Memoria Algoritmo A Estrella

Nerea Jiménez González
Yhondri Acosta Novas

Marzo 2020

UNIVERSIDAD
COMPLUTENSE
MADRID

Índice

1. Descripción del sistema

2. Detalles de la implementación

2.1. Lenguaje utilizado

Para la implementación, se utiliza el lenguaje JAVA.

2.2. Procedimiento seguido para la implementación

El procedimiento que se ha seguido para la implementación ha sido el siguiente:
En primer lugar, se resolvió el ejemplo dado en el enunciado, y se comenzó a programar este ejemplo en concreto. Durante esta explicación, nos referimos a nodos y casillas. El nodo representa la casilla, por lo que puede que se utilicen ambas denominaciones en algún momento.
Empezamos por definir las entidades a tener en consideración, las cuales son:

- **Coordinate**
  Coordenada compuesta por fila y columna, para evitar confusiones a la hora de programar.
- **Coordinate type**
  Enumerado utilizado para saber que tipo de movimiento es para dibujar en el tablero este.
- **Node**
  Contiene toda la información del nodo: g, h, f y coordenada, que es la casilla del tablero en cuestión.

Comenzamos por las comprobaciones a donde se puede expandir el nodo de forma que, no se puede expandir donde:

- Casillas inaccesibles
  Hablaremos sobre ellas en el apartado de ampliaciones.
- Casillas fuera del tablero

Una vez tenemos las casillas a donde podemos avanzar, estas las decidimos guardar en una cola de nodos de prioridad ordenada de menor a mayor por f. Esta cola, *openNodesPriorityQueue*, se correspondería a la lista abierta.
Para guardar los nodos que vamos cerrando, lo que se correspondería a la lista cerrada, utilizamos una lista de coordenadas, *closedCoordinateList*, ya que una vez que cerramos un nodo, sólo nos interesa su coordenada, que sería como el nombre del nodo.
El algoritmo en sí es un bucle *while*, del cual se sale cuando no nos quedan nodos que recorrer en *openNodesPriorityQueue*, o bien cuando el nodo meta, *goalNode*, es null.
Si se sale del bucle porque no tenemos para nodos que recorrer, y no se ha

llegado a la casilla meta, *goalNode=null*, no existirá un camino.
Cuando las coordenadas del *goalNode* coinciden con las coordenadas del nodo Meta, *goalCoordinate*, se ha encontrado el mejor camino posible y el algoritmo termina.

2.3. Ampliaciones realizadas

Las ampliaciones realizadas son las siguientes:

2.3.1. Casillas inaccesibles

Son las casillas que se colorean de color rojo. Estas casillas actúan como un muro, y no se puede pasar por ellas.

2.3.2. Casillas con penalización

Estás casillas conllevan una penalización al pasar por ellas, por lo que puede que el primer camino que pensamos que es el óptimo para llegar a la casilla meta, deje de serlo por esto. Las casillas son de color amarillo, y la penalización sobre la f del nodo es de 10.

2.3.3. Casillas de paso obligatorio

Estas casillas son casillas por las que hay que pasar obligatoriamente en nuestro camino, *waypoints*.

2.4. Otros elementos de interés

3. Código ejecutable

Se adjunta en el archivo tipo zip con nombre **Código** zip, y un ejecutable tipo jar con nombre **Ejecutable**.

4. Manual de usuario

Para empezar con la aplicación, clickeamos dos veces sobre el archivo ejecutable.
A la hora de marcar las casillas, seleccionaremos en el menú izquierdo, bajo los botones de empezar y reiniciar, el tipo de casilla a dibujar.
Una vez seleccionado el tipo de casilla que queremos, clickeamos las casillas que queremos que sean de dicho tipo.
Una vez satisfechos con nuestro tablero, clickeamos en el botón *Empezar*.
Si queremos limpiar el mapa, clickeamos sobre el botón *Reiniciar*.