Tamás Tettamanti:

Introduction to SUMO microscopic road traffic simulator

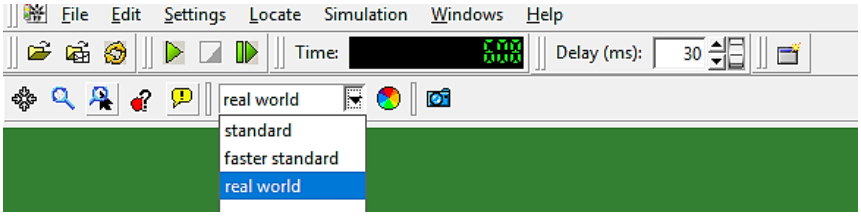
- SUMO: **S**imulation of **U**rban **MO**bility by Institute of Transportation Systems
- free and open traffic simulation suite since 2001.
- modeling of intermodal traffic systems:
  - including road vehicles,
  - public transport and pedestrians.
- Tools:
  - route finding,
  - visualization,
  - network import,
  - emission calculation.
- SUMO can be enhanced with custom models and provides various APIs to remotely control the simulation.
- https://www.eclipse.org/sumo/about/

Download and install – important tricks

- https://sumo.dlr.de/wiki/Simulation_of_Urban_MObility_-_Wiki
- **Install to the root (not to C:\Program Files\Sumo\), but to C:\Sumo\ !**
- Install Python (latest is good)! https://www.python.org/downloads/

SUMO GUI

- Use FreeCommander for easy file handling and text file operation.
- Start from here: C:\Sumo\bin\sumo-gui.exe
- Open simulation/ -> C:\Sumo\doc\examples\sumo\simple_nets\cross\



Demo simulation – OSM WebWizard

https://sumo.dlr.de/wiki/Tutorials/OSMWebWizard

- Command Prompt (CMD) start
- Navigate to Sumo/tools folder:
  >>cd C:\Sumo\tools
  >> osmWebWizard.py
  To break command running in CMD → Ctrl+C (same in Matlab)
- Show the alternative: Open Command Line frim FreeCommander: Ctrl+D
- Az osmWebWizard.py runs a Python program which opens the browser with OSM map in order to generate SUMO scenario,

- OSM Web Wizard randomly chooses a departure and arrival edge for vehicles.
- *Through Traffic Factor*: A value between [0.5 100]. A big value implies that many vehicles depart and arrive at the boundary of the simulation area, which correspond to a scenario with a lot of through traffic.
- *Count*: A value between [0.2 100]. How many vehicles are generated **per hour and per kilometer**.

- Command *"Generate Scenario"* creates the simulation files and places into the folder: C:\Sumo\tools\<today's date and time>
- Advantage of OSM Web Wizard: no need to edit/convert OSM map file with extension ***.osm**; it simply downloads a scenario together with traffic loads.
- The downloaded SUMO project then can be easily further elaborated.
- Start from NetEdit C:/SUMO/…/bin folder → Edit/Open → **C:\Sumo\doc\examples\sumo\busses**
- Show a few tools of NetEdit, e.g. Inspect, Traffic Lights
- You can open NetEdit in SUMO GUI: Edit/Open in NetEdit

What is XML?

- XML = Extensible Markup Language
- XML is not a programming language! However, it can be used by any programming language!
- XML allows authors to indicate **what the content is** rather than **how the content looks like**.
- XML does this using tags:
  start-tag: `<section>`
  end-tag: `</section>`
- Example XML:

```
<book>
<title>1984</title>
<author>George Orwell</author>
<summary>
This story remains eternally fresh and reflects societal trends of today.
</summary>
</book>
```

- XML defines data clearly and simply → it means the same for everybody!
- It is up to the user, what to do with this content, e.g. represent on a website using with bold Arial font 12.

First SUMO simulation - "Hello Sumo"

https://sumo.dlr.de/wiki/Tutorials/Hello_Sumo
Street network consists of:

- nodes (junctions)
- edges (streets connecting the junctions).

