VTS User Manual

Version 3.6

VTS-MU-G-69-SPB-7.4

DOCUMENT HISTORY

Issue	Rev.	Date	Reference	Author(s)	Reasons for evolution
07	04	23/11/2021	VTS/MU/69	SPACEBEL SAS	Release VTS 3.6
07	03	14/12/2020	VTS/MU/69	SPACEBEL SAS	Release VTS 3.5 Timeshift
07	02	09/12/2019	VTS/MU/69	SPACEBEL SAS	Release VTS 3.4 Moon
07	01	07/12/2018	VTS/MU/69	SPACEBEL SAS	Release VTS 3.3 Ozone
07	00	07/12/2017	VTS/MU/69	SPACEBEL SAS	Release VTS 3.2 Zenith
06	02	07/02/2017	VTS/MU/69	SPACEBEL SAS	Release VTS 3.1 Kourou
06	01	27/09/2016	VTS/MU/69	SPACEBEL SAS	Minor changes
06	00	05/07/2016	VTS/MU/69	SPACEBEL SAS	Release VTS 3.0 OmniView
05	02	27/07/2015	VTS/MU/69	SPACEBEL SAS	Release VTS 2.7 DORIS
05	01	14/11/2014	VTS/MU/69	SPACEBEL SAS	Release VTS 2.6 Rosetta
05	00	06/03/2014	VTS/MU/69	SPACEBEL SAS	Release VTS 2.5 (English version)
04	04	03/12/2013	VTS/MU/69	SPACEBEL SAS	Release VTS 2.4
04	03	03/07/2013	VTS/MU/69	SPACEBEL SAS	Release VTS 2.3
04	02	19/02/2013	VTS/MU/69	SPACEBEL SAS	Release VTS 2.2
04	01	13/09/2012	VTS/MU/69	SPACEBEL SAS	Minor changes
04	00	24/07/2012	VTS/MU/69	SPACEBEL SAS	Release VTS 2.0
03	01	05/06/2012	VTS/MU/69	SPACEBEL SAS	Release VTS 1.3.1
03	00	27/02/2012	VTS/MU/69	SPACEBEL SAS	Release VTS 1.3.0
02	00	19/05/2011	VTS/MU/69	SPACEBEL SAS	Manual generation from wiki
01	01	13/10/2010	VTS/MU/69	SPACEBEL SAS	Minor changes
01	00	15/03/2010	CELEST/MU/69	SPACEBEL SAS	Document creation

CONTENTS:

CHAPTER

ONE

INTRODUCTION

1.1 Overview of VTS

VTS is a visualization toolkit for space data. Its primary goal is to animate satellites in both 2D and 3D environments. Its architecture also allows it to be extended with any number of compatible applications.

The toolkit comes with a configuration utility, which allows the definition of the elements in the visualization: 3D models, geometry and mobile parts of the satellites, data sources for all positions, attitudes, angle of rotations, etc. It also allows configuration for satellite sensors and ground stations, for the visualization scenario, and for the applications involved in the visualization. VTS then uses this configuration to run the animation. It handles starting and synchronizing all the chosen applications. Its core element, the *Broker*, offers functionality for navigating in visualization time, controlling 3D cameras and defining a range of other display attributes. It also broadcasts visualization settings associated with each scenario state to client applications.

VTS is designed in a way that allows connected applications to control the visualization time (eg. time ticks from a simulator, time navigation in a plotting software, etc.). As for the visualization data, they can be provided either through files or network streams (which are broadcasted to all clients). The synchronization protocol for client applications is specific to VTS; however the data files are based on a CCSDS standard, allowing direct compatibility with all tools handling this european format.

On the technical side, VTS relies on Celestia as its default 3D visualization tool. Celestia is free software, well known for its performance and the realism of its rendering. All of Celestia's many features are thus available to VTS users. As for portability, the toolkit is written using C++ and the Qt framework, ensuring stable behavior across Linux and Windows platforms. Lastly, available client applications come from a variety of origins. As an example, the PrestoPlot plotting tool is fully interfaced with VTS.

With these characteristics, VTS becomes an essential tool for all data production activities in the satellites flight dynamics field. It enables graphical validation of behaviors and attitude strategies, and can be used as a practical exchange support for all actors in satellite AOCS and space mechanics.

1.2 VTS Quick Start

1.2.1 Installing the toolkit

The VTS toolkit comes as a zip archive. To install it, simply decompress it in a folder. This installation folder will be referred to as `VTS/` throughout this manual.

1.2.2 Starting the toolkit

The VTS toolkit can be started by double-clicking the `startVTS.exe` executable file under Windows, or by running the command `./startVTS` under Linux.

The main window that opens is the configuration utility. It is used to create a project and define its elements: satellites, sensors, ground stations, and client applications. This quick start guide will only cover opening and launching an existing project. If the directory containing the executable file doesn't allow writing, a warning message appears in the notification area.

1.2.3 Opening an existing project

- Click the `Open project`  button in the toolbar, or in the `File` menu.
- The `Open Project File` dialog opens. Select the `Cubesat.vts` project file located under `VTS/Data/CubeSat/` and click `Open`.
- The project is loaded into the project tree.

1.2.4 Starting the visualization

- To start the project visualization, click the `Run`  button in the toolbar, or in the `Project` menu.
- The *Broker* window opens, with all the client applications defined for the project: the 2D window, Celestia, PrestoPlot, etc.

1.2.5 Interacting with the Broker

The Broker is the core application of the visualization phase. It starts the client applications, regulates the visualization time, sends user commands to the client applications, and much more.

Start of a visualization

When visualization is started from the *VTS configuration utility*, the Broker starts all the *client applications* defined in the VTS project. During this initialization phase, the Broker displays the *Initializing* message in the text fields of the time control area. Time does not start flowing until all client applications have signaled they are ready.

Upon connection of the various client applications, the Broker's tabs are populated with commands and information regarding these applications.

Visualization automatically starts to play with time ratio 1 once initialization is finished, unless specified otherwise in the VTS project.

End of a visualization

The visualization can be stopped directly by clicking the top-right cross button to close the Broker. It also stops once all client applications have been closed by the user, or if the VTS configuration utility is closed (if the visualization was started from there).

Time management

Controls and information regarding time are available in all Broker display modes. The following interface allows interacting with visualization time:

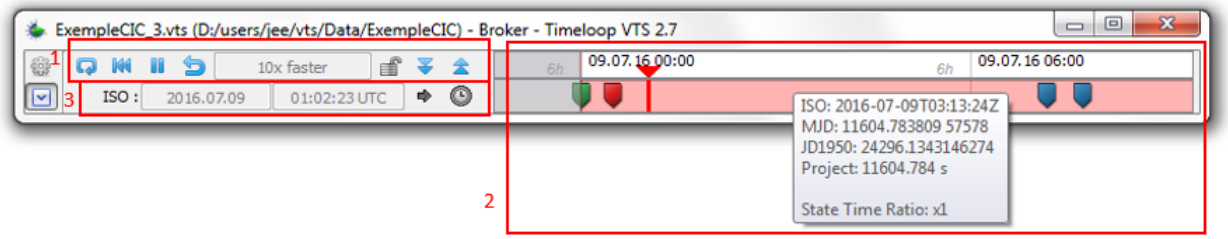


Fig. 1.1: Time control areas

Time controls

The buttons in area (1) control time in all client applications:

- **Time Loop**: Toggles loop playback for the visualization.
- **Restart**: Resets time to the project's start date. The play/pause status remains unchanged.
- **Play/Pause**: Plays/pauses the time flow. Pausing does not prevent interaction with the visualized entities in the client applications, or navigation in visualization time using the timeline.
- **Reverse Time**: Every click on this button inverts the time flow direction. The time ratio remains unchanged.
- **Use Scenario State Time Ratio**: When this button is unchecked, the user have a full control over the time ratio. When the button is checked, every time the time bar crosses a state, the state time ratio is applied to the visualization. The modification of the time ratio value is only applied when the `Use Scenario State Time Ratio` is toggled from `OFF` to `ON`.
- **Slower**: Decreases the time ratio. Two clicks on this button result in a 5 times slower time flow. The current time ratio is displayed next to this button.
- **Faster**: Increases the time ratio. Two clicks on this button result in a 5 times faster time flow.

Timeline

The timeline (2) displays the visualization's time range. Dragging the cursor controls the visualization's current time. While moving the cursor, the current time is kept updated in all client applications synchronously. For finer or coarser-grained time control, the timeline can be zoomed in/out using the mouse wheel.

Current time

A text area (3) displays the current visualization time. The default format is the ISO time format (date and time in UTC). The arrow button circles through other available time formats: CNES julian day (JD1950, fractional days, with reference date January 1st, 1950) and modified julian day (MJD, days and seconds, reference date November 17th, 1858). The `Edit date...` button pops up a dialog in which the current visualization time can be accurately defined (in all of the above time formats).

1.3 Software requirements for VTS

This page describes the software requirements for proper operation of the VTS toolkit.

1.3.1 Write access permission

VTS needs write access permission in its whole folder hierarchy. Many tools create temporary files for configuration and visualization. Moreover, installation of new VTS clients consists in copying the client directory in the `VTS/Apps` folder.

1.3.2 System font size

VTS is currently designed with a compact display. Some UI might be better looking by setting the font scaling factor on Windows: Right click on Desktop, Personalize, Display, Check SMaller (100%)

1.3.3 Supported operating systems

List of supported operating systems:

- Windows 7
- Windows 10
- Linux CentOS 6 32/64bits
- Linux CentOS 7

VTS is known to work in other major Linux distributions although not officially supported (additional packages may be required).

1.4 Hardware requirements for VTS

This page describes the hardware requirements for proper operation of the VTS toolkit.

1.4.1 Recommended system requirements

- Core i5 processor or better
- 4 GB RAM memory
- Video card (nVidia is recommended for Celestia) with 1 GB memory
- 1 GB free disk space

For nVidia video cards, prefer midrange cards : GeForce [23456]50 series.

Here are some example systems on which VTS would run very smoothly:

Compo-nent	Laptop	Desktop
Processor	3rd gen Intel Core i7-3610M (3.30GHz, 6MB)	3rd gen Intel Core i7-3770 (3.40GHz, 8MB)
Memory	Dual-channel SDRAM DDR3 4 GB, 1600 MHz	Dual-channel SDRAM DDR3 8 GB, 1600 MHz
Video	NVIDIA GeForce GT 650M 2 GB	NVIDIA GeForce GT 640 1 GB
Hard drive	SATA 1TB 5400 rd/min	SATA 1TB 7200 rd/min

1.4.2 Minimal system requirements

- 1GHz processor
- Dedicated video card with 128 MB memory (nVidia GeForce 5000 or equivalent), embedded video card for modern processors (Intel HD Graphics 3000)
- 512 MB memory
- 500 MB free disk space

Such a system will run VTS, but won't handle the more realistic rendering options in Celestia.

1.4.3 Video card drivers

In order to get the best performance from a machine, video card drivers should be kept up-to-date. This will solve the majority of display problems encountered while using Celestia.

Driver updates are available on the website of the card's chipset maker (nVidia, ATI or Intel), or on the website of the system maker for some laptops.

Issues with Celestia and sensor swath: if Celestia closes forcefully while running a visualization with sensor swath enabled, it may be due to outdated graphics card drivers. If drivers are up-to-date and the problem persists, with an Intel chipset, it may be because the chipset is incompatible with the rendering of sensor swath items. Sensor swath must then be disabled for the project (see http://www.opengl.org/wiki/FAQ#Why_is_my_GL_version_only_1.4_or_lower.3F).

CHAPTER

TWO

GENERAL

2.1 Conventions in VTS

As in every software, VTS sets up conventions for some of its elements. This page lists those conventions.

2.1.1 Frames

In VTS, frame axes form an orthonormal direct basis.

2.1.2 Quaternions

The first element of a quaternion corresponds to its scalar part.

2.2 Date formats in VTS

2.2.1 Formats list

Date format	Abbreviation	Example: May 17th, 2011, 16h22	Distance to JD1950	Leap seconds	Usage in VTS
CNES Julian day	JD1950	22416.681944444440		Yes	• Synchronization protocol • Display • Project file
Modified julian day	MJD	55698 58920.000000	33282	Yes	• CIC/CCSDS files • Display • Project file
Calendar date	ISO	2011.05.17 16:22:00	NA	Yes	Display
ISO 8601 date (forced UTC)	ISO	2011-05-17T16:22:00.000Z	NA	Yes	Project file
Celestia Julian day	jjCEL	2455699.181944	2433282.5	No	Celestia data files
Project-relative date	relative	*with project start*: May 15th, 2011, 12h39 186180.000 s	variable	Yes	Display

The internal date representation in VTS uses the reference date of JD1950, and is expressed with two fields for days and seconds (like MJD).

VTS uses the time system UTC both for display and internal calculations.

2.2.2 Entering a date

When a date needs to be specified anywhere in VTS, the same user interface is used. The date can be entered in the *JD1950*, *MJD* and *ISO* formats.

The dialog also contains 3 buttons :

- The first button allows to copy the date in the clipboard with a specific format.
- The second button pastes a date from the clipboard.
- The third button sets the date to the current date.

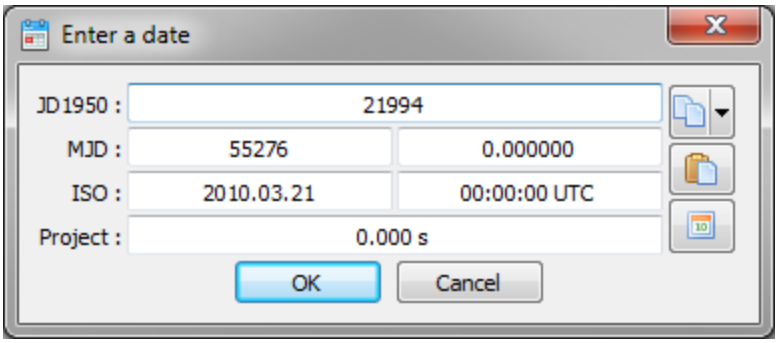


Fig. 2.1: Date dialog in VTS

2.3 Data sources in VTS

There are three types of data sources in VTS:

- `Fixed`: a fixed value
- `File`: a time-dependent value read from a file
- `Stream`: a value provided by a network stream

2.3.1 *Fixed* data source

This data source provides a constant, user-defined value. The actual value is defined by the user upon selection of the source.

This source can be used eg. for the position and orientation of fixed satellite components and sensors.

2.3.2 *File* data source

This data source is based on a file, specified by the user upon selection of the source. The values are read from the file, which must follow the standard *CIC/CCSDS format*.

This source can be used for the ephemerides of celestial bodies, satellites, as well as for the position and orientation of mobile parts on satellites. It is most adequate for *offline* visualization, once the position and attitude data have been saved to files.

2.3.3 *Stream* data source

This data source is based on a network stream, defined by the user upon selection of the source.

This source can be used for the ephemerides of celestial bodies, satellites, as well as for the position and orientation of mobile parts on satellites. It is most adequate for *online* visualization, in which the position and attitude data are streamed on-the-fly by external applications, such as a simulator.

The *Stream* data source provides two modes of operation:

- `INTERPOL`: in order for visualization clients (SurfaceView, Celestia, etc.) to use interpolated values, the Broker will introduce a slight delay between data transmission and time synchronization with visualization clients.
- `DIRECT`: the data update rate is not taken into account while computing time synchronization delay. Beware that no extrapolation is done in standard VTS clients.

See also the documentation for *Real-time VTS* in the *Synchronization protocol for VTS clients* chapter.

2.4 Position of objects in VTS

The position of an object in VTS is defined as a translation from the object's reference frame towards its location. The position can be fixed, sampled, or streamed. This section is dedicated to the contents of position data, no matter its source. For a definition of the various data sources, see the *Data sources in VTS* section.

2.4.1 Reference frames

Each object in VTS has its position defined in a reference frame. For example, the position of solar arrays is relative to the satellite frame.

An object is located at the origin of its reference frame when its position vector is null.

Reference frames for the positions of various object types are defined in the table below:

Object type	Reference frame
Central body	Parent object's frame
Ground station	Central body's local frame
Satellite	When a satellite orbits around a celestial body, two reference frames are supported, depending on data input type : • EME2000 (Earth Mean Equator at epoch J2000) • Central body's local frame (only MEM files with lat/long/alt) When another satellite is defined as the center of the reference frame, the following frame types are available (also applied to orientation) • EME2000 • BodyFixed • TNW • QSW
Component	• Satellite's frame for a top-level component • Parent component's frame for components of other levels
Satellite sensor	• Satellite's frame for a top-level sensor • Parent component's rotation center for sensors attached to a component
Ground station sensor	Ground station's frame

2.4.2 Position of satellite components

The position for a top-level component is defined in the satellite's frame. The position for a sub-component, i.e. a component not directly attached to a satellite but rather to another component, is defined in its parent component's frame.

When a component is translated or rotated, all its sub-components are translated and rotated as well. This can be illustrated by the operation of a robotic arm: each segment is animated locally relative to the previous segment.

There is a special case to the relative position of sub-components, when using 3D files containing the position of sub-components relative to the satellite's frame. This case is discussed in section *3D file format in VTS*.

2.4.3 See also

- *Conventions in VTS*
- *3D file format in VTS*
- *Examples for position and orientation of objects in VTS*

2.5 Orientation of objects in VTS

The orientation of an object in VTS is defined as a rotation from the object's reference frame towards its attitude. The orientation can be fixed, sampled, or streamed. This section is dedicated to the contents of orientation data, no matter its source. For a definition of the various data sources, see the *Data sources in VTS* section.

2.5.1 Reference frames

Each object in VTS has its orientation defined in a reference frame. For example, satellites around Earth use reference frame EME2000.

An object is aligned with the axes of its reference frame when its rotation vector is null. In the case of a satellite defined by a 3D file (see the *3D file format in VTS* section), the X axis of the 3D mesh is aligned with the X axis of the EME2000 frame. The same applies for the Y and Z axes.

Reference frames for the orientations of various object types are defined in the table below:

Object type	Reference frame
Central body	Parent object's frame
Ground station	-Z : body center +X : towards geographic North pole At poles : undefined
Ground station sensor	Ground station's frame
Satellite	When a satellite orbits around a celestial body: • EME2000, Earth Mean Equator at epoch J2000 When another satellite is defined as the center of the reference frame, the following frame types are available (also applied to position) • EME2000 • BodyFixed • TNW • QSW
Component	Satellite's frame for a top-level component Parent component's frame for components of other levels
Satellite sensor	Satellite's frame for a top-level sensor, component frame for sub-level sensors. Aligned with the satellite's frame and the sensor points along the Z axis. Parent component's frame for a sensor attached to a component

2.5.2 Rotation center

The rotation center defines the point around which a 3D object is rotated. It depends on the object type, and is defined in the table below:

Object type	Rotation center	Reference frame for the rotation center
Central body	Central body's center	Central body's frame
Ground station	Ground station's position	Central body's frame
Ground station sensor	Sensor's position	Ground station's frame
Satellite	Center of gravity (user-defined)	Satellite's frame
Component	Rotation center (user-defined)	Component's frame
Satellite sensor	Sensor's position	Satellite's frame for a top-level sensor Parent component's frame for a sensor attached to a component

2.5.3 Orientation types

In VTS, an orientation can be defined by several types of rotations in the reference frame.

Quaternion

A quaternion defines a rotation from an object's reference frame towards its local frame.

The convention used for a quaternion sets its real part as the first component.

A null rotation is defined by the following quaternion:

```
1.0 0.0 0.0 0.0
```

All orientations can be reached. Quaternions are normalized by VTS. An all-zeroes value (0 0 0 0) is invalid.

Euler angles

Euler angles define a sequence of three rotations from an object's reference frame towards its local frame.

The convention used for Euler angles defines the order of rotations `Z`, `X'`, `Z''` as:

- Precession around axis `Z` of the reference frame
- Nutation around axis `X'` of the frame resulting from the precession
- Intrinsic rotation around axis `Z''` of the frame resulting from the two previous rotations

Angles are expressed in degrees.

All orientations can be reached. There is no invalid value.

Axis and angle

Axis and angle rotations allow easy definition of the orientation of objects physically rotating around an axis, e.g. solar arrays.

The rotation axis must be defined, i.e. non null (0 0 0). Its coordinates are expressed in the object's reference frame.

The rotation angle in degrees is applied to the object around the rotation axis, in the counterclockwise direction. A positive angle implies a counterclockwise rotation around the axis ; convention dictates that this corresponds to *screwing* along the direction of the axis, i.e. in clockwise direction while looking along the direction of the axis.

Direction

A direction defines a rotation so that the `X` axis of the object points in the specified direction. It should be noted that this kind of orientation leaves a degree of freedom (3rd rotation of the Euler angles), and should thus only be used for solids of revolution around the `X` axis (e.g. a vector along the `X` axis).

Azimuth and elevation

Azimuth and elevation define a sequence of two rotations, first around axis `Z` then around axis `−Y'` (resulting of the previous rotation). If both rotations are null, the object points towards `X`. As for the direction orientation, this kind of orientation leaves a degree of freedom around the aim axis. Notice that in this orientation mode, conventions for sensor orientations pointing toward `+Z` is changed for pointing towards `+X`.

2.5.4 Validity domains

The different kinds of orientations are not relevant for all object types. The table below lists the valid orientation kinds for all object types. It should be noted that combinations marked as not relevant are still available in VTS, so that they may be used should the need arise. Combinations marked with a dash are however not possible to use with VTS.

Object type	Quaternion	Euler angles	Axis and angle	Direction	Azimuth elevation
Central body	**Yes** *E.g.: Comet*	Possible	No	No	No
Satellite	**Yes** *E.g.: Attitude*	Possible	No	No	No
Component	**Yes** *E.g.: Mobile part*	**Yes** *E.g.: Onboard instrument*	**Yes** *E.g.: Solar arrays*	**Yes** *E.g.: Representation of a vector*	**Yes** *E.g.: Direction of an antenna*
Satellite sensor	**Yes** *Ex.: Star tracker*	**Yes** *Ex.: Star tracker*	No	**Yes**	**Yes**

NB: Currently, the orientation of ground stations and ground station sensors cannot be modified.

2.5.5 See also

- Examples for position and orientation of objects in VTS
- *Conventions in VTS*
- *3D file format in VTS*

2.6 Examples for position and orientation of objects in VTS

This section illustrates the positioning and orientation of an object with regards to its parent satellite. Relevant definitions can be found in the *Position of objects in VTS* and *Orientation of objects in VTS* sections.

2.6.1 Methodology

In order to fully understand the mechanisms at hand in the 3D views of VTS for the display of satellites and their parts (components, sensors, etc.), the following parameters will be analyzed:

- Position relative to the satellite (or in a more general way, relative to the parent object)
- **Orientation, with the following parameters:**
  - Center of rotation
  - Axis of rotation
  - Angle of rotation

The position and orientation (be it quaternion, Euler angles, axis and angle, etc.) of an object can be defined in the *Position/orientation* dialog of the object. For components, the center of rotation can be defined in the *3D Properties* dialog. Refer to the *VTS configuration utility user manual* for more details.

The 3D object used for this demonstration is the 3-axes trihedral usually used in VTS to materialize reference frames. The X, Y, and Z axes are respectively colored in red, green, and blue. This object represents here the satellite or any parent object. The length of each axis is one meter. Then, the same model is used, scaled down and tarnished, to represent the object positioned by the transformations described below.

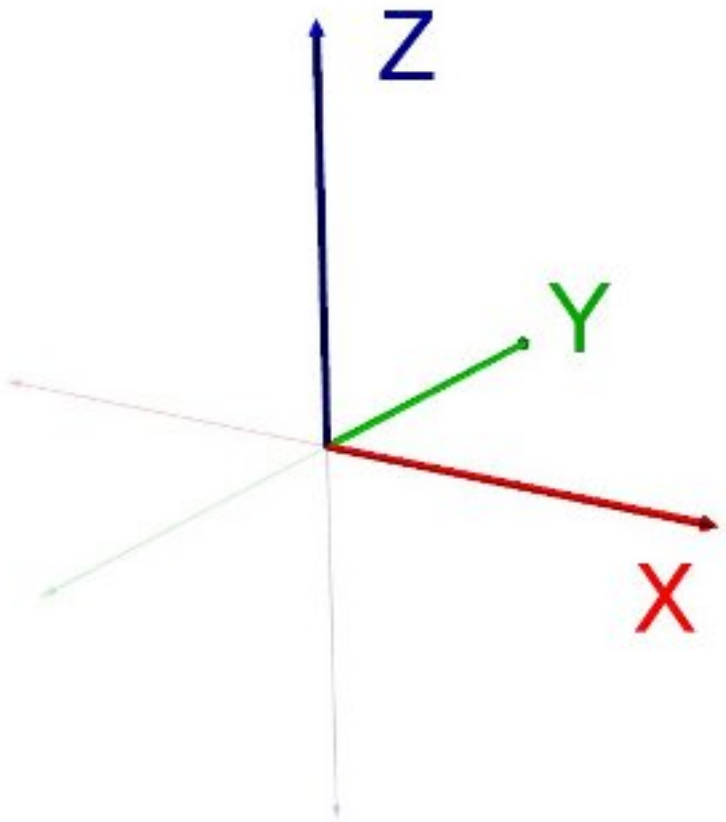


Fig. 2.2: Materialization of the axes of the reference frame

2.6.2 List of tests

Number	Designation	Position	Center of rotation	Axis of rotation	Angle of rotation
1	No modification	X=0 Y=0 Z=0	X=0 Y=0 Z=0	X=0 Y=1 Z=0	0°
2	Single translation	X=1 Y=0 Z=0	X=0 Y=0 Z=0	X=0 Y=1 Z=0	0°
3	Single rotation	X=0 Y=0 Z=0	X=0 Y=0 Z=0	X=0 Y=1 Z=0	30°
4	Translation and rotation	X=1 Y=0 Z=0	X=0 Y=0 Z=0	X=0 Y=1 Z=0	30°
5	Offset rotation	X=0 Y=0 Z=0	X=1 Y=0 Z=0	X=0 Y=1 Z=0	30°
6	Translation and offset rotation	X=1 Y=0 Z=0	X=1 Y=0 Z=0	X=0 Y=1 Z=0	30°

2.6.3 Illustration of the result

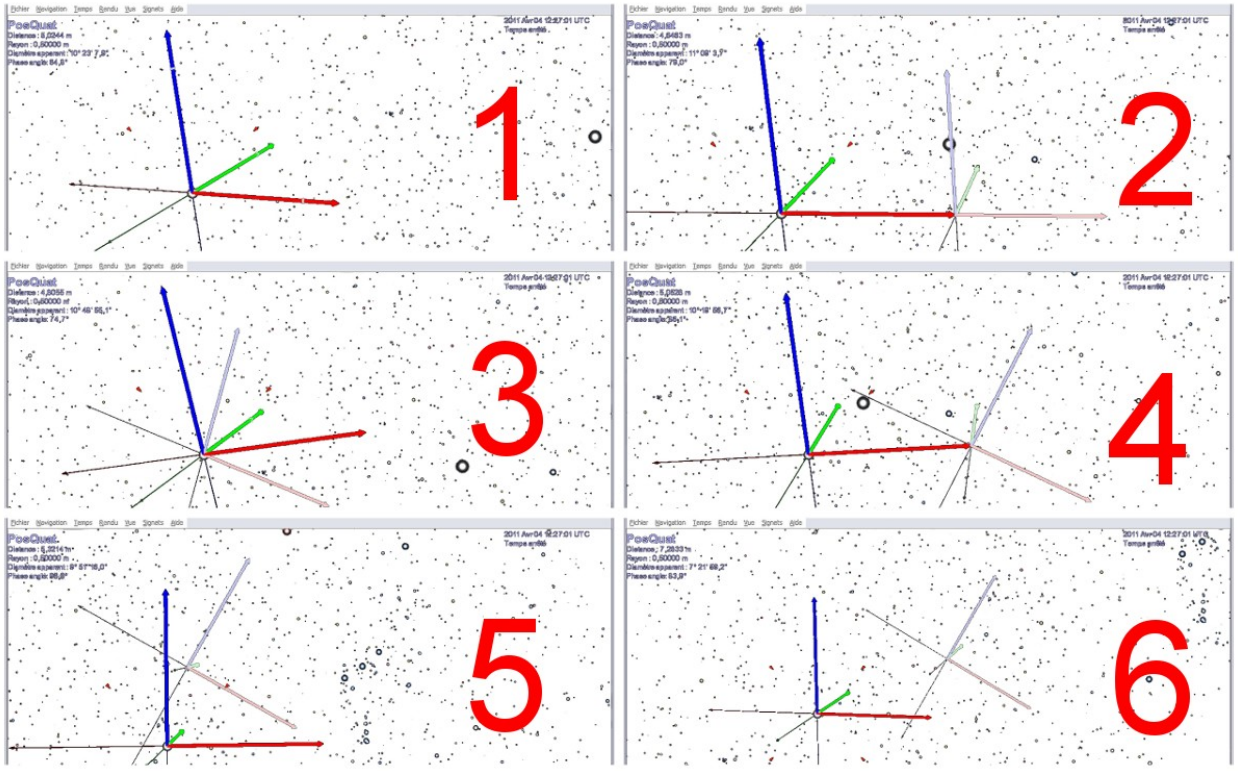


Fig. 2.3: Examples of various positions and orientations

2.6.4 Description of the result

Case 1: no modification

- Position: X=0 Y=0 Z=0
- Center of rotation: X=0 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 0°

The position and the angle of rotation are null. In this case, the object is at the same position and is aligned with its reference frame. It should be noted that since the angle of rotation is null, the axis and center of rotation have no influence.

Case 2: single translation

- Position: X=1 Y=0 Z=0
- Center of rotation: X=0 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 0°

In this case, the position of the object has been modified. Its value defines a position 1 meter forward along the X axis. The object is now centered at the end of the X axis of the reference frame.

Case 3: single rotation

- Position: X=0 Y=0 Z=0
- Center of rotation: X=0 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 30°

The object's position vector is null and its center is located at the origin of its reference frame.

A single rotation has been applied. The center of rotation is null. The axis of rotation is 0 1 0, i.e. the rotation is counterclockwise around the Y axis (as if *screwing* in the direction of the axis). The diagram shows that the Y axis is shared by the two objects, while the X and Z axes have rotated 30°. The direction of the rotation can also be observed, clockwise when looking in the direction of the axis of rotation.

Case 4: translation and rotation

- Position: X=1 Y=0 Z=0
- Center of rotation: X=0 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 30°

This case combines both cases. It illustrates the order in which the transformations are applied: **the object is first rotated, then translated to its position**.

Case 5: offset rotation

- Position: X=0 Y=0 Z=0
- Center of rotation: X=1 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 30°

For a single rotation, the origin of the reference frame is used as rotation center. The rotation center can be modified so that the object is rotated around another point (e.g. for solar arrays). The coordinates of the center of rotation are 1 0 0. The object is rotated 30° around an axis located at the end of the red X axis, and directed along the green Y axis. Note that this rotation axis is not displayed on the diagram, hence the object appearing as if *translated* and not only rotated, due to the rotation center being offset.

Case 6: translation and offset rotation

- Position: X=1 Y=0 Z=0
- Center of rotation: X=1 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 30°

This example adds a translation of 1 meter along the X axis on top of the transformations of case 5. The diagram confirms that rotation occurs before translation.

2.7 Backward compatibility in VTS

2.7.1 Backwards compatibility of project files

Principle

VTS features backwards compatibility of its project files. This means that project files from an earlier version of VTS will still be read correctly by newer versions of the toolkit. However, they will always be saved in the most recent file format if they are modified.

This mechanism is available from VTS 1.1 onwards.

Compatibility table

From revision	VTS version	Revision tag
From r8625 onwards	3.5, 3.5.1, 3.5.2, 3.5.3	<project>
From r8200 to r8624	3.4.2	<project>
From r7408 to r8199	3.3	<project>
From r6842 to r7407	3.2	<project>
From r6541 to r6841	3.1.1	<project>
From r6016 to r6540	3.0.2	<project>
From r4982 to r6015	3.0	<project>
From r4437 to r4981	2.7, 2.7.1, 2.8	<project>
From r3829 to r4436	2.6	<project>
From r3244 to r3828	2.4, 2.5	<project>
From r1991 to r3243	2.0, 2.1, 2.1.1, 2.2, 2.3	<project>
From r1697 to r1990	1.3.1	<project>
From r1554 to r1696	1.3.0	<project>
From r1246 to r1553	1.2.3	<project>
From r1050 to r1245	1.2.1	<general>
From r1000 to r1049	1.1	<general>
Before r1000	NA	NA

2.7.2 Upgrading VTS

When a new version of VTS is available and if VTS can reach the CNES webpage (, a message is displayed to indicate the availability of a new VTS version. This message only appears when using the VTS configuration utility.

2.8 Object paths in VTS

The various objects displayed by VTS can all be addressed according to a naming scheme. For practical reasons, this naming scheme is directly derived from the one used by Celestia.

2.8.1 Allowed characters

Since VTS use the *slash* character as a separator, it cannot be used in objects' identifiers.

The following characters are forbidden and will prevent VTS to start the visualization:

- Slash /
- Antislash \

More generally, other characters than the following are considered unsafe:

- Letters, both lowercase and uppercase
- Digits : 0, 1, 2, etc.
- Space
- Dot `.`, comma `,` colon `:`, and semicolon `;`
- Dash -
- Underscore _
- Degree °
- Percent `%`
- Dollar `$`
- At `@`
- Amperstand `&`
- Parenthesis `(` and `)`
- Pipe `|`
- Equal sign = and plus sign +

2.8.2 Definitions

The table below lists all the definitions used to identify objects in VTS:

Identifier	Description	Examples
name	Name of the object	`Earth` `GS`
fullName	Full identifier of object	`Sol/Earth` `Sol/Earth/CubeSat/GS`
parentPath	Identifier of the object's parent	`Sol` `Sol/Earth/CubeSat`
celestiaRefName	Name of the object's reference in Celestia	`Earth` `GS_ref`
celestiaParent-FullName	Path to the object's parent in Celestia	`Sol` `Sol/Earth/CubeSat_ref/CubeSat`
celestiaFullRef-Name	Full path to the object's reference in Celestia	`Sol/Earth` `Sol/Earth/CubeSat_ref/CubeSat/GS_ref`
celestiaFullName	Full path to the object in Celestia	`Sol/Earth` `Sol/Earth/CubeSat_ref/CubeSat/GS_ref/GS`

2.8.3 Complex names for Points and Regions

In order to easily differentiate many points and regions in the VTS configuration utility, it is possible to use a special naming rule ("complex names"). When a point or a region has pipes in its name (eg: `Europe|France|Point1`), the name is interpreted as a hierarchical naming, and only the last part of the name is displayed in OmniView. This allows the user to easily differentiate many points in the VTS configuration utility, without cluttering the OmniView display.

2.8.4 Central body of an object

Objects orbiting Earth are identified starting with the full path to Earth, i.e. `Sol/Earth`.

2.8.5 Usage

The object name (*name*) and the path to its parent (*parentPath*) are the two main elements visible to VTS users. They can be found most notably in:

- The *VTS project file format*
- The *Synchronization protocol for VTS clients*

2.9 Central bodies in VTS

Central bodies are celestial objects (such as planets, moons or comets) that can be used to define a mission. They support multiple features such as **region of interests**, **ground stations** and **map layers**. Satellites defined in a VTS project must orbit one of the defined celestial bodies.

2.9.1 Categories of bodies

All celestial bodies share the same properties in VTS, but a distinction is made between bodies *supported* by VTS and *custom* bodies.

For **supported celestial bodies** there is no need to define its ephemerides as they are "known" and configured by VTS.

For **custom celestial bodies**, the ephemerides and properties (such as its radius) must be defined by the user.

The table below lists the *supported* celestial bodies. All other celestial bodies are considered *custom* celestial bodies.

Celestial body	Parent body (Catalog)
Sun (Sol)	
Mercury	Sol
Venus	Sol
Earth	Sol
Moon	Sol/Earth
Mars	Sol
Jupiter	Sol
Saturn	Sol
Uranus	Sol
Neptune	Sol

2.9.2 Bodies definition

Central bodies ephemerides are defined in their **parent body**'s reference frame.

For supported bodies, if the parent body is different from the Catalog (see table above), the ephemerides must be specified by the user.

> **Warning:** The Moon uses Earth-centered ephemerides in catalog mode. Defining the Moon as a central body also requires defining the Earth in the project. If required, the Moon can be attached to the Sun by specifying custom ephemerides.

Existing natural bodies may also be partially supported: some celestial bodies can be supported out of the box by Celestia but not in other VTS applications. In this case, VTS will try to use existing features (such as textures) but will not use default ephemerides.

2.9.3 Origin of ephemeris

Default ephemeris mode

Client applications choose their default mode to select the central bodies ephemerides. They can use their own calculation or shipped data, or VTS ephemerides catalog. This option can be also used for application that don't animate the central body or if this information is not relevant for them.

In this mode Celestia uses its own calculation, and SurfaceView uses catalog ephemerides files.

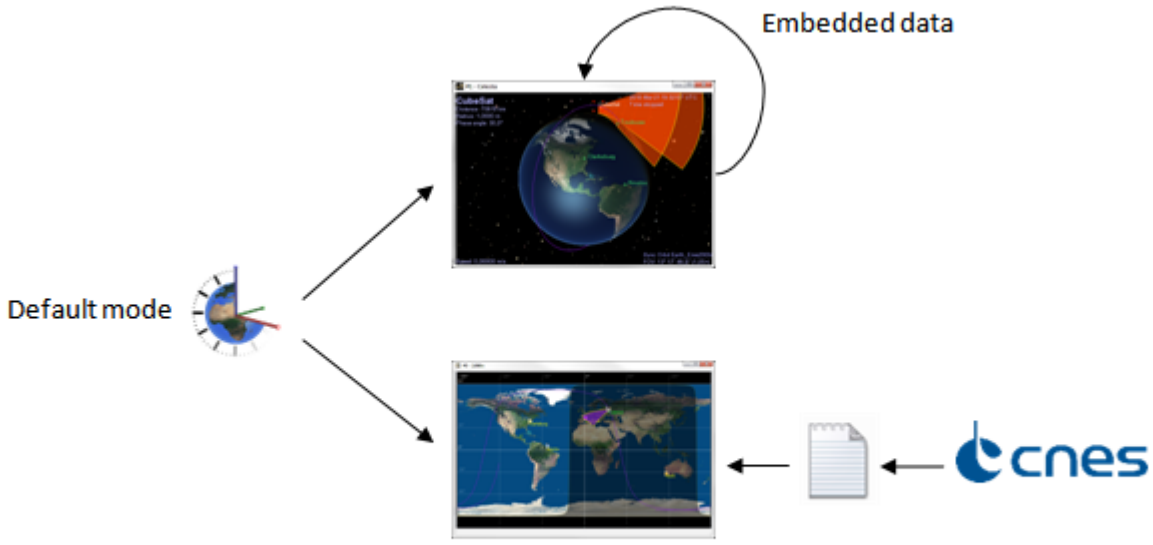


Fig. 2.4: Ephemeris source in default mode

Catalog ephemeris mode

Client applications use the VTS ephemeris catalog data in this mode. The ephemeris catalog contains CIC files describing the position and the orientation of the solar system planets between 01/01/1950 and 31/12/2100. Position files contain one point per day and orientation files contain 6 point per day. These files are provided by CNES.

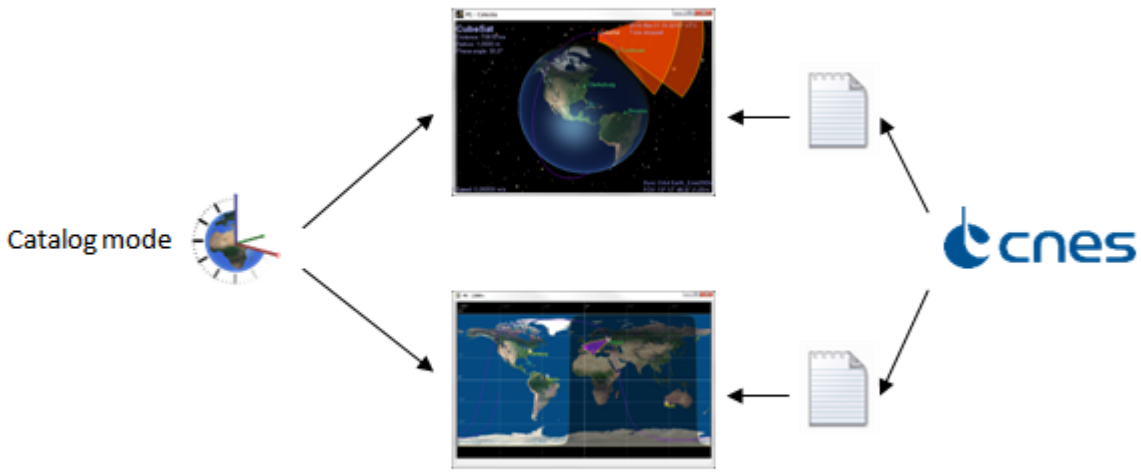


Fig. 2.5: Catalog ephemeris mode

Custom ephemeris mode

Redefine the ephemerides of an existing Celestia body or define the ephemerides of a body not available in Celestia. Custom ephemerides must be defined if the current body is not available in Celestia.

2.9.4 Central body frame

Coordinates of a central body are expressed in the heliocentric EME2000 frame.

North Pole of a central body is always the rotation North. It is used to display body axes and the planetographic grid.

2.10 Structure of satellites in VTS

The structure of satellites in VTS matches the organization of the elements composing a satellite.

This structure can be found in:

- the VTS configuration utility, in the tree hierarchy displayed in the left pane. Refer to the *VTS configuration utility user manual* chapter.
- the contents of the project file. Refer to the *VTS project file format* chapter.
- the *View Properties* and *3D Cameras* tabs of the Broker. Refer to the *Broker user manual* chapter.

The structure can be described as follows:

- A project contains central bodies and satellites
- A satellite is composed of a main component which defines the body of the satellite.
- A component may contain sensors.

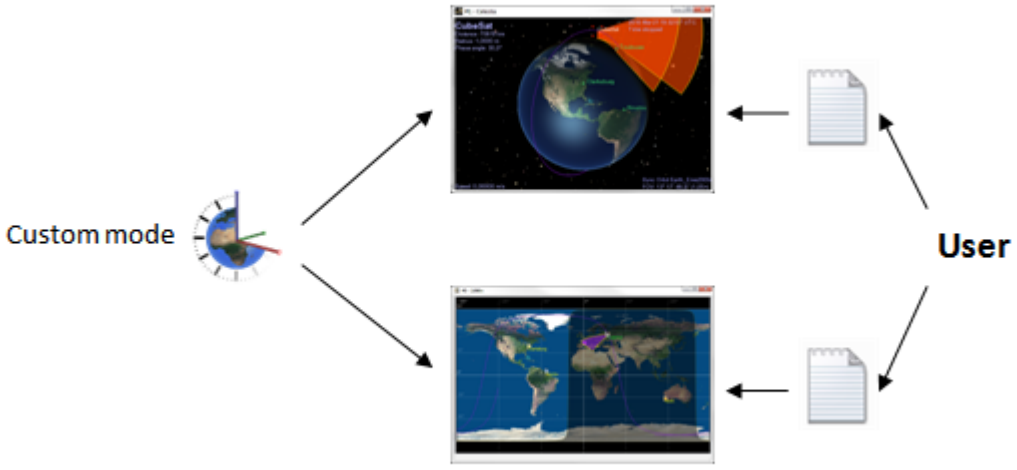


Fig. 2.6: Custom ephemeris mode

- A component may contain sub-components.
- These sub-components may themselves contain sub-components.

The naming scheme of objects relies on this structure, e.g. `Sol/Earth/CubeSat/GS`. See also the chapter about *Object paths in VTS*.

NB: Throughout this documentation and the VTS GUI, satellite components are sometimes referred to as "subparts".

2.11 Sensors in VTS

A sensor is either an onboard element on a satellite or an element of a ground station. This section presents the properties of sensors in VTS, and their representation in client applications.

2.11.1 Properties of a sensor

The sensors' base can be elliptical or rectangular for both satellites and ground stations, or an angular sector of space constrained in azimuth-elevation for ground stations only.

Elliptical or rectangular sensors

The cone is characterized by two half-angles around the X and Y axes (see the *Orientation of objects in VTS* chapter concerning the reference frame). These angles define rotations of the sensors's aim axis around the X axis (in the YZ plane) and Y axis (in the XZ plane).

Azimuth-elevation sensors

The azimuth-elevation sensor type can be used to represent an angular sector of space. This type of sensor makes more sense when defined inside a ground station. It is defined by a minimum and a maximum azimuth and a minimum and a maximum elevation.

When azimuth and elevation are equal to zero the sensor points towards X in the sensor frame. When azimuth is equal to zero and elevation is equal to PI/2 the sensor points towards Z in the sensor frame.

The mimimum and maximum azimuth angles are ordered, and the first angle is wrapped between [0,2PI] and the second angle is wrapped between [0;4PI].

The mimimum and maximum azimuth angles are defined in the interval [0;PI/2].

Sensors orientation

When orientated by a quaternion, euler angles or axis and angles, a sensor is defined by a conical aim volume around the Z axis, its apex being located at the origin of the sensor's local frame.

When orientated by a direction or elevation and azimuth, an additional rotation is made: X becomes the main axis, i.e. the sensor looks at the given direction. Since the third (self) rotation is not reliable, this orientation mode should be used only with elliptical sensors with similar X and Y apertures (circular sensors).

See the *Orientation of objects in VTS* chapter concerning the reference frame.

There is currently no limit to the effective range of a sensor. Half-angles are internally limited to 90° in OmniView and Celestia.

2.11.2 Visualization of a sensor

Satellite sensor

In the 2D view, a satellite sensor is represented by the intersection of its volume with its target central body.

In the 3D view, a satellite sensor is represented by its aim volume. In order not to overload the visualization, a maximum range is defined for the sensor in the VTS configuration utility. Beyond this range, the sensor's volume will not be displayed. As a rule of thumb, the maximum sensor range should be taken equal to the maximum distance between the satellite and the center of the central body. Beware that a too short range could result in an erroneous computation of the sensor volume's intersection with the sensor target.

Moreover, the 3D view allows rendering the scene from the point of view of the sensor, so that what the sensor perceives can be displayed (refer to the *3D Cameras* tab section in the Broker's user manual).

VTS also allows displaying the sensor's swath, both in SurfaceView and Celestia. **Important note:** a sensor coverage nadir optimization can be activated. See the *SurfaceView user manual* for more information.

Caution: To preserve reasonable performances on the widest possible range of computers, the computation algorithm for sensor swath in Celestia is approximate. This has the consequence that when a sensor covers a large portion of its target body, the computed sensor swath is erroneous near the sides of the body disc.

In case of forceful close of Celestia when visualizing sensor swath: see the section on *Hardware requirements for VTS*.

Ground station sensor

A ground station sensor has the same parameters as a *satellite sensor*. It is oriented in its ground station's frame (see *Orientation of objects in VTS*).

A ground station always owns a "Default" main sensor attached to it, with a fixed orientation : its aim (Z) axis is directed from the body center towards the station position. This sensor can be used to obtain the station visibility circle : to show it at minimum elevation, the half-angles should be set to the complementary angles of the half-angles (90° - half-angle around X or Y).

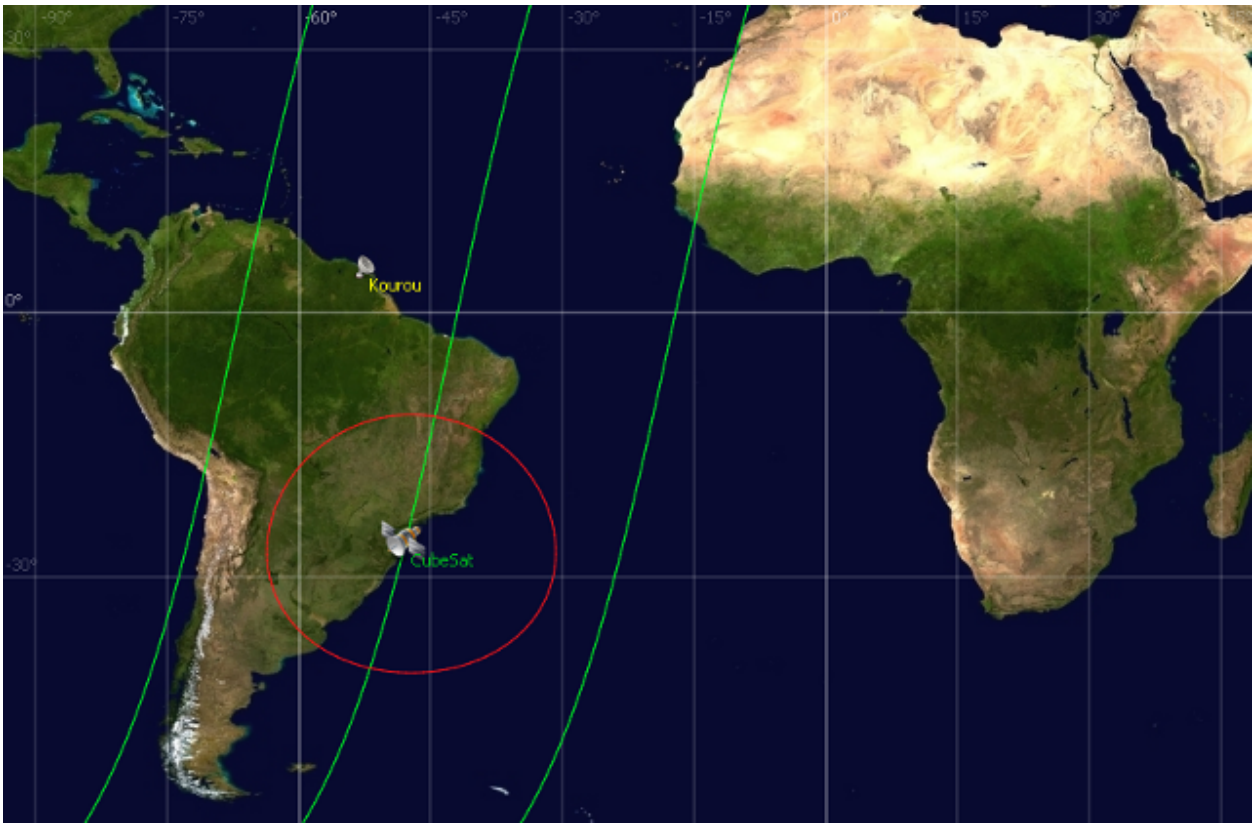


Fig. 2.7: Elliptical sensor in SurfaceView

Fig. 2.8: Rectangular sensor in SurfaceView

Secondary sensors can be created attached to a ground station. These sensors can be freely oriented in the ground station's frame.

Visualization of ground station sensors in 2D

In the 2D view, a ground station sensor is represented by the projection of the base of its aim volume, at a given altitude, back on the surface of its central body. The altitude can also be set to the altitude of one of the project's satellites, and will then vary with time. Attaching more satellites will introduce a second dashed line which displays the visibility area for the higher and the lower tracked satellites.

Visualization of ground station sensors in 3D

The "Default" main sensors are not displayed in Celestia. There is no station visibility circle neither. Orientable secondary sensors are displayed as a aim volume at a fixed altitude.

2.12 Physical masks in VTS

A physical mask describes irregularities of the aiming volume of a ground station due to obstacles (relief, buildings, ...). It applies to the full visibility area (no minimal elevation angle).

Mask point coordinates describe the contour of a closed shape in elevation/azimuth. To represent a line of sight, it's necessary to close to the surface along the horizon. Using this convention, any shape can be represented as a mask.

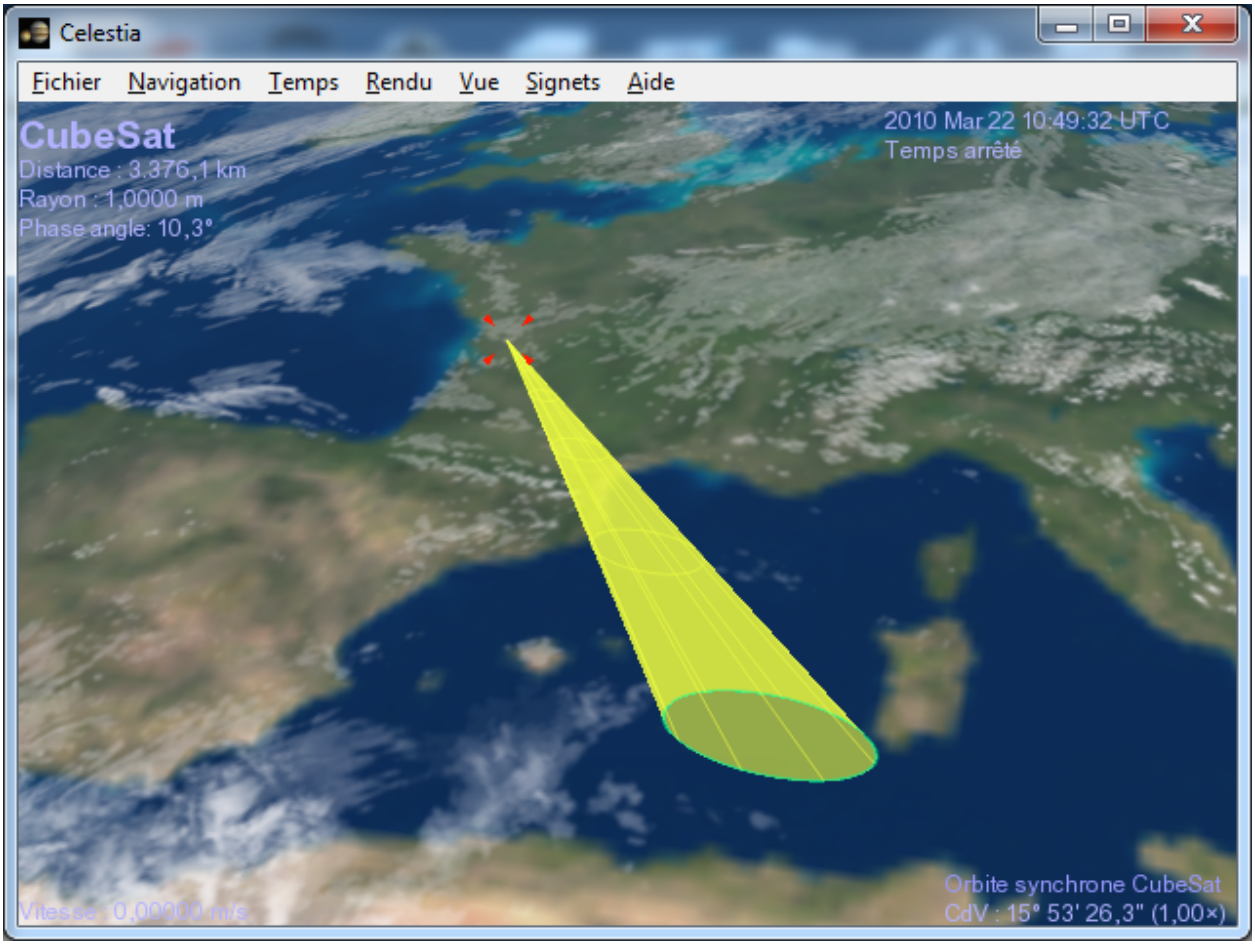


Fig. 2.9: Elliptical sensor in Celestia

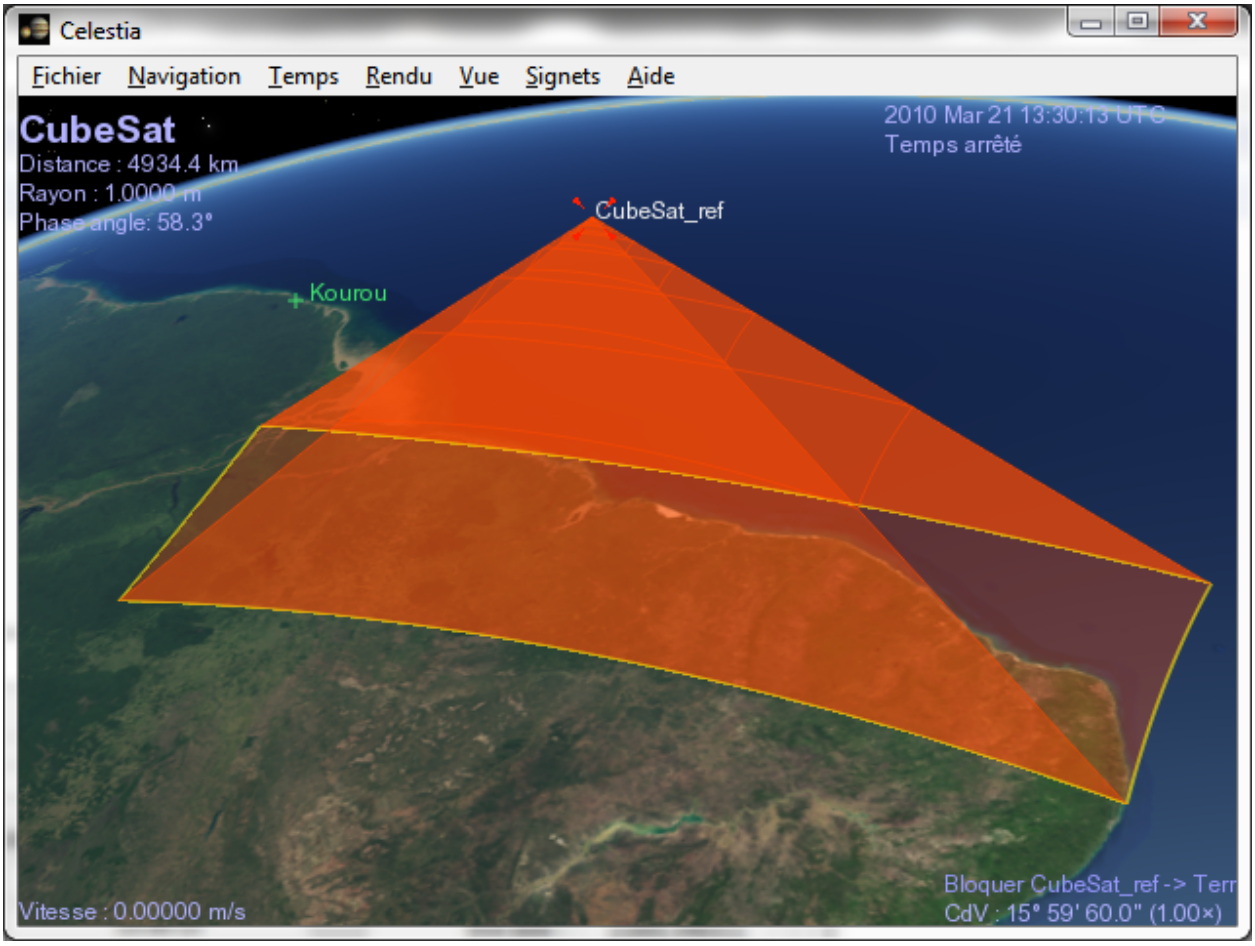


Fig. 2.10: Rectangular sensor in Celestia

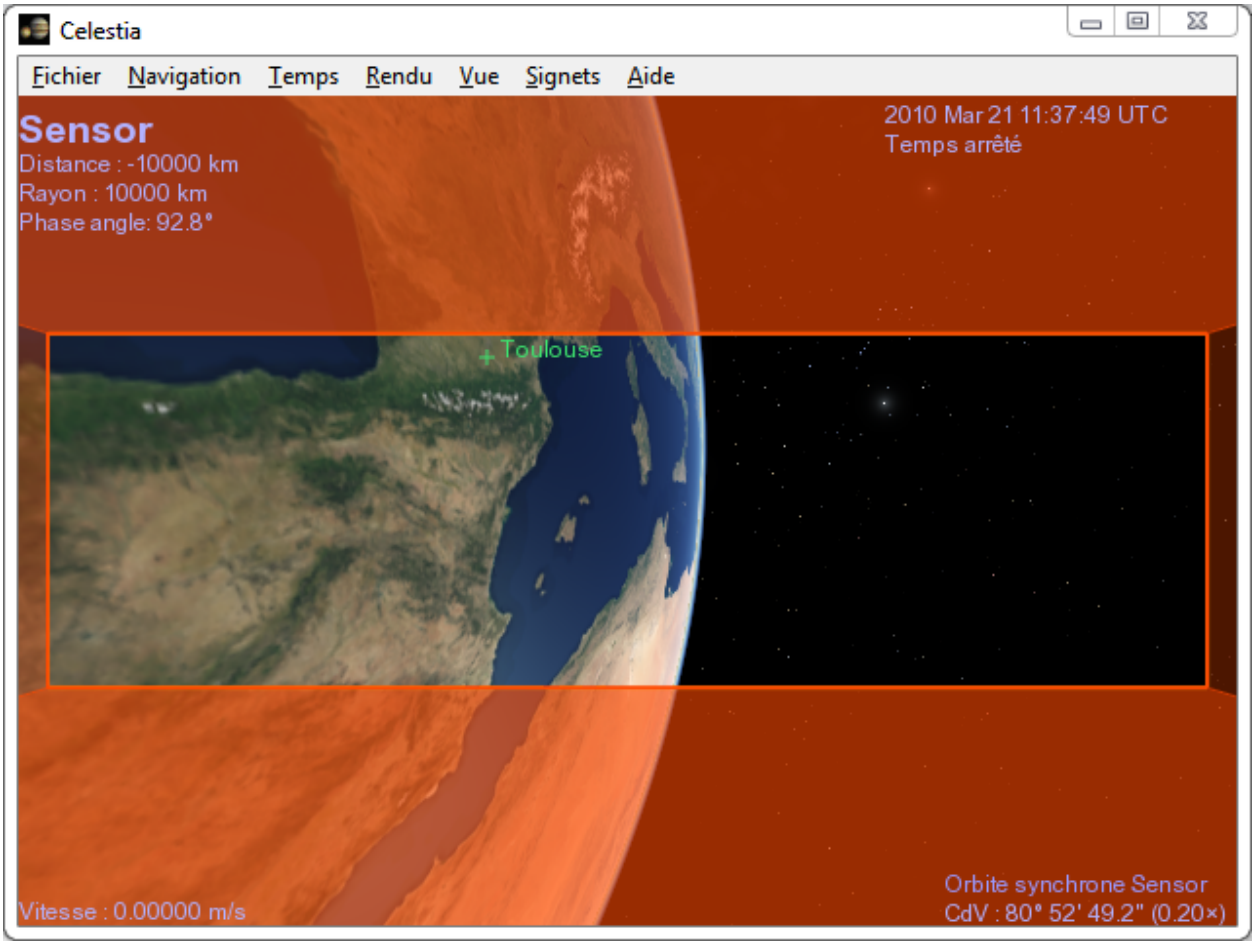


Fig. 2.11: Sensor camera in Celestia

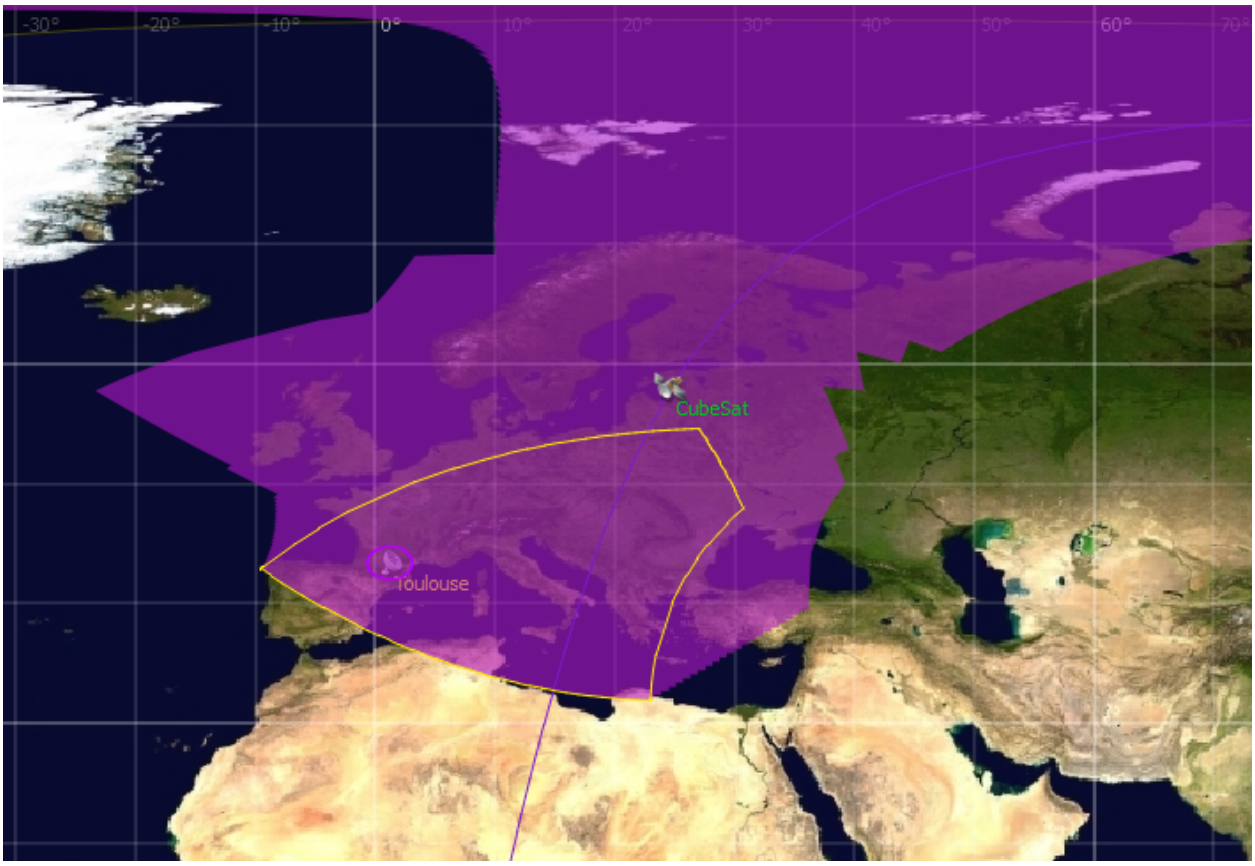


Fig. 2.12: Sensor swath in SurfaceView

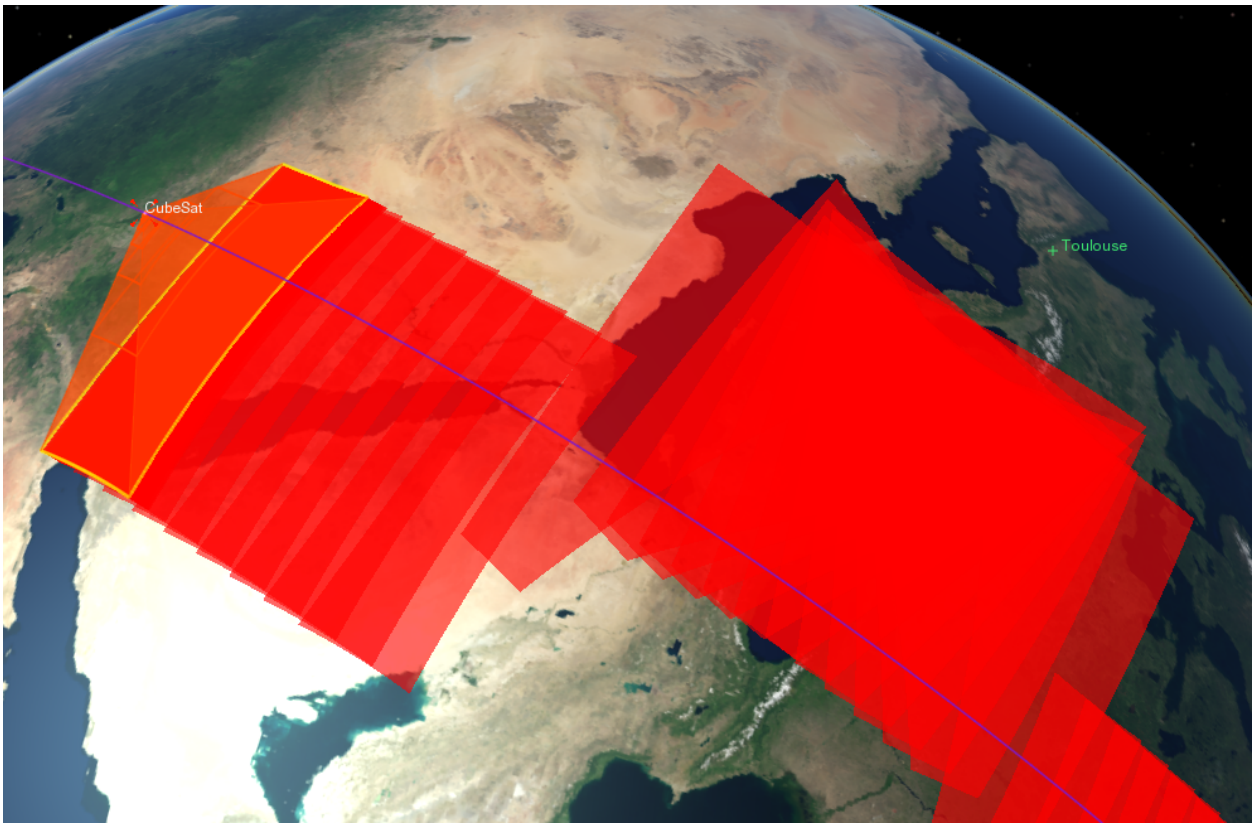


Fig. 2.13: Sensor swath in Celestia

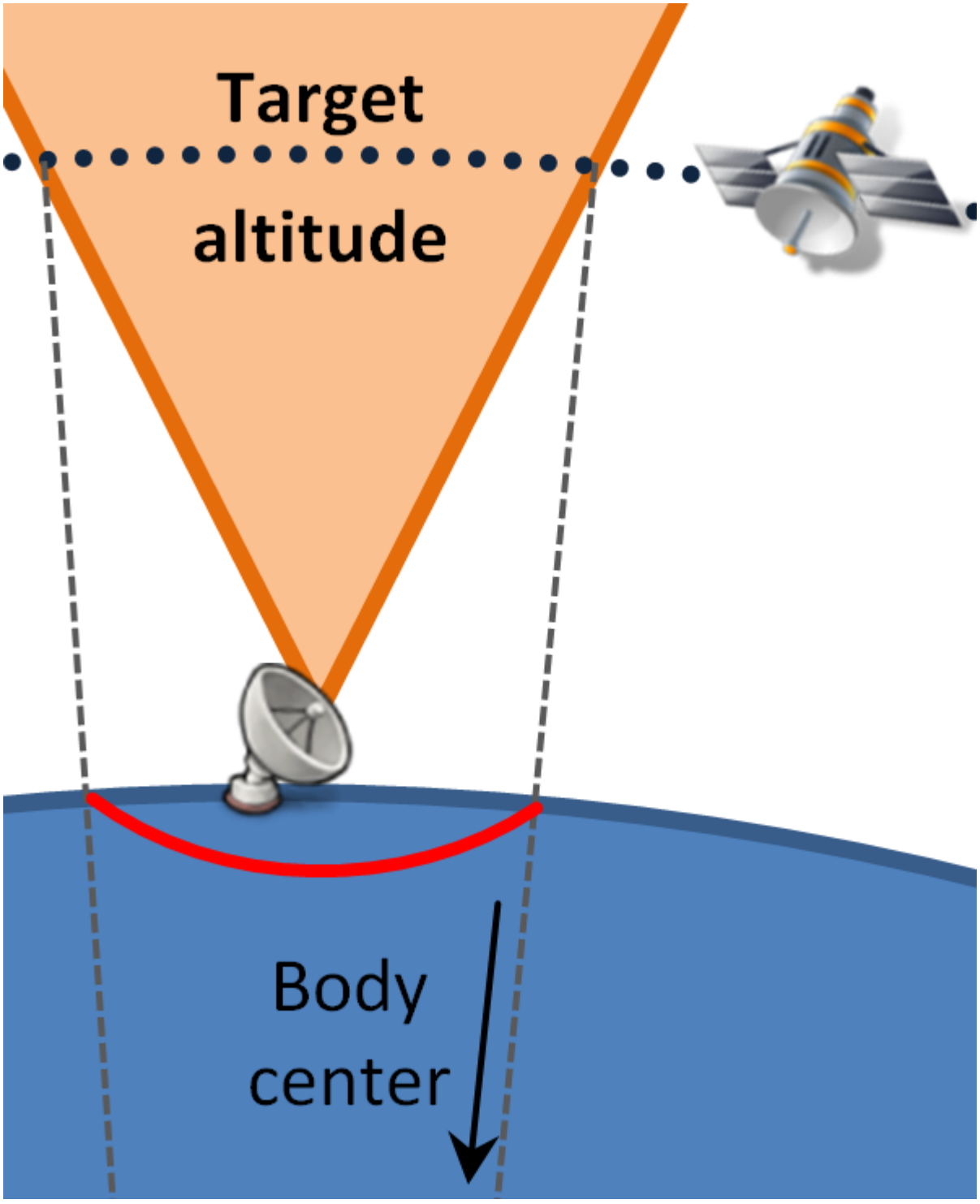


Fig. 2.14: Projection of a ground station sensor in SurfaceView

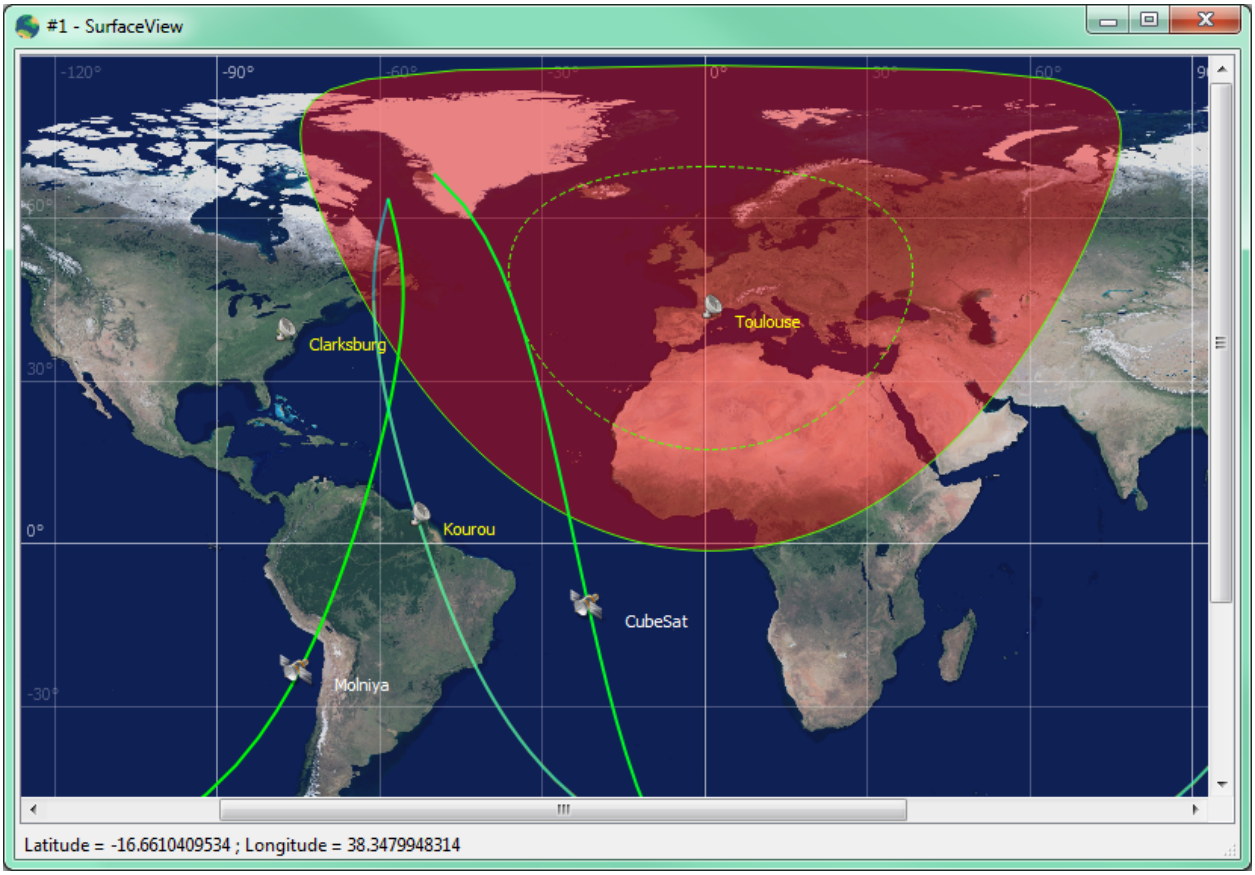


Fig. 2.15: Ground station sensor in SurfaceView

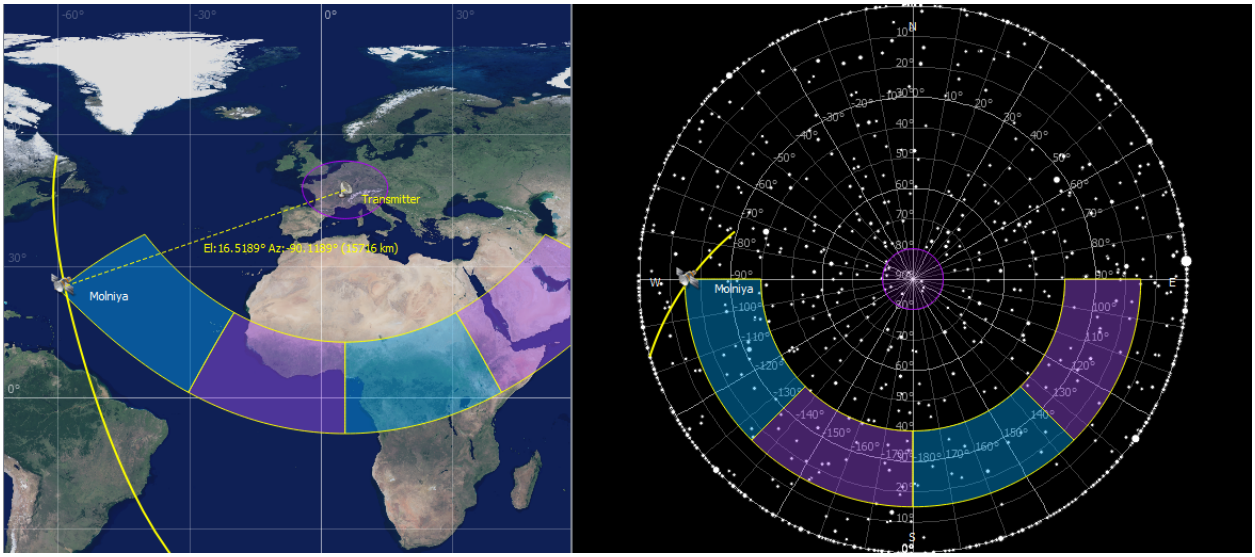


Fig. 2.16: Ground station sensor with azimuth-elevation constraints in SurfaceView and ZenithView

