

# RWR 4015

# Traffic Simulation for Planning Applications

Dr. Ahmad Mohammadi

Week 10 | Hands-on:  
3D Simulation in Planning I

Fall 2026

RoadwayVR



[roadwayvr.github.io/TrafficSimulationforPlanningApplications](https://roadwayvr.github.io/TrafficSimulationforPlanningApplications)



# Agenda

- Install Sumo2Unity tool**
- Install Unity Game Engine**
- Visualize a Single Lane Road with one Unsignalized Intersection**

# Sumo2Unity Tool

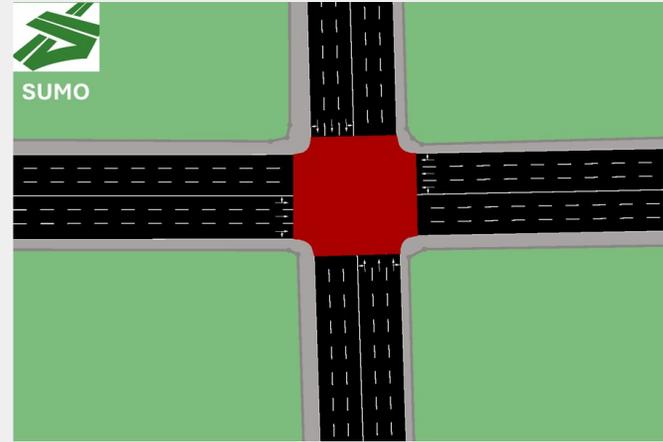
- Download Sumo2Unity V2.0.0**
- SUMO Installation and Quit Start**
- Unity VR Installation**
- Visualize a Single Lane Road with one Unsignalized Intersection**

# Sumo2Unity Tool

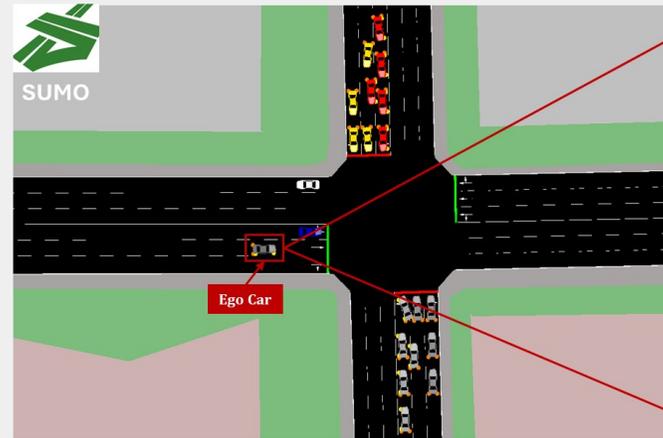
Generate:

- Complex Road Network
- Complex Traffic System
- Performance Functions (Analytics)

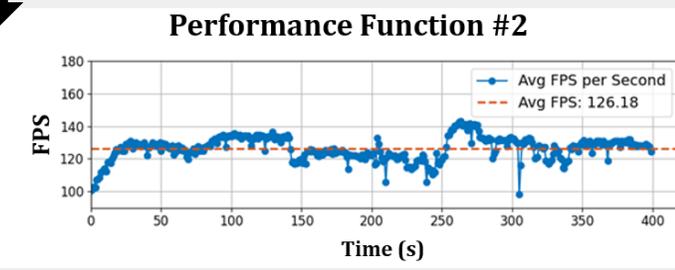
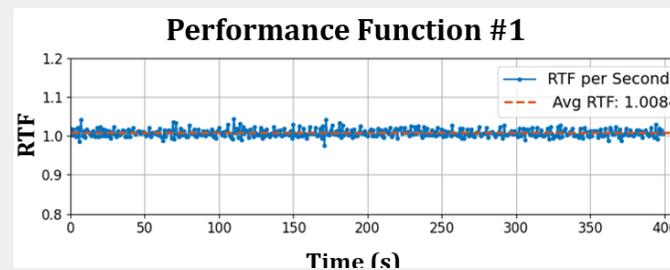
1



2



3



# Unity Installation

1. Install Unity HUB
2. Install Unity Editor (Version 6000.0.53f1)
3. Install Visual Studio
4. Install Visual Studio Dependencies



Watch:  
“Unity VR Tutorial Part 1.1. Install Unity HUB and Visual Studio”



[https://youtu.be/ngccSGH3-\\_8](https://youtu.be/ngccSGH3-_8)

# Sumo2Unity Tutorials

- Scenario 1 (Single Lane Road with one Unsignalized Intersection)**
- Scenario 2 (Multi Lane Road with Signalized Intersection)**
- Scenario 3 (Bicycle)**
- Scenario 4 (Scooters)**
- Scenario 5 (Different Traffic Density: Level of Service)**

# Scenario 1:

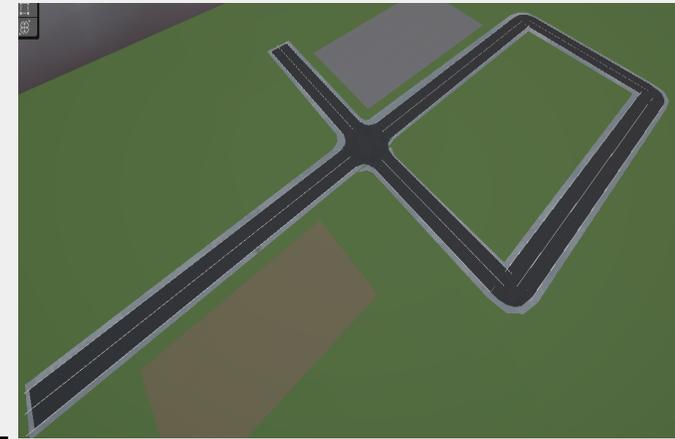
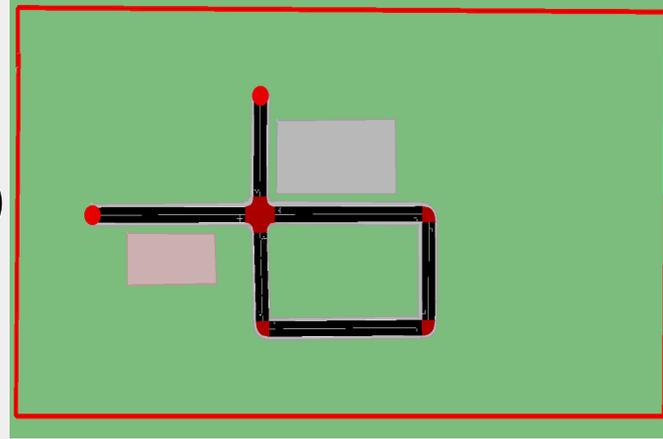
## Single Lane Road with one Unsignalized Intersection

1. Create Road Network

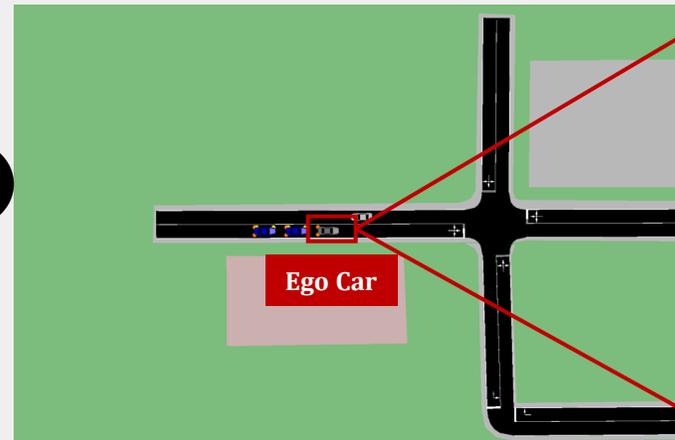
2. Run Sumo2Unity Integration

3. Generate Performance Functions

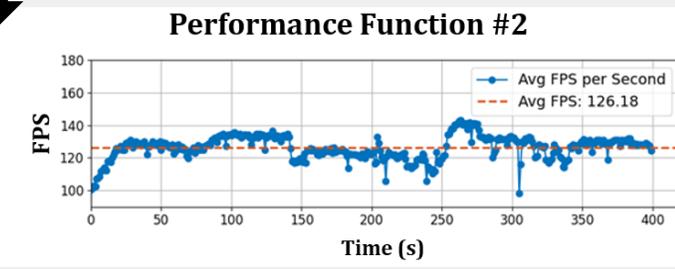
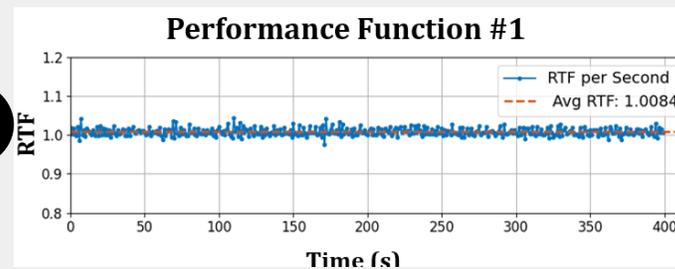
1



