

ENTREGA FINAL - RETO MOVILIDAD URBANA

Modelación de Sistemas Multiagentes con Gráficas Computacionales
TC2008B.5

Dra. Raquel Landa Cavazos

Dra. Lorena Martínez Elizalde



Equipo: TECTRAFFIC

A01700519 - Romel Aldaír Vázquez Molina

A01632229 - Carolina Gómez Manzano

A01720476 - David Sánchez Magaña

A01720829 - Jose A. Kaun Sada

A 7 de septiembre de 2021.

ÍNDICE

CONTENIDO

Índice	2
Introducción	3
Créditos	3
Problema	6
Objetivos generales	7
Restricciones	7
Requerimientos	8
Funcionales	8
No funcionales	10
Descripción del Sistema Multiagente	11
Diagrama de Actividad	11
Diagrama de Clases	12
Modelo de los Agentes	13
Agente 1 - Vehículo	13
Agente 2 - Semáforo	15
Modelo del escenario	17
Modelo de la Interacción	18
Modelación 3D	19
Metodología Agile y Comunicación	23
Product Backlog	24
Comunicación	25
Drive	25
Logo	26
Referencias	27

INTRODUCCIÓN

La situación problema de este bloque TC2008B.5 está enfocada al gran número de vehículos que circulan por las calles todos los días, esto generando grandes filas de tráfico en las ciudades y mayores tiempos de traslado para los usuarios. Cada año se tiene un incremento de automóviles circulando, lo que ocasiona cada vez más congestión vehicular, se estima que la Ciudad de México tiene el peor tráfico a nivel mundial, en donde se tienen retrasos de 227 horas por parte de los usuarios[3]. Así mismo, otras ciudades de México también están presentando problemas similares, como lo es Monterrey, la cual está entre las 50 ciudades con más tráfico en el mundo y se estima que los usuarios pierden hasta 144 horas al año en el tráfico[4][5].

Es por esto que se creará una solución que optimice el flujo vehicular de un cruce en forma de cruz de 4 vialidades (2 calles cruzadas), cada una con ambos sentidos y un semáforo respectivamente en cada vialidad, teniendo en total 4 de estos, esta tendrá agentes que se comuniquen entre ellos e interactúen de tal manera que el flujo del tráfico sea lo más eficaz posible. Utilizaremos como agentes semáforos y automóviles los cuales serán programados con ayuda de Python y simulados dentro de Unity. Dentro del modelo estos agentes estarán conectados entre ellos para poder crear una vialidad segura y eficiente para los automóviles.

CRÉDITOS

Romel Vázquez - Programador

Fortalezas

- Programación competitiva
- Desarrollo de modelos matemáticos
- Disciplinado

Áreas de Oportunidad

- Procrastinar demasiado
- Desempeño en la documentación del proyecto
- Colaborar en equipo

Expectativas del Bloque

Espero realmente aprender mucho en estas dos áreas de las ciencias computacionales, sobre todo en la parte de multiagentes, debido a que tengo planeado especializarme en esto.

Carolina Gómez - Programación

Fortalezas

- Conocimiento de Unity
- Trabajo en equipo
- Responsabilidad

Áreas de Oportunidad

- Organización del tiempo
- Procrastinación
- Paciencia

Expectativas del Bloque

De este bloque espero salir con mejor experiencia en crear proyectos de Unity en 3D, diferentes librerías de Python y otros usos que tiene este lenguaje, algunas cosas de diseño, conocimientos técnicos sobre vectores, matrices y otros temas de la materia. Así mismo, trabajo colaborativo desde github y otras plataformas.

David Sánchez - Diseño

Fortalezas

- Conocimiento de python y c#
- Buen manejo de tiempo
- Responsable

Áreas de Oportunidad

- Tardes ocupadas para hacer tarea
- Buscar más calidad para el trabajo
- Documentación de trabajo propio

Expectativas del Bloque

De este bloque yo espero salir con una buena comprensión de cómo la inteligencia artificial puede ser programada a nivel muy básico, para más adelante poder aplicar este conocimiento en algunas cosas más avanzadas.

Jose Alberto Kaun - Diseño y Funcionalidad

Fortalezas

- Conocimiento de JS React, CSS, C++, C# y poco python.
- Responsable y Honesto
- Me gusta fortalecer mis áreas de conocimiento.

Áreas de Oportunidad

- Organización de tiempo
- Procrastinación
- A veces mi rendimiento en clases en línea lo siento bajo.
- Documentación de avances.

Expectativas del Bloque

El bloque del cuarto semestre, llevamos un bloque llamado Construcción de software y toma de decisiones, fue un bloque divertido e interesante lo cual nos motivo a sacar lo mejor de nosotros, esa es mi expectativa para este bloque, quiero fortalecer mis habilidades de programación y de ingeniería para poder crear una solución viable hacia la situación problema.

Qué esperamos lograr

Durante el desarrollo de este proyecto esperamos poder aplicar los conocimientos que iremos adquiriendo a lo largo de estas cinco semanas, con el fin de poder desarrollar un proyecto de calidad que pueda cumplir con el objetivo de mejorar la vialidad urbana. Así mismo, adquirir conocimientos que nos sirvan a lo largo de nuestra carrera y vida profesional, como también habilidades de trabajo en equipo, comunicación y habilidades técnicas.

Compromisos

Para poder cumplir con lo esperado hemos establecido algunos compromisos que nos ayudarán durante el desarrollo del proyecto:

- Entregar en tiempo y forma las actividades individuales de acuerdo al plan de trabajo establecido.
- Entregar en tiempo y forma las actividades grupales.
- Estar al pendiente de la comunicación en el grupo de whatsapp. Así mismo contestar los mensajes y preguntas que se hagan.

- Apoyar a los demás miembros en caso de que una entrega demande demasiado tiempo o sea muy complicada.
- Asistir a las juntas programadas dentro del equipo.
- Cumplir con el trabajo solicitado.
- Respetar las ideas, pensamientos y propuestas de los compañeros.
- Las decisiones se tomarán por mayoría de votos, luego de escuchar todas las opiniones.

PROBLEMA

En la actualidad, existen grandes partes de congestionamiento en el tránsito vehicular, parte de este problema es ocasionado por la programación genérica de los semáforos, donde en horarios de mayor movimiento vial terminan retrasando aún más a los conductores para llegar a su destino.



Imagen 1. Situación actual de la vialidad en México. [2]

El problema que va a ser resuelto es la sincronización automatizada de semáforos en la calle para permitirles gestionar su propio tiempo de acuerdo a la cantidad de vehículos que sean capaces de percibir. De esta manera se podrá asegurar que todos los vehículos en una intersección de cuatro vialidades (dos calles) con cuatro semáforos respectivamente tengan un tiempo de espera menor, al darle prioridad de tiempo de activación a las calles con mayor número de vehículos. [1]

OBJETIVOS GENERALES

El objetivo que buscamos cubrir con nuestra solución es reducir el tiempo de traslado de los usuarios que se mueven en coche en una intersección de cuatro vialidades (2 calles), cada vialidad con su propio semáforo. Esta problemática resulta relevante ya que hoy en día el tiempo que se tarda una persona en llegar a otro punto es un problema debido a que el tráfico en las ciudades ha ido aumentando significativamente. Este aumento en tráfico lleva directamente a un incremento contaminación que a fin de cuentas tienen un impacto muy negativo en nuestro planeta.

