SUMO Visualization with Unity

2020 Spring ICG Final Project
Group 24
R09944016 周良冠
R09922155 簡丞珮

I. Introduction

Motivation and Background

Our research background is about intelligent vehicles and traffic control, and SUMO is one of the simulation softwares we use for experiments. Since the graphical user interface of SUMO is dull, we think it will be interesting to build a visualization tool for it. On the other hand, Unity is one of the most popular 3D engines. There are tons of resources, including free 3D assets and well-written documentation to help us. Thus, we decided to implement the visualization of SUMO with Unity.

Introduction of SUMO

SUMO is the abbreviation of “Simulation of Urban MObility”. According to its website documentation [1], it is an open source, highly portable, microscopic and continuous traffic simulation package designed to handle large networks. It allows for intermodal simulation including pedestrians and comes with a large set of tools for scenario creation. It is mainly developed by employees of the Institute of Transportation Systems at the German Aerospace Center. In our project, SUMO is used to simulate random traffic in a road network.

Introduction of Unity

According to Unity website [2], Unity is the world’s leading platform for creating and operating interactive, real-time 3D content, providing the tools to make amazing games and publish them to a wide range of devices. Although Unity is mainly used for gaming development, it can also be used for graphic rendering, physics simulations or simply a visualization tool. In this project, Unity is used as a visualization tool to construct a pretty urban view for SUMO.

II. Implementation

The interface (Traci_one)

We use TraCI.NET (http://TraCI.NET) [3] as the interface between SUMO and Unity. First, when we start SUMO, it loads the simulation scenario and starts listening to the appointed port. After that, we use a C# script named Traci_one.cs to set up a TCP connection to that port. With the connection, we can retrieve the information of the simulation scenario from SUMO by using the functions provided by TraCI.NET (http://TraCI.NET), and set the values of the corresponding objects in Unity. Also, we can get values of the objects from Unity, and manipulate the behaviors of the vehicles in SUMO, as shown in Figure 1.

