Práctica 1 TSI

Víctor Manuel Arroyo Martín
Grupo 3 de Prácticas

Comportamiento deliberativo

Nivel 1. Comportamiento deliberativo simple

Este nivel trata de, en un mapa con exclusivamente muros y un portal, llegar a este último en el menor número de pasos posible. Para ello, he creado un algoritmo A* que busca desde una posición inicial, el camino más corto a la posición final dada. Éste genera cuatro hijos, que son los cuatro posibles movimientos en cada tick: arriba, abajo, izquierda y derecha y los añade a la lista de abiertos si no son un muro. Cada nodo lleva su posición y orientación, que será la llave para el hash de cerrados. También he comprobado si se ha llegado a esa posición y a esa orientación con un camino más largo (es decir, está en cerrados), y si es así, elimino el nodo de cerrados y añado el hijo actual a abiertos, como es propio de un algoritmo A*.

Cabe explicar la clase nodo que se encarga, como he dicho anteriormente, de llevar la posición y la orientación. Pues bien, para simplificar la escritura del A*, también he hecho que calcule su h, g y f (lo cual es análogo a que se calcule justo después de crear un nodo más simple en el código de A*). Para la h he hecho la distancia Manhattan hasta el nodo final, la g es la del padre más uno, pues cada movimiento ocupa un tick y la f es la suma de ambas.

Por último, en el agente, he comprobado que no hubiera gemas ni enemigos para poder ejecutar este A* pasándole la posición y orientación del avatar como inicial y la del portal como final.

Nivel 2. Comportamiento deliberativo compuesto

En este nivel ha sido en el que más problemas de tiempo he tenido. Primero, probé a hacer un algoritmo A* que medía las distintas posibilidades de recorrer las 12 gemas con cada orientación. Para ahorrar tiempo, creé un hash antes de proceder con A* que guardaba todos los caminos entre gemas con todas las orientaciones (haciendo uso del anterior nivel) y a penas se iba un cuarto del tiempo en ello. Los nodos llevaban como h la distancia greedy que les quedaba hasta llegar al portal y su orientación, haciendo uso del hash anteriormente mencionado. Pero a la hora de ejecutar el A* me di cuenta de que eran demasiados nodos y tardaba cerca de 2 minutos en ejecutarse por completo.

Como segunda opción simplifiqué el A* aunque con ello no me diera la solución óptima y supuse en el hash del principio sólo orientación a la derecha y los nodos también ignorarían la orientación, con lo cual se reducía mucho su número. También para recortar aún más, los nodos que llevaban una g mayor que el valor del greedy iban directos a cerrados. Esto tardaba unos 18 segundos así que también era inviable para resolver el problema.

Con esos tiempos, la posibilidad de devolver el movimiento nil al principio y seguir con A* también era impensable así que lo que he hecho es una metaheurística basada en el Simulated Annearing con la que he encontrado buenas soluciones, muchas veces la óptima.
En este algoritmo lo que he hecho es, partiendo de una secuencia aleatoria de gemas y haciendo uso del hash del anterior párrafo, cambiar el orden en el que se van a recorrer dos gemas y si la distancia de recorrido de esta nueva secuencia es mejor que la anterior (aquí es donde se usa el hash para ir de gema a gema y calcular esa distancia de recorrido) se establece como secuencia actual esta nueva. Si no, con una cierta probabilidad que beneficia (que hace más probable) las grandes diferencias de distancia entre la mejor que tenías y la nueva, se revierte ese cambio de gemas. Todo esto se hará mientras quede tiempo, asegurándome de esta manera que voy a hacer el mayor número de cambios de gemas y que no me paso de tiempo.

Este algoritmo me daba siempre soluciones mejores que el greedy y varias veces el óptimo aunque como depende de la aleatoriedad, puede pasar que dé soluciones por debajo del greedy muy cercanas, pero aún así mejores por el gran número de cambios realizados de gemas.

