:

Grupo 12

Kevin Huayta, Email: contactohuayta@gmail.com   
Gastón Rampazzo, Email: gastonrampazzo@gmail.com

Trabajo Práctico – Attack on Titan, Final Season

Programación I – Comisión 05

**Índice**

**Introducción 2**

**Descripción 3**

**Implementación 9**

**Conclusiones 41**

**Introducción**

El siguiente trabajo práctico consistió en el desarrollo de un videojuego denominado “Attack on Titan, Final Season” en el que la protagonista, llamada Mikasa, debe derrotar a la mayor cantidad de Kyojines posibles en el lapso de un minuto.

Los Kyojines son titanes de tamaño colosal que derrotan a Mikasa con solo alcanzarla, por lo que Mikasa deberá escapar de ellos al mismo tiempo que intenta destruirlos para lo cual cuenta con dos armas principales, un arma que lanza proyectiles a distancia y un suero que aparecerá cada cierto tiempo que transforma a Mikasa en Kyojina por un tiempo hasta que mate a un Kyojin.

El juego finaliza cuando Mikasa muere a manos de un Kyojin o cuando el tiempo llega a 0 (cero), y se determina el rendimiento del jugador en base a un sistema de puntos que se obtienen matando Kyojines.

Para el desarrollo del videojuego se utilizaron clases, objetos y métodos escritos en lenguaje java, y se conto con un entorno predefinido en el que se obtuvo las herramientas para dibujar en pantalla dichos objetos.

**Descripción**

Las siguientes clases conforman el videojuego:

* Juego
* Mikasa
* Titan
* Obstaculos
* Obstaculos2
* Suero
* Disparo
* Tiempo

**Clase Juego**

**Variables de instancia:**

private Entorno entorno;

private Mikasa mikasa;

private Suero suero;

private Tiempo tiempo;

private Image imageFondo;

private Lista<Disparo> disparos;

private Obstaculos obstaculos;

private boolean item;

private Lista<Titan> titanes;

private int tickUltimoTitan;

private int puntos;

private int eliminados;

private Clip sonido;

private Clip sonido2;

**Métodos:**

Juego (): Constructor de la clase que instancia los objetos necesarios para el correcto funcionamiento del juego.

Tick (): Actualiza el estado del personaje, los dinos, sus disparos y láser. Además, chequea colisiones y en el caso de que se deba eliminar un objeto, lo hace.

titanChocaObs (Titan ttn, Obstaculos2 obs): Recibe un objeto Titan y uno Obstaculos2, compara sus posiciones x e y. Si existe colisión el Kyojin choca con el obstáculo.

mikasaChocaObs (Obstaculos2 obs): Igual que con el Kyojin compara posiciones x e y de Mikasa y con cada obstáculo. Si existe colisión Mikasa choca con el obstáculo.

mikasaChocaTtn (Nodo <T> ttn): Verifica si Mikasa colisiona con un Kyojin en estado de Kyojina, en ese caso elimina dicho Kyojin.

crearSuero (): Crea un objeto suero con posición aleatoria cada un tiempo determinado.

funcionSuero (): Verifica la colision entre Mikasa y el suero. Si existe colisión Mikasa cambiara de estado a Kyojina.

procesarDisparos (): Procesa el movimiento de cada disparo, los dibuja en el entorno, y los elimina en caso de que impacten con otro objeto.

crearTitan (): Agrega un objeto Titan a la Lista titanes con coordenadas aleatorias.

procesarTitan (): Procesa el movimiento de cada titan, los dibuja en el entorno, y verifica en caso de que impacten con otro objeto.

impactoTitan (): Recorre la lista de disparos y verifica si colisionan con cada Titan en la lista de titanes. Si existe colisión el titan se elimina y se suman puntos y Kyojines eliminados.

impactoObs (): Verifica la colision de cada disparo con cada obstaculo. Si existe colisión se elimina dicho disparo.

colisionTitanes (): Compara cada Titan en la lista de titanes con los demás titanes de la lista con un id distinto del mismo. Si existe colisión los titanes chocan.

main (): Ejecuta el juego.

**Clase Mikasa**

**Variables de instancia:**

private double x;

private double y;

private double radio;

private double angulo;

private double distancia;

boolean convertir = false;

private Lista<Disparo> disparos;

private int ultimoDisparo;

private int sentido = 0;

private Image image=Herramientas.*cargarImagen*("mikasa1.png");

private Image image2 = Herramientas.*cargarImagen*("mikasa-Titan.png";

**Métodos:**

Mikasa (): Constructor del objeto Mikasa.

dibujarSprite (): Dibuja a Mikasa en el entorno con una imagen determinada, que cambia si Mikasa se encuentra en estado de Kyojina.

disparar (): Agrega un objeto Disparo a la lista de disparos de Mikasa.

moverAdelante (): Si Mikasa se encuentra dentro de las dimensiones del entorno avanza en el eje X e Y.

moverAtras (): Si Mikasa se encuentra dentro de las dimensiones del entorno retrocede en el eje X e Y.

moverDerecha (): Si Mikasa avanza aumenta el ángulo, si retrocede disminuye el ángulo.

moverIzquierda (): Si Mikasa avanza disminuye el ángulo, si retrocede aumenta el ángulo.

distancia (): Calcula la distancia entre las coordenadas X, Y de Mikasa y otro par ordenado.

