Reporte Reto

TC2008B: Modelación de sistemas con gráficas computacionales (Gpo 301)

30.11.2023

—

Pablo Banzo Prida - A01782031

María Fernanda Cortés Lozano - A01026613

Índice

Planteamiento del problema

Este proyecto aborda el problema de la congestión del tráfico y la movilidad urbana ineficiente en las ciudades mexicanas. Enfrentamos retos como el aumento del uso de vehículos privados, la inadecuada planificación de rutas y la falta de alternativas de transporte eficientes. El principal desafío es mejorar la fluidez del tráfico y reducir la congestión, lo cual es crucial para el desarrollo económico y la calidad de vida de los ciudadanos.

Solución propuesta

La solución propuesta para el desafío de la movilidad urbana en México consistió en desarrollar una simulación multiagentes de tráfico urbano, utilizando tanto Unity para una representación 3D detallada como un modelo de simulación de mesa para análisis y pruebas rápidas.

La simulación incluye un sistema dinámico de gestión del tráfico, donde los vehículos se mueven por una red de calles definidas por un mapa, interactuando entre sí y con elementos del entorno urbano, como semáforos y otros vehículos. Se utilizaron algoritmos para optimizar las rutas, reducir tiempos de viaje y evitar congestiones, además de variables “humanas” para agregar cierta inteligencia a los agentes.

Esta simulación se desarrolló en un entorno virtual que representa de manera realista las condiciones de tráfico en una ciudad mexicana, proporcionando una herramienta valiosa para la planificación urbana y la toma de decisiones en políticas de transporte.

Cabe agregar, debido a los requerimientos del sistema solución, fue necesario transformar los puntos del *mesh* de los agentes coche mediante matrices compuestas, las cuales fueron codificadas manualmente, lo cual añadió cierta complejidad a la implementación.

Diseño de agentes

En términos de diseño, los agentes muestran proactividad mediante variables humanas como “greediness” que permiten que los vehículos detecten obstáculos y los eviten mediante costos adicionales en la implementación de A* (forzandolos a cambiar de carril).
Por su lado, se puede hablar de reactividad en cuanto a los vehículos y sus reacciones frente a semáforos y a otros agentes (evitando colisiones).

A continuación, se desglosa de manera más detallada cada uno de los agentes necesarios para la simulación de tráfico y sus características específicas.

Coche

Los agentes coche tienen como objetivo llegar a un destino asignado de manera aleatoria, evitando generar congestiones.

Capacidad Efectora: Cuentan únicamente con la capacidad de moverse hacia enfrente de manera directa o en diagonal, lo cual naturalmente modifica el entorno en el que se encuentra. Los agentes también pueden optar por la inacción, que también modifica el entorno para otros agentes.
Percepción: Pueden percibir al agente que se encuentre en la celda de enfrente, saber el estado de los semáforos (verde o rojo) y detectar obstáculos (edificios y destinos ajenos al propio). También cuentan con memoria para saber si la ruta actual los está llevando a una congestión.
Modos de Operación: El agente coche utiliza el algoritmo A* con la heurística de Distancia Euclidiana para buscar cuál es el camino más corto a su destino, para encontrar este camino se debe tomar en cuenta las direcciones de las calles.
Actuadores y Sensores: Los agentes interactúan con su entorno a través de sus capacidades de movimiento, y perciben su entorno mediante sensores frontales (detectan lo que hay en la siguiente celda en el camino calculado por A*).

Destinos

Los agentes destino buscan eliminar agentes de tipo coche que hayan llegado correctamente a sus destinos y contabilizarlos.

Capacidad Efectora: Únicamente pueden eliminar agentes cuyo destino sea propio.
Percepción: La celda en la que se encuentra el agente es la que determina si hay un vehículo en su posición.
Actuadores y Sensores: Los agentes interactúan con su entorno a través de la limpieza de vehículos.

Semáforos

Los agentes semáforo tienen como objetivo promover el flujo eficiente del tráfico, mediante un estado (rojo y verde), cambiando a verde cuando existan 2 autos enfilados o continuar operación normal en caso contrario.

Cabe mencionar que además orientan las calles subyacentes al semáforo en el primer paso de la simulación.

Capacidad Efectora: Los agentes de este tipo pueden modificar su ambiente al cambiar de estado para permitir o no permitir el flujo de tráfico.
Percepción: Perciben los vehículos en una vecindad de radio 4 que se encuentren en la misma dirección que ellos mismos.
Actuadores y Sensores: En el primer step, utilizan la información de su calle subyacente para orientarse. En pasos subsecuentes tienen sensores para detectar embotellamiento.

