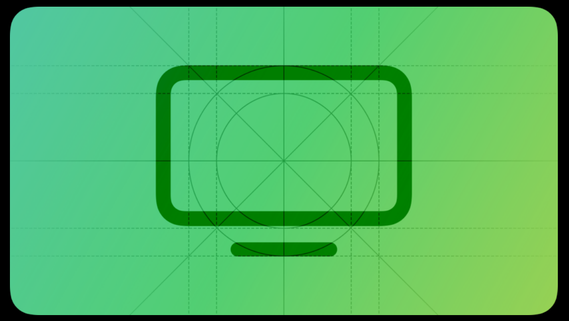


Focus sur tvOS

Intro à tvOS

Spécificités

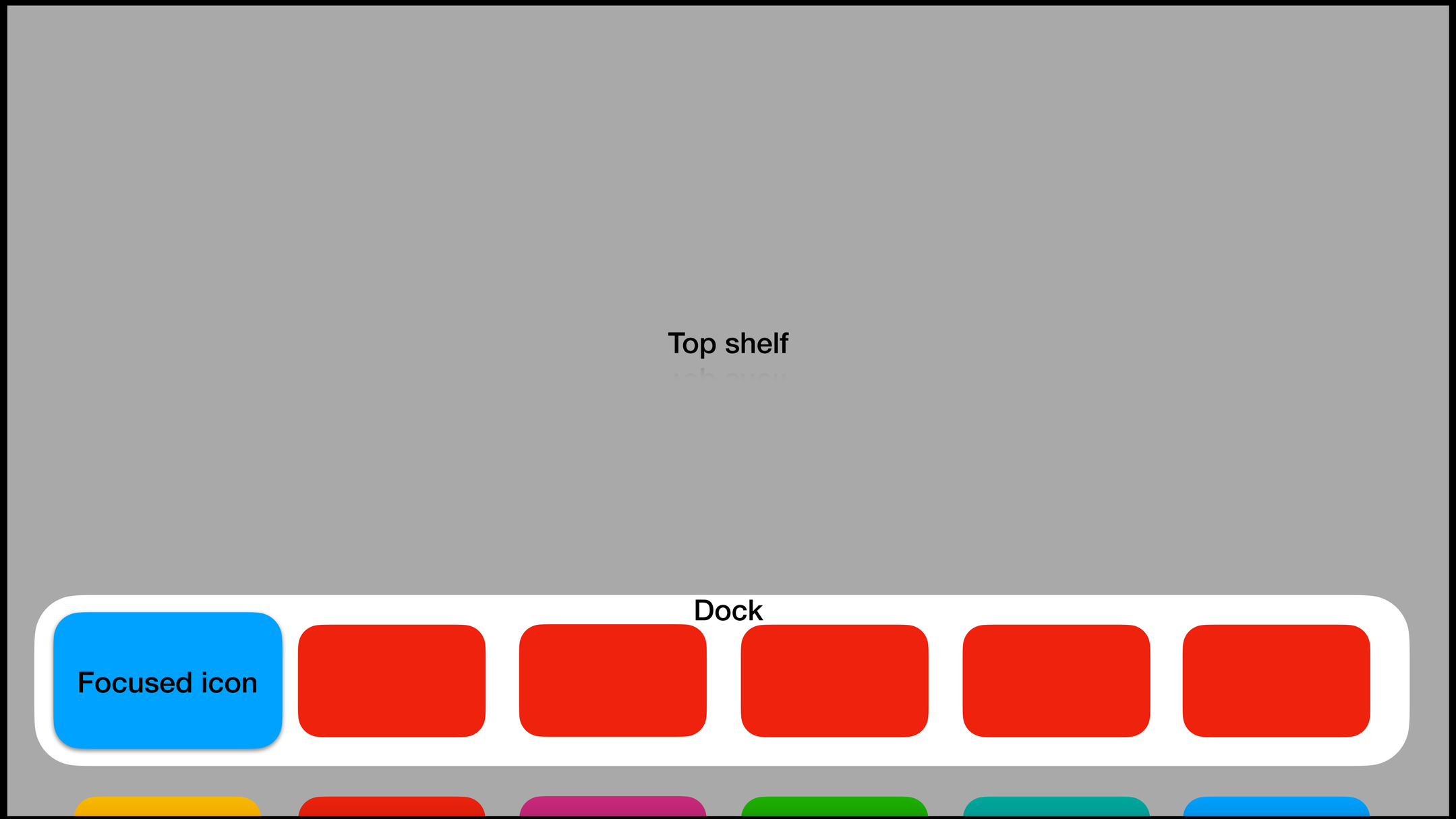
- Utilise très certainement le plus grand écran du foyer
- S'utilise de loin, parfois même en mouvement
- Se contrôle via une télécommande, une manette de jeu, la voix, ou une application sur un device iOS
- Interactions longues, qui se comptent parfois en heures

