INTELIGENCIA ARTIFICIAL

CURSO 2024-25

PRACTICA 2: Repertorio de preguntas para la autoevaluación de la práctica 2.

APELLIDOS Y NOMBRE	Muñoz Velasco Jesús		
GRUPO TEORÍA	DGIIM	**GRUPO PRÁCTICAS**	1

Instrucciones iniciales

En este formulario se proponen preguntas que tienen que ver con ejecuciones concretas del software desarrollado por los estudiantes. También aparecen preguntas que requieren breves explicaciones relativas a como el estudiante ha hecho algunas partes de esa implementación y que cosas ha tenido en cuenta.

En las preguntas relativas al funcionamiento del software del alumno, estas se expresan haciendo uso de la versión de invocación en línea de comandos cuya sintaxis se puede consultar en el guion de la práctica.

El estudiante debe poner en los recuadros la información que se solicita.

En los casos que se solicita una captura de pantalla (***ScreenShot***), extraer la imagen de la ejecución concreta pedida (sustituyendo la llamada `practica2SG` por `practica2`). En los **niveles 0 y 1**, esta captura será de la situación final de la simulación en el modo "mapa" en la que se ve lo que los agentes descubrieron. En los **niveles 2 y 3** donde aparezca la línea de puntos que marca el recorrido (justo en el instante en el que se construye obtiene el plan). Además, en dicha captura debe aparecer al menos el nombre del alumno. Ejemplos de imágenes se pueden encontrar en Imagen1 y en Imagen2.

Consideraciones importantes:

- Antes de empezar a rellenar el cuestionario, actualiza el código de la práctica con los cambios más recientes. Recuerda que puedes hacerlo o bien realizando **git pull upstream main** si has seguido las instrucciones para enlazar el repositorio con el de la asignatura, o bien descargando desde el enlace de GitHub el zip correspondiente, y sustituyendo los ficheros **rescatador.cpp, rescatador.hpp, auxiliar.cpp** y **auxiliar.hpp** por los vuestros.
- Si en alguna ejecución consideras que tu agente se ha visto perjudicado puedes añadirlo a los comentarios en el comentario final (al final del documento).

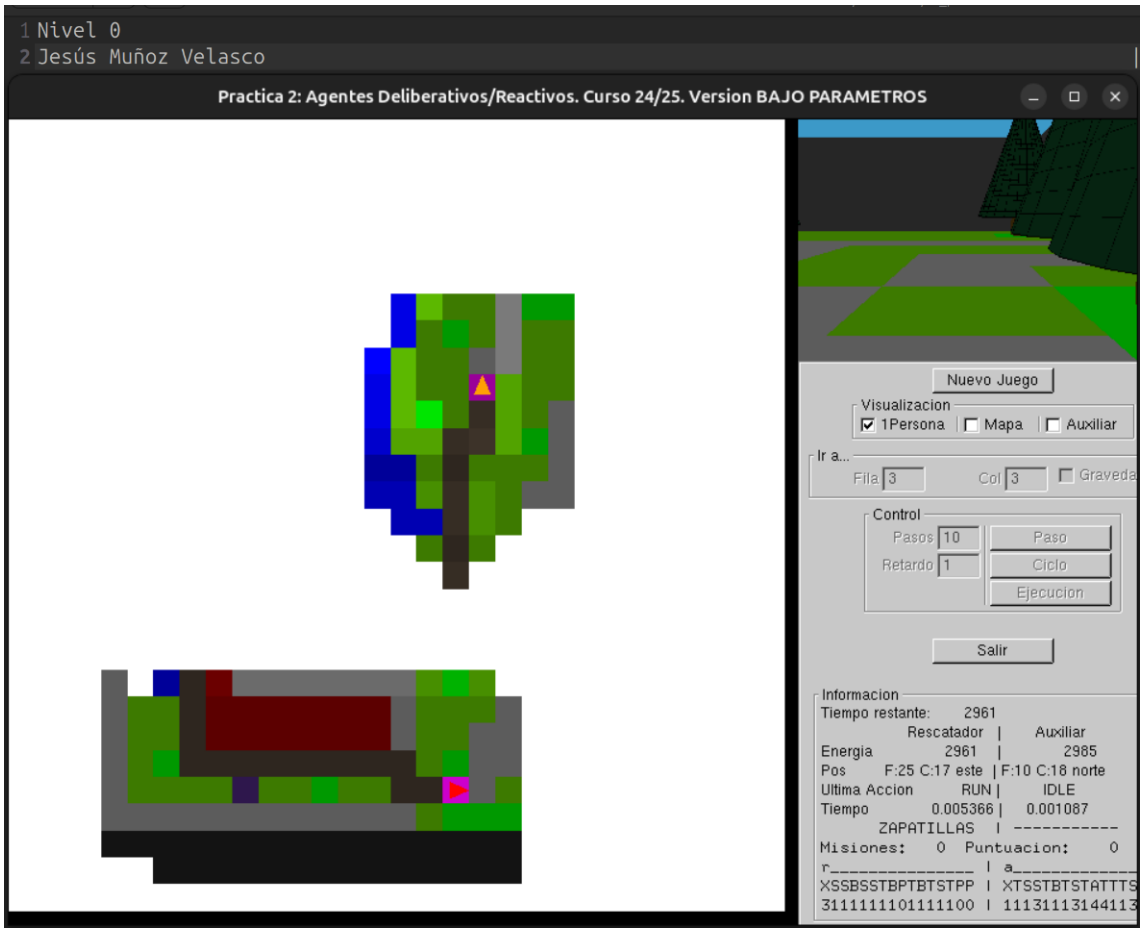
Enumera los niveles presentados en su práctica (Nivel 0, Nivel 1, Nivel 2, Nivel 3, Nivel 4):

Nivel 0, Nivel 1, Nivel 2, Nivel 3, Nivel 4

Nivel 0-El Despertar Reactivo

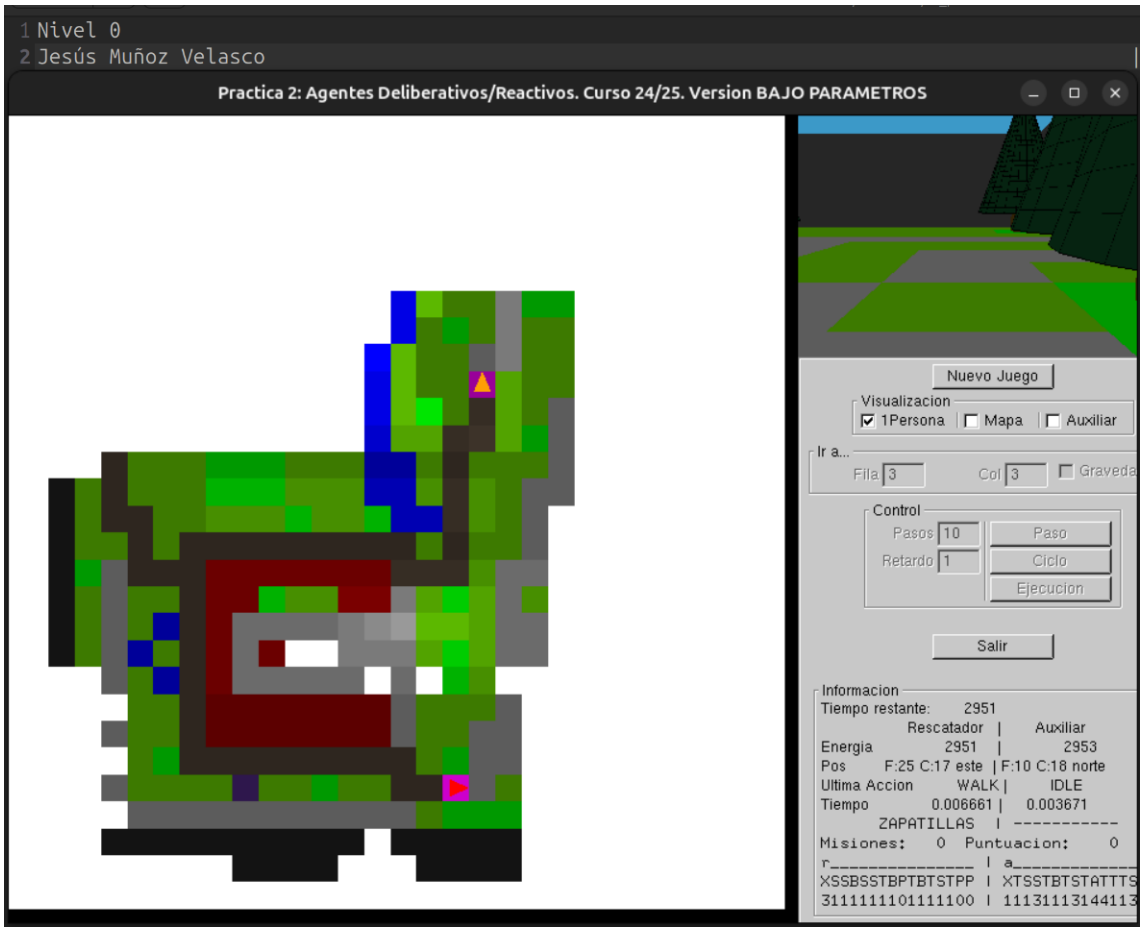
(a) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa30.map 0 0 24 10 2 17 17 0 3 3 0

