



## A visual interface for satellite simulation using Google Earth

Pereira, M. O.<sup>1 2</sup>, Dos Santos, W. A.<sup>1</sup>

<sup>1</sup> National Institute for Space Research, São José dos Campos, SP, Brasil

<sup>2</sup> Doctoral student on Space Engineering and Technologies - ETE.

mateusop@yahoo.com.br

---

**Abstract.** This work presents the interface between a C implemented satellite simulator into Google Earth software. This approach aims to augment reality on satellite missions simulations such as image processing, radar, and mapping. The focus of this paper is to describe the connection link between the simulator and GE interface as well as the camera control and positioning of satellite in space.

---

**keywords:** Satellite Simulator; Google Earth; Mission Analysis.

### 1. Introduction

Simulations are essential for products or even ideas to be tested before any prototypes are built. As computing power increases, simulators tend to become more and more complex and capable of fulfilling remaining gaps between virtual environment and real world. A well known software to perform satellite simulations, called STK (Systems Tool Kit, formerly Satellite Tool Kit) [Analytical Graphics 2018], performs a large variety of missions. Another simulator available is called VSSGS by Taitus Software [Taitus 2018]. A simple screen-shot can be seen in Figure 1.

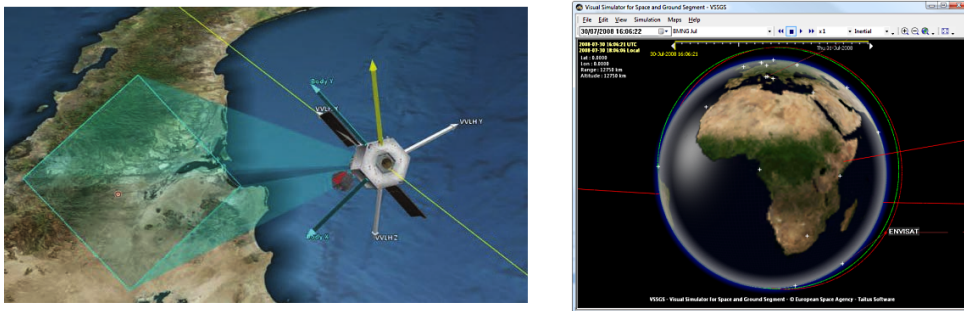


Figure 1. Examples of STK (left) and Taitus VSSGS (right). [Adapted from: [Analytical Graphics 2018] and [Taitus 2018] ]

Even though both softwares are the current the state of the art in mission simulations, they are costly and sometimes intangible for small research groups. Motivated by the history of open-source communities, this work then proposes an open source implementation to synchronize a custom simulator with Google Earth software, providing real imagery which can be used to augment reality at any satellite mission design.



## 2. Methodology

For this work, the following methodology was adopted: collect attitude data from simulator, rewrite values into Google Earth's format and dispatch via TCP/IP connection, as shown in Figure 2.

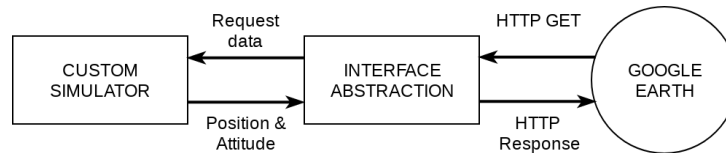


Figure 2. High level design for communication

To attempt software communication, two options were found. First, to simulate positioning via GPS NMEA messages. The NMEA format is available here [SiRF 2012] and an example of communication can be seen here [Google Earth Help 2018]. Although this approach seems more robust and convenient to real life cases, the GPS messages does not control camera, which is actually used for attitude and orientation purposes. Another way to communicate with GE is creating a link between files, as performed by [Zachwieja, J. 2012]. Using this method, all available GE parameters can be sent using the Google's KML format. See KML documentation for details [Google 2016].

The second method was chosen for this work because it provides full access to the KML tags. The idea presented by [Zachwieja, J. 2012] is to position the GE camera based on GPS received messages. As GPS messages are by default updated at a frequency of 1Hz, the file to file implementation works out with any problem. If higher update rates are needed, the file-write becomes a bottle neck in the communication link. This work main contribution is to provide a direct software link without the use of an external updated file, avoiding massive disk I/O usage and possible operating system instability.

