
Software Requirements Specification

for

Traffic Light Simulator

Version 1.0 approved

Prepared by Yael Octavio Perez Mendez

José Eduardo Rosas Ponciano

Emiliano Caballero Mendoza

Manuel Olmos Antillón

Instituto Tecnológico de Estudios Superiores Monterrey

Thursday November 27, 2024

Table of Contents

1. Descripción del sistema multiagente	2
1.1 Descripción de Problemas	2
1.2 Descripción PEAS	2
1.3 Diagrama de agentes	6
1.4 Protocolo de interacción entre agentes	7
2. Instalación	7
1.5 Liga de Github	7
1.6 Descripción del Proceso de Instalación	7
1.7 Configuración	7
3. Análisis de la solución desarrollada	8
1.9 Yael Octavio Perez Mendez	8
1.10 José Eduardo Rosas Ponciano	8
1.11 Manuel Olmos Antillón	8
1.12 Emiliano Caballero Mendoza	8
1.13 Reflexión Aprendizaje	8
Apéndice A. Videos	13
Apéndice B. Código Mesa	14
Apéndice C. Código Unity	15

Revision History

Name	Date	Reason For Changes	Version
Versión 1	27/11/2024		

1. Descripción del sistema multiagente

1.1 Descripción de Problemas

El desafío se centra en abordar uno de los problemas más críticos de las ciudades mexicanas: la congestión vehicular y sus efectos adversos en la economía, el medio ambiente y la salud pública. En México, el uso intensivo del automóvil ha sido asociado con el progreso, pero esta relación ha llevado a un aumento significativo en los Kilómetros-Auto Recorridos (VKT), con un crecimiento alarmante desde los años 90. Este aumento ha impulsado una serie de consecuencias negativas, entre las que se incluyen el smog, un alza en los accidentes y enfermedades, y congestionamientos constantes en las vías.

Para lograr que las ciudades mexicanas puedan competir en un entorno global y mejorar la calidad de vida de sus habitantes, es imprescindible replantear y mejorar la movilidad urbana, incorporando soluciones innovadoras y sostenibles. Este reto propone una simulación gráfica para abordar la congestión vehicular, utilizando un sistema multiagente que permita representar el flujo de tráfico en entornos urbanos. El objetivo de la simulación es reducir la saturación en las calles y optimizar la circulación.

Para poder llegar al objetivo propuesto, una de las soluciones clave es la coordinación inteligente de los semáforos, permitiendo que ajusten sus tiempos de luz verde en función del flujo de vehículos. A través de esta simulación, se puede prever el momento en el que un vehículo cruzará una intersección, lo que permitiría a los semáforos ajustar su tiempo y duración en verde para optimizar el flujo y reducir embotellamientos. Esta propuesta busca contribuir a una movilidad urbana sostenible, priorizando la reducción del tráfico vehicular y mejorando la eficiencia en las ciudades mexicanas.

1.2 Descripción PEAS

<i>Agente</i>	<i>Performance Measure</i>	<i>Environment</i>	<i>Actuators</i>	<i>Sensors</i>
<i>Semáforo Vial</i>	<p><i>Permite controlar la congestión vehicular.</i></p> <p><i>Evita accidentes entre cruces.</i></p> <p><i>Mejora la fluidez del tráfico y seguridad vial</i></p>	<p><i>Calles, cruces vehiculares</i></p> <ul style="list-style-type: none"> <i>90% Accesible, los sensores detectan el tránsito vehicular.</i> <i>90% Determinist</i> 	<i>Luces de color (verde, amarillo, rojo)</i>	<i>Sensores de detección para el volumen de afluencia en la dirección del semáforo.</i>

		<p>a, cambia verde a rojo mediante un temporizador</p> <ul style="list-style-type: none"> • No episódico, tiene funciones independientes. • 80% Estático, el entorno suele ser estático. • Discreto, cuenta con acciones finitas que son los estados verde, amarillo, rojo. 		
<i>Semáforo Peatonal</i>	<p>Permite cruce seguro de peatones.</p> <p>Garantiza la seguridad de los mismos.</p> <p>Tiene una coordinación con el semáforo vial.</p>	<p>Banquetas, cruces peatonales.</p> <ul style="list-style-type: none"> • 80% Accesibles, detectan. peatones • 90% Determinista, detección de peatones y uso de temporizador. • No episódico, tiene funciones independientes. • 80% Estático, el entorno suele ser 	<p>Luces (verde y rojo) para indicar cuándo es seguro cruzar.</p>	<p>Temporizador, sensor de paso peatonal.</p>

