TSI

PRACTICA 4

David Erena Casado 3ºCSI Grupo 1

DELIBERATIVO:

Como algoritmo de búsqueda heurística, he elegido A*, y he usado las transparencias de teoría como referencia para implementar el algoritmo de búsqueda heurística.

Pseudocódigo del algoritmo implementado:

Si el plan no está vacío, retornamos la última acción y la eliminamos de este.
Insertamos el nodo inicial en abiertos.
Mientras no superemos el tiempo establecido:

1.- Si abiertos está vacío, salimos del bucle.
2.- Seleccionamos el mejor nodo de abiertos.
3.- Si es nodo solución, llenamos plan. Devolvemos y eliminamos la última acción del plan.
4.- Generamos hijos del nodo actual.
5.- Para cada hijo:
· Si está en abiertos, actualizamos con el mejor padre (mejor valor de F).
· Si está en cerrados, reinsertamos en abiertos si hay una mejora en F, y actualizamos descendientes en abiertos.
· En otro caso, insertamos en abiertos.
6.- Añadimos el nodo actual en cerrados.

Para hacer la ejecución más rápida y eficiente, he realizado las tres medidas siguientes:

1.-Para la lista de nodos abiertos, uso una cola con prioridad que guarda los nodos según el valor de f de cada uno en orden ascendente, de forma que podemos acceder en O(1) al mejor nodo disponible.

2-En cuanto a la lista de nodos cerrados, uso una conjunto hash para guardar los nodos según un código hash que proporcionemos.
Esto nos proporciona una forma de ver si un nodo pertenece a cerrados con eficiencia O(1) (aunque si queremos buscarlo, para comprobar su f por ejemplo, necesitamos recorrer el conjunto).

3.-Además, para ahorrar enormemente en la cantidad de nodos a generar, los cambios de orientación y un movimiento en esa nueva orientación se generan como un único nodo.

Sobre los nodos, estos contienen su valor de coste, heurística y f, la posición y orientación que representan, su nodo padre, y la acción (o acciones, si se necesita cambiar de orientación) para llegar del padre a él.

Como método de calcular distancias, y por tanto, base de las heurísticas consideradas, he usado la distancia Manhattan:

distanciaManhattan(n1, n2) = |n1.x – n2.x| + |n1.y – n2.y|

De esta forma, tendremos una heurística admisible para el nivel 1, es decir, distanciaManhattan(n, objetivo) <= h*(n)

Esto es porque, primero, el jugador se puede mover únicamente en horizontal o vertical, y segundo, el movimiento del jugador solo puede ser en la dirección en la que esté mirando, es decir, necesita un cambio de orientación para poder moverse en otra. Por esto, la distancia mínima entre el jugador y un punto será la distancia Manhattan. De forma práctica, en el caso de que no haya obstáculos hasta un objetivo, el jugador tendrá que ir primero hacia la izquierda (o derecha) y luego arriba (o abajo) o al contrario para llegar a él, ya que esto minimiza los cambios de orientación a efectuar, que son pasos que no mueven y, por consiguiente, no aportan tanto.

A consecuencia de esto último, sabemos que cuando se llegue a un nodo, se hace por el camino de menor coste. Gracias a esta propiedad, podemos descartar el pseudocódigo anteriormente especificado para A*, reemplazándolo por:

Si el plan no está vacío, retornamos la última acción y la eliminamos de este.
Insertamos el nodo inicial en abiertos.
Mientras no superemos el tiempo establecido:
1.- Si abiertos está vacío, salimos del bucle.
2.- Seleccionamos el mejor nodo de abiertos.
3.- Si es nodo solución, llenamos plan. Devolvemos y eliminamos la última acción del plan.
4.- Generamos hijos del nodo actual.
5.- Para cada hijo:
· Si no está en cerrados, se añade a abiertos.
6.- Añadimos el nodo actual en cerrados.

Gracias a este enorme ahorro, cobra sentido el uso del conjunto hash para almacenar los nodos cerrados. Anteriormente, tendríamos que recorrerlo en caso de encontrar un nodo dentro, con lo cual no lo diferenciaba mucho de usar un vector “común”, por así decirlo. Pero ahora, al ahorrar esa posible búsqueda dentro, sí que conviene que la comprobación de que un nodo pertenezca a cerrados sea O(1).

