

INFORME-
PRÁCTICA DE
BÚSQUEDA

Implementación del algoritmo A* en un entorno simulado. Pathfinding.

Realizado por
Jorge O. Blanchard Cruz
Sabato Ceruso
Kevin Martín Chinaea

Contenido

Path-finding..... 3

Entorno del problema..... 3

Aplicación web..... 4

 Equipo de desarrollo..... 4

 Herramientas de desarrollo..... 4

 Árbol de directorios..... 4

 Implementación del Algoritmo A* 6

 Puesta en marcha rápida..... 7

 Descripción de la Interfaz de usuario..... 8

 Elementos zona roja..... 8

 Elementos zona verde..... 8

 Descripción de la escena..... 9

 Utilización del ratón..... 9

 Elementos del Mapa..... 9

Aplicación Generador de Mapa..... 11

 Puesta en marcha..... 11

Descripción de la Interfaz de usuario..... 14

 Elementos zona roja..... 14

 Elementos zona verde..... 14

Control de errores en el mapa..... 15

Sistema basado en agentes..... 17

 Definiciones básicas..... 17

 Tipo de agente..... 17

 Funcionalidades..... 17

Arquitectura software..... 17

Path-finding.

“La búsqueda de ruta en el contexto de los videojuegos se refiere a la forma en que una entidad en movimiento encuentra un camino alrededor de un obstáculo; el contexto más frecuente se encuentra en los videojuegos de estrategia en tiempo real (en el que el jugador orienta las unidades en torno a una área de juego que contiene los obstáculos), pero esta modalidad se encuentra en los videojuegos más modernos. La importancia de la búsqueda de rutas ha crecido tanto en importancia como los entornos de los juegos se han vuelto cada vez más complejos, y como resultado, varios paquetes de software de IA ya han desarrollado los algoritmos para resolver el problema.

Los videojuegos de estrategia en tiempo real generalmente contienen grandes áreas de terreno abierto que suele ser relativamente fácil de atravesar, en el que es muy común que más de una unidad esté viajando a la vez; esto crea una necesidad diferente, y a menudo son necesarios algoritmos más complejos para evitar un atasco de tráfico en los puntos más circulados del mapa o terreno, o cuando las unidades están en contacto unas con otras. En los juegos de estrategia los mapas normalmente están divididos en tiles, que actúan como nodos en el algoritmo de pathfinding.

Géneros estructurados más abiertos, tales como los videojuegos de disparo en primera persona que a menudo tienen más áreas cerradas (o una mezcla entre abierta y cerrada) que no están tan simplemente divididos en nodos, ha dado lugar al uso de mallas de navegación. Estos son construidos por la colocación de nodos en el mundo del juego, que almacenan detalles de qué nodos son accesibles desde el mismo.”

Wikipedia

Entorno del problema.

El entorno del problema es una cueva de dimensiones variables con distintos obstáculos que impiden el paso a nuestro agente. Cada uno de los escenarios de la cueva se caracteriza por tener una entrada, que será la posición inicial del agente, y una salida que será el objetivo del agente.

Nuestro robot deberá encontrar el camino mínimo a la salida, moviéndose en las cuatro direcciones: norte, sur, este y oeste; y, evitando los obstáculos (agua o piedras).

Aplicación web

Equipo de desarrollo

- CPU: Intel Core i7 3.60GHz (caché 10MB)
- Memoria RAM: 16,0 GB distribuidos en 4 canales.
- GPU: AMD ATI Radeon R9 290 4GB GDDR5 DirectX 11.2
- O.S: Windows 8.1 Pro N x64

Con éste equipo se configura la aplicación con todas las opciones gráficas activadas siendo el rendimiento óptimo con mapas de dimensión 99x99.

Herramientas de desarrollo

Para el desarrollo de este proyecto, se ha optado por el desarrollo web. Esto permite una gran portabilidad del proyecto, aunque también los navegadores difieren en características y esto presenta algunos problemas. También se ha escogido debido a la facilidad de implementar entornos 3D sencillos.

El navegador para el cual se ha desarrollado este proyecto es Mozilla Firefox, sabiendo que presenta errores con Google Chrome. Este proyecto ha sido testado bajo las últimas versiones de Mozilla Firefox presentes hasta la fecha, esto es la versión: 33.1.

Por tanto, la página web se ha escrito en HTML, utilizando también CSS. El código se ha desarrollado en Javascript.

