Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Unidad de formación:
Multiagentes y gráficas computacionales

Fecha de entrega:
15 de Marzo 2024

Pablo Barriuso A01763259

Propuesta al problema planteado

La propuesta se centra en la creación de un sistema multiagente dinámico y adaptable que gestione una intersección de tráfico compleja, compuesta por 4 vías, cada una con un carril para cada sentido, y 4 semáforos que regulan el flujo vehicular. Este sistema debe asegurar no solo la fluidez del tráfico sino también prevenir accidentes, optimizando los tiempos de espera y adaptándose a las variaciones en el flujo de vehículos. La simulación debe reflejar situaciones realistas de tráfico. El objetivo es demostrar cómo un enfoque basado en agentes puede ofrecer soluciones eficientes a problemas de tráfico urbano y que podrían aplicarse en la vida real.

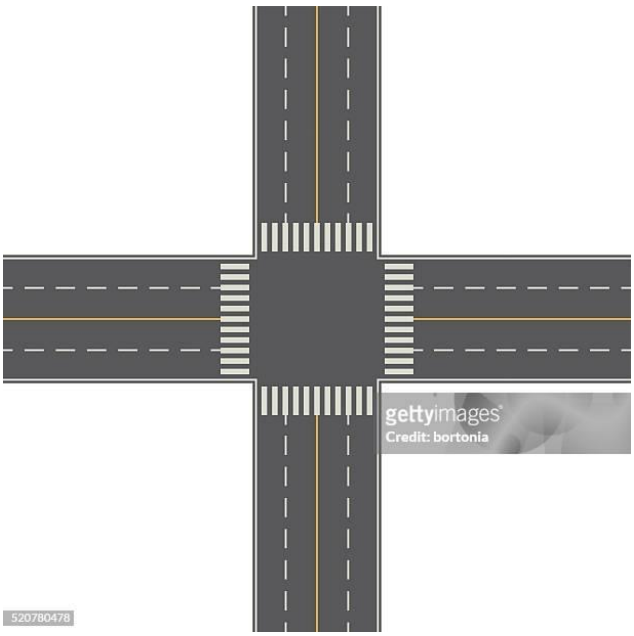

Ilustración 1- Intersección de 4 vías

Este sistema multiagente cuenta con dos principales agentes. Los semáforos y los vehículos.

- Vehículos: actúan como agentes individuales con la capacidad de moverse y tomar decisiones autónomas basadas en el estado de los semáforos y en otros vehículos cercanos.
- Semáforos: agentes encargados de coordinar el tráfico y asegurar un flujo vehicular eficiente y seguro. Se comunican entre sí el numero de vehículos en su carril para coordinarse correctamente.

Análisis de la solución desarrollada

¿Por qué seleccionaron el modelo multiagentes utilizado?

Como se ha descrito anteriormente, contamos con dos agentes principales en nuestra simulación del tráfico.

En primer lugar, tenemos los **vehículos**. Hemos diseñado los vehículos como **agentes reactivos**, esto significa que no tienen un estado interno con información propia o del ambiente (no tienen memoria). Además, tienen unas acciones predeterminadas para distintas situaciones. Estos agentes siguen una serie de reglas a la hora de actuar:

- Se detienen cuando tienen un semáforo en rojo cerca.
- Se detienen cuando tienen un coche enfrente a poca distancia.
- En la intersección toman cualquier dirección.

Para esto, los agentes hacen uso de distintas funciones y métodos. Las dos funciones más relevantes son la función ***see*** y la función ***action*.**

- Función see(): ve los coches que tienen engrente y el semáforo de su carril
- Función action(): se mueve en su dirección a no ser que haya un coche cercano o el semáforo esté en rojo.

Existen más funciones características de este agente pero que no tienen tanta relevancia en este análisis.

Se ha escogido este diseño de agente debido a dos razones fundamentales. La primera es la sencillez a la hora de implementar el agente en el entorno de Python. Los agentes reactivos son más simples y no requieren mucho trabajo más allá de definir correctamente las reglas que siguen. El segundo motivo es que no es necesaria más complejidad para un agente como éste. A un nivel muy básico, un coche se mueve, gira y se detiene.

