Revisión del avance 2

Equipo 2:

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

Actualización del diagrama de clases

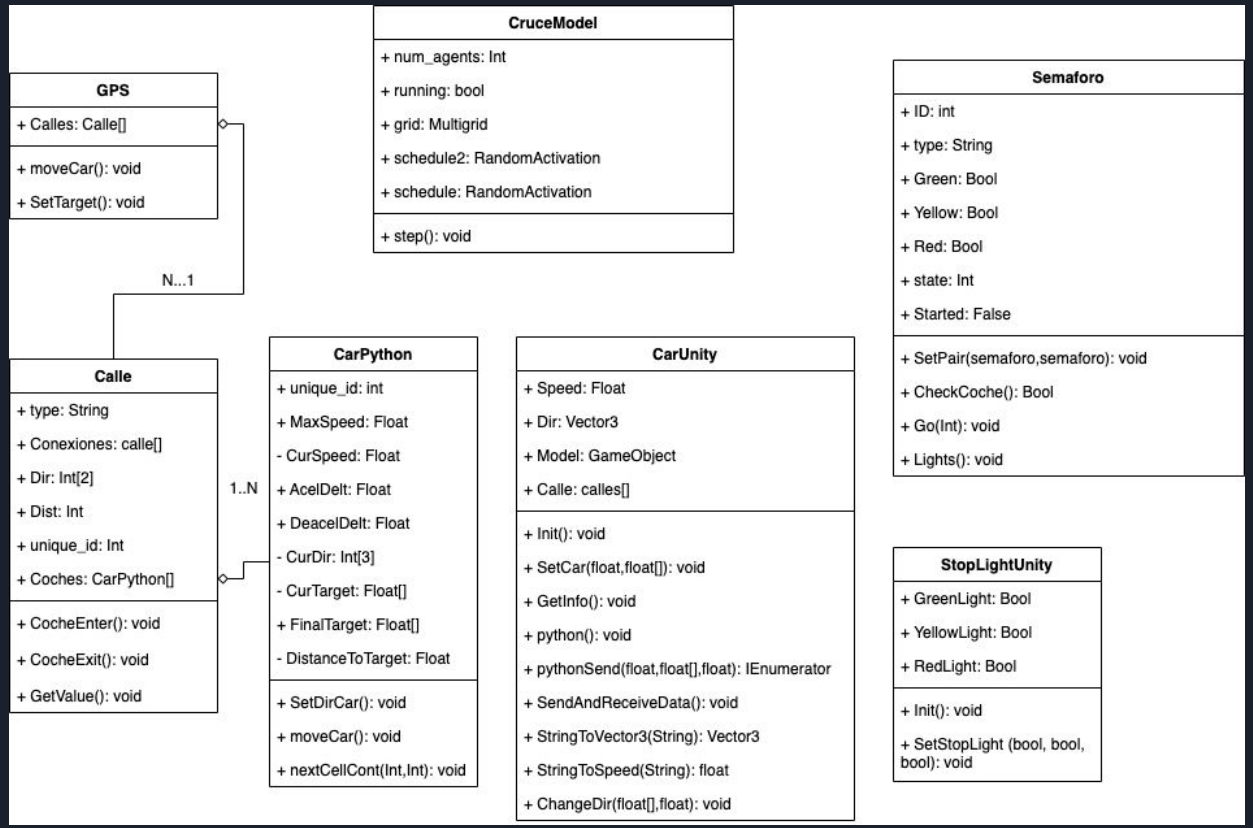
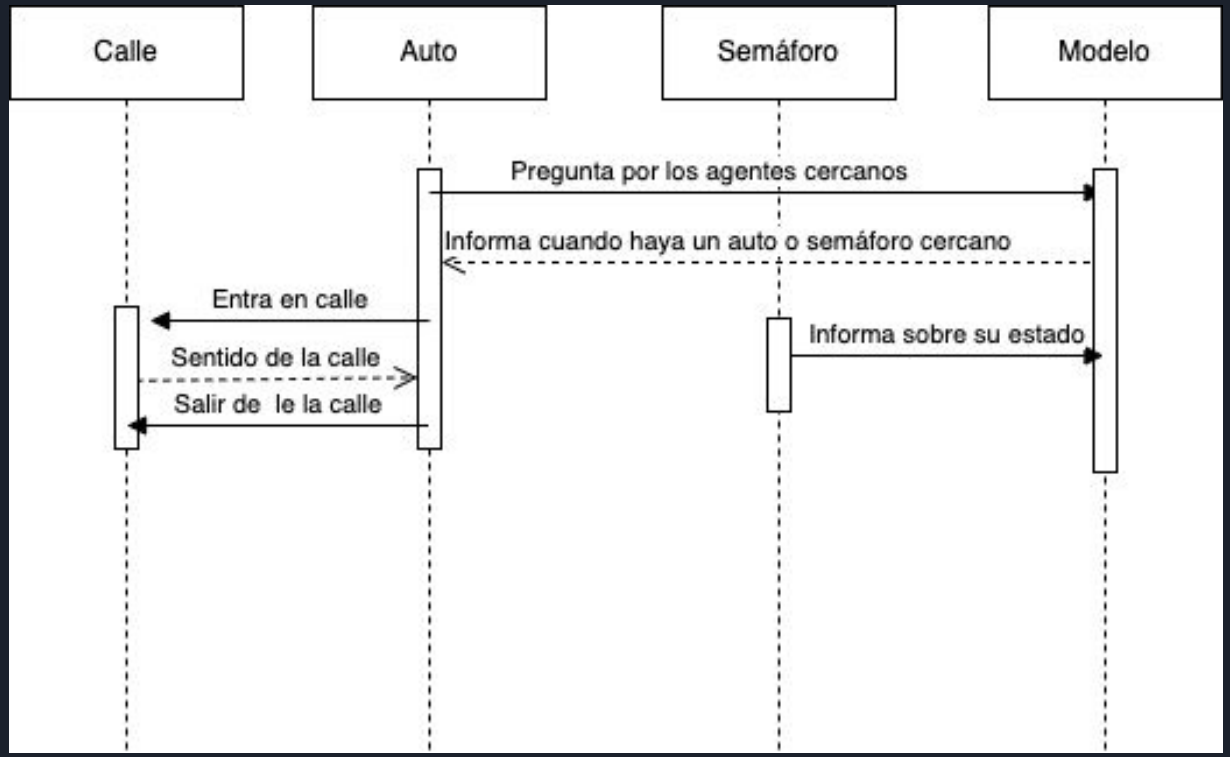


