UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

Memoria Práctica 2

Agentes Deliberativos: Los Extraños mundos de Belkan

Inteligencia Artificial

Torres Ramos, Juan Luis

2ºC - Grupo 2

12 mayo de 2022

Nivel 0

En este nivel implementamos el algoritmo de búsqueda de profundidad, que es la demo de la práctica, debemos seguir el tutorial. Comprendemos cómo va el algoritmo. Vamos expandiendo todos los nodos que se van localizando, de forma recurrente en un camino concreto. Cuando ya no quedan más nodos que visitar en ese camino, damos media vuelta y repetimos el proceso por todos los nodos hermanos

Nivel 1

Nos pide calcular el número de acciones mínimo para llegar a un objetivo, para ello aplicaremos el algoritmo de búsqueda de anchura. Funciona similar al algoritmo de búsqueda de profundidad, sin embargo visitamos los nodos de manera distinta en vez de usar un stack para ordenar los nodos abiertos, usaremos una queue para recorrer los nodos. La profundidad se basa en el contenedor LIFO y la búsqueda de anchura se basa en FIFO. Vamos viendo los nodos nivel por nivel. También hay que tener en cuenta que ahora usamos el método top de la queue antes que el front del stack.

Nivel 2

Para el nivel 2 he implementado los algoritmos de CU y A*

Primero realizamos una función de cálculo coste para saber el coste de un nodo. Ahora el nodo de CU o A* tiene el valor de coste. También , a la hora de escoger el camino mas optimo veo si tiene zapatillas o tiene bikini el nodo para ir por bosque o agua, para ello al igual que orientación, fila, columna, lo comparó también en el ComparaEstado.

Para el A* al igual que el CU sin embargo en la cola de cerrado y abierto no tratamos con estados sino con los nodos mismos, ya que al final tenemos que ver si en la lista de cerrado hay nodos repetidos , para buscarlo en la cola de cerrados uso la función find, que antes la cola rellena de estados no podía.

La función manhattan está en el problema sin embargo no la uso, devuelve 0, ya que estuve haciendo comprobaciones de cómo implementar el algoritmo y si hacía falta, sin embargo como el muñeco se puede mover en diagonal tmb no hace falta dicha función

Nivel 3

Lo primero que he hecho es una función de Calcular Punto para que cuando en los sensores encuentre una casilla bikini, zapas o recarga vaya hacia a ella, teniendo en cuenta un tiempo de intervalo. Busca casillas cuando el tiempo de intervalo se ha acabado, También he hecho una función de emergencia para habilitar los sensores en caso de que coja un bikini rodeado de bosque, por lo que debo de habilitar los sensores para salir ya que las zapatillas las habré perdido. También actualice el ver el mapa para que vea en diagonal y el muñeco hace la función de recargar.
Primero se comporta como un agente reactivo , moviéndose por el mapa de forma aleatoria explorando, se comportara deliberativa mente cuando encuentre una casilla de interés, por lo que se moverá a ella calculando su punto y llamando a pathfinding, luego vuelve a su carácter reactivo

Nivel 4

Aquí he implementado una función para calcular la posición del muñeco ya que los sensores no funcionan, si no no puedo pintar bien el mapa, para ello primero se hace siempre un whereis cuando no se sabe dónde está. La primera acción siempre es un whereis, luego como no hay objetivos se calcula una tanda, cuando objetivos este vacío vuelvo a calcular más objetivos. El muñeco calcula el plan para el primer objetivo, y va descubriendo el mapa, cuando no puede avanzar, hay algún obstáculo el muñeco se encarga de recalcular de nuevo un plan con los nuevos datos que ha ido recogiendo visualizando el mapa y así para todos los objetivos. En caso de colisión, se resetean los valores a 0 y posteriormente se hace un whereis para volver de nuevo a un estado correcto. El muñeco no puede moverse ni pintar mapa mientras se haya chocado, el sensor colisiona este vacío. Si ha llegado al objetivo se pasa al siguiente