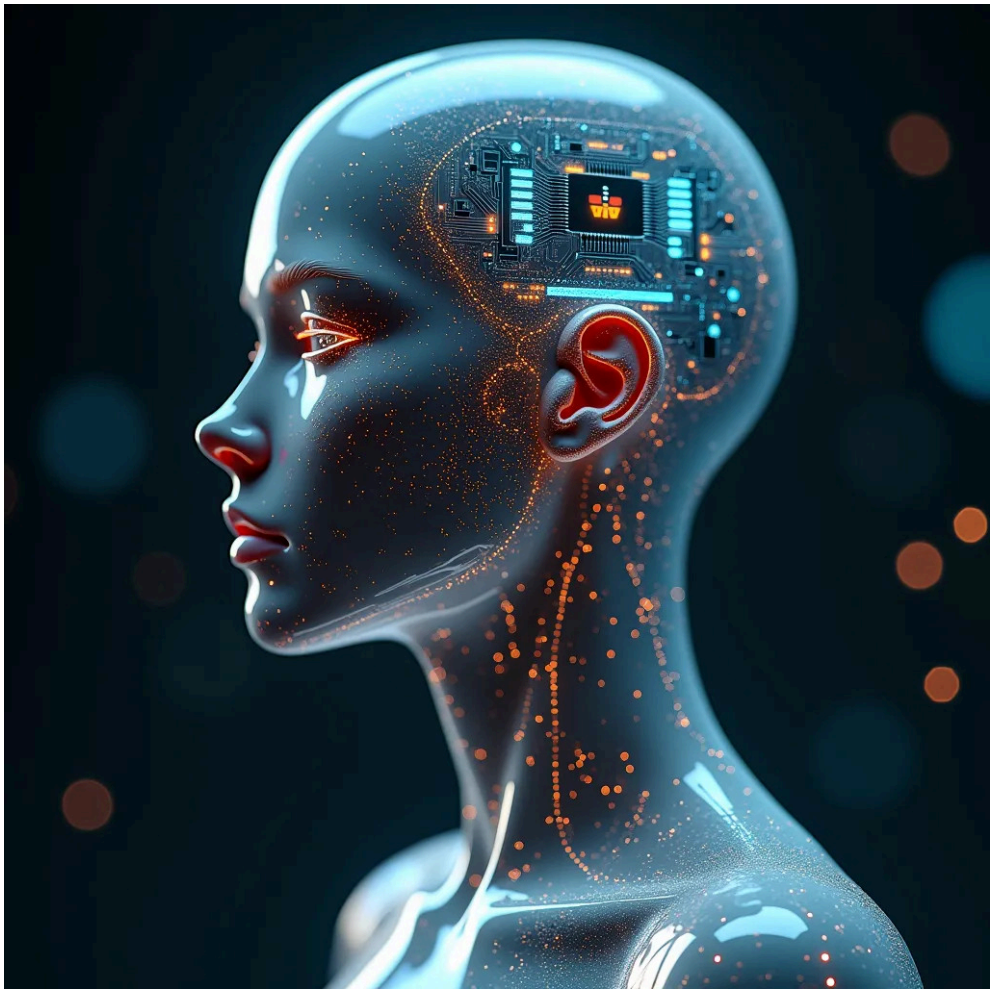
UD01 Práctica: Robocode Tank Royale. Construye el mejor, derrótalos a todos.



1. Introducción

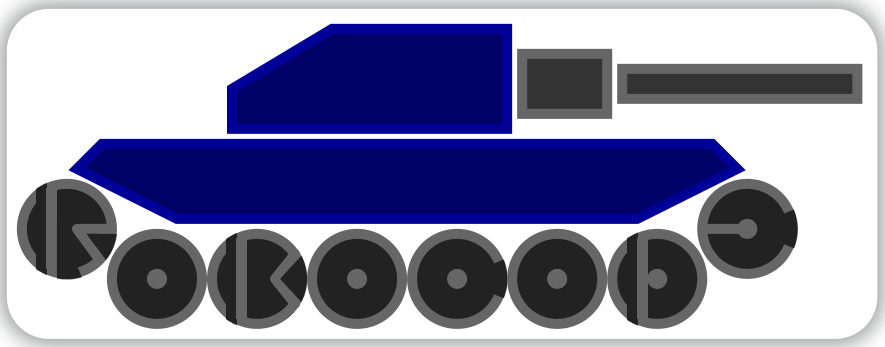
Robocode es un juego de programación donde el objetivo es codificar un bot en forma de tanque virtual para competir contra otros bots en un campo de batalla virtual. El jugador es el programador del bot, que no tendrá influencia directa en el juego. En cambio, el jugador debe escribir un programa para el cerebro del robot. El programa dice cómo debe comportarse y reaccionar el robot ante los eventos que ocurren en el campo de batalla.

El nombre Robocode proviene de una versión anterior del juego y es una abreviatura de "Código de robot". Con esta nueva versión, se utiliza el mundo "bot" en lugar de "robot".

