UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

/ UGR / decsai

INTELIGENCIA ARTIFICIAL

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

Práctica 2

Agentes Reactivos/Deliberativos

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
UNIVERSIDAD DE GRANADA
Curso 2024-2025

1. Introducción

La segunda práctica de la asignatura ***Inteligencia Artificial*** consiste en el diseño e implementación de un agente reactivo/deliberativo. Se trabajará con un simulador software adaptado para incluir algoritmos de búsqueda. Nos centraremos en implementar el comportamiento de un "personaje virtual". Utilizaremos las técnicas estudiadas en los temas 2 y 3 de la asignatura para el diseño de agentes reactivos y deliberativos.

2. Descripción

En esta práctica tomamos como punto de partida el mundo de las aventuras gráficas de los juegos de ordenador para intentar construir sobre él personajes virtuales que manifiesten comportamientos propios e inteligentes dentro del juego. Intentamos situarnos en un problema habitual en el desarrollo de juegos para ordenador y vamos a jugar a diseñar personajes que interactúen de forma autónoma usando agentes reactivos/deliberativos.

2.1. La historia

Nos situamos en el mundo del rescate de excursionistas que se accidentan en la naturaleza en terrenos con una orografía complicada. En concreto, trataremos de dar vida y comportamientos inteligentes a un par de agentes (Rescatador y Auxiliar) que en este mundo virtual e imaginario formarán el equipo de rescate. Ellos serán los encargados de atender a las llamadas de socorro que se reciban. Entre los cometidos de este equipo estará verificar el correcto estado de los caminos y senderos de la zona de trabajo, así como de proceder a la búsqueda y rescate de los potenciales accidentados que visiten la zona. Denominaremos "juego" o "simulación" a las acciones que realiza este grupo virtual de rescate entre un instante de tiempo inicial y otro final.

Antes de describir al equipo y al resto de participantes en el juego, pasaremos a definir los elementos que forman parte del escenario.

2.2. 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

un terreno que serán considerados como inmutables, es decir, los elementos dentro del mapa que no cambian durante el desarrollo del juego.

El mapa queda representado con una matriz donde la primera componente representa la fila y la segunda representa la columna. Como ejemplo usaremos un mapa de tamaño 100x100 de caracteres. Fijamos 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].

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 este convenio, 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 del terreno son los siguientes:

- ***Bosque***, codificado con el carácter '**B**' y se representan en el mapa como casillas de color verde claro. En general son casillas no transitables, aunque para algún agente y bajo ciertas circunstancias sí que podrían hacerse transitables.

- ***Matorral***, codificado con el carácter '**T**' y se representan en el mapa como casillas de color verde oscuro. Representa un terreno abrupto donde se hace complicado el caminar. Se puede transitar, pero es costoso avanzar.
- ***Agua***, codificado con el carácter '**A**' y tiene asociado el color azul. Es un obstáculo transitable para algunos agentes e intransitable para otros. Para los agentes que es transitable, pasar por ellas les conlleva gastar mucha energía.
- ***Precipicios***, codificado con el carácter '**P**' y tiene asociado el color negro. Estas casillas se consideran intransitables y caer en ella supone el final de la simulación.
- ***Camino***, codificado con el carácter '**C**' y tiene asociado el color marrón. Son los lugares donde se hace más fácil caminar. Normalmente, los caminos permiten conectar los distintos puestos base que haya en el mapa.
- ***Senderos***, codificado con el carácter '**S**' y tiene asociado el color gris. Son también sitios transitables donde el andar es fácil, aunque consumiendo más energía que en los caminos.
- ***Obstáculos***, codificados con el carácter '**M**' y son rojo oscuro. Representan los obstáculos compuestos por antiguas construcciones que hay en el lugar o bien por piedras grandes que interrumpen el paso de los agentes. No son lugares transitables.
- ***Zapatillas***, codificadas con el carácter '**D**' y son moradas. Cuando un agente pasa por esta casilla adquiere una mejora de sus habilidades. La mejora depende del tipo de agente.
- ***Puesto Base***, codificado con el carácter **'X'** y de color rosa.
- ***Casilla aún desconocida*** se codifica con el carácter **'?'** y se muestra en blanco (representa la parte del mundo que aún no has explorado).

Todos los mundos que usaremos en esta práctica son cerrados, ya que en todos se verifica 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 pueda haber más precipicios en el resto del mapa.

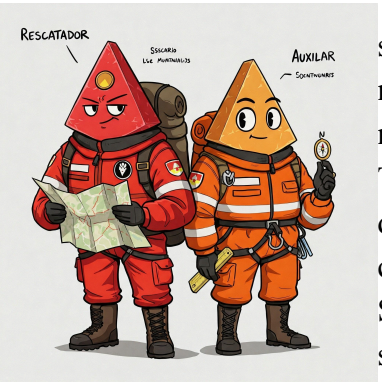
También hay que indicar que cada casilla del mapa se encuentra a una determinada altura o cota. La altura corresponde a un número entero entre 0 y 5. El valor 0 denota la cota más baja posible y las casillas que la tengan serán consideradas intransitables para los agentes y

se comportan como si fueran precipicios, es decir, caer en ella supone el final de la simulación. Es importante indicar que la diferencia de altura entre casillas adyacentes influye en la posible movilidad de los agentes, así como en el coste asociado a ese movimiento. La regla general que rige la posibilidad de movimiento es que solo puedo acceder a aquellas casillas que tengan un valor de altura con un diferencial máximo de 1 en valor absoluto, es decir, a casillas con la misma altura que la original o a lo sumo un valor de una unidad más o menos de altura.

¿Qué ocurre si se intenta acceder a una casilla que no respete esta regla? Si el movimiento se hace hacia una casilla que tiene más altura, se producirá un choque, es decir, el agente se quedará en la casilla origen del movimiento sin haberlo realizado. Si el movimiento es hacía una casilla de menor valor de altura, se comportará como si hubiera caído a un precipicio, es decir, la simulación terminará.

2.3. Los agentes que se mueven sobre el escenario

Sobre este mapa pueden existir algunos agentes que tienen la capacidad de moverse por sí mismos. Los agentes que aquí consideraremos son:



- ***Rescatador***, se codifica con el carácter '**r**' y se muestra como un triángulo rojo. Este es el miembro más ágil del equipo de rescate y normalmente es el primero en llegar al accidentado. Tiene la posibilidad de correr. Además, la posesión de las ***Zapatillas*** le permite poder avanzar a casillas con una diferencia de altura de 2 en valor absoluto. Sólo habrá un Rescatador sobre el mapa en cada simulación.
- ***Auxiliar*** se codifica con el carácter **'a'** y se muestra como un triángulo naranja en el mapa general. Es el otro miembro del equipo de rescate. Es menos ágil que el Rescatador y es usado por este como ayuda en algunas situaciones de rescate, sobre todo cuando necesita ayuda para trasladar al accidentado. Las ***Zapatillas*** hacen que las casillas tipo árbol no sean un obstáculo para él. Cuando tiene las ***Zapatillas,*** las

casillas tipo 'B' se comportan a todos los efectos como casillas tipo 'C', es decir, matorral.

- ***Excursionistas***, se codifica con el carácter '**e**' y se muestra como un cuadrado naranja. Son otros agentes que están por el lugar disfrutando de la naturaleza. Se desplazan de allí para acá por el mapa. Hay que estar atentos con ellos porque en algunos casos podrían molestar las labores de rescate. Son molestos porque son elementos con los que el Rescatador o el Auxiliar pueden colisionar.