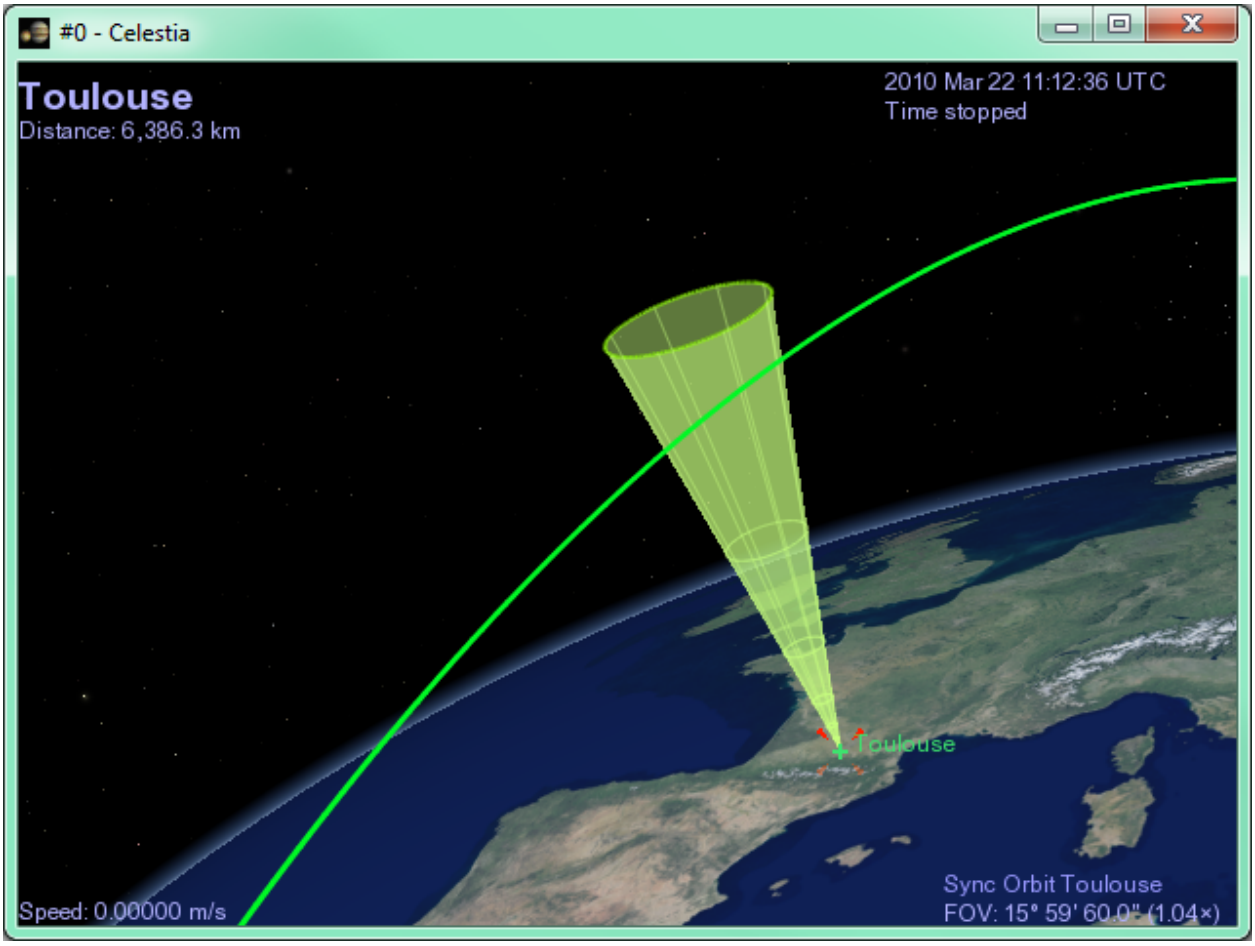


Fig. 2.17: Ground station sensor in Celestia

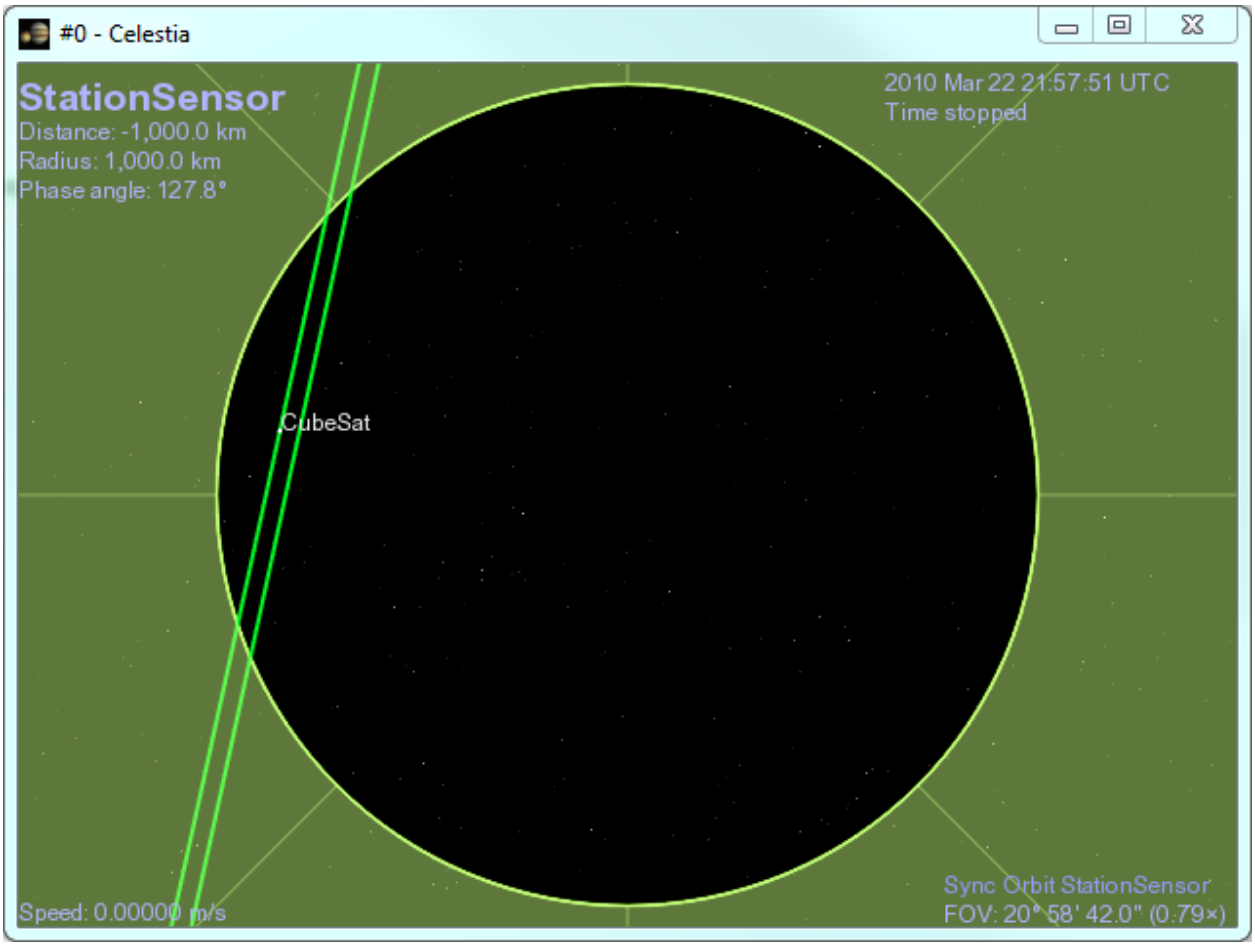


Fig. 2.18: Ground station sensor camera view in Celestia

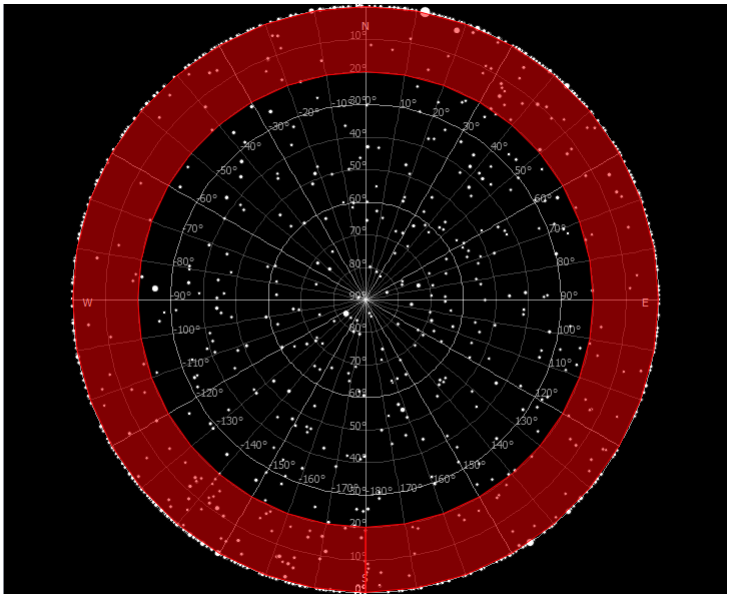


Fig. 2.19: Ring shaped mask in ZenithView

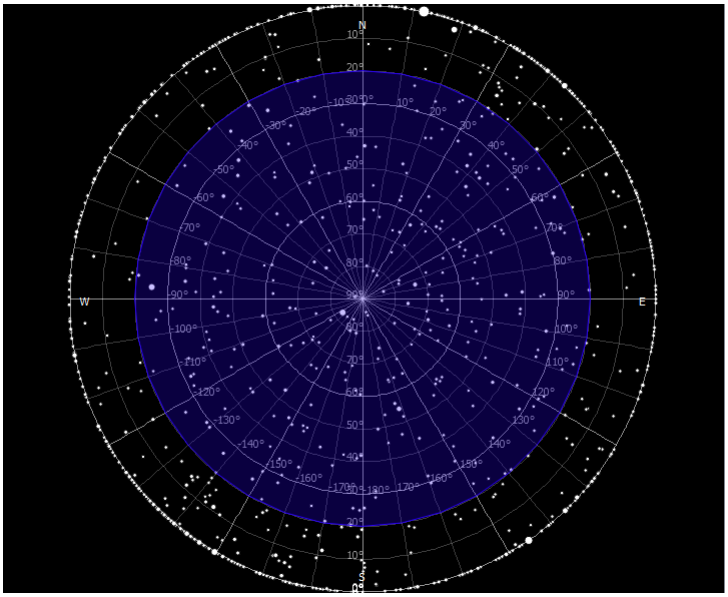


Fig. 2.20: Disc shaped mask in ZenithView

2.12.1 CIC/CCSDS mask file format

Masks are defined in CIC/CCSDS files in MPM format. For more information on the CIC/CCSDS file format, refer to the *CIC-CCSDS data files in VTS* chapter.

A CIC/CCSDS mask file must have the following characteristics:

- The `USER_DEFINED_PROTOCOL` must be `NONE`.
- The `USER_DEFINED_CONTENT` may be arbitrary.
- The `USER_DEFINED_SIZE` must be `2`.
- The `USER_DEFINED_TYPE` must be `REAL`.
- The `USER_DEFINED_UNIT` must be `[deg]`.

The data values in the file are coordinates in elevation/azimuth.

In masks files, the special data value `180 180` is interpreted as a separator and will create a new polygon with the following sequence of coordinates.

2.13 Scale factors in VTS

Scale factors in VTS can be applied to central bodies or satellites of a project, for Celestia and client applications that support scale factors. This feature allows easy visualization of both the satellite's attitude and its orbital position. All satellite components are affected by the satellite's scale factor. Note that when a central body's scale factor is different than 1, the computations for sensor-body intersection will be erroneous in clients affected by the scale factor.

The scale factor for central bodies and satellites can be dynamically adjusted in the View Properties tab of the Broker, using the **Body scale** and **Satellite scale** properties.

Scale factors also allow for easy visualization of motion for the whole solar system. The **Solar system scale** property in the *View Properties* tab of the Broker allows dynamically setting the scale factor of the whole solar system. Note

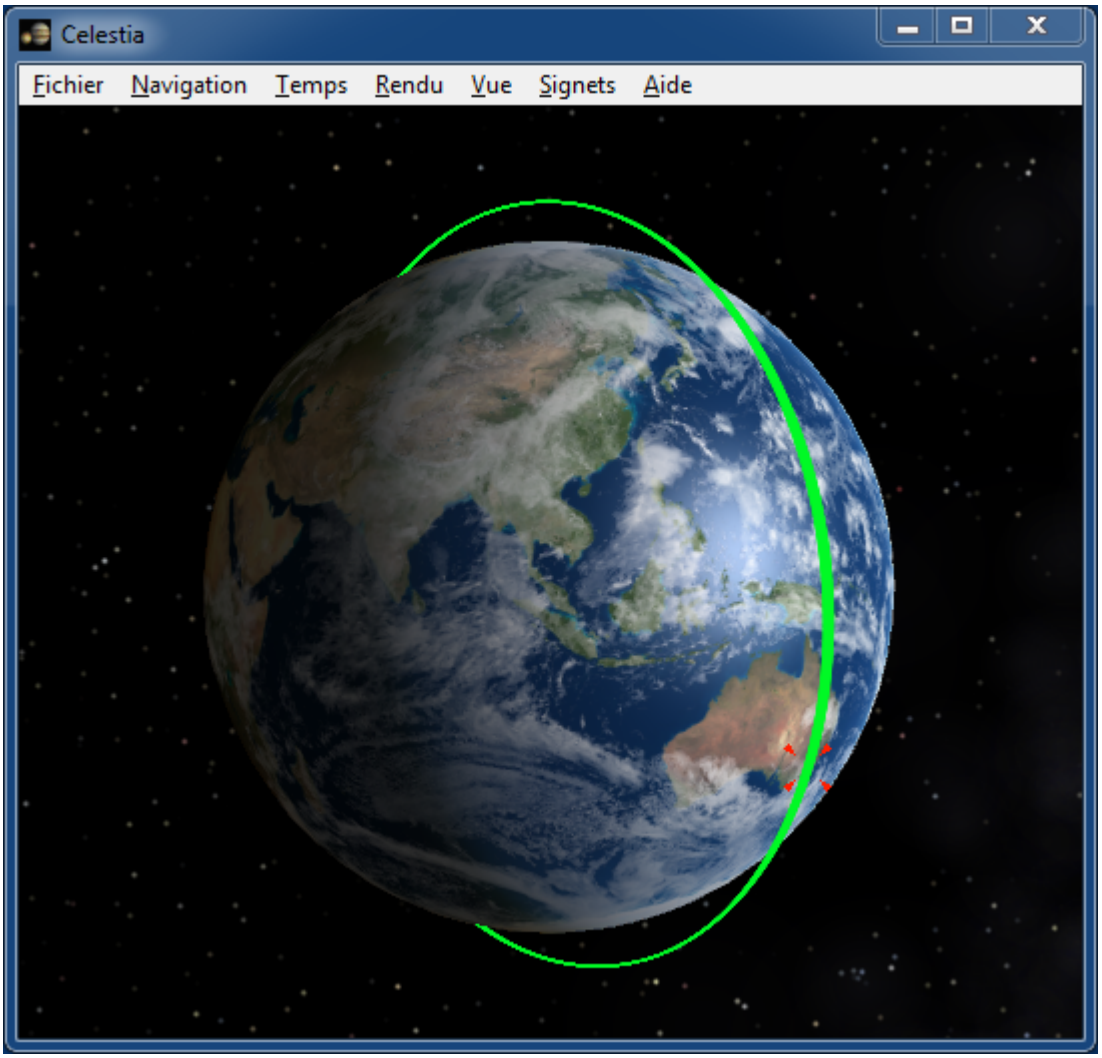


Fig. 2.21: Scale factor 1 for Earth and the project's satellite

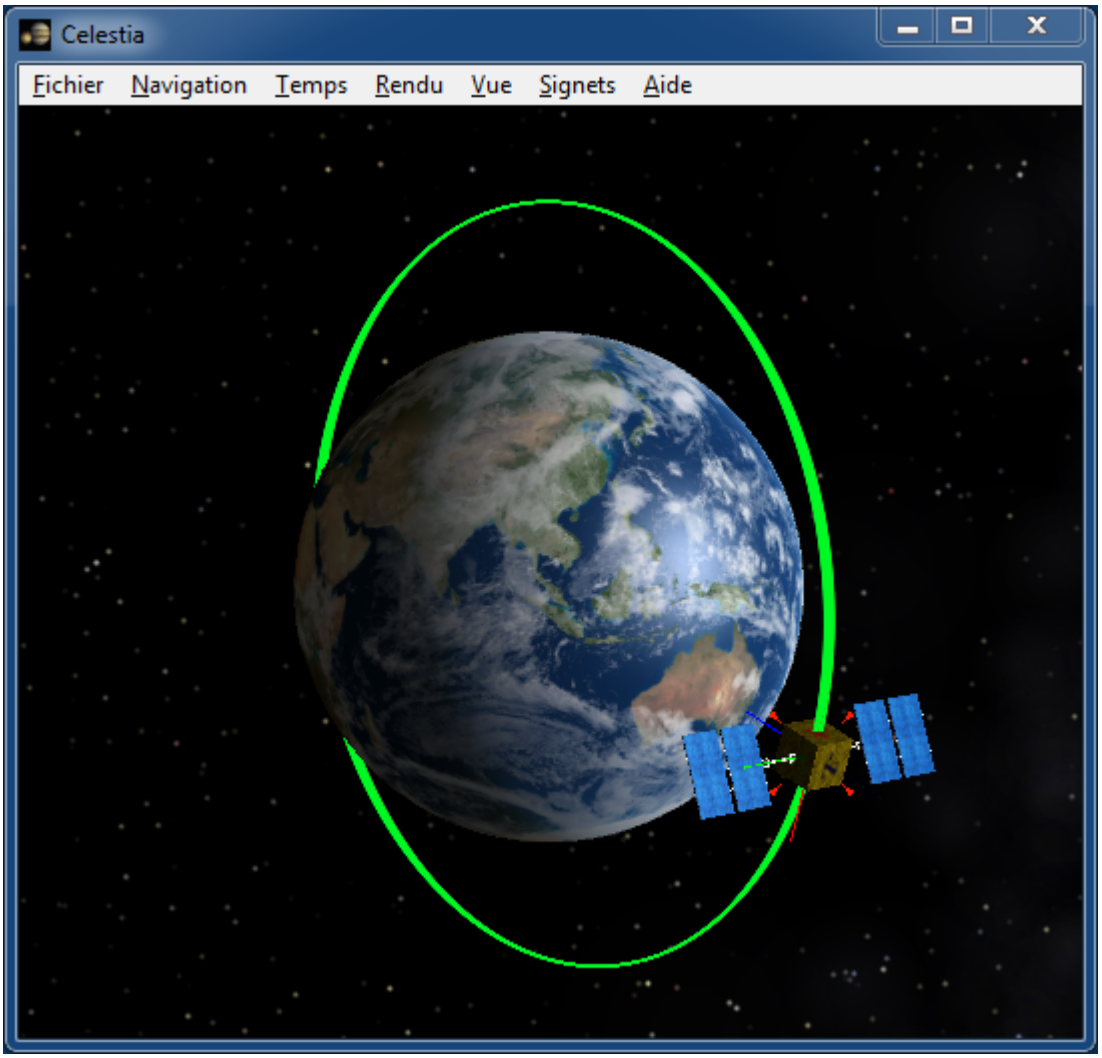


Fig. 2.22: Scale factor 0.75 for Earth and 500 000 for the project's satellite

that the Sun is not affected by this scale factor. Scale factors can be set for the whole solar system and for independent central bodies within in, e.g. to decrease the size of the gas giants.

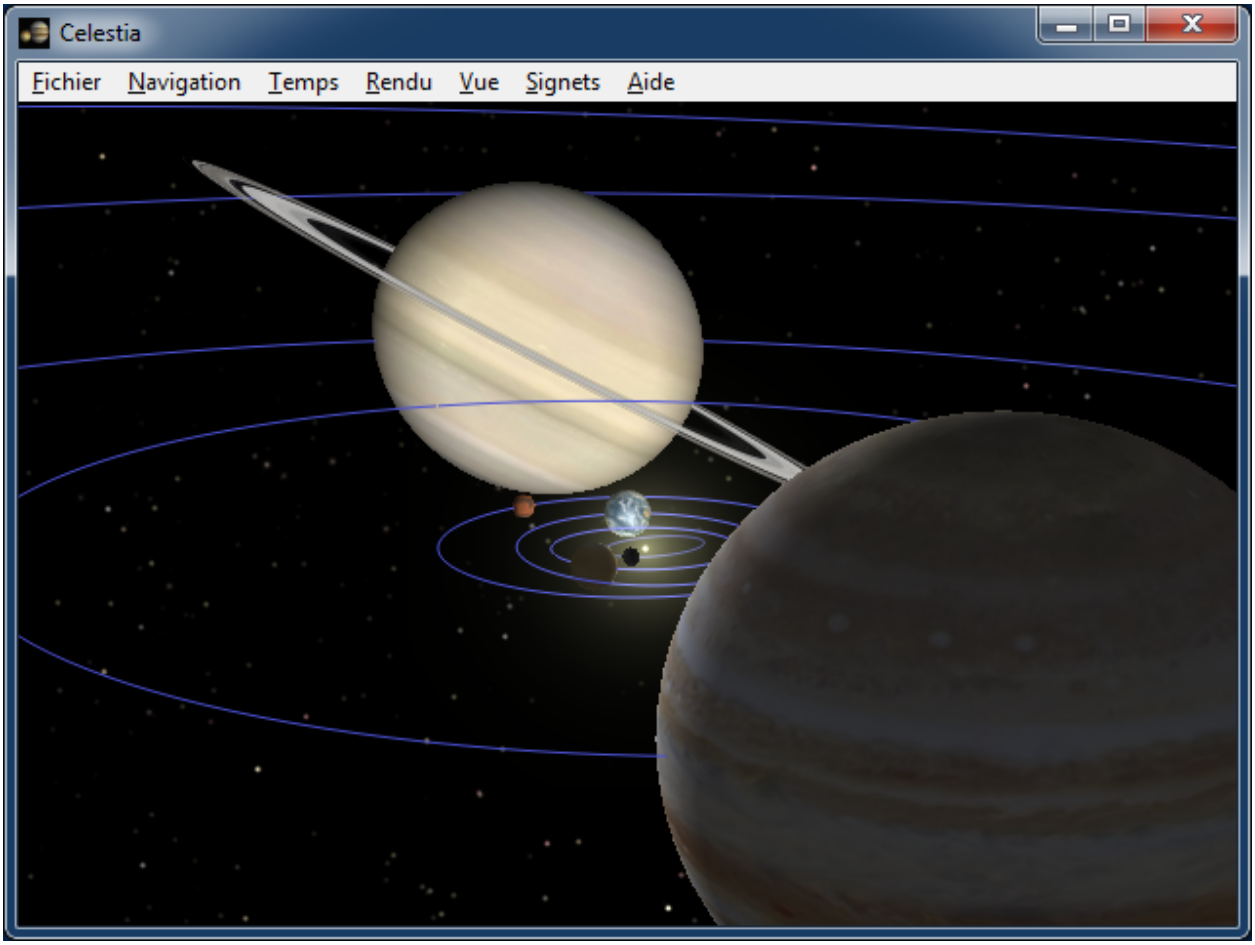


Fig. 2.23: Scaled solar system

The scale factor editor in the *view properties editor* allows editing the scale factor as follows:

- by directly entering a numerical value in the text field
- by moving the slider left/right to increase/decrease the factor
- by setting the factor to 1 using the `x1` button
- by setting the factor to 100 000 using the `x1e5` button (satellites only)

2.14 Clusters in VTS

In VTS, a cluster represents a population of objects. The objects aren't directly exposed to user in the configurator.

Each object consists of a name and an orbit file.

A cluster is attached to a central body which will be used as a reference frame to position the different objects.

2.15 Scenario in VTS

The VTS project scenario consists of a set of visualization states applied at precise instants of the visualization.

These states hold visualization properties specific to each client, such as position and orientation of the camera, visibility of satellite components, visibility of the equatorial grid, etc. Special behavior can then be triggered at precise dates through visualization states, e.g. momentarily switching to sensor view when an acquisition occurs on some instrument.

VTS offers two tools to edit a project scenario: the timeline and the view properties editor. In the VTS configuration utility, these tools are available in the **Scenario Editor** tab.

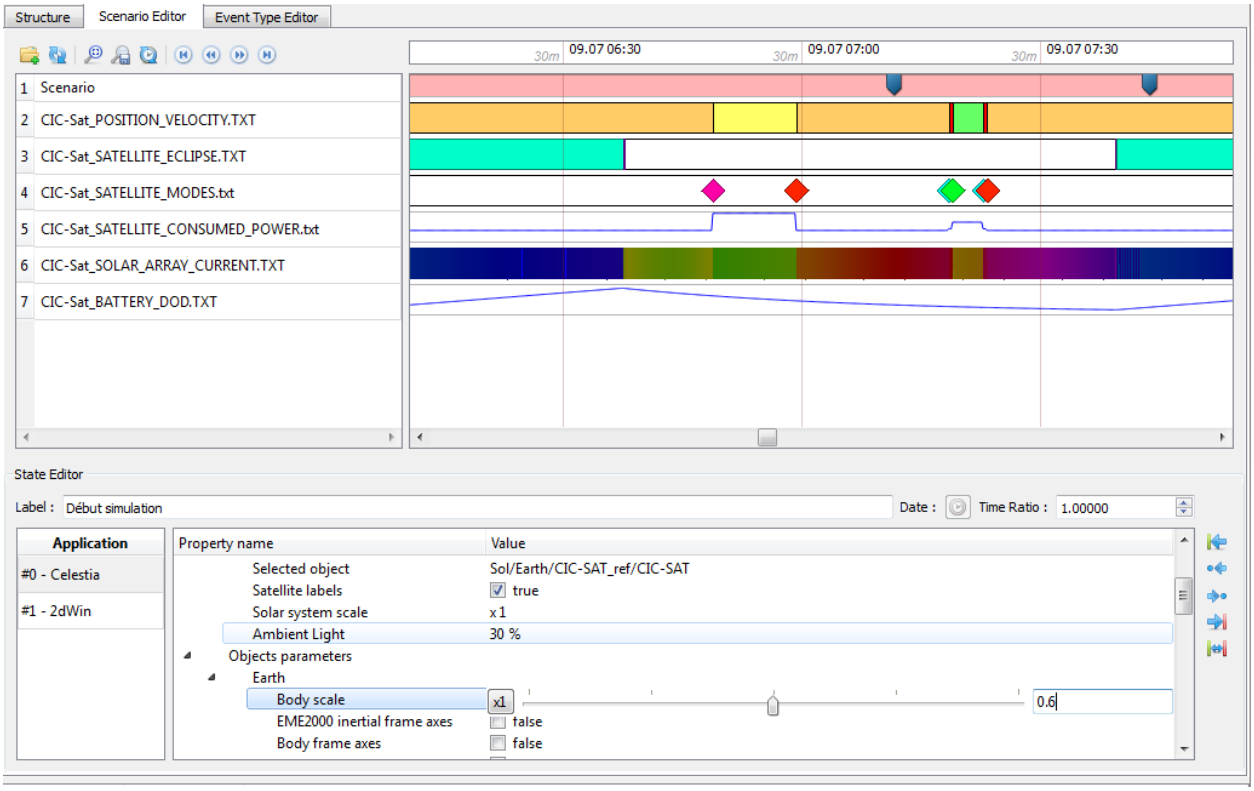


Fig. 2.24: Scenario editor in the VTS configuration utility

In the Broker, they are available in the **Timeline** and **View Properties** tab. Parts of the timeline, displaying dates and the project scenario, are always visible in the Broker, no matter which tab is currently selected.

2.15.1 Timeline

The timeline displays and allows selecting, creating and removing visualization states in the project scenario. It also displays the contents of the project's data files and event files.

In the VTS configuration utility, modifications to the project in the **Structure** tab are not reflected automatically in the timeline. The **Refresh** button must be clicked to update the view.

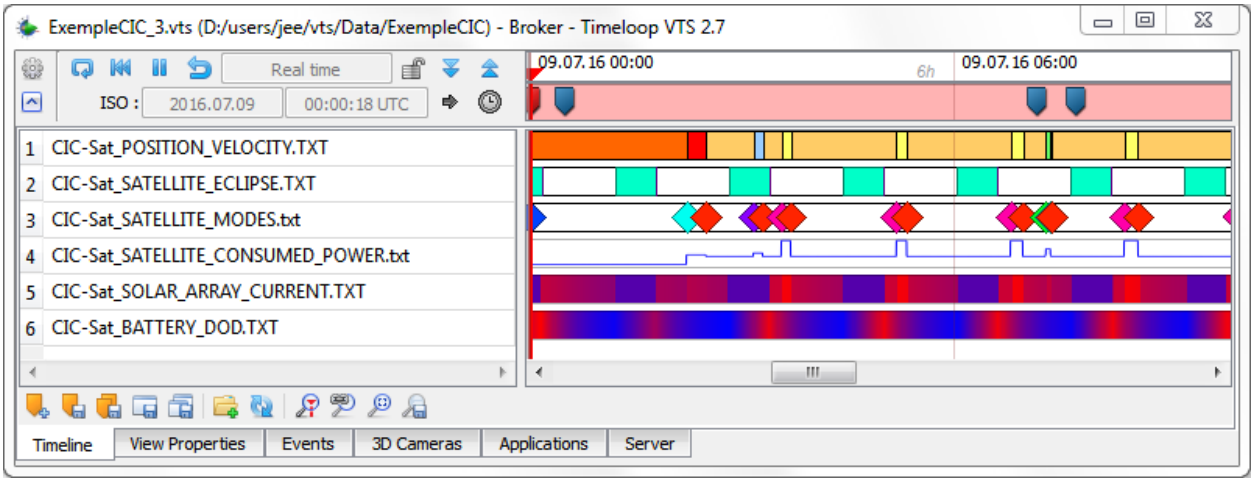


Fig. 2.25: Timeline in the Broker

Badges

Timeline files can be categorized using badges. A **badge** is a stamp identified by a *name* and a *color* that is displayed at the right of the file name. Badges are displayed in the Scenario Editor and the Broker.

To assign a new badge to a file, right-click on a file, select "*Assign*" and then select a badge from the list. New badges can be created using the "*New badge*" option. All files can have badges except for the special "Scenario" line.

Existing badges can be modified or removed from the project using the "*Badges*" option.

Except for the scenario, badges can also be used to filter specific files from the Timeline.

Badges are defined at project-level and saved in the VTS project file.

Time window

The timeline can be zoomed in/out using the mouse wheel, and panned by drag-and-dropping when the mouse cursor has the shape of a hand. This allows for great flexibility during visualization of a project, since the timeline can easily display an overview of the whole project scenario or a detailed view of a small time frame. The maximum zoom level is limited depending on the scenario duration.

Dates outside of the date range of the project are grayed out in the timeline.

By default, the timeline displays the whole project scenario when the Broker starts. By using the *Save View* button, users can save :

- **If the cursor is currently followed:**
  - Where it is locked (in % of the window)
  - The width of the window (in hours)
- **If it is not cursor-locked:**
  - The date corresponding to the left of the window
  - The width of the window (in hours)

Moreover, this button saves the current time format of the timeline.

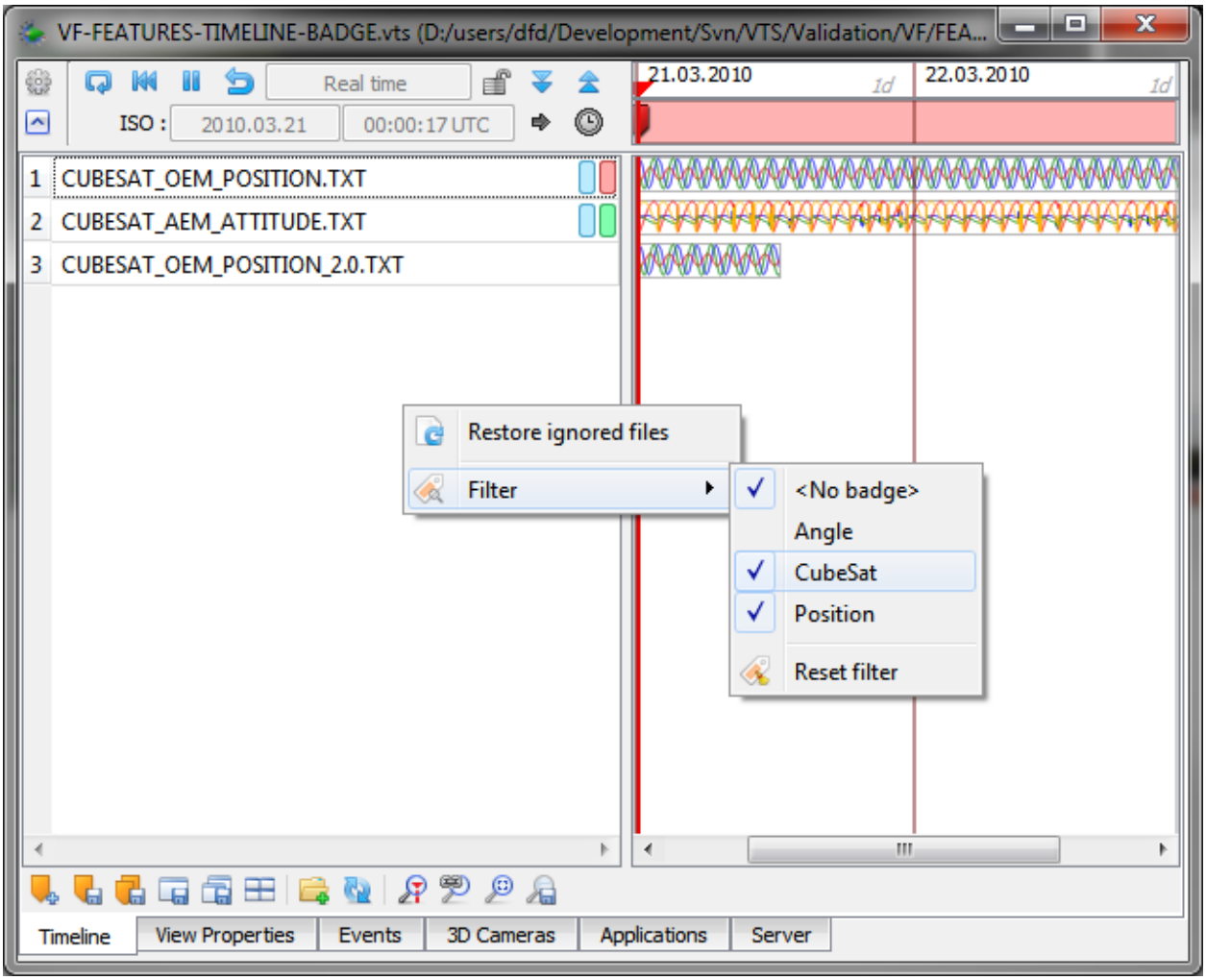


Fig. 2.26: Filter Timeline files using badges

Time cursor

The timeline's red time cursor indicates the current visualization date. It can be drag-and-dropped to modify the visualization date. While moving the time cursor, the visualization date is broadcasted synchronously to all client applications.

Double-clicking in the timeline moves the time cursor to the mouse location (and sets the visualization date accordingly). Hovering above the time cursor displays a tooltip indicating the visualization date in ISO format. When zooming the timeline, holding the *Ctrl* key locks the zoom on the time cursor.

The time cursor is not displayed in the VTS configuration utility, where time does not flow.

Project scenario

The project scenario is represented as a red line, labelled *Scenario* in the VTS configuration utility, and displayed in the compact mode GUI of the Broker.

Scenario states

Scenario states appear as flags in the project scenario line:

- a blue flag for a regular state not currently selected
- a red flag for the active state (the one displayed in the view properties editor)
- an green flag for the initial state of the visualization
- a grey flag for a disabled state (Broker only)

Also, an orange star appears on states which have been modified but not saved.

In the VTS configuration utility, clicking a state makes it become the active state. In the Broker, the active state is the first enabled state to the left of the time cursor.

When the time cursor moves past a new state in the Broker, it becomes active and its visualization properties are sent to the corresponding client applications. This allows, for example, precisely setting the camera at a user-defined location upon key events of the visualization (instrument acquisition, ground station communication, orbital maneuver, etc.). Note that a disabled state is never active: when disabling the active state, the first enabled previous state becomes active.

The initial state of the visualization is a fictional state mirroring all scenario states prior to the project's start date. It cannot be removed or disabled, and its date is automatically adjusted when the project's start date changes.

State time ratio

Each state can be associated with a timeratio. When playing the visualization, the current timeratio will be changed by the Broker if:

- The project is configured as using *Use scenario state time ratio* in the *Time & Options* panel of the Configurator
- The *Use Scenario State Time Ratio* button has not been unchecked in the Broker

Interacting with the scenario

Right-clicking the scenario displays a context menu with the following actions:

- The **Create state** entry creates a new scenario state at the selected location.

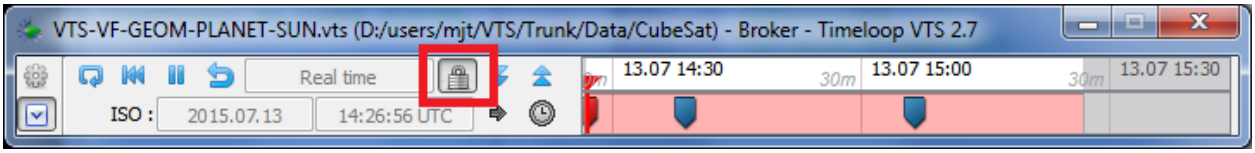


Fig. 2.27: State time ratio activation with the Broker

- The **Select current state** entry marks as active the first enabled state to the left of the selected location.
- The **Disable all states/Enable all states** entry disables/enables all states in the scenario (except for the initial state).
- The **Delete all states** entry removes all states in the scenario (except for the initial state)

The properties of a new state are copied from the previous state at the date of the new state.

Right-clicking a scenario state displays a context menu with the following actions:

- The **Go to state** entry sets the visualization date to the date of the selected state.
- The **Disable state/Enable state** entry disables/enables the selected state.
- The **Delete state** entry removes the selected state.

Right-clicking on the initial state of the scenario displays a specific context menu:

- The **Go to state** entry sets the visualization date to the date of the selected state.
- The **Create state** entry creates a new scenario state at the start of the scenario.

Mission events

Mission events attached to project entities are displayed in the first lines of the timeline. Each entity has its own line of events. Refer to the *Mission events in VTS* chapter for more information on events.

Graphical representation of events

The header of each line shows the name of the entity for the events on the line.

Each event is represented by its decoration, configured in the event types editor. Refer to the *Configuring event types* section of the *VTS configuration utility user manual* chapter for more information.

Each line can be resized by resizing its header's line number. Lines can be reordered by drag-and-dropping their headers' line numbers.

Interacting with events

Right-clicking on an event displays a context menu with the following actions:

- The **Go to event** entry sets the visualization date to the date of the selected event (Broker only).
- The **Select state for event** entry marks as active the first enabled state to the left of the selected event.
- The **Create state at event** entry creates a new scenario state at the date of the selected event.

Project scripts

VTS scripts used by the project are displayed in the timeline. Each script has its own line. Refer to the *Scripts and macros in VTS* chapter for more information on script files.

Graphical representation of scripts

The header of each line shows the name of the script.

Script commands are displayed as red diamonds at the date at which they occur.

Each line can be resized by resizing its header's line number. Lines can be reordered by drag-and-dropping their headers' line numbers.

Interacting with scripts

The following actions are available for script headers:

- Double-clicking the header sets the zoom level and pans the timeline so that the line's width fits in.
- The **File properties...** button opens a pop-up window with information on the script file.
- The **Remove file** button removes the script from both the timeline and the project.

Right-clicking on a script command displays a context menu with the following actions:

- The **Go to command** entry sets the visualization date to the date of the selected command (Broker only).
- The **Select state for command** entry marks as active the first enabled state to the left of the selected command.
- The **Create state at command** entry creates a new scenario state at the date of the selected command.

Project data files

Data files used by the project (position and orientation of satellites, sensors, etc.) are displayed in the timeline. Each file has its own line.

Graphical representation of files

The header of each line shows the name of the file.

The following display modes are available:

- **Normal** mode: Each value is displayed as a rectangle extending from the date of the value to the date of the next value. Rectangles of identical color indicate identical value. In the screenshot below, this display mode is used for line 2.
- **Block** mode: The whole file is displayed as a single rectangle. This display mode is used for files with more than 300 lines or bigger than 500Kb, so as not to overload the timeline. When the file contents have not been read, the block is colored in gray and no tooltip is available. Otherwise, the block is colored and tooltips are available. In the screenshot below, this display mode is used for line 3.
- **Event** mode: Each value is displayed as a diamond at the date of the value. Diamonds of identical color indicate identical value. In the screenshot below, this display mode is used for line 4.

- **Interval** mode: Identical to **Normal** mode, except that "neutral" values (0 for real or integer data) are not displayed. This mode offers better contrast to visualize the contents of files with on/off status, such as in/out of eclipse status for a satellite, etc. In the screenshot below, this display mode is used for line 5.
- **Gradient** mode: The whole file is displayed as a single rectangle filled with a color gradient indicating the value variations in the file. This mode offers better visualization of the evolution of values in a file, but may not be suited to files of dimension greater than 1. In the screenshot below, this display mode is used for line 6.
- **Color** mode: Identical to **Normal** mode, except that the color of each rectangle is derived by interpreting the corresponding value as a color. This mode allows displaying the contents of color files, and is only available for data which may be interpreted as a color, i.e. files of real or integer data of dimension 3 or 4. In the screenshot below, this display mode is used for line 7.
- **Graph** mode: all values are displayed as a graph. This is a preview of the content of the file and may not replace a proper plotter software such as PrestoPlot. In the screenshot below, this display mode is used for line 8.

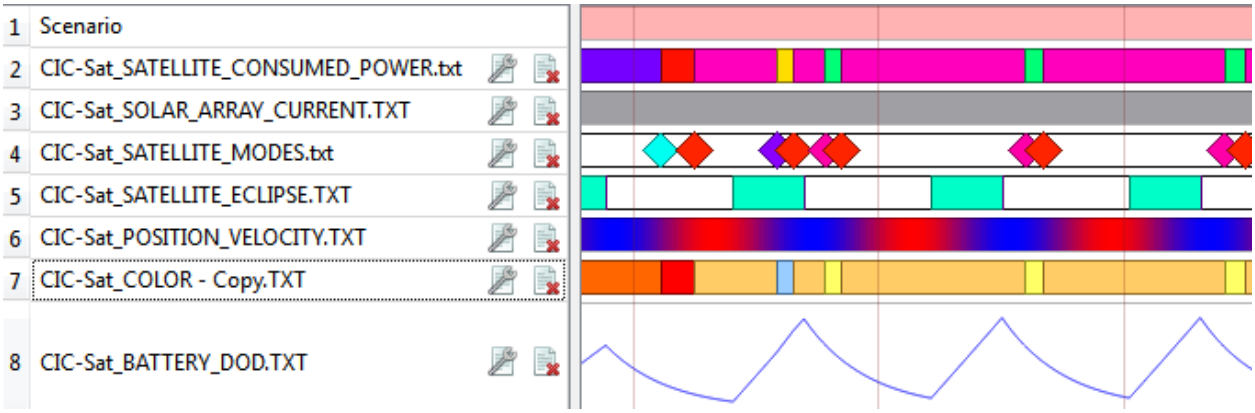


Fig. 2.28: Display modes available for data files in the Timeline

In order to avoid long loading times, files bigger than 500Kb are only partially loaded. The full contents will however be read if the user selects a mode other than **Block** for the file. When hovered, files for which the full contents have been read display a tooltip containing the value at the current location of the mouse.

Each line can be resized by resizing its header's line number. Lines can be reordered by drag-and-dropping their headers' line numbers.

Color files associated with OEM position files are used to color the line of their position file.

User-selected display modes are saved in the VTS project file and restored upon loading of the project in the VTS configuration utility or the Broker. Be aware that this might result in longer loading times if a mode other than **Block** is selected for files of significant size.

Interacting with files

The following actions are available for file headers:

- Double-clicking the header sets the zoom level and pans the timeline so that the line's width fits in.
- **Right-clicking the header displays a context menu with the following entries:**
  - The **File properties...** entry opens a pop-up window with information on the file and parameters controlling its appearance in the timeline.
  - The **Edit file...** entry opens the file in a text editor.

  - The **Ignore file** or **Remove file** entry removes the file from the timeline. Files referenced in the project structure are not removed from the project, they are merely hidden in the timeline and will remain so until they have been added back or restored.

- Right-clicking a blank zone below the file header list displays a context menu with the following entries:
- **Restore ignored files**: Restore the previously ignored files.

The file properties pop-up, apart from displaying information on the file, offers the following actions:

- The **Edit file…** button opens the file in a text editor.
- The **Display mode** drop-down list allows selecting the display mode for the file.
- The **Merge identical values** option enables/disables the merging of consecutive identical values in the file into a single value item (rectangle or diamond). This is on by default to improve the clarity of the timeline's contents.
- The **Color overlay** option enables/disables the color file overlay for OEM position files.

Note that the number of lines is not displayed in the *Properties* tab for partially loaded files.

Right-clicking on values in the file's line displays a context menu with the following actions (not available in **Block** and **Gradient** modes):

- The **Go to event/Go to interval start/Go to interval end** entry sets the visualization date to the date/start date/end date of the selected diamond/rectangle (Broker only).
- The **Select state for event/Select state for interval start/Select state for interval end** entry marks as active the first enabled state to the left of the selected diamond/rectangle start/end.
- The **Create state at event/Create state at interval start/Create state at interval end** entry creates a new scenario state at the date of the selected diamond/rectangle start/end.

Timeline toolbar

The timeline's toolbar offers the following actions:

- The **Add files…** button opens a browser to add one or several files (data files or scripts) to the timeline. These files will have to be saved in the project folder if they are located outside of it. Files not part of the structure added this way will be saved and opened on future loads of the project. Files may also be added by drag-and-dropping them onto the timeline.
- The **Refresh timeline** button updates the timeline. The project dates, files (events and data) and the contents of these files are updated. The timeline may only be updated if the project is in a valid state.
- The **Center cursor** button centers the timeline's view on the time cursor (Broker only).
- The **Follow cursor** button automatically locks the view on the time cursor as it moves with the visualization date (Broker only).
- The **Reset view** button zooms in/out the timeline and pans it so that the whole project is displayed. Unless the user interacts with the timeline, upon change of the project limits the timeline will be adjusted to keep the whole project displayed.
- The **Save view** button saves the current view settings so that they can be restored upon next load of the project. See the *Time window* section above for more information on the saved and restored view settings.
- The **Change date format** button changes the date format displayed in the timeline (VTS configuration utility only).
- The **Arrange all client application windows** button shows a visual tool helping arranging client application windows for the current state.

In the VTS configuration utility, the following buttons allow navigation between scenario states:

- The **Initial state** button marks the project scenario's initial state as active.
- The **Previous state** button marks the active state's previous state as active.
- The **Next state** button marks the active state's next state as active.
- The **Final state** button marks the project scenario's last state as active.

In the Broker, the following buttons allow saving view properties of client applications into scenario states:

- The **Create state** buttons creates a new scenario state at the current visualization date and fills it with view properties gathered from all client applications. The new state is saved (others are not).
- The **Save current state only** (Ctrl + S) button updates the currently active scenario state with view properties gathered from all client applications. The active state is saved (others are not).
- The **Save all states button** (Ctrl + Shift + S) button updates the currently active scenario state with view properties gathered from all client applications, and saves all scenario states.
- The **Save window positions for current state only** button updates the currently active scenario state with the window position of all client applications. The active state is saved (others are not).
- The **Save window positions for whole scenario** button updates all scenario states with the window position of all client applications. All scenario states are saved.

2.15.2 View properties editor

The view properties editor allows editing the view properties of client applications attached to each scenario state.

Scenario state properties

The **Label** field displays the label of the current scenario state. This label can be modified and is displayed in the state tooltip.

The **Date** button opens a dialog to view and edit the date of the current scenario date. However, it is not possible to modify the date of the fictional initial scenario state.

The **Filter** button (*Ctrl+F*) shows an editor allowing filtering state properties using case-insensitive regexp patterns .

The **Application** list displays a list of all client applications. Selecting a client application in the list displays its **view properties** in the tree hierarchy to the right.

There are two types of **view properties**: application parameters, which describe global properties specific to the application; and objects parameters, which describe display properties of project entities (central bodies, satellites, sensors, etc.) for the application.

Each property has a label describing its purpose. Clicking or double-clicking a property's value to the right of its label opens an editor to modify the value.

- Properties with bold labels have had their values modified but not saved. Scenario states for which at least one of their properties has had its value modified appear with an orange star in the timeline.
- Properties with italic labels have a unique value for the whole scenario. Modifying their value automatically affects all scenario states.

Each property can be reset to default by right-click. In the Broker, this action can lead to a message to the concerned application.

- ***Boolean***

  Editor

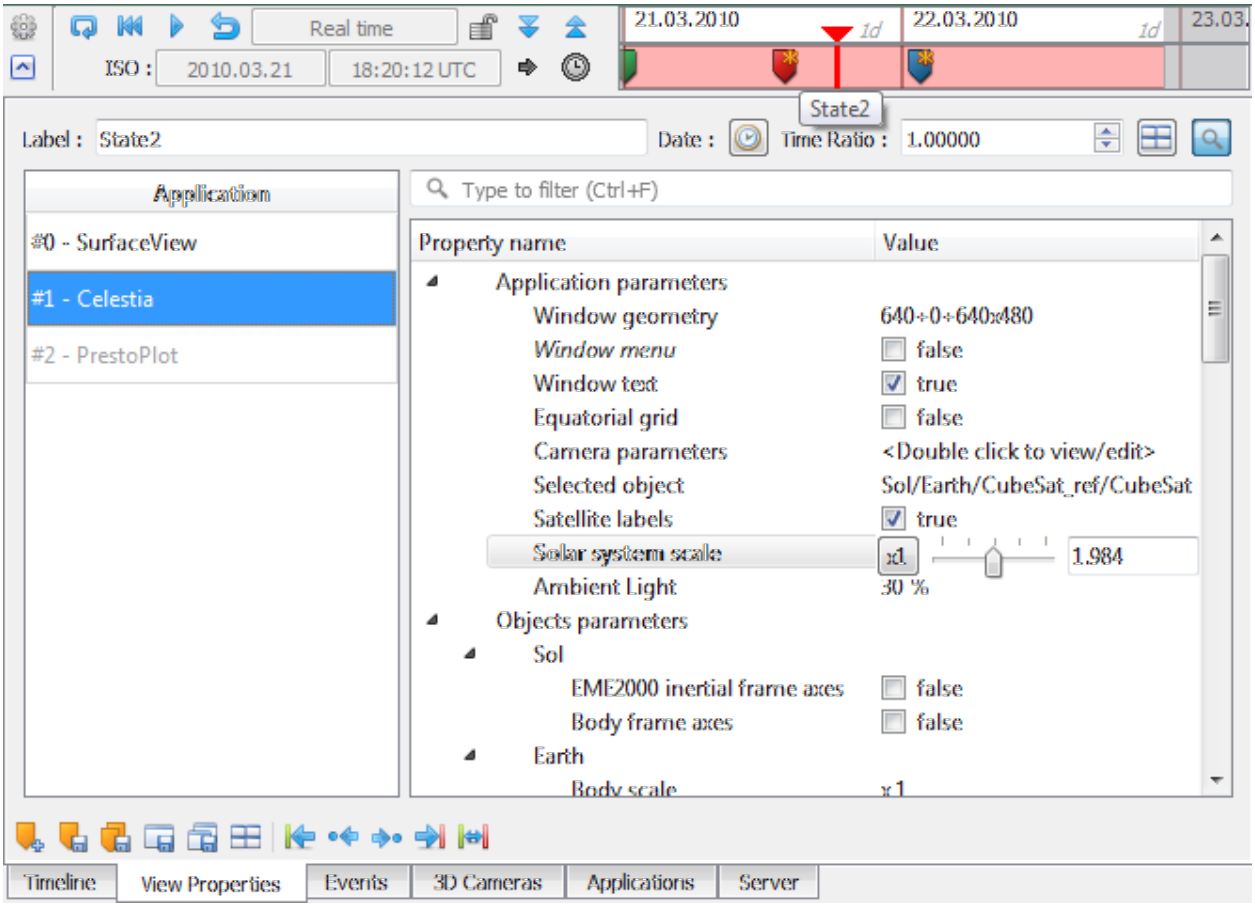


Fig. 2.29: View properties editor in the Broker

Description

Editor for a boolean property (e.g.: equatorial grid visibility in Celestia).
Clicking the checkbox toggles the property.

- ***Integer or real number***

  Editor

  Description

  Editor for an integer or real number property.
  The integer or real value can be entered manually or increased/decreased using the buttons.

- ***Character string***

  Editor

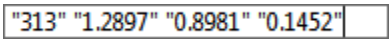
  

  Description

  Editor for a string property (e.g.: selected object in Celestia).
  The string must be entered manually.

- ***Character string list***

  Editor

  Description

  Editor for a string list property (e.g.: visible event types in SurfaceView).
  The string list must be entered manually. Syntax rules of the VTS synchronization protocol apply (except the newline rule). Refer to the *Message syntax* section of the *Synchronization protocol for VTS clients* chapter for a list of these rules.

- ***Scale factor***

  Editor

  Description

  Editor for a scale factor property (e.g.: satellite scale factor in Celestia).
  The scale factor value can be adjusted by dragging the slider left or right, or entered manually, or reset using the buttons.

- ***Range***

  Editor

  Description

  Editor for a range property (e.g.: ambient light level in Celestia).
  The range value can be set by moving the slider, or entered manually, or reset using the button.

- ***Rectangle***

  Editor

Description

Editor for a rectangle property (e.g.: window position and size).

The X, Y coordinates and width, height of the rectangle can each be entered manually or increased/decreased using the buttons.

- ***Camera***

  Editor

  Description

  Editor for camera parameters (e.g.: point of view in Celestia).

  – The camera being configured is selected in the **View** drop-down list. The Celestia view layout can be controlled using the **Change view layout...** button.

  – The reference frame the camera is attached to is defined by an object selected in the **Reference** drop-down list, and a frame selected in the **Frame** drop-down list.

  – The position of the camera in its reference frame can be entered in the **Position** group, in kilometers.

  – The orientation of the camera in its reference frame can be entered in the **Orientation** group, either as a quaternion, as Euler angles (in degrees), as an axis and angle (in degrees), or as a direction and up vector.

  – The field of view of the camera can be entered in the **FOV** field, in degrees.

- ***File***

  Editor

  Description

  Editor for file properties.

  The file path (either relative or absolute) can be entered in the text field or selected through a file browser using the button.

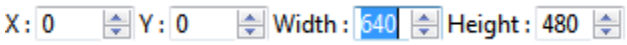
- ***Time Window***

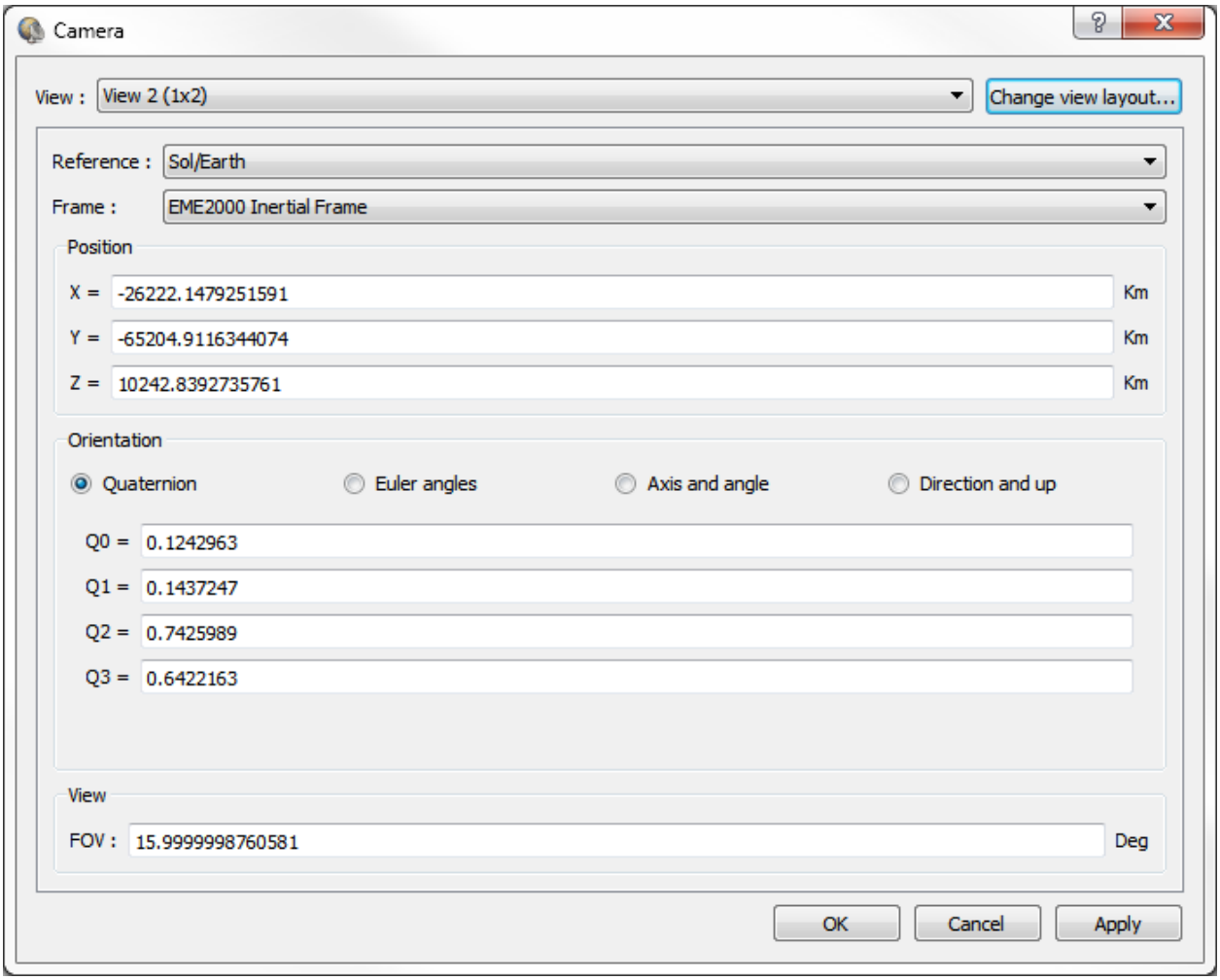
  Editor

  Description

  Editor for a time window. It represents the time distribution before and after a dated event. Durations are in hours.

Further documentation on the properties of standard client applications can be found in the *Synchronization protocol for VTS clients* chapter. In particular, the *Messages received by Celestia* and *Messages received by SurfaceView* sections describe properties of the Celestia and SurfaceView client applications.




Camera
View :
View 2 (1x2)
Change view layout...
Reference :
Sol/Earth
Frame :
EME2000 Inertial Frame
Position
X =
-26222.1479251591
Km
Y =
-65204.9116344074
Km
Z =
10242.8392735761
Km
Orientation
Quaternion
Euler angles
Axis and angle
Direction and up
Q0 =
0.1242963
Q1 =
0.1437247
Q2 =
0.7425989
Q3 =
0.6422163
View
FOV :
15.9999998760581
Deg
OK
Cancel
Apply


TM/HKTMR_20120222.tm
...


1.34
0.66
Total :
2

View properties editor toolbar

The view properties editor offers buttons to propagate view properties between scenario states. Upon selection of one or several properties in the tree hierarchy, these buttons allow copying the selected properties for the same application in one or several other scenario states:

- The **Propagate selection to all previous states** button copies the selected view properties to all previous scenario states.
- The **Propagate selection to previous state** button copies the selected view properties to the previous scenario state.
- The **Propagate selection to next state** button copies the selected view properties to the next scenario state.
- The **Propagate selection to all next states** button copies the selected view properties to all next scenario states.
- The **Propagate selection to all states** button copies the selected view properties to all scenario states.

These actions allow easily setting a property value for multiple scenario states. Several properties can be selected by holding down the *Ctrl* or *Shift* keys while selecting the properties to propagate. If a tree node is selected, all properties beneath it will be propagated.

In the Broker, the view properties editor also offers the same buttons as those available to save states in the timeline.

2.15.3 Window geometry manager

For information, see *Window Geometry Manager*

2.16 Window Geometry Manager

VTS client applications can be arranged via the Window Geometry Manager. This tool is available by clicking on the *Manage window geometry* button ( ) from the **Scenario Editor** tab or from the **VTS Broker**.

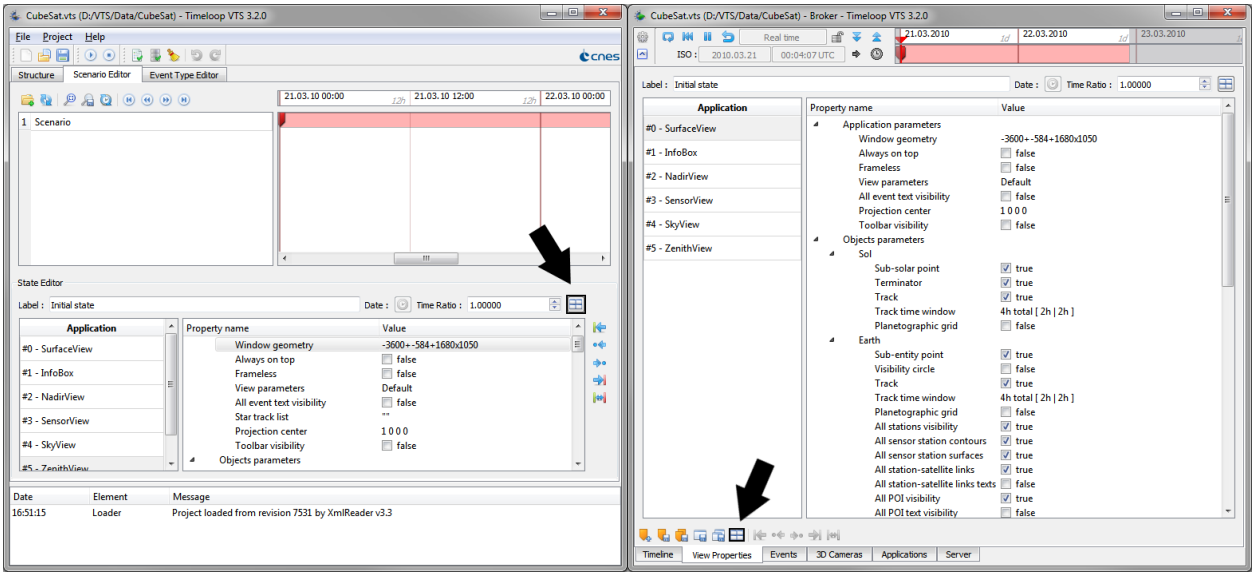


Fig. 2.30: Window Geometry Manager button in the Scenario Editor tab (left) and the Broker (right)

The Window Geometry Manager is divided into four sections :

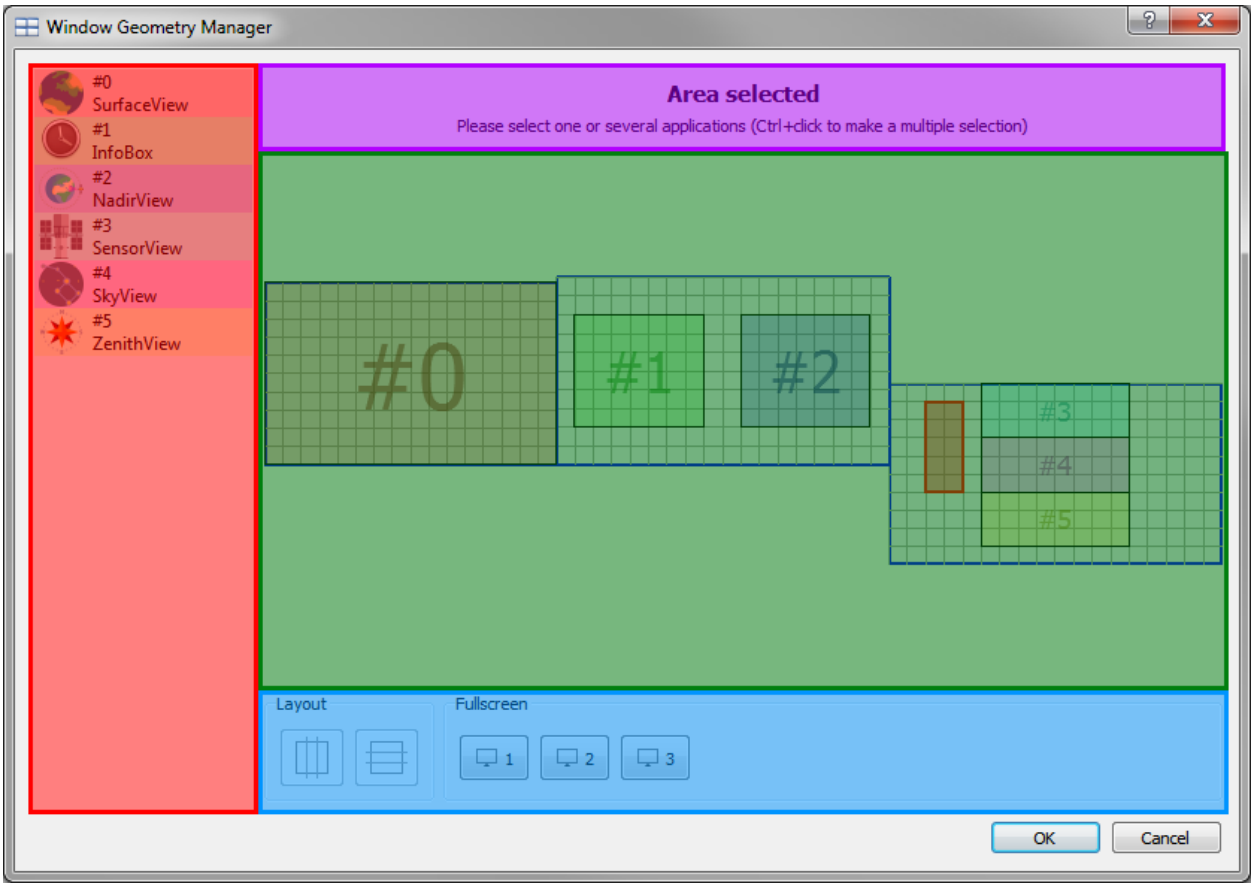


Fig. 2.31: Window Geometry Manager interface

- The left section (red) contains the list of all VTS client application defined in the VTS project or used during the visualization as dynamic application.
- The top-right section (violet) is used to display information and help about the tool.
- The middle-right section (green) shows the system's screen configuration as detected by VTS and the position of each client application window. A grid is displayed to help arrange the position of the windows. The resolution of the grid can be configured via VTS's **Settings dialog**.
- The bottom-right section (blue) contains options to arrange client application windows automatically. There are as many **Fullscreen** buttons as screens detected and they are ordered from the top-left screen to the bottom-right screen.

2.16.1 Change the order of stacked windows

The order of stacked window can be change by dragging up and down application items in the list in the left section. The feature can fail if a client application doesn't implement the "CMD SERVICE ActivateWindow" protocol command.

Clicking on an application item will also raise or activate the corresponding application.

2.16.2 Contextual menu actions

A right click on the left section list will show the available actions :

- *Activate all windows*: raise and activate all applications (needs *"ActivateWindow"* protocol command implementation)
- *Activate selected windows*: raise and activate selected application (needs *"ActivateWindow"* protocol command implementation)
- *Toggle always on top*: Change the state property for the always on top flag (needs the *"AlwaysOnTop"* application property implementation)
- *Toggle frameless*: Change the state property for the always on top flag (needs the *"Frameless"* application property implementation)

The **order of stacked windows** can be change by dragging up and down application items in the list in the left section. The feature can fail if a client application doesn't implement the *"ActivateWindow" protocol command*.

2.16.3 Arrange client applications windows

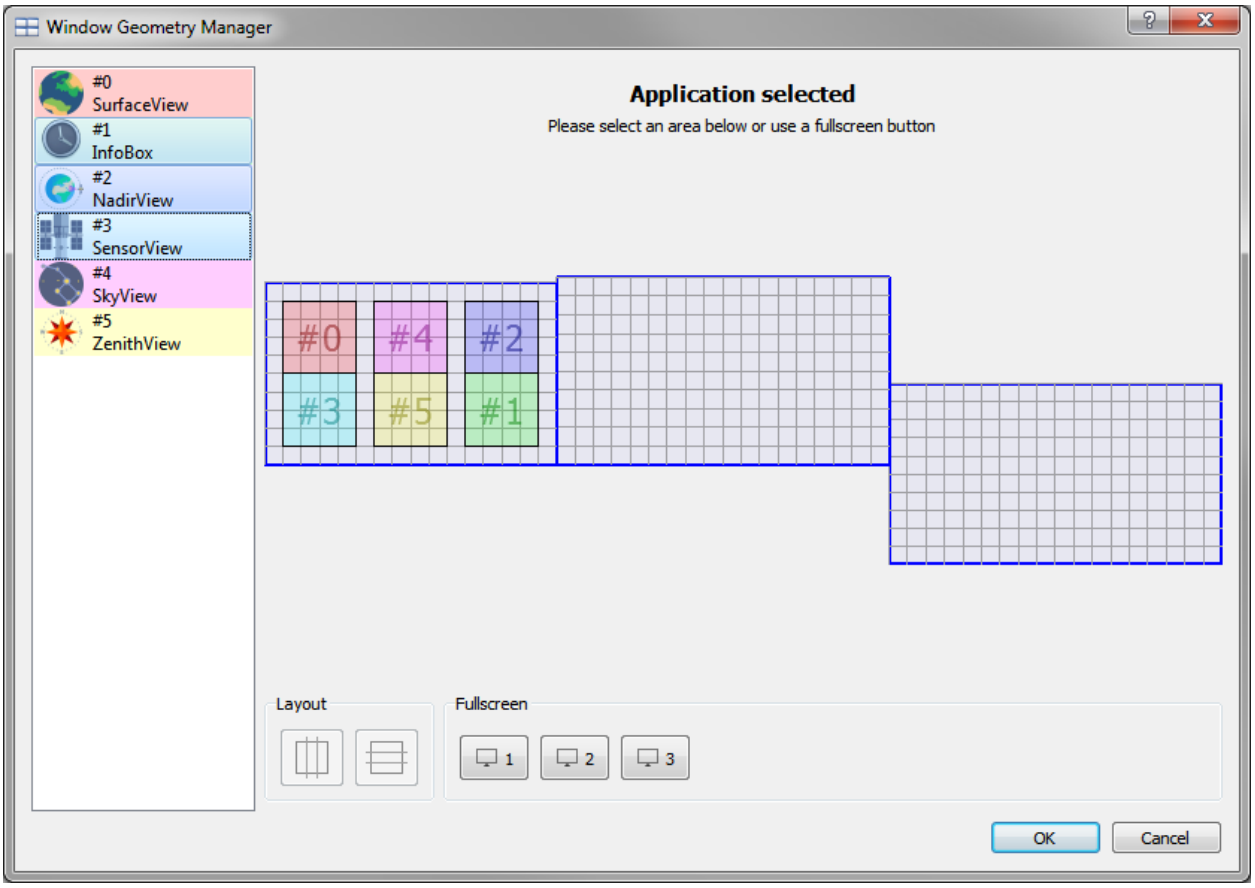
In order to organize the visualization, it requires two actions:

- Area selection: select an area in one of the screen grids. A preview of the selected area will be briefly displayed in the screen. The entire screen can be selected using the corresponding **Fullscreen** button.
- Application selection: select an application from the list. Multiple applications can be selected using `Ctrl` key while clicking.

