//

// GameScene.swift

// Spike Hopper

//

// Created by Sevan Productions on 8/22/15.

// Copyright (c) 2015 Evan Murray. All rights reserved.

//

import SpriteKit

import UIKit

import AVFoundation

import GameController

import GameKit

var screenBounds: CGRect = UIScreen.main.bounds

// Math Helpers

extension Float {

static func clamp(min: CGFloat, max: CGFloat, value: CGFloat) -> CGFloat {

if (value > max) {

return max

} else if (value < min) {

return min

} else {

return value

}

}

static func range(min: CGFloat, max: CGFloat) -> CGFloat {

return CGFloat(Float(arc4random()) / 0xFFFFFFFF) \* (max - min) + min

}

}

// Touch phases

enum UITouchPhase : Int {

case Began

case Moved

case Stationary

case Ended

case Cancelled

}

//Variable for the player's score

var score: Int = 0

var HighScore: Int = 0

//Variable for saving the high score number

var HighScoreNumber = UserDefaults.standard.integer(forKey: "HighScoreSaved")

var phase : UITouchPhase = .Began

class GameScene: SKScene, SKPhysicsContactDelegate, GKGameCenterControllerDelegate {

//Variable for game music

//shows leaderboard screen

func showLeader() {

let vc = self.view?.window?.rootViewController

let gc = GKGameCenterViewController()

gc.gameCenterDelegate = self

vc?.present(gc, animated: true, completion: nil)

}

func saveHighscore(score:Int) {

//check if user is signed in

if GKLocalPlayer.localPlayer().isAuthenticated {

print("authenticated")

let scoreReporter = GKScore(leaderboardIdentifier: "grp.spikehopperleaderboard") //leaderboard id here

print("ScoreReporter: \(scoreReporter)")

scoreReporter.value = Int64(score) //score variable here (same as above)

let scoreArray: [GKScore] = [scoreReporter]

GKScore.report(scoreArray, withCompletionHandler: {(error : Error?) -> Void in

if error != nil {

print("error")

}

else{

print("reported correctly")

}

})

}

}

//hides leaderboard screen

func gameCenterViewControllerDidFinish(\_ gameCenterViewController: GKGameCenterViewController)

{

gameCenterViewController.dismiss(animated: true, completion: nil)

}

//Collider types

enum ColliderType:UInt32 {

case phoenix = 1

case spike = 2

}

// Variable for the game music

var music = NSURL(fileURLWithPath: Bundle.main.path(forResource: "music", ofType: "m4a")!)

var audioPlayermusic = AVAudioPlayer()

var audioPlayerding = AVAudioPlayer()

//Variable for the ding sound when the score increases

var ding = NSURL(fileURLWithPath: Bundle.main.path(forResource: "ding", ofType: "wav")!)

//variable for how fast the spikes go across the screen

var spike\_speed = 13.7

//variable for how fast the spikes go across the screen

var spike\_speed\_substitute = 13.7

//variable for how fast the floor goes across the screen

var floor\_speed = 830.0

//variable for how fast the floor goes across the screen

var floor\_speed\_substitute = 830.0

// Variable for Phoenix the main character

var phoenix = SKSpriteNode(imageNamed: "phoenix")

let jump\_speed:Float = 200

// Sets the variable for a timer in the game that can be used anywhere in the code

var startTime : NSDate!

// Variable for the floor

var floor = SKSpriteNode(imageNamed: "bar")

// Time Values

var delta = TimeInterval(0)

var last\_update\_time = TimeInterval(0)

//Determines whether the character is touching the ground or not

var onGround = true

//Variable for the velocity of the character (later used for jumping)

var velocityY = CGFloat(0)

//Variable for the gravity of the world so that the character falls

let gravity = CGFloat(0.6)

//Variable for the shorter spikes

var spike1 = SKSpriteNode(imageNamed: "Spikes")

var spike2 = SKSpriteNode(imageNamed: "Spikes")

//Variable for a taller spike

var SpikeTall = SKSpriteNode(imageNamed: "Spikebox")

//Variable for the maximum x position of the spikes

var spikeMaxX = CGFloat(0)

//Variable for the starting position of the spikes

var origSpikePositionX = CGFloat(0)

//Variable for the score text

let scoreText = SKLabelNode(fontNamed: "Visitor TT2 BRK")

//Variable for the pause button

let pauseButton = SKSpriteNode(imageNamed:"pausebutton")

//Variable for the background in each menu

var Background = SKSpriteNode(imageNamed:"Background")

var Backgroundbig = SKSpriteNode(imageNamed:"Backgroundbig")

