Universidad Nacional de General Sarmiento

Trabajo Práctico:

Al rescate de los gnomos

Grupo 11

Integrantes:

- Vanesa Licero Puma. DNI:95124735. licerovanesa@gmail.com
- Rocio Gonzalez. DNI: 4119161. roccio.gonzz@gmail.com
- Carolina Gota. DNI: 44690278 carito.gota@gmail.com

Profesores:

- César Niveyro
- Nahuel Sauma

Introducción:

Se nos entregó el paquete entorno.jar y nuestro trabajo es realizar la implementación necesaria para la inteligencia del juego.

El juego está basado en islas flotantes siendo la más alta la casa de los gnomos y estos están siendo atacados por tortugas que caen del cielo excepto en la casa, los gnomos careciendo de mucha inteligencia salen corriendo de su casa y terminan cayendo al vacío o envenenados por las tortugas. En esta situación los gnomos deciden contratar los servicios de Pep, un caballero medieval.

El objetivo del juego es crear un simulador en el que Pep vaya rescatando a los gnomos.

Descripción:

Para llevar adelante la ejecución del juego tuvimos que implementar diferentes clases y métodos. Implementamos una clase por objeto, la clase **Personaje**, **BolaDeFuego**, **Gnomos**, **CasaDeLosGnomos**, **Islas**, **Tortugas** y además una clase auxiliar llamada **MetodosParaJuego**.

La clase **Personaje** cuenta con:

- **Variables:** x, y, ancho, alto, desplazamiento, booleano direccionDer (si está en true el personaje va para la derecha, de lo contrario va para la izquierda), booleano saltando, alturaSalto (se le da un entero como límite de salto), booleano estaApoyado, imágenes Izq y Der.
- **Métodos:**
  - public Personaje(double x, double y, double ancho, double alto, double desplazamiento, boolean direccionDer): es el constructor del objeto Personaje.
  - public void dibujarse(Entorno e): dibuja en pantalla con las imágenes dadas según la dirección del personaje.
  - public void moverArriba(): disminuye el valor de y del personaje.
  - public void moverAbajo(): incrementa el valor de y del personaje si no está apoyado.
  - public void moverDerecha(): si está apoyado o saltando, el booleano se pone en true, e incrementa el x del personaje dándole movilidad hacia la derecha.

- public void moverIzquierda(): si está apoyado o saltando, el booleano se pone en false, y disminuye el x del personaje dándole movilidad hacia la izquierda.
- public void saltar(): este método es para que el personaje salte.
- public boolean colisionaPorDerecha(Entorno e): detecta si el personaje colisiona con el borde derecho de la pantalla.
- public boolean colisionaPorIzquierda(Entorno e): detecta si el personaje colisiona con el borde izquierdo de la pantalla.
- public boolean colisionaPorArriba(Entorno e): detecta el tope que se le da al personaje para que no pueda subir a la isla donde está la casa de los gnomos.
- public boolean colisionaPorAbajo(Entorno e): detecta la colisión del personaje con el borde inferior de la pantalla.
- public boolean estaColisionandoPorAbajo(Islas[] islas): detecta si el borde inferior del personaje colisiona con el borde superior de la isla.
- public boolean estaColisionandoPorArriba(Islas[] islas): detecta si el borde superior del personaje colisiona con el borde inferior de la isla.
- public boolean estaColisionandoPorDerecha(Islas[] islas): detecta si el borde derecho del personaje colisiona con el borde izquierdo de la isla.
- public boolean estaColisionandoPorIzquierda(Islas[] islas): detecta si el borde izquierdo del personaje colisiona con el borde derecho de la isla.
- public boolean colisionConTortuga(Tortugas t): detecta si el personaje colisiona con el objeto Tortuga.
- public boolean colisionConGnomo(Gnomo gnomo): detecta si el personaje con el objeto Gnomo.
- Getters de las variables: x, y, ancho, alto, direccionDerecha

La clase **BolaDeFuego** cuenta con:

- **Variables**: x, y, radio, velocidad, alto, dirección, imágenes Izq y Der
- **Métodos:**
  - public BolaDeFuegoPersonaje(double x, double y, boolean activo, boolean direccion): es el constructor.
  - public void dibujar(Entorno entorno): dibuja en la pantalla tomando las imágenes Izq y Der según su dirección.
  - public void mover(Personaje p): le da movilidad a la bola de fuego según la dirección que el personaje lo lanza.
  - public boolean estaColisionandoPorDerecha(Islas[] islas): detecta si el borde derecho de la bola de fuego colisiona con el borde izquierdo de la isla.
  - public boolean estaColisionandoPorIzquierda(Islas[] islas): detecta si el borde izquierdo de la bola de fuego colisiona con el borde derecho de la isla.

