Práctica 1: Técnicas de Búsqueda Heurística:

UNIVERSIDAD DE GRANADA

Luis González Romero, XXXXXXXXX, luisgonromero@correo.ugr.es
Grupo 2: Viernes 17:30-19:30

Escuela Técnica Superior de Ingeniería informática y Telecomunicaciones

5 de abril de 2020

Práctica 1: Técnicas de Búsqueda Heurística

Memoria sobre la primera práctica de la asignatura Técnicas de los Sistemas Inteligentes cursada en la ETSIIT, de la UGR.

Luis González Romero, XXXXXXXXX, luisgonromero@correo.ugr.es

Abril de 2020

Índice

Índice de figuras

1. Comportamiento deliberativo

1.1. Comportamiento deliberativo simple

En el deliberativo simple he usado el algoritmo *A** para explorar los nodos y encontrar el camino óptimo en coste de casillas. He usado este algoritmo porque es bastante rápido para nuestro tamaño del problema y no priorizo el uso de memoria que se podría ganar con *IDA**. Por lo que desarrollé una clase *PathFinder* para encontrar ese camino en base a la posición de partida(en este caso la del avatar) y la posición final(en este caso la del portal), trabajando con nodos como precisa *A** gracias a una clase *Node* que he creado. Usando para la lista de nodos abiertos y cerrados la clase *PriorityQueue*, gracias al *compareTo* del coste total de cada objeto de tipo *Node*.

La implementación del comportamiento es bastante sencilla teniendo el método que calcula el camino óptimo, simplemente vamos obteniendo las acciones en base al camino que obtengamos. Este método simula el comportamiento clásico del *A**, pero sin computar las casillas que son muros.

1.2. Comportamiento deliberativo compuesto

Este comportamiento se basa en su modalidad simple, reutilizando su funcionalidad(reutilización de código) para encontrar una gema. Cuando se encuentra una gema se recalcula el camino a la siguiente gema y así hasta que el avatar haya recogido las diez gemas. En cuanto no queden gemas se dirigirá al portal. Cada vez que se dirige a una gema, se ha comprobado el coste a cada una de las gemas y se ha escogido la que tiene el menor.

Las gemas se obtienen por proximidad al avatar por ahorrar tiempo ya que en los comportamientos posteriores se añadirá la complejidad de la presencia de enemigos con mapa de calor.

2. Comportamiento reactivo

Para el reactivo he creado una clase *Heatmap* para poder tener en cuenta la presencia de enemigos. Cada casilla de la matriz estará a 0 por defecto, y cuando hay enemigos en el mapa se genera su calor en base a su posición actual, que tomará el valor de la constante definida para el calor disminuyendo según nos alejamos de este.

Cada vez que se ejecuta *act* se comprueba si hay enemigos y se genera el mapa de calor correspondiente.

Como un mapa de calor puede ser representado con rangos de colores lo he cambiado a rango de números: cuanto mayor sea el número, mayor es el riesgo y menor la distancia con el enemigo.

2.1. Comportamiento reactivo simple

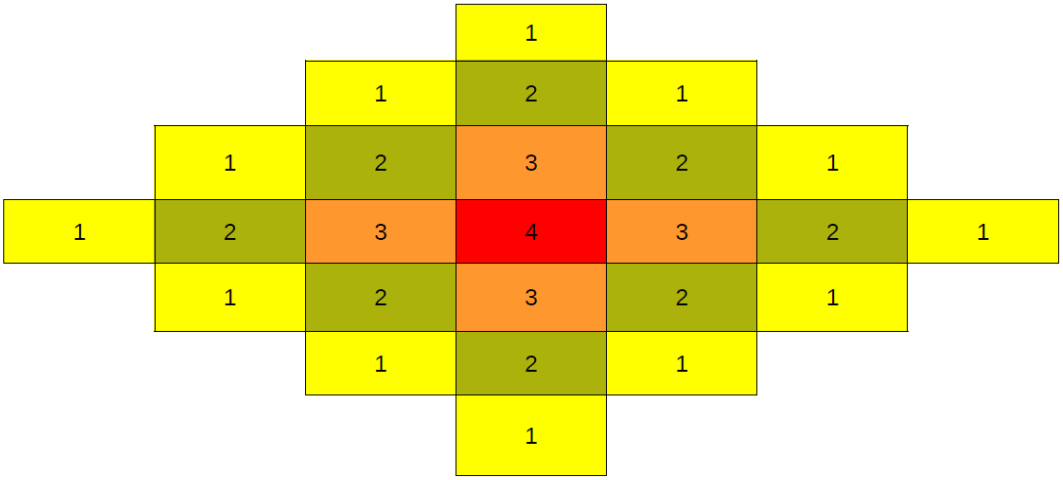

Figura 1: Mapa de calor generado para un enemigo con rango 0-4

Cuando el avatar se encuentra con una casilla con calor mayor que 0, se plantea a donde debe dirigirse para salir de la zona de peligro. Para esto compruebo en donde se encuentra el enemigo con respecto al avatar usando el método *scapeFrom()*.

El avatar siempre intentará escapar a una zona sin muros y con el menor valor de calor en las casillas alcanzables por este.

2.2. Comportamiento reactivo compuesto

En el caso del compuesto se añadió que cuando se superpongan los valores de calor de los enemigos, el valor de la casilla se sume y tomando como referencia para huir al enemigo que más calor aporta al mapa. Así las casillas tendrán un peligro mayor según cuantos enemigos pueden ir a esa casilla.

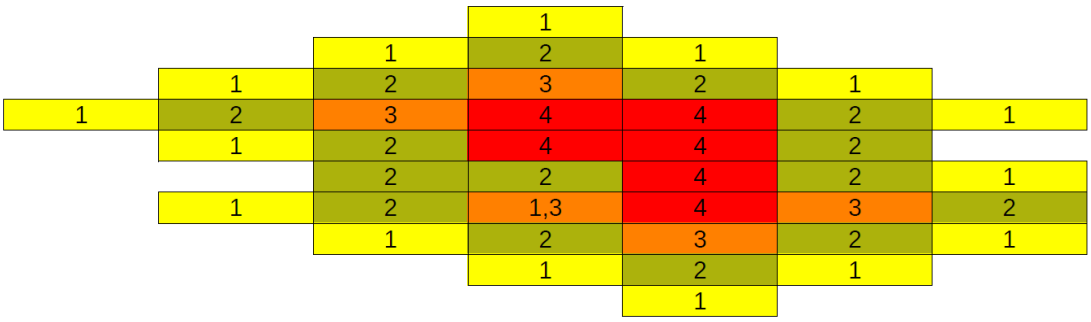

Figura 2: Ejemplo de superposición del calor generado por dos enemigos

El comportamiento sigue el mismo principio que el anterior, alejarse de la presencia de peligro. Siempre que acabe el avatar en una casilla con un calor superior a 0 tratará de tomar la decisión de ir a una casilla con el menor calor.

3. Comportamiento reactivo-deliberativo

En este caso el comportamiento lo he simulado transicionando entre los posibles estados: reactivo, deliberativo-simple, deliberativo-compuesto. Dando prioridad al reactivo, para escapar de un enemigo en cuanto estemos dentro de una zona con peligro.

Si no está en peligro pasará a dirigirse al portal(solo una vez, ya que gana si se llega. En el caso de no ganar sería porque esta probándose su habilidad de escape). Si detecta que hay gemas sin recoger, procederá a recogerlas con el estado deliberativo-compuesto. En cuanto recoja todas ya solo le quedaría el portal, por lo que tendrá abierta la opción de transicionar al estado deliberativo-simple.