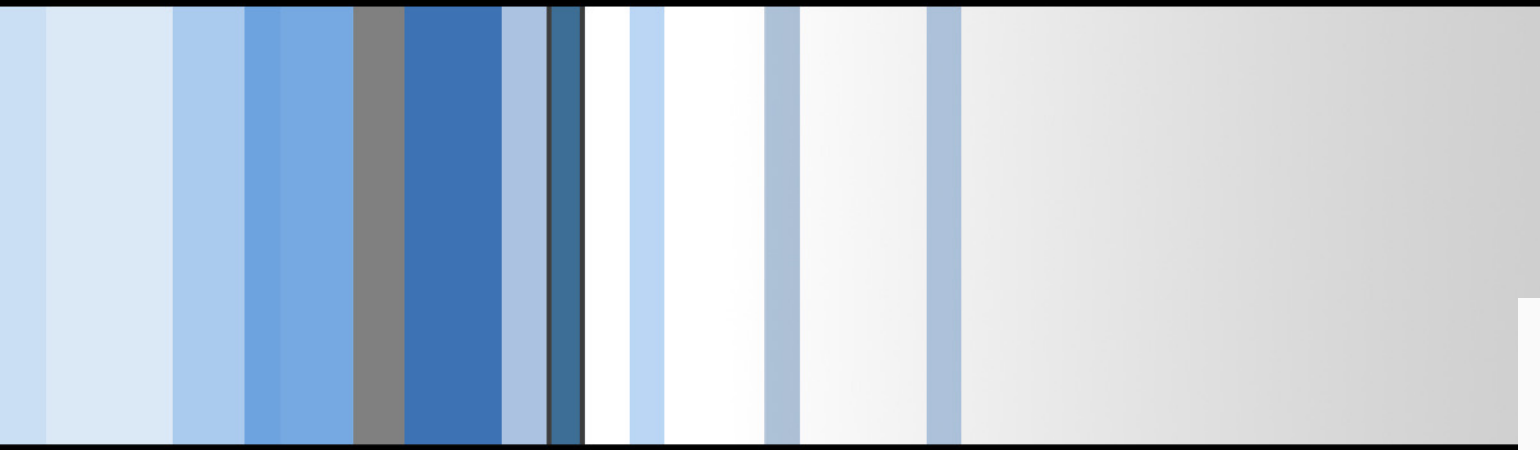
Class 20

CSC 496: iOS App Development
SpriteKit (3)

Si Chen (schen@wcupa.edu)



SpriteKit

Spawn Fruits at a random position

```
func spawnFruit(){
    let collectible = Collectible(collectibleType: CollectibleType.fruit)

    // set random position
    let margin = collectible.size.width * 2
    let dropRange = SKRange(lowerLimit: frame.minX + margin, upperLimit: frame.maxX - margin)
    let randomX = CGFloat.random(in: dropRange.lowerLimit...dropRange.upperLimit)
    collectible.position = CGPoint(x:randomX, y:player.position.y * 2.5)
    addChild(collectible)

    collectible.drop(dropSpeed: TimeInterval(1.0), floorLevel: player.frame.minY)
}
```

Download Picaku_v2.zip

Working with Physics and Collision Detection

- Add physics body to player node (Player.swift -> init())

```
init(){
    // set default texture
    let texture = SKTexture(imageNamed: "frame_0")

    // call to super.init
    super.init(texture: texture, color: .clear, size: texture.size())

    self.name = "player"
    self.setScale(1.0)
    self.anchorPoint = CGPoint(x: 0.5, y:0.0)
    self.zPosition = Layer.player.rawValue

    // add physics body
    self.physicsBody = SKPhysicsBody(rectangleOf: self.size, center: CGPoint(x: 0.0, y:
        self.size.height/2))
    self.physicsBody?.affectedByGravity = false

}
```

Working with Physics and Collision Detection

- Add physics body to collectible node, aka fruit (Collectible.swift -> init())

```
class Collectible:SKSpriteNode{
    private var collectibleType: CollectibleType = .none

    init(collectibleType: CollectibleType){
        var texture: SKTexture!
        self.collectibleType = collectibleType

        // set the texture based on the Type
        switch self.collectibleType{
        case .fruit:
            texture = SKTexture(imageNamed: "fruit")
        case .none:
            break
        }
        super.init(texture: texture, color: SKColor.clear, size: texture.size())

        // set up the collectible

        self.name = "co_\(collectibleType)"
        self.anchorPoint = CGPoint(x: 0.5, y: 1.0)
        self.setScale(0.3)
        self.zPosition = Layer.collectible.rawValue

        // add physics body
        self.physicsBody = SKPhysicsBody(rectangleOf: self.size, center: CGPoint(x: 0.0, y:
            -self.size.height/2))
        self.physicsBody?.affectedByGravity = false
    }
```