\*Todas las variables de instancia tienen sus correspondientes Getters y Setters.

**Clase Titan**

**Variables de instancia:**

double x;

double y;

double radio;

double angulo;

double distancia;

Image image = Herramientas.*cargarImagen*("titan2.png");

private boolean impacto=false;

**Métodos:**

Titan (): Constructor del objeto Titan.

dibujar (): Muestra al titan en el entorno.

moverTitan (): Avanza al titan en los ejes X, Y.

direccionTitan (): Determina el angulo del titan para que avance siempre hacia Mikasa.

distancia (): Calcula la distancia entre las coordenadas X, Y del titan y otro par ordenado.

colisiona (): Recibe dos radios y calcula si son mayores a la distancia. Retorna un valor booleano.

rebotar (): Verifica las coordenadas X, Y de dos titanes. Si existe colisión chocan entre sí.

\*Todas las variables de instancia tienen sus correspondientes Getters y Setters.

**Clase Obstaculos**

**Variables de instancia:**

public Obstaculos2[] obstaculos2;

private double distancia;

**Métodos:**

Obstaculos (): Constructor del objeto Obstaculos. Contiene una lista de objetos de tipo Obstaculos2.

generarObs (): Crea los objetos de la lista en las coordenadas determinadas.

distancia (): Calcula la distancia entre las coordenadas X, Y del obstaculo y otro par ordenado.

colisiona (): Recibe dos radios y calcula si son mayores a la distancia. Retorna un valor booleano.

\*Todas las variables de instancia tienen sus correspondientes Getters y Setters.

**Clase Obstaculos2**

**Variables de instancia:**

private double x;

private double y;

private double radio;

private Image image =Herramientas.*cargarImagen*("obs.png");

**Métodos:**

Obstaculos2 (): Constructor del objeto Obstaculos2.

dibujarSprite (): Dibuja al obstaculo en el entorno con una imagen determinada.

\*Todas las variables de instancia tienen sus correspondientes Getters y Setters.

**Clase Suero**

**Variables de instancia:**

private double x;

private double y;

private double radio;

private double distancia;

private Image image =Herramientas.cargarImagen("Suero.png");

private Image aux = this.image;

**Métodos:**

Suero (): Constructor del objeto Suero.

dibujarSprite (): Dibuja al suero en el entorno con una imagen determinada.

distancia (): Calcula la distancia entre las coordenadas X, Y del suero y otro par ordenado.

colisiona (): Recibe dos radios y calcula si son mayores a la distancia. Retorna un valor booleano.

\*Todas las variables de instancia tienen sus correspondientes Getters y Setters.

**Clase Disparo**

**Variables de instancia:**

private double x, y, radio, angulo, distancia;

private Image image =Herramientas.cargarImagen("Proyectil.png");

private boolean impacto=false;

**Métodos:**

Disparo (): Constructor del objeto Disparo.

dibujarSprite (): Dibuja al Disparo en el entorno con una imagen determinada.

mover (): Avanza el Disparo en los ejes X, Y.

distancia (): Calcula la distancia entre las coordenadas X, Y del disparo y otro par ordenado.

colisiona (): Recibe dos radios y calcula si son mayores a la distancia. Retorna un valor booleano.

estaEnPantalla (): Verifica que el disparo se encuentre dentro de las dimensiones del entorno. Devuelve un valor booleano.

\*Todas las variables de instancia tienen sus correspondientes Getters y Setters.

**Clase Tiempo**

**Variables de instancia:**

private boolean inicia;

private double contar;

private double timer;

**Métodos:**

Tiempo (): Constructor del objeto Tiempo.

\*Todas las variables de instancia tienen sus correspondientes Getters y Setters.

**Problemas y soluciones:**

Suero:

Al principio tuvimos problemas en cómo hacer que desaparezca cuando mikasa colisione con el objeto, pero después se me ocurrió medir la distancia entre el X e Y del suero y de mikasa, y luego con el método "colisiona" medir la suma de sus radios para ver que cuando la distancia sea menor a la suma de estos se produzca una colision. Otro problema fue que al momento de colocar el sprite al no haberlo hecho con nodos, si colocaba una condición para que cuando colisione se vuelva null (Suero.setImage(null)) solo aparecía una vez por obvias razones, esto lo solucionamos creando un sprite auxiliar.

Obstaculos:

Al principio se trabajó con cuadrados, pero como no podíamos lograr la colisión entre ellos decidimos trabajar con círculos implementando los métodos "distancia" y "colisiona", una vez obtenido el cuándo deben colisionar, faltaban las condiciones para que el objeto no avance, esto último se solucionó mediante la lógica ya que solo había que analizar la posición de las X e Y.

Dichos métodos nos permitieron resolver la colisión de los obstáculos con cada uno de los demás objetos (Titan, Mikasa y Disparo).

Por otra parte, hubo un problema con la posición de los obstáculos ya que cambiaban de posición cada vez que se iniciaba el juego y en algunas ocasiones se superponían, esto se solucionó dándoles una posición fija debido a que en los requerimientos obligatorios no especificaba nada sobre eso.

