



Modelación de sistemas multiagentes y gráficas computacionales

Reto: Movilidad urbana

Integrantes

Diego Enrique Jiménez Urgell	A01652617
Agustín Abreu Callejas	A01653126
Yusdivia Molina Román	A01653120

Profesores

Dr. Sergio Ruiz Loza
Dr. David Christopher Balderas Silva

Índice

1. Equipo de trabajo	3
1.1. Conformación del equipo	3
1.1.1 Diego Jiménez	3
1.1.1.1. Fortalezas	3
1.1.1.2. Áreas de oportunidad	3
1.1.1.3. Expectativas	3
1.1.2 Agustín Abreu	3
1.1.2.1. Fortalezas	3
1.1.2.2. Áreas de oportunidad	4
1.1.2.3. Expectativas	4
1.1.3 Yusdivia Molina	4
1.1.3.1. Fortalezas	4
1.1.3.2. Áreas de oportunidad	4
1.1.3.3. Expectativas	4
1.2. Expectativas como equipo y compromisos	4
2. Creación de herramientas de trabajo colaborativo	5
2.1. Repositorios de GitHub	5
3. Propuesta formal del reto	5
3.1. Descripción del reto a desarrollar	5
3.2. Identificación de los agentes involucrados	7
3.2.1. Diagrama de clase	8
3.2.2. Diagrama de protocolos de interacción	9
3.3. Plan de trabajo y aprendizaje adquirido	10
3.3.1. Plan de trabajo	10
3.3.2. Aprendizajes adquiridos	11
3.3.2.1 Gráficas computacionales	11
3.3.2.2 Sistemas multiagentes	12
3.3.3. Las actividades pendientes y el tiempo en el que se realizarán.	13
3.3.4. Actividades planeadas	14
4 Resultados e instalación	14
4.1 Proceso de instalación	14
4.2 Análisis de solución desarrollada	26
4.3 Reflexión proceso de aprendizaje	29
Anexos	30
Referencias	31

1. Equipo de trabajo

1.1. Conformación del equipo

Indicar los integrantes del equipo de trabajo. Además deben identificar las fortalezas y áreas de oportunidad de cada uno de ustedes. Así como las expectativas que tienen del bloque. Posteriormente, elaborar un breve listado de lo que esperan lograr y obtener como equipo de trabajo en el presente bloque, así como sus compromisos para lograrlo.

1.1.1 Diego Jiménez

1.1.1.1. Fortalezas

Mi principal fortaleza es la resiliencia, soy capaz de seguir adelante incluso cuando la situación no parece favorable. Esto está ligado con la dedicación, puedo llegar a ser muy obsesivo con las cosas que me interesan, y planeo focalizar parte de esa energía en el bloque. Considero que soy asertivo al comunicarme, y puedo expresarme de manera adecuada y amigable cuando no estoy de acuerdo con algo. Además, me tomo muy en serio mis compromisos cuando impactan a otras personas, como en el caso de un equipo de trabajo.

1.1.1.2. Áreas de oportunidad

Una de mis debilidades actuales es la distribución y organización de mi tiempo. Debido a otras actividades que tengo, en algunas ocasiones no me da suficiente tiempo para realizar mis tareas como me gusta. En muchos de estos casos sacrifico mis horas de sueño, lo que se va acumulando y me genera un agotamiento crónico para el final del bloque. Uno de mis objetivos en este bloque es disminuir esto. Otra área de oportunidad es que me distraigo cuando un tema no me gusta, y eso me atrasa después. Además puedo llegar a ser impaciente.

1.1.1.3. Expectativas

En este bloque sobre todo espero aprender mucho de sistemas multiagentes. Esto no solo es útil para videojuegos, sino para muchas aplicaciones del mundo real que requieren inteligencia artificial. Este es un campo en el que me he interesado desde antes, y me parece una buena oportunidad para explorarlo. En cuanto a gráficas computacionales, me parece interesante conocer más de ellas puesto que son un área crucial de las ciencias computacionales, aunque no me veo en este campo en un futuro.

1.1.2 Agustín Abreu

1.1.2.1. Fortalezas

Personalmente me considero una persona creativa y crítica que siempre busca realizar las actividades de la mejor forma posible. Por otra parte, también soy una persona bastante apasionada por los temas de su interés, y es esta idea la que me motiva cada día para seguir

haciendo un mejor trabajo. Finalmente, también soy alguien a quien le gusta tomar riesgos e intentar nuevas formas de hacer las cosas.

1.1.2.2. Áreas de oportunidad

En cuanto a las áreas de oportunidad que tengo como integrante del equipo es que a veces no organizo mi tiempo para entregar las actividades, y por tanto algunas ocasiones el trabajo no lo hago como se esperaba. Otro punto que debo de mejorar es que la mayoría de las veces no soy conciso explicando mis ideas, por lo que puede haber malentendidos en las juntas de equipo.

1.1.2.3. Expectativas

Durante este bloque espero aprender y aplicar nuevos temas de computación. Por otro lado, también espero que la resolución del reto sea una actividad que en algún momento pueda ocupar en mi vida profesional o incluso en la personal.

1.1.3 Yusdivia Molina

1.1.3.1. Fortalezas

Me gusta mucho seguir aprendiendo y siempre perfeccionar mis habilidades. Soy muy dedicada y me gusta formar opiniones e ideas con base en la evidencia y el análisis. No me gusta hacer trabajos a medias o con el mínimo esfuerzo. Soy ambiciosa y me gusta estar orientada a objetivos concretos. Siempre estoy abierta a nuevas ideas.

1.1.3.2. Áreas de oportunidad

Llego a ser impaciente, puedo llegar a ser muy crítica y no me gusta seguir a una persona, idea o regla ciegamente me gusta entender el porqué.

1.1.3.3. Expectativas

Determinar y definir si es que el área de inteligencia artificial y/o de gráficas es de mi preferencia para especializarme en algún futuro. Aprender y enriquecer mis habilidades como ingeniera en esta área.

1.2. Expectativas como equipo y compromisos

Esperamos tener un buen desempeño en todo el bloque. Esperamos desarrollar el reto final cumpliendo con las expectativas del curso y conforme lo aprendido en el curso. Para ello deberemos tener una muy buena comunicación y apoyo mutuo en todo momento. Cada integrante del equipo se compromete a trabajar de manera honesta, siempre dando lo mejor de sí. Comunicando si existiese algún percance, o una área de oportunidad en el conocimiento o producto a entregar. Asimismo, nos comprometemos a siempre entregar las actividades respetando los tiempos límites de entrega.

Dado que el desarrollo de una solución para el reto requiere tanto conocimientos técnicos como pensamiento creativo, en el equipo esperamos seguir un flujo de trabajo que favorezca ambos enfoques. Cuando se presente un problema, cada miembro deberá de pensar en posibles soluciones de manera individual, realizando la investigación conveniente. Posteriormente, tendremos una sesión de ideación en la que se compartirán las propuestas y se analizarán sus ventajas y desventajas. En este punto debemos tomar en cuenta la factibilidad técnica, así como definir un plan de implementación. Finalmente, se repartirán las labores de desarrollo para agilizar el proceso. Claramente, cada integrante se compromete a cumplir con sus asignaciones a su propio ritmo, pero con cierta fecha límite definida por nosotros mismos.

Por otro lado, nos comprometemos a estar constantemente revisando los canales de comunicación con el equipo y con los profesores. Además de que nos comprometemos a apoyarnos entre nosotros al revisar los entregables y escuchar de manera atenta y con respeto las ideas de los demás. Adicionalmente, esperamos tener un ambiente de trabajo ameno, con el objetivo de tener una mejor productividad y mayor compromiso con el proyecto y con el equipo. Todo lo anterior con la finalidad de tener una sinergia que nos permita seguir creciendo y aprendiendo tanto conocimientos técnicos como las habilidades relacionadas al trabajo en equipo. Anteriormente habíamos trabajado en equipo, por lo que este aspecto ya tiene cierto grado de avance.

2. Creación de herramientas de trabajo colaborativo

2.1. Repositorios de GitHub

Gráficas computacionales: <https://github.com/E1-CarpoolProject/carpool-graphics>

Multiagentes: <https://github.com/E1-CarpoolProject/carpool-multiagents>

3. Propuesta formal del reto

3.1. Descripción del reto a desarrollar

La movilidad urbana habla acerca del desplazamiento que tienen las personas y la mercancía dentro de una ciudad. Este objetivo de llegar de un lado al otro se hace utilizando diferentes medios de transporte como el público y privado, de forma peatonal, entre muchas otras. Por otro lado, desde la segunda revolución industrial se consideraba al automóvil como un símbolo de progreso. Además que desde su invención supuso una revolución en el mundo del transporte, al ser más seguro y cómodo (Muñoz, 2015).

Actualmente, el uso indiscriminado de esta gran herramienta ha generado efectos negativos en diferentes sectores. Por ejemplo, nos ha traído problemas de contaminación al medio ambiente, horas perdidas en el tráfico e inclusive accidentes que se podrían fácilmente evitar. Estos problemas enlistados y demás comprometen el bienestar y la calidad de vida de las personas.

Añadiendo que este panorama se complica año con año, ya que según el INEGI en México el número de automóviles en circulación aumentó en un 25.9% del 2015 al 2020 (2021).

Ante esta necesidad se deben plantear nuevas soluciones las cuales cuentan con planes de movilidad que permita responder ante las necesidades de la ciudadanía. Es decir, construir ciudades incluyentes y sustentables, donde se tenga como punto central el desarrollo del ser humano. Una parte importante de una ciudad incluyente es tener un plan de enfoque que reduzca la congestión vehicular. Ante ello, desarrollaremos una propuesta de solución, utilizando herramientas de gráficas computacionales y representando la salida de un sistema multiagentes.

En particular, se presentará un modelo de solución a un problema muy específico al que estamos expuestos de manera frecuente: el *carpooling*. Consiste en compartir el vehículo con otras personas de tal manera que la ruta del conductor no sea drásticamente modificada, y se pueda dejar a cada uno de los pasajeros en sus destinos durante el viaje. Esta práctica tiene un gran número de beneficios sociales y ambientales. Debido a que se utilizan menos vehículos, se reduce el consumo de combustibles y la generación de gases del efecto invernadero, lo que a su vez disminuye la contaminación del aire. Para el conductor y los pasajeros también hay beneficios que van desde una mejora de productividad y estado de ánimo, hasta una mejora notable en las finanzas personales puesto que el gasto de los trayectos se reparte equitativamente (Shaheen et al, 2018).

A pesar de que el carpooling es altamente atractivo, también existen muchas problemáticas que dificultan su implementación. La mayoría de las soluciones de *carpooling* están basadas en la tecnología, puesto que permite vincular fácilmente a los conductores y los pasajeros por medio de un dispositivo móvil. El sistema computacional debe de procesar una gran cantidad de información para determinar la repartición óptima de los pasajeros entre el conjunto de conductores. Sin embargo, existen varias maneras de evaluar la optimización, puesto que se podrían minimizar variables como el tiempo total estimado de recorrido, la distancia global recorrida, la cercanía de los vehículos con los pasajeros, etc. En este sentido, modelar el sistema utilizando multiagentes es favorable puesto que cada uno de ellos tiene un objetivo específico y un conjunto de reglas a seguir.

Adicionalmente, existe el problema de la infraestructura. Al ser tan grande la cantidad de datos a evaluar, una computadora normal sería subóptima para el procesamiento. Además, en una implementación de la vida real, debe de existir algún mecanismo centralizado de control con la capacidad de dar servicio concurrente y en tiempo real a todos los usuarios. Para esto se pueden utilizar plataformas en la nube, que virtualiza la infraestructura y permite una escalabilidad sencilla. Es por esto que la participación de IBM como socio formador es de gran valor para nuestra solución, pues nos permitirá implementar un sistema de manera similar a como se haría en el mundo real, utilizando tecnología de alto nivel.

3.2. Identificación de los agentes involucrados

Para la solución del carpooling se deben tomar en cuenta cuatro agentes principales:

- Automóvil (junto con conductor): Cada vehículo debe considerarse como un agente que parte de un punto inicial y desea llegar a un destino final. Su propósito es recoger a ciertos pasajeros en el camino y llevarlos a su propio destino. Sin embargo, le interesa realizarlo maximizando el número de pasajeros y minimizando el aumento en la distancia recorrida en comparación con la ruta original.
- Pasajeros (usuarios): Los pasajeros también tienen un punto inicial y un destino final. Desean que un automóvil pase a recogerlos, pero prefieren ser asignados al que esté más cercano para reducir su tiempo de espera. A ellos no les afecta la ruta final de automóvil, pero preferirían que su recorrido específico también sea lo más directo posible.
- Semáforos: Estos agentes se encargan de controlar el flujo de vehículos en los cruces. Para este caso particular no se espera optimizar su comportamiento. Sin embargo, deben de seguir ciertas reglas básicas de sincronización, por lo que es útil plantearlos como agentes que pueden comunicarse entre sí. Cada semáforo pertenece a un cruce.
- Intersección: Esta serie de agentes se encargan de manejar, administrar y coordinar los semáforos en cada cruce que exista entre los coches. Estos se encuentran sincronizados de tal manera que toda la columna avanza a la par y se guarda un orden para evitar tráfico y embotellamientos.
- Calle: Este agente contiene la información acerca de la dirección por dónde los coches pueden circular.
- Banqueta: Agente que sirve para colocar a los pasajeros, tanto en un punto inicial como en uno final.
- CarpoolModel: Este modelo se encarga de dar información acerca de la topología a los agentes automóviles.. Cumple una función similar a lo que haría Google Maps o Waze. Para realizar los cálculos, se realizará una abstracción de la ciudad a una estructura de matriz, en la que habrá nodos de tipo cruce y de tipo pasajero.

