



Tecnológico
de Monterrey

Modelación de Sistemas Multiagentes con Gráficas Computacionales

Grupo 601

Gustavo García Téllez | A01644060

Ayetza Y Infante Garcia | A01709011

Fernanda Ríos Juárez | A01656047

Álvaro Solano González | A01643948

Sebastián Borjas Lizardi | A01748052

Lesly Citlaly Gallegos Acosta | A01563036

Evidencia Final

Iván Axel Dounce Nava | Sistemas Multiagentes

Mauricio Bezares Peñúñuri | Gráficas Computacionales

9 de marzo del 2024

Conformación del equipo

Gustavo Garcia Tellez:

Fortalezas	Área de Oportunidad	Expectativas
- Programación - Algoritmos - Modelado 3D	- Front End - Diseño	Espero divertirme creando un proyecto que me divierta, conociendo nuevas tecnologías y nueva gente

Ayetza Yunnuen Infante García

Fortalezas	Área de Oportunidad	Expectativas
- Análisis - Backend y Frontend - Creatividad	- Modelado 3D - Texturizado - Animación en Unity	Desarrollarme mucho mas profesionalmente, mejorar mis habilidades en Unity y seguir aprendiendo.

Fernanda Ríos Juárez

Fortalezas	Área de Oportunidad	Expectativas
- Diseño - Backend y Frontend - Creatividad - Lógica	- Modelado 3D - Texturizado - Animación en Unity	Crecer profesionalmente y seguir aprendiendo para así, poder mejorar mis habilidades en el desarrollo de tecnología.

Sebastián Borjas

Fortalezas	Área de Oportunidad	Expectativas
- Diseño UI - Backend y Frontend - PM - Conflict resolution	- Modelado 3D - Animación en Unity - Lógica teórica sobre Agentes	Poder construir un buen proyecto con mi equipo de trabajo y crecer profesionalmente durante el proceso.

Álvaro Solano

Fortalezas	Área de Oportunidad	Expectativas
- Lógica - Creatividad - Algoritmos - Unity -	- Modelado 3D - Diseño	seguir desarrollando nuevas habilidades y fortalecerse aún más como ingeniero.

Lesly Gallegos

Fortalezas	Área de Oportunidad	Expectativas
<ul style="list-style-type: none"> - Lógica - Creatividad - Unity 	<ul style="list-style-type: none"> - Modelado 3D - Implementación 	Seguir ampliando mis conocimientos y crecer profesionalmente.

Esperanzas y Compromisos:

Esperamos desarrollar un sistema de agentes inteligentes en Unity, optimizando su interacción y toma de decisiones. Nos comprometemos a seguir buenas prácticas de programación, colaborar eficientemente y cumplir con los plazos establecidos.

Creación de herramientas de trabajo colaborativo

Link al Github:

<https://github.com/ayetzainfante/da-team.git>

Método de Comunicación:

- Whatsapp
- Google Teams o Zoom

Propuesta Formal

Descripción del Reto

Contexto

La movilidad urbana es un factor clave en el desarrollo económico, social y en la calidad de vida de las personas. Durante décadas, el uso del automóvil ha sido considerado un símbolo de progreso, pero en la actualidad, el crecimiento descontrolado de su uso ha generado un impacto negativo en diversos aspectos, incluyendo la contaminación ambiental, la congestión vehicular, el aumento de accidentes y problemas de salud relacionados con la contaminación del aire.

En México, el número de kilómetros recorridos por automóvil se ha triplicado en las últimas décadas, lo que ha generado una crisis en la movilidad urbana; esta situación afecta la productividad, incrementa los costos de transporte y reduce la calidad de vida de los habitantes.

Para que México pueda posicionarse entre las economías más grandes del mundo, es de suma importancia mejorar la movilidad en sus ciudades a través de soluciones innovadoras que optimicen el tráfico y reduzcan la congestión vehicular.

Objetivo del Reto

El reto consiste en desarrollar una solución innovadora que contribuya a mejorar la movilidad urbana en México mediante la reducción de la congestión vehicular; para ello, se deberá implementar un modelo de simulación gráfica basado en un sistema multiagente, que represente el tráfico urbano y permita evaluar estrategias para optimizar el flujo vehicular.

La solución debe enfocarse en aplicar estrategias inteligentes que minimicen el tiempo de tránsito, reduzcan la emisión de contaminantes y optimicen los recursos de infraestructura vial.

Identificación de los Agentes

Vehículos (motos y autos), dron, oficial

Definición de PEAS:

Performance

El desempeño de los agentes se medirá con base en los siguientes criterios:

1. **Colisiones evitadas:** Número de colisiones resueltas por el policía/dron
2. **Congestiones resueltas:** Celdas congestionadas eliminadas (con 3+ vehículos)
3. **Tiempo de respuesta del policía:** Cantidad de pasos necesarios para resolver el problema
4. **Multas emitidas:** Vehículos sancionados por exceso de velocidad.
5. **Caso de éxito:** La simulación se completa cuando el policía y el dron logran eliminar todas las colisiones y excesos de velocidad, manteniendo un flujo de tráfico estable durante el tiempo establecido
6. **Finalización:** La simulación termina si >70% de vehículos están bloqueados por 3 minutos.

Environment

El ambiente está compuesto por los siguientes elementos relevantes:

- **Policía:** Emite órdenes para controlar el tráfico y dirigir a los agentes, además decide mediante el aprendizaje Q-Learning si solicita ayuda al dron o no.
- **Otros agentes (vehículos):** Son agentes autónomos con aprendizaje Q-Learning para definir sus acciones.
- **Grid 10x10:** Es el espacio discreto donde los agentes se mueven y que representa la avenida
- **Dron:** interviene cuando es llamado por el agente policía.

Actuators

Los agentes contarán con los siguientes actuadores para ejecutar sus acciones:

- **Policía:**
 - resolve_myself: Reduce velocidad en colisiones/congestiones.
 - call_drone: Sigue a la ayuda al dron.
 - issue_ticket: Multa a vehículos con exceso de velocidad.
- **Vehículos:**

- accelerate(), decelerate(), move(): Movimiento estocástico.
- obey_instructions(): Decide acciones con Q-Learning.
- **Dron:**
 - resolve_collisions_and_congestion(): Reduce velocidad de vehículos problemáticos.

Sensores

Para obtener información del entorno, los agentes dispondrán de los siguientes sensores:

- **Policia:**
 - get_state(): Detecta colisiones y congestiones a través del modelo.
- **Vehículos:**
 - get_state(): Velocidad, proximidad al policía, congestión en su celda.
- **Dron:**
 - Acceso a model.cars y model.motorcycles para detectar problemas.

Plan de Trabajo

Nuestro plan de trabajo sigue la metodología SCRUM, el cual es un enfoque ágil que permite organizar y desarrollar el proyecto de manera iterativa e incremental. Se divide en sprints, que son períodos cortos de tiempo en los que el equipo se enfoca en completar tareas específicas para alcanzar los objetivos del proyecto.

El propósito de este plan es estructurar el desarrollo del sistema basado en agentes inteligentes para mejorar la movilidad urbana, asegurando una entrega constante de avances funcionales y fomentando la colaboración entre los miembros del equipo.

Estructura del Plan de Trabajo

El trabajo se organiza en 4 sprints, cada uno con actividades clave, responsables definidos y entregables específicos. Además, se incluyen reuniones diarias (Daily Scrum) para mantener una comunicación fluida y detectar obstáculos a tiempo.

Cada sprint sigue el siguiente flujo de trabajo:

- Sprint Planning (Planificación del Sprint): Se seleccionan las tareas del backlog y se definen prioridades.
- Ejecución del Sprint: Desarrollo y prueba de funcionalidades asignadas.
- Daily Scrum (Reuniones diarias): Breves reuniones para revisar avances y bloqueos.
- Sprint Review (Revisión del Sprint): Presentación de avances y feedback.
- Sprint Retrospective (Retrospectiva): Evaluación de lo que se puede mejorar para el siguiente sprint.

Roles y Responsabilidades

Cada miembro del equipo tiene un rol específico para garantizar un trabajo estructurado y eficiente.

Descripción de Situación a Modelar

¿Qué está pasando?

En un cruce de tráfico en el que los coches intentarán cumplir su ruta tomando en cuenta las instrucciones que les da el policía. Los coches deben tener un propósito, tal como: avanzar en su ruta y evitar cometer infracciones viales.

¿Qué agentes toman parte en esta situación?

