Instituto Tecnológico y de Estudios Superiores de Monterrey.

Campus Estado de México

Modelación de sistemas multiagentes con gráficas computacionales

TC2008B.301

Avance al 60%

Revisión 3

Profesores

Jorge Adolfo Ramírez Uresti

Sergio Ruiz Loza

Integrantes

Julio Cesar Vivas Medina..... ITC | A01749879

Sebastián Espinoza Farías..... ITC | A01750311

Ulises Jaramillo Portilla..... ITC | A01798380

Jesús Ángel Guzmán Ortega..... ITC | A01799257

Fecha de entrega: 22 de noviembre del 2024

Índice

1. Objetivos de la entrega

El objetivo de esta entrega, se basa principalmente en la identificación y desarrollo de los siguientes puntos:

- Diseñar un sistema multiagente para simular una intersección de tráfico controlada por semáforos inteligentes.
- Especificar los agentes involucrados, su tipo y roles en la cooperación para el control del tráfico.
- Describir la forma de interacción entre los agentes, estableciendo cómo se comunican, coordinan y responden a cambios en el tráfico.
- Definir y caracterizar el ambiente (entorno) donde los agentes interactúan, incluyendo detalles relevantes para la simulación del tráfico y la detección de vehículos.

2. Descripción detallada del medio ambiente

2.1. Descripción del entorno

El entorno son todas aquellas intersecciones, donde ocurren los cambios de tráfico y se procesan las interacciones entre agentes. Sus características son las siguientes:

2.1.1. Entorno Físico (Intersecciones y cruces)

- Carriles de acceso: La intersección tiene múltiples carriles de entrada y salida en cada dirección, cada uno con un semáforo controlado por un agente.
- Zonas de detección de vehículos: Sensores ubicados estratégicamente en la entrada y salida de cada carril miden el flujo de vehículos, la velocidad y la densidad.

2.1.2. Condiciones de Tráfico Variables

La densidad de tráfico varía a lo largo del día (horas pico, flujo moderado, flujo bajo) y es monitoreada a través de los agentes en el rango. Existen eventos aleatorios, como la petición de un peatón en cruces con o sin semáforo.

2.1.3. Entorno de Comunicación y Datos

La red permite que los agentes intercambien datos en tiempo real y se comuniquen entre ellos, para coordinar patrones de tráfico en la zona.

2.1.4. Reglas de Tráfico

Cada agente semáforo sigue reglas básicas de tráfico y sincronización para evitar colisiones y coordinar el paso de vehículos en las direcciones correspondientes. Las reglas iniciales iniciales son las siguiente:

- Mientras no haya un vehículo cercano, el semáforo estará en luz amarilla.

- Cuando un vehículo se acerque a la intersección, enviará un mensaje con el tiempo estimado de arribo.
- El semáforo dará luz verde al semáforo más cercano y establecerá un programa de luces entre los semáforos de la intersección para permitir la circulación de los vehículos.

Los agentes de semáforo, vehículos y peatones operan de acuerdo con reglas preestablecidas, pero también adaptativas, permitiendo cambios en tiempo real según las condiciones y las decisiones que tomen en base a la información que perciban de los mismos.

Proponemos una modelación eficiente y segura ante las consideraciones iniciales de los agentes que se describen a continuación en los siguientes puntos. Los agentes pueden ajustarse a las condiciones de tráfico en tiempo real, coordinando sus acciones para maximizar el flujo y minimizar la congestión en la intersección.

2.2. Descripción del medio ambiente por Agente

2.2.1. Vehículo

El ambiente de los vehículos consiste en calles de doble sentido de un carril, rotondas, edificios estacionamientos y estructuras que rodean a las calles, los semáforos que indican si el coche puede avanzar o deben detenerse y otras instancias del vehículo que obligan al agente a mantener distancia entre otros de su mismo tipo..

Accesibilidad:

Es 80% accesible, ya que debe de tener vista del semáforo que lo puede controlar así como otras instancias de vehículos que podrían obligarlo a mantener una distancia, lo único a lo que no tiene acceso es a terreno que no sea carretera.

Determinista:

Es 75% determinista debido a que el estado de movimiento se puede predecir dependiendo de la respuesta de los semáforos y de la distancia a otros coches, el factor que no podemos determinar es el tiempo en que dan respuesta los semáforos dependiendo de que tantos coches existan en una avenida que corte la circulacion de nuestro vehiculo inicial.

No Episódico:

Es 90% no episódico ya que no necesitamos registrar en memoria la acción previa del vehículo para ninguna futura situación, ni dividimos en fracciones de tiempo

predeterminadas las acciones, aunque necesitamos la respuesta anterior del semáforo para determinar si nos movemos o no.

Dinámico:

Es 95% dinámico, todos los elementos del ambiente a excepción de los edificios y estacionamientos, están en constante cambio, y mandan diferentes estados y respuestas en cada momento.

Discreto:

Es 90% discreto porque tenemos un número finito de siguientes estados y decisiones que el vehículo puede tomar, por ejemplo solo puede avanzar en su carril o tomar vuelta a la derecha o izquierda en caso de que exista una calle con ese sentido, por lo que tenemos un número de posibilidades para las percepciones y acciones.

