Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

Fecha de entrega: 28 de noviembre del 2022

Evidencia 2. Avances y presentación del reto

Modelación de Sistemas Multiagentes con Gráficas Computacionales (Gpo 301)

Profesorado:

Octavio Navarro Hinojosa

Jorge Adolfo Ramírez Uresti

Alumnado:

José Luis Madrigal Sánchez A01745419

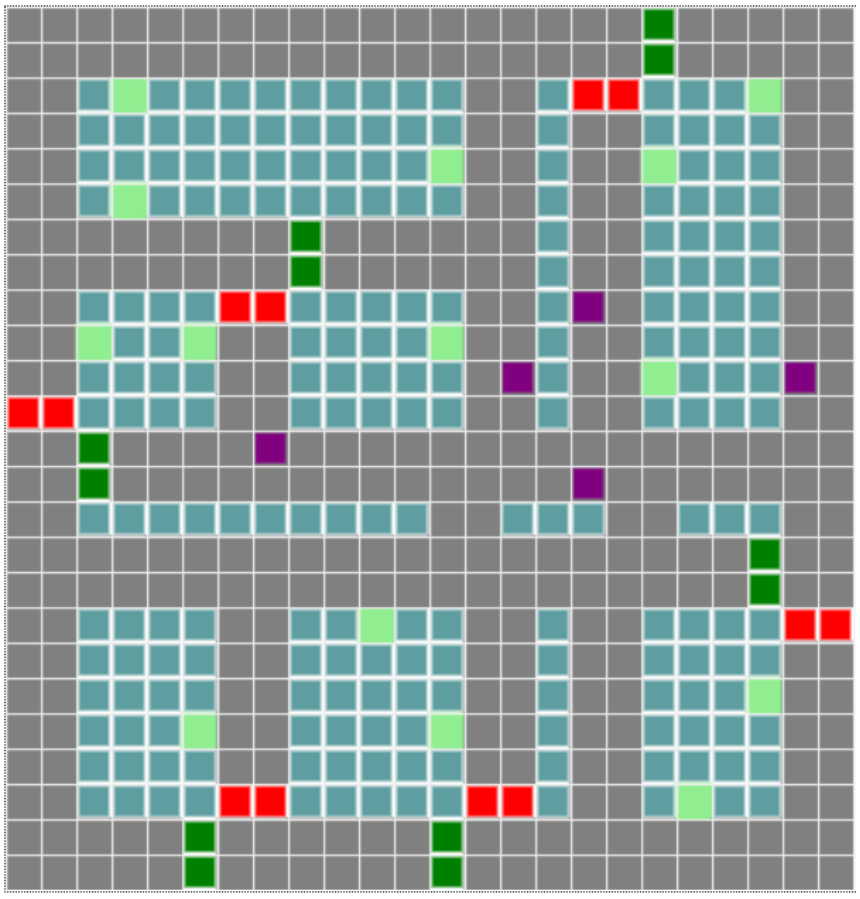
César Emiliano Palome Luna A01746493

Christian Parrish Gutiérrez Arrieta A01751584

Jorge Isidro Blanco Martínez A01745907

Descripción y análisis de ambiente

El ambiente será una ciudad con un posicionamiento constante de sus elementos, es decir, no se tendrá ninguna modificación del acomodo de las distintas partes que la conforman, también cabe mencionar que desde un inicio se generarán sus componentes en celdas específicas, por lo que la estructura general del ambiente no es aleatoria. Las partes que conforman a la ciudad son las calles, los semáforos, los edificios y los destinos, los cuales serán necesarios para definir el comportamiento del único agente real que son los autos. También es importante mencionar que para facilitar la manipulación de estos elementos, se hace uso de un archivo de texto que contiene su acomodo representándolos con símbolos, los cuales también ayudan a definir el sentido de las calles. Como se puede observar en la imagen, se tienen calles señaladas con color gris, en el contorno, así como en la parte media, colindando con algunos edificios, que estan representantes por cuadros de color azul, que contienen destinos señalados en verde claro, igualmente se encuentran parejas de semáforos en algunas intersecciones, los cuales se sincronizan para tener colores contrarios que permitan transitar en un entorno organizado y seguro de los vehculos señalados con color morado.