Hay diversos tipos de agentes, los cuales representarán los diferentes tipos de vehículos en nuestro modelo, un dron de ayuda y el usuario. Cada agente deberá tener un funcionamiento diferente dependiendo de sus características como vehículo y el objetivo del mismo. Por su parte, el policía deberá ser capaz de comunicar leyes de tránsito o sancionar a quienes las infrinjan, además de requerir ayuda al dron de ser necesario. Los diferentes agentes serán los siguientes: vehículo (moto y auto), dron y oficial.

¿Cuál es el contexto previo?

El modelo va a considerar un cruce de tráfico en el que no hay semáforos, por lo que es necesario que un policía sea el que le da las direcciones a los autos de cuando pueden avanzar y pueda prevenir los accidentes viales.

¿Qué esperan lograr los agentes?

Los agentes deben cumplir uno de dos posibles objetivos, o cumplir con su simple ruta establecida o intentar no provocar accidentes, en el caso de los vehículos. Por su parte, en el caso del policía, dirigir correctamente el flujo de tránsito o sancionar a quienes infrinjan las reglas de flujo vial. Y en el caso del dron, se espera que este apoye al policía en la gestión del tráfico. Además de estos objetivos se espera que los agentes respeten las indicaciones que les da el policía y que actúan lógicamente de acuerdo a los coches que tienen enfrente.

Descripción de Agentes

Agente policía

Rol

- Decide entre resolver colisiones/congestiones o llamar al dron usando Q-Learning.
- Emite multas a vehículos con exceso de velocidad.

Tipo de razonamiento

- Razonamiento práctico: toma decisiones basadas en reglas y objetivos

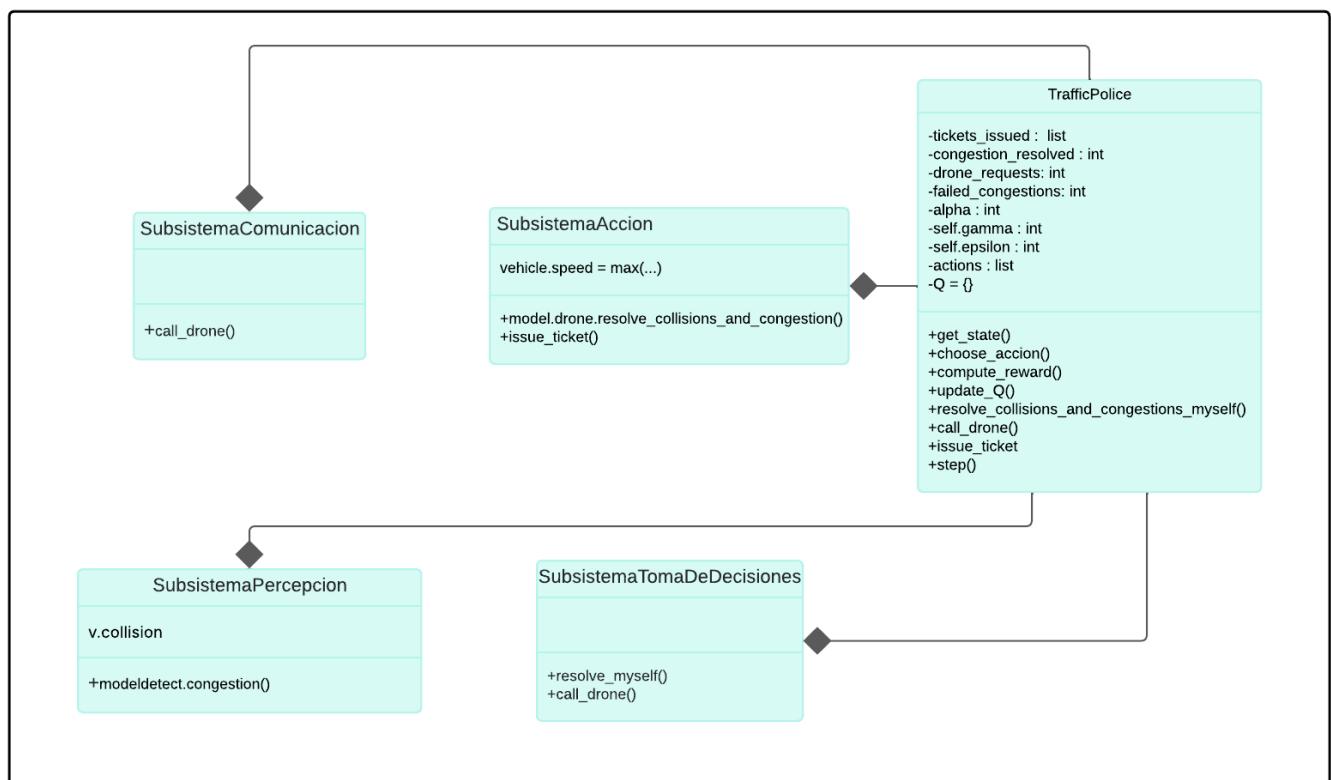
Arquitectura

- Vertical: basada en la toma de decisiones progresiva, desde la percepción hasta la acción

Principales subsistemas

(e.g. percepción, acción, bbf, creencias, etc., según sea necesario)

- Percepción: Detecta colisiones activas, la velocidad de los vehículos e identifica celdas congestionadas
- Acción: Llama al dron, resuelve conflictos y emite multas
- Toma de decisiones: Selecciona entre resolver las problemáticas por su propia cuenta o llamar al dron
- Comunicación: Pide ayuda al dron



Sensores y Actuadores

- Sensores:
 - Colisiones activas
 - Celdas congestionadas
- Actuadores:
 - Órdenes
 - Multas
 - Pedir ayuda

Agente Vehicle (Moto y Auto)

Rol

- Representa la clase base para todos los vehículos en la simulación (autos y motos).
- Define comportamientos comunes como movimiento, detección de colisiones y ajuste de velocidad.
 - Autos: Obedecen órdenes con 90% de probabilidad (velocidad límite=5).
 - Motos: Obedecen con 50% de probabilidad (velocidad límite=7).

Tipo de razonamiento

- Razonamiento práctico: Basado en Q-Learning, toma decisiones para maximizar recompensas

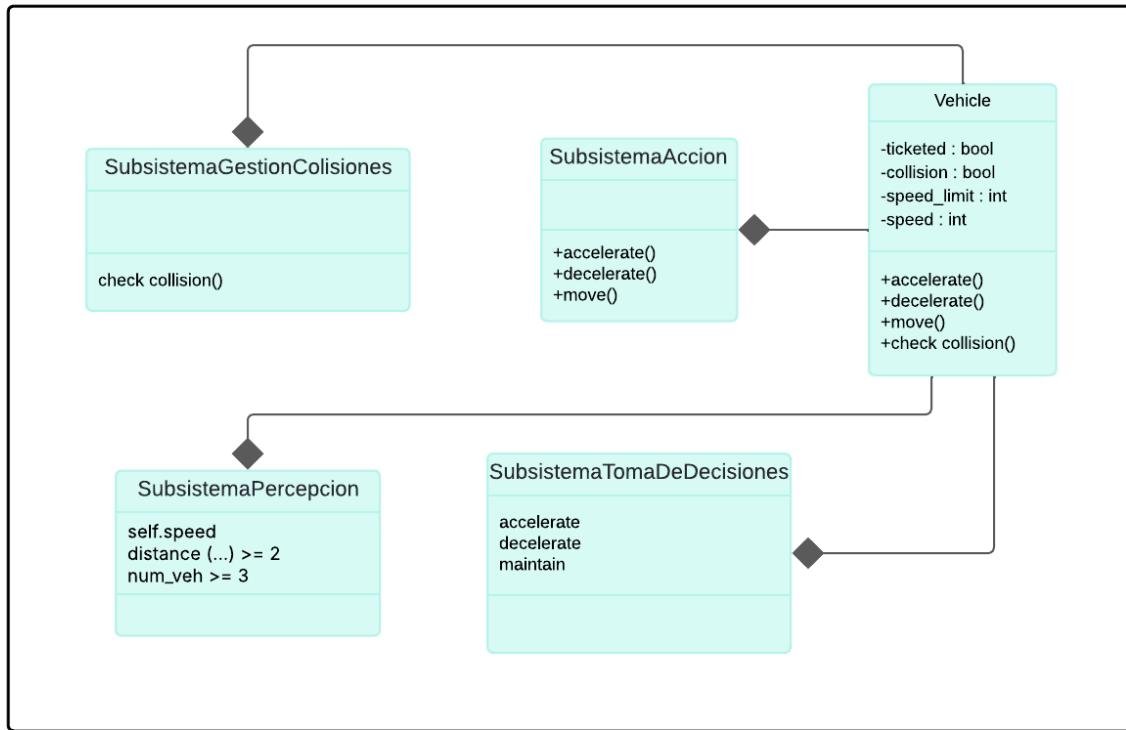
Arquitectura

- Vertical: basada en la toma de decisiones progresiva, desde la percepción hasta la acción

Principales subsistemas

(e.g. percepción, acción, brf, creencias, etc., según sea necesario)

- Percepción: Detecta su velocidad actual, la proximidad al policía y la congestión en su celda
- Toma de decisiones: Selecciona alguna acción (acelerar, desacelerar, mantener velocidad)
- Acción: Movilización en función de su ruta, ajusta su velocidad
- Gestión de colisiones: marca las colisiones



Sensores y Actuadores

- Sensores:
 - Velocidad actual
 - Proximidad al policía
 - Congestión local
- Actuadores:
 - Aceleración
 - Desaceleración
 - Mantenimiento de velocidad

Agente Dron

Rol

- Interviene cuando el policía lo solicita para resolver colisiones/congestiones
- Actúa directamente sobre vehículos

Tipo de razonamiento

- Reactivo: Responde a las solicitudes del policía para resolver problemas específicos (colisiones y congestiones).