Types d'app

- tvOS supporte différents types d'app
  - App traditionnelles, comme sur iOS
    - UIKit est le framework d'origine, complété par TVUIKit
  - App Client-Serveur utilisant TVMLKit
    - Utilise du JS

tvOS Anatomy


3:02 PM
tv


Top shelf
Dock
Focused icon

tvOS Anatomy

- Top shelf
  - Affiche une image ou un contenu dynamique mis en avant
- Dock
  - Permet un accès rapide aux apps les plus utilisées
  - Seules les apps du Dock affichent du contenu en top shelf
- App mosaic
  - Affiche toutes les autres apps et dossiers d'apps

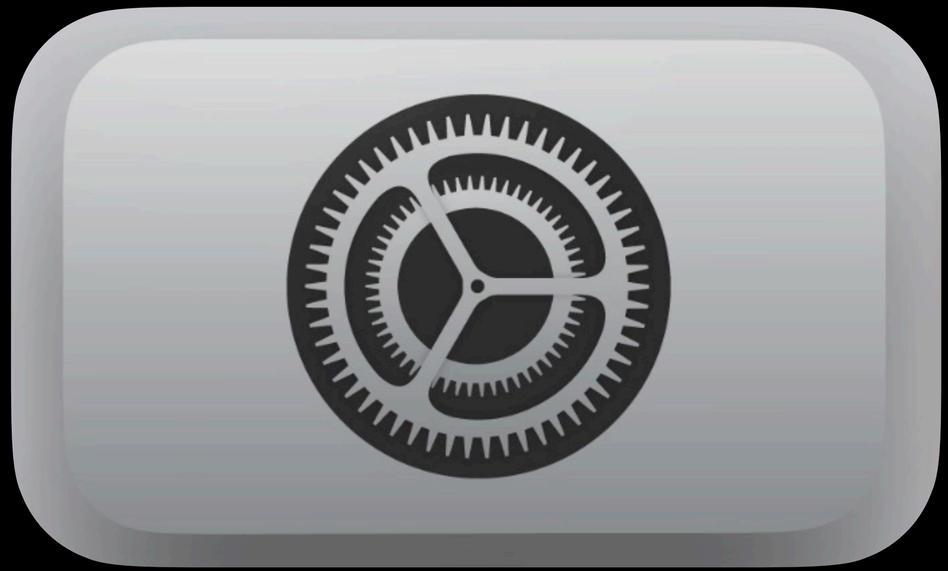
Ressources graphiques

Graphisme

- Comme iOS, on retrouve la notion d'images @1x et @2x
  - Les TV HD utilisent les assets @1x
  - Les TV 4K utilisent les assets @2x
- tvOS utilise des effets de soulèvement et de parallaxe pour représenter le focus

Icône de l'app

- Utilise une layered icon
- Entre 2 et 3 layer
- 400 x 240 points
- Les layers se déplacent en fonction du focus avec un effet de parallaxe



Icône marketing

- Utilisée pour l'App Store
- 1280x768 px

Top Shelf Image

- S'affiche dans le Top Shelf par défaut
- 2320x720 pt

Top Shelf

"The Apple TV Home Screen provides an area called Top Shelf, which showcases your content in a rich, engaging way while also giving people access to their favorite apps in the Dock."