Titanes:

En principio se usó una lista similar a la lista de obstáculos, pero no lográbamos que choquen entre sí y como no se trabajó con nodos era imposible desaparecerlos mediante un null, así que se solucionó justamente utilizando nodos.

Para hacer esta lista y la de disparos se utilizaron dos clases (Lista y Nodo) que permitieron la construcción de listas enlazadas para tener un mayor control sobre los objetos.

**Implementación**

**Codigo Fuente:**

Clase Juego

package juego;

import java.awt.Color;

import java.awt.Image;

import java.util.Iterator;

import java.util.Random;

import entorno.Entorno;

import entorno.Herramientas;

import entorno.InterfaceJuego;

import utils.Tiempo;

import utils.Lista;

import utils.Nodo;

import javax.sound.sampled.Clip;

*@SuppressWarnings*("unused")

public class Juego extends InterfaceJuego {

// El objeto Entorno que controla el tiempo y otros

private Entorno entorno;

private Mikasa mikasa;

private Suero suero;

private Tiempo tiempo;

private Image imageFondo;

private Lista<Disparo> disparos;

private Obstaculos obstaculos;

private boolean item;

private Lista<Titan> titanes;

private int tickUltimoTitan;

private int puntos;

private int eliminados;

private Clip sonido;

private Clip sonido2;

public Juego() {

// Inicializa el objeto entorno

this.entorno = new Entorno(this, "Sakura Ikebana Delivery - Grupo N - Apellido1 - Apellido2 -Apellido3 - V0.01",

800, 600);

this.mikasa = new Mikasa(400, 300, 40, 0, false);

this.suero = new Suero(0);

this.tiempo = new Tiempo(true, 0, 60);

this.disparos = this.mikasa.getDisparos();

this.imageFondo = Herramientas.*cargarImagen*("FondoSeis.jpg");

this.obstaculos = new Obstaculos(5);

this.obstaculos.generarObs();

this.item = false;

this.tickUltimoTitan = -1;

this.titanes = new Lista<Titan>();

this.puntos=0;

this.eliminados=0;

this.sonido = Herramientas.*cargarSonido*("musicaJuego2.wav");

this.sonido2 = Herramientas.*cargarSonido*("musicaGameOver.wav");

// Inicializar lo que haga falta para el juego

// ...

// 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

// ...

if (tiempo.isInicia()) {

if ((int) tiempo.getTimer()==0) {

tiempo.setInicia(false);

}

this.sonido.start();

// dibujar

entorno.dibujarImagen(this.imageFondo, 400, 400, 0, 1.7);

mikasa.dibujarSprite(entorno);

// instrucciones

if (entorno.estaPresionada('w') || entorno.estaPresionada(entorno.TECLA\_ARRIBA))

mikasa.moverAdelante();

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

mikasa.moverIzquierda();

if (entorno.estaPresionada('s') || entorno.estaPresionada(entorno.TECLA\_ABAJO))

mikasa.moverAtras();

if (entorno.estaPresionada('d') || entorno.estaPresionada(entorno.TECLA\_DERECHA))

mikasa.moverDerecha();

// titanes

if (titanes.largo() < 6) {

crearTitan();

}

this.titanes.forEachNodo(titan->{

impactoTitan(titan);

return null;

});

procesarTitan();

// disparo

if (this.entorno.estaPresionada(this.entorno.TECLA\_ESPACIO)

&& this.mikasa.getUltimoDisparo() + 50 <= this.tiempo.getContar() && !mikasa.convertir) {

this.mikasa.disparar((int) this.tiempo.getContar());

}

procesarDisparos();

// suero

crearSuero();

funcionSuero();

// obstaculos

// por cada obstaculo chequea la distancia entre el titan y el obs, si el metodo colisiona es true,

// y la distancia entre los circulos es menor a 100, resta o suma en sus respectivos X e Y. del titan

for (Obstaculos2 obs : this.obstaculos.getObstaculos2()) {

this.titanes.forEachNodo(ttn -> {

ttn.getElemento().distancia(ttn.getElemento().getX(), ttn.getElemento().getY(), mikasa.getX(),

mikasa.getY());

obstaculos.distancia(obs.getX(), obs.getY(), ttn.getElemento().getX(), ttn.getElemento().getY());

if (obstaculos.colisiona(obs.getRadio(), ttn.getElemento().getRadio(), obstaculos.getDistancia())

&& obstaculos.getDistancia() <= 100) {

titanChocaObs(ttn.getElemento(), obs);

}

if (!mikasa.isConvertir() && ttn.getElemento().colisiona(ttn.getElemento().getRadio(),

mikasa.getRadio(), ttn.getElemento().getDistancia())

&& ttn.getElemento().getDistancia() <= 50) {

tiempo.setInicia(false);

}

if (mikasa.isConvertir() && ttn.getElemento().colisiona(ttn.getElemento().getRadio(),

mikasa.getRadio(), ttn.getElemento().getDistancia())

&& ttn.getElemento().getDistancia() <= 100) {

mikasaChocaTtn(ttn);

}

obs.dibujarSprite(entorno);

ttn.getElemento().dibujar(entorno);

return null;

});

// obs.dibujar2(entorno);

