

ITESM Campus Santa Fe

Nombre del bloque:

TC2007B: Integración de seguridad informática en redes y sistemas de software (Gpo 402)

Nombre del entregable:

MA. Actividad: Roomba - Informe

Alumno:

Diego Cordova Rodríguez, A01781166

Profesor:

Octavio Navarro Hinojosa

Fecha de entrega:

22 de noviembre de 2025

Índice

1.- Introducción.....	3
Problema a Resolver.....	3
Propuesta de Solución.....	4
2.- Diseño de los agentes.....	5
Objetivo.....	5
Capacidad efectora.....	5
Percepción.....	6
Características.....	7
Métricas de desempeño.....	8
3.- Arquitectura de subsunción de los agentes.....	9
Simulación 1.....	9
Simulación 2.....	10
4.- Características del ambiente.....	11
PEAS.....	14
3.- Estadísticas recolectadas.....	15
Simulación 1.....	15
Simulación 2.....	20
4.- Análisis de resultados.....	25
5.- Conclusiones.....	27
6.- Referencias.....	28

1.- Introducción

Problema a Resolver

El problema planteado consiste en desarrollar una simulación para la limpieza de una habitación de $M \times N$ espacios a través de una roomba o un conjunto de estas, de manera eficiente. Cada roomba debe ser capaz de limpiar la habitación de basura en su totalidad, evitando obstáculos, a la vez que considera su batería restante. El espacio un cierto porcentaje inicial de obstáculos (ObstacleAgent) y de basura (TrashAgent), así como un tiempo máximo de ejecución, predeterminado a 3000 pasos.

Cuando una roomba tiene poca batería, debe ser capaz de regresar a su estación de carga de manera independiente. Además, cada roomba tiene capacidades sociales, intercambiando información cuando se encuentra con otra, para limpiar de manera más eficiente el espacio dado.

Consideraciones del problema:

- La batería de cada roomba inicia en 100% de carga.
- Cada acción que realiza el agente le quita 1% de batería. Si no realiza una acción, no pierde batería.
- El agente tiene que tener la capacidad de regresar a cargar batería para poder seguir limpiando.
- Cada episodio que el agente esté en la estación de carga, se recarga 5% de la batería.
- El agente tiene como objetivo limpiar el cuarto lo más que pueda en el tiempo máximo de ejecución, sin quedarse sin batería.

El sistema será probado en 2 simulaciones:

- Simulación 1: Una sola roomba que inicia en su estación de carga, en la celda [1, 1] de la habitación.
- Simulación 2: Múltiples roombas colocadas de forma aleatoria, cada una con una estación de carga inicial.

Propuesta de Solución

La propuesta de la solución consiste en que cada roomba sea capaz de regresar a su estación de carga de forma eficiente, desperdiциando la menor cantidad de pasos posible. Cada roomba es capaz de percibir únicamente sus 8 celdas vecinas (alrededor de ella), contando con una máquina de estados para decidir qué acción tomar en cada paso, de acuerdo a su ambiente y sus propios estados.

Para que la roomba sea capaz de regresar a alguna de las estaciones de carga conocidas, se plantea el uso del algoritmo A*. Inicialmente, la roomba tiene un arreglo de coordenadas para las estaciones de carga que conoce. Este mismo arreglo es inicializado con su estación de carga propia, y puede agregar nuevas estaciones de carga posteriormente, conforme las encuentra o son compartidas por otras roombas.

En cada paso, la roomba calcula la distancia para regresar a su estación de carga más cercana, mediante la distancia de Chebyshev, dada por:

$$\max(\text{abs}(x_{\text{roomba}} - x_{\text{estacion}}), \text{abs}(t_{\text{roomba}} - y_{\text{estacion}}))$$

Esto le permite comparar su batería actual con la necesaria para regresar a la estación. Si su nivel de batería es menor o igual a la distancia Chebyshev, calcula el recorrido hacia la estación con A* (Kamble, V. H., & Dale, M. P., 2022).

Cuando varias roombas (más de 1) coexisten en el mismo ambiente, cada una tiene habilidades sociales para poder limpiar de manera más eficiente. Cuando dos roombas cruzan caminos, intercambian información de celdas visitadas y sus propias estaciones de carga. Esto permite que en el futuro, cuando una roomba tenga poca batería, sea capaz de calcular la ruta hacia la estación de carga más cercana, siendo incapaz de tomar las estaciones donde ya se encuentra una roomba cargándose.

Además, se propone la posibilidad de que una roomba sea capaz de buscar la ruta más cercana hacia una celda no visitada cuando esta ya exploró las 8 celdas en su vecindario. De la misma manera en la que funciona A*, busca entre sus vecinos todas las ramas hasta

encontrar la primera celda que no haya sido visitada; después de esto, calcula la ruta hasta ella mediante A*.

Al momento de tener muchas roombas en un mismo ambiente, existe la posibilidad de que 2 o más compitan por una misma estación de carga, si es que esta es la más cercana para ellas. En este caso, la primera roomba que llegue a la estación de carga la ganará, mientras que las otras serán capaces de detectar que está siendo utilizada y esperarán a un lado, evitando desperdiciar energía en buscar otra estación, lo que la podría llevar a quedarse sin batería.

2.- Diseño de los agentes

En este caso, los únicos agentes presentes en el ambiente son las roombas. Cada roomba consiste en un pequeño robot completamente independiente, que es capaz de tomar decisiones en base a lo que percibe de su ambiente y sus propios estados internos. A continuación se define su diseño, así como sus respectivas características para resolver las necesidades del problema:

Objetivo

Limpiar la mayor cantidad de basura posible antes de que termine el tiempo máximo de ejecución de la simulación, evitando quedarse sin batería.

Capacidad efectora

Las roombas son capaces de interactuar y modificar su entorno de manera directa a través de la recolección de basura. Además, cuentan con diferentes acciones posibles en cada episodio:

- Revisar su propio nivel de batería
 - Si tiene más de la batería necesaria para regresar a su estación de carga, revisa sus celdas vecinas para buscar basura.
 - Si tiene la batería justa para regresar a su estación de carga, regresa.
 - No consume batería

- Cargar batería en estación de carga
 - +5% de batería por episodio
- Revisar celdas vecinas
 - Revisa sus 8 celdas vecinas (dirección vertical, horizontal o diagonal)
 - No consume batería
- Moverse a una celda determinada
 - Se puede mover máximo 1 celda en cualquier dirección (vertical, horizontal o diagonal) por episodio.
 - Despues de moverse, calcula la ruta hacia sus estaciones de carga y guarda la ruta más corta.
 - -1% de batería
- Intercambiar información con otras roombas
 - Si hay una roomba en una celda vecina, agrega las estaciones de carga conocidas y celdas visitadas de la otra roomba a las suyas, y viceversa.
- Limpiar celda actual
 - Elimina la basura en su celda actual
 - -1% de batería

Percepción

Las roombas son capaces de percibir el ambiente de diversas maneras. Por un lado, son capaces de detectar sus ocho celdas vecinas (en sentido horizontal, vertical y diagonal) en busca de obstáculos, basura u otras roombas. Esto les permite tomar decisiones para moverse hacia la celda adecuada. De la misma forma, son capaces de percibir su propia celda, con la finalidad de detectar si hay basura, y en dado caso, limpiarla.

Por otro lado, son capaces de percibir otras roombas en sus celdas vecinas, lo que les permite intercambiar información con ellas. Además, si se da el caso en el que todas las celdas vecinas han sido visitadas, son capaces de buscar la celda no visitada/basura más cercana a través de sus vecinos, mediante una búsqueda similar a la realizada por A*.