Diagrama de interacción actualizado



Código Python

```python
from mesa import Agent, Model
from mesa.time import RandomActivation
from mesa.space import MultiGrid
from mesa.visualization.modules import CanvasGrid
from mesa.visualization.ModularVisualization import ModularServer
from mesa.visualization.UserParam import UserSettableParameter
import random

class Semaforo(Agent):
    def __init__(self, unique_id, model,s):
      super().__init__(unique_id, model)
      self.type = "SEMAFORO"
      self.state = s
      self.Yellow = True
      self.Green = False
      self.Red = False
      self.Started = False
      self.rpos = [0.0,0.0]
      self.dir = 0

    def SetPair(self,Ne,No):
      self.Spar = Ne
      self.Sop = No

    def CheckCoche(self):
        cellCont = self.model.grid.get_cell_list_contents((self.pos[0], self.pos[1]))
        d = [0,0]
        if len(cellCont) > 0:
            for index in range(len(cellCont)):
                if cellCont[index].type == "CALLE":
                    d = cellCont[index].Dir
                    if d[0] < 0:
                        self.rpos = [self.pos[0], self.pos[1]+0.5]
                        self.dir = 0
                    elif d[0] > 0:
                        self.rpos = [self.pos[0], self.pos[1]-0.5]
                        self.dir = 1
                    elif d[0] == 0:
                        if d[1] < 0:
                            self.rpos = [self.pos[0]-0.5, self.pos[1]]
                            self.dir = 2
                        elif d[1] > 0:
                            self.rpos = [self.pos[0]+0.5, self.pos[1]]
                            self.dir = 3
        else:
            return False
        cellCont = self.model.grid.get_cell_list_contents(self.model.grid.torus_adj((self.pos[0]-d[0], self.pos[1]-d[1])))
        if len(cellCont) > 0:
            for index in range(len(cellCont)):
                if cellCont[index].type == "COCHE":
                    self.Sop.Go(8)
                    self.Sop.Spar.Go(0)
                    self.Spar.Go(0)

                    self.Go(8)
                    return True
        else:
            return False

        return False


    def Go(self, t):
        if self.Started == False:
            self.Started = True
            self.state = t

    def Lights(self):
        if self.Started == True:
            if self.state > 0:
                if self.state < 4:
                    self.Red = False
                    self.Green = False
                    self.Yellow = True
                else:
                    self.Yellow = False
                    self.Green = True
                    self.Red = False
            elif self.state == 0:
                self.Yellow = False
                self.Green = False
                self.Red = True
            self.state -= 1
            if self.state <= -8:
                self.state = 8
        if self.Started == False:
            self.Started = self.CheckCoche()

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

```python
class Calle(Agent):
    def __init__(self, unique_id, model, dir):
        super().__init__(unique_id, model)
        self.type = "CALLE"
        self.Dir = [0,0]
        self.Dir[0] = dir[0]
        self.Dir[1] = dir[1]

class Cruce(Agent):
    def __init__(self, unique_id, model, dir,dir2):
        super().__init__(unique_id, model)
        self.type = "CRUCE"
        self.Dir = [0,0]
        self.Dir[0] = dir[0]
        self.Dir[1] = dir[1]
        self.Dir2 = [0,0]
        self.Dir2[0] = dir2[0]
        self.Dir2[1] = dir2[1]
        print(self.Dir,self.Dir2)

class Obstaculo(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.type = "OBSTACULO"

class Coche(Agent):
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.type = "COCHE"
        self.isCollecting = True
        self.curpos = [4,4]
        self.target = [4,4]
        self.Dir = [0,0]
        self.cruzando = False

    def nextCellCont(self,X,Y):
        return self.model.grid.get_cell_list_contents(self.model.grid.torus_adj((self.pos[0]+X, self.pos[1]+Y)))
```

```python
    def GetDir(self):
        cellCont = self.model.grid.get_cell_list_contents((self.pos[0], self.pos[1]))

        if len(cellCont) > 0:
            for index in range(len(cellCont)):
                if cellCont[index].type == "CALLE":
                    self.cruzando = False
                    return cellCont[index].Dir
                elif cellCont[index].type == "CRUCE":
                    if self.cruzando == False:
                        self.cruzando = True
                        rng1 = random.randrange(0,2)
                        if rng1 == 0:
                            return cellCont[index].Dir
                        if rng1 == 1:
                            return cellCont[index].Dir2

        return self.Dir

    def Obstacle(self, cellCont):
        if len(cellCont) > 0:
            for index in range(len(cellCont)):
                if cellCont[index].type == "COCHE":
                    return True
                elif cellCont[index].type == "OBSTACULO":
                    return True
                elif cellCont[index].type ==  "SEMAFORO":
                    if cellCont[index].Red == True:
                        return True

        return False

    def move(self):

        self.curpos[0] = self.pos[0]
        self.curpos[1] = self.pos[1]

        self.Dir = self.GetDir()
        X = self.Dir[0]
        Y = self.Dir[1]

        nextCell = self.nextCellCont(X,Y)
        obstruct = self.Obstacle(nextCell)
        if (obstruct == False):
            self.model.grid.move_agent(self, (self.pos[0]+X, self.pos[1]+Y))

