DESARROLLO DE VIDEOJUEGOS

ADAPTADOS A MOVILES CON SOFTWARE LIBRE

Javier Osuna Herrera

Índice

- Que se va a hacer en el taller
- ¿Que es libGDX?
- Lógica del videojuego
- Estructura de LibGDX
- Ciclo de vida de una aplicación
- Clase Texture y SpriteBacht
- Ejercicio1
- Clase Vector2 y Rectangle
- Ejercicio2
- Clase BitmapFont
- Ejercicio3
- Clase Preferences, Music y Sound
- Ejercicio4
- Twitter y Blog

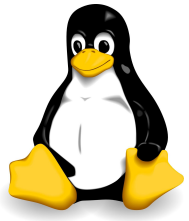
Slippery Penguin

Github: https://github.com/javosuher/Taller_Slippery_Penguin

libGDX

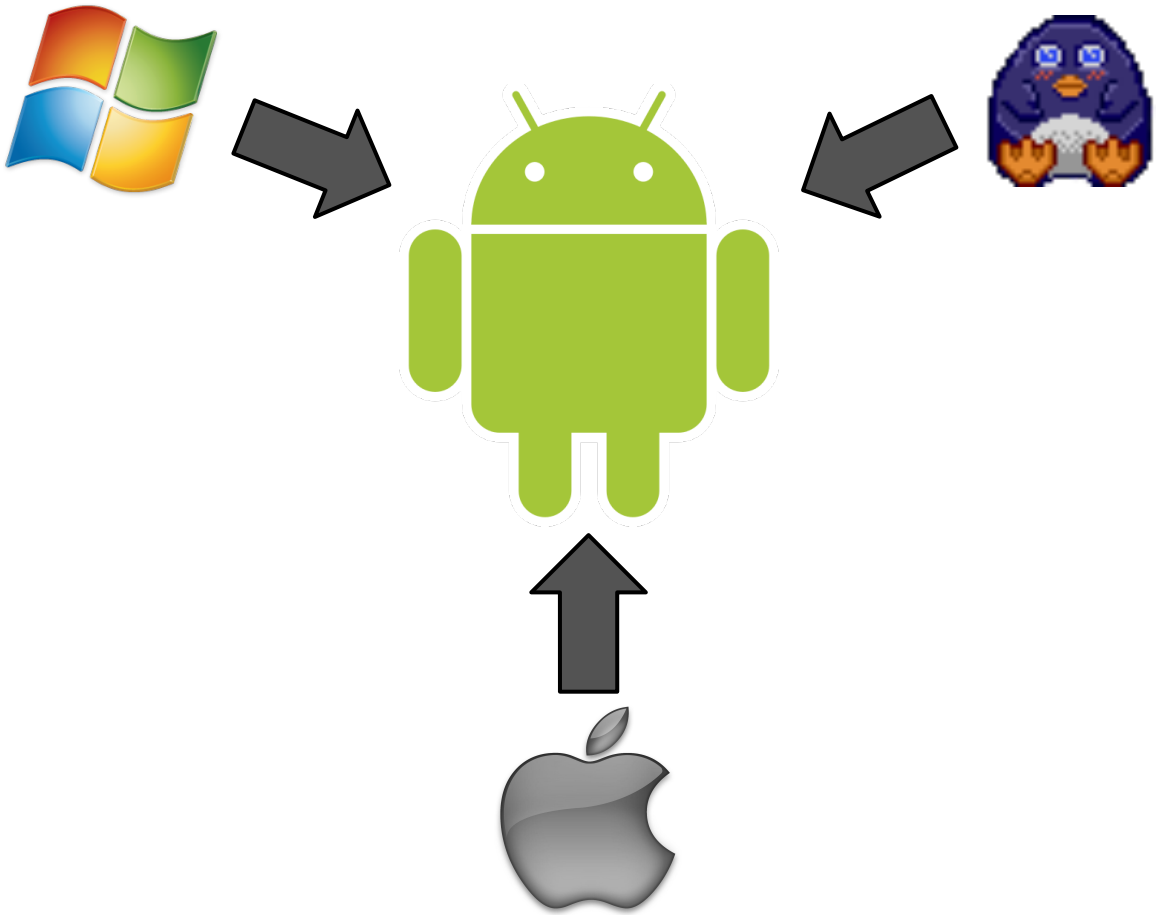
¿Qué es?