3.2.1. Diagrama de clase

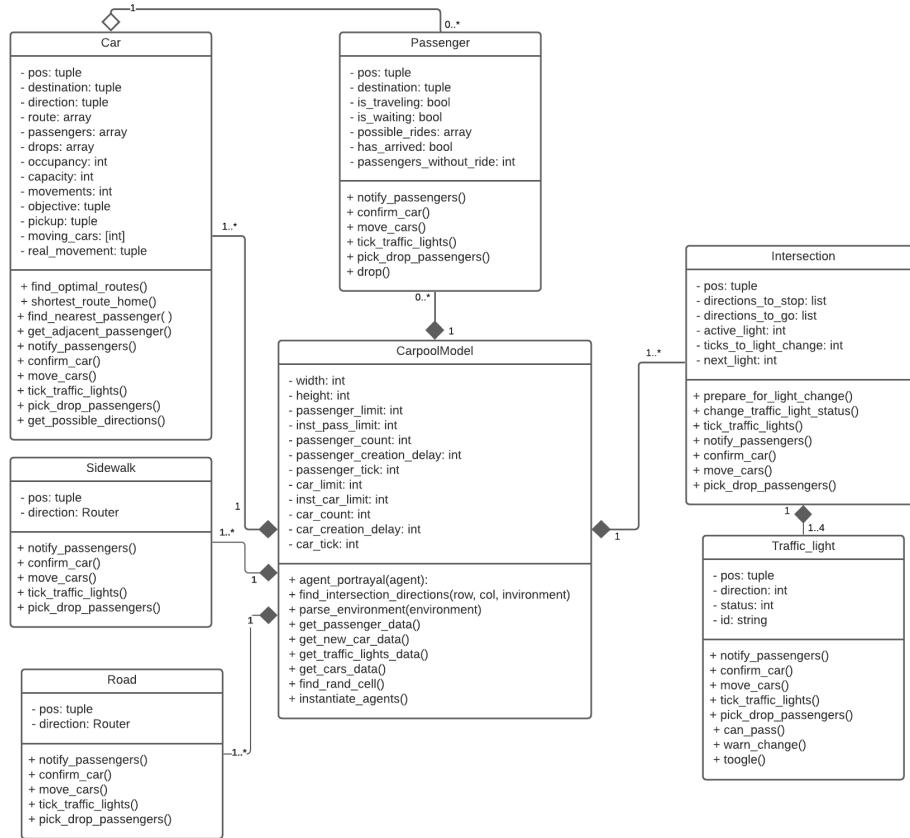


Imagen 1 Diagrama de clases

3.2.2. Diagrama de protocolos de interacción

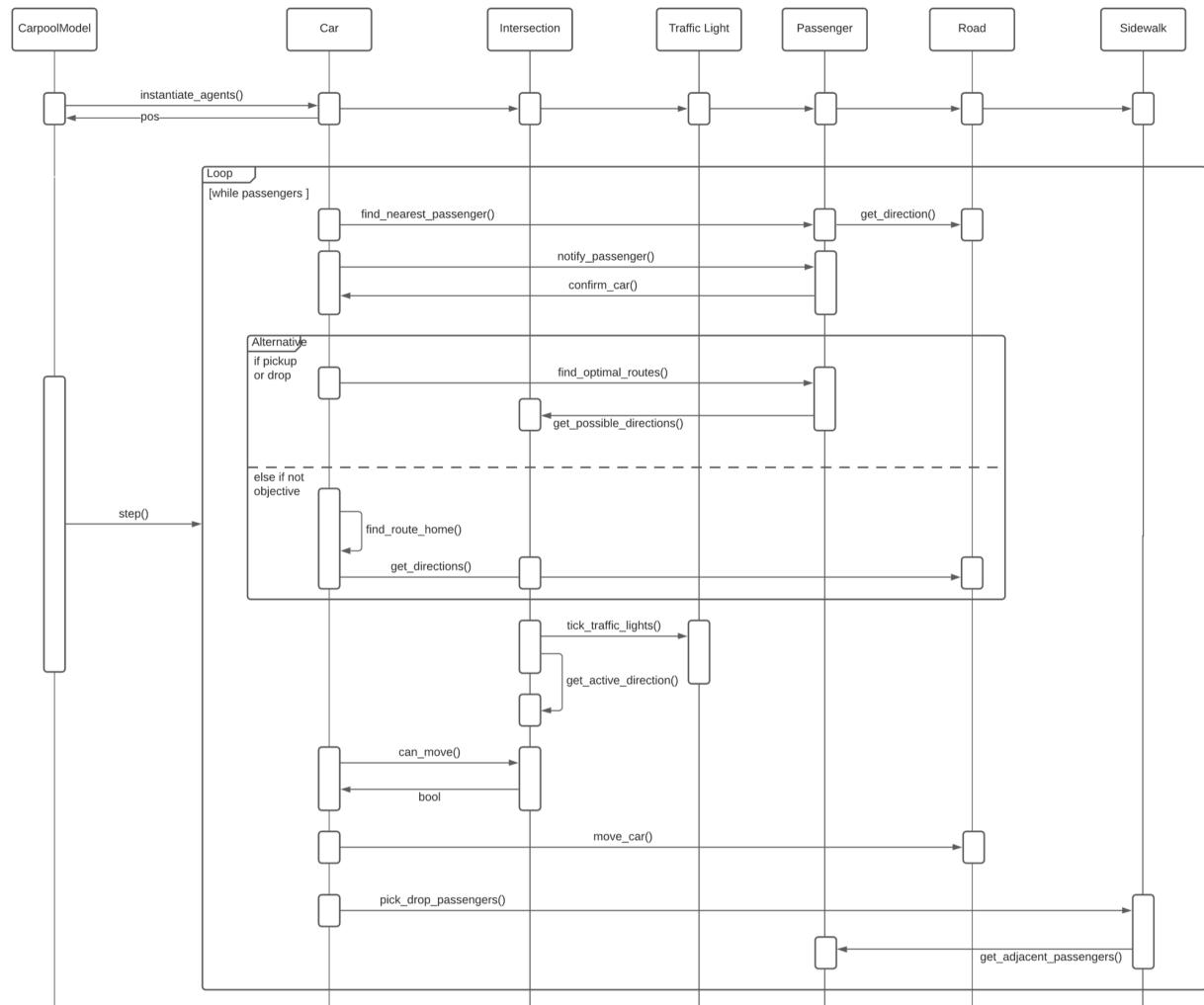


Imagen 2 Diagrama de secuencia

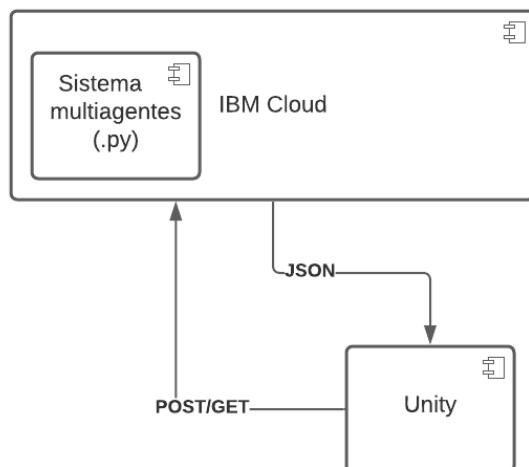


Imagen 3 Diagrama de componentes

3.3. Plan de trabajo y aprendizaje adquirido

3.3.1. Plan de trabajo

Como una herramienta para tener una mejor visión acerca de lo que se solicita en cada entregable y en el reto en general, se realizó un plan de trabajo, el cual hasta el día viernes 12 de noviembre, queda de la siguiente forma:

PLAN DE TRABAJO EQUIPO 1

TAREA/ACTIVIDAD	FECHA INICIO	FECHA FIN	INICIA EN	DURACIÓN EN DÍAS	ESFUERZO ESTIMADO	RESPONSABLE	PORCENTAJE COMPLETO
Semana 2							
Etapa 1.1	11/8	11/12	0	4	5	Todos	95%
Conformación del equipo	11/9	11/10	1	1	1	Todos	100%
Descripción del reto a desarrollar	11/8	11/10	0	2	1	Yus	100%
Diagrama de clases	11/10	11/12	2	2	2	Agustín	100%
Diagrama de protocolos de interacción	11/10	11/12	2	2	3	Diego	100%
Plan de trabajo	11/8	11/9	0	1	1	Agustín y Yus	100%
Aprendizaje adquirido	11/11	11/12	3	1	1	Diego y Agustín	100%
Referencias y subir el documento	11/12	11/12	4	0	1	Yus y Diego	90%
Conseguir modelos 3D	11/8	11/12	0	4	2	Agustín	85%
Registro a IBM Cloud	11/10	11/11	2	1	1	Todos	100%
Configuración de repositorio GitHub	11/9	11/10	1	1	2	Agustín y Yus	100%
Presentación de avances	11/11	11/12	3	1	1	Todos	100%
Semana 3							
Etapa 1.2	11/15	11/19	7	4	21	Todos	85%
Diseño del sistema 3d (calles)	11/15	11/17	7	2	13	Agustín y Yus	90%
Simulación en Unity	11/16	11/18	8	4	13	Diego y Agustín	20%
Simulación de coches en línea recta	11/17	11/19	9	2	8	Todos	100%
Simulación coches dando la vuelta	11/17	11/19	9	2	8	Agustín	100%
Simulación de semáforos en funcionamiento	11/15	11/18	7	3	8	Yus y Diego	50%
Diseño de un coche por nosotros	11/17	11/19	9	2	13	Diego	100%
Entregable .py	11/17	11/19	9	2	8	Todos	40%
Definición clases con atributos y métodos	11/18	11/19	10	1	8	Yus	40%
Semana 4							
Etapa 2.1	11/22	11/26	14	4	34	Todos	85%
Identificación de negociación entre agentes	11/22	11/23	14	1	34	Diego	100%
Simulación de negociación	11/23	11/26	15	3	34	Agustín	60%
Entregable en Unity	11/22	11/26	14	4	21	Diego y Agustín	80%
Movimiento de los coches en Unity	11/24	11/26	16	2	21	Yus y Diego	100%
Conexión Unity con IBM Cloud	11/25	11/26	17	1	13	Diego y Agustín	90%
Entregable .py	11/22	11/25	14	3	21	Yus y Diego	60%
Semana 5							
Etapa 2.2.	11/29	12/1	21	2	21	Todos	100%
Entregable en Unity	11/29	11/30	21	1	34	Diego y Agustín	100%
Comunicación usando IBM cloud	11/29	12/1	21	2	21	Yus y Diego	100%
Entregable .py	11/29	11/30	21	1	34	Agustín y Yus	100%
Documentación final	11/30	12/1	22	1	13	Todos	100%
Presentación socio formador	12/1	12/1	23	0	3	Todos	100%

*Se usa la serie de Fibonacci
1,1,2,3,5,8,13,21,34

Imagen 4 Plan de trabajo

En este podemos observar que las actividades para la segunda semana del bloque están casi cubiertas en su totalidad. Dejando como pendiente subir este documento y encontrar más

modelos 3D que nos puedan ser de utilidad para la resolución del reto. Asimismo, las actividades de las semanas subsecuentes aún no se encuentran totalmente detalladas, esto lo estaremos realizando conforme desarrollemos el proyecto. Por otro lado, decidimos estimar el esfuerzo de cada actividad con la serie de Fibonacci; donde podemos ver que actividades como crear una cuenta está dentro de las más fáciles y las relacionadas al código se encuentran con una mayor complejidad ya que creemos será lo que más nos tomará tiempo de realizar.