		<ul style="list-style-type: none"> estático. Discreto, cuenta con acciones finitas que son los estados verde y rojo. 		
<i>Vehículos</i>	<p><i>Trasladarse a un punto específico.</i></p> <p><i>Minimizar el tiempo del viaje.</i></p> <p><i>Evitar colisiones garantizando la seguridad vial.</i></p>	<p><i>Calles, avenidas</i></p> <ul style="list-style-type: none"> 60% Accesibles cuentan con sensores que proporcionan información del medio ambiente, 70% No determinista ya que está afectado por decisiones del conductor o del tráfico, No episódico, cada decisión es independiente, 70% dinámico ya que cambia de acuerdo al tráfico. 100% Dinámico, hay cambios en el entorno Continuo, como lo puede ser las velocidades y posiciones 	<i>Chasis, motor, dirección, llantas, luces de señalización.</i>	<i>Visión al observar el semáforo vial.</i>
<i>Peatones</i>	<p><i>Trasladarse a un destino.</i></p> <p><i>Minimizar el tiempo de cruce.</i></p>	<p><i>Banquetas, calles, cruces peatonales</i></p> <ul style="list-style-type: none"> 40% debido a que solo 	<i>Movimiento para caminar o detenerse.</i>	<i>Visión al observar el semáforo peatonal.</i>

	<p><i>Cruzar de manera segura.</i> <i>Evitar inferencias con el tráfico vehicular.</i></p>	<p><i>interactúa con los semáforos, peatones y vehículos del cruce donde se encuentra.</i></p> <ul style="list-style-type: none"> ● <i>80% No determinista por acciones que se toman por sí mismo o por otros peatones</i> ● <i>Episodico, actúan a situaciones específicas</i> ● <i>95% Dinámico, existen cambios del medio ambiente que afectan la movilidad</i> ● <i>Continuos, movimientos continuos y adaptables</i> 		
--	---	---	--	--