libGDX

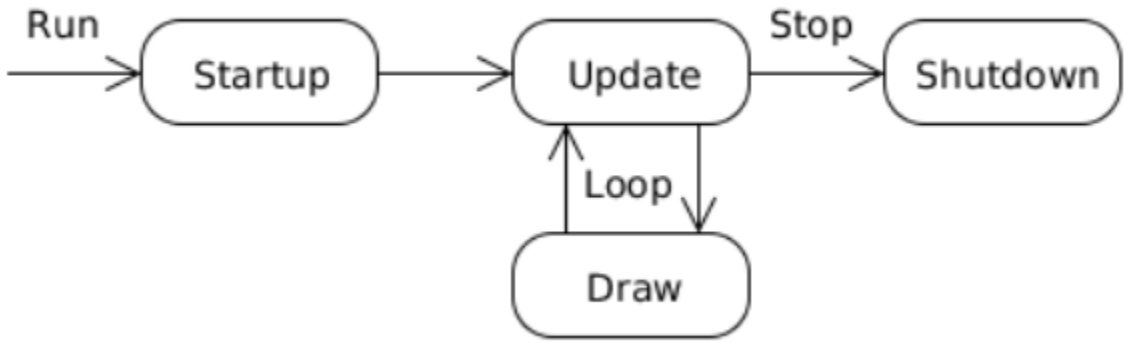


iOS

libGDX



Lógica del videojuego

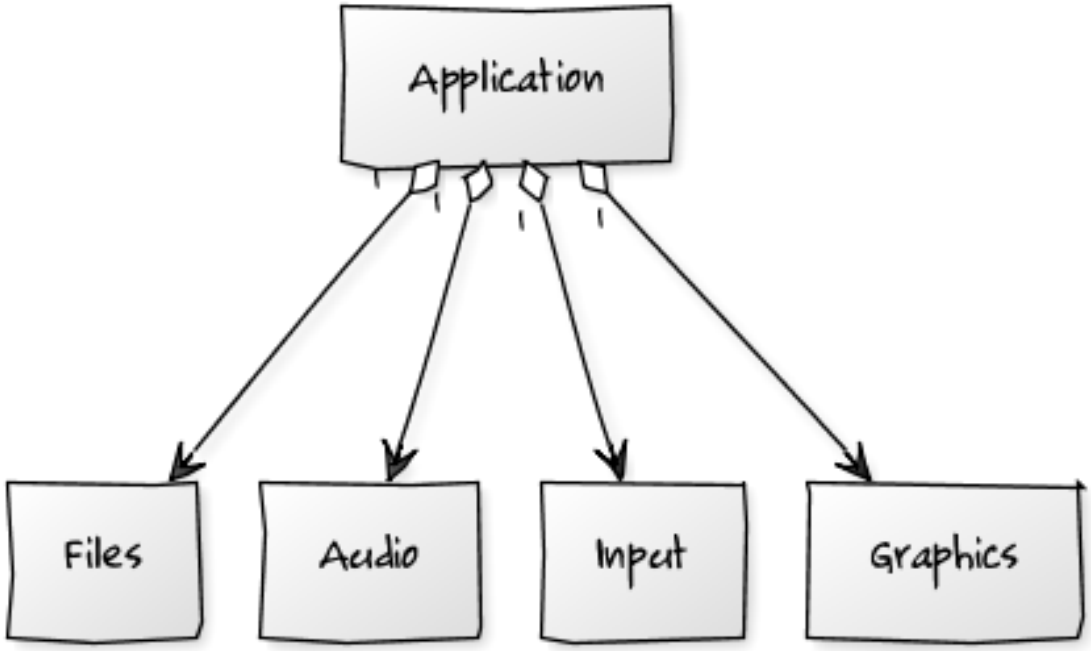


1º - Inicialización

2º - Game Loop

3º - Terminar y liberar memoria

Estructura de LibGDX



Estructura de LibGDX

- Marco de aplicación: Manejará el bucle principal y además estará encargado del ciclo de vida.

- Gráficos: Permitirá gestionar la representación de imágenes y objetos gráficos en la pantalla.

- Audio: Facilitará el acceso a los sonidos y música de la aplicación.

- Entrada y salida (Files): Permitirá para leer y escribir los diferentes ficheros de datos.

- Entrada (Input): Gestionará la entrada a través del teclado, pantalla táctil o acelerómetro.

Ciclo de vida de una aplicación en LibGDX

```
public class MyGame implements ApplicationListener {
        public void create () {
                // STUB
        }

        public void render () {
                // STUB
        }

        public void resize (int width, int height) {
                // STUB
        }

        public void pause () {
                // STUB
        }

        public void resume () {
                // STUB
        }

        public void dispose () {
                // STUB
        }
}
```

Ciclo de vida de una aplicación en LibGDX

- **Create:** Se llama una vez cuando se crea la aplicación.

- **Resize(int width, int height):** Se llama a este método cada vez que la pantalla del juego cambia su tamaño y el juego no está en el estado de pausa.

