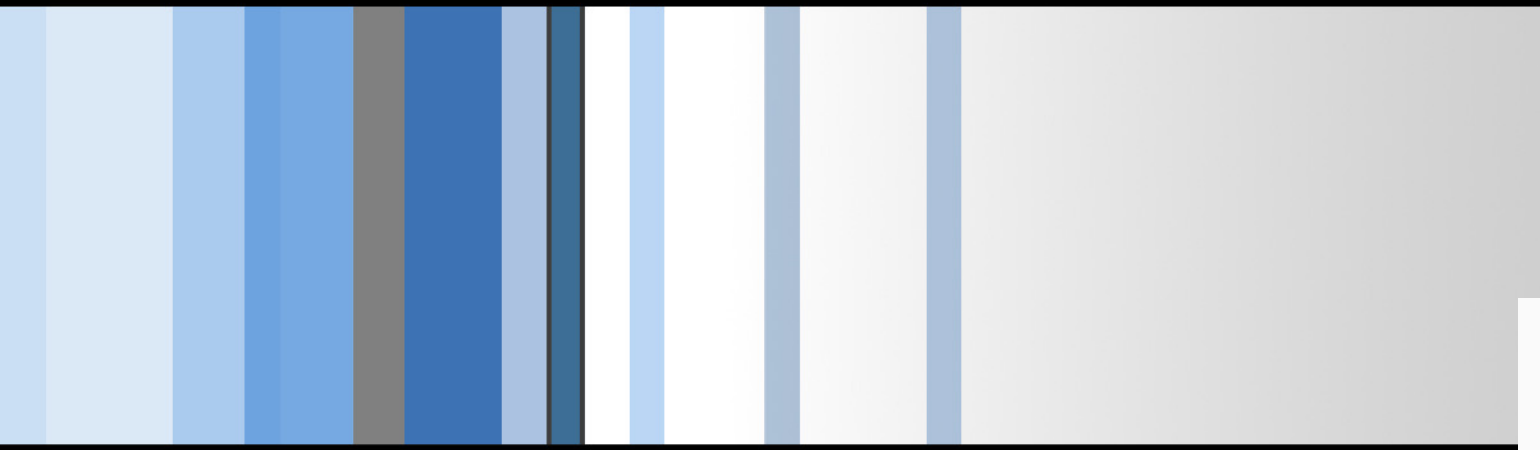
Class 19

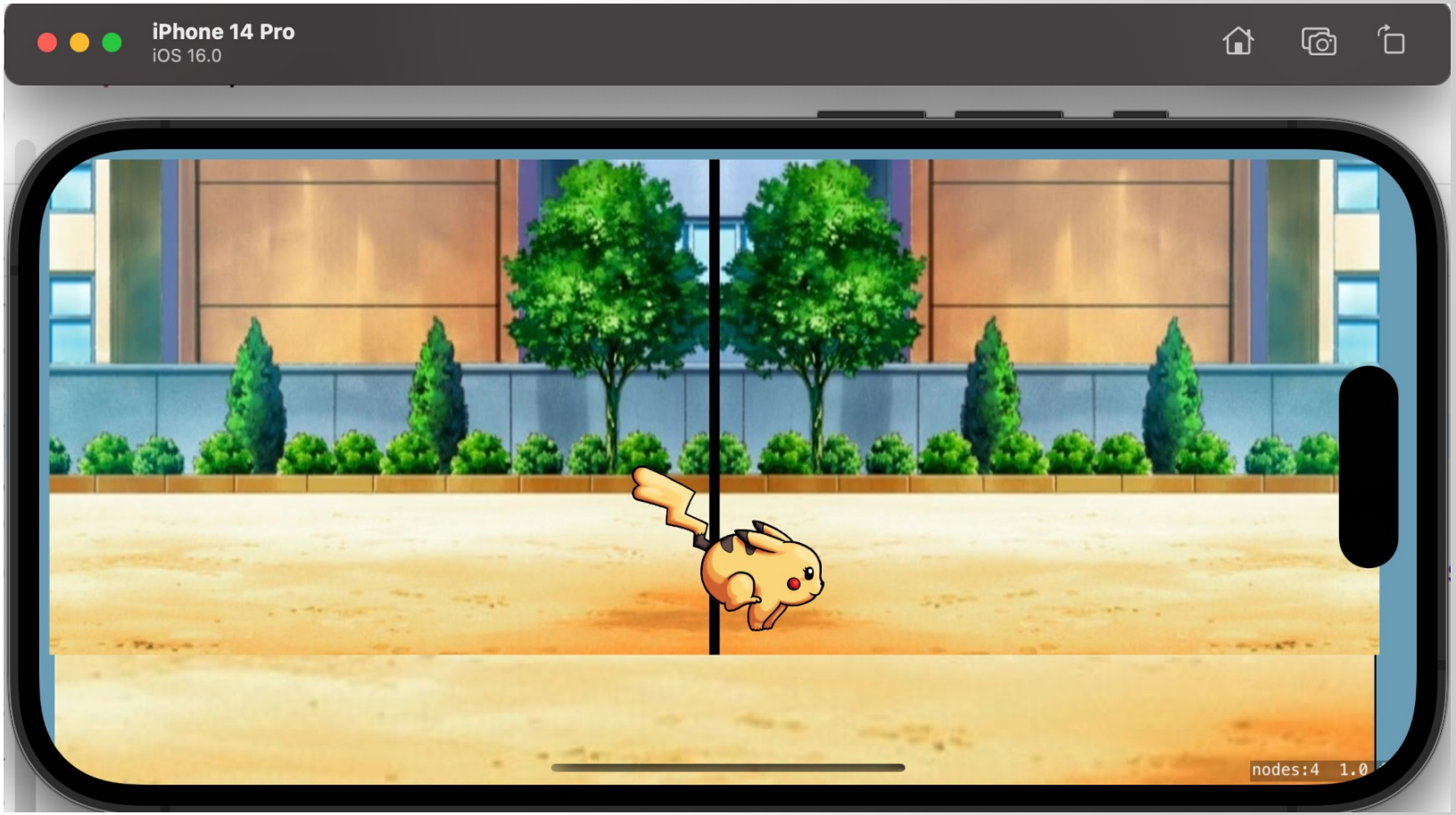
CSC 496: iOS App Development
SpriteKit (2)

Si Chen (schen@wcupa.edu)



SpriteKit

Run our game



Use Constrains to Limit Movement

```
class Player:SKSpriteNode{
    // MARK: - PROPERTIES

    // MARK: - 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
    }
    required init?(coder aDecoder: NSCoder){
        fatalError("init(coder:) has not been implemented")
    }

    func setupConstraints(floor: CGFloat){
        let range = SKRange(lowerLimit: floor, upperLimit: floor)
        let lockToPlatform = SKConstraint.positionY(range)
        constraints = [lockToPlatform]
    }

    func moveToPosition(pos: CGPoint, speed: TimeInterval){
        let moveAction = SKAction.move(to: pos, duration: speed)
        run(moveAction)
    }
```