Por lo cual se tiene el objetivo de tener un decremento del 10% en el tiempo de espera promedio de vehículos en intersecciones de dos calles cuando apliquemos nuestra solución. Esto será comprobable al visualizar el tiempo que le tomaba al usuario realizar su traslado antes de implementar la solución, y después con la solución en práctica. Para comprobar los resultados planeamos medir el tiempo de 20 vehículos simulados, calculando el tiempo promedio que se tarda en pasar la intersección, contra el tiempo que se obtiene con la solución propuesta.

RESTRICCIONES

Tomando en cuenta que implementar este proyecto será realizado en únicamente cinco semanas, se tomarán en cuenta las siguientes restricciones:

1. La simulación estará construida con un máximo de 20 vehículos, 1 cruce compuesto de 4 vialidades y 4 semáforos.
2. Se utilizarán diferentes señales de tránsito, banquetas, edificios, postes, entre otros objetos; los cuales fungirán como decoración.
3. Los vehículos podrán ir únicamente hacia adelante en sentido recto, es decir que no se podrán dar vueltas entre las vialidades.
4. Se contará con 4 modelos diferentes de vehículos.
5. Las vialidades tendrán el mismo largo y ancho.
6. Las vialidades están conformadas por 2 carriles.
7. Los semáforos tienen las luces verde, rojo, no se incluye la luz para dar vuelta ni la luz amarilla.
8. Los vehículos no cambiarán de carril.
9. A pesar que se incluyen cruces peatonales en la escena estos son únicamente decoración; no hay peatones dentro de la simulación.
10. Los objetos colocados dentro de la simulación en Unity serán obtenidos de assets ya creados dentro de la Unity Asset Store, menos los vehículos.

11. Las calles son conformadas por 4 carriles, en donde cada calle se divide en 2 vialidades, es decir que cada vialidad son 2 carriles. En donde cada vialidad (dos carriles) van hacia un sentido y los otros dos al opuesto. Haciendo que cada vialidad vaya a un sentido diferente.
12. La separación entre vialidades se indicará con líneas amarillas pintadas en la calle donde colindan las vialidades.
13. No va a ser posible realizar esto de manera física, puesto que implementar esto en la vida real representa un gasto significativo tanto para nosotros, como para el municipio. Por lo que se hará una simulación computacional para verificar que nuestra solución es viable. Limitando el número de vehículos, calles, otros objetos y que todas las decisiones de los vehículos son controladas a diferencia de la realidad.

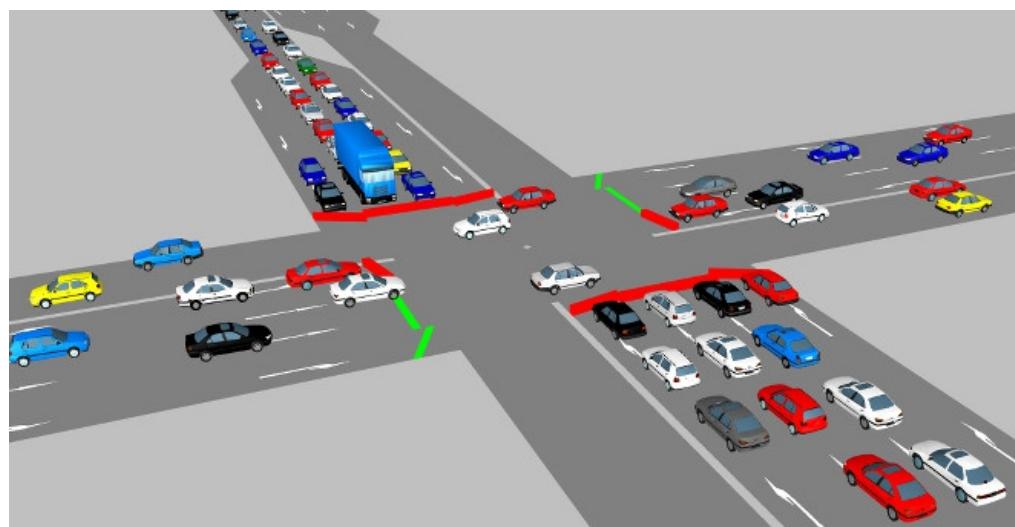


Imagen 2. Ejemplo de implementación de simulación [6]

REQUERIMIENTOS

A continuación se listan los requerimientos funcionales y no funcionales que se tendrán en cuenta dentro del sistema a simular.

FUNCIONALES

1. Agentes	
ID Req.	Requerimiento Funcional

RF1001	Los agentes de los semáforos son capaces de comunicarse entre ellos para coordinar sus tiempos.
RF1002	Los agentes de los vehículos comprenden las reglas de los semáforos y toman acciones dependiendo de la luz mostrada <ul style="list-style-type: none"> a. Verde: continuar b. Rojo: detenerse
RF1003	Los agentes de los semáforos son capaces de contar cuánto tiempo llevan en luz verde o roja y tomar una decisión en base a esta información.
RF1004	Los agentes de los vehículos pueden detectar otros vehículos y determinar la distancia que hay entre ellos para evitar una colisión.

2. Ambiente	
ID Req.	Requerimiento Funcional
RF2001	El sentido de las dos calles será bi direccional (Norte a Sur y viceversa, Este a Oeste y viceversa).
RF2002	Se tendrá un solo escenario con dos calles (4 vialidades), cuatro semáforos y 20 vehículos con interacción simultánea como máximo.

3. Simulación	
ID Req.	Requerimiento Funcional
RF3001	La apariencia de la simulación es en 3D con modelos 3D.
RF3002	Se hará uso de un data frame para tener una comunicación entre el programa Python y Unity
RF3003	Los agentes de los semáforos son capaces de contar cuánto tiempo llevan en luz verde o roja y tomar una decisión en base a esta información.
RF1004	El tiempo promedio de los vehículos pasando por la intersección es medible.

NO FUNCIONALES

1. Equipo de computo

ID Req.	Requerimiento No Funcional
RN1001	Esta simulación está disponible para equipos con sistema operativo Windows 10 o superior.
RN1002	Para la instalación se requiere una capacidad de 5 GB en disco duro.
RN1003	Al momento de ejecutar el programa se consumirá una cantidad de memoria RAM que no supere los 2GB.
RN1004	El sistema podrá trabajar con un máximo de 20 agentes, para esto requiere un procesador que sea al menos intel i5 de novena generación.
RN1005	Dado que el sistema hace uso del procesamiento de imágenes, se requiere tener una tarjeta gráfica nvidia 7900 GS o superior.