- Vándalos, se codifican con el carácter **'v'** y se muestran como un hexágono rosa. Se hacen pasar por excursionistas, pero en realidad tienen mal carácter. En un principio se comportan como el resto de los montañeros, pero en situaciones especiales pueden sacar un carácter agresivo. En lugar de ayudar en el rescate, se dedican a retardar todo el proceso obstaculizando los movimientos del Rescatador y del Auxiliar y llegando en ocasiones hasta incluso poder empujarlos. Empujar a un Rescatador consiste en desplazarlo a una casilla en la dirección en la que avanzaría el vándalo. Para que un vándalo pueda empujar al agente es necesario que esté en una casilla adyacente a la del agente y además encarado hacia él y también la casilla donde se terminaría desplazando el Rescatador tiene que ser transitable.

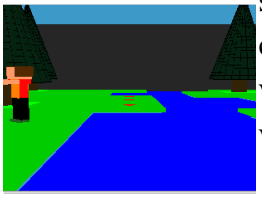
2.3. El agente *Rescatador*

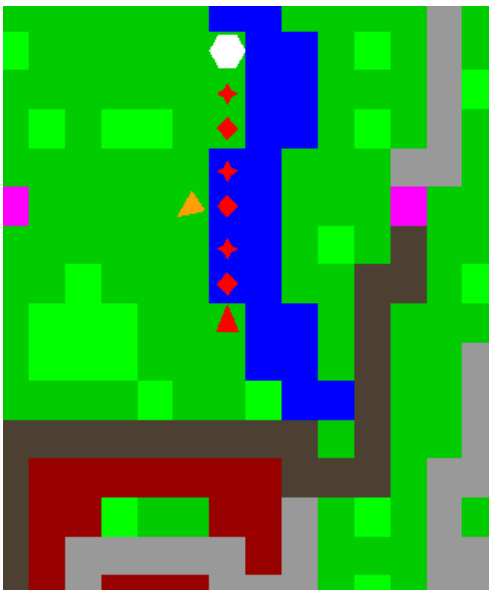
Durante el desarrollo de esta práctica tendremos que definir comportamientos para el que llamaremos agente ***Rescatador.*** Entre los objetivos de este juego estará realizar comportamientos reactivos como reconocer todos los caminos que hay en su área de trabajo o comportamientos deliberativos que impliquen desplazar a una parte o a la

totalidad del equipo de rescate desde la posición en la que se encuentre en ese momento (origen) a la posición donde se encuentre un accidentado (destino).

2.3.1. Sistema Sensorial del agente *Rescatador*

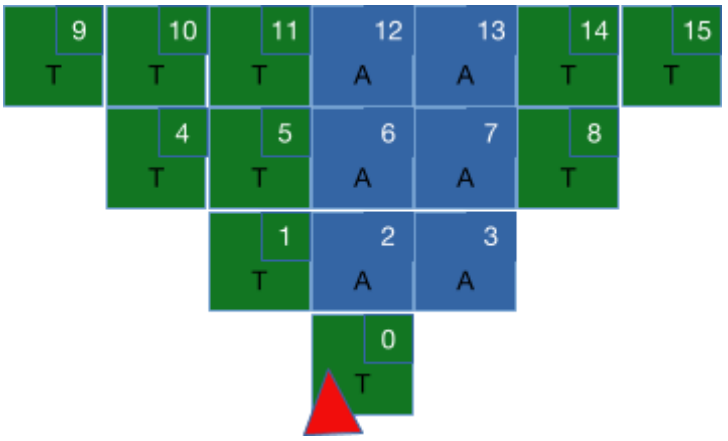
El ***Rescatador*** en el simulador viene representado en forma de un triángulo rojo. Cuenta con un sistema visual que le permite observar las cosas que se le aparecen dentro de un cono de visión. Dicho cono de visión se representa usando tres vectores de tamaño 16 de tipo *"unsigned char"*. El primero de ellos, lo denominaremos **superficie** y es capaz de observar los elementos inmóviles del mapa. El segundo de ellos, que llamaremos **agentes**, es capaz de mostrarnos qué otros agentes se encuentran en nuestro cono de visión (es decir, al Auxiliar, a otros excursionistas o vándalos). El tercero y último de ellos lo denominaremos **cota**, y nos muestra la altura a la que se encuentra cada parte del terreno que estamos observando.





Para entender cómo funciona este sistema pondremos el siguiente ejemplo:

En este caso, el sensor superficie contiene **TTAATTAATTTTAATT** y su representación real sobre un plano será la siguiente: El primer carácter (posición 0) representa el tipo de accidente geográfico sobre el que se encuentra el agente. El tercer carácter (posición 2) es justo el tipo de terreno que tengo justo delante de él. Los caracteres de las posiciones 4, 5, 6, 7 y 8 representan lo que está delante de mí, 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 viéndolos de izquierda a derecha. La figura representa las posiciones del vector en su distribución bidimensional (los números), y el carácter y su representación por colores cómo quedaría en un mapa.

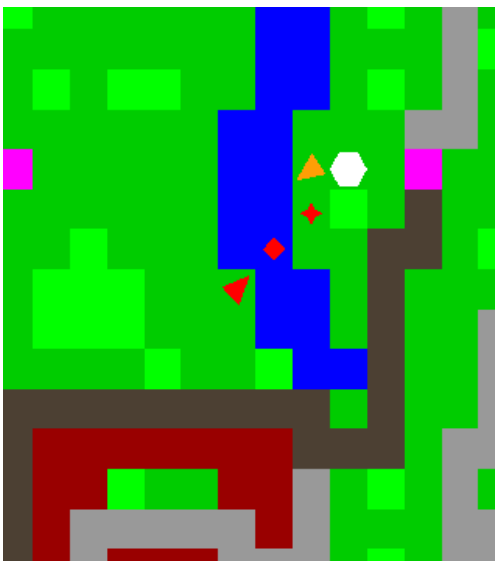


De igual manera se estructura el vector **agentes**, pero, en este caso, indicando que agentes se encuentran en cada una de esas casillas. Los valores posibles en este sensor serán: **'a'** para indicar que hay un Auxiliar, **'r'** para indicar que hay un Rescatador, **'v'** para indicar la posición de un vándalo, **'e'** para indicar la posición de un excursionista y **'_'** si la casilla está desocupada.

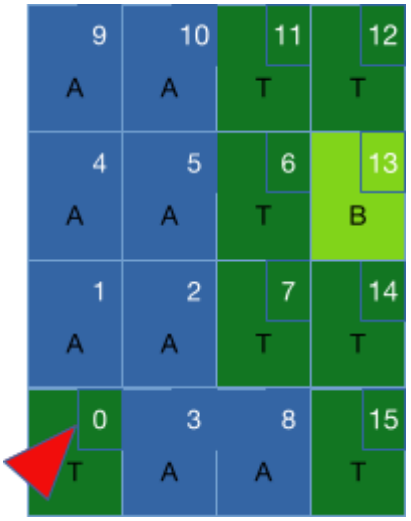
En el caso del vector **cota**, aparecerán los valores de la altura a la que se encuentran las casillas del cono de visión y la estructura e interpretación del sensor/vector es igual a la de los vectores anteriores.

La interpretación de los sensores con respecto

al mapa real es la anterior cuando el agente se encuentra orientado en las posiciones **norte**, **este**, **sur** y **oeste**.



