Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

Modelación de sistemas multiagentes con gráficas computacionales

TC2008B.1

Actividad integradora

Alumno:

Daniel Esteban Maldonado Espitia A01657967

Fecha:

30 de noviembre del 2021

Diagramas de clases:

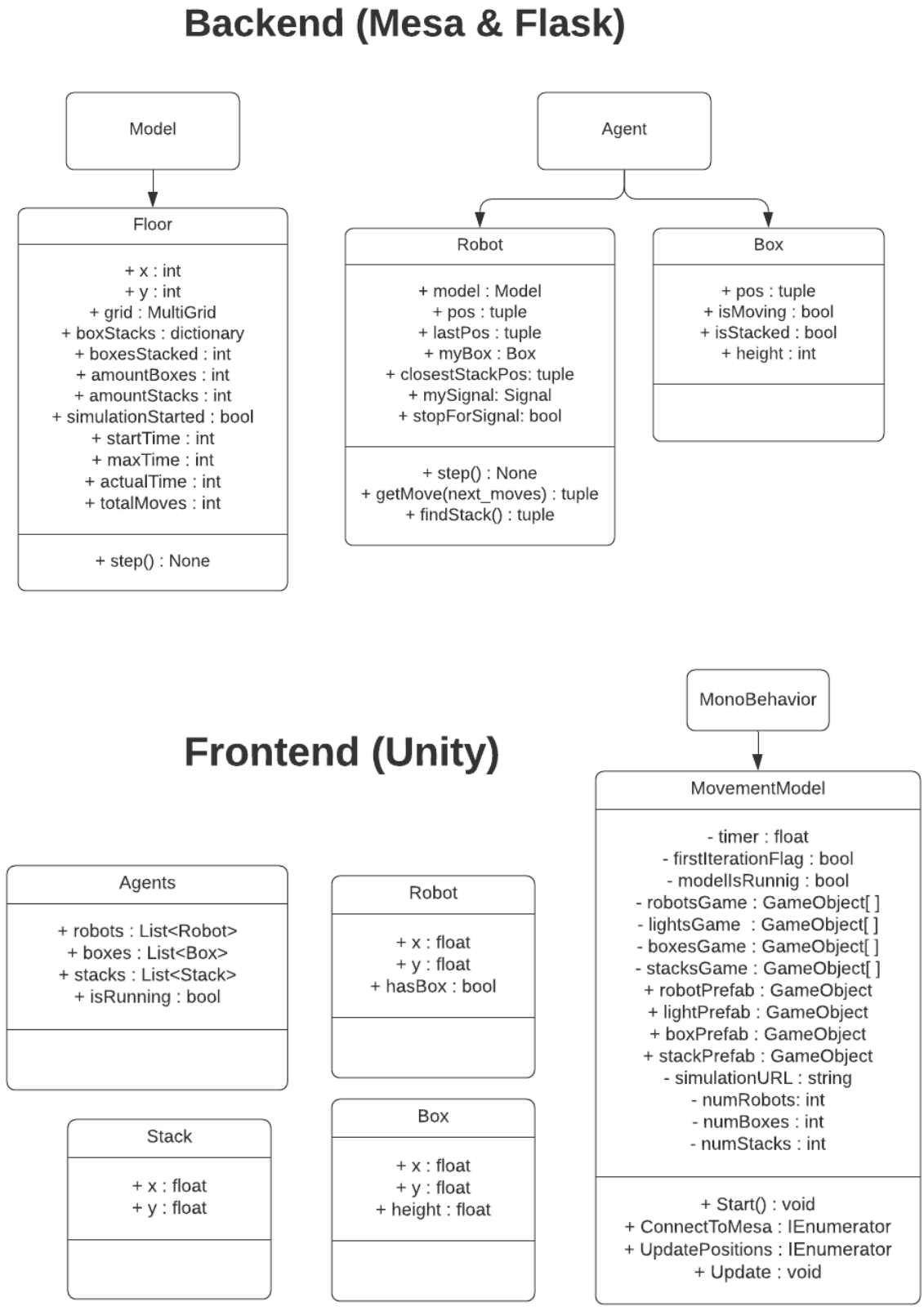
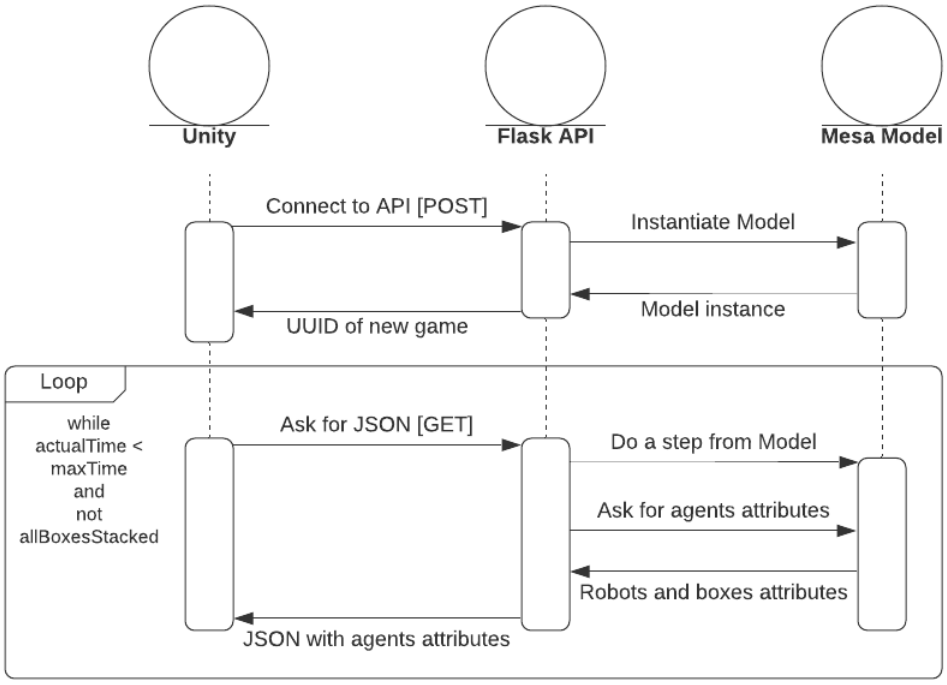