2. Simulación

ID Req.	Requerimiento No Funcional
RN2001	La versión de Unity a utilizar es 2019.4.19f1.
RN2002	La simulación será realizada en Unity 3D.
RN2003	El tiempo de espera para iniciar la ejecución será no mayor a 5 minutos.
RN2004	Se utilizarán assets obtenidos de Unity Asset Store junto con assets desarrollados dentro del equipo.
RN2005	El trabajo en equipo se realizará con ayuda de la plataforma de Github.

3. Agentpy

ID Req.	Requerimiento No Funcional
---------	----------------------------

RN3001	Se programará utilizando el lenguaje Python.
RN3002	Se requiere tener una versión de python 3.7 o superior.
RN3003	Se hará uso de librerías externas.

DESCRIPCIÓN DEL SISTEMA MULTIAGENTE

A continuación se muestran los dos agentes necesarios para que el sistema pueda realizar la simulación, así mismo los diagramas que fueron de utilidad y la base para generar la programación de los agentes en Agentpy.

DIAGRAMA DE ACTIVIDAD

En el siguiente diagrama se tienen cinco distintos tipos de actores (Vehículos, Modelo, Semáforos, Servidor y Controlador).

Primeramente el usuario de ambiente, precarga todo lo necesario para que los agentes (Semáforo y Vehículos) de la simulación puedan funcionar al momento de iniciarla. Una vez hecho esto, este sistema seguirá cargando nuevos agentes de vehículos para que estos puedan ser cargados posteriormente a que la simulación comience.

El vehículo tendrá como tarea analizar el ambiente (detectar la luz del semáforo, ver otros vehículos) para evitar alguna colisión. Mientras que el semáforo se encargará de contar la cantidad de vehículos para determinar un tiempo de espera el cual será enviado y recibido por otros semáforos.

Cada una de las posiciones de los vehículos, así como las luces de los carros. Serán enviadas a una base de datos, que estará alojada en un Servidor para que toda esta información pueda ser descargada por un Controlador de Unity y así dar inicio a la animación gráfica.

Todo esto se realizará por una cantidad de tiempo indefinida y será finalizada por el usuario.

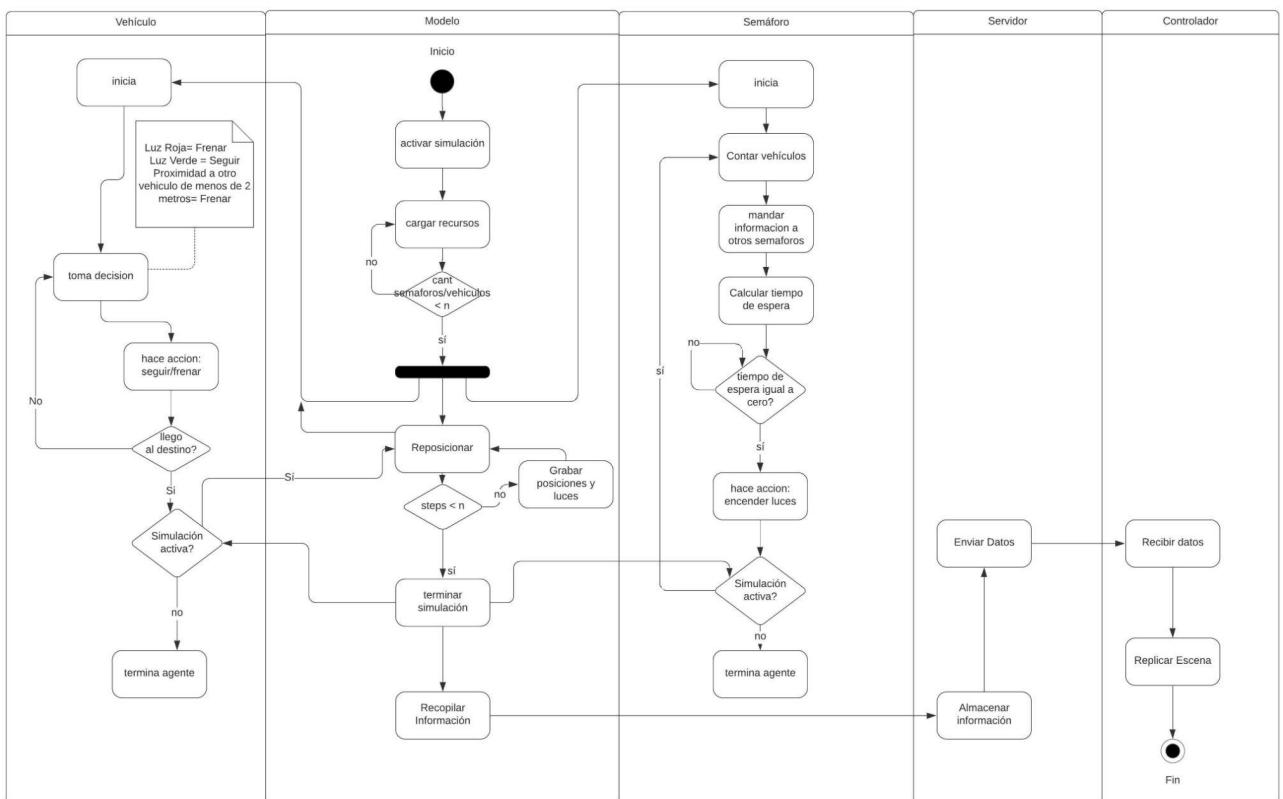


Imagen 3. Diagrama de actividad

DIAGRAMA DE CLASES

En el siguiente diagrama podemos ver los tres diferentes actores que van a estar trabajando dentro de nuestra solución. También podemos ver todos los atributos que contiene cada uno de ellos junto con las funcionalidades que van a poder realizar.