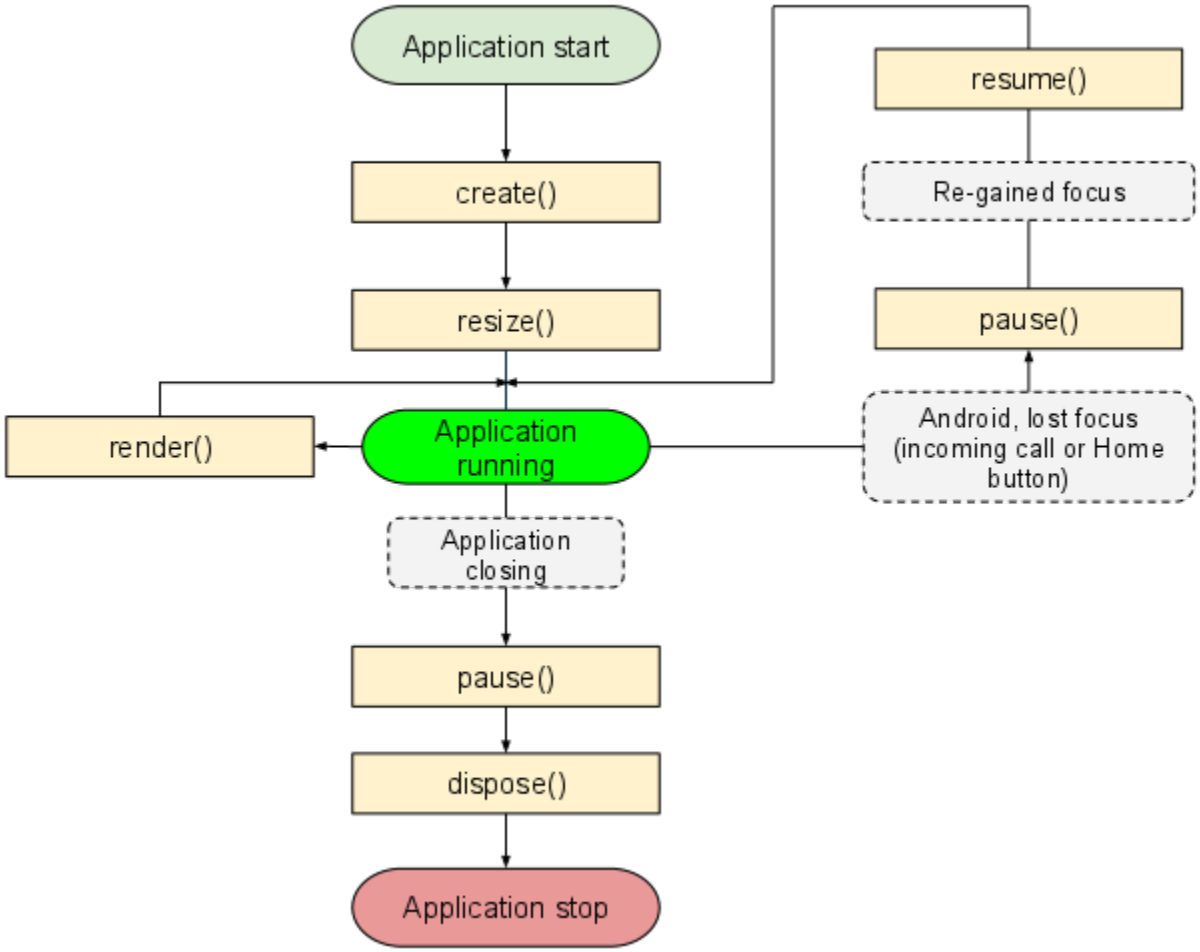
- **Render:** Método llamado por el bucle del juego de la aplicación cada vez que se renderiza. La actualización del juego también tiene lugar aquí antes de la representación real.

- **Pause:** El método de pausa se llama justo antes que se destruya la aplicación. En Android se llama cuando el botón de inicio se presiona o haya una llamada entrante.

- **Resume():** Este método es llamado sólo en Android, cuando la aplicación recibe el foco.

- **Dispose:** Se le llama cuando la aplicación se destruye. Es precedido por un Pause.

Ciclo de vida de una aplicación en LibGDX



Ejecutar mi juego

¿Cómo ejecuto mi juego en mi PC?

¿Y en Android?

¿?

Clase Texture

Texture: Es una clase que envuelve una textura estandar de OpenGL, se utiliza para imagenes simples. Ejemplo:

```
Texture Textura;
Textura = new Texture("data/miTextura.png");
Textura.setFilter(TextureFilter.Linear, TextureFilter.Linear);
```

Con setFilter controlamos la forma en la que la imagen se reescala, le añadimos el parametro TextureFilter.Linear en ambos casos, para que este reescalado sea lineal.

