Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Sistemas multiagentes

Módulo de Reto:

Avance de proyecto

Equipo #4

Carlos Francisco Sanchez Llanes - A01741201

Andrés Eduardo Gomes Brandt - A01781321

Saúl Roberto Orozco Villanazul - A00831554

Fernando Adrián Fuentes Martínez - A01028796:

23- Enero- 2025

Simulación multiagentes en movilidad urbana

La simulación multiagentes ofrece una solución innovadora para abordar los complejos desafíos de la movilidad urbana moderna:

• Modelado realista del comportamiento individual de vehículos, peatones y semáforos.

• Evaluación de diferentes configuraciones de infraestructura

• Optimización de flujos de tráfico en tiempo real.

Objetivos de investigación del proyecto

Objetivo general: Optimizar el flujo vehicular en intersecciones urbanas mediante simulación multiagentes, identificando las configuraciones óptimas de infraestructura y control de tráfico.

Objetivos específicos:

- Comparar peatones que obedecen y desobedecen las leyes de tránsito, ver cuanto tardan en llegar a sus destinos y graficar los resultados.
- Emplear del método de A* con peso para poder observar como los peatones obtienen la ruta mas eficiente para poder llegar a su destino

1. `Obstacle`

Rol:

Representa un obstáculo en el entorno, impidiendo el movimiento de otros agentes (peatones y automóviles).

Comportamiento:

- No permite que otros agentes pasen por su ubicación.
- Se mantiene estático en la simulación.

Atributos:

- `weight = -1`: Indica que esta celda es intransitable.

Métodos:

- `setup()`: Inicializa el peso del obstáculo en `-1` para impedir el movimiento sobre él.

---

2. `Spawn`

Rol:

Es un punto de generación de peatones en la simulación.

Comportamiento:

- Se utiliza como punto inicial para los peatones que aparecen en la ciudad.
- No afecta el tránsito de otros agentes.

Atributos:

- `weight = 0`: No afecta el tránsito.

Métodos:

- `setup()`: Inicializa su peso en `0`.

---

3. `Building`

Rol:

Representa los edificios dentro del entorno, actuando como destinos para los peatones.

Comportamiento:

- No es transitable.
- Sirve como destino en la planificación de rutas para los peatones.

Atributos:

- `weight = 0`: No afecta el tránsito.

Métodos:

- `setup()`: Inicializa su peso en `0`.

---

4. `Stoplight`

Rol:

Controla el tráfico vehicular mediante un sistema de semáforos.

Comportamiento:

- Alterna su estado entre verde (`True`) y rojo (`False`).
- Afecta el flujo de vehículos y peatones en las intersecciones.

Atributos:

- `id`: Identificador único del semáforo.
- `change_time`: Tiempo que tarda en cambiar de estado.
- `state`: `True` si está en verde, `False` si está en rojo.
- `pos`: Ubicación del semáforo.
- `myRoads`: Lista de carreteras que controla.
- `weight = 0`: No impide el tránsito directo.
- `rayados`: Lista de carreteras con paso peatonal regulado por el semáforo.

Métodos:

- `setup()`: Inicializa el semáforo.
- `execute()`:
  - Alterna el estado del semáforo y actualiza las carreteras controladas.

---

5. `Road`

Rol:

Representa una carretera por la que circulan vehículos y peatones.

Comportamiento:

- Puede contener señales de alto o pasos peatonales.
- Su dirección define el sentido de circulación de los autos.

Atributos:

- `dir`: Dirección de la carretera.
- `paso_peatonal`: `True` si tiene un cruce peatonal.
- `stop`: `True` si hay un semáforo en rojo o señal de alto.
- `weight = 10`: Indica la dificultad relativa para moverse en ella.
- `flag`: Indica si la carretera está ocupada.
- `p_stop`: Indica si un peatón debe detenerse antes de cruzar.

Métodos:

- `setup()`: Inicializa los atributos de la carretera.