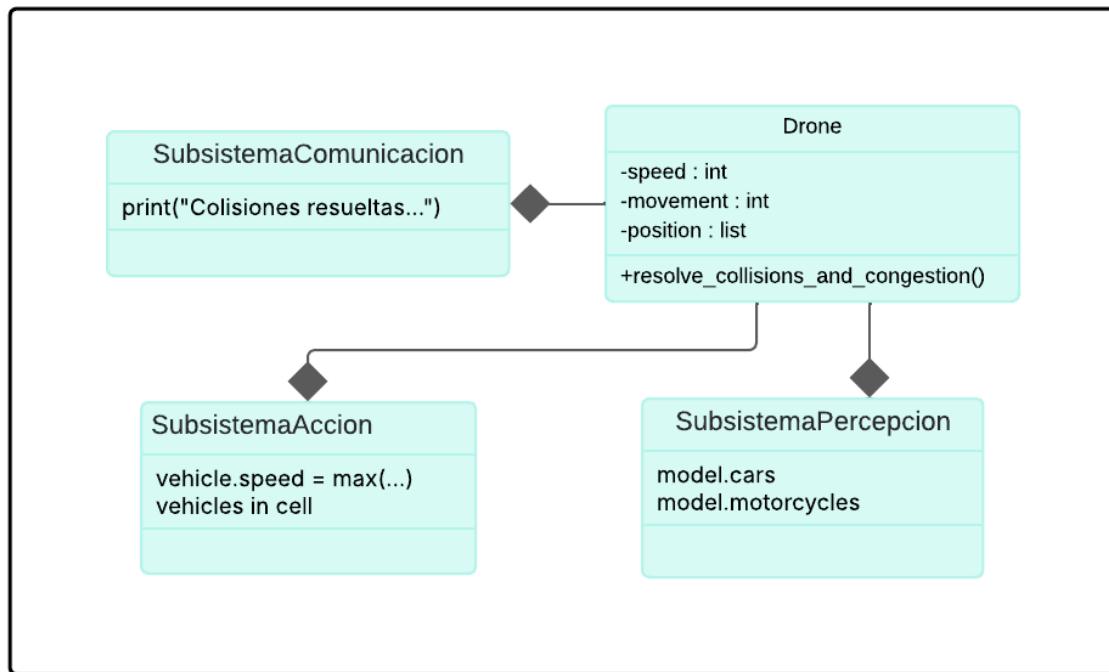
Arquitectura

- Horizontal: Ejecuta reglas en paralelo (resolver colisiones y congestiones).

Principales subsistemas

(e.g. percepción, acción, brf, creencias, etc., según sea necesario)

- Percepción: Acceso a vehículos problemáticos
- Acción: Resuelve colisiones y reduce la velocidad en celdas congestionadas
- Comunicación: Notifica al policía tras intervenir



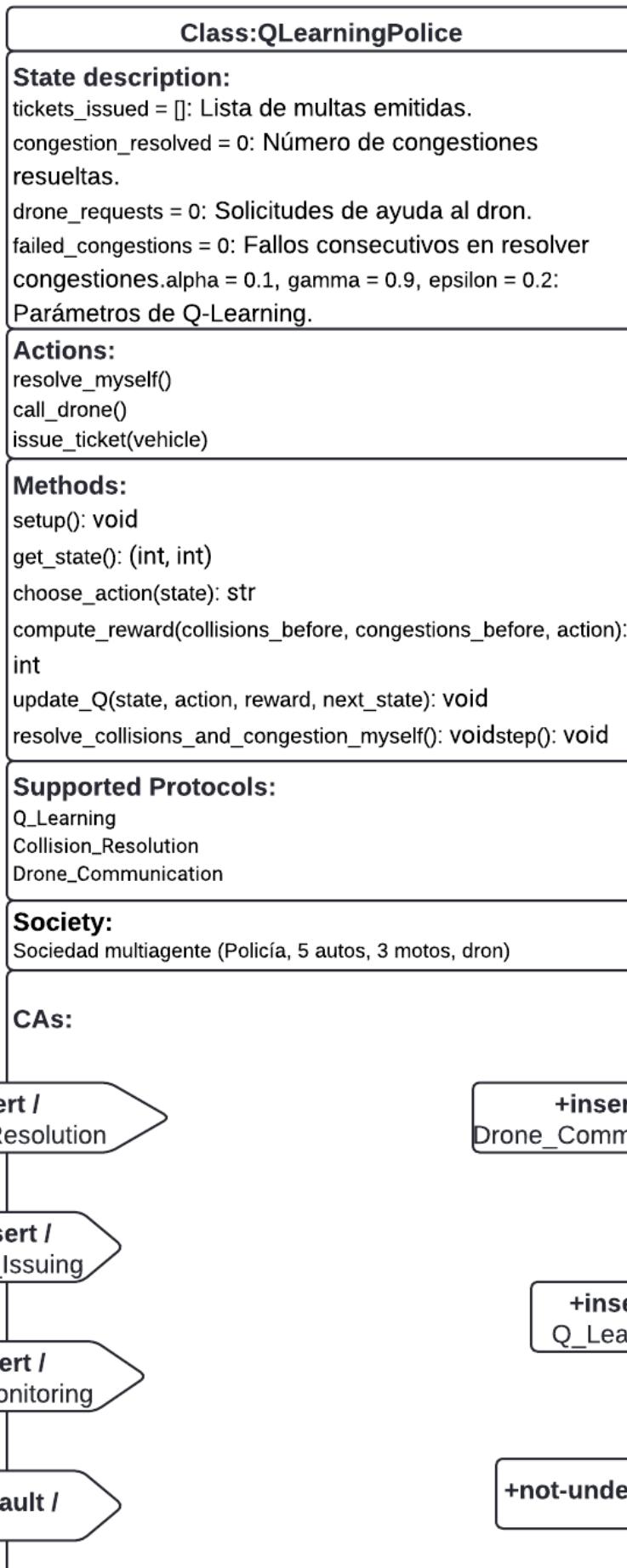
Sensores y Actuadores

- Sensores:
 - Solicitudes del policía
 - Acceso al modelo (vehículos y sus estados)
- Actuadores:
 - Intervención directa

