Título: El Juego: Al rescate de los gnomos.

Materia: Programación I

Integrantes:

 Sebastián Alberto Sosa - E-mail: [sebasosa98@outlook.com.ar](mailto:sebasosa98@outlook.com.ar)

      Nahuel Lencina - Email: [nahuel.hxc@gmail.com](mailto:nahuel.hxc@gmail.com)

 Federico daniel Maldonado - Email: [maldonadofederico@gmail.com](mailto:maldonadofederico@gmail.com)

 Sabrina Belén Rodriguez Q. - Email: [rodriguezquinterosbelen@gmail.com](mailto:rodriguezquinterosbelen@gmail.com)

Fecha de entrega: 29/10/2024

**Introducción**

Este trabajo práctico tiene como objetivo desarrollar un video juego, donde Pep vaya rescatando a los gnomos antes de que caigan al vacío o sean destruidos por las tortugas de caparazones con púas.

El proyecto fue implementado en lenguaje JAVA partimos de un paquete base que contiene la clase Entorno y la clase Herramientas. Al instanciar y crear el objeto, el mismo puede ser capaz de encargarse de la interfaz gráfica y la interacción con el usuario. IDE de desarrollo Eclipse.

**Descripción del Trabajo**

**Clases Implementadas**

Clase 1: Juego

Título: El Juego: Al rescate de los gnomos.

Materia: Programación I

Integrantes:

 Sebastian Alberto Sosa - E-mail: [sebasosa98@outlook.com.ar](mailto:sebasosa98@outlook.com.ar)

      Nahuel Lencina - Email: [nahuel.hxc@gmail.com](mailto:nahuel.hxc@gmail.com)

 Federico daniel Maldonado - Email: [maldonadofederico@gmail.com](mailto:maldonadofederico@gmail.com)

 Sabrina Belen Rodriguez Q. - Email: [rodriguezquinterosbelen@gmail.com](mailto:rodriguezquinterosbelen@gmail.com)

Fecha de entrega: 29/10/2024

**Clases Implementadas**

**Clase 1: Juego**

Variables de instancia

**private Entorno entorno;**

El objeto que representa la ventana gráfica y permite dibujar en pantalla.

**private Image fondo;**

Imagen de fondo del juego.

**private Image casita;**

Imagen de la casita que se dibuja en el escenario.

**private Gnomo[] gnomos;**

Array que contiene los objetos de tipo Gnomo en el juego.

**private Tortugas[] tortugas;**

Array que contiene las tortugas.

**private Isla[] islas;**

Array que almacena las islas colocadas en el escenario.

**private int vidaActual = 3;**

Contador de vidas disponibles del jugador (Pep).

**private int tickContador = 0;**

Contador que mide los "ticks" o instantes del juego.

**private int tiempoParaAparecer;**

Ticks necesarios hasta que aparezca un nuevo gnomo o tortuga.

**private Random random = new Random();**

Generador de números aleatorios utilizado para la lógica de aparición.

**private Pep pep;**

Objeto Pep, el personaje principal del juego.

**private int direccion = 1;**

Indica la dirección en la que se mueven los personajes (1 = derecha, -1 = izquierda).

**private boolean enMovimiento = false;**

Bandera para indicar si el juego está en movimiento tras presionar ENTER.

**Inicialización del entorno y objetos**:

* Configura la ventana del juego mediante el objeto Entorno.
* Carga las imágenes necesarias, como el fondo, la casita, y las islas.
* Inicializa los elementos del juego (gnomos, tortugas, Pep) para que estén listos para interactuar.

**Control del ciclo del juego**:

* Implementa el método tick(), que se ejecuta en cada instante del juego. Este método actualiza los estados de los objetos, verifica las entradas del usuario y maneja la aparición de nuevos elementos.
* Usa un contador de tiempo (tickContador) para controlar cuándo deben aparecer nuevos gnomos y tortugas.

**Interacción del usuario**:

* Controla los movimientos del personaje principal, *Pep*, usando las teclas de dirección (izquierda, derecha, arriba).
* Permite que *Pep* dispare mediante la tecla c.

**Detección de colisiones**:

* Verifica si los gnomos o tortugas están sobre las islas.
* Detecta colisiones entre gnomos y tortugas, lo que reduce la vida del jugador.
* Implementa lógica para que los personajes se muevan de forma coherente y rebote en los bordes de las islas.

**Clase 2: isla**

**private int x;**

* + Coordenada horizontal de la posición de la isla.

**private int y;**

* + Coordenada vertical de la posición de la isla.

**private int ancho;**

* + Anchura de la isla.

**private int alto;**

* + Altura de la isla.

