INTELIGENCIA ARTIFICIAL

CURSO 2024-25

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

APELLIDOS Y NOMBRE	Linari Perez Pablo		
GRUPO TEORÍA	DGIIM	**GRUPO PRÁCTICAS**	A2

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).

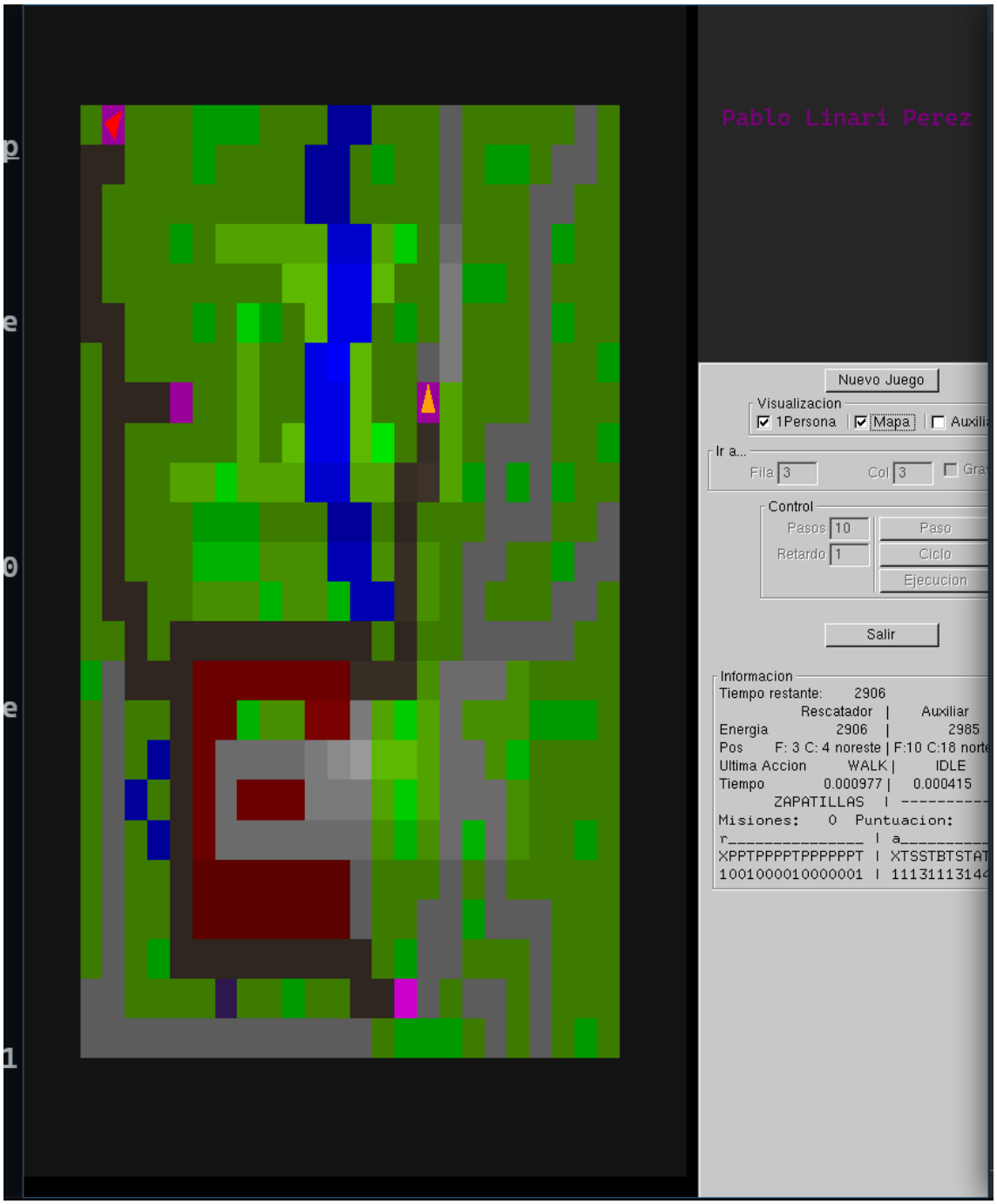
Enumera los niveles presentados en su práctica (Nivel 0, Nivel 1, Nivel 2, Nivel 3, Nivel 4):

