INFORME PROYECTO PROGRAMACION I

Super Elizabeth Sis, Volcano Edition

Integrantes:

Nombre: Jorge Lizarraga

Legajo: 37941969

Email: jorgelizarraga1994@gmail.com

Nombre: Alex Zelaya

Legajo: 40623214

Email: zelaya-alex@outlook.com

Docentes:

- Rodrigo González
- Sabrina Castro
- Lucas Bidart Gauna

Año: 2024

Introducción:

El presente proyecto se centró en la creación de un video juego llamado Super Elizabeth Sis, Volcano Edition, el cual se centra en la travesia que tiene atravezar la princesa elizabeth para rescatar a su mascota del malvado rey camir y sus dinosaurios mutantes. Para que la princesa logre su objetivo tiene que recorrer 4 pisos repletos de dinosaurios que lanzan bombas hasta llegar a la cima donde se encuentra su mascota prisionera.

El objetivo del desarrollo del videojuego es que la princesa pueda superar todos los obstaculos propuestos, destruir a los dinosaurios sin perder la vida en el intento.

El juego esta desarrollado en el lenguaje de alto nivel java y utiliza la biblioteca Entorno para crear una interfaz grafica,a si mismo en este informe, se analizarán las clases y las variables utilizadas, se presentará la implementación del juego y se ofrecerán algunas reflexiones sobre el proyecto.

Descripción:

Clase Bloques:

Esta clase está conformada por todos los bloques (utilizamos el objeto rectangle para construirlo) duros(irrompibles) y blandos (rompibles por la princesa) y el conjunto de estos forma los 4 pisos donde se encuentra la princesa, los dinosaurios y la mascota.

variables de instancia:

```
	//coordenadas de posicion de bloques
	private double x;
	private double y;
	//dimensión de bloques
	private int ancho;
	private int alto;
	//booleano para diferenciar a los bloques rompibles de los no rompibles
	private boolean seRompe;
	private double angulo;
	private double escala;
	//imagenes de los bloques
	private Image bloqueSeRompe = Herramientas.cargarImagen("bloqueSeRompe.png");
	private Image bloqueNoSeRompe = Herramientas.cargarImagen("bloqueNoSeRompe.png");
```

Constructor de clase bloque:

```
public Bloques(int x, int y, int ancho, int alto , boolean seRompe ){
    this.x = x;
    this.y = y;
    this.ancho = ancho;
    this.alto = alto;
    this.angulo = 0;
    this.escala = 3;
    this.seRompe = seRompe;
}
```

Metodos:

CrearPiso: este metodo itera y crea de forma aleatoria bloques duros y blandos de forma secuencial en cada uno de los pisos, utilizamos una variable que comienza en 25 para evitar que una mitad del primer bloque izquierdo se inicialice fuera de la pantalla.

dibujar: este metodo utiliza una herramienta provista por el entorno que dibuja, toma una imagen (previamente cargada en eclipse) y la crea en el entorno del juego.

```
//----------Recibe un array de bloques y sus atributos y crea el piso---------------//
public void crearPiso(ArrayList<Bloques> todosLosPisos, Bloques [] conjuntoBloques , int y , int ancho, int alto) {
    int suma = 25;
    Random aleatorio = new Random();
    for (int i = 0; i < conjuntoBloques.length; i++) {
        if((aleatorio.nextInt(3)) % 6 == 0 || i == 3 ) {
            conjuntoBloques[i] = new Bloques(suma , y , ancho, alto , true);
            todosLosPisos.add(conjuntoBloques[i]);
            suma += 50;
        }
        else {
            conjuntoBloques[i] = new Bloques(suma , y , ancho, alto , false);
            todosLosPisos.add(conjuntoBloques[i]);
            suma += 50;
        }
    }
}
//----------Recibe un array de bloques y los dibuja---------------//
public void dibujar(Entorno entorno , ArrayList<Bloques> todosLosPisos) {
    Color color;
    for (int i = 0; i < todosLosPisos.size(); i++) {
        Bloques bloque = todosLosPisos.get(i);
        if (bloque == null) {
            continue;
        }
        if(bloque.seRompe) {
            //color = Color.red;
            //entorno.dibujarRectangulo(bloque.x, bloque.y, bloque.ancho, bloque.alto, 0, color);
            entorno.dibujarImagen(bloqueSeRompe, bloque.x, bloque.y, this.angulo, this.escala);
        }
        else {
            //color = Color.BLUE;
            //entorno.dibujarRectangulo(bloque.x, bloque.y, bloque.ancho, bloque.alto, 0, color);
            entorno.dibujarImagen(bloqueNoSeRompe, bloque.x, bloque.y, this.angulo, this.escala);
        }
    }
}
```

Getters:

```
//----------Getters y Setter----------//
public double getX() {
    return x;
}

public double getY() {
    return y;
}

public int getAncho() {
    return ancho;
}

public int getAlto() {
    return alto;
}

public boolean GetseRompe() {
    return seRompe;
}
```

Clase Bomba:

En esta clase se diseñan las bombas que utilizan los dinosaurios mutantes y la direccion(izquierda o derecha) del mismo.

Variables de instancia:

```java
    //posicion de la bomba en pantalla
    private double x;
    private double y;
    //dimensiones constantes de la bomba
    private int ANCHO = 10;
    private int ALTO = 10;
    //Imagen de la bomba
    private Image imagen;
```

Constructor de la bomba: contiene la posicion, las dimenciones y la imagen de la bomba del dinosaurio.

```java
//----------------Constructores----------------//
public Bomba(int x, int y, int ANCHO, int ALTO) {
    this.x = x;
    this.y = y;
    this.ANCHO = ANCHO;
    this.ALTO = ALTO;
    this.imagen = Herramientas.cargarImagen("bombaTiranosaurio.png");
}
```

Getters y Setters

```java
}
//---------------Getters y Setters----------------//
public double getX() {
    return x;
}

public void setX(int x) {
    this.x = x;
}

public double getY() {
    return y;
}

public void setY(int y) {
    this.y = y;
}

public int getANCHO() {
    return ANCHO;
}

public void setANCHO(int ANCHO) {
    this.ANCHO = ANCHO;
}

public int getALTO() {
    return ALTO;
}

public void setALTO(int ALTO) {
    this.ALTO = ALTO;
}
```

Metodos:

disparoDer: este metodo indica que la bomba tiene que dirigir su movimiento hacia la derecha

disparoIzq: este metodo indica que la bomba tiene que dirigir su movimiento hacia la izquierda

```java
public void disparoDer(Entorno entorno ,int velocidadDisparo) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ANCHO, this.ALTO, 0, Color.blue);
    double bala =  this.x += velocidadDisparo;
    this.x = bala;
    entorno.dibujarImagen(imagen, this.x, this.y, 0, 0.09);
}

public void disparoIzq(Entorno entorno , int velocidadDisparo) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ANCHO, this.ALTO, 0, Color.red);
    double bala =  this.x -= velocidadDisparo;
    this.x = bala;
    entorno.dibujarImagen(imagen, this.x, this.y, 0, 0.09);
}
```

Clase Princesa:

Esta es la clase de la protagonista en el cual se le asigna todos los atributos necesarios para poder atravesar todos los obstaculos creados en el entorno , entre los cuales estan la velocidad de movimiento, la gravedad, la habilidad de disparar un proyectil,la direccion de desplazamiento,su salto y deteccion de piso.

Variables de instancia:

```java
    //posicion de la princesa en pantalla
    private double x;
    private double y;
    //dimensiones constantes de la princesa
    private int ANCHO;
    private int ALTO;
    //posicion(del piso inferior) inicial donde nace la princesa
    private double piso = 750;
    /*Se le asigna una imagen cuando se desplaza hacia la izquierda
    y otra cuando lo hace hacia la derecha */
    private Image imagenDer;
    private Image imagenIzq;
    private double angulo;
    private double escala;
    //booleano para diferenciar los lados de disparo de la princesa
    private boolean disparoLadoPrincesa;
    //----------Variables de salto----------//
    // Variable para verificar si el personaje está en el suelo
    private boolean enElSuelo = true;
    // Velocidad vertical del personaje,se inicializa en 0 hasta que detecte la tecla "X"
    private double velocidadY = 0;
    //Representa la velocidad de caida cuando no detecte un bloque
    private double gravedad = 0.1;
```

Metodos:

Constructor de la princesa

Se le asigna la posicion en “X” e “Y”, sus dimensiones, inclinacion, y su correspondiente imagen dependiendo si se desplaza a la derecha o izquierda

```java
//----------constructor----------//
public Princesa(int x, int y, int ancho, int alto) {
    this.x = x;
    this.y = y;
    this.ANCHO = ancho;
    this.ALTO = alto;
    this.imagenDer = Herramientas.cargarImagen("princesaDer.png");
    this.imagenIzq = Herramientas.cargarImagen("princesaIzq.png");
    this.angulo = 0;
    this.escala = 0.03;
}
```

Getters y setters

```java
//----------Getters y Setters----------//
public double getVelocidadY() {
    return velocidadY;
}

public void setVelocidadY(double velocidadY) {
    this.velocidadY = velocidadY;
}

public double getX() {
    return x;
}

public void setX(double x) {
    this.x = x;
}

public double getY() {
    return y;
}

public void setY(double y) {
    this.y = y;
}

public int getAncho() {
    return ANCHO;
}

public void setAncho(int ancho) {
    this.ANCHO = ancho;
}

public int getAlto() {
    return ALTO;
}

public void setAlto(int alto) {
    this.ALTO = alto;
}

public double getAngulo() {
    return angulo;
}

public void setAngulo(double angulo) {
    this.angulo = angulo;
}
```

```java
public double getEscala() {
    return escala;
}

public void setEscala(double escala) {
    this.escala = escala;
}

public double getGravedad() {
    return gravedad;
}

public void setGravedad(double gravedad) {
    this.gravedad = gravedad;
}

public double getPiso() {
    return piso;
}

public void setPiso(double piso) {
    this.piso = piso;
}

public boolean getEnElSuelo() {
    return enElSuelo;
}

public void setEnElSuelo(boolean enElSuelo) {
    this.enElSuelo = enElSuelo;
}

public boolean getDisparoLadoPrincesa() {
    return disparoLadoPrincesa;
}

public void setDisparoLadoPrincesa(boolean disparoLadoPrincesa) {
    this.disparoLadoPrincesa = disparoLadoPrincesa;
}
```

Metodos:

DibujarDer: utliza la variable “private Image imagenDer” si se desplaza hacia la derecha (se la llama en la clase juego)

DibujarIzq: utliza la variable “private Image imagenIzq” si se desplaza hacia la izquierda (se la llama en la clase juego)

MovIzq , movDer:Representa la velocidad en X de desplazamiento laterales

Saltar: habilidad que permite esquivar o subir al piso consecutivo, se puede utilizar cuando la princesa colisiona su extremo inferior con el extremo superior de algun bloque

CaidaSalto: cuando la princesa esta en el aire producto del salto, a una determinada altura o si colisiona con un bloque la gravedad la direcciona nuevamente al piso

```
public void dibujarDer(Entorno entorno) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ancho, this.alto, 0, Color.blue);
    entorno.dibujarImagen(imagenDer, x, y, angulo, escala);
}

public void dibujarIzq(Entorno entorno) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ancho, this.alto, 0, Color.blue);
    entorno.dibujarImagen(imagenIzq, x, y, angulo, escala);
}

public void movIzq() {
    this.x -= 3;
}

public void movDer() {
    this.x += 3;
}

public void saltar() {
    // Detectar si se presionó la tecla de salto
    if(this.enElSuelo){
        velocidadY = -7; //fuerza del salto
        this.enElSuelo = false;
    }
}

public void caidaSalto() {
    // Aplicar gravedad en cada frame
    if (!enElSuelo) {
        velocidadY += gravedad;
        this.y += velocidadY;
    }
    if (this.y >= this.piso) {
        this.setY(this.piso);
        this.velocidadY = 0;
        this.enElSuelo = true;
    }
}
```

Clase Proyectil:

En esta clase se diseñan los proyectiles que utiliza la princesa y la direccion de desplazamiento(izquierda o derecha) del mismo.

