TLE to JSON & Satellite Tracking

Maria Hjorth

January 8, 2018

Introduction

This protocol gives an overview of the two programs "`Project_Maria_JSON.py`" and "`Project_Maria_orbit.py`", and contains the user manual for these. A large part of the project has also been discussing the layout and needs for the control room software, as well as coordinating different tasks among each other. A sketch of a suggestion of the possible layout of the control room can be seen on figure 1. Of these subjects, I worked on converting the Two-Line Element set (TLE) to JavaScript Object Notation (JSON), and Anders, Peter and I also worked on satellite tracking and displaying its orbit.

The converter from TLE to JSON was coded in python 2.7. It takes a given list of a number of TLEs (`all.txt`), and from this finds specific TLE(s) of a given satellite(s) (specified in `whitelist.txt`). These TLEs are then converted to JSON. The two files containing many TLEs and the name of specific satellites were supplied by Nestor, but can be changed to any file with similar layout. The final product is a JSON file containing easy-readable different orbital characteristics. This format is useful, if the characteristics should be displayed in a browser.

The programme which tracks the satellite and displays its orbit was also coded in python 2.7. Even though Anders, Peter and I worked on the same project, we did not work together, but rather in parallel, only using each other to exchange and discuss ideas. Therefore, our products are rather similar but still has some differences. The key difference for my programme, is that it is fed by a list of a number of TLEs (`all.txt`) and a name of the specific satellite we want to display (`whitelist_test.txt`), as was the case for the TLE to JSON converter. From this, the programme plots the past and future orbits of the chosen satellite on two different maps of the Earth.

In the following, the user manual will be given for the two programs. This is only aimed at people who want to use the programs, without necessarily understanding how they work. If the reader wants behind-the-scenes information on the two programs, they should read the `README.txt` file.

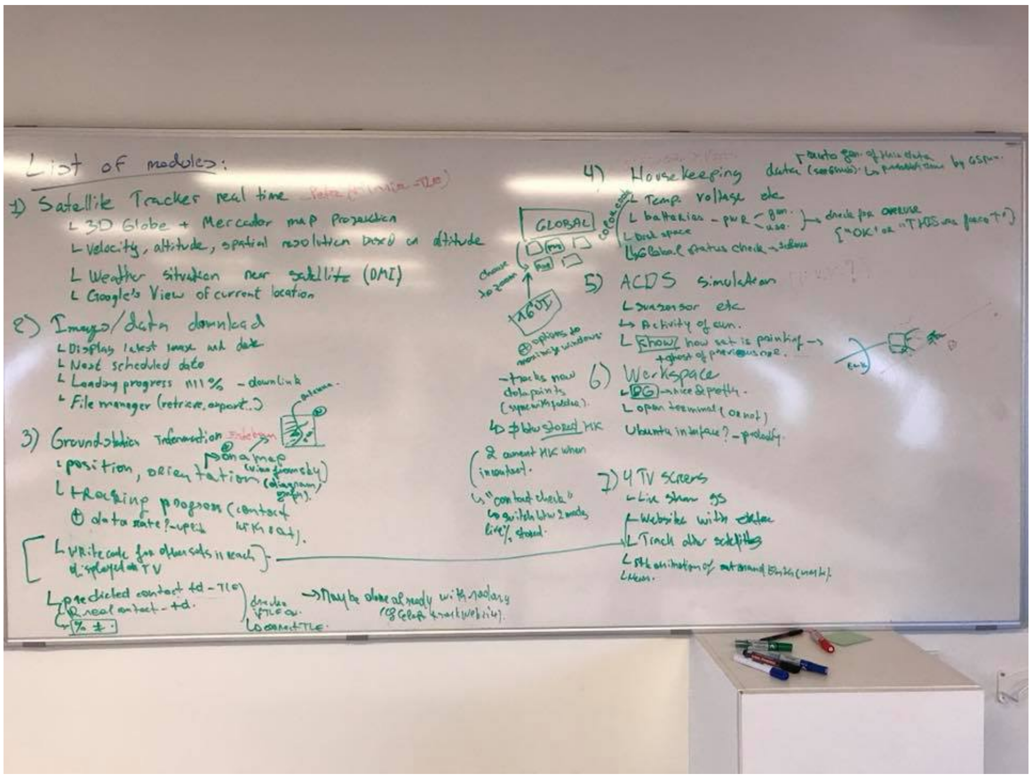
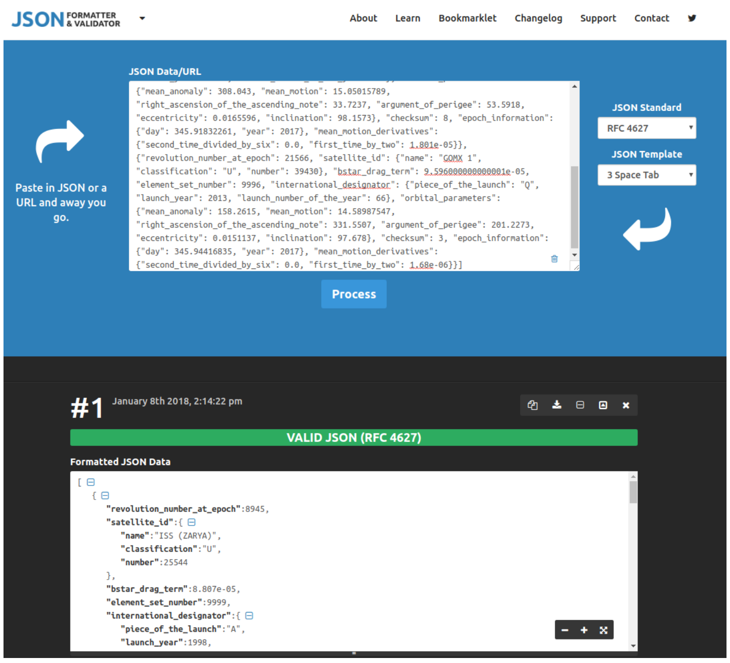