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

Nivel 0-El Despertar Reactivo

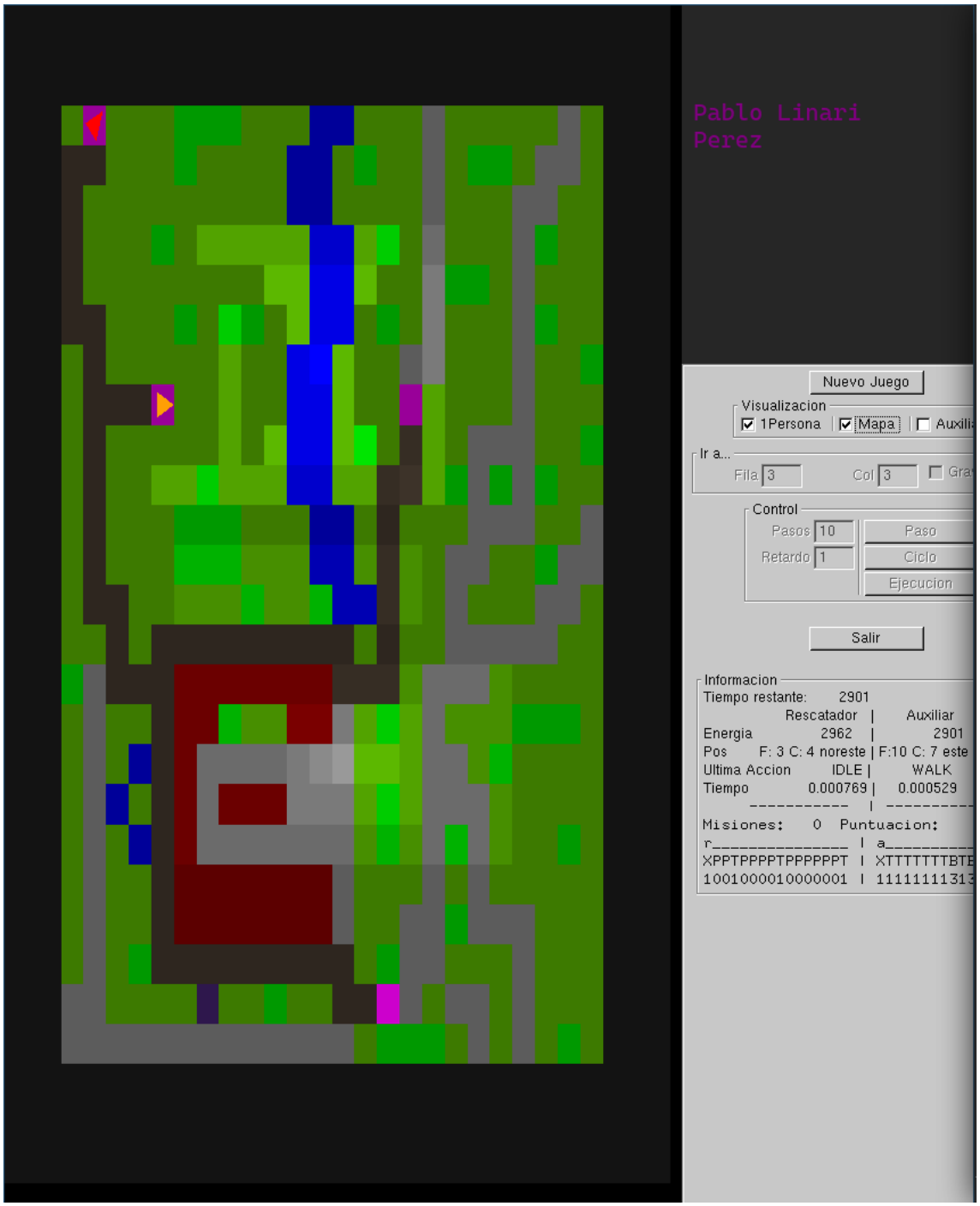
(a) Rellena los datos de la tabla con el resultado de aplicar

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

ScreenShot	
Instantes de simulación no consumidos	2906
Coste de Energía (Rescatador)	94
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	2901
Coste de Energía (Rescatador)	38
Coste de Energía (Auxiliar)	99

(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	2054
Coste de Energía (Rescatador)	114
Coste de Energía (Auxiliar)	946

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)

En ambos agentes he implementado una matriz para saber qué casillas han sido pisadas previamente y cuáles no. En cada posición, el agente comprueba esta información, además de la superficie y la altura, y se dirige a la casilla menos pisada a la que pueda acceder. De esta forma, los agentes siempre intentarán acceder a caminos y senderos nuevos y solo transitarán los caminos ya explorados cuando no quede otra opción.

Cada vez que los agentes entran a una casilla, su valor de 'memoria[posf][posc]' se incrementa en 1, y si gastan mucho tiempo haciendo giros por defecto (en el caso de que no se pueda acceder a ninguna casilla de las que el agente ve en ese instante), en la casilla también se aumenta, ya que a esa casilla no es tan interesante para volver por el coste que supone y los pocos caminos que se abren ante ella.

Cuando los agentes han elegido a qué casilla deben trasladarse, en la función principal del nivel se decide, según el número de casilla, si el movimiento necesita hacer giros extras o walks extras para poder acceder a una casilla por distintos caminos, por ejemplo:
acceder a la casilla 6 por la casilla 1, 2 o 3, dependiendo de cuál sea transitable.

Por último, destacar que en la función que busca la mejor casilla a la que ir se intenta, como primera opción, siempre caminar hacia delante para así ahorrar energía y avanzar más.

He usado ambos agentes ya que no se interfieren entre ellos y, al tener dos agentes recorriendo el mapa, las posibilidades de encontrar caminos nuevos son mayores.

(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	98.0159

(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	79.6999

(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	68.63115

Nivel 2-La Búsqueda del Camino de Dijsktra

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

La definición de estado que he usado incluye 3 datos de tipo int, uno para la brújula y 2 para saber la posición del agente en ese estado (fila, columna) y un dato de tipo bool para saber si tiene o no zapatillas.

