Modelación de sistemas multiagentes con gráficas computacionales

TC2008B.523

Sergio Ruiz Loza

David Christopher Balderas Silva

Reto Movilidad Urbana

Hazel Astrid Ángeles Quevedo - A01275792

José Ernesto Gómez Aguilar - A01658889

Karen Rocío Macías Ávila - A01657935

Carlos Eduardo Córdoba Hilton - A01658948

1 de diciembre de 2021

Índice

Reto

Descripción

El reto consiste en la elaboración de una propuesta que brinde una solución al problema de la movilidad urbana en México, enfocándonos en la reducción de la congestión vehicular simulando de manera gráfica el tráfico de la ciudad con la representación de un sistema multi agentes.

El problema se aborda utilizando una de las siguientes estrategias.

- Controlar y asignar los espacios de estacionamiento disponible en una zona de la ciudad, evitando así que los autos estén dando vueltas para encontrar estacionamiento.
- Compartir tu vehículo con otras personas. Aumentando la ocupación de los vehículos, reduciría el número de vehículos en las calles.
- Tomar las rutas menos congestionadas. Quizás no más las cortas, pero las rutas con menos tráfico. Más movilidad, menos consumo, menos contaminación.
- Que permita a los semáforos coordinar sus tiempos y, así, reducir la congestión de un cruce. O, quizás, indicar en qué momento un vehículo va a cruzar una intersección y que de esta forma, el semáforo puede determinar el momento y duración de la luz verde.

Agentes involucrados

Agentes

- **Automóviles:** se relacionan entre ellos mismos debido a que transitan juntos sobre las calles, son agentes basados en una meta por que tratan de llegar a un destino.
- **Semáforos:** Son agentes basados en utilidad que se relacionarán con los automóviles permitiendo avanzar o detenerse.
- **Ambiente:** Las calles que cruzan los automóviles son el ambiente de manera secuencial.

Diagrama de clase

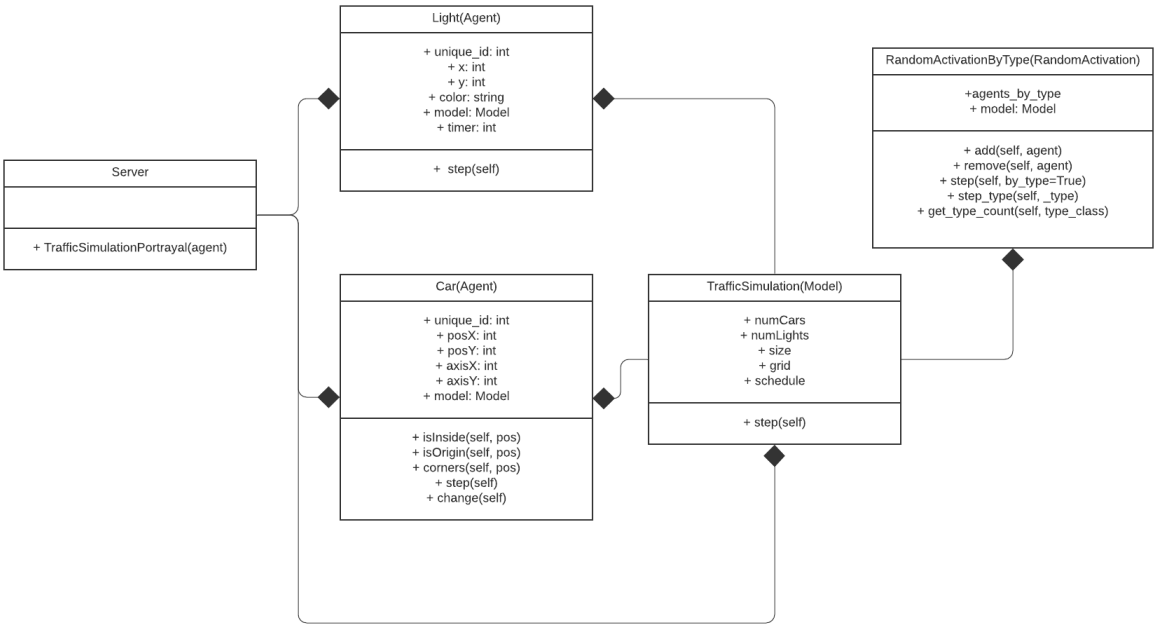

Figura 1. Diagrama de clases[1]

[1] Se puede visualizar a través del siguiente enlace: https://bit.ly/30FbeTA

Diagrama de protocolos

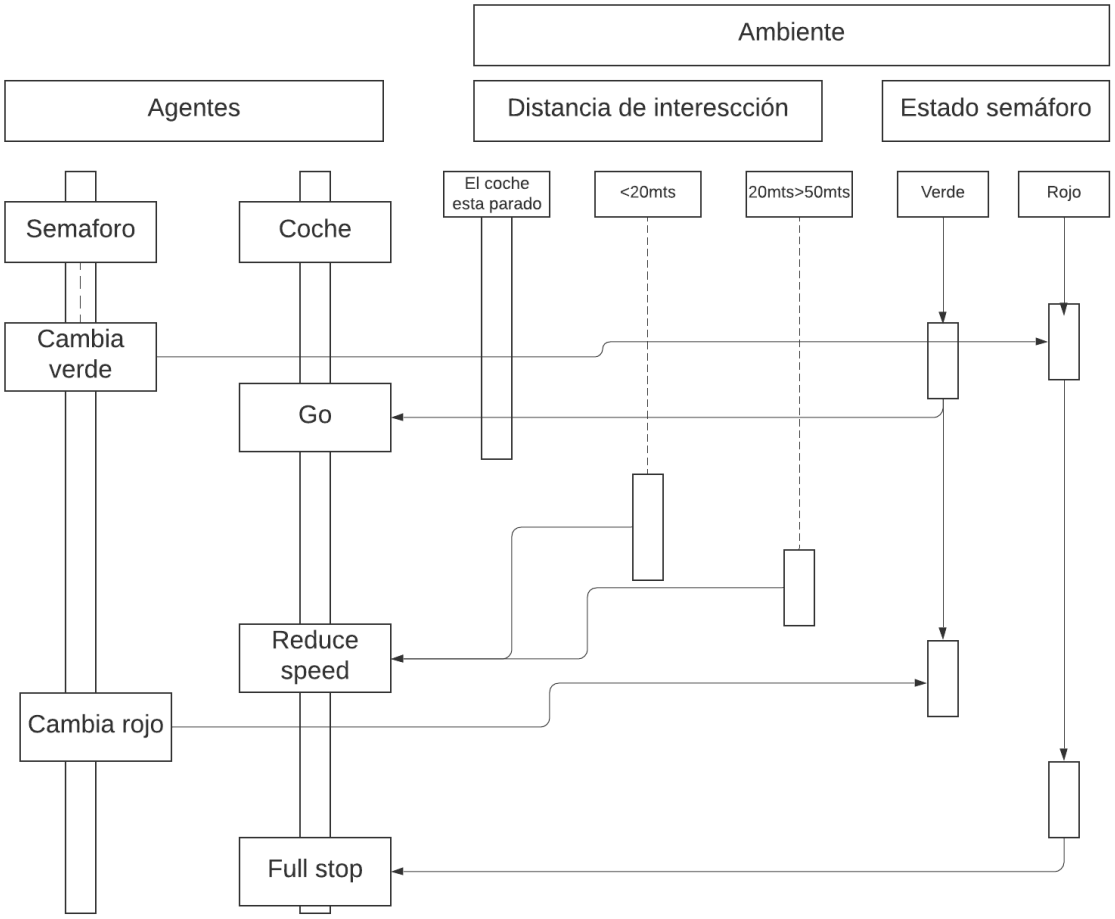

Figura 2. Diagrama de protocolos[2]

[2] Se puede visualizar a través del siguiente enlace: https://bit.ly/30FbeTA

Modelos de Unity

El proyecto debe de ser capaz de correr en el software de Unity, donde para verse estéticamente mejor, se tomaron referencias de varios modelos de la tienda de assets de Unity.
Entre estos modelos están los automóviles que se utilizaran para transitar en nuestro reto:

Figura 3. Vista de los automóviles

Entre ellos hay un modelo que se creó desde cero armado en su totalidad por los integrantes del equipo:

Figura 4. Automóvil modelado por el equipo

Ya que conseguimos todos los modelos de automóviles necesarios, ahora necesitamos conseguir un ambiente en donde estos se muevan, una calle. O en este caso un cruce:

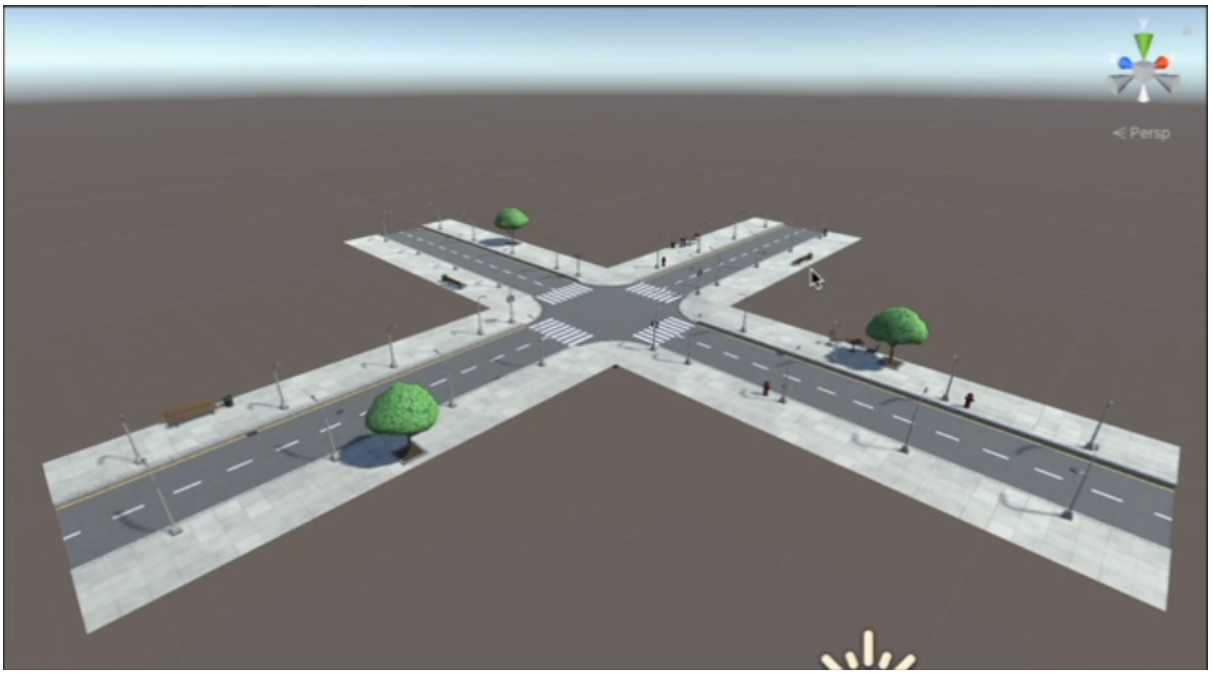

Figura 5. Cruce del camino

El cruce viene con los semáforos a utilizar, por lo que ya tenemos ambos agentes y el ambiente que conforman la escena de nuestro proyecto.

Implementación del Código

Liga del Equipo Github: https://github.com/KarenMacias/TC2008B.523_Equipo7

A continuación se muestra el código necesario para la representación de agentes por medio de python.
En la figura 6 podemos observar el algoritmo del servidor que contiene los datos de visualización de los agentes en el grid por lo que se tienen ahí los colores, tamaños y se llama al servidor.

```
from mesa.visualization.ModularVisualization import ModularServer
from mesa.visualization.modules import CanvasGrid
from mesa.visualization.UserParam import UserSettableParameter

from agents import Light
from agents import Car
from model import TrafficSimulation

def TrafficSimulationPortrayal(agent):
    if agent is None:
        return

    portrayal = { }

    if type(agent) is Light:
        portrayal["Layer"] = 1
        portrayal["Color"] = agent.color
        portrayal["Shape"] = "rect"
        portrayal["Filled"] = "true"
        portrayal["w"] = 1
        portrayal["h"] = 1
    elif type(agent) is Car:
        portrayal["Layer"] = 1
        portrayal["Color"] = "blue"
        portrayal["Shape"] = "rect"
        portrayal["Filled"] = "true"
        portrayal["w"] = 1
        portrayal["h"] = 1

    return portrayal

canvas_element = CanvasGrid(TrafficSimulationPortrayal, 20, 20, 600, 600)

model_params = { }

server = ModularServer(TrafficSimulation, [canvas_element], "Traffic Simulation", model_params)
server.port = 8521
```

Figura 6. Código de servidor

La figura 7 contiene el algoritmo de schedule que brinda las funciones para activar los agentes, eliminarlos y su funcionamiento durante los steps, este código no se modificó venía de referencia en el código base.

```python
from collections import defaultdict

from mesa.time import RandomActivation

class RandomActivationByType(RandomActivation):
    def __init__(self, model):
        super().__init__(model)
        self.agents_by_type = defaultdict(dict)

    def add(self, agent):
        self._agents[agent.unique_id] = agent
        agent_class = type(agent)
        self.agents_by_type[agent_class][agent.unique_id] = agent

    def remove(self, agent):
        del self._agents[agent.unique_id]
        agent_class = type(agent)
        del self.agents_by_type[agent_class][agent.unique_id]

    def step(self, by_type=True):
        if by_type:
            for agent_class in self.agents_by_type:
                self.step_type(agent_class)
            self.steps += 1
            self.time += 1
        else:
            super().step()

    def step_type(self, _type):
        agent_keys = list(self.agents_by_type[_type].keys())
        self.model.random.shuffle(agent_keys)
        for agent_key in agent_keys:
            self.agents_by_type[_type][agent_key].step()

    def get_type_count(self, type_class):
        return len(self.agents_by_type[type_class].values())
```

Figura 7. Código de schedule

En la figura 8 tenemos el algoritmo del modelo en el que inicializamos todos los agentes de tipo semáforo y coche con las posiciones seleccionadas y posteriormente en la función step recopilamos la posición de cada elemento.

```python
from agents import Light
from agents import Car
from schedule import RandomActivationByType

class TrafficSimulation(Model):
    def __init__(self, cars = 4, lights = 4, size = 20):
        super().__init__()
        self.numCars = cars
        self.numLights = lights
        self.size = size
        self.grid = MultiGrid(size, size, False)
        self.schedule = RandomActivationByType(self)

        #Obtenemos las posiciones iniciales de donde surgirán los elementos con su respectiva orientación, es decir, hacia que lugar caminará
        initialPos = [[19, 10], [10, 0]]
        axis = [[-1,0],[0,1]]

        #Creamos los automóviles
        for i in range(self.numCars):
            rand = self.random.randint(0, 1)
            x1 = initialPos[rand][0]
            y1 = initialPos[rand][1]
            x2 = axis[rand][0]
            y2 = axis[rand][1]
            c = Car(i, (x1, y1), (x2, y2), self)
            self.grid.place_agent(c, (x1, y1))
            self.schedule.add(c)

        #Obtenemos las posiciones de los semáforos y les determinamos un color
        lightPos = [[10,11],[9,10],[10,9],[11,10]]
        colors = ["red", "green", "red", "green"]
        counter = self.numCars

        #Creamos los semáforos
        for i in range(self.numLights):
            x = lightPos[i][0]
            y = lightPos[i][1]
            l = Light(counter, (x, y), colors[i], self)
            counter += 1
            self.grid.place_agent(l, (x, y))
            self.schedule.add(l)

    #Para cada paso que de se van a guardar las posiciones de los agentes en un array y estos datos se darán en unity
    def step(self):
        self.schedule.step()
        ps = []
        for i in range(self.numCars):
            xy = self.schedule.agents[i].pos
            p = [xy[0], xy[1], 0]
            ps.append(p)
        return ps
```

Figura 8. Código de model

En las figuras 9, 10 y 11 tenemos el código para los agentes de tipo semáforo y carro, en estos declaramos sus variables importantes como posición, orientación, color, etcétera y en su función de step generamos un algoritmo optimizado que ayuda a liberar el tráfico. Para el agente coche tenemos una función que moviliza el agente a través del canvas y al acercarse a un semáforo se comporta dependiendo el color manteniéndolo en la misma posición o cambiandola.

```python
import random
from mesa import Agent

class Light(Agent):
    def __init__(self, unique_id, pos, color, model):
        super().__init__(unique_id, model)
        self.x = pos[0]
        self.y = pos[1]
        self.color = color
        self.timer = 3

    def step(self):
        self.timer = self.timer - 1
        if self.timer == 0:
            if(self.color == "red"):
                self.timer = 3
                self.color = "green"
            else:
                self.timer = 3
                self.color = "red"
```

Figura 9. Código del agente semáforo

```python
class Car(Agent):
    def __init__(self, unique_id, pos, axis, model):
        super().__init__(unique_id, model)
        self.posX = pos[0]
        self.posY = pos[1]
        self.axisX = axis[0]
        self.axisY = axis[1]

    #Para saber si un coche se encuentra dentro del espacio definido
    def isInside(self, pos):
        if((pos[0] >= 0 and pos[0] < self.model.size) and (pos[1] >= 0 and pos[1] < self.model.size)):
            return True
        else:
            return False

    #Para saber si se encuentra en las posicciones de inicio
    def isOrigin(self, pos):
        if(pos == (10,0) or pos == (19,10)):
            return True
        else:
            return False

    def step(self):
        self.change()
```