Agentes Obstáculos (Edificios)

Únicamente son objetos. No toman decisiones, no se mueven, ni perciben el ambiente.

Arquitecturas de subsunción

A continuación, se presenta la arquitectura de subsunción de cada uno de los agentes la cual corresponde con la implementación en código de la simulación. Se trata de una metodología de diseño robótico que enfatiza la importancia de comportamientos autónomos y distribuidos y ordena las acciones de menor a mayor prioridad a partir de eventos, condiciones y resultados.

Coches

Este tipo de agente efectúa acciones básicas como detectar un camino libre, pero también procesos más complejos como el manejo de interrupciones debido al tráfico o semáforos. Esta jerarquía evidencia la flexibilidad del sistema para adaptarse a cambios dinámicos, manteniendo un flujo de control claro y eficiente que asegura que los vehículos avancen hacia sus destinos.

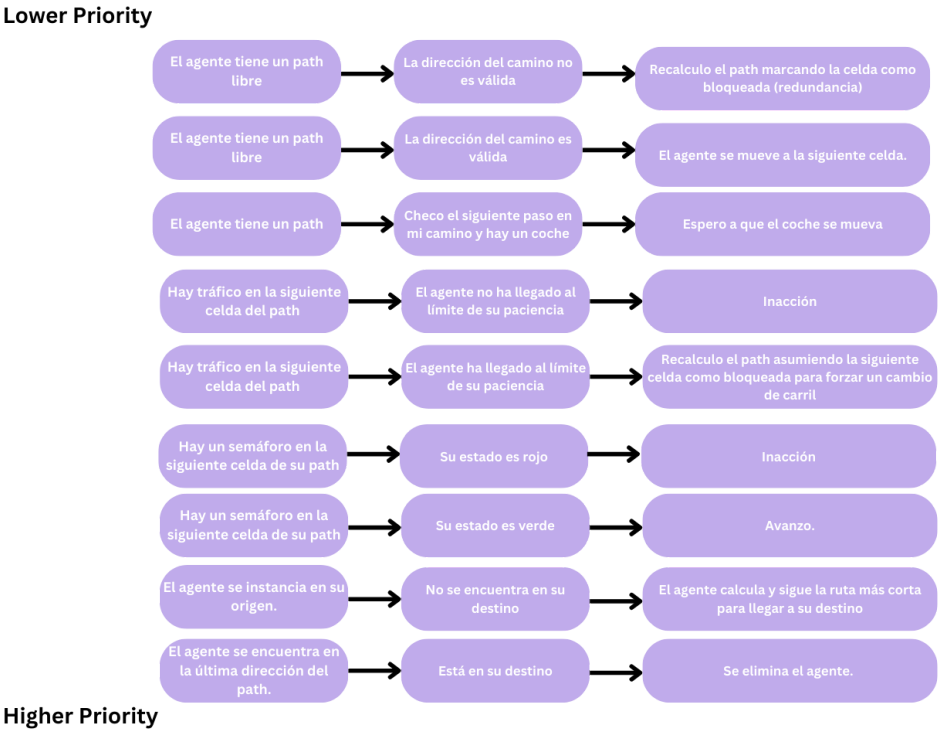

Imagen 1: Arquitectura de Subsunción de los agentes “Coche”

Semáforos

Para los agentes semáforo también se delinean los criterios y las condiciones bajo las cuales estos dispositivos cambian de estado y afectan el flujo vehicular. Aquí, la arquitectura de subsunción permite a los semáforos operar de manera autónoma y lógica, garantizando un tránsito fluido y seguro.

