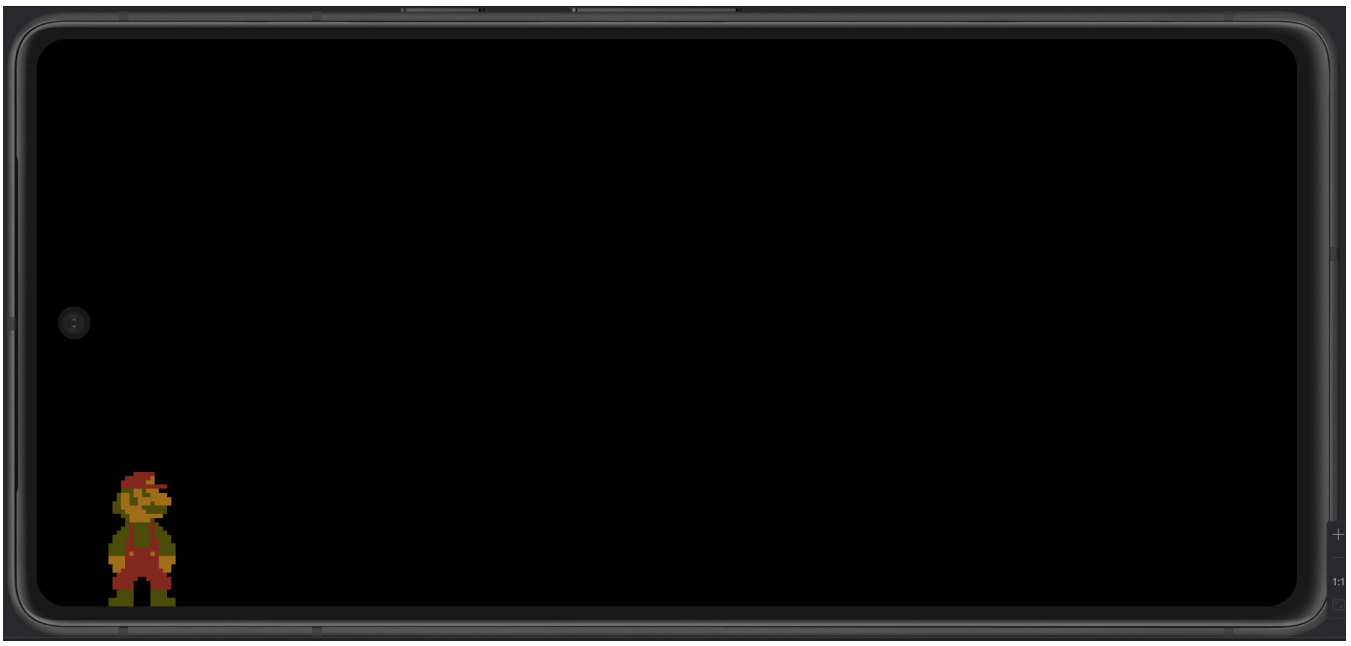
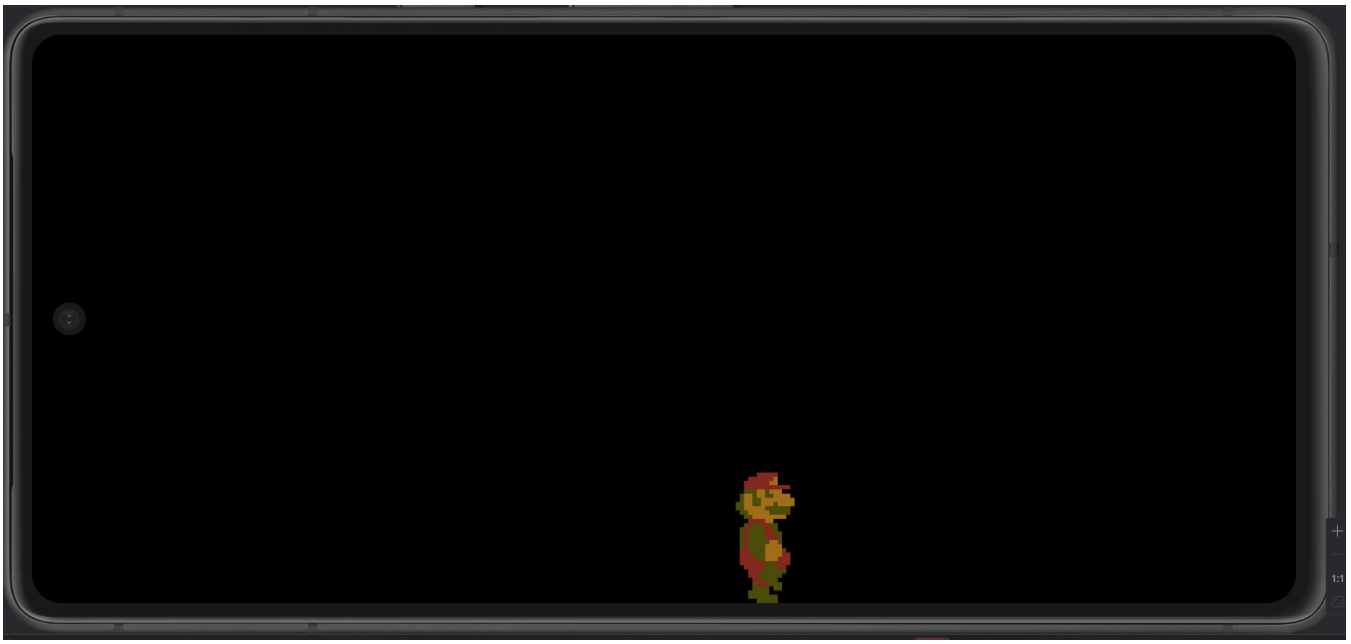
Tarea individual 10 - Animacion 2D Mario