Variables de instancia:

```
    //posición del proyectil
    private double x;
    private double y;
    //dimensiones constantes del proyectil
    private int ANCHO = 10;
    private int ALTO = 10;
    //Booleano Indica (inicialmente en false) si el disparo del proyectil esta recorriendo    el
    entorno
    private boolean disparoActivo = false;
    //imagen del proyectil
    private Image imagen;
```

Constructor: Se crea la posicion , el alto y ancho y la ruta de la imagen del proyectil en formato png

```java
//---------------Constructores----------------//
public Proyectil(double x, double y, int ANCHO, int ALTO) {
    this.x = x;
    this.y = y;
    this.ANCHO = ANCHO;
    this.ALTO = ALTO;
    imagen = Herramientas.cargarImagen("bolaDeFuego.png");
}
```

Getters y setters:

```java
public double getX() {
    return x;
}

public void setX(double d) {
    this.x = d;
}

public double getY() {
    return y;
}

public void setY(double d) {
    this.y = d;
}

public int getANCHO() {
    return ANCHO;
}

public void setANCHO(int ANCHO) {
    this.ANCHO = ANCHO;
}

public int getALTO() {
    return ALTO;
}

public void setALTO(int ALTO) {
    this.ALTO = ALTO;
}

public boolean getDisparoActivo() {
    return disparoActivo;
}

public void setDisparoActivo(boolean disparoActivo) {
    this.disparoActivo = disparoActivo;
}
```

DisparoDer y DisparoIzq: se determina la velocidad del proyectil y se dibuja en el entorno

```
//----------------Metodos----------------//
public void disparoDer(Entorno entorno ,int velocidadDisparo) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ANCHO, this.ALTO, 0, Color.blue);
    double bala =  this.x += velocidadDisparo;
    this.x = bala;
    entorno.dibujarImagen(imagen, this.x, this.y, 0, 0.2);
}

public void disparoIzq(Entorno entorno , int velocidadDisparo) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ANCHO, this.ALTO, 0, Color.red);
    double bala =  this.x -= velocidadDisparo;
    this.x = bala;
    entorno.dibujarImagen(imagen, this.x, this.y, 0, 0.2);
}
```

Clase Tiranosaurio:

En esta clase se diseñan los monstruos mutantes que recorren todos los pisos custidiando a la mascota que tienen prisionera

Variables de instancia:

```
    //posición del tiranosaurio en pantalla
    private int x;
    private int y;
    //dimensiones del tiranosaurio
    private int ANCHO;
    private int ALTO;
    //booleanos indican si el Tiranosaurio toca algun borde lateral del entorno
    private boolean tocoPantallaIzq = false;
    private boolean tocoPantallaDer = false;
    //booleado indica el lado de disparo del tiranosaurio
    private boolean disparoLadoTiranosaurios;
    //piso del tiranosaurio
    private int piso;
    //Representan la imagen en formato gif de los Tiranosaurios
    private Image imagen = Herramientas.cargarImagen("dino.gif");
    private double angulo;
    private double escala;
    //----------Variables de salto----------//
    /*Estas variables las utilizamos para que el Tiranosaurio
    reconozca el piso y en caso contrario caiga a un piso inferior */
    private boolean enElSuelo = true;
    private double velocidadY = 0;
    private double gravedad = 0.3;
```

Constructor: Los tiranosaurios cuentan con una posicion en “X” e “Y”, sus dimensiones y ademas contiene una variable que detecta en que piso se encuentra el tiranosaurio

```java
//----------------Constructores----------------//
public Tiranosaurio() {}
public Tiranosaurio(int x, int y, int ANCHO, int ALTO) {
    this.x = x;
    this.y = y;
    this.ANCHO = ANCHO;
    this.ALTO = ALTO;
    this.imagen = Herramientas.cargarImagen("dino.gif");
    this.angulo = 0;
    this.escala= 1;
    this.piso = this.y + this.ALTO/2;
}
```

Getters y setters

```java
//----------------Getters y Setters----------------//
public int getX() {
    return x;
}

public void setX(int x) {
    this.x = x;
}

public int getY() {
    return y;
}

public void setY(int y) {
    this.y = y;
}

public double getANCHO() {
    return ANCHO;
}

public void setANCHO(int ANCHO) {
    this.ANCHO = ANCHO;
}

public double getALTO() {
    return ALTO;
}

public void setALTO(int ALTO) {
    this.ALTO = ALTO;
}

public double getAngulo() {
    return angulo;
}

public void setAngulo(double angulo) {
    this.angulo = angulo;
}

public double getEscala() {
    return escala;
}

public void setEscala(double escala) {
    this.escala = escala;
}
```

```java
public void setEscala(double escala) {
    this.escala = escala;
}

public boolean getDisparoLadoTiranosaurios() {
    return disparoLadoTiranosaurios;
}

public void setDisparoLadoTiranosaurios(boolean disparoLadoTiranosaurios) {
    this.disparoLadoTiranosaurios = disparoLadoTiranosaurios;
}

public int getPiso() {
    return piso;
}
public void setPiso(int piso) {
    this.piso = piso;
}
public boolean getEnElSuelo() {
    return enElSuelo;
}
public void setEnElSuelo(boolean enElSuelo) {
    this.enElSuelo = enElSuelo;
}
```

movIzq: inicializa el desplazamiento del Tiranosaurio hacia la Izquierda hasta que toque el borde de la pantalla izquierda cambiando su direccion de desplazamiento (realiza esta secuencia en forma infinita o hasta que sea destruido)

MovDer: idem que el metodo movIzq pero hacia la derecha

TocoPantallaDinoComienzaSentidoDer: este metodo trabaja en conjunto con y “movDer”, cambiando el valor booleando de la variable tocoPantallaDer, para que de esta manera nos indique si el objeto llego al borde de la pantalla o esta recorriendo algun piso

TocoPantallaDinoComienzaSentidoIzq:Idem que el metodo anterior pero hacia la izquierda

CrearTiranosaurio: Recorre el array de tiranosaurios y los agrega en la lista de tiranosaurios

dibujar: Recorre cada tiranosaurio dentro de la lista y se le asigna sus atributos a la imagen, para que se dibuje en ciertos puntos de la pantalla en un determinado angulo y escala.

caida: cuando el tiranosaurio no detecta el suelo(es decir algun bloque) la gravedad lo hace caer hasta encontrar un nuevo bloque y lo setea la variable enElSuelo para que lo tome como nuevo piso

```java
public void movIzq() {
    if(tocoPantallaIzq == false) {
        this.x -= 1;
        disparoLadoTiranosaurios = true;
    }
    else if(tocoPantallaIzq == true){
        this.x = this.x+=1;
        disparoLadoTiranosaurios = false;
    }
}

public boolean tocoPantallaDinoComienzaSentidoIzq() {
    if(this.x < 31) {
        tocoPantallaIzq = true;
    }
    if(this.x > 975) {
        tocoPantallaIzq = false;
    }
    return tocoPantallaIzq;
}

public void movDer() {
    if(tocoPantallaDer == false) {
        this.x += 1;
        disparoLadoTiranosaurios = false;
    }
    else if(tocoPantallaDer == true){
        this.x = this.x-=1;
        disparoLadoTiranosaurios = true;
    }
}

public boolean tocoPantallaDinoComienzaSentidoDer() {
    if(this.x < 31) {
        tocoPantallaDer = false;
    }
    if(this.x > 975) {
        tocoPantallaDer = true;
    }
    return tocoPantallaDer;
}

public void crearTiranosaurio(ArrayList<Tiranosaurio> tiranosaurios, Tiranosaurio [] tiranosaur ) {
    for (int i = 0; i < tiranosaur.length; i++) {
        tiranosaurios.add(tiranosaur[i]);
    }
}
```

```java
  public void crearTiranosaurio(ArrayList<Tiranosaurio> tiranosaurios, Tiranosaurio [] tiranosaur ) {
      for (int i = 0; i < tiranosaur.length; i++) {
          tiranosaurios.add(tiranosaur[i]);
      }
  }

  public void dibujar(Entorno entorno , ArrayList<Tiranosaurio> tiranosaurios) {
      for (int i = 0; i < tiranosaurios.size(); i++) {
          Tiranosaurio tira = tiranosaurios.get(i);
              entorno.dibujarImagen(imagen, tira.getX(), tira.getY()-15, tira.getAngulo(), tira.getEscala());
      }
  }

  public void Caida() {
      // Aplicar gravedad en cada frame
      if(!enElSuelo) {
          velocidadY += gravedad;
          this.y += velocidadY;
      }
      if(this.y >= this.piso) {
          this.setY(this.piso);
          this.velocidadY = 0;
          this.enElSuelo = true;
      }
  }
```

Clase Juego:

En esta clase incluimos y trabajamos con todos los objetos creados

Variables de Instancia:

```java
    private Entorno entorno;
```

```
    private Bloques bloque;
    private Bloques [] piso1;
    private Bloques [] piso2;
    private Bloques [] piso3;
    private Bloques [] piso4;
    private ArrayList <Bloques> todosLosPisos;
    private ArrayList <Tiranosaurio> listaTiranosaurio;
    private ArrayList <Bomba> bombaDino;
    private ArrayList <Integer> pisos;
    private Princesa princesa;
    private Tiranosaurio [] tiranosaurios;
    private Tiranosaurio tiranosaurio;
    private Proyectil proyectil;
    private Random numeroAleatorio = new Random();
    private Image inicio;
    private Image fondo;
    private Image gameOver;
    private Image win;
    private Image gatito;
    private Clip sonidoInicio;
    private Clip sonidoJuego;
    private int enemigosEliminados;
    private int puntaje;
    private int vidas;
    private boolean pantallaDeInicio = true;
    private boolean end = false;
    private boolean winner = false;
```

Constructor juego:

En principio inicializamos el puntaje , las vidas y la cantidad de enemigos eliminados en 0, luego inicializamos el juego (entorno) con las medidas de alto y ancho provistas por defecto, cargamos el sonido de inicio y del juego , junto con las imagenes de inicio, de fondo y una imagen extra cuando perdemos y otra cuando ganamos. Por otra parte inicializamos los objetos Bloque,princesa, proyectil y tiranosaurio con los atributos provistos por parametros.

Seguidamente reservamos 20 espacios de bloques en el array de cada uno de los pisos, creamos un ArrayList que contenga todos los pisos y creamos los 4 niveles de pisos individualmente tomando como altura (del eje y del entorno) 800,600,400 y 200 y bloques de 50x50 que seran almacenados desde la función crearPiso en el ArrayList creado con anterioridad.

Luego reservamos 8 espacios para tiranosaurios , inicializamos el ArrayList que va a almacenar el array de tiranosaurios y los creamos mediante el metodo crearTiranosaurio que recibe como parametro el array de tiranosaurios y la lista.