En cuanto a las características que tiene el ambiente en un sistema multiagentes, se tiene lo siguiente:

- Accesible o Inaccesible: es accesible debido a que el semáforo recibirá información de su semáforo vecino y del número de vehículos, así como el vehículo detectará el color que tendrá el semáforo y el sentido de las calles.
- Determinista o no determinista: es determinista porque los estados del semáforo dependen de las posiciones de los vehículos y a su vez de otros semáforos para generar un programa de luces.
- Episódico o no episódico: es episódico debido a que las acciones tomadas por los agentes dependen de las acciones anteriores, ya que los vehículos van tomando sus propias decisiones, queriendo llegar a cierta posición por medio de un camino propio, con lo cual los semáforos se irán adaptando, por lo que el tránsito vial es un factor importante para ir realizando cada paso.
- Estático o no dinámico: es estático porque no se tiene modificación alguna de los elementos cuando los agentes toman decisiones, simplemente se tiene una actualización de las posiciones de los vehículos y los colores de los semáforos.
- Discreto o continuo: es continuo debido a que no hay un límite de acciones, puesto que la simulación seguirá ocurriendo mientras haya coches que transiten para establecer intersecciones programadas.

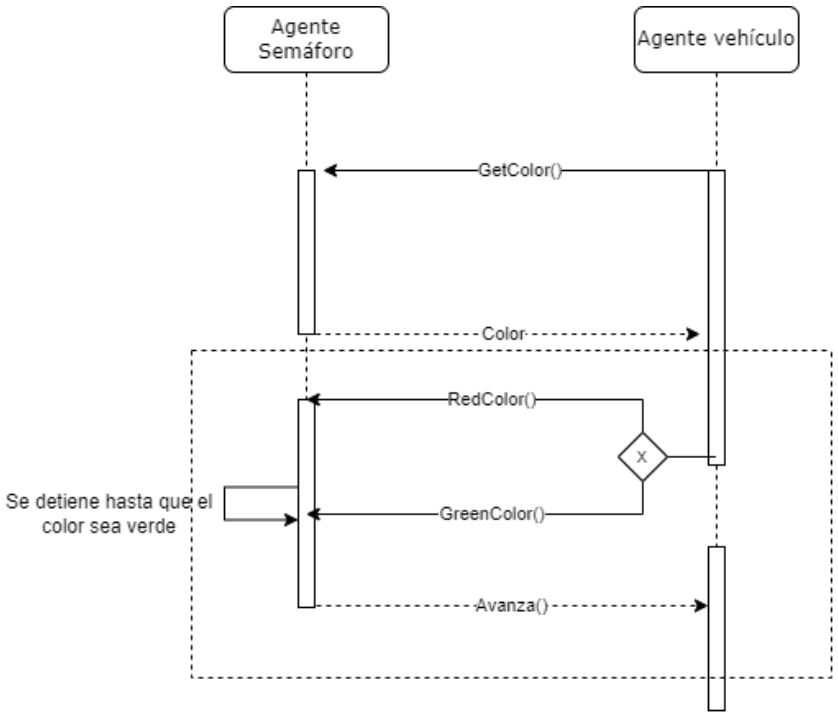
Diagramas de agentes AUML

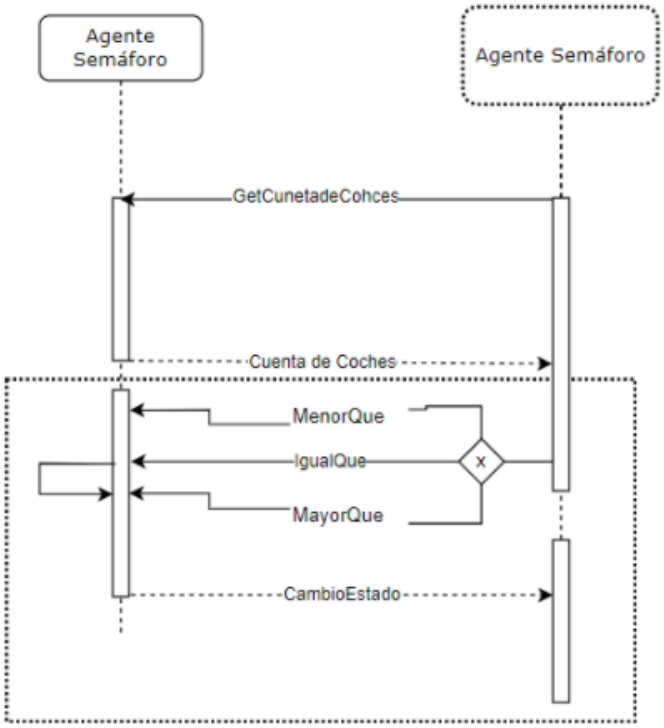
Vehículo
Grupo: Transporte
Rol: Transitante
Servicio: Moverse por la ciudad
Protocolo: Avanzar en las calles viendo si el semáforo está en verde o amarillo
Evento: Cambio de semáforo
Metas: Transitar y respetar los semáforos y vueltas
Plan: no plan
Acciones: Avanzar, frenar
Información: Color de los semáforos, coches vecinos, sentido de la calle

Semáforo
Grupo: Admin
Rol: Moderador
Servicio: Indica cuando los autos deben avanzar o parar
Protocolo: Coordinar los tiempos entre los semáforos, poder modificar el tiempo de duración de cada semáforo dependiendo de la cantidad de vehículos
Evento: Llegada de coches
Metas: Reducir la congestión de un cruce y mejorar la movilidad urbana
Plan: no plan
Acciones: Cambiar el color del semáforo entre verde a rojo para que se detengan y disminuir o aumentar la duración de cada color dependiendo de que tan llena está la calle
Información: cantidad de vehículos en calle

Los siguientes elementos no se incluyen en este diagrama porque sólo actúan como objetos, pero igualmente serán programados como agentes para colocarlos de forma más sencilla en las celdas: calle, edificio, destino.