}

// lo mismo pero con mikasa

for (Obstaculos2 obs : this.obstaculos.getObstaculos2()) {

obstaculos.distancia(obs.getX(), obs.getY(), mikasa.getX(), mikasa.getY());

if (!mikasa.isConvertir()

&& obstaculos.colisiona(obs.getRadio(), mikasa.getRadio(), obstaculos.getDistancia())

&& obstaculos.getDistancia() <= 70) {

mikasaChocaObs(obs);

} else if (mikasa.isConvertir()

&& obstaculos.colisiona(obs.getRadio(), mikasa.getRadio(), obstaculos.getDistancia())

&& obstaculos.getDistancia() <= 85) {

mikasaChocaObs(obs);

}

}

for (Obstaculos2 obs : this.obstaculos.getObstaculos2()) {

impactoObs(obs);

}

// controla el fin del juego

// if ((int) tiempo.getContar() > 40) {

// this.tiempo = new Tiempo(false, 0);

// }

// tiempo.setContar(0.015);

tiempo.setContar(1);

tiempo.setTimer(0.01111102);

// diujar datos

entorno.cambiarFont("Arial Black", 25, Color.***BLACK***);

entorno.escribirTexto("KYOJINES ELIMINADOS: " + 0, 4, 588);

entorno.cambiarFont("Arial Black", 25, Color.***WHITE***);

entorno.escribirTexto("TIEMPO: " + (int) tiempo.getTimer(), 620, 23);

entorno.escribirTexto("PUNTAJE: " + this.puntos, 4, 23);

} else {

entorno.cambiarFont("Arial", 32, Color.***WHITE***);

entorno.escribirTexto("GAME OVER", 300, 300);

entorno.cambiarFont("Arial", 25, Color.***WHITE***);

entorno.escribirTexto("PUNTAJE TOTAL: " + this.puntos, 270, 330);

this.sonido.close();

this.sonido2.start();

}

}

// metodos

public void titanChocaObs(Titan ttn, Obstaculos2 obs) {

if (ttn.getX() <= obs.getX() && ttn.getY() <= obs.getY()) {

ttn.setX(ttn.getX() - 2);

ttn.setY(ttn.getY() - 2);

} else if (ttn.getX() >= obs.getX() && ttn.getY() >= obs.getY()) {

ttn.setX(ttn.getX() + 2);

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

} else if (ttn.getX() >= obs.getX() && ttn.getY() <= obs.getY()) {

ttn.setX(ttn.getX() + 2);

ttn.setY(ttn.getY() - 2);

} else if (ttn.getX() <= obs.getX() && ttn.getY() >= obs.getY()) {

ttn.setX(ttn.getX() - 2);

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

}

}

public <T> void mikasaChocaTtn(Nodo<T> ttn) {

this.titanes.forEachNodo(nodoTitan -> {

if (ttn.getId() == nodoTitan.getId() && mikasa.isConvertir()) {

this.titanes.quitarPorId(nodoTitan.getId());

mikasa.setConvertir(false);

mikasa.setRadio(40);

this.puntos += 50;

this.eliminados += 1;

}

return null;

});

}

public void mikasaChocaObs(Obstaculos2 obs) {

if (mikasa.getX() <= obs.getX() && mikasa.getY() <= obs.getY()) {

mikasa.setX(mikasa.getX() - 2);

mikasa.setY(mikasa.getY() - 2);

} else if (mikasa.getX() >= obs.getX() && mikasa.getY() >= obs.getY()) {

mikasa.setX(mikasa.getX() + 2);

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

} else if (mikasa.getX() >= obs.getX() && mikasa.getY() <= obs.getY()) {

mikasa.setX(mikasa.getX() + 2);

mikasa.setY(mikasa.getY() - 2);

} else if (mikasa.getX() <= obs.getX() && mikasa.getY() >= obs.getY()) {

mikasa.setX(mikasa.getX() - 2);

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

}

}

public void crearSuero() {

for (Obstaculos2 obs : this.obstaculos.getObstaculos2()) {

if (tiempo.getContar() == 600/\* 0.015 \*/) {

suero.setX(Math.*random*() \* (50 - 750) + 750);

suero.setY(Math.*random*() \* (50 - 550) + 550);

suero.setRadio(40);

// suero.distancia2(suero.getX(), suero.getY(), obs.getX(), obs.getY());

// if (suero.colisiona(obs.getRadio(), suero.getRadio(), suero.getDistancia2())) {

if (suero.getX() <= obs.getX() && suero.getY() <= obs.getY()) {

suero.setX(suero.getX() + 40);

suero.setY(suero.getY() + 40);

} else if (suero.getX() >= obs.getX() && suero.getY() >= obs.getY()) {

suero.setX(suero.getX() + 40);

suero.setY(suero.getY() + 40);

} else if (suero.getX() >= obs.getX() && suero.getY() <= obs.getY()) {

suero.setX(suero.getX() + 40);

suero.setY(suero.getY() - 40);

} else if (suero.getX() <= obs.getX() && suero.getY() >= obs.getY()) {

suero.setX(suero.getX() - 40);

suero.setY(suero.getY() + 40);

