



Instituto Tecnológico y de Estudios Superiores de Monterrey  
Campus Puebla

**Modelación de sistemas multiagentes y gráficas  
computacionales**

TC2008B.1

**Actividad Integradora**

Alumnos:

David Zárate López

A01329785

Fecha:

29 de Noviembre de 2021

En esta actividad integradora se presenta el caso de ser el propietario de 5 robots nuevos y un almacén lleno de cajas. El dueño anterior del almacén lo dejó en completo desorden, por lo que depende los robots organizar las cajas en stacks. Para ello, el almacén se encuentra con dimensiones de  $M \times N$  filas y columnas respectivamente y los robots pueden desplazarse en 4 celdas adyacentes, distinguiendo entre si está libre o no, dependiendo de los agentes que la ocupen. Inicialmente, se cuentan con  $K$  cajas, a nivel de piso y 5 robots, ambos tipos de agentes en posiciones aleatorias.

Para poder realizar dichas acciones, se utilizaron dos tecnologías principales, la primera, siendo el framework de Mesa, que contiene todo el comportamiento del modelo y los diferentes agentes de este, programado en Python y la segunda Unity, en el cual se utilizó el lenguaje de C# y nos permite visualizar en 3D el modelo desarrollado.

Además, se realizaron diagramas de clases y de secuencia del sistema para poder explicar más a detalle cómo es que funciona el modelo utilizado y posteriormente se explica la estrategia colaborativa utilizada. Los diagramas fueron realizados en LucidChart y son mostrados a continuación con su respectivo link en caso de que no se alcance a observar completamente el diagrama.

## Diagrama de Clases:

[https://lucid.app/lucidchart/b1174d98-984a-4f51-a8fb-709f72431f26/edit?viewport\\_loc=-330%2C-55%2C3945%2C2039%2C0\\_0&invitationId=inv\\_65b9c752-7e90-4c8d-9279-0d745b3a3a52](https://lucid.app/lucidchart/b1174d98-984a-4f51-a8fb-709f72431f26/edit?viewport_loc=-330%2C-55%2C3945%2C2039%2C0_0&invitationId=inv_65b9c752-7e90-4c8d-9279-0d745b3a3a52)

Robot (Agent)	
+type	String
+pos	Tuple(int,int)
+movements	Int
+limpiadas	Int
+cargando	Bool
+travelingCoordinate	Bool
+path	Int[ ]
+indexPath	Int
+endCoordinate	Tuple(int,int)
+stackIndex	Int
<hr/>	
+getAgent	Agent
+step	Void
+checkIfClean	Bool
+getPath	Int[ ][ ]

Block (Agent)	
+pos	Tuple (int,int)
+type	String
+limpio	Bool
+boxes	Int
+stack	Bool
<hr/>	
-Void changeToClean	
-Void changeToStack	

Warehouse (Model)	
+rows	int
+columns	int
+robots	int
+time	int
+dirtyBlocks	int
+grid	Multigrid
+matrix	int[ ][ ]
+stackPositions	int [ ][ ]
+stackNumber	int
+cleanBlocksPositions	int[ ][ ]
<hr/>	
-Void createMatrix()	
-Void placeRobots()	
-Void placeDirtyBlocks()	
-Void placeCleanBlocks()	
-Void step	

[https://lucid.app/lucidchart/9070a9bb-291b-4f7b-888f-e49711625fea/edit?viewport\\_loc=805%2C-1086%2C5572%2C2881%2C0\\_0&invitationId=inv\\_4de1c780-e803-400e-af99-4022cb8097f2](https://lucid.app/lucidchart/9070a9bb-291b-4f7b-888f-e49711625fea/edit?viewport_loc=805%2C-1086%2C5572%2C2881%2C0_0&invitationId=inv_4de1c780-e803-400e-af99-4022cb8097f2)



Para poder evitar la colisión de los robots con otros agentes, solamente se le permite moverse si la casilla próxima se encuentra vacía y al momento de encontrar una caja, la “absorbe” y con el algoritmo de AStar, de la librería de PathFinding de Python, encuentra el camino óptimo entre su posición y el stack. De igual manera, a cada robot se le es asignado un stack diferente, el robot identifica si existe un stack donde puede depositar la caja, en caso de que no haya, se genera uno de los espacios libres del tablero y en caso contrario, se dirige directamente al stack que le corresponde.

