Taller de desarrollo de videojuegos

con java y LibGDX

Javier Osuna Herrera

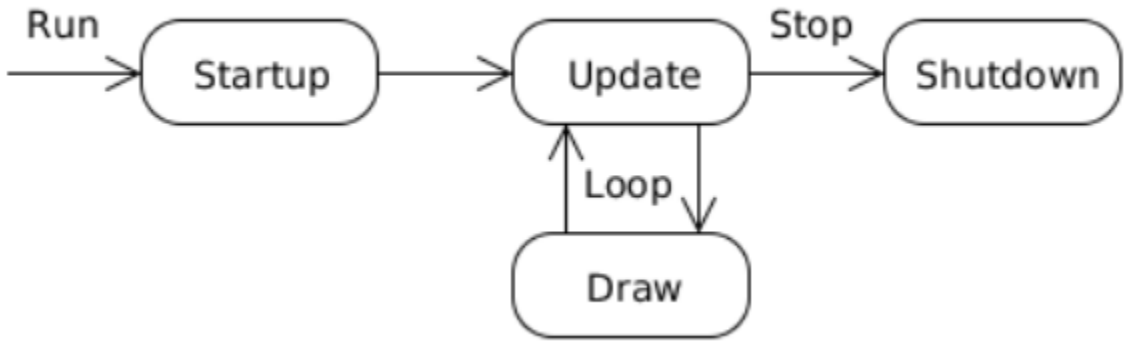
No vamos a hacer esto….

Índice

LibGDX

¿Qué es?

