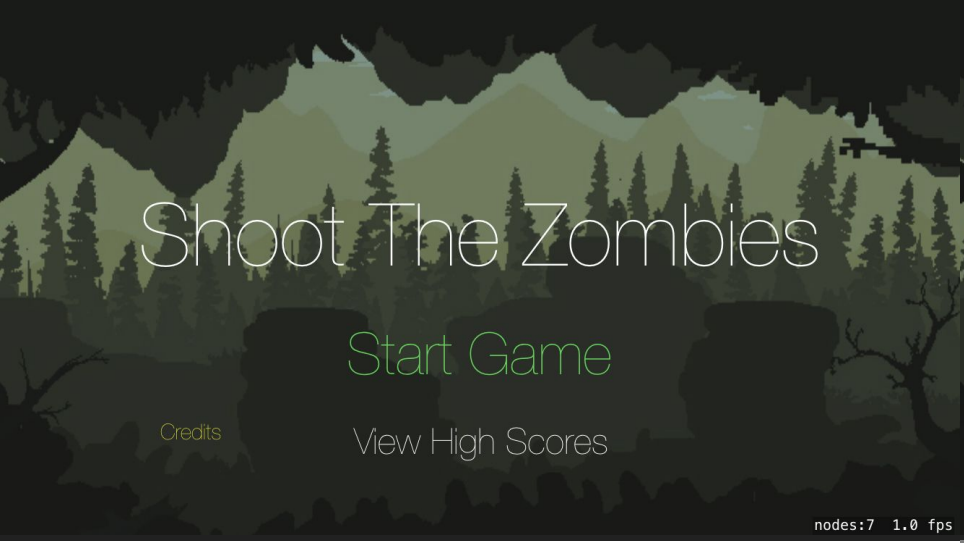
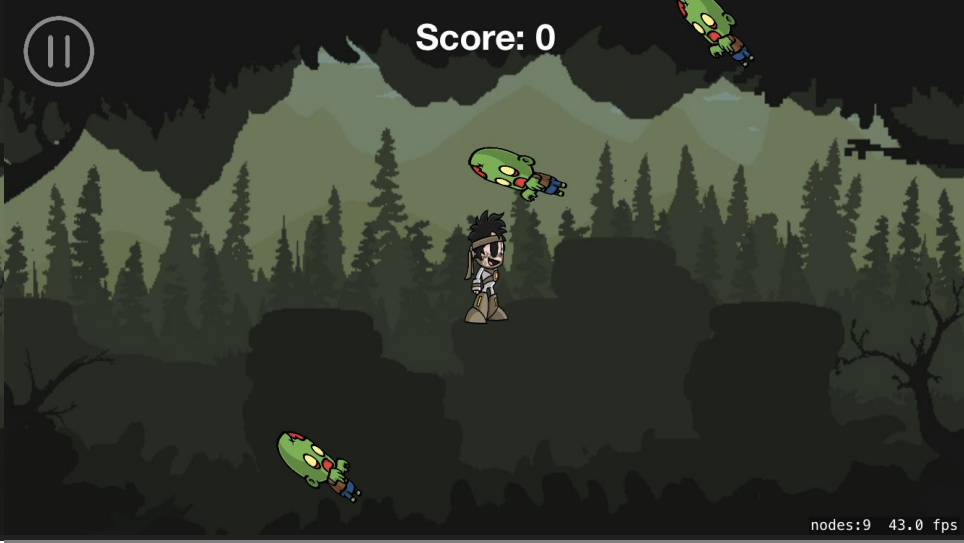
Shoot The Zombies