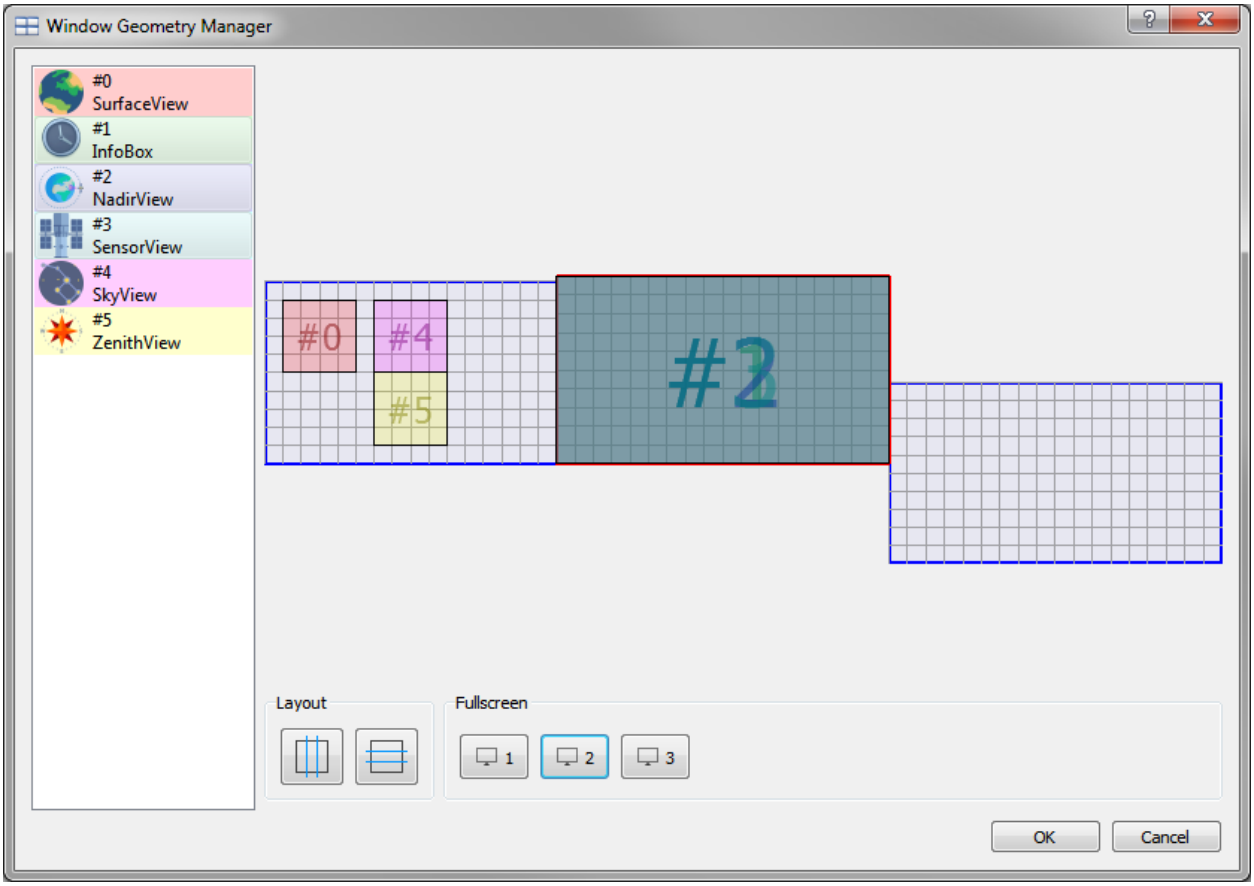
When both and **area** and an **one or more applications** are selected, the geometry of the selected application(s) will be modified to cover the selected area.

If several applications are selected they will be overlapped in the order of the selection. They can also be arranged in an horizontal or vertical layout proportionnaly using the **Layout** buttons. Clicking one more time will remove the layout (windows will return to the overlapped layout).

An example is shown below:


Window Geometry Manager
#0
SurfaceView
#1
InfoBox
#2
NadirView
#3
SensorView
#4
SkyView
#5
ZenithView
Application selected
Please select an area below or use a fullscreen button
#0
#4
#2
#3
#5
#1
Layout
Fullscreen
1
2
3
OK
Cancel

Click the **Fullscreen** "2" button and then select applications #1, #2 and #3. Applications windows are shown overlapped on screen 2.



Click vertical layout button to arrange windows vertically.

Click horizontal layout button to arrange windows horizontally.

Click cascade layout button to arrange windows horizontally.

Note: The "cascade layout", not visible in the above screenshots, will try to arrange windows as a cascade. It might fail if some applications don't implement the *"ActivateWindow" protocol command*.

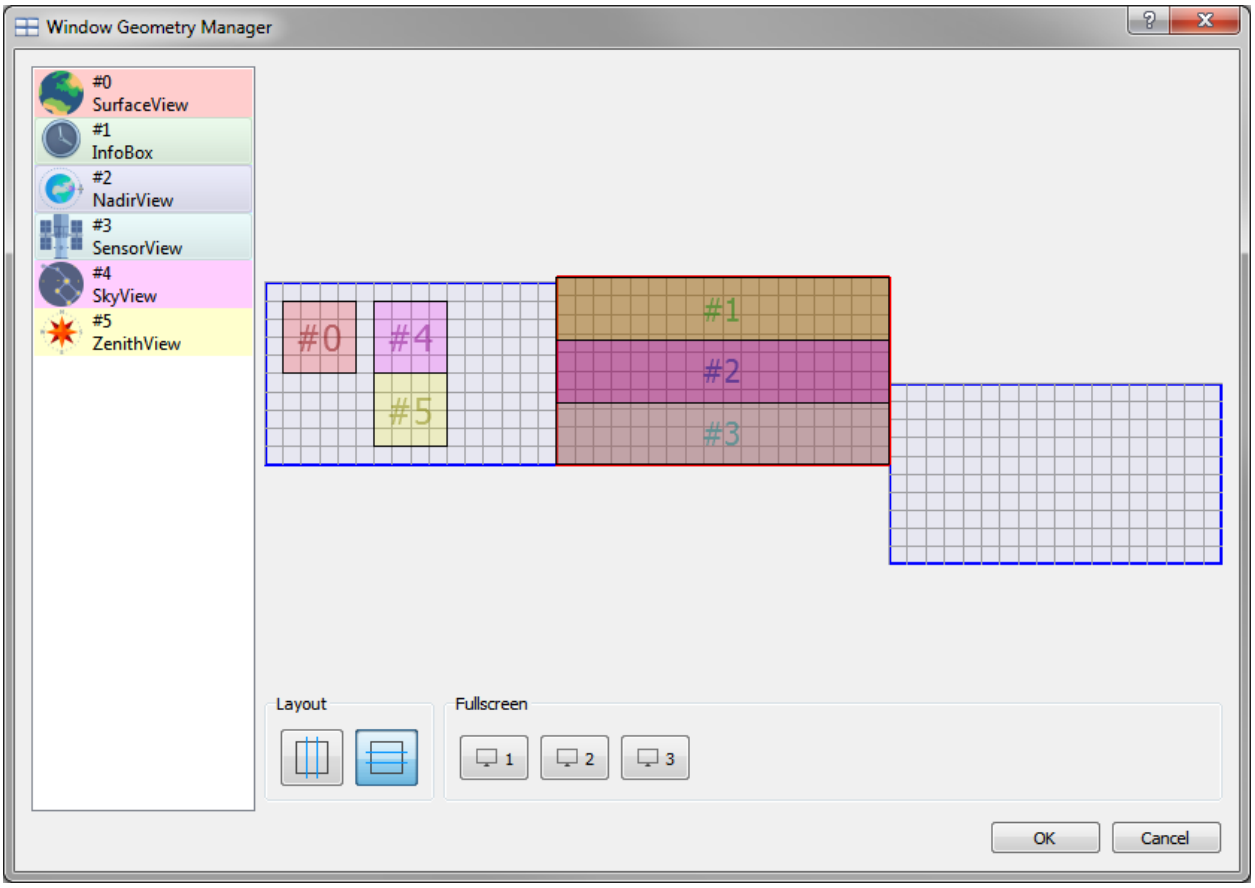
Click **OK** to save or **Cancel** to revert changes and close the Window Geometry Manager.

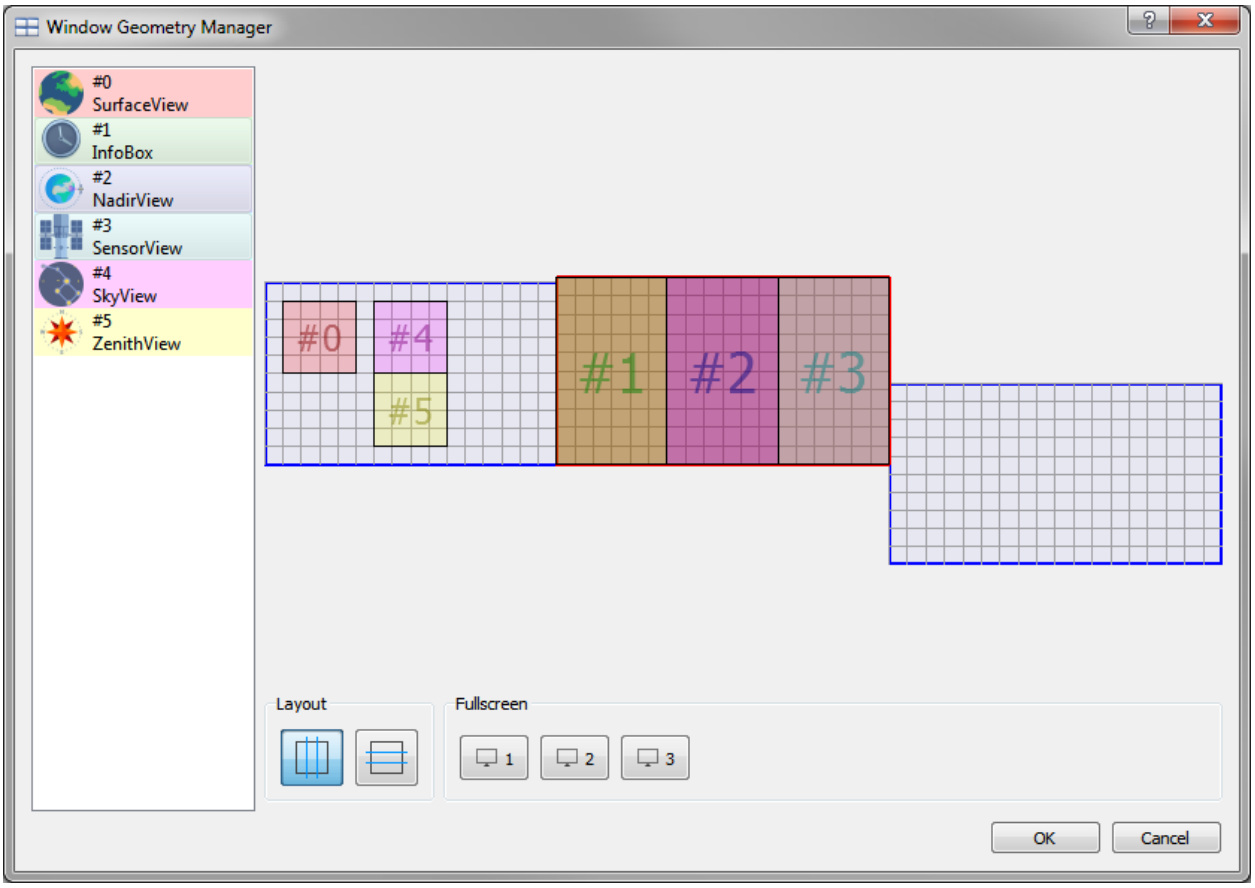
2.17 Mission events in VTS

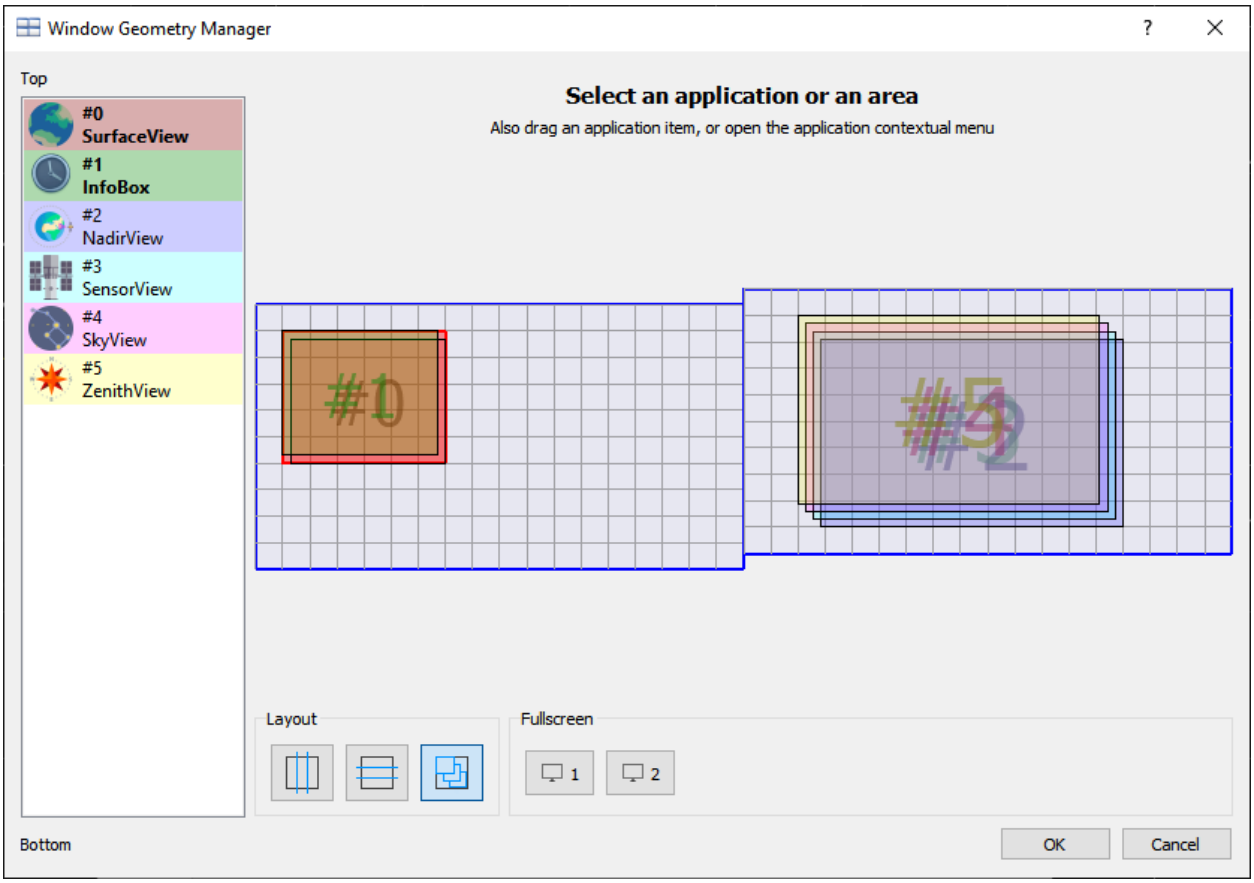
Mission events in VTS are timestampped punctual events attached to visualization entities. Currently, events may only be attached to satellites.

Events are displayed in the project timeline and in compatible client applications. Currently, the only standard VTS client application with events capability is SurfaceView.

An event is defined by its event type and timestamp. Text metadata may also be attached to an event.


Window Geometry Manager
#0
SurfaceView
#1
InfoBox
#2
NadirView
#3
SensorView
#4
SkyView
#5
ZenithView
#0
#4
#5
#1
#2
#3
Layout
Fullscreen
1
2
3
OK
Cancel


Window Geometry Manager
#0
SurfaceView
#1
InfoBox
#2
NadirView
#3
SensorView
#4
SkyView
#5
ZenithView
#0
#4
#5
#1
#2
#3
Layout
Fullscreen
1
2
3
OK
Cancel


Window Geometry Manager
Top
#0
SurfaceView
#1
InfoBox
#2
NadirView
#3
SensorView
#4
SkyView
#5
ZenithView
Select an application or an area
Also drag an application item, or open the application contextual menu
#1
#5
#2
Layout
Fullscreen
1
2
Bottom
OK
Cancel

2.17.1 Event type

The event type is a category describing the event that occurs at the dates of instances of this type. For example, events of the `OEF/EARTH_IN_SENSOR_END (2)` event type are orbit events at the dates when the Earth leaves the field of view of the second star tracker.

Event types in VTS are organized hierarchically. An event type full name includes the names of all its parent types, separated with slashes, e.g. `PASS/KRU/TC_EMISSION_END`.

2.17.2 CIC/CCSDS event files

Events are described in CIC/CCSDS files in MEM format. For more information on the CIC/CCSDS file format, refer to the CIC-CCSDS?data files in VTS chapter.

A CIC/CCSDS event file must have the following characteristics:

- The `USER_DEFINED_PROTOCOL` must be `NONE`.
- The `USER_DEFINED_CONTENT` is the top-level event type name for all events in the file. It will be implicitly prepended to the event type names found in the file.
- The `USER_DEFINED_SIZE` must be `1` or more. The first column (after the date) must be the event type full name (not including the top-level type, *USER_DEFINED_CONTENT*). All other columns will be used as text description/metadata for the event.
- The `USER_DEFINED_TYPE` must be `STRING`.
- The `USER_DEFINED_UNIT` must be `[n/a]`.

The following is a sample CIC/CCSDS event file:

```
CIC_MEM_VERS    = 1.0
CREATION_DATE   = 2013-07-18T16:19:51
ORIGINATOR      = PROTON

META_START

COMMENT PROTON OEF file
COMMENT Misc columns: PSO Lat Long Cycle Orbit

OBJECT_NAME = PLEIADES
OBJECT_ID   = PHR

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = OEF
USER_DEFINED_SIZE = 6
USER_DEFINED_TYPE = STRING
USER_DEFINED_UNIT = [n/a]

TIME_SYSTEM = UTC

META_STOP

56273 0.000000       "DAS_Update_TC_CHCOMRLDDAS"      "25.7"   "25.39"  "-26.5"  "8"
↪"272"
56273 126.809000     "AUS/0_DEG_AOS"                  "33.44"  "33.05"  "-28.48" "8"
↪"272"
56273 282.221000     "AUS/MAX_ELEVATION_PASS)"        "42.9"   "42.38"  "-31.3"  "8"
↪"272"
```

```
56273 438.342000     "AUS/0_DEG_LOS"                  "52.4"   "51.68"  "-34.89" "8"
↪"272"
56273 453.909000     "STH/0_DEG_AOS"                  "53.35"  "52.6"   "-35.32" "8"
↪"272"
56273 510.640000     "STH/PHYSICAL_AOS"               "56.8"   "55.95"  "-36.99" "8"
↪"272"
56273 566.297000     "STH/MAX_ELEVATION_PASS"         "60.18"  "59.21"  "-38.91" "8"
↪"272"
56273 616.599000     "KRN/0_DEG_AOS"                  "63.24"  "62.14"  "-40.93" "8"
↪"272"
56273 663.121000     "KRN/PHYSICAL_AOS"               "66.07"  "64.82"  "-43.15" "8"
↪"272"
56273 678.672000     "STH/PHYSICAL_LOS"               "67.01"  "65.71"  "-43.98" "8"
↪"272"
56273 678.672000001  "STH/0_DEG_LOS"                  "67.01"  "65.71"  "-43.98" "8"
↪"272"
56273 769.656000     "AUTONOMOUS_MODE_GAP/SUP_START"  "72.54"  "70.8"   "-50.17" "8"
↪"272"
56273 780.478000     "KRN/MAX_ELEVATION_PASS"         "73.2"   "71.39"  "-51.11" "8"
↪"272"
56273 803.998000     "EARTH_IN_SENSOR_START (3)"      "74.63"  "72.66"  "-53.34" "8"
↪"272"
56273 808.998000     "SUN_IN_SENSOR_START (3)"        "74.93"  "72.92"  "-53.85" "8"
↪"272"
56273 872.998000     "SUN_IN_SENSOR_END (3)"          "78.82"  "76.19"  "-62.02" "8"
↪"272"
56273 875.835000     "KRN/PHYSICAL_LOS"               "78.99"  "76.33"  "-62.47" "8"
↪"272"
56273 908.478000     "NPL/0_DEG_AOS"                  "80.98"  "77.86"  "-68.28" "8"
↪"272"
56273 935.329000     "NPL/PHYSICAL_AOS"               "82.61"  "79."    "-74.17" "8"
↪"272"
56273 942.531000     "AUTONOMOUS_MODE_GAP/SUP_END"    "83.05"  "79.29"  "-75.95" "8"
↪"272"
56273 944.742000     "KRN/0_DEG_LOS"                  "83.18"  "79.37"  "-76.51" "8"
↪"272"
56273 947.195000     "SHADOW_PENOMBRA"                "83.33"  "79.47"  "-77.15" "8"
↪"272"
56273 957.008000     "PENOMBRA_LIGHT"                 "83.93"  "79.83"  "-79.82" "8"
↪"272"
```

2.17.3 Event decorations

Event decorations define the appearance of events in the project timeline and in compatible client applications.

Each event type may have its own custom decoration, in order to be easily distinguishable from other event types. Note that all events of a single event type have the same appearance.

Thanks to the hierarchy of event types, event decorations are inherited: all children type of an event type share the same decoration as their parent type. This inheritance mechanism can be overridden by specifically customizing the decoration of an event type.

For more information on the specifics of event decoration configuration, refer to the *Configuring event types* section of the *VTS configuration utility user manual* chapter.

2.18 POIs and ROIs in VTS

POIs and ROIs in VTS are files defining Points and Regions of Interest for the visualization. They are displayed in 2D and 3D applications that support them. Both SurfaceView and Celestia support displaying POIs and ROIs.

2.18.1 CIC/CCSDS POI and ROI file format

POIs and ROIs are defined in CIC/CCSDS files in MPM format. For more information on the CIC/CCSDS file format, refer to the *CIC-CCSDS data files in VTS* chapter.

A CIC/CCSDS POI or ROI file must have the following characteristics:

- The `USER_DEFINED_PROTOCOL` must be `NONE`.
- The `USER_DEFINED_CONTENT` may be arbitrary.
- The `USER_DEFINED_SIZE` must be `2`, or `3` for POIs only.
- The `USER_DEFINED_TYPE` must be `REAL`, or `STRING` for POIs only with a size of 3.
- The `USER_DEFINED_UNIT` must be `[deg]`, or `n/a` for POIs only with a size of 3.

The data values in the file are coordinates in latitude/longitude.

In ROI files, the special data value `180 180` is interpreted as a separator and will create a new polygon with the following sequence of coordinates.

A single POI file may define multiple points of interest. In POI files with 3 columns, the last column defines a label for the corresponding coordinates.

2.18.2 Sample POI file

The following is a sample CIC/CCSDS POI file:

```
CIC_MPM_VERS = 1.0
CREATION_DATE = 2014-04-23T15:25:04.268055
ORIGINATOR = VTS

META_START

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = POINTS OF INTEREST
USER_DEFINED_SIZE = 3
USER_DEFINED_TYPE = STRING
USER_DEFINED_UNIT = [n/a]

META_STOP

48.861348   2.345248    "Siege social"
48.843445   2.390097    DLA
43.561948   1.481500    CST
5.208395    -52.775477  CSG
```

2.18.3 Sample ROI file

The following is a sample CIC/CCSDS ROI file containing two polygons:

```
CIC_MPM_VERS = 1.0
CREATION_DATE = 2014-04-23T15:25:04.268055
ORIGINATOR = VTS

META_START

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = REGION OF INTEREST
USER_DEFINED_SIZE = 2
USER_DEFINED_TYPE = REAL
USER_DEFINED_UNIT = [deg]

META_STOP

-30 30
30 30
0 0
180 180
0 0
-25 -25
25 -25
```

2.19 Other visualisable objects

2.19.1 Ellipsoid (Positional covariance of a satellite)

The positional covariance can displayed in the 3D view as an ellipsoid representing the uncertainty of an orbit state. It can be defined as a fixed value or as a CIC/CCSDS OEM or MEM file as 3 real values in kilometers. Different reference frames are available (EME2000, BodyFixed, TNW, QSW), relative to the current satellite.

Graphic properties such as visibility, color, transparency, scale can be configured in the scenario state editor.

2.19.2 Grid

A grid displays 3 different planes (XY, YZ, XZ) and can be attached to a satellite. It is defined with a grid size (Xmin, Ymin, Xmax, Ymax for each plane) and a cell size (spacing between lines defined as width and height). All units are in kilometers.

The visualizer is attached to its parent entity, and aligned to the a reference frame defined by a center entity and a frame type (EME2000, BodyFixed, TNW, QSW).

Graphic properties such as visibility, labels, color, transparency, can be configured in the scenario state editor.

2.19.3 Spherical Shell

A spherical shell is a generalization of an annulus to three dimensions. It is the region of a ball between two concentric spheres of differing radii. It is defined with 3 dimensions [latitude min, latitude max] in degrees, [longitude min, longitude max] in degrees, [altitude min, altitude max] in kilometers, and a step size for the grid cells for each dimension.

The visualizer is attached to its parent entity, and aligned to the a reference frame defined by a center entity and a frame type (EME2000, BodyFixed, TNW, QSW).

Graphic properties such as visibility, labels, color, transparency, can be configured in the scenario state editor.

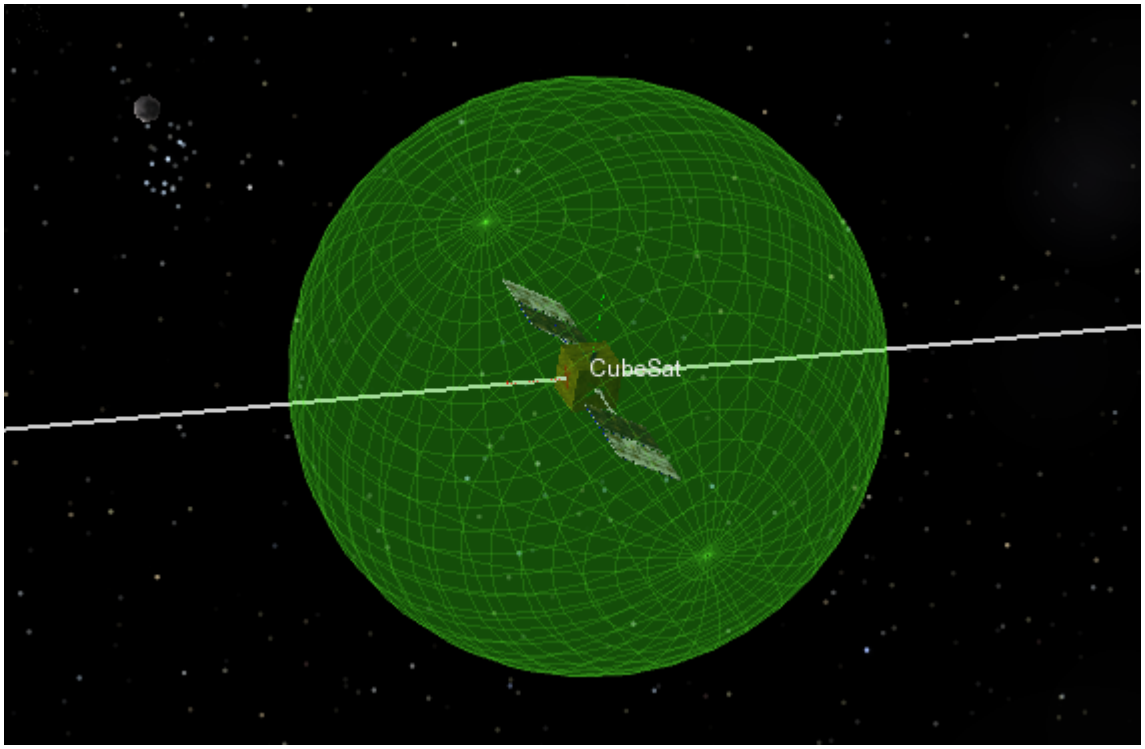


Fig. 2.32: Positional covariance ellipsoid in Celestia

2.20 Scripts and macros in VTS

Scripts and macros in VTS are commands from the VTS synchronization protocol, which are sent by the Broker to client applications during visualization.

Scripts are timestamped commands sent at predefined dates, while macros are non-timestamped commands sent in a batch upon macro execution. When the visualization date reaches the timestamp of a script command, the command is sent to all client applications specified in the recipient field of the script line.

Scripts are displayed in the project timeline. Macros are listed in the Broker menu. For more information on how to use scripts and macros in VTS, refer to the *Timeline* section in the *Scenario in VTS* chapter, and the *Broker menu* section in the *Broker user manual* chapter.

2.20.1 CIC/CCSDS script and macro file format

Scripts are written as CIC/CCSDS files in MEM format. Macros are written as CIC/CCSDS files in MPM format. For more information on the CIC/CCSDS file format, refer to the *CIC/CCSDS data files in VTS* chapter.

A CIC/CCSDS script or macro file must have the following characteristics:

- The `USER_DEFINED_PROTOCOL` must be `NONE`.
- The `USER_DEFINED_CONTENT` must be `SCRIPT` for a script, `MACRO` for a macro.
- The `USER_DEFINED_SIZE` must be `2`. The first column (after the date in the case of scripts) must be the recipient specification for the command (see below). The second column must be the command to send (see below).
- The `USER_DEFINED_TYPE` must be `STRING`.

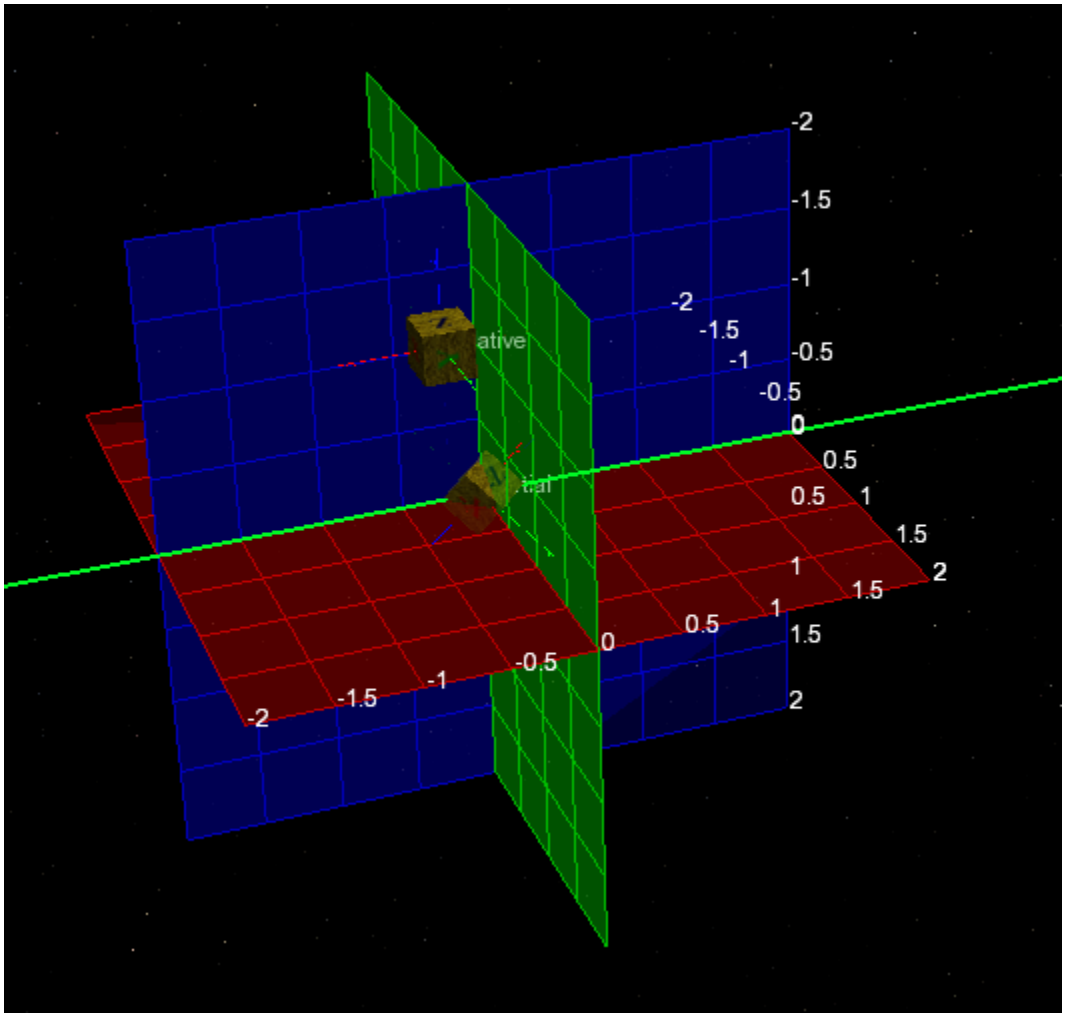


Fig. 2.33: A grid aligned on the TNW frame of a satellite

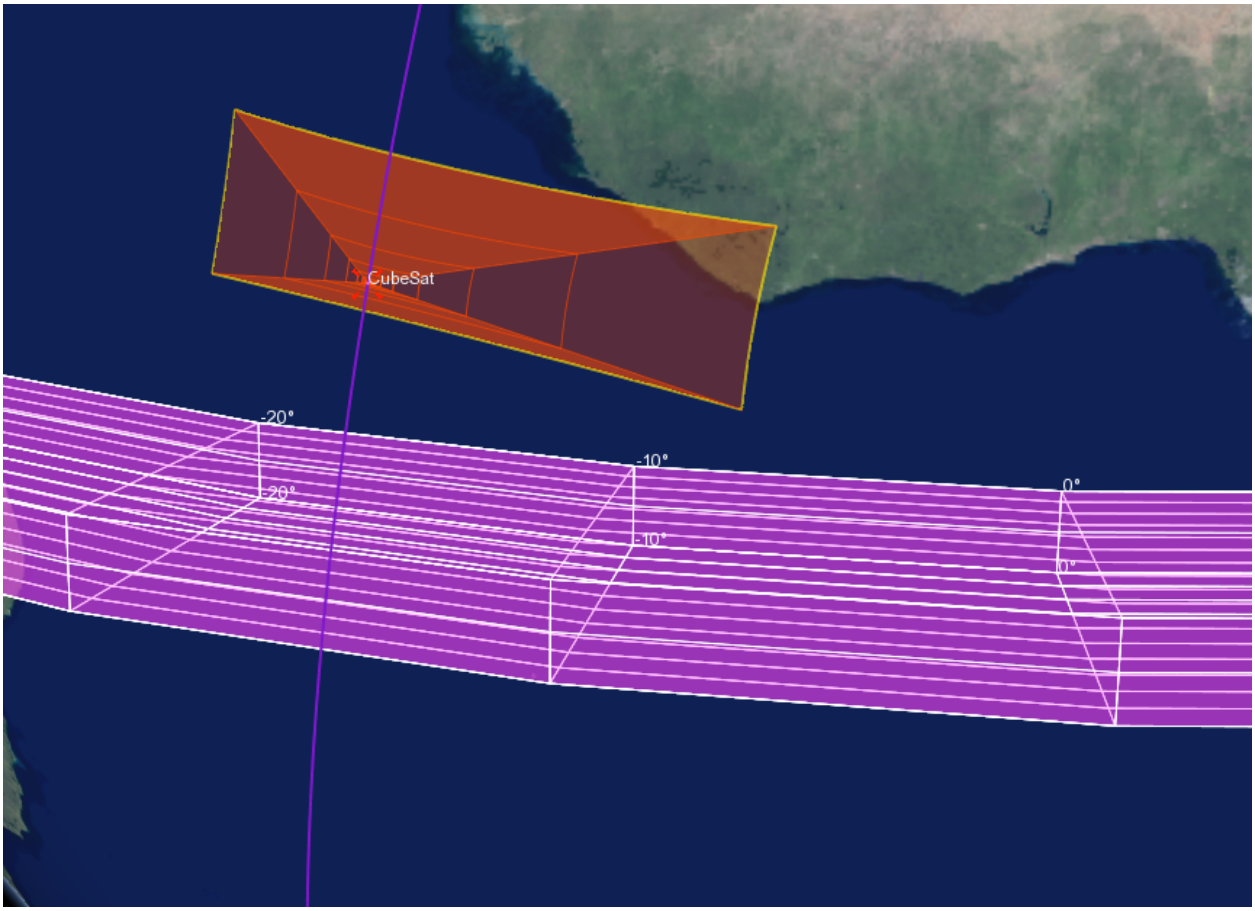


Fig. 2.34: A spherical shell centered on a satellite defined with a fixed position at the center of the Earth in a body fixed frame

- The `USER_DEFINED_UNIT` must be `[n/a]`.

2.20.2 Command recipient specification

The recipient for a script or macro command may be specified using any of the following:

- `*` indicates that the command shall be sent to all clients
- `broker` indicates that the command shall be sent to the Broker itself
- `<Name>` indicates that the command shall be sent to all client applications with the specified name
- `<ID>` indicates that the command shall be sent to the client with the specified ID
- `<Name>:<#>` indicates that the command shall be sent to the *n*-th instance of the client application with the specified name
- `:<#>` indicates that the command shall be sent to the *n*-th client application (using the order of the **Applications** tab of the Broker)

2.20.3 Command contents

Refer to the *Synchronization protocol for VTS clients* chapter for full details on the syntax of commands, and for a list of available commands in the standard client applications.

In order to allow "customized" commands to be sent to several client applications with a single line in a script or macro file, some special strings in script and macro file commands are automatically replaced before sending the command to clients:

- `%APPNAME%` gets replaced with the recipient client application name and ID, in the following format: `<Name>-<ID>`
- `%DATE%` gets replaced with the sending date, in the following format: `yyyy-MM-dd--hh-mm-ss-zzz`
- `%COUNT%` gets replaced with a sequence number, counting from 0 and shared between all script files, macro files and client applications in the project
- `%PROJECT%` gets replaced with the full path to the project folder

Note that when taking screenshots in Celestia, the `%PROJECT%` string is required at the beginning of the screenshot name to store the screenshot in the project folder. Otherwise, Celestia stores the screenshot in the `Apps/Celestia/bin` directory of the main VTS folder.

2.20.4 Sample script file

The following is a sample CIC/CCSDS MEM script file:

```
CIC_MEM_VERS    = 1.0
CREATION_DATE   = 2014-02-18T16:19:51
ORIGINATOR      = VTS

META_START

OBJECT_NAME = SCRIPT_SCREENSHOT
OBJECT_ID   = SCR001

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = SCRIPT
```

```
USER_DEFINED_SIZE = 2
USER_DEFINED_TYPE = STRING
USER_DEFINED_UNIT = [n/a]

TIME_SYSTEM = UTC

META_STOP

55276  3000 celestia   "CMD PROP equatorialgrid true"
55276  3001 *          "CMD SERVICE TakeScreenshot %PROJECT%/%DATE%_%APPNAME%_%COUNT%"
55276  3002 celestia   "CMD PROP equatorialgrid false"
55276  3003 *          "CMD SERVICE TakeScreenshot %PROJECT%/%DATE%_%APPNAME%_%COUNT%"
55276  3004 2dwin:0    "CMD STRUCT OrbitVisible \"Sol/Earth/CubeSat\" false"
55276  3005 *          "CMD SERVICE TakeScreenshot %PROJECT%/%DATE%_%APPNAME%_%COUNT%"
```

2.20.5 Sample macro file

The following is a sample CIC/CCSDS MPM macro file:

```
CIC_MPM_VERS    = 1.0
CREATION_DATE   = 2014-06-12T16:54:12
ORIGINATOR      = VTS

META_START

OBJECT_NAME = MACRO_4SPLIT
OBJECT_ID   = MAC001

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = MACRO
USER_DEFINED_SIZE = 2
USER_DEFINED_TYPE = STRING
USER_DEFINED_UNIT = [n/a]

META_STOP

:1 "CMD PROP WindowGeometry 0 0 640 480"
:2 "CMD PROP WindowGeometry 640 0 640 480"
:3 "CMD PROP WindowGeometry 0 480 640 480"
:4 "CMD PROP WindowGeometry 640 480 640 480"
```

2.21 VTS Broker

The Broker is the core application during visualization of a VTS project. It controls and synchronizes all client applications in time.

Client applications see the Broker as a "socket" server to which they connect and with which they communicate. Each client has a dedicated communication channel with the Broker. When an action or event within the Broker requires communication with its client applications, the Broker decides whether to broadcast the corresponding message to all clients or only to specific ones. Client-Broker communications follow a protocol based on text socket, described in more detail in the *Synchronization protocol for VTS clients* section.

Messages from the Broker to its clients concern:

- time synchronization

- view controls
- camera controls
- events management
- scripted commands
- context saving
- relayed messages from other clients

Some messages expect reply messages from the clients. Refer to the *Synchronization protocol for VTS clients* chapter for more information.

The Broker also has the responsibility of managing the life cycle of its clients (either configured in the VTS project or started by user interaction): spawning new clients, relaying client log messages to the user, displaying connection status about the clients, informing the user of client terminations (or crashes). External clients not started by the Broker obviously fall outside of this responsibility.

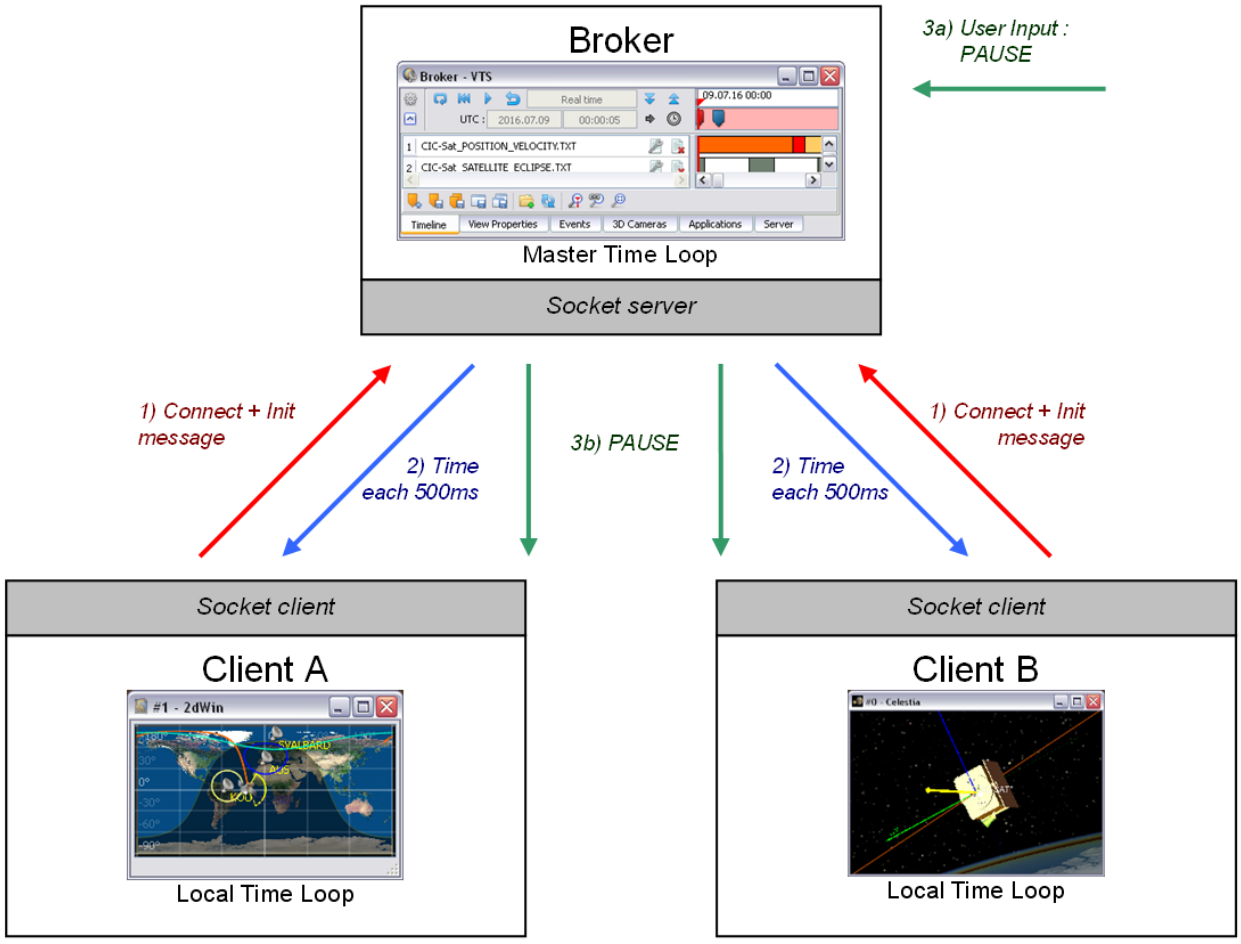


Fig. 2.35: Basic synchronization between Broker and clients

CHAPTER

THREE

FILES USED BY VTS

3.1 Data description for VTS projects

3.1.1 Hierarchy of a project folder

A VTS project consists in a folder containing one or several project files (`.vts` file extension), and a set of data files used by client applications during visualization. The project folder is the root folder containing the project file.

The project file contains the description in standard XML format of all the objects in the project. This description refers to data files as paths relative to the project folder. Portability concerns of project folders dictate that data files must reside within sub-folders of the project folder. Should the user select data files outside of the project folder, VTS will automatically offer to copy these files inside the project folder.

Remarks:

- It is recommended to create one folder per VTS project, so that modifying a data file for a given project will not silently affect other projects.
- The project folder can be stored on any device (hard disk drive, USB stick, CD) and must not necessarily be located in a sub-folder of the main VTS folder.
- Some client applications require the availability of data located in folders following a strict nomenclature. The README files for these applications describe their requirements.

3.1.2 2D icons and textures

Objects projected on the main surface in 2D views are displayed as icons. These icons may be in any of the following formats: *.bmp*, *.gif*, *.ico*, *.jpg*, *.mng*, *.pbm*, *.pgm*, *.png*, *.ppm*, *.svg*, *.svgz*, *.tga*, *.tif*, *.tiff*, *.xbm*, *.xpm*. Icon files are also used for event decorations. The same formats are supported.

Custom textures for central bodies may also be in any of the above formats.

3.1.3 3D models and textures

The description of a satellite refers to 3D model files. Refer to the *3D file format in VTS* chapter for more information.

3.1.4 Importing 2D icons and 3D models from the catalog

VTS supplies a catalog of icons and 3D model. Clicking on the **import file from catalog**  button next to an icon or model **browse...** button will open a dialog window. Validating the import will offer to copy these files inside the project folder.

Concerning 3D models, a folder contains only one 3DS file and its textures files. Importing the model will import all the files of the folder at once.

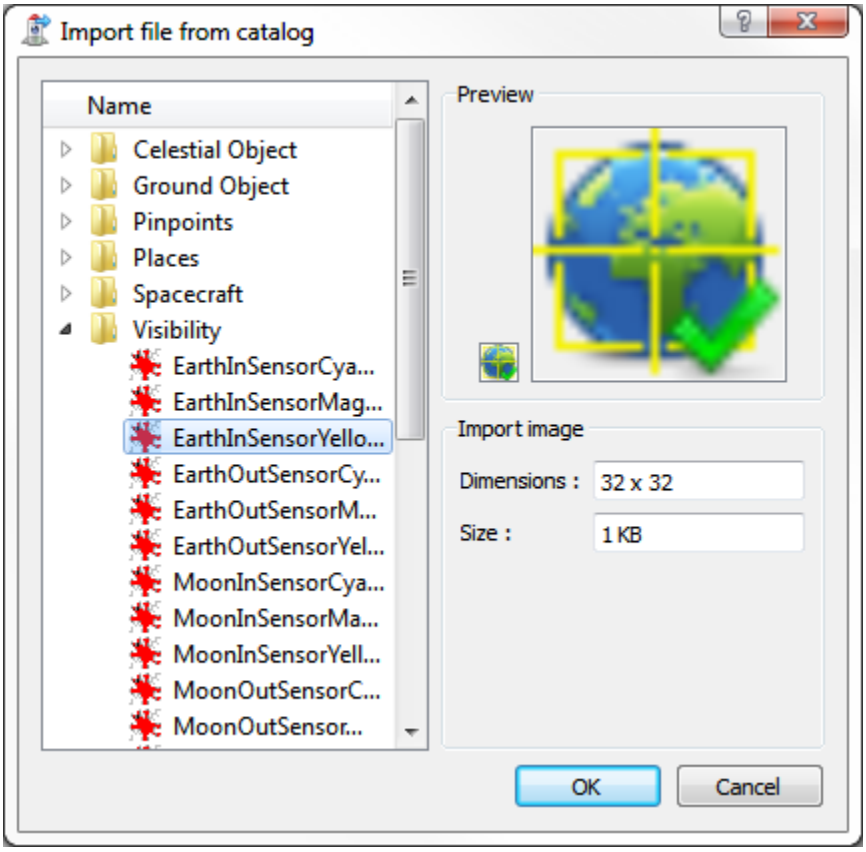


Fig. 3.1: Importing an icon from the catalog

3.1.5 The CIC/CCSDS file format

This file format is used for various text data, e.g. position and attitude ephemerides. Refer to the *CIC/CCSDS data files in VTS* chapter for more information.

3.2 3D file format in VTS

VTS relies on the *3ds* file format, from the *Autodesk 3DS Max* software suite, but also on the *obj* file format. These are widely-used file formats for 3D files.

3.2.1 File format of textures

3D files can make use of external texture files. These must be in *bmp*, *jpg*, *jpeg*, *png*, *tif* or *tga* format, and be located in the same folder as the 3D file.

The *obj* file format also relies on a *mtl* file which must be located in the same folder as the *obj* file. It represents the definition of the materials used by the 3D file.

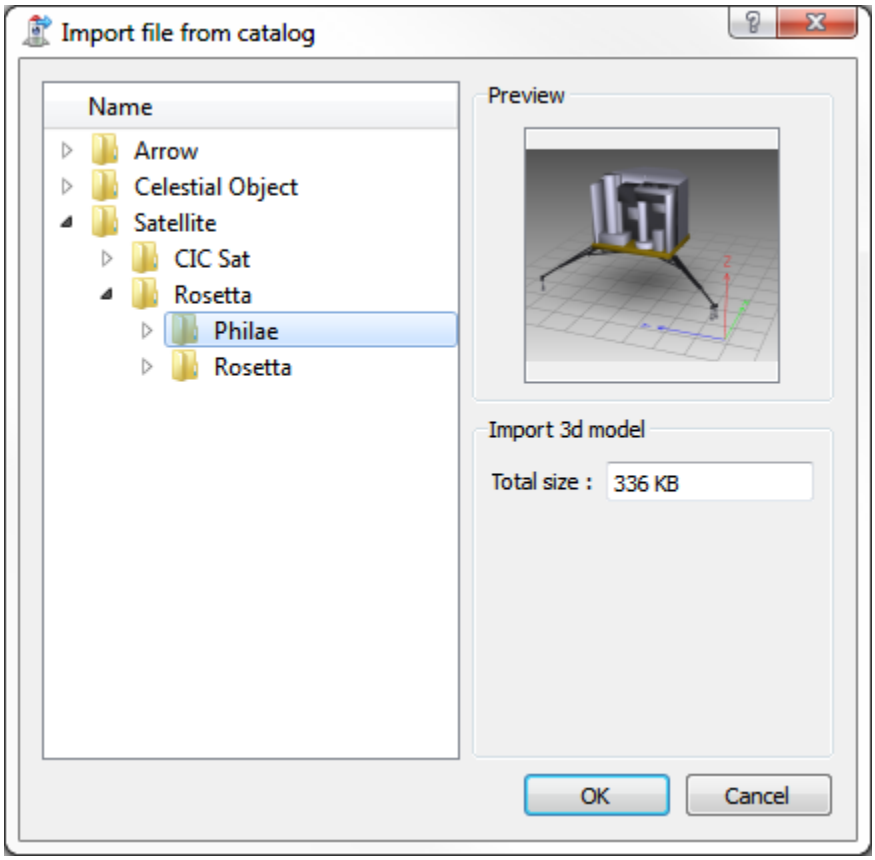


Fig. 3.2: Importing a model from the catalog

3.2.2 Origin of the reference frame for the 3D file

The position and orientation of objects in 3D views depends on their reference frames. Refer to the following pages:

- *Position of objects in VTS*
- *Orientation of objects in VTS*

When position and orientation values are null, the object's local frame is aligned with its reference frame. VTS offers two options to define the local frame of a 3D object:

- Either use the 3D coordinates contained in the 3D file. The origin of the local frame corresponds to the point of null coordinates in the file.
- Either define the origin of the local frame as the center of the 3D file object's bounding box.

These options can be selected in the VTS configuration utility, independently for each 3D object. The *"Use file coordinates"* setting must be checked to enable use of the 3D file's origin, unchecked to use the geometric center of the object as origin. Refer to the *VTS configuration utility user manual* chapter for more information.

It must be well understood that these options make it easier to define satellites composed of multiple components. When all 3D files for a satellite's components and sub-components are extracted from a single master 3D file, all those files use the same local frame and thus the coordinates can be read directly from the files. The *SMOS* example supplied with VTS perfectly illustrates this situation.

3.2.3 Dimensions and units

As above, the size of a 3D object can either be read as is from the 3D file, or defined by VTS. This choice also depends on the option used for the origin of the local frame.

When the origin is the geometric center of the object (the *"Use file coordinates"* option is unchecked), the size of the object is defined in the VTS configuration utility. The specified size, in meters, is used as the dimension of the smallest bounding box for the object.

When the origin is taken from the 3D file, the specified size is used as an hint of the object's size, for the 3D views to optimize its display. The given size should be accurate within a factor of 2 (hence the *Approximate size* label). The user must also specify the unit of the 3D file's coordinates. The default unit is the meter.

3.3 CIC/CCSDS data files in VTS

3.3.1 Introduction

The data files used by VTS and its client applications are written in the CIC/CCSDS format, a column-based text format derived from the CCSDS OEM, AEM and MPM formats.

The CIC/CCSDS file format is described in the reference document *CIC Data Exchange Protocol V2.0* (DCT/DA/PA - 2009.0021267). This document is included in the VTS root directory. You will also find many examples of CIC/CCSDS files in the "Data" directory.

Some features of the CIC/CCSDS file format:

- Standardized header depending on file type
- Data as timestamped lines
- MJD date format with 2 fields respectively for days and seconds (see the *Date formats in VTS* section)
- Satellite position in kilometers

- Orientation as quaternions from EME2000 towards satellite local frame
- Heterogeneous data units within the same file

VTS handles World Geodetic System (WGS84) in SurfaceView.

3.3.2 Heterogeneous data units

CCSDS files can store data with multiple units. This allows for the use of data such as Lat/Long/Alt (degrees, degrees, meters). When explicitly specified, the unit is defined in the `USER_DEFINED_UNIT` field of the CCSDS header, enclosed by square brackets.

Multiple units can be defined by putting pipes between different units. The order of the units corresponds to the order of the data columns.

```
USER_DEFINED_UNIT = [deg|deg|m]
```

CHAPTER

FOUR

STARTING VTS

4.1 Starting the VTS configuration utility

The VTS toolkit can by started by double-clicking the `startVTS.exe` file under Windows, or by executing the `./startVTS` command under Linux. This displays the main window of the VTS configuration utility. It allows creating a project by setting up the entities to be visualized: satellites, sensors, ground stations, and client applications.

If the launcher is executed from the command line with the `--project <ProjectFile.vts>` argument, the VTS configuration utility automatically loads the given project on startup.

4.2 Starting the visualization from the command line

The visualization can be started automatically from the command line via the launcher. The Broker then opens and starts the visualization, without going through the VTS configuration utility.

In batch mode, the following arguments are mandatory:

- `--batch` instructs the launcher to start the Broker directly
- `--project <ProjectFile.vts>` specifies the project file to load. The path can be either relative or absolute.

Other command-line arguments for the Broker can also be given in batch mode.

Sample command line: `startVTS.exe --batch --project C:\Project\CubeSat.vts`

On Linux only, the version of VTS can be obtained with the command: `./startVTS.exe --version`

4.3 Troubleshooting a crash of VTS

If the execution of VTS has a problem, an email could be sent to the Spacebel contact (vts-team@spacebel.fr). The environment being really important to understand, the VTS user can generate the log of environment by clicking on the tab `Help->System Information...` from the main interface. It will produce a `vtsdiag.log` in the VTS root project that can be attached to the email.

CHAPTER

FIVE

VTS APPLICATIONS USER MANUALS

5.1 VTS configuration utility user manual

The VTS configuration utility is the entry point for the VTS toolkit. It allows creating and editing VTS projects, as well as starting the visualization of a project.

A project is composed of a set of configurable entities such as satellites, central bodies, ground stations, and client applications. These entities will be used during the visualization phase. It is also possible to configure a scenario of view properties which will be played back during the visualization. The following paragraphs describe in detail how to edit a VTS project.

5.1.1 Command line arguments

Command line arguments for the VTS configuration utility can be passed from the *startVTS* binary using the following syntax:

```
startVTS [VTS configuration utility arguments]
```

The following command line arguments are available:

- `--project <project.vts>` : Open the specified VTS project file upon start of the VTS configuration utility.
- `--migrate <project.vts> <migrated_project.vts>` : Migrate the specified VTS project file at the current VTS project version and save it at specified location.
- `--readonly` : Start VTS in read-only mode, meaning it will not write to its installation directory. It may still write project files outside its installation directory. This mode is enabled automatically if VTS detects it cannot write to its installation directory.

> **Note:** In this mode, changes to the VTS settings or to a project file located inside the installation directory will be written to a temporary directory instead, and discarded when VTS is closed.

5.1.2 User interface description

The user interface is composed of:

- A menu and toolbar **(1)**
- A pane containing the project entities tree **(2)**
- A pane displaying properties of the currently selected entity in the project tree **(3)**

- A text area for notifications (warnings, errors, etc.) **(4)**
- **A tab area displaying the editors** [**(5)**]
  - The **Scenario Editor** tab features all the tools required to edit the project's scenario. These tools are detailed in the chapter about concepts specific to VTS, in the *Scenario in VTS* section.
  - The **Event Type Editor** tab allows customizing the appearance of mission events attached to the project's satellites.

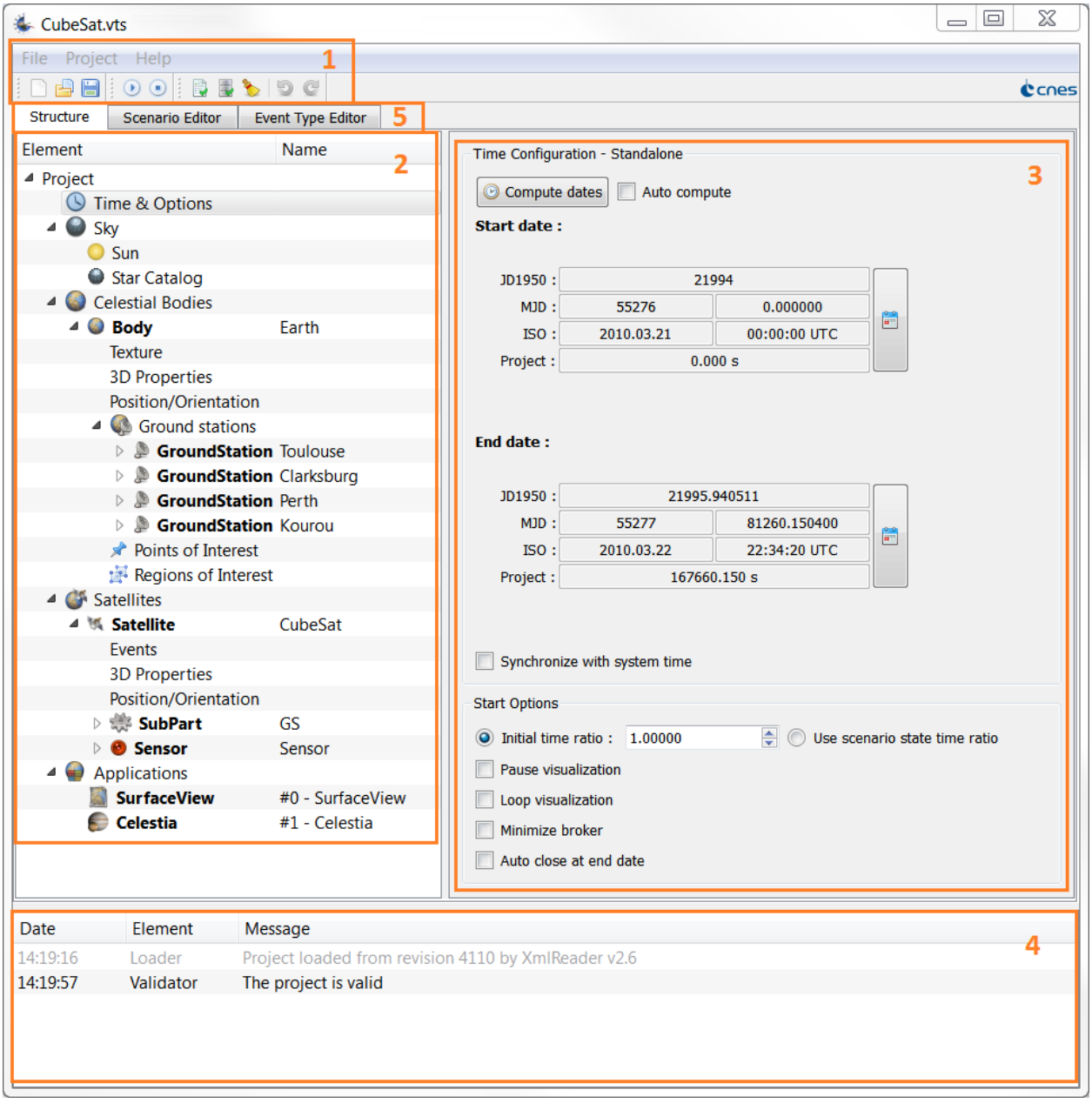


Fig. 5.1: User interface of the VTS configuration utility

The position of the configuration utility window is saved between two sessions of the application. Multiple monitors environments are supported.

The configuration utility window is guaranteed not to disappear in the case of a change of the desktop geometry. When

a new session of the application begins, if the monitor displaying the window is no longer available, the window will move itself inside the closest available monitor.

Messages displayed in the notification area are also stored in a log file (vts.log). This file can be found in the "Apps/Logs" directory or in the temporary directory (if readonly is on).

5.1.3 Toolbar actions

The actions below are available in the toolbar of the VTS configuration utility:

- **New Project** button (*Ctrl+N*): create a new project
- **Open Project** button (*Ctrl+O*): load an existing project
- **Save Project** button (*Ctrl+S*): save the current project
- **Run** button (*Ctrl+R*): start the visualization
- **Stop** button: stop the current visualization
- **Check project** button: check that the project is valid
- **Check project and data files** button: check that the project and its data files are valid
- **Clear Logger** button: clear all messages in the notification pane (4)
- **Undo** button (*Ctrl+Z*): undo the previous modification of the project
- **Redo** button (*Ctrl+Y*): redo the next modification of the project (only available with a previously undone modification)
- **Start Propagator** : start the *Propagator*

A popup window displaying the history of all modifications to the current project can be displayed with the *Ctrl+Alt+Z* keyboard shortcut.

Use *Ctrl+F* to filter the project hierarchy using case-insensitive regexp patterns.

Validity of the project

The **Check project** button triggers a validation of the project integrity. If any error or warning is encontoured, they will be reported on the logger panel. Double-clic an error will focus the project to the errered item.

The following items are checked:

- **Applications check**
  - Client applications of the project exist
  - Launchers exist for each of those client applications
- **Central bodies check**
  - At least one central body is defined
  - All central bodies have unique names
  - Central body textures are valid
  - If applicable, all 3D models used by central bodies exist
  - If applicable, all data files used by central bodies are valid (shallow check)
  - All ground stations on a single body have unique names

  - All POI on a single body have unique names
  - All data files used by POI are valid (shallow check)
  - All ROI on a single body have unique names
  - All data files used by ROI are valid (shallow check)
- **Star Catalog check**
  - If the Star Catalog is configured in mode *Custom*, the specified catalog file is checked (shallow check)
- **Satellites check**
  - All satellites have unique names
  - All satellites are attached to a central body
  - All 3D models used by satellites exist
  - If applicable, all data files used by satellites are valid (shallow check)
  - **Components check**
    - All components of a single satellite have unique names
    - All 3D models used by components exist
    - If applicable, all data files used by components are valid (shallow check)
    - Sub-components check
  - **Sensors check**
    - All sensors on a given satellite have unique names
    - If applicable, all data files used by sensors are valid (shallow check)
- **Dates check**
  - Project start date is before its end date
  - All data files used by the project cover the time range of the project
  - Catalog ephemerides for default bodies used in the project cover the time range of the project
- **Additional checks**
  - All additional files displayed in the timeline are valid (shallow check)
  - All script files used by the project are valid (shallow check)

Checks on data files are shallow, i.e. only the header and file metadata are checked:

- No errors in the header
- **Data type described in the metadata matches what VTS expects**
  - Data type
  - Data dimension
  - Data unit

Validity of the project and its data files

The **Check project and data files** action covers the perimeter of the **Check project** action, and also ensures the contents of all data files can be loaded without errors.

5.1.4 Managing a project

Creating a project

By definition, the project folder is the folder containing the current project file (".vts). This folder also contains data referenced by the project. A VTS project file is an XML file describing the entities of the visualization, the start and end dates of the visualization, the client applications for the visualization, and the states of the visualization scenario. While editing a project, if the user selects some data or 3D file outside the project folder, a popup will offer to copy the file under the project folder.

Several project files may reside in a common folder. However caution must be taken when altering a shared data file, as this will impact all projects using it.

A new project can be created by doing the following:

- In the toolbar, click the **New project** button, or in the **File** menu click the **New project** entry.
- The **Create a new project** dialog opens, asking for a location to save the project. Select a folder which will contain all files related to the project, fill in the project's name, and click **Save**.

Opening a project

An existing project can be opened by doing one of the following:

- **Select the project file on disk**
  - In the toolbar click the **Open project** button, or in the **File** menu click the **Open project** entry.
  - The **Open project file** dialog opens. Select a project file (*.vts*) and click **Open**.
- Select the project file in the list of the last 15 project files available in the **File** menu.
- Drag-and-drop the project file from a file browser onto the window of the VTS configuration utility.

Opening a project file by double-clicking on it is currently not supported, due to the VTS toolkit being portable.

Saving a project

The current project can be saved to disk by doing one of the following:

- In the toolbar click the **Save** button, or in the **File** menu click the **Save** entry.
- To save the project under a different name, click the **Save as...** entry in the **File** menu and modify the name of the project file. Beware that if the project is saved under a different folder, all relative paths in the project will become erroneous.

Exporting an archived project

The current project can be exported as a zip archive by clicking on the **Create a Project Archive...** entry in the **File** menu. The .vts project file, all the data files in use, all the symbol files in use, and all the 3ds model files in use will be archived. Notice that all the image files (bmp, png or jpg) found in the used models folders will be exported as VTS doesn't know which texture files are effectively reference by the models files.

Visualizing a project

The current project can be visualized by doing the following:

- Start the visualization by clicking the **Run** button in the toolbar, or the **Run** entry in the **Project** menu.

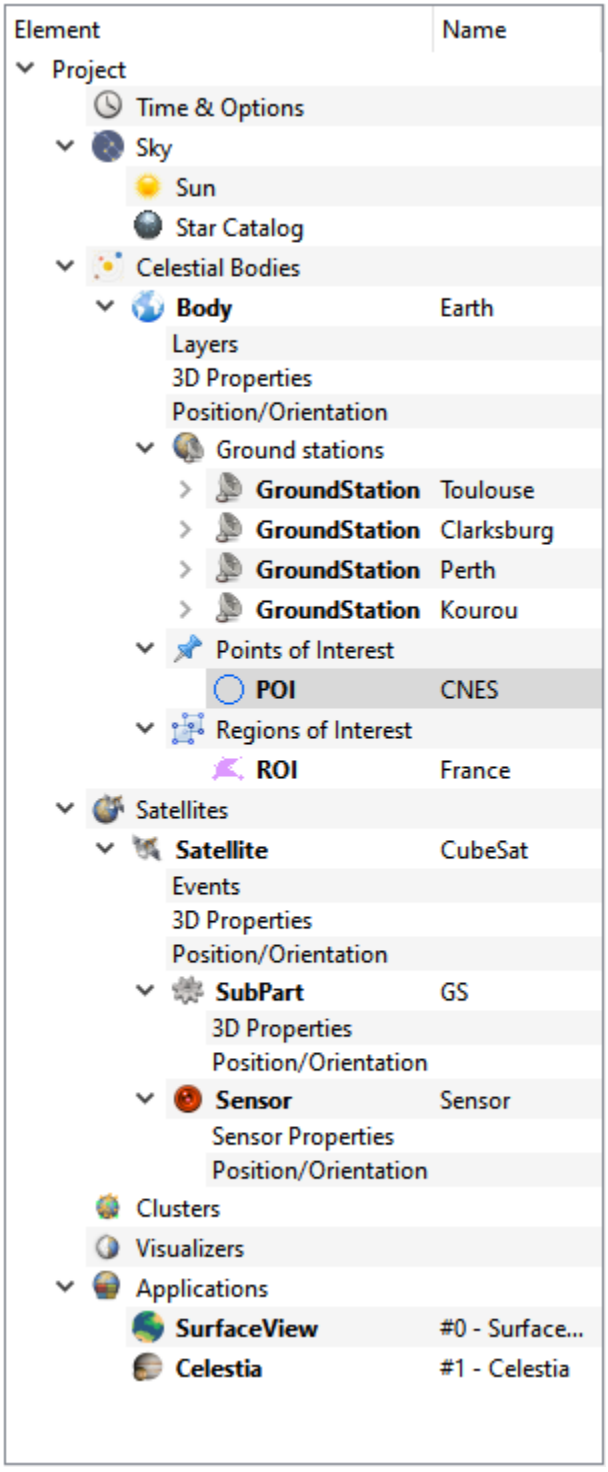
When the visualization ends, a popup offers to load any changes made to the project through the Broker during visualization. These changes can then be saved or discarded.

The Project Hierarchy

The project hierarchy describes the project's entities in a hierarchical fashion. The hierarchy can be modified through context-sensitive menus.

The following actions can be accessed by right-clicking the project hierarchy:

- **on any item**
  - *Collapse* folds the selected tree item
  - *Expand* unfolds the selected tree item
  - *Copy* puts a copy of the selected tree item in the clipboard
  - *Paste* pastes a previously copied item onto the selected tree item
- **on the Project item**
  - *View XML* opens the project file in the text editor associated with the *.vts* file extension
  - *Open project location* opens a file browser in the project folder
- **on the Celestial Bodies item**
  - *Add Body* adds a new body to the list of central bodies
- **on a Body item**
  - *Remove* removes the selected body
- **on a Ground stations item**
  - *Add GroundStation* adds a new ground station to the current body
- **on a Ground station item**
  - *Remove* removes the selected ground station
- **on a Points Of Interest item**
  - *Add Point Of Interest* adds a new set of points of interest to the current body
- **on a POI item**
  - *Remove* removes the selected set of points of interest
- **on a Regions Of Interest item**
  - *Add Region Of Interest* adds a new region of interest to the current body
- **on a ROI item**
  - *Remove* removes the selected region of interest
- **on the Satellites item**
  - *Add Satellite* adds a new satellite to the list of satellites


Element
Name
Project
Time & Options
Sky
Sun
Star Catalog
Celestial Bodies
Body
Earth
Layers
3D Properties
Position/Orientation
Ground stations
GroundStation
Toulouse
GroundStation
Clarksburg
GroundStation
Perth
GroundStation
Kourou
Points of Interest
POI
CNES
Regions of Interest
ROI
France
Satellites
Satellite
CubeSat
Events
3D Properties
Position/Orientation
SubPart
GS
3D Properties
Position/Orientation
Sensor
Sensor
Sensor Properties
Position/Orientation
Clusters
Visualizers
Applications
SurfaceView
#0 - Surface...
Celestia
#1 - Celestia

- **on a Satellite item**
  - *Add SubPart* adds a new component to the selected satellite
  - *Add Sensor* adds a new sensor to the selected satellite
  - *Add Visualizer* adds a new visualizer to the selected satellite
  - *Remove* removes the selected satellite
- **on a SubPart item**
  - *Add SubPart* adds a new sub-component to the selected component
  - *Add Sensor* adds a new sensor to the selected component
  - *Remove* removes the selected component
- **on a Sensor item**
  - *Remove* removes the selected sensor
- **on the Clusters item**
  - *Add Cluster* adds a new cluster to the list of clusters
- **on the Visualizers item**
  - *Add Link* adds a new link between two satellites of the project
- **on the Applications item**
  - *Add Application* adds a new client application to the project (selected in the dropdown menu)
- **on any client application item**
  - *Remove* removes the selected client application

The *Del* keyboard shortcut can be used to remove an item from the hierarchy, if the selected item can be removed.

Use *Ctrl+F* to filter the project hierarchy using case-insensitive regexp patterns.

Copy-pasting entities

The copy-paste feature can be used to quickly create entities in the project hierarchy.

An entity can be copied through the clipboard either through the *Ctrl+C* keyboard shortcut, or through the *Copy* entry in the context menu. Central bodies, satellites, components, sensors, ground stations, points and regions of interest can be copied.

A previously copied entity can be pasted through the *Ctrl+V* keyboard shortcut, or through the *Paste* entry in the context menu, once a compatible destination entity has been selected (e.g. a satellite to paste a component, a central body to paste a ground station, etc.). If the selected entity is not a valid destination for the entity currently in the clipboard, the paste operation will fail. If the destination is the same entity as the one currently in the clipboard (i.e. when hitting *Ctrl+C* and *Ctrl+V* in sequence), a duplicate of the selected entity is created on the same level of the hierarchy. Note that in order to paste a component as a sub-component of itself, a special paste entry is available in the context menu as *Paste as subpart*.

When copy-pasting an entity, all of its structural properties (i.e. those set in the *Structure* tab) are copied and pasted, as well as all its scenario properties (i.e. those set per-application and per-state in the *Scenario editor* tab). Entities are pasted as they were upon copy: if an entity is copied, modified, then pasted, the new entity will have the properties its source entity had at the time of the copy.

The following table lists the compatible destination entities for each source entity:

Table 5.1: Destinations for pasting a copied entity

Source entity type	Valid destinations
Central body	**Project, Celestial bodies item**
Ground station	**Body, Ground stations item**
Point Of Interest	**Body, Points Of Interest item**
Region Of Interest	**Body, Regions Of Interest item**
Satellite	**Project, Satellites item, Body**
Component	**Satellite, SubPart**
Sensor	**Satellite**
Cluster	**Project, Clusters item, Body**

Importing entities from an external project

VTS allows entities to be imported into the current project, from an external project file. This feature is available through the **Import entities** entry in the **Project** menu.

The hierarchy of the source project is displayed in the left pane, while the hierarchy of the current project is displayed in the right pane.

To import an entity from the source project, simply select the entity to be imported in the source hierarchy, its destination entity in the destination hierarchy, and click the import arrow button located between the two panes. Valid destinations are listed in the above table (in the *#Copy-pasting entities section* section). The import of an entity can be canceled by selected it in the right pane and clicking the cross button located between the two panes.

The **Files** group allows enabling or disabling the following options:

- **Import 3D model**: import the 3D models and textures used by imported entities
- **Import icons**: import the icons used by imported entities
- **Import CIC files**: import the CIC/CCSDS data files used by imported entities

The relative paths to the imported files in the source project will kept after the import. If a filename collision occurs, the user will have to decide whether to skip the import of the conflicting file, rename it, or overwrite the existing file.

The **Scenario** group is only enabled when client applications are identical in both the source and destination projects. It allows enabling or disabling the following options:

- **Import scenario states**: import the scenario states from the source project which do not already exist in the destination project
- **Remove states outside current project dates**: do not import scenario states from the source project which are outside the date range of the destination project

The **Events** allows enabling or disabling the following options:

- **Import default event decorations**: import the default event decorations for all satellites
- **Import satellite event decorations**: import event decorations for the imported satellites (only available when at least one satellite has been selected for import)

Once the import has been configured, it can be applied by clicking the **OK** button, or canceled by clicking the **Cancel** button. Once the import has been applied, it can still be undone thanks to the **Undo** feature of VTS. Beware that file operations will not be undone.