También he definido un operador = para asegurar la copia de estados y dos operadores lógicos, el primero '==' para saber cuándo un estado es el mismo que otro y uno '<' para poder ordenar los estados.

Esta estructura, junto con la de Nodo, permite resolver el problema propuesto. El coste de una secuencia de estados no se almacena en el propio estado, sino en otro struct llamado nodo en el cual se guarda el estado, la secuencia de pasos seguidos para llegar hasta él y el coste que han tenido todos esos pasos.

(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.

Sí, primero compruebo si en la posición inicial tengo zapatillas y se lo aplico a mi estado inicial. Una vez en el bucle que aplica el algoritmo, cada vez que genero un hijo, compruebo si ese hijo está en una posición donde hay zapatillas; de ser así, le asigno al valor de zapatillas del estado hijo a true. De esta manera, el estado hijo y los descendientes de dicho hijo tendrán el valor de zapatillas como verdadero y podrán aplicar las acciones que se habilitan al tener zapatillas.

(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.

Sí, lo tuve en cuenta, ya que Dijkstra, dependiendo de la estructura de datos que se use, puede dar una solución que no sea la óptima. Para ello, usé un 'priority_queue' que ordena los nodos, ofreciendo primero los de menor coste. De esta manera, nos aseguramos de que el primer camino que ofrece el algoritmo de Dijkstra es el óptimo, ya que es el que usa todos los nodos con menor coste.

(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.

Este for genera las acciones (RUN,WALK,TURNL,TURNSR)

```
  for (const auto &accion : genera_acciones) {//genera acciones es un vector con dichas acciones
    NodoR hijo = actual;
    hijo.estado = applyR(accion, actual.estado, terreno, altura);
    if (terreno[hijo.estado.f][hijo.estado.c] == 'D') {
      hijo.estado.zapatillas = true;
    }
    int coste_accion =
        CalcularCoste(accion, actual.estado, hijo.estado, terreno, altura);
    hijo.coste = actual.coste + coste_accion;

    auto it = mejor_coste.find(hijo.estado);
    if (it == mejor_coste.end() || it->second > hijo.coste) {
      mejor_coste[hijo.estado] = hijo.coste;
      hijo.secuencia.push_back(accion);
      frontier.push(hijo);
    }
  }
```

Parta generar el hijo Run uso esta funcion para validar que se pueda hacer la accion , esta funcion se usa en la funcion applyR cuando se le pasa como accion RUN , a la hora de calcular la casilla simplemente aplico dos veces nextcasilla :

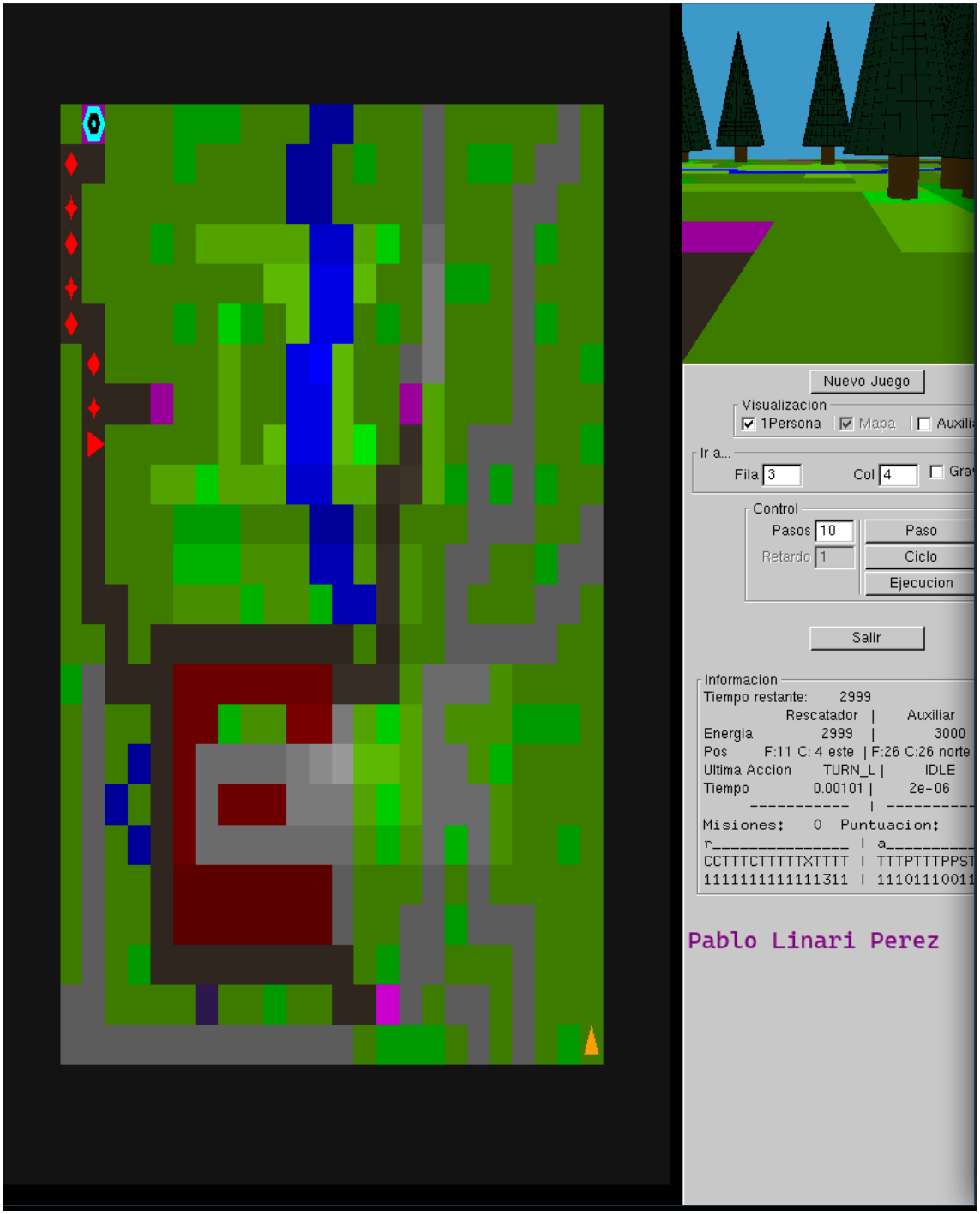
```
bool CasillaAccesibleRunR(const EstadoR &st,
                          const vector<vector<unsigned char>> &terreno,
                          const vector<vector<unsigned char>> &altura) {

  EstadoR intermedio = NextCasillaR(st);
  EstadoR final = NextCasillaR(intermedio);
  bool check1 = false, check2 = false, check3 = false;
  check1 = terreno[intermedio.f][intermedio.c] != 'P' and
           terreno[intermedio.f][intermedio.c] != 'M' and
           terreno[intermedio.f][intermedio.c] != 'B';
  check2 = terreno[final.f][final.c] != 'P' and
           terreno[final.f][final.c] != 'M' and
           terreno[final.f][final.c] != 'B';
```

```
  check3 = ViablePorAlturaR(altura[final.f][final.c] - altura[st.f][st.c],
                            st.zapatillas);
  return check1 and check2 and check3;
}
```