//Variable for the playbutton

var playButton = SKSpriteNode(imageNamed: "playgame")

//Variable for the menubutton on the pause menu

var menuButton = SKSpriteNode(imageNamed: "menubutton")

//Variable for the "paused" text

var pauseText = SKSpriteNode(imageNamed: "paused")

// SKScene Initialization

override func didMove(to: SKView) {

self.physicsWorld.contactDelegate = self

//Sets up the background music and plays it

do {

try audioPlayermusic = AVAudioPlayer(contentsOf: music as URL!, fileTypeHint:nil)

} catch {

//Handle the error

}

audioPlayermusic.numberOfLoops = -1

audioPlayermusic.prepareToPlay()

audioPlayermusic.play()

//Sets up the ding sound for when the score increases

do {

try audioPlayerding = AVAudioPlayer(contentsOf: ding as URL!, fileTypeHint:nil)

} catch {

//Handle the error

}

audioPlayerding.prepareToPlay()

//initiates the functions for setting up the character and setting up the floor

initFloor()

initPhoenix()

//sets the position and size of the shorter spike and adds it off the screen

//Also gives it a physicsbody and collider type

self.spike1.position = CGPoint(x:1000 + self.spike1.size.width, y: 80)

self.spike1.zPosition = 21

self.spike1.xScale = 0.4

self.spike1.yScale = 0.4

self.spike1.physicsBody = SKPhysicsBody(rectangleOf: CGSize(width:80, height:30))

self.spike1.physicsBody?.isDynamic = false

self.spike1.physicsBody?.categoryBitMask = ColliderType.spike.rawValue

self.spike1.physicsBody?.contactTestBitMask = ColliderType.spike.rawValue

self.spike1.physicsBody?.collisionBitMask = ColliderType.spike.rawValue

self.addChild(spike1)

//Gives the spike a name so that it is recognized

self.spike1.name = "spike1"

//Randomizes when the shorter spike will move across the screen

spikeStatuses["spike1"] = SpikeStatus(isRunning: false, timeGapForNextRun: random(), currentInterval: UInt32(0))

//sets the position and size of the shorter spike and adds it off the screen

//Also gives it a physicsbody and collider type

self.spike2.position = CGPoint(x: 1000 + self.spike2.size.width, y: 80)

self.spike2.zPosition = 21

self.spike2.xScale = 0.4

self.spike2.yScale = 0.4

self.spike2.physicsBody = SKPhysicsBody(rectangleOf: CGSize(width:80, height:30))

self.spike2.physicsBody?.isDynamic = false

self.spike2.physicsBody?.categoryBitMask = ColliderType.spike.rawValue

self.spike2.physicsBody?.contactTestBitMask = ColliderType.spike.rawValue

self.spike2.physicsBody?.collisionBitMask = ColliderType.spike.rawValue

self.addChild(spike2)

//Gives the spike a name so that it is recognized

self.spike2.name = "spike2"

//Randomizes when the shorter spike will move across the screen

spikeStatuses["spike2"] = SpikeStatus(isRunning: false, timeGapForNextRun: random(), currentInterval: UInt32(0))

//sets the position and size of the taller spike and adds it off the screen

//Also gives it a physicsbody and collider type

self.SpikeTall.position = CGPoint(x: 1000 + self.SpikeTall.size.width, y: 80)

self.SpikeTall.zPosition = 21

self.SpikeTall.xScale = 0.4

self.SpikeTall.yScale = 0.4

self.SpikeTall.physicsBody = SKPhysicsBody(rectangleOf: CGSize(width:80, height:70))

self.SpikeTall.physicsBody?.isDynamic = false

self.SpikeTall.physicsBody?.categoryBitMask = ColliderType.spike.rawValue

self.SpikeTall.physicsBody?.contactTestBitMask = ColliderType.spike.rawValue

self.SpikeTall.physicsBody?.collisionBitMask = ColliderType.spike.rawValue

self.addChild(SpikeTall)

//Gives a name to the spike so that it is recognized

self.SpikeTall.name = "SpikeTall"

//Randomizes when the taller spike will move across the screen

spikeStatuses["SpikeTall"] = SpikeStatus(isRunning: false, timeGapForNextRun: random(), currentInterval: UInt32(0))

//Sets up the score text

self.scoreText.text = ""

self.scoreText.zPosition = 21

self.scoreText.fontSize = 100

self.scoreText.position = CGPoint(x: self.frame.midX, y: self.frame.midY)

self.addChild(scoreText)