Para generar el entorno 3D, así como para el *look and feel* de la web, se han utilizado los siguientes recursos:

- **jQuery:** es una librería utilizada ampliamente en la web que facilita el acceso a los elementos del árbol DOM de HTML.
- **Metro UI:** es una librería que proporciona una estética similar a la utilizada en la interfaz metro de Windows 8 y Windows 8.1. Dotando así a la página web de cierto estilo característico
- **Three.js:** es una librería que funciona bajo OpenGL empleada en la interfaz de gráficos 3d.

Árbol de directorios

- **ApplIssues:** Se encuentra la aplicación GeneradorMapa desarrollada en CSharp.
- **css:** Ficheros de estilos de la aplicación web.
- **data:** base de datos necesarios para el correcto funcionamiento de la aplicación.
- **fonts:** estilos de letras.
- **js:** Contiene los ficheros de código fuente de la aplicación en lenguaje *javascript* así como las librerías necesarias.
 - **main.js**
Fichero de código principal de funcionamiento del UI y creación inicial de las instancias de clase.
 - **C3DWorld.js**
Clase que contiene las estructuras y métodos para el funcionamiento de la escena en 3d.
 - **CMothership.js**

Clase que provee de información al robot.

· **CAgent.js**

Clase que implementa al robot y su algoritmo de búsqueda A*.

Implementación del Algoritmo A*

Para la implementación de A* se utilizó los apuntes datos en clase y la estructura y funcionamiento básico es exactamente el mismo.

Para implementar este algoritmo, suponemos que el agente dispone de un mapa topográfico de la zona. La razón de esto es que si no tuviera un mapa, el agente debería desplazarse hasta la última posición conocida de una trayectoria para poder ramificarla; aumentando considerablemente el número de pasos para poder realizar el camino mínimo. Sin embargo, al tener mapa, no le hace falta desplazarse en ningún momento hasta encontrar el camino mínimo, pudiendo hacer el menor número de pasos hasta la salida a costa de utilizar memoria para guardar el mapa.

Otro aspecto a tener en cuenta son los obstáculos, en este caso, cada vez que se va a ramificar una trayectoria, no se tienen en cuenta las posiciones adyacentes que poseen un obstáculo, pues, se consideran inalcanzables.

Breve descripción:


1. Tenemos dos listas: ABIERTA y CERRADA.
2. En la lista CERRADA se van guardando las trayectorias que son elegidas de la lista ABIERTA con coste mínimo.
3. En una lista ABIERTA, derivamos primeramente el nodo salida como asignando el coste de cada trayectoria y ordenamos la lista de forma ascendente.
4. De esa primera rama o trayectoria que ha quedado en la lista hacemos de nuevo la derivación obteniendo los nuevos costes y trayectorias, volvemos a ordenar la lista.
5. De esta forma vamos a explorar todas las ramas que tengan menor valor acumulado más su función de heurística (distancia Manhattan).
6. Las trayectorias de la lista ABIERTA que tengan nodo final común y con costes superiores o iguales son eliminadas, quedando solo una.
7. Si alguna de las trayectorias de la lista ABIERTA tienen nodo final común con las de la lista CERRADA y es de mayor o igual coste es eliminada de la lista.
8. Al encontrar una rama que llegue al objetivo, eliminamos todas aquellas que su coste sea mayor o igual a la del objetivo y seguimos avanzando por las demás hasta que una de las ramas con nodo objetivo quede en la primera posición de la lista ABIERTA ordenada. En ese momento tendremos el camino mínimo.
9. Si la lista ABIERTA se vacía antes de encontrar el nodo objetivo, el problema no tiene solución posible.

Puesta en marcha rápida

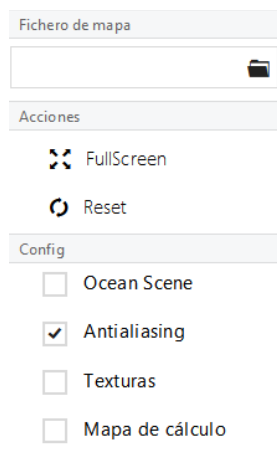
1. Abra con Mozilla Firefox (mínimo versión 33.1) el fichero 'index.html'

Si el navegador muestra una barra superior en tonos negros y un fondo central en color azul celeste es que se ha ejecutado correctamente.

La configuración 3d viene configurada por defecto en Firefox.


