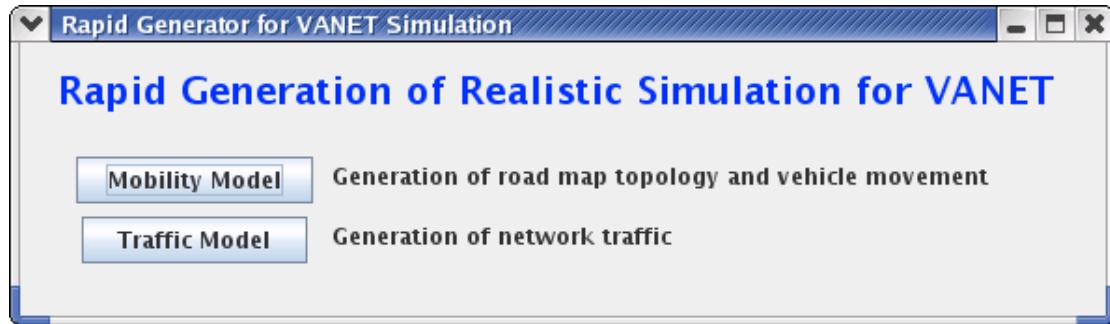


# Rapid Generation of Realistic Simulation for VANET

## Manual

Updated 30 July 2015

As developing VANET in the practical application is too costly, the development will have to rely on simulation.



### Install environment

To use this software, you will need the following software installed on your computer:

- Linux or Windows XP
- Java SDK 1.6 - <http://java.sun.com>
- SUMO version: 0.12.3 - <http://sumo.sourceforge.net>

If SUMO installs in Linux:

- Xerces (XML-parser) - <http://xerces.apache.org/xerces-c/index.html>
- FOX-Toolkit (GUI Toolkit) - <http://www.fox-toolkit.org/>
- PROJ (Cartographic Projections Library):  
<http://www.remotesensing.org/proj/>
- GDAL (Geospatial Data Abstraction Library):  
<http://www.remotesensing.org/gdal/>
- NS2 version: 2.34 (all-in-one) or later - <http://www.isi.edu/nsnam/ns/>

*Note: Refer to each software's website for details on installation. This document only shows how to install the program for specific version. However, according to the versions of Xerces, FOX-Toolkit, PROJ, and GDAL, please check SUMO's requirement.*

In Ubuntu, you need to install extra packages for SUMO.

#### **A. SUMO part**

```
# sudo apt-get update
```

For more detail, please refer SUMO wiki

<http://sourceforge.net/apps/mediawiki/sumo/index.php?title=LinuxBuild>

#### **B. NS2 part**

```
# sudo apt-get install build-essential  
# sudo apt-get install tcl8.4 tcl8.4-dev tk8.4 tk8.4-dev  
# sduo apt-get install libxmu-dev libxmu-headers
```

In Window platform, please refer the detail in SUMO wiki.

<http://sourceforge.net/apps/mediawiki/sumo/index.php?title=WindowsBuild>

### ■ Installation Xerces part:

Reference link: <http://xerces.apache.org/xerces-c/build-winunix.html>

For instance at your install path = /home/temp/xerces-c-src\_2\_7\_0

Next set your Xerces-C++ root path as follows:

```
export XERCESCROOT=/home/temp/xerces-c-src_2_7_0
```

One of the common ways to build Xerces-C++ is as follows:

```
cd src/xercesc  
./runConfigure -plinux -cgcc -xg++ -minmem -nsocket -tnative -rpthread
```

Build xercesc

```
gmake  
make install
```

Finish.

### ■ Installation FOX-Toolkit part:

You can configure simply as:

```
./configure --with-opengl=yes --prefix=$HOME  
make  
make install
```

Finish.

### ■ Installation PROJ and GDAL part:

You can configure simply as:

```
./configure --prefix=$HOME  
make  
make install
```

Finish.

## ■ Installation SUMO part:

Reference link: <http://sumo.sourceforge.net/wiki/index.php/LinuxBuild>

You can configure simply as:

```
./configure --with-fox=$HOME --with-proj-gdal=$HOME --with-xerces=$HOME --prefix=$HOME  
make  
make install
```

Enjoin!

Note: If you have gcc version problem, we suggest you input the command:

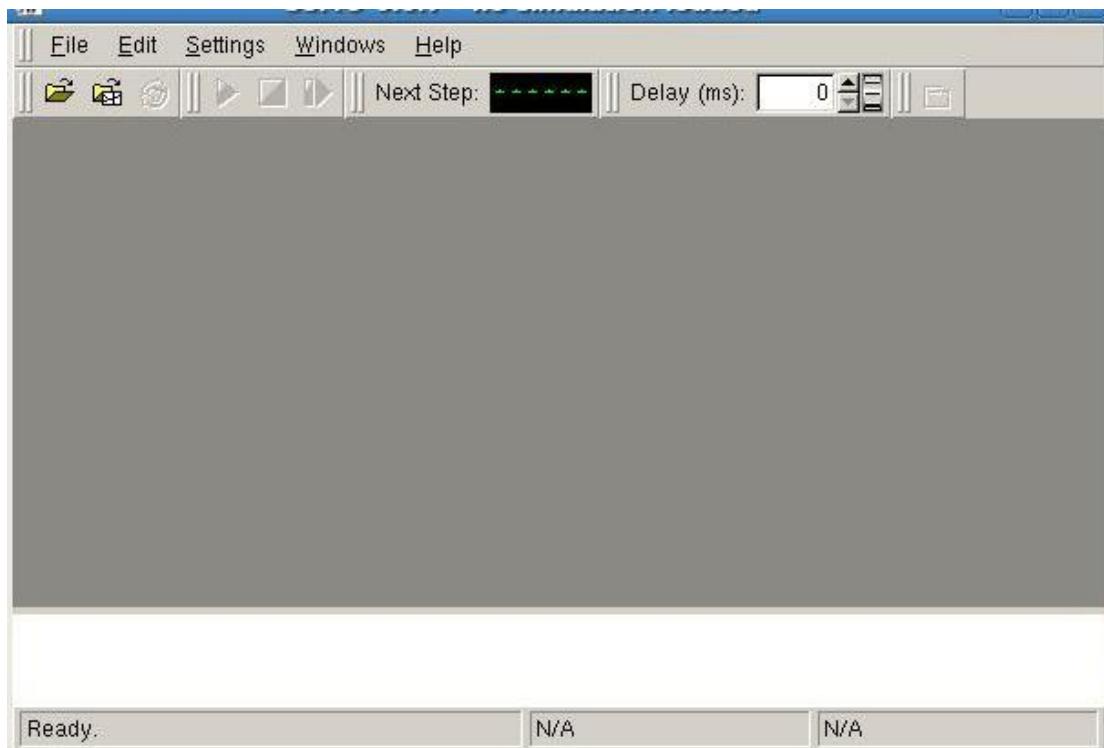
```
CC=gcc34  
CXXFLAGS=-fpermissive ./configure --with-fox=$HOME --with-proj-gdal=$HOME --with-xerces=$H  
OME --prefix=$HOME
```

You can use **GUI SUMO** to check install successes or not.

For instance, We are installation path = /home/temp/sumo-0.9.8

```
cd /home/temp/sumo-0.9.8/src  
./sumo-guisim
```

If you installation successes, you will see under picture.



## ■ Installation NS2 part:

Reference: <http://www.isi.edu/nsnam/ns/ns-build.html#allinone>

When you have download **ns-allinone-2.31.tar.gz**.

```
[root@localhost root]# tar -zvxf ns-allinone-2.31.tar.gz
```

Install NS2

```
[root@localhost root]# cd ns-allinone-2.31
[root@localhost ns-allinone-2.31]# ./install
=====
* Testing for Cygwin environment
=====
Cygwin not detected, proceeding with regular install.
=====
* Build XGraph-12.1
=====
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal
.....
```

Install finish.

Now, we must setting realation environment variable, and edit user home directory for .bashrc file. We will additional environment variable including PATH, LD\_LIBRARY\_PATH, and TCL\_LIBRARY.

```
export NS_HOME='pwd'/ns-allinone-2.31
export PATH=$NS_HOME/tcl8.4.14/unix:$NS_HOME/bin:$NS_HOME/tk8.4.14/unix:$PATH
export LD_LIBRARY_PATH=$NS_HOME/otcl-1.13:$NS_HOME/lib:$LD_LIBRARY_PATH
export TCL_LIBRARY=$NS_HOME/tcl8.4.14/library
```

We also suggest edit to ~/.bash\_profile file, and you can avoid every time login and resetting environment variable.

## Building the sources

To build the sources simply compile with "javac \*.java"  
To run it simply do "java vanetsim"

## Or you can execute MOVE by jar file without building the sources

To run it simply do "java -jar MOVE.jar"

Note: If your trace is very large, you can input: **Java -Xmx512m -jar MOVE.jar**

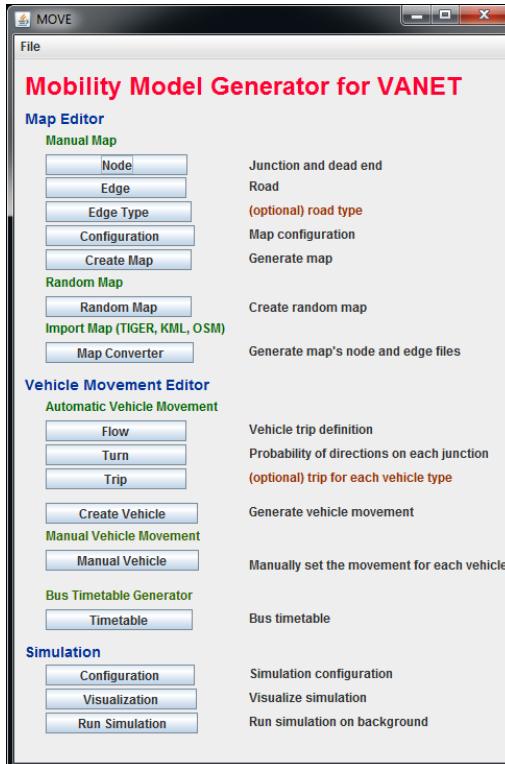
## Software binaries

Ensure that all the paths of the software binaries are exported to the shell. Therefore you are able to call the software from any current working directories. This is important as this software will not be able to set the binary paths.

## How to use this software

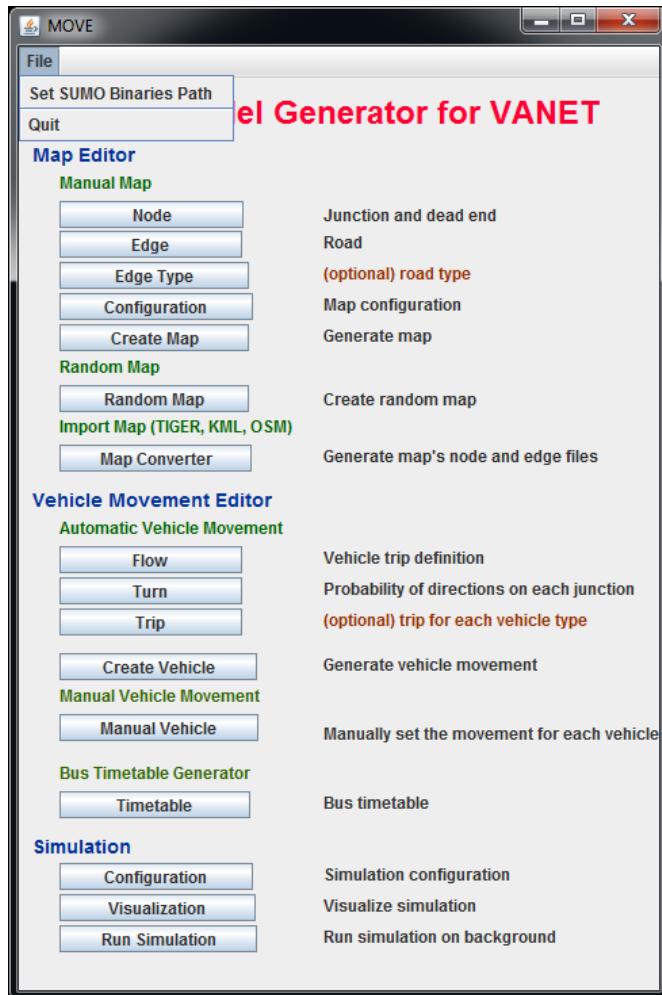
### I. Mobility Generation

This part of the software (called MOVE - MObility model generator for VEhicular networks) will generate the mobility model created by SUMO. Firstly select "Mobility Model" on the main top level menu.



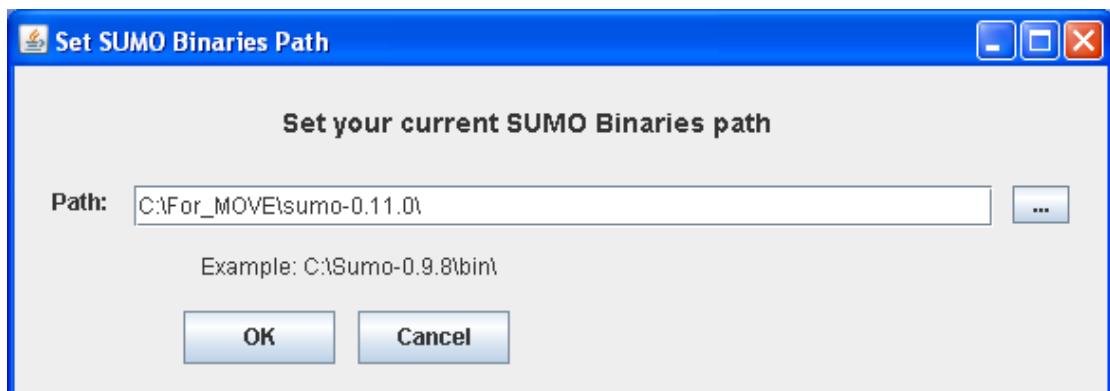
## [Set SUMO Binaries Path only use on windows]

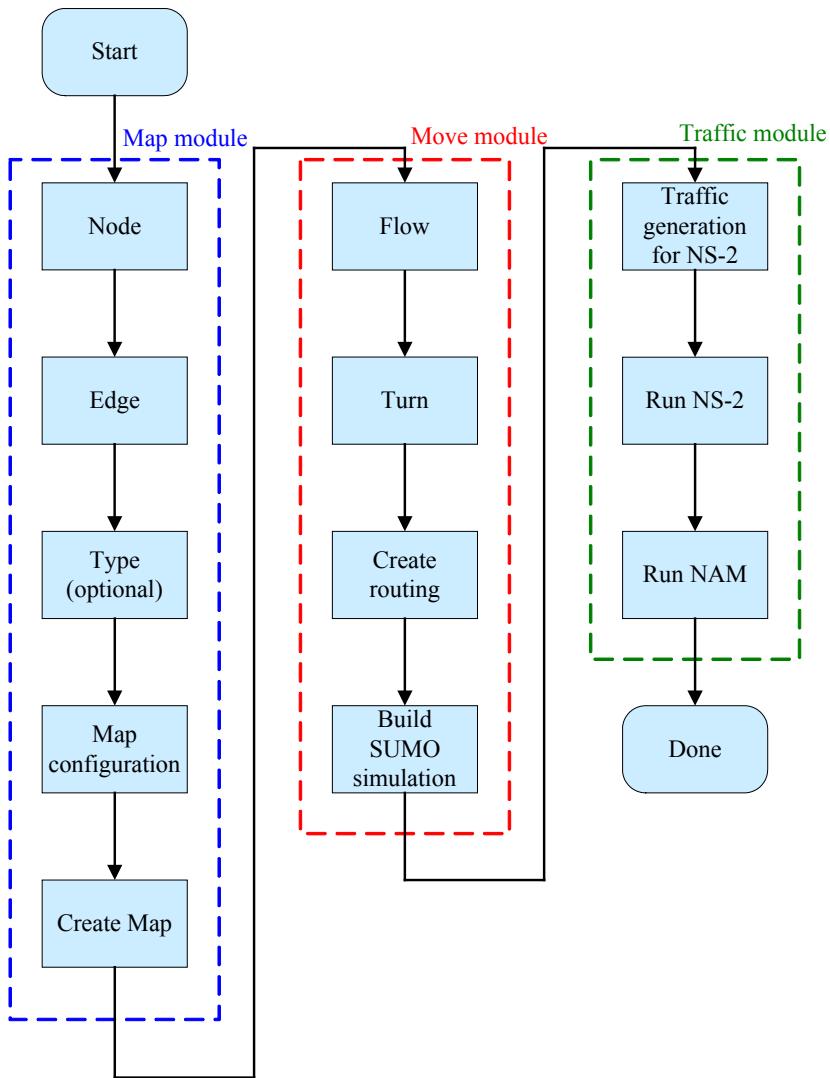
Step 1: To click “File” then choose “Set SUMO Binaries Path”.



Step 2: Please you set SUMO binaries path. For instance, we set sumo path that is located at “C:\For\_MOVE\sumo-0.12.0\

Hint: You only set the path once, MOVE will save the configuration.





This is roadmap in the VANET manual. It is main let user friendly using and practice VANET.

We use example and step-by-step description in figure 1.

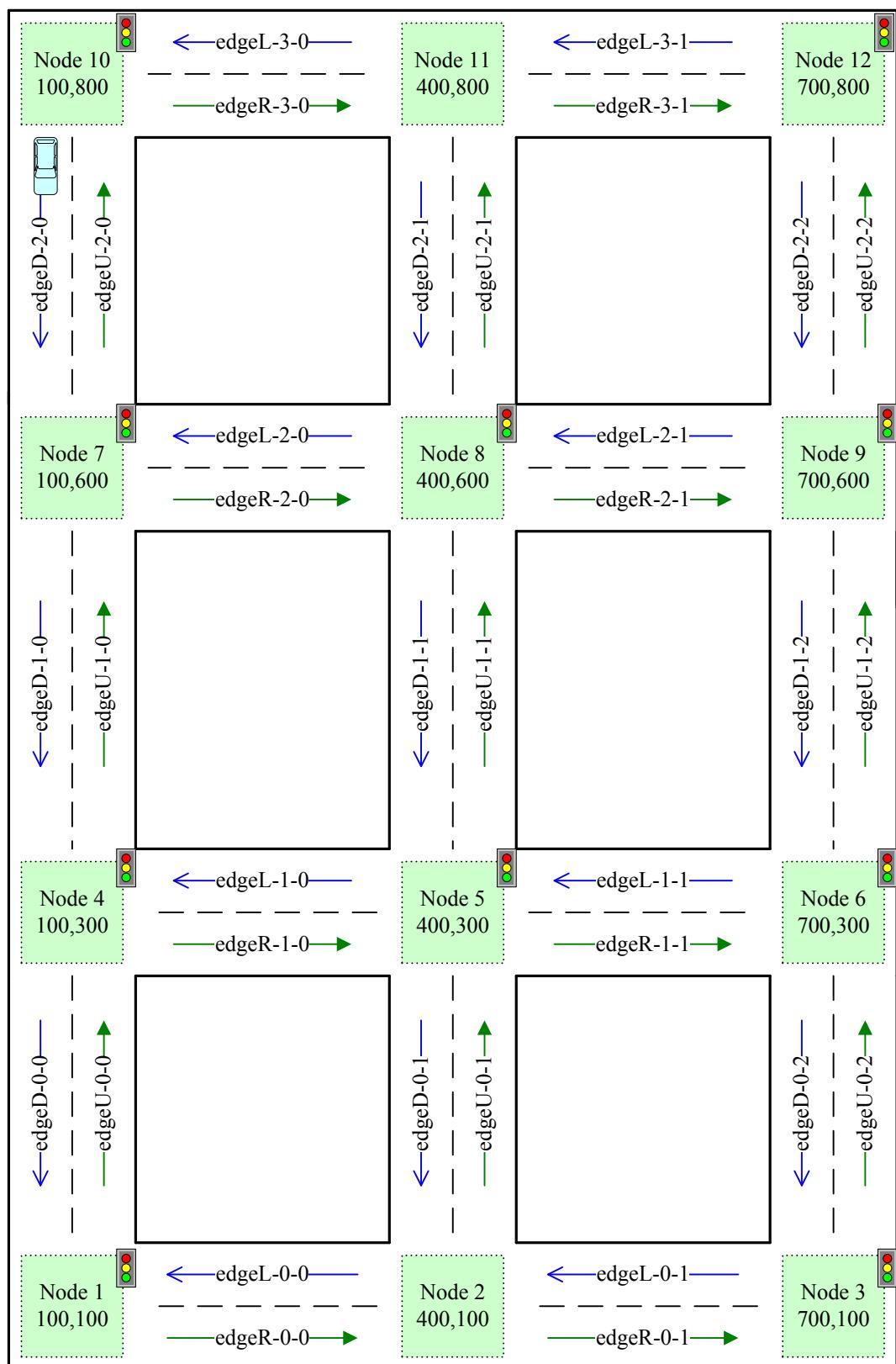
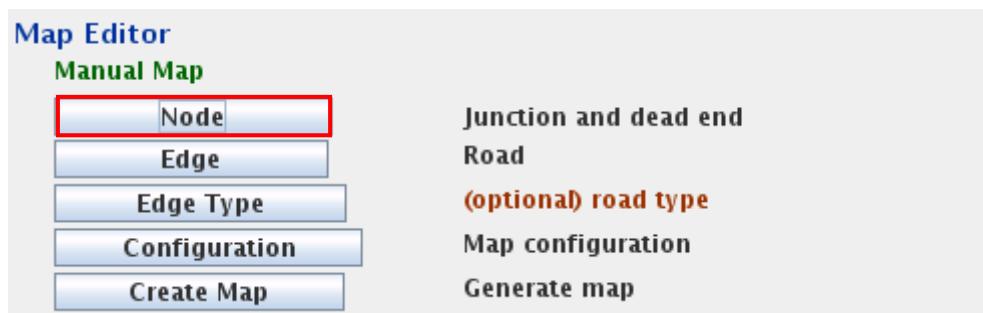


Fig.1. Example

## I.i Map generation

### Step 1: Manually create your own map nodes (ex\_NODE.nod.xml)

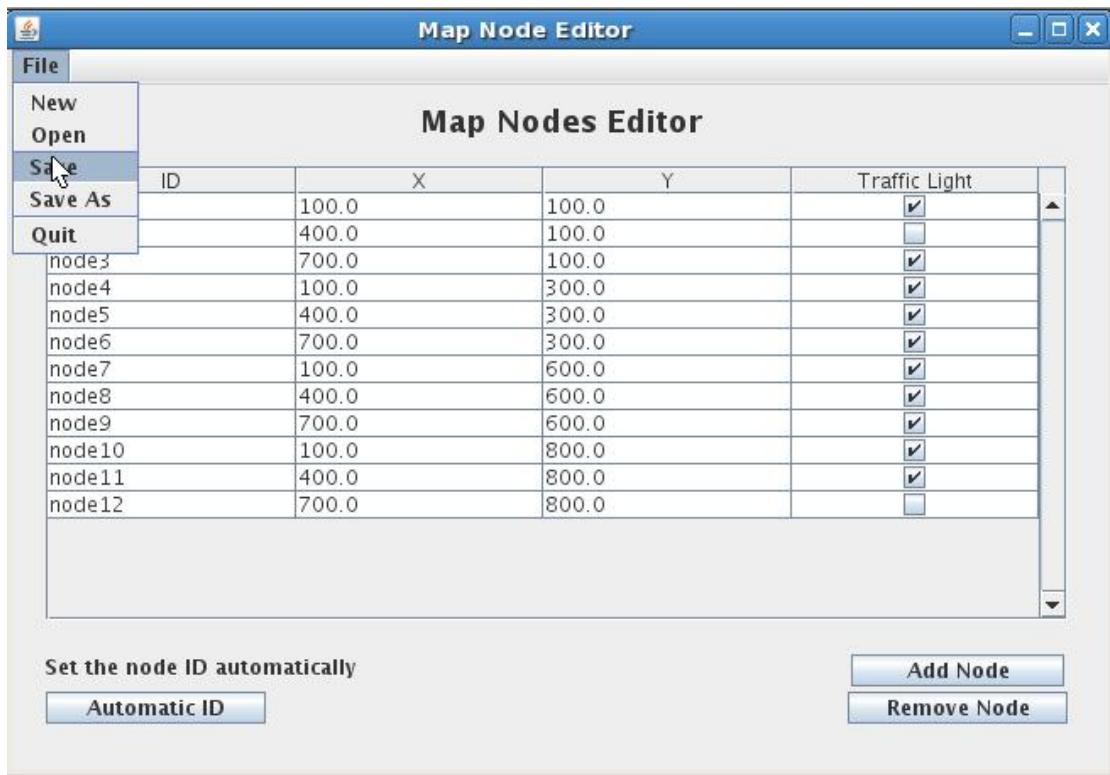


First, select “Node” from MOVE main menu. This is simply where all the map nodes are.

