gnss_visualisation Documentation

Release 0.0

am, yd

September 14, 2016

CHAPTER 1

Contents:

The aim of the project is to compare simulated and broadcast data. For that you will be able to create scenario that you want to launch on a Multi-GNSS Simulator Spectracom, to run this scenario and then, thanks to a simple graphical interface visualize a range of data coming from the simulator and the receiver(s).

The main documentation for the site is organized into a couple sections:

1.1 Define the scenario

There is already some scenario define in `data/scenariotest` but you can also define your own scenario

1.1.1 Scenarios already define

6 scenarios have been defined for you to test your devices:

- test_1: This is the *static case*, for a minute, data are taken from your simulator and/or your receiver(s).

- test_2: This is the *temporal square test*, the simulator run a path representing a square at constant speed , each section last a minute. It basically test if the receiver(s) is well responding to a brutal change of direction.

- test_3: This is the *temporal circle test*, the simulator run a path representing a circle at constant speed, during 2 minutes.

- test_4: This is the test of *constant acceleration* in straight line.

- test_5: This is the *sensitivity test*, staying static, the C/N0 is increased. The goal is to see until which C/N0 it is still possible to receive something

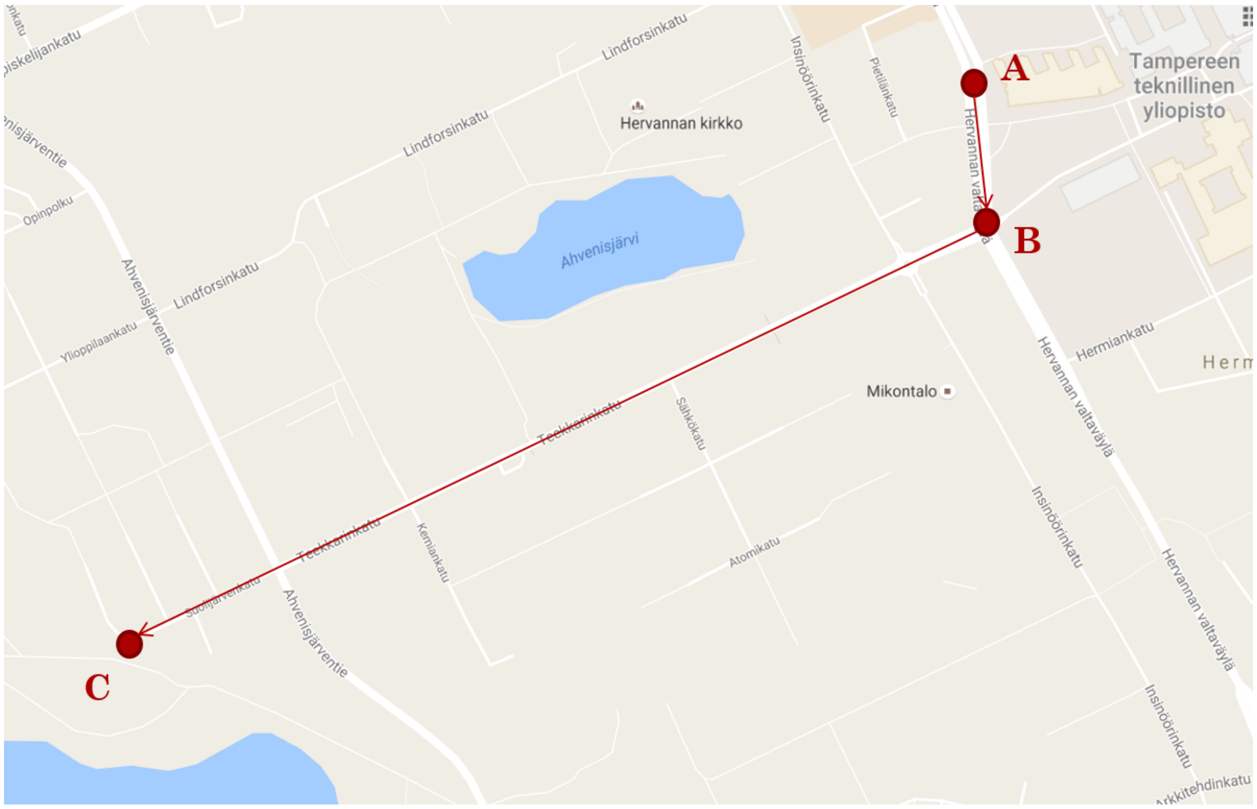
- test_6: This is the *Free to Urban space test*. The goal is to test if the receiver can keep its reliability when passing from a free space to a urban space.