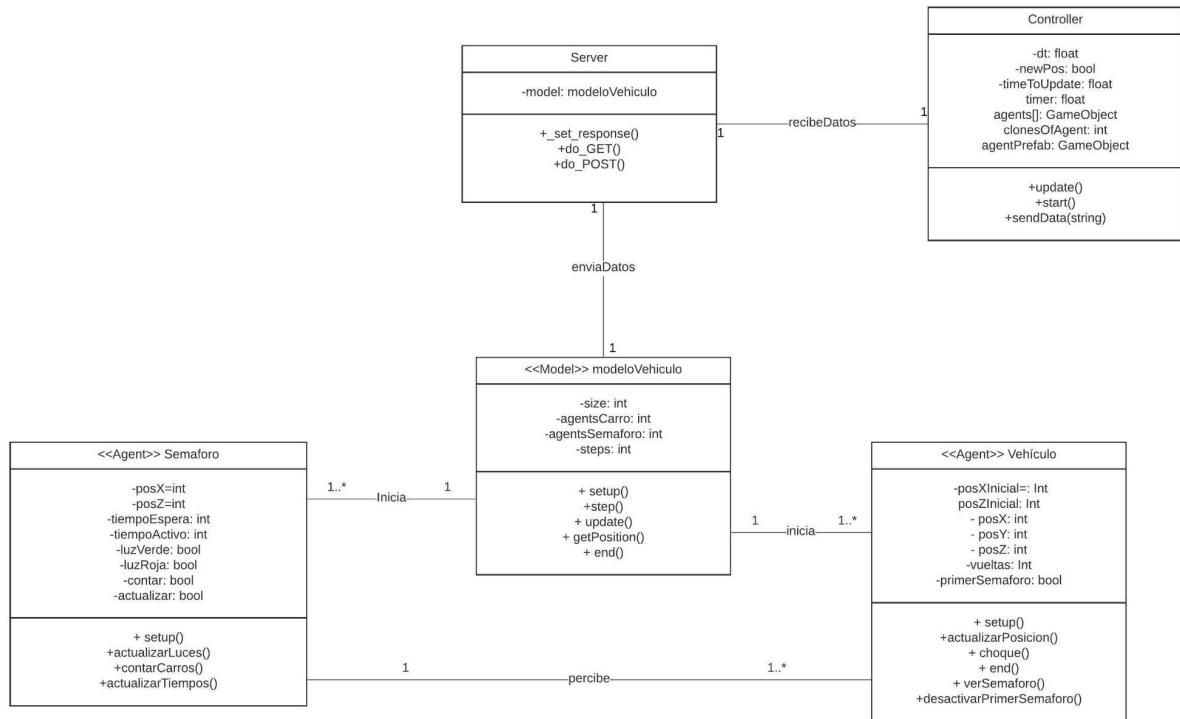


Imagen 4. Diagrama de clases

MODELO DE LOS AGENTES

Para poder plantear una solución más detallada y específica se hizo un modelo de los 2 agentes a utilizar, los cuales son vehículo y semáforo. Estos agentes van a trabajar en conjunto dentro de nuestro sistema para poder implementar nuestra solución. El ambiente dentro del que van a estar trabajando es una intersección de dos calles (4 vialidades) con cuatro semáforos donde los semáforos son parte de un sistema de multiagentes que se van a estar comunicando información, como la cantidad de vehículos que tienen dentro de su vialidad y el estado (color: verde o rojo). Cada semáforo va a tener la capacidad de contar cuántos vehículos hay en su calle y va a poder hacer el cálculo de cuánto tiempo es requerido para que estos vehículos pasen según los números que registran los demás semáforos, es decir que se les da prioridad pero no implica que todos los demás usuarios esperen a que todos los vehículos pasen, si no que se generará un balance. Una vez que tenga este cálculo se lo va a compartir a los otros semáforos para poder coordinar el orden de cuál semáforo le toca prender su luz verde y por cuánto tiempo.

AGENTE 1 - VEHÍCULO

Vehículo

13

Tipo de agente: reactivo

Ambiente: Calle (Presencia de semáforos y/o otros vehículos)

Sensor: proximidad

Actuador: acelerador y freno

- **Creencias**

Este agente tendrá un destino establecido, con la que hará un recorrido por el mapa de la ciudad, permitiendo que entre varios imiten una escena de tráfico. El vehículo podrá avanzar y detenerse cuando encuentre otro vehículo o cuando un semáforo le señale una luz roja. Sus movimientos serán interpretados en coordenadas, en las cuales únicamente se moverá en X y Z, puesto que el escenario es totalmente plano no necesita moverse en Y.

- **Planes**

Este agente recibe las condicionales dadas por el semáforo de su carril, con estas tomará la decisión de quedarse en su lugar o cruzar al otro lado. Las direcciones serán dadas únicamente hacia adelante y con posibilidad de dar vuelta. Así mismo, tomará en cuenta a los otros vehículos y obstáculos para detenerse.

- **Cooperación**

El agente podrá retornar un valor booleano el cual indicará si es que este se encuentra activo o ya haya finalizado su recorrido, permitiendo así llevar un registro de los agentes dentro de la simulación.

- **Aprendizaje**

El agente no tiene aprendizaje automático.

- **Representación gráfica**

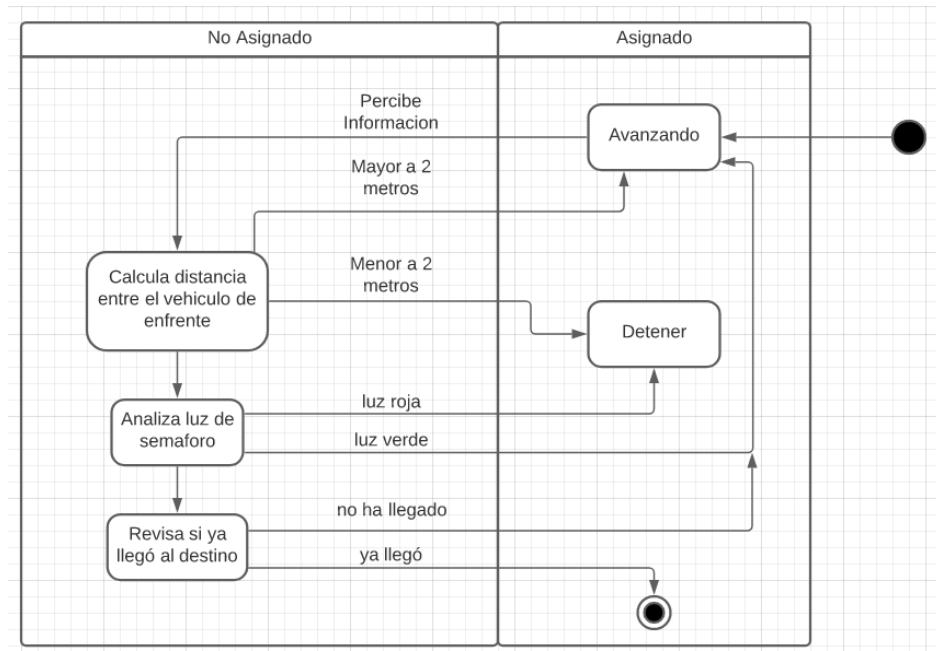
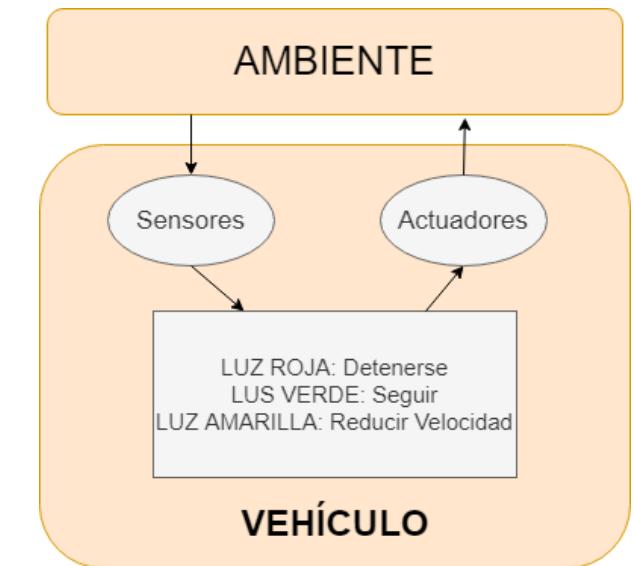


Imagen 5. Diagrama de estados del agente vehículo.