Asimismo, realizamos un diagrama de Gantt con el objetivo de visualizar de mejor manera la ruta crítica del proyecto, y qué tareas son las que podemos paralelizar, para aprovechar de mejor manera los recursos con los que contamos (en este caso el tiempo). Este quedó de la siguiente manera:

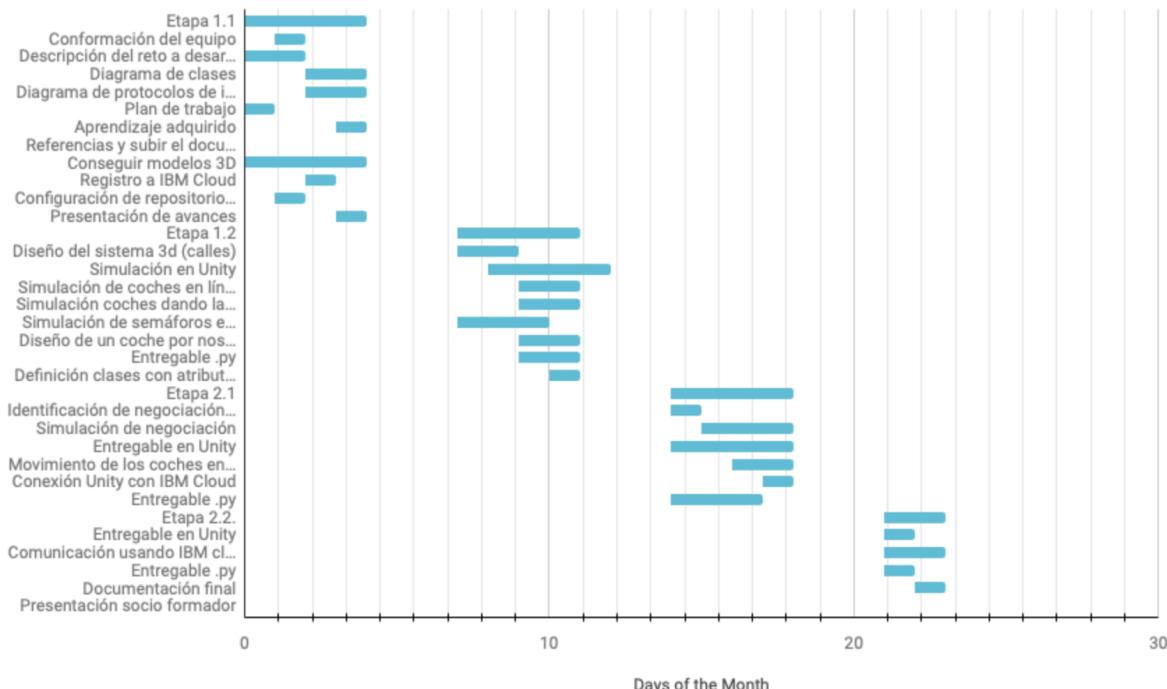


Imagen 5 Diagrama de Gantt

Para una mejor visualización del plan de trabajo se puede consultar la liga siguiente:

<https://docs.google.com/spreadsheets/d/1VUu2x0GdWkeFjR5Wqz8ogRXiVcVxM9z49T1Q9Nsx73w/edit?usp=sharing>

3.3.2. Aprendizajes adquiridos

Desde el inicio de este bloque, hemos tenido múltiples conocimientos en torno a dos ejes centrales, las gráficas computacionales y sistemas multiagentes; mismos que serán desarrollados a continuación.

3.3.2.1 Gráficas computacionales

Durante las semanas 1 y 2 se adquirieron conocimientos acerca de gráficas computacionales, desde cómo se crea una figura en tres dimensiones, hasta las nuevas tecnologías que ayudan a

que un objeto se vea más real. Las principales herramientas que se han utilizado para aprender sobre estos temas son las matemáticas, que nos dan la teoría necesaria para entender el tema, y Unity, como motor gráfico para aplicar los conocimientos obtenidos durante las clases. Algunos de los temas vistos en clase son:

- Modelación en 3D
 - *Rendering Pipeline*
 - Geometría
 - Topología
 - GPU
 - Mallas poligonales
 - Archivos OBJ
- Transformación de figuras geométricas
 - Traslación
 - Escalabilidad
 - Rotación
- Iluminación
 - Iluminación difusa
 - Iluminación especular
 - Iluminación ambiental
- Texturas
 - UV *mapping*
 - *Normal map*

Durante la semana 3 se vieron los siguiente temas:

- Radar VFC -> Optimización
- Diagrama de arquitectura de la situación problema
 - Ejemplo con Unity y un servidor HTTP hecho por Python

Durante la semana 4 se tuvieron los siguientes aprendizajes:

- Conexión con localhost
- Conexión Unity - IBM Cloud
- Repaso general acerca de iluminación y transformación de figuras

Finalmente, durante la semana 5 se revisó el tema de shaders

3.3.2.2 Sistemas multiagentes

Ahora bien, como se mencionó anteriormente, la siguiente área de la computación que se ha estudiado, son los agentes computacionales. En el desarrollo del reto, este tema será sumamente importante para que la solución que se busca tenga la suficiente capacidad para que la interacción de los diversos agentes sea la esperada. Las principales herramientas que se han utilizado para este tema es Python, como ambiente de aplicación de los conocimientos obtenidos. Algunos de los temas vistos en clase son:

- Agentes computacionales

- Agentes inteligentes
- Arquitectura de agentes
 - De razonamiento deductivo
 - De razonamiento práctico
- Modelación e implementación de agentes inteligentes
 - Modelado de agentes
 - Librerías de desarrollo
 - Mesa
- Comunicación entre agentes
 - Mecanismos de comunicación
 - Cooperación entre agentes
 -

Durante la semana 3 se vieron los siguiente temas:

- Optimización
 - Búsqueda exhaustiva (TSP)
 - Búsqueda aleatoria
- Hill climbing
- Gradiente descendiente y el método de Newton
- Algoritmos genéticos
 - Ciclo de reproducción
 - Mutación
 - Crossover
- Mesa
 - Grid
 - Estadística
- Optimización por enjambre de partículas

En la semana 4 tuvimos los siguientes aprendizajes:

- Teoría de juegos
 - Tipos de juegos
- Aprendizaje por refuerzo
 - Machine Learning
 - Deep learning
- Machine learning
 - Aprendizaje supervisado
 - Aprendizaje no supervisado
 - Aprendizaje por refuerzo
- Dilema de explorar y explotar

3.3.3. Las actividades pendientes y el tiempo en el que se realizarán.

Como se ha concluido exitosamente con este bloque, no se tienen actividades pendientes por realizar relacionadas al reto.

3.3.4. Actividades planeadas

Se puede ver en nuestro plan de trabajo las actividades realizadas, los responsables, las fechas establecidas por nosotros mismos según las fechas de entrega del reto y un número de la serie de Fibonacci para estimar el esfuerzo que implica cada actividad.

4 Resultados e instalación

4.1 Proceso de instalación

Proceso de instalación configuración y ejecución de la simulación:

Para poder correr de manera efectiva la simulación creada del carpool dentro de una ciudad, se necesita instalar el motor de videojuegos “Unity”.

Para ello es necesario primero descargar “Unity Hub”, esto se hace entrando a su sitio web:

<https://unity3d.com/es/get-unity/download>

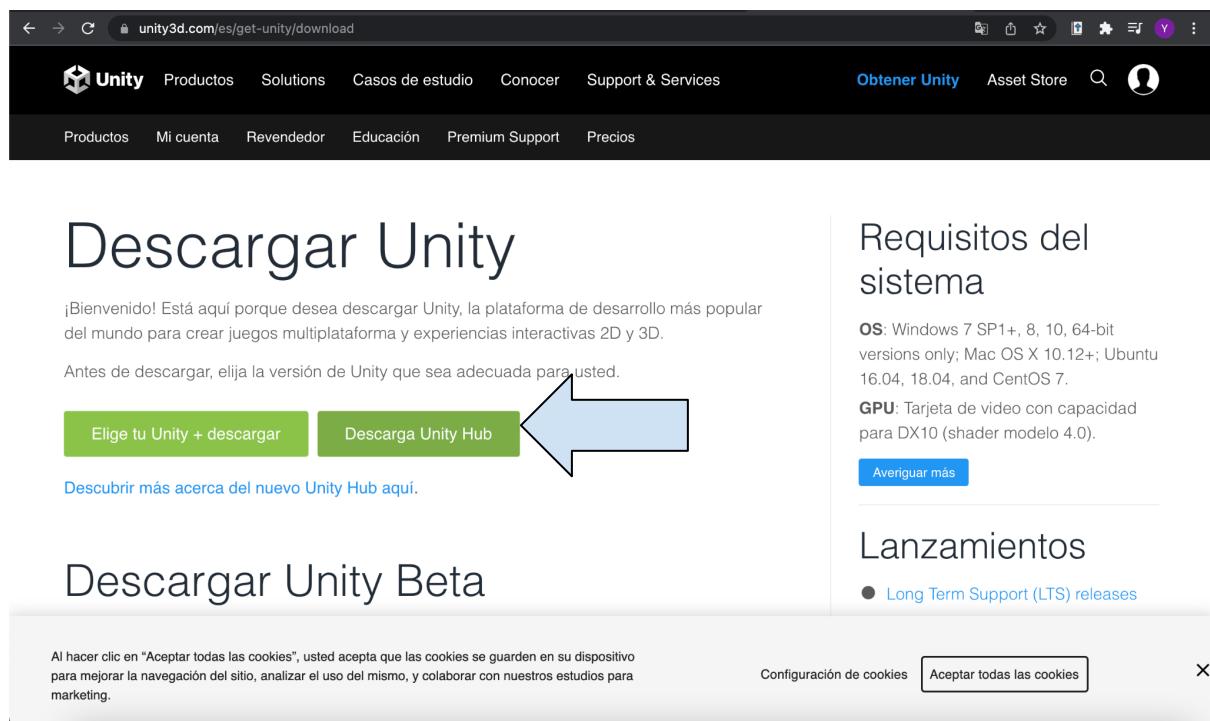


Imagen 6 Descarga Unity Hub

Como se muestra en la figura 6 se le da click a la parte de “Descarga Unity Hub”. Una vez completado el proceso se abre el archivo descargado, se leen y aceptan las políticas, como se muestra a continuación:

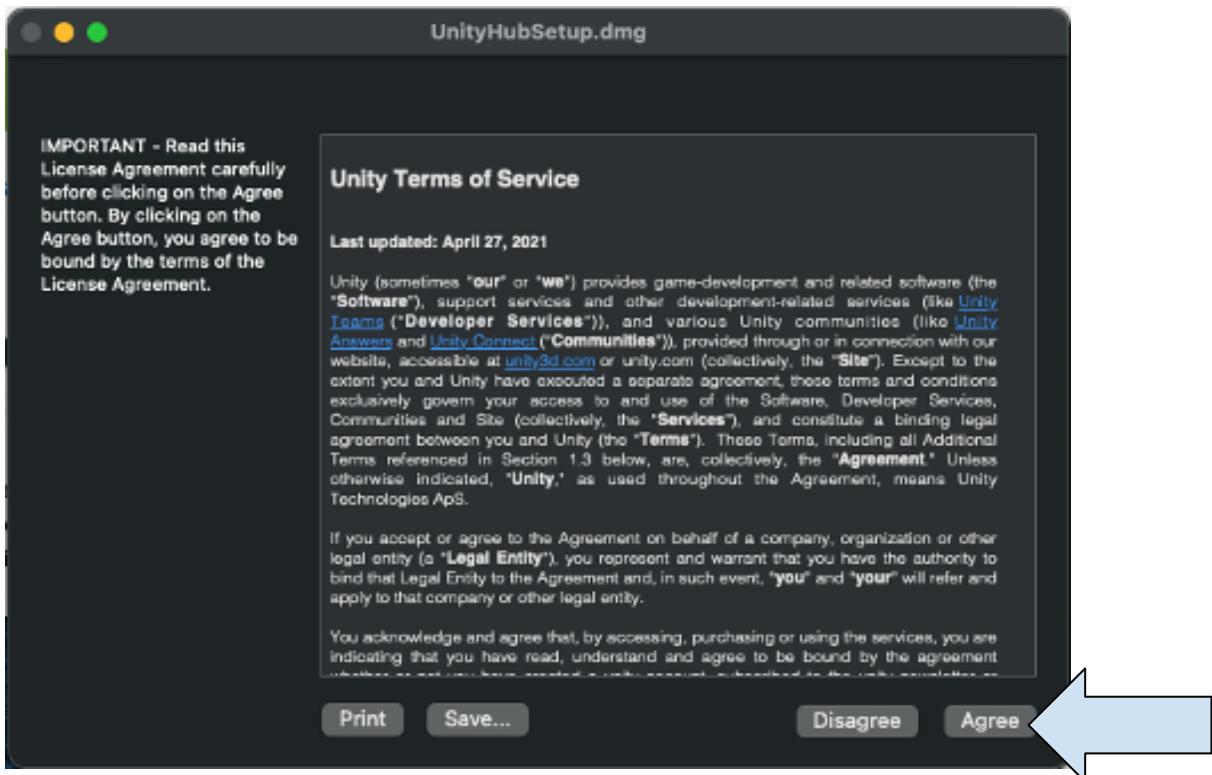


Imagen 7 Aceptación Términos y condiciones

Automáticamente se le solicitará mover Unity Hub a su carpeta de aplicaciones.