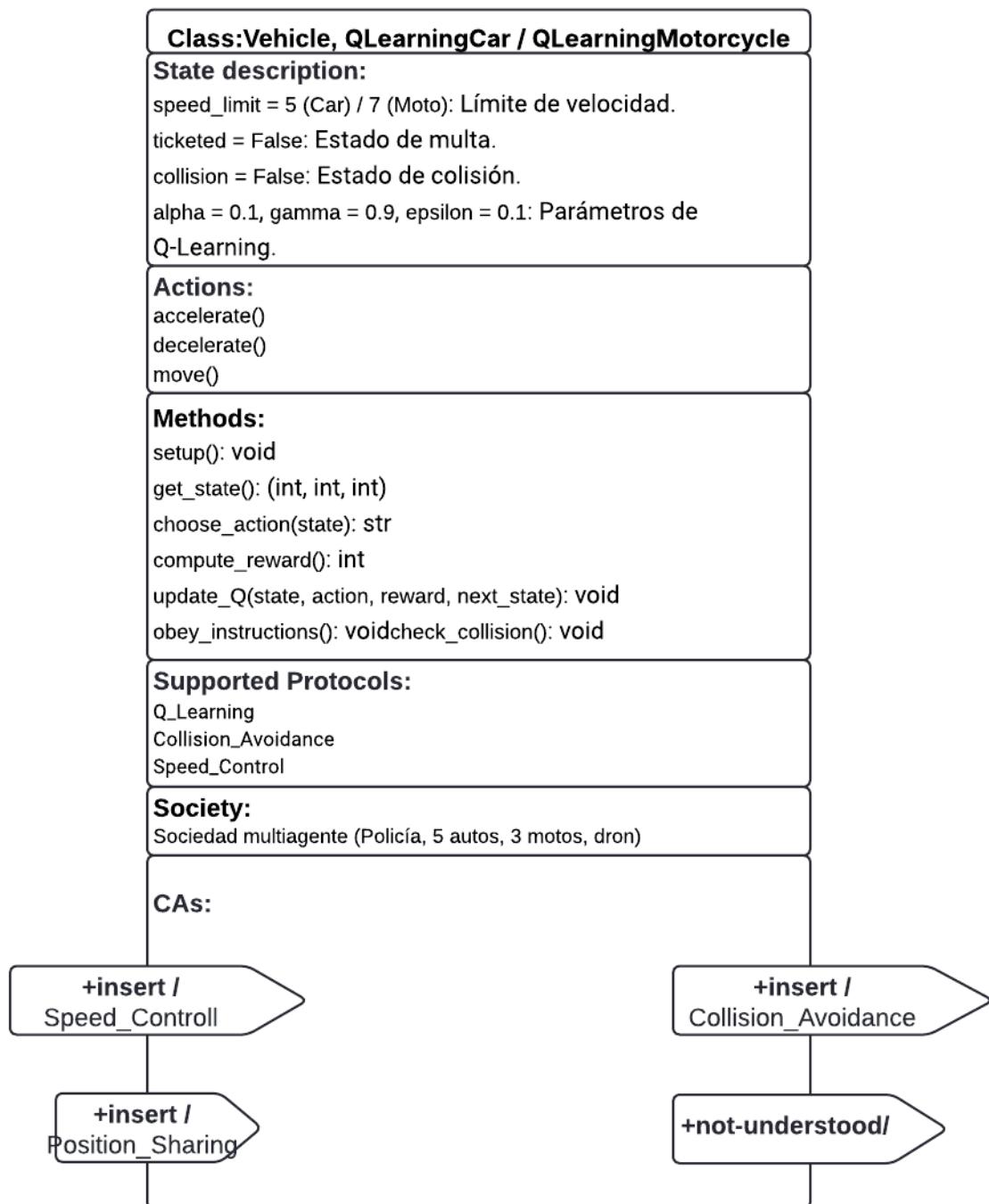
Diagramas UML

Diagramas de Agentes

Policía



Vehículos



Dron

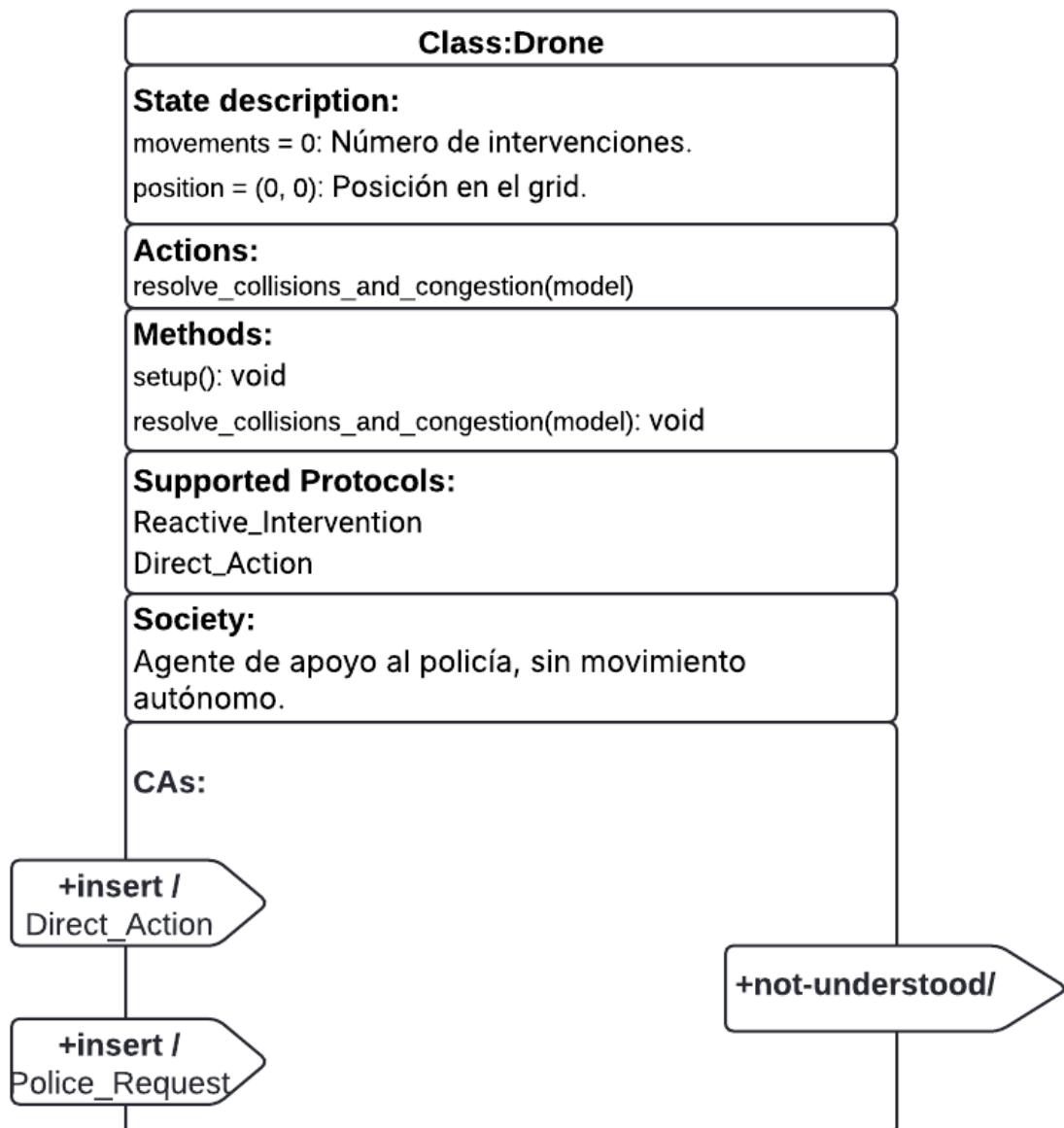


Diagrama de Protocolos

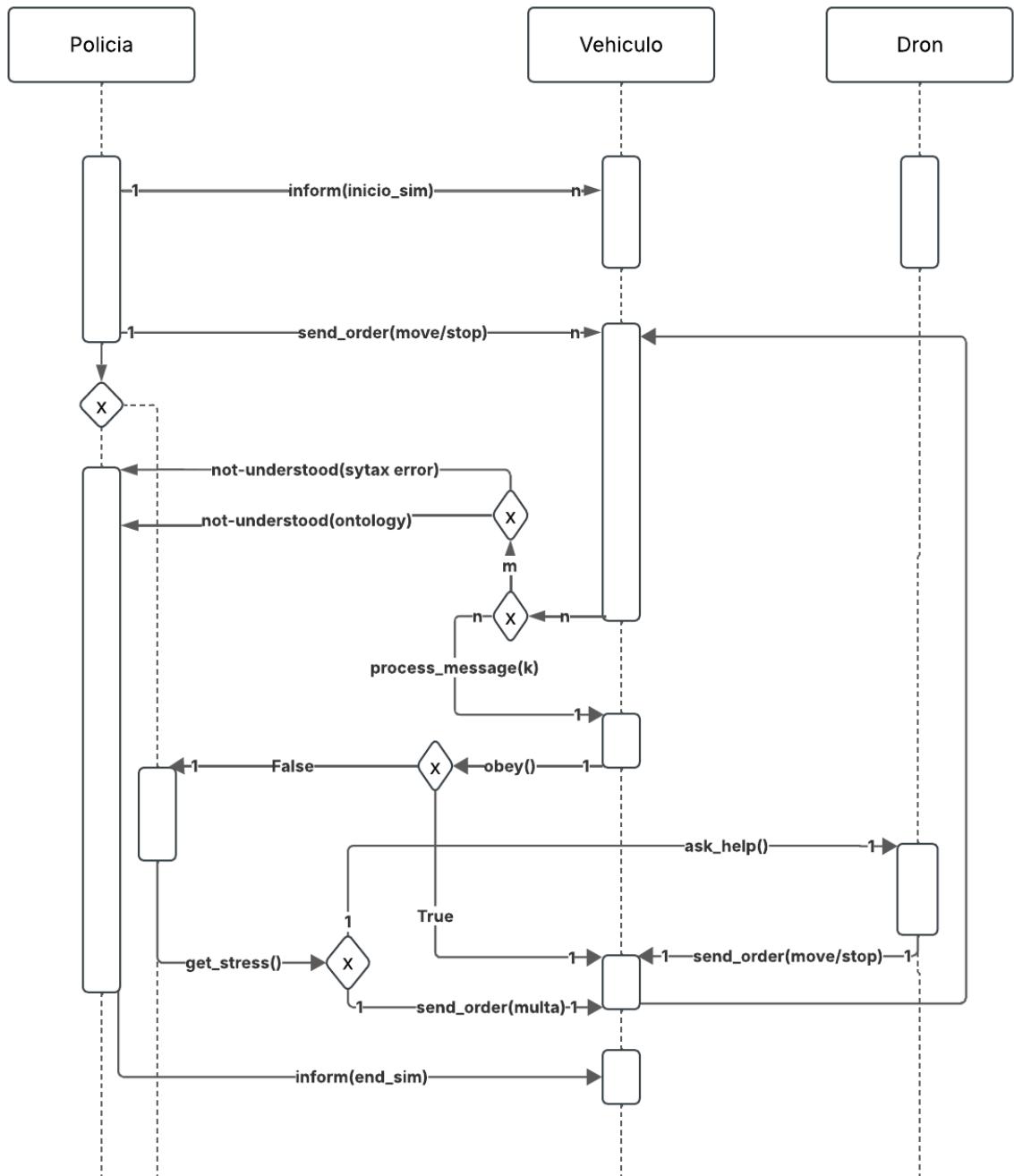
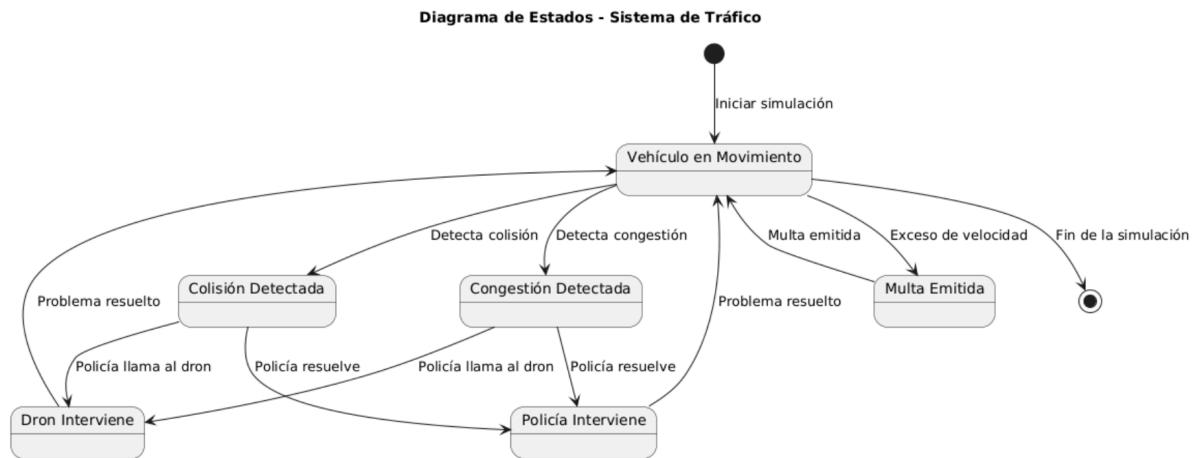


Diagrama de Estados



Aprendizaje multiagente

En esta implementación se ha integrado aprendizaje por refuerzo mediante Q-Learning en tres tipos de agentes: Car (QLearningCar), Motorcycle (QLearningMotorcycle) y el Police (QLearningPolice); en donde cada uno define sus propios estados y acciones en función de la dinámica del tráfico y mantiene una estructura de valores Q que mide la utilidad estimada de cada acción en cada estado, estas tablas Q se actualizan de forma iterativa, conforme los agentes ejecutan acciones y reciben recompensas o penalizaciones, permitiendo que definan su comportamiento de manera autónoma y adaptativa.

1. Q-Learning en Car y Motorcycle:

- Cada vehículo define su estado a partir de factores como el nivel de velocidad, la cercanía del policía y la posible congestión en su celda.
- En cada paso, el agente selecciona una acción (acelerar, mantener o desacelerar) con un criterio epsilon-greedy, equilibrando exploración y explotación de su política.