- public boolean colisionaPorDerecha(Entorno e): detecta si la bola de fuego colisiona con el borde derecho de la pantalla.
- public boolean colisionaPorIzquierda(Entorno e): detecta si la bola de fuego colisiona con el borde izquierdo de la pantalla.
- Getters de las variables: x, y, radio.
- Setters de las variables: x, y, radio.

La clase **Gnomo**

- **Variables:** x, y, ancho, alto, movimientoHorizontal (si es igual a 1 el gnomo va para la derecha y si vale –1 va para la izquierda), velocidad (toma un valor entero para la velocidad del gnomo), random(crea un objeto ramdon que se utiliza en el método cambiarDireccionAleatoria()), enIsla (es un booleano que no indica si el gnomo está en isla), y por ultimo las imágenes Izq y Der.
- **Métodos:**
  - public Gnomo(double x, double y, double ancho, double alto, double despl, int vel) : es el constructor del objeto Gnomo.
  - public void dibujar(Entorno e): dibuja el objeto con las imágenes según la dirección del personaje.
  - public void mover(): le da movilidad al gnomo.
  - public void cambiarDireccionAleatoria(): toma un numero ramdon entre cero y uno, si es cero el movimiento cambia hacia la izquierda y si es uno cambia hacia la derecha.
  - public boolean hayColisionDerecha(Entorno e): detecta el borde derecho de la pantalla.
  - public boolean hayColisionIzquierda(Entorno e): detecta el borde izquierdo de la pantalla.
  - public void cambiarMovimiento(): cambia de dirección del gnomo.
  - public void caer(Entorno e): este método incrementa la variable y del gnomo generando gravedad.
  - public boolean colisionIsla (Islas[] is): detecta si el gnomo colisiona con el borde superior de la isla.
  - public boolean bordeInferiorEntorno(Entorno e): detecta el borde inferior de la pantalla.
  - public boolean colisionConTortuga(Tortugas t): detecta si el gnomo colisiona con una tortuga.
  - public boolean colisionConPersonaje(Personaje pep): detecta si el gnomo colisiona con el personaje.
  - Getters de las variables x, y, alto, ancho, enIsla.
  - Setters de las variables x, y, alto, ancho, enIsla.

La clase **CasaDeLosGnomos**

- **Variables:** x e y, la imagen
- **Métodos:**
  - public CasaDeLosGnomos(double x, double y, double ancho, double alto): Es el constructor del objeto CasaDeLosGnomos
  - public void dibujar(Entorno e): este método nos imprime en pantalla el objeto con la imagen.

La clase **Islas** contiene:

- **Variables**: x, y, ancho y alto del objeto, y una variable de imagen isla.
- **Métodos**:
  - void dibujar(): dibuja la variable isla en el entorno.
  - getters y setters de x, y, ancho y alto.

La clase **Tortugas** contiene:

- **Variables**: x, y, alto y ancho del objeto, desplazamiento, velocidad y dos variables de imagen izq y der.
- **Métodos**:
  - void dibujar(): dependiendo del desplazamiento dibuja la imagen izq o der en el entorno.
  - void caer(): incrementa la posición y.
  - void moverDerecha(): incrementa la posición x con el desplazamiento multiplicado por la velocidad.
  - void moverIzquierda(): disminuye la posición x con el desplazamiento multiplicado por la velocidad.
  - void cambiarMovimiento(): multiplica el desplazamiento por menos uno.
  - boolean colisionaPorDerecha(Entorno e): detecta si la posición x del objeto es mayor o igual al ancho del entorno.
  - boolean colisionaPorIzquierda(Entorno e): detecta si la posición x del objeto es menor o igual a cero.
  - boolean estaColisionandoPorAbajo(Islas [] islas): detecta si el objeto esta sobre las islas.
  - boolean llegaAlBorde(Islas [] islas): detecta si el objeto está en los bordes de las islas.
  - boolean colisionBolaDeFuego(BolaDeFuegoPersonaje b): detecta si el objeto colisiona con la BolaDeFuego.
  - getters y setters de x, y, alto y ancho.

La clase **MetodosParaJuego** contiene:

- **Métodos:**
  - void mostrarYouWin(int gnomoSalvado, Juego juego): muestra el mensaje YouWin en el entorno junto con la cantidad de gnomoSalvado.

