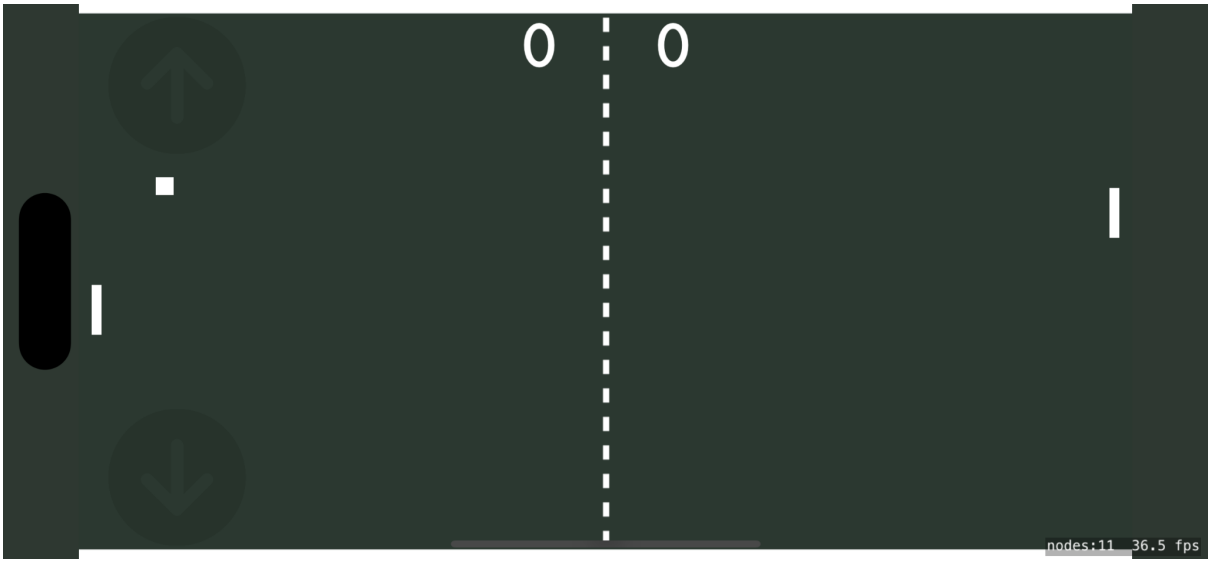
Diego Estuardo Hidalgo Romero A01730979
Ricardo Alfonso Ricárdez Tinoco A01327530
Guillermo Ramos Domínguez A01736484

Documentación de PongPlus

Descripción del juego:

El videojuego desarrollado consiste en una versión similar del videojuego Pong. El juego consiste en una pelota que se mueve a través de la pantalla con el objetivo de salir por los lados izquierdo o derecho.

El juego inicia al presionar el botón de “Comenzar”, lo que creará una pelota en el centro de la pantalla, y acelerará cada que se detecte un rebote con el jugador o el oponente.
Un jugador y un oponente controlan barras capaces de provocar un rebote sobre la pelota, con el objetivo de defender su lado del área de juego.



Cuando la pelota sale de pantalla se atribuye una puntuación, el jugador recibe un punto a favor si el oponente no consigue evitar que la pelota salga del lado derecho. El oponente recibe un punto si el jugador no consigue evitar que la pelota salga del lado izquierdo.

Se finalizará la partida cuando uno de los marcadores haya llegado al objetivo de 3 puntos, lo que detendrá las acciones del jugador, y mostrará un mensaje de victoria o derrota según correspondan los marcadores.

Variables:

1. **scorePlayer**: Contador para los puntos marcados por el usuario.
2. **scoreOponent**: Contador para los puntos marcados por el oponente.
3. **player**: El objeto SKSpriteNode de la barra controlada por el usuario.
4. **opponent:** El objeto SKSpriteNode de la barra controlada por el oponente.
5. **ball**: Un objeto SKSpriteNode usado para hacer referencia a la pelota generada en cada ronda de la partida.
6. **scorePlayerLabel**: La etiqueta que muestra los puntos del usuario.
7. **scoreOpponentLabel**: La etiqueta que muestra los puntos de oponente.
8. **gameStart**: La etiqueta que sirve como botón para iniciar una partida.
9. **gameResult**: La etiqueta que muestra el resultado de la puntuación en una partida.
10. **buttonUp**: El objeto SKSpriteNode que sirve como botón para permitir al usuario mover la barra hacia arriba.
11. **buttonDown**: El objeto SKSpriteNode que sirve como botón para permitir al usuario mover la barra

```
var scorePlayer = 0
var scoreOponent = 0
var player: SKSpriteNode!
var opponent: SKSpriteNode!
var ball: SKSpriteNode!
var scorePlayerLabel: SKLabelNode!
var scoreOpponentLabel: SKLabelNode!
var gameStart : SKLabelNode!
var gameResult : SKLabelNode!
var buttonUp : SKSpriteNode!
var buttonDown : SKSpriteNode!
var fondo: SKSpriteNode!
var borderPadding: CGFloat = 11.0;

//control de propiedades de pelota
var ballVelocityX: CGFloat = 0.0;
var ballVelocityY: CGFloat = 0.0;
var ballVelocityIncrease: CGFloat = 0.8;

//control para botones de movimiento
var movePlayerUp = false;
var movePlayerDown = false;

//control propiedades de barras
var playerVelocity: CGFloat = 8.0;
var opponentVelocity: CGFloat = 7.0;
var opponentReactionTime: CGFloat = 1 / 60;
```