En segundo lugar, tenemos los **semáforos**. Los semáforos en nuestro sistema multiagente son diseñados como agentes híbridos, lo que significa que incorporan características de agentes reactivos y algunas capacidades de los agentes con estado. Esta hibridación les permite responder de manera efectiva y eficiente a los cambios dinámicos en el entorno, como la variación en el flujo de vehículos, al tiempo que mantienen una memoria limitada y la capacidad de comunicación con otros agentes.

Los semáforos se caracterizan por las siguientes funcionalidades y comportamientos clave:

- Mecanismo de Subasta: los semáforos participan en un mecanismo de subasta que determina cuál de ellos debe activar su luz verde en función de

la cantidad de vehículos que esperan en sus respectivas vías. Este mecanismo garantiza que el semáforo que atiende al carril con mayor demanda vehicular obtenga la prioridad, optimizando así el flujo general del tráfico.

- Comunicación: se comunican entre sí mediante mensajes de tipo 'inform', donde cada semáforo reporta la cantidad de vehículos que detecta en su carril.

Las funciones más relevantes de los semáforos son:

- Función see(): permite a cada semáforo detectar la cantidad de vehículos que esperan en su carril, lo cual es un dato crucial para participar efectivamente en la subasta.
- Función action(): una vez finalizada la subasta, el semáforo actúa en consecuencia, activando su luz verde si resulta ganador o permaneciendo en rojo en caso contrario.

Este diseño de agente semáforo se ha elegido por varias razones clave. En primer lugar, por la capacidad de adaptarse a las condiciones cambiantes del tráfico en tiempo real, esto es vital para evitar congestionamientos y garantizar la seguridad. Otra razón clave es la mezcla entre autonomía y coordinación. Los semáforos operan de manera autónoma en cuanto a la detección de vehículos, pero también se coordinan para tomar decisiones que benefician al sistema en su conjunto.

¿Cuáles fueron las variables que se tomaron al momento de tomar la decisión?

A la hora de diseñar el sistema multiagente se consideraron varias variables clave que influyen directamente en la dinámica del tráfico y el desempeño en general de la simulación.

- Densidad de tráfico: esta es una variable muy importante ya que con ella podemos simular intersecciones muy concurridas o intersecciones sencillas. En el entorno de Python podemos encontrar referencias a esta variable como n_vehicles, siendo ésta un parámetro de la simulación.
- Intervalo en verde de los semáforos: otra variable muy representativa es el tiempo que un semáforo tarda en ponerse en rojo una vez está en verde. En nuestra simulación este valor está fijado a 5 steps.

Hay otras muchas variables que influyen en esta simulación pero éstas son las más importantes.

¿Cuál es la interacción de esas variables con respecto al resultado de la simulación?

La densidad de tráfico afecta directamente a la toma de decisiones de los semáforos. Un número alto de vehículos implica que ciertos carriles estarán más concurridos que otros. Esto hará que los semáforos de esos carriles tengan más peso en la subasta y que finalmente se pongan en verde.

El tiempo que un semáforo permanece en verde (fijado a 5 steps) está diseñado para que dé tiempo a pasar al suficiente número de vehículos antes de iniciar una nueva subasta. Si el tiempo en verde es demasiado corto, puede no ser suficiente para permitir que muchos de los vehículos esperando pasen, creando cuellos de botella. Por otro lado, si es demasiado largo, podría resultar en tiempos de espera innecesarios en otras direcciones, especialmente en situaciones de tráfico asimétrico.

¿Por qué seleccionaron el diseño gráfico presentado?

