1.5. Robocode

Robocode es una plataforma de programación y un entorno de desarrollo diseñado para crear y competir con robots virtuales en un entorno de combate simulado. Fue creado por Mathew Nelson y es ampliamente utilizado como una herramienta educativa para aprender conceptos de programación, inteligencia artificial y estrategias de toma de decisiones.

1.5.1. Características principales

- **Robots programables**: Los usuarios pueden programar robots en Java, .NET (C#) u otros lenguajes compatibles. Cada robot tiene un conjunto de acciones que puede realizar, como moverse, disparar, escanear a otros robots y evitar obstáculos.
- **Combates simulados**: Los robots compiten en una arena virtual, donde el objetivo es destruir a los oponentes mientras evitan ser destruidos. El combate es en tiempo real y se basa en la estrategia y el código implementado.
- **Física y mecánica realista**: Robocode simula aspectos como la velocidad, la rotación del cañón, el radar y la energía, lo que obliga a los programadores a considerar factores como la precisión, el movimiento estratégico y la gestión de recursos.
- **Aprendizaje de programación**: Es una excelente herramienta para aprender y practicar conceptos de programación orientada a objetos, algoritmos, inteligencia artificial y lógica de control.
- **Comunidad y competencias**: Robocode tiene una comunidad activa de entusiastas que comparten códigos, estrategias y organizan torneos. Esto fomenta la mejora continua y el intercambio de ideas.
- **Plataforma multiplataforma**: Robocode es compatible con Windows, macOS y Linux, lo que lo hace accesible para una amplia audiencia.

1.5.2. Instalación de Robocode

Para instalar Robocode, sigue los siguientes pasos:

1. Descarga el instalador desde la página oficial: http://robocode.sourceforge.net/.
2. Ejecuta el archivo de instalación: `robocode-1.9.3.2-setup.jar`.
3. En sistemas Windows, puedes crear un acceso directo durante el proceso de instalación.
4. Para instrucciones detalladas de instalación, consulta la guía en http://robowiki.net/wiki/Robocode/Download_And_Install.

1.5.3. Recursos adicionales

Para aprender más sobre Robocode, puedes consultar los siguientes recursos:

- **Página oficial:** http://robocode.sourceforge.net/
- **Guía de instalación y uso:** http://robowiki.net/wiki/Robocode/Download_And_Install
- **Tutorial en video:** https://www.youtube.com/watch?v=V7YvEsmPVUQ&list=PLt8O2QMwVRH0index=5

1.5.4. Funcionamiento

- Los robots son programas que implementan una serie de métodos para controlar su comportamiento, como :
  - `onScannedRobot` (cuando detecta un oponente),
  - `onHitByBullet` (cuando es golpeado por un disparo)
  - `run` (el comportamiento principal del robot).
- Los combates pueden ser entre dos robots o en modo "melee" (todos contra todos).
- El robot ganador es el que sobrevive o acumula más puntos basados en su desempeño.

1.5.5. Usos de Robocode

- **Educación**: Es una herramienta popular en cursos de programación e inteligencia artificial.
- **Entretenimiento**: Muchos lo usan como un pasatiempo para crear robots cada vez más eficientes.
- **Competición**: Los torneos de Robocode son comunes en universidades y comunidades de programadores.

En resumen, Robocode combina programación, estrategia y diversión, ofreciendo una manera interactiva y desafiante de mejorar habilidades de codificación y pensamiento lógico. ¡Es como un juego de ajedrez, pero con robots programables!

1.5.6. Elementos de un tanque en Robocode

En **Robocode**, cada robot (o tanque) está compuesto por tres elementos principales: la **base**, el **cañón** y el **radar**. Cada uno de estos elementos tiene características y funcionalidades específicas que permiten al robot moverse, disparar y detectar a otros robots. A continuación, se describen en detalle:

Base (Cuerpo del robot)

La base es el cuerpo principal del robot y es responsable de su movimiento y posición en la arena.

- **Movimiento**: La base puede moverse hacia adelante, hacia atrás y girar a la izquierda o a la derecha.
- **Velocidad**: La velocidad de movimiento puede ajustarse, pero está limitada por un valor máximo.
- **Inercia**: Robocode simula la inercia, por lo que el robot no se detiene instantáneamente al dejar de moverse.
- **Colisiones**: Si el robot choca con una pared o con otro robot, se detiene y puede sufrir daños.
- **Energía**: La base tiene un nivel de energía que disminuye cuando es golpeada por disparos. Si la energía llega a cero, el robot es destruido.