Nodes: Location (x- and y-coordinate, describing distance to the origin in meters) and ID.

```
<nodes>
    <node id="1" x="0.0" y="0.0" />
    <node id="2" x="+200.0" y="0.0" />
    <node id="3" x="+400.0" y="0.0" />
</nodes>
```

By using Notepad++, we save the above code as hello.nod.xml where **.nod.xml** is the default suffix for Sumo node files.
Save it to C:\<myname>\HelloSumo folder!

Edges: Now we are connecting the nodes with edges. Edges are directed: each vehicle enters an edge at the node given as **from** and ends it at the node given as **to**.

```
<edges>
    <edge id="1to2" from="1" to="2" />
```

```
    <edge id="2to3" from="2" to="3" />
    <edge id="3to2" from="3" to="2" />
    <edge id="2to1" from="2" to="1" />
</edges>
```

Save it as `hello.edg.xml` into C:\<myname>\HelloSumo folder!

Now we have nodes and edges: we can call the first SUMO tool "NETCONVERT" to create a network.

Navigate to C:\<myname>\HelloSumo folder in FreeCommander and start CMD (Ctrl+D), then run:

```
netconvert --node-files hello.nod.xml --edge-files hello.edg.xml --output-file
hello.net.xml
```

This will generate our network called `hello.net.xml`!

Look in the hello.net.xml → explain the code.

Open the C:\Sumo\bin\netedit.exe GUI program and open hello.net.xml!

Routes: Create `hello.rou.xml` file!

```
<routes>
    <route id="route1" edges="1to2 2to3"/>
    <route id="route2" edges="3to2 2to1"/>
    <flow id="flow1" color="0,100,200" begin="0" end= "3600" vehsPerHour="800"
route="route1"/>
    <flow id="flow2" color="500,0,0" begin="0" end= "3600" vehsPerHour="800"
route="route2" />
</routes>
```

Configuration: Now we glue everything together into a configuration file ***.sumocfg** !

```
<configuration>
    <input>
        <net-file value="hello.net.xml"/>
        <route-files value="hello.rou.xml"/>
    </input>
    <time>
        <begin value="0"/>
        <end value="3600"/>
    </time>
</configuration>
```

Save this to `hello.sumocfg` !

Start the simulation in CMD by either

- without GUI: `sumo hello.sumocfg`
- or with GUI by: `sumo-gui hello.sumocfg`
- if you also want to make start: `sumo-gui -c hello.sumocfg --start`

Routes settings 2nd time: Recreate `hello.rou.xml` file for a single vehicle!

```
<!--
    <flow id="flow1" ...
    <flow id="flow2" ...
-->
```

```
<routes>
   <vType id="type1" accel="0.8" decel="4.5" length="5" maxSpeed="13.9"  />
   <vehicle id="1" type="type1" route="route1" depart="60"/>
</routes>
```

Rerun the GUI simulation with the modified hello.rou.xml file!

Delay setting by code:

Create a new text file with content:

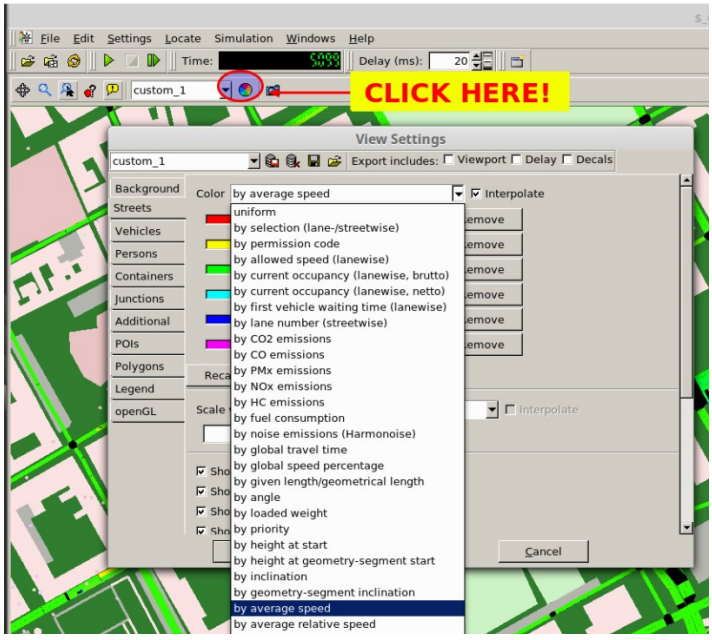
```
<viewsettings>
      <delay value="45"/>
</viewsettings>
```