Protocolos de interacción





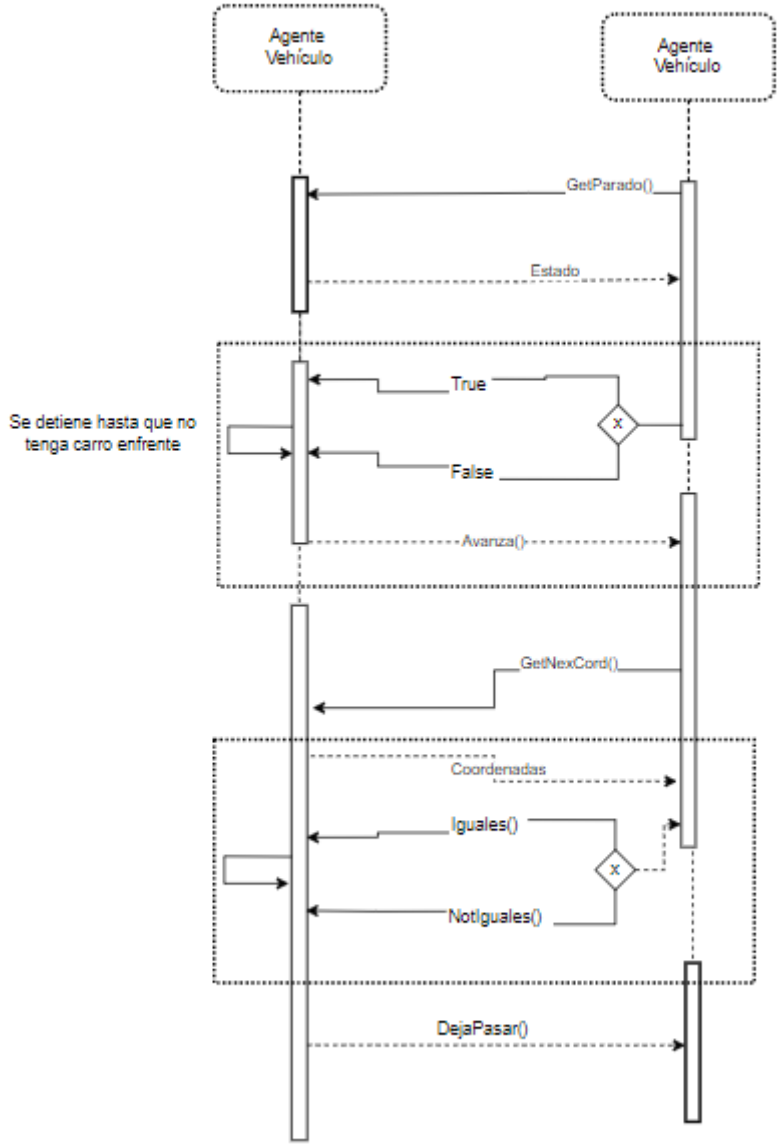
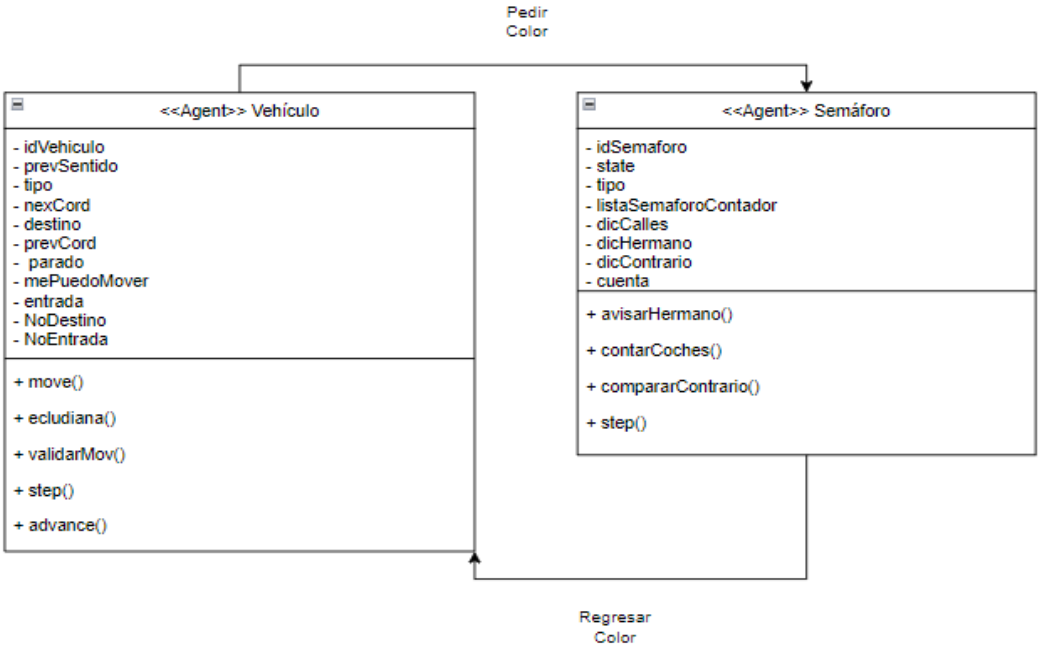

Agente
Vehículo
Agente
Vehículo
GetParado()
Estado
Se detiene hasta que no
tenga carro enfrente
True
X
False
Avanza()
GetNexCord()
Coordenadas
Iguales()
X
NotIguales()
DejaPasar()

Diagrama Organización SMA



Implementación completa de los agentes

Liga de Repositorio:

https://github.com/A01745419/movilidad_urbana.git

Modelo con visualizador local de grid Mesa se encuentra en carpeta MesaLocalViz.

Modelo con Servidor Flask en carpeta MesaServerUnity.

