JGOMAS

JGOMAS

JADE Game Oriented MultiAgent System

Sesión 2

Sistemas Inteligentes
FI, 2006

Toni Barella
tbarella@dsic.upv.es

1

JGOMAS

Índice

- Algoritmos A*
- Algoritmo GraphSearch
- Mapas en JGOMAS
- Trabajo a realizar

1

JGOMAS

Índice

- ► **Algoritmos A***
- Algoritmo GraphSearch
- Mapas en JGOMAS
- Trabajo a realizar

JGOMAS

Algoritmos A* (I)

- Un algoritmo de búsqueda es de tipo A, cuando emplea una función heurística de la forma:

$$f(n) = g(n) + h(n)$$

  - g(n) : coste del camino de coste mínimo entre el nodo inicial S y el nodo *n* (este valor se puede determinar).
  - h(n) : estimación del coste del camino de coste mínimo entre el nodo *n* y un nodo meta (información heurística).

- Un algoritmo es del tipo A* cuando:

$$\forall\ n:\ \{\ h(n) \leq h^*(n)\}$$

  - h*(n): coste del camino de coste mínimo entre el nodo *n* y un nodo meta.

JGOMAS

Algoritmos A* (II)

- El conocimiento heurístico sobre el problema queda reflejado en la función h(n):
  - h(n) = 0 ausencia de conocimiento
  - h(n) = h*(n) conocimiento máximo necesario
  - h(n) > h*(n) más conocimiento del necesario
- Un algoritmo A1 está más informado que un algoritmo A2, si h1(n) > h2(n): en este caso A2 expande al menos tantos nodos como A1.

JGOMAS

Algoritmos A* (III)

- Admisibilidad
  - **h(n) = 0**: coste cálculo de h(n) nulo. (Búsqueda lenta. Admisible.)
  - **h(n) = h*(n):** coste cálculo de h(n) grande (Búsqueda rápida. Admisible).
  - **h(n) > h*(n):** coste cálculo de h(n) muy grande. (Búsqueda muy rápida. No admisible)
- h* normalmente no se conoce, pero, en general, puede determinarse si un h es minorante de h*

JGOMAS

Índice

- Algoritmos A*
- ► **Algoritmo GraphSearch**
- Mapas en JGOMAS
- Trabajo a realizar

1

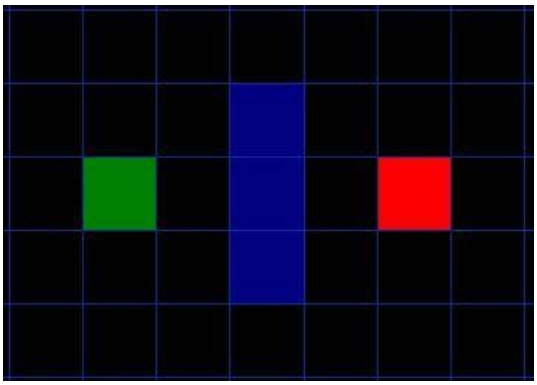
JGOMAS

Algoritmo GraphSearch

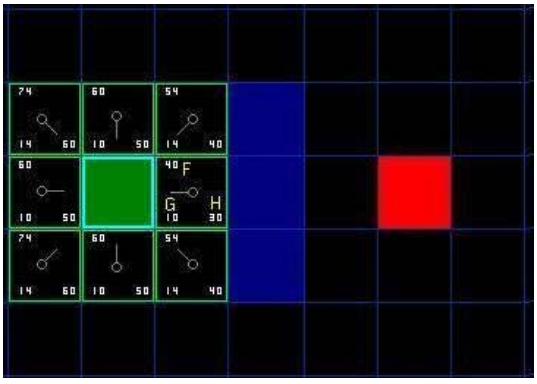
```
1 Create a node containing the goal state: node_goal
2 Create a node containing the start state: node_start
3 Put node_start on the OPEN list
4 while the OPEN list is not empty
5 {
6   Get the node off the OPEN list with the lowest f and call it node_current
7   If node_current is the same state as node_goal: break from the while loop
8   Generate each state node_successor that can come after node_current
9   For each node_successor of node_current
10    {
11      Set the cost of node_successor to be the cost of node_current plus the cost to
          get to node_successor from node_current
12      Find node_successor on the OPEN list
13      If node_successor is on the OPEN list but the existing one is as good or
          better then discard this successor and continue with next successor
14      If node_successor is on the CLOSED list but the existing one is as good or
          better then discard this successor and continue with next successor
15      Remove occurences of node_successor from OPEN and CLOSED
16      Set the parent of node_successor to node_current
17      Set h to be the estimated distance to node_goal (using the heuristic function)
18      Add node_successor to the OPEN list
19    }
20    Add node_current to the CLOSED list
21 }
```