ScreenShot	
Instantes de simulación no consumidos	2961
Coste de Energía (Rescatador)	39
Coste de Energía (Auxiliar)	15

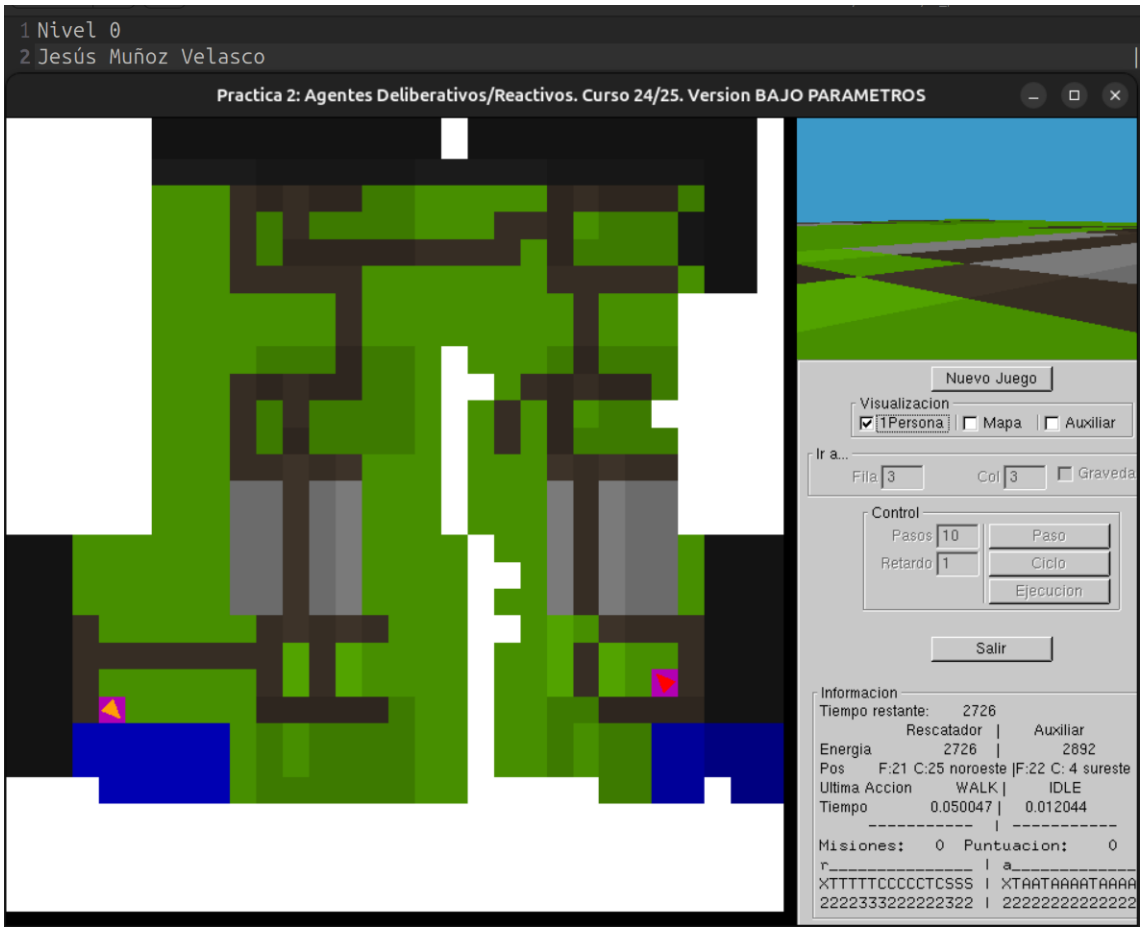
(b) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa30.map 0 0 16 9 2 16 14 6 3 3 0

ScreenShot	
Instantes de simulación no consumidos	2951
Coste de Energía (Rescatador)	49
Coste de Energía (Auxiliar)	47

(c) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/gemini2.map 0 0 3 10 2 3 13 6 3 3 0

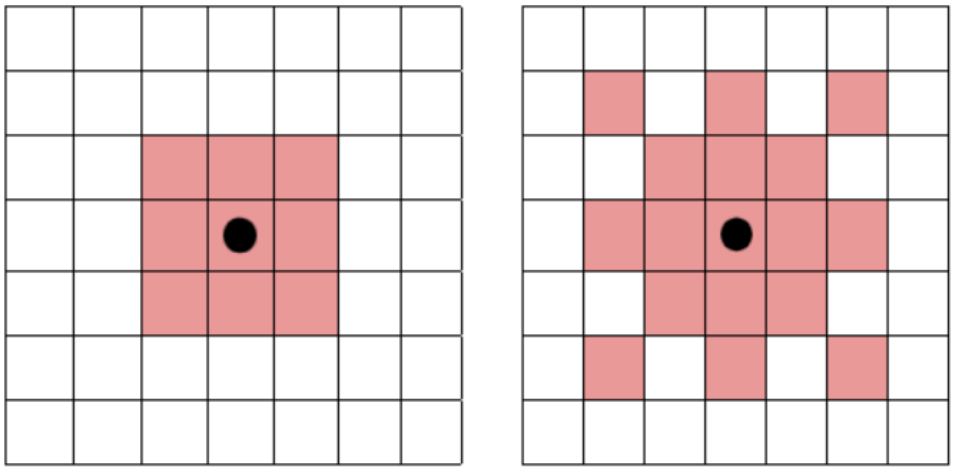
ScreenShot	
Instantes de simulación no consumidos	2726
Coste de Energía (Rescatador)	274
Coste de Energía (Auxiliar)	108

Nivel 1-La cartografía de lo Desconocido

(a) Describe brevemente cuál es el comportamiento que has implementado en los agentes para explorar el mundo. Indica si has usado los dos. Si has usado los dos indica describe las diferencias que hubiera entre ellos. (enumera los cambios y describe brevemente cada uno de ellos)

El comportamiento se ha basado en primer lugar en ver las casillas accesibles. Para ello se han considerado las casillas 1, 2, y 3 del campo de visión y ver si son viables por altura (si no lo son pasan a considerarse casillas de tipo 'P', precipicio).

Después se ha comprobado si hay algún agente en el campo de visión (en el caso del rescatador el auxiliar y viceversa) que en un único movimiento pueda acceder a alguna casilla a la que pueda llegar el otro agente. Se han considerado según el siguiente gráfico:

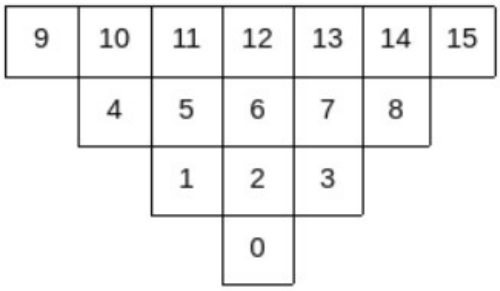


Auxiliar Rescatador

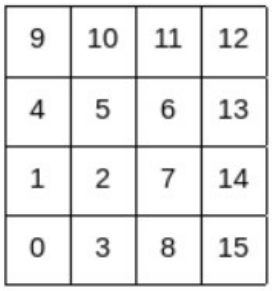
El rescatador considera que cuando el auxiliar está en la situación del círculo negro (en el gráfico de la izquierda) puede llegar según la orientación en la que se encuentre a cualquiera de las casillas del área roja. En el caso del auxiliar considera que el rescatador puede llegar a las del gráfico de la derecha (ya que puede correr).

