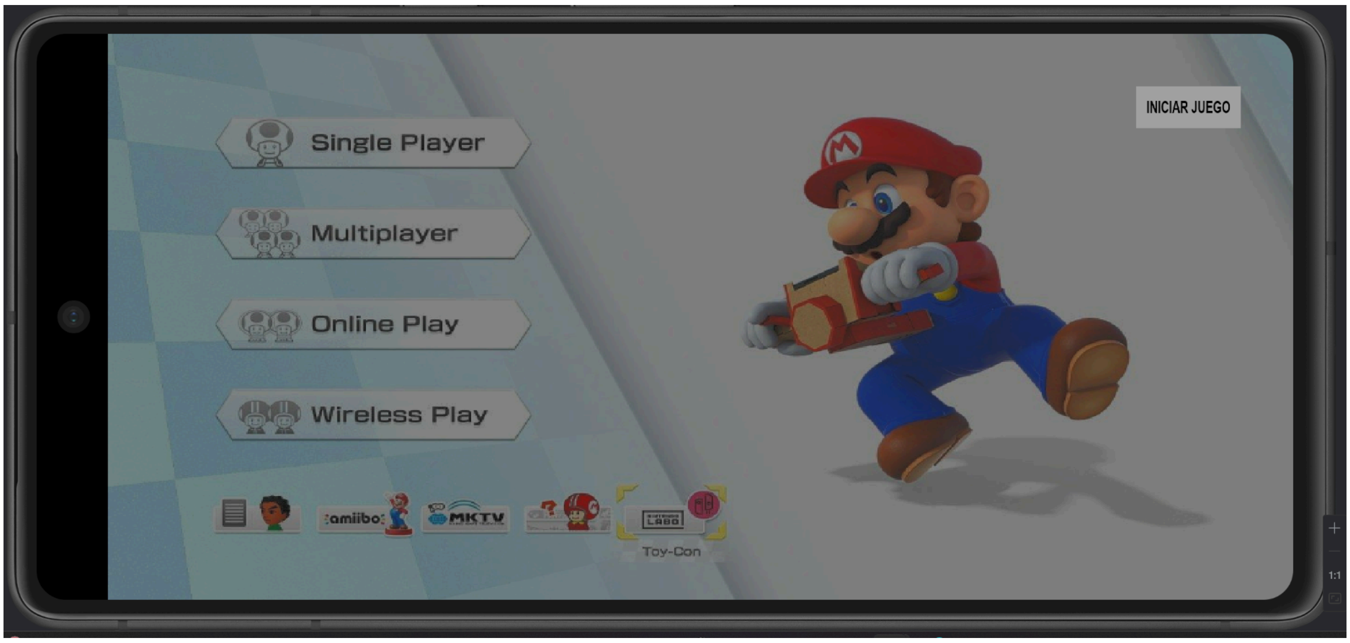
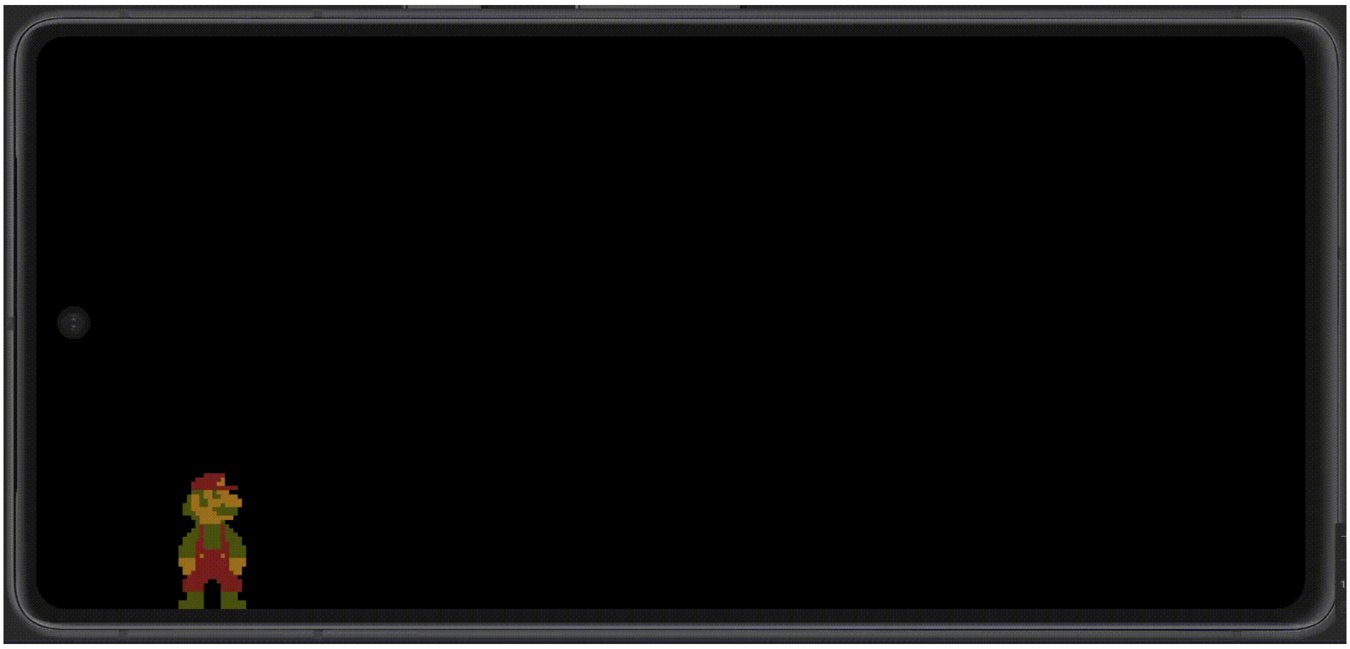
Tarea individual 11 - Menú y Salto con gravedad

Inicia el juego con un botón arriba a la derecha, ‘INICIAR JUEGO’:



Gif con salto y gravedad:



Hecho por Jorge Varela Zamora, de 2º del C.F.G.S. de D.A.M, curso 2024-2025, I.E.S. Vista Alegre

Captura de vídeo con la app en ejecución:

Códigos en Kotlin:

```
package io.github.mario

import MainGame
import com.badlogic.gdx.Gdx
import com.badlogic.gdx.Input
import com.badlogic.gdx.Screen
import com.badlogic.gdx.graphics.GL20
import com.badlogic.gdx.graphics.g2d.*
import com.badlogic.gdx.utils.ScreenUtils
import com.badlogic.gdx.utils.Array

class GameScreen(private val game: Main) : Screen { // Modifica el
constructor para recibir 'Main'
   private lateinit var batch: SpriteBatch
   private lateinit var textureAtlas: TextureAtlas
   private lateinit var idleAnimation: Animation<TextureRegion>
   private lateinit var leftAnimation: Animation<TextureRegion>
   private lateinit var rightAnimation: Animation<TextureRegion>
   private lateinit var upAnimation: Animation<TextureRegion>
   private lateinit var downAnimation: Animation<TextureRegion>
   private var currentAnimation: Animation<TextureRegion>? = null
   private var stateTime = 0f
   private var x = 0f
   private var y = 0f
   private val speed = 100f

   private var isJumping: Boolean = false
   private val gravity: Float = -500f
   private var velocityY: Float = 0f
   private val JUMP_VELOCITY: Float = 400f   // Altura salto
   private val GROUND_Y: Float = 0f

   override fun show() {
       batch = SpriteBatch()
       textureAtlas =
TextureAtlas(Gdx.files.internal("Mario_and_Enemies.pack"))
       loadAnimations()
       currentAnimation = idleAnimation // Inicializar con animación de
"parado"
```

```
    }

    private fun loadAnimations() {
        val region = textureAtlas.findRegion("big_mario")

        val totalFrames = 20
        val frameWidth = region.regionWidth / totalFrames
        val frameHeight = region.regionHeight

        val idleFrame = TextureRegion(region, 0, 0, frameWidth,
frameHeight)
        idleAnimation = Animation(0.1f, idleFrame)

        val downFrame = TextureRegion(region, 96, 0, frameWidth,
frameHeight)
        downAnimation = Animation(0.1f, downFrame)

        val upFrame = TextureRegion(region, 80, 0, frameWidth, frameHeight)
        upAnimation = Animation(0.1f, upFrame)

        leftAnimation = createAnimation("big_mario", totalFrames, 3, 0.1f)