AGENTE 2 - SEMÁFORO

Semáforo

Tipo de agente: modelo

Ambiente: Calle (Cantidad de carros en la vialidad)

Sensor: Cámara

Actuador: Luces

- **Creencias**

Este semáforo tendrá la capacidad de observar y contar los vehículos que se encuentran en la vialidad, por lo que en base a estos podrá definir el tiempo que la luz verde estará activa, así como comunicarlo con otros semáforos para sincronizarlo con el tiempo de espera. Esto con el fin de evitar que haya una luz verde en una calle que no contenga vehículos.

- **Planes**

Por medio de un entero calcula el tiempo que prenderá la luz verde, también recibe un número flotante que determina la cantidad de segundos que seguirá teniendo activa la luz roja.

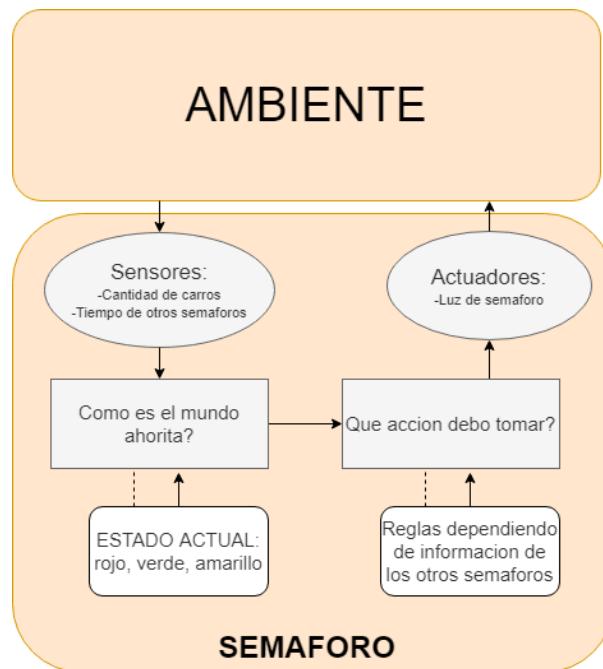
- **Cooperación**

El agente retornará un valor flotante el cual serán los segundos que estará activo en la luz verde, lo cual incrementará el tiempo de espera en los otros semáforos.

- **Aprendizaje**

El agente no tiene aprendizaje automático.

- **Representación gráfica**



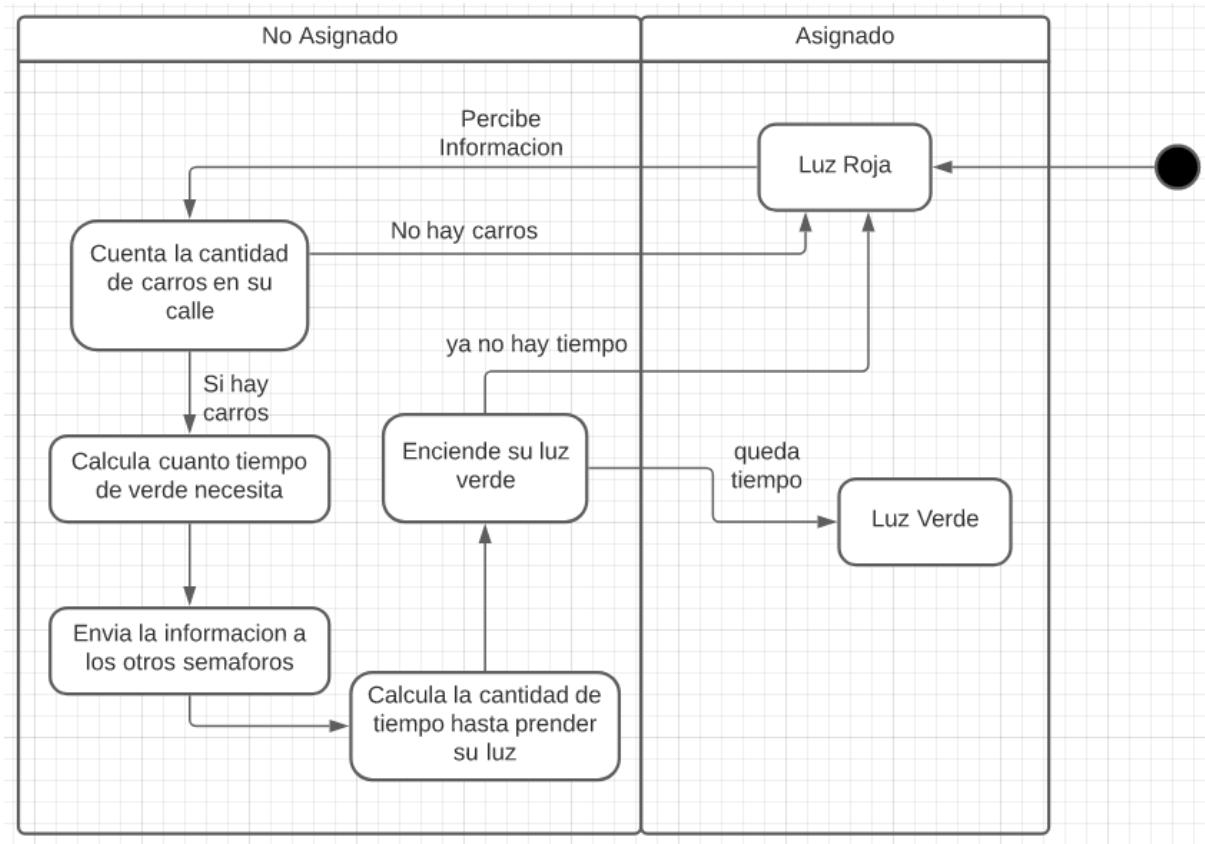


Imagen 6. Diagrama de estados del agente semáforo.

MODELO DEL ESCENARIO

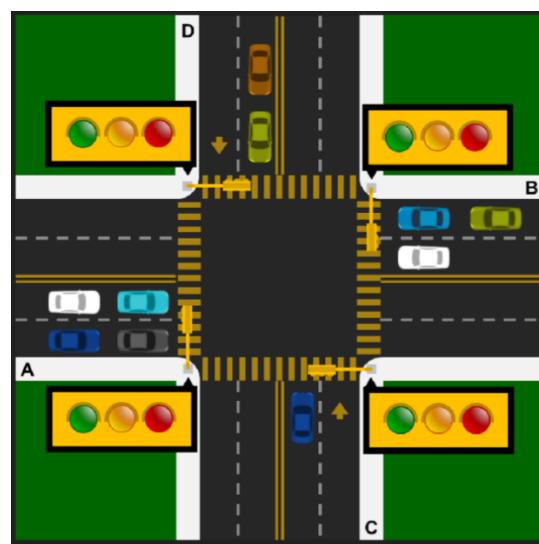


Imagen 7. Gráfico del agente semáforo.