Lógica del videojuego

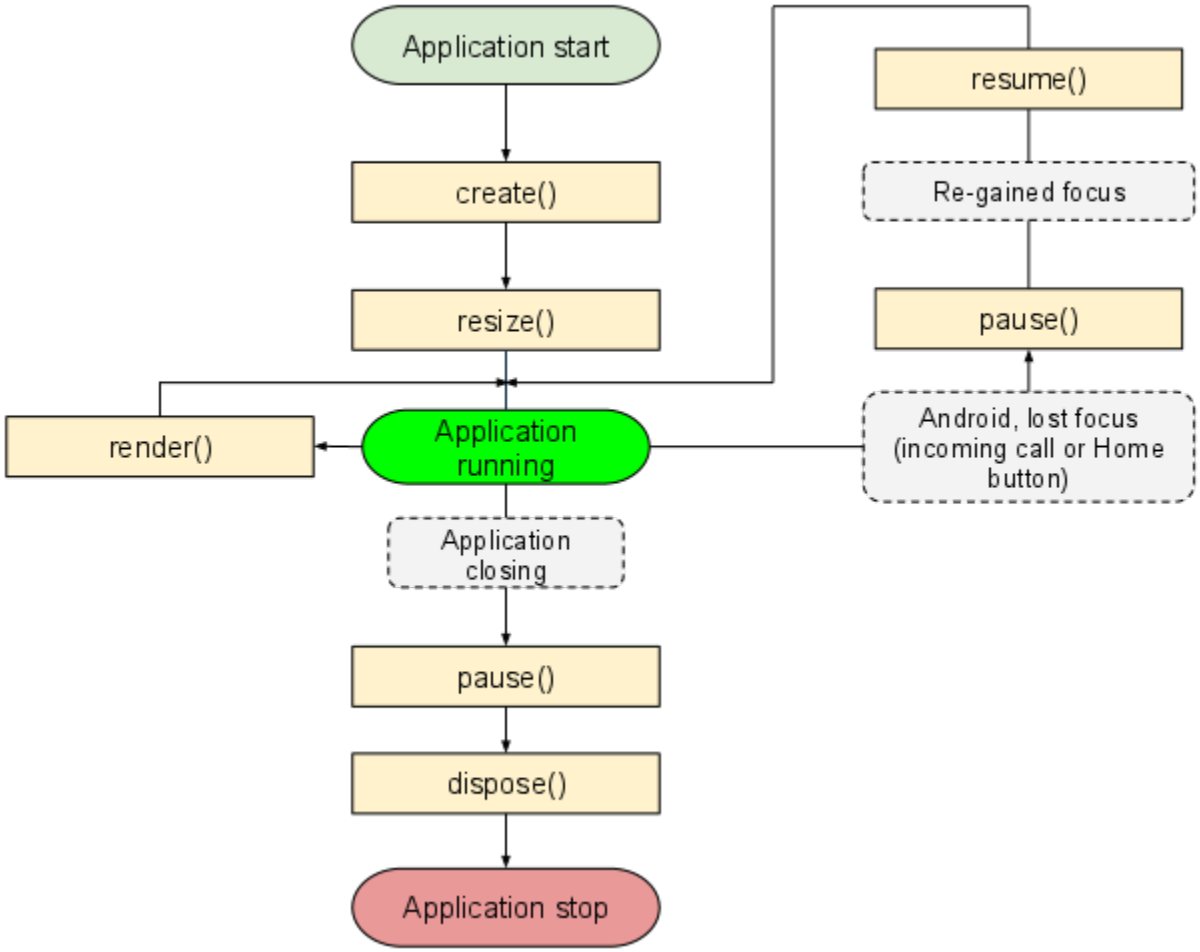


1º - Inicialización

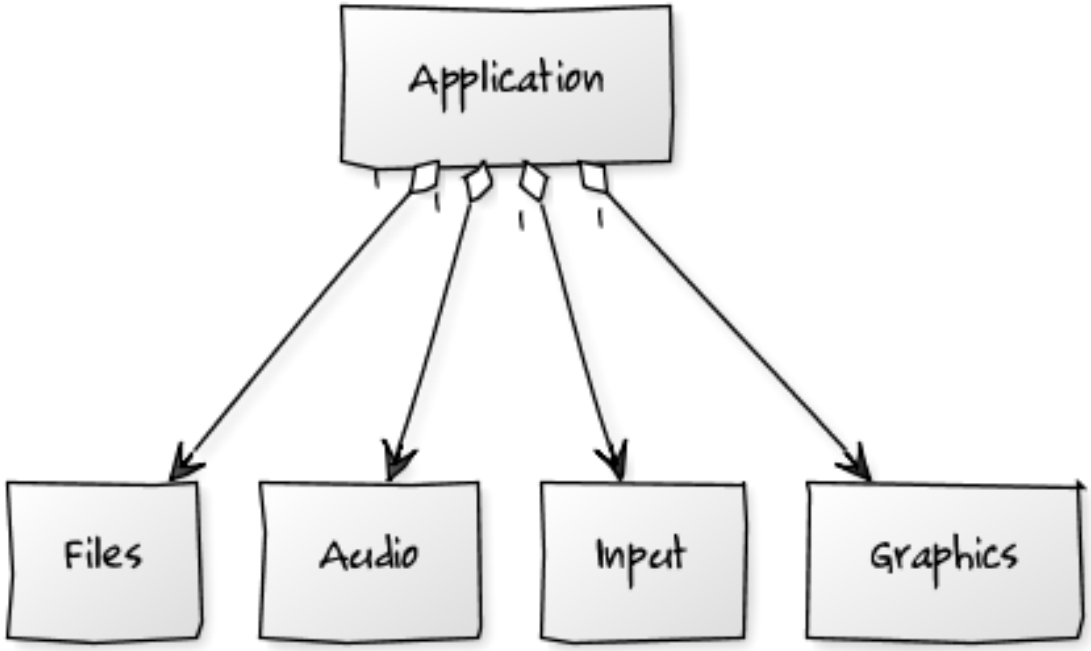
2º - Game Loop

3º - Terminar y liberar memoria

Lógica del videojuego



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.