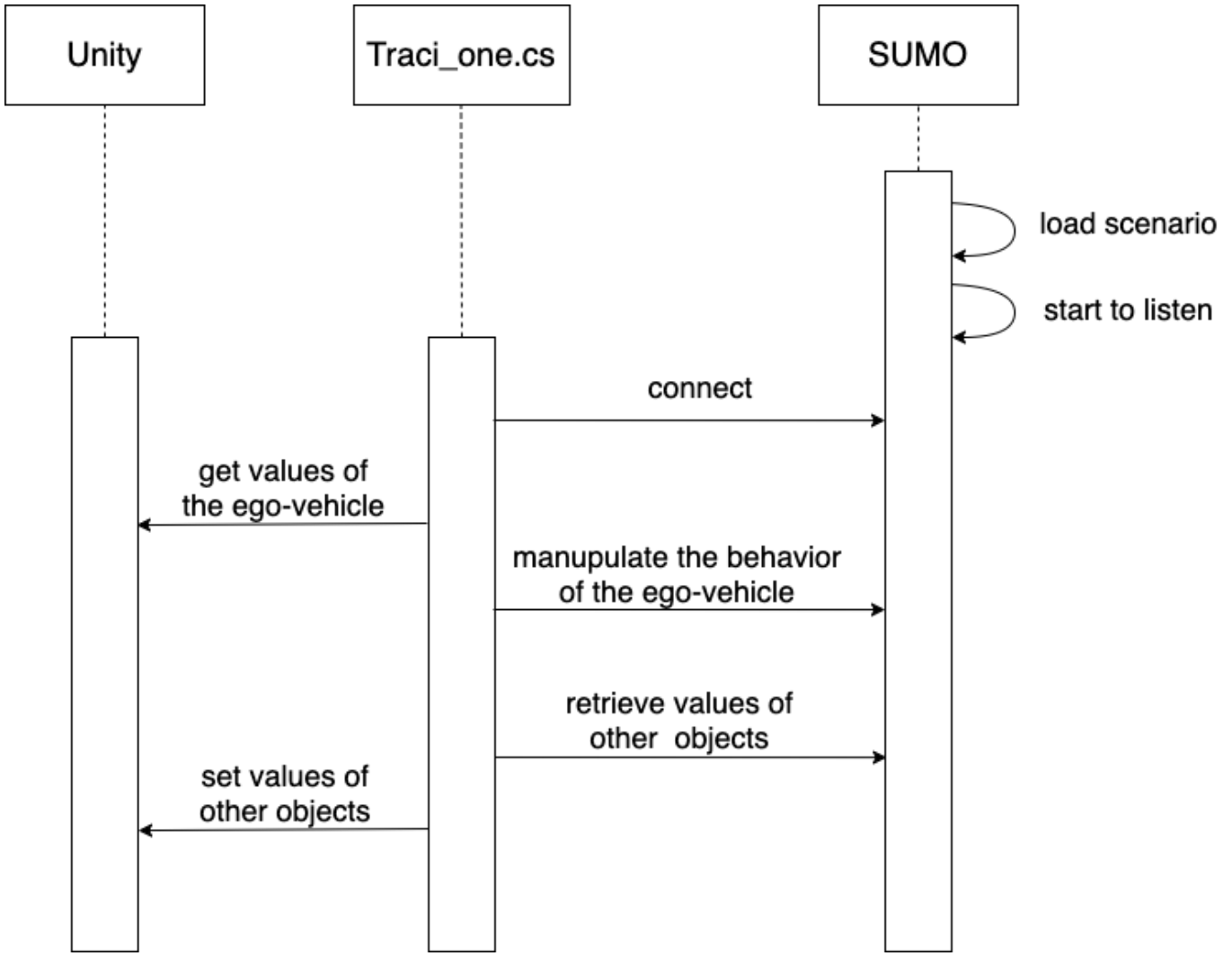

Fig. 1. Interface between SUMO and Unity

Build on top of which project

Our project is built on top of the project named Real-time-Traffic-Simulation-with-3D-Visualisation [4] on GitHub. This project utilized TraCI.NET (http://TraCI.NET) mentioned above to synchronize between Unity and SUMO. It contains a basic control of an ego vehicle, and a skeleton scene.

III. Added Features

Prettify the scene

To visualize SUMO, we need 3D assets of vehicles, buildings, roads, traffic lights, trees, etc. We downloaded free assets that meet our requirements from the asset store in Unity, and imported them into our project. Figure 2 is the overview of the whole scene. Figure 3 shows the details.