El diseño gráfico seleccionado para la simulación fue intencionadamente simple y estilizado en un estilo low poly para facilitar la identificación clara de cada elemento relevante del sistema. Este enfoque minimalista cumple con varios objetivos clave:

- Claridad Visual: un diseño low poly ayuda a evitar la sobrecarga visual y asegura que se pueda reconocer rápidamente y sin ambigüedad los vehículos, los semáforos y las vías. La simplicidad del diseño asegura que el foco se mantenga en la mecánica de la simulación y en la interacción de los agentes.
- Rendimiento: al evitar detalles excesivos en los modelos podemos correr la simulación de manera fluida. Esto es particularmente ventajoso para simulaciones que pueden escalar a un número significativo de agentes.
- Universalidad: este estilo puede ser interpretado y comprendido fácilmente por muchas personas, independientemente de su experiencia previa con simulaciones o videojuegos, lo que hace que la simulación sea accesible para un público más amplio.

¿Cuáles son las ventajas que encuentras en la solución final presentada?

La principal ventaja de esta solución es la sencillez, lo que implica varios beneficios:

- Facilidad de comprensión: los agentes tienen objetivos claros y actúan de manera intuitiva. La implementación es sencilla a la vez que efectiva y es fácilmente entendible.
- Rendimiento: la simulación es representada con modelos sencillos lo que hace que el rendimiento sea mayor.
- Enfoque en aspectos clave: una solución sencilla permite centrarse en los aspectos más importantes del sistema, como el flujo de tráfico, en lugar de distracciones innecesarias que podrían surgir de modelos más complejos
- Facilidad de modificación: teniendo una base sólida y sencilla, implementar mejoras no debería ser muy costoso.

¿Cuáles son las desventajas que encuentras en la solución final presentada?

A pesar de las numerosas ventajas de la solución final basada en la simplicidad, hay algunas desventajas potenciales a considerar:

- Falta de detalle: los agentes reactivos, debido a su diseño simple pueden no capturar la complejidad y la variabilidad del comportamiento humano real en la conducción.
- Diseño por mejorar: pese a que el diseño planteado es efectivo y robusto puede que no sea el óptimo dejando mucho margen de mejora para el futuro.

¿Qué modificaciones podrías hacer para reducir o eliminar las desventajas mencionadas?

- Incrementar la complejidad de los agentes: para capturar mejor la variabilidad del comportamiento humano, los vehículos podrían mejorarse con modelos más complejos y sofisticados que incluyan factores psicológicos de los conductores, como la impaciencia ante un semáforo en rojo o que alguien se pueda saltar el semáforo en rojo.
- Mejorar la representación de la simulación: se podría implementar un pequeño menú en Unity donde se pueda “jugar” con los parámetros de la simulación.
- Optimizar el modelo: se podría analizar el valor óptimo de los distintos parámetros para una mejor fluidez en el tráfico.

Reflexión sobre el proceso de aprendizaje

Tenía las expectativas muy altas en esta asignatura ya que mezclaba dos componentes de la rama de informática que me gustan mucho como es Unity y como es la inteligencia artificial. Sin embargo, creo que la asignatura se ha quedado un poco corta. Esto podría ser por falta de tiempo o por falta de organización.

En la parte de gráficas computacionales creo que le dimos muchas vueltas a las matemáticas detrás de los motores gráficos como puede ser Unity. Las matemáticas son fundamentales para que estos sistemas funcionen, pero creo que demoramos mucho tiempo en esto sin luego aplicarlo realmente en el reto. Me esperaba ir más al grano con Unity y ver alguna característica del motor más avanzada y sofisticada.

La parte de agentes me sorprendió, ya que era algo de lo que no había oído hablar como tal (por dar contexto, soy un alumno foráneo de cuarto año que ha venido de intercambio a terminar la carrera). Sin embargo, creo que estos conceptos tan abstractos es mejor situarlos con ejemplos concretos según son explicados teóricamente. La primera vez que vimos algo de código ya había muchos conceptos explicados y se podía hacer un poco difícil seguir la clase. En general creo que falta acompañar los conceptos teóricos con algo más de práctica.

Por último, quiero decir que la experiencia de trabajar en equipo no me ha gustado mucho. Si bien trabajar en equipo es algo muy importante de cara al futuro creo que ha faltado un poco más de reconocimiento al trabajo individual de cada uno y de lo que aporta al equipo. Siento que he trabajado de más y que no se ha valorado suficientemente mi esfuerzo, compromiso y dedicación al proyecto.