UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

Memoria Práctica 1

Agentes Reactivos: Los Extraños mundos de Belkan

Inteligencia Artificial

Torres Ramos, Juan Luis

2ºC - Grupo 2

3 abril de 2022

En esta memoria voy a explicar el procedimiento que he seguido para definir el movimiento del agente reactivo, hablaré un poco de las variables más importantes y de las funciones más importantes que he requerido.

1. Variables

- ➔ fil (int) - corresponde a sensores.poF
- ➔ col (int) - corresponde a sensores.posT
- ➔ brújula (int) - corresponde a sensores sentido [norte 0, este 1,sur 2, oeste 3]
- ➔ ultimaAccion (action): devuelve la última acción
- ➔ bien_situado (boolean): true me permite escribir en el mapa
- ➔ modo_aleatorio (boolean): true modo aleatorio
- ➔ modo_busqueda (boolean): true modo buscador
- ➔ girar_derecha- (bool) true gira a la derecha
- ➔ bikini(boolean) true si está en la casilla de bikini
- ➔ zapatillas(boolean) true si está en la casilla de zapatillas
- ➔ recarga (boolean) true si está en la casilla de recarga
- ➔ comienzo (boolean): para que no de un paso al principio
- ➔ posicionamiento (int): posición guardada cuando he encontrado una casilla
- ➔ contador (int): si llega a 0 habilito bikini y zapatillas.

2. Procedimiento

Para comenzar, necesitare definir el rango de visión de mi agente, para esto creó la función de Actualizar Mapa(); donde, según la posición de la brújula, escribo en mapaResultado los sensores terreno que tenga enfrente para registrar las zonas de terreno que ve mi agente. Para el nivel 0, como los sensores de posición y brújula no he de preocuparme, pero en el resto de niveles, no podré escribir en la matriz hasta que no pase por una casilla azul, para ello creo la funcion de sensor vista_nivel para diferenciar estos dos casos.

Sensor VistaNivel(Sensores sensores), si estamos en el nivel 0 la variable que permite escribir en el mapa bien_situado está a true, sino sensores.terreno[0] == G , que esté en una casilla azul y actualizo fil y col y bien_situado a true.

Tras hacer esto, he optado por que el robot tenga dos modos de funcionamiento, uno en el que busque las casillas especiales si lo capta por los sensores de posicionamiento y vaya a esta y otro modo en el que el agente se mueva aleatorio, cuando llegue a una pared, entre una probabilidad de 50% gire a la izq o a la derecha. EL agente empieza en modo aleatorio. SI no está en modo busqueda, esta en modo aleatorio

MoverAleatorio(); Está compuesto de varias subfunciones, Primero si recarga = true hace la acción de Recargar(); [El agente se espera el tiempo de recarga de 5 segundos sin hacer nada, luego se mueve]. Si no comprueba los sensores, mi función SensoresAvanzar() donde reconoce por donde puede ir el agente, además implementó las condición de zapatillas y bikini, es decir, si las lleva puesto puede pasar por bosque o agua para que así no gaste demasiada batería. Si no el agente gira con la función Girar, gira a la izquierda o derecha según la variable girar_derecha, que la actualizo luego con girar_derecha = (rand() % 2 == 0);

Mover_Busqueda();, Para hacer que el agente vaya a una casilla que queramos, nos basamos en el uso de los sensores terreno, si en la visión del agente ve una casilla de posición G, una de bikini K o una de recarga X, se guarda la posición del terreno con la función VistaAgente(), En vista Agente tenemos 4 variables booleanas G_encontrada, D_encontrada, K Encontrada y X_ Encontrado si encuentra una de estas casillas, recuerdo que con una vez que encuentre una de estas casillas es suficiente. Tras encontrar la casilla, la guardo en la variable posicionamiento y cambio a modo búsqueda cambiando su variable booleana.
La acción que va a hacer lo determina MoverBusqueda(); Esta función, primero de todo calculamos el número de filas y columnas que habría que avanzar para llegar a la casilla, luego avanza las x filas que tenga que avanzar y cuando se hayan acabado, gira dependiendo de la variable de posicionamiento a la izquierda o derecha, y por último avanza x columnas restantes, cuando llegue a la casilla, el modo búsqueda pasa a false. Las va buscando en orden, primero busca la casilla azul, luego la casilla de zapas, luego la casilla de bikini y por último la casilla de recarga, si lo busco todo a la vez, hay casos en los que el agente se confunde.

Por último, para los lobos y aldeanos, primero de todo tenemos que considerarlos como muros ya que si no nos produciría un segmentation fault. Para los lobos en caso de que nos coma , con el sensor de vida, si este esta true llamamos a una función de reinicio

También tengo un contador que se resta cuando ha hecho una acción repetida, cuando baje a cero habilito zapas y bikini a pesar de que vaya a gastar más batería. Esto lo hago en el caso de que un lobo me coma y aparezca en el bosque.

Si mi muñeco tiene la bateria baja, decrementa un contador que hara que el muñeco gire más durante el transcurso de la partida

3. RESUMEN

Belkan, lo primero que hace es moverse aleatoriamente por el mapa, cada vez que choque gira a la izquierda o a la derecha de forma aleatoria y va buscando las casillas especiales, la primera que busca es la de posición, si la encuentra con sus sensores va hacia ella, es decir, pasa al modo de búsqueda, luego vuelve al modo aleatorio y busca la siguiente casilla, la de zapatillas, luego la de bikini y por último busca la de recarga, solo pasa 1 vez por posicionamiento, zapatillas y bikini y por recarga puede pasar cada vez que la encuentre en el mapa. Si un lobo se come a belka, este aparece en un sitio nuevo con sus variables reiniciadas y tendrá que volver a buscar las casillas dichas anteriormente. Para terminar, cuando Belkan tiene batería baja, en el modo aleatorio, a parte de que gire cada vez que choque, girará cada vez que de 10 pasos. Belkan en un principio no puede pasar por bosque o agua hasta que no pase por la casilla. Si Belkan se queda pillado en algún lugar sin hacer nada o girando, se habilita zapatillas y bikini para que pase por todos lados