



Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Monterrey

**Evidencia final del Reto de Movilidad Urbana  
Modelación de sistemas multiagentes con gráficas computacionales**



Santiago Andrés Serrano Vacca	A01734988
Erick Daniel Padilla Verdugo	A01740287
Andrea Garza	A00832444
Omar Alonso López Valenzuela	A01284179

---

# DISEÑO DE UN SISTEMA MULTIAGENTE

---

## ÍNDICE

### CONTENIDO

---

Índice	2
<b>1. Introducción</b>	<b>3</b>
<b>2. Créditos</b>	<b>3</b>
<b>3. Contexto y Problema</b>	<b>3</b>
<b>4. Objetivos generales</b>	<b>4-5</b>
<b>5. Restricciones</b>	<b>5</b>
<b>6. Historias de usuario</b>	<b>5-6</b>
6.1 Funcionales	6
6.2 No funcionales	6
<b>7. Descripción del sistema Multiagente</b>	<b>6-11</b>
7.1 Modelo de los Agentes	6-9
7.2 Modelo del escenario	10
7.3 Modelo de la Interacción	11
<b>8. Modelación 3D</b>	<b>12-20</b>
8.1 Descripción de la escena a modelar	12
8.2 Descripción de Componentes Gráficos	14
8.3 Descripción de Prefabs	18
8.4 Descripción de Scripts	20
<b>9. Entregables de administración de proyecto</b>	<b>21</b>
<b>10. Referencias</b>	<b>22</b>

## 1. INTRODUCCIÓN

---

**Describe la situación específica que tu equipo va a resolver (cruce, estacionamiento, rotonda, peatones, etc.), el problema general y cual es la estrategia de solución que van a modelar.**

Nuestro equipo se enfocará en resolver la situación problema de eficientar el tiempo de traslado de un lado a otro. Se modelaron las características de una ciudad transcurrida, en donde se podrá visualizar el flujo del momento de la calle como el movimientos de los vehículos. Con el objetivo de ayudar a minimizar los tiempos de desplazamiento de los ciudadanos desde su casa al trabajo y obteniendo como producto final un programa que informara si el viaje que uno desea hacer se verá más o menos afectado por el tránsito según la hora de partida. Como también, dando instrucciones sobre cómo llegar, utilizando información del tráfico en tiempo real para encontrar la mejor ruta que te lleve a su destino.

El mapa de la ciudad se verá representado por un gráfico dirigido ponderado, utilizando el algoritmo de Floyd-Warshall la cual usaremos para determinar el camino mínimo entre todos los pares de vértices de un grafo, comparando todos los posibles caminos logra mejorar paulatinamente la estimación hasta llegar a la más óptima. También estaremos utilizando el uso de múltiples agentes, modelos y funciones, que se realizará en el lenguaje de Python. Mientras la parte gráfica se realizará dentro de plataforma de Unity , de esta manera podremos ver el funcionamiento y visualización del tráfico y del programa.

## 2. CRÉDITOS

---

### ● Programador:

- Santiago Andrés Serrano Vacca
- Andrea Garza

### ● Gráficos:

- Omar Alonso López Valenzuela
- Erick Daniel Padilla Verdugo

### ● Diseñador:

- Santiago Andrés Serrano Vacca
- Andrea Garza
- Omar Alonso López Valenzuela
- Erick Daniel Padilla Verdugo

### 3. CONTEXTO Y PROBLEMA

---

**Describe el problema que vas a solucionar de forma específica (tipo de crucero, tipo de estacionamiento, con semáforos, otros controles, etc). Investiga en fuentes confiables, soluciones o propuestas de mejora para problemas similares, además de parámetros utilizados en la operación de vialidades de tipo de las que vas a simular.**

La capital mexicana ha ocupado el primer puesto de la ciudad más congestionada del planeta durante muchos años. Además de los desafíos de la planificación urbana, cientos de funcionarios tienen la tarea de lidiar con el tráfico en todos los rincones de la metrópolis todos los días. La investigación muestra que si todos los residentes recuperarán el tiempo perdido en el tráfico urbano mexicano, equivaldría a 11 días adicionales de vacaciones por año. Esta situación afecta a todos los ciudadanos en especial a los que van a trabajar todos los días, ya que los días en que los ciudadanos están atrapados en el tráfico también se traducen en tiempo de producción perdido. Es por eso que nuestro proyecto busca reducir el tiempo de viaje en las rutas requeridas (hacia el área laboral).

El programa calculará la mejor ruta a su destino en función de la congestión en la calle. Para esto, se realizará una animación gráfica que nos funcionará como el simulador del problema, en donde usaremos la interacción entre los agentes para su mayor funcionamiento. Su base predeterminada gráfica será de una ciudad con sus características, se podrá visualizar el movimiento de los automóviles, dirección de las calles, semáforos, etc.. Este producto simulará una aplicación de tráfico y navegación en donde se podrá visualizar las calles con mayor congestión vehicular. Nuestro alcance en esta situación problema es darle al usuario una mejor experiencia en el tiempo de ida y vuelta de su trabajo a su hogar.