Uso Animation ,AtlasRegion y TextureRegion para crear animaciones diferentes.
Sin dar a ningún botón:

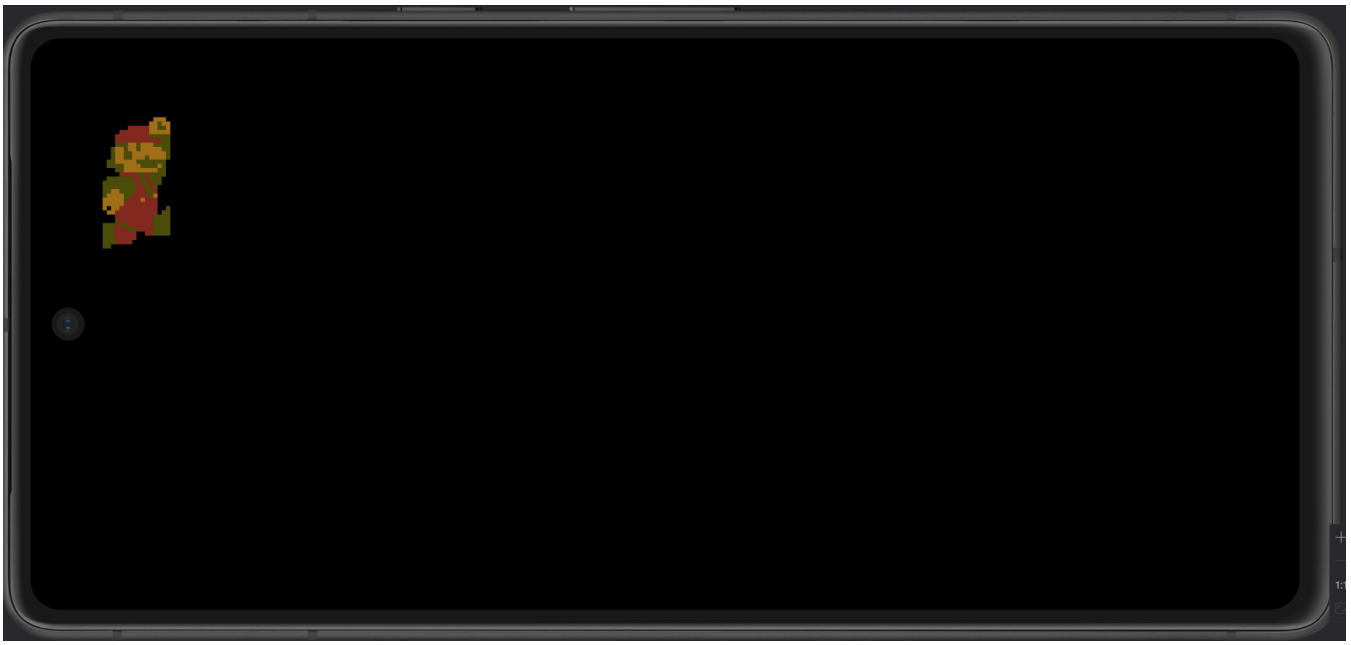


Hacia la derecha:

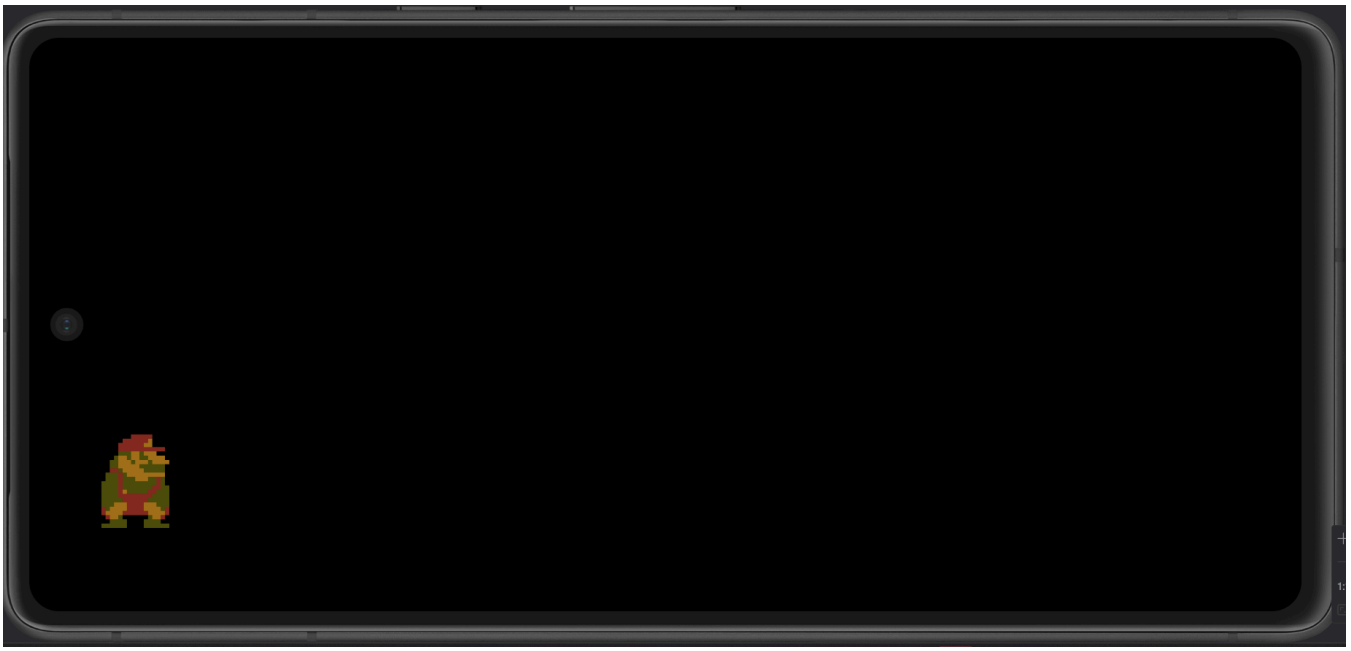


Hecho por Jorge Varela Zamora, de 2º del C.F.G.S. de D.A.M, curso 2024-2025, I.E.S. Vista Alegre

