Tarea individual 10 - Animacion 2D Mario

Irene Jurado Castillo

Objetivo

Crear un juego en LibGDX que muestre una animación de Mario con dos imágenes cuando se mueva utilizando el teclado. Posteriormente, implementa una animación utilizando un `AtlasMap` para manejar las imágenes de forma eficiente.

Instrucciones

1. **Crea un proyecto de LibGDX.**
   Inicia el proyecto en LibGDX y configura el entorno de desarrollo necesario.
2. **Diseño de la animación con dos imágenes:**

   Inicia el juego con una animación de Mario que tenga dos imágenes representando dos estados (por ejemplo, un estado de "correr" y otro de "detenerse"). Utiliza el teclado para mover al personaje.

   - Utiliza las teclas de flecha o las teclas "A" y "D" para mover al personaje hacia izquierda, derecha, arriba y abajo.
   - Alterna entre las imágenes según el movimiento del personaje.
3. **Animación con AtlasMap:**

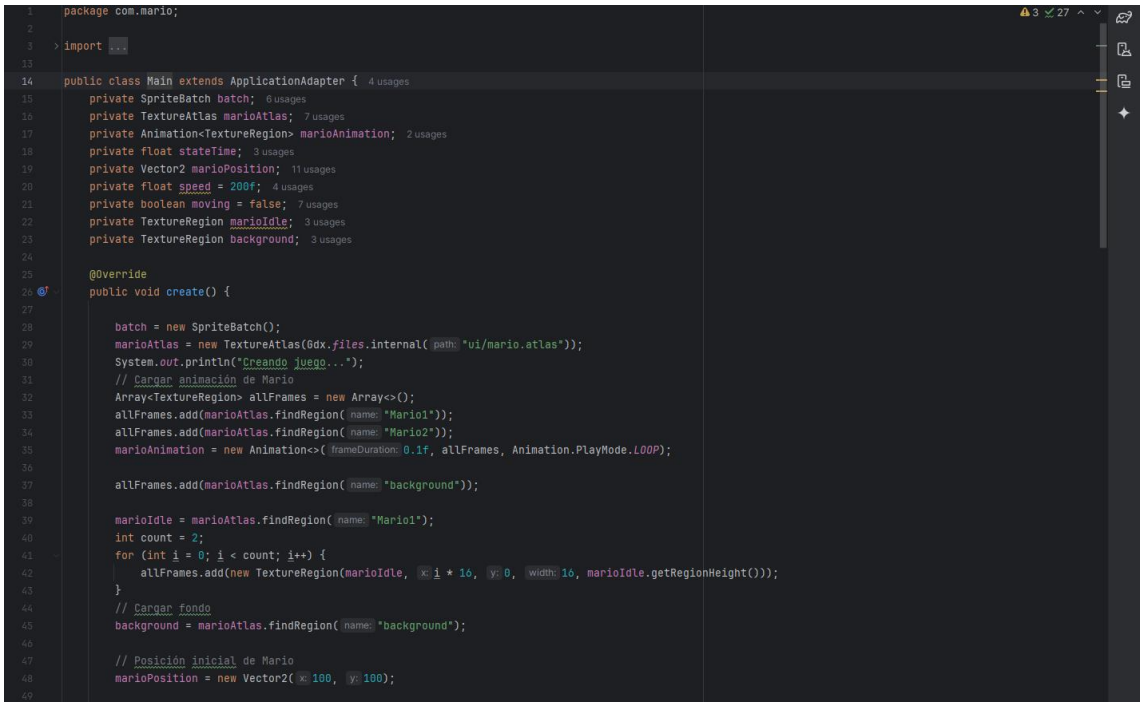
   Utiliza un `TextureAtlas` para gestionar la animación de Mario de forma más eficiente.

   - Define un archivo `atlas` con las texturas necesarias para la animación de Mario.
   - Usa `Animation ,AtlasRegion y TextureRegion` para crear la animación.

```
Array<TextureRegion> allFrames = new Array<>();
for (int i=0;i < count;i++){
    allFrames.add(new TextureRegion(bigMarioRegion, i * 16, 0, 16,
bigMarioRegion.getRegionHeight()));
// En render()
stateTime += Gdx.graphics.getDeltaTime();
TextureRegion currentFrame = bigMarioAnimation.getKeyFrame(stateTime ,
true);
```

4. Haz que la animación se reproduzca cuando Mario se mueva, alternando entre las imágenes del atlas.
5. Control del personaje: Asegúrate de que el personaje se mueva correctamente según las teclas presionadas y que la animación se reproduzca de manera fluida.