The workaround for faster communication uses a single and static KML file to instruct GE the address of a TCP/IP connection, file name and update rate. Using this method, GE can attempt a direct connection with any TCP/IP software. Hence, instead of updating a file, the contents are directly sent via socket and thereby no disk is used. An example of the link can be seen in Figure 3.

```
<NetworkLink>
  <name>Satellite TCP (epoch)</name>
  <open>1</open>
  <TimeStamp><when>2018-05-09T03:00:00Z</when></TimeStamp>
  <flyToView>1</flyToView>
  <Link>
    <href>http://127.0.0.1:1628</href>
    <refreshMode>onInterval</refreshMode>
    <refreshInterval>0.0666</refreshInterval>
  </Link>
</NetworkLink>
```

Figure 3. Example of kml link structure.

Using a KML file with this link structure, it is possible to instruct GE in how to read the actual position KML file from network. Each time GE requests this position file, the simulator collects



position and attitude data of the simulated satellite orbit and creates a new response message containing the up to date simulated status. This message is sent back to GE and all the valid KML tags are then processed and updated. Note that GE has a smooth fly-to mechanism and it must be bypassed by the KML tag `<flyToView>` in the link structure. This response message, in full format, is shown in Figure 4, which is an example of a particular status response.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
  <Placemark>
    <name>Simulated Satellite</name>
    <Camera>
      <longitude>16.601803</longitude>
      <latitude>-35.297547</latitude>
      <altitude>753777.451216</altitude>
      <heading>90.803777</heading>
      <tilt>57.856152</tilt>
      <roll>-46.808791</roll>
      <altitudeMode>absolute</altitudeMode>
    </Camera>
  </Placemark>
</kml>
```

Figure 4. Example of kml response structure.

After the communication link between simulator and Google Earth was established, the next step was to define the variables which need to be sent to GE in order to position the camera accordingly to satellite position and attitude. Table 1 lists and maps simulation variables into GE tags on KML.

Table 1: List of Simulation variables.

Var	KML tag	Unit	Description
Latitude	<code>&lt; latitude &gt;</code>	Degeees	Latitude position of ground track
Longitude	<code>&lt; longitude &gt;</code>	Degees	Longitude position of ground track
