

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

**ОТЧЕТ
ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент:

Косолапов Степан Эдуардович

Группа: НПИбд-01-20

МОСКВА

2023 г.

Постановка задачи

Установить и изучить среду для моделирования сетей VANET – SUMO.

Установить ns3 и приобрести навыки работы с ns3.

Выполнение работы

1. Установка SUMO

Выполнять установку будем на операционной системе macOS.

1. Установим sumo через менеджера пакетов brew. Для этого переходим на официальный сайт SUMO и проходим по указанным шагам:

<https://sumo.dlr.de/docs/Installing/index.html#macos>.

2. Обновляем brew:

```
brew update
```

3. Устанавливаем XQuartz:

```
brew install --cask xquartz
```

4. Устанавливаем SUMO:

```
brew tap dlr-ts/sumo
```

```
brew install sumo
```

5. Добавляем в файл конфигурации shell(мы используем zsh) переменную окружения SUMO_HOME:

```
export SUMO_HOME="/usr/local/opt/sumo/share/sumo"
```

Проверяем, что переменная добавилась:

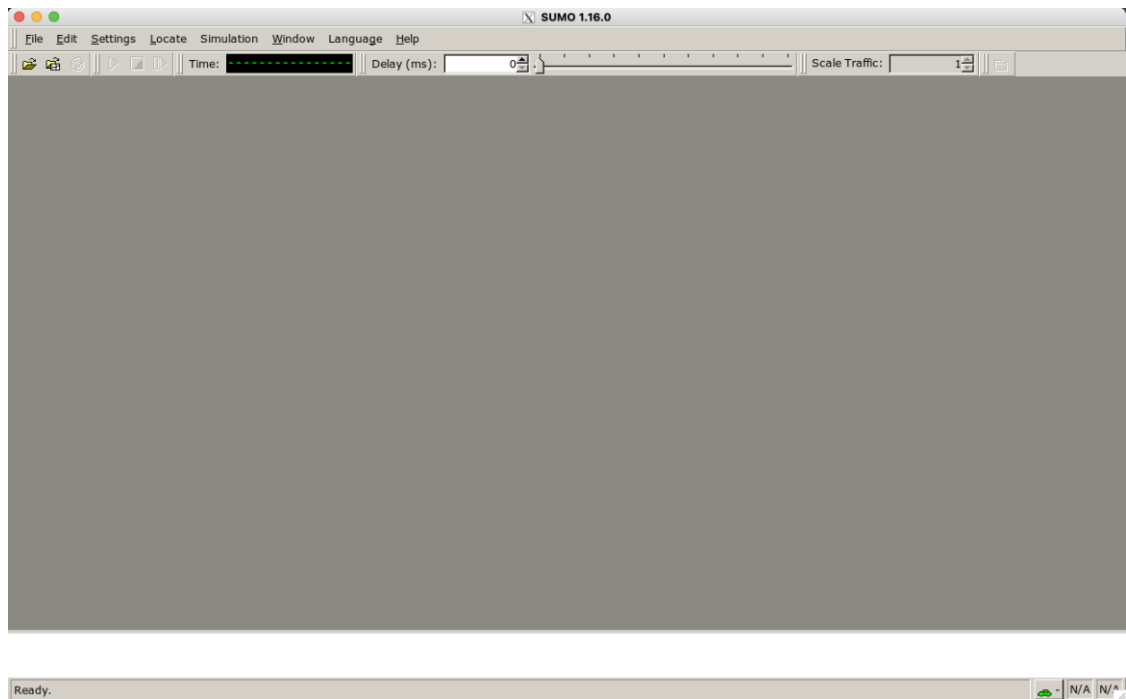
```
→ practice git:(master) echo $SUMO_HOME  
/usr/local/opt/sumo/share/sumo
```

6. Далее установим графический интерфейс для sumo – sumo-gui. Делаем это так же через менеджера пакетов brew:

```
brew install --cask sumo-gui
```

7. Теперь надо перезапустить операционную систему.

8. После этого нам доступно приложение sumo-gui. Открываем его:

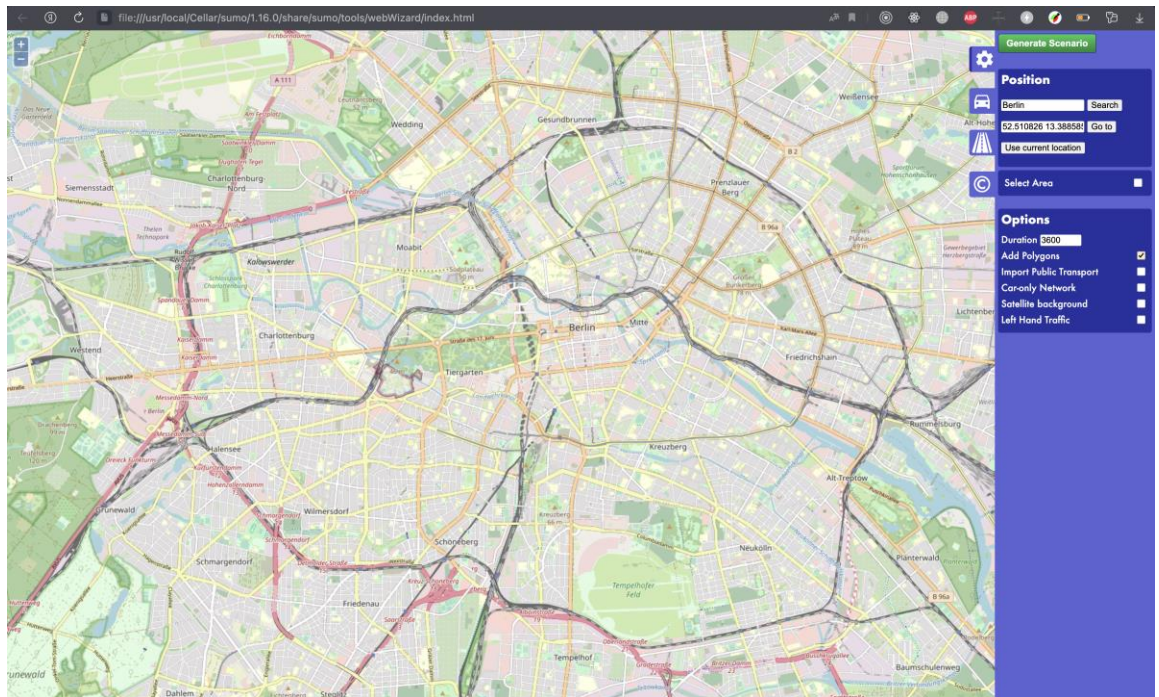


2. Симуляция в SUMO

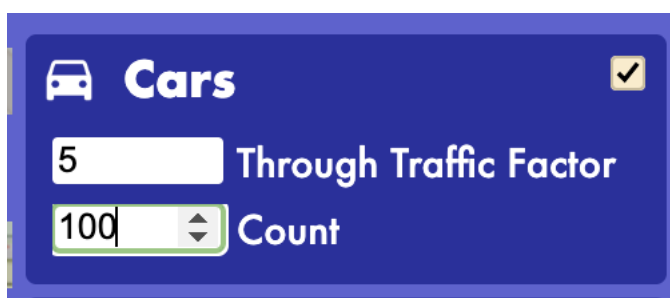
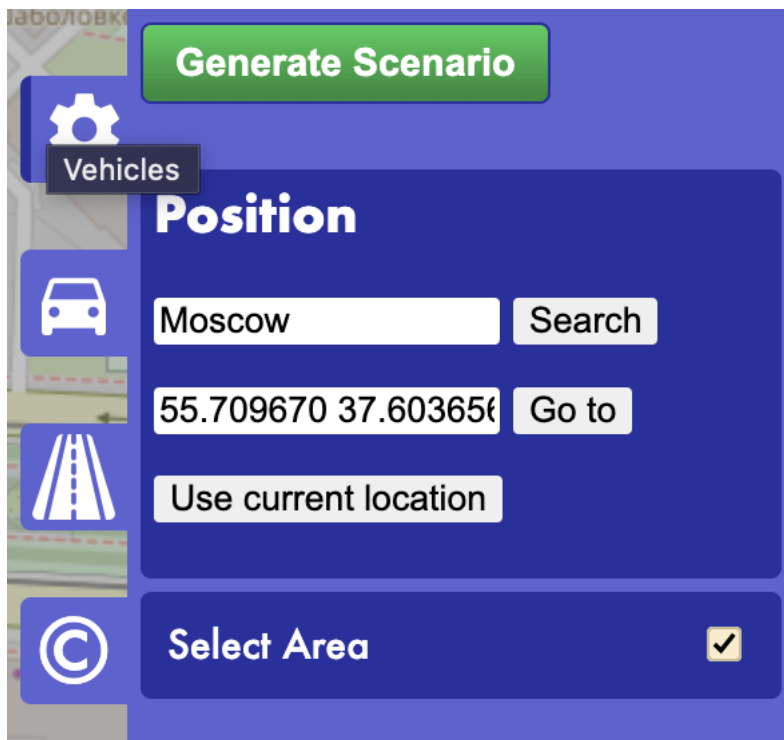
1. Для ознакомления попробуем запустить какую-нибудь симуляцию.
Есть много способов создать файл симуляции, но самый быстрый для нас сейчас – это использовать [Web Wizard](#). Это программа, позволяющая нам создать файл симуляции через графический интерфейс. При этом мы можем просимулировать практически любое место на карте. Web Wizard использует [openstreetmap](#) для создания симуляции.
2. Чтобы запустить Web Wizard – откроем официальную документацию и пройдем по шагам:
<https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html>
3. Нам нужно запустить файл `osmWebWizard.py`. Он лежит в директории `tools`.

```
→ sumo pwd
/usr/local/opt/sumo/share/sumo
→ sumo cd tools
→ tools ls | grep WebWizard
osmWebWizard.py
```

