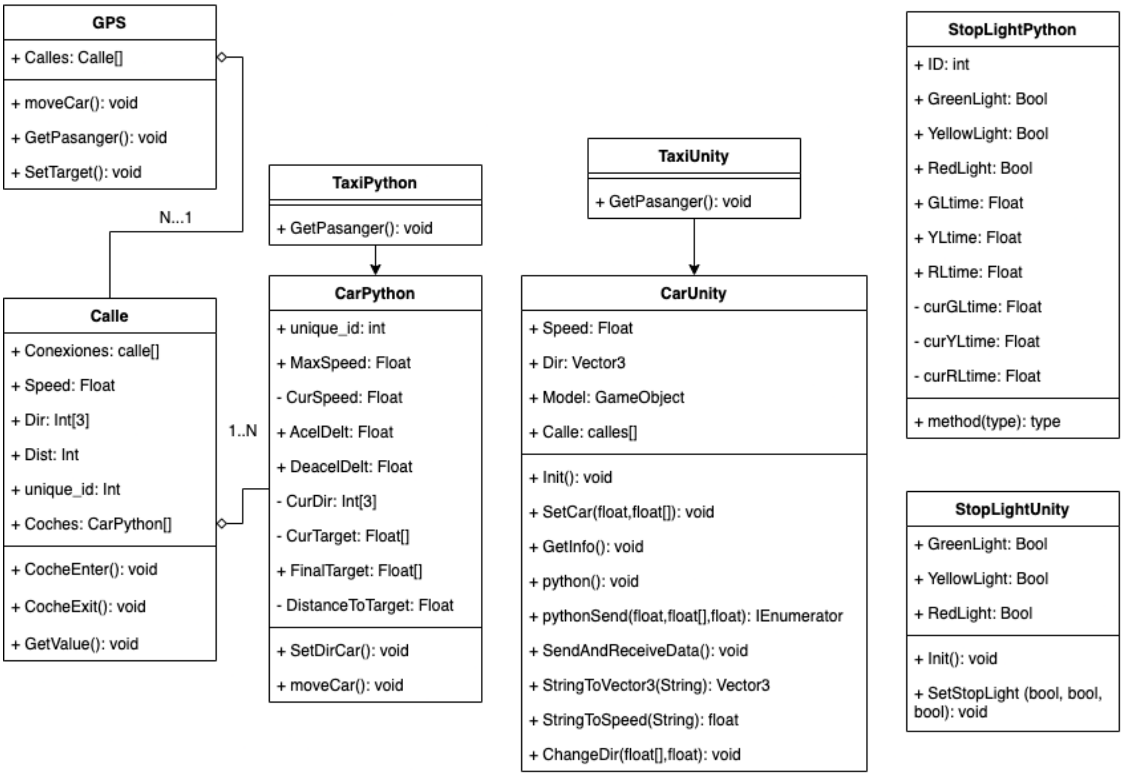
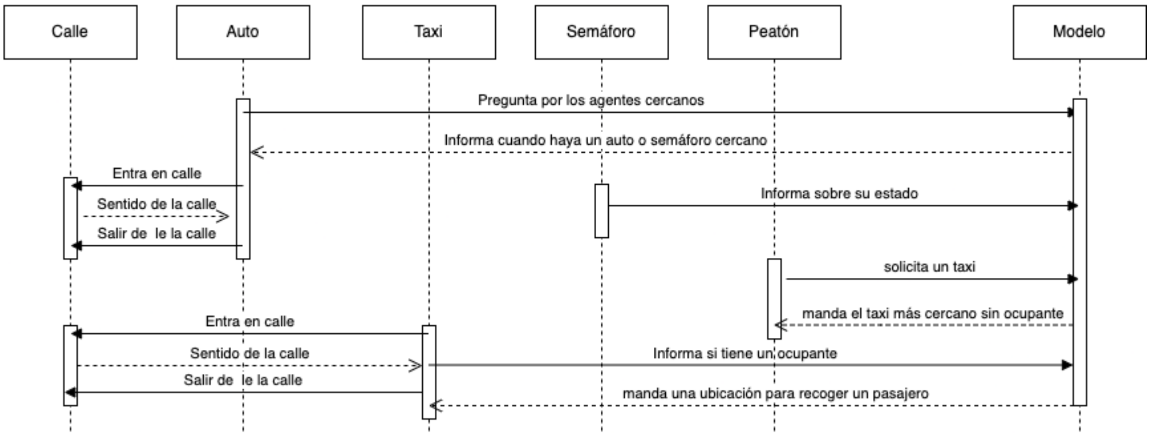
Reto Movilidad Urbana
Avance 1

