Tecnológico de Monterrey

Modelación de sistemas multiagentes con gráficas computacionales

TC2008B.523

Profesores
Sergio Ruiz Loza
David Christopher Balderas Silva

Parte 1. Sistemas multiagentes

Alumna
Karen Rocío Macías Ávila
A01657935

23 de noviembre de 2021

La situación problema nos menciona que somos propietarios de 5 robots nuevos y un almacén lleno de cajas, el objetivo es ordenar el lugar haciendo uso de estos para convertir el lugar en un negocio exitoso.

A continuación se muestra el diagrama de clases el cuál contiene todas las clases con sus respectivos métodos y atributos para dar solución a la problemática.

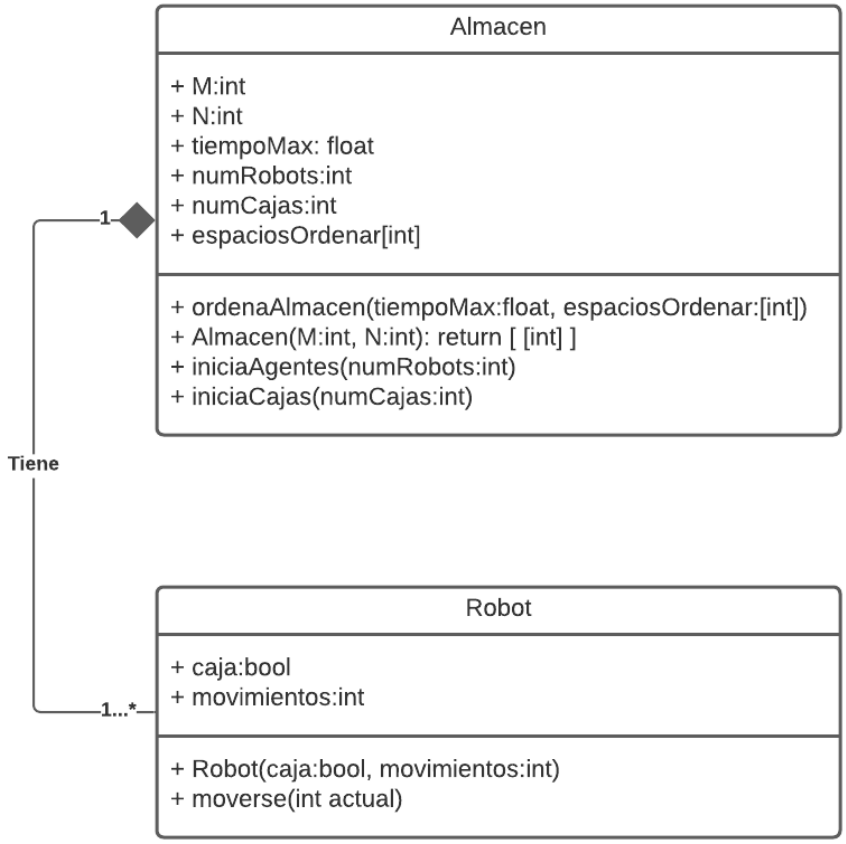

Figura 1. Diagrama de clases

El diagrama se constituye de dos clases, una que contiene los agentes que son los robots y otra que corresponde al ambiente donde interactúan y hacen su trabajo que es el almacén, ahora hablaremos sobre los atributos y métodos de cada una.

El robot tiene dos atributos, un elemento booleano que dice si tiene una caja y el número de movimientos que ha realizado, por parte de sus funciones tenemos su constructor que recibe ambos atributos y la de movimiento que revisa las 4 celdas alrededor de la posición actual en la que se encuentra el agente donde se recopilarán los datos de cada celda y se tomará una decisión antes de avanzar, ya sea agarrar una caja, ir por un camino que no tenga un robot, ordenar las cajas, etcétera y de cada elección hecha se aumenta un movimiento.

El almacén tiene 6 atributos, primero los que funcionan para crear el espacio del almacén que son **M** y **N** que es el tamaño del tablero que se va a generar, cabe mencionar que todos estos espacios contendrán vectores de elementos int que representarán el número de cajas que tiene cada lugar, esto se va a realizar en el constructor de la clase. Después tenemos el número de robots que se lo pasamos como parámetro a la función de **iniciaAgentes** que crea los robots y los posiciona en lugares aleatorios vacíos del tablero que se creó. Para crear cajas utilizamos la función de **iniciaCajas** que recibe el número de cajas y las pone en lugares aleatorios vacíos solo una por cada lugar. Finalmente tenemos el último método que se llama **ordenaAlmacen** que recibe el tiempo máximo y los espacios del tablero donde se tienen que acomodar las cajas, funciona mientras el tiempo máximo no se alcance y todavía tengas cajas en el tablero desordenadas.

A continuación se muestra el diagrama de protocolos el cuál contiene todas las interacciones entre clases para dar solución a la problemática.