        rightAnimation = createAnimation("big_mario", totalFrames, 3, 0.1f)

        currentAnimation = idleAnimation
    }

    private fun createAnimation(regionName: String, totalFrames: Int,
numFrames: Int, frameDuration: Float): Animation<TextureRegion> {
        val frames = Array<TextureRegion>()
        val region = textureAtlas.findRegion(regionName)
        val frameWidth = region.regionWidth / totalFrames
        val frameHeight = region.regionHeight

        for (i in 0 until numFrames) {
            frames.add(TextureRegion(region, i * frameWidth, 0, frameWidth,
frameHeight))
        }

        return Animation(frameDuration, frames, Animation.PlayMode.LOOP)
    }

    override fun render(delta: Float) {
        stateTime += Gdx.graphics.deltaTime
        handleInput(Gdx.graphics.deltaTime)
        applyGravity(Gdx.graphics.deltaTime)

        ScreenUtils.clear(0f, 0f, 0f, 1f)
        currentAnimation = when {
            Gdx.input.isKeyPressed(Input.Keys.LEFT) ||
Gdx.input.isKeyPressed(Input.Keys.A) -> leftAnimation
            Gdx.input.isKeyPressed(Input.Keys.RIGHT) ||
Gdx.input.isKeyPressed(Input.Keys.D) -> rightAnimation
            Gdx.input.isKeyPressed(Input.Keys.DOWN) ||
Gdx.input.isKeyPressed(Input.Keys.S) -> downAnimation
            Gdx.input.isKeyPressed(Input.Keys.SPACE) ||
Gdx.input.isKeyPressed(Input.Keys.W) ||
Gdx.input.isKeyPressed(Input.Keys.UP) -> upAnimation
            else -> idleAnimation
        }

        val currentFrame = currentAnimation!!.getKeyFrame(stateTime, true)

        batch.begin()

        val scale = 8f  // Dibujar a Mario con el tamaño correcto
        batch.draw(
            currentFrame,
            x, y,
            currentFrame.regionWidth * scale,
            currentFrame.regionHeight * scale
        )

        batch.end()
    }

    override fun resize(width: Int, height: Int) {
        // Actualiza el viewport si es necesario
    }

    override fun pause() {
        // Lógica para cuando la aplicación se pausa
    }

    override fun resume() {
        // Lógica para cuando la aplicación se reanuda
    }

    override fun hide() {
        // Lógica para cuando la pantalla se oculta
    }

    private fun handleInput(delta: Float) {
        val movementSpeed = speed * delta * 4
        if (Gdx.input.isKeyPressed(Input.Keys.LEFT) ||
Gdx.input.isKeyPressed(Input.Keys.A)) {
            x -= movementSpeed
        }
        if (Gdx.input.isKeyPressed(Input.Keys.RIGHT) ||
Gdx.input.isKeyPressed(Input.Keys.D)) {
            x += movementSpeed
        }