Cuando su orientación es **noreste**, **sureste**, **suroeste** y **noroeste** la representación cambia ligeramente, de manera que, si el vector **superficie** contiene **TAAAAATTAAATTBTT**, su representación real sobre un plano será la siguiente:



El agente cuenta además con los sensores siguientes:

- **Sensor de choque (`choque`)**: Es una variable booleana. Tomará el valor verdadero cuando tras intentar realizar una acción de avance, dicho avance no se haya podido realizar y el agente se ha quedado en la casilla que se encontraba antes de realizar el movimiento. Este sensor se puede activar cuando se produce una colisión con otro agente, con un árbol, con un obstáculo o cuando la casilla a la que intentaba acceder tiene una diferencia positiva de altura superior a 1 que le impide realizar el movimiento.
- **Sensor de vida (`reset`)**: Es una variable booleana que toma el valor de verdad si el agente realiza una acción de avance no permitida (normalmente es que se cae por un precipicio).
- **Sensores de posicionamiento (`posF`, `posC`)**: Devuelve la posición de fila y columna que ocupa el agente Rescatador.

- **Sensor de orientación (`rumbo`)**: Devuelve la orientación del agente Rescatador de entre las 8 posibles.
- **Sensor de la ubicación destino** (**`destinoF`, `destinoC`**): Mantiene la información de la casilla donde se encuentra el accidentado. En la primera se describe la coordenada de la fila y en la segunda la coordenada de la columna.
- **Sensor de energía (`energia`)**: Informa del nivel de energía del agente. Inicialmente, tiene un valor de 3000 puntos y dependiendo de la acción realizada va perdiendo valor. La simulación termina cuando el nivel de energía llega al valor 0.
- **Sensor de nivel (`nivel`):** Este es un sensor que informa en qué nivel del juego se encuentra. Los valores posibles del sensor están entre 0 y 4 y están asociados con

  Nivel 0: Despertar Reactivo

  Nivel 1: La Cartografía de los Desconocido

  Nivel 2: La Búsqueda del Camino de Dijkstra

  Nivel 3: La Ascensión del A*uxiliar

  Nivel 4: Misión de Rescate
- **Sensor de tiempo consumido (`tiempo`):** Este sensor informa de los instantes de simulación que se llevan consumidos.
- **Sensor de asistencia (`gravedad`):** Este sensor indica al Rescatador si requiere del Auxiliar para el traslado del accidentado. Es booleano y solo se activa cuando el Rescatador llega a la casilla del accidentado.

2.3.2. Sistema Sensorial del agente *Auxiliar*

El agente ***Auxiliar*** tiene definido el mismo sistema sensorial que el agente Rescatador salvo el sensor de asistencia. Es decir, tiene el mismo sistema visual conformado por los

tres sensores/vectores **superficie**, **agentes** y **cota**, tiene los sensores de colisión, reset, posicionamiento, rumbo, destino, energía, nivel y tiempo.

Por su parte, el Auxiliar tiene un sensor propio que no tiene el Rescatador:

- **Sensor de invocación (`venpaca`):** Este sensor se activa por una orden del agente Rescatador. Suele usarse para indicar al Auxiliar que el Rescatador necesita de su asistencia para atender a un accidentado.

Algo importante a tener en cuenta es que el sensor destino funciona algo diferente de cómo funciona en el agente Rescatador. Mientras el primero tiene por defecto las coordenadas de donde debe ir aportadas por el sistema, el agente Auxiliar no tiene acceso a esa información a menos que el agente Rescatador pida su asistencia. Es decir, que el valor de destino por defecto que tiene este agente no tiene por qué coincidir con el de las coordenadas del accidentado.

2.3.3. Datos de los agentes compartidos con el entorno

Entre los datos a los que tienen acceso los agentes se encuentran dos matrices llamadas **`mapaResultado`** y **`mapaCotas`** que le permite interactuar con el mapa. **`mapaResultado`** representa la parte visible del mapa y cualquier modificación que se haga sobre él se verá reflejada en la interfaz gráfica. Por otro lado, **`mapaCotas`** informa sobre la altura (cota) de cada casilla del mapa.

2.3.4. Acciones que puede realizar los agentes

Acciones del Agente Rescatador

- **`WALK`**: le permite al Rescatador avanzar a la siguiente casilla del mapa siguiendo su orientación actual. Para que la operación finalice con éxito es necesario que la casilla de destino sea transitable para él y la diferencia de altura entre ambas casillas sea inferior a 2 si no tiene las zapatillas e inferior a 3 si está en posesión de las zapatillas.
- **`RUN`**: el agente Rescatador avanza dos casillas siguiendo su orientación actual. Para que la acción finalice con éxito es necesario que la casilla final sea transitable y la diferencia de altura entre ambas casillas sea inferior a 2 si no

tiene las zapatillas e inferior a 3 si está en posesión de las zapatillas. La casilla intermedia debe ser transitable, no estar ocupada por otro agente.

- **`TURN_L`**: le permite mantenerse en la misma casilla y girar a la izquierda 90º teniendo en cuenta su orientación.
- **`TURN_SR`**: le permite mantenerse en la misma casilla y girar a la derecha 45º teniendo en cuenta su orientación.
- **`CALL_ON`**: activa el sensor de invocación al agente Auxiliar. Además, coloca en el sensor de destino del agente Auxiliar las coordenadas donde se encuentra el accidentado.
- **`CALL_OFF`**: desactiva el sensor de invocación al agente Auxiliar.
- **`IDLE`**: no hace nada

Acciones del Agente Auxiliar

- **`WALK`**: le permite al Auxiliar avanzar a la siguiente casilla del mapa siguiendo su orientación actual. Para que la acción finalice con éxito es necesario que la casilla de destino sea transitable para el agente Auxiliar.
- **`TURN_SR`**: le permite al agente permanecer en la misma casilla, pero que cambie su orientación 45º a la derecha.
- **`IDLE`**: no hace nada.

Dinámica de aplicación de las acciones en el simulador.

Es importante conocer que la secuencia en la que el simulador aplica las acciones de los diferentes agentes involucrados es siempre la misma y su funcionamiento es el siguiente: antes de empezar el siguiente instante de simulación se actualiza el sistema sensorial de cada agente. Una vez hecho eso, el primero en decidir y aplicar su acción será el agente Rescatador, después el agente Auxiliar y después lo harán el resto de los agentes (excursionistas y vándalos).

2.3.5. Coste de las acciones

Cada acción realizada por el agente tiene un coste en tiempo de simulación y en consumo de energía. En cuanto al tiempo de simulación, todas las acciones consumen un instante independientemente de la acción que se realice y del terreno donde se encuentre el agente. En cuanto al consumo de energía, hay que decir que:

- Cada agente tiene su propio nivel de energía que va consumiendo conforme va ejecutando esas acciones.
- Las acciones `IDLE`, `CALL_ON` y `CALL_OFF` consumen **0** de energía.
- ***El consumo asociado al resto de las acciones depende del tipo de terreno a donde el agente inició la acción y de la altura relativa entre casillas destino y fin***. Así en las acciones `TURN_L`, `TURN_SR`, `WALK` y `RUN` dependen de la casilla de inicio del agente. Esto es, si el agente Rescatador está en una casilla de agua (etiquetada con una 'A') y avanza a una casilla de matorral (etiquetada con una 'T'), el coste en energía es el asociado a la casilla de agua, es decir, a la casilla inicial donde se produce la acción de avanzar.