Características

- **Proactividad:**

Los roombas son proactivas, pues son capaces de tomar la iniciativa de cambiar de celda para cumplir sus objetivos de limpiar toda la basura del ambiente y no quedarse sin batería. De la misma forma, son capaces de regresar a su estación de carga si detectan un bajo nivel de batería. De esta manera, actúan en base a sus reglas establecidas para lograr sus objetivos.

- **Reactividad:**

Son reactivas al poder de percibir su entorno mediante la detección de celdas vecinas, ya sea en busca de obstáculos, basura, otras roombas o incluso celdas no visitadas. Cada roomba responde oportunamente a los cambios: Si hay basura, la recoge; si no hay celdas visitadas, busca un camino hacia ellas; si tiene poca batería, calcula el regreso a su estación de carga más cercana.

- **Habilidad social:**

En el caso de la simulación 1, la única roomba no cuenta con ningún tipo de habilidad social al no haber otras roombas presentes. Por otro lado, en la simulación 2, las roombas sí son capaces de interactuar con otras, intercambiando información de estaciones de carga conocidas y celdas previamente visitadas. De esta forma, las roombas son capaces de interactuar con otras para cumplir sus objetivos de limpiar basura y no quedarse sin batería.

- **Racionalidad:**

Las roombas se pueden considerar como agentes racionales a un bajo nivel. Buscan tomar acciones para lograr sus objetivos con su información actual (celdas vecinas, celdas visitadas, nivel de batería propio, etc.). Si bien no son capaces de evaluar consecuencias como deseables/indeseables más allá de sus reglas establecidas, su desempeño se puede medir con base a su batería, pasos dados en total, porcentaje de basura limpiada y tiempo total de limpieza dentro de la simulación.

A pesar de no ser omniscientes ni ser capaces de planificar a largo plazo (futuros pasos), consiguen información de sus celdas vecinas para calcular su batería mínima necesaria para ser capaces de regresar a su estación de carga, actuando de forma independiente, sin intervención de otros agentes que le digan cómo comportarse directamente, tomando acciones para optimizar su uso de batería.

Métricas de desempeño

Las roombas cuentan con diferentes métricas de desempeño para medir si se ha cumplido con su objetivo o no, y en qué medida. A continuación se presentan las métricas de desempeño definidas:

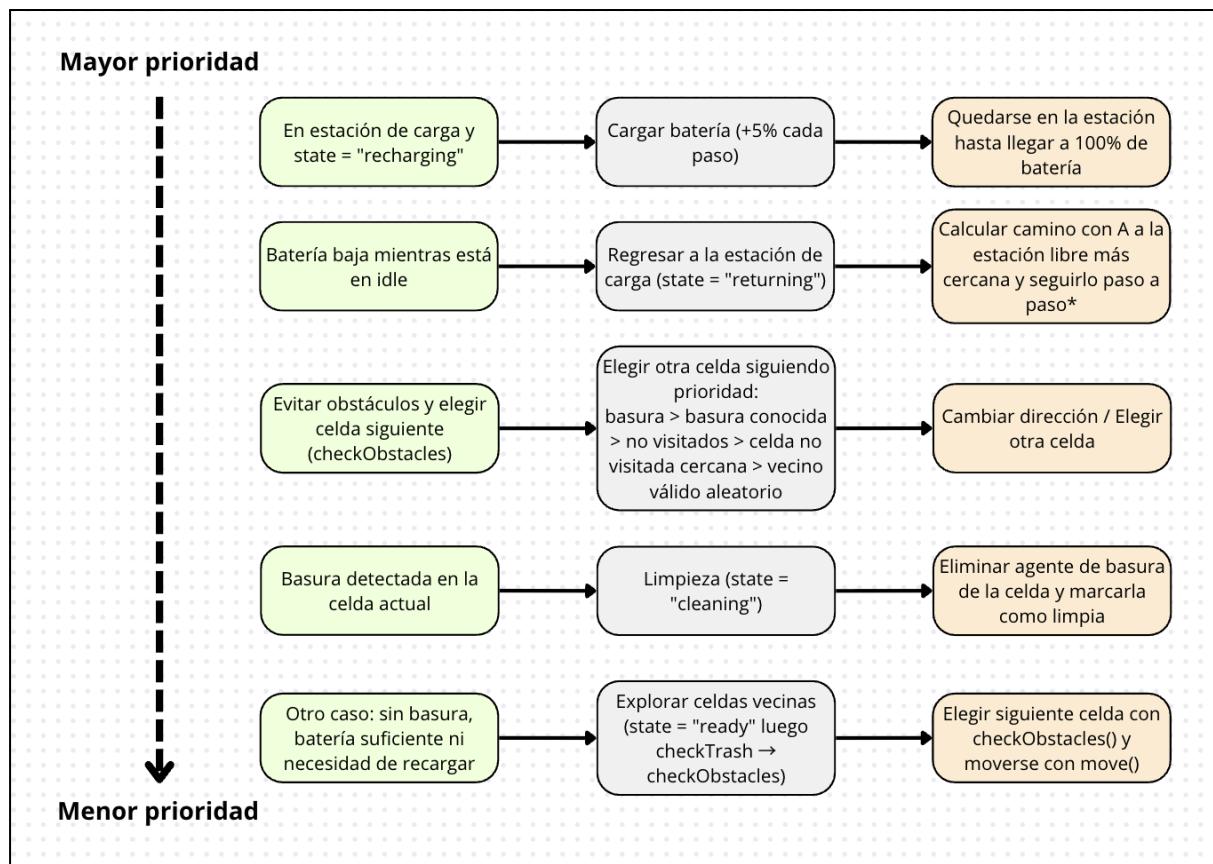
- Tiempo necesario para limpiar el cuarto (pasos)
 - Es el tiempo (número de pasos dentro de la simulación) que le toma a la roomba limpiar toda la basura del ambiente.
 - Se define un tiempo máximo de limpieza (predeterminado a 3000).
 - Sirve para identificar la velocidad con la que las roombas son capaces de limpiar basura.
- Porcentaje de celdas limpias al terminar la simulación
 - Con esta métrica, se puede identificar el porcentaje de basura que la roomba logró limpiar en el tiempo de ejecución de la simulación.
- Número total de movimientos realizados por cada roomba
 - Se refiere al número de veces que la roomba se movió a otra celda durante el tiempo de ejecución de la simulación.
 - El objetivo de las roombas es limpiar la basura en el menor tiempo posible, por lo que cada paso que den debe ser crucial para acercarlas a este objetivo.
- Porcentaje de batería de la roomba
 - Porcentaje final de batería de una roomba al finalizar la simulación.
 - Si una roomba llega a quedarse sin batería, no será capaz de limpiar el cuarto.

