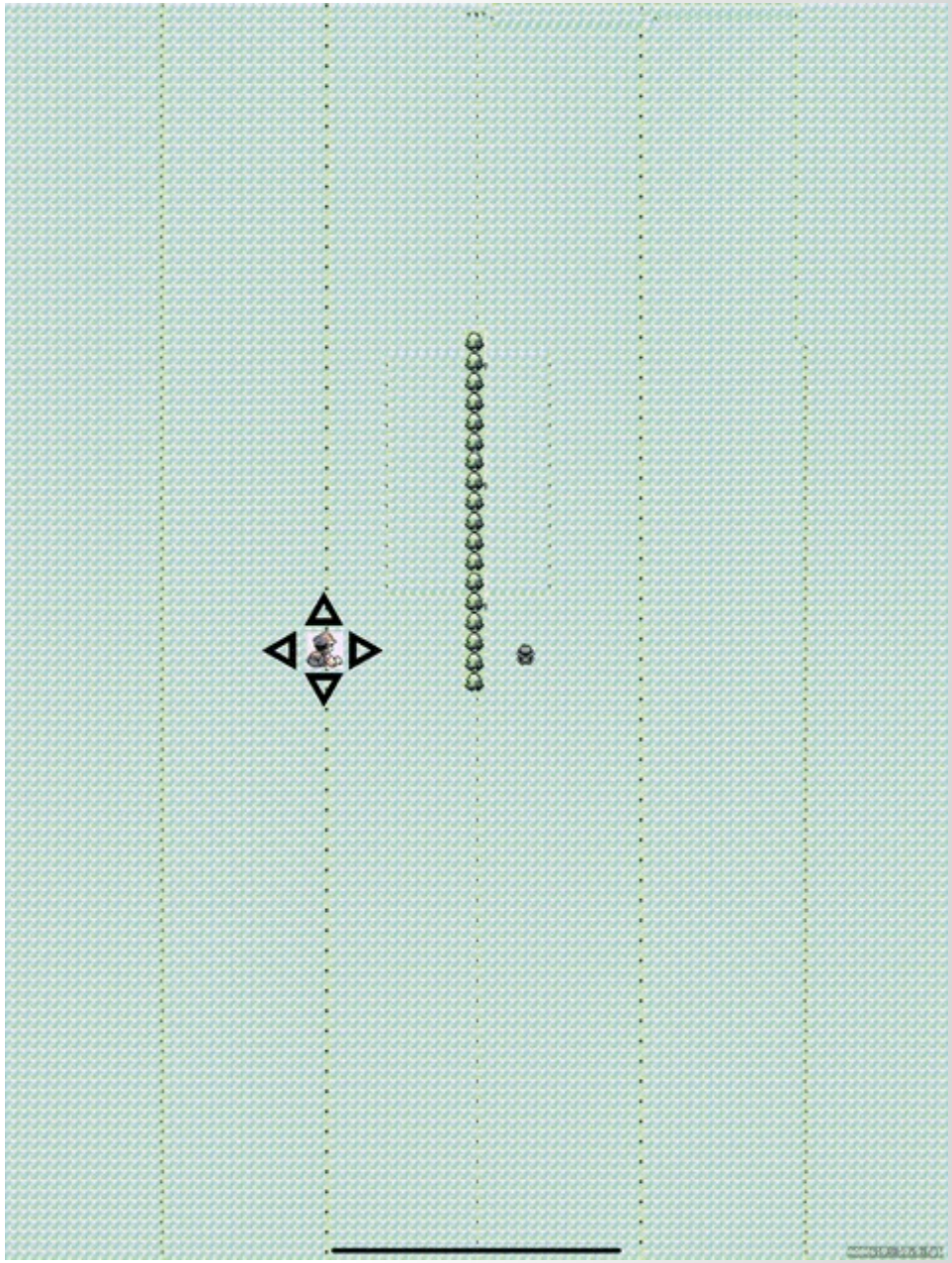
Class 22

CSC 496: iOS App Development
SpriteKit (4): Use of Scene Editor

Si Chen (schen@wcupa.edu)

SpriteKit

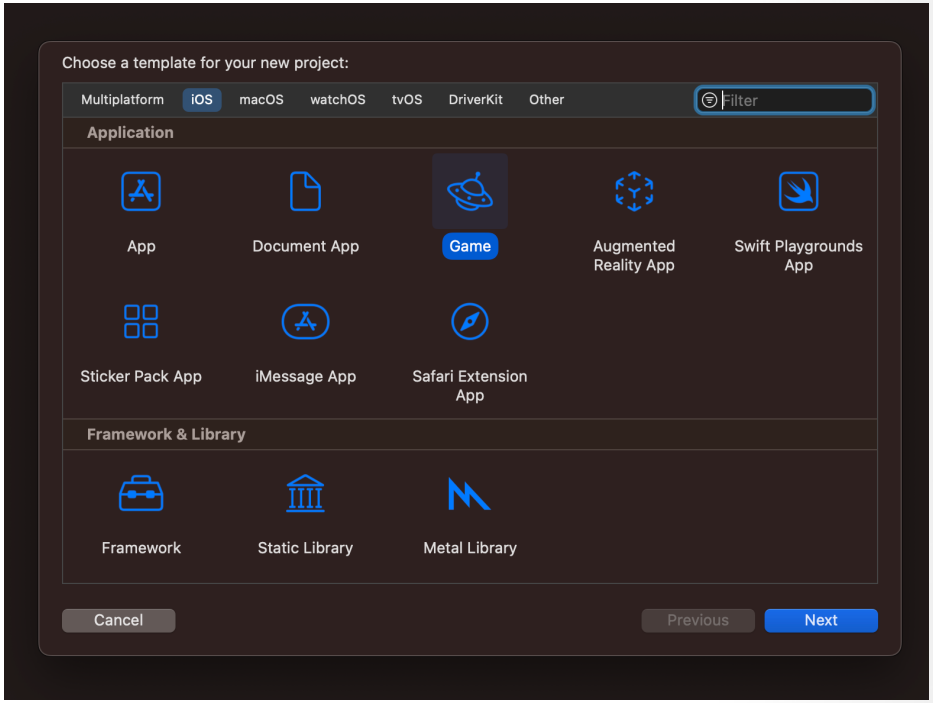
Pokemon2D game

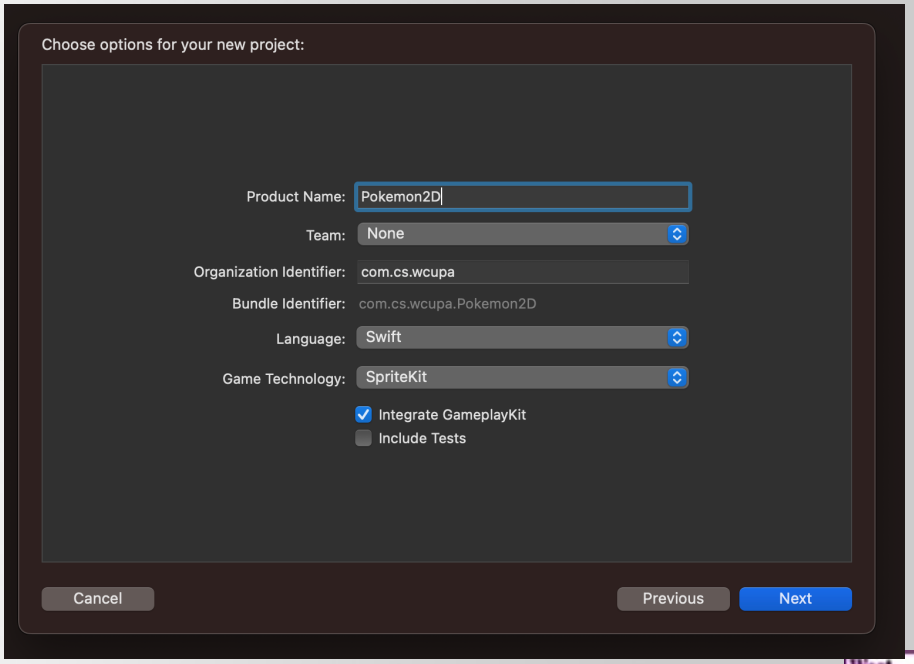


Use the Scene Editor to build 2D games

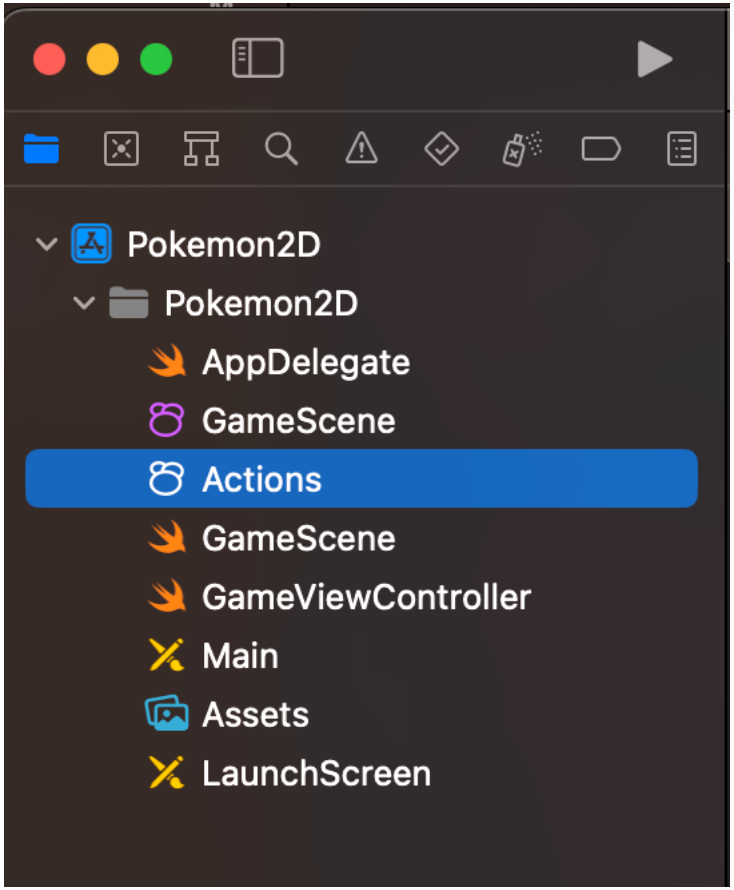
- In this project, we will use the Scene Editor to build a 2D game.
- The Scene Editor is a built-in feature of Xcode that allows us to create SpriteKit nodes and assign values to them.

