



Universidad Nacional
de General Sarmiento

Programación 1

Trabajo práctico “La invasión de los Zombies Grinch”

integrantes :

Cavila Andres

Galeano Adriana

Medina Solange

Correos electronicos:

andrescaliva25@gmail.com

adrianagaleano077@gmail.com

solemedina193@gmail.com

Docentes :

Verónica Moyano

Cesar Niveyro

Introducción:

En el pueblo de Whoville, el Grinch se convirtió en zombi y tiene la capacidad de copiarse a sí mismo, en la época de Navidad sigue atacando al pueblo para robar los regalos. El científico del pueblo creó algunos nuevos tipos de plantas con la capacidad de atacar y defenderse de los Zombies Grinch. Por ejemplo está Rose Blade una rosa que tira bolas de fuego y Wall-Nut una nuez con una gran resistencia y defensa, esta última sirve para frenar el ataque de los Zombies Grinches.

El objetivo es desarrollar un video juego para prepararse y poder proteger los regalos del ataque de los Zombies Grinch en esta Navidad.

Descripción:

A continuación, se dará una explicación general de cada clase implementada, describiendo las variables de instancia y dando una breve descripción de cada método.

Clase: Casillero

La clase `Casillero` representa una casilla de un tablero dentro del juego. Cada casillero tiene una posición, un tamaño y un color que indica su estado (verde brillante o verde oscuro). La clase permite dibujar la casilla en pantalla y ver sus propiedades.

Variables de instancia

Nombre	Tipo	Descripción
x	int	Coordenada X del centro del casillero en la ventana del juego.
y	int	Coordenada Y del centro del casillero en la ventana del juego.
ancho	int	Ancho del casillero.
alto	int	Alto del casillero.
esVerde	boolean	Indica si el casillero se dibuja en verde brillante (<code>true</code>) o verde oscuro (<code>false</code>).

Constructores

```
public Casillero(int x, int y, int ancho, int alto, boolean esVerde)
```

- Inicializa un objeto `Casillero` con su posición, tamaño y color.
- Parámetros:
 - `x`, `y`: posición del casillero.
 - `ancho`, `alto`: dimensiones del casillero.

- o `esVerde`: estado de color del casillero.

:

Métodos:

Método	Tipo de Descripción retorno	
<code>public void dibujar(Entorno entorno)</code>	<code>void</code>	Dibuja el casillero en la pantalla usando la librería <code>Entorno</code> . Si <code>esVerde</code> es <code>true</code> , se dibuja verde; si es <code>false</code> , verde oscuro.
<code>public int getX()</code>	<code>int</code>	Devuelve la coordenada X del casillero.
<code>public int getY()</code>	<code>int</code>	Devuelve la coordenada Y del casillero.
<code>public int getAncho()</code>	<code>int</code>	Devuelve el ancho del casillero.
<code>public int getAlto()</code>	<code>int</code>	Devuelve el alto del casillero.

Clase: `Regalos`

La clase `Regalos` representa un objeto regalo dentro del juego. Cada regalo tiene una posición en la pantalla y un tamaño fijo. La clase permite dibujar el regalo en la ventana del juego y consultar sus coordenadas.

Variables de instancia

Nombr e	Tipo	Descripción
x	double	Coordenada X del centro del regalo en la ventana del juego.
y	double	Coordenada Y del centro del regalo en la ventana del juego.

Constructores

```
public Regalos(double x, double y)
```

- Inicializa un objeto `Regalos` con su posición en la pantalla.
- Parámetros:
 - `x`: coordenada horizontal del regalo.
 - `y`: coordenada vertical del regalo.

Métodos

Método	Tipo de Descripción retorno	
public void dibujar(Entorno e)		Dibuja el regalo como un cuadrado azul de tamaño fijo en la ventana del juego utilizando la librería Entorno.
public double getX()		Devuelve la coordenada X del regalo.
public double getY()		Devuelve la coordenada Y del regalo.

Clase: Planta

La clase `Planta` representa una planta dentro del juego, que puede ser de tipo normal o especial (`roseBlade`). Cada planta tiene posición, vida, tamaño y puede disparar proyectiles si es del tipo `roseBlade`.

Variables de instancia

Nombre	Tipo	Descripción
x	int	Coordenada horizontal (fila) de la planta.
y	int	Coordenada vertical (columna) de la planta.
vida	int	Cantidad de vida que le queda a la planta.
diametro	int	Tamaño visual de la planta al dibujarla.
seleccionada	boolean	Indica si la planta está actualmente seleccionada por el jugador.