Equipo:

Diógenes Grajales Corona	| A01653251
Victoria Estefanía Vázquez Morales	| A01654095
Rodolfo León Gasca	| A01653185

Diagramas de clase y protocolos de interacción actualizados




Calle
Auto
Taxi
Semáforo
Peatón
Modelo
Pregunta por los agentes cercanos
Informa cuando haya un auto o semáforo cercano
Entra en calle
Sentido de la calle
Salir de le la calle
Informa sobre su estado
solicita un taxi
manda el taxi más cercano sin ocupante
Entra en calle
Sentido de la calle
Salir de le la calle
Informa si tiene un ocupante
manda una ubicación para recoger un pasajero

Código de implementación de los agentes

```
from mesa import Agent, Model
#import random
import time
import socket

host, port = "127.0.0.1", 25001  # poner host y puerto
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((host,port))

class Grafo():
  def __init__(self):
    self.Calles = []

  def SetCalles(self,c):
    self.Calles.append(c)

class Calle(Agent):
  def __init__(self, unique_id,Speed,Dir):
    #super().__init__(unique_id, model)
    self.Speed = Speed
    self.Dir = Dir
    self.Conexiones = []
    self.Dist = 30
    self.unique_id = unique_id

  def SetConexion(self,c):
    self.Conexiones = c

  def  __ne__(self,other):
    return self.unique_id != other.unique_id

class CarPython(Agent):
  def __init__(self, unique_id,CurDir,CurTarget,FinalTarget):
    #super().__init__(unique_id,model)
    self.MaxSpeed = 10
    self.CurSpeed = 0
    self.AcelDelt = 0.25
    self.DeacelDelt = -0.25
    self.CurDir = CurDir
    self.CurTarget = CurTarget
    self.FinalTarget = FinalTarget

  def SetDirCar(self,CurTarget):
    print("Calle:" + str(CurTarget.unique_id))
    self.CurDir = CurTarget.Dir
    self.DistanceToTarget = 30
    self.CurTarget = CurTarget
    self.CurSpeed = self.CurSpeed /2

  def MoveCar(self):
    print(self.DistanceToTarget)
    self.CurSpeed += self.AcelDelt
    if(self.CurSpeed > self.MaxSpeed):
      self.CurSpeed = self.MaxSpeed

    self.DistanceToTarget -= self.CurSpeed
    tmp = []
    tmp = self.CurDir
    tmp.append(self.CurSpeed)
    print(tmp)
    SpeedString = ','.join(map(str,tmp))
    tmp.pop()
    sock.sendall(SpeedString.encode("UTF-8"))

class CarModel(Model):
    """A model with some number of agents."""
    def __init__(self, N):
        self.num_agents = N
        a = CarPython(N,self)
        # Create agents
        #for i in range(self.num_agents):
            #a = MoneyAgent(i, self)

class StopLight(Agent):
  def __init__(self,model,unique_id):
    super().__init__(unique_id)
    self.GL = False
    self.YL = True
    self.RL = False
    self.GLT = 3
    self.YLT = 1
    self.RLT = 4



def main():
  GPS = Grafo()
  #Semaforo1 = StopLight(1)
  calle1 = Calle(1,20,[0,0,1])
  GPS.SetCalles(calle1)

  #Semaforo2 = StopLight(2)
  calle2 = Calle(2,20,[1,0,0])
  GPS.SetCalles(calle2)
```

```
#Semaforo3 = StopLight(3)
calle3 = Calle(3,20,[0,0,-1])
GPS.SetCalles(calle3)

#Semaforo4 = StopLight(4)
calle4 = Calle(4,20,[-1,0,0])
GPS.SetCalles(calle4)

#Semaforo5 = StopLight(5)
calle5 = Calle(5, 20,[0,0,1])
GPS.SetCalles(calle5)

#Semaforo6 = StopLight(6)
calle6 = Calle(6, 20,[1,0,0])
GPS.SetCalles(calle6)

#Semaforo7 = StopLight(7)
calle7 = Calle(7, 20,[0,0,-1])
GPS.SetCalles(calle7)

#Semaforo8 = StopLight(8)
calle8 = Calle(8, 20,[1,0,0])
GPS.SetCalles(calle8)

#Semaforo9 = StopLight(9)
calle9 = Calle(9, 20,[0,0,1])
GPS.SetCalles(calle9)

#Semaforo10 = StopLight(10)
calle10 = Calle(10, 20,[-1,0,0])
GPS.SetCalles(calle10)

tmp = [calle2,calle5]
calle1.SetConexion(tmp)
tmp = [calle8,calle3]
calle2.SetConexion(tmp)
tmp = [calle4]
calle3.SetConexion(tmp)
tmp = [calle1]
calle4.SetConexion(tmp)
tmp = [calle6]
calle5.SetConexion(tmp)
tmp = [calle7]
calle6.SetConexion(tmp)
tmp = [calle3,calle8]
calle7.SetConexion(tmp)
tmp = [calle9]
```

```
  calle8.SetConexion(tmp)
  tmp = [calle10]
  calle9.SetConexion(tmp)
  tmp = [calle7]
  calle10.SetConexion(tmp)

  timeout = time.time() + 60
  McQueen = CarPython(1,calle1.Dir,calle1, calle9)
  McQueen.SetDirCar(calle1)

  i = 0

  while (McQueen.CurTarget != McQueen.FinalTarget):
    if(i == 0):
      if(McQueen.DistanceToTarget < 0):
        i+= 1
        McQueen.SetDirCar(calle2)
      else:
        McQueen.MoveCar()
    elif(i == 1):
      if(McQueen.DistanceToTarget < 0):
        i+= 1
        McQueen.SetDirCar(calle8)
      else:
        McQueen.MoveCar()
    elif(i == 2):
      if(McQueen.DistanceToTarget < 0):
        i+= 1
        McQueen.SetDirCar(calle9)
      else:
        McQueen.MoveCar()

    time.sleep(1)


main()
```