2. Para cargar un mapa existente: Haga click en el botón del 'engranaje'  que se encuentra en la barra superior de la aplicación ->


Mostrará un desplegable:



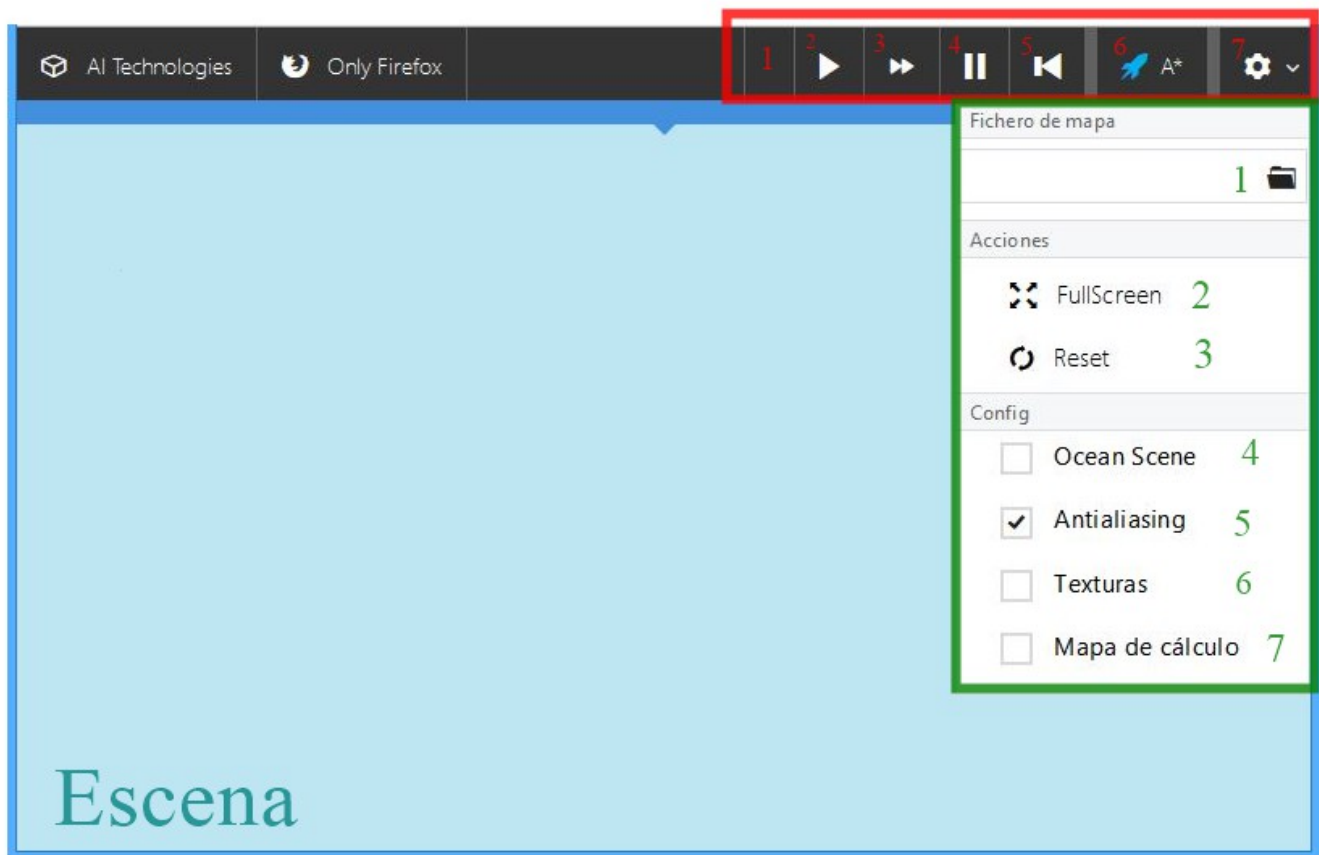
➔ y a continuación podrá hacer click en el icono de la carpeta  para seleccionar un fichero de su máquina.

En el directorio 'mapfiles/' encontrará ejemplos de mapas ya generados.

