Tecnológico de Monterrey

Evidence 1

Cesar Moran Macias | A01645209

Ana Paola Jiménez Sedano | A0164453

Ariana Guadalupe Rosales Villalobos | A01644773

Demmí Elizabeth Zepeda Rubio | A01709619

Luisa Merlo García | A01067715

Modeling of Multi-Agent Systems with Computer Graphics

Professor Iván Axel Dounce Nava

August 20th, 2025

Description of the Problem

The following problem focuses on the simulation of a Multi-Agent System applied to the organization of a jewelry display case; in this scenario the user assumes the role of the jewelry store's owner, and since the employee of the last shift left in disarray the jewelry in display, it is up to the owner's three micro robots to organize the jewelry into something more professional, according to the jewelry's color and with the following constraints:

- Each robot is associated with one color, therefore it can only collect jewels of its color.
- The robots start in random empty positions and operate under a limit of time (Tmax).
- The total time and total movements it took all robots to collect all jewels must be recorded.
- The environment is modeled with the size of $U \times V$, with K number of jewels.

The robots possess omnidirectional wheels, manipulators to pick up jewelry, and sensors that allow them to distinguish between free spaces, walls, and other robots. They can also know if they are carrying a jewel.

Description of the Solution

To solve this problem, the solution was divided into two main stages: computational modeling and simulation implementation.

The first stage consisted in modeling the components in Blender, such as the robots, and the jewels. Each robot was assigned a different color to identify them and the jewels they can

collect. The environment was designed as a grid base model, this is so that it allows systematic checking of adjacent cells for walls, jewels, empty space, or other robots.

The second stage consisted of importing the assets into Unity to implement and program the agent's behaviour and interactions. Each robot follows a Finite State machine (FSM) structure following these states: searching, picking up, carrying, delivering and waiting. This simulation looks to integrate:

- Random initialization of jewels and robots positions.
- Sensors that allow robots to detect jewels, walls or other robots.
- Determination of priority rules to handle collisions, and to know in which order the robots would advance or wait.
- Performance recording to log the total execution time until all jewels are organized, and the number of moves that each robot made.

This approach to the system allows for visual representation, simulation, and measuring of results, so that the system can be tested for efficiency and be improved by making result based decisions.

Design Decisions and their Justification

- Grid-based logical (UxV)
    - Even though the display case is represented in a continuous 3D environment, a logical grid was defined to simplify the movement and sensing of robots
    - This type of setup allows discrete checks of adjacency, such as walls, other robots, jewels. This looks to ensure computational efficiency and easier debugging
- Color based specialization

- Each robot is restricted to only collect jewels of its own color
- This way, the design guarantees division of labor and avoids conflicts of jewels ownership, reducing complexity.

Class Diagrams and Agents Protocol

The following diagrams are meant to facilitate the comprehension of the agents' behavior in the solution presented for the problem-situation mentioned above. This is the link for better visualization: Lucidchart.

