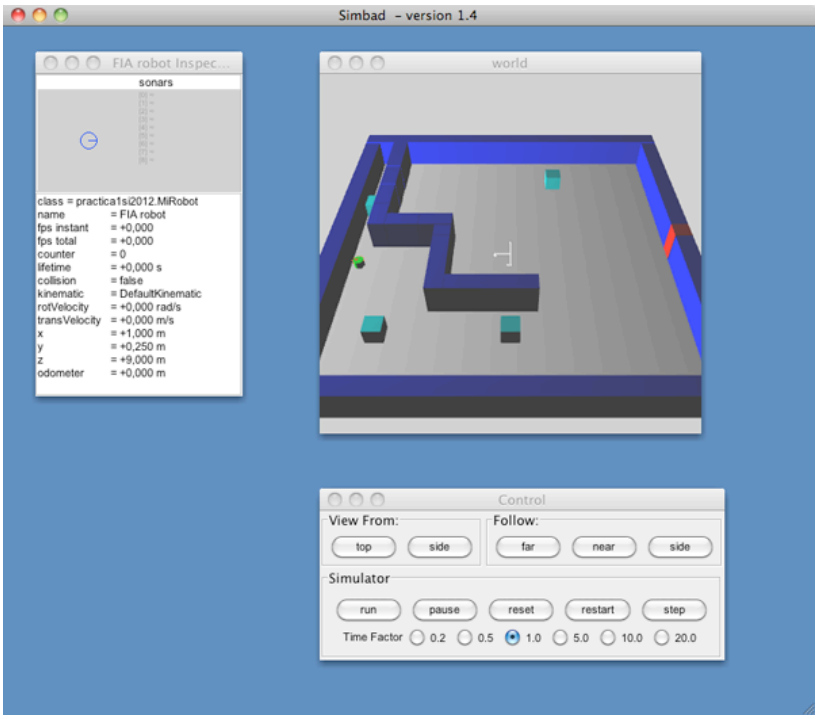
PRÁCTICA 2: BÚSQUEDA HEURÍSTICA. A*

1. Objetivos

- Comprender el funcionamiento de la búsqueda heurística y en concreto del algoritmo A*.
- Implementar el algoritmo A* y saber cómo seleccionar una heurística apropiada al problema.
- Realizar un análisis cuantitativo respecto al número de nodos explorados con este algoritmo.

2. Enunciado

En esta práctica y en la siguiente se desarrollará un sistema capaz de guiar a un robot en su entorno desde el punto de partida hasta la meta. En primer lugar, en esta práctica, se utilizará el algoritmo A* para calcular el camino óptimo para llegar a la meta y posteriormente, en la siguiente práctica, se utilizará un sistema experto difuso para guiar al robot hasta la meta siguiendo el camino calculado por A*.



2.1 Simbad

Para el desarrollo de esta práctica vamos a utilizar Simbad. Es un simulador robótico 3D desarrollado en Java. No es un simulador de un mundo real, sino que ha sido desarrollado para fines académicos, intentando tener una base simple para el estudio de algoritmos de inteligencia artificial. Este simulador nos permite escribir el controlador del robot, modificar el entorno y hacer uso de los sensores disponibles.

Es posible encontrar más información en la página del proyecto:
http://simbad.sourceforge.net/.

2.2 Creación del controlador del robot

Para crear un controlador del robot son necesarias tres cosas:

- Un programa principal
- Una descripción del entorno
- Una clase del robot

Programa principal
El programa principal debe lanzar Simbad con la descripción del entorno deseado.

```
Simbad frame = new Simbad(new MiEntorno(), false);
```

Descripción del entorno
Esta clase debe especificar cómo es el entorno, las paredes, los obstáculos y el punto de inicio del robot. Se pueden utilizar objetos de diferentes tipos:

```
add(new Wall(new Vector3d(10, 0, 0), 20, 1, 2, this));
add(new Box(new Vector3d(10,0,-(10)), new Vector3f(1, 1, 1), this);
add(new Arch(new Vector3d(3,0,-3), this));
```

Y el robot se añade como:

```
add(new MiRobot(new Vector3d(0,0,0), "mi robot");
```

La clase del robot
Esta clase debe contener el controlador del robot. Deriva de la clase Agente. Es necesario sobrescribir dos métodos: initBehavior y performBehavior.

- **initBehavior**: Este método lo llama el simulador al principio de la vida del agente. Es donde debemos introducir el código para inicializar el robot.
- **performBehavior**: El simulador llama a este método en cada paso (20 veces/segundo). Es en este método donde se debe introducir el comportamiento del robot.

Estas clases ya están creadas en el proyecto NetBeans que tenéis disponible y que se explica en el punto siguiente. En la práctica únicamente tendréis que modificar la clase MiRobot.