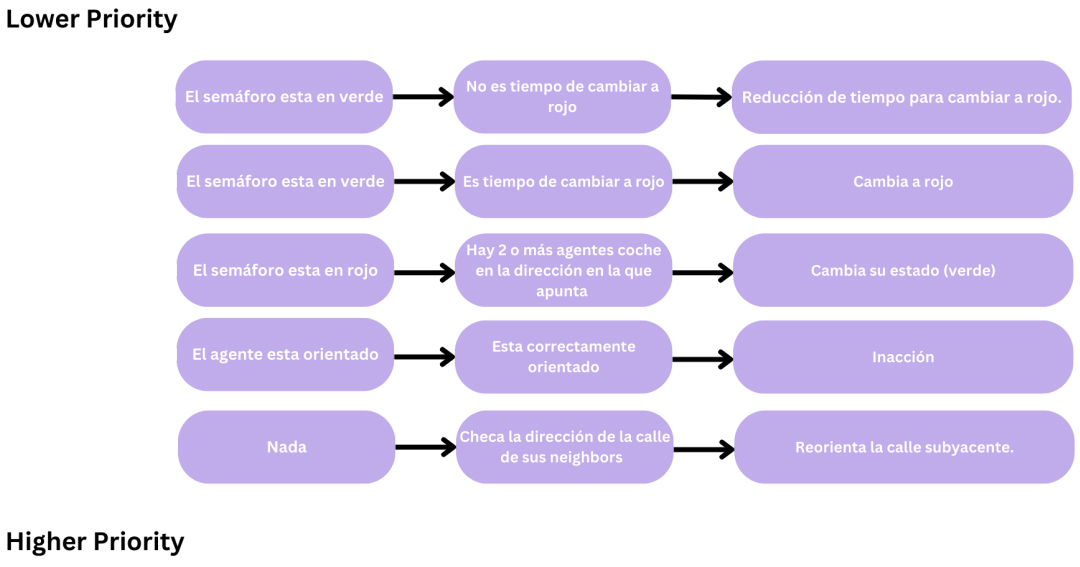

Imagen 2: Arquitectura de Subsunción de los agentes "Semáforo"

Destinos

Por último, se presenta una arquitectura más sencilla para los destinos que solo tienen un condicional en su lógica para mantener la simulación limpia y visualmente entendible.

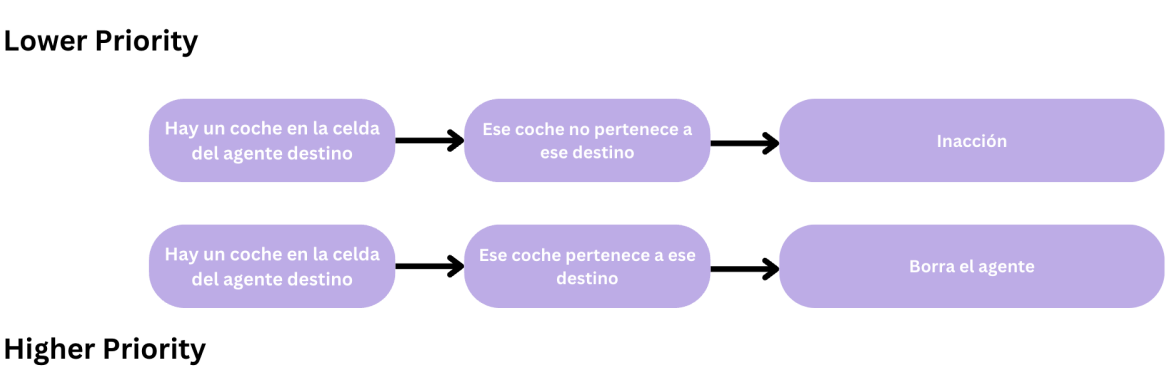

Imagen 3: Arquitectura de Subsunción de los agentes "Destino"

Características del ambiente

El ambiente en cuestión, es completamente inaccesible para los agentes coche, ya que, estos no tienen conocimiento completo del mapa en un momento dado; solo conocen las áreas exploradas y según su proactividad asumen o no asumen como obstáculo los embotellamientos. Esto implica que los agentes no son omniscientes y se asemeja a la realidad, dado que los humanos tampoco tienen conocimiento del sistema de tráfico en todo momento y, al igual que los agentes, solo conocen su entorno inmediato y las condiciones cambiantes. Además, el ambiente es no determinista debido a la incertidumbre asociada con la interacción entre los agentes, su cálculo de destinos y las variables humanas que introducen cierta aleatoriedad en el movimiento. Cabe mencionar que como los agentes semáforo son dinámicos en sus decisiones (dado que operan según los coches y los coches tienen aleatoriedad), también contribuyen a que el ambiente sea no determinista.

En cuanto a su estructura temporal, el ambiente es no episódico. Pese a que la temporalidad se mide en episodios discretos, los agentes planifican acciones futuras basándose en experiencias previas; esto sucede tanto para el contador dinámico de los semáforos como para el historial de posiciones propias que guardan los agentes vehículo.

El ambiente es dinámico, considerando que durante el tiempo en el que los agentes están deliberando, el estado de su ambiente podría cambiar (ya sea un cambio de luz en semáforo o un movimiento de otro agente). Cabe mencionar que esto está implementado mediante la activación aleatoria de los agentes, intentando emular situaciones más realistas. Una limitante en este sentido es que los agentes suelen optar por inacción frente a estos escenarios, para simplificar su proceso de decisión.

Finalmente, se puede considerar el ambiente como discreto, caracterizado por una retícula de tamaño predefinido y un conjunto limitado de percepciones posibles en un momento dado.

