

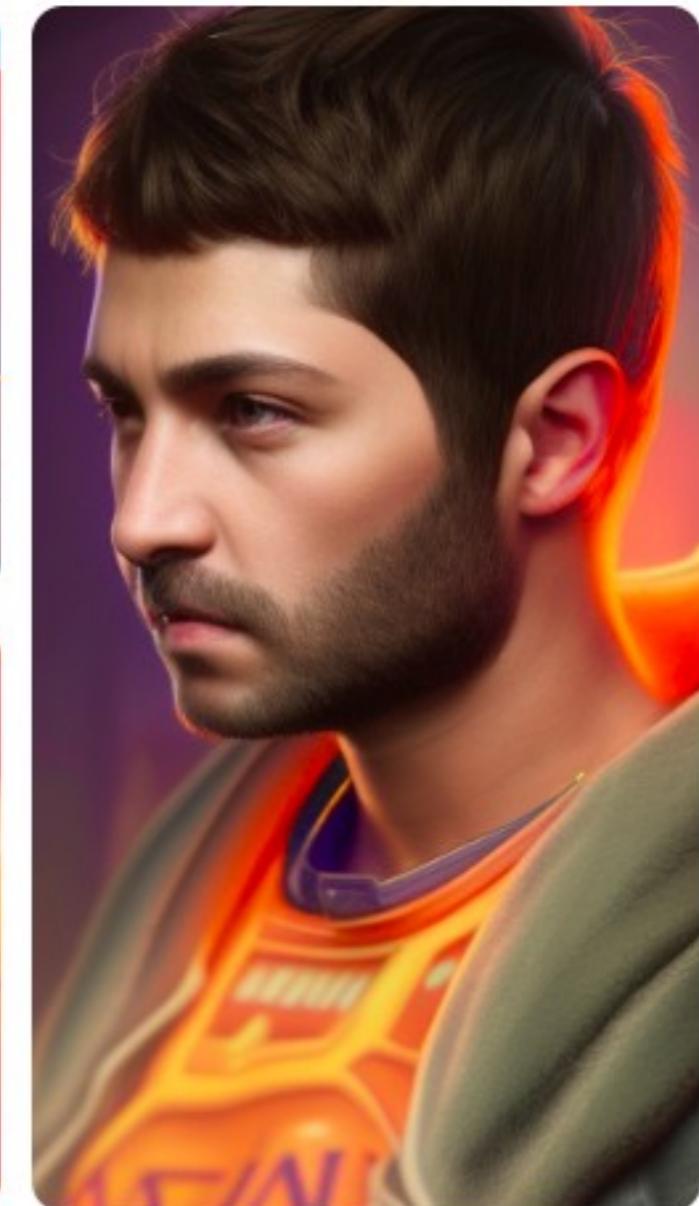
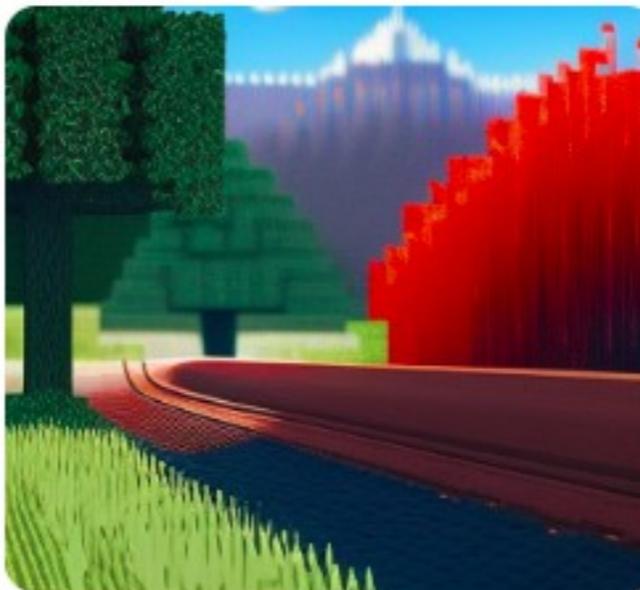
# Inteligencia Artificial



Universidad de Granada

## Práctica2

Agentes  
Reactivos/Deliberativos



Departamento de Ciencias de la  
Computación e Inteligencia Artificial



- **Introducción**
- Breve descripción del juego
- Objetivos
- Software
- Método de evaluación/entrega de prácticas



En esta práctica abordaremos la implementación de algunos algoritmos típicos de búsqueda usados en IA (sin y con información)

- Estos algoritmos lo aplicaremos a una versión muy parecida al juego que ya se utilizó en la práctica 1.
- A continuación volvemos a describir brevemente los elementos que forman parte del juego.

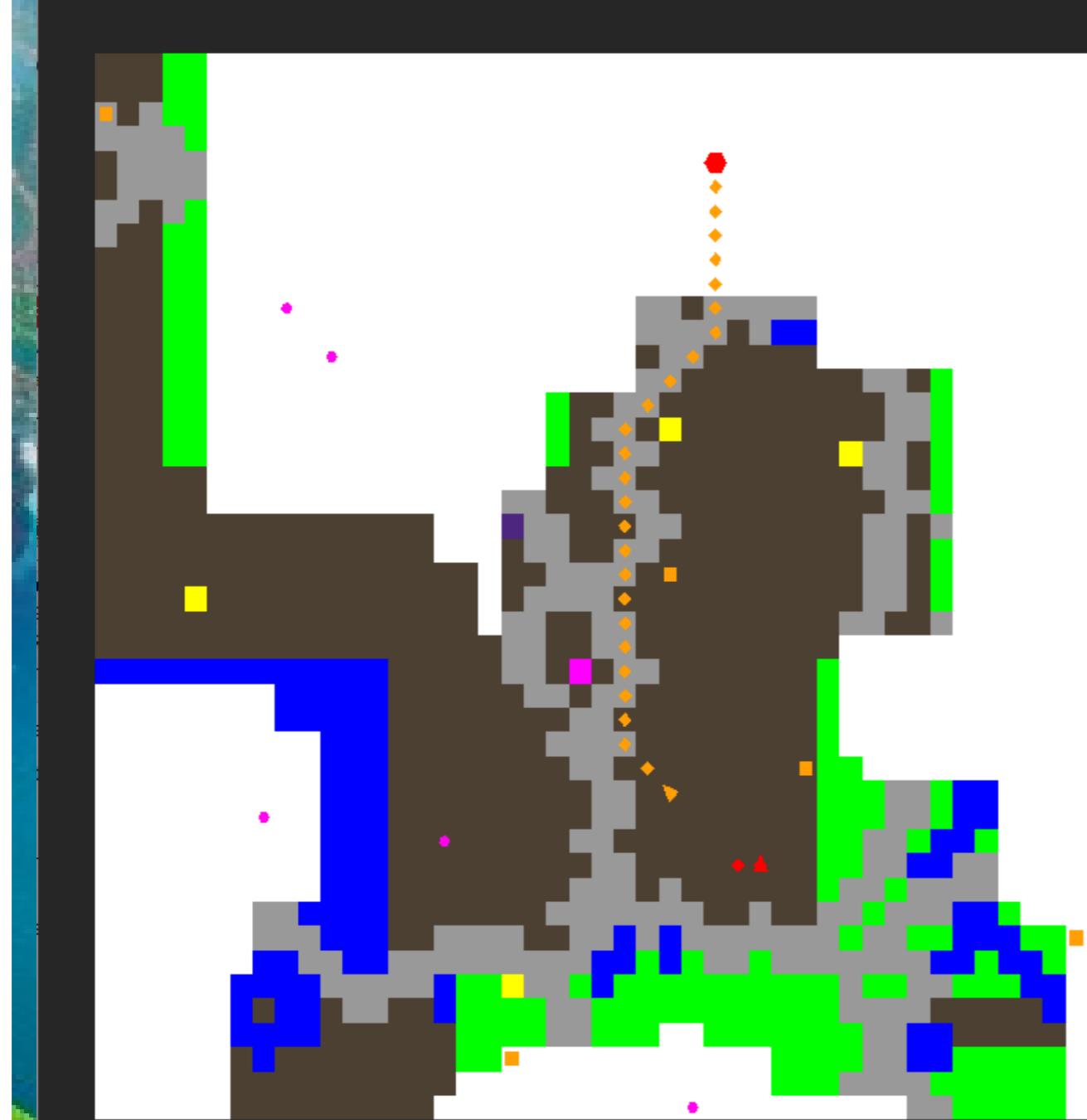




- Introducción
- Breve descripción del juego
- Objetivos
- Software
- Método de evaluación/entrega de prácticas



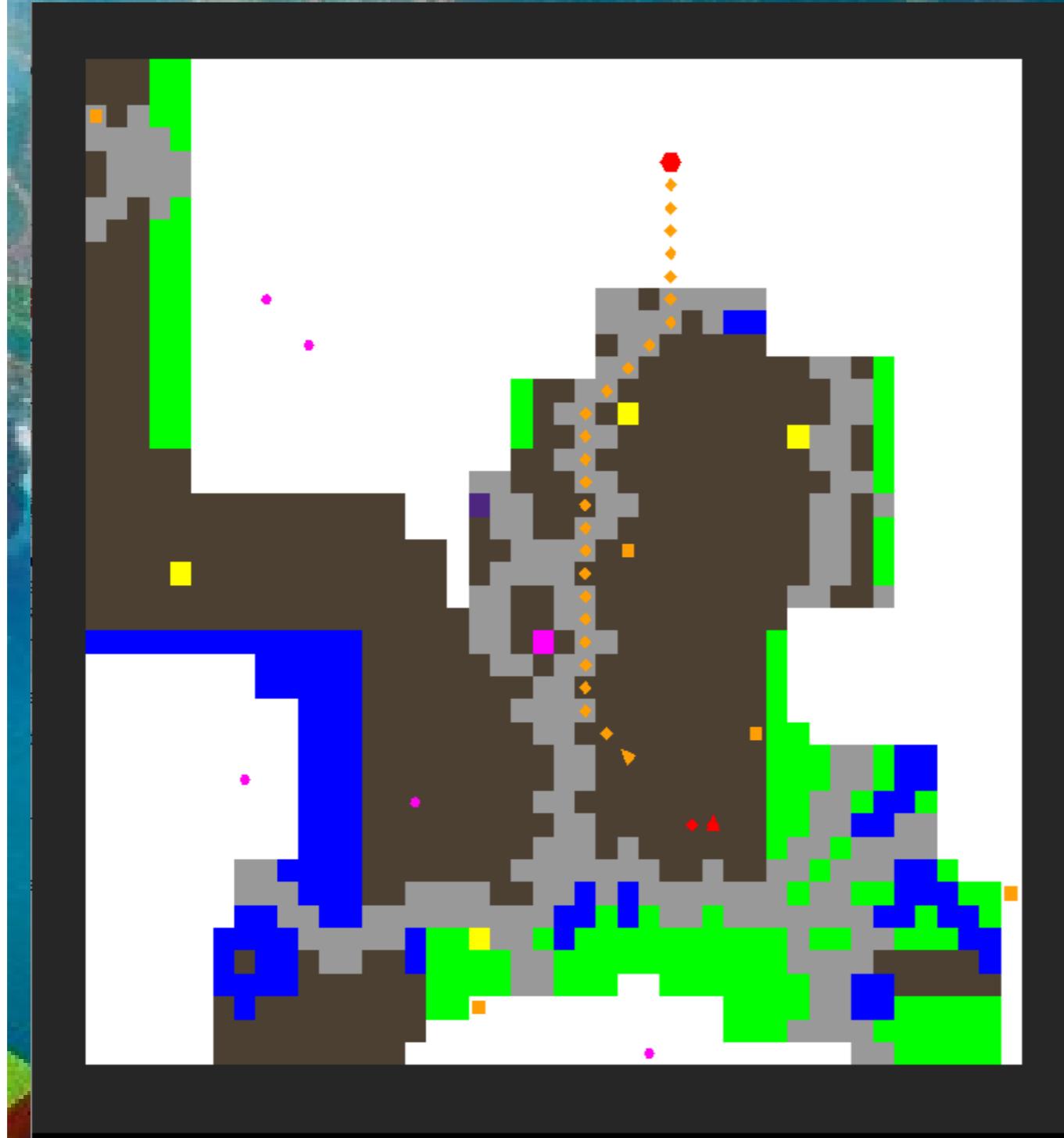
# Elementos inmutables del terreno



'B'	Árboles
'A'	Agua
'P'	Precipicios
'S'	Suelo pedregoso
'T'	Suelo arenoso
'M'	Muros
'K'	Bikini
'D'	Zapatillas
'X'	Recarga
'?'	Casillas desconocida