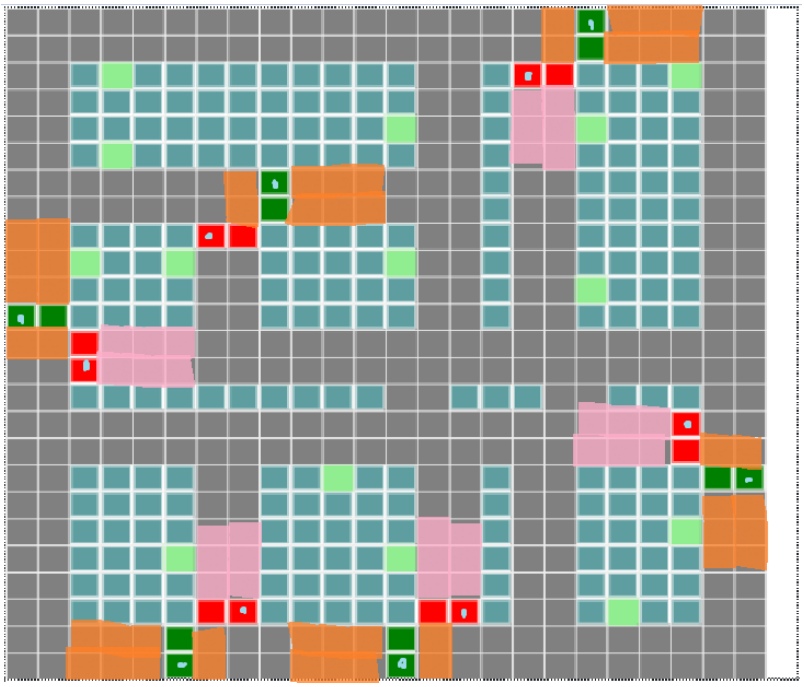
Proyecto de gráficas en carpeta MovilidadUrbanaUnity.

La simulación en Mesa incluye la colocación de los objetos en el entorno (calles, destinos y edificios), los cuales se mantienen constantes durante todos los pasos y solamente sirven para definir algunas restricciones para el comportamiento de los agentes.

-En cuanto a la implementación de los coches:

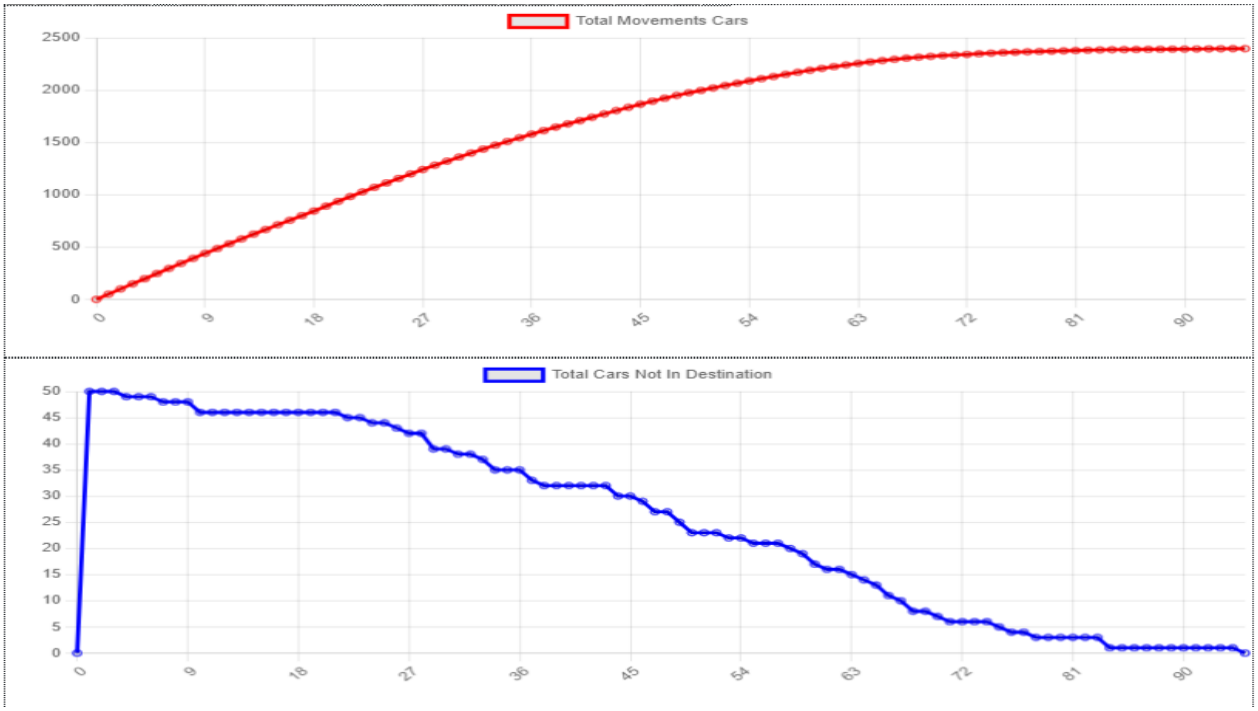
Los coches pueden avanzar en las 4 direcciones y no esquinas, en un radio de 1 celda. Cuando se inicializan eligen un destino aleatorio y se dirigen hacia él respetando los sentidos y semáforos, además de que cuando se encuentran un cruce, hace un cálculo de las distancias de las celdas en sus posibles direcciones al destino, con el cual determina la mejor opción para llegar más rápido.