El escenario que se va a simular es similar a la imagen anterior, es decir que consta de un cruce en forma de cruz, en donde cada lado tiene 2 calles (cada una para un sentido diferente), los vehículos dentro de la simulación tratarán de cruzar las calles hacia el sentido correspondiente respetando el semáforo de su carril; se contarán con 4 semáforos para todo el cruce, cada uno en una calle de la cruz. Así mismo, se simularán los vehículos que pasarán por dicho cruce.

Se utilizará una cantidad de vehículos específica, la cual servirá para poder medir la duración, esto se realizará de forma que se empezará a contar el tiempo al inicio de la simulación y se detendrá cuando haya pasado una cantidad definida de vehículos por el cruce. También se obtendrá un promedio de tiempo entre las simulaciones variando el tiempo de los semáforos en verde y rojo.

El tipo de ambiente se describe de la siguiente manera:

- Parcialmente observable
- Estocástico
- Secuencial
- Dinámico
- Continuo
- Multiagente

MODELO DE LA INTERACCIÓN

Los dos agentes tendrán varias interacciones entre sí, así como interactuar con los otros agentes para poder crear un flujo de tráfico eficaz. Por ejemplo, un carro interactúa con otro para que los dos mantengan su distancia para evitar choques, en este caso las luces de freno. Otro mensaje que se presenciará será del semáforo, si es verde puede avanzar, roja se frena y si es amarilla, dependiendo de la distancia que esté el carro puede o avanzar o reducir la velocidad para frenar ya que no tiene tiempo para avanzar. El semáforo también se comunicará con otros semáforos, indicando el tiempo en el que deben de quedarse en espera (luz roja), así mismo para tener la mayor seguridad para los usuarios, se comunicarán para no ponerse en luz verde al mismo tiempo si están cruzados (únicamente se podrá considerar la posibilidad que ambos estén en verde cuando van a la misma dirección).

Una vez que suceden todas las interacciones en el modelo, estas pasarán al servidor, el cual manda todo a un controlador en Unity el cual dará órdenes a las acciones en la escena.

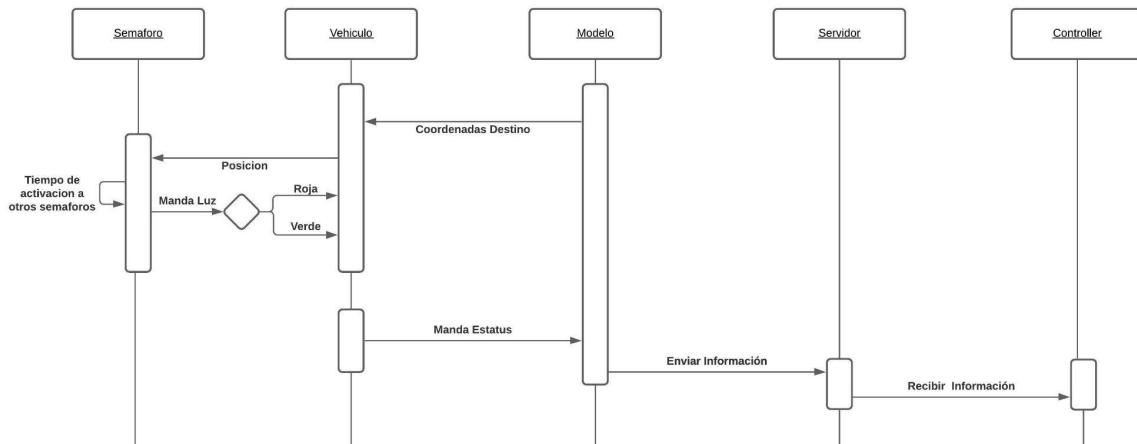


Imagen 8. Protocolos de comunicación de los agentes.

MODELACIÓN 3D

La descripción de la escena consta de un cruce de 4 calles, cada una con ambos sentidos incorporados, lo que permite que los vehículos puedan circular hacia el frente, así mismo se utilizan semáforos en cada cruce que guiarán la vialidad del mismo, también se utilizan elementos decorativos como banquetas y pasto.



Imagen 9. Primera modelación de la escena en Unity.

Los avances realizados para la escena fueron en decorar las calles y espacios en blanco que se tenían, esto se hizo colocando diferentes edificaciones, así mismo, se colocaron luminarias en diferentes puntos para mejorar la visibilidad pero que siga con el concepto de ser calles, por lo que en cada lámpara se utilizó una luz de tipo 'spot'.

Por otro lado, la longitud de las calles se alargó, esto para poder tener más espacio para los de agentes vehículo; también sobre los prefabs de los coches, se reimportaron con sus respectivos materiales para que tengan colores.



Imagen 10. Avance en la modelación de la escena en Unity.



Imagen 11. Vehículos con color en la escena.

Los assets a utilizar son:

- Vehículos
- Calles

- Semáforos
- Luces y postes para colocar los semáforos
- Cruces peatonales
- Banquetas
- Pasto

De los assets mencionados, los vehículos serán diseñados por cada integrante del equipo, cada uno teniendo la libertad de seleccionar el vehículo y diseño de su preferencia.

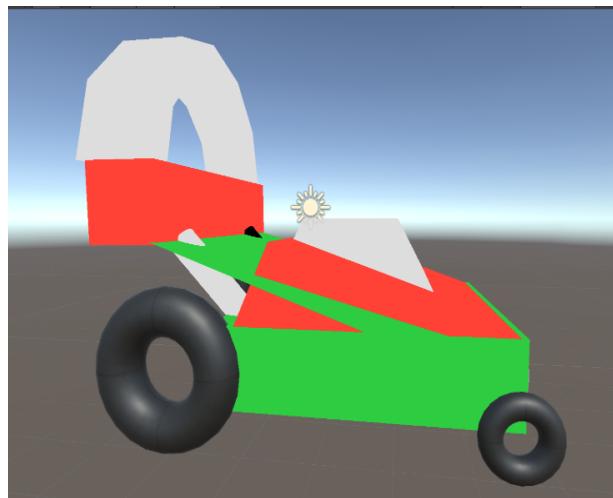


Imagen 12. Asset hecho por Romel Vázquez.



Imagen 13. Asset hecho por Romel Vázquez.

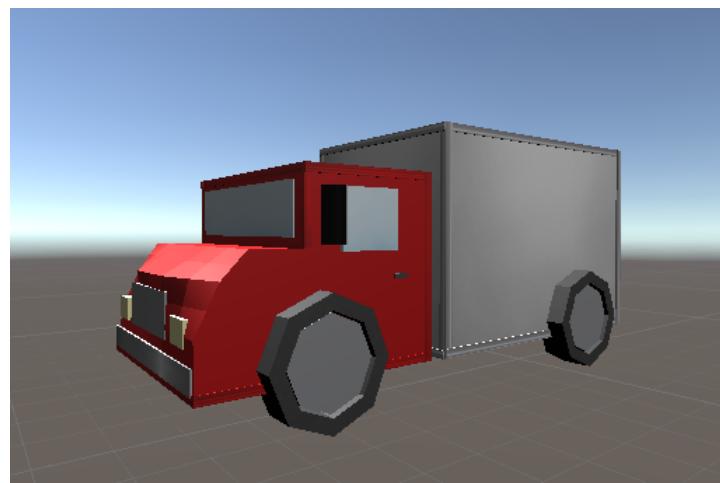


Imagen 14. Asset hecho por Carolina Gómez.