3.- Arquitectura de subsunción de los agentes

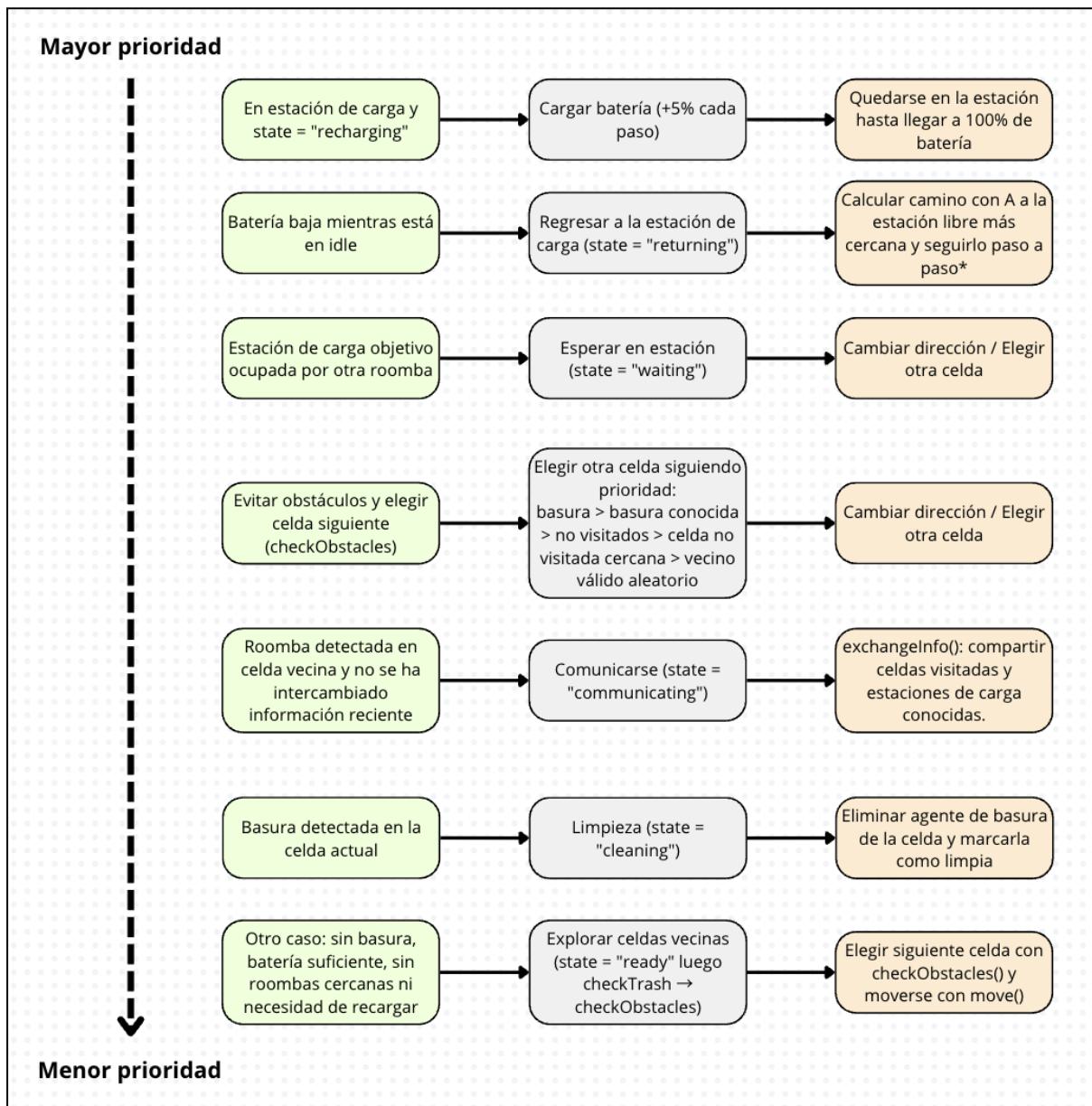
La toma de decisiones de las roombas se realizó tomando en cuenta la prioridad de cada una de sus acciones, organizadas en capas jerárquicas. Primero se atienden las necesidades básicas del agente, comenzando por no quedarse sin batería, seguido de la detección de su ambiente para cooperar con otras roombas, limpieza de basura o buscar rutas hacia celdas no visitadas.

En este sentido, estas acciones se definieron a través de una máquina de estados basada en percepciones locales (nivel de batería propio) y del ambiente (celdas vecinas). A continuación se presenta un diagrama para la toma de decisiones de la roomba a través de su máquina de estados, organizada de mayor a menor prioridad, para cada simulación:

Simulación 1



Simulación 2



La única diferencia entre ambas simulaciones es que en la simulación 2, las roombas tienen capacidad de intercambiar información con otras sombras. De igual forma, las estaciones de carga pueden llegar a ser bloqueadas por otras roombas.

Después de cada acción de la roomba, pierde 1% de batería. Las acciones que comprenden estos casos son las siguientes:

- Moverse a una celda
- Limpiar basura

De esta manera, cada roomba es capaz de tomar decisiones en base a la información actual con la que cuenta, aumentando su utilización de batería de manera eficiente, cooperando con otras roombas, moviéndose de manera eficiente y limpiando la mayor cantidad de basura en el menor tiempo posible.

4.- Características del ambiente

El ambiente de ambas simulaciones consiste en un espacio en forma de matriz (grid) de NxM celdas, donde cada una representa un espacio dentro de una habitación. Dentro del ambiente, se pueden encontrar diferentes tipos de celdas. A continuación se presenta una tabla que detalla cada una de ellas:

Celda	Descripción	Representación en el grid
Con basura (TrashAgent)	Las roombas deben limpiar la celda, estando sobre esta.	“X” verde
Obstáculo (ObstacleAgent)	Bloquea el paso de las roombas. En ambas simulaciones, se inicializa un cierto porcentaje de celdas de este tipo, con un predeterminado de 10%.	Recuadro gris
Estación de carga (Station)	Celda inicial de una roomba, mismas que pueden volver a esta para recargar su batería. En la simulación 1, la única estación de carga presente se inicia en la celda [1,1]. En la simulación 2, hay una estación de carga por cada roomba existente, en celdas aleatorias.	Triángulo rojo
Con roomba (Roomba)	Representa una roomba moviéndose por el grid. En la simulación 1, la única roomba existente se inicia en la celda [1,1]. En la simulación 2, las roombas se crean en celdas aleatorias, dependiendo del número de roombas definidas, con	Círculo azul

	un número predeterminado de 5 roombas iniciales.	
Suelo (Ground)	<p>Es una celda normal del ambiente.</p> <p>Su color ayuda a identificar en la visualización si ya ha sido explorada o no previamente por una roomba.</p> <p>Son generadas en cada celda del ambiente.</p>	<p>Si ha sido visitada: Recuadro gris claro</p> <p>Si no ha sido visitada: Recuadro blanco</p>

Al iniciar la simulación, las celdas de cada tipo son inicializadas con valores por defecto, mismos que pueden ser modificados por el usuario en la interfaz lateral izquierda de la pantalla.

Consideraciones adicionales:

- Una roomba solo puede ocupar una celda a la vez.
- Dos roombas nunca podrán compartir la misma estación de carga al mismo tiempo.
- Las roombas no pueden pasar por encima de celdas con obstáculos.
- Las estaciones de carga permiten recargar 5% de la batería de la roomba por cada episodio que permanezca sobre ella.
- El ambiente permanece idéntico para ambas simulaciones; sin embargo, para la simulación 1, no se puede cambiar la cantidad de roombas y esta, junto con su estación de carga, siempre estarán inicializadas en la coordenada [1, 1].

A continuación se presentan las características principales del ambiente de las simulaciones:

- **Parcialmente Accesible:**

El ambiente de ambas simulaciones es parcialmente accesible, pues los agentes (roombas) solo tienen acceso a su propia celda y celdas vecinas inmediatas. Cabe destacar que las roombas no tienen acceso a todas las celdas del grid, siendo únicamente capaces de tomar decisiones en base a la información que va recopilando a lo largo de la ejecución del programa, como es a través de basura detectada, estaciones de carga conocidas o celdas previamente visitadas.

- **No determinista:**

El ambiente es no determinista. A pesar de que las roombas tienen una preferencia al momento de elegir celdas y moverse a ellas (con basura → no visitadas → otras), el ambiente genera celdas de basura y obstáculos de manera aleatoria, por lo que las roombas no siempre se moverán hacia las mismas celdas. Por otro lado, incluso si se sabe que una roomba se moverá hacia una celda de basura, si tiene 2 celdas de este tipo entre sus vecinos, se moverá hacia una aleatoria.

- **No episódico:**

El ambiente es no episódico, pues solo se altera en base al episodio actual, siendo que cada acción de las roombas depende de sus acciones previas. A pesar de esto, en el método checkObstacles de la clase Roomba, se tienen en cuenta celdas visitadas durante episodios previos de la simulación para elegir la próxima celda a visitar, utilizando esta información para influenciar sus decisiones actuales. No obstante, las roombas no necesitan razonar sobre las interacciones entre este y futuros episodios.