//Sets the position of the maximum spike position on the x axis

self.spikeMaxX = 0 - self.spike1.size.width / 2

//Sets the position of the original spike position

self.origSpikePositionX = self.spike1.position.x

//Sets up the pause button

self.pauseButton.position = CGPoint(x: self.frame.minX + 35, y: self.frame.maxY - 65)

self.pauseButton.zPosition = 22

self.pauseButton.xScale = 1

self.pauseButton.yScale = 1

self.addChild(pauseButton)

//Sets up the background

if screenBounds.size.width == 1366 && screenBounds.size.height == 1024 {

Background.removeFromParent()

self.Backgroundbig.anchorPoint = CGPoint(x: 0, y: 0)

self.Backgroundbig.position = CGPoint(x: 0, y: 0)

self.Backgroundbig.zPosition = 0

self.addChild(Backgroundbig)

}else{

self.Background.anchorPoint = CGPoint(x: 0, y: 0)

self.Background.position = CGPoint(x: self.frame.minX, y: self.frame.minY)

self.Background.zPosition = 0

self.addChild(Background)

}

}

//The function that is run when the spike is added to the scene so that it randomly decides when to move across the screen

func random() -> UInt32 {

let range = UInt32(1)..<UInt32(500)

return range.startIndex + arc4random\_uniform(range.endIndex - range.startIndex + 1)

}

//Dictionary for the SpikeStatus

var spikeStatuses:Dictionary<String,SpikeStatus> = [:]

//Init phoenix

func initPhoenix() {

phoenix.position = CGPoint(x: 100, y:85)

phoenix.xScale = 0.5

phoenix.yScale = 0.5

phoenix.zPosition = 20

self.addChild(phoenix)

//Sets up the physics body of phoenix and the collider

self.phoenix.physicsBody = SKPhysicsBody(rectangleOf: CGSize(width:30, height:60))

self.phoenix.physicsBody?.affectedByGravity = false

self.phoenix.physicsBody?.categoryBitMask = ColliderType.phoenix.rawValue

self.phoenix.physicsBody?.contactTestBitMask = ColliderType.spike.rawValue

self.phoenix.physicsBody?.collisionBitMask = ColliderType.spike.rawValue

self.phoenix.physicsBody?.allowsRotation = false

//Animates the character so that it looks like he runs

let texture1: SKTexture = SKTexture(imageNamed: "phoenixr")

let texture2: SKTexture = SKTexture(imageNamed: "phoenixl")

let textures = [texture1, texture2]

phoenix.run(SKAction.repeatForever(SKAction.animate(with:textures, timePerFrame: 0.1)))

}

//initFloor

func initFloor() {

floor = SKSpriteNode()

addChild(floor)

//Creates an infinate scroll effect so the floor looks like it's moving

for i in 0...2 {

let tile = SKSpriteNode(imageNamed: "bar")

tile.anchorPoint = CGPoint.zero

tile.yScale = 0.85

tile.position = CGPoint(x: CGFloat(i) \* 640.0, y: 0.0)

tile.name = "bar"

tile.zPosition = 19

floor.addChild(tile)

}

}

//Moves the floor across the screen

func moveFloor() {

let posX = -floor\_speed \* delta

floor.position = CGPoint(x: floor.position.x + CGFloat(posX), y: 0.0)

floor.enumerateChildNodes(withName: "bar") { (node, stop) in

let floor\_screen\_position = self.floor.convert(node.position, to: self)

if floor\_screen\_position.x <= -node.frame.size.width {

node.position = CGPoint(x: node.position.x + (node.frame.size.width \* 2), y: node.position.y)

}

}

}

//Called when the user touches anywhere in the screen

override func touchesBegan(\_ touches: Set<UITouch>, with: UIEvent?) {

for touch: AnyObject in touches {

let location = touch.location(in: self)

//Called when the user touches the pause button

if self.atPoint(location) == self.pauseButton {

//Stops the background music

audioPlayermusic.stop()

//Removes the pause button from the game

self.pauseButton.removeFromParent()

//Play button setup

self.playButton.anchorPoint.y = 0.5

self.playButton.position = CGPoint(x: self.frame.minX + 100, y: self.frame.minY + 100)

self.playButton.zPosition = 30

self.addChild(playButton)

//Menu button setup

self.menuButton.anchorPoint.y = 0.5

self.menuButton.position = CGPoint(x: self.frame.maxX - 100, y:self.frame.minY + 100)

self.menuButton.zPosition = 30

self.addChild(menuButton)

//Pause text setup

self.pauseText.xScale = 0.8

self.pauseText.yScale = 0.8

self.pauseText.position.x = self.frame.midX

self.pauseText.position.y = self.frame.maxY - 100

self.pauseText.zPosition = 30

self.addChild(pauseText)