By Josh Eres



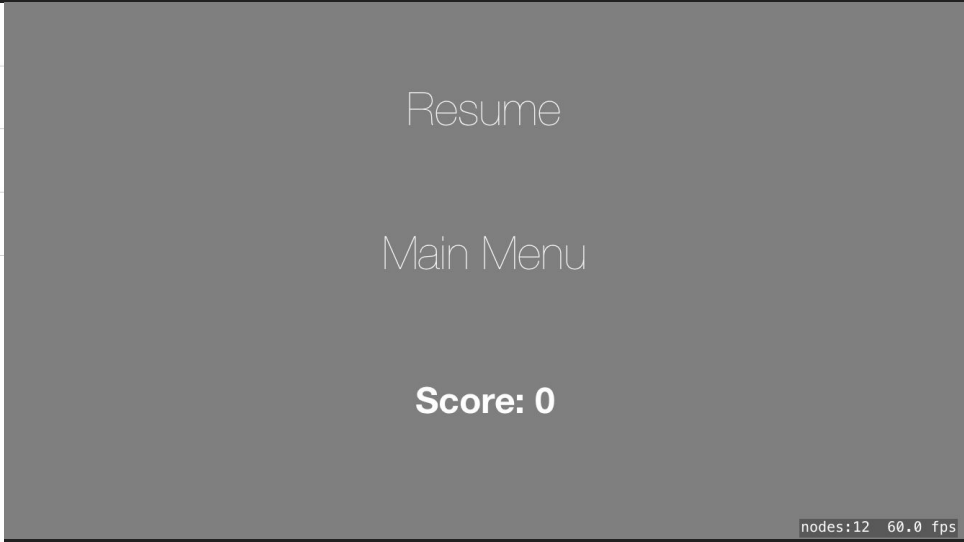


Back to Menu

High Score - Player: 180

Player - Score: 40

Player - Score: 0



```swift
class OpeningScene: SKScene {


    override func didMove(to view: SKView) {
        let startGame: SKSpriteNode = SKSpriteNode()
        startGame.name = "startGame"
        startGame.position = CGPoint(x: 0.0, y: -60.0)
        startGame.size = CGSize(width: 190, height: 45)

        startGame.zPosition = CGFloat(20.0)


        addChild(startGame)

        let highScoresLabel:SKLabelNode = SKLabelNode(text: "View High Scores")
        highScoresLabel.name = "high scores label"
        highScoresLabel.fontSize = 25
        highScoresLabel.position = CGPoint(x: 0, y: -130)
        highScoresLabel.zPosition = 30
        addChild(highScoresLabel)

        let creditsLabel: SKLabelNode = SKLabelNode(text: "Credits")
        creditsLabel.name = "credits label"
        creditsLabel.fontSize = 15
        creditsLabel.fontColor = UIColor.yellow
        creditsLabel.position = CGPoint(x: -200, y: -120)
        creditsLabel.zPosition = 30
        addChild(creditsLabel)

    }

```

```
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        for touch in touches {
            let location = touch.location(in: self)
            let touchedNode = self.atPoint(location)

            if touchedNode.name == "startGame" {
                print("Start game tapped")

                if let scene = SKScene(fileNamed: "GameScene") {
                    scene.scaleMode = .resizeFill
                    self.view?.presentScene(scene)
                }
            } else if touchedNode.name == "high scores label" {
                print("High scores tapped")

                // Initialize HighScoreTableViewController
                let highScoreVC = HighScoreTableViewController()

                // Embed HighScoreTableViewController in a navigation controller
                let navController = UINavigationController(rootViewController: highScoreVC)
                navController.modalPresentationStyle = .fullScreen

                // Present the navigation controller
                self.view?.window?.rootViewController?.present(navController, animated: true, completion: nil)
            } else if(touchedNode.name == "credits label"){


                view!.presentScene(SKScene(fileNamed: "CreditsScene"))


            }
        }
    }
```

```swift
import CoreMotion
import SpriteKit

class Zombie: SKSpriteNode{}
class Player: SKSpriteNode{}
class Bullet: SKSpriteNode{}

enum CollisionType: UInt32 {
    case player = 1
    case playerBullet = 2
    case enemy = 4

}
let model = GameModel()
```

