# TP Programación 1 COM-04

## Super-Elizabeth


- Grupo: 4
- Integrantes:
  Bustamante Andrés,
  Cyrulies Nicolas,
  Gonzalez Santiago,
  Marciano Leandro.

# Contenido

# Introducción

El presente informe describe el desarrollo del juego de plataforma "Super Elizabeth", realizado en el lenguaje de programación Java como trabajo práctico final de la materia Programación I, de la Universidad Nacional de General Sarmiento. El juego consiste en una princesa que debe escapar de un volcán esquivando y matando dinosaurios mientras sube por una serie de plataformas hasta el piso superior. A medida que la princesa elimina un dinosaurio, se ganan puntos, y si un dinosaurio toca o tira una bomba a la princesa, se pierde el juego.

# Descripción

## Clases Implementadas:

Juego: Clase principal del juego. Se encarga de controlar el flujo general, como la inicialización de elementos, la actualización del estado del juego y la representación gráfica.

Entorno: Clase brindada por los docentes. Proporciona las herramientas para la interacción con el usuario, como la detección de teclas presionadas y el dibujo de elementos en la pantalla.

Princesa: Clase que representa al personaje principal del juego, controlando su movimiento, salto, disparo y estado de vida.

DisparoPrincesa: Clase que controla los disparos realizados por la princesa, controlando su trayectoria y colisiones.

Dinosaurio: Clase que representa a los enemigos del juego, controlando su movimiento, comportamiento y colisiones.

Fondo: Clase que introduce el fondo del juego, proporcionando una imagen estática de fondo.

Menu: Clase que representa el menú principal del juego, permitiendo al jugador iniciar el juego.

MenuPerder: Clase que representa el menú que se muestra cuando la princesa pierde el juego, mostrando el puntaje y las bajas de dinosaurios.

MenuGanar: Clase que representa el menú que se muestra cuando la princesa gana el juego, mostrando el puntaje y las bajas de dinosaurios.

Bloque: Clase objeto de los bloques del juego. Controla su posición, tipo y colisiones.

### Métodos:

tick(): Método principal del juego que se ejecuta en cada frame, actualizando el estado del juego y dibujando los elementos en la pantalla.

movimientosPrincesa(): Método que controla el movimiento de la princesa en base a las teclas presionadas.

detectarColisiones(): Método que detecta las colisiones entre la princesa, los dinosaurios y los bloques.

dibujarPisoSolido(): Método que dibuja los bloques del piso.

inicializarBloques(): Método que inicializa los bloques del juego en una disposición aleatoria.

inicializarDinosaurios(): Método que inicializa los dinosaurios del juego en una distribución aleatoria. Al iniciar el juego, dos por nivel, exceptuando el inferior.

incrementarPuntaje(): Método que incrementa el puntaje del juego.

## Implementación

El código fuente del juego se encuentra adjunto al final de este informe. El código está correctamente formateado y comentado para facilitar su comprensión. A continuación, se describirán los problemas, lecciones y conclusiones que se desarrollaron en el desarrollo del proyecto.

### Problemas encontrados y soluciones:

Se encontró un problema con la detección de colisiones entre la princesa y los dinosaurios, que no funcionaba correctamente. Se solucionó el problema

redefiniendo el método colisiona() para que considerara el tamaño y la posición de ambos objetos de manera más precisa.

Se encontró un problema con el movimiento de los dinosaurios, que no se movían de manera fluida. Se solucionó el problema ajustando la velocidad de movimiento de los dinosaurios.

Tuvimos inconvenientes con la gravedad, ya que cuando un personaje bajaba de nivel, caía por fuera de la pantalla. Se solucionó con un bucle que se repite hasta detectar una colisión con el bloque inferior.

## Las lecciones aprendidas durante el desarrollo del juego son las siguientes:

La importancia de la planificación y el diseño del código antes de comenzar a programar.

La necesidad de realizar pruebas exhaustivas para detectar y corregir errores.

La importancia de trabajar en equipo para lograr un objetivo común y como cada persona puede aportar algo distinto desde sus conocimientos, haciendo el resultado final mucho más enriquecedor.