Create a new Pokemon2D game project





Clean up the default template



Delete Action.sks
Do NOT delete the GameScene.sks

Clean up the default template

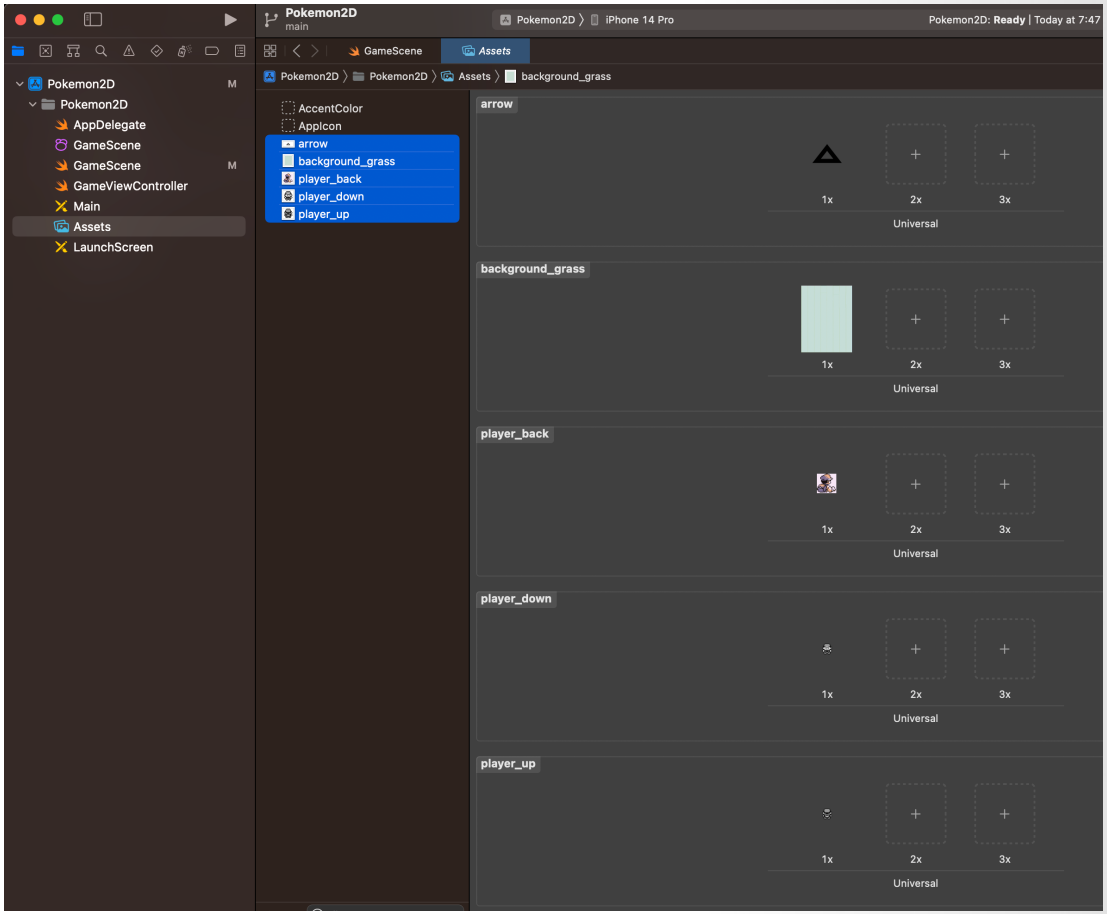
```
import SpriteKit
import GameplayKit

class GameScene: SKScene {

    var entities = [GKEntity]()
    var graphs = [String : GKGraph]()

    private var lastUpdateTime : TimeInterval = 0

    override func sceneDidLoad() {

        self.lastUpdateTime = 0

    }


    func touchDown(atPoint pos : CGPoint) {

    }

    func touchMoved(toPoint pos : CGPoint) {

    }

    func touchUp(atPoint pos : CGPoint) {

    }

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        for t in touches { self.touchDown(atPoint: t.location(in: self)) }
    }

    override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
        for t in touches { self.touchMoved(toPoint: t.location(in: self)) }
    }

    override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
        for t in touches { self.touchUp(atPoint: t.location(in: self)) }
    }

    override func touchesCancelled(_ touches: Set<UITouch>, with event: UIEvent?) {
        for t in touches { self.touchUp(atPoint: t.location(in: self)) }
    }
```

In GameScene.swift

1. Remove all of the code in the *sceneDidLoad()* method, leaving only the line *self.lastUpdateTime = 0*
2. Remove all of the code in the *touchdown()*, *touchMoved()* and *touchUp()* methods
3. Remove the If statement inside the *touchesBegan()* method.
4. Remove the *label* and *spinnyNode* properties on the top