---

6. `Peaton`

Rol:

Simula el comportamiento de un peatón en la ciudad, desplazándose hacia un destino determinado.

Comportamiento:

- Se mueve a lo largo de una ruta definida mediante el algoritmo A*.
- Puede respetar o ignorar las reglas de tránsito de manera probabilística.
- Interactúa con semáforos y otros agentes para evitar colisiones.

Atributos:

- `id`: Identificador único.
- `env`: Referencia al entorno.
- `route`: Lista de posiciones que sigue hasta su destino.
- `next_step`: Índice del siguiente paso en la ruta.
- `respect`: Determina si respeta las reglas de tránsito (probabilidad aleatoria).
- `step`: Contador de pasos dados.

Métodos:

- `setup()`: Define el comportamiento inicial del peatón.
- `execute()`:
  - Determina si sigue su ruta o se detiene según el tráfico y los semáforos.
  - Registra si el peatón obedece o desobedece las normas de tránsito.
- `get_position()`: Retorna la posición actual del peatón.

---

7. `Car`

Rol:

Representa un automóvil en la ciudad, que se mueve de acuerdo con las reglas de tránsito y evita colisiones.

Comportamiento:

- Se mueve en la dirección asignada.
- Evita colisiones con otros autos y peatones.
- Puede cambiar de dirección en intersecciones.

Atributos:

- `id`: Identificador único.
- `dir`: Dirección actual.
- `prev_dir`: Dirección anterior.

Métodos:

- `setup()`: Inicializa el auto.
- `move_car()`:
  - Determina el próximo movimiento evitando colisiones.
  - Respeta los semáforos y las intersecciones.
- `execute()`:
  - Mueve el auto en la dirección válida.
  - Si no puede moverse, se elimina de la simulación.
  -

Ambiente

El entorno de la simulación está definido como una ciudad, generada a partir de un archivo de texto que luego es convertido a una matriz y posteriormente a un grid que toma como dimensiones las filas y columnas de dicha matriz. Este ambiente tiene las siguientes características :

Accesible: El ambiente es accesible ya que los peatones poseen conocimiento completo sobre el ambiente, siendo la posición de los distintos agentes que se encuentran en el grid (lo cual es cierto exclusivamente para los otros coches) la única cosa que desconocen

Determinístico: Se debe dar a entender que pese a que los agentes interactuan en gran medida con el ambiente, el estado sigue sin cambiar

No Episódico: El ambiente no es episódico debido a que los semáforos son dependientes del estado en el que se encontraban en episodios anteriores para tener un conteo de cuando deberían cambiar de estado.

Estático: Debido a que toda la comunicación y manipulación que se realiza dentro del ambiente se realiza entre los agentes. De haberse implementado algunos de los agentes como parte del ambiente, se consideraría como un ambiente dinámico

Discreto: Todos los valores que se trabajan dentro del ambiente son discretos al ser un grid.

