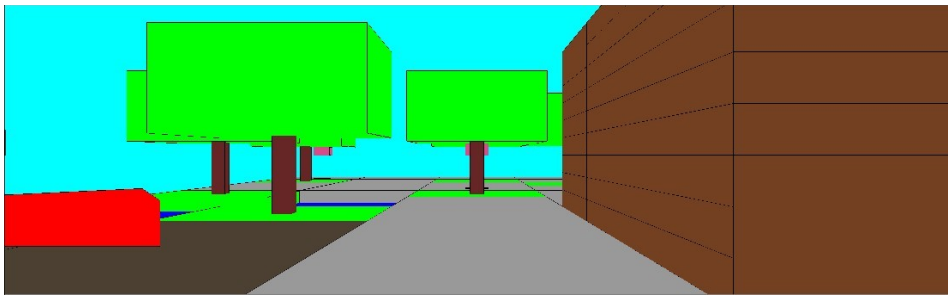
INTELIGENCIA ARTIFICIAL

E.T.S. de Ingenierías Informática y de Telecomunicación

Práctica 2



Agentes Reactivos/Deliberativos: los extraños mundos de BelKan

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
UNIVERSIDAD DE GRANADA
Curso 2018-2019

1. Introducción

La segunda práctica de la asignatura ***Inteligencia Artificial*** consiste en el diseño e implementación de un agente reactivo/deliberativo, capaz de percibir el ambiente y actuar considerando una representación de las consecuencias de sus acciones y siguiendo un proceso de búsqueda. Se trabajará con un simulador software. Para ello, se proporciona al alumno un entorno de programación, junto con el software necesario para simular el entorno.

En esta práctica se diseñará e implementará un agente reactivo y deliberativo basado en los ejemplos del libro *Stuart Russell, Peter Norvig, "Inteligencia Artificial: Un enfoque Moderno", Prentice Hall, Segunda Edición, 2004*. El simulador que utilizaremos ha sido desarrollado por los profesores de la asignatura, a partir de la versión inicial del profesor Tsung-Che Chiang de la NTNU (Norwegian University of Science and Technology, Trondheim).

Originalmente, el simulador estaba orientado a experimentar con comportamientos en aspiradoras inteligentes. Las aspiradoras inteligentes son robots de uso doméstico que disponen de sensores de suciedad, un aspirador y motores para moverse por el espacio (ver Figura 1). Cuando una aspiradora inteligente se encuentra en funcionamiento, esta recorre toda la dependencia o habitación donde se encuentra, detectando y succionando suciedad hasta que, o bien termina de recorrer la dependencia, o bien aplica algún otro criterio de parada (batería baja, tiempo límite, etc.).

Figura 1: Aspiradora inteligente

Este tipo de robots es un ejemplo comercial más de máquinas que implementan técnicas de Inteligencia Artificial y, más concretamente, de Teoría de Agentes. En su versión más simple (y también más barata), una aspiradora inteligente presenta un comportamiento *reactivo* puro: busca suciedad, la limpia, se mueve, detecta suciedad, la limpia, se mueve, y continúa con este ciclo hasta que se cumple alguna condición de

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

parada. En otras versiones más sofisticadas, el robot es capaz de *recordar* mediante el uso de representaciones icónicas como mapas, lo cual permite que el aparato ahorre energía y sea más eficiente en su trabajo. Finalmente, las aspiradoras más sofisticadas (y más caras) pueden, además de todo lo anterior, planificar su trabajo de modo que se pueda limpiar la suciedad en el menor tiempo posible y de la forma más eficiente. Son capaces de detectar su nivel de batería y volver automáticamente al cargador cuando esta se encuentre a un nivel bajo. Estas últimas pueden ser catalogadas como *agentes deliberativos*.

En esta práctica, centraremos nuestros esfuerzos en implementar el comportamiento de un "personaje virtual" asumiendo un comportamiento reactivo y deliberativo. Utilizaremos las técnicas estudiadas en los temas 2 y 3 de la asignatura para el diseño de agentes reactivos y deliberativos.

2. Los extraños mundos de BelKan

En esta práctica tomamos como punto de partida el mundo de las aventuras gráficas de los juegos de ordenador para construir sobre él personajes virtuales que manifiesten comportamientos propios e inteligentes dentro del juego. Nos situaremos en un problema habitual en el desarrollo de juegos para ordenador y diseñaremos personajes que interactuarán de forma autónoma usando agentes reactivos/deliberativos.

