Searching the World for Physics Bodies

Cast a ray to find the physics bodies in the scene that intersect it.

Overview

Sometimes, it is necessary to find physics bodies in a scene. For example, you might need to:

- >> Discover whether a physics body is located in a region of the scene.
- >> Detect when a physics body (such as the one controlled by the player) crosses a particular line.
- >> Trace the line of sight between two physics bodies, to see whether another physics body, such as a wall, is interposed between the two objects.
- >> Occlude an audio node if a body is between it and a scene's listener node.

In some cases, you can implement these interactions using the collisions and contacts system. For example, to discover when a physics body enters a region, you could create a physics body and attach it to an invisible node in the scene. Then, configure the physics body's collision mask so that it never collides with anything, and its contact mask to detect the physics bodies you are interested in. Your contact delegate is called when the desired interactions occur.

However, it's not easy to implement concepts such as line of sight using this design. To implement these, you use the scene's physics world. With the physics world, you can search for all physics bodies along a ray or physics bodies that intersect a particular point or rectangle.

Cast a Ray from the Center of the Scene

An example illustrates the basic technique. The following code shows one possible implementation of a line-of-sight detection system. It casts a ray from the origin of the scene in a particular direction, searching for the nearest physics body along the ray. If it finds a physics body, it tests the category mask to see whether this is a target it should attack. If it sees a target designated for attack, it shoots the cannon.

```
func isTargetVisibleAtAngle(angle: CGFloat, distance: CGFloat) -> Bool {
    let rayStart = CGPoint.zero
    let rayEnd = CGPoint(x: distance * cos(angle),
                         y: distance * sin(angle))

    let body = scene.physicsWorld.body(alongRayStart: rayStart, end: rayEnd)

    return body?.categoryBitMask == targetCategory
}
func attackTargetIfVisible() {
    if isTargetVisibleAtAngle(angle: cannon.zRotation, distance: 512) {
        shootCannon()
    }
}
```

Try Other Search Methods

Another way to implement the same behavior is to set the starting and ending positions of the ray to those of two physics bodies in your scene. For example, you might use the location of the player's game object as one position and the position of an enemy unit as the other position. You can also perform searches for physics bodies that intersect a point or rectangle using the `body(at:)` and `body(in:)`methods. Sometimes you can't make a simple determination based on the closest physics body within the scene. For example, in the logic of your game, you might decide that not all physics bodies block the line of sight. In this case, you need to enumerate all of the physics bodies along the ray using the `enumerateBodies(alongRayStart:end:using:)` method. You supply a block that is called once for each body along the ray. You can then use this information to make a more informed decision about whether the line of sight exists to a target.