Avance 1

Leonardo Arturo Morales López	A01652673
Paola Fernández Gutiérrez Zamora	A01768087
Axel González Carreto	A01652775

Diagrama de clases para la solución

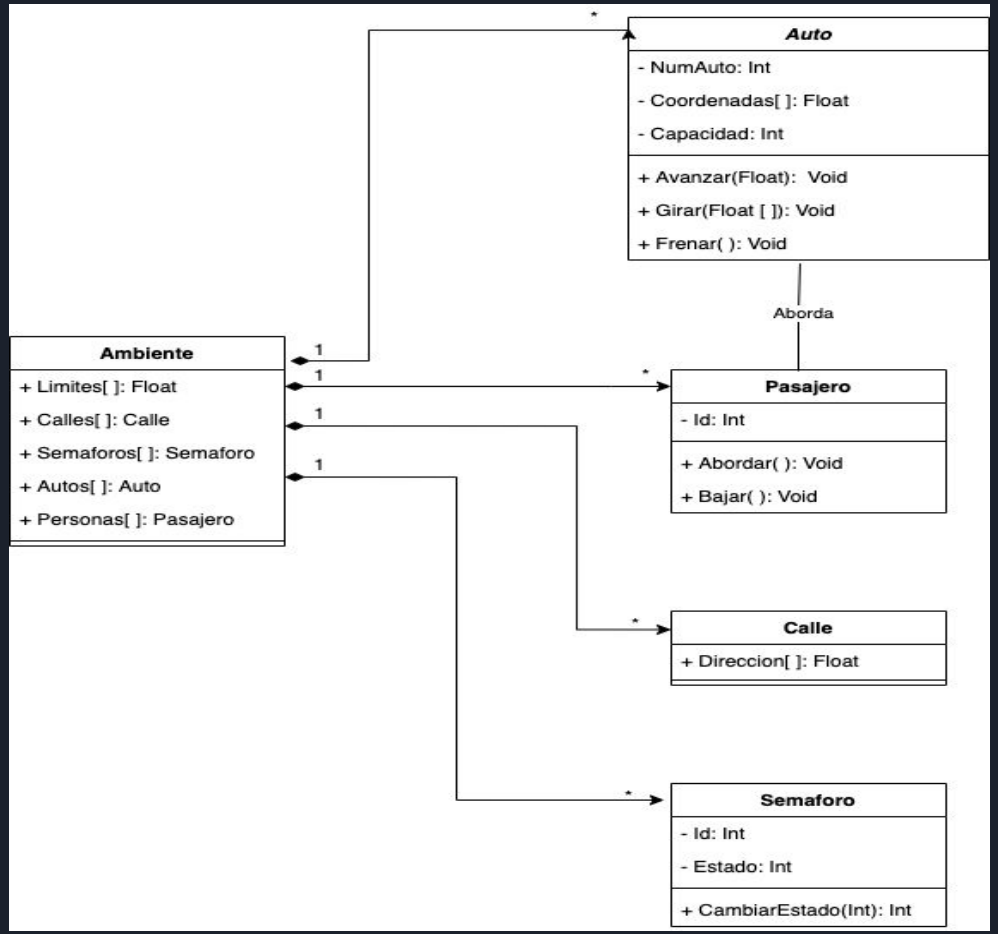
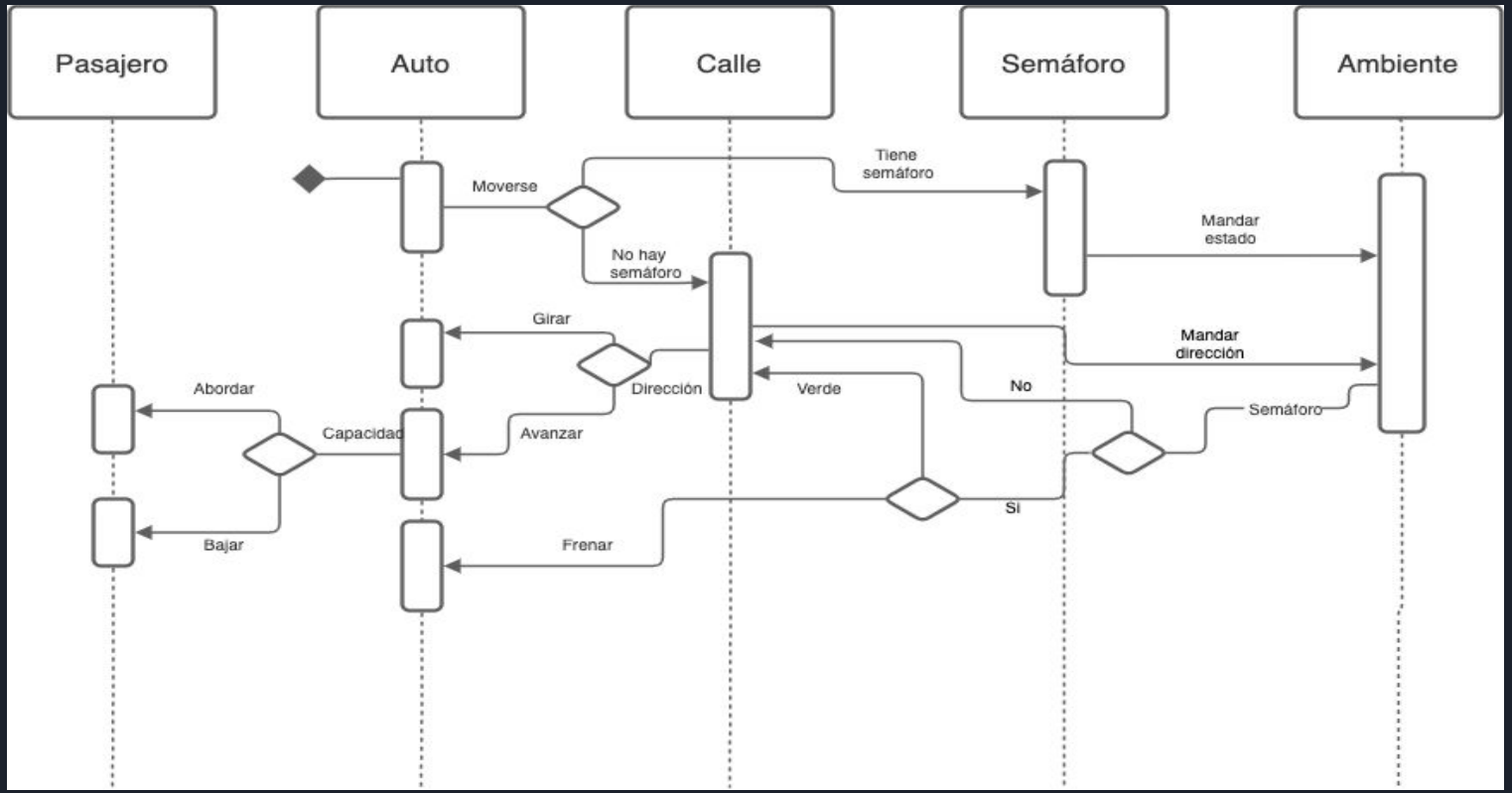