No existe la casilla de posicionamiento 'G' que existía en la práctica 1

# Agentes con movilidad sobre el terreno



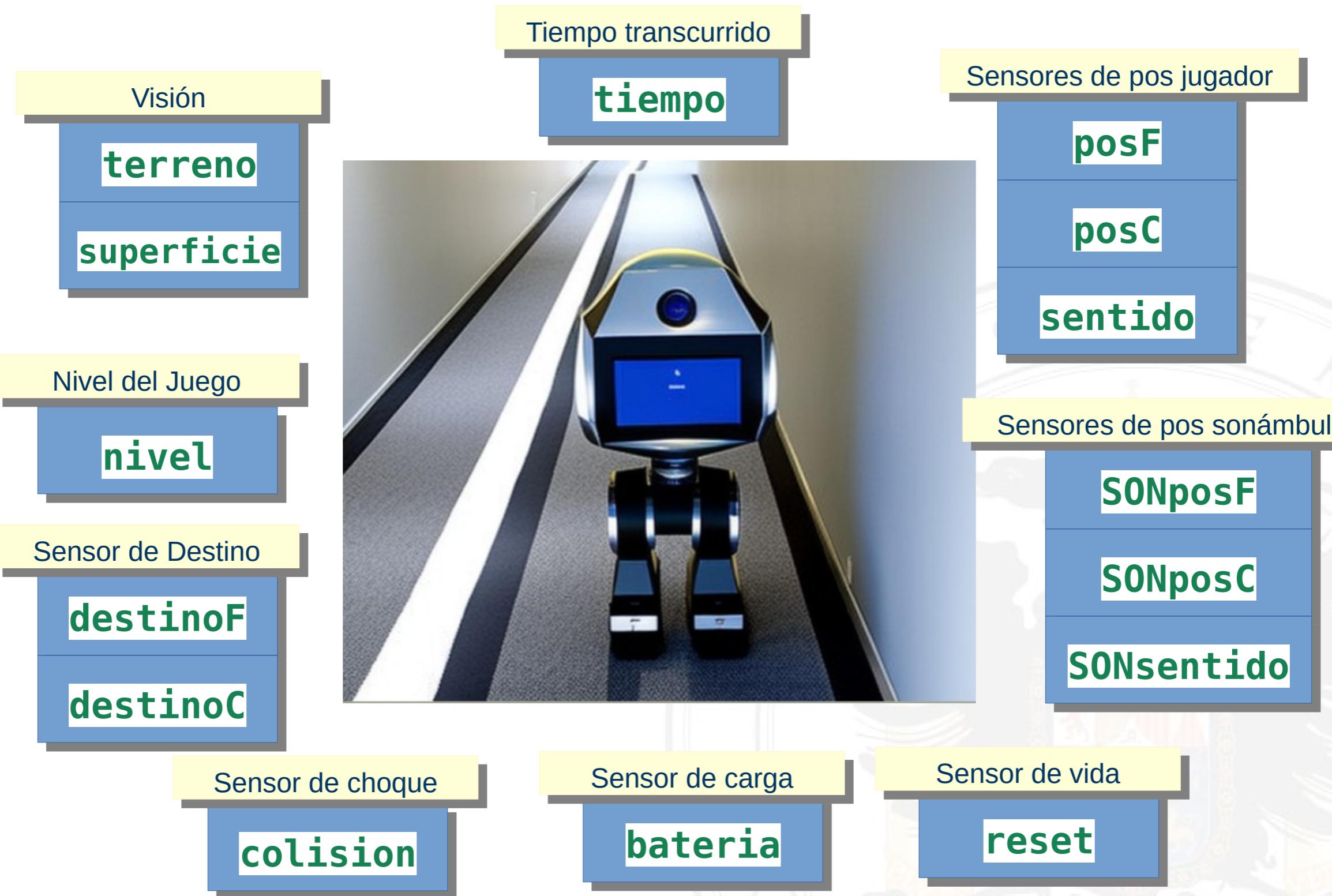
▲	'j'	Jugador
▲	's'	Sonámbulo
■	'a'	Aldeano
●	'l'	Lobo
—	'_'	Casilla desocupada
●		Objetivo



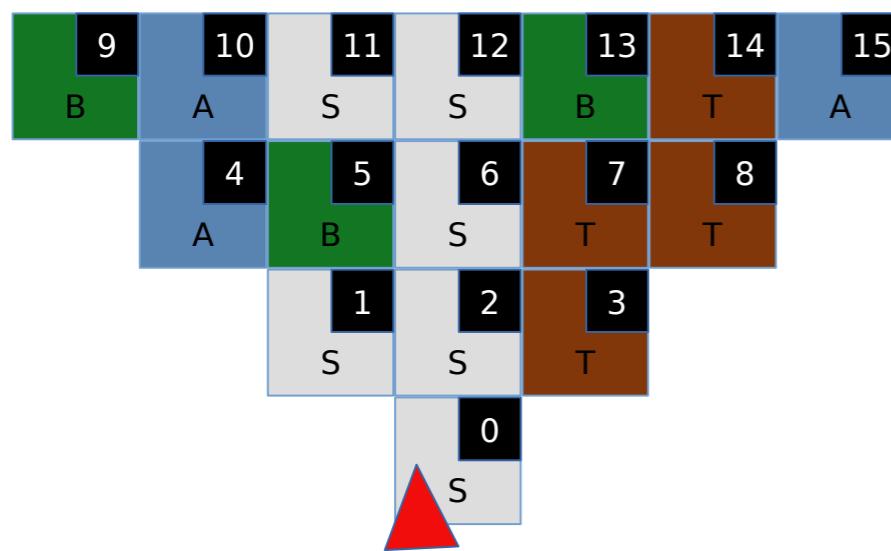
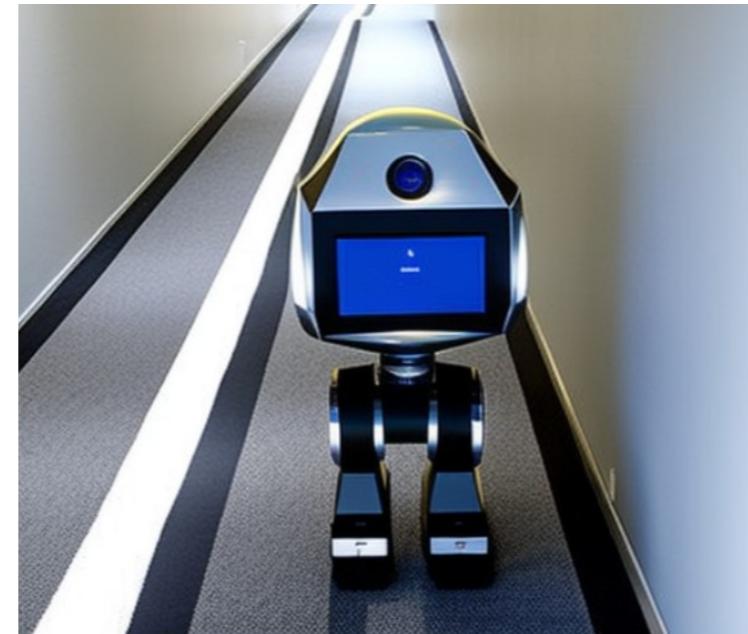
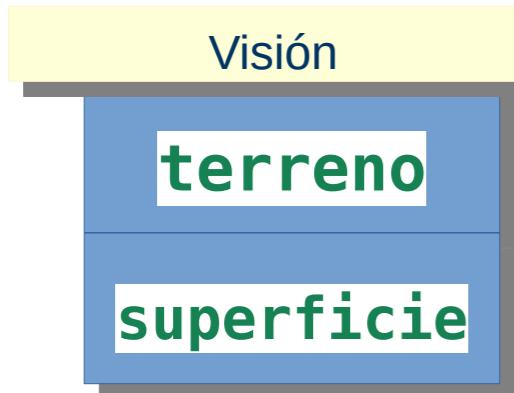
## Características del juego

- Hay dos agentes involucrados: el agente jugador y el agente sonámbulo.
- Se consideran 8 orientaciones: **norte, este, sur y oeste**, pero también las de **noreste, sureste, suroeste y noroeste**.
- Hay nuevos sensores: para indicar las coordenadas de la casilla objetivo y para conocer la posición y orientación del otro agente.
- Uno de los agentes solo puede hacer giros de 45°, mientras que el otro solo puede hacerlos de 90°.

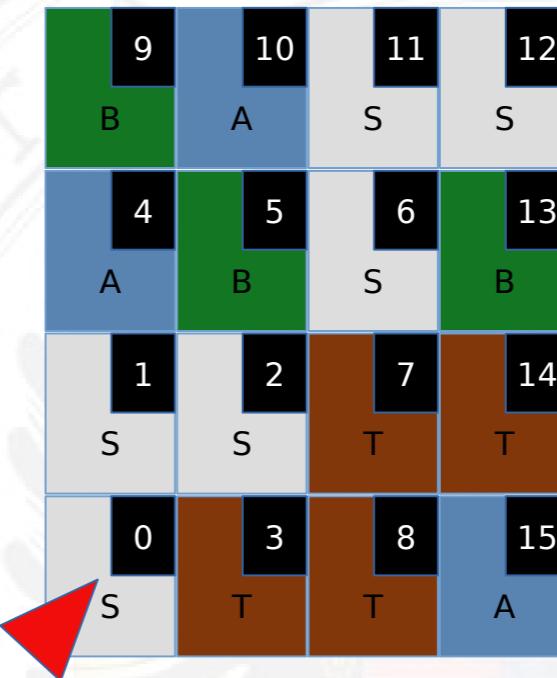
# Sistema sensorial



# Sistema sensorial



Estructura de la visión en las orientaciones norte, este, sur y oeste



Estructura de la visión en las orientaciones noreste, sureste, suroeste y noroeste

# Sistema sensorial

Agente Jugador choca contra un muro o contra otro agente.

El lobo le da un empujón al agente Jugador.

Agente Sonámbulo choca contra un muro.