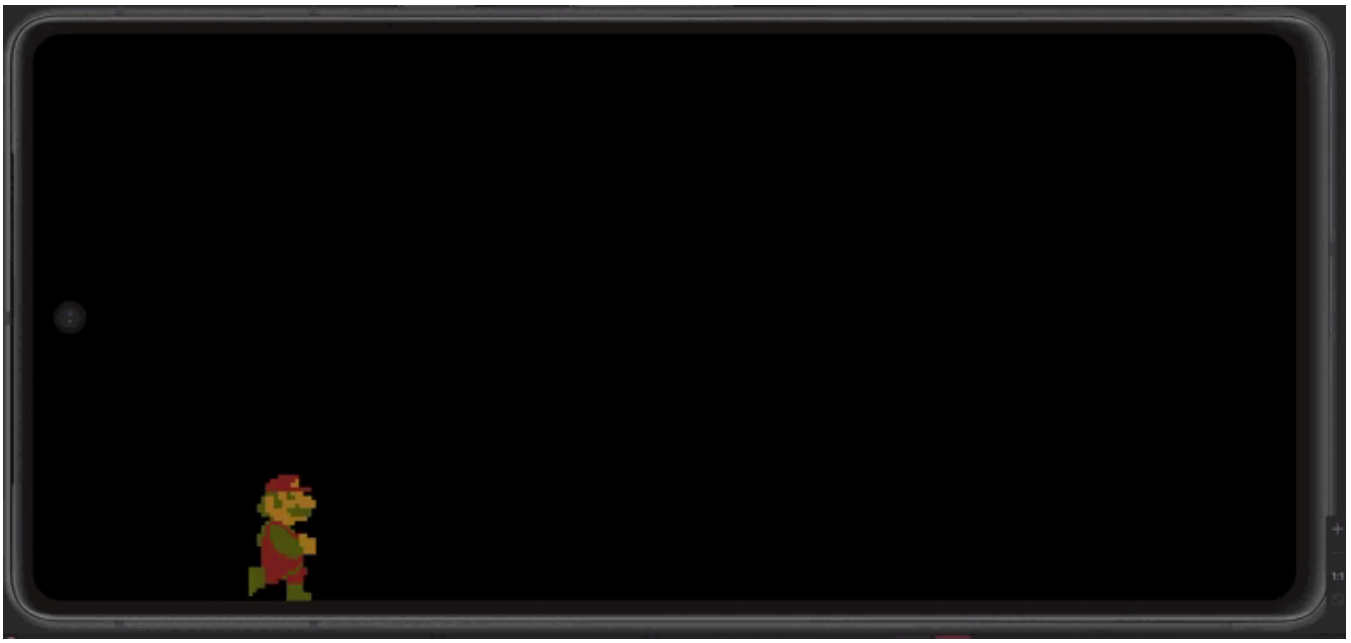
Hacia arriba:



Hacia abajo:



Captura de vídeo con la app en ejecución:



Código en **Kotlin**:

```
package io.github.mario

import com.badlogic.gdx.ApplicationAdapter
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 : ApplicationAdapter() {
    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() {
        batch = SpriteBatch()
        textureAtlas =
TextureAtlas(Gdx.files.internal("Mario_and_Enemies.pack"))
```

```
        loadAnimations()
    }

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

        // Determinar el número total de frames en la imagen
        val totalFrames = 20
        val frameWidth = region.regionWidth / totalFrames
        val frameHeight = region.regionHeight

        // Extraer SOLO UN FRAME cuando esta parado:
        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)
//VAN DE 16 EN 16
        upAnimation = Animation(0.1f, upFrame)


        // Crear animaciones de movimiento
        leftAnimation = createAnimation("big_mario", totalFrames, 3, 0.1f)
        rightAnimation = createAnimation("big_mario", totalFrames, 3, 0.1f)

        // Inicializar con la animación de 'parado'
        currentAnimation = idleAnimation
    }

    /**
     * Crea una animación tomando solo una parte específica de la textura.
     *
     * @param regionName Nombre de la región en el TextureAtlas.
     * @param totalFrames Cantidad total de frames en la fila de la imagen.
     * @param numFrames Cantidad de frames a usar en la animación.
     * @param frameDuration Duración de cada frame en la animación.
     */
    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
```

```
        // Agregar los `numFrames` primeros frames a la animación
        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() {
        stateTime += Gdx.graphics.deltaTime
        handleInput(Gdx.graphics.deltaTime)

        ScreenUtils.clear(0f, 0f, 0f, 1f)

        // Determinar la animación en función del input
        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.UP) ||
Gdx.input.isKeyPressed(Input.Keys.W) -> upAnimation
            else -> idleAnimation
        }

        // Obtener el frame actual de la animación
        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()
    }

    private fun handleInput(delta: Float) {        //Cambia la posision de
Mario en la pantalla segun la tecla que se pulse
        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)) {
            y += movementSpeed
        }
        if (Gdx.input.isKeyPressed(Input.Keys.DOWN) ||
Gdx.input.isKeyPressed(Input.Keys.S)) {
            y -= movementSpeed
        }
    }

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