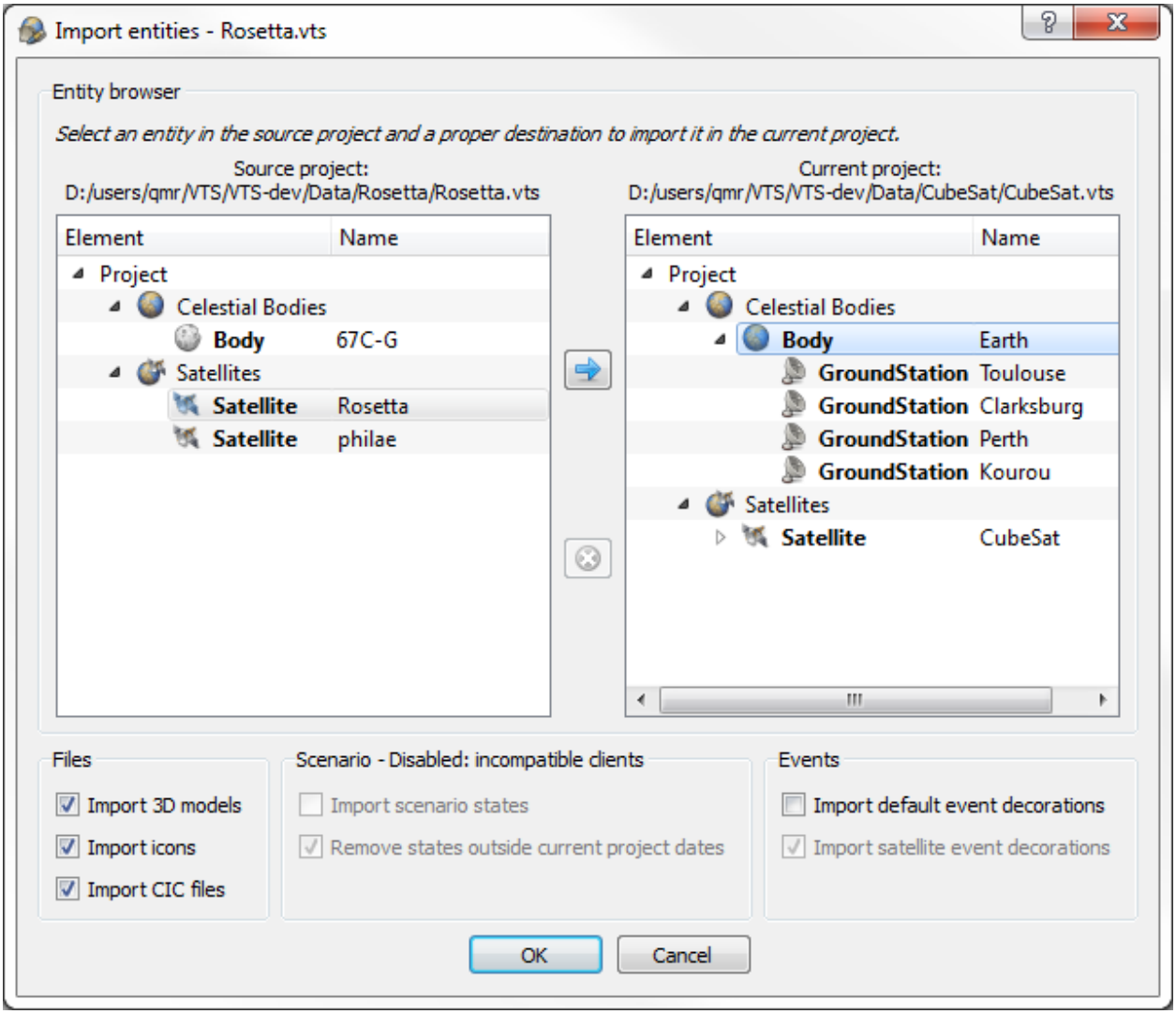


Fig. 5.2: Dialog for importing entities from an external project

5.1.5 Configuring the VTS project and its entities

General parameters of a project

The general parameters of a project are its start and end dates, and the initial properties of the visualization (used by the Broker).

Configuring the project dates

- The **Compute dates** button automatically computes the date range of the project based on the widest common time interval of all ephemerides files used in the project.

  Note: Dates are not updated when a data file or entity is added/removed from the project.

  Note: The date range is computed from the bodies and the satellites entities. Events dates are **NOT** taken into account.

- The **Auto compute** option enables automatic computation of the project's date range when the user starts the visualization. This option is not saved in the project XML and can be changed manually after loading the project if needed. By default this option is unchecked, but this can be modified in the **Options/Settings...** dialog.
- The **Change date...** button located besides the project start and end dates allows manually editing the dates in CNES julian day format (JD1950), modified julian day format (MJD), calendar format (ISO), and relative to project start date (Project).
- The **Synchronize with system time** option allows synchronizing the visualization time with the computer's system clock. In this mode, time controls are disabled during visualization. Data files must provide data for the visualization time range.

Refer to the *Date formats in VTS* for further information on dates.

Configuring initial properties of the visualization

- The **Initial time ratio** setting allows specifying the initial visualization time ratio
- The **Pause visualization** option allows starting the visualization paused/unpaused
- The **Loop visualization** option allows the visualization to start over once it reaches the end of the time range
- The **Minimize broker** option allows the Broker to start minimized in the taskbar
- The **Auto close at end date** option allows the Broker to automatically close once the visualization reaches the end of the time range

Configuring the timeshifting

- The **Enabled** setting allows activating the timeshifting feature

Refer to the *Timeshifting in VTS* for further information about this feature.

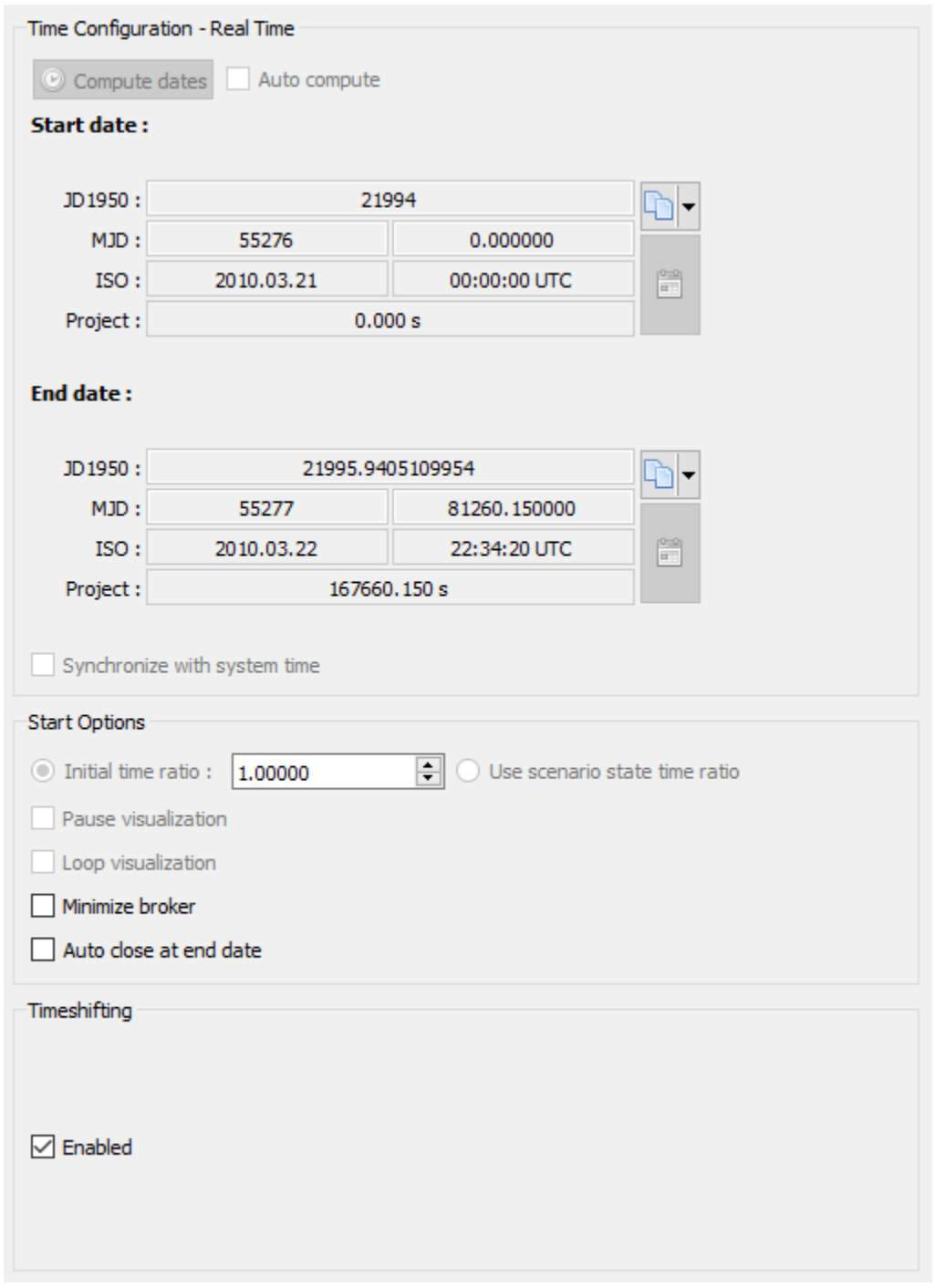


Fig. 5.3: Configuration of the project dates and initial properties of the visualization

Configuring an entity

Several entities in the project rely on some common parameters, which are described in the current section. Parameters specific to an entity are described in the sections concerning their respective entities.

Position and orientation of an entity

This section describes the common mechanisms used for defining the position and orientation of an entity. Details specific to the various entities (such as the units used) are described in their respective sections.

For further information on how position and orientation are defined, refer to the *Position of objects in VTS* and *Orientation of objects in VTS* sections.

For further information on the available data sources, refer to the *Data sources in VTS* section.

Reference frame

By default, all coordinates are expressed in a reference frame defined by convention for each type of entity (refer to the *Position of objects in VTS* and *Orientation of objects in VTS* sections).

For satellites whose coordinates are expressed in a local reference frame relative to another satellite, it is possible to select a different type of reference frame (EME2000, BodyFixed, QSW, TNW).

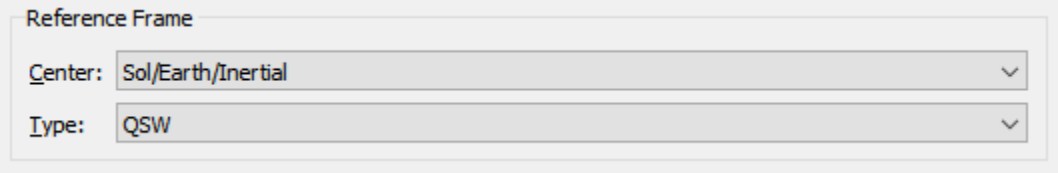


Fig. 5.4: Satellite Reference Frame

Fixed position

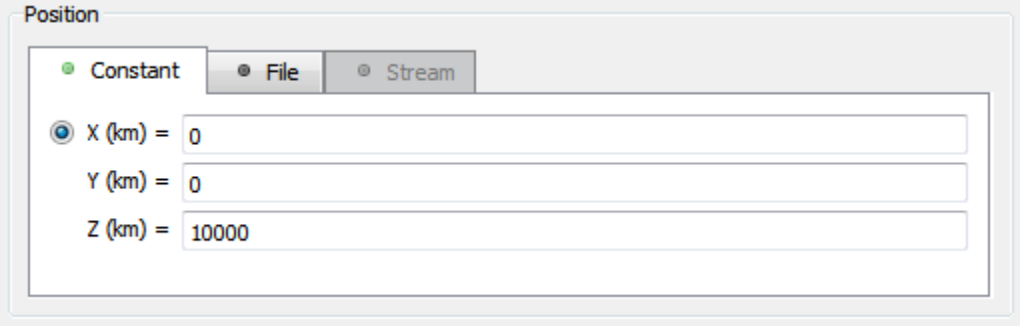


Fig. 5.5: Fixed position

- The radio button in the **Constant** tab selects a fixed position for the current entity. Note that in the EME2000 reference frame, a fixed position is usually meaningless. This position mode is mainly used for defining the coordinates of a component in the satellite's local frame.

- By default, positions are expressed in a Cartesian coordinate system. A combo box next to the fixed position value allows to change to a geographic coordinate system. This case is useful if the reference frame is defined as BodyFixed relative to a central body.

Position from a file

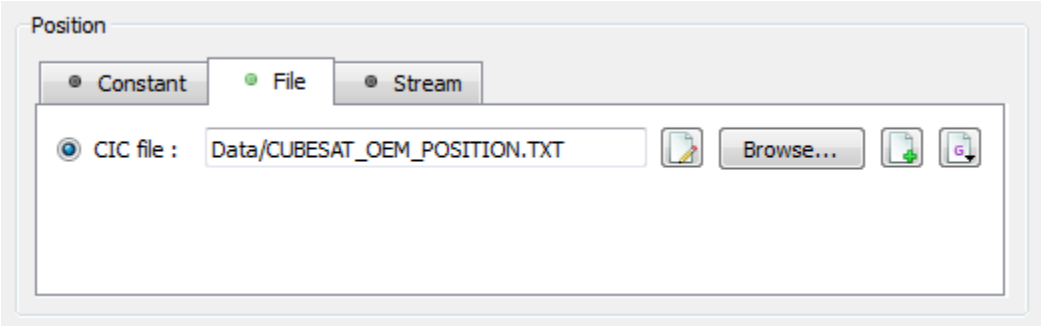


Fig. 5.6: Position defined by an ephemerides file

- The radio button in the **File** tab allows specifying the path to a CIC/CCSDS file containing the position ephemerides for the current entity. The CIC/CCSDS file format is defined in the CIC/CCSDS data files in VTS section.
- The **Edit file...** button opens the currently selected file in a text editor.
- The **Browse...** button opens a dialog to select an existing CIC/CCSDS file from disk. If the selected file is not located in a sub-folder of the project folder, a dialog offers to copy it inside the project folder.
- The **New CIC file...** button creates an empty CIC/CCSDS file for the user to edit manually. Note that the CIC/CCSDS file header will depend on the currently selected type of position or orientation.

Position as a stream

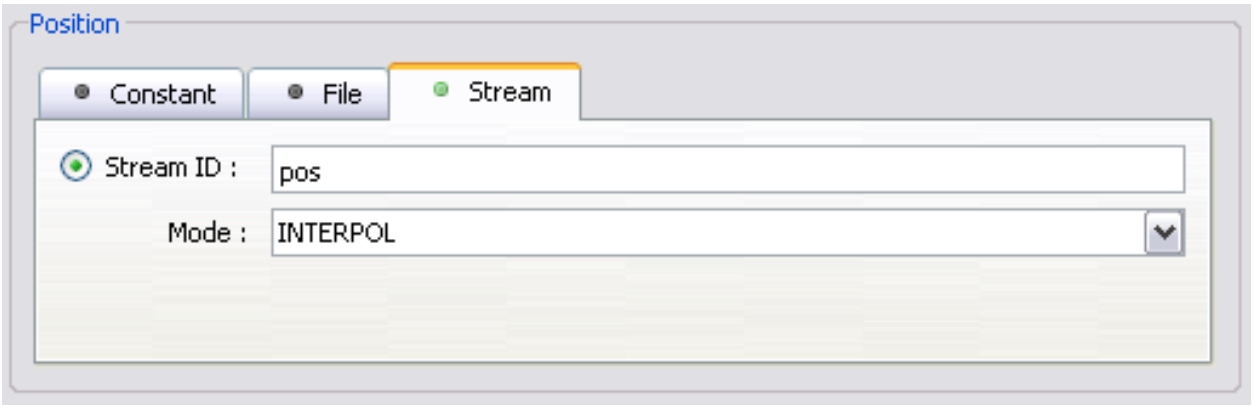


Fig. 5.7: Position defined by a stream

- The radio button in the **Stream** tab allows specifying the ID of a data stream.
- The stream mode can be selected as *INTERPOL* or *DIRECT*. In *INTERPOL* mode, data will be interpolated, while in *DIRECT* mode data will be used as is, without interpolation. Note that the *INTERPOL* mode induces a delay in the visualisation.

Refer to the *Real-time VTS* section in the *Synchronization protocol for VTS clients* chapter for further information.

Orientation modes

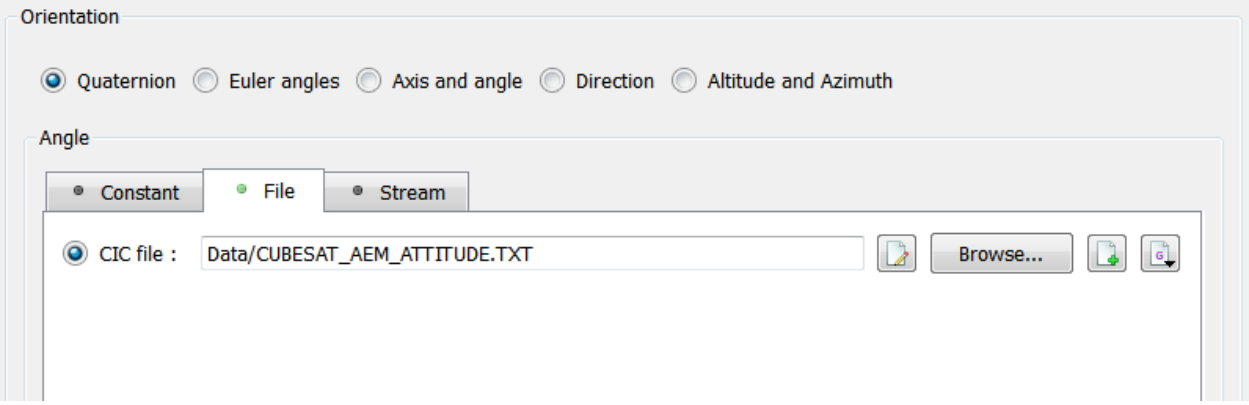


Fig. 5.8: Orientation defined by an ephemerides file

- Five radio buttons allow selecting the orientation mode of an entity: **Quaternion**, **Euler angles**, **Axis and angle**, **Direction**, **Azimuth and elevation**. Each of these orientation modes can be configured to use a fixed value, a sampled value in a CIC/CCSDS data file, or a value stream (as described above for the configuration of the position).

Orientation by quaternion

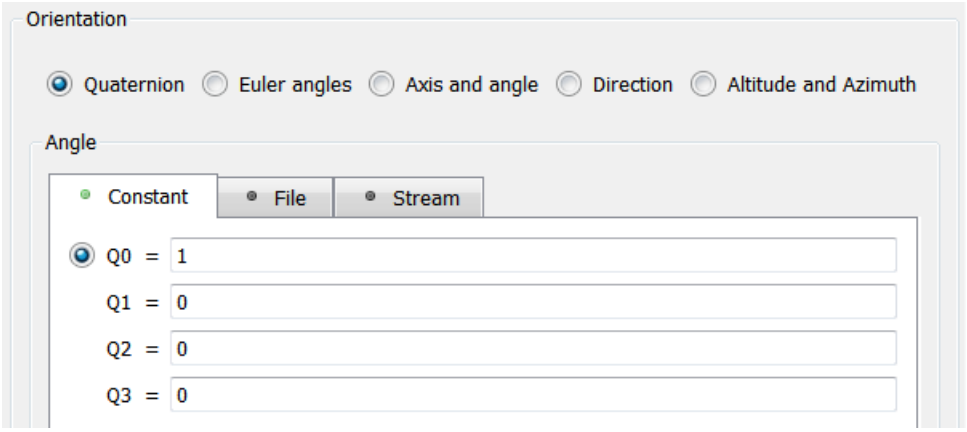


Fig. 5.9: Orientation by constant quaternion

- An orientation by quaternion is defined by its four components: **Q0**, **Q1**, **Q2** and '*Q3*'.

Orientation by Euler angles

- An orientation by Euler angles is defined by a sequence of three rotations as follows: **Z**, **X'**, **Z"**

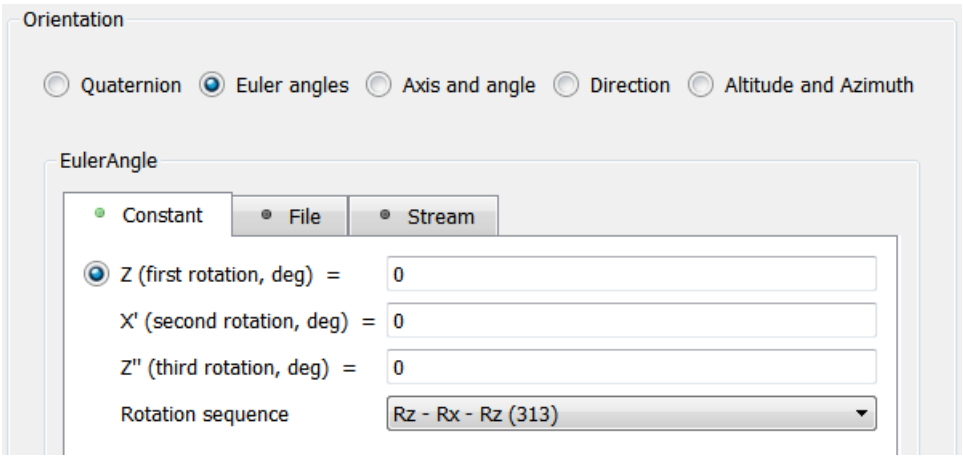


Fig. 5.10: Orientation by Euler angles

Orientation by axis and angle

- An orientation by axis and angle is defined by the direction vector of the axis and the angle of rotation (in degrees). Both can be defined independently.
- It is possible to define a fixed axis and a sampled/streamed angle, as well as a sampled/streamed axis and a fixed angle. However, it is not possible to define both a sampled/streamed axis and a sampled/streamed angle.

Orientation by direction

- An orientation by direction is defined by a direction vector in the entity's local frame. **This orientation mode does not strictly define an orientation**, there remains a degree of freedom around the direction of the vector. This mode offers a simple solution to define the orientation of a satellite component based for example on a file containing the direction of the Sun in the satellite's local frame.

Orientation by azimuth and elevation

- An orientation by azimuth and elevation is defined by two angles for the azimuth and elevation, both in degrees. **This orientation mode does not strictly define an orientation**, there remains a degree of freedom around the direction of aim.

Icon configuration

An **icon** can be configured in the following panel:

- **Anchor**: The anchor allows to modify the center of the icon. It can be useful with pinpoints icons
- **Size**: A predefined size from values between 9x9 pixels and 129x129 pixels. The values are odd so the positioning of the center pixel is accurate according to the real coordinates.
- **Opacity**: 0 = transparent, 100 = opaque
- **Layers: An icon can be composed by many layers. Each layer can be move up and down in the layer stack**
  - **Image layer**: Can be a default icon or a custom icon (can be imported from the catalog)

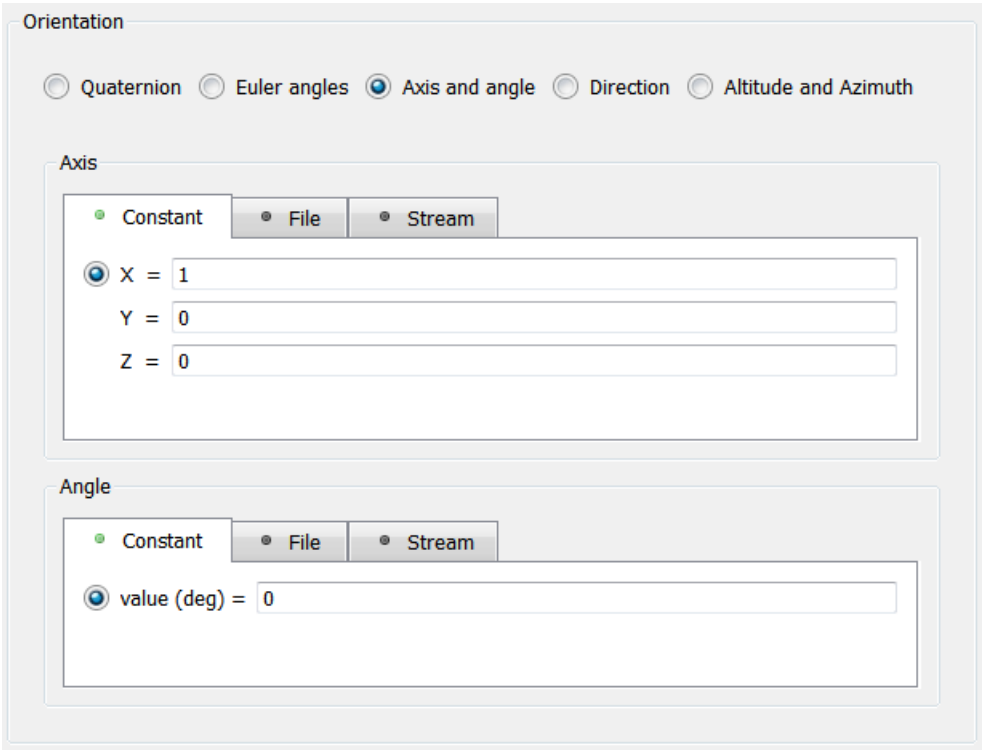


Fig. 5.11: Orientation by axis and angle

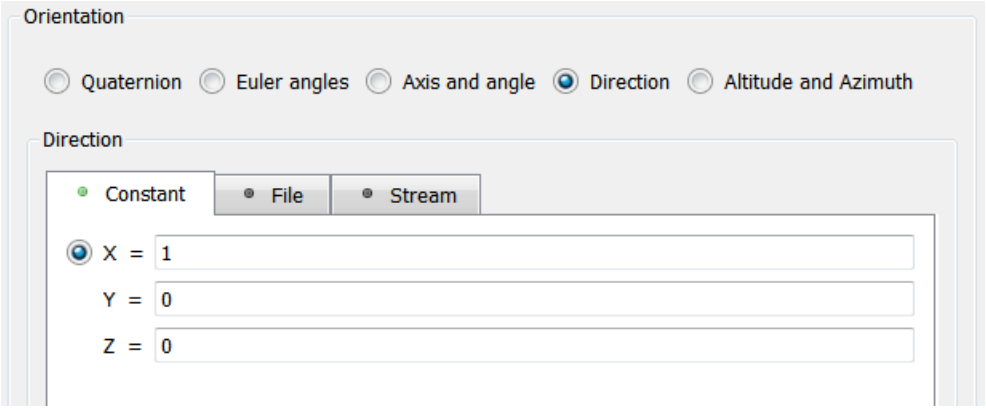


Fig. 5.12: Orientation by direction

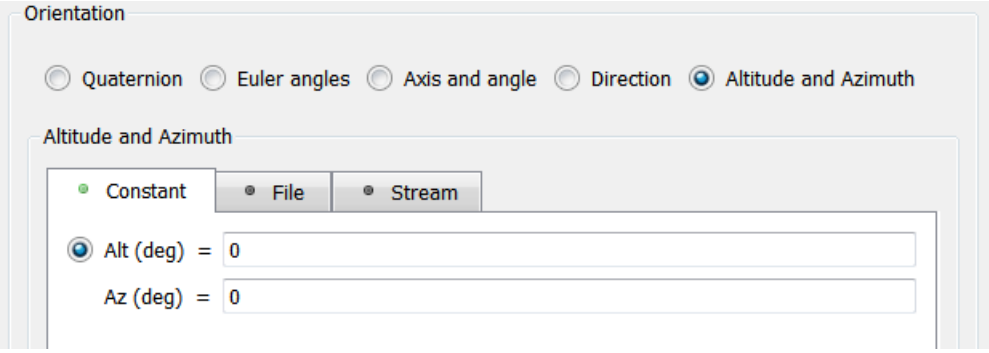
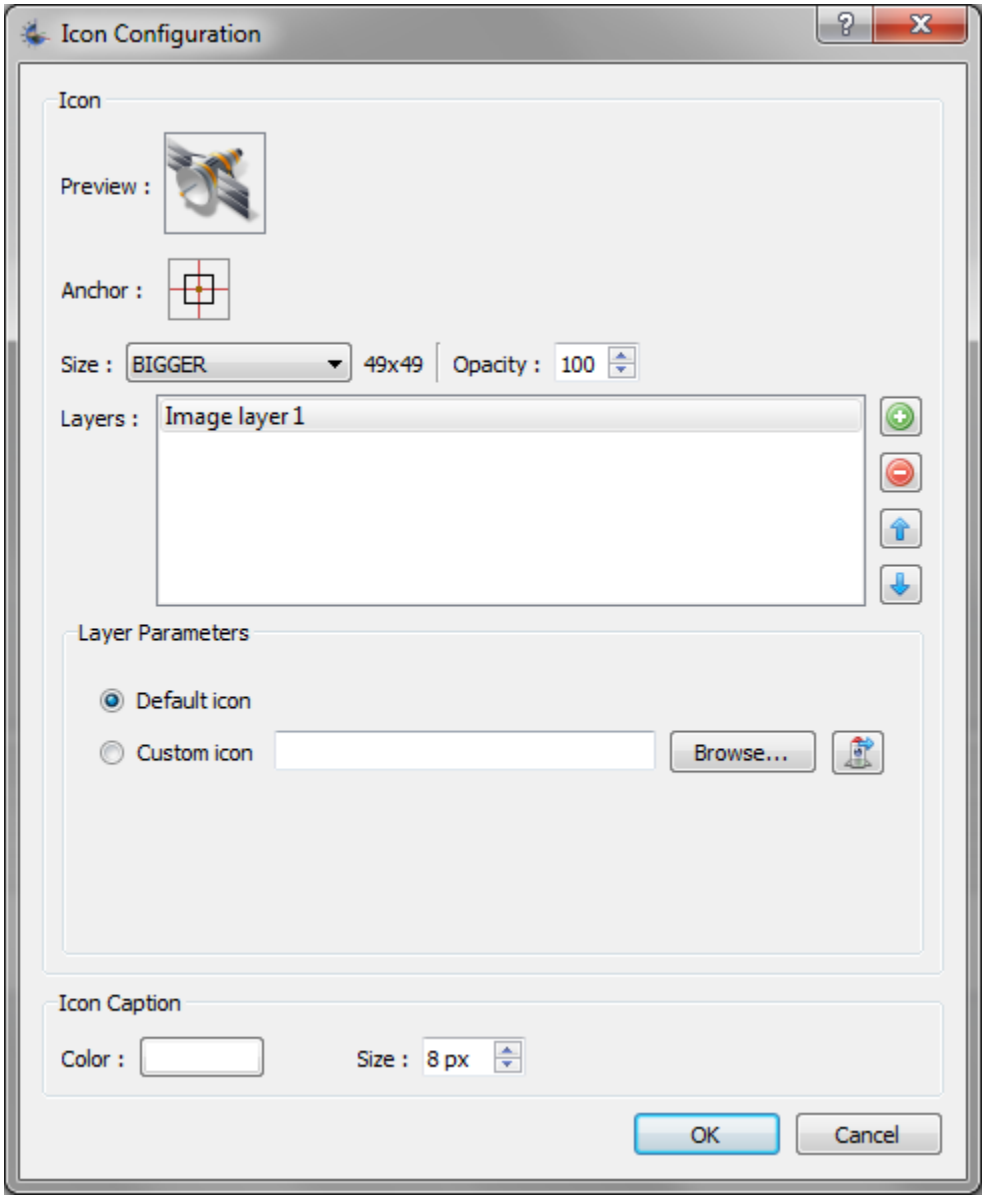
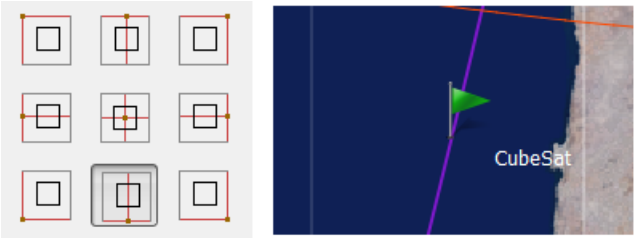


Fig. 5.13: Orientation by azimuth and elevation



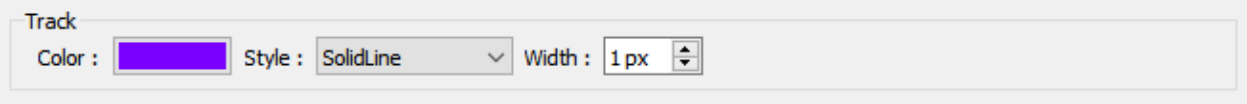


  - **Shape layer**: A predefined shape of which color and filling can be configured

The **icon caption** can be configured with a **color** and a **size** (limited to 100px).

Track configuration

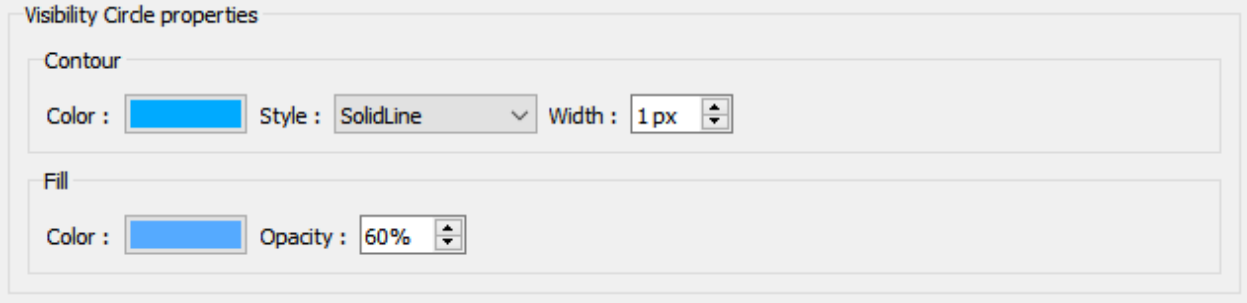
The ground **track** of satellites and celestial bodies can be configured in the following panel:



- **Color**: defines the color of the track.
- **Style**: defines how the track is traced (with dashes, with dots, . . .).
- **Width**: defines the thickness of the track.

Visibility circle configuration

The **visibility circle** of the satellite and celestial bodies can be configured in the following panel:

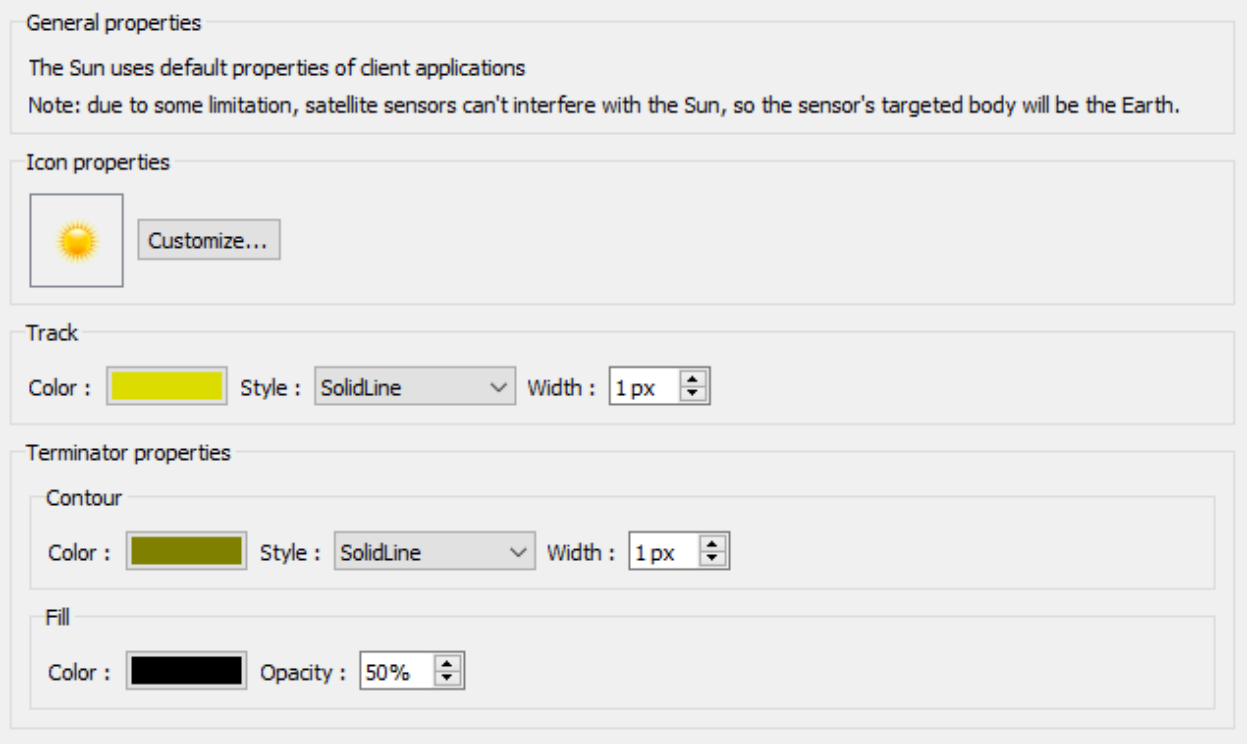


- **Contour: defines the contour of the visibility circle.**
  - **Color**: defines the color of the contour.
  - **Style**: defines how the contour is traced (with dashes, with dots, . . .).
  - **Width**: defines the thickness of the contour.
- **Fill: define the fill of the visibility circle.**
  - **Color**: defines the color of the fill.
  - **Opacity**: defines the opacity of the fill.

Configuring Sky

Configuring the Sun

The Sun uses default properties of client applications which means that properties such as its 3D model or texture cannot be modified. However, the icon, track and the terminator properties can be modified from the VTS Configuration utility.



Please refer to *Icon configuration*, *Track configuration* and *Visibility circle configuration* sections in order to configure these properties for the sun.

”Note that the **Sun terminator** properties are similar to those of a visibility circle although the sun terminator affects the opposite region (the obscured face of the body)”.

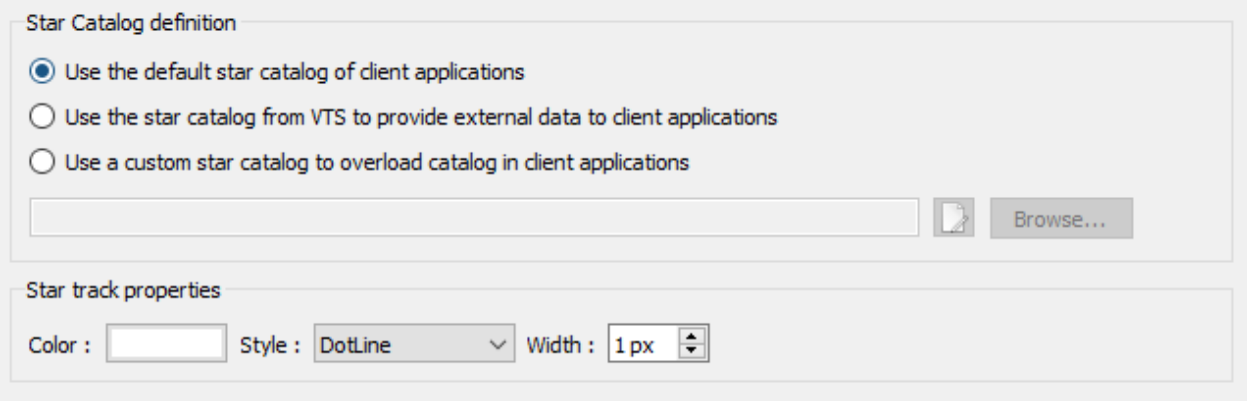
Configuring the Star Catalog

This panel allows you to define the specific star catalog that will be used in client applications. Note however that some client applications may still use their built-in star catalog, so please refer to the specific client application manual to check.

Star Catalog definition

There are three modes to load the star catalog:

- *Use the default star catalog of client applications*: This mode will notify client applications to use their built-in star catalog.



- *Use the star catalog from VTS to provide external data to client applications*: This mode will notify client applications to use the star catalog provided with VTS (see Hipparcos Catalog provided with VTS for more information).
- *Use a custom star catalog to overload catalog in client applications*: Allows to specify a custom star catalog file.

If the *custom* catalog mode is selected, you can specify your own star catalog file. Note that this file must be a valid CIC/CCSDS file and that it must have the same structure of the provided catalog (please see Hipparcos Catalog provided with VTS for information about the file structure).

Star Track properties

The graphical properties of the stars tracks can be configured for this Star Catalog. Please refer to *Track configuration* sections.

Hipparcos Catalogue provided with VTS

VTS provides the *Hipparcos Catalogue* in a CIC/CCSDS format. The provided catalog file has the following columns:

- **HIP**: Identifier (HIP number).
- **RAdeg**: Right ascencion in degrees.
- **DEdeg**: Declination in degrees.
- **Vmag**: Magnitude in Johnson V.

The star catalog file provided with VTS has the following characteristics:

- Stars are sorted by magnitude.
- Stars without RAdeg/DEdeg have been removed.

Note: the provided star catalog was created based on the VizieR catalogue access tool. Please refer to this site for more information.

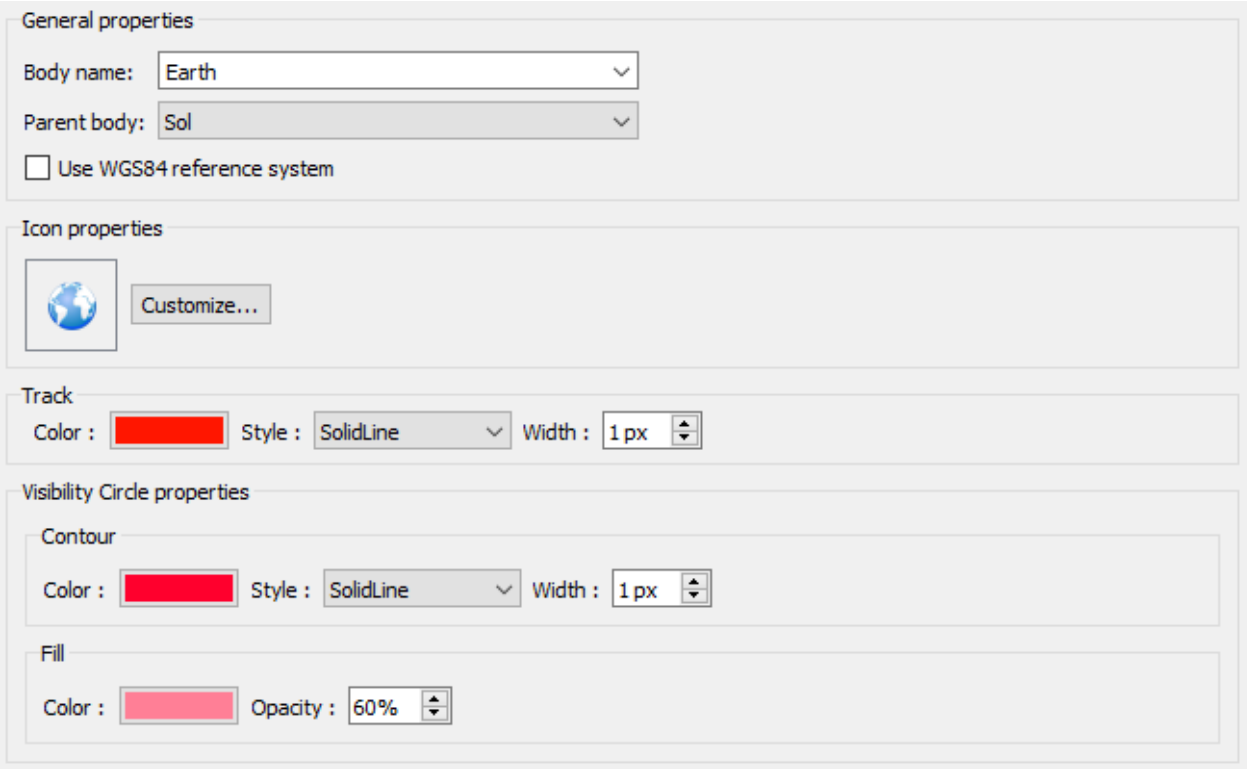
Configuring a central body

Missions visualized in VTS are defined around central bodies. A central body can be a planet or an object orbiting the Sun. A spacecraft is always attached to a central body. At least one central body must be defined for the VTS project to be valid. New projects use the Earth as default central body.

General properties of a body

The general properties of a body are its:

- Name
- Parent body name
- UseWGS84 reference system : in this mode, the flattening (ellipticity, or oblateness) of the body is taken into account. For the moment, only SurfaceView viewing Earth handles this configuration.
- Icon
- Ground Track
- Visibility circle over other celestial bodies.



- If the central body has a valid Celestia body name, its default ephemerides can be used or overridden (refer to the *Position and orientation of a body* section below). A drop-down list allows choosing a body name amongst the planets of the solar system. The planet's radius is automatically filled in in the 3D properties of the body.
- A body not known to Celestia can be defined. However, it will then be necessary to provide user-defined 3D properties and ephemerides.

Other properties of a body

Refer to *Icon configuration*, *Track configuration* and *Visibility circle configuration* sections in order to configure these properties for a central body.

Textures of a body

The textures of a body defines its appearance in the 2D and 3D views. A body can have many textures (zero or more) organized as stacked layers.

Notes:

- User-defined textures must correspond to a plate carrée projection, i.e. with a 1:2 ratio and centered on longitude 0°.
- Due to limitations in Celestia, only one central body may used the *Fixed*, *Timed* or *Cyclic* texture types in the project.

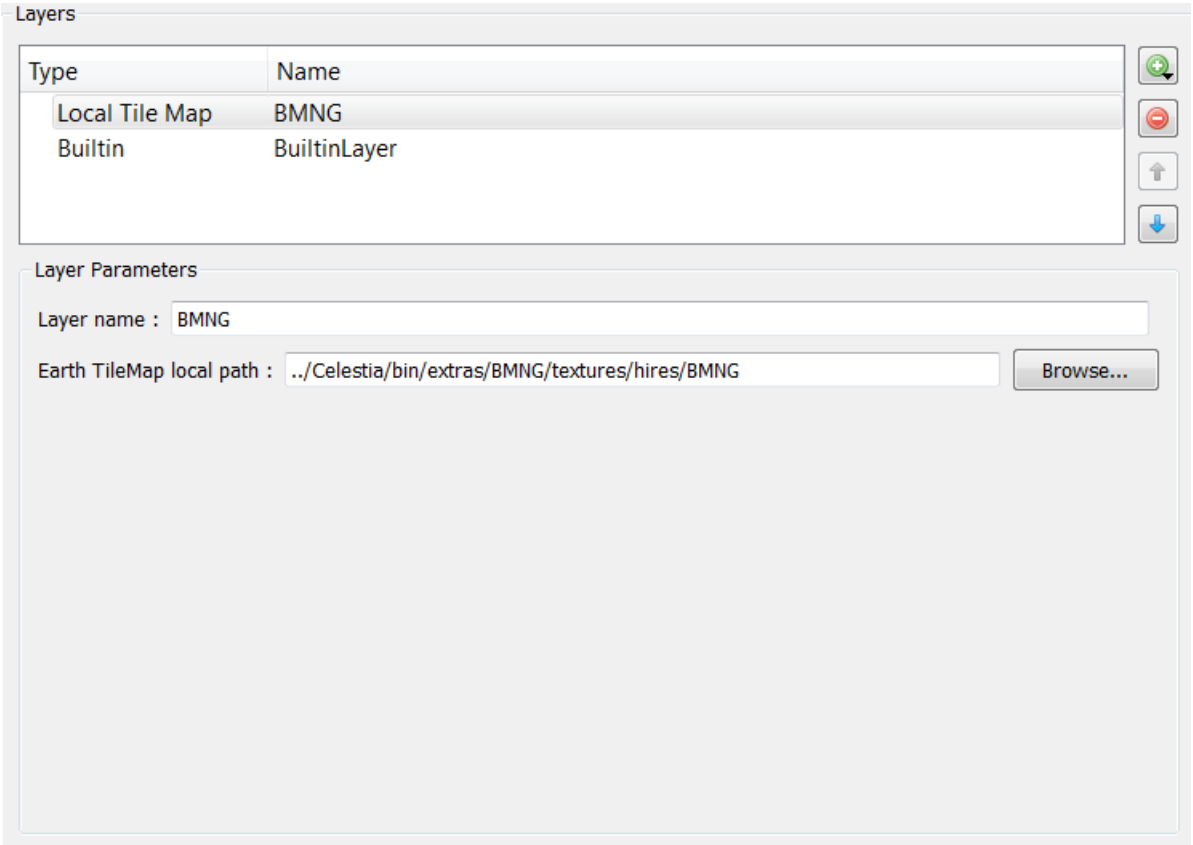
Body layers

A body's layers list can be modified through its layers property page. The layers list is shown on the upper-part of this view. Selecting a layer in this list displays the layer's properties on the lower-part of the view (according to the layer's type).

A layer can be added without the menu displayed by the **+** button.
 A layer can be removed by selecting it in the list and clicking on the **-** button.

The layers are organized as a stack. Therefore, the top-most layer in the list will be the top-most layer of the body. As a consequence, if a layer covers (without transparency) the whole body, then the layers located under it will never be visible. The ordering of the layers can changed bo selecting a layer in the list, and clicking on the **up** and **down** buttons.

In order to be recognized, each layer has a name that must be unique within a body.



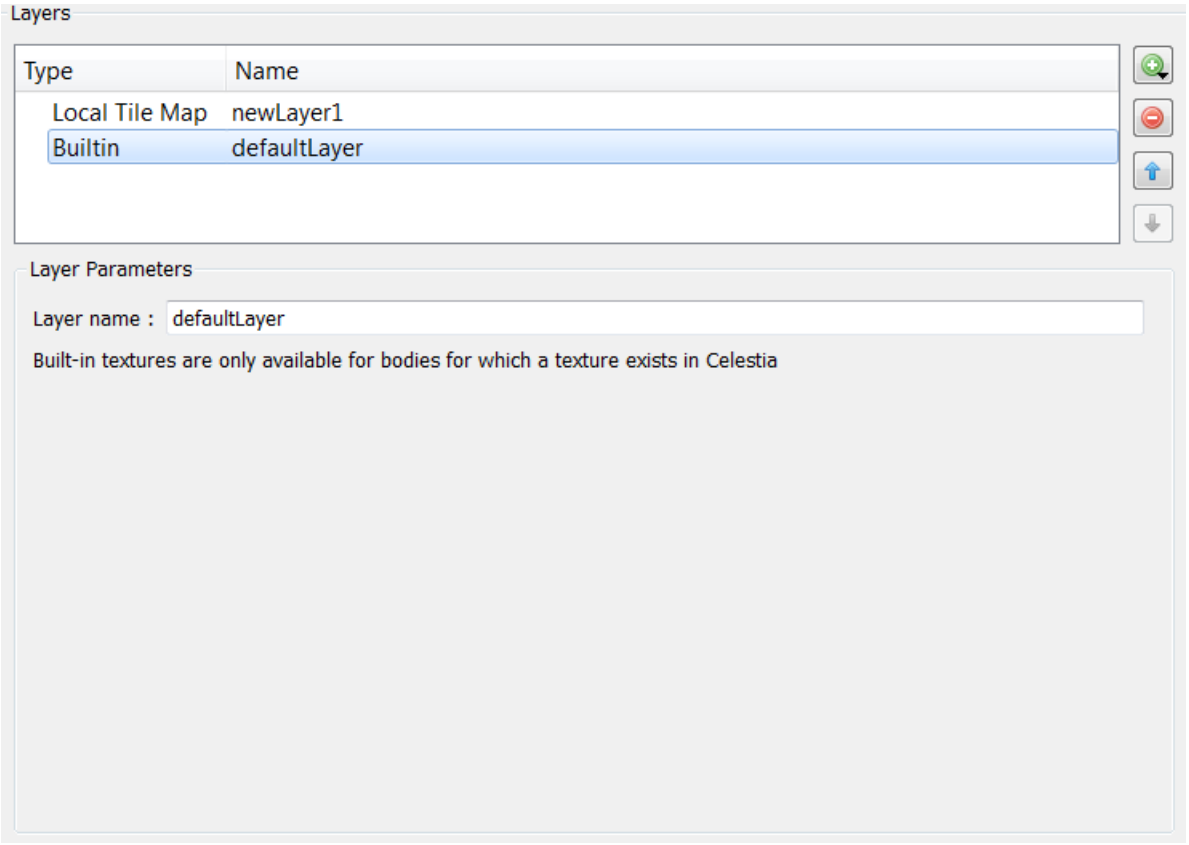
New bodies come by default with a **built-in** layer, which can be removed if wanted.

Texture types

A texture can be of 6 different types, selected by the **new layer** button: **Built-in**, **Fixed**, **Timed** (**acyclic** or **cyclic**) and **Tile Map** (**local** or **WMS**).

Built-in texture

The texture is provided by Celestia, if available. Celestia has built-in textures for all planets of the solar system and some of their natural satellites.

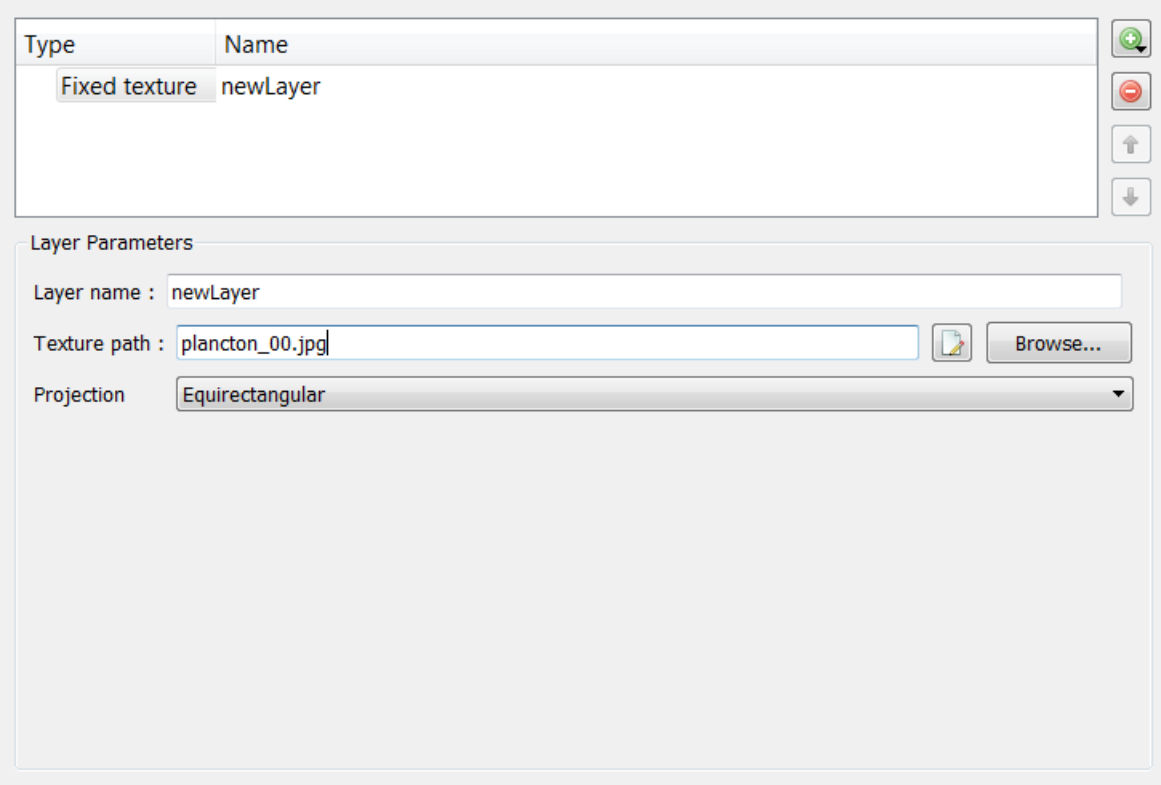


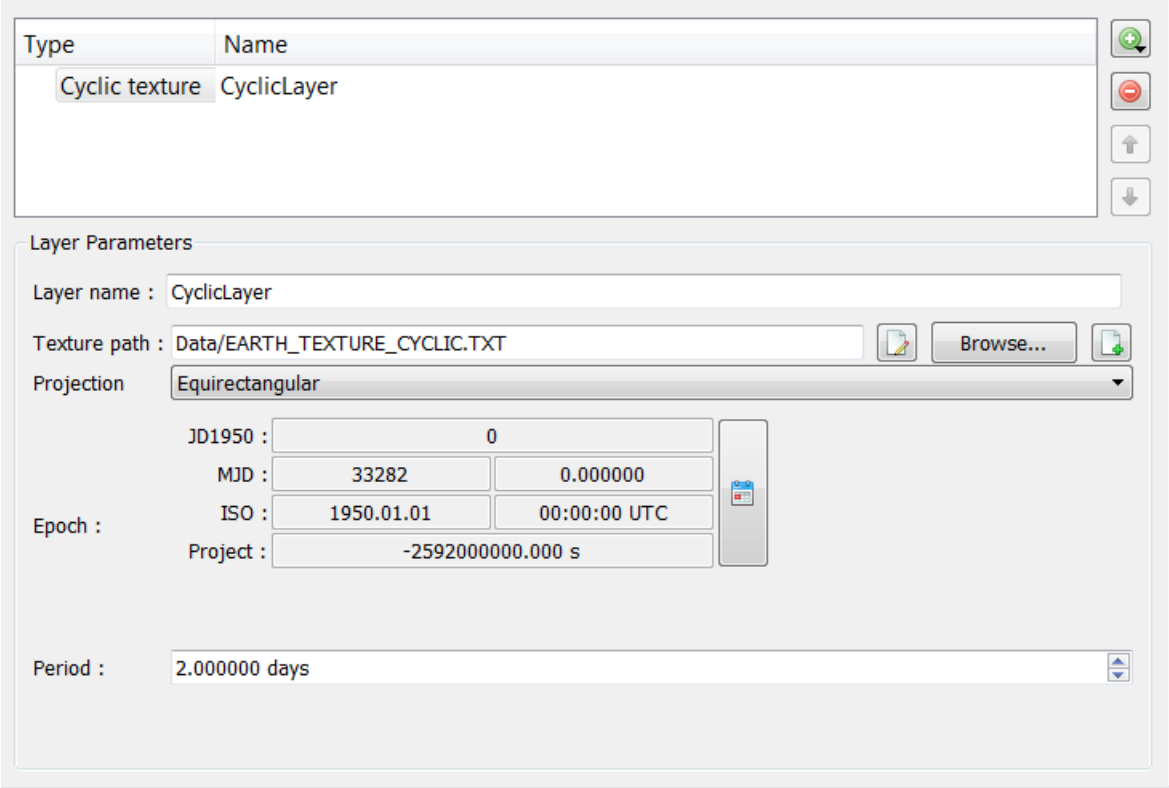
Fixed texture

The texture is defined by an image file in the project folder.

Timed texture

The texture is defined through a CIC/CCSDS file in the MEM format. This file contains the relative path (in the project folder) to the texture to use with regards to the current visualization date.


Type
Name
Fixed texture
newLayer
Layer Parameters
Layer name :
newLayer
Texture path :
plancton_00.jpg
Browse...
Projection
Equirectangular


Type
Name
Cyclic texture
CyclicLayer
Layer Parameters
Layer name :
CyclicLayer
Texture path :
Data/EARTH_TEXTURE_CYCLIC.TXT
Browse...
Projection
Equirectangular
Epoch :
JD1950 :
0
MJD :
33282
0.000000
ISO :
1950.01.01
00:00:00 UTC
Project :
-2592000000.000 s
Period :
2.000000 days

The MEM timed texture file, containing *STRING* data of dimension *1*, must be defined for all timed textures (cyclic and acyclic).

Notes:

- If a line of the MEM file contains the string *DEFAULT*, the built-in texture of the body is used for this date

Beware, using a timed texture can make the visualization jerky, especially when using Celestia. The following circumstances may cause performance issues:

- timed texture with frequent texture changes (every few seconds of wall-clock time)
- timed texture with textures of significant sizes
- timed texture visualized at high time ratio (causing texture changes every few seconds of wall-clock time)

Acyclic timed texture

For an acyclic texture, dates in the MEM file are the dates at which the texture changes occur. For example, the following line:

```
56018 0 "Textures/Chl_a_april 2012.png"
```

specifies that image file *Textures/Chl_a_april 2012.png* is to be used as texture for the central body at MJD date *56018 0*.

Cyclic timed texture

For a cyclic texture, dates in the MEM file describe the instants of the cycle at which texture changes occur. Two additional parameters are required:

- The epoch of the cycle
- The period of the cycle in fractional days

The epoch of the cycle must be prior to the start date of the project. Periods of an variable number of days, such as a month or a year, may not be specified. The best solution in this case is to use an acyclic timed texture with texture changes adjusted for the project.

Dates in the MEM file are interpreted as durations from the start of the cycle. This means that a timed texture MEM file must contain dates in the MJD date format. For example, the following lines:

```
0     0 "Textures/plankton_00.jpg"
0  7200 "Textures/plankton_01.jpg"
```

specify that the image file *Textures/plankton_00.jpg* is to be used as texture at the start of a cycle, and that the image file *Textures/plankton_01.jpg* is to be used as texture two hours after the start of a cycle.

Beware, if MJD date *0 0* is not present in the MEM file, no texture will be displayed at the start of a cycle.

Tile Map textures

The tilemap mechanism allows you to load and display high definition maps as background images, depending on the zoom level. The tilemap can be configured either by a local URL pointing to a multi-level image hierarchy or a WMS server.

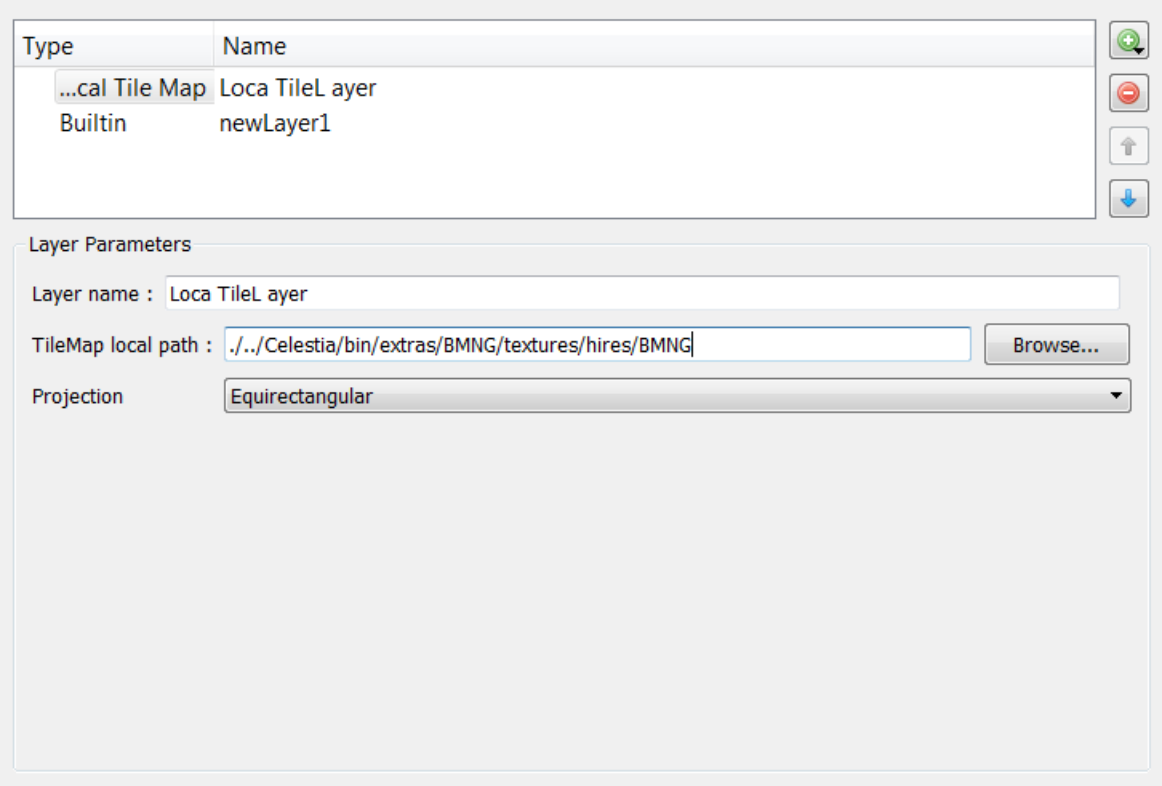
Local tilemap

The local tilemap mechanism is compatible with Celestia. Some examples are provided with the level 0 and level 1 tiles :

- ../Celestia/bin/extras/BMNG/textures/hires/BMNG
- ../Catalog/Maps/OpenStreetMap

Note that the path is relative to the application directory.

These default textures were obtained from NASA's Blue Marble Next Generation.



Textures directory

A tilemap texture directory should can contain up to 15 tilemap levels, each level corresponding to a directory named as `levelN` where `N` is the level number (from 0 to 14).

Each level directory contains tile texture files named `tx_<C>_<R>.<ext>` where <C> is the column index, <R> is the row index and <ext> the image file extension (many formats supported by Qt). A tile must not necessary be complete, it can contain just a few files corresponding to a region close-up.

The total texture count of a level can be calculated as 2^{2n+1}.

Creating custom tilemap

The method used for creating textures in Celestia can also be used in SurfaceView :

- Creating Textures for Celestia

- Texture Virtuelle (french)

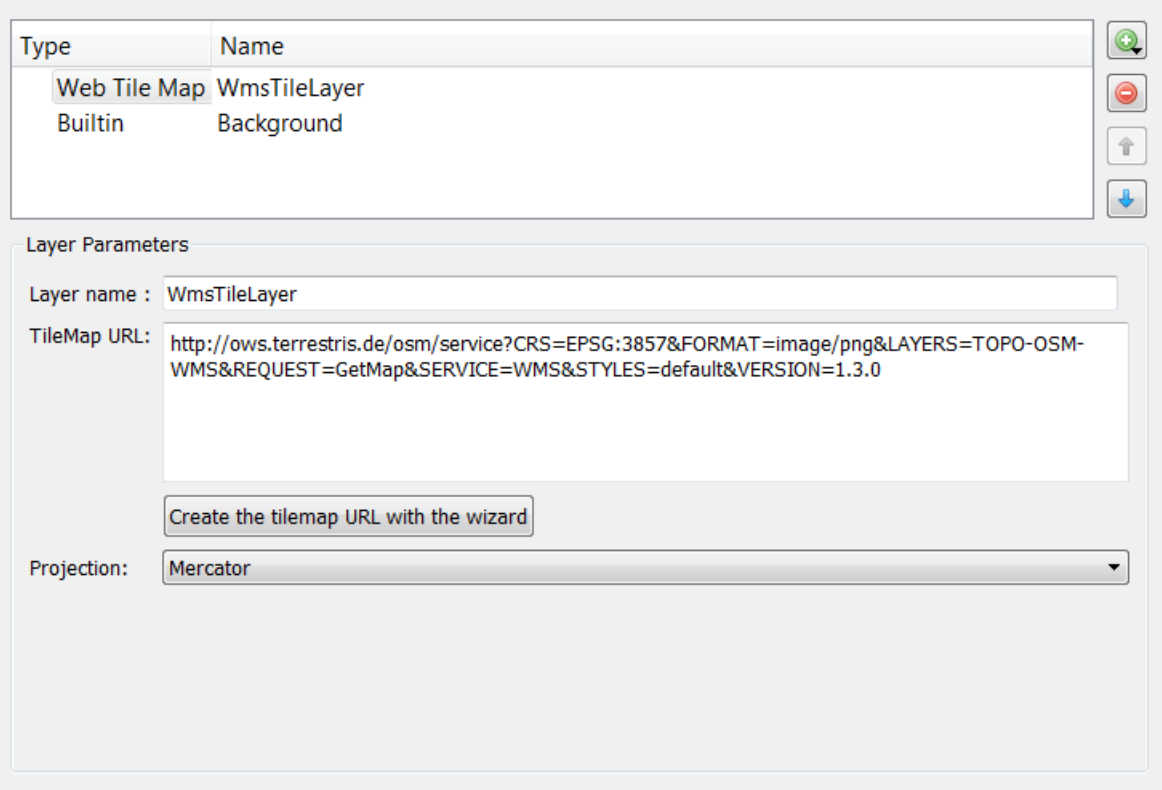
Using Celestia Virtual Textures

SurfaceView is compatible with Celestia's Virtual Texture mechanism, all existing virtual textures for Celestia can be used with SurfaceView :

- Earth textures
- Earth Close-Ups

Using a Web Tile Map service

Tilemaps can also be retrieved from a Web Map Service (WMS) or a Web Tile Map Service (WMTS). *Some WMS formats might not be supported yet.* Currently, the following coordinates reference system are well-supported: `EPSG:4326` (equirectangular, WGS84), `EPSG:3857` (Mercator, WGS84). Please refer to epsg.io in order to get information about these CRS.

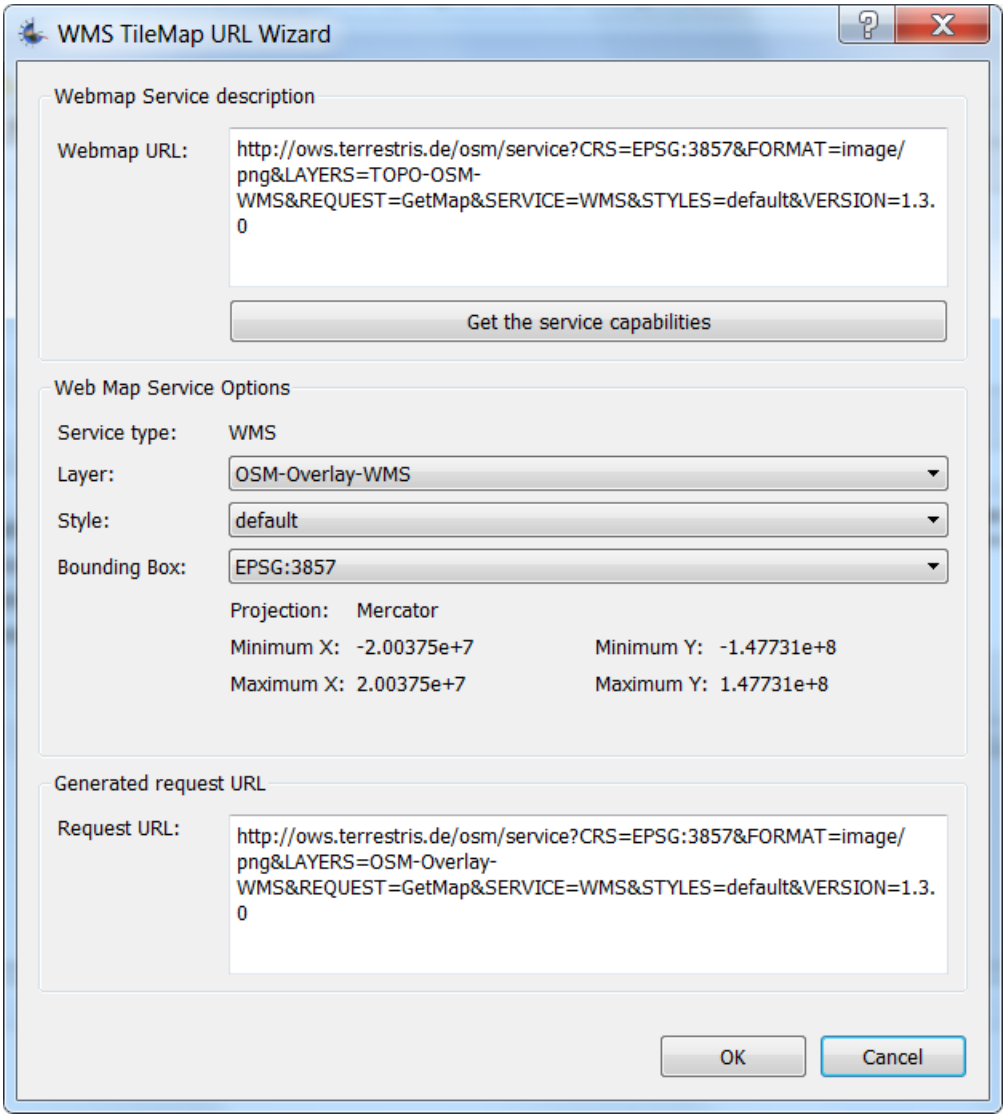


The specified URL must be well-formatted and must correspond to a GetMap (WMS) or GetTile (WMTS) request. You can either write your own URL, or use the TileMap URL Wizard as shown below.

You can access the TileMap URL wizard through the "Create the tilemap URL with the wizard" button.

The wizard is splitted into three parts, vertically:

- The **Webmap URL**: this is the root URL that corresponds to the server URL. It must include the full url root and the service type (WMS or WMTS) requested. It can include supplementary parameters not related to the service: these parameters will be kept. For example:


WMS TileMap URL Wizard
Webmap Service description
Webmap URL:
http://ows.terrestris.de/osm/service?CRS=EPSG:3857&FORMAT=image/png&LAYERS=TOPO-OSM-WMS&REQUEST=GetMap&SERVICE=WMS&STYLES=default&VERSION=1.3.0
Get the service capabilities
Web Map Service Options
Service type:
WMS
Layer:
OSM-Overlay-WMS
Style:
default
Bounding Box:
EPSG:3857
Projection:
Mercator
Minimum X:
-2.00375e+7
Minimum Y:
-1.47731e+8
Maximum X:
2.00375e+7
Maximum Y:
1.47731e+8
Generated request URL
Request URL:
http://ows.terrestris.de/osm/service?CRS=EPSG:3857&FORMAT=image/png&LAYERS=OSM-Overlay-WMS&REQUEST=GetMap&SERVICE=WMS&STYLES=default&VERSION=1.3.0
OK
Cancel

http://server.arcgisonline.com/arcgis/rest/services/ESRI_Imagery_World_2D/MapServer/WMTS?SERVICE=WMTS

- **The Web Map Service Options: this part of the wizard is accessible by clicking on the Get the service capabilities. The opt**
  - Layer: all the layers supported by VTS
  - Style: all the available styles for the specified layer.
  - **Bounding Box / Tile Matrix Set: all the available bounding boxes for the specified layer. After selecting a boundi**
    - The projection type (equirectangular or mercator).
    - The bounding box boundaries (expressed in the CRS units).
- The **Generated request URL**: this field is read-only and corresponds to the generated URL, resulting from the chosen options.

When you have configured the different options, you can click the OK button: the generated URL will be transferred to the layer's tilemap URL, as well as the projection type.

If you want to write your own URL, you need to respect the following rules :

- The URL must correspond to a GetMap (for WMS) or GetTile (for WMTS) request.
- The requested FORMAT **must be** `image/png`. No other format is supported as for now.
- The URL must tell the service type that is requested through the `SERVICE` parameter, values can be `WMS` or `WMTS`.
- The URL must provide any necessary information about the requested information: layer name, CRS/SRS name, style name, bounding box (for WMS), tile matrix set (for WMTS).

You can use the `GetCapabilities` request to get information about the service. Example:

http://ows.terrestris.de/osm/service?service=WMS&request=GetCapabilities

Then create a `GetMap` request defining an equidistant projection of the whole world, often `EPSG:4326` in WGS 84 (World Geodetic System), the image format, a SurfaceView compatible protocol version (To Be Defined).

*Please note that WMS tiles are stored in a local cache directory. This cache can be clean using the **Clear all client application data caches** button in the Settings menu of the VTS configuration utility*.

Example using an OpenStreetMap based WMS

The `GetMap` request :

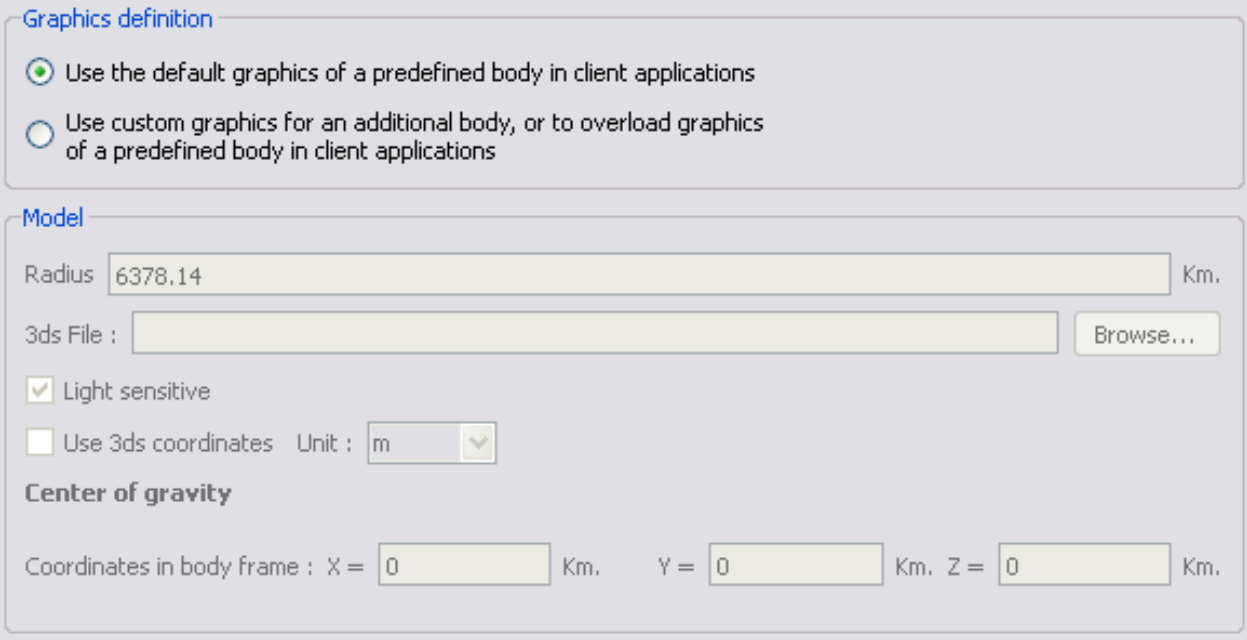
http://ows.terrestris.de/osm/service?service=WMS&request=GetMap&CRS=EPSG:4326&format=image/png&VERSION=1.3.06&styles&LAYERS=OSM-WMS

3D properties of a body

The 3D properties of a body define its appearance in 3D views.

