SwiftUI / ImmersiveSpace

Structure

ImmersiveSpace

A scene that presents its content in an unbounded space.

visionOS 1.0+

```
struct ImmersiveSpace<Content, Data> where Content : ImmersiveSpace
Content, Data : Decodable, Data : Encodable, Data : Hashable
```

Overview

Use an immersive space as a container for a view hierarchy that your app presents. The hierarchy that you declare as the immersive space's content serves as a template for it:

```
@main
struct SolarSystemApp: App {
    var body: some Scene {
        ImmersiveSpace {
            SolarSystem()
        }
    }
}
```

If you want to create a bounded scene instead, use one of the types that creates a window or a volume, like `WindowGroup` or `DocumentGroup`.

Style the immersive space

By default, immersive spaces use the `mixed` style which places virtual content in a person's surroundings. You can select a different style for the immersive space by adding the `immersion`

Style(selection:in:) scene modifier to the scene. For example, you can completely control the visual experience using the full immersion style:

```
@main
struct SolarSystemApp: App {
    @State private var style: ImmersionStyle = .full

    var body: some Scene {
        ImmersiveSpace {
            SolarSystem()
        }
        .immersionStyle(selection: $style, in: .full)
    }
}
```

You can change the immersion style after presenting the immersive space by changing the modifier's `selection` input, although you can only use one of the values that you specify in the modifier's second parameter. For any style of immersion, the other parts of your app's interface — namely its windows — remain visible. However, the immersion style affects how windows interact with virtual objects in the environment:

- For the mixed style, a virtual object obscures part or all of a window that's behind the object. Similarly, a window obscures a virtual object that's behind the window.
- For other styles, windows always render in front of virtual content, no matter how someone positions the window or the content. This helps people to avoid losing track of windows behind virtual content when passthrough is partially or completely off.

Open an immersive space

You can programmatically open an immersive space by giving it an identifier. For example, you can label the solar system view from the previous example:

```
ImmersiveSpace(id: "solarSystem") {
    SolarSystem()
}
```

Elsewhere in your code, you use the openImmersiveSpace environment value to get the instance of the OpenImmersiveSpaceAction structure for a given Environment. You call the instance directly — for example, from a button's closure, like in the following code — using the identifier:

```swift
struct NewSolarSystemImmersiveSpace: View {
    var solarSystem: SolarSystem
    @Environment(\.openImmersiveSpace) private var openImmersiveSpace

    var body: some View {
        Button("Present Solar System") {
            Task {
                await openImmersiveSpace(id: "solarSystem")
            }
        }
    }
}
```

Mark the call to the action with `await` because it executes asynchronously. When your app opens an immersive space, the system hides all other visible apps. The system allows only one immersive space to be open at a time. Be sure to close the open immersive space before opening another one.

Dismiss an immersive space

You can dismiss an immersive space by calling the `dismissImmersiveSpace` action from the environment. For example, you can define a button that dismisses an immersive space:

```swift
struct DismissImmersiveSpaceButton: View {
    @Environment(\.dismissImmersiveSpace)
    private var dismissImmersiveSpace

    var body: some View {
        Button("Close Solar System") {
            Task {
                await dismissImmersiveSpace()
            }
        }
    }
}
```

The dismiss action runs asynchronously, like the open action. You don't need to specify an identifier when dismissing an immersive space because there can only be one immersive space open at a time.

Present an immersive space at launch

When an app launches, it opens an instance of the first scene that's listed in the app's body. However, to open an immersive space at launch, you need to provide additional configuration information in your app's `Info.plist` file. In particular, set the `UIApplicationPreferredDefaultSceneSessionRole` key in the scene manifest to the value `UISceneSessionRoleImmersiveSpaceApplication`.

To configure the style of the immersive space that opens at launch, add a scene configuration to the scene session role. Use the `UISceneInitialImmersionStyle` key together with a value that indicates one of the mixed, full, or progressive styles. See the initial immersion style key for more information.

Topics

Creating an immersive space

`init(content:)`

Creates an immersive space.

Identifying an immersive space

~~`init(id:content:)`~~

Creates the immersive space associated with the specified identifier.

Deprecated

Creating a data-driven immersive space

`init(for:content:)`

Creates the immersive space for a specified type of presented data.

`init(id:for:content:)`

Creates the immersive space associated with an identifier for a specified type of presented data.

Providing default data to an immersive space

`init(for:content:defaultValue:)`

Creates an immersive space.

`init(id:for:content:defaultValue:)`

Creates the immersive space associated with an identifier for a specified type of presented data, and a default value, if the data is not set.

Supporting types

`struct ImmersiveSpaceViewContent`

Immersive space content that uses a SwiftUI view hierarchy as the content.

`protocol ImmersiveSpaceContent`

A type that you can use as the content of an immersive space.

Initializers

`init<C>(for: Data.Type, makeContent: (Binding<Data?>) -> C)`

`init<C>(for: Data.Type, makeContent: (Binding<Data>) -> C, defaultValue: () -> Data)`

`init<C>(id: String, for: Data.Type, makeContent: (Binding<Data?>) -> C)`

`init<C>(id: String, for: Data.Type, makeContent: (Binding<Data>) -> C, defaultValue: () -> Data)`

`init(id:makeContent:)`

Creates the immersive space associated with the specified identifier.

`init<C>(makeContent: () -> C)`

Relationships

Conforms To

`Scene`

See Also

Creating an immersive 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`