Article

Creating fully immersive experiences in your app

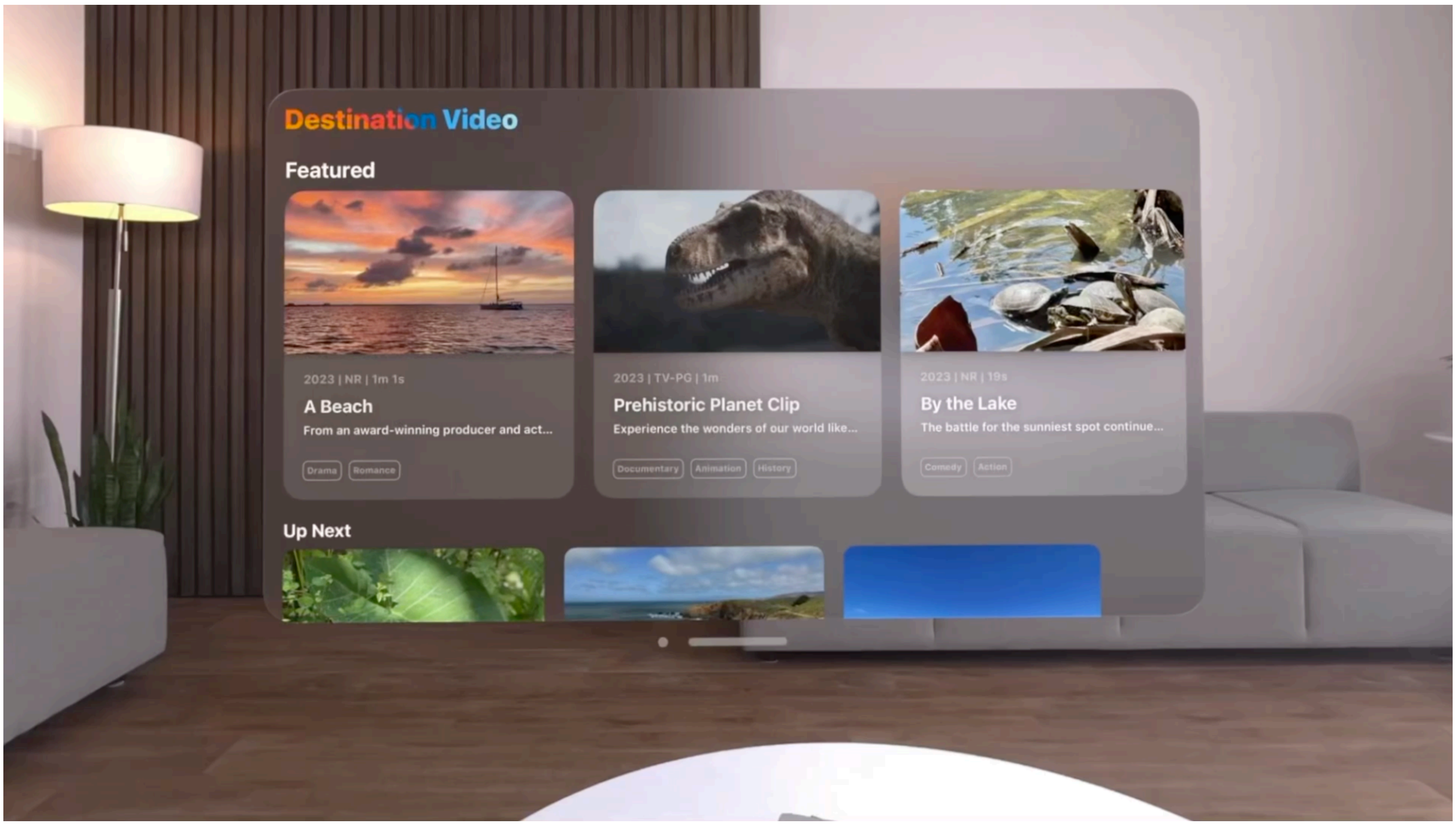
Build fully immersive experiences by combining spaces with content you create using RealityKit or Metal.

Overview

A fully immersive experience replaces everything the person sees with custom content you create. You might use this type of experience to:

- Offer a temporary transitional experience
- Create a distraction-free space for your content
- Implement a virtual reality (VR) game
- Present a virtual world to explore

With a fully immersive experience, you're responsible for everything that appears onscreen. The system hides passthrough video and displays the content you provide, showing the person's hands only when they come into view. To achieve the best performance, use RealityKit or Metal to create and animate your content.