2.1. El escenario de juego

Este juego se desarrolla sobre un mapa cuadrado bidimensional discreto que contiene como máximo 100 filas y 100 columnas. El mapa representa los accidentes geográficos de la superficie de parte de un planeta semejante a la Tierra. Sus elementos son considerados inmutables, es decir, el mapa no cambia durante el desarrollo del juego.

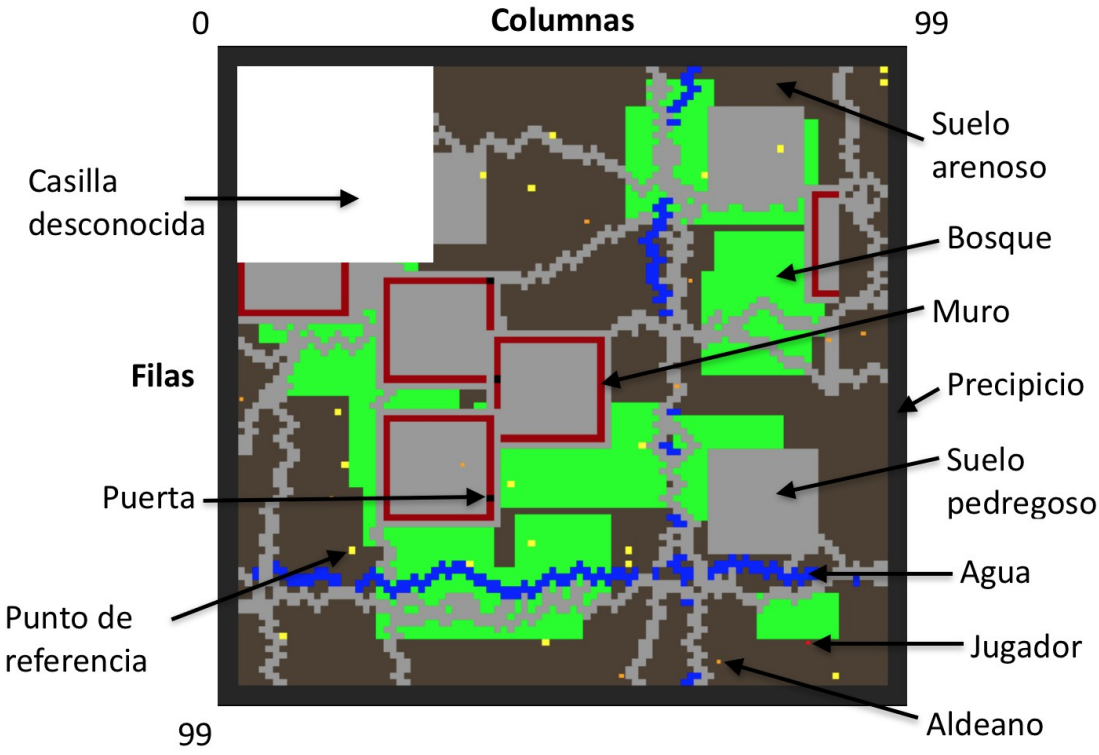
Representaremos dicha superficie usando una matriz donde la primera componente representa la fila y la segunda representa la columna dentro de nuestro mapa. Como ejemplo usaremos un mapa de tamaño 100x100 de caracteres. Fijaremos sobre este mapa las siguientes consideraciones:

- La casilla superior izquierda del mapa es la casilla [0][0].
- La casilla inferior derecha del mapa es la casilla [99][99].

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

Teniendo en cuenta las consideraciones anteriores, diremos que un elemento móvil dentro del mapa va hacia el NORTE si en su movimiento se decrementa el valor de la fila. Extendiendo esta convención, irá al SUR si incrementa su valor en la fila, irá al ESTE si incrementa su valor en las columnas y, por último, irá al OESTE si decrementa su valor en columnas.

Los elementos permanentes en el terreno son los siguientes:



- *Árboles o Bosque*, que se codifican con el carácter '**B**' y se representan en el mapa como casillas de color verde.
- *Agua*, que se codifica con el carácter '**A**' y tiene asociado el color azul. Nuestro personaje no sabe nadar… con lo cual si cae en una de estas casillas morirá.
- *Precipicios*, que se codifica con el carácter '**P**' y tiene asociado el color negro. Nuestro personaje no sabe volar… con lo cual si cae en una de estas casillas morirá.
- *Suelo pedregoso*, que se codifica con el carácter '**S**' y tiene asociado el color gris.