Запускаем этот файл через интерпретатор python:

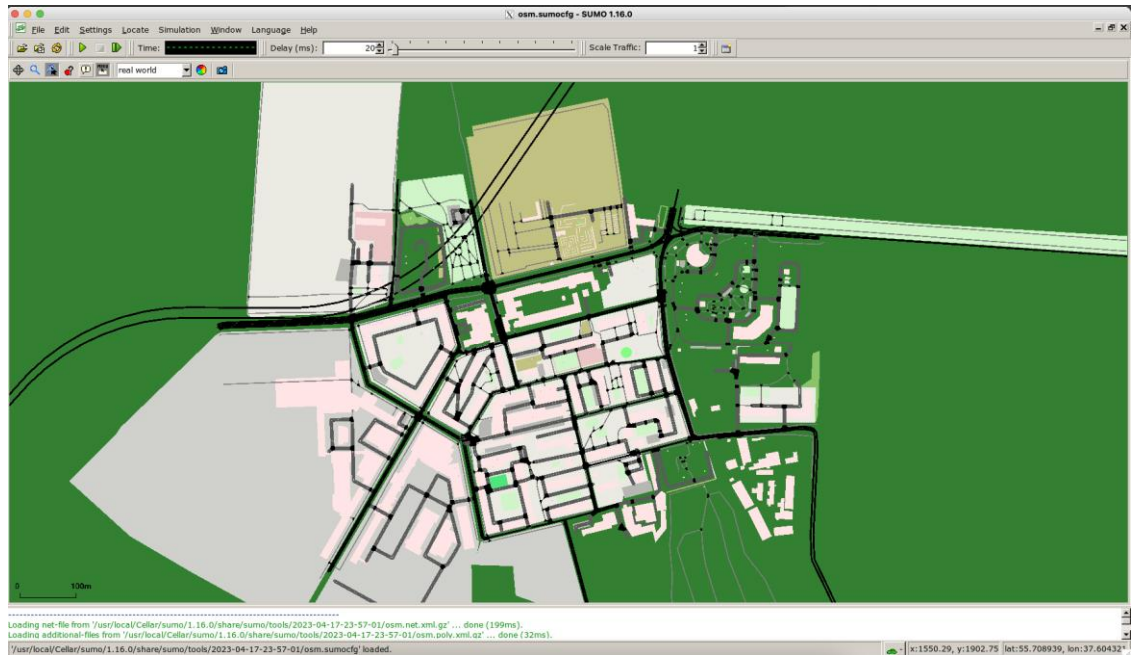


4. Открывается графический интерфейс. Здесь мы можем выбрать интересующие нас параметры. Например город и количество автомобилей на дорогах:



5. Нажимаем generate scenario и файл с сгенерированной сетью

открывается в sumo:



6. В консоли снизу мы видим откуда был открыт файл.

```
Loading net-file from '/usr/local/Cellar/sumo/1.16.0/share/sumo/tools/2023-04-17-23-57-01/osm.net.xml.gz' ... done (199ms).  
Loading additional-files from '/usr/local/Cellar/sumo/1.16.0/share/sumo/tools/2023-04-17-23-57-01/osm.poly.xml.gz' ... done (32ms).  
Loading done.
```

```
→ sumo cd /usr/local/Cellar/sumo/1.16.0/share/sumo/tools/2023-04-17-23-57-01  
→ 2023-04-17-23-57-01 ls  
build.bat          osm.polycfg  
osm.net.xml.gz     osm.sumocfg  
osm.netccfg        osm.view.xml  
osm.passenger.trips.xml osm_bbox.osm.xml.gz  
osm.poly.xml.gz    run.bat
```

7. Для интереса мы можем его изучить:

Мы видим, что это файл xml. В его начале есть какая-то конфигурация:

```
?xml version="1.0" encoding="UTF-8"?>

<!-- generated on 2023-04-17 23:57:03 by Eclipse SUMO netconvert Version 1.16.0
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/net
convertConfiguration.xsd">

  <input>
    <type-files value="{SUMO_HOME}/data/typemap/osmNetconvert.typ.xml"/>
    <osm-files value="osm_bbox.osm.xml.gz"/>
  </input>

  <output>
    <output-file value="osm.net.xml.gz"/>
    <output.street-names value="true"/>
    <output.original-names value="true"/>
  </output>

  <projection>
    <proj.utm value="true"/>
  </projection>

  <processing>
    <geometry.remove value="true"/>
    <roundabouts.guess value="true"/>
  </processing>

  <tls_building>
    <tls.discard-simple value="true"/>
    <tls.join value="true"/>
    <tls.guess-signals value="true"/>
    <tls.default-type value="actuated"/>
  </tls_building>

  <ramp_guessing>
    <ramps.guess value="true"/>
  </ramp_guessing>

  <junctions>
    <junctions.join value="true"/>
    <junctions.corner-detail value="5"/>
  </junctions>

</configuration>
```