Una vez hecho esto se ha implementado una función llamada **TomarDecision** a la que se le pasan como parámetros un puntero a función que comprueba el tipo de casilla (luego se desarrollará este punto), en segundo lugar el vector de superficie con las casillas accesibles del campo de visión y las que no (ya procesadas según se han explicado en los puntos anteriores). Se le pasa también un vector con las veces que se han pisado dichas casillas (solo las del campo de visión). Finalmente se le pasa la orientación (únicamente si es par, que serían norte, sur, este y oeste o si es impar, es decir, noreste, sureste, suroeste y noreste) y los sensores.

Esta función comprueba para cada una de las casillas del área de visión si hay algún camino que nos permita llegar a cada casilla. Para ello se tienen varios niveles de prioridad, en primer lugar la casilla menos visitada y después la casilla que nos permita avanzar más recto, más lejos y en menos movimientos (en ese orden). Por tanto, el orden (después de la casilla menos visitada) de las casillas ha sido 6, 2, 8, 4, 7, 5, 3, 1. Estas casillas son las siguientes en el diagrama:



Orientación par



Orientación impar

En primer lugar se utiliza esta función para buscar zapatillas si no tiene (‘D’) y después cualquier tipo de casilla transitable (‘S’ or ‘C’ or ‘X’ or ‘D’ si tiene zapatillas) de forma que si no encuentra ninguna casilla “accesible” zapatillas toma la decisión para las casillas que tiene a la vista. Si no se encuentra ninguna casilla accesible se devuelve 0.

En el caso del rescatador se consideran rutas adicionales teniendo en cuenta que puede correr (hay casillas accesibles por altura que no lo son andando 2 veces) pero el resto del razonamiento es común a ambos (aunque no pueda girar a la izquierda el auxiliar sí que tiene posibilidad de llegar a la misma posición con 6 giros a la derecha).

Las rutas tienen en cuenta la orientación (ya que hay casillas que se tocan solo en la mitad de las orientaciones como la 2 y la 8) y se devuelve solo la primera acción a realizar y si tienen que girar (para no sacar del campo de visión la casilla “objetivo”). De esta forma, siendo viable la ruta se empieza y conforme se vayan haciendo iteraciones al final se llegará a dicha casilla a excepción de que se encuentre por el camino otra mejor (en cuyo caso se comenzarán a ejecutar las acciones necesarias para alcanzar esta casilla y así sucesivamente). Se han considerado las siguientes rutas:

- Objetivo : 6 Orientación: - Ruta: Directa (RUN) Giros: 0 Agentes: R
- Objetivo : 6 Orientación: - Ruta: 2 Giros: 0 Agentes: R A
- Objetivo : 6 Orientación: par Ruta: 3 Giros: 0 Agentes: R A
- Objetivo : 6 Orientación: par Ruta: 1 Giros: 0 Agentes: R A

- Objetivo : 2 Orientación: - Ruta: Directa (WALK) Giros: 0 Agentes: R A

- Objetivo : 8 Orientación: - Ruta: 3 Giros: 0 Agentes: R A
- Objetivo : 8 Orientación: impar Ruta: 2 Giros: 2 Agentes: R A
- Objetivo : 8 Orientación: par Ruta: 2 6 7 Giros: 2 Agentes: R A

- Objetivo : 4 Orientación: - Ruta: 1 Giros: 0 Agentes: R A
- Objetivo : 4 Orientación: impar Ruta: 2 Giros: -2 Agentes: R A
- Objetivo : 4 Orientación: par Ruta: 2 6 5 Giros: -2 Agentes: R A

- Objetivo : 7 Orientación: - Ruta: 2 Giros: 1 Agentes: R A
- Objetivo : 7 Orientación: - Ruta: 3 Giros: 0 Agentes: R A

- Objetivo : 5 Orientación: - Ruta: 2 Giros: -1 Agentes: R A
- Objetivo : 5 Orientación: - Ruta: 1 Giros: 0 Agentes: R A

- Objetivo : 3 Orientación: - Ruta: Directa (TURN_L) Giros: 0 Agentes: R

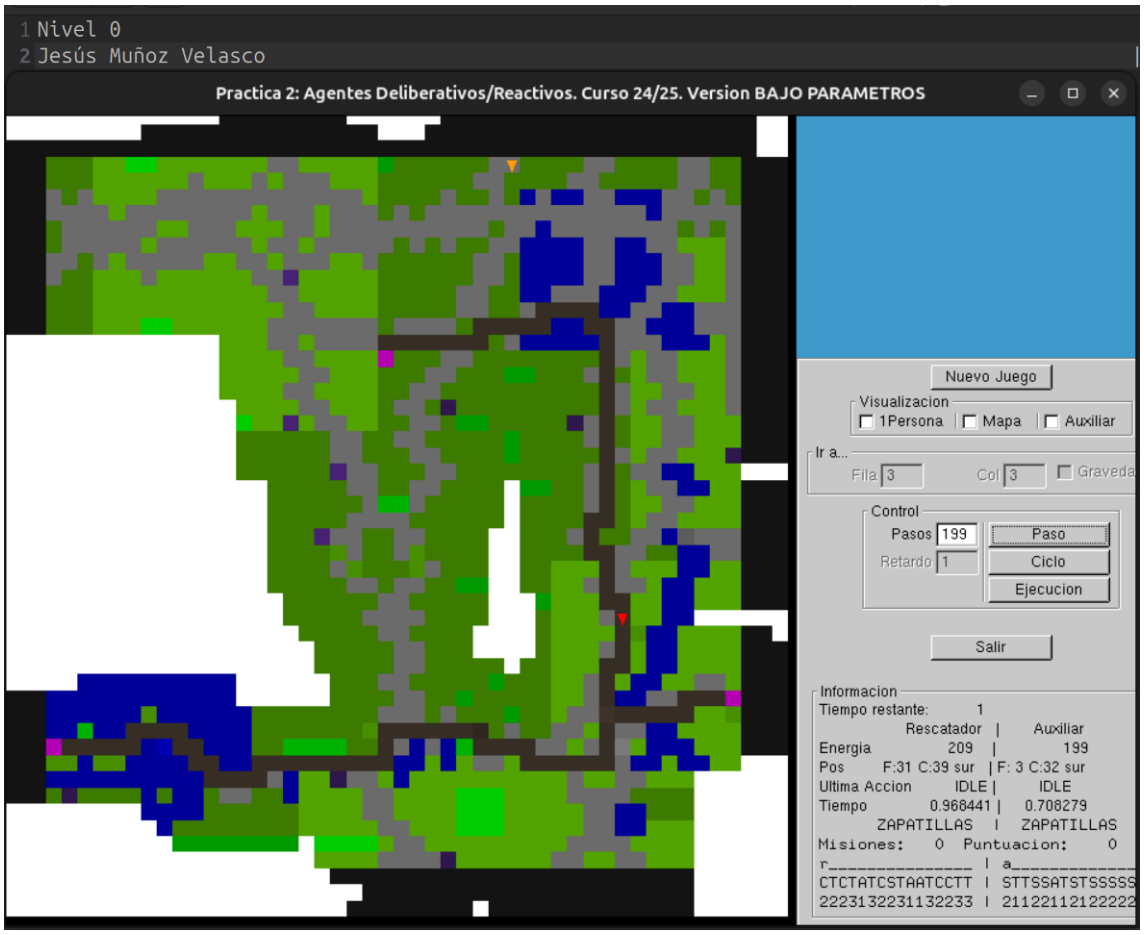
- Objetivo : 1 Orientación: - Ruta: Directa (TURN_SR) Giros: 0 Agentes: R A

Donde “objetivo” es la casilla a la que quiero llegar, “orientación” puede ser ‘-’ que significa cualquiera, ‘par’ que son las posiciones norte, sur, este y oeste o ‘impar’ que son noreste, sureste, suroeste, noroeste; “ruta” son las casillas por las que voy a pasar antes de llegar a la destino; “giros” que es el número de giros de 45º que tengo que hacer después de moverme (en caso de ser negativo será la izquierda y si es positivo será hacia la derecha). Agentes son los agentes que comprueban esas opciones (‘R’ para rescatador y ‘A’ para auxiliar).

Al comenzar a pensar, el agente actualizará las variables de estado (aumenta las veces que ha pisado la casilla actual, ajusta el mapa de cotas y de superficie, comprueba si ha llegado a zapatillas y si tiene un giro pendiente). Después comprueba el nivel de energía (si es menor a un umbral hace IDLE para recargar energía). En caso de haber seguido el pensamiento se ejecuta la función descrita anteriormente para las casillas que tiene a la vista y decide finalmente una acción a ejecutar.