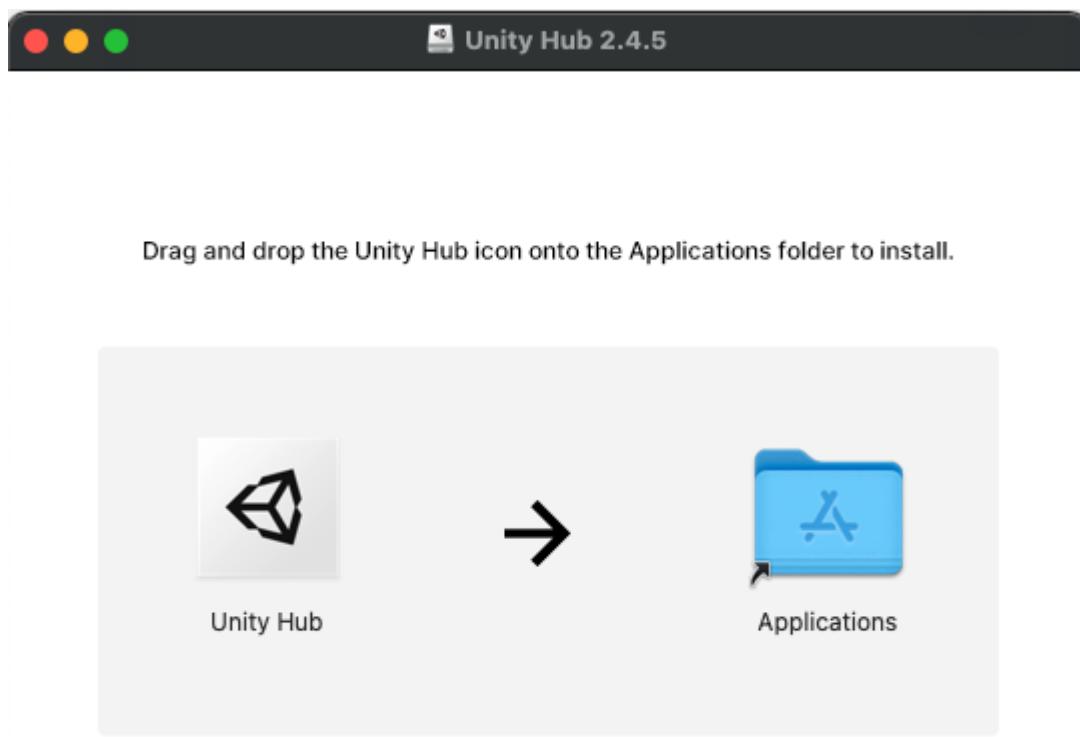


Imagen 8 Mover Unity Hub a Aplicaciones

Posteriormente, abra esta aplicación, dando click sobre su ícono. Si aparece una advertencia como la mostrada en la imagen XX, puede darle "open".

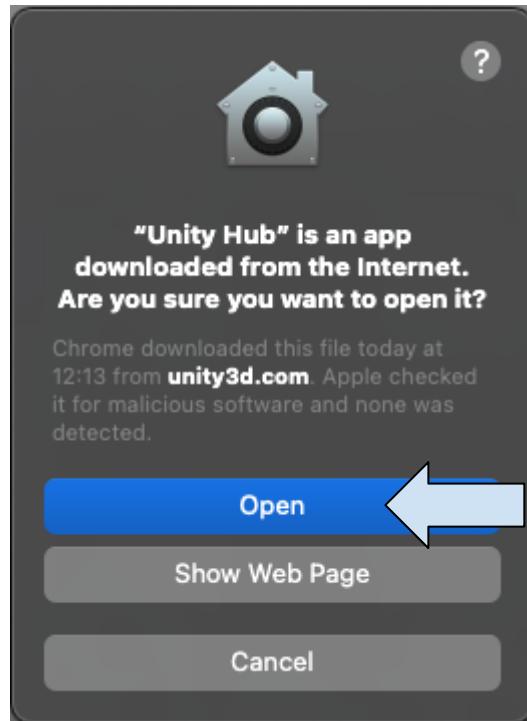


Imagen 9 Abrir Unity Hub

Se abrirá algo parecido a la imagen 9, solo que vacío en la parte de proyectos.

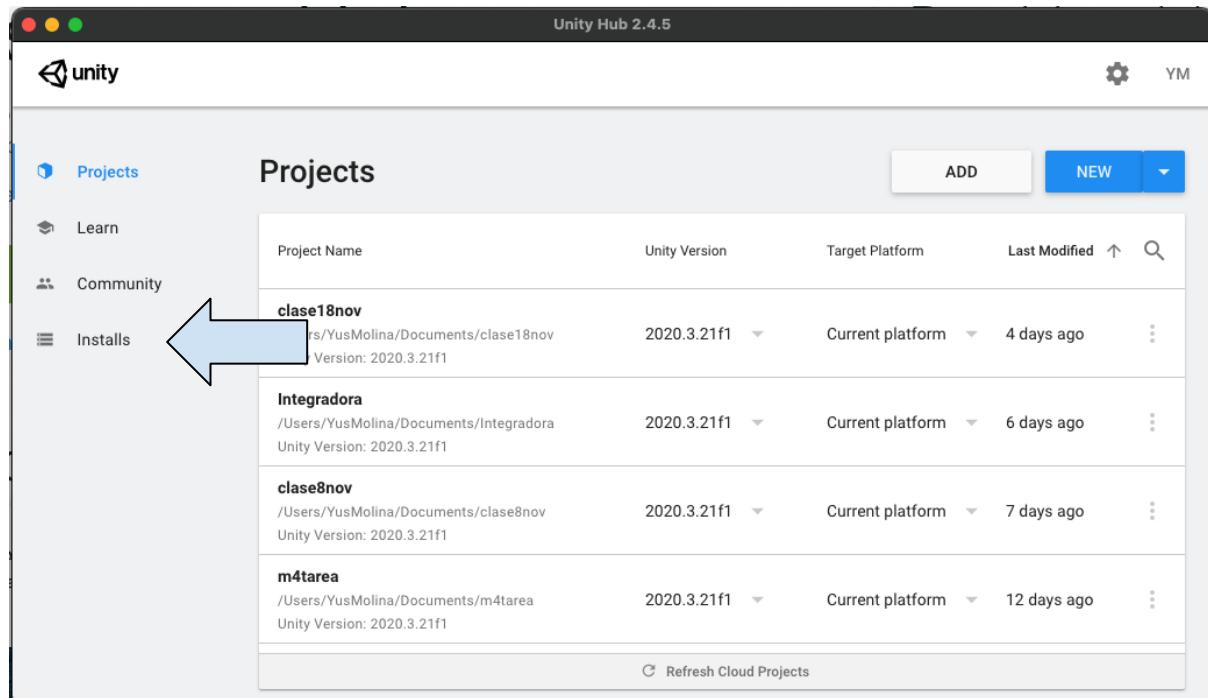


Imagen 10 Pantalla principal Unity Hub

Dará click en la parte de “Installs”, como se muestra en la imagen 10.

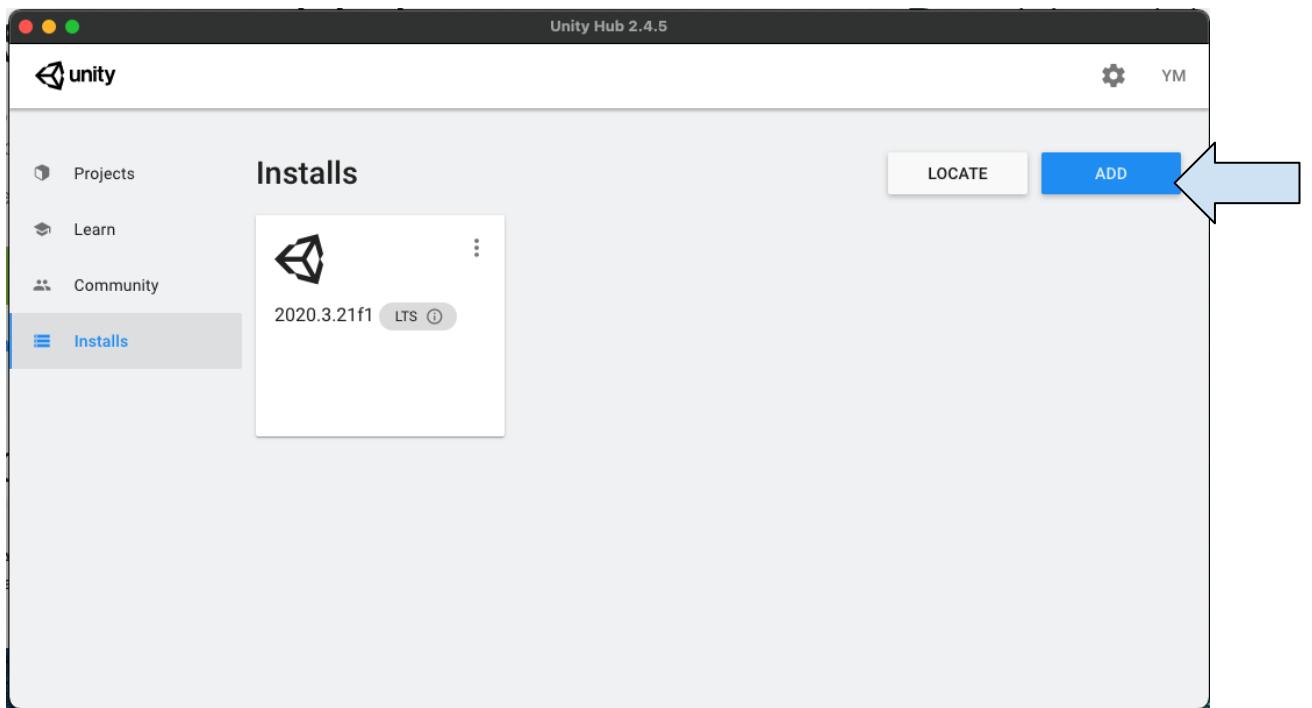


Imagen 11 Pantalla “Installs” en Unity Hub

Dé click en “ADD” como se muestra en la imagen anterior (Imagen 11).

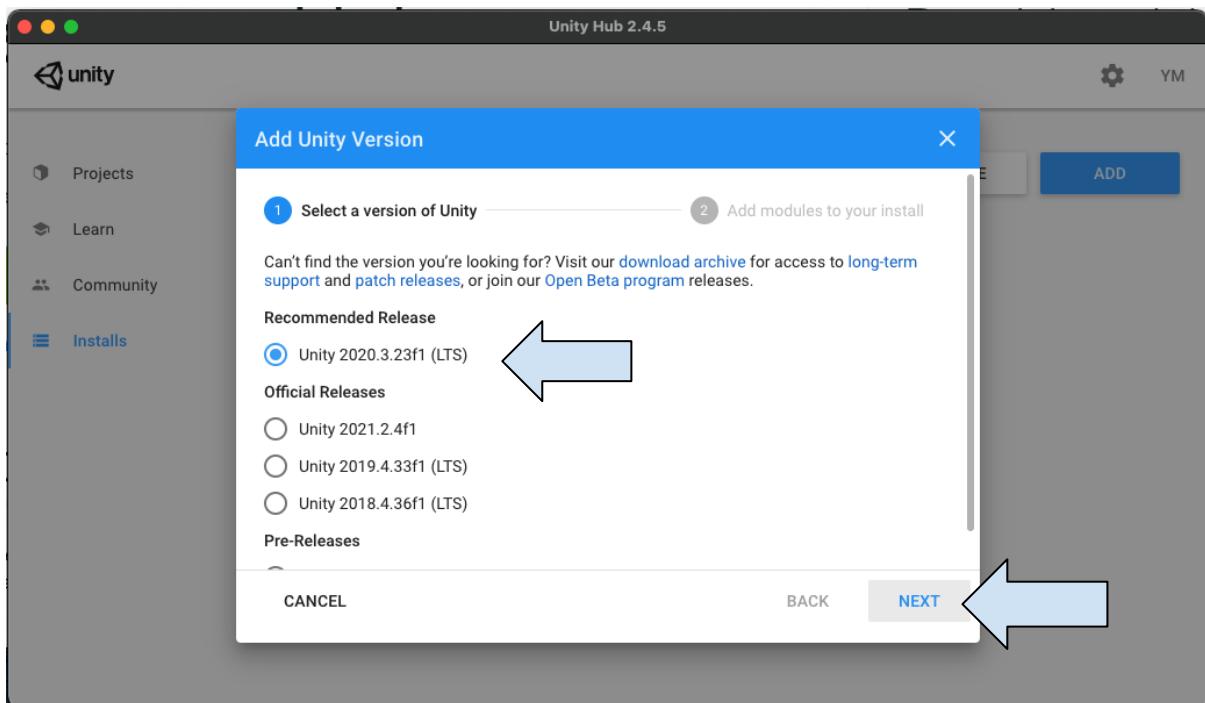


Imagen 12 Instalación de Unity parte 1

Seleccione la versión de Unity 2020.3.23f1. Posteriormente, dé click en “NEXT”.