Save it as gui-settings.cfg!

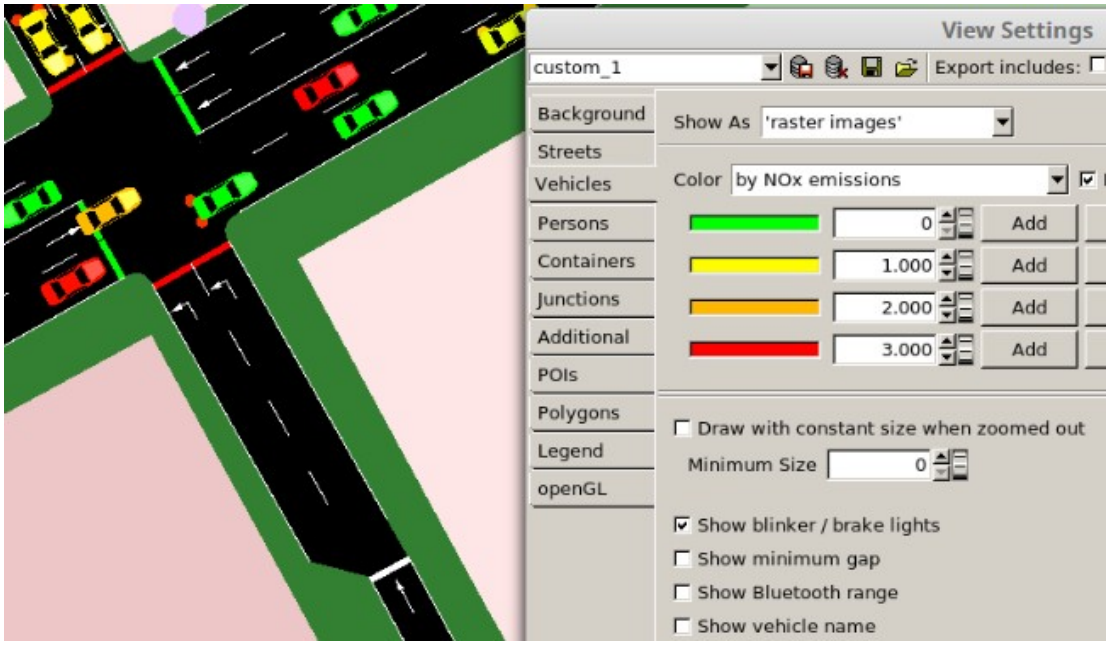
Add a plus line to hello.sumocfg file's body:

```
<gui-settings-file value="gui-settings.cfg"/>
```

View Settings for visualization



For example: colored cars by NOx emission:



Homework: "Driving in Circles"