The screenshot shows the 'Map Node Editor' window with the title 'Map Node Editor'. The main area is a table titled 'Map Nodes Editor' with columns for 'ID', 'X', 'Y', and 'Traffic Light'. The table contains 12 rows of data, each representing a node with ID, X, Y coordinates, and a checked 'Traffic Light' box. Below the table are three buttons: 'Set the node ID automatically' (with 'Automatic ID' as a sub-option), 'Add Node', and 'Remove Node'.

ID	X	Y	Traffic Light
node1	100.0	100.0	<input checked="" type="checkbox"/>
node2	400.0	100.0	<input type="checkbox"/>
node3	700.0	100.0	<input checked="" type="checkbox"/>
node4	100.0	300.0	<input checked="" type="checkbox"/>
node5	400.0	300.0	<input checked="" type="checkbox"/>
node6	700.0	300.0	<input checked="" type="checkbox"/>
node7	100.0	600.0	<input checked="" type="checkbox"/>
node8	400.0	600.0	<input checked="" type="checkbox"/>
node9	700.0	600.0	<input checked="" type="checkbox"/>
node10	100.0	800.0	<input checked="" type="checkbox"/>
node11	400.0	800.0	<input checked="" type="checkbox"/>
node12	700.0	800.0	<input type="checkbox"/>

You can set node follow upper legend.



Then select ‘File’ -> ‘save’ or ‘save as’ when you are done editing.

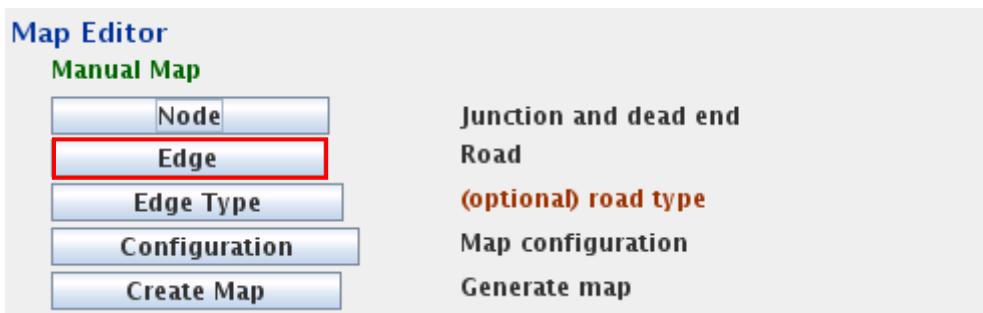
Save the file as <name>.nod.xml (example.nod.xml).

The example detail in ex\_NODE.nod.xml:

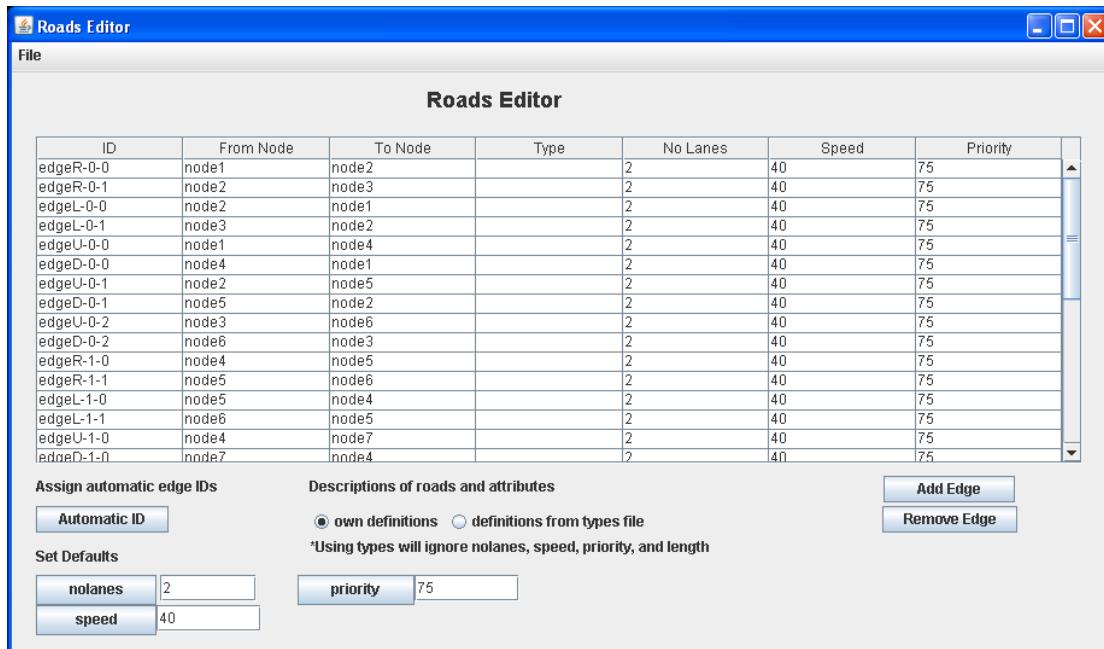
```

<nodes>
  <node id="node1" x="100.0" y="100.0" type="traffic_light"/>
  <node id="node2" x="400.0" y="100.0" type="priority"/>
  <node id="node3" x="700.0" y="100.0" type="traffic_light"/>
  <node id="node4" x="100.0" y="300.0" type="traffic_light"/>
  <node id="node5" x="400.0" y="300.0" type="traffic_light"/>
  <node id="node6" x="700.0" y="300.0" type="traffic_light"/>
  <node id="node7" x="100.0" y="600.0" type="traffic_light"/>
  <node id="node8" x="400.0" y="600.0" type="traffic_light"/>
  <node id="node9" x="700.0" y="600.0" type="traffic_light"/>
  <node id="node10" x="100.0" y="800.0" type="traffic_light"/>
  <node id="node11" x="400.0" y="800.0" type="traffic_light"/>
  <node id="node12" x="700.0" y="800.0" type="priority"/>
</nodes>
```

## Step 2: Manually create your own map edge (ex\_EDGE.edg.xml)



First, select “Edge” from MOVE main menu. This is similar to node editor. This is where you can specify all the roads (a road will create a connection between two nodes created previously).



You can set edge follow upper legend.

Save the file as <name>.edg.xml (example.edg.xml).

The example detail in ex\_EDGE.edg.xml:

```
<edges>
<edge id="edgeR-0-0" fromnode="node1" tonode="node2" priority="75"
nolanes="2" speed="40" />
<edge id="edgeR-0-1" fromnode="node2" tonode="node3" priority="75"
nolanes="2" speed="40" />
<edge id="edgeL-0-0" fromnode="node2" tonode="node1" priority="75"
nolanes="2" speed="40" />
<edge id="edgeL-0-1" fromnode="node3" tonode="node2" priority="75"
nolanes="2" speed="40" />
```

```

<edge id="edgeU-0-0" fromnode="node1" tonode="node4" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeD-0-0" fromnode="node4" tonode="node1" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeU-0-1" fromnode="node2" tonode="node5" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeD-0-1" fromnode="node5" tonode="node2" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeU-0-2" fromnode="node3" tonode="node6" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeD-0-2" fromnode="node6" tonode="node3" priority="75"
nolanes="2" speed="40"/>

<edge id="edgeR-1-0" fromnode="node4" tonode="node5" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeR-1-1" fromnode="node5" tonode="node6" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeL-1-0" fromnode="node5" tonode="node4" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeL-1-1" fromnode="node6" tonode="node5" priority="75"
nolanes="2" speed="40"/>

<edge id="edgeU-1-0" fromnode="node4" tonode="node7" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeD-1-0" fromnode="node7" tonode="node4" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeU-1-1" fromnode="node5" tonode="node8" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeD-1-1" fromnode="node8" tonode="node5" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeU-1-2" fromnode="node6" tonode="node9" priority="75"
nolanes="2" speed="40"/>
<edge id="edgeD-1-2" fromnode="node9" tonode="node6" priority="75"
nolanes="2" speed="40"/>

<edge id="edgeR-2-0" fromnode="node7" tonode="node8" priority="75"
nolanes="2" speed="40"/>

```

```

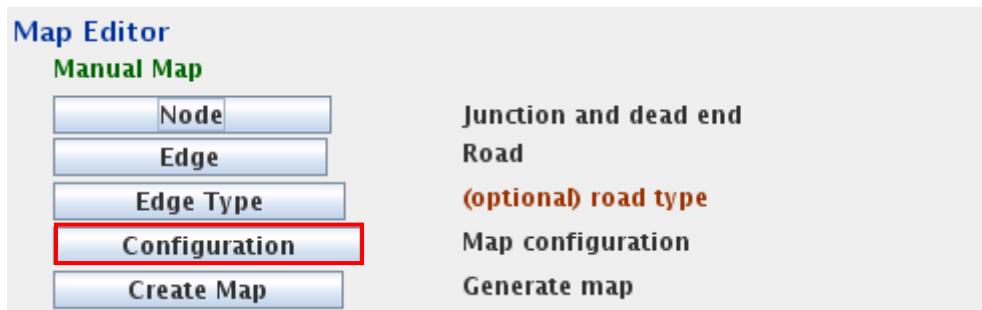
<edge id="edgeR-2-1" fromnode="node8" tonode="node7" priority="75"
nolanes="2" speed="40" />
<edge id="edgeL-2-0" fromnode="node8" tonode="node9" priority="75"
nolanes="2" speed="40" />
<edge id="edgeL-2-1" fromnode="node9" tonode="node8" priority="75"
nolanes="2" speed="40" />

<edge id="edgeU-2-0" fromnode="node7" tonode="node10" priority="75"
nolanes="2" speed="40" />
<edge id="edgeD-2-0" fromnode="node10" tonode="node7" priority="75"
nolanes="2" speed="40" />
<edge id="edgeU-2-1" fromnode="node8" tonode="node11" priority="75"
nolanes="2" speed="40" />
<edge id="edgeD-2-1" fromnode="node11" tonode="node8" priority="75"
nolanes="2" speed="40" />
<edge id="edgeU-2-2" fromnode="node9" tonode="node12" priority="75"
nolanes="2" speed="40" />
<edge id="edgeD-2-2" fromnode="node12" tonode="node9" priority="75"
nolanes="2" speed="40" />

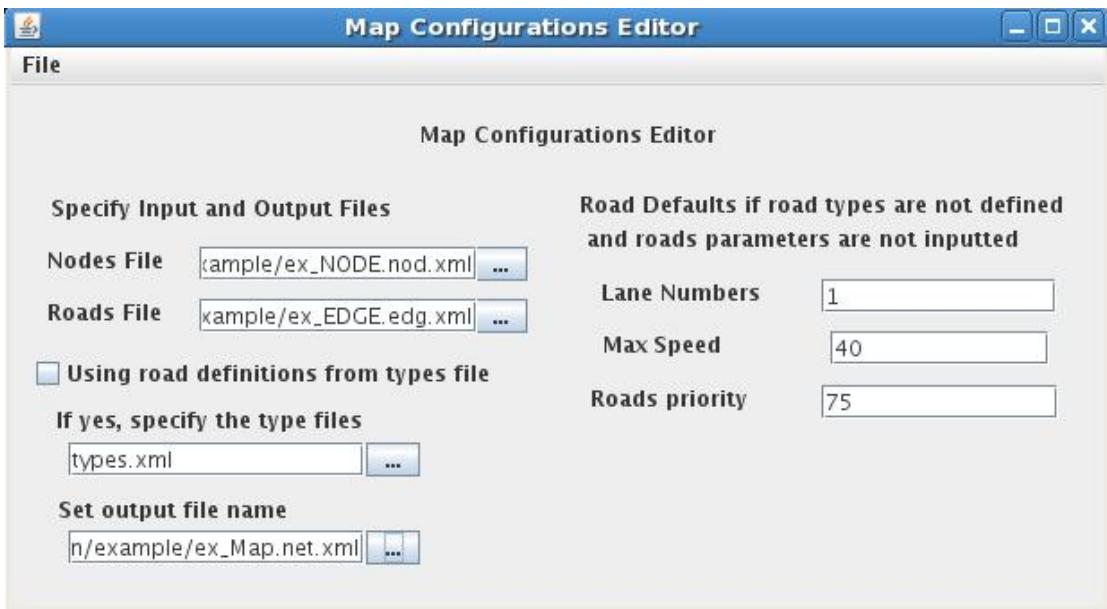
<edge id="edgeR-3-0" fromnode="node10" tonode="node11" priority="75"
nolanes="2" speed="40" />
<edge id="edgeR-3-1" fromnode="node11" tonode="node10" priority="75"
nolanes="2" speed="40" />
<edge id="edgeL-3-0" fromnode="node11" tonode="node12" priority="75"
nolanes="2" speed="40" />
<edge id="edgeL-3-1" fromnode="node12" tonode="node11" priority="75"
nolanes="2" speed="40" />
</edges>

```

### Step 3: Map configuration editor (ex\_Map.netc.cfg)



After that, choose “Configuration” from MOVE main menu.



Select you are previously created/edited ‘**nod**’ and ‘**edg**’ flies location. There are some defaults here. They only work when you did not fill the properties in the “**Edge**” editor or the types file is missing.

Save the file as <name>.netc.cfg (**ex\_Map.netc.cfg**)

Example detail in **ex\_Map.netc.cfg**:

<*configuration*>

```
<input
    xml-node-files="C:\For_MOVE\MOVE_example\ex_NODE.nod.xml"
    xml-edge-files="C:\For_MOVE\MOVE_example\ex_EDGE.edg.xml"
    xml-connection-files=""
    type-file="">
/>

<output
    output-file="C:\For_MOVE\MOVE_example\ex_Map.net.xml">
/>

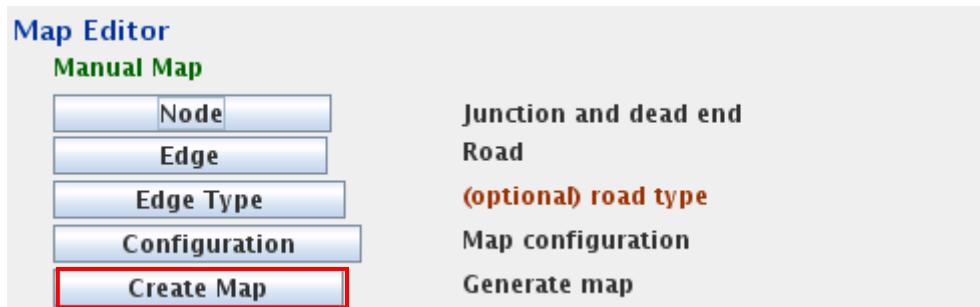
<defaults
    type="Unknown"
    lanenumber="1"
    speed="40"
```

```

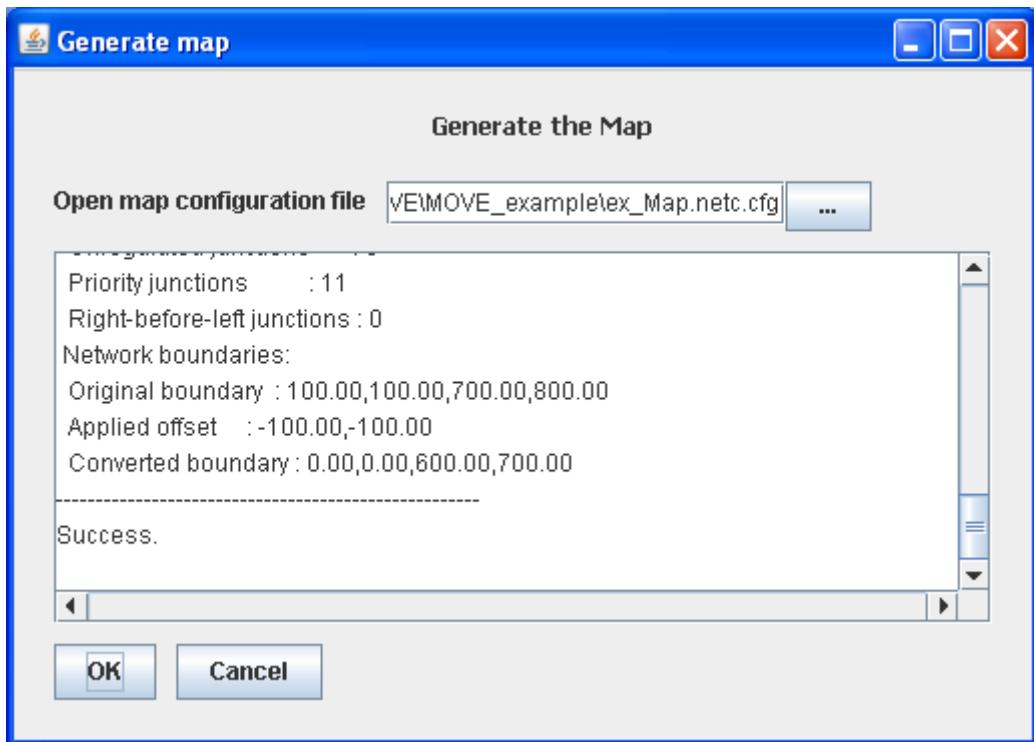
    priority="75"
    capacity-norm=""
/>
<reports
    print-options="false"
/>
<processing
    speed-in-kmh="false"
    no-turnarounds="false"
    remove-geometry="x"
/>
</configuration>

```

#### Step 4: Generate the Map (ex\_Map.net.xml)



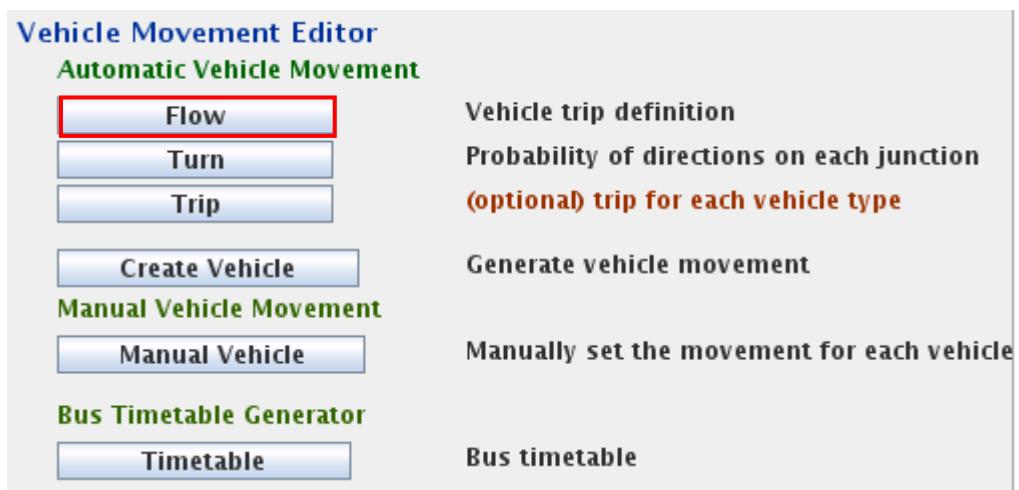
Finally, select “Create Map” from MOVE main menu.