### 4. OBJETIVOS GENERALES

---

#### Objetivos:

1. Que el programa pueda visualizar las calles con mayor flujo vehicular, cuando se abra la app. (Mostrar las calles con mayor flujo vehicular en un radio de 2 km desde la ubicación actual calculando la cantidad de agentes que están ubicados en dichas calles en la menor cantidad de tiempo posible dependiendo de la cantidad de datos y la velocidad de la conexión en el momento).
2. El programa calculará la mejor ruta, así para poder reducir el tiempo de ruta hacia el destino deseado.

(Calcular la distancia más rápida tomando en cuenta el tiempo y la distancia de traslado en relación a la cantidad de flujo de cada ruta, en el menor tiempo posible gracias a la ayuda del cálculo del algoritmo de floyd warshall..)

### 3. Visualización del camino por donde el vehículo debe ir.

(Mostrar las 3 alternativas que menor tiempo requieren recorrer en la pantalla del usuario y puede seleccionar la que crea más conveniente y en el momento en que la selecciona se mostrará la ruta que tiene que recorrer.)

### 4. Recacular dirección al ir por el lado erróneo.

(Calcular rutas alternativas después de 5 segundos si la aplicación detecta que no se tomó la ruta indicada)

## 5. RESTRICCIONES

---

**Comenta que restricciones va a presentar el sistema y/o diseño que presentarás como solución. Por ejemplo: Sentido y/o dirección de las calles, los coches pueden dar vuelta o no, cantidad de cajones de estacionamiento a modelar, tipos de cajones de estacionamiento, tiempos de semáforo, tiempos de estancia en estacionamiento, carpooling, etc.**

El sistema contará con las siguientes restricciones:

- Flujo vehicular: El programa calculará el flujo de las calles que pasara el vehículo para llegar a su destino y escoge la mejor ruta.
- Sentido de la calle: El programa recalculará al ver que el vehículo tomó una vuelta errónea y lo mandara por una dirección correcta. \*pendiente\*

## 6. HISTORIAS DE USUARIO

---

Genera las historias de usuario del sistema, tanto multiagentes como en la aplicación gráfica.

**Perfil:** Ciudadano que quiere llegar a su trabajo

**Necesidad:** Ayudar a los usuarios a visualizar el flujo de las calles y encontrar la mejor opción de ruta de llegada.

**Propósito:** Eficientizar el tiempo de llegada hacia el destino.

### 6.1 FUNCIONALES

---

### **Lista de requerimientos funcionales:**

- Controladores que permitirán la navegación dentro de la plataforma de Unity.
- Scripts que dan funcionamiento a la simulación.
- Cambio de color entre las calles para visualizar el flujo vehicular.
- Agente y Entorno del automóvil.

### **6.2 NO FUNCIONALES**

---

### **Lista de historias no funcionales:**

- Animaciones intuitivas para el usuario dentro de la simulación.
- Movimiento de los Vehículos.
- Cambio de color de los Semáforos
- Movimiento de la cámara dentro de la simulación.

## **7. DESCRIPCIÓN DEL SISTEMA MULTIAGENTE**

---

**Explica completamente cómo se va a comportar la solución multiagente. Esta descripción debe contener la idea general y las interacciones generales. En las siguientes secciones se profundizará sobre los temas específicos.**

El sistema utilizará múltiples agentes para su funcionamiento, entre ellos está el agente “Automóvil”, “Semaforo” y “Waze”. También para el funcionamiento de uno de los agentes se utilizará el código de Floyd-Warshall que será primordial para la utilización y funcionamiento del agente “Waze”. Dentro del mismo sistema contendrá la clase “Ciudad” que será un modelo de mesa, esta nos ayudará a representar la estructura de nuestras calles. También, ya contará con algunos valores predeterminados entre ellos está el ancho del carril que será de 3 metros, que representara las calles con dos carriles, cada una representando la dirección opuesta en donde los automóviles estarán pasando. Esta información, simulará el recorrido de “n” cantidad de vehículos en la ciudad, recorriendo cada uno en su dirección. Estos mismos automóviles tendrán diferentes velocidades para dar una mejor experiencia al usuario dentro de la simulación, esto para llegar a nuestro objetivo de ayudar a minimizar los tiempos de desplazamiento de los ciudadanos.

