

Práctica 2.Movimiento de una agente en un mundo bidimensional

DISEÑO BASADO EN AGENTESCURSO 2023/2024



**UNIVERSIDAD
DE GRANADA**



MANUEL JESÚS COBO MARTÍN

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL |
UNIVERSIDAD DE GRANADA

1 Contexto

En la práctica anterior hemos visto el funcionamiento global de JADE, desde como lanzar el servidor, a como crear un agente, añadirle diferentes tipos de comportamientos y ejecutar el agente en el servidor.

Estos conocimientos son básicos a la hora de crear un sistema multiagentes, y los usaremos para el desarrollo de las siguientes prácticas.

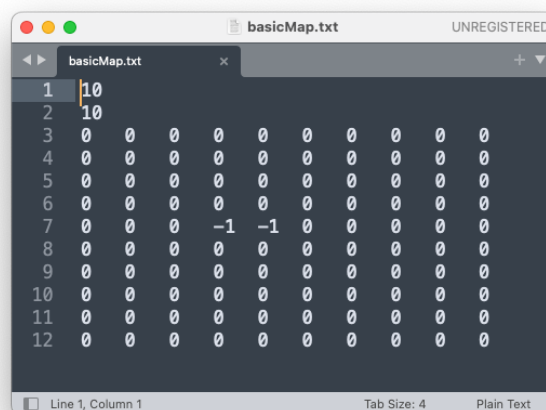
En esta práctica, desarrollaremos un agente inteligente que sea capaz de desplazarse por un mundo de dos dimensiones, evitando obstáculos, y dirigiéndose hacia una posición objetivo. Usando la terminología explicada en clase, el agente que implementaremos será reactivo con estados internos.

La práctica se realizará en grupos de 4, salvo excepciones debidamente justificadas. Todos los componentes del equipo trabajar por igual de forma equitativa. Cabe resaltar que todos los componentes no tienen por qué tener la misma calificación, en los casos en los que se estime que su contribución al grupo ha sido menor.

2 Descripción de la práctica

Para el desarrollo de la práctica tendremos que desarrollar todas las clases y métodos necesarios para que el agente se desplace por un mundo bi-dimensional. Como mínimo, necesitaremos los siguientes componentes: mapa, entorno y agente.

El mapa lo representaremos como una matriz de NxM de enteros. En cada celda habrá un valor numérico que nos determinará si la celda se encuentra vacía, o hay un obstáculo por lo que el agente no podría situarse sobre esa celda. De este modo, un valor de -1 indica que la celda no es accesible, y un valor de 0 indicará que la celda está libre. El mapa lo almacenaremos como un fichero de texto, en el que cada línea será una fila de la matriz, estando las columnas delimitadas por tabulaciones. Para poder leer de forma correcta el mapa, necesitamos saber las dimensiones de este, por lo que la primera línea representará el número de filas y la segunda línea representará el número de columnas. Así, por ejemplo, un mapa sencillo podría representarse de la siguiente manera:



1	10								
2	10								
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	-1	-1	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0

Figura 1. Fichero de texto conteniendo un mapa de 10x10 con dos obstáculos

Hemos de señalar que en el fichero que representa al mapa no se guarda la posición inicial del agente, ni la posición del objetivo. Dichas posiciones iniciales deben indicarse en la configuración del agente, bien desde código o bien a través de argumentos. De este modo se podrá ejecutar el agente desde diferentes posiciones, para ver si es capaz de llegar siempre al objetivo.

El mapa, única y exclusivamente debe representar el entorno “físico” en el que se desenvuelve el agente. Usando el mapa, debemos determinar la información que necesitamos, y por lo tanto los sensores que tenemos que implementar. Dicho de otro modo, cómo ve el agente el mundo que le rodea. Como limitación, nuestro agente sólo podrá ver las celdas inmediatamente contiguas. Por lo que, si hay un obstáculo a dos celdas de distancia, el agente no lo podrá ver desde su posición actual.

Para actualizar la vista que tiene el agente del entorno que lo rodea, podríamos implementar una función `see()`, que dado la posición actual del agente, nos actualizará sus percepciones.

Respecto al agente, tenemos que modelar los comportamientos que vamos a añadirle, y el tipo de estos. Su utilización debe ser razonada. Como mínimo, el agente deberá ver el entorno, calcular la acción que más utilidad tiene (la que más se aproxima al objetivo) y ejecutar la acción. Para ayudarnos a visualizar donde está el agente, se deberá mostrar el mapa, con la posición del agente, y la posición del objetivo, y si fuera posible, las celdas por las que el agente ha pasado (la traza de ejecución del agente).

3 Ejercicios a resolver

Como se comentó al principio, el agente tendrá que ser capaz de dirigirse hacia su objetivo de la forma más eficiente posible, evitando obstáculos, y por supuesto, evitando salirse de los márgenes de su mundo.

En particular, el agente deberá, como mínimo, poder resolver los siguientes mundos o mapas:

- Sin obstáculos, yendo directo al objetivo.
- Con obstáculos básicos. Una línea horizontal de obstáculos, así como una línea vertical
- Con obstáculos cóncavos. El agente tendrá que ser capaz de enfrentarse a obstáculos en forma de U invertida.
- Con obstáculos convexos. El agente tendrá que ser capaz de enfrentarse a obstáculos en forma de U.

4 Evaluación de la práctica

La práctica se evaluará sobre 10 puntos, considerando los siguientes apartados:

- Correcta resolución de los mundos
 - o Mundo sin obstáculos y mundo con obstáculos básicos (1 punto).
 - o Mundo con obstáculos cóncavos (1 punto).
 - o Mundo con obstáculos convexos (1 puntos).
- Correcta resolución de un mundo sorpresa (1.5 punto).
- Código. Se valorará la correcta resolución de los ejercicios, el diseño e implementación de los distintos algoritmos y del sistema completo (3 puntos).
- Defensa de la práctica. Al igual que en las prácticas anteriores, en clase de prácticas se evaluará el funcionamiento y se pedirá que se expliquen ciertas partes del código. (1 punto)
- Presentación. En clase de teoría se realizará una presentación de 5 minutos explicando las decisiones de diseño e implementación que se han tomado, así como los errores cometidos y como se han resuelto los mismos. La presentación se evaluará de forma colectiva (1.5 puntos)

5 Documentos a entregar

Para la evaluación de la práctica se deberá entregar el código fuente, una memoria explicando la estructura de clases, así como las decisiones de diseño e implementación tomadas, y la presentación o video que se usará el día de la presentación.