Programación I - Primer Semestre 2024
Trabajo Práctico: Super Elizabeth Sis, Volcano Edition

El malvado Rey Camir vuelve a hacer de las suyas, quien sigue a cargo de la patrulla de maltrato animal, esta vez secuestró a la mascota de la princesa Elizabeth. Ella será nuestra heroína en esta nueva aventura.

El malvado Rey Camir estuvo experimentando con fósiles de dinosaurios que ahora son monstruos resistentes a las altas temperaturas volcánicas. Nuestra heroína debe ir a rescatar a su gatito que está en la cima del volcán y escapar de los enemigos que quieran impedirlo.

El objetivo de este trabajo práctico es desarrollar un video juego en el cual la Princesa Elizabeth vaya subiendo por los distintos niveles del volcán hasta llegar a la cima. La princesa deberá romper y subir por los distintos bloques de ladrillos hasta encontrar la salida, además deberá eliminar los enemigos que van apareciendo.

Figura 1: Ejemplo del juego.

El Juego: Super Elizabeth Sis, Volcano Edition

Requerimientos obligatorios

1. En pantalla deberá verse el interior de un volcán, con una serie de pisos o filas de bloques. Cada piso debe tener al menos un bloque que pueda ser destruido cuando la princesa Elizabeth salte desde el piso inferior. La cantidad de pisos en la pantalla queda a criterio de cada grupo, aunque deben ser no menos de 4 pisos y no más de 6 pisos.

2. Al iniciar el juego, la princesa Elizabeth debe aparecer en el centro de la fila inferior de bloques (ver Figura 1).

3. Mientras la princesa esté parada sobre una fila de bloques podrá: moverse hacia la izquierda o hacia la derecha presionando las teclas izquierda o derecha respectivamente, y podrá saltar si se presiona la tecla 'x'. Si no se presiona ninguna tecla direccional, la princesa debe quedarse quieta. Si la princesa no está parada sobre una fila de bloques debe caer en forma vertical hasta el siguiente fila de bloques.

4. Cuando la princesa salte podrá destruir/romper el bloque superior con el que colisionó, haciendo que éste desaparezca, es decir el objeto quede nulo (null). De esa manera, en un próximo salto podrá subir de un salto a la fila de arriba. Cada fila de bloques deberá ocupar el ancho de la pantalla y debe tener al menos un bloque que se rompa fácilmente con golpe de la princesa y otros bloques de acero que no pueden ser destruidos con nada. Ambos tipos de bloques deben poder distinguirse.

5. Cada fila de bloques debe tener inicialmente 2 Tiranosaurios Rex mutantes caminando sobre la misma. Ellos se podrán mover de izquierda a derecha y viceversa, y deben cambiar de dirección al llegar a la pantalla. Si llegan a un hueco en la fila de bloques, deben caer de manera vertical y seguir caminando por la fila de abajo en la misma dirección en la que venían. Los tiranosaurios no se pueden superponer (no puede ir caminando un tiranosaurio encima de otro), aunque sí pueden cruzarse por el mismo piso. Los tiranosaurios deben ir apareciendo en posiciones aleatorias de manera que siempre haya al menos 2 en la pantalla.

6. La princesa tiene una habilidad secreta: puede lanzar su disparo cuando se presione la tecla 'c'. El disparo debe salir en la dirección donde está caminando o caminaba la princesa. Cuando el disparo hace contacto con un Tiranosaurio Rex, lo mata y debe desaparecer tanto el disparo como el enemigo eliminado. La princesa solo puede realizar un disparo por vez.

7. Los Tiranosaurios Rex tienen una mutación que les permite lanzar una bomba como proyectil. Pueden lanzar sólo una a la vez, que se moverá hacia la dirección donde está caminando. Si la bomba hace colisión con la princesa, perdemos el juego. Si la bomba sale de pantalla o colisiona con un disparo de la princesa, la bomba es eliminada. Los proyectiles lanzados pueden atravesar otras bombas y a otros Tiranosaurios Rex.

