Programación I - Primer Semestre 2020
Trabajo Práctico: Super Elizabeth Sis

La asociación *Global Animal Team Organization (GATO)* tiene como objetivo concientizar a niños y niñas acerca de la importancia del cuidado de nuestras mascotas. Para ello nos encargaron desarrollar un video juego similar al Super Mario Bros pero con algunas importantes diferencias que mencionamos a continuación.

El Juego: Super Elizabeth Sis

El malvado Rey Camir, al frente de la patrulla del maltrato animal, secuestró al gatito Carlos, la mascota de la princesa Elizabeth. La princesa Elizabeth transformada ahora en nuestra heroína se lanza a combatir a los soldados de la patrulla del maltrato animal para poder rescatar al gatito Carlos (ver Figura 1).

A medida que la princesa Elizabeth avanza se va encontrando con distintos obstáculos que tiene que evitar y enemigos que tiene que eliminar. Cabe aclarar que en nuestro juego para simular el avance de la princesa, los obstáculos y los enemigos se desplazan hacia la izquierda de la pantalla creando la ilusión de que la princesa está avanzando. Es decir, la princesa avanza "automáticamente" a medida que los obstáculos y los enemigos que se mueven hacia la izquierda.

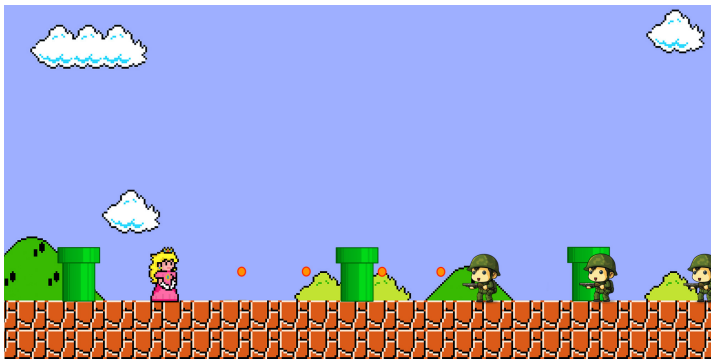

Figura 1: Ejemplo del juego.

Los soldados de la patrulla del maltrato animal van apareciendo desde la derecha de pantalla y nuestra heroína va acumulando puntos a medida que los elimina.

El objetivo del juego es conseguir el mayor puntaje posible evitando chocarse con los obstáculos y con los soldados. El juego termina cuando la princesa pierde sus tres vidas, por haber chocado contra los obstáculos o con los soldados.

Requerimientos obligatorios

Los requerimientos obligatorios son los siguientes:

1. Al iniciar el juego, la princesa Elizabeth debe aparecer en el sector de la pantalla de la mitad hacia la izquierda (ver Figura 1).

2. Mientras la princesa esté en sobre el piso: Si se presiona la fecha hacia atrás, la princesa puede retroceder. Si se presiona la fecha hacia adelante, la princesa puede avanzar. La princesa siempre se debe mantener en el sector izquierdo de la pantalla.

3. A medida que avanza el juego deberán aparecer desde el lado derecho de la pantalla los obstáculos que la princesa tendrá que ir saltando.

   La cantidad y distribución de los obstáculos queda a criterio de cada grupo aunque se sugiere que sean entre tres y cinco. Además deben tener cuidado de que no queden demasiado juntos de manera que sea difícil saltarlos ni tampoco demasiado separados que no presente ninguna dificultad.

4. Todos los obstáculos se desplazan constantemente hacia la izquierda y siempre con la misma velocidad.

   Cuando un obstáculo llega al borde izquierdo de la pantalla debe reaparecer en el borde derecho.

5. Durante el transcurso del juego, aleatoriamente, deben aparecer desde el extremo derecho de la pantalla, los soldados de la patrulla del maltrato animal, que irán desplazándose hacia la izquierda de la pantalla para tratar de alcanzar a la princesa Elizabeth.

   Los soldados no se pueden superponer entre ellos, pero sí pueden pasar por delante de los obstáculos.