The *Graphics Definition* radio buttons select the 3D properties to use for the body:

- default: use the graphics properties available if Celestia, if any.



- custom: redefine the 3D mesh of an existing Celestia body or define the 3D mesh of a body not available in Celestia. The parameters are similar to those described in the *3D properties of a satellite* section. Values are expressed in kilometers.

All custom 3D properties field must be filled in when redefining the 3D mesh of an existing Celestia body.

Custom 3D properties must be selected and a 3D mesh must be defined if the current body is not available in Celestia.

Position and orientation of a body

The VTS toolkit is mainly geared towards the visualization of missions around central bodies of the solar system. In general, the ephemerides of these bodies are known. However it remains possible to override the ephemerides of predefined bodies, or to define the ephemeris of a non-default body.

The *Ephemeris Definition* radio buttons select the position and orientation to use for the body:

- Default ephemeris mode : client applications choose their preferred ephemeris origin
- Catalog ephemeris mode : client applications use VTS catalog ephemeris files
- Custom ephemeris mode : the user provides the ephemeris for the body as described in the *Position and orientation of an entity* section.

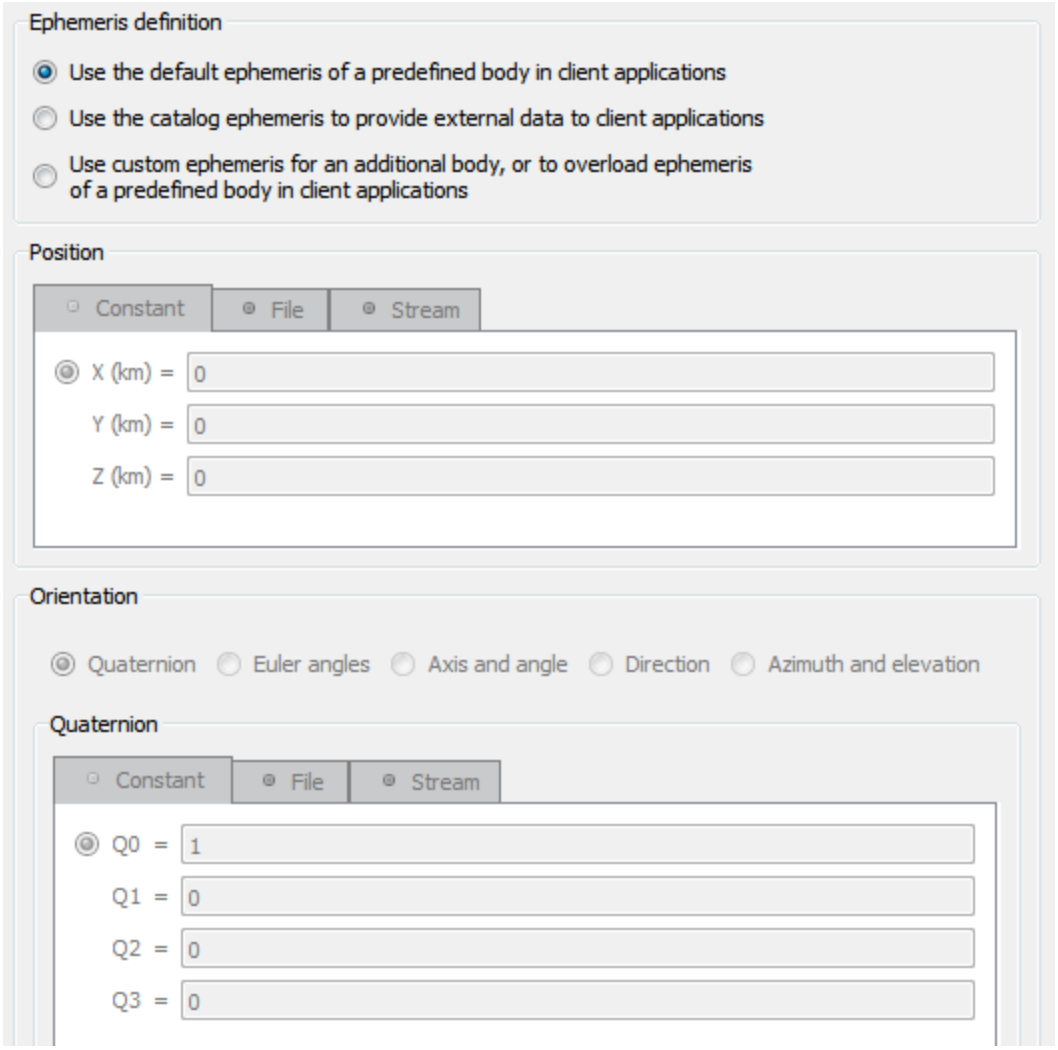
See the *Central bodies in VTS* section for more details.

Configuring a satellite

A satellite is composed of a main component and sub-components. General, 3D, and geometric properties are configured at the component level. Common properties are configured only for the main component.

General properties of a satellite

The general properties of a satellite are its:


Ephemeris definition
Use the default ephemeris of a predefined body in client applications
Use the catalog ephemeris to provide external data to client applications
Use custom ephemeris for an additional body, or to overload ephemeris of a predefined body in client applications
Position
Constant
File
Stream
X (km) = 0
Y (km) = 0
Z (km) = 0
Orientation
Quaternion
Euler angles
Axis and angle
Direction
Azimuth and elevation
Quaternion
Constant
File
Stream
Q0 = 1
Q1 = 0
Q2 = 0
Q3 = 0

- Name
- Central body
- Icon
- Orbit path
- Visibility circle over other celestial bodies
- Eclipse circle over the attached body

Name and central body of a satellite

The name of the satellite will be the satellite's unique identifier during visualization, used to identify the satellite in view properties and messages between the Broker and client applications.

The central body of the satellite defines the reference frame for the satellite, in which the satellite's position is expressed.

Orbit path of a satellite

The orbit path of a satellite is displayed in the 2D and 3D views. It has the same properties as a *track*. But it is also possible to colorize the satellite's orbit path with a CIC/CCSDS color file.

Note: the displayed length of the satellite's orbit path is now configured in the *project scenario*.

Note: Color files format must meet the following requirements:

- MEM format with 3 (or 4) real fields ranging from 0 to 1, for the RGB components (and transparency)
- MEM format with 3 (or 4) integer fields ranging from 0 to 255, for the RGB components (and transparency)
- MEM format with 1 string field in HTML color format ("#RRGGBB" in hexadecimal), for the RGB components (no transparency)

Eclipse circle of a satellite

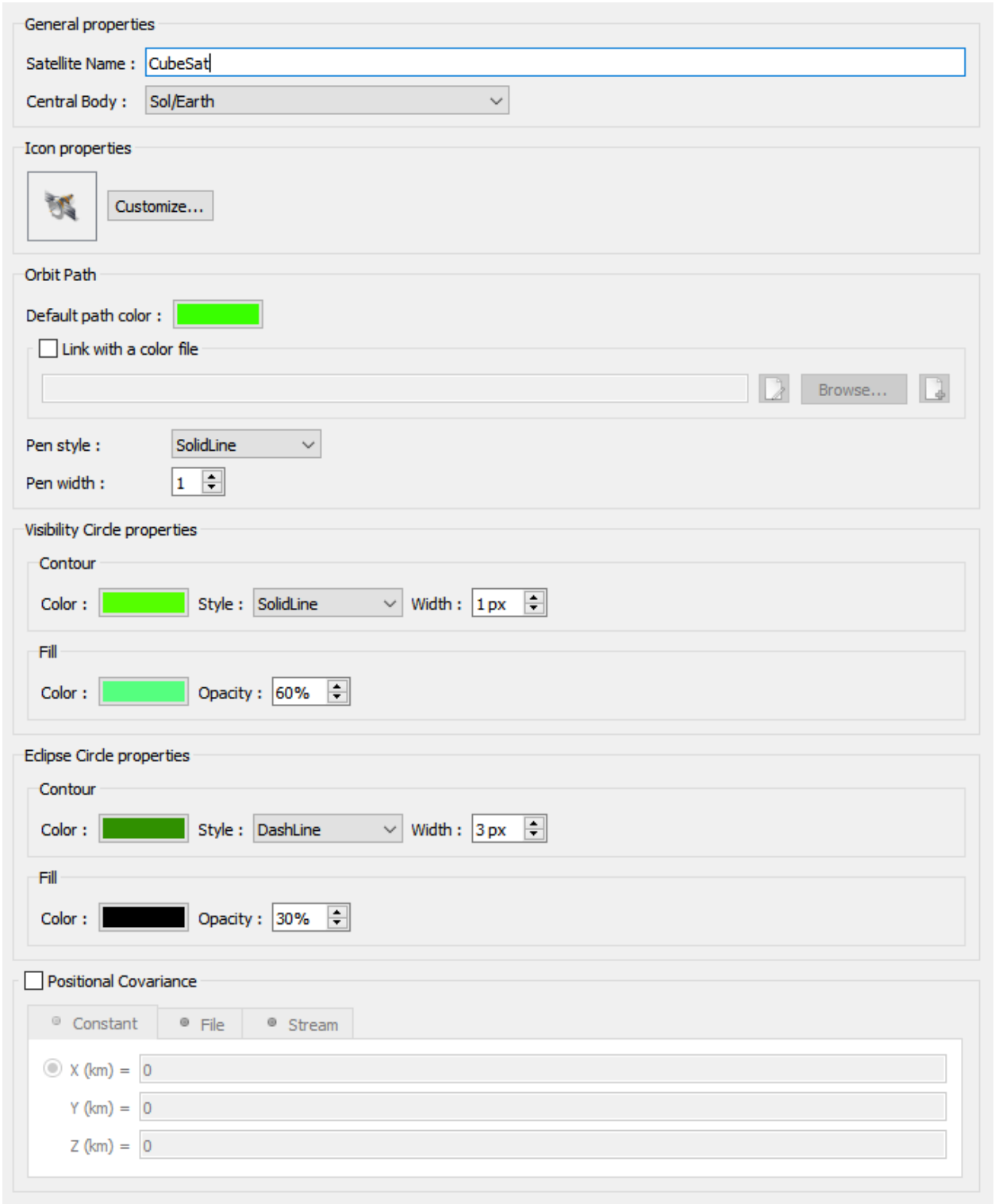
The eclipse circle of a satellite is the projection on its parent body of the area where the satellite is in the umbra cone of its parent body at a steady altitude.

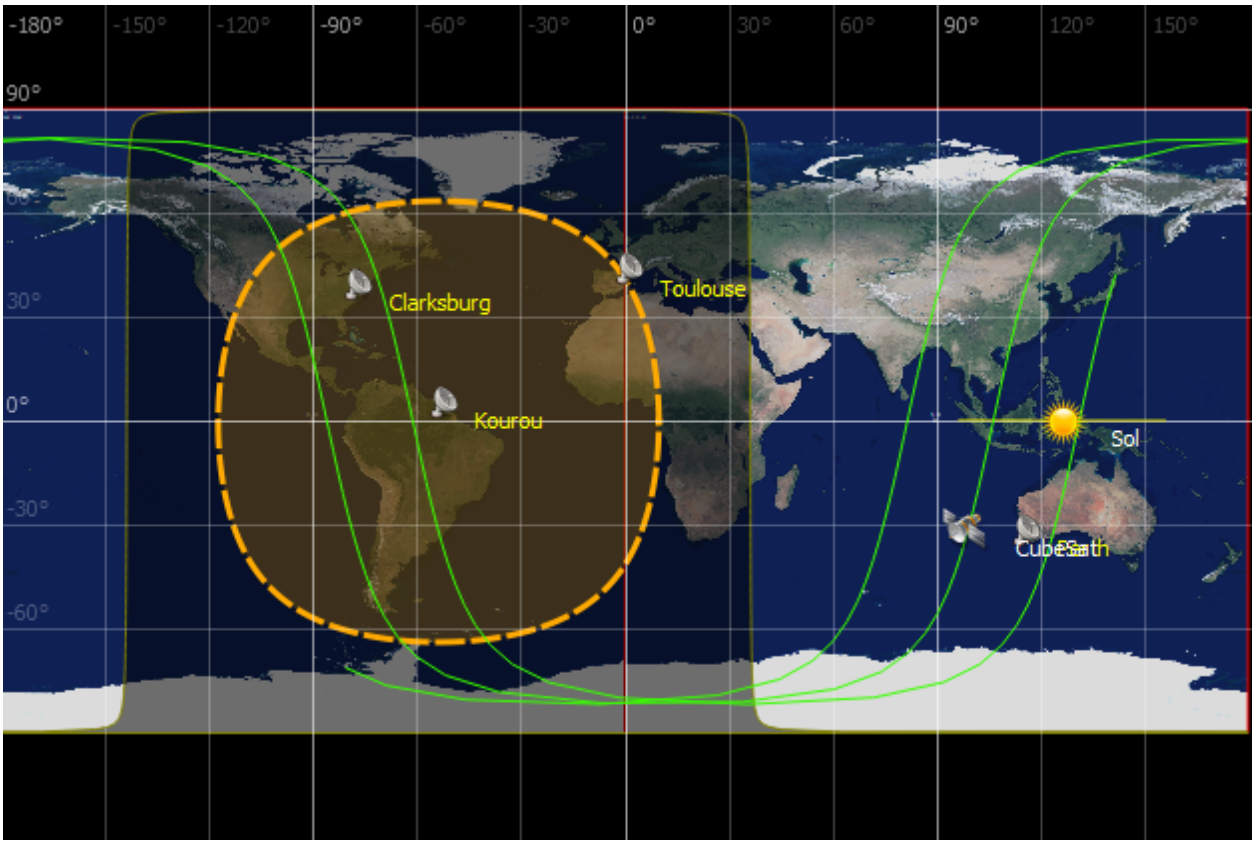
This can be visualised in SurfaceView by enabling its visibility in its configuration in the SurfaceView configurator (available in the Scenario Editor tab)

Position from a file

For a Satellites, the **New CIC file...** button allows you to select the type of input for the position of the satellite. Two reference frames are supported :

- EME2000 (Earth Mean Equator at epoch J2000)
- Central body's local frame (only with file MEM with lat/long/alt)


General properties
Satellite Name : CubeSat
Central Body : Sol/Earth
Icon properties
Customize...
Orbit Path
Default path color :
Link with a color file
Browse...
Pen style : SolidLine
Pen width : 1
Visibility Circle properties
Contour
Color : Style : SolidLine Width : 1 px
Fill
Color : Opacity : 60%
Eclipse Circle properties
Contour
Color : Style : DashLine Width : 3 px
Fill
Color : Opacity : 30%
Positional Covariance
Constant
File
Stream
X (km) = 0
Y (km) = 0
Z (km) = 0



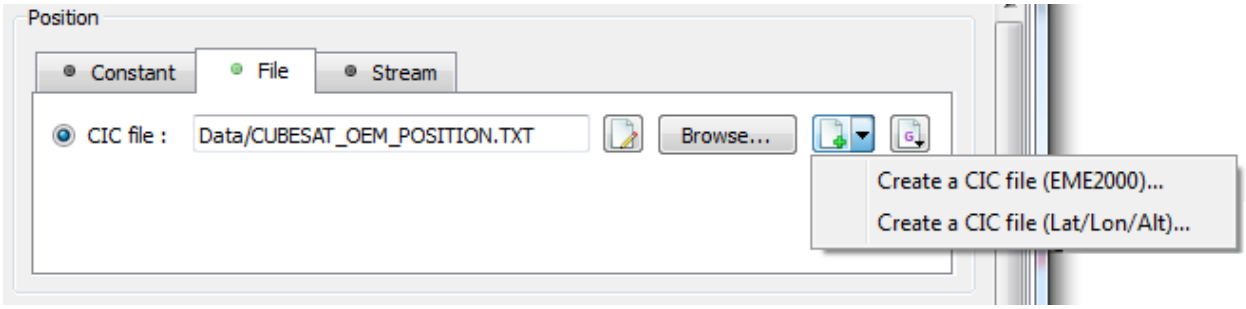


Fig. 5.14: Position of a Satellite defined by an ephemerides file

Other properties of a satellite

Refer to *Icon configuration* and *Visibility circle configuration* sections in order to configure these properties for a satellite.

Events attached to a satellite

Mission events can be attached to the satellite to be displayed in the project timeline and in compatible client applications. These events are provided in CIC/CCSDS files in MEM format.

- The **Add...** button allows selecting an additional CIC/CCSDS event file
- The **Create...** button allows creating an empty CIC/CCSDS event file
- The **Edit file...** button on each line opens the selected CIC/CCSDS event file in a text editor
- The **Remove file** button on each line removes the selected CIC/CCSDS file from the list

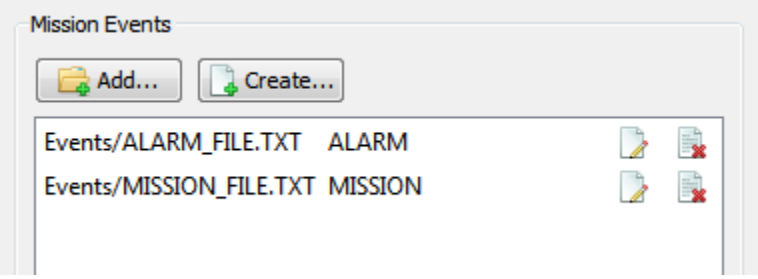


Fig. 5.15: List of mission event files

Each file has its top-level event type displayed next to its name on its line. This event type is used to construct the full event type name for events of the file.

The decoration of all events can be controlled in the *Event Type Editor* tab of the VTS configuration utility (see the *Configuring event types* section). The visibility of the events can be controlled during visualization through the Broker (see the *Events* tab section in the *Broker user manual*).

Refer to the *Mission events in VTS* chapter for more information on events in VTS.

3D properties of a satellite

The 3D properties of a satellite define its appearance in 3D views.

- **The Approximate size field only applies to the current component (3ds file). It has two meanings:**
  - If the '*Use 3ds coordinates*' option is enabled, it defines the bounding sphere of the 3D mesh to be used by clipping mechanisms. This parameter should be set to a realistic value.
  - If the '*Use 3ds coordinates*' option is disabled, it defines the actual dimensions to be applied to the 3D mesh of the component.
- The '*3ds File*' field defines the 3ds mesh file to use for the component. It must be point to an AutoDesk 3DS Max file. If the selected file is not located in a sub-folder of the project folder, a dialog will offer to copy it inside the project folder.
- The **Light sensitive** option enables or disables shading of the 3D mesh. Objects such as satellite bodies or solar arrays usually should be shaded, while abstract objects such as frame axes or vectors should not. Unshaded

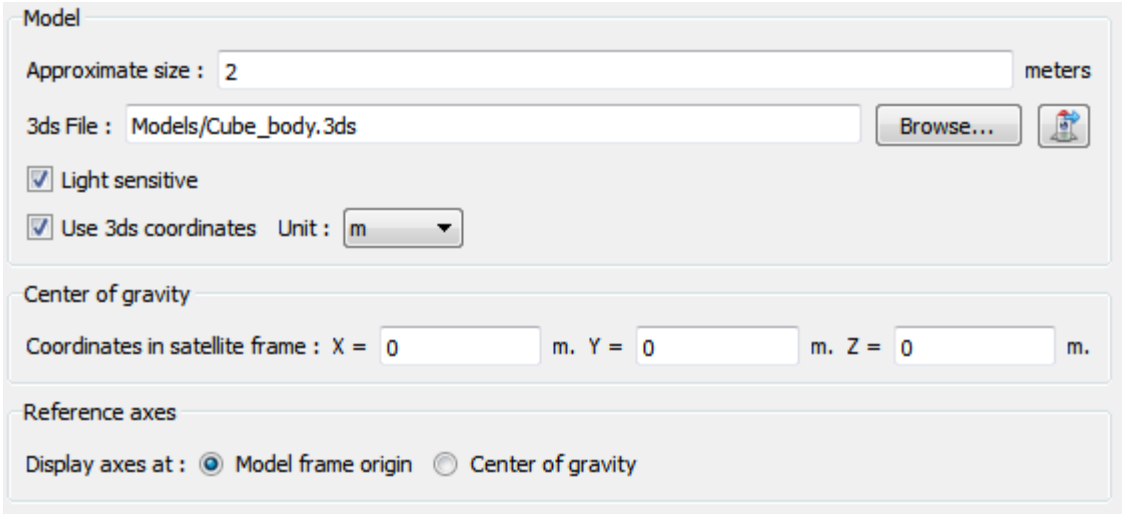


Fig. 5.16: 3D properties of a satellite

objects are only lighted in a diffuse manner, and do not show specular reflections from light sources such as the Sun.

- The '*Use 3ds coordinates*' option allows using the intrinsic coordinates of the 3ds mesh file. If the 3ds mesh file originates from a known CAD source, it is advised to leave the option enabled and to set the **Approximate size** field accordingly to define a bounding sphere for the 3D mesh. If the 3ds mesh file originates from an unknown source, it is advised to disable the option and to set the **Approximate size** field to the desired size for the 3D mesh. The **Unit** drop-down list defines the unit to use when reading coordinates in the 3ds mesh file.
- The **Center of gravity** (for a satellite) or **Rotation center** (for a subpart) field defines the XYZ coordinates of this point in the current item's local frame, in meters.
- The **Display axes at** option allows to specify the position of the axes, such as satellite reference frame or sun direction. If **Model frame origin** is selected, the axes will be placed at the origin of 3D model, i.e. the satellite frame. If **Center of gravity** is selected, the axes will be placed at the specified center of gravity of the object. This option is not available for subparts.

Note: Modifying the center of gravity of a satellite has no effect on the origin of the satellite's frame. The satellite's local frame can only be altered by directly modifying the 3D mesh using specialized 3D software.

Position and orientation of a satellite

The position and orientation of a satellite can be configured as described in the *Position and orientation of an entity* section. They are expressed in the EME2000 reference frame (inertial equatorial earth-centered) when the center of the reference frame is a celestial body. When another satellite is defined as the center of the reference frame, local reference frame are also available (BodyFixed, QSW, TNW). The unit for position values is the kilometer.

Configuring a sub-component

A sub-component is a part of a satellite detached from the main component. A satellite is composed of a main component, and may be composed of any number of sub-components. A sub-component itself may be composed of any number of sub-components. Sub-components can be animated in the local frame of their parent component. The definition of a sub-component is quite similar to the definition of the main component of a satellite.

General properties of a sub-component

The general properties of a component define its name in the project hierarchy. This name will be used as a unique identifier during visualization, by commands interacting with entities.

3D properties of a sub-component

The 3D properties of a sub-component define its appearance in 3D views. They are similar to the 3D properties of a satellite. Refer to the *3D properties of a satellite* for further information.

- The coordinates of the rotation center are expressed in the satellite's frame, in meters.

Position and orientation of a sub-component

The position and orientation of a sub-component are defined as described in the *Position and orientation of an entity* section. The transformations are expressed in the parent component's frame. For example, for a solar array split in several levels of components, the coordinates of the first component are expressed in the satellite's frame, those of the second component are expressed in the first component's frame, etc.

The unit for position values is the meter.

Configuring a satellite sensor

Satellite sensors can be attached to the main component of a satellite.

General properties of a satellite sensor

The general properties of a satellite sensor define the name of the sensor in the project hierarchy. This name will be used as a unique identifier during visualization, by commands interacting with entities.

Properties of a satellite sensor

By convention, in its canonical position a satellite sensor frame is aligned with the satellite's frame and the sensor points along the Z axis.

Physical properties of a satellite sensor

A sensor is described by the following physical properties:

- The sensor's shape: the **Type** drop-down list defines either an *Elliptical* (conical) or *Rectangular* (pyramidal) base for the sensor, or *azimuth and elevation* constraints to represent an angular sector of space.
- The sensor's aperture: the **Half-angle on X** and **Half-angle on Y** fields define the half-angles of aperture around the X axis (in the YZ plane) and around the Y axis (in the XZ plane). These angles are expressed either in degrees or radians. They are internally limited to 90° in OmniView and Celestia.
- Aimuth and elevation constraints: the **minimum azimuth** and **maximum azimuth** define the horizontal limits of an angular sector of space. The **minimum elevation** and **maximum elevation** define the vertical limits of an angular sector of space.

See the *Sensors in VTS* chapter for more information.

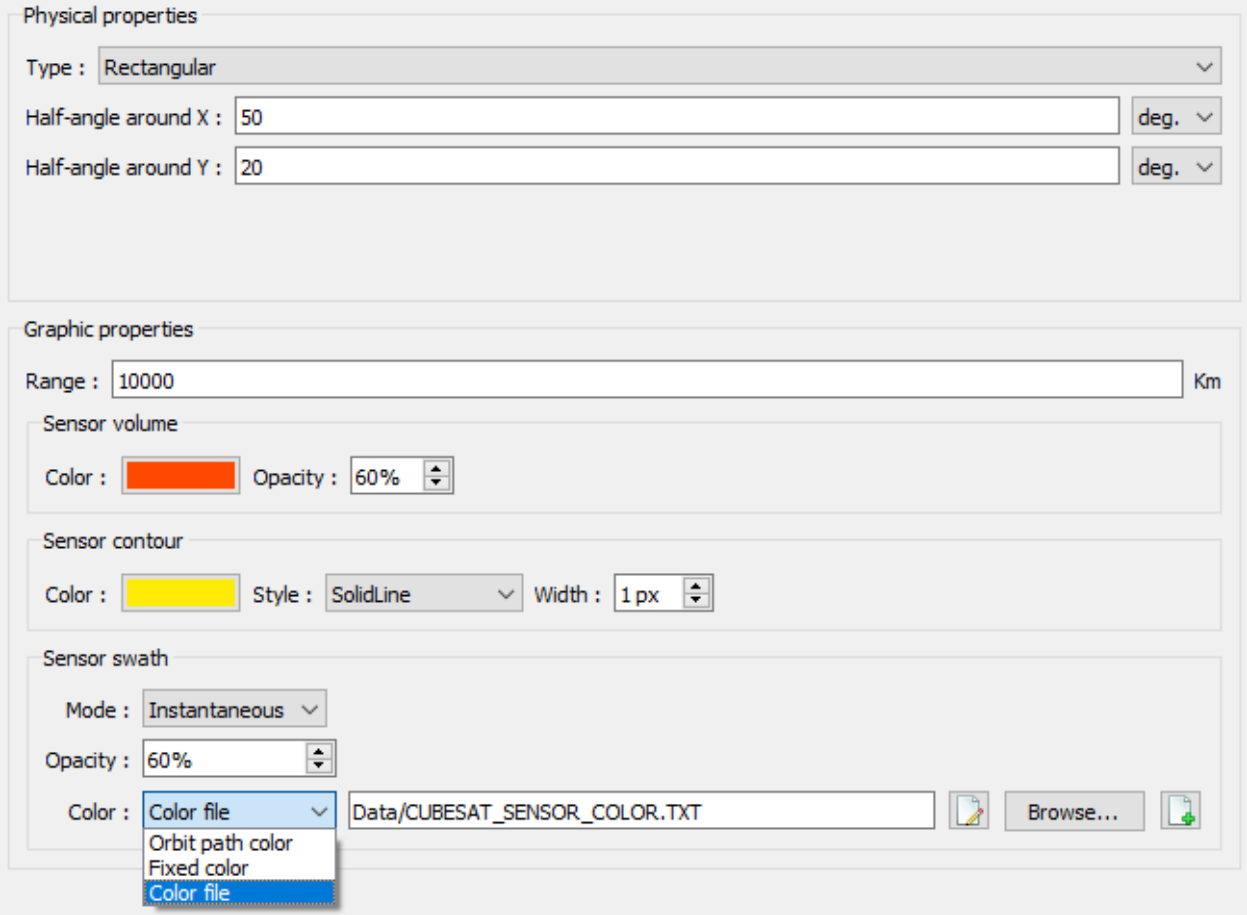


Fig. 5.17: Properties of a satellite sensor

Graphic properties of a satellite sensor

The graphical properties of a sensor define its appearance in the 2D and 3D views:

- The **Range** field defines the maximum length of the sensor volume in the 3D views, expressed in kilometers
- **Sensor volume (3D views only)**
  - The **Color** button defines the color of the sensor volume. This color is also used in the graphical interfaces to identify the sensor.
  - The **Opacity** field defines the opacity of the sensor volume (0: transparent, 100: opaque)
- **Sensor contour (2D and 3D views)**
  - The **Color** button defines the color of the sensor contour
- **Sensor swath (2D and 3D views)**
  - **The Mode list defines the display mode of the sensor swath**
    - The **Instantaneous** mode only displays the instant sensor swath (for the current location of the sensor; this corresponds to a trace of null duration)
    - The **Duration** mode allows defining the residual length of the sensor swath, expressed in hours or days
  - The **Opacity** field defines the opacity of the sensor swath (0: transparent, 100: opaque)
  - **The Color list allows defining the sensor swath color using one of the following 3 methods:**
    - The **Orbit path color** mode uses the same color as the one used for the satellite's orbit path (either defined in the general properties of the satellite, or by the CIC/CCSDS color file attached to the position file of the satellite)
    - The **Fixed color** mode uses a fixed user-defined color
    - The **Fixed file** mode uses the colors specified in a user-defined CIC/CCSDS color file

Color files format must meet the following requirements:

- MEM format with 3 (or 4) real fields ranging from 0 to 1, for the RGB components (and transparency)
- MEM format with 3 (or 4) integer fields ranging from 0 to 255, for the RGB components (and transparency)
- MEM format with 1 string field in HTML color format ("#RRGGBB" in hexadecimal), for the RGB components (no transparency)

Note: When using a color file, the black color (0 0 0) is used to disable rendering of the sensor swath while it is in effect. This allows drawing the sensor swath by "patches", e.g. to visualize instrument power-on/power-off cycles.

Position and orientation of a satellite sensor

The position and orientation of a satellite sensor are defined as described in the *Position and orientation of an entity* section. Transformations are expressed in the satellite's frame.

Note: The *Direction* and *Azimuth and elevation* modes are not available for the orientation of a sensor as the induced degree of freedom implies the orientation of the sensor cannot be correctly specified.

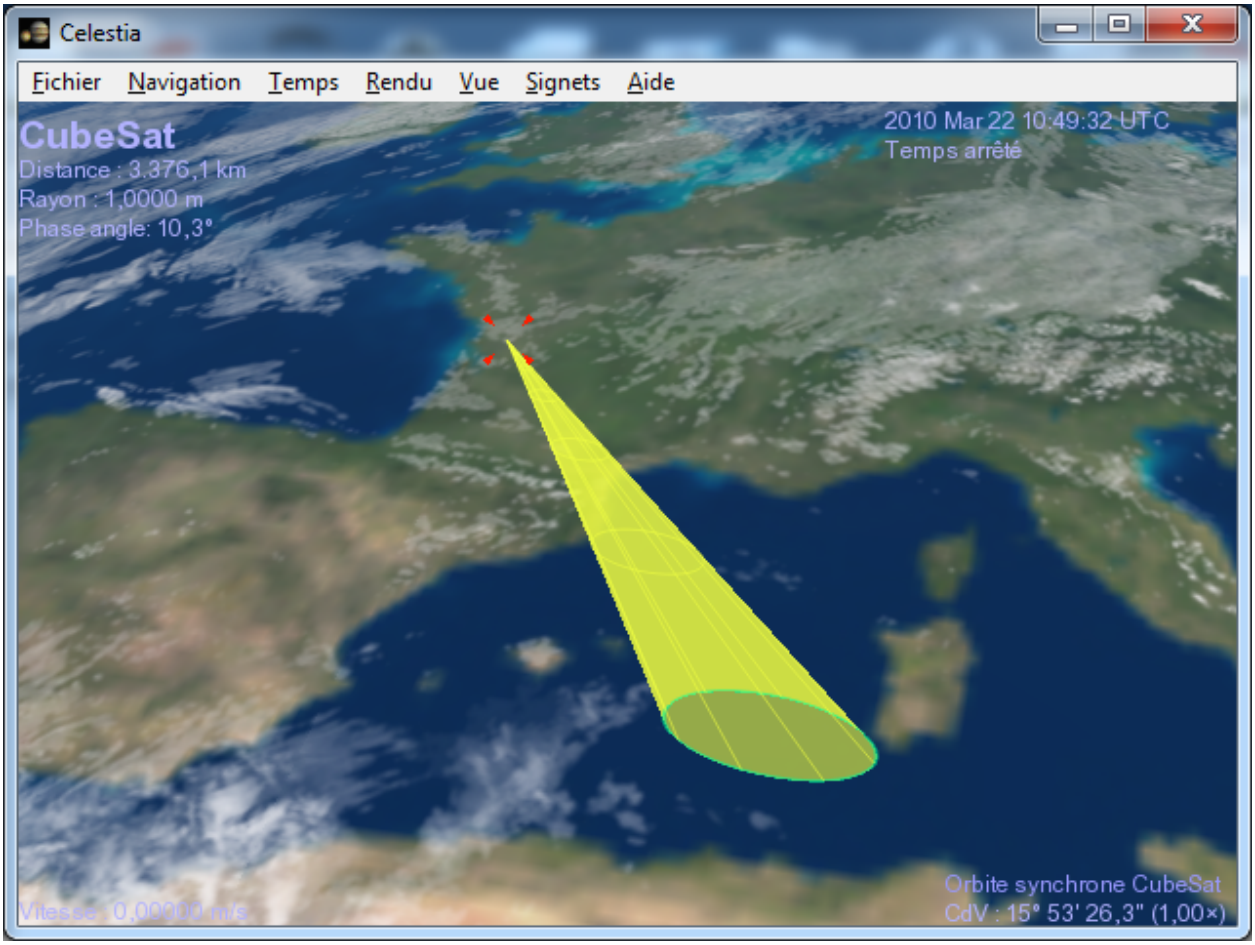


Fig. 5.18: Satellite sensor in Celestia

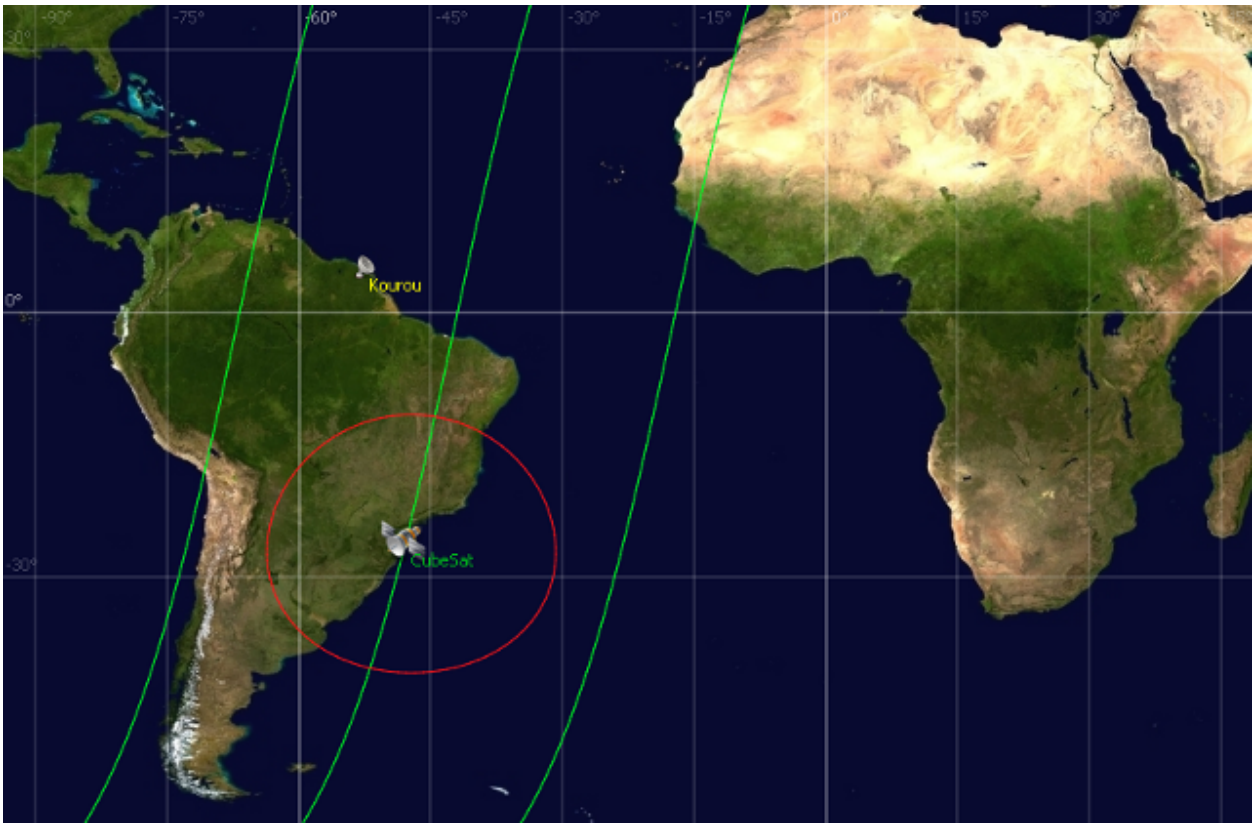


Fig. 5.19: Satellite sensor in SurfaceView

Configuring a visualizer

A visualizer is a virtual entity making it possible to display different physical quantities in the scene. Support for different viewers depends on the ability of each client application to display them.

See :

- *Positional covariance ellipsoid*
- *Grid*
- *Spherical shell*

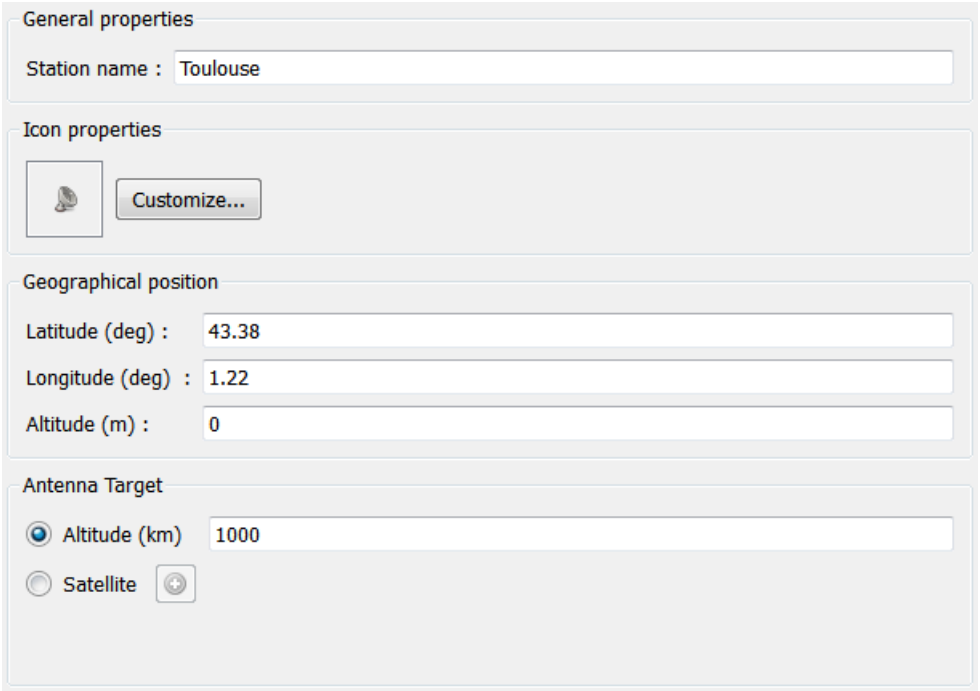
General properties of a sub-component

Configuring a ground station

Any number of ground stations can be attached to central bodies in the project. Please read the section *Ground station sensor* of the chapter *Sensors in VTS* for more detailled information.

General properties of a ground station

The general properties of a ground station define its name, icon, position on its central body, and the target of its antenna.



- The name of a ground station must be unique on its parent body.
- The geographical coordinates (longitude and latitude, in degrees) of a ground station must be defined. Its altitude (in meters) is optional.
- The target of a ground station's antenna can be either a fixed altitude, or the altitude of one or more project's satellites.

Note that when a ground station is configured targeting a satellite :

- A specific camera will be available for each ground station to view the satellite from or towards it
- The geometrical visibility between the satellite and the ground station will be display in SurfaceView

Refer to *Icon configuration* sections in order to configure the icon for a ground station.

Properties of a ground station sensor

By convention, in its canonical position a ground station sensor frame is aligned with the ground station's frame and the sensor points along the Z axis.

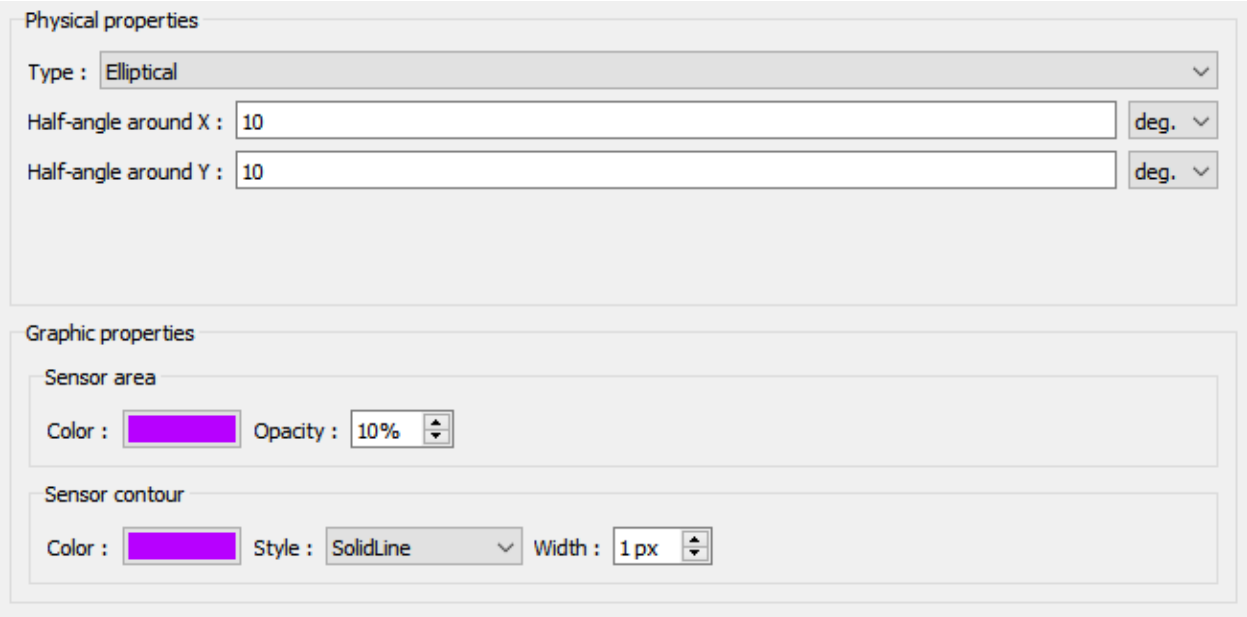


Fig. 5.20: Properties of a ground station sensor

Physical properties of a ground station sensor

These properties are identical to those of a satellite sensor. Refer to the *Physical properties of a satellite sensor* section for further information.

Graphic properties of a ground station sensor

The graphic properties of a ground station sensor define the appearance of the sensor.

- **Sensor area**
  - **Color**: defines the color of the sensor area .
  - **Opacity**: defines the opacity value for the transparency.
- **Sensor contour**
  - **Color**: defines the color of the contour.

- **Style**: defines how the contour is traced (with dashes, with dots, . . .).
- **Width**: defines the thickness of the contour.

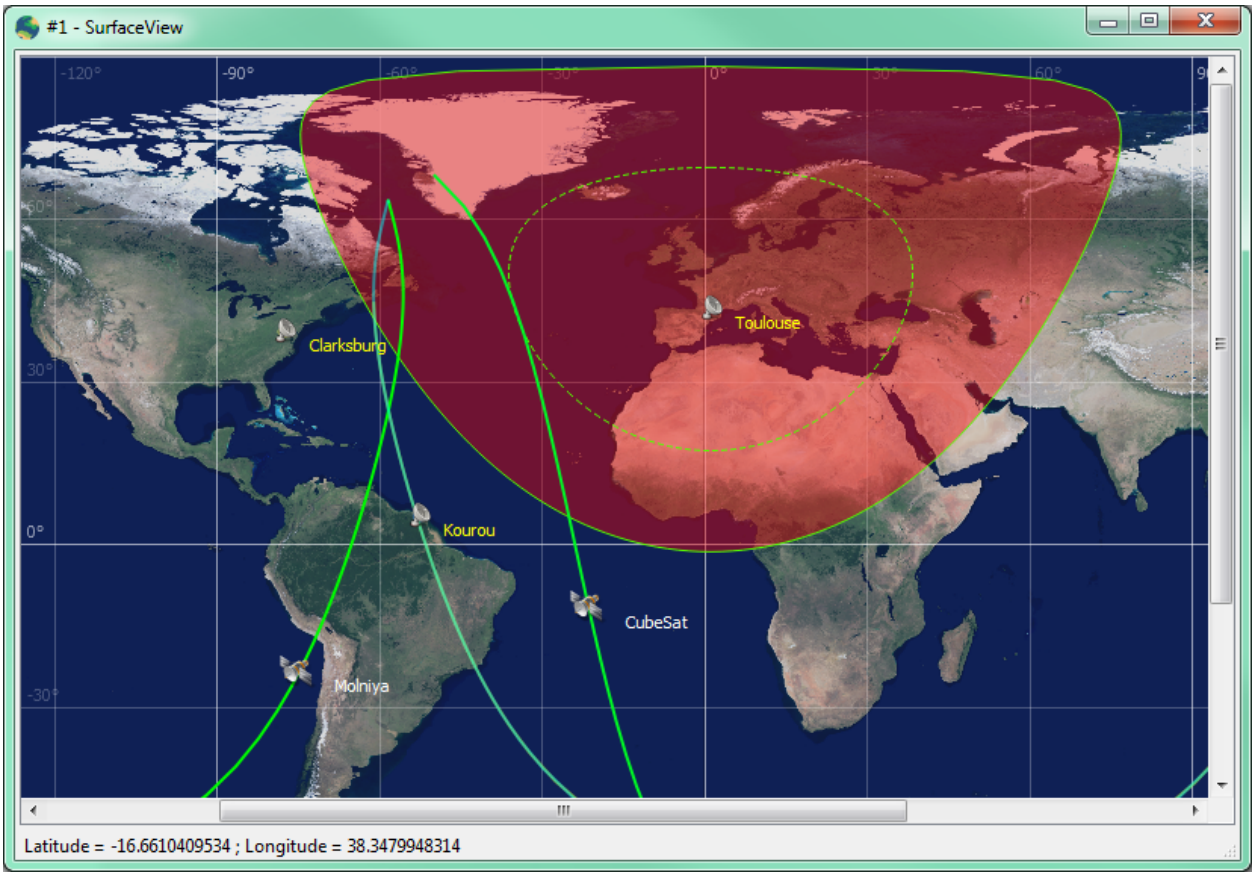


Fig. 5.21: Ground station sensor in SurfaceView

Masks of a ground station

See the *Physical masks in VTS* section for further information.

Secondary ground station sensor

See the *Ground station sensor* section of the chapter *Sensors in VTS* for further information.

Configuring a Point Of Interest

A Point Of Interest (abbreviated POI) represents one or many locations in geographical coordinates. Any number of POIs can be attached to central bodies in the project.

Refer to the *POIs and ROIs in VTS* chapter for more information of the POI file format.

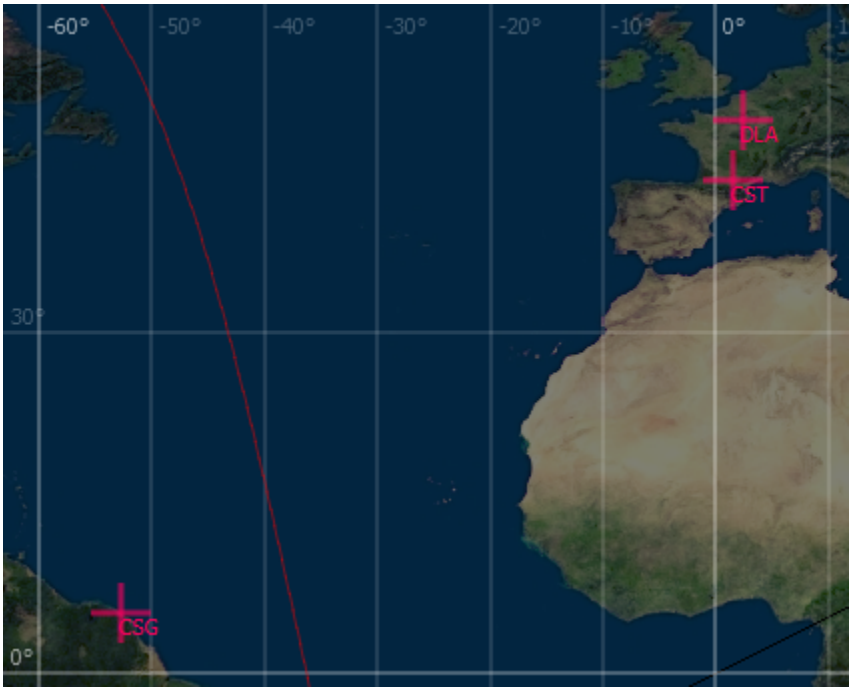


Fig. 5.22: Points Of Interest in SurfaceView

General properties of a POI

The general properties of a POI define its name. It must be unique among other POIs in the same parent body. POIs support the complex naming rule described in *Object paths in VTS*.

Fig. 5.23: Properties of a Point Of Interest

Graphic properties of a POI

The graphic properties of a POI define the shape of the location markers, as well as their color and opacity. (see the *Icon configuration section* of the VTS configuration user manual).

Configuring a Region Of Interest

A Region Of Interest (abbreviated ROI) represents one or many polygons in geographical coordinates. Any number of ROIs can be attached to central bodies in the project.

Refer to the *POIs and ROIs in VTS* chapter for more information of the ROI file format.

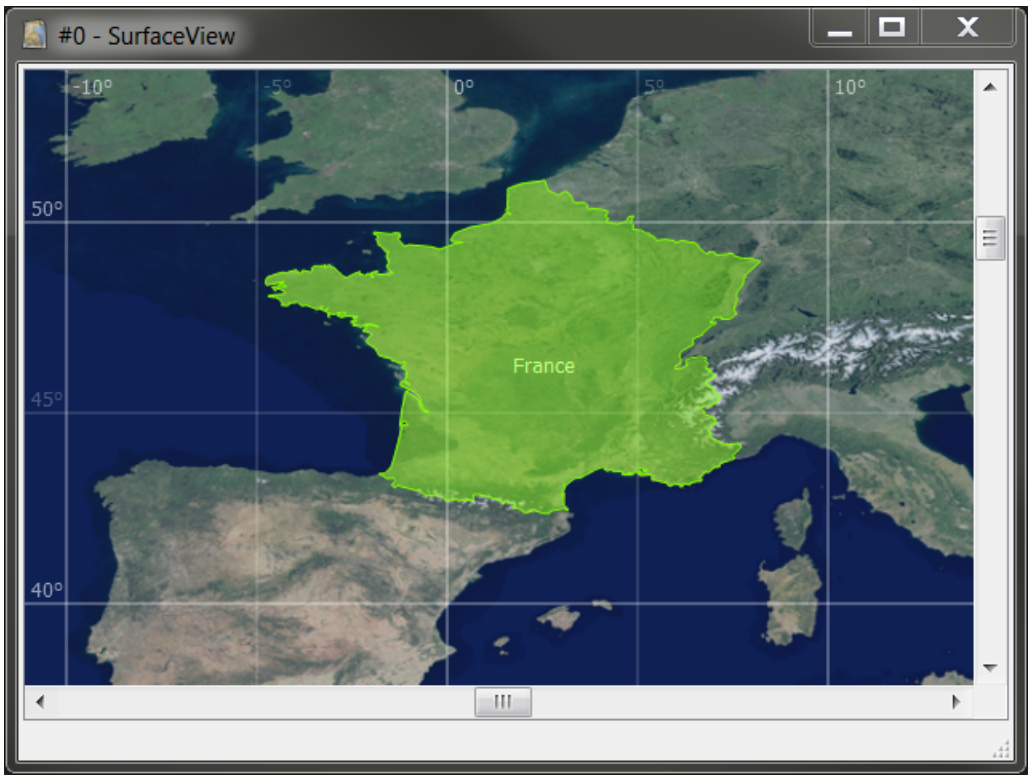


Fig. 5.24: Region Of Interest in SurfaceView

General properties of a ROI

The general properties of a ROI define its name. It must be unique among other ROIs in the same parent body. POIs support the complex naming rule described in *Object paths in VTS*.

Graphic properties of a ROI

The graphic properties of a ROI define the color of the polygon and its opacity.

Configuring a Cluster

A cluster represent a set of objects where each of them an orbit.

The cluster entity serves two purposes: - facilitating the creation of hundreds or thousands entities - improving the visualisation performance for SurfaceView and Celestia

The population of the cluster will be represented by an icon and a label whose text is the unique ID of the object in the cluster.

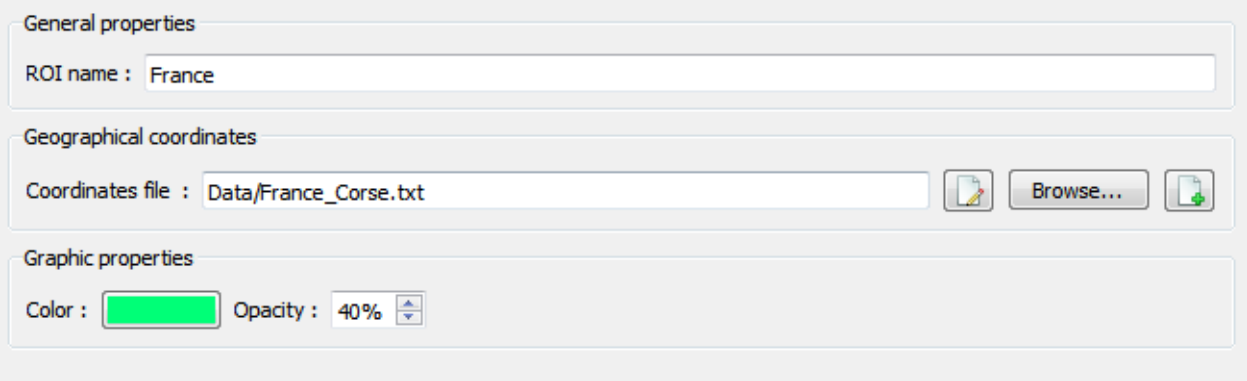


Fig. 5.25: Properties of a Region Of Interest

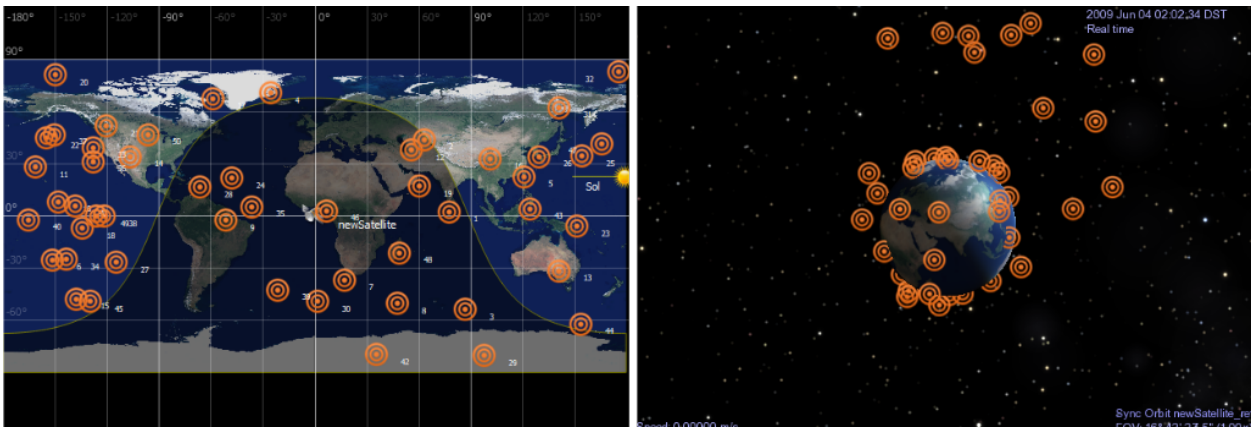


Fig. 5.26: A Cluster displayed in SurfaceView and Celestia

General properties of a Cluster

The general properties of a cluster are its:

- Name
- Central body
- Radius
- Object radius
- Population folder

Name and central body of a cluster

The name of the cluster will be the cluster's unique identifier during visualization, used to identify the cluster in view properties and messages between the Broker and client applications.

The central body of the cluster defines the reference frame for the cluster, in which the cluster object's positions are expressed.

Radius and object radius of a cluster

The radius is the approximate radius in kilometers of its global bounding sphere. In 3D: if an element disappears when the camera is turned towards the outside of the bounding sphere, it is necessary to increase this radius.

The object radius is the approximate radius for the objects composing the cluster. In 3D: small items will disappear quicker when the camera is moved away from the object.

Population folder of a cluster

All files with the following extensions :

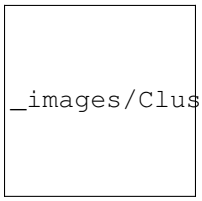
- *.oem
- *.txt
- *.TXT

contained at the root of the specified folder will be imported if and only they are CIC files with oem content. All these files will be copied if necessary in the project folder.

Graphics properties of a cluster

The graphic properties of a cluster consist in:

- an icon : like for satellites, the icon can be customized in appareance and positionning. Only the appearance will be taken into account in Celestia.
- a label : color and size of the text will only be applied in SurfaceView.



Configuring a Link

The general properties of a link are:

- its name
- the entities that are linked together
- the color of the link

Configuring a client application

Any number of client applications may take part in the visualization of a project. The view properties of these applications can be configured in the *project scenario*.

Selecting an application in the project hierarchy shows some information and parameters:

- Text information about the application (found in the application's *README* file).
- **Start Options**
  - Giving an application label allows to specify a custom name for the application. This label will replace the application name during the visualization.
  - Unchecking "Launch *application* #*N* when Broker starts" allows to delay the start of the application.
- Some applications define parameters which must be passed on to the application once it has successfully connected to the visualization. Those can be specified in the **Initial States** area.
- Some applications require specific parameters to be passed on to their launchers. Those can be specified in the **Specific Args** field.

5.1.6 Configuring event types

Event types can be configured in the **Event Type Editor** tab of the VTS configuration utility. The appearance of the event types in the Timeline and 2D views can be specified for all events of the project. The GUI is composed of the following:

- An action toolbar
- A hierarchical view of all event types
- An editing pane for the currently selected event type

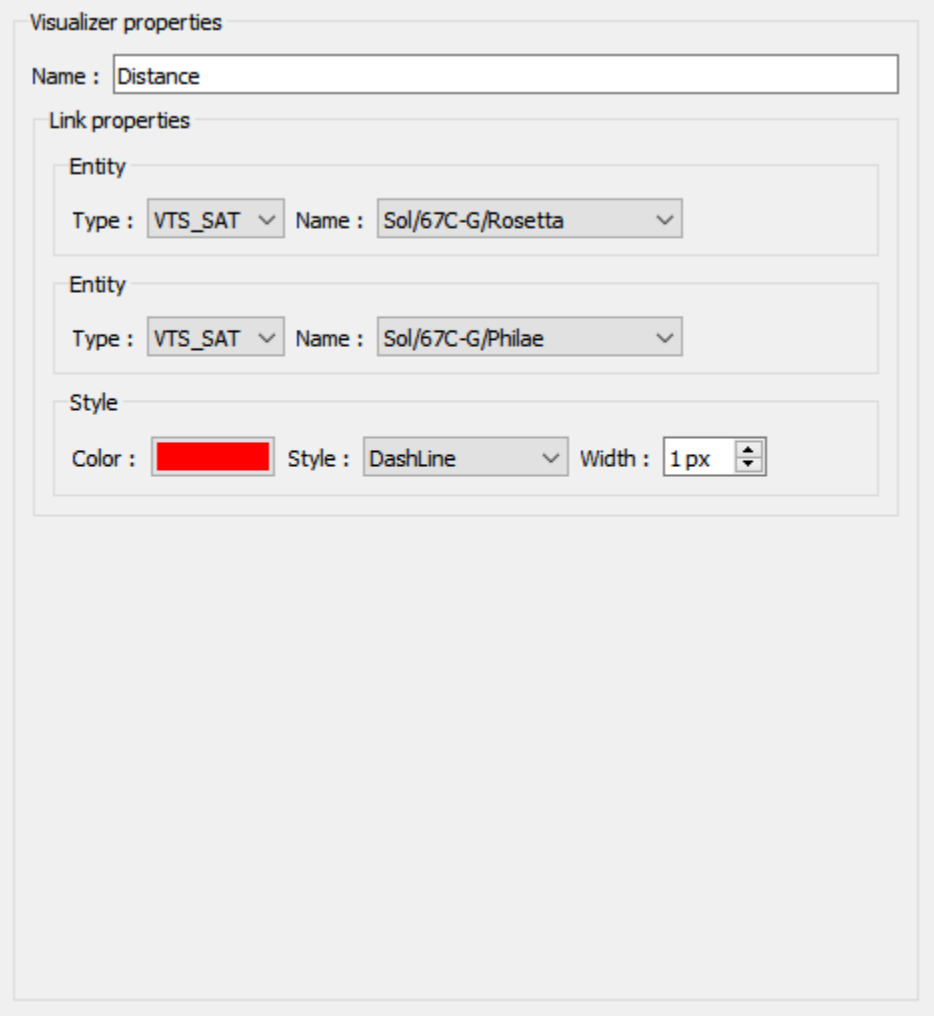
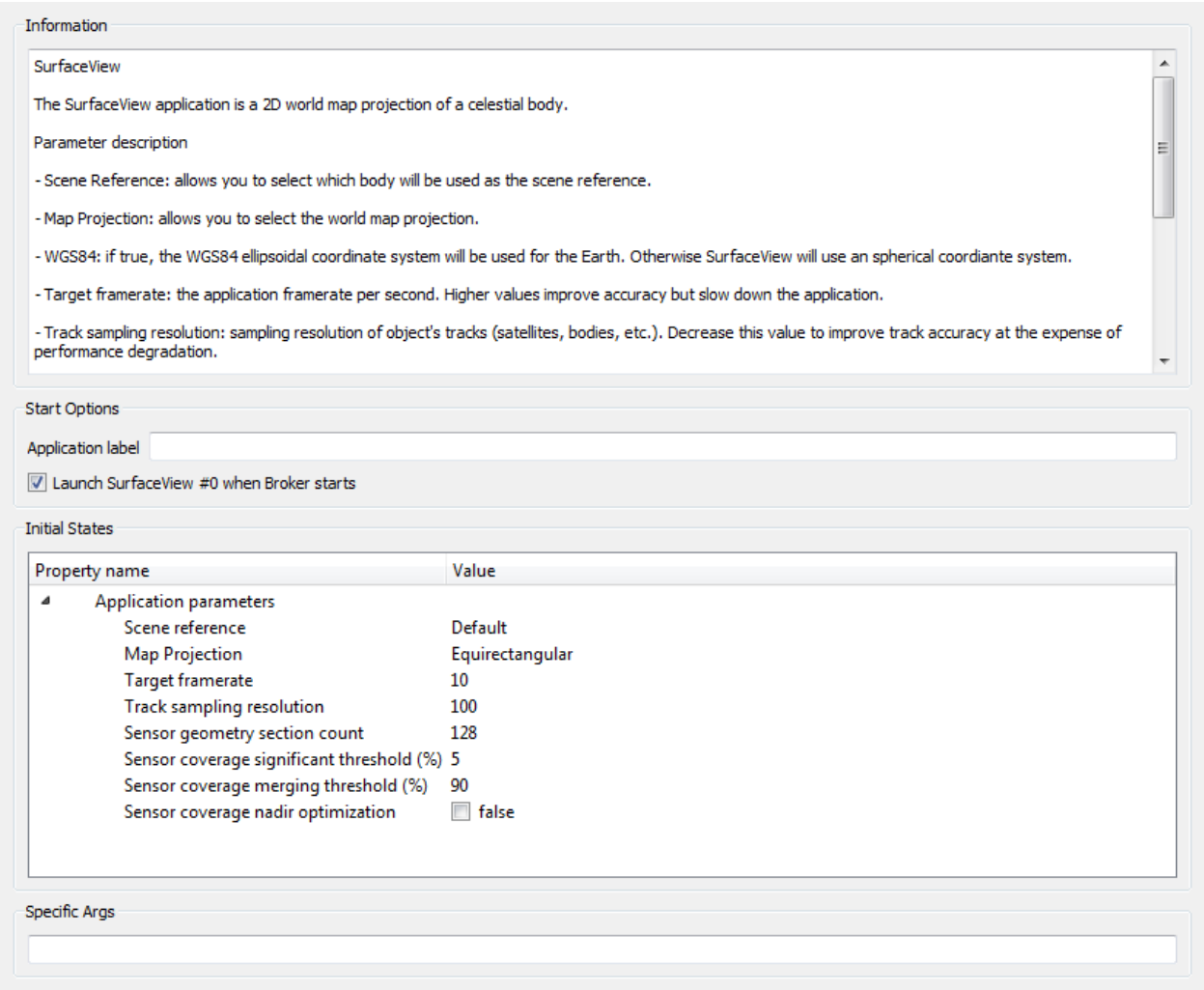


Fig. 5.27: Properties of a link



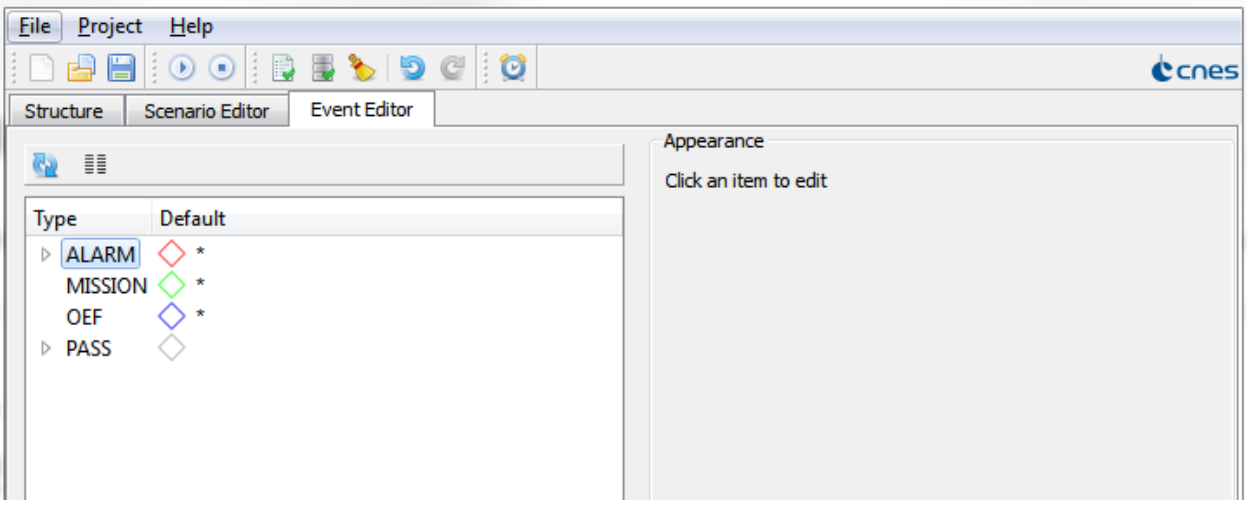


Fig. 5.28: Event type editor

Loading event files

The event types displayed in the tree view are loaded from the event files attached to the project's satellites. Clicking the **Reload Events**  button in the toolbar reads all event files and displays the event types available in these files. This should be done manually each time an event file is added to or removed from the project, or if existing event files are modified. Decorations of existing event types are not affected by a reload.

Configuring default event type decorations

When the **Toggle Edit Entity Decorations Mode**  button is untoggled, the event type editor is in "basic" mode and only the **All satellites** column is visible. Decorations configured in this mode will apply to all satellites in the project.

The default decoration displayed in the tree for all types is a gray diamond. Clicking the decoration on the line of an event type selects it for edition in the editing pane. The **Customized** option can then be enabled to customize the event type's decoration. See the *Icon configuration section* of the VTS configuration user manual for more information about editing an icon.

The preview area displays the event type decoration as it will appear in client applications that handle events.

Stars in the hierarchical view indicate event types with custom decorations. Decorations are inherited by all descendant types of a decorated event type, unless one of these descendants itself has its decoration customized.

Configuring satellite-specific event type decorations

When the **Toggle Edit Entity Decorations Mode**  button is toggled, the event type editor switches to "advanced" mode. In this mode, event type decorations specified the **All satellites** column are applied by default, unless satellite-specific decorations are set in the per-satellite columns of the hierarchical view.

Note: All custom satellite-specific decorations will be lost when switching from "advanced" to "basic" mode. When loading a VTS project with satellite-specific decorations, the "advanced" mode is automatically selected.

5.1.7 Settings dialog

General options independent from the project can be set in the **File/Settings...** menu item.

Project options

Use auto compute dates by default

By default this option is unchecked : the **compute dates** button must be pressed to take in consideration all data files in use for determining the start and end dates of the project . Check this option to automatically compute date on each visualization run, but it might take a while on large data files.

Clear all client application data caches

This action triggers an action for all compatible VTS applications that clears the temporary files. This action is available for each client application providing an application cleaner (see documentation in the *developer manual*)

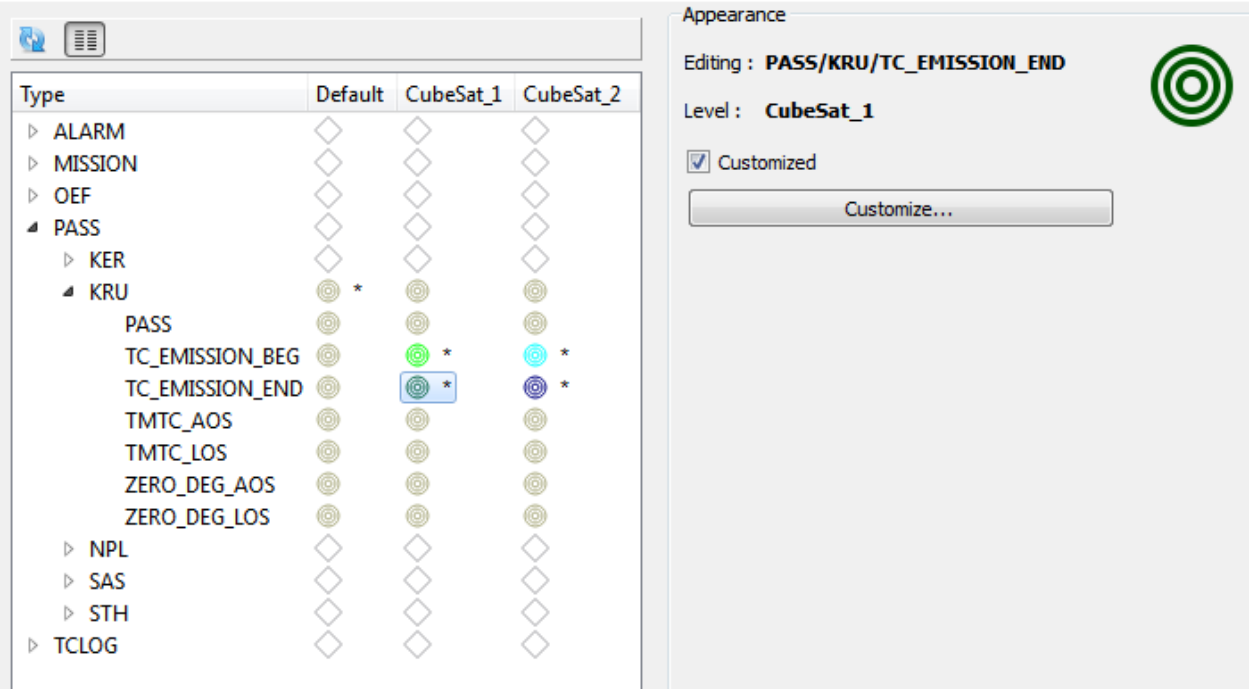
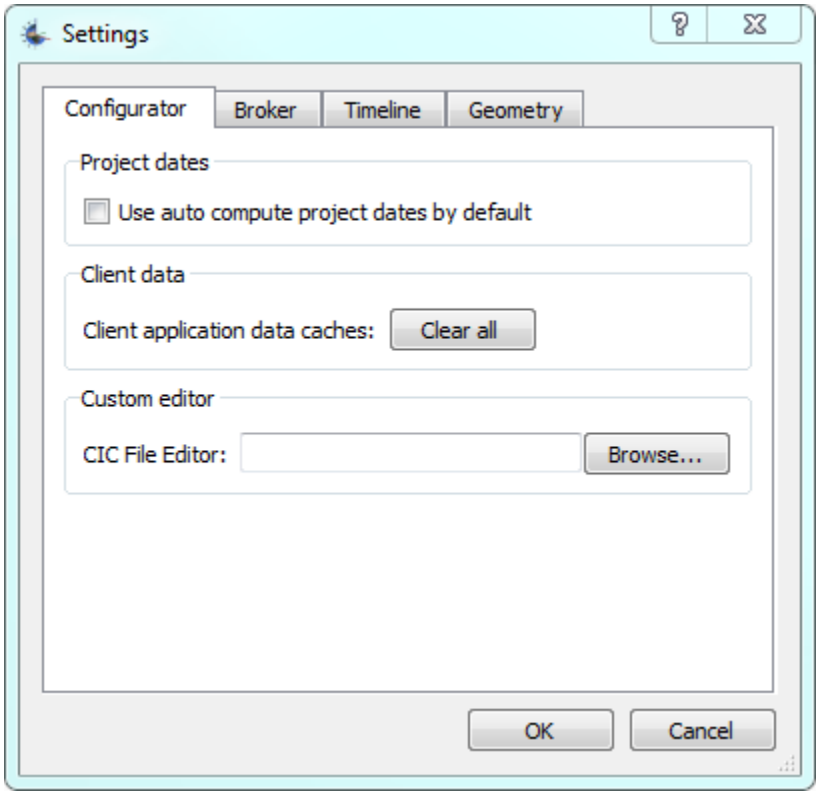


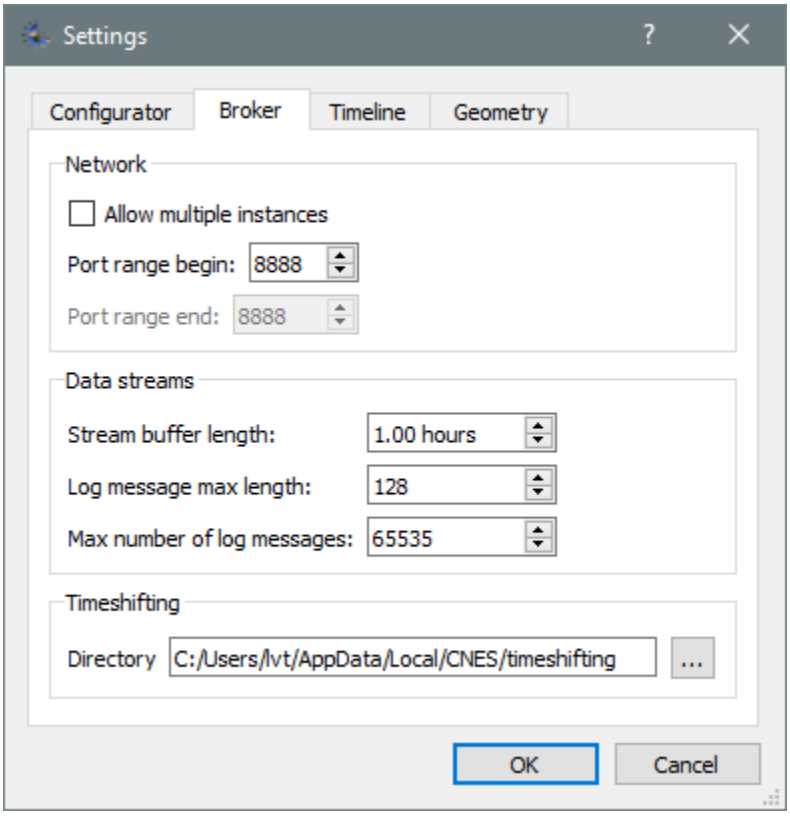
Fig. 5.29: Entity-based event type decorations



Custom editors

This section allows you to define custom editor or executable to open CIC files. The specified executable will be launched by VTS with the CIC file path as a single parameter. If no executable is defined, the system's default application for this file type will be used.