3. Para ejecutar la estrategia de búsqueda A* pulse el botón con el icono del 'Cohete'  de color azul.

4. Pulse el botón 'Play'  para ver el camino mínimo representado por el robot.

Descripción de la Interfaz de usuario



A continuación se describirán los distintos elementos del interfaz identificados en colores y números:

Elementos zona roja

1. **INFO:** Se muestra información de la posición del robot cuando se ejecuta la representación del camino mínimo.
2. **PLAY:** Ejecutará la representación del camino mínimo.
3. **SPEED:** Se utiliza para cambiar la velocidad del robot. Tres velocidades: Lenta, media, rápida.
4. **PAUSE:** Hace una pausa en el movimiento del robot.
5. **REV:** Rebobina la marcha del robot a su estado inicial de trayectoria mínima y coordenadas de salida.
6. **A*:** Ejecución del algoritmo de búsqueda A*.
7. **CONFIG:** Al pulsar este botón mostrará el **panel de Configuración** marcado en verde.

Elementos zona verde

1. **Fichero de mapa:** Sirve para seleccionar un fichero de mapa.
2. **Fullscreen:** Ejecuta la aplicación en pantalla completa.
3. **Reset:** Restablece los valores de configuración.
4. **OceanScene:** Muestra en la escena un fondo de océano y cielo. Es necesario pulsar **Reset** para modificar este parámetro. (Ésta configuración consume recursos)
5. **Antialiasing:** Muestra la escena con bordes redondeados. Es necesario pulsar **Reset** para modificar este parámetro. (Ésta configuración consume recursos)
6. **Texturas:** Añade texturas a los distintos elementos del mapa que se cargue posteriormente en la escena. Es necesario pulsar **Reset** para modificar este parámetro. (Ésta configuración consume recursos)

7. **Mapa de cálculo:** Al ejecutar A* mostrará los caminos que ha tenido que realizar el robot para obtener la trayectoria mínima. Se representa en cubos de color violeta, y cuanto el color es más intenso más cálculos ha tenido que realizar.