2.3 Puesta en marcha del proyecto

Se debe descargar el proyecto de Netbeans disponible en Campus Virtual y abrirlo en Netbeans.
Se deben añadir al proyecto las librerías:

- Simbad1.4.SI.jar
- jFuzzyLogic_2.0.6.jar

Para esto se debe acceder a las propiedades del proyecto (pulsando el botón derecho sobre la raíz del proyecto). Y en el apartado Librerías incluir estas dos librerías desde el botón Add Library…
Para utilizar este simulador es necesaria la librería Java3D. Si no está instalada en vuestra máquina la debéis instalar. Es posible descargar esta librería desde su página: http://java3d.java.net/

2.4 Estructura del proyecto

Este proyecto se compone de diferentes clases:

- **Main.java**. Es la clase ejecutable del proyecto. Esta clase lee los parámetros de entrada y crea una instancia de la clase Practica1.java. El programa requiere que se le pase como argumento un fichero.txt con la estructura del entorno.
- **Practica1.java.** Esta clase contiene los parámetros que definen el mundo (el tamaño del mundo, el punto de origen del robot y el punto destino y el mundo en sí). Lee el mundo desde el fichero que se le pasa como parámetro y crea una instancia del simulador Simbad pasándole el entorno.

- **MiEntorno.java.** Esta es la clase que define el entorno del robot en el simulador Simbad. Crea el entorno mediante paredes y cajas y crea el robot (como instancia de la clase MiRobot.java).
- **MiRobot.java.** Esta es la clase que define el comportamiento del robot. En esta clase definiremos el algoritmo A*.

2.5 Especificación del mundo

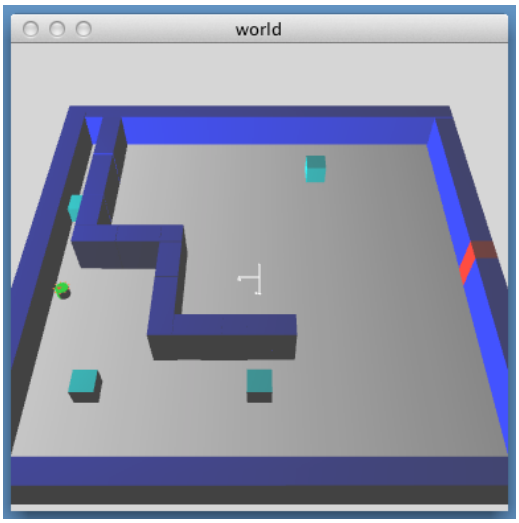
La especificación del mundo se realiza en un fichero de texto que se pasa como parámetro al programa.
Este fichero de texto tendrá la siguiente estructura:

- En la primera línea se especifica el tamaño del mundo. El mundo será un cuadrado. Por ejemplo, si se indica un 10, se creará un mundo de 10x10. (El simulador no muestra correctamente mundos de un tamaño mayor de 20).
- En la segunda línea se debe indicar el punto origen del robot. Se especifica en qué fila empezará el robot, ya que la columna será siempre la primera.
- En la tercera línea se especifica el punto destino del robot. Se especifica la fila en que se desea que termine el robot, ya que la columna será la última.
- El resto de filas especificarán el entorno en sí teniendo en cuenta que:
    - Si tiene un * (asterisco) es una pared
    - Si tiene un – (guión) es un obstáculo
    - Si tiene un . (punto) es un espacio vacío

A continuación se muestra un ejemplo de mundo:

```
20
10
10
* * * * * * * * * * * * * * * * * * * *
* . * . . . . . . . . . . . . . . . . *
* . * . . . . . . . . . . . . . . . . *
* . * . . . . . . . . . . - . . . . . *
* . * . . . . . . . . . . . . . . . . *
* . * . . . . . . . . . . . . . . . . *
* - * . . . . . . . . . . . . . . . . *
* . * . . . . . . . . . . . . . . . . *
* . * . . . . . . . . . . . . . . . . *
* . * * * * * . . . . . . . . . . . . *
* . . . . . * . . . . . . . . . . . . *
* . . . . . * . . . . . . . . . . . . *
* . . . . . * . . . . . . . . . . . . *
* . . . . . * * * * * * . . . . . . . *
* . . . . . . . . . . . . . . . . . . *
* . . - . . . . . - . . . . . . . . . *
* . . . . . . . . . . . . . . . . . . *
* . . . . . . . . . . . . . . . . . . *
* * * * * * * * * * * * * * * * * * * *
```

Este mundo se mostrará en el simulador Simbad como:



3. Diseño del algoritmo de búsqueda A*

En este problema de planificación el robot se encuentra en un mundo y debe encontrar un objetivo, por lo tanto, tiene que calcular una ruta para llegar allí. El objetivo principal será encontrar el camino de coste mínimo. Para ello vamos a utilizar el algoritmo A*. Éste es un algoritmo de búsqueda heurística, por lo tanto utiliza una función heurística, que nos dará un valor para cada celda. La función heurística es una estimación optimista de cómo de lejos está el objetivo.

```
h(x,y) ~<= distancia al objetivo
```