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

- *Suelo Arenoso*, que se codifica con el carácter '**T**' y tiene asociado el color marrón.
- *Punto de Referencia o PK*, que se codifica con el carácter '**K**' y se muestra en amarillo (explicaremos su utilidad más adelante).
- *Muros*, se codifican con el carácter '**M**' y son rojo oscuro.
- *Puertas*, se codifican con el carácter '**D**' y son gris oscuro.
- *Casilla aún desconocida*, se codifica con el carácter '?' y se muestra en blanco (representa la parte del mundo que aún no ha explorado).

Una característica peculiar de este mundo es que es cerrado. Eso significa que no se puede salir de él, ya que las tres últimas filas visibles al Norte son precipicios, y lo mismo pasa con las tres últimas filas/columnas del Sur, Este y Oeste. Esto no quiere decir que no hay precipicios en el resto del mapa.

Sobre esta superficie pueden existir elementos que tienen la capacidad de moverse por sí mismos. Estos elementos son:

- *Jugador*, que se codifica con el carácter '**j**' y se muestra como un triángulo rojo. Éste es nuestro personaje, sólo habrá un jugador a la vez.
- *Aldeanos*, que se codifican con el carácter '**a**' y se muestran como un cuadrado naranja. Son habitantes anónimos del mundo que se desplazan a través del mapa sin un cometido específico, simplemente intentan molestarnos en nuestros movimientos. Son sólo molestos, no son peligrosos.

2.2. Nuestro Personaje

Obviamente nuestro personaje es el protagonista de la historia y debe llevar a cabo su objetivo: ir de un punto específico en el mapa (origen) a otro punto específico en el mapa (destino).

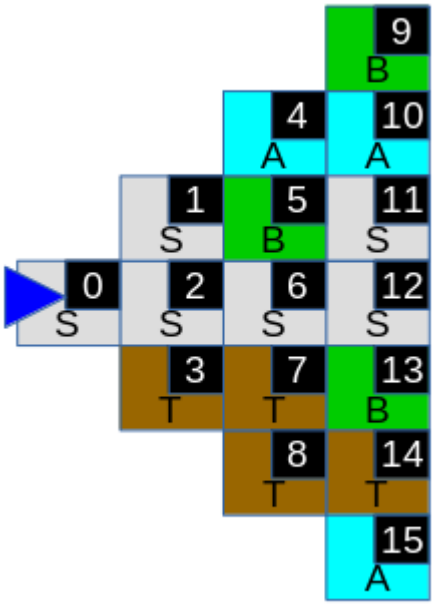
2.2.1. Sensores del personaje

El personaje en el simulador tiene forma de un triángulo y es rojo. Además, cuenta con un sistema visual que le permite ver las cosas que tiene delante. Esta visión se representa usando dos vectores de caracteres de tamaño 16. El primero de ellos lo denominaremos terreno y representa los elementos inmóviles de mapa, es decir, el terreno.

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

El segundo del ellos, que llamaremos superficie, representa qué objetos móviles se encuentran en la visión del personaje (es decir, aldeanos).

Para entender cómo funciona este sistema, veamos un ejemplo. Supongamos que el vector terreno contiene **SSSTABSTTBASSBTA**; así, su representación sobre un plano será la siguiente:



El primer carácter (posición 0) representa el tipo de terreno sobre el que se encuentra nuestro personaje. El tercer carácter (posición 2) es justo el tipo de terreno que tiene justo delante. Los caracteres de posiciones 4, 5, 6, 7 y 8 representan lo que está delante, pero con una casilla más de profundidad y apareciendo de izquierda a derecha. Por último, los caracteres de posiciones de la 9 a la 15 son aquellos que están a tres casillas, vistos de izquierda a derecha. La figura anterior representa las posiciones del vector en su distribución bidimensional (los números) y el carácter y su representación por colores como quedaría en un mapa.

De igual manera se estructura el vector **superficie,** pero en este caso indicando qué objetos móviles se encuentran en cada una de esas casillas.

