Campus Santa Fe

Fecha de Entrega: 30/11/2023

Materia: Modelación de Sistemas Multiagentes con Gráficas Computacionales (Gpo 302)

Reto: Movilidad Urbana

Domingo Mora | A01783317

Cristina Gonzalez | A01025667

Problema:

El problema que se está presentando es el siguiente: Una simulación de tráfico donde los agentes(coches) respeten su orden en el tráfico y llegar a su destino. Los vehículos deben evadir obstáculos, evitar choques, y respetar los semáforos dentro de la simulación. Considerando estas cosas cuando el vehículo llega a su destino se borra del mapa, y en caso de que ya no se puedan crear nuevos agentes; la simulación se detiene.

Propuesta de Solución:

La propuesta se compone de dos elementos que sirven simultáneamente; primero el servidor. El servidor se encarga de una simulación 2D que se abre en el puerto predeterminado donde actúan 3 scripts de python.

Primero el servidor:

El servidor es un script que reconoce los colores de aquellos componentes de la simulación 2D, las especificaciones del grid (Importando el archivo .txt que incluye el layout de la ciudad), el servidor y finalmente el puerto junto con la iniciación para abrir el puerto del servidor.

También hay otro servidor que tiene los API's/Endpoints necesarios para la simulación los cuales son:

- Recibe Parámetros

- Recibe Agentes
- Recibe Obstáculos
- Recibe Semáforos
- Actualización de todos los agentes
- Correr los Endpoints

Luego el modelo:

El modelo es donde la mayoría de la funcionalidad de la simulación ocurre. Comienza reconociendo el script .txt dentro de un folder llamado "City_files". Empieza el script leyendo el json donde reconoce que cada letra del .txt (Destino, Obstáculo, Orientación de la calle, y los movimientos que puede utilizar el agente). Empiezan la funciones para la simulación:

- Generar Coches
- getDestination
- Road Directions
- Position (Available and Valid)
- Edges (Traffic Lights, Weight, Destination, Road)
- Grafo dirigido

Diseño de Agentes:

El agente contiene los estados de cada uno de los componentes (Agente, Luces, Obstáculos, Calle, y el Destino). El estado de todos los componentes menos el agente y las luces; solo se les da un identificador. Los semáforos tienen además de un identificador un estado para ver en qué color está el semáforo dependiendo del intervalo de tiempo determinado.

El vehículo y sus acciones son el principal funcionamiento de este script. Primero se inicializa con todos los componentes que va a reconocer, luego calcula un camino para llegar al destino que escoge aleatoriamente usando la función predeterminada de a_star de la librería "Networkx". Reconoce en qué instancias se puede mover dependiendo si hay coches en frente o por el estado del semáforo, y tiene una función para cuando esté muy lleno cierto carril pueda recalcular su camino para evitar crear tráfico.

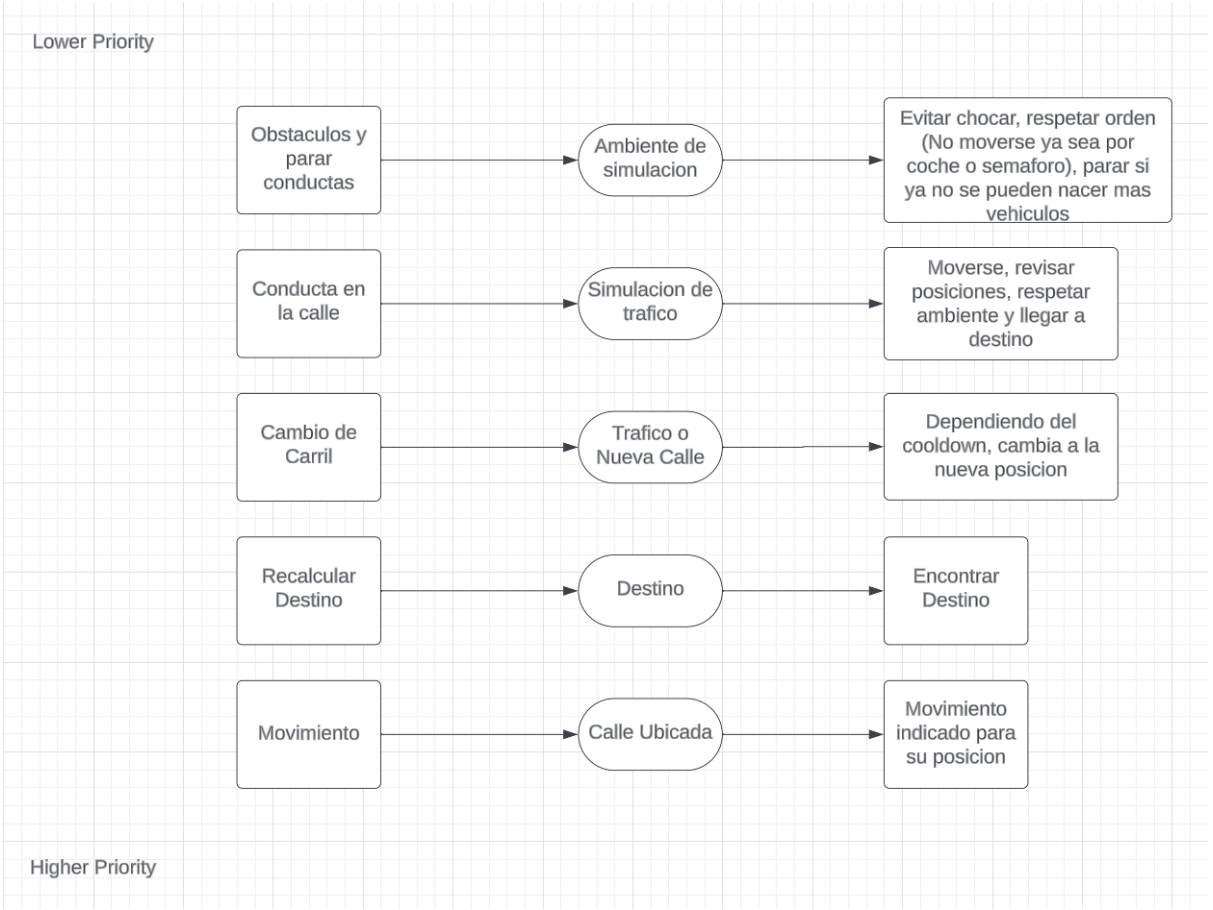
Existe también una función llamada "update_lane_and_path" que es esencial para este script. Incluye un contador que mantiene cuenta del tiempo desde el último cambio, revisa el cooldown de la acción previa y después revisa la celda de enfrente. Si las condiciones descritas se cumplen se hace el cambio de carril o de calle. Si la condición comenzó, recalcula su ruta o sigue con el destino determinado al principio. Finalmente, las acciones de movimiento donde revisa si existe el camino, se mueve si existe y regresa el siguiente movimiento, revisa su destino, actualiza su posición y mantiene conciencia de los obstáculos o si debe de parar el agente. Se puede mover únicamente de las siguientes maneras