0000 0000 0000 0000 0000 0000 0000 0001
0000 0000 0000 0000 0000 0000 0000 0010
0000 0000 0000 0000 0000 0000 0000 0100

```swift
class GameScene: SKScene, SKPhysicsContactDelegate, ObservableObject {
    var motionManager: CMMotionManager?
    var lastSpawnTimeInterval: TimeInterval = 0

    let player: Player = Player(imageNamed: "Main Character")
    let scoreLabel = SKLabelNode(text: "Score: \(model.score)")
    let gameNode = SKNode()
    let pauseNode = SKNode()
    let pauseScoreLabel = SKLabelNode(text: "Score: 0")
```

```swift
    override func didMove(to view: SKView) {
        addChild(gameNode)
        print("test")
        let pauseButton:SKSpriteNode = SKSpriteNode(imageNamed: "PauseWhite")
        pauseButton.name = "pauseButton"
        pauseButton.position = CGPoint(x: -295.0, y: 150.0)
        pauseButton.size = CGSize(width: 50, height: 50)
        pauseButton.zPosition = 20
        pauseButton.alpha = 0.5

        scoreLabel.name = "score label"
        scoreLabel.zPosition = 20
        scoreLabel.alpha = 1
        scoreLabel.fontSize = 25
        scoreLabel.fontName = "HelveticaNeue-Bold"

        scoreLabel.position = CGPoint(x: 0, y: 150)

        gameNode.addChild(scoreLabel)

        gameNode.addChild(pauseButton)

        physicsWorld.contactDelegate = self

        player.name = "player"
        player.position = CGPoint(x: 0, y: 0)
        player.size = CGSize(width: 45, height: 90)
        player.zPosition = 5
        player.alpha = 1
        var playerSize: CGSize = player.size
        playerSize.width -= 30
        playerSize.height -= 30
        //player.texture = SKTexture(imageNamed: "Main Character")
        player.physicsBody = SKPhysicsBody(rectangleOf: playerSize)
        player.physicsBody?.categoryBitMask = CollisionType.player.rawValue
        player.physicsBody?.collisionBitMask = CollisionType.enemy.rawValue
        player.physicsBody?.contactTestBitMask = CollisionType.enemy.rawValue
        player.physicsBody?.allowsRotation = false
        player.physicsBody?.restitution = 0
        player.physicsBody?.friction = 0
        physicsWorld.gravity = CGVector(dx: 0, dy: 0)
        player.physicsBody?.isDynamic = true

        gameNode.addChild(player)

        //physicsBody = SKPhysicsBody(edgeLoopFrom: frame)

        motionManager = CMMotionManager()
        motionManager?.startGyroUpdates()

    }
```

```swift
func didBegin(_ contact: SKPhysicsContact) {

    guard let nodeA = contact.bodyA.node else { return }
    guard let nodeB = contact.bodyB.node else { return }

    if nodeA.name == "zombie" {
        collisionBetween(zombie: nodeA, object: nodeB)
    } else if nodeB.name == "zombie" {
        collisionBetween(zombie: nodeB, object: nodeA)
    }
}

func collisionBetween(zombie: SKNode, object: SKNode){
    if(object.name == "bullet"){
        print("bullet hit")
        object.removeFromParent()
        zombie.removeFromParent()
        model.score += 10
        scoreLabel.text = "Score: \(model.score)"
    }
    else if(object.name == "player"){
        print("Game Over")
        gameOver()
    }
}
```

```swift
func gameOver() {
    self.isPaused = true

    // Create a node for the game over screen
    let gameOverNode = SKNode()
    gameOverNode.zPosition = 100

    // Add a semi-transparent black background to the game over screen
    let background = SKSpriteNode(color: .black, size: self.size)
    background.alpha = 0.8
    background.position = CGPoint(x: frame.midX, y: frame.midY)
    gameOverNode.addChild(background)

    // Add the "GAME OVER" label
    let gameOverLabel = SKLabelNode(text: "GAME OVER")
    gameOverLabel.fontName = "HelveticaNeue-Bold"
    gameOverLabel.fontSize = 40
    gameOverLabel.fontColor = .white
    gameOverLabel.position = CGPoint(x: frame.midX, y: frame.midY + 50)
    gameOverNode.addChild(gameOverLabel)

    // Add the final score label
    let scoreText = "You scored: \(model.score)!"
    let scoreLabel = SKLabelNode(text: scoreText)
    scoreLabel.fontName = "HelveticaNeue-Bold"
    scoreLabel.fontSize = 30
    scoreLabel.fontColor = .white