hacia abajo.
12. **fondo**: El objeto SKSpriteNode de la cancha que define el límite de movimiento de la pelota.
13. **borderPadding**: Cantidad de pixeles para los bordes superior e inferior del campo de juego.
14. **ballVelocityX**: La velocidad inicial de la bola en x.
15. **ballVelocityY**: La velocidad inicial de la bola en y.
16. **ballVelocityIncrease**: La cantidad que incrementa la velocidad de la bola, en función del tiempo.
17. **movePlayerUp**: Bandera lógica para movilizar la barra del usuario hacia arriba.
18. **movePlayerDown**: Bandera lógica para movilizar la barra del usuario hacia abajo.
19. **playerVelocity**: La velocidad con la que se traslada la barra del usuario.
20. **opponentVelocity**: La velocidad con la que se traslada la barra del oponente.
21 **opponentReactionTime**: Variable usada inicialmente para dar un tiempo de espera entre movimientos del personaje

Funciones:

func didMove():

Se sobreescribe la función didMove() para inicializar todas las variables de SKSpriteNode y SKLabelNode.

Se asignan atributos del motor de físicas para detectar colisiones de la barra del jugador y la barra del oponente.

```
override func didMove(to view: SKView) {
    backgroundColor = SKColor(red: 46.0 / 255.0, green: 56.0 / 255.0, blue

    //asigna propiedades a barra del jugador
    player = (self.childNode(withName: "//player") as! SKSpriteNode)
    player.zPosition = 0
    player.physicsBody = SKPhysicsBody(rectangleOf: player.size)
    player.physicsBody?.isDynamic = true
    player.physicsBody?.categoryBitMask = PhysicsCategory.bar
    player.physicsBody?.contactTestBitMask = PhysicsCategory.ball
    player.physicsBody?.collisionBitMask = PhysicsCategory.none
    player.physicsBody?.usesPreciseCollisionDetection = true

    //asigna propiedades a barra del oponente
    opponent = (self.childNode(withName: "//opponent") as! SKSpriteNode)
    opponent.zPosition = 0
    opponent.physicsBody = SKPhysicsBody(rectangleOf: opponent.size)
    opponent.physicsBody?.isDynamic = true
    opponent.physicsBody?.categoryBitMask = PhysicsCategory.bar
    opponent.physicsBody?.contactTestBitMask = PhysicsCategory.ball
    opponent.physicsBody?.collisionBitMask = PhysicsCategory.none
    opponent.physicsBody?.usesPreciseCollisionDetection = true

    //asigna objetos de labels
    scorePlayerLabel = (self.childNode(withName: "//scorePlayer") as! SKLa
    scoreOpponentLabel = (self.childNode(withName: "//scoreOpponent") as!

    //asigna objetos de botones
    gameResult = (self.childNode(withName: "//resultado") as! SKLabelNode)
    gameStart = (self.childNode(withName: "//gameStart") as! SKLabelNode)
    buttonUp = (self.childNode(withName: "//buttonUp") as! SKSpriteNode)
    buttonDown = (self.childNode(withName: "//buttonDown") as! SKSpriteNod
```

func launchBall():

Función que genera una nueva pelota.

La función asigna la posición inicial de la pelota al centro de la escena.

Se asigna un vector de movimiento inicial mediante la descomposición del vector a velocidades iniciales en los eje X y Y.

Se asignan propiedades del motor de físicas para detectar colisiones con las barras del jugador y oponente.