2



3



# Step 1: Create Road Network

## 1.1. SUMO Steps

- A) Adding Lane
- B) Adding Terrain
- C) Adding Roadside
- D) Adding Residential
- E) Adding Wood

## 1.2. Unity Steps

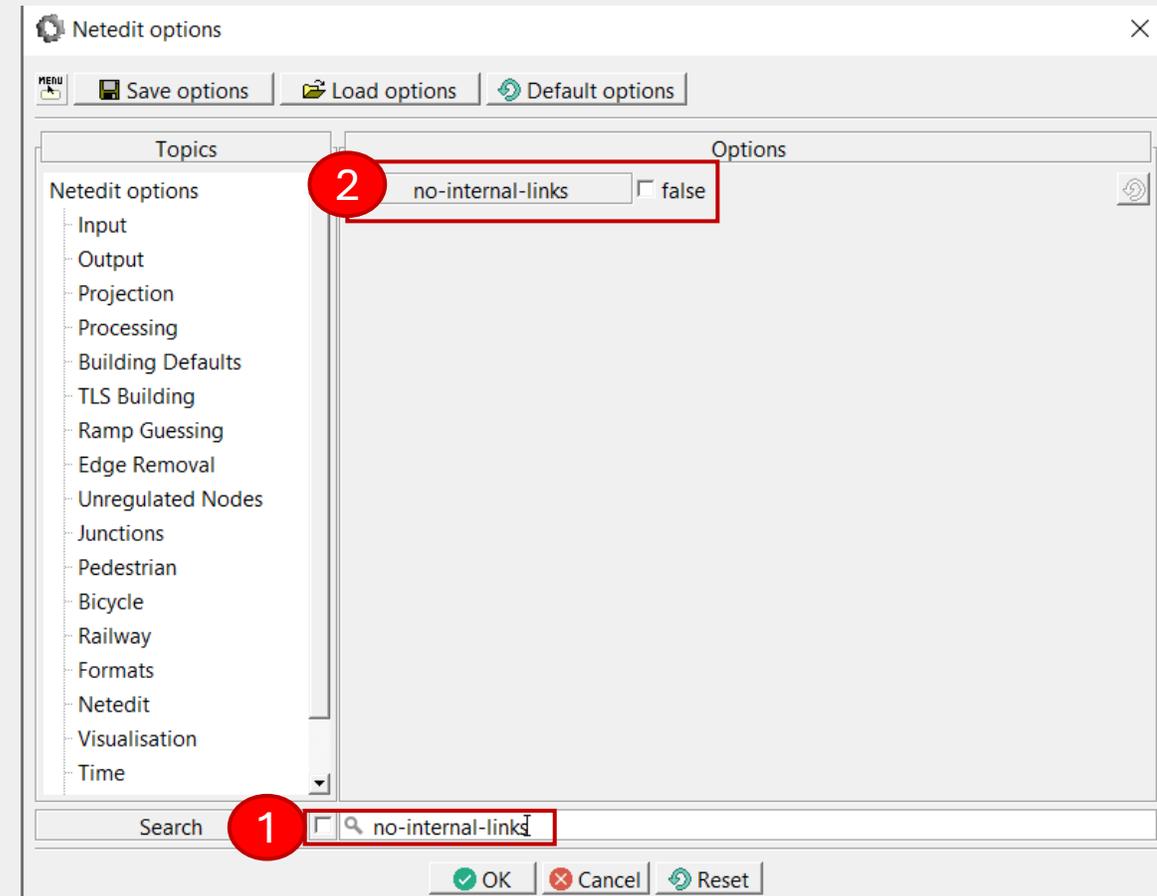
- F) Import SUMO Road Network
- G) Road Marking As Decals: Stamp an image on a 3D model
- H) Add Stop Signs, and Navigation Arrow
- I) Add Trees, Buildings, and Road Signs

# Step 1: Create Road Network

## A) Adding Lanes

❑ Open netedit → Processing → Option → Search “no-internal-links”

→ Make sure it is like the image

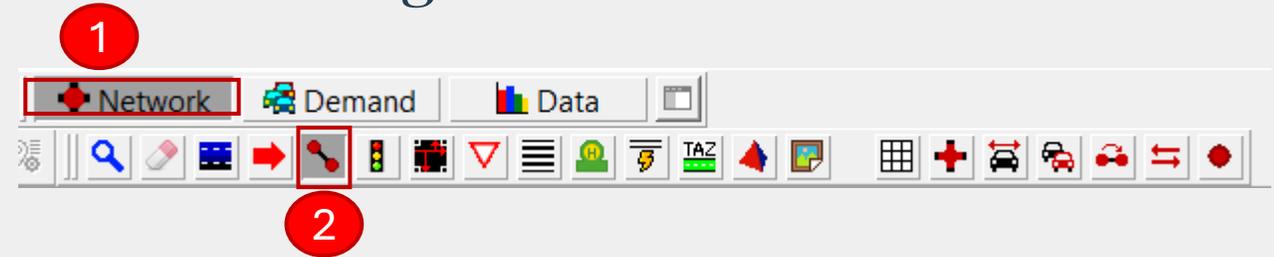


# Step 1: Create Road Network

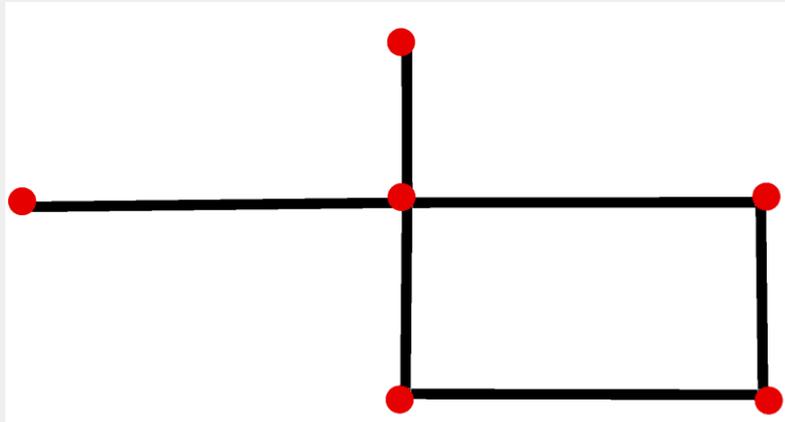
## A) Adding Lanes

❑ netedit → File → New Network

❑ UI → Network → Select “Creating Junction and Edges Tool”



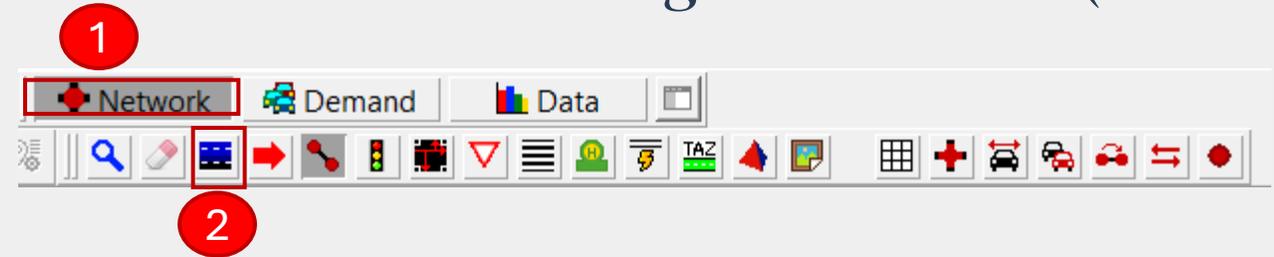
❑ Create a simple network like below



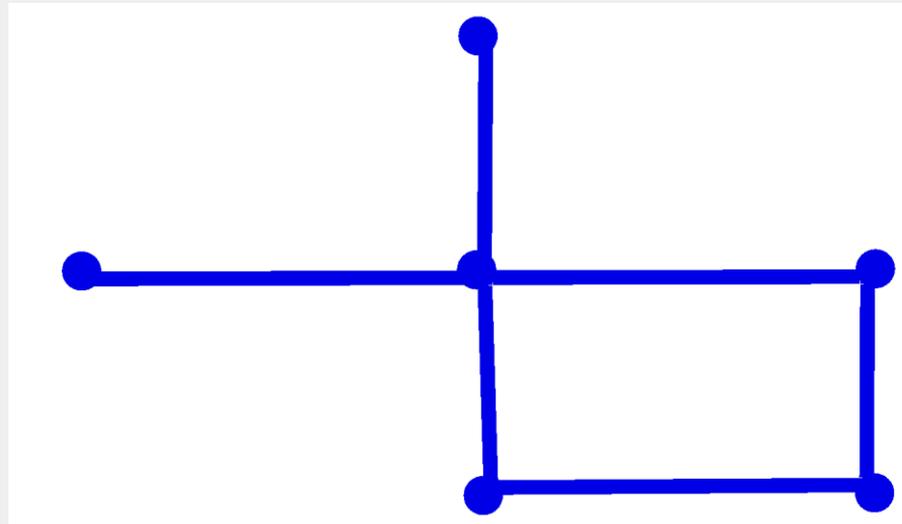
# Step 1: Create Road Network

## A) Adding Lanes

- UI → Network → Select “Selection Element” → Select All Edges and Nodes (roads & intersections)