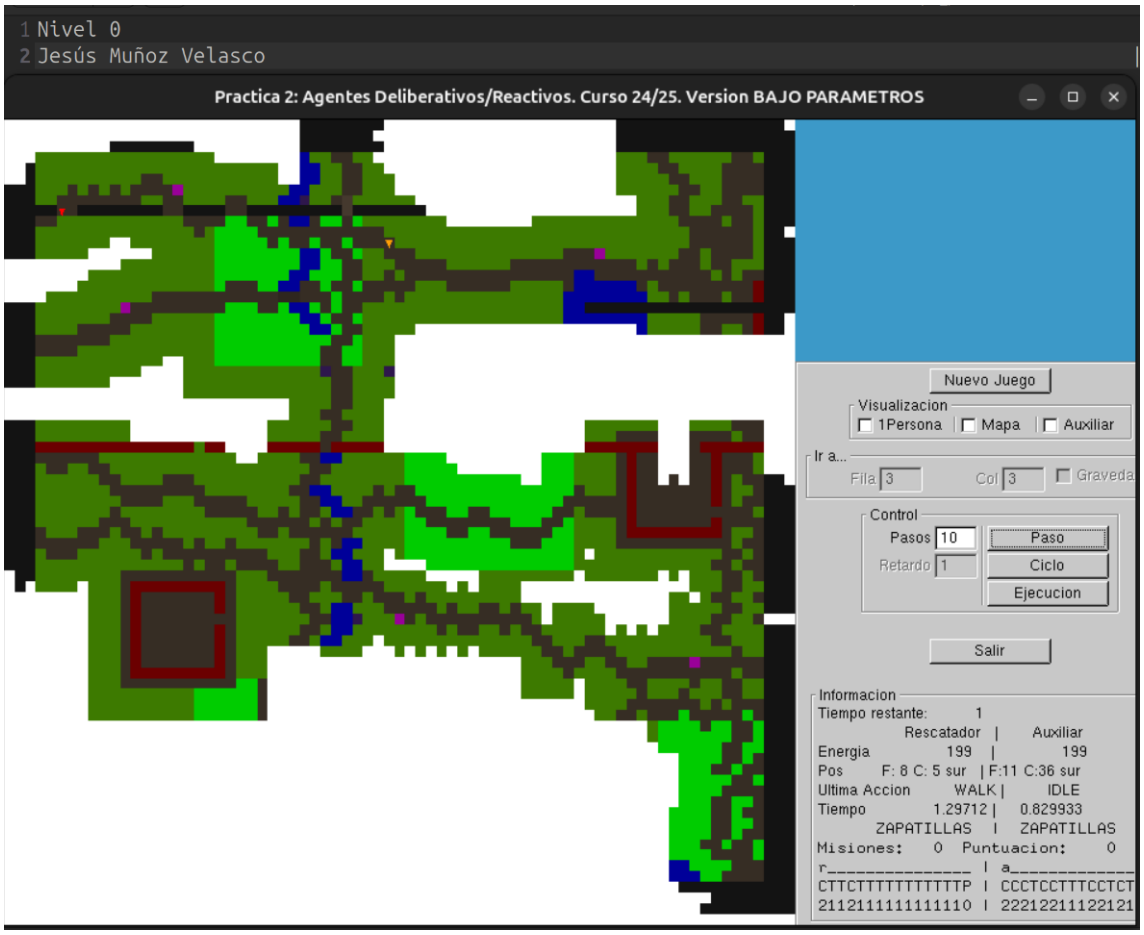
(b) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa50.map 0 1 5 3 2 36 44 6 3 3 0

ScreenShot	
Porcentaje descubierto de caminos y senderos	100

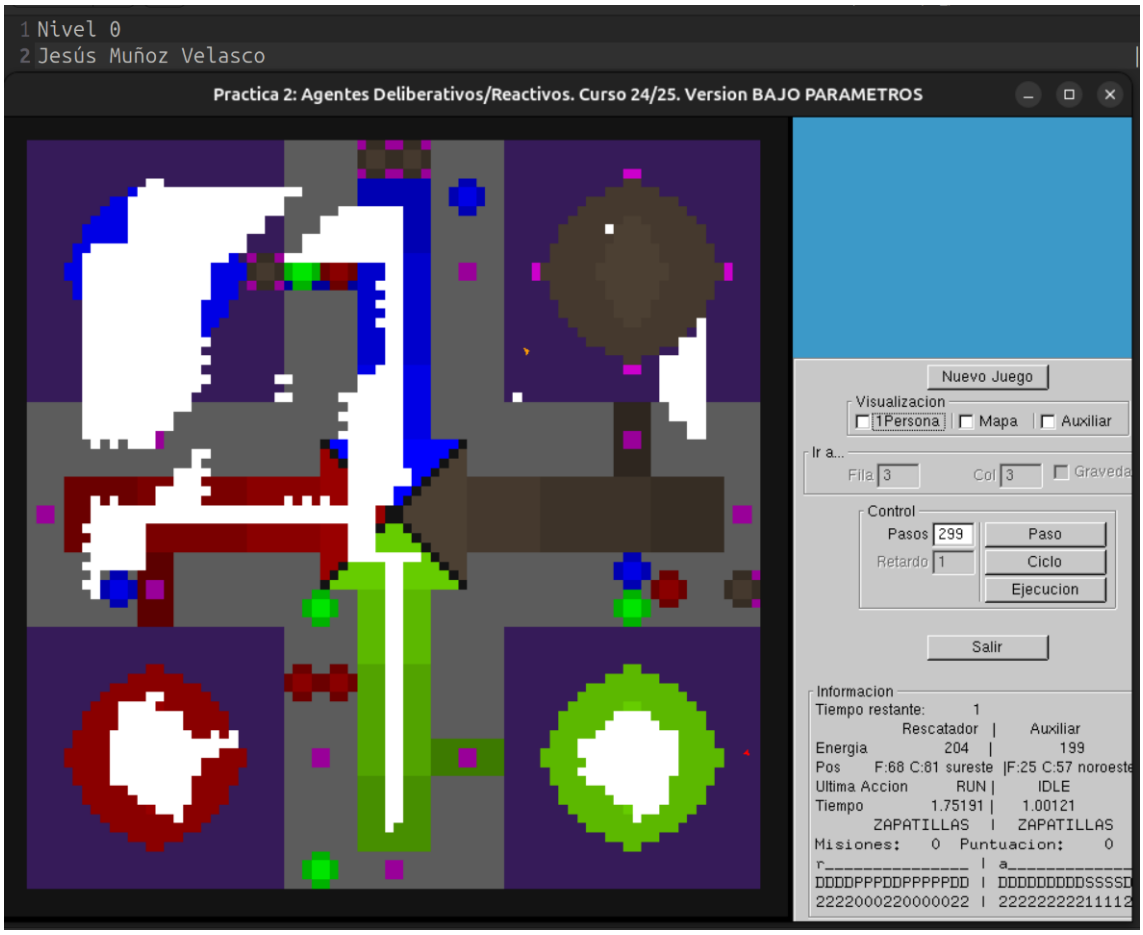
(c) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa75.map 0 1 20 9 2 15 23 6 3 3 0

ScreenShot	
Porcentaje descubierto de caminos y senderos	77.9347

(d) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/parchis.map 1 1 32 68 6 17 68 4 3 3 0

ScreenShot	
Porcentaje descubierto de caminos y senderos	95.3833

Nivel 2-La Búsqueda del Camino de Dijsktra

(a) Indica cuál ha sido la definición de estado para resolver este problema. Justifica la definición.

El estado es la situación posicional del agente recogiendo la información relativa a fila y columna, dirección y si tiene zapatillas.

(b) ¿Has incluido dentro del algoritmo de búsqueda que si pasas por una casilla que da las zapatillas, considere en todos los estados descendientes de él, se está en posesión de las zapatillas? En caso afirmativo, explicar brevemente cómo.

Al crear el hijo compruebo si en sus nuevas coordenadas (hijo.f, hijo.c) hay una casilla de tipo zapatillas ('D') y en caso afirmativo se actualiza su variable correspondiente (hijo.zapatillas) a true. De esta forma cuando se crea un nodo hijo se actualiza por completo su estado (habiendo calculado el resto de información aplicando la función **apply** relativa a cada agente).

(c) En el algoritmo de búsqueda en anchura, el primer nodo que se genera compatible con la solución es la solución óptima y se puede detener la búsqueda en ese punto. Esto no se verifica en el algoritmo de Dijkstra. ¿Tuviste en cuenta esto en tu implementación? Describe brevemente como lo tuviste en cuenta.