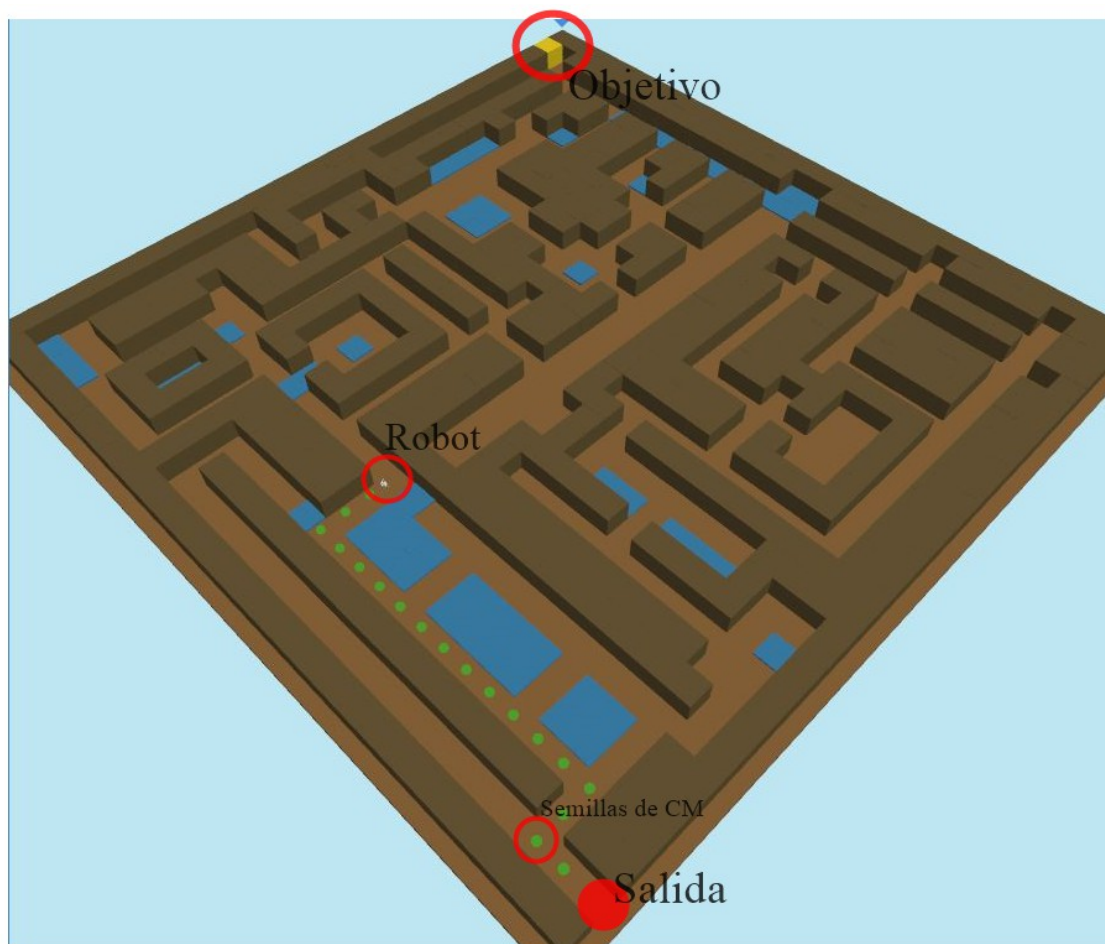
Descripción de la **escena**



UTILIZACIÓN DEL RATÓN

1. **Botón izquierdo:** Manténgalo pulsado para rotar la escena.
2. **Botón central:** Rueda arriba acerca el zoom, rueda abajo aleja el zoom. También puede mantenerlo pulsado y mover el ratón hacia arriba y abajo para acercar y alejar el zoom.
3. **Botón derecho:** Manténgalo pulsado para desplazar la escena.

ELEMENTOS DEL MAPA



Mapa de cálculo representado.



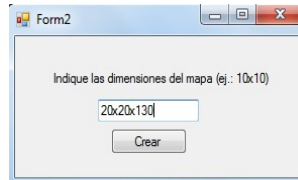
Aplicación Generador de Mapa

Puesta en marcha

1. Creamos el mapa.

Crear

He introducimos según la siguiente nomenclatura las características que tendrá nuestro mapa: “*númerocolumnasxnúmerofilasxnúmeroobstaculosaleatorios*”.



2. Se crea nuestro mapa según los datos introducidos, ahora podemos modificar el terreno a placer según los botones especificados en la parte inferior izq. de nuestro programa.



Inicialmente el número de objetos especificado al crear el mapa a la hora de representarlo es agua.

Ejemplo:

Mapa aleatorio creado.



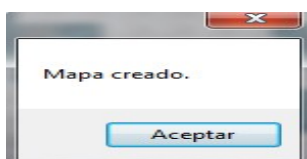
Mapa modificado.



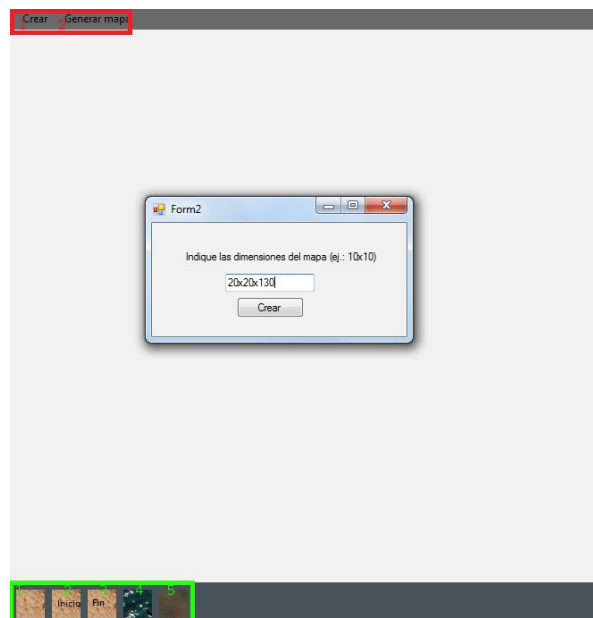
3. Generación del mapa, apretando este botón se crea un fichero.

Generar mapa

Este fichero contiene los datos de nuestro mapa, el camino mínimo y además los lugares que ocupan cada objeto especificado en nuestro mapa incluyendo el tipo de objeto del que se trate (agua, una roca, tierra, el inicio o el final) . Este es el fichero con el que se deberá abrir la aplicación web detallada en el apartado anterior.



Descripción de la Interfaz de usuario



Elementos zona roja

1. **Crear:** Abre una nueva ventana en donde introducir los valores del número de filas, número de columnas y número de obstáculos que abran en el mapa.
2. **Generar mapa:** Creará un fichero donde se describirá nuestro mapa con todos los detalles de este para posteriormente abrirlo con la aplicación web. Este fichero se creado en la carpeta mapfiles y lleva por defecto el nombre *generado*.

Elementos zona verde

Estos elementos son específicos para cuando ya tengamos un mapa creado, son los distintos tipos de terrenos que podemos añadir a nuestro mapa.