Código de implementación de la parte gráfica

```
using System.Collections;
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Text;
using UnityEngine;
using System.Threading;
```

```
[System.Serializable]
public struct Calle
{
    public Calle[] c;
    public Vector2 dir;
    public float speed;
}

public class CarUnity : MonoBehaviour
{
    Thread mThread;
    public string connectionIP = "127.0.0.1";
    public int connectionPort = 25001;
    IPAddress localAdd;
    TcpListener listener;
    TcpClient client;
    Vector3 receivedPos = Vector3.zero;

    public Vector3 dir;
    public GameObject CarModel;
    public float speed;

    public Calle[] calles;
    public int i;


    bool running;

    private void Start()
    {
        ThreadStart ts = new ThreadStart(GetInfo);
        mThread = new Thread(ts);
        mThread.Start();
    }

    /*void Start()
    {
        i = 0;
        python();
    }*/

    void Update()
    {
        //dir.z = 1 + (dir. * dir.y) * (1 / 2);

        dir = dir.normalized;
        transform.Translate(dir * Time.deltaTime * speed);
        //transform.position = dir;
        Quaternion newRotation = Quaternion.LookRotation(new Vector3(dir.x, 0, dir.z));
        CarModel.transform.rotation = newRotation;
    }

    public void ChangeDir(float[] vector, float s)
    {
        dir.x = vector[0];
        dir.z = vector[1];

        speed = s;

        dir = dir.normalized;
    }


    //MOCK python

    public void python()
    {
        Calle c = calles[i];
        Vector2 vtmp = c.dir;
        float[] tmp = { vtmp.x, vtmp.y };
        float t = 3.25f;
        float s = c.speed;
        i++;

        StartCoroutine(pythonSend(t, tmp,s));
    }

    IEnumerator pythonSend(float wait, float[] vector, float ss)
    {
        ChangeDir(vector, ss);
        yield return new WaitForSeconds(wait);
        if (i >= calles.Length)
        {
            i = 0;
        }
        python();
    }

    void GetInfo()
    {
        localAdd = IPAddress.Parse(connectionIP);
        listener = new TcpListener(IPAddress.Any, connectionPort);
        listener.Start();

        client = listener.AcceptTcpClient();
```

```
        running = true;
        while (running)
        {
            SendAndReceiveData();

        }
            speed = 0;

        listener.Stop();
    }

    void SendAndReceiveData()
    {
        NetworkStream nwStream = client.GetStream();
        byte[] buffer = new byte[client.ReceiveBufferSize];

        //---receiving Data from the Host----
        int bytesRead = nwStream.Read(buffer, 0, client.ReceiveBufferSize); //Getting data in
Bytes from Python
        string dataReceived = Encoding.UTF8.GetString(buffer, 0, bytesRead); //Converting
byte data to string

        if (dataReceived != null)
        {
            //---Using received data---
            receivedPos = StringToVector3(dataReceived); //<-- assigning receivedPos value
from Python
            speed = StringToSpeed(dataReceived);
            print("received pos data, and moved the Cube!");
            dir = receivedPos.normalized;



            //---Sending Data to Host----
            byte[] myWriteBuffer = Encoding.ASCII.GetBytes("Hey I got your message Python!
Do You see this massage?"); //Converting string to byte data
            nwStream.Write(myWriteBuffer, 0, myWriteBuffer.Length); //Sending the data in
Bytes to Python
        }
        else
        {
            speed = 0;
        }
    }
    public static Vector3 StringToVector3(string sVector)
    {
        // Remove the parentheses
        if (sVector.StartsWith("(") && sVector.EndsWith(")"))
        {
            sVector = sVector.Substring(1, sVector.Length - 2);
        }

        // split the items
        string[] sArray = sVector.Split(',');

        // store as a Vector3
        Vector3 result = new Vector3(
            float.Parse(sArray[0]),
            float.Parse(sArray[1]),
            float.Parse(sArray[2]));

        return result;
    }

    public static float StringToSpeed(string sVector)
    {
        // Remove the parentheses
        if (sVector.StartsWith("(") && sVector.EndsWith(")"))
        {
            sVector = sVector.Substring(1, sVector.Length - 2);
        }

        // split the items
        string[] sArray = sVector.Split(',');

        // store as a Vector3
        float result =
            float.Parse(sArray[3]);

        return result;
    }


}
```