- Tras ejecutar la acción, se computa una recompensa con base en posibles colisiones, multas o conducción adecuada; el agente actualiza sus valores en su tabla de Q-Learning, ajustando paulatinamente sus decisiones para minimizar choques y sanciones.

2. Q-Learning en Police:

- El policía observa el estado del entorno en términos de cantidad de colisiones activas y congestionamientos en el mapa.
- Decide entre dos acciones principales: resolver los problemas por sí mismo o llamar al dron.
- Se define una recompensa que penaliza la persistencia de colisiones y congestiones, además de incluir un costo por solicitar apoyo al dron; de este modo, la política del

policía evoluciona para intentar optimizar la seguridad y el flujo del tráfico sin abusar del recurso externo.

La razón para utilizar Q-Learning en estos agentes es que se trata de un algoritmo model-free, ya que no necesita un modelo completo del entorno, capaz de aprender de la interacción constante sin requerir un conocimiento previo de las dinámicas de tráfico; de este modo los agentes de Car, Motorcycle y Police pueden ir refinando sus estrategias de manera autónoma, adaptándose a las condiciones y eventos que surgen durante la simulación; por otra parte el agente dron, funciona con reglas directas y no requiere este proceso de aprendizaje, ya que solo es activado por el policía para resolver ciertos eventos, en este caso congestiones y colisiones.