2.2.2. Semáforo

El ambiente de los semáforos consiste en las calles colindantes a la instancia, así como todos los elementos contenidos en estas calles como el flujo de vehículos y peatones y el estado de otros semáforos.

Accesibilidad:

El 90% del ambiente es accesible a través de los mensajes que recibe de otros semáforos y de la distancia de los coches en un radio predeterminado, a lo que no tenemos acceso por otra parte es a los edificios y espacios fuera de las calles.

No Determinista:

Es 80% no determinista debido a que no se sabe cuántos coches llegarán por cada lado despues del radio, por lo que no sabemos la frecuencia de como entraran al ambiente, y tampoco sabemos cuál será la frecuencia de las respuestas de otros semáforos, aunque las calles siempre mantendrán su sentido y dirección.

No Episódico:

Es 90% no episódico, no necesitamos registrar en memoria la acción previa de ningún semáforo para ninguna futura situación, ni dividimos en fracciones de tiempo predeterminadas la manera en que interactúa son totalmente aleatorias.

Dinámico:

95% dinámico, todos los elementos que interactúan a excepción de los edificios están en constante cambio, por lo que mandan diferentes respuestas en cada momento

Discreto:

95% discreto porque tenemos un número finito de siguientes estados y decisiones que el semáforo puede tomar, por ejemplo solo puede indicar si estamos en verde, amarillo o en rojo, por lo que tenemos un número de posibilidades para las percepciones y acciones.

2.2.3. Peatón

El ambiente del peatón consiste en las aceras, cruces peatonales, señales y semáforos diseñados para peatones, además de la interacción con los vehículos, estado de semáforos y el flujo de personas en las calles.

Accesibilidad:

El entorno es aproximadamente 75% accesible para el peatón, ya que puede ver las señales de tránsito y los vehículos en su camino inmediato, así como los semáforos que le indican cuándo cruzar. Sin embargo, el peatón no tiene acceso a información más allá de su campo de visión, como el tráfico en calles adyacentes o la respuesta de otros semáforos.

Determinista:

El ambiente es 60% determinista para el peatón, ya que puede predecir si es seguro cruzar en función de los semáforos y el tráfico visible. Sin embargo, la imprevisibilidad de los vehículos y la respuesta de otros peatones hace que no sea completamente determinista.

No Episódico:

Es 80% no episódico ya que el peatón, generalmente, no necesita recordar acciones pasadas para tomar decisiones futuras. Cada cruce o interacción suele ser independiente, y las decisiones se basan en el estado actual de los semáforos y la proximidad de los vehículos.

Dinámico:

El ambiente es 90% dinámico ya que la cantidad de vehículos, peatones, y los cambios en los semáforos están en constante cambio, exigiendo respuestas rápidas por parte del peatón.

Discreto:

Es 75% discreto ya que, aunque el peatón tiene un número limitado de acciones cómo avanzar, detenerse o retroceder, el ambiente permite variaciones, como la dirección de los otros peatones o la velocidad de los vehículos, lo cual le da cierto grado de continuidad.

3. Descripción PEAS por Agente

3.1. PEAS: Vehículos			
Performance	Environment	Actuators	Sensors
Capaces de desplazarse de forma eficiente, reducir el tiempo de viaje y adaptarse a diferentes estados de tráfico. Pueden tomar decisiones autónomas sobre la ruta, buscar estacionamiento, y reaccionar a la señalización (semáforos y peatones).	Calles, intersecciones, espacios de estacionamiento, semáforos.	Acelerador, freno, dirección, cambio de carril, sistema de señalización (luces), estacionamiento.	Sensor de detección de semáforos y peatones utilizando proximidad.

Tabla 1. PEAS de Vehículos

3.2. PEAS: Semáforos			
Performance	Environment	Actuators	Sensors

Regular el flujo vehicular y peatonal de manera eficiente, adaptarse a cambios en el tráfico en tiempo real y coordinarse con otros semáforos para optimizar el flujo en varias intersecciones.	Intersecciones viales, pasos peatonales, y áreas de tráfico variado.	Los semáforos tienen cambios de luces (Rojo, Amarillo, Verde).	Sensores de detección de peatones cercanos, sensores de detección de vehículos.

Tabla 2. PEAS de Semáforos

3.3. PEAS: Peatones			
Performance	Environment	Actuators	Sensors
Capacidad de cruzar calles de manera segura, interactuar con semáforos, y adaptarse al entorno para evitar colisiones.	Calles, pasos peatonales, intersecciones, semáforos.	Los peatones tienen movimientos en cualquier dirección, se pueden detener, detectan la luz roja, amarilla y verde del semáforo.	Sensor de colores para saber cuándo cruzar las calles dependiendo del semáforo. sensor de proximidad para cruces peatonales

Tabla 3. PEAS de Peatones

Se realizaron mínimas modificaciones en los diagramas de las tablas a comparación de la primera entrega, esto debido a la adaptación y análisis de los comportamientos de los agentes.