Clase SpriteBatch

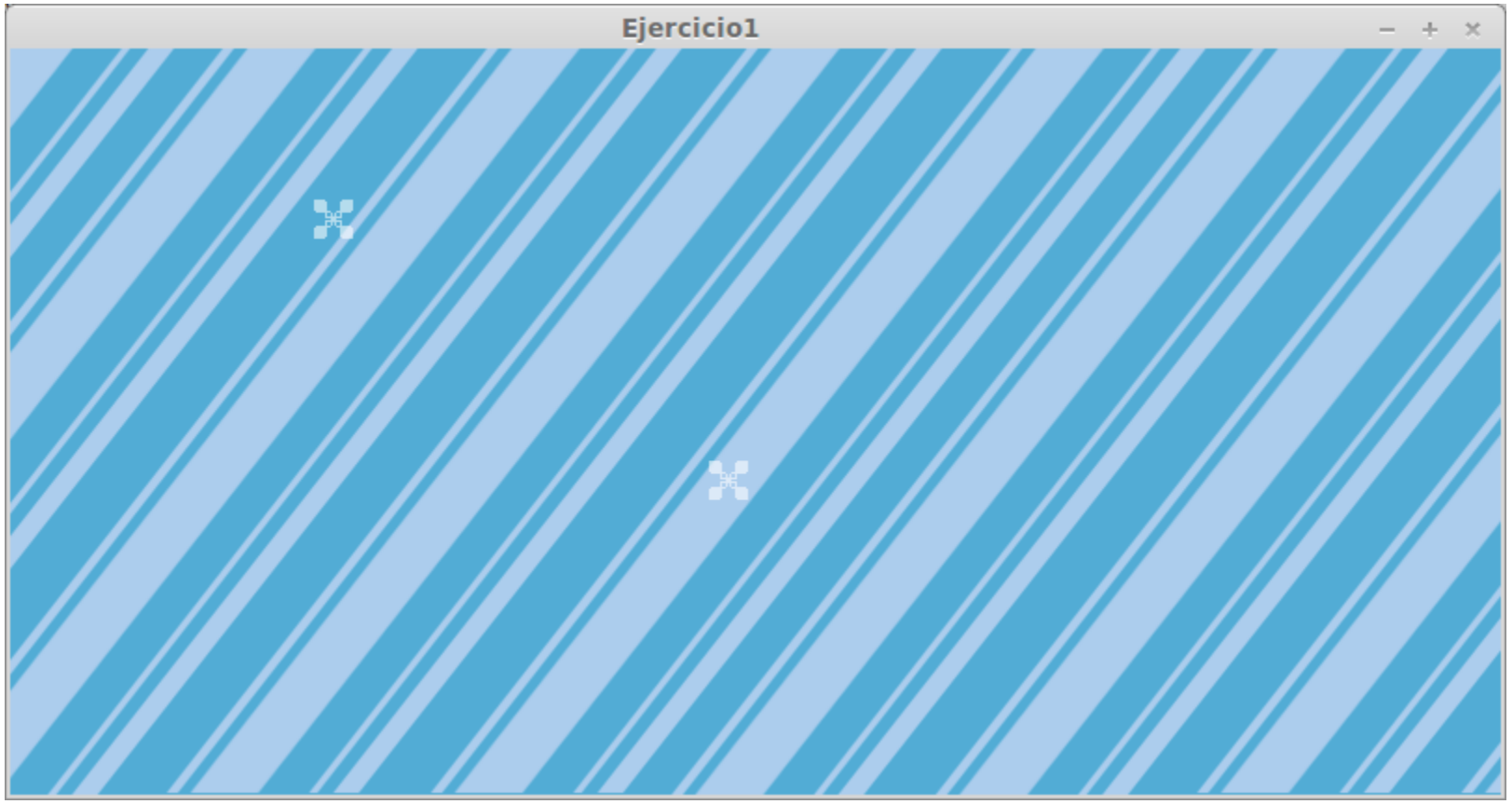
SpriteBatch: Nos permite dibujar rectángulos como referencias a texturas, es necesario para mostrar todo por pantalla (Grupo de Sprites (imagenes)) . Ejemplo:

```
SpriteBatch batch = new SpriteBatch;
batch.begin();
batch.draw(Textura, 0, 0, Textura.getWidth(), Textura.getHeight());
batch.end();
```

El segundo y tercer parametro del draw es la posición en el eje “x” y eje “y” donde se quiere dibujar la textura.

Ejercicio 1

Mostrar ventana con fondo del juego



Ejercicio 1

¡A trabajar!

Clase Vector2

Vector2: Vector de dos elementos.

```
Vector2 posicion = new Vector2(10,10);
```

Los parametros que se le pasan en el constructor son la posición en eje “x” y en el eje “y” respectivamente. Tiene su utilidad para pasarselo a los personajes del juego y con el controlar su movimiento.

Clase Rectangle

Rectangle: Crea un rectangulo 2D. Nos sirve para nuestros personajes. Ejemplo:

```
Rectangle bordes;
bordes = new Rectangle(posicion.x, posicion.y, anchura, altura);
```

El primer parametro es la posición en el eje “x”, el segundo la posición en el eje “y”, el tercero es la anchura del rectangulo, y por último la altura.

