Víctor López Lanuza Adrián García Campillo

Ejercicio 2

En la creación de los códigos "Leader2.java" y "Droids.java" para Robocode, nos hemos basado en los conocimientos adquiridos durante el ejercicio 1. En ese ejercicio, exploramos la programación de robots individuales capaces de participar en batallas en el simulador de Robocode. Estos robots individuales se enfrentaban a desafíos autónomos y demostraban sus habilidades de movimiento y disparo.

Estrategias Consideradas y Utilizadas

Líder Coordinador: El líder es el núcleo del equipo, encargado de tomar decisiones estratégicas basadas en la información disponible y coordinar las acciones de los droides. Escanea enemigos, decide cuándo es apropiado disparar y transmite información importante a los droides.

Droides Ejecutores: Los droides son unidades de combate que siguen las órdenes del líder. Están programados para moverse de manera estratégica en el campo de batalla, esquivar disparos y, bajo las indicaciones del líder, realizar disparos precisos contra los enemigos.

Esta estrategia busca aprovechar la comunicación entre los miembros del equipo para una mayor efectividad en la batalla.
Nuestra estrategia se ha basado en un todos contra uno. El líder detecta al enemigo que tiene más cercano, el cual debería ser el robot que más problemas le puede causar, y indica a los droides que le están apuntando que le ataquen. Si este puede ser víctima de fuego amigo, informa a los droides de manera que estos dejan de disparar hasta que el lider salga de rango. De la misma manera el líder solo ataca a un enemigo cuando lo tiene cerca, evitando así atacar a droides aliados y dandole la opción de seguir haciendo movimientos aleatorios por el mapa.

Documentación de Leader2.java

Descripción

El archivo "Leader2.java" contiene el código fuente de un robot que actúa como líder en el juego Robocode. El líder coordina un equipo de robots y toma decisiones estratégicas basadas en la posición de los enemigos y otros eventos del juego.

Clases y Métodos

run(): Método principal que controla el comportamiento del robot líder. En este método, el líder gestiona la posición de los enemigos, apunta y dispara a los enemigos cercanos, y toma decisiones estratégicas, como moverse aleatoriamente.

onScannedRobot(ScannedRobotEvent e): Este método se llama cuando el robot escanea un enemigo. Aquí, el líder calcula la posición del enemigo, realiza disparos estratégicos y coordina acciones con los robots aliados.

onHitWall(HitWallEvent event): Se ejecuta cuando el líder choca contra una pared, lo que le permite realizar maniobras de recuperación.

onHitByBullet(HitByBulletEvent event): Este método se llama cuando el líder es alcanzado por un disparo enemigo. Reacciona girando y moviéndose para evitar más daño.

onHitRobot(HitRobotEvent event): Maneja el evento cuando el líder choca con otro robot. Realiza acciones de giro y retroceso para evitar colisiones.

Variables

En cuanto a las variables usadas en nuestro código "Leader2.java", tenemos las siguientes:

HashMaps

private Map<String, Point> enemiesPositions = new HashMap<String, Point>():
Guarda las posiciones (coordenadas X e Y) de los enemigos escaneados en el campo de batalla. Asocia el nombre de cada enemigo con su posición.

private Map<String, Double> enemiesBearings = new HashMap<String, Double>():
Guarda los valores de los Bearings (ángulos) de los enemigos en relación con el robot líder. Asocia el nombre de cada enemigo con su Bearing.

private Map<String, Double> enemiesEnergies = new HashMap<String, Double>():
Guarda los niveles de energía de los enemigos escaneados en el campo de batalla. Asocia el nombre de cada enemigo con su nivel de energía.

Booleanos

boolean liderEntreEnemigoDroide:
Indica si el líder se encuentra entre un enemigo y un Droid aliado. Ayuda a tomar decisiones estratégicas sobre si los Droids deben disparar o no.