Broker options



Allow multiple instances

By default, the broker server runs on port 8888. If a visualization is run on a second instance of VTS on the same computer, the Broker will fail. Check this option to fill a range of ports and when the Broker starts, it will scan the range for an open port.

Let this option unchecked but change the port range begin value to force a single value different from 8888.

Warning: If the server port is different from `8888`, the launchers will be started with a `--serverport <portNum>` option in command line.

Stream buffer length

Configure the length of the DATA buffer, which is used to display streamed values in the Timeline.

Log message max length

Configure the message max width displayed in the Server/Received packets or Server/Sent packets. Wider messages are truncated.

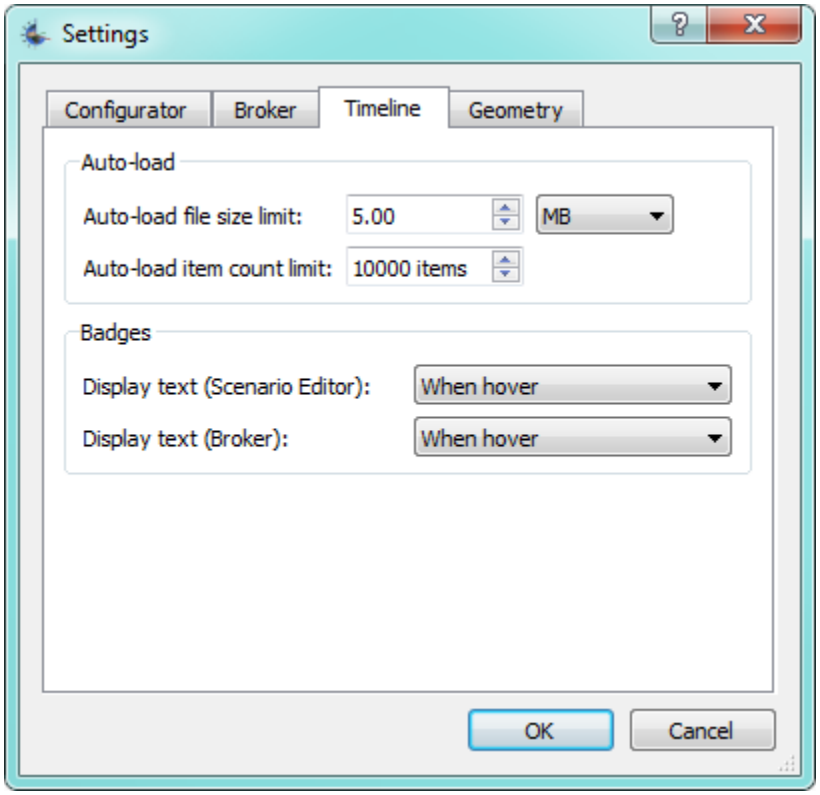
Max number of log messages

Configure the number of messages displayed in the tabs *Received packets* and *Send packets* of the *Server* tab of the Broker. If this value is increased, older messages can be viewed but the Broker's performance can be affected when many network messages are received at the same time.

Timeshifting directory

Configure the folder path which will be used to save databases during the playback of project supporting timeshifting.

Timeline options



Auto-load file size limit

Configure the maximum size for files displayed in the project timeline. The data for files above this size will not be loaded: the file will appear as a gray box without tooltips.

Note: Please note that the dates and file type are still loaded.

Tuning this setting allows to speed up the loading time of the timeline for projects with large files.

Auto-load item count limit

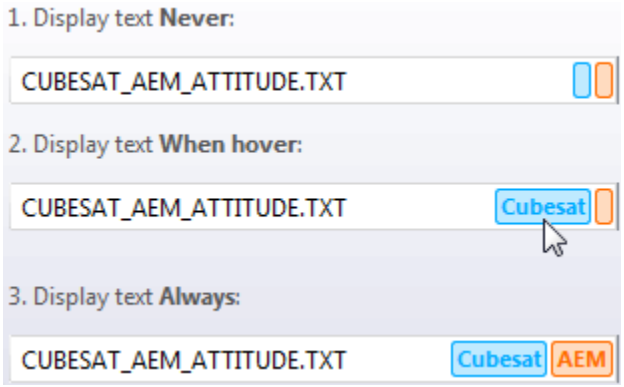
Configure the maximum number of data lines for files displayed in any mode other than *Block* mode in the project timeline. Files containing more lines will have their data loaded but displayed in colored *Block* mode (with tooltips showing the data).

Tuning this setting allows to improve the responsiveness of the timeline for projects with large files.

Display text (Scenario Editor, Broker)

Controls how badges are displayed in the Timeline (Scenario Editor, Broker). See image below.

- 1. Display text **Never**: badge text is never displayed
- 2. Display text **When hover**: badge text is only displayed when hover with mouse cursor
- 3. Display text **Always**: badge text is always displayed



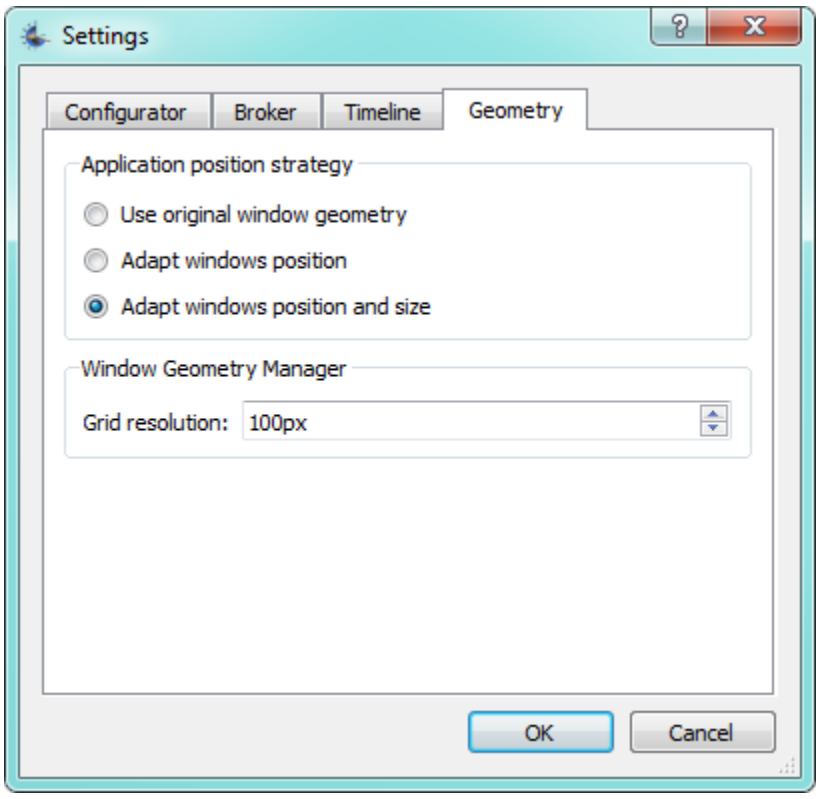
Geometry options

Application position strategy

The definition of a VTS scenario is tighgly related to the user's monitor configuration. For example, a given scenario can display a VTS client application in the bottom-right part of the primary screen and another full-screen on the secondary screen. In some cases, the monitor configuration used to define a VTS project it is different to the one used to visualize it.

In order to adapt as best as possible the position and size of client application windows to the current monitor configuration, VTS allows you to choose the strategy to apply when visualizing a VTS project. Available options are:

- *Use original window geometry*: VTS will not modify the geometry (size, position) of client applications's windows
- *Adapt windows position*: VTS will adapt window positions to the current monitor configuration. If a window was displayed in a secondary screen in the original monitor configuration it will be displayed in the secondary



screen of the current monitor configuration, regardless of the logical position of the screen on the system's virtual desktop.

- *Adapt windows position and size* (default): VTS will adapt windows position to the current monitor configuration as the previous option. Windows will be resized to cover as much screen space as it covered in the original VTS project. If a window covered 100% of a screen in the original monitor configuration, it will cover 100% of the screen regardless of any change in the screen size.

Grid resolution

To arrange all client application windows on the user's monitor configuration, the Window Geometry Manager uses a grid where each cell represents a portion of the monitor. The dimension of the portion is guided by the grid resolution.

The grid resolution gives a square which represents the ideal cell of the grid. But, in order to avoid truncated cells, cell dimension might be modified to create a grid of rectangle where all cells are identical.

For example, if the grid resolution is equal to 192 for a monitor with a 1920x1080 resolution, each cell will be a 192x180 rectangle.

5.2 Broker user manual

The Broker is the core application of the visualization phase. It starts the client applications, regulates the visualization time, sends user commands to the client applications, and much more.

5.2.1 Command-line options

The Broker is started by the VTS configuration utility when the **Run** button is clicked.

It can also be started from the command line, either directly or through the *startVTS* binary. The latter uses the following syntax:

```
startVTS --batch [Broker options]
```

When started from the command line, the following options can be specified:

–project <File.vts>

This option tells the Broker which VTS project file to load and visualize.

Example:

```
broker --project C:\Project\CubeSat.vts
```

–specificargs <appIdName> <SpecificArgs>

This option defines additional arguments which will be passed on to client applications upon startup.

Client applications can be identified in several ways:

- The application ID (1, 2, etc.)
- **The application name ("Celestia", "SurfaceView", etc.)**
  - If no number is specified, then the given arguments will be passed on to all clients with the given name (i.e. all instances of Celestia, SurfaceView, etc.)
  - If an additional number is specified ("Celestia:1", "PrestoPlot:3", etc.), then the given arguments will only be passed on to the n-th instance of the application

Examples:

```
broker --project C:\Project\CubeSat.vts --specificArgs 1 "--someOption"
broker --project C:\Project\CubeSat.vts --specificArgs Celestia "--url exampleUrl"
broker --project C:\Project\CubeSat.vts --specificArgs SurfaceView:2 "--colorLayer"
```

The *–specificArgs* arguments must be specified after the *–project* option on the command line.

–datadir <dataDirectory>

Use an alternate data directory for the project's data: file names referenced by the project will be resolved from this directory instead of the project file's directory.

Example:

```
broker --project project-without-data.vts --datadir /path/to/project/data
```

This is used internally by VTS when the project file's directory is read-only. The project is then run from a temporary directory, using the data located in the original project file's directory.

–tempdir <tempDirectory>

Start the Broker in read-only mode, meaning it will not write to its installation directory. It may still write to project files outside its installation directory.

In this mode, application data for the visualization and changes to a project file located inside the installation directory will be written to the specified temporary directory instead. It the directory is `""`, the Broker will create one.

Example:

```
broker --tempdir /tmp
```

This is used internally by VTS when its installation directory is read-only, or when the configuration utility has been started with the *–readonly* option.

–port

Force the Broker to use a specific port number to start its server. By default, the port 8888 is used.

Example:

```
broker --port 9999
```

–close

The Broker is started as a client and try to connect to an already running Broker, according to the –port option. When connected, the AUTOCLOSE command is send and the Broker closes.

Example:

```
broker --close --port 9999
```

–closeIfProjectAppsEnd

This option programs the Broker shutoff when all client applications are closed, regardless dynamic or external applications connected.

Example:

```
broker --closeIfProjectAppsEnd
```

–restart

The Broker is started as a client and try to connect to an already running Broker, according to the –port option. When connected, the AUTORESTART command is send and the Broker restarts or closes, depending on if the option –externalrestart has been specified.

Example:

```
broker --restart
```

–externalRestart

This infrastructure option tells the Broker to not restart itself when it receives an ANTORESTART command. Instead, it closes with a special exit code. This option is used by the configuration utility.

Example:

```
broker --project C:\Project\CubeSat.vts --externalRestart
```

–stdout

This option tells the Broker to write every information message to the standard output instead of the logger tab.

Example:

```
broker --project C:\Project\CubeSat.vts --stdout
```

–nocheckvalidity

This option disables the project validity check. This option can improve the startup speed of a project visualization but should be used only with a valid project.

Example:

```
broker --project C:\Project\CubeSat.vts --nocheckvalidity
```

–logdir <logDirectory>

This option redirects log get by the Broker (which means log from the Broker or from applications during the visualization) in a specific directory. This specific directory has to exist before using the option.

Example:

```
broker --project C:\Project\CubeSat.vts --logdir C:\Project\Logs
```

5.2.2 Start of a visualization

When a visualization is started from the *VTS configuration utility*, the Broker starts all the *client applications* defined in the VTS project. During this initialization phase, the Broker displays the *Initializing* message in the text fields of the time control area. Time does not start flowing until all client applications have signaled they are ready.

Upon connection of the various client applications, the Broker's tabs are populated with commands and information regarding these applications.

Visualization automatically starts to play with time ratio 1 once initialization is finished, unless specified otherwise in the VTS project.

5.2.3 End of a visualization

The visualization can be stopped directly by clicking the top-right cross button to close the Broker. It also stops once all client applications have been closed by the user, or if the VTS configuration utility is closed (if the visualization was started from there).

5.2.4 Broker menu

The Broker menu can be reached through the gears icon in the top-left corner of the Broker.

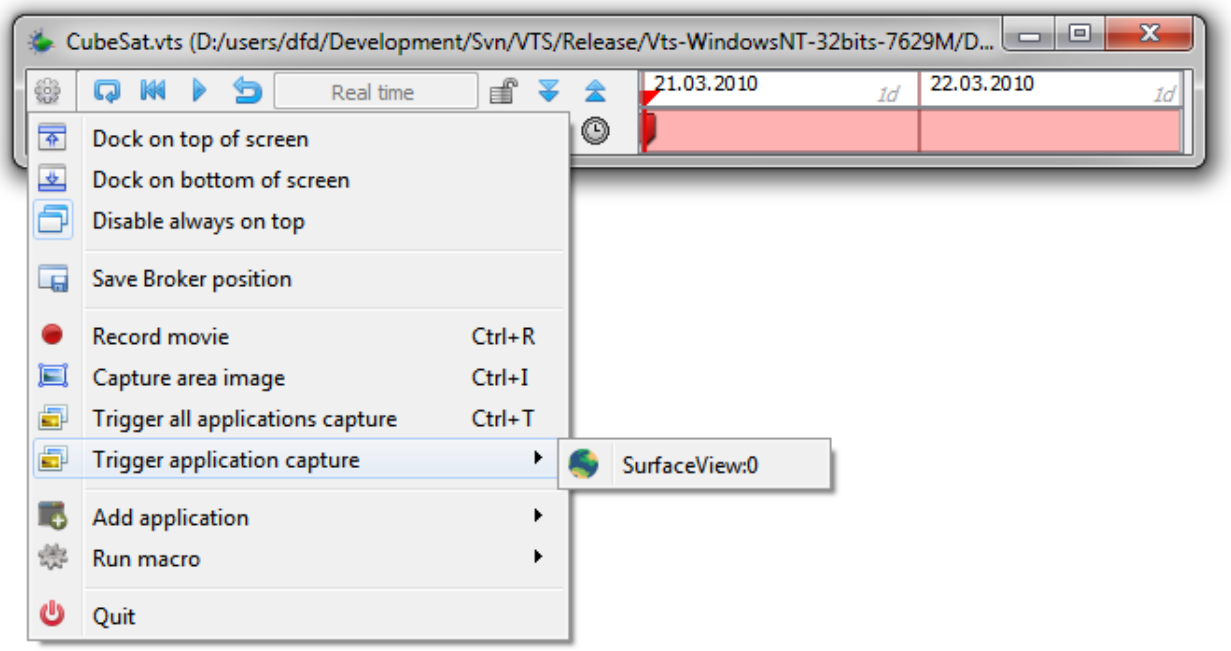


Fig. 5.30: Broker menu

It offers the following options:

- **Dock on top of screen / Dock on bottom of screen / Enable always on top**: Control special Broker display modes (refer to the *Display modes* section below for more information)
- **Save Broker position**: Save the Broker window state and position, so that it can be restored at the next startup of project visualization. The Broker is guaranteed not to disappear in the case of a change of the desktop geometry. If the Broker monitor is no longer available, the Broker window will move itself inside the closest available monitor.
- **Record movie**: Record a movie (refer to the *Recording movies* section below for more information)
- **Capture area**: Save an image of a screen area
- **Capture all applications**: Request all application to capture itself into an image
- **Capture application**: Request an application to capture itself into an image
- **Add application**: Start a new dynamic client application (refer to the *Applications* tab section below for more information)
- **Run macro: Run a CIC/CCSDS macro file (refer to the *Scripts and macros in VTS* chapter for more information)**
  - **Choose macro...**: Browse for a CIC/CCSDS macro file to run
  - **Project macros**: Run a CIC/CCSDS macro file found in the *Macros* subfolder of the project folder
  - **VTS macros**: Run a CIC/CCSDS macro found in the *Apps/Broker/macros* subfolder of the VTS installation folder

- **Quit**: Close the visualization

5.2.5 Display modes

The Broker can either display itself in compact or full mode. In compact mode, only time control buttons are displayed and the Broker is pinned in the foreground (always on top of other windows). In full mode, the tabbed interface is displayed, allowing access to the scenario timeline, all client commands, and information on the state of the visualization. By default, at the start of a visualization, the Broker is in compact mode. The **Unfold** arrow button allows switching to full mode.

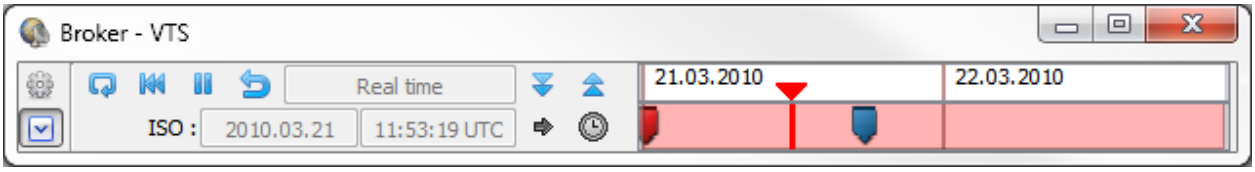


Fig. 5.31: Compact display mode

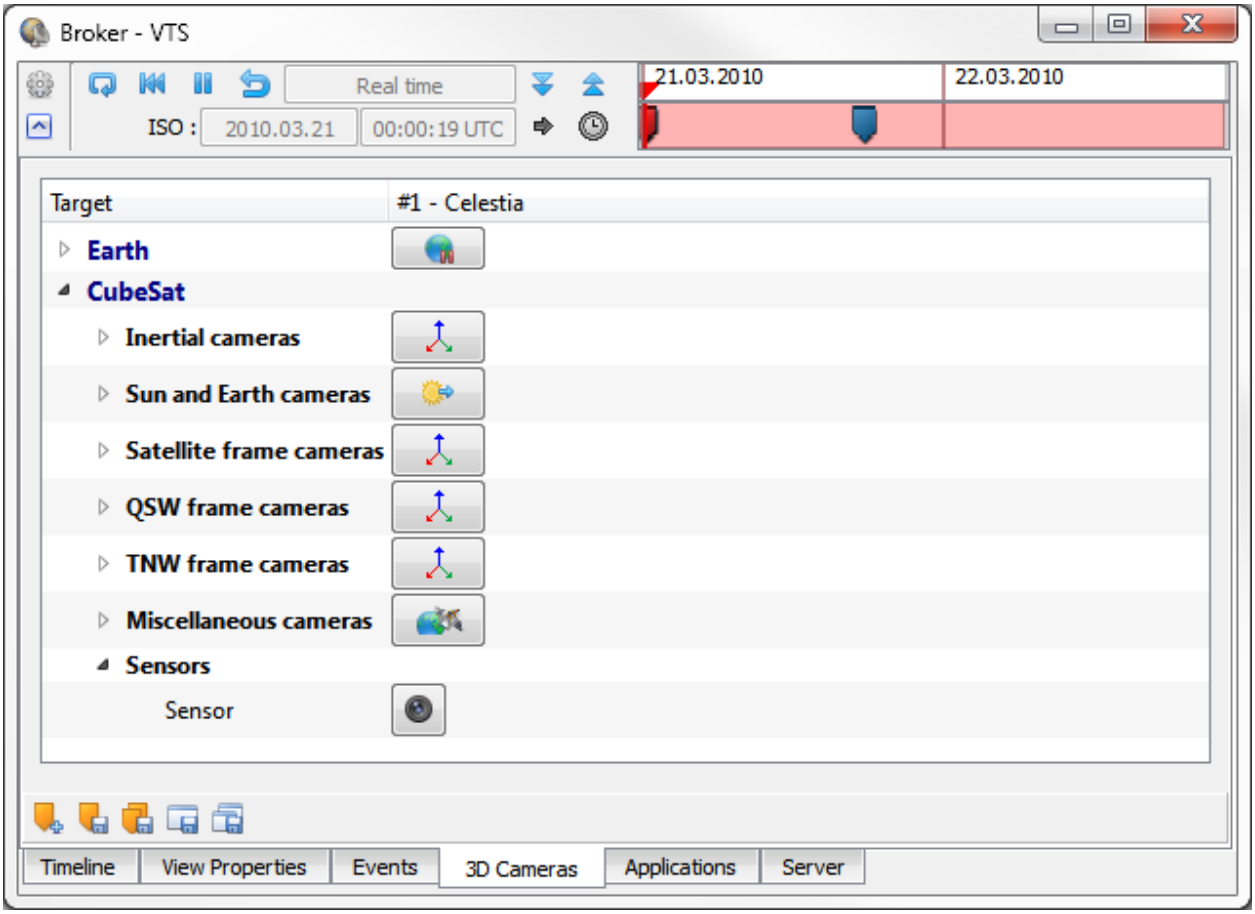


Fig. 5.32: Full display mode

The gears menu in the top-left corner of the Broker offers the following options:

- **Dock on top of screen** : dock the Broker in compact mode at the top of the current screen

- **Dock on bottom of screen** : dock the Broker in compact mode at the bottom of the current screen
- **Enable always on top** : whether or not the Broker window should remain always on top of other windows (enabled by default in compact mode)

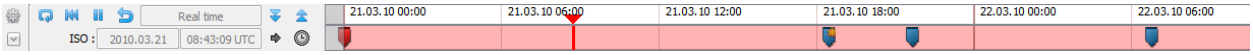


Fig. 5.33: Docked display mode

5.2.6 Time management

Controls and information regarding time are available in all Broker display modes. The following interface allows interacting with visualization time:

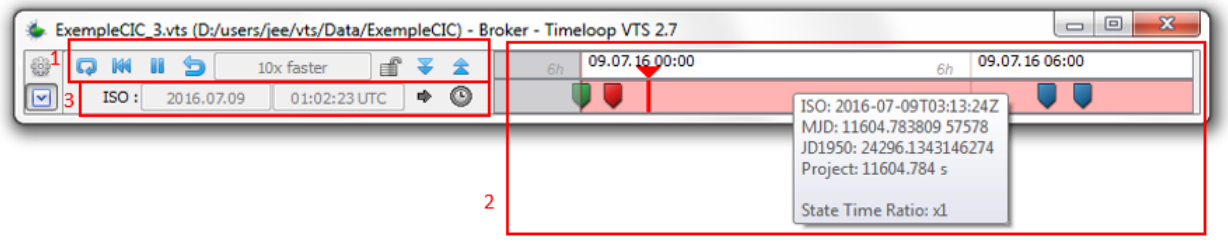


Fig. 5.34: Visualization time interaction areas

Time controls

The buttons in area (1) control time in all client applications:

- **Time Loop**: Toggles loop playback for the visualization.
- **Restart**: Resets time to the project's start date. The play/pause status remains unchanged.
- **Play/Pause**: Plays/pauses the time flow. Pausing does not prevent interaction with the visualized entities in the client applications, or navigation in visualization time using the timeline.
- **Reverse Time**: Every click on this button inverts the time flow direction. The time ratio remains unchanged.
- **Use Scenario State Time Ratio**: When this button is unchecked, the user have a full control over the time ratio. When the button is checked, every time the time bar crosses a state, the state time ratio is applied to the visualization. The modification of the time ratio value is only applied when the "Use Scenario State Time Ratio" is toggled from OFF to ON.
- **Slower**: Decreases the time ratio. Two clicks on this button result in a 5 times slower time flow. The current time ratio is displayed next to this button.
- **Faster**: Increases the time ratio. Two clicks on this button result in a 5 times faster time flow.

Timeline

The timeline (2) displays the visualization's time range. Dragging the cursor controls the visualization's current time. While moving the cursor, the current time is kept updated in all client applications synchronously. For finer or coarser-grained time control, the timeline can be zoomed in/out using the mouse wheel.

Time information

A text area (3) displays the current visualization time. The default format is the ISO time format (date and time in UTC). The arrow button circles through other available time formats: CNES julian day (JD1950, fractional days, with reference date January 1st, 1950), modified julian day (MJD, days and seconds, reference date November 17th, 1858) and proect-relative (Project, seconds, reference project start date). The **Edit date...** button pops up a dialog in which the current visualization time can be accurately defined (in all of the above time formats).

Refer to the *Date formats in VTS* for further information on dates.

Timeshifting

Timeshifting is a feature which allows the user the go back in time while the scenario is running and then catch up on the current time. This feature is only enabled on project containing data which are streamed. It has to be activated during *the configuration phase of the project*. Once activated, the streamed data are stored to be resent to all connected clients during the playback phase. To enter in the playback phase, the following actions are possible:

- Press the **Pause** button
- Enter a date which is before the current date of the scenario
- Drag the time cursor to an earlier date

To catch up on the current time of the scenario, users can increase the current time ratio while playback or drag the time cursor to current date of the scenario

5.2.7 Interacting with client applications

The *Timeline* tab displays graphical information on the progress of the visualization.

The *View Properties* and *3D Cameras* tabs allow interacting with client applications. Available actions are hierarchised according to the project structure, for each application.

The *Events* tab allows controlling the visibility of events in client applications. Event types are displayed in a tree structure for applications that support events.

The *Applications* tab allows managing the Broker's client applications.

The *Server* tab displays information, warning and error messages from the Broker or its client applications, provides a log of all messages received by the Broker and sent to its clients, and displays some technical information on all currently connected clients.

5.2.8 *Timeline* tab

The *Timeline* tab displays graphical information on the progress of the visualization.

The contents of the timeline are described in the *Timeline* section of the *Scenario in VTS* chapter.

5.2.9 *View Properties* tab

The *View Properties* tab allows interacting with client applications and defining the properties of all project scenario states.

The contents of this tab are described in the *View properties editor* section of the *Scenario in VTS* chapter.

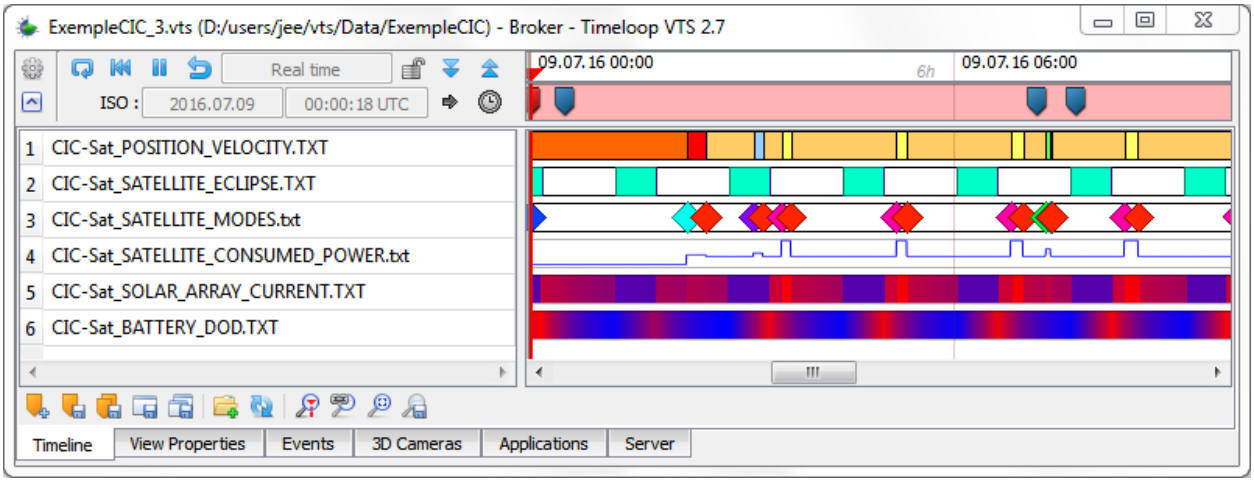


Fig. 5.35: Project timeline

5.2.10 *Events* tab

The *Events* tab allows controlling the visibility of mission events in client applications that support them.

- The list of client applications allows selecting the target client when setting the visibility of mission events.
- The event type hierarchy displays all event types and allows setting the visibility of an event type for all satellites or for each satellite individually. The value indicated in parentheses is the number of events of the corresponding event type in all event files.
- The visibility status of all event types is maintained in the scenario states, and is hence saved between states and visualization sessions.
- Mission events are also displayed in the *Timeline* tab, for each satellite.

For more information on events in VTS, refer to the *Mission events in VTS* chapter.

5.2.11 *3D Cameras* tab

The *3D Cameras* tab allows interacting with standard visualization cameras in 3D applications. The various cameras are displayed in a hierarchical fashion for all visualization entities, and each column corresponds to a specific instance of a currently running 3D client application.

Cameras are positionned relatively to a reference frame axis, such as EME2000 axes, satellite axes, etc. The parameter "Display axes at…" of the section "3D properties of a satellite" also select if the camera is attached to the satellite local frame axis or to its center of gravity.

Central bodies

Central bodies such as **Earth** are top-level entities of the visualized project. All central bodies defined in the project appear in the hierarchy. Available actions are:

- '*Fixed in *Body* frame*': The camera is positioned in the body's local frame, pointed at the body.
- **Inertial**: The camera is positioned in an inertial frame attached to the body, pointed at the body. Expanding this item allows positioning the camera on all axes of the inertial frame.

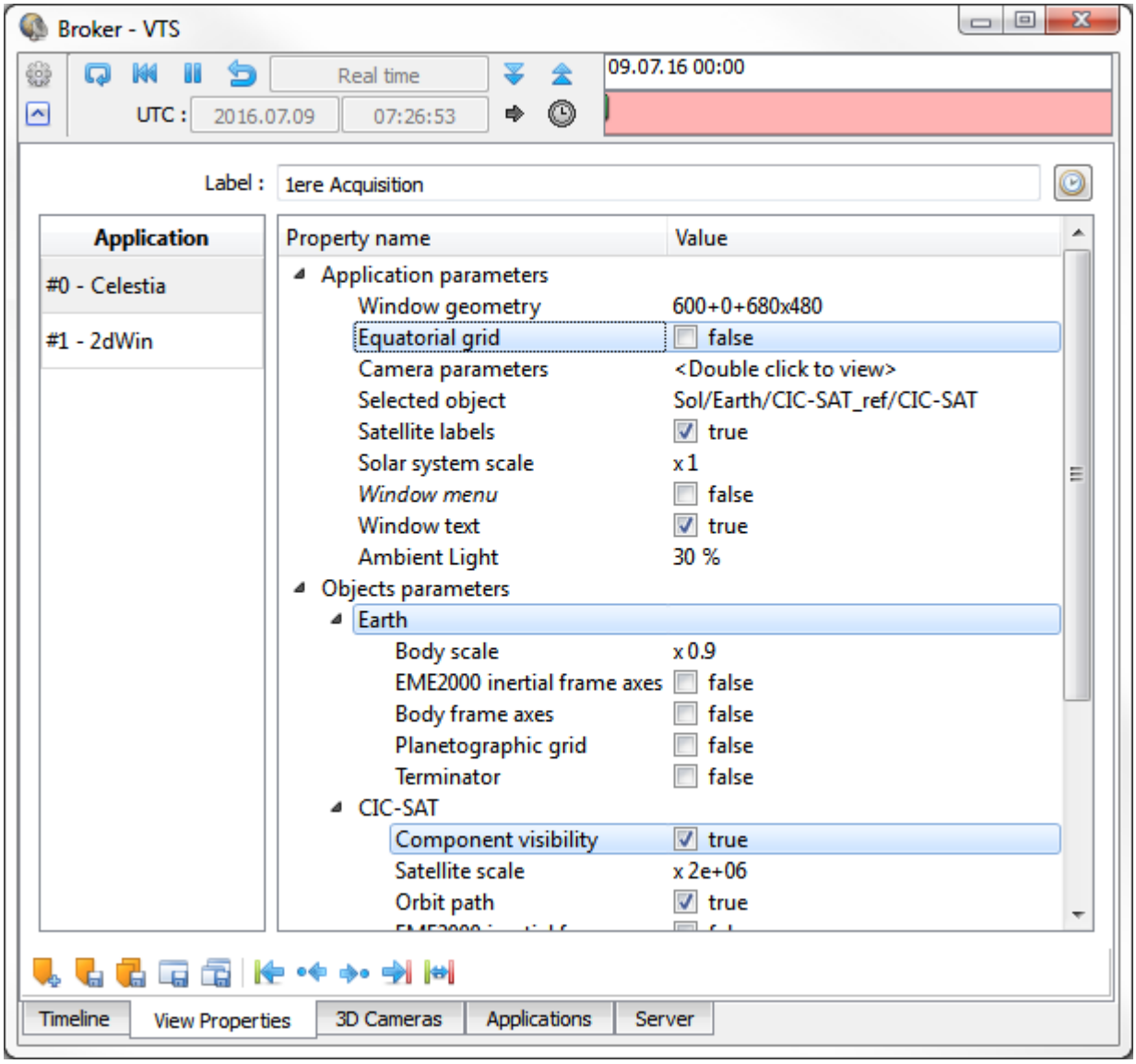


Fig. 5.36: Client application view properties

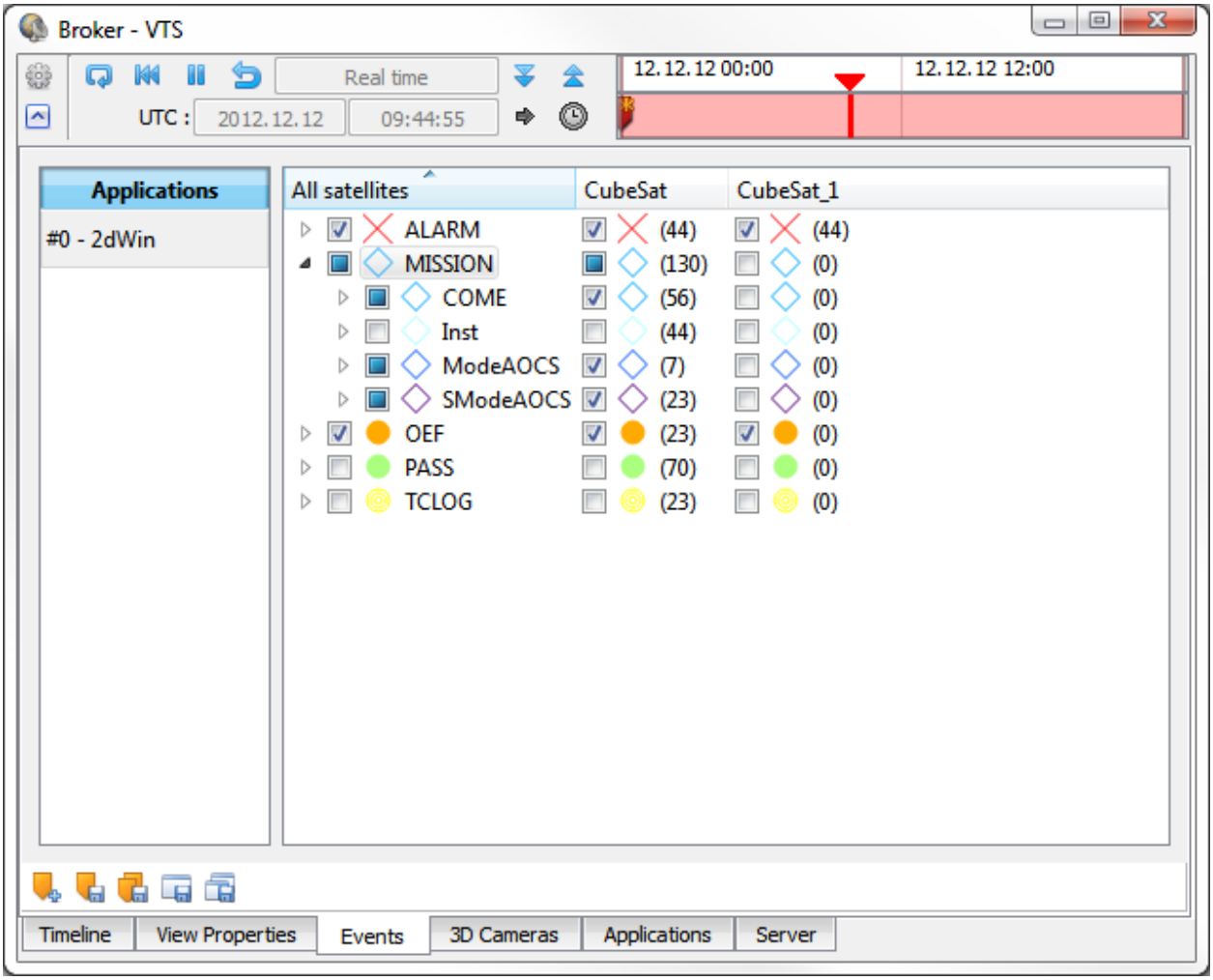


Fig. 5.37: Mission events visibility

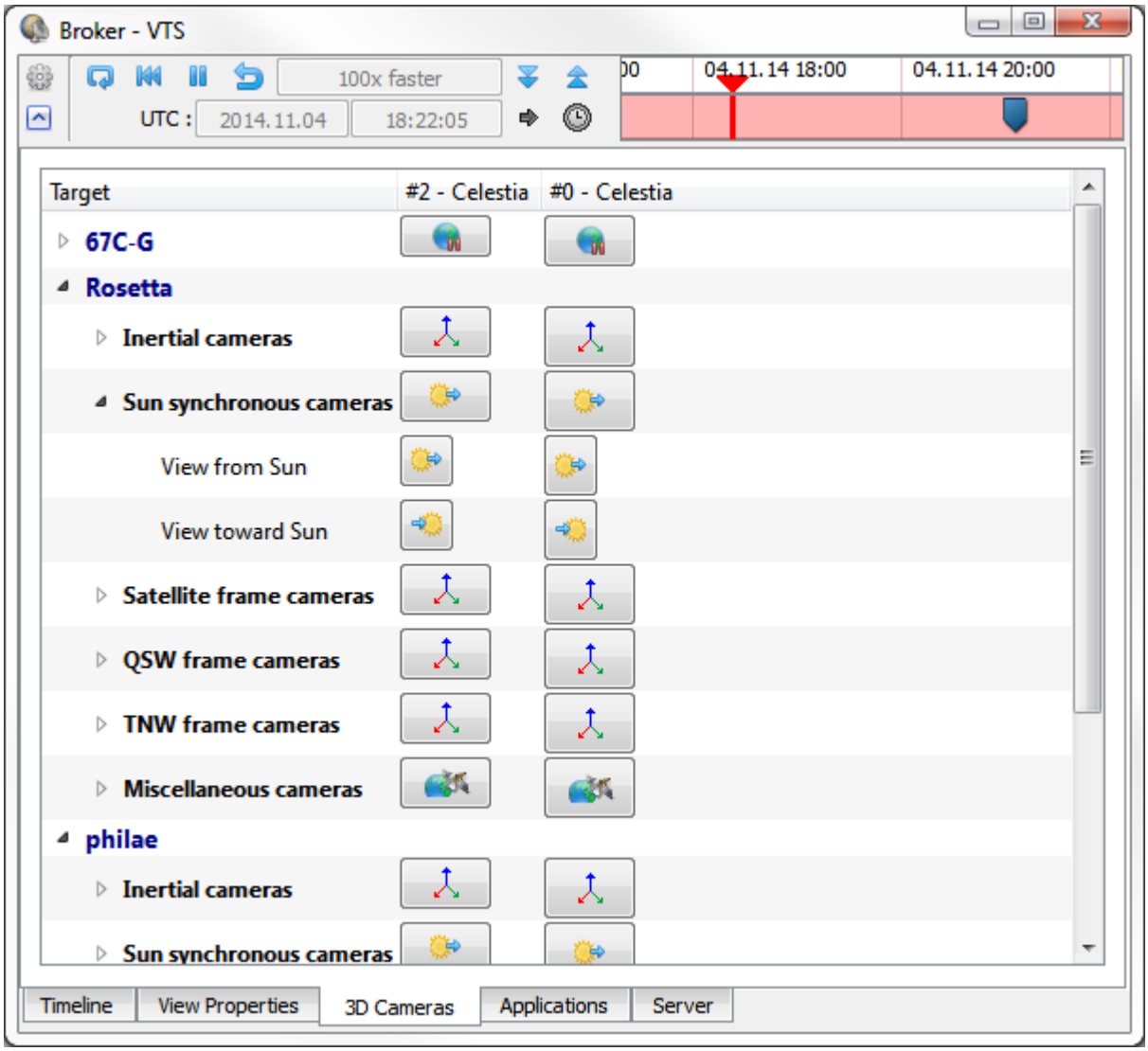


Fig. 5.38: Camera controls

- **Body frame**: The camera is positioned in the body's local frame, pointed at the body. Expanding this item allows positioning the camera on all axes of the body frame.
- **Goto**: The camera is pointed at the body, travelled so that it occupies as much space as possible in the 3D window, and attached to the body's local frame. The UP vector of the camera is undefined.
- **Center**: The camera is simply pointed at the body. Its reference frame is unchanged. The UP vector of the camera is undefined.
- **North pole**: The camera is positioned above the North pole of the body, pointed towards it, and attached to the body's local frame. The UP vector of the camera is along the Y axis of the local frame.
- **South pole**: The camera is positioned above the South pole of the body, pointed towards it, and attached to the body's local frame. The UP vector of the camera is along the -Y axis of the local frame.
- **Ground Stations**: The camera is pointing from or towards a ground station. This entry is available only if a ground stations is configured as targeting a satellite altitude (see the *Configuring a ground station* section).

Satellites

Satellites are top-level entities of the visualized project. All satellites defined in the project appear in the hierarchy. Available actions are:

- **Inertial cameras**: The camera is positioned in an inertial frame attached to the satellite, pointed at the satellite. Expanding this item allows positioning the camera on all axes of the inertial frame.
- '*Sun and *Body* cameras*': The camera is positioned in a Sun/body-synchronous frame attached to the satellite, pointed at the satellite. Expanding this item allows positioning the camera so that it points towards the Sun/body. The UP vector of the camera is undefined.
- **Satellite frame cameras**: The camera is positioned in the satellite's local frame, pointed at the satellite. Expanding this item allows positioning the camera on all axes of the satellite frame.
- **QSW frame cameras**: The camera is positioned in the QSW local orbital frame, pointed at the satellite. Expanding this action allows positioning the camera on all axes of the QSW frame.
- **TNW frame cameras**: The camera is positioned in the TNW local orbital frame, pointed at the satellite. Expanding this action allows positioning the camera on all axes of the TNW frame.
- **Miscellaneous cameras**
  - **Orbit**: The camera is positioned at a distance of the satellite's central body, pointed towards the body in a direction normal to the satellite's velocity, so that the orbit of the satellite can be observed. The UP vector of the camera is aligned on the Z axis of the central body's local frame. Hold the CTRL key while clicking to observe the scene from the opposite side of the body.
  - **Goto**: The camera is pointed at the satellite, travelled so that it occupies as much space as possible in the 3D window, and attached to the satellite's local frame. The UP vector of the camera is undefined.
  - **Center**: The camera is simply pointed at the satellite. Its reference frame remains unchanged. The UP vector of the camera is undefined.
- **Sensors**: These cameras are only available when sensors are attached to the satellite. The camera is positioned at the location of the sensor (plus an offset which can be set in the application parameters), pointed in the direction of its aim vector. Expanding the action allows selecting each sensor of the satellite. The UP vector of the camera is the X axis of the sensor. Hold the CTRL key while clicking to use the Y axis as the UP vector.

Tracking shot

The *tracking* node allows you to perform camera animations in an object frame. The animation will be saved in the current state. It is recommended to create a state beforehand in order to fix the position and orientation of the camera, then another state in which the animation properties will be saved.

1. Create a state, setup a camera (preferably with a camera button and adjust with mouse) an save the state (*Save current state* button)
2. Create a new state a moment later
3. Click the *From* button: the tracking source position is set
4. Move the camera using mouse or another preconfigured camera button **in the same reference frame** (don't mix *Satellite frame* with *TNW frame* position for example)
5. Click the *To* button: the tracking target position is set
6. Set the *Go...* button: a dialog box shows up
7. Configure the animation using the following parameters:

- *Duration*: Number of seconds the goto should take. Default is 5 seconds.
- *Start interpolation*: The point in time during the goto (expressed as a percent number between 0 and 100), at which the observer should begin turning from the initial orientation to the final orientation. Default is 25.
- *End interpolation*: The point in time during the goto (expressed as a percent number between 0 and 100) at which the observer should finish turning from the initial orientation to the final orientation. Default is 75.
- *Acceleration time*: Indicates (as a percentage number between 0 and 100) how much of the time during the goto should be spent accelerating away from the initial position. It also represents the amount of time that will be spent decelerating towards the final position. The remainder of the time during the goto is spend cruising. Default is 25, minimum is 1 and maximum is 50.

8. Click on OK: the animation is played
9. Click on the *Save current state* button to save the animation
10. You can redo any step to modify the animation and save the changes

5.2.12 *Applications* tab

The *Applications* tab allows managing currently connected client applications, as well as starting new clients.

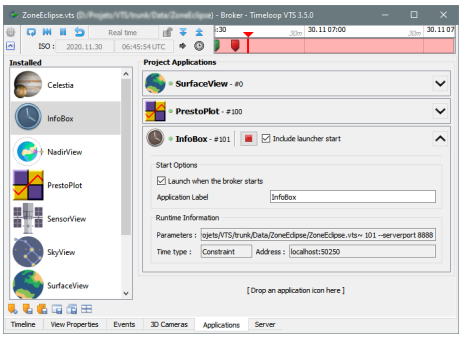


Fig. 5.39: Client applications

Managing client applications

- To start a new instance of a client application, double-click its icon in the *Installed* list or drag-and-drop it into the right area of the window. Clients started this way are listed in the *Dynamic Applications* category.
- Clients not started by the Broker but connected to it are listed in the *External Applications* category.
- **More details about a client can be displayed by clicking the arrow button next to each client.**
  - To stop a client, click the red **Stop** button.
  - To restart a stopped client, click the green **Play** button. A checkbox allows specifying whether or not to execute the client application's launcher before restarting the application, in order to update the client's data.
  - To remove a dynamic application from the list, click the red cross button. The client will be stopped if it is currently running.
  - To change application start options, ensure this application is promoted as a project application. The new options will be applied when you restart the broker. Options are:
    - **Launch when the broker starts**: The application is started just after the broker is started
    - **Application Label** : Title and label of the application

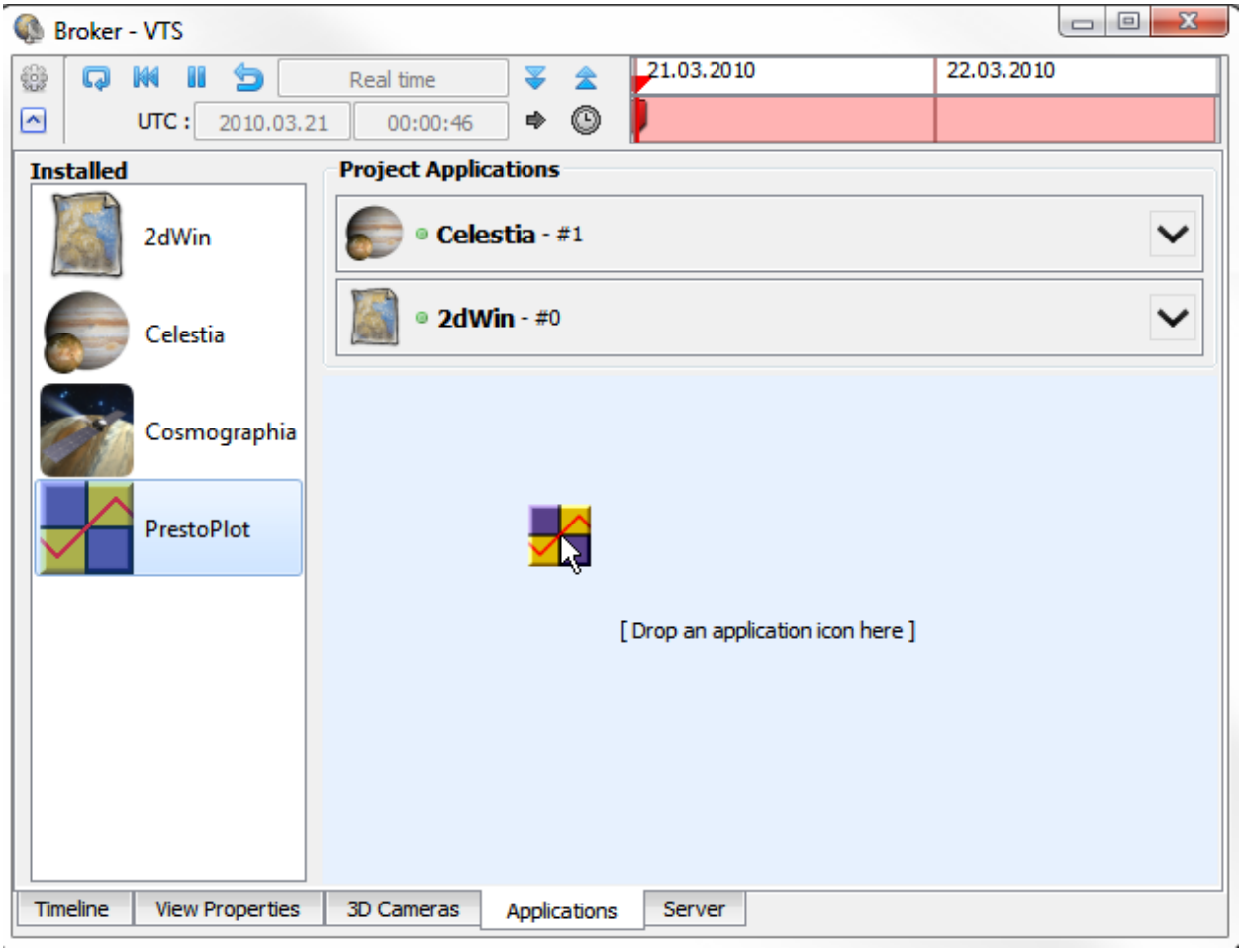


Fig. 5.40: Starting a client by drag-and-drop

Promoting a dynamic application into a project application

Dynamic applications can be promoted to project applications by drag-and-dropping their icons into the *Project Applications* list. When closing the Broker, a pop-up dialog will offer to save the changes made to the VTS project. If changes are saved, the promoted client will appear in the list of project applications in the VTS configuration utility and its view properties will be saved in the project scenario.

5.2.13 *Server* tab

The *Server* tab displays information on the connection and communication of client applications with the Broker.

Log tab

The *Log* tab displays the log of all messages emitted by the Broker or client applications during visualization. Since messages are collected from applications executed in different processes, the order in which the messages are displayed should not be taken as an accurate representation of the order in which the messages have actually been emitted.

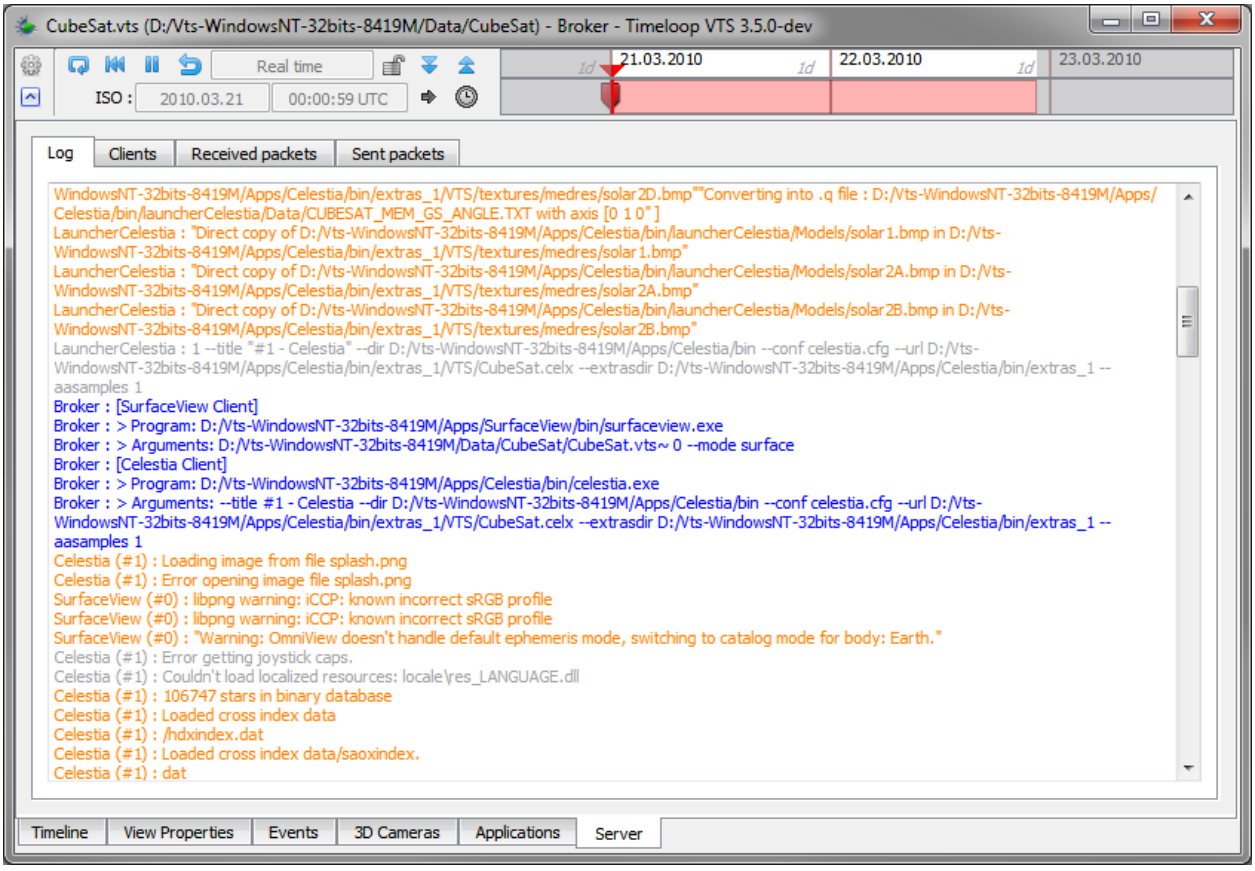


Fig. 5.41: Logged messages

The color code of logged messages is as follows:

- **Blue**: informational messages about the state of the visualization. Only emitted by the Broker.
- **Orange**: warning messages. Only emitted by the Broker. These messages usually appear when an action fails, without causing the visualization to stop.

- **Red**: error messages. Only emitted by the Broker. These messages usually appear when a client application crashes.
- **Grey**: messages printed to the standard output of client applications. These messages are prefixed with the name of the client application that emitted them. They can be either informational, warning or error messages.

Logged messages are also stored in a log file (broker.log). This file can be found in the "Apps/Logs" directory or in the temporary directory (if readonly is on).

Note: Always remember to send a copy of the message log to the VTS support team when an error occurs during visualization.

Clients tab

The *Clients* tab displays the connection state of client applications.

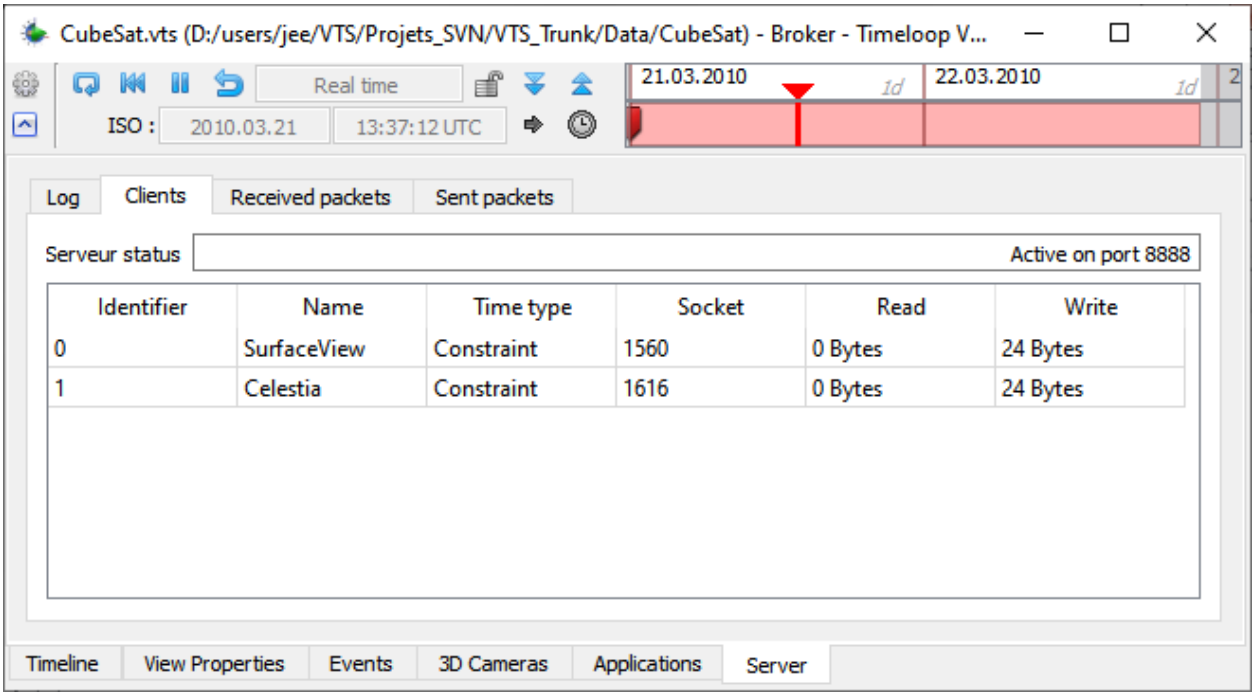


Fig. 5.42: Connection state of client applications

- **Server status**: indicates the TCP port used by the Broker.

The following information is available for each client:

- **Identifier**: the client's ID
- **Name**: the client application's name
- **Time type**: the time behaviour of the client (*Constraint* or *Regulating*). Refer to the *Synchronization protocol for VTS clients* section for further information.
- **Socket**: the socket ID for Broker-client communication.
- **Read**: Number of incoming bytes that are waiting to be read from this client.
- **Write**: Number of bytes that are waiting to be written to this client.

Received packets and *Sent packets* tabs

The *Received packets* and *Sent packets* tabs display the messages respectively received and sent by the Broker, from or to client applications.

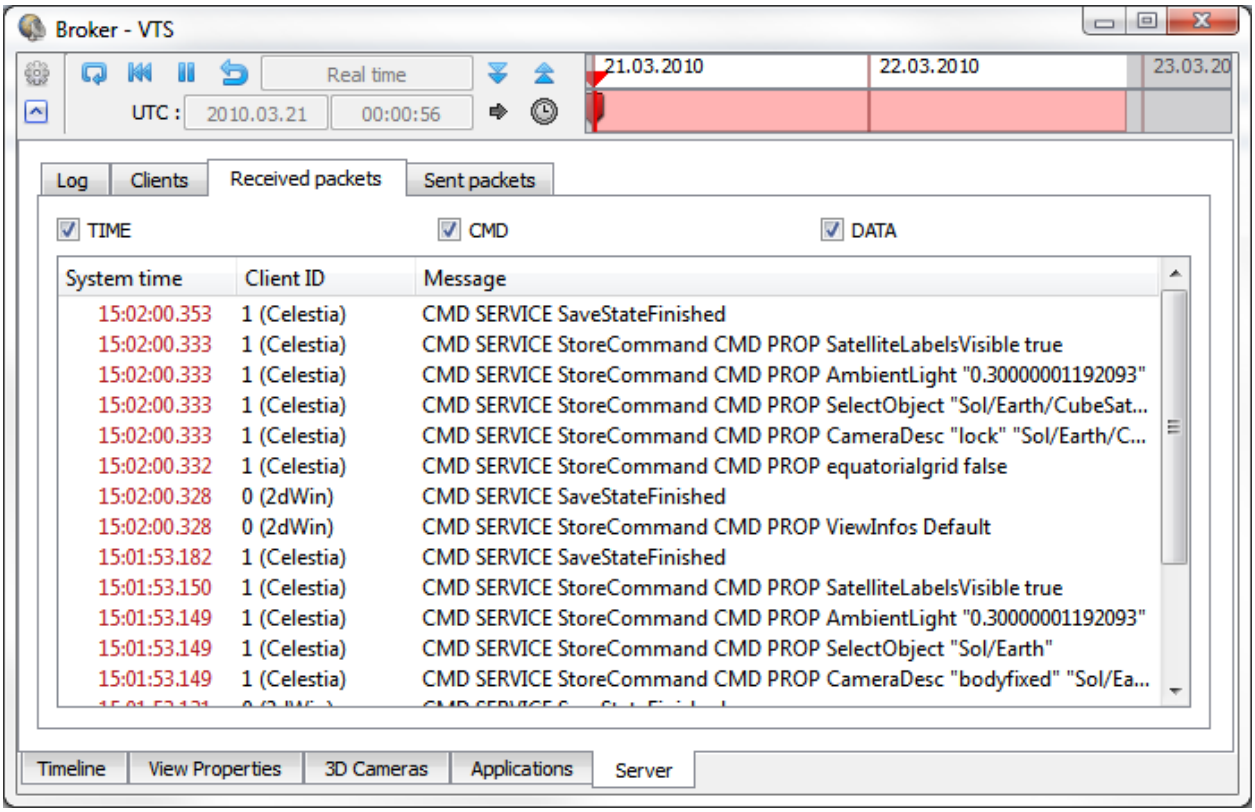


Fig. 5.43: Messages received from clients

TIME messages deal with time synchronization of clients, while *CMD* messages are related to client commands. All messages are displayed truncated when they are too long. Further information on these messages can be found in the *Synchronization protocol for VTS clients* section.

The **TIME**, **CMD** and **DATA** checkboxes allow enabling or disabling the display of these classes of messages in the list.

5.2.14 Recording movies

Movies can be recorded by the Broker during visualization. When recording a movie, client applications are synchronized frame by frame and an image is captured at each frame. The output movie is set at a framerate of 25 images per second.

Note that recording a movie is not possible when the Broker is in real-time mode. For more information on real-time mode, refer to the *Real-time VTS* section of the *Synchronization protocol for VTS clients* chapter.

To start recording a movie, click the **Record movie** entry in the Broker's action menu:

A configuration dialog pops up and visualization is paused. Interaction with the Broker is disabled while the dialog is opened, which means that the visualization must be correctly configured before starting the recording.

The following parameters must be provided:

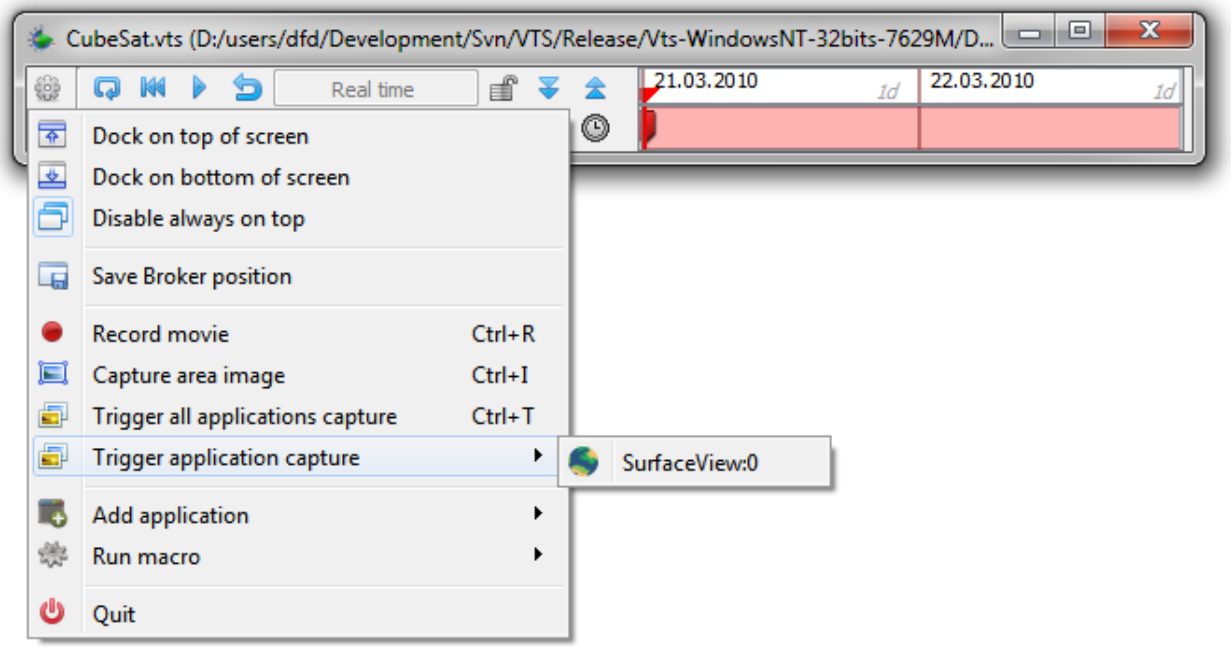


Fig. 5.44: Recording a movie

- **Output file**: Name of the movie output file. The file extension will be either *.avi* (MPEG-4) or *.mpg* (MPEG-2). For correct integration of the recorded movie within Microsoft PowerPoint, the *.mpg* format must be selected.
- **Preset**: List of standard movie definitions. A *Custom* definition allows manual specification of the movie width and height.
- **Width**, **Height**: Movie definition (the mouse wheel may be used on these fields for faster adjustment of the values). For technical reasons linked to the requirements of the encoding codec of the movie, these values must be multiples of 8. If not, they will be automatically rounded up to the nearest multiple of 8 at the start of the recording.
- **X offset**, **Y offset**: Position of the recorded area (the mouse wheel may be used on these fields for faster adjustment of the values).
- **Overlay: An overlay could be added on the movie to show an image in the foreground.**
  - **Standard** combobox: A list of standard logos. You can also put your own image files in the Apps/Broker/overlays directory.
  - **Custom image file**: Path of an image to be used as an overlay for the movie. If not path is given, the movie will not contain any overlay.
  - **Anchor**: Anchor of the image in the recording area. By clicking on the icon, it shows the different possibilities for anchors.
  - **X offset**, **Y offset**: Position of the image from the anchor position in the recording area.
  - **Side**: The largest side of the image used for scaling
- **Quality** (*Low*, *Medium*, *High*): Quality factor determining the bitrate of the recording.
- **Pause visualization when starting recording**: Option allowing to start the recording paused at the current visualization time.

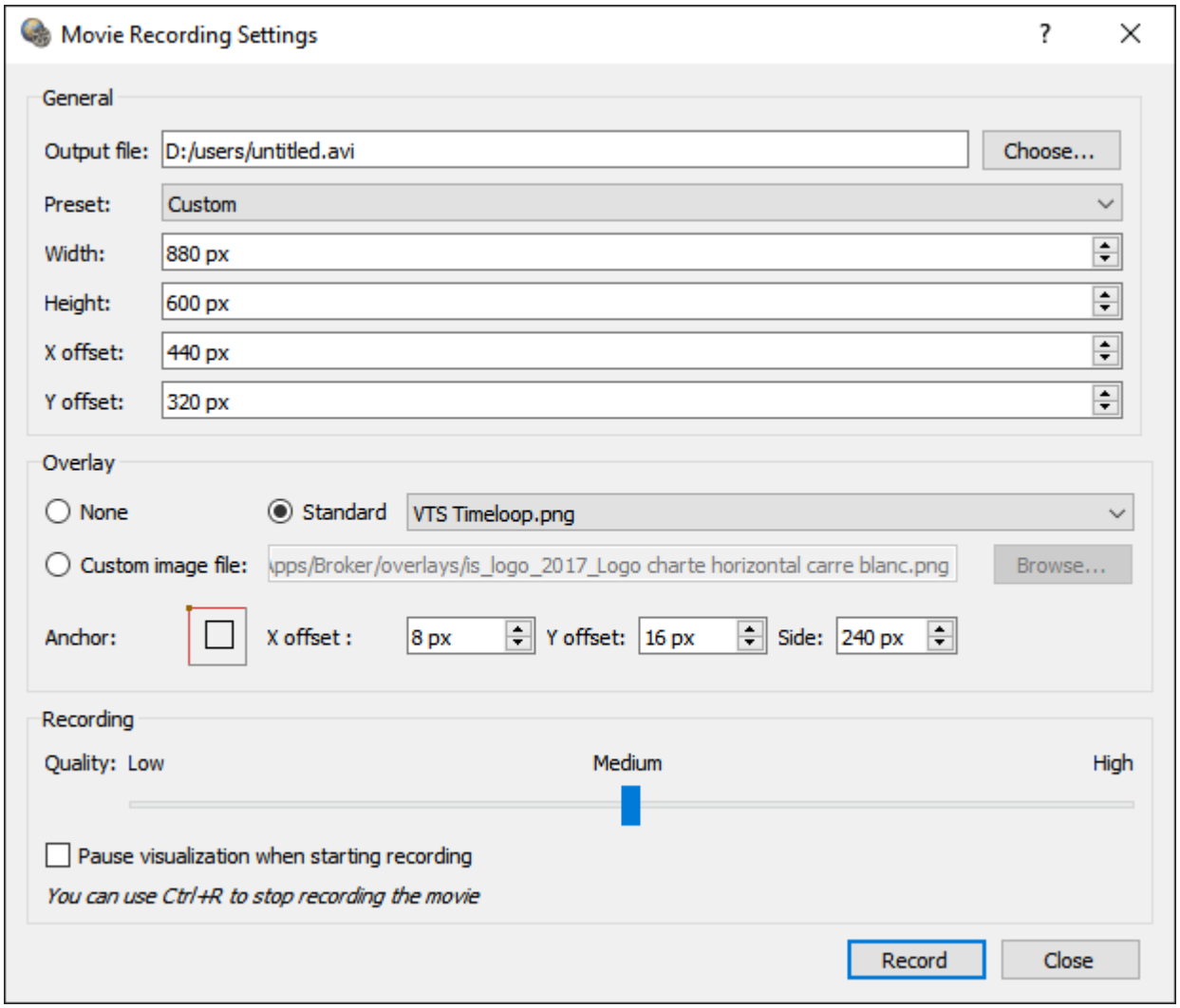


Fig. 5.45: Recording settings

Notes:

- The movie bitrate is proportional to the movie definition.
- The quality factor increases or decreases the quality of the movie. A *Low* quality factor produces compression artifacts on moving parts in the video, but smaller file size. A *High* quality factor produces good quality video, but significant file size.

The recording can be stopped with the **Stop recording** entry in the Broker's action menu. The *Ctrl+R* keyboard shortcut can also be used when the Broker window is focused.

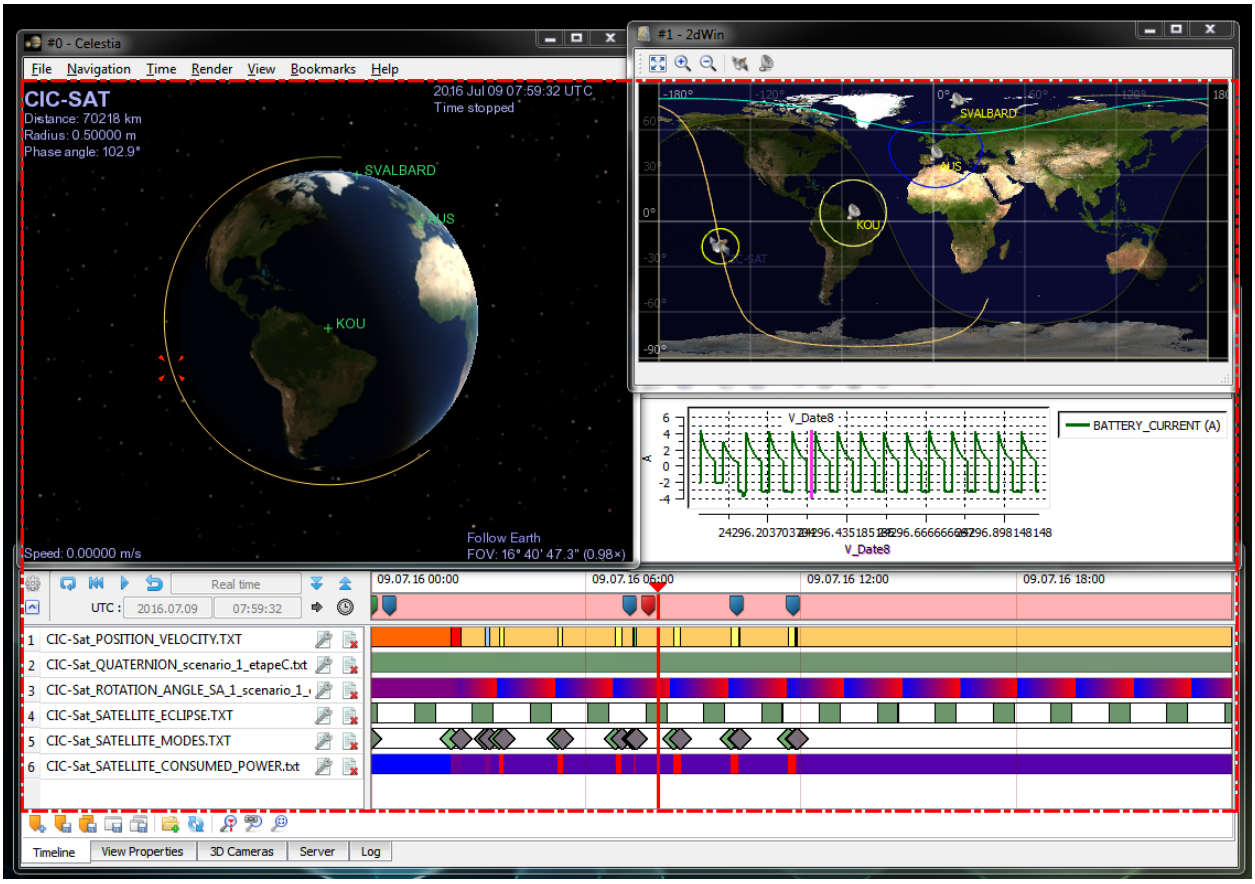


Fig. 5.46: Recording area

The screen area located within the red frame is recorded in the movie. The mouse pointer is not recorded. During the recording, it is advised not to move recorded windows and to use camera states rather than manual camera movements. Otherwise, since the movie is not recorded at real-time speed, camera movements may appear to be slower or faster than expected.

