Sample Code

Configuring Collision in RealityKit

Use collision groups and collision filters to control which objects collide.

Download

iOS 18.0+ | iPadOS 18.0+ | Xcode 16.0+

Overview

By default, in RealityKit physics simulations, all physics objects collide with all other physics objects. Often, this is the desired behavior, but sometimes, you need to specify that certain objects won't collide with certain other objects. In RealityKit, that's accomplished using `CollisionGroup` in combination with `CollisionFilter`.

This sample code project shows how to use collision groups and collision filters together to take precise control over which entities collide in a RealityKit scene. In this sample, entities can be dragged around the scene. As they are dragged, they will collide with unlike entities, but ignore (or pass through) like entities. So, for example, if you drag a sphere into a cube, they will collide, causing the cube to move. However, if you drag a sphere into another sphere, they'll pass through each other.

Collision filters define which entities will collide with which other entities based on the other entity's collision group. That means all entities that share the same filter are part of the same group or groups and have the same collision properties. More than one filter can be created for each collision group, however, so not all members of the same collision group necessarily collide with the same other objects.

`CollisionGroup` conforms to `OptionSet`, which means you typically create individual groups by assigning each group a different bit flag value. If each group has a distinct bit representing it, that allows you to configure collision filters that collide with, or ignore any combination of individual collision groups using `OptionSet`'s various methods.

Here is how the four collision groups in this sample project are defined:

```
    let planeGroup = CollisionGroup(rawValue: 1 << 0)
    let cubeGroup = CollisionGroup(rawValue: 1 << 1)
    let beveledCubeGroup = CollisionGroup(rawValue: 1 << 2)
    let sphereGroup = CollisionGroup(rawValue: 1 << 3)
```

You don’t assign collision groups directly to entities. Instead, you create filters that define the collision properties for individual entities. Assigning a collision filter to an entity automatically places that entity into the collision filter’s group or groups.

For example, here is how this sample creates the filter for the `cubeGroup`, used by cubes so they collide with everything except other cubes:

```
    let cubeMask = CollisionGroup.all.subtracting(cubeGroup)
    let cubeFilter = CollisionFilter(group: cubeGroup,
                                     mask: cubeMask)
```

Note

Group membership does not have to be exclusive. Although it’s common to pass an individual group to `CollisionFilter`‘s `group` parameter as is done in this project, but you don’t have to. You can pass a `group` parameter that includes more than one individual group, just like you can with the `mask` parameter, and any entity that gets assigned that filter will be a member of all of the specified groups.

After creating a collision filter, assign it to all the entities that should use that filter, like this:

```
    cube1.collision?.filter = cubeFilter
    cube2.collision?.filter = cubeFilter
    cube3.collision?.filter = cubeFilter
```

Getting Started

Requires Xcode 16.2, iOS 18, and an iOS device with an A9 or later processor. ARKit is not supported in iOS Simulator.

See Also

Collision shapes and groups

{} Simulating physics with collisions in your visionOS app

Create entities that behave and react like physical objects in a RealityKit view.

`struct CollisionComponent`

A component that gives an entity the ability to collide with other entities that also have collision components.

`enum Mode`

A mode that dictates how much collision data is collected for a given entity.

`class ShapeResource`

A representation of a shape.

`enum ShapeResourceError`

`struct CollisionGroup`

A bitmask used to define the collision group to which an entity belongs.

`struct CollisionFilter`

A set of masks that determine whether entities can collide during simulations.

`class TriggerVolume`

An invisible 3D shape that detects when objects enter or exit a given region of space.