El personaje cuenta con sensores que miden éstas y otras cuestiones:

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

- **Sensor de choque (`colision`)**: Es una variable booleana que se pondrá en verdadero en caso de que la última acción del jugador haya ocasionado un choque con un árbol, muro o puerta.
- **Sensor de vida (`reset`)**: Es una variable booleana que se podrá en verdadero en caso de que la última acción del jugador le haya llevado a su muerte… por ejemplo, si se ha precipitado al vacío en un precipicio. No os pongáis tristes, ya que morir en el juego implica que se vuelve a reaparecer en el mismo mundo, con una nueva posición y siempre mirando al norte.
- **Sensores de mensaje (`mensajeF`, `mensajeC`)**: En algunas ocasiones se le enviarán al jugador mensajes a través de estos dos sensores de tipo entero (ej: mensajeF = 10, mensajeC = 55), como puede ser su posición en el mapa (más detalles en la siguiente sección).
- **Sensor de destino (`destinoF`, `destinoC`)**: En todo momento, la posición del destino se puede saber a través de estos dos sensores de tipo entero. Las coordenadas de destino se pueden cambiar por la interfaz del usuario (pinchando en algún lugar del mapa o cambiando a mano los valores de Fila y Columna en Ir a…, ver captura de pantalla en la sección 4.2.1). Esta modificación del destino no puede realizarse en la versión sin interfaz.
- **Sensor de comportamiento (`nivel`):** Este es un sensor que informa en que nivel del juego se encuentra. Los valores posibles del sensor están entre 1 y 4 y tienen la siguiente interpretación:

  1 : El algoritmo de búsqueda es el de profundidad.

  2 : El algoritmo de búsqueda es el de anchura.

  3 : El algoritmo de búsqueda es el de costo uniforme.

  4 : Estamos en el nivel 2 del juego.
- **Sensor de tiempo consumido (`tiempo`):** Este sensor informa del tiempo que lleva consumido el agente en el cálculo de los caminos.

2.2.2. Datos del personaje compartidos con el entorno

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

Dentro de la definición del agente hay una matriz llamada **`mapaResultado`** en donde se puede interactuar con el mapa. Todo cambio en esta matriz se verá reflejado automáticamente en la interfaz gráfica.

2.2.3. Acciones que puede realizar el personaje

Nuestro personaje puede realizar varias acciones distintas durante el juego:

- *actFORWARD:* le permite avanzar a la siguiente casilla del mapa siguiendo su orientación actual. Para que la operación se finalice con éxito es necesario que la casilla de destino sea transitable para nuestro personaje.
- *actTURN_L:* le permite mantenerse en la misma casilla y girar a la izquierda 90º teniendo en cuenta su orientación.
- *actTURN_R:* le permite mantenerse en la misma casilla y girar a la derecha 90º teniendo en cuenta su orientación.
- *actIDLE:* como su nombre indica, no hace nada.

2.2.4. Coste de las acciones

Cada acción realizada por el agente tiene un coste relacionado con el tiempo que consume dicha acción (medida en instantes de simulación). En concreto, consideraremos que las acciones actTURN_L, actTURN_R y actIDLE tienen un coste de una unidad cada vez que se aplican medido ***en términos de instantes de simulación*** que consume su ejecución. El coste de actFORWARD también será de 1 excepto cuando el personaje se encuentre en la siguientes situaciones:

- si está en suelo arenoso el coste será de 2
- si está en bosque el coste será de 5
- si está en agua el coste será de 10

3. Objetivo de la práctica

El objetivo de esta práctica es dotar de un comportamiento inteligente a nuestro personaje usando un agente reactivo/deliberativo para definir las habilidades que le permitan alcanzar una meta concreta dentro del juego según el nivel seleccionado.

Para que sea más fácil resolver esta práctica, se han diseñado dos niveles con dificultad incremental, teniendo el primer nivel 3 comportamientos (2 a realizar por el estudiante) y un único comportamiento global en el segundo nivel (comportamiento 4).

Nivel 1: Agente deliberativo básico

La misión del agente es ir a un punto destino del mapa. En este nivel nuestro agente conoce perfectamente el terreno y no hay aldeanos en el mismo. Nuestro agente no puede atravesar muros, ni puertas, ni precipicios, y por lo tanto, deberá esquivar estos tipos de terreno ya que en otro caso chocará (en el caso de los muros y puertas) o morirá (en caso de precipicios).