4. Diagramas de Agente usando AUML

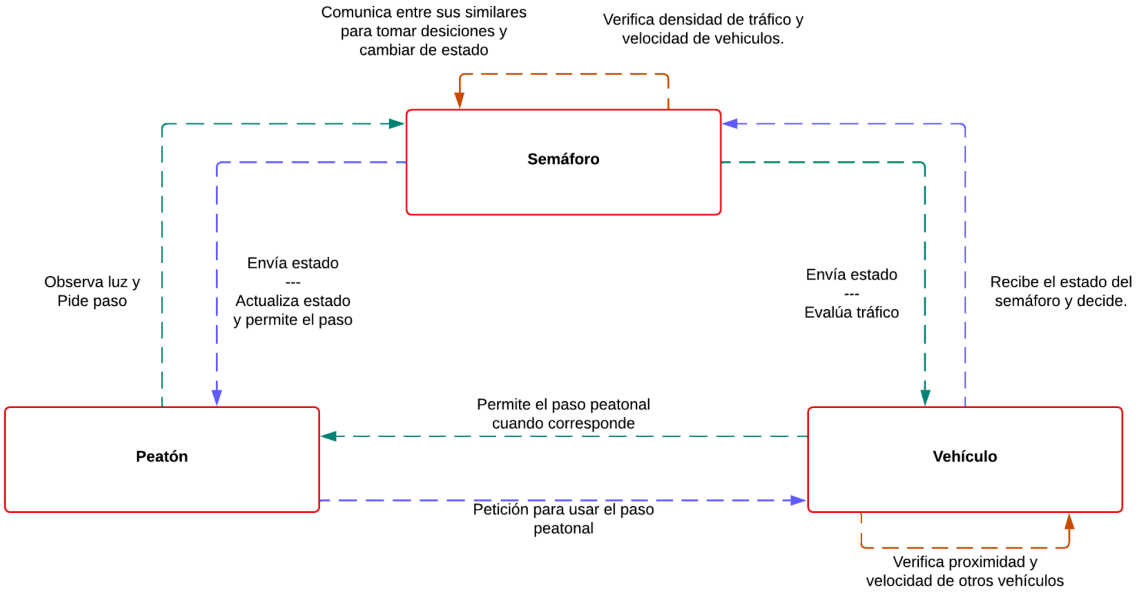
Diagramas de Agente AUML:

	Diagrama de agentes	
Vehículo	Semáforo	Peatón
Grupo: Vehículos	Grupo: Control de tráfico	Grupo: Peatones
Rol: Participante en el tráfico de la ciudad	Rol: Regulador de tráfico	Rol: Generador de tráfico
Servicio: Moverse por las calles, estacionarse, obedecer semáforos, ceder el paso a los peatones.	Servicio: Proporcionar una regulación en las intersecciones y zonas de cruces peatonales para el tráfico	Eventos - acciones:
Protocolo: Búsqueda de rutas óptimas, búsqueda de	Protocolo: Coordinación de semáforos	Detección de tráfico -> Si el peatón detecta tráfico que le

estacionamiento, reacción a semáforos y vehículos, ceder paso.				impide cruzar la calle, pide paso en el semáforo o intersección.
Eventos: Cambio de ruta, estacionamiento, frenado en semáforos, reacción a señales de otros vehículos, vueltas/cambio de dirección.		Eventos: Cambios por volúmen de tráfico, cambio por peatón		Detección de estado de semáforo -> Si el peatón detecta el rojo, cruza la calle.
Objetivos: Alcanzar el final de la ruta, encontrar estacionamientos, obedecer adecuadamente a los semáforos, simular el comportamiento de un vehículo y su comportamiento en una ciudad, respetar el cruce del peatón cediendo el paso.		Objetivos: Optimizar el flujo de vehículos y peatones en la ciudad para un traslado eficiente.		Detección de estado de semáforo -> Si el peatón detecta el verde, pide paso o espera.
Plan: Sin planes		Plan: Ajustar la duración de los estados del semáforo de acuerdo al tráfico.		Detección de estado de semáforo -> Si el peatón detecta el amarillo, sale de la calle, espera.
Acciones: Acelerado, frenado, cambio de carril, cambio de dirección.		Acciones: Cambiar el color (Verde, amarillo, rojo) y la duración de los estados.		Detección de cruces peatonales -> Si el peatón tiene en su campo de visión un semáforo que haga su paso más seguro que un cruce sin semáforo, lo prioriza.
Conocimiento: Carriles de la ciudad, señales de tráfico, estado de semáforos, vehículos cercanos .		Conocimiento: Flujo y volúmen de tráfico, estado de otros semáforos, peatones cercanos.		