Tabla 1. Tabla PEAS

1.3 Diagrama de agentes

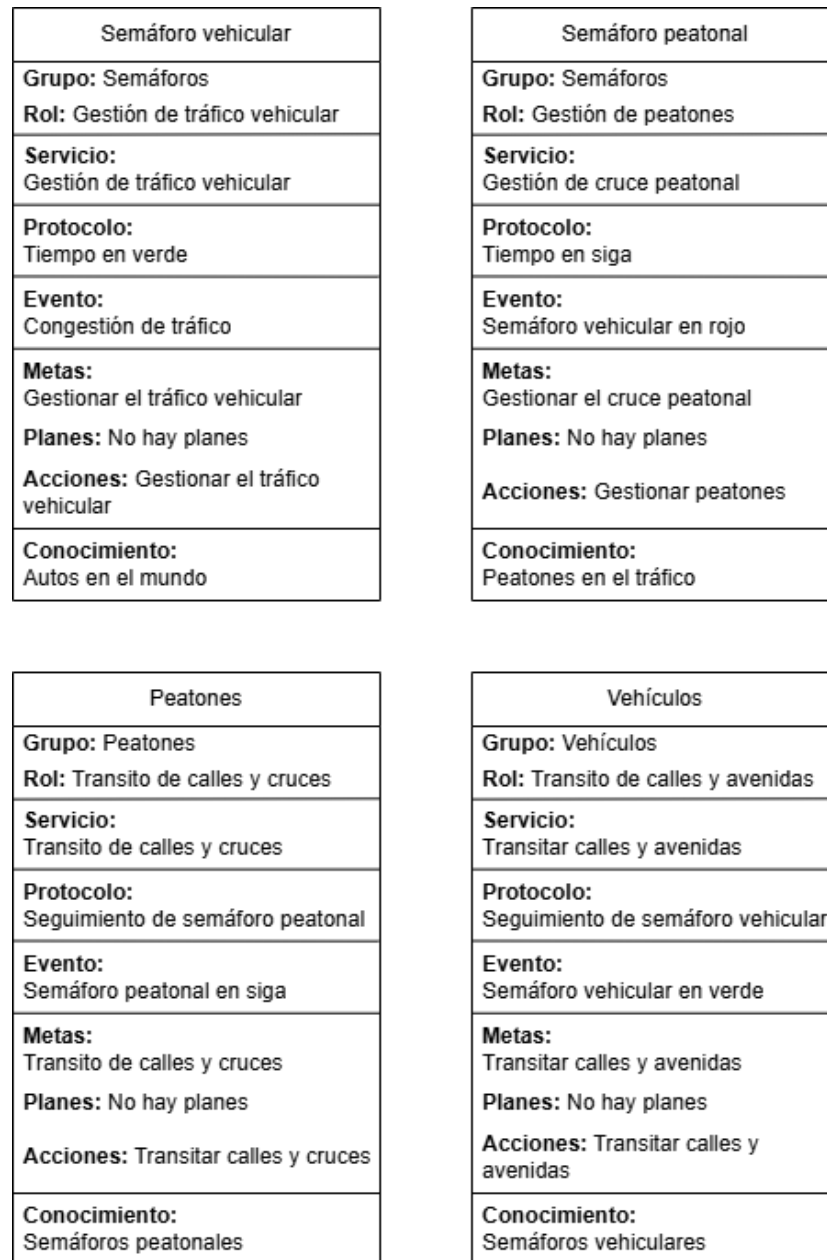


Figura 1. Diagrama de agentes

1.4 Protocolo de interacción entre agentes

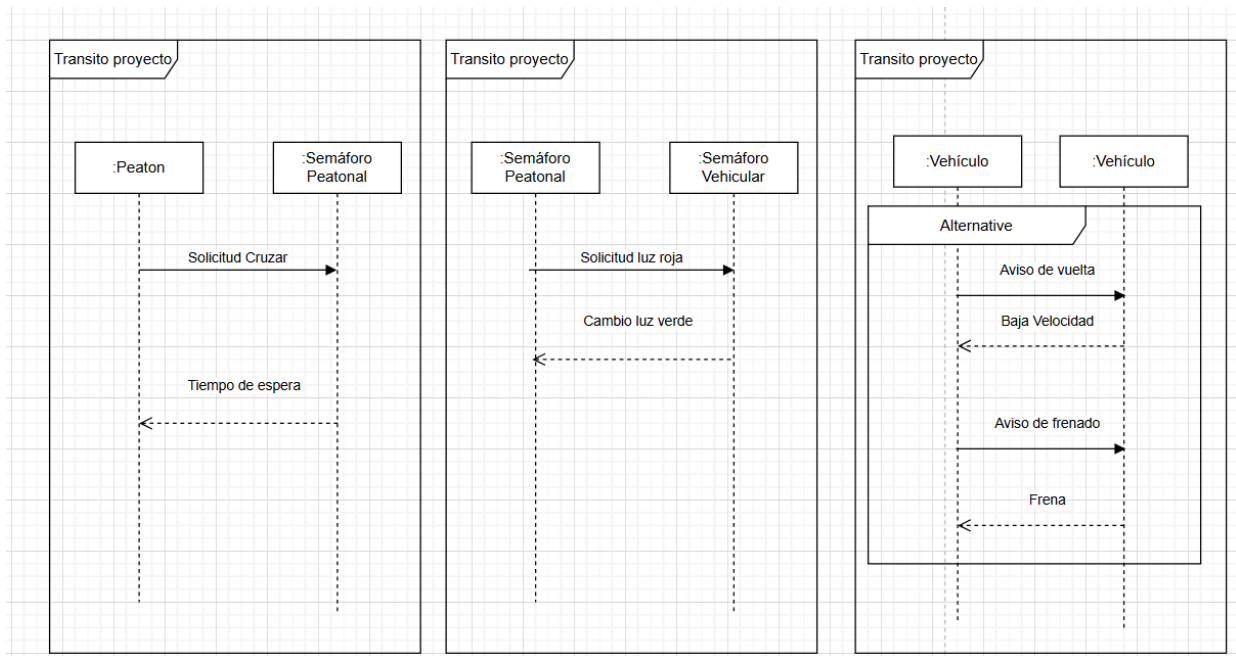


Figura 2. Diagrama protocolo de interacción entre agentes

2. Instalación

1.5 Liga de Github

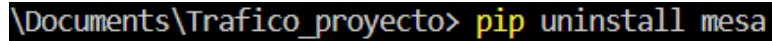
[Yael-PM/Traffic Project: Este repositorio es creado para el proyecto de Modelación de sistemas multiagentes con gráficas computacionales. Contiene el código fuente de la aplicación que tiene como tarea mostrar la simulación de movilidad urbana para solucionar y hacer análisis de diferentes tipos de problemáticas.](#)

1.6 Descripción del Proceso de Instalación

Estos son los pasos y requerimientos para el correcto funcionamiento de nuestra simulación

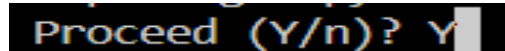
SMA

1. Descargar e instalar Python
2. Descargar e instalar um IDE (Vs code, Anaconda, etc.)
3. Instalar bibliotecas
 - a. Si tienes ya instalado MESA escribe la siguiente línea de comando `>pip uninstall mesa`



```
\Documents\Trafico_proyecto> pip uninstall mesa
```

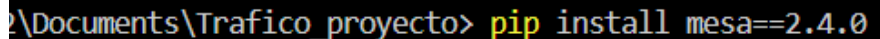
Figura 3. Línea de comando uninstall mesa



```
Proceed (Y/n)? Y
```

Figura 4. Línea de comando proceed (Y/n)

- b. Si no tienes instalado MESA o ya hiciste el paso a, escribe la siguiente línea de comando
>pip install mesa==2.4.0



```
\Documents\Trafico proyecto> pip install mesa==2.4.0
```

Figura 5. Línea de comando install mesa

- c. Instalar biblioteca flask, escribe el siguiente comando > pip install flask

