RealityKit / Entity

Class

Entity

An element of a RealityKit scene to which you attach components that provide appearance and behavior characteristics for the entity.

iOS 13.0+ | iPadOS 13.0+ | Mac Catalyst 13.0+ | macOS 10.15+ | tvOS 26.0+ | visionOS

```
@MainActor @preconcurrency
class Entity
```

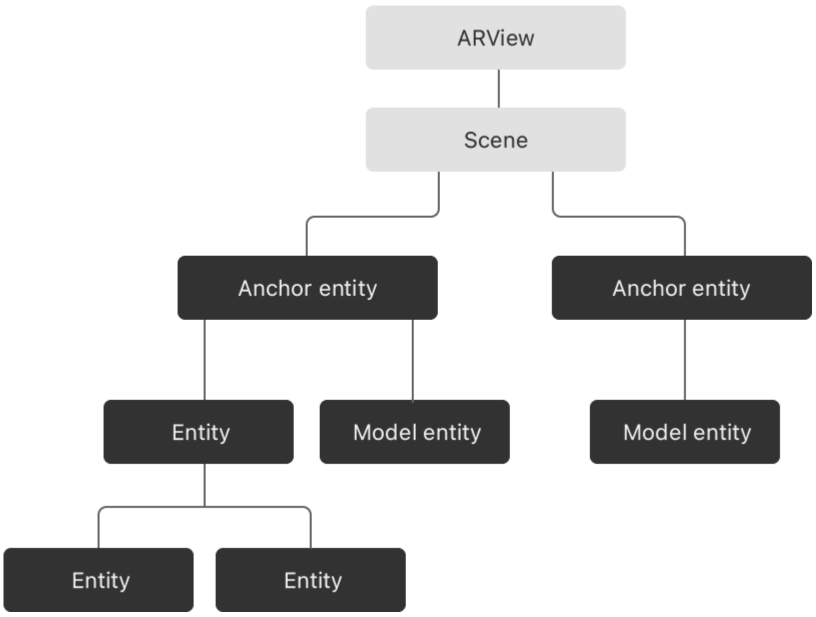
Mentioned in

- Loading Reality Composer files manually without generated code
- Improving the Accessibility of RealityKit Apps
- Adding interactivity to behaviors
- Arranging elements in a scene

Overview

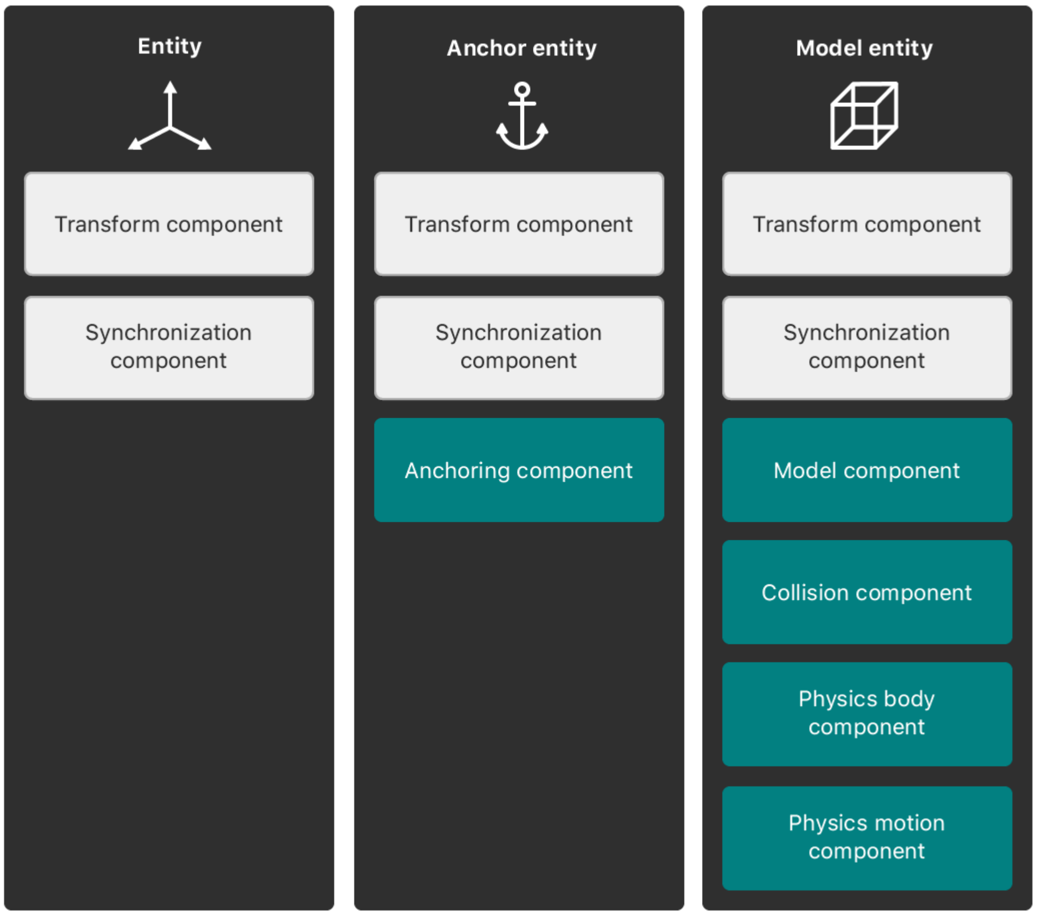
You create and configure entities to embody objects that you want to place in the real world in an AR app. You do this by adding `Entity` instances to the `Scene` instance associated with an `ARView`.