- void mostrarGameOver(int gnomoSalvado, Juego juego): muestra el mensaje GameOver en el entorno junto con la cantidad de gnomoSalvado.
- void reiniciarJuego(Juego juego): llama al método reiniciarJuego() que se encuentra en la clase Juego.
- Islas[] crearIslas(Entorno e): dibuja un arreglo de islas en forma de pirámide en el entorno.
- void agregarGnomo(Gnomo[] gnomos,Entorno entorno): si un gnomo queda en null dibuja otro.
- void agregarTortuga(Tortugas[] tortugas, Entorno entorno, int posXinferior, int posXsuperior): dibuja tortugas en una posición x aleatoria sin tener en cuenta las posiciones posXinferior a posXsuperior, además también llama a los métodos posicionOcupada y estaLejos.
- boolean posicionOcupada(int x, Tortugas[] tortugas): verifica si hay una tortuga en la posición.
- boolean estaLejos(int newX, Tortugas[] tortugas): verifica una distancia mínima entre las tortugas.

Implementación:

Código de la clase Juego:

```
package juego;


import java.awt.Color;

import java.awt.image.*;

import java.io.IOException;

import java.io.InputStream;

import javax.imageio.ImageIO;

import entorno.Entorno;

import entorno.InterfaceJuego;


public class Juego extends InterfaceJuego {

   // El objeto Entorno que controla el tiempo y otros
```

```
    private Entorno entorno;

    // Variables y métodos propios de cada grupo
    private CasaDeLosGnomos casa;
    private Gnomo[] gnomos;
    private Tortugas[] tortugas;
    private Islas[] islas;
    private Personaje personaje;
    private BolaDeFuegoPersonaje bolaDeFuego;

    private int posXinferior;
    private int posXsuperior;

    //variables de tiempo para los gnomos
    private long lastGnomoTime;
    private final int tiempoSpawneo = 3000; // 3 segundos en milisegundos
    //contadores en pantalla
    private int contadorBordeInferior;
    private int contadorColisionTortugas;
    private int gnomoSalvado;

    //tiempo de juego
    private long tiempoJuego; // variable para almacenar el tiempo de juego
    private long lastUpdateTime; // para calcular el tiempo transcurrido entre ticks
```

```
    private BufferedImage imagenFondo;


    Juego()
    {
        // Inicializa el objeto entorno
        this.entorno = new Entorno(this, "Al rescate de los Gnomos", 1000, 700);


        // Inicializar lo que haga falta para el juego
       this.personaje = new Personaje (entorno.ancho()- (entorno.ancho()/25), entorno.alto()/2, 20, 60, 3, true);
       this.casa = new CasaDeLosGnomos(entorno.ancho()/2, entorno.alto()-(entorno.alto()-75), 60, 75);


        this.gnomos = new Gnomo[4];
        this.tortugas= new Tortugas[8];
        islas = MetodosParaJuego.crearIslas(entorno);


        //posicion prohibida para tortuga
        this.posXinferior=entorno.ancho()/2-100;
        this.posXsuperior=entorno.ancho()/2+100;


        this.lastGnomoTime = System.currentTimeMillis(); // Inicializa el temporizador
        this.contadorBordeInferior = 0;
        this.contadorColisionTortugas = 0;
        this.gnomoSalvado = 0;
        this.tiempoJuego = 0; // iniciar el tiempo desde cero
```

```
        this.lastUpdateTime = System.currentTimeMillis(); // inicializar el tiempo de la última actualización


        // Cargar la imagen de fondo
        try {
            InputStream is = getClass().getResourceAsStream("/imagenes/imagenFondo.jpg.jpg");
            if (is != null) {
                imagenFondo = ImageIO.read(is);
            } else {
                System.err.println("No se encontró la imagen de fondo.");
            }
        } catch (IOException e) {
            System.err.println("Error al cargar la imagen de fondo: " + e.getMessage());
            e.printStackTrace();
        }


        // Inicia el juego!
        this.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() {

        //procesamiento de un instante de tiempo

        long currentTime = System.currentTimeMillis();

       long deltaTime = currentTime - lastUpdateTime; // Calcular el tiempo transcurrido desde la última actualización

        tiempoJuego += deltaTime;// Actualizar el tiempo de juego

        lastUpdateTime = currentTime;// Actualizar lastUpdateTime para la próxima tick

        // ...


        if(imagenFondo!=null) {//dibuja la imagen de fondo del juego

          entorno.dibujarImagen(imagenFondo,  entorno.ancho()-entorno.ancho()/2, entorno.alto()-(entorno.alto()/2), 0);

        }


        casa.dibujar(entorno);//dibuja la casa de los gnomos


        long tiempoDeJuego = System.currentTimeMillis();


        // Crear un nuevo gnomo cada 3 segundos si hay espacio

        if (tiempoDeJuego - lastGnomoTime >= tiempoSpawneo) {

            MetodosParaJuego.agregarGnomo(gnomos,entorno);

            lastGnomoTime = tiempoDeJuego;

        }

        //dibuja los gnomos

        for (int i = 0; i < gnomos.length; i++) {

            Gnomo gnomo = gnomos[i];
```

```
            if(gnomo != null) {
                gnomo.dibujar(entorno);
                gnomo.caer(entorno);

                //verifica colision cambia la direccion de manera aleatoria dentro del metodo
                if(gnomo.colisionIsla(islas)){
                    gnomo.mover();
                }

                //Verifica los laterales del entorno y cambia de direccion
                if (gnomo.hayColisionDerecha(entorno) || gnomo.hayColisionIzquierda(entorno))
{
                    gnomo.cambiarMovimiento();
                }
                //verifica que el gnomo cae al vacio y lo elimina
                if (gnomo.bordeInferiorEntorno(entorno)){
                    gnomos[i] = null;
                    contadorBordeInferior++; // Incrementar contador gnomos precipitados
