Structure

ImagePresentationComponent

A component that supports general image presentation.

visionOS 26.0+

```
struct ImagePresentationComponent
```

Overview

`ImagePresentationComponent` supports the presentation of three different kinds of images in RealityKit:

- A traditional 2D image.
- A *spatial photo*, which is a stereoscopic photo with additional spatial metadata, as captured on iPhone 15 Pro or later and Apple Vision Pro.
- A *spatial scene*, which is a 3D image generated from an existing 2D image or photo.

To present a 2D image or a spatial photo, create a new `ImagePresentationComponent` from a local file URL for the existing image, or from an existing `CGImageSource`.

To generate and present a spatial scene from an existing image, create a `ImagePresentationComponent.Spatial3DImage` from the image; call the `generate()` method on your new spatial 3D image; then create a new `ImagePresentationComponent` with `init(spatial3DImage:)`.

Viewing modes

By default, an `ImagePresentationComponent` presents its image in a `mono` viewing mode, regardless of what kind of image you create it with.

To request a different viewing mode, set the component's `desiredViewingMode` property, and add the component to an entity with the updated value.

To discover which viewing modes a component's image supports, query the component's `availableViewingModes` property. The `availableViewingModes` set always contains `mono` as an option.

When you create this component with a valid spatial photo, the set of available viewing modes also contains `spatialStereo` and `spatialStereoImmersive`.

If you created the component with a `ImagePresentationComponent.Spatial3DImage`, and have called the `generate()` method to generate it, the set of available viewing modes will also contain `spatial3D` and `spatial3DImmersive`.

> **Note**
>
> The component may not present the image with the `desiredViewingMode` you choose. Query `viewingMode` to check what viewing mode the component is using.

Topics

Creating a component from a 2D image or spatial photo

`init(contentsOf: URL) async throws`

Creates a new component by loading a monoscopic texture and (if present) a pair of spatial textures from the provided image file URL.

`init(imageSource: CGImageSource) async throws`

Creates a new component by loading a monoscopic texture and (if present) a pair of spatial textures from the provided image source.

Creating a component from a spatial scene

`class Spatial3DImage`

A 3D spatial scene created from a 2D image.

`init(spatial3DImage: ImagePresentationComponent.Spatial3DImage)`

Creates a new image presentation component to present a spatial 3D image.

Setting and discovering viewing modes

`struct ViewingMode`

Image content's rendering mode.

`var viewingMode: ImagePresentationComponent.ViewingMode`

The currently active viewing mode of the presented image.

`var desiredViewingMode: ImagePresentationComponent.ViewingMode`

The user-selected preferred content viewing mode.

`var availableViewingModes: Set<ImagePresentationComponent.ViewingMode>`

The currently valid viewing modes for the image being presented.

`static func supportedViewingModes(for: CGImageSource) -> Set<ImagePresentationComponent.ViewingMode>`

The viewing modes supported by the provided image source.

Retrieving the current image size

`func aspectRatio(for: ImagePresentationComponent.ViewingMode) -> Float?`

The aspect ratio of the image this component will present for the requested viewing mode.

`var screenImageDimension: SIMD2<Float>`

The image resolution of the currently presented image, in pixels.

Retrieving the current screen mesh size

`var screenHeight: Float`

The height of the screen mesh, in meters, when the image is presented in a non-immersive viewing mode.

`var presentationScreenSize: SIMD2<Float>`

The size of the screen presenting the image, with the format [width, height] in meters.

Type Methods

`static supportedViewingModes(for:)`

The viewing modes supported by the provided image source.

Relationships

Conforms To

`Component`

See Also

Visual adjustments

`struct HoverEffectComponent`

A component that applies a visual effect to a hierarchy of entities when a person looks at or selects an entity.

`struct BillboardComponent`

A component that orients an entity instance so that it continuously points toward the active camera.

`struct EnvironmentBlendingComponent`

A component that controls how an entity blends visually with objects in the local environment.

`struct LensDistortionData`

A description of estimated lens distortion that can be used to rectify images.