Figura 10. Código del agente coche parte 1

```
def change(self):
    #Obtenemos un nuevo espacio para que se muevan los agentes
    newSpace = (self.pos[0] + self.axisX, self.pos[1] + self.axisY)

    initialPos = [[19, 10], [10, 0]]
    #Cuando llegue al límite del canvas
    if self.isInside(newSpace) == False:
        #Reinicia la posición del agente en una del origen
        if(newSpace[0] == 10):
            newSpace = (initialPos[1][0], initialPos[1][1])
        elif(newSpace[1] == 10):
            newSpace = (initialPos[0][0], initialPos[0][1])

    #Obtenemos los agentes cercanos al nuevo espacio
    agents = self.model.grid.get_cell_list_contents(newSpace)
    #Obtenemos arrays dependiendo el tipo de objeto
    lights = [obj for obj in agents if isinstance(obj, Light)]
    cars = [obj for obj in agents if isinstance(obj, Car)]

    #Si no se detectaron agentes y el coche actual está dentro del canvas
    if len(cars) == 0 and len(lights) == 0 and self.isInside(newSpace) == True:
        #Si los coches están en el origen
        if self.isOrigin(self.pos):
            rand = self.random.randint(0,1)
            #Si obtenemos 1
            if rand == 1:
                #Dependiendo la posición del coche camina hacia adelante en su eje
                if(self.pos == (10,0)):
                   moveDir = [0,1,10,1]
                elif(self.pos == (19,10)):
                    moveDir = [-1,0,18,10]
                if self.axisX != moveDir[0] and self.axisY != moveDir[1]:
                    self.axisX = moveDir[0]
                    self.axisY = moveDir[1]
                    self.model.grid.move_agent(self, (moveDir[2], moveDir[3]))
            #Si no que avance dependiendo la posición original
            else:
                self.model.grid.move_agent(self, newSpace)
        #Si no está en el origen que se mueva a la posición original
        else:
            self.model.grid.move_agent(self, newSpace)
    #Si encontramos semáforos
    elif len(lights) != 0:
        #Si es rojo se mantiene en esa posición
        if lights[0].color == "red":
            return
        #Si es verde
        else:
            #Avanza dos espacios cada agente, primero uno hacia adelante y después a una posición random
            self.model.grid.move_agent(self, newSpace)
            moveDir = [[0,1,10,9],[-1,0,10,10]]
            rand = self.random.randint(0, 1)
            if self.axisX != moveDir[rand][0] and self.axisY != moveDir[rand][1]:
                self.axisX = moveDir[rand][0]
                self.axisY = moveDir[rand][1]
                self.model.grid.move_agent(self, (moveDir[rand][2], moveDir[rand][3]))
    #Si tenemos más coches no pasa nada
    elif len(cars) != 0:
        return
```

Figura 11. Código del agente coche parte 2

Proceso de instalación y ejecución de la simulación

Proceso IBM Cloud

1. Crear una cuenta de IBM Cloud
2. Entrar a https://github.com/IBM-Cloud/ibm-cloud-cli-release/releases/ y descargar el IBM Cloud Cli segun el sistema operativo
3. Instalar
4. Reiniciar la máquina
5. Al volver a encender la máquina abrir la terminal

6. Ingresar ibmcloud login
7. Posteriormente identificarse con el correo y contraseña de IBM Cloud
8. Seleccionar la región us-south
9. Ejecutar el comando ibmcloud cf install
10. Ejecutar el comando ibmcloud api https://api.ng.bluemix.net
11. Debe marcar el proceso como correcto
12. Ejecutar el comando ibmcloud target –cf
13. El comando anterior debe mostrar Punto Final de API, Región, Usuario, Cuenta, Grupo de Recursos, Punto Final de API de CF, Organización y Espacio. Verificar que los datos sean correctos
14. Ir al sitio IBM Cloud: https://cloud.ibm.com/
15. Del lado izquierdo del menú. Elegir Cloud Foundry
16. Una vez dentro de Cloud Foundry, recorrer hacia abajo hasta encontrar: Tiempos de Ejecuciones y seleccionar Python
17. Realizar la configuracion del recurso
18. Escribir el nombre de la App, crear el dominio y aceptar
19. Una vez que termina, debe indicar: “En ejecución”
20. Volver a la terminal
21. Ejecutar el comando ibmcloud.exe cf apps
22. Descargar el archivo server-agentes.zip y descomprimirlo
23. Editar el archivo manifest.yml
24. Cambiar name por tu servidor creado en la nube
25. Abrir nuevamente la terminal y entrar a la carpeta donde se encuentra el código
26. Ejecutar el comando ibmcloud cf push
27. Subir todos los archivos excepto lo que se indique en .cfignore
28. Verificar que el resultado diga: Estado en Ejecución
29. Ejecutar el comando: ibmcloud.exe cf logs nombre_app –recent
30. Corregir de acuerdo a los errores encontrados en el Log.
31. Una vez corregidos los errores. Volver a ejecutar el comando push descrito anteriormente hasta que se ejecute correctamente.
32. Entrar a la liga del dominio creado y la aplicación se ejecutará de manera correcta