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

```
class CruceModel(Model):
    def __init__(self, N, width, height):
        self.num_agents = N
        self.running = True
        self.grid = MultiGrid(width, height, True)
        self.schedule2 = RandomActivation(self)
        self.schedule = RandomActivation(self)
        self.Calles = []

        dirx = 1
        diry = -1
        for g in range(round((width/2)-1),round((width/2)+1)):
            for p in range(round((width/2)-1),round((width/2)+1)):
                o = Cruce(g,self,[dirx,0],[0,diry])
                dirx *= -1
                self.grid.place_agent(o,(g,p))
                self.Calles.append(o)
            diry *= -1


        o = Cruce(1000,self,[0,-1],[0,-1])
        self.grid.place_agent(o,(0,5))
        obs1 = Calle(1000,self,[1,0])
        self.grid.place_agent(obs1, (0,4))
        self.Calles.append(o)
        self.Calles.append(obs1)

        o = Cruce(1001,self,[0,1],[0,1])
        self.grid.place_agent(o,(9,4))
        obs1 = Calle(1001,self,[-1,0])
        self.grid.place_agent(obs1, (9,5))
        self.Calles.append(o)
        self.Calles.append(obs1)

        o = Cruce(1002,self,[1,0],[1,0])
        self.grid.place_agent(o,(4,0))
        obs1 = Calle(1002,self,[0,1])
        self.grid.place_agent(obs1, (5,0))
        self.Calles.append(o)
        self.Calles.append(obs1)

        o = Cruce(1003,self,[-1,0],[-1,0])
        self.grid.place_agent(o,(5,9))
        obs1 = Calle(1003,self,[0,-1])
        self.grid.place_agent(obs1, (4,9))
        self.Calles.append(o)
        self.Calles.append(obs1)
```

```
        s1 = Semaforo(111,self,0)
        self.grid.place_agent(s1, (round((width/2)-1), round((height/2)+1)))
        s2 = Semaforo(112,self,0)
        self.grid.place_agent(s2, (round((width/2)), round((height/2)-2)))

        s3 = Semaforo(113,self,8)
        self.grid.place_agent(s3, (round((width/2)-2), round((height/2)-1)))

        s4 = Semaforo(114,self,8)
        self.grid.place_agent(s4, (round((width/2)+1), round((height/2))))
        s4.SetPair(s2,s3)
        s2.SetPair(s4,s1)
        s3.SetPair(s1,s4)
        s1.SetPair(s3,s2)

        self.schedule2.add(s1)
        self.schedule2.add(s2)
        self.schedule2.add(s3)
        self.schedule2.add(s4)

        for g in range(round((width-1)/2)):
            for p in range(round((width-1)/2)):
                o = Obstaculo(g,self)
                self.grid.place_agent(o,(g,p))

        for g in range(round((width-1)/2)):
            for p in range(round((width+2)/2),width):
                o = Obstaculo(g,self)
                self.grid.place_agent(o,(g,p))

        for g in range(round((width+2)/2),width):
            for p in range(round((width+2)/2),width):
                o = Obstaculo(g,self)
                self.grid.place_agent(o,(g,p))

        for g in range(round((width+2)/2),width):
            for p in range(round((width-1)/2)):
                o = Obstaculo(g,self)
                self.grid.place_agent(o,(g,p))

        for i in range(self.num_agents):
          a =  Coche(i, self)
          self.grid.place_agent(a, (4, 4))
          self.grid.move_to_empty(a)
          print(a.pos)

          self.schedule.add(a)
```

```
        for x in range(1,round((width-1)/2)):
            obs1 = Calle(x,self,[1,0])
            self.grid.place_agent(obs1, (x, round((width-1)/2)))
            obs2 = Calle(x+10,self,[-1,0])
            self.grid.place_agent(obs2, (x, round(width/2)))
            self.Calles.append(obs2)
            self.Calles.append(obs1)

        for x in range(round((width+1)/2), width-1):
            obs1 = Calle(x,self,[1,0])
            self.grid.place_agent(obs1, (x, round((width-1)/2)))
            obs2 = Calle(x+10,self,[-1,0])
            self.grid.place_agent(obs2, (x, round(width/2)))
            self.Calles.append(obs2)
            self.Calles.append(obs1)

        for y in range(1,round((height-1)/2)):
            obs1 = Calle(y,self,[0,-1])
            self.grid.place_agent(obs1, (round((height-1)/2),y))
            obs2 = Calle(y+10,self,[0,1])
            self.grid.place_agent(obs2, (round(height/2),y))
            self.Calles.append(obs2)
            self.Calles.append(obs1)

        for y in range(round((height+1)/2), height-1):
            obs1 = Calle(y,self,[0,-1])
            self.grid.place_agent(obs1, (round((height-1)/2),y))
            obs2 = Calle(y+10,self,[0,1])
            self.grid.place_agent(obs2, (round(height/2),y))
            self.Calles.append(obs2)
            self.Calles.append(obs1)

    def step(self):
        self.schedule.step()

        ps = []
        for i in range(self.num_agents):
            xy = self.schedule.agents[i].pos
            p = [xy[0],0,xy[1]]
            ps.append(p)
        return ps

    def step2(self):
        self.schedule2.step()
        ps = []
        for i in range(len(self.schedule2.agents)):
            p = self.schedule2.agents[i]
            print(p)
            ps.append(p)
        return ps
```