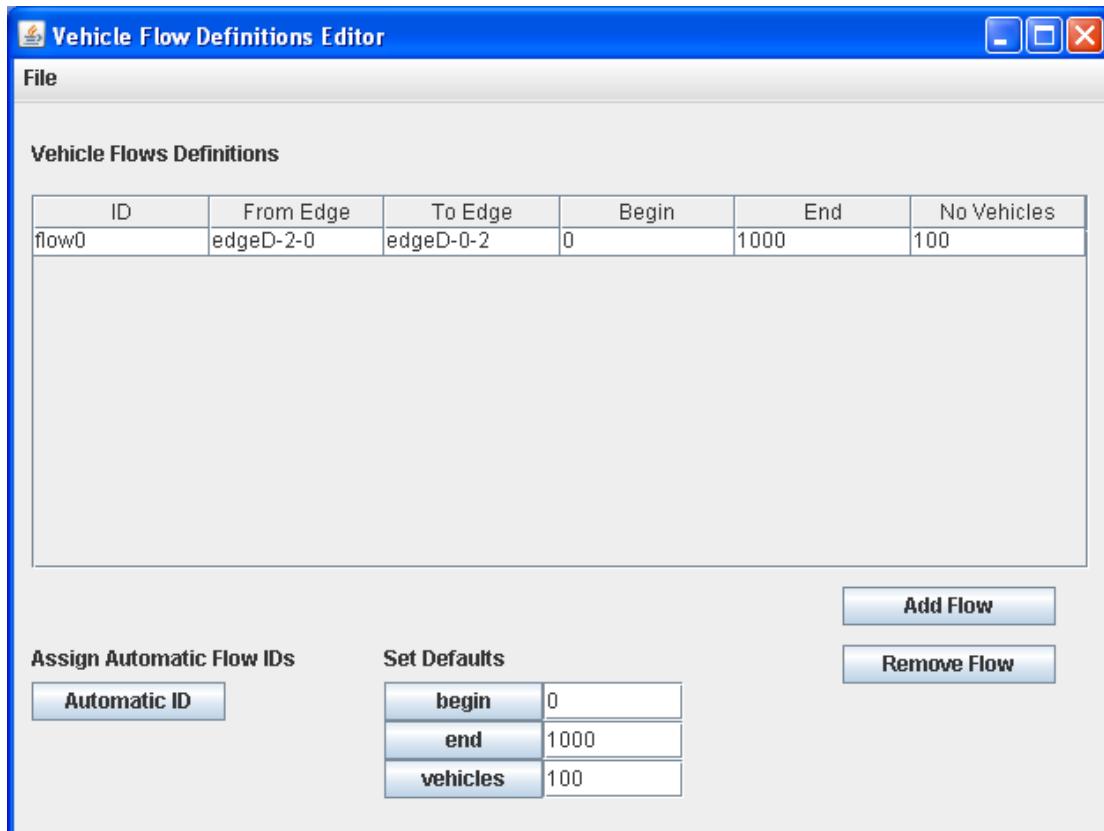
Simply select the netc file (**ex\_Map.netc.cfg**) and click OK. A <name>.net.xml (**ex\_Map.net.xml**) will be automatically generated. This is your map file.

## I.ii Vehicle movements generation

### Step 1: Using flow definition (ex\_FLOW.flow.xml)



Simply select “Flow” from MOVE main menu. This editor will specify the groups of vehicle movements flow on simulation.



You can set flow follow upper legend.

Using these groups of flows, movements of the vehicles will be generated between the two specified edges. For example if you have:

edge1 (connected to node 1 and 2)

edge2 (connected to node 2 and 3)

edge3 (connected to node 3 and 4)

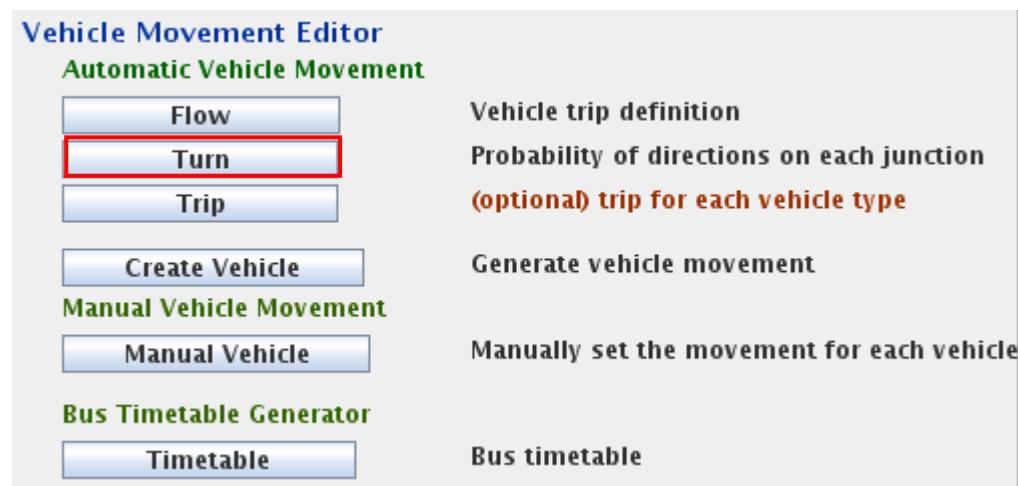
If you want a car to move from node 1 to 2 to 3 (1-2-3), you must assign the flow to be from edge1 to edge3 (NOT edge1 to edge2). If you assign it from edge1 to edge2, it will only move halfway (from 1- 2). Just remember that the movement pattern is all ways from the beginning node of a source edge to the beginning node of a destination edge.

Save the file as <name>.flow.xml (**ex\_FLOW.flow.xml**). You can also open and edit an existing .flow.xml file.

Example detail in **ex\_FLOW.flow.xml**:

```
<flows>
<flow id="flow0" from="edgeD-2-0" to="edgeD-0-2" begin="0" end="1000"
no="100" />
</flows>
```

## Step 2: Using junction turning ratio (not implement in the example)

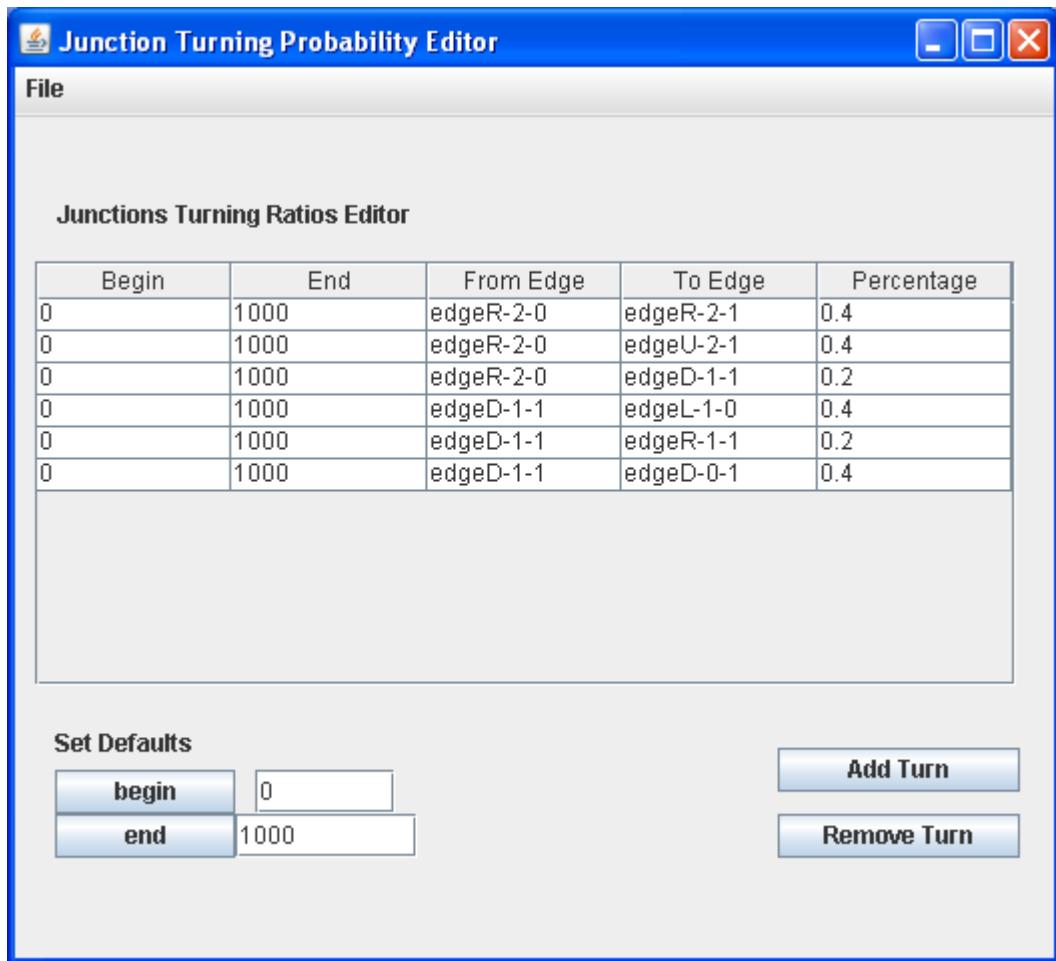


Simply select “Turn” from MOVE main menu. This is when you to specify a turning ratio for each junction. You must also create flow definitions as previously explained for the vehicle movements. **In the editor, you must group percentages together so they sum up as 1.0 (100%)**

Note: If you want setting same probability in MOVE, you can try input as follows:

```
sumo-jtrrouter -n ex_Map.net.xml -f ex_FLOWflow.xml -t ex_TURN.turn.xml -v -o
ex_ROU.rou.xml -b 0 -e 1000 --continue-on-unbuild --turn-defaults=30,30,40
```

*The probability format is “turning left, go straight, and turning right”*



You can set junctions turning ratios follow upper legend.

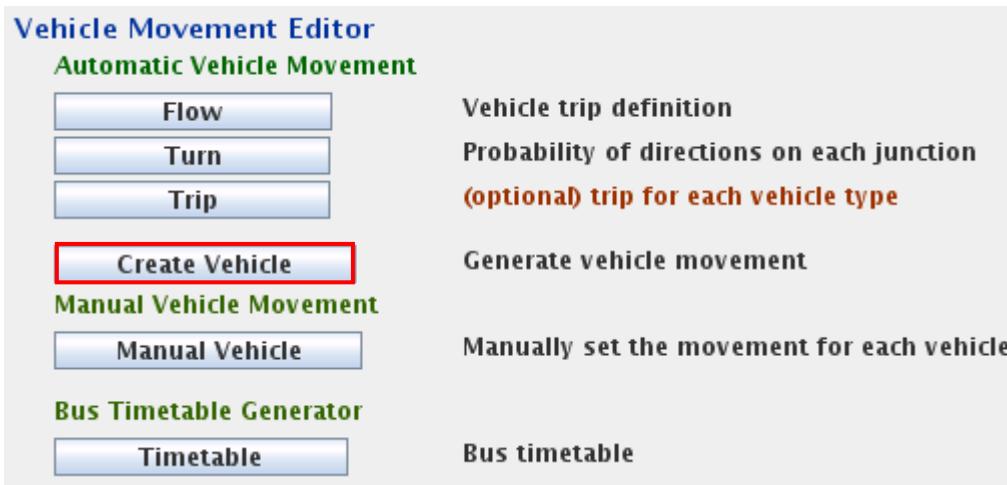
Example detail in (**ex\_TURN.turn.xml**)

```

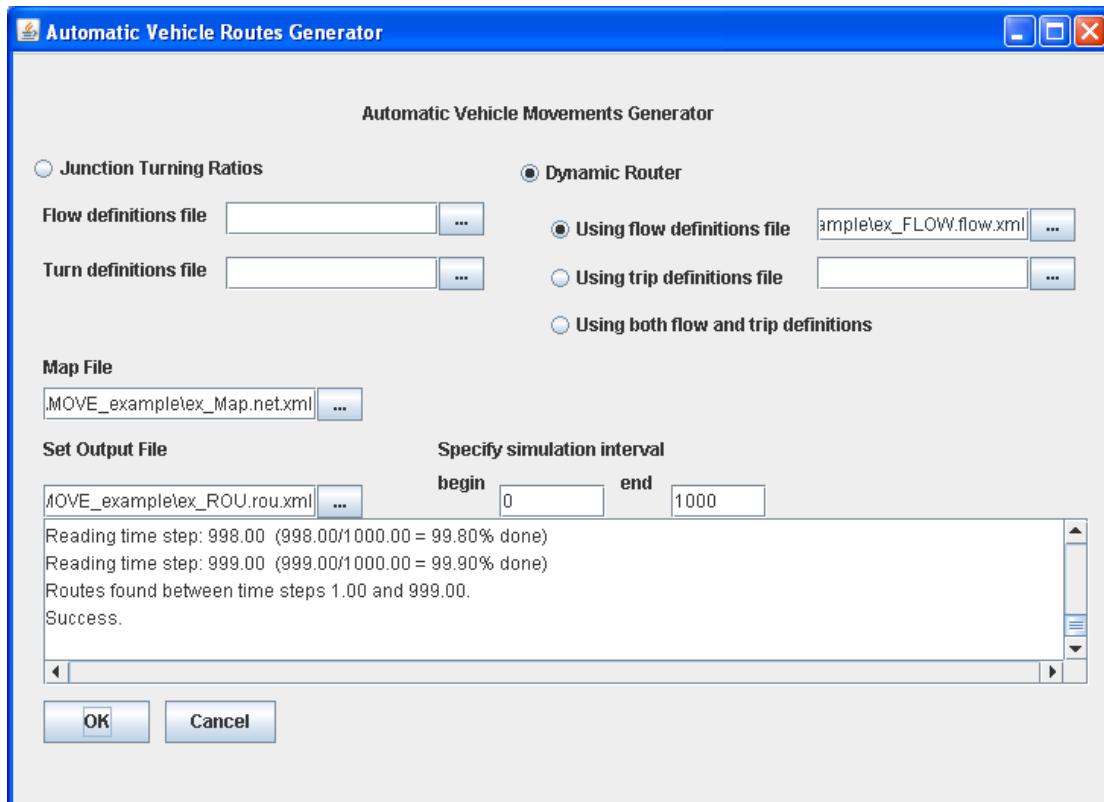
<turn-defs>
<interval begin="0" end="1000">
    <fromedge id="edgeR-2-0">
        <toedge id="edgeR-2-1" probability="0.4"></toedge>
        <toedge id="edgeU-2-1" probability="0.4"></toedge>
        <toedge id="edgeD-1-1" probability="0.2"></toedge>
    </fromedge>
    <fromedge id="edgeD-1-1">
        <toedge id="edgeL-1-0" probability="0.4"></toedge>
        <toedge id="edgeR-1-1" probability="0.2"></toedge>
        <toedge id="edgeD-0-1" probability="0.4"></toedge>
    </fromedge>
</interval>
</turn-defs>

```

### Step 3: Automatic vehicle movements (ex\_ROU.rou.xml)



Simply select “Create Vehicle” from MOVE main menu. This will simply create a number of vehicles at the start of simulation. For our example, we use junction turning ratios function.

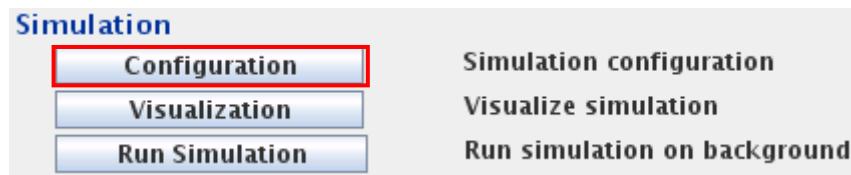


Select your previously created map file (e.g. **ex\_Map.net.xml**). Specify your output file location and name it as <name>.rou.xml (**ex\_ROU.rou.xml**). Set the beginning and end time of simulation. Finally click OK. The rou.xml file will be automatically generated.

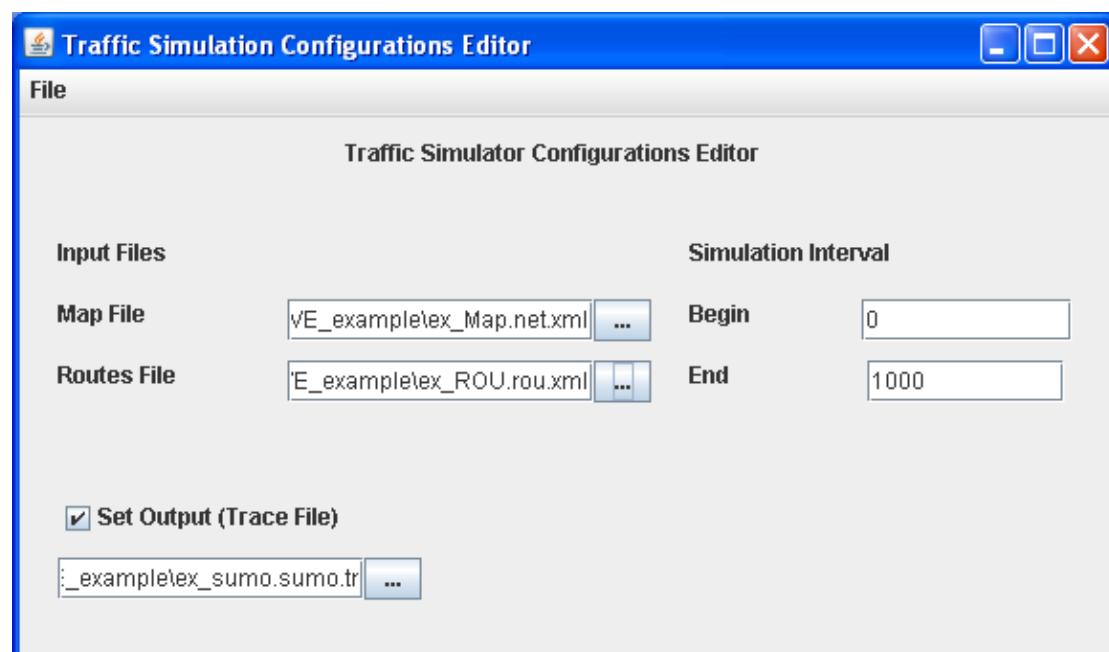
NOTE: this starting number of vehicles is the only way for generating movements for

random maps since the edge and nodes are unknown. May you occur error in sometime (e.g. short routing), when your map is small.

#### Step 4: Simulation setup (ex\_SUMOTRACE.sumo.tr and ex\_SUMO.sumo.cfg)



After the map and movement is complete, you will need to specify the configurations of the simulation. Select “Configuration” at the bottom on MOVE main menu.



Specify the <name>.net.xml (i.e. **ex\_Map.net.xml**) and <name>.rou.xml (i.e. **ex\_ROU.rou.xml**) location and specify the beginning and end time of simulation. If you want to create the trace file, don't forget to check the checkbox and specify your trace output name (e.g. <name>.sumo.tr). Then save the file as <name>.sumo.cfg (i.e. **ex\_SUMO.sumo.cfg**)

#### Example detail in (**ex\_SUMO.sumo.cfg**)

<configuration>

```
<input  
    net-file="C:\For MOVE\MOVE_example\ex_Map.net.xml"  
    route-files="C:\For MOVE\MOVE_example\ex_ROU.rou.xml"  
    additional-files=""
```

```

junction-files=""
/>


<output
    netstate-dump="C:\For_MOVE\MOVE_example\ex_sumo.sumo.tr"
    tripinfo-output="output-tripinfos.xml"
    emissions-output="output-emissions.xml"
    vehroute-output="output-vehroutes.xml"
/>

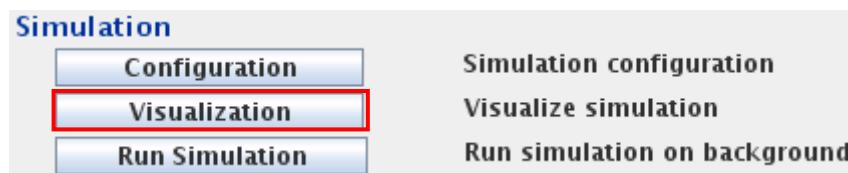

<time
    begin="0"
    end="1000"
    time-to-teleport="-1"
    srand="23423"
    route-steps="-1"
/>


<reports
    print-options="false"
/>

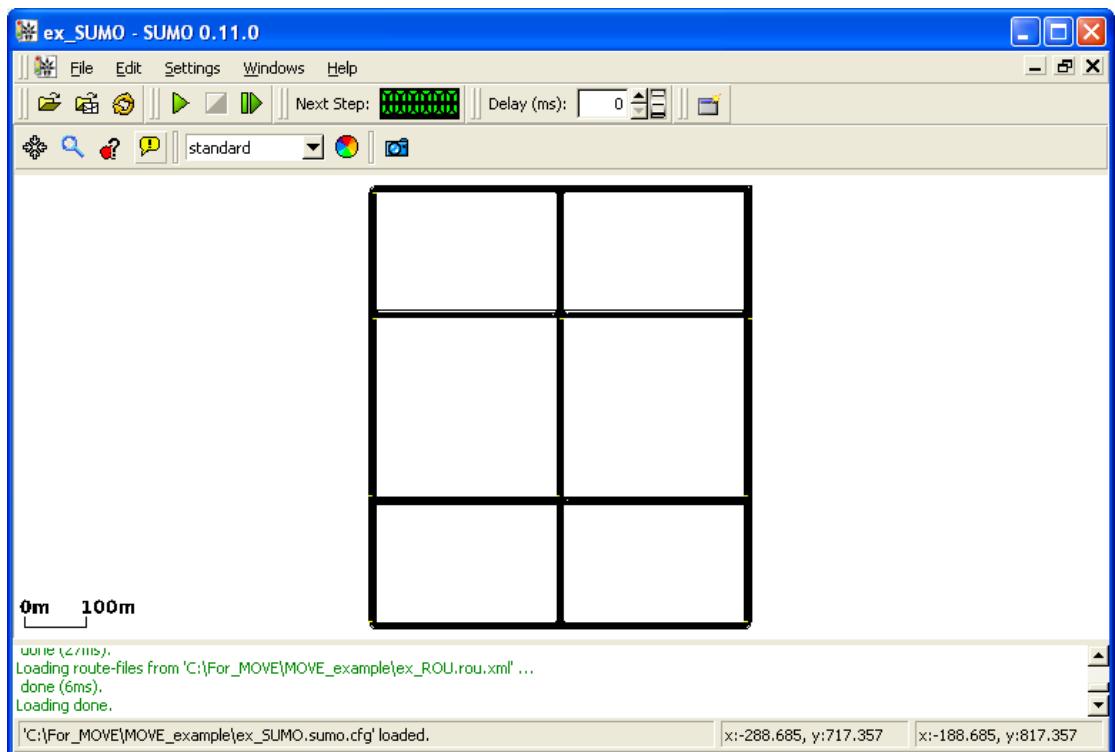

</configuration>

