Some notes about Sumo:

1) Installed it on Windows.

2) Then I defined an environment variable on Windows prompt (cmd):

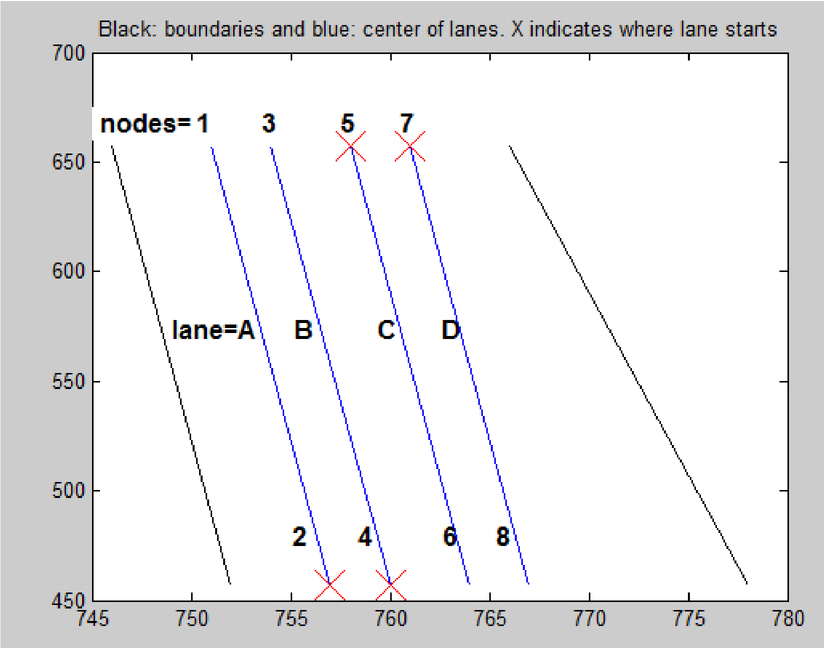
set SUMO_HOME="C:\Program Files (x86)\DLR\Sumo\"

3) Read and followed the Tutorial:

http://sumo.dlr.de/wiki/Tutorials/Hello_Sumo

4) Started the following project from scratch, creating the ita.nod.xml

```
<nodes>
 <node id="1" x="751" y="657" />
 <node id="2" x="757" y="457" />
 <node id="3" x="754" y="657" />
 <node id="4" x="760" y="457" />
 <node id="5" x="758" y="657" />
 <node id="6" x="764" y="457" />
 <node id="7" x="761" y="657" />
 <node id="8" x="767" y="457" />
</nodes>
```

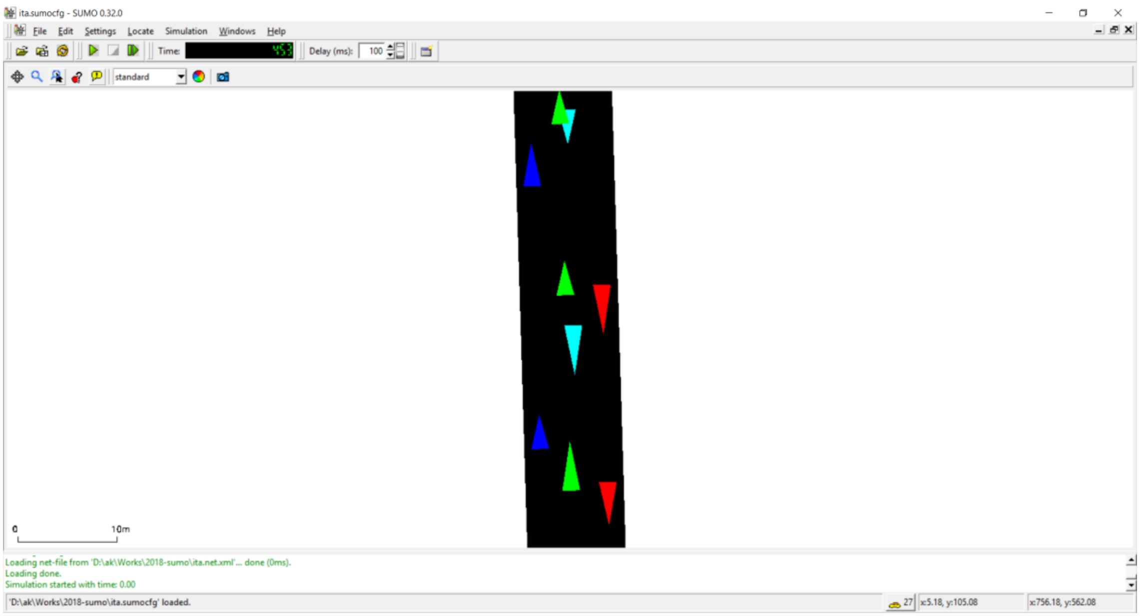


Created node and edge files, and then:

D:\ak\Works\2018-sumo>"C:\Program Files (x86)\DLR\Sumo\bin\netconvert" --node-files=ita.nod.xml --edge-files=ita.edg.xml --output-file=ita.net.xml

Which said: Success.