    scoreLabel.position = CGPoint(x: frame.midX, y: frame.midY)
    gameOverNode.addChild(scoreLabel)

    // Add the "Restart" button
    let restartLabel = SKLabelNode(text: "Restart")
    restartLabel.name = "restart"
    restartLabel.fontName = "HelveticaNeue-Bold"
    restartLabel.fontSize = 30
    restartLabel.fontColor = .white
    restartLabel.position = CGPoint(x: frame.midX, y: frame.midY - 50)
    gameOverNode.addChild(restartLabel)

    // Add the "Main Menu" button
    let mainMenuLabel = SKLabelNode(text: "Main Menu")
    mainMenuLabel.name = "mainMenu"
    mainMenuLabel.fontName = "HelveticaNeue-Bold"
    mainMenuLabel.fontSize = 30
    mainMenuLabel.fontColor = .white
    mainMenuLabel.position = CGPoint(x: frame.midX, y: frame.midY - 100)
    gameOverNode.addChild(mainMenuLabel)

    let playerName = "Player"
    DataModelManager.shared.saveScore(playerName: playerName, score: model.score)
    // Add the game over node to the scene
    addChild(gameOverNode)
}
```

```swift
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        guard let touch = touches.first, let scene = self.scene else { return }
        let location = touch.location(in: scene)
        print(location)

        // Check if the game is not paused and if the touched point is not the pause button
        if !gameNode.isPaused && self.atPoint(location).name != "pauseButton" {
            let bullet = Bullet(imageNamed: "bullet")
            bullet.name = "bullet"
            bullet.position = player.position
            bullet.zPosition = 6
            var bulletSize = bullet.size
            bulletSize.width -= 5
            bulletSize.height -= 5

            bullet.physicsBody = SKPhysicsBody(rectangleOf: bulletSize)
            bullet.physicsBody?.categoryBitMask = CollisionType.playerBullet.rawValue
            bullet.physicsBody?.collisionBitMask = CollisionType.enemy.rawValue
            bullet.physicsBody?.contactTestBitMask = CollisionType.enemy.rawValue
            bullet.physicsBody?.allowsRotation = false
            bullet.physicsBody?.isDynamic = true
            bullet.physicsBody?.mass = 0.001
            bullet.zRotation = calcBulletAngle(touchLocation: location, playerLocation: player.position)
            gameNode.addChild(bullet)
            model.bulletsInPlay.append(bullet)
            print("Bullet added")
            bullet.physicsBody?.applyImpulse(calculateBulletVector(touchLocation: location, playerLocation: player.position, power: 0.75))
        }

        // Handle other touch interactions on specific nodes
        for t in touches {
            let node = self.atPoint(t.location(in: self))

            switch node.name {
            case "pauseButton":
                if !gameNode.isPaused {
                    print("Pause")
                    pauseGame()
                    addChild(pauseNode)
                }
            case "resume":
                pauseNode.removeFromParent()
                gameNode.isPaused = false
                player.physicsBody?.isDynamic = true
            case "home label":
                if let homeScene = SKScene(fileNamed: "OpeningScene") {
                    homeScene.scaleMode = .resizeFill
                    view?.presentScene(homeScene)
                    model.score = 0
                }
            case "restart":
                restartGame()
            case "mainMenu":
                goToMainMenu()
            default:
                break
            }
        }
    }
```

```swift
func pauseGame(){
    gameNode.isPaused = true
    player.physicsBody?.isDynamic = false

    let backgrnd = SKSpriteNode(color: UIColor.gray, size: frame.size)
    backgrnd.name = "background"
    backgrnd.position = CGPoint(x: 0, y: 0)
    backgrnd.zPosition = 50

    // Remove existing pauseScoreLabel if it exists
    pauseNode.childNode(withName: "pauseScoreLabel")?.removeFromParent()

    let pauseScoreLabel = SKLabelNode(text: "Score: \(model.score)")
    pauseScoreLabel.name = "pauseScoreLabel" // Assign a name for easy identification