```

### Step 5: Visualize Simulation

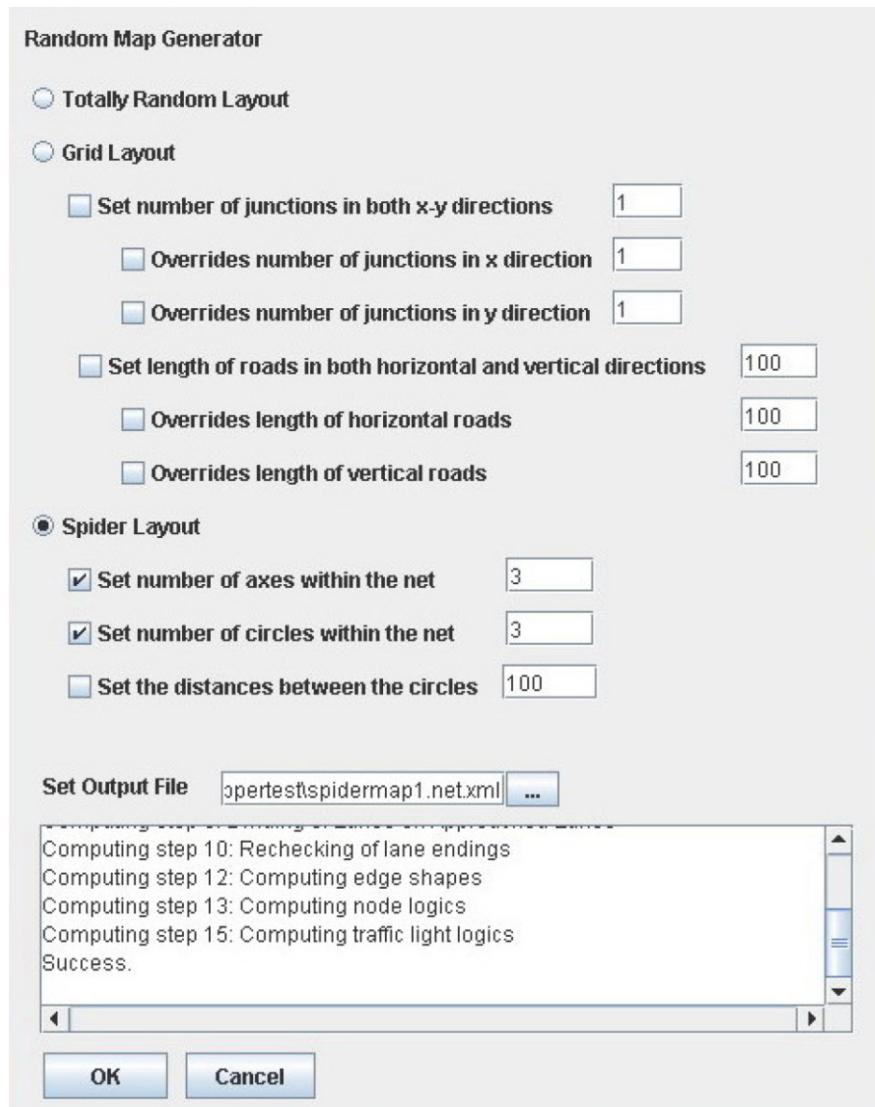


Now, you can select “Visualization” see the actual movements on vehicle.



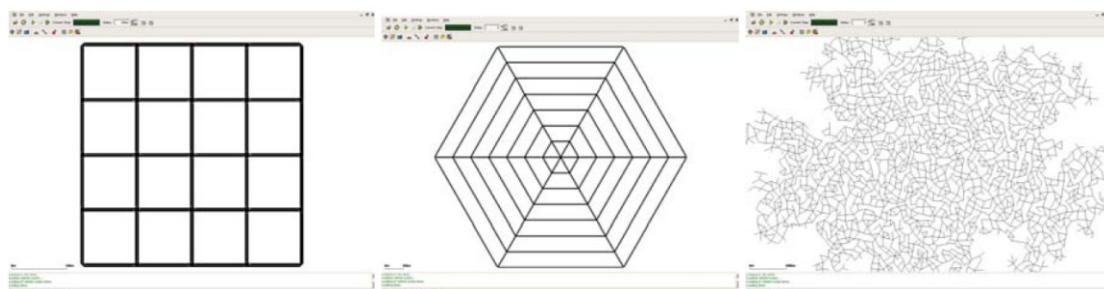
For details on using visualization with SUMO, please refer to the SUMO's manual.

### I.iii Automatically generated map (not implement in the example)



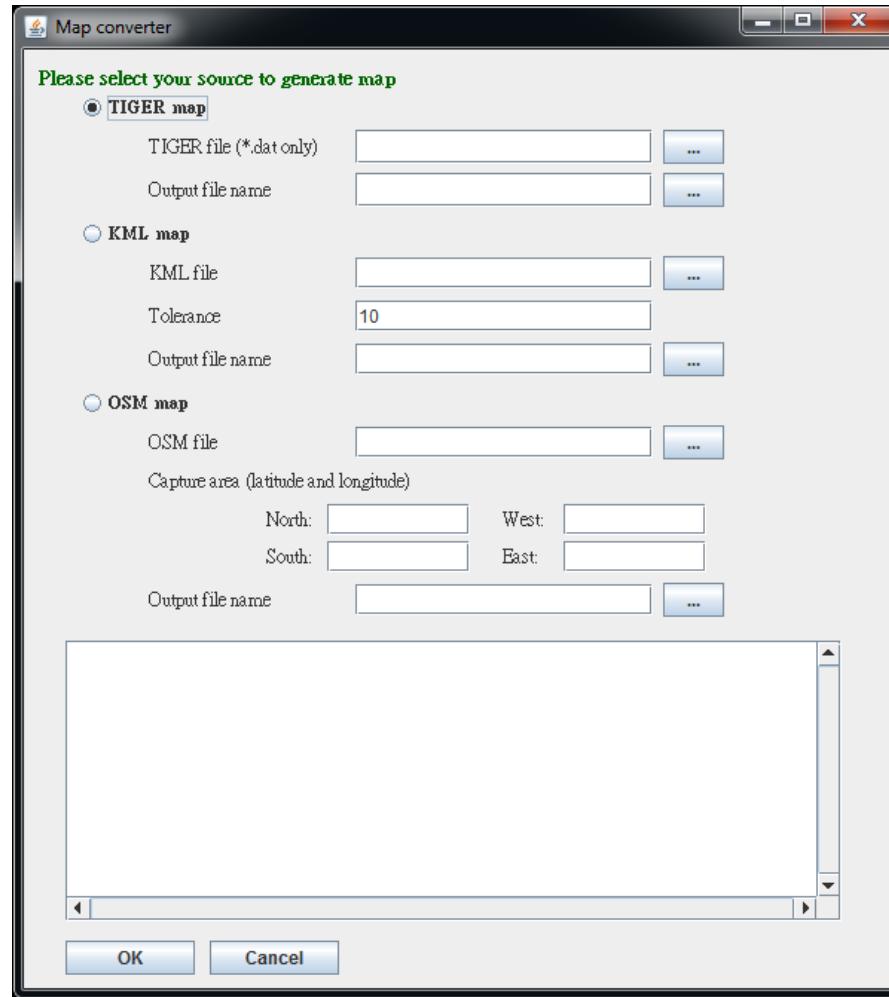
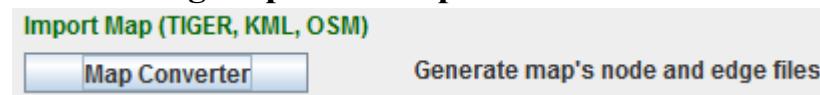
Choose "Random Map" from MOVE main menu to do this.

You can create a custom auto-generated map with three types of layout: grid, spider, or total random.



Select the options, then select your targeted <name>.net.xml file then click OK. A <name>.net.xml will be generated. This is your map file.

#### I.iv Creating map from Map converter



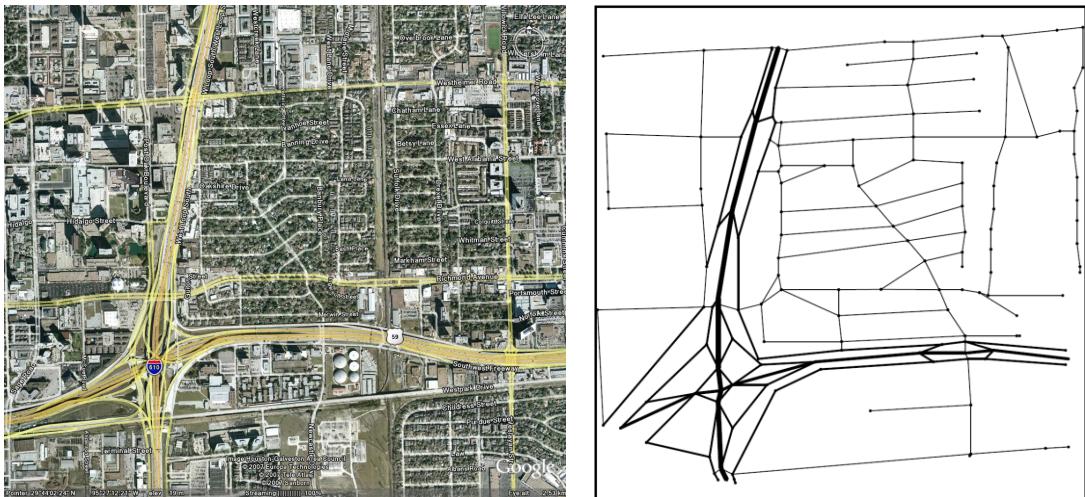
##### (a) TIGER map

If you have a TIGER Line map file, you can convert this into a MOVE map. Choose "Convert Tiger" from MOVE main menu. Choose the tiger file (usually <name>.dat) Then select the target filename.

**WARNING:** do not put any extension for the filename. Simply put the first name. i.e., /home/tigeroutput (no extension). This will generate <name>.nod.xml, <name>.edg.xml, <name>.netc.cfg, and <name>.net.xml (your map file)

*Note: current version of MOVE will regard all types of roads specified in the TIGER Line file as one type of road (can be seen by opening the <name>.edg.xml file). To support more types of roads, the source file tiger.java*

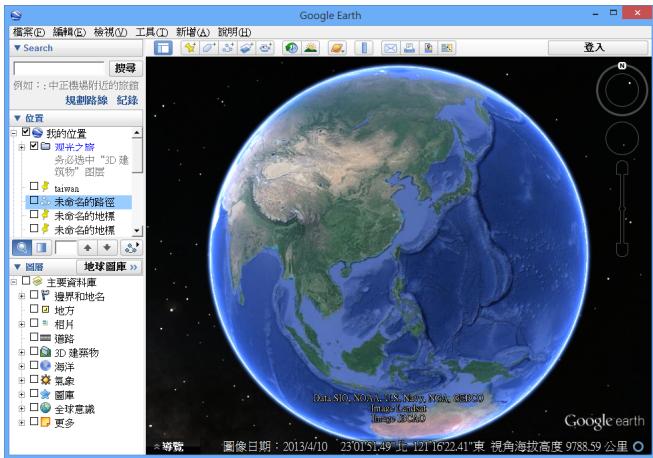
Under the figures show google map and TIGER line map.



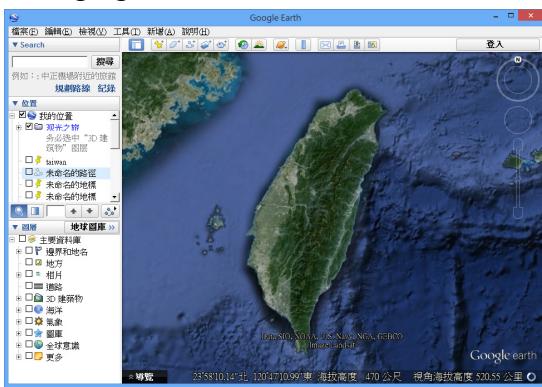
### (b) KML map

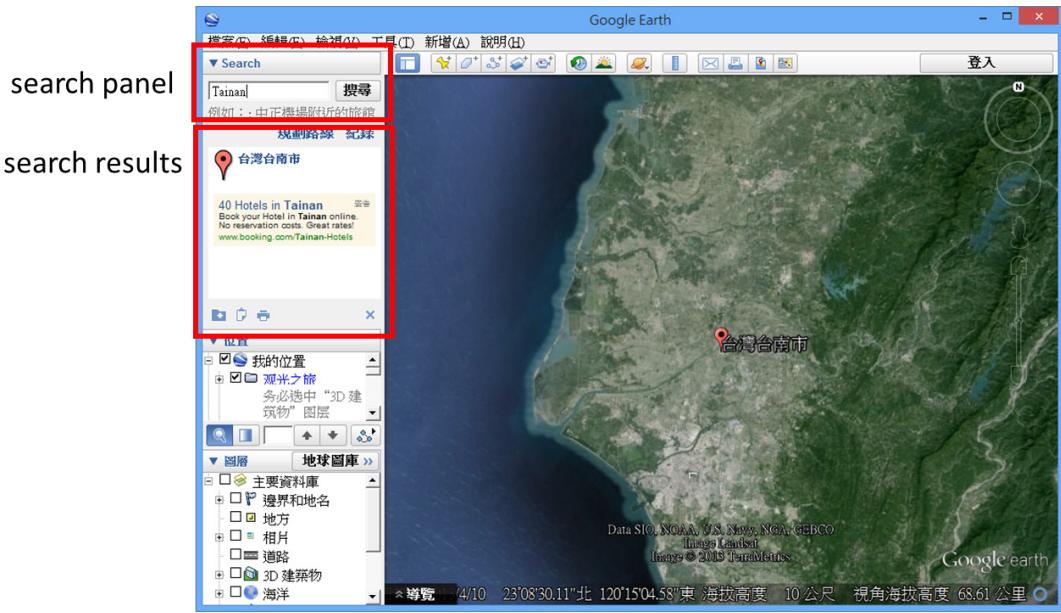
Before use this function, user needs to create KML file by Google Earth.

Step 1: Open Google earth

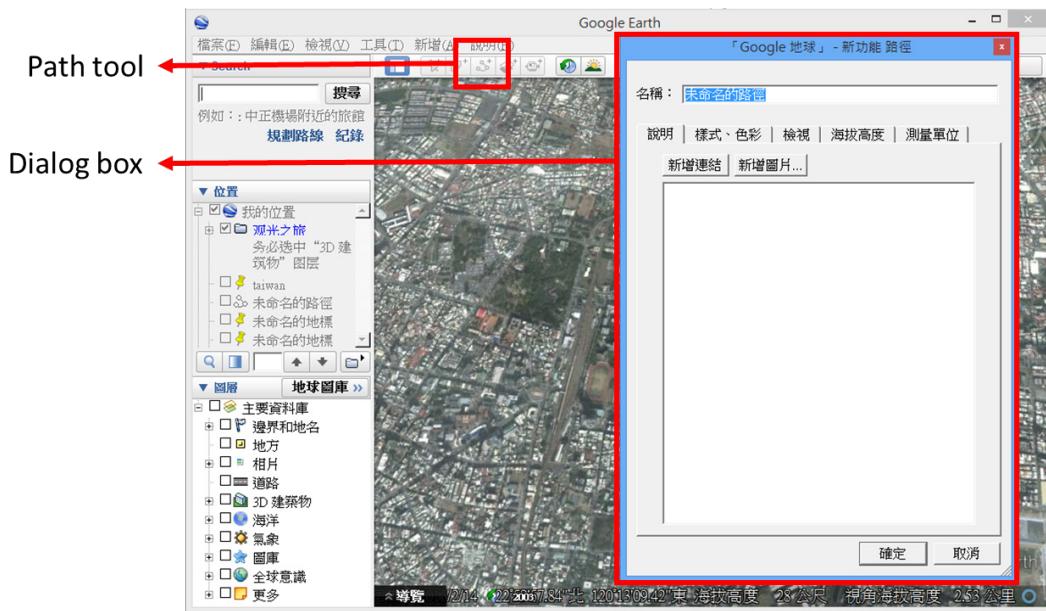


Step 2: Zooming to a target region or using the search panel to find places and managing search results.

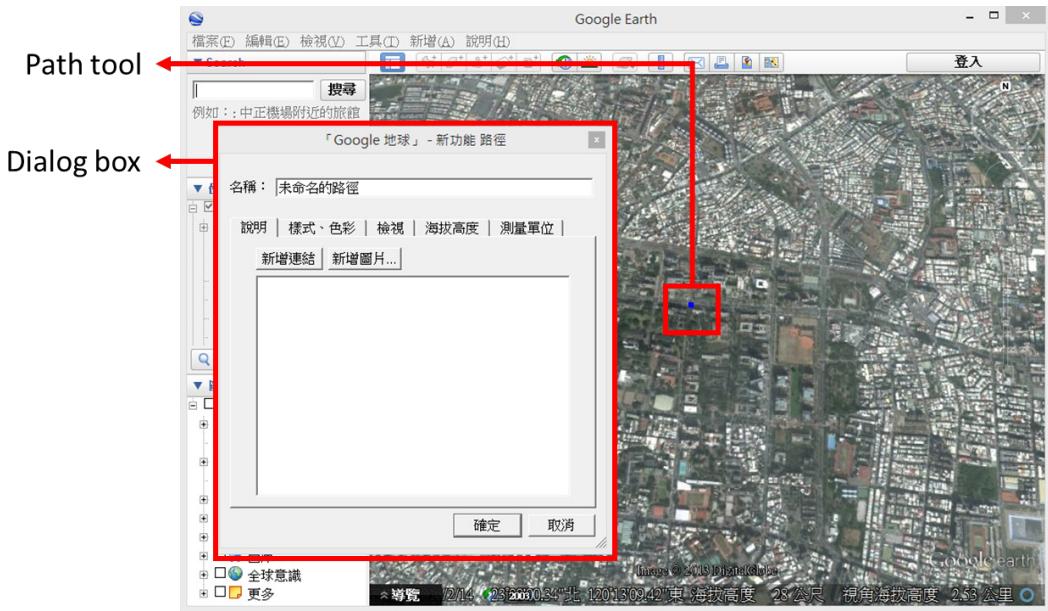




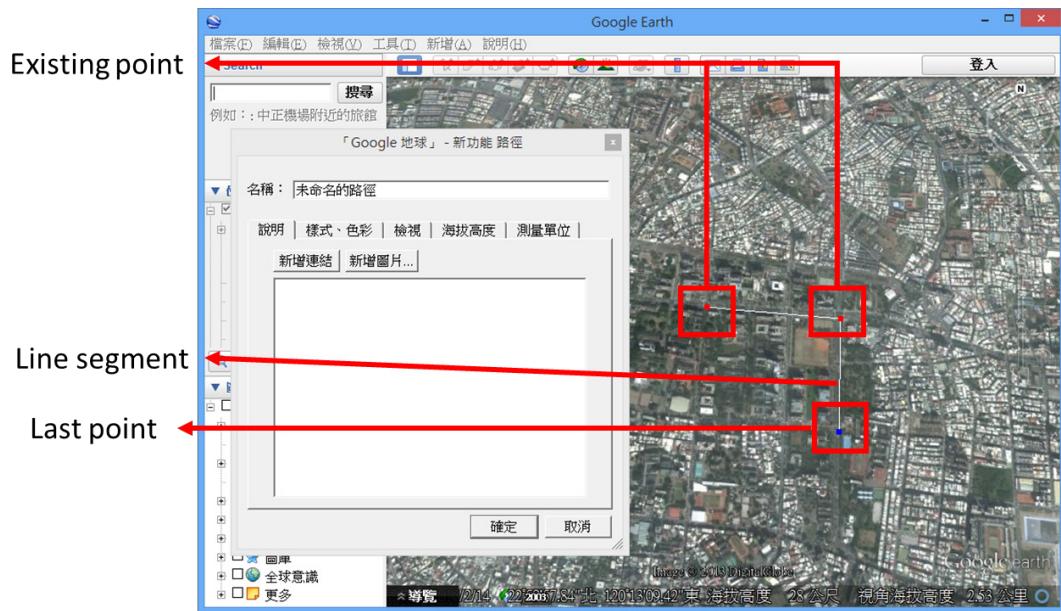
Step 3: Click the path tool to add a path (line or lines). Google earth would automatically show the dialog box.



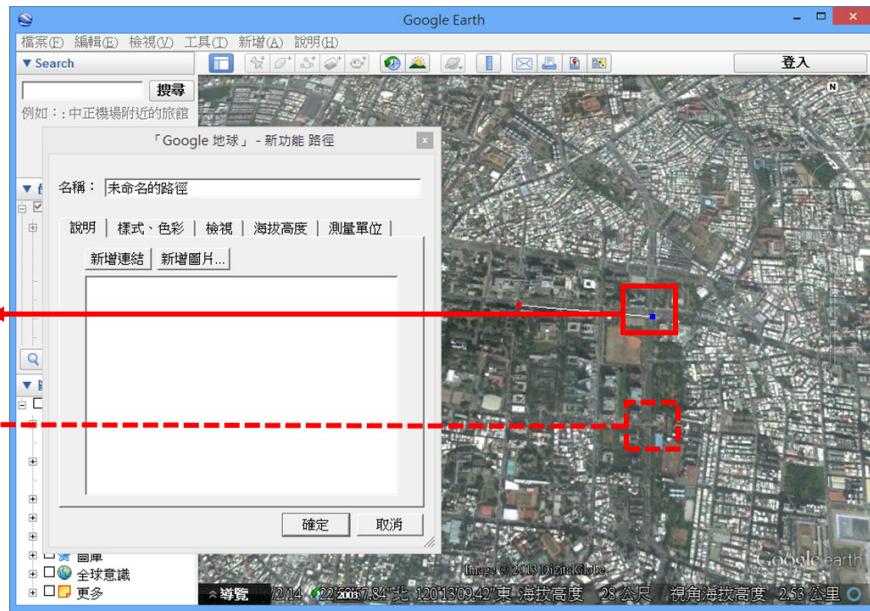
Step 4: Setting in this dialog box include Name, description, color etc. Position the beginning point (blue point) in the viewer.