**private Image image;**

* + Imagen que representa visualmente la isla.

**Clase 3: Gnomo**

### ****Variables de Instancia****

**private int x;**

Posición horizontal del gnomo en la pantalla (coordenada X).

**private double y;**

Posición vertical del gnomo (coordenada Y).

**private int direccion;**

Indica la dirección en la que se mueve el gnomo (positivo para derecha, negativo para izquierda).

**private int ancho;**

Ancho del gnomo.

**private int alto;**

Alto del gnomo.

**private Image image;**

Imagen que representa visualmente al gnomo.

**Clase 4: Tortuga**

Variables de instancia:

**private int direccion;**

* Almacena la dirección o velocidad del movimiento de la tortuga (puede ser positiva o negativa).

**private double y, x;**

* **y**: Coordenada vertical de la posición de la tortuga.
* **x**: Coordenada horizontal de la posición de la tortuga.

**private int ancho, alto;**

* **ancho**: Anchura del objeto Tortugas.
* **alto**: Altura del objeto Tortugas.

**private Image image;**

* Referencia a la imagen utilizada para representar gráficamente la tortuga.

**Clase 5: Pep**Variables de instancia:

private Image imagen;

private int x, y;

private int direccion; // 1 = derecha, -1 = izquierda

private boolean saltando;

private int velocidadY;

private int direccionUltima;

Problemas y soluciones aplicadas para el desarrollo del Tp.

Para trabajar con el entorno gráfico, los requisitos mínimos eran JAVA 17 y una de nuestras PC cuenta con Windows 7 y un procesador de 32 bits. La aplicación nos lanzaba una excepción y no ejecutaba el juego. No hay actualizaciones de JAVA 17 para x86.

Solución encontrada:

Descargamos una versión desde openjdk Java 17 y funciono perfecto.

CODIGO:

Clase Juego

package juego;

import java.awt.Image;

import entorno.Entorno;

import entorno.Herramientas;

import entorno.InterfaceJuego;

import java.util.Random;

public class Juego extends InterfaceJuego

{

//Variables de instancia

private Entorno entorno;

private Image fondo;

private Image casita;

private Gnomo[] gnomos;

private Tortugas[] tortugas;

private Isla[] islas;

private int vidaActual=3;

private int tickContador = 0; // Contador de ticks

private int tiempoParaAparecer; // Ticks hasta la próxima aparición

private Random random = new Random(); // Generador de números aleatorios

private Pep pep; // Agregamos a Pep

private int direccion = 1;

private boolean enMovimiento = false;

// Metodos

Juego()

{

// Inicializa el objeto entorno

this.entorno = new Entorno(this, "Al Rescate de los Gnomos", 800, 600);

// Inicializar lo que haga falta para el juego

this.islas = new Isla[15]; // Declaramos la lista de islas

this.gnomos = new Gnomo[5]; // Declaramos la lista de gnomos

this.tortugas = new Tortugas[4];

this.pep = new Pep(Herramientas.*cargarImagen*("Imagenes/Pep.png"), entorno.ancho()/2, 370);

// Se llama al metodo cargar el fondo

cargarFondo();

// Se llama al metodo para cargar las islas

cargarIslas();

// Se llama al metodo para cargar la casita

cargarCasita();

// 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()

{

if(entorno.sePresiono(entorno.TECLA\_ENTER) || enMovimiento==true) {

if(vidaActual>0){

actualizar();

enMovimiento=true;

}

else {

entorno.escribirTexto("Game Over"+ vidaActual, entorno.ancho()/2, 550);

}

}

// Procesamiento de un instante de tiempo

imprimirFondo();

imprimirCasita();

dibujarIslas();

actualizarGnomos();

actualizarTortugas();

actualizarPep();

// Control de teclas para Pep

if (entorno.estaPresionada(entorno.TECLA\_IZQUIERDA) || entorno.sePresiono('a')) {

pep.moverIzquierda();

} else if (entorno.estaPresionada(entorno.TECLA\_DERECHA) || entorno.sePresiono('d')) {

pep.moverDerecha();

}

if (entorno.sePresiono(entorno.TECLA\_ARRIBA) || entorno.sePresiono('w')) {

pep.saltar();

}

if (this.entorno.sePresiono('c')) {

pep.disparar();