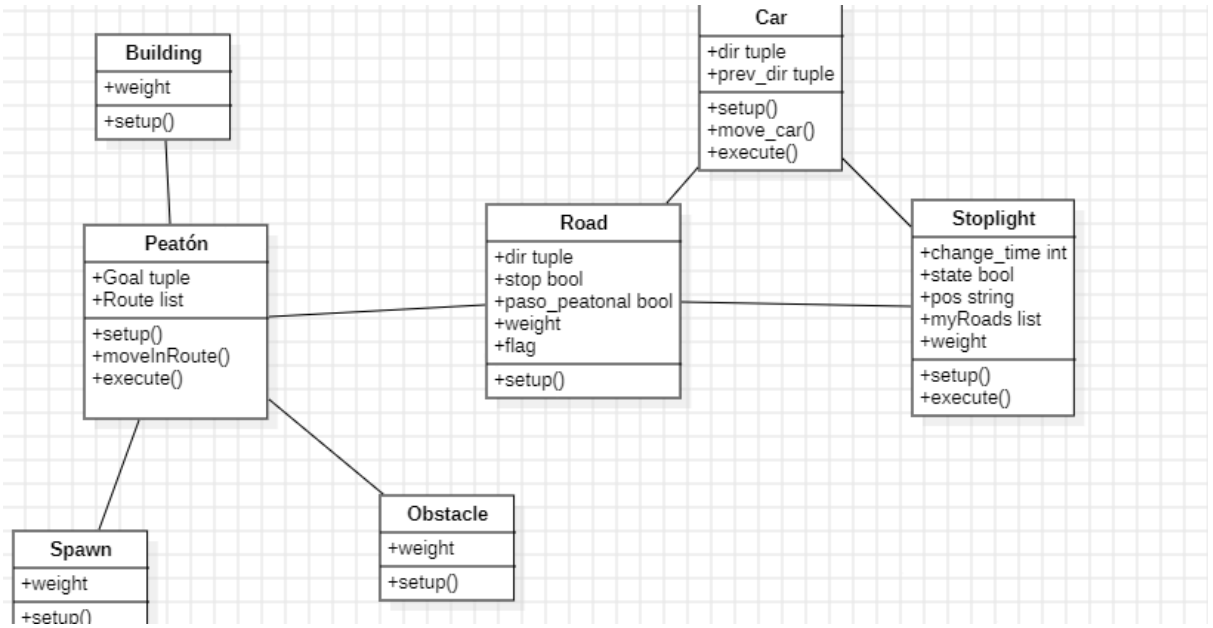
Métricas

Tiempo promedio desde origen hasta meta: Evalúa la eficiencia del tráfico y los desplazamientos peatonales.