En las siguientes tablas se muestran los valores de consumo de energía en función de la acción a realizar, la casilla inicial de la acción y el incremento o decremento de energía en función del diferencial de altura entre casilla inicial y la final.

WALK		
Casilla Inicial	**Gasto energía**	**Incr/Decr por cambio de altura**
'A'	100	10
'T'	20	5
'S'	2	1
Resto de Casillas	1	0

RUN (Rescatador)		
Casilla Inicial	**Gasto Normal energía**	**Incr/Decr por cambio de altura**
'A'	150	15
'T'	35	5
'S'	3	2
Resto de Casillas	1	0

TURN_L (Rescatador)		
Casilla Inicial	**Gasto Normal energía**	**Incr/Decr por cambio de altura**
'A'	30	0
'T'	5	0
'S'	1	0
Resto de Casillas	1	0

TURN_SR		
Casilla Inicial	**Gasto Normal energía**	**Incr/Decr por cambio de altura**
'A'	16	0
'T'	3	0
'S'	1	0
Resto de Casillas	1	0

La columna de incremento/decremento de energía se aplica solo cuando hay diferencia de altura entre la casilla origen y la casilla destino de la acción, por esa razón, no tienen aplicación de incremento las acciones de giro.

Se aplica un incremento positivo en el consumo de energía en aquellas acciones de movimiento que implican que el agente termine en una casilla como una altura superior a la de la casilla origen. El incremento será negativo, es decir, se restará al consumo base de energía, cuando el agente termine en una casilla con una altura asociada inferior a la de partida.

Ilustraremos el uso de la tabla anterior con algunos ejemplos para aclarar bien cómo funciona el consumo de energía:

- Si el agente Rescatador aplica una operación **WALK** en una casilla de Agua y de altura 3 para llevarle a una casilla de altura 4, el valor de energía asociado a esa acción sería 100 por partir del agua. Tendría un incremento de 10 por estar la casilla destino más alta que la casilla origen. El consumo total sería de 110 puntos.
- Si el agente Rescatador está en una casilla de matorral con altura 1 y aplica una acción de **RUN.** La casilla destino está a una altura de 3, la casilla intermedia tiene una altura de 1 y está en posesión de las Zapatillas, entonces el consumo será de 35 por ser el valor estándar de moverse por el matorral más un incremento de 5 por estar la casilla destino más alta que la inicial. Por consiguiente, el consumo total es de 35 puntos. Es importante observar que para que esta acción se pueda realizar es necesario estar en posesión de las Zapatillas, ya que en otro caso sería imposible acceder a una casilla con diferencial de +2 en la altura. Además, que la altura de la

casilla intermedia no sea superior a la casilla inicial. También se puede observar que la diferencia de altura sea de más de 1 no afecta en cuanto al incremento aplicado. Lo único a considerar para restar o sumar la cantidad adicional es si es hacia arriba o hacia abajo. Hacía arriba se incrementa, hacia abajo se decrementa.

- El agente Auxiliar se encuentra en una casilla de sendero con altura 4 y aplicando una acción **WALK** pasa a una casilla de altura 3 el consumo será de 3 por la acción. Se le aplica un decremento de 2 por el cambio de altura, y, por consiguiente, el consumo final será de 1 punto de energía.

3. Objetivo de la práctica

El objetivo de esta práctica es dotar de un comportamiento inteligente a los dos agentes del equipo de rescate que les permitan completar los distintos niveles que se les proponen. Los niveles han sido diseñados para que vayan incrementando en dificultad, empezando por un nivel 0 que está muy dirigido para ayudar al estudiante a iniciar la práctica.

El objetivo de esta práctica no es simplemente programar, sino insuflar vida a dos almas digitales: Rescatador y Auxiliar. Juntos, deberán navegar por un laberinto de niveles, cada uno un capítulo de una saga épica, donde los peligros acechan en cada esquina.

Nivel 0: [R] El Despertar Reactivo

El objetivo de este nivel consiste en hacer llegar a los dos agentes, Rescatador y Auxiliar a casillas de puesto base. Las condiciones en las que se desarrolla este nivel son las siguientes:

- El mapa es desconocido por los dos agentes.

- Ambos agentes parten en su posición inicial de una casilla de tipo camino.
- Para conseguir el objetivo, los agentes tienen obligatoriamente que pasar exclusivamente por casillas de tipo camino.
- El comportamiento que se defina debe ser reactivo.
- Cada agente debe llegar a un puesto base distinto.

En el Nivel 0, los agentes darán sus primeros pasos, aprendiendo a sentir el pulso del entorno y a reaccionar con presteza. Será como enseñar a un recién nacido a gatear, un primer contacto con el mundo digital.

Nivel 1: [R] La Cartografía de lo Desconocido

La misión que tienen los agentes en este nivel es intentar descubrir la mayor cantidad posible de casillas de tipo camino y sendero que se encuentran en el territorio. Las condiciones en las que se desarrolla este nivel serán las siguientes:

- El mapa es desconocido por los dos agentes.
- Ambos agentes parten en su posición inicial de una casilla de tipo camino o sendero.
- El comportamiento que se defina debe ser reactivo.
- Cada parte del mapa que vayan reconociendo debe ser almacenado en la variable global **`mapaResultado`**.
- Para evaluar el grado en el que se ha conseguido el objetivo, se comparará el contenido de **`mapaResultado`** con el mapa real. El grado de coincidencia de ambos mapas sobre las casillas de camino y sendero establecerá el grado de cumplimiento del objetivo.

El Nivel 1 los llevará a explorar tierras inexploradas, donde cada casilla es un misterio y cada sendero una incógnita. Deberán desplegar sus sentidos digitales, construyendo un mapa mental del territorio, como antiguos cartógrafos que dibujaban los confines del mundo.

Nivel 2: [D] La Búsqueda del Camino de Dijkstra

En este nivel se persigue encontrar el recorrido que tiene que realizar el agente Rescatador para que partiendo de la casilla inicial pueda alcanzar la casilla objetivo. Las condiciones que se consideran en este nivel son las siguientes:

- El mapa es conocido.
- La casilla inicial y la casilla final puede ser cualquiera de las transitables.
- El recorrido encontrado debe verificar ser el que **menor consumo de energía** le implique al agente Rescatador y se debe obtener aplicando el **algoritmo de Dijkstra** (costo uniforme).
- Hay que recordar que la diferencia de altura entre casillas altera ligeramente el valor del consumo de energía, así como que usar Zapatillas podría permitir acceder por casillas con un diferencial de altura mayor.
- El agente Auxiliar solo tiene la tarea de no entorpecer al Rescatador, así que deberá apartarse si es que puede llegar a ser un obstáculo para él. El consumo de energía que haga el Auxiliar no es relevante para la consecución de este nivel.

En el Nivel 2, el Rescatador se enfrentará al desafío de encontrar la ruta más segura, un laberinto de decisiones donde cada paso consume energía y cada desvío puede ser fatal. Deberá dominar el arte de la búsqueda, invocando el poder del algoritmo de Dijkstra.

Nivel 3: [D] La Ascensión del A*uxiliar

A diferencia del nivel anterior, en este es el agente Auxiliar el que debe llegar a la casilla objetivo. Las condiciones que se deben considerar para plantear la solución a este nivel son las siguientes:

- El mapa es conocido.
- La casilla inicial y la casilla final puede ser cualquiera de las transitables. Al igual pasa con las casillas que formen parte del recorrido.
- El recorrido encontrado debe verificar ser el que **menor consumo de energía** le implique al agente Auxiliar y se debe obtener aplicando el **algoritmo A***.
- Hay que recordar que usar las Zapatillas hace que las casillas de tipo árbol sean transitables para el agente Auxiliar.
- El agente Rescatador solo tiene la tarea de no entorpecer al Auxiliar, así que deberá apartarse si es que puede llegar a ser un obstáculo para él. El consumo de energía que haga el Rescatador no es relevante para la consecución de este nivel.

