Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

TC2008B.402: Modelación de Sistemas Multiagentes con Gráficas Computacionales

Evidencia 2 - Revisión Final

Profesores:

Dr. Luciano García Bañuelos

Dr. Iván Olmos Pineda

Estudiantes:

Daniel Francisco Acosta Vázquez / A01736279

Diego García de los Salmones Ajuria / A01736106

Oskar Adolfo Villa López / A01275287

Raúl Díaz Romero / A01735839

Periodo Agosto - Diciembre 2023

15 / Octubre / 2023

Tabla de Contenidos:

Revisión Final

I. **Github:** https://github.com/RaulDiazR/MultiAgent_Intersection
II. **Video de presentación:** https://drive.google.com/file/d/1nwf74ud9-XQHwYSMEtn-DDP4Z2n9kzA6/view?usp=sharing
III. **Explicación de los archivos y directorios encontrados en el directorio principal del proyecto**
    A. **Objetos3D:** directorio que contiene múltiples archivos de objetos modelados para utilizar en la simulación.
    B. **Textures:** directorio que contiene los archivos bmp necesarios para las texturas que se utilizan.
    C. **backend:** directorio que contiene todos los archivos necesarios del apartado lógico (MESA) del proyecto.
    D. **revisiones_evidencia2:** directorio que contiene las múltiples revisiones del proyecto, cada revisión incluye una explicación general del proyecto en fechas distintas desde el inicio del proyecto.
    E. **CarO.py:** archivo utilizado para el apartado gráfico (OpenGl) del proyecto, este archivo se encarga de definir una clase para los carros utilizados en la simulación.
    F. **objloader.py:** archivo encargado de cargar los modelos de los objetos utilizados en la simulación, algunos de estos modelos incluyen: carros, semáforos, edificios, etc.
    G. **simulation.py:** archivo principal del apartado gráfico (OpenGl) del proyecto, este archivo se encarga de recibir los datos lógicos del directorio backend y de modelar los objetos correctamente en la simulación.
    H. **README.md:** breve guía informativa sobre el proyecto.
    I. **.gitignore:** archivo utilizado para indicar al repositorio que archivos ignorar, esto se utiliza para no incluir archivos temporales en el repositorio.

IV. **Avances de Código**
    A. **Implementación de los agentes:** Se ha terminado por completo la implementación de los agentes, pues el modelo actúa correctamente, generando carros inteligentes, los cuales actúan conforme a su entorno. Así mismo, los semáforos y las calles han sido implementados correctamente, por lo que cada agente ha sido implementado

efectivamente en el modelo. Estos avances se pueden encontrar en la rama ‘turn’ del repositorio del proyecto.

B. **Implementación de la parte gráfica de la solución:** Se ha finalizado la implementación gráfica en OpenGl, generando el ambiente gráfico correctamente al diseñar por completo los componentes de la ciudad, incluyendo: calles, semáforos, edificios, estanques, árboles y carros. Adicionalmente se utiliza un skybox texturizando una esfera para simular el cielo, al igual que la opción de que el observador en la simulación siga a un coche en específico.

V. Aspectos del proyecto.

A. Descripción del reto final

El proyecto consiste en la creación de un sistema multiagente que emule el comportamiento de múltiples carros al atravesar una intersección con semáforos. Este proyecto implica el desarrollo de algoritmos avanzados para coordinar la interacción entre los carros, ajustando sus velocidades con el propósito de evitar colisiones. Además, se requiere una gestión eficiente y efectiva de varios agentes, ya que se contempla la incorporación de varios carros que buscarán cruzar la intersección simultáneamente.

El proyecto engloba la creación del entorno donde interactúan los carros y semáforos, el apartado lógico. Para llevar a cabo tanto el desarrollo del sistema como la implementación de los carros y semáforos, se planea utilizar la librería 'Mesa' a través del lenguaje de programación ‘Python’, dicha librería facilita la creación, manipulación e interacción de los agentes del modelo.

Además, el apartado gráfico del proyecto se llevará a cabo utilizando la herramienta OpenGL, en la cual se modelan los objetos utilizados en el proyecto, tales como: la ciudad, los carros, los semáforos y las calles.

Por último, se planea conectar el apartado lógico (Mesa) con el apartado gráfico (OpenGL) a través del uso de una arquitectura cliente-servidor, dada por el Rest Api. A través de dicha arquitectura, se enviarán datos desde Mesa hacía OpenGl en formato Json, estos datos definirán los estados y acciones de los agentes en el modelo en cada momento de la simulación.