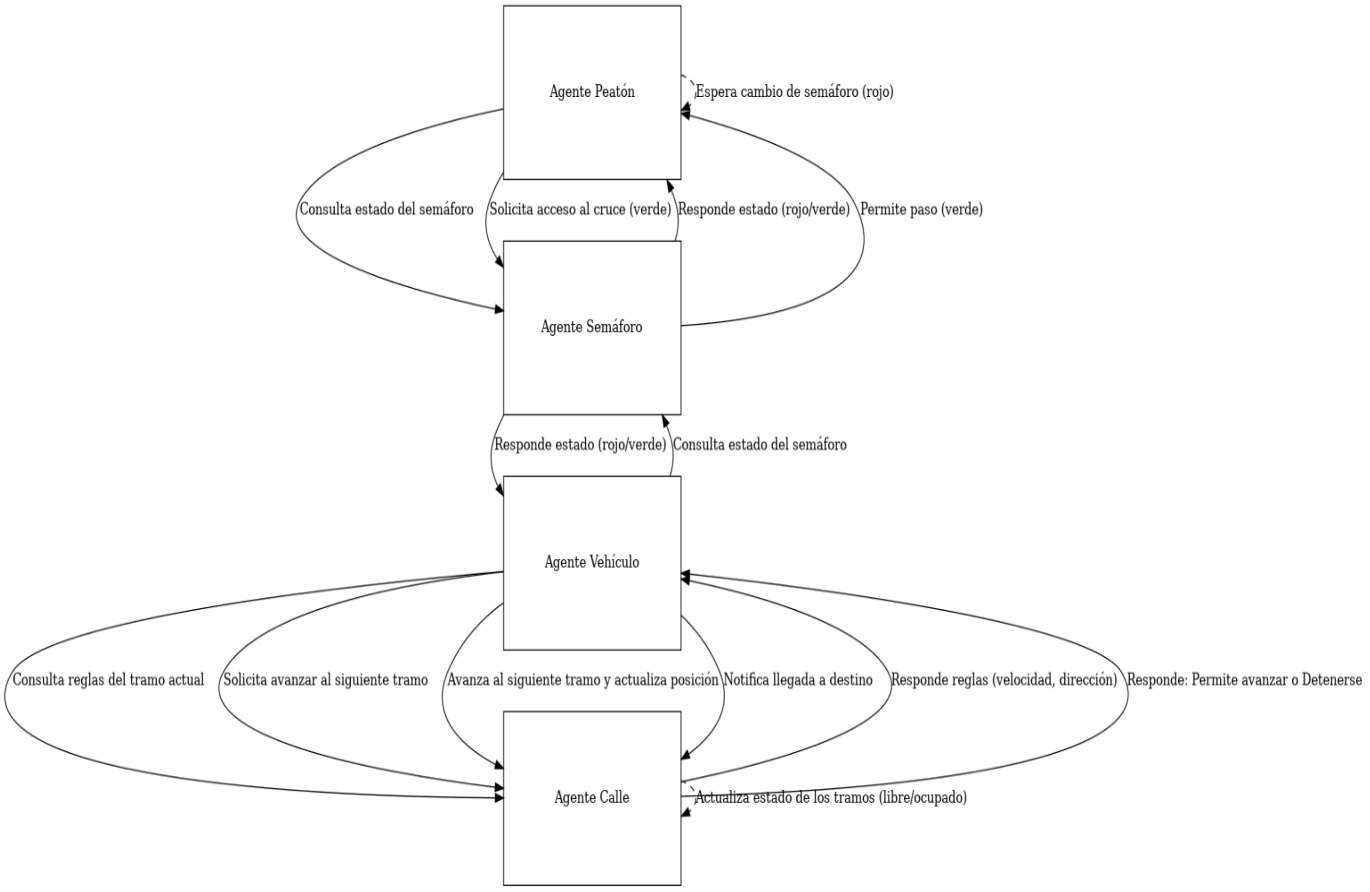
Steps de Peatones obedientes vs no obedientes: Compara si los peatones desobedientes llegan más rápido que los obedientes