Diagrama de protocolos de interacción (actualizado)



Plan de trabajo avance 1

- Desarrollo de ambiente y creación de elementos en unity (40%)
- Código de sistema multiagentes en Python (40%)
- Corrección del diagrama de protocolos

https://trello.com/b/OgSEJTjZ/modelaci%C3%B3n

Creación ambiente y elementos en Unity

- Se logró crear el ambiente que se utilizará para la solución del reto

- Se modelaron intersecciones y las calles donde van a transitar los coches

- Se agregaron los primeros modelos de los agentes que se utilizarán para la solución del reto

- Se crearon automóviles hechos por el equipo y otros importados de paquetes de Unity

Código de sistema multiagentes en Python

```python
from mesa import Agent, Model

class CarAgent(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.availabeSeats = 4
        self.passengers = []

    def move(self):
        cellmates = self.model.grid.get_cell_list_contents([self.pos])
            if len(cellmates) > 1:
                other = cellmates[0]
                if type(other) == TrafficLightAgent:
                    if other.state == 0 or other.state == 1:
                        possible_steps = self.model.grid.get_neighborhood(
                            self.pos,
                            moore=True,
                            include_center=False)
                        new_position = self.random.choice(possible_steps)
                        self.model.grid.move_agent(self, new_position)

    def pickUp(self, cellmates):
        cellmates = self.model.grid.get_cell_list_contents([self.pos])
        if len(cellmates) > 1:
            other = cellmates[0]
            if type(other) == PersonAgent:
                other.in_car = True
                other.car = self.unique_id
                self.availabeSeats -= 1
                self.passengers.append(other.unique_id)


    def step(self):
        self.move()
        if self.availabeSeats > 0:
            self.pickUp
```

```python
class TrafficLightAgent(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.state = 0

    def changeState(self):
        if seltf.state = 0:
            self.state = 1
        elif self.state = 1:
            self.state = 2
        elif self.state = 2:
            self.state = 0

    def step(self):
        self.changeState()

class PersonAgent(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.in_car = False
        self.car = None
```

- Se creó un servidor local para la conexión de un código de python con Unity y se modeló un sistema multiagente como ejemplo
- Se encontraron diferentes ejemplos de modelado de una intersección con semáforos utilizando a los coches y semáforos como agentes y se comenzó la implementación (avance de la tarea 3)
- Se crearon los modelos para los coches, las personas y los semáforos

https://towardsdatascience.com/simulating-traffic-flow-in-python-ee1eab4dd20f

https://towardsdatascience.com/traffic-intersection-simulation-using-pygame-689d6bd7687a

Aprendizajes del avance 1

- Modelado en Unity de las calles y agentes (coches)
- Crear objetos propios en Unity
- Conexión de un sistema multiagentes en Python con Unity y simulación
- Creación de modelos de agentes en Python con Mesa
- Ejemplos de modelado de intersecciones con semáforos en Python utilizando agentes
- La comunicación en el equipo mejoró
- Se tienen que realizar las actividades con mayor planeación

Próximas actividades

- Implementación completa de sistemas multiagentes en Python
- Agregar personas en el ambiente y juntarlo con los demás agentes para que interactúen
- Agregar el modelado de personas en Unity
- Montar servidor en IBM Cloud
- Pruebas del código de Python en Unity
- Mejoras y correcciones encontradas

Gracias