B. Cambios realizados desde la propuesta inicial del reto

En nuestra propuesta inicial, concebimos la creación de un sistema multiagente destinado a simular el comportamiento de varios vehículos en una intersección. Nos

enfocamos principalmente en la dinámica de interacción entre estos vehículos, ponderando la necesidad de ajustar sus velocidades para prevenir colisiones y la eficiente gestión de múltiples automóviles que cruzarían la intersección simultáneamente. La idea central era que el cruce en sí funcionara como un agente inteligente, coordinando y supervisando la seguridad de los vehículos que lo atravesaran.

A medida que hemos avanzado en el desarrollo del proyecto, hemos implementado cambios significativos. En particular, decidimos abandonar la noción de cruces inteligentes debido a su complejidad y las limitaciones de tiempo y recursos del equipo. En su lugar, hemos otorgado la inteligencia a los vehículos mismos, dotándolos de la capacidad de evaluar continuamente su entorno y tomar decisiones acertadas en función de las condiciones existentes. Además, hemos introducido la incorporación de semáforos para mejorar la gestión de los vehículos. Estos semáforos permiten que los automóviles se adapten de manera más efectiva a las señales de tráfico, mejorando así la seguridad y la eficiencia en la intersección. Este enfoque nos ha permitido simplificar el proyecto y mejorar su capacidad de implementación, al tiempo que ha enriquecido la dinámica de simulación.

Por último, se agregaron nuevos agentes al proyecto, pues inicialmente se plantaban solamente 1 agente (carro) y 1 modelo (cruce), después de semanas de trabajo y análisis, se decidió utilizar los siguientes agentes y modelo:

- Car (Agente)
- TrafficLight (Agente)
- Street (Agente)
- City (Modelo):

Como se puede observar, se añadieron 2 agentes y se intercambio el modelo por una nueva idea (City) y el viejo modelo (Cruce) fue descartado. Además, se añadieron los agentes TrafficLight y Street, TrafficLight se encarga de generar los semáforos en el modelo y su comportamiento y Street es un elemento puramente visual para poder observar el comportamiento correcto de los agentes en el apartado lógico (MESA).

Fuera de los cambios ya mencionados, no se realizaron cambios significativos en cuanto a la idea del proyecto, la estructura y diseño de la ciudad permaneció prácticamente igual a través del desarrollo del proyecto, los comportamientos de los

agentes, fuera de los ya mencionados, permanecieron igual y la lógica detrás del proyecto permaneció similar.

En cuanto a la propuesta original del proyecto en el apartado gráfico, habíamos propuesto el modelar la ciudad con múltiples edificios al igual que utilizar fuentes de luz para simular un atardecer y simular la luz dada por los coches al igual que los semáforos y postes de luz.

Sin embargo, por cuestiones de tiempo y recursos, tuvimos que cambiar la visión del proyecto a una ciudad un poco más simple, integrando parques y simulando que una calle es un puente con texturas de agua, al igual que agregar árboles y un skybox. También algo que no previamente habíamos considerado pero terminamos integrando, es la funcionalidad de mover la cámara sobre uno de los coches.

VI. Descripción del apartado gráfico

En cuanto al apartado gráfico del programa, construimos la ciudad en base al modelo utilizado en MESA. Utilizamos planos para realizar la base, banquetas, y las calles de la ciudad. Para el pasto de los parques y el agua, utilizamos texturas que cargamos previamente y dibujamos planos igualmente con ellas. Cargamos y utilizamos modelos 3D, para los edificios, coches, árboles y semáforos. Estos solo los rotamos y escalamos para que tengan la orientación correcta o tamaño distinto en caso de los edificios. De igual manera, utilizamos una textura blanca para evitar problemas con las texturas y los modelos 3D. Finalmente con una textura adicional, texturizamos una esfera que dibujamos alrededor de la escena para simular nuestro skybox.

Para el dibujo de los coches, utilizamos los datos mandados por JSON de parte de mesa para inicializar y obtener las coordenadas, orientación y velocidad de los coches. Utilizando la orientación y coordenadas así aseguramos que se dibujan correctamente.

Finalmente, para obtener la cámara encima de uno de los coches, utilizamos una variable de control que al estar actividades, hace que cada update del coche la cámara se actualice automáticamente mirando hacia al frente del coche y se posicione justo encima de este. Una vez se desactive, moviendo la cámara, esta regresara a su posición original.