Diagramas

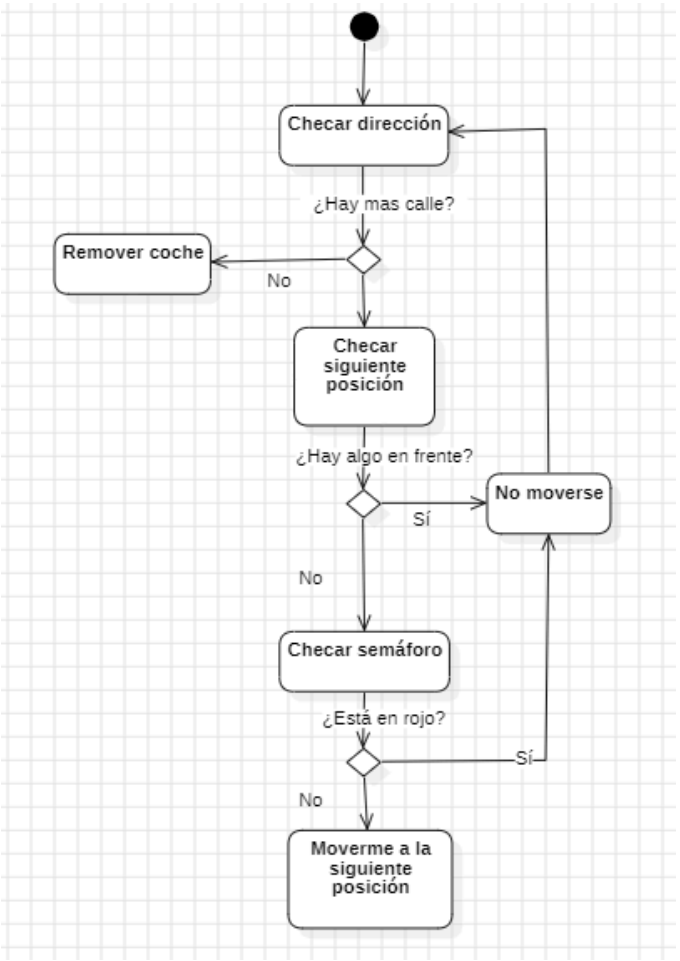
Diagrama de clases de los agentes



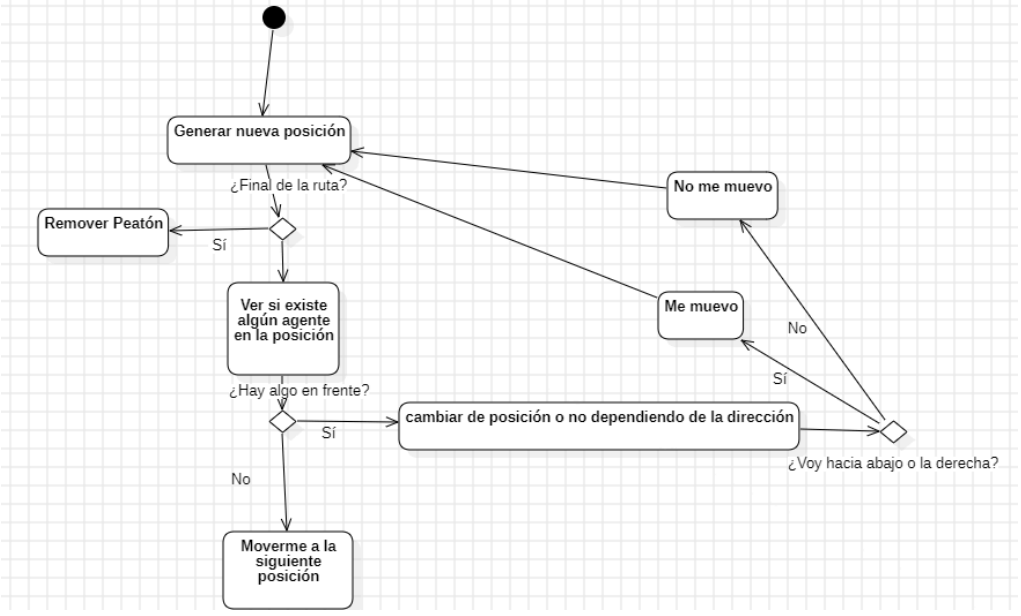
Protocolos de interacción



Movimiento de coche:



Movimiento del Peatón:



Base del mapa del modelo:

```
3    B00000000^v00000000B
4    000000000^v000000000
5    000OOO000^v000OOO000
6    B00000000^v00000000B
7    000OOO000^v000OOO000
8    000000000^v000000000
9    000000000^vS00000000
10   <<<<<<<<<++<<<<<<<<<
11   >>>>>>>>>++>>>>>>>>>
12   00000000s^v000000000
13   000000000^v000000000
14   000OOO000^v000OOO000
15   B00000000^v00000000B
16   000OOO000^v000OOO000
17   B00000000^v00000000B
18   000OOO000^v000OOO000
19   000000000^v000000000
20   00P000P00^v000P000P0
```