Destination Video

Featured

2023 | NR | 1m 1s
A Beach
From an award-winning producer and act...
Drama Romance

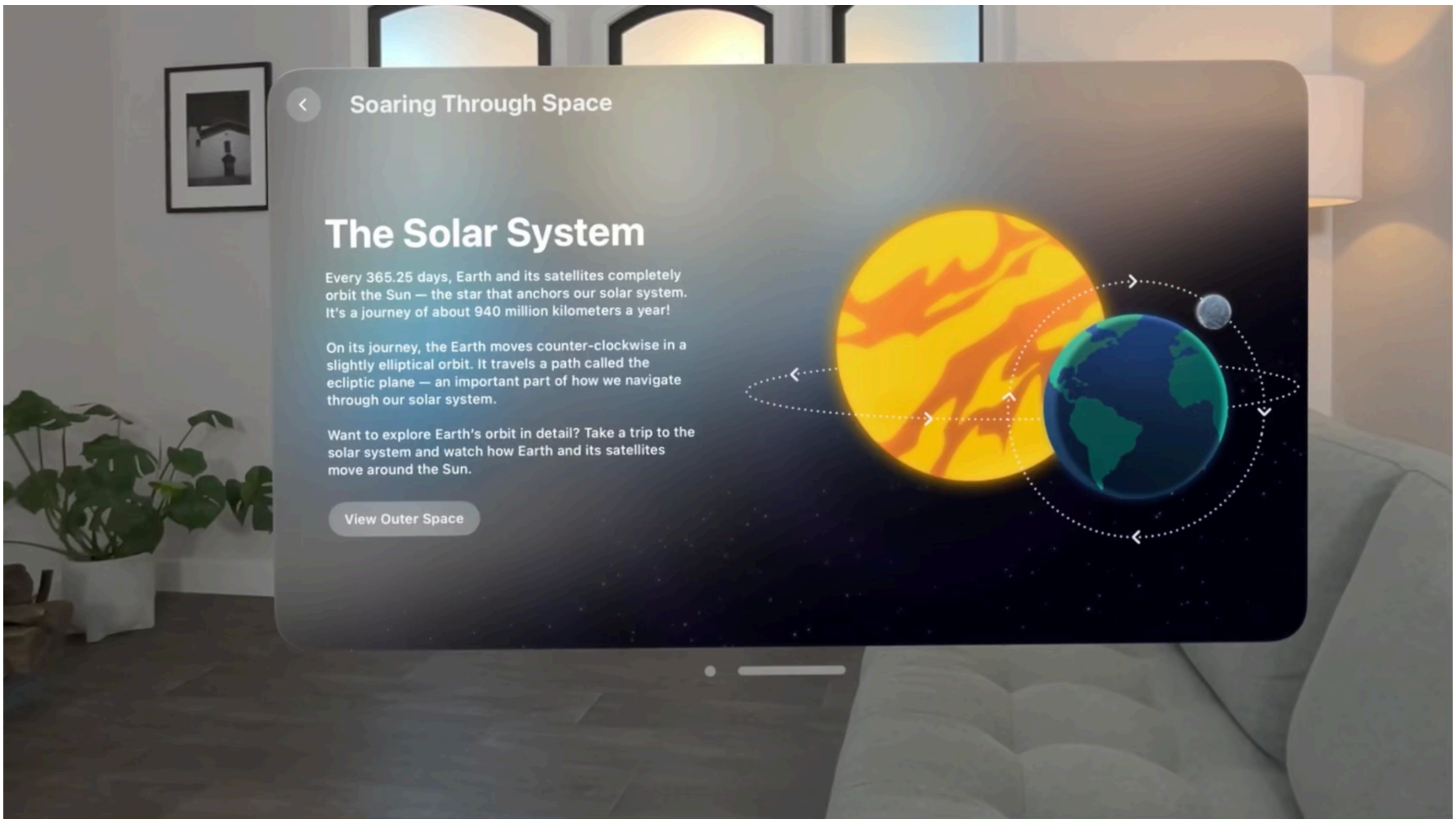
2023 | TV-PG | 1m
Prehistoric Planet Clip
Experience the wonders of our world like...
Documentary Animation History

2023 | NR | 19s
By the Lake
The battle for the sunniest spot continue...
Comedy Action

Up Next

Play ⊙

Typically, you combine a fully immersive experience with other types of experiences and provide transitions between them. When you display a window first and then offer controls to enter your immersive experience, you give people time to prepare for the transition. It also gives them the option to skip the experience if they prefer to use your app's windows instead.