VII. Descripción de Modelo y Agentes involucrados

A. Agentes y Modelo a utilizar:

1. Car (Agente): Este agente representa la lógica de un vehículo con comportamiento realista en la simulación. Este agente posee múltiples atributos como su posición, orientación y referencias al entorno, incluidos los semáforos. En cada paso de la simulación, el agente decide si el automóvil debe acelerar, desacelerar o frenar según las condiciones del tráfico, la proximidad de semáforos y la proximidad de otros carros. Los métodos adicionales del agente, como carAhead y checkTurn, permiten al automóvil detectar otros vehículos y giros necesarios. Además, se gestionan las interacciones con los semáforos a través del método traffic_light_ahead y se encuentra el semáforo más cercano con findNearestTrafficLight. En conjunto, el agente Car simula el comportamiento realista de un vehículo en un escenario de tráfico dentro del modelo de simulación.
2. TrafficLight (Agente): Este agente representa un semáforo con comportamiento realista en la simulación. Este agente tiene propiedades como la orientación del semáforo (sur, este, norte u oeste), la duración predeterminada de las luces verde, amarilla y roja, así como los tiempos de transición entre ellas. El agente tiene tres estados que representan las luces del semáforo (verde, amarillo y rojo) y un temporizador que cuenta hacia abajo en cada estado. En cada paso de la simulación, el semáforo cambia de estado y actualiza el temporizador correspondiente cuando se alcanza cero. Esto simula la operación cíclica de un semáforo real. La duración de las luces se ajusta según la orientación del semáforo para coordinar el flujo de tráfico. En resumen, la clase TrafficLight modela el funcionamiento de un semáforo de tráfico en un entorno de simulación, gestionando las transiciones entre las luces de manera realista y coordinada según su orientación.
3. Street (Agente): Este agente es un añadido sencillo, pues su única función es la de mostrar visualmente las calles en el apartado lógico. Dicho agente se utilizó únicamente en el desarrollo individual del apartado lógico, cuando todavía no se realizaba la conexión con el apartado gráfico.
4. City (Modelo): El modelo de la ciudad representa el entorno general donde interactúan los carros, los semáforos y las calles. Los carros se mueven dentro de esta ciudad, respetando las reglas de tráfico y respondiendo a los semáforos.

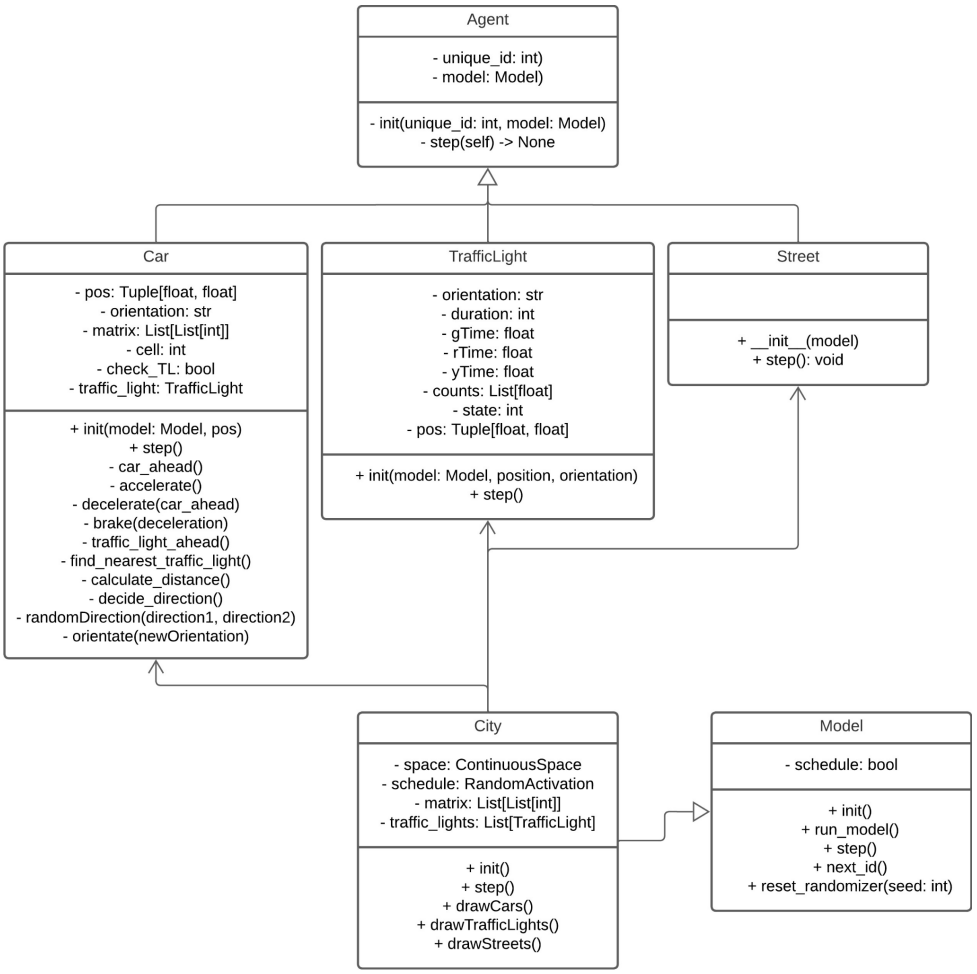
Además, el modelo posee una matriz, la cual representa las diferentes acciones y limitaciones que tienen los carros a través de la ciudad, tales como: sentido de dirección, opciones de giro, posición de las calles y edificios, etc. Por último, el modelo se encarga de generar a cada uno de los agentes con sus parámetros correspondientes.