- Result

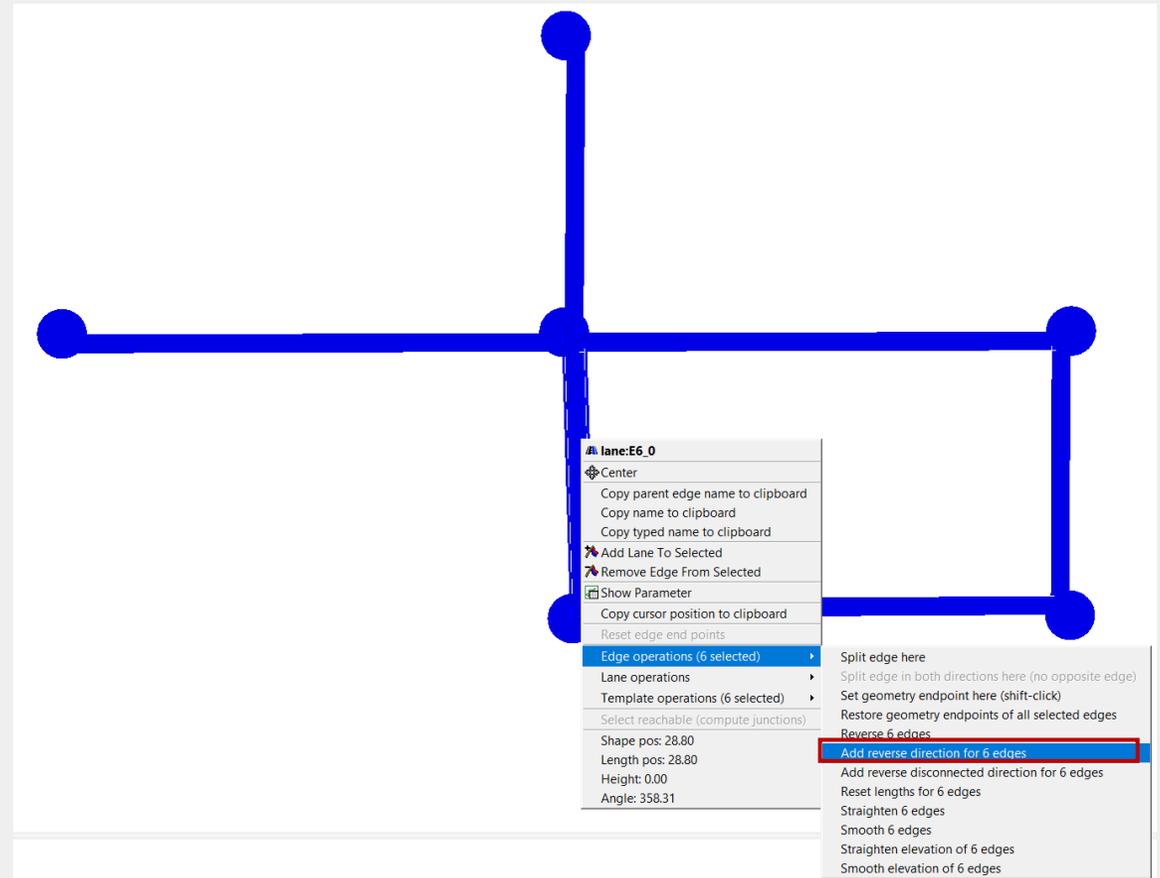


# Step 1: Create Road Network

## A) Adding Lanes

❑ Right Click On Roads → Add Reverse Direction for 6 Edges

❑ Processing → Compute Junctions



# Step 1: Create Road Network

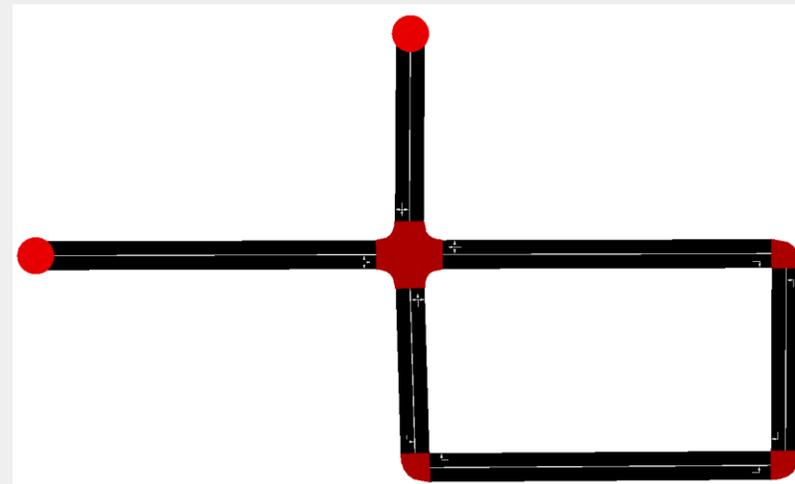
## A) Adding Lanes

- ❑ File → Save Network → Save in Folder “SUMOData” → Name it Sumo2Unity
- ❑ File → Netedit config → Save in Folder “SUMOData” → Name it Sumo2Unity
- ❑ File → Sumo config → Save in Folder “SUMOData” → Name it Sumo2Unity

❑ You should have the below three files

❑ Your network should look like this  
(processing → compute junction)

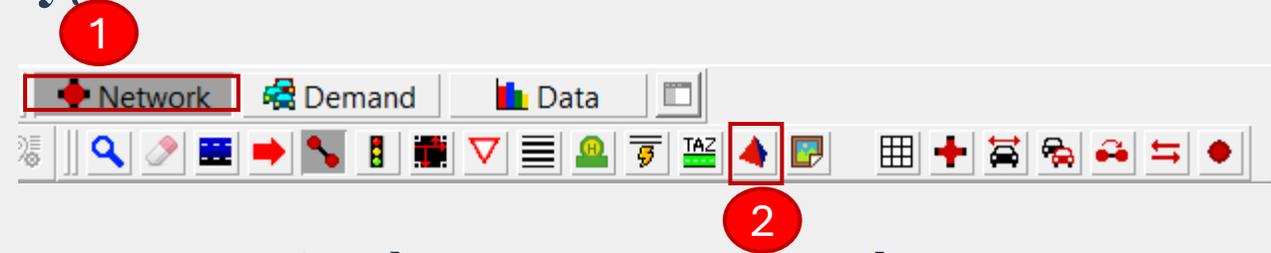
Results	2025-07-24 5:19 AM	File folder	
Sumo2Unity.net.xml	2025-07-24 12:42 PM	Microsoft Edge H...	13 KB
Sumo2Unity.netecfg	2025-07-24 12:44 PM	NETECFG File	1 KB
Sumo2unity.sumocfg	2025-07-24 12:45 PM	SUMO Configurati...	1 KB



