Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Puebla

Modelación de Sistemas Multiagentes con Gráficas Computacionales (Gpo 1)

TC2008B.1

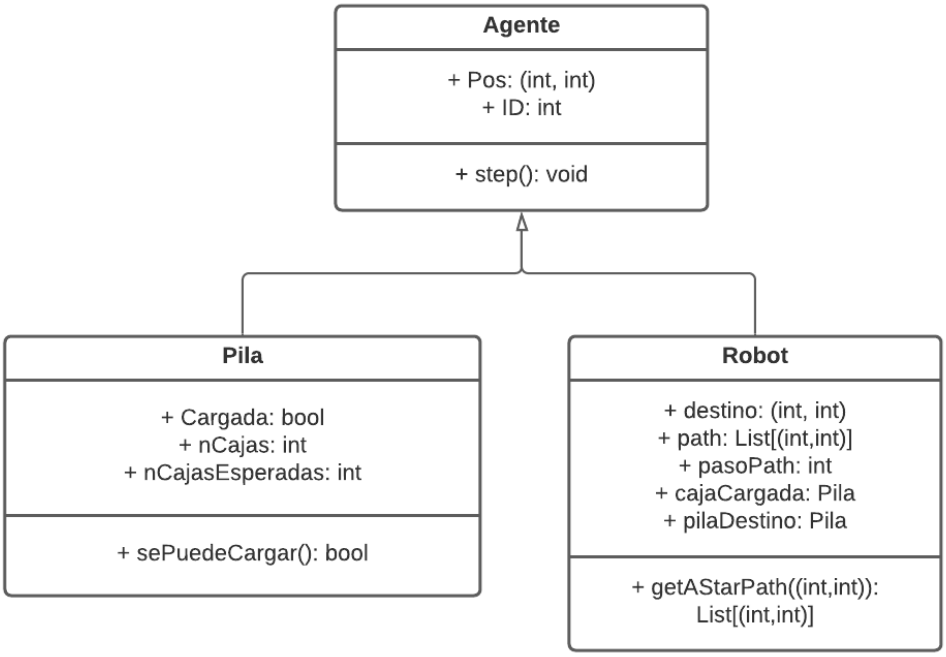
Actividad Integradora

Ituriel Mejia Garita A01730875

Fecha:
29 de Noviembre del 2021

Diagramas de clases

Para la realización de diagramas de clases se consideraron tanto agentes como modelo. Por parte de los agentes, tenemos el de Robot, que contendrá la lógica que le permitirá a los robots hacer sus respectivas labores; y el de Pila, que consta de una pila de cajas que puede ir desde una sola, hasta las 5 que pueden ser apiladas, y contiene los datos necesarios para que los robots puedan interactuar con ellas.



Por parte del modelo, tenemos al almacén, que contendrá toda la información acerca del espacio y tiempo de la simulación, además de que se encargará de inicializar a los agentes en la simulación.