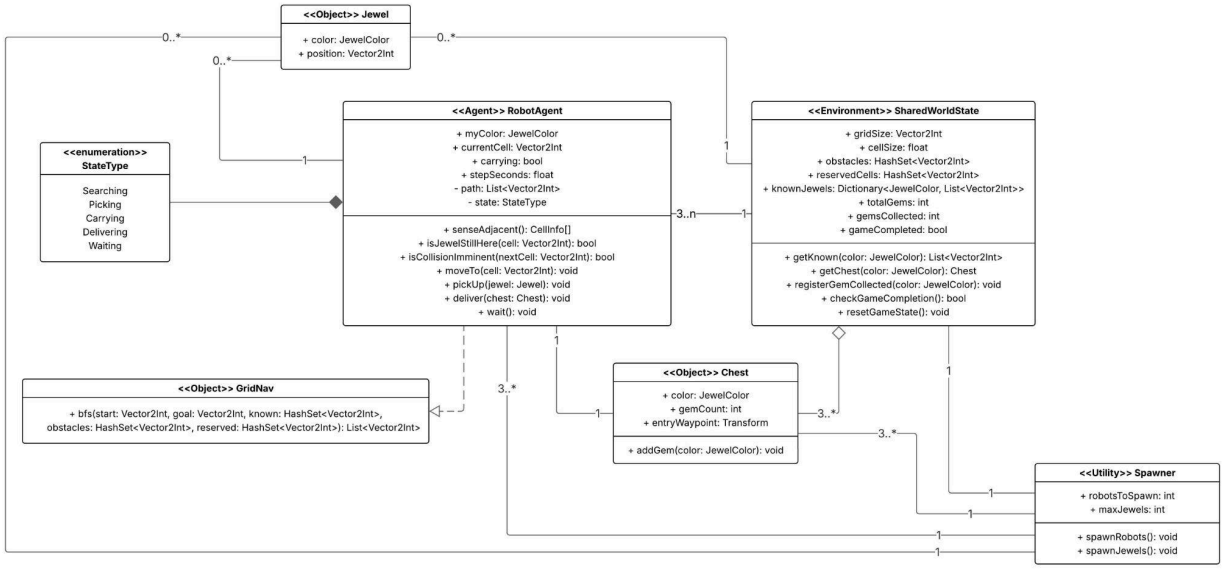


Figure 1. UML Class Diagram.