Se crea una instancia de la pelota con las propiedades definidas.

```
func launchBall(fromPlayer: Bool) {
    if (gameResult.isHidden) {
        ball = SKSpriteNode(imageNamed: "ball")
        //genera la pelota al centro
        ball.position = CGPoint(x: 0.0, y: 0.0)
        ball.zPosition = 0

        ballVelocityX = 7.0
        //decide quien lanza pelota
        if !fromPlayer {
            ballVelocityX = ballVelocityX * -1
        }
        ballVelocityY = 0.0

        //asigna fisica
        ball.physicsBody = SKPhysicsBody(circleOfRadius: ball.size.width / 2)
        ball.physicsBody?.isDynamic = true
        ball.physicsBody?.categoryBitMask = PhysicsCategory.ball
        ball.physicsBody?.contactTestBitMask = PhysicsCategory.bar
        ball.physicsBody?.collisionBitMask = PhysicsCategory.none
        ball.physicsBody?.usesPreciseCollisionDetection = true

        //instancia
        addChild(ball)
    }
}
```

func bounceBall():

Esta función revisa si la bola está tocando los bordes del campo y si toca el borde, dependiendo de si es el borde superior o inferior, invierta la dirección de la velocidad en y.

```
func bounceBall() {
    if ball != nil {
        //supero el limite de a
        if ball.position.y > size.height / 2 - ball.size.height / 2 - borderPadding {
            ballVelocityY = ballVelocityY * -1
            //sonido al rebotar
            run(SKAction.playSoundFileNamed("ballSFX.mp3", waitForCompletion: false))
        }
        //paso limite abajo
        if ball.position.y < -size.height / 2 + ball.size.height / 2 + borderPadding {
            ballVelocityY = ballVelocityY * -1
            //sonido al rebotar
            run(SKAction.playSoundFileNamed("ballSFX.mp3", waitForCompletion: false))
        }
    }
}
```

func catchBall():

Esta función le da el comportamiento a la barra del oponente, la barra sigue la bola, revisando si la posición de la bola en relación a los bordes de la barra y dependiendo de si está arriba o abajo le suma opponentVelocity a su posición actual.

```
func catchBall() {

    //posición inical del Sprite oponente
    var newPositionY = opponent.position.y

    //pelota arriba del oponente
    if ball.position.y > opponent.position.y + opponent.size.height / 2 {
        newPositionY += opponentVelocity
    }
    //pelota abajo del oponente
    if ball.position.y < opponent.position.y - opponent.size.height / 2 {
        newPositionY -= opponentVelocity
    }

    //actualiza posiciòn del Sprite oponente
    opponent.position.y = newPositionY
}
```

func ballColliderWithBar():

Esta función acelera la velocidad en el vector X de la pelota e invierte dicha velocidad para provocar un rebote.

También genera una velocidad en Y que dependerá de la distancia de la pelota con el centro de la barra, asignando una velocidad mayor si la colisión se detecta cerca de una esquina.

```
func ballCollidedWithBar(bar: SKSpriteNode, ball: SKSpriteNode) {
    //aumenta velocidad de pelota en x
    ballVelocityX = ballVelocityX + (ballVelocityX > 0 ? ballVelocityIncrease : -ballVelocityIncrease)
    //invierte la velocidad en x (rebote)
    ballVelocityX = ballVelocityX * -1

    //desvía pelota en Y si rebota lejos del centro de la barra
    ballVelocityY = 10.0 * (ball.position.y - bar.position.y) / (bar.size.height / 2)

    //sonido al colisionar
    run(SKAction.playSoundFileNamed("ballSFX.mp3", waitForCompletion: false))
}
```

func didBegin():

función llamada cuando el motor de físicas detecta una colisión. Valida que la colisión fué entre el objeto pelota y una de las barras. De ser así, envía los objetos que colisionaron a la función ballCollideWithBar()

```
func didBegin(_ contact: SKPhysicsContact) {
    if contact.bodyA.categoryBitMask == PhysicsCategory.bar && contact.bodyB.categoryBitMask == PhysicsCategory.ball {
        ballCollidedWithBar(bar: contact.bodyA.node as! SKSpriteNode, ball: contact.bodyB.node as! SKSpriteNode)
    }
}
```

func startGame():

Esta función sirve para reiniciar los valores de los contadores, las etiquetas de los contadores, regresa las barras del usuario y el oponente a sus posiciones iniciales, y oculta las etiquetas para comenzar la partida y mostrar el resultado de la partida.

```
func startGame() {
    //reinicia contadores
    scorePlayer = 0
    scoreOponent = 0
    scorePlayerLabel.text = String(scorePlayer)
    scoreOpponentLabel.text = String(scoreOponent)

    //reinicia posicion de barras
    player.position.y = 0.0
    opponent.position.y = 0.0

    //oculta boton "Comenzar"
    gameStart.isHidden = true
    gameResult.isHidden = true

    //lanzamiento inicial hacia el jugador
    launchBall(fromPlayer: false)
}
```

func gameOver():