}

}

public void actualizarPep() {

// Actualizamos a Pep

pep.dibujar(this.entorno);

pep.actualizar();

if (!pepSobreIsla(pep) && !pep.estaSaltando()) {

pep.setY(pep.getY() + 1); // Pep cae si no está sobre una isla o saltando

}

}

private boolean pepSobreIsla(Pep pep) {

for (Isla isla : islas) {

if (isla != null) {

boolean tocandoX = pep.getX() >= isla.getX() - isla.getAncho() / 2 &&

pep.getX() <= isla.getX() + isla.getAncho() / 2;

boolean tocandoY = pep.getY() >= isla.getY() - isla.getAlto() / 2 &&

pep.getY() <= isla.getY() + isla.getAlto() / 2;

if (tocandoX && tocandoY) {

return true;

}

}

}

return false;

}

private void imprimirFondo() {

// Imprimimos el fondo en la ventana, usamos la sobreCarga del metodo dibujarImagen para ampliar la misma

this.entorno.dibujarImagen(fondo, entorno.ancho() / 2, entorno.alto()/2, 0, 1.1);

}

private void imprimirCasita(){

this.entorno.dibujarImagen(casita, entorno.ancho()/ 2, 80, 0, 0.2);

}

private void dibujarIslas() {

for (Isla isla : islas) {

entorno.dibujarImagen(isla.getImagen(), isla.getX(), isla.getY(), 0, 0.3);

}

}

private void moverGnomo(Gnomo gnomo, int index) {

gnomo.setY(gnomo.getY() + 1);

if (gnomo.getY() > 550 || colisionConTortuga(gnomo)) {

this.gnomos[index] = null; // Eliminar el gnomo si cae o colisiona

} else {

int nuevaDireccion = direccionAleatorio(direccion);

gnomo.setDireccion(nuevaDireccion);

}

}

private void actualizarGnomos() {

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

Gnomo gnomo = gnomos[i];

if (gnomo != null) {

gnomo.dibujar(this.entorno);

if (!gnomoSobreIsla(gnomo)) {

moverGnomo(gnomo, i);

} else {

gnomo.mover();

if(colisionConTortuga(gnomo)) {

this.gnomos[i] = null;

// System.out.println("Muere Gnomo");

}

}

}

}

}

private void actualizarTortugas() {

for (Tortugas tortuga : tortugas) {

if (tortuga != null) {

tortuga.dibujar(this.entorno);

if (tortugaSobreIsla(tortuga)) {

tortuga.mover();

if (colisionBordeIsla(tortuga)) {

tortuga.setDireccion(-tortuga.getDireccion());

}

} else {

tortuga.setY(tortuga.getY() + 2);

}

}

}

}

// Método para verificar si un gnomo está sobre alguna isla

private boolean gnomoSobreIsla(Gnomo gnomo) {

for (Isla isla : islas) {

if (isla != null) {

// Verifica si el gnomo está dentro de los límites de la isla

boolean tocandoX = gnomo.getX() >= isla.getX() - isla.getAncho() / 2 &&

gnomo.getX() <= isla.getX() + isla.getAncho() / 2;

boolean tocandoY = gnomo.getY() >= isla.getY() - isla.getAlto() / 2 &&

gnomo.getY() <= isla.getY() + isla.getAlto() / 2;

if (tocandoX && tocandoY) {

return true; // El gnomo está sobre esta isla

}

}

}

return false; // El gnomo no está sobre ninguna isla

}

private boolean colisionConTortuga(Gnomo gnomo) {

int margen = 20; // Margen de colisión

for (Tortugas tortuga : tortugas) {

if (gnomo != null && tortuga != null) {

// Verificar si las posiciones están dentro del margen

boolean colisionX = Math.*abs*(tortuga.getX() - gnomo.getX()) < margen;

boolean colisionY = Math.*abs*(tortuga.getY() - gnomo.getY()) < margen;

if (colisionX && colisionY) {

System.***out***.println("Muere Gnomo");

vidaActual--;

return true;

}

}

}

return false;

}

private boolean colisionBordeIsla(Tortugas tortuga) {

for (Isla isla : islas) {

// Calcula los límites de la isla

int limiteIzquierdo = isla.getX() - isla.getAncho() / 2;

int limiteDerecho = isla.getX() + isla.getAncho() / 2;

int limiteSuperior = isla.getY() - isla.getAlto() / 2;

int limiteInferior = isla.getY() + isla.getAlto() / 2;

// Verifica si la tortuga está dentro de los límites de la isla (con margen de 10)

if( (tortuga.getX()-10) == limiteIzquierdo && tortuga.getY() >= limiteSuperior && tortuga.getY() <= limiteInferior) {

tortuga.setX(tortuga.getX()+5);

return true;

}

if(tortuga.getX()+10 == limiteDerecho && tortuga.getY() >= limiteSuperior && tortuga.getY() <= limiteInferior){

tortuga.setX(tortuga.getX()-5);

return true;