# Step 1: Create Road Network

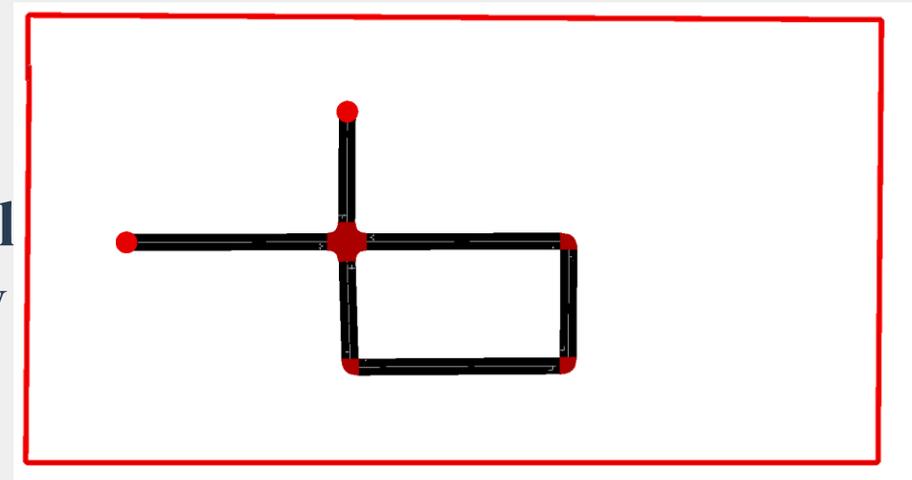
## B) Adding Terrain

□ UI → Network → Select “Creating Polygon”



□ Zoom out → in Shapes Window → Type: Terrain → Start Drawing → Create A rectangle that serves as “terrain” → Stop Drawing

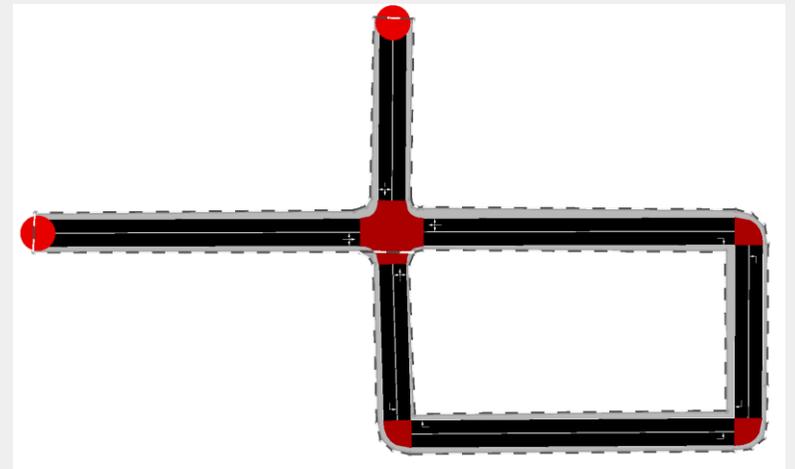
□ File → Additional and Shapes → Save Additional  
→ File Filter: xml files → name as Sumo2Unity.Poly



# Step 1: Create Road Network

## B) Adding Roadside

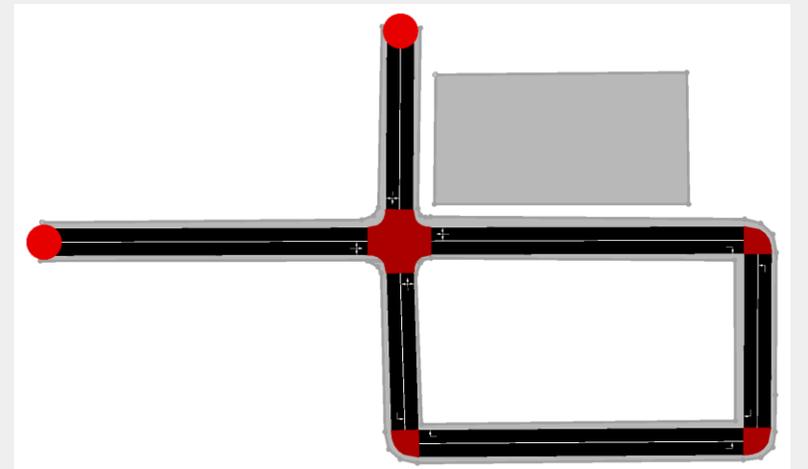
- ❑ UI → Network → Select “Creating Polygon”
- ❑ in Shapes Window → Type: Roadside → Start Drawing → Create A roadside area that serves as “roadside” → Stop Drawing
- ❑ in Shapes Window → Fill: true → color: 122,122,122 (grey)
- ❑ File → Additional and Shapes → Save Additional



# Step 1: Create Road Network

## B) Adding Residential

- ❑ UI → Network → Select “Creating Polygon”
- ❑ in Shapes Window → Type: Residential → Start Drawing → Create A rectangular that serves as “residential” area → Stop Drawing
- ❑ in Shapes Window → Fill: true → color: 122,122,122 (grey)
- ❑ File → Additional and Shapes → Save Additional





# Step 1: Create Road Network

## 1.2. Unity Steps

**Note: Rename the Existing Scene to Scenario1**

**F) Import SUMO Road Network**

**G) Road Marking As Decals (Stamp an image on a 3D model)**

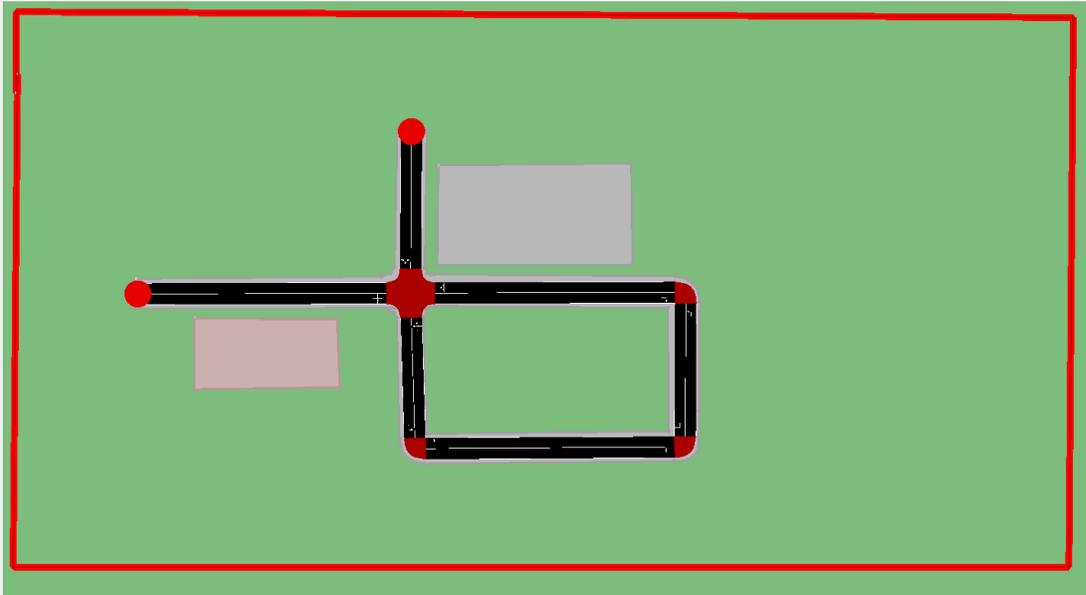
**H) Add Stop Signs, and Navigation Arrow**

**I) Add Trees, Buildings, and Road Signs**

# Step 1: Create Road Network

## F) Import SUMO Road Network

- ☐ Menu Bar → Sumo2Unity → 1. Create Road Network → Set SUMO Files Folder as Directory\SUMO2Unity\Scenario1 → Start



# Step 1: Create Road Network

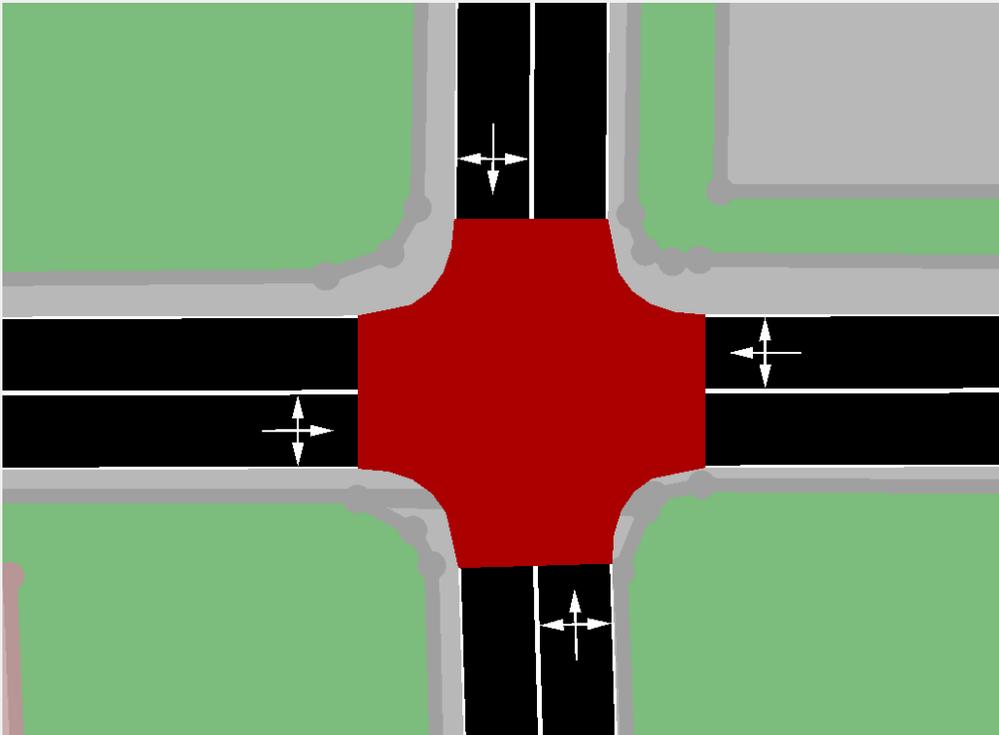
**G) Road Marking As Decals (Stamp an image on a 3D model)**



