Sample Code

Building an immersive media viewing experience

Add a deeper level of immersion to media playback in your app with RealityKit and Reality Composer Pro.

Download (1.2 GB)

visionOS 2.0+ | Xcode 16.0+

Overview

visionOS provides powerful features for building immersive media playback apps. It supports playing 3D video and Spatial Audio, which helps bring the content to life and makes the viewer feel like they’re part of the action. Starting in visionOS 2, you can take your app’s playback experience even further by creating custom environments using RealityKit and Reality Composer Pro.

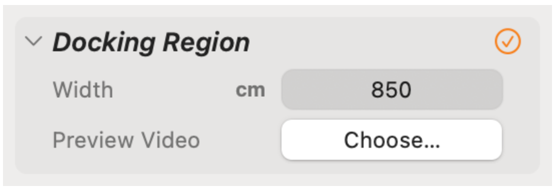
The Destination Video sample includes a custom environment, Studio. The Studio environment provides a large, open space that’s specifically designed to provide an optimal media viewing experience, as shown in the following image.

Define a video docking location

Destination Video uses AVPlayerViewController to present video, which enables the app to provide a playback experience across platforms that matches system apps like TV and Music. In visionOS, AVPlayerViewController participates in the system docking behavior. When you play video in a full-window player then open an immersive experience, the system docks the video screen in a fixed location and presents streamlined playback controls that keep your focus on the content.

The system determines the docking location for the scene by default. In visionOS 2, you can customize this location by specifying a custom docking region.

The environment in Destination Video anchors the video player in front of the walkway at the top of the staircase. To have the video player dock to this location, the project defines a `Player` entity and adds a `DockingRegionComponent` to it in Reality Composer Pro's Inspector. This component defines the bounding region for the video player, which has a depth of `0` and uses a fixed 2.4:1 aspect ratio. Because the aspect ratio is fixed, to configure the docking region's size use the `width` property.



Docking Region

Width	cm	850
Preview Video		Choose...

Reality Composer Pro provides a template to set up a configuration for Docking Region and related media reflections. You can access this template from the Insert menu by selecting Insert > Environment > Video Dock.

Choose a location for your docking region that provides a comfortable viewing angle to avoid causing strain or discomfort during longer viewing sessions. Avoid placing objects between the viewer and the video. Using Reality Composer Pro to define the docking region helps to visualize how it looks in context. Review your environment and docking region placement on Apple Vision Pro to get an appropriate sense of scale and layout.

Enhance the realism of your scene with reflections

To make your environment feel like a real, dynamic space, enable reflections from the video player on your environment surfaces. Reality Composer Pro supports two types of reflections that it exposes as Shader Graph nodes:

Reflection Specular
: A direct reflection of media content. Apply this reflection type on glossy surfaces like metals, mirrors, and water.

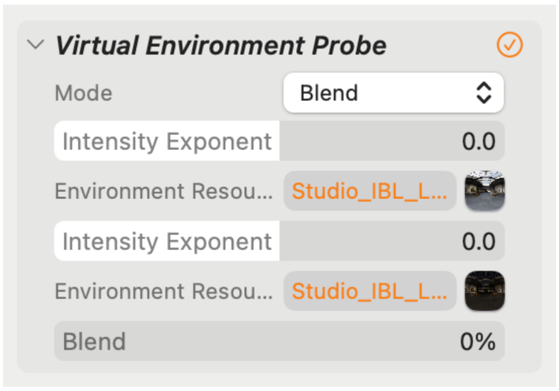
Reflection Diffuse
: A softer falloff of media content. Apply this reflection type to rougher, more organic surfaces like concrete or wood floor.

Destination Video uses both types of reflections in its custom environment to create a more realistic experience that better grounds the video player in the scene.

To learn more about how the custom environment uses reflections, see Enabling video reflections in an immersive environment.

Define the virtual scene lighting

When Destination Video presents the Studio environment in a progressive immersion style, the scene provides a source of indirect lighting to the system. RealityKit represents this light source using an instance of `VirtualEnvironmentProbeComponent`. To configure this lighting, the Studio scene defines an `EnvironmentProbe` entity and adds a Virtual Environment Probe component to it in Reality Composer Pro's Inspector. It defines the source as a blend between the project's light and dark image-based lighting (IBL) files, as shown below.


Virtual Environment Probe

Mode: Blend

Intensity Exponent 0.0

Environment Resou... Studio_IBL_L...

Intensity Exponent 0.0

Environment Resou... Studio_IBL_L...

Blend 0%

When the app presents the light or dark variant, it looks up the probe and configures the source with an appropriate blend value. The app passes a value of `0.0` for the light variant and `1.0` for the dark, which effectively toggles which image the component uses as its source.