Esta función es llamada una vez que la condición de victoria o derrota es satisfecha.
Revisa quien ganó la partida, en el caso de que se cumpla, se regresas las barras a su posición inicial, y se despliega un texto que avisa quién ganó y se pone otro texto que le pregunta al usuario si quiere volver a jugar.

```
func gameOver() {
    var fin = false
    if scorePlayer == 3 {
        gameResult.text = "Ganaste !!"
        fin = true
        run(SKAction.playSoundFileNamed("winSFX.mp3", waitForCompletion: false))
    } else if scoreOponent == 3 {
        gameResult.text = "Perdiste :["
        fin = true
        run(SKAction.playSoundFileNamed("loseSFX.mp3", waitForCompletion: false))
    }
    if fin {
        //reinicia posicion de barras
        player.position.y = 0.0
        opponent.position.y = 0.0

        //muestra resultado y botón "Comenzar"
        gameStart.text = "Revancha"
        gameStart.isHidden = false
        gameResult.isHidden = false
    }
}
```

func touchesBegan():

Se sobreescribe esta función para detectar cuando el usuario ha presionado la pantalla.

Si se ha presionado sobre el botón de “Comenzar”, ejecuta la función startGame().

Si se ha presionado un botón de movimiento, se habilita la variable que permitirá desplazar la barra en la dirección indicada.

```
override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
    let touch = touches.first!
    //se presiono el boton "Comenzar"
    if !gameStart.isHidden && gameStart.contains(touch.location(in: self)) {
        startGame()
    }
    //la partida está en curso
    if gameStart.isHidden {
        if buttonUp.contains(touch.location(in: self)) {
            movePlayerUp = true
        }
        if buttonDown.contains(touch.location(in: self)) {
            movePlayerDown = true
        }
    }
}
```

func touchesEnded():

Se sobreescribe esta función para detectar cuando el usuario ha dejado de presionar la pantalla.

Si se ha dejado de presionar un botón de movimiento, se desactiva la variable que permitía desplazar la barra en la dirección deseada.

```
override func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?) {
    let touch = touches.first!
    //la partida está en curso
    if gameStart.isHidden {
        if buttonUp.contains(touch.location(in: self)) {
            movePlayerUp = false
        }
        if buttonDown.contains(touch.location(in: self)) {
            movePlayerDown = false
        }
    }
}
```

func update():

Se sobreescribe la funcion update() para otorgar la funcionalidad que cambiará conforme a la velocidad de actualización de fotogramas.

Si hay una partida en curso, primero se realizará el movimiento del jugador que se encuentre habilitado en las variables de control.

A continuación, se llama a la función catchBall() para determinar si es necesario mover al oponente.

Posteriormente se detecta si la pelota está fuera del campo de juego, y de ser así se le asigna un punto al marcador correspondiente.

Después de asignar el punto, se llama a la función gameOver() para validar si la partida debe continuar o se ha cumplido la condición de victoria o derrota.

Finalmente, si la pelota sigue dentro del campo, se aplica un cambio de posición mediante sus velocidades en los vectores X y Y.

```swift
override func update(_ currentTime: TimeInterval) {
    //solo ejecuta codigo si la partida ha comenzado
    if gameStart.isHidden {
        //valida si el usuario esta manteniendo el boton de mover arriba
        if movePlayerUp {
            if player.position.y < size.height / 2 - player.size.height / 2 - borderPadding {
                player.position.y += playerVelocity
            }
        }
        //valida si el usuario esta manteniendo el boton de mover abajo
        if movePlayerDown {
            if player.position.y > -size.height / 2 + player.size.height / 2 + borderPadding {
                player.position.y -= playerVelocity
            }
        }

        //valida que hay una pelota instanciada
        if ball != nil {

            //mueve al oponente para atrapar la pelota
            catchBall()

            //detecta si la pelota ha salido de pantalla
            if ball.position.x < -fondo.size.width / 2 - ball.size.width / 2 {
                //salio del lado del jugador, punto para oponente
                ball.removeFromParent()
                scoreOponent += 1
                scoreOpponentLabel.text = String(scoreOponent)
                run(SKAction.playSoundFileNamed("scoredSFX.mp3", waitForCompletion: false))
                gameOver()
                launchBall(fromPlayer: true)
            } else if ball.position.x > fondo.size.width / 2 + ball.size.width / 2 {
                //salio del lado del oponente, punto para jugador
                ball.removeFromParent()
                scorePlayer += 1
                scorePlayerLabel.text = String(scorePlayer)
                run(SKAction.playSoundFileNamed("scoredSFX.mp3", waitForCompletion: false))
                gameOver()
                launchBall(fromPlayer: false)
            } else {
                //no ha salido, sigue moviendo
                ball.position.x += ballVelocityX
                ball.position.y += ballVelocityY

                bounceBall()
```

Enlace del repositorio:
https://github.com/ritirial/Pong.git

Enlace del video de funcionamiento:
https://drive.google.com/file/d/10tRy_bRHDOodKPqQSdb4xh0G_7Su0LiM/view?usp=sharing