Getting Started with Physics Bodies

Create and assign a physics body to enable physics.

Overview

An SKPhysicsBody object defines the shape and simulation parameters for a physics body in the system. When the scene simulates physics, it performs the calculations for all physics bodies connected to the scene tree. So, you create an SKPhysicsBody object, configure its properties, and then assign it to a node's physicsBody property.

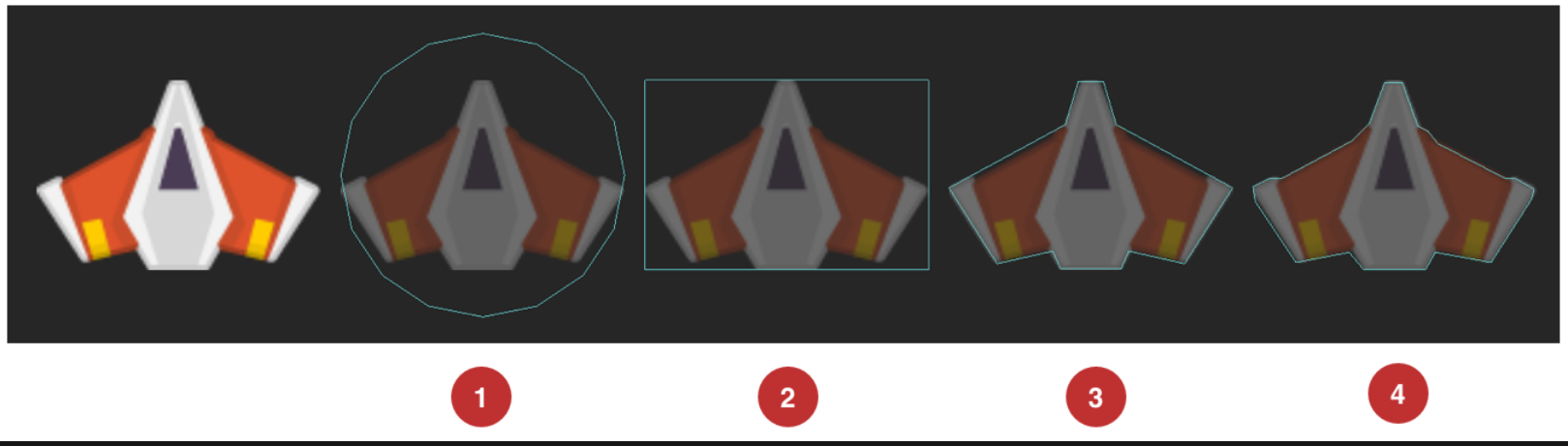
There are three kinds of physics bodies:

>> A dynamic volume simulates a physical object with volume and mass that can be affected by forces and collisions in the system. Use dynamic volumes to represent items in the scene that need to move around and collide with each other.

>> A static volume is similar to a dynamic volume, but its velocity is ignored and it is unaffected by forces or collisions. However, because it still has volume, other objects can bounce off it or interact with it. Use static volumes to represent items that take up space in the scene, but that should not be moved by the simulation. For example, you might use static volumes to represent the walls of a maze. While it is useful to think of static and dynamic volumes as distinct entities, in practice these are two different modes you can apply to any volume-based physics body. This can be useful because you can selectively enable or disable effects for a body.

>> An edge is a static volume-less body. Edges are never moved by the simulation and their mass doesn't matter. Edges are used to represent negative space within a scene (such as a hollow spot inside another entity) or an uncrossable, invisibly thin boundary. For example, edges are frequently used to represent the boundaries of your scene. The main difference between an edge and a volume is that an edge permits movement inside its own boundaries, while a volume is considered a solid object. If edges are moved through other means, they only interact with volumes, not with other edges.