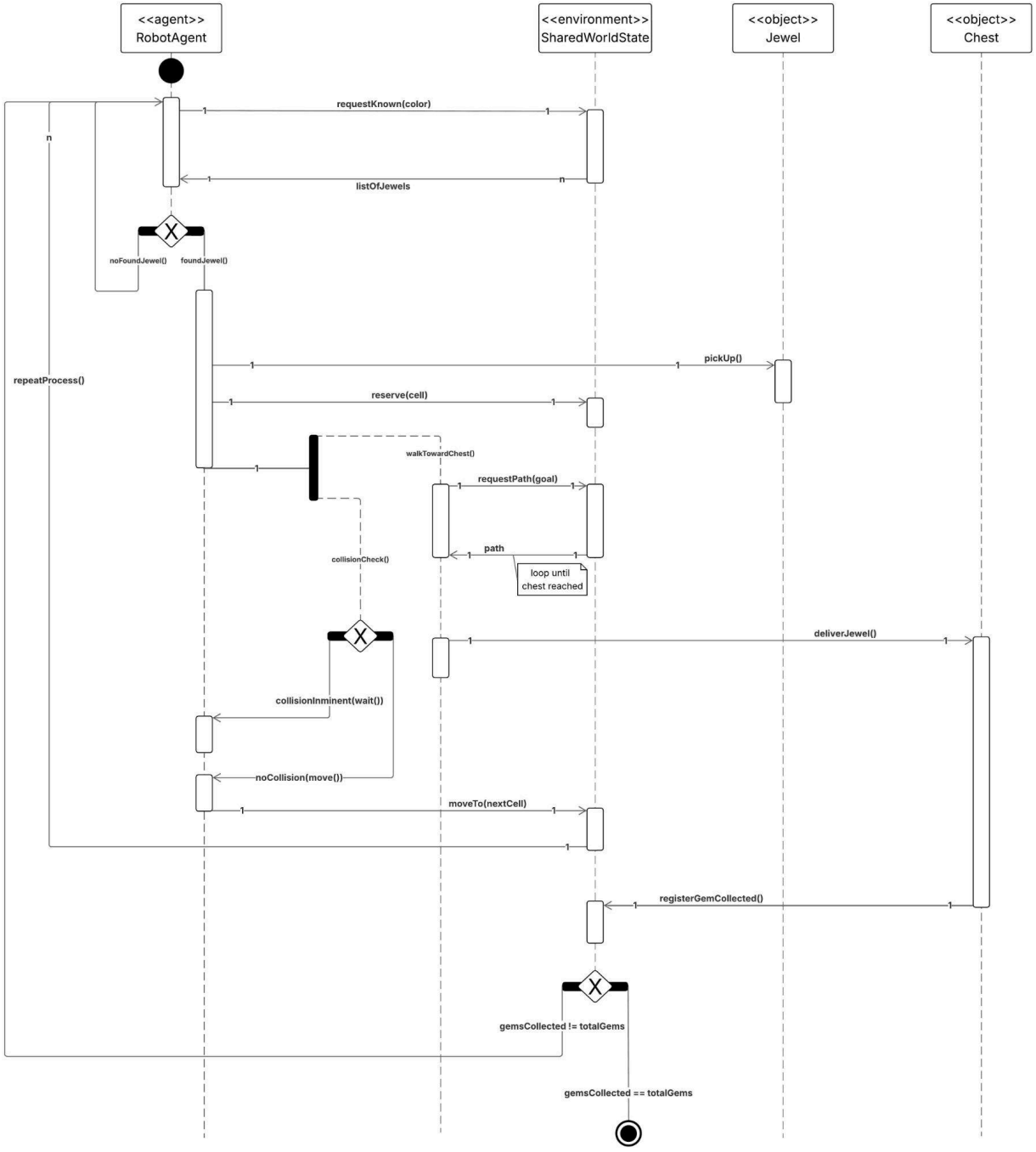
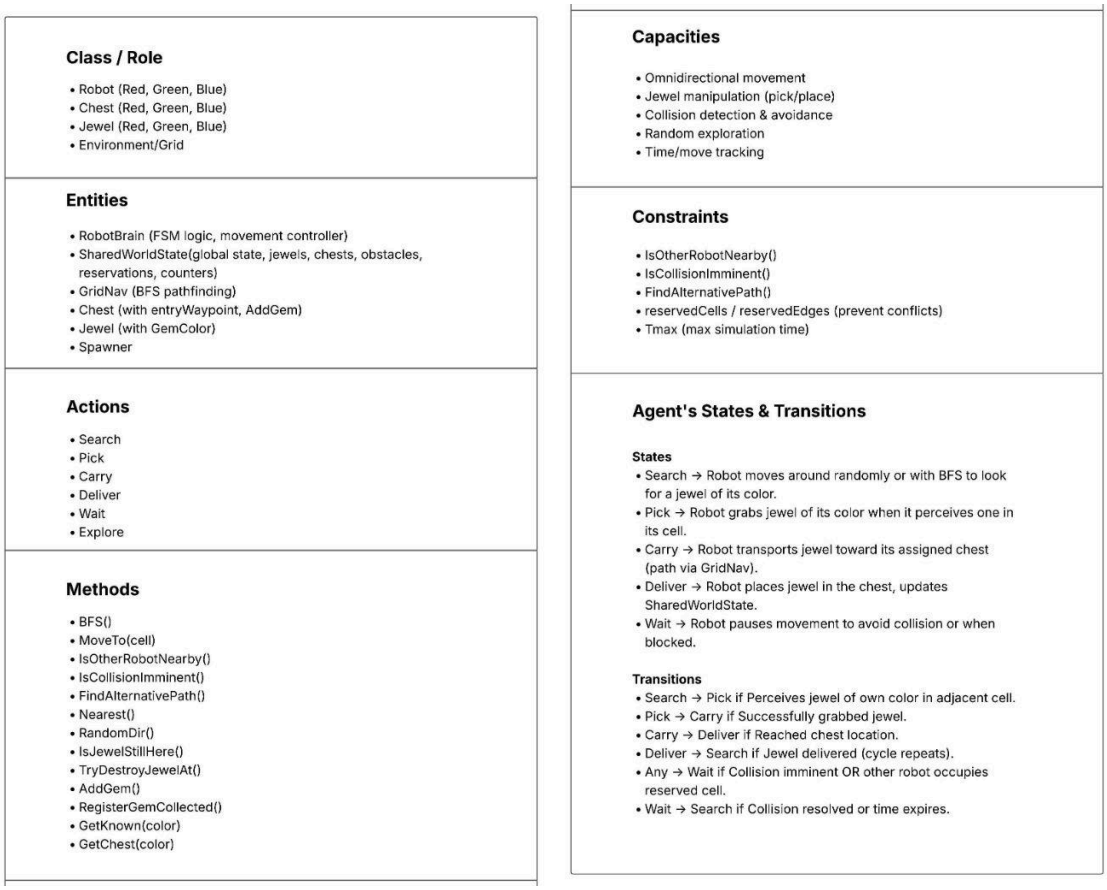


Figure 2. UML Agent Protocol Diagram.



Figures 3, 4. Stack Agent Protocols Diagram.