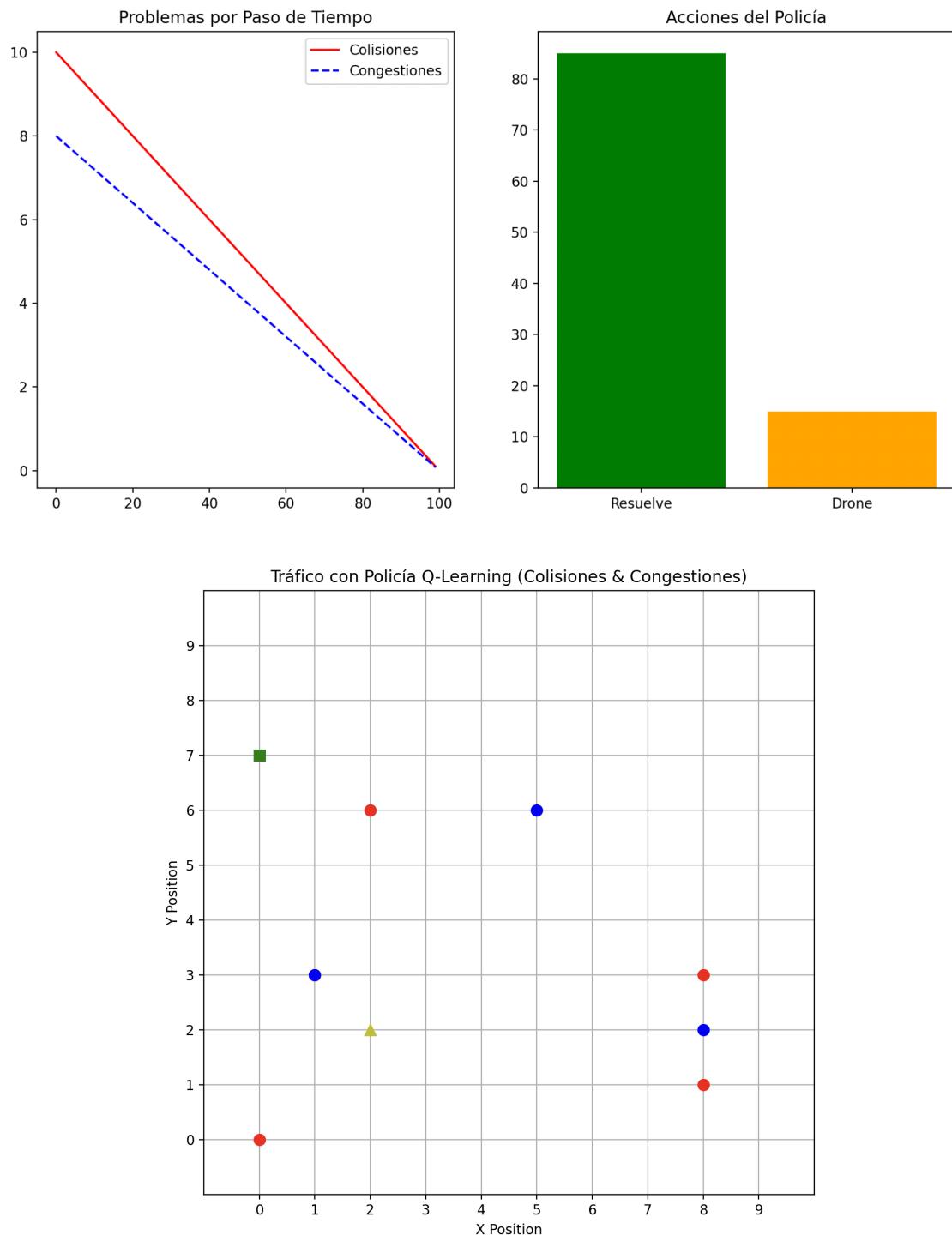
Resultados de la Simulación

Metas Cuantitativas

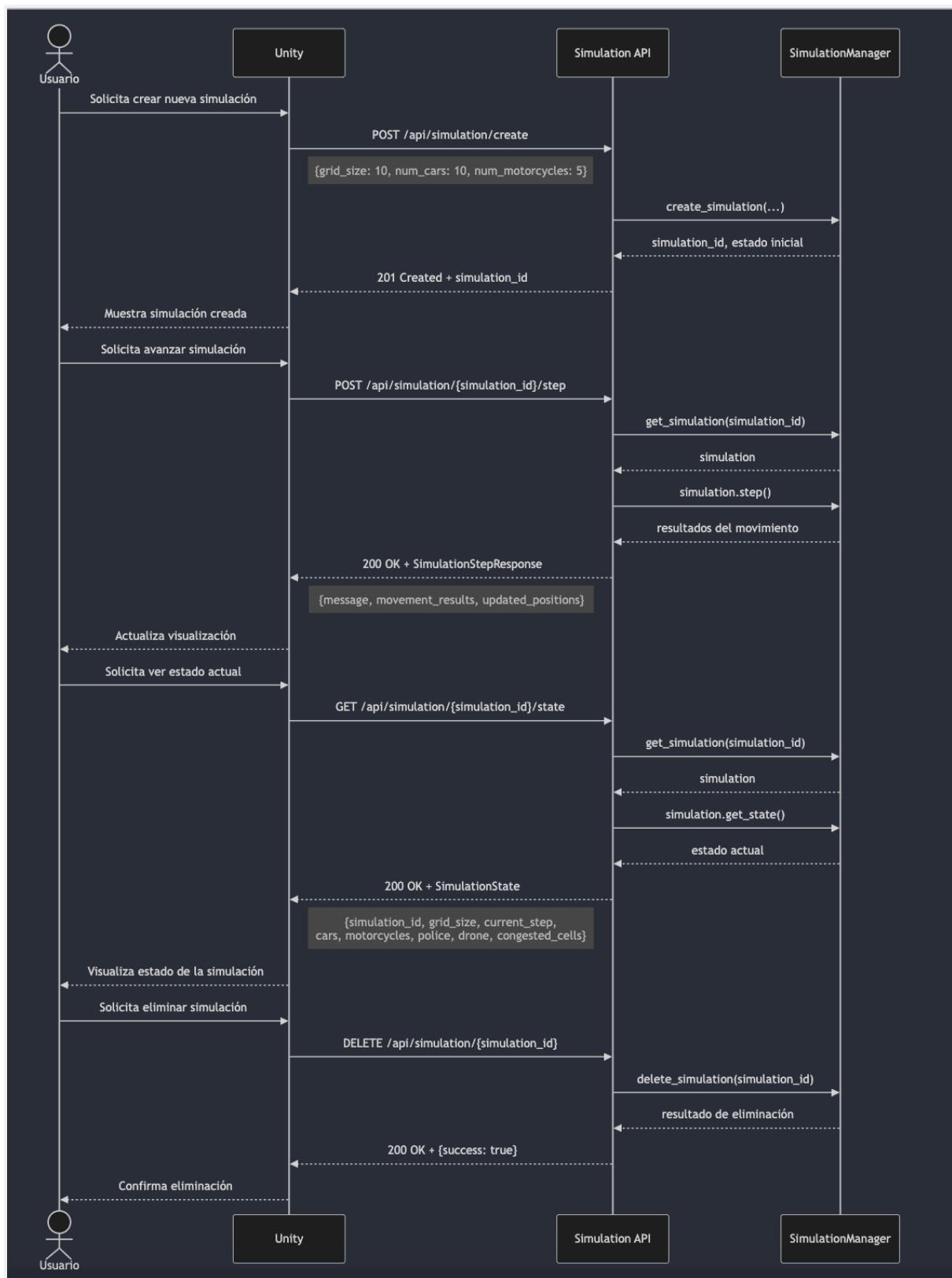
- Reducir colisiones en $\geq 70\%$ en 50 pasos
- Resolver congestiones en ≤ 3 pasos
- Policía resuelve problemáticas por sí mismo $\geq 80\%$
- $\leq 20\%$ de uso de drone

Descripción de Resultados

- Conforme pasa el tiempo de ejecución, las colisiones reducen significativamente, logrando alcanzar así el 70% o más de reducción de este incidente en 50 pasos
- Al poder terminar la ejecución y reducir las congestiones, se puede observar que estas no se juntan y se pueden resolver en 3 pasos o menos, de lo contrario, la ejecución se habría finalizado antes de que se terminara el tiempo
- El policía resuelve las problemáticas como colisiones, congestiones y multar, la mayor parte del tiempo por su propia cuenta, haciendo así que sus acciones sean equivalente al 80% o más, debido a que aprende a hacerlo sin la necesidad de pedirle mucha ayuda al Dron.
- Como el policía hace la mayoría de las acciones por su propia cuenta, no necesita usar al dron con tanta frecuencia para ayudarle a resolver las problemáticas, logrando así que el uso de este sea igual o menor al 20%



UML de apoyo acerca de la integración entre Sistemas Multiagentes y Gráficas computacionales



(Diagrama hecho en mermaid.md por Claude AI y editado por Sebastián Borjas)

El API cuenta con cuatro endpoints:

The screenshot shows the 'simulation' section of the API documentation. It includes four endpoints:

- POST /api/simulation/create**
- DELETE /api/simulation/{simulation_id}**
- GET /api/simulation/{simulation_id}/state**
- POST /api/simulation/{simulation_id}/step** (description: Avanzar la simulación por un paso)

El primer request que se tiene que mandar es a /api/simulation/create y se escribe de la siguiente manera:

```
curl -X 'POST' \
'http://127.0.0.1:5000/api/simulation/create' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "grid_size": 10,
  "num_cars": 10,
  "num_motorcycles": 5
}'
```