SpriteKit provides a few standard shapes—those based on arbitrary paths and those generated from the alpha channel of a texture. The following figure shows the shapes available:



The following code shows how to generate the four physics bodies called out above:

```
let spaceShipTexture = SKTexture(imageNamed: "spaceShip.png")

// Spaceship 1: circular physics body
let circularSpaceShip = SKSpriteNode(texture: spaceShipTexture)
circularSpaceShip.physicsBody = SKPhysicsBody(circleOfRadius:
max(circularSpaceShip.size.width / 2,

circularSpaceShip.size.height / 2))

// Spaceship 2: rectangular physics body
let rectangularSpaceShip = SKSpriteNode(texture: spaceShipTexture)
rectangularSpaceShip.physicsBody = SKPhysicsBody(rectangleOf: CGSize(width:
circularSpaceShip.size.width,
                                                                     height:
circularSpaceShip.size.height))

// Spaceship 3: polygonal physics body
let polygonalSpaceShip = SKSpriteNode(texture: spaceShipTexture)
let path = CGMutablePath()
path.addLines(between: [CGPoint(x: -5, y: 37), CGPoint(x: 5, y: 37), CGPoint(x: 10,
y: 20),
                        CGPoint(x: 56, y: -5), CGPoint(x: 37, y: -35), CGPoint(x:
15, y: -30),
                        CGPoint(x: 12, y: -37), CGPoint(x: -12, y: -37), CGPoint(x:
-15, y: -30),
                        CGPoint(x: -37, y: -35), CGPoint(x: -56, y: -5), CGPoint(x:
-10, y: 20),
                        CGPoint(x: -5, y: 37)])
path.closeSubpath()
polygonalSpaceShip.physicsBody = SKPhysicsBody(polygonFrom: path)

// Spaceship 4: physics body using texture's alpha channel
let texturedSpaceShip = SKSpriteNode(texture: spaceShipTexture)
texturedSpaceShip.physicsBody = SKPhysicsBody(texture: spaceShipTexture,
                                              size: CGSize(width:
circularSpaceShip.size.width,
                                                           height:
circularSpaceShip.size.height))
```

The shape of a physics body affects performance. A circular physics body offers the best performance and can be significantly faster than other physics bodies. If your simulation contains many physics bodies, circular bodies are the best solution. Rectangular and polygonal shapes improve collision accuracy with reduced speed. Physics bodies created from the alpha channel of a texture offer the best fidelity at the highest performance cost.

Create and Assign Physics Bodies

A physics body is created by calling one of the SKPhysicsBody class methods. Each class method defines whether a volume-based or edge-based body is being created and what shape it has.