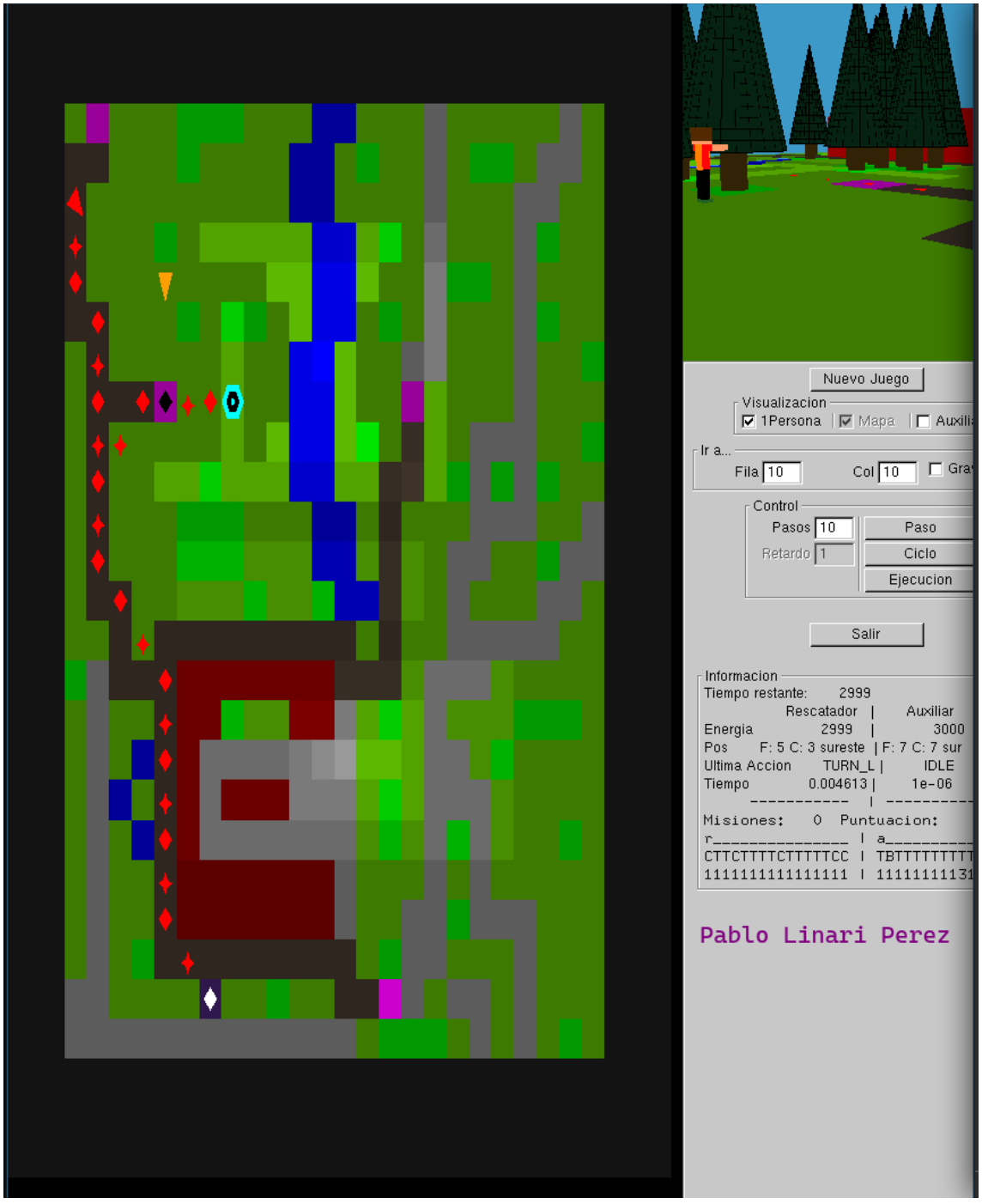
(e) Rellena los datos de la tabla con el resultado de aplicar

