Sample Code

Controlling Entity Collisions in RealityKit

Create collision filters to control which objects collide.

Download

iOS 13.0+ | iPadOS 13.0+ | Xcode 12.0+

Overview

By default, objects that participate in RealityKit simulations collide with all other participating objects in the scene. To prevent particular objects from colliding with specific other objects, use `CollisionGroup` in combination with `CollisionFilter`. This sample code project shows how to use collision groups and collision filters to take precise control over which entities collide in a scene.

In this sample, the user can drag entities around the scene. As they do so, those entities collide with other entities that have different shapes or colors, but ignore (or pass through) entities of the same shape and color. For example, if the user drags a sphere into a cube, the entities will collide, causing the cube to move. However, if the user drags a red sphere into another red sphere, the entities will pass through each other.

Configure the Sample Code Project

The sample code project requires Xcode 12.0, iOS 14.0, and an iOS device with an A9 or later processor. ARKit isn't supported in iOS Simulator.

Define Collision Groups

Collision groups provide a way to group entities in a scene. Each group represents entities that respond the same way during a collision. Because `CollisionGroup` conforms to `OptionSet`,

groups can be combined in many different ways to create new groups. The sample project defines a set of groups that represent each distinct type of objects, and assigns each one a different bit flag.

The sample project defines four collision groups, as shown here:

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

Collision groups aren't assigned directly to entities. Instead, collision filters define the collision properties for entities by specifying which groups the object collides with. Assigning a collision filter to an entity automatically places that entity into the collision filter's group or groups.

Create a Collision Filter

Collision filters define the entities that collide with other entities based on group membership. 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.

The sample project, for example, creates a filter for the `cubeGroup`, which defines the filter that causes cubes to collide with everything except other cubes:

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

> **Note**
>
> Group membership doesn't have to be exclusive. A `CollisionFilter`'s `group` parameter is an `OptionSet`, so it can include any combination of defined groups. Any entity that gets assigned a filter will be a member of all of that filter's groups.

The sample code project assigns `cubeFilter` to all the cube entities, which places all of the cube entities into the `cubeGroup` collision group, like this:

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

```
    cube2.collision2.filter = cubeFilter
```

All cubes are part of the `cubeGroup` because that is the group assigned to their filter.