Далее идет само описание сгенерированной сети:

```
net version="1.16" junctionCornerDetail="5" limitTurnSpeed="5.50" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/net_file.xsd">
  <location netOffset="-410754.35,-6172665.82" convBoundary="0.00,0.00,2802.57,4246.34" origBoundary="37.579848,55.691812,37.624164,55.730058" projParameter="+proj=utm +zone=37 +ellps=WGS84 +datum=WGS84 +units=m
+no_defs"/>

  <type id="highway.bridleway" priority="1" numLanes="1" speed="2.78" allow="pedestrian" oneway="1" width="2.00"/>
  <type id="highway.bus_guideway" priority="1" numLanes="1" speed="27.78" allow="bus" oneway="1"/>
  <type id="highway.cycleway" priority="1" numLanes="1" speed="5.56" allow="bicycle" oneway="0" width="1.00"/>
  <type id="highway.footway" priority="1" numLanes="1" speed="2.78" allow="pedestrian" oneway="1" width="2.00"/>
  <type id="highway.ford" priority="1" numLanes="1" speed="2.78" allow="army" oneway="0"/>
  <type id="highway.living_street" priority="2" numLanes="1" speed="2.78" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.motorway" priority="14" numLanes="2" speed="39.44" allow="private emergency authority army vip passenger hov taxi bus coach delivery truck trailer motorcycle evehicle custom1 custom2" oneway=
"1"/>
  <type id="highway.motorway_link" priority="9" numLanes="1" speed="22.22" allow="private emergency authority army vip passenger hov taxi bus coach delivery truck trailer motorcycle evehicle custom1 custom2" on
eway="1"/>
  <type id="highway.path" priority="1" numLanes="1" speed="5.56" allow="pedestrian bicycle" oneway="0" width="2.00"/>
  <type id="highway.pedestrian" priority="1" numLanes="1" speed="2.78" allow="pedestrian" oneway="1" width="2.00"/>
  <type id="highway.primary" priority="12" numLanes="2" speed="27.78" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.primary_link" priority="7" numLanes="1" speed="22.22" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.raceway" priority="15" numLanes="2" speed="83.33" allow="vip" oneway="0"/>
  <type id="highway.residential" priority="3" numLanes="1" speed="13.89" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.secondary" priority="11" numLanes="1" speed="27.78" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.secondary_link" priority="6" numLanes="1" speed="22.22" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.service" priority="1" numLanes="1" speed="5.56" allow="pedestrian delivery bicycle" oneway="0"/>
  <type id="highway.steps" priority="1" numLanes="1" speed="1.39" allow="pedestrian" oneway="1" width="2.00"/>
  <type id="highway.step" priority="1" numLanes="1" speed="1.39" allow="pedestrian" oneway="1" width="2.00"/>
  <type id="highway.steps" priority="1" numLanes="1" speed="1.39" allow="pedestrian" oneway="1" width="2.00"/>
  <type id="highway.tertiary" priority="10" numLanes="1" speed="22.22" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.tertiary_link" priority="5" numLanes="1" speed="22.22" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.track" priority="1" numLanes="1" speed="5.56" allow="pedestrian motorcycle moped bicycle" oneway="0"/>
  <type id="highway.trunk" priority="13" numLanes="2" speed="27.78" disallow="pedestrian bicycle tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.trunk_link" priority="8" numLanes="1" speed="22.22" disallow="pedestrian bicycle tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.unclassified" priority="4" numLanes="1" speed="13.89" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="highway.unsurfaced" priority="1" numLanes="1" speed="8.33" disallow="tram rail_urban rail_rail_electric rail_fast ship" oneway="0"/>
  <type id="railway.highspeed" priority="21" numLanes="1" speed="69.44" allow="rail_fast" oneway="1"/>
  <type id="railway.light_rail" priority="19" numLanes="1" speed="27.78" allow="rail_urban" oneway="1"/>
  <type id="railway.preserved" priority="16" numLanes="1" speed="27.78" allow="rail" oneway="1"/>
  <type id="railway.rail" priority="20" numLanes="1" speed="46.44" allow="rail" oneway="1"/>
  <type id="railway.subway" priority="18" numLanes="1" speed="27.78" allow="rail_urban" oneway="1"/>
  <type id="railway.tram" priority="17" numLanes="1" speed="13.89" allow="tram" oneway="1"/>

  <edge id=":10281813592_0" function="internal">
    <lane id=":10281813592_0_0" index="0" allow="pedestrian delivery bicycle" speed="5.56" length="4.36" shape="1720.06,2117.79 1718.77,2117.82 1717.88,2117.83 1717.00,2117.80 1715.71,2117.71"/>
  </edge>
  <edge id=":10281813592_1" function="internal">
    <lane id=":10281813592_1_0" index="0" allow="pedestrian delivery bicycle" speed="3.65" length="4.67" shape="1720.06,2117.79 1718.85,2117.01 1718.43,2116.22 1718.81,2115.42 1720.00,2114.59"/>
  </edge>
  <edge id=":10281813592_2" function="internal">
    <lane id=":10281813592_2_0" index="0" allow="pedestrian delivery bicycle" speed="5.56" length="4.03" shape="1715.97,2114.52 1717.16,2114.61 1717.98,2114.63 1718.80,2114.62 1720.00,2114.59"/>
  </edge>
  <edge id=":10281813592_3" function="internal">
    <lane id=":10281813592_3_0" index="0" allow="pedestrian delivery bicycle" speed="3.65" length="4.67" shape="1715.97,2114.52 1717.10,2115.41 1717.43,2116.24 1716.97,2117.01 1715.71,2117.71"/>
  </edge>
  <edge id=":10577420365_0" function="internal">
    <lane id=":10577420365_0_0" index="0" allow="pedestrian delivery bicycle" speed="5.56" length="5.23" shape="1743.55,2116.93 1742.00,2116.82 1740.94,2116.78 1739.88,2116.82 1738.33,2116.90"/>
  </edge>
  <edge id=":10577420365_1" function="internal">
    <lane id=":10577420365_1_0" index="0" allow="pedestrian delivery bicycle" speed="3.65" length="4.67" shape="1743.55,2116.93 1742.42,2116.04 1742.08,2115.21 1742.55,2114.44 1743.81,2113.74"/>
  </edge>
  <edge id=":10577420365_2" function="internal">
    <lane id=":10577420365_2_0" index="0" allow="pedestrian delivery bicycle" speed="5.56" length="5.67" shape="1738.15,2113.71 1739.83,2113.62 1740.98,2113.58 1742.13,2113.62 1743.81,2113.74"/>
  </edge>
  <edge id=":10577420365_3" function="internal">
```