Note: You can remark that all the files stored in the database are saved with an “s” as the first letter of their name. If you want to add your own files, make sure to keep the same notation, and to save it in .txt format.

1.1.2 Create your scenario

You can also create your own scenario, let’s explain that with an example.

Let’s imagine you want to go from the point A to C passing by B.



- 1st step:

Create a new .ini file and use this template:

```
[START]
LAT:
LONG:
ALT:
Duration:
Heading:
Speed:
Acceleration:
Rateheading:
Turnrate:
Turnradius:
Cn0:
Propagation:
Antenna:
Tropo:
Iono:
keepalt:
ECEFpos:
Multipath:
SpeedOverGround:
Verticalspeed:
Enuvel:
Ecefvel:
VerticalAcceleration:
ENUAccel:
ECEFAccel:
PRYattitude:
```

```
DPRYattitude:
Kepler:
[SECTION 1]
LAT:
LONG:
ALT:
Duration:
Heading:
Speed:
Acceleration:
Rateheading:
Turnrate:
Turnradius:
Cn0:
Propagation:
Antenna:
Tropo:
Iono:
keepalt:
signaltype:
ECEFpos:
Multipath:
SpeedOverGround:
Verticalspeed:
Enuvel:
Ecefvel:
VerticalAcceleration:
ENUAccel:
ECEFAccel:
PRYattitude:
DPRYattitude:
Kepler:
[END]
```

Note: In the [START] section, just fill the Latitude, Longitude and Altitude information of your departure position.

In our case the [START] section will look like:

```
[START]
LAT: latitude of A in decimal degrees
LONG: longitude of A in decimal degrees
ALT: altitude of A in meters
Duration:
Heading:
Speed:
Acceleration:
Rateheading:
Turnrate:
Turnradius:
Cn0:
Propagation:
Antenna:
Tropo:
Iono:
keepalt:
ECEFpos:
Multipath:
```

```
SpeedOverGround:
Verticalspeed:
Enuvel:
Ecefvel:
VerticalAcceleration:
ENUAccel:
ECEFAccel:
PRYattitude:
DPRYattitude:
Kepler:
```

- 2nd step:

Copy/Paste the number of [SECTION] needed. In our example, there is two sections:

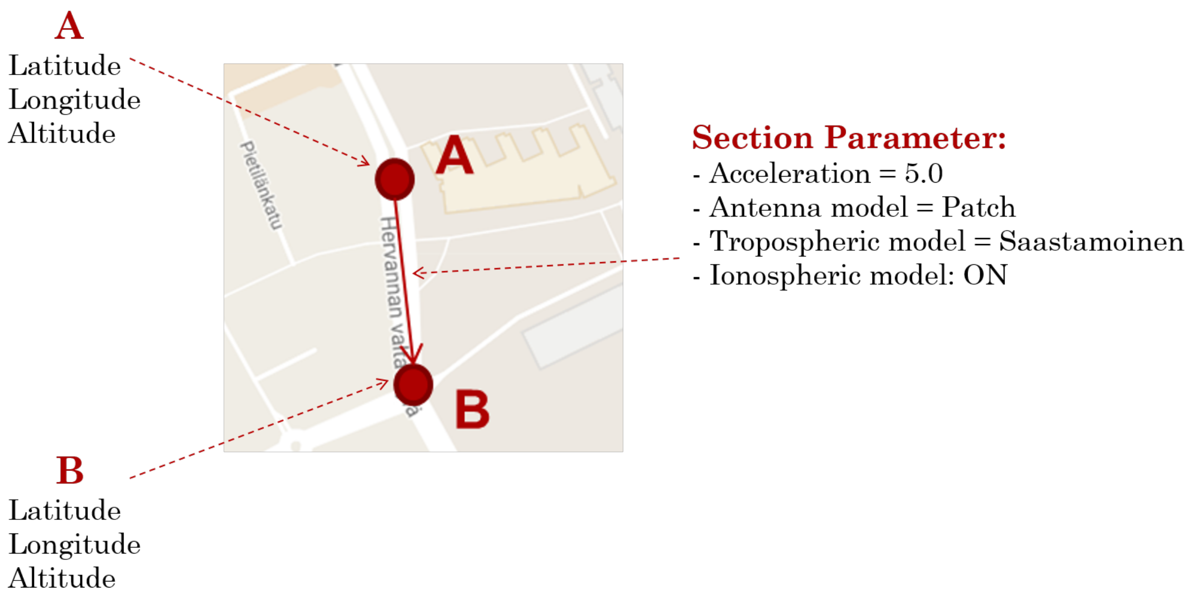
- Section 1 between A and B
- Section 2 between B and C

