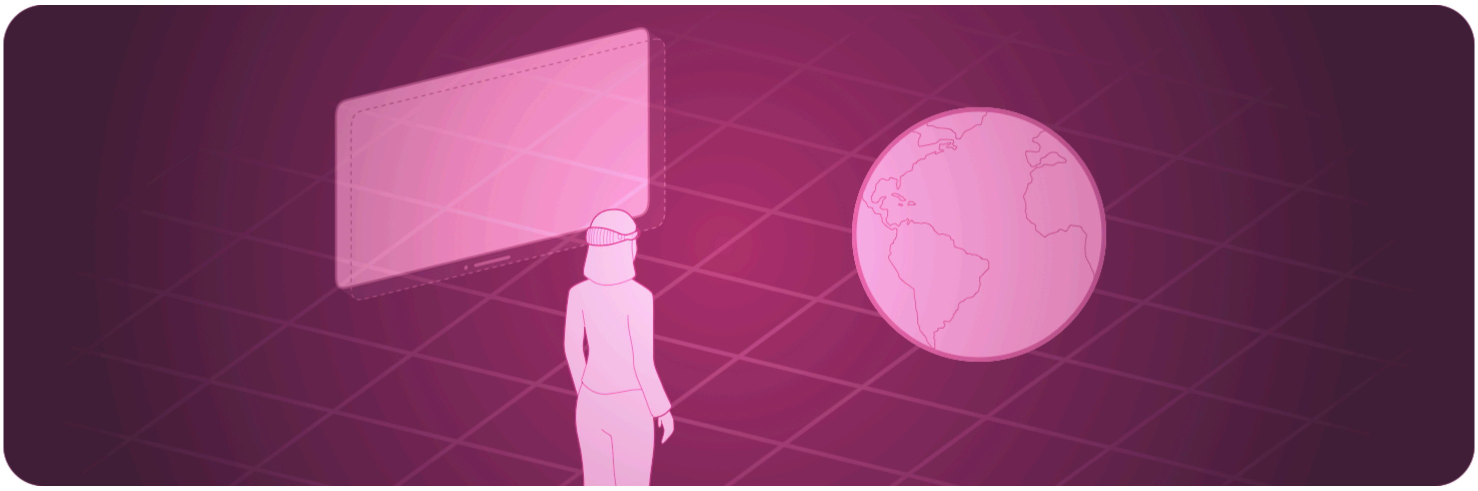
API Collection

Immersive spaces

Display unbounded content in a person’s surroundings.

Overview

Use an immersive space in visionOS to present SwiftUI views outside of any containers. You can include any views in a space, although you typically use a `RealityView` to present RealityKit content.



You can request one of three styles of spaces with the `immersionStyle(selection:in:)` scene modifier:

- The `mixed` style blends your content with passthrough. This enables you to place virtual objects in a person’s surroundings.
- The `full` style displays only your content, with passthrough turned off. This enables you to completely control the visual experience, like when you want to transport people to a new world.
- The `progressive` style completely replaces passthrough in a portion of the display. You might use this style to keep people grounded in the real world while displaying a view into another world.

When you open an immersive space, the system continues to display all of your app's windows, but hides windows from other apps. The system supports displaying only one space at a time across all apps, so your app can only open a space if one isn't already open.

Topics

Creating an immersive space

`struct ImmersiveSpace`

A scene that presents its content in an unbounded space.

`struct ImmersiveSpaceContentBuilder`

A result builder for composing a collection of immersive space elements.

`func immersionStyle(selection: Binding<any ImmersionStyle>, in: any ImmersionStyle...) -> some Scene`

Sets the style for an immersive space.

`protocol ImmersionStyle`

The styles that an immersive space can have.

`var immersiveSpaceDisplacement: Pose3D`

The displacement that the system applies to the immersive space when moving the space away from its default position, in meters.

`struct ImmersiveEnvironmentBehavior`

The behavior of the system-provided immersive environments when a scene is opened by your app.

`struct ProgressiveImmersionAspectRatio`

Opening an immersive space

`var openImmersiveSpace: OpenImmersiveSpaceAction`

An action that presents an immersive space.

`struct OpenImmersiveSpaceAction`

An action that presents an immersive space.

Closing the immersive space

`var dismissImmersiveSpace: DismissImmersiveSpaceAction`

An immersive space dismissal action stored in a view's environment.

`struct DismissImmersiveSpaceAction`

An action that dismisses an immersive space.

Hiding upper limbs during immersion

`func upperLimbVisibility(Visibility) -> some Scene`

Sets the preferred visibility of the user's upper limbs, while an `ImmersiveSpace` scene is presented.

`func upperLimbVisibility(Visibility) -> some View`

Sets the preferred visibility of the user's upper limbs, while an `ImmersiveSpace` scene is presented.

Adjusting content brightness

`func immersiveContentBrightness(ImmersiveContentBrightness) -> some Scene`

Sets the content brightness of an immersive space.

`struct ImmersiveContentBrightness`

The content brightness of an immersive space.

Responding to immersion changes

`func onImmersionChange(initial: Bool, (ImmersionChangeContext, ImmersionChangeContext) -> Void) -> some View`

Performs an action when the immersion state of your app changes.

`struct ImmersionChangeContext`

A structure that represents a state of immersion of your app.

Adding menu items to an immersive space

`func immersiveEnvironmentPicker<Content>(content: () -> Content) -> some View`

Add menu items to open immersive spaces from a media player's environment picker.

Handling remote immersive spaces

`struct RemoteImmersiveSpace`

A scene that presents its content in an unbounded space on a remote device.

`struct RemoteDeviceIdentifier`

An opaque type that identifies a remote device displaying scene content in a `RemoteImmersiveSpace`.

See Also

App structure

App organization

Define the entry point and top-level structure of your app.

Scenes

Declare the user interface groupings that make up the parts of your app.

Windows

Display user interface content in a window or a collection of windows.

Documents

Enable people to open and manage documents.

Navigation

Enable people to move between different parts of your app's view hierarchy within a scene.

Modal presentations

Present content in a separate view that offers focused interaction.

Toolbars

Provide immediate access to frequently used commands and controls.

Search

Enable people to search for text or other content within your app.

App extensions

Extend your app’s basic functionality to other parts of the system, like by adding a Widget.