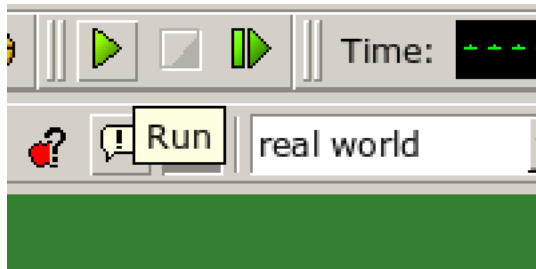
Из понятного – мы видим тут ребра, которые отображаются на карте.

Так же можно разобрать, что на нашей карте будут перекрёстки – они

описаны в тэгах junction

```
<junction id="3854485990" type="dead_end" x="1303.75" y="2176.85" inclanes="38225519783.0 38225519841.0" intlanes="" shape="1305.76,2176.23 1306.11,2176.26 1301.09,2175.36 1300.73,2177.33 1301.47,2177.62 1301.66,2177.86 1301.73,2178.15 1301.67,2178.50 1301.50,2178.91 1303.26,2179.86 1303.99,2178.84 1304.40,2178.51 1304.83,2178.29 1305.28,2178.20"/>
<junction id="3855315258" type="dead_end" x="1316.33" y="2179.12" inclanes="38225519782.0 382338543.0" intlanes="" shape="1318.63,2180.54 1318.98,2178.57 1318.16,2177.93 1318.08,2177.31 1318.23,2176.47 1318.60,2175.43 1319.20,2174.19 1317.16,2173.77 1316.73,2174.35 1316.55,2174.40 1316.39,2174.30 1316.25,2174.05 1316.13,2173.64 1314.18,2174.07 1314.49,2176.15 1314.44,2176.85 1314.27,2177.33 1313.98,2177.58 1313.55,2177.60 1313.19,2179.57"/>
<junction id="3859751954" type="dead_end" x="1386.74" y="2112.37" inclanes="44094097682.0 44218835481.0" intlanes="" shape="1388.97,2113.94 1389.44,2111.99 1384.58,2110.82 1384.11,2112.76 1384.89,2113.13 1385.13,2113.41 1385.27,2113.76 1385.31,2114.17 1385.25,2114.65 1387.21,2115.05 1387.54,2114.28 1387.80,2114.04 1388.13,2113.91 1388.52,2113.87"/>
<junction id="3859751956" type="dead_end" x="1378.49" y="2152.69" inclanes="38275624882.0 44218835480.0" intlanes="" shape="1380.92,2154.01 1381.17,2152.03 1380.38,2151.76 1380.13,2151.52 1379.98,2151.20 1379.93,2150.82 1379.98,2150.37 1378.03,2149.96 1377.02,2154.87 1378.98,2155.27 1379.33,2154.48 1379.61,2154.22 1379.97,2154.06 1380.41,2153.99"/>
<junction id="3859752157" type="right_before_left" x="1488.83" y="2160.49" inclanes="31906943180.0 31906943181.0" intlanes="3859752157.0 0 3859752157.1 0 3859752157.2 0 3859752157.3 0" shape="1487.42,2159.92 1488.15,2157.62 1489.80,2158.42 1489.52,2158.67 1488.16,2158.84 1479.72,2158.91 1479.21,2158.88 1478.90,2160.87 1479.75,2161.14 1480.80,2161.38 1480.15,2161.69 1480.19,2162.07 1480.14,2162.53 1486.41,2163.82"/>
```

