API Collection

Materials, textures, and shaders

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

Overview

Materials define the appearance of 3D models in RealityKit, by defining how their surfaces interact with light. Materials can also occlude objects behind them, hold shaders, or act as portals to view other worlds with `PortalMaterial`.

RealityKit includes a few material types to help you get started, including `SimpleMaterial` for basic color and textures, and `PhysicallyBasedMaterial` for realistic lighting effects such as reflections, roughness, and emission.

Topics

Simple materials

`struct SimpleMaterial`

A basic material that responds to lights in the scene.

`typealias BaseColor`

The type used to represent base color.

`typealias Texture`

The type used to represent textures.

`typealias FaceCulling`

An alias for the cull mode object that's appropriate for this material class.

`typealias TriangleFillMode`

Unlit materials

`struct UnlitMaterial`

A material that doesn't respond to lights in the scene.

`typealias BaseColor`

The type used to represent base color.

`typealias Blending`

The type used to represent opacity information.

`typealias Texture`

The type used to represent textures.

`typealias FaceCulling`

An alias for the cull mode object that's appropriate for this material class.

`typealias TextureCoordinateTransform`

An alias for the texture coordinate transform that's appropriate for this material class.

`typealias TriangleFillMode`

Realistic materials

Applying realistic material and lighting effects to entities

Enhance the appearance of objects in a RealityKit scene with Physically Based Rendering (PBR).

`struct PhysicallyBasedMaterial`

A material that simulates the appearance of real-world objects.

`struct BaseColor`

An object that defines an entity's base color.

`struct Roughness`

An object that defines the roughness of an entity's surface.

`struct Metallic`

An object that defines the reflectiveness of an entity.

`struct Normal`

An object that specifies an entity's normal map.

`enum Blending`

The object that defines the transparency of an entity.

`struct AmbientOcclusion`

An object that defines the ambient occlusion of an entity's surface.

`struct Specular`

An object that defines the specular highlights of an entity.

`struct SheenColor`

An object that defines the color of an entity's sheen.

`struct Clearcoat`

An object that defines the intensity of an entity's clear, shiny coating.

`struct ClearcoatRoughness`

An object that defines the degree to which an entity's clear, shiny coating scatters light to create soft highlights.

`struct AnisotropyLevel`

An object that defines the degree to which an entity reflects light to create stretched or oblong highlights.

`struct AnisotropyAngle`

An object used to define a material's anisotropy angle.

`struct EmissiveColor`

An object that defines the color of the light an entity emits.

`typealias TextureCoordinateTransform`

An alias for the texture coordinate transform that's appropriate for this material class.

`typealias FaceCulling`

An alias for the face culling object that's appropriate for this material class.

`typealias Texture`

The texture type to use for materials of this class.

Portals

`struct` **`PortalMaterial`**

A material that makes the mesh part a portal to a different world.

`typealias` **`FaceCulling`**

An alias for the cull mode object that's appropriate for this material class.

`typealias` **`TriangleFillMode`**

An alias for the triangle fill mode object that's appropriate for this material class.

`struct` **`PortalComponent`**

A component that turns mesh surfaces into portals to a different world.

`struct` **`WorldComponent`**

A component that defines a portal world.

`struct` **`PortalCrossingComponent`**

A component that allows entities to cross portal boundaries.

Texture resources

`class` **`TextureResource`**

A representation of a texture.

`struct` **`CreateOptions`**

An object that holds texture resource creation options.

`typealias` **`SamplingQuality`**

An object for controlling the texture-sampling quality.

`enum` **`MipmapsMode`**

An enumeration for specifying how to allocate and generate mipmaps for a texture.

`enum` **`Semantic`**

An object for specifying the intended use of a texture.

Texture drawing

{} Rendering a windowed game in stereo

Bring an iOS or iPadOS game to visionOS and enhance it.

Creating a dynamic height and normal map with low-level texture

Create a low-level texture and update its pixel data on the GPU to form a dynamic height and normal map.