Clase Rectangle

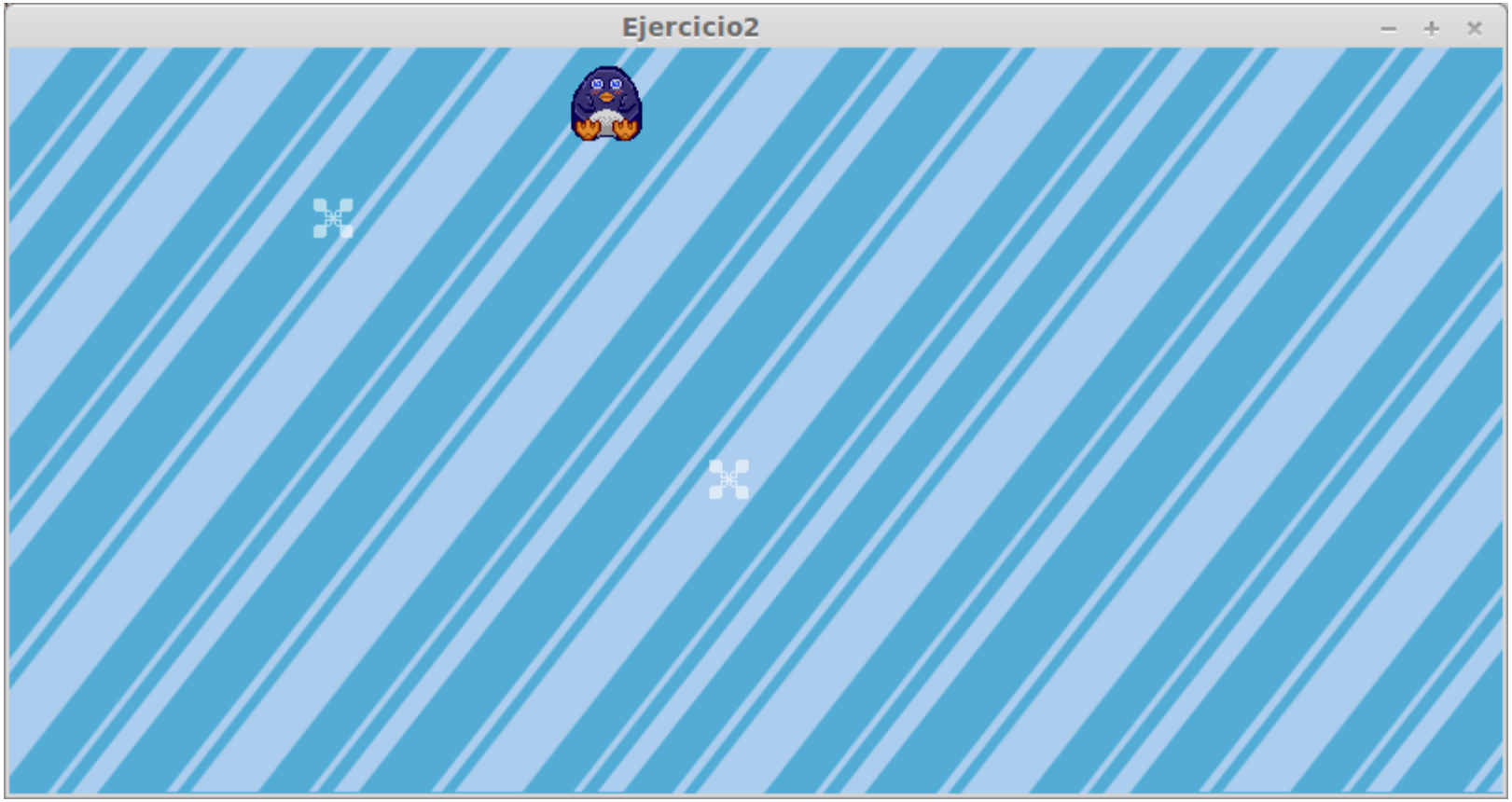
Esta clase tiene un método que sirve para ver si dos rectángulos se solapan:

```
rectanguloA.overlaps(rectanguloB);
```

Devuelve “true” si se solapan, y “false” en caso contrario.

Ejercicio 2

Mover el pingüino en PC y Android



Ejercicio 2

¡A trabajar!