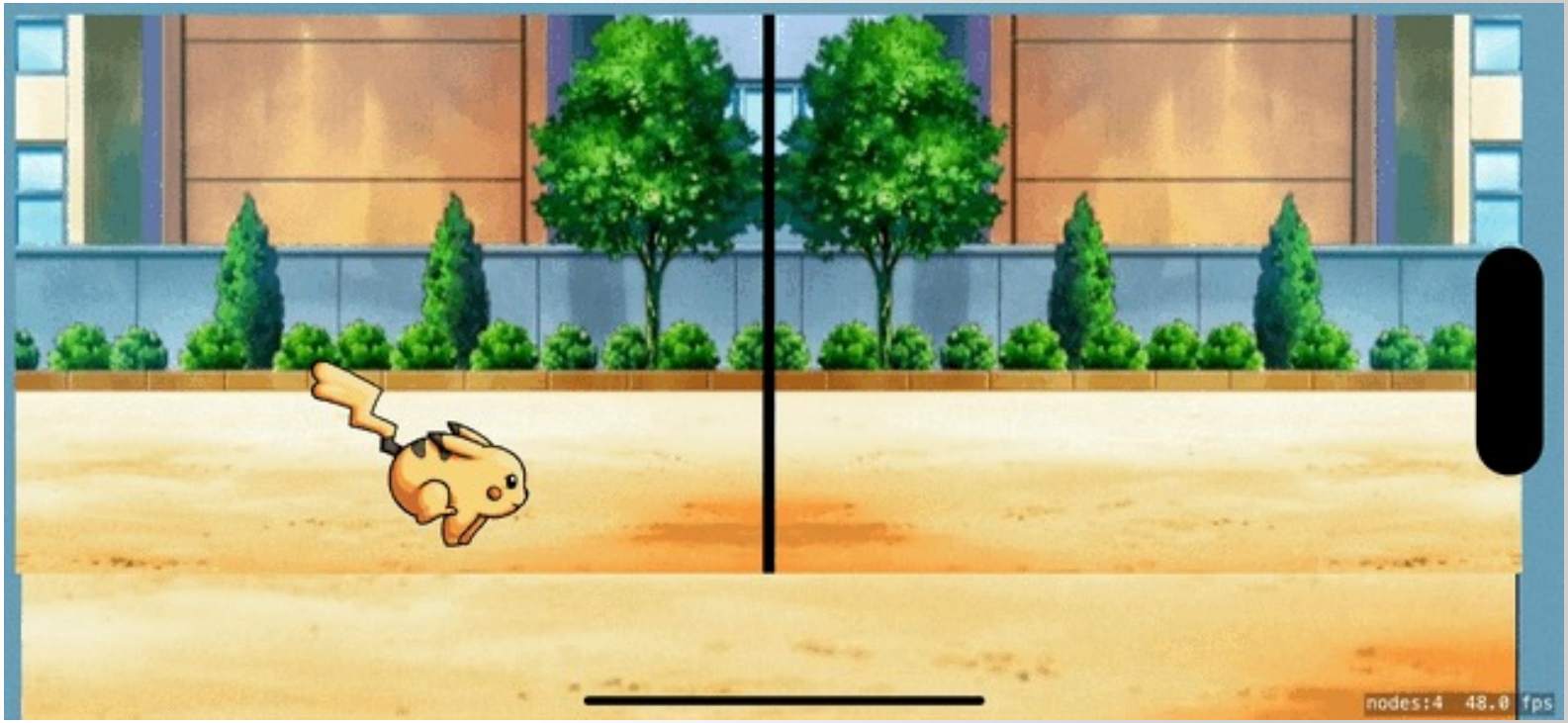

Please add setupConstraints() to Player.swift

Use Constrains to Limit Movement

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

Calling setupConstraints() in GameScene.swift

Use Constrains to Limit Movement



Set Player's Direction Using Scales

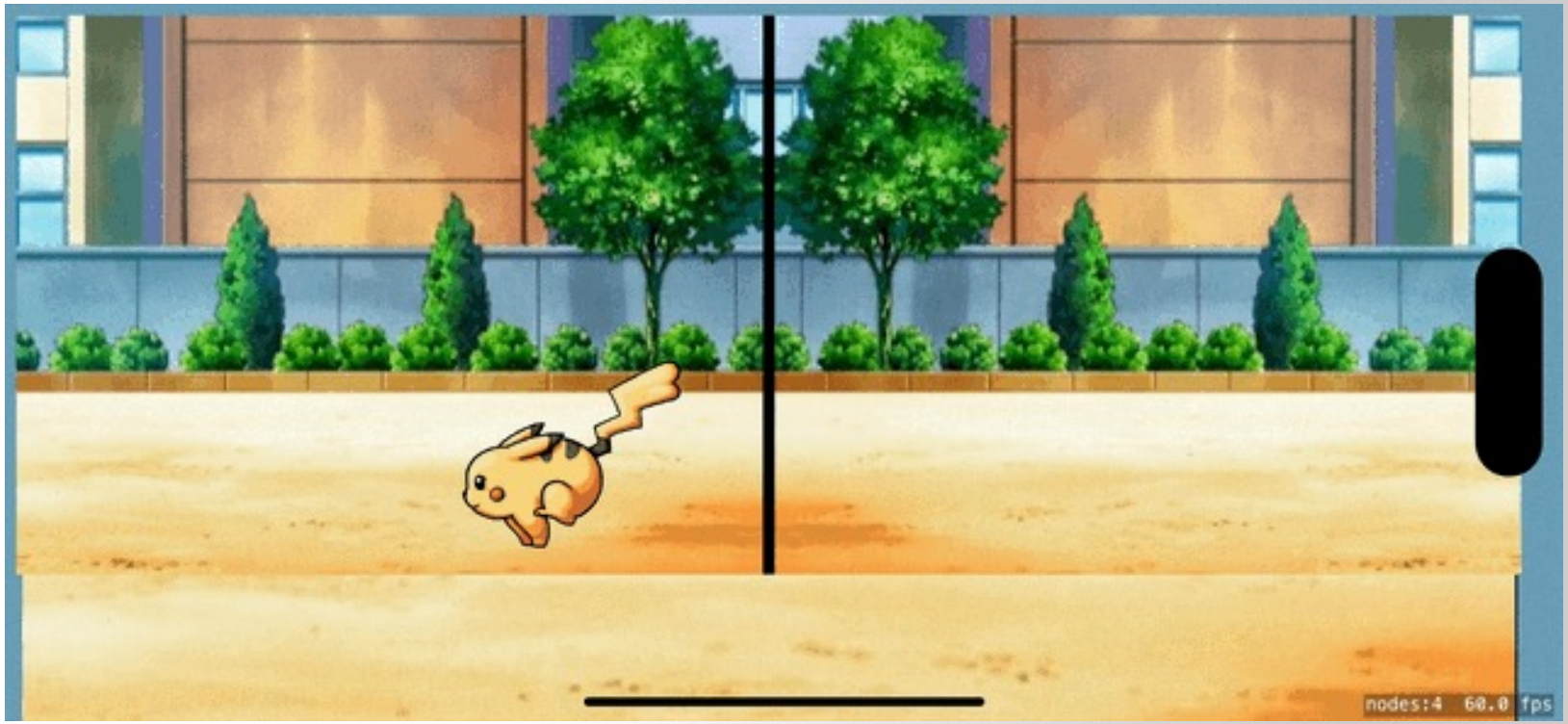
```
func moveToPosition(pos: CGPoint, direction: String, speed: TimeInterval){
    switch direction{
    case "L":
        xScale = -abs(xScale)
    default:
        xScale = abs(xScale)
    }

    let moveAction = SKAction.move(to: pos, duration: speed)
    run(moveAction)
}
```

Update moveToPosition() in Player.swift

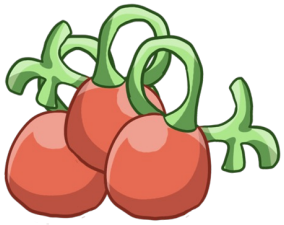
```
func touchDown(atPoint pos: CGPoint){
    if pos.x < player.position.x {
        player.moveToPosition(pos: pos, direction: "L", speed: 1.0)
    } else {
        player.moveToPosition(pos: pos, direction: "R", speed: 1.0)
    }
}
```