<code>roseBlade</code>	<code>boolean</code>	Indica si la planta es de tipo especial que puede disparar proyectiles.
<code>intervalosDisparo</code>	<code>int</code>	Cantidad de ticks entre disparos para plantas tipo <code>roseBlade</code> .
<code>contadorDisparos</code>	<code>int</code>	Contador para controlar cuándo la planta puede disparar de nuevo.
<code>juego</code>	<code>Juego</code>	Referencia al objeto <code>Juego</code> para interactuar con otras mecánicas (por ejemplo, disparar proyectiles).

Constructores

```
public Planta(int x, int y, boolean roseBlade, Juego juego)
```

- Inicializa una planta en la posición (`x`, `y`).
- Parámetros:
 - `x`, `y`: coordenadas de la planta.
 - `roseBlade`: indica si es del tipo especial que dispara.
 - `juego`: referencia al juego para ejecutar disparos.

- Define el tamaño y el intervalo de disparos si corresponde.

Métodos

Método	Tipo de Descripción retorno
public void dibujar(Entorno e)	Dibuja la planta como un círculo amarillo o azul (si está seleccionada). Si es <code>roseBlade</code> , dibuja un círculo rojo más pequeño encima.
public void actualizarEstado()	Controla el disparo automático de la planta si es <code>roseBlade</code> . Cada <code>intervalosDisparos</code> ticks dispara un proyectil a la derecha usando el método <code>juego.disparar()</code> .
public boolean contienePunto(int px, int py)	Verifica si un punto (<code>px, py</code>) está dentro del área de la planta (útil para selección con clic).
public void moverPlanta(int dx, int dy)	Mueve la planta sumando desplazamientos a sus coordenadas X e Y.
public void recicbirDanio(int danio)	Reduce la vida de la planta según el daño recibido.

public boolean plantaMuerta()	boolean	Devuelve true si la vida de la planta es 0 o menor.
public int getX()	int	Retorna la coordenada X de la planta.
public int getY()	int	Retorna la coordenada Y de la planta.
public int getDiametro()	int	Retorna el tamaño de la planta.
public boolean isSeleccionada()	boolean	Indica si la planta está seleccionada.
public boolean isRoseBlade()	boolean	Indica si la planta es del tipo especial.
public void setSeleccionada(boolean seleccionada)	void	Permite cambiar el estado de selección de la planta.
public void moverArriba(int desplazamiento)	void	Mueve la planta hacia arriba.
public void moverAbajo(int desplazamiento)	void	Mueve la planta hacia abajo.
public void moverIzquierda(int desplazamiento)	void	Mueve la planta hacia la izquierda.
public void moverDerecha(int desplazamiento)	void	Mueve la planta hacia la derecha.

```
public void setX(int x)    void      Cambia la coordenada X de la
                           planta.
```

Clase: *Grinch*

La clase *Grinch* representa a un zombie del juego que se desplaza hacia la izquierda, puede recibir daño y morir. También puede interactuar con los objetos *Regalos* para detectar colisiones, se dibuja como un conjunto de círculos de colores rojo y negro.

Variables de instancia

Nombre	Tipo	Descripción
x	double	Coordenada horizontal del grinch (fila).
y	double	Coordenada vertical del grinch (columna).
velocidad	double	Velocidad a la que se mueve hacia la izquierda.
vida	double	Cantidad de vida que le queda al grinch.
tamanio	int	Tamaño visual del grinch (radio aproximado para colisiones).

Constructores

```
public Grinch(double x, double y)
```

- Inicializa un grinch en la posición (`x`, `y`).

- Configura valores por defecto:

- `velocidad = 1`

- `vida = 10`

- `tamanio = 20`

Métodos

Método	Tipo de Descripción retorno
<pre>public void dibujarGrinch(Entorno e)</pre>	Dibuja el grinch en el entorno usando varios círculos concéntricos de colores rojo y negro para darle apariencia.
<pre>public void moverIzquierda()</pre>	Mueve el grinch hacia la izquierda según su velocidad, siempre que no salga del borde izquierdo (<code>x > 0</code>).

public boolean grinchMuerto()	boolean	Devuelve <code>true</code> si la vida del grinch es 0 o menor.
public boolean chocacConRegalo(Regalos r)	boolean	Comprueba si el grinch colisiona con un objeto <code>Regalos</code> . Calcula la distancia entre los centros y la compara con la suma de los radios (semilados) de ambos objetos.
public void recibirDanio(int danio)	void	Reduce la vida del grinch según el daño recibido.
public double getX()	double	Retorna la coordenada X del grinch.
public void setX(int x)	void	Modifica la coordenada X del grinch.
public double getY()	double	Retorna la coordenada Y del grinch.
public void setY(int y)	void	Modifica la coordenada Y del grinch.
public int getTamanio()	int	Retorna el tamaño visual del grinch.