- **Estático / Dinámico:**

En el caso de la simulación 1, se cuenta con un ambiente estático, pues el ambiente permanecerá sin cambios de no ser por la realización de acciones por la única roomba presente. En este caso, la acción concreta que modifica el ambiente es la limpieza de basura.

Por otro lado, en la simulación 2, se cuenta con un ambiente dinámico, pues cada roomba es capaz de limpiar diferentes celdas y contribuir con información de celdas visitadas a otros. Además, el hecho de que una roomba llegue a una estación de carga antes que otra genera una situación en la que el agente que no pudo recargar su batería no fue capaz de controlar. Así, aunque una roomba no realice una acción en concreto, otra puede modificar el entorno por su cuenta.

- **Discreto:**

En ambas simulaciones, el ambiente es discreto, pues se encuentra formado por un espacio con NxM celdas ocupadas por agentes, quienes solo pueden moverse entre sus vecinos en cada paso. Además, los agentes tienen un conjunto limitado de acciones (moverse, recoger basura, recargarse, etc.) por lo que hay un número fijo y finito de acciones y percepciones posibles.

PEAS

- **Performance**

El ambiente cuenta con diferentes métricas de rendimiento para evaluar la capacidad de los agentes de cumplir su objetivo. Entre estas, se encuentran las siguientes: Porcentaje de celdas limpias al final de la simulación, pasos hasta limpiar todas las celdas, número de movimientos de cada roomba y porcentajes de batería.

- **Environment**

El ambiente consiste en un espacio de NxM celdas de basura (TrashAgent), obstáculos que bloquean el paso de roombas (ObstacleAgent), estaciones de carga para que las roombas recarguen su batería (Station) y roombas (Roomba), cuyo objetivo es limpiar toda la basura en el menor tiempo posible, antes de llegar al tiempo máximo de la simulación, colaborando entre ellas.

- **Actuators**

En cuanto a interacciones en el entorno, las roombas poseen diferentes actuadores. Por ejemplo, son capaces de eliminar celdas de basura, ocupar estaciones de carga o incluso detectar obstáculos a su alrededor para preferir ir hacia celdas no visitadas previamente.

- **Sensors**

Dentro del ambiente, las roombas son capaces de percibir información de este, revisando directamente las celdas vecinas para verificar que no tengan obstáculos, no hayan sido visitadas y que preferiblemente contengan basura. Además, son capaces de percibir su información propia, como su porcentaje de batería, celdas visitadas previamente y diferentes estaciones de carga.

3.- Estadísticas recolectadas

Las estadísticas obtenidas de las simulaciones fueron obtenidas de manera gráfica (en la interfaz de solara), así como a través de la terminal de la aplicación.

Simulación 1

A continuación se presentan las estadísticas recolectadas a través de diferentes iteraciones de la simulación 1, donde una sola roomba es la encargada de limpiar toda la basura del ambiente.

Para recopilar la información, se recopilaron los datos de 2 formas principales:

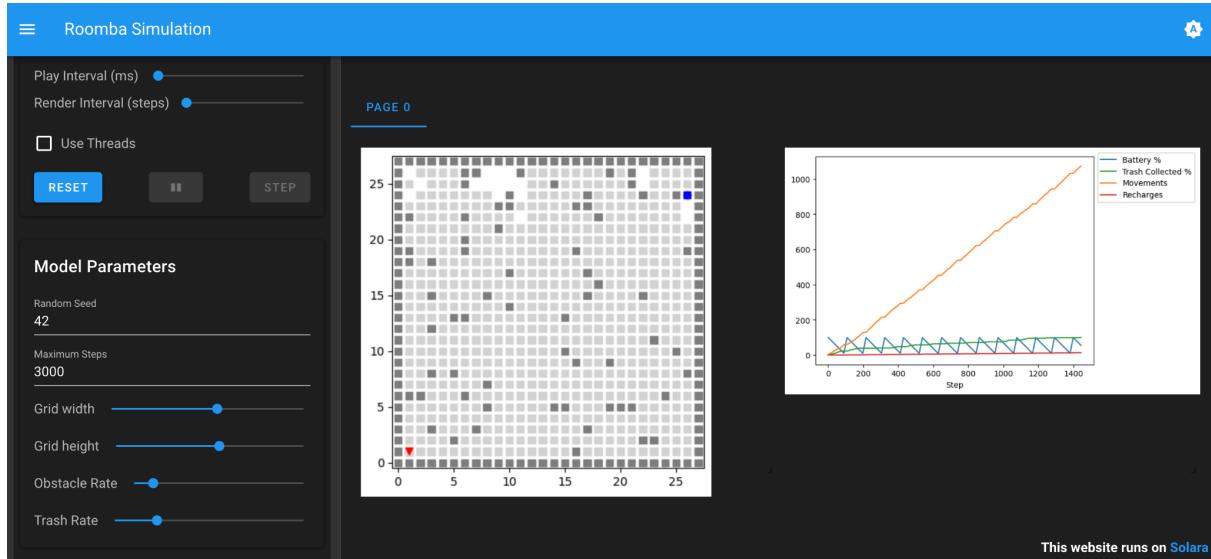
- **Gráfica de líneas con datos generales:** Contiene el porcentaje de batería de la roomba, porcentaje de basura recolectada, número de movimientos y cantidad de recargas a lo largo del tiempo de ejecución de la simulación (Número de pasos).
- **Detalles numéricos de la simulación:** Dentro de la terminal, se imprimen los principales datos recopilados, como el número de pasos precisos de la simulación y datos puntuales de la roomba, como su batería y número de movimientos.

En un inicio, se realizó una simulación con los siguientes parámetros iniciales:

- Semilla: 42
- Pasos máximos: 3000
- Dimensiones: 28x28
- Porcentaje de obstáculos: 10%
- Porcentaje de basura: 20%

A continuación se presentan los resultados obtenidos al finalizar la ejecución de la simulación:

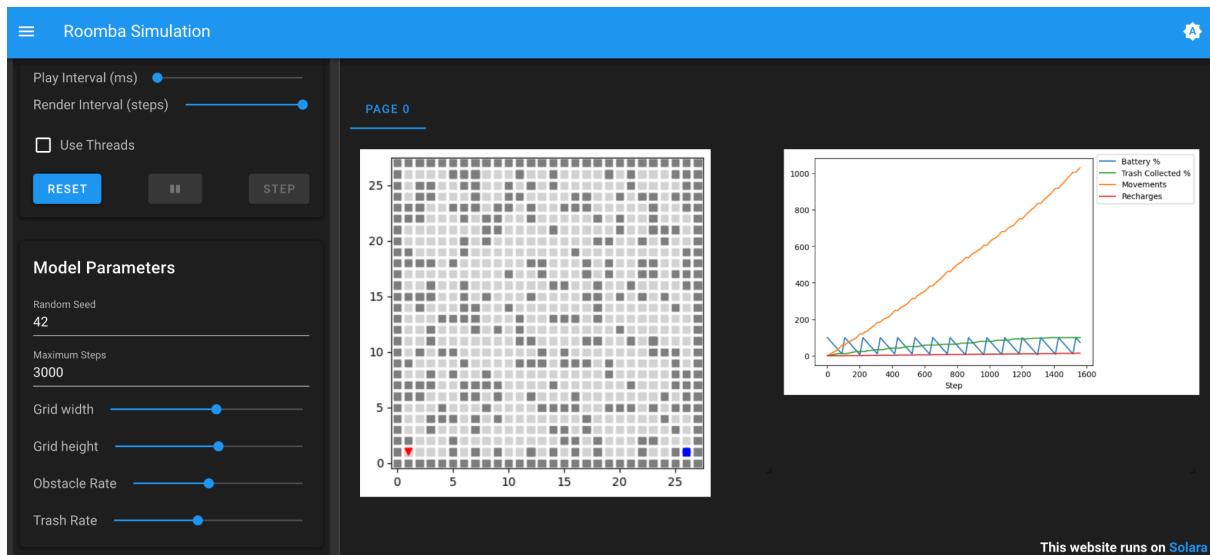
Resultados de simulación:



----- Simulation Results -----
Time (Steps): 1443.0
Trash Collected %: 100.0
Battery %: 55.0
Movements: 1073.0
Recharges: 13.0

Como se puede observar, la roomba fue capaz de limpiar todas las celdas por su cuenta en tan solo 1443 pasos, un número menor a la cantidad máxima de pasos establecidos (3000). A simple vista, el dato que más destaca en la gráfica es la gran cantidad de movimientos de la roomba. Esto indica que, a pesar de que pasa bastante tiempo recargándose (13 recargas en total), es muy superior la cantidad de pasos que da a los que pasa cargándose, dando como resultado un mayor tiempo de exploración en la simulación.