For each section, you can now fill all parameters you need but pay attention:

- Either fill a position information or a duration and a heading, not both
- fill LAT, LONG, ALT or ECEFpos not both
- fill ENUvel or ECEFvel or Speed or SpeedOverGround not four of them
- fill ENUaccel or ECEFaccel or Acceleration not three of them
- fill PRYattitude or DPRYattitude not both

For our example:

We know latitude, longitude and altitude of point A and B, and we want to put this 4 parameters (acceleration, antenna model, ionosperic model and tropospheric model).

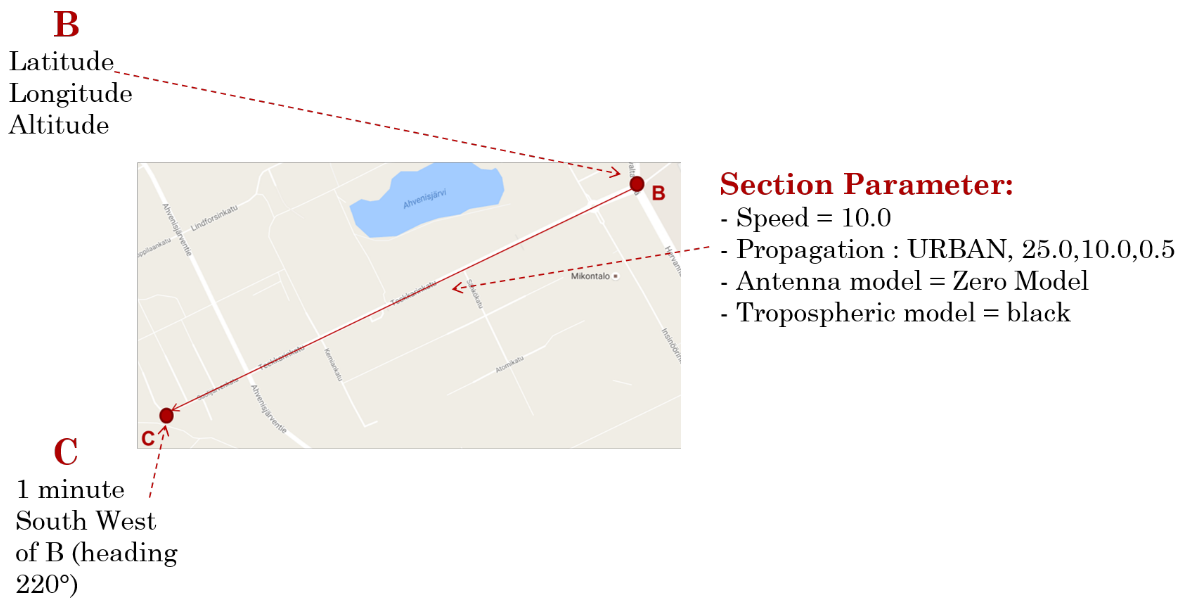


Then our [SECTION1] will look like:

```
[SECTION1]
LAT: latitude of B in decimal degrees
LONG: longitude of B in decimal degrees
```

```
ALT: altitude of B in meters
Duration:
Heading:
Speed:
Acceleration: 5.0
Rateheading:
Turnrate:
Turnradius:
Cn0:
Propagation:
Antenna: Patch
Tropo: Saastamoinen
Iono: ON
keepalt:
ECEFpos:
Multipath:
SpeedOverGround:
Verticalspeed:
Enuvel:
Ecefvel:
VerticalAcceleration:
ENUAccel:
ECEFAccel:
PRYattitude:
DPRYattitude:
Kepler:
```