Tabla 4. Diagrama de Agentes

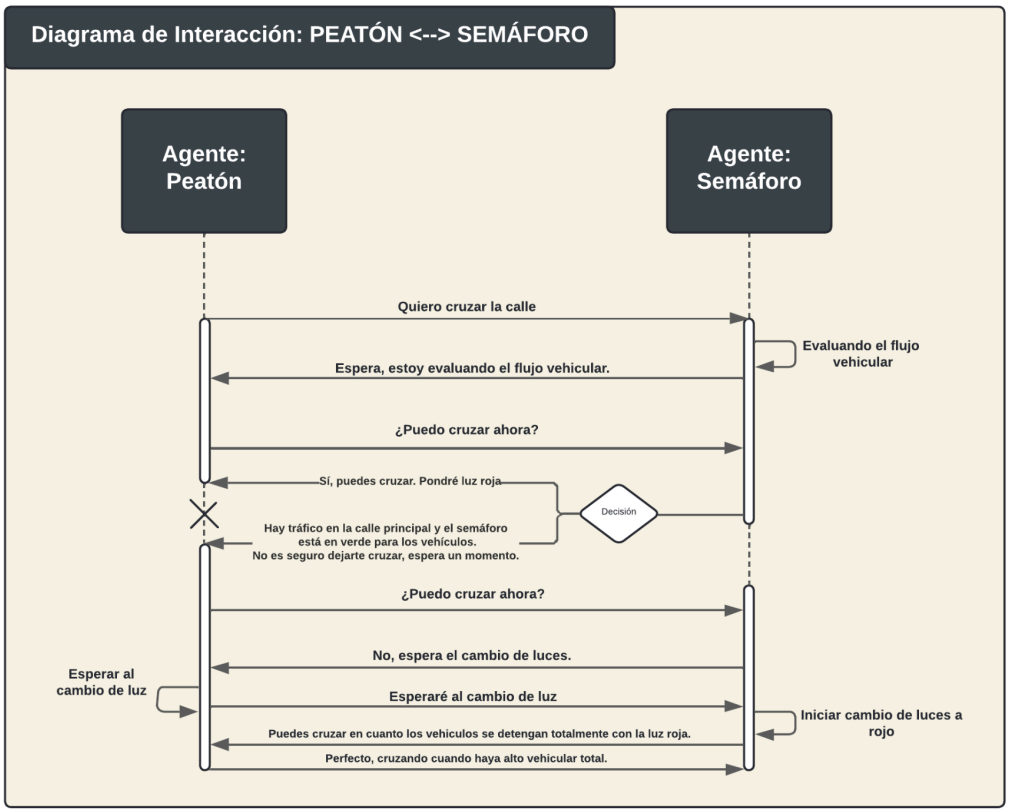
5. Diagrama de organización SMA



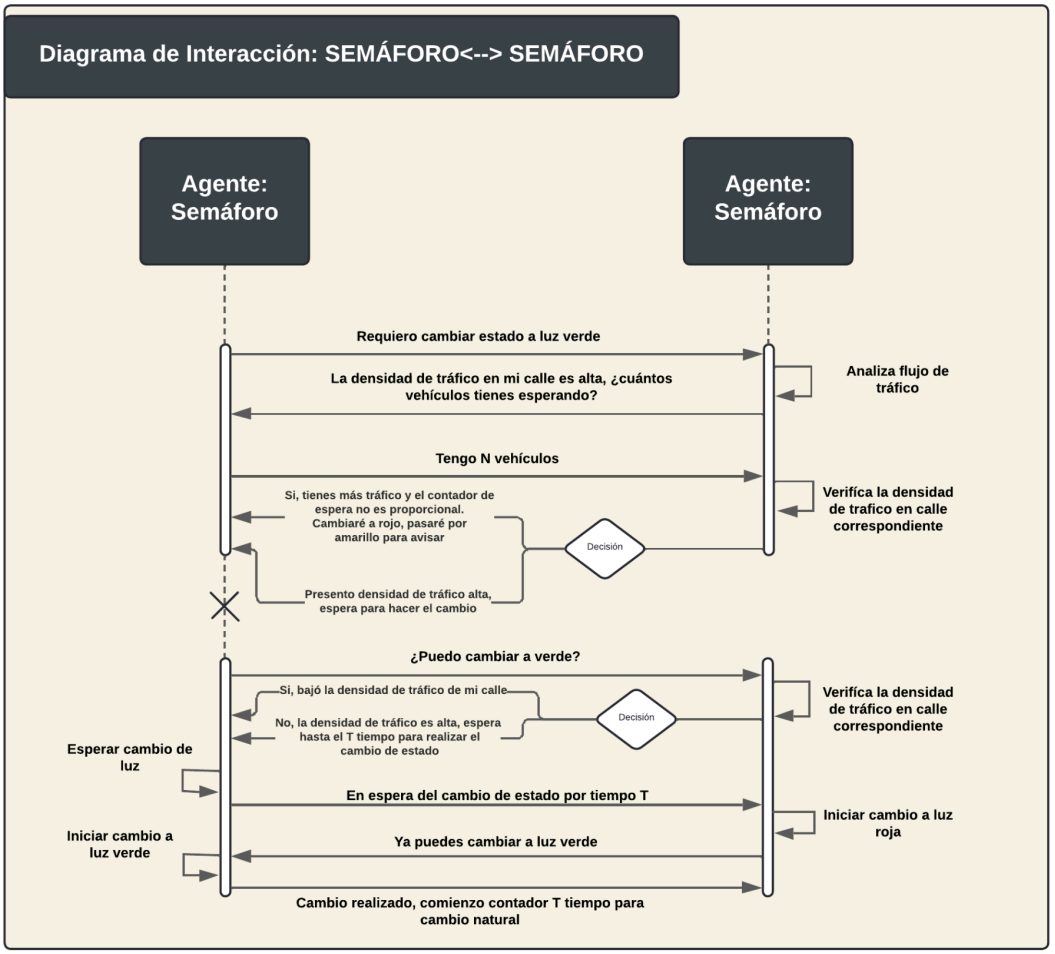
Imágen 1. Diagrama de organización SMA

El diagrama anterior muestra una vista general sobre la interacción y organización entre los agentes presentados para este reto, indicando las comunicaciones entre dichos agentes y sus parámetros de decisión o petición, así como propias interacciones entre agentes del mismo tipo.