}

}

return false; // La tortuga no está sobre ninguna isla

}

// Método para verificar si un tortuga está sobre alguna isla

private boolean tortugaSobreIsla(Tortugas tortuga) {

for (Isla isla : islas) {

if (isla != null) {

// Verifica si el gnomo está dentro de los límites de la isla

boolean tocandoX = tortuga.getX() >= isla.getX() - isla.getAncho() / 2 &&

tortuga.getX() <= isla.getX() + isla.getAncho() / 2;

boolean tocandoY = tortuga.getY()>180 && tortuga.getY() >= isla.getY() - isla.getAlto() / 2 &&

tortuga.getY() <= isla.getY() + isla.getAlto() / 2;

if (tocandoX && tocandoY) {

return true; // El gnomo está sobre esta isla

}

}

}

return false; // El gnomo no está sobre ninguna isla

}

private int generarTiempoAleatorio() {

return random.nextInt(400);// Genera tiempo aleatorio hasta 400 tick

}

public void actualizar() {

tickContador++; // Incrementa el contador de ticks

// Verifica si es el momento de aparecer un nuevo Gnomo

if (tickContador >= tiempoParaAparecer) {

// Intenta agregar un nuevo Gnomo si hay espacio

agregarGnomo();

agregarTortuga();

// Resetea el contador y genera el nuevo tiempo para la próxima aparición

tickContador = 0;

tiempoParaAparecer = generarTiempoAleatorio();

}

}

private void agregarTortuga() {

//direccion = 1;

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

if(tortugas[i]==null){

int x= random.nextInt(80,720);

int y=0;

int ancho = 5;

int alto = 10;

Image imagenTortuga = Herramientas.*cargarImagen*("Imagenes/Tortuga.png");

tortugas[i] = new Tortugas(imagenTortuga, x, y, direccion,ancho, alto); // Crea el nuevo Gnomo en la posición fija

break;// Salimos del bucle después de agregar la tortuga

}

}

}

private void agregarGnomo() {

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

if (gnomos[i] == null) {

// Usar coordenadas fijas para la posición del Gnomo

int x = entorno.ancho()/2; // Coordenada fija en X

int y = 95; // Coordenada fija en Y

int ancho = 5;

int alto = 10;

int newDireccion = direccionAleatorio(direccion);

direccion=newDireccion;

Image imagenGnomo = Herramientas.*cargarImagen*("Imagenes/gnomo.png");

gnomos[i] = new Gnomo(imagenGnomo, x, y, direccion, ancho, alto); // Crea el nuevo Gnomo en la posición fija

break; // Salimos del bucle después de agregar el Gnomo

}

}

}

private int direccionAleatorio(int numero) {

Random random = new Random();

// Genera un booleano aleatorio para determinar el signo

boolean esNegativo = random.nextBoolean(); // true o false aleatoriamente

// Si esNegativo es true, devuelve el número negativo, de lo contrario el número positivo

return esNegativo ? -numero : numero;

}

// Este metodo carga el fondo del juego

public void cargarFondo(){

//Cargamos el fondo

try {

this.fondo = Herramientas.*cargarImagen*("Imagenes/fondo.jpg");

} catch (Exception e) {

// **TODO**: handle exception

System.***out***.println("Error al cargar la imagen de respaldo: " + e.getMessage());

}

}

// Este metodo se encarga de cargar las cordenadas de las Islas en forma de piramide

public void cargarIslas() {

int inicioY = 120; // Posicion inicial para la primer isla en el eje Y

int nivel = 5; // Número de niveles de islas (filas)

int columna = 1; // Columnas iniciales en cada nivel

int fila=80; // Espacio vertical entre niveles

int distancia = 160; // Espacio horizontal entre islas

int indice=0; // Índice para recorrer el array de islas

int ancho=120;

int alto=40;

Image imagenIsla = Herramientas.*cargarImagen*("Imagenes/isla.jpg"); // Cargamos la Imagen

for(int nivelActual=0; nivelActual<nivel; nivelActual++) {

int inicioX = entorno.ancho()/2; // Centramos en el eje X

for(int columnaActual=0; columnaActual<columna; columnaActual++) {

// Crear una isla y agregarla al array

this.islas[indice] = new Isla(imagenIsla, inicioX-(80\*nivelActual), inicioY, ancho, alto);

inicioX+=distancia; // Espacio entre islas

indice++; // Avanzar al siguiente índice en el array de islas

}

inicioY+=fila; // Bajar al siguiente nivel

columna++; // Aumentar la cantidad de columnas en el siguiente nivel

}

}

public void cargarCasita()