Imagen 15. Assets hecho por David Sánchez.



Imagen 16. Assets hecho por David Sánchez.

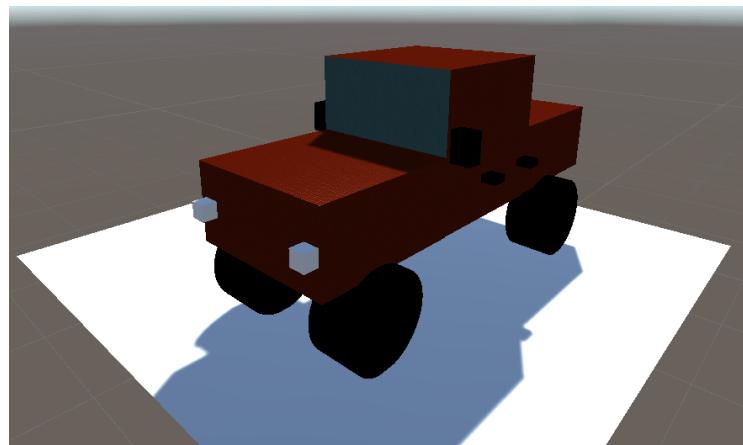


Imagen 17. Asset hecho por Jose Alberto Kaun.

De los demás assets, las calles, pasos peatonales, banquetas y simulación del pasto fueron importados del CITY package hecho por pixel estudios, este puede ser visualizado en el siguiente link:

<https://assetstore.unity.com/packages/3d/environments/urban/city-package-107224>

Mientras que las luces, semáforos y postes para colocar los semáforos fueron obtenidos de FREE Traffic Essentials Asset Pack hecho por Ferocious Industries, se puede visualizar en:

<https://assetstore.unity.com/packages/3d/props/free-traffic-essentials-asset-pack-125092>

METODOLOGÍA AGILE Y COMUNICACIÓN

Este reto será realizado con el apoyo de la metodología Agile, esto con el objetivo de tener una mejor planeación de tiempos, tareas pendientes y delegación de actividades. Así mismo, cada semana se tendrán las actividades del Sprint, las cuales deben de ser completadas en las fechas acordadas.

PRODUCT BACKLOG

Elemento	Historia	Prioridad	Estimado (horas)	Sprint	Estatus
1	Definir solución	Alta	1	1	Terminado
2	Documentación	Alta	8	1	Terminado
3	Buscar assets	Media	1	2	Terminado
4	Definición de requerimientos	Media	2	1	Terminado
5	Definición de agentes	Alta	2	1	Terminado
6	Definición de ambiente	Alta	1	1	Terminado
7	Seguir el planeamiento del backlog	Media	3	1	Terminado
8	Empezar escenario básico en Unity	Media	4	2	Terminado
9	Seguir el planeamiento del backlog	Media	3	2	Terminado
10	Finalizar documento de diseño	Alta	4	2	Terminado
11	Modelo Agentpy	Alta	17	2 3 4	Terminado
12	Crear prefabs	Alta	16	2	Terminado
13	Crear escenario en Unity	Alta	8	3	Terminado
14	Codificación de agentes en Python	Alta	8	3	Terminado
15	Diagramas UML	Media	3	3	Terminado
16	Seguir el planeamiento del backlog	Media	2	3	Terminado
17	Conexión de Unity con Python	Alta	6	4	Terminado
18	Pruebas	Alta	3	5	Terminado
19	Seguir el planeamiento del backlog	Media	2	4	Terminado
20	Presentación final	Media	2	4	En proceso

Imagen 18. Product Backlog hasta el 1 de septiembre del 2021.

Sprint 1						
Elemento	Historia	Tarea	Dueño de la tarea	Tiempo para completar (hr)	Tiempo Planeado (hr)	Estatus
1	Definir solución	Lluvia de ideas	Todos	1	2	Terminado jueves 12/agosto
2	Definir solución	Seleccionar idea	Todos	1	1	Terminado jueves 12/agosto
3	Documentación	Introducción	Romel	2	1	Terminado jueves 12/agosto
4	Documentación	Créditos	Todos	2	1	Terminado miércoles 18/agosto
5	Documentación	Problema	David	1	1	Terminado jueves 12/agosto
6	Documentación	Objetivos generales	José	2	1	Terminado jueves 12/agosto
7	Documentación	Restricciones	Caro	1	1	Terminado jueves 12/agosto
8	Definir requerimientos	Especificar requerimientos funcionales	David	2	2	Terminado jueves 12/agosto
9	Definir requerimientos	Especificar requerimientos no funcionales	Romel	2	2	Terminado jueves 12/agosto
10	Definición de agentes	Definir agente vehículo	Romel	1	1	Terminado domingo 15/agosto
11	Definición de agentes	Definir agente semáforo	Romel	1	1	Terminado domingo 15/agosto
12	Definición de agentes	Gráfico agente vehículo	David	1	2	Terminado domingo 15/agosto
13	Definición de agentes	Gráfico agente semáforo	David	1	2	Terminado domingo 15/agosto
14	Definición de ambiente	Ambiente	Caro	1	2	Terminado jueves 12/agosto
15	Documentación	Modelo de interacción	Romel	1	1	Terminado jueves 12/agosto
16	Documentación	Presentación	Romel y Caro	2	3	Terminado lunes 16/agosto
17	Documentación	Verificar documento	Caro	2	2	Terminado miércoles 18/agosto
18	Seguir el planteamiento del backlog	Incluir las actividades en la documentación	Caro	2	2	Terminado miércoles 18/agosto

Imagen 19. Sprint semana 1.