./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

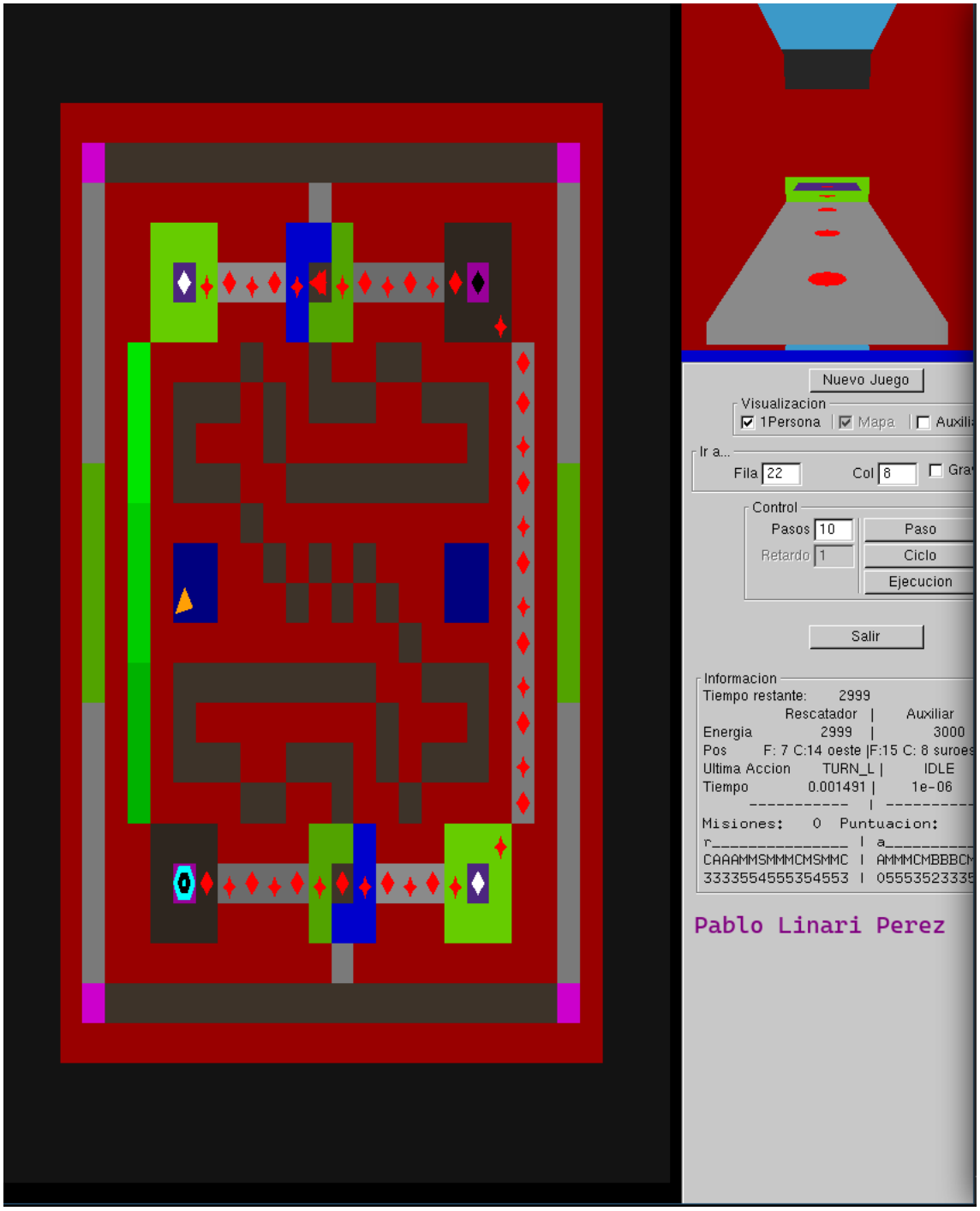
(f) Rellena los datos de la tabla con el resultado de aplicar

./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 el algoritmo A* la diferencia que se encuentra respecto a Dijkstra es el uso de una heurística , para ello he diseñado una función que calcula la distancia entre dos Estados usando la norma infinito . También he tenido que cambiar el struct que uso para comparar lpos nodos y ordenarlos en el priority queue ya que la comparación ahora se debe hacer sumandole la heurística al coste dee cada nodo . Por lo demás el código es el mismo salvando las diferencias de las acciones que cada agente puede hacer.