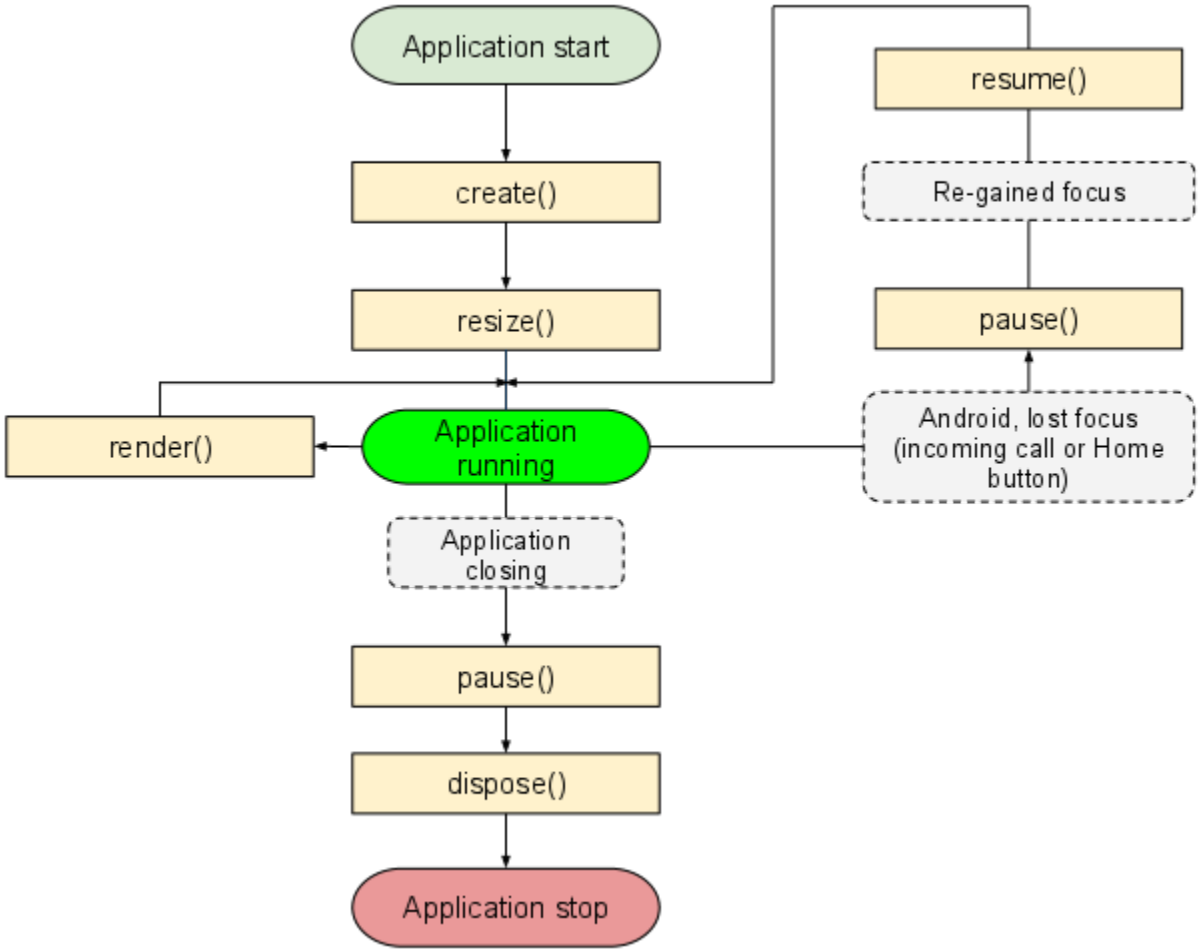
- **Render:** Método llamado por el bucle del juego de la aplicación cada vez que se renderiza. La actualización del juego también tiene lugar aquí antes de la representación real.

