TC2008B. Modelación de Sistemas Multiagentes con Gráficas Computacionales - Actividad Integradora

Diego Carrillo Torres – A01612532

Parte 1. Sistemas multiagentes

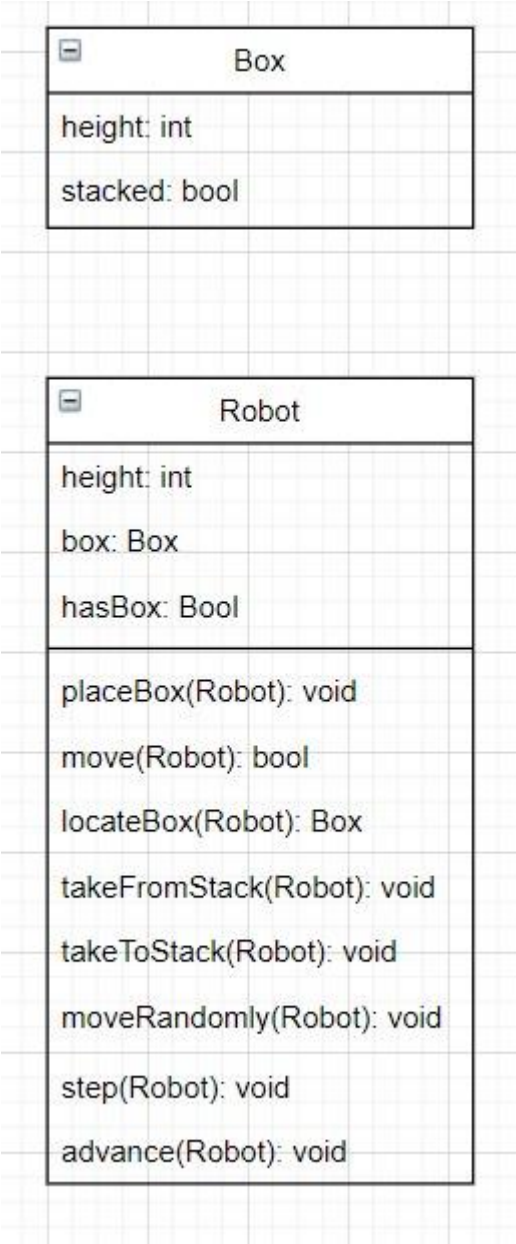
Estrategia que podría disminuir el tiempo dedicado y cantidad de movimientos realizados

Me parece que es posible implementar algunas estrategias para disminuir el tiempo que dedican los agentes en organizar las cajas, así como la cantidad de movimientos que realizan, por ejemplo, la primer estrategia que considero optimizaría el trabajo de los robots es que tuvieran la capacidad de leer y hacer movimientos en diagonal, es decir en las 8 direcciones posibles que los rodean, en lugar de solamente 4, ya que esto me parece una limitación importante, ya que hay que realizar prácticamente el doble de movimientos para llegar a lugares que podrían tomar solo uno. Otra estrategia que optimizaría el trabajo es que la capacidad de carga de los robots fuera mayor, es decir, que puedan cargar con más de una caja a la vez, ya que en ocasiones los robots pasan cerca o por donde hay cajas mientras van cargando una, y lo mejor es que se aprovecharan esos movimientos y que cargaran las cajas que se encuentran en el camino. También, si los robots tuvieran la capacidad de encontrar cajas a mayor distancia el trabajo sería optimizado considerablemente, ya que darían menos movimientos en vano al tener la capacidad de encontrar un objetivo a mayor distancia. Por último, la estrategia que me parece que podría tener más impacto es que los robots tuvieran la capacidad de memorizar en que espacios no hay cajas, y compartan esa información entre todos los robots, para que así, busquen donde aún no ha buscado ninguno de los otros robots.

TC2008B. Modelación de Sistemas Multiagentes con Gráficas Computacionales - Actividad Integradora