Nivel 3, el Auxiliar tomará el relevo, ascendiendo por senderos escarpados y superando obstáculos insalvables. El algoritmo A* será su brújula, guiándolo a través de la oscuridad hacia la luz de la meta.

Nivel 4: Misión de rescate

El objetivo de este nivel es evaluar cómo de preparado está el equipo de rescate para atender a excursionistas accidentados en situaciones donde no conocen bien la zona. Para ello se irán recibiendo llamadas de auxilio para que el equipo las vaya atendiendo. La forma de rescate será que primero se acercará el Rescatador hacía la casilla destino, una vez allí determinará la gravedad del accidentado. Si no es grave, la misión se habrá

completado y se obtendrán dos puntos. Si es grave, el Rescatador avisará al Auxiliar para que también vaya a atender al accidentado. Cuando llegue, la misión se habrá cumplido y se obtendrá una valoración de 7. El equipo de rescate tratará de maximizar dicha puntuación.

Las condiciones en las que se desarrolla este nivel son:

- El mapa empieza siendo desconocido para ambos agentes.
- La casilla inicial y la casilla final pueden ser cualquiera de las transitables.
- En el sensor de destino (**`destinoF`**, **`destinoC`**) del agente Rescatador se encuentran las coordenadas del accidentado.
- En el sensor de destino del agente Auxiliar aparecerán los valores **`(-1,-1)`** indicando que no se conoce la posición del accidentado. Si se desea que el Auxiliar conozca dicha localización, el agente Rescatador debe hacer un **`CALL_ON`**.
- Cuando el Rescatador llega a la casilla del accidentado, el sensor **`gravedad`** puede activarse:

(a) **No se activa**, entonces la misión se considera terminada y se incrementa la puntuación del equipo de rescate en 2.

(b) **Sí se activa**, entonces para completar la misión es necesario que el auxiliar acuda para ayudar a socorrer al accidentado. Aquí se tienen dos opciones:

- No aceptar la misión, en cuyo caso el agente Rescatador debe ejecutar la acción **`CALL_OFF`**. A partir de ese momento la misión se considera terminada y se incrementará en 1 la puntuación del equipo de rescate.
- La otra opción es aceptar la misión, en cuyo caso el Rescatador esperará en la casilla del accidentado hasta que el Auxiliar esté en una casilla en la que en su línea de visión se encuentre la casilla objetivo. Cuando esto último ocurra, la misión se considerará terminada y la puntuación del equipo de rescate se incrementará en 7 puntos.

- Cada vez que se cumpla una misión se considere terminada, una nueva casilla de accidentado aparecerá en el sensor de destino del Rescatador. Esta nueva casilla objetivo será desconocida para el agente Auxiliar.

- La simulación terminará cuando se produzca una de las siguientes condiciones:
  (a) Se agotaron los ciclos de simulación que inicialmente son 3000.
  (b) Se agotó la energía de alguno de los dos agentes que inicialmente también son 3000.
  (c) Se agotó el tiempo máximo de simulación que son 300 segundos. Este tiempo es el consumido por los dos agentes para elaborar sus planes y su toma de decisiones a lo largo de la simulación.
  (d) Alguno de los dos agentes cae al vacío (bien por un precipicio o por acceder a una casilla que tiene un desnivel mayor al que pueden tolerar).

- En el territorio puede haber excursionistas y vándalos que dificultarán las tareas del equipo de rescate para acceder a la posición del accidentado.
- Los algoritmos de búsqueda que se utilicen en este nivel pueden ser los realizados en los niveles anteriores. También pueden definirse algoritmos de búsqueda distintos a los anteriores si el estudiante lo considera así.

Debido al desconocimiento inicial del mapa y a que intervienen otros agentes en el mundo, parece que para obtener una solución satisfactoria de este nivel será necesario definir comportamientos reactivos y deliberativos.

El Nivel 4, la prueba final, los agentes deberán unir sus fuerzas para salvar a los excursionistas perdidos en la montaña. Cada llamada de auxilio será un grito en la noche, y cada rescate un acto de valentía. Deberán combinar la rapidez de la reacción con la sabiduría de la planificación, demostrando que son dignos del título de héroes digitales.

4. El Software

Para la realización de la práctica se proporciona al estudiante una implementación tanto del entorno simulado del mundo en donde se mueven los agentes como de la estructura básica de los agentes reactivo/deliberativos.

4.1. Instalación

Se proporciona sólo versión para el sistema operativo Linux y se puede encontrar en https://github.com/ugr-ccia-IA/practica2. La versión del software está preparada para ser usada para la distribución de UBUNTU, aunque es fácil de adaptar para cualquier otra distribución con pequeños cambios. En la versión proporcionada se incluye un archivo 'install.sh' para cargar las librerías necesarias y compilar el programa. Para otras distribuciones de linux es necesario cambiar lo que respecta al comando que instala paquetes y a cómo se llama ese paquete dentro en esa distribución. La lista de bibliotecas necesarias es: ***freeglut3 freeglut3-dev libjpeg-dev libopenmi-dev openmpi-bin openmpi-doc libxmu-dev libxi-dev cmake libboost-all-dev***.

El proceso de instalación es muy simple y consiste en seguir las instrucciones que se proporcionan en el repositorio de GitHub (https://github.com/ugr-ccia-IA/practica2) para acceder y trabajar con el software. Leer detenidamente las instrucciones que se proporcionan en ese repositorio para conseguir una correcta configuración del software.

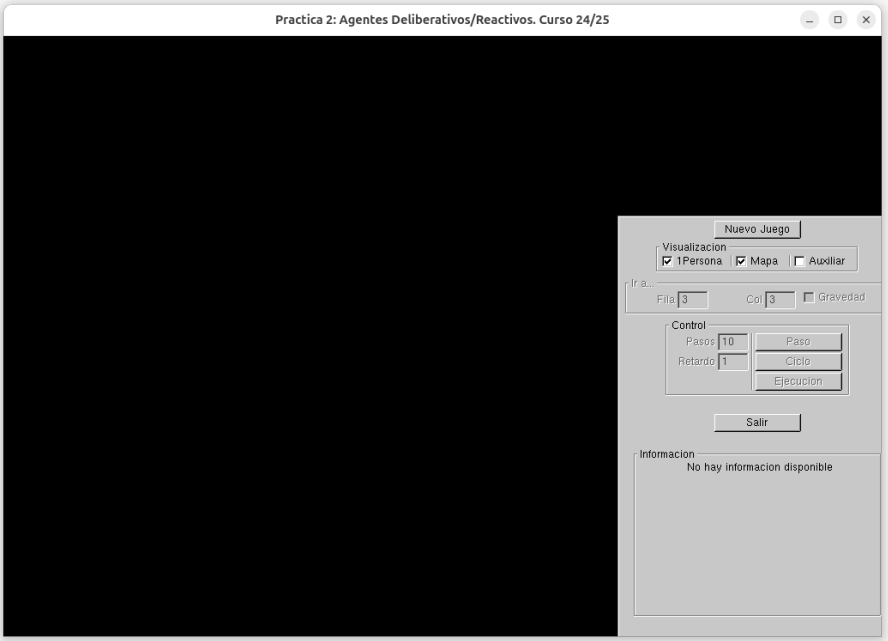
4.2. Funcionamiento del Programa

Existen dos ficheros ejecutables: practica2 y practica2SG. El primero corresponde al simulador con interfaz gráfica, mientras que el segundo es un simulador en modo *batch* sin interfaz. La segunda versión sin entorno gráfico se ofrece para poder hacer tareas de "debugger" o de depuración de errores.