Proceso Anaconda

1. Nos dirigimos a la pagina https://www.anaconda.com/

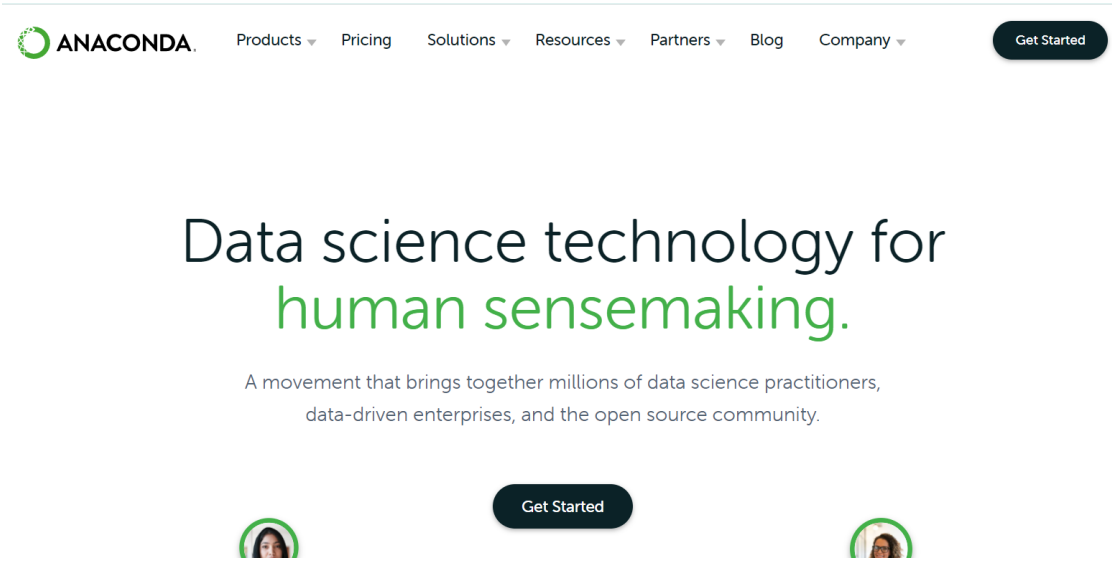

Figura 12. Página de Anaconda

2. Comenzamos y nos vamos a la parte de descarga

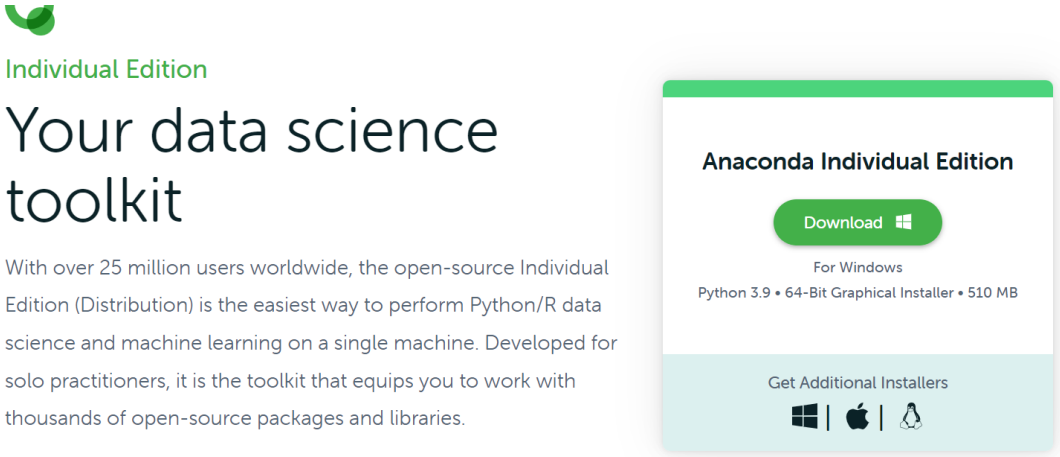

Figura 13. Descarga de Anaconda

3. Descargamos según el sistema operativo de nuestra máquina.
4. Guardamos y ejecutamos

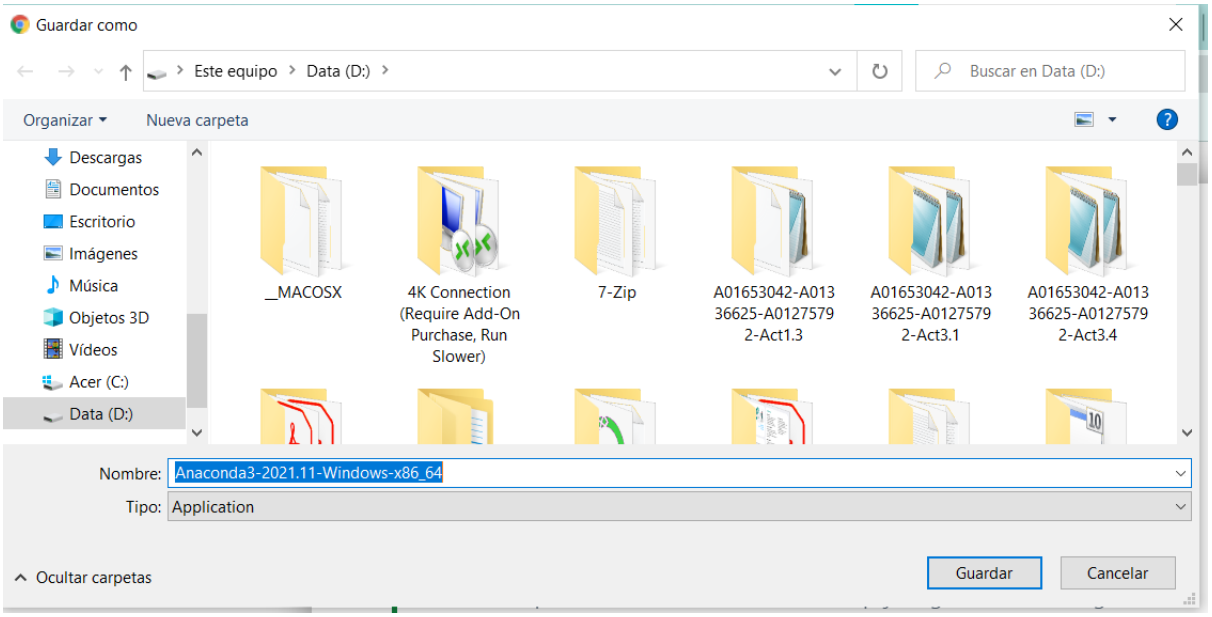

Figura 14. Guardar el ejecutable

5. Escogemos el espacio en donde se instalará

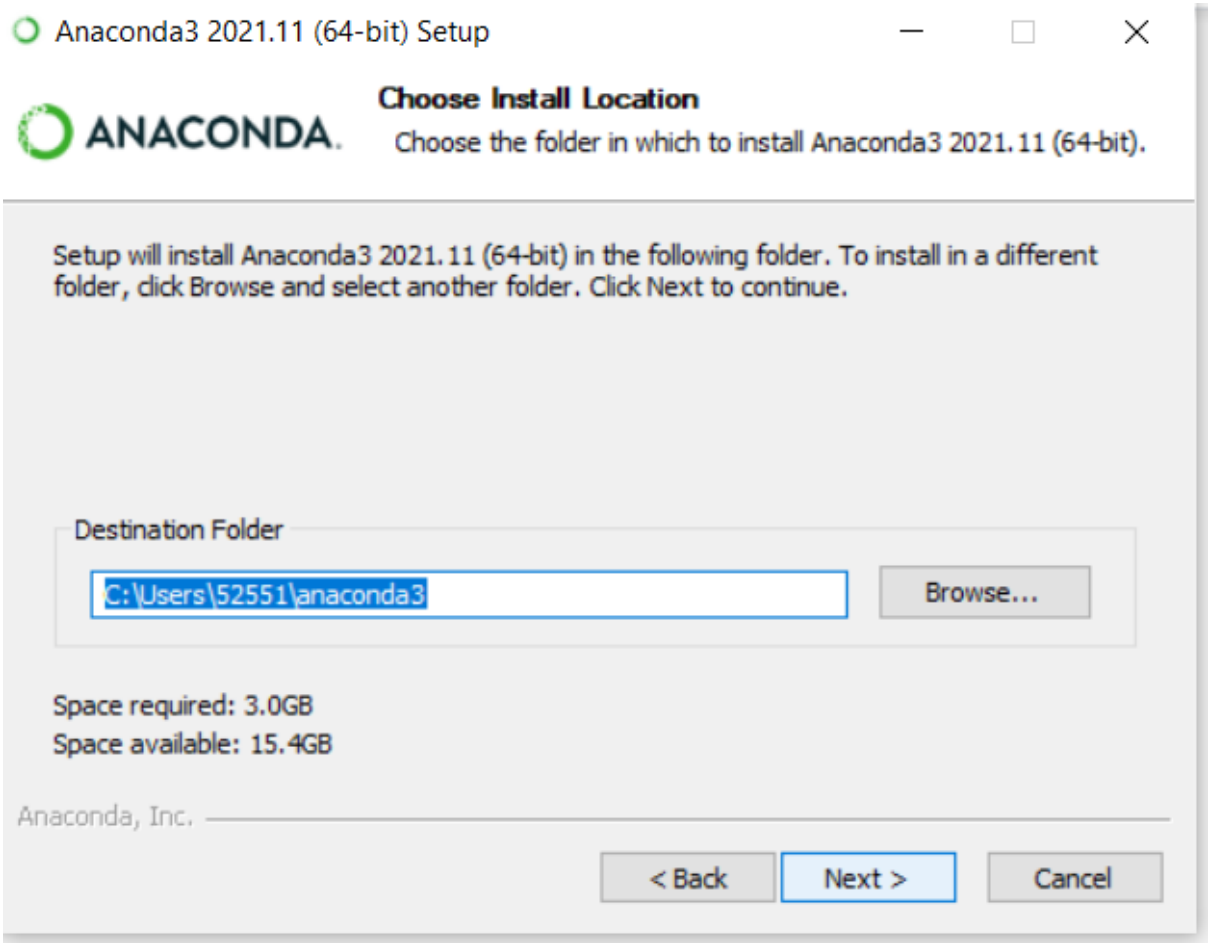

Figura 15. Instalar en la locación

6. Guardamos y continuamos y ya podemos utilizar el ambiente de anaconda

Proceso Mesa

1. Entrar en el ambiente de anaconda

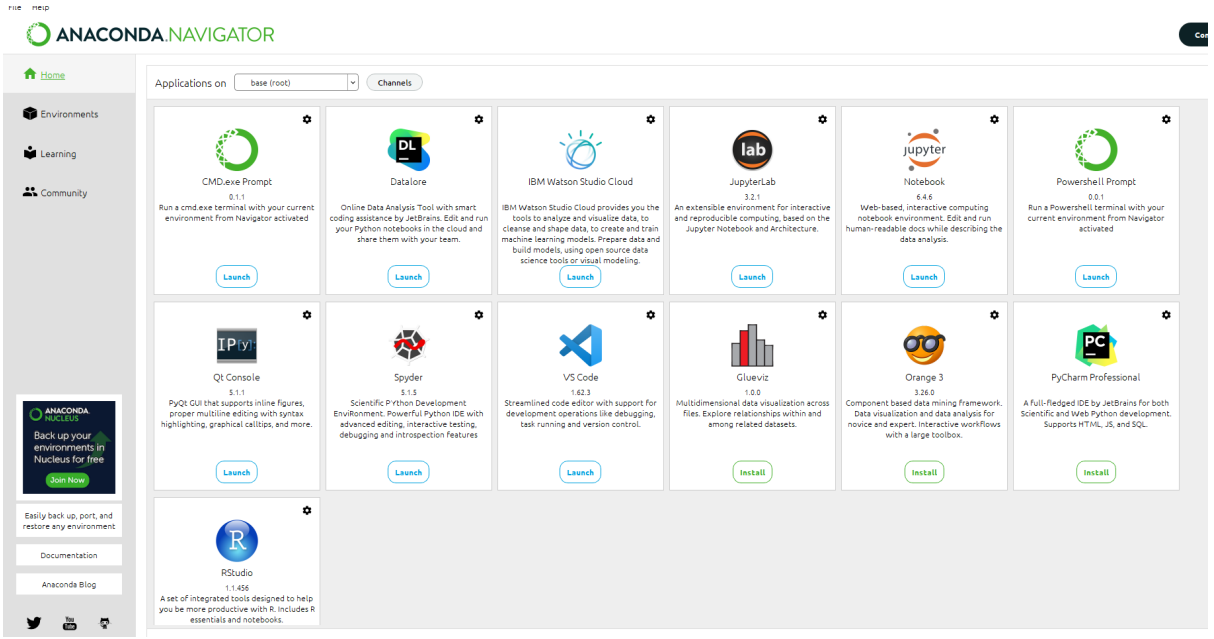

Figura 16. Navegador Anaconda