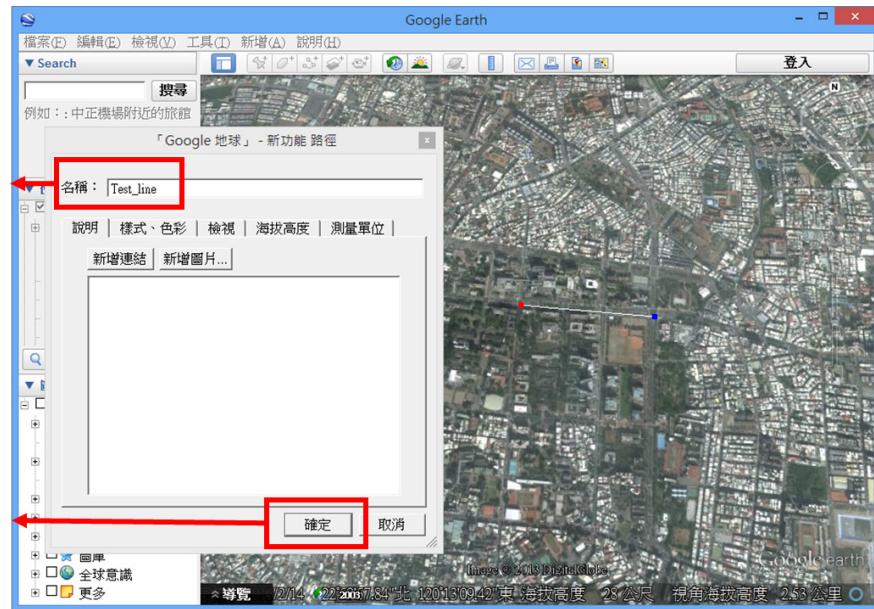
Step 5: The red dot indicates the previous point and a line connects to it as you move the Mouse.



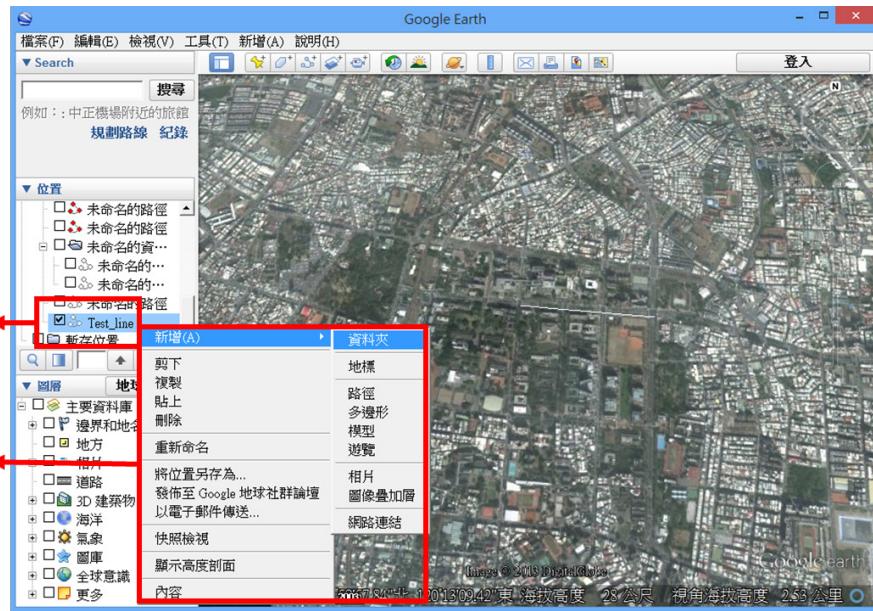
Step 6: User can press backspace on a keyboard to recovery previous situation.



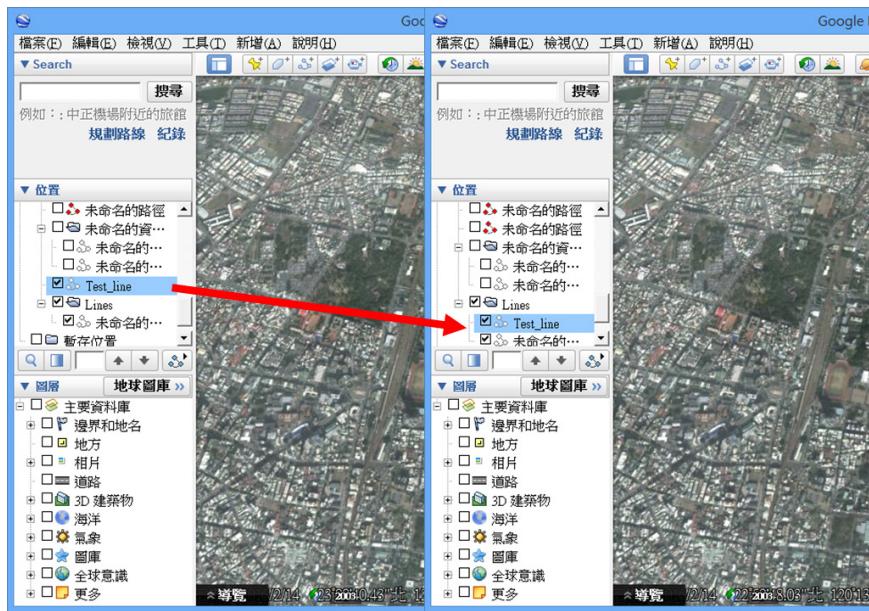
Step 7: Google earth does not support functionality of closing the line. Just press finish button to finish the procedure.



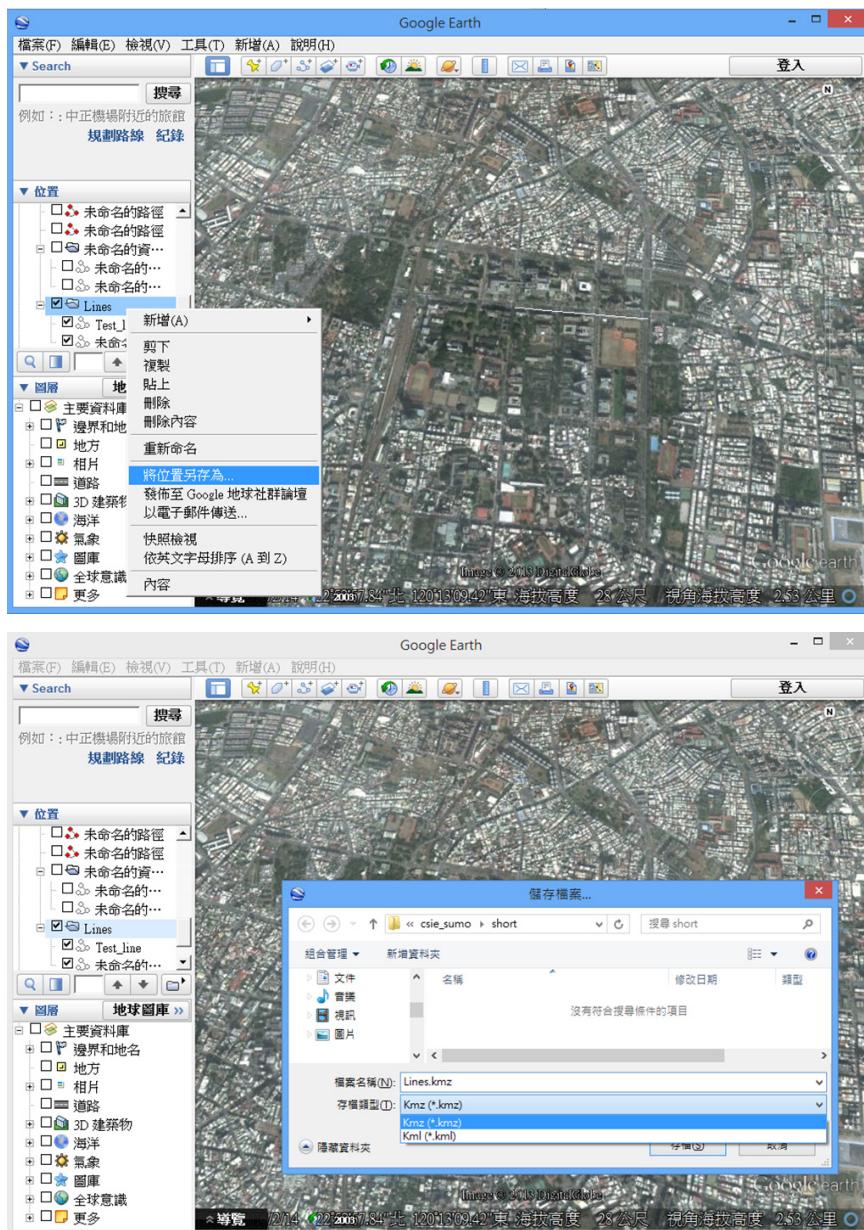
Step 8: User can create the folder to categorize a lot of line in your way.



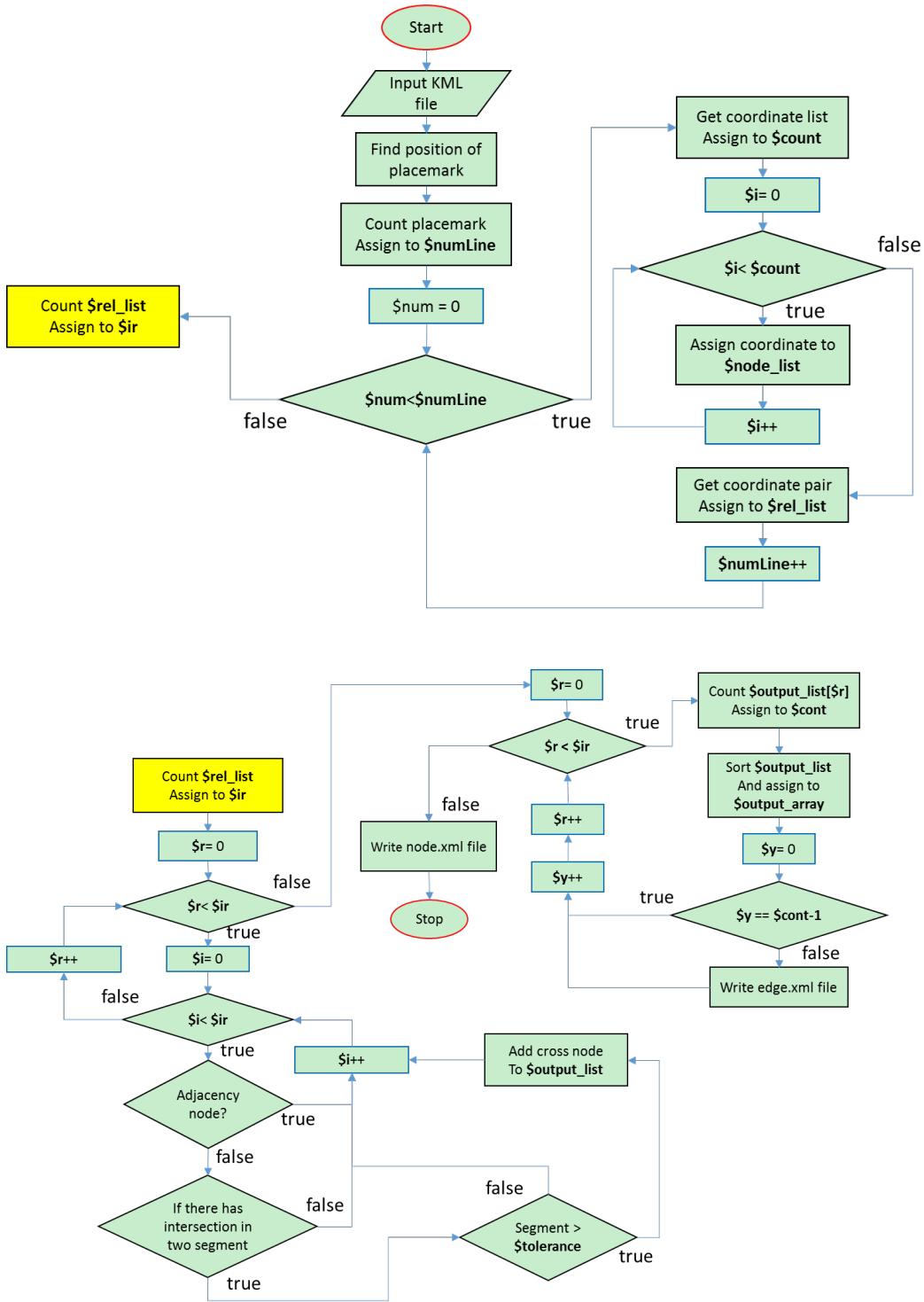
Step 9: Moving the line by using the drag and drop to the specific folder.



Step 10: Once you finish recording and have downloaded your file. In order to download KML file, right click on the folder then would show you the dialog. Click on save as... and browse the path you want to restore. **Default setting is KMZ file format, please select to the KML file format to download.**



The flowchart of converting algorithm in KML shows in the following:



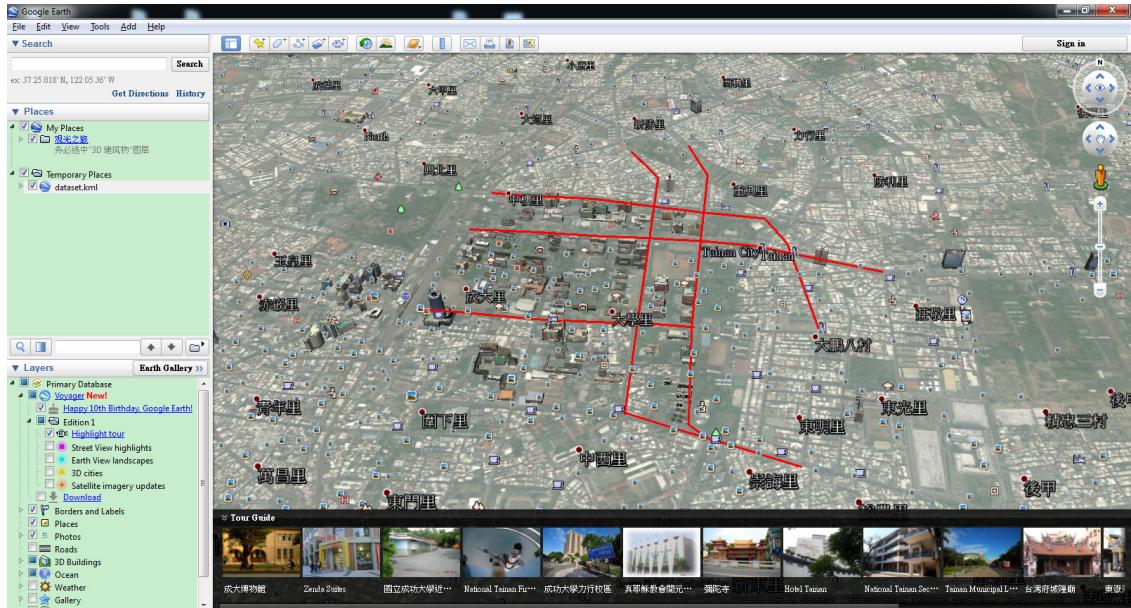
Variable:

- **\$numLine** : calculate a volume of line
- **\$count** : Get coordinate list then calculate it
- **\$node\_list** : list all node information
- **\$rel\_list** : list all segment relationship with “from” node and “to” node
- **\$ir** : the number of relationship in \$rel\_list

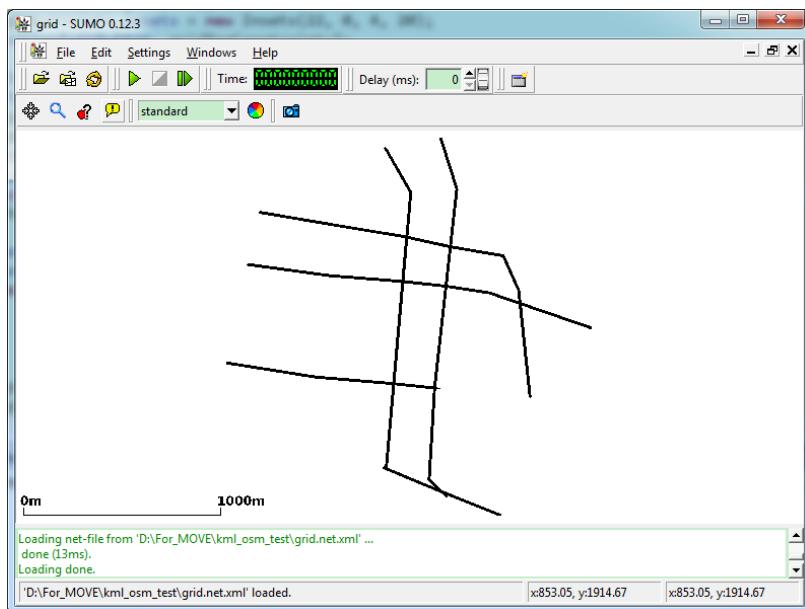
- **\$tolerance** : user input of tolerance (unit : meter)
- **\$output\_list** : record each reference of vertex in each line
- **\$output\_array** : sorting the coordinate then determining the order in each line

Once you get the KML file, you can feed the file into MOVE for map generation.

An example KML file,



The result shows in the following:

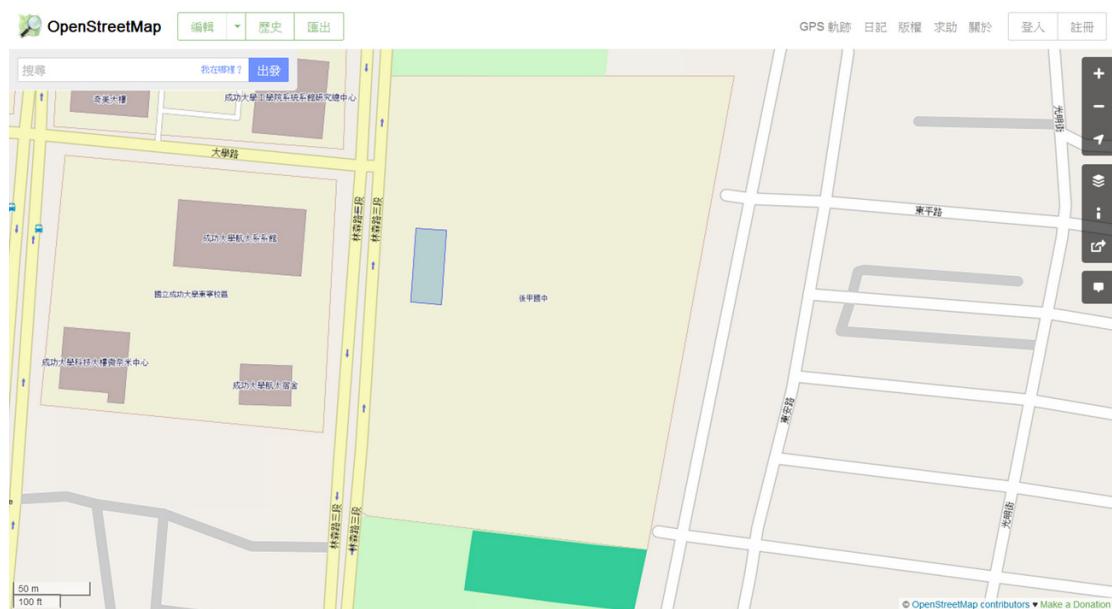


### (c) OSM map

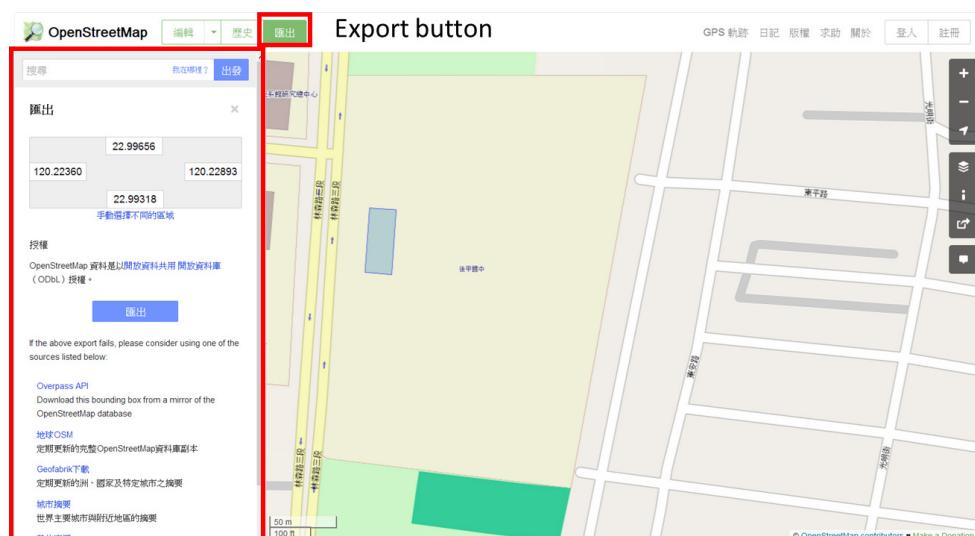
Similar, OSM also needs to generate OSM file to feed into MOVE.

Step 1: Enter into [www.openstreetmap.org](http://www.openstreetmap.org) Navigate so that the view tab shows the

area you are interested in

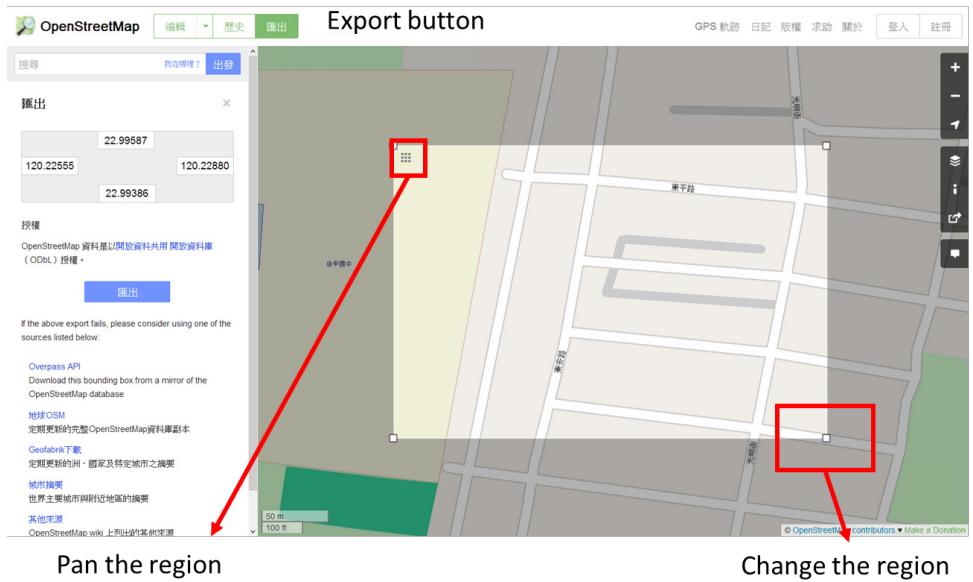


Step 2: Click the export button and webpage will display a setting panel.