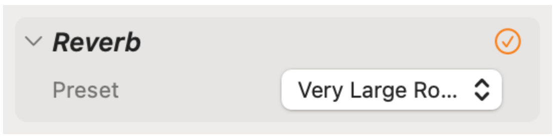
```
private func setVirtualEnvironmentProbeComponent(blendParam: Float) {
    let virtualEnvironmentProbeEntityName = "EnvironmentProbe"

    guard let virtualEnvironmentProbeEntity = findCommonEntityByName(virtualEnvironm
        logger.warning("\(virtualEnvironmentProbeEntityName) not found.")
        return
    }

    if var probeComponent = virtualEnvironmentProbeEntity.components[VirtualEnvironm
        if case VirtualEnvironmentProbeComponent.Source.blend(let firstProbe, let se
            probeComponent.source = .blend(from: firstProbe, to: secondProbe, t: ble
            virtualEnvironmentProbeEntity.components[VirtualEnvironmentProbeComponer
        }
    }
}
```

Enhance audio immersion

By default, visionOS reverberates spatial audio sources by simulating the acoustics of the user's real environment. When you present a custom environment in a progressive or fully immersive space, you can enhance the level of immersion by applying reverb that matches the visuals of your scene.

> **Note**
>
> When your app presents an environment in an immersive space using the `progressive` immersion style, turning the Digital Crown blends the acoustics of the real and virtual spaces to match the visual level of immersion.

The Studio environment defines a `Reverb` entity and adds a `ReverbComponent` to it in Reality Composer Pro's Inspector. The component defines a single `reverb` property to indicate a specific preset to apply. There are several high-quality reverb presets to choose from including various rooms, hall, and outside spaces. The app uses the `Very Large Room` preset, which best fits the environment's visuals.


Reverb
Preset: Very Large Ro...

Define a reverb component in your scene even if it doesn't provide custom audio. The system still uses the reverb preset to spatialize system sounds such as UI interactions.

> Note
>
> To optimize the viewing experience, in most cases lower the volume of environment sounds, or stop their playback altogether, when video playback begins.

Specify content brightness and surroundings effects

Destination Video presents the Studio environment by opening an immersive space in the `progressive` immersion style. This style works well for media apps because people can customize their level of immersion by turning the Digital Crown - from no immersion to fully immersive.

To enhance the media presentation and create a more immersive experience when presenting the Studio environment, the app customizes the space in the following ways:

- It specifies a content brightness value for the immersive space, which indicates the overall brightness of the scene. The system uses this value to tailor the video presentation to best fit its surroundings.
- It sets a custom tint color for the video passthrough of the user's hands and surroundings. The app defines tint color that matches the light or dark variant of the environment, and sets the appropriate tint color using the `preferredSurroundingsEffect(_:)` view modifier.

```
// Defines an immersive space to present an environment in which to watch the video.
ImmersiveSpace(id: ImmersiveEnvironmentView.id) {
    ImmersiveEnvironmentView()
        .environment(immersiveEnvironment)
        .onAppear {
            contentBrightness = immersiveEnvironment.contentBrightness
            surroundingsEffect = immersiveEnvironment.surroundingsEffect
        }
        .onDisappear {
            contentBrightness = .automatic
            surroundingsEffect = nil
        }
        // Set the surroundings effect for the immersive space.
        .preferredSurroundingsEffect(surroundingsEffect)
}
// Set the content brightness for the immersive space.
.immersiveContentBrightness(contentBrightness)
.immersionStyle(selection: .constant(.progressive), in: .progressive)
```

Present a custom scene in the environment picker

In visionOS 2, `AVPlayerViewController` automatically displays a button for a person to pick an environment. When a person presses the button, the system displays a list of viewing environments in which they can watch the video. Selecting an item from the list opens the environment, and docks the player into its ideal viewing location within the scene. By default, the environment picker lists the most recently used system environments, but you can configure the list to show your custom environments as well.

Destination Video adds the Studio environment's light and dark variants to the list by attaching the new `immersiveEnvironmentPicker(content:)` view modifier to the player view. This modifier takes a `ViewBuilder` that defines a button for each environment entry you're adding.

```
PlayerView()
    .immersiveEnvironmentPicker {
        ImmersiveEnvironmentPickerView()
    }
```

This sample passes the modifier a custom view that defines two buttons: one for the light variant and one for the dark. Each button displays a title, thumbnail image, and subtitle that indicates which variant it is.

```
struct ImmersiveEnvironmentPickerView: View {
    @Environment(ImmersiveEnvironment.self) private var immersiveEnvironment

    var body: some View {
        studioButton(state: .dark)
        studioButton(state: .light)
    }

    private func studioButton(state: EnvironmentStateType) -> some View {
        Button {
            immersiveEnvironment.requestEnvironmentState(state)
            immersiveEnvironment.loadEnvironment()
        } label: {
            Label {
                Text("Studio", comment: "Show Studio environment")
            } icon: {
                Image(["studio_thumbnail", state.displayName.lowercased()].joined(se
            }
            Text(state.displayName)
        }
    }
}
```

After adding these buttons to the environment picker, they appear alongside the system environments:

See Also

Related samples

{} Destination Video

Leverage SwiftUI to build an immersive media experience in a multiplatform app.

Related articles

{} Enabling video reflections in an immersive environment

Create a more immersive experience by adding video reflections in a custom environment.

Related videos

Enhance the immersion of media viewing in custom environments

Create custom environments for your immersive apps in visionOS