2. Crear un nuevo ambiente para nuestro proyecto

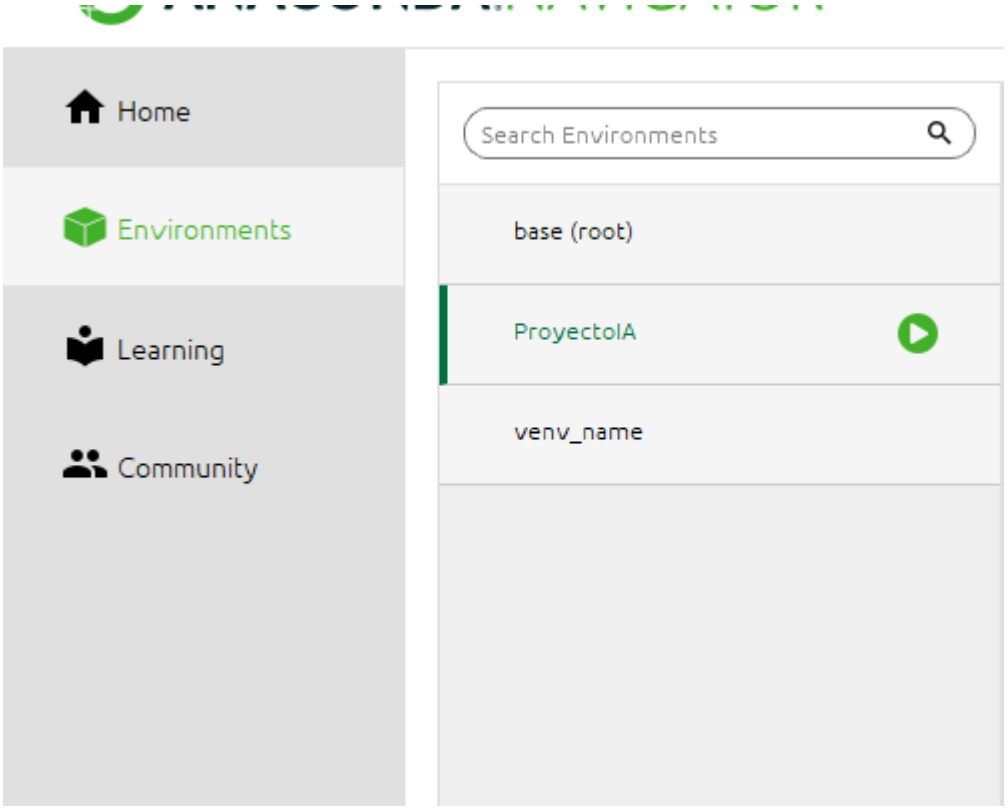

Figura 17. Ambiente Anaconda

3. Abrir la terminal dentro del ambiente que estamos trabajando

4. Ejecutar el comando conda install -c conda-forge mesa

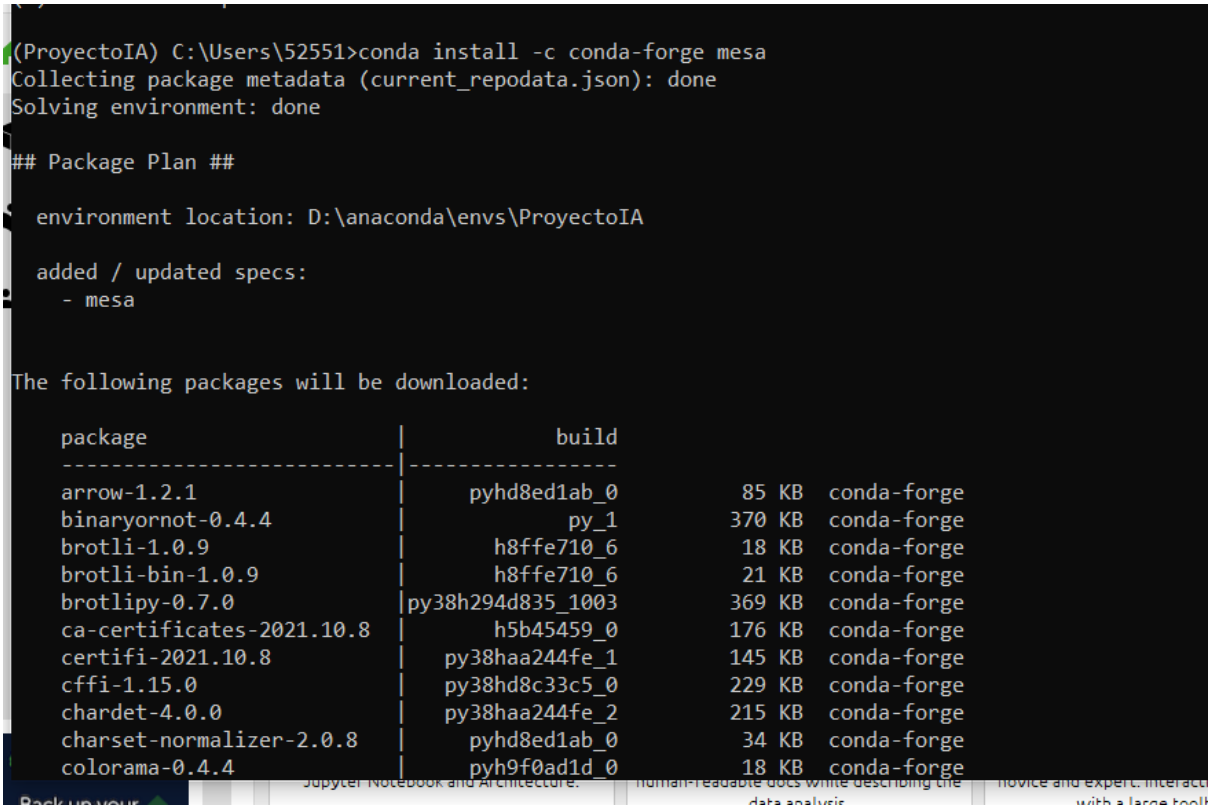

Figura 18. Instalar comando anaconda

5. Esperamos a que extraiga todos los paquetes necesarios

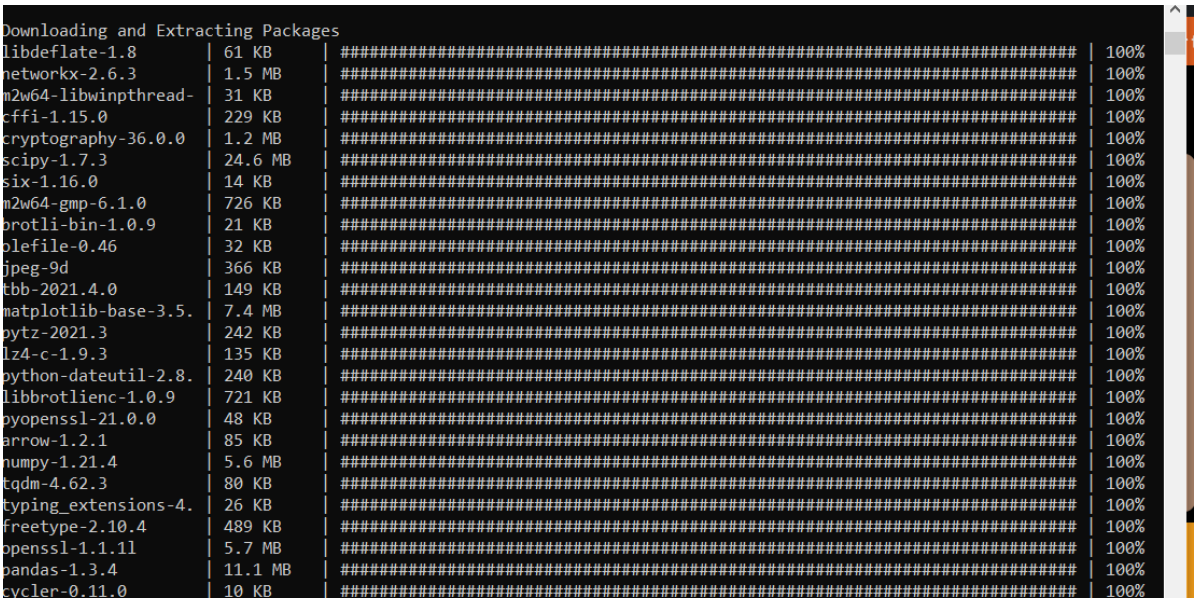

Figura 19. Descarga de paquete

6. Ya se puede utilizar mesa dentro del proyecto.

Proceso Unity

1. Entrar a la pagina https://unity.com/es/download
2. Crear una cuenta y descargar según el sistema operativo
3. Iniciamos la instalación y si es necesario reiniciar la máquina
4. Abrir Unity Hub
5. Dirigirse a install