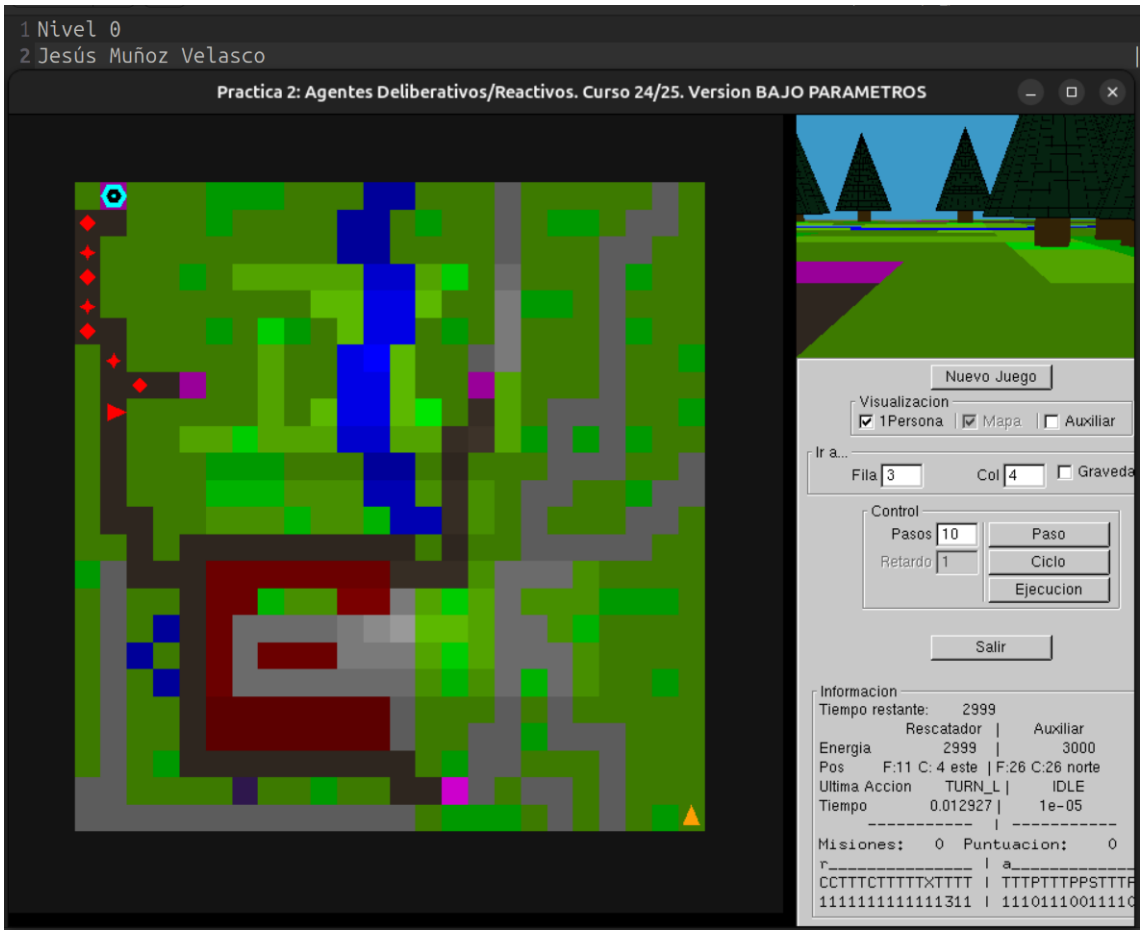
Al utilizar una cola con prioridad para la frontera, se va cogiendo en cada caso el nodo al que se requiere menos energía llegar. De esta forma se comprueba si es solución al sacarlo de **frontier** (y no antes) para que, al sacarlo, todas las demás posibilidades (ramificaciones) requieran una energía mayor y por tanto no va a existir una solución más óptima (esto se debe a que la función de energía es creciente y por tanto si hubiese una solución más óptima, los nodos anteriores requerirían menos energía y se estudiarían antes que el nodo que hemos considerado solución).

(d) Incluye en el siguiente recuadro de texto el trozo de código de tu implementación del algoritmo de Dijkstra que genera el nodo descendiente de aplicar la acción RUN sobre el estado actual. Si tu proceso de generación de descendientes es genérico, pon como trozo de código todo el ciclo donde esté incluida esa generación de los descendientes.

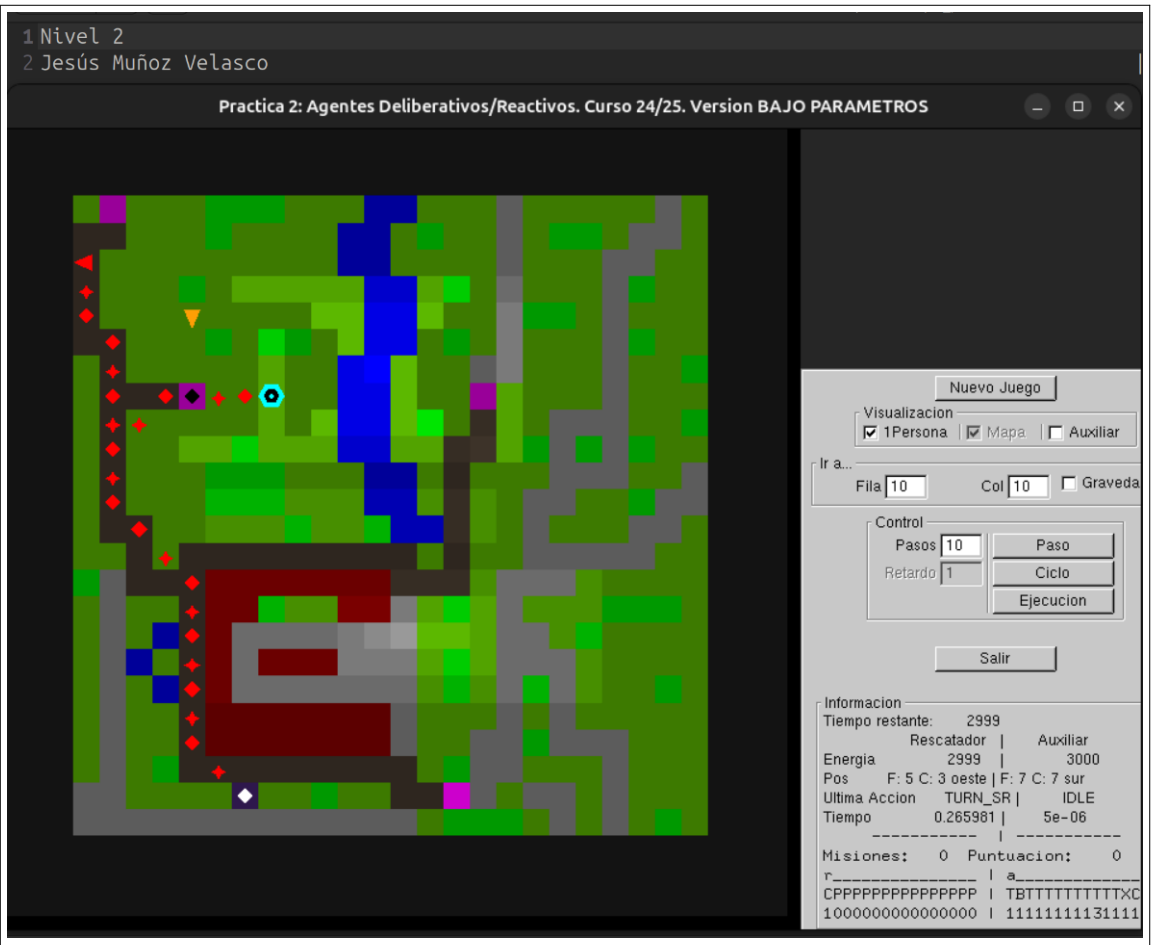
```
// Acciones posibles
const vector<Action> acciones = {RUN, WALK, TURN_SR, TURN_L};

// [...] aquí hay más código pero no se pide (he incluido la línea de arriba para que se vea la variable acciones)

for (int i = 0; i < acciones.size(); ++i)
{
	Action a = acciones[i];

	NodoR hijo = current_node;
	hijo.estado = applyR(a, current_node.estado, terreno, altura, agentes);

	int alt_actual = altura[current_node.estado.f][current_node.estado.c];
	int alt_destino = altura[hijo.estado.f][hijo.estado.c];
	int diff_altura = alt_destino - alt_actual;

	hijo.costo += CosteR(a, terreno[current_node.estado.f][current_node.estado.c], diff_altura);

	hijo.secuencia.emplace_back(a);

	if (explored.find(hijo) == explored.end())
	{
		frontier.emplace(hijo);
	}
}
```

