



## **Reporte Actividad M1**

Alumno:

Manuel Camacho Padilla

Profesor:

Dr. Oscar Fuentes

Fecha:

7 de Noviembre del 2022

Link al repositorio: <https://github.com/manu-camacho/Multi-Agent-System/tree/main/roomba-vacuum-cleaner>

## **Introducción:**

Elaborar un código en Python utilizando la librería de Mesa o Agentpy, para desarrollar una simulación de una Roomba Vacuum Cleaner, donde se presenta un espacio cuadrangular dado el número de casillas en columnas y filas. Este programa deberá de contener agentes que modelan la simulación de varios ciclos al modelo y se deberá obtener resultados al finalizar y hacer un análisis por cada aspecto de la simulación.

## **Variables de entrada del programa:**

- Columns
- Rows
- Number of agents
- Number of moves
- Percentage of dirty cells

## **Heurística de la simulación:**

Dado que los agentes dependen inicialmente de los steps asignados a los agentes, este podría seguir limpiando el cuarto aunque ya no haya suciedad en él. Por eso se limitó el número de movimientos que cada agente puede dar, y se detendrá la simulación. Así mismo, si todos los agentes han limpiado todas las casillas sucias, entonces la modelación finalizará.

## **Comportamiento emergente:**

Esta simulación de agentes no tiene comportamiento emergente, debido a que cada movimiento que se realiza en la aspiradora es de manera aleatoria en cada una de las casillas vecinas, y no comparten datos con otros agentes. Haciendo este algoritmo ineficiente.

## **Preceptos del programa:**

Terminar de limpiar todas las celdas en el menor tiempo posible o hasta que los movimientos por agentes hayan sido utilizados.

## **Condiciones de la cuadrícula:**



El tamaño de la cuadrícula es ingresado directamente desde el código del programa, así como las posiciones iniciales de los agentes que por defecto empiezan en (0,0). Los demás datos si son ingresados por el usuario en las simulaciones.


```

simulation_params = {
    "agents": UserSettableParameter(
        "number",
        "Number of Agents",
        5,
        description="Number of Agents",
    ),
    "rows": 25,
    "columns": 25,
    "moves": UserSettableParameter(
        "number",
        "Moves",
        25,
        description="Number of moves",
    ),
    "percentage_dirty_cells": UserSettableParameter(
        "slider",
        "Percentage of dirty cells",
        value=45,
        min_value=1,
        max_value=100,
        step=1
    ),
    "x_start": 0,
    "y_start": 0
}

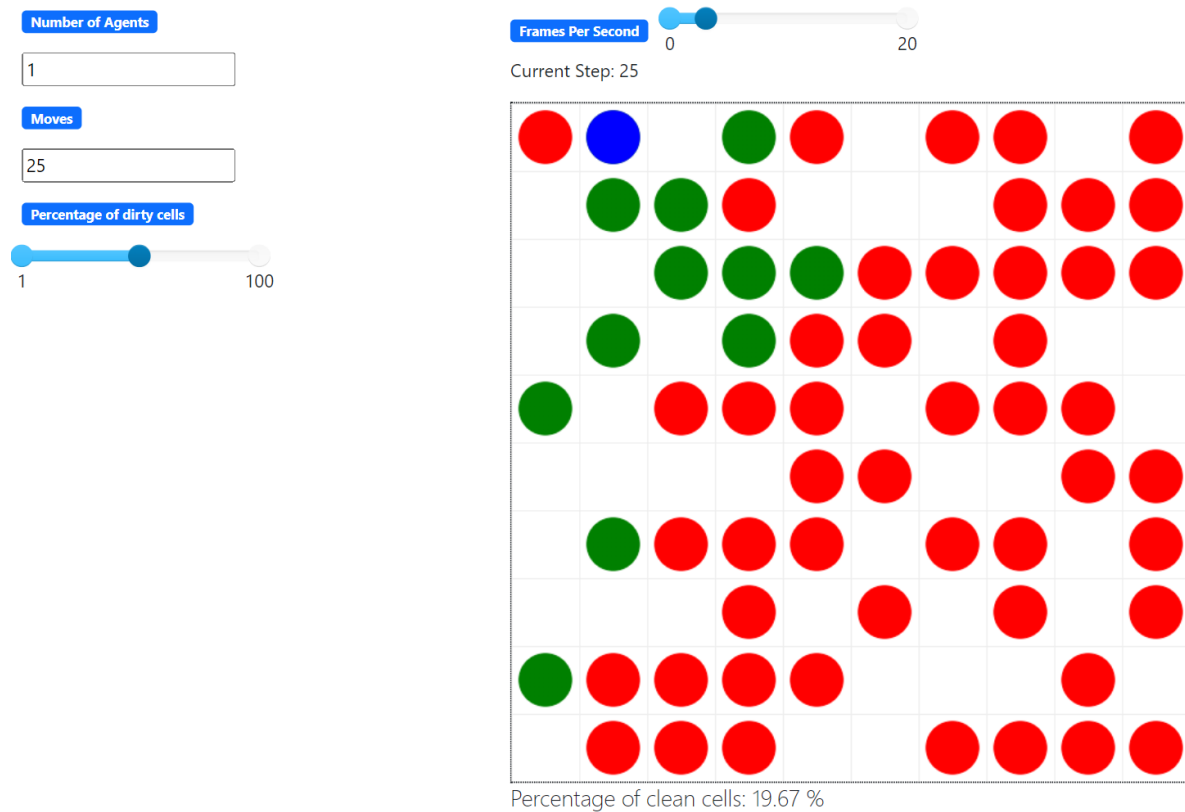
```