6. Si se presiona la fecha hacia arriba, la princesa debe saltar. De esta manera podrá pasar por encima de los obstáculos y de los soldados.

7. Si la princesa toca un obstáculo o un soldado pierde una vida.

8. La princesa tiene una habilidad secreta: puede lanzar fuego! Cuando se presione la barra espaciadora, la princesa debe lanzar una pequeña bola de fuego.

9. Cuando la bola de fuego toque a un soldado éste muere y en ese momento: debe desaparecer de la pantalla la bola de fuego y el soldado. Por cada soldado que la princesa elimine se incrementa en 5 su puntaje.

10. Durante todo el juego deberá mostrarse en algún lugar de la pantalla el puntaje acumulando (inicialmente 0) y la cantidad de vidas que tenemos (inicialmente 3).

11. Si la princesa pierde las tres vidas, perdemos y se termina el juego.

12. El código del proyecto **deberá tener un buen diseño** de modo que cada objeto tenga **bien delimitadas sus responsabilidades**.

La implementación a entregar debe cumplir como mínimo con todos los requerimientos obligatorios planteados arriba, pero si el grupo lo desea, puede implementar nuevos elementos para enriquecer la aplicación. Sugerimos a continuación algunas ideas:

- Los soldados no pueden pasar por delante de los obstáculos, deberán saltarlos o bien rebotar cuando toquen un obstáculo.
- Los obstáculos puedan crecer y achicarse verticalmente de manera que la princesa deba calcular con mayor precisión sus saltos.
- Agregar items que otorguen puntos extras si la princesa los toca.
- Ganar el juego: Al pasar cierto tiempo o al matar una cantidad determinada de soldados o conseguir un cierto puntaje, que el juego se dé por ganado mostrando al rey Camir derrotado y al gatito Carlos siendo rescatado por la princesa Elizabeth.
- Muchos niveles: La posibilidad de comenzar un nivel nuevo después de ganar el juego, por ejemplo al matar una cantidad determinada de soldados. Incluso se podría incrementar la dificultad y/o velocidad de cada nivel.

Sobre el informe a desarrollar y las condiciones de entrega

Además del código, la entrega debe incluir un documento en el que se describa el trabajo realizado, que debe incluir las siguientes secciones:

- Un encabezado del documento con los nombres, números de legajo y dirección de email de cada integrante del grupo.
- Una introducción describiendo el trabajo realizado.
- Una explicación de cada clase implementada describiendo las variables de instancia y dando una breve descripción de cada método. Se pide además el invariante de representación para cada clase (salvo para la clase `Juego`).
- Un resumen de los problemas encontrados y de las decisiones de implementación que les permitieron resolverlos.
- Un apéndice con el código (se pueden omitir las partes del código que no se consideren relevantes para el trabajo).

Tanto este informe como el código fuente del trabajo práctico deben ser enviados a los docentes de la materia. El trabajo práctico se debe hacer en grupos de **exactamente tres** personas.

Fecha de entrega: Verificar en el moodle de la comisión correspondiente.

Implementación base

Junto con este enunciado, se les entrega el paquete `entorno.jar` que contiene la clase `Entorno`. Esta clase permite crear un objeto capaz de encargarse de la interfaz gráfica y de la interacción con el usuario. Así, el grupo sólo tendrá que encargarse de la implementación de la inteligencia del juego. Para ello, se deberá crear una clase llamada `Juego` que respete la siguiente signatura[1]:

```
public class Juego extends InterfaceJuego {
    private Entorno entorno;

    // Variables y métodos propios de cada grupo
    // ...

    public Juego() {
        // Inicializa el objeto entorno
        entorno = new Entorno(this, "Super Elizabeth Sis - Grupo 3 - v1", 800, 600);

        // Inicializar lo que haga falta para el juego
        // ...

        // Inicia el juego
        entorno.iniciar();
    }

    public void tick() {
        // Procesamiento de un instante de tiempo
        // ...
    }

    public static void main(String[] args) {
       Juego juego = new Juego();
    }

}
```

El objeto `entorno` creado en el constructor del `Juego` recibe el juego en cuestión y mediante el método entorno.iniciar() se inicia el simulador. A partir de ahí, en cada instante de tiempo que pasa, el entorno ejecuta el método `tick()` del juego. Éste es el método más importante de la clase `Juego` y aquí el juego debe actualizar su estado interno para simular el paso del tiempo. Como mínimo se deben realizar las siguientes tareas:

- Actualizar el estado interno de todos los objetos involucrados en la simulación.
- Dibujar los mismos en la pantalla (ver más abajo cómo hacer esto).
- Verificar si algún objeto aparece o desaparece del juego.