CAPTURAS DEL CODIGO USANDO .ATLAS



```java
package com.mario;

import ...

public class Main extends ApplicationAdapter {
    private SpriteBatch batch;
    private TextureAtlas marioAtlas;
    private Animation<TextureRegion> marioAnimation;
    private float stateTime;
    private Vector2 marioPosition;
    private float speed = 200f;
    private boolean moving = false;
    private TextureRegion marioIdle;
    private TextureRegion background;

    @Override
    public void create() {

        batch = new SpriteBatch();
        marioAtlas = new TextureAtlas(Gdx.files.internal("ui/mario.atlas"));
        System.out.println("Creando juego...");
        // Cargar animación de Mario
        Array<TextureRegion> allFrames = new Array<>();
        allFrames.add(marioAtlas.findRegion("Mario1"));
        allFrames.add(marioAtlas.findRegion("Mario2"));
        marioAnimation = new Animation<>(0.1f, allFrames, Animation.PlayMode.LOOP);

        allFrames.add(marioAtlas.findRegion("background"));

        marioIdle = marioAtlas.findRegion("Mario1");
        int count = 2;
        for (int i = 0; i < count; i++) {
            allFrames.add(new TextureRegion(marioIdle, i * 16, 0, 16, marioIdle.getRegionHeight()));
        }
        // Cargar fondo
        background = marioAtlas.findRegion("background");

        // Posición inicial de Mario
        marioPosition = new Vector2(100, 100);
```

```java
    // Inicializar estado de animación
    stateTime = 0f;
    moving = false;

}


@Override
public void render() {
    // LIMPIAR RESIDUOS BUFFER
    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

    handleInput(Gdx.graphics.getDeltaTime());

    // Solo incrementar el tiempo si Mario está en movimiento
    if (moving) {
        stateTime += Gdx.graphics.getDeltaTime();
    }

    TextureRegion currentFrame = marioAnimation.getKeyFrame(stateTime, looping: true);

    batch.begin();

    // Dibujar el fondo si existe
    if (background != null) {
        batch.draw(background, 0, 0);
        System.out.println("BIEN");
    }
    if (currentFrame == null) {
        System.out.println("Error: currentFrame es null");
        return;
    }
```

```
        // Dibujar a Mario con su tamaño real
        batch.draw(currentFrame, marioPosition.x, marioPosition.y);

        batch.end();
    }


    private void handleInput(float deltaTime) {  1 usage
        moving = false;

        if (Gdx.input.isKeyPressed(Input.Keys.D)) {
            marioPosition.x += speed * deltaTime;
            moving = true;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.A)) {
            marioPosition.x -= speed * deltaTime;
            moving = true;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.W)) {
            marioPosition.y += speed * deltaTime;
            moving = true;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.S)) {
            marioPosition.y -= speed * deltaTime;
            moving = true;
        }

        // Evitar que Mario salga de los límites de la pantalla
        marioPosition.x = Math.max(0, Math.min(marioPosition.x, Gdx.graphics.getWidth() - 64));
        marioPosition.y = Math.max(0, Math.min(marioPosition.y, Gdx.graphics.getHeight() - 64));

       // System.out.println("Posición de Mario: " + marioPosition.x + ", " + marioPosition.y);
    }
```

```
    @Override
    public void dispose() {
        batch.dispose();
        marioAtlas.dispose();
    }
}
```