Agente Sonámbulo choca contra otro agente  
**Se reinicia en otro punto del mapa.**

Agente Jugador o Sonámbulo caen en un precipicio:  
**Termina la simulación.**

Sensor de choque

**colision**

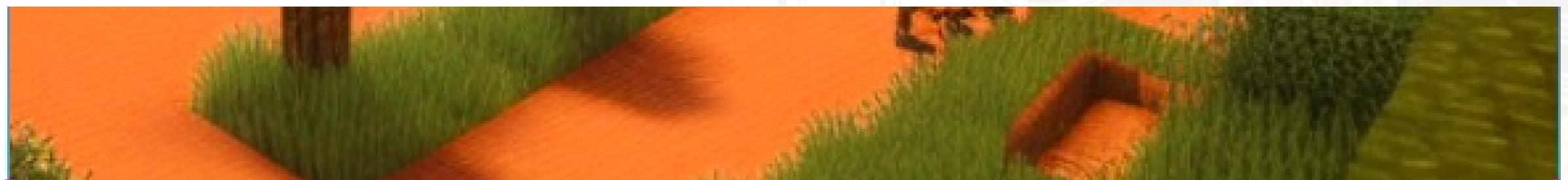
Sensor de vida

**reset**



# Conjunto de Acciones

- *actFORWARD*: le permite al agente **jugador** 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.
- *actTURN\_L*: le permite al agente **jugador** mantenerse en la misma casilla y girar a la izquierda 90º teniendo en cuenta su orientación
- *actTURN\_R*: le permite al agente **jugador** mantenerse en la misma casilla y girar a la derecha 90º teniendo en cuenta su orientación.



# Conjunto de Acciones

- *actSON\_FORWARD*: le permite al agente **jugador** ordenar al agente **sonámbulo** que avance a la siguiente casilla en el sentido que tiene el agente sonámbulo. Para que la acción se realice con éxito es necesario que la casilla destino sea transitable y esté desocupada y que el agente jugador tenga en su sensor de superficie al agente sonámbulo.
- *actSON\_TURN\_SL*: le permite al agente **jugador** ordenar al agente **sonámbulo** que gire la izquierda  $45^\circ$  teniendo en cuenta su orientación. Para que la operación se realice con éxito es necesario que el agente jugador tenga en su sensor de superficie al agente sonámbulo.
- *actSON\_TURN\_SR*: le permite al agente **jugador** ordenar al agente **sonámbulo** que gire la derecha  $45^\circ$  teniendo en cuenta su orientación. Para que la operación se realice con éxito es necesario que el agente jugador tenga en su sensor de superficie al agente sonámbulo.

# Conjunto de Acciones

- *actWHEREIS*: sitúa en los sensores posF, posC y sentido del agente **jugador** y en los sensores SONposF, SONposC y SONsentido del agente **sonámbulo** los valores correctos de fila, columna y orientación en el instante siguiente de simulación. (solo es útil en el último nivel de la práctica).
- *actIDLE*: pues como su nombre indica, no hace nada.





- Cada acción realizada por el agente tiene un **coste** en **tiempo de simulación** y en **consumo de batería**.
- 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 jugador.**
  - Cada simulación conlleva 3000 instantes de tiempo.
  - Cada acción consume 1 instante de tiempo.

## •Consumo de Batería

- actIDLE** consume 0 puntos de batería
- ActWHEREIS** consume 200 puntos de batería

actFORWARD		
Casilla	Gasto Normal	Gasto Reducido
'A'	100	10 (con <b>Bikini</b> )
'B'	50	15 (con <b>Zapatillas</b> )
'T'	2	2
Resto	1	1

actSON_FORWARD		
Casilla	Gasto Normal	Gasto Reducido
'A'	100	10 (con <b>Bikini</b> )
'B'	50	15 (con <b>Zapatillas</b> )
'T'	2	2
Resto	1	1

actTURN_L / actTURN_R		
Casilla	Gasto Normal	Gasto Reducido
'A'	25	5 (con <b>Bikini</b> )
'B'	5	1 (con <b>Zapatillas</b> )
'T'	2	2
Resto	1	1

actSON_TURN_SL / actSON_TURN_SR		
Casilla	Gasto Normal	Gasto Reducido
'A'	7	2 (con <b>Bikini</b> )
'B'	3	1 (con <b>Zapatillas</b> )
'T'	1	1
Resto	1	1

**Los muros y precipicios no son transitables!**

No hay sensor que indique si disponemos del bikini o de las zapatillas, por lo que debemos crear las variables de estado correspondientes.



- Introducción
- Breve descripción del juego
- **Objetivos**
- Software
- Método de evaluación/entrega de prácticas





# Objetivos

- El objetivo de esta práctica es dotar de un comportamiento inteligente al agente jugador que permita ir resolviendo los niveles de dificultad que se han fijado en la práctica.
- La idea es encontrar caminos de forma inteligente dentro de un mapa (pidiendo ciertas optimalidades o planificando con falta de información) bien para el agente jugador o bien para el agente sonámbulo).
- Se han considerado cinco niveles o tareas a realizar en esta práctica.



**NIVEL 0: Encontrar el camino con el mínimo número de acciones que lleve al agente **JUGADOR** a una casilla objetivo**

**NIVEL 1: Encontrar el camino con el mínimo número de acciones que lleve al agente **SONÁMBULO** a una casilla objetivo**

**NIVEL 2: Encontrar el camino con el mínimo consumo de batería usando el ALGORITMO DE DIJKSTRA (coste uniforme) que lleve al agente **JUGADOR** a una casilla objetivo**

**NIVEL 3: Encontrar el camino con el mínimo consumo de batería usando el ALGORITMO A\* que lleve al agente **SONÁMBULO** a una casilla objetivo.**

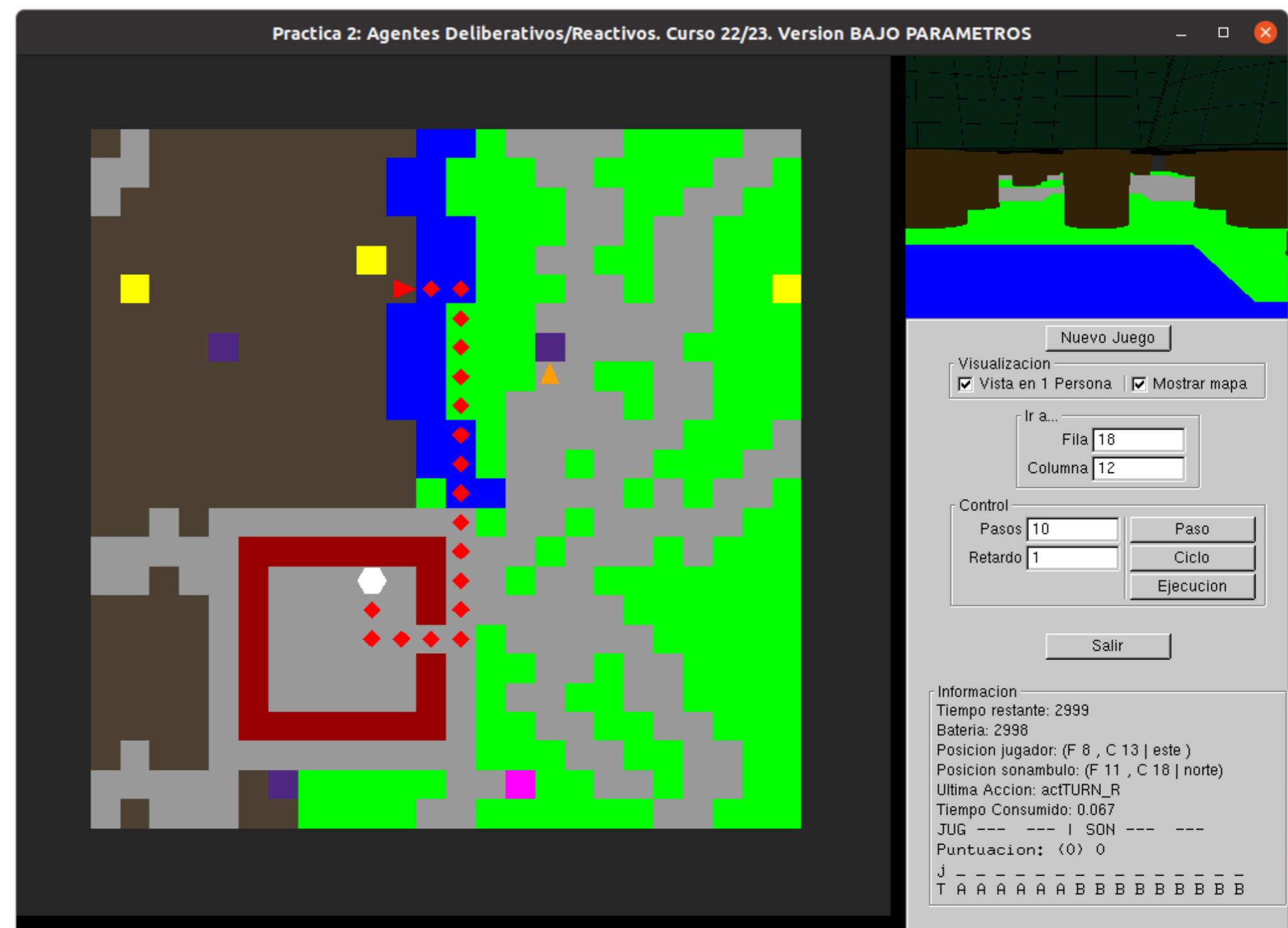
**NIVEL 4: Reto (Maximizar la puntuación en los objetivos alcanzados)**



## Nivel 0: Anchura agente jugador

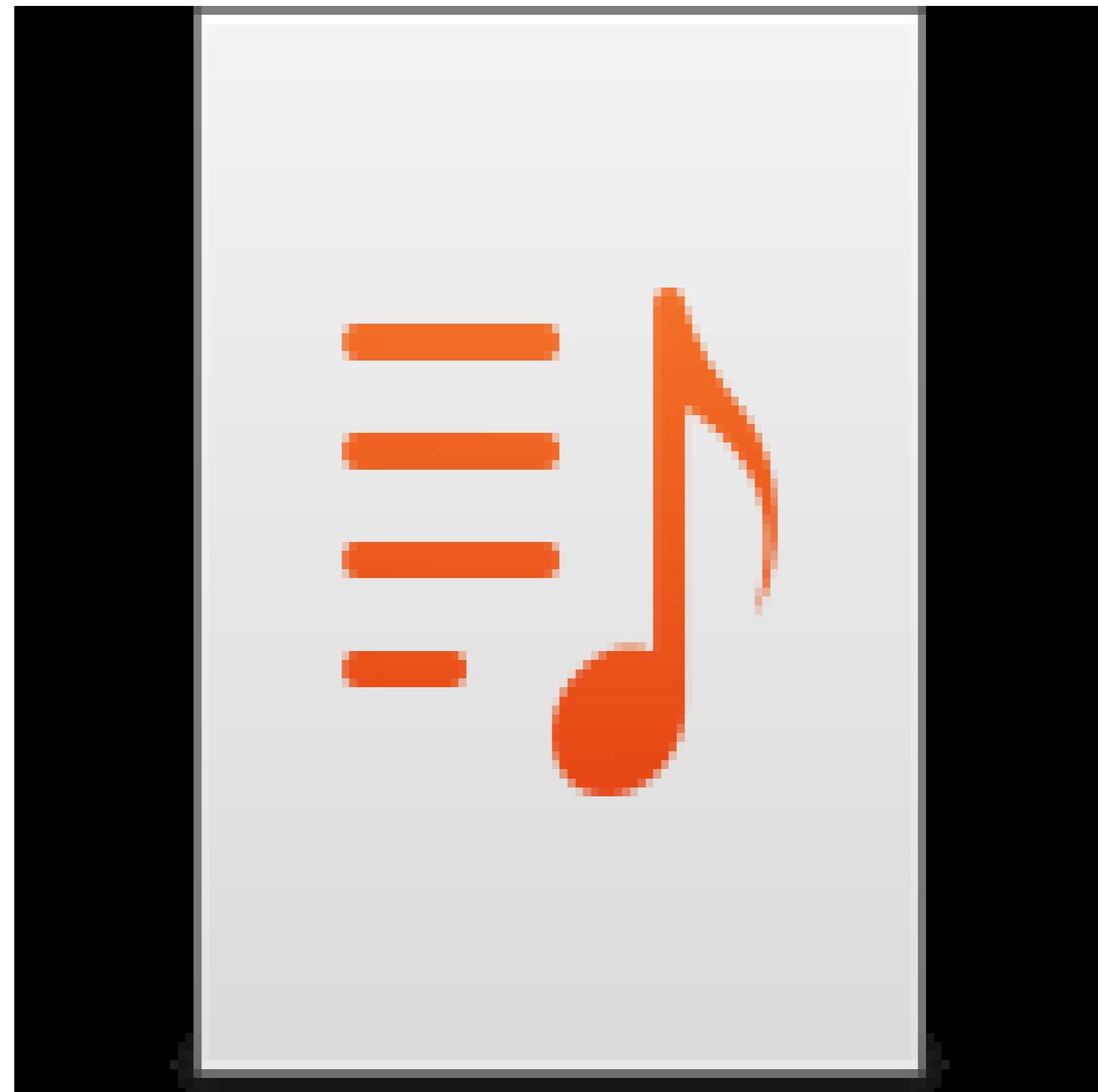
- El desarrollo del nivel 0 se encuentra en el tutorial que se adjunta con el material de la práctica.
- En dicho tutorial se describe la estructura básica del programa principal que es común para los niveles del 0 al 3, así como se resuelve el nivel 0.
- En estos primeros 4 niveles, el mundo es conocido y el sistema sensorial de los dos agentes funciona correctamente.
- El algoritmo que se pide aquí es la búsqueda en anchura para llevar al agente jugador a la casilla destino.