Una vez obtenido el modelo multiagente en Mesa con Python, se utilizó un backend realizado con Flask, para mandar los datos a Unity y realizar una simulación exitosa. Para ello se mandó un objeto con toda la

información de los agentes involucrados, con sus coordenadas en X y Y, tipo de agente, cuantas cajas contiene (para los stacks) y si se encuentra cargando una caja (para los robots). Una vez ejecutando el backend, se utilizó un script en C# que permite realizar dicha conexión y que al mismo tiempo genera todos los objetos en 3D que se necesitarán en la simulación, con sus respectivas posiciones.

Una vez que todas las casillas sean limpiadas o en caso de que el tiempo de ejecución se termine, la información de los movimientos realizados por cada robot será desplegada. Al haber analizado el tiempo que se necesita para limpiar todo el almacén, ordenar las cajas en stacks y el número de movimientos de los robots es posible identificar que existe una gran variación entre diferentes pruebas, esto tiene sentido puesto que los robots se mueven de manera aleatoria hasta encontrar una caja y ahí se mueve de manera óptima. Asimismo, entre más grande sea el tablero, más tiempo tardarán los robots en encontrar todas las cajas, debido a este comportamiento aleatorio.

Por ejemplo, en las siguientes capturas se muestran los resultados de simulaciones de un tablero de 7x7, 10 casillas con cajas y 5 robots. Se dio un tiempo de ejecución de 400 movimientos, lo cual tuvo como resultado que se limpiaran todas las casillas sucias en alrededor de 39-45 movimientos por parte de los robots, teniendo un total de 190-230 movimientos entre todos los robots. Estas pequeñas variaciones dependen de la suerte de cada robot al encontrar una caja, ya que sus movimientos al no cargar alguna, son de manera aleatoria, lo cual significa que no es posible mantener un resultado constante.

```
{ "type": "get_step", "step": 41 }
{ "type": "get_step", "step": 42 }
{ "type": "get_step", "step": 43 }
{ "type": "get_step", "step": 44 }
{ "type": "get_step", "step": 45 }
{ "type": "get_step", "step": 46 }
{ "type": "get_step", "step": 47 }
Se han limpiado todas las casillas sucias en un tiempo de ejecucion de 40 movimientos, teniendo un total de 200
```

```
{ "type": "get_step", "step": 45 }
{ "type": "get_step", "step": 46 }
{ "type": "get_step", "step": 47 }
{ "type": "get_step", "step": 48 }
{ "type": "get_step", "step": 49 }
Se han limpiado todas las casillas sucias en un tiempo de ejecucion de 45 movimientos, teniendo un total de 225
```

```
{ "type": "get_step", "step": 41 }
{ "type": "get_step", "step": 42 }
{ "type": "get_step", "step": 43 }
Se han limpiado todas las casillas sucias en un tiempo de ejecucion de 39 movimientos, teniendo un total de 195
```

Por lo cual, una gran área de mejora para el modelo podría ser optimizar el movimiento de los robots cuando no se encuentran cargando una caja y que, en vez de ser de manera aleatoria, se la asignaran diferentes cajas a cada uno para utilizar el PathFinding y optimizar tanto el tiempo de ejecución como la cantidad de movimientos realizados por los robots.

Finalmente, me gustaría mencionar que al trabajar en esta actividad integradora fue posible integrar tanto los conocimientos obtenidos de Mesa en Python como de gráficas computacionales en 3D con Unity, lo cual me resultó bastante interesante y resulta un gran primer acercamiento a la creación de sistemas multiagentes y encaminado a Machine Learning.