{

this.casita = Herramientas.*cargarImagen*("Imagenes/casita.png");

}

*@SuppressWarnings*("unused")

public static void main(String[] args)

{

Juego juego = new Juego();

}

}

Clase Tortugas

package juego;

import java.awt.Image;

import entorno.Entorno;

public class Tortugas{

private int direccion;

private double y , x ;

int ancho , alto;

private Image image;

//Constructor que recibe la imagen y las coordenadas

public Tortugas(Image image, double x, int y, int direccion, int ancho, int alto)

{

this.image = image;

this.x = x;

this.y = y;

this.direccion = direccion;

this.ancho = ancho;

this.alto = alto;

}

public void dibujarTortugas(Entorno entorno) {

entorno.dibujarImagen(image, x, y, 0, 0.05);

}

// metodos

public double getX() {

return x;

}

public double getY() {

return y;

}

public Image getImage() {

return image;

}

public void iniciar(int velocidad) {

this.direccion = velocidad;

}

public void mover() {

this.x = this.x + this.direccion;

}

public void dibujar(Entorno entorno) {

entorno.dibujarImagen(getImage(),getX(), getY(), 0, 0.03);

}

public void setY(double d) {

this.y = d;

}

public void setX(double d) {

this.x = d;

}

public void setDireccion(int direccion) {

this.direccion = direccion;

}

public int getDireccion() {

return direccion;

}

public int getAlto() {

return alto;

}

public int getAncho() {

return ancho;

}

}

Clase Pep

package juego;

import java.awt.Image;

import entorno.Entorno;

public class Pep {

private Image imagen;

private int x, y;

private int direccion; // 1 = derecha, -1 = izquierda

private boolean saltando;

private int velocidadY;

private int direccionUltima;

public Pep(Image imagen, int x, int y) {

this.imagen = imagen;

this.x = x;

this.y = y;

this.direccion = 1;

this.saltando = false;

this.velocidadY = 0;

}

public void dibujar(Entorno entorno) {

entorno.dibujarImagen(imagen, x, y, 0, 0.05);

}

public void moverIzquierda() {

this.x -= 2;

this.direccion = -1;

this.direccionUltima = -1;

}

public void moverDerecha() {

this.x += 2;

this.direccion = 1;

this.direccionUltima = 1;

}

public void saltar() {

if (!saltando) {

this.velocidadY = -10;

this.saltando = true;

}

}

public void actualizar() {

if (saltando) {

this.y += velocidadY;

this.velocidadY += 1; // Gravedad

if (this.y > 300) { // Simulamos que Pep está de nuevo sobre una isla

this.y = 300;

this.saltando = false;

this.velocidadY = 0;

}

}

}

public void disparar() {

// Implementación para disparar una bola de fuego

System.***out***.println("Pep dispara una bola de fuego en dirección " + direccionUltima);

}

public int getX() {

return x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

public boolean estaSaltando() {

return saltando;

}

}

Clase Isla

package juego;

import java.awt.Image;

public class Isla {

private int x, y, ancho, alto; // Posiciones X e Y de la isla

private Image image; // Imagen de la isla

// Constructor que recibe la imagen y las coordenadas

public Isla(Image image, int x, int y, int ancho, int alto ) {

this.image = image;

this.x = x;

this.y = y;

this.ancho = ancho;

this.alto = alto;

}

// metodos para acceder a las propiedades

public int getX() {

return x;

}

public int getY() {

return y;

}

public Image getImagen() {

return image;

}

public int getAncho() {

return ancho;

}

public int getAlto() {

return alto;

}

}

Clase Gnomo

package juego;

import entorno.Entorno;

import java.awt.Image;

public class Gnomo{

private int x, direccion, ancho, alto;

private double y;

private Image image;

//Constructor que recibe la imagen y las coordenadas

public Gnomo(Image image, int x, int y, int direccion, int ancho, int alto)

{

this.image = image;

this.x = x;

this.y = y;

this.direccion = direccion;

this.ancho = ancho;

this.alto = alto;

}

// metodos

public double getX() {

return x;

}

public double getY() {

return y;

}

public Image getImage() {

return image;

}

public void iniciar(int velocidad) {

this.direccion = velocidad;

}

public void mover() {

this.x = this.x + this.direccion;

}

public void dibujar(Entorno entorno) {

entorno.dibujarImagen(getImage(),getX(), getY(), 0, 0.03);

}

public void setY(double d) {

this.y = d;

}

public void setX(int x) {

this.x = x;

}

public void setDireccion(int direccion) {

this.direccion = direccion;

}

public int getAlto() {

return alto;

}

public int getAncho() {

return ancho;

}

}