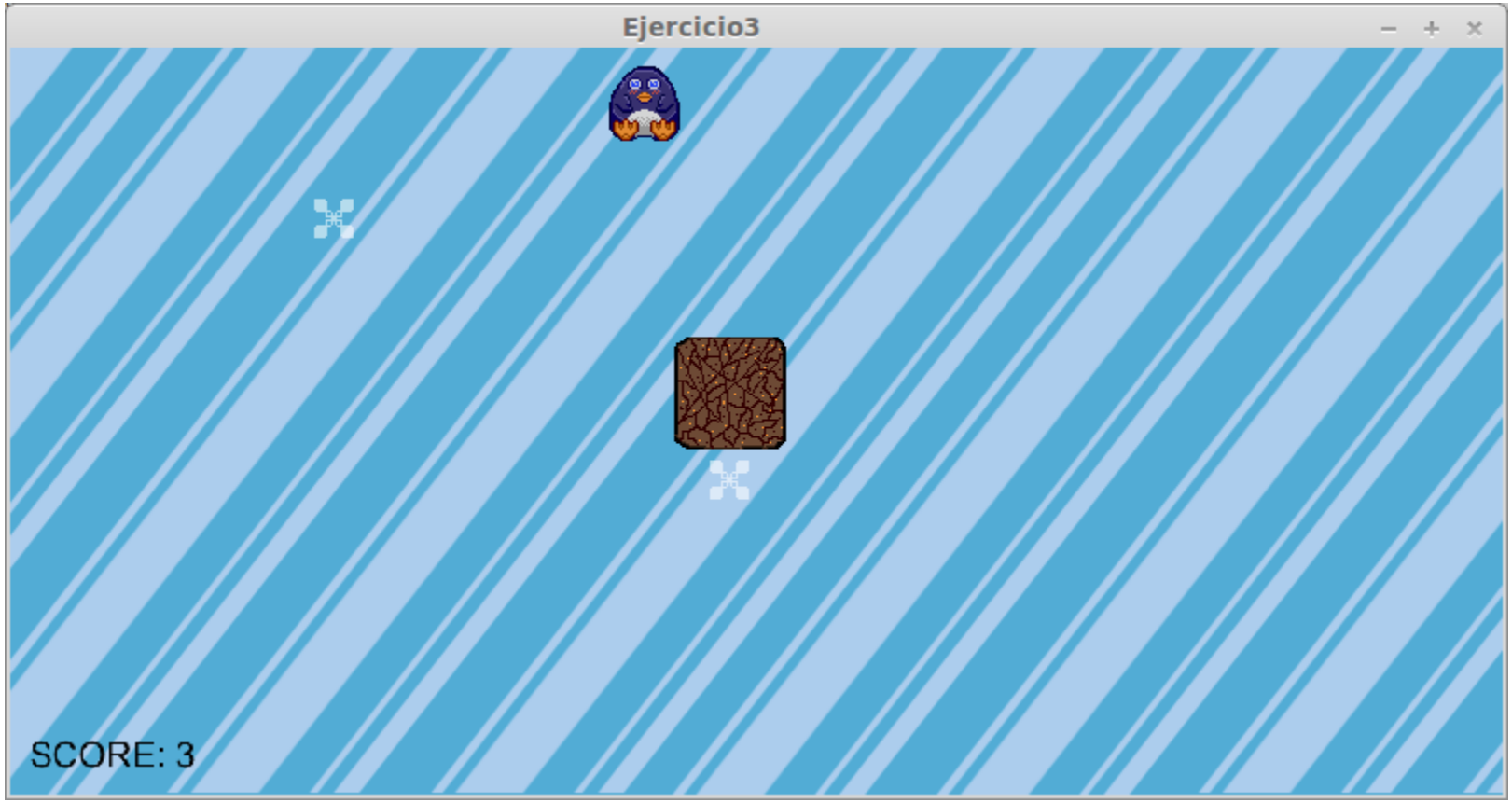
Clase BitmapFont

BitmapFont: Permite dibujar cadenas de texto. Se necesitan dos archivos, un archivo imagen con las fuentes dibujadas en él, y un archivo con extensión ".fnt" donde se indican los datos de la fuente, su nombre, coordenadas de cada carácter, espaciado, tamaño, etc.

```
BitmapFont font;
font = new BitmapFont(Gdx.files.internal("data/arial.fnt"),
                      Gdx.files.internal("data/arial.png"), false);
font.draw(batch, "Hola, soy un pingüino", 100, 500);
```

Ejercicio 3

Mover las rocas hacia arriba y realizar contador de puntos



Ejercicio 3

¡A trabajar!

Clase Preferences

Preferences: Permite almacenar y extraer datos de un fichero. Este fichero se aloja en la memoria del dispositivo independientemente de la plataforma que se use.

```
Preferences preferencias;
preferencias = Gdx.app.getPreferences("NombreDelFicheroDePreferencias");
```

Es muy útil para almacenar datos en memoria una vez se cierre el juego. Si no existe el fichero, se crea automáticamente.

Clase Preferences

Al guardar un valor hay que asignarle una clave para poder identificarlo de los diferentes elementos que contiene el fichero de preferencias.

```
puntuacion = preferencias.getInteger("Puntuacion", 0);
puntuacion += 1;
preferencias.putInteger("Puntuacion", puntuacion);
preferencias.flush();
```

Al usar el método “getInteger”, si ese campo no está en el fichero, lo crea con el valor del segundo parámetro.