A continuación inicializamos una lista de bomba, y recorremos cada cada tiranosaurio del ArrayList listaTiranosaurio y le asignamos una bomba a cada uno de ellos.

Por ultimo inicializamos el arrayList pisos que va a recibir los valores de los 4 niveles de pisos.

```
    //Inicialización de puntajes
    enemigosEliminados = 0;
    puntaje = 0;
```

```
        vidas = 10;
        //Inicialización del objeto entorno
        entorno = new Entorno(this, " Super Elizabeth Sis, Volcano Edition - Grupo ... - v1", 1000, 850);
        //Inicialización del sonido de inicio
        sonidoInicio = Herramientas.cargarSonido("inicio.wav");
        //Inicialización del sonido del juego
        sonidoJuego = Herramientas.cargarSonido("stratosphere.wav");
        //Inicializar lo que haga falta para el juego
        inicio = Herramientas.cargarImagen("inicio.jpg");
        //carga de imagen de fondo
        fondo = Herramientas.cargarImagen("interiorVolcan.jpeg");
        //Carga de imagen de Game Over
        gameOver = Herramientas.cargarImagen("gameOver.jpg");
        //Carga de fondo negro para Winner
        win = Herramientas.cargarImagen("winner.jpg");
        //Carha de imagen de gatito
        gatito = Herramientas.cargarImagen("gatito.gif");
        bloque = new Bloques();
        princesa = new Princesa(entorno.ancho()/2 , 750 , 30, 50);
        proyectil = new Proyectil(princesa.getX(), princesa.getY() , 20, 10);
        tiranosaurio = new Tiranosaurio();

        //Reserva de 20 espacios para bloques
        piso1 = new Bloques[20];
        piso2 = new Bloques[20];
        piso3 = new Bloques[20];
        piso4 = new Bloques[20];
        //Reserva de 8 espacios para tiranosaurios
        tiranosaurios = new Tiranosaurio[8];
        //Inicialización de la lista de Bloques y creación de pisos
        todosLosPisos = new ArrayList<Bloques>();
        bloque.crearPiso(todosLosPisos,piso1 , 800,50,50);
        bloque.crearPiso(todosLosPisos,piso2, 600,50,50);
        bloque.crearPiso(todosLosPisos,piso3 , 400,50,50);
        bloque.crearPiso(todosLosPisos,piso4 , 200,50,50);
        pisos = new ArrayList <Integer>();
        //Inicialización de la lista de Tiranosaurio y creación de Tiranosaurios
        listaTiranosaurio = new ArrayList<Tiranosaurio>();
        bombaDino = new ArrayList <Bomba>();
        tiranosaurios[0] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 761 , 10 , 30);
        tiranosaurios[1] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 761 , 10 , 30);
        tiranosaurios[2] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 561 , 10 , 30);
        tiranosaurios[3] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 561 , 10 , 30);
        tiranosaurios[4] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 361 , 10 , 30);
        tiranosaurios[5] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 361 , 10 , 30);
        tiranosaurios[6] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 161 , 10 , 30);
        tiranosaurios[7] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 161 , 10 , 30);
        tiranosaurio.crearTiranosaurio(listaTiranosaurio, tiranosaurios);
        //Proyectil tiranosaurio
        for (int i = 0; i < listaTiranosaurio.size(); i++) {
                bombaDino.add(new Bomba(listaTiranosaurio.get(i).getX(),listaTiranosaurio.get(i).getY(),20,20));
        }
        pisos.add(161);
        pisos.add(361);
        pisos.add(561);
        pisos.add(761);

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

Tick:

La variable end se inicializa con el valor true, utilizamos esta variable como condición para el comienzo del juego, si la variable end es distinta de true, el juego comienza mostrandonos una pantalla de inicio hasta que se presione la tecla ENTER. Una vez presionada la tecla Enter el booleano de pantalla de inicio cambia su estado a false y nos introduce directamente en el juego , entrando en el metodo dibujar pantalla() que crea todo el entorno y luego entrando en actualizarJuego() que va a recorrer una y otra vez diferentes metodos de verificación por cada vuelta del tick. Estos dos metodos van a ser recorridos y actualizados una y otra vez por el tick hasta que la variable end o winner cambien de estado, este cambio se dara cuando el jugador gané o pierda una partida.

```
public void tick(){
    if(end != true) {
        if(pantallaDeInicio) {
            mostrarPantallaInicio();
            if(this.entorno.sePresiono((char) KeyEvent.VK_ENTER)) {
                pantallaDeInicio = false;
            }
        }
        else {
            dibujarPantalla();
            actualizarJuego();
        }
    }
    if(end == true){
        gameOverPerdedor();
    }
    if(winner == true) {
        gameOverGanador();
    }
}
```

Metodos de verificacion:

verificacionMovimientoIzquierdaPrincesa();verifica si se esta presionando la flecha izquierda llama al metodo moverIzquierdaPrincesa

verificacionMovimientoDerechoPrincesa(); idem que el metodo anterior pero hacia la derecha.

verificacionProyectilBomba(); llama al metodo colisionProyectilBomba, recorre todas las bombas y verifica si alguna colisiona con el proyectil de la princesa, en caso afirmativo se crea una nueva bomba y proyectil en sus respectivos lugares de origen.

verificacionTiranosauriosBloques();llama al metodo ColisionPiesTiranosauriosBloque y colisionTiranosaurioBloque y verifica constantemente que el Tiranosaurio este tocando un bloque, de esta manera nos aseguramos que si colisiona con null caiga a un piso inferior

verificacionProyectilTiranosaurios(); llama al metodo ColisionTiranosaurioProyectil y verifica si los proyectiles de la princesa colisionan con el Tiranosaurio, en caso afirmativo se remueve al Tiranosaurio junto con su bomba, incrementa el puntaje y el contador de enemigosEliminados

verificacionDeColisionesPrincesaBloque();llama a los metodos ColisionPiesBloque, ColisionCabezaBloque y recorre todos los bloques y llama a la funcion colisionCabezaBloque, si detecta un bloque se setea la velocidad en Y en +10, y utilizamos otro condicional para que setee automaticamente el piso cuando entre en contacto los pies de la princesa con algun bloque

verificacionDeColisionesPrincesaTiranosaurio(): llama al metodo ColisionPrincesaTiranosaurio y verifica si la princesa toca a un dinosaurio(en caso afirmativo la princesa pierde una vida)

verificacionDeColisionesPrincesaBomba(); llama al metodo ColisionPrincesaBomba y verifica si la princesa colisiona con una bomba de algun tiranosaurio(en caso afirmativo pierde una vida)

verificacionDeDisparosPrincesa(); Verifica si el usuario esta presionando la tecla “C” y hacia donde esta la direccion de la princesa para ejecutar un proyectil dirigido hacia la direccion actual de la princesa

verificaciónVidas(); En caso de que vidas llegué a 0, el estado del booleano end cambia a true.

verificaciónWinner(); verifica si la princesa entra en contacto con el extremo superior derecho de la pantalla, en caso afirmativo termina el juego y muestra una imagen de fondo.

ColisionPrincesaBloque: Este metodo es la conjuncion entre ColisionPiesBloque y ColisionCabezaBloque, diseñamos este metodo general de colisiones para que la princesa no traspase los bloques cuando salte

```
//-----------------------------------------------METODOS DE VERIFICACIÓN-------------------------------------//
//-----------------------------------------------------------------------------------------------------------//
//Verificación de movimiento izquierdo de la princesa
private void verificacionMovimientoIzquierdaPrincesa() {
    if(this.entorno.estaPresionada(this.entorno.TECLA_IZQUIERDA) && princesa.getX() > 31) {
        moverIzquierdaPrincesa();
    }
}
//Verificación de movimiento Derecho de la princesa
private void verificacionMovimientoDerechoPrincesa() {
    if(this.entorno.estaPresionada(this.entorno.TECLA_DERECHA) && princesa.getX() < 970) {
        moverDerechaPrincesa();
    }
}
//Verificación constante entre bombas y proyectiles
private void verificacionProyectilBomba() {
    for (int i = 0; i < bombaDino.size(); i++) {
        if(colisionProyectilBomba(proyectil , bombaDino.get(i))) {
            proyectil = new Proyectil(princesa.getX() , princesa.getY() , 20, 20);
            Bomba nuevoProyectil = new Bomba(listaTiranosaurio.get(i).getX() ,listaTiranosaurio.get(i).getY(),20,20);
            bombaDino.set(i, nuevoProyectil);
        }
    }
}
//VERIFICACIÓN DE COLISION CONSTANTE ENTRE LOS TIRANOSAURIOS Y LOS BLOQUES
//se busca captar el momento en el que los tiranosaurio se encuentran con un bloque null
private void verificacionTiranosauriosBloques() {
    for (int i = 0; i < listaTiranosaurio.size(); i++) {
        listaTiranosaurio.get(i).Caida();
        boolean colisionDetectada = false;
        for (int j = 0; j < todosLosPisos.size(); j++) {
            if (colisionPiesTiranosauriosBloque(todosLosPisos.get(j), listaTiranosaurio.get(i))) {
                colisionDetectada = true;
                break; //Salimos del bucle interno ya que detectamos una colisión
            }
        }
        if(colisionDetectada) {
            //si hay colision no hacemos nada
        } else {
            listaTiranosaurio.get(i).setEnElSuelo(false);
            listaTiranosaurio.get(i).setPiso(762);
              for (int j = 0; j < todosLosPisos.size(); j++) {
                if(colisionTiranosaurioBloque(todosLosPisos.get(j), listaTiranosaurio.get(i))){
                    listaTiranosaurio.get(i).setPiso(listaTiranosaurio.get(i).getY()-50);
                }
             }
        }
    }
```

```java
//Verificación Proyectil-Tiranosaurio y sumatoria de puntos
private void verificacionProyectilTiranosaurios() {
    for (int i = 0; i < listaTiranosaurio.size(); i++) {
        if(colisionTiranosaurioProyectil(proyectil,listaTiranosaurio.get(i))) {
            if(listaTiranosaurio.size() > 2) {
                listaTiranosaurio.remove(i);
                bombaDino.remove(i);
                enemigosEliminados ++;
                puntaje += 2;
            }
            else {
                int contador= numeroAleatorio.nextInt(0,3);
                Tiranosaurio tiranosaurio = new Tiranosaurio(numeroAleatorio.nextInt(50, 900), pisos.get(contador) , 10 , 30);
                listaTiranosaurio.set(i, tiranosaurio);
            }
        }
    }
}
/*Recorremos cada bloque de la lista, y verificamos si hay colisión entre la cabeza y el bloque,
en caso de dar true la función de colisión cabeza-bloque, pasamos a setear la velocidad de salto
para que frene en la parte inferior del bloque*/
private void verificacionDeColisionesPrincesaBloque() {
    for(Bloques bloque:todosLosPisos) {
        if(colisionCabezaBloque(bloque)) {
            princesa.setVelocidadY(+10);
        }
        if(colisionPiesBloque(bloque)){
            princesa.setPiso(princesa.getY());
        }
    }
}
//Verificación de colision entre princesa y tiranosaurio
private void verificacionDeColisionesPrincesaTiranosaurio() {
    for (int i = 0; i < listaTiranosaurio.size(); i++) {
        if(colisionPrincesaTiranosaurio(listaTiranosaurio.get(i))) {
            vidas--;
        }
    }
}
//Verificación de colision entre princesa y bomba de dinosaurio
private void verificacionDeColisionesPrincesaBomba() {
    for (int i = 0; i < bombaDino.size(); i++) {
        if(colisionPrincesaBomba(bombaDino.get(i))){
            vidas--;
        }
    }
}
```