Altitude	<code>&lt; altitude &gt;</code>	Meters	Altitude above mean sea level
Pitch ( $\theta$ )	<code>&lt; tilt &gt;</code>	Degees	Angle from Zenith to camera view
Roll ( $\phi$ )	<code>&lt; roll &gt;</code>	Degees	Camera rotation (Clockwise)
Yaw ( $\Psi$ )	<code>&lt; heading &gt;</code>	Degees	Camera direction from North (Clockwise)

From these variables, fully satellite positioning and attitude can be described. LLA positioning is well known and will not be detailed here. Attitude angles, though, depends on which reference they are measured. As seen in [Carrara 2012] and [Grooves 2008] the more usual way to define a satellite attitude is relative to the ECI frame.

For GE, the angles are relative to current camera position. In other words, the  $\theta$ ,  $\phi$  and  $\Psi$  angles are measured in NED frame. Furthermore, the zero `<tilt>` angle means a top-down view while a zero pitch angle means a leveled horizon. Then a conversion must be done. Finally, all this variable translation must be done on interface abstract prior to sending GE responses. Figure 5 illustrates how these angles are expected to be measured. Note that, as satellites rotate around their center of mass, the `<range>` tag is suppressed or always zero.

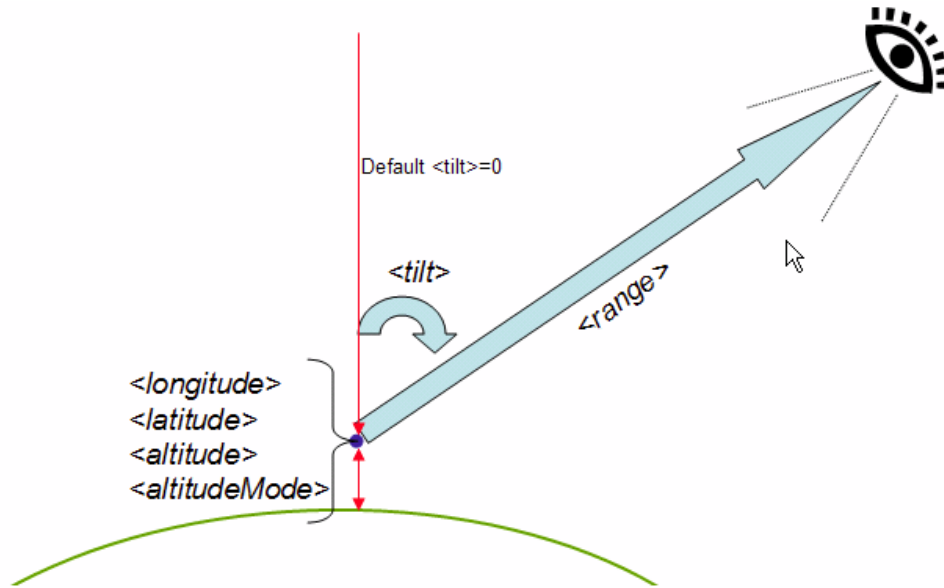


Figure 5. Google Earth camera angles. [Source: [Google 2016]]

After defined all data which must be sent to Google Earth, the timing behavior must be understood. Unfortunately, there is no way to send data to GE, in fact, its automatically updated by GE itself as defined by the KML tag `<updateInterval>` in Figure 3. At a fixed refresh rate, a TCP connection is opened to the simulator interface, see Figure 6 item (a). Then the interface requests the simulator engine for latest status of simulated variables (see item (b)). The necessary information is gathered during item (c) and translated into GE's KML units. Finally, the item (d) represents the final response message being sent back to GE.

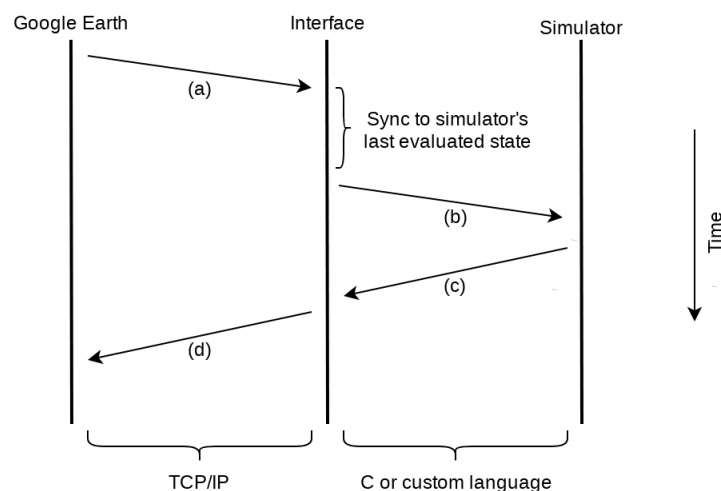


Figure 6. Interface timings

Initially, only forward view was implemented, which was in a constant direction along satellite  $x+$  axis. For extended user experience, extra modes of orientation were defined. Table 2 lists all implemented and tested modes of camera orientation.



Table 2: List of camera control modes

Mode	Description
Forward	Looks along satellite body $x+$ axis.
