MANUAL TÉCNICO

Contenido

Introducción 3
Objetivos 3
1. Requerimientos 4
2. Ejecución 4
3. Almacenamiento de Datos 4
4. Métodos Constructores 5
   4.1. Constructor Jugador 5
   4.2. Constructor Enemigo 5
5. Hilos 6
   5.1. Items (Operaciones Aritméticas) 6
   5.2. Items (Bucles temporales) 7
6. Variables Globales 7

Introducción

El documento da a conocer cada una de las funcionalidades que cumplen los métodos y/o funciones de las clases utilizadas en la construcción del videojuego Invasores Espaciales.

Objetivos

Dar a conocer el uso adecuado del programa, para el acceso y buen funcionamiento del mismo, mostrando la secuencia en que se ejecuta, asimismo la descripción de los archivos relevantes del sistema los cuales resguardan la información requerida.

Brindar soporte a nuevos técnicos sobre cómo dar un mantenimiento óptimo del software o para poder mejorarlo.

1. Requerimientos

Para que el sistema pueda ser utilizado en cualquier sistema operativo debe cumplir con los siguientes requerimientos:

- Tener instalado JDK 14.0.2 o superiores de Oracle.
- Tener instalado Java 8 Update 281.

2. Ejecución

Existe más de una forma para ejecutar el programa, sin embargo, a continuación, se describen las formas más sencillas para hacer uso del software en sistemas operativos Windows.

La primera opción está enfocada es mediante el Símbolo del sistema o cmd.

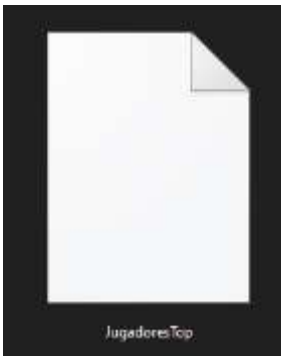
1. Usar la combinación de teclas Win + R, y luego escribir cmd y Aceptar.
2. Escribir “cd” (sin comillas), luego espaciar una vez y escribir la ruta en donde se encuentra el archivo ejecutable (.jar). Ejemplo: cd C:\User\Proyecto
3. Escribir el comando: java –jar; seguido del nombre del archivo con su respectiva extensión. Ejemplo: java -jar Proyecto.jar

Después de realizar estos pasos el programa comenzará su ejecución automáticamente.

La otra opción, que es más común, es ejecutarla haciendo doble click sobre el ejecutable como con cualquier otro programa.

3. Almacenamiento de Datos

Para mantener las configuraciones del juego y recuperar las puntuaciones más altas, aunque no esté en ejecución, se implementó Serialización, de tal modo que, al ingresar al videojuego o al terminar la ejecución, las configuraciones y puntuaciones serán codificadas en archivos binarios para que al volver a ejecutar el programa se realice un proceso opuesto al de la codificación para que el programa recupere inmediatamente los datos de configuración y puntaje tal y como fueron codificados en la ejecución anterior.



4. Métodos Constructores

4.1. Constructor Jugador

Es el método que otorga las propiedades al componente del jugador, tanto las características físicas como lógicas, además de implementar un arreglo de componentes con el fin del generar los proyectiles. Esta clase hereda de java swing, específicamente un JPanel e implementa hilos.

```
public JPanel[] laser = new JPanel[1000];
public static boolean thread = false,alerta1 = true,alerta2 = true;
public static int puntos = 0,deltaS;
public int lasers = 0;
String tecla;
public Jugador() {
    int lado = 55;
    this.setOpaque(false);
    this.setLayout(null);
    this.add(new Icono("imagenes/Jugador/Jugador.png",lado));
    for (int i = 0; i < laser.length; i++) {
        laser[i]= new JPanel();
        laser[i].setSize(30,7);
        laser[i].setLayout(null);
        laser[i].setOpaque(false);
        laser[i].add(new Icono(Estatico.azul,30,7));
    }
}
```

4.2. Constructor Enemigo

La instancia de esta clase resultará en la construcción de paneles, debido a que herede de java swing, de JPanel para ser exacto, además recibe como parámetro un string que será la ruta de la imagen que lo identificará como enemigo.

```
package Nave;
import Interfaz.Icono;
public class Enemigo extends JPanel {
    private static final long serialVersionUID = 1L;
    public Enemigo(String image) {
        int lado = 55;
        this.setSize(lado,lado);
        this.setOpaque(false);
        this.setLayout(null);
        this.add(new Icono(image,lado));
    }
}
```

5. Hilos

5.1. Items (Operaciones Aritméticas)

Los ítems que añaden y restan tiempo, puntos y velocidades son activados con coordenadas, ya que se detecta colisión al coincidir bordes y puntos en específico dentro de rangos aceptados y definidos tanto por el tamaño del ítem como del jugador.

```java
package AnimacionObjetosJuego;
import static Ajustes.ControladorAjustes.config;
public class ItPlusT extends JPanel implements Runnable {
    private static final long serialVersionUID = 1L;
    int x;
    int y;
    public ItPlusT(int x,int y) {
        this.x = x;
        this.y = y;
        this.setSize(33,33);
        this.setOpaque(false);
        this.setLayout(null);
        Thread hilo = new Thread(this);
        hilo.start();
    }
    public void run() {
        try {
            int velocidad;
            if(config[0].getVel().equals("Normal")) {
                velocidad = 2;
            }else {
                velocidad = 3;
            }
            for(int i = x; i >= -30; i-=velocidad) {
                item(this);
                this.setLocation(i,y);
                if(EstadoBarra.hilo == true && this.getX() <= Espacio.player.getX()+55 &&
                        ((this.getY() >= Espacio.player.getY() && this.getY() <= Espacio.player.getY()+55) ||
                        Espacio.player.getY() >= this.getY() && Espacio.player.getY() <= this.getY()+33)) {
                    EstadoBarra.AumDisT = 10;
                    new Sonido(Estatico.aumento);
                    this.setVisible(false);
                    Principal.pantalla.espacio.remove(this);
                    break;
                }
                Thread.sleep(20);
            }
            this.setVisible(false);
            Principal.pantalla.espacio.remove(this);
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
    public JPanel item(JPanel panel) {
        panel.add(new Icono(Estatico.item1,33,33));
        panel.setVisible(true);
        return panel;
    }
}
```

5.2. Items (Bucles temporales)

Los ítems que añaden y restan velocidades son activados con coordenadas, ya que se detecta colisión al coincidir bordes y puntos en específico dentro de rangos aceptados y definidos tanto por el tamaño del ítem como del jugador. Además de que dentro del bucle existe una condición que al cumplir con la condición de colisión se activará para determinar el periodo de tiempo que tendrá efecto el haber colisionado con el ítem.

6. Variables Globales

Se usaron variables globales para el uso de hilos, además de los parámetros que reciben algunos constructores para posteriormente inicializarlos y utilizarlos en los hilos.

También fueron utilizados componentes establecidos como globales, como labels, botones campos de texto etc., con el fin de generar componentes a partir de clases heredadas.