`class LowLevelTexture`

A container for texture data allowing you to create and update textures using your own format.

`struct Descriptor`

An object that you use to configure new `LowLevelTexture` objects.

`class Drawable`

A drawable associated with a drawable queue

`class DrawableQueue`

A drawable queue that may be used to update a texture resource dynamically

`struct Descriptor`

Describes the texture managed by the drawable queue

Shaders

`struct ShaderGraphMaterial`

A material that comes from a shader graph in a Reality Composer Pro project, or a MaterialX shader.

`typealias FaceCulling`

An alias for the cull mode object that's appropriate for this material class.

`typealias TriangleFillMode`

An alias for the triangle fill mode object that's appropriate for this material class.

Modifying RealityKit rendering using custom materials

Write Metal shader functions to implement custom rendering effects.

`struct CustomMaterial`

A material that works with custom Metal shader functions.

`struct SurfaceShader`

The custom material's surface shader function.

`struct` `GeometryModifier`

The custom material's optional shader function that can manipulate an entity's vertex data.

`protocol` `MaterialFunction`

The abstract superclass for objects representing compute functions for RealityKit custom materials .

`class` `Program`

An object that represents the backing shader compilation required for custom materials.

`struct` `Descriptor`

An object that specifies all parameters necessary to initialize `CustomMaterial` programs

`enum` `CustomShaderStage`

Object occlusion

`struct` `OcclusionMaterial`

An invisible material that hides objects rendered behind it.

`typealias` `FaceCulling`

An alias for the cull mode object that's appropriate for this material class.

Video materials

`struct` `VideoMaterial`

A material that supports animated textures.

`typealias` `FaceCulling`

An alias for the cull mode object that's appropriate for this material class.

`typealias` `TriangleFillMode`

Custom material types

`struct` `Custom`

An object that defines the custom properties for the material.

`struct` `CustomMaterialTexture`

A texture object that you use to create custom materials.

`enum` `LightingModel`

An object that defines how the framework renders a custom material.

`struct` `BaseColor`

An object that defines an entity's base color.

`struct` `Roughness`

An object that defines how the surface of an entity scatters the light it reflects.

`struct` `Metallic`

An object that defines an entity's reflectiveness.

`struct` `Normal`

An object that stores fine surface details for an entity in an image texture.

`struct` `EmissiveColor`

An object that defines the color of the light an entity emits.

`enum` `Blending`

An object that specifies the transparency of an entity.

`struct` `Opacity`

An object that defines the transparency options for a custom material.

`struct` `AmbientOcclusion`

An object that defines an entity's exposure to ambient light.

`struct` `Specular`

An object that defines the specular highlights of an entity.

`struct` `Clearcoat`

An object that defines the intensity of an entity's clear, shiny coating.

`struct` `ClearcoatNormal`

An object that defines the clearcoat normal map texture.

`struct` `ResourceStorage`

A container for resources that will be encoded into a CustomMaterial's custom uniforms argument buffer.

`typealias` `TextureCoordinateTransform`

The object type that custom material use to hold UV texture coordinates.

`typealias FaceCulling`

The type of object used to control the removal of polygons that aren't visible to the user.

`typealias Texture`

The object type that custom materials use to hold texture properties.

Error types

`enum CustomMaterialError`

Errors generated when loading custom material functions.

Material types

`protocol Material`

A type that describes the material aspects of a mesh, like color and texture.

`typealias Color`

An alias for the color type that's appropriate for the current platform.

`typealias Parameters`

The parameter type that custom materials uses for properties the framework passes to shader functions.

`struct MaterialParameterTypes`

A set of types that material parameters can use.

`struct MaterialParameters`

`enum MaterialColorParameter`

The color parameter applied to a material.

`enum MaterialScalarParameter`

The scalar parameter applied to a material.

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.

Anchors

Lock virtual content to the real world.

Lights and cameras

Control the lighting and point of view for a scene.

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.