8. Попробуем запустить теперь эту симуляцию. Для этого нажимаем кнопку Run в SUMO:



9. Мы сразу увидим, как по городу начнут передвигаться желтые автомобили:



10. Мы можем уменьшить скорость их передвижения с помощью изменения параметра Delay в большую сторону:



11. Мы так же можем выполнять симуляцию пошагово кнопкой Step:



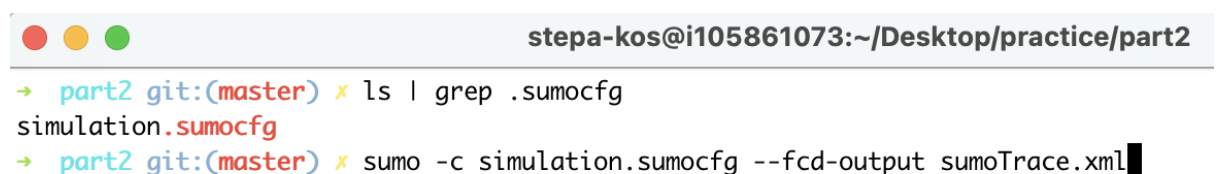
3. Экспорт трейса из SUMO

1. В этапе 2 мы рассматривали создание файла .sumocfg через WebWizard. Это файл конфигурации, который описывает сценарий нашей симуляции. Мы можем прогнать этот сценарий через sumo и на выходе получить файл, который будет описывать состояние системы в каждый промежуток времени. Чтобы узнать как это сделать, мы можем обратиться к официальной документации -

https://sumo.dlr.de/docs/Tutorials/Trace_File_Generation.html.

2. Пройдем по шагам, указанным в документации:

Запускаем sumo, передавая на вход файл нашего сценария. В качестве output указываем, что нам нужен fcd файл, то есть floating car data файл(см [документацию](#)), и название выходного файла, назовем его sumoTrace.xml:

A terminal window with a title bar showing the user 'stepa-kos' and the path '~/Desktop/practice/part2'. The terminal shows two commands being executed. The first command is 'ls | grep .sumocfg' which returns 'simulation.sumocfg'. The second command is 'sumo -c simulation.sumocfg --fcd-output sumoTrace.xml' which is shown with a cursor at the end.

```
stepa-kos@i105861073:~/Desktop/practice/part2
→ part2 git:(master) x ls | grep .sumocfg
simulation.sumocfg
→ part2 git:(master) x sumo -c simulation.sumocfg --fcd-output sumoTrace.xml
```

3. Видим логи запуска симуляции:

Тут есть кое-какая статистика по итогам симуляции. Из интересного, видим, что в программе, видимо, был использован алгоритм Дейкстры для поиска кратчайшего пути. В документации мы можем найти подтверждение этой догадке -

https://sumo.dlr.de/docs/Simulation/Routing.html#routing_algorithms

В документации так же говорится, что мы даже можем выбрать astar вместо алгоритма Дейкстры, если нам это нужно, просто указав опцию --routing-algorithm


```

→ part2 git:(master) x sumo -c simulation.sumocfg --fcd-output sumoTrace.xml
Warning: Could not set locale to 'C'.
Loading net-file from '../sumo/share/sumo/tools/2023-04-17-23-57-01/osm.net.xml.gz'
... done (178ms).
Loading additional-files from '../sumo/share/sumo/tools/2023-04-17-23-57-01/osm.poly.xml.gz' ... done (30ms).
Loading done.
Simulation version 1.16.0 started with time: 0.00.
Simulation ended at time: 3712.00
Reason: All vehicles have left the simulation.
Performance:
  Duration: 0.97s
  Real time factor: 3846.63
  UPS: 50600.000000
Vehicles:
  Inserted: 712
  Running: 0
  Waiting: 0
Statistics (avg of 712):
  RouteLength: 696.39
  Speed: 10.66
  Duration: 68.58
  WaitingTime: 10.56
  TimeLoss: 24.77
  DepartDelay: 0.71

DijkstraRouter answered 712 queries and explored 99.43 edges on average.
DijkstraRouter spent 0.02s answering queries (0.03ms on average).