SIMPLE:

Para el comportamiento simple, al tener un único nodo objetivo, uso como heurística la distancia Manhattan desde el nodo hasta la puerta objetivo, como se comentó anteriormente.

COMPUESTO:

He considerado varias heurísticas para este comportamiento:

1.- Greedy: Cogemos la mínima distancia de una gema al nodo.

2.- Greedy anticipando una gema (distancia mínima) : Consideramos el valor heurístico de un nodo como la menor suma de la distancia de este a una gema más la menor distancia de esta a otra gema.

h(n) = min(manhattan(n , g) + min(manhattan(g , g'))) , siendo n nodo, y g, g' gemas.

3.- Greedy anticipando una gema (distancia máxima): Consideramos el valor heurístico de un nodo como la menor suma de la distancia de este a una gema más la mayor distancia de esta a otra gema.

h'(n) = min(manhattan(n , g) + max(manhattan(g , g'))), siendo n nodo, y g, g' gemas.

En cuanto a resultados, la que mejores dio fue la opción 3. Esto puede ser porque con la mínima distancia estamos haciendo una ampliación del Greedy normal, con lo que eso conlleva (buenas elecciones a corto plazo, malas a largo). Sin embargo, con la máxima distancia, estamos aceptando lo que conlleva usar una filosofía Greedy e intentamos paliarlo, mediante la elección del mejor de los peores casos a los que nos pueda llevar.

(Dentro de los ficheros, ambos comportamientos se encuentran dentro de la misma función heurística).

REACTIVO:

Tanto para el reactivo simple como compuesto, hago uso de un mapa (mapaPrioridades en el código) que alberga un entero para cada posición del nivel, y que indica la prioridad para moverse a dicha posición. A mayor valor, más prioridad (multiplica el valor heurístico de un nodo por la mitad del valor de su posición en el mapa) .
Esto, en el caso de mi agente, se traduce en que se valoran de forma negativa las posiciones adyacentes a paredes y, especialmente, esquinas y sus alrededores.
Mapa usado:

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 0]
[0, 2, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 2, 0]
[0, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 0]
[0, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 0]
[0, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 0]
[0, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 0]
[0, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 0]
[0, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 0]
[0, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 0]
[0, 2, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 2, 0]
[0, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Cabe señalar que de haber huecos estrechos en el mapa, con dos o menos espacios libres dentro, estos también se valoraran negativamente, ya que no conviene que el agente los trasverse.

SIMPLE:

La forma de actuar del agente es que escoge el movimiento que más vaya a alejarlo del enemigo.
Para cada movimiento posible, calculamos la posición a la que nos movería y la distancia entre esta y el enemigo. Cogemos la que más distancia nos de.
Hay un par de aspectos que no he tenido en cuenta, pero viendo que sin hacerlo, se seguían obteniendo resultados muy favorables, no he visto conveniente intentar tratarlos:

- Cambios de orientación, es decir, da igual la orientación, todos los movimientos se consideran igual.
- El agente se mueve de izquierda a derecha de forma contínua entre 2 posiciones cuando llega a un lugar seguro.

COMPUESTO:

Al igual que en el nivel anterior, generamos la posición al efectuar un movimiento (sin tener en cuenta la orientación).
Para cada posición nueva, cojo la menor distancia de un enemigo a esa posición. Luego, de estas distancias, cojo la mayor.

Opté por esta idea tras descartar la de maximizar la suma de todas las distancias entre enemigo y jugador (No dio buenos resultados tras unas cuantas pruebas).
Esta opción prioriza evitar el peligro más cercano. Da buenos resultados, pero diría que podría sufrir al estar entre dos enemigos a igual distancia ya que no tiene en cuenta la posición del otro enemigo, aunque este tipo de situaciones no suele darse muy a menudo, o durante muchos ticks.

(Similar a la heurística en el comportamiento deliberativo, dentro de los ficheros fuente, ambos comportamientos surgen de la misma función).

DELIBERATIVO – REACTIVO:

Para este apartado, uso el comportamiento deliberativo todo lo posible hasta que se sobrepase un perímetro de seguridad, en este caso, cuando un enemigo esté a 4 pasos o menos de el agente.
Cada vez que se tenga que usar el comportamiento reactivo, limpio el plan que se haya hecho y las listas de abiertos y cerrados que hubiese en el momento, ya que cuando vuelva a usar el comportamiento deliberativo, el jugador puede que esté en una posición muy diferente a la que estuviese antes, y por tanto, necesita re-planificar.