1

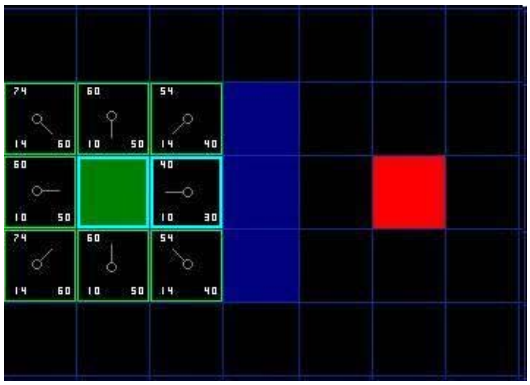
Ejemplo (I)



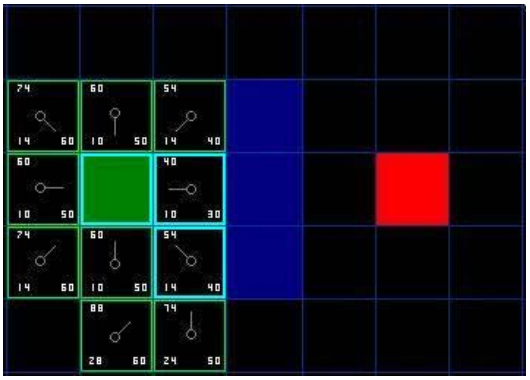
Ejemplo (II)



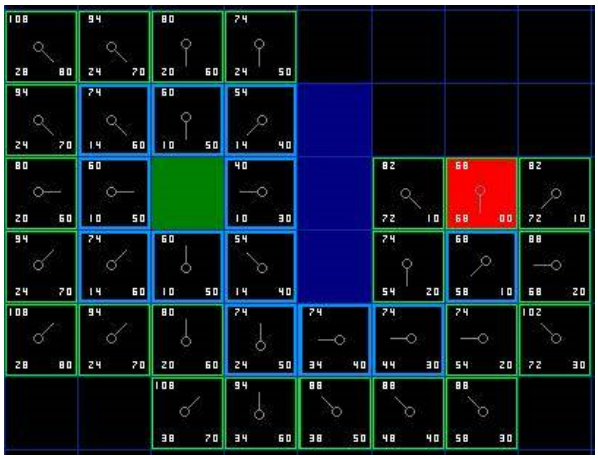
Ejemplo (III)



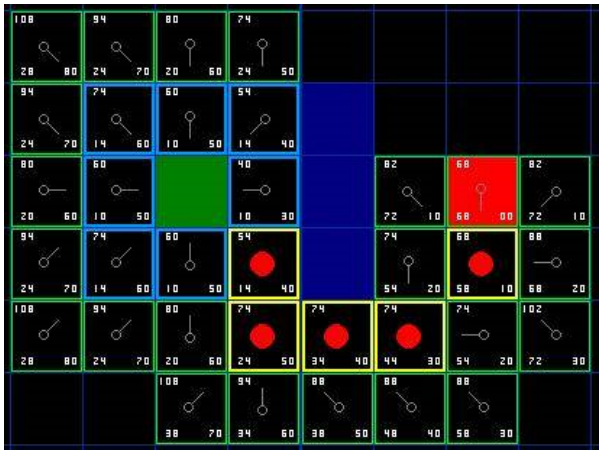
Ejemplo (IV)



Ejemplo (V)