Posteriormente a esto, se realizó una simulación con una mayor cantidad de obstáculos y basura, ambos en 40%. Los resultados obtenidos fueron los siguientes:

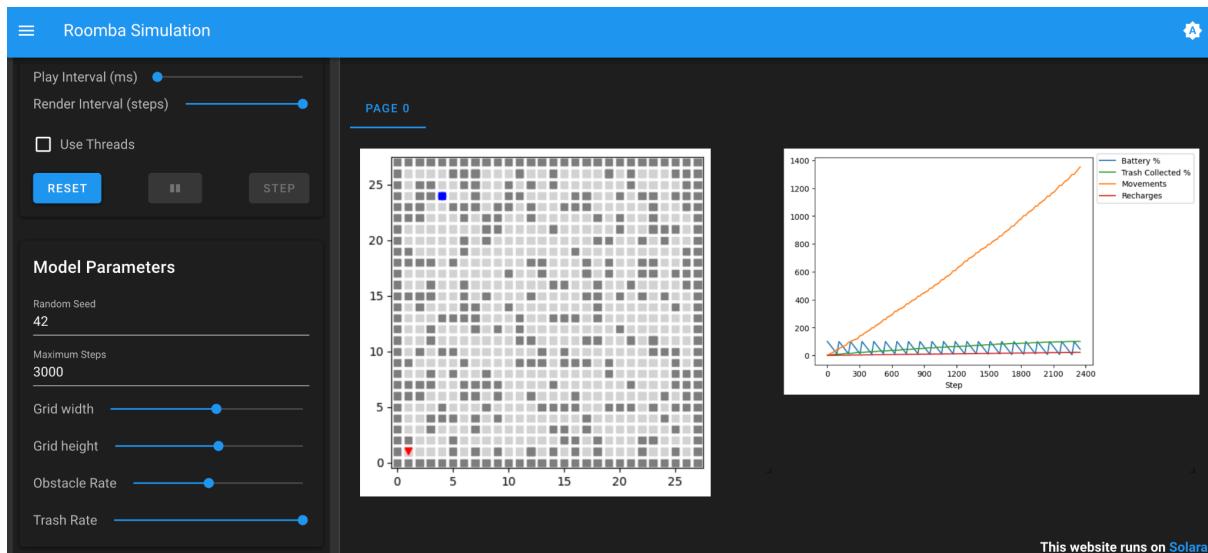


----- **Simulation Results** -----
Time (Steps): 1559.0
Trash Collected %: 100.0
Battery %: 73.0
Movements: 1030.0
Recharges: 14.0

De acuerdo a lo observado, una mayor cantidad de obstáculos y basura resultó en un aumento de aproximadamente 100 pasos totales con respecto a la simulación anterior. Sin embargo, la cantidad de movimientos de la roomba se mantuvo relativamente igual, incluso menor que antes. Esto indica que a pesar de un ligero incremento en la cantidad de basura y obstáculos, la roomba es capaz de limpiar con un buen nivel de eficiencia. El incremento de tiempo total de simulación se puede atribuir a un mayor número de recargas, donde la roomba debe permanecer un cierto tiempo sin moverse para recuperar su batería.

Después de esto, se realizó una simulación con 90% de celdas de basura, manteniendo el 40% de obstáculos. Este último no fue incrementado ya que lleva a situaciones donde la

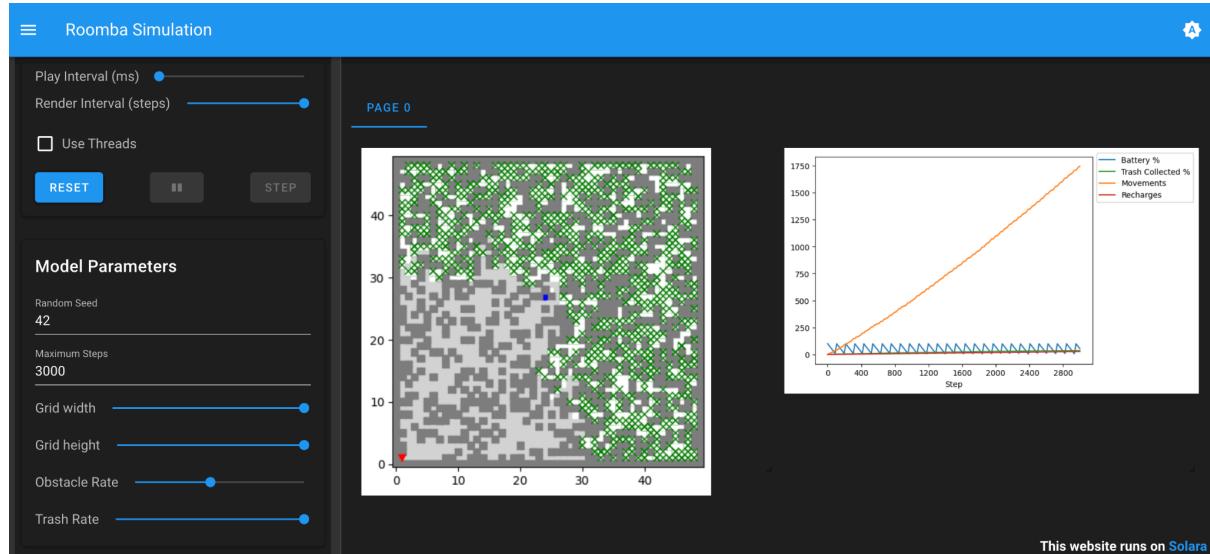
basura es inaccesible por la roomba, estando rodeada de obstáculos. Los resultados obtenidos fueron los siguientes:



----- **Simulation Results** -----
Time (Steps): 2347.0
Trash Collected %: 100.0
Battery %: 44.0
Movements: 1352.0
Recharges: 21.0

En esta ocasión, la cantidad de movimientos de la roomba aumentó un total de aproximadamente 300 pasos. De igual manera, esto influyó en la cantidad de pasos totales de la simulación, al igual que el número de recargas. Al haber más basura por recoger, la roomba debía utilizar más batería, pues recordemos que la acción de recoger basura le resta 1%.

Finalmente, se realizó una prueba con un tamaño de ambiente (grid) de 50x50 celdas, el tamaño máximo permitido para la simulación. Se mantuvo el 90% de celdas de basura y 40% de obstáculos. Los resultados fueron los siguientes:



----- Simulation Results -----
Time (Steps): 3000.0
Trash Collected %: 36.5171249397
Battery %: 53.0
Movements: 1744.0
Recharges: 27.0

En este caso, la roomba no fue capaz de recoger toda la basura del ambiente. A pesar de nunca quedarse sin energía, no fue lo suficientemente veloz para limpiar todas las celdas. Este se trata de un caso especial, donde el ambiente tiene un gran tamaño en comparación a las capacidades de la roomba. Este caso en particular ejemplifica cómo la roomba se podría beneficiar de colaborar con otras, siendo capaces de limpiar el ambiente de manera más eficiente.