### **7.1 MODELO DE LOS AGENTES**

---

**En esta sección, deberás mencionar todos los agentes que vas a utilizar. Describiendo sus características:**

#### **Automóvil**

- **Creencias:**

Llegar a su destino

- **Planes:**

Transitar la ruta más corta para llegar al destino, reaccionando apropiadamente a los semáforos y el comportamiento de otros conductores

- **Cooperación:**

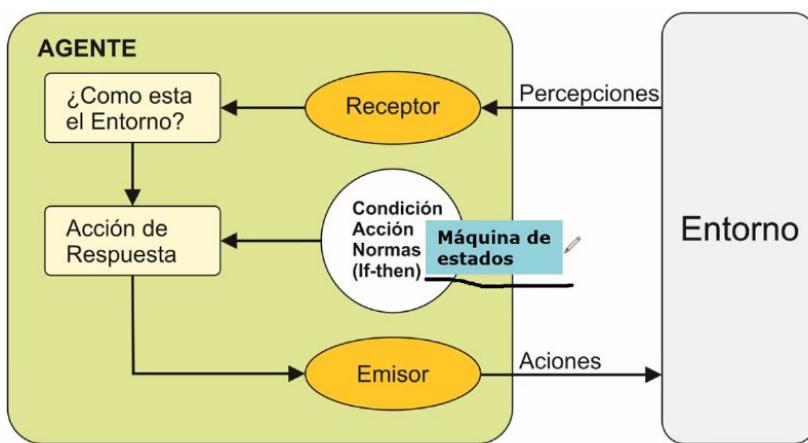
Reaccionar apropiadamente al comportamiento de otros automóviles para ralentizar el tráfico lo menos posible después de cada maniobra.

- **Aprendizaje:**

Conocer la ruta que produzca una menor congestión de tráfico para así tardar menos tiempo, ya sea en cualquier escenario o según la hora y el destino al que se dirija.

- **Modelo Agente:**

## Automóvil



### Percepciones

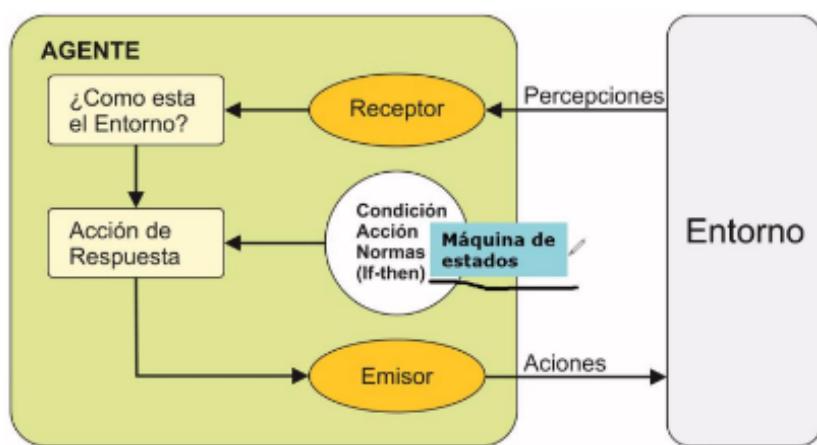
- Ruta más rápida según el “grafo global”. Se va actualizando en tiempo real.
- Coches delante mío.
- Mi propia posición en el mapa.
- Color del semáforo que tengo adelante (rojo/verde/amarillo).
- Distancia al próximo cruce.

### Acciones

- Reportar “Estoy en X arista” al grafo global.
- Reportar “Tiempo que me demoré de nodo A a nodo B” al grafo global.
- Cambiar de carril/adelantar.
- Detenerme.
- Moverme hacia adelante.



## Grafo global (Waze)



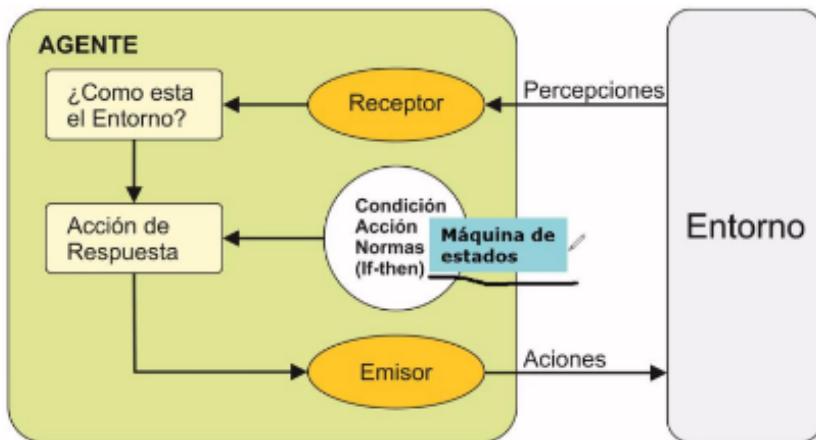
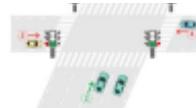
### Percepciones

- Por cada auto, tiempo tardado de nodo A a nodo B.
- En qué arista está cada auto actualmente.