(e) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa30.map 1 2 11 4 4 26 26 0 3 4 0

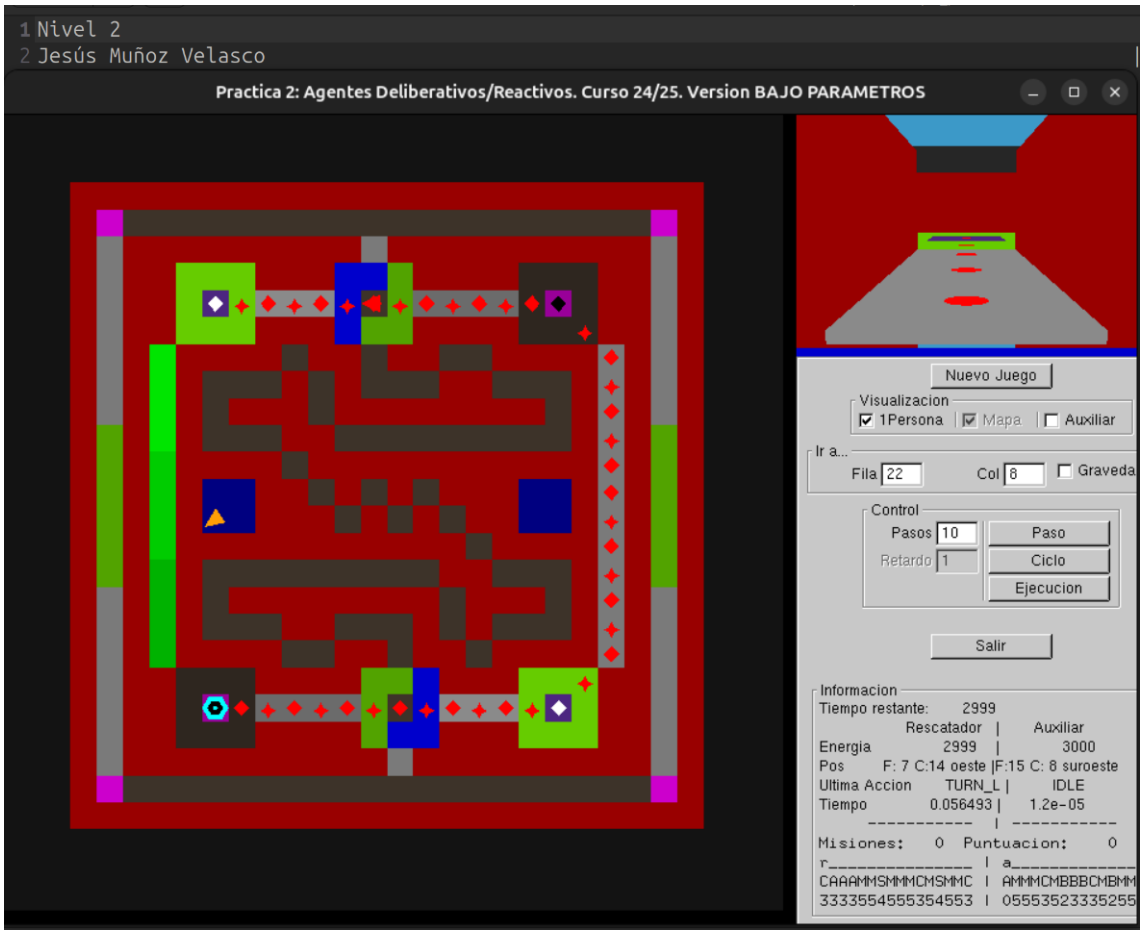
ScreenShot	
Longitud del camino (Rescatador)	11
Coste de Energía (Rescatador)	11

(f) Rellena los datos de la tabla con el resultado de aplicar
./practica2SG ./mapas/mapa30.map 0 2 5 3 5 7 7 4 10 10 0

ScreenShot	
Longitud del camino (Rescatador)	40
Coste de Energía (Rescatador)	64

(g) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/scape25.map 1 2 7 14 0 15 8 5 22 8 0

ScreenShot	
Longitud del camino (Rescatador)	32
Coste de Energía (Rescatador)	61

Nivel 3-El Ascenso del A*uxiliar

(a) ¿Qué diferencia este algoritmo del de Dijsktra que tuviste que implementar en el nivel anterior? (enumera los cambios y describe brevemente cada uno de ellos y que han implicado en la implementación)

En primer lugar implementé una función de heureística con la distancia de Manhattan (máximo de la distancia entre filas y columnas). Añadí una nueva variable a NodoA (en el caso del auxiliar que era lo que se pedía en este nivel) que se llama **estimacion** y que almacena justo el resultado de aplicar esta nueva Heureística al nodo. De esta forma ahora los nodos se ordenan según la suma de **costo** y **estimación**. Además, implementé varios cambios para mejorar la eficiencia (añadí un unordered map para la lista explored y cambié varias cosas de sitio).

(b) Copia y pega en el siguiente recuadro de texto la heurística seleccionada. Además, describela y justifica la razón que hace que sea admisible para este problema.

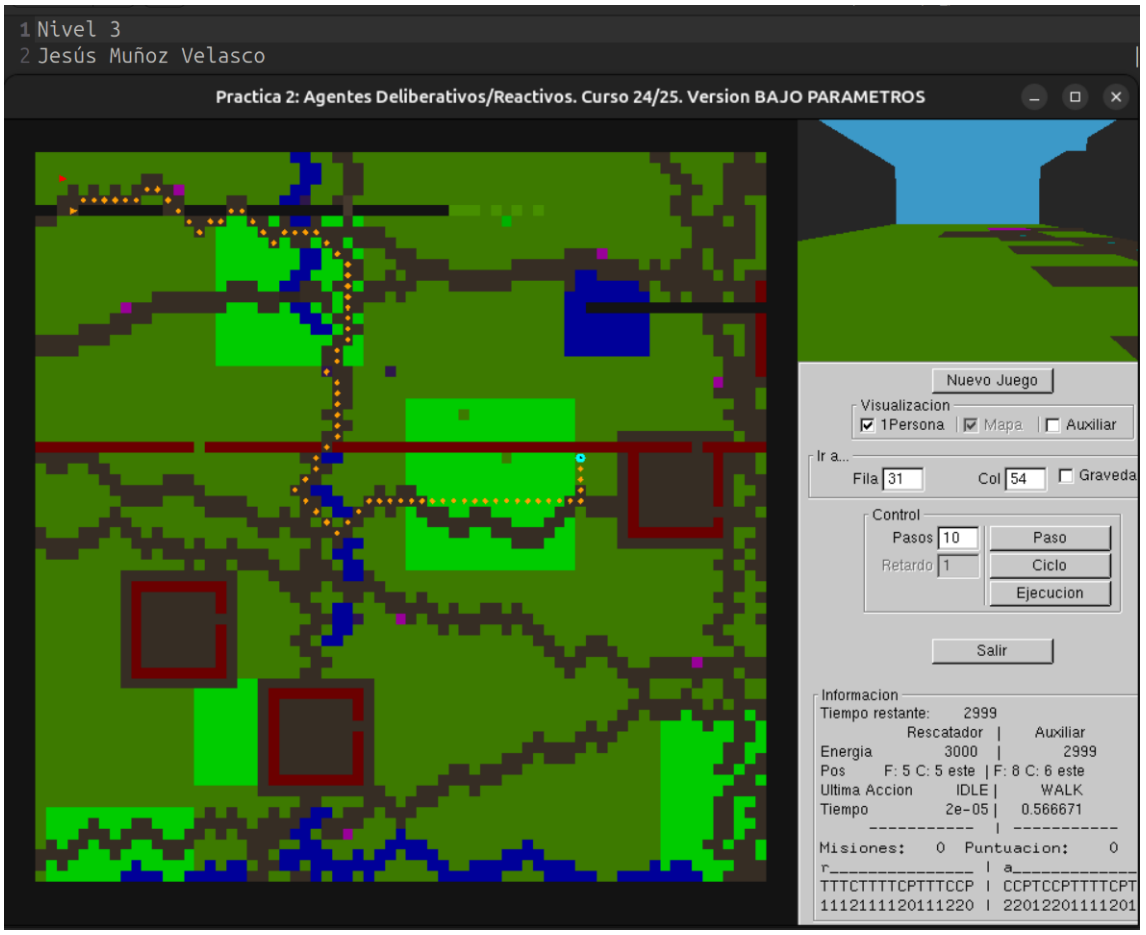
```
// Función de Heureística del Algoritmo A* (coste estimado del camino más barato desde n al objetivo)
int HeureisticaA(const EstadoA &st, const EstadoA &objetivo)
{

	int f = abs(st.f - objetivo.f);
	int c = abs(st.c - objetivo.c);

	int res;

	if (f > c)
	{
		return f;
	}
	else
	{
		return c;
	}
}

// Esta heureística se basa en la distancia del taxi o distancia de Manhattan. En este caso se considera que el mejor camino es la “línea recta”. Hay 3 casos posibles:

- Si el objetivo está en la misma fila, la distancia en columnas es 0 por lo que se tomará como resultado la línea recta (la diferencia del número de filas)

- Si el objetivo está en la misma columna, la distancia en filas es 0 por lo que ahora la estimación será la distancia en columnas (la línea recta de nuevo).

- Si el objetivo no se encuentra ni en la misma fila ni columna podemos dividir el movimiento en 2 partes. Una primera en la que se mueve en diagonal hasta coincidir con la misma fila o
```