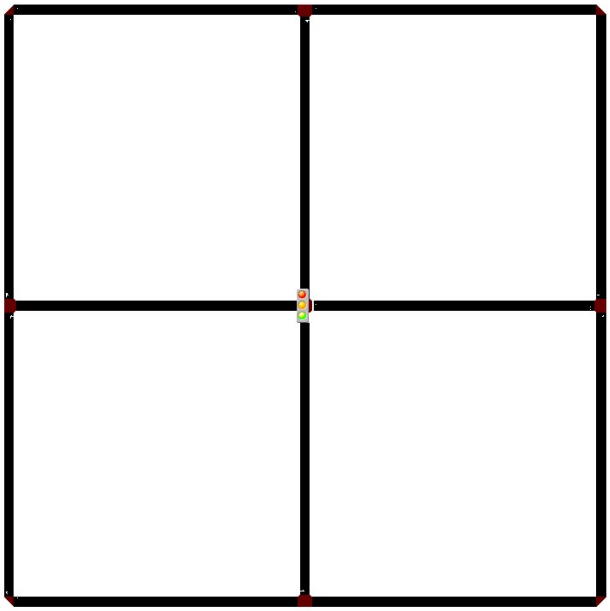
https://sumo.dlr.de/wiki/Tutorials/Driving_in_Circles

Grid network generation, O-D matrix, random trips, static/actuated traffic lights

Use the following command for arbitrary grid network generation:
`netgenerate --grid=true --grid.number=3 --grid.length=200 --output-file=mynetwork.net.xml`

Help: https://sumo.dlr.de/wiki/Networks/Abstract_Network_Generation#Grid-like_Networks
Option list for NETGENERATE function:
https://sumo.dlr.de/wiki/NETGENERATE#Tls_Building

- Open the mynetwork.net.xml file in Netedit
- O-D matrix generation per each second from 0 to 3599 sec:
  `randomTrips.py -n mynetwork.net.xml -e 3600`
  ➔ this creates a trips.trip.xml file!
- Traffic assignment (route creation among O-D pairs)
  https://sumo.dlr.de/wiki/Demand/Shortest_or_Optimal_Path_Routing#Usage_Examples
  `duarouter --trip-files trips.trips.xml --net-file mynetwork.net.xml --output-file mynetwork.rou.xml`
- Create a Traffic Lights (TLS) for the center junction (nr. "1/1"):



- Save it and open the mynetwork.net.xml file → look for “tlLogic”…

  Example:

```
<tlLogic id="2/2" type="static" programID="0" offset="30">
    <phase duration="5" state="rrrrrrrrrrr"/>
    <!-- intergreen time -->
    <phase duration="2" state="uuurrruuurrr"/>
    <!-- u=yellow+green -->
    <phase duration="20" state="GGgrrrGGgrrr"/>
    <!-- G=green of direction with priority, g=green with no priority -->
   <phase duration="3" state="yyyrrryyyrrr"/>
    <phase duration="5" state="rrrrrrrrrrr"/>
    <!-- intergreen time -->
    <phase duration="2" state="rrruuurrruuu"/>
    <phase duration="20" state="rrrGGgrrrGGg"/>
  <phase duration="3" state="rrryyyrrryyy"/>
</tlLogic>
```

- Actuated traffic light: use the TLS as follows within the .net.xml

  https://sumo.dlr.de/wiki/Simulation/Traffic_Lights#Based_on_Time_Gaps

  Example:

```
<tlLogic id="1/2" programID="gap-based_1/2" type="actuated" offset="0">
  <param key="max-gap" value="3.1"/>
  <param key="detector-gap" value="10"/>
  <param key="file" value="out_12.xml"/>
  <param key="freq" value="60"/>
  <phase duration="5" state="rrrrrrrrrrr"/>
<!-- intergreen -->
  <phase duration="2" state="uuurrruuurrr"/>
<!-- u=yellow+green -->
  <phase duration="20" minDur="5" maxDur="30" state="GGgrrrGGgrrr"/>
<!-- G=green with priority, g=green without priority -->
  <phase duration="3" state="yyyrrryyyrrr"/>
  <phase duration="5" state="rrrrrrrrrrr"/>
<!-- intergreen -->
  <phase duration="2" state="rrruuurrruuu"/>
  <phase duration="20" minDur="5" maxDur="30" state="rrrGGgrrrGGg"/>
  <phase duration="3" state="rrryyyrrryyy"/>
</tlLogic>
```