UNIVERSIDAD DE GRANADA

MEMORIA PRÁCTICA 2: Los extraños mundos de Belkan

Inteligencia Artificial

Asignatura: Inteligencia Artificial
Grupo: 2ºC

Mena Barrera, Miguel Ángel

Introducción a la práctica:

En esta práctica tomamos como punto de partida el mundo de las aventuras gráficas de los juegos de ordenador para intentar construir sobre él personajes virtuales que manifiesten comportamientos propios e inteligentes dentro del juego. Intentamos situarnos en un problema habitual en el desarrollo de juegos para ordenador y vamos a jugar a diseñar personajes que interactúen de forma autónoma usando agentes reactivos/deliberativos.

Que he hecho: ***(solo hasta nivel 2)***

Para esta practica he llegado hasta el nivel 2, he implementado el algoritmo A* cuyo psedocodigo es el siguiente:

ABIERTOS := [INICIAL] //inicialización
CERRADOS := []
f'(INICIAL) := h'(INICIAL)
repetir
si ABIERTOS = [] entonces FALLO
si no // quedan nodos
extraer MEJORNODO de ABIERTOS con f' mínima
// cola de prioridad
mover MEJORNODO de ABIERTOS a CERRADOS
si MEJORNODO contiene estado_objetivo entonces
SOLUCION_ENCONTRADA := TRUE
si no
generar SUCESORES de MEJORNODO
para cada SUCESOR hacer TRATAR_SUCESOR
hasta SOLUCION_ENCONTRADA o FALLO

En mi caso para la implementacion del A* he utilizado basicamente dos estructuras de datos: *priority_queue* y *set.* La cola con prioridad almacena la lista de abiertos y en el set tenemos los estados que ya hemos explorado, usando el set nos evitamos duplicarlos. En la cola con prioridad hemos metido objetos del struct Movimiento que acmacena la posicion, plan y la f (para la heuristica). Exploramos los nodos (avanzo. izquierda y derecha) se insertan en abiertos y pasamos a cerrados el actual. Si posicion == destino quiere decir que hemos encontrado un plan por lo que devolvemos true, si por tanto no encontrase plan se devolveria false.

Para facilitar la programación de el algoritmo he programado varias funciones auxiliares para ver si es posible avanzar, avanzar y ambos giros. Tambien dos funciones adicionales al A* para sacar fuera del codigo de path finding las exploraciones de los nodos derecha e izquierda.

Para resolver las colisiones con el aldeano basicamente lo que he introducido es una espera, no se regenera el plan si te topas con uno de ellos, simplemente esperas a que se mueva y te permita avanzar.

Problemas encontrados

Implementando el A* los problemas que he encontrado basicamente han residido en las estructuras de datos que he tenido que usar, al principio he tenido problemas con la insercion en el set por no acordarme de las sobrecargas tanto para la priority queue como para el set.

Eliminé erroneamente parte de codigo del think y generaba cores aleatoriamente, fue un fallo dificil de detectar