P: Spawn points

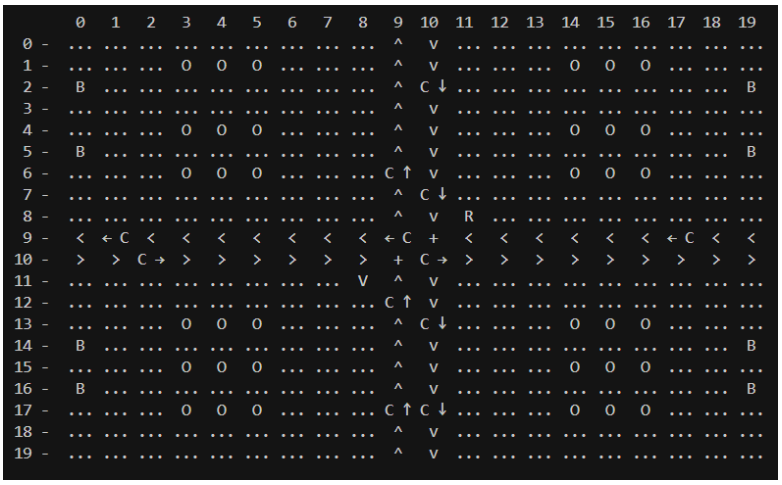
+,<,>,v,^: calles

O: Obstáculos

S, s: Semáforos

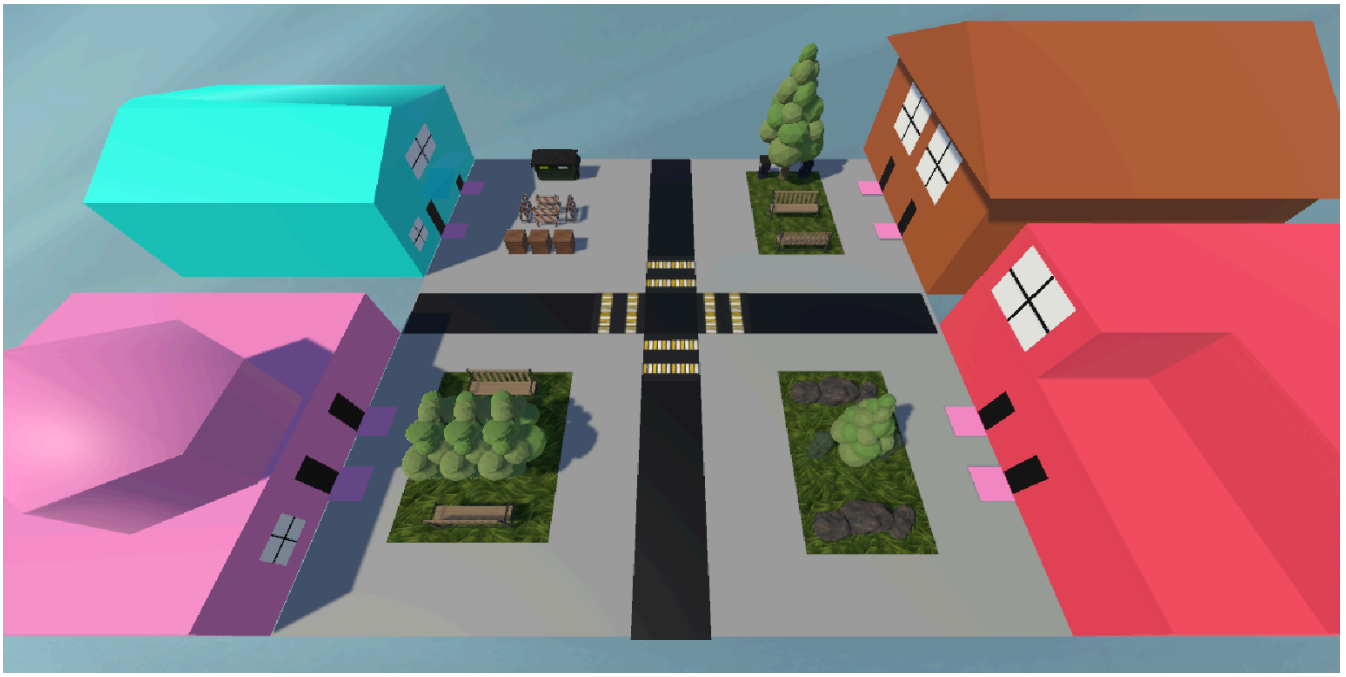
B: Edificios

Otro print del modelo, note que los spawn points son invisibles en esta representación y las C representan coches