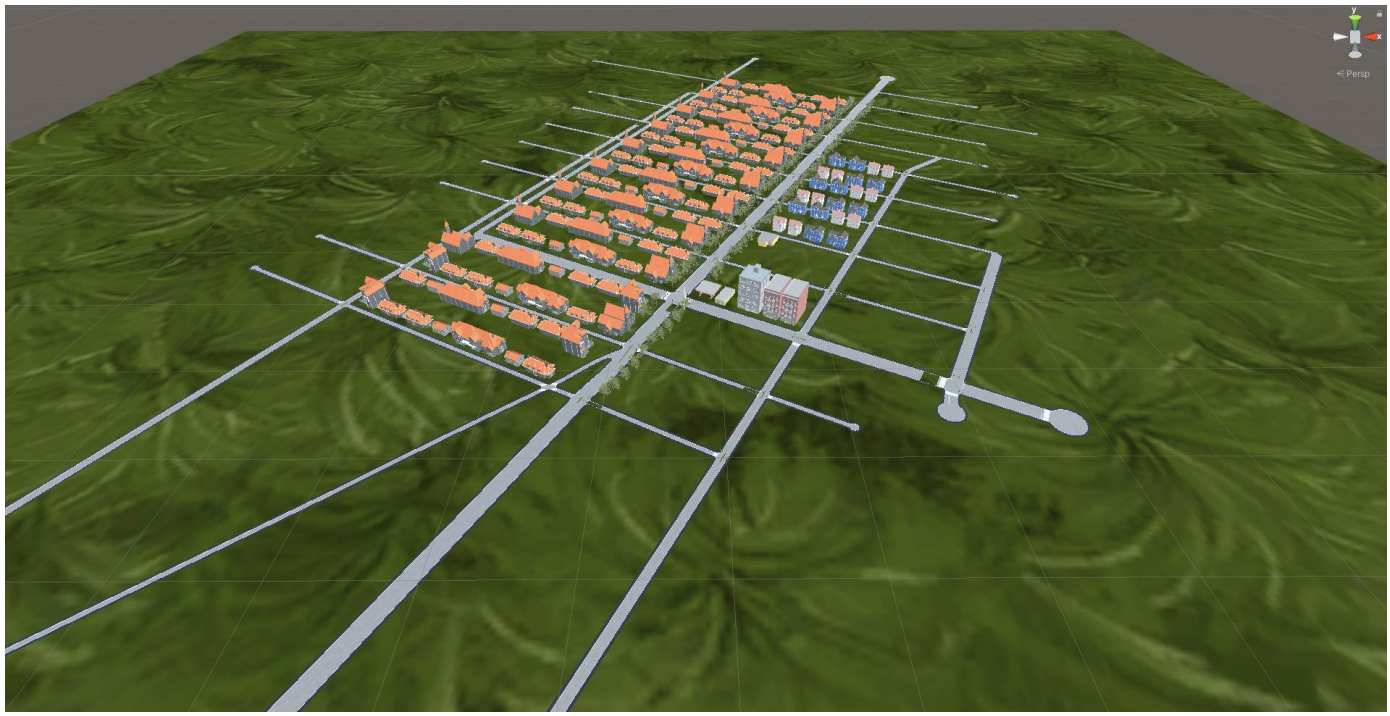

Fig. 2. Overview of the scene.

Fig. 3. The details of the scene.

Add objects to the scene

SUMO supports adding new objects during runtime. We implemented this feature so that users can add three different kinds of objects to SUMO via Unity, including emergency vehicles, pedestrians and obstacles. All the newly added objects will also be controlled by SUMO and behave just like the one created at the beginning. Figure 4, figure 5 and figure 6

shows both how SUMO and Unity look like while adding new objects.

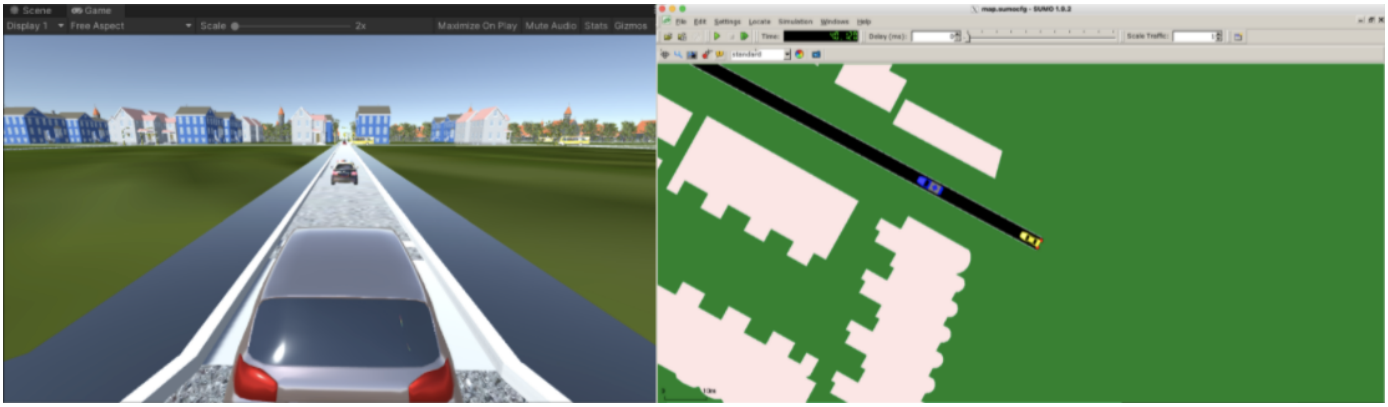

Fig. 4. Add an emergency vehicle.

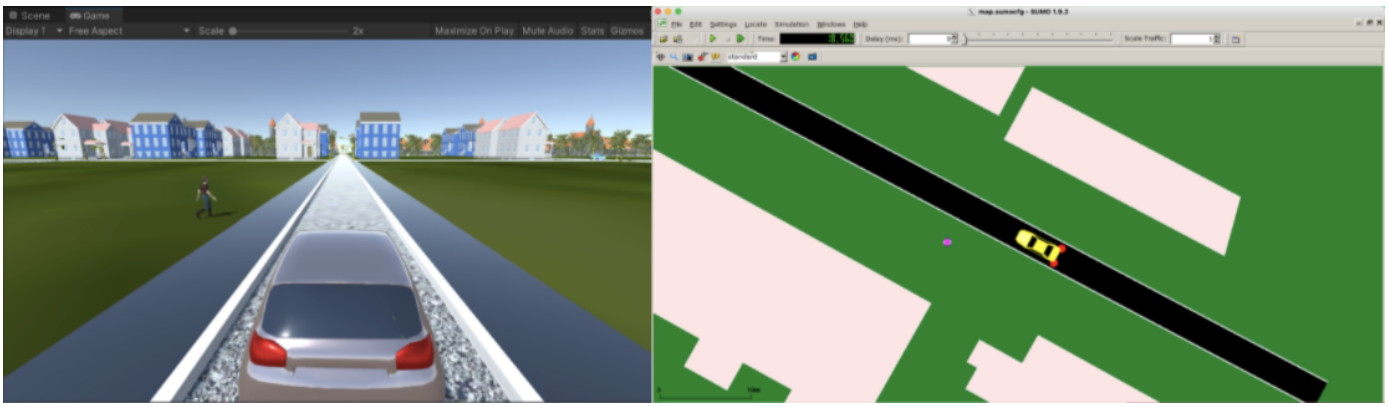

Fig. 5. Add a pedestrian.