Servidor

```python
# TC2008B. Sistemas Multiagentes y Gráficas Computacionales
# Python server to interact with Unity
# Sergio. Julio 2021

from http.server import BaseHTTPRequestHandler, HTTPServer
from flask import Flask, render_template, request, jsonify
import logging
import json, os, atexit

import numpy as np
from model import CruceModel

app = Flask(__name__, static_url_path = '')

model = CruceModel(4,10,10)

def positionsToJSON(ps):
    posDICT = []
    for p in ps:
        pos = {
            "x" : p[0],
            "y" : p[1],
            "z" : p[2]
        }
        posDICT.append(pos)
    return json.dumps(posDICT)

def BoolsToJSON(g):
    posDICT = []
    for p in g:
        pos = {
            "Green" : p.Green,
            "Yellow" : p.Yellow,
            "Red" : p.Red,
            "Posx" : p.rpos[0],
            "Posz" : p.rpos[1],
            "dir" : p.dir
        }
        posDICT.append(pos)
    return json.dumps(posDICT)
```

```python
def CallesToJSON(g):
    posDICT = []
    for p in g:
        pos = {
            "x" : p.pos[0],
            "y" : -1,
            "z" : p.pos[1]
        }
        posDICT.append(pos)
    return json.dumps(posDICT)

port = int(os.getenv('PORT',8585))

@app.route('/')

def root():
    return jsonify([{"message":"Hello"}])

@app.route('/calles', methods=['GET','POST'])

def calles():
    positions = model.Calles
    return CallesToJSON(positions)

@app.route('/muliagentes', methods=['GET','POST'])

def multiagentes():
    positions = model.step()
    return positionsToJSON(positions)

@app.route('/semaforos', methods=['GET','POST'])

def semaforos():
    lights = model.step2()
    return BoolsToJSON(lights)


if __name__ == '__main__':
    app.run(host='0.0.0.0',port = port, debug = True)
```