}

}

if (tiempo.getContar() == 2000) {

this.item = false;

suero.setX(Math.*random*() \* (50 - 750) + 750);

suero.setY(Math.*random*() \* (50 - 550) + 550);

suero.setRadio(40);

if (suero.getX() <= obs.getX() && suero.getY() <= obs.getY()) {

suero.setX(suero.getX() + 40);

suero.setY(suero.getY() + 40);

} else if (suero.getX() >= obs.getX() && suero.getY() >= obs.getY()) {

suero.setX(suero.getX() + 40);

suero.setY(suero.getY() + 40);

} else if (suero.getX() >= obs.getX() && suero.getY() <= obs.getY()) {

suero.setX(suero.getX() + 40);

suero.setY(suero.getY() - 40);

} else if (suero.getX() <= obs.getX() && suero.getY() >= obs.getY()) {

suero.setX(suero.getX() - 40);

suero.setY(suero.getY() + 40);

}

}

}

}

public void funcionSuero() {

if ((tiempo.getContar() > 600 && tiempo.getContar() < 1500) && !item) {

suero.distancia(suero.getX(), suero.getY(), mikasa.getX(), mikasa.getY());

if (suero.getDistancia() <= 60

&& suero.colisiona(mikasa.getRadio(), suero.getRadio(), suero.getDistancia())) {

this.item = true;

mikasa.setConvertir(true);

mikasa.setRadio(105);

mikasa.dibujarSprite(entorno);

suero.setImage(null);

} else {

suero.dibujarSprite(entorno);

}

} else if ((tiempo.getContar() > 2000 && tiempo.getContar() < 4500) && !item) {

suero.distancia(suero.getX(), suero.getY(), mikasa.getX(), mikasa.getY());

if (suero.getDistancia() <= 60

&& suero.colisiona(mikasa.getRadio(), suero.getRadio(), suero.getDistancia())) {

this.item = true;

suero.setAux(null);

mikasa.setConvertir(true);

mikasa.setRadio(105);

mikasa.dibujarSprite(entorno);

} else {

suero.dibujarSprite(entorno);

}

}

}

private void procesarDisparos() {

this.mikasa.getDisparos().forEachNodo(disparo -> {

if (disparo.getElemento().getImpacto()==false) {

disparo.getElemento().dibujarSprite(this.entorno);

disparo.getElemento().mover();

}

if (disparo.getElemento().getImpacto()==true || !disparo.getElemento().estaEnPantalla()) {

this.disparos.quitarPorId(disparo.getId());

}

return null;

});

}

private void crearTitan() {

if (this.tickUltimoTitan == -1 || this.tickUltimoTitan + 50 < tiempo.getContar()) {

this.titanes.agregarAtras(new Titan(Math.*random*() \* (100 - 700) + 700, Math.*random*() \* (50 - 500) + 500, 120, 0));

this.tickUltimoTitan = (int) tiempo.getContar();

}

}

private void procesarTitan() {

this.titanes.forEachNodo(nodoTitan -> {

colisionTitanes(nodoTitan);

if (nodoTitan.getElemento().getImpacto()==false) {

nodoTitan.getElemento().direccionTitan(this.mikasa.getX(), this.mikasa.getY());

nodoTitan.getElemento().dibujar(this.entorno);

nodoTitan.getElemento().moverTitan();}

if (nodoTitan.getElemento().getImpacto()==true) {

this.titanes.quitarPorId(nodoTitan.getId());

}

return null;

});

}

private void impactoTitan(Nodo<Titan> titan) {

this.disparos.forEachNodo(disparo -> {

Disparo dis = disparo.getElemento();

dis.distancia(dis.getX(),dis.getY(),titan.getElemento().getX(), titan.getElemento().getY());

if (titan.getElemento().colisiona(titan.getElemento().getRadio(),dis.getRadio(), dis.getDistancia()) && dis.getDistancia() < 60) {

dis.setImpacto(true);

titan.getElemento().setImpacto(true);

this.puntos+=15;

this.eliminados+=1;

}

return null;

});

}

private void impactoObs(Obstaculos2 obs) {

this.disparos.forEachNodo(disparo -> {

Disparo dis = disparo.getElemento();

dis.distancia(dis.getX(),dis.getY(),obs.getX(), obs.getY());

if (dis.colisiona(obs.getRadio(),dis.getRadio(), dis.getDistancia()) && dis.getDistancia() < 50) {

dis.setImpacto(true);

}

return null;

});

}

private void colisionTitanes(Nodo<Titan> t2) {

this.titanes.forEachNodo(titan ->{

Titan t=titan.getElemento();

Titan otro=t2.getElemento();

if (titan.getId()!=t2.getId()) {

t.distancia(t.getX(),t.getY(),otro.getX(), otro.getY());

if (t.colisiona(t.getRadio(),otro.getRadio(), t.getDistancia()) && t.getDistancia()<60){

t.rebotar(t, otro);

}

}

return null;

});

}

public static void main(String[] args) {

Juego juego = new Juego();

}

}

Clase Mikasa:

package juego;

import java.awt.Image;

import entorno.Entorno;

import entorno.Herramientas;

import utils.Lista;

public class Mikasa {

private double x;

private double y;

private double radio;

private double angulo;

private double distancia;

boolean convertir = false;

private Lista<Disparo> disparos;

private int ultimoDisparo;

private int sentido = 0;

private Image image = Herramientas.cargarImagen("mikasa1.png");

private Image image2 = Herramientas.cargarImagen("mikasaTitan.png");

public Mikasa(double x, double y, double radio, double angulo, boolean convertir) {

this.x = x;

this.y = y;

this.radio = radio;

this.angulo = angulo;

this.convertir = convertir;

this.disparos = new Lista<Disparo>();