-En cuanto a la implementación de los semáforos:



Ya que cada semáforo completo está conformado por 2 agentes semáforos distintos, solo se eligió uno de ellos para hacer el conteo de coches en su calle (alcance de 3 celdas en ambos carriles, total de 6), y ese mismo sería el que le avisa al semáforo junto a él (su hermano) en que color estar, en la imagen son los que tienen el punto azul. De igual manera, al tener la posibilidad de tener el mismo número de coches en las 2 calles de una intersección, se designaron los semáforos prioritarios (aquellos en una calle que sigue avanzando derecho, sin dar vuelta), que en la imagen son los que están en verde, y de hecho tienen 2 celdas más que sensar, para mantener la prioridad cuando ya pasaron los coches. También cabe mencionar que los prioritarios son los únicos que hacen la comparación con su contrario, para no repetir el mismo proceso 2 veces, en este caso, como se había mencionado, cada uno de los verdes al ser prioritario es el que manda la señal al otro para que se haga rojo.

Para tener una idea de la ejecución del código, se incluye una prueba con 50 coches (es importante mencionar que cuando llegan a su destino, se eliminan del grid y scheduler):



Los 50 coches llegaron a sus destinos en 95 steps, realizando en total 2396 movimientos.

Implementación completa de la interfaz gráfica de la simulación

Liga de Repositorio:

https://github.com/A01745419/movilidad_urbana.git

Modelo con visualizador local de grid Mesa se encuentra en carpeta MesaLocalViz.

Modelo con Servidor Flask en carpeta MesaServerUnity.

Proyecto de gráficas en carpeta MovilidadUrbanaUnity.

Documentación describiendo el proceso de instalación

https://youtu.be/QUCXsx-yALA

Ejecución de la simulación

https://youtu.be/6_PICK3ejnM

Plan de trabajo

Actividad	Estado	Responsable(s)	Fecha estimada	Esfuerzo estimado	Esfuerzo real	Diferencia estimación y real
Descripción y planeación del reto	Terminado	Christian Parrish	03/11/2022	30 minutos	30 minutos	0 Minutos
Identificación de agentes	Terminado	José Luis Christian Parrish César Palome Jorge Blanco	04/11/2022	110 minutos	150 minutos	40 Minutos más lento
Plan de trabajo	Terminado	Jorge Blanco	03/11/2022	20 minutos	20 minutos	0 Minutos
Reflexiones individuales	Terminado	José Luis Christian Parrish César Palome Jorge Blanco	03/11/2022	30 minutos	30 minutos	0 Minutos
Descripción y análisis de medio ambiente	Terminado	José Luis	27/11/2022	45 minutos	30 minutos	15 minutos
Descripción PEAS agenes	Terminado	José Luis	27/11/2022	30 minutos	30 Minutos	0 Minutos
Diagramas AUML	Terminado	César	27/11/2022	30 minutos	35 Minutos	5 minutos más lento
Diagrama Organización SMA	Terminado	Jorge	27/11/2022	45 minutos	40 Minutos	5 Minutos
Protocolos de interacción	Terminado	Christian	27/11/2022	50 minutos	40 Minutos	10 Minutos

Código Mesa	Terminado	José Luis César	02/12/2022	45 Horas	50 Horas	5 Horas más lento
Obtención de Prefabs	Terminado	Christian	27/11/2022	60 minutos	90 Minutos	30 Minutos más lento
Servidor Flask	Terminado	Jorge	02/12/2022	120 minutos	60 minutos	60 Minutos más rápido
Código C# Unity para lectura de posiciones y estados	Terminado	Christian Jorge	02/12/2022	10 Horas	6 Horas	4 Horas menos