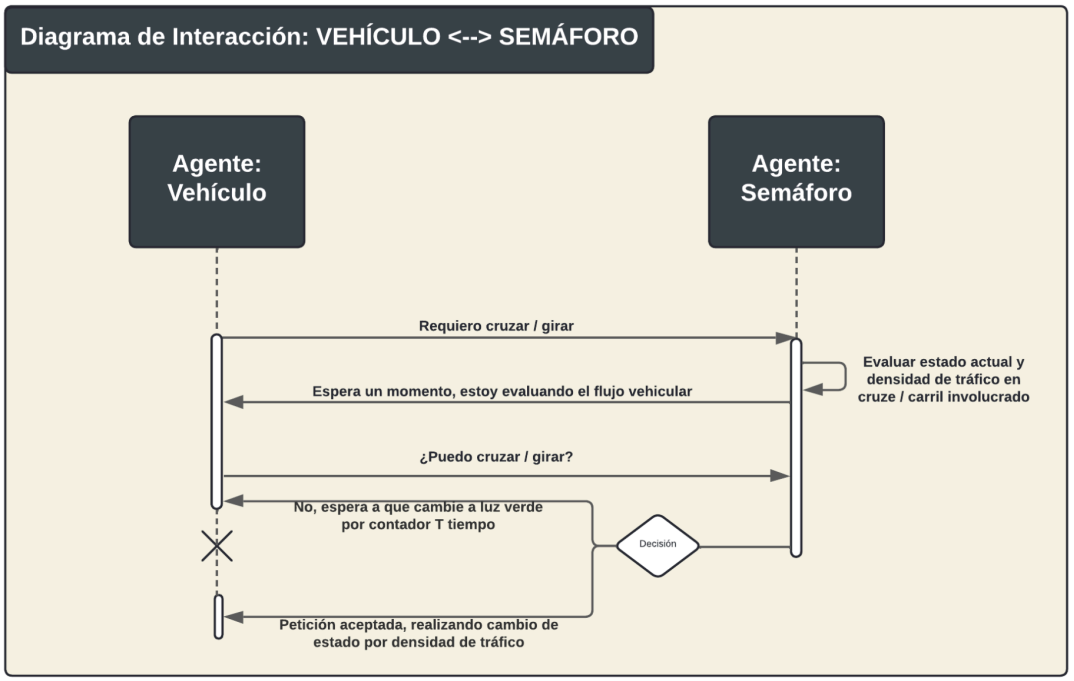
6. Diagramas de interacción entre agentes