### Acciones

- Enviar ruta más eficiente a cada auto.

## Semáforos de una intersección (los 4, actúan como un solo agente)



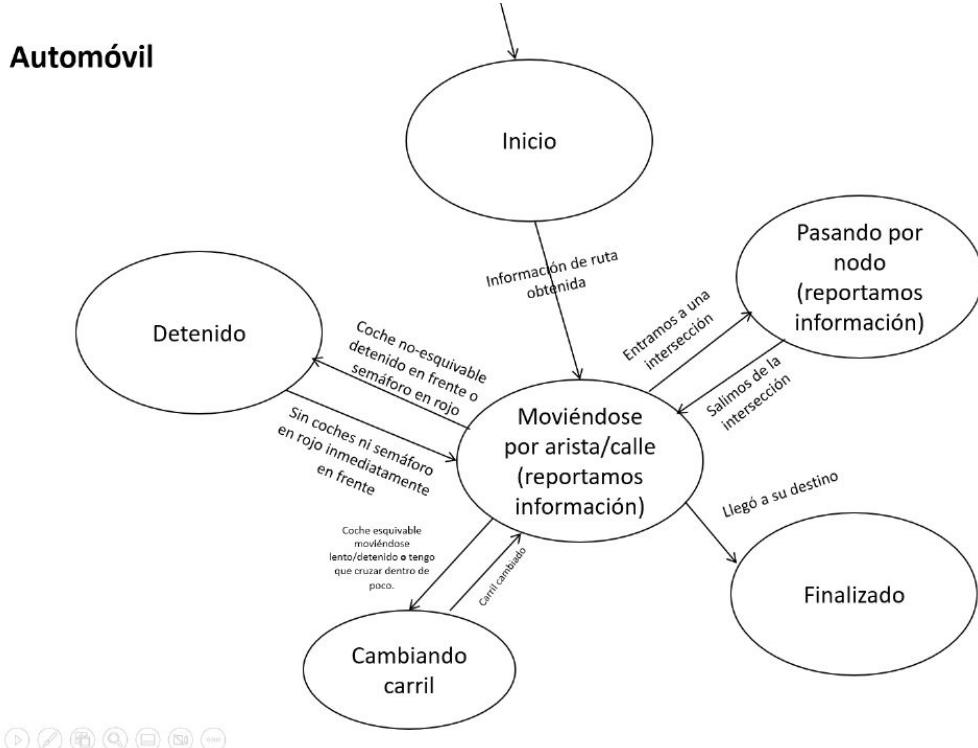
### Percepciones

- Cuántos autos hay en cada una de las vías conectadas a la intersección.

### Acciones

- Cambiar los colores de los semáforos.

- **Modelo de Estado:**

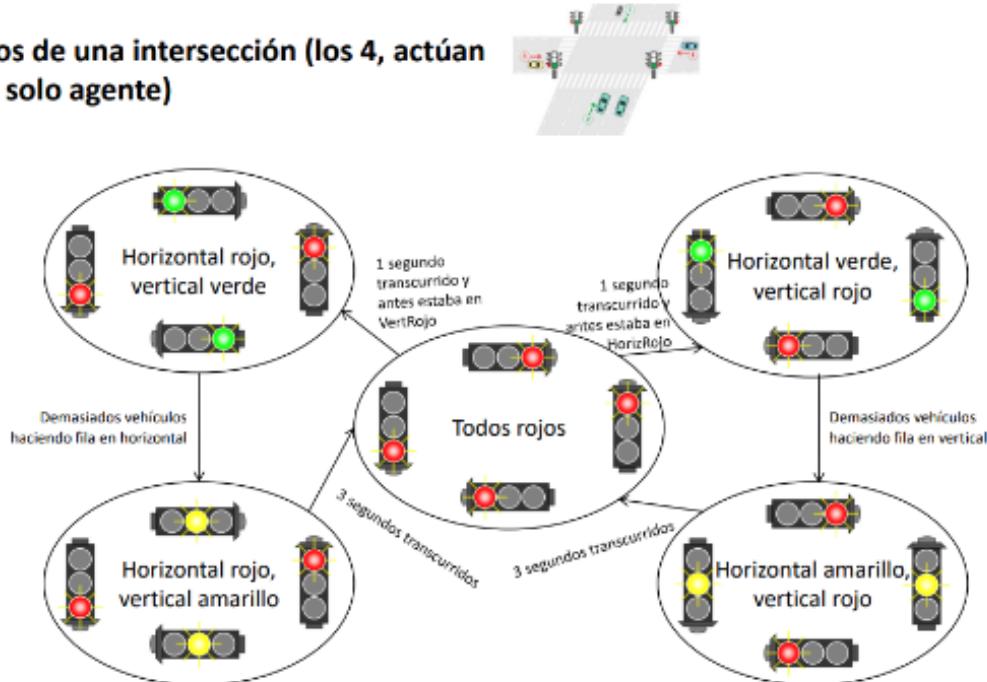


### Grafo global (Waze)



Puesto que este agente tendrá que ejecutar múltiples operaciones en paralelo, y no va pasando de un estado a otro, tiene únicamente un estado.