If I read it using the Sumo's net editor gives for the four lanes:



When using the GUI I had to add a delay of 70 ms otherwise the cars would pass too fast.

Now started defining other types of cars:
http://sumo.dlr.de/wiki/Definition_of_Vehicles,_Vehicle_Types,_and_Routes

It is important to study the "Speed Distributions":

Caution:
Using speed distributions is highly advisable to achieve realistic car following behaviour.

I decreases the sampling interval from 1 to 0.5 using –steplength:

D:\ak\Works\2018-sumo>"C:\Program Files (x86)\DLR\Sumo\bin\sumo.exe" -c ita.sumocfg --fcd-output ak.txt --step-length 0.5

I am not going to read it, but can define the distribution of vehicle types using http://sumo.dlr.de/wiki/Definition_of_Vehicles,_Vehicle_Types,_and_Routes#Route_and_vehicle_type_distributions. Maybe more than we need is the Python tool below:

Note:
The python tool createVehTypeDistributions.py can be used to generate large distributions that vary multiple *vType* parameters independently of each other.

I guess we need to use such scripts to simulate jam, putting several vehicles in the street.

I will try to use the following to control how many cars are in the lane. When I use probability=1 there are several (around 7) cars while probability close to 0 leads to few.

But it is not easy to generate a jam, very congested scenario:

http://sumo.dlr.de/wiki/FAQ#How_do_I_get_high_flows.2Fvehicle_densities.3F

Flows with a random number of vehicles

Both DUAROUTER and SUMO support loading of `<flow>` elements with attribute `probability`. When this attribute is used (instead of `vehsPerHour`,`number` or `period`), a vehicle will be emitted randomly with the given probability each second. This results in a binomially distributed flow (which approximates a Poisson Distribution for small probabilities. When modeling such a flow on a multi-lane road it is recommended to define a `<flow>` for each individual lane.

There are many other tricks described in
http://sumo.dlr.de/wiki/TraCI/Vehicle_Value_Retrieval

To obtain the position of each car, it is possible to use
http://sumo.dlr.de/wiki/Simulation/Output#Introduction
fcd output: Floating Car Data includes name, position, angle and type for every vehicle

If we use Python, more information is provided at
http://sumo.dlr.de/wiki/TraCI/Vehicle_Value_Retrieval

The sampling period can be controlled by doing a simulation with very small sampling period and later throwing away (eliminating) some vehicles using
http://sumo.dlr.de/wiki/Tools/TraceExporter

Processing Options

Several options allow to fine-tune the processing.

The output file for

D:\ak\Works\2018-sumo>"C:\Program Files (x86)\DLR\Sumo\bin\sumo.exe" -c ita.sumocfg --fcd-output ak.txt --step-length 0.5

Has the positions we need:

<vehicle id="laneADeterministic.28" x="19.69" y="4.70" angle="357.46" type="Car" speed="1.25" pos="4.64" lane="laneA_0" slope="0.00"/>

<vehicle id="typeB.21" x="5.87" y="5.04" angle="177.46" type="Car" speed="14.16" pos="170.06" lane="laneB_0" slope="0.00"/>

<vehicle id="typeB.22" x="4.20" y="42.87" angle="177.46" type="Bus" speed="14.57" pos="132.19" lane="laneB_0" slope="0.00"/>

<vehicle id="typeB.23" x="3.08" y="68.06" angle="177.46" type="Car" speed="12.79" pos="106.98" lane="laneB_0" slope="0.00"/>

<vehicle id="typeB.24" x="2.16" y="89.00" angle="177.46" type="Car" speed="13.88" pos="86.01" lane="laneB_0" slope="0.00"/>

<vehicle id="typeB.25" x="0.83" y="118.90" angle="177.46" type="Car" speed="14.56" pos="56.09" lane="laneB_0" slope="0.00"/>

<vehicle id="typeB.26" x="-0.31" y="144.67" angle="177.46" type="Car" speed="10.55" pos="30.29" lane="laneB_0" slope="0.00"/>

```
        <vehicle id="typeB.27" x="-1.08" y="161.99" angle="177.46" type="Car" speed="5.92"
pos="12.95" lane="laneB_0" slope="0.00"/>

        <vehicle id="typeB.28" x="-1.43" y="169.94" angle="177.46" type="Truck" speed="0.97"
pos="5.00" lane="laneB_0" slope="0.00"/>

    </timestep>

    <timestep time="58.50">

        <vehicle id="laneADeterministic.21" x="12.80" y="160.40" angle="357.46" type="Car"
speed="12.22" pos="160.49" la
```