Down	Looks down (Nadir) despite satellite attitude.
Path	Looks along satellite orbit path, with $\theta = -30$ deg.
Back	Same as <i>Path</i> but backwards.
At	Look at specific $\Psi$ and $\theta$ respect to body.
Sun	Points toward the sun in sky

All functions presented here were implemented directly into a simulator using C language. Although the simulator was written in C, as communications to Google Earth are performed using TCP/IP, any language which supports socket connections can be used to re-implement this feature. In fact, simulator and GE can be ran in different machines. The results of this work are described in next section.

### 3. Results and Discussion

The interface between a custom simulator and Google Earth performed well and satisfactory. The orbit could be visualized in real time during the simulation. Fly-by time and revisiting times can be evaluated accordingly to any mission needs. Figure 7 shows an example of the satellite in GE.

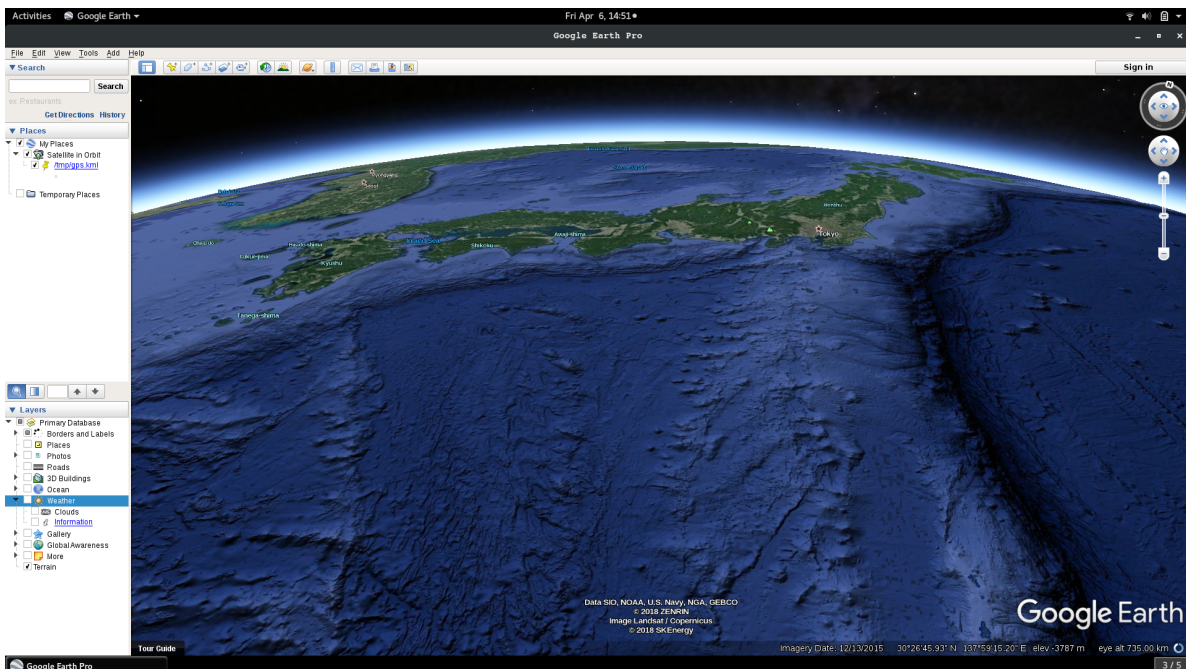


Figure 7. Satellite in orbit on Google Earth (Path view)

The performance of this communication is within interactive time. Update rates up to 30Hz were tested and used. Due to the possibility of accelerating the simulation time and achieve faster satellite speeds, a value of 15Hz was used to not overload imagery update.

Though, some limitations were found: (i) Google Earth cache is limited up to 2GB, and therefore world imagery is always fetched from servers; (ii) the time stamp is stored in the link (not



response) KML file, which is static, and could not be changed by simulator. If day/night or sun orientation are needed, the epoch should be kept fixed; (iii) Near the earth poles Euler angles have a singularity and therefore will occur if satellite passes close to them in polar orbits.

Despite this limitations being found, it is possible to overcome items i and ii. However it is likely not possible to eliminate singularities detected by item iii, as it is the way GE was implemented.

During this work, two additional tests were made and shown in Figure 8 with look mode into sun at left and looking down at right.

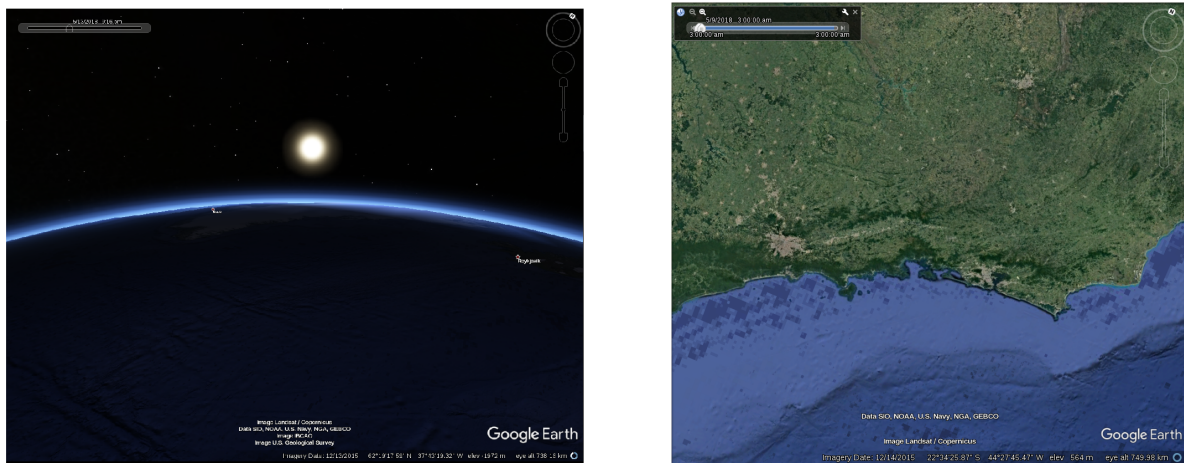


Figure 8. Satellite in orbit on Google Earth (Sun and down view)

Future implementations are widely identified for this work and the most important are listed below.

