Análisis y Reflexión Individual - Reto Movilidad Urbana

Juan Daniel Rodríguez Oropeza A01411625

Monterrey, Nuevo León, México

Diciembre 03, 2022

Ing. César Raúl García Jacas

Ing. Sebastián Ulises Adán Saldívar

Instituto Tecnológico y de Estudios Superiores de Monterrey

Modelación de Sistemas Multiagentes con Gráficas Computacionales (Gpo 301)

Un análisis de la solución desarrollada, deberás enfocarte en contestar estas preguntas:

¿Por qué seleccionaron el modelo multiagentes utilizado?

El modelo que utilizamos incluye al tipo de Agente de Reflejos Simples, ya que el agente carro y los demás agentes no evolucionan, solamente siguen las reglas y condiciones establecidas. El agente solamente toma en cuenta el entorno que puede analizar en su momento.

Debido a que hay varios agentes, el tipo de matriz (grid) es Multigrid para que varios tipos de agente convivan en la misma celda, ya que los agentes de Camino (Road), Left, Right, y el de la Intersección (FourWay) representan un espacio de la matriz en donde se moverán los agentes autos, por lo que un carro y cualquier otro tipo de agente estarán en la misma celda. Aún así, se establece que ningún agente auto puede estar en la misma celda que otro, para que de esta forma los carros no colisionen.

El tipo de planificador implementado fue el simultáneo debido a que todos estos tipos de agentes conviven en el mismo entorno al mismo tiempo, principalmente para que los carros sepan cuándo y dónde detenerse, así como avanzar, tomando en consideración las posiciones de los demás y sus estados para finalmente tomar una decisión. Por ejemplo, detenerse si el semáforo rojo está justo en frente de ellos, o si hay otro carro en frente de ellos.

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

Se tomó en cuenta la posición y dirección del auto, así como lo que rodea al entorno, para que pueda moverse o detenerse dependiendo de la posición y estado de los demás agentes (principalmente autos y la intersección, esta última también representa a los semáforos).

También es importante considerar el valor de las celdas, para que el agente auto pueda identificar si el camino está bloqueado (0), se puede transitar de forma normal, generalmente hacia enfrente, (1), se puede girar a la derecha (2), a la izquierda (3), o si es la intersección (4), que es donde se ubican los semáforos.

Tomamos en cuenta que principalmente lo que tenía que hacer el auto es cruzar la intersección, nos enfocamos primero en que cruzase hacia enfrente. Dejamos como segundo plano el que pudiera dar vuelta en la intersección en caso de que tuviéramos tiempo de desarrollarlo, pero aún así diseñamos el mapa para que el auto tuviera que dar vuelta tanto a la izquierda como a la derecha después de cruzar la intersección.

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

Primeramente, la velocidad depende del estado de movimiento del vehículo, si está detenido, la velocidad es 0, si está en movimiento, la velocidad aumenta. A su vez el estado de movimiento del vehículo está relacionado con su razón de detenimiento (en caso de que así sea), ya sea porque el semáforo en color rojo está justo en frente, o porque hay otro carro en frente.

El semáforo tiene estos estados:

- Verde: Para que los autos puedan avanzar, y por ende, cambien de posición y aumente su velocidad.
- Amarillo: Solo es un detalle visual para indicarle al usuario que pronto cambiará a rojo, a nivel de lógica en el código, actual igual que el verde, de hecho en la parte de Mesa no está presente, solamente en Unity.
- Rojo: Indica a los autos que deben frenar, por lo que la velocidad es 0 y los autos se detienen y se quedan en fila. Para esto es el estado de la razón porque se detienen, para aprender a diferenciar y distinguir en qué lugar detenerse y que no vayan a colisionar un carro con otro y no se haga una carambola, aquí entra en juego la cantidad de agentes autos.

Al final, la intersección solamente permite al auto cruzar hacia adelante por cuestiones de tiempo y complejidad, además de que el espacio que se considera para que el auto se pueda mover por la matriz representa una calle continua (que forma un circuito) de un sólo sentido.

El resultado es que los automóviles pueden transitar con fluidez respetando el espacio y posición de otros carros, así como deteniéndose si cuando hay un semáforo en rojo justo en frente de ellos, y que continúen avanzando cuando el semáforo esté en verde.

¿Por qué seleccionaron el diseño gráfico presentado?

Usamos Unity para representar la simulación de Mesa en un entorno tridimensional, los modelos 3D fueron obtenidos de la Asset Store de Unity para enfocar más nuestro tiempo en la lógica de la simulación más que en lo visual, ya que los modelos no tienen que ser 100% realistas.

Además, el mapa lo hicimos en forma de circuito para que haya continuidad en la posición de los autos.

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

Los autos pueden cruzar la intersección de manera ordenada y se puede visualizar como los autos respetan las posiciones.

Es portable, se puede adaptar a cualquier escenario, siempre y cuando las calles sean de un sentido. Se tendría que cambiar el escenario en Unity y en Mesa para probarlo.

Además, otra ventaja de nuestra solución es que no es necesario descargar el código de Python para ejecutarlo, porque la API ya está desplegada en línea, por lo que solo es necesario el Unity Package para ejecutarlo.

Con este proyecto, al representar visualmente la movilidad del tráfico, sería factible (si se sigue trabajando en mejorar este proyecto presentado) en un futuro solucionar la sincronización de los semáforos de acuerdo a la velocidad permitida de los carros para disminuir el congestionamiento vehicular, así como la contaminación y el estrés de los conductores.