```java
//Verificación de disparos de la princesa
private void verificacionDeDisparosPrincesa() {
    //Disparo Izquierda
    if(this.entorno.estaPresionada('c') && proyectil != null && princesa.getDisparoLadoPrincesa() == true && proyectil.getX() >0 ) {
        proyectil.setDisparoActivo(true);
        disparoPrincesa(proyectil , princesa.getDisparoLadoPrincesa());
    }
    //Disparo Derecha
    if(this.entorno.estaPresionada('c') && proyectil != null && princesa.getDisparoLadoPrincesa() == false && proyectil.getX() < 1000 ) {
        proyectil.setDisparoActivo(true);
        disparoPrincesa(proyectil , princesa.getDisparoLadoPrincesa());
    }
}
//Verificación vidas
private void verificaciónVidas() {
    if(vidas == 0) {
        end = true;
    }
}
//Verificación Winner
private void verificaciónWinner() {
    if(princesa.getY() < 162 && princesa.getX() > 970) {
        winner = true;
    }
}
```

Metodos de la princesa:

MoverIzquierdaPrincesa: inicializa un proyectil en la posicion actual de la princesa,dibuja la imagen izquierda de la princesa y comienza a desplazarse a la izquierda (siempre y cuando la flecha izquierda este presionada)

MoverIzquierdaPrincesa: idem que el metodo anterior pero hacia la derecha

DisparoPrincesa: la princesa dispara un proyectil dependiendo para que lado este mirando la princesa, ademas cuando el proyectil colisione con algun borde de la pantalla se destruye automaticamente (null) y al mismo tiempo instancia nuevamente un proyectil

SaltoPrincesa: habilidad de la princesa para poder saltar(manteniendo apretado la tecla “X”, en caso de que colisione con algun bloque, caerá nuevamente hacia el piso donde se encontraba

```
//---------------------------------------Metodos Princesa-----------------------------------------------//
//------------------------------------------------------------------------------------------------------//
//Movimiento izquierda de la princesa
private void moverIzquierdaPrincesa() {
    this.proyectil = new Proyectil(princesa.getX(), princesa.getY()  , 20, 20);
    princesa.dibujarIzq(this.entorno);
    princesa.movIzq();
    princesa.setDisparoLadoPrincesa(true);
    for(int i = 0; i < todosLosPisos.size(); i++) {
        if(colisionPrincesaBloque(todosLosPisos.get(i))) {
        }
        else {
            princesa.setEnElSuelo(false);
            princesa.setPiso(750);
            if(colisionPiesBloque(todosLosPisos.get(i))){
                princesa.setPiso(princesa.getY());
            }
        }
    }
}
//Movimiento hacia la derecha de la princesa
private void moverDerechaPrincesa() {
    this.proyectil = new Proyectil(princesa.getX(), princesa.getY()  , 20, 20);
    princesa.movDer();
    princesa.setDisparoLadoPrincesa(false);
    for(int i = 0; i < todosLosPisos.size(); i++) {
        if(colisionPrincesaBloque(todosLosPisos.get(i))) {
        }
        else {
            princesa.setEnElSuelo(false);
            princesa.setPiso(750);
            if(colisionPiesBloque(todosLosPisos.get(i))){
                princesa.setPiso(princesa.getY());
            }
        }
    }
}
//Disparo de la princesa
private void disparoPrincesa(Proyectil proyectil , boolean disparoLado) {
    //Disparo Izquierda
    if(proyectil!= null && proyectil.getDisparoActivo() == true && disparoLado == true) {
        proyectil.disparoIzq(entorno,10);
    }
    //Disparo Derecha
    if(proyectil!= null && proyectil.getDisparoActivo() == true && disparoLado == false) {
        proyectil.disparoDer(entorno,10);
    }
```

```
//Salto de la princesa
private void saltoPrincesa() {
    if(this.entorno.estaPresionada('x')) {
        princesa.saltar();
        romperBloque();
        for(Bloques bloque:todosLosPisos) {
            if(colisionPiesBloque(bloque) && bloque == null) {
                princesa.setVelocidadY(+10);
            }
        }
    }
}
```

Metodos de los Tiranosaurios

moverTiranosaurios: Recorre el ArrayList de listaTiranosaurio y le asigna moviemiento izquierdo (a los dinosaurios que nacen en el lateral derecho) y viceversa para el movimiento derecho

disparoTiranosaurio: recorre la lista de las bombas y acciona un disparo a cada Tiranosaurio dependiendo la direccion de desplazamiento actual del mismo, si alguna bomba toca algun borde de la pantalla , se crea una nueva bomba y se le asigna la posicion del Tiranosaurio que ejecuto ese disparo

```
                        //----------------------------Metodos tiranosaurios-------------------------------------//
                        //--------------------------------------------------------------------------------------//
//Movimiento de los tiranosaurios y colisión con la princesa
private void moverTiranosaurios() {
    for (int i = 0; i < listaTiranosaurio.size(); i++) {
        if(i % 2 == 0) {
            listaTiranosaurio.get(i).movDer();
        }
        else {
            listaTiranosaurio.get(i).movIzq();
        }
        listaTiranosaurio.get(i).tocoPantallaDinoComienzaSentidoIzq();
        listaTiranosaurio.get(i).tocoPantallaDinoComienzaSentidoDer();
    }
}
//Disparo de bombas del tiranosaurio y colisión entre bombas y princesa
private void disparoTiranosaurio(ArrayList<Bomba> bomba) {
    //Control de dirección del disparo
    for (int i = 0; i < bomba.size(); i++) {
        if(bomba.get(i)!= null && listaTiranosaurio.get(i).getDisparoLadoTiranosaurios() == true) {
            bomba.get(i).disparoIzq(entorno, 3);
        }
        if(bomba.get(i)!= null && listaTiranosaurio.get(i).getDisparoLadoTiranosaurios() == false ) {
            bomba.get(i).disparoDer(entorno,3);
        }
        //Luego de disparar, ponemos el proyectil en null y volvemos a crearlo
        if(bomba != null && bomba.get(i).getX() >= 1000 ||bomba.get(i) != null && bomba.get(i).getX() <= 0) {
            Bomba nuevoProyectil = new Bomba(listaTiranosaurio.get(i).getX() ,listaTiranosaurio.get(i).getY(),20,20);
            bomba.set(i, nuevoProyectil);
        }
    }
}
```

Metodos del juego

dibujarBloques() : recibe un arrayList de bloques que son recorridos y dibujados en el juego.

romperBloque() : recorre el arrayList de todos los pisos y en caso de que alla una colisión entre la cabeza y el bloque la velocidad de la princesa disminuye y se setea el bloqué de colisión al valor null.

GameOverGanador: si la princesa logra alcanzar el extremo superior derecho del entorno sale del juego y se muestra una imagen de victoria con el puntaje y los enemigos eliminados.

GameOverPerdedor: si la princesa falla en el camino y pierde todas sus vidas , sale del juego y muestra una imagen.

```java
//Recibe arrays de pisos y los dibuja
private void dibujarBloques(ArrayList<Bloques> todosLosPisos) {
    for (int i = 0; i < todosLosPisos.size(); i++) {
        if (todosLosPisos.get(i) != null) {
            todosLosPisos.get(i).dibujar(entorno, todosLosPisos);
        }
    }
}
//Eliminación de bloques luego de la colision
private void romperBloque() {
    for (int i = 0; i < todosLosPisos.size(); i++) {
        Bloques bloque = todosLosPisos.get(i);
        if(bloque != null && bloque.GetseRompe()){
            if(colisionCabezaBloque(bloque)) {
                princesa.setVelocidadY(+10);
                todosLosPisos.set(i, null);
            }
        }
    }
}
//Game Over Ganador
private void gameOverGanador() {
    sonidoJuego.close();
    entorno.dibujarImagen(win,entorno.ancho()/2,entorno.alto()/2,0);
    entorno.cambiarFont("",60, Color.white);
    entorno.escribirTexto("WINNER!", 200,300);
    entorno.escribirTexto("Puntaje: "+ puntaje, 200,350);
    entorno.escribirTexto("Enemigos Eliminados: "+ enemigosEliminados, 200,400);
}
//Game over Perdedor
private void gameOverPerdedor() {
    sonidoJuego.close();
    entorno.dibujarImagen(gameOver,entorno.ancho()/2,entorno.alto()/2,0);
}
```

Problemas durante el desarrollo del juego :

1. Durante la creación del objeto bloque , tuvimos inconvenientes sobre como definir bloques duros o rompibles, y que condición poner para mezclar estos mismos en el entorno, en principio quisimos tomar los bloques de color rojo como duros y azules como blandos para que la princesa pueda reconocer y romper los blandos, no tuvimos exito y optamos por crear un booleano que le atribuya true a unos bloques y false a otros de forma aleatoria, con esto conseguimos que el juego no sea tan monotono y la distribucion de los bloques siempre sea al azar cuando inicia el juego.

2. Tuvimos inconvenientes cuando intentábamos que la princesa rompiera un bloque ya que no detectaba correctamente las colisiones, incluso traspasaba algunos bloques , probamos agrandando y reduciendo el ancho de la princesa pero no funciono, pudimos resolverlo cuando tomamos como ejemplo base el proyecto de la barra-pelotita vista en clase y fuimos haciendo leves modificaciones para que se adapte a nuestro proyecto.

3. Las primeras pruebas de colisiones y de caidas de la princesa a traves de los niveles lo hicimos experimentando con el piso superior a ella, sin embargo, cuando quisimos avanzar a los demas pisos, nos vimos obligados a crear una lista que contenga todos los niveles de pisos que estaban en arrays diferentes, esto nos ahorró la creación de varios for que verifiquen colisiones en cada nivel.

4. Durante la creación de las bombas de los tiranosaurios nos dimos cuenta que no podiamos implementar la misma logica que utilizamos con los proyectiles de la princesa, por lo que se nos ocurrió crear una lista de bombas que luego van a ser asignadas a cada uno de los dinosaurios dependiendo la ubicación en pantalla de cada uno de ellos.

Implementacion

Clase Bloques:

```java
package juego;

import java.awt.Color;

public class Bloques {
    //coordenadas de posicion de bloques
    private double x;
    private double y;
    //dimensión de bloques
    private int ancho;
    private int alto;
    //booleano para diferenciar a los bloques rompibles de los no rompibles
    private boolean seRompe;
    private double angulo;
    private double escala;
    //imagenes de los bloques
    private Image bloqueSeRompe = Herramientas.cargarImagen("bloqueSeRompe.png");
    private Image bloqueNoSeRompe = Herramientas.cargarImagen("bloqueNoSeRompe.png");
    //----------Constructores----------//
    public Bloques() {}
    public Bloques(int x, int y, int ancho, int alto , boolean seRompe ){
        this.x = x;
        this.y = y;
        this.ancho = ancho;
        this.alto = alto;
        this.angulo = 0;
        this.escala = 3;
        this.seRompe = seRompe;
    }

    //----------Getters y Setter----------//
    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public int getAncho() {
        return ancho;
    }

    public int getAlto() {
        return alto;
    }

    public boolean GetseRompe() {
        return seRompe;
```

```java
public int getAlto() {
    return alto;
}

public boolean GetseRompe() {
    return seRompe;
}
//----------Metodos----------//

