

[RealityKit](#) / [RealityView](#)

Structure

RealityView

A view that contains RealityKit content.

RealityKit | SwiftUI | iOS 18.0+ | iPadOS 18.0+ | Mac Catalyst 18.0+ | macOS 15.0+ | tvOS 26.0+ | visionOS 1.0+

```
@MainActor @preconcurrency
struct RealityView<Content> where Content : View
```

Mentioned in

Implementing scene understanding and reconstruction in your RealityKit app

Overview

Use `RealityView` to display rich 3D RealityKit content in your app, including content you author in Reality Composer Pro. `RealityView` passes a structure that conforms to [RealityView ContentProtocol](#) to its make and update closures, which you can use to add and remove RealityKit entities to your view.

Here is a simple example showing how you can display a custom [ModelEntity](#) using `RealityView`:

```
struct ModelExample: View {
    var body: some View {
        RealityView { content in
            if let robot = try? await ModelEntity(named: "robot") {
                content.add(robot)
            }
        }
    }
}
```

```

        Task {
            // Asynchronously perform any additional work to configure
            // the content after the system renders the view.

        }
    }
}
}

```

Note that the closure in the example above is a sync, and can be used to load contents from your app's bundle or from any URL in the background. While your content is loading, `RealityView` will automatically display a placeholder view, which you can customize using the optional `placeholder` parameter.

Tip

Load your content asynchronously to avoid introducing a hang in your app.

You can also use the optional `update` closure on your `RealityView` to update your `RealityKit` content in response to changes in your view's state. `RealityView` displays your `RealityKit` content inline in true 3D space, occupying the available space in your app's 3D bounds. The [RealityViewContent](#) type on visionOS, and [RealityViewCameraContent](#) on other platforms represents the content of your `RealityView`.

If you want to run code every frame (to do animations or simulations), you can use a [System](#) or directly subscribe to the engine's `SceneEvents.Update`:

```

RealityView { content in
    let entity = ModelEntity(mesh: .generateSphere(radius: 0.1))
    content.add(entity)
    _ = content.subscribe(to: SceneEvents.Update.self) { event in
        entity.position.y -= Float(event.deltaTime)
    }
}

```

`RealityView` has a flexible size by default, and does not size itself based on the `RealityKit` content it displays. For more advanced uses of `RealityKit`, such as subscribing to `RealityKit` events, performing coordinate conversions, or working with AR capabilities, refer to the [RealityViewContentProtocol](#) types.

Topics

Creating a reality view for visionOS

```
init(make: (inout RealityViewContent) async -> Void, update: ((inout RealityViewContent) -> Void)?)
```

Creates a new reality view for visionOS with an optional update closure.

```
init<P>(make: (inout RealityViewContent) async -> Void, update: ((inout RealityViewContent) -> Void)?, placeholder: () -> P)
```

Creates a new reality view for visionOS with an optional update closure and placeholder view.

```
init<A>(make: (inout RealityViewContent, RealityViewAttachments) async -> Void, update: ((inout RealityViewContent, RealityViewAttachments) -> Void)?, attachments: () -> A)
```

Creates a reality view for visionOS, with attachments and an optional update closure.

```
init<A, P>(make: (inout RealityViewContent, RealityViewAttachments) async -> Void, update: ((inout RealityViewContent, RealityViewAttachments) -> Void)?, placeholder: () -> P, attachments: () -> A)
```

Creates a reality view for visionOS, with attachments, an optional update closure, and placeholder view.

Creating a reality view for iOS and macOS

```
init(make: (inout RealityViewCameraContent) async -> Void, update: ((inout RealityViewCameraContent) -> Void)?)
```

Creates a reality view for iOS and macOS, with an optional update closure.

```
init<P>(make: (inout RealityViewCameraContent) async -> Void, update: ((inout RealityViewCameraContent) -> Void)?, placeholder: () -> P)
```

Creates a reality view for iOS and macOS, with an optional update closure and placeholder view.

Inspecting the content within a reality view

```
typealias DefaultPlaceholder
```

Initializers

```
init(make:update:)
```

Creates a reality view for iOS and macOS, with an optional update closure.

```
init(make:update:placeholder:)
```

Creates a reality view for iOS and macOS, with an optional update closure and placeholder view.

Relationships

Conforms To

Sendable, SendableMetatype, View

See Also

SwiftUI scene presentation

```
struct RealityViewContent
```

The content of a visionOS reality view.

```
struct RealityViewCameraContent
```

The content of a reality view that is displayed through a camera.

```
protocol RealityViewContentProtocol
```

A protocol representing the content of a reality view.

```
struct RealityViewDefaultPlaceholder
```

A view that represents the default placeholder for a RealityView.

```
struct RealityViewEntityCollection
```

A collection of entities in a RealityView.

```
struct RealityViewLayoutOption
```

Options that specify the frame sizing and content alignment option for RealityView.

```
protocol EntityCollection
```

An ordered, mutable collection of entities.