GC

1. Descargar e instalar Unity Hub desde el sitio oficial [Start Your Creative Projects and Download the Unity Hub | Unity](#)
2. Instalar la version (2022.3.53f1) en Unity Hub

1.7 Configuración

La configuración que se mostrará a continuación incluirá pasos a realizar tanto en Unity como en Python, por lo que se recomienda ir siguiendo paso a paso y en todo momento se indicará si es un paso que se tiene que hacer en Unity o en Python.

1. (Unity) Crear un proyecto nuevo con plantilla 3D
2. Se recomienda quitar la luz por default que trae el proyecto.
3. Descargar el archivo .unitypackage
4. En el proyecto, ir a la pestaña de Assets/ Import Package / Custom Package y buscar el archivo descargado. Dar click y esperar.
5. Una vez descargado aparecerá una imagen como la siguiente Dar clic en importar
6. Se abrirá la escena de la ciudad.
7. (Python) Descomprimir el archivo .zip
8. Arrastrar la carpeta hacia el IDLE.
9. Observaras 4 archivos que son los el funcionamiento de los SMA.

1.8 Ejecución de la simulación

Visualización MESA: Para poder visualizar el programa de traffic light se tuvieron que haber echo los pasos ya explicados anteriormente. Una vez realizados:

1. Dirigirnos al IDE con la carpeta ya abierta
2. Abrir la terminal e introducir el siguiente comando > python Mapa.py

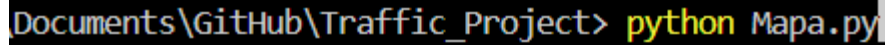


Figura 6. Línea de comando ejecutar Mapa.py

3. Otra opción más intuitiva y fácil es seleccionar el archivo Mapa.py y dirigirte a la flecha que se encuentra en la parte superior derecha del IDLE, como se muestra en la imagen.