B. Relación entre agentes:

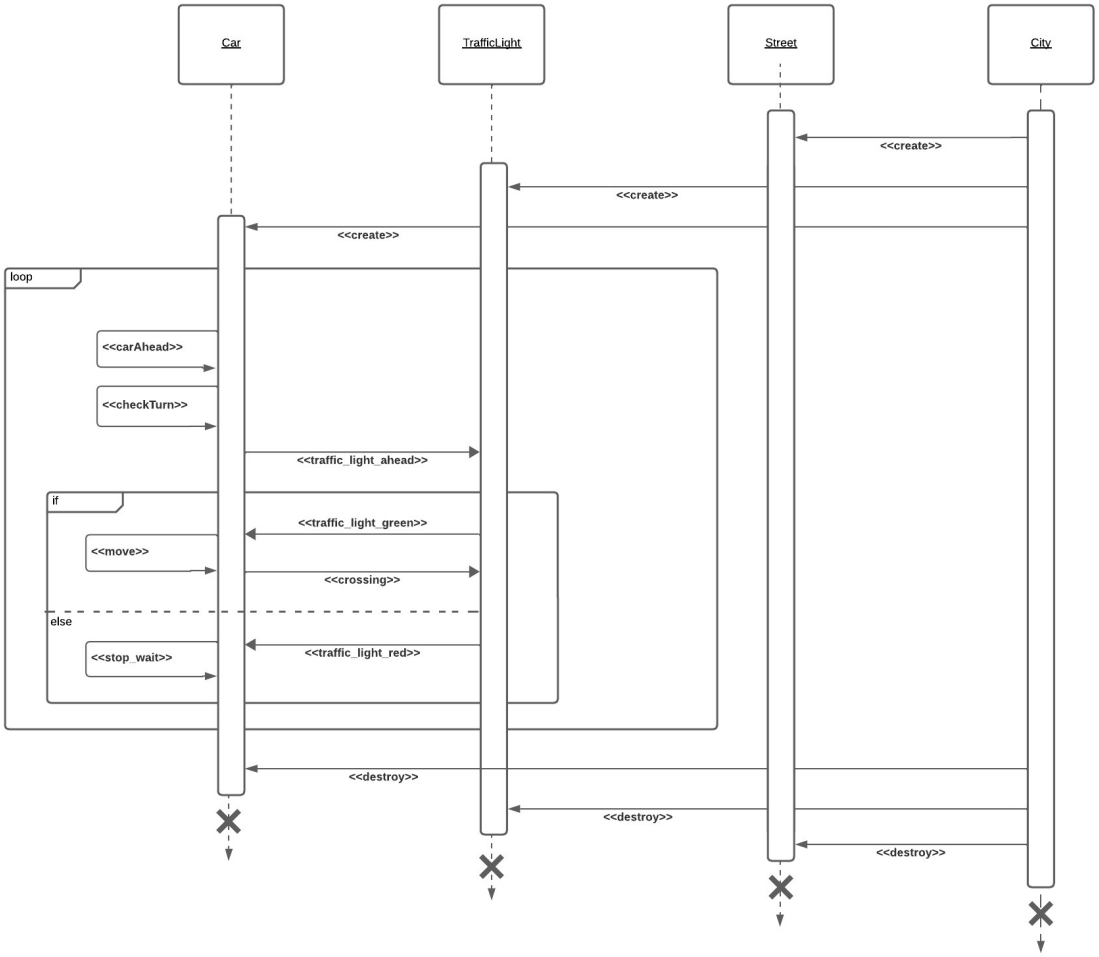
En este proyecto, los carros y semáforos son agentes interdependientes que colaboran para simular el flujo de tráfico en una intersección. Los carros representan vehículos en movimiento y están sujetos a las señales de tráfico controladas por los semáforos y los demás carros situados en la simulación. Los semáforos, por su parte, desempeñan un papel crucial al regular el tráfico en la intersección, alternando entre los estados de luz verde, amarilla y roja. Los carros deben interpretar estos estados y actuar en consecuencia, deteniéndose ante la luz roja y avanzando con precaución cuando la luz está en verde. El modelo de ciudad, como entidad central, coordina y supervisa estas interacciones entre carros y semáforos, asegurando que el tráfico fluya sin problemas y se eviten colisiones. En resumen, esta relación colaborativa entre agentes y el modelo de ciudad es esencial para crear una simulación realista y segura de carros cruzando una intersección con semáforos.