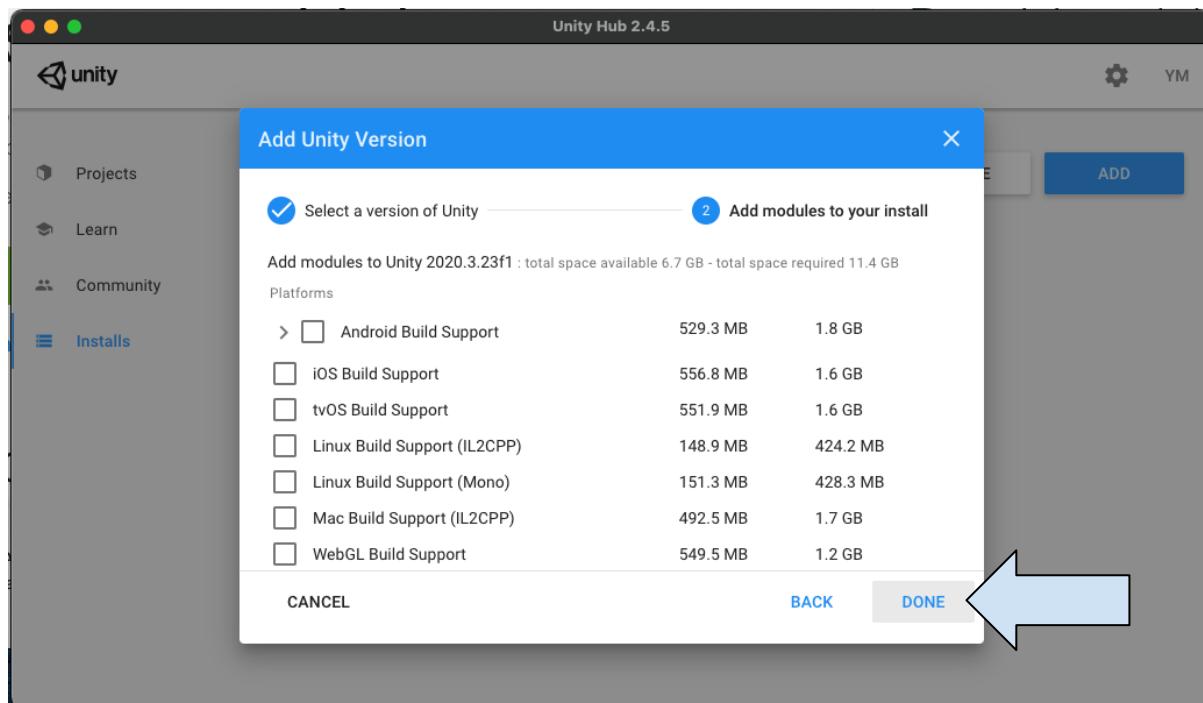


Imagen 13 Instalación de Unity parte 1

No debe seleccionar nada más adicional, puede dar click en “DONE”

Posteriormente, espere hasta que la descarga se haya completado, asegúrese que tiene el espacio en memoria suficiente para realizar la descarga.

Una vez la descarga finalice se debe visualizar algo similar a esto:

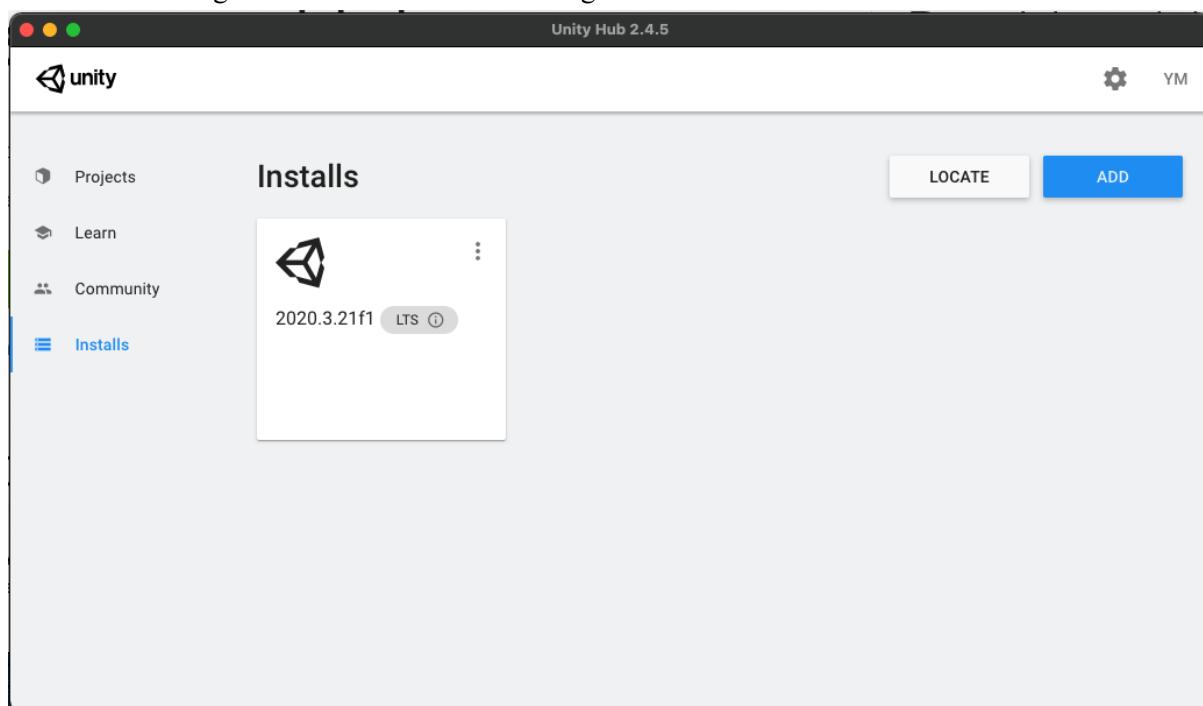


Imagen 14 Pantalla “Installs” en Unity Hub

Regresamos a la pantalla de proyectos:

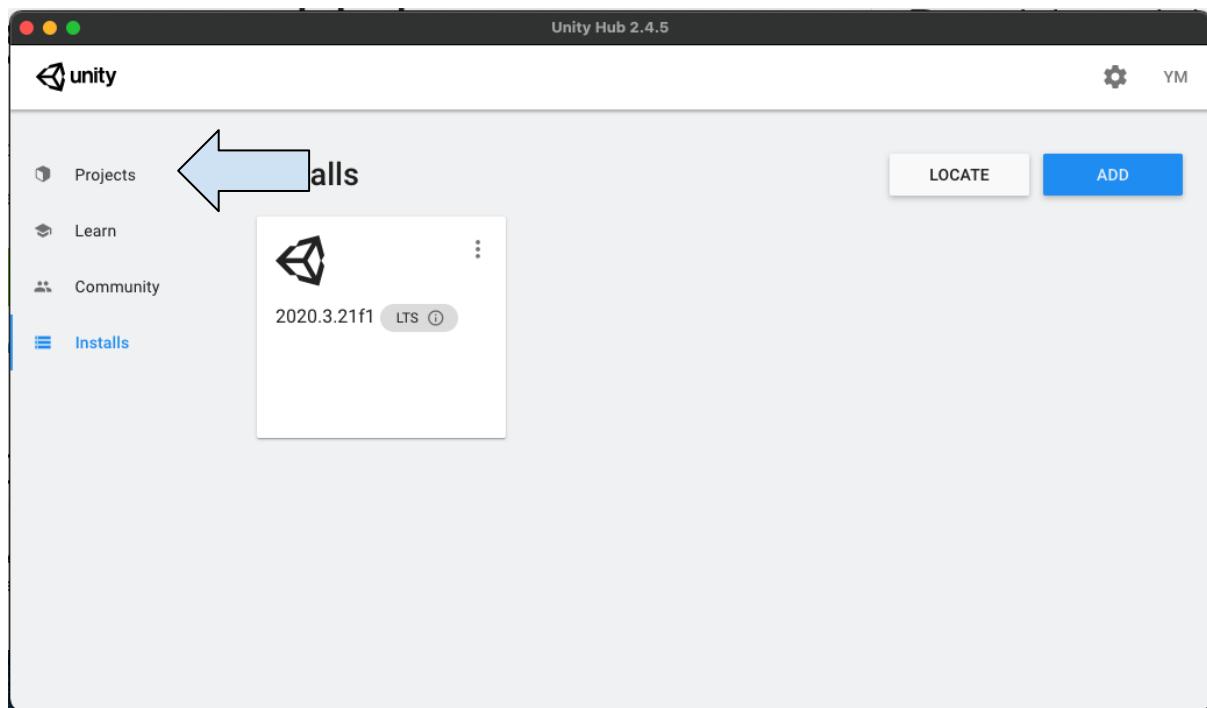


Imagen 15 Pantalla “Installs” en Unity Hub

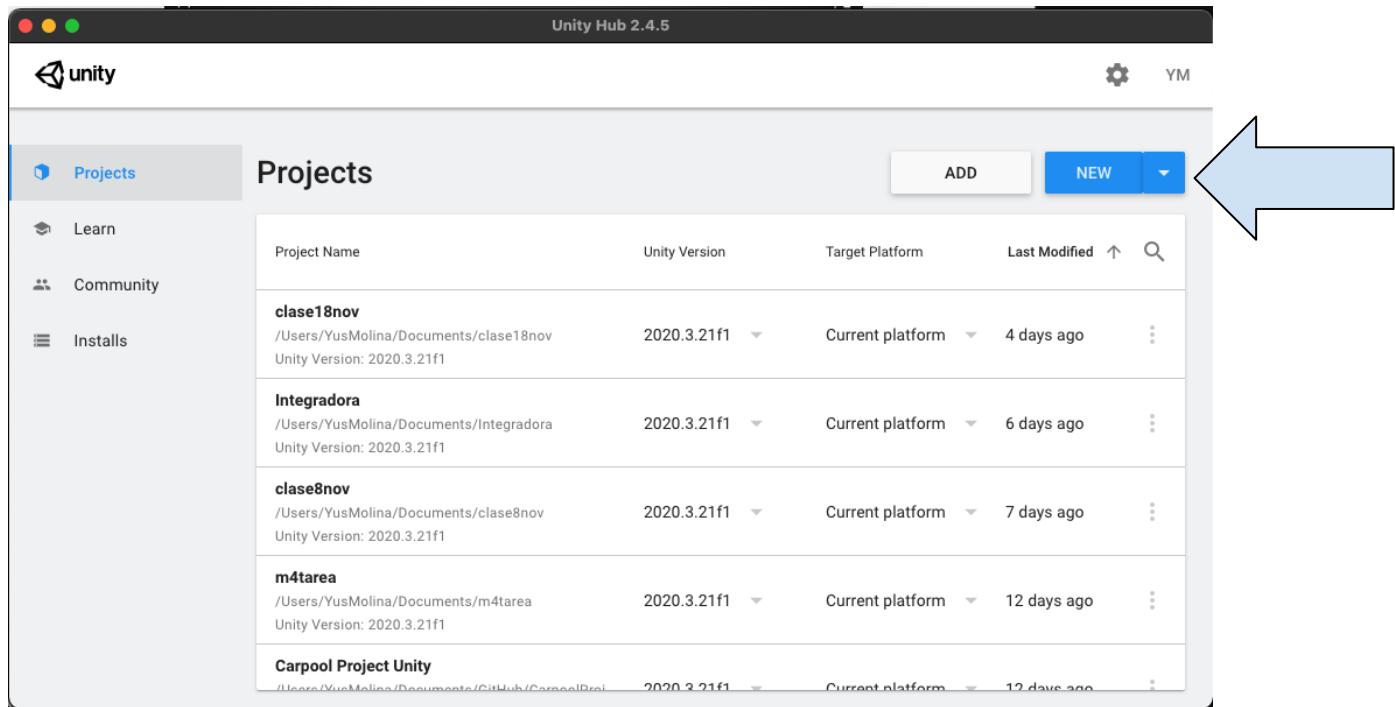


Imagen 16 Pantalla “Projects” en Unity Hub

Una vez en los proyectos se da click a “NEW” para crear un nuevo proyecto en Unity.