Imágen 2. Diagrama de interacción Peatón Semáforo

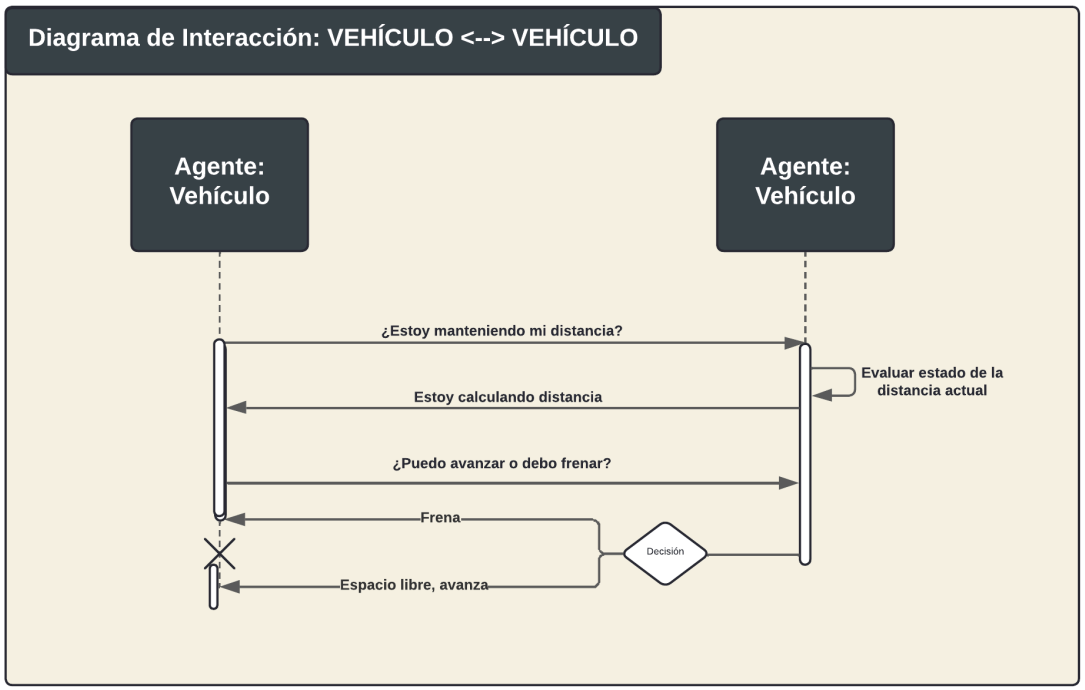


Imágen 3. Diagrama de interacción Semáforo Semáforo

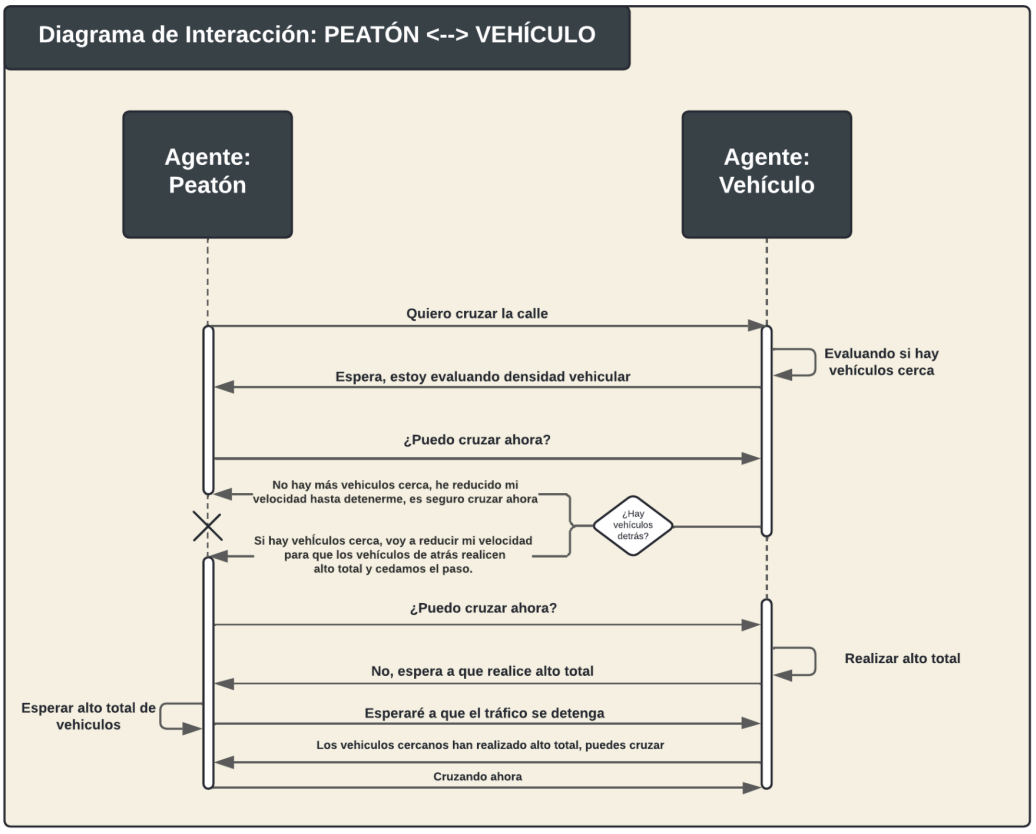


i

Imágen 4. Diagrama de interacción Vehículo Semáforo



Imágen 5. Diagrama de interacción Vehículo Vehículo



Imágen 6. Diagrama de interacción Peatón Vehículo

7. Planeación de trabajo

7.1. Roles y responsabilidades

Como se mencionó en la primera revisión, se continúa con la delegación de líderes o encargados principales en ciertas áreas, con el objetivo de contribuir y ser más productivos con las funcionalidades, además de distribuir la carga de trabajo, solo para recordar dicha distribución de trabajo en términos generales es la siguiente:

- **Sebastián Espinoza:** Diseño gráfico, Manejo de objetos 3D, Desarrollo de código en simulador gráfico, Diseño de entorno virtual de simulación.
- **Ulises Jaramillo:** Calidad de Software, Desarrollo de código en simulador gráfico, Testing, Manejo de objetos 3D, Diseño Gráfico.
- **Jesús Guzmán:** Desarrollo Back-End, Desarrollo de agentes, Diseño de entorno virtual de simulación.
- **Julio Vivas:** Planificación, Calidad de Software, Desarrollo Back-End, Desarrollo de agentes, Configuración de interconexión.

Esta distribución de trabajo solo delega encargados en áreas que identificamos importantes en el desarrollo del proyecto, pero como se mencionó anteriormente, todos y cada uno de los integrantes del equipo nos comprometemos a participar en el desarrollo de cada funcionalidad, así como de comprender cada aspecto relevante del mismo.

7.2. Planificación general de la entrega

La planificación, refiriéndonos a plazos queda de la siguiente forma:

- **Semana 1:** Introducción teórica e identificación de fortalezas y áreas de oportunidad.
- **Semana 2:** Planeación de trabajo, definición inicial de contenidos y funcionalidades del proyecto.
- **Semana 3 (Estado actual):** Modelación de agentes, gráficas en tres dimensiones y animación gráfica en tres dimensiones (individual por objeto no inteligente).
- **Semana 4:** Interacción entre agentes y animación gráfica en tres dimensiones (sobre la interacción en objetos inteligentes).
- **Semana 5:** Integración final de sistemas y protocolos de interacción, testing final y preparación de entrega.

La organización y distribución de trabajo para esta entrega fue la siguiente:

- **Sebastián Espinoza:** Revisión de Diagramas, desarrollo de código en servidor y Unity, desarrollo de modelo en mesa.
- **Ulises Jaramillo:** Revisión de Diagramas, desarrollo de código en unity y diseño del mapa

- **Jesús Guzmán:** Revisión de Medio Ambiente y PEAS, desarrollo de modelo en mesa
- **Julio Vivas:** Revisión de medio ambiente, desarrollo y corrección de modelo en mesa, desarrollo de servidor.

