Structure

VideoPlayerComponent

A component that supports general video-playback experience with an AV player.

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

```
struct VideoPlayerComponent
```

Overview

To streamline and enhance video playback controls, a video player component empowers an app to support captions, subtitles, light spill, and passthrough tinting (visionOS only). Attach a `VideoPlayerComponent` to an entity to start configuring playback controls.

The example below shows an entity with a `VideoPlayerComponent` attached:

Play ⦿

The following code example shows the basic setup for initializing a `VideoPlayerComponent` with an `AVPlayer`:

```
// Create an entity for display.
let videoEntity = Entity()

// Create an AV player with a URL.
let player = AVPlayer(url: "PLACEMENT_URL")

// Create a video player component with the AV player.
let videoPlayerComponent = VideoPlayerComponent(avPlayer: player)

// Attach the video player component to the entity.
videoEntity.components.set(videoPlayerComponent)
```

Here are some common usage scenarios before attaching the video player component to the entity:

```
// Set the desired viewing mode to stereo to enable
// stereoscopic playback of the video, if available.
videoPlayerComponent.desiredViewingMode = VideoPlaybackController.ViewingMode.stereo

// Enable passthrough tinting during video playback.
videoPlayerComponent.isPassthroughTintingEnabled = true
```

Here are a few event subscription examples for `VideoPlayerEvents`:

```
RealityView { content in
    // Set up a video player component with an AV player.
    let entity = Entity()
    let player = AVPlayer(url: "PLACEMENT_URL")
    let videoPlayerComponent = VideoPlayerComponent(avPlayer: player)
    player.play()
    entity.components.set(videoPlayerComponent)

    var subscription: EventSubscription?

    // Subscribe to the video screen size change event.
    subscription = content.subscribe(to:
        VideoPlayerEvents.VideoSizeDidChange.self) { event in
        print("video size did change: \(event.videoDimension)")
    }

   // Subscribe to the current viewing mode change event.
    subscription = content.subscribe(to:
        VideoPlayerEvents.ViewingModeDidChange.self) { event in
        print("viewing mode did change: \(event.currentViewingMode)")
    }

    // Subscribe to the content type change event.
    subscription = content.subscribe(to:
        VideoPlayerEvents.ContentTypeDidChange.self) { event in
        print("content type did change: \(event.contentType.rawValue)")
    }

    content.add(entity)
}
```

Playing immersive video

On visionOS, you can also use `VideoPlayerComponent` to play immersive media with RealityKit. Watch videos in a window alongside other Shared Space apps, or watch immersive video with a 180-degree field of view in a fully immersive space, or watch immersive video with a progressive view in a progressive immersive space

Playing immersive video with a 180-degree field of view

To play the immersive media with a 180-degree field of view, first set up a fully immersive space, and then set `desiredImmersiveViewingMode` to `VideoPlayerComponent.ImmersiveViewingMode.full`.

If you want to switch the immersive-viewing mode to `full` while immersive-media playback is in `VideoPlayerComponent.ImmersiveViewingMode.portal` mode, wait for the scene event named `VideoPlayerEvents.ImmersiveViewingModeDidChange` to trigger after you set `desiredImmersiveViewingMode` to `full`. Then, dismiss the window scene and open up a fully immersive space scene.

Playing immersive video with a progressive view

To play the immersive media with a progressive view, first set up a progressive immersive space, and then set `desiredImmersiveViewingMode` to `VideoPlayerComponent.ImmersiveViewingMode.progressive`.

If you want to switch the immersive-viewing mode to `progressive` while immersive-media playback is in `VideoPlayerComponent.ImmersiveViewingMode.portal` mode, wait for the scene event named `VideoPlayerEvents.ImmersiveViewingModeDidChange` to trigger after you set `desiredImmersiveViewingMode` to `progressive`. Then, dismiss the window scene and open up a progressive immersive space scene.

Playing immersive video in a portal window

To play the immersive video in a portal window, set up a window scene in the Shared Space, and then set `desiredImmersiveViewingMode` to `VideoPlayerComponent.ImmersiveViewingMode.portal`.

If you want to switch the immersive-viewing mode to `portal` while immersive-media playback is in `full` or is in `progressive` mode, wait for the scene event named `VideoPlayerEvents.ImmersiveViewingModeDidChange` to trigger after you set `desiredImmersiveViewingMode` to `portal`. Then, dismiss the fully/progressive immersive space and open up a window scene.

Updating the UI during transitions

The system triggers scene events named `VideoPlayerEvents.ImmersiveViewingModeWillTransition` and `VideoPlayerEvents.ImmersiveViewingModeDidTransition` at the start and end of the immersive-viewing mode transitions, respectively. Apps can listen to these scene events to turn playback controls or other UI items on or off during transitions.

Topics

Creating a video player component

`init(avPlayer: AVPlayer)`

Creates a video player component from an AV player object.

`init(videoRenderer: AVSampleBufferVideoRenderer)`

Creates a video player component from a sample buffer video renderer object.

Configuring the video player

`var isPassthroughTintingEnabled: Bool`

A Boolean value that indicates whether the passthrough camera feed is tinted, emphasizing the video content.

`var desiredViewingMode: VideoPlaybackController.ViewingMode`

The viewer's selected content-viewing mode.

Accessing video player properties

`var avPlayer: AVPlayer?`

The AV player that the component plays.

`var playerScreenSize: SIMD2<Float>`

The screen entity size of the current video player in meters.

`var screenVideoDimension: SIMD2<Float>`

The video resolution size.

`var videoRenderer: AVSampleBufferVideoRenderer?`

The component's video renderer.

`var viewingMode: VideoPlaybackController.ViewingMode?`

The current content-viewing mode for video playback.

Playing immersive media

`var desiredImmersiveViewingMode: VideoPlayerComponent.ImmersiveViewingMode`

The viewer's selected immersive-viewing mode.

`var immersiveViewingMode: VideoPlayerComponent.ImmersiveViewingMode?`

The current immersive-viewing mode.

Instance Properties

`var currentRenderingStatus: VideoPlayerComponent.RenderingStatus`

`var desiredSpatialVideoMode: VideoPlayerComponent.SpatialVideoMode`

The viewer's selected spatial video rendering mode.

`var spatialVideoMode: VideoPlayerComponent.SpatialVideoMode`

The currently active spatial video rendering mode.

Enumerations

`enum ImmersiveViewingMode`

Options for viewing the video during immersive-media playback.

`enum RenderingStatus`

`enum SpatialVideoMode`

Spatial Videos's rendering mode.

`enum VideoComfortMitigation`

Relationships

Conforms To

`Component`

See Also

Video player configurations

`enum ImmersiveViewingMode`

Options for viewing the video during immersive-media playback.

`struct VideoMaterial`

A material that supports animated textures.

`class VideoPlaybackController`

An object that controls the playback of video for a video material.

`enum ViewingMode`

Options for viewing video playback.