¿Cuáles son las desventajas que existen en la solución presentada?

Solamente hay un carril para los autos, aunque el comportamiento sería muy similar con dos carriles, sería interesante ver cómo los autos se cambian de carril para rebasar a otros carros. Esto va también de mano con que la velocidad es constante cuando el auto puede avanzar. Esto tiene que ver con que en Mesa no está definido el comportamiento de rebasar, ya que solamente van en fila los autos, y que también en Unity el carro se mueve con el método de Transform, el cual no tiene físicas, por lo que tampoco tiene aceleración, haciendo que el auto se comporte más de forma arcade y que no pueda desacelerar (frena de golpe) ni derrapar.

Aunque también al tener dos carriles se hubiera podido probar con tener 4 semáforos en la intersección en lugar de dos, permitiendo que los autos crucen en cualquiera de los dos sentidos, así como la posibilidad de dar vuelta en la intersección.

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

Primeramente, para corregir el movimiento, habría que hacer que los autos se muevan mediante Wheel Colliders para que los carros tengan aceleración, otros aspectos de física.

Para que se puedan usar dos carriles, hay dos casos, para que ambos carriles sean de un sentido, solamente sería agregar en la matriz celdas a lado con su valor respectivo (ya sea 1 para camino normal, 2 para derecha, 3 para izquierda, y 4 para intersección, que es donde están los semáforos). En este caso también se podrá aprovechar para que un agente auto detecte si hay otro auto a su lado y también si la celda diagonal (la que está al frente del otro carro) está desocupada (para ello se utilizaría *Moore*) y si hay calle para que sea posible que rebase a dicho auto.

En caso de que el otro carril sea para que la calle se convierta en doble sentido, tendría que cambiar el valor de todas las celdas para diferenciar los tipos de celdas con su respectivo sentido, también se tomaría en cuenta la dirección del carro para que los vehículos no puedan ir en sentido contrario.

Y con este caso, se pudiera incorporar más semáforos en la intersección, los cuales tendrían que estar coordinados con los demás para que el tráfico fluya adecuadamente.

Y para que den vuelta en la intersección, se tendría que implementar todo lo anteriormente mencionado y cambiar la distribución del mapa en Unity y la matriz en Mesa para que sea factible. La intersección se representaría por un grupo de celdas con distintos valores para diferenciar en qué parte específicamente se puede dar vuelta en la intersección, y en cual se debería avanzar recto el carro.

Una reflexión sobre tu proceso de aprendizaje. Para ello, revisa el documento original que contiene tus expectativas al inicio del bloque y compáralo con las experiencias que viviste a lo largo de estas semanas.

Antes de que comenzara el curso, vi el resultado del reto de esta materia en una historia de Instagram de uno de mis compañeros de la carrera, era sobre cómo corría la simulación de tráfico en Unity y había quedado anonadado. Desde ese momento me emocioné por cursar esta Unidad de Formación.

Mi expectativa al comienzo del curso era aprender más sobre Unity, ya que el semestre anterior vimos Unity en 2D, así que para esta ocasión estaba esperando ver sobre el 3D, sobre todo porque al ser en 3D están involucrados otros aspectos como la física.

Al ver el comportamiento de los autos en esa historia, me maravillé de la coordinación de los autos entre ellos junto con el semáforo. Me emocioné también porque ya quería ver sobre Inteligencia Artificial para saber cómo las entidades se coordinan y entienden su ambiente para saber cómo comportarse.

Ahora que se terminó el curso, me quedo muy satisfecho con todo lo aprendido sobre Unity, ya que no solo se cumplieron mis expectativas, sino que también profundizamos más en como programar y administrar los GameObjects de forma más efectiva, considerando también el rendimiento del juego o simulación al ejecutarse. También quiero hacer mención a que las actividades ayudaron a comprender los conceptos que aplicamos en el reto, principalmente porque para las actividades tuvimos que explicar detalladamente lo que aprendimos e hicimos.

Por la parte de Mesa, a pesar de haber sido difícil en un inicio, donde para poder progresar tuve que hacer muchas pruebas y leer la documentación oficial, estoy satisfecho por haber aprendido sobre los agentes, los diferentes tipos de planificadores y matrices para que éstos convivan y actúen, y que a su vez los datos que se manipulan puedan ser reconocidos por Unity a través de una API.

Trabajar con Mesa permite que desarrolles tu lógica y pienses cómo debería actuar uno o varios agentes en un escenario real, lo cual te permite simular y predecir cómo sería el resultado en la vida real en base a la simulaciones hechas con Mesa, las cuales posteriormente se pueden visualizar de forma tridimensional en Unity.

La colaboración, comunicación, y trabajo en equipo fue agradable y enriquecedor para que juntos hayamos sacado a flote este proyecto. Quiero hacer mención al apoyo, disponibilidad, dedicación, y diálogo que tuve con mi equipo durante el intercambio de ideas mientras trabajamos en el reto.

Sobre todo reconocer la comunicación que hubo entre nosotros para que el mapa de Unity y la matriz de Mesa fuera lo más compatible y parecido posible. Al principio teníamos muchas más ambiciones, pero tuvimos que ser realistas y acoplarnos a la fecha de entrega, además de que el mapa de Unity antes era muy grande y con muchas calles, por lo que el desarrollo en Mesa de la matriz y el comportamiento de los agentes iba ser más complejo.

También quiero reconocer el apoyo de los profesores con sus consejos y sugerencias para resolver nuestras dudas.