About Collisions and Contacts

Learn how to set up nodes for collision detection.

Overview

SpriteKit supports two kinds of interaction between physics bodies that come into contact or attempt to occupy the same space:

>> A contact is used when you need to know that two bodies are touching each other. In most cases, you use contacts when you need to make gameplay changes when a collision occurs.

>> A collision is used to prevent two objects from interpenetrating each other. When one body strikes another body, SpriteKit automatically computes the results of the collision and applies impulse to the bodies in the collision.

Interaction Limits

Your game configures the physics bodies in the scene to determine when collisions should occur and when interactions between physics bodies require additional game logic to be performed. Limiting these interactions is not only important for defining your game's logic, it is also necessary in order to get good performance from SpriteKit.

SpriteKit uses two mechanisms to limit the number of interactions in each frame:

>> Edge-based physics bodies never interact with other edge-based physics bodies. This means that even if you move them by repositioning the nodes, edge-based physics bodies never collide or contact each other.

>> Every physics body is categorized. Categories are defined by your app; each scene can have up to 32 categories. When you configure a physics body, you define which categories it belongs to and which categories of bodies it should interact with. You define contacts and collisions separately.

The following code creates a red and a blue ball, and a ground object constructed from a series of points. By giving the red ball a `collisionBitMask` that matches the ground's `categoryBitMask`, you make the red ball bounce off the ground. However, the blue ball's `collisionBitMask` is set to a different value and doesn't interact with the ground.

```
let ballRadius: CGFloat = 20
let redBall = SKShapeNode(circleOfRadius: ballRadius)
redBall.fillColor = .red
redBall.position = CGPoint(x: 280, y: 320)
redBall.physicsBody = SKPhysicsBody(circleOfRadius: ballRadius)

let blueBall = SKShapeNode(circleOfRadius: ballRadius)
blueBall.fillColor = .blue
blueBall.position = CGPoint(x: 360, y: 320)
blueBall.physicsBody = SKPhysicsBody(circleOfRadius: ballRadius)

var splinePoints = [CGPoint(x: 0, y: 300),
                    CGPoint(x: 100, y: 50),
                    CGPoint(x: 400, y: 110),
                    CGPoint(x: 640, y: 20)]
let ground = SKShapeNode(splinePoints: &splinePoints,
                         count: splinePoints.count)
ground.physicsBody = SKPhysicsBody(edgeChainFrom: ground.path!)
ground.physicsBody?.restitution = 0.75

redBall.physicsBody?.collisionBitMask = 0b0001
blueBall.physicsBody?.collisionBitMask = 0b0010
ground.physicsBody?.categoryBitMask = 0b0001
```

The following image shows the paths taken by both balls when the code above is executed.