./practica2 mapas/mapa30.map 1 0 8 13 0 11 18 0 18 12





```
./practica2 mapas/mapa30.map 1 0 8 13 0 11 18 0 18 12
```

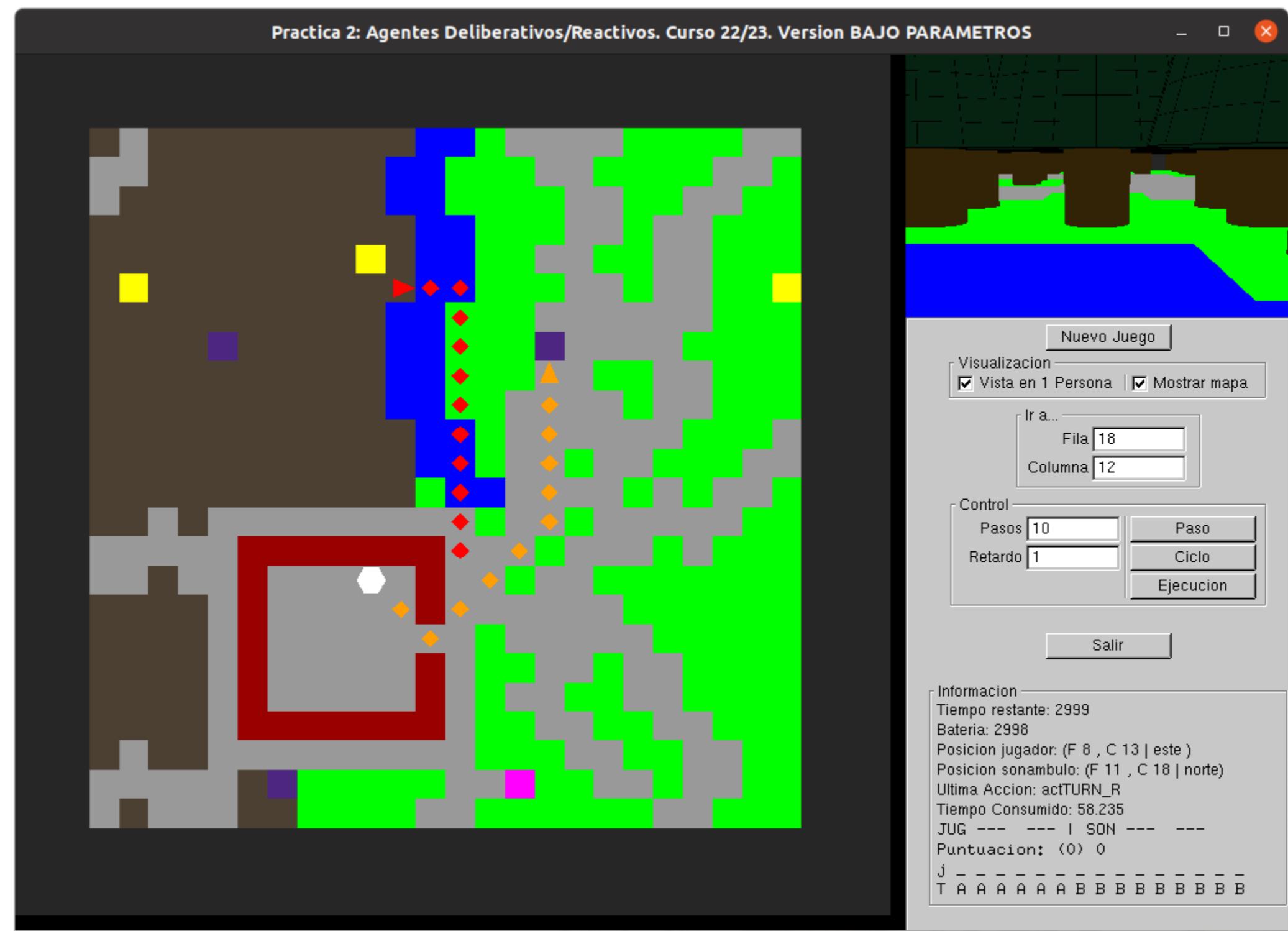




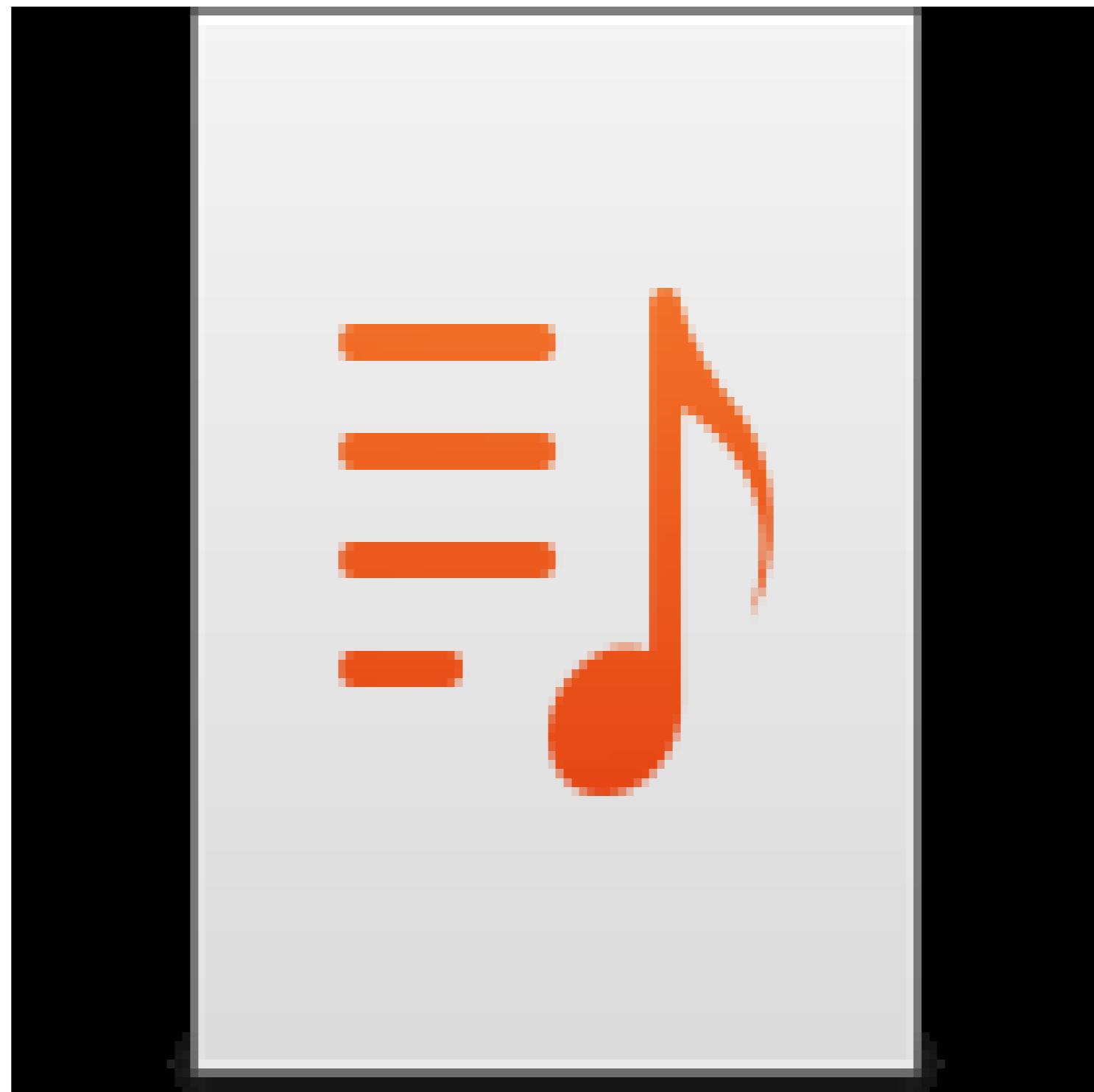
## Nivel 1: Anchura agente sonámbulo

- Se pide implementar el mismo algoritmo del nivel 0 pero en este caso para llevar al agente sonámbulo a la casilla destino.
- El agente sonámbulo no puede moverse por si mismo, así que el plan debe ser trazado por el agente jugador.
- El nivel 1 se puede considerar como una readaptación del nivel 0 para un problema en el que el concepto de estado cambia.