# Step 1: Create Road Network

## H) Add Stop Sign and Navigation Arrow

Hierarchy Window → Rendering → URP Decal Projector



# Step 1: Create Road Network

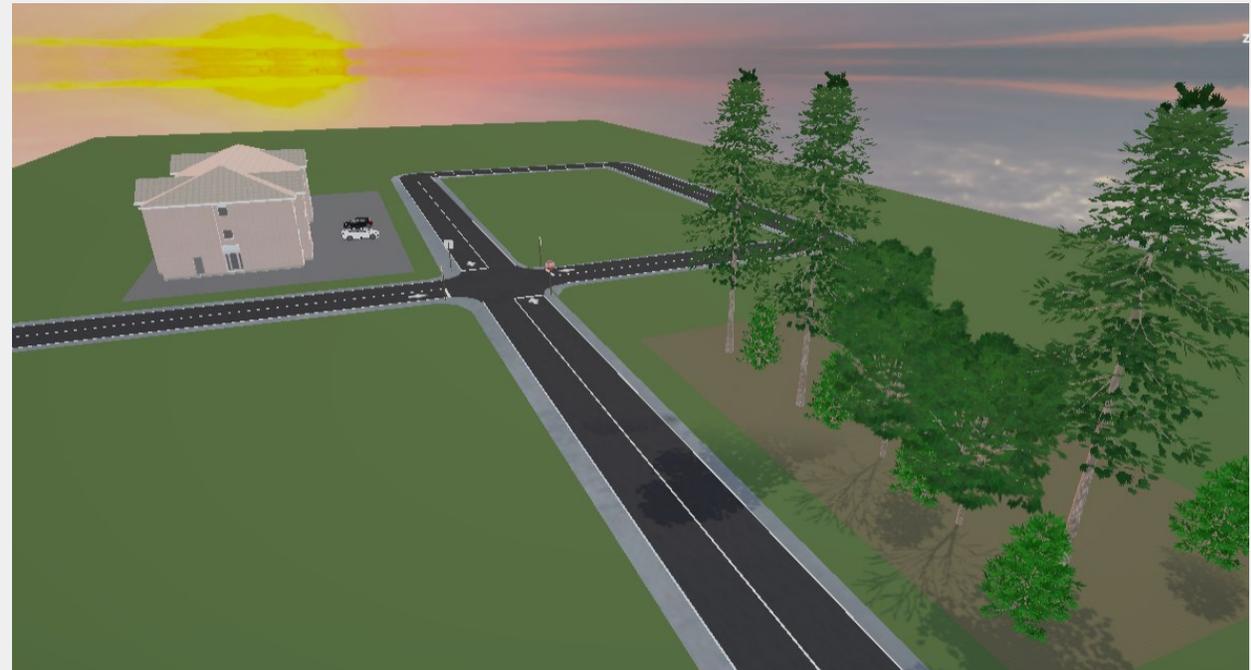
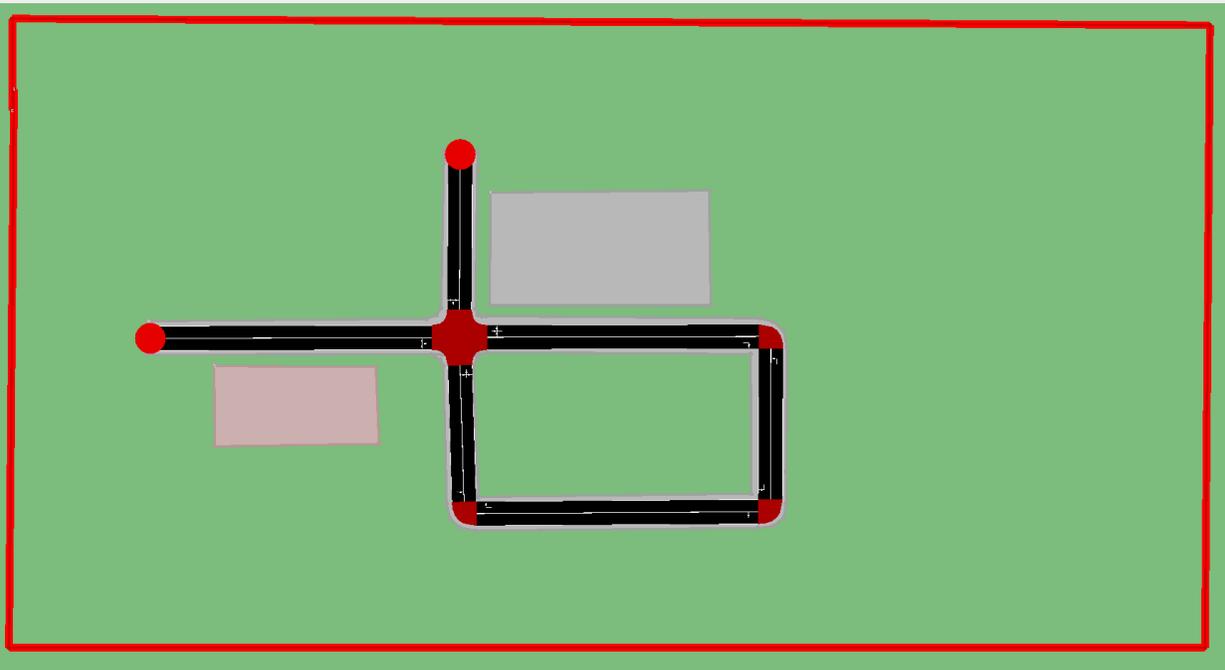
## H) Add Trees, Buildings, and Road Signs

- ❑ Project Window → Resources → Trees → Drag and Drop Some Trees in Wood Area
- ❑ Project Window → Resources → Buildings → Drag and Drop Some Buildings in Residential Area
- ❑ Project Window → Resources → Road Signs → Drag and Drop Some Road Signs in Road Signs Area