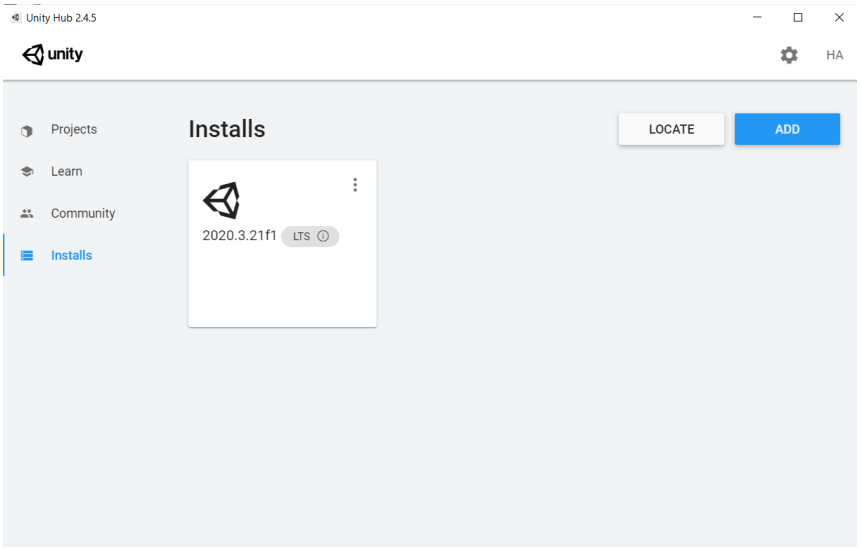

Figura 20. Instalador unity

6. Presionar Add y escoger la versión que deseamos

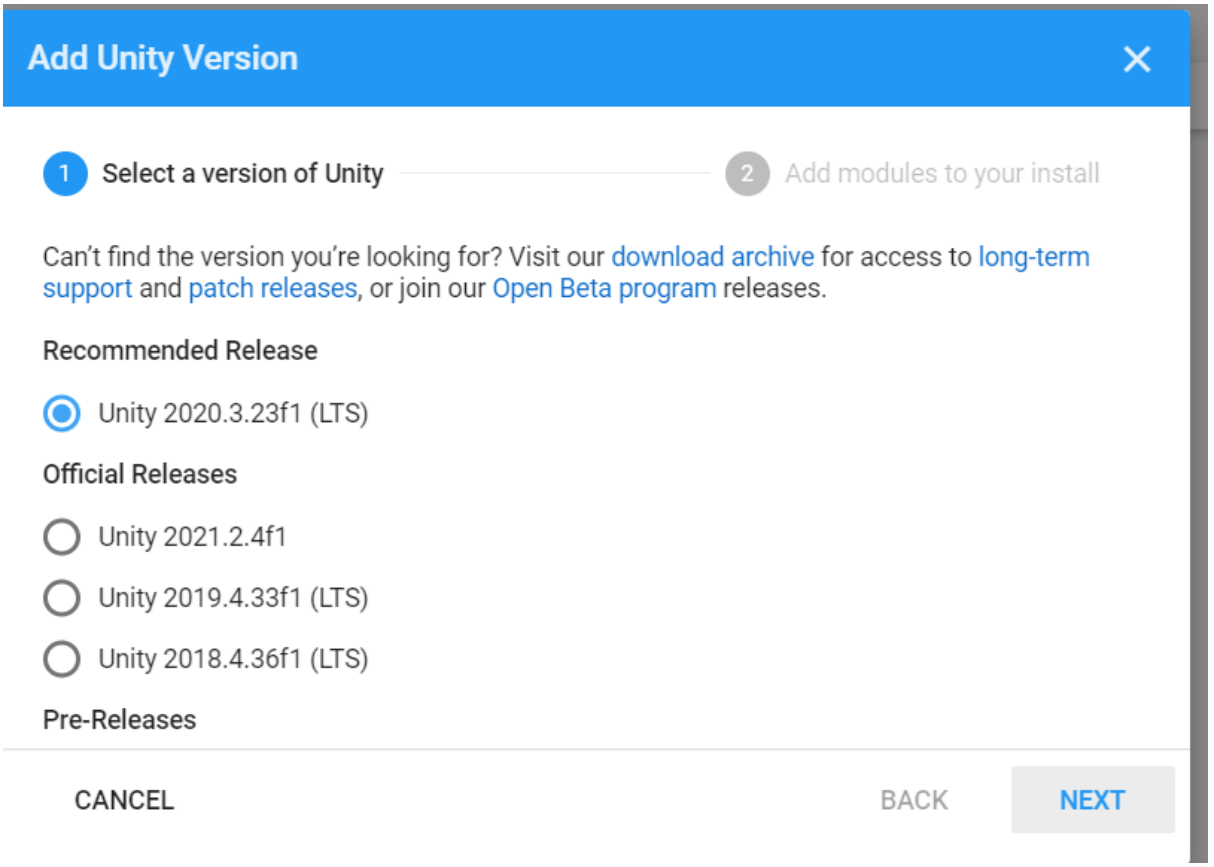

Figura 21. Versión de unity

7. Presionar NEXT y continuar la instalación
8. Una vez instalado presionar en Projects y en NEW creamos uno nuevo

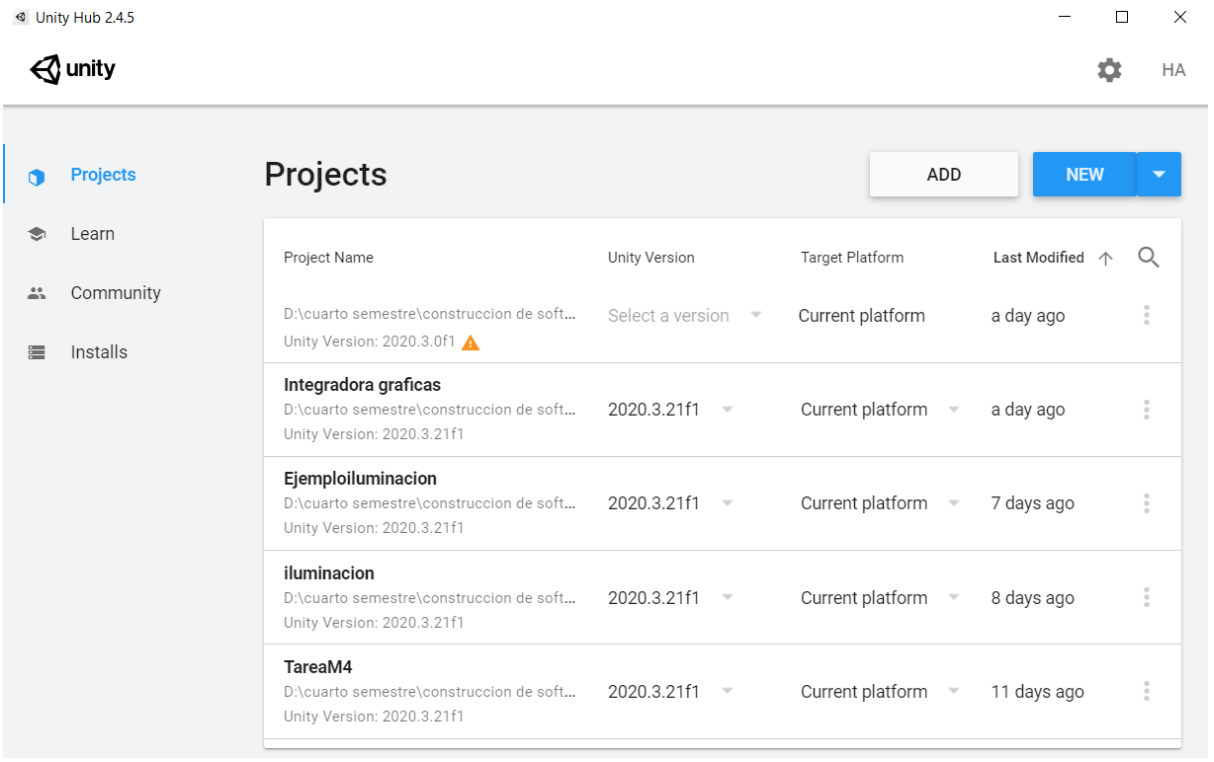

Figura 22. Descarga de Anaconda

9. Elegir el nombre, la ubicación, y tipo de proyecto según lo que queremos desarrollar y creamos el proyecto

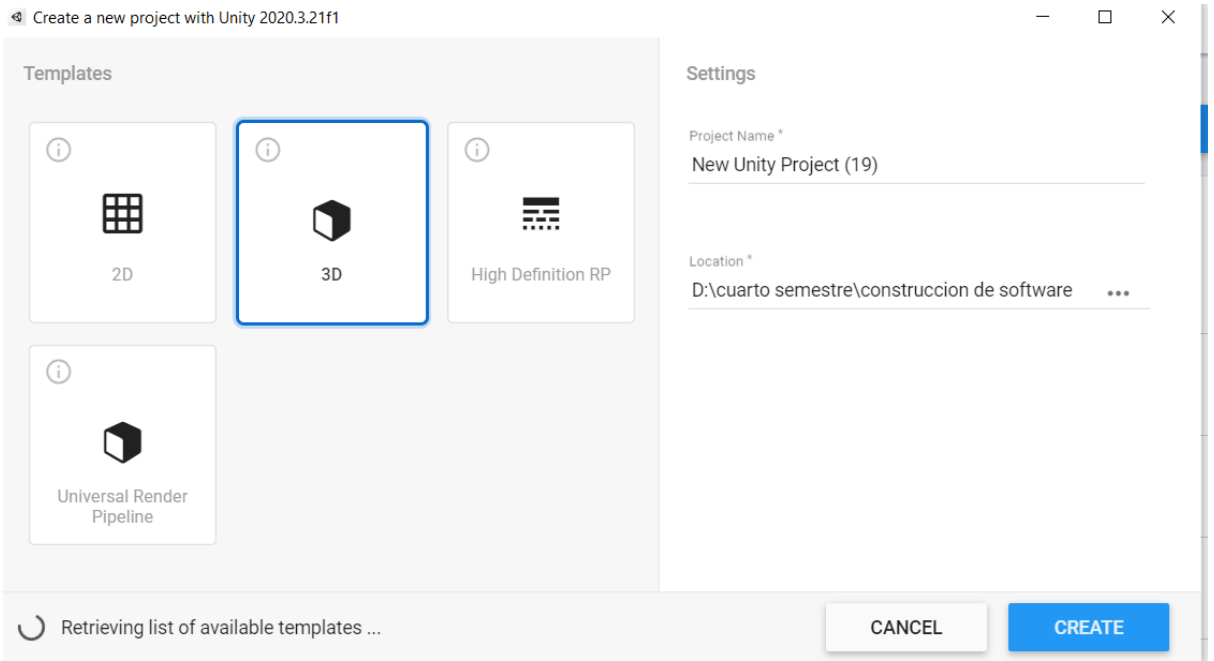

Figura 23. Descarga de Anaconda

10. Comenzar a crear el proyecto

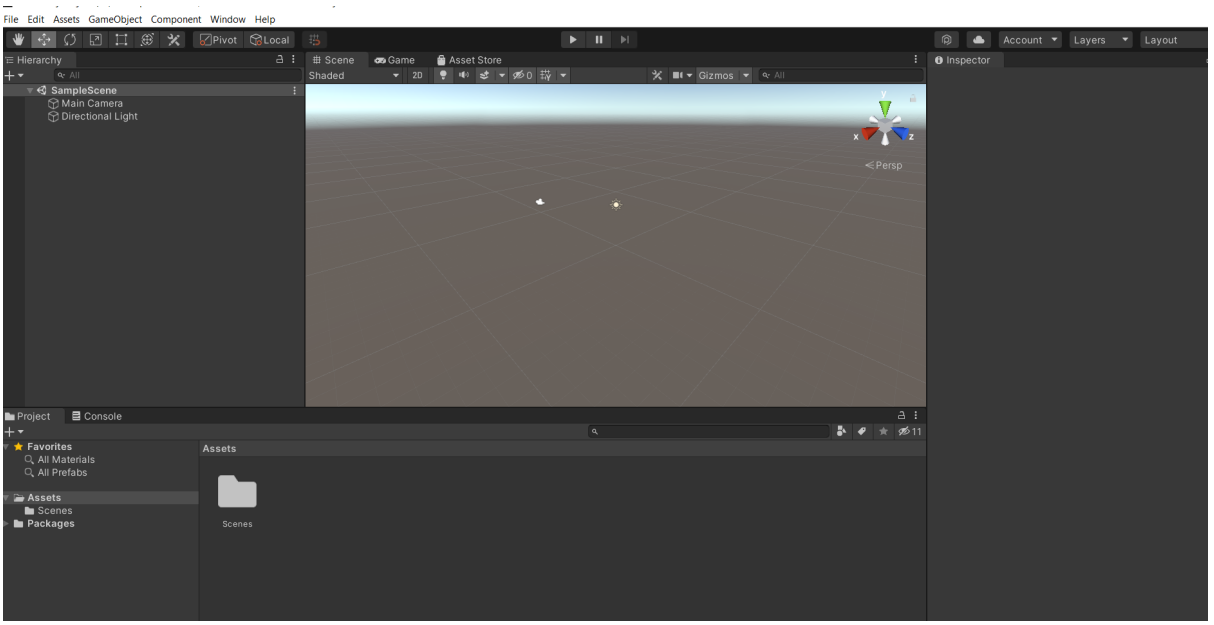

Figura 24. Vista de inicio de unity

11. Finalmente ya se puede abrir el archivo de la simulación

Proceso ejecución

1. Encontrar el archivo de unity
2. Abrir el archivo en editor
3. Ejecutar la simulación

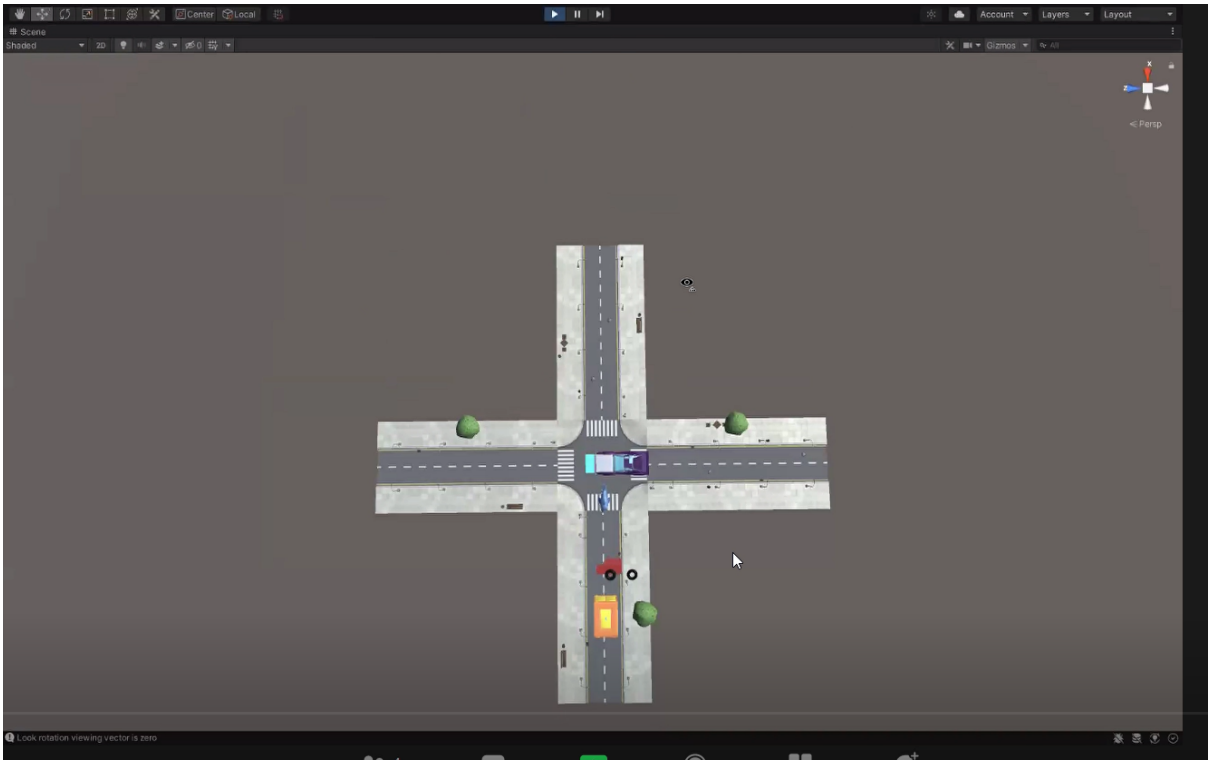

Figura 25. Simulación de la intersección

4. Visualizar los movimientos de los agentes

Figura 26. Movimientos de los agentes

Análisis Individual de la Solución Desarrollada

Karen: El modelo multiagentes se eligió debido a que cumple con todas las especificaciones del reto y representa una buena oportunidad para su solución, en este caso como consiste en simular tráfico lo que se realizó fue un cruce de semáforos que coincide con las calles de la CDMX en las que comúnmente tenemos de opción de seguir adelante o moverse hacia un extremo dependiendo el sentido de las calles para que así la simulación fuera más realista.

Para la simulación se tomaron en cuenta las variables del ambiente y los agentes con respecto a lo aprendido en clase. El equipo observó detalladamente las variables con las que se iban a trabajar y las dividimos correspondientemente dando como resultado que los agentes terminaron siendo los semáforos y automóviles mientras que el ambiente es la calle. La idea principal es que los agentes se comuniquen entre sí a través del ambiente para conseguir un flujo automovilístico más eficiente. La interacción se lleva a cabo principalmente entre los semáforos y

los carros ya que los semáforos mediante su color le dan la indicación a los carros de avanzar si la luz es verde, o detenerse si es roja. El ambiente es el que comunica la información entre los agentes, así la calle le comunica a los carros el color del semáforo y también las posiciones de otros carros para que estos no choquen.