Ejemplo (VI)



JGOMAS

Índice

- Algoritmos A*
- Algoritmo GraphSearch
- ► **Mapas en JGOMAS**
- Trabajo a realizar

1

JGOMAS

Mapas (I)

- Una partida se desarrolla en un entorno virtual (mundo virtual) de 256 × 256
- Posición de los agentes:

  $\forall x \in [0..255], \ \forall z \in [0..255]$

- Cada agente tiene acceso al mapa donde se desarrolla la partida:
  - Atributo: **`m_Map`**

1

JGOMAS

Mapas (II)

- **`m_Map`** contiene elementos estáticos del mapa, en coordenadas del mundo virtual:
  - Situación inicial de la bandera objetivo:
    - **`double`** `GetTargetX()`
      - `valor devuelto ∈ [0..255]`
    - **`double`** `GetTargetZ()`
      - `valor devuelto ∈ [0..255]`
  - Posición de la base aliada y del eje

JGOMAS

Mapas (III)

- **`m_Map`** también es una mapa de costes:
  - Malla de dimensiones 32 × 32
  - Métodos disponibles:
    - **`int`** `GetSizeX()`
      - `valor devuelto ∈ [0..31]`
    - **`int`** `GetSizeZ()`
      - `valor devuelto ∈ [0..31]`
    - **`boolean`** `CanWalk(int x, int z)`
      - `∀ x ∈ [0..31], ∀ z ∈ [0..31]`

JGOMAS

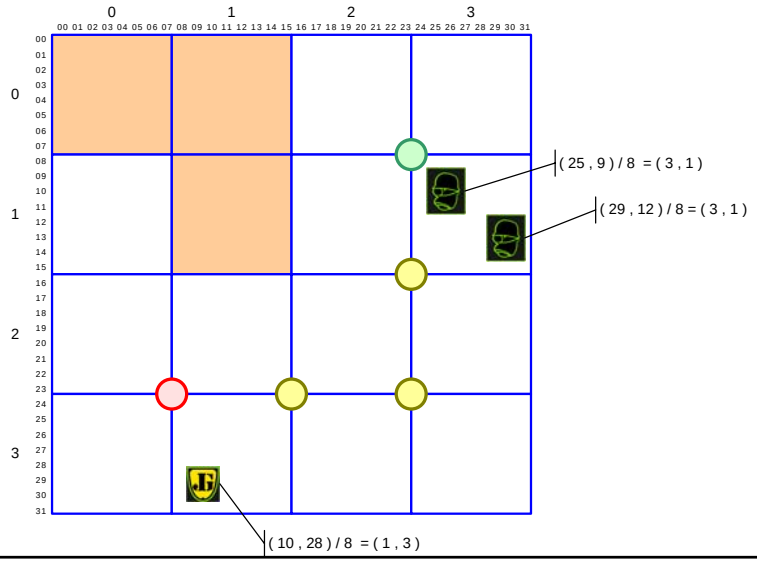
Mapas (IV)

- Para utilizar el mapa de costes, es necesario realizar una transformación de escala:

```
int x = m_Movement.getPosition().x;
x /= 8;
int z = m_Movement.getPosition().z;
z /= 8;
if ( CanWalk(x, z) ) {
  . . .
}
```

JGOMAS

Mapas (V)



JGOMAS

Índice

- Algoritmos A*
- Algoritmo GraphSearch
- Mapas en JGOMAS
- ► **Trabajo a realizar**

1

JGOMAS

Trabajo a Realizar (I)

- Objetivo:
  - Implementar un agente aliado que disponga de un método de encaminamiento que le permita capturar la bandera y volver a la base en un mapa del tipo del mapa 12 suministrado, en un máximo de 15 minutos.

1

Trabajo a Realizar (II)

- Entrega:
  - Ficheros *.java desarrollados
  - Fichero <login_alumno>.jar
  - Breve descripción del desarrollo (1 página)

1

JGOMAS

JGOMAS

JADE Game Oriented MultiAgent System

Sesión 2

Sistemas Inteligentes
FI, 2006

Toni Barella
tbarella@dsic.upv.es

1