                    //System.out.print("gnomo fuera ");
                }
                // Verificar colisión con tortugas
                for (Tortugas tortuga : tortugas) {
                    if (tortuga != null && gnomo.colisionConTortuga(tortuga)) {
                        gnomos[i] = null; // Eliminar el gnomo
                        contadorColisionTortugas++; // Incrementar contador
                        break;
```

```
                }
            }
            //gnomo colision con personaje
            if(gnomo.colisionConPersonaje(personaje)&&(gnomo.getY()>entorno.alto()/2)){
                gnomos[i] = null; // Pep salva Gnomo
                //System.out.println("pepGnomo... ");
                gnomoSalvado++;// incrementar contador
            }
        }
    }

    // dibujo las islas
    for (Islas isla : islas) {
        if (islas != null && islas.length > 0) {
            isla.dibujar(entorno);
        }
    }
    // dibujo las tortugas
    for (int i = 0; i < tortugas.length; i++) {
        Tortugas tortuga = tortugas[i];
        if(tortuga!=null){
            tortuga.dibujar(entorno);
            tortuga.caer();

            //colision tortugas - islas
```

```
            if(tortuga.estaColisionandoPorAbajo(islas)) {
                tortuga.moverIzquierda();

                //colision tortugas - entorno
              if (tortuga.colisionaPorDerecha(entorno) ||
tortuga.colisionaPorIzquierda(entorno)) {
                  tortuga.cambiarMovimiento();
                }
                //movimiento tortugas sobre islas
                if(!tortuga.llegaAlBorde(islas)) {
                  tortuga.cambiarMovimiento();
                }
                //colision bolaDeFuego-tortuga
                if(tortuga.colisionBolaDeFuego(bolaDeFuego)) {
                  tortugas[i]=null;
                  bolaDeFuego=null;
                }
            }
        }
        else {
            //si una tortuga queda en null

MetodosParaJuego.agregarTortuga(tortugas,entorno,this.posXinferior,this.posXsuperior)
; //dibuja otra tortuga
        }
    }
    if (personaje != null){
```

```
        //dibujo del personaje

        personaje.dibujarse(entorno);


        //Colisiones personaje - entorno

       if(entorno.estaPresionada(entorno.TECLA_DERECHA)                    &&
!personaje.colisionaPorDerecha(entorno)                                    &&
!personaje.estaColisionandoPorDerecha(islas)) {

           personaje.moverDerecha();

        }

       if(entorno.estaPresionada(entorno.TECLA_IZQUIERDA)                  &&
!personaje.colisionaPorIzquierda(entorno)                                  &&
!personaje.estaColisionandoPorIzquierda(islas)) {

           personaje.moverIzquierda();

        }

       if(entorno.estaPresionada(entorno.TECLA_ARRIBA)                     &&
!personaje.colisionaPorArriba(entorno) && !personaje.estaColisionandoPorArriba(islas)) {

           personaje.saltar();}


       if                (!personaje.estaColisionandoPorAbajo(islas)       ||
personaje.estaColisionandoPorArriba(islas))

           personaje.moverAbajo();


        //verifica que Pep cae al vacio y lo elimina

        if (personaje!= null && personaje.colisionaPorAbajo(entorno)) {

           personaje = null;

           MetodosParaJuego.mostrarGameOver(gnomoSalvado, this);// gameover

           //System.out.print("Pep muerto ");

        }
```

```
        //verifica si el pep rescato el objetivo de gnomos para finalizar el juego
        if(personaje!= null && gnomoSalvado == 10) {
            personaje = null;
            MetodosParaJuego.mostrarYouWin(gnomoSalvado, this);// youwin
            //System.out.println("GANO ... ");
        }
        // Verificar colisión con tortugas
      for (Tortugas tortuga : tortugas) {
         if (tortuga != null && personaje!=null && personaje.colisionConTortuga(tortuga)) {
                personaje = null; // Eliminar a Pep
                MetodosParaJuego.mostrarGameOver(gnomoSalvado, this);// gameover
                //System.out.println("peptortu ... ");
                break;
            }
        }
        // se crea una bola de fuego
        if(entorno.sePresiono(entorno.TECLA_ESPACIO) && bolaDeFuego==null) {
       this.bolaDeFuego = new BolaDeFuegoPersonaje(personaje.getX(),
personaje.getY(), true,personaje.getdireccionDerecha());
        }
        if(this.bolaDeFuego!=null) {
            //dibuja una bola de fuego
            bolaDeFuego.dibujar(entorno);
            bolaDeFuego.mover(personaje);
            //colision con entorno, islas o tortugas de la bola de fuego
```

```
if(bolaDeFuego.colisionaPorDerecha(entorno)||bolaDeFuego.colisionaPorIzquierda(entorno) ||bolaDeFuego.estaColisionandoPorDerecha(islas) || bolaDeFuego.estaColisionandoPorIzquierda(islas)) {
                bolaDeFuego=null;
            }
          }
       }
       // Dibujar contadores en la parte superior
       entorno.cambiarFont("Arial", 18, Color.black);
       entorno.escribirTexto("Gnomos perdidos: " + contadorBordeInferior, 20, 30);
      entorno.escribirTexto("Gnomos eliminados x tortugas: " + contadorColisionTortugas, 20, 50);
       entorno.escribirTexto("Gnomos salvados: " + gnomoSalvado, 20, 80);

       // Calcular minutos y segundos
       long totalSegundos = tiempoJuego / 1000;
       long minutos = totalSegundos / 60;
       long segundos = totalSegundos % 60;

       // Mostrar el tiempo de juego en formato minutos:segundos
      entorno.escribirTexto(String.format("Tiempo de juego: %02d:%02d", minutos, segundos), 20, 100);
   }

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