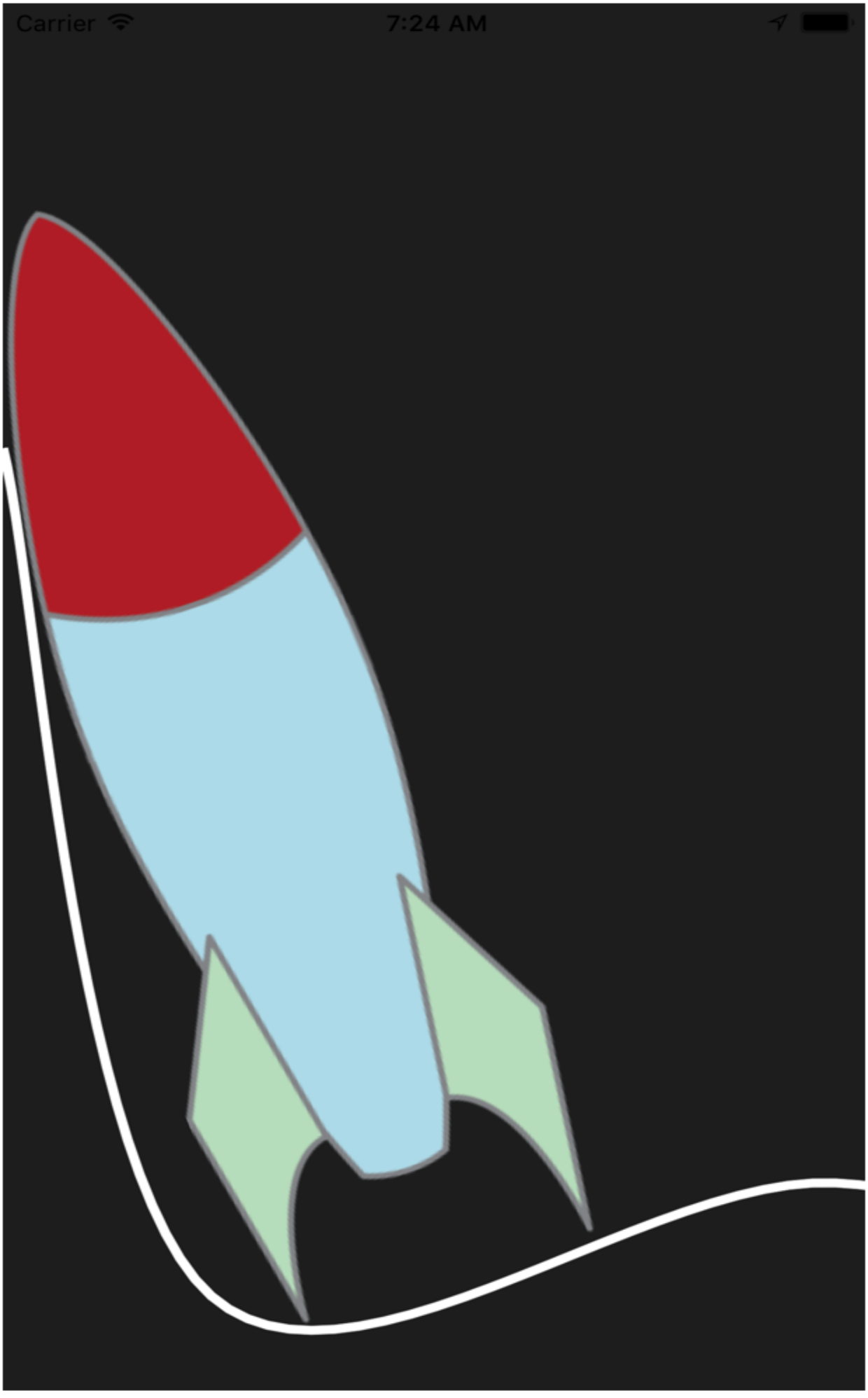
As soon as a node has its physics body set and is added to a scene, it begins to participate in the physics simulation. The following code shows how to create a simple scene containing a shape node that is the curvy ground and a sprite node—the rocket ship. The shape node's physics body is created from its path and the sprite node's physics body is created from the texture automatically generated by SpriteKit when the node was initialized. Because the ground node has its isDynamic set to false, it is not affected by gravity and remains stationary.

```
// Create the rocket ship node and physics body
let spriteNode = SKSpriteNode(imageNamed: "rocketShip")
spriteNode.position = CGPoint(x: 320, y: 320)
spriteNode.physicsBody = SKPhysicsBody(texture: spriteNode.texture!,
                                       size: spriteNode.texture!.size())
spriteNode.physicsBody?.usesPreciseCollisionDetection = true

// Create the ground node and physics body
var splinePoints = [CGPoint(x: 0, y: 500),
                    CGPoint(x: 100, y: 50),
                    CGPoint(x: 400, y: 110),
                    CGPoint(x: 640, y: 20)]
let ground = SKShapeNode(splinePoints: &splinePoints,
                         count: splinePoints.count)
ground.lineWidth = 5
ground.physicsBody = SKPhysicsBody(edgeChainFrom: ground.path!)
ground.physicsBody?.restitution = 0.75
ground.physicsBody?.isDynamic = false

// Add the two nodes to the scene
scene.addChild(spriteNode)
scene.addChild(ground)
```

After running for a few moments, the above code creates a scene that looks like the following illustration—the sprite node has come to rest in one of the valleys of the curvy ground line.


Carrier
7:24 AM