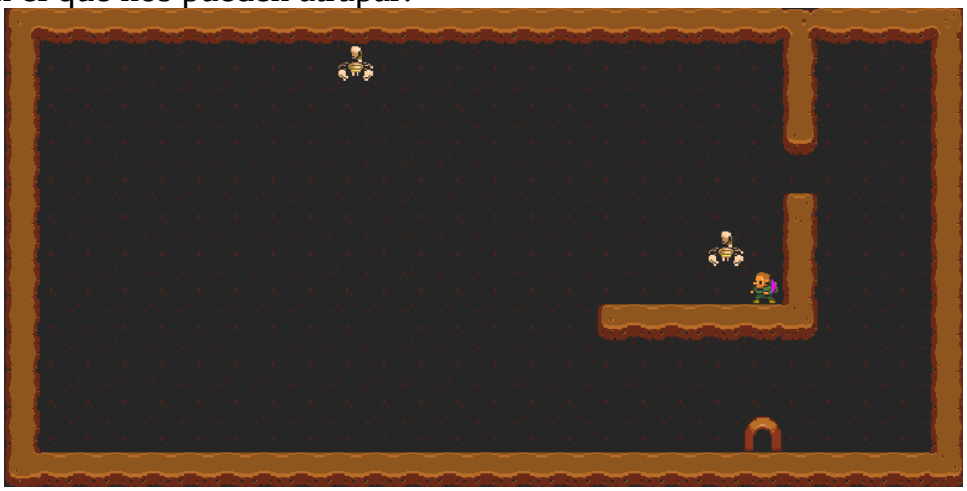
Comportamiento reactivo

Niveles 3 y 4. Reactivo simple y compuesto

Para estos dos niveles, he usado la misma estrategia.
Primero, si el avatar no está a menos de una cierta distancia euclídea del enemigo más cercano (5’5 en el código, tras hacer pruebas he visto que esta era la mejor) se mueve aleatoriamente por el mapa sin un rumbo fijo, ya que en este nivel sólo se pide aguantar los 2000 ticks que dura el juego.

En cambio, si se encuentra a menos de esa distancia, lo que hace es huir. Para ello, se generan las ocho casillas adyacentes y se mira cuál de ellas es una casilla que llamaremos segura a la que ir.
Una casilla la consideraremos segura si al movernos a ella, la distancia a la que estaremos del enemigo más cercado desde ella es menor que en la que estamos ahora. Añadimos las casillas adyacentes seguras a un array y comprobaremos si en ese array está la casilla que tenemos justo delante, es decir, la que está hacia donde estamos mirando para así consumir sólo un tick. Si no está, se elige una casilla adyacente que no esté en diagonal respecto del avatar para ahorrar ticks, y si sólo hay casillas seguras diagonales, se toma una aleatoria. Se usa A* para llegar a esa casilla y se devuelve la secuencia de movimientos al agente.
Si no hubiera casillas seguras quiere decir que nos movamos hacia donde nos movamos nos acercaremos al enemigo así que lo único que nos queda es quedarnos quietos y comprobar hacia dónde se mueve el enemigo para huir en el siguiente tick.

Hay algunas situaciones en las que estamos en una esquina y el enemigo está justo en frente. Si en ese momento coincide que el enemigo va hacia nuestra posición, lo más seguro es que nos alcance ya que él no tiene orientación y nosotros un tick para orientarnos así que nos caza. También ocurre igual si dos enemigos nos tienen acorralado contra una pared. Exceptuando estas situaciones, en las que no es posible huir siempre de ninguna forma, en todas las demás escapa.
El agente sabe que está en el nivel 3 o 4 si sólo hay enemigos y no hay gemas.
Ejemplo en el que nos pueden atrapar:



Comportamiento reactivo-deliberativo

Nivel 5.

Ahora nos pedían una fusión de todos los niveles anteriores. Para ello, el agente comprueba si hay enemigos y gemas a la vez.
El procedimiento seguido es calcular en el constructor del agente el camino de las gemas mediante Simulated Annearing, como en el nivel 2, ignorando los enemigos ya que se moverán aleatoriamente por todo el mapa. Después, en cada tick, comprobará si hay que escapar de un enemigo, en tal caso se procederá como en los niveles 3 y 4 y tras haber escapado, sigue con el camino normal de gema a gema.

También hay una situación en particular que he tenido en cuenta, y es que si mientras estamos escapando de un enemigo cogemos una gema, eliminaremos ésta de la secuencia de gemas y pasaremos directamente a la siguiente cuando fuera su turno.