CAPTURAS DEL JUEGO EN MOVIMIENTO

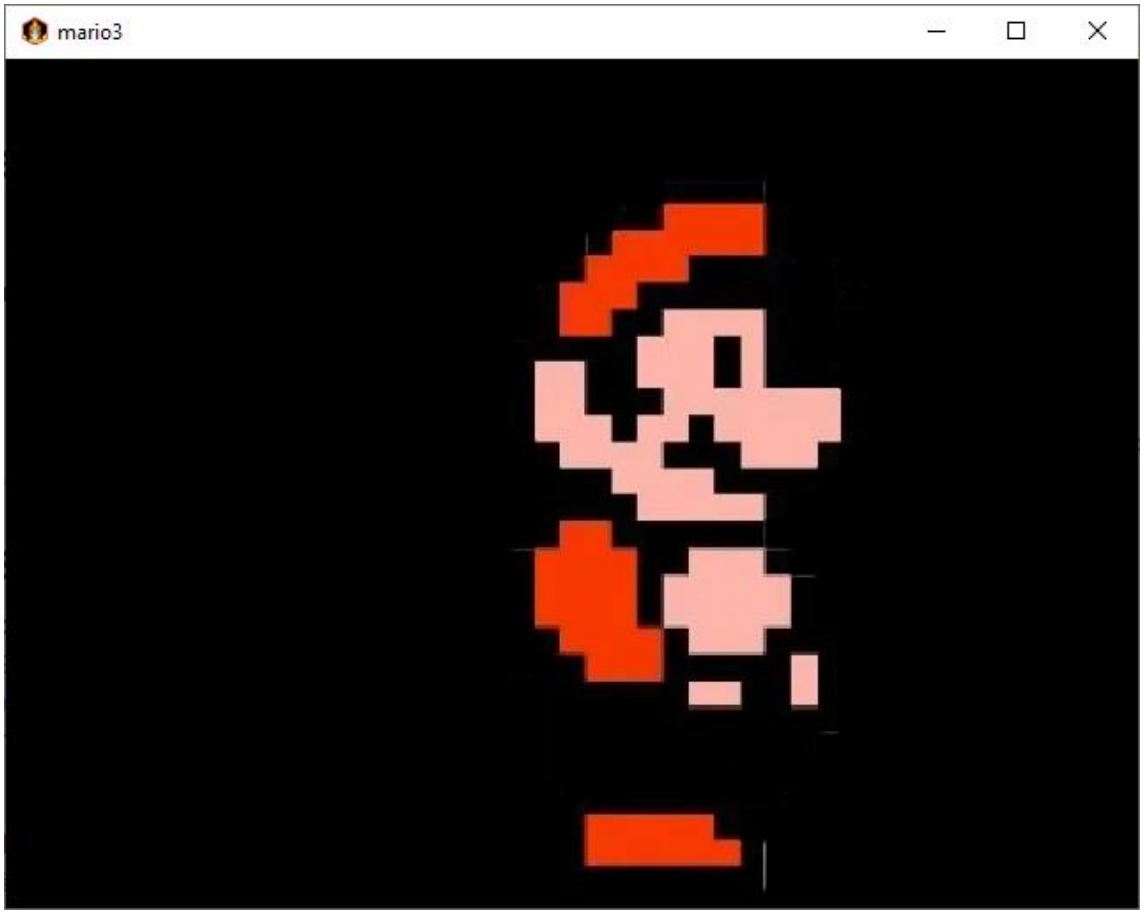
SIN ATLAS SOLO CON 2 IMÁGENES

```
package com.mario;                                                    ⚠ 3  ✓ 29

import ...

public class Main extends ApplicationAdapter {  4 usages
    private SpriteBatch batch;  5 usages
    private Texture mario1Texture, mario2Texture;  4 usages
    private Texture currentTexture;  4 usages
    private Vector2 marioPosition;  11 usages
    private float speed = 200f;  4 usages
    private boolean moving = false;  6 usages

    @Override
    public void create() {
        batch = new SpriteBatch();

        // Cargar las dos texturas de los sprites de Mario
        mario1Texture = new Texture(Gdx.files.internal( path: "ui/Mario1.png"));
        mario2Texture = new Texture(Gdx.files.internal( path: "ui/Mario2.png"));

        // Establecer el sprite inicial de Mario
        currentTexture = mario1Texture;

        // Inicializar la posición de Mario
        marioPosition = new Vector2( x: 100,  y: 100);
    }

    @Override
    public void render() {
        // Limpiar la pantalla (blanco)
        Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);

        // Llamar a handleInput para obtener el movimiento
        handleInput(Gdx.graphics.getDeltaTime());

        // Dibujar el sprite de Mario en la pantalla
        batch.begin();
        batch.draw(currentTexture, marioPosition.x, marioPosition.y);
        batch.end();
    }
```

```
    }

    private void handleInput(float deltaTime) {  1 usage
        moving = false;

        // Detectar las teclas presionadas para mover a Mario
        if (Gdx.input.isKeyPressed(Input.Keys.D)) {
            marioPosition.x += speed * deltaTime;
            moving = true;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.A)) {
            marioPosition.x -= speed * deltaTime;
            moving = true;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.W)) {
            marioPosition.y += speed * deltaTime;
            moving = true;
        }
        if (Gdx.input.isKeyPressed(Input.Keys.S)) {
            marioPosition.y -= speed * deltaTime;
            moving = true;
        }


        if (moving) {
            currentTexture = mario2Texture; // Si se mueve, usar el segundo sprite
        } else {
            currentTexture = mario1Texture; // Si no se mueve, usar el primer sprite
        }

        // Evitar que Mario salga de los límites de la pantalla
        marioPosition.x = Math.max(0, Math.min(marioPosition.x, Gdx.graphics.getWidth() - 64));
        marioPosition.y = Math.max(0, Math.min(marioPosition.y, Gdx.graphics.getHeight() - 64));
    }
```

```
    @Override
    public void dispose() {
        // Liberar recursos
        batch.dispose();
        mario1Texture.dispose();
        mario2Texture.dispose();
    }
}
```

CAPTURAS EN MOVIMIENTO


mario3


mario3