C. Diagrama de clase presentando los distintos agentes involucrados.

Link de LucidChart:
https://lucid.app/lucidchart/c94515f0-5335-46ee-aa3e-4d3b04cd8552/edit?view_items=rMypwTKdzM2C&invitationId=inv_0e690c61-fa72-413c-9408-77d80e37e626



D. Diagrama de Secuencia de protocolos de interacción.



VIII. Reglas de tránsito para el modelo

Para definir las reglas de tránsito utilizadas en el modelo, se deben definir los tipos de intersecciones a utilizar. A continuación, se definen las intersecciones a utilizar dentro de la simulación.

Las intersecciones que se planean utilizar son dos intersecciones de 4 vías tipo cruz y dos intersecciones de 3 vías tipo T. Cabe mencionar que las intersecciones poseen semáforos, los cuales se encargan de gestionar el tráfico.

Las reglas de tránsito utilizadas en el modelo incluye las reglas básicas de ambos tipos de intersección.

Primero, para la intersección de 4 vías, se utilizan las siguientes reglas:

- Los vehículos sólo pueden avanzar cuando su semáforo respectivo muestra luz verde.
- Una vez que la luz verde esté encendida, los vehículos tienen la opción de seguir recto en la dirección de la calle de la que provienen o girar hacia la derecha.

Segundo, para la intersección de 3 vías, se utilizan las siguientes reglas:

- Los vehículos sólo pueden avanzar cuando su semáforo respectivo muestra luz verde.
- Para los vehículos que se desplazan desde la calle principal y se ubican en el carril adyacente a la calle que se une a la intersección, tienen la opción de continuar recto o girar hacia la derecha.
- Los vehículos que provienen de la calle principal y se sitúan en el carril opuesto al que se une a la intersección, sólo pueden seguir recto.
- Los vehículos que llegan desde la calle que se incorpora a la calle principal y desean ingresar a la intersección, tienen la opción de girar hacia la derecha o avanzar ligeramente recto hasta llegar al carril siguiente, donde pueden realizar un giro hacia la izquierda.

IX. Guía de instalación, configuración y ejecución de la simulación.

A. Tecnologías necesarias a instalar:

1. Python: esta herramienta se puede instalar desde la página oficial de Python (https://www.python.org/downloads/). En dicho link debes escoger la versión correspondiente a su sistema operativo, en el desarrollo de este proyecto se utilizó la versión 3.10.5 en Windows.
2. Visual Studio Code, IDE para ejecutar el código, pero basta con utilizar la terminal incluída en el sistema. Para instalar Visual Studio Code se puede instalar desde su página oficial (https://code.visualstudio.com/). En caso de utilizar la terminal integrada en su sistema operativo, no es necesario instalar nada.
3. Anaconda, utilizado para crear un ambiente virtual en python (venv). Esta herramienta se puede descargar desde su página oficial (https://www.anaconda.com/download). Una vez esté instalado Anaconda, se debe de crear un ambiente virtual en donde instalará las librerías de python y se ejecutará el código, para ello es necesario tener Visual Studio Code instalado, pues es a través de esta herramienta que se trabajará en el ambiente virtual.

B. Librerías de Python a instalar:

1. numpy
2. mesa

3. flask
4. json
5. uuid
6. pygame
7. PyOpenGL
8. math
9. requests

Primero, se debe abrir Anaconda y entrar al ambiente virtual creado previamente, dentro de este, se debe abrir Visual Studio Code a través de Anaconda.

Una vez dentro del ambiente virtual y Visual Studio Code, para instalar las librerías previamente mencionadas, se debe utilizar el comando 'pip install *librería_a_instalar*', por ejemplo, 'pip install numpy'. Dicho comando debería funcionar si la instalación de Python se realizó correctamente, pues el comando pip está instalado junto a Python. Además, existe la opción de utilizar el comando de instalación con múltiples librerías a la vez, por ejemplo 'pip install numpy mesa flask'.