Conclusiones

La simulación desarrollada es una herramienta poderosa para enfrentar los desafíos de la movilidad urbana en México. Al integrar una representación 3D (Unity) y un modelo de simulación multiagentes, se logra una comprensión más profunda de los patrones de tráfico y se proporcionan insights valiosos para la planificación urbana. La capacidad de simular diferentes escenarios y estrategias de gestión de tráfico nos permitió evaluar el impacto de ellas en la eficiencia de los agentes y el ambiente.

Para probar la simulación y conocer su desempeño, primero se probó con el parámetro especificado en la descripción del reto: “Se van agregando vehículos al ambiente cada 10 episodios en los puntos de inicio (4 esquinas) de la simulación”. Bajo estas circunstancias el modelo se comporta extremadamente bien, desalojando todos los vehículos a tiempo, sin colisiones y con muy baja probabilidad de embotellamientos, según nuestras pruebas. Sin embargo, para evaluar el modelo en condiciones subóptimas, se realizaron tres pruebas con distintas características:

Características de la prueba	**Cantidad de agentes en su destino**	**Episodios a los que se llegó**
- Se deben agregar 4 agentes (uno por cada esquina del mapa) cada 3 episodios. - Límite de 1000 episodios - Se realizó desde Unity	132	254
- Se deben agregar 4 agentes (uno por cada esquina del mapa) cada 3 episodios. - Límite de 1000 episodios - Se realizó desde Mesa	546	1000
- Se deben agregar 4 agentes (uno por cada esquina del mapa) cada 2 episodios. - Límite de 1000 episodios - Se realizó desde Mesa	863	1000

En suma, es factible decir que el modelo cumple su objetivo, incluso bajo condiciones más retadoras.

Dado lo anterior, fue más sencillo identificar áreas de mejora respecto a la simulación, muchas de las cuales fueron resueltas (como mejorar flujos, implementando semáforos

inteligentes e implementando tolerancias variables en los agentes). Sin embargo, también se encontraron extensiones que fueron implementadas por cuestiones temporales.

Posibles extensiones

Por un lado, los agentes semáforo podrían ser más inteligentes, funcionando en pares en lugar de funcionar de manera individual y comunicándose con el resto de semáforos en la intersección con los datos de la cantidad de coches que se encuentran en las calles aledañas. Por otro lado, estamos conscientes de que lo idóneo sería que los semáforos se orienten durante la inicialización del modelo y no durante la primera activación (llamada al *step()*) de los agentes.

Otro punto importante podría ser la comunicación entre agentes y la implementación de otras variables que permitan controlar su comportamiento. En términos de comunicación, se implementó preliminarmente el uso de direccionales y sería lógico esperar mejores resultados en este caso, sin embargo, esto no acabó en el código final. Por último, consideramos que sería fructífero explorar preferencias (como dar paso al de la derecha en un cruce) y reglas de tránsito para mejorar el rendimiento de la simulación y hacerla aún más realista.

Finalmente, se propone la optimización del código existente para mejorar su rendimiento. Los puntos cruciales serían los siguientes:

- Reducción de Complejidad en get neighbors: La función get neighbors podría ser optimizada. Actualmente, realiza múltiples iteraciones sobre los contenidos de las celdas y verifica varios tipos de direcciones. Podría ser más eficiente si se simplifican estas operaciones. Por ejemplo, en lugar de recuperar y filtrar el contenido de cada celda vecina en cada llamada, se podría considerar un enfoque donde primero se determinen las celdas vecinas válidas y luego se realice una sola consulta para obtener sus contenidos. Esto podría implementarse como un grafo dirigido en lugar de una matriz con vecinos direccionados.

- Optimización del Algoritmo A*: El algoritmo A* se ejecuta en cada llamada a find_path. Podría ser útil implementar una memoria interna (memoization) en el código para almacenar rutas ya calculadas, especialmente ya que el ambiente de la simulación no cambia drásticamente entre episodios consecutivos.

- Eficiencia en la gestión de Agentes: Las funciones step de los agentes realizan varias comprobaciones y operaciones en cada paso. Revisar y optimizar estas comprobaciones podría mejorar el rendimiento. Por ejemplo, en Car.step, se podría verificar primero si el agente está atascado antes de realizar otras comprobaciones.

- Uso de implementaciones predeterminadas: Por obvias razones, la implementación manual tanto de A* (para movimiento de agentes), como de matrices de transformación (para el movimiento en Unity) no es ideal, puesto que se podrían utilizar métodos integrados como *transform* o librerías como *pathfinding* para asegurar un mejor desempeño, sin embargo, vimos un espacio de aprendizaje en la implementación manual, por lo que optamos por omitir esto.