Inicialmente el agente aparecerá de forma aleatoria sobre un mapa concreto conociendo su posición (a través de los sensores **`mensajeF`** y **`mensajeC`** que se activan cuando el jugador debe decidir su primera acción) y orientación sobre el mapa (siempre mira al norte al iniciar). El mapa es estático, es decir no hay cambios en su contenido de ningún tipo. El objetivo del agente es crear y llevar a cabo un plan de movimientos en el mapa para llegar con seguridad desde el origen al destino usando los algoritmos de búsqueda que se describen más abajo. Por supuesto, no debe morir injustificadamente ni chocarse con nada. Los algoritmos que se han de implementar son:

- **Comportamiento 1**: Búsqueda en profundidad.
- **Comportamiento 2**: Búsqueda en anchura.
- **Comportamiento 3**: Búsqueda de costo uniforme.

Nivel 2: Agente reactivo/deliberativo complejo

Para realizar este nivel es necesario haber hecho previamente el nivel anterior.

En el nivel 2 el agente no conoce el mapa ni sabe en donde se encuentra. Eso si, sabe que **mira hacia el norte al empezar la simulación**. El agente debe ir descubriendo el mapa poco a poco. Pero para empezar debe saber en que fila y columna está. Para ello debe recorrer el mapa hasta encontrar una casilla **PK** (punto de referencia) que le indicará

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

la posición real en fila y columna y desde ese momento podrá ir construyendo su propio mapa a medida que se dirija al destino.

Por lo tanto, el agente debe planificar un camino hacia un destino meta concreto (aunque no se conozca el mapa en su totalidad). Obviamente, al no conocer el mapa en su totalidad es posible que el agente planifique un camino por zonas del espacio que no le interesen (por coste o porque no sea posible pasar) y requiera tomar alguna decisión al respecto. Además, en este nivel hay aldeanos que se pasean sin un destino fijo. Un aldeano puede moverse continuamente, puede quedarse quieto, etc. Es decir, para ir de un punto a otro del mapa puede que nuestro plan inicial no funcione correctamente debido a que nos crucemos en el camino con un aldeano que nos entorpezca. Los aldeanos pueden detectarse gracias al sistema visual que tiene nuestro personaje; como se ha explicado anteriormente, no podemos saber donde están los aldeanos en el mapa, pero si podemos percibirlos si estamos cerca de ellos.

En estas situaciones deberemos implementar una aproximación combinando comportamiento reactivo y deliberativo, de manera que el agente debe encontrar un plan de navegación y, durante su ejecución, el agente debe considerar cómo actuar ante un posible fallo en la ejecución del plan.

En este nivel cada vez que el agente llegue al destino, éste cambiará de sitio. Es decir, deberá tratar de conseguir llegar a la mayor cantidad de destinos posible utilizando como máximo 3000 instantes de simulación o 300'' totales en tiempo de pensar el plan (lo que suceda antes). También es importante recordar que el agente no debe morir injustificadamente ni chocarse con ningún objeto inmóvil (si puede hacerlo con aldeanos).

4. El Software

Para la realización de la práctica se proporciona al alumno una implementación tanto del entorno simulado del mundo en donde se mueve nuestro personaje como de la estructura básica del agente reactivo/deliberativo.

4.1. Instalación

La versión que se proporciona del software es para el sistema operativo Linux (Distribución Ubuntu). Dicho software se puede encontrar en la parte de prácticas de la plataforma docente DECSAI.

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

El proceso de instalación es muy simple:

1. Se accede a la sección de Material de la Asignatura y se selecciona el archivo comprimido '***practica2.zip***'.
2. Se descarga en la carpeta de trabajo.
3. Se descomprime generando una nueva carpeta llamada 'practica2' y accedemos a dicha carpeta.
4. Existe un fichero '***install.sh***' que instala los paquetes necesarios y compila el código. Luego se ejecuta '***make***' para compilar cada vez que se edite el código.

Hay un elemento adicional que hay que tener en cuenta si se desea instalar este software en un ordenador personal o portátil cuando éste trabaja bajo sistema operativo Linux (esto es aplicable a los Mac también). Es necesario tener instalada la librería '***freeglut3***'.