} }

public void dibujarSprite(Entorno e) {

if (this.convertir) {

e.dibujarImagen(this.image2, x + 5, y, angulo, 0.1);

} else {

e.dibujarImagen(image, this.x, this.y, this.angulo, 0.1);

}

}

public void disparar(int contadorTicks) {

this.disparos.agregarAtras(new Disparo(this.x, this.y, 10, this.angulo));

this.ultimoDisparo = contadorTicks;

}

// movimiento

public void moverAdelante() {

this.sentido = 0;

if (this.x > 20 && ((this.x) < 820) && this.y < 590 && ((this.y) > -30)) {

this.x += Math.cos(this.angulo) \* 2;

this.y += Math.sin(this.angulo) \* 2;

}

if (this.x <= 20 && !this.convertir) {

this.x += 0.5;

} else if (this.x <= 35 && this.convertir) {

this.x += 5;

}

if ((this.x) >= 785 && !this.convertir) {

this.x -= 5;

} else if (this.x >= 755 && this.convertir) {

this.x -= 5;

}

if (this.y >= 580 && !this.convertir) {

this.y -= 5;

} else if (this.y >= 575 && this.convertir) {

this.y -= 5;

}

if ((this.y) <= 20 && !this.convertir) {

this.y += 5;

} else if (this.y <= 35 && this.convertir) {

this.y += 5;

}

}

public void moverDerecha() {

if (this.sentido == 0) {

this.angulo += 0.08;

} else {

this.angulo -= 0.08;

}

}

public void moverIzquierda() {

if (this.sentido == 0) {

this.angulo -= 0.08;

} else {

this.angulo += 0.08;

}

}

public void moverAtras() {

this.sentido = 1;

if (this.x > 15 && ((this.x) < 820) && this.y < 590 && ((this.y) > -30)) {

this.x -= Math.cos(this.angulo) \* 2;

this.y -= Math.sin(this.angulo) \* 2;

}

if (this.x <= 15 && !this.convertir) {

this.x += 5;

} else if (this.x <= 30 && this.convertir) {

this.x += 5;

}

if ((this.x) >= 785 && !this.convertir) {

this.x -= 5;

} else if (this.x >= 755 && this.convertir) {

this.x -= 5;

}

if (this.y >= 580 && !this.convertir) {

this.y -= 5;

} else if (this.y >= 575 && this.convertir) {

this.y -= 5;

}

if ((this.y) <= 20 && !this.convertir) {

this.y += 5;

} else if (this.y <= 35 && this.convertir) {

this.y += 5;

}

}

public void distancia(double x1, double y1, double x2, double y2) {

this.distancia = Math.sqrt((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1));

}

// getters and setters

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 double getRadio() {

return radio;

}

public void setRadio(double radio) {

this.radio = radio;

}

public double getAngulo() {

return angulo;

}

public void setAngulo(double angulo) {

this.angulo = angulo;

}

public double getDistancia() {

return distancia;

}

public void setDistancia(double distancia) {

this.distancia = distancia;

}

public Image getImage() {

return image;

}

public void setImage(Image image) {

this.image = image;

}

public boolean isConvertir() {

return convertir;

}

public void setConvertir(boolean convertir) {

this.convertir = convertir;

}

public Image getImage2() {

return image2;

}

public void setImage2(Image image2) {

this.image2 = image2;

}

public int getUltimoDisparo() {

return this.ultimoDisparo;

}

public Lista<Disparo> getDisparos() {

return this.disparos;

}

public void setDisparos(Lista<Disparo> disparos) {

this.disparos = disparos;

}}

Clase Disparo:

package juego;

import java.awt.Color;

import java.awt.Image;

import entorno.Entorno;

import entorno.Herramientas;

public class Disparo {

private double x, y, radio, angulo,distancia;

private Image image = Herramientas.cargarImagen("Proyectil.png");

private boolean impacto=false;

public Disparo(double x, double y, double radio ,double angulo) {

this.x = x;

this.y = y;

this.radio= radio;

this.angulo = angulo;

}

public void dibujarSprite(Entorno e) {

e.dibujarImagen(image, x, y-1, angulo,0.1);

}

public void mover() {

this.x += Math.cos(this.angulo) \* 4;

this.y += Math.sin(this.angulo) \* 4;

}

public boolean colisiona(double radio1, double radio2, double dist) {

return (radio1 + radio2) > dist;

}

public void distancia(double x1, double y1, double x2, double y2) {

this.distancia = Math.sqrt((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1));

}

public boolean estaEnPantalla() {

return (this.x<800 && this.x>0 && this.y<600 && this.y>0);

}

// getters and setters

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 double getRadio() {

return radio;

}

public void setRadio(double radio) {

this.radio = radio;

}

public double getAngulo() {

return angulo;

}

public void setAngulo(double angulo) {

this.angulo = angulo;

}

public double getDistancia() {

return distancia;