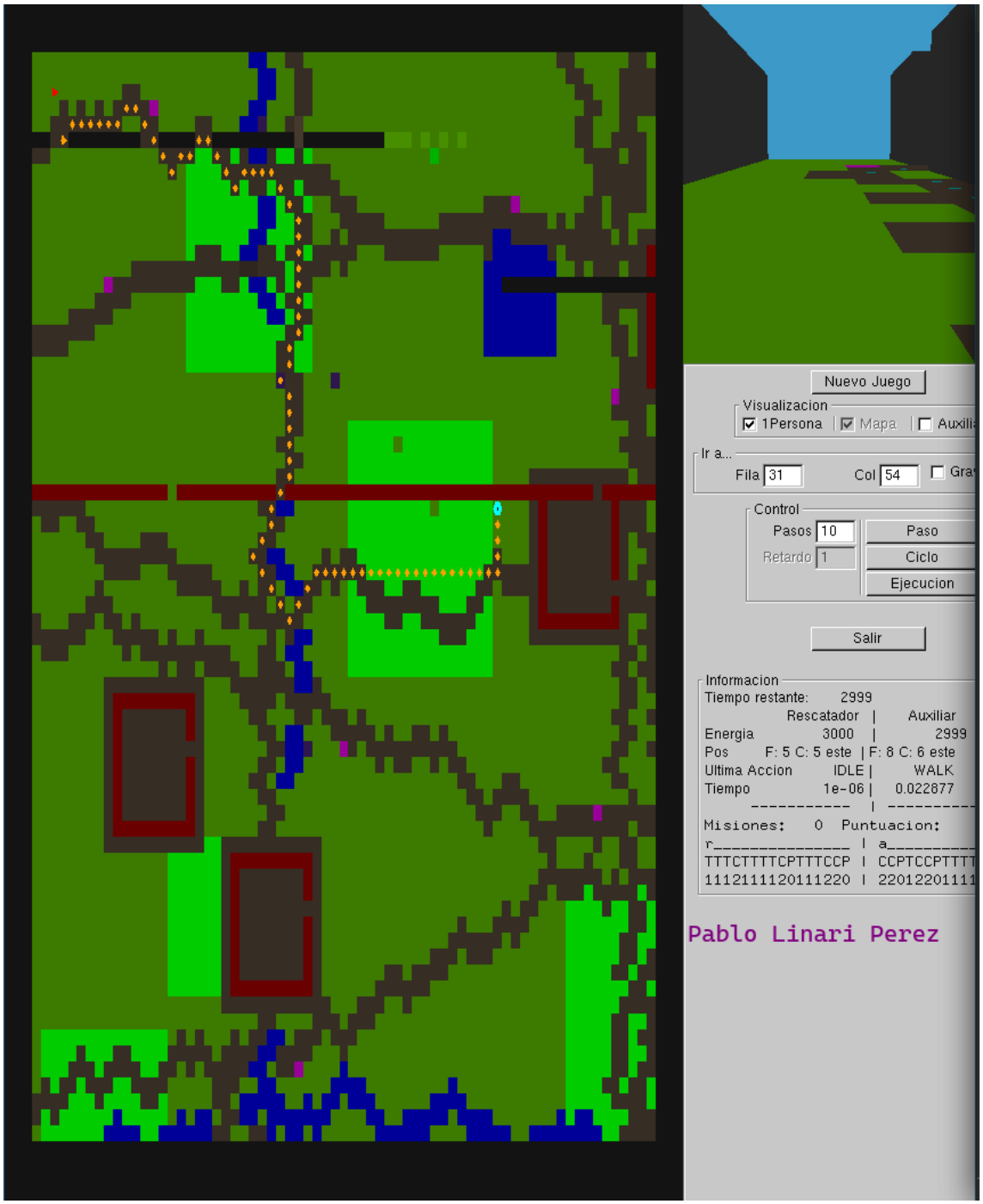
(b) Copia y pega en el siguiente recuadro de texto la heurística seleccionada. Además, describela y justifica la razón que hace que sea admisible para este problema.

```
int HeuristicaAestrella(const EstadoA &a, const EstadoA &b) {
  return std::max(abs(a.f - b.f), abs(a.c - b.c));
}
```

En la función anterior calculo el maximo valor entre la distancia por filas o la distancia por columnas de la posicion actual de un estado a la posicion de otro , el cual será el final en este problema . La heuristica es admisible si no sobreestima el coste para llegar al objetivo , por tanto será admisible si h(a,b) <=c(a,b) donde h es la heurística y c el coste real de ir desde a hasta b . Esta heurística es admisible ya que el coste de moverse de una casilla a otra colindante es como mínimo 1 y la distancia entre una casilla y otra colindante usando esta heurística simpre será 1 por su definición por tanto se cumple la condición necesaria para que sea admisible.