Clase: *BarraSuperior*

La clase **BarraSuperior** representa una franja ubicada en la parte superior de la pantalla del juego.

Su función principal es dibujar una barra horizontal de color gris que sirve para mostrar la información, ya sea el tiempo transcurrido y cuantos zombies voy eliminando.

Variables de instancia

Nombre	Tipo	Descripción
<code>a</code>	<code>int</code>	Altura en píxeles de la barra superior. Determina el grosor de la franja que se dibuja en la parte superior de la ventana.

Constructores

La clase no define un constructor explícito, por lo tanto utiliza lo predeterminado. Este inicializa la variable `alturaBanda` con el valor **70**.

Métodos

Método	Tipo de retorno	Descripción
<code>public void dibujar(Entorno entorno)</code>	<code>void</code>	Dibuja la barra superior en la pantalla utilizando la librería <code>Entorno</code> . El rectángulo se centra horizontalmente en la ventana, ocupa todo el ancho de la pantalla y se dibuja en color gris.
<code>public int int getAlturaBanda()</code>	<code>int</code>	Devuelve la altura de la barra superior, permitiendo que otras clases conozcan el espacio que ocupa en la parte superior de la pantalla

Clase: Disparo

La clase **Disparo** representa un disparado dentro del juego.

Cada disparo tiene una posición, un tamaño, una velocidad, un color y un valor de daño.

Variables de instancia

Nombre	Tipo	Descripción
<i>x</i>	<i>double</i>	Coordenada X del centro del disparo en la ventana del juego.
<i>y</i>	<i>double</i>	Coordenada Y del centro del disparo en la ventana del juego.
<i>radio</i>	<i>int</i>	Radio del disparo (tamaño del círculo que lo representa).
<i>velocidad</i>	<i>double</i>	Velocidad con la que el disparo se desplaza hacia la derecha.
<i>activado</i>	<i>boolean</i>	Indica si el disparo está activo (true) o desactivado (false).
<i>color</i>	<i>Color</i>	Color con el que se dibuja el disparo.
<i>daño</i>	<i>int</i>	Cantidad de daño que el disparo provoca al impactar.

Constructores

```
public Disparo(double x, double y, int radio, double velocidad, Color color, int daño)
```

Inicializa un objeto **Disparo** con su posición, tamaño, velocidad, color y daño.

Parámetros:

- *x, y*: posición del disparo en la pantalla.
- *radio*: tamaño del disparo.
- *velocidad*: rapidez del movimiento horizontal.
- *color*: color del proyectil.
- *daño*: valor de daño que causa el disparo.

Métodos

Método	Tipo de Descripción retorno	
<pre>public void void <i>moverDerecha()</i></pre>		Mueve el disparo hacia la derecha sumando su velocidad a la coordenada x.
<pre>public void void <i>dibujar(Entorno e)</i></pre>		Dibuja el disparo en pantalla como un círculo amarillo si está activado.

<code>public void desactivar()</code>	<code>void</code>	Desactiva el disparo (deja de mostrarse en pantalla).
<code>public double getX()</code>	<code>double</code>	Devuelve la coordenada X actual del disparo.
<code>public double getY()</code>	<code>double</code>	Devuelve la coordenada Y actual del disparo.
<code>public int getRadio()</code>	<code>int</code>	Devuelve el radio del disparo.
<code>public boolean isActivado()</code>	<code>boolean</code>	Indica si el disparo está activado.
<code>public Color getColor()</code>	<code>Color</code>	Devuelve el color actual del disparo.
<code>public int getDanio()</code>	<code>int</code>	Devuelve el daño que el disparo infinge.

Clase: Juego

La clase **Juego** controla la lógica principal del juego, la creación del tablero, las plantas, los zombies (Grinch), los disparos, los regalos y la interfaz gráfica a través de **Entorno**. Se encarga de actualizar el estado del juego en cada instante mediante el método `tick()`.