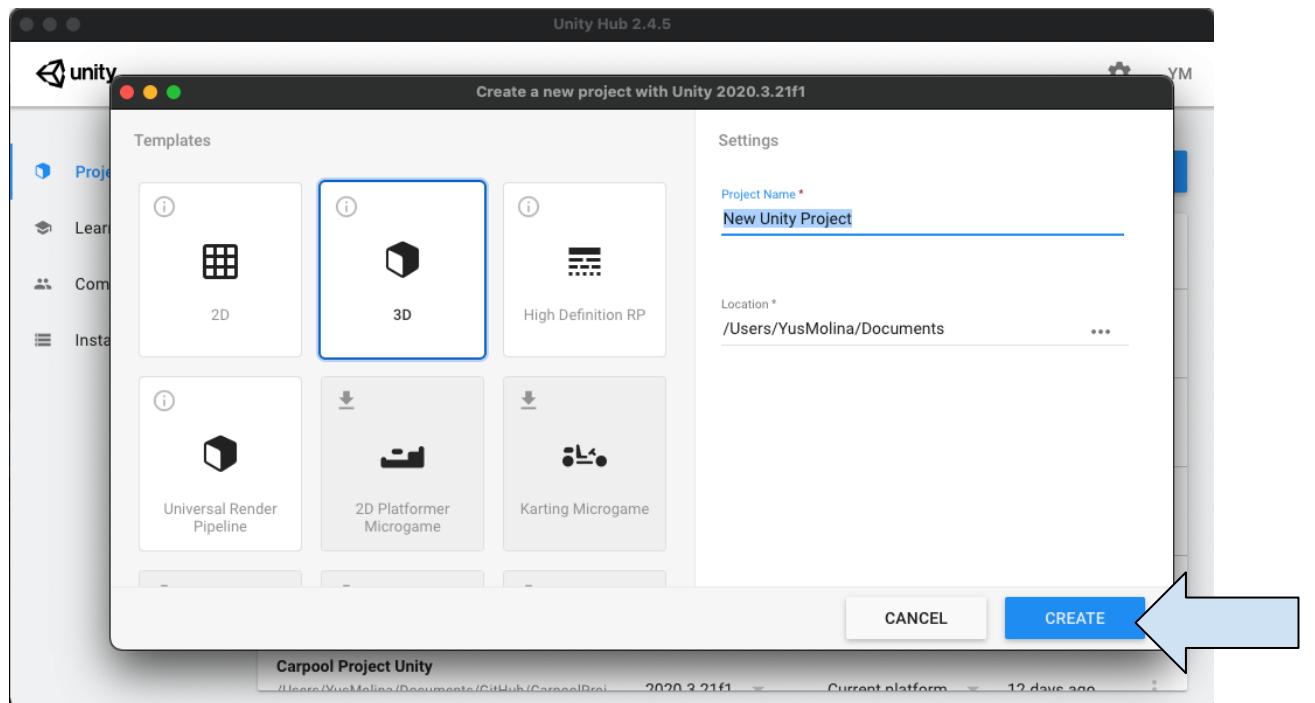


Imagen 17 Creación de un nuevo proyecto

Se le da el nombre deseado al proyecto, se escoge la opción de 3D, se le asigna el directorio donde se va a crear. Al terminar de configurar esto, se le da “CREATE”; hay que esperar unos segundos en lo que se genera el archivo. Tal y como se muestra en la figura 17.

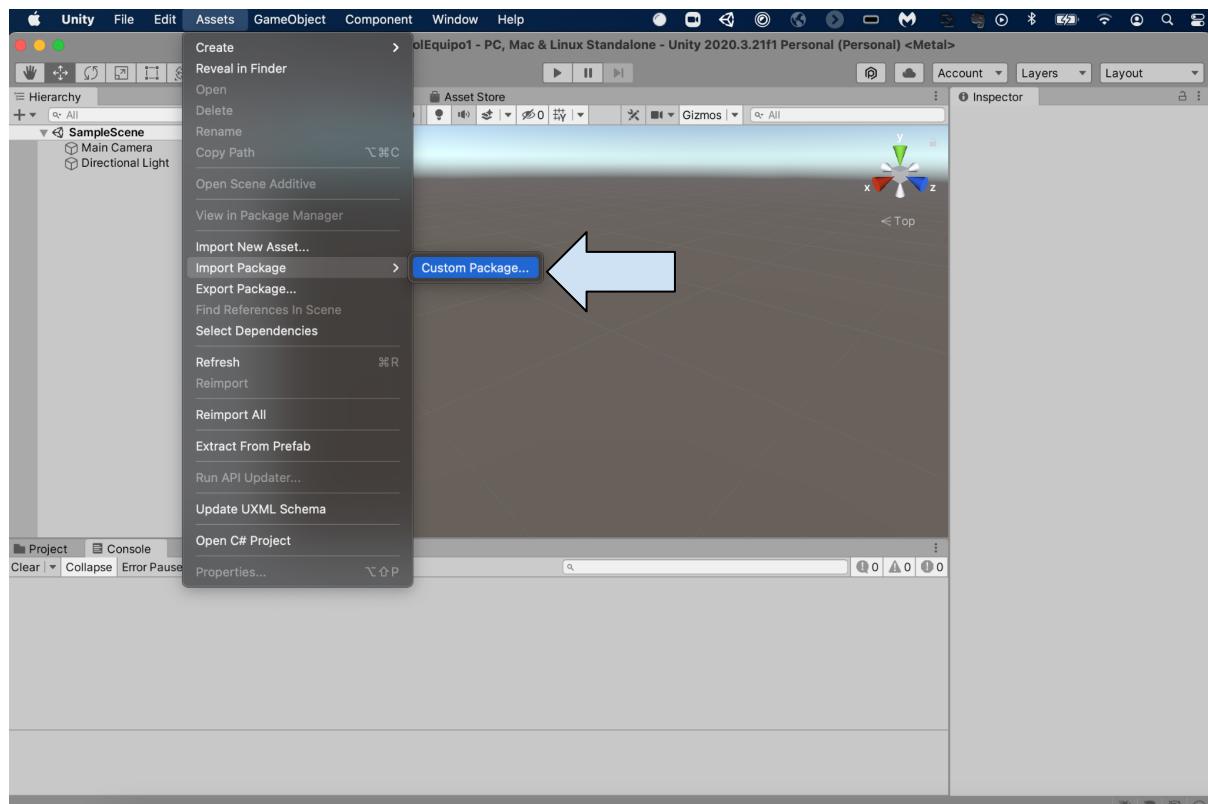


Imagen 18 Importar un Package

Una vez abierto el proyecto en blanco, te diriges a “Assets” -> “Import Package” -> “Custom Package...”. Se desplegará su directorio para ubicar y abrir el archivo .unitypackage generado por el equipo.

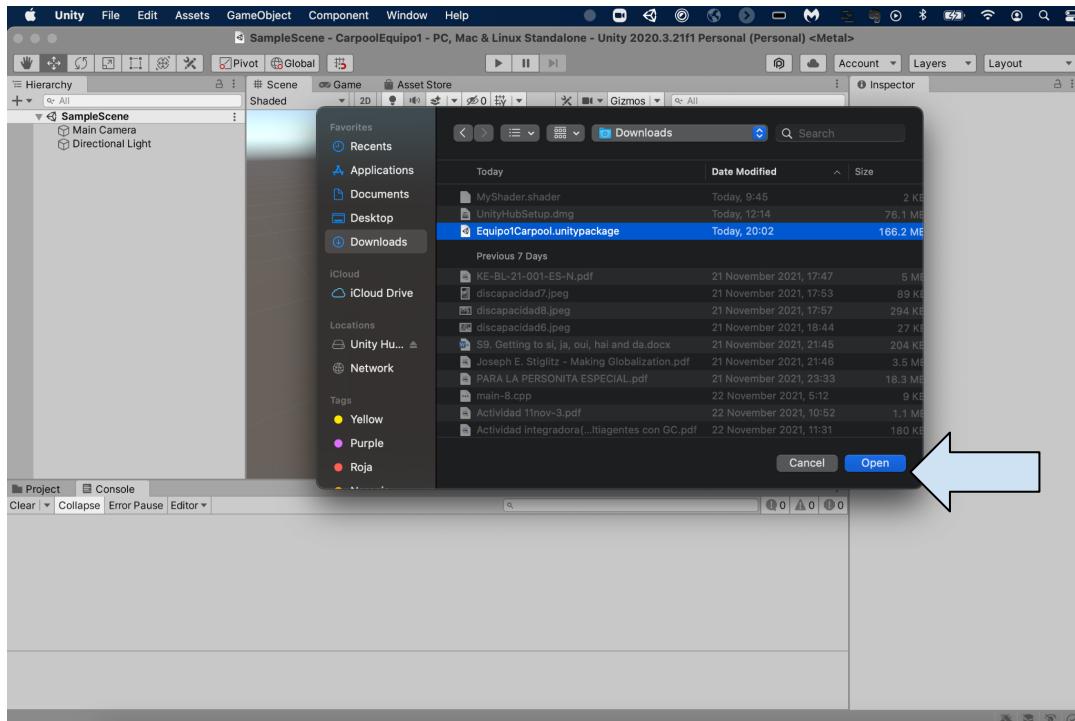


Imagen 19 Abrir el package

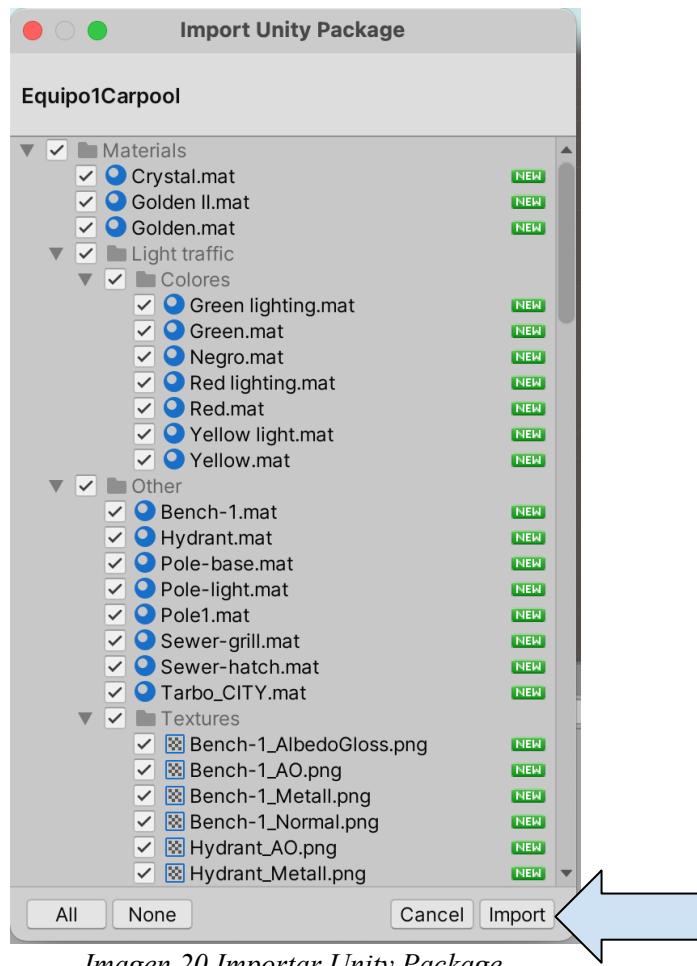


Imagen 20 Importar Unity Package

Una vez abierto el archivo se preguntará si se quiere importar todos los archivos. Cuando todos estén seleccionados (por defecto aparece de este modo), se le da click en “Import”. Como se muestra en la figura 20.

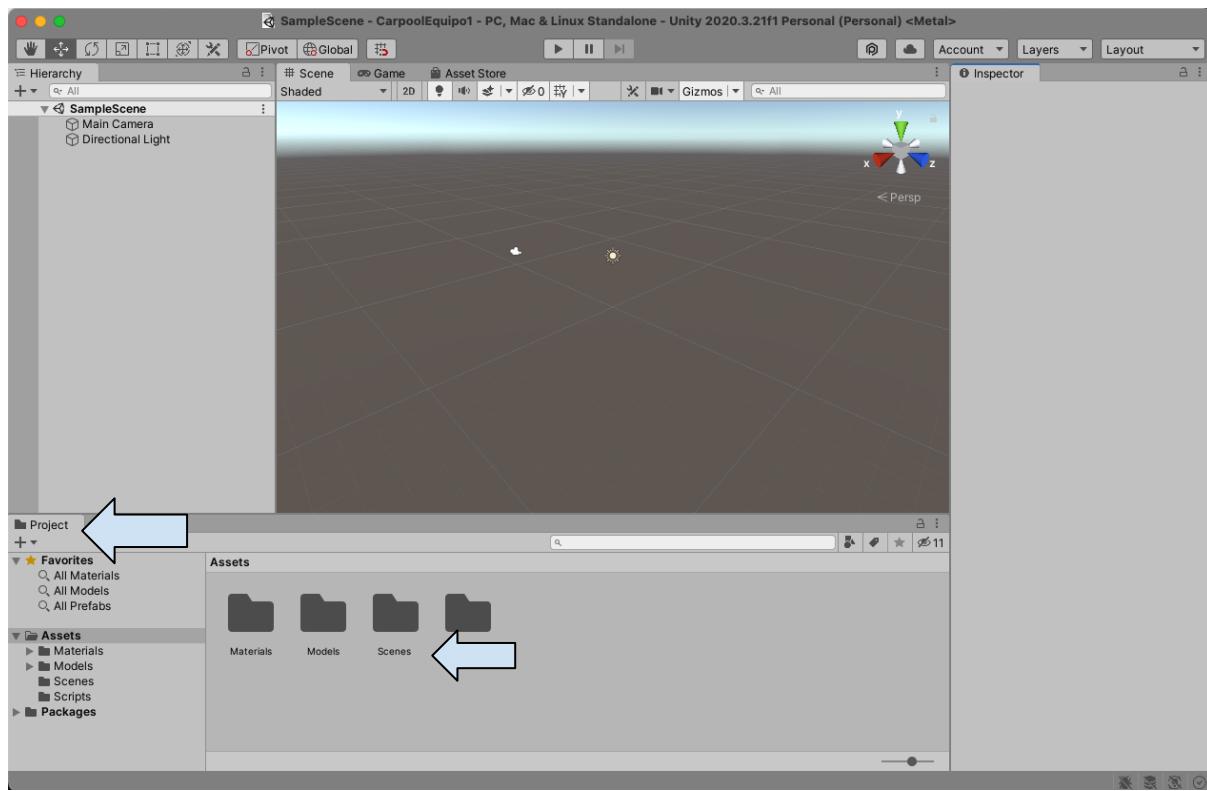


Imagen 21 Interfaz de Unity

Dirigirse a la pestaña de “Project”, ubicada en la parte inferior de Unity y presionar la carpeta de Scenes.