```

4. После выполнения сценария мы получаем на выходе xml файл с трейсом:

```

→ part2 git:(master) x ls | grep sumoTrace
sumoTrace.xml

```

5. Взглянем на получившийся файл:

В нем мы видим набор тэгов timestep, которые содержат в себе состояние системы в определенный промежуток времени. Текущий промежуток времени определяется атрибутом time.

В нашем случае – внутри timestep присутствуют тэги, обозначающие состояние автомобиля. Каждый автомобиль имеет свой уникальный идентификатор, координаты, тип, скорость и другие параметры.

```

<fcd-export xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <timestep time="0.00">
    <vehicle
      id="veh0"
      x="1580.46"
      y="1698.85"
      angle="252.74"
      type="veh_passenger"
      speed="0.00"
      pos="5.10"
      lane="-27589844_0"
      slope="0.00"
    />
  </timestep>
  <timestep time="1.00">
    <vehicle
      id="veh0"
      x="1578.98"
      y="1698.39"
      angle="252.74"
      type="veh_passenger"
      speed="1.55"
      pos="6.65"
      lane="-27589844_0"
      slope="0.00"
    />
  </timestep>

```

Можно заметить, что с течением времени старые автомобили пропадают(выезжают за карту), а новые автомобили добавляются:

```

<timestep time="3574.00">
  <vehicle id="veh784" ;
  <vehicle id="veh785" ;
  <vehicle id="veh786" ;
  <vehicle id="veh788" ;
  <vehicle id="veh791" ;
  <vehicle id="veh793" ;
  <vehicle id="veh794" ;
  <vehicle id="veh795" ;
  <vehicle id="veh797" ;
  <vehicle id="veh798" ;
  <vehicle id="veh799" ;
</timestep>

```

4. Установка ns-3

1. Теперь нам нужно установить ns-3. Для начала скачаем с [официального сайта](#) архив с кодом:

Download

Please click the following link to download ns-3.38, released March 17, 2023:

- [ns-allinone-3.38](#) (compressed source code archive)

The [sha1sum](#) of the ns-allinone-3.38.tar.bz2 release archive is
b33980d82fe865965ac9ff95cfd9157bff5a91e4

Распаковываем архив и видим ряд файлов,

```
→ practice git:(master) x ls ns-3/ns-allinone-3.38
README.md      build.py      netanim-3.109 util.py
bake           constants.py  _ns-3.38
```

2. Теперь мы можем запустить файл build.py:

```
→ ns-allinone-3.38 git:(master) x ./build.py --enable-examples --enable-tests
# Build NetAnim
```

Стоит заметить, что предварительно требуется установить clang и gcc, cmake и python.

Началась компиляция файлов:

topology-read	traffic-control	uan
virtual-net-device	wave	wifi
wimax		

Modules that cannot be built:

brite	click	fd-net-device
mpi	openflow	tap-bridge
visualizer		

```
-- Configuring done (8.5s)
-- Generating done (3.8s)
-- Build files have been written to: /Users/stepa-kos/Desktop/practice/ns-3/ns-allinone-3.38/ns-3.38/cmake
Finished executing the following commands:
mkdir cmake-cache
cd cmake-cache; /usr/local/bin/cmake -DCMAKE_BUILD_TYPE=default -DNS3_ASSERT=ON -DNS3_LOG=ON -DNS3_WARNI
N -DNS3_TESTS=ON -G Unix Makefiles .. ; cd ..
=> /usr/local/opt/python@3.11/bin/python3.11 ns3 build
[ 0%] Building CXX object scratch/nested-subdir/CMakeFiles/scratch-nested-subdir-lib.dir/lib/scratch-ne
[ 1%] Building CXX object CMakeFiles/stdlib_pch_exec.dir/cmake_pch.hxx.pch
[ 1%] Building CXX object CMakeFiles/stdlib_pch-default.dir/cmake_pch.hxx.pch
[ 1%] Building CXX object src/test/CMakeFiles/libtest.dir/___/___/build-support/empty.cc.o
[ 1%] Building CXX object src/test/CMakeFiles/libtest.dir/ns3tcp/ns3tcp-loss-test-suite.cc.o
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/ns3tcp/ns3tcp-no-delay-test-suite.cc.o
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/csma-system-test-suite.cc.o
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/traced/traced-callback-typedef-test-suite.cc.o
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/ns3tcp/ns3tcp-socket-test-suite.cc.o
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/ns3tcp/ns3tcp-state-test-suite.cc.o
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/ns3tcp/ns3tcp-socket-writer.cc.o
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/traced/traced-value-callback-typedef-test-sui
[ 2%] Linking CXX static library /Users/stepa-kos/Desktop/practice/ns-3/ns-allinone-3.38/ns-3.38/build/
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/ns3tc/fq-cobalt-queue-disc-test-suite.cc.o
[ 2%] Building CXX object CMakeFiles/stdlib_pch_exec.dir/build-support/empty-main.cc.o
[ 2%] Building CXX object CMakeFiles/stdlib_pch-default.dir/build-support/empty.cc.o
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/ns3tc/fq-codel-queue-disc-test-suite.cc.o
[ 2%] Linking CXX executable ns3.38-stdlib_pch_exec-default
[ 2%] Building CXX object src/test/CMakeFiles/libtest.dir/ns3tc/fq-pie-queue-disc-test-suite.cc.o
[ 2%] Building CXX object src/antenna/CMakeFiles/libantenna-obj.dir/model/angles.cc.o
[ 2%] Building CXX object src/antenna/CMakeFiles/libantenna-obj.dir/model/antenna-model.cc.o
[ 2%] Building CXX object src/antenna/CMakeFiles/libantenna-obj.dir/model/cosine-antenna-model.cc.o
[ 2%] Building CXX object src/core/CMakeFiles/libcore-obj.dir/model/int64x64-128.cc.o
[ 2%] Building CXX object src/core/CMakeFiles/libcore-obj.dir/model/unix-fd-reader.cc.o
```