Código Unity

CarTest.cs | CallesClient.cs | SemaforosClient.cs

```cs
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEditor;
using UnityEngine;
using UnityEngine.Networking;

public class CarTest : MonoBehaviour
{
    public List<Robot> agents;
    public List<Vector3> Positions;
    public float timeToUpdate = 5.0f;
    private float timer;
    bool started = false;
    public GameObject CarPref;

    IEnumerator SendData(string data)
    {
        WWWForm form = new WWWForm();
        form.AddField("bundle", "the data");
        string url = "http://localhost:8585/muliagentes";
        //using (UnityWebRequest www = UnityWebRequest.Post(url, form))
        using (UnityWebRequest www = UnityWebRequest.Get(url))
        {
            byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(data);
            www.uploadHandler = (UploadHandler)new UploadHandlerRaw(bodyRaw);
            www.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
            //www.SetRequestHeader("Content-Type", "text/html");
            www.SetRequestHeader("Content-Type", "application/json");

            yield return www.SendWebRequest();          // Talk to Python
            if (www.isNetworkError || www.isHttpError)
            {
                Debug.Log(www.error);
            }
            else
            {

                List<Vector3> newPositions = new List<Vector3>();
                string txt = www.downloadHandler.text.Replace('\'', '\"');
                txt = txt.TrimStart('"', '{', 'd', 'a', 't', 'a', ':', '[');
                txt = "{\"" + txt;
                txt = txt.TrimEnd(']', '}');
                txt = txt + '}';
                string[] strs = txt.Split(new string[] { "}, {" }, StringSplitOptions.None);

                for (int i = 0; i < strs.Length; i++)
                {
                    strs[i] = strs[i].Trim();
                    if (i == 0) strs[i] = strs[i] + '}';
                    else if (i == strs.Length - 1) strs[i] = '{' + strs[i];
                    else strs[i] = '{' + strs[i] + '}';
                    Vector3 test = JsonUtility.FromJson<Vector3>(strs[i]);
                    newPositions.Add(test);
                }

                Positions = newPositions;

                if (!started)
                {
                    started = true;
                    foreach (Vector3 p in Positions)
                    {
                        Robot a = Instantiate(CarPref, p * 5f, Quaternion.identity, null).GetComponent<Robot>();
                        agents.Add(a);
                    }
                }
                else
                {

                    for (int i = 0; i < agents.Count; i++)
                    {
                        if (agents[i].target != Positions[i] * 5f)
                        {
                            agents[i].changeDir(Positions[i] * 5f);
                        }
                    }
                }
            }
        }

    }

    // Start is called before the first frame update
    void Start()
    {
        InvokeRepeating("CallServer", 2, 1);
        /*
#if UNITY_EDITOR

        Vector3 fakePos = new Vector3(3.44f, 0, -15.707f);
        string json = EditorJsonUtility.ToJson(fakePos);
```

```
84      // Start is called before the first frame update
85      void Start()
86      {
87          InvokeRepeating("CallServer", 2, 1);
88          /*
89  #if UNITY_EDITOR
90
91          Vector3 fakePos = new Vector3(3.44f, 0, -15.707f);
92          string json = EditorJsonUtility.ToJson(fakePos);
93
94          StartCoroutine(SendData(json));
95          timer = timeToUpdate;
96  #endif*/
97      }
98
99      void CallServer()
100     {
101         Vector3 fakePos = new Vector3(3.44f, 0, -15.707f);
102         string json = EditorJsonUtility.ToJson(fakePos);
103
104         StartCoroutine(SendData(json));
105     }
106 }
107
```

CarTest.cs | CallesClient.cs | SemaforosClient.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEditor;
using UnityEngine;
using UnityEngine.Networking;

public class CallesClient : MonoBehaviour
{
    List<Vector3> Positions;

    public GameObject calle;

    IEnumerator SendData(string data)
    {
        WWWForm form = new WWWForm();
        form.AddField("bundle", "the data");
        string url = "http://localhost:8585/calles";
        //using (UnityWebRequest www = UnityWebRequest.Post(url, form))
        using (UnityWebRequest www = UnityWebRequest.Get(url))
        {
            byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(data);
            www.uploadHandler = (UploadHandler)new UploadHandlerRaw(bodyRaw);
            www.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
            //www.SetRequestHeader("Content-Type", "text/html");
            www.SetRequestHeader("Content-Type", "application/json");

            yield return www.SendWebRequest();          // Talk to Python
            if (www.isNetworkError || www.isHttpError)
            {
                Debug.Log(www.error);
            }
            else
            {