*Pseudocódigo del algoritmo A**

```
Alg A*
	listaInterior = vacio
	listaFrontera = inicio

	mientras listaFrontera no esté vacía
		n = obtener nodo de listaFrontera con menor f(n) = g(n) + h(n)
		listaFrontera.del(n)
		listaInterior.add(n)

		si n es meta
			devolver
			reconstruir camino desde la meta al inicio siguiendo los punteros
		fsi

		para cada hijo m de n que no esté en listaInterior
			g'(m) = n·g + c(n, m)  //g del nodo a explorar m

			si m no está en listaFrontera
				almacenar la f, g y h del nodo en (m.f, m.g, m.h)
				m.padre = n
				listaFrontera.add(m)
			sino si g'(m) es mejor que m.g  //Verificamos si el nuevo camino es
mejor
				m.padre = n
				recalcular f y g del nodo m
			fsi
		fpara
	fmientras
	devolver no hay solución
falg
```

3.1 Implementación del algoritmo de búsqueda A*

La implementación del algoritmo A* se debe realizar en la clase MiRobot.java.
En esta clase tenéis ya creado un método:

```
public int AEstrella()
```

En este método es donde se debe implementar el algoritmo A*. Ya que es este método el llamado desde la función **initBehavior**.
Podréis crear métodos auxiliares que necesitéis y nuevas clases.
El algoritmo A* debe mostrar al final de su ejecución la solución al problema de la siguiente manera:

- En primer lugar debe mostrar el camino óptimo obtenido.
- En segundo lugar debe mostrar los nodos explorados.

Ejemplo de solución para el mundo mostrado anteriormente.

```
Entra en initBehavior
Solución del A*
Camino
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. X X X X X . . . . . . . . . . . . X .
. . . . . X . . . . . . . . . . . . X .
. . . . . X . . . . . . . . . . . . X .
. . . . . X . . . . . . . . . . . . X .
. . . . . X . . . . . . . . . . . . X .
. . . . . X X X X X X X X X X X X X X .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
Nodos explorados
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1  0  1  2  3  4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 33 -1
-1 -1 14 10  7  5 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1
-1 -1 -1 11  9  6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 31 -1
-1 -1 -1 -1 13  8 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 30 -1
-1 -1 -1 -1 -1 12 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 29 -1
-1 -1 -1 -1 -1 15 16 17 18 19 20 21 22 23 24 25 26 27 28 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Para esto, ya están creadas dos matrices:

- La variable char camino[][], debe contener el camino solución. Esta variables se inicializa a '.' en el constructor, y se debe indicar con una 'X' (mayúscula) el camino solución del A*.
- La variable int expandidos[][], debe contener el orden en el que se expanden los nodos. Esta variable está inicializada a -1, y se debe indicar cuándo se expande cada nodo.

4. Documentación

La documentación es una parte muy importante de la práctica. Como mínimo debe contener:

- Pseudocódigo del algoritmo A* que se ha implementado. Explicando con ejemplos cada uno de los posibles casos que nos podemos encontrar a la hora de evaluar nuevos nodos.
- Explicación de cuál es la mejor heurística para este problema y que ocurre cuando *h* se acerca o aleja de *h**. Analizando varias heurísticas y viendo cómo repercuten en el número de nodos explorados.
- Explicación y traza de un problema pequeño donde se observe el funcionamiento del algoritmo A*.
- Se debe realizar un conjunto de pruebas bien diseñado que abarque todas las casuísticas del problema.

5. Entrega de la práctica

La fecha límite de entrega es el lunes 26 de noviembre de 2012 hasta las 12:00 de la noche.

La entrega se realizará a través de Campus Virtual en el apartado de Evaluación>Controles. Para ello se seleccionará la asignatura FIA. La entrega constará de un fichero .ZIP que contendrá dos carpetas:

- src, donde se incluirá el proyecto de netbeans sin el directorio /lib.
- doc donde estará disponible la documentación en formato PDF explicando de forma detallada la implementación del algoritmo con las pruebas realizadas.

El nombre del fichero ZIP tendrá el siguiente formato: "NombreApellido1Apellido2.ZIP". Un fichero de ejemplo sería RaulMartinezSerra.zip

- IMPORTANTE!!! no cumplir cualquiera de las normas de formato/entrega anteriores supondrá un suspenso en la práctica.
- IMPORTANTE!!! Recordad que las prácticas son individuales y NO se pueden hacer en parejas o grupos. Cualquier código copiado supondrá un suspenso de la práctica para todas las personas implicadas en la copia.
- IMPORTANTE!!! la documentación deberá incluir una **sección de experimentación** en la que se describan las pruebas realizadas, dejando bien claro el objetivo de las pruebas, cómo se han llevado a cabo, qué información se ha recopilado a partir de las mismas, y qué conclusiones se han extraído. Una documentación sin este apartado se considerará suspensa. Se deberán tener las dos partes de la práctica (documentación y código) aprobadas por separado para poder aprobar la práctica.

Software útil

- Netbeans (www.netbeans.org/)