Clase Music y Sound

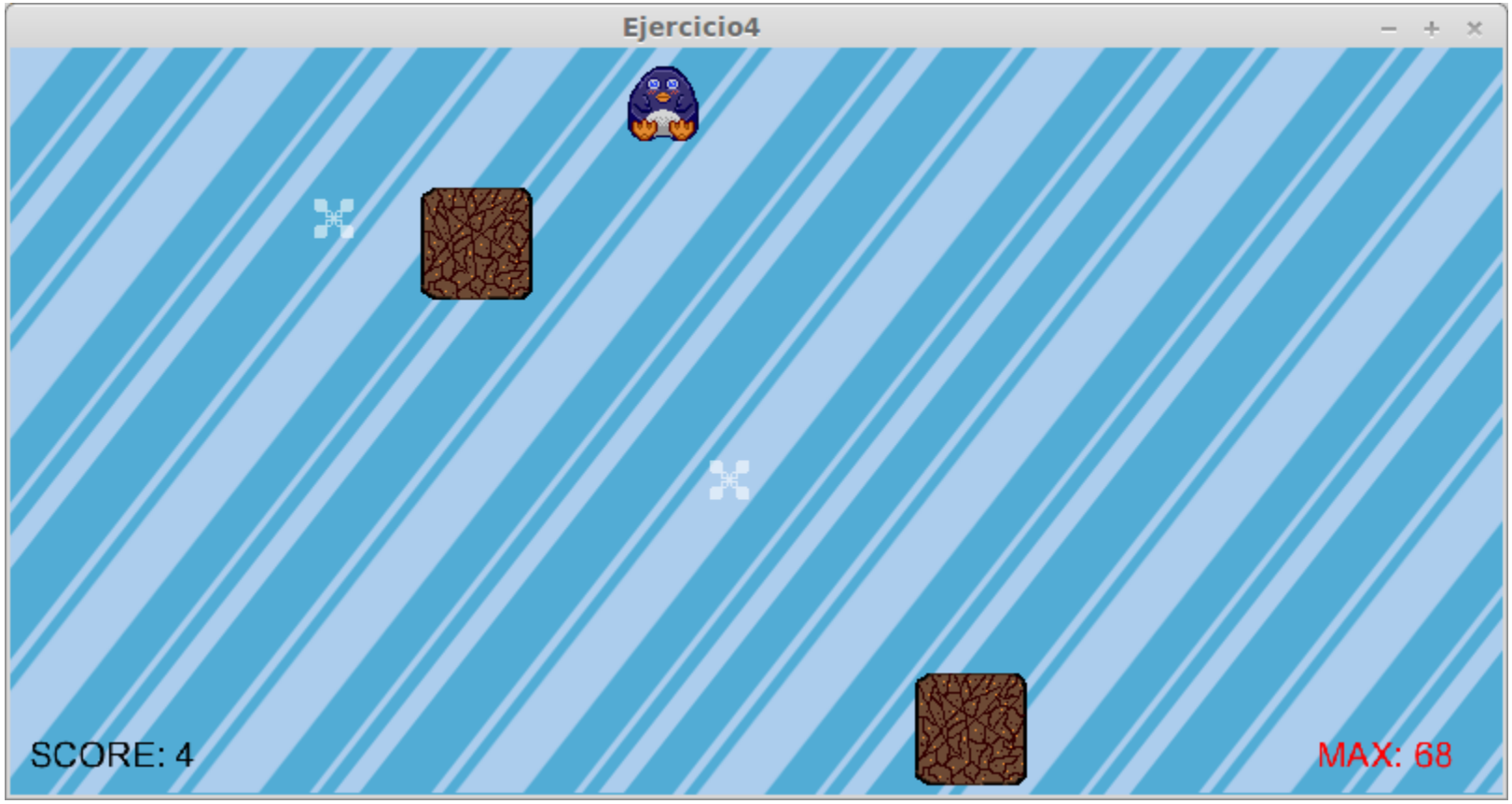
Music y Sound: Se usan para reproducir la música y el sonido del juego.

```
Music musica;
Sound sonido;
musica.play();
sonido.play();
```

Los sonidos sirven para audio de menor duración que la música. Así ahorramos espacio en memoria.

Ejercicio 4

Añadir la persistencia de la puntuación máxima, reiniciar el juego al morir y reproducir música y sonidos



Ejercicio 4

¡A trabajar!

¡Siguenos!