        if ((Gdx.input.isKeyPressed(Input.Keys.UP) ||
Gdx.input.isKeyPressed(Input.Keys.W)) && !isJumping) {
            isJumping = true
            velocityY = JUMP_VELOCITY
        }

        if (Gdx.input.isKeyPressed(Input.Keys.DOWN) ||
Gdx.input.isKeyPressed(Input.Keys.S)) {
            y -= movementSpeed
        }

        if(Gdx.input.isKeyPressed(Input.Keys.M)) {
            game.screen = MainGame(game)            //sI PRESIONAS m SALE
AL MENU
        }

    }

    override fun dispose() {
        batch.dispose()
        textureAtlas.dispose()
    }

    private fun applyGravity(delta: Float) {
        if (isJumping) {
            velocityY += gravity * delta
        }

        y += velocityY * delta

        if (y <= GROUND_Y) {
            y = GROUND_Y
            velocityY = 0f
            isJumping = false
        }
    }
}
```

```
package io.github.mario
```

```
import MainGame
import com.badlogic.gdx.ApplicationAdapter
import com.badlogic.gdx.Game
import com.badlogic.gdx.Gdx
import com.badlogic.gdx.Input
import com.badlogic.gdx.graphics.g2d.*
import com.badlogic.gdx.utils.ScreenUtils
import com.badlogic.gdx.utils.Array

class Main : Game() {
   private lateinit var batch: SpriteBatch
   private lateinit var textureAtlas: TextureAtlas
   private lateinit var idleAnimation: Animation<TextureRegion>
   private lateinit var leftAnimation: Animation<TextureRegion>
   private lateinit var rightAnimation: Animation<TextureRegion>
   private lateinit var upAnimation: Animation<TextureRegion>
   private lateinit var downAnimation: Animation<TextureRegion>
   private var currentAnimation: Animation<TextureRegion>? = null
   private var stateTime = 0f
   private var x = 0f
   private var y = 0f
   private val speed = 100f

   override fun create() {
       setScreen(MainGame(this))
   }

   override fun render() {
      super.render()
   }

   override fun dispose() {
       screen?.dispose()
   }


}
```

```
import com.badlogic.gdx.Gdx
import com.badlogic.gdx.Screen
import com.badlogic.gdx.graphics.GL20
import com.badlogic.gdx.graphics.Texture
import com.badlogic.gdx.graphics.g2d.SpriteBatch
import com.badlogic.gdx.scenes.scene2d.Stage
import com.badlogic.gdx.utils.viewport.ScreenViewport
import io.github.mario.GameScreen
```

```kotlin
import io.github.mario.Main
import io.github.mario.MenuButton


class MainGame(private val game: Main) : Screen {

   private lateinit var batch: SpriteBatch
   private lateinit var stage: Stage
   private lateinit var menuTexture: Texture