Lo cual crea una simulación de 10x10, con 10 carros y 5 motocicletas. A esta se le puede agregar el número que guste de carros y motocicletas que se quiera tener en la simulación. Al ejecutar este request, el API regresa una respuesta como la siguiente:

```
{
  "simulation_id": "0a45c352-96ea-4890-a2eb-ddf45204c15b",
  "message": "Simulacion iniciada",
  "initial_state": {
    "simulation_id": "0a45c352-96ea-4890-a2eb-ddf45204c15b",
    "grid": {
      "size": 10
    },
    "current_step": 0,
    "agents": {
      "police": {
        "position": [
          3,
          0
        ],
        "speed": 0,
        "movements": 0,
        "tickets_issued_count": 0,
        "congestion_resolved": 0,
        "drone_requests": 0,
        "failed_congestions": 0
      },
      "drone": {
        "position": [
          6,
          0
        ],
        "speed": 0,
        "movements": 0,
        "tickets_issued_count": 0,
        "congestion_resolved": 0,
        "drone_requests": 0,
        "failed_congestions": 0
      }
    }
  }
}
```

```
    1
    ],
  "speed": 0,
  "movements": 0
},
"cars": [
{
  "position": [
    3,
    2
  ],
  "speed": 2,
  "movements": 0,
  "id": 0,
  "ticketed": false,
  "collision": false,
  "speed_limit": 5
},
{
  "position": [
    8,
    1
  ],
  "speed": 0,
  "movements": 0,
  "id": 1,
  "ticketed": false,
  "collision": false,
  "speed_limit": 5
},
{
  "position": [
    4,
    9
  ],
  "speed": 0,
  "movements": 0,
  "id": 2,
  "ticketed": false,
  "collision": false,
  "speed_limit": 5
},
{
  "position": [
    1,
    4
  ],
  "speed": 4,
  "movements": 0,
  "id": 3,
  "ticketed": false,
  "collision": false,
  "speed_limit": 5
},
{
  "position": [
    0,
    7
  ],
  "speed": 0,
```

```
        "speed": 2,
        "movements": 0,
        "id": 4,
        "ticketed": false,
        "collision": false,
        "speed_limit": 5
    },
    {
        "position": [
            2,
            3
        ],
        "speed": 0,
        "movements": 0,
        "id": 5,
        "ticketed": false,
        "collision": false,
        "speed_limit": 5
    },
    {
        "position": [
            8,
            7
        ],
        "speed": 5,
        "movements": 0,
        "id": 6,
        "ticketed": false,
        "collision": false,
        "speed_limit": 5
    },
    {
        "position": [
            5,
            0
        ],
        "speed": 1,
        "movements": 0,
        "id": 7,
        "ticketed": false,
        "collision": false,
        "speed_limit": 5
    },
    {
        "position": [
            6,
            9
        ],
        "speed": 4,
        "movements": 0,
        "id": 8,
        "ticketed": false,
        "collision": false,
        "speed_limit": 5
    },
    {
        "position": [
            6,
            6
        ]
    }
]
```

```
],
  "speed": 2,
  "movements": 0,
  "id": 9,
  "ticketed": false,
  "collision": false,
  "speed_limit": 5
}
],
  "motorcycles": [
    {
      "position": [
        5,
        2
      ],
      "speed": 1,
      "movements": 0,
      "id": 0,
      "ticketed": false,
      "collision": false,
      "speed_limit": 7
    },
    {
      "position": [
        9,
        8
      ],
      "speed": 4,
      "movements": 0,
      "id": 1,
      "ticketed": false,
      "collision": false,
      "speed_limit": 7
    },
    {
      "position": [
        7,
        3
      ],
      "speed": 2,
      "movements": 0,
      "id": 2,
      "ticketed": false,
      "collision": false,
      "speed_limit": 7
    },
    {
      "position": [
        1,
        2
      ],
      "speed": 1,
      "movements": 0,
      "id": 3,
      "ticketed": false,
      "collision": false,
      "speed_limit": 7
    }
  ]
}
```

```

    "position": [
      3,
      8
    ],
    "speed": 3,
    "movements": 0,
    "id": 4,
    "ticketed": false,
    "collision": false,
    "speed_limit": 7
  }
],
},
"congested_cells": [],
"status": {
  "task_completed": false,
  "task_completion_time": null,
  "failed_congestions": 0,
  "total_movements": 0,
  "start_time": "2025-03-11T13:16:04.910387"
}
}
}

```

Donde se puede observar la información inicial sobre la simulación.

El siguiente endpoint que el cliente debe llamar es el step para avanzar en la simulación, el request se verá de la siguiente manera.

```

curl -X 'POST' \
'http://127.0.0.1:5000/api/simulation/0a45c352-96ea-4890-a2eb-ddf45204c15b/step' \
-H 'accept: application/json' \
-d "

```

Tomando como parámetro el simulation id obtenido en el request de create.

Este request deberá regresar información sobre los movimientos realizados en la simulación:

```

{
  "message": "Simulation step processed",
  "movement_results": {
    "cars_moved": 6,
    "motorcycles_moved": 1,
    "collisions_detected": 0,
    "congested_cells": [
      [
        0,
        7
      ]
    ],
    "game_over": true,
    "reason": "Tres fallos consecutivos en resolver congestiones",
    "task_completed": false
  },
  "updated_positions": {
    "cars": [
      {
        "id": 0,
        "position": [

```

```
    6,
    3
  ],
},
{
  "id": 1,
  "position": [
    7,
    9
  ],
},
{
  "id": 2,
  "position": [
    3,
    9
  ],
},
{
  "id": 3,
  "position": [
    9,
    7
  ],
},
{
  "id": 4,
  "position": [
    0,
    7
  ],
},
{
  "id": 5,
  "position": [
    5,
    9
  ],
},
{
  "id": 6,
  "position": [
    5,
    9
  ],
},
{
  "id": 7,
  "position": [
    0,
    0
  ],
},
{
  "id": 8,
  "position": [
    9,
    0
  ],
}
```

```
        },
        {
          "id": 9,
          "position": [
            0,
            0
          ]
        }
      ],
      "motorcycles": [
        {
          "id": 0,
          "position": [
            0,
            7
          ]
        },
        {
          "id": 1,
          "position": [
            0,
            6
          ]
        },
        {
          "id": 2,
          "position": [
            9,
            9
          ]
        },
        {
          "id": 3,
          "position": [
            0,
            7
          ]
        },
        {
          "id": 4,
          "position": [
            3,
            0
          ]
        }
      ],
      "police": {
        "position": [
          3,
          0
        ]
      },
      "drone": {
        "position": [
          6,
          1
        ]
      }
    }
  ]
```

```
}
```

En caso de que el cliente quiera ver el estado actual detalladamente deberá llamar al siguiente endpoint:

```
curl -X 'GET' \
'http://127.0.0.1:5000/api/simulation/0a45c352-96ea-4890-a2eb-ddf45204c15b/state' \
-H 'accept: application/json'
```

Mediante el cual recibe información detallada en el siguiente formato:

```
{
  "simulation_id": "0a45c352-96ea-4890-a2eb-ddf45204c15b",
  "grid": {
    "size": 10
  },
  "current_step": 9,
  "agents": {
    "police": {
      "position": [
        3,
        0
      ],
      "speed": 0,
      "movements": 26,
      "tickets_issued_count": 14,
      "congestion_resolved": 3,
      "drone_requests": 6,
      "failed_congestions": 3
    },
    "drone": {
      "position": [
        6,
        1
      ],
      "speed": 0,
      "movements": 7
    },
    "cars": [
      {
        "position": [
          6,
          3
        ],
        "speed": 8,
        "movements": 10,
        "id": 0,
        "ticketed": true,
        "collision": false,
        "speed_limit": 5
      },
      {
        "position": [
          7,
          9
        ],
        "speed": 8,
        "movements": 10,
        "id": 1,
        "ticketed": false,
        "collision": false,
        "speed_limit": 5
      }
    ]
  }
}
```

```
"speed": 10,  
"movements": 10,  
"id": 1,  
"ticketed": true,  
"collision": false,  
"speed_limit": 5  
},  
{  
  "position": [  
    3,  
    9  
  ],  
  "speed": 5,  
  "movements": 9,  
  "id": 2,  
  "ticketed": true,  
  "collision": false,  
  "speed_limit": 5  
},  
{  
  "position": [  
    9,  
    7  
  ],  
  "speed": 10,  
  "movements": 10,  
  "id": 3,  
  "ticketed": true,  
  "collision": false,  
  "speed_limit": 5  
},  
{  
  "position": [  
    0,  
    7  
  ],  
  "speed": 5,  
  "movements": 10,  
  "id": 4,  
  "ticketed": true,  
  "collision": false,  
  "speed_limit": 5  
},  
{  
  "position": [  
    5,  
    9  
  ],  
  "speed": 9,  
  "movements": 9,  
  "id": 5,  
  "ticketed": true,  
  "collision": false,  
  "speed_limit": 5  
},  
{  
  "position": [  
    5,  
    9
```

```
],
  "speed": 9,
  "movements": 10,
  "id": 6,
  "ticketed": true,
  "collision": false,
  "speed_limit": 5
},
{
  "position": [
    0,
    0
  ],
  "speed": 9,
  "movements": 10,
  "id": 7,
  "ticketed": true,
  "collision": false,
  "speed_limit": 5
},
{
  "position": [
    9,
    0
  ],
  "speed": 9,
  "movements": 10,
  "id": 8,
  "ticketed": true,
  "collision": false,
  "speed_limit": 5
},
{
  "position": [
    0,
    0
  ],
  "speed": 8,
  "movements": 10,
  "id": 9,
  "ticketed": true,
  "collision": false,
  "speed_limit": 5
}
],
  "motorcycles": [
  {
    "position": [
      0,
      7
    ],
    "speed": 8,
    "movements": 10,
    "id": 0,
    "ticketed": true,
    "collision": false,
    "speed_limit": 7
  },
  {
```

```
"position": [
  0,
  6
],
"speed": 9,
"movements": 10,
"id": 1,
"ticketed": true,
"collision": false,
"speed_limit": 7
},
{
  "position": [
    9,
    9
  ],
  "speed": 4,
  "movements": 10,
  "id": 2,
  "ticketed": false,
  "collision": false,
  "speed_limit": 7
},
{
  "position": [
    0,
    7
  ],
  "speed": 8,
  "movements": 10,
  "id": 3,
  "ticketed": true,
  "collision": false,
  "speed_limit": 7
},
{
  "position": [
    3,
    0
  ],
  "speed": 10,
  "movements": 10,
  "id": 4,
  "ticketed": true,
  "collision": false,
  "speed_limit": 7
}
],
"congested_cells": [
  [
    0,
    7
  ]
],
"status": {
  "task_completed": false,
  "task_completion_time": null,
  "failed_congestions": 0
}
```

```
        "total_movements": 181,  
        "start_time": "2025-03-11T13:16:04.910387"  
    }  
}
```