- **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



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

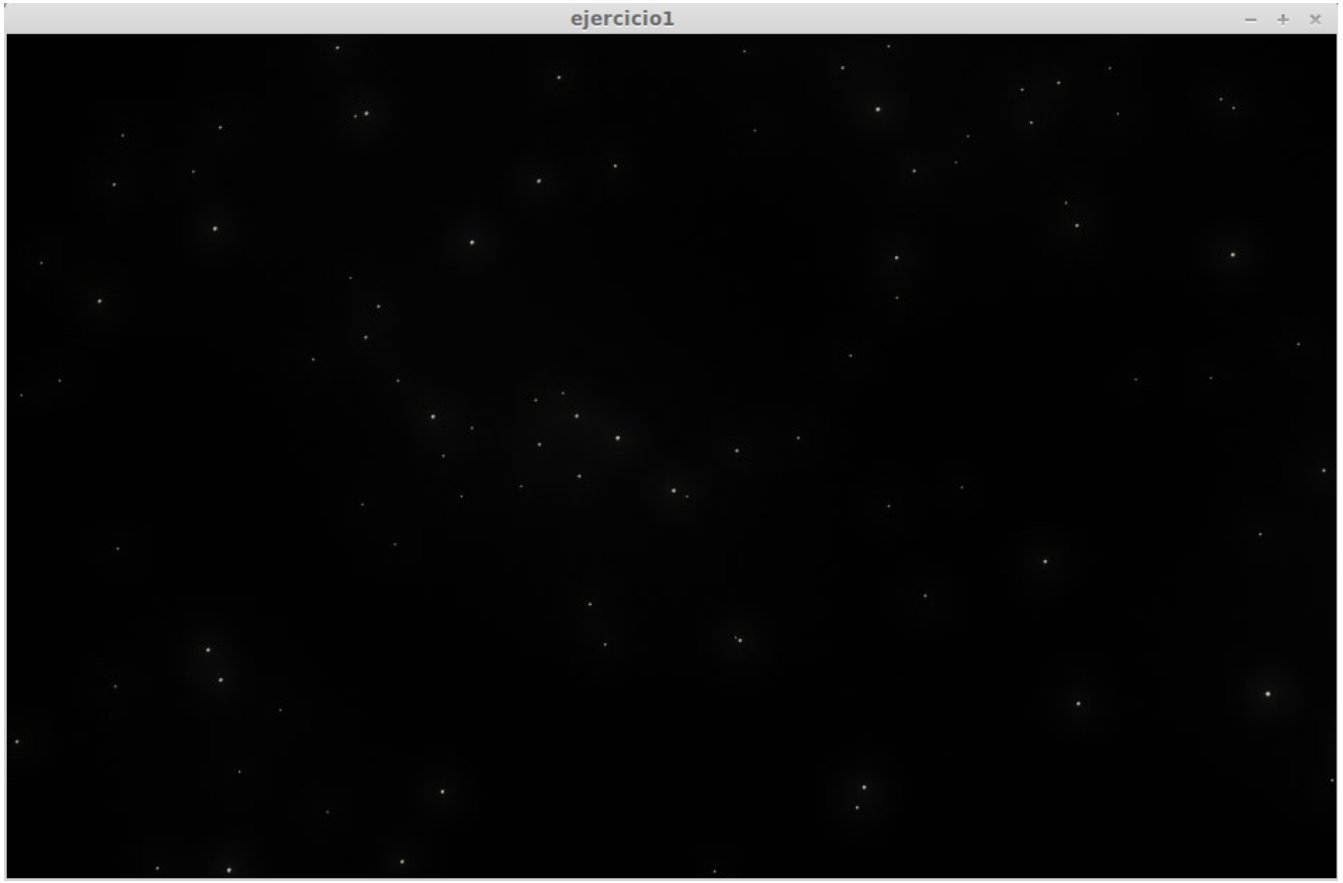
SpriteBatch: Nos permite dibujar rectángulos como referencias a texturas, es necesario para mostrar todo por pantalla (Grupo de Sprites (imagenes)) . Ejemplo:

```
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.

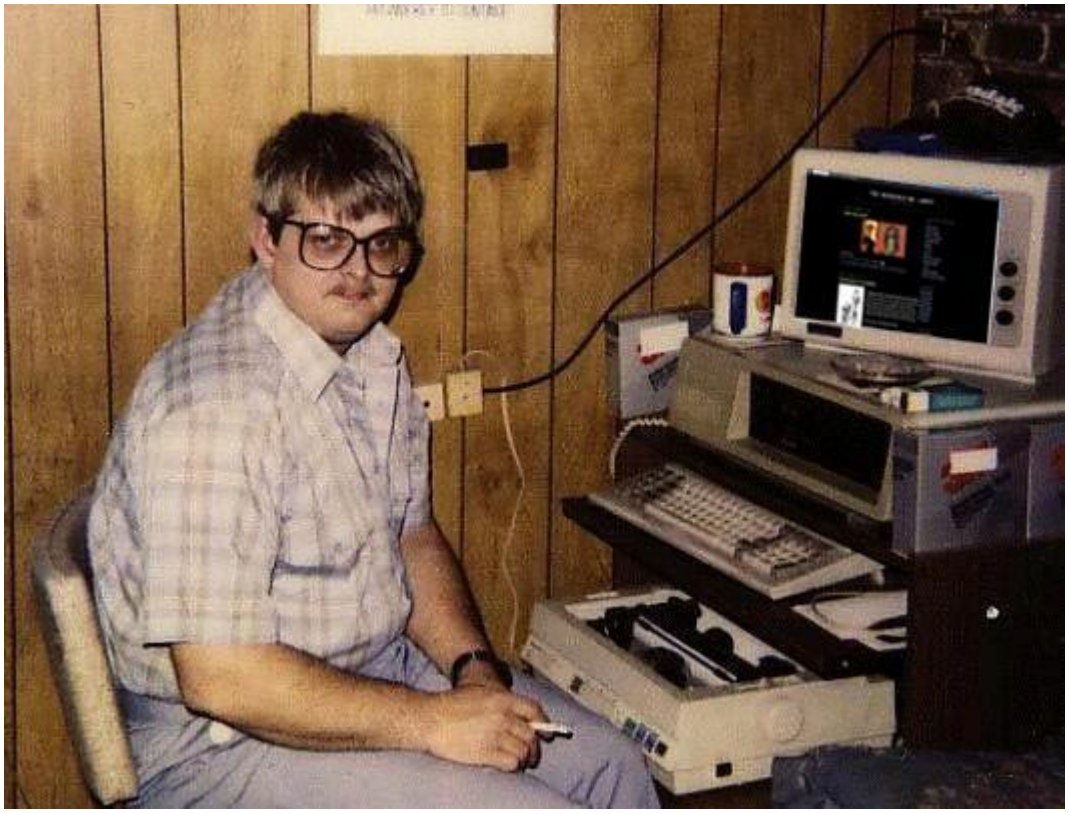
Ejercicio 1

Mostrar ventana del juego con su fondo



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

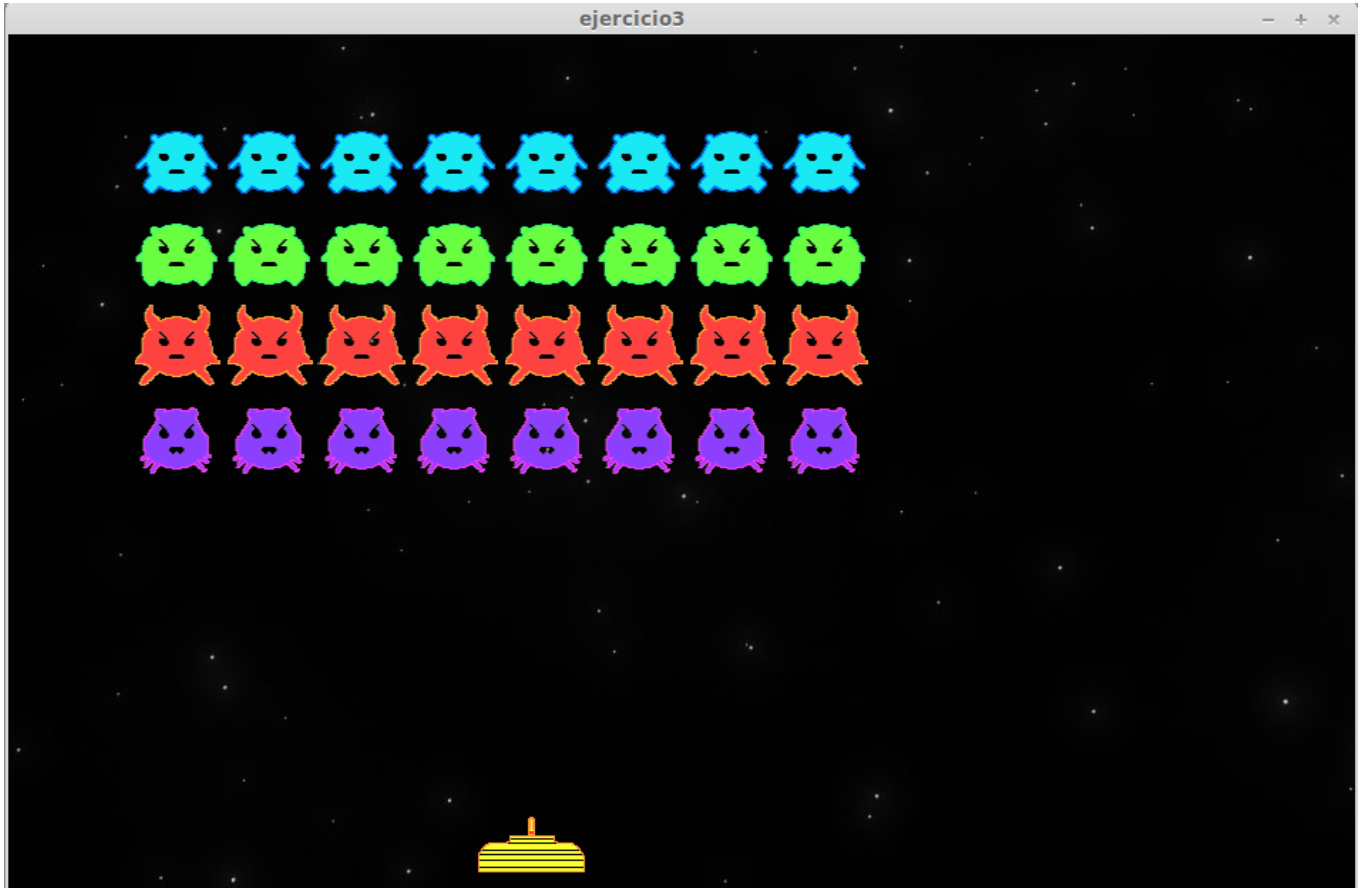