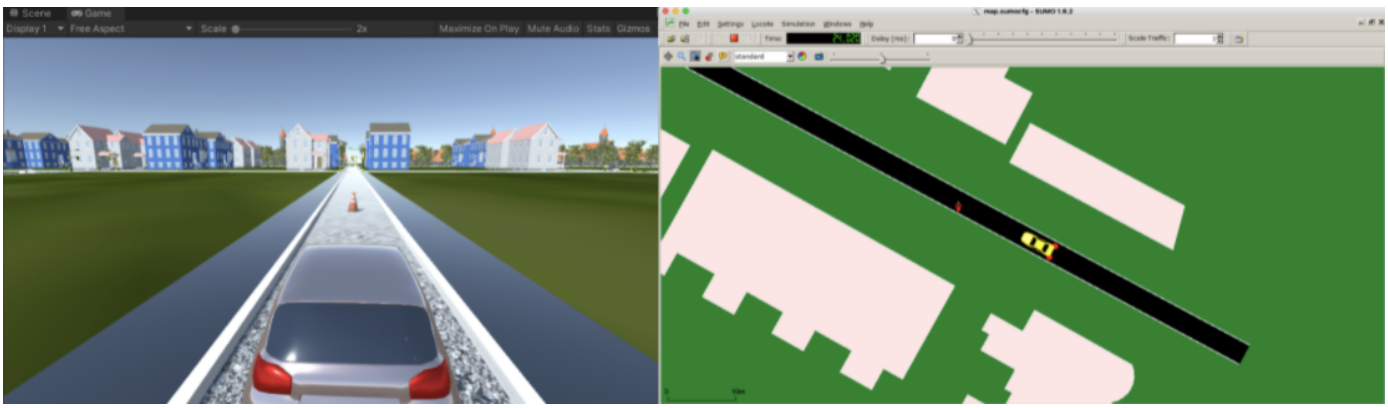

Fig. 6. Add an obstacle.

Add different views

In the original project, it only supports a fixed viewpoint behind the ego vehicle. We added view rotation by mouse control. We also implemented the top view and the rearview mirror.

Fig. 7. First person view and view rotation.

Fig. 8. Side view.

Fig. 9. Top view.

Fig. 10. Rearview mirror.

Show information about the scene

SUMO is a powerful traffic simulator that is able to calculate and record a variety of data, including position, speed, emission, etc. We ported the data from SUMO to Unity so that users can view the real time traffic information in Unity.

Fig. 11. Show speed and distance.

Fig. 12. Show other traffic information.

IV. How to Use

The following steps describe the flow of using this project.

1. Select an Area in OSM Web Wizard and Export the File

Open OSM Web Wizard and select the area of interest to export the corresponding osm file, as shown in Figure 13.

Fig. 13. Select an Area in OSM Web Wizard

2. Use CityEngine to Generate a 3D Model From the OSM File

Import the osm file generated from the previous step into CityEngine, and then generate the corresponding 3D model as shown in Figure 14.