Now, to reach C from B, we have to head South West for a minute. During this second section, we want to set 4 other parameters (speed, propagation model, antenna model and tropospheric model).



Then our [SECTION2] will look like:

```
[SECTION2]
LAT:
LONG:
ALT:
Duration: 00:00:01:00
```

```
Heading: 220
Speed: 10.0
Acceleration:
Rateheading:
Turnrate:
Turnradius:
Cn0:
Propagation: URBAN,25.0,10.0,0.5
Antenna: Zero model
Tropo: black
Iono:
keepalt:
ECEFpos:
Multipath:
SpeedOverGround:
Verticalspeed:
Enuvel:
Ecefvel:
VerticalAcceleration:
ENUAccel:
ECEFAccel:
PRYattitude:
DPRYattitude:
Kepler:
```

- 3rd step

Now you just have to put a [END] at the end of the file!

1.2 Run the scenario

1.2.1 Choose your scenario

Once you have create your own scenario, or if you want to run a pre-defined scenario, in the main file fill the argument of `tools.read_scen()` with the name of your scenario like this for example:

```
scenario = tools.read_scen('data/scenariotest/test_2.ini')
```

1.2.2 Ublox initialisation

There is different steps to configure the receiver(s):

- First of all, fill the COM port like this:

```
    ubloxcnx = Ublox(com='COM6')
```

- Then you have to choose which reset you want to make between **Cold RST**, **Warm RST** or **Hot RST**, like this:

```
    ubloxcnx.reset(command='Cold RST')
```

- Then you have to enable and disable which message you want or don't want to receive from the receiver, you can:

    enable:

    – ephemerides message thanks to **'EPH'**

  - ionospheric messages thanks to **'HUI'**
  - pseudo-range messages thanks to **'RAW'**
  - if you want all the UBX data just put **'UBX'**
  - position messages thanks to **'GGA'**
  - if you want all the NMEA data just put **'NMEA'**

disable:

  - all NMEA message thanks to **'NMEA'**
  - all UBX message thanks to **'UBX'**

Here is an example of how to enable/disable message:

```
ubloxcnx.enable(command='NMEA')
ubloxcnx.disable(command='UBX')
```

Note: If you don't want to get any UBX message, please comment the following lines:: thread_3 = AcquireData(3) thread_3.start()

1.2.3 Spectracom initialisation

To make the connexion with the Spectracom, you need to fill the argument of `Spectracom()`, like this for example:

```
spectracomcnx = Spectracom('USB0::0x14EB::0x0060::200448::INSTR')
```

You can if you want get the almanach data and/or the latest observation file by uncomment line 84 and/or 87 in the main file:

```
# Get almanach of Spectracom
spectracomcnx.get_almanach()

# Get latest observation file
spectracomcnx.get_latest()
```