                List<Vector3> newPositions = new List<Vector3>();
                string txt = www.downloadHandler.text.Replace('\'', '\"');
                txt = txt.TrimStart('"', '{', 'd', 'a', 't', 'a', ':', '[');
                txt = "{\"" + txt;
                txt = txt.TrimEnd(']', '}');
                txt = txt + '}';
                string[] strs = txt.Split(new string[] { "}, {" }, StringSplitOptions.None);
                Debug.Log("strs.Length:" + strs.Length);
                for (int i = 0; i < strs.Length; i++)
                {
                    Debug.Log(strs[i]);
                    strs[i] = strs[i].Trim();
                    if (i == 0) strs[i] = strs[i] + '}';
                    else if (i == strs.Length - 1) strs[i] = '{' + strs[i];
                    else strs[i] = '{' + strs[i] + '}';
                    Debug.Log(strs[i]);
                    Vector3 test = JsonUtility.FromJson<Vector3>(strs[i]);
                    newPositions.Add(test);
                }

                Positions = newPositions;

                foreach (Vector3 p in Positions)
                {
                    Vector3 vec = p;
                    vec.x *= 5;
                    vec.z *= 5;

                    GameObject a = Instantiate(calle, vec, Quaternion.identity, null);

                }


            }
        }

    }


    // Start is called before the first frame update
    void Start()
    {
        CallServer();
        /*
#if UNITY_EDITOR

        Vector3 fakePos = new Vector3(3.44f, 0, -15.707f);
        string json = EditorJsonUtility.ToJson(fakePos);

        StartCoroutine(SendData(json));
        timer = timeToUpdate;
#endif*/
    }

    void CallServer()
    {
        Debug.Log("hola");
        Vector3 fakePos = new Vector3(3.44f, 0, -15.707f);
        string json = EditorJsonUtility.ToJson(fakePos);