## Semáforos de una intersección (los 4, actúan como un solo agente)



## 7.2 MODELO DEL ESCENARIO

Explica con detalle el escenario que vas a modelar, es decir, describe el ambiente y cómo se va a modelar en la simulación. Recuerda mencionar las características del ambiente:

1. Determinista o estocástico
2. Estático o dinámico
3. Discreto o continuo
4. Agente simple o multiagente

Comenta cómo vas a manejar el tiempo (con pasos o cómo) y qué estás haciendo para modelar el espacio. Ejemplo: utilizar una malla (matriz de espacios).

La simulación se modelará utilizando los activos de componentes básicos que visualizan el entorno urbano. Será posible visualizar edificios como las bases que contienen las ciudades, y contendrá vehículos en movimiento como una visualización del flujo de calles. Para ello, se utilizan múltiples agentes (Automóvil, Waze y Semáforo) que fueron explicados anteriormente en este documento para programar la simulación. La modelación del mapa se estará representado por un gráfico dirigido ponderado, donde los pesos dependen del tráfico que haya, por lo que podamos utilizar el algoritmo de Floyd-Warshall para calcular la ruta evitando el máximo tráfico posible.

Dentro del modelo ambiental representará carreteras, movimiento de los vehículos, objetos que deban evitarse y por donde sí pueden conducir. De hecho, dentro de la programación se fijó que en el caso de que sea un coche en sentido contrario no será relevante, por lo tanto es imposible chocar ya que cada automóvil irá por su carril. Cada uno de los automóviles se preocupará sólo por el que tenga adelante, por lo tanto los de atrás ya sabrán también cómo comportarse. También se podrá visualizar el movimiento de los vehículos a diferentes velocidades para dar una mejor experiencia y precisión de lo que es una ciudad real. De igual manera contendrá semáforos en donde los vehículos se comportarán de acuerdo con el estatus del mismo semáforo, es decir si el semáforo está en rojo el vehículo frenara, si está en amarillo disminuirá velocidad y si está en verde continuará con su camino.

### 7.3 MODELO DE LA INTERACCIÓN

---

**Describe a detalle la interacción del sistema multiagentes con la simulación en el sistema gráfico, cómo vas a intercambiar la información entre el módulo de multiagentes y la interfaz gráfica. Detalla el uso que le dará Unity a la información que recibe.( Explicar el comportamiento, descripción del proyecto y funcionamiento)(¿como vamos a pasar la info de mesa a unity? y ¿qué info vamos a pasar (coordenada, status, ocupación, etc ?)**

El programa estará conformado por tres clases agentes: Automóvil, Semáforo y Waze. El agente “Automóvil” tiene el objetivo de llegar de un punto A a un punto B en el menor tiempo posible. Esta sección tendrá los siguientes datos predeterminados: la orientación del vehículo y su estado dentro de la simulación. El agente “Semáforo” representa a los cuatro semáforos de una intersección. Cambian de color cuando pasan t segundos o cuando hay más de x coches esperando de un lado de la intersección. Finalmente, el agente “Waze” que sabe cuánto tiempo en promedio se está demorando un auto de un nodo a otro y con base a esto, es capaz de proveer a los automóviles con la ruta más rápida, contiene el grafo dirigido que representa la ciudad, cuyos nodos son las intersecciones y aristas las calles. Para el funcionamiento de este agente se utilizó el código de “Floyd-Warshall” que es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. Este mismo algoritmo es el que encuentra el camino entre todos los pares de vértices en una única ejecución.

Después, tenemos el modelo clase “Ciudad”, que representa tanto el tamaño de la ciudad como la cantidad de automóviles. Este modelo instancia las intersecciones, semáforos, nodos y la matriz de adyacencia, este también alinea a los vehículos a establecer su orientación. Como mencionado anteriormente en el documento se puso el tamaño de las calles como 3 metros, que esto suponiendo que la ubicación de un coche corresponde a su centro, si el centro de una vía de 2 carriles (ida y regreso) está en la ubicación y=100 m, los coches que van al este se moverán por y=98.5m, mientras que los que van al oeste se moverán por y=101.5m. Como mencionado anteriormente dentro del

desarrollo del movimiento se instancian los automóviles e intersecciones como los semáforos, estas instancias estarán mandando mensajes en la programación cada segundo para poder tener la visualización y los datos de las coordenadas de los vehículos como el estado de los semáforos.

