



Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Puebla

**Modelación de sistemas multiagentes y gráficas
computacionales**

TC2008B.1

Documento PDF de actividad integradora

Alumno:

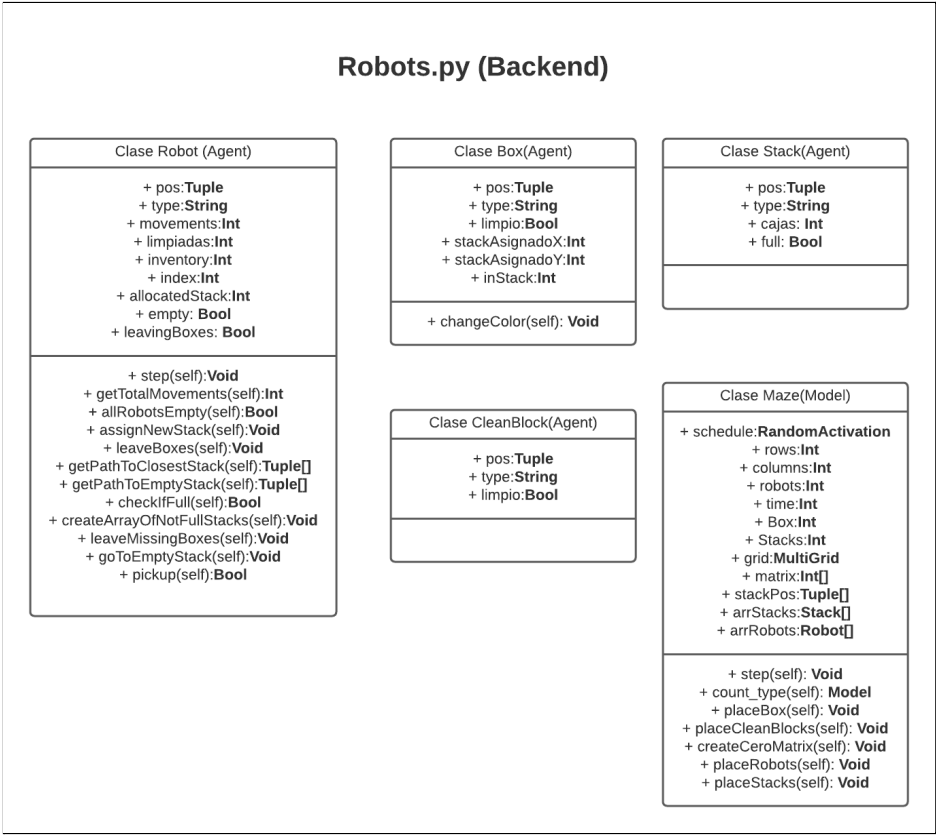
José Antonio Bobadilla García

A01734433

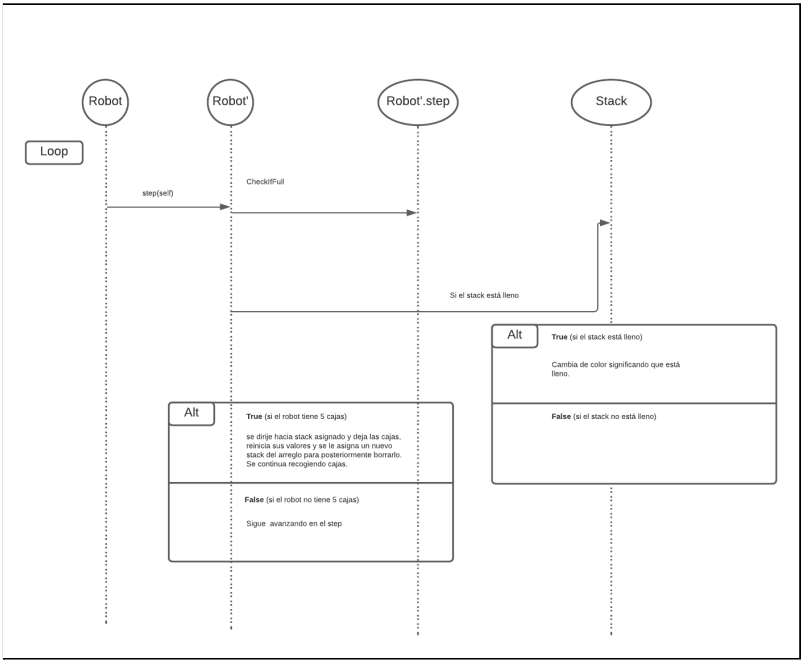
Fecha:

29 de Noviembre del 2021

I. Diagramas de clases



II. Diagramas de protocolos



III. Estrategia cooperativa

Antes de iniciar el programa se definieron ciertas reglas que el modelo debe de llevar a cabo para que la tarea de recoger cajas y apilarlas de 5 en 5 fueran exitosas. Primeramente se construyó el modelo a partir de parámetros como columnas, filas, robots, tiempo y cajas. Posteriormente se crearon n cantidad de stacks en base a las cajas definidas anteriormente, para esto se dividió la cantidad de cajas entre 5 para sacar la cantidad de stacks para cada conjunto de cajas. Se crearon también 4 arreglos, siendo el primero una matriz de ceros para identificar los bloques normales de las cajas. El segundo es un arreglo el cual guardará la información de coordenadas de cada Stack creado, guardando estos como una tupla, dando como resultado un arreglo de tuplas de enteros representados por coordenadas x,y. También se crearon 2 arreglos que guardarán objetos de stacks y robots para su futuro uso.

Los agentes usados en el programa fueron 4:

- Robot
- Box
- CleanBlock
- Stack

Al momento de colocar los robots se les asigna un stack inicial aleatorio del arreglo general de stacks, el cual será borrado posteriormente para mantener solo los arreglos que están vacíos y evitar que algún robot se dirija a un stack lleno. Cuando el robot junta 5 cajas, se dirige a su stack establecido, deja las cajas y se selecciona otro stack para repetir las mismas acciones hasta que no haya cajas vacías.

Se utilizó el algoritmo de AstarFinder para encontrar la ruta más cercana hacia el stack correspondiente de cada robot. Cabe mencionar que en cada paso se verifica la variable de tiempo para terminar el programa en cuanto se llega al límite. Si por alguna razón el tablero se queda sin cajas pero algunos robots aún tienen en su inventario, estos se redirigen hacia los stacks que no están llenos y dejan sus cajas, si sobran, el agente se queda con ellas y se dirige al siguiente stack, si no, el stack se setea como lleno y el robot deja de moverse, dejando en movimiento solo con los robots que aún tienen cajas en su inventario. igualmente si los stacks están llenos y todas las cajas se recogieron y ningún robot tiene cajas, se muestran los movimientos totales.