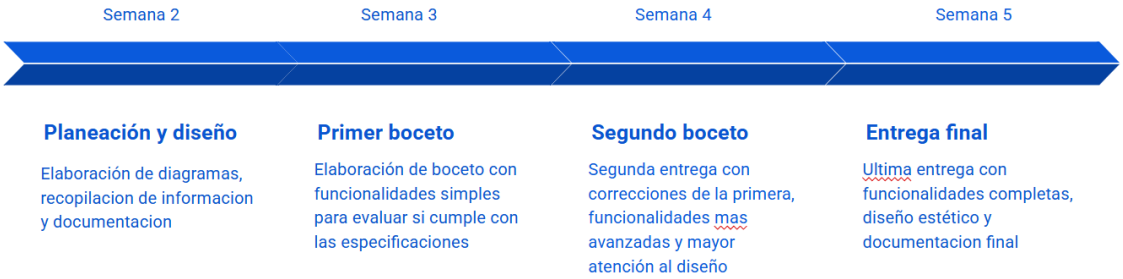
.

Representación y diseño en Unity



Plan de trabajo

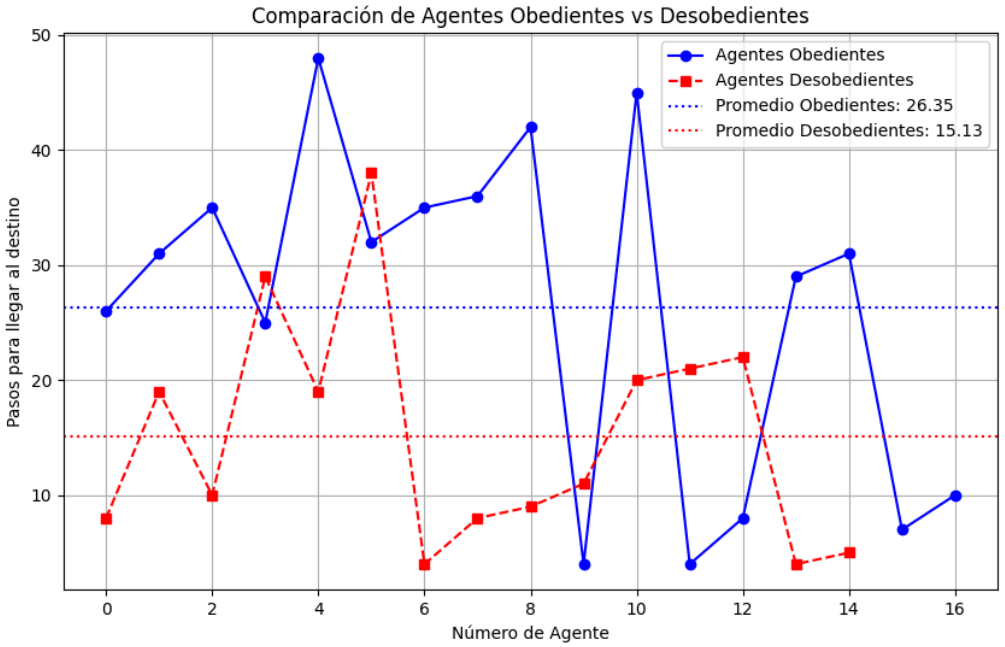
Planeador de milestones en tiempo



Herramientas y plataformas

Herramientas	Plataformas
Repositorio remoto	Github
Planeador de tareas	Jira
Asistente	ChatGPT
Redaccion de documentacion	Google Docs
Lenguaje de programacion	Python
Entorno de programacion	Unity

Conclusiones:



- Los agentes obedientes tomaron mas pasos de ejecución para alcanzar su meta
- Los agentes desobedientes llegaron antes a sus destinos debido a que evitan perder tiempo en semaforos y encuentran rutas más óptimas cortando camino a traves de la calle

Link de Github con proyecto y video de ejecución

https://github.com/Remy404/respaldo_multiagentes.git