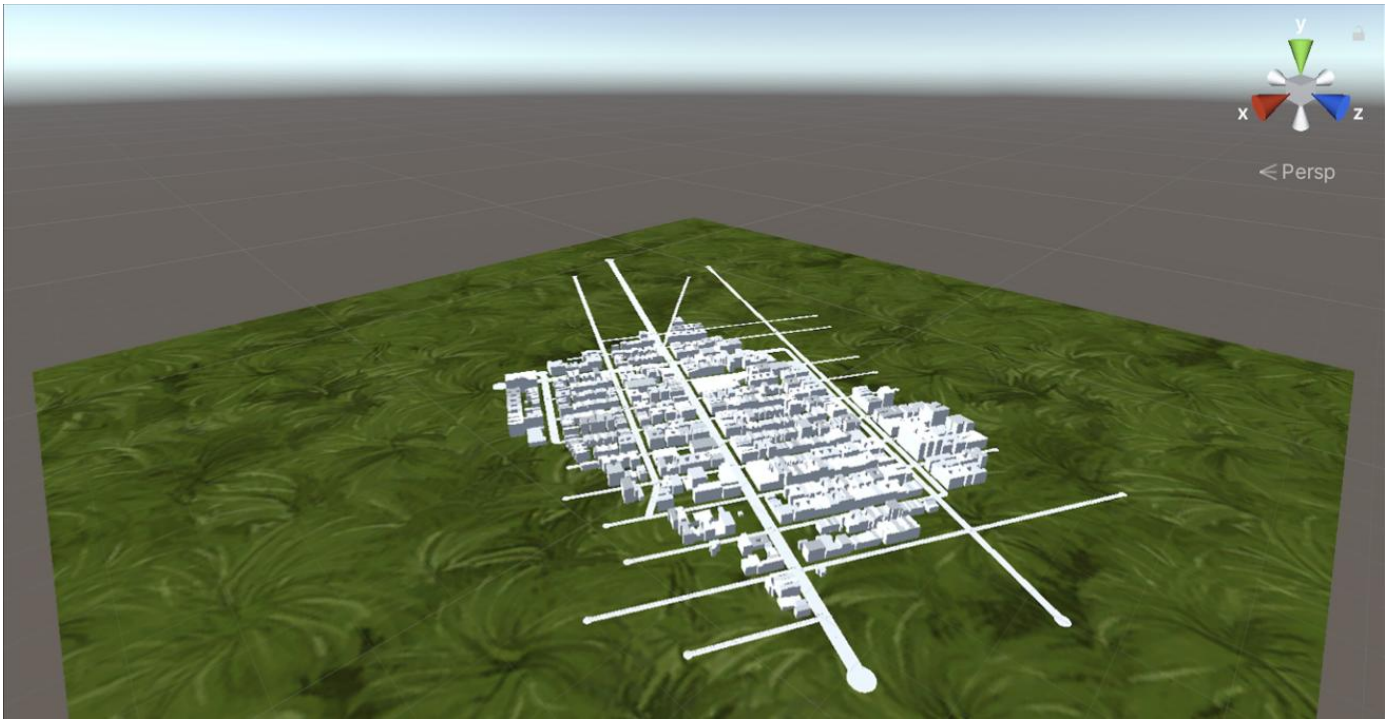

Fig. 14. The 3D model generated from the osm file.

3. Generate the input data for SUMO

To start SUMO simulation, we need the following four types of files: (1) net file, (2) polygon file, (3) route file, and (4) sumo configuration file.

(1) net file (.net.xml)

The net file is used to define the road network in the simulation scenario. The following command is used to generate the corresponding net file from the osm file.

```
netconvert --osm-files map.osm.xml -o map.net.xml
```

(2) polygon file (.poly.xml)

The polygon file is used to define the building in the simulation scenario. The following command is used to generate the corresponding poly file from the osm file and the net file.

```
polyconvert --net-file map.net.xml --osm-files map.osm --type-file typemap.xml -o map.poly.xml
```

(3) route file (.rou.xml)

The route file is used to define the traffic in the simulation scenario. The following command is used to randomly generate a route file from the net file.

```
python sumo/tools/randomTrips.py -n map.net.xml -r map.rou.xml
```

(4) sumo configuration file (.sumocfg)

The sumo configuration file is used to define the files loaded for the simulation.
We need to write the three types of files above in the sumo configuration file.

4. Start SUMO Simulation and Click Play in Unity

The following command is used to start SUMO and listen to port 4001.

```
sumo-gui -c map.sumocfg --start --remote-port 4001
```

After starting sumo, we can click the “Play” button in Unity and start synchronization.

V. Reference

[1] SUMO Documentation (https://sumo.dlr.de/docs/index.html)
[2] Unity (https://unity.com/products/unity-platform)
[3] CodingConnected.TraCI (https://github.com/CodingConnected/CodingConnected.Traci)
[4] Real-time-Traffic-Simulation-with-3D-Visualisation (https://github.com/DarraghMac97/Real-time-Traffic-Simulation-with-3D-Visualisation)

VI. Work Distribution

周良冠:

- Run [4].
- Prettify the scene.
- Add view rotation and optimize the user experience when adding new objects.
- Record demo.

簡丞珮：

- Add objects to the Unity scene and synchronize in SUMO.
- Add the top view and the rearview mirror.
- Show information about the simulation scenario.
- Record presentation.