Plan de trabajo actualizado

Tercera semana (15 al 21 de noviembre)

- Elaborar el diseño gráfico del entorno en Unity
    - Tiempo empleado aproximado: 5 horas
    - Responsable: Rodolfo
- Código de implementación de los agentes en Python
    - Tiempo empleado aproximado: 6 horas
    - Responsable: Victoria
- Código de la implementación gráfica
    - Tiempo empleado aproximado: 4 horas
    - Responsable: Diógenes

Cuarta semana (22 al 28 de noviembre)

- Diseñar el algoritmo Dijkstra en python para que los agentes encuentren la ruta más corta
    - Responsables: Victoria, Diógenes y Rodolfo
    - Esfuerzo estimado: 7 horas
- Introducir nuestro proyecto a IBM Cloud
    - Responsables: Victoria, Diógenes y Rodolfo
    - Esfuerzo estimado: 3 horas
- Conectar Unity con nuestro servidor
    - Responsables: Rodolfo
    - Esfuerzo estimado: 4 horas
- Terminar las clases para los agentes en Python
    - Responsables: Victoria
    - Esfuerzo estimado: 5 horas
- Terminar el código de la parte gráfica en Unity
    - Responsables: Diógenes
    - Esfuerzo estimado: 7 horas

Quinta semana (29 de novembre al 3 de diciembre)

- Mostrar los datos gráficamente de python en Unity
- Prueba de errores