}

public boolean getImpacto() {

return impacto;

}

public void setImpacto(boolean a) {

this.impacto=a;

}

}

Clase Obstaculos:

package juego;

public class Obstaculos {

public Obstaculos2[] obstaculos2;

private double distancia;

public Obstaculos(int tamanio) {

this.obstaculos2 = new Obstaculos2[tamanio];

}

public void generarObs() {

if (this.obstaculos2[0] == null) {

this.obstaculos2[0] = new Obstaculos2(50, 500, 70);

}

if (this.obstaculos2[1] == null) {

this.obstaculos2[1] = new Obstaculos2(100, 95, 70);

}

if (this.obstaculos2[2] == null) {

this.obstaculos2[2] = new Obstaculos2(360, 507, 70);

}

if (this.obstaculos2[3] == null) {

this.obstaculos2[3] = new Obstaculos2(700, 350, 70);

}

if(this.obstaculos2[4] == null){

this.obstaculos2[4] = new Obstaculos2(440, 200, 70);

}

}

public boolean colisiona(double radio1, double radio2, double dist) {

return (radio1 + radio2) > dist;

}

public void distancia(double x1, double y1, double x2, double y2) {

this.distancia = Math.sqrt((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1));

}

// getters and setters

public Obstaculos2[] getObstaculos2() {

return obstaculos2;

}

public void setObstaculos2(Obstaculos2[] obstaculos2) {

this.obstaculos2 = obstaculos2;

}

public double getDistancia() {

return distancia;

}

public void setDistancia(double distancia) {

this.distancia = distancia;

}

}

Clase Obstaculos2:

package juego;

import java.awt.Color;

import java.awt.Image;

import entorno.Entorno;

import entorno.Herramientas;

public class Obstaculos2 {

private double x;

private double y;

private double radio;

private Image image = Herramientas.cargarImagen("obs.png");

public Obstaculos2(double x, double y, double radio) {

this.x = x;

this.y = y;

this.radio = radio;

}

public void dibujarSprite(Entorno e) {

e.dibujarImagen(image, x + 30, y + 5, 0, 1.2/\* 1.2 \*/);

}

// getters and setters

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 double getRadio() {

return radio;

}

public void setRadio(double radio) {

this.radio = radio;

}

public Image getImage() {

return image;

}

public void setImage(Image image) {

this.image = image;

}

}

Clase Titan:

package juego;

import java.awt.Color;

import java.awt.Image;

import entorno.Entorno;

import entorno.Herramientas;

public class Titan {

double x;

double y;

double radio;

double angulo;

double distancia;

Image image = Herramientas.*cargarImagen*("titan2.png");

private boolean impacto=false;

public Titan(double x, double y, double radio, double angulo) {

this.x = x;

this.y = y;

this.radio = radio;

this.angulo = angulo;

}

public void dibujar(Entorno e) {

e.dibujarImagen(image, x, y, 0, 0.15);

}

public void moverTitan() {

this.x -= Math.*cos*(this.angulo) \* 0.3;

this.y -= Math.*sin*(this.angulo) \* 0.3;

}

public void rebotar(Titan t1, Titan t2) {

if (t1.getX() <= t2.getX() && t1.getY() <= t2.getY()) {

t1.setX(t1.getX() - 2);

t1.setY(t1.getY() - 2);

} else if (t1.getX() >= t2.getX() && t1.getY() >= t2.getY()) {

t1.setX(t1.getX() + 2);

t1.setY(t1.getY() + 2);

} else if (t1.getX() >= t2.getX() && t1.getY() <= t2.getY()) {

t1.setX(t1.getX() + 2);

t1.setY(t1.getY() - 2);

} else if (t1.getX() <= t2.getX() && t1.getY() >= t2.getY()) {

t1.setX(t1.getX() - 2);

t1.setY(t1.getY() + 2);

}

}

public void direccionTitan(double x, double y) {

double dx = this.x - x;

double dy = this.y - y;

this.angulo = Math.*atan2*(dy, dx);

}

public boolean colisiona(double radio1, double radio2, double dist) {

return (radio1 + radio2) > dist;

}

public void distancia(double x1, double y1, double x2, double y2) {

this.distancia = Math.*sqrt*((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1));

}

// getters and setters

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 double getRadio() {

return radio;

}

public void setRadio(double radio) {

this.radio = radio;

}

public double getAngulo() {

return angulo;

}

public void setAngulo(double angulo) {

this.angulo = angulo;

}

public Image getImage() {

return image;

}

public void setImage(Image image) {

this.image = image;

}

public double getDistancia() {

return this.distancia;

}

public boolean getImpacto() {

return impacto;

}

public void setImpacto(boolean a) {

this.impacto=a;

}

}

Clase Suero:

package juego;

import java.awt.Image;

import entorno.Entorno;

import entorno.Herramientas;

public class Suero {

private double x;

private double y;

private double radio;

private double distancia;

private Image image = Herramientas.cargarImagen("Suero.png");

private Image aux = this.image;

public Suero(double radio) {

this.radio = radio;

}

// public void dibujar(Entorno e) {

// e.dibujarCirculo(this.x, this.y, this.radio, Color.CYAN);

// }

public void dibujarSprite(Entorno e) {

if (this.image != null) {

e.dibujarImagen(image, x, y, 0, 0.1);