Figure 1: Shows the group-made mindmap of a possibly layout of the control room software. Photo credit: Katia Verteletska.

User Manual: TLE to JSON

The file of ”`Project_Maria_JSON.py`” converts TLE to JSON. It takes two inputs: the name of the file containing many TLEs (parameter: all_list), and the name of the satellite(s) which should be searched for (parameter: white_list). The corresponding files in the directory are supplied by Nestor, but can be changed or updated, as long as they keep their current format and layout. The output of the programme is the ”`data.json`” file, which contains the JSON-converted TLEs of the chosen satellite(s). To run the programme, simply open the python 2.7 shell in the programme directory and type `execfile('Project_Maria_JSON.py')`. To check whether the output is truly a JSON file, the ”`data.json`” is submitted to `jsonformatter.curiousconcept.com` as suggested by Nestor. In figure 2, it can be seen that the resulting file is indeed in the format of JSON. One future application of this could be to implement the orbital characteristics on a web page and then display them directly in a browser in the control room.



```
{"mean_anomaly": 308.043, "mean_motion": 15.05015789,
"right_ascension_of_the_ascending_note": 33.7237, "argument_of_perigee": 53.5918,
"eccentricity": 0.0165596, "inclination": 98.1573}, "checksum": 8, "epoch_information":
{"day": 345.91832261, "year": 2017}, "mean_motion_derivatives":
{"second_time_divided_by_six": 0.0, "first_time_by_two": 1.801e-05}},
{"revolution_number_at_epoch": 21566, "satellite_id": {"name": "GOMX 1",
"classification": "U", "number": 39430}, "bstar_drag_term": 9.596000000000001e-05,
"element_set_number": 9996, "international_designator": {"piece_of_the_launch": "Q",
"launch_year": 2013, "launch_number_of_the_year": 66}, "orbital_parameters":
{"mean_anomaly": 158.2615, "mean_motion": 14.58987547,
"right_ascension_of_the_ascending_note": 331.5507, "argument_of_perigee": 201.2273,
"eccentricity": 0.0151137, "inclination": 97.678}, "checksum": 3, "epoch_information":
{"day": 345.94416835, "year": 2017}, "mean_motion_derivatives":
{"second_time_divided_by_six": 0.0, "first_time_by_two": 1.68e-06}}]
```

```
[
   {
      "revolution_number_at_epoch":8945,
      "satellite_id":{
         "name":"ISS (ZARYA)",
         "classification":"U",
         "number":25544
      },
      "bstar_drag_term":8.807e-05,
      "element_set_number":9999,
      "international_designator":{
         "piece_of_the_launch":"A",
         "launch_year":1998,
```