A pesar de tener instalada la librería y de que no dé errores de compilación, esto no asegura que (la parte gráfica del software) vaya a funcionar perfectamente. Sería largo de explicar la razón, pero aquí incluimos una explicación breve sacada de Wikipedia:

> "*Direct3D y OpenGL son interfaces para la programación de aplicaciones (API acrónimo del inglés Application Programing Interfaces) competitivas que pueden ser usadas en aplicaciones para representar gráficos por computadora (o también llamado computación gráfica) en 2D y 3D, tomando como ventaja de la aceleración por hardware cuando se dispone de esta. Las unidades de procesamiento gráfico (GPU acrónimo del inglés Graphics Processing Unit) modernas, pueden implementar una versión particular de una o ambas de estas interfaces*."

Y de aquí proviene la razón de que no funcione bien del todo. Si la tarjeta gráfica está diseñada específicamente para Direct3D (que es la que usan la mayoría de los juegos de ordenador), trabajará con OpenGL en modo simulación y, a veces, esa simulación no es del todo buena. Un síntoma de esto suelen ser los problemas de refresco de ventanas (no se ven los botones, sólo actualiza una ventana o parte de una ventana y el resto no, …). Este problema no tiene solución evidente.

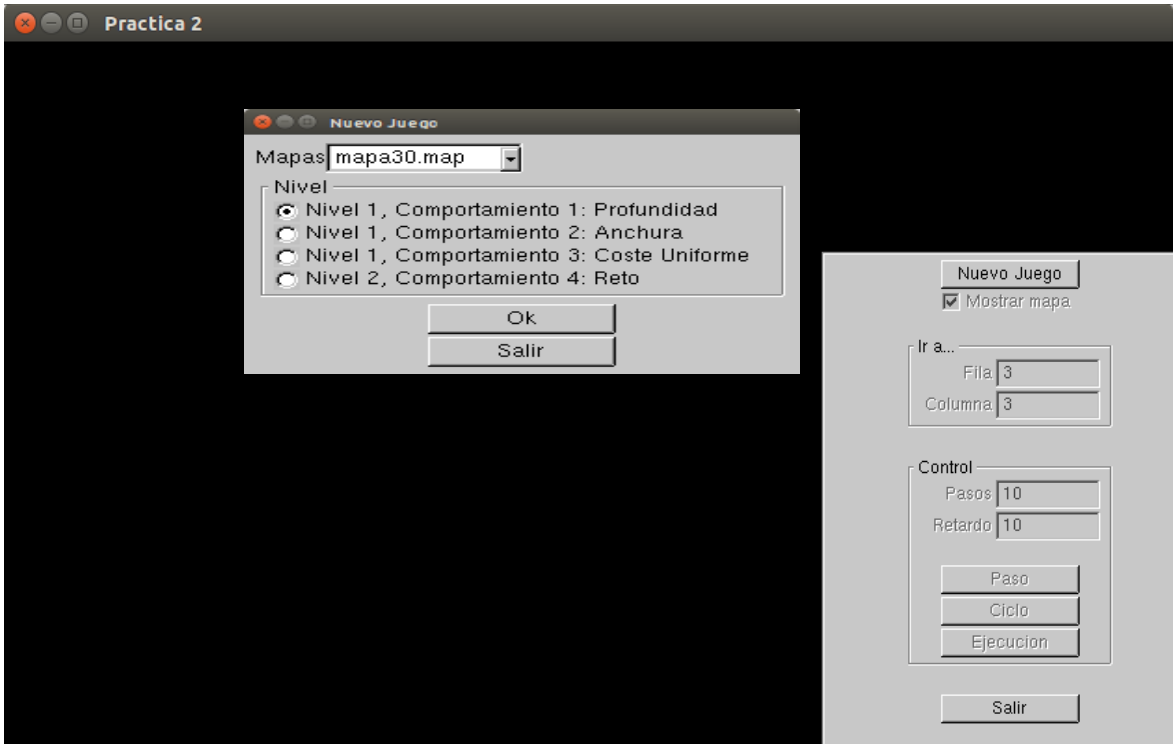
4.2. Funcionamiento del Programa

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

Existen dos ficheros ejecutables: **Belkan** y **BelkanSG**. El primero corresponde al simulador con interfaz gráfica, mientras que el segundo es un simulador en modo *batch* sin interfaz.