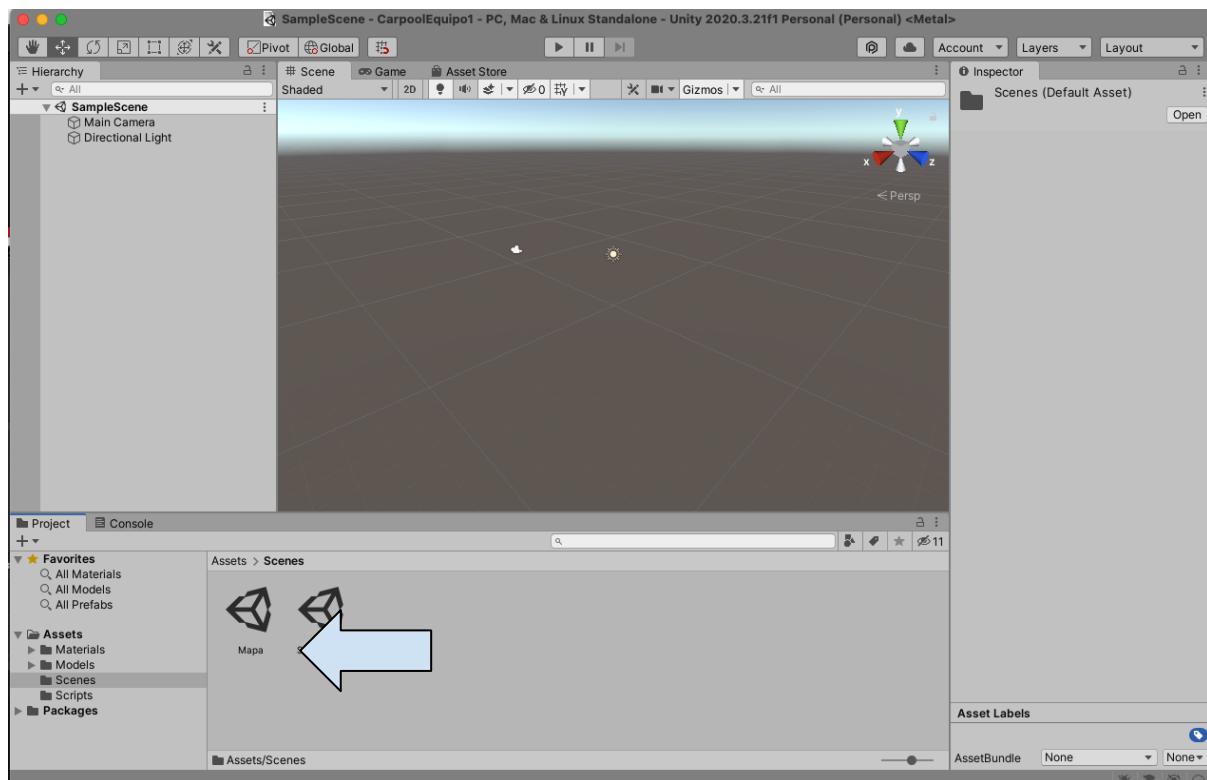


Imagen 22 Carpeta Scenes

Abrir la escena llamada “Mapa”, automáticamente en la parte de “Scene” (recuadro central) se desplegará una ciudad.



Imagen 23 Despliegue de la escena precargada

En un navegador coloque la siguiente URL: <https://smaa01653126.us-south.cf.appdomain.cloud/prueba2>

Esto se hace con el fin de reiniciar la simulación desde la creación de los pasajeros y los coches.

Para iniciar la simulación, presionar play en la parte superior de la pantalla.

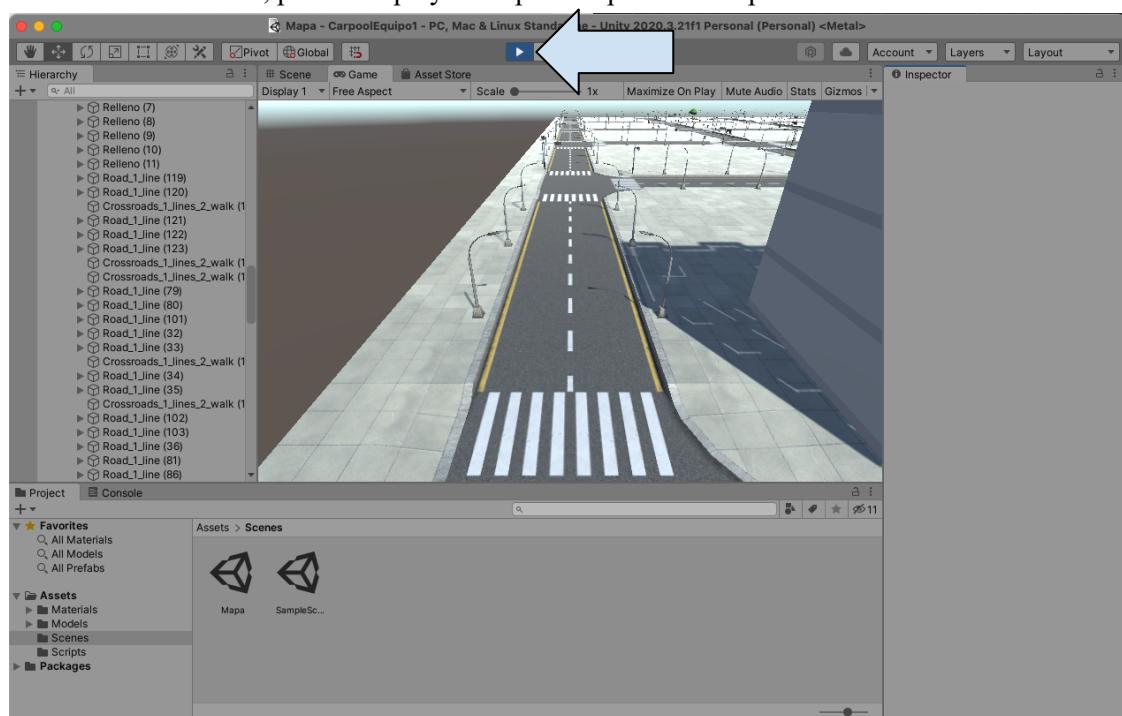


Imagen 24 Iniciar la simulación

Puede desplazarse alrededor de la escena utilizando su mouse.

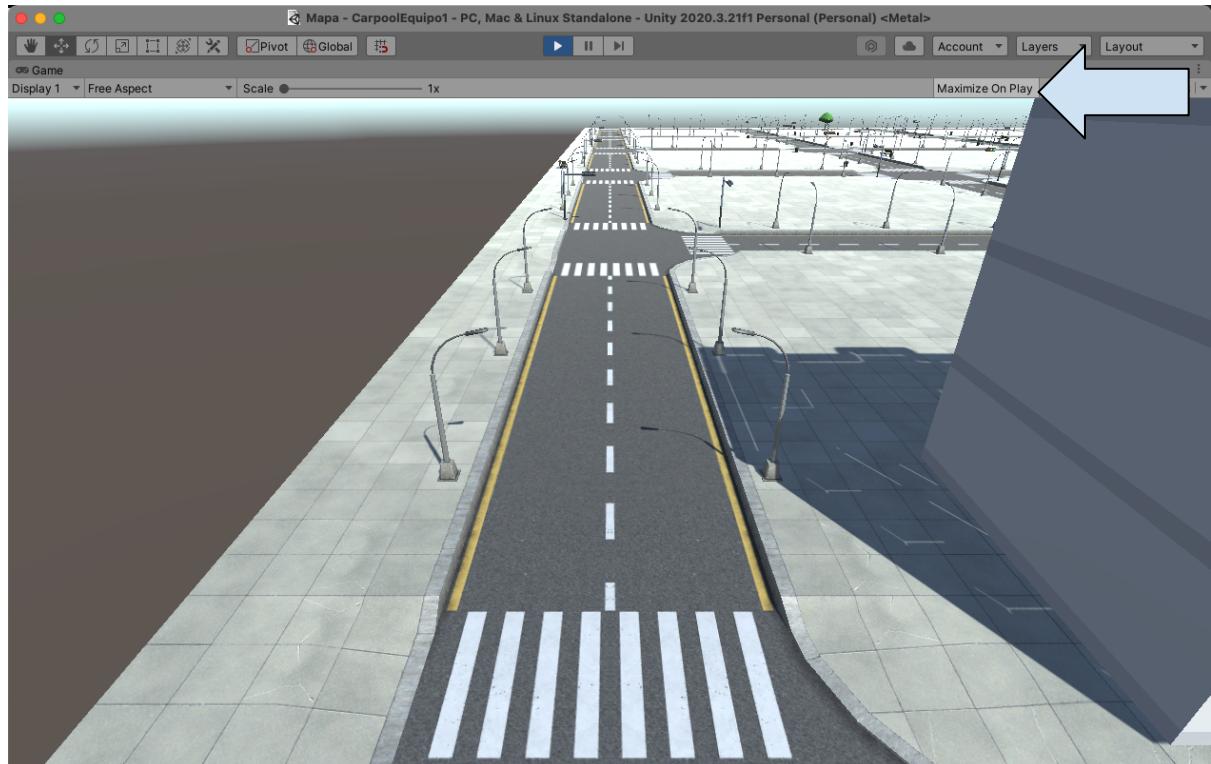


Imagen 25 Pantalla grande en la simulación

Si desea visualizarla como pantalla completa, presione el botón “Maximize On Play”, tal y como se muestra en la imagen 25.

En el modelo propuesto se puede visualizar el modelo de un automóvil realizado por nosotros, desde la parte de modelado, mapeo UV y el uso de texturas. El cual se muestra a continuación:

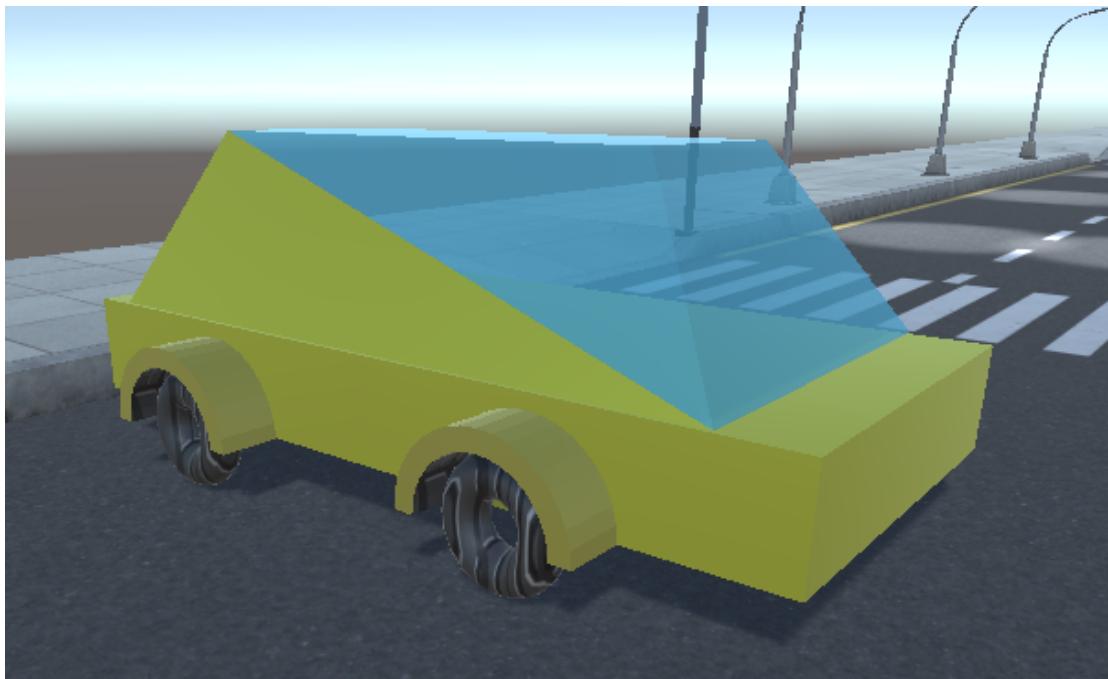


Imagen 26 Modelo 3D del coche

4.2 Análisis de solución desarrollada

La congestión de vehículos en las ciudades, particularmente en la Ciudad de México, ha llegado a un punto extremo. Es una situación que vivimos todos en el día a día. El reto planteaba cuatro posibles rumbos de investigación para la mitigación, y en nuestro equipo escogimos diseñar e implementar un modelo de Carpooling.