<-- The final code should look like this

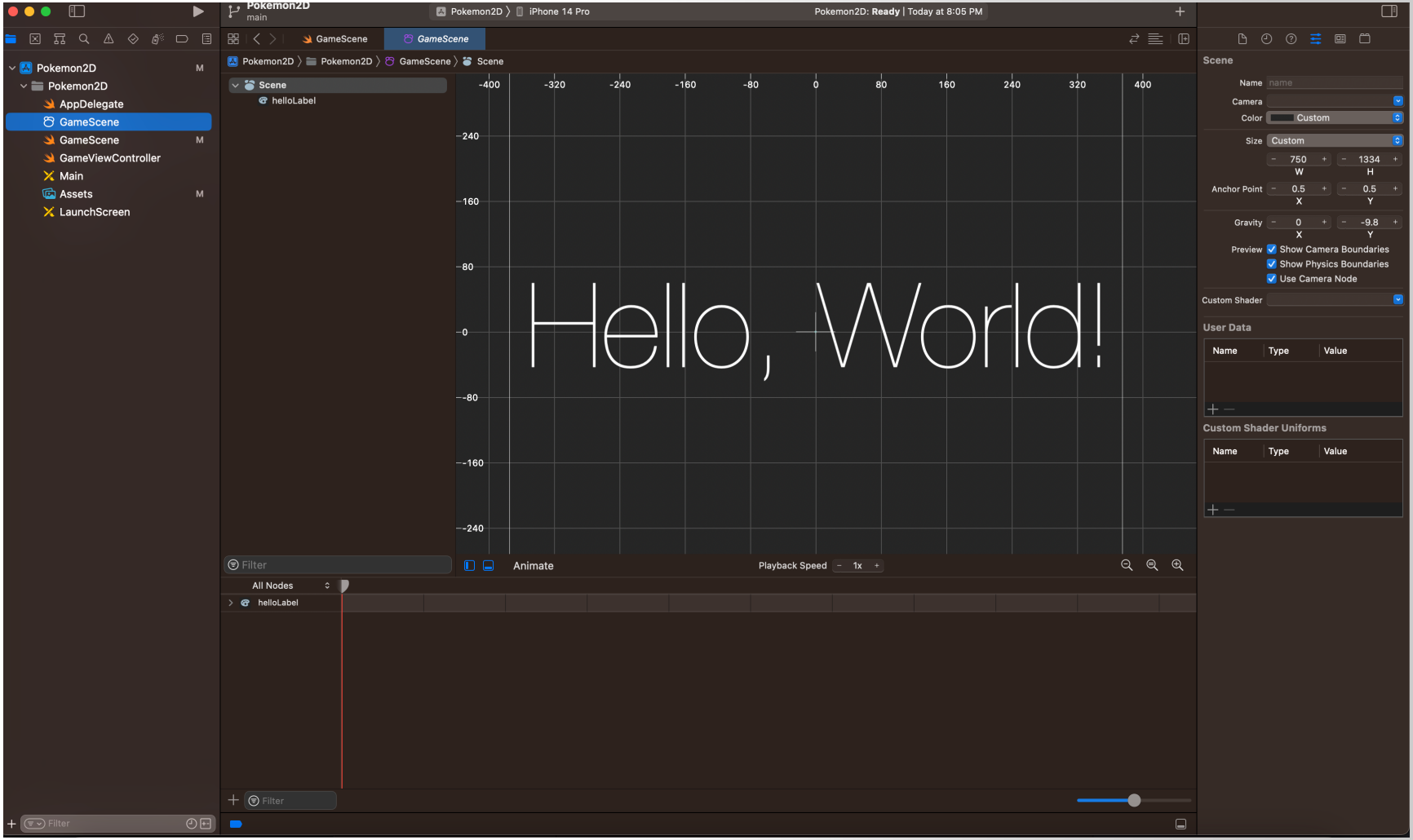
Adding the Assets



Download assets from class website,

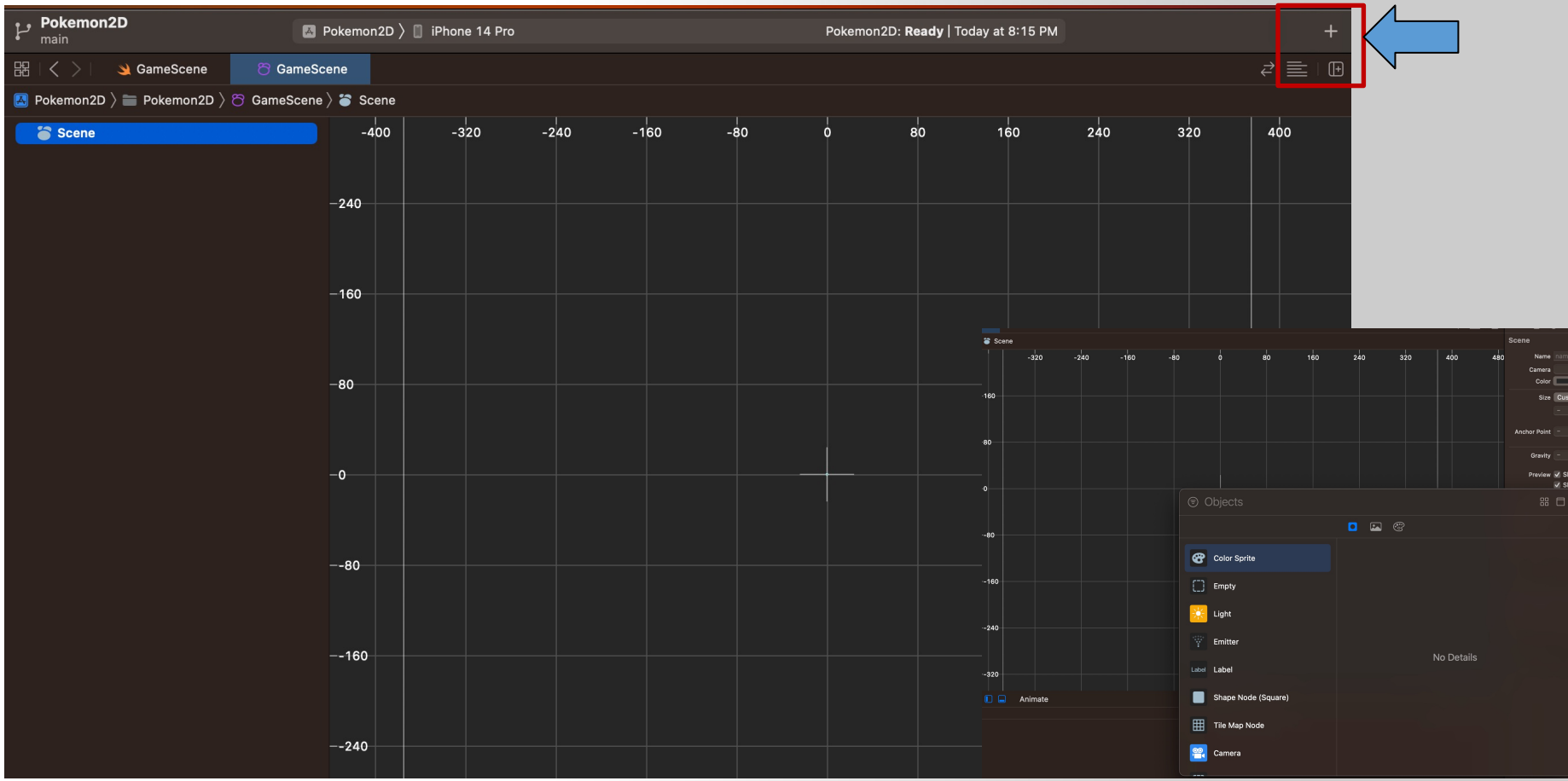
Drag and drop them into the Assets folder

Use Scene Editor to Add nodes



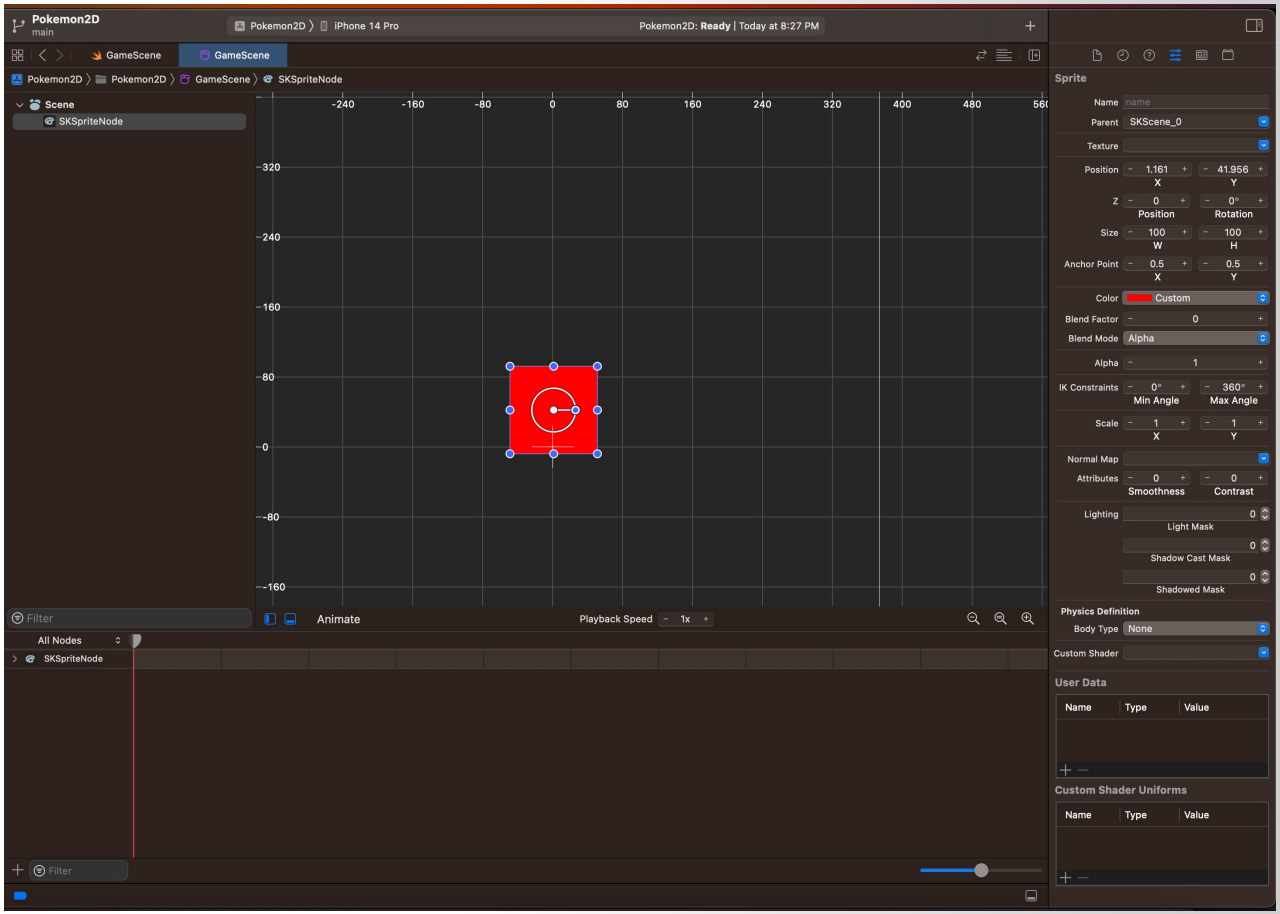
Click GameScene.sks

Use Scene Editor to Add nodes



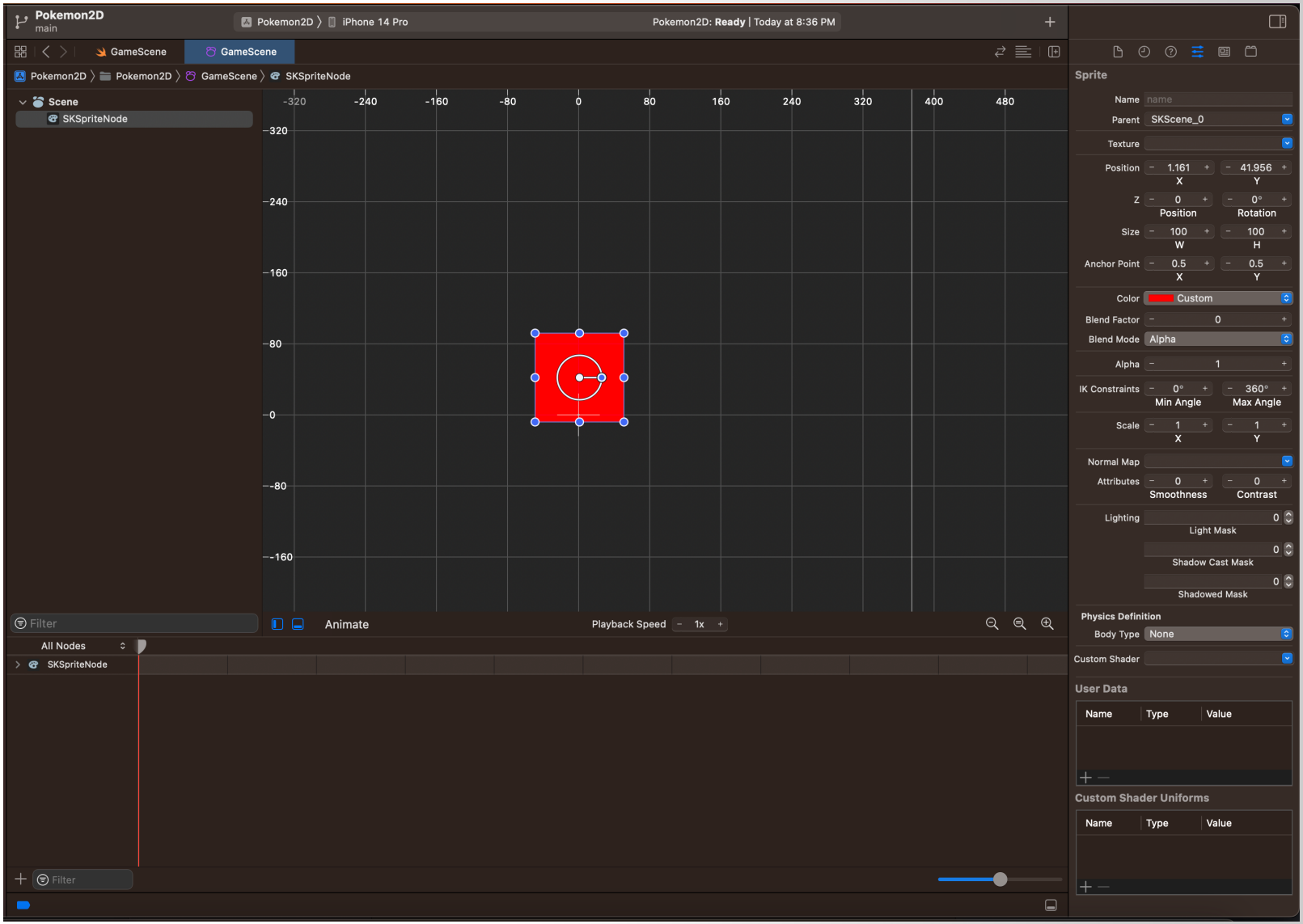
1. Delete the default “Hello world” Node
2. Click the “+” sign
3. Drag and drop the “Color Sprite” to the Scene

