Campus Monterrey

ACTIVIDAD INTEGRADORA

Materia
Modelación de sistemas multiagentes con gráficas computacionales (Gpo 4)

Profesores
Edgar Covantes Osuna

Integrantes

Nicolás Cárdenas Valdez A01114959

Agosto 26, 2021

Indice

Introduccion

Con 5 robots nuevos y un almacén lleno de cajas se nos asignó ordenarlo para convertir está situación en un negocio exitoso. Cada uno de nuestros robots está equipado con ruedas omnidireccionales y, por lo tanto, puede conducir en las cuatro direcciones. Con sus manipuladores, luego llevarlas a otra ubicación e incluso construir pilas de hasta cinco cajas. Todos los robots están equipados con la tecnología de sensores más nueva que les permite recibir datos de sensores de las cuatro celdas adyacentes. Con está tecnología, nuestra tarea fue enseñarle a los robots a ordenar dicho almacén.

Definición de la problemática

Con está información debemos crear una simulación que haga que nuestros robots ordenen el almacén, pero primero debemos definir el problema y analizar los componentes que serán necesarios para programar un sistema de este calibre. Primero que nada, sabemos que:

- Los robots pueden conducir en las cuatro direcciones adyacentes de manera horizontal y vertical.
- Los robots cargan las cajas que estén en su misma casilla.
- El objetivo es llevarlas a otra ubicación e incluso construir pilas de hasta cinco cajas
- Los robots pueden detectar (en estas posiciones adyacentes) si un campo está libre, es una pared, contiene una pila de cajas o está ocupado por otro robot y se puede mover o no de acuerdo a la información que tiene.
- Los robots saben si llevan una caja en ese momento.

En cuanto a la organización de los agentes, para que todas las cajas terminen en pilas ordenadas de cinco, optamos predefinir los lugares a donde se van a llevar las cajas, lo cual simularia un ordenamiento más limpio pero también asimilan más a una situación de vida real donde todas las cajas deben estar en un cierto lugar. Para está simulación en específico, escogimos que estos lugares de pilas asignadas estuvieran hacia la parte superior izquierda de nuestro “almacén”.

Al principio de la simulación se crea un grid de tamaño determinado por las variables de entrada y esparce al azar a un número de robots y un número de cajas alrededor de este grid. Cuando se esparcen cada robots y caja tienen una casilla para esos mismos, es decir, las cajas al principio no aparecen acumuladas y todos los agentes empiezan en posiciones aleatorias vacías.

Para la simulación: está se corre hasta que esté ordenado todo en las pilas o se haya llegado al tiempo máximo de ejecución determinado por el usuario.

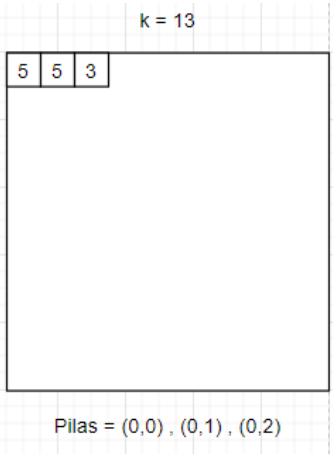
Deberemos recopilar la siguiente información durante nuestra simulación:

- El tiempo necesario hasta que todas las cajas estén en pilas de máximo 5 cajas.
- El número de movimientos realizados por todos los robots.

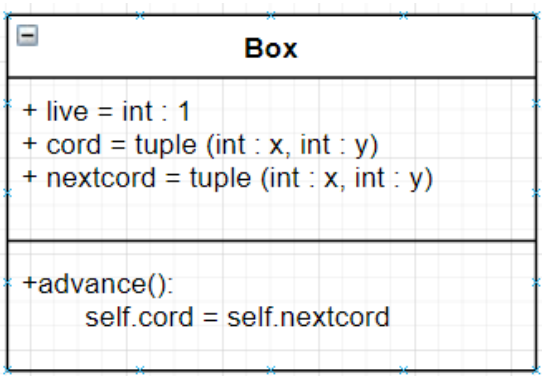
Si el número de cajas es más grande que MxN-NumAgentes, o si hay más robots que el tamaño del grid (que en nuestro caso solo seria 5), entonces es una simulación invalida ya que no queremos aparecer a los robots en cajas