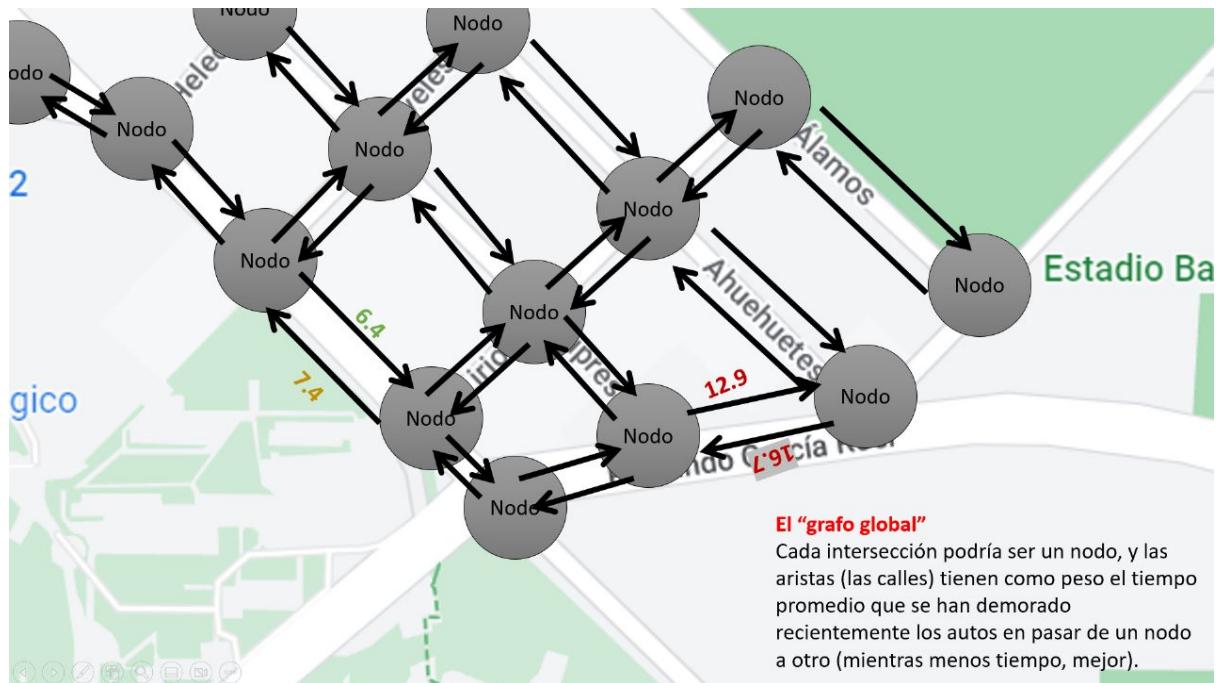
El resultado de esta programación de multiagentes se representará gráficamente en el programa de Unity, mientras que el script de Python será el que procese el sistema multiagentes mientras los dos programas se comunicarán mediante JSON. Para esto, pasaremos la información de las coordenadas de los automóviles y por cada segundo esta compartiría la ubicación de cada carro. Tambien, se pasara el Id de los semáforos (verde[2], amarillo[1] y rojo[0]) y esta tambien seria por cada segundo. Todo esto se pasará con el formato JSON que nos funcionará como puente para transmitir la información realizada en Python hacia la plataforma de Unity. El JSON que se estará utilizando para el medio de comunicación será el siguiente:

```
{
    "car_positions": [
        {
            "id": 0,
            "x": 52.325222,
            "z": 103.5
        },
        {
            "id": 1,
            "x": 154.5,
            "z": 344.234652
        },
        {
            "id": 2,
            "x": 352.32333,
            "z": 204.5
        }
    ],
    "intersection_lights_colors": [
        {
            "id": 0,
            "vertical_on": 0,
            "horizontal_on": 2
        },
        {
            "id": 1,
            "vertical_on": 2,
            "horizontal_on": 0
        },
        {
            "id": 2,
            "vertical_on": 0,
            "horizontal_on": 1
        }
    ]
}
```

## 8. MODELACIÓN 3D

### 8.1 DESCRIPCIÓN DE LA ESCENA A MODELAR

Aquí es necesario agregar de inicio un draft de la escena y explicarla. Posteriormente cuando se tenga diseñada en Unity agregar una imagen del resultado final. Una especie de expectativa vs realidad :)



**Descripción:** La idea es representar una ciudad en 3D y que en el juego se pueda mover la cámara dentro de este entorno. Esta foto proviene del videojuego Cities Skylines, que estaríamos tomando como inspiración para darnos una idea de lo que queremos lograr en un escenario ideal.