C. Ejecución:

Para ejecutar el proyecto es necesario seguir los siguientes pasos:

1. Abrir Anaconda y acceder al ambiente virtual creado para el proyecto, este ambiente virtual contiene las librerías de python necesarias para ejecutar correctamente el programa.
2. Ejecutar el archivo backend.py en una terminal única, pues este proceso debe ejecutarse en segundo plano para funcionar correctamente. El comando para ejecutar un código en python es 'python backend.py', cabe mencionar que el archivo backend.py está dentro de la carpeta con el nombre 'backend', todos los archivos de esta carpeta se encargan del apartado lógico del programa.
3. Una vez ejecutado el archivo backend.py, deberá abrir una nueva terminal en la cual ejecutará el archivo simulation.py, cabe mencionar que el archivo simulation.py está en el directorio principal del proyecto, al igual que todos los componentes del apartado gráfico del programa.
4. ¡Felicidades! Al seguir los pasos previamente establecidos, el programa ahora debería ejecutarse sin problema alguno, cabe resaltar que el programa puede tener

una velocidad de ejecución lenta en caso de que la computadora utilizada tenga pocos recursos.

D. Video de explicación:

https://drive.google.com/file/d/1IAbD4imvJK4ttZBxuz0S1YEEIBpeBrF5/view?usp=sharing

X. Plan de trabajo

A. Actividades pendientes:

- Presentación final el día lunes 16 de octubre.

B. Actividades planeadas para la revisión final (9 de octubre -13 de octubre):

- Todas las actividades han sido completadas.

C. Actividades completadas:

- Implementar las clases para cada agente en MESA (Car, Street y TrafficLight). Responsable: Raúl. Intervalo de esfuerzo estimado: medio.
- Completar el modelo y el comportamiento correcto de los agentes en Mesa. Responsables: Raúl y Oskar. Intervalo de esfuerzo estimado: medio.
- Obtener los modelos de los objetos utilizados en el sistema para OpenGl. Responsables: Diego y Daniel. Intervalo de esfuerzo estimado: bajo.
- Conectar el modelo con el ambiente. Responsables: Raúl y Oskar. Intervalo de esfuerzo estimado: medio.
- Actualización del diagrama de secuencia. Responsable: Oskar. Intervalo de esfuerzo estimado: medio.
- Actualización del Diagrama de clases. Responsable: Raúl. Intervalo de esfuerzo estimado: medio.
- Construir el ambiente gráfico en OpenGL. Responsables: Diego y Daniel. Intervalo de esfuerzo estimado: medio.
- Actualizar Secciones de texto del documento. Responsable: Todo el equipo. Intervalo de esfuerzo estimado: Bajo.

XI. Reflexiones Personales

A. Raúl Díaz Romero: Este proyecto muestra el gran impacto que la tecnología tiene en la sociedad. La capacidad de recrear de manera realista situaciones de tráfico en

un entorno controlado es crucial para abordar la seguridad vial y la eficiencia en el transporte.

Lo más destacado de este proyecto radica en la combinación de herramientas y lenguajes de programación como Python, Mesa, OpenGL y REST API, que se unen de manera sinérgica para ofrecer una solución completa. Esto ejemplifica cómo la tecnología puede ser una aliada poderosa para resolver problemas complejos en la sociedad actual.

La flexibilidad que se presenta en este proyecto, con la capacidad de realizar cambios significativos a medida que evoluciona, demuestra cómo la tecnología puede adaptarse a las necesidades cambiantes de manera efectiva.

La colaboración entre diferentes agentes, como vehículos y semáforos, refleja cómo la tecnología puede ayudar a coordinar y optimizar la interacción de múltiples elementos en la sociedad. Esta simulación de una ciudad también nos recuerda cómo las tecnologías pueden influir en la planificación urbana y la movilidad.

En resumen, este proyecto ejemplifica el potencial de la tecnología para mejorar la seguridad vial, la eficiencia en el transporte y la adaptabilidad en la resolución de desafíos en una sociedad en constante cambio. Aprender cómo estas tecnologías pueden interactuar para generar simulaciones realistas y complejas fue sumamente útil.