//-----------Recibe un array de bloques y sus atributos y crea el piso---------------//
public void crearPiso(ArrayList<Bloques> todosLosPisos, Bloques [] conjuntoBloques , int y , int ancho, int alto) {
    int suma = 25;
    Random aleatorio = new Random();
    for (int i = 0; i < conjuntoBloques.length; i++) {
        if((aleatorio.nextInt(3)) % 6 == 0 || i == 3 ) {
            conjuntoBloques[i] = new Bloques(suma , y , ancho, alto , true);
            todosLosPisos.add(conjuntoBloques[i]);
            suma += 50;
        }
        else {
            conjuntoBloques[i] = new Bloques(suma , y , ancho, alto , false);
            todosLosPisos.add(conjuntoBloques[i]);
            suma += 50;
        }
    }
}
//-----------Recibe un array de bloques y los dibuja---------------//
public void dibujar(Entorno entorno , ArrayList<Bloques> todosLosPisos) {
    Color color;
    for (int i = 0; i < todosLosPisos.size(); i++) {
        Bloques bloque = todosLosPisos.get(i);
        if (bloque == null) {
            continue;
        }
        if(bloque.seRompe) {
            //color = Color.red;
            //entorno.dibujarRectangulo(bloque.x, bloque.y, bloque.ancho, bloque.alto, 0, color);
            entorno.dibujarImagen(bloqueSeRompe, bloque.x, bloque.y, this.angulo, this.escala);
        }
        else {
            //color = Color.BLUE;
            //entorno.dibujarRectangulo(bloque.x, bloque.y, bloque.ancho, bloque.alto, 0, color);
            entorno.dibujarImagen(bloqueNoSeRompe, bloque.x, bloque.y, this.angulo, this.escala);
        }
    }
}
```

Clase Bomba:

```java
package juego;

import java.awt.Color;

public class Bomba {
    //posicion de la bomba en pantalla
    private double x;
    private double y;
    //dimensiones constantes de la bomba
    private int ANCHO = 10;
    private int ALTO = 10;
    //Imagen de la bomba
    private Image imagen;
    //---------------Constructores----------------//
    public Bomba(int x, int y, int ANCHO, int ALTO) {
        this.x = x;
        this.y = y;
        this.ANCHO = ANCHO;
        this.ALTO = ALTO;
        this.imagen = Herramientas.cargarImagen("bombaTiranosaurio.png");
    }
    //---------------Getters y Setters----------------//
    public double getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public double getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public int getANCHO() {
        return ANCHO;
    }

    public void setANCHO(int ANCHO) {
        this.ANCHO = ANCHO;
    }

    public int getALTO() {
        return ALTO;
    }
```

```java
    public void setALTO(int ALTO) {
        this.ALTO = ALTO;
    }

    public void disparoDer(Entorno entorno ,int velocidadDisparo) {
        //entorno.dibujarRectangulo(this.x, this.y, this.ANCHO, this.ALTO, 0, Color.blue);
        double bala =  this.x += velocidadDisparo;
        this.x = bala;
        entorno.dibujarImagen(imagen, this.x, this.y, 0, 0.09);
    }

    public void disparoIzq(Entorno entorno , int velocidadDisparo) {
        //entorno.dibujarRectangulo(this.x, this.y, this.ANCHO, this.ALTO, 0, Color.red);
        double bala =  this.x -= velocidadDisparo;
        this.x = bala;
        entorno.dibujarImagen(imagen, this.x, this.y, 0, 0.09);
    }

}
```

Clase Juego:

```java
package juego;

import java.awt.Color;
import java.awt.Image;
import java.awt.event.KeyEvent;
import java.util.ArrayList;
import java.util.Random;
import javax.sound.sampled.Clip;
import entorno.Entorno;
import entorno.Herramientas;
import entorno.InterfaceJuego;

public class Juego extends InterfaceJuego {
    private Entorno entorno;
    private Bloques bloque;
    private Bloques [] piso1;
    private Bloques [] piso2;
    private Bloques [] piso3;
    private Bloques [] piso4;
    private ArrayList <Bloques> todosLosPisos;
    private ArrayList <Tiranosaurio> listaTiranosaurio;
    private ArrayList <Bomba> bombaDino;
    private ArrayList <Integer> pisos;
    private Princesa princesa;
    private Tiranosaurio [] tiranosaurios;
    private Tiranosaurio tiranosaurio;
    private Proyectil proyectil;
    private Random numeroAleatorio = new Random();
    private Image inicio;
    private Image fondo;
    private Image gameOver;
    private Image win;
    private Image gatito;
    private Clip sonidoInicio;
    private Clip sonidoJuego;
    private int enemigosEliminados;
    private int puntaje;
    private int vidas;
    private boolean pantallaDeInicio = true;
    private boolean end = false;
    private boolean winner = false;
```

```
Juego() {
    //Inicialización de puntajes
    enemigosEliminados = 0;
    puntaje = 0;
    vidas = 10;
    //Inicialización del objeto entorno
    entorno = new Entorno(this, " Super Elizabeth Sis, Volcano Edition - Grupo ... - v1", 1000, 850);
    //Inicialización  del sonido de inicio
    sonidoInicio = Herramientas.cargarSonido("inicio.wav");
    //Inicialización  del sonido del juego
    sonidoJuego =  Herramientas.cargarSonido("stratosphere.wav");
    //Inicializar lo que haga falta para el juego
    inicio = Herramientas.cargarImagen("inicio.jpg");
    //carga de imagen de fondo
    fondo = Herramientas.cargarImagen("interiorVolcan.jpeg");
    //Carga de imagen de Game Over
    gameOver = Herramientas.cargarImagen("gameOver.jpg");
    //Carga de fondo negro para Winner
    win = Herramientas.cargarImagen("winner.jpg");
    //Carha de imagen de gatito
    gatito = Herramientas.cargarImagen("gatito.gif");
    bloque = new Bloques();
    princesa = new Princesa(entorno.ancho()/2 , 750 , 30, 50);
    proyectil = new Proyectil(princesa.getX(), princesa.getY() , 20, 10);
    tiranosaurio = new Tiranosaurio();

    //Reserva de 20 espacios para bloques
    piso1 = new Bloques[20];
    piso2 = new Bloques[20];
    piso3 = new Bloques[20];
    piso4 = new Bloques[20];
    //Reserva de 8 espacios para tiranosaurios
    tiranosaurios = new Tiranosaurio[8];
    //Inicialización de la lista de Bloques y creación de pisos
    todosLosPisos = new ArrayList<Bloques>();
    bloque.crearPiso(todosLosPisos,piso1 , 800,50,50);
    bloque.crearPiso(todosLosPisos,piso2, 600,50,50);
    bloque.crearPiso(todosLosPisos,piso3 , 400,50,50);
    bloque.crearPiso(todosLosPisos,piso4 , 200,50,50);
    pisos = new ArrayList <Integer>();
    //Inicialización de la lista de Tiranosaurio y creación de Tiranosaurios
    listaTiranosaurio = new ArrayList<Tiranosaurio>();
    bombaDino = new ArrayList <Bomba>();
    tiranosaurios[0] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 761 , 10 , 30);
    tiranosaurios[1] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 761 , 10 , 30);
    tiranosaurios[2] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 561 , 10 , 30);
    tiranosaurios[3] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 561 , 10 , 30);
    tiranosaurios[4] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 361 , 10 , 30);
```

```
        //Inicialización de la lista de Tiranosaurio y creación de Tiranosaurios
        listaTiranosaurio = new ArrayList<Tiranosaurio>();
        bombaDino = new ArrayList <Bomba>();
        tiranosaurios[0] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 761 , 10 , 30);
        tiranosaurios[1] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 761 , 10 , 30);
        tiranosaurios[2] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 561 , 10 , 30);
        tiranosaurios[3] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 561 , 10 , 30);
        tiranosaurios[4] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 361 , 10 , 30);
        tiranosaurios[5] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 361 , 10 , 30);
        tiranosaurios[6] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 161 , 10 , 30);
        tiranosaurios[7] = new Tiranosaurio(numeroAleatorio.nextInt(50, 900) , 161 , 10 , 30);
        tiranosaurio.crearTiranosaurio(listaTiranosaurio, tiranosaurios);
        //Proyectil tiranosaurio
        for (int i = 0; i < listaTiranosaurio.size(); i++) {
            bombaDino.add(new Bomba(listaTiranosaurio.get(i).getX(),listaTiranosaurio.get(i).getY(),20,20));
        }
        pisos.add(161);
        pisos.add(361);
        pisos.add(561);
        pisos.add(761);

        // Inicia el juego!
        entorno.iniciar();
    }
    /**
     * Durante el juego, el método tick() será ejecutado en cada instante y por lo
     * tanto es el método más importante de esta clase. Aquí se debe actualizar el
     * estado interno del juego para simular el paso del tiempo (ver el enunciado
     * del TP para mayor detalle).
     */
    public void tick(){
        if(end != true) {
            if(pantallaDeInicio) {
                mostrarPantallaInicio();
                if(this.entorno.sePresiono((char) KeyEvent.VK_ENTER)) {
                    pantallaDeInicio = false;
                }
            }
            else {
                dibujarPantalla();
                actualizarJuego();
            }
        }
        if(end == true){
            gameOverPerdedor();
        }
        if(winner == true) {
            gameOverGanador();
```

```
}
public void mostrarPantallaInicio() {
    entorno.dibujarImagen(inicio, entorno.ancho()/2,entorno.alto()/2,0);
    sonidoInicio.loop(1);
}
public void dibujarPantalla () {
    //cierre de sonido de inicio
    sonidoInicio.close();
    //apertura de sonido del juego
    sonidoJuego.loop(3);
    //Dibuja el fondo del juego
    entorno.dibujarImagen(fondo, entorno.ancho()/2,entorno.alto()/2,0);
    entorno.dibujarImagen(gatito, entorno.ancho() - 40, 161, 0, 0.5);
    //Dibuja los bloques de cada piso
    dibujarBloques(todosLosPisos);
    //Cambio de fuente
    entorno.cambiarFont("",25, Color.white);
    //Dibuja el puntaje
    entorno.escribirTexto("Puntaje: "+ this.puntaje, 15,20);
    //Dibuja enemigos eliminados.
    entorno.escribirTexto("Enemigos Eliminados: "+ this.enemigosEliminados, 15,50);
    ////Dibuja cantidad de vidas.
    entorno.escribirTexto("vidas: " + this.vidas, 15, 80);
    //Dibuja a la princesa
    princesa.dibujarDer(this.entorno);
    //Dibuja tiranosaurio
    tiranosaurio.dibujar(entorno, listaTiranosaurio);
}
```