    pauseScoreLabel.fontName = "HelveticaNeue-Bold"
    pauseScoreLabel.fontSize = 25
    pauseScoreLabel.zPosition = 100
    pauseScoreLabel.position = CGPoint(x: 0, y: -100)

    let resumeLabel = SKLabelNode(text: "Resume")
    resumeLabel.name = "resume"
    resumeLabel.zPosition = 100
    resumeLabel.position = CGPoint(x: 0, y: 100)

    let homeLabel = SKLabelNode(text: "Main Menu")
    homeLabel.name = "home label"
    homeLabel.zPosition = 100
    homeLabel.position = CGPoint(x: 0, y: 0)

    // Ensure all children are added only once
    if pauseNode.children.isEmpty {
        pauseNode.addChild(backgrnd)
        pauseNode.addChild(resumeLabel)
        pauseNode.addChild(homeLabel)
    }
    pauseNode.addChild(pauseScoreLabel) // Add pauseScoreLabel after checking other children
}
```

```swift
func calcBulletAngle(touchLocation: CGPoint, playerLocation: CGPoint) -> CGFloat{
    let x: CGFloat = touchLocation.x - playerLocation.x
    let y: CGFloat = touchLocation.y - playerLocation.y
    var ang: CGFloat = atan(y/x)
    if(x<0){
        ang += .pi
    }

    return ang

}

func calculateBulletVector(touchLocation: CGPoint, playerLocation: CGPoint, power: CGFloat) -> CGVector {
    let x: CGFloat = touchLocation.x - playerLocation.x
    let y: CGFloat = touchLocation.y - playerLocation.y
    let hyp:CGFloat = sqrt(x*x + y*y)
    return CGVector(dx: (x/hyp)*power, dy: (y/hyp)*power)
}


func addZombie(at position: CGPoint) {
    let zombie = Zombie(imageNamed: "Zombie Character")
    zombie.name = "zombie"
    zombie.position = position
    zombie.size = player.size
    var rect = player.size
    rect.width -= 30
    rect.height -= 30
    zombie.physicsBody = SKPhysicsBody(rectangleOf: rect)
    zombie.physicsBody?.isDynamic = false

    zombie.physicsBody?.categoryBitMask = CollisionType.enemy.rawValue
    zombie.physicsBody?.collisionBitMask = CollisionType.player.rawValue | CollisionType.playerBullet.rawValue
    zombie.physicsBody?.contactTestBitMask = CollisionType.player.rawValue | CollisionType.playerBullet.rawValue
    gameNode.addChild(zombie)
    model.zombiesInPlay.append(zombie)
}
```

```
342     override func update(_ currentTime: TimeInterval) {
343         if let accelerometerData = motionManager?.gyroData{
344             if(!gameNode.isPaused){
345                 physicsWorld.gravity = CGVector(dx: accelerometerData.rotationRate.x*5, dy: accelerometerData.rotationRate.y*5)
346             }
347             else{
348                 physicsWorld.gravity = CGVector(dx: 0, dy: 0)
349             }
350             //print(physicsWorld.gravity)
351         }
352         keepPlayerInBounds()
353         for child in children{
354             if child is Bullet{
355                 if(!frame.intersects(child.frame)){
356                     child.removeFromParent()
357                     if let index = model.bulletsInPlay.firstIndex(of: child as! Bullet){
358                         model.bulletsInPlay.remove(at: index)
359                         print("removed")
360                         print(model.bulletsInPlay.count)
361                     }
362                 }
363             }
364         }
365         spawnZombiesIfNeeded(currentTime)
366 
367                 // Update zombie positions
368         moveZombiesTowardsPlayer()
369     }
370 
371     func spawnZombiesIfNeeded(_ currentTime: TimeInterval) {
372         if(!gameNode.isPaused){
373             // Adjust spawn rate as needed
374             if currentTime - lastSpawnTimeInterval > 1 {
375                 lastSpawnTimeInterval = currentTime
376 
377                 // Spawn zombies from the front or back randomly
378                 let spawnFromFront = Bool.random()
379 
380                 // Adjust spawn position based on the direction of movement
381                 let spawnY: CGFloat = spawnFromFront ? frame.maxY : frame.minY
382                 let spawnX = CGFloat.random(in: frame.minX...frame.maxX)
383 
384                 addZombie(at: CGPoint(x: spawnX, y: spawnY))
385             }
386         }
387     }
```

Thank you!

Questions?