Results and their Analysis

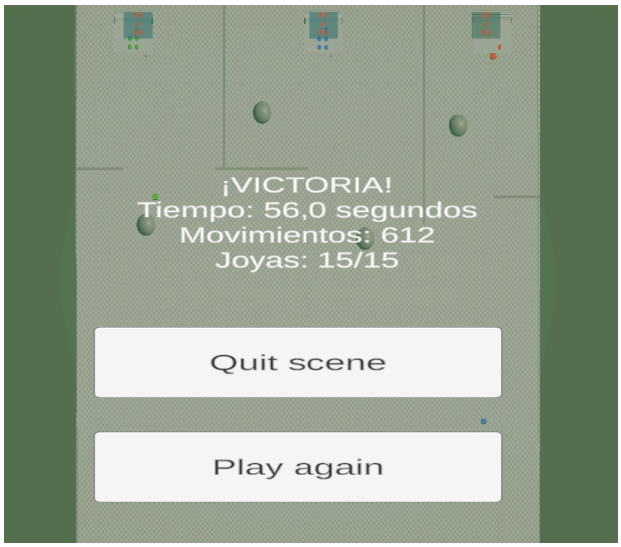
The simulation was executed in Unity, with three robot agents (Red, Green, and Blue), each one of them constrained to collect only jewels of its assigned color. The environment was initialized randomly for both robot positions and jewel distribution, under the global time limit of $Tmax$.

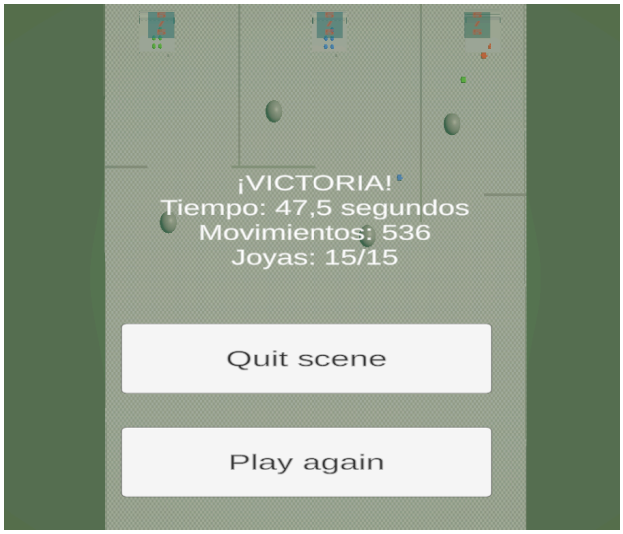
During the multiple experiments, the following outcomes were:

- Task Completion: All jewels were successfully collected and delivered to their respective chests in multiple runs.

- Performance Metrics. The system recorded total execution time and number of movements per robot, allowing for direct comparison across runs.
- Collision Management: The constraint layer prevented deadlocks and collisions, which brought a smooth multi-agent coordination.
- Strategy Efficiency:
    - Robots relied on a Finite State Machine strategy: *Search* → *Pick* → *Carry* → *Deliver* → *Search*, with *Wait* interruptions in case of conflict.
    - The grid-based model allowed efficient pathfinding and reduced computational complexity in movement decisions.
    - The priority rules for collisions (yielding and waiting when paths intersected) reduced wasted time compared to purely random avoidance.
    - Specializing robots by color prevented conflicts over jewel ownership, and accelerated task distribution and parallelization.

Overall, the strategy used allowed the agents to cooperate through the rules and environment, which made them complete the organization task in the shortest time possible given their constraints.