El juego está diseñado para ayudarte a aprender a programar y mejorar tus habilidades de programación y divertirte mientras lo haces. Robocode también es útil a la hora de estudiar o mejorar el aprendizaje automático (En nuestro caso solo Inteligencia Artificial) en un juego rápido en tiempo real.

Las batallas de Robocode tienen lugar en un campo de batalla, donde pequeños robots tanque automatizados luchan hasta que solo queda uno, como en un juego Battle Royale. De ahí el nombre Tank Royale (RoboCode TankRoyale [RCTR].

Robocode no contiene sangre, visceras, personas ni política. Las batallas son simplemente por la emoción de la competición que tanto nos gusta.



Documentación del juego: https://robocode-dev.github.io/tank-royale/

Repositorio en GitHub: https://github.com/robocode-dev/tank-royale

Documentación de la API:

- **Java (JVM)**
    - API overview
    - Bot API
- **.Net**

- API overview
- Bot API

> **Importante** ¡**Ojo**! RCTR se basa en una versión más antigua de RoboCode, con un API diferente, y por lo tanto los robots anteriores, y el funcionamiento del campo de batalla han cambiado sustancialmente.

Por lo tanto, **mucha de la documentación que encontrareis es sobre el sistema antiguo**, en la que la parte de estrategias es válida, pero no el código que la acompaña. La tarea de traducción del antiguo sistema al nuevo se ha llevado a cabo en forma de un "bridge" entre las dos versiones, y está disponible en GitHub: https://github.com/robocode-dev/robocode-api-bridge

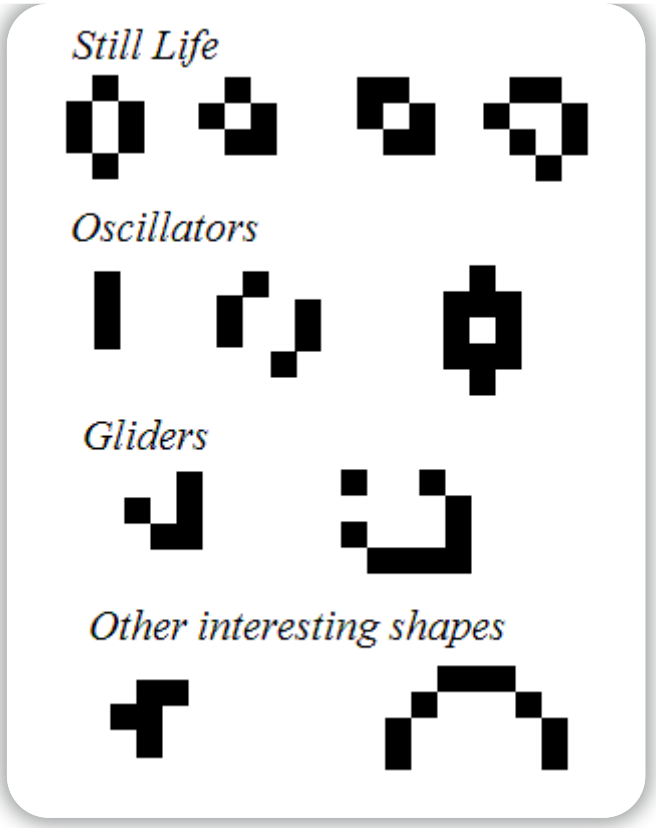
En cuanto a la documentación y estrategias, la API antigua todavía se puede encontrar en: https://robocode.sourceforge.io/docs/robocode/

Y una página que contaba con muchisima información sobre estrategias, robots, código, etc ya solo está disponible en archive.org (de momento no accesible por hackeo), la última versión cacheada que he encontrado disponible está en: https://web.archive.org/web/20200323061702/http://robowiki.net/

El Juego de la vida es un autómata celular diseñado por el matemático británico John Horton Conway en 1970. Es un juego de cero jugadores, en el que su evolución es determinada por un estado inicial, sin requerir intervención adicional.

Desde su publicación, ha atraído mucho interés debido a la gran variabilidad de la evolución de los patrones. Se considera que el Juego de la vida es un buen ejemplo de emergencia y autoorganización. Es interesante para científicos, matemáticos, economistas y otros observar cómo patrones complejos pueden provenir de la implementación de reglas muy sencillas. De hecho hay estudios sobre el uso de estos patrones para simular comportamientos humanos (Gómez & Guerreo, 2019 [El juego de la vida: una interpretación basada en el contagio de la euforia durante la formación de burbujas especulativas])

El Juego de la vida tiene una variedad de patrones reconocidos que provienen de determinadas posiciones iniciales. Poco después de la publicación, se descubrieron el pentaminó R, el planeador o caminador (en inglés: glider, conjunto de células que se desplazan) y el explosionador (células que parecen formar la onda expansiva de una explosión), lo que atrajo un mayor interés hacia el juego. Contribuyó a su popularidad el hecho de que se publicó justo cuando se estaba lanzando al mercado una nueva generación de miniordenadores baratos, lo que significaba que se podía jugar durante horas en máquinas que, por otro lado, no se utilizarían por la noche.