Era evidente que teníamos que utilizar alguna estructura de datos y algoritmo que permitieran realizar un match eficiente entre pasajero y vehículo. La primera dificultad fue abstraer el mapa de una ciudad a una estructura computacional. Inicialmente, se propuso la utilización de un grafo. Las intersecciones serían los nodos y las calles los vértices, que tendrían dirección, y peso, para representar la distancia. Sin embargo, pronto nos dimos cuenta de que habrían ciertas dificultades.

Para el caso de uso de la simulación, no es suficiente con encontrar rutas óptimas y determinar qué pasajeros deben ir con qué vehículo. Además, es necesario conservar el estado del sistema considerando steps de una unidad de distancia constante. Es decir, los carros deben avanzar cierto número de casillas o unidades en cada step del modelo, y eso debe representarse en Unity. Esto es difícil de abstraer en un grafo, puesto que el vehículo solo puede estar en un nodo, que es una intersección. La misma situación se presenta con los pasajeros, que pueden aparecer en cualquier posición siempre que sea una acera. Al utilizar grafos, teníamos que encontrar una manera de posicionar lógicamente un pasajero en una intersección, pero físicamente aplicar un offset para que pudiera estar en distintas posiciones en la acera. Llegó un punto en el que se acumularon demasiadas complicaciones, y fue evidente que el modelo inicial no era adecuado.

Después de aprender más sobre Mesa y de tomar cierto tiempo para reflexionar, pensé que un enfoque basado en Grid sería la mejor opción tanto por la compatibilidad con Mesa, puesto que ya no tendríamos que mantener dos estructuras lógicas, como con el modelo de Carpooling, puesto que no se limitan las posiciones de los agentes únicamente a los nodos. Compartí esta nueva propuesta con mi equipo y, después de hacerle algunos cambios, optamos por cambiar el modelo. En retrospectiva, esta parte del trabajo nos absorbió bastante tiempo, aunque considero que fue bien invertido puesto que nos facilitó la implementación.

Después de definir la representación de la información, procedimos a modelar el sistema multiagentes. De igual manera sufrió modificaciones con el tiempo, pero al final definimos seis agentes distintos: carro, pavimento, pasajero, banqueta, semáforo e intersección. Inicialmente podría parecer que tenemos agentes redundantes o inútiles, como en el caso del pavimento, el semáforo o la banqueta. Sin embargo, fue crucial conocer que pueden haber agentes pasivos que no tienen un objetivo, pero que generan un impacto en el sistema. De esta manera, fue posible introducir los constraints de posición y direcciones al Grid de mesa e integrar todo en una sola matriz en lugar de tener una matriz para los agentes y otra para las restricciones del mapa.

También hay dos agentes activos, la intersección y el pasajero, que no tienen metas muy complejas pero sí tienen un objetivo definido. En el primer caso el objetivo es controlar a los semáforos dependiendo de las direcciones involucradas y asegurar que nunca hay dos de ellos en verde al mismo tiempo. En el caso del pasajero, su objetivo es llegar a su destino escogiendo el vehículo que le proporciona la menor distancia. Notemos que el pasajero no realiza el cálculo de la ruta, sino que recibe una notificación por parte del carro.

Finalmente, tenemos un agente cognitivo, que busca realizar cálculos y optimización: los vehículos. En cada turno, un carro tiene diversas opciones y tareas a realizar. En primer lugar, debe determinar el pasajero que sigue esperando que le queda más cerca, en caso de que aún exista uno. Después, debe calcular la ruta para dejar a cada uno de los pasajeros que lleva a bordo. Con esta información, decide si es óptimo ir a recoger al pasajero más cercano o ir a dejar a algún pasajero a bordo. En caso de que ya no tenga objetivo a recoger o dejar, puede calcular la ruta a su propio punto de destino y llegar hasta el.

Algo muy interesante es que se manejan dos tiempos a la vez. En primer plano, el carro debe ser capaz escanear todo el mapa y definir su próximo objetivo, de manera atemporal. En segundo plano, el carro debe ir ejecutando un step a la vez, es decir, un desplazamiento de una sola unidad en lugar del recorrido completo, además de considerar las restricciones del ambiente como el color de los semáforos o la posición de los otros carros. Si el semáforo está en rojo o si hay un vehículo en la siguiente posición, entonces no debe hacer un movimiento.

Como se puede inferir, las variables que utilizamos para la definición de la ruta y para el trayecto *per se* de los vehículos fueron de naturalezas distintas. En primer lugar, y la más

importante, está la distancia entre dos puntos. Sin embargo, no es la distancia Euclíadiana, sino algo parecido a la distancia de Manhattan, solo que tomando en cuenta las restricciones de las direcciones de las calles, puesto que son de un solo sentido. Además, se consideraron otras variables ambientales, como ya se mencionó la posición de los otros vehículos y los semáforos. El resultado de utilizar estas variables es que siempre se escoge el destino más cercano, por lo que utilizamos un enfoque Greedy. Sin embargo, es importante recordar que este tipo de algoritmos no siempre generan soluciones óptimas, como se analizará a continuación.

Nuestro modelo tiene varias ventajas importantes en comparación con un modelo sin carpooling. Realizamos una serie de experimentos en las que colocamos primero a 240 vehículos, cada uno con su destino final, y la contraparte de 200 pasajeros con tan solo 40 vehículos. Se mantiene constante el número de personas desplazándose, pero se reduce la cantidad de tránsito. Esto es evidente en la simulación. A simple vista se puede apreciar que cuando hay 240 pasajeros el mapa luce muy saturado, con muchos cuellos de botella y tiempos de espera por parte de los vehículos. Por otro lado, cuando solo se utilizaron 40 vehículos, el movimiento se ve más fluido y no se crean las filas que se generan en el modelo anterior.

Para tener resultados numéricos, obtuvimos la cantidad total de movimientos carro que realizan los agentes. Se encontró que en el modelo de carpooling se realizan 11% menos movimientos que en el modelo normal. A pesar de que podría parecer una reducción minúscula, consideremos que eso significa 11% menos consumo de gasolina y emisión de gases del efecto invernadero. Esto sería muy valioso en una ciudad tan contaminada como la CDMX. Cabe mencionar que el experimento se realizó cinco veces, puesto que la generación de las posiciones tanto de los carros y de los pasajeros es aleatoria.

Sin embargo, debido a nuestro modelo multiagentes, la solución generada tiene áreas de oportunidad. Además de medir la cantidad de movimientos carro, también se determinó el tiempo que tomaba a todas las personas llegar a su destino. Resulta que en el modelo de carpooling tardan el doble de tiempo en comparación con la simulación en la que cada uno tiene su carro. Esto no es del todo incorrecto, cuando se viaja en UberPool, se tiene que esperar primero a que llegue el vehículo, y también se pierde tiempo en lo que se realizan pequeñas desviaciones para dejar o recoger a los demás. Sin embargo, en estos casos las rutas no se desvían tanto para cada pasajero. En nuestro caso, es posible que se recoja a un pasajero que va a cambiar completamente el rumbo tanto para el conductor como para el resto de los pasajeros a bordo.

Para resolver este problema se debe cambiar el enfoque, en lugar de utilizar un algoritmo Greedy se debe utilizar un con cierto grado de dinamismo. No solo se debe considerar la distancia de llegada, sino también qué tanto modifica la ruta. Esto agrega mucha complejidad computacional puesto que se deben escanear todos los pasajeros en el mapa, no solo el más cercano, puesto que un pasajero en el centro puede ser más óptimo que el pasajero más cercano del momento. En el caso de Uber, cuentan con sistemas de información geográfica

altamente optimizados que pueden encontrar rutas de una manera más eficiente, solo que esto requiere de mucha infraestructura algorítmica y de recursos. En el futuro sería interesante mejorar el algoritmo.

Finalmente, algo que fue muy importante para la simulación fue la representación gráfica. El modelo verdaderamente cobra vida cuanto se puede visualizar en tercera dimensión. Inicialmente se utilizó la visualización que ofrece el servidor integrado de Mesa, pero no era posible obtener tantos insights como fue posible cuando se conectó el modelo con Unity. Algo muy importante es que el mapa de Unity coincide perfectamente con el de Mesa. Se generó una representación que permitiera apreciar las calles y los vehículos fácilmente, sin saturación. La visualización de los pasajeros también fue crucial, puesto que se muestran en rojo cuando están esperando, y en verde cuando han llegado a su destino final, lo que permite apreciar el avance de la simulación.

4.3 Reflexión proceso de aprendizaje

Este bloque, a pesar de su corta duración, ha sido uno de mis favoritos hasta el momento. La problemática planteada es relevante en el mundo real. Además, es un problema difícil de resolver desde la modelación del sistema, la implementación de los multiagentes, el diseño e implementación de la visualización y la comunicación entre la vista y el modelo.

Definitivamente aprendí muchísimo en el bloque, gracias a las clases de los profesores y a investigación individual y con mi equipo. Como esperaba al inicio del bloque, yo me dediqué un poco más a la parte de sistemas multiagentes que gráficas computacionales, puesto que va más orientado a mis metas a futuro. Sin embargo, gracias a las actividades y al reto, también pude aprender muchas cosas de gráficas. Algo que fue constante en la implementación del modelo fue la búsqueda de documentación e investigación. Esto me gustó mucho porque cada vez obtenía un mayor entendimiento de Mesa. Creo que utilizar este paquete fue un gran acierto, pues nos obligó a pensar en términos de agentes y ambiente. Al inicio me frustraba mucho y hubiera preferido hacerlo sin el paquete, diseñando yo mismo las clases y las mecánicas desde cero. Sin embargo, después de entender cómo se visualizan los agentes, los steps y el ambiente pude apreciar las virtudes de Mesa y cómo permite repartir la responsabilidad de manera equitativa entre los agentes.

La conformación del equipo fue crucial para el buen resultado del proyecto. Desde un inicio tuvimos varias sesiones de zoom, de larga duración, en las que discutíamos las distintas maneras de modelar el problema. En general, me siento muy satisfecho y orgulloso del proyecto que realizamos. Logramos generar un algoritmo de carpooling no muy complejo (como otros basados en KNN u otros métodos de machine learning), pero que es funcional y proporciona cierto grado de optimización. Lo que más me gustó fue cómo se integró todo en el producto final. Al conectarlo con la vista se generó un resultado muy llamativo y que verdaderamente hace lucir el modelo.

Anexos

Repositorio:

Gráficas computacionales: <https://github.com/E1-CarpoolProject/carpool-graphics>

Multiagentes: <https://github.com/E1-CarpoolProject/carpool-multiagents>

Plan de trabajo:

<https://docs.google.com/spreadsheets/d/1VUu2x0GdWkeFjR5Wqz8ogRXiVcVxM9z49T1Q9Nsx73w/edit?usp=sharing>

Referencias

EsmartCity. (2021). Movilidad Urbana. ESMARTCITY. Recuperado el 9 de noviembre de 2021, de <https://www.esmartcity.es/movilidad-urbana>

Instituto Nacional de Estadística y Geografía (INEGI). (2021). Parque vehicular. Inegi.org.mx. Recuperado el 9 de noviembre de 2021, de <https://www.inegi.org.mx/temas/vehiculos/>

Muñoz, G. (2015). El automóvil y el progreso moral. Información. Recuperado el 9 de noviembre de 2021, de <https://www.informacion.es/opinion/2015/06/15/automovil-progreso-moral-6393485.html>

Shaheen, S., Cohen, A., & Bayen, A. (2018). The Benefits of Carpooling. *UC Berkeley: Transportation Sustainability Research Center*. <http://dx.doi.org/10.7922/G2DZ06GF> Recuperado de <https://escholarship.org/uc/item/7jx6z631>