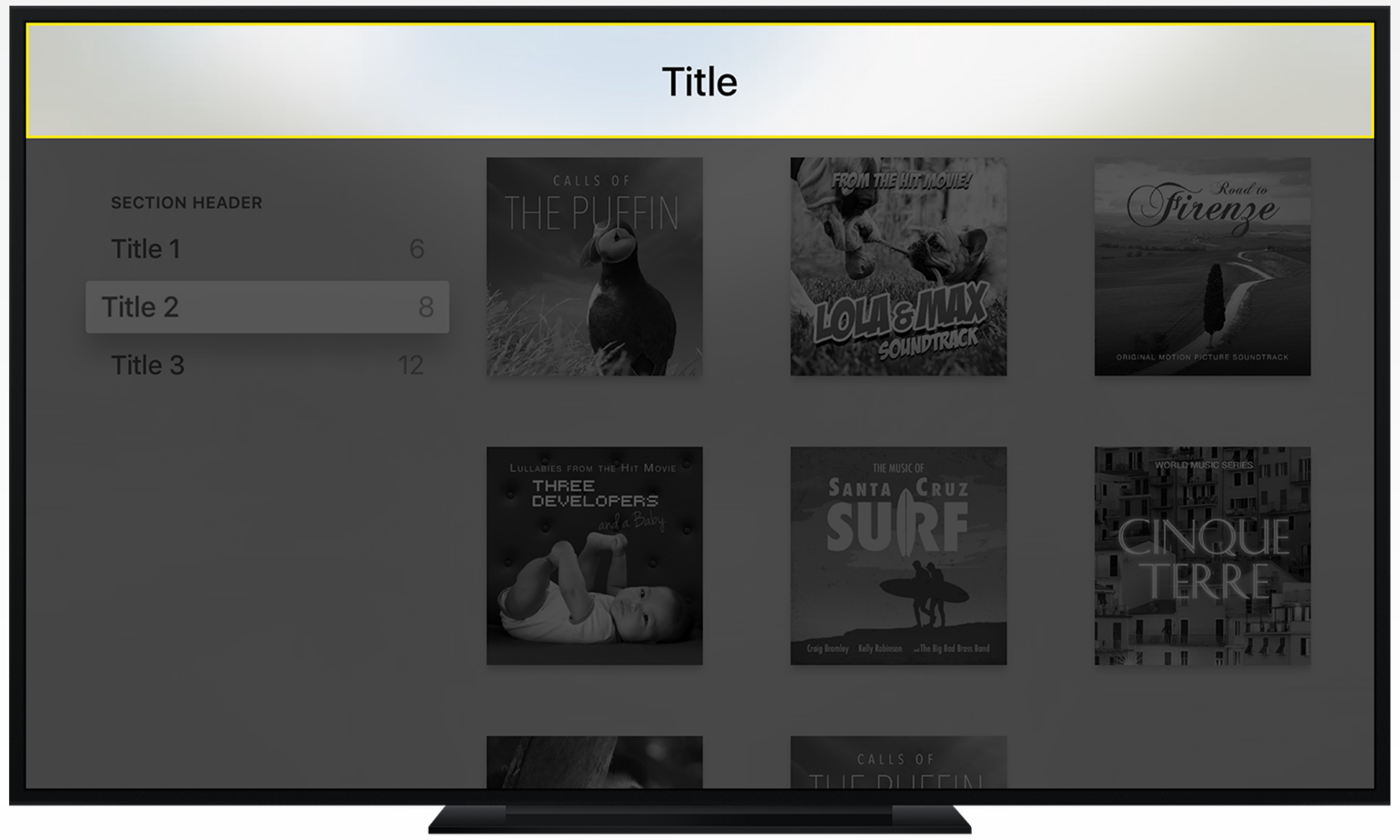


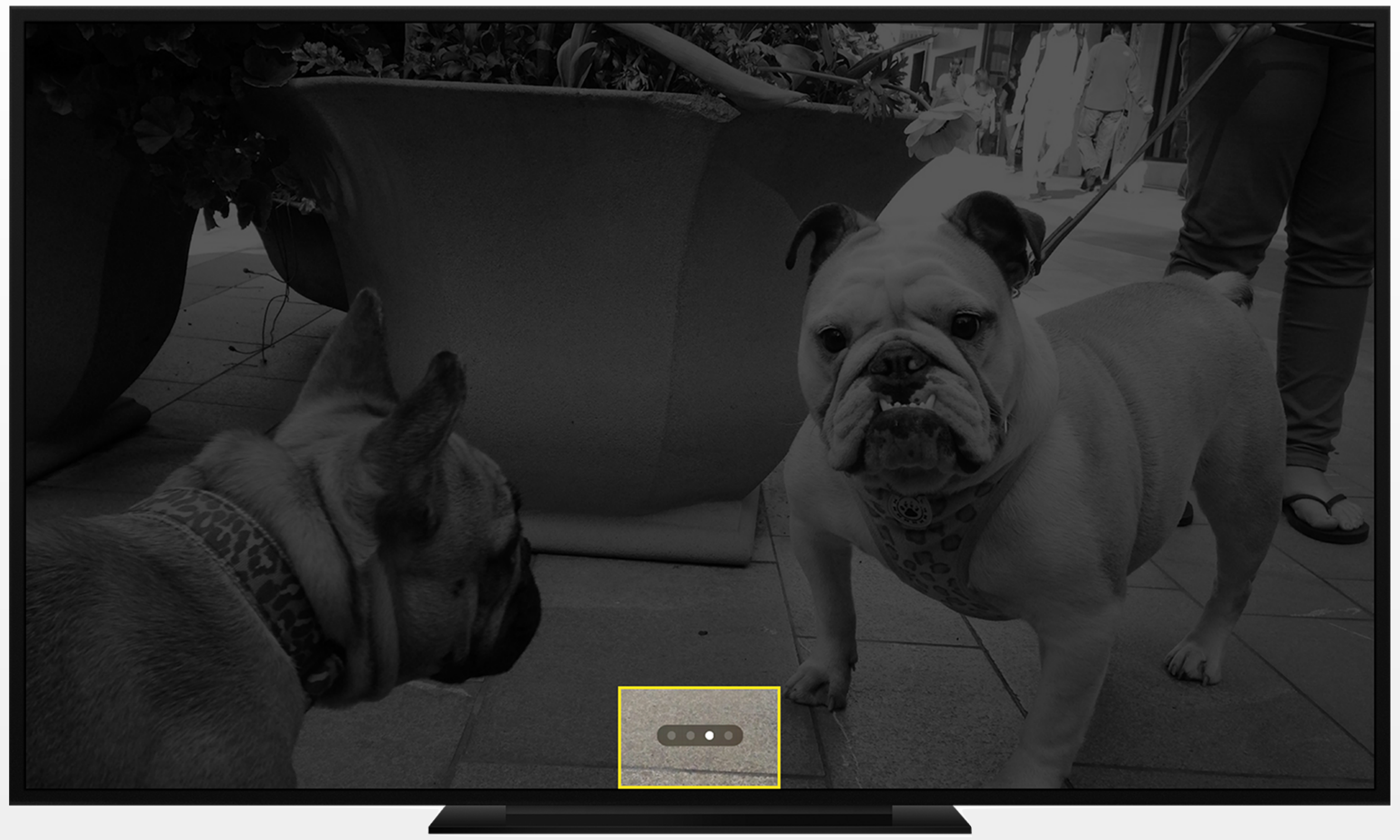
- Split Views (UISplitViewController)
  - **Two side-by-side panes** of content (1/3 vs. 2/3 by default)
  - Put persistent, focusable content in the primary pane, related information in the secondary pane (e.g., filterable content)



Navigation Bars & Page Controls

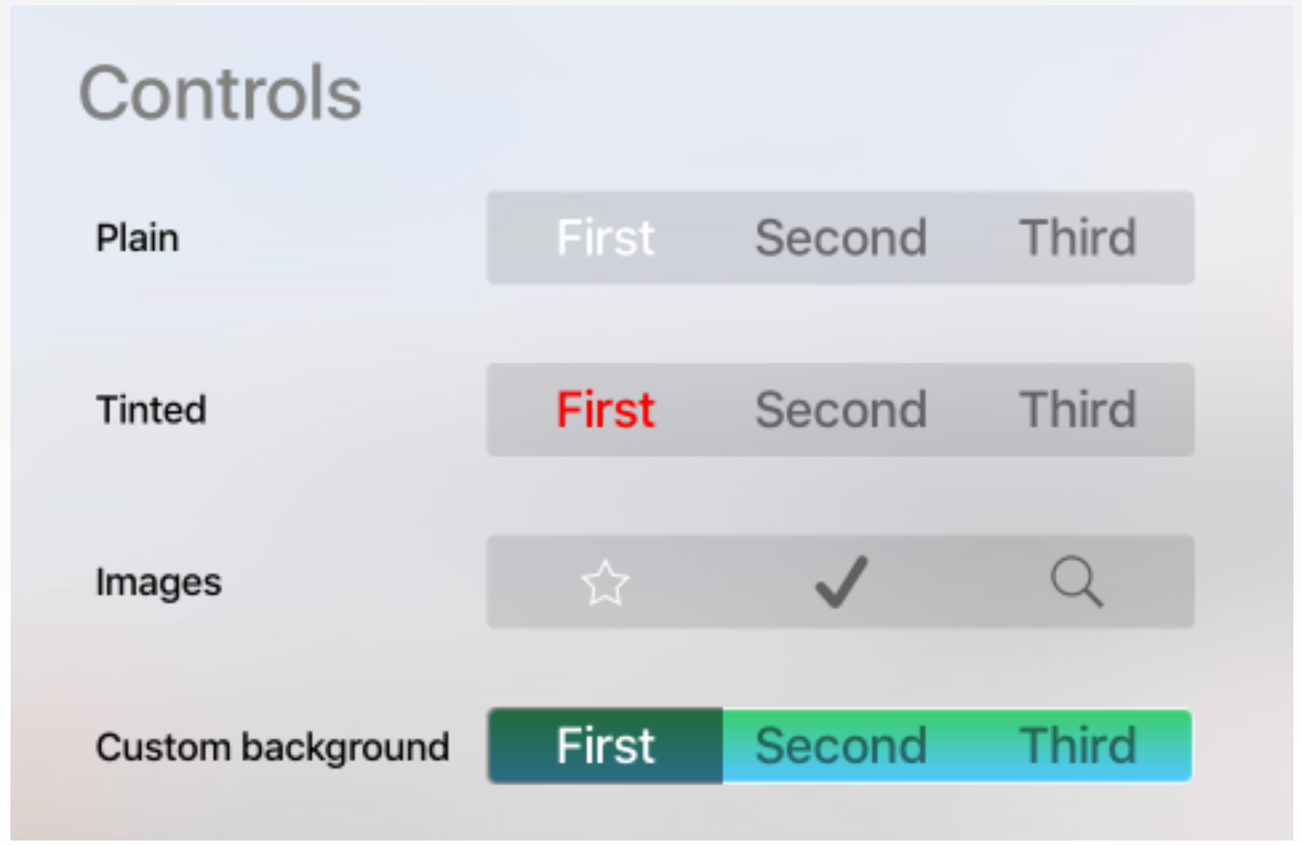
- Navigation Bars (UINavigationBar)
  - On **top of a view** to display a title, navigation buttons, and other interface elements
  - The navigation bat is **transparent** by default–fade content as it scrolls under the bar
- Page Controls (UIPageControl)
  - Communicates the **number of pages** and which one is currently active
  - Use them on collations of full-screen pages, but **do not display too many pages**

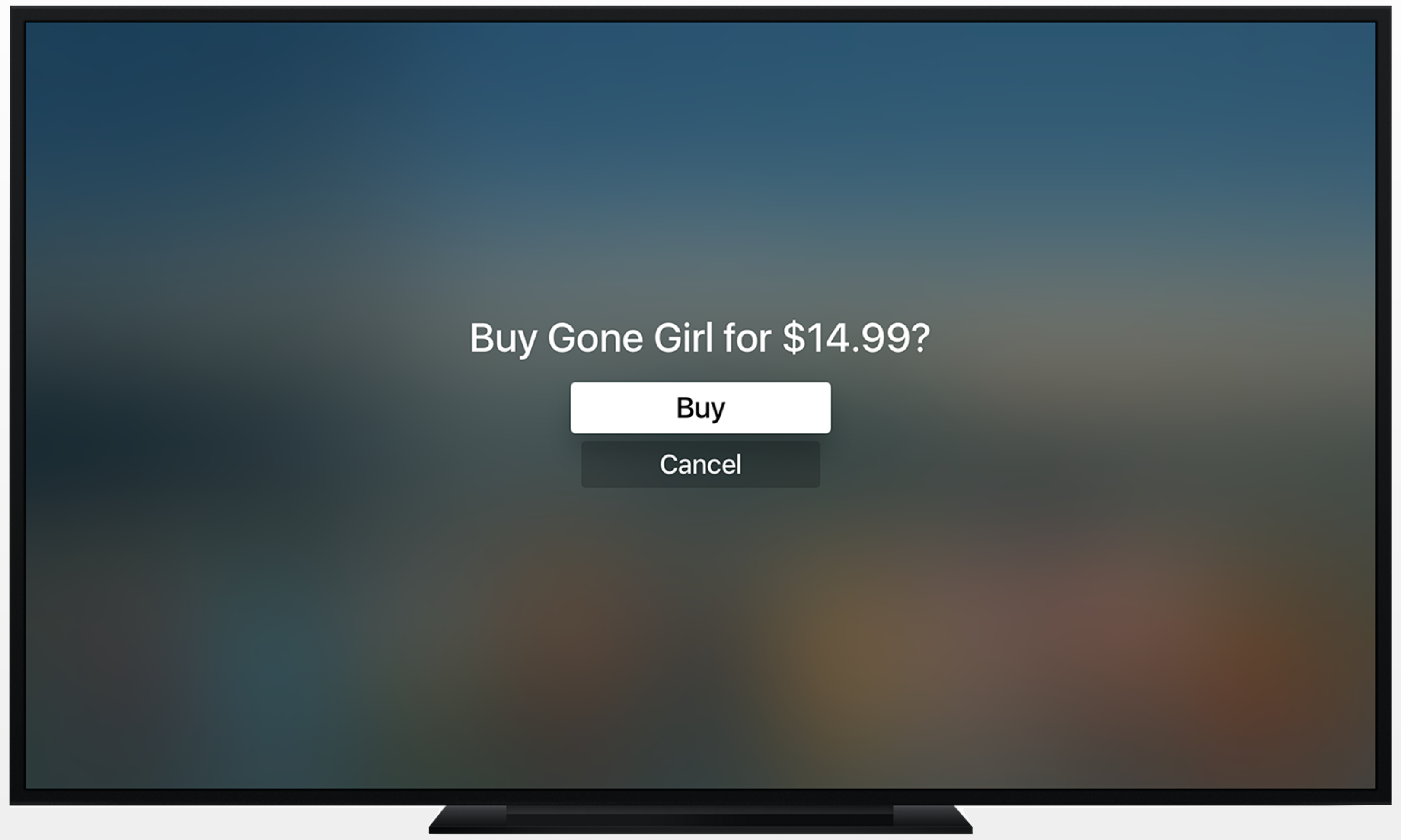




Segmented Controls, Alerts & Buttons

- Segmented Controls (UISegmentedControl)
  - **Linear set of mutually exclusive segments** to display a different view (e.g., playlist vs. album)
  - Do not put focusable items next to segments as they become already selected when focus moves to them
  - At max. seven segments–should be of equal width
  - Segment icons vs. text (**nouns!**)
- Alerts (UIAlertController)
  - Required (**multiword!**) **title**, optional message, one or more **buttons**
  - Use them only for important situations, e.g., to confirm destructive actions (UIAlertActionStyleDestructive)
  - Pressing the Menu button should have the same effect as clicking the cancellation button