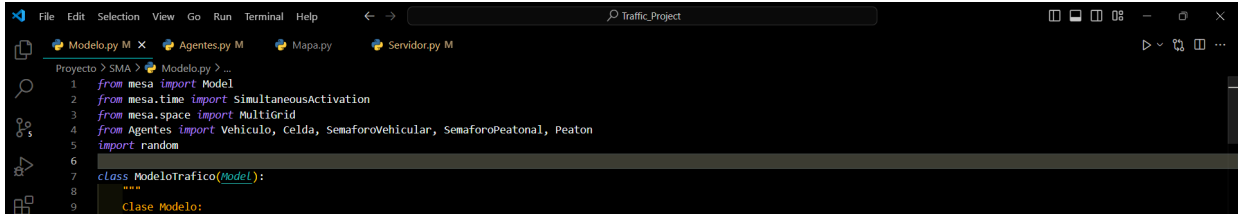


Figura 7. Imagen de como ejecutar desde IDE

4. Una vez ejecutado el programa se mostrará la siguiente visualización.

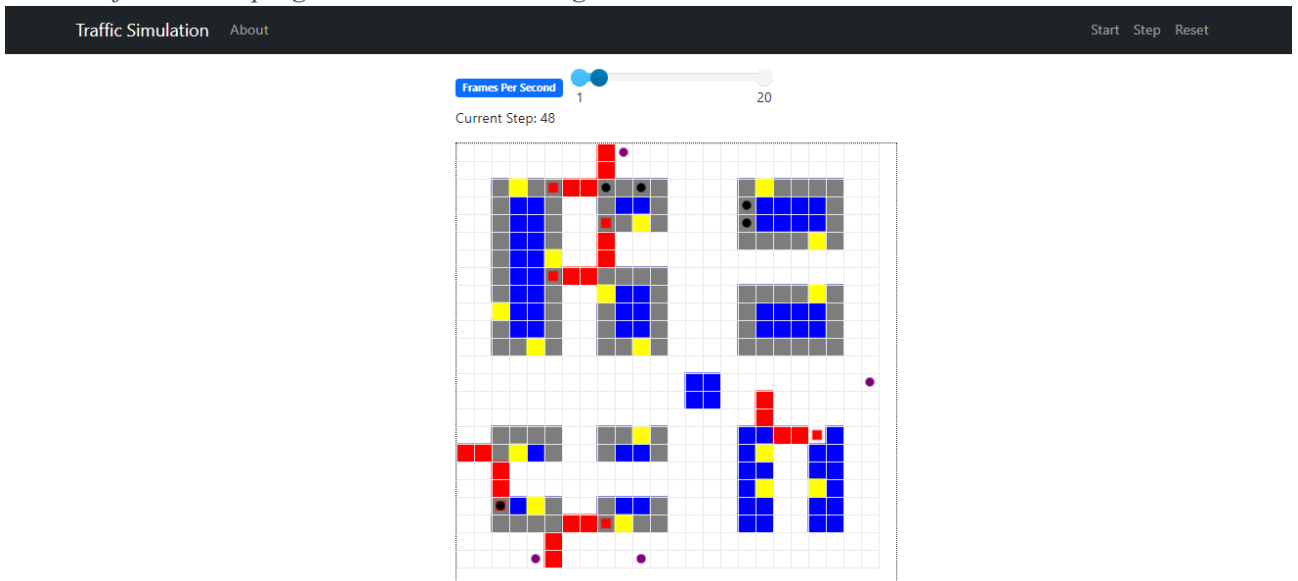


Figura 8. Visualizador Traffic Simulator en MESA

5. Para que la animación pueda ser visualizada puedes optar por dos opciones: Start y Step
6. Una vez que se haya terminado, puedes dar click en reset para volver a realizar la visualización.
7. Si deseas cambiar el número de agentes tanto para vehículo como para peatones. Selecciona el archivo Modelo.py
8. Busca en el archivo Modelo.py, busca el `for i in range(5)` e introduce el número de agentes deseados.

```
for i in range(5): # Cambia el rango según el número de vehículos deseado
    origen = random.choice(list(estacionamientos.values())) # Coordenada (x, y)
    destino = random.choice(list(estacionamientos.values())) # Coordenada (x, y)

    # Asegúrate de que el destino no sea igual al origen
    while destino == origen:
        destino = random.choice(list(estacionamientos.values()))
```

Figura 9. Parte del código Modelo.py para vehículos

```
# inicializar peatones en el modelo
for i in range(5): # Cambia el rango según el número de peatones deseado
    origen = random.choice(self.banquetas)
    destino = random.choice(self.banquetas)

    # Asegurarse de que el destino no sea igual al origen
    while destino == origen:
        destino = random.choice(self.banquetas)
```

