Practica 2: Agente deliberativo

Por: Arturo Cortés Sánchez

Comportamiento del agente: El comportamiento del agente viene definido por dos funciones, Reactivo() y Deliberativo(), las cuales se van alternado a lo largo de la ejecución según las siguientes condiciones. Mientras el agente no esté bien situado o el tiempo sea inferior a cierto umbral se llamará a Reactivo(). En caso contrario se comprueba que el terreno sobre el que esta el agente no es bosque o agua y se llama a Deliberativo(), de ser agua o bosque, se vuelve a llamar a Reactivo(). El umbral mencionado anteriormente es inicialmente 150, pero luego es modificado para ser igual al tiempo mas el logaritmo del tiempo por diez, he elegido la función logaritmo ya que necesitaba que el umbral aumentase con el tiempo pero no demasiado, y la lentitud de crecimiento de los logaritmos es ideal en este caso.

E agente va guardando el mapa resultado, un mapa de prioridades en función de donde ha pasado, y un mapa de la superficie. Para cada mapa hay ademas un mapa temporal en en que se escribe hasta que el agente se orienta, una vez orientado, dichos mapas temporales son copiados a los pricipales

La función Reactivo() consta de una serie de condiciones según las cuales se van llamando a avanza() o gira() (funciones que gestionan el movimiento basándose en la matriz de prioridades) o directamente a una acción en caso de tener un objeto delante. Por ejemplo, si tiene un bosque o agua delante y ha cogido unas botas o un bikini, tratará de sacarlo de la mochila para seguir avanzando. O si se encuentra con un objeto de interés (solo bikini, botas o regalo, he optado por ignorar el hueso y la llave para poder intercambiar objetos fácilmente) lo cogerá en caso de no tenerlo

La función Deliberativo() inicialmente comprueba si está ejecutando un plan, si esta ejecutando uno se limita a devolver las acciones de este. En caso contrario mira si el agente tiene un regalo, si lo tiene, buscará al rey mas cercano y hará un plan para ir a entregárselo, si no lo tiene, ordena los regalos por cercanía e intenta hacer un plan para ir a por el mas cercano, si no puede prueba con el siguiente y asi sucesivamente. En caso de haber recorrido el vector de regalos ordenados sin éxito, avisará de que no hay regalos accesibles o son demasiado lejanos, lo cual activará el comportamiento reactivo temporalmente.

Si el agente está ejecutando un plan y se encuentra un obstáculo delante, comprueba si es un objeto útil, y lo coge en caso de no tenerlo, si es un aldeano o un lobo esperará a que este se aparte, y si es un rey, comprobará si tiene un regalo y se lo dará. Si el obstáculo es cualquier otra cosa aborta el plan

Para el algoritmo de búsqueda he optado por el A*, para ello he creado un nuevo struct llamado estadoP que consta de un estado, del plan para llegar hasta ese estado, y del peso, el cual es la suma de el tamaño del plan del estadoP y la distancia hasta el objetivo. Ademas he sobrecargado el operador < para que compare los estadosP en función a su peso. Para el algoritmo he creado una cola con prioridad de estadosP llamada abiertos y un set de estados llamado cerrados y una serie de estadoP con los que trabajar. Inicialmente guardo el estadoP origen en la cola de abiertos y entro en un bucle while que acaba cuando se acaba abiertos, se encuentra el destino o el algoritmo supera las 5000 iteraciones.

Dentro del bucle expando el primer estadoP de abiertos y guardo su estado en cerrados y antes de meter a los hijos en abiertos compruebo que no estén en cerrados, ademas, en el caso del hijo generado por el operados avanzar, compruebo que el terreno de dicho estado es pisable o que es solución.

Cuando el bucle acaba se comprueban las iteraciones para ver si ha acabado por el limite de iteraciones. Si es asi, se devuelve false, en caso contrario se devuelve como plan el plan asociado al primer nodo de abiertos.