Diagrama de protocolos (secuencias):



Estrategia de solución del problema:

Para la primera parte de la actividad en la que se solicitaba crear el modelo para dar solución a la problemática en cuestión, se decidió tomar como agentes dentro de la simulación a los robots apiladores, ya que estos van a contar con cierta autonomía que les permite moverse libremente en el almacén para encontrar las cajas y llevarlas al estante más cercano. Así mismo, se usaron a las cajas como agentes dentro del modelo, ya que estas tienen la capacidad de moverse (cuando un robot las toma) y apilarse encima de otras al estar en un estante.

Ahora bien, para permitir que los robots se movieran libremente en el área asignada en busca de las cajas, se decidió implementar un método llamado getMove que analiza las 4 posiciones a las que se puede mover el robot y selecciona la más adecuada dependiendo de su estado. Este es un punto importante, ya que si el robot se encuentra llevando una caja, este va a calcular la pila más cercana a él y se moverá a ella. Sin embargo, si no tiene caja, descartará la posibilidad de moverse a la posición previa, la cual evidentemente no debe tener nada, y tratará de encontrar una caja seleccionando aleatoriamente una posición vecina. En caso de que se encuentren al lado de un robot o stack, descartan automáticamente la posibilidad de moverse a dicha posición y seleccionan otra opción.

En cuanto a las cajas, estas se mantienen en una posición fija si no han sido encontradas, se mueven junto con el robot que las haya encontrado y se ponen encima de las cajas que estén antes de ella en un estante o pila. Para lograr que en las cajas se pongan encima del robot al ser movidas y se apilen en la altura adecuada cuando lleguen a su pila más cercana, estas

cuentan con un atributo llamado ‘height’ que, como su nombre lo dice, define la altura a la que deben colocarse cuando son transportadas por un robot o cuando se encuentran en una pila. Cabe destacar que la altura que deben ocupar en la stack se define por la cantidad de objetos que tiene dicha pila; esto quiere decir que, previo a que el robot coloque en la pila esa caja que llevaba, se consulta la cantidad de cajas que tiene dicha pila y en base a ello se calcula la altura a la que debe ir dicha caja.

Cabe destacar que los robots siempre buscarán moverse a la pila más cercana que tenga menos de 5 cajas. En caso de que lleguen a una pila que justo en el step anterior se llenó, el robot debe calcular la otra pila más cercana que tiene menos de 5 cajas, y empezar a moverse a ella.

Por último, las pilas o estantes no se manejan como agentes dentro del modelo, sino como un diccionario que pertenece a la clase de Floor, el cual tiene en cada clave una tupla con la posición de dicha stack y en su valor, la cantidad de cajas que tiene. Decidí manejarlo de esta forma ya que considero que son más un elemento ilustrativo que algo que deba interactuar con el modelo, ya que estas stacks no se mueven al correr la simulación.

Análisis de si existe una estrategia que podría disminuir el tiempo dedicado, así como la cantidad de movimientos realizados

Analizando el movimiento de mis robots al ejecutar la simulación, me percaté de que estos se suelen permanecer en el mismo sector de habitación ya que es muy poco probable que seleccionen de forma aleatoria un movimiento que los haga ir hacia la derecha en repetidas ocasiones, por ejemplo, haciendo que se alejen del sector en el que se encontraban antes y en el que no han encontrado nada.

Para mejorar esta situación, se me ocurren 3 alternativas de solución que pueden resultar útiles para disminuir el tiempo de apilado de las cajas y los movimientos que realizan. Las 3 opciones se presentan a continuación.

1. Definirle a cada robot una sección de la habitación, para que así pase de forma ordenada por todas las casillas de dicha sección y no falten por recorrer ninguna celda de la habitación. Esto también evitaría que los robots se concentren en un solo sector de la habitación y no recorran otras partes que no han sido visitadas.
2. Mejorar el movimiento aleatorio de los robots haciendo que estos, después de recorrer un sector en el que no encuentran cajas, se muevan repetidamente a otra dirección específica hasta llegar a otro sector y allí repetir el mismo proceso de búsqueda aleatoria. Esto evitará que los robots se estanquen en un solo sector.
3. Crear un set que guarde las casillas que se han visitado para así hacer que los robots siempre intenten moverse a las partes que menos se han recorrido. A pesar de que esta estrategia implicaría más espacio en la memoria, esto no sería tan significativo si el tamaño de la habitación no es tan grande.