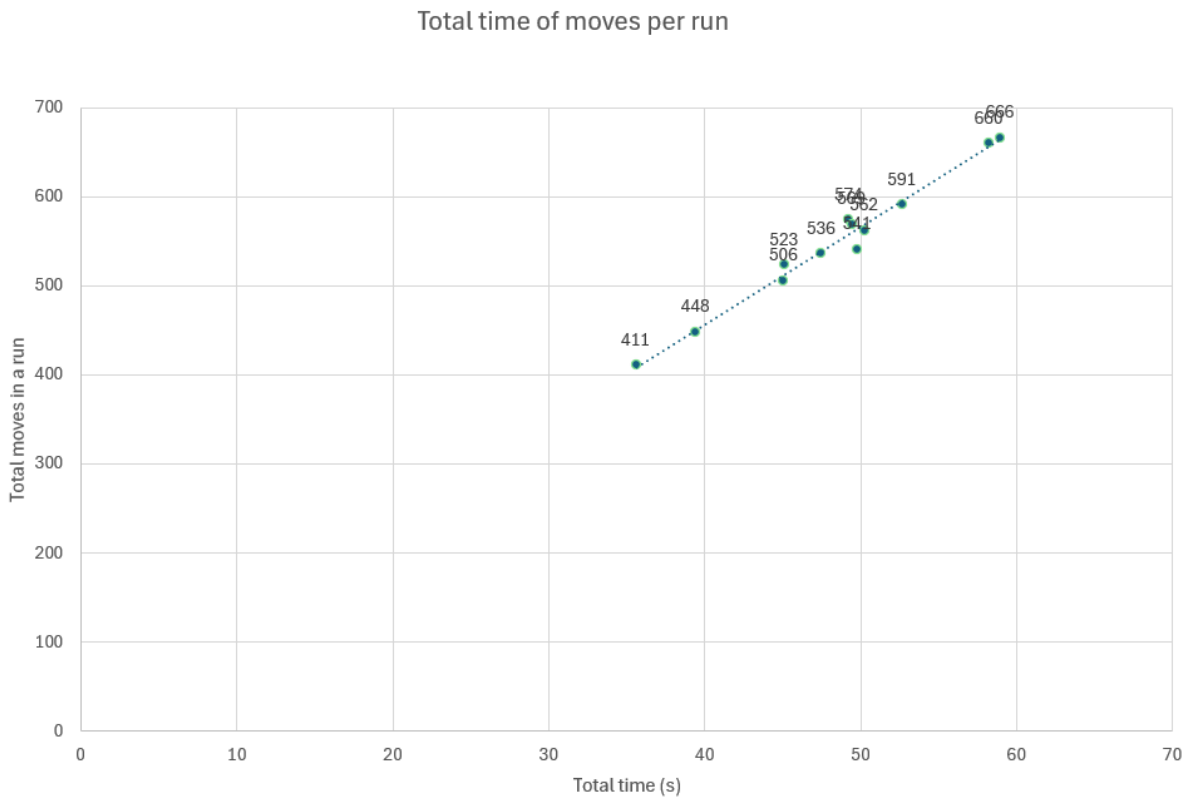

¡VICTORIA!
Tiempo: 56,0 segundos
Movimientos: 612
Joyas: 15/15
Quit scene
Play again


¡VICTORIA!
Tiempo: 47,5 segundos
Movimientos: 536
Joyas: 15/15
Quit scene
Play again