[1] **Importante:** las palabras clave "`extends InterfaceJuego`" en la definición de la clase son fundamentales para el buen funcionamiento del juego.

- Verificar si hay objetos interactuando entre sí (colisiones por ejemplo).
- Verificar si los usuarios están presionando alguna tecla y actuar en consecuencia (ver más abajo cómo hacer esto).

Para dibujar en pantalla y capturar las teclas presionadas, el objeto `entorno` dispone de los siguientes métodos, entre otros:

`void dibujarRectangulo(double x, double y, double ancho, double alto, double angulo, Color color)`
↪ Dibuja un rectángulo *centrado* en el punto (`x`,`y`) de la pantalla, rotado en el ángulo dado.

`void dibujarTriangulo(double x, double y, int altura, int base, double angulo, Color color)`
↪ Dibuja un triángulo *centrado* en el punto (`x`,`y`) de la pantalla, rotado en el ángulo dado.

`void dibujarCirculo(double x, double y, double diametro, Color color)`
↪ Dibuja un círculo *centrado* en el punto (`x`,`y`) de la pantalla, del tamaño dado.

`void dibujarImagen(Image imagen, double x, double y, double ang)`
↪ Dibuja la `imagen` *centrada* en el punto (`x`,`y`) de la pantalla rotada en el ángulo dado.

`boolean estaPresionada(char t)`
↪ Indica si la tecla `t` está presionada por el usuario en ese momento.

`boolean sePresiono(char t)`
↪ Indica si la tecla `t` fué presionada en este instante de tiempo (es decir, no estaba presionada en la última llamada a `tick()`, pero sí en ésta). Este método puede ser útil para identificar eventos particulares en un único momento, omitiendo `tick()` futuros en los cuales el usuario mantenga presionada la tecla en cuestión.

`void escribirTexto(String texto, double x, double y)`
↪ Escribe el `texto` en las coordenadas `x` e `y` de la pantalla.

`void cambiarFont(String font, int tamano, Color color)`
↪ Cambia la fuente para las próximas escrituras de texto según los parámetros recibidos.

Notar que los métodos `estaPresionada(char t)` y `sePresiono(char t)` reciben como parámetro un `char` que representa "la tecla" por la cual se quiere consultar, e.g., `sePresiono('A')` o `estaPresionada('+')`. Algunas teclas no pueden escribirse directamente como un `char` como por ejemplo las flechas de dirección del teclado. Para ellas, dentro de la clase entorno se encuentran las siguientes definiciones:

Valor	Tecla Representada
`TECLA_ARRIBA`	Flecha hacia arriba
`TECLA_ABAJO`	Flecha hacia abajo
`TECLA_DERECHA`	Flecha hacia la derecha
`TECLA_IZQUIERDA`	Flecha hacia la izquierda
`TECLA_ENTER`	Tecla “enter”
`TECLA_ESPACIO`	Barra espaciadora
`TECLA_CTRL`	Tecla “control”
`TECLA_ALT`	Tecla “alt”
`TECLA_SHIFT`	Tecla “shift”
`TECLA_INSERT`	Tecla “ins”
`TECLA_DELETE`	Tecla “del” (o “supr”)
`TECLA_INICIO`	Tecla “start” (o “inicio”)
`TECLA_FIN`	Tecla “end” (o “fin”)

De esta manera, para ver, por ejemplo, si el usuario está presionando la flecha hacia arriba se puede consultar, por ejemplo, si `estaPresionada(entorno.TECLA_ARRIBA)`.