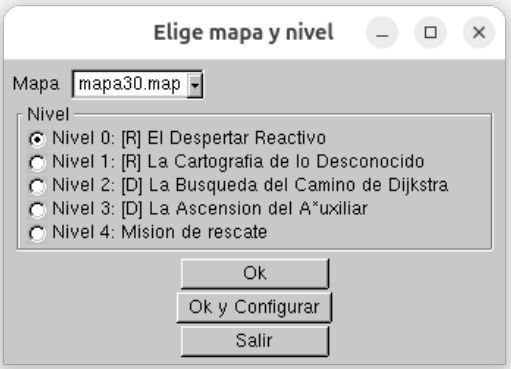
Empezamos describiendo la versión con entorno gráfico.

4.2.1. Interfaz gráfica

Para ejecutar el simulador con interfaz hay que escribir "***./practica2***".

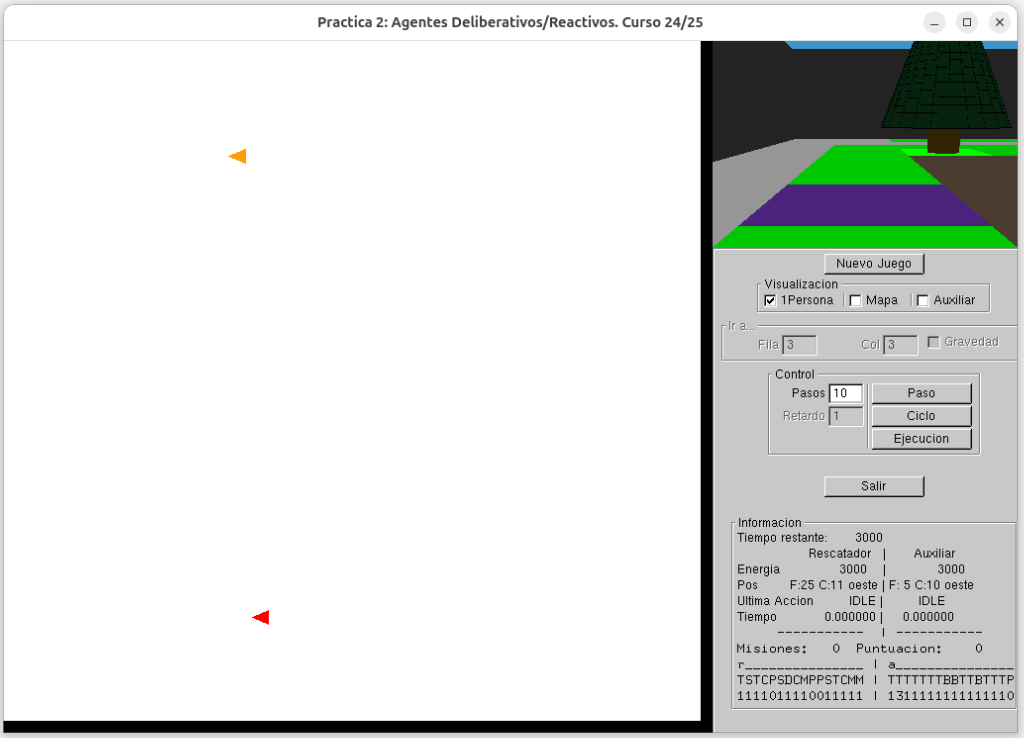


Al arrancar el programa nos mostrará una ventana que es la ventana principal. Para iniciar el programa se debe elegir el botón **Nuevo Juego** que abriría la siguiente ventana:



En esta nueva ventana se puede elegir el mapa con el cual trabajar (debe estar dentro de la carpeta "mapas") y el nivel deseado. El objetivo es ir poco a poco ofreciendo en el software la funcionalidad que se propone en cada nivel.

Seleccionaremos el **Nivel 0: El despertar Reactivo** fijando como mapa **mapa30.map**, y presionaremos el botón de **Ok**.



La ventana principal se actualizará y podremos entonces distinguir tres partes: la izquierda está destinada a dibujar el mapa del mundo, la superior derecha mostrará una visión del mundo desde el punto de vista de uno de los agentes, y la inferior derecha contiene los botones que controlan el simulador e información sobre los valores de los sensores.

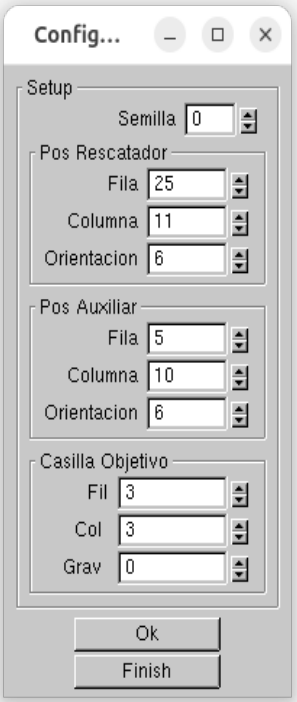
Los botones **Paso**, **Ciclo** y **Ejecucion** son las tres variantes que permite el simulador para avanzar en la simulación. El primero de ellos, **Paso**, invoca al agente que se haya definido y devuelve la siguiente acción. El botón **Ciclo** realiza el número que se indica en el campo **Pasos**. Por último, el botón **Ejecucion** realiza una ejecución completa de la simulación. Se puede detener la ejecución completa, si se pulsa el botón **Paso**.

El último botón que podemos encontrar es **Salir** que cierra la aplicación.

Dentro del grupo de actuadores denominados "Visualizacion", podemos decidir si deseamos que se refresque o no la visión en primera persona del agente activando o desactivando la opción **1Persona**. El interruptor **Mapa** permite ver el mapa que se lleva reconocido frente a la visión completa del mapa. Este interruptor sólo tiene aplicación en los niveles 0, 1 y 4 que son los niveles en el que el mapa original no se conoce. El interruptor **Auxiliar** permite cambiar la visión de primera persona entre el agente Rescatador y el agente Auxiliar. Cuando el interruptor está deshabilitado se muestra la visión del Rescatador.

Se puede observar que bajo el nombre "Ir a .." tenemos la **Fila** y **Columna** de la casilla objetivo. Existe la posibilidad de cambiar el destino desde esta ventana. Si se sitúa el ratón sobre el mapa en la casilla que se desea que sea destino y se pulsa el botón derecho, automáticamente el destino que en ese momento se encuentre activo se cambiará en el mapa y en los campos **Fila** y **Columna**.

No es esta la única forma de poder cambiar la configuración, pero sí lo es sin reiniciar la simulación. Para otra opción que sí requiere una reiniciación debemos volver a pulsar el botón de **Nuevo Juego** y ahora en lugar de dar **Ok**, pulsamos el botón **Ok y Configurar**. Nos aparecerá la ventana que nos permite cambiar los parámetros de la simulación.