### Contemplaciones de la visualización:

Descripción	Representación
Agente aspiradora de color azul	
Agente suciedad de color rojo	

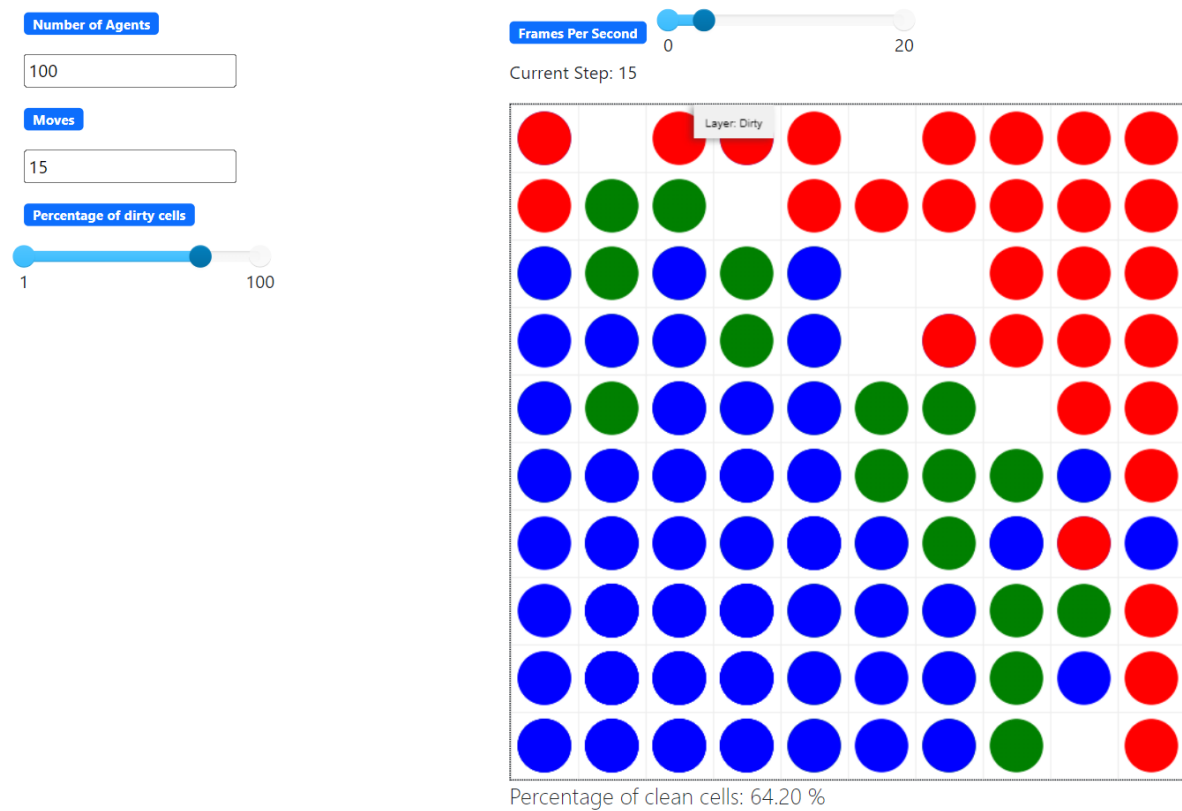
Agente suciedad limpiado de color verde	
Porcentaje de casillas limpiadas	Percentage of clean cells: 0.00 %
Número de agentes aspiradoras	<div>Number of Agents</div> <input type="text" value="1"/>
Número de movimientos permitidos por agente	<div>Moves</div> <input type="text" value="25"/>
Porcentaje de celdas sucias en la habitación	<div>Percentage of dirty cells</div> <div> <input type="range" value="1"/> <span>1</span> <span>100</span> </div>
Controles de la simulación	<div>Start Step Reset</div>
Velocidad de la simulación	<div>Frames Per Second</div> <div> <input type="range" value="0"/> <span>0</span> <span>20</span> </div>
Tiempo de la simulación representado en steps	Current Step: 0
Tiempo restante a la simulación	Time to finish: 0