Working with Physics and Collision Detection

- Add physics body to foreground node (GameScene.swift -> didMove())

```
let foreground = SKSpriteNode(imageNamed: "foreground")
foreground.anchorPoint = CGPoint(x: 0, y: 0)
foreground.position = CGPoint(x:15, y:158)
foreground.zPosition = Layer.foreground.rawValue
// add physics body
foreground.physicsBody = SKPhysicsBody(edgeLoopFrom: foreground.frame)
foreground.physicsBody?.affectedByGravity = false
addChild(foreground)
```

Configure Physics Categories

- Open the SpriteKitHelper.swift, add the following code:

```
import Foundation
import SpriteKit

enum Layer:CGFloat{
    case background
    case foreground
    case player
    case collectible
}

// SpriteKit Physics Categories
enum PhysicsCategory{
    static let none: UInt32 = 0
    static let player: UInt32 = 0b1 // 1
    static let collectible: UInt32 = 0b10 // 2
    static let foreground: UInt32 = 0b100 // 4
}
```

Set up physics categories for contacts

- Set up physics categories for contacts to foreground node (GameScene.swift -> didMove())

```
// add physics body
foreground.physicsBody = SKPhysicsBody(edgeLoopFrom: foreground.frame)
foreground.physicsBody?.affectedByGravity = false

foreground.physicsBody?.categoryBitMask = PhysicsCategory.foreground
foreground.physicsBody?.contactTestBitMask = PhysicsCategory.collectible
foreground.physicsBody?.collisionBitMask = PhysicsCategory.none

addChild(foreground)
```

Set up physics categories for contacts

```
init(){
    // set default texture
    let texture = SKTexture(imageNamed: "frame_0")

    // call to super.init
    super.init(texture: texture, color: .clear, size: texture.size())

    self.name = "player"
    self.setScale(1.0)
    self.anchorPoint = CGPoint(x: 0.5, y:0.0)
    self.zPosition = Layer.player.rawValue

    // add physics body
    self.physicsBody = SKPhysicsBody(rectangleOf: self.size, center: CGPoint(x: 0.0, y:
        self.size.height/2))
    self.physicsBody?.affectedByGravity = false
    // set up physics categories for contacts
    self.physicsBody?.categoryBitMask = PhysicsCategory.player
    self.physicsBody?.contactTestBitMask = PhysicsCategory.collectible
    self.physicsBody?.collisionBitMask = PhysicsCategory.none

}
```

- Add physics categories to player node (Player.swift -> init())

Set up physics categories for contacts

- Add physics categories for contacts to collectible node, aka fruit (Collectible.swift -> init())

```
init(collectibleType: CollectibleType){
    var texture: SKTexture!
    self.collectibleType = collectibleType

    // set the texture based on the Type
    switch self.collectibleType{
    case .fruit:
        texture = SKTexture(imageNamed: "fruit")
    case .none:
        break
    }
    super.init(texture: texture, color: SKColor.clear, size: texture.size())

    // set up the collectible

    self.name = "co_\(collectibleType)"
    self.anchorPoint = CGPoint(x: 0.5, y: 1.0)
    self.setScale(0.3)
    self.zPosition = Layer.collectible.rawValue

    // add physics body
    self.physicsBody = SKPhysicsBody(rectangleOf: self.size, center: CGPoint(x: 0.0, y:
        -self.size.height/2))
    self.physicsBody?.affectedByGravity = false
    // set up physics categories for contacts
    self.physicsBody?.categoryBitMask = PhysicsCategory.collectible
    self.physicsBody?.contactTestBitMask = PhysicsCategory.player | PhysicsCategory.foreground
    self.physicsBody?.collisionBitMask = PhysicsCategory.none
}
```

Configure the Physics Contact Delegate

- Open GameScene.swift, and at the bottom of the file, add a new extension to handle the collision detection.

```
extension GameScene:SKPhysicsContactDelegate{
}
```

- This extension declares that the GameScene class can act as a delegate for SKPhysicsContactDelegate. You need to make it official by adding the following line in didMove() method

```
class GameScene: SKScene {
    let player = Player()

    override func didMove(to view: SKView) {
        // set up the physics world contact delegate
        physicsWorld.contactDelegate = self
```