./practica2 mapas/mapa30.map 1 1 8 13 0 11 18 0 18 12



./practica2 mapas/mapa30.map 1 1 8 13 0 11 18 0 18 12



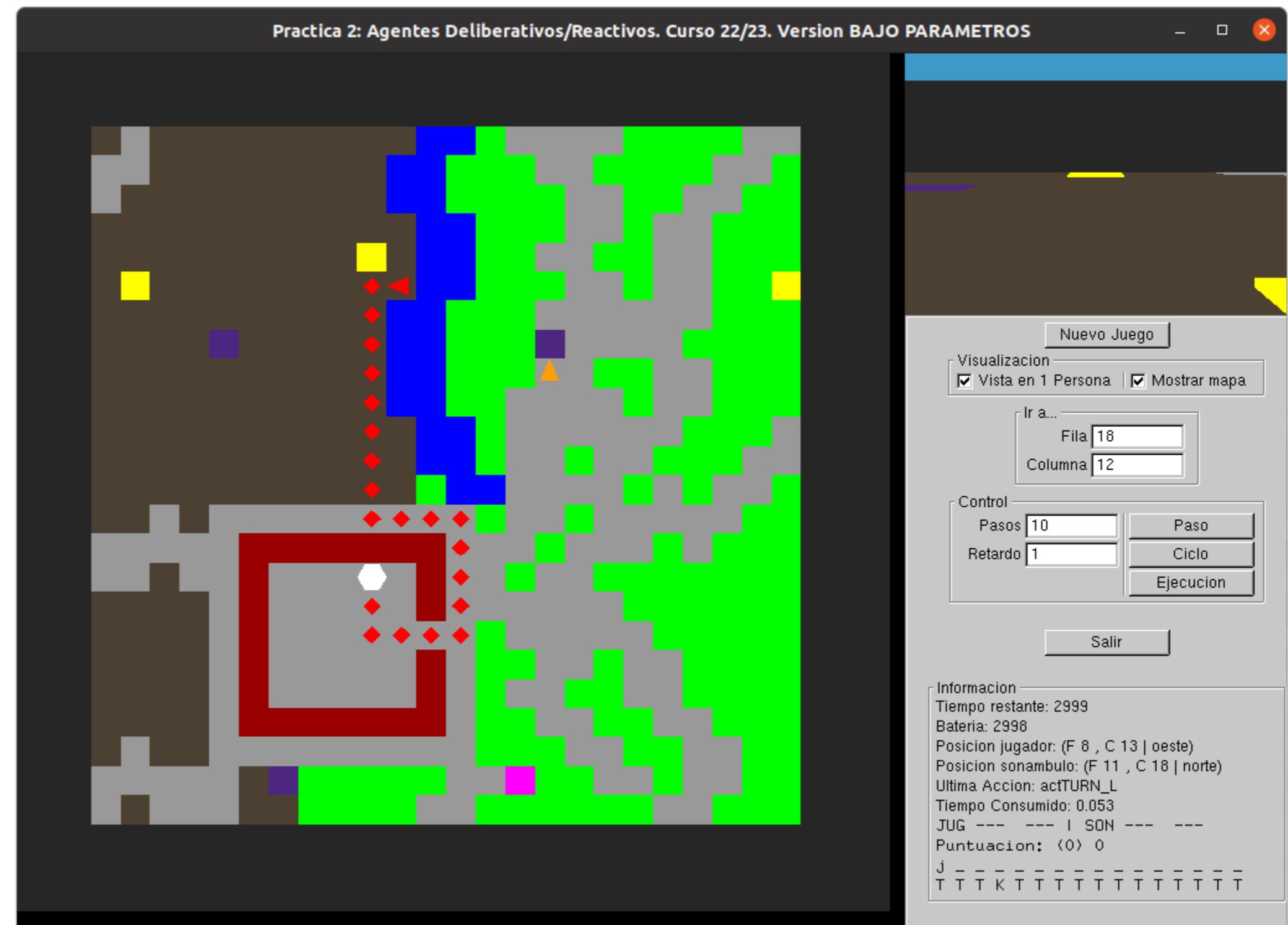


## Nivel 2: Dijkstra agente jugador

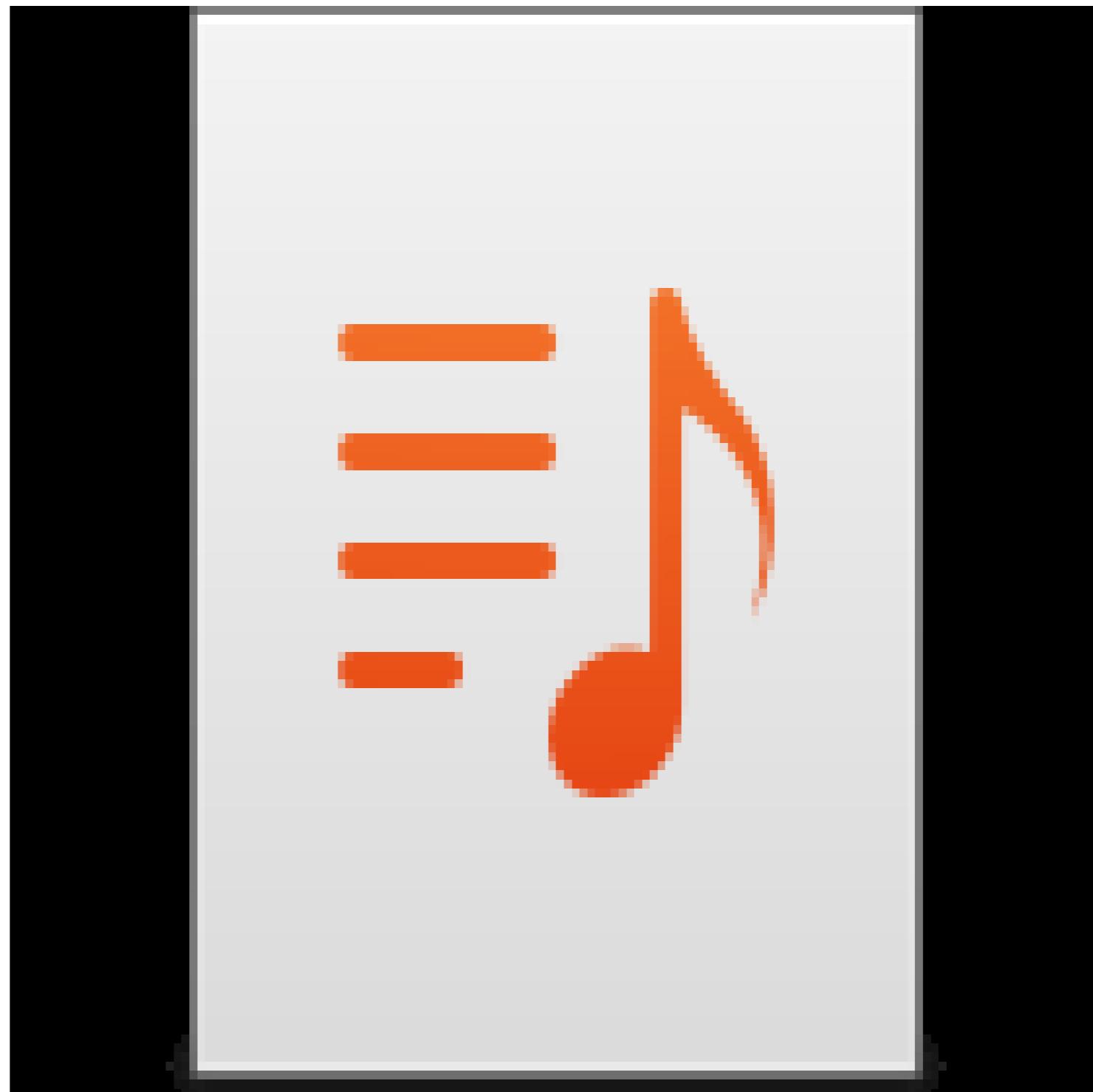
- En este nivel se pide el algoritmo de Dijkstra, también conocido como algoritmo de coste uniforme.
- En este caso, vuelve a ser el agente jugador el que debe ser trasladado a la casilla destino.
- A diferencia de los dos niveles anteriores, a partir de este nivel es relevante el coste de las acciones.