Use Scene Editor to Add nodes

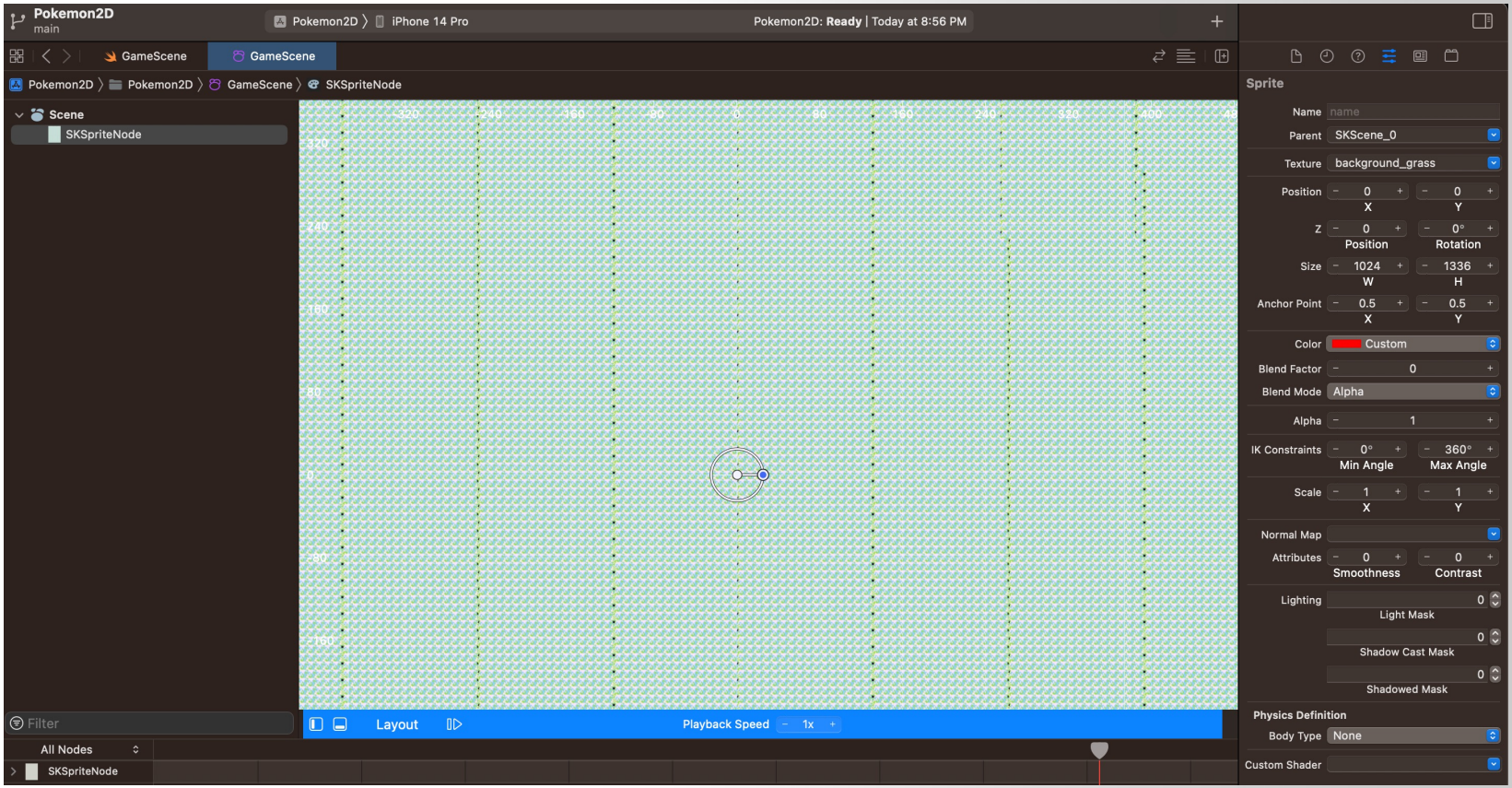


1. Delete the default “Hello world” Node
2. Click the “+” sign
3. Drag and drop the “Color Sprite” to the Scene

Use Scene Editor to Add nodes

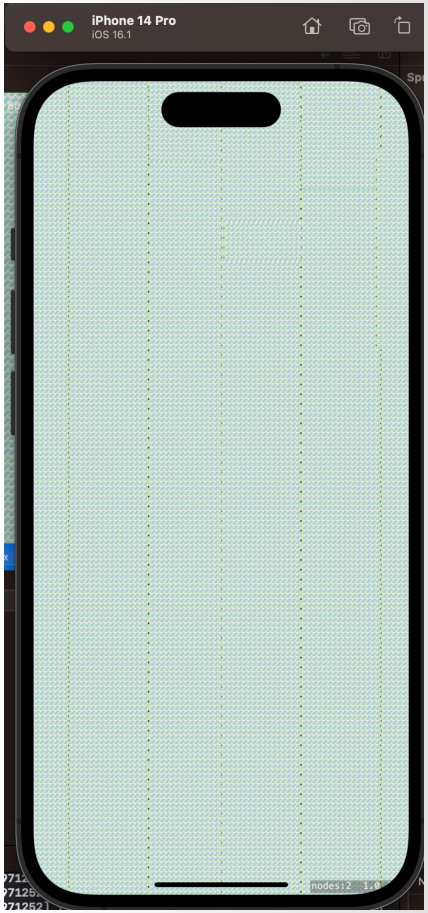


Use Scene Editor to Add nodes

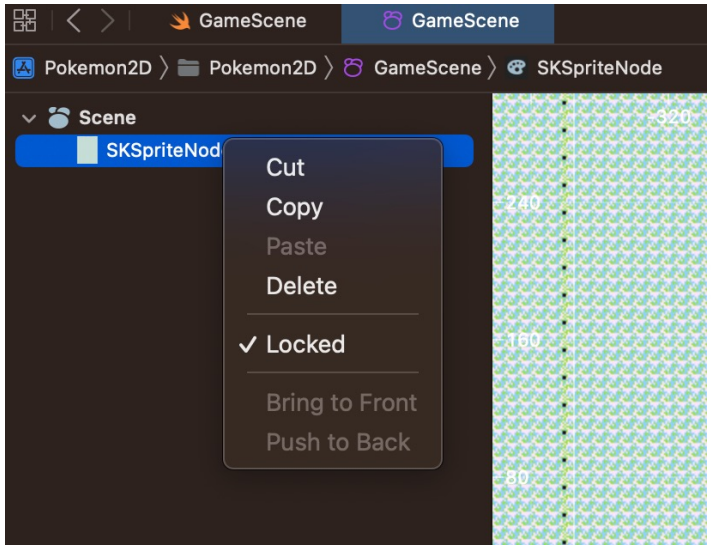


- Change texture to “background_grass”
- Set the position to x: 0 and y: 0