Modelado de Agentes y Diagramas

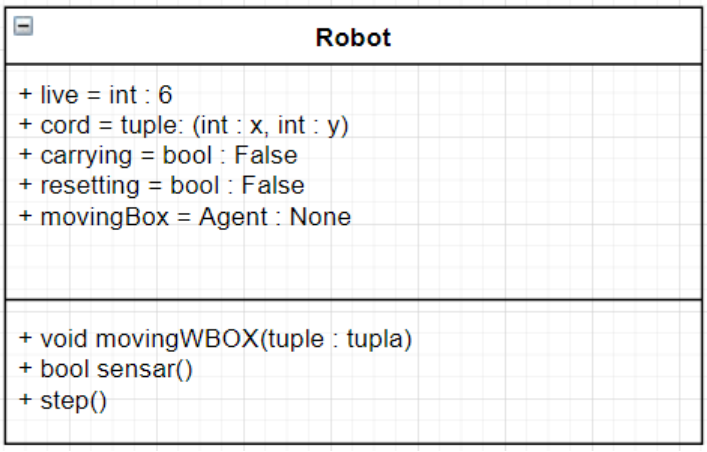
Primero que nada debemos saber cómo funciona nuestro modelos de pilas, para la imagen inferior suponemos que tenemos trece cajas en nuestra simulación. Con un máximo de cinco cajas por pilas necesitamos dos llenas y una con tres. Por lo cual tendrán las siguientes que se muestran en la imagen. A esto le definimos como pilas y se ubicaran siempre a la superior izquierda.



En cuanto a los agentes tenemos lo siguiente:

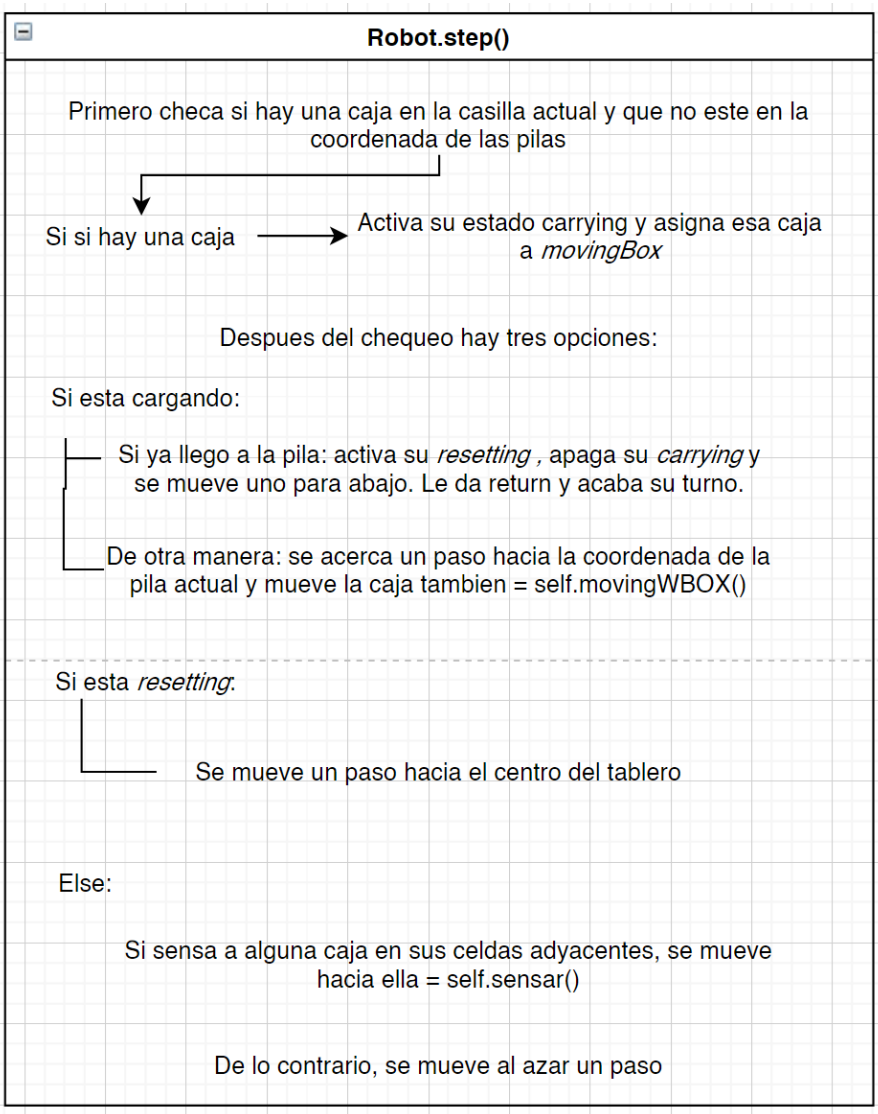


Para el agente de caja, para marcarlo en el grid tenemos el atributo live que usamos para el agente caja y el robot. Se llaman igual para cuando recorremos el grid checar el mismo atributo para todos. Tenemos también el atributo de cord para las coordenadas de la caja en el grid y la *nextcord* que lo utilizamos para moverla en advance, el agente caja predefine la nextcord para simular moverla. El agente caja solo cuenta con las funciones que necesita para la simulación ya que para modelar algo parecido a la vida real, la caja no está realizando nada ya que no tiene sensores, ni se puede mover, entre otras cosas.



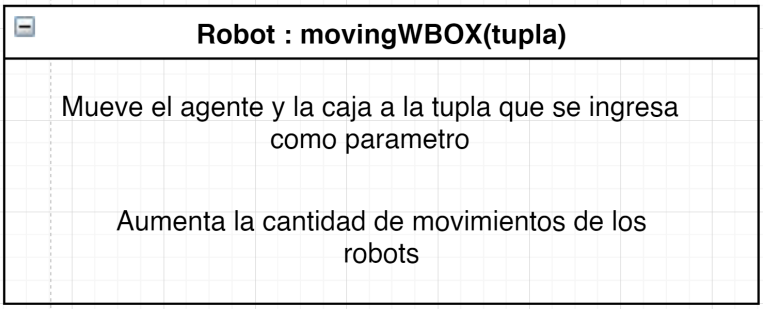
La clase de robot es la principal en este programa, es la que se encarga de mover los agentes de tipo caja a las coordenadas de las pilas que se asignan al principio del programa. Como atributos de está clase tenemos el *live*, que como ya mencionamos es el que utilizamos para colocarlos en el grid que se usa para la animación, este tiene un valor de seis ya que siempre tiene que ser el más alto para que aparezca como blanco, ya que el blanco es el valor más alto en nuestro *colormap* de *matplotlib*. Al igual que la caja tenemos nuestro atributo *cord* para las coordenadas pero también tenemos el atributo de *carrying* y el atributo de *resetting*. Estos atributos nos indican si el robot está actualmente cargando una caja hacia la pila o si está “restaurando”. La manera en que diseñamos la simulación, quisimos que después de dejar una caja, los robots se

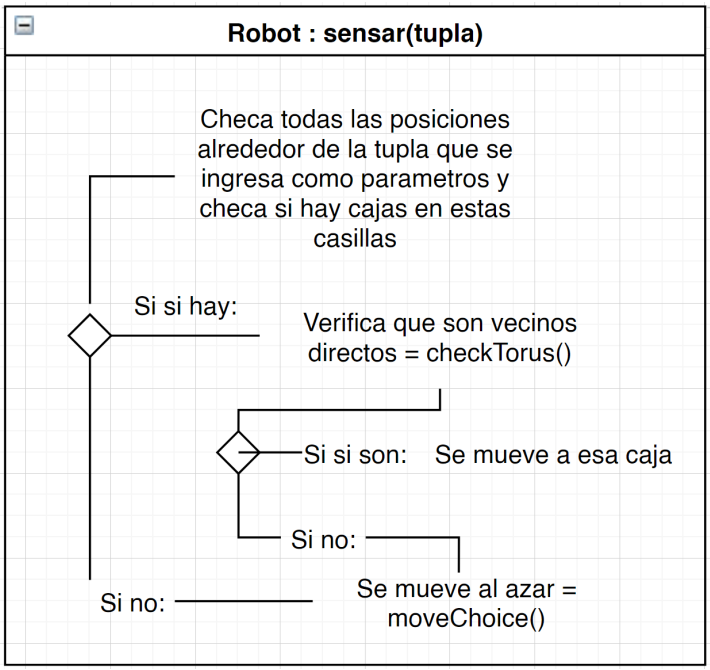
movieran hacia el centro y después de llegar al centro que se movieran a partir de ahí, aunque toma más movimientos regresar al centro, encontramos que nos daba un mejor resultado debido a que si salian a partir de la esquina, muchas veces se quedaban atorados ahí. El hecho de que salen del centro apoya a que tengan que moverse menos para encontrar otra caja que mover y mejora la oportunidad para acomodar todo. Solo tiene uno de estos atributos (*carrying, resetting*) activo a la vez, son más que nada para definir qué comportamiento debe tomar en su función de *step()*. Por último, tenemos *movingBox,* el cual sirve para que el robot sepa qué caja está cargando. En cuanto a funciones tenemos tres para está clase pero empezaremos describiendo la de *step()* que vendría siendo la principal que se encargaría de hacer la simulación. La describiremos mediante el diagrama siguiente, ya que tiene varias posibilidades.