Variables de instancia principales

Nombre	Tipo	Descripción
--------	------	-------------

<i>entorno</i>	<i>Entorno</i>	Objeto que maneja la ventana, gráficos y eventos del juego.
<i>fila</i>	<i>int</i>	Número de filas del tablero (5).
<i>columna</i>	<i>int</i>	Número de columnas del tablero (10).
<i>tablero</i>	<i>Casillero[][],</i>	Matriz de casilleros que representa el tablero de juego.
<i>zombieGrinch</i>	<i>Grinch[],</i>	Arreglo de zombies (Grinch) activos en el juego.
<i>plantas</i>	<i>Planta[][],</i>	Matriz que contiene las plantas colocadas en el tablero.
<i>regalos</i>	<i>Regalos[],</i>	Arreglo de regalos.
<i>disparos</i>	<i>Disparo[],</i>	Arreglo que guarda los disparos realizados por las plantas.
<i>barraSuperior</i>	<i>BarraSuperior</i>	Objeto que muestra información del juego .
<i>arrastrando</i>	<i>boolean</i>	Indica si el jugador está arrastrando una carta de planta.
<i>plantaPrevizualizada</i>	<i>Planta</i>	Planta que se muestra en vista previa mientras se arrastra.

<i>cartaRoseX, cartaRoseY, int</i>		Posición y tamaño de la carta de planta en la interfaz.
<i>juegoTerminado,</i> <i>juegoGanado</i>	<i>boolean</i>	Indican si el juego terminó y si fue ganado.
<i>tiempolnicio</i>	<i>long</i>	Marca el momento en que inicia el juego (ms).
<i>tiempo</i>	<i>int</i>	Contador de tiempo para generación de zombies.
<i>random</i>	<i>Random</i>	Generador de números aleatorios para lógica del juego.
<i>totalEnemigos,</i> <i>enemigosGenerados,</i> <i>maxZombiesSimultaneos,</i> <i>zombiesEliminados,</i> <i>zombiesRestantes</i>	<i>int</i>	Contadores de enemigos y control de la dificultad.
<i>minimoTiempoRegeneracion int</i> , <i>maximoTiempoRegeneracion</i>		Tiempo mínimo y máximo para generar un nuevo zombie.

Constructores

```
public Juego()
```

Inicializa el juego, creando:

- El tablero de casilleros con colores alternados.
- Los regalos por fila.

- Las matrices de plantas y disparos.
- Arreglo de zombies.
- Barra superior y temporizadores.
- Posición inicial de prueba de una planta.
- Finalmente, inicia la ejecución del juego con `entorno.iniciar()`.

Métodos principales

Método	Tipo de retorno	Descripción
<code>public void tick()</code>	<code>void</code>	Método central que se ejecuta cada frame. Actualiza el juego: dibuja tablero, plantas, zombies, disparos , maneja arrastre y colocación de plantas, controla colisiones y verifica victoria o derrota.
<code>public void disparar(double x, double y)</code>	<code>void</code>	Crea un nuevo disparo en las coordenadas X, Y si hay espacio en el arreglo de disparos.

<code>public int[] int[] coodenadasCasillero(int x, int y)</code>	Retorna la fila y columna del casillero que contiene las coordenadas X, Y. Devuelve null si no hay casillero.
<code>public void void deseleccionarTodas()</code>	Desmarca todas las plantas del tablero como no seleccionadas.
<code>public static void void main(String[] args)</code>	Inicia el juego creando un objeto Juego .

Problemas encontrados:

Como apartado final, explicaremos un poco de las cosas en las que tuvimos problemas:

Al principio tuvimos varios problemas mientras hacíamos el juego. Por ejemplo, con la clase Casillero no sabíamos muy bien cómo dibujar los casilleros en la ventana, así que al principio usábamos **System.out.println** y se imprimían todos los colores y tamaños en la consola muchas veces, en vez de mostrarse en el juego.

También nos costó entender cómo funcionaban algunos métodos y cómo actualizar todo en el entorno gráfico. A veces los objetos se dibujaban en lugares equivocados o no se actualizaban bien. Otro problema que nos dificultaba el avance en el trabajo práctico fue confundir los índices de filas y columnas, o no inicializar algún arreglo, lo que hacía que algunas plantas aparecieran fuera del tablero o no funcionaran.

Los disparos al principio tampoco detectaban bien a los enemigos, y los zombies a veces atravesaban todo sin recibir daño. La generación de enemigos y el control del tiempo también nos dio problemas al principio porque no aparecían como queríamos.

Problemas encontrados:

Cuando se realizó la clase casillero al principio en lugar de codificar bien en el entorno y dibujarlo en la ventana gráfica, se escribió en el método tick el comando sistema.out.println con los colores y demás características , entonces imprimía en consola las características de los casilleros en lugar de dibujarlos, ejemplo; salía impreso el color las cantidad de veces de filas y columnas.

Otra dificultad hallada fue que cuando intentábamos agregar la barra superior que indica el estado del juego cometimos el error de solo achicar el alto y ancho de las casillas hasta que nos quedo tan pequeño que los zombies quedaban fuera de los casilleros, lo resolvimos creando la clase barra superior.