        StartCoroutine(SendData(json));
    }
}
```

CarTest.cs | CallesClient.cs | SemaforosClient.c

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEditor;
using UnityEngine;
using UnityEngine.Networking;

public class SemaforosClient : MonoBehaviour
{
    public List<Semaforo> agents;
    public List<DataSemaforo> Positions;
    public float timeToUpdate = 5.0f;
    private float timer;
    public GameObject s1,s2,s3,s4;
    bool started = false;

    IEnumerator SendData(string data)
    {
        WWWForm form = new WWWForm();
        form.AddField("bundle", "the data");
        string url = "http://localhost:8585/semaforos";
        //using (UnityWebRequest www = UnityWebRequest.Post(url, form))
        using (UnityWebRequest www = UnityWebRequest.Get(url))
        {
            byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(data);
            www.uploadHandler = (UploadHandler)new UploadHandlerRaw(bodyRaw);
            www.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
            //www.SetRequestHeader("Content-Type", "text/html");
            www.SetRequestHeader("Content-Type", "application/json");

            yield return www.SendWebRequest();          // Talk to Python
            if (www.isNetworkError || www.isHttpError)
            {
                Debug.Log(www.error);
            }
            else
            {

                List<DataSemaforo> newPositions = new List<DataSemaforo>();
                string txt = www.downloadHandler.text.Replace('\'', '\"');
                txt = txt.TrimStart('"', '{', 'd', 'a', 't', 'a', ':', '[');
                txt = "{\"" + txt;
                txt = txt.TrimEnd(']', '}');
                txt = txt + '}';
                string[] strs = txt.Split(new string[] { "}, {" }, StringSplitOptions.None);
                Debug.Log("strs.Length:" + strs.Length);
                for (int i = 0; i < strs.Length; i++)
                {
                    Debug.Log(strs[i]);
                    strs[i] = strs[i].Trim();
                    if (i == 0) strs[i] = strs[i] + '}';
                    else if (i == strs.Length - 1) strs[i] = '{' + strs[i];
                    else strs[i] = '{' + strs[i] + '}';

                    Debug.Log(strs[i]);
                    DataSemaforo test = JsonUtility.FromJson<DataSemaforo>(strs[i]);
                    newPositions.Add(test);
                }

                Positions = newPositions;
```

```
62                  if (!started)
63                  {
64                      started = true;
65                      foreach (DataSemaforo p in Positions)
66                      {
67                          Semaforo a = null;
68
69                          if (p.dir == 0) {
70                              a = Instantiate(s1, this.transform.position, Quaternion.identity, null).GetComponent<Semaforo>();
71
72                          }
73                          else if (p.dir == 1)
74                          {
75                              a = Instantiate(s1, this.transform.position, Quaternion.identity, null).GetComponent<Semaforo>();
76                              a.transform.eulerAngles = new Vector3(0, 180, 0);
77                          }
78                          else if (p.dir == 2)
79                          {
80                              a = Instantiate(s1, this.transform.position, Quaternion.identity, null).GetComponent<Semaforo>();
81                              a.transform.eulerAngles = new Vector3(0, -90, 0);
82                          }
83                          else if (p.dir == 3)
84                          {
85                              a = Instantiate(s1, this.transform.position, Quaternion.identity, null).GetComponent<Semaforo>();
86                              a.transform.eulerAngles = new Vector3(0, 90, 0);
87                          }
88
89                          EqualSemaforo(a, p);
90                          agents.Add(a);
91                      }
92                  }
93                  else
94                  {
95
96                      for (int i = 0; i < agents.Count; i++)
97                      {
98                          EqualSemaforo(agents[i], Positions[i]);
99                      }
00                  }
01
02              }
03          }
04
05      }
```

```
107         void EqualSemaforo(Semaforo s, DataSemaforo ds)
108         {
109             s.Posz = ds.Posz*5;
110             s.Green = ds.Green;
111             s.Yellow = ds.Yellow;
112             s.Red = ds.Red;
113             s.Posx = ds.Posx*5;
114 
115         }
116 
117         // Start is called before the first frame update
118         void Start()
119         {
120             InvokeRepeating("CallServer", 2, 1);
121             /*
122     #if UNITY_EDITOR
123 
124             Vector3 fakePos = new Vector3(3.44f, 0, -15.707f);
125             string json = EditorJsonUtility.ToJson(fakePos);
126 
127             StartCoroutine(SendData(json));
128             timer = timeToUpdate;
129     #endif*/
130         }
131 
132         void CallServer()
133         {
134             Debug.Log("hola");
135             Vector3 fakePos = new Vector3(3.44f, 0, -15.707f);
136             string json = EditorJsonUtility.ToJson(fakePos);
137 
138             StartCoroutine(SendData(json));
139         }
140     }
```

Plan de trabajo actualizado

Lo que se realizó esta semana

Cuarta semana (22 al 26 de noviembre**)**

- **Introducir nuestro proyecto a IBM Cloud**
  - Responsables: Victoria, Diógenes y Rodolfo
  - Tiempo empleado aproximado: 3 horas
- Conectar Unity con nuestro **servidor**
  - Responsables: Rodolfo
  - Tiempo empleado aproximado: 4 horas
- Terminar las clases para los agentes en Python
  - Responsables: Victoria
  - Tiempo empleado aproximado: 5 horas
- Terminar el código de la parte gráfica en Unity
  - Responsables: Diógenes
  - Tiempo empleado aproximado: 7 horas

Quinta semana (29 de noviembre al 1 de diciembre)

- Ajustar el algoritmo Dijkstra en python para que los agentes encuentren la ruta más corta
  - Responsables: Victoria, Diógenes y Rodolfo
  - Esfuerzo estimado: 7 horas
- Realizar las conexiones finales entre Unity-IBMCloud-Python
  - Responsable: Rodolfo
  - Esfuerzo estimado: 6 horas
- Generar el mapa final en Unity
  - Responsable: Diógenes
  - Esfuerzo estimado: 5 horas
- Realizar pruebas de errores
  - Responsable: Victoria
  - Esfuerzo estimado: 3 horas