Figura 10. Parte del código Modelo.py para peatones

9. Una vez cambiado, guarde con `ctrl + s` y a continuación repita el paso 2

Visualización Unity: Para una correcta visualización en el ambiente unity, es necesario realizar los siguientes pasos:

1. En el IDE python con la carpeta que contiene los archivos MESA
2. Abrir la terminal y escribir el siguiente comando `> python Server.py`

```
\Documents\GitHub\Traffic_Project> python Server.py
```

Figura 11. Línea de comando ejecutar Server.py

3. Una vez ejecutado comenzará a correr el server.

```
Peatón 4: Celda vecina (1, 3) no es transitable.
Peatón 4: Celda vecina (3, 3) no es transitable.
Peatón 4: Celda vecina (2, 4) no es transitable.
Peatón 4: Celda vecina (3, 1) no es transitable.
Peatón 4: Celda vecina (3, 3) no es transitable.
Peatón 4: Celda vecina (4, 1) no es transitable.
Peatón 4: Celda vecina (4, 3) no es transitable.
Peatón 4: Celda vecina (6, 2) no es transitable.
Peatón 4: Celda vecina (5, 1) no es transitable.
Peatón 4: Celda vecina (4, 3) no es transitable.
Peatón 4: Celda vecina (6, 3) no es transitable.
Peatón 4: Celda vecina (5, 4) no es transitable.
Peatón 4 no encontró una ruta válida.
Vehículo 0: origen=(2, 14), destino=(9, 2)
Vehículo 1: origen=(3, 21), destino=(3, 6)
Vehículo 2: origen=(17, 6), destino=(5, 17)
Vehículo 3: origen=(20, 4), destino=(10, 7)
Vehículo 4: origen=(17, 4), destino=(10, 12)
* Debugger is active!
* Debugger PIN: 185-035-691
```

Figura 12. Terminal al momento de ejecutar Server.py

4. A continuación abrirás el proyecto de unity y darás play en la parte superior central

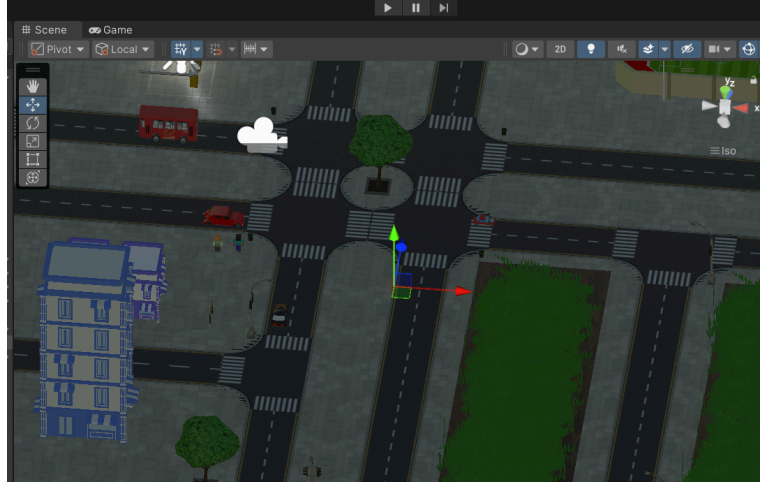


Figura 13. Imagen de la simulación en Unity

3. Análisis de la solución desarrollada

1.9 Yael Octavio Perez Mendez

Para abordar la situación problema, se decidió implementar un Sistema Multiagentes (SMA) reactivo debido a su capacidad para modelar comportamientos autónomos y responder directamente a estímulos del entorno. Este enfoque permitió representar de manera realista las interacciones entre cuatro agentes clave: Vehículo, Semáforo vehicular, Semáforo peatonal y Peatón, logrando una simulación precisa y cercana a situaciones del mundo real. El modelo multiagentes fue seleccionado porque permite gestionar interacciones entre diferentes entidades que operan de forma autónoma dentro de un ambiente. En este caso, los agentes reaccionan de manera inmediata a estímulos, lo que lo hace ideal para representar un sistema de tráfico urbano. Por ejemplo, un vehículo podía comunicarse con otro para identificar si estaba detenido en un cruce por un semáforo en rojo, ajustando así su comportamiento. De manera similar, el peatón interactuaba con el semáforo peatonal, que a su vez se sincronizaba con el semáforo vehicular para garantizar un flujo seguro y ordenado. En conclusión, este modelo reactivo permitió identificar áreas críticas en el diseño urbano y propuso soluciones para optimizar la seguridad y el flujo de tránsito, contribuyendo a una ciudad más eficiente y segura. Sin embargo, existe un margen de mejora en términos de predicción y detalle, lo que puede ser abordado en futuras iteraciones del proyecto.