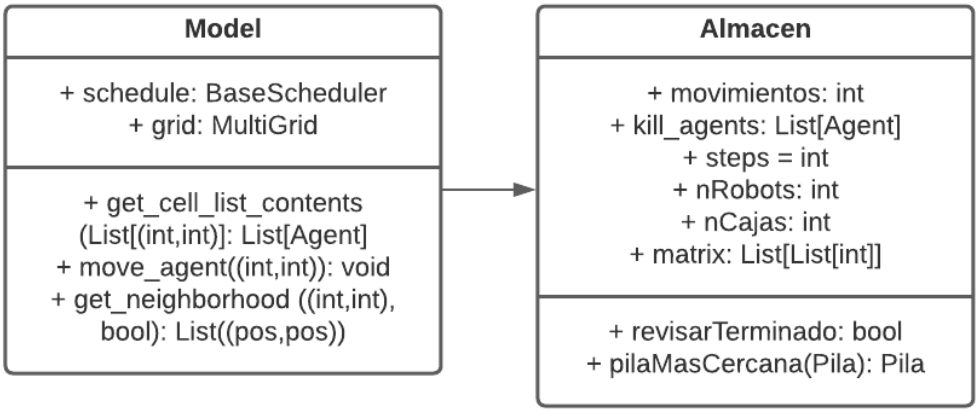
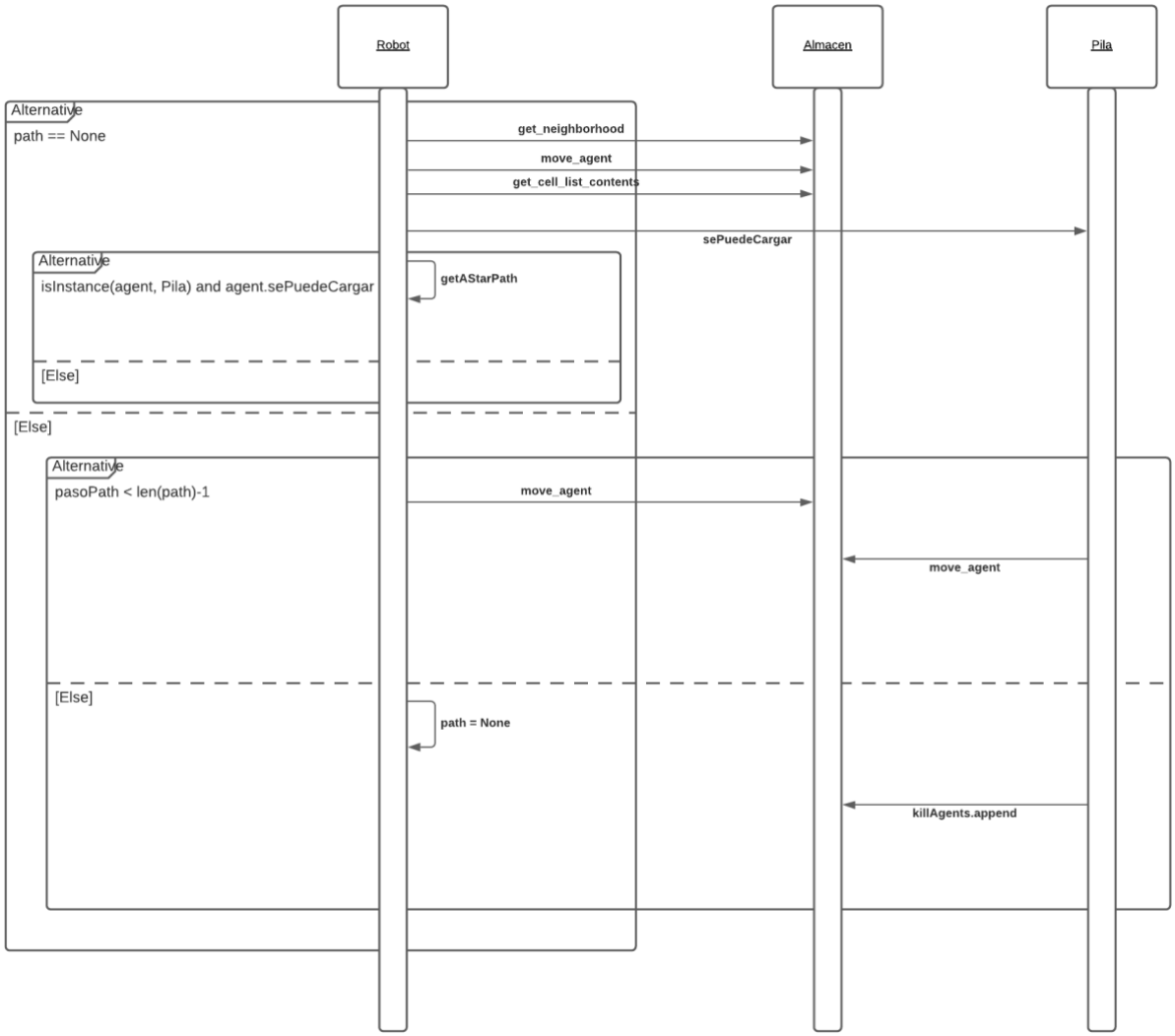


Diagrama de protocolo

En el diagrama de protocolo se describe la interacción que tendrán los agentes en la simulación. Se describe específicamente lo que realizará el robot en cada una de las iteraciones de la simulación en el modelo.



Lo primero que realiza es revisar si su variable *path* es nula, lo que significaría que aún no está cargando una caja porque no tiene un destino en específico. Por lo tanto, utilizará los métodos de Almacén para moverse a una de sus celdas vecinas al azar. Una vez realizado el movimiento, se revisa si se ha desplazado a una celda con una pila que tenga las características necesarias para ser cargada (que solo sea una caja, por ejemplo). Y de ser así, se establecen los datos para que el robot comience a cargar esa caja para llevarla a un destino en específico para apilar, por lo que se obtiene el *path* con el método *getAStarPath.*

Si el robot se encuentra siguiendo un recorrido, es decir, está llevando una caja a una pila destino, entonces simplemente se va moviendo de celdas conforme lo indica el *path* hasta llegar a una celda antes del destino, es decir, cuando llegue junto a la pila donde se pondrá la caja. Cuando esto suceda, el *path* se vuelve a convertir en nulo y la pila que estaba cargando se elimina porque a la pila destino se le aumentará una caja en su número de cajas, sustituyendo así a la caja eliminada.

Estrategia cooperativa para la solución del problema

Para la resolución completa del problema intervinieron varias partes. La primera, es el modelo en sí que dictamina el comportamiento y la interacción que tendrán los agentes. Esto ha sido descrito a detalle en las secciones anteriores de este documento, y fue desarrollado en Python con la librería de Mesa.

Otra parte importante fue el modelado en Unity, que consiste en la recopilación de modelados 3D para representar a los agentes y la realización del escenario.

Una vez realizado esto, para conectar ambas partes, se desarrolló una API en Flask, que proporcionaría información del modelo de Mesa, es decir, las coordenadas que tendría cada agente en cada una de las iteraciones del modelo. El proyecto en Unity haría llamadas a esta API para poder posicionar a los agentes donde corresponden en cada iteración.