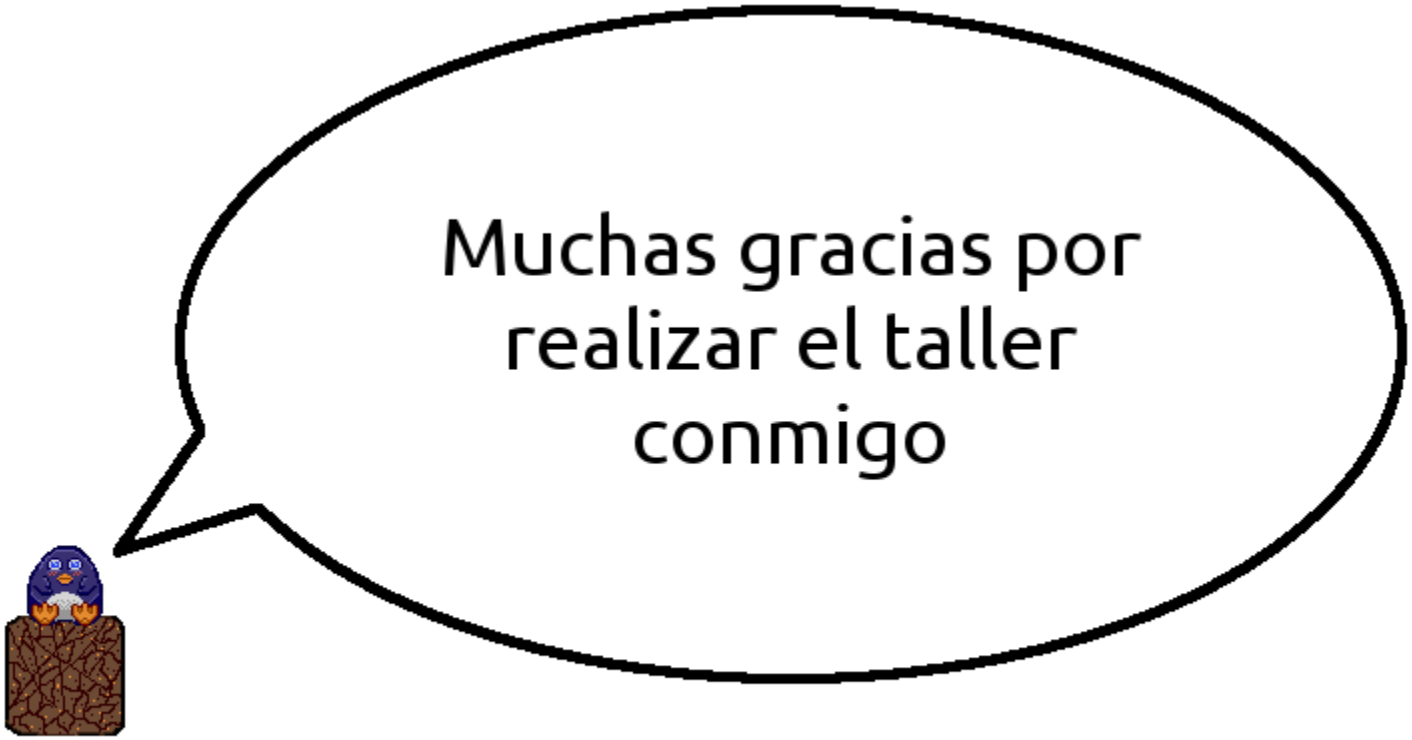
Twitter:

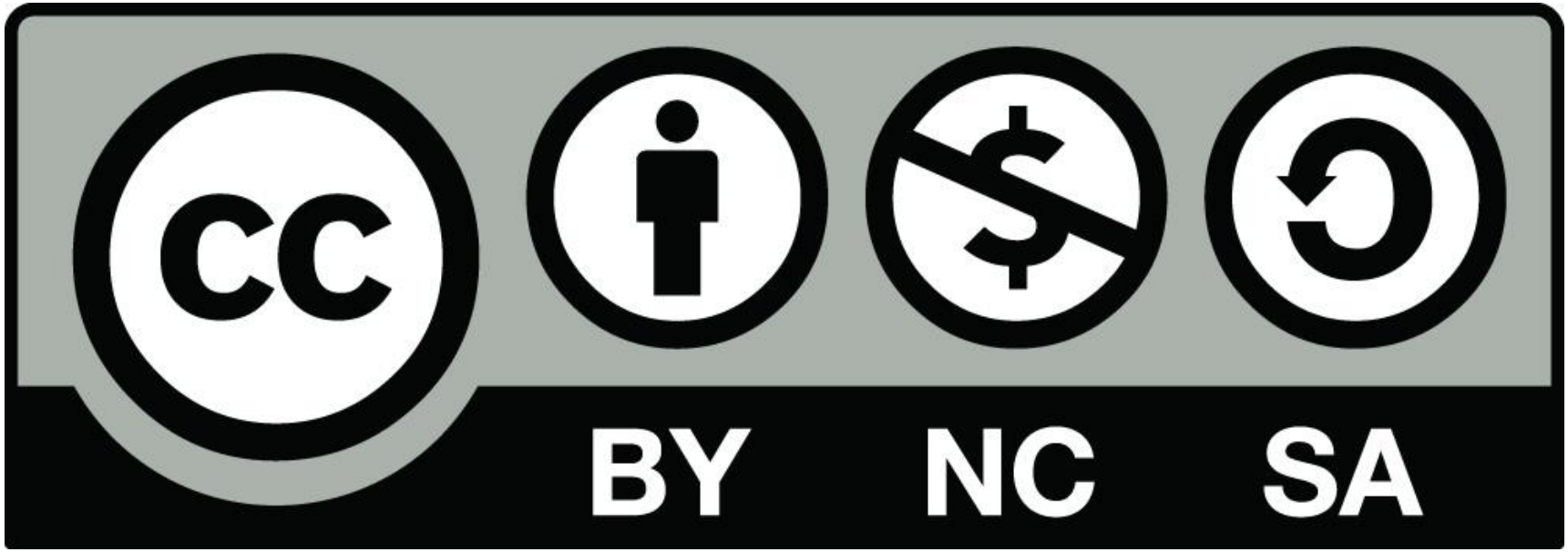
@TallerADVUCA

Blog:

http://talleradvuca.wordpress.com/

¡Gracias por asistir!





Financiado por la Actuación Avalada EL DESARROLLO DE VIDEOJUEGOS COMO REFUERZO DE CONOCIMIENTOS DE PROGRAMACIÓN: UNA EXPERIENCIA CON TECNOLOGÍAS MÓVILES (código AAA_14_024) de la convocatoria 2013/14 de la Universidad de Cádiz