## 8.2 DESCRIPCIÓN DE COMPONENTES GRÁFICOS

---

<b>Nombre</b>	Low-Poly Simple Nature Pack
<b>Descripción</b>	Pack que contiene árboles y demás elementos de naturaleza
<b>Imagen</b>	
<b>Autor/Fuente</b>	Unity Asset Store: <a href="https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153">https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153</a>

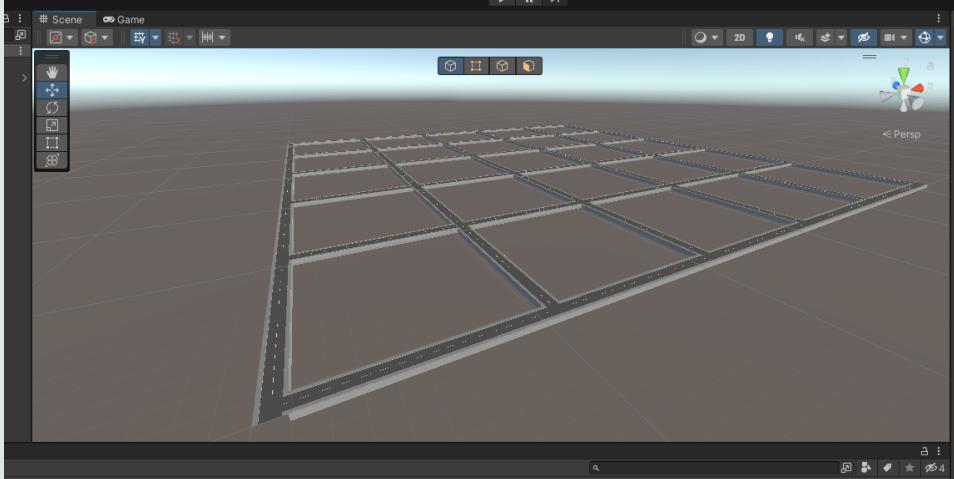
<b>Nombre</b>	Cartoon Low Poly City Pack Lite
<b>Descripción</b>	Pack que contiene calles y edificios de ciudad
<b>Imagen</b>	
<b>Autor/Fuente</b>	Unity Asset Store: <a href="https://assetstore.unity.com/packages/3d/environments/urban/cartoon-low-poly-city-pack-lite-166617">https://assetstore.unity.com/packages/3d/environments/urban/cartoon-low-poly-city-pack-lite-166617</a>

<b>Nombre</b>	Low Poly Road Pack
<b>Descripción</b>	Pack que contiene calles de todo tipo
<b>Imagen</b>	
<b>Autor/Fuente</b>	Unity Asset Store: <a href="https://assetstore.unity.com/packages/3d/environments/roadways/low-poly-road-pack-67288">https://assetstore.unity.com/packages/3d/environments/roadways/low-poly-road-pack-67288</a>

<b>Nombre</b>	Low Poly Buildings Lite
<b>Descripción</b>	Pack que contiene dos casas.
<b>Imagen</b>	
<b>Autor/Fuente</b>	Unity Asset Store: <a href="https://assetstore.unity.com/packages/3d/environments/low-poly-buildings-lite-98836">https://assetstore.unity.com/packages/3d/environments/low-poly-buildings-lite-98836</a>

### 8.3 DESCRIPCIÓN DE PREFABS

---

<b>Nombre</b>	Roads
<b>Descripción</b>	Todas las calles de la ciudad juntas.
<b>Imagen</b>	 A screenshot of the Unity Editor's 3D Viewport. The scene displays a 3D grid of roads, consisting of several rectangular planes arranged in a grid pattern. The roads are represented by blue outlines and grey fills. The Unity interface, including toolbars and a top menu bar, is visible around the 3D view.
<b>Scripts</b>	Ninguno
<b>Autor/Fuente</b>	Erick Daniel Padilla , pero los assets de las calles provienen de: <a href="https://assetstore.unity.com/packages/3d/environments/roadways/low-poly-road-pack-67288">https://assetstore.unity.com/packages/3d/environments/roadways/low-poly-road-pack-67288</a>

<b>Nombre</b>	Carro
<b>Descripción</b>	Un automóvil rojo.
<b>Imagen</b>	
<b>Scripts</b>	Carro.cs
<b>Autor/Fuente</b>	Erick Daniel Padilla

#### 8.4 DESCRIPCIÓN DE SCRIPTS

---

Indicar el uso de cada script y en caso de estar reutilizando código agregar la fuente.