Tipos de Datos Primitivos

double, int, String, boolean:
Estos tipos de datos primitivos se utilizan para representar valores numéricos, cadenas de texto y valores booleanos necesarios para el funcionamiento del robot. Por ejemplo, se utilizan para almacenar coordenadas, ángulos, nombres de enemigos, decisiones estratégicas, etc.

Estructura Mensaje

Dentro de la estructura de mensaje, tenemos las siguientes variables:

private int codigo:

Variable encargada de indicar qué se envía en el mensaje. Puede tomar diferentes valores según el tipo de información que se transmita.
Cod = 0 → Información de nuestra posición (Posición de un aliado)
Cod = 1 → Información del enemigo más cercano a nuestra posición

private double x:
Variable encargada de guardar la coordenada X del enemigo. Se utiliza para transmitir la posición X de un aliado o un enemigo en el mensaje.

private double y:
Variable encargada de guardar la coordenada Y del enemigo. Se utiliza para transmitir la posición Y de un aliado o un enemigo en el mensaje.

private String Name:
Variable encargada de guardar el nombre del enemigo. Se utiliza para identificar a un enemigo específico en el mensaje.
private int shouldNotFire:
Variable encargada de indicar si se debe o no realizar un disparo. Ayuda a coordinar las acciones de los Droids en función de la posición del líder y otros factores.

Funcionalidad

El robot líder en "Leader2.java" realiza las siguientes tareas:

- Escanea y rastrea a los enemigos en el campo de batalla.
- Calcula la distancia y el ángulo a los enemigos más cercanos.
- Dispara a los enemigos cuando sea seguro hacerlo.
- Dispara a los enemigos cercanos para eliminarlos cuanto antes.
- Realiza movimientos aleatorios para evitar ser un objetivo fácil.
- Coordina con otros robots aliados y comparte información estratégica.
- Envía mensajes a los droides para indicar la posición de los enemigos a los droides.
- Cambia de dirección aleatoriamente cuando es golpeado para evitar que le golpeen de nuevo.
- Si choca con un enemigo, cambia de dirección para continuar avanzando.

Documentación de Droids.java

Descripción

El archivo "Droids.java" contiene el código fuente de un robot Droid en el juego Robocode. El Droid es parte de un equipo liderado por otro robot y puede recibir mensajes del líder para coordinar sus acciones en el campo de batalla.

Clases y Métodos

run(): El método principal que controla el comportamiento del Droid. En este método, el Droid se mueve aleatoriamente, girando y avanzando en direcciones aleatorias.

onMessageReceived(MessageEvent e): Se llama cuando el Droid recibe un mensaje del líder. El código interpreta los mensajes y toma acciones basadas en su contenido, como disparar a los enemigos o esquivar disparos.

onHitWall(HitWallEvent event): Maneja el evento cuando el Droid choca contra una pared, permitiendo maniobras de recuperación.

onHitByBullet(HitByBulletEvent event): Se ejecuta cuando el Droid es alcanzado por un disparo enemigo. Reacciona girando y moviéndose para evitar más daño.

onHitRobot(HitRobotEvent event): Gestiona el evento cuando el Droid choca con otro robot. Realiza acciones de giro y retroceso para evitar colisiones.

Variables

En cuanto a las variables usadas en nuestro código "Droids.java", tenemos las siguientes:

Tipos de Datos Primitivos
double, int, boolean:
Estos tipos de datos primitivos se utilizan para representar valores numéricos y booleanos necesarios para el funcionamiento de los Droids. Por ejemplo, se utilizan para rastrear el ángulo en grados del Droid, contadores de eventos y decisiones estratégicas.

Funcionalidad

El robot Droid en "Droids.java" realiza las siguientes tareas:

- Se mueve aleatoriamente en el campo de batalla, evitando ser un objetivo fácil.
- Puede recibir mensajes del líder y tomar acciones estratégicas basadas en esos mensajes.
- Dispara a los enemigos o esquiva disparos según las instrucciones del líder.
- Reacciona ante colisiones con otros robots, evitando el daño.
- Evita disparar al líder si este se encuentra entre un enemigo y el droide.