B. **Diego García de los Salmones Ajuria**: Con este proyecto culminamos este curso de multiagentes y gráficas computacionales, en donde hemos aplicado todo lo aprendido sobre Python, Mesa, OpenGL y REST API, haciendo posible la simulación del tráfico en una ciudad y utilizando de manera conjunta las herramientas tecnológicas anteriormente mencionadas. Este proyecto es una muestra de la gran utilidad que pueden tener las simulaciones tecnológicas para abordar situaciones del mundo real en la toma de decisiones y análisis de escenarios complejos , y un ejemplo muy bueno de esto es el tráfico en una ciudad, el cual puede servir para actividades como la planificación urbana, optimización de las redes viales, análisis de movilidad de los vehículos e identificación de riesgos. Para concluir, creo que este tipo de proyectos pueden tener un gran potencial no únicamente en el ámbito académico, sino también en el mundo real y pueden llegar a mejorar la calidad de vida de las personas y la eficiencia en las ciudades.

C. **Daniel Francisco Acosta Vazquez:** Este proyecto tiene una gran relevancia en nuestro desarrollo como programadores. No solo aprendemos y ponemos en práctica

habilidades en dos áreas muy importantes, las cuales son graficación y modelación multi agentes incluso dándoles inteligencia a dichos agentes, pero también el practicar y aprender a implementar diferentes frameworks y herramientas que nos permiten realizar simulaciones o programas más complejos. Sin duda un gran reto y algo de lo que personalmente siento que quedo muy bien en el proyecto fue el apartado gráfico, pues utilizamos herramientas y conceptos que no habíamos utilizado anteriormente como iluminación, importación de modelos y un manejo más detallado de texturas. Esto claro sin mencionar que todo tuvo que ser modelado en base a los datos que mandaba la api de REST para asegurarnos de que fuera una buena representación gráfica de lo modelado en MESA. En conclusión, considero que este proyecto realmente nos impulsó a aprender y a mejorar nuestras habilidades y trabajo en equipo para realizar un producto complejo utilizando herramientas distintas.

D. **Oskar Adolfo Villa López:** Con la realización de este proyecto conocí la capacidad de la aplicación de los sistemas multiagentes en simulaciones de situaciones en las que existen interacciones entre distintos objetos con características específicas. Además, pude tener contacto con la comunicación entre distintos programas mediante el protocolo HTTP con objetos JSON. Una de las principales dificultades que enfrentamos fue la diferencia entre la representación de objetos entre MESA y OpenGL, ya que MESA dibuja los objetos desde una esquina y OpenGL los dibuja utilizando el centro como punto de referencia, por lo que las representaciones se veían diferente y se complicaba el desarrollo de la lógica.

XII. Aprendizaje adquirido.

A. **Programación y Algoritmos Avanzados**: En este proyecto hemos necesitado implementar algoritmos para coordinar el movimiento de múltiples agentes de manera eficiente, lo que incluye la optimización de rutas, la detección de colisiones, la sincronización de semáforos virtuales, etc, al igual que utilizar conceptos de programación para coordinar la lógica entre agentes y entre el apartado gráfico de la simulación.

B. **Simulación y Modelado**: En este proyecto hemos modelado el comportamiento de los carros, su interacción con la intersección y otros carros, al igual que un modelado de una vialidad simulando una ciudad.

C. **Coordinación y Colaboración**: Hemos aprendido sobre la importancia de la coordinación y colaboración entre agentes para evitar colisiones y maximizar la eficiencia en la intersección.

D. **Diseño de Sistemas Complejos**: En este proyecto hemos tenido que considerar cómo los diferentes componentes interactúan entre sí, cómo gestionar la comunicación y la toma de decisiones entre los agentes, y cómo mantener la estabilidad del sistema.

E. **Programación Orientada a Objetos**: Al modelar carros y la intersección como agentes, hemos trabajado con conceptos de programación orientada a objetos, como el diseño de clases, propiedades y métodos, y la lógica de los agentes en objetos.

F. **Optimización y Eficiencia**: Al trabajar con varios agentes al mismo tiempo hemos requerido trabajar en la optimización y la eficiencia de nuestra simulación con el fin de obtener una simulación correcta. De igual manera en el apartado gráfico hemos aprendido cuando es apto utilizar modelos de alta o baja cantidad de polígonos al igual que cuando conviene mejor utilizar texturas.

G. **Desarrollo de agentes inteligentes**: Hemos aprendido cómo crear agentes inteligentes e implementar algoritmos que permitan que estos puedan interactuar de manera efectiva en un entorno complejo como una intersección de tráfico.

H. **Modelado de objetos 3D**: La implementación de modelos para representar calles, edificios, semáforos y carros en OpenGL nos ha permitido aprender sobre modelado 3D y cómo importar y representar objetos tridimensionales de manera efectiva en una simulación.