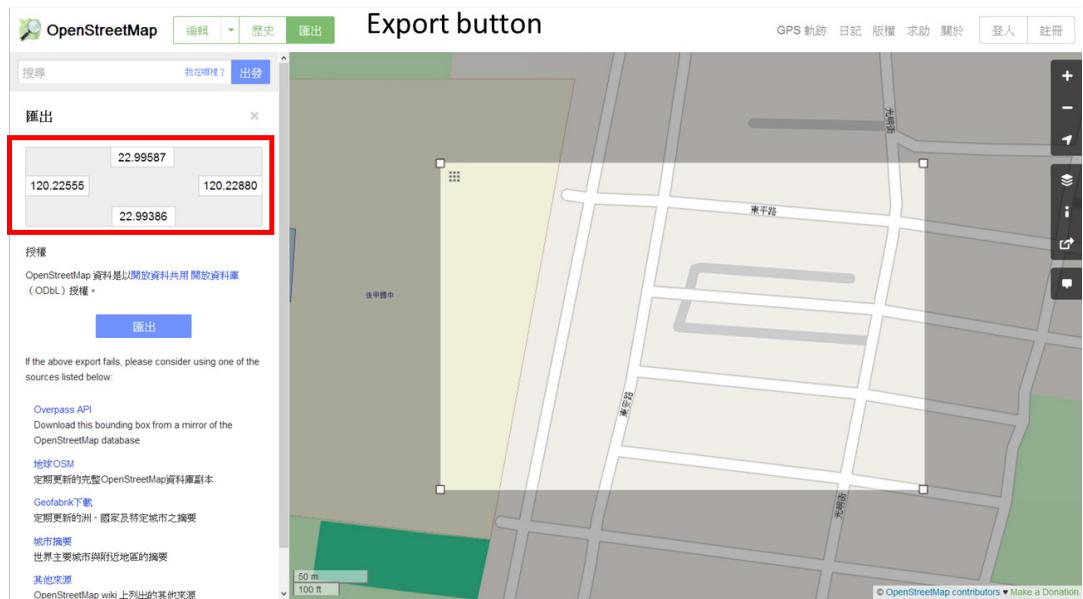


Setting panel

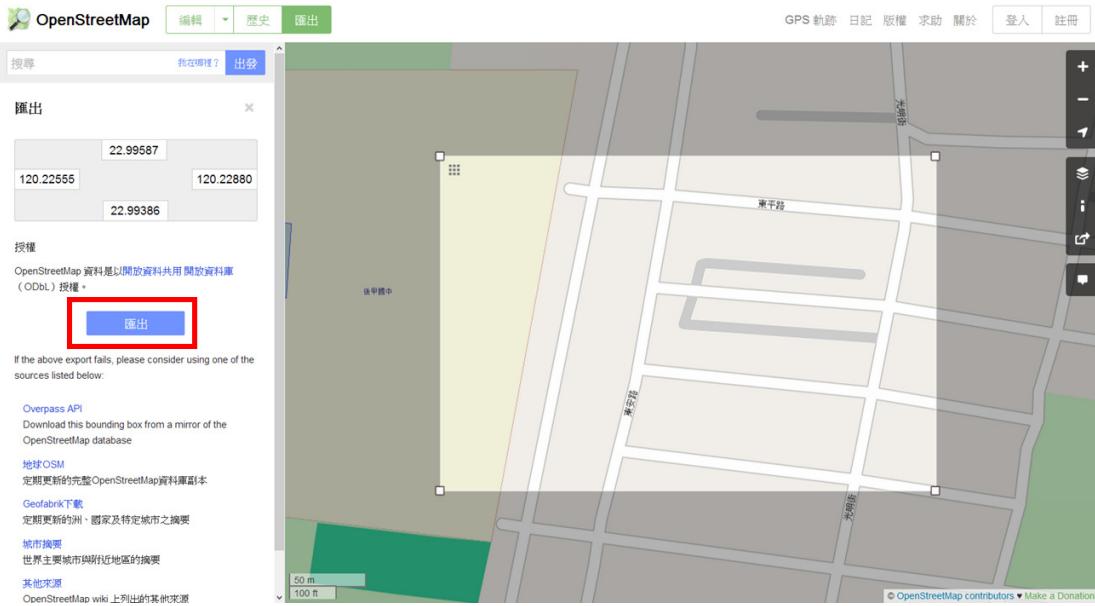
Step 3: Click manually select a different area and then drag a box to select exactly the region you want.



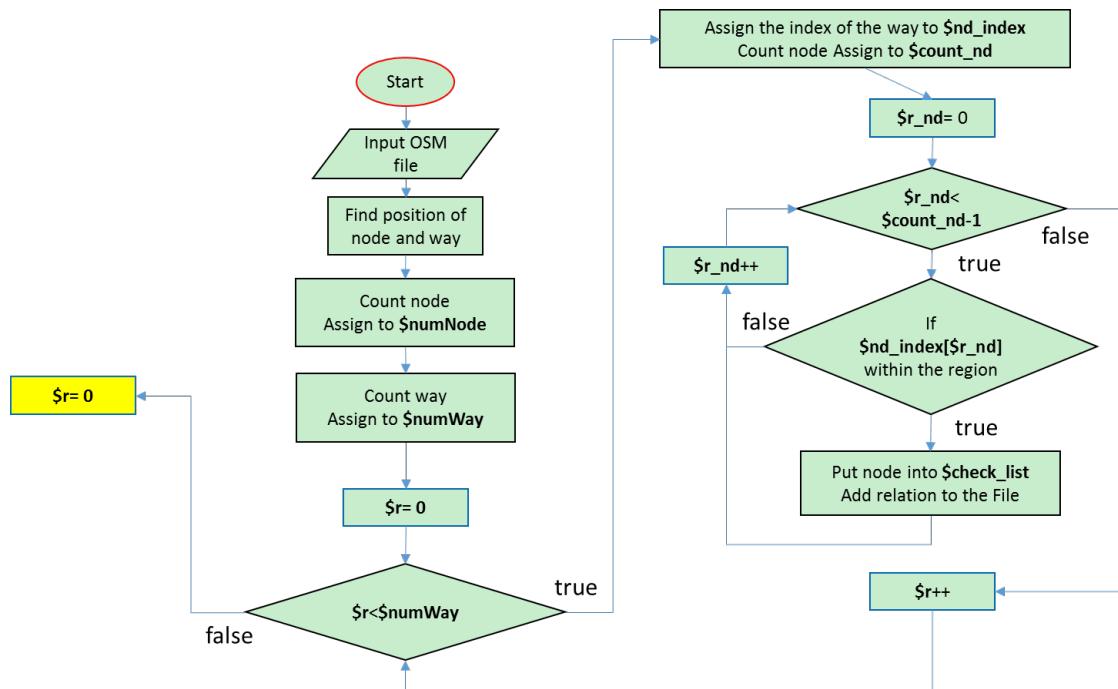
Step 4: Recording the bounding box value, which consists of a minimum and maximum latitude and longitude. It uses for make sure all feature within the specific area.

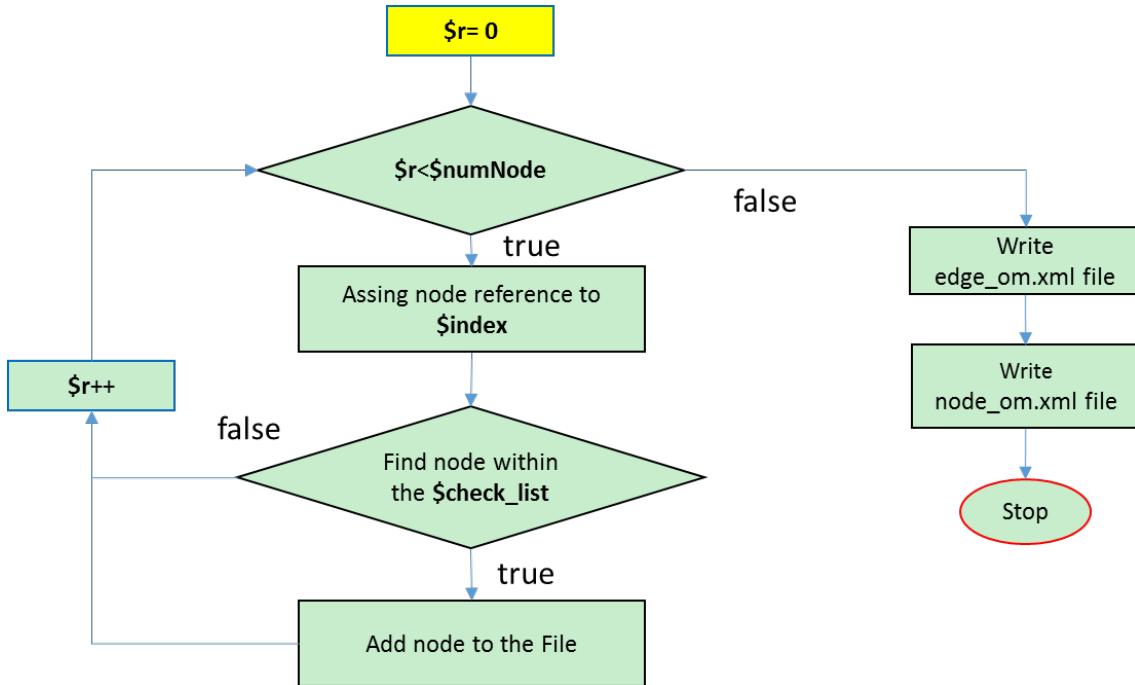


Step 5: Click the export button to download the OSM file.



The flowchart of converting algorithm in OSM shows in the following:



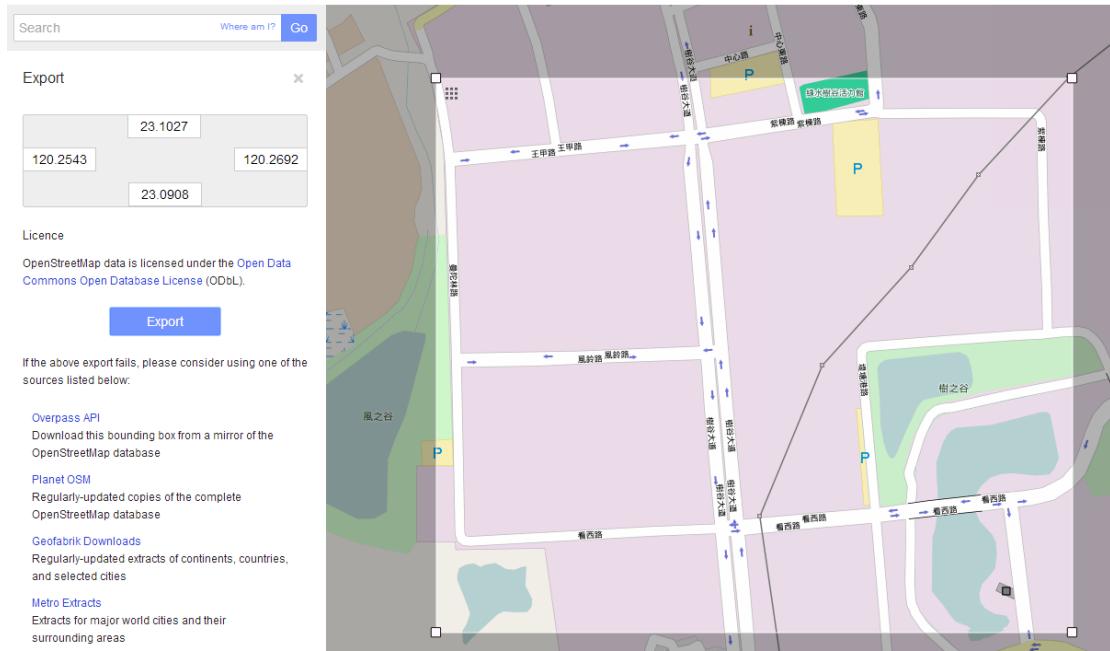


Variable:

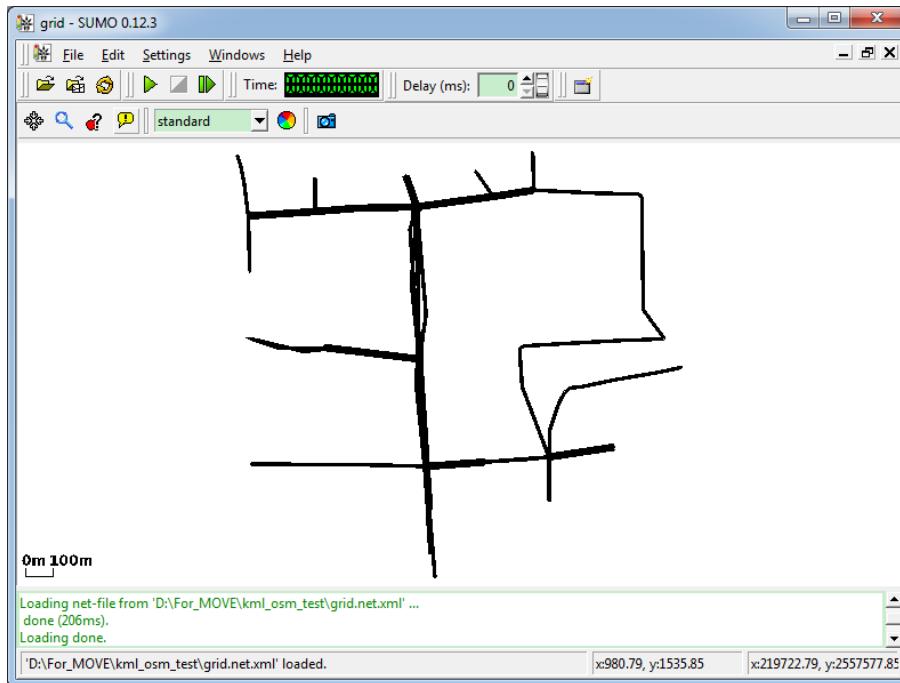
- **\$north** : north bound
- **\$south** : south bound
- **\$west** : west bound
- **\$east** : east bound
- **\$numNode** : the number of node
- **\$numWay** : the number of way
- **\$count\_nd** : the number of node in the way
- **\$nd\_index** : the index of way
- **\$index** : the node of reference id
- **\$check\_list** : the list of node within the way

Once you get the OSM file, you can feed the file into MOVE for map generation.

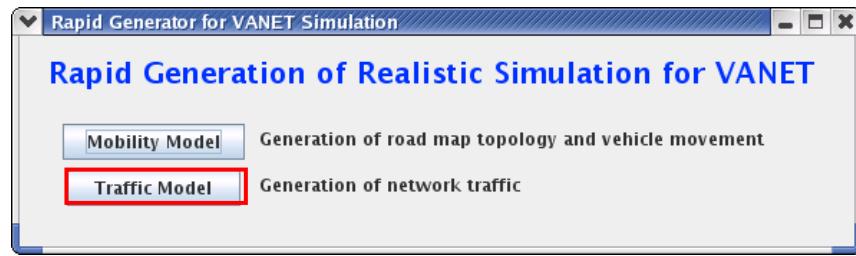
An example OSM file,



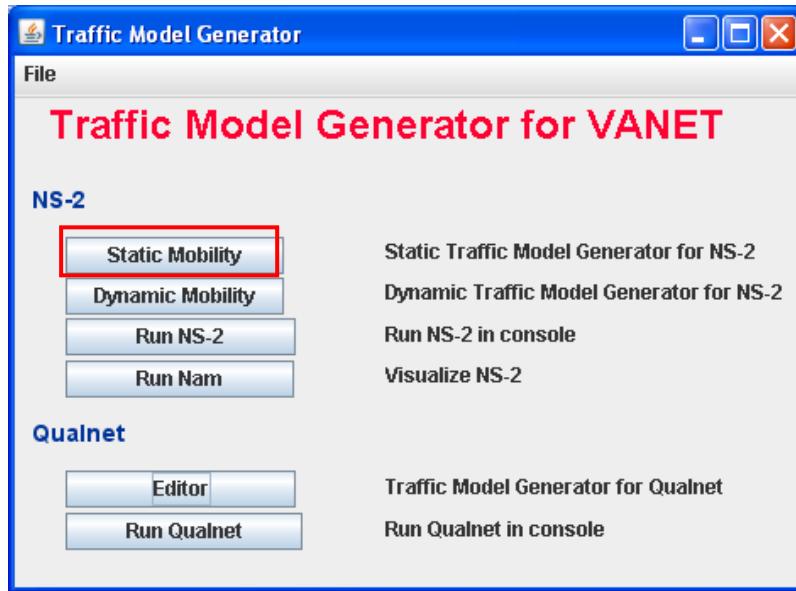
The result shows as follows.



## II Traffic Model Generation



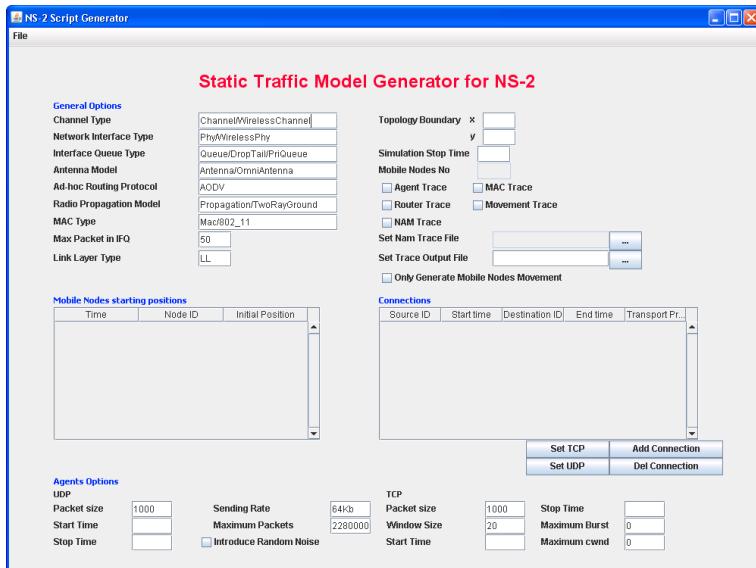
First, Select “Traffic Model” from MOVE main menu entering traffic model,



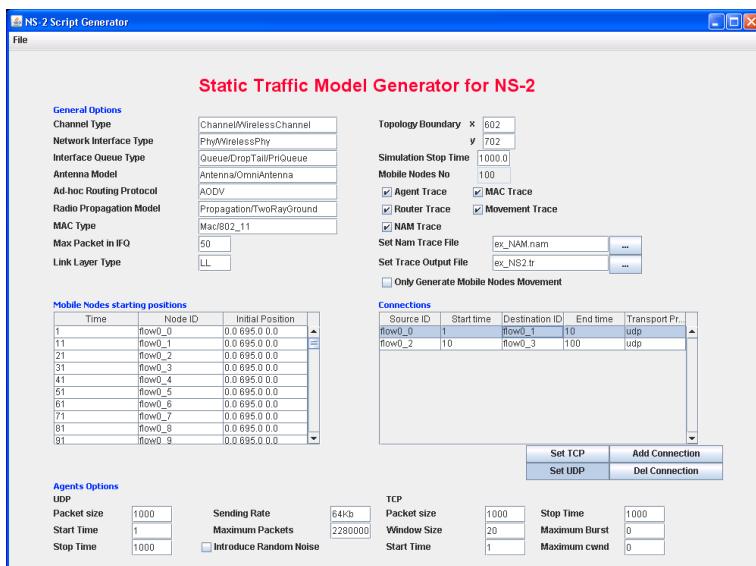
The traffic model generator consists of two main sections: for NS-2 and Qualnet. In this example, we mainly use NS-2 in the example. Right now, MOVE only supports static mobility available. We will try to deploy dynamic mobility in future.

## II.i NS-2 editor (static mobility)

### Step 1: Traffic Model Generator for NS-2 (ex\_NS2.tcl)



This editor will generate the traffic simulation file (a tcl file) for NS-2 simulation tool. First import MOVE Trace (i.e. **ex\_SUMOTRACE.sumo.tr**) and .net.xml (i.e. **ex\_Map.net.xml**) file for script generator.



Wait for loading file (it will depend on **file size** and your **computer hardware**). After loading is done, specify the options of the TCL simulation. For more details about the options, please refer to the NS-2 manual.

For the example choose NAM trace file and specify the file location. Then we add the connections between each mobile node specified in the table on the left side and assign the transport protocol (e.g. tcp/udp).

In “mobile nodes starting positions” table, you can see different time, node ID, and

initial position. You can refer this information to add connection in right-side.