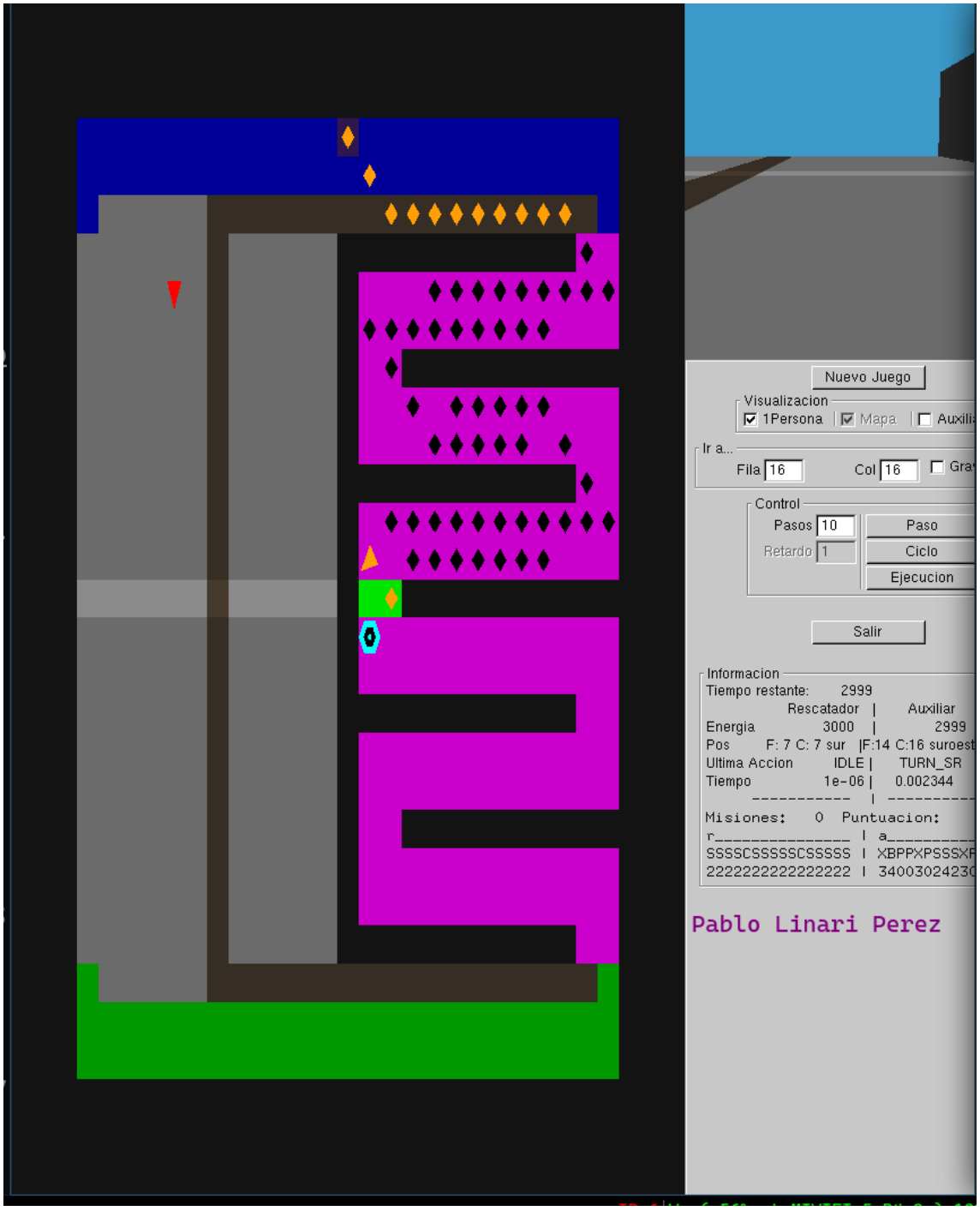
(c) Rellena los datos de la tabla con el resultado de aplicar

