Práctica 2: Documentación

Autor: Fco Javier Bolívar Expósito **Grupo: 2ºD1**

Nivel 1

La implementación de la búsqueda por anchura es muy similar a la búsqueda por profundidad proporcionada, usando para la lista de abiertos una cola y solo introduciendo los hijos generados en abiertos si su estado no se encuentra ya en abiertos.

Para implementar la búsqueda por coste uniforme es necesario asignar un coste a cada camino. Cada nodo tiene asociado un coste, calculado como el coste de su padre más el coste de la acción a realizar, en el caso de girar el coste es 1, en el caso de avanzar he creado una función unsigned int costeCasilla(Estado estado) que devuelve el coste de avanzar según el tipo de casilla. Ahora la lista de abiertos será una cola con prioridad ordenada por coste y solo se introducirá un hijo generado si su estado no está en abiertos o el estado de abiertos tiene un coste mayor.

Nivel 2

Agente Reactivo
Mientras el agente no conoce su posición utiliza un comportamiento reactivo para encontrar una casilla amarilla.

Para este comportamiento el agente usa una matriz de tiempos inicializada a ceros que tiene el doble de tamaño que la matriz del mapa y un contador inicializado a 0. Empezando la posición del agente en el centro de la matriz de tiempos cada vez que piensa y decide la acción a realizar asigna a la casilla en la que se encuentra el valor del contador e incrementa el contador en 1. Si la casilla de delante no es transitable le asigna un valor muy alto y gira a la derecha. Si la casilla es transitable usa la función int elegirCasilla() que devuelve que casilla de las adyacentes al agente tiene menor valor y por lo tanto no ha visitado en más tiempo, si esta casilla es la de enfrente el agente avanza o espera si hay un aldeano, si no, gira.

Con este comportamiento el agente reactivo ya o tiene problemas para explorar el mapa y evita recorrer solo una parte de este, pero para que tenga un funcionamiento más eficiente realiza funciones adicionales.

Cada vez que piensa llama a las funciones:
void actualizarMapaReactivo(Sensores sensores)
Si el agente detecta en sus sensores una casilla amarilla le asigna en la matriz de tiempos un -2 a su posición y un -1 a sus 2 posiciones a la derecha, a la izquierda, por encima y por debajo. De esta manera si el agente ve una casilla amarilla irá hacia ella.

void actualizarMapaPreliminar(Sensores sensores);
Aunque el agente aun no sepa su posición este guarda la información que recibe del mapa en un mapaPreliminar el doble de grande que el real suponiendo que su posición inicial es el centro.

En el momento en que el agente se situa calcula la diferencia entre la posición que ha usado hasta ahora el agente suponiendo que empezaba en el centro y su posición real y llama a la función void copiarMapa(int difFil, int difCol) que pasa el mapaPreliminar al mapa real (mapaResultado).

Agente Deliberativo

Cuando el agente está situado este actúa de forma deliberativa para calcular un plan óptimo y llegar a su objetivo. Cada vez que piensa actualiza el mapa añadiendo la información que detecta en sus sensores y si no hay un plan lo calcula usando la búsqueda de coste uniforme.

Después el agente sigue el plan calculado, pero debido a la falta de información o a los aldeanos que se mueven por el mapa pueden surgir imprevistos. Si la acción va a llevar al agente a una casilla no transitable el agente vuelve a calcular un plan ahora que conoce estas casillas y si va a llevar a chocarse con un aldeano el agente espera y cuando se aparte continua el plan.