8. Cuando la princesa haga colisión con un Tiranosaurio Rex o una bomba perdemos el juego. Cuando la princesa salga por el sector superior de la pantalla ganamos el juego. El salto de la princesa debe permitirle esquivar las bombas de los tiranosaurios.

9. **Aclaramos que cuando un Tiranosaurio Rex, bomba, disparo o bloque de roca ya no esta más en la pantalla, los objetos deben ser eliminados, no vale únicamente «ocultar» las imágenes del juego.**

10. Durante todo el juego deberá mostrarse en algún lugar de la pantalla: la cantidad de enemigos eliminados y el puntaje obtenido. Por cada enemigo eliminado se gana 2 puntos.

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

Requerimientos opcionales

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:

- Magma: Mostrar el magma del volcán subiendo constantemente, si llegara a tener contacto con la Princesa perdemos el juego.
- Pisos extras: cuando se llegue a la fila de más arriba, que toda la pantalla haga un desplazamiento hacia abajo para que puedan visualizarse más pisos del volcán.
- Dos jugadores: Permitir el juego para dos jugadores para que trabajen en equipo y puedan subir.
- Niveles: La posibilidad de comenzar un nuevo nivel después de jugar determinada cantidad de tiempo o de puntaje acumulados, e incrementar la dificultad y/o velocidad de cada nivel.
- Que aparezca un un TiranoRex mutante colosal como jefe final o de nivel, podrá ocupar dos filas de bloques
- Vidas: cada vez que muera la princesa se le reste una vida
- Item: los bloques podrán devolver ítems que den resistencia a la princesa o vidas extras.

Informe solicitado

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

Encabezado Una carátula del documento con nombres, apellidos, legajos, y direcciones de email de cada integrante del grupo.

Introducción Una breve introducción describiendo el trabajo práctico.

Descripción Una explicación general de cada clase implementada, describiendo las variables de instancia y dando una breve descripción de cada método.

También deben incluirse los problemas encontrados, las decisiones que tomaron para poder resolverlos, y las soluciones encontradas.

Implementación Una sección de implementación donde se incluya el código fuente correctamente formateado y comentado, si corresponde.

Conclusiones Algunas reflexiones acerca del desarrollo del trabajo realizado, y de los resultados obtenidos. Pueden incluirse lecciones aprendidas durante el desarrollo del trabajo.

Condiciones de entrega

El trabajo práctico se debe hacer en grupos de **exactamente tres** personas.

El nombre del proyecto de Eclipse debe tener el siguiente formato: los apellidos en minúsculas de los tres integrantes, separados con guiones, seguidos de `-tp-p1`.
Por ejemplo, `amaya-benelli-castro-tp-p1`.

Cada grupo deberá entregar tanto el informe como el código fuente del trabajo práctico.

Consultar la modalidad de entrega de cada comisión (email, google drive, git, etc).

- Si la modalidad de entrega es mediante email: especificar en el asunto del email, los apellidos en minúsculas de los tres integrantes. En el cuerpo del email colocar, nombres, apellidos, legajos, y direcciones de email de cada integrante del equipo. Adjuntar al email tanto el informe como el código fuente del trabajo práctico.
- Si la modalidad de entrega es mediante repositorio en Gitlab: asegurarse de que el repositorio sea privado y que el nombre del proyecto de Gitlab tenga los apellidos en minúsculas de los tres integrantes, separados con guiones, seguidos del string `-tp-p1`. Por ejemplo, `amaya-benelli-castro-tp-p1`. Consultar el username de cada docente y agregar a los docentes como `maintainer`'s. El informe del trabajo práctico puede incluirse directamente en el repositorio.

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

Apéndice: 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 metodos 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` fue presionada en este instante de tiempo (es decir, no estaba presionada en la última llamada a `tick()`, pero sá 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)`.