During the recording, additional information is displayed in the *Log* tab of the *Server* tab of the Broker:

- **Bitrate**: the current movie bitrate
- **Movie length**: the current length of the recorded movie
- **Recording FPS**: the current record framerate (this does not affect the output framerate)

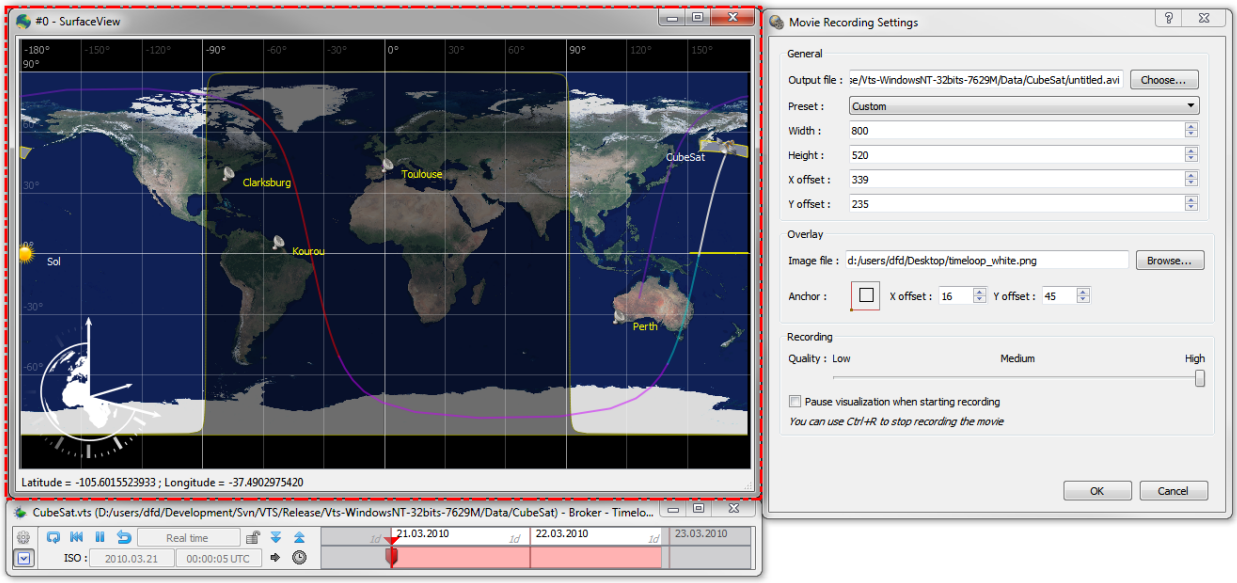


Fig. 5.47: Recording of a SurfaceView with an overlay at the bottom left of the recording area.

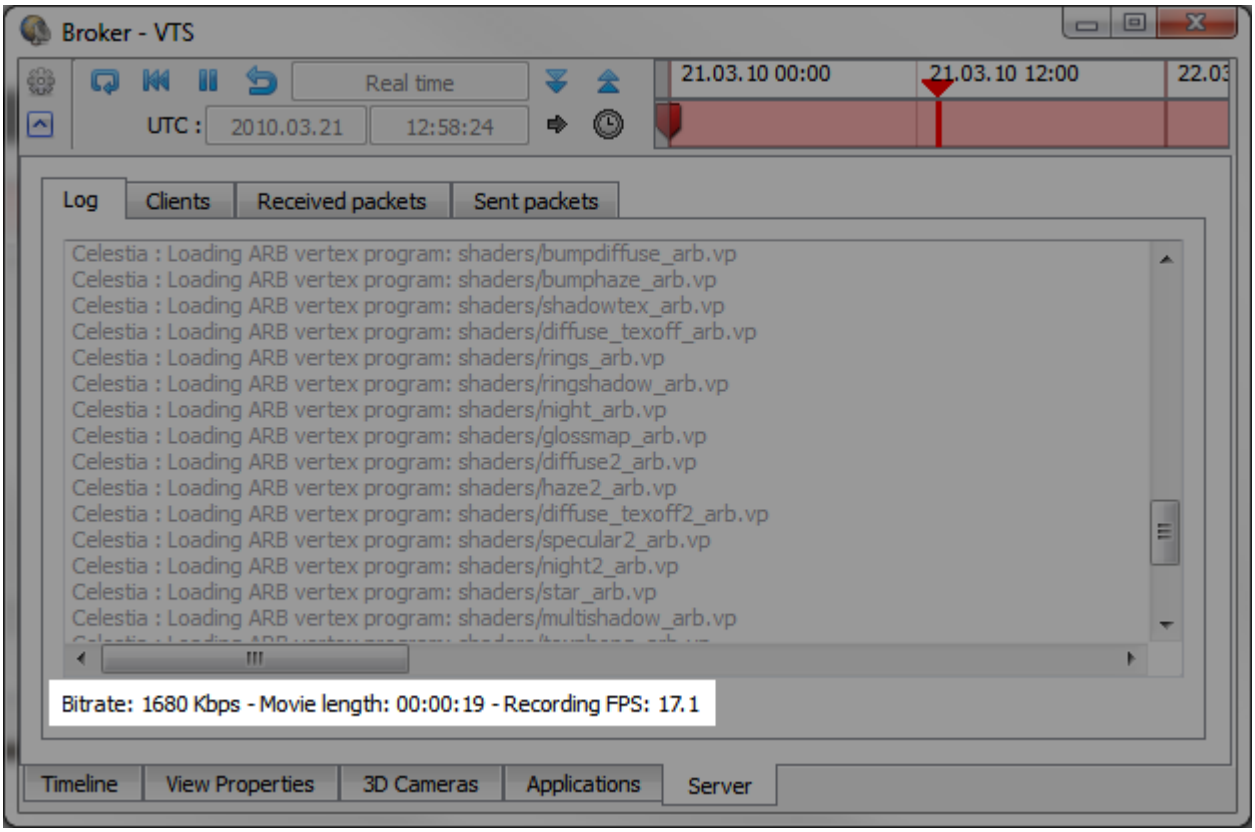


Fig. 5.48: Recording information

5.2.15 Other interactions

The *Ctrl+Shift+C* keyboard shortcut copies to the clipboard the full path to the VTS project file.

5.2.16 Image capture

Images can be captured by two different ways. On the one hand the Broker can capture the desktop by defining an area. This way, all applications can be captured, including the Broker. On the other hand, the Broker can request each application to capture itself. If the application handle the "TakeScreenshot" VTS protocol command, an image will be produced as definied in the application (usually without window decoration).

Area capture

To open the capture settings click the **Capture Area** entry in the in the Broker's action menu:

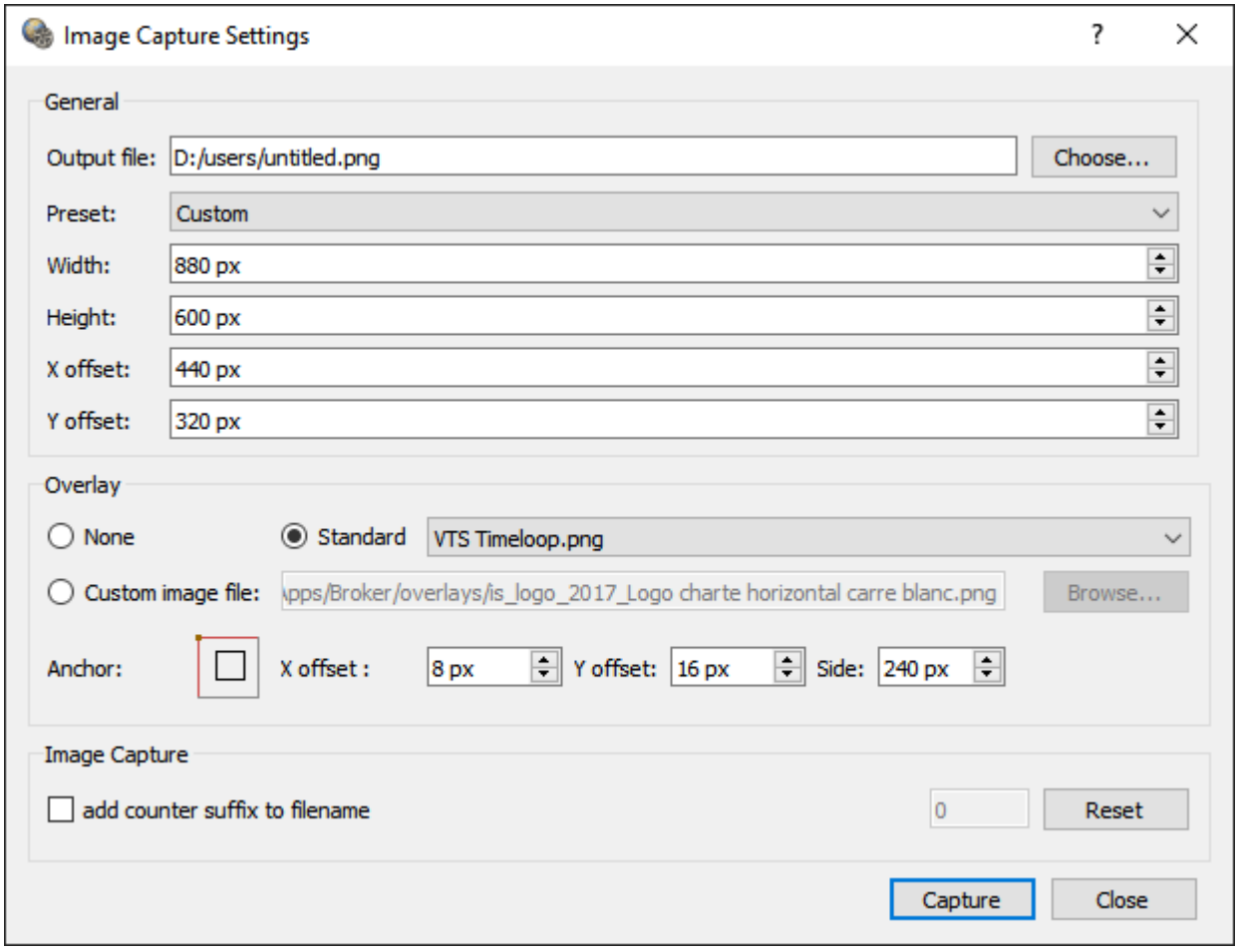


Fig. 5.49: Area capture

The following parameters must be provided:

- **Output file**: Name of the image output file.
- **Preset**: List of standard definitions.

- **Width**, **Height**: Image definition (the mouse wheel may be used on these fields for faster adjustment of the values).
- **X offset**, **Y offset**: Position of the captured area.
- **Overlay**: An overlay could be added on the cpature to show an image in the foreground. See the *Recording Movies* section for more details.
- **add counter suffix to the filename**: When checked, a counter will be added to the filename. Click on **Reset** to start it over.

Click on the **Capture** button to capture the image. The settings window will briefly disappear during the image capture.

Capture by application

Using the **Capture All Applications** or the **Capture Application** entry, a VTS protocol command "CMD SERVICE TakeScreenshot <fileInProjectFolder>" will be sent the concerned applications. Each application can handle this command in different ways. Captured images will be produced in the project folder.

The *Ctrl+T* keyboard shortcut can also be used to send a **Capture All Applications** request when the Broker window is focused.

This command might not be handled by some applications so this action is not guaranteed contrary to the area capture method. All applications bundled with VTS handle the TakeScreenshot command, excepting PrestoPlot. See the *TakeScreenshot command* section of the *Synchronization protocol for VTS clients* chapter for more information about this command.

5.3 SurfaceView user manual

The SurfaceView client application displays a planisphere of the project's main central body in different projections.

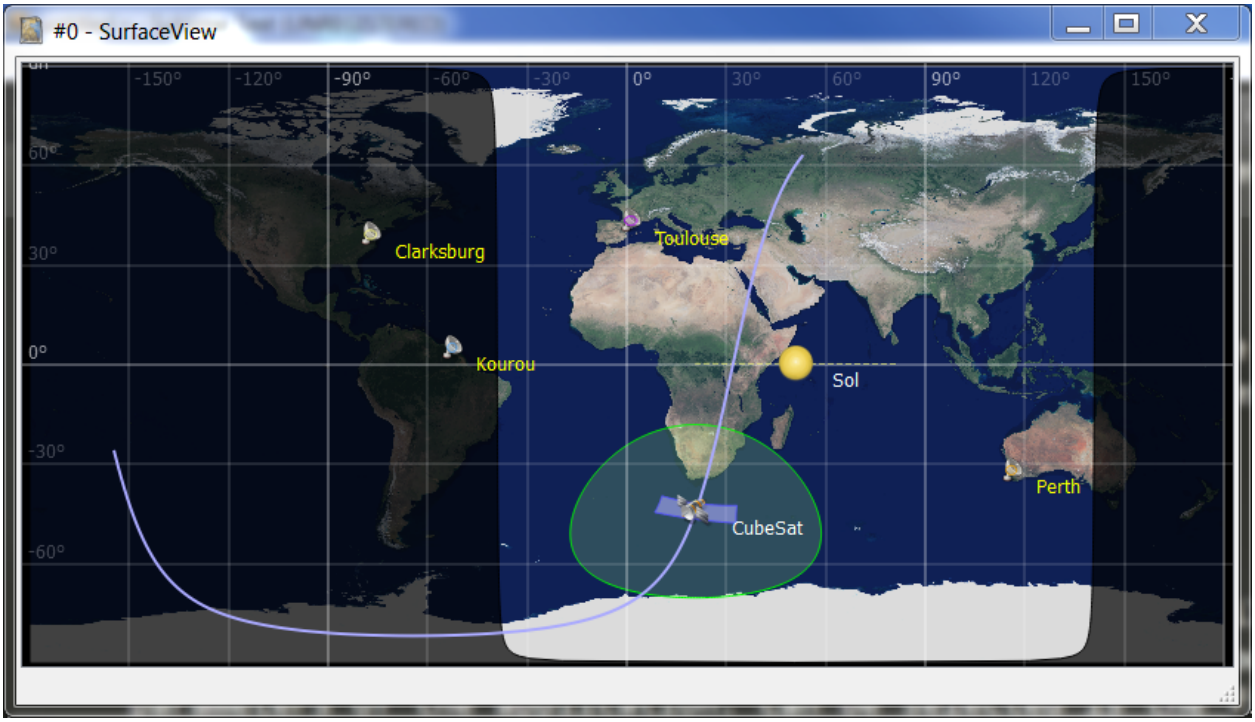
5.3.1 Introduction

SurfaceView displays a 2D scene based on a projection of one of the bodies of the VTS project tree. Other entities defined in the project tree (such as satellites, ground stations or other celestial bodies) are represented as 2D items over their projected position.

What is displayed in SurfaceView

The following items are displayed on the planisphere of SurfaceView:

- ***Main body*** The plate carre projection of the main body's surface is displayed in the background of the scene.
- ***Latitude/Longitude grid*** The grid automatically adapts to the current zoom level. The latitude and longitude values are displayed respectively at the left and top of the window.
- ***Sun terminator*** The fictional line that separates the sunlit and obscured faces of the body is displayed in yellow and the obscured face appearing darker than the sunlit one. Refer to the Configuring the sun terminator section in the VTS configuration utility user manual chapter for more information.
- ***Subsolar point*** The the point at the surface of the body where the Sun is at the zenith is represented as an icon. Refer to the Configuring the sun icon section in the VTS configuration utility user manual chapter for more information.



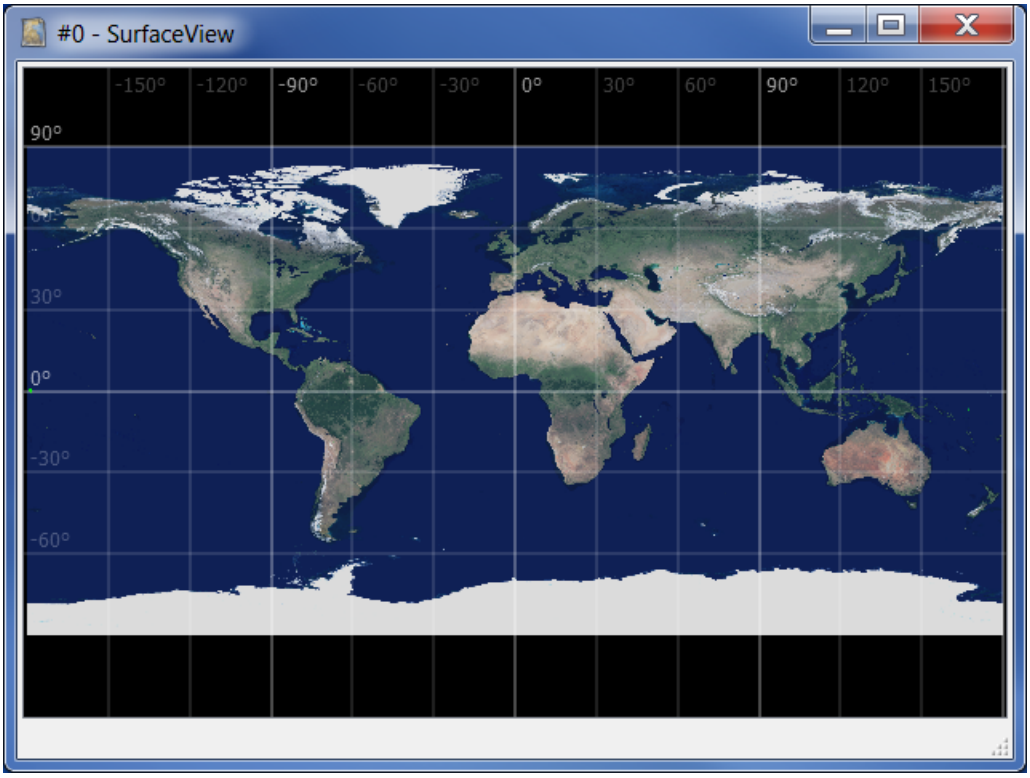


Fig. 5.50: Interacting with SurfaceView

- ***Celestial bodies*** Celestial bodies different from the main body are displayed as icons over their projected position.

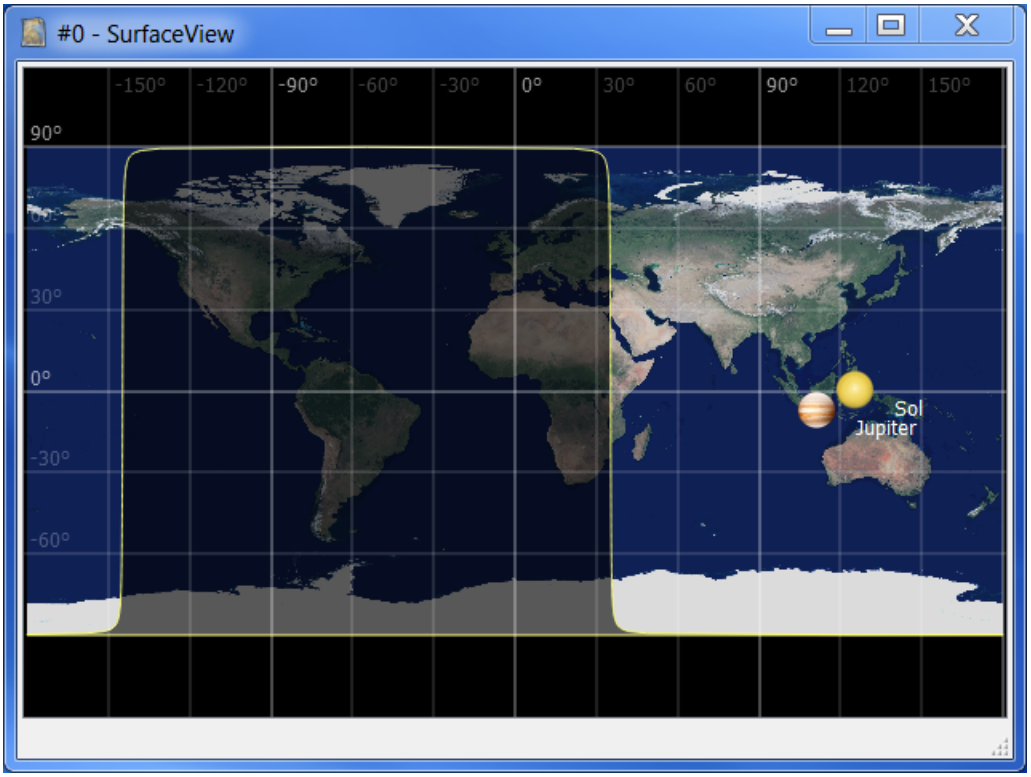


Fig. 5.51: SurfaceView Sun terminator and other celestial bodies

- ***Satellites*** Satellites orbiting the main body are displayed as icons over the projection of their orbit position. Icons are automatically oriented in the direction of the satellite motion.
- ***Satellite sensor footprint*** The satellite sensor footprint is the intersection of the satellite sensor's aim volume with the surface of the body. Refer to the Configuring a satellite sensor section in the VTS configuration utility user manual chapter for more information.
- ***Satellite / Sun / Celestial bodies ground track*** The ground tracks of the sun, satellites and other celestial bodies appear before and after each entity. Their lengths can be defined in the VTS configuration utility. In SurfaceView, the tracks are displayed in the body's local frame (as opposed to an inertial frame in Celestia).
- ***Satellite / Celestial bodies visibility region*** The visibility regions of satellites and other celestial bodies is the region of the earth visible from the current position of the entity. Their graphic properties can be defined in the VTS configuration utility.
- ***Ground stations*** Ground stations attached to the main body. Refer to the Configuring a ground station section in the VTS configuration utility user manual chapter for more information.
- ***Station sensor footprint*** The station sensor footprint is the intersection of the ground station sensor's aim volume with a virtual sphere at a specified altitude. This altitude can be either fixed or linked to the satellite altitude. Refer to the General properties of a ground station section in the VTS configuration utility user manual chapter for more information.
- ***Geometrical visibility*** The geometrical visibility between a satellite and a ground station targeting its altitude. Refer to the General properties of a ground station section in the VTS configuration utility user manual chapter for more information.

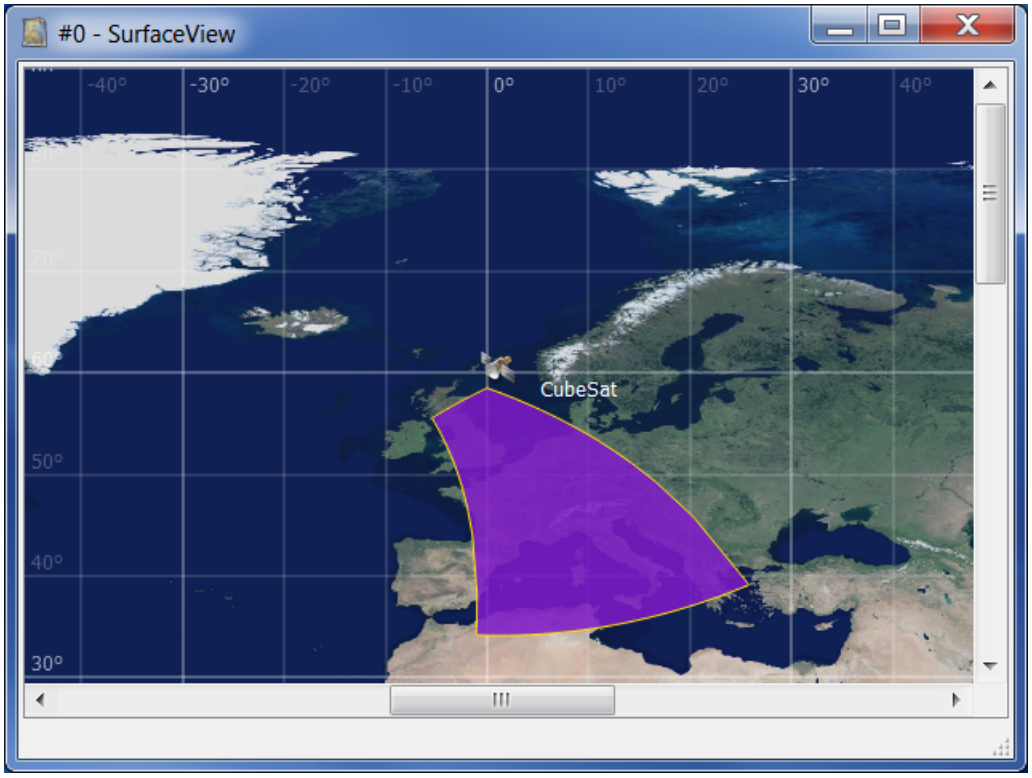


Fig. 5.52: SurfaceView Satellite and sensor footprint

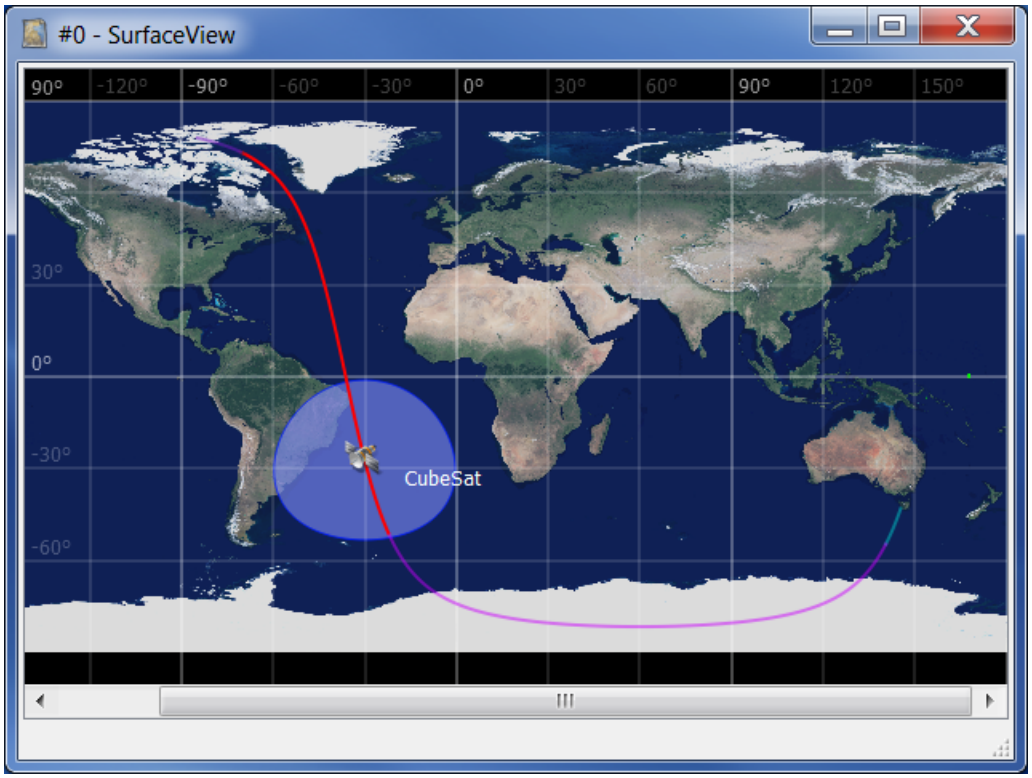


Fig. 5.53: SurfaceView Satellite track and visibility circle

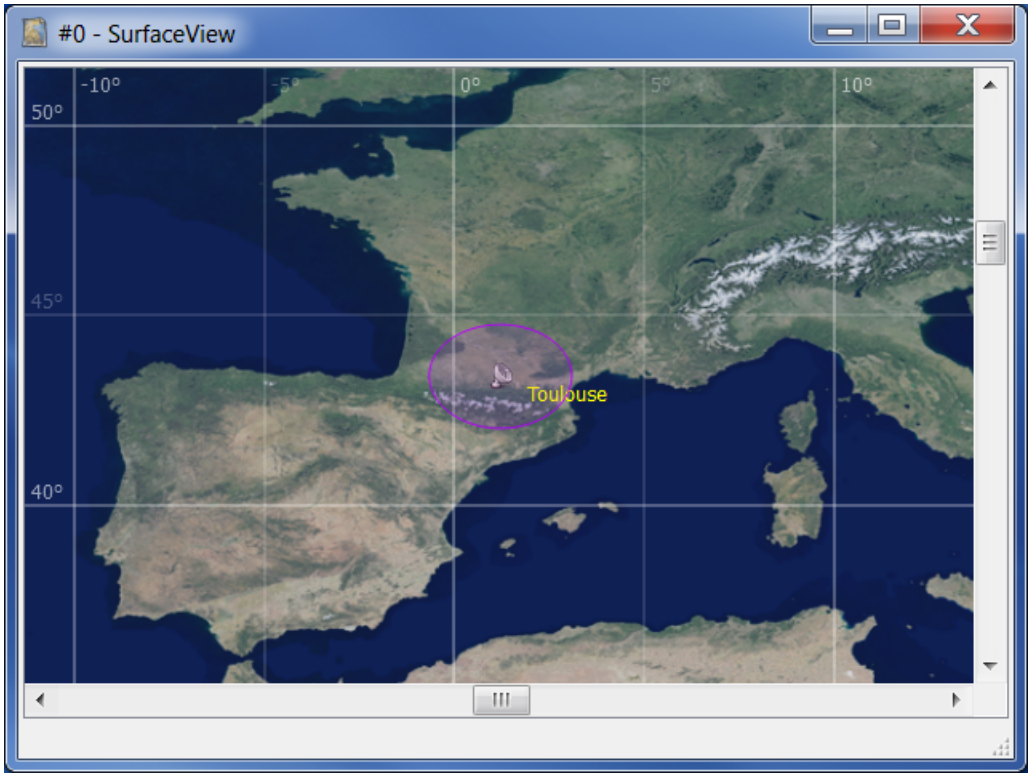


Fig. 5.54: SurfaceView Ground Station

- ***Regions of Interest (ROI)*** Regions Of Interest attached to the main body. Refer to the Configuring a Region Of Interest section in the VTS configuration utility user manual chapter for more information.
- ***Points of Interest (POI)*** Points Of Interest attached to the main body. Refer to the Configuring a Point Of Interest section in the VTS configuration utility user manual chapter for more information.
- ***Mission events*** The mission events attached to all satellites of the main body. Mission events appear at the date of their occurrence on the ground track of the satellite they are attached to. Refer to the Events attached to a satellite section in the VTS configuration utility user manual chapter for more information.
- ***Clusters*** Clusters are attached to the main body and represent a population of objects.

5.3.2 Configuration in VTS

The entities and properties defined in the VTS configuration utility will be displayed in SurfaceView. The main body can be defined using the initial parameters of the application, otherwise the first central body defined in the VTS project tree will be used. Refer to the *Configuring a central body* section in the *VTS configuration utility user manual* chapter for more information.

5.3.3 Application parameters in VTS

When adding SurfaceView as a VTS client application, some parameters can be set by clicking on the SurfaceView entry in the VTS project tree:

- ***Scene Reference*** This combo box allows you to select the main body from the list of celestial bodies defined in the VTS project tree. The selected body will become the scene reference and in the SurfaceView instance

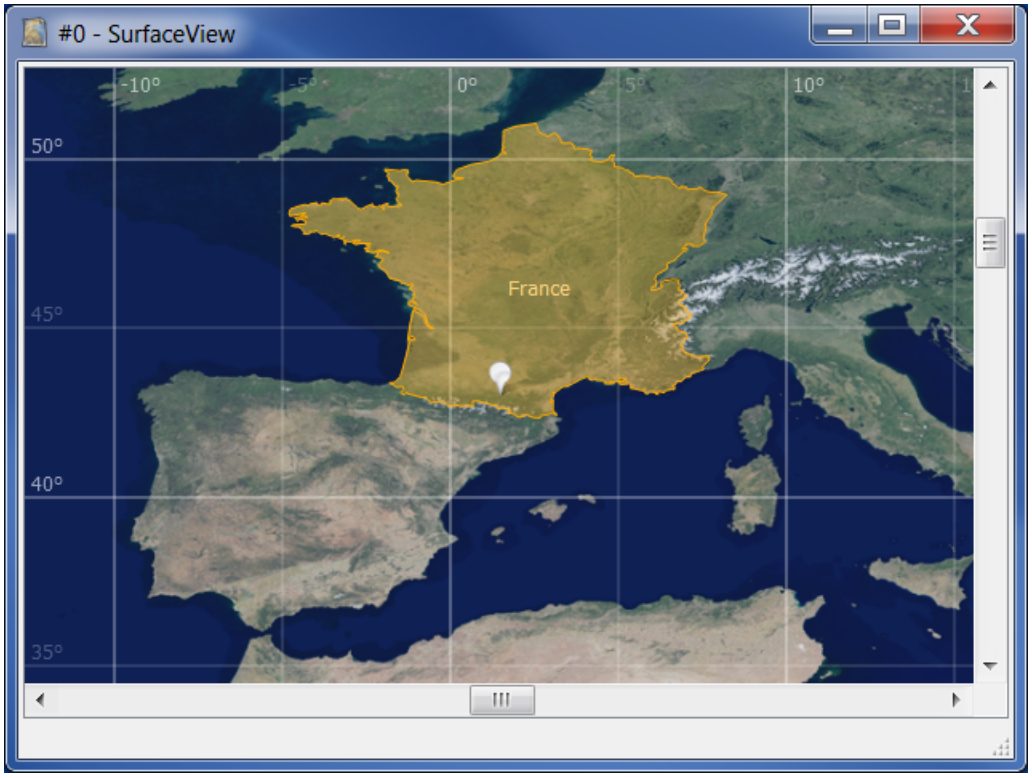


Fig. 5.55: SurfaceView ROI and POI

and the position of all other entities will be projected on it. By default, the first body from the VTS project tree is selected as a scene reference.

- ***Map Projection*** This combo box allows you to select the world map projection that will be used in the SurfaceView instance. Currently supported projections are Equirectangular (plate carree), Mercator and Polar.
- ***Target framerate*** The application target framerate (frames per seconds). A higher value improves the accuracy but slows down the application. An accurate simulation can be achieved with a framerate of 40, however framerates higher than 50 will make little difference in the accuracy.
- ***Track sampling resolution*** This parameter allows you to modify the sampling resolution of all the tracks in VTS. Tracks represent the trajectory of different objects (satellites, celestial bodies, stars). Decrease this value to improve track accuracy at the expense of performance degradation.
- ***Sensor geometry section count*** Number of points making up the outline polygon of the aiming sensor surface. Increase this parameter if you want to focus on sensor swath accuracy. Decrease it for long missions coverage to improve performance.
- ***Sensor coverage significant threshold*** The sensor swath is composed by a set of instantaneous sensor aiming surfaces represented by polygons. An instantaneous aiming surface is retained when a significant percentage of new coverage is provided by the new surface. This parameter sets the percentage of novelty (minimum threshold).
  - Use a small value for an agile satellite.
  - Use a high value for a long coverage mission.

  This parameter has no effect if the Sensor coverage nadir optimization is enabled.
- ***Sensor coverage merging threshold*** The sensor residual trace is composed by many polygons. Its length is maintained by deleting polygons at the end of the tail. For better performance, polygons are merged

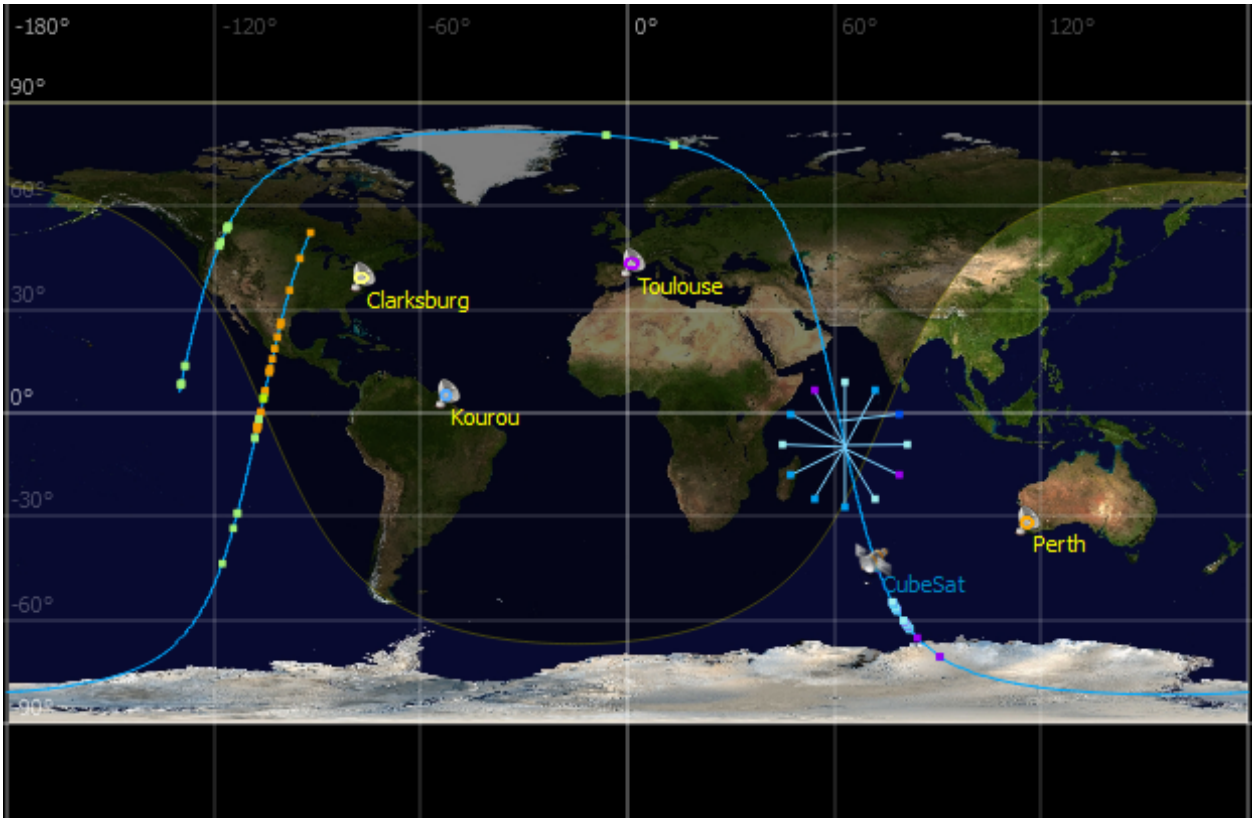


Fig. 5.56: SurfaceView mission events attached to a satellite

Initial States

Property name	Value
Application parameters	
Scene reference	Default
Map Projection	Equirectangular
Target framerate	10
Earth Tilemap URL	../Celestia/bin/extras/BMNG/textures/hires/BMNG
Sensor geometry section count	128
Sensor coverage significant threshold (%)	5
Sensor coverage merging threshold (%)	90
Sensor coverage nadir optimization	☐ false

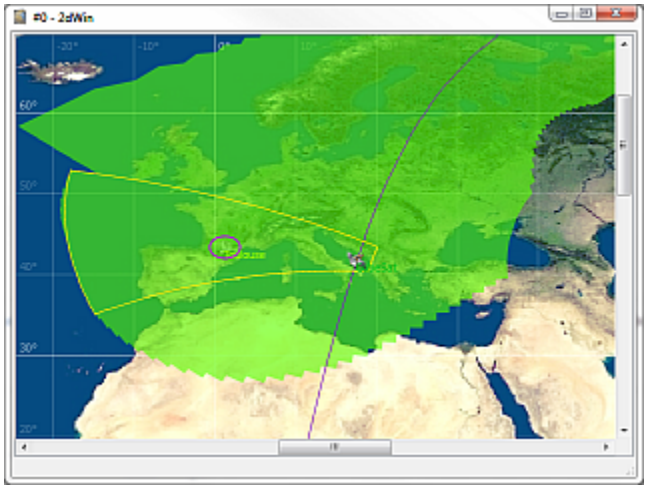


Fig. 5.57: A significant threshold of 5% of novelty is ideal for an agile satellite maneuver.

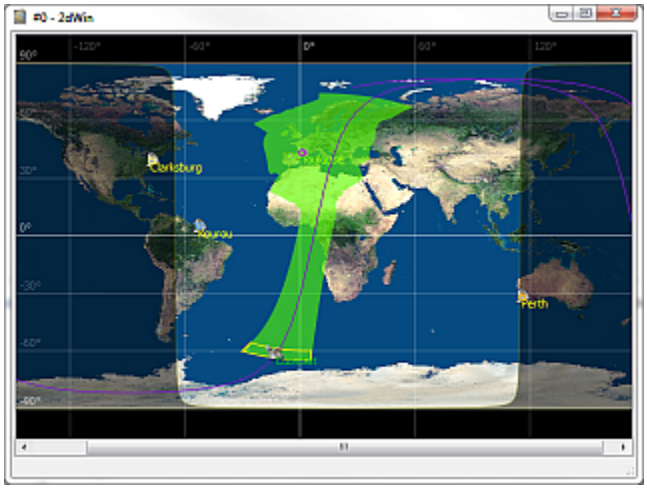


Fig. 5.58: A small threshold works well for long coverage missions but slows down the display.

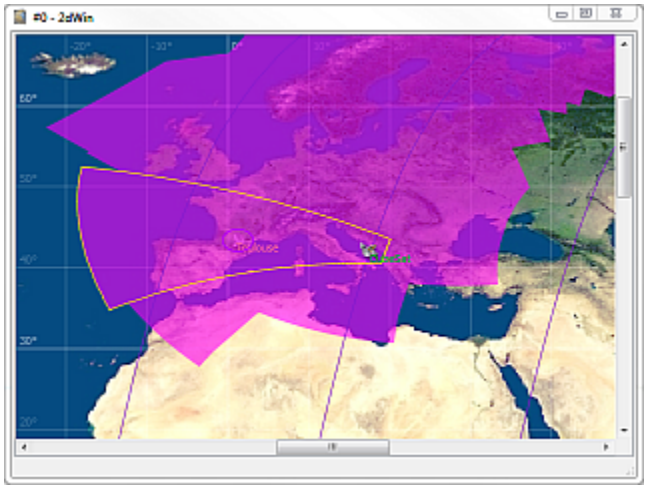


Fig. 5.59: A significant threshold of 40% gives poor results during agile satellite maneuvers.

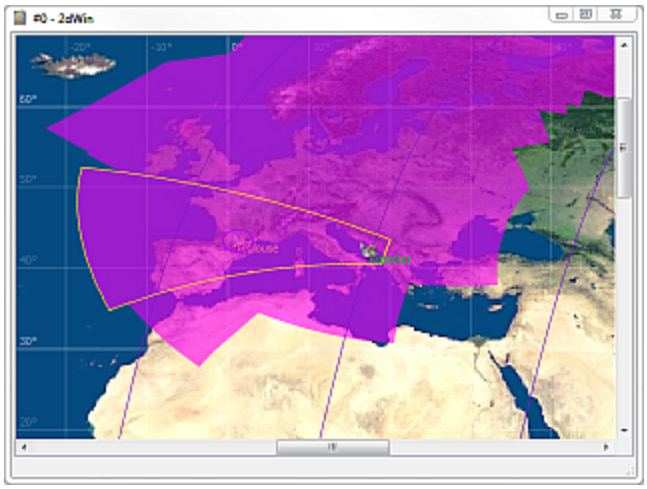


Fig. 5.60: A high threshold gives good results for long coverage missions with good performance.

together when two polygons have at least a certain percentage of area in common. This parameter sets the percentage of novelty (minimum threshold).

- Choose a small value when the length of the sensor residual trace must be accurate (0% of merging will keep all polygons separated).
- Choose a high value for better performance (99% of merging will merge huge blocks of polygons, which will be removed one by one, making the residual trace length vary a lot).
- Choose the special value of 100% to only merge by orbit path color : this mode is useful for long coverage missions, where only the current orbit pass should be displayed.

This parameter has no effect if the Sensor coverage nadir optimization is enabled.

- ***Sensor coverage nadir optimization*** For satellites pointing nadir, this optimization improves significantly the performance of the sensor coverage calculation at the cost of reducing the accuracy: the computed coverage is less accurate but the simulation is faster and it shows satellites coverages overlap. If the optimization is disabled, the sensor coverage will follow the exact sensor surface.

  If the optimization is enabled, the Sensor coverage significant threshold and Sensor coverage merging threshold parameters will have no effect in the application.

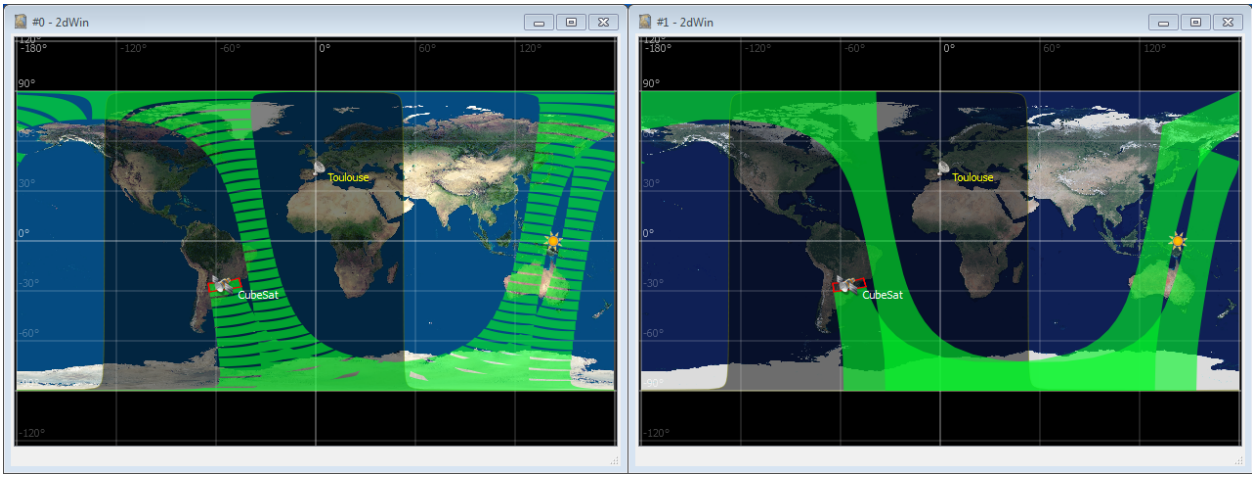
  Refer to the *Coverage swath* section for more information.

5.3.4 Coverage swath

SurfaceView has an option to optimize the coverage swath of satellites pointing nadir.

Without the nadir optimization, the sensor coverage is exact because the sensor surface is used as a stamp. However, the coverage could be incomplete if the time ratio is high enough (some sensor stamps will be missed). By the default, the coverage is kept in memory during one hour.

With the nadir optimization, the sensor coverage will be always complete even if the time ratio is high and it will allow the simulation of a higher number of satellites' coverages without slowing down the application. However, the accuracy of the coverage can be reduced at the poles or if the satellite is not pointing nadir. The nadir optimization is enabled by default.

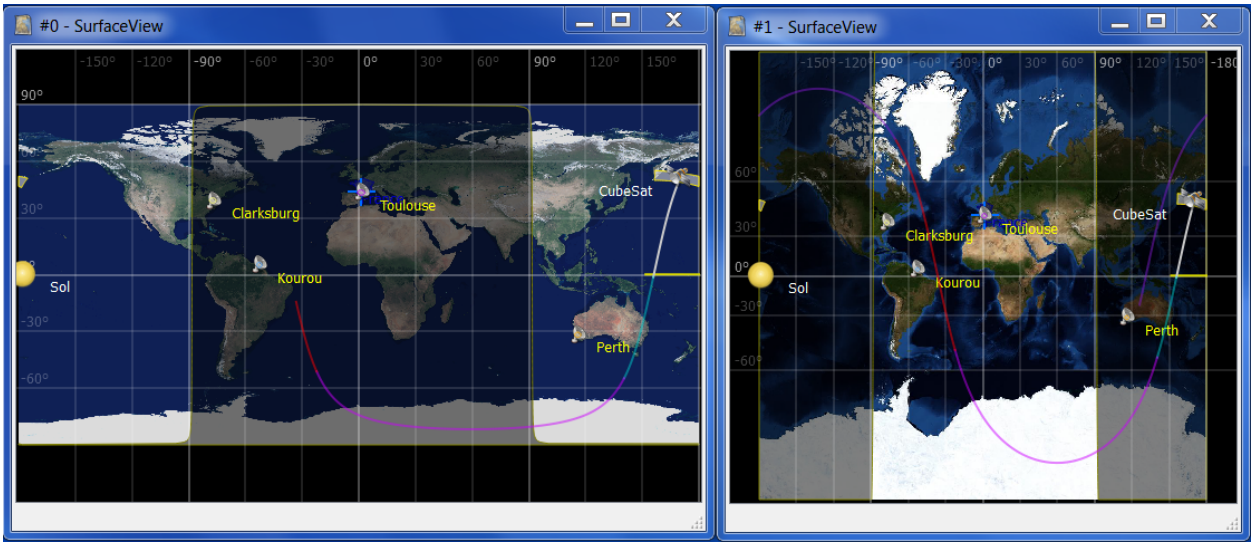


5.3.5 Map Projections

Currently SurfaceView supports the following projections:

- Equirectangular projection (*plate carree*)
- Mercator
- Polar

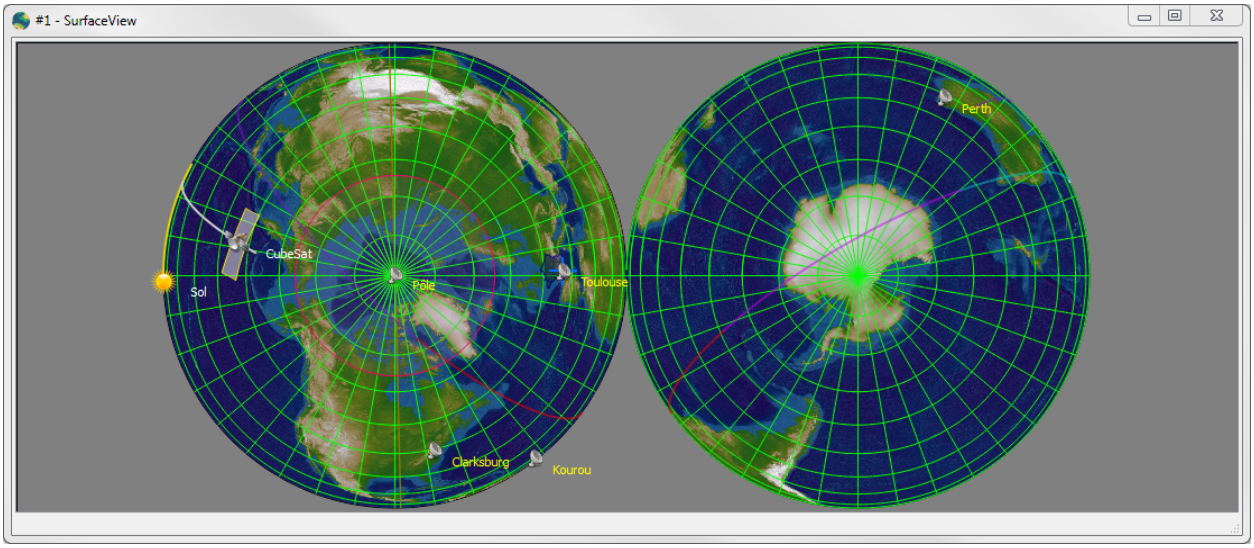
Each SurfaceView instance can be configured with a different projection using from the initial parameters of the application. Please refer to *Application parameters in VTS*.



5.3.6 Tile map

The tilemap mechanism allows you to load and display high definition maps as background images, depending on the zoom level. The tilemap is configured by a local URL pointing to a multi-level image hierarchy or a WMS server.

Please refer to *Tile Map textures* for more information about local or WMS tile maps.



When using a WMS tile map, in SurfaceView, the following parameters will be automatically added according the current zoom level.

- Latitude/longitude tile's border coordinates: `&BBOX=<west>,<south>,<east>,<north>`
- Tile size in pixels: `&WIDTH=<width>&HEIGHT=<height>`

5.3.7 Clusters

Clusters are used to display a large population of objects. Information about each object can be displayed in real-time.

User can configure it by editing the following parameters:

- ***Cluster visibility*:** String defining a partial or a total visibility of the cluster. Allowed strings are :
  - `all`: all objects are visible
  - `none`: all objects are hidden
  - `all except obj1, obj2, ..., objN`: all objects are visible except the objects specified in the list (comma-separated object names)
  - `none except obj1, obj2, ..., objN`: all objects are hidden except the objects specified in the list (comma-separated object names)

5.3.8 Interacting with SurfaceView

Context menu

The context menu accessible by right-clicking anywhere in the application window offers the entries:

These tools can also be accessed through the window toolbar. The toolbar is hidden by default and can be shown/hidden by pressing **F12**

- **F4** Activates the "normal" manipulation mode. This mode allows manipulating the view and centering on object via a left click.

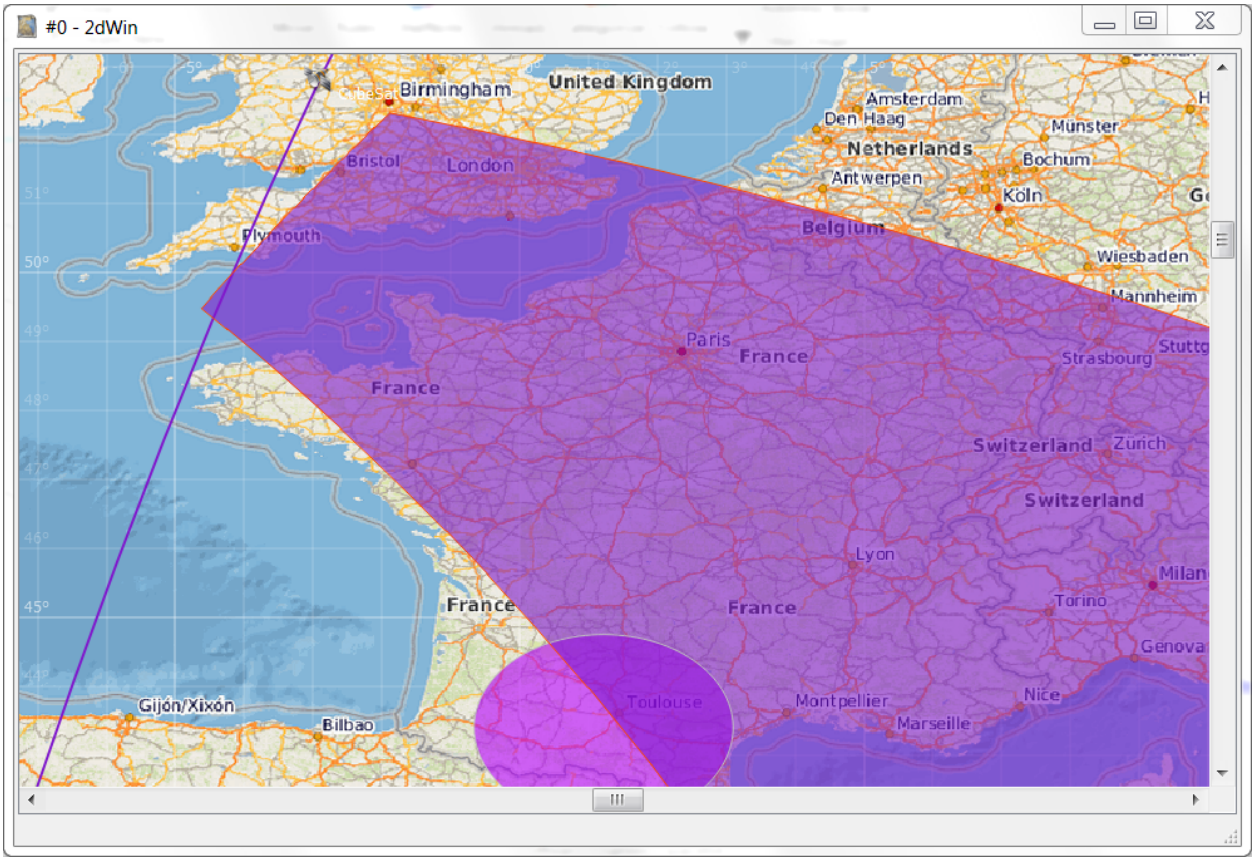


Fig. 5.61: Open Street Map in SurfaceView

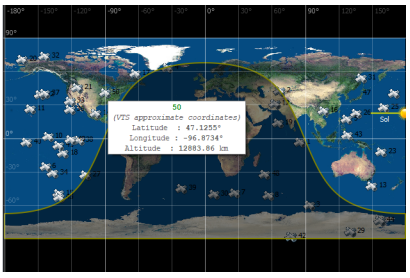
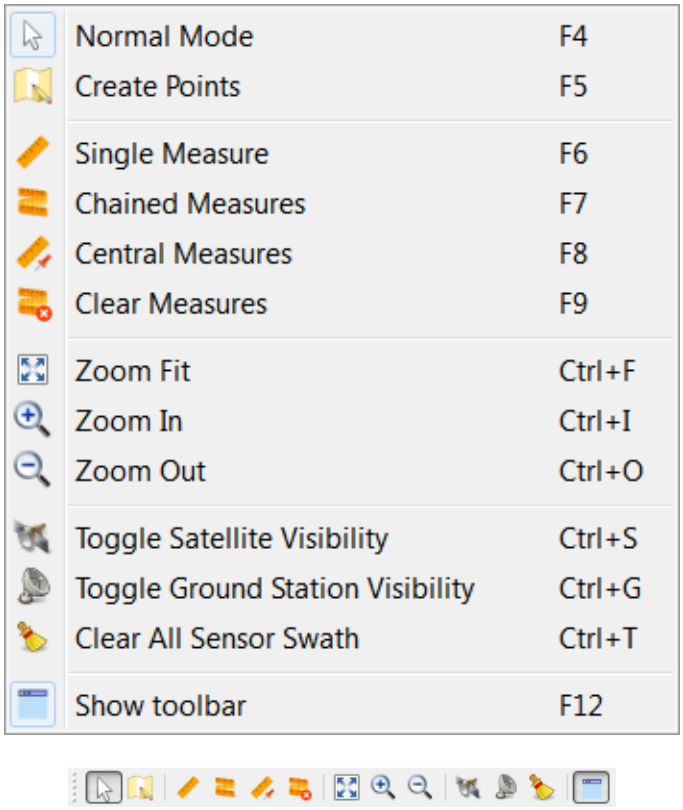


Fig. 5.62: Information about an object of the population



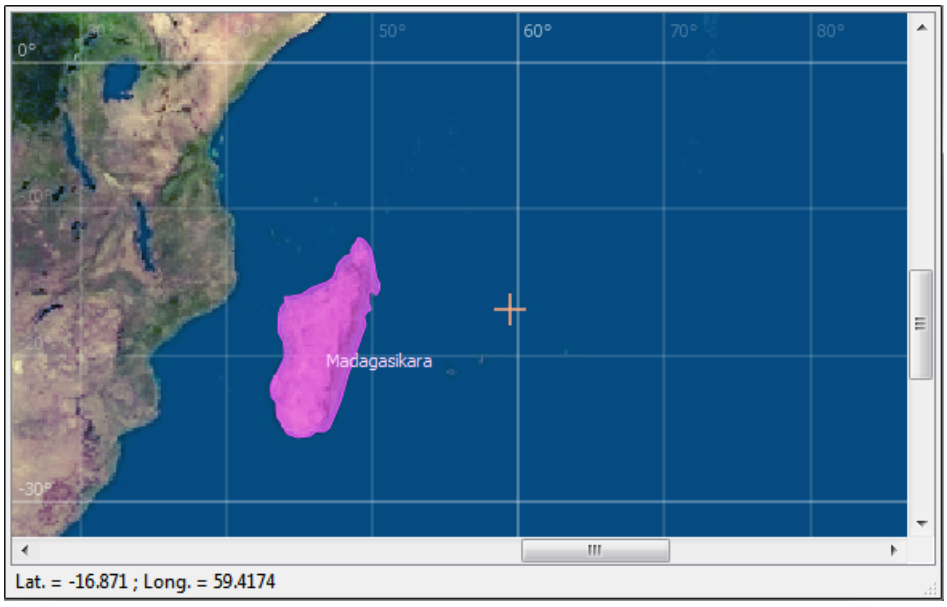
- **F5** Activates the “editing” manipulation mode. When this mode is enabled, shift+left clicking on the map stores the click geographic location in the system clipboard.
- **F6** Activates the “single measure” tool. This mode allows creating orthodromic measures from points to points.
- **F7** Activates the “chained measure” tool. This mode allows creating chained measures : the end point of a measure becomes the beginning point of the next one.
- **F8** Activates the “central measure” tool. This mode allows creating central measures : all measures are done from the same first “central” point.
- **F9** Clears all the measures done beforehand.
- **Ctrl+F** Adjusts the zoom level so that the whole planisphere fits in the window.
- **Ctrl+I** **Ctrl+O** Zooms in and out the planisphere. This can also be achieved using the mouse wheel. The zoom is then centered on the current mouse position in the window.
- **Ctrl+S** Shows/hides all satellite icons.
- **Ctrl+G** Shows/hides all ground station icons.
- **Ctrl+T** Clears the sensor swath overlay, if enabled in the project.
- **F12** Shows/hides the toolbar.

Cursor coordinates

The cursor coordinates can be displayed by pressing the `SHIFT` key. The coordinates expressed in latitude/longitude are displayed in the window status bar.

Use `SHIFT` + `left click` to copy the cursor coordinates into the clipboard. When the `Edit mode` is active, the points are kept after the release of `SHIFT`, so that more points can be accumulated. When the `Normal mode` is active, the points are cleaned after the release of `SHIFT`. Added new points will reset the previous points in the clipboard.

You can copy multiple coordinates by maintaining the `SHIFT` key pressed and left clicking on different locations. The distance between the selected points is also displayed in the status bar. The locations marks will remain visible until the SHIFT key is released. This is useful to create a POI/ROI as the coordinates can be directly pasted in the POI/ROI file.



Projection center

It is possible to shift the longitude coordinate of the map projection's center and thus to center the scene at a meridian other than the prime meridian (Greenwich).

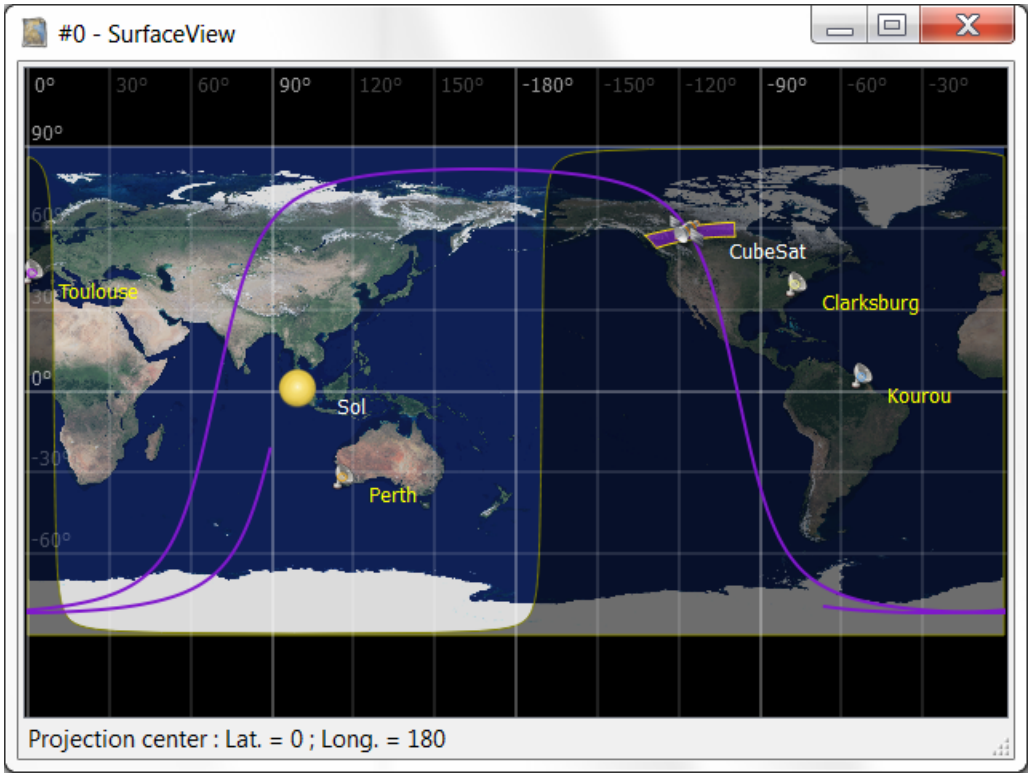
This is achieved by holding the `CTRL` key while dragging the background image. The coordinates of the projection center are displayed in the window status bar.

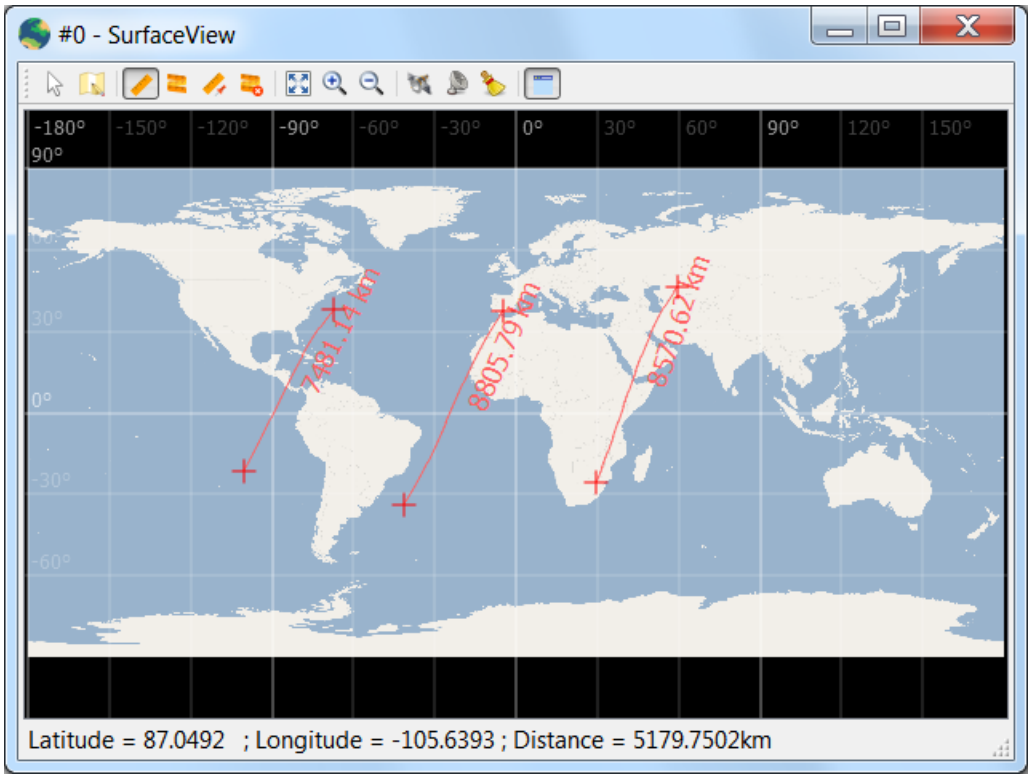
Orthodromic measures

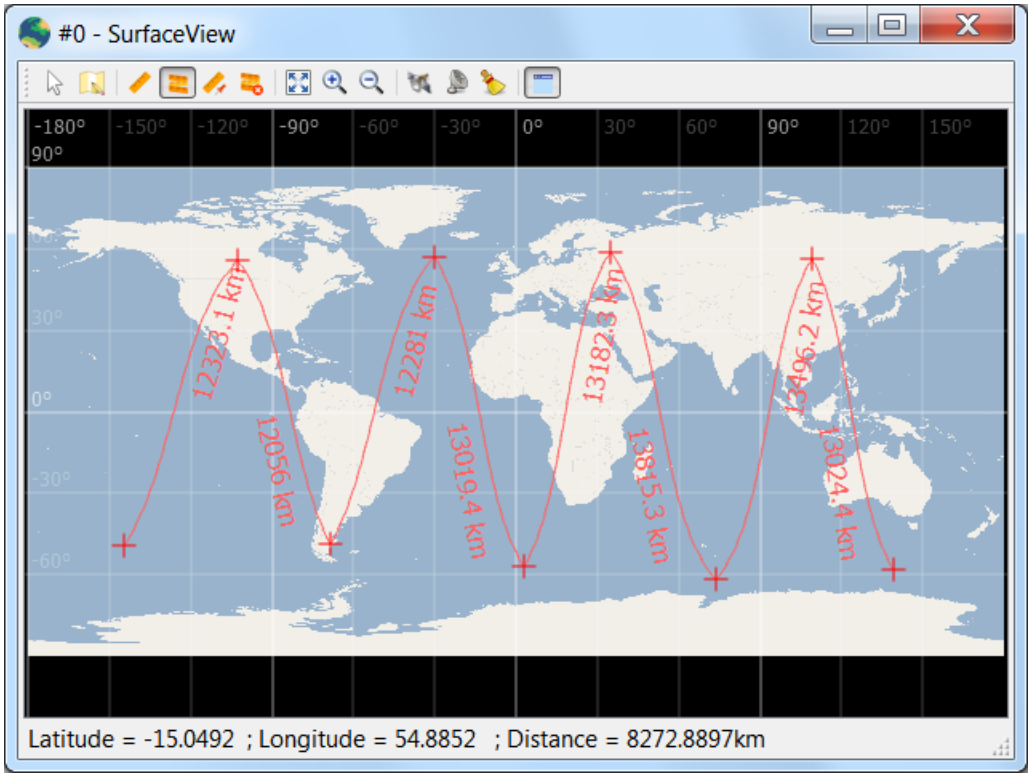
You can create measures in SurfaceView using one of the three availables tools, described below. The `LEFT` button of the mouse is used to create measures.

The three measuring tools proposed by SurfaceView are:

- **Single Measures** (F6) : allows creating orthodromic measures from points to points.
- **Chained Measures** (F7) : allows creating chained measures : the end point of a measure becomes the beginning point of the next measure.


#0 - SurfaceView
0°
30°
60°
90°
120°
150°
-180°
-150°
-120°
-90°
-60°
-30°
90°
0°
-30°
-60°
Toulouse
CubeSat
Clarksburg
Kourou
Sol
Perth
Projection center : Lat. = 0 ; Long. = 180


#0 - SurfaceView
-180°
-150°
-120°
-90°
-60°
-30°
0°
30°
60°
90°
120°
150°
90°
30°
0°
-30°
-60°
7481.14 km
8805.79 km
8570.62 km
Latitude = 87.0492 ; Longitude = -105.6393 ; Distance = 5179.7502km



- **Central Measures** (F8) : allows creating central measures : all measures a done from the same first "central" point.

Pressing `F9` clears all the measures.

Interacting with a satellites and other entities

Centering lock

Clicking on a satellite, a station or any celestial body will center and lock the view on that entity during the animation. Zoom actions are also centered on the entity instead of the mouse cursor. Clicking the map unlocks the view.

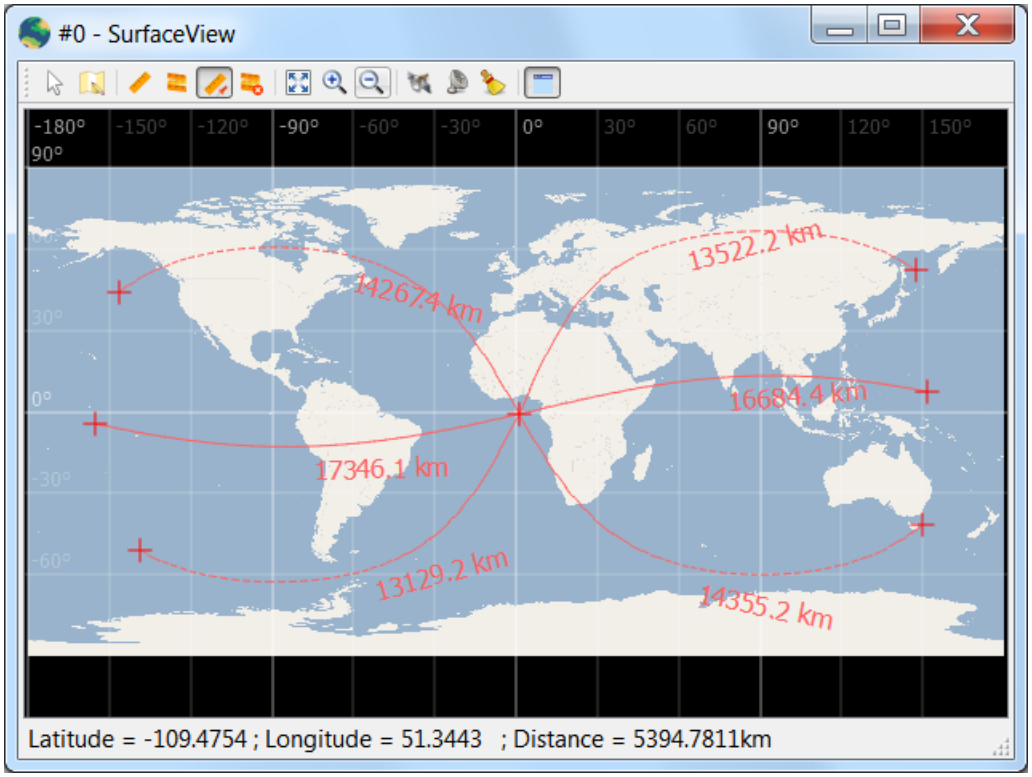
Geographical coordinates

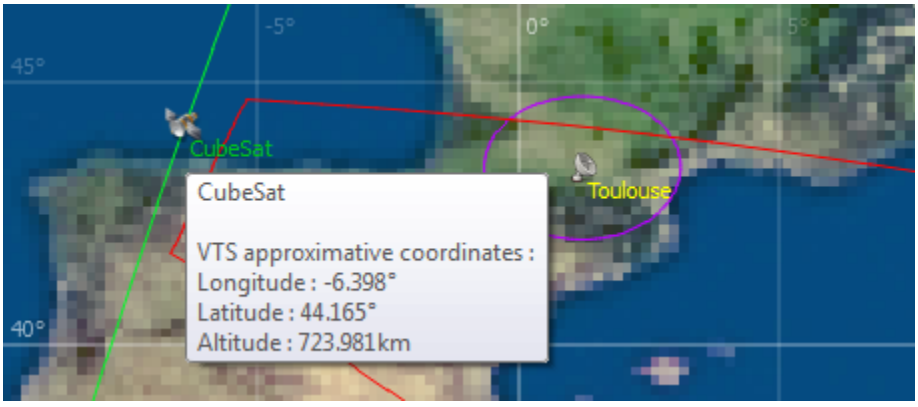
Hovering the mouse over a satellite, ground station or celestial body, will display a tooltip containing the geographical coordinates of the entity.

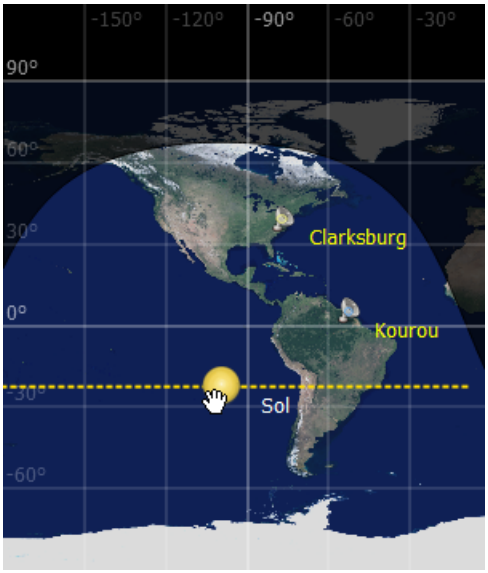
These coordinates are computed by VTS and are approximate, especially when computing the longitude near the poles.

Moving the time

The Sun, satellites and celestial bodies defined in the VTS project have a **ground track** associated which can be used to drag the entity along it in order to move forwards or backwards in time. The dragged entity automatically snaps to the nearest point of its ground track. The corresponding date is sent to the Broker and all applications are synchronized.


#0 - SurfaceView
-180° -150° -120° -90° -60° -30° 0° 30° 60° 90° 120° 150°
90°
0°
-30°
-60°
14267.4 km
13522.2 km
16684.4 km
17346.1 km
13129.2 km
14355.2 km
Latitude = -109.4754 ; Longitude = 51.3443 ; Distance = 5394.7811km


-5°
0°
5°
45°
40°
CubeSat
Toulouse
CubeSat
VTS approximative coordinates :
Longitude : -6.398°
Latitude : 44.165°
Altitude : 723.981km



It is possible to "jump" from orbit to orbit if the ground track is long enough (for example, greater than the period of the satellite's orbit). This results in a similar "jump" in visualization time.

Dragging satellites near track intersections can become very "unstable" (typically near the poles, or if the track time window is greater than the period of the satellite's orbit). The satellite may "jump" from orbit to orbit depending on the position and movement of the mouse. It is advised to avoid dragging a satellite in these areas if "jumping" from orbit to orbit is not desired.