Simulación 2

A continuación se presentan las estadísticas recolectadas a través de diferentes iteraciones de la simulación 2, donde muchas roombas son capaces de colaborar, compartiendo su información de celdas visitadas, así como estaciones de carga conocidas. Primero se realizó una simulación con los siguientes parámetros iniciales.

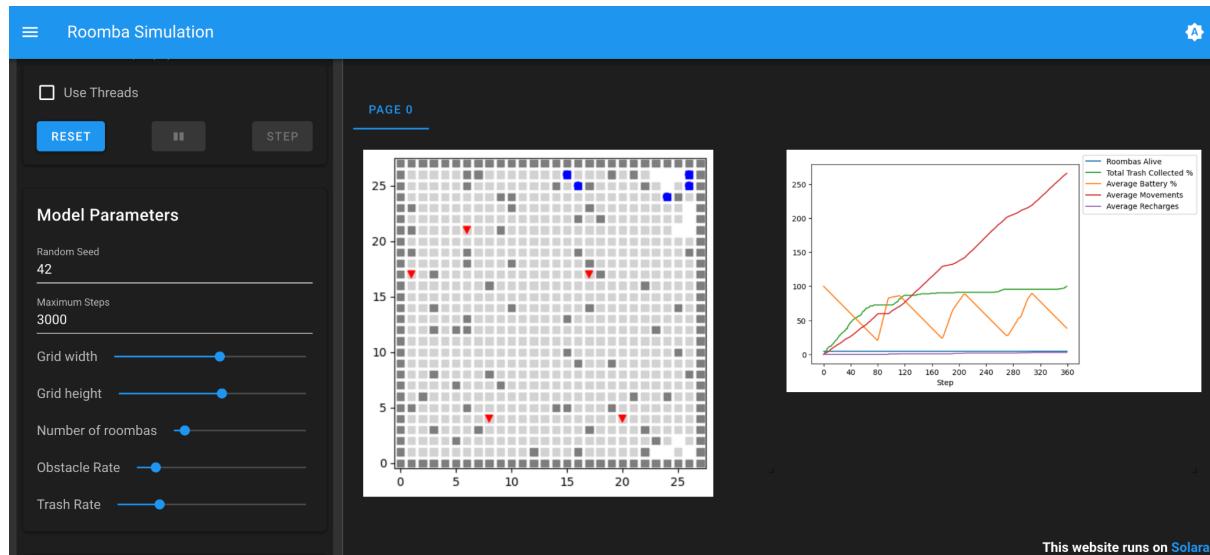
De la misma manera que con la simulación 1, se recopilaron los datos de 2 formas principales:

- **Gráfica de líneas con datos generales:** Contiene promedios de porcentaje de batería de las roombas, porcentaje de basura total recolectada, número promedio de movimientos y de cantidad de recargas a lo largo del tiempo de ejecución de la simulación (Número de pasos).
- **Detalles numéricos de la simulación:** Dentro de la terminal, se imprimen los principales datos recopilados, como el número de pasos precisos de la simulación y datos puntuales de cada roomba, como su batería y número de movimientos.

Parámetros iniciales utilizados:

- Semilla: 42
- Pasos máximos: 3000
- Dimensiones: 28x28
- Número de roombas: 5
- Porcentaje de obstáculos: 10%
- Porcentaje de basura: 20%

A continuación se presentan los resultados obtenidos al finalizar la ejecución de la simulación:

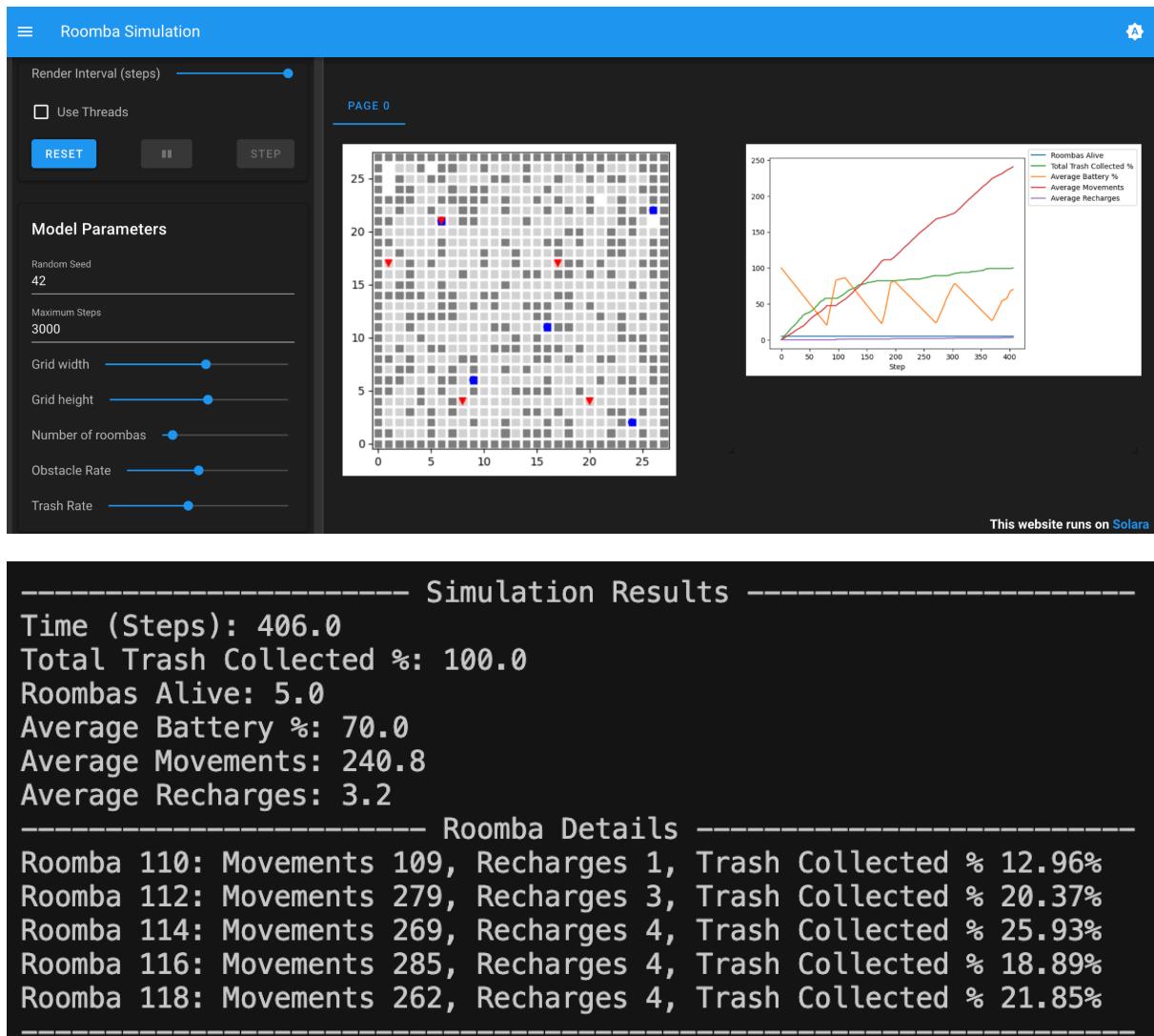


----- Simulation Results -----
Time (Steps): 359.0
Total Trash Collected %: 100.0
Roombas Alive: 5.0
Average Battery %: 38.4
Average Movements: 265.6
Average Recharges: 3.0
----- Roomba Details -----
Roomba 110: Movements 268, Recharges 3, Trash Collected % 25.19%
Roomba 112: Movements 274, Recharges 3, Trash Collected % 20.00%
Roomba 114: Movements 270, Recharges 3, Trash Collected % 15.56%
Roomba 116: Movements 267, Recharges 3, Trash Collected % 14.07%
Roomba 118: Movements 249, Recharges 3, Trash Collected % 25.19%

Como se puede observar, 5 roombas fueron capaces de limpiar las celdas en un tiempo sustancialmente menor que una sola. Esto indica que la capacidad de colaboración de las roombas les permiten reducir los tiempos totales para la recolección de basura. En este caso, se tardaron tan solo 359 pasos. Gracias a la capacidad de las roombas de esperar al lado de una estación de carga si está ocupada, su probabilidad de quedarse sin energía se reduce significativamente.

