

207: Sprite Kit

Part 2: Demo Instructions

Copyright © 2014 Razeware LLC.

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written per- mission of the copyright owner.

This book and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connec- tion with the software or the use or other dealings in the software.

All trademarks and registered trademarks appearing in this book are the property of their respective owners.

# Basic Gameplay

In this demo, you will add some basic gameplay to the game Drop Charge.

The steps here will be explained in the demo, but here are the raw steps in case you miss a step or get stuck.

## Step 1: Add Title

At the top of **GameScene.swift**, add to bottom of the list of properties:

var title: SKSpriteNode!

In setupLevel(), add the following:

title = SKSpriteNode(imageNamed: "DropCharge\_title")

fgNode.addChild(title)

## Step 2: Position Title

In setupLevel(), add the following:

title.position = CGPoint(x: size.width/2, y: size.height \* 0.7)

## Step 3: Add Player

At the top of the file, add to the bottom of the list of properties:

let player = SKSpriteNode(imageNamed: "player01\_fall\_1.png")

In setupPlayer(), add the following:

player.position = CGPoint(x: size.width / 2, y: 80)

fgNode.addChild(player)

## Step 4: Add Bomb

At the top of the file, add to the bottom of the list of properties:

let bomb = SKSpriteNode(imageNamed: "bomb\_1")

In switchToWaitingForBomb(), implement the “Scale out title”, “Add bomb”, and “Bounce bomb” sections as follows:

// Scale out title

let scale = SKAction.scaleTo(0, duration: 0.5)

title.runAction(scale)

// Add bomb

bomb.position = player.position

fgNode.addChild(bomb)

// Bounce bomb

let scaleUp = SKAction.scaleTo(1.25, duration: 0.25)

let scaleDown = SKAction.scaleTo(1.0, duration: 0.25)

let sequence = SKAction.sequence([scaleUp, scaleDown])

let repeat = SKAction.repeatActionForever(sequence)

bomb.runAction(repeat)

## Step 5: Z Positioning

Add to the bottom of setupPlayer():

player.zPosition = ForegroundZ.Player.rawValue

In switchToWaitingForBomb(), add to the bottom of the “Add bomb” section:

bomb.zPosition = ForegroundZ.Bomb.rawValue

## Step 6: Switch to Playing

In switchToWaitingForBomb(), implement the “Switch to playing state” section as follows:

// Switch to playing state

runAction(SKAction.sequence([

SKAction.waitForDuration(2.0),

SKAction.runBlock(switchToPlaying)

]))

In switchToPlaying(), implement the “Stop bomb” section as follows:

bomb.removeFromParent()

## Step 7: Add Player Physics Body

Add to the bottom of setupPlayer():

player.physicsBody = SKPhysicsBody(circleOfRadius: player.size.width / 2)

player.physicsBody!.dynamic = false

player.physicsBody!.allowsRotation = false

player.physicsBody!.categoryBitMask = PhysicsCategory.Player

player.physicsBody!.collisionBitMask = 0

In switchToPlaying(), implement the “Start player movement” section as follows:

// Start player movement

player.physicsBody!.dynamic = true

## Step 8: Boost Player

In switchToPlaying(), add to the bottom of the “Start player movement” section:

superBoostPlayer()

In setPlayerVelocity(), add the following:

player.physicsBody!.velocity = CGVector(

dx: player.physicsBody!.velocity.dx,

dy: max(player.physicsBody!.velocity.dy, amount))

## Step 9: Contact Detection

In setupPhysics(), add the following:

physicsWorld.contactDelegate = self

At the top of the file, mark GameScene as implementing SKPhysicsContactDelegate:

class GameScene: SKScene, SKPhysicsContactDelegate {

In didBeginContact(), implement the four cases as follows:

case PhysicsCategory.CoinNormal:

if let coin = other.node as? SKSpriteNode {

coin.removeFromParent()

jumpPlayer()

}

case PhysicsCategory.CoinSpecial:

if let coin = other.node as? SKSpriteNode {

coin.removeFromParent()

boostPlayer()

}

case PhysicsCategory.PlatformNormal:

if let platform = other.node as? SKSpriteNode {

if player.physicsBody!.velocity.dy < 0 {

jumpPlayer()

}

}

case PhysicsCategory.PlatformBreakable:

if let platform = other.node as? SKSpriteNode {

if player.physicsBody!.velocity.dy < 0 {

platform.removeFromParent()

jumpPlayer()

}

}

## Step 10: Player Movement

In handlePlayingTouches(), uncomment the following code:

let touchTarget = touch.locationInNode(self)

let xVelocity = touchTarget.x < player.position.x ?

CGFloat(-150.0) : CGFloat(150.0)

player.physicsBody!.velocity =

CGVector(dx: xVelocity, dy: player.physicsBody!.velocity.dy)

In updatePlayer(), uncomment the following code *if you have a device to test on*:

// Set velocity based on core motion

player.physicsBody?.velocity = CGVector(dx: xAcceleration \* 400.0, dy: player.physicsBody!.velocity.dy)

In updatePlayer(), uncomment the following code whether you have a device or not:

// Wrap player around edges of screen

if player.position.x < -player.size.width/2 {

player.position.x = size.width + player.size.width/2

}

else if player.position.x > size.width + player.size.width/2 {

player.position.x = -player.size.width/2

}

## Step 11: Camera Movement

In updateCamera(), uncomment the following code:

let target = player.position

var targetPosition = CGPoint(

x: worldNode.position.x,

y: -(target.y - size.height \* 0.4))

var newPosition = targetPosition

self.fgNode.position = newPosition

self.mgNode.position = newPosition

self.bgNode.position = newPosition

## Step 12: Parallax Scrolling

In updateCamera(), change the last 2 lines as follows:

self.mgNode.position = CGPoint(x: newPosition.x/5.0, y: newPosition.y/5.0)

self.bgNode.position = CGPoint(x: newPosition.x/10.0, y: newPosition.y/10.0)

Congrats, at this time you should have the basic gameplay complete, and learned a lot about Sprite Kit along the way! You are ready to move on to the lab.