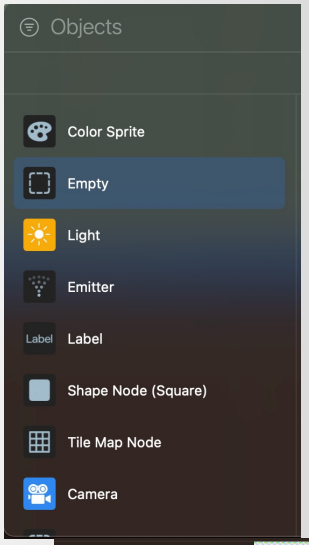
Running it



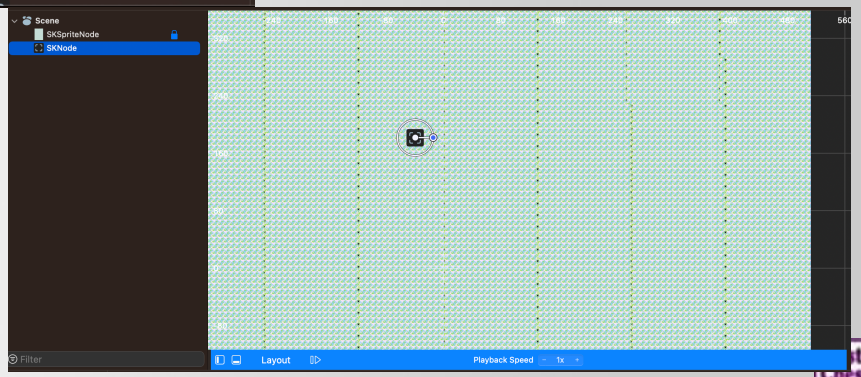
Adding other nodes



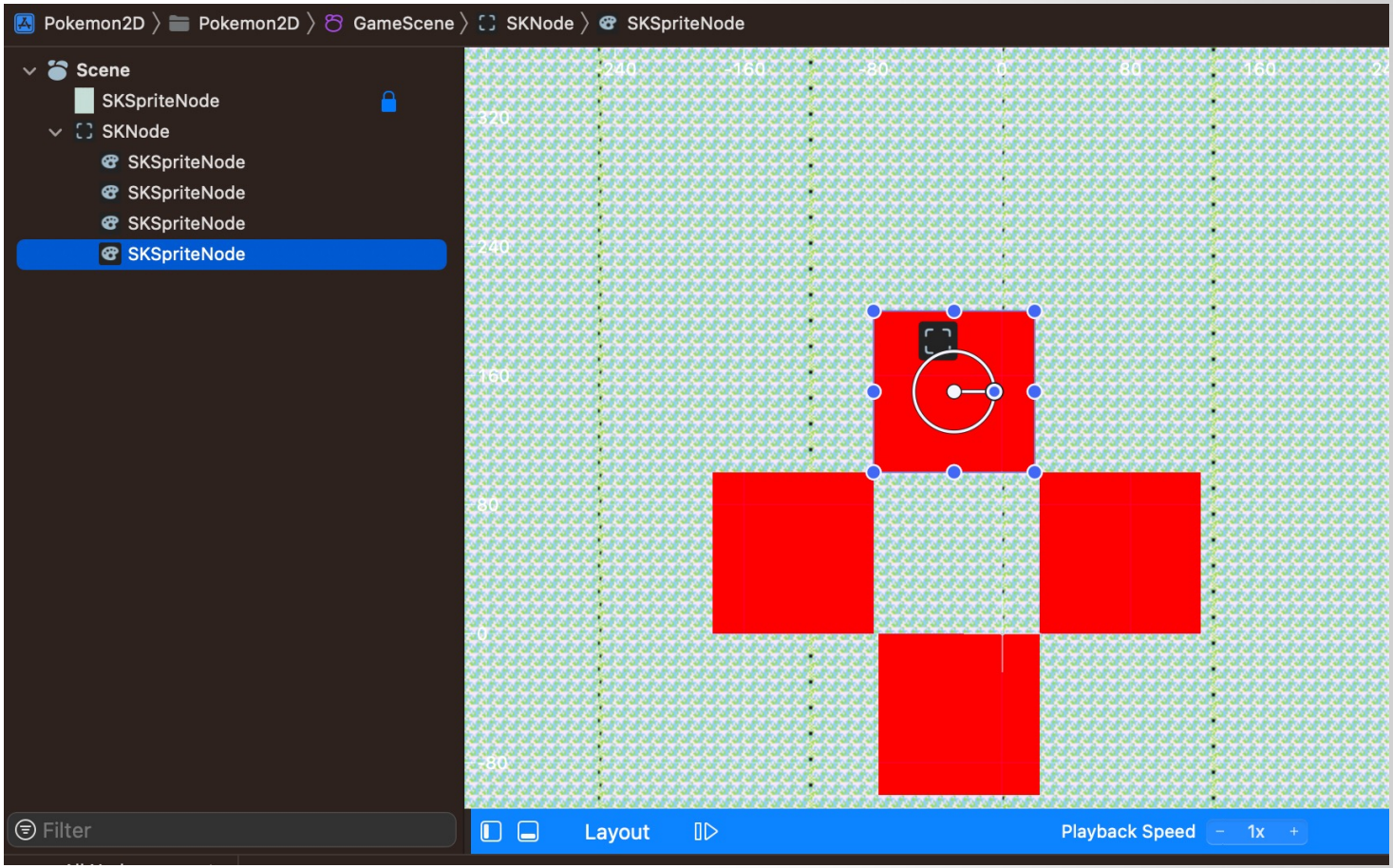
Right click the SKSpriteNode (background) and lock it



Drag “Empty” to the scene

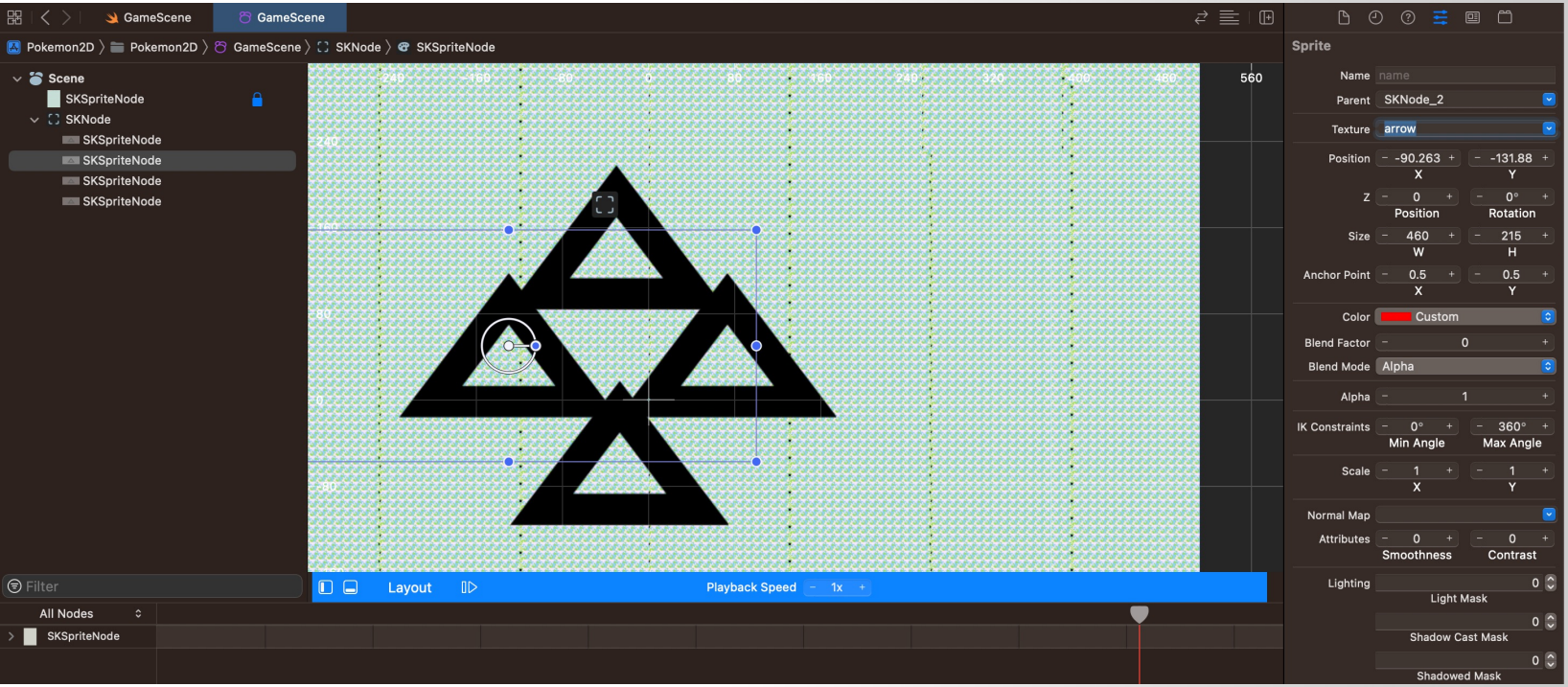


Adding other nodes



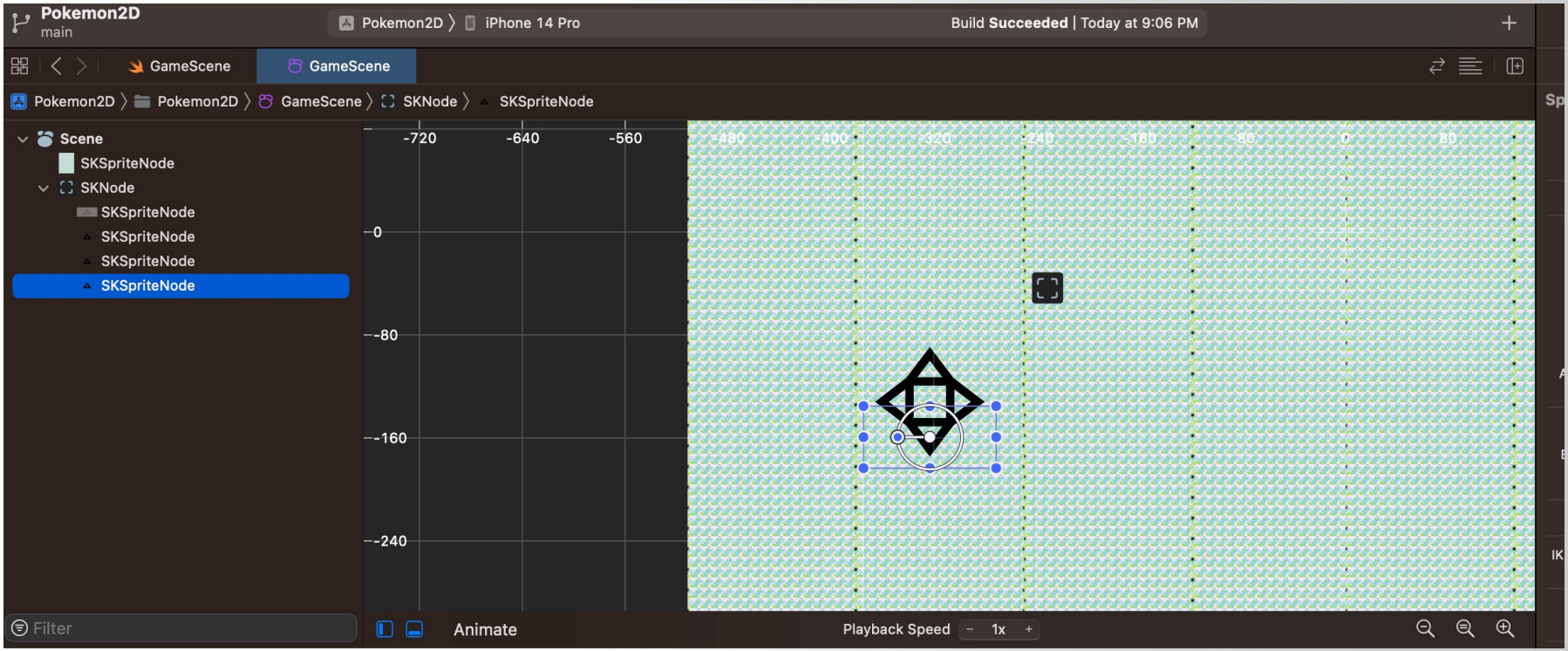
Add four more “Color Sprite” to the Scene and drag them under the SKNode (to group them up)