## Observaciones en primera simulación:



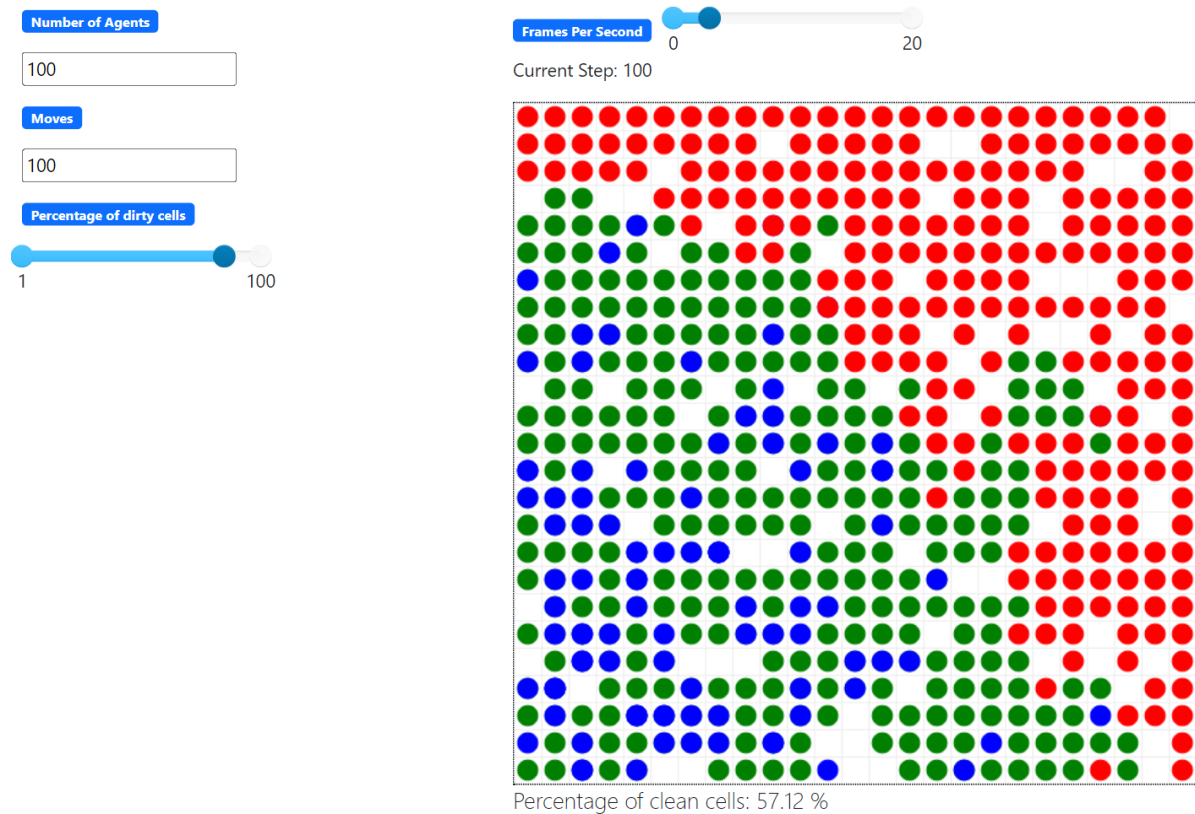
Dado que en esta simulación se presentó un solo agente se obtuvieron resultados poco eficientes, limpiando el 19.67% de la habitación. Además estaba limitado a solo dar 25 movimientos. Por lo que era imposible que el agente se pudiera desplazar por toda la habitación, Ya que, está era una cuadrícula de 10x10 casillas, dando como resultado 100 celdas posibles en las que podía estar, sin contemplar que sus movimientos son random.