./practica2 mapas/mapa30.map 1 2 8 13 0 11 18 0 18 12



./practica2 mapas/mapa30.map 1 2 8 13 0 11 18 0 18 12



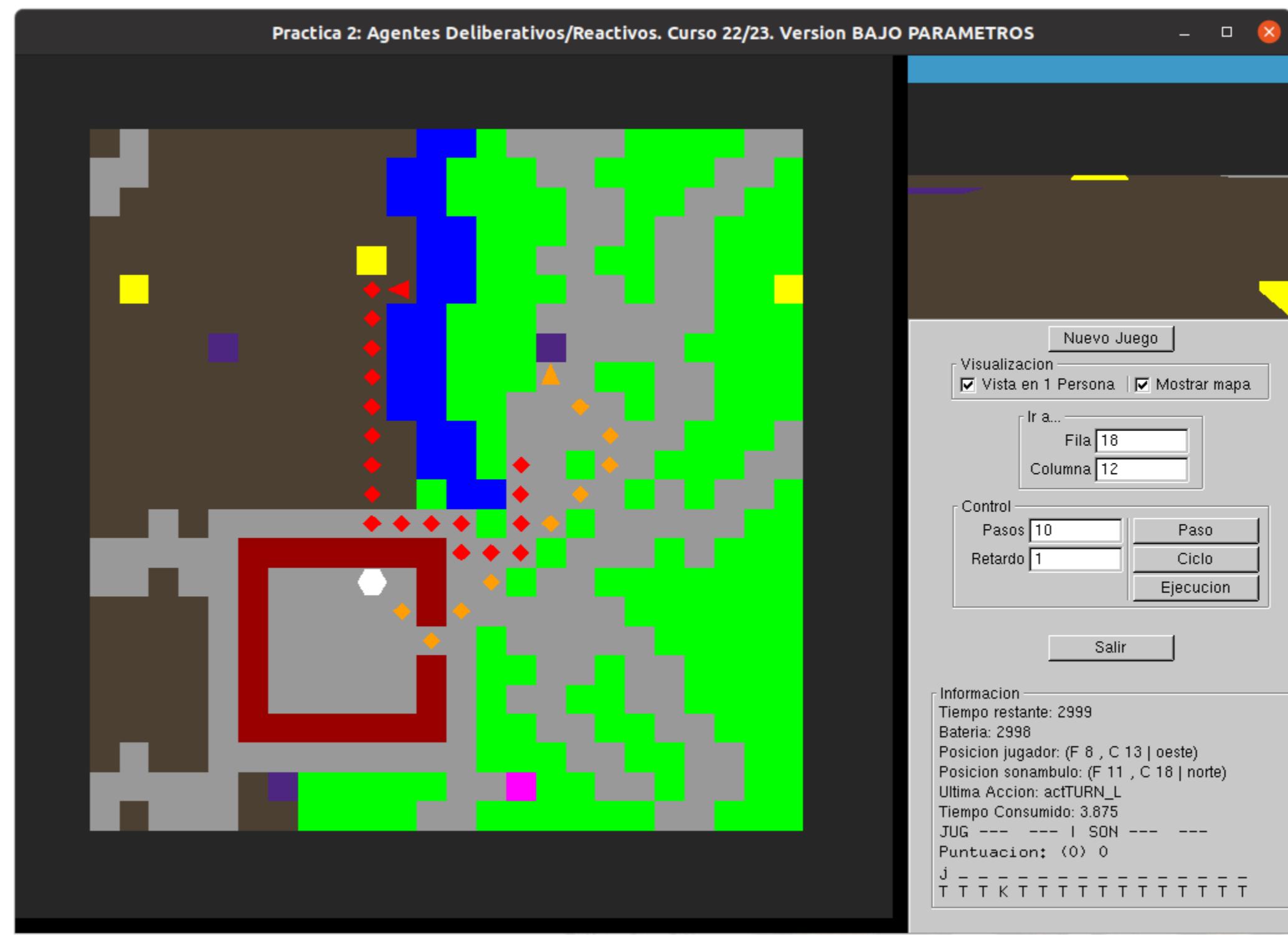


## Nivel 3: A\* agente sonámbulo

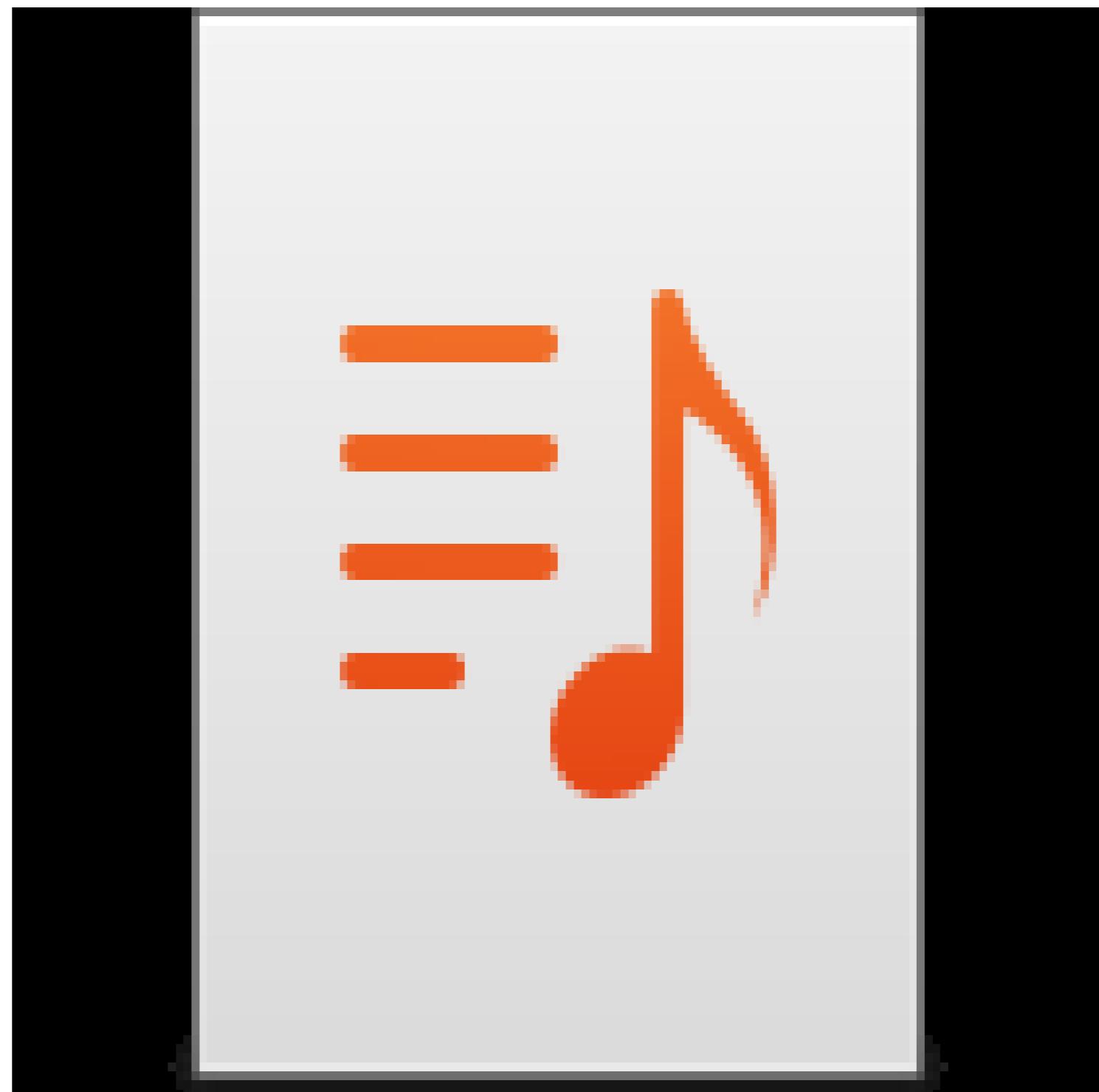
- En este nivel se pide el algoritmo de A\* para llevar al agente sonámbulo a la casilla destino.
- Se pide que la solución propuesta sea óptima en consumo de batería, por consiguiente se debe definir una heurística que permita obtener este tipo de soluciones.



```
./practica2 mapas/mapa30.map 1 3 8 13 0 11 18 0 18 12
```



./practica2 mapas/mapa30.map 1 3 8 13 0 11 18 0 18 12





## Nivel 4: Reto

- Este nivel es diferente a los anteriores y es el que realmente se puede considerar como un juego.
- El agente no conoce el mapa ni su posición, ni su orientación, ni la posición y orientación del sonámbulo (estos sensores solo funcionarán cuando se usa la acción `actWHEREIS`).
- Se irán proponiendo nuevas casillas objetivo conforme se vayan consiguiendo. El agente debe ir proponiendo planes para él o para el sonámbulo.



## Nivel 4: Reto

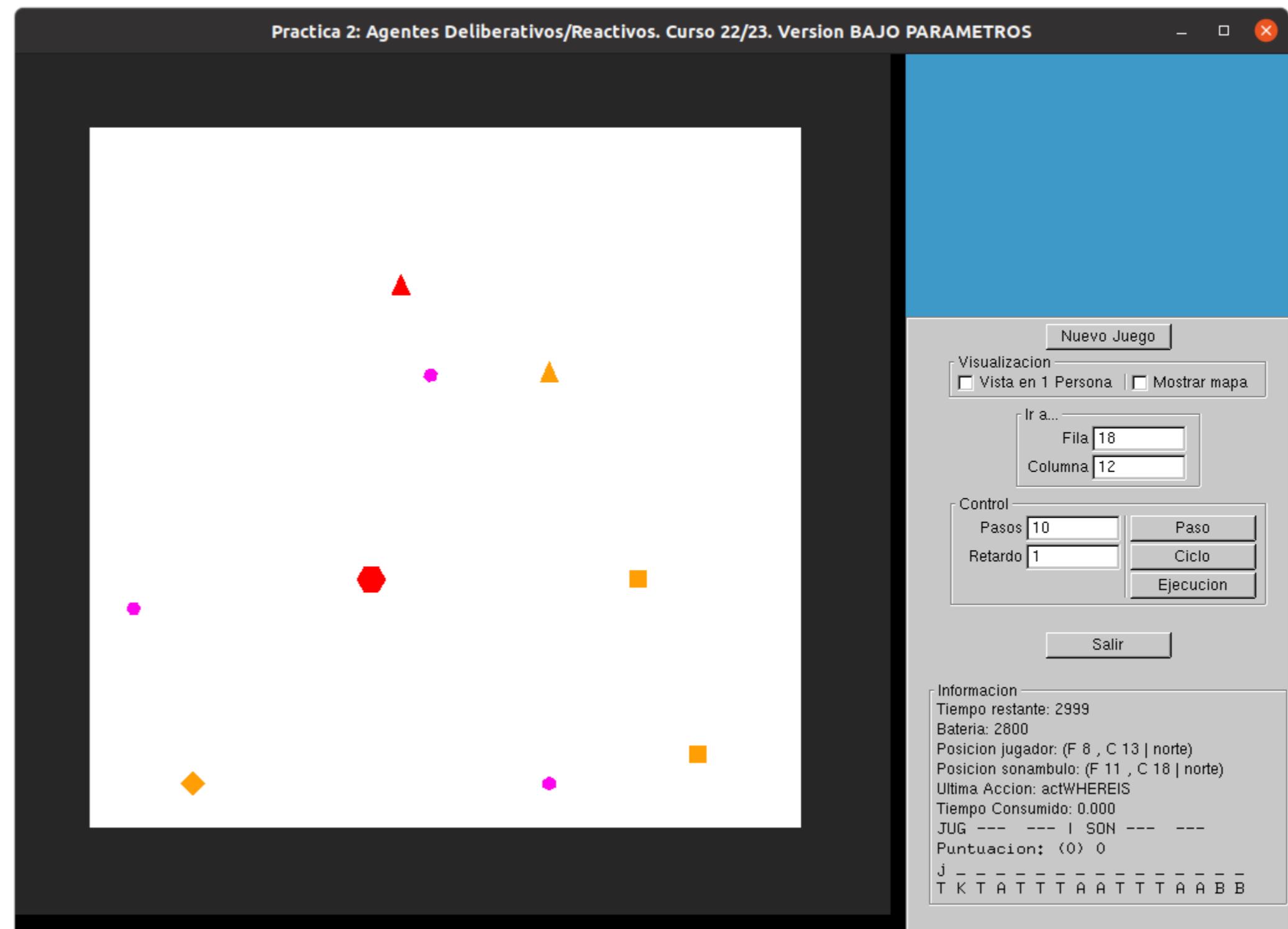
- En este nivel además puede haber aldeanos y lobos. Estos agentes con movilidad sobre el mapa provocan choques y reinicios del agente jugador o sonámbulo.
- El no conocer inicialmente el mapa y la presencia de los aldeanos y lobos hace que el sistema sea dinámico y posiblemente los planes no terminen con éxito por lo que es necesario replanificar.
- El no tener información del sensor de superficie y terreno del agente sonámbulo también implica una dificultad cuando el mapa no es conocido.



## Nivel 4: Reto

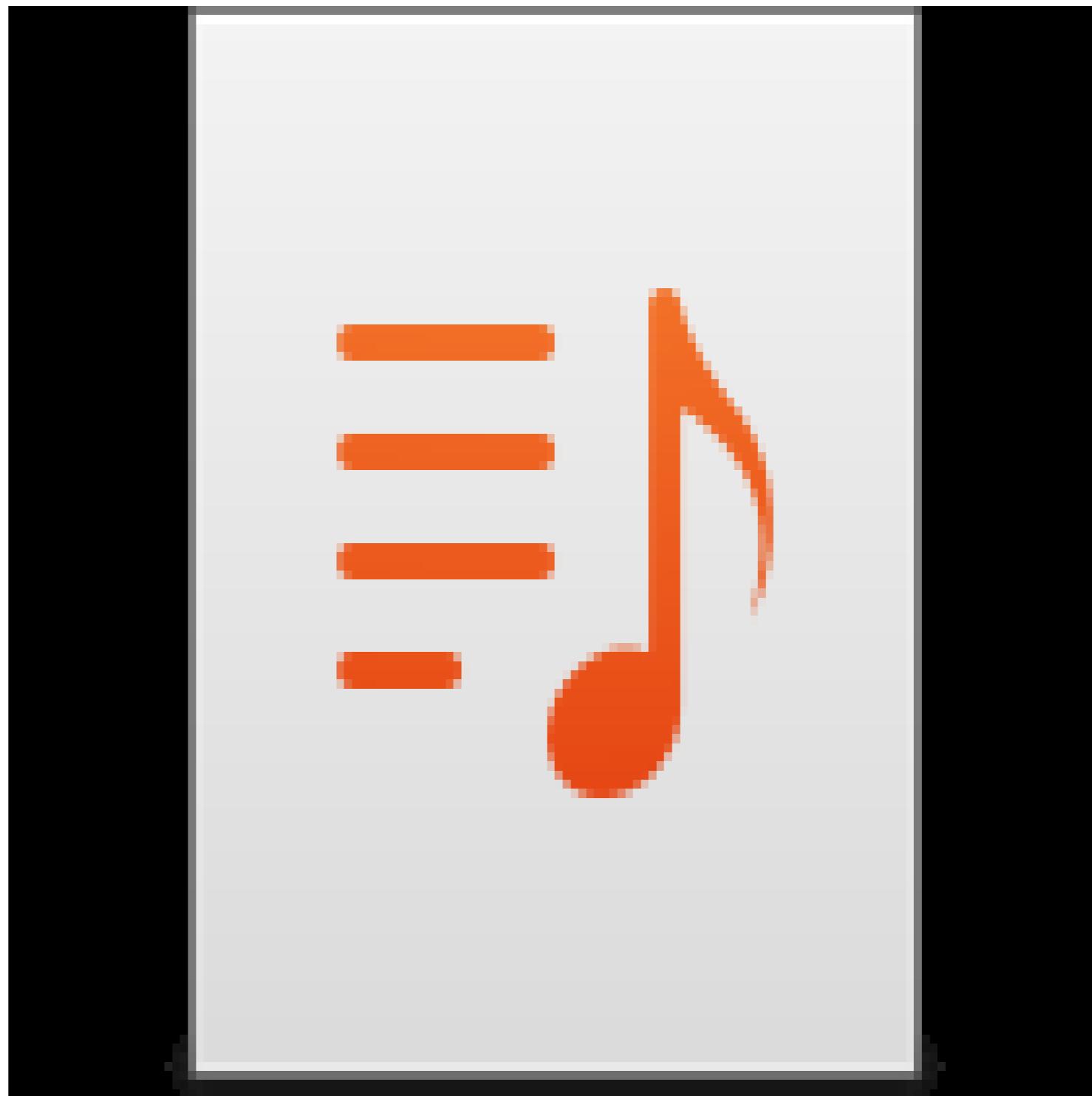
- Por consiguiente la idea de este nivel es definir un comportamiento inteligente con la idea de maximizar la puntuación en las misiones realizadas combinando comportamientos reactivos y deliberativos.
- El juego durará una simulación. Cada simulación empieza con 3000 instantes de tiempo, 3000 puntos de batería y 300 segundos para pensar. La simulación termina cuando alguno de ellos se agota.
- Las misiones que culmina el sonámbulo valen 10 puntos. Las del agente jugador 1 punto.