Adding other nodes



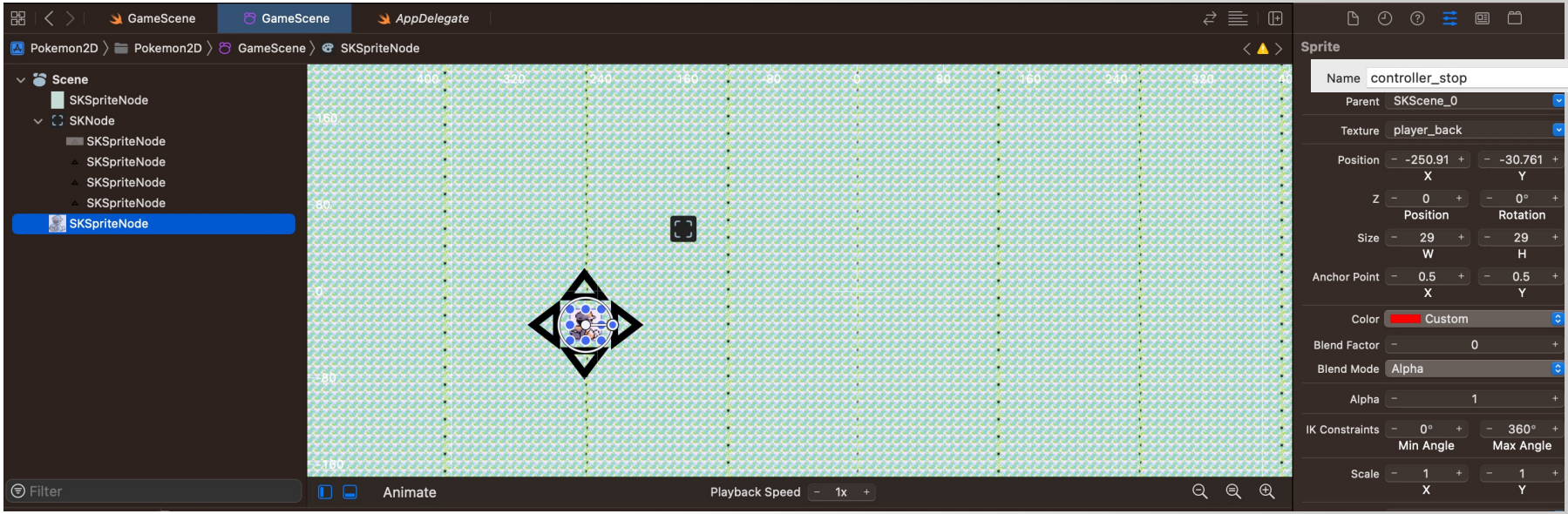
Change the texture of all four Color Sprite Nodes to “arrow”

Adding other nodes



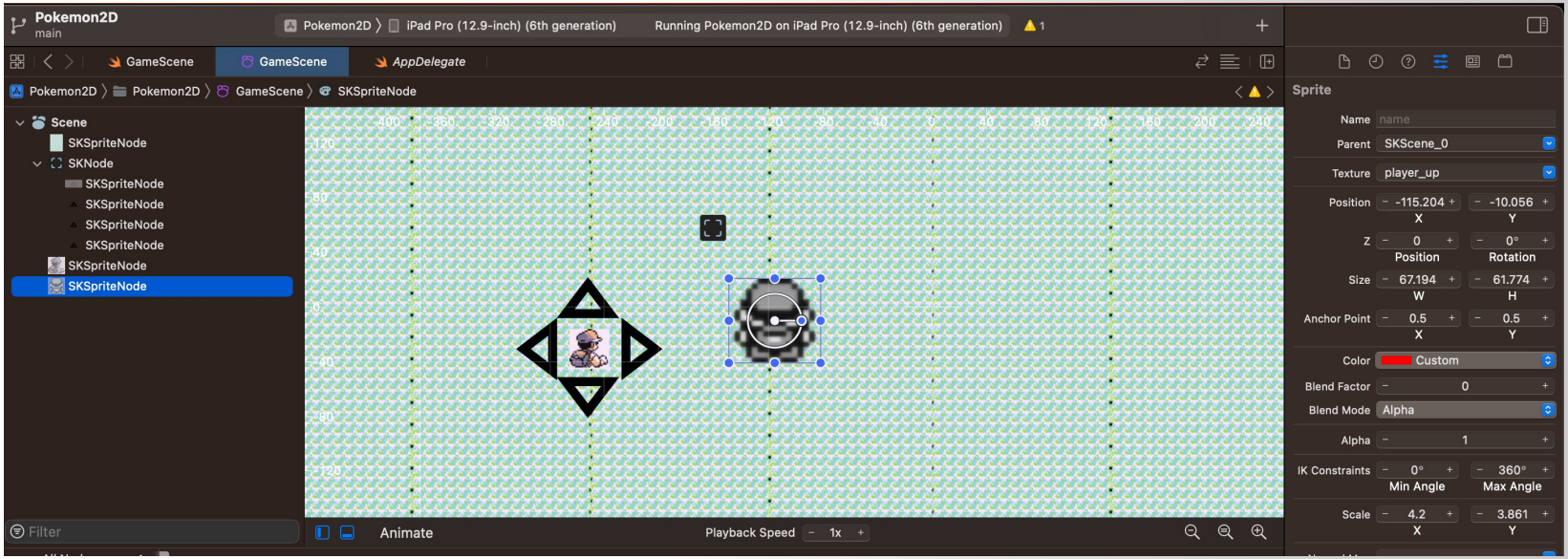
- adjust the size and rotation to make it like an on-screen controller (directional pad, aka D-Pad)

Adding other nodes



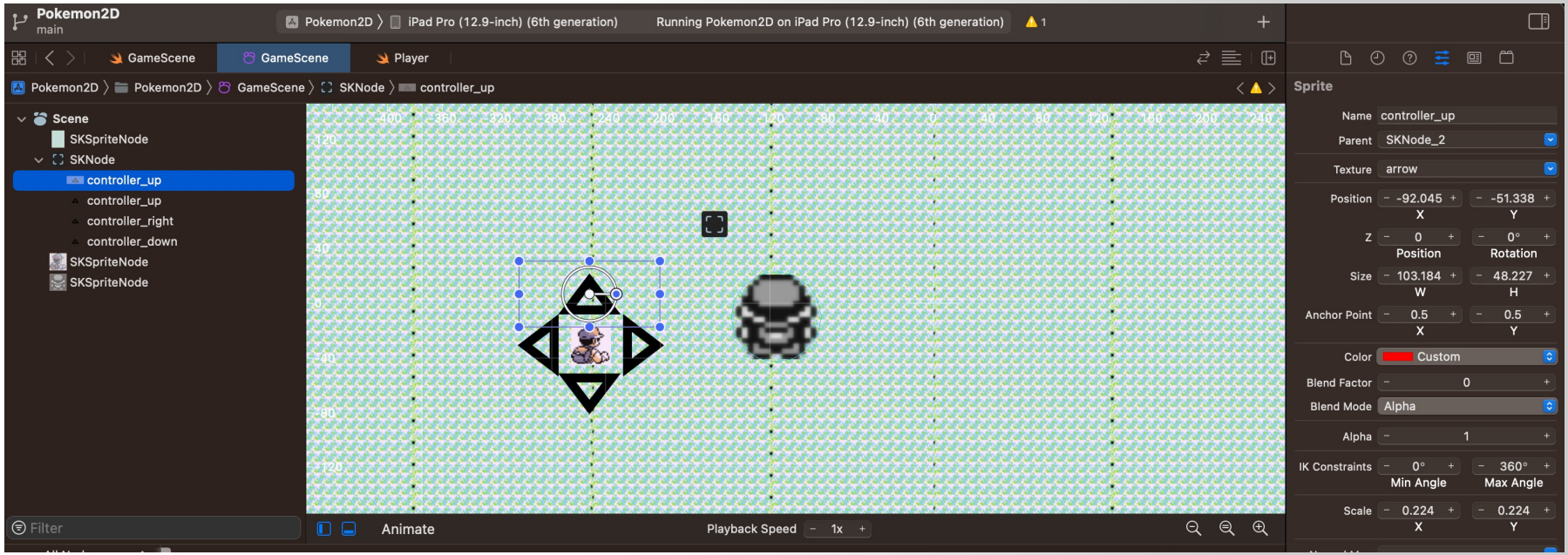
Add another Color Sprite Node and change to texture to “player_back”
And change it’s name to controller_stop

Adding other nodes



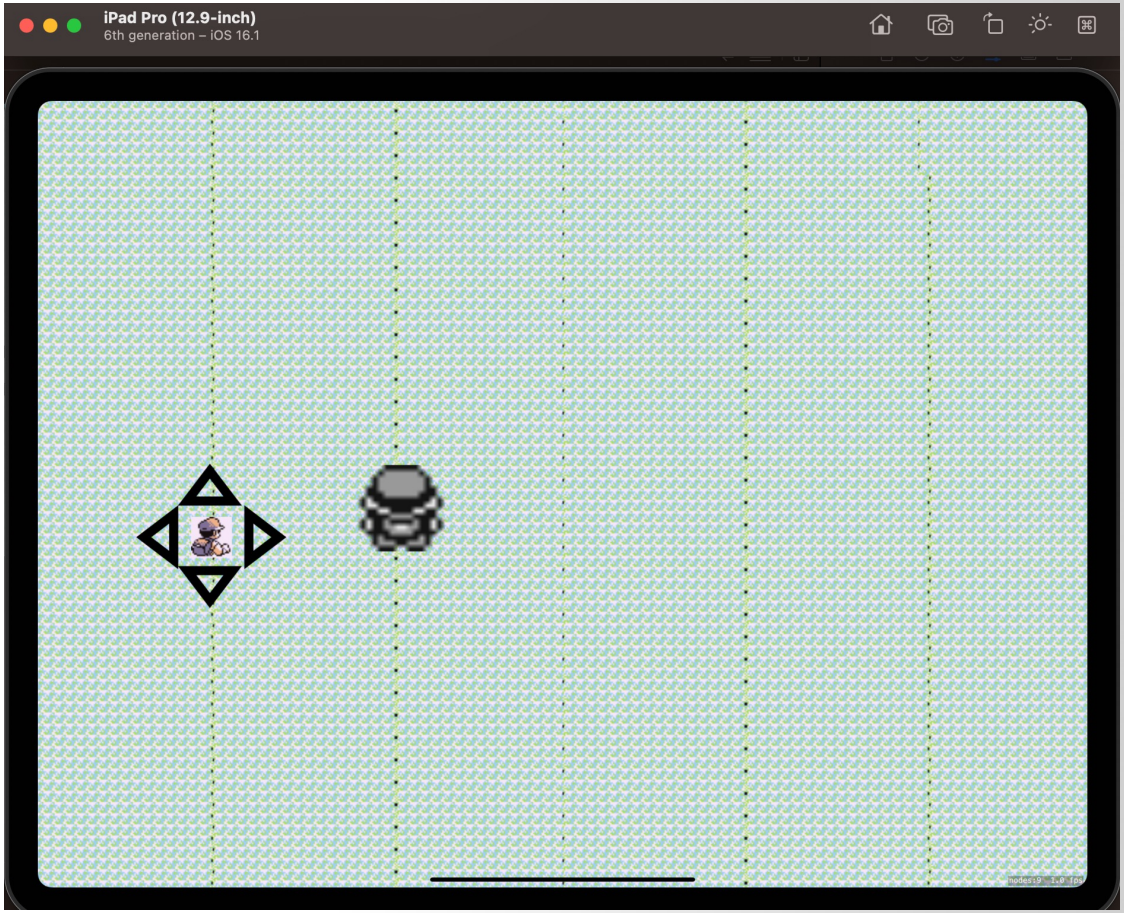
- Finally, add another Color Sprite Node (the player) and change the texture to “player_up”. You can adjust the size if you want.