# Step 1: Create Road Network

□ Final Output



# Step 2: Run Sumo2Unity Integration

## 2.1. SUMO Steps

### A) Add Ego Vehicle:

A.1. Create Vehicle Type for EgoCar

A.2. Add Vehicle To Network

### B) Add Traffic Volume

B.1. Create Vehicle Types for Traffic Cars

B.2. Add Vehicles To Network

### C) Assign Ego Vehicle and Traffic Volume in Unity

### D) Prepare and Run Python Code (Sumo2Unity.py)

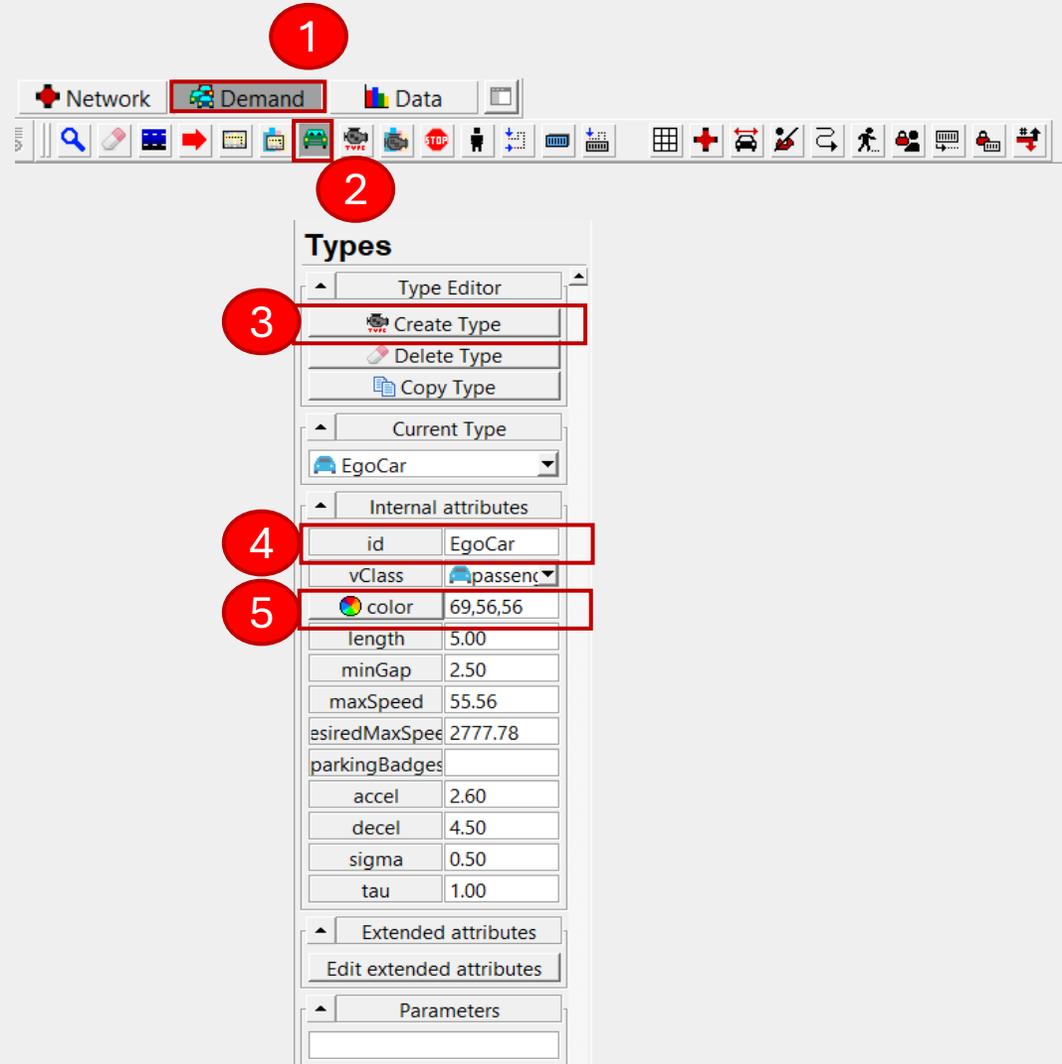
# Step 2: Run Sumo2Unity integration

## A) Add Ego Vehicle (A.1. Create Vehicle Type for EgoCar)

❑ UI → Demand → Select “Creating Vehicles”

❑ Create vehicle types EgoCar (Black) (69,56,56) –  
Follow Steps in the Image

❑ File → Demand Element → Save Demand Element  
→ Name it as Sumo2Unity



# Step 2: Run Sumo2Unity integration

## A) Add Ego Vehicle (A.2. Add Vehicle To Network)

❑ UI → Demand → Select “Creating Vehicles”

❑ Follow steps in Image

The image shows the SUMO software interface with the 'Demand' menu open. The 'Vehicles' panel on the left is configured with the following settings:

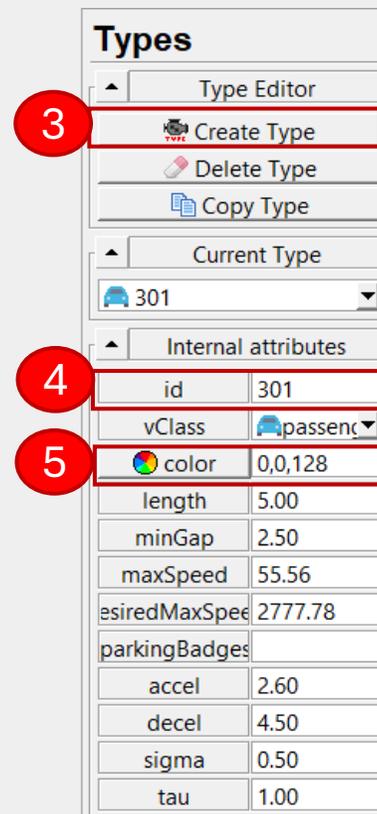
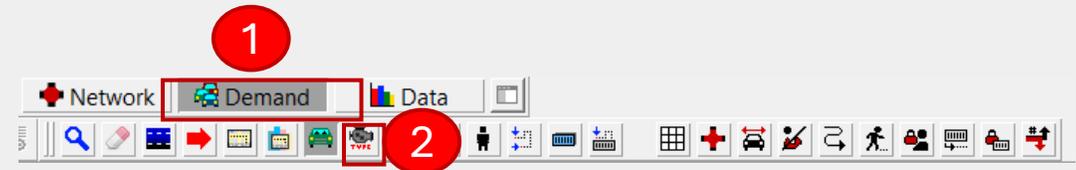
- trip (from-to edges)
- Parent vType: DEFAULT\_VEHTYPE
- Internal attributes:
  - id: f.0.0
  - color: 36,36,36
  - departLane: first
  - departPos: base
  - departSpeed: 0
  - arrivalLane: current
  - arrivalPos: max
  - arrivalSpeed: current
  - line:
  - personNumber: 0
  - containerNumber: 0
  - departPosLat: center
  - arrivalPosLat: center
  - insertionCheck: ~"
  - depart: 540
- Parameters: (empty)
- Netedit attributes:
  - route file: /nity.rou.xml
  - Help: (icon)
- Route creator:
  - Selected edges: 5
  - Path edges: 9
  - Length: 515.26
  - Average speed: 13.89
  - Finish route creation

The main window shows a network diagram with a red border. A purple vehicle is positioned on a horizontal edge. A red circle with the number 7 is placed on the vehicle. A red circle with the number 8 is placed on the 'Finish route creation' button.

# Step 2: Run Sumo2Unity integration

## B) Add Traffic Volume (B.1. Create Vehicle Types for Traffic Cars)

- ❑ UI → Demand → Select “Creating Vehicles”
- ❑ Create vehicle types 301 (blue), 302 (grey), 303(black), 304 (red), 305(gold), 306(white)
- ❑ See 301 (blue as an example)
- ❑ File → Demand Element → Save Demand Element  
→ Name it as Sumo2Unity

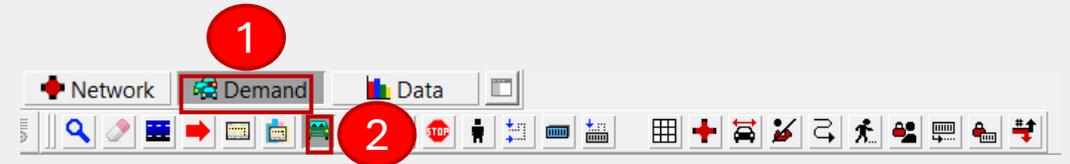


# Step 2: Run Sumo2Unity integration

## B) Add Traffic Volume (B.2. Add Vehicle To Network)

UI → Demand → Select “Creating Vehicles”

Follow Steps



**Vehicles**

3  flow (from-to edges)