1.10 José Eduardo Rosas Ponciano

Se decidió optar por el sistema de multiagentes para la solución de este problema ya que se tomó en consideración todos los aspectos en los que se necesitaba una interacción de los elementos dentro de la ciudad, además de la forma en cómo estos interactúan. Por ello, nos dimos a la tarea de definir los agentes necesarios para implementar nuestra solución: Vehículo, Peatón, Semáforo Vehicular y Semáforo Peonatol.

El modelo multiagentes fue seleccionado debido a su capacidad para simular comportamientos independientes y dinámicos, permitiendo una representación realista de la interacción entre agentes y el entorno. Las variables consideradas en la elección incluyeron la densidad de tráfico, las zonas de cruce peatonal, la duración de los ciclos de semáforos y el flujo de peatones. Estas variables interactúan

directamente con los resultados de la simulación, ya que determinan la fluidez del tráfico, la seguridad de los peatones y la eficiencia del sistema urbano.

El diseño gráfico presentado fue seleccionado por su claridad y capacidad para representar de manera intuitiva los elementos clave de la ciudad, como calles, banquetas, glorietas y zonas de cruce. Este diseño facilita la identificación de problemas y permite una visualización directa de las soluciones implementadas.

Las ventajas de la solución final incluyen la flexibilidad del modelo para adaptarse a diferentes configuraciones urbanas, la capacidad de analizar comportamientos emergentes y su utilidad para la toma de decisiones en la planificación urbana. Sin embargo, una de las principales desventajas es la dependencia del modelo en los parámetros iniciales y la complejidad computacional al aumentar la escala de la simulación.

Para reducir estas desventajas, se podrían implementar técnicas de optimización en la calibración de parámetros y usar algoritmos más eficientes para manejar simulaciones a gran escala. Asimismo, integrar métodos de aprendizaje automático permitiría mejorar la adaptación del modelo a escenarios urbanos complejos y dinámicos.

1.11 Manuel Olmos Antillón

Se implementó un sistema multiagente capaz de simular la interacción de diversos agentes en un entorno urbano. Este enfoque permitió modelar una ciudad en la que los agentes coexisten y toman decisiones en función de su entorno y otros agentes. Cada agente fue diseñado con comportamientos específicos y reglas de interacción que imitan escenarios del mundo real.

Un ejemplo destacado es el agente peatón, que decide cruzar la calle únicamente después de interactuar con el semáforo peatonal para solicitar el paso, lo que añade realismo al modelo. Esta capacidad de decisión basada en estímulos externos resalta la flexibilidad del sistema para manejar diferentes tipos de agentes con comportamientos autónomos.

El sistema es escalable, lo que significa que se pueden integrar nuevas funcionalidades y optimizar aspectos como el rendimiento y las interacciones entre agentes. Además, se identificaron áreas clave para futuras mejoras, como la optimización del código, la gestión eficiente de recursos y el desarrollo de interacciones más complejas entre un mayor número de agentes. Estas mejoras permitirían enriquecer la simulación y aumentar su fidelidad respecto a escenarios reales.