Change D-pad nodes' name

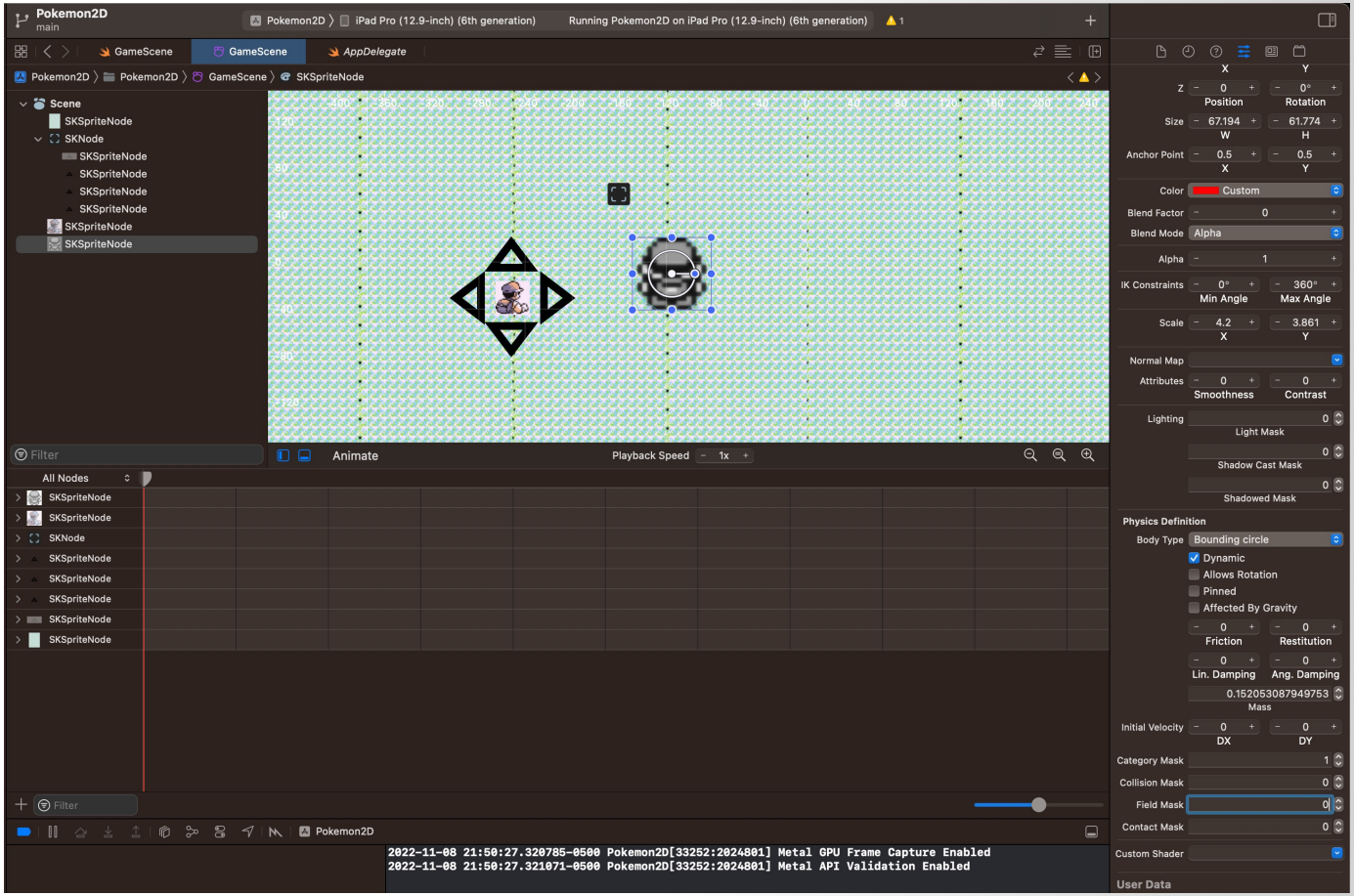


change the "name" of the d-pad button (arrow)'s name to controller_up, controller_down, controller_left, and controller_right accordingly

Run the game

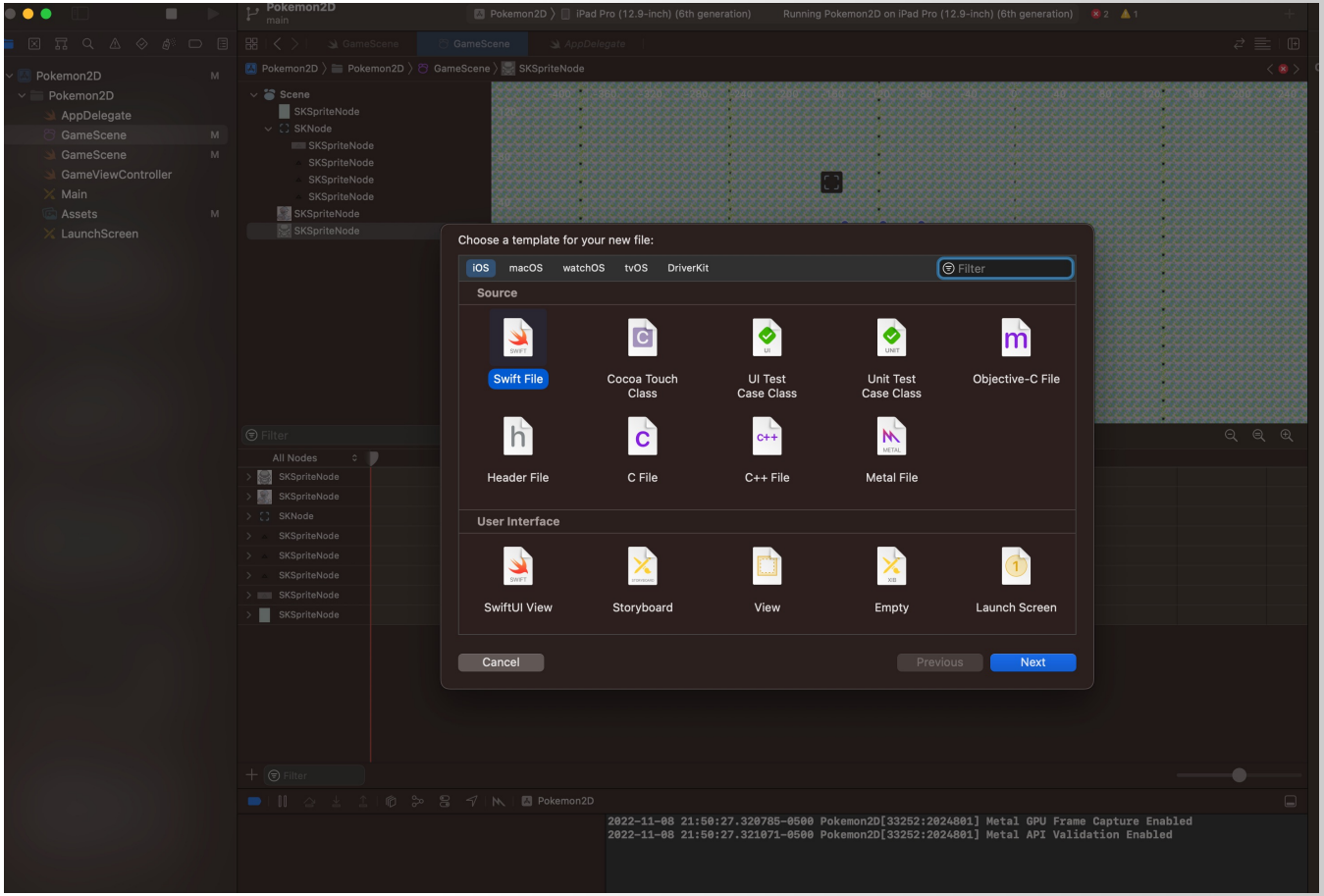


Use the Scene Editor to Add Physics



Click the Player node, change the “Body Type” to Bounding Circle, select “Dynamic” and deselect the allows rotation and affected by gravity options.
Change Fraction, restitution, Lin. Damping and Ang. damping to 0.
Set Category mask to 1, collision mask to 0, the field mask to 0 and the contact mask to 0