Métodos comunes:

- `ahead(double distance)`: Mueve el robot hacia adelante una distancia específica.
- `back(double distance)`: Mueve el robot hacia atrás una distancia específica.
- `turnLeft(double degrees)` y `setTurnRight(double degrees)`: Giran la base del robot.
- `getVelocity()`: Devuelve la velocidad actual del robot.

Cañón (Arma)

El cañón es el componente que permite al robot disparar a otros robots. Está montado sobre la base y puede girar independientemente.

- **Rotación**: El cañón puede girar a la izquierda o a la derecha, independientemente de la base.
- **Disparo**: El robot puede disparar balas con diferentes niveles de potencia. La potencia del disparo afecta la velocidad de la bala, el daño causado y el retroceso que sufre el robot.
- **Enfriamiento**: Después de disparar, el cañón necesita un tiempo de enfriamiento antes de poder disparar nuevamente.
- **Retroceso**: Disparar con mayor potencia hace que el robot retroceda, lo que puede afectar su posición y estrategia.

Métodos comunes:

- `fire(double power)`: Dispara una bala con una potencia específica.
- `turnGunLeft(double degrees)` y `turnGunRight(double degrees)`: Giran el cañón.
- `getGunHeat()`: Devuelve el nivel de çalor"del cañón (si es mayor que 0, no se puede disparar).

Radar (Sistema de detección)

El radar es el componente que permite al robot detectar y rastrear a otros robots en la arena. Está montado sobre el cañón y puede girar independientemente.

- **Rotación**: El radar puede girar a la izquierda o a la derecha, independientemente de la base y el cañón.
- **Detección**: El radar escanea el entorno en busca de otros robots. Cuando detecta uno, se activa el evento `onScannedRobot`.
- **Rastreo**: El radar puede usarse para seguir a un robot enemigo y mantenerlo en el campo de visión.
- **Ángulo de escaneo**: El radar tiene un ángulo de escaneo limitado, por lo que debe girarse estratégicamente para cubrir toda el área.

Métodos comunes:

- `turnRadarLeft(double degrees)` y `turnRadarRight(double degrees)`: Giran el radar.
- `onScannedRobot(ScannedRobotEvent e)`: Evento que se activa cuando el radar detecta un robot enemigo.

Interacción entre los elementos

- **Independencia**: La base, el cañón y el radar pueden girar de manera independiente, lo que permite estrategias avanzadas como mover la base en una dirección mientras se apunta el cañón en otra.
- **Coordinación**: Una buena estrategia en Robocode implica coordinar los movimientos de la base, la dirección del cañón y el escaneo del radar para maximizar la eficiencia en combate.
- **Energía compartida**: La energía del robot es compartida entre todos los componentes. Si el robot es golpeado, pierde energía, lo que afecta su capacidad para moverse, disparar y escanear.

Ejemplo de uso en código (Java)

```
import robocode.Robot;
import robocode.ScannedRobotEvent;

/**
 * EjemploRobot: Un robot simple que se mueve, gira y dispara().
 */
public class EjemploRobot extends Robot {

    /**
     * Método principal que define el comportamiento del robot.
```

```
     */
    @Override
    public void run() {
        // Bucle principal del robot
        while (true) {
            // Mover la base de inmediato
            ahead(100); // Mueve el robot 100 píxeles hacia adelante
            turnRight(90); // Gira la base 90 grados a la derecha

            // Girar el cañón de inmediato
            turnGunRight(360); // Gira el cañón 360 grados a la derecha

            // Escanear con el radar de inmediato
            turnRadarRight(360); // Gira el radar 360 grados a la derecha

            // Disparar de inmediato si el cañón está listo
            if (getGunHeat() == 0) { // Verifica si el cañón está frío
                fire(1); // Dispara una bala con potencia 1
            }
        }
    }

    /**
     * Método que se ejecuta cuando el radar detecta un robot enemigo.
     * @param e Evento que contiene información sobre el robot detectado.
     */
    @Override
    public void onScannedRobot(ScannedRobotEvent e) {
        // Apuntar el cañón hacia el enemigo detectado
        double bearing = e.getBearing(); // Ángulo relativo del enemigo
        turnGunRight(bearing); // Gira el cañón hacia el enemigo

        // Disparar de inmediato si el cañón está listo
        if (getGunHeat() == 0) { // Verifica si el cañón está frío
            fire(1); // Dispara una bala con potencia 1
        }
    }
}
```