## Observaciones de segunda simulación:



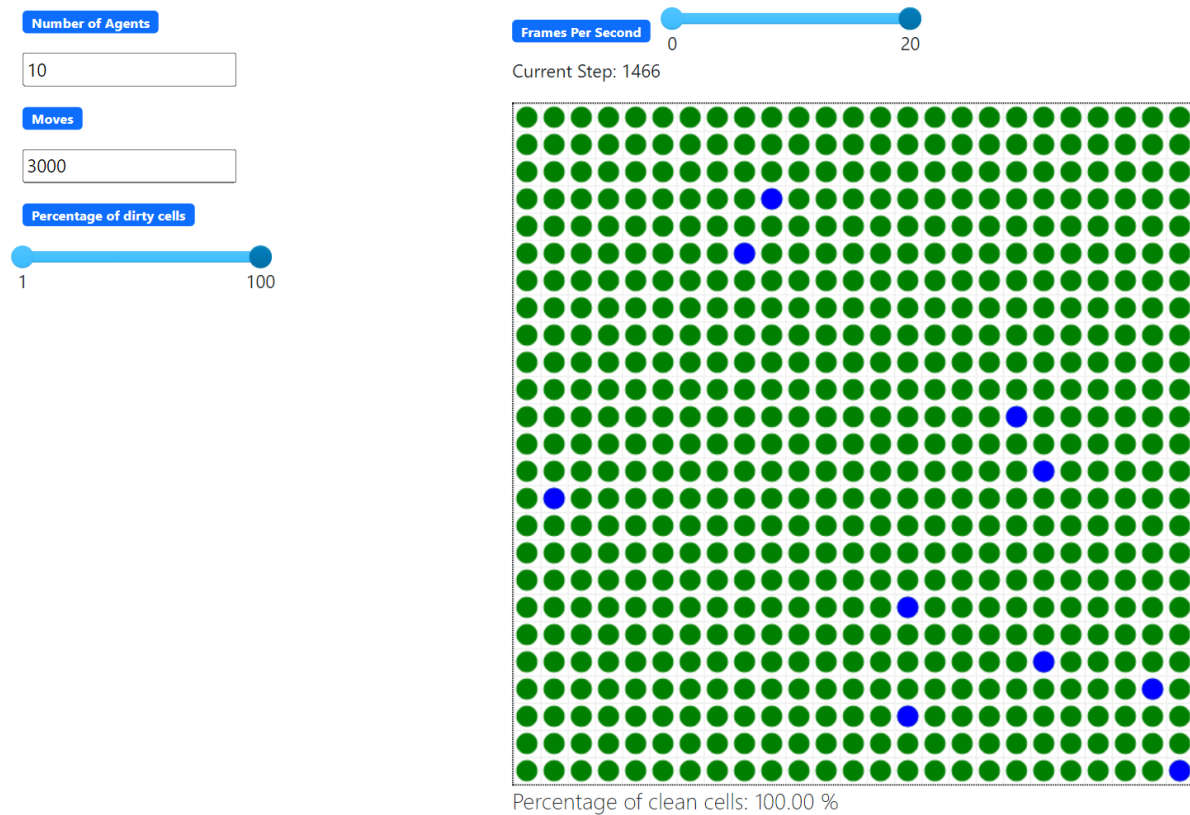
En esta simulación se tienen 100 agentes pero cada uno está limitado a dar 15 pasos, mientras que las celdas sucias aumentaron al 75%, tienen más espacio en los que limpias. Se obtuvo un porcentaje de de 64.20% celdas limpiadas.

## Observaciones de tercera simulación:



En esta simulación se presentaron 100 agentes con 100 movimientos, esperando que el porcentaje de limpieza aumentará, pero dado que el programa no es eficiente con los movimientos random, estas características no mejoraron en esta habitación donde la cuadrícula fue de 25x25 casillas. Complicando que los agentes se pudieran mover hacia la esquina opuesta de donde empezaron.

## Observaciones de cuarta simulación:



En este caso sí se obtuvo una porcenta satisfactorio de limpieza en el cuadrícula de 25 x 25, pero está vez con cada agente 3000 movimientos disponibles, a pesar de que solo fueron 10 agentes, pero se necesitó 1466 movimientos, o tiempo para que el algoritmo pudiera terminar con su función.

## Conclusión:

Esta actividad me ayudó a entender el funcionamiento básico de la librería de Mesa en Python, además de aclarar el funcionamiento de agentes en una simulación como esta. Este modelo fue poco eficiente, debido a que la lógica del movimiento era poco eficiente, dejando como resultado tiempos muy altos del programa o que no finalizará de limpiar todas las celdas. Pero se logró los resultados que se pedían para esta actividad. Me quedo satisfecho.