API Collection

Lights and cameras

Control the lighting and point of view for a scene.

Overview

In RealityKit, cameras define the point of view from which RealityKit renders a scene. In AR mode, RealityKit creates and controls the camera for you based on the location and orientation of the device. In non-AR mode, you can control the camera by adding an entity that holds one of the camera components below and changing its `Transform`.

Lighting and shadows are an essential part of creating immersive and realistic experiences. You can craft dynamic and visually appealing environments that respond naturally to the viewer's movement and other actions by strategically placing and configuring lights and cameras.

Topics

Cameras

`struct PerspectiveCameraComponent`

A component that defines a virtual camera and its controls.

`struct OrthographicCameraComponent`

A component that defines an orthographic virtual camera and its settings.

`enum CameraFieldOfViewOrientation`

The orientations that a camera's field-of-view degrees can apply.

`struct ProjectiveTransformCameraComponent`

A component that defines a virtual camera with a custom projection matrix.

`class PerspectiveCamera`

A virtual camera that establishes the rendering perspective.

Point lights

`struct PointLightComponent`

A component that defines a point light source.

Directional lights and their shadows

`struct DirectionalLightComponent`

A component that defines a directional light source.

`struct Shadow`

A directional light component that adds shadows to entities that it illuminates

`enum ShadowProjectionType`

`typealias ShadowMapCullMode`

Spotlights and their shadows

`struct SpotLightComponent`

A component that defines a spotlight source.

`struct Shadow`

A spotlight component that adds shadows to entities that it illuminates.

`enum ShadowClippingPlane`

An object that specifies the mode of a shadow clipping plane.

`typealias ShadowMapCullMode`

Image-based lights

`struct ImageBasedLightComponent`

`enum Source`

`struct ImageBasedLightReceiverComponent`

General shadows

`struct GroundingShadowComponent`

A component that controls an entity’s grounding shadow.

`struct DynamicLightShadowComponent`

A component that controls an entity’s shadow from dynamic (virtual) lights.

Environment

`class EnvironmentResource`

An environmental resource that contains background and lighting information for a scene.

`struct EnvironmentLightingConfigurationComponent`

A component that scales the amount of light that an entity receives from its environment.

`struct VirtualEnvironmentProbeComponent`

A component that provides environment lighting for entities you place within the same virtual world.

`struct Probe`

A sample of the environment around a point in a scene the system uses for environment-based lighting.

`enum Source`

Options that define the source of diffuse and specular lighting for environment lighting calculations.

Entity compliance

`protocol HasPerspectiveCamera`

An interface that enables you to configure a virtual camera that you can use to define the rendering perspective when you’re not in an AR session.

`class PointLight`

An entity that produces an omnidirectional light for virtual objects.

`protocol HasPointLight`

An interface that defines a point light source component.

`class SpotLight`

An entity that illuminates virtual content in a cone-shaped volume.

`protocol HasSpotLight`

An interface that defines a spot light source component.

`class DirectionalLight`

An entity that casts a virtual light in a particular direction.

`protocol HasDirectionalLight`

An interface that defines a directional light source component.

See Also

Scene content

{} Hello World

Use windows, volumes, and immersive spaces to teach people about the Earth.

{} Enabling video reflections in an immersive environment

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

{} Creating a spatial drawing app with RealityKit

Use low-level mesh and texture APIs to achieve fast updates to a person’s brush strokes by integrating RealityKit with ARKit and SwiftUI.

{} Generating interactive geometry with RealityKit

Create an interactive mesh with low-level mesh and low-level texture.

{} Combining 2D and 3D views in an immersive app

Use attachments to place 2D content relative to 3D content in your visionOS app.

{} Transforming RealityKit entities using gestures

Build a RealityKit component to support standard visionOS gestures on any entity.

{} Responding to gestures on an entity

Respond to gestures performed on RealityKit entities using input target and collision components.

- Models and meshes

  Display virtual objects in your scene with mesh-based models.

- Materials, textures, and shaders

  Apply textures to the surface of your scene’s 3D objects to give each object a unique appearance.

- Anchors

  Lock virtual content to the real world.

- Content synchronization

  Synchronize the contents of entities locally or across the network.

- Audio

  Create personalized and realistic spatial audio experiences.

- Videos

  Present videos in your RealityKit experiences.

- Images

  Present images and spatial scenes in your RealityKit experiences.