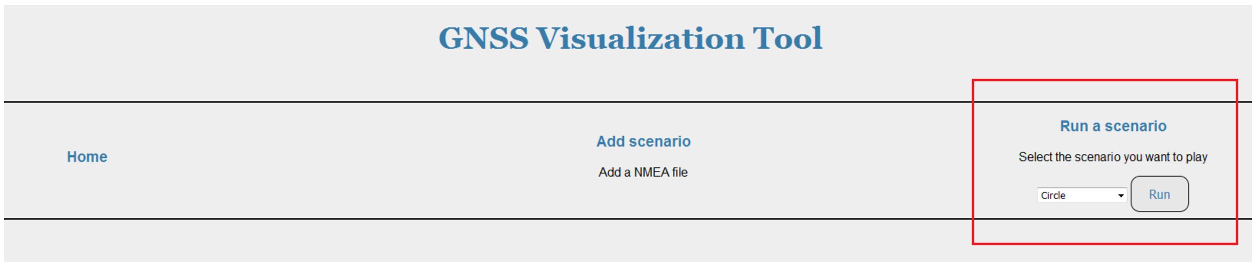
1.3 GNSS Visualisation Tools

1.3.1 Launch the application

First of all, you have to run the flask.py file. You can find it in GNSS_visualization_tools > GNSSTools > visualisation > flaskr. This will launch the server on http://127.0.0.1:5000. Then, if you want to get the home page, just go to http://127.0.0.1:5000/home.

1.3.2 Play a scenario

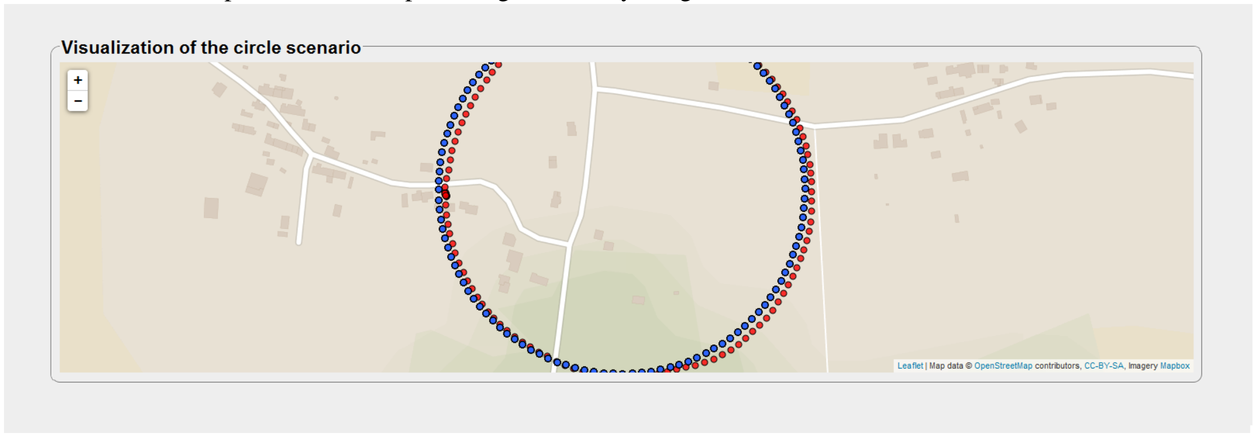
Six scenarios are already stored in the database : Circle, Square, Static, Acceleration, Sensivity and Free to urban tests. You can choose to run any of these in order to visualize the trajectory. You just have to select it in the list, then click on the "Run" button.



A map appears on the screen, centered on the coordinates where the scenario is played.

There are two differents trajectories for each scenario :

- The blue one represents the estimated position calculated by the receiver
- The red one represents the true position generated by the generator

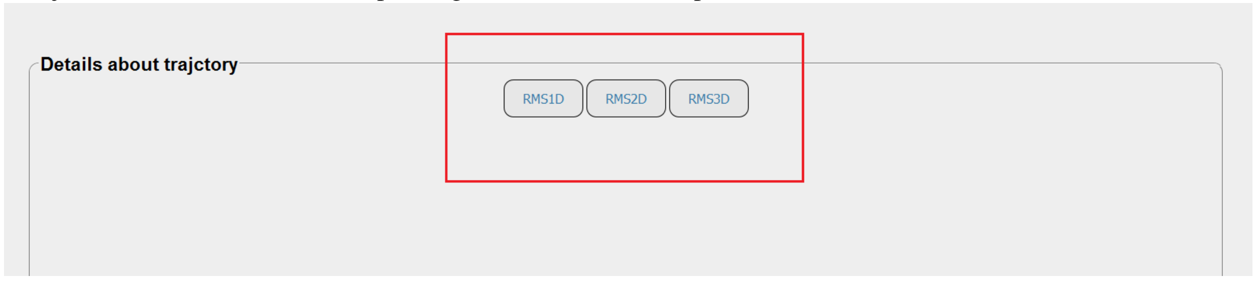

Visualization of the circle scenario
+
−
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery Mapbox