Apple Human Interface Guidelines

Top Shelf

- Affiche une image statique ou un contenu dynamique
  - Image statique fournie dans le Asset Catalog
  - Contenu dynamique fourni par une Top Shelf Extension

Top Shelf Extension

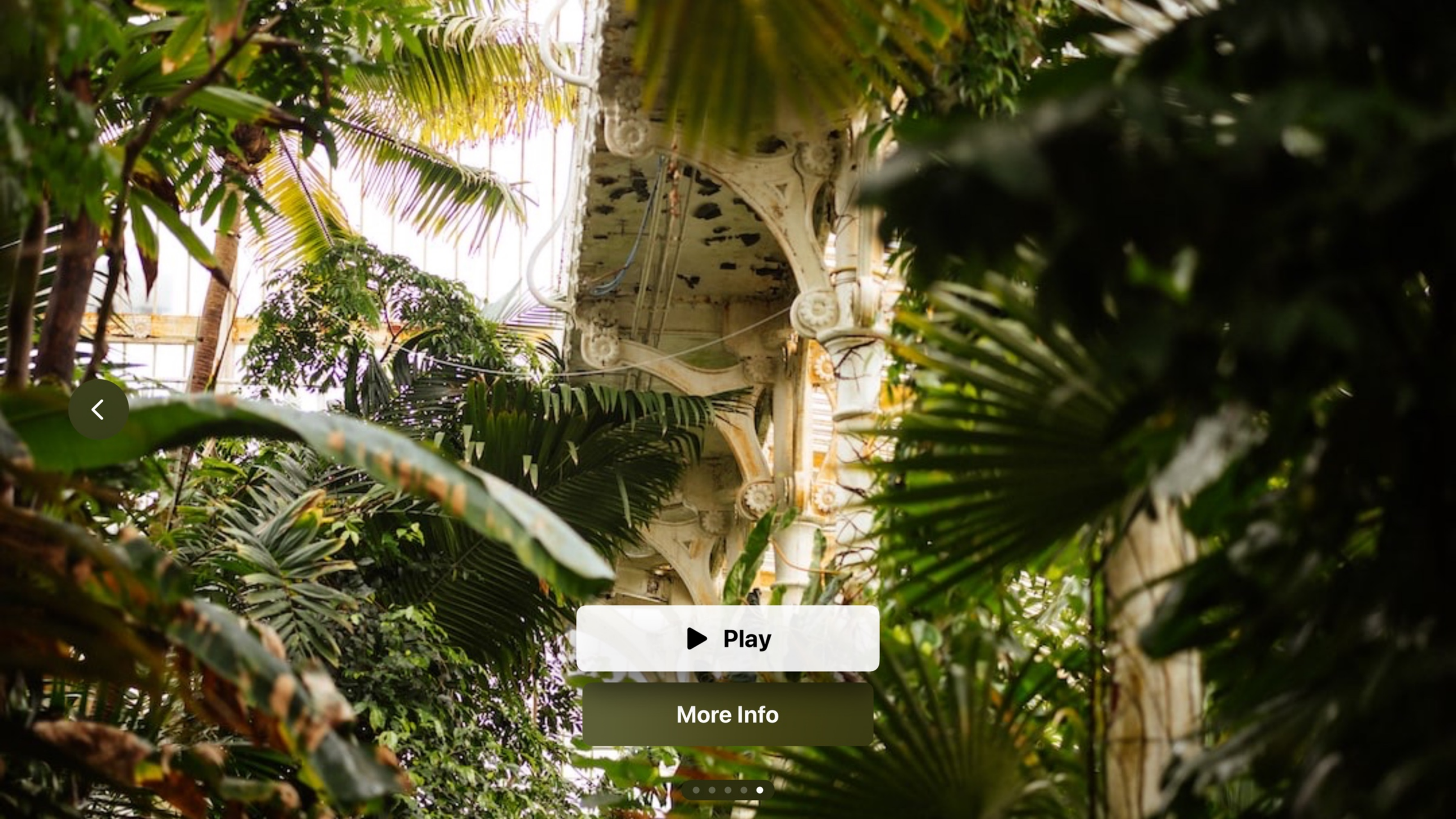
- Se charge de préparer le contenu qui doit s'afficher dans le Top Shelf
  - Implémentation d'une sous-classe de TVTopShelfContentProvider et surcharge d'une méthode

```
open func loadTopShelfContent() async -> TVTopShelfContent?
open func loadTopShelfContent(completionHandler: @escaping (TVTopShelfContent?) -> Void)
```