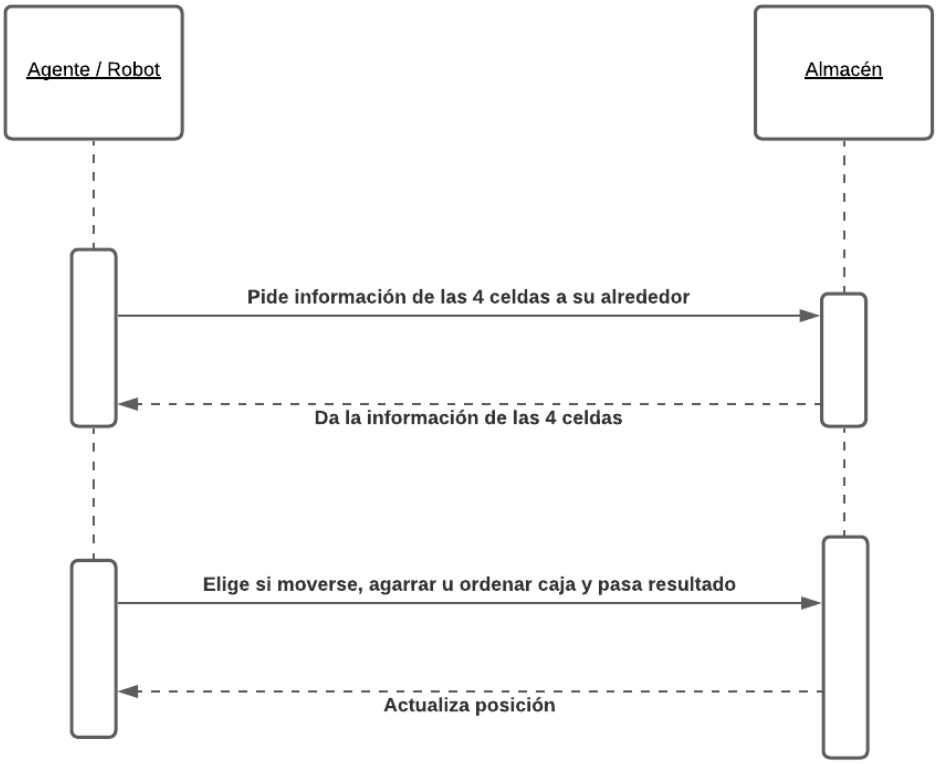

Figura 2. Diagrama de protocolos

En el diagrama se puede observar la interacción entre los agentes, es decir, los robots y el almacén que es el ambiente, primero que nada como el diagrama de protocolos se basa en todos los datos que se pueden transmitir entre ambas partes no se puede visualizar la parte de la construcción del tablero, la inicialización de agentes y de cajas, debido a que el ambiente contiene los datos para realizarlo, sin embargo, lo que sí

podemos encontrar es la comunicación de variables entre los agentes y el ambiente que son la información de las celdas de la posición actual de cada robot, se recopila la información y tomando en cuenta esas variables se toma una decisión que involucra moverse, finalmente se actualiza la posición del robot.

Para optimizar el código de esta problemática tengo dos opciones, la primera es tomando en cuenta solamente el código y es transportar el robot en vez de moverlo por casilla, en este caso como el almacén tiene de atributo las posiciones donde se ordenan las cajas el robot sólo debe agarrarla y aparecer en una casilla antes de la que contiene las cajas para posicionarla. La segunda opción es cuando se tiene un espacio real, lo que se podría hacer es dividir en cantidad de celdas iguales a cada robot, de esta manera no tiene que revisar si otro robot está en su camino, solo debe agarrar cajas, moverse en el espacio disponible y al final de cada espacio colocar las cajas apilando 5, en esta parte también sería importante que los agentes pudieran mandar datos entre sí, para que cuando uno haya finalizado su labor se pueda encargar de revisar de sus propias filas si algunas pueden tener más o menos cajas y conforme vayan terminando los demás acomodar todas las cajas de forma que unas no tengan menos que otras.

La solución que fue planteada en este documento solo contempla datos estáticos, sin embargo, debe tomarse en cuenta también la velocidad con la que el agente se mueve, los lugares que no puede atravesar fijos como son los racks que contienen las cajas y la ruta que puede seguir o cambiar dependiendo el espacio que le toca.

Referencias de los assets

- free warehouse shelves 3d model (turbosquid.com)
- free 3ds model wooden boxes pack (turbosquid.com)
- 3D wooden box model - TurboSquid 1304417
- Free pallet model | 1144189 | TurboSquid
- 3D wood boxes model - TurboSquid 1418297
- max warehouse pbr gaming (turbosquid.com)
- transpallet 3d 3ds (turbosquid.com)
- free 3ds mode robot spike (turbosquid.com)