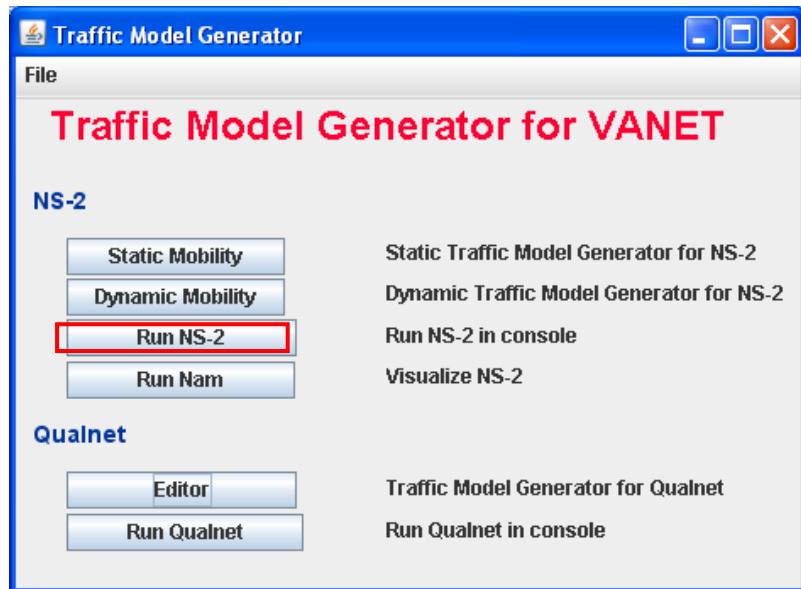
Note, the node ID depend on the name of flow in SUMO. For example, if a flow calls test0, the generated node ID should be called test0\_0, test0\_1, ..., test0\_x, where x is index in the flow.

Please check the node ID. If you input a non-exist node in the table, the TCL file will have not correct node ID.

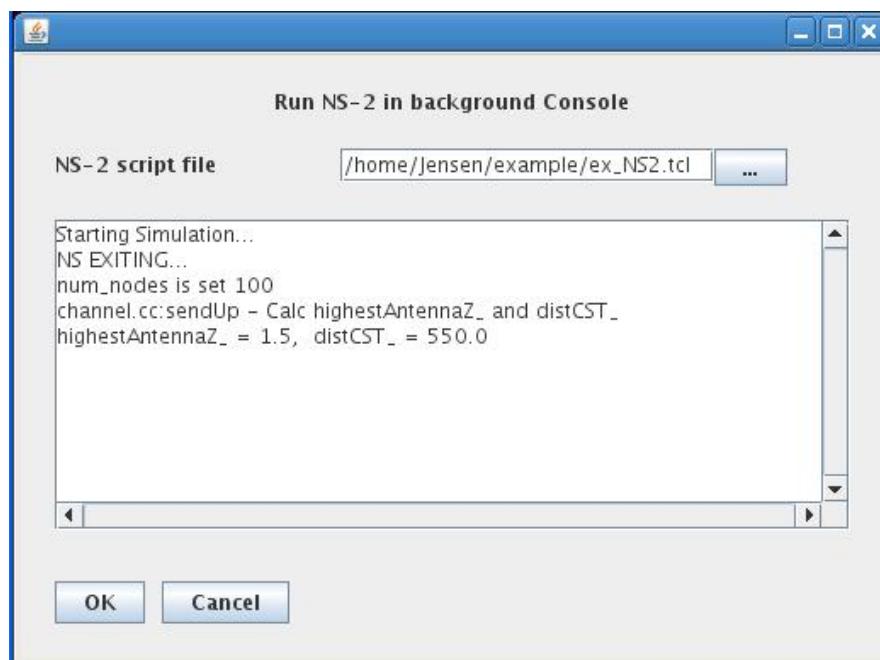
When you are done, select File->Save or Save As and put it as <name>.tcl  
**(ex\_NS2.tcl)**

NOTE: please use ns-2 in linux platform.

**Step 2: Run NS-2 in background console (only Linux platform and without TraCI) (ex\_NAM.nam and ex\_NS2.tr)**



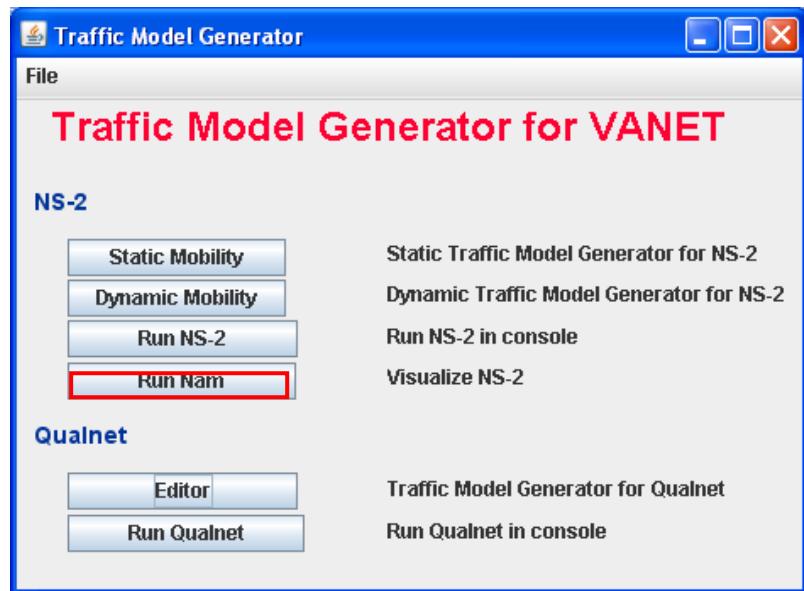
After, select “Run NS-2” runs NS-2 in console.



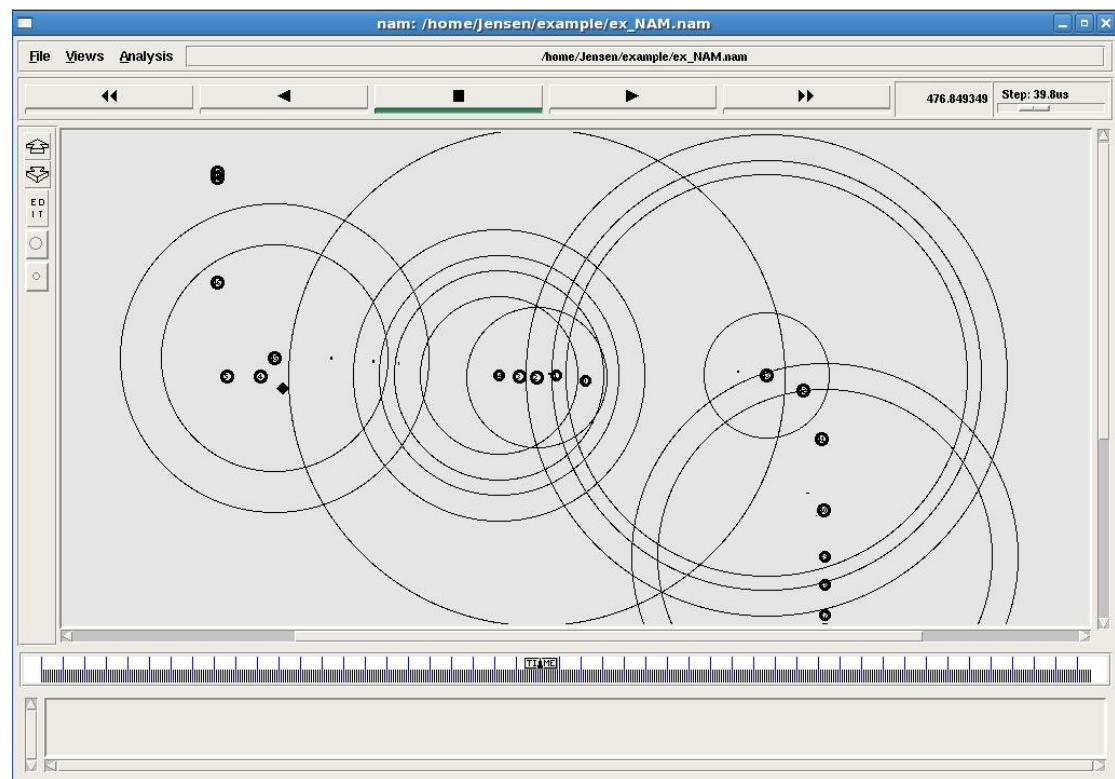
Finally, run the simulation with <name>.tcl (**ex\_NS2.tcl**)

After that, you can either run the NS-2 script in your own shell or using the program's NS-2 script runner (by choosing the Run NS-2 button from the traffic model main menu).

### Step 3: Run NAM visualize NS-2 in example (only Linux platform)



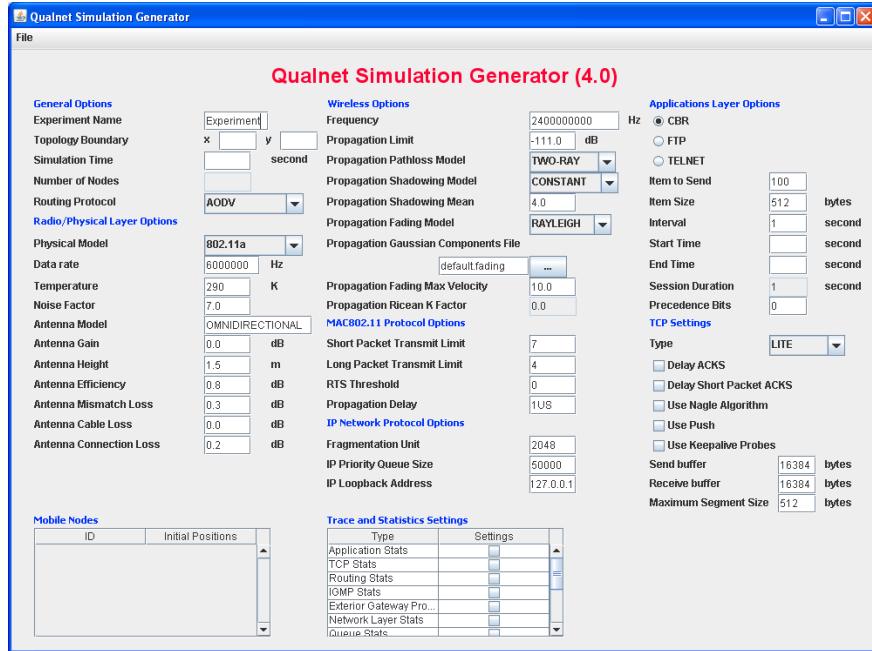
Finally, you can call the NAM trace runner from the main menu.



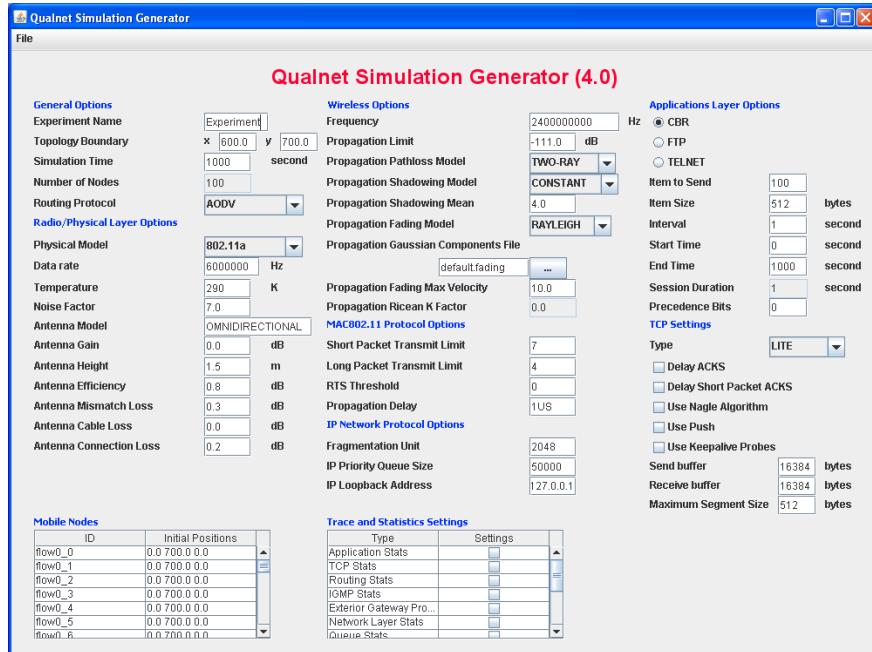
You can play NAM and see the actual movements in example.

## II.ii Qualnet editor

### Step 1: Traffic Model Generator for Qualnet



This editor will generate the traffic simulation file (config files) for Qualnet simulation tool. First import MOVE Trace (i.e. **ex\_SUMOTRACE.sumo.tr**) and .net.xml (i.e. **ex\_Map.net.xml**) file for script generator.



Wait for loading file (it will depend on **file size** and your **computer hardware**). After loading is done, specify the options of the simulation. For more details about the options, please refer to the Qualnet manual.

In here, we only generate mobility pattern in Qualnet. We also plan to create connection editor in future.

## Appendix I: Advance traffic light control

Reference link: Documentation is available at

[http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Advanced\\_Traffic\\_Lights\\_Simulation](http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Advanced_Traffic_Lights_Simulation)

First take your converted network, open it in a text editor, copy the traffic lights definition (look for ">YOUR\_TLS\_NAME<" and copy the section that is embedded in <tl-logic>), paste it into a new file, embedding it in a further xml-tag, "<additional>".

So you should have a file which looks like:

```
<additional>
CUT_TLS_DEFINITION
</additional>
```

Ok, replace the “programID” by something else. You may now change the durations of all phases. You can load this file by adding its name to the additional-section of your configuration.

For example, we change period of the traffic light in node8. You can get information from map file.

File name: *traffic\_duration.add.xml*

<additional>

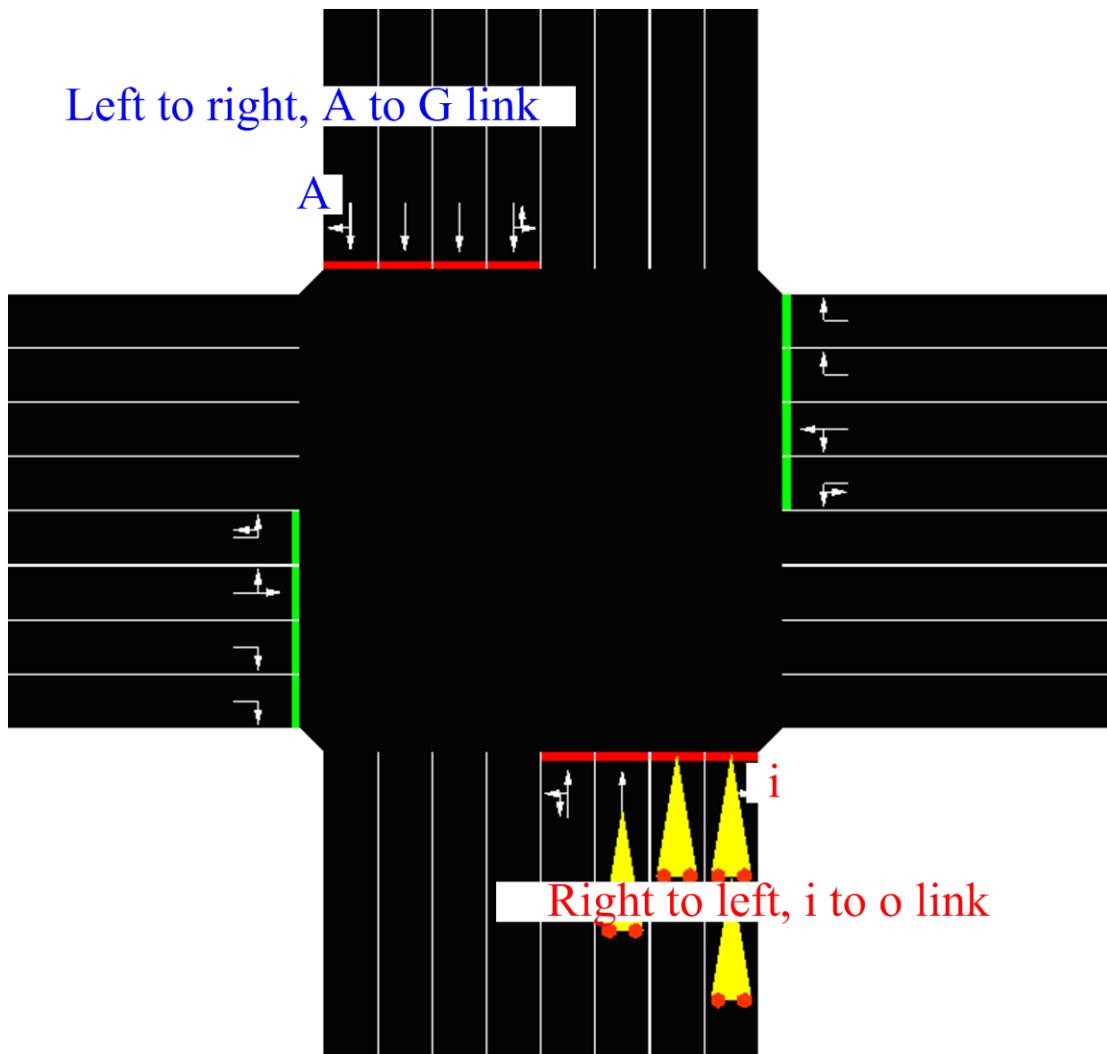
```

<tl-logic id="node8" type="static" programID="modify" offset="0">
    <phase duration="35" state="GGGGGggrrrrrrrGGGGGggrrrrrrr"/> //Green light Index A to
    index G highlight with blue color as shown in under Figure. Index I to index O highlight with red color.

    <phase duration="5" state="yyyyyggrrrrrrrryyyyggrrrrrrr"/> //Yellow light
    <phase duration="6" state="rrrrrGGrrrrrrrrrrrGGrrrrrrr"/> //Green light for turn left
    <phase duration="5" state="rrrrryyrrrrrrrrrryyrrrrrrr"/> // Yellow light for turn left
    <phase duration="31" state="rrrrrrGGG GggrrrrrrrGGG Ggg"/> // Similar as above
configuration, just different directions.

    <phase duration="5" state="rrrrrrrryyyygggrrrrrrrrryyyyggg"/>
    <phase duration="6" state="rrrrrrrrrrrGGGrfffffrrrGGG"/>
    <phase duration="5" state="rrrrrrrrrrrryyrrrrrrrrrryy"/>
</tl-logic>
```

Left to right, A to G link



File name: *ex\_SUMO.sumo.cfg*

<configuration>

```
<input>
  <net-file value="/home/Jensen/example/ex_Map.net.xml"/>
  <route-files value="/home/Jensen/example/ex_Map.net.xml"/>
  <additional-files value="/home/Jensen/example/traffic_duration.add.xml "/>
  <junction-files value="" />
</input>

<output>
  <netstate-dump value="/home/Jensen/example/grid.sumo.tr"/>
  <tripinfo-output value="output-tripinfos.xml"/>
  <emissions-output value="output-emissions.xml"/>
  <vehroute-output value="output-vehroutes.xml"/>
</output>

<time>
  <begin value="0"/>
  <end value="1000"/>
  <time-to-teleport value="-1"/>
  <srand value="23423"/>
  <route-steps value="-1"/>
</time>

<reports>
  <print-options value="false"/>
</reports>

</configuration>
```

PS: This part thanks SUMO developer Daniel to guide.

TAIWAN, NCKU university, CSIE, LENS lab.  
Updater: Chine-Ming Chou

## Appendix II: Random trip

Due to SUMO remove random routes generation; introduced a [script for generating random trips](#) instead. The script requires Python.

- Python 2.7 or later - <http://www.python.org/download/>

The script is located in: <**SUMO**>/tools/trip

"**randomTrips.py**" generates a set of random trips for a given network (option "-n"). It does so by choosing source and destination edge either uniformly at random or weighted by length (option "-l"), by number of lanes (option "-L") or both. The resulting trips are stored in an xml file (option "-o", default trips.trips.xml) suitable for the [DUAROUTER](#) which is called automatically if the "-r" option (with a filename for the resulting route file) is given. The trips are distributed evenly in an interval defined by begin (option "-b", default 0) and end time (option "-e", default 3600) in seconds. The number of trips is defined by the repetition rate (option "-p", default 1) in seconds. Every trip has an id consisting of a prefix (option "-t", default "t") and a running number. Example call:

```
randomTrips.py -n input_net.net.xml -e 1000 -l
```

Note that the script does not check whether the chosen destination may be reached from the source. This task is performed by the router.

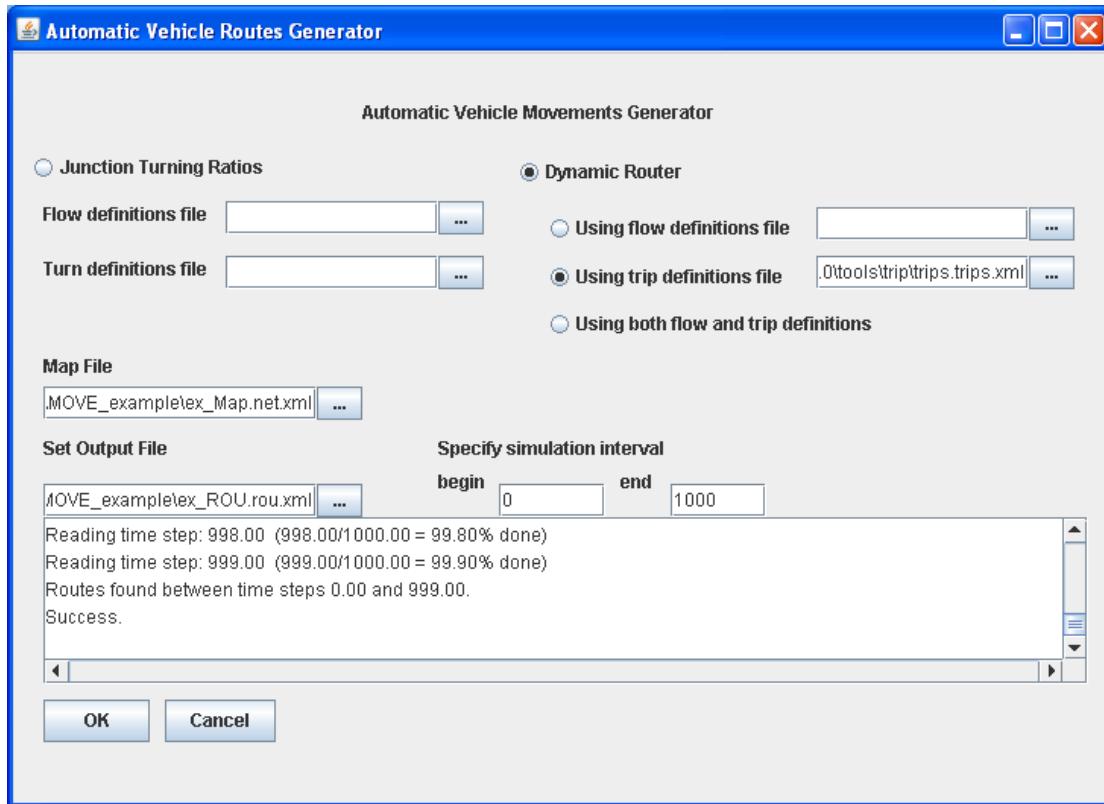
```
C:\ForMOVE\sumo-0.12.3\tools\trip>dir
 Volume in drive C is 114G
 Volume Serial Number is B6CD-9CC6

 Directory of C:\ForMOVE\sumo-0.12.3\tools\trip

2013/12/02    下午 03:16    <DIR>          .
2013/12/02    下午 03:16    <DIR>          ..
2011/02/10    上午 01:26            17,073 generateTripsXml.py
2011/12/01    上午 02:22            331,030 grid.net.xml
2011/02/10    上午 01:26            3,249 randomTrips.py
2011/02/10    上午 01:26            2,326 removeGeometryInTrips.pl
2011/02/10    上午 01:26            2,143 removeTripsIfNotInInterval.pl
2011/02/10    上午 01:26            2,343 route2trips.py
2011/02/10    上午 01:26            2,222 splitTripsByHours.pl
2013/12/02    下午 03:17            6,536 trips.trips.xml
                           8 File(s)        366,922 bytes
                           2 Dir(s)   10,919,682,048 bytes free

C:\ForMOVE\sumo-0.12.3\tools\trip>randomTrips.py -n grid.net.xml -e 100 -l
C:\ForMOVE\sumo-0.12.3\tools\trip>
```

When you generated trips file (default is **trips.trips.xml**), you can feed it into MOVE.