Figure 2: Shows the JSON format test of the output file ”`data.json`” using `jsonformatter.curiousconcept.com`. It is seen, that the file is indeed in JSON format and layout.

User Manual: Satellite Tracking and Orbit Display

The file of ”`Project_Maria_orbit.py`” searches for a given satellite in a list of a number of TLEs, and plots past and future orbits on two different kind of maps. It uses two required inputs: the name of the files containing the specific satellite we are interested in (parameter: white_list) and the name of the file containing many TLEs (parameter: all_list). Furthermore, three optional inputs can be supplied: the time step of each data point in minutes (parameter: td; default: 1), the number of future orbits that should be displayed (parameter: N; default: 5), and at what time we wish the satellite to be displayed (parameter: tle_time; default: current time). The output of the programme are two figures showing the past and future orbits of the satellite, and an example can be seen in figure 3. To run the programme, open the python 2.7 shell in the programme directory and type `execfile('Project_Maria_orbit.py')`. Since this code is not meant to run through the browser, the programme does not use the TLE to JSON converter, but instead uses the python package `ephem.py` to convert the TLE information to latitude, longitude and elevation of the satellite. A further expansion for future students could be to include weather details, other satellites or live-feed/updates.

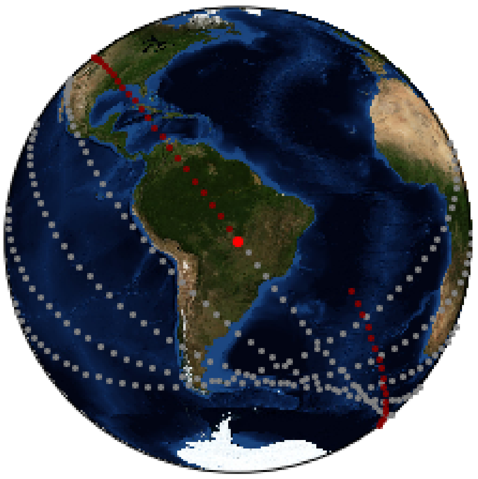
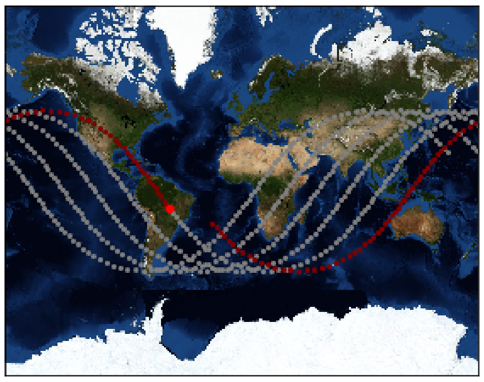



Figure 3: Shows the position of the ISS at a given time (bright red), as well as its past orbit (dark red) and five future orbits (grey). The TLEs used are the ones supplied by Nestor, but is easily changeable. The two maps are the projected round Earth (left) and a 2D Mercator projection (right).

Conclusion

This protocol describes the use of two different control room software programs: A programme which converts the TLE of a given satellite to JSON format, and a programme which plots two different orbital maps for a given satellite. Both use two text files containing many TLEs and the name(s) of the given satellite respectively, while the later program also has additional optional parameter inputs, namely the time step of each data point, the number of future orbits displayed and the time of the current satellite.

Both programs is a part of a larger (suggested) layout for the control room software. Both can be expanded by future students: The resulting JSON file can be used to display orbital characteristic in a browser in the control room, and the satellite tracking programme can be coded to also include e.g. the current weather and constant updates.