```
public void actualizarJuego() {
    //Movimiento izquierda y caida si bloque es null
    verificacionMovimientoIzquierdaPrincesa();
    //Movimiento derecha y caida si bloque es null
    verificacionMovimientoDerechoPrincesa();
    //Verificación de colisión constante entre proyectil-bomba y seteo de las mismas
    verificacionProyectilBomba();
    //verificación de colision constante entre los tiranosaurios y los bloques
    verificacionTiranosauriosBloques();
    //Verificación de colisión constante entre los proyectiles de la princesa y los tiranosaurios
    verificacionProyectilTiranosaurios();
    //Verificación de colisión entre princesa y bloques
    verificacionDeColisionesPrincesaBloque();
    //Verificación de colisiones entre princesa y tiranosaurio
    verificacionDeColisionesPrincesaTiranosaurio();
    //Verificación de colisiones entre princesa y bombas de tiranosaurios
    verificacionDeColisionesPrincesaBomba();
    //verificacion de disparos, cuando se presiona la tecla 'c' la princesa dispara proyectiles
    verificacionDeDisparosPrincesa();
    //Verificación de vidas
    verificaciónVidas();
    //Verificación de estado Winner
    verificaciónWinner();
    //Disparo princesa
    disparoPrincesa(this.proyectil, princesa.getDisparoLadoPrincesa());
    //Salto de la princesa
    saltoPrincesa();
    //Automatización de movimientos de los tiranosaurios
    moverTiranosaurios();
    //Disparo tiranosaurios
    disparoTiranosaurio(bombaDino);
    //Caida del salto de la princesa
    princesa.caidaSalto();
}
//-----------------------------------------------METODOS DE VERIFICACIÓN-----------------------------------------//
//---------------------------------------------------------------------------------------------------------------//
//Verificación de movimiento izquierdo de la princesa
private void verificacionMovimientoIzquierdaPrincesa() {
    if(this.entorno.estaPresionada(this.entorno.TECLA_IZQUIERDA) && princesa.getX() > 31) {
        moverIzquierdaPrincesa();
    }
}
//Verificación de movimiento Derecho de la princesa
private void verificacionMovimientoDerechoPrincesa() {
    if(this.entorno.estaPresionada(this.entorno.TECLA_DERECHA) && princesa.getX() < 970) {
        moverDerechaPrincesa();
    }
}
```

```java
//-----------------------------------------------METODOS DE VERIFICACIÓN-----------------------------------------//
//---------------------------------------------------------------------------------------------------------------//
//Verificación de movimiento izquierdo de la princesa
private void verificacionMovimientoIzquierdaPrincesa() {
    if(this.entorno.estaPresionada(this.entorno.TECLA_IZQUIERDA) && princesa.getX() > 31) {
        moverIzquierdaPrincesa();
    }
}
//Verificación de movimiento Derecho de la princesa
private void verificacionMovimientoDerechoPrincesa() {
    if(this.entorno.estaPresionada(this.entorno.TECLA_DERECHA) && princesa.getX() < 970) {
        moverDerechaPrincesa();
    }
}
//Verificación constante entre bombas y proyectiles
private void verificacionProyectilBomba() {
    for (int i = 0; i < bombaDino.size(); i++) {
        if(colisionProyectilBomba(proyectil , bombaDino.get(i))) {
            proyectil = new Proyectil(princesa.getX() , princesa.getY() , 20, 20);
            Bomba nuevoProyectil = new Bomba(listaTiranosaurio.get(i).getX() ,listaTiranosaurio.get(i).getY(),20,20);
            bombaDino.set(i, nuevoProyectil);
        }
    }
}
//VERIFICACIÓN DE COLISION CONSTANTE ENTRE LOS TIRANOSAURIOS Y LOS BLOQUES
//se busca captar el momento en el que los tiranosaurio se encuentran con un bloque null
private void verificacionTiranosauriosBloques() {
    for (int i = 0; i < listaTiranosaurio.size(); i++) {
        listaTiranosaurio.get(i).Caida();
        boolean colisionDetectada = false;
        for (int j = 0; j < todosLosPisos.size(); j++) {
            if (colisionPiesTiranosauriosBloque(todosLosPisos.get(j), listaTiranosaurio.get(i))) {
                colisionDetectada = true;
                break; //Salimos del bucle interno ya que detectamos una colisión
            }
        }
        if(colisionDetectada) {
            //si hay colision no hacemos nada
        } else {
            listaTiranosaurio.get(i).setEnElSuelo(false);
            listaTiranosaurio.get(i).setPiso(762);
              for (int j = 0; j < todosLosPisos.size(); j++) {
                if(colisionTiranosaurioBloque(todosLosPisos.get(j), listaTiranosaurio.get(i))){
                    listaTiranosaurio.get(i).setPiso(listaTiranosaurio.get(i).getY()-50);
                }
             }
        }
    }
}
```

```java
//Verificación Proyectil-Tiranosaurio y sumatoria de puntos
private void verificacionProyectilTiranosaurios() {
    for (int i = 0; i < listaTiranosaurio.size(); i++) {
        if(colisionTiranosaurioProyectil(proyectil,listaTiranosaurio.get(i))) {
            if(listaTiranosaurio.size() > 2) {
                listaTiranosaurio.remove(i);
                bombaDino.remove(i);
                enemigosEliminados ++;
                puntaje += 2;
            }
            else {
                int contador= numeroAleatorio.nextInt(0,3);
                Tiranosaurio tiranosaurio = new Tiranosaurio(numeroAleatorio.nextInt(50, 900), pisos.get(contador) , 10 , 30);
                listaTiranosaurio.set(i, tiranosaurio);
            }
        }
    }
}
/*Recorremos cada bloque de la lista, y verificamos si hay colisión entre la cabeza y el bloque,
en caso de dar true la función de colisión cabeza-bloque, pasamos a setear la velocidad de salto
para que frene en la parte inferior del bloque*/
private void verificacionDeColisionesPrincesaBloque() {
    for(Bloques bloque:todosLosPisos) {
        if(colisionCabezaBloque(bloque)) {
            princesa.setVelocidadY(+10);
        }
        if(colisionPiesBloque(bloque)){
            princesa.setPiso(princesa.getY());
        }
    }
}
//Verificación de colision entre princesa y tiranosaurio
private void verificacionDeColisionesPrincesaTiranosaurio() {
    for (int i = 0; i < listaTiranosaurio.size(); i++) {
        if(colisionPrincesaTiranosaurio(listaTiranosaurio.get(i))) {
            vidas--;
        }
    }
}
//Verificación de colision entre princesa y bomba de dinosaurio
private void verificacionDeColisionesPrincesaBomba() {
    for (int i = 0; i < bombaDino.size(); i++) {
        if(colisionPrincesaBomba(bombaDino.get(i))){
            vidas--;
        }
    }
}
```

```java
//Verificación de disparos de la princesa
private void verificacionDeDisparosPrincesa() {
    //Disparo Izquierda
    if(this.entorno.estaPresionada('c') && proyectil != null && princesa.getDisparoLadoPrincesa() == true && proyectil.getX() >0 ) {
        proyectil.setDisparoActivo(true);
        disparoPrincesa(proyectil , princesa.getDisparoLadoPrincesa());
    }
    //Disparo Derecha
    if(this.entorno.estaPresionada('c') && proyectil != null && princesa.getDisparoLadoPrincesa() == false && proyectil.getX() < 1000 ) {
        proyectil.setDisparoActivo(true);
        disparoPrincesa(proyectil , princesa.getDisparoLadoPrincesa());
    }
}
//Verificación vidas
private void verificaciónVidas() {
    if(vidas == 0) {
        end = true;
    }
}
//Verificación Winner
private void verificaciónWinner() {
    if(princesa.getY() < 162 && princesa.getX() > 970) {
        winner = true;
    }
}
//---------------------------------------Metodos Princesa-----------------------------------------//
//------------------------------------------------------------------------------------------------//
//Movimiento izquierda de la princesa
private void moverIzquierdaPrincesa() {
    this.proyectil = new Proyectil(princesa.getX(), princesa.getY()  , 20, 20);
    princesa.dibujarIzq(this.entorno);
    princesa.movIzq();
    princesa.setDisparoLadoPrincesa(true);
    for(int i = 0; i < todosLosPisos.size(); i++) {
        if(colisionPrincesaBloque(todosLosPisos.get(i))) {
        }
        else {
            princesa.setEnElSuelo(false);
            princesa.setPiso(750);
            if(colisionPiesBloque(todosLosPisos.get(i))){
                princesa.setPiso(princesa.getY());
            }
        }
    }
}
```

```java
//Movimiento hacia la derecha de la princesa
private void moverDerechaPrincesa() {
    this.proyectil = new Proyectil(princesa.getX(), princesa.getY()  , 20, 20);
    princesa.movDer();
    princesa.setDisparoLadoPrincesa(false);
    for(int i = 0; i < todosLosPisos.size(); i++) {
        if(colisionPrincesaBloque(todosLosPisos.get(i))) {
        }
        else {
            princesa.setEnElSuelo(false);
            princesa.setPiso(750);
            if(colisionPiesBloque(todosLosPisos.get(i))){
                princesa.setPiso(princesa.getY());
            }
        }
    }
}
//Disparo de la princesa
private void disparoPrincesa(Proyectil proyectil , boolean disparoLado) {
    //Disparo Izquierda
    if(proyectil!= null && proyectil.getDisparoActivo() == true && disparoLado == true) {
        proyectil.disparoIzq(entorno,10);
    }
    //Disparo Derecha
    if(proyectil!= null && proyectil.getDisparoActivo() == true && disparoLado == false) {
        proyectil.disparoDer(entorno,10);
    }
    //Luego de que el proyectil salga de la pantalla, ponemos al proyectil en null y volvemos a crearlo
    if(proyectil != null && this.proyectil.getX() >= 1000 ||proyectil != null && this.proyectil.getX() <= 0) {
        this.proyectil = null;
        this.proyectil = new Proyectil(princesa.getX(), princesa.getY()  , 20, 20);
    }
}
//Salto de la princesa
private void saltoPrincesa() {
    if(this.entorno.estaPresionada('x')) {
        princesa.saltar();
        romperBloque();
        for(Bloques bloque:todosLosPisos) {
            if(colisionPiesBloque(bloque) && bloque == null) {
                princesa.setVelocidadY(+10);
            }
        }
    }
}
```