RealityKit defines a few concrete subclasses of `Entity` that provide commonly used functionality. For example, you typically start by creating an instance of `AnchorEntity` to anchor your content, and add the anchor to a scene's `anchors` collection. You might then instantiate a `ModelEntity` to represent a physical object in the scene, and add that as a child entity to the anchor. You can also create custom entities, and add them either to an anchor, or as children of another entity.



You can load predefined entities or a hierarchy of entities from your app's bundle, or from a file on disk, using the methods in Stored entities. Alternatively, you can create entities programmatically.

Entities contain components (instances conforming to the `Component` protocol) that provide appearance and behaviors for the entity. For example, the `Transform` component contains the scale, rotation, and translation information needed to position an entity in space. You store components in the entity's `components` collection, which can hold exactly one of any component type. It makes sense to have only a single `Transform` component, one `ModelComponent` (specifying the visual appearance of the entity), and so on.



All entities inherit a few common components from the `Entity` base class: the `Transform` component for spatial positioning, and `SynchronizationComponent`, which enables synchronization of the entity among networked peers. Specific types of entities define additional behaviors. For example, the model entity has components for visual appearance (`ModelComponent`), collision detection (`CollisionComponent`), and physics simulations (`PhysicsBodyComponent` and `PhysicsMotionComponent`).

Topics

Creating an entity

`init()`

Creates a new entity.

`func clone(recursive: Bool) -> Self`

Duplicates an entity to create a new entity.

`func didClone(from: Entity)`

Tells a newly cloned entity that cloning is complete.

Loading an entity from a file

Loading entities from a file

Retrieve an entity from storage on disk using a synchronous or an asynchronous load operation.

Stored entities

Manage entities that you store as assets on disk.

Creating USD files for Apple devices

Generate 3D assets that render as expected.

`convenience init(contentsOf: URL, withName: String?) async throws`

Creates an entity by asynchronously loading it from a file URL.

`convenience init(named: String, in: Bundle?) async throws`

Creates an entity by asynchronously loading it from a bundle.

`struct ReferenceComponent`

A component that can load another entity from a file.

Loading an entity from a configuration catalog

`convenience init(from: Entity.ConfigurationCatalog, configurations: [String : String]?) async throws`

Loads an entity from a configuration catalog and a dictionary of configuration choices.

`struct ConfigurationCatalog`

A collection of alternative representations of an entity you can choose from.

Positioning entities in space

`protocol HasTransform`

An interface that enables manipulating the scale, rotation, and translation of an entity.

`struct Transform`

A component that defines the scale, rotation, and translation of an entity.

`func transformMatrix(relativeTo: Entity.CoordinateSpaceReference) -> float4x4?`

Returns the 4 x 4 transform matrix of an entity relative to the given coordinate space.

`enum CoordinateSpaceReference`

Defines the coordinate space reference for transform conversion.

`enum ForwardDirection`

Defines the forward direction for an entity.

Relating entities

`struct ChildCollection`

A collection of child entities.