columna (en la dirección hacia la que se acerca al objetivo) y después una vez coincide se aplicaría para el resto lo mismo que en los 2 casos anteriores.

Esta es la interpretación geométrica pero si se calcula al final resulta en max{distanciaFila, distanciaColumna} y es la distancia que se aplica en un mundo cuadrado (por los motivos que se han descrito)

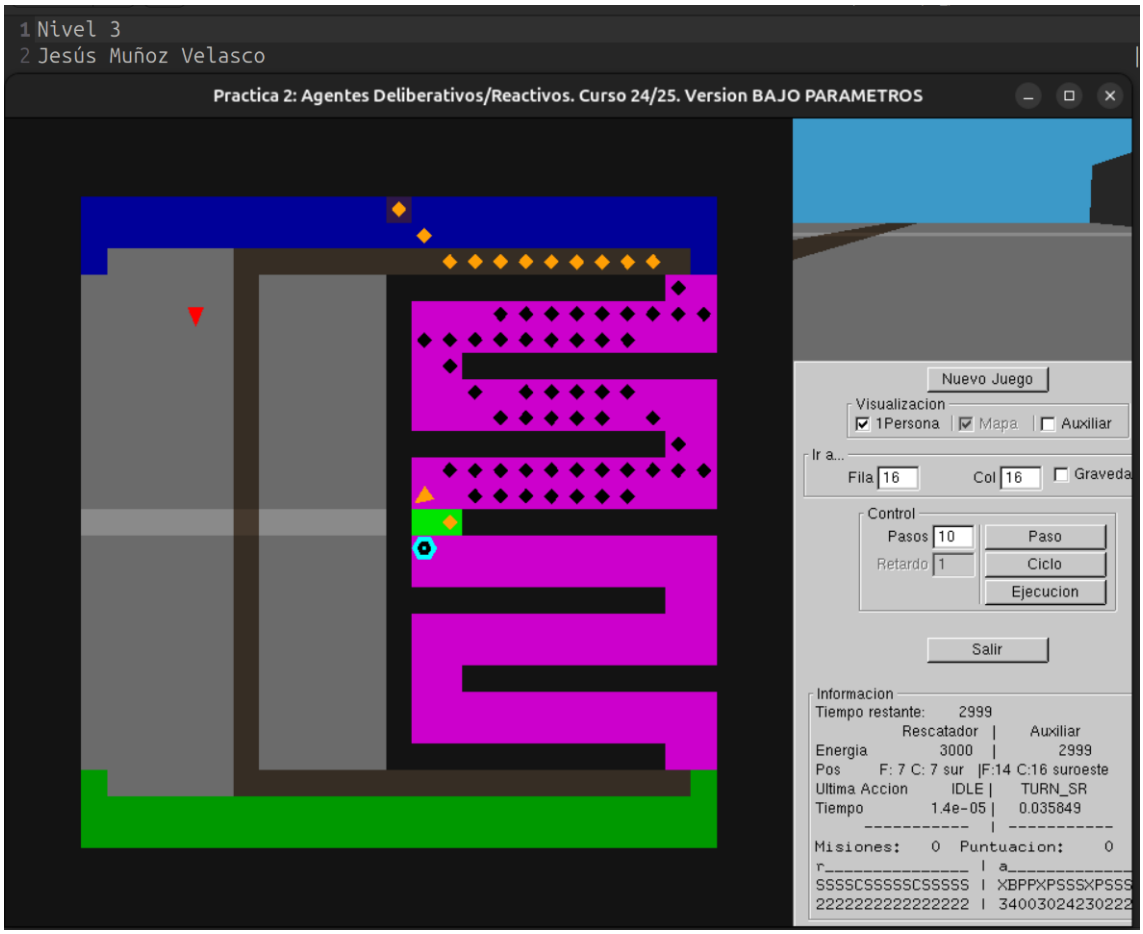
(c) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa75.map 1 3 5 5 2 8 5 2 31 54 0

ScreenShot	
Longitud del camino (Auxiliar)	165
Coste de Energía (Auxiliar)	203

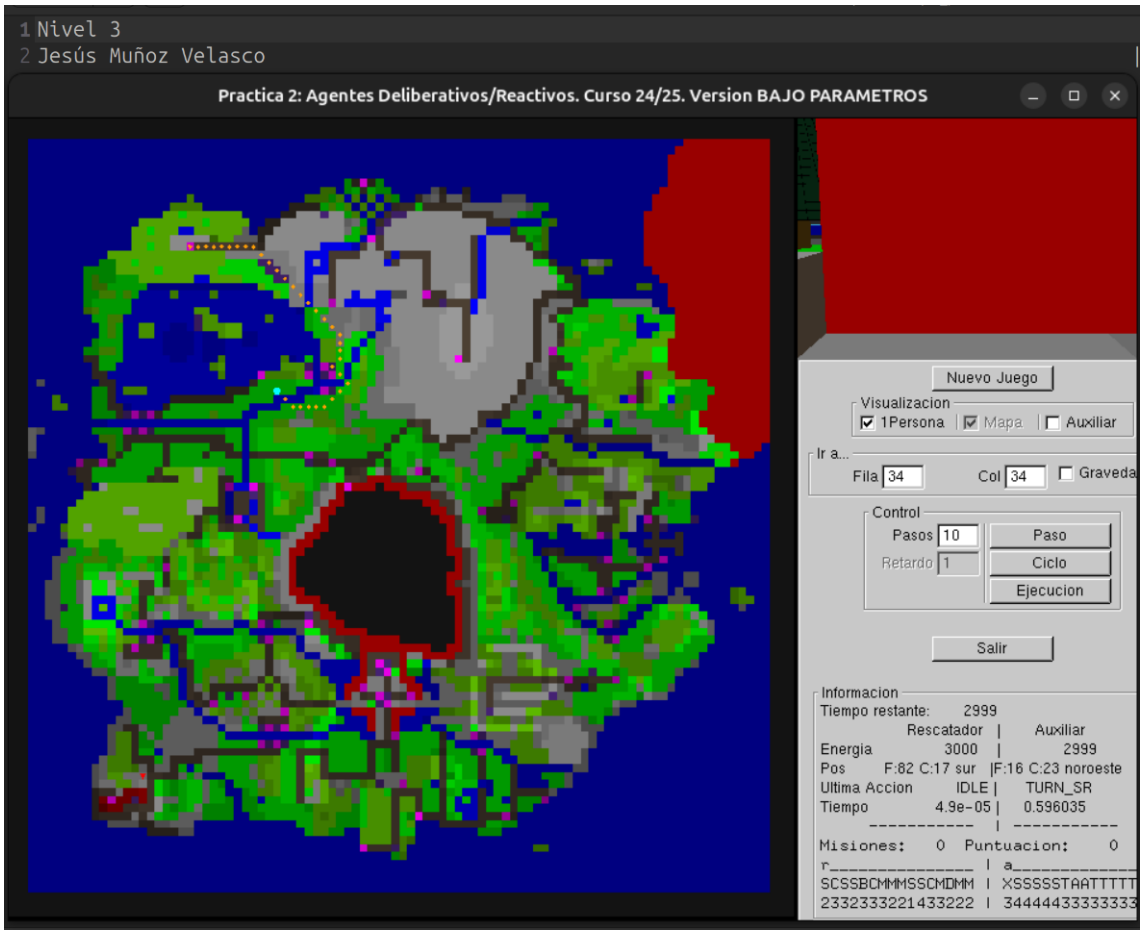
(d) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/2ez.map 0 3 7 7 4 14 16 4 16 16 0

ScreenShot	
Longitud del camino (Auxiliar)	157
Coste de Energía (Auxiliar)	365

(e) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/paldea25.map 0 3 82 17 4 16 23 6 34 34 0

ScreenShot	
Longitud del camino (Auxiliar)	51
Coste de Energía (Auxiliar)	246

Nivel 4-Misión de Rescate

(a) Haz una descripción general de tu estrategia general con la que has abordado este nivel. Explica brevemente las razones de esos criterios.