Detect Contact Between Physics Bodies

Open GameScene.swift, make change

```
extension GameScene:SKPhysicsContactDelegate{
    func didBegin(_ contact: SKPhysicsContact) {
        let collision = contact.bodyA.categoryBitMask | contact.bodyB.categoryBitMask

        if collision == PhysicsCategory.player | PhysicsCategory.collectible{
            print("player hit collectible")
        }

        if collision == PhysicsCategory.foreground | PhysicsCategory.collectible{
            print("collectible hit foreground")
        }
    }
}
```

Handle Contact Between Physic Bodies

Add these two methods to Collectible.swift

```
func collected(){
    let removeFromParent = SKAction.removeFromParent()
    self.run(removeFromParent)
}

func missed(){
    let removeFromParent = SKAction.removeFromParent()
    self.run(removeFromParent)

}
```

Handle Contact Between Physic Bodies

Open GameScene.swift, make change

```
extension GameScene:SKPhysicsContactDelegate{
    func didBegin(_ contact: SKPhysicsContact) {
        let collision = contact.bodyA.categoryBitMask | contact.bodyB.categoryBitMask

        if collision == PhysicsCategory.player | PhysicsCategory.collectible{
            print("player hit collectible")
            let body = contact.bodyA.categoryBitMask == PhysicsCategory.collectible ? contact.bodyA.node
                : contact.bodyB.node

            if let sprite = body as? Collectible{
                sprite.collected()
            }
        }

        if collision == PhysicsCategory.foreground | PhysicsCategory.collectible{
            print("collectible hit foreground")

            let body = contact.bodyA.categoryBitMask == PhysicsCategory.collectible ? contact.bodyA.node
                : contact.bodyB.node

            // verify the object is a collectible

            if let sprite = body as? Collectible{
                sprite.missed()
            }
        }
    }
}
```

Adding Labels to the Game

- In GameScene.swift

```
class GameScene: SKScene {
    let player = Player()
    // labels
    var scoreLevel: SKLabelNode = SKLabelNode()
```

- In SpriteKitHelper.swift

```
enum Layer:CGFloat{
    case background
    case foreground
    case player
    case collectible
    case ui
}
```

Adding Labels to the Game

- In GameScene.swift, add the following method

```
func setupLabels(){
    scoreLevel.name = "score"
    scoreLevel.fontColor = .red
    scoreLevel.fontSize = 55.0
    scoreLevel.horizontalAlignmentMode = .right
    scoreLevel.verticalAlignmentMode = .center
    scoreLevel.zPosition = Layer.ui.rawValue
    scoreLevel.position = CGPoint(x:frame.maxX - 50, y: 700)
    scoreLevel.text = "Score: 0"
    addChild(scoreLevel)
}
```

Adding Labels to the Game

- In GameScene.swift, calling the new added method

```
override func didMove(to view: SKView) {
    physicsWorld.contactDelegate = self

    let background = SKSpriteNode(imageNamed: "background")
    background.anchorPoint = CGPoint(x: 0, y: 0)
    background.position = CGPoint(x: 10, y: 330)
    background.zPosition = Layer.background.rawValue
    addChild(background)

    let foreground = SKSpriteNode(imageNamed: "foreground")
    foreground.anchorPoint = CGPoint(x: 0, y: 0)
    foreground.position = CGPoint(x: 15, y: 158)
    foreground.zPosition = Layer.foreground.rawValue
    foreground.physicsBody = SKPhysicsBody(edgeLoopFrom: foreground.frame)
    foreground.physicsBody?.affectedByGravity = false

    foreground.physicsBody?.categoryBitMask = PhysicsCategory.foreground
    foreground.physicsBody?.contactTestBitMask = PhysicsCategory.collectible
    foreground.physicsBody?.collisionBitMask = PhysicsCategory.none
    addChild(foreground)


    player.position = CGPoint(x: size.width/2, y: foreground.frame.maxY)
    player.setupConstraints(floor: foreground.frame.maxY)
    addChild(player)
    spawnMultipleFruits()
    setupLabels()
}
```

Use Variable to Monitor Game States

- In GameScene.swift

```
// labels
var scoreLevel: SKLabelNode = SKLabelNode()
var score : Int = 0{
    didSet{
        scoreLevel.text = "Score: \(score)"
    }
}
```

-

```
extension GameScene:SKPhysicsContactDelegate{
    func didBegin(_ contact: SKPhysicsContact) {
        let collision = contact.bodyA.categoryBitMask | contact.bodyB.categoryBitMask

        if collision == PhysicsCategory.player | PhysicsCategory.collectible{
            print("player hit collectible")
            score += 10
            let body = contact.bodyA.categoryBitMask == PhysicsCategory.collectible ? contact.bodyA.node
                : contact.bodyB.node

            if let sprite = body as? Collectible{
                sprite.collected()
            }
        }
```

