Reto: Movilidad Urbana

Daniel Isaac Ruiz Cruz - A01652366
a0652366@tec.mx

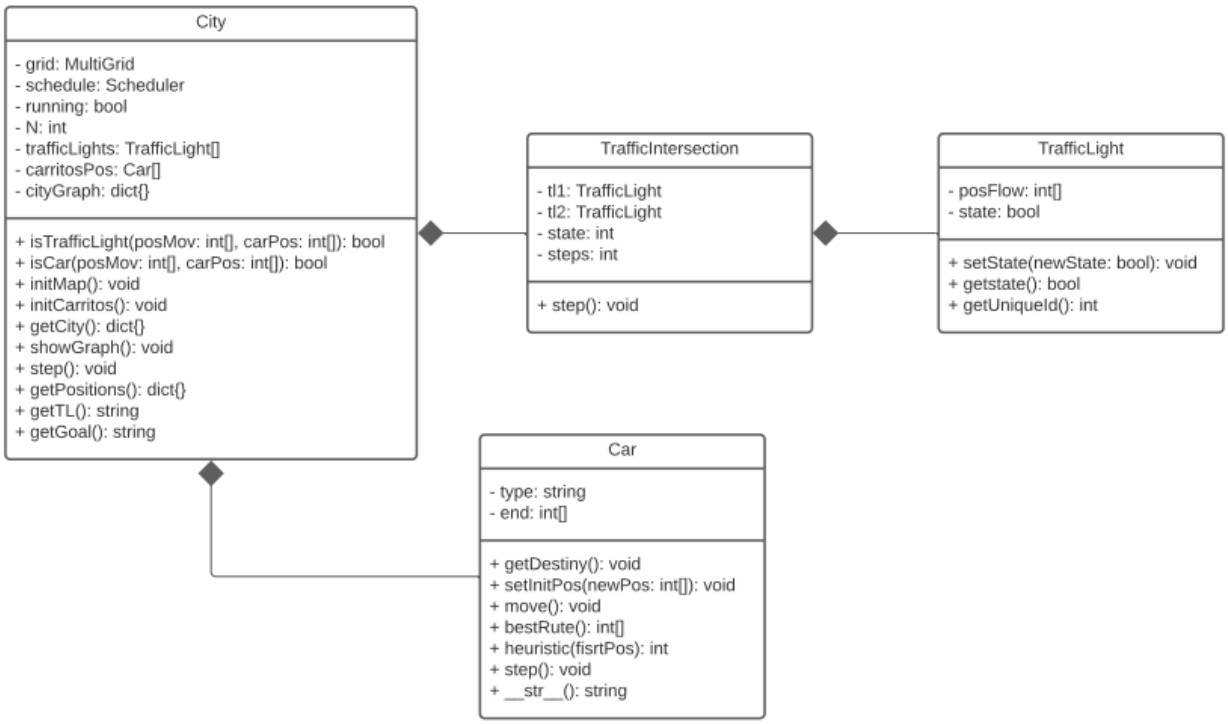
Modelación de sistemas multiagentes con gráficas computacionales

Profesores:
Sergio Ruiz Loza
David Christopher Balderas Silva

Fecha de entrega: 1 de diciembre del 2021

Parte 1: sistemas multiagentes

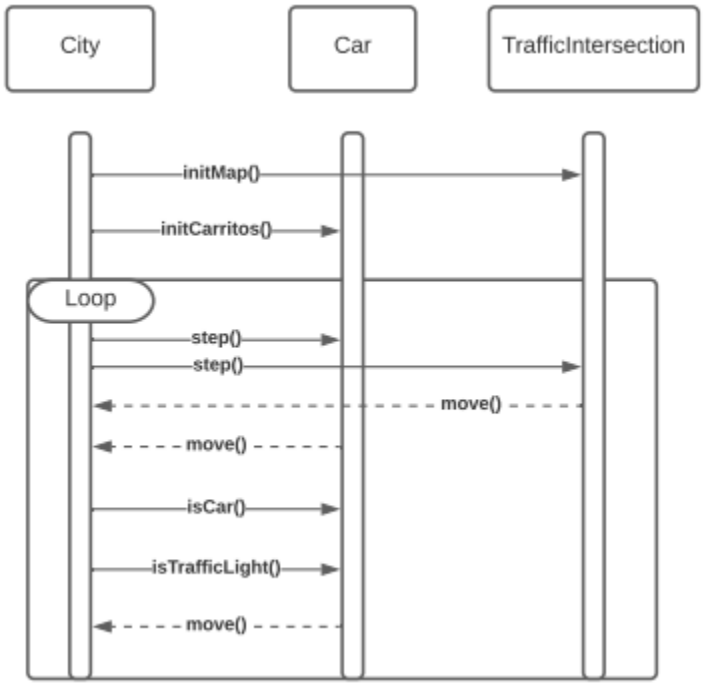
Primeramente, seleccionamos un modelo que pudiera representar embotellamientos en las calles para que nuestro agente pudiera implementar el algoritmo de A*, de tal manera que tomara rutas alternas que lo hicieran llegar más rápido a su destino. Por esta razón, tuvimos que declarar el ambiente y los agentes de esta forma:



Este diseño fue hecho de tal manera que el agente “Car” pudiera comunicarse con el agente “City”, el cual cuenta con la información de los semáforos “TrafficLight” y las posiciones de los demás agentes. Con esta conexión cada agente podía determinar el trafico en la ruta y si podía o no avanzar dependiendo si había un automóvil enfrente o la luz del semáforo estaba en rojo. Asimismo, al momento de diseñarlo, con base en la problemática del tráfico y la decisión de tomar nuevas rutas, fue necesario tomar en consideración como es que el carro podría determinar el trafico en la ruta, como podía identificar que había un carro enfrente para evitar colisiones y como podía visualizar el estado del semáforo para detenerse o avanzar. Es por ello que al final se tomó la decisión de implementar un modelo con estas características, las cuales le brindaban al agente toda la información necesaria para movilizarse dentro del ambiente.

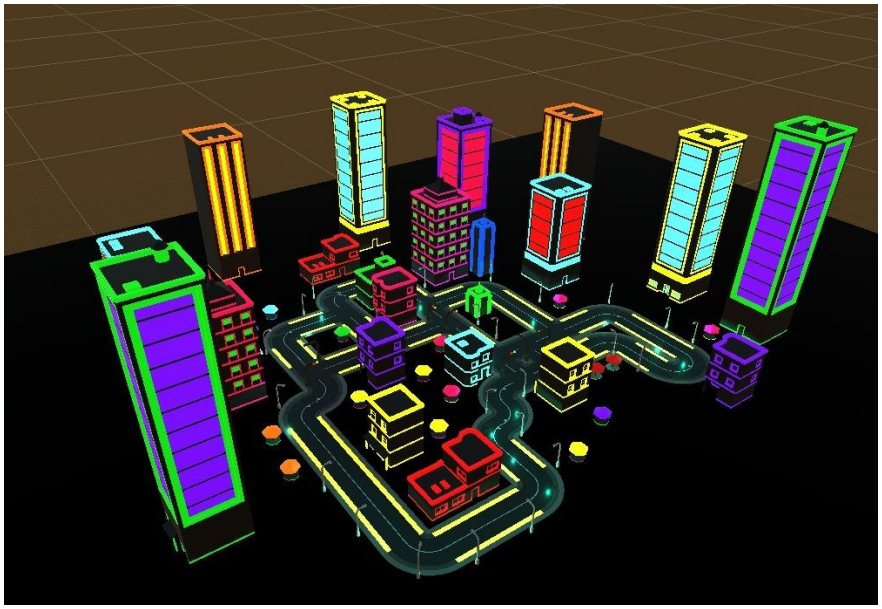
Por otro lado, hablando específicamente de las variables o condiciones antes mencionadas, fue de gran importancia el poder generar métodos que hicieran que la comunicación entre ambiente y agente fueran de la mejor manera para poder representar

la solución a la problemática de manera óptima. Esta interacción está representada por el siguiente diagrama de protocolos de interacción:



En este diagrama podemos observar que el ambiente “City” es el que crea a los agentes “Car” y “TrafficIntersection”, para después empezar a movilizar a los agentes “Car” dentro de él. Posteriormente, el ambiente le ordena a los agentes a moverse por lo que cada uno ejecuta el método “move()”. En el caso del “TrafficIntersection”, la función “move()” cambia el estado de los semáforos, es decir, cambia su color de rojo a verde o viceversa. Por otro lado, en el caso del agente “Car” este requiere pedirle información al ambiente acerca del estado de los semáforos en las intersecciones por las que va a pasar y, también, identificar si es que tiene un automóvil enfrente de él. Con esta información el agente puede ejecutar el algoritmo de búsqueda para encontrar la ruta más rápida a su destino. Este proceso se repite infinitamente, puesto que cuando un agente llega a su destino, este escoge uno nuevo automáticamente y se empieza a movilizar hacia esa posición.