./practica2 mapas/mapa30.map 1 4 8 13 0 11 18 0 18 12





```
./practica2 mapas/mapa30.map 1 4 8 13 0 11 18 0 18 12
```



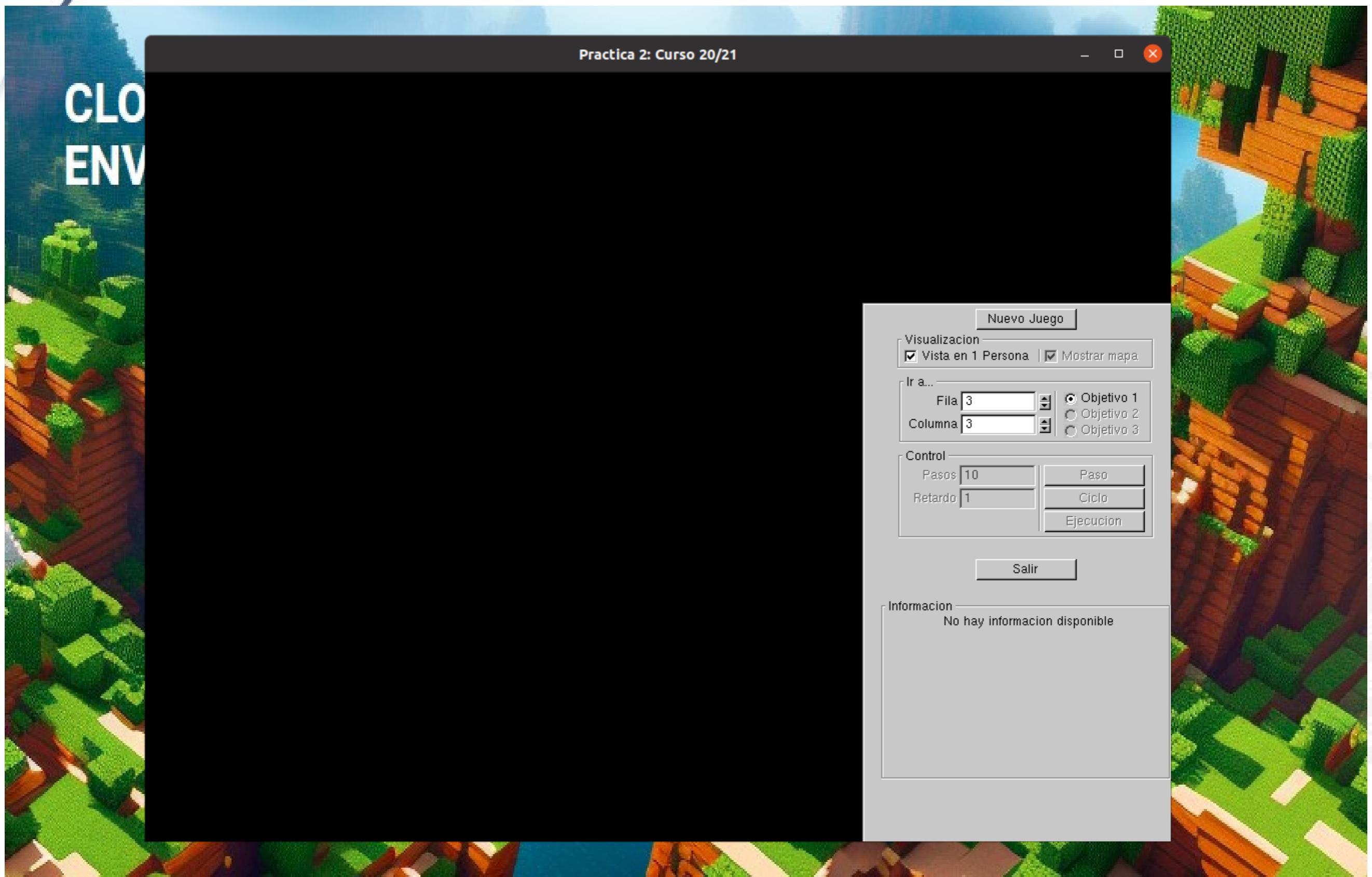


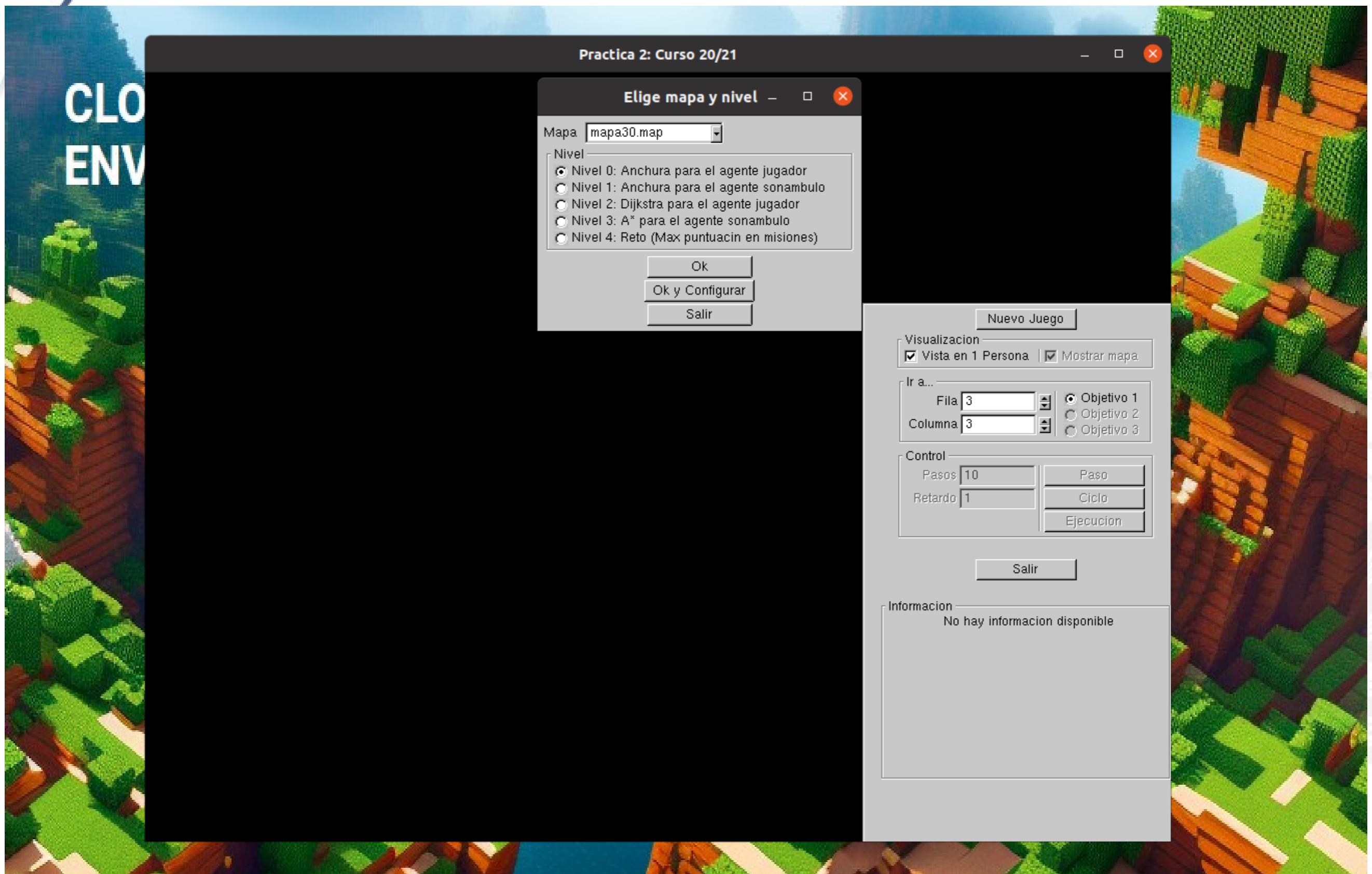
- Introducción
- Los extraños mundos de Belkan
- Objetivo
- **Software**
- Método de evaluación y entrega de prácticas