1. **Tierra:** Este es el único terreno por donde nuestro robot se podrá desplazar equivale a no tener ningún obstáculo en el camino.
2. **Inicio:** Posición inicial de donde partirá nuestro robot.

3. **Fin:** Casilla final a la que nuestro robot tiene que llegar realizando previamente el camino mínimo.
4. **Agua:** Este es uno de los obstáculos que nuestro robot debe evitar, se trata de la representación de un charco de agua.
5. **Roca:** Segundo obstáculo que nuestro robot tendrá que evitar, se trata de la representación de una piedra.

Control de errores en el mapa

Con el fin de gestionar una correcta generación del mapa están implementados los siguientes controladores, en el caso de que haya algún tipo de error en el mapa no permite generarlo, dando un error mostrado mediante un mensaje en pantalla.

Introducir en el mapa dos inicios.



Intento de crear otro mapa cuando ya hay uno existente en proceso de desarrollo.



Sistema basado en agentes.

Definiciones básicas.

Percepciones: Sensores sobre el entorno (obstáculos, posición libre, mineral, etc), memoria común con resto de agentes y sistema de comunicación directa.

Objetivos: Retirar minerales de forma eficiente y salir.

Entorno: Una cueva con minerales, tiene varias entradas y los minerales se distribuyen en cúmulos o zonas.

Acciones: Moverse, recoger mineral, descargar mineral y comunicar.

Tipo de agente.

El tipo de agente que se utilizará será el agente reactivo basado en modelos. La ventaja de utilizar este tipo de agente es que además de tener información sobre el estado actual también se tiene información sobre como evoluciona el entorno del agente lo que es importante para la situación actual. Recordemos que nuestro problema es el de una cueva que contiene minerales y, los agentes deberán ser capaces de encontrar el mejor camino hasta los minerales, recogerlos y traerlos nuevamente por donde entraron.

Por lo tanto, es útil conocer, no sólo el estado actual del entorno, sino también la evolución del mismo para tenerlo en cuenta a la hora de realizar las acciones; por ejemplo, saber cuantos minerales se han recogido para saber si ya es suficiente, o saber de dónde se han recogido para realizar búsquedas más intensas en esas zonas esperando que hayan más minerales.

Funcionalidades.

Cada uno de los agentes deberá tener la capacidad de moverse por el entorno, tener sensores que le den información del mismo (si se puede desplazar a una posición adyacente, si hay un mineral o un obstáculo, etc.) cargar minerales, detectar si se ha cargado algún mineral y descargarlos.

También hace falta que exista un sistema de comunicación entre los agentes y una memoria común para poder pasarse información útil, tal como los caminos que ya se han explorado, obteniendo así información sobre las posiciones donde no hay minerales, donde los había, donde todavía hay y donde hay obstáculos. De esta manera se pueden agilizar las búsquedas, de tal modo que los agentes no repitan rutas ya exploradas, que vayan directamente a donde ya saben que hay minerales, o que si no saben donde hay pero si donde había, que exploren en las cercanías de las posiciones donde hubo minerales en busca de otros adyacentes.

Arquitectura software.

La arquitectura software que se utilizará será una arquitectura reactiva jerárquica multiagente cooperativa, donde la decisión de qué acción tomar se efectúa a través de módulos de conducta ordenados de forma jerárquica por capas permitiendo priorizar unas acciones frente a otras.

Las conductas a adoptar por el agente serían las siguientes:

- 1-Si encuentra obstáculo, sortearlo.
- 2-Si no, y el agente transporta un mineral, llevar y descargar mineral a la salida.
- 3- Si no carga un mineral y detecta un mineral en una posición adyacente, cargar mineral.
- 4-Si no carga un mineral, mirar en la memoria común de agentes si hay algún mineral en algún sitio y no se espera cargar por ningún agente.
- 5- Si no carga un mineral, mirar en la memoria común de agentes donde se han recogido mineral en busca de un cúmulo de minerales en posiciones cercanas a esa.
- 6- Si no, y no se ha recogido ningún mineral en toda la cueva o bien todas las zonas de donde se han recogido se han acabado la cantidad de minerales esperados por cúmulo y no se han recogido la cantidad total de minerales esperados; moverse por caminos, en lo posible no explorados por otros agentes, en busca de más minerales.

7-Si no, si se han recogido la cantidad de minerales esperados o bien se ha explorado toda la cueva y no se han encontrado más minerales, salir de la cueva.