```
    }

    //metodo para reiniciar el juego

    public  void reiniciarJuego() {

        // Reinicia el personaje

     this.personaje = new Personaje(entorno.ancho() - (entorno.ancho() / 25), entorno.alto()
/ 2, 20, 60, 3, true);



        // Reinicia la casa

     this.casa = new CasaDeLosGnomos(entorno.ancho() / 2, entorno.alto() - (entorno.alto()
- 75), 60, 75);



        // Reinicia los gnomos

        this.gnomos = new Gnomo[4];

        this.lastGnomoTime = System.currentTimeMillis(); // Reinicia el temporizador

        this.contadorBordeInferior = 0;

        this.contadorColisionTortugas = 0;

        this.gnomoSalvado = 0;



        // Reinicia las tortugas

        this.tortugas = new Tortugas[9];



        // Reinicia las islas

        this.islas = MetodosParaJuego.crearIslas(entorno);



        // Reinicia la bola de fuego

        this.bolaDeFuego = null;
```

```
        //variables que necesiten reiniciarse
        this.tiempoJuego=0;

        // Vuelve a dibujar el entorno inicial
        entorno.repaint();
    }
}
```

Conclusión:

En conclusión, pudimos llevar adelante la resolución de todo el trabajo de manera correcta, corrigiendo en el camino alguno errores que se nos fueron presentando. Entre las integrantes del grupo pudimos resolver los problemas o consultando a los profesores. Llegamos a implementar todos los puntos obligatorios del TP para tener un juego eficiente donde el personaje trate de salvar a los gnomos evitando a las tortugas y se desarrolle en el entorno de juego.

En muchos casos durante la ejecución del juego tuvimos problemas que fácilmente fueron resueltos después de mover una parte del código dentro de algún if que ya estaba presente, nos dimos cuenta que a pesar de nuestra correcta ejecución de los métodos si alguna parte del código se llama antes que otra o si ponemos alguna parte dentro de algún if que no corresponde entonces nuestro código nos mostrara un error. Un ejemplo de ello fue con personaje.dibujar() que se encontraba por fuera del if(personaje!=null), debido a esto tuvimos un error cada vez que personaje estaba en null.