} else if (this.aux != null) {

e.dibujarImagen(aux, x, y, 0, 0.1);

} else {

return;

}

}

public boolean colisiona(double radio1, double radio2, double dist) {

return (radio1 + radio2) > dist;

}

public void distancia(double x1, double y1, double x2, double y2) {

this.distancia = Math.sqrt((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1));

}

// getters and setters

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 double getRadio() {

return radio;

}

public void setRadio(double radio) {

this.radio = radio;

}

public double getDistancia() {

return distancia;

}

public void setDistancia(double distancia) {

this.distancia = distancia;

}

public Image getImage() {

return image;

}

public void setImage(Image image) {

this.image = image;

}

public Image getAux() {

return aux;

}

public void setAux(Image aux) {

this.aux = aux;

}

}

Clase Tiempo:

package utils;

public class Tiempo {

private boolean inicia;

private double contar;

private double timer;

public Tiempo(boolean inicia, double cont, double timer) {

this.inicia = inicia;

this.contar = cont;

this.timer = timer;

}

// getters and setters

public double getContar() {

return contar;

}

public boolean isInicia() {

return inicia;

}

public void setInicia(boolean inicia) {

this.inicia = inicia;

}

public void setContar(double contar) {

this.contar += contar;

}

public double getTimer() {

return timer;

}

public void setTimer(double timer) {

this.timer -= timer;

}

}

Clase Lista:

package utils;

import java.util.function.Function;

public class Lista<T> {

Nodo<T> primero;

private int proximoId;

public Lista() {

this.primero = null;

this.proximoId = 0;

}

public void mostrar() {

Nodo<T> actual = this.primero;

System.out.print("[");

while (actual != null) {

System.out.print(actual.id + " ");

actual = actual.siguiente;

}

System.out.println("]");

}

public void agregarAtras(T n) {

Nodo<T> nuevo = new Nodo<T>(n, proximoId);

this.proximoId += 1;

if (this.primero == null) {

this.primero = nuevo;

} else {

Nodo<T> actual = this.primero;

while (actual.siguiente != null) {

actual = actual.siguiente;

}

actual.siguiente = nuevo;

}

}

public void agregarAdelante(T n) {

Nodo<T> nuevo = new Nodo<T>(n, proximoId);

this.proximoId += 1;

if (this.primero == null) {

this.primero = nuevo;

} else {

nuevo.siguiente = this.primero;

this.primero = nuevo;

}

}

public int largo() {

Nodo<T> actual = this.primero;

int cant = 0;

while (actual != null) {

cant++;

actual = actual.siguiente;

}

return cant;

}

public void quitarDePosicion(int pos) {

Nodo<T> actual = this.primero;

if (this.primero != null && pos == 0) {

this.primero = this.primero.siguiente;

}

if (pos >= this.largo())

return;

int i = 1;

while (actual != null && actual.siguiente != null) {

if (i == pos) {

actual.siguiente = actual.siguiente.siguiente;

return;

}

actual = actual.siguiente;

i++;

}

}

public void quitarPorId(int id) {

Nodo<T> actual = this.primero;

if (this.primero != null && this.primero.id == id) {

this.primero = this.primero.siguiente;

}

while (actual != null && actual.siguiente != null) {

if (actual.siguiente.id == id) {

actual.siguiente = actual.siguiente.siguiente;

return;

}

actual = actual.siguiente;

}

}

public T obtenerDePosicion(int pos) {

Nodo<T> actual = this.primero;

int i = 0;

while (actual != null && actual.siguiente != null) {

if (i == pos) {

return actual.elemento;

}

actual = actual.siguiente;

i++;

}

return null;

}

public T obtenerPorId(int id) {

Nodo<T> actual = this.primero;

while (actual != null && actual.siguiente != null) {

if (actual.id == id) {

return actual.elemento;

}

actual = actual.siguiente;

}

return null;

}

public void forEachNodo(Function<Nodo<T>, Void> callback) {

Nodo<T> actual = this.primero;

while (actual != null) {

callback.apply(actual);

actual = actual.siguiente;

}

}

public void forEachElement(Function<T, Void> callback) {

Nodo<T> actual = this.primero;

while (actual != null) {

callback.apply(actual.elemento);

actual = actual.siguiente;

}

}

}

Clase Nodo:

package utils;

public class Nodo<T> {

T elemento;

Nodo<T> siguiente;

int id;

public Nodo() {

}

public Nodo(T elemento, int id) {

this.elemento = elemento;

this.siguiente = null;

this.id = id;

}

public T getElemento() {

return this.elemento;

}

public int getId() {

return this.id;

}

}

**Conclusiones**

Este trabajo práctico nos presentó un desafío en el que pudimos poner en práctica los conocimientos adquiridos a lo largo del cuatrimestre. Se trabajó en conjunto para solucionar los inconvenientes que fueron surgiendo y se pudo trabajar en conjunto con los profesores para resolver las dudas que se nos presentaron. Para poder trabajar en el proyecto actualizado a su ultima version, los integrantes del grupo contamos con un repositorio en github.com en el que se subió el proyecto, además de un grupo de WhatsApp en el que se estuvo en constante comunicación para llevar a cabo el proyecto.