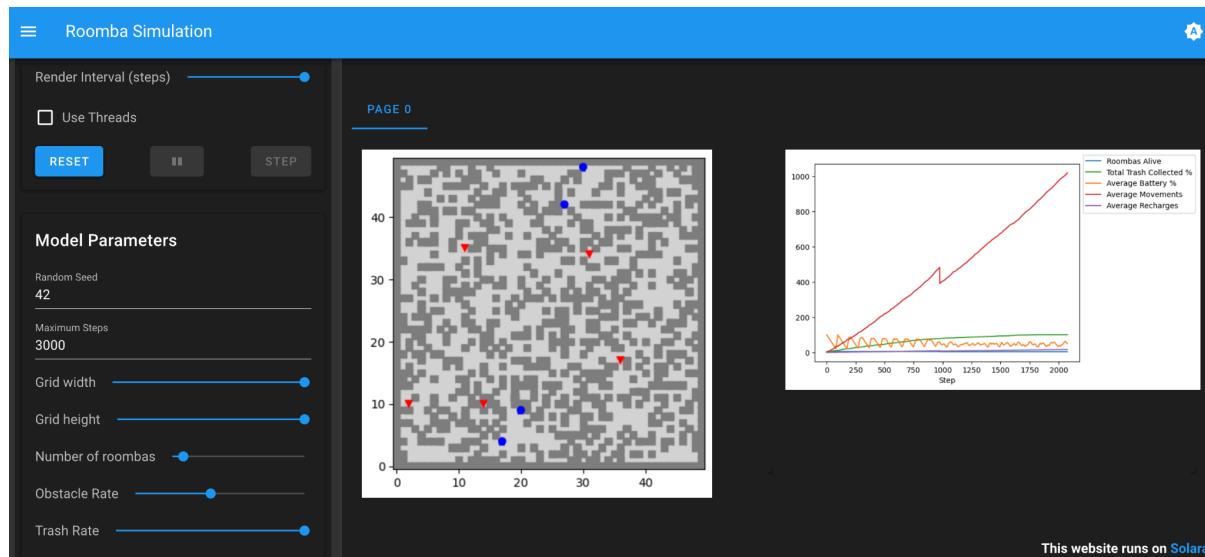
Además, se puede observar que, aunque cada roomba recogió una cantidad de basura diferente, todas superaron el 10%. Este porcentaje depende de gran forma de que una roomba se encuentre con otra. A pesar de esto, su cantidad de movimientos se mantuvo relativamente similar, teniendo todas un valor aproximado de 265 movimientos en total.

Después de esto, se realizó una simulación aumentando los valores de obstáculos y basura al 40%. Los resultados fueron los siguientes:



En este caso, las roombas fueron capaces de limpiar toda la basura del ambiente con gran velocidad, aunque aumentando ligeramente su cantidad de movimientos y número de veces que tuvieron que recargar su batería. En este caso, hubo una roomba que se recargó una sola vez y recolectó tan solo el 12.96% de la basura. Esto se debe a que estuvo esperando bastante tiempo por una estación de carga, en estado “waiting”. A pesar de esto, ninguna roomba se quedó sin energía a lo largo de toda la simulación.

Después de esto, se aumentó el porcentaje de basura a 90% y tamaño del ambiente a 50x50 celdas. Los resultados fueron los siguientes:



----- **Simulation Results** -----

Time (Steps): 2075.0
Total Trash Collected %: 100.0
Roombas Alive: 4.0
Average Battery %: 49.4
Average Movements: 1018.0
Average Recharges: 16.4

----- **Roomba Details** -----

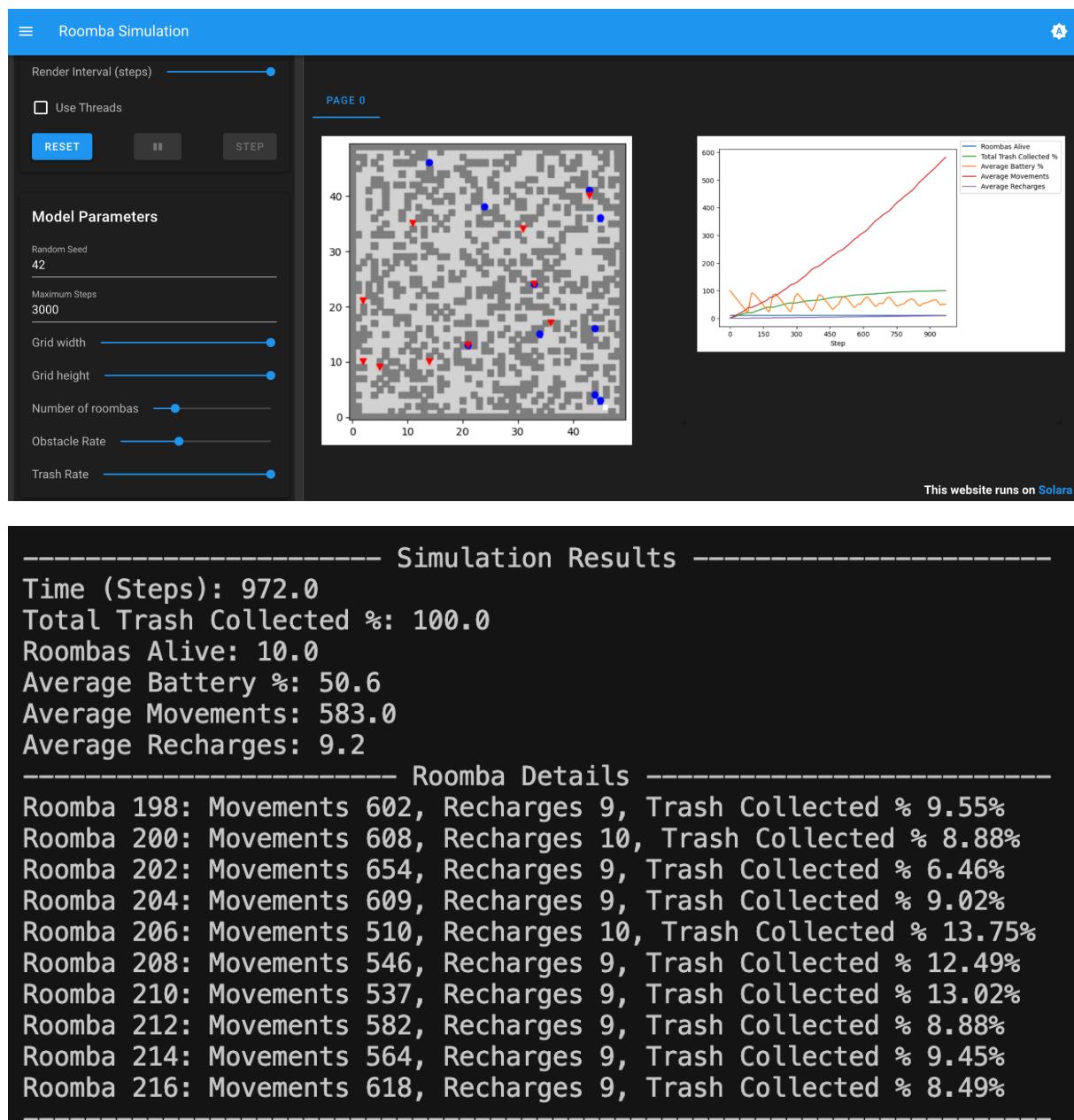
Roomba 200: Movements 1319, Recharges 20, Trash Collected % 16.98%
Roomba 202: Movements 1317, Recharges 20, Trash Collected % 19.05%
Roomba 204: Movements 1213, Recharges 21, Trash Collected % 23.59%
Roomba 206: Movements 1241, Recharges 21, Trash Collected % 22.82%

En este caso, las roombas fueron capaces de limpiar el 100% de la basura antes del límite de pasos establecido (3000). A pesar de esto, una roomba llegó a quedarse sin batería. La razón por la que esto sucedió, a pesar de que una roomba sea capaz de esperar en una estación hasta que esté desocupada, fue la manera en la que se calcula la distancia a las estaciones conocidas.