Mover la nave y ponerle límite a la derecha e izquierda

Ejercicio 2

¡A trabajar!

Ejercicio 3

Mover los aliens por la pantalla

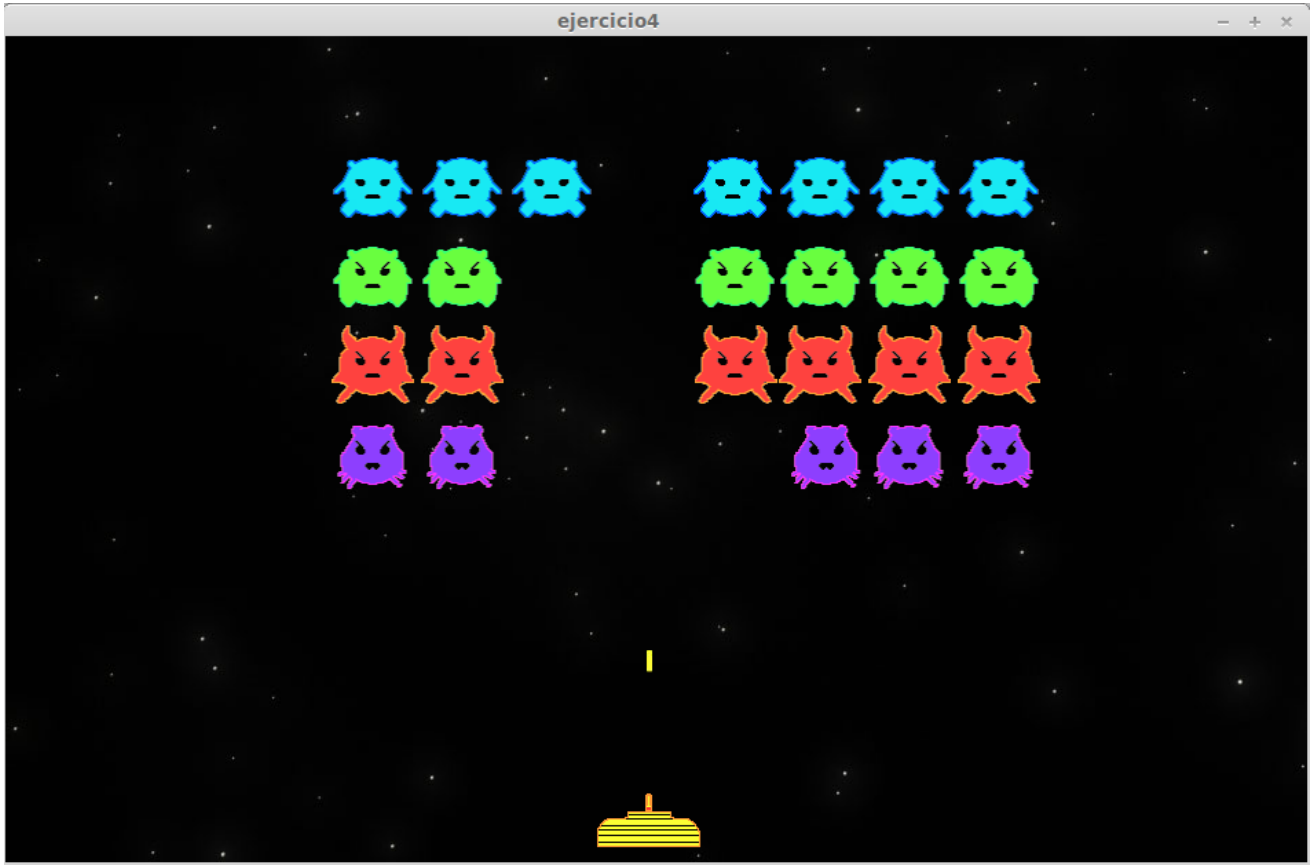


Ejercicio 3

¡A trabajar!

Ejercicio 4

Movimiento del disparo, que mate a los aliens y crear clase GameOverScreen



Ejercicio 3