Realizamos un trabajo colaborativo, pero cada integrante dando más énfasis a su trabajo asignado, la aportación y esfuerzo fue equivalente para esta entrega.

Para esta entrega estimamos el tiempo de desarrollo de toda la semana hasta el jueves 22 de noviembre, sin embargo, por peso de la demás carga académica, nos tomó en general 1 día adicional el desarrollo y revisión completa de la entrega. Sin embargo, la estimación que hicimos fue correcta ya que al agregarse un día adicional de trabajo, pudimos sobrellevar de manera muy relajada la entrega.

7.3. Progreso 60% SMA

El desarrollo realizado para la entrega de esta revisión con respecto a la parte de SMA se desglosa de la siguiente forma:

- **Restructuración de mapeo y puntos de interés:** Realizamos una reestructuración en la forma en la que se ingresa el mapa original, adaptado a las mediciones de mesa, esto a través de un parser que genera la cuadrícula y los agentes en dichas posiciones. Utilizando una lista de listas que almacena las direcciones, y tipos de edificios o posiciones.

- **Generación de agentes estáticos:** Una vez conocidos los puntos de interés estáticos, estos se inicializan como agentes en sus debidas celdas para ser monitoreados o para representar visualmente los elementos de la ciudad.

- **Integración de diseño en caminos para identificar direcciones:** Además de dicha lista anteriormente mencionada, se complementa la información del entorno con los puntos de interés especiales, como salidas y entradas de los estacionamientos y posibles dobles direcciones de algunas celdas vecinas válidas.

- **Reconfiguración de movimiento de agente *SimpleCar*: Movimiento random a Movimiento "Inteligente":** Inicialmente se contaba con un sistema de movimiento/navegación rústica en la que el agente *SimpleCar* elegía de las posibles posiciones válidas cercanas a su posición real (los vecinos) de forma aleatoria, eventualmente llegando a su destino final, dicha navegación no resolvía eficazmente la problemática de navegación. Es por esa razón que se complementa dicha movilidad con funciones auxiliares pertenecientes a la clase del agente *SimpleCar* que a partir de un diccionario de puntos con posibles direcciones se generó un grafo que nos permite realizar algoritmos de búsqueda para poder encontrar rutas óptimas, haciendo al

agente “inteligente” y capaz de identificar mejores rutas o re-calcular en caso de obstrucción.

- **Reconfiguración de agente *TrafficLight*: Estado temporizado a estado “Inteligente”:** Inicialmente el agente *TrafficLight* realizaba los cambios de estado por temporizador estático, es por ello que se optó por desarrollar una nueva funcionalidad en dicho agente, que permite comunicarse con el agente de su mismo tipo más cercano, negociando el cambio de estado por la cantidad de agentes *SimpleCar* en espera una respectiva dirección, para cambiar el estado del semáforo y/o aumentar la duración de algún estado del mismo.

- **Generación de grafo de posiciones válidas:** Se utilizó el módulo de *Networkx* para generar un grafo dirigido, donde el nodo es la celda (x, y) y las aristas son las direcciones posibles desde un nodo específico, estas determinadas por una clase auxiliar de utilidad realizada que permite identificar y aplicar transformaciones de movimiento. Este facilitó la generación de ruta más corta haciendo uso de las funciones que el mismo módulo proporciona, esta posteriormente con una función se transforma en una lista de direcciones que son enviadas al *SimpleCar*, donde esté siguiendo las normas de tránsito llega a su destino en menor tiempo y más eficientemente.

- **Integración de Slider para selector de cantidad de agentes SimpleCar a visualizar en simulación:** Para facilitar la visualización del modelo, integramos un slider interactivo para seleccionar la cantidad de autos y probar dicho modelo en el servidor que genera *MESA*.

- **Código documentado y estandarizado: Se realizó la documentación por función y estructuración limpia de código según las normas de estandarización de código:** Esto permite mantener una legibilidad y verificación de progreso del desarrollo del proyecto.

Como faltantes o pendientes por integrar a la simulación identificamos los siguientes puntos:

- **Complemento del listado de puntos y direcciones válidas:** Para mejorar la eficiencia de la funcionalidad del recálculo de ruta en el agente *SimpleCar*, se piensa complementar algunas de las direcciones posibles de nodos específicos.

- **Integración de agente *Pedestrian* al modelo:** Para simular el movimiento e interacciones de un peatón con los demás elementos del entorno, se tiene planeado realizar una integración del *Pedestrian* donde este pueda moverse entre las construcciones, pedir el paso y/o cruzar el cruce peatonal.

- **Comunicación entre agentes *Pedestrian - SimpleCar* y *Pedestrian - TrafficLight*:** Una vez implementado el agente *Pedestrian*, se planea realizar para cada agente un sistema de comunicación y negociación.
- **Implementar una jerarquía de clases con *Vehicle* como clase base y *SimpleCar* como clase derivada para organizar diferentes tipos de vehículos:** Para mantener un orden en el desarrollo del sistema, es por ello que pensamos en la situación de implementar más vehículos con diferentes funciones, wow.