Top Shelf Extension

- Différent types d'affichage
  - Un carousel d'images et vidéo plein écran, avec deux boutons et un texte de détail
    - TVTopShelfCarouselContent
  - Une vue avec des sections, et des images, dans différents formats
    - TVTopShelfSectionedContent
  - Une vue avec des images en quasi full screen
    - TVTopShelfInsetContent


10:16 AM
Swipe up for full screen


Play
More Info

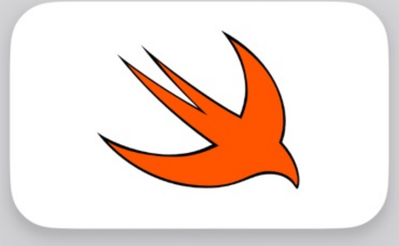
10:24 AM

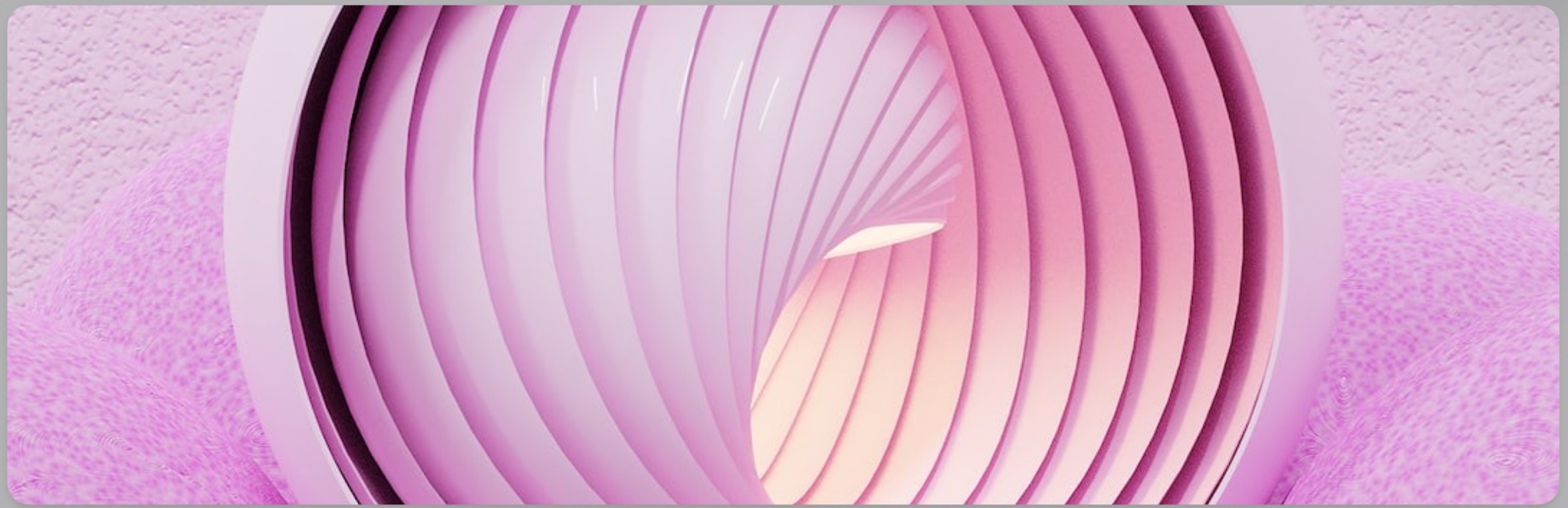







10:30 AM



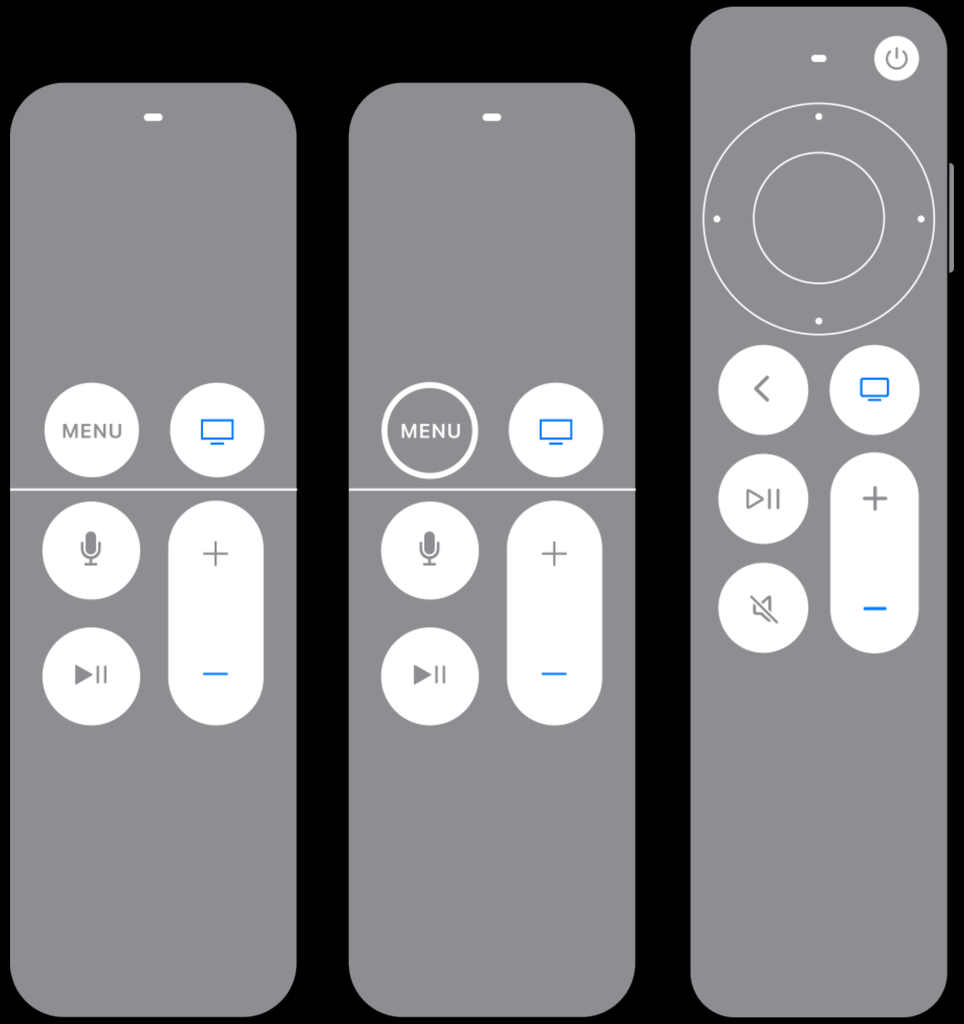


Top Shelf

- Possibilité de lancer l'app via une URL deeplink via 2 actions