Obs: I tried to merge below a deterministic and a probabilistic flow to simulate jam, but did not work.

```
<routes>

    <vTypeDistribution id="typedist1">

        <vType id="Car" departSpeed="max" accel="2.6" decel="4.5" length="3.91"
maxSpeed="30.0" speedDev="0.1" sigma="0.2" minGap="0.3" probability="0.6"/>

        <vType id="Truck" accel="2.0" decel="4" length="4.41" maxSpeed="25.0" speedDev="0.1"
sigma="0.2" minGap="0.3" probability="0.2"/>

        <vType id="Bus" accel="2.0" decel="4" length="5" maxSpeed="20.0" speedDev="0.1"
sigma="0.2" minGap="0.3" probability="0.2"/>

    </vTypeDistribution>

    <flow id="laneAProbabilistic" color="1,0,0" begin="0" end= "3000" probability="0.99"
type="typedist1">

        <route edges="laneA"/>

    </flow>

    <flow id="laneADeterministic" color="0,0,1" begin="0" end= "3000"
vehsPerHour="2000000" type="typedist1">

        <route edges="laneA"/>

    </flow>

    <flow id="typeB" color="0,1,0"  begin="0" end= "3000" probability="0.95" type="typedist1">
```

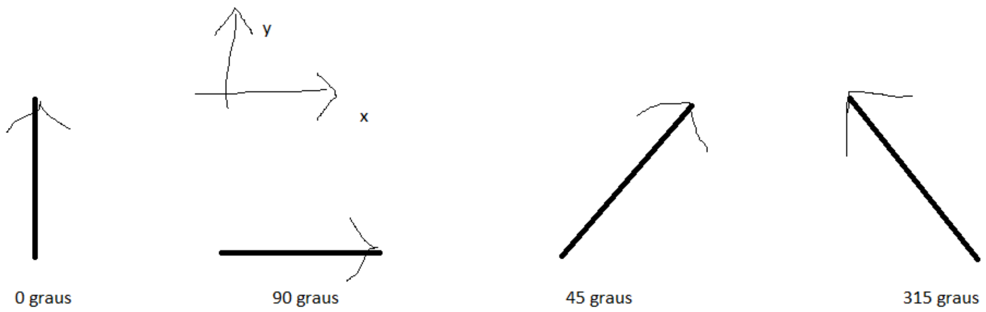
```
      <route edges="laneB"/>

   </flow>

</routes>
```

```
// rotate angle so 0 is east (in Sumo (TraCI's) angle interpretation 0 is north, 90 is east)
```

The angle above is in degrees, with respect to the y-axis:



The source code of TraCIConnection.cc in software Veins (http://veins.car2x.org/documentation/faq/) has functions to convert coordinates and angles:

```
void TraCIConnection::setNetbounds(TraCICoord netbounds1, TraCICoord netbounds2, int margin) {

        this->netbounds1 = netbounds1;

        this->netbounds2 = netbounds2;

        this->margin = margin;

}


Coord TraCIConnection::traci2omnet(TraCICoord coord) const {

        return Coord(coord.x - netbounds1.x + margin, (netbounds2.y - netbounds1.y) - (coord.y - netbounds1.y) + margin);

}


std::list<Coord> TraCIConnection::traci2omnet(const std::list<TraCICoord>& list) const {

        std::list<Coord> result;
```

```
	std::transform(list.begin(), list.end(), std::back_inserter(result), traci2omnet_functor(*this));

	return result;

}


TraCICoord TraCIConnection::omnet2traci(Coord coord) const {

	return TraCICoord(coord.x + netbounds1.x - margin, (netbounds2.y - netbounds1.y) - (coord.y - netbounds1.y) + margin);

}


std::list<TraCICoord> TraCIConnection::omnet2traci(const std::list<Coord>& list) const {

	std::list<TraCICoord> result;

	std::transform(list.begin(), list.end(), std::back_inserter(result), std::bind1st(std::mem_fun<TraCICoord, TraCIConnection, Coord>(&TraCIConnection::omnet2traci), this));

	return result;

}


double TraCIConnection::traci2omnetAngle(double angle) const {


	// rotate angle so 0 is east (in TraCI's angle interpretation 0 is north, 90 is east)

	angle = 90 - angle;


	// convert to rad

	angle = angle * M_PI / 180.0;


	// normalize angle to -M_PI <= angle < M_PI

	while (angle < -M_PI) angle += 2 * M_PI;

	while (angle >= M_PI) angle -= 2 * M_PI;
```

```
        return angle;
}

double TraCIConnection::omnet2traciAngle(double angle) const {

        // convert to degrees
        angle = angle * 180 / M_PI;

        // rotate angle so 0 is south (in OMNeT++'s angle interpretation 0 is east, 90 is north)
        angle = 90 - angle;

        // normalize angle to -180 <= angle < 180
        while (angle < -180) angle += 360;
        while (angle >= 180) angle -= 360;

        return angle;
}

}
```