4  301

5

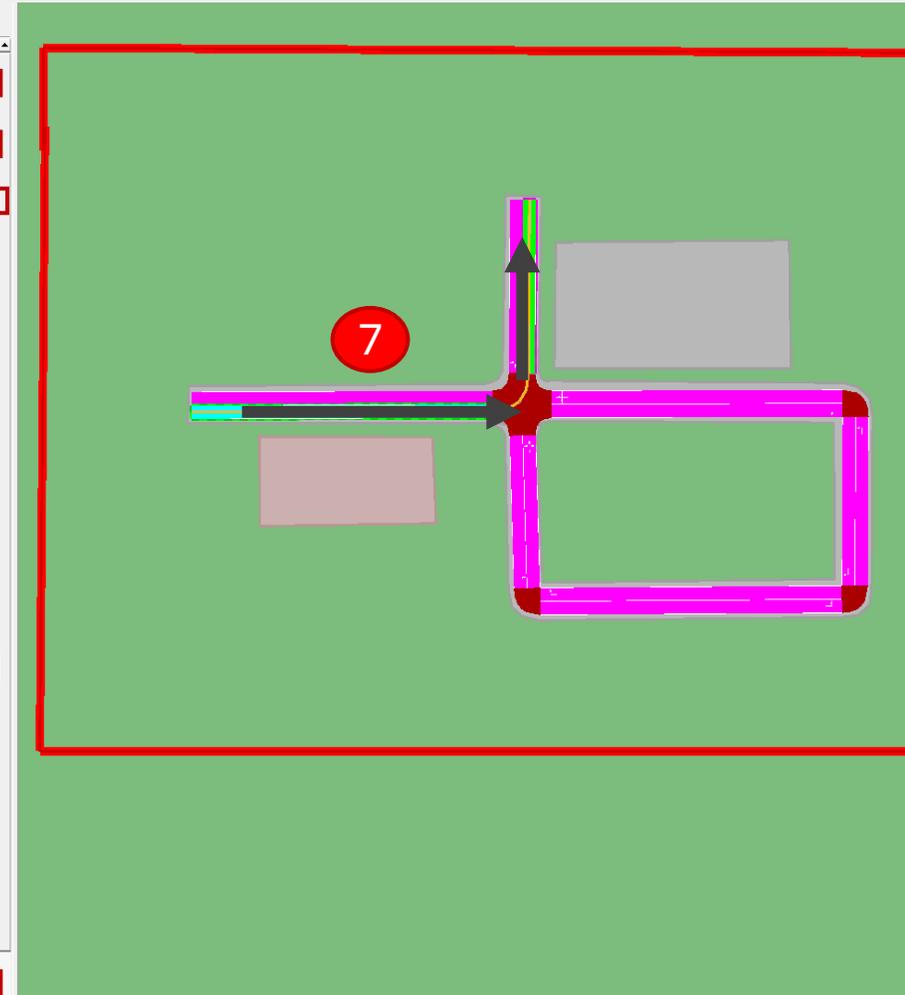
id	f_1
color	yellow
departLane	first
departPos	base
departSpeed	0
arrivalLane	current
arrivalPos	max
arrivalSpeed	current
line	
personNumber	0
containerNumber	0
departPosLat	center
arrivalPosLat	center
insertionChecks	all
begin	0.00

6

Flow attributes	
terminate	end
spacing	vsPerHour
end	3600.00
vehsPerHour	200

8

Finish route creation



# Step 2: Run Sumo2Unity Integration

## B) Add Traffic Volume (B.2. Add Vehicle To Network)

Do this for 302

The screenshot shows the NetEdit software interface. The top toolbar has a red circle '1' over the 'Demand' button. The 'Vehicles' panel on the left has several elements highlighted with red circles and boxes:

- 3: 'flow (from-to edges)' dropdown menu
- 4: '302' dropdown menu for 'Parent vType'
- 5: 'id' field with value 'f\_2'
- 6: 'vehsPerHour' field with value '200'
- 8: 'Finish route creation' button

The 2D network diagram on the right shows a green background with a network of edges. A red circle '7' is placed over a grey rectangular area in the upper right quadrant of the network.

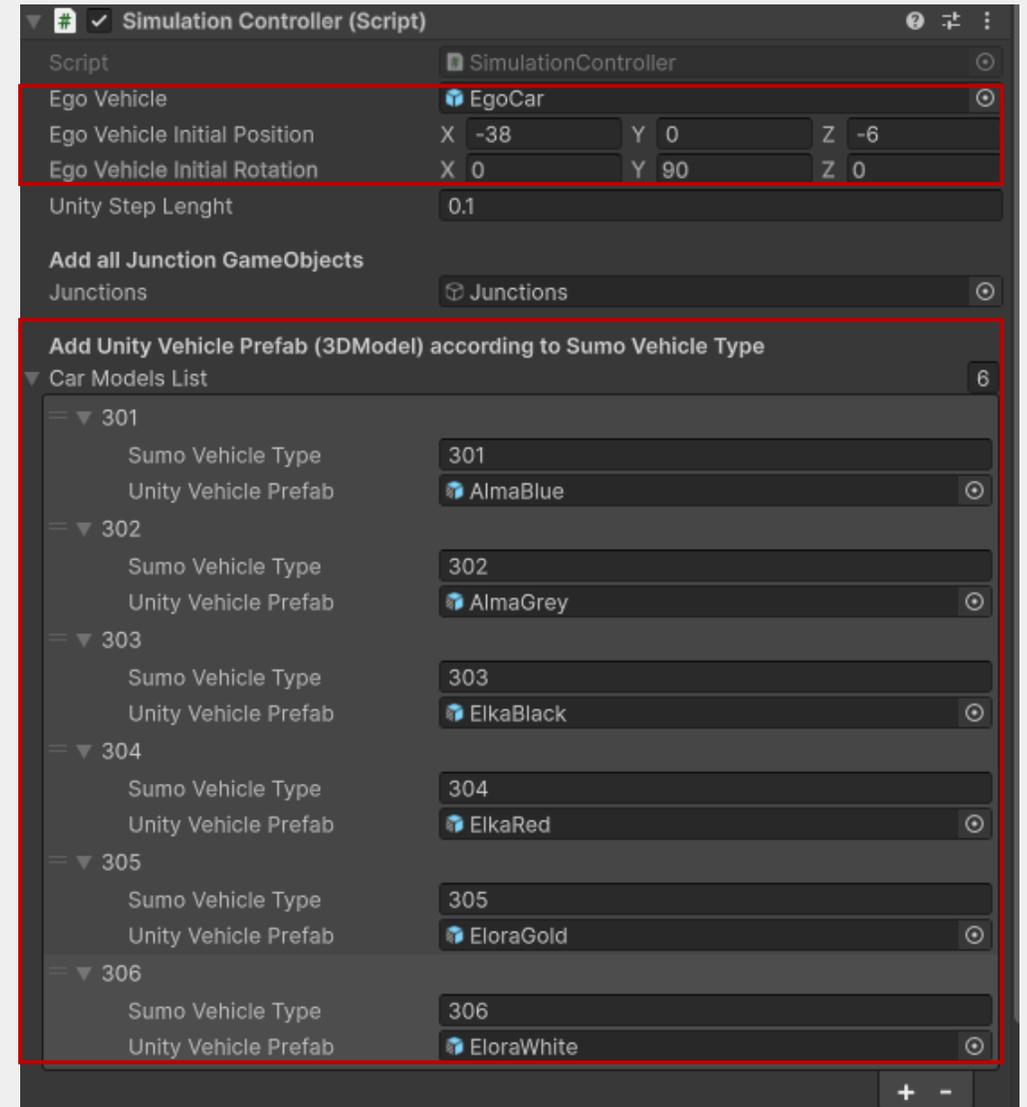
# Step 2: Run Sumo2Unity integration

## 2.1. Unity Steps

- C) Assign Ego Vehicle and Traffic Volume in Unity
- D) Prepare and Run Python Code (Sumo2Unity.py)

# Step 2: Run Sumo2Unity Integration

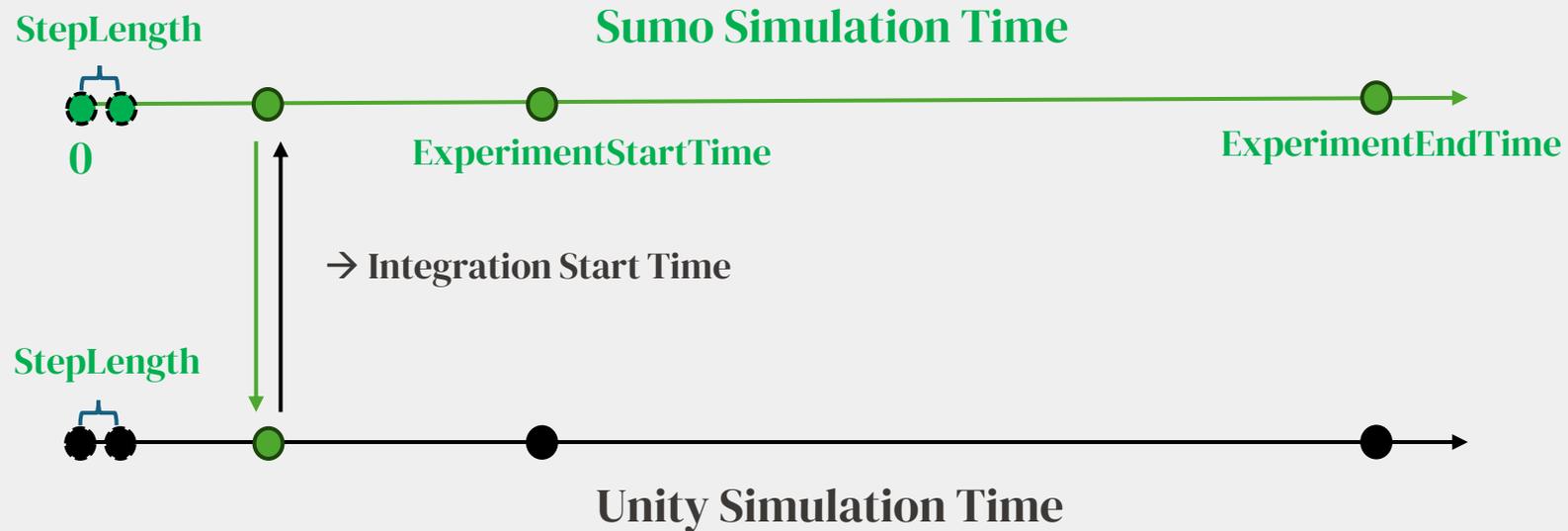
## C) Assign Ego Vehicle and Traffic Volume in Unity