`protocol HasHierarchy`

An interface that provides access to a parent entity and child entities.

Managing components

`var components: Entity.ComponentSet`

All the components that an entity stores.

`struct ComponentSet`

A collection of components that an entity stores.

Inspecting an entity

`var scene: Scene?`

The scene that owns the entity.

`var name: String`

The name of the entity.

`func findEntity(named: String) -> Entity?`

Recursively searches all descendant entities for one with the given name.

`var debugDescription: String`

A human readable description of the entity.

Managing the entity’s state

`var isEnabled: Bool`

A Boolean that you set to enable or disable the entity and its descendants.

`var isEnabledInHierarchy: Bool`

A Boolean that indicates whether the entity and all of its ancestors are enabled.

`var isActive: Bool`

A Boolean that indicates whether the entity is active.

`var isAnchored: Bool`

A Boolean that indicates whether the entity is anchored.

Finding the nearest anchor

`var anchor: (any HasAnchoring)?`

The nearest ancestor entity that can act as an anchor.

Creating a collision shape

`func generateCollisionShapes(recursive: Bool)`

Creates the shape used to detect collisions between two entities that have collision components.

```
func generateCollisionShapes(recursive: Bool, static: Bool)
```

Creates the shape used to detect collisions between two entities that have collision components.

Animating an entity

```
var availableAnimations: [AnimationResource]
```

The list of animations associated with the entity.

```
func playAnimation(AnimationResource, transitionDuration: TimeInterval,
blendLayerOffset: Int, separateAnimatedValue: Bool, startsPaused: Bool,
clock: CMClockOrTimebase?) -> AnimationPlaybackController
```

Plays an animation with the specified options.

```
func playAnimation(AnimationResource, transitionDuration: TimeInterval,
blendLayerOffset: Int, separateAnimatedValue: Bool, startsPaused: Bool,
clock: CMClockOrTimebase?, handoffType: AnimationHandoffType) ->
AnimationPlaybackController
```

Plays an animation with the specified options.

```
func playAnimation(AnimationResource, transitionDuration: TimeInterval,
startsPaused: Bool) -> AnimationPlaybackController
```

Plays the given animation on the entity.

```
func stopAllAnimations(recursive: Bool)
```

Stops all playing of animations on this entity.

```
var defaultAnimationClock: CMClockOrTimebase
```

Returns the default animation clock for this entity.

```
var parameters: Entity.ParameterSet
```

Represents a reference to the parameters for a particular entity.

```
struct ParameterSet
```

Represents a reference to the parameters for a particular entity.

~~func playAnimation(named: String, transitionDuration: TimeInterval, startsPaused: Bool, recursive: Bool) -> AnimationPlaybackController~~

Plays all the animations with the given name on the entity.

Deprecated

`var bindableValues: BindableValuesReference`

`subscript(BindTarget.EntityPath) -> Entity?`

Resolves the entity from the given entity path.

Animating and controlling characters

`var characterController: CharacterControllerComponent?`

The character controller component for the entity.

`var characterControllerState: CharacterControllerStateComponent?`

The character controller state for the entity.

```
func moveCharacter(by: SIMD3<Float>, deltaTime: Float, relativeTo:
Entity?, collisionHandler: ((CharacterControllerComponent.Collision) ->
Void)?) -> CharacterControllerComponent.CollisionFlags
```

Moves the character along a specified vector over a period of time.

`func teleportCharacter(to: SIMD3<Float>, relativeTo: Entity?)`

Moves the character instantly to a new position.

Playing audio

`func playAudio(AudioResource) -> AudioPlaybackController`

Prepares and plays a new audio playback instance on this entity.

```
func playAudio(configuration: AudioGeneratorConfiguration, Audio.
GeneratorRenderHandler) throws -> AudioGeneratorController
```

Prepares and plays a real-time audio playback instance.

```
func prepareAudio(configuration: AudioGeneratorConfiguration, Audio.
GeneratorRenderHandler) throws -> AudioGeneratorController
```

Prepares a real-time audio playback instances.

`func prepareAudio(AudioResource) -> AudioPlaybackController`

Prepares an audio resource for playback.

`func stopAllAudio()`

Stops playback for all audio on this entity.

`var spatialAudio: SpatialAudioComponent?`

The component that configures the spatial rendering of sounds from this entity.