Diego Carrillo Torres – A01612532

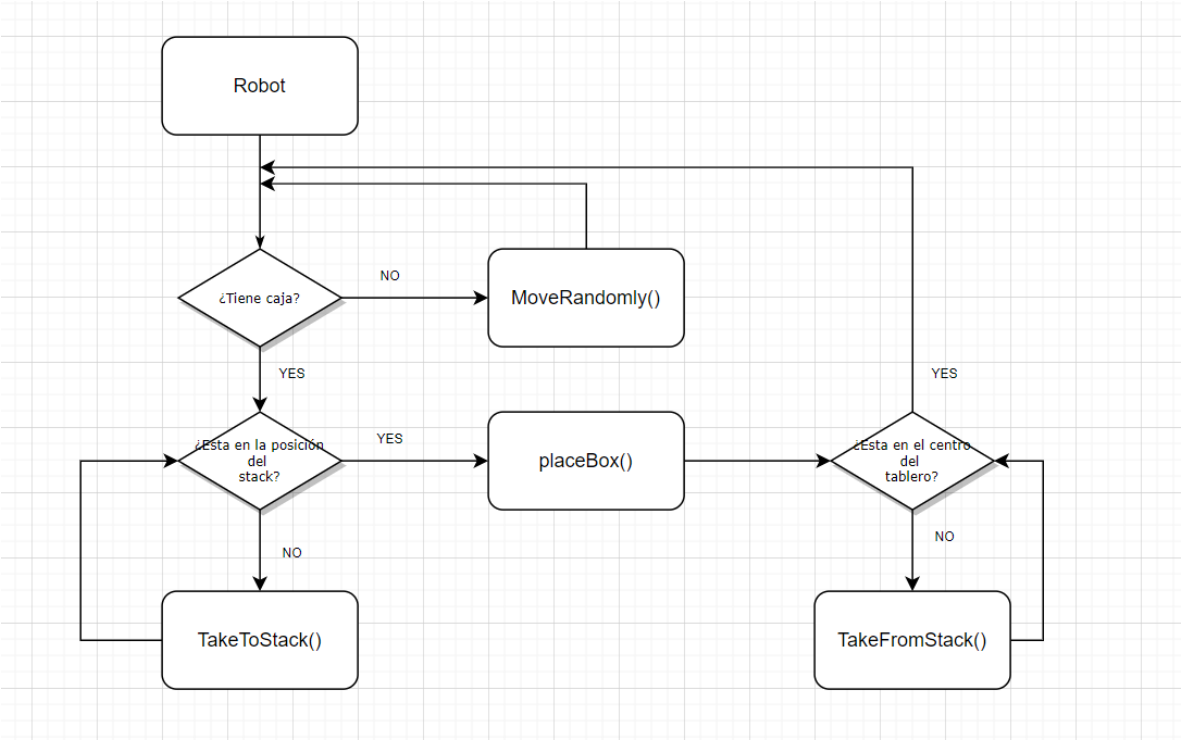
Diagrama de clases



TC2008B. Modelación de Sistemas Multiagentes con Gráficas Computacionales - Actividad Integradora

Diego Carrillo Torres – A01612532

Protocolo de agentes



Estrategia cooperativa

Al iniciar la ejecución, todas las *k* cajas aparecen en posiciones aleatorias generadas en una matriz. De igual manera, se crean 5 instancias de agentes (robots) también ubicados en posiciones aleatorias dentro de la misma matriz. En caso de que haya un 0 en alguna posición de la matriz, no se insertan ni cajas ni robots, en caso de que haya un 1 se inserta una caja, y en el caso de que el valor sea un 2, se inserta un robot. De esta manera, aseguramos que no van a compartir posiciones en el Grid y tanto las cajas como los robots van a aparecer en posiciones diferentes.

Posteriormente, los robots comienzan su búsqueda de cajas, estos se moverán en posiciones aleatorias, para así tratar de encontrar a una caja. Esta tarea se realiza a través del método de locateBox(), donde en caso de encontrarse con un agente tipo Caja en cualquiera de sus 4 sentidos (arriba, abajo, izquierda o derecha), capturará su información y lo igualará a su atributo de tipo Box. En caso de que no encuentre, este valor seguirá siendo igual a -1, para indicar que no logro identificar una caja, también, en cada iteración del modelo se pregunta si se ha encontrado una caja.

En caso de que ya tenga caja, el robot se desplazará hacia la posición del stack que aún no cuente con 5 cajas, esto se hace mediante la función TakeToStack() y posteriormente placeBox(), ambas funciones trabajan en conjunto para lograr que los robots suelten las cajas que poseen y sigan con su tarea de ordenar las k cajas ubicadas dentro del tablero. Una vez depositadas las cajas, se utiliza función takeFromStack() es llamada, la cual realiza un desplazamiento hacia el centro del tablero

para después comenzar a moverse de manera aleatoria de nuevo. Cada vez que se realiza un movimiento, anteriormente se realiza una verificación para ver si no hay algún robot en esa posición a través de la función de move().

Se realiza una comprobación de en qué estado se encuentra el agente tipo Robot. Si su atributo de hasBox es 1, significa que aún tiene una caja que debe de ir a depositar, entonces se utiliza el método de takeToStack(). Si es igual a 2, significa que acaba de realizar este depósito y debe desplazarse hacia el centro del tablero, de lo contrario, deberá de seguirse moviendo de forma aleatoria.