¡A trabajar!

Futuro

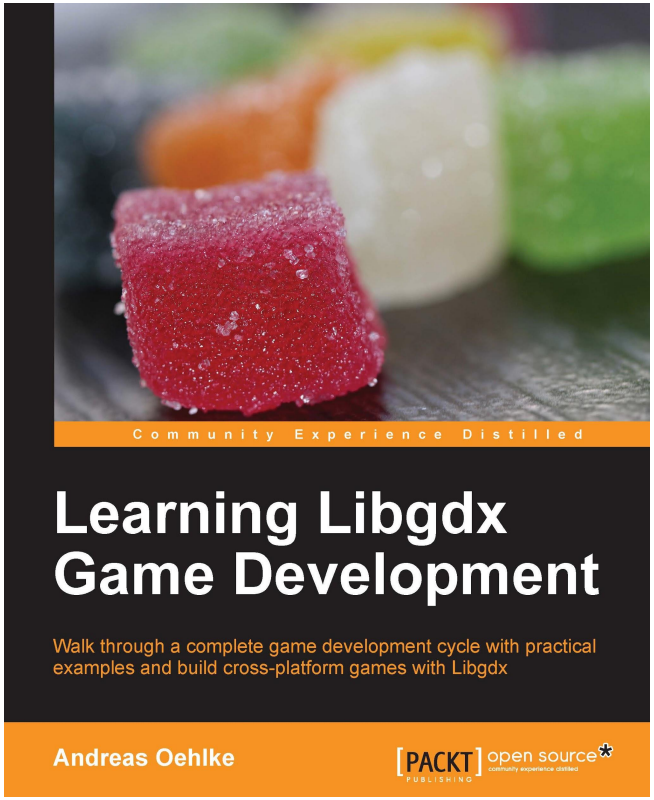
¡El juego se puede mejorar!

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

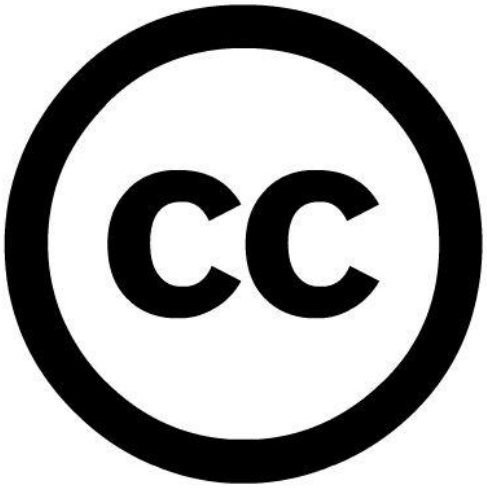
Twitter:

@TallerADVUCA

Bibliografía



¡Muchas 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