1.3.3 Avalaible data of the trajectory

You can have a sight of the Root Mean Square Error of the estimated trajectory. This information is avalaible :

- in 1D : RMS errors in altitude, longitude and altitude
- in 2D : RMS error in the plan latitude-longitude
- in 3D : RMS error in the space latitude-longitude-altitude

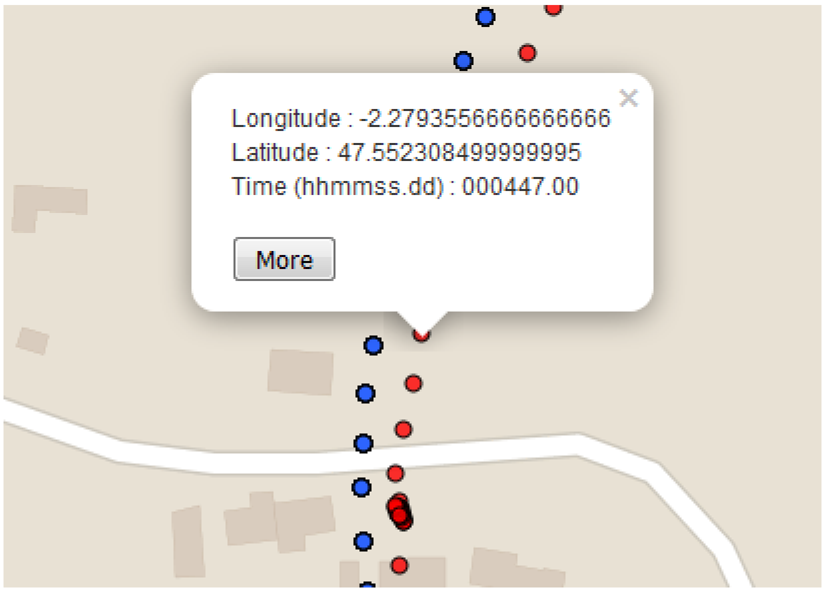
You just have to click on the corresponding button, below the map.