- **Interconectar el sistema de MESA con Unity a través de un servidor *Flask para* realizar la integración de las modificaciones en tiempo real del servidor:** Conectar en tiempo real las rutas y posiciones de agentes así como sus estados a través de un servidor en Flask.

Es por lo mencionado anteriormente, que consideramos que llevamos aproximadamente entre un 65% y 75% de avance en cuanto a lo que SMA corresponde, el modelo junto con el servidor de visualización son completamente funcionales y se encuentran en el repositorio de GitHub.

7.4. Progreso 60% Gráficas Computacionales

Para la presente entrega, en la parte de gráficas se entrega todo lo siguiente:

- **Código del servidor. (App.py):** Código en Flask con la API de rutas. Al momento de la entrega tiene rutas predefinidas para realizar los tests necesarios. Sin embargo, cuenta con el formato que se recibirá a partir del modelo de mesa para poder se integrado de manera sencilla.

- **Código para mover un solo vehículo. (Maze.cs):** Dentro de Unity, se tiene el código para mover un solo agente con sus transformaciones necesarias completas, código utilizado para pruebas que se adaptará para convertirse en la visualización de los peatones en el desarrollo del 40% restante del reto.

- **Código para obtener una sola ruta (Route.cs):** Dentro de Unity, se tiene el código que obtiene una sola ruta y la guarda en una clase de datos, la cual el código Maze utiliza para la ruta del agente.

- **Código para ver las transformaciones de múltiples agentes a partir de la respuesta del servidor (CarController.cs):** Código que se le asigna al prefab en unity para que sepa el comportamiento que debe tomar cuando se instancie y siga una ruta obtenida del servidor utilizando RouteManager. Este código ya es de utilidad completa para los vehículos y puede ser ligeramente modificado para que se pueda aplicar en los peatones.

- **Código para administrar las rutas obtenidas (RouteManager):** Código que ya puede recibir más de una ruta, además de un identificador del agente que le corresponde. Se conecta correctamente con el servidor para la obtención de los datos.

- **Escena en Unity con elementos de iluminación 85%, corrección de dimensiones, edificios finales:** Todo esto se entrega en el repositorio del equipo. El código del servidor se encuentra en la carpeta server, mientras que los demás códigos, pueden ser encontrados en el unitypackage “SMA_Visualizador_Equipo6”.

Por lo anterior, podemos concluir que hemos completado alrededor de 65-75% del trabajo en Unity. Contamos con código funcional para manejar las transformaciones de los agentes basado en los datos recibidos del servidor en Flask. Además, el esqueleto para las implementaciones restantes está listo, proporcionando una base sólida para la etapa final de desarrollo.

En cuanto al diseño del mapa, este se encuentra casi terminado, quedando solo por ajustar detalles menores como la señalización. Según el plan de trabajo, avanzamos conforme a lo programado, con un progreso constante que asegura la finalización exitosa del reto sin ningún problema.

Sobre el 30-35% restante, podemos identificar:

- Adición de señales de tránsito
- Colores en semáforo de acuerdo a su estado en mesa
- Revisión y adaptación del código de transformación del vehículo para peatones.
- Interconexión con mesa

7.5. Actividades pendientes

En cuanto a las actividades pendientes próximas identificamos las siguientes, listadas por prioridad (alta - baja):

- Interconexión de mesa al servidor
- Complemento del listado de puntos y direcciones válidas.
- Integración de agente *Pedestrian* al modelo.
- Comunicación entre agentes
- Jerarquía de clases
- Adición de señales de tránsito
- Colores en semáforo de acuerdo a su estado en mesa
- Revisión y adaptación del código de transformación del vehículo para peatones.

Las tareas anteriores conforman el porcentaje restante por desarrollar para la conclusión correcta del reto en tiempo y forma. De acuerdo a nuestra planeación de trabajo, continuamos teniendo un buen ritmo de trabajo y esperamos la finalización del reto la próxima semana 5.

En cuanto a los responsables de la realización de tareas, hemos coincidido que todos trabajaremos en las tareas pendientes, debido a que son más enfocadas a la integración del proyecto y requieren de la participación total del equipo, utilizaremos los roles asignados para definir líderes de tareas de acuerdo a las necesidades. Siguiendo este orden de trabajo grupal en cada tarea, esperamos tener la finalización del reto para el miércoles 27 de noviembre.

7.6 Aprendizaje adquirido

En cuanto a los aprendizajes adquiridos, logramos comprender varios errores que estábamos cometiendo al desarrollar el código del simulador, además de lograr integrar un modelo tridimensional, completamente espejo, del proporcionado para las simulaciones, por lo que podemos determinar que fue un gran salto a comparación de las entregas pasadas, dados todos estos conocimientos adquiridos logramos alcanzar y superar en ligera medida el 60% del avance del reto, Identificando las fortalezas de cada integrante del equipo, logramos todos tener una comprensión del proyecto mejor, gracias a la comunicación efectiva. Entre los aprendizajes destacables, podemos mencionar:

- Desarrollo de servidor en Flask para comunicación con Unity.
- Conversión de datos JSON a transformaciones con operaciones vectoriales en Unity.
- Desarrollo de Modelos complejos con MESA y Python.
- Diseño de diagramas de comunicación entre agentes.
- Implementación de grafos y algoritmos de búsqueda en situaciones reales.