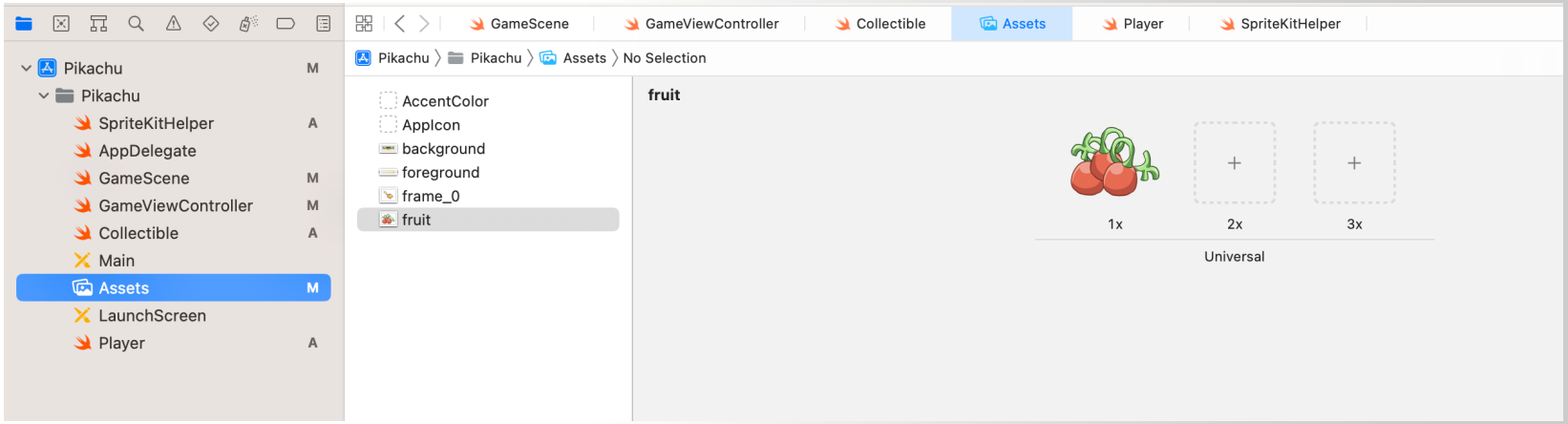
Update touchDown() in GameScene.swift

Set Player's Direction Using Scales

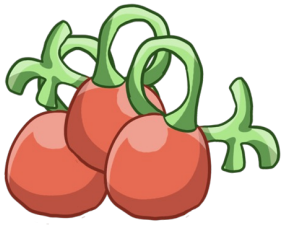


Add Your First Collectible Item -- Fruit





Add Your First Collectible Item -- Fruit



```
import Foundation
import SpriteKit


class Collectible:SKSpriteNode{

}
```

Create a new file: Collectible.swift

In Collectible.swift, above the Collectible class, add the following code:

```
import Foundation
import SpriteKit

enum CollectibleType: String{
    case none
    case fruit
}
class Collectible:SKSpriteNode{

}
```

Add Your First Collectible Item -- Fruit

- Inside SpriteKitHelper.swift, update the Layer enum

```
import Foundation
import SpriteKit

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

Add Your First Collectible Item -- Fruit

- Inside and at the top of the collectible class, add the following block of code:

```
enum CollectibleType: String{
    case none
    case fruit
}
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
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
}
```

Add Your First Collectible Item -- Fruit

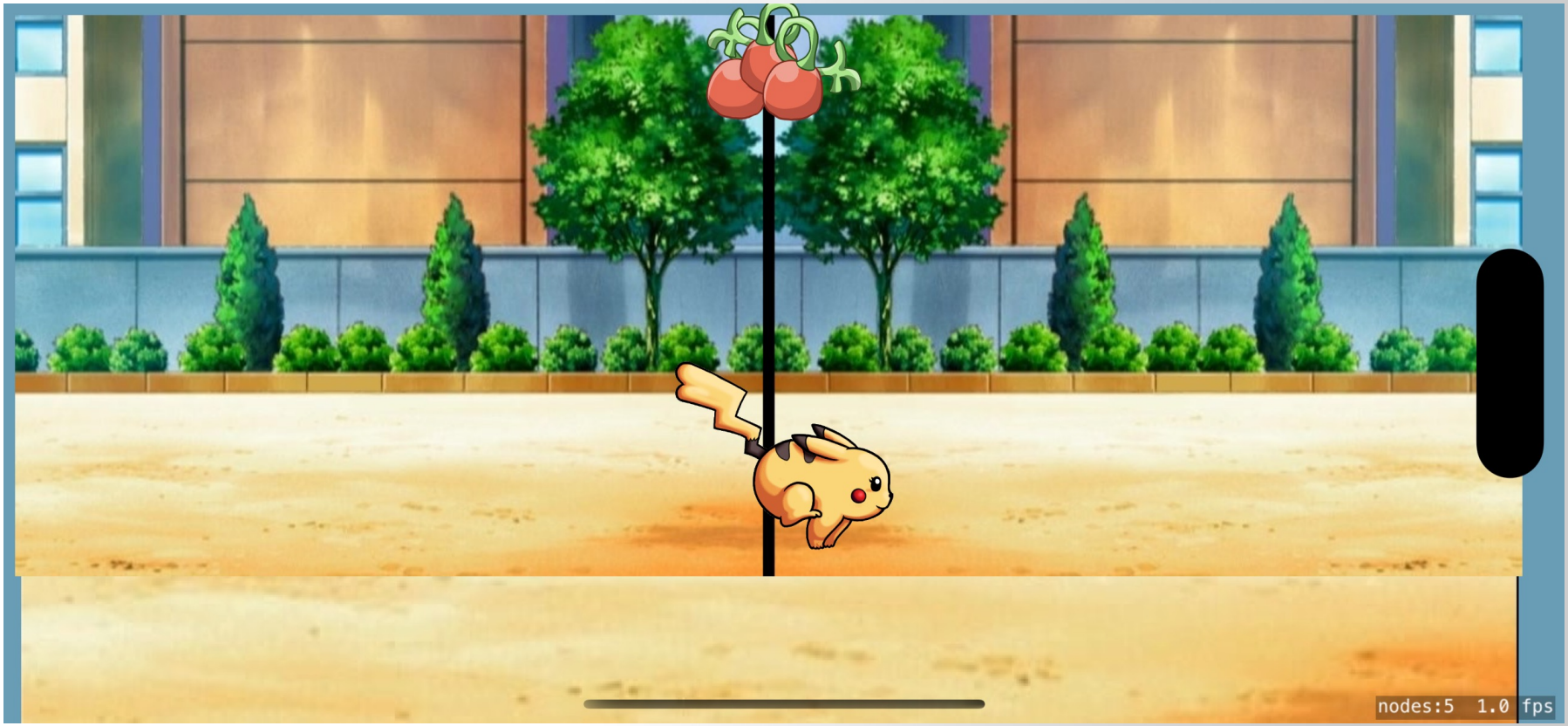
- Inside GameScene.swift, immediately below the didMove() method, add the following code:

```
func spawnFruit(){
    let collectible = Collectible(collectibleType: CollectibleType.fruit)
    collectible.position = CGPoint(x:player.position.x, y:player.position.y * 2.5)
    addChild(collectible)
}
```

- And call the spawnFruit() method --> inside the didMove() method, add spawnFruit() at the bottom.

```
override func didMove(to view: SKView) {
    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
    addChild(foreground)

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

Add Your First Collectible Item -- Fruit



Chain Actions Together to Create a Sequence

- Open Collectible.swift file, below the initialization methods, add the drop() method:

```
func drop(dropSpeed: TimeInterval, floorLevel: CGFloat){
    let pos = CGPoint(x: position.x, y: floorLevel)
    let scaleX = SKAction.scaleX(to: 0.5, duration: 1)
    let scaleY = SKAction.scaleY(to: 0.5, duration: 1)
    let scale = SKAction.group([scaleX, scaleY])

    let appear = SKAction.fadeAlpha(to: 1, duration: 0.25)
    let moveAction = SKAction.move(to: pos, duration: dropSpeed)
    let actionSequence = SKAction.sequence([appear, scale, moveAction])

    self.scale(to: CGSize(width: 0.25, height: 1.0))
    self.run(actionSequence, withKey: "drop")
}
```

In GameScene.swift file, at the end of the spawnFruit() method, add this line of code:

```
func spawnFruit(){
    let collectible = Collectible(collectibleType: CollectibleType.fruit)
    collectible.position = CGPoint(x:player.position.x, y:player.position.y * 2.5)
    addChild(collectible)
    collectible.drop(dropSpeed: TimeInterval(1.0), floorLevel: player.frame.minY)
}
```

Chain Actions Together to Create a Sequence

Spawn Multiple Fruits!