1.3.4 Avalaible data for each point

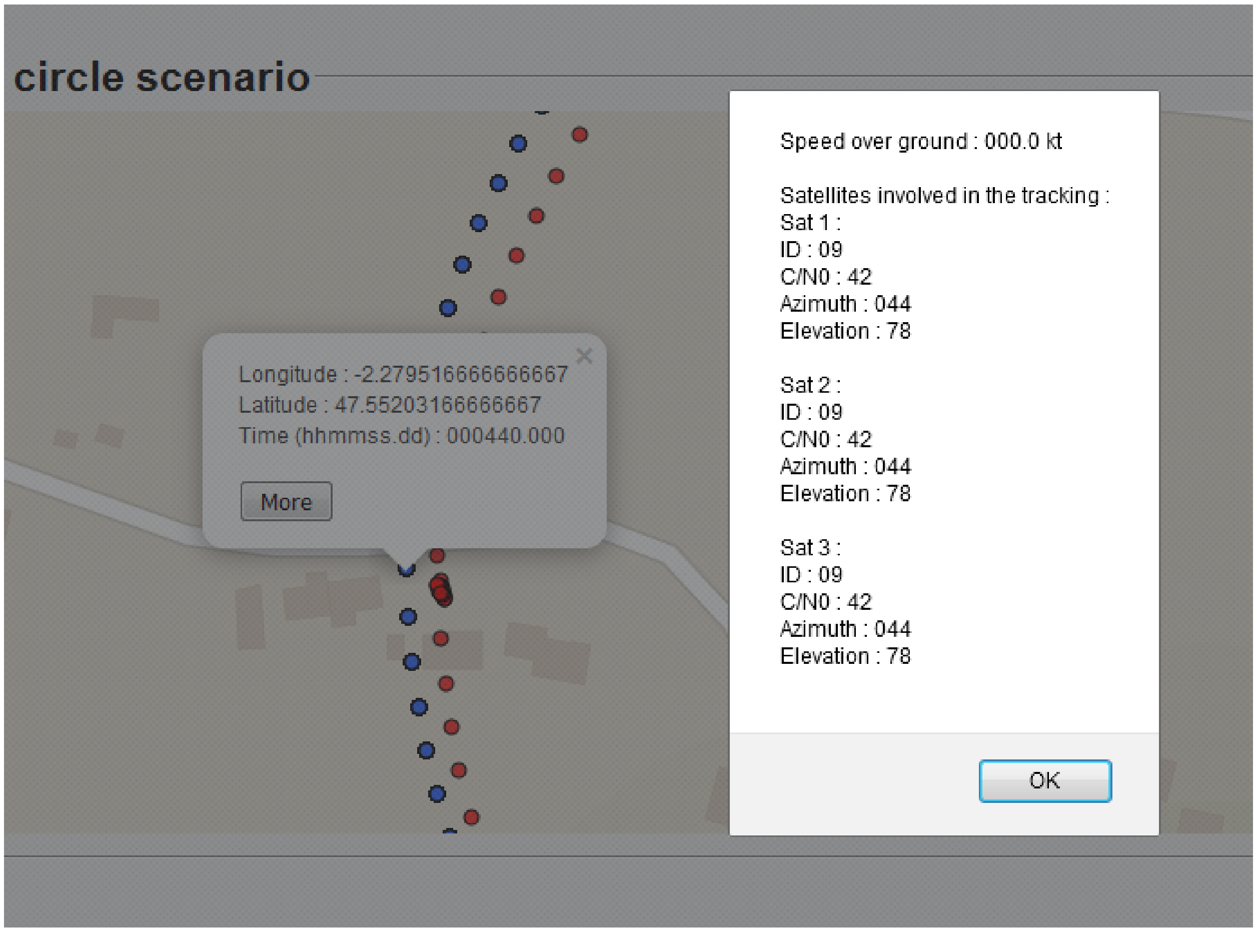
You can have a sight of several data of any point of the trajectory. You just have to click on the point you want. Then a first popup appears, giving two pieces of information :

- The coordinates of the point
- The time when the receiver was at that point
- The latitude and longitude errors of positionning


Longitude : -2.2793556666666666
Latitude : 47.552308499999995
Time (hhmmss.dd) : 000447.00

If you want to learn more, you need to click on th "More" button at the bottom of the popup. A alert window will appear and you will be given many data :

- The velocity of the receiver at that point
- The ID, elevation, azimuth and C/N0 of any satelite involved in the positionning at that point



Longitude : -2.279516666666667
Latitude : 47.55203166666667
Time (hhmmss.dd) : 000440.000

Speed over ground : 000.0 kt

Satellites involved in the tracking :
Sat 1 :
ID : 09
C/N0 : 42
Azimuth : 044
Elevation : 78

Sat 2 :
ID : 09
C/N0 : 42
Azimuth : 044
Elevation : 78

Sat 3 :
ID : 09
C/N0 : 42
Azimuth : 044
Elevation : 78

1.3.5 How to run your own files

Note: The “Add a nmea file” page hasn’t been written yet. The associated button isn’t linked to any page.

For the moment, there is a esay way to avoid this issue.

You may have remarked there is a “Test” choice in the list of the scenarios. There are likewise two files in the database : stest_ublox.txt and stest_spectracom.txt. These two files are empty, and dedicated to your own files. Let’s remind that all the stored files have the .txt format and their names begin with a “s”.

Thus you have two solutions :

- You can copy/paste the content of your nmea files in these two ones
- You can save your files with the same name “stest_ublox(spectracom).txt” in data > database and erase the empty ones

Don’t forget to refresh the web application. Now you can select the “Test” scenario and visualize it on the map.

CHAPTER 2

Features

- Create scenario
- Control Spectracom
- Control Ublox
- Visualize data

CHAPTER 3

Installation

CHAPTER 4

Contribute

- Issue Tracker: https://github.com/tutgnss/GNSS_visualization_tools/issues
- Source code: https://github.com/tutgnss/GNSS_visualization_tools

CHAPTER 5

License

The project is licensed under the MIT license.