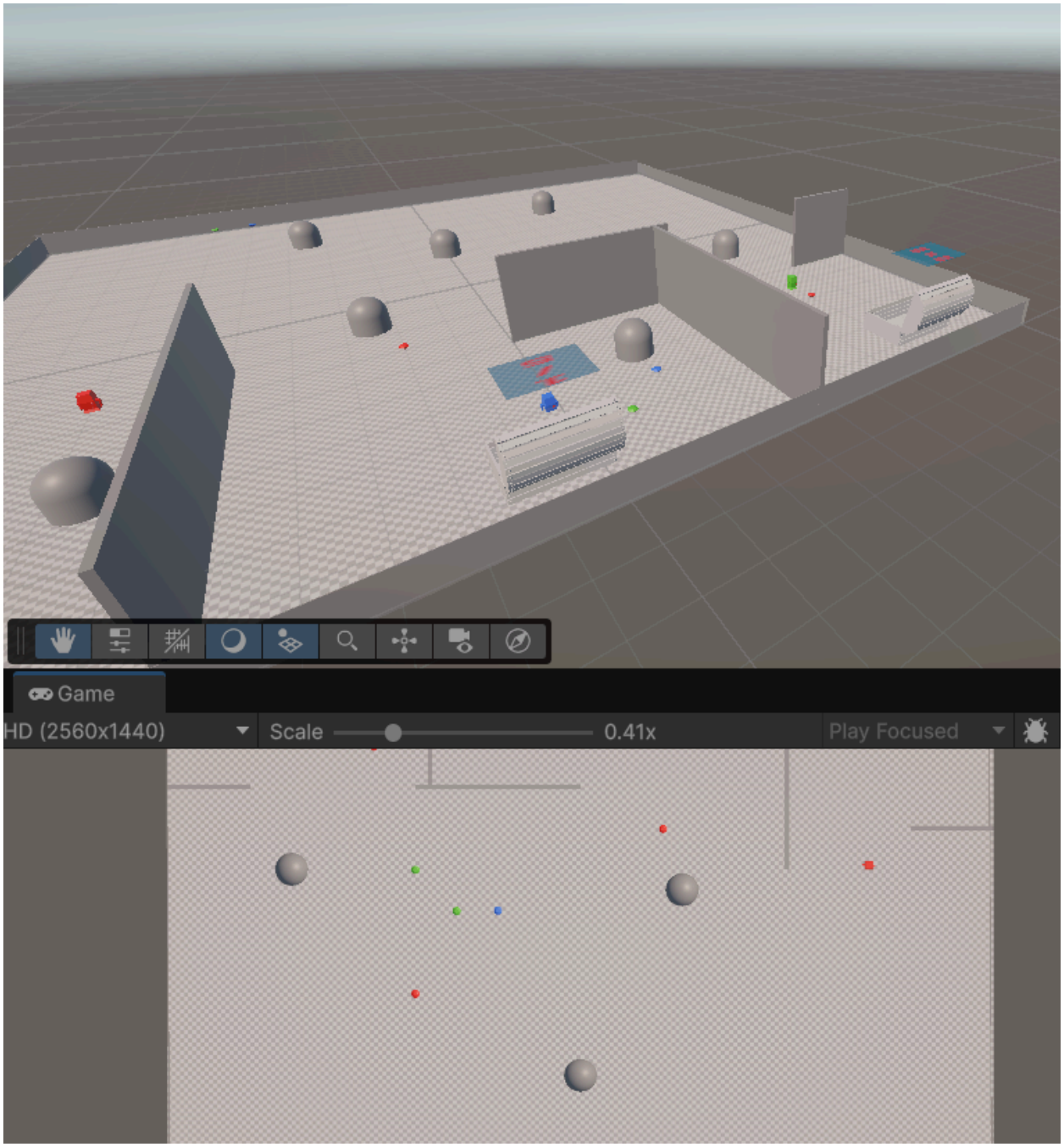

¡VICTORIA!
Tiempo: 51,7 segundos
Movimientos: 540
Joyas: 15/15
Quit scene
Play again

Graph of dispersion

The following graph shows the relationship between total execution time and the number of movements performed in each run. A positive linear trend can be observed: the longer the time, the greater the number of movements, indicating that the agents maintain a constant pace of action proportional to the duration of the simulation.



date_time	total_time_	total_move	gems_colle	total_gems	grid_U	grid_V	cellSize	maxSeconds
31/08/2025 21:38	47.468	536	15	15	20	20	10	120
31/08/2025 22:00	45.056	506	15	15	20	20	10	120
31/08/2025 22:04	49.787	541	15	15	20	20	10	120
31/08/2025 22:06	49.236	574	15	15	20	20	10	120
31/08/2025 22:07	35.579	411	15	15	20	20	10	120
31/08/2025 22:08	58.948	666	15	15	20	20	10	120
31/08/2025 22:09	52.65	591	15	15	20	20	10	120
31/08/2025 22:09	49.417	569	15	15	20	20	10	120
31/08/2025 22:10	58.233	660	15	15	20	20	10	120
31/08/2025 22:11	39.418	448	15	15	20	20	10	120
31/08/2025 22:12	50.219	562	15	15	20	20	10	120
31/08/2025 22:13	45.089	523	15	15	20	20	10	120



Video functioning:

joyasBots - SampleScene - Windows, Mac, Linux - Unity 6 (6000.0.37f1) _DX11_ 2025-08-31 21-50-13.mp4

Conclusions

The proposed solution demonstrates that a multi-agent system with simple but coordinated rules can achieve effective organization. We believe our strategy allowed a very decent performance of our agents, however, we also believe there is room for improvement:

- Adaptive pathfinding: Currently, our robots use BFS with fixed rules, and we believe that by incorporating something like adaptive or heuristic-based pathfinding could reduce the time it takes for the robots to move.
- Communication between agents: Coordination is indirect through reservations in the environment, but explicit inter-agent communication could minimize more waiting.
- Load balancing: If the jewel distribution is uneven, then some robots finish earlier than others, and maybe we could incorporate temporary role assignment to improve this, though it isn't of the highest priority.
- Learning mechanisms: Adding reinforcement learning or experience-based strategies would allow the robots to optimize their movement patterns across multiple runs.

In conclusion, while our strategy fulfills the problem requirements and highlights great effectiveness of MAS for task organization, there can be future enhancements in what we have already mentioned.