- Arriba
- Abajo
- Izquierda

- Derecha
- Diagonal Izquierda
- Diagonal Derecha

Esto es lo que conforma el servidor de la simulación 2D; Sin embargo faltan los scripts de la simulación que se lleva a cabo en Unity Engine. Dentro de la simulación no se tiene que hacer otra simulación gracias a los endpoints en el script de ServerA.py, los únicos archivos que se crean en esta parte del proyecto son para aplicarlos a los objetos 3D que hay en este programa. Primero el ApplyT que se encarga del movimiento del vehículo junto con su archivo pareja que es el HW_Transforms que se encarga de dar las matrices de traslación, escala y rotación. Luego está el script del citymaker que se encarga de recibir las instrucciones del mapa (Orientación, ubicación de obstáculos, destino y semáforos) y también de transformar el ambiente a uno 3D. Un script muy importante de la simulación es el AgentController, que como dice su nombre es el que instala y recibe todos aquellos agentes que tienen algún comportamiento en la simulación. Se encarga de tener los datos de los agentes, designar posiciones previas y nuevas, actualizar los agentes y la simulación, mandar la configuración, y tener todo funcionando adecuadamente. Los últimos scripts son solo para designar las luces de los semáforos y esconder objetos en caso de que sea necesario (Casos ya descritos).

Arquitectura de Subsunción:



Ambiente:

El ambiente se basa en 4 componentes; estos componentes están posicionados en las ubicaciones que describen los archivos .txt y se leen según las instrucciones que dicta el diccionario json. El ambiente se constituye de calles de un solo sentido que tienen dos carriles cada uno, luego existen los obstáculos que son representados por edificios y en caso de unity por .obj. Luego hay semáforos en cada intersección del ambiente uno en cada sentido de la calle, y finalmente, los destinos son ubicaciones predeterminadas donde los agentes basan su ruta de comienzo hasta llegar.

Dentro de este ambiente los agentes son colocados en los 4 extremos exteriores del mapa para comenzar la simulación de una "Ciudad".

Conclusiones:

En fin, este reto consiste de muchos componentes y programas diferentes para tener esta simulación sirviendo en 2D y en 3D. Lo más importante fue tener el backend (2D) sirviendo porque ahí es donde ocurre toda la simulación en términos de especificaciones de movimientos, instancias entre otros componentes descritos anteriormente. Al ya tener esta parte lo siguiente fue conectar los endpoints para poder comenzar con la simulación en Unity. Esta parte fue menos carga porque el movimiento de los coches ya había sido implementada en un trabajo previo de la clase junto con su script pareja de las matrices. Hubo algunos ajustes necesarios para aplicarlo a este ambiente pero más que nada el verdadero reto fue la parte backend del proyecto. En términos de frontend fue mucho sobre la implementación de la visualización del reto y tener una ciudad tridimensional funcionando como en el servidor del back. Esta parte fue divertida al poner en práctica la creatividad dentro del simulador. Fue un reto lleno de aprendizaje gracias a que este proyecto necesitaba mucha comunicación entre códigos; lo cual mejoró nuestras habilidades de fundamentos computacionales que hemos desarrollado a lo largo de esta carrera.