:

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……………………………………………….. 8**

**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;

**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.

**Implementación**