./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

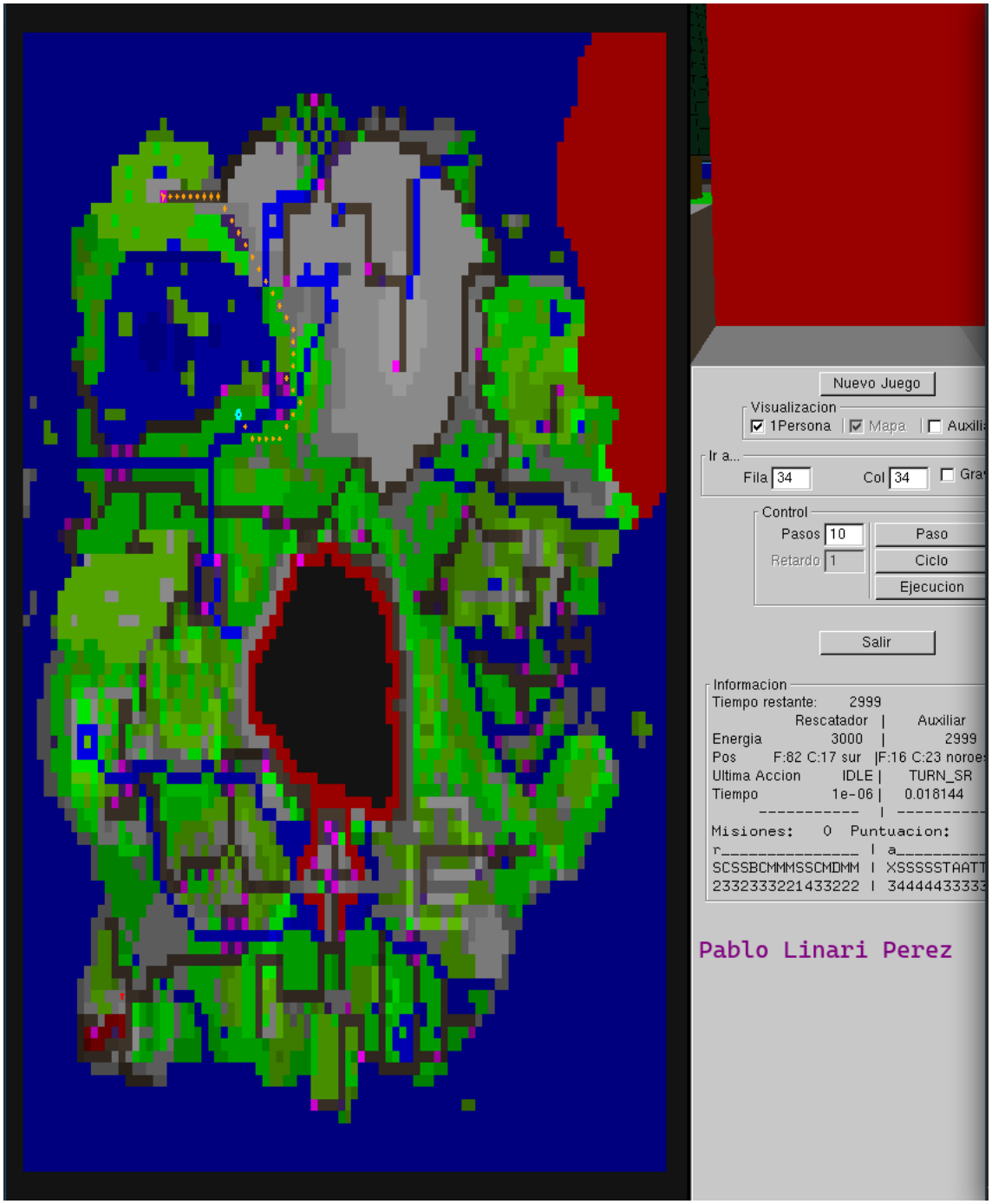
(d) Rellena los datos de la tabla con el resultado de aplicar

./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.

Para este nivel he usado un algoritmo A* en ambos agentes para buscar un camino hacia cada objetivo . En el rescatador he puesto la misma heurística que en el auxiliar para que sea más rápido a la hora de encontrar soluciones ya que nos interesa un camino y no hace falta necesariamente que sea el optimo .

El algoritmo se aplica igual pero cambiando la función coste y casillaAccesible .

Para la función casilla accesible he modificado el valor al que puede acceder incluyendo las casillas de tipo '?' , cuando se valora este tipo de casilla no se tiene en cuenta la altura ya que al no sabe su cota no podemos compararla con la nuestra real y por tanto muchas veces nos negaría el paso por casillas que si serían transitables . Si la casilla es conocida si se valora la cota .

En la función coste se le han asignado a las casillas del tipo '?' coste 2 ya que nos interesa que descubra mapa pero una vez que ya conozca suficiente que no use este valor siempre y tengan mayor prioridad los caminos y senderos .

Por último en la función principal del nivel cada vez que se choca se busca un plan nuevo y cada vez que se intenta acceder a una casilla se comprueba la altura para evitar caer al vacío si es una casilla desconocida (ha sido evaluada sin cota).
Si no se puede acceder a la casilla por el motivo anterior o no tenemos un plan trazado se aplica una lógica parecida a la del nivel 1 para poder salir de la situación actual y volver a calcular un plan .

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

Uso en ambos agentes el algoritmo A* que obtiene el camino óptimo de una forma más rápida que Dijkstra y en ambos he usado la misma heurística . En el rescatador he usado la misma para que sea más agresivo a la hora de buscar .

(c) ¿Bajo qué condiciones replanifica tu agente?

El agente replanifica si se choca contra algún objeto del mapa , o si la casilla que se creía accesible al crear el plan no lo es . Antes de ereplanificar el agente ejecuta movimientos parecidos al nivel 1 para no ejecutar el plan desde el mismo estado en el que no funcionó.

(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 asignado el valor 2 ya que tras varia pruebas era el que mejor resultados obtenía y de esta manera sigo priorizando caminos y senderos y como segunda prioridad tengo casillas desconocidas.

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

Si , la recarga de energía se efectua cuando el agente auxiliar o el rescatador tiene menos de 2000 de energía , si pasan por una casilla 'X' y se cumple esa condición el agente se espera en esa casilla hasta que su energía supere los 2000 , de nuevo tras varias pruebas este valor era el que mejor resultado daba.

(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.

(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	**2581**
Tiempo Consumido	**0.020204**
Nivel Final de Energía (Rescatador)	**0**
Nivel Final de Energía (Auxiliar)	**2057**

Objetivos encontrados	(11) 32

(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	1981
Tiempo Consumido	0.16592
Nivel Final de Energía (Rescatador)	0
Nivel Final de Energía (Auxiliar)	899
Objetivos encontrados	(10)30

(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	734
Tiempo Consumido	0.4379
Nivel Final de Energía (Rescatador)	0
Nivel Final de Energía (Auxiliar)	1521
Objetivos encontrados	(33)131

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.