- playAction
  - Déclenché si appui sur play/pause ou bouton Play (carousel)
- displayAction
  - Déclenché lors de la selection, ou de l'appui sur le bouton Détails (carousel)

Remote et focus

Remote et focus



Remote et focus

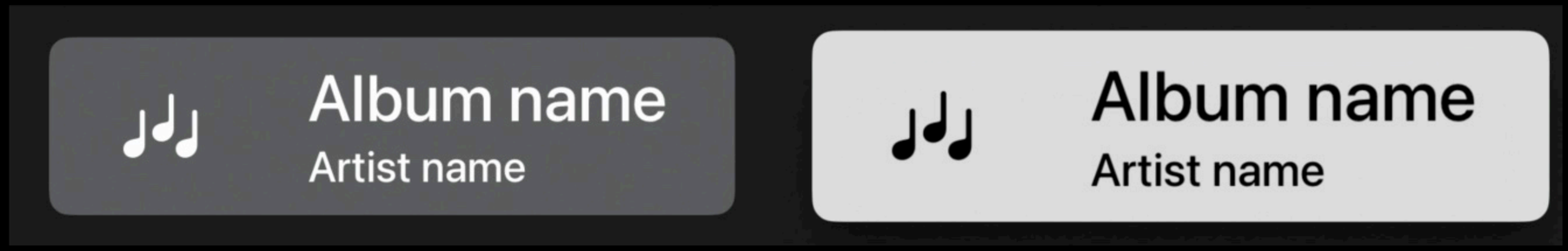
- Manipulation indirecte de l'UI via la télécommande
- Absence de curseur comme généralement le cas en manipulation indirecte
- Utilisation de la notion de focus pour manipuler les éléments
- Focus initial sur l'élément le plus haut et/ou le premier dans le sens de lecture
  - Possibilité d'altérer ce comportement avec prefersDefaultFocus(_:, in:)