It is known that different cameras have different fields of view. As there is no option to actually *zoom* at the current position and fit different cameras FoV, the GE camera altitude can be changed to a lower or higher value, accordingly respective to narrow or wide fields of view.

Another proposal is to load 3D objects into Google Earth and use them as target or reference for mission analysis, increasing simulation reality.

Finally, some method for changing the epoch date initially adjusted in GE is necessary to simulate day and night transitions properly.

#### 4. Conclusion

A small contribution to open-source projects and small research groups was accomplished with this work. The implementation of the interface using the TCP/IP direct method was simple and reliable compared to direct file-write or simulating a GPS receiver into Google Earth. Real imagery can be used in any mission analysis to better identify suitable payload sensors and imaging devices.

**Acknowledgments:** *The author would like to thank INPE for the acceptance into doctoral program and CAPES for the scholarship support. Last but not least, all my professors whose lectures made this possible.*

#### References

Analytical Graphics, Inc (2018). Systems Tool Kit, available at: <http://www.agi.com/home>  
Accessed on August 6th 2018.



- Taitus Software Italia Srl (2018). Visual Simulator for Space and Ground Segment, available at: <http://www.taitussoftware.com/products/applications/vssgs-visual-simulator-space-ground-segment/> Accessed on August 6th 2018.
- Carrara, V (2012). Cinemática e Dinâmica de Satélites Artificiais São José dos Campos, INPE, available at: <http://urlib.net/8JMKD3MGP7W/3B96GD8>.
- Grooves, P. D. (2012). Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems. Boston: Artech house, 2008. 505 p.
- Google LLC (2016). KML Documentation Introduction, available at: <https://developers.google.com/kml/documentation/> Accessed on August 6th 2018.
- SiRF Technology, Inc (2007). NMEA Reference Manual Revision 2.1, December 2007 San Jose, CA, U.S.A.
- Google LLC (2018). Real-Time GPS Tracking, available at: <https://support.google.com/earth/answer/148095?hl=en> Accessed on August 6th 2018.
- Jaroslav Zachwieja (2012). Google earth gpsd (GEgpsd) version 0.3, June 2012, available at: [https://warwick.ac.uk/fac/sci/csc/people/computingstaff/jaroslav\\_zachwieja/gegpsd/](https://warwick.ac.uk/fac/sci/csc/people/computingstaff/jaroslav_zachwieja/gegpsd/) Accessed on August 6th 2018.