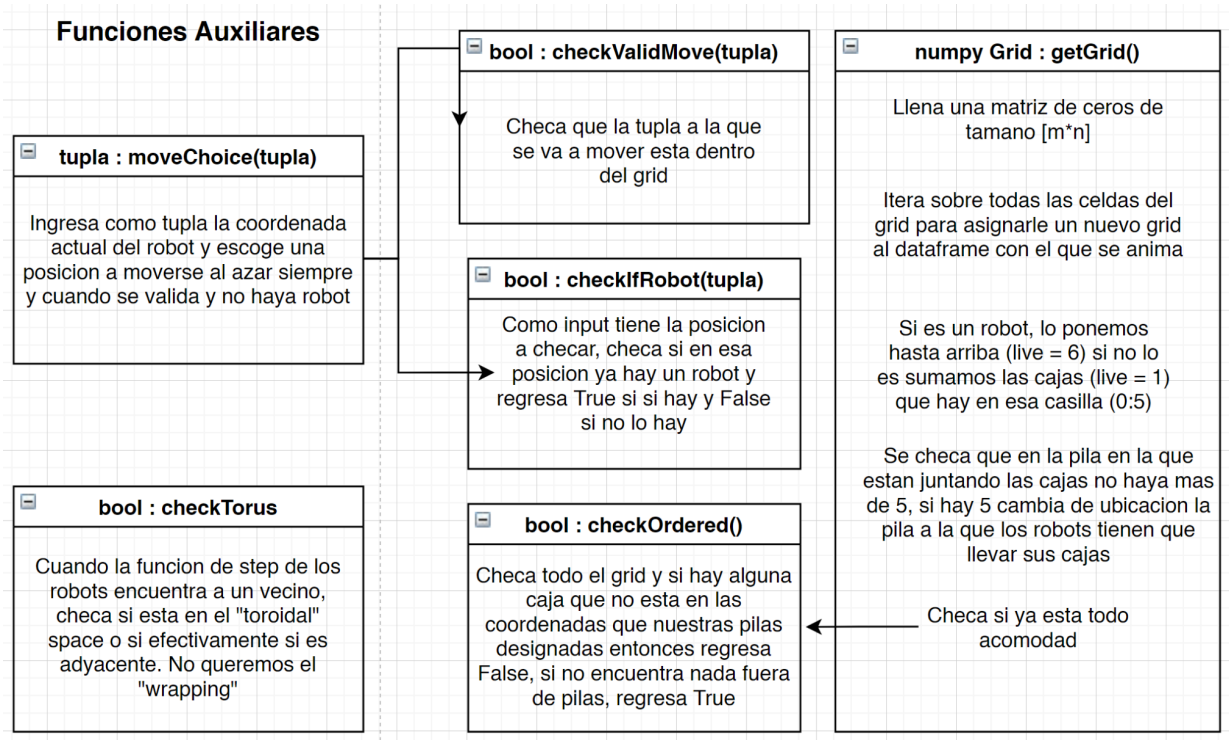
Como se puede observar, está función es la que describe el comportamiento de casi todo el modelo ya que el robot es el que se mueve y mueve la caja, por lo tanto esto fue lo más importante de programar precisamente para que los robots supieran cómo moverse a través del

almacén y cumplieran con los comportamientos que se describieron en la introducción. Cabe recalcar que cada vez que se hace un movimiento en cualquier situación, se le agrega un movimiento al registro global de movimientos realizados por los robots = *moves*. Está funcion de *step()* hace uso de las otras funciones que se mencionaron anteriormente, las cuales son las siguientes:

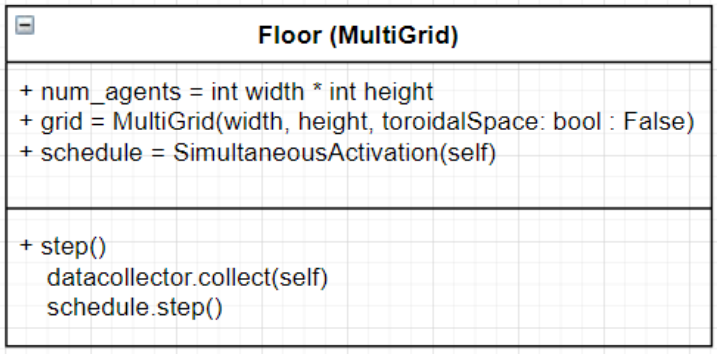




Además de estas funciones dentro de la clase de agente tenemos otras funciones auxiliares a nivel global que mas que nada nos ayudan a determinar las siguientes cosas:



Cabe destacar que la función de *getGrid()* es función auxiliar de la clase Floor de tipo MultiGrid para poder llenar una data frame para después simularlo con la ayuda de *matplotlib* pero más sobre esto en la sección de "Funcionamiento del programa".



Por último tenemos nuestra clase Floor de tipo Multigrid la cual usamos para mover a los agentes sobre ella. Como atributos, tenemos los que se deben definir para poder crear el grid: *num_agents* que nos dice el tamaño del grid a crear, qué tipo de grid queremos crear (en nuestro caso *MultiGrid*) y por último el schedule que es parte de lo necesario para simular con la ayuda del paquete *MESA* de python. Como paso tenemos el *step()* el cual da un "paso" en la simulación mediante la recolección de la información y la ayuda de la función que mencionamos anteriormente de *getGrid()*.

Funcionamiento del programa

Requerimientos

Python 3+
Paquetes de python

- Sys
- Mesa
- Matplotlib
- Numpy
- Pandas
- Random
- Math
- Time
- Datetime

Parámetros de entrada para la simulación

- Alto : m
- Ancho : n
- Número de cajas : k
- Número de robots : numAgents
- Tiempo máximo de ejecución (segundos) : tiempoMax

El programa está escrito en Python 3. La simulación se realiza con la ayuda del paquete de MESA que sirve para poder mover los agentes sobre un grid y darles a estos agentes diferentes características. La visualización se realiza por medio de matplotlib con ayuda de numpy y pandas.

Limitaciones

Con el programa realizado analizamos que pudiéramos mejorar o qué aspectos podríamos optimizar para disminuir el tiempo dedicado o la cantidad de movimientos realizados:

- Sería más rápido que tuvieran las pilas en el centro pero se optó por realizarlo de está manera porque asimila más a una en vida real, estar cerca de la puerta o la salida o algo parecido.
- Los robots pudieran estar activamente buscando en lo que se están restaurando
- Que los robots pudieran moverse en diagonal
- Permitir a los robots sensar en un radio más grande

Anexos

Diagramas actividad integradora :
https://drive.google.com/file/d/1O8u5J9ryiYyWZqIjqKy28Hfps_iNKDdd/view?usp=sharing

Referencias

Cómo Organizar un Inventario: 5 Consejos Infalibles. (2019). Retrieved 27 August 2021, from https://www.tradelog.com.ar/blog/como-organizar-inventario/

What robots do (and don't do) at Amazon fulfilment centres. (2019). Retrieved 27 August 2021, from https://www.aboutamazon.co.uk/amazon-fulfilment/what-robots-do-and-dont-do-at-amazon-fulfilment-centres#:~:text=In%2026%20of%20them%2C%20robots,%2C%20transport%2C%20and%20stow%20packages.&text=Robots%20pick%20up%20heavy%20items,now%20come%20directly%20to%20employees.