3. После установки заходим в директорию ns-3.38.

```
→ ns-allinone-3.38 git:(master) ✗ ls
README.md  __pycache__  bake  build.py  constants.py  netanim-3.109  ns-3.38  util.py
```

4. В ней лежат все скомпилированные файлы, они готовы к использованию:

```
→ ns-allinone-3.38 git:(master) ✗ ls ns-3.38
AUTHORS      CONTRIBUTING.md  RELEASE_NOTES.md  build  contrib  ns3  test.py
CHANGES.md  LICENSE          VERSION           build-support  doc      scratch  utils
CMakeLists.txt  README.md        bindings          cmake-cache  examples  src      utils.py
```

5. Мы можем провалидировать установку, запустив команду ./test.py:

```
→ ns-3.38 git:(master) ✗ ./test.py
Finished executing the following commands:
cd cmake-cache; /usr/local/bin/cmake --build . -j 11 ; cd ..
[1/758] PASS: TestSuite wimax-fragmentation
[2/758] PASS: TestSuite wimax-mac-messages
[3/758] PASS: TestSuite wimax-tlv
[4/758] PASS: TestSuite wifi-channel-access-manager
[5/758] PASS: TestSuite wifi-devices-dcf
```

6. Все скомпилировалось успешно:

758 of 758 tests passed (758 passed, 0 skipped, 0 failed, 0 crashed, 0 valgrind errors)

7. Теперь мы можем запустить скомпилированный ns3. Так как мы указали `--enable-examples` при компиляции, то у нас есть ряд примеров, которые мы можем запустить. Например, запустим hello-simulator:

```
→ ns-3.38 git:(master) x ./ns3 run hello-simulator
Hello Simulator
→ ns-3.38 git:(master) x █
```

8. Видим, что все корректно работает.

5. Преобразование трейса в файл .tcl

1. Теперь, когда у нас есть трейс, мы можем преобразовать его в тот формат, который мы можем использовать в ns3. Например .tcl. Чтобы узнать как это сделать – смотрим официальную документацию sumo -

<https://sumo.dlr.de/docs/Tools/TraceExporter.html#ns2ns3>

2. Пройдем по шагам из документации. Нужно запустить скрипт `traceExporter.py` из папки `tools`:

```
→ part2 git:(master) x ls ../sumo/share/sumo/tools | grep traceExporter
traceExporter.py
→ part2 git:(master) x python ../sumo/share/sumo/tools/traceExporter.py -
-fcd-input sumoTrace.xml --ns2mobility-output mobility.tcl
```

3. После выполнения скрипта – видим наш .tcl файл:

```
→ part2 git:(master) x ls | grep mobility
mobility.tcl
_
```

4. Открыв этот файл, мы видим список нод:

Тут после `at` указан `timestep`. Для каждого `timestep` указана строка, применяющая позицию(команда `setdest`) к каждому узлу.

Как мы и ожидали, у нас столько же `timestep` как и в трейсе, а так же мы видим, что последний узел – 711, а значит общее количество узлов 712, как было в трейсе. То есть файл сконвертировался корректно:

```
part2 > mobility.tcl
50407 $ns_ at 3600.0 "$node_(707) setdest 1721.72 1536.27 1000.00 17.72"
50408 $ns_ at 3600.0 "$node_(708) setdest 1536.27 1000.00 17.72"
50409 $ns_ at 3600.0 "$node_(709) setdest 1215.31 2052.24 12.97"
50410 $ns_ at 3600.0 "$node_(710) setdest 1669.04 1803.61 10.56"
50411 $node_(711) set X_ 1334.53
50412 $node_(711) set Y_ 1737.74
50413 $node_(711) set Z_ 0
```

5. Теперь мы можем использовать этот файл в программе для ns3

Вывод

Мы изучили графический интерфейс среды для симуляции SUMO. А также попробовали запустить простую симуляцию. Установили ns3.