This part refers website of SUMO:

[http://sourceforge.net/apps/mediawiki/sumo/index.php?title=AdditionalTools\\_Trip#randomTrips.py](http://sourceforge.net/apps/mediawiki/sumo/index.php?title=AdditionalTools_Trip#randomTrips.py)

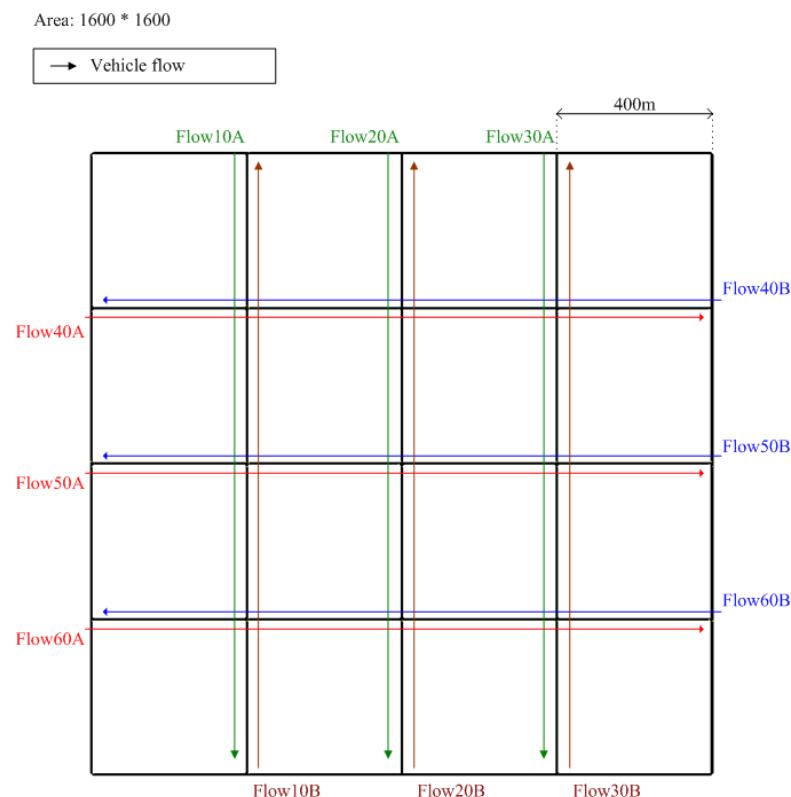
### Appendix III: TraCI using and example (for advanced user)

According to TraCI information, please refer SUMO official website.

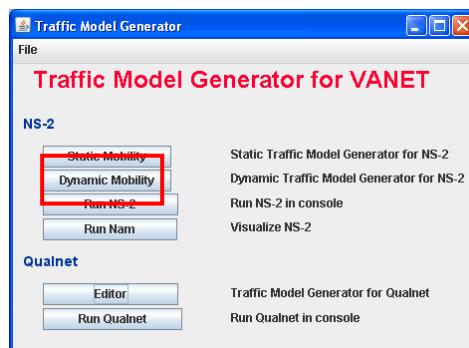
<http://sourceforge.net/apps/mediawiki/sumo/index.php?title=TraCI>

Our work is referred relation SUMO's material to implement.

Here we only provide some examples as us implemented in MOVE for TraCI. Please use our example files (for trace-version) to perform this case study. The scenario is employed a grid map as show in the following figure.



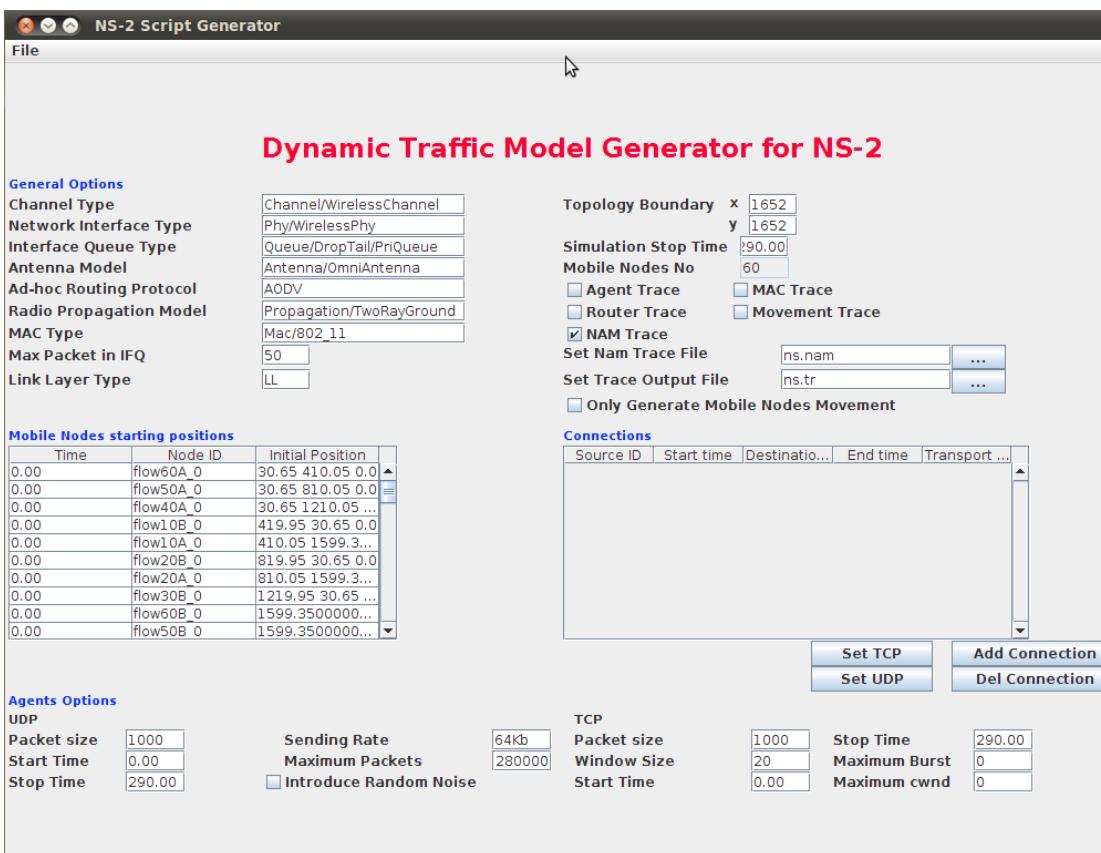
Step A: please select “dynamic mobility”



Note that, this GUI only generates TCL file for TraCI simulation.

TAIWAN, NCKU university, CSIE, LENS lab.  
Updater: Chine-Ming Chou

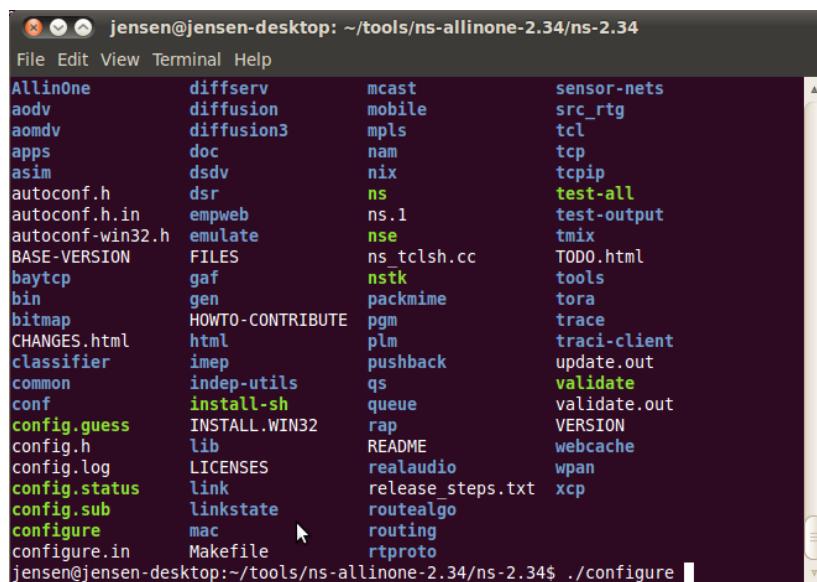
**Step B: Import your map file and route file.**



In this example, we save this file to “ex\_traci.tcl”.

Before you run simulation, you should install ns2 trace-patch.

```
# directly copy all files into your ns-2 folder (e.g. tcpip, traci-client, and Makefile.in).  
# in ns-2 folder input “./configure”
```



# and then “make”

TAIWAN, NCKU university, CSIE, LENS lab.  
Updater: Chine-Ming Chou

In this example, we will show some TraCI API as follows:

1. command-GetRoadID [nsID] [sumoID]

# Returns the id of the edge the named vehicle was at within the last step; error value:

""

```
# $ns_ at 3.0 "$mobilityInterfaceClient command-GetRoadID 6 flow20A_0"
```

2. command-SetMaxSpeed [nsID] [sumoID] [max speed]

# Sets the vehicle's maximum speed to the given value

```
# $ns_ at 5.0 "$mobilityInterfaceClient command-SetMaxSpeed 0 flow60A_0 5"
```

3. command- changeTarget [nsID] [sumoID] [target edge]

# The vehicle's destination edge is set to the given. The route is rebuilt.

```
# $ns_ at 5.0 "$mobilityInterfaceClient command-ChangeTarget 0 flow60A_0 4344-2"
```

4. command-changeRoute [nsID] [sumoID] [number of edges] [edges lists]

# Assigns the list of edges as the vehicle's new route assuming the first edge given is the one the vehicle is currently at: The first occurrence of the edge is currently at is searched within the new route; the vehicle continues the route from this point in the route from. If the edge the vehicle is currently does not exist within the new route, an error is generated.

```
# $ns_ at 5.0 "$mobilityInterfaceClient command-changeRoute 0 flow60A_0 7 0111-1,1011-2,1020-1,2021-1,2131-1,3141-1,3141-2"
```

### Step C: Execution TraCI server (Important)

Before you run ns-2 simulation, you need to execute TraCI server side every time.

The command format is

```
# Sumo -n [your map file] -r [your route file] --remote-port 8888 --penetration 1 --route-steps -1
```

For our example, you should input

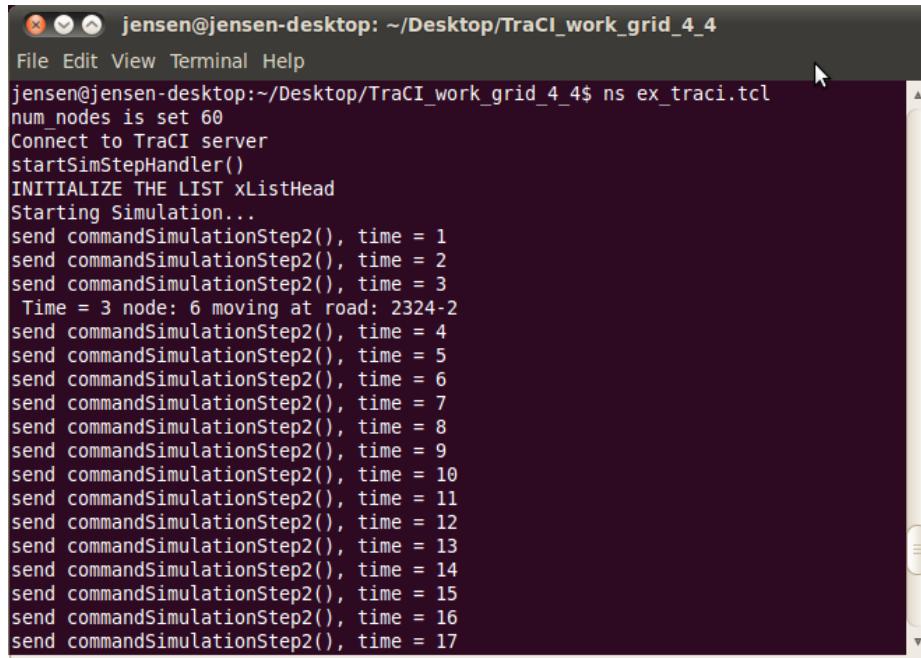
```
# Sumo -n /yourPath/TraCI_work_grid_4_4/grid.net.xml -r /yourPath/TraCI_work_grid_4_4/grid.rou.xml --remote-port 8888 --penetration 1 --route-steps -1
```

#### Step D: Test command-GetRoadID

Please remove comment (i.e. #) in “ex\_traci.tcl”.

```
$ns_ at 3.0 "$mobilityInterfaceClient command-GetRoadID 6 flow20A_0"
```

You will get road ID “2324-2” as show in the following figure.



A screenshot of a terminal window titled "jensen@jensen-desktop: ~/Desktop/TraCI\_work\_grid\_4\_4". The window shows the command "ns ex\_traci.tcl" being run. The output indicates the simulation is starting and a node is moving along road "2324-2" at various time steps from 1 to 17. The terminal has a dark background with white text and standard OS X window controls.

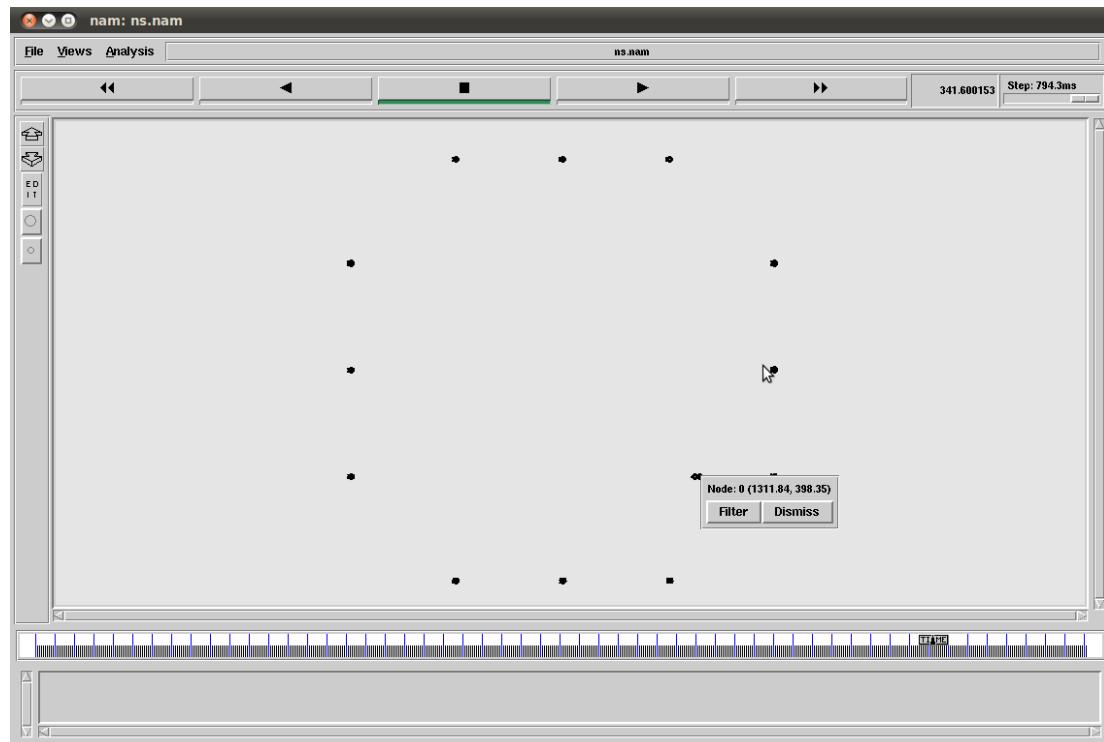
```
jensen@jensen-desktop:~/Desktop/TraCI_work_grid_4_4$ ns ex_traci.tcl
num nodes is set 60
Connect to TraCI server
startSimStepHandler()
INITIALIZE THE LIST xListHead
Starting Simulation...
send commandSimulationStep2(), time = 1
send commandSimulationStep2(), time = 2
send commandSimulationStep2(), time = 3
  Time = 3 node: 6 moving at road: 2324-2
send commandSimulationStep2(), time = 4
send commandSimulationStep2(), time = 5
send commandSimulationStep2(), time = 6
send commandSimulationStep2(), time = 7
send commandSimulationStep2(), time = 8
send commandSimulationStep2(), time = 9
send commandSimulationStep2(), time = 10
send commandSimulationStep2(), time = 11
send commandSimulationStep2(), time = 12
send commandSimulationStep2(), time = 13
send commandSimulationStep2(), time = 14
send commandSimulationStep2(), time = 15
send commandSimulationStep2(), time = 16
send commandSimulationStep2(), time = 17
```

### Step E: Test command-SetMaxSpeed

Please remove comment (i.e. #) in “ex\_traci.tcl”.

```
# $ns_ at 5.0 "$mobilityInterfaceClient command-SetMaxSpeed 0 flow60A_0 5"
```

Once you change the vehicle speed, which node will by specific speed until you change it again. The results show in the following figure. Node 0 moves slow in this simulation due to maximum speed is 5 meter/sec.

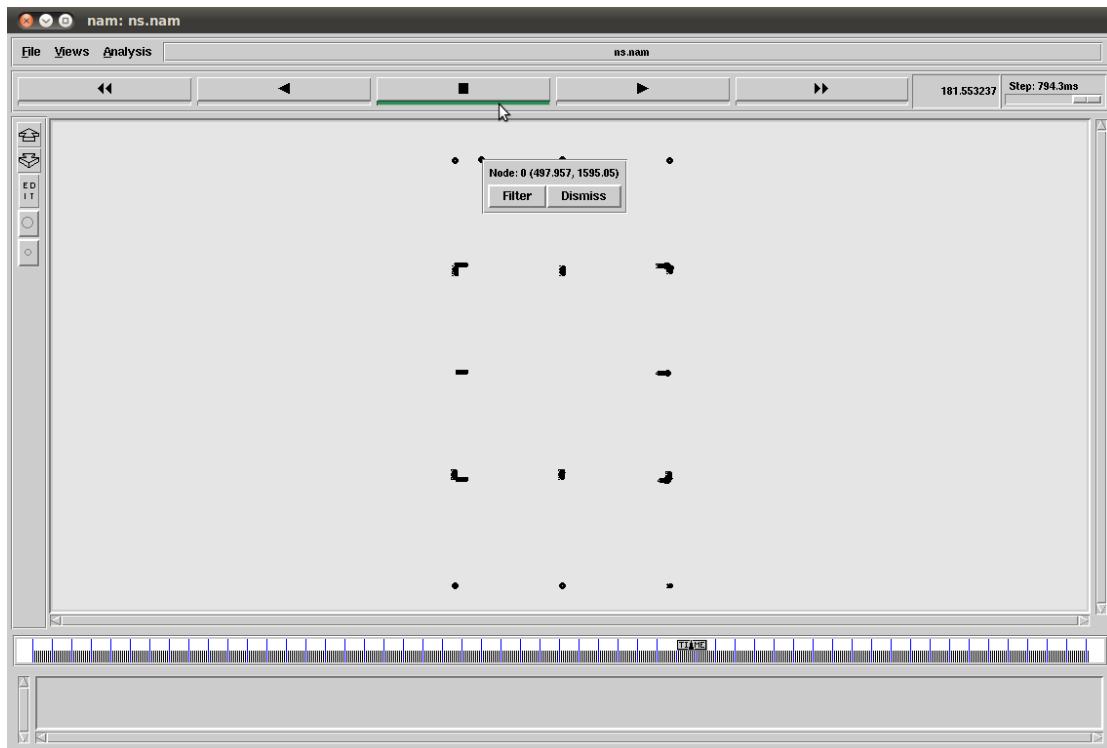


### Step F: Test command-ChangeTarget

Please remove comment (i.e. #) in “ex\_traci.tcl”.

```
# $ns_ at 5.0 "$mobilityInterfaceClient command-ChangeTarget 0 flow60A_0  
4344-2"
```

When you perform this command, node will change route by the shortest path (auto calculated by SUMO). In this example, node 0 (i.e. flow60A\_0 will move different path to target edge “4344-2”.



### Step G: Test command-changeRoute

Please remove comment (i.e. #) in “ex\_traci.tcl”.

```
# $ns_ at 5.0 "$mobilityInterfaceClient command-changeRoute 0 flow60A_0 7  
0111-1,1011-2,1020-1,2021-1,2131-1,3141-1,3141-2"
```

For this command, you need to input the vehicle “current edge” for first edge (i.e. 0111-1), and then input each edge from current edge to destination edge. For example, 0111-1,1011-2,1020-1,2021-1,2131-1,3141-1,3141-2, here 0111-1 is current edge and 3141-2 is destination edge.

The results show in the following. Node 0 is moving to the other way (highlight with red circle)