```java
//Colisión entre el pie de la princesa y la parte superior del bloque
private boolean colisionPiesBloque(Bloques bloque) {
    boolean colisionY = colisionPrincesaBloque(bloque) &&
            princesa.getY() + princesa.getAlto()/2 > bloque.getY() - bloque.getAlto()/2 &&
            princesa.getY() + princesa.getAlto()/2 < (bloque.getY() - (bloque.getAlto()/2)+20);
    return colisionY;
}
//Colisión entre la cabeza de la princesa y la parte inferior del bloque
private boolean colisionCabezaBloque(Bloques bloque) {
    boolean colisionY = colisionPrincesaBloque(bloque) &&
            princesa.getY() - princesa.getAlto()/2 < bloque.getY() + bloque.getAlto()/2 &&
            princesa.getY() - princesa.getAlto()/2 > (bloque.getY() - bloque.getAlto()/2);
    return colisionY;
}
//Colisión general entre la princesa y el bloque
private boolean colisionPrincesaBloque(Bloques bloque) {
    if(bloque != null) {
        boolean colisionX = princesa.getX() - princesa.getAncho()/2 < bloque.getX() + bloque.getAncho()/2 &&
                            princesa.getX() - princesa.getAncho()/2 > bloque.getX() - bloque.getAncho()/2 ||
                            princesa.getX() + princesa.getAncho()/2 > bloque.getX() - bloque.getAncho()/2 &&
                            princesa.getX() + princesa.getAncho()/2 < bloque.getX() + bloque.getAncho()/2;
        boolean colisionY = princesa.getY() - princesa.getAlto()/2 > bloque.getY() - bloque.getAlto()/2 &&
                            princesa.getY() - princesa.getAlto()/2 < bloque.getY() + bloque.getAlto()/2 ||
                            princesa.getY() + princesa.getAlto()/2 > bloque.getY() - bloque.getAlto()/2 &&
                            princesa.getY() + princesa.getAlto()/2 < bloque.getY() + bloque.getAlto()/2;
        boolean colision = colisionX && colisionY;
        return colision;
    }
    else {
        return false;
    }
}
//Colisión Princesa Tiranosaurio
private boolean colisionPrincesaTiranosaurio(Tiranosaurio tiranosaurio) {
    if(tiranosaurio != null) {
        boolean colisionX = princesa.getX() - princesa.getAncho()/2 < tiranosaurio.getX() + tiranosaurio.getANCHO()/2 &&
                            princesa.getX() - princesa.getAncho()/2 > tiranosaurio.getX() - tiranosaurio.getANCHO()/2 ||
                            princesa.getX() + princesa.getAncho()/2 > tiranosaurio.getX() - tiranosaurio.getANCHO()/2 &&
                            princesa.getX() + princesa.getAncho()/2 < tiranosaurio.getX() + tiranosaurio.getANCHO()/2;
        boolean colisionY = tiranosaurio.getY() - tiranosaurio.getALTO()/2 > princesa.getY() - princesa.getAlto()/2 &&
                            tiranosaurio.getY() - tiranosaurio.getALTO()/2 < princesa.getY() + princesa.getAlto()/2 ||
                            tiranosaurio.getY() + tiranosaurio.getALTO()/2 > princesa.getY() - princesa.getAlto()/2 &&
                            tiranosaurio.getY() + tiranosaurio.getALTO()/2 <= princesa.getY() + princesa.getAlto()/2;
        return colisionX && colisionY;
    }
    else {
        return false;
    }
}
```

```java
private boolean colisionPrincesaBomba(Bomba bomba) {
    if(bomba != null) {
        boolean colisionX = princesa.getX() - princesa.getAncho()/2 < bomba.getX() + bomba.getANCHO()/2 &&
                            princesa.getX() - princesa.getAncho()/2 > bomba.getX() - bomba.getANCHO()/2 ||
                            princesa.getX() + princesa.getAncho()/2 > bomba.getX() - bomba.getANCHO()/2 &&
                            princesa.getX() + princesa.getAncho()/2 < bomba.getX() + bomba.getANCHO()/2;
        boolean colisionY = bomba.getY() - bomba.getALTO()/2 > princesa.getY() - princesa.getAlto()/2 &&
                            bomba.getY() - bomba.getALTO()/2 < princesa.getY() + princesa.getAlto()/2 ||
                            bomba.getY() + bomba.getALTO()/2 > princesa.getY() - princesa.getAlto()/2 &&
                            bomba.getY() + bomba.getALTO()/2 <= princesa.getY() + princesa.getAlto()/2;
        return colisionX && colisionY;
    }
    else {
        return false;
    }
}
                            //----------------------------Metodos tiranosaurios-------------------------------------//
                            //--------------------------------------------------------------------------------------//
//Movimiento de los tiranosaurios y colisión con la princesa
private void moverTiranosaurios() {
    for (int i = 0; i < listaTiranosaurio.size(); i++) {
        if(i % 2 == 0) {
            listaTiranosaurio.get(i).movDer();
        }
        else {
            listaTiranosaurio.get(i).movIzq();
        }
        listaTiranosaurio.get(i).tocoPantallaDinoComienzaSentidoIzq();
        listaTiranosaurio.get(i).tocoPantallaDinoComienzaSentidoDer();
    }
}
//Disparo de bombas del tiranosaurio y colisión entre bombas y princesa
private void disparoTiranosaurio(ArrayList<Bomba> bomba) {
    //Control de dirección del disparo
    for (int i = 0; i < bomba.size(); i++) {
        if(bomba.get(i)!= null && listaTiranosaurio.get(i).getDisparoLadoTiranosaurios() == true) {
            bomba.get(i).disparoIzq(entorno, 3);
        }
        if(bomba.get(i)!= null && listaTiranosaurio.get(i).getDisparoLadoTiranosaurios() == false ) {
            bomba.get(i).disparoDer(entorno,3);
        }
        //Luego de disparar, ponemos el proyectil en null y volvemos a crearlo
        if(bomba != null && bomba.get(i).getX() >= 1000 ||bomba.get(i) != null && bomba.get(i).getX() <= 0) {
            Bomba nuevoProyectil = new Bomba(listaTiranosaurio.get(i).getX() ,listaTiranosaurio.get(i).getY(),20,20);
            bomba.set(i, nuevoProyectil);
        }
    }
}
```

```java
//Colisión pies tiranosaurio-bloque
private boolean colisionPiesTiranosauriosBloque(Bloques bloque, Tiranosaurio tiranosaurio) {
    boolean colisionY = colisionTiranosaurioBloque(bloque, tiranosaurio) &&
                        tiranosaurio.getY() + ((tiranosaurio.getALTO() / 2) + 3) > bloque.getY() - bloque.getAlto() / 2 &&
                        tiranosaurio.getY() + ((tiranosaurio.getALTO() / 2) + 3) < bloque.getY() + (bloque.getAlto() / 2);
    return colisionY;
}
//Colisión tiranosaurio-proyectil
private boolean colisionTiranosaurioProyectil(Proyectil proyectil, Tiranosaurio tiranosaurio) {
    if(tiranosaurio != null && proyectil != null) {
        boolean colisionX = (tiranosaurio.getX() - tiranosaurio.getANCHO() / 2 < proyectil.getX() + proyectil.getANCHO() / 2 &&
                             tiranosaurio.getX() - tiranosaurio.getANCHO() / 2 > proyectil.getX() - proyectil.getANCHO() / 2) ||
                            (tiranosaurio.getX() + tiranosaurio.getANCHO() / 2 > proyectil.getX() - proyectil.getANCHO() / 2 &&
                             tiranosaurio.getX() + tiranosaurio.getANCHO() / 2 < proyectil.getX() + proyectil.getANCHO() / 2);
        boolean colisionY = (tiranosaurio.getY() - tiranosaurio.getALTO() / 2 > proyectil.getY() - proyectil.getANCHO() / 2 &&
                            tiranosaurio.getY() - tiranosaurio.getALTO() / 2 < proyectil.getY() + proyectil.getANCHO() / 2) ||
                            (tiranosaurio.getY() + tiranosaurio.getALTO() / 2 > proyectil.getY() - proyectil.getANCHO() / 2 &&
                            tiranosaurio.getY() + tiranosaurio.getALTO() / 2 <= proyectil.getY() + proyectil.getANCHO() / 2);
        return colisionX && colisionY;
    }
    else {
        return false;
    }
}
//Colisión tiranosaurio-Bloque
private boolean colisionTiranosaurioBloque(Bloques bloque, Tiranosaurio tiranosaurio) {
    if(tiranosaurio != null && bloque != null) {
        boolean colisionX = (tiranosaurio.getX() - tiranosaurio.getANCHO() / 2 < bloque.getX() + bloque.getAncho() / 2 &&
                             tiranosaurio.getX() - tiranosaurio.getANCHO() / 2 > bloque.getX() - bloque.getAncho() / 2) ||
                            (tiranosaurio.getX() + tiranosaurio.getANCHO() / 2 > bloque.getX() - bloque.getAncho() / 2 &&
                             tiranosaurio.getX() + tiranosaurio.getANCHO() / 2 < bloque.getX() + bloque.getAncho() / 2);
        boolean colisionY = (tiranosaurio.getY() - tiranosaurio.getALTO() / 2 > bloque.getY() - bloque.getAlto() / 2 &&
                            tiranosaurio.getY() - tiranosaurio.getALTO() / 2 < bloque.getY() + bloque.getAlto() / 2) ||
                            (tiranosaurio.getY() + tiranosaurio.getALTO() / 2 > bloque.getY() - bloque.getAlto() / 2 &&
                            tiranosaurio.getY() + tiranosaurio.getALTO() / 2 <= bloque.getY() + bloque.getAlto() / 2);
        return colisionX && colisionY;
    }
    else {
        return false;
    }
}
```

```
    //Colisión proyectil-bomba
    private boolean colisionProyectilBomba(Proyectil proyectil , Bomba bomba) {
        if(proyectil != null && bomba != null) {
            boolean colisionX = bomba.getX() - bomba.getANCHO()/2 < proyectil.getX() + proyectil.getANCHO()/2 &&
                                bomba.getX() - bomba.getANCHO()/2 > proyectil.getX() - proyectil.getANCHO()/2 ||
                                bomba.getX() + bomba.getANCHO()/2 > proyectil.getX() - proyectil.getANCHO()/2 &&
                                bomba.getX() + bomba.getANCHO()/2 < proyectil.getX() + proyectil.getANCHO()/2;
            boolean colisionY = bomba.getY() - bomba.getALTO()/2 > proyectil.getY() - proyectil.getALTO()/2 &&
                                bomba.getY() - bomba.getALTO()/2 < proyectil.getY() + proyectil.getALTO()/2 ||
                                bomba.getY() + bomba.getALTO()/2 > proyectil.getY() - proyectil.getALTO()/2 &&
                                bomba.getY() + bomba.getALTO()/2 <= proyectil.getY() + proyectil.getALTO()/2;
            return colisionX && colisionY;
        }
        else {
            return false;
        }
    }

    //Recibe arrays de pisos y los dibuja
    private void dibujarBloques(ArrayList<Bloques> todosLosPisos) {
        for (int i = 0; i < todosLosPisos.size(); i++) {
            if (todosLosPisos.get(i) != null) {
                todosLosPisos.get(i).dibujar(entorno, todosLosPisos);
            }
        }
    }
    //Eliminación de bloques luego de la colision
    private void romperBloque() {
        for (int i = 0; i < todosLosPisos.size(); i++) {
            Bloques bloque = todosLosPisos.get(i);
            if(bloque != null && bloque.GetseRompe()){
                if(colisionCabezaBloque(bloque)) {
                    princesa.setVelocidadY(+10);
                    todosLosPisos.set(i, null);
                }
            }
        }
    }
    //Game Over Ganador
    private void gameOverGanador() {
        sonidoJuego.close();
        entorno.dibujarImagen(win,entorno.ancho()/2,entorno.alto()/2,0);
        entorno.cambiarFont("",60, Color.white);
        entorno.escribirTexto("WINNER!", 200,300);
        entorno.escribirTexto("Puntaje: "+ puntaje, 200,350);
        entorno.escribirTexto("Enemigos Eliminados: "+ enemigosEliminados, 200,400);
    }
```

```
    //Game Over Ganador
    private void gameOverGanador() {
        sonidoJuego.close();
        entorno.dibujarImagen(win,entorno.ancho()/2,entorno.alto()/2,0);
        entorno.cambiarFont("",60, Color.white);
        entorno.escribirTexto("WINNER!", 200,300);
        entorno.escribirTexto("Puntaje: "+ puntaje, 200,350);
        entorno.escribirTexto("Enemigos Eliminados: "+ enemigosEliminados, 200,400);
    }
    //Game over Perdedor
    private void gameOverPerdedor() {
        sonidoJuego.close();
        entorno.dibujarImagen(gameOver,entorno.ancho()/2,entorno.alto()/2,0);
    }
    @SuppressWarnings("unused")
    public static void main(String[] args) {
        Juego juego = new Juego();
    }
```

Clase Princesa:

```java
package juego;

import java.awt.Color;

public class Princesa {
    //posicion de la princesa en pantalla
    private double x;
    private double y;
    //dimensiones constantes de la princesa
    private int ANCHO;
    private int ALTO;
    //posicion(del piso inferior) inicial donde nace la princesa
    private double piso = 750;
    /*Se le asigna una imagen cuando se desplaza hacia la izquierda
    y otra cuando lo hace hacia la derecha */
    private Image imagenDer;
    private Image imagenIzq;
    private double angulo;
    private double escala;
    //booleano para diferenciar los lados de disparo de la princesa
    private boolean disparoLadoPrincesa;
    //----------Variables de salto----------//
    // Variable para verificar si el personaje está en el suelo
    private boolean enElSuelo = true;
    // Velocidad vertical del personaje,se inicializa en 0 hasta que detecte la tecla "X"
    private double velocidadY = 0;
    //Representa la velocidad de caida cuando no detecte un bloque
    private double gravedad = 0.1;

    //----------constructor----------//
    public Princesa(int x, int y, int ancho, int alto) {
        this.x = x;
        this.y = y;
        this.ANCHO = ancho;
        this.ALTO = alto;
        this.imagenDer = Herramientas.cargarImagen("princesaDer.png");
        this.imagenIzq = Herramientas.cargarImagen("princesaIzq.png");
        this.angulo = 0;
        this.escala = 0.03;
    }

    //----------Getters y Setters----------//
    public double getVelocidadY() {
        return velocidadY;
    }
```

```java
public void setVelocidadY(double velocidadY) {
    this.velocidadY = velocidadY;
}

public double getX() {
    return x;
}

public void setX(double x) {
    this.x = x;
}

public double getY() {
    return y;
}

public void setY(double y) {
    this.y = y;
}

public int getAncho() {
    return ANCHO;
}

public void setAncho(int ancho) {
    this.ANCHO = ancho;
}

public int getAlto() {
    return ALTO;
}

public void setAlto(int alto) {
    this.ALTO = alto;
}

public double getAngulo() {
    return angulo;
}

public void setAngulo(double angulo) {
    this.angulo = angulo;
}

public double getEscala() {
    return escala;
}
```

```java
public void setEscala(double escala) {
    this.escala = escala;
}

public double getGravedad() {
    return gravedad;
}

public void setGravedad(double gravedad) {
    this.gravedad = gravedad;
}

public double getPiso() {
    return piso;
}

public void setPiso(double piso) {
    this.piso = piso;
}

public boolean getEnElSuelo() {
    return enElSuelo;
}

public void setEnElSuelo(boolean enElSuelo) {
    this.enElSuelo = enElSuelo;
}

public boolean getDisparoLadoPrincesa() {
    return disparoLadoPrincesa;
}

public void setDisparoLadoPrincesa(boolean disparoLadoPrincesa) {
    this.disparoLadoPrincesa = disparoLadoPrincesa;
}

//----------Getters y Setters----------//
public void dibujarDer(Entorno entorno) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ancho, this.alto, 0, Color.blue);
    entorno.dibujarImagen(imagenDer, x, y, angulo, escala);
}

public void dibujarIzq(Entorno entorno) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ancho, this.alto, 0, Color.blue);
    entorno.dibujarImagen(imagenIzq, x, y, angulo, escala);
}
```

```java
public void dibujarIzq(Entorno entorno) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ancho, this.alto, 0, Color.blue);
    entorno.dibujarImagen(imagenIzq, x, y, angulo, escala);
}

public void movIzq() {
    this.x -= 3;
}

public void movDer() {
    this.x += 3;
}

public void saltar() {
    // Detectar si se presionó la tecla de salto
    if(this.enElSuelo){
        velocidadY = -7; //fuerza del salto
        this.enElSuelo = false;
    }
}

public void caidaSalto() {
    // Aplicar gravedad en cada frame
    if (!enElSuelo) {
        velocidadY += gravedad;
        this.y += velocidadY;
    }
    if (this.y >= this.piso) {
        this.setY(this.piso);
        this.velocidadY = 0;
        this.enElSuelo = true;
    }
}
```

Clase Proyectil:

```java
package juego;

import java.awt.Color;

public class Proyectil {
    //posición del proyectil
    private double x;
    private double y;
    //dimensiones constantes del proyectil
    private int ANCHO = 10;
    private int ALTO = 10;
    //Booleano Indica (inicialmente en false) si el disparo del proyectil esta recorriendo  el entorno
    private boolean disparoActivo = false;
    //imagen del proyectil
    private Image imagen;
    //----------------Constructores-----------------//
    public Proyectil(double x, double y, int ANCHO, int ALTO) {
        this.x = x;
        this.y = y;
        this.ANCHO = ANCHO;
        this.ALTO = ALTO;
        imagen = Herramientas.cargarImagen("bolaDeFuego.png");
    }
    //----------------Getters y Setters-----------------//
    public double getX() {
        return x;
    }

    public void setX(double d) {
        this.x = d;
    }

    public double getY() {
        return y;
    }

    public void setY(double d) {
        this.y = d;
    }

    public int getANCHO() {
        return ANCHO;
    }

    public void setANCHO(int ANCHO) {
        this.ANCHO = ANCHO;
    }
```

```java
public int getALTO() {
    return ALTO;
}

public void setALTO(int ALTO) {
    this.ALTO = ALTO;
}

public boolean getDisparoActivo() {
    return disparoActivo;
}

public void setDisparoActivo(boolean disparoActivo) {
    this.disparoActivo = disparoActivo;
}
//----------------Metodos----------------//
public void disparoDer(Entorno entorno ,int velocidadDisparo) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ANCHO, this.ALTO, 0, Color.blue);
    double bala =  this.x += velocidadDisparo;
    this.x = bala;
    entorno.dibujarImagen(imagen, this.x, this.y, 0, 0.2);
}

public void disparoIzq(Entorno entorno , int velocidadDisparo) {
    //entorno.dibujarRectangulo(this.x, this.y, this.ANCHO, this.ALTO, 0, Color.red);
    double bala =  this.x -= velocidadDisparo;
    this.x = bala;
    entorno.dibujarImagen(imagen, this.x, this.y, 0, 0.2);
}
```

Clase tiranosaurio:

```java
package juego;

import java.awt.Color;

public class Tiranosaurio {
    //posición del tiranosaurio en pantalla
    private int x;
    private int y;
    //dimensiones del tiranosaurio
    private int ANCHO;
    private int ALTO;
    //booleanos indican si el Tiranosaurio toca algun borde lateral del entorno
    private boolean tocoPantallaIzq = false;
    private boolean tocoPantallaDer = false;
    //booleado indica el lado de disparo del tiranosaurio
    private boolean disparoLadoTiranosaurios;
    //piso del tiranosaurio
    private int piso;
    //Representan la imagen en formato gif de los Tiranosaurios
    private Image imagen = Herramientas.cargarImagen("dino.gif");
    private double angulo;
    private double escala;
    //----------Variables de salto----------//
    /*Estas variables las utilizamos para que el Tiranosaurio
    reconozca el piso y en caso contrario caiga a un piso inferior */
    private boolean enElSuelo = true;
    private double velocidadY = 0;
    private double gravedad = 0.3;

    //----------------Constructores----------------//
    public Tiranosaurio() {}
    public Tiranosaurio(int x, int y, int ANCHO, int ALTO) {
        this.x = x;
        this.y = y;
        this.ANCHO = ANCHO;
        this.ALTO = ALTO;
        this.imagen = Herramientas.cargarImagen("dino.gif");
        this.angulo = 0;
        this.escala= 1;
        this.piso = this.y + this.ALTO/2;
    }
    //----------------Getters y Setters----------------//
    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }
```

```java
public int getY() {
    return y;
}

public void setY(int y) {
    this.y = y;
}

public double getANCHO() {
    return ANCHO;
}

public void setANCHO(int ANCHO) {
    this.ANCHO = ANCHO;
}

public double getALTO() {
    return ALTO;
}

public void setALTO(int ALTO) {
    this.ALTO = ALTO;
}

public double getAngulo() {
    return angulo;
}

public void setAngulo(double angulo) {
    this.angulo = angulo;
}

public double getEscala() {
    return escala;
}

public void setEscala(double escala) {
    this.escala = escala;
}

public boolean getDisparoLadoTiranosaurios() {
    return disparoLadoTiranosaurios;
}

public void setDisparoLadoTiranosaurios(boolean disparoLadoTiranosaurios) {
    this.disparoLadoTiranosaurios = disparoLadoTiranosaurios;
}
```

```java
public int getPiso() {
    return piso;
}
public void setPiso(int piso) {
    this.piso = piso;
}
public boolean getEnElSuelo() {
    return enElSuelo;
}
public void setEnElSuelo(boolean enElSuelo) {
    this.enElSuelo = enElSuelo;
}
//----------------Metodos-----------------//
public void movIzq() {
    if(tocoPantallaIzq == false) {
        this.x -= 1;
        disparoLadoTiranosaurios = true;
    }
    else if(tocoPantallaIzq == true){
        this.x = this.x+=1;
        disparoLadoTiranosaurios = false;
    }
}

public boolean tocoPantallaDinoComienzaSentidoIzq() {
    if(this.x < 31) {
        tocoPantallaIzq = true;
    }
    if(this.x > 975) {
        tocoPantallaIzq = false;
    }
    return tocoPantallaIzq;
}

public void movDer() {
    if(tocoPantallaDer == false) {
        this.x += 1;
        disparoLadoTiranosaurios = false;
    }
    else if(tocoPantallaDer == true){
        this.x = this.x-=1;
        disparoLadoTiranosaurios = true;
    }
}
```

```java
public void movDer() {
    if(tocoPantallaDer == false) {
        this.x += 1;
        disparoLadoTiranosaurios = false;
    }
    else if(tocoPantallaDer == true){
        this.x = this.x-=1;
        disparoLadoTiranosaurios = true;
    }
}

public boolean tocoPantallaDinoComienzaSentidoDer() {
    if(this.x < 31) {
        tocoPantallaDer = false;
    }
    if(this.x > 975) {
        tocoPantallaDer = true;
    }
    return tocoPantallaDer;
}

public void crearTiranosaurio(ArrayList<Tiranosaurio> tiranosaurios, Tiranosaurio [] tiranosaur ) {
    for (int i = 0; i < tiranosaur.length; i++) {
        tiranosaurios.add(tiranosaur[i]);
    }
}

public void dibujar(Entorno entorno , ArrayList<Tiranosaurio> tiranosaurios) {
    for (int i = 0; i < tiranosaurios.size(); i++) {
        Tiranosaurio tira = tiranosaurios.get(i);
            entorno.dibujarImagen(imagen, tira.getX(), tira.getY()-15, tira.getAngulo(), tira.getEscala());
    }
}

public void Caida() {
    // Aplicar gravedad en cada frame
    if(!enElSuelo) {
        velocidadY += gravedad;
        this.y += velocidadY;
    }
    if(this.y >= this.piso) {
        this.setY(this.piso);
        this.velocidadY = 0;
        this.enElSuelo = true;
    }
}
```

Conclusión:

Consideramos que fué un gran desafio y hemos tenido un enorme aprendizaje acerca de la utilización de la programación orientada a objetos, en un principio nos costó un poco acostumbrarnos a la utilización de clases y encapsulamiento, sin embargo, pudimos tomar ritmo y llegamos a un avance significativo que nos pone muy contentos y nos da una gran satisfacción.