Documentación

Descripción de la solución

El bot se ha desarrollado para que sea capaz de realizar el **Nivel 3**. Dispone de una matriz de 200x200 para poder trazar planes cuando desconoce el mapa y una serie de variables de estado que le ayudan a conocer su situación actual.

El bot con cada movimiento va escribiendo en esa matriz (y en **mapaResultado** por si se utiliza interfaz gráfica) para recordar las posiciones.

Su comportamiento se divide en dos:

- Comportamiento reactivo: Al iniciar la ejecución, pues el bot desconoce su ubicación, explorará el mapa de una manera determinada (se detalla a continuación) hasta que encuentra un punto PK en su visión. En ese momento localiza la posición en la matriz (no en **mapaResultado**) recorriendo la zona circundante a él, y llama a **pathFinding** para trazar un plan hasta ella (en el caso de que viese otro punto PK más cercano por el camino no varía su ruta).

  La forma de explorar consiste en avanzar 6 casillas y girar hacia un lado (alternando giros a derecha y a izquierda para evitar trazar círculos), con la condición de que se pueda realizar ese movimiento. Se ha elegido esta forma porque es la que aprovecha al máximo la visión del personaje y permite atisbar la mayor parte del mapa posible hasta encontrar el punto PK.

- Comportamiento deliberativo: Cuando el agente encuentra un punto PK y se ubica, comienza su comportamiento deliberativo. Consiste en trazar planes mediante el algoritmo de A* hasta el objetivo, replanificando cuando es necesario.

  Del algoritmo A* se han modificado lo siguiente:
  - Se ha eliminado la comprobación de cerrados, por lo que es posible que la ruta no sea la más óptima, pero puesto que la heurística es aceptable (se usa la distancia Manhattan) siempre se encuentra una solución.
  - Calcula el coste del camino, es decir, devuelve el valor **g** del último nodo.
  - Si se indica que hay un aldeano delante suya, se altera temporalmente el valor de la matriz para que considere esa casilla como no válida.

Respecto a los aldeanos, en el momento en el que el agente descubre que la casilla a la que tiene que avanzar está ocupada (si fuera una posición no válida replanificaría automáticamente) se calcula el coste de un nuevo plan (considerando esa casilla como intransitable), si el coste de este es inferior al doble del actual se sustituye el nuevo plan por el anterior. En caso de que sea superior el bot considera que es mejor esperar a que el aldeano se aparte del camino, por lo que espera hasta que la casilla queda libre.

Desarrollo de la práctica

En un principio se utilizó el algoritmo de búsqueda en anchura (basado en el pseudocódigo del libro de Russell) hasta que era capaz de resolver el nivel 3 en mapa30. Luego se probó con un mapa de 100x100 pero le resultaba imposible calcularlo en menos de 5 minutos, por lo que sustituyó por el algoritmo A* (también del libro de Russell).

Cuando el algoritmo funcionaba correctamente, me planteé la mejora para la interacción con los aldeanos. Encontraba molesto (y menos inteligente) que en ocasiones diese rodeos cuando podía ser más rápido esperar a que el aldeano se apartase, por lo que modifiqué el algoritmo para que devolviese el coste del camino, y añadí la opción de espera en la función **think**.

Los principales problemas que tuve fueron de implementación, pues planteando las ideas en pseudocódigo la práctica se resolvía con mayor facilidad. En un principio tuve fallos por la inserción de los nodos en el plan (hacía push_back al retroceder en vez de push_front), pero el mayor tiempo de depuración se pasó revisando los índices (para no salirme de la matriz) y con el Mergesort, el cual se utilizó para ordenar la lista.