No he tenido mucho tiempo y sé que se podría haber hecho mejor pero aquí va mi razonamiento:

Para empezar el rescatador comprueba para cada instante el camino más óptimo (en movimientos y con el algoritmo A*) desde su posición hasta la casilla destino (suponiendo que las casillas desconocidas son transitables tanto por terreno como por superficie). Esto es algo **muy** ineficiente y podría haberlo recalculado solo si hubiese cambiado algo en la ruta (le han empujado o ha pasado a conocer más mapa o bueno habría varias formas pero no las he implementado). Además calcula la energía requerida para llegar a dicha casilla.

A su vez en cada instante calcula la mejor ruta (con el mismo criterio anterior) para llegar a una base (en el momento en el que ve la primera), además de la energía requerida para llegar a dicha base.

En cada instante tiene que decidir si es mejor ir a una base o a por un excursionista en apuros (casilla destino). Esto se basa en si tiene la suficiente energía para llegar a una base (más un umbral por posible modificación de la ruta). En caso de que no tenga la suficiente va a dicha base (de nuevo con el umbral le es posible llegar o por lo menos en los casos que he probado) y una vez está en la base hace **IDLE** hasta tener la suficiente energía como para ir y volver 2 veces a la casilla destino (de nuevo con un umbral). Esto hace que sea capaz de llegar al destino y volver a la base a recargar en caso de necesidad.

Una vez llega al destino si está grave llama siempre al rescatador (no he pensado mucho esto) y se pone a dar vueltas (con **TURN_SL** para optimizar el número de giros necesarios para ver al auxiliar).

En este punto se activa el auxiliar que funciona exactamente igual que el rescatador pero adaptando las funciones a sus movimientos. En este caso cuando tiene a la vista la casilla destino se para (esperando a que el rescatador girando llegue a verle). En este punto se completa la misión y se suman los 7 puntos.

Se continúa así hasta que se quede sin energía o movimientos (o sin tiempo de ejecución que es lo más normal).

(b) ¿Qué algoritmo o algoritmos de búsqueda usas en el nivel 4? Explica brevemente la razón de tu elección.

Utilizo el algoritmo A* ya que al ser tan ineficiente en tiempo por lo menos elijo el algoritmo más rápido de los que ya tenía hechos. El resto de la explicación está en el apartado anterior.

(c) ¿Bajo qué condiciones replanifica tu agente?

Bajo todas (en cada movimiento recalcula la ruta).

(d) Explica el valor de coste que le has dado a la casilla desconocida en la construcción de planes cuando el mapa contiene casillas aun sin conocer. Justifica ese valor.

Le he dado el mismo valor que a las demás ya que funciona por número de movimientos y en este caso fomento además acercarse a las zonas desconocidas (ya que al considerar todas enteramente transitables hace muy eficiente la ruta al atravesar estas zonas, ya que se puede hacer en línea recta). De esta forma cada vez se conoce mejor el mapa y se pueden elegir mejor las rutas.

(e) ¿Has tenido en cuenta la recarga de energía? En caso afirmativo, describe la política usada por los agentes.

Ya la he explicado en el primer apartado. Ambos agentes funcionan igual en este punto.

(f) Añade aquí todos los comentarios que desees sobre el trabajo que has desarrollado sobre este nivel, qué consideras son importantes para evaluar el grado en el que te has implicado en la práctica y que no se puede deducir de la contestación a las preguntas anteriores.

Le he dedicado bastante tiempo a pensar posibles comportamientos (sobre todo mejores en eficiencia) pero al final por temas de tiempo no he podido implementar nada más. Funciona más o menos en niveles con mapas pequeños pero no tiene mucho más. Como algoritmo es bastante poco realista por la eficiencia pero prefería implementar aunque fuesen las ideas generales (ya que tenía muchas ideas y por lo menos quería implementar algo).

(g) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa50.map 1 4 28 25 4 28 20 2 36 23 0 39 8 0 46 26 1 39 34 0 26 37 0 18 46 0 3 46 0 3 3 0 10 17 1 39 45 0 9 16 0 38 13 0 27 23 0 31 18 0 45 31 0 35 7 0 12 6 1 40 7 0 20 6 1 10 25 1 41 30 0 14 31 0 26 24 1 38 26 1 38 20 1 44 14 0 17 40 0 45 3 1 4 9 0 33 44 0 17 3 1 3 11 0 42 13 1 26 18 1 38 25 1 33 26 0 46 46 1 36 14 0 36 31 1 17 34 0 8 22 1 44 41 1 16 11 0 44 17 0 29 32 0 42 21 0 46 19 1 40 34 0 45 24 0 46 7 0 44 32 1 21 30 1 14 39 1 15 22 1 11 9 0 13 27 1 20 8 1 45 5 0

Instantes de simulación no consumidos	1983
Tiempo Consumido	0.297202
Nivel Final de Energía (Rescatador)	883
Nivel Final de Energía (Auxiliar)	2062
Objetivos encontrados	(15) 40

(h) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa100.map 1 4 63 31 6 63 32 2 66 40 0 75 24 0 85 36 1 83 6 0 60 10 0 33 11 0 84 7 0 86 40 0 68 77 1 79 91 0 19 33 0 76 25 0 55 47 0 62 36 0 51 95 0 91 63 0 71 14 1 24 13 0 80 15 1 21 51 1 83 61 0 29 63 0 52 49 1 78 52 1 76 40 1 90 28 0 39 80 0 91 6 1 94 52 0 8 19 0 66 89 1 34 6 0 6 23 1 85 26 1 53 37 1 79 51 0 70 53 1 3 43 0

Instantes de simulación no consumidos	2941
Tiempo Consumido	0.000664
Nivel Final de Energía (Rescatador)	2942
Nivel Final de Energía (Auxiliar)	3000
Objetivos encontrados	2 (4)

(i) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/F_islas.map 1 4 47 53 2 49 53 2 41 56 0 52 53 0 74 54 1 74 47 0 46 42 0 71 56 0 83 52 0 58 65 0 85 43 1 92 39 0 81 68 0 91 48 0 21 95 0 92 14 0 88 64 0 43 61 0 28 78 1 30 44 0 22 18 1 27 55 1 41 16 0 90 10 0 12 49 1 76 68 1 38 74 1

Instantes de simulación no consumidos	2681
Tiempo Consumido	1.09768
Nivel Final de Energía (Rescatador)	1275
Nivel Final de Energía (Auxiliar)	2730
Objetivos encontrados	13 (36)

Comentario final

Consigna aquí cualquier tema que creas, que es de relevancia para la evaluación de tu práctica o que quieras hacer saber al profesor.

Tengo que añadir que le he dedicado mucho más tiempo al nivel 1 que al resto de niveles (e indirectamente al 0 por compartir prácticamente todo el código) y muy poco al 4 (al menos a la parte de implementación). Esto se refleja bastante bien en las pruebas realizadas (si no se ha cambiado conseguí la posición 13 de la leaderboard en el nivel 1 y tan solo la 60 en el nivel 4)

Me ha gustado la práctica y el concepto de los tests y la leaderboard me ha parecido bastante motivador pero sí es verdad que me hubiese gustado dedicarle más tiempo a la práctica (sé que es mucho pedir pero hubiese hecho una práctica para todo el cuatrimestre entregando los niveles a plazos para distribuir mejor el tiempo entre ellos).

Por cierto el tiempo del nivel 4 no es nada realista. No sé exactamente qué valor muestra en el output pero me tarda mucho más de lo que pone (termina en los dos últimos casos debido a que se agotó el tiempo de deliberación).