Los parámetros modificables son la semilla del generador de números aleatorios, la posición y orientación inicial del agente Rescatador, la posición y orientación inicial del agente auxiliar y el objetivo. Asociada a la fila y columna del objetivo se añade un campo llamado **Grav**. Este campo si toma valor 1 hará activar el sensor Gravedad para el agente Rescatador cuando tenga la casilla entre las que aparecen en su sensor de visión.

Aparecen dos botones asociados con esta ventana. El botón **Ok** se debe pulsar para confirmar los cambios

que se hayan hecho en estos parámetros. El botón **Finish** para cerrar esta ventana.

Para finalizar con la descripción del entorno gráfico, bajo el título de *Informacion*, se recogen los valores en cada instante de algunos de los sensores del agente, así como de alguna información adicional. En concreto, se informa de:

- **Tiempo restante**: cantidad de ciclos de simulación que quedan para terminar.
- **Energía**: cantidad de energía que le queda a cada agente.
- **Posición de los agentes**: indica fila, columna y orientación de cada agente.
- **Última Acción**: indica la última acción que realizó cada agente.
- **Tiempo Consumido**: expresado en segundos la cantidad acumulada de tiempo que ha utilizado cada agente en tomar las decisiones hasta este momento de la simulación.
- **Zapatillas:** indicará si los agentes están en posesión de este objeto. Si no la tiene aparecerá con el valor "----------".
- **Puntuación**: indica el valor 1 en los primeros 4 niveles cuando alcance la casilla objetivo y la puntuación por las misiones conseguidas hasta el momento en el nivel 4.

Bajo el texto ********** **Visión** ********** se indican los valores de los que informan los sensores de superficie, agentes y cota en este instante para cada uno de los agentes con la interpretación que ya se indicó anteriormente en este documento.

4.2.2. Sistema *batch*

- Se incluye con el software la posibilidad de crear un ejecutable sin interfaz gráfica para dar la posibilidad de realizar las operaciones de depuración de errores con mayor facilidad, ya que las librerías gráficas incluyen programación basada en eventos que alteran el normal funcionamiento de las herramientas como el conocido debugger **`gdb`.** Al hacer **`make`** se generan automáticamente los dos ejecutables ***practica2*** y ***practica2SG***. Tanto la versión gráfica como la versión sin gráficos hacen uso los archivos que describen el comportamiento del agente, por esta razón, su uso principal será para rastrear errores en vuestro propio código.

Ya que no tiene versión gráfica, cuando se usa **practica2SG** es necesario pasar todos los parámetros en la línea de comandos para que funcione correctamente. Una descripción de su sintaxis para su invocación sería la siguiente:

```
./practica2SG <mapa> <semilla> <nivel> <filaR> <colR> <oriR> <filaA> <colA> <oriA>
<filO_i> <colO_i> <GO_i>
```

donde

1. **<mapa>** es el camino y nombre del mapa que se desea usaremos
2. **<semilla>** es un número entero con el que se inicia el generador de números aleatorios
3. **<nivel>** es un número entero entre 0 y 4 indicando que nivel se quiere ejecutar
4. **<filaR>** fila inicial donde empezará el agente Rescatador
5. **<colR>** columna inicial donde empezará el agente Rescatador
6. **<oriR>** un número entre 0 y 7 indicando la orientación con la que empezará el agente Rescatador, siendo 0=norte, 1=noreste, 2=este, 3=sureste, 4=sur, 5=suroeste, 6=oeste y 7=noroeste.
7. **<filaA>** fila inicial donde empezará el agente auxiliar
8. **<colA>** columna inicial donde empezará el agente auxiliar
9. **<oriA>** un número entre 0 y 7 indicando la orientación con la que empezará el agente auxiliar, siendo 0=norte, 1=noreste, 2=este, 3=sureste, 4=sur, 5=suroeste, 6=oeste y 7=noroeste.
10. **<filO$_i$>** fila de la casilla del objetivo i-ésimo
11. **<colO$_i$>** columna de la casilla del objetivo i-ésimo.
12. **<GO$_i$>** 1 para activar el sensor de Gravedad.

Por ejemplo, podemos ejecutar **./practica2 mapas/mapa30.map 1 0 4 3 0 8 4 4 12 5 0** lo cual utilizará el mapa llamado **mapa30.map** indicado con una semilla **1** en el nivel **0** situando al agente Rescatador en la posición de fila **4** y columna **3**, con orientación **norte** (**0**), situando al agente auxiliar en la fila **8** y columna **4** con orientación **sur** (**4**). Los tres siguientes valores indican la casilla objetivo que se debe alcanzar (sería ir a la casilla de

fila 12 y columna 5 y el accidentado que se encontrarían ahí no está grave (0). Al estar en el nivel 0 (al igual ocurriría en el nivel 1), estos tres últimos valores no se considerarán.

En el caso de no indicar destinos suficientes (esto solo ocurre en el nivel 4), el simulador los elegirá al azar. Si se proponen más destinos de los necesarios, los no necesarios se ignorarán.

Al finalizar la ejecución, en función del nivel te devuelve la siguiente información:

- En el nivel 0 se muestra el gasto de energía de los agentes Rescatador y Auxiliar.
- En el nivel 1 se muestra el porcentaje de casillas de tipo camino y sendero que se han descubierto. Cada error que cometa resta un acierto.
- En el nivel 2 se muestra el coste en energía y el número de iteraciones consumidas por el agente Rescatador.
- En el nivel 3 se muestra el coste de energía y el número de iteraciones consumidas por el agente Auxiliar.
- En el nivel 4 aparece una información más completa de la simulación al igual que cuando se termina usando la versión gráfica. La información es:
    - *Instantes de simulación no consumidos.*
    - *Tiempo Consumido.*
    - *Nivel Final de Energía (Rescatador).*
    - *Nivel Final de Energía (Auxiliar).*
    - *Colisiones.*
    - *Empujones.*
    - *Porcentaje de mapa descubierto.*
    - *Porcentaje descubierto de caminos y senderos.*
    - *Objetivos encontrados y Puntuación.*

4.2.3. Sistema *batch* y entorno gráfico

Se incluye una tercera posibilidad de ejecución que consiste en combinar el modelo *batch*, para poder indicarle las condiciones de la simulación, con visualizar el comportamiento real del agente levantando el entorno gráfico. La forma de invocar esta opción es igual: usar los mismos parámetros en el mismo orden con el mismo significado que se describen en la versión *batch* pero aplicado sobre **practica2**, es decir,

```
./practica2 <mapa> <semilla> <nivel> <filaR> <colR> <oriR> <filaA> <colA> <oriA>
<filO_i> <colO_i> <GO_i>
```

Algo a destacar cuando se toma esta opción para ejecutar el software es que la simulación queda detenida tras la ejecución de la primera acción del agente.