## Conclusiones

El desarrollo del juego "Super Elizabeth" ha sido una experiencia de aprendizaje enriquecedora. Se ha aprendido a trabajar con diferentes clases y objetos en Java, y como estas interactúan entre sí. Se aprendió a manejar eventos de usuario, a detectar colisiones y a crear una interfaz gráfica atractiva. Nos sirvió a trabajar en equipo, ya que cuando surgía una duda por un integrante del grupo, cualquier otro integrante estaba para proporcionar asistencia y una explicación. Esto nos brindó un resultado super gratificante y un aprendizaje compartido.

El juego finalizado es funcional y entretenido, y cumple con los objetivos iniciales del grupo Consideramos que el desarrollo del juego y del proyecto ha sido un éxito.

# Anexos

## Clase principal “Juego”:

```
public class Juego extends InterfaceJuego {
        // El objeto Entorno que controla el tiempo y otros
        private Entorno entorno;
        private int ticksCounter = 0;
        private Princesa princesa;
        private DisparoPrincesa disparoPrincesa;
 private List<Dinosaurio> dinosaurios;
        private Fondo fondo;
        private Menu menu;
        private MenuPerder MenuPerder;
        private MenuGanar MenuGanar;
        private int puntaje;
        private int muertes;
 private boolean enJuego;
 public int tipoMenu = 0;
 private Clip musicaJuego = Herramientas.cargarSonido("sounds/musica-juego.wav");
        private Bloque[] bloques = new Bloque[600];
        int ANCHO_DE_BLOQUES = 27;
        int SEPARACION = 150;
        // Variables y métodos propios de cada grupo
        // ...

        Juego() {
                Random rand = new Random();
                this.entorno = new Entorno(this, " Super Elizabeth Sis, Volcano Edition - Grupo ... -
v1", 800, 600);
```

```
                this.puntaje = 0;

                //Carga menu
menu = new Menu();
MenuPerder = new MenuPerder();
                MenuGanar = new MenuGanar();
enJuego = false;


                inicializarBloques(570,27,false);

                fondo = new Fondo(400,300);
                princesa = new Princesa(400, 535, this);
                dinosaurios = new ArrayList<>();

inicializarDinosaurios();

                this.entorno.iniciar();
        }


        public void tick() {

                if (!enJuego) {
          if (tipoMenu == 0) {
           menu.dibujar(entorno);
           menu.play(entorno);
           if (menu.isJuegoIniciado()) {
            enJuego = true;
            musicaJuego.start();
           }
          } else if (tipoMenu == 1) {
           MenuPerder.dibujar(entorno, puntaje, muertes);
          } else if (tipoMenu == 2) {
           MenuGanar.dibujar(entorno, puntaje, muertes);
```

```
            }
        } else {

                    ticksCounter += 2;
                    fondo.dibujarse(entorno);
                    entorno.cambiarFont("Arial", 20, java.awt.Color.WHITE);
            entorno.escribirTexto("Puntaje: " + puntaje, entorno.ancho() / 4 - 180, entorno.alto() -
575);
            entorno.escribirTexto("Dinosaurios muertos: " + muertes, entorno.ancho() / 4 - 180,
entorno.alto() - 550);

                    if (princesa.estaViva()) {
                        princesa.dibujarse(entorno, ticksCounter);
                        movimientosPrincesa();
                        princesa.aplicarGravedad(bloques);
                    }
                    dibujarPisoSolido();

                    for (Dinosaurio dinosaurio : dinosaurios) {
                        dinosaurio.dibujarse(entorno, ticksCounter);
                        dinosaurio.aplicarGravedad(bloques);
                    }

                    if (princesa.estaViva()) {
                        princesa.dibujarDisparos(entorno, dinosaurios);
                    }

                    detectarColisiones();
                }
            }

            private void movimientosPrincesa() {
                if (entorno.estaPresionada(entorno.TECLA_DERECHA) && princesa.getX() < 780){
                  princesa.moverDerecha();
                }
```

```
                if (entorno.estaPresionada(entorno.TECLA_IZQUIERDA) && princesa.getX() > 20) {
                  princesa.moverIzquierda();
                }

                if (!entorno.estaPresionada(entorno.TECLA_IZQUIERDA) &&
!entorno.estaPresionada(entorno.TECLA_DERECHA)) {
                        princesa.estaQuieta();
                }

                if (entorno.estaPresionada(entorno.TECLA_ARRIBA)) {