Still Life
Oscillators
Gliders
Other interesting shapes

2. Objetivo de la práctica

2.1. Robocode

Usando RoboCode Tank Royale (RCTR) intentaremos ponernos en el papel de los primeros programadores que dotaban de "inteligencia" a los primeros sistemas. Tal y como hemos visto en el apartado de teoría **estos sistemas solo reaccionan ante estímulos que previamente haya previsto su programador**, carecen de intuición (a no ser que sea simulada) y el resultado de su inteligencia es tan bueno como lo sea su programación.

- Analizar el código del Robot más simple (MyFirstBot), estudiando su funcionamiento y respuesta ante los estímulos. Igualmente, simularemos una partida para ver su comportamiento en el campo de batalla.
- Elegir un Robot más complejo, analizar su estrategia y comportamiento ante los esímulos en el campo de batalla.
- Simular una batalla entre ambos tipos y analizar los resultados.

2.2. El Juego de la vida

Conocer el funcionamiento de un autómata celular y ver su evolución en el tiempo.

3. Robocode. Pasos a seguir

1. **Preparación del entorno**

   En una sesión conjunta prepararemos nuestro entorno de trabajo, instalaremos todo lo necesario y haremos algún combate de pruebas.

2. **Mi primer Bot**

   Siguiendo la guía de la documentación, y con la ayuda del profesor todo el alumnado generará su primer Bot de muestra, aprenderemos a añadirlo a la batalla y a depurar su funcionamiento.

3. **¿Cómo mejoro mi Bot?**

   Estudiaremos otros bots diferentes y las estrategias y mejoras que aplican para dotarle de cierta inteligencia..

4. **Investigación y resultados**

   Deberéis simular/investigar el comportamiento de los bots, justificando los resultados de han obtenído.

4. El Juego de la vida

El juego

Se trata de un juego de cero jugadores, lo que quiere decir que su evolución está determinada por el estado inicial y no necesita ninguna entrada de datos posterior. El "**tablero de juego**" es una malla plana formada por cuadrados (las "células") que se extiende por el infinito en todas las direcciones. Por tanto, cada célula tiene 8 células "vecinas", que son las que están próximas a ella, incluidas las diagonales. Las células tienen dos estados: están "**vivas**" o "**muertas**" (o "encendidas" y "apagadas"). El estado de las células evoluciona a lo largo de unidades de tiempo discretas (se podría decir que por turnos).

El estado de todas las células se tiene en cuenta para calcular el estado de las mismas al turno siguiente. T**odas las células se actualizan simultáneamente en cada turno, siguiendo estas reglas:**

- **Nace**: Si una célula muerta tiene exactamente 3 células vecinas vivas "nace" (es decir, al turno siguiente estará viva).
- **Muere**: una célula viva puede morir por uno de 2 casos:
    - **Sobrepoblación**: si tiene más de tres vecinos alrededor.
    - **Aislamiento**: si tiene solo un vecino alrededor o ninguno.
- **Vive**: una célula se mantiene viva si tiene 2 o 3 vecinos a su alrededor.

Estados finales

Normalmente, después de un determinado número de ciclos, se puede llegar a alguno de los siguientes estados finales:

- **Extinción**: Al cabo de un número finito de generaciones desaparecen todos los miembros de la población o células vivas.
- **Estabilizacion**: Al cabo de un número finito de generaciones la población queda estabilizada, ya sea de forma rígida o bien de forma oscilante entre dos o más formas
- **Crecimiento constante**: La población crece turno tras turno y se mantiene así un número infinito de generaciones. En un principio esta evolución solo se contemplo de forma teórica, aunque más tarde se encontrarán patrones que crecían de forma indefinida, durante un número infinito de turnos.

  Algunos ejemplos de los patrones que se van generando y su explicación los tienes disponibles https://es.wikipedia.org/wiki/Juego_de_la_vida

Práctica

Utilizando un simulador del Juego de la vida https://conwaylife.com/ debes crear un patrón más o menos simple y simular su evolución en el tiempo. El patrón generado no se debe extinguir rápidamente, lo ideal es obtener un patrón estable.

5. ¿Qué debo entregar?

A través de moodle el alumno deberá entregar **un archivo documento** que contenga:

La memoria en formato ODT debe contener al menos los siguientes apartados:

- Datos del alumno
- Robocode
    - Descripción del funcionamiento del bot simple: estructura, métodos, análisis y evolución de la solución adoptada, etc.
    - Descripción del funcionamiento del bot inteligente seleccionado: estructura, métodos, análisis y evolución de la solución, etc.
    - Descripción detallada del comportamiento en una batalla.
    - Conclusiones
    - Bibliografía
- Juego de la vida
    - Patrón utilizado
    - Datos de la simulación realizada
    - Patrón final obtenído
    - Comparación del patrón con alguno de los conocidos
    - Conclusiones

6. Fuentes de información

- Wikipedia
- GhatGPT
- Modelos de Inteligencia Artificial (Ed. Marcombo)
- https://iep.utm.edu/artificial-intelligence/
- https://es.wikipedia.org/wiki/Juego_de_la_vida
- https://conwaylife.com/