4.2.1. Interfaz gráfica

Para ejecutar el simulador con interfaz hay que escribir "**./Belkan**". Se tiene la posibilidad de cambiar la semilla del generador de números seguido de un número. Por



ejemplo, "**./Belkan 1**".

Al arrancar el programa, nos mostrará una ventana como la que se muestra en la figura anterior, que es la ventana principal. Para iniciar el programa se debe elegir el botón "Nuevo Juego" que abriría la siguiente ventana:

En esta nueva ventana se puede elegir el mapa con el cual trabajar (debe estar dentro de la carpeta "mapas") y el nivel deseado. Seleccionaremos el "Nivel 1a" como ejemplo con el mapa "mapa30.map", y presionaremos el botón de "Ok".

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

La ventana principal se actualizará (ver imagen anterior) y podremos entonces distinguir tres partes: la izquierda, que está destinada a dibujar el mapa del mundo; la superior derecha, que mostrará una visión del mapa desde el punto de vista de nuestro personaje; y la inferior derecha, que contiene los botones que controlan el simulador y los valores de los sensores de nuestro personaje.

Los botones 'Paso', 'Ciclo' y 'Ejecución' son las tres variantes que permite el simulador para avanzar en la simulación. 'Paso' invoca al agente que se haya definido y devuelve la siguiente acción. 'Ciclo' realiza varios pasos con un retardo especificado (en décimas de segundo). 'Ejecución' realiza una ejecución completa de la simulación.

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

4.2.2. Sistema *batch*

Para ejecutar el simulador con sin interfaz gráfica hay que escribir "./BelkanSG" seguido de una serie de parámetros:

- fichero de mapa
- semilla para inicializar el generador de números aleatorios
- comportamiento (1, 2, 3, 4)
- fila origen
- columna origen
- pares de (fila, columna) destino

Por ejemplo podemos ejecutar "./BelkanSG mapas/mapa100.map 1 3 4 4 3 3 5 2" lo cual utilizará el mapa indicado con una semilla 1 en el nivel 3 e intentará enviar el jugador desde la posición (4,4) hasta la posición (3,3) y luego la (5,2). En caso de quedarse sin destinos, los elegirá al azar.

Al finalizar la ejecución, el programa nos presentará el tiempo consumido, la cantidad de fallos que hemos cometido al descubrir el mapa, el número de colisiones que hemos sufrido, la cantidad de muertes por caídas a algún precipicio y la cantidad de destinos alcanzados.

4.3. Descripción del Software

De todos los archivos que se proporcionan para compilar el programa, el alumno sólo puede modificar 2 de ellos: ***'jugador.hpp'*** y ***'jugador.cpp'***. Estos archivos contendrán los comportamientos implementados para nuestro agente deliberativo/reactivo. Además, dentro de estos dos archivos, no se puede eliminar nada de lo que hay originalmente, únicamente se puede añadir. Para aclarar esto mejor, pasamos a ver con un poco más de detalle cada uno de estos archivos.

En el archivo ***'jugador.hpp'***, en la parte inferior del código, vemos la declaración de los datos privados. Tiene declaradas las siguientes variables de estado:

- `fil`, `col`: posición en fila y columna donde esta posicionado el jugador. Se inicializa en 99 tanto fil como col.
- `brujula`: orientación del jugador (0 -> norte, 1 -> este, 2 -> sur, 3 -> oeste). Se inicializa en 0.
- `destino`: datos sobre la fila, columna y orientación del objetivo. Se inicializan en -1 todos los campos de su estructura.
- `plan`: lista de acciones que determinan un plan de movimientos. Se inicializa en una lista vacía.

El estudiante puede agregar tantas variables de estado como crea convenientes.

Como ya vimos, existe una variable **mapaResultado** que se incluye a partir del fichero ***'comportamiento.hpp'*** en donde se encuentra el mapa. En los tres primeros niveles (1a, 1b y 1c) esta variable se utilizaría básicamente como si fuera de solo lectura, mientras que en el nivel 2 la matriz viene inicializada con '?' y se debe ir completando a medida que se descubre el mapa (al llenarla se irá automáticamente dibujando en la interfaz gráfica).