`var ambientAudio: AmbientAudioComponent?`

The component that configures the ambient rendering of sounds from this entity.

`var channelAudio: ChannelAudioComponent?`

The component that configures the channel-based rendering of sounds from this entity.

Saving an entity and its descendants

`func write(to: URL) async throws`

Exports the entity as a RealityKit file to a location in the file system.

Configuring accessibility features

Improving the Accessibility of RealityKit Apps

Incorporate assistive technologies in your augmented reality app.

`var isAccessibilityElement: Bool`

A Boolean value indicating whether the receiver is an accessibility element that an assistive application can access.

`var accessibilityLabelKey: LocalizedStringResource?`

A succinct label that identifies the entity, in a localized string key.

`var accessibilityCustomActions: [LocalizedStringResource]`

An array of custom actions supported by the entity, identified by their localized string key.

`var accessibilityCustomContent: [AccessibilityComponent.CustomContent]`

The Custom Content API is useful for delivering accessibility information from complex data sets to your users in measured portions. Using this API allows you to leverage assistive technologies to present only the accessible content your app’s users need, when they need it.

`var accessibilityCustomRotors: [AccessibilityComponent.RotorType]`

An array of supported rotors.

`var accessibilityLabelKey: LocalizedStringResource?`

A succinct label that identifies the entity, in a localized string key.

`var accessibilitySystemActions: AccessibilityComponent.SupportedActions`

The set of supported accessibility actions.

`var accessibilityTraits: UIAccessibilityTraits`

The combination of accessibility traits that best characterize the entity.

`var accessibilityValue: LocalizedStringResource?`

A localized string key that represents the current value of the entity.

~~`var accessibilityDescription: String?`~~

A longer description of the entity for use by assistive technologies.

Deprecated

~~`var accessibilityLabel: String?`~~

A succinct label that identifies the purpose of the image.

Deprecated

~~`var accessibilityDescription: String?`~~

A longer description of the entity for use by assistive technologies.

Deprecated

Comparing entities

`static func == (Entity, Entity) -> Bool`

Indicates whether two entities are equal.

`func hash(into: inout Hasher)`

Hashes the essential components of the entity by feeding them into the given hash function.

Observing entities

`struct Observable`

An observable interface to an entity's properties and components, enabling reactive updates using Swift's Observation framework.

`var observable: Entity.Observable`

Initializers

`convenience(components:)`

Creates an entity with multiple components.

`convenience init(from: Data, named: String?) async throws`

Creates an entity by asynchronously loading it from the in-memory contents of a file stored in a Data object.

Instance Properties

`var pins: EntityGeometricPins`

The entity’s geometric pins.

Instance Methods

`func applyTapForBehaviors() -> Bool`

Apply a tap to an Entity or one of its ancestors to trigger a RealityComposer behavior if one is present.

`func attach(GeometricPin?, to: GeometricPin)`

Attach an entity to a target pin owned by another entity with an optional specified source pin This utility function has the same effect of adding an AttachedTransformComponent created with the same parameter to the entity you are calling upon

Type Methods

`static func animate(Animation, body: () -> Void, completion: (() -> Void)?)`

Default Implementations

- CustomDebugStringConvertible Implementations
- Equatable Implementations
- Hashable Implementations

Relationships

Inherited By

`AnchorEntity`
`BodyTrackedEntity`
`DirectionalLight`
`HasAnchoring`
`HasHierarchy`
`HasPhysicsMotion`
`HasSceneUnderstanding`
`HasSynchronization`
`HasTransform`
`ModelEntity`
`PerspectiveCamera`
`PointLight`
`SpotLight`
`TriggerVolume`
`ViewAttachmentEntity`

Conforms To

`CoordinateSpace3D`
`CoordinateSpace3DFloat`
`Copyable`
`CustomDebugStringConvertible`
`Equatable`
`EventSource`
`HasHierarchy`
`HasSynchronization`
`HasTransform`
`Hashable`
`Identifiable`
`Observable`
`RealityCoordinateSpace`
`Sendable`
`SendableMetatype`

See Also

Essentials

Understanding the modular architecture of RealityKit

Learn how everything fits together in RealityKit.

{} Building an immersive experience with RealityKit

Use systems and postprocessing effects to create a realistic underwater scene.

`protocol Component`

A representation of a geometry or a behavior that you apply to an entity.