Boutons et focus

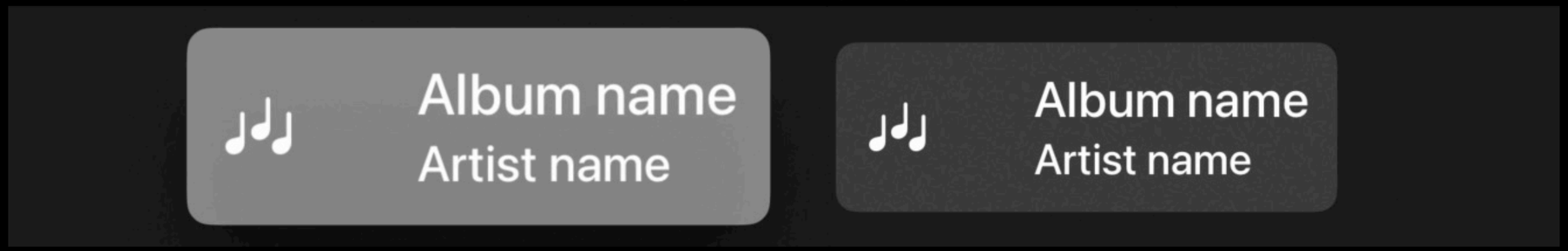
- Un bouton possède automatiquement un "hover effect" qui gère le focus
- Le style par défaut d'un bouton est bordered
  - Il bénéficie d'un effet de focus simple
- tvOS possède un style de bouton card qui applique un mouvement de parallaxe au hover

Boutons et focus

Style par défaut (.bordered)



Style de carte (.card)

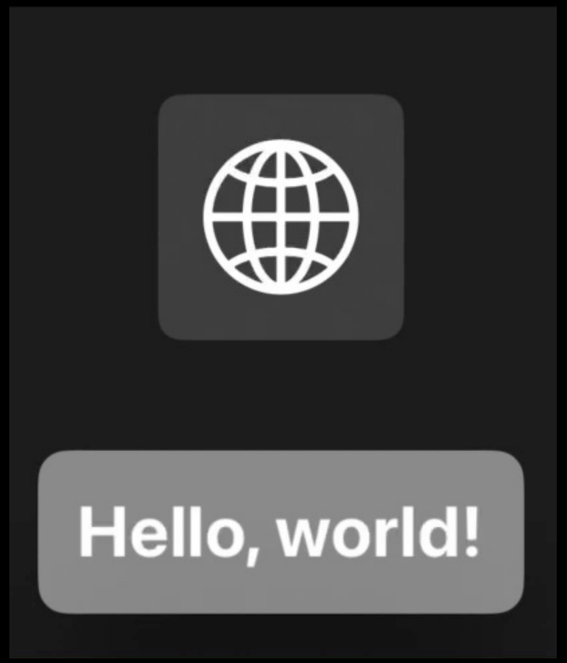


Boutons et focus

- Beaucoup d'éléments ne sont pas focusable
- On peut leur donner ce comportement avec .focusable()
- On peut leur ajouter également un .hoverEffect()
- On modifier leur apparence en surveillant l'état avec .focused() et une variable @FocusState