                        princesa.saltar();
                }else{
                        princesa.liberarTeclaSalto();
                }
                if (entorno.sePresiono(entorno.TECLA_ESPACIO)) {
                        princesa.disparar(entorno);

                }

                if (!entorno.estaPresionada(entorno.TECLA_ESPACIO)) {
                        princesa.resetDisparo(); // Permitir disparar de nuevo al soltar la tecla
                }

        }

        private void detectarColisiones() {
                for (int i = 0; i < bloques.length; i++) {
                        Bloque bloque = bloques[i];
                        if (bloque != null && colisiona(princesa, bloque)) {
                                if (princesa.estaSaltando()) {
                                        if (bloque.esRompible()) {
                                                bloques[i] = null; // Destruir el bloque rompible
                                                princesa.aterrizar(bloque.getY() - 30); // Ajustar
posición de la princesa
```

```
                                        } else if (princesa.getY() < bloque.getY()) {
                                                princesa.aterrizar(bloque.getY() - 30); // Ajustar
posición de la princesa
                                        } else {
                                                princesa.detenerSalto(bloque.getY()); // Detener
el salto si colisiona desde abajo
                                        }
                                }
                        }
                }

                for (Dinosaurio dinosaurio : dinosaurios) {
                        if (colisiona(princesa, dinosaurio)) {
                                princesa.muerta();
                                tipoMenu =1;
                                enJuego = false;
                                break;
                        }

                }
        }



        private boolean colisiona(Princesa princesa, Bloque bloque) {
           return princesa.getX() < bloque.getX() + 30 &&
              princesa.getX() + 30 > bloque.getX() &&
              princesa.getY() < bloque.getY() + 30 &&
              princesa.getY() + 30 > bloque.getY();
         }
        private boolean colisiona(Princesa princesa, Dinosaurio dinosaurio) {
                return princesa.getX() < dinosaurio.getX() + 30 &&
                                princesa.getX() + 30 > dinosaurio.getX() &&
                                princesa.getY() < dinosaurio.getY() + 30 &&
                                princesa.getY() + 30 > dinosaurio.getY();
```

```
    }

    private void dibujarPisoSolido() {
      for (Bloque bloque : bloques) {
        if (bloque != null) {
          bloque.dibujarse(entorno);
        }
      }
    }

    private void inicializarBloques(int altura, int cantidad, boolean rompible) {
      int x = 10;
      int numFilas = 4;
      Random r = new Random();

      for (int fila = 0; fila < numFilas; fila++) {
        int posicionInicio = r.nextInt(ANCHO_DE_BLOQUES - 2); // Posición de inicio para los 3
bloques consecutivos

        for (int i = 0; i < ANCHO_DE_BLOQUES; i++) {
          int index = fila * ANCHO_DE_BLOQUES + i;
          int alturaBloque = altura - (SEPARACION * fila);
          boolean esRompible = (i >= posicionInicio && i < posicionInicio + 3);
          bloques[index] = new Bloque(x, alturaBloque, esRompible);
          x += 30;
        }
        x = 10;
      }
    }

    private void inicializarDinosaurios() {
```

```
                // Distribuye 2 dinosaurios por piso en los pisos superiores (excepto el piso
inferior)
                Random rand = new Random();
            int numFilas = 4;
            int dinosPorPiso = 2;
            int alturaInicial = 570;
            int anchoMaximo = 780;
            int margen = 20;

            for (int fila = 1; fila < numFilas; fila++) {
             int altura = alturaInicial - (SEPARACION * fila);
             for (int i = 0; i < dinosPorPiso; i++) {
               int xPos = rand.nextInt(anchoMaximo - 2 * margen) + margen;
               dinosaurios.add(new Dinosaurio(xPos, altura - 30));
             }
            }
           }

           public void incrementarPuntaje() {
            puntaje+=2;
            muertes++;
           }

          @SuppressWarnings("unused")
          public static void main(String[] args) {
                Juego juego = new Juego();
          }
}
```