Finalmente, para el desarrollo visual de la implementación del modelo, se decidió implementar un diseño neón futurista con inspiración en una ciudad estilo “cyberpunk”. Esto se decidió debido a que se quería poner en practica los conocimientos adquiridos durante el semestre en materia de texturas, animación y luces. Este estilo nos daba la oportunidad de jugar con las texturas de los autos, edificios y semáforos, debido a los colores tan vibrantes y las fuentes de iluminación. En este ultimo apartado, el hecho de que la escena estuviera de noche, hacia que fuera mas vistoso el tema de la iluminación por parte de los automóviles y los semáforos.



Como se logra ver en la segunda imagen, el tema neón hace que podamos jugar con la iluminación y sobras dentro de la escena, lo permite que realicemos reflexiones de luz como en este ejemplo.

Parte 2: ventajas y desventajas

Las ventajas que se pueden encontrar en nuestra solución planteada es la fácil interacción que llegan a tener los agentes con el modelo o ambiente. Al tener representado el ambiente como un grafo dirigido, era muy sencillo visualizar le movimiento del agente a través del ambiente, así como identificar los nodos ocupados como otros agentes. Esto ultimo ayudo a evitar colisiones e implementar los algoritmos de decisión para que el agente tomara la ruta menos congestionada. Asimismo, este mismo diseño de grafos ayuda a que la representación visual en el ambiente grafico de Unity sea directa y eficiente debido a que las coordenadas representadas en el ambiente son las mismas que están declaradas en Unity.

Las desventajas que encontramos en nuestra solución son en relación con la funcionalidad de los semáforos y del algoritmo de búsqueda implementado. Esto se debe a que los semáforos no actúan de manera inteligente, su comportamiento es un ciclo definido como lo hay en la vida real por lo que llega a ocurrir que un semáforo se encuentra en rojo sin ningún automóvil que cruce. Por otro lado, el algoritmo que implementamos, aunque hace eficiente el desarrollo y la selección de ruta, puede llegar a tener un problema cíclico en ciertas situaciones específicas. Esto repercute en la solución, puesto que el agente únicamente da vueltas en una ruta especifica sin llegar a su destino. Por último, otro problema es que las calles no pueden ser bidireccionales con esta implementación, lo que reduce el porcentaje de realismo que se le puede inyectar al modelo. Al tener todas las calles con una sola dirección, es necesario tener un ambiente bien diseñado, debido a que, aunque los agentes siempre optan por la ruta mas corta, si solo hay una vía por donde llegar al destino el agente tendrá que pasar por un embotellamiento.

Parte 3: próximas versiones

Para las próximas versiones de este programa es necesario corregir e implementar un nuevo modelo que solucione las desventajas planteadas anteriormente. Para el caso de los semáforos, es necesario implementar un algoritmo que identifique el trafico en su flujo, tal como lo hace el agente para identificar la ruta menos congestionada. Con esto se lograría tener un semáforo inteligente que también ayude a mejorar el flujo del trafico y todo el ambiente fluya de la manera mas optima junto con los algoritmos de A* por parte del agente. Asimismo, el modelo se puede ampliar de tal manera que pueda haber un grafo, si unidireccional, pero con la ventaja de que ahora se pueden seleccionar rutas bidireccionales de manera visual y de implementación. Finalmente, es necesario pulir el algoritmo del agente para que no entre en un ciclo vicioso, lo cual puede ser solucionado mediante la implementación de una variable que guarde los nodos o rutas ya vistadas para evitar esto y siempre seleccionar una ruta eficiente y diferente.

Parte 4: reflexión

Al inicio de estas 5 semanas del curso mis expectativas eran altas debido a que los temas eran relacionados a lo que yo me quiero dedicar, por lo que estaba muy entusiasmado de poder, por fin, realizar un proyecto que empatara con mis gustos. Después de haber desarrollado, desplegado y entregado esta solución al fin de este curso, puedo confirmar que he aprendido mas de lo que esperaba en este corto tiempo y, además, pude poner esos conocimientos en practica con este reto. Cada semana de trabajo fue intensa y muy laboriosa debido al poco tiempo que teníamos para aprender y poner en practica los temas vistos en clase, por lo que fue necesario dedicarle muchas horas extra para completar de la mejor manera esta materia. No obstante, al final del camino, todo el tiempo extra que fue invertido en la materia se pudo reflejado en el producto final, aunque la sensación es agridulce. Como la gran mayoría de las clases dentro de este modelo, el ritmo de trabajo y el tiempo no te dejan asimilar y disfrutar lo que estas haciendo, ni tampoco te deja desarrollar todo lo que tienes en la cabeza. En el caso particular de este proyecto, tuvimos muchas ideas que al final no pudieron concretarse debido al poco tiempo que teníamos para desarrollar toda la solución. De cualquier manera, creo que le pude sacar provecho a todo lo que nos enseñaron y, al final del día, el progreso y lo que aprendimos es lo mas importante.