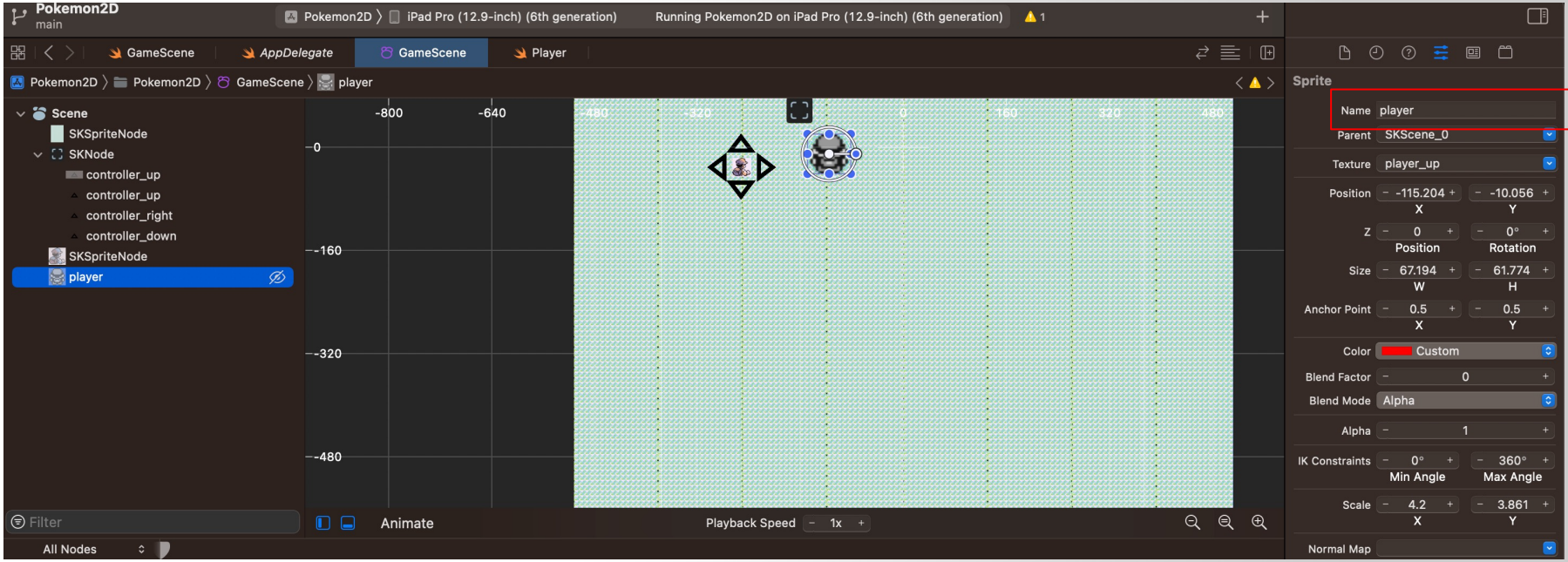
Create a Player class



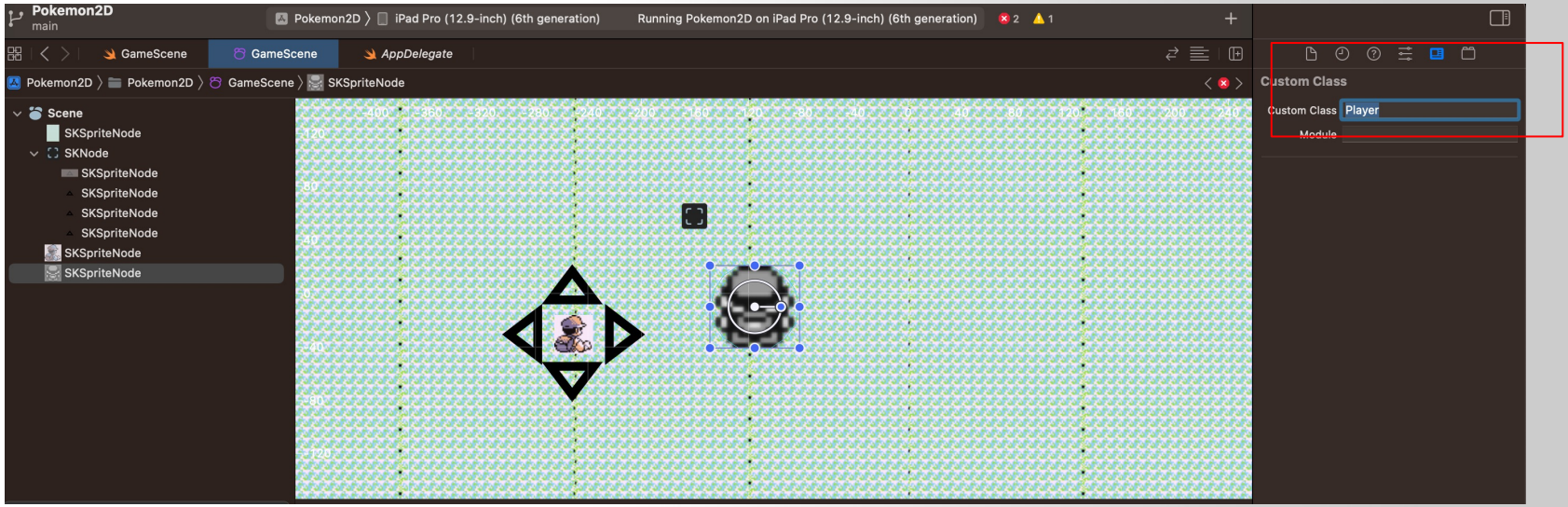
Create a Player class

```
import Foundation
import SpriteKit

enum Direction: String {
    case stop
    case left
    case right
    case up
    case down
}


class Player: SKSpriteNode{
    func move(_ direction: Direction){
        print("player move: \(direction.rawValue)")
    }


    func stop(){
        print("Stop")
    }
}
```

Assign Player node to the Player class



Assign Player node to the Player class

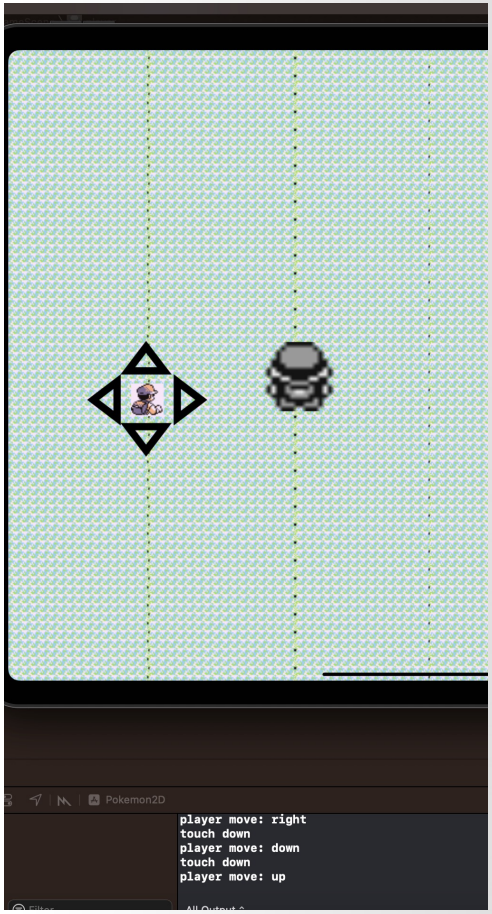


Move the player using Physics

- Open GameScene.swift file, override the didMove() method and update the touchdown() method:

```
import SpriteKit
import GameplayKit

class GameScene: SKScene {

    var entities = [GKEntity]()
    var graphs = [String : GKGraph]()

    private var lastUpdateTime : TimeInterval = 0
    private var player: Player?

    override func sceneDidLoad() {

        self.lastUpdateTime = 0

    }

    override func didMove(to view: SKView) {
        player = childNode(withName: "player") as? Player
        player?.move(.stop)
    }

    func touchDown(atPoint pos : CGPoint) {
        print("touch down")
        let nodeAtPoint = atPoint(pos)
        if let touchedNode = nodeAtPoint as? SKSpriteNode{
            if touchedNode.name?.starts(with: "controller_") == true{
                let direction = touchedNode.name?.replacingOccurrences(of: "controller_", with: "")
                player?.move(Direction(rawValue: direction ?? "stop")!)
            }
        }
    }
```

Run and test the game



Move the player use the Velocity method

```
import Foundation
import SpriteKit

enum Direction: String {
    case stop
    case left
    case right
    case up
    case down
}


class Player: SKSpriteNode{
    func move(_ direction: Direction){
        print("player move: \(direction.rawValue)")
        switch direction{
        case .up:
            self.physicsBody?.velocity = CGVector(dx: 0, dy: 100)
        case .down:
            self.physicsBody?.velocity = CGVector(dx:0, dy: -100)
        case .left:
            self.physicsBody?.velocity = CGVector(dx:-100, dy: 0)
        case .right:
            self.physicsBody?.velocity = CGVector(dx:100, dy: 0)
        case .stop:
            stop()
        }
    }


    func stop(){
        print("Stop")
        self.physicsBody?.velocity = CGVector(dx: 0, dy: 0)
    }
}
```

Exercise: 2D RPG Game

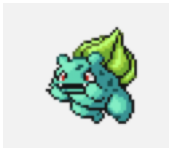
You can either start a new project or download Pokemon2DII from website

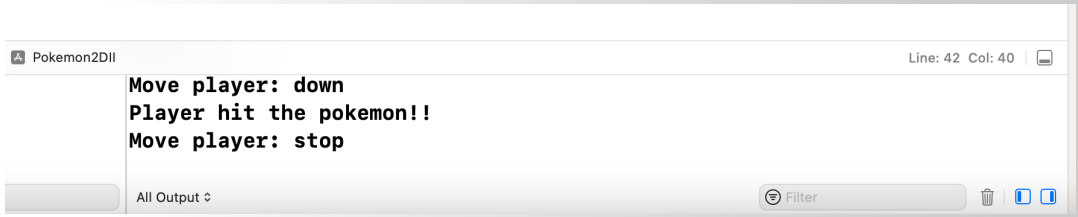
Tasks:

1. Add a new node – “tree” – to the game (the image can be downloaded from class website).

2. Use multiple tree nodes to create a maze that the player cannot pass through. (Hint: Set the Category Mask and Collision Mask for both the player and the tree node.)

3. Add a Pokémon node (node name: pokemon), and when the player hits the Pokémon, it will print "Player hit the Pokémon" in the console.





Q & A