I. **Importado de objetos 3D**: Al utilizar modelos pre creados y agregarlos a nuestra simulación, hemos aprendido a importar y trabajar con modelos y material externo y agregarlo al proyecto.

J. **Conexión entre lógica y gráficos**: Hemos aprendido acerca de la importancia de la comunicación entre la lógica y la parte gráfica de tu proyecto. Esto es fundamental para lograr una representación visual precisa de la simulación que refleje el estado actual de los agentes y el entorno.

K. **Escalamiento y Adaptabilidad**: Hemos aprendido a escalar y adaptar los datos obtenidos por la parte lógica del sistema y trasladarlos a la parte gráfica del mismo manteniendo su integridad y asegurando una correcta representación de la simulación.

L. **Arreglo de Errores**: Hemos aprendido y reforzado nuestra habilidad de encontrar y arreglar errores cuando hay una conexión entre diferentes sistemas, al tener que reconocer si un error es lógico o gráfico y saber resolverlo de la manera adecuada.

M. **Trabajo Organizado**: Hemos aprendido a realizar el proyecto siguiendo el plan de trabajo al mismo tiempo que a organizar las tareas y juntar tanto el apartado gráfico como el lógico para poder ver una armonía entre ambos sistemas y tener una simulación eficaz.

XIII. Conformación del equipo

A. Integrantes del equipo:

1. **Daniel Francisco Acosta Vázquez**
   a) Fortalezas: Resiliencia, Adaptabilidad, Comunicación e Ingenio
   b) Áreas de oportunidad: Responsabilidad y Organización
   c) Expectativas: Espero poder entender e implementar un sistema multiagentes para simulaciones complejas al igual que reforzar los conocimientos de graficación mediante OpenGL al igual que su uso en diferentes áreas.
2. **Diego García de los Salmones Ajuria**
   a) Fortalezas: Responsabilidad, Cumplimiento y Ganas de Aprender y Mejorar.
   b) Áreas de oportunidad: Organización y Participación Activa.
   c) Expectativas: Resolver el problema que plantea este proyecto por medio de las herramientas, técnicas y habilidades tecnológicas y computacionales que hemos adquirido y desarrollado a lo largo del curso.
3. **Oskar Adolfo Villa López**
   a) Fortalezas: Responsabilidad, experiencia en áreas de robótica.
   b) Áreas de oportunidad: Planeación inicial.
   c) Expectativas: Espero que nuestro modelo pueda realizarse de forma sencilla y que pueda modelar correctamente una situación real.
4. **Raúl Díaz Romero**
   a) Fortalezas: Comunicación Activa, Responsabilidad, Organización y Adaptabilidad.
   b) Áreas de Oportunidad: Enfocarse demasiado en pequeños detalles e Indecisión
   c) Expectativas: Espero poder dominar la implementación de sistemas multiagentes para proyectos complejos, tales como el proyecto a completar

en este bloque. Además, espero aprender más profundamente la implementación de videojuegos desde su apartado gráfico. Finalmente, anhelo que esta materia sea mucho más práctica que otras, priorizando el aprendizaje a través de proyectos y actividades fructíferas.

B. Logros y Compromisos esperados como Equipo de Trabajo:

1. Logros:

a) Crear un modelo de sistema multiagente que simula de manera precisa y realista la interacción de carros en una intersección.

b) Implementar algoritmos de coordinación que permitan a los carros cruzar la intersección de manera segura y eficiente.

c) Proponer enfoques novedosos y creativos para resolver los desafíos de tráfico en intersecciones.

d) Programar la solución de manera efectiva, utilizando la librería ‘Mesa’ y técnicas de programación avanzadas para lograr una simulación y coordinación de carros precisa y fluida.

2. Compromisos:

a) Establecer reuniones regulares para revisar el progreso, identificar obstáculos y ajustar el plan según sea necesario.

b) Familiarizarse con la librería 'Mesa' y ‘OpenGL’, comprendiendo sus características y funcionalidades para su implementación eficaz.

c) Distribuir tareas según las fortalezas individuales, asegurando una distribución equitativa de la carga de trabajo.

d) Implementar el código de manera modular y bien documentada, siguiendo las mejores prácticas de programación para facilitar la colaboración y el mantenimiento.

e) Realizar pruebas exhaustivas en diferentes escenarios de tráfico para verificar la precisión y robustez del sistema multiagente.

f) Documentar adecuadamente el proceso de diseño, implementación y pruebas, incluyendo explicaciones detalladas de los algoritmos y la lógica utilizada.

g) Mantener una comunicación abierta y constante entre los miembros del equipo, compartiendo actualizaciones y resolviendo problemas de manera colaborativa.