   private lateinit var btnTexture: Texture

   override fun show() {
       batch = SpriteBatch()
       menuTexture = Texture(Gdx.files.internal("menu.png"))
       btnTexture = Texture(Gdx.files.internal("btnStartGame.png"))

       stage = Stage(ScreenViewport())
       Gdx.input.inputProcessor = stage

       val buttonWidth = 200f
       val buttonHeight = 80f
       val buttonX = (Gdx.graphics.width - buttonWidth) -200f / 2f
       val buttonY = (Gdx.graphics.height - buttonHeight) - 200f / 2f

       val button = MenuButton(buttonX, buttonY, buttonWidth,
buttonHeight, btnTexture) {
           game.screen = GameScreen(game) // Pasa la instancia de 'Main'
       }

       stage.addActor(button)
   }

   override fun render(delta: Float) {
       Gdx.gl.glClearColor(1f, 1f, 1f, 1f)
       Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT)

       batch.begin()
       batch.draw(menuTexture,
           0f, 0f,
           Gdx.graphics.width.toFloat(), Gdx.graphics.height.toFloat(),
           0, 0,
           menuTexture.width, menuTexture.height,
           false, false)
       batch.end()

       stage.act(delta)
       stage.draw()
   }
```

```
    override fun resize(width: Int, height: Int) {
        stage.viewport.update(width, height, true)
    }

    override fun pause() {}
    override fun resume() {}
    override fun hide() {}

    override fun dispose() {
        batch.dispose()
        menuTexture.dispose()
        btnTexture.dispose()
        stage.dispose()
    }
}
```

```
package io.github.mario

import com.badlogic.gdx.Gdx
import com.badlogic.gdx.graphics.Texture
import com.badlogic.gdx.graphics.g2d.Batch
import com.badlogic.gdx.scenes.scene2d.Actor
import com.badlogic.gdx.scenes.scene2d.InputEvent
import com.badlogic.gdx.scenes.scene2d.InputListener

class MenuButton(x: Float, y: Float, width: Float, height: Float, texture:
Texture, val onClick: () -> Unit) : Actor() {

    init {
        setBounds(x, y, width, height)
        addListener(object : InputListener() {
            override fun touchDown(event: InputEvent?, x: Float, y: Float,
pointer: Int, button: Int): Boolean {
                try {
                    onClick()
                } catch (e: Exception) {
                    Gdx.app.error("MenuButton", "Error al hacer clic en el
botón", e)
                }
                return true
            }
        })
    }

    private val buttonTexture = texture
    override fun draw(batch: Batch, parentAlpha: Float) {
        batch.draw(buttonTexture, x, y, width, height)
    }
}
```