<b>Nombre</b>	InstanciarCoches
<b>Descripción</b>	Coloca 200 carros en calles aleatorias de la ciudad.
<b>Interacciones</b>	Interactúa con los Carros ya que los coloca en su posición, escala y orientación correctas.
<b>Autor/Fuente</b>	Erick Daniel Padilla

<b>Nombre</b>	Carro
<b>Descripción</b>	Hace que el carro avance hacia adelante y dé vuelta en caso de que no haya una calle hacia el frente (evita salirse de la ciudad).
<b>Interacciones</b>	N/A
<b>Autor/Fuente</b>	Erick Daniel Padilla

<b>Nombre</b>	Camera
<b>Descripción</b>	Permite controlar la cámara mediante el teclado y mouse.
<b>Interacciones</b>	N/A
<b>Autor/Fuente</b>	Erick Daniel Padilla

## 9. ENTREGABLES DE ADMINISTRACIÓN DE PROYECTO

- **Liga al Product & sprint backlog:**

<https://trello.com/invite/b/xtaIYimk/ATTIfc5c4e65e53fa7cc1e36b4316e32d8f52AAA59DE/e/quip-1-sprint-product-backlog>

- **Archivos de UNITY (unity package):**

- Escena (con todos los modelos creados).
- Scripts (adentro del unity package).
- **Liga:**

[https://drive.google.com/drive/folders/1Ow1PSjeg\\_z0IfMFJXE8qCZBUKSdCepO?usp=sharing](https://drive.google.com/drive/folders/1Ow1PSjeg_z0IfMFJXE8qCZBUKSdCepO?usp=sharing)

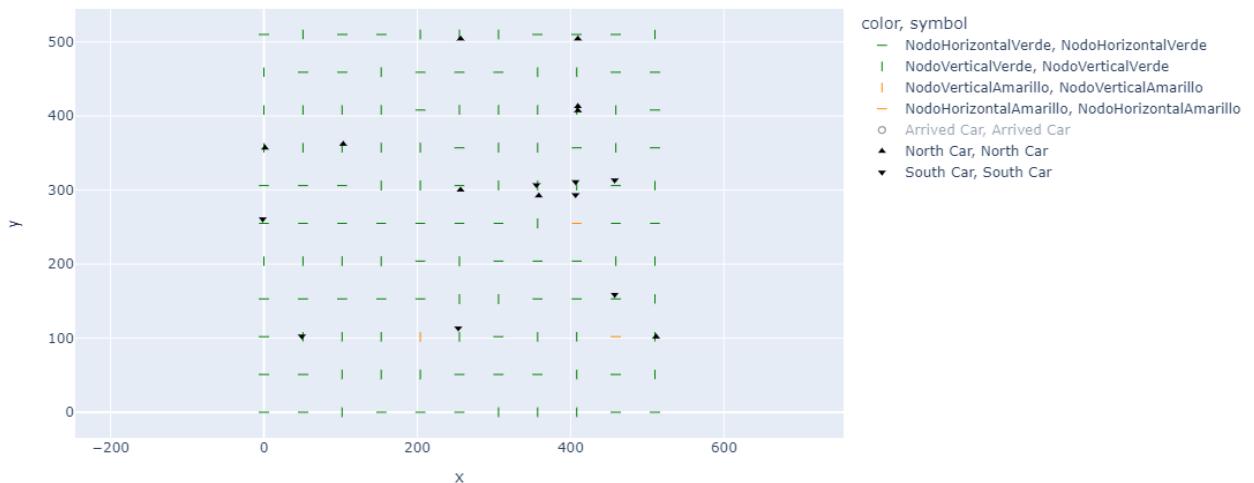
- **Código del sistema multiagente (archivo o archivos python y enlace a Google colab).**

[Para esta entrega aún puede quedar pendiente el tema del servidor para el envío de la información]

- Clases Agente. (Todos los agentes completos)
- Clase Modelo. (Completo)
- Main, generando información que se va a utilizar en la comunicación con Unity.
- **Liga Google Colab:**

[https://drive.google.com/drive/folders/1ZmwI4cqXobz1hHtG4\\_oRBh\\_rZr8HuTj1?usp=sharing](https://drive.google.com/drive/folders/1ZmwI4cqXobz1hHtG4_oRBh_rZr8HuTj1?usp=sharing)

- **Resultado del Main (imagen):**



- **Funcionamiento del sistema Video:**

[https://drive.google.com/drive/folders/1Ow1PSjeg\\_z0IfMFJXE8qCZBUKSdCepO?usp=sharing](https://drive.google.com/drive/folders/1Ow1PSjeg_z0IfMFJXE8qCZBUKSdCepO?usp=sharing)

## 10. REFERENCIAS

---

Anota las referencias que utilizaste en formato IEEE (puedes utilizar este widget:

<https://www.citethisforme.com/citation-generator/ieee>

- La congestión del tránsito urbano: Causas y consecuencias económicas y sociales. (2001). SEPAL, 87. [https://repositorio.cepal.org/bitstream/handle/11362/6381/1/S01060513\\_es.pdf](https://repositorio.cepal.org/bitstream/handle/11362/6381/1/S01060513_es.pdf)
- Política, E. and Galván, M. (2019) *El congestionamiento vial cuesta a los mexicanos hasta 18 días por año*, ADNPolítico. Available at: <https://politica.expansion.mx/mexico/2019/09/10/mexicanos-pierden-18-dias-anual-trafico-imco> (Accessed: January 18, 2023).