Con el fin de hacer la ejecución de la simulación más eficiente, las roombas únicamente estiman la distancia hacia sus estaciones conocidas a través de la distancia Chebyshev, que no considera si hay celdas con obstáculos o no. A pesar de que las roombas cuenten con un margen de 20 pasos para regresar a su estación, esta estimación de distancia puede provocar que se lleguen a quedar sin batería, en un caso donde haya una gran cantidad de obstáculos.

Una forma en la que se podría solucionar esta situación sería calculando la distancia mediante A*, que sí podría considerar la ruta óptima hacia la estación de carga más cercana; sin embargo, sería sumamente costoso computacionalmente realizar este cálculo para muchas roombas a la vez.

Finalmente, se probó la misma simulación, con los mismos parámetros, pero aumentando la cantidad de roombas a 10. Los resultados obtenidos fueron los siguientes:



En este caso, al momento de haber más roombas, cada una debe recorrer una mayor cantidad de distancia para limpiar celdas. Esto es debido a la colaboración y comunicación de celdas no visitadas conocidas, evitando que se vuelvan a explorar celdas visitadas por otras roombas. Al alejarse menos, en promedio, de las estaciones de carga, ninguna roomba se quedó sin energía. Así, se logró realizar la limpieza completa del ambiente en tan solo 972 pasos, casi una tercera parte de la máxima cantidad definida (3000).

4.- Análisis de resultados

De acuerdo a lo observado en las estadísticas recolectadas, el tiempo de limpieza de las roombas es directamente afectado por factores como el número de obstáculos y basura, pues son obligados a recorrer el ambiente con más precisión para encontrarlos, con el fin de que sea una búsqueda eficiente. Además, mientras mayor sea la cantidad de obstáculos en el ambiente, mayor será la probabilidad de que se queden sin batería, sobre todo cuando se tienen grandes cantidades de celdas. Esto es debido a que la distancia que cada roomba calcula es una estimación basada en la distancia de Chebyshev.

En cuanto a la simulación 1, los resultados demostraron ser consistentes a pesar de tener 40% de obstáculos y de objetos en el ambiente. En este sentido, el promedio de pasos en los que la roomba fue capaz de limpiar todas las celdas fue 1400-2000 para ambientes de 28x28 celdas. De la misma forma, el número de movimientos de la roomba se mantuvo entre 1000 y 1300, lo que refleja menos pasos que el número total de estos, correspondientes al recorrido necesario para que la roomba regrese a la estación de carga o vuelva a encontrar basura que recoger.

Para ambientes controlados, las roombas fueron capaces de limpiar el 100% de la basura en menos de 3000 pasos, sin llegar a quedarse sin batería. Sin embargo, al momento de introducir ambientes de 50x50 celdas, una sola roomba no tuvo la suficiente velocidad para completar la limpieza en el tiempo requerido; a pesar de esto, se mantuvo constante en su movimiento y gestión de batería.

Para la simulación 2, se realizaron simulaciones con 5 y 10 roombas capaces de colaborar para limpiar la basura del ambiente en el menor tiempo posible. Debido a la capacidad de las roombas de compartir información, se observó una drástica reducción en la

cantidad de pasos necesaria para recolectar toda la basura, oscilando entre 300-400 para ambientes de 28x28 celdas.

En este caso, el factor que parece afectar de mayor forma la velocidad de recolección de basura es el número de roombas y cantidad de obstáculos. Mientras que 5 roombas fueron capaces de limpiar todas las celdas en aproximadamente, llegaron a presentar bajas al momento de enfrentarse a ambientes más grandes, donde cada roomba debe recorrer una mayor cantidad de celdas, y por lo tanto, alejarse más de su estación de carga.

Al momento de estimar la distancia de Chebyshev, a pesar de contar con un margen de 20 pasos, no fueron suficientes para evitar que al menos una roomba se quedara sin batería. Este problema se puede solucionar aumentando la cantidad de roombas, reduciendo la la cantidad de movimientos de cada una y evitando que la estimación de distancia de Chebyshev falle al no contemplar obstáculos.

Otro de los principales desafíos al que se enfrentaron las roombas fue el de competir por las estaciones de carga. Esto fue solucionado mediante la implementación del estado “wait”, donde las roombas calculan la estación más cercana conocida y, en el caso de que esté ocupada, esperan a que esté disponible. Esto demostró ser especialmente eficiente, pues a pesar de que una roomba pueda llegar a pasar aproximadamente 20 pasos sin moverse (esperando su estación de carga), evita el riesgo de que se quede sin batería una vez llega a su estación de carga. Como fue observado en los datos recopilados, mientras más roombas haya, se puede llegar a recolectar basura en menor tiempo.

Cabe destacar que el estado de “wait” no se considera una acción que reduce la batería en cada paso, sino que se trata de una condición en la que las roombas simplemente se quedan “apagadas” hasta que su estación de carga esté disponible.

5.- Conclusiones

Realizar esta actividad fue útil para comprender la forma en la que se pueden definir los comportamientos de los agentes para cumplir con sus objetivos de manera eficiente, a través de máquinas de estados. Además, se demostró que la implementación de algoritmos de búsqueda como A* permite obtener caminos con gran precisión para guiar a los agentes en el ambiente. De la misma manera, se exploraron cálculos alternativos que sirvieron para optimizar la complejidad del código, como fue implementando la distancia de Chebyshev para calcular la distancia entre una roomba y su estación de carga, siendo un cálculo considerablemente más rápido que calcular A* en cada paso.

Por otro lado, los resultados obtenidos demostraron que las capacidades de cooperación mejoraron significativamente la eficiencia. La simulación 2 demostró que cuando las roombas cooperan y comparten información, el proceso de limpieza se vuelve más rápido. Esto fue evidenciado cuando 5 roombas fueron capaces de limpiar en su totalidad un ambiente de 50x50 celdas con 90% de basura y 40% de obstáculos, misma tarea que una roomba no fue capaz de completar en menos del tiempo máximo establecido de ejecución (3000 pasos).

La optimización y detallada consideración de la toma de decisiones de las roombas fue crucial, sobre todo al momento de interactuar con estaciones de carga. En este sentido se observó que hacer que las roombas esperen algunos pasos extra para poder cargarse aumenta las probabilidades de limpiar todas las celdas en menor tiempo, pues si se llegan a quedar sin batería, retrasarían la velocidad total de limpieza.

6.- Referencias

Kamble, V. H., & Dale, M. P., (2022). Machine learning approach for longitudinal face recognition of children. *Machine Learning for Biometrics*. ScienceDirect.
<https://www.sciencedirect.com/topics/computer-science/chebyshev-distance>