Soaring Through Space

The Solar System

Every 365.25 days, Earth and its satellites completely orbit the Sun — the star that anchors our solar system. It's a journey of about 940 million kilometers a year!

On its journey, the Earth moves counter-clockwise in a slightly elliptical orbit. It travels a path called the ecliptic plane — an important part of how we navigate through our solar system.

Want to explore Earth's orbit in detail? Take a trip to the solar system and watch how Earth and its satellites move around the Sun.

View Outer Space

Play ⊙

Prepare someone for your app’s transitions

Give people control over when they enter or exit fully immersive experiences, and provide clear transitions to and from those experiences. Clear visual transitions make it easier to adjust to such a large change. Sudden transitions might be disorienting, unpleasant, or make the person think something went wrong.

At launch time, display windows or other content that allows the person to see their surroundings. Add controls to that content to initiate the transition to the fully immersive experience, and provide a clear indication of what the controls do. Inside your experience, provide clear controls and instructions on how to exit the experience.

Warning

When you start a fully immersive experience, visionOS defines a system boundary that extends approximately 1.5 meters from the initial position of the person’s head. If their head moves outside of that zone, the system automatically stops the immersive experience and turns on the external video again. This feature is an assistant to help prevent someone from colliding with objects.

For guidelines on how to design fully immersive experiences, see Human Interface Guidelines.

Open an immersive space

To create a fully immersive experience, open an `ImmersiveSpace` and set its style to `full`. An immersive space is a type of SwiftUI scene that lets you place content anywhere in the person’s surroundings. Applying the `full` style to the scene tells the system to hide passthrough video and display only your app’s content.

Declare spaces in the `body` property of your app object, or anywhere you manage SwiftUI scenes. The following example shows an app with a main window and a fully immersive space. At launch time, the app displays the window.

```swift
@main
struct MyImmersiveApp: App {
    @State private var currentStyle: ImmersionStyle = .full

    var body: some Scene {
        WindowGroup() {
            ContentView()
        }

        // Display a fully immersive space.
        ImmersiveSpace(id: "solarSystem") {
            SolarSystemView()
        }.immersionStyle(selection: $currentStyle, in: .full)
    }
}
```

To display an ImmersiveSpace, open it using the openImmersiveSpace action, which you obtain from the SwiftUI environment. This action runs asynchronously and uses the provided information to find and initialize your scene. The following example shows a button that opens the space with the solarSystem identifier:

```
Button("Show Solar System") {
    Task {
        let result = await openImmersiveSpace(id: "solarSystem")
        if case .error = result {
            print("An error occurred")
        }
    }
}
```

An app can display only one space at a time, and it's an error for you to try to open a space while another space is visible. To dismiss an open space, use the dismissImmersiveSpace action.

For more information about displaying spaces, see the ImmersiveSpace type.

Draw your content using RealityKit

RealityKit works well when your content consists of primitive shapes or existing content in USD files. Organize the contents of your scene using RealityKit entities, and animate that content using components and systems. Use Reality Composer Pro to assemble your content visually, and to attach dynamic shaders, animations, audio, and other behaviors to your content. Display the contents of your RealityKit scene in a RealityView in your scene.

To load a Reality Composer Pro scene at runtime, fetch the URL of your Reality Composer Pro package file, and load the root entity of your scene. The following example shows how to create the entity for a package located in the app's bundle:

```
import MyRealityBundle

let url = MyRealityBundle.bundle.url(forResource:
         "MyRealityBundle", withExtension: "reality")
```

```
let scene = try await Entity(contentsOf: url)
```

For more information about how to display content in a `RealityView` and manage interactions with your content, see Adding 3D content to your app.

Draw your content using Metal

Another option for creating fully immersive scenes is to draw everything yourself using Metal. When using Metal to draw your content, use the Compositor Services framework to place that content onscreen. Compositor Services provides the code you need to set up your Metal rendering engine and start drawing.

For details on how to render content using Metal and Compositor Services, and manage interactions with your content, see Drawing fully immersive content using Metal.

See Also

App construction

Creating your first visionOS app

Build a new visionOS app using SwiftUI and add platform-specific features.

Adding 3D content to your app

Add depth and dimension to your visionOS app and discover how to incorporate your app's content into a person's surroundings.

Drawing sharp layer-based content in visionOS

Deliver text and vector images at multiple resolutions from custom Core Animation layers in visionOS.

Introductory visionOS samples

Learn the fundamentals of building apps for visionOS with beginner-friendly sample code projects.