Note that visualization time is automatically paused when dragging an entity.

Mission events

SurfaceView displays mission events attached to satellites and allows some interaction with them. The events are displayed along the satellite's ground track at the corresponding date of the event. Refer to the *Events attached to a satellite* section in the *VTS configuration utility user manual* chapter for more information.

The line segment linking the icon of an event to its actual position on the satellite ground track is the same color (only a bit lighter) as that of the satellite ground track.

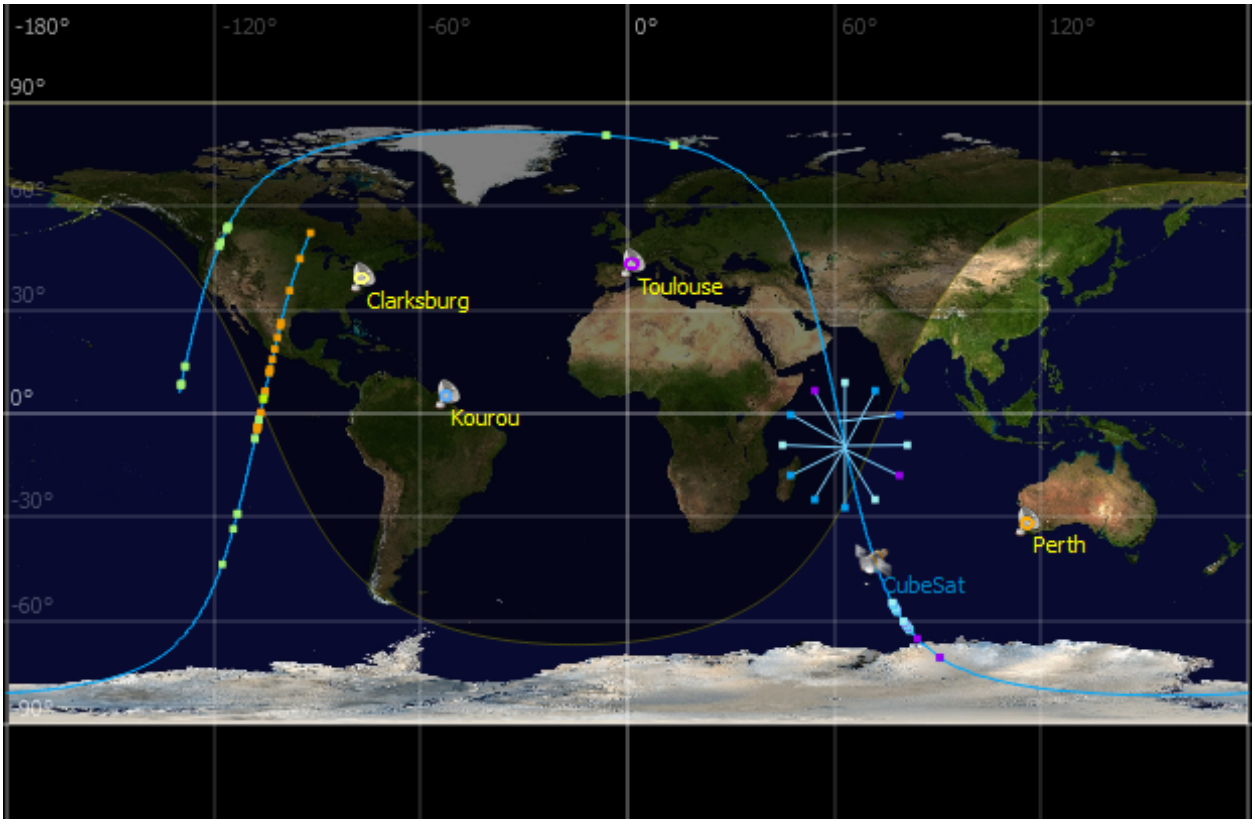
The name of all events are always displayed if the *All event text visibility* parameter is checked in the Broker.

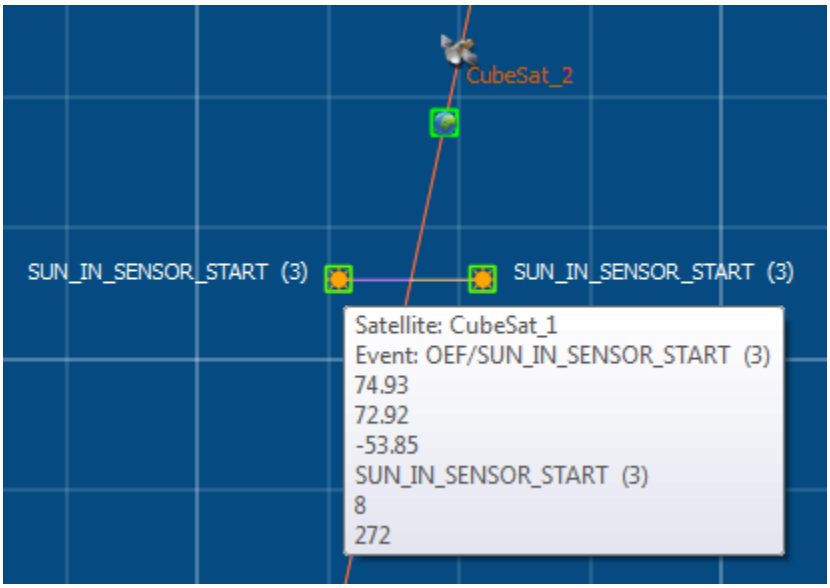
The visibility of all event types can be controlled from the *Events* tab of the Broker. Refer to the *Events* tab section of the *Broker user manual* chapter for more information.

The appearance of mission events can be defined in the *Event Type Editor* tab of the VTS configuration utility. Refer to the *Configuring event types* section in the *VTS configuration utility user manual* chapter for more information.

Interacting with events

- **Hovering: Hovering above a single event will display a tooltip containing the name of the satellite the event is attached to,**
  - Events at the exact same date – or at dates close enough that they appear to be at the same date at the current zoom level – will animate and split when the mouse is hovered above them.
- **Double-click**: Double-clicking an event will set the current visualization time to the date of the event. The satellite then appears located at the position of the event. The visualization is also paused.


-180°
-120°
-60°
0°
60°
120°
90°
30°
0°
-30°
-60°
Clarksburg
Toulouse
Kourou
Perth
CubeSat


CubeSat_2
SUN_IN_SENSOR_START (3)
SUN_IN_SENSOR_START (3)
Satellite: CubeSat_1
Event: OEF/SUN_IN_SENSOR_START (3)
74.93
72.92
-53.85
SUN_IN_SENSOR_START (3)
8
272

5.3.9 Technical notes

Rotation model

SurfaceView uses the VTS ephemeris catalog to display an accurate positioning and Sun terminator (see the [Central bodies in VTS] section for more information about ephemeris origin). For custom positioning or unsupported bodies, suitable ephemerides files will need to be provided to override the catalog ones. Refer to the *Position and orientation of a body* section of the *VTS configuration utility user manual* chapter for more information.

Central body texture

The current architecture of the SurfaceView application enforces a few constraints:

- The *built-in* texture for Earth is embedded in SurfaceView. It is based on the Blue Marble photos from NASA (image link , documentation link).
- For other solar system bodies or celestial bodies available in Celestia, the *built-in* texture will be taken from Celestia, using the equidistant cylindrical (plate carree) projection.
- For bodies not available in Celestia, a black texture will appear in *built-in* mode. A custom texture should then be defined. Refer to the Texture of a body section of the *VTS configuration utility user manual* chapter for more information.

5.4 SkyView user manual

The SkyView client application displays a rectangular start chart centered on a specific satellite.

5.4.1 Introduction

SkyView displays a 2D star chart based on the equirectangular projection of the celestial sphere and centered on a specific satellite. Project entities (such as satellites and celestial bodies) are projected into the celestial sphere according to their position relatie to the satellite.

5.4.2 Configuration in VTS

The entities and properties defined in the VTS configuration utility will be displayed in SkyView.

Application parameters in VTS

When adding SkyView as a VTS client application, some parameters can be set by clicking on the SkyView entry in the VTS project tree.

Since SkyView shares some parameters with SurfaceView, only SkyView specific parameters will be described here. Please refer to *SurfaceView user manual* for more detail on the application parameters in common. :

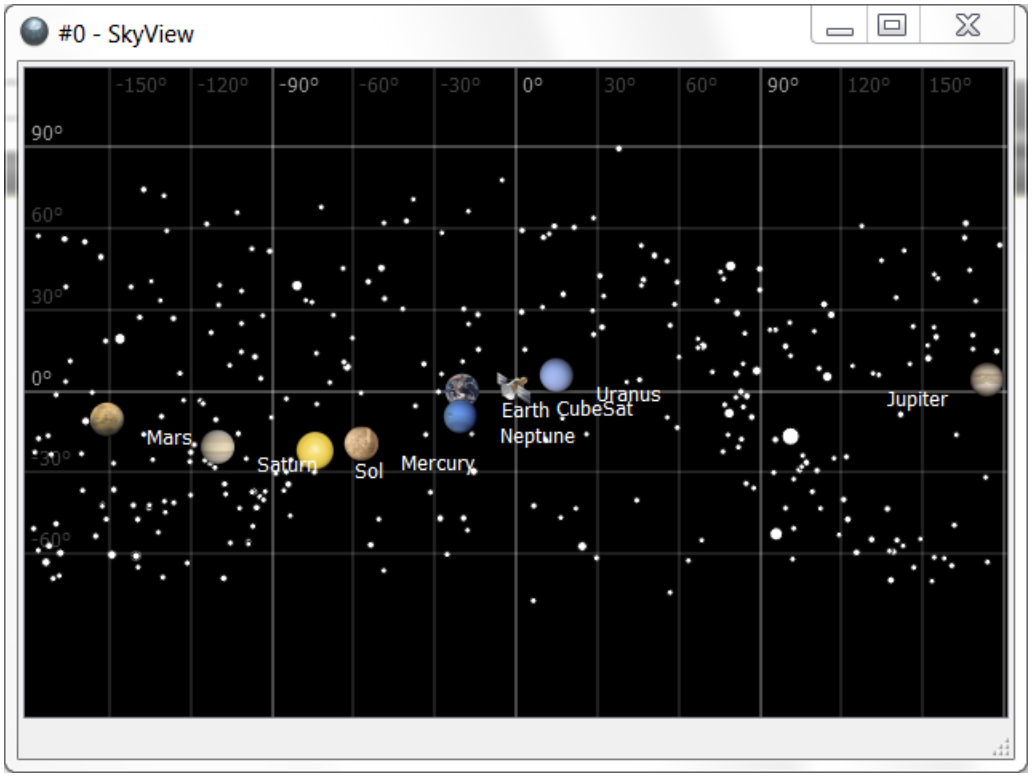


Fig. 5.63: SkyView client application

Initial States

Property name	Value
Application parameters	
Scene reference	Default
Map Projection	Equirectangular
Min visible star count	300
Loaded star count from catalog	10000
Target framerate	10
Sensor geometry section count	128
Sensor coverage significant threshold (%)	5
Sensor coverage merging threshold (%)	90

Fig. 5.64: SkyView application parameters.

Parameter	Description
Scene Reference	This combo box allows you to select a satellite from the list of satellites defined in the VTS project tree. The selected satellite will become the scene reference in the SkyView instance. By default, the first satellite from the VTS project tree is selected as a scene reference.
Min visible star count	Minimum number of stars displayed at any time or zoom level.
Loaded star count from catalog	Number of stars to load from the specified star catalog.

5.4.3 Satellite frame axes

The intersection of a satellite frame axes with the visualization plan can be enabled through the satellites proprerties. The following frame axes can be represented:

- EME2000 inertial frame axes,
- QSW frame axes,
- Satellite frame axes.

Axes are represented by crosses (red, green and blue) followed by the axe name: `<SatelliteName>_<AxisName>`.

- EME2000: Xeme2000 (red), Yeme2000 (green), Zeme2000 (blue),
- QSW: Q (red), S (green), W (blue),
- Satellite: Xsat (red), Ysat (green), Zsat (blue).

5.5 SensorView user manual

The SensorView client application displays the celestial sphere as seen from a specific satellite sensor.

5.5.1 Introduction

The SensorView application displays an equirectangular projection of the celestial sphere as seen from a specific satellite sensor (centered on it). Project entities (such as satellites and celestial bodies) are projected into the celestial sphere according to their position relative to the sensor.

5.5.2 Configuration in VTS

The entities and properties defined in the VTS configuration utility will be displayed in SensorView.

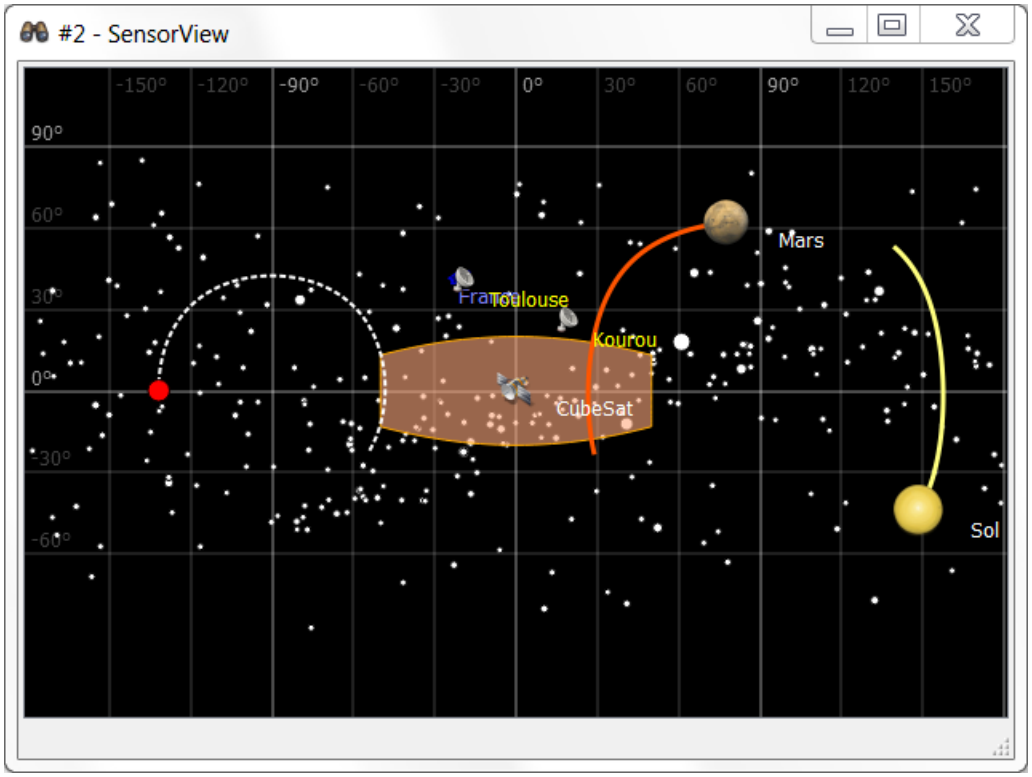


Fig. 5.65: SensorView client application

Application parameters in VTS

When adding SensorView as a VTS client application, some parameters can be set by clicking on the SensorView entry in the VTS project tree.

Since SensorView shares its parameters with SurfaceView and SkyView, please refer to *SurfaceView user manual* and *SkyView user manual* for more detail on the application parameters in common.

5.5.3 Satellite frame axes

The intersection of a satellite frame axes with the visualization plan can be enabled through the satellites proprerties. The following frame axes can be represented:

- EME2000 inertial frame axes,
- QSW frame axes,
- Satellite frame axes.

Axes are represented by crosses (red, green and blue) followed by the axe name: `<SatelliteName>_<AxisName>`.

- **EME2000**: Xeme2000 (red), Yeme2000 (green), Zeme2000 (blue),
- **QSW**: Q (red), S (green), W (blue),
- **Satellite**: Xsat (red), Ysat (green), Zsat (blue).

5.6 NadirView user manual

The NadirView client application displays an equirectangular projection of the celestial sphere as seen from a specific satellite (centered on the Nadir, X axis aligned with velocity).

5.6.1 Introduction

The NadirView client application displays an equirectangular projection of the celestial sphere as seen from a specific satellite (centered on the Nadir, X axis aligned with velocity). Project entities (such as satellites and celestial bodies) are projected into the celestial sphere according to their position relative to this frame.

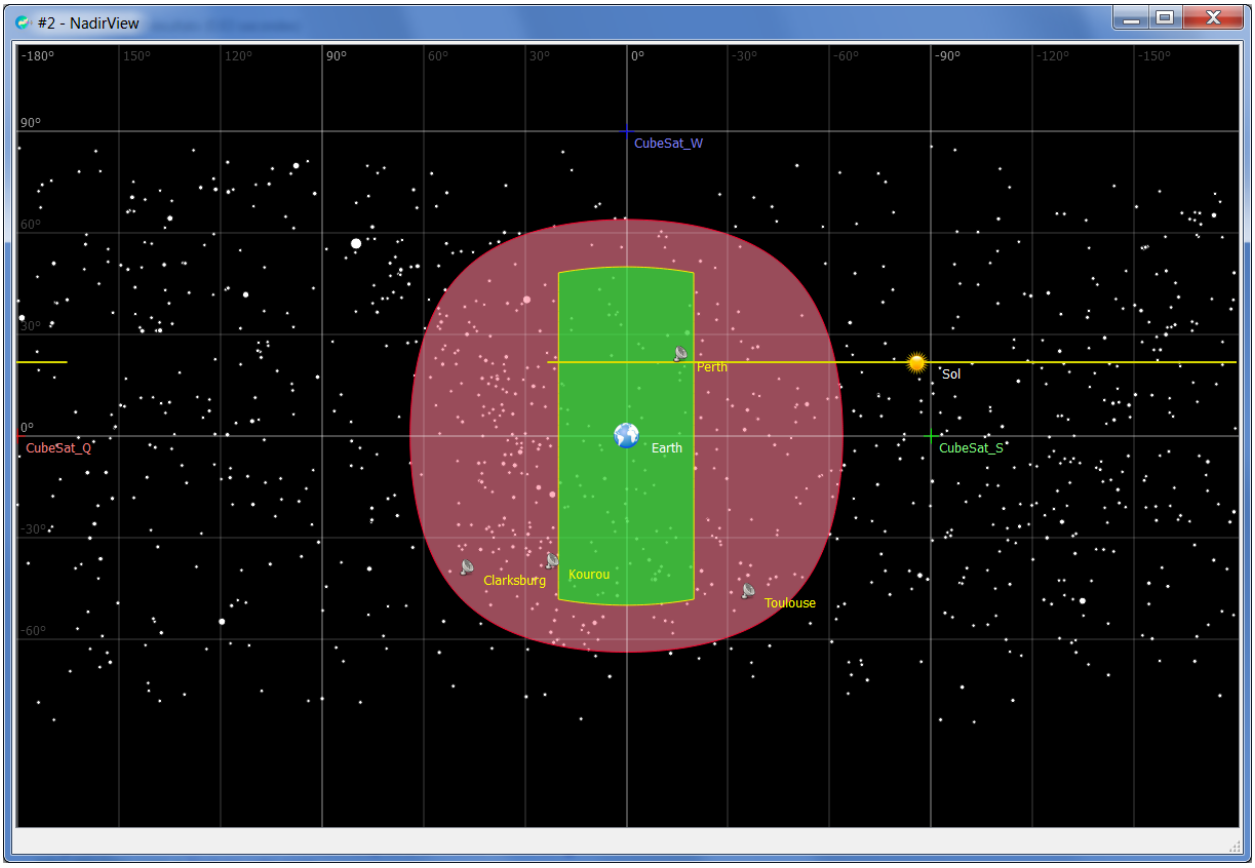


Fig. 5.66: NadirView application

5.6.2 Configuration in VTS

The entities and properties defined in the VTS configuration utility will be displayed in NadirView.

Application parameters in VTS

When adding NadirView as a VTS client application, some parameters can be set by clicking on the NadirView entry in the VTS project tree.

Since NadirView shares its parameters with SurfaceView and SkyView, please refer to *SurfaceView user manual* and *SkyView user manual* for more detail on the application parameters in common.

5.6.3 Satellite frame axes

The intersection of a satellite frame axes with the visualization plan can be enabled through the satellites proprerties. The following frame axes can be represented:

- EME2000 inertial frame axes,
- QSW frame axes,
- Satellite frame axes.

Axes are represented by crosses (red, green and blue) followed by the axe name: `<SatelliteName>_<AxisName>`.

- **EME2000**: Xeme2000 (red), Yeme2000 (green), Zeme2000 (blue),
- **QSW**: Q (red), S (green), W (blue),
- **Satellite**: Xsat (red), Ysat (green), Zsat (blue).

5.7 ZenithView user manual

The ZenithView client application displays half of the celestial sphere as seen from a specific ground station.

5.7.1 Introduction

The ZenithView application displays a Lambert azimuthal equal-area projection of the celestial sphere as seen from a specific ground station (centered on it). Project entities (such as satellites and celestial bodies) are projected into the celestial sphere according to their position relative to the station.

5.7.2 Configuration in VTS

The entities and properties defined in the VTS configuration utility will be displayed in ZenithView.

Application parameters in VTS

When adding ZenithView as a VTS client application, some parameters can be set by clicking on the ZenithView entry in the VTS project tree.

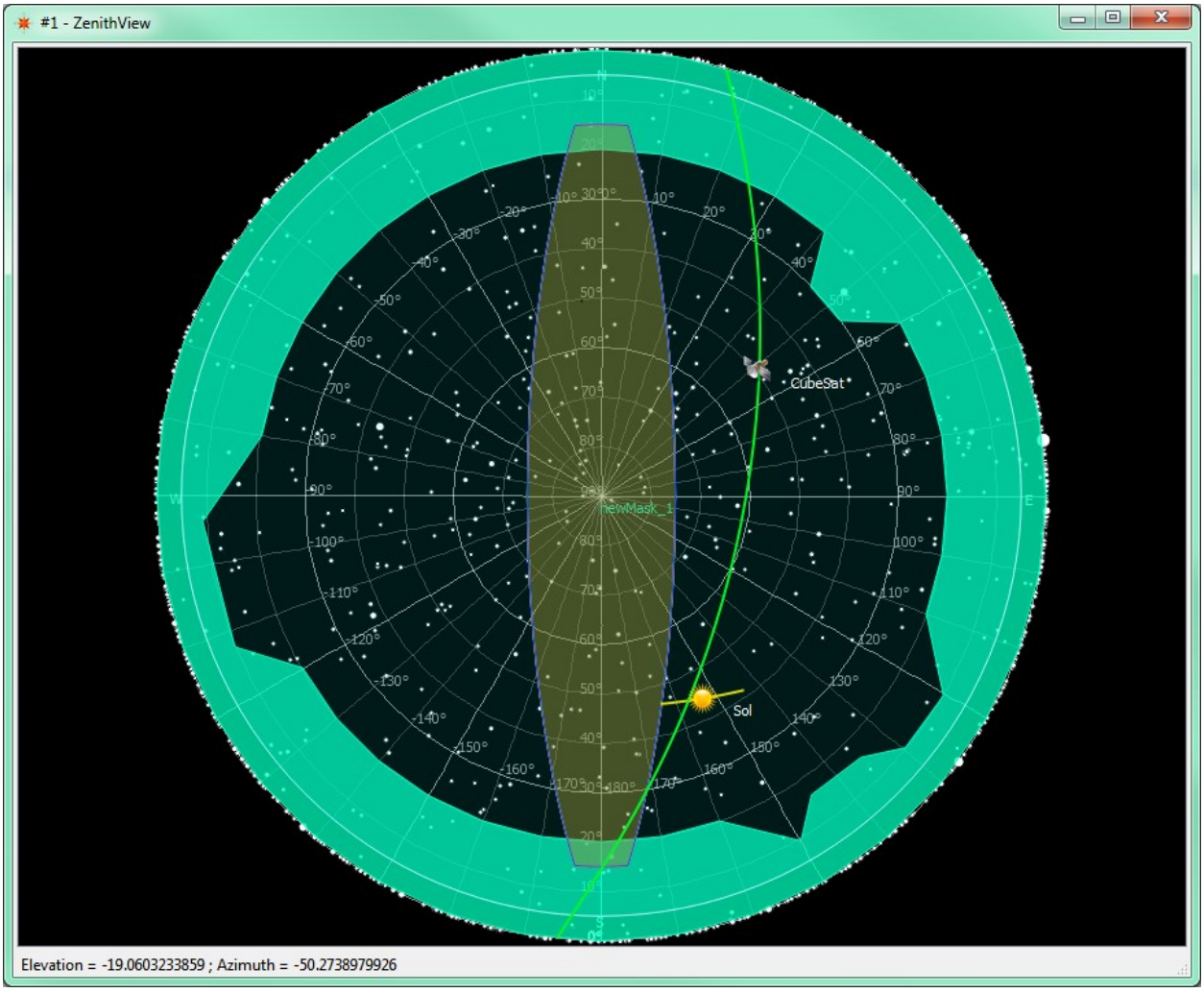


Fig. 5.67: ZenithView client application

Parameter	Description
Scene Reference	This combo box allows you to select a ground station from the list of ground stations defined in the VTS project tree. The selected ground station will become the scene reference in the ZenithView instance. By default, the first ground station from the VTS project tree is selected as a scene reference.
Project on horizon when outside the view	If true, entities whose projection are outside the limits of the current view (i.e. a satellite not visible from the ground station) will be displayed on the horizon of the view.
Projection field of view	The field of view of the projection in degrees.

Since ZenithView shares its parameters with SkyView, please refer to *SkyView user manual* for more detail on the application parameters in common.

5.8 Celestia user manual

The main 3D client application in VTS is Celestia. Celestia is open-source software, developed, amongst others, by Chris Laurel.

VTS relies on the latest source version of Celestia (http://www.shatters.net/celestia). It is interfaced with VTS through LUA scripts.

5.8.1 Integration with VTS

Celestia is fully integrated with VTS. All of Celestia's features related to the visualization of satellites within the perimeter of VTS are available directly from the Broker. These features are described in detail in the *Messages received by Celestia* section of the *Synchronization protocol for VTS clients* chapter, and in the *3D Cameras* tab section of the *Broker user manual* chapter.

5.8.2 Navigating in Celestia

The main navigation controls in Celestia are:

- Left click + mouse move: camera pointing
- Right click + mouse move: rotation around the selected object
- Mouse wheel: zoom in/out on the selected object
- Shift + left click + mouse move: change field of view
- Left/right arrow keys: camera roll

All controls are described in Celestia's *Help* menu (the menu bar must be enabled in the Broker).

If Celestia is used inside a virtual machine (VirtualBox, VMware, etc.), one should disable mouse integration in order to have correct mouse move handling.

5.8.3 Specific application parameters in VTS

When adding Celestia as a VTS client application, some parameters can be set by clicking on the Celestia entry in the VTS project tree:

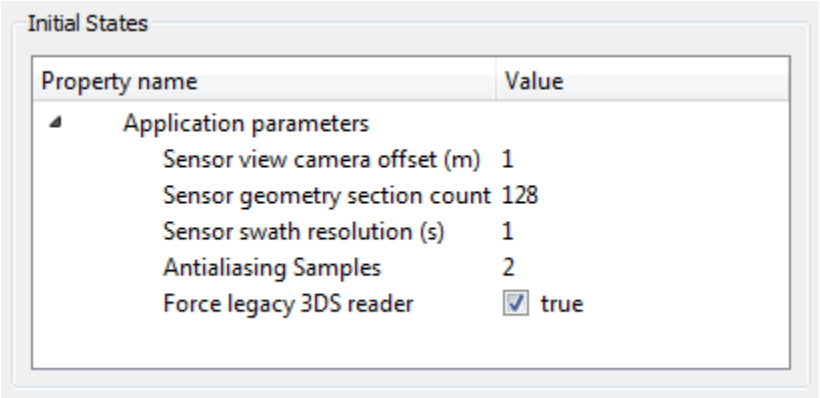


Fig. 5.68: Celestia application parameters

- **Sensor view camera offset**: The sensor camera view, accessible through 3D Camera tab in the Broker, sets the camera position at the sensor position plus an offset in meters to avoid some artifacts or obstructions of sight. This offset moves the camera along the Z axis.
- **Sensor geometry section count**: Number of points making up the outline polygon of the aiming sensor surface. Performance and display accuracy depend on this parameter.

One the one hand, performance may be affected by a too high setting and a long residual trace. On the other hand, accuracy is better with a high setting when the satellite attitude has a large angle with the nadir.

If you want to focus on sensor swath accuracy: increase the SensorGeometrySectionCount parameter and favor a short residual trace. If you want to view long missions coverage: lower the SensorGeometrySectionCount parameter with a long residual trace.

- **Sensor swath resolution**: Change the interval (in seconds) between two instantaneous sensor aiming surfaces.

Use a small value for an agile satellite, but affects the performance. Use a high value for a long coverage mission with good performance.

- **Antialiasing samples**: Set the level of multisample antialiasing. Not all 3D graphics hardware support antialiasing, through most newer graphics chipsets do. Larger values will result in smoother edges with a cost in rendering speed. 4 is a sensible setting for recent, higher-end graphics hardware; 2 is probably better for mid-range graphics hardware. 0 or 1 values disables antialiasing.
- **Force legacy 3DS reader**: When activated, the Celestia legacy 3D model reader is used. If a 3DS file can't be read, try another reader by disabling this option.

5.8.4 Specific properties in VTS

Each entity defined in the VTS project can have specific properties in the Celestia application. They are available in the Scenario Editor panel for the Celestia application :

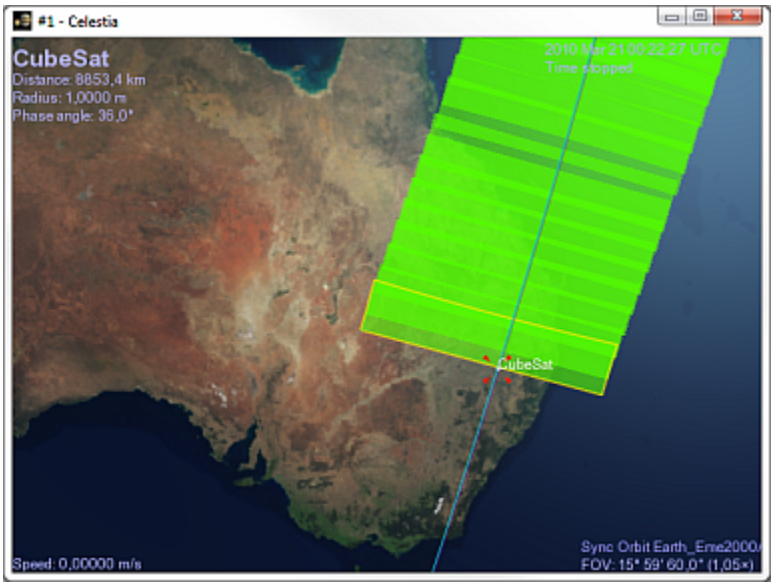


Fig. 5.69: The default setting creates a lot of overlapping traces

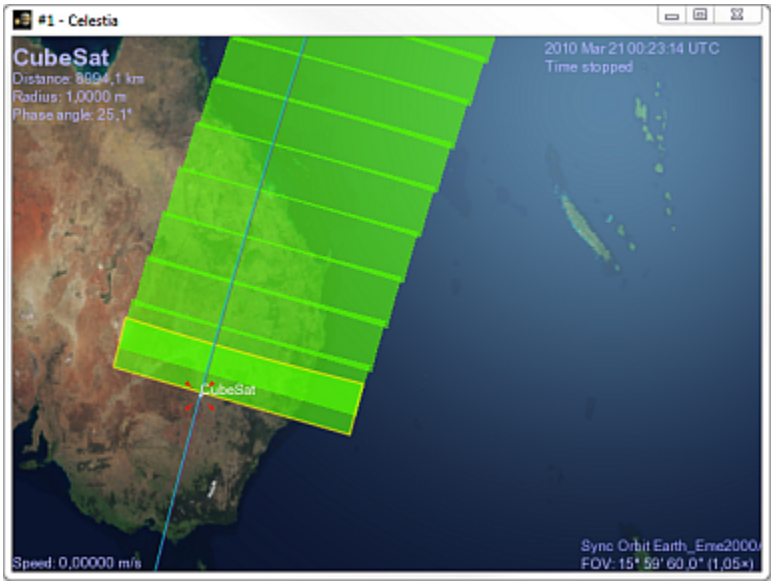


Fig. 5.70: A matching value between sensor width and satellite speed gives a better appearance

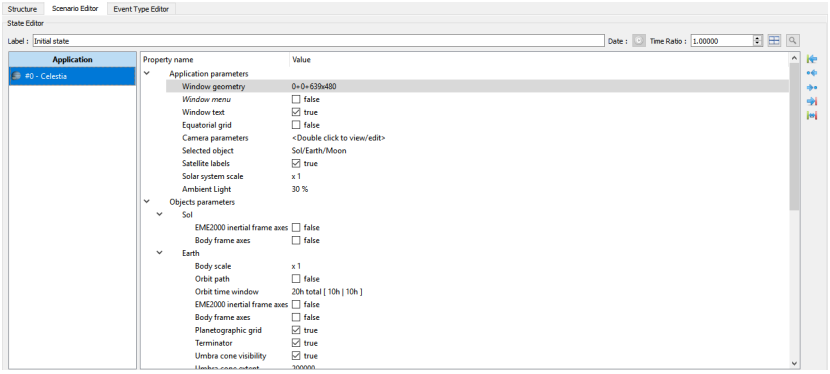


Fig. 5.71: Configuration of entity properties for Celestia

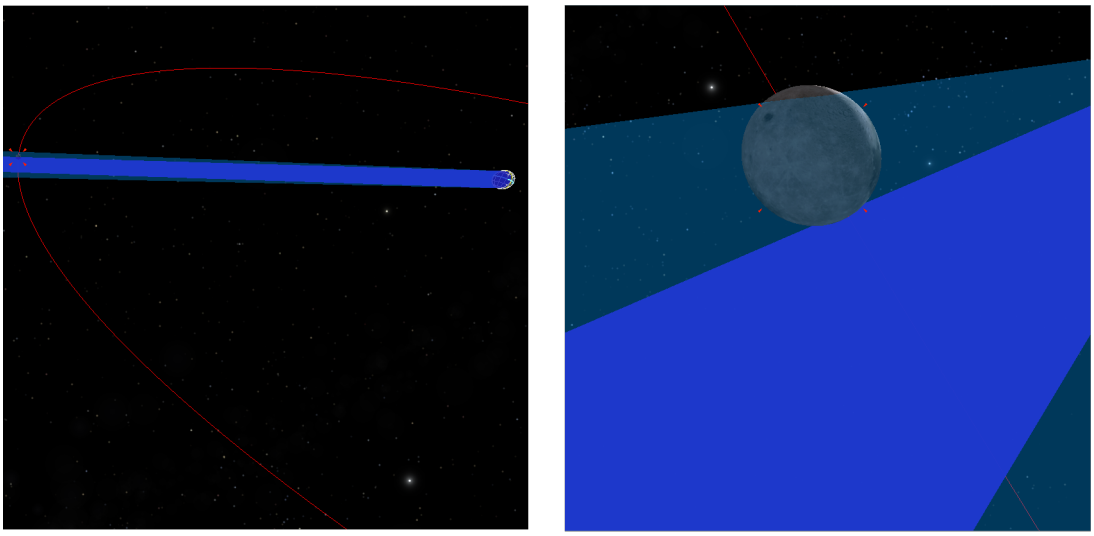


Fig. 5.72: Earth's umbra and penumbra during a lunar eclipse

Umbra and Penumbra cone

Available only for body entities, their umbra and penumbra cone can be displayed in celestia. For each cone, can be configured:

- **{Umbra|Penumbra} cone visibility:** Either the cone is displayed or not (hidden by default)
- **{Umbra|Penumbra} cone color:** The cone color
- **{Umbra|Penumbra} cone extent:** The cone extent (in km)from the center of the body (400000 km by default). The cone will be contained in the sphere center on the body center and of radius equal to the extent.

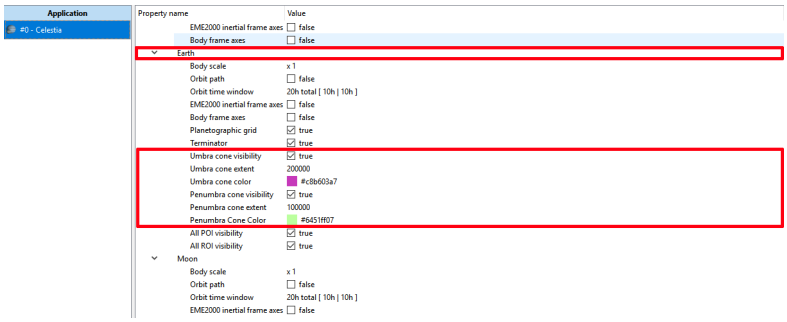


Fig. 5.73: Configuration of the umbra and penumbra cone for the Earth entity

---

Note: The solar or lunar eclipse are not perfectlty matching with the informations given by the IMCCE whether CNES or Celestia ephemeris are used. E.g. for the total the lunar eclipse on the 26/05/2021:

- IMCCE forecast are available at : https://promenade.imcce.fr/fr/images/ecl/LE2021May26T.pdf
- The U1 event happens about 2 min 43 s later with CNES ephemeris
- The U1 event happens about 1 min 45 s later with Celestia ephemerds

---

Clusters

In Celestia, the population of a cluster is represented by an icon.

It can be configured in terms of appearance and visibility.

- **Cluster visibility:** String describing the visibility. Accepted values are:
  - all : all objects are displayed
  - none : no object is displayed
  - all except obj1, obj2n, . . . , objN : all objects are visible except those designated by name in the comma separated list
  - none except obj1, obj2n, . . . , objN : all objects are hidden except those designated by name in the comma separated list

Celestial bodies texture layers

Celestia handles all texture layers except web tile map layers. As layer transparency is not supported, the cloud map is used to store the first fixed texture layer which handles transparency. In order to use it, you might make the first layer invisible.

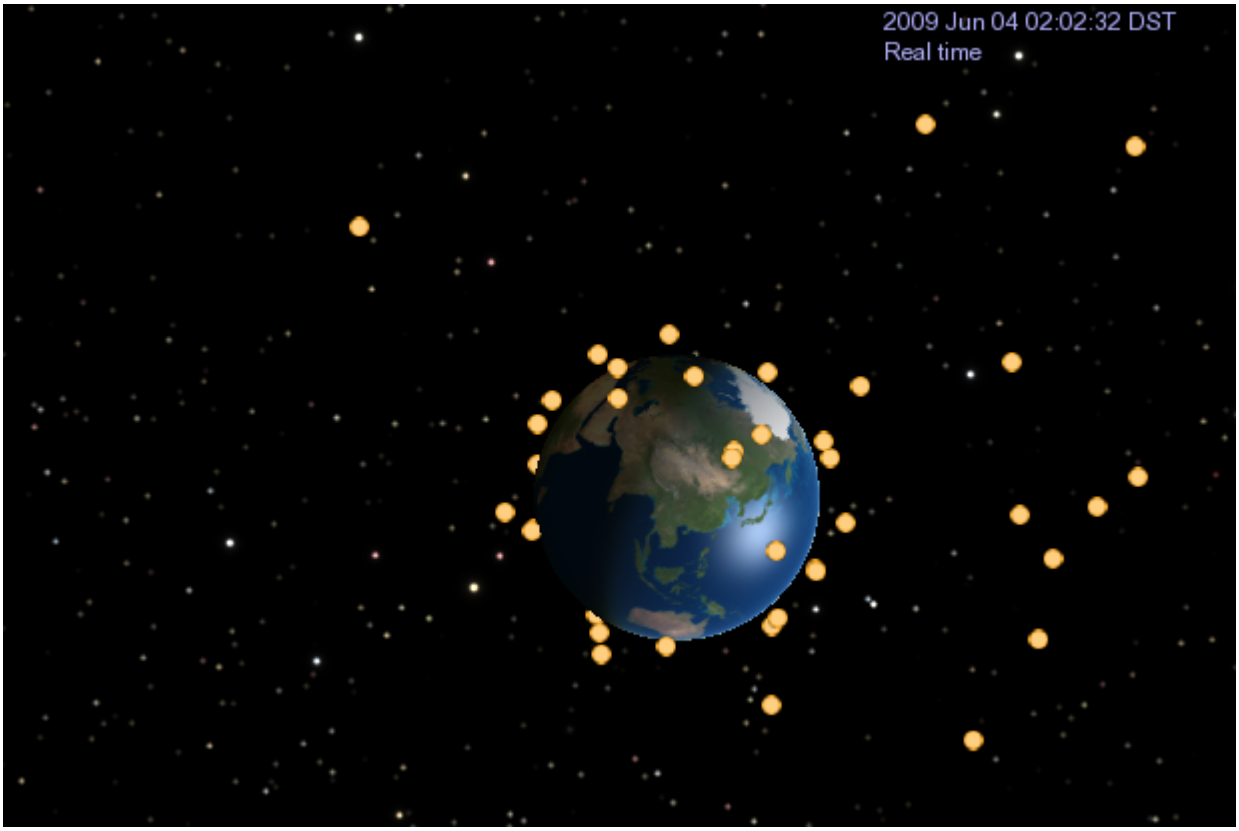


Fig. 5.74: A cluster in celestia

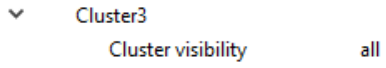


Fig. 5.75: Cluster configuration

- **Top layer overlay:** The top layer texture used as “cloud map” is displayed or not (hidden by default)

5.9 InfoBox user manual

The InfoBox plugin allows to display the simulation date and time and can also display data loaded from a CIC file.

5.9.1 System requirements

The InfoBox application displays a half-transparent overlayed window so the display can be placed over other applications. Most systems are compatible with transparent windows, as long as :

- the window manager supports Compositing,
- an advanced drawing API is available (XRender or OpenGL),
- advanced “desktop effects” are enabled.

Under RedHat 6.5, desktop effects can be enabled through the System Configuration Panel (see General > Appearance > Desktop > Desktop Effects > Enable desktop Effects). XRender or OpenGL can be chosen within the Advanced Options tab.

5.9.2 Configuration in the VTS configuration utility

The InfoBox application can be included in a VTS project from the Applications item (right click, Add Application, Infobox).

Fig. 5.76: InfoBox application in the VTS project tree

InfoBox parameters in VTS

When adding InfoBox as a VTS client application, the following parameters can be set by clicking on the InfoBox entry in the VTS project tree:

- **Time format**: Determines the format of the time. By default, the value is “hh:mm::ss”.
- **Date format**: Determines the format of the date. By default, the value is “yyyy-MM-dd”.
- **Time standard**: Defines the time standard used for the time. By default, the value is “UTC” but the user can choose “TAI” to add leap seconds.
- **Mode format**: Determines the format of the string for the time standard and the potential shifts. By default, the value is “${mode}${hour} (${second}s)” where ${mode} represents the time standard, ${hour} and ${second} represents a shift in hour and second. It means that if the application was set up with the UTC time standard with +4 hours shift and -12 seconds shift, it will show “UTC+4 (-12s)”.
- **Hour shift**: Shifts the displayed time by X hours according to the time standard. By default, the value is 0.
- **Second shift**: Shifts the displayed time by X seconds according to the time standard. By default, the value is 0.

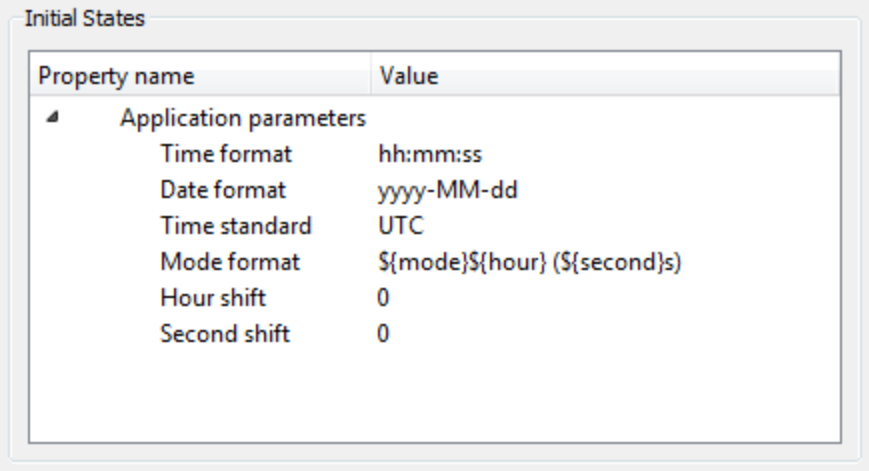


Fig. 5.77: InfoBox application parameters

5.9.3 Using InfoBox

Structure of the InfoBox display

Once the visualization is launched, the time and date of the simulation are displayed. The InfoBox display can be moved or closed with the help of the icons placed at the top-left of the display.

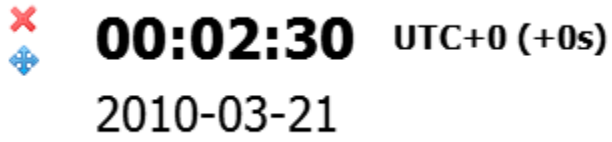


Fig. 5.78: InfoBox application

The background is transparent so the display can be placed over other applications such as SurfaceView or Celestia.

Specific properties of the InfoBox application

The following InfoBox properties can be modified during the simulation:

- **Window geometry**: The size of the window.
- **Base color**: Determines the base color of the display elements.
- **Dynamic color file**: Allows to dynamically change the color of the display elements by a CIC color file.
- **Data file**: Allows to display additional information from a CIC data file (see below).
- **Display # previous data lines**: Determines how many previous data lines should be displayed.
- **Display # following data lines**: Determines how many following data lines should be displayed.

And for each display elements, there are these properties:

- **visibility**: Determine the visibility of the corresponding element in the display.
- **dynamic color**: If set, the corresponding element will take its color from the CIC color file.

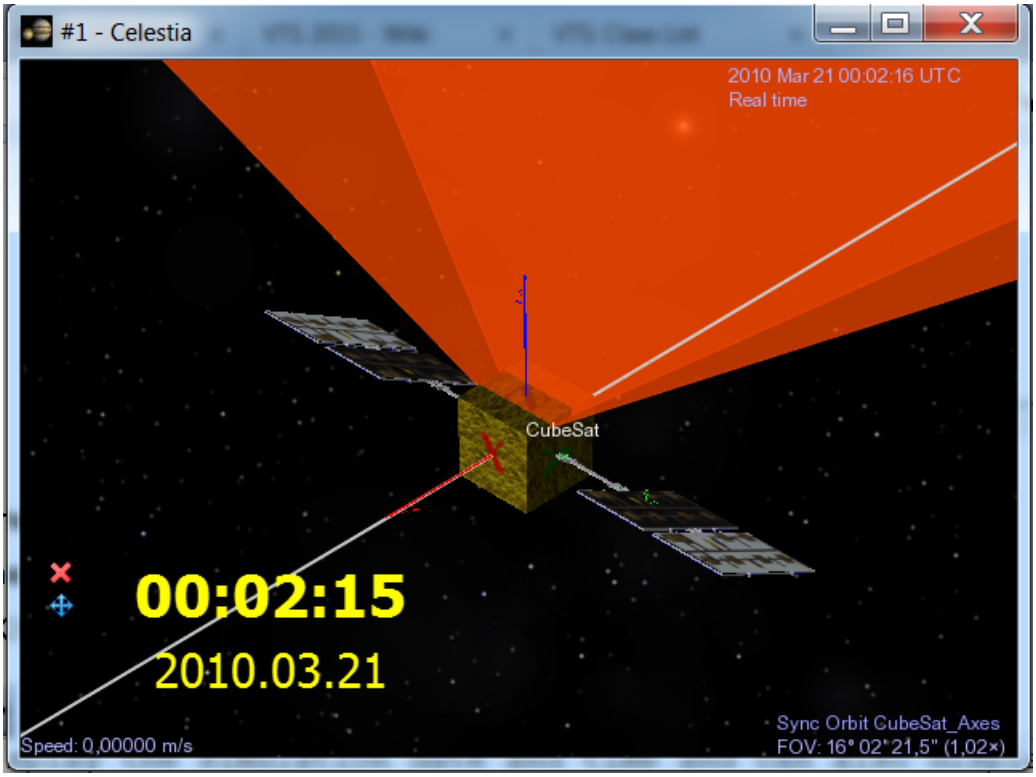


Fig. 5.79: InfoBox over the Celestia application

- **font**: Determines the font of the specified element.

Displaying a CIC data file

When a CIC data file is specified, the data is displayed according to the simulation time. In the following image, 4 lines are displayed and the current line is emphasized in a bold font.

When the special value `EVENTS <satellite full name>` (e.g. `EVENTS Sol/Earth/CubeSat`) is used for the `Data file` property, the events attached to the specified satellite are displayed instead of a CIC data file.

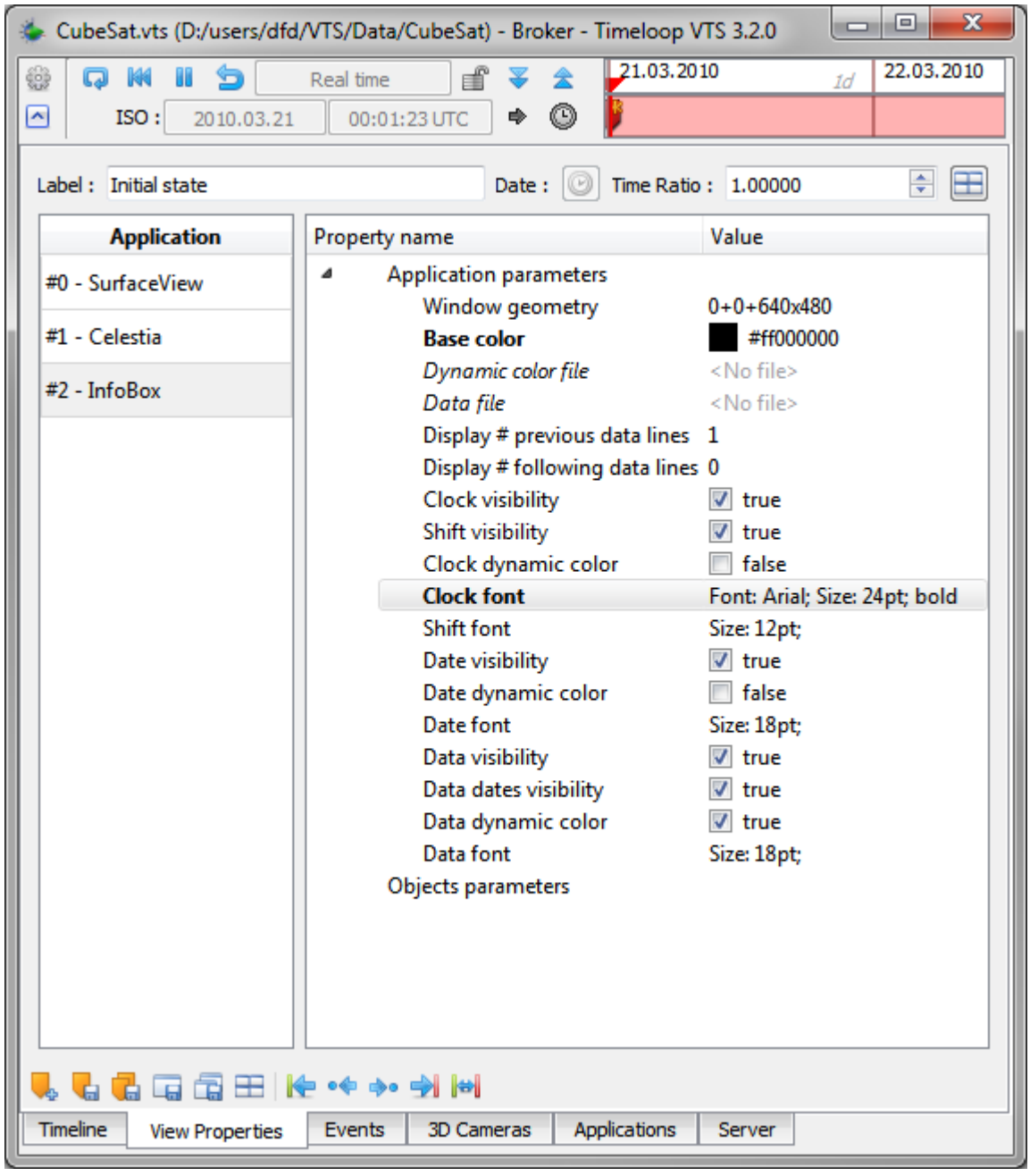


Fig. 5.80: Specific InfoBox properties

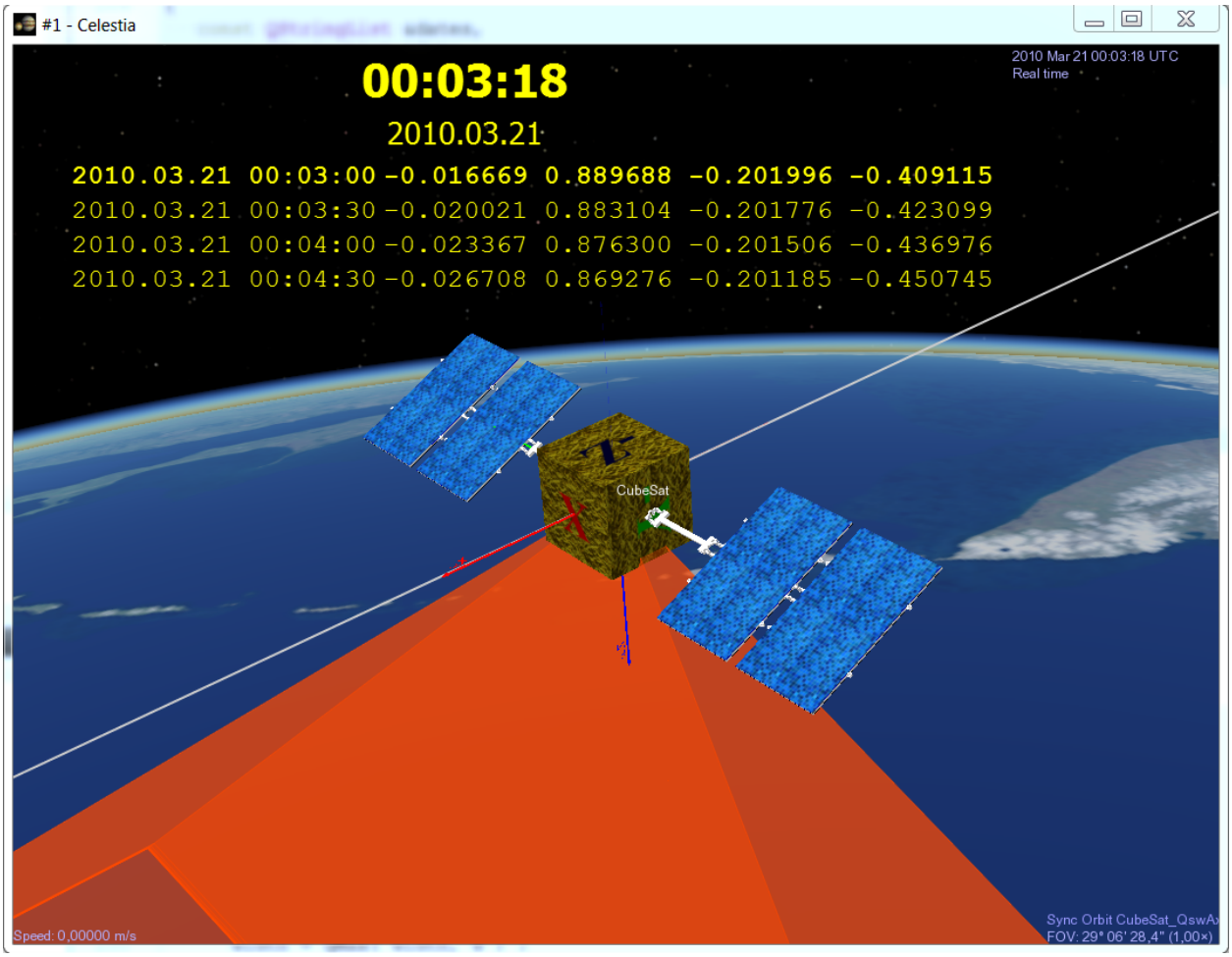


Fig. 5.81: InfoBox displaying data from a CIC data file

CHAPTER

SIX

PLUGIN DEVELOPMENT

6.1 Client applications in VTS

VTS is designed to interact with **client applications**. Client applications are started by VTS Broker and can communicate with it using the *Synchronization protocol for VTS clients*. This chapter describes VTS's client application architecture.

6.1.1 Architecture overview

A VTS project defines a set of entites (such as satellites, ephemeris, 3D models) and the client applications used to visualize them.

Client applications can be configured in the VTS configuration utility.

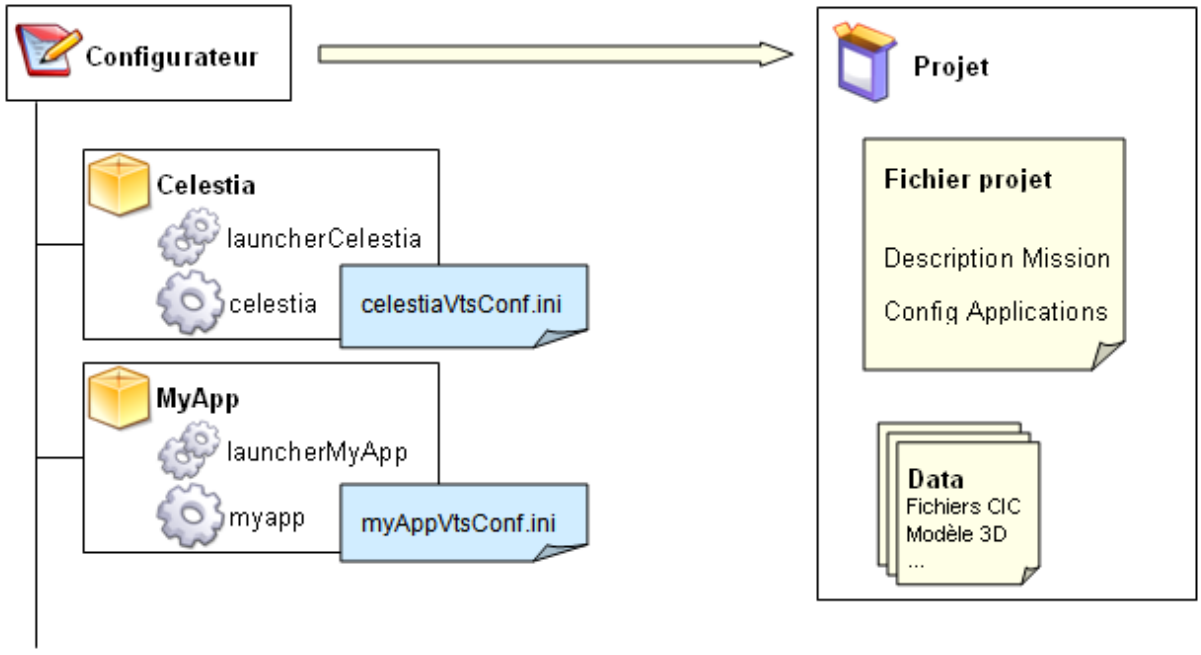


Fig. 6.1: Project configuration

Multiple instaces of the same application can be started by VTS, each instance has a unique **Application ID**.

Upon visualization startup, the Broker starts the **application launchers** for each instance of a client applications defined in the project file with the *AutoStarted* parameter.

Application launchers receive VTS parameters (such as the path to the project file and its application ID) and may perform any task in order to prepare the client application environment. The Broker uses the application luncher output to starts each client applications with the specified arguments.

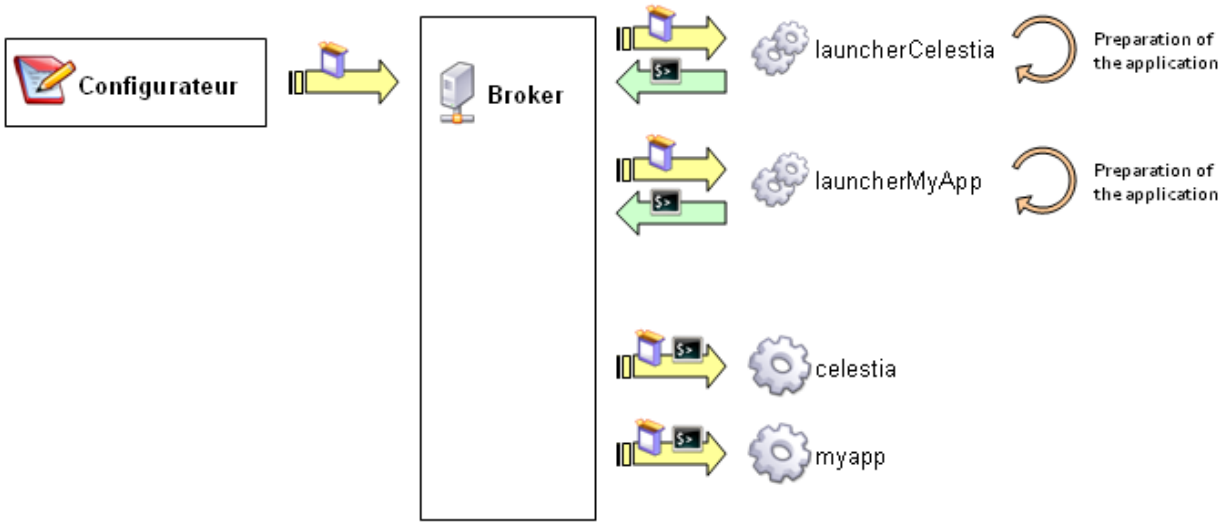


Fig. 6.2: Visualization startup

6.1.2 Initialization Sequence

1. The Broker reads the VTS project file and detects the list of client application instances to be started.
2. The Broker executes the corresponding application launchers for each client application instance.
3. The application launchers perform any task required to execute the client application and return the command line for application startup on standard ouput.
4. The Broker executes all client application instances.
5. Clients connect to the Broker on its socket and send the initialization message to indicate they are ready.
6. The Broker sends the commands for the initial states of the application declared in the *myAppVtsConf.ini* file, with default values or user-defined values from the VTS project.
7. Clients communicate with the Broker according the the synchronization protocol described in the *Synchronization protocol for VTS clients* chapter.

6.1.3 Files and directory structure

VTS automatically detects client applications based on the following:

- The application folder must be located in the `Apps/` folder of the VTS installation directory. The application folder name must start with an uppercase letter. This name will be used as the application's name.
- The `bin` subfolder of the application folder should contain: - the client application executable - the application launcher executable: its name must start by `launcher` followed by the application name (case-insensitive), see the dedicated chapter below. - The application cleaner executable: its name must start by `cleaner` followed by the application name (case-insensitive), see the dedicated chapter below.
- The `doc` subfolder of the application folder should contain: - a `vtsclient.json` configuration file. - a `README` file (its contents will be displayed in the VTS configuration utility). - a text file with its name starting by the application name and followed by `VtsConf.ini`. If present, this file must contain a description of the application's view properties (appearing in the view properties editor and initial properties editor for the application). - an icon file with `.png` or `.ico` file extension. If present, this icon will be used in the *Applications* tab of the Broker. If not present, the application executable's icon will be used instead.

Example for the Celestia client application:

Apps	Celestia	bin	celestia.exe
			launcherCelestia.exe
			cleanerCelestia.bat
		doc	README
			celestiaVtsConf.ini
			vtsclient.json
			icon.png

6.1.4 Client configuration file

The `vtsclient.json` configuration file enables advanced options to specify the process environment and the executables to be used.

File structure overview:

```
{
   "environment":
   {
      "VARIABLE": "value",
      "VARIABLE_FROM_ENV": "${EXISTING_ENV_VARIABLE}"
      ...
   },
   "winnt" :
   {
      "environment":
      {
         "VARIABLE": "winnt_value",
         "PATH": "C:/Python;${PATH}"
         ...
      },
      "launcher":
      {
         "environment":
         {
            "VARIABLE": "winnt_launcher_value"
         },
```

```
         "executable": "launcher.exe",
         "positionals": [ "file.txt"],
         "optionals": ["--option", "value"]
      },
      "client":
      {
         "environment":
         {
            "VARIABLE": "winnt_application_value"
         },
         "executable": "python.exe",
         "positionals": ["launcher.py"],
         "optionals": ["--option", "value"]
      },
      "cleaner":
      {
      }
   },
   "linux" :
   {
   }
}
```

The *environment* key

The system environment is inherited by all processes. Their environment can be modified by inserting a key corresponding to the variable name and a value.

- Define a new variable with a given value
- Define a new variable with the value of an existing variable. The following syntax is used: `${VARIABLE}`. Notice that this syntax is not OS-specific (as it is interpreted by VTS). However, the path separator is specific to the OS (';' on Windows and ':' on Linux).
- Overload an existing variable at any level: global (*root*), OS-specific (`winnt`, `linux`) or process-specific (`launcher`, `client`, `cleaner`)
- An injected environment variable (see below) can be used as a value

Injected environment variable

This variable are auto injected to a process environment.

- `VTS_WORKING_DIR`: points to the `bin/` subfolder of the application folder by default. This is the current directory of the process (equivalent to `$PWD`). A relative path will be relative to this folder. Modifying this variable will force the process current folder.
- `VTS_ROOT_DIR`: points to the VTS root folder.
- `VTS_LIBS_DIR`: points to the `Apps/Libs` folder.
- `VTS_APPLICATION_ID`: is the client application ID.
- `VTS_APPLICATION_TITLE`: is the client application title (`#<ID> - <label>`, the ID is dynamic, the label can be defined on the application configuration panel).

OS specific section

The following objects are used to describe a process by OS:

- `winnt`: Windows
- `linux`: Linux

Application specific section

The following objects are used to describe a process by application:

- `launcher`: application launcher executable
- `client`: client application executable
- `cleaner`: application cleaner executable

An application specific section can contain:

- `environment` object
- `executable` key: the value must be an executable file. It can be an abolute file path, or relative to the `VTS_WORKING_DIR`, or found in the `PATH`.
- `positionals` array key: can contain parameters passed as first arguments on the command line. Can be used to pass a script file to an interpreter as python or java.
- `optionals` array key: can contain parameters passed as last arguments on the command line.

For specific needs an additional option can change the behaviour of launchers output interpretation:

- `launcher-output` key: the value defines the launcher output mode and can be one of the following:
  - `ARGUMENTS` (default): launcher output is used as application arguments
  - `LOG`: launcher output is only used for logging
  - `DISABLED`: launcher is not started
  - `LEGACY`: launcher output is used as application arguments and the first argument must be the application ID

6.1.5 Application launchers

An application launcher is an executable intended for preparing the application's environnement and to specify how the application should be started. To these ends, the launcher can analyse the VTS project file in order to pre-process visualization data, generate scripts or even copy and move files. The only mandatory output of a launcher is the command line arguments to be passed to its client application.

Executing application launchers

Launchers are started by the Broker with the following command-line arguments:

```
[POSITIONALS] --project <projectFile.vts> --appid <appID> [OPTIONALS] [BROKER_OPTIONS]
```

Argument	Description
`[POSITIONALS]`	positional options specified in the `vtsclient.json` file
`--project <projectFile.vts>`	absolute path to the VTS project file
`--appid <appID>`	client application ID, which uniquely identifies the application instance during the visualization (as many instances of the same application may exsist)
`[OPTIONALS]`	additional options specified in the `vtsclient.json` file
`[BROKER_OPTIONS]`	optional Broker arguments: • `--serverport <port>`: server port number. See *Connecting to the Broker* for more information • `--datadir <dataDir>` directory containing project data files if different from project file's directory Relative file names must be resolved with this data directory. This is used by VTS when in read-only mode. • `--tempdir <tempDir>` temporary directory where launcher and application data should reside. This is used by VTS when in read-only mode

Preparing application environnement

The launcher should setup the client application's environment. This may include setting up configuration files, moving or copying project data files, or even converting data files into application-specific file formats (e.g. CIC/CCSDS files into *.xyz* or *.q* files for Celestia).

This step can be ommited if no specific task is needed to execute the client application.

Building the client application's command line

The launcher can provide the command line arguments for its client application. These will be appended to the application command line by the Broker. To return the command line arguments to the Broker, the launcher must print them on its standard output. Only the command line arguments may be printed to standard output. However, informational messages may be printed to standard error and will be logged by the Broker.

NOTE: since VTS 3.2 returning a command line argument is not mandatory anymore. See client configuration file described above.

Example of lines printed to standard ouput by applications launchers are shown below.

Celestia launcher:

```
0 --dir "C:/users/VTS/Apps/Celestia/bin" --conf "celestia.cfg" --url "C:/users/VTS/
↪Apps/Celestia/bin/extras_0/VTS/CubeSat.celx" --extrasdir "C:/users/VTS/Apps/
↪Celestia/bin/extras_0"
```

- The first field is mandatory and is the ID for the Celestia instance given as input to the launcher

- The following fields are the actual command line arguments for Celestia, and are standard arguments handled by the Celestia executable

SurfaceView launcher:

```
1 "C:/MyProject/visu.vts" 1
```

- The first field is mandatory and is the ID for the Celestia instance given as input to the launcher
- The second field is the absolute path to the project file, also given as input to the launcher, and required by the SurfaceView client application
- The third field is the client application ID given as input to the launcher, and required by the SurfaceView client application

Application launchers examples

The following scripts implement minimal launchers that directly return and empty command line argument for the client application. Only the client application ID is printed (mandatory).

- DOS Batch Programming:

```
@rem Input parameter %1 is the path to the VTS project file
@rem Input parameter %2 is the client application ID

@rem Mandatory client application ID output:
@echo %2

@rem End of file
```

- Shell Script:

```
#!/bin/sh
# Input parameter $1 is the path to the VTS project file
# Input parameter $2 is the client application ID

# Mandatory client application ID output:
echo $2

# End of file
```

6.1.6 Application cleaners

A **Clear all client application data caches** action is available from the *Configurator settings menu*. Each client application can provide a cleaner that removes all temporary files which could be let after many visualization runs. The executable will be started with the *–clear* option, as other options could appear in the future.

- Example for Celestia cleaner under Windows, launched by :

```
Apps/Celestia/bin/cleanerCelestia.bat --clear
```

- This cleaner removes all temporary folders containing 3D models and data files duplicated and generated for each run. It removes all the *extras_* temporary folders :

```
@echo off
for %%p in (%*) do (
   @rem Remove temporary folders
```

```
    if "%%p"=="--clear" (
        for /d %%a in (%~dp0/extras_*) do rd /s /q "%%a"

        @rem Return success
        exit /B 0
    )
)
```

6.2 Application IDs in VTS

Application IDs in a VTS visualization uniquely identify any client application taking part in the visualization.

There are three classes of application IDs: project IDs, dynamic IDs, and external IDs. The distinction between the three is merely used to quickly determine the origin of an application, and is not required to be strictly followed.

- **Project IDs** are attributed in sequence by the VTS configuration utility when adding a new client application to the project. The ID is chosen as the first available (i.e. not already attributed) application ID, starting from 1.
- **Dynamic IDs** are attributed in sequence by the Broker when starting a new client application for the visualization. The ID is chosen as the first available (i.e. not already attributed) application ID, starting from 100.
- **External IDs** are attributed in sequence by the Broker upon connection of a new client, if it was not started by the Broker and requires automatic attribution of an ID (by passing the `-1` ID in the `INIT` message sent to the Broker by the client). The ID is chosen as the first available (i.e. not already attributed) application ID, starting from 1000.

6.3 Synchronization protocol for VTS clients

Once all client application launchers have successfully terminated, the Broker starts all client applications with their respective command-line arguments. Client applications must then connect to the Broker via TCP. Once connection has been established, the VTS synchronization protocol is used for communication between the Broker and its clients.

6.3.1 Message syntax

Messages must abide by the following syntax rules in order to be correctly parsed by VTS and standard client applications:

1. Sent messages must be followed by a terminating newline character (`\n`)
2. Message fields which contain whitespace (spaces, tabs) must be quoted using double quotes (e.g. `"field with whitespace"`)
3. Literal quotes in message fields must be escaped with a backslash (e.g. `"field with \"literal quotes\""`)
4. Literal backslashes in message fields must be escaped with an additional backslash (e.g. `"C:\\path\\to\\file"`)

6.3.2 Connecting to the Broker

In order to engage communication with the Broker, **client applications must connect to a socket server setup by the Broker**.

- By default, the Broker server port is 8888. In this case, the launchers are started with no specific port option.
- If VTS is configured for open ports scanning, the launchers are started with a `--serverport <portNum>` option in command line. See the chapter *Broker Options* of the section *VTS configuration utility user manual* for more information.

6.3.3 Messages received by the Broker

This section describes the commands client applications may send to the Broker.

INIT message: connection to the Broker

Once connected to the server, client applications must sent the following initialization command to engage communication:

```
INIT <Name> <ClientType> [ProtocolVersion] [ClientId]
```

The parameters for this command are described below:

Parameter	Required	Description	Format (unit)	See also
Name	Yes	Name of the client	Character string	
ClientType	Yes	Time behavior of the client	CONSTRAINT or REGULATING	Real-time VTS
ProtocolVersion	Optional (default: 1.0)	Synchronization protocol version	Integer.Integer	
ClientId	Optional (auto attribution of an external ID if not provided)	Unique client ID	Integer	Application IDs in VTS

Commands can be sent by the client before INIT, but no TIME or DATA paquets will be received until the INIT message is sent to the Broker.

Below are some sample *INIT* commands:

```
INIT PrestoPlot CONSTRAINT 1.0 2
```

```
INIT Simulator REGULATING
```

TIME messages

Client applications may set the visualization date and time ratio. This is used for example when drag-and-dropping a satellite in the SurfaceView client application.

TIME commands have the following syntax:

```
TIME <TimeJD1950> [TimeRatio]
```

Parameter	Required	Description	Format (unit)	See also
TimeJD1950	Yes	Visualization date in JD1950 format	Real number	Date formats in VTS
TimeRatio	Optional	Time ratio	Real number	

The date must be provided in JD1950 format. The time ratio defines the speed ratio between visualization time and wall-clock time. If it is not provided, the current time ratio is kept.

For example, the following messages set the visualization date to July 9th, 2016 at 00:00:03, with a time ratio of 10 compared to real time (10 seconds of visualization time pass with each second of wall-clock time):

```
TIME 24296.000042 10.00000
```

Only clients with *CONSTRAINT* time behavior may provide a time ratio. For *REGULATING* clients, the actual time ratio is computed by the Broker.

CMD messages

Command messages allow some level of control over the visualization. They are divided in several categories:

- *CMD TIME* commands alter the flow of time (but do not modify the visualization date or time ratio: refer to the *TIME* messages above for this)
- *CMD SERVICE* commands control various aspects of the visualization unrelated to time
- *CMD TIMELINE* commands deal with the CIC/CCSDS files displayed in the timeline
- *CMD EVENT* commands deal with CIC/CCSDS mission event files for the visualization

CMD TIME messages

PAUSE command

The *PAUSE* command stops the time flow in the Broker. This is equivalent to pressing the **Pause** button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME PAUSE
```

When paused, the Broker keeps sending *TIME* messages with the current visualization date (even though it remains constant).

PLAY command

The *PLAY* command resumes the time flow in the Broker. This is equivalent to pressing the **Play** button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME PLAY
```

IncreaseTimeRatio command

The *IncreaseTimeRatio* command increases the time ratio in the Broker. The increase depends on the current time ratio. The predefined time ratio sequence is: 1x, 2x, 5x, 10x, 20x, 50x, 100x, etc. This is equivalent to pressing the corresponding button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME IncreaseTimeRatio
```

DecreaseTimeRatio command

The *DecreaseTimeRatio* command decreases the time ratio in the Broker. It is the opposite of the *IncreaseTimeRatio* command. This is equivalent to pressing the corresponding button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME DecreaseTimeRatio
```

SetTimeRatio command

The *SetTimeRatio* command sets a custom time ratio in the Broker. It is redundant with the *[TimeRatio]* parameter of the *TIME* message, but does not require a visualization date to be provided.

This command has the following syntax:

```
CMD TIME SetTimeRatio <TimeRatio>
```

Parameter	Required	