//Removes character from screen

self.phoenix.removeFromParent()

//Stops moving everything so that the game is paused

spike\_speed = 0

floor\_speed = 0

}

//Called when the user touches the play button on the pause menu

if self.atPoint(location) == self.playButton {

//plays the background music

audioPlayermusic.play()

//Adds the pause button back to the game

self.addChild(pauseButton)

//Removes all the items on the pause screen

self.playButton.removeFromParent()

self.menuButton.removeFromParent()

self.pauseText.removeFromParent()

//Puts the character back on the screen

self.addChild(phoenix)

//Sets the floor and spike speed back to where they were

spike\_speed = spike\_speed\_substitute

floor\_speed = floor\_speed\_substitute

}

//Called when the user touches the menu button on the pause menu

if self.atPoint(location) == self.menuButton {

//Sets up transition

let transition = SKTransition.fade(withDuration: 2)

let scene3 = MainMenu(size: self.size)

let skView3 = self.view as SKView!

skView3!.ignoresSiblingOrder = true

scene3.scaleMode = .resizeFill

scene3.size = skView3!.bounds.size

skView3!.presentScene(scene3, transition: transition)

}

}

//Makes the character jump

if onGround{

self.velocityY = -16.0

self.onGround = false

}

}

//Called when the user runs into a spike

func didBegin(\_ contact:SKPhysicsContact) {

died()

}

//What happens when you die

func died() {

floor\_speed = 0

floor\_speed\_substitute = 0

spike\_speed = 0

spike\_speed\_substitute = 0

phoenix.removeFromParent()

audioPlayerding.stop()

audioPlayermusic.stop()

//Sets up transition

let transition = SKTransition.fade(withDuration: 2)

//Sets the high score number if needed

if (score > HighScoreNumber) {

UserDefaults.standard.set(score, forKey: "HighScoreSaved")

UserDefaults.standard.synchronize()

HighScore = score

saveHighscore(score: score)

}

//loads the scene in the class "GameScene"

let scene2 = gameover(size: self.size)

let skView2 = self.view as SKView!

skView2!.ignoresSiblingOrder = true

scene2.scaleMode = .resizeFill

scene2.size = skView2!.bounds.size

skView2!.presentScene(scene2, transition: transition)

}

//Called when the user lets go of the screen

override func touchesEnded(\_ touches: Set<UITouch>, with: UIEvent?) {

//Makes the start to fall

if self.velocityY < -8.0 {

self.velocityY = -8.0

self.onGround = false

}

}

//Frames Per Second

override func update(\_ currentTime: CFTimeInterval) {

//Calls the function to move the floor

delta = (last\_update\_time == 0.0) ? 0.0 : currentTime - last\_update\_time

last\_update\_time = currentTime

self.moveFloor()

//Puts gravity into effect when the character jumps

self.velocityY += self.gravity

self.phoenix.position.y -= velocityY

//Tells the game what to do when the character is on the ground

if self.phoenix.position.y < 85 {

phoenix.position.y = 85

velocityY = 0.0

onGround = true

}

//Runs the function spikeRunner

spikeRunner()

}

//Puts the spike randomizer into effect so that the spikes move at random

func spikeRunner() {

for(spike, spikeStatus) in self.spikeStatuses {

let thisSpike = self.childNode(withName: spike)!

if spikeStatus.shouldRunBlock() {

spikeStatus.timeGapForNextRun = random()

spikeStatus.currentInterval = 0

spikeStatus.isRunning = true

}

if spikeStatus.isRunning {

if thisSpike.position.x > spikeMaxX {

thisSpike.position.x -= CGFloat(spike\_speed)

}else {

//Happens when the spike reaches the max position on the screen

thisSpike.position.x = self.origSpikePositionX

spikeStatus.isRunning = false

score += 1

audioPlayerding.play()

//Increases the ground and spike speed each time the player's score increases by 5

if ((score % 5) == 0) {

floor\_speed += 60.58394160583942

floor\_speed\_substitute += 60.58394160583942

spike\_speed += 1

spike\_speed\_substitute += 1

//Tells the ground and spike speed to stop increasing when the player reaches a score of 80

}else {

if (score >= 50){

floor\_speed = 1435.8394160583942

floor\_speed\_substitute = 1435.8394160583942

spike\_speed = 23.7

spike\_speed\_substitute = 23.7

}

}

self.scoreText.text = String(stringInterpolationSegment: score)

}

}else {

spikeStatus.currentInterval += 1

}

}

}

}