4.3. Descripción del Software

```
#ifndef COMPORTAMIENTORESCATADOR_H
#define COMPORTAMIENTORESCATADOR_H

#include <chrono>
#include <time.h>
#include <thread>

#include "comportamientos/comportamiento.hpp"

class ComportamientoRescatador : public Comportamiento
{

public:
  ComportamientoRescatador(unsigned int size = 0) : Comportamiento(size)
  {
    // Inicializar Variables de Estado Niveles 0,1,4
  }
  ComportamientoRescatador(std::vector<std::vector<unsigned char>> mapaR) : Comportamiento(mapaR)
  {
    // Inicializar Variables de Estado Niveles 2,3
  }
  ComportamientoRescatador(const ComportamientoRescatador &comport) : Comportamiento(comport) {}
  ~ComportamientoRescatador() {}

  Action think(Sensores sensores);

  int interact(Action accion, int valor);

  Action ComportamientoRescatadorNivel_0(Sensores sensores);
  Action ComportamientoRescatadorNivel_1(Sensores sensores);
  Action ComportamientoRescatadorNivel_2(Sensores sensores);
  Action ComportamientoRescatadorNivel_3(Sensores sensores);
  Action ComportamientoRescatadorNivel_4(Sensores sensores);

private:
  // Variables de Estado
};

#endif
```

De todos los archivos que se proporcionan para compilar el programa, el estudiante solo puede modificar 4 de ellos, los archivos ***'rescatador.hpp', 'rescatador.cpp' 'auxiliar.hpp'*** y ***'auxiliar.cpp'*** que se incluyen en la carpeta Comportamiento_Jugador. Estos archivos contendrán los comportamientos implementados para nuestros agentes reactivo/deliberativos. Además, dentro de estos cuatro archivos, no se puede eliminar nada

de lo que hay originalmente, únicamente se puede añadir. Pasamos a ver con un poco más de detalle algunos de estos archivos.

Empecemos con el archivo ***'rescatador.hpp'***, que junto con ***'rescatador.cpp'*** son los dos archivos necesarios para definir el comportamiento del agente Rescatador.

De los métodos implementados en la parte pública vamos a destacar 3 de ellos:

- El constructor usado para los niveles 0, 1 y 4. Aquí se tendrán que inicializar las variables de estado que se consideren necesarias para resolver dichos niveles.

  `ComportamientoRescatador(unsigned int size) : Comportamiento(size)`

- El constructor usado en los niveles 2 y 3. Aquí se tendrán que inicializar las variables de estado que se consideren necesarias para resolver estos niveles.

  `ComportamientoRescatador(std::vector< std::vector< unsigned char> > mapaR)`

- El método que describe el comportamiento del agente y que debe ser desarrollado por el estudiante para ir incorporándole la resolución de los distintos niveles de la prácticas.

  `Action think(Sensores sensores)`

  Se ha definido una estructura de decisión de tipo **`switch`** dentro del método **`think`** para redirigir la solución de cada nivel en una función distinta de nombre **`ComportamientoRescatadorNivel_?`**.

  Esta estructura se propone a modo de sugerencia. El estudiante puede decidir completamente cómo será la estructura que siga su método **`think`**.

```cpp
#include "../Comportamientos_Jugador/rescatador.hpp"
#include "motorlib/util.h"

Action ComportamientoRescatador::think(Sensores sensores)
{
    Action accion = IDLE;

    switch (sensores.nivel)
    {
    case 0:
        // accion = ComportamientoRescatadorNivel_0 (sensores);
        break;
    case 1:
        // accion = ComportamientoRescatadorNivel_1 (sensores);
        break;
    case 2:
        // accion = ComportamientoRescatadorNivel_2 (sensores);
        break;
    case 3:
        // accion = ComportamientoRescatadorNivel_3 (sensores);
        break;
    case 4:
        // accion = ComportamientoRescatadorNivel_4 (sensores);
        break;
    }

    return accion;
}

int ComportamientoRescatador::interact(Action accion, int valor)
{
    return 0;
}

Action ComportamientoRescatador::ComportamientoRescatadorNivel_0(Sensores sensores)
{
    // El comportamiento de seguir un camino hasta encontrar un puesto base.
}
```

Como ya vimos, existe una variable **`mapaResultado`** que se incluye a partir del fichero ***'comportamiento.hpp'*** en donde se encuentra el mapa. En los niveles del 2 y 3 esta variable se utiliza básicamente como si fuera de sólo lectura (junto con la variable **`mapaCotas`** que indica la altura de cada casilla en el mapa), mientras que en los niveles 0, 1 y 4 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 los niveles 0, 1 y 4, la variable **`mapaCotas`** se encuentra inicializada a 0.

De forma similar, en los archivos ***'auxiliar.cpp'*** y ***'auxiliar.hpp'*** se describirá el comportamiento del agente Auxiliar.

Revisad el tutorial para ver cómo se han de modificar dichos ficheros ya que, en su versión inicial, el método **`think`** qué es el que hay que desarrollar se encuentra vacío.

5. Método de evaluación y entrega de prácticas

5.1. Entrega de prácticas

Se pide desarrollar un programa (modificando el código de los ficheros del simulador ***rescatador.cpp***, ***rescatador.hpp***, ***auxiliar.cpp*** y ***auxiliar.hpp***) con el comportamiento requerido para cada agente en cada nivel. Estos cuatro ficheros deberán entregarse en la plataforma PRADO de la asignatura, en un fichero ZIP, que no contenga carpetas, de nombre ***practica2.zip***. Este archivo ZIP deberá contener sólo el código fuente de estos cuatro ficheros con la solución del estudiante.

No se evaluarán aquellas prácticas que no contengan exclusivamente estos 4 ficheros

5.2. Cuestionario de autoevaluación

Tras la entrega de la práctica se habilitará un plazo de 2 días para que los estudiantes realicen un proceso de autoevaluación de su trabajo. Para ello se suministrará un documento en el que se pedirá al estudiante que responda una serie de preguntas sobre cómo realizó la práctica, sobre algunas cuestiones de diseño y que ponga a prueba su software a partir de una serie de configuraciones iniciales que se le propondrán. El objetivo es determinar si se alcanza el grado de satisfacción para considerar superados los niveles presentados por los estudiantes.

5.3. Método de evaluación

En el método de evaluación se asocia una valoración máxima en puntos a cada uno de los niveles que se piden en esta práctica, siendo la distribución de puntos y requisitos para obtener dichos puntos los que se describen a continuación:

Nivel	Puntuación	Requisito
0	1	Sin requisitos
1	2.5	Tener calificación mayor a 0 el nivel 0

2	2	Sin requisitos
3	2	Tener calificación mayor a 0 el nivel 2
4	2.5	Tener calificación mayor a 0 en los niveles 2 y 3

El valor exacto que el estudiante obtendrá en cada nivel dependerá del número de test que pase satisfactoriamente sus agentes en cada uno de ellos. Estos test de evaluación son los que se aportarán en el documento de autoevaluación. En concreto, en el nivel 1, la calificación estará relacionada con el porcentaje correcto de casillas de tipo camino y sendero que se hayan localizado. En el nivel 4 la puntuación estará relacionada con la puntuación final obtenida en la consecución de misiones de rescate.

5.4. Fechas Importantes

La fecha tope para la entrega será el **DOMINGO 4 DE MAYO DE 2025** antes de las **23:00 horas** y desde el 5 DE MAYO estará disponible la entrega para el cuestionario de autoevaluación hasta **EL MIÉRCOLES 8 DE MAYO** a las **23:00 horas**.

5.5. 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 estudiantes. 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 estudiante para enfrentarse a este problema. Por consiguiente, se asume que todo el código que aparece en su práctica ha sido introducido por el estudiante por alguna razón y que, por lo tanto, domina perfectamente el código que entrega. Así, si durante cualquier momento del proceso de defensa el estudiante no justifica adecuadamente algo de lo que aparece en su código, la práctica se considerará copiada y, por tanto, suspensa, reservándose los profesores elevar a instancias superiores si la falta se considerará grave.

Por esta razón, aconsejamos que el estudiante no incluya nada en su código que 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 vengan acompañadas de la entrega del documento de autoevaluación realizado por el estudiante, se considerarán como no entregadas y por consiguiente se 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 estudiante podría tener una penalización más grave.