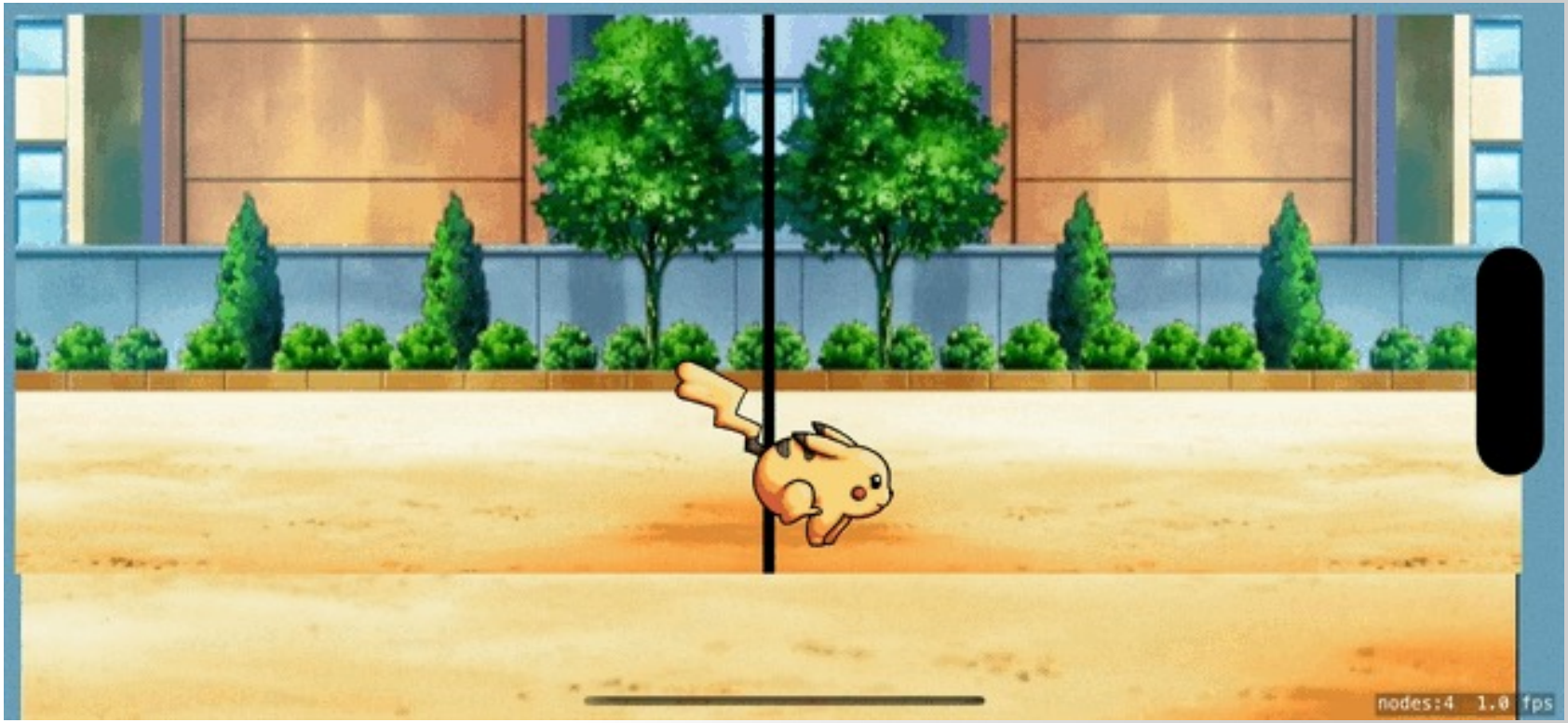
In GameScene.swift, below the spawnFruit() method, add the following new method

```
func spawnMultipleFruits(){
    // set up repeating action

    let wait = SKAction.wait(forDuration: TimeInterval(1.0))
    let spawn = SKAction.run {
        self.spawnFruit()
    }
    let sequence = SKAction.sequence([wait, spawn])
    let repeatAction = SKAction.repeat(sequence, count: 10)
    run(repeatAction, withKey: "fruit")
}
```

And change the spawnFruite() in didMove() to spawnMultipleFruits()

```
override func didMove(to view: SKView) {
    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
    addChild(foreground)

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

Spawn Multiple Fruits!



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)
}
```

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 setupLables(){
    scoreLevel.name = "score"
    scoreLevel.fontColor = .black
    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) {
    // set up the physics world contact delegate
    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
    // 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)

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

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

Q & A