Para el diseño gráfico primero se decidió buscar assets de calles que fueran fáciles de colocar para que las calles fueran acordes a nuestra intersección. Para los carros buscamos assets diferentes para tener variedad y que de preferencia tuvieran pocos polígonos, además de diferentes dimensiones para dar realismo. Además de esto también buscamos en ellos que el diseño se viera bien.
Nuestra simulación tiene ventajas y desventajas, pero hablando de las ventajas creo que la mayor es que la simulación nos da una solución funcional y eficiente para mejorar el tráfico vial evitando choques y respetando los semáforos, aunque como desventaja creo que está un poco incompleta pensando en un caso real pues aunque los agentes en la simulación siempre respetan el semáforo, en la vida real no todas las personas lo hacen y eso puede significar un caso fuera de los parámetros de la simulación. Para dar solución a esto podríamos hacer más carriles para repartir más el flujo de vehículos. Otra mejora importante para la simulación es hacer que los semáforos decidan a qué carriles darles la luz verde dependiendo de cuál tiene más carros sobre él, para que así se ponga luz roja a los carriles con menos carros y le demos prioridad a los que tienen un mayor flujo reduciendo el tráfico causado por semáforos.

Hazel: El modelo de multiagentes que se seleccionó fue elegido así debido a que con el problema que se plantea la interacción se ve completamente entre los automóviles y el semáforo no es de manera directa sino que se comunican mediante el ambiente y este hace la conexión con la cual logran interactuar, así mismo estos dos agentes se ven involucrados con el camino porque de este depende que los automóviles puedan avanzar o cambiar de camino y que los semáforos puedan cambiar su estado.

Las variables que se tomaron en cuenta para tomar esta decisión fue que realmente se lograra esa interacción de manera inteligente, otro factor importante fue la eficiencia que pueda llegar a tener, otra parte importante a considerar fue que se necesitaban posiciones de todos los agentes para que estos tuvieran conocimiento entre ellos especialmente con los semáforos verificar que se puede cambiar de color y las calles puedan contener los automóviles sin tener que salirse del camino. La interacción que se tuvo con las variables fue bastante acertada ya que en el

resultado de la simulación se puede ver cómo se realiza la interacción planteada, que los coches tiene su movilidad y que pueden llegar a su destino por que son agentes por meta, igual los semáforos cambian según lo indicado y para lo que están destinados.