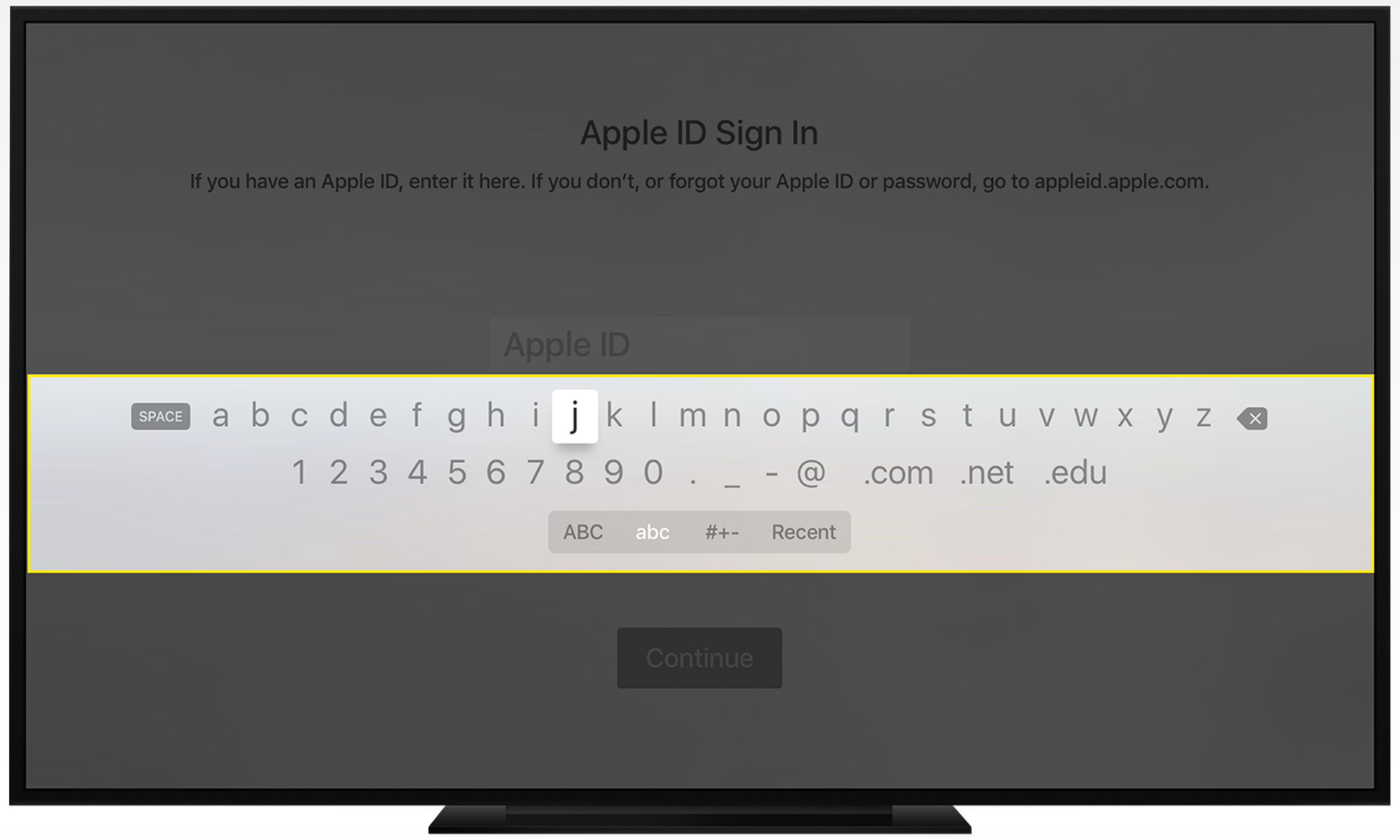




Text entry on Apple TV is tedious–
minimize text entry in your app!

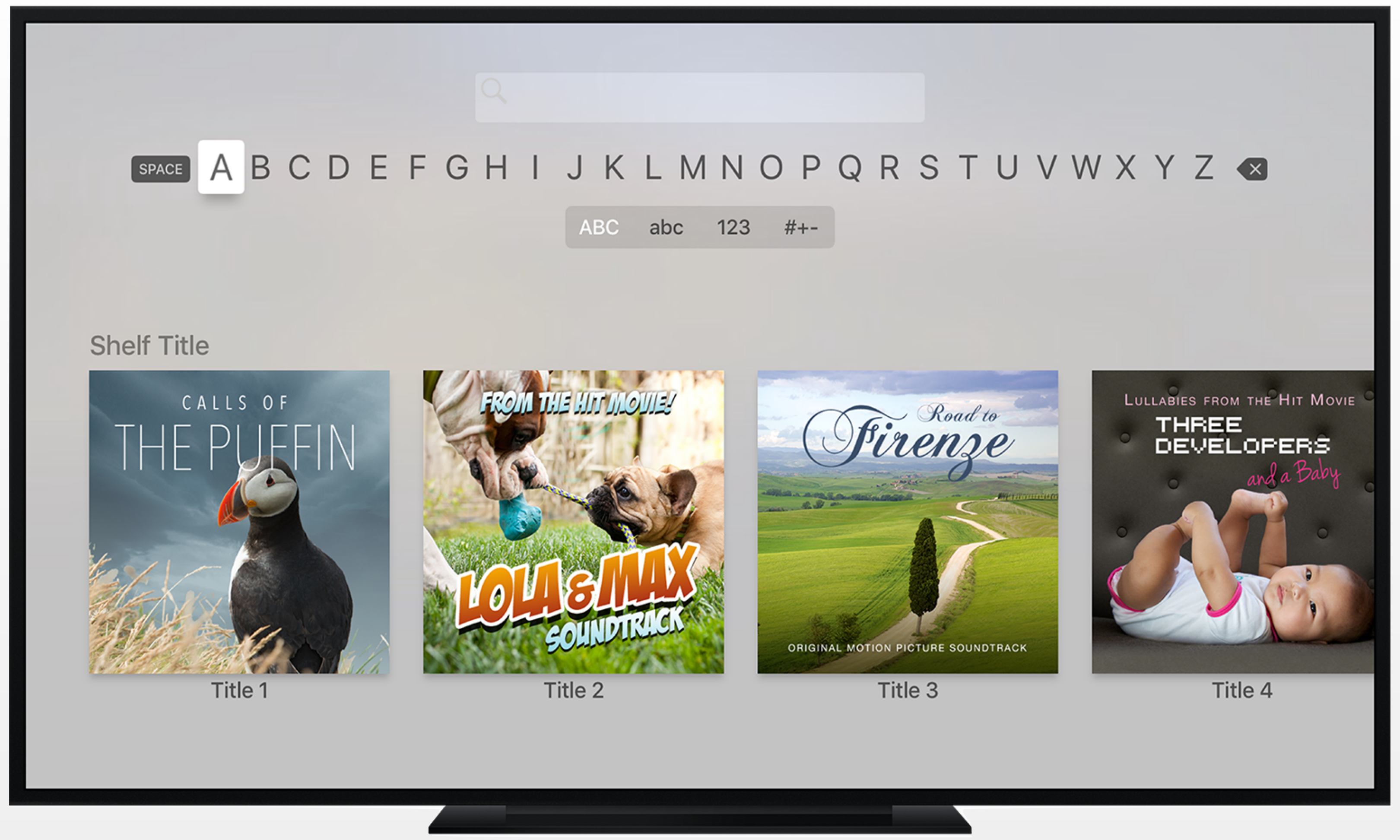
Text & Search

- Keyboards
  - Keyboard appears on screen when the user clicks on a text field
  - Different **layouts** dependent on type of text field (UIKeyboardType)
  - Siri remote: **linear** keyboard, other controllers: grid keyboard
- Search
  - Allow for recent searches
  - Show text before loading images (increased responsiveness)
  - Deleting and reordering table rows takes longer on Apple TV than on iOS!


Apple ID Sign In

If you have an Apple ID, enter it here. If you don't, or forgot your Apple ID or password, go to appleid.apple.com.

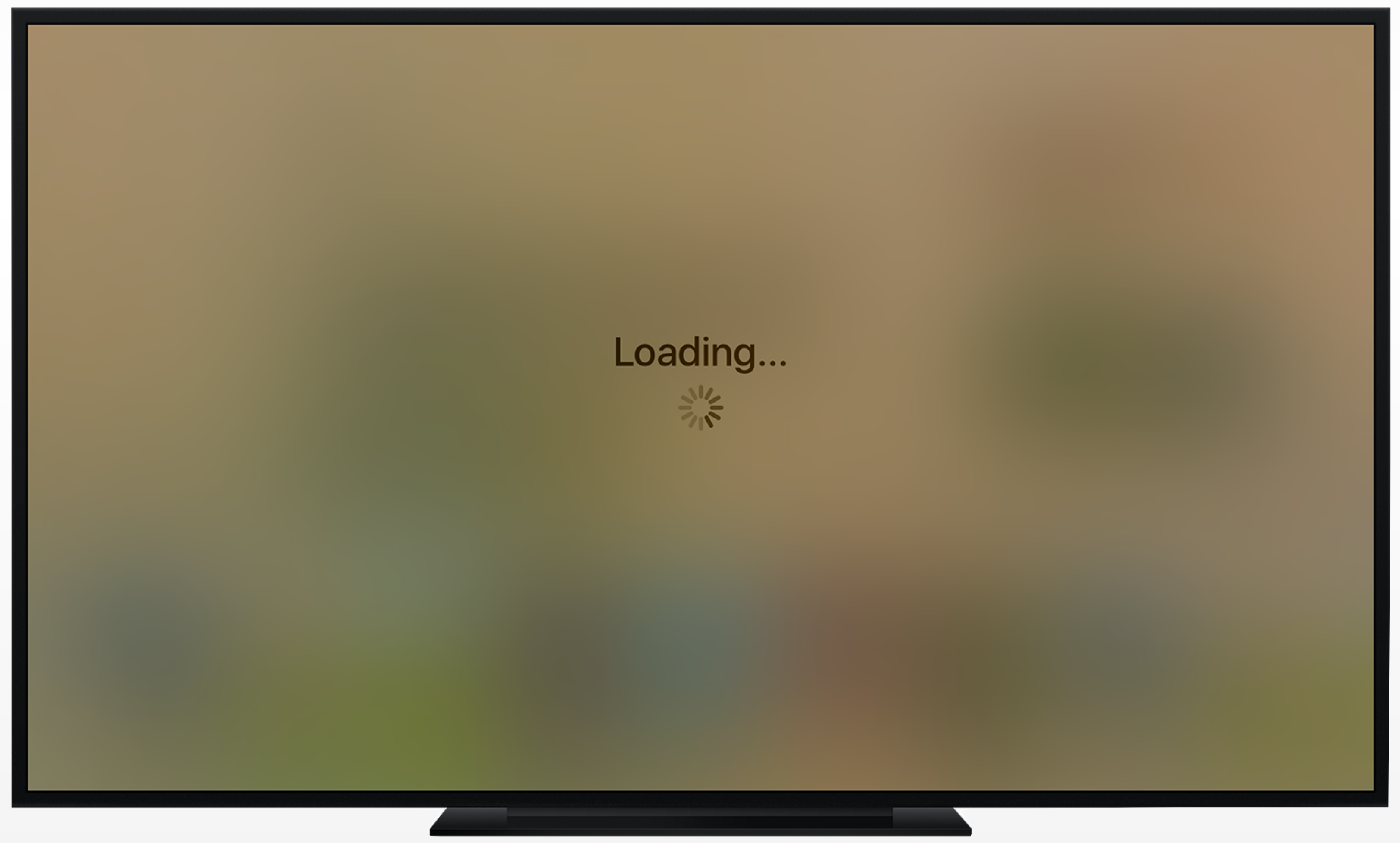
Apple ID
SPACE a b c d e f g h i j k l m n o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 0 . _ - @ .com .net .edu
ABC abc #+- Recent
Continue


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
ABC abc 123 #+-
Shelf Title
CALLS OF THE PUFFIN
FROM THE HIT MOVIE! LOLA & MAX SOUNDTRACK
Road to Firenze
ORIGINAL MOTION PICTURE SOUNDTRACK
LULLABIES FROM THE HIT MOVIE THREE DEVELOPERS and a Baby
Title 1
Title 2
Title 3
Title 4

Media Computing Group | RWTH AACHEN UNIVERSITY

Progress Indicators

- Loading content will be an issue (on-demand resources)
- Inform your users that the app is not stalled and tell them how long they have to wait
- Activity Indicators (UIActivityIndicatorView)
  - Activity indicators spin–**keep it spinning** so users know that something is happening
  - If helpful, provide additional information while the user is waiting (text label)
- Progress Bars (UIProgressView)
  - Fills from left to right
  - Use them **for tasks with well-defined duration** and report progress accurately
  - Prefer Progress Bars over Activity Indicators
- **Educate or entertain** if appropriate
  - E.g., a video continues on the game's plot while the next level is being loaded



Native App Development

App Development

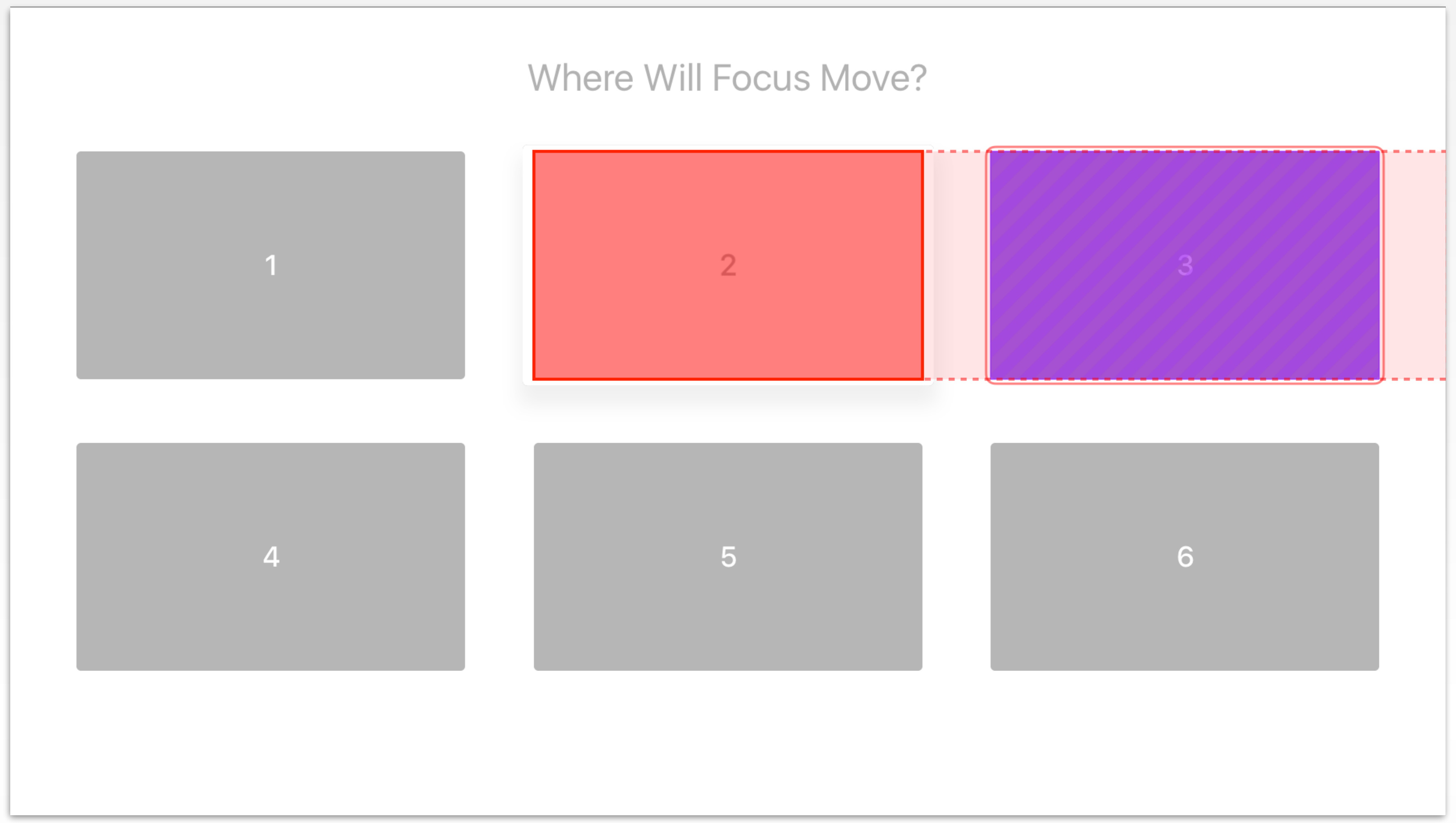
- You can use your iOS app as starting point
  - add a new target to your Xcode project and add new storyboards for tvOS)
- `#define TARGET_OS_TV 1` macro for tvOS specific code
- iOS and tvOS apps are distinct entities, but can be bundled as a universal purchase
- Native apps (support iOS frameworks) vs. TVML apps (can also be mixed)
- Supported languages: Objective-C, Swift, (JavaScript for TVML apps)
- New, tvOS-specific frameworks:
  - TVMLJS: JavaScript API to load TVML pages
  - TVMLKit: Incorporate JavaScript and TVML elements into your (native) app
  - TVServices: For Top Shelf extension

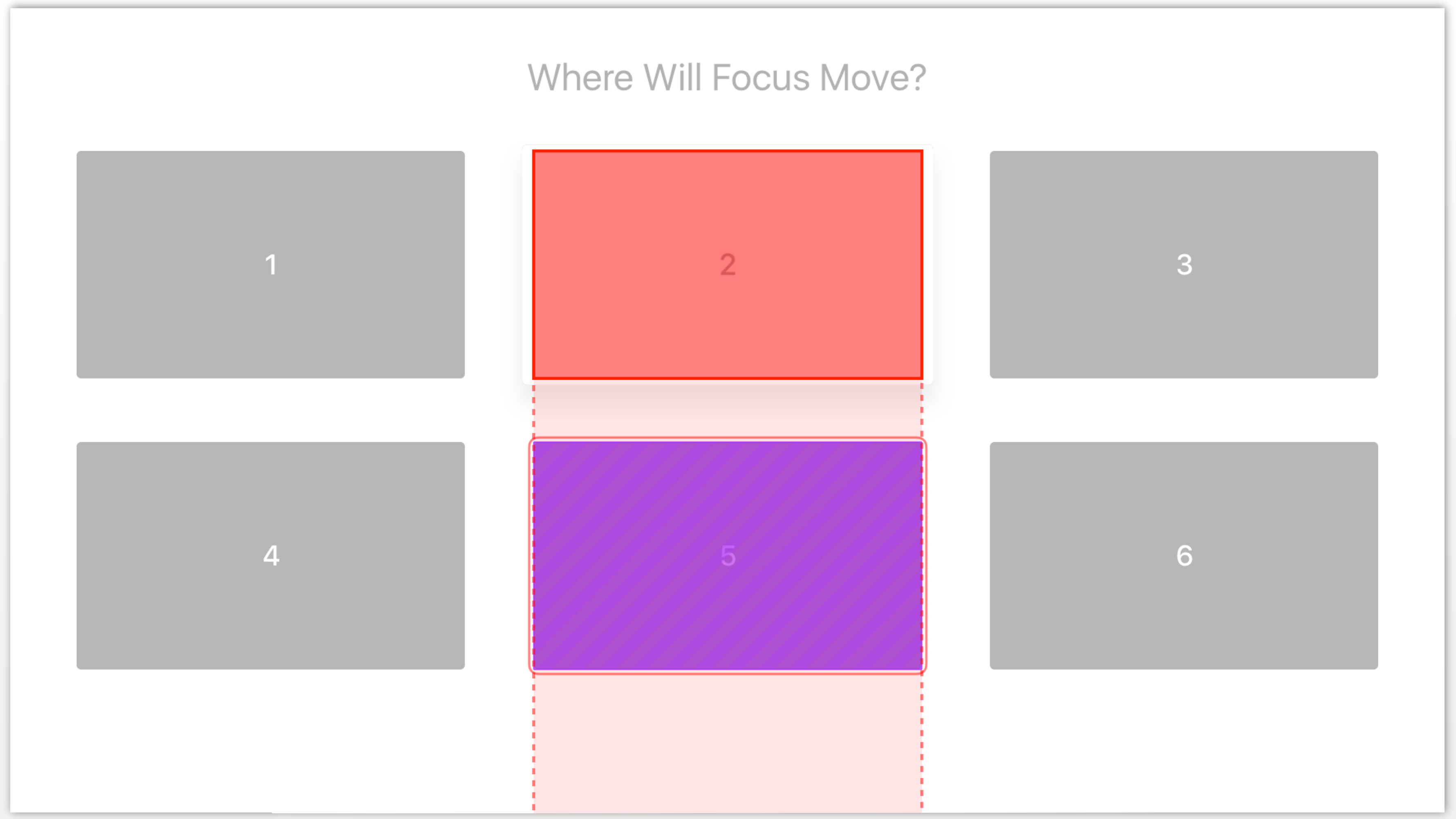
Controlling the UI with the Remote

- Interaction is indirect and based on the focus model
- You can ask for focus updates programmatically, but you **cannot set or move focus!**–Why?
- `UIButton`, `UITextField`, `UITableView`, `UICollectionView`, `UITextView`, `UISegmentedControl`, `UISearchBar` support focus by default
- Focus engine
  - Listens for incoming focus-movement events from input devices
  - Then automatically determines where focus should update and notifies the app
  - Communicates with your app via the `UIFocusEnvironment` protocol
  - Override UIFocusEnvironment methods in your view to control focus behavior

Focus Engine: Deciding Where to Move Focus

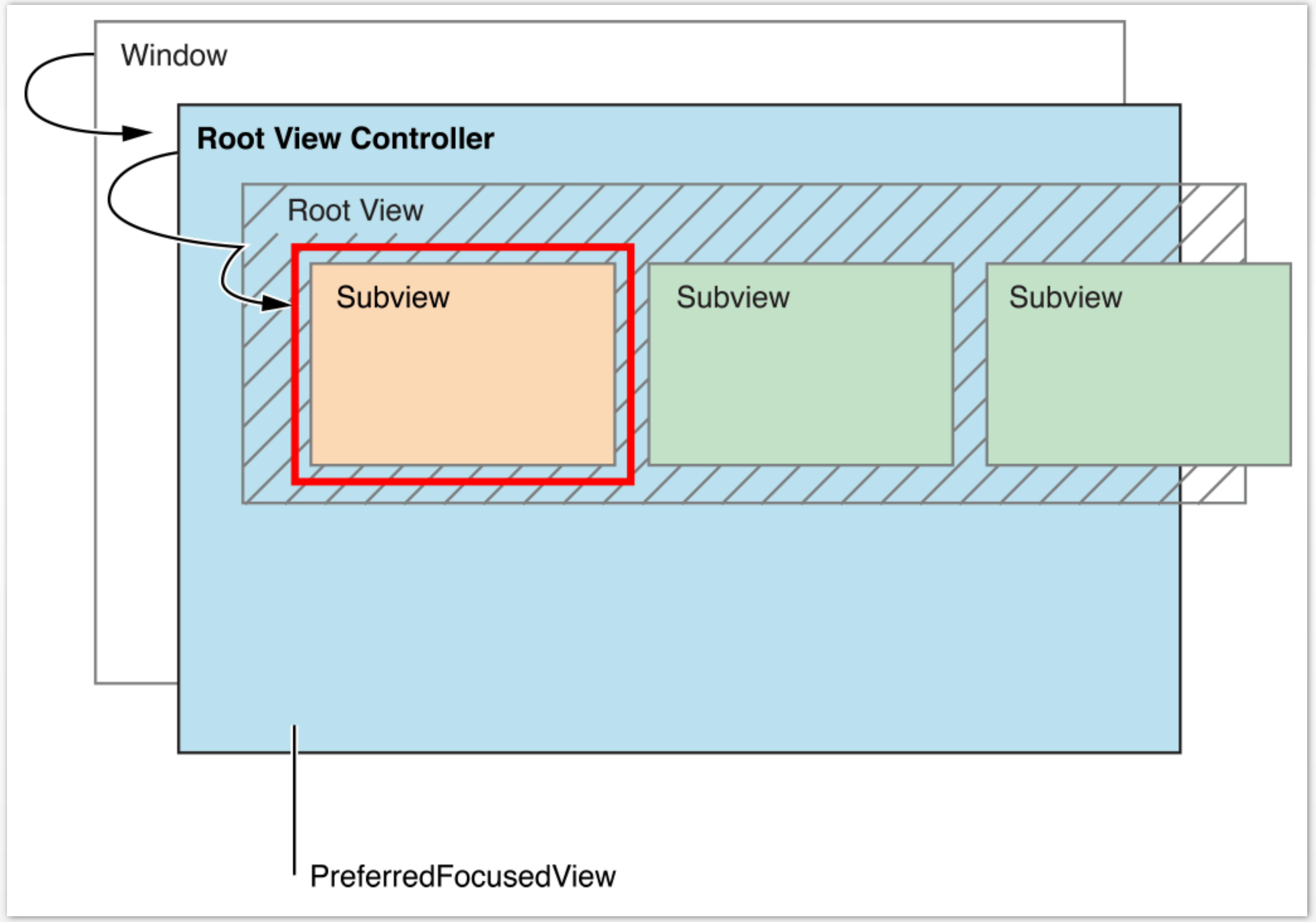
- Focus Engine (FE) takes **internal picture** of UI and will only consider all visible, focusable views
- FE starts from the currently focused view and finds any focusable regions in the **path of motion**
- The size of the search area is related to the size of the currently focused view
- Before focus moves, `shouldUpdateFocusInContext:` is called
  - If `false` is returned, the move is cancelled





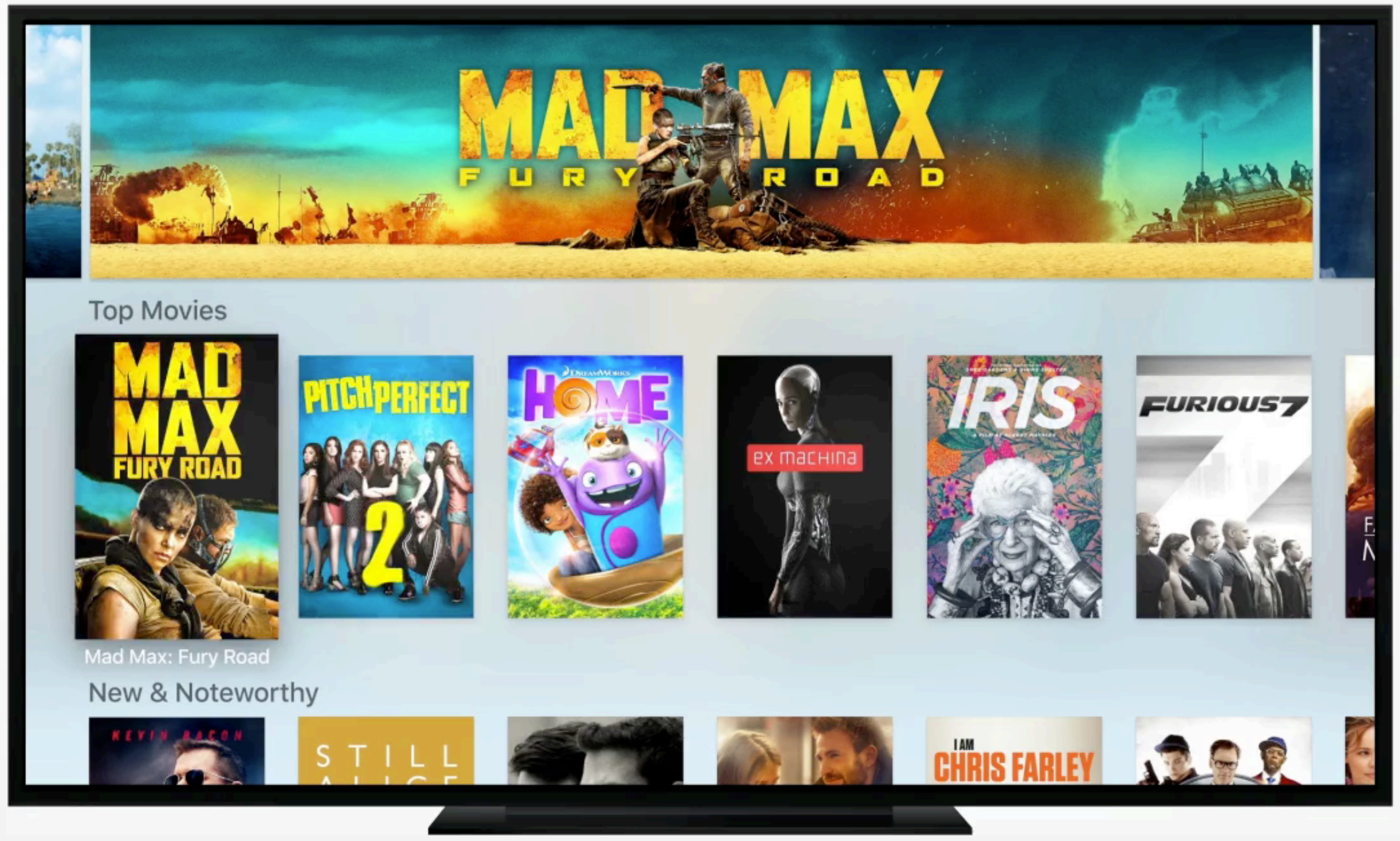
Focus Engine: Deciding Where to Move Focus

- By default the closest focusable view to the top-left corner is focused
- `UIFocusEnvironment` protocol
  - FE asks Window for preferred focus view → returns root view controller's `preferredFocusedView` object (which is a view and thus, again, conforms to `UIFocusEnvironment` protocol)
  - FE then asks the `UIView` object for its preferred focus view and so on
  - Focus chain ends when it reaches a view that returns nil or self
  - Hence, `UIViewController` controls focus-related behavior for its root view and descendants, and `UIView` controls focus behavior for itself and descendants



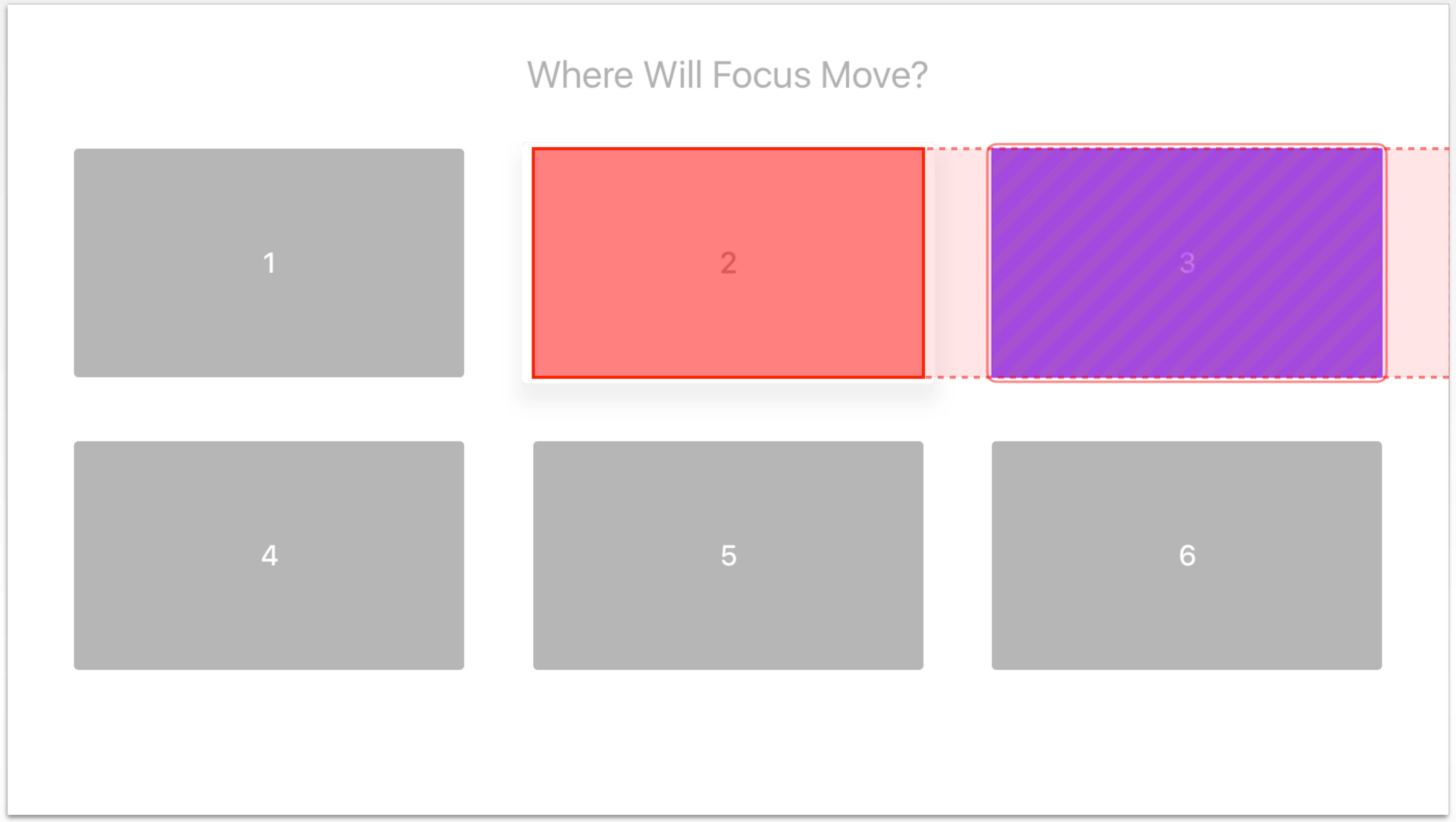
Focus Updates

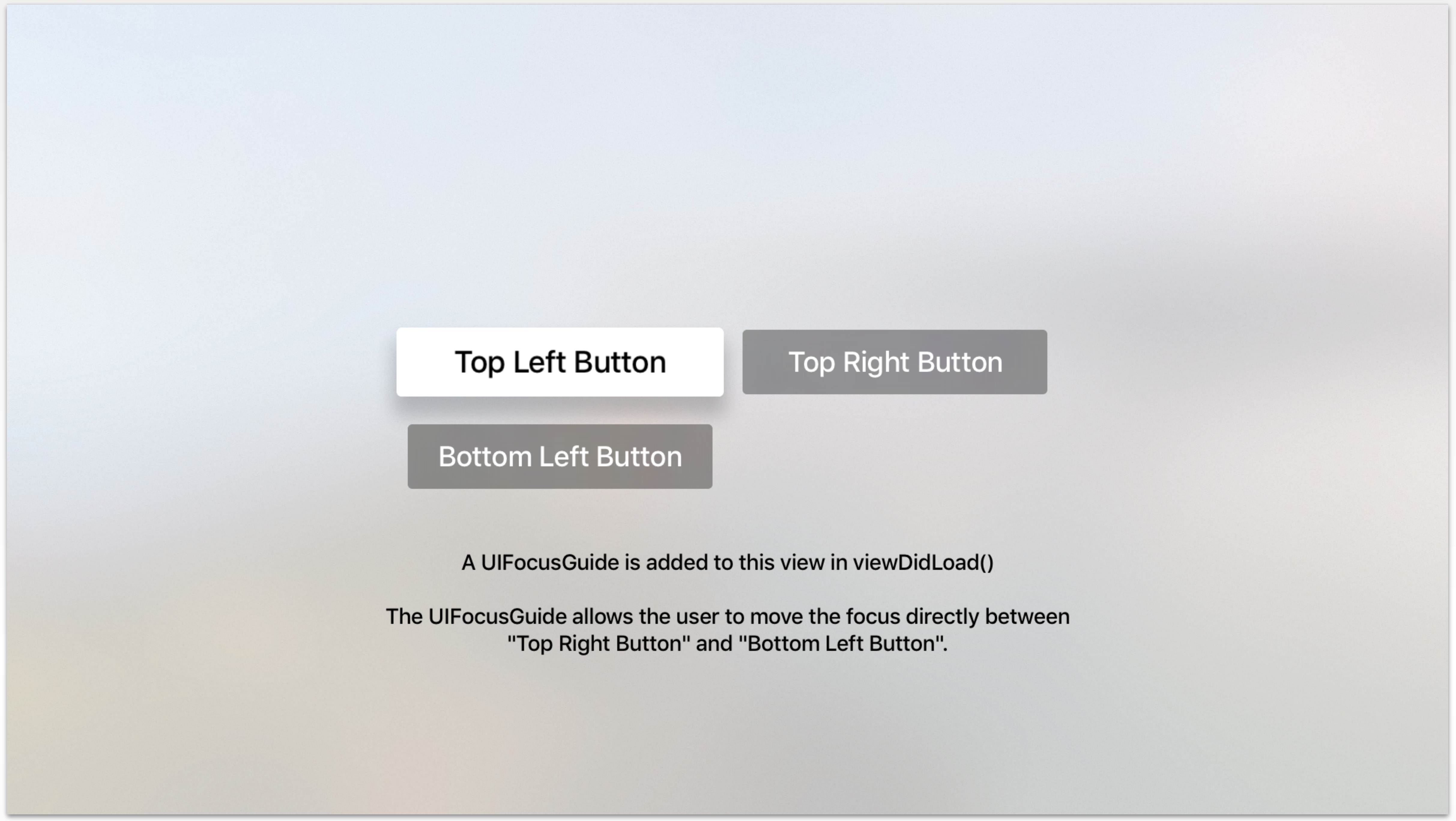
- System-generated focus updates
  - E.g., when a new view controller is presented over the currently focused view
- Updating focus programmatically
  - By calling `setNeedsFocusUpdate`, focus is reset to `preferredFocusView`
  - Then FE determines anew which item will be at focus–you can't set focus manually!
  - E.g., split view with menu items on the left, grid on the right: update focus to first item in the grid when new menu item is selected



Supporting Focus

- View Controllers
  - Override `preferredFocusView` to specify where focus should start by default
  - Override `shouldUpdateFocusInContext:` to define where focus is allowed to move
  - Override `didUpdateFocusInContext:…` to respond to focus updates when they occur
- Views
  - Override `canBecomeFocused` if your *custom* view needs to be focusable
    - E.g., a disabled button should not be focusable
  - Override `preferredFocusedView` to redirect focus, e.g., to a subview
  - Override `didUpdateFocusInContext:…` to respond to focus updates when they occur
- Collection Views and Table Views (delegates)
  - `collectionView:canFocusItemAtIndexPath:`
  - `tableView:canFocusRowAtIndexPath:`
  - `remembersLastFocusedIndexPath`





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".

Demos

Debugging: Where Will Focus Move?
Focus Guides: UIKitCatalog

Remote: Gestures & Button Presses

- `UIGestureRecognizer` and `UIResponder` classes include new methods to respond when buttons on the remote are pressed or released
- Supported gestures by the touch pad: pan, swipe
- Detecting the Play/Pause button (add code to your ViewController):

```
let tapRecognizer = UITapGestureRecognizer(target: self, action: "tapped:")
tapRecognizer.allowedPressTypes = [NSNumber(integer: UIPressType.PlayPause.rawValue)];
self.view.addGestureRecognizer(tapRecognizer)
```

- Detecting a swipe gesture (add code to your ViewController):

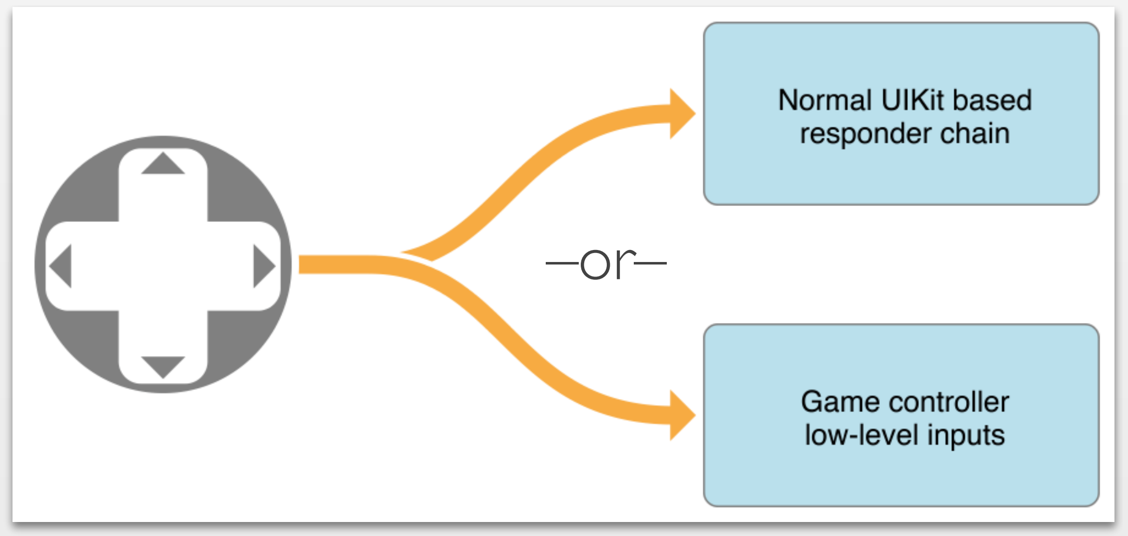
```
let swipeRecognizer = UISwipeGestureRecognizer(target: self, action: "swiped:")
swipeRecognizer.direction = .Right
self.view.addGestureRecognizer(swipeRecognizer)
```

Remote: Low-Level Event Handling

```swift
override func pressesBegan(presses: Set<UIPress>, withEvent event: UIPressesEvent?) {
    for item in presses {
        if item.type == .Select {
            self.view.backgroundColor = UIColor.greenColor()
        }
    }
}

override func pressesEnded(presses: Set<UIPress>, withEvent event: UIPressesEvent?) {
    for item in presses {
        if item.type == .Select {
            self.view.backgroundColor = UIColor.whiteColor()
        }
    }
}

override func pressesChanged(presses: Set<UIPress>, withEvent event: UIPressesEvent?) {
    // ignored
}

override func pressesCancelled(presses: Set<UIPress>, withEvent event: UIPressesEvent?) {
    for item in presses {
        if item.type == .Select {
            self.view.backgroundColor = UIColor.redColor()
        }
    }
}
```

Game Controllers

- Game controllers can be used as input device for any focus-based UI by default
    - For low-level input, use a `CGEventViewController`—here, events are not processed by UIKit
    - Use `controllerUserInteractionEnabled` property to toggle responder chains
- For low-level input, use the Game Controller framework; specifically for Apple TV:
    - `GCMicroGamepad` controller profile targets the capabilities of the Apple TV remote
    - `GCEventViewController` class can be used to control how controller and remote inputs are routed through the app
- Design requirements when using game controllers:
    - Your game **must support the Siri remote**
    - Your game must support the extended control layout when using a game controller
    - Games must be playable with standalone controllers
    - You must support the pause button, that will pause the game
        - In a menu, the pause button moves to the previous screen
- A maximum of two game controllers (plus the remote) can be connected to the Apple TV

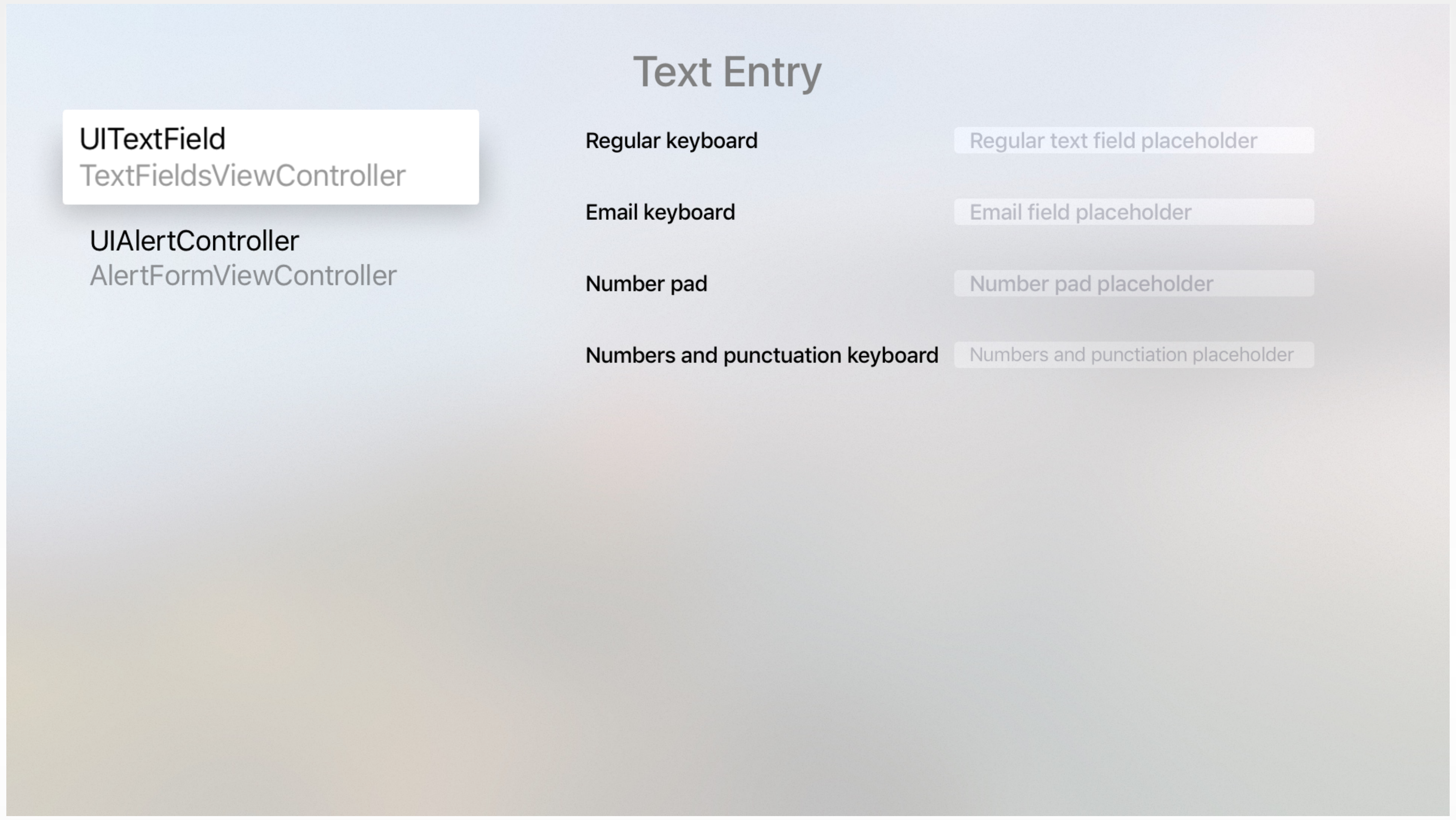

Normal UIKit based responder chain
–or–
Game controller low-level inputs

Remote as Game Controller

- Remote acts as a `CGController` object, supports `CGMotion` and `CGMicroGamepad` profiles
- Touchpad can be used as d-pad (provides analog input data)
- Touchpad is available as digital button "A"
- Play/Pause = button "X"
- Menu button is used to pause gameplay
- Remote can be used in portrait or landscape mode
- Remote **cannot determine its altitude or rotation**

Keyboard Input

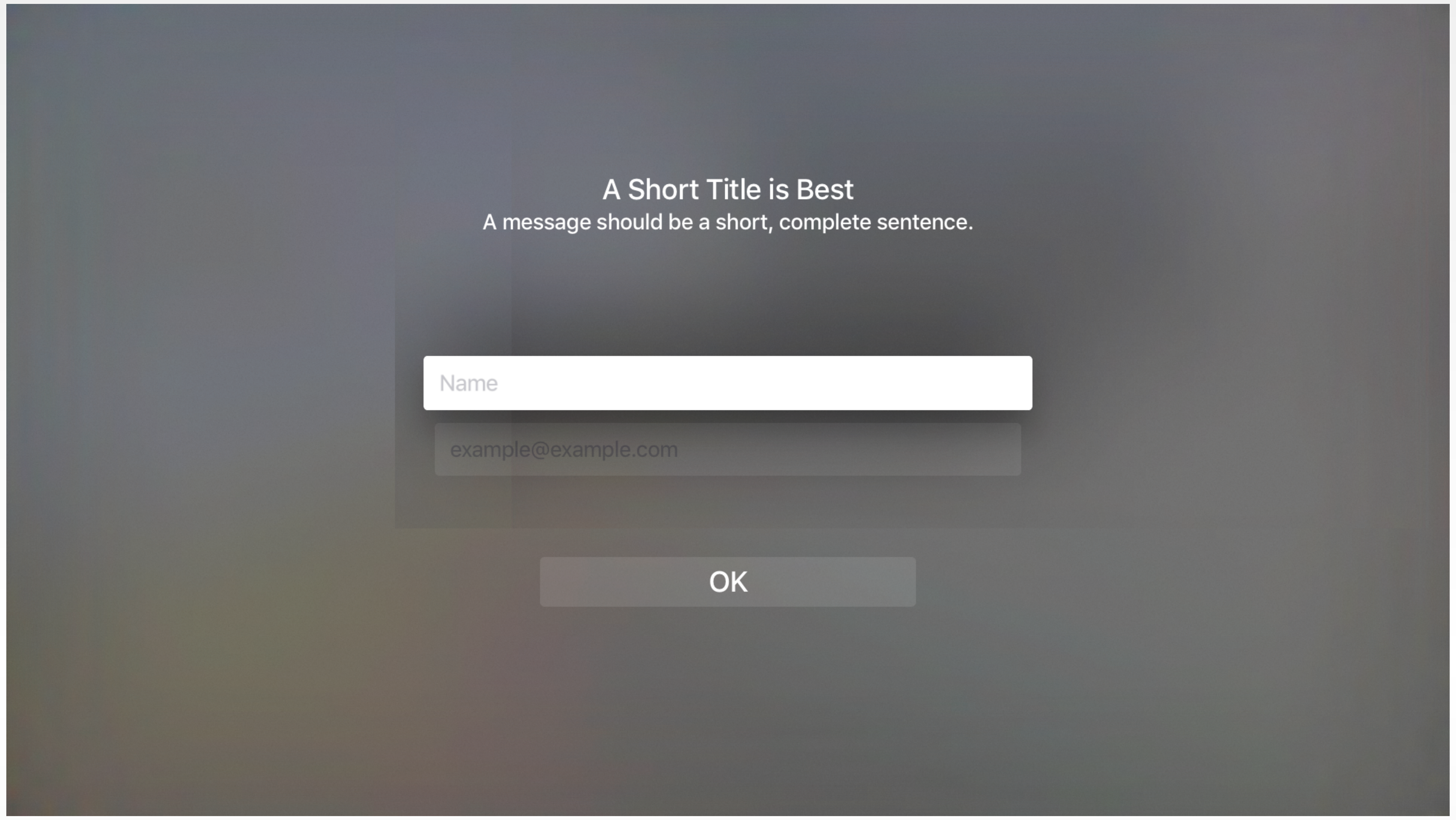
- When possible, **try to avoid text input** in your UI
  - e.g., write an iOS companion app, connect e.g., with Bonjour
- Two types of keyboards: **normal** and **inline**
- `UIAlertController`
  - Allows to add any number of text fields and buttons
  - However, the user has to click the remote a number of times to enter information
- `UITextField`
  - Places a full-screen keyboard on the screen
  - User navigates between text fields with Next and Previous buttons from the keyboard
  - More work for the developer: need to create and lay out all views
- Support for Bluetooth keyboards and Apple Remote App


Text Entry

UITextField
TextFieldsViewController

UIAlertController
AlertFormViewController

Regular keyboard	Regular text field placeholder
Email keyboard	Email field placeholder
Number pad	Number pad placeholder
Numbers and punctuation keyboard	Numbers and punctiation placeholder


A Short Title is Best
A message should be a short, complete sentence.

Name
example@example.com
OK

Demos

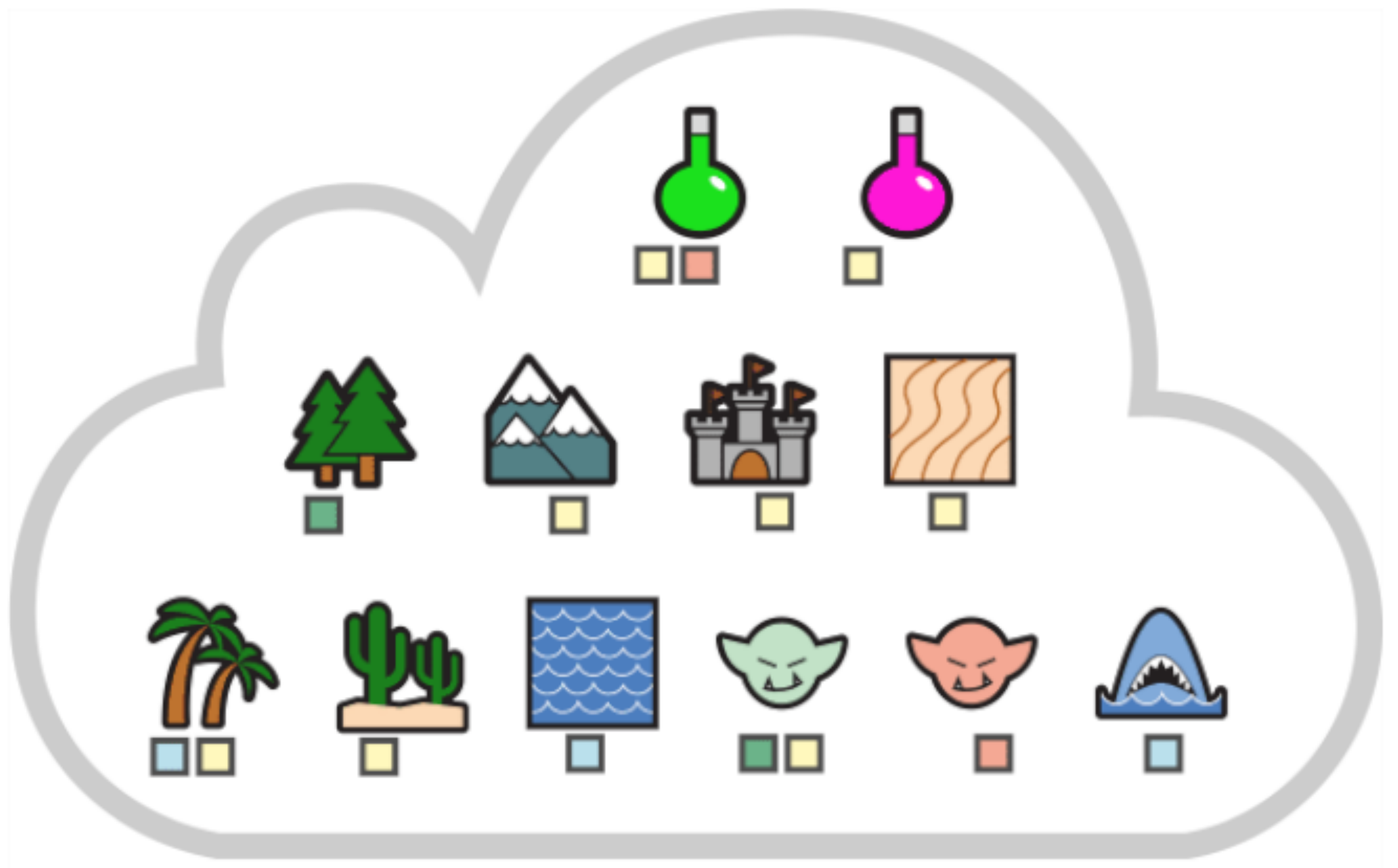
Keyboard Input with UIAlertController and UITextField: UIKitCatalog

On-Demand Resources (ODR)

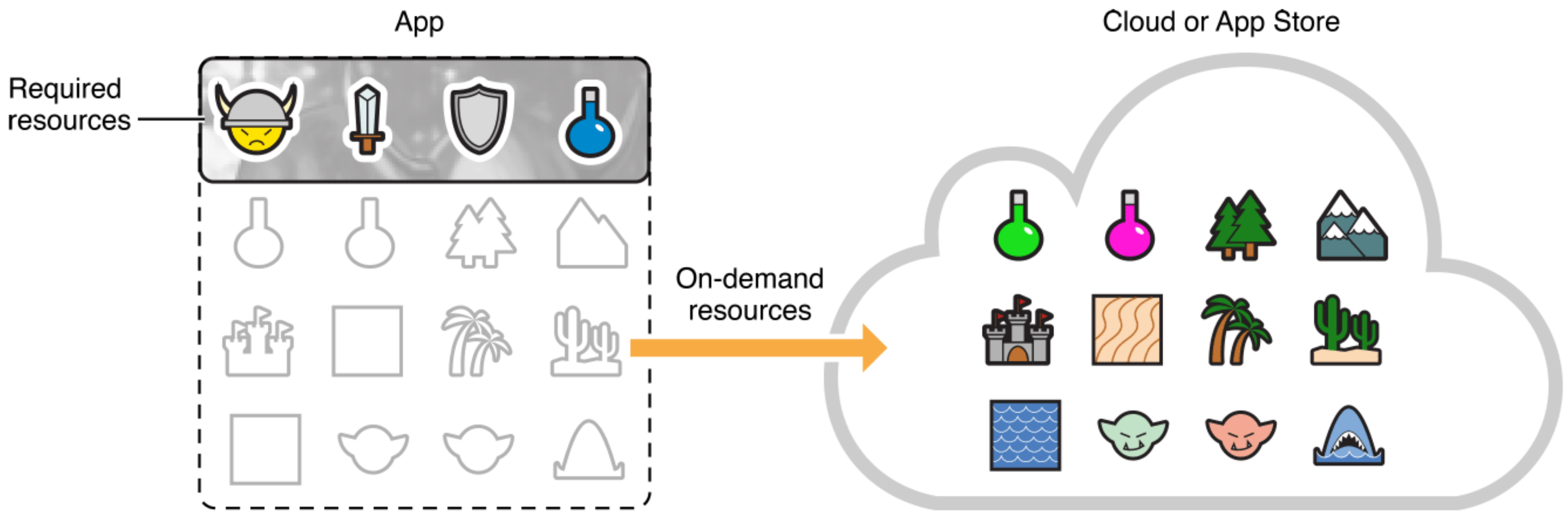
- ODR are app contents **hosted on the App Store**
- Enable smaller app bundles, faster download
- App requests ODR and tvOS/iOS manages download and storage
  - App contents may be **purged** by OS when the app is not in use!
- Maximum app size: 200 MB
  - You must use ODR to extend this limit
- Use Xcode to add **tags** to your assets
  - Assets will be downloaded on your request
  - To enable ODR in Xcode: Target → Build Phases → Assets → Enable On-Demand Resources: `Yes`
  - Assigning tags in Xcode: Target → Resource Tags or in the Attributes Inspector (for asset catalogs)

On-Demand Resources (ODR)

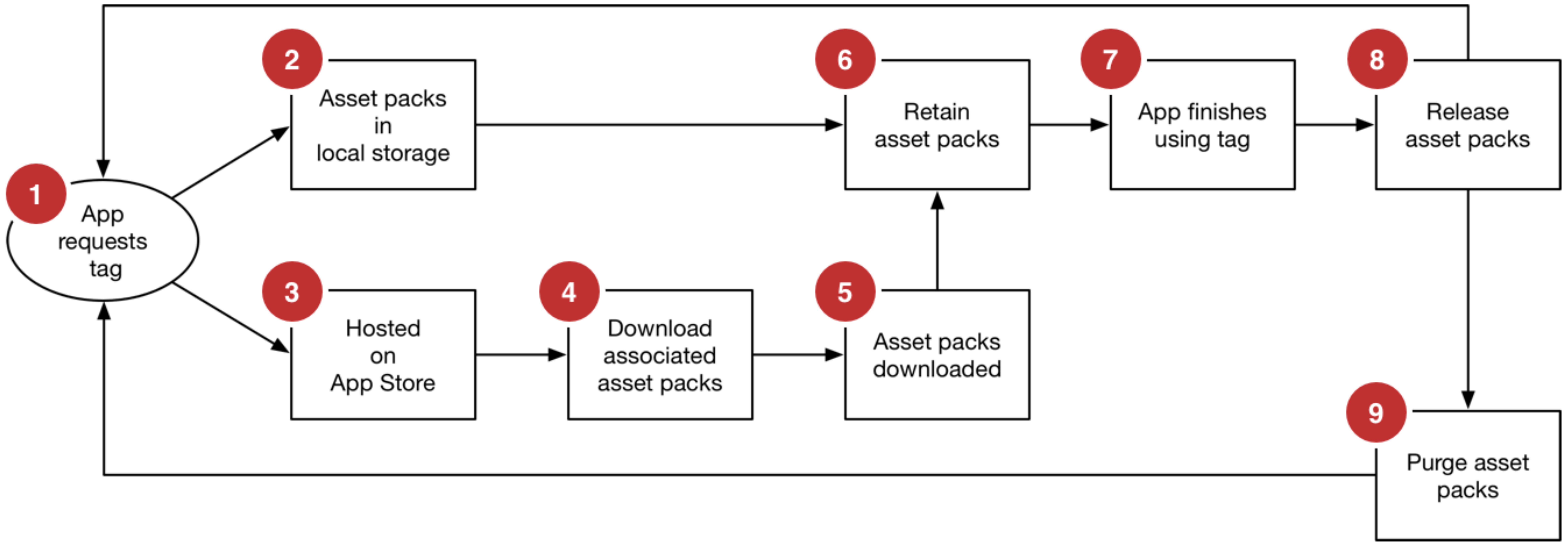
- Resources can be of any type supported by bundles **except code**
- ODR benefits
  - Smaller app size
  - Lazy loading of resources (e.g., level-based game)
  - Remote storage of rarely uses resources (e.g., app tutorial)
  - Remote storage of in-app purchase resources (e.g., "Avatar Design Studio Extension")
- ODR are identified and requested by String tags (e.g., `level-5`)
  - ODR are retained are retained in storage until app finished using them



Tags	
🟥	Level 1
🟨	Level 2
🟩	Forest
🟦	Ocean



ODR Life Cycle



ODR Tags

- Prefetching tags
  - For resources that are needed important the first time the app launches or soon after launch
  - Specify in Xcode
- Three prefetch categories
  - **Initial Install Tags**: Downloaded at the same time as the app (included in total size of app in App Store)
  - **Prefetched Tag Order**: Downloaded after the app is installed
  - **Download Only On Demand** (default): Downloaded when requested by the app

Sizes for ODR

Item	Size	Slicing?
iOS App bundle	2 GB	✔
tvOS App bundle	200 MB	✔
Tag	512 MB	✔
Asset packs	1000	✔
Initial install tags	2 GB	✔
Initial install and prefetched tags	4 GB	✔
In use on-demand resources	2 GB	✔
Hosted on-demand resources	20 GB	–

Accessing & Downloading ODR

- NSBundleResourceRequest
  - Request access to ODR
  - Inform OS when access is no longer needed
  - Update the priority of an ODR download
    - Set property `loadingPriority` (ranges from 0.0 to 1.0)
  - Track the progress of an ODR download
    - Use KVO for property `fractionCompleted`
  - Check for a notification of low disk space
    - Observe `NSBundleResourceRequestLowDiskSpaceNotification`

Accessing & Downloading ODR

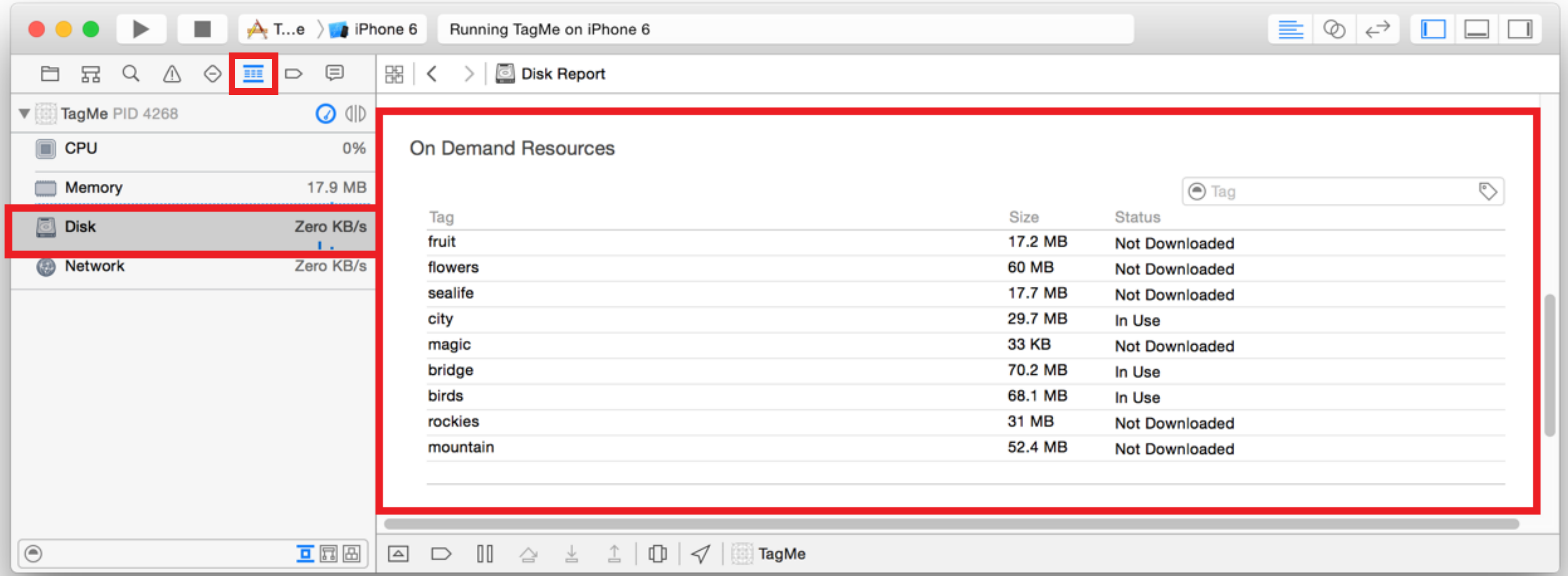
```
let tags : Set = ["birds", "bridge", "city"]

let resourceRequest = NSBundleResourceRequest(tags: tags)

resourceRequest.conditionallyBeginAccessingResourcesWithCompletionHandler {resourcesAvailable in

   if resourcesAvailable {

    // The resources are loaded, start using them
    }

    else {
    // The resources are not on the device and need to be loaded
    // Queue up a call to a custom method for loading the tags using
    // beginAccessingResourcesWithCompletionHandler:

       NSOperationQueue.mainQueue().addOperationWithBlock({

        // do something
       })
    }
}


resourceRequest.endAccessingResources()   // end access to a request
```

ODR Design Principles & Patterns

- Download what you need (64 MB **chunks**)
- **Download early** (While the user is playing Level 1, download Level 2)
- **Release resources** when you are done
- Optimize with **testing** (e.g., simulate different network bandwidths)

Pattern	Example	Design Approach
Random access	Browsing app	Many and small tags, progressive loading
Limited prediction	Open world game	Many and small tags, progressive loading, quickly ending access to unused tags
Linear progression	Leveled game	Download in advance, ending access to tags when done

Debugging ODR States



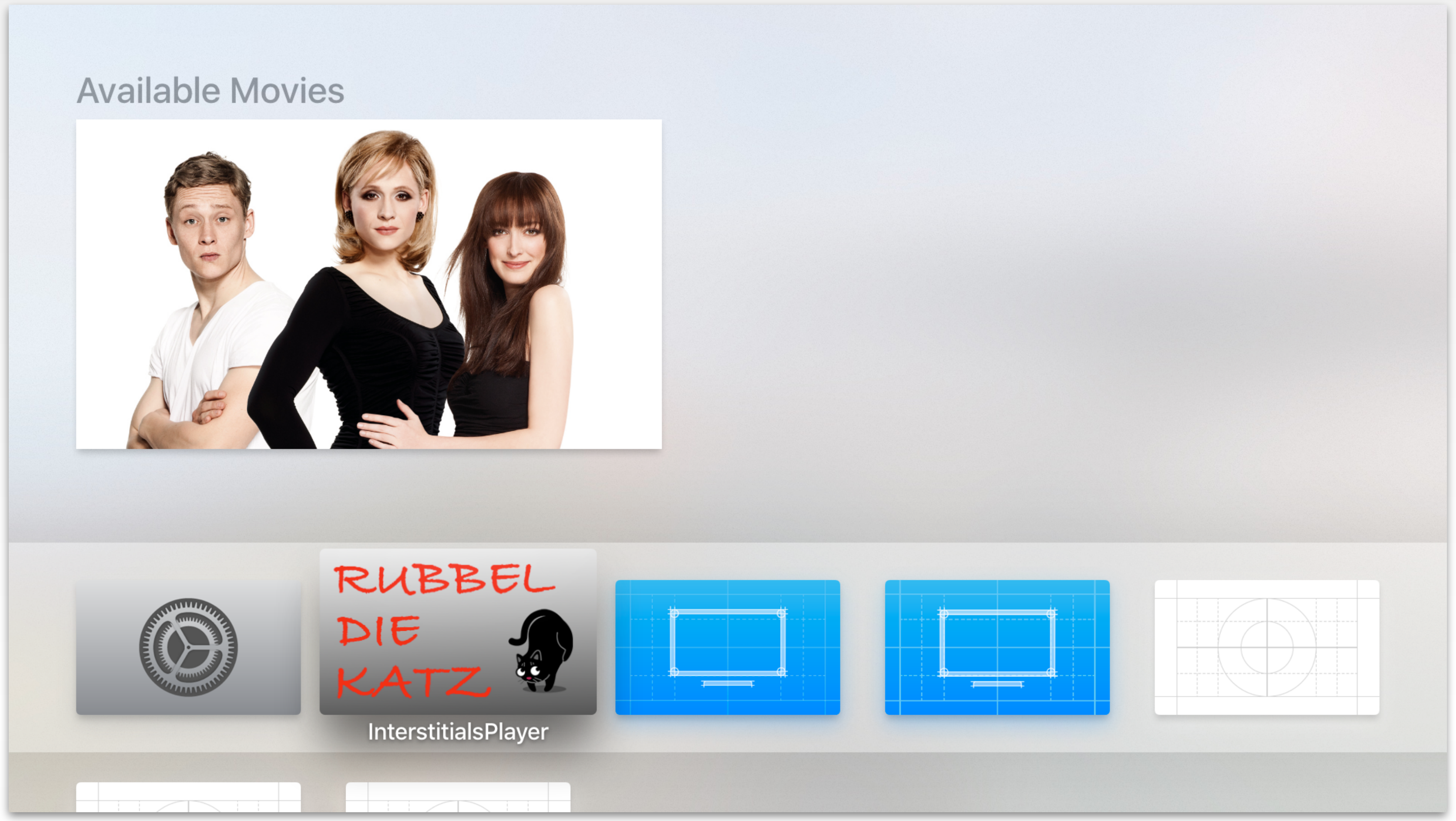
On Demand Resources

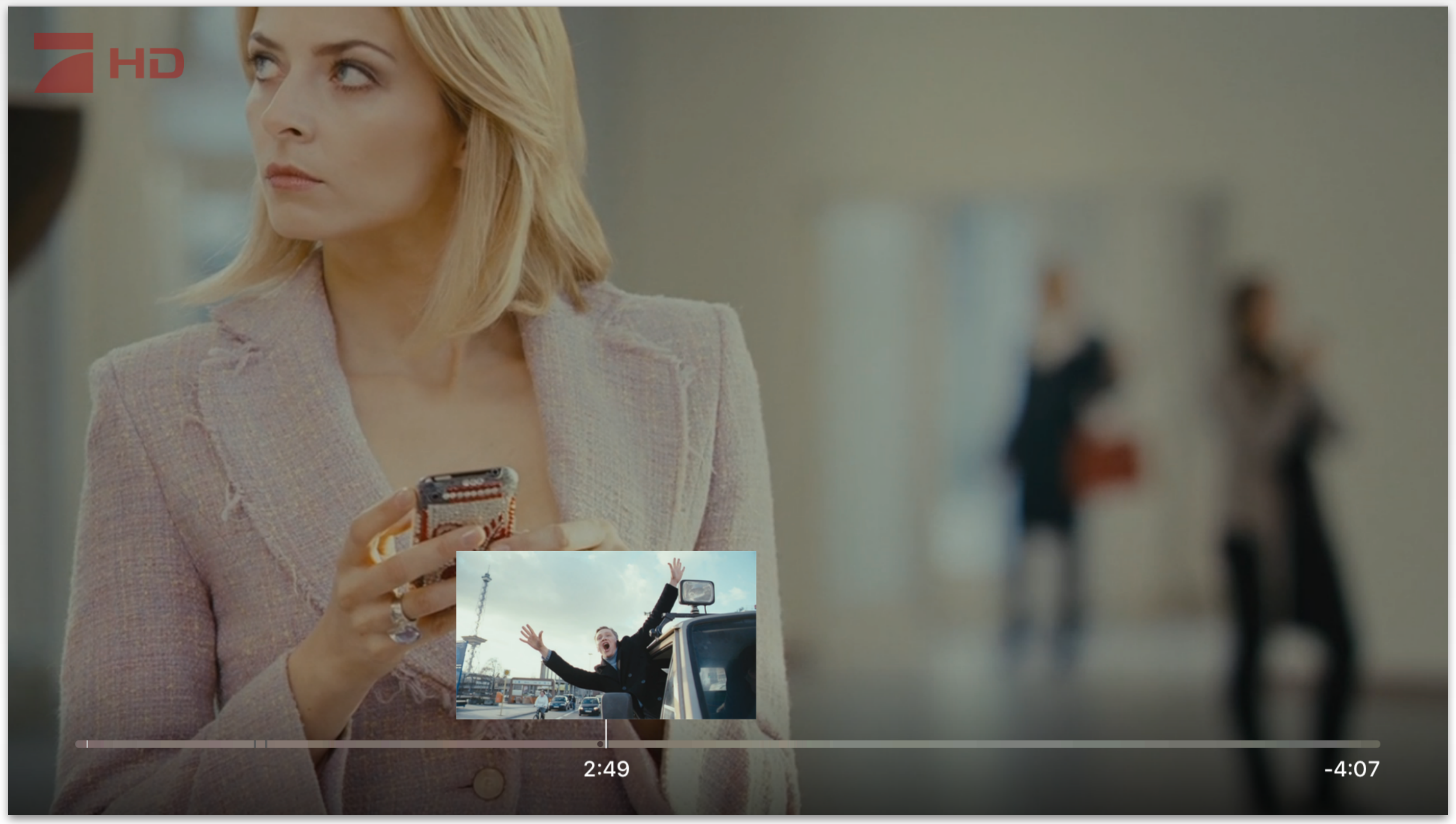
Tag	Size	Status
fruit	17.2 MB	Not Downloaded
flowers	60 MB	Not Downloaded
sealife	17.7 MB	Not Downloaded
city	29.7 MB	In Use
magic	33 KB	Not Downloaded
bridge	70.2 MB	In Use
birds	68.1 MB	In Use
rockies	31 MB	Not Downloaded
mountain	52.4 MB	Not Downloaded

Disk Gauge in Xcode (only in Debug Mode)

iCloud Storage

- There is no guarantee that information stored on the device will be available the next time a user opens an app!
- Two shared storage options
  - iCloud Key-Value Storage (KVS) vs. CloudKit
- When to use **KVS**:
  - Storage needs < 1MB
  - Only the owner of the app needs access to the information
  - KVS automatically synchronizes information across all of a user's devices
- When to use **CloudKit**:
  - Storage needs > 1 MB
  - Information is also accessible by another user (useful e.g., in games)





Demo App

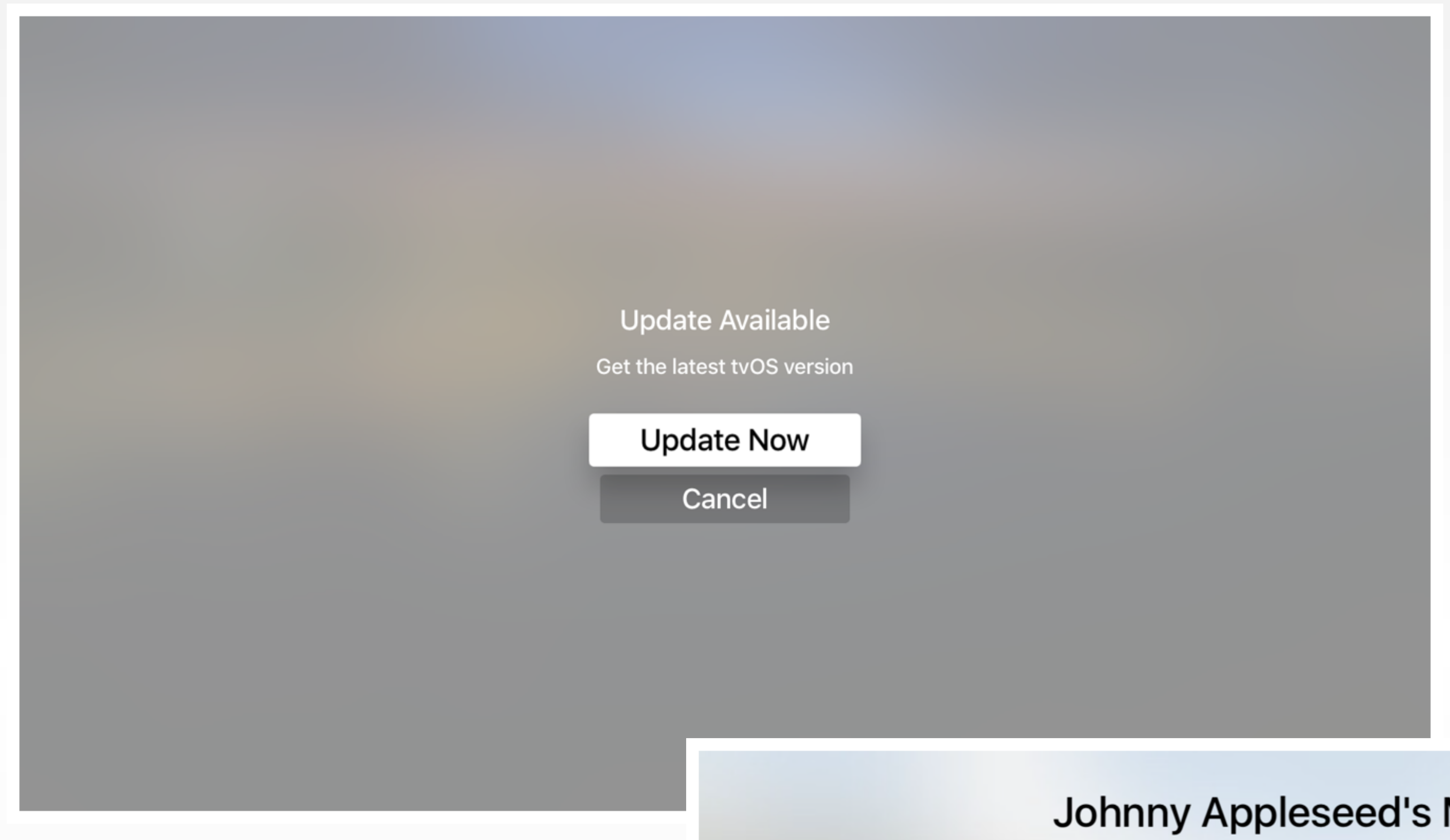
AVPlayer & Top Shelf

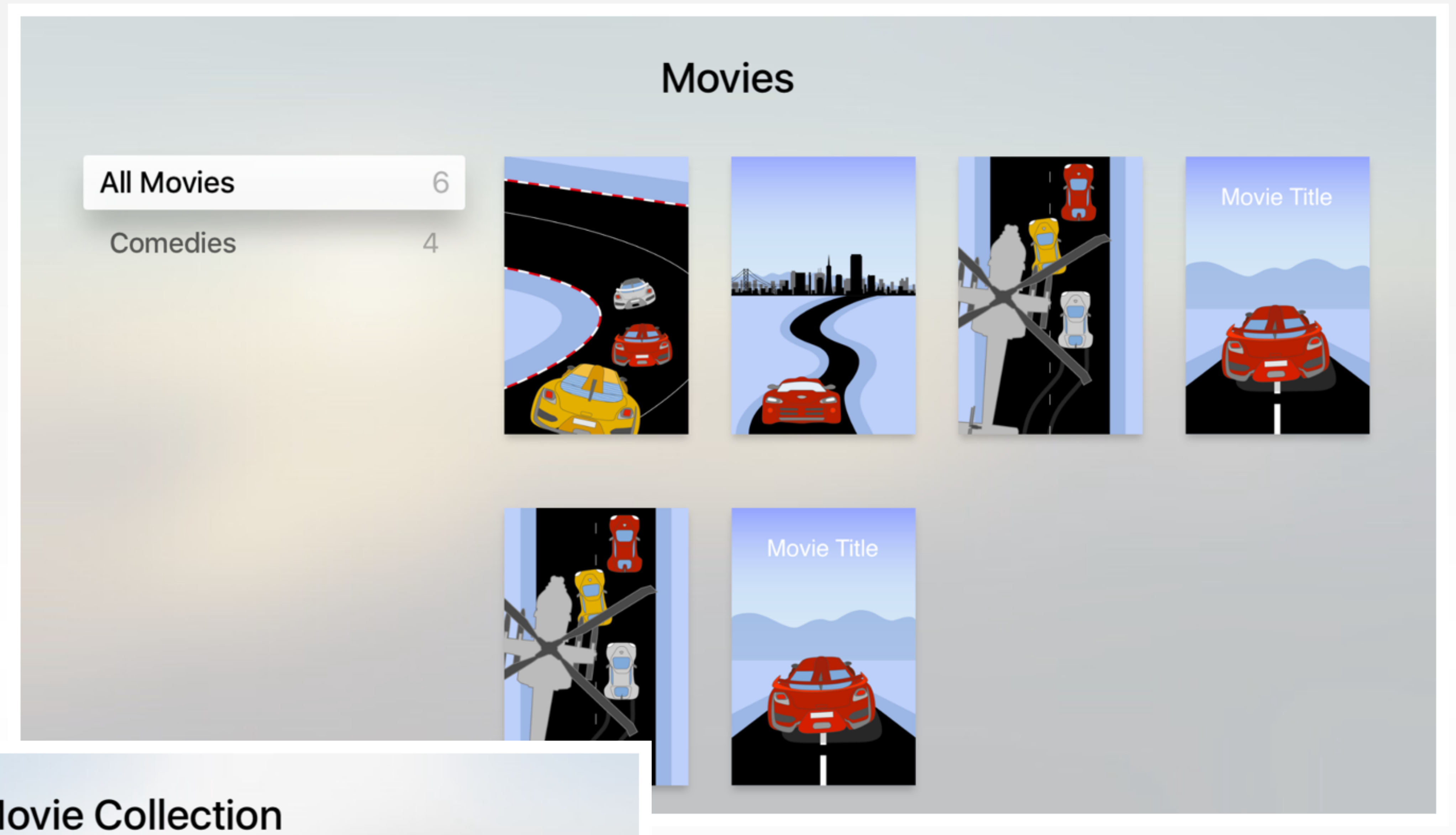
TVML Apps

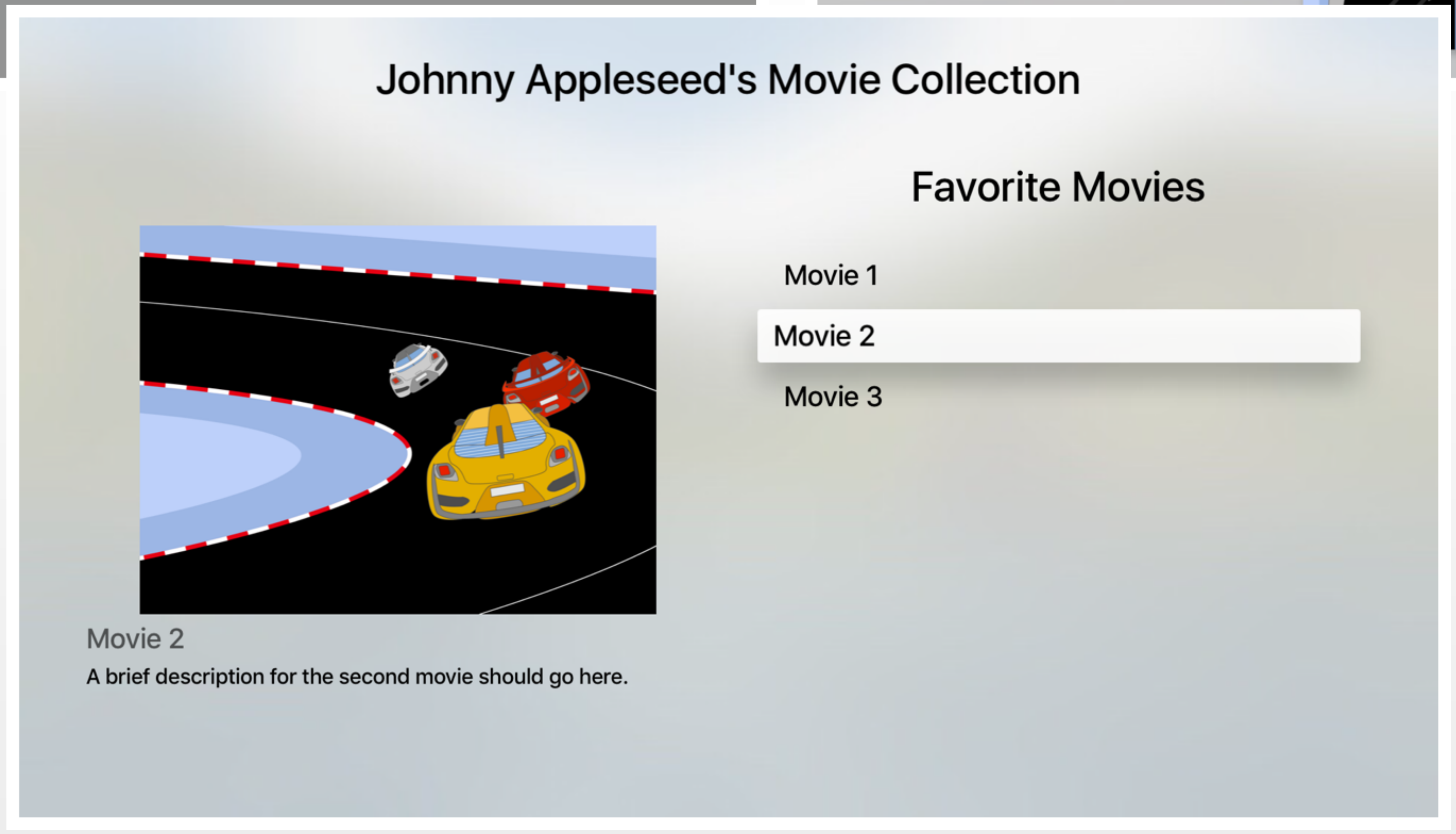
TVML Apps

- Use template to specify the layout of your app's screen(s)
- Use JavaScript to manage different screens, handle media content, handle events, manage memory, etc.
- Use case: Standard apps where the focus is on content and not the interaction

Examples: TVML App



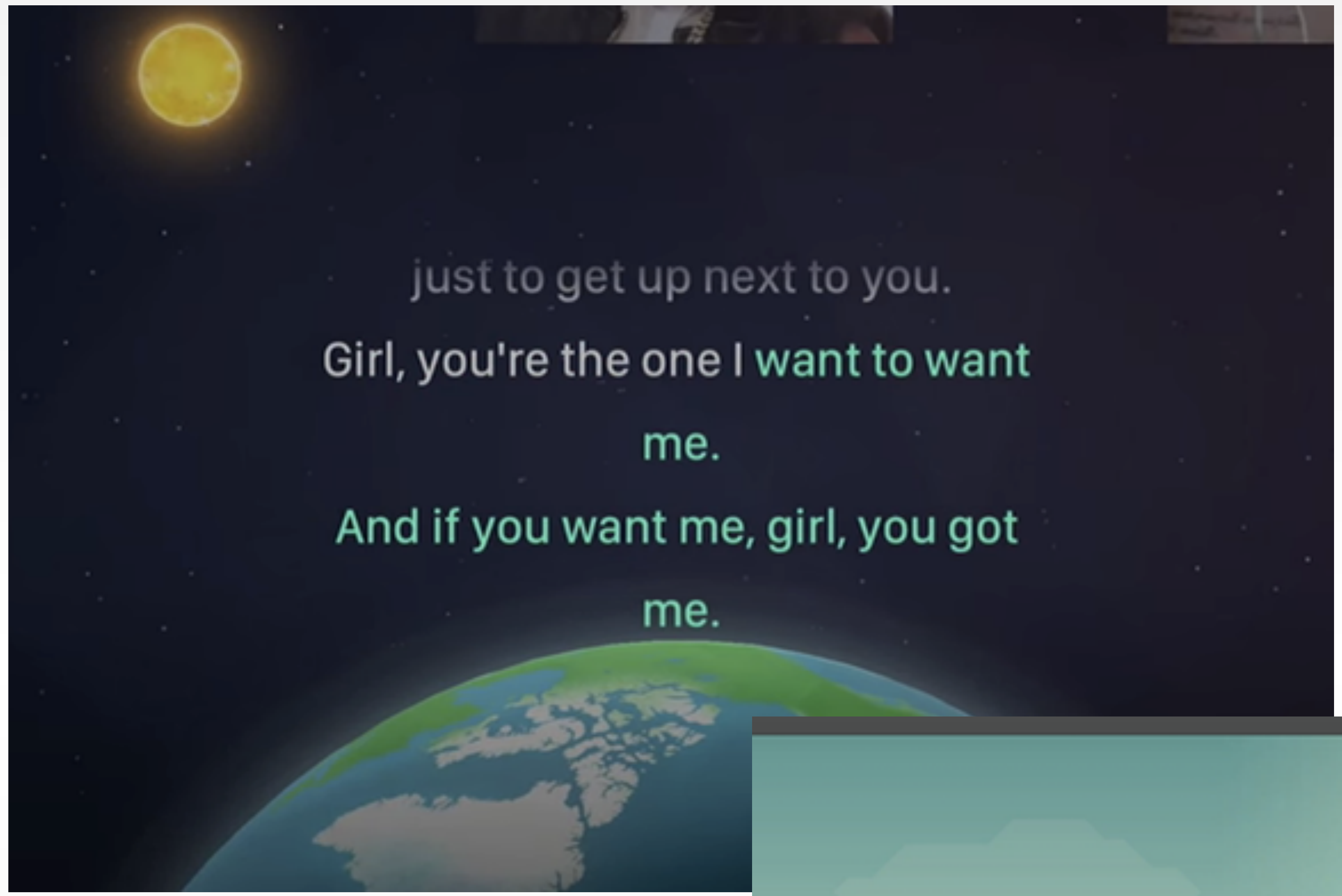


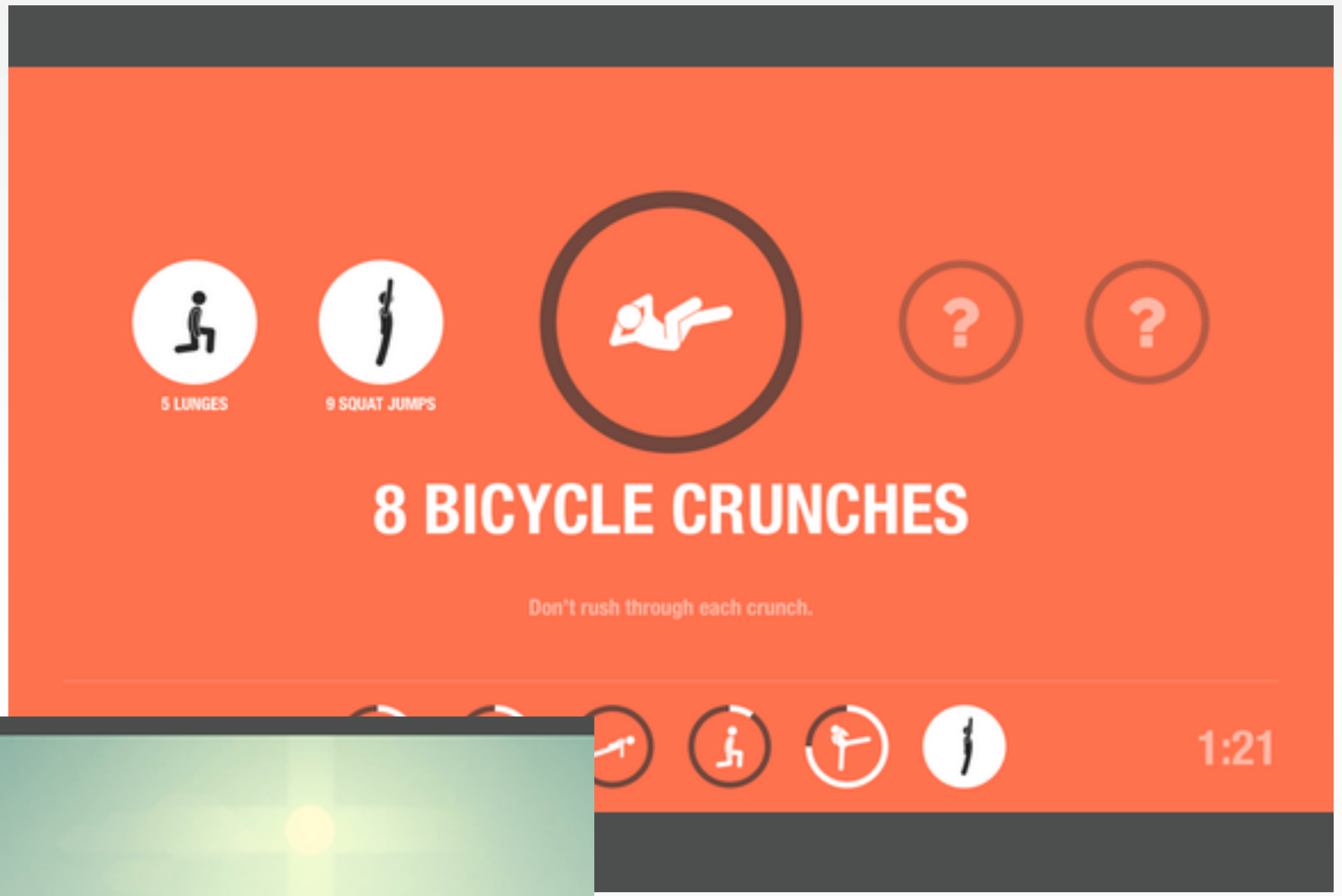


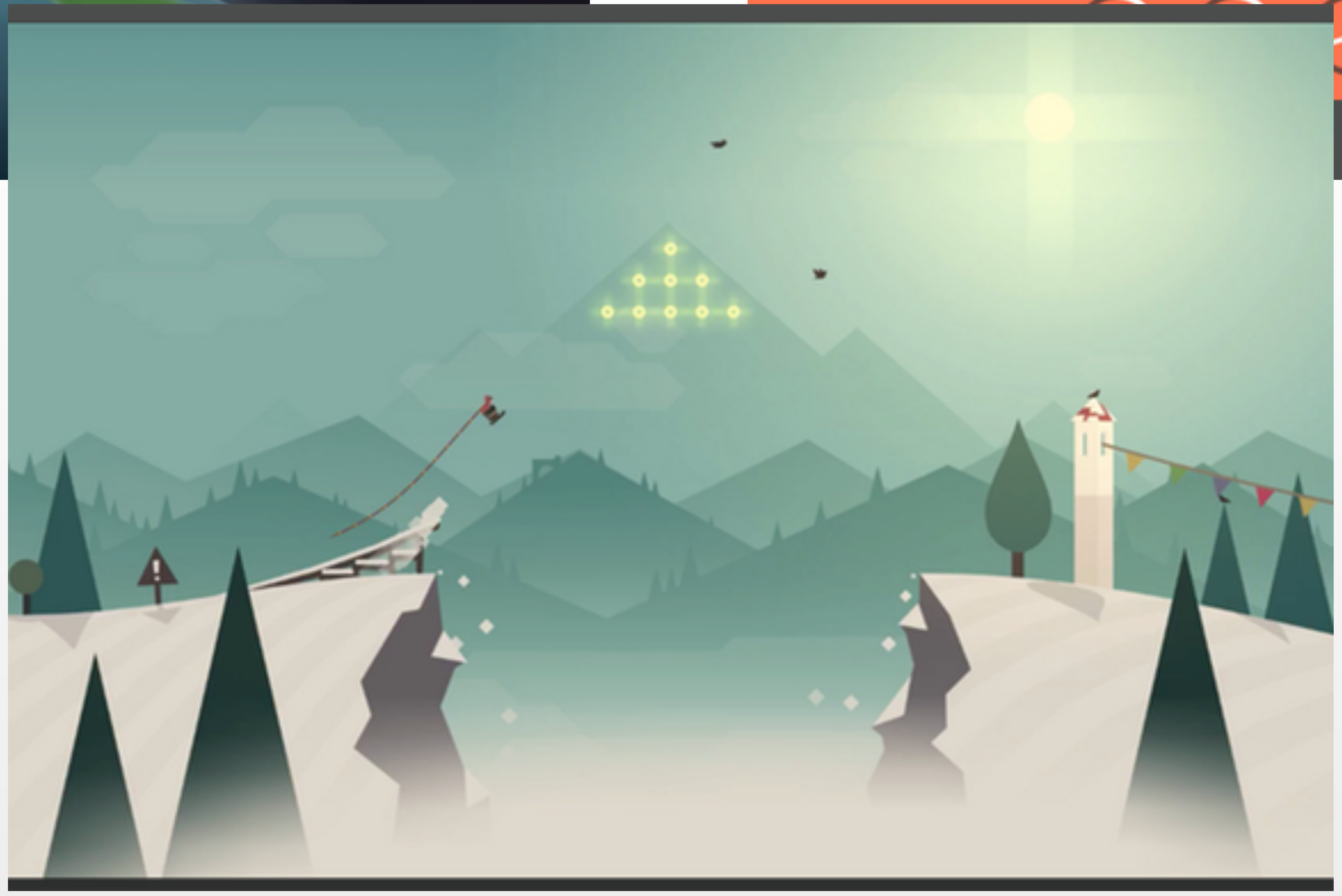
A brief description for the second movie should go here.

www.apple.com

Example: Custom Apps

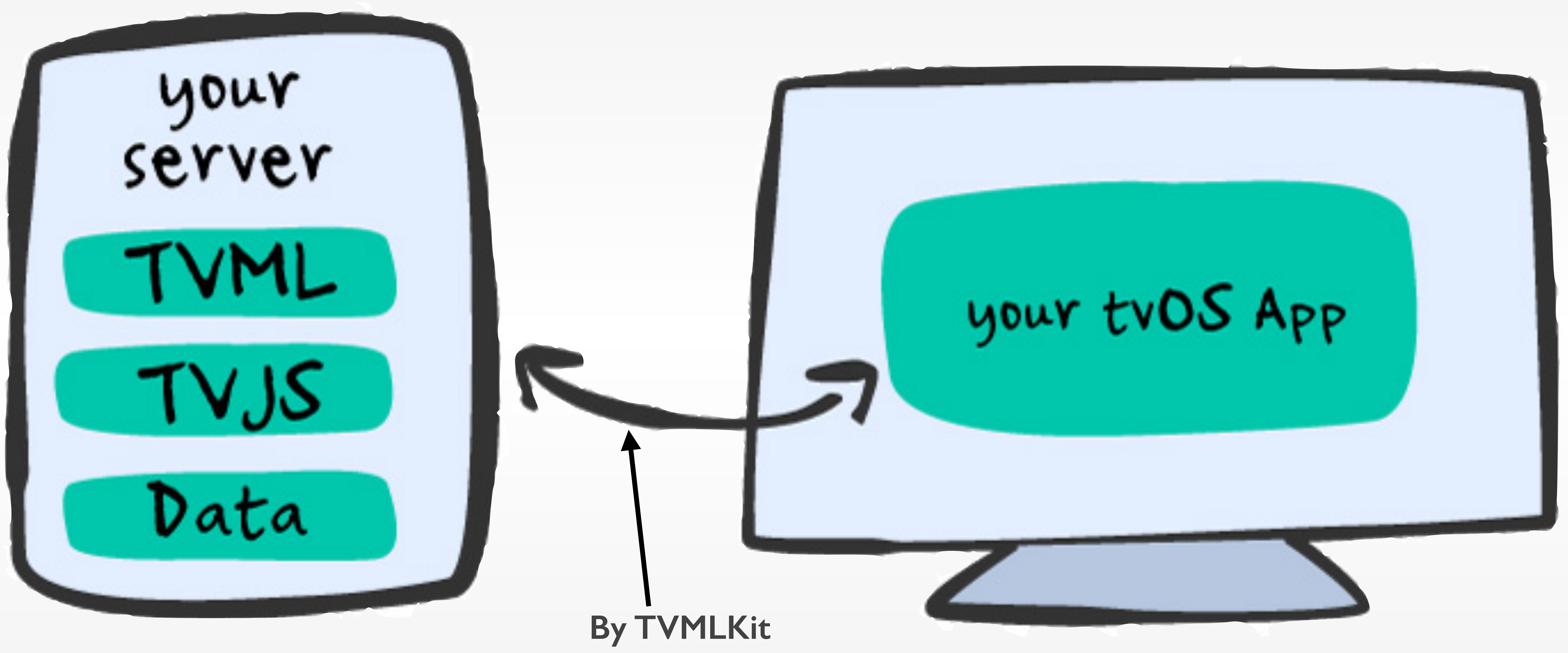

just to get up next to you.
Girl, you're the one I want to want me.
And if you want me, girl, you got me.





www.staticworld.net

Overview



www.raywenderlich.com

TVML

TVML

- TeleVision Markup Language
- A form of XML
- To specify the layout of your tvOS app's screen
- Each TVML specification is called a document

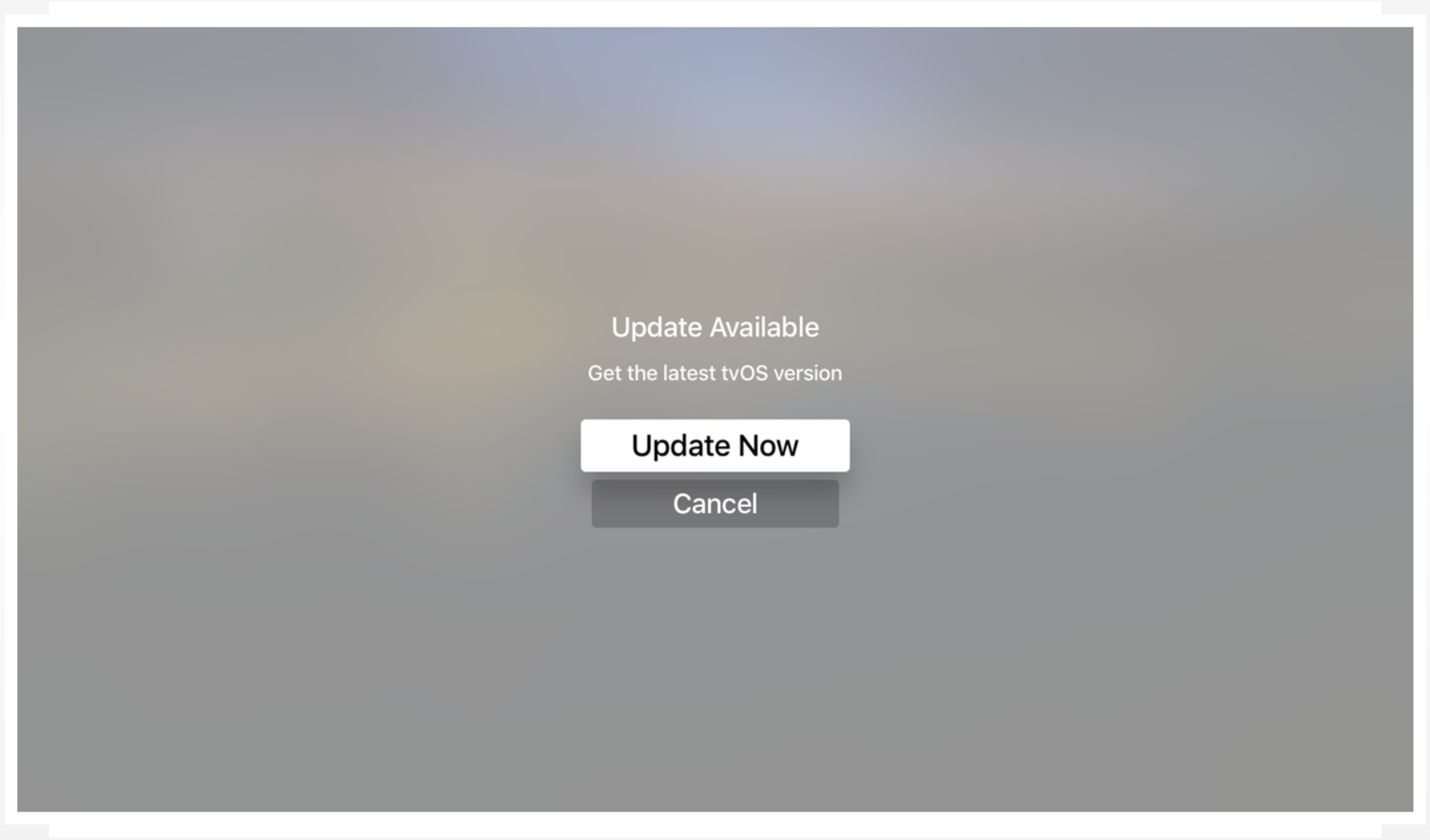
```
<student>
 <firstName>John</firstName>
 <lastName>Doe</lastName>
 <age>25</age>
 <nationality>United Kingdom</nationality>
 <gender>M</male>
</student>
```

TVML

- Can be specified as a separate XML file or in a JavaScript file
- Use one of the 18 templates that Apple offers
  - alertTemplate
  - catalogTemplate
  - searchTemplate
  - menuBarTemplate
  - …

Example: alertTemplate

```
<alertTemplate>
   <title>…</title>
   <description>…</description>
   <button>
      <text>…</text>
   </button>
   <text>…</text>
</alertTemplate>
```



TVJS

TVJS

- TeleVision JavaScript
- A set of JavaScript APIs to display/remove TVML documents, stream media, handle events,…
- Classes
  - App
  - NavigationDocument
  - XMLHttpRequest
  - Keyboard
  - Storage
  - …

App

- To respond to an app's life cycle events: onLaunch, onExit, onSuspend, onResume, onError
- onLaunch
  - Entry point
  - Load the first TVML document to be displayed on launch

TVMLKit

TVMLKit

- Connects your tvOS app to TVML and TVJS
- Steps
    1. Import TVMLKit
    2. Implement **TVApplicationControllerDelegate** to observe and manage the different states of tvOS app
    3. Setup launch options (server path, script path) and create an instance of **TVApplicationController**

In-Class Demo: TED Talks Viewer

Conclusion

- What sort of apps can you expect in the future?
    - Already: Games, Shopping, … and not just TV-based services
- Expect improvements to tvOS APIs (refinements to templates, bug fixes, etc.)
    - Better debugging tools (especially for TVJS)
- A new media streaming service like Netflix, Hulu, etc.? :)

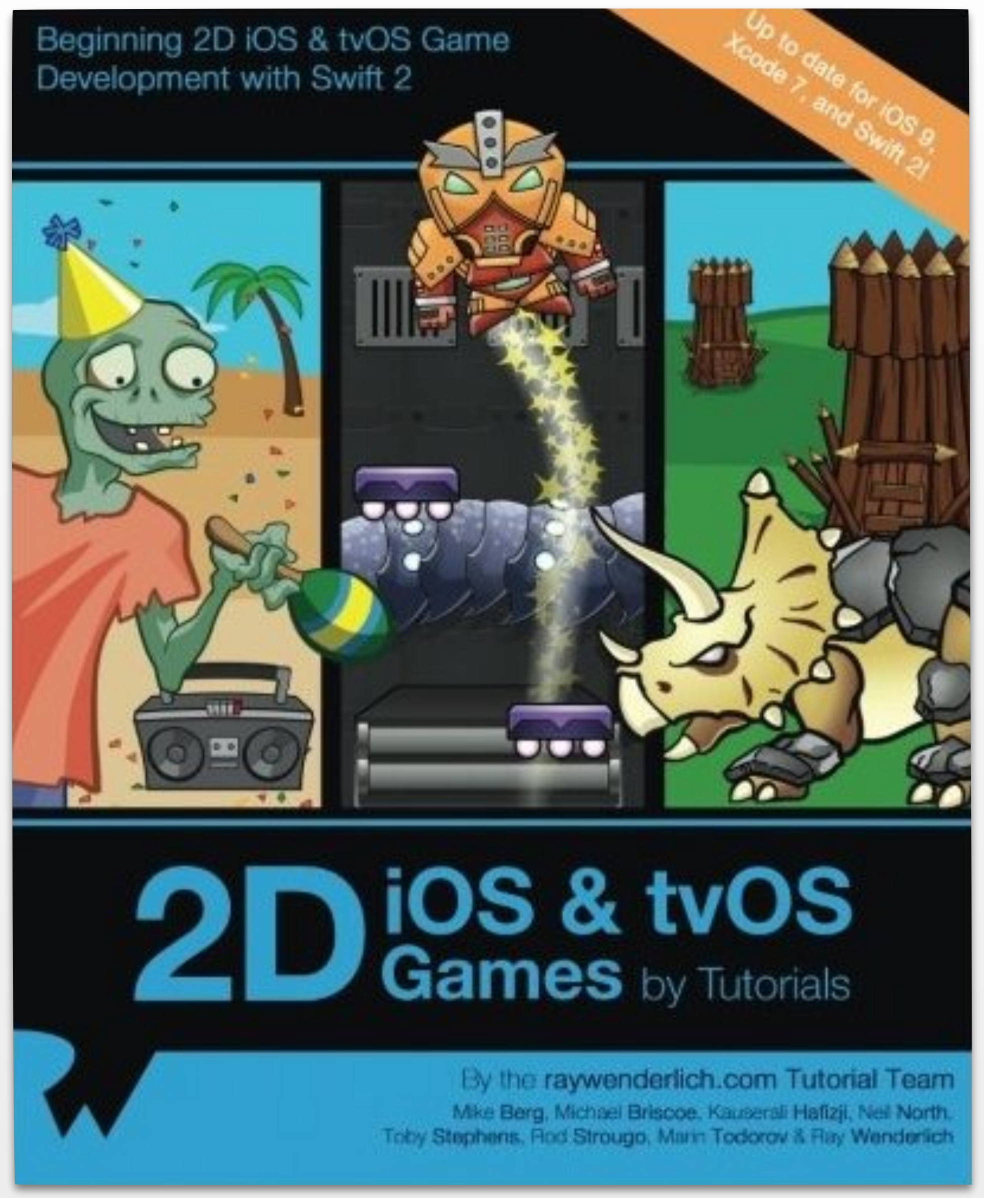
"'I'd like to create an integrated television set that is completely easy to use,' he told me. 'It would be seamlessly synced with all of your devices and with iCloud.' No longer would users have to fiddle with complex remotes for DVD players and cable channels. 'It will have the simplest user interface you could imagine. ***I finally cracked it.****'"*

- *"Steve Jobs"* by Walter Isaacson

Media Computing Group | RWTH AACHEN UNIVERSITY

Further Reading

- Apple TV Human Interface Guidelines
  - https://developer.apple.com/tvos/human-interface-guidelines
- App Programming Guide for tvOS
  - https://developer.apple.com/library/prerelease/tvos/documentation/General/Conceptual/AppleTV_PG/
- tvOS game development:
  - Sample code from Apple: DemoBots
  - 2D iOS & tvOS Games by Tutorials: Beginning 2D iOS and tvOS Game Development with Swift 2 by Ray Wenderlich et al.


Beginning 2D iOS & tvOS Game Development with Swift 2
Up to date for iOS 9, Xcode 7, and Swift 2!
2D iOS & tvOS Games by Tutorials
By the raywenderlich.com Tutorial Team
Mike Berg, Michael Briscoe, Kauserali Hafizji, Neil North, Toby Stephens, Rod Strougo, Marin Todorov & Ray Wenderlich