GameOver()

```
func gameOver(){
    // remove repeatable action on main scene
    removeAction(forKey: "fruit")

    // Loop through child nodes and stop actions on collectibles

    enumerateChildNodes(withName: "//co_*") {
        (node, stop) in
        node.removeAction(forKey: "drop")
        node.physicsBody = nil
    }
}
```

Put it in GameScene.swift

GameOver() – add a restart button

```
func gameOver(){
    removeAction(forKey: "fruit")

    enumerateChildNodes(withName: "//co_*"){
        (node, stop) in
        node.removeAction(forKey: "drop")
        node.physicsBody = nil
    }

    let restartButton = SKSpriteNode(color: .blue, size: CGSize(width: 120, height: 60))
    restartButton.position = CGPoint(x: frame.midX, y: frame.midY)
    restartButton.name = "restartButton"
    restartButton.zPosition = Layer.ui.rawValue
    addChild(restartButton)

    let buttonText = SKLabelNode(text: "Restart")
    buttonText.fontColor = .white
    buttonText.fontSize = 30
    buttonText.verticalAlignmentMode = .center
    restartButton.addChild(buttonText)
}
```


Put it in GameScene.swift

Update didBegin() in extension GameScene:SKPhysicsContactDelegate

```
class GameScene: SKScene {
    let player = Player()

    // labels
    var scoreLevel: SKLabelNode = SKLabelNode()
    var score: Int = 0{
        didSet{
            scoreLevel.text = "Score: \(score)"
        }
    }
    var missedCount = 0
    var maxAllowedMisses = 5
```

add variable missedCount and maxAllowedMisses

GameScene.Swift

```
  if collision == PhysicsCategory.foreground | PhysicsCategory.collectible{
      print("collectible hit foreground")

      let body = contact.bodyA.categoryBitMask == PhysicsCategory.collectible ? contact.bodyA.node : contact.bodyB.node

      if let sprite = body as? Collectible{
          sprite.missed()
          missedCount += 1
          if missedCount >= maxAllowedMisses {
              gameOver()
          }
      }
  }
```

GameScene.Swift

add restartGame() in GameScene.Swift

```
func restartGame() {
    score = 0
    missedCount = 0
    enumerateChildNodes(withName: "//co_*") { (node, _) in
        node.removeFromParent()
    }
    enumerateChildNodes(withName: "restartButton") { (node, _) in
        node.removeFromParent()
    }

    spawnMultipleFruits()
}
```

GameScene.Swift

Update touchesBegin() function to support Button Press

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

        if nodes.contains(where: { $0.name == "restartButton" }) {
            // restart game
            restartGame()
        } else {
            self.touchDown(atPoint: location)
        }
    }
}
```

Lab4 (10%): Build a mini coin collection game

- Richman 4 was one of my favorite childhood games.
- Similar to the Monopoly board game, players aim to amass the greatest wealth and/or bankrupt their opponents as they move around a game board filled with property plots and special tiles.
- The objective is primarily achieved by purchasing properties and collecting rent from opponents, or by investing in a simulated stock market.
- The gameplay typically includes a variety of minigames, special items, and random events that can alter the standings.

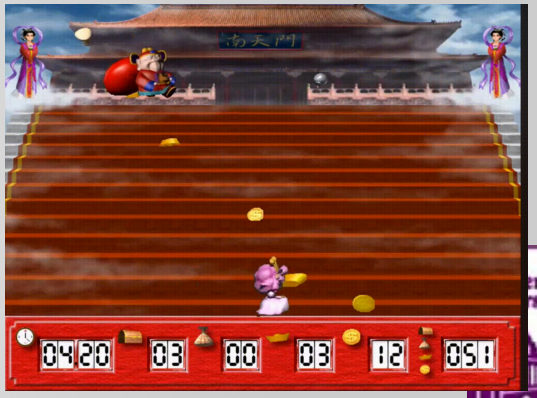
Lab3 (10%): Build a mini coin collection game

- One of the mini-game in Richman 4 is moving characters and collecting coins and avoiding the bomb.

Lab4 (10%): Build a mini coin collection game

- In Lab 4, you will create a game similar to Richman 4 using SpriteKit, with the following requirements:
- The player must be able to move left and right along the x-axis.
- Coins and bombs are randomly generated and drop from the top of the screen.
- Coins and bombs will disappear upon hitting the ground.
- The player must collect coins before they hit the ground, with the total number of coins collected being tracked and displayed.
- The game ends immediately if the player contacts a bomb.
- You may design your own character profile and build your own game UI.

https://www.gameartguppy.com/



Q & A