# Step 2: Run Sumo2Unity integration

## D) Prepare and Run Python Code (Sumo2Unity.py)

```
15 #Initial Variables
16 IntegrationStartTime = 540 #
17 ExperimentStartTime = 600 #E
18 ExperimentEndTime = 720 #Exp
19 steplength = 0.1 #Sumo step
20
```



# Step 2: Run Sumo2Unity Integration

## D) Prepare and Run Python Code (Sumo2Unity.py)

- ❑ **IntegrationStartTime:** The time when both SUMO and Unity run.
- ❑ **ExperimentStartTime:** Normally 10 min (600 Seconds) for the simulation to run before putting ego vehicle into simulation. This is called warm-up period.
- ❑ **ExperimentEndTime:** The time you want to put the participant in the simulation, for example, if your experiment is 2 min, then end time is  $600 + 120 \text{ second} = 780 \text{ seconds}$
- ❑ **Step length:** is data exchange rate between SUMO and Unity. Default value is 0.1 second. Lower value means more exchanging, and higher accuracy, but it takes a lot of resources. This value should be always equal to Unity Step Length in Unity in Simulation Controller inspector.

# Step 2: Run Sumo2Unity Integration

## D) Prepare and Run Python Code (Sumo2Unity.py)

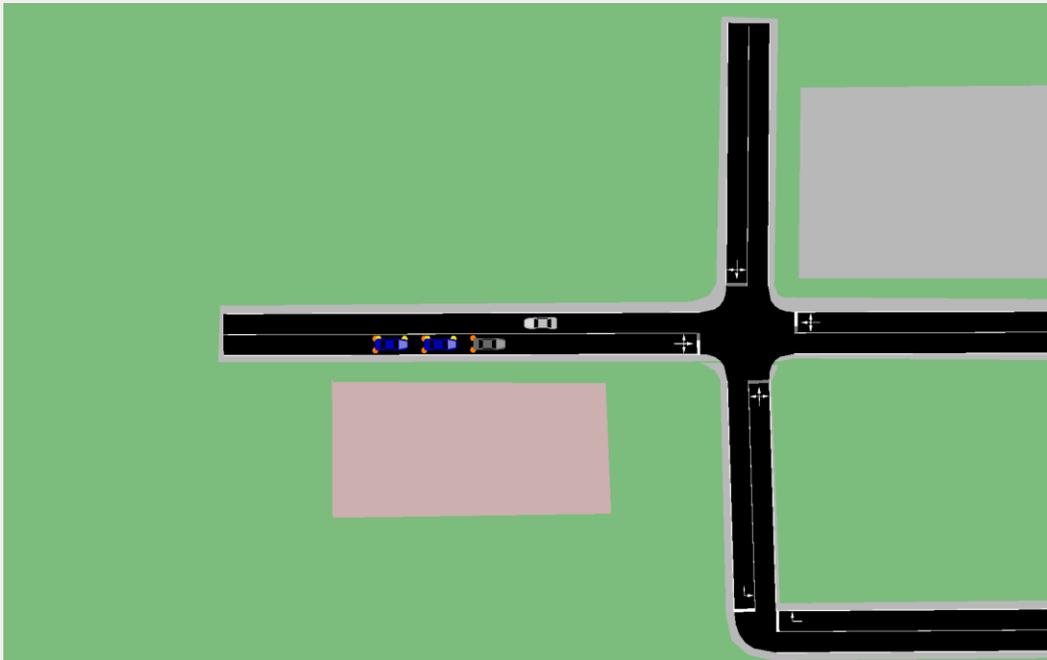
- ❑ Make sure you have below files including Sumo2Unity.sumofg in the proper folder “SUMO2Unity\Scenario1”

 Sumo2Unity.net.xml	2025-07-23 7:24 AM	Microsoft Edge HT...	10 KB
 Sumo2Unity.netecfg	2025-07-23 7:48 AM	NETECFG File	2 KB
 Sumo2Unity.Poly.xml	2025-07-23 7:48 AM	Microsoft Edge HT...	2 KB
 Sumo2Unity.py	2025-07-23 8:23 AM	Python Source File	28 KB
 Sumo2Unity.rou.xml	2025-07-23 7:18 AM	Microsoft Edge HT...	1 KB
 Sumo2Unity.sumocfg	2025-07-23 7:48 AM	SUMO Configurati...	1 KB

# Step 2: Run Sumo2Unity integration

## D) Prepare and Run Python Code (Sumo2Unity.py)

- Run Python
- When it reaches second 540, ego car is added to SUMO, then Run Unity.



# Step 3: Generate Performance Functions

- ❑ **RTF (Real time Factor):**  $\text{Sumo simulation time} / \text{Unity simulation time}$ : Always should be 1
- ❑ **FPS (Frame Rate per Second):** number of frames (images render in each second):  
Recommended more than 90

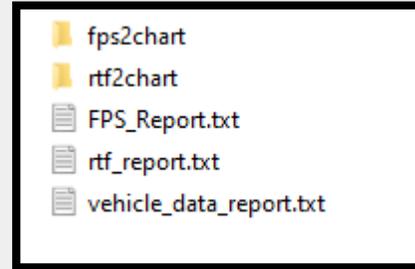
Conduct the experiment multiple times and generate RTF  
and FPS before inviting people.

# Python Installation and Dependencies

1. **Visual Studio Code → Download Python Extension**
2. **Install Python 3.12 (Recommended)**
3. **During install check “Add Python to PATH” and choose “Install for all users”**
4. **Cmd → `python -version`**
5. **`pip list` → To show all Python dependencies**
6. **`pip install matplotlib`**

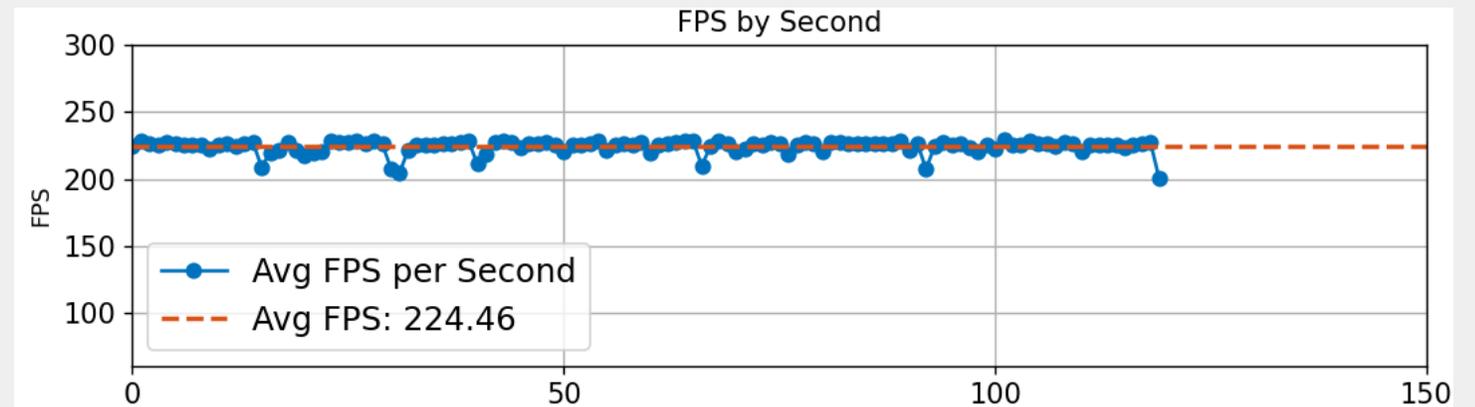
# Step 3: Generate Performance Functions

□ Folder Results →



□ Copy and Paste FPS\_Report.txt → Folder “fps2chart” → Replace with “FPS\_Report.txt”

□ Run fps2chart.py →



# Step 3: Generate Performance Functions

❑ Copy and Paste `rtf_report.txt` → Folder “`rtf2chart`” → Replace with “`rtf_report.txt`”

❑ Run `rtf2chart.py` →