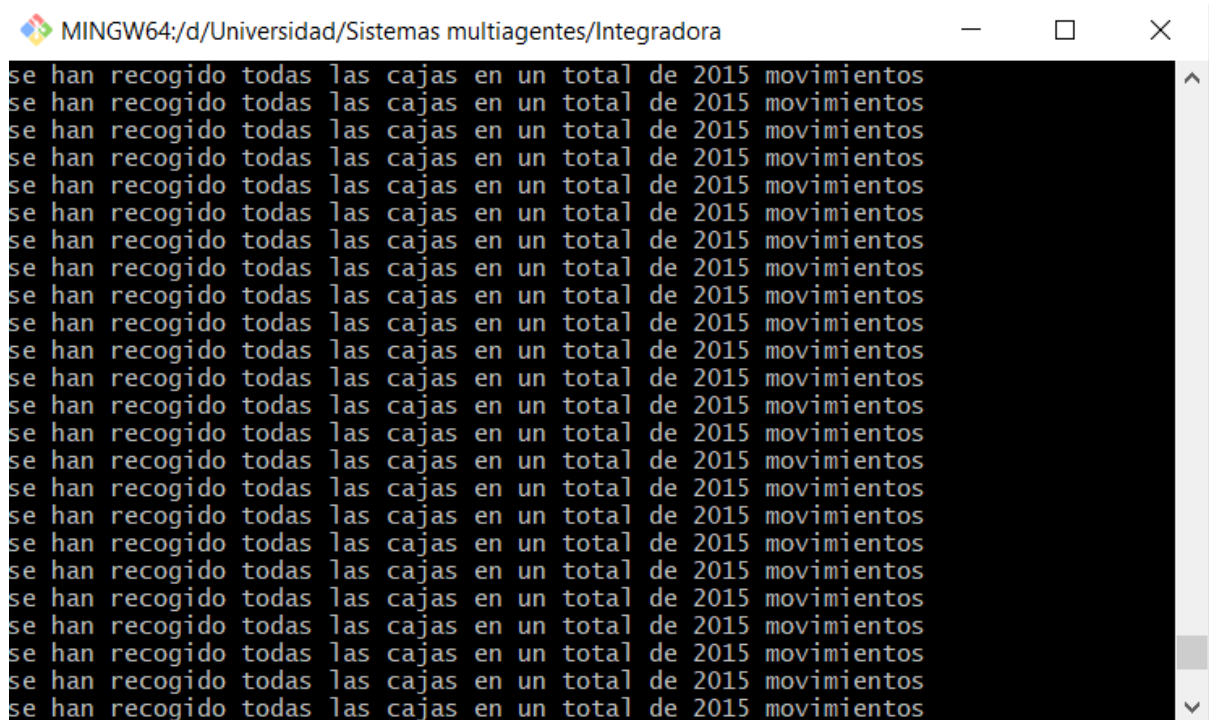
Una vez que se programó la lógica en Python se mandan estos datos mediante una API desarrollada en Flask a Unity, para que este mueva los robots, posicione las cajas y los stacks.

A continuación se muestra un video de la ejecución del programa con los siguientes parámetros:

<https://youtu.be/oe-SMM3upIk>

- Robots: 5
- Cajas: 60
- Stacks: 12
- Casillas: 100

El número total de movimientos que realizaron los robots fue de:



A screenshot of a terminal window titled "MINGW64:/d/Universidad/Sistemas multiagentes/Integradora". The terminal displays a single line of text, "se han recogido todas las cajas en un total de 2015 movimientos", repeated approximately 20 times, one per line. The text is in a light green color on a black background. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

Una forma de agilizar el programa sería reduciendo los movimientos que realiza cada agente, en este caso los robots. Una de las formas sería que en cada step, cada uno de los robots utilice el algoritmo aStarFinder para encontrar la caja más cercana a el y reducir drásticamente la cantidad de movimientos realizados. Si deseamos utilizar esta implementación también tendríamos que solucionar las colisiones de que 2 o más robots vayan por la misma caja, entonces una solución es añadir a cada caja un atributo de “localizada,” y si ese atributo está activo, que los robots busquen otra, es decir, que ya está siendo localizada por otro robot.

En esta actividad integradora se utilizaron conocimientos de sistemas multiagentes, los cuales se desarrollaron en Python, para posteriormente mandar estos datos a Unity, modelar los agentes, robots y cajas para poder simular estas acciones en un escenario tridimensional.

IV. Implicaciones éticas

Al momento de desarrollar un sistema como este, se deben tomar en cuenta situaciones como por ejemplo que funcione sin ningún tipo de error tomando en cuenta más variables de las que se tomaron en cuenta para esta actividad integradora, una de estas podría ser que este programa o desarrollo de software no sea usado para fines para los cuales no se desarrolló y aprovecharse de esto.