//opengameart.org

Actions.sks

//GameViewController.swift ⇒ view.showPhysics

//GameScene.sks

(0,0) ⇒ left Bottom for the anchor point

Preview ⇒ everything checked

only in landscape mode ⇒ also change p-list ⇒ delete the portrait mode

Texture ⇒ change background

name the bird and the boxes

//GameScene.swift (example)

```
import SpriteKit
import GameplayKit
class GameScene: SKScene {
    var bird2 = SKSpriteNode()
    override func didMove(to view:SKView){
        let texture = SKTexture(imageNamed: "bird")
        bird2 = SKSpriteNode(texture: texture)
        bird2.position = CGPoint(x:0, y:0) //relative to the anchor point in GameScene.sks
        bird2.size = CGSize(width: self.frame.width/16, height: self.frame.height/10)
        bird2.zPosition = 1
        self.addChild(bird2)
    }
}
```

//GameScene.swift

```
import SpriteKit
import GameplayKit
class GameScene: SKScene, SKPhysicsContactDelegate {
    var bird = SKSpriteNode()
    var box1 = SKSpriteNode()
    ↓ similar ones
    var gameStarted = true
    var originalPosition: CGPoint?

    var score = 0
    var scoreLabel = SKLabelNode()


    enum ColliderType: UInt32 {  //1, 2, 4, 8, 16 ->
        case Bird = 1
        case Box = 2
        // case Ground = 4
        // case Tree = 8
    }
    override func didMove(to view:SKView){
        self.physicsBody = SKPhysicsBody(edgeLoopFrom: frame)
        self.scene?.scaleMode = .aspectFit
        self.physicsWorld.contactDelegate = self // in order to detect the collisions

        bird = childNode(withName: "bird") as! SKSpriteNode
        // same name as in GameScene.sks          // return SKNode -> force cast to SKSpriteNode
        let birdTexture = SKTexture(imageNamed: "bird")  // same name as in Assets.xcassets
```

```
bird.physicsBody = SKPhysicsBody(circleOfRadius: birdTexture.size().height/10)
bird.physicsBody?.affectedByGravity = false
bird.physicsBody?.isDynamic = true
bird.physicsBody?.mass = 0.15 // 0.15 kg
originalPosition = bird.position

bird.physicsBody?.contactTestBitMask = ColliderType.Bird.rawValue
bird.physicsBody?.categoryBitMask = ColliderType.Bird.rawValue
bird.physicsBody?.collisionBitMask = ColliderType.Box.rawValue


let boxTexture = SKTexture(imageNamed: "brick")
let size = CGSize(width: boxTexture.size().width/6, height: boxTexture.size().height/6)

box1 = childNode(withName: "box1") as! SKSpriteNode
box1.physicsBody = SKPhysicsBody(rectangleOf: size)
box1.physicsBody = SKPhysicsBody(rectangleOf: size)
box1.physicsBody?.isDynamic = true
box1.physicsBody?.affectedByGravity = true
box1.physicsBody?.allowsRotation = true
box1.physicsBody?.mass = 0.4

box1.physicsBody?.collisionBitMask = ColliderType.Bird.rawValue
↓ similar ones

//Label
scoreLabel.fontName = "Helvetica"
scoreLabel.size = 60
scoreLabel.text = "0"
scoreLabel.position = CGPoint(x: 0, y: self.frame.height/4)
scoreLabel.zPosition = 2
self.addChild(scoreLabel)

}
func didBegin(_ contact: SKPhysicsContact){      // available with SKPhysicsContactDelegate
    if contact.bodyA.collisionBitMask == ColliderType.Bird.rawValue || contact.bodyB.collisionBitMask ==
        ColliderType.Bird.rawValue{
            score += 1
            scoreLabel.text = String(score)
    }
}



override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?){
    /*
    bird.physicsBody?.applyImpulse(CGVector(dx: 0, dy: 100))
    bird.physicsBody?.affectedByGravity = true
    */

    if gameStarted == false{
        if let touch = touches.first{                    → // can be scene or node
```

```
            let touchLocation = touch.location(in: self)
            let touchNodes = nodes(at: touchLocation)

            if touchNodes.isEmpty == false{
                for node in touchNodes{ //sknode, not skSpriteNode
                    if let sprite = node as? SKSpriteNode{
                        if sprite == bird{
                            bird.position = touchLocation
                        }
                    }
                }
            }
        }
    }
}

override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?){
    if gameStarted == false{
        if let touch = touches.first{ // can be scene or node
            let touchLocation = touch.location(in: self)
            let touchNodes = nodes(at: touchLocation)

            if touchNodes.isEmpty == false{
                for node in touchNodes{ //sknode, not skSpriteNode
                    if let sprite = node as? SKSpriteNode{
                        if sprite == bird{
                            bird.position = touchLocation
                        }
                    }
                }
            }
        }
    }
}

override func touchesEnded(_ touches: Set<UITouch> with event: UIEvent?){
    if gameStarted == false{
        if let touch = touches.first{ // can be scene or node
            let touchLocation = touch.location(in: self)
            let touchNodes = nodes(at: touchLocation)

            if touchNodes.isEmpty == false{
                for node in touchNodes{ //sknode, not skSpriteNode
                    if let sprite = node as? SKSpriteNode{
                        if sprite == bird{
                            let dx = -(touchLocation.x - originalPosition!.x)
                            let dy = -(touchLocation.y - originalPosition!.y)

                            let impulse = CGVector(dx:dx, dy:dy)
                            bird.physicsBody?.applyImpulse(impulse)
                            bird.physicsBody?.affectedByGravity = true
                            gameStarted = true
                        }
                    }
                }
            }
        }
    }
}
```

```
override func update(_ currentTime: TimeInterval){      //called before each frame is rendered
    if let birdPhysicsBody = bird.physicsBody{
        if birdPhysicsBody.velocity.dx <= 0.1 && birdPhysicsBody.velocity.dy <= 0.1 &&
            birdPhysicsBody.angularVelocity <= 0.1 && gameStarted == true {

            bird.physicsBody?.affectedByGravity = false
            bird.physicsBody?.velocity = CGVector(dx: 0, dy: 0)
            bird.physicsBody?.angularVelocity = 0
            bird.zPosition = 1
            bird.position = originalPosition!
            score = 0
            scoreLabel.text = String(score)
            gameStarted = false

        }
    }
}

}
```