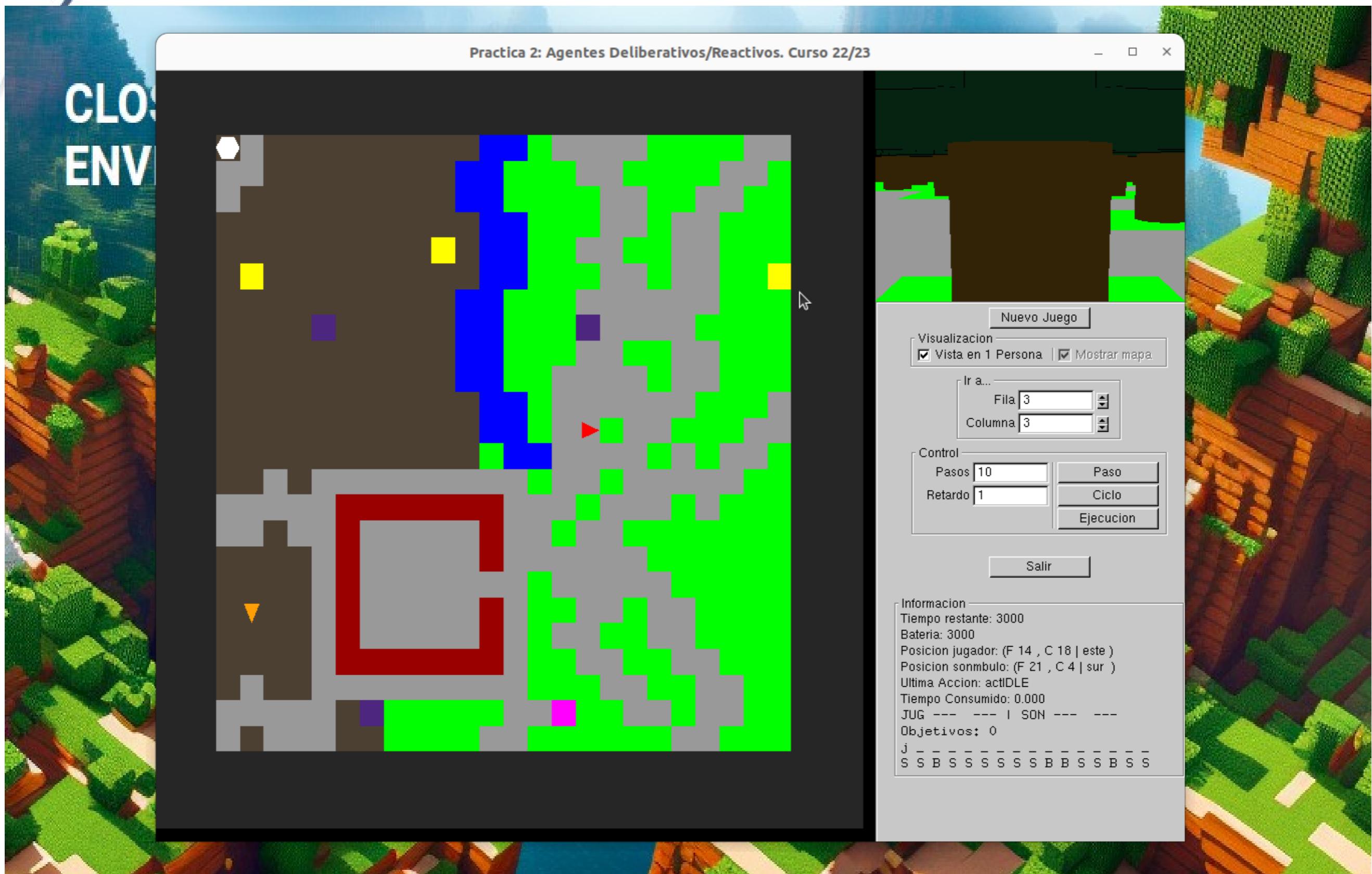


Se proporciona el software para el sistema operativo Linux en el repositorio de GitHub  
<https://github.com/ugr-ccia-IA/practica2>

- Existen dos versiones del software: `practica2` y `practica2SG`. El primero corresponde al simulador con interfaz gráfica, mientras que el segundo es un simulador en modo *batch* sin interfaz.
- Todos los detalles y explicación de la instalación, uso y detalles las variables necesarias para su desarrollo se encuentra en el guion de prácticas asociado a esta presentación y en el repositorio de GitHub (en la parte de instalación).







# Sistema *batch*

**./practica2 mapas/mapa30.map 1 0 4 5 1 7 7 3 12 5**

- fichero de mapa
- semilla generador de números aleatorios
- Nivel (0, 1, 2, 3 o 4)
- Fila, columna y orientación del agente jugador
- Fila, columna y orientación del agente sonámbulo
- pares de (fila, columna) destino

En caso del nivel 4, si se queda sin destinos, los generará al azar.

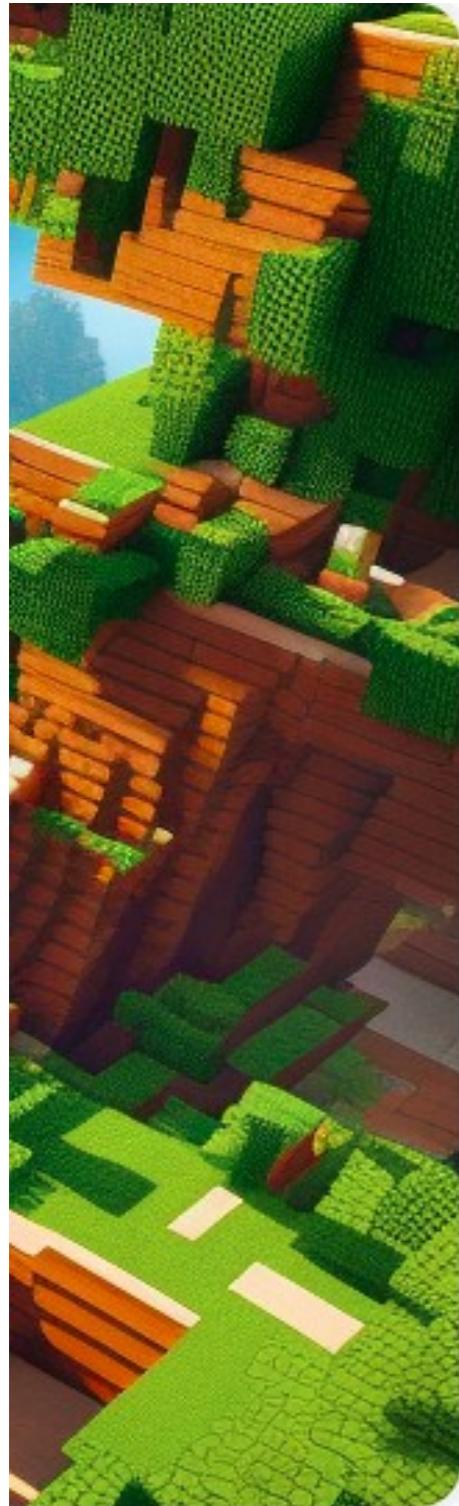
Se incluye esta opción para acelerar la ejecución completa de una simulación por un lado y por otro, para que el entorno gráfico no sea un inconveniente a la hora de depurar errores.

La orientación se codifica en la llamada como un número entre 0 y 7, siendo 0 norte y el resto de valores se distribuyen en el sentido de las agujas del reloj.

## CLOSED WORLD

Al finalizar la ejecución, se nos proporciona el tiempo consumido, la cantidad de batería con la que terminó la simulación, el número de colisiones que hemos sufrido (por obstáculos u otros agentes), la cantidad de reinicios y la cantidad de destinos alcanzados.

```
Mision alcanzada!
Instantes de simulacion no consumidos: 2969
Tiempo Consumido: 0.003062
Nivel Final de Bateria: 1374
Colisiones: 0
Muertes: 0
Objetivos encontrados: 1
```



- Introducción
- Los extraños mundos de Belkan
- Objetivo
- Software
- **Método de evaluación y entrega de prácticas**





# Entrega

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 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 dos ficheros con la solución del alumno.



# 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.

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 los niveles presentados por el estudiantes superados.

# Método de Evaluación

Nivel	Puntuación	Requisito
0	0	Terminar el nivel 0
1	2	Tener correcto el nivel 0
2	3	Tener correcto el nivel 1 y 0
3	2	Tener correcto el nivel 2, 1 y 0
4	3	Tener correcto el nivel 2, 1 y 0

Los niveles del 0 al 3 se evalúan si se satisfacen las condiciones que se le exigen a cada nivel.

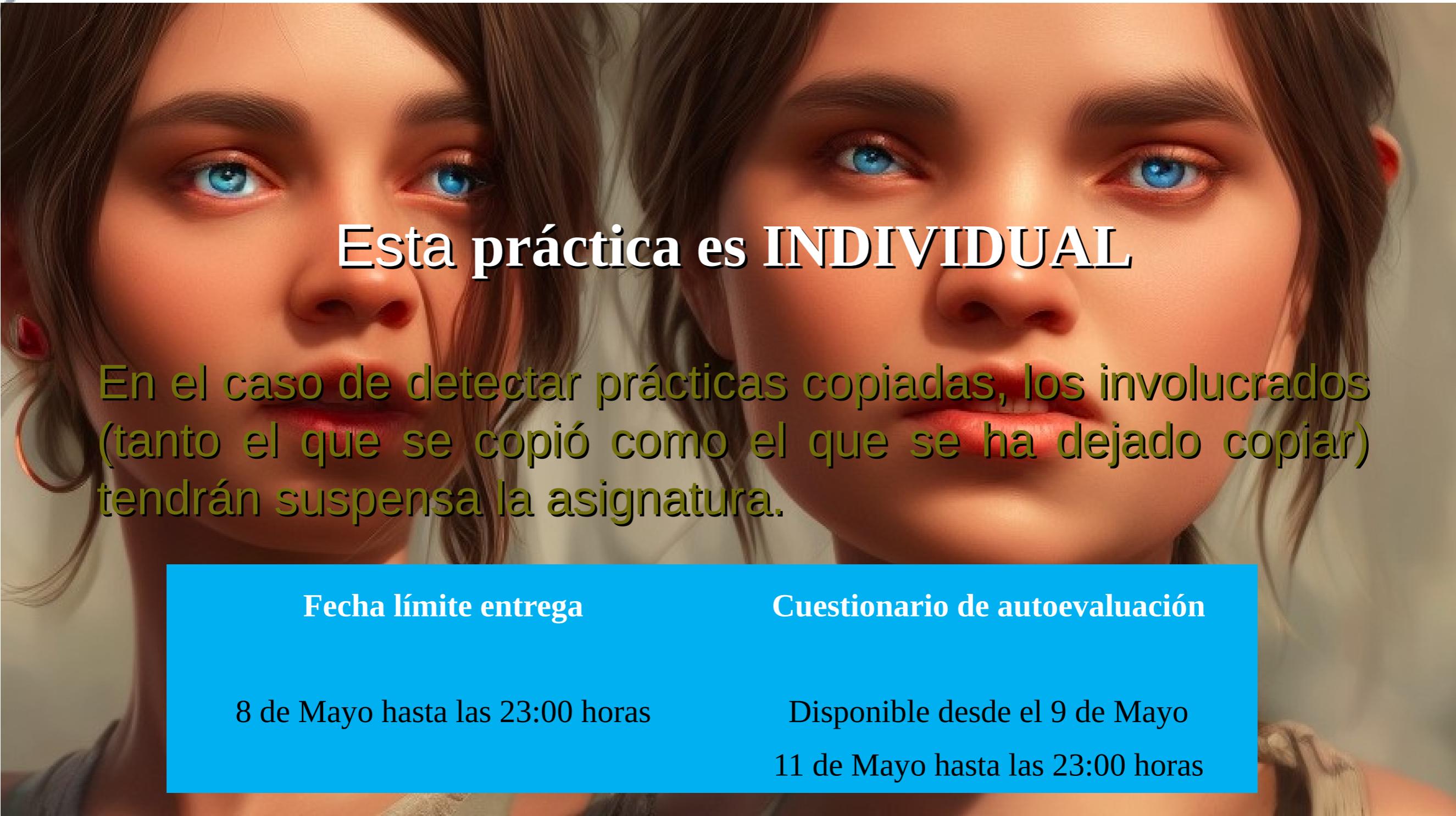
Para que el nivel 3 sea evaluable es necesario haber hecho los niveles 0, 1 y 2.

Para el nivel 4 no es necesario entregar o que esté correcto el nivel 3, pero si todos los niveles anteriores.



# Método de Evaluación

- El nivel 4 puede usar uno de los algoritmos de búsqueda ya implementados o bien puede definir uno distinto.
- El objetivo es maximizar la puntuación en las misiones en el tiempo, batería e instantes de simulación que se ofrecen.
- Para valorar este nivel se realizarán pruebas sobre distintos mapas con distintas configuraciones de posiciones iniciales y de listas de objetivos. En base al resultado de esas pruebas se otorgará una calificación entre 0 y 3 puntos.
- La nota final se calculará sumando los puntos obtenidos en cada nivel, teniendo en cuenta las restricciones descritas en la tabla.



## Esta práctica es INDIVIDUAL

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.

### Fecha límite entrega

8 de Mayo hasta las 23:00 horas

### Cuestionario de autoevaluación

Disponible desde el 9 de Mayo  
11 de Mayo hasta las 23:00 horas