Para terminar y borrar la simulación, se debe llamar al endpoint DELETE api/simulation/id:

```
curl -X 'DELETE' \  
'http://127.0.0.1:5000/api/simulation/0a45c352-96ea-4890-a2eb-ddf45204c15b' \  
-H 'accept: application/json'
```

Y de esta manera, se obtiene una confirmación:

```
{  
    "success": true,  
    "message": "Simulacion 0a45c352-96ea-4890-a2eb-ddf45204c15b eliminada exitosamente"  
}
```

Conclusiones Finales

El presente proyecto de simulación de tráfico urbano basado en agentes inteligentes y aprendizaje por refuerzo ha demostrado ser una herramienta efectiva para modelar y optimizar la gestión del tráfico en entornos urbanos. A través del uso de Q-Learning y la implementación en Unity, se logró desarrollar un sistema capaz de tomar decisiones en tiempo real para mejorar el flujo vehicular y reducir la congestión.

Durante el desarrollo del proyecto, se identificaron varios desafíos técnicos, incluyendo la definición y cálculo de los parámetros de aprendizaje y de las recompensas adecuadas. Al igual que la integración de la simulación con una API para su evaluación externa. No obstante, estos obstáculos fueron superados mediante ajustes iterativos y pruebas exhaustivas, lo que permitió mejorar el desempeño del sistema y obtener resultados más precisos.

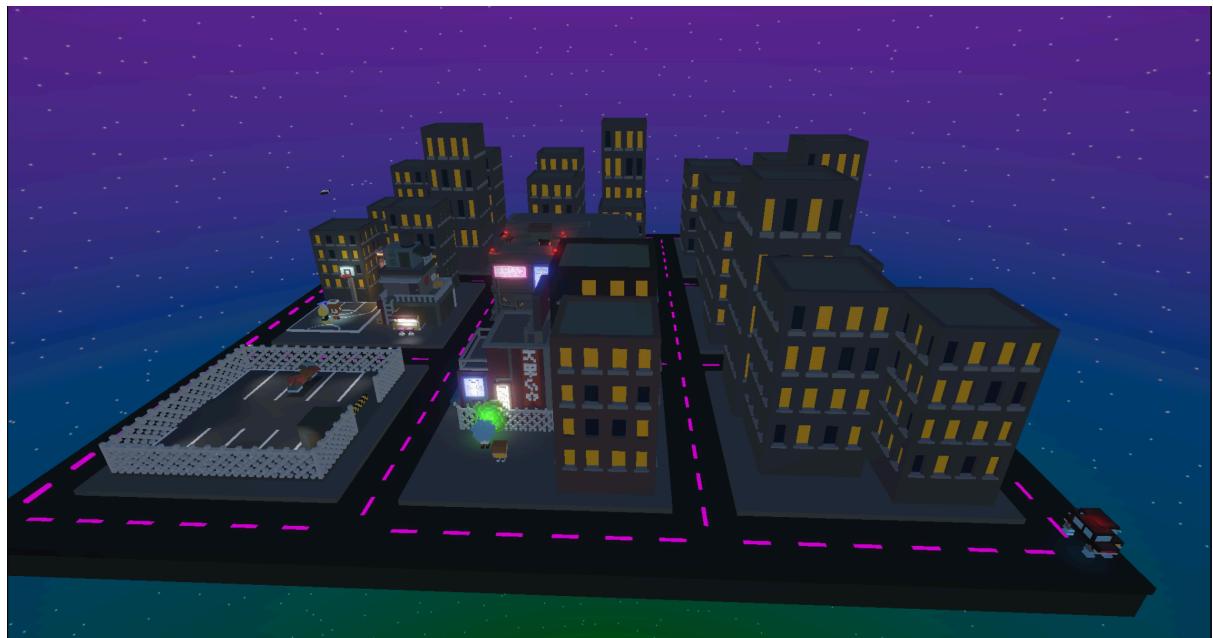
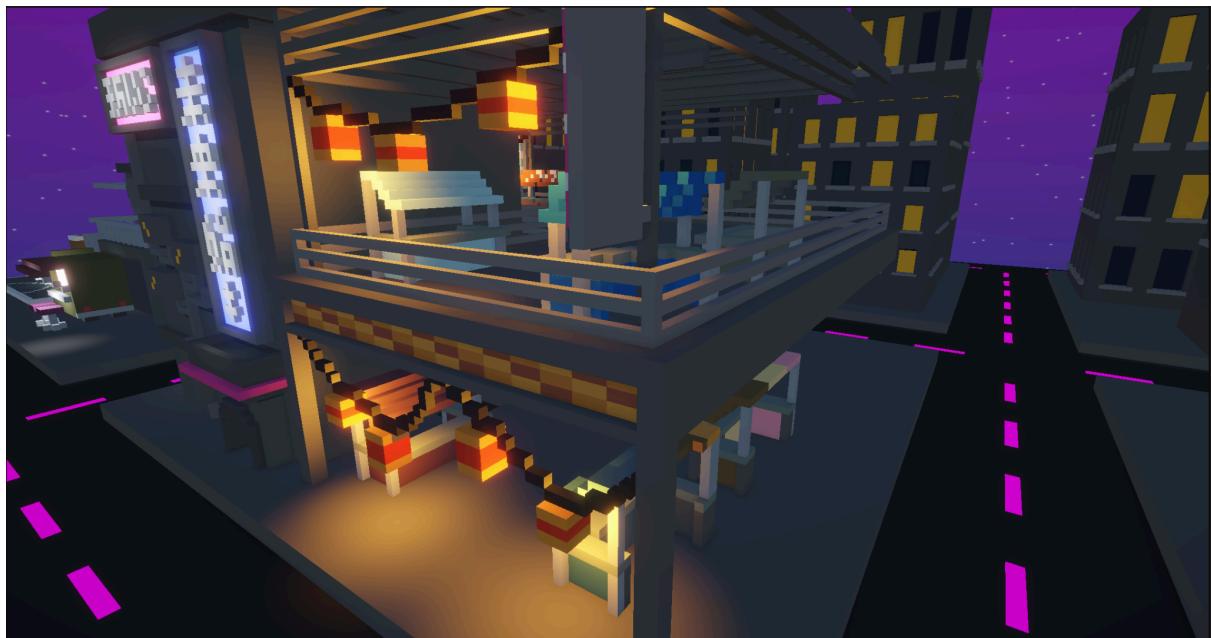
Los principales hallazgos del proyecto incluyen:

- La implementación del algoritmo Q-Learning, el cual permitió que los agentes mejoraran su capacidad de decisión con el tiempo, ajustando dinámicamente las acciones que estos toman.
- La simulación en Unity, la cual brindó una plataforma flexible y visualmente comprensible para observar el comportamiento de los vehículos y evaluar diferentes estrategias de control.
- La API desarrollada, la cual facilitó la recolección datos, permitiendo la conexión entre los sistemas multiagentes y las gráficas computacionales.

Para mejorar la toma de decisiones de los agentes, en el futuro, se pueden emplear técnicas de aprendizaje más avanzadas y redes neuronales. Asimismo, la implementación de entornos más complejos con factores adicionales, como condiciones climáticas o interacciones con peatones, permitiría enriquecer la simulación y obtener resultados más realistas. En conclusión, el proyecto demuestra el potencial del aprendizaje por refuerzo en la gestión del tráfico y abre nuevas oportunidades para la aplicación de inteligencia artificial en la movilidad urbana.

Anexos





Los modelos fueron realizados utilizando el software *Magicavoxel* y sus respectivas herramientas, como principalmente fuente de inspiración se utilizaron imágenes de ciudades *Cyberpunk* en Pinterest. Para la iluminación se utilizó una combinación de de point lights y texturas con alto brillo para simular el color neón de los carteles.