En el archivo ***'jugador.cpp'*** se describe el comportamiento del agente. En este archivo podemos añadir tantas funciones nuevas como necesitemos, pero las únicas funciones que se pueden modificar, y sólo para añadir código, son ***'pathFinding'*** y ***'think'***. El método ***'think'*** se dedica a definir las reglas que regirán su comportamiento. Este método implementa el comportamiento del agente y, como se puede observar, devuelve un valor de tipo ***'Action'***. Este es un tipo de dato enumerado declarado en ***'comportamiento.hpp'*** que puede tomar los valores {**actFORWARD**, **actTURN_L**, **actTURN_R**, **actIDLE**}, como ya hemos comentado anteriormente.

5. Método de evaluación y entrega de prácticas

5.1. Método de evaluación

A partir del comportamiento entregado en los ficheros antes mencionados y del nivel elegido, se realizarán varias simulaciones con diferentes mapas (de diferente tamaño, pero con un máximo de 100x100) y con diferentes destinos. El simulador no debe "colgarse" en ningún momento y debe, por supuesto, compilar sin problemas.

Se valorará la cantidad de destinos alcanzados en cada simulación y restará puntos cualquier colisión con objetos estáticos del mapa o la muerte del personaje.

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es

La nota final se calculará de la siguiente manera:

Nota final = Nota Practica * Defensa

donde "Defensa" es un valor en [0,1] indicando en que medida el alumno ha sabido defender los comportamientos implementados en su agente en el proceso de defensa.

- Si se opta por el Nivel 1 se tendrán que completar los comportamientos 2 y 3. En este caso, la nota máxima alcanzable será de 5.
- Si se opta por el Nivel 2, esto implicará haber completado el Nivel 1 completamente y además hacer el comportamiento 4. En este caso, la nota máxima alcanzable será de 10.

5.2. Entrega de prácticas

Se pide desarrollar un programa (modificando el código de los ficheros del simulador '***jugador.cpp***' y '***jugador.hpp***') con el comportamiento requerido para el agente. Estos dos ficheros deberán entregarse mediante la plataforma web de la asignatura, en un fichero ZIP que no contenga carpetas. El archivo ZIP deberá contener sólo el código fuente de estos dos ficheros con la solución del alumno, así como un fichero de documentación en formato PDF que describa el comportamiento implementado con un máximo de 5 páginas.

No se evaluarán aquellas prácticas que contengan ficheros ejecutables o virus

5.3. Fechas Importantes

La fecha fijada para la entrega de las prácticas para cada uno de los grupos será la siguiente:

Grupo de prácticas	Fecha límite entrega	Fecha defensa
TODOS	28 de abril hasta las 23:00 horas	Semana del 6 de Mayo

Nota: Las fechas límite de entrega representan el concepto de hasta esa fecha, por consiguiente, es posible hacer la entrega antes de la fecha arriba indicada.

5.4. Observaciones Finales

Esta **práctica es INDIVIDUAL**. El profesorado, para asegurar la originalidad de cada una de las entregas, someterá a estas a un procedimiento de detección de copias. En el caso de detectar prácticas copiadas, los involucrados (tanto el que se copió como el que se ha dejado copiar) tendrán suspensa la asignatura. Por esta razón, recomendamos que en ningún caso se intercambie código entre los alumnos. No servirá como justificación del parecido entre dos prácticas el argumento "es que la hemos hecho juntos y por eso son tan parecidas", o "como estudiamos juntos, pues...", ya que como se ha dicho antes, **las prácticas son INDIVIDUALES**.

Como se ha comentado previamente, el objetivo de la defensa de prácticas es evaluar la capacidad del alumno para enfrentarse a este problema. Por consiguiente, se asume que todo el código que aparece en su práctica es original y ha sido introducido por alguna razón, de forma que el alumno domina perfectamente el código que entrega. Así, si durante cualquier momento del proceso de defensa el alumno no justifica adecuadamente algo de lo que aparece en su código, la práctica se considerará copiada y tendrá suspensa la asignatura. Por esta razón, aconsejamos no incluir nada en el código que el alumno no sea capaz de explicar qué misión cumple dentro de su práctica y que revise el código con anterioridad a la defensa de prácticas.

Por último, las prácticas presentadas en tiempo y forma, pero no defendidas por el alumno, se considerarán como no entregadas y el alumno obtendrá la calificación de 0. El supuesto anterior se aplica a aquellas prácticas no involucradas en un proceso de copia. En este último caso, el alumno tendrá suspensa la asignatura.

E.T.S. de Ingenierías Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada - Tlf. +34 958 244019 - http://decsai.ugr.es