El diseño gráfico fue seleccionado primeramente por la compatibilidad que pudiera tener con la versión de unity, de igual forma que en los coches si pudiera tener un mapeo UV que no fuera extremadamente complicado para poder colorear y finalmente se eligió esto para tener un buen diseño.

Las ventajas que encuentro durante la solución final es que se puede llegar a mejorar la movilidad urbana con un programa que pueda realizar cosas similares en los semáforos, también algo importante es que dentro de la solución se logró crear un sistema inteligente capaz de cambiar y avanzar según las condiciones que en las que se encuentre la sea con muchos autos o pocos siempre se podrá adaptar a esto. Las desventajas que se tienen dentro de la solución es que puede llegar a ser un poco pesado el programa para las computadoras por lo que se necesitan equipos con buena capacidad que logren soportar la simulación debida, otra desventaja que veo dentro de esto es que faltaría tomar en cuenta algo más de eficiencia. La modificación finalmente que haría es encontrar una manera más eficaz de poder controlar el tráfico y poder considerar los peatones que lleguen a cruzar dentro de los caminos ya que si aumentamos esto la situación del semáforo cambiara completamente y habría que buscar otros factores que puedan contribuir a una mejor solución haciendo así una simulación más eficiente y con temas importantes a considerar que puede ser implementado en la vida real tomando en cuenta cada uno de los factores.

Jose Ernesto: Para este trabajo se optó por seleccionar la intersección con semáforos para controlar el tráfico por diversas razones. Una de ellas es que decidimos que el control del tráfico con los semáforos era la solución más viable de aplicarse si se piensa en un escenario real, ya que implicaría generar un sistema automatizado que no dependiera de decisiones humanas durante su funcionamiento, de esta manera el tráfico sería controlado de la manera más eficiente todo el tiempo lo que significaría una gran reducción de embotellamientos viales.
Las variables utilizadas para esta intersección fueron los semáforos, automóviles y el camino, ya que estos serían los agentes que deben de interactuar entre sí para

compartir información entre ellos y en base a esto tomar decisiones para mejorar el tráfico vehicular. Para esta interacción tenemos al semáforo que según su color da la indicación a los automóviles de detenerse o continuar, para lo que se implementó un temporizador que define cada cuánto tiempo el semáforo cambiará de color. Cuando el automóvil se acerca a un semáforo, el camino le comunica su estado, es decir su color, haciendo que de esta manera el automóvil al conocer el ciclo del semáforo siga la orden de detenerse o continuar. Finalmente los automóviles también deben de identificar su cercanía con otros automóviles para evitar choques, para lo que el camino transmite las posiciones de los vehículos entre ellos así haciéndoles saber qué vehículos hay en la cercanía.
El diseño gráfico fue seleccionado en base a la selección de assets así como al realismo. Decidimos utilizar un asset con calles que permitieran armar la intersección seleccionada, además de que se optó por un diseño cercano a la realidad por lo que también se utilizaron distintos tipos de automóviles de diferentes tamaños.
Las ventajas encontradas en esta solución fueron principalmente el control de choques y el control de tráfico. Este trabajo muestra una solución a como debe de fluir el tráfico vehicular respetando los semáforos así como evitando choques, lo que también implica una reducción en la cantidad de embotellamientos.
A pesar de esto, la solución también muestra algunas desventajas y es que la simulación solo permite a los automóviles tomar decisiones pero no a los semáforos, lo que representa un área de oportunidad importante a mejorar para optimizar la simulación al permitir que el semáforo pudiera decidir en base a la cantidad de vehículos en cada carril, a qué carril darle la luz verde para reducir el congestionamiento.

Carlos: Nuestro equipo seleccionó la intersección con semáforos automáticos, esto principalmente se debió a que este modelo se utilizó mucho de ejemplo durante nuestras clases, por lo que con el tiempo se generaron ideas con respecto a este. Una vez eligiendo el modelo a trabajar hicimos recuento de las variables con las que teníamos que trabajar por lo que decidimos que los automóviles y los semáforos serían nuestros agentes, mientras que la calle sería el ambiente en el que estarían.
Se decidió usar una serie de modelos tipo “low poly” para facilitar la implementación de materiales y el modelado 3D al reto sin sacrificar la elaboración del código a realizar. Finalmente logramos presentar un producto que funciona correctamente, debido a que permite el paso vehicular manteniendo las calles libre de tráfico lo más que se puede evitando accidentes. Considero que se podría

optimizar el código de alguna manera analizando a lo mejor más a futuro los posibles congestionamientos o sacrificando el tráfico de uno de los lados para descongestionar otro.

Reflexión Individual acerca del proceso de aprendizaje

Karen: Considero que por la cantidad de temas que se revisaron y el reto en general era algo muy ambicioso para resolver en cinco semanas. Por parte del módulo de gráficas mi aprendizaje fue bueno ya que entendí la parte teórica de los problemas, sin embargo, no me siento apta para resolver ejercicios por mi cuenta. En cuanto al contenido del curso de este módulo me pareció muy interesante y de valor para mi vida profesional ya que la modelación con uso de unity es algo que puede utilizarse de diversas maneras y es bueno de saber para futuros trabajos. Puedo decir que cumplí varias de las expectativas que mencioné al principio del bloque ya que me voy con muchos conocimientos nuevos y curiosidad por seguir investigando de ellos en el futuro. Tal vez no pude comprender al 100% todos los tópicos vistos en la clase, pero estoy segura de que di mi mejor esfuerzo en cada uno de ellos.

Hazel: Durante las cinco semanas el proceso de aprendizaje que tuve fue bastante complicado por que al ser poco tiempo había demasiados temas por comprender y en ocasiones me costaba un poco lograr entender todo. En la parte de graficas el aprendizaje que tuve fue bastante bueno mucho más de lo que esperaba, entendí cosas que no me imaginaba y en un principio durante la parte teórica si me resulto un poco complicado lograr hacer que las cosas tuvieran sentido para mí, pero durante la parte práctica todo fue tomando su lugar y fue bastante entretenido con aprendizajes de mucho valor y aplicables en distintas áreas que puedo llegar a ocupar. Para la parte de inteligencia artificial de la misma cuenta si era un poco difícil entender las cosas y más cómo se podrían ocupar esos conocimientos para la implementación final del reto, algo importante a considerar es que tenía demasiado tiempo que no utilizaba Python por lo que si me costo trabajo retomar la sintaxis que utiliza, sin embargo con un poco de práctica lo pude retomar no como quisiera pero sí con un gran avance, además de que aprendí bastante cosas sobre algoritmos y uso el cual es fundamental en el mundo de la programación.

Jose Ernesto: El reto de estas 5 semanas me agarro desprevenido, ya que justo llegamos a el de un módulo igualmente complicado pero hecho con el doble de tiempo, a pesar de esto estaba emocionado por aprender acerca de la Inteligencia Artificial y cómo la utilizamos para nuestro reto. También he de decir que tenía

mucha seguridad en mí mismo al descubrir que utilizaríamos Unity una vez más ya que en módulos anteriores había tenido la oportunidad de aprender mucho de este software que además disfruté mucho utilizando. Mis expectativas al inicio eran bastante altas aunque no se cumplieron por completo pues aunque logramos crear una solución funcional y de calidad, me hubiera gustado haber implementado más optimizaciones a esta, aunque por mis conocimientos y el reducido tiempo que tuvimos para resolverlo esto no fue posible, sin embargo este era un aspecto que ya había considerado desde el inicio. A pesar de todo el estrés del trabajo y el cansancio, me llevo una buena experiencia pues disfruté mucho los temas y puedo decir que me voy con un dominio considerable de ellos, de todas formas espero poder seguir practicando con temas así y generar experiencia para llegar a tener un mejor dominio de mis temas favoritos de la carrera que son IA y videojuegos.

Carlos: Este módulo he de decir que se me paso extremadamente rápido, siento que había más área de oportunidad para ver más a fondo algunos temas, pero el tiempo lentamente se nos vino encima para ir trabajando en el reto final y a pesar de solo ser 5 semanas, la dificultad claramente no disminuye. A pesar de esto, puedo decir que he tenido una experiencia placentera al realizar este proyecto. No solo trabajamos con Unity que es un software que siempre me ha llamado mucho la atención. Sino que también tuvimos la oportunidad de modelar en 3D y aprender de inteligencia artificial! Dos temas que siempre me llamaron la atención pero nunca me había aventurado a investigar por mi cuenta. Se que al principio del módulo una de mis expectativas era ser más activo en torno a la codificación del reto, sin embargo termine tomando el camino del modelado en 3D, que a pesar de ser un poco diferente a lo que estamos acostumbrados a ver en la carrera, es algo que disfruto hacer y que me permitió ayudar a mi equipo. Esto sin duda también mejoró un poco el área de oportunidad más importante que era la de “Tener más confianza en mi trabajo” ya que con este equipo me sentí en confianza suficiente para enseñar mi trabajo con seguridad y recibir retroalimentación de ellos de ser necesario. Por lo que a pesar de haber sido un módulo corto y bastante difícil, lo considero bastante fructífero.