1.12 Emiliano Caballero Mendoza

Decidimos optar por el sistema de multiagentes ya que era lo más adecuado para simular un sistema complejo donde existen diversidad de agentes que interactúan entre ellos y en el medio ambiente. Con esto observamos el comportamiento de la ciudad. Se tomaron en cuenta la escala, para representar en el mapa de la ciudad. Visualización, es compatible con herramientas para poder visualizar el comportamiento en tiempo real. Flexibilidad que es la posibilidad de integrar otros agentes. Como se mencionó anteriormente la escalabilidad representa el entorno urbano, la visualización que mejora el cómo se puede visualizar los resultados. Flexibilidad permite agregar nuevas funcionalidades como en nuestro caso el estacionamiento. Seleccionamos el diseño gráfico que es clara la visualización y nos permite identificar todos los agentes puestos, a su vez identificar patrones que permitan ver de forma más clara algún problema con el simulador. Los agentes adoptan un comportamiento realista, comparado con una ciudad. Existe una ventaja en que se puede experimentar sin tanta dificultad con los agentes, permitiendo observar comportamientos que en ciudades conocidas como “Megalópolis” son reales. Un ejemplo sería cuando hay demasiadas personas que no permiten avanzar. A pesar de que existe una facilidad de incluir agentes, la máquina en la que trabajos determinará el límite, por lo cual puede ser una desventaja clara. También a pesar de que el diseño gráfico es bueno, la ciudad puede no ser realista en cuestión del medio ambiente. Mejorar el código computacional para poder hacer más ligero la simulación y en cuestión de gráfico, se podrían usar bibliotecas más avanzadas que permita un modelo que se asemeje a la realidad

1.13 Reflexión Aprendizaje

1.13.1 Yael Octavio Perez Mendez

Como aprendizajes puedo recalcar que durante estas cinco semanas pude desarrollar mi pensamiento modular y sistémico, así como mi capacidad de resolución de problemas enfocadas a las comunicaciones en los sistemas multi agentes. Igualmente, pude mejorar mi pensamiento matemático respecto al álgebra lineal vista en el módulo de gráficas computacionales, lo que me permitió utilizar mi pensamiento crítico al momento de realizar un análisis del comportamiento de cada agente. Además, claramente hubo una mejoría en la resolución de problemas porque ahora tengo una herramienta(los agentes) nueva, que me permite afrontar diferentes situaciones y desarrollar la solución de las mismas.

1.13.2 Jose Eduardo Rosas Ponciano

Considero de gran valor el aprendizaje obtenido en estas 5 semanas de formación, destacando la parte de gráficas computacionales, en el cual reforcé mis habilidades con Unity, aprendí nuevas formas de utilizar este software y otra manera de enlazar y hacer conexiones por medio de API, a su vez la recepción de datos con otro lenguaje de programación.

No sin demeritar a la parte de SMA, en la cual desarrolle mis habilidades con python, y comprendí una nueva forma de solucionar un problema, analizando con antelación la pregunta, ¿Lo resolvemos con objetos o con agentes?

1.13.3 Manuel Olmos Almillon

A través del desarrollo del sistema multiagente, aprendí a modelar comportamientos autónomos y a gestionar interacciones complejas en un entorno simulado. Esto me permitió comprender mejor cómo diseñar agentes que respondan a estímulos del entorno y a optimizar el código para mejorar el rendimiento. Además, identificamos la importancia de llevar un control de versiones estable y organizado. Esto nos ayudó al desarrollo del mismo y la mejora continua. Por lo que me llevo un gran aprendizaje en cuanto al reto, pero sobre todo muchas cosas que mejorar.

1.13.4 Emiliano Caballero Mendoza

Durante estas 5 semanas, espero adquirir conocimientos para implementar y generar modelos de sistemas computacionales, a su vez conociendo los multiagentes y las gráficas computacionales. Esas eran mis expectativas al comenzar al bloque que a mi consideración fueron cumplidas, ya que empecé con cero conocimientos en la parte de SMA y al finalizar este bloque es impresionante lo que pueden lograr, es algo que en verdad llamó mi atención y que puedo decir que estudiaré más a fondo para desarrollar un conocimiento sólido de este tema. Por parte de gráficas, es que fue bastante nutritivo salirnos un poco de la programación en una parte fue bastante retador. Y hacer funcionar esos modelos, darles vida por así decirlo fue algo que me sorprendió.

Appendix A: Videos


Video Instalación y configuración

<https://youtu.be/FZCWoQRMrm0>


Video de la simulación Unity

▣ Videos SMA-GC

Appendix B: Código Mesa

 *Codigo_MESA*

Appendix C: Código Unity

 *Codigo_Unity*