Sprint 2						
Elemento	Historia	Tarea	Dueño de la tarea	Tiempo para completar (hr)	Tiempo Planeado (hr)	Estatus
1	Buscar assets	Investigar assets de coches, semáforos y decoraciones	Todos	1	1	Terminado miércoles 11/agosto
2	Empezar escenario básico en Unity	Crear repositorio en Github	Caro	1	1	Terminado jueves 19/agosto
3	Empezar escenario básico en Unity	Importar assets en el escenario	Caro	1	1	Terminado lunes 23/agosto
4	Empezar escenario básico en Unity	Colocar calles base	Caro	2	2	Terminado lunes 23/agosto
5	Empezar escenario básico en Unity	Colocar semáforos base	Caro	1	1	Terminado lunes 23/agosto
6	Crear prefabs	Crear prefab coche	Romel	4	4	Terminado lunes 23/agosto
7	Crear prefabs	Crear prefab coche	David	4	4	Terminado lunes 23/agosto
8	Crear prefabs	Crear prefab coche	Kaun	4	4	Terminado lunes 23/agosto
9	Crear prefabs	Crear prefab camión	Caro	4	4	Terminado lunes 23/agosto
10	Finalizar documento de diseño	Diagrama de actividades	David	1	1	Terminado lunes 23/agosto
11	Finalizar documento de diseño	Diagrama de clases	Kaun	1	1	Terminado lunes 23/agosto
12	Finalizar documento de diseño	Documentar assets utilizados en el documento	Caro	1	1	Terminado lunes 23/agosto
13	Modelo Agentpy	Programación de la clase agente	Romel	3	3	Terminado lunes 23/agosto
14	Modelo Agentpy	Diseño de la clase modelo	Romel	1	1	Terminado lunes 23/agosto
15	Seguir el planteamiento del backlog	Presentación checkpoint	Todos	2	2	Terminado lunes 23/agosto
16	Finalizar documento de diseño	Verificar documento	Caro	2	2	Terminado miércoles 25/agosto
17	Seguir el planteamiento del backlog	Incluir las actividades en la documentación	Caro	2	2	Terminado miércoles 25/agosto
18	Finalizar documento de diseño	Crear logo	Kaun	1	1	Terminado miércoles 25/agosto
19	Empezar escenario básico en Unity	Colocar todos los autos en el escenario	David	1	1	Terminado miércoles 25/agosto

Imagen 20. Sprint semana 2.

Sprint 3							
Elemento	Historia	Tarea	Dueño de la tarea	Tiempo para completar (hr)	Tiempo Planeado (hr)	Estatus	Fecha estimada de finalización
1	Crear escenario en Unity	Incluir elementos decorativos	Caro	4	4	Terminado	lunes 30/agosto
2	Crear escenario en Unity	Importar vehículos con color	Caro	1	1	Terminado	miércoles 1/septiembre
3	Crear escenario en Unity	Incluir iluminación	Caro	1	1	Terminado	lunes 30/agosto
4	Diagramas UML	Diagrama de actividades	David	1	1	Terminado	lunes 30/agosto
5	Diagramas UML	Diagrama de clases	David	1	1	Terminado	lunes 30/agosto
6	Diagramas UML	Protocolos de comunicación	David	1	1	Terminado	lunes 30/agosto
7	Diagramas UML	State charts	David	1	1	Terminado	lunes 30/agosto
8	Codificación de agentes en Python	Codificación agente vehículo	Romel	4	6	Terminado	miércoles 1/septiembre
9	Codificación de agentes en Python	Codificación agente semáforo	Romel	4	6	Terminado	miércoles 1/septiembre
10	Seguir el planteamiento del backlog	Incluir las actividades en la documentación	Caro	2	2	Terminado	miércoles 1/septiembre

Imagen 21. Sprint semana 3.

Sprint 4							
Elemento	Historia	Tarea	Dueño de la tarea	Tiempo para completar (hr)	Tiempo Planeado (hr)	Estatus	Fecha estimada de finalización
1	Conexión de Unity con Python	Preparar Unity para la conexión	David	3	3	Terminado	martes 7/septiembre
2	Conexión de Unity con Python	Preparar Python para la conexión	David	3	3	Terminado	martes 7/septiembre
3	Modelo Agentpy	Terminar agentes	Romel	5	5	Terminado	martes 7/septiembre
4	Pruebas	Realizar pruebas y verificar que funcione correctamente	Caro	3	3	Terminado	martes 7/septiembre
5	Seguir el planteamiento del backlog	Incluir las actividades en la documentación	Caro	2	2	Terminado	martes 7/septiembre
6	Conexión de Unity con Python	Posiciones de Agentpy en los vehículos de Unity	David	4	4	Terminado	martes 7/septiembre
7	Conexión de Unity con Python	Semáforos funcionales en Unity	David	3	3	Terminado	martes 7/septiembre
8	Conexión de Unity con Python	Corregir prefabs vehículos	David	1	1	Terminado	martes 7/septiembre
9	Documentación	Corregir diagramas del documento	Romel	1	1	Terminado	martes 7/septiembre
10	Presentación final	Grabar video descriptivo	Caro	2	2	Terminado	martes 7/septiembre
11	Presentación final	Realizar presentación final para Socio Formador	Romel, Caro	2	2	En proceso	viernes 10/septiembre

Imagen 22. Sprint semana 4.

COMUNICACIÓN

Mensajes directos del equipo

- WhatsApp

<https://chat.whatsapp.com/DtO7iT2kyYFDbEzYLifPgZ>

Archivos compartidos

- Carpeta compartida en Drive

<https://drive.google.com/drive/folders/1szBeR6ouOeHer3KbF9ia74Ixt25ajTw2?usp=sharing>

Trabajo en Unity y Python

- Github

<https://github.com/car099gm/Tectraffic>

Reuniones del equipo

- Discord

<https://discord.gg/28MEGC8P>

DRIVE

Liga al Drive:

<https://drive.google.com/drive/folders/1qwMCIAkqE5V4KiUlmgxQVYXqAPgaXZ?usp=sharing>



Imagen 18. Logo Tectraffic.

REFERENCIAS

- [1] Munguía Torres, I., 2016. *SISTEMA DE OPTIMIZACIÓN DE TRÁFICO VEHICULAR APLICADO A LA GLORIETA SANTA FE*. [online] CIMAT. Available at: <<https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/531/1/TE%20613.pdf>> [Accessed 15 August 2021].
- [2] Quintanar, J., 2020. 'Pega' a CdMx contaminación y tráfico vehicular. [online] Milenio. Available at: <<https://www.milenio.com/estados/ciudad-mexico-nube-contaminacion-trafico-vehicular?image=2>> [Accessed 18 August 2021].
- [3] Montiel, G., 2021. Ciudad de México tiene el peor tráfico del mundo • Actualidad • Forbes México. [online] Forbes México. Available at: <<https://www.forbes.com.mx/ciudad-de-mexico-tiene-el-peor-trafico-del-mundo/>> [Accessed 18 August 2021].
- [4] Castillo, A., 2021. Está Monterrey entre las 50 ciudades con más tráfico del mundo. [online] Milenio. Available at: <<https://www.milenio.com/politica/comunidad/esta-monterrey-entre-las-50-ciudades-con-mas-trafico-d-el-mundo>> [Accessed 18 August 2021].
- [5] Silva, E., 2021. Tráfico en Monterrey, podemos hacer poco y colaborar mucho. [online] El Financiero. Available at: <<https://www.elfinanciero.com.mx/monterrey/trafico-en-monterrey-podemos-hacer-poco-y-colaborar-mucho/>> [Accessed 18 August 2021].
- [6] Tecnocarreteras. 2015. ¿Cómo y por qué usar simuladores de tráfico para optimizar la gestión del mismo? - Tecnocarreteras. [online] Available at: <<https://www.tecnocarreteras.es/2015/03/19/como-y-por-que-usar-simuladores-de-trafico-para-optimizar-la-gestion-del-mismo/>> [Accessed 18 August 2021].