Boutons et focus

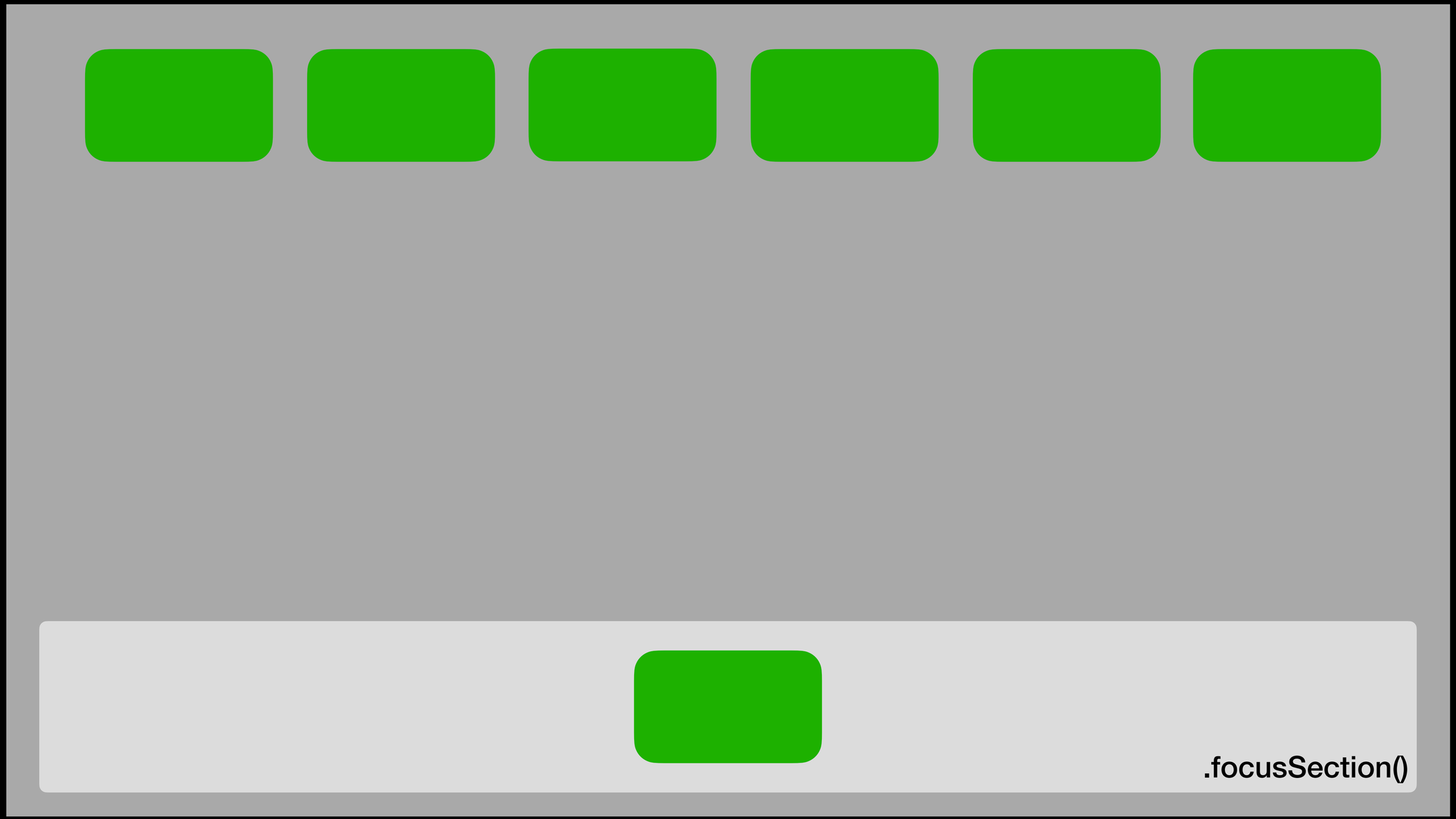
```
struct ContentView: View {

    @FocusState private var textFocused

    var body: some View {
        VStack {
            Image(systemName: "globe")
                .padding()
                .hoverEffect()
                .focusable()
            Text("Hello, world!")
                 .padding()
                 .hoverEffect()
                 .bold(textFocused)
                 .focusable()
                 .focused($textFocused)
                 .onTapGesture {
                     print("tapped")
                 }
        }
    }
}
```



Vues et focus

- Lors de la création de vues custom, on peut utiliser la variable d'environnement isFocused pour savoir si le parent de notre vues est focused
- On peut alors modifier aisément l'apparence de notre vue pour refléter ce changement

Déplacement du focus

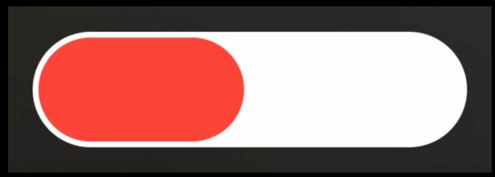


Télécommande

- Dispose d'un pavé tactile avec commandes de direction
  - onMoveCommand
- Dispose d'un bouton menu
  - onExitCommand
- Dispose d'un bouton Play/Pause
  - onPlayPauseCommand

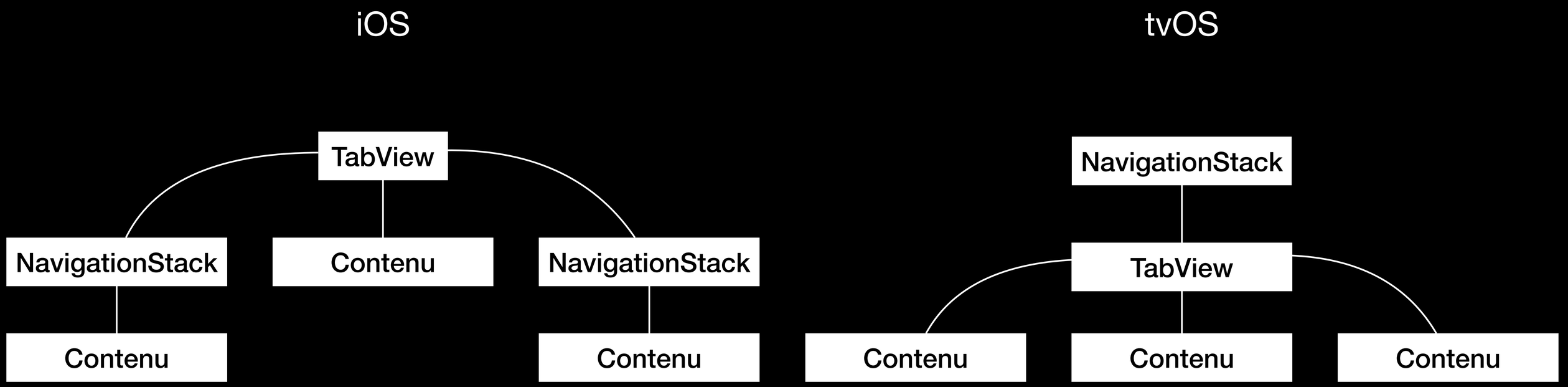
Télécommande

```
struct CustomToggle: View {

    @State private var alignment: HorizontalAlignment = .leading
    @FocusState private var thumbFocus

    var body: some View {
        Capsule()
            .focusable()
            .focused($thumbFocus)
            .frame(width: 300, height: 80)
            .overlay(alignment: Alignment(horizontal: alignment, vertical: .center)) {
                Capsule()
                    .foregroundColor(alignment == .leading ? .red : .green)
                    .padding(thumbFocus ? 4 : 8)
                    .frame(width: 150)
            }
            .scaleEffect(thumbFocus ? 1.2 : 1.0)
            .onPlayPauseCommand {
                alignment = alignment == .leading ? .trailing : .leading
            }
            .onMoveCommand { direction in
                switch direction {
                case .left:
                    alignment = .leading
                case .right:
                    alignment = .trailing
                default:
                    break
                }
            }
            .animation(.default, value: alignment)
            .animation(.default, value: thumbFocus)
    }
}
```



Navigation

iOS vs. tvOS


iOS
TabView
NavigationStack
Contenu
NavigationStack
Contenu
Contenu
tvOS
NavigationStack
TabView
Contenu
Contenu
Contenu

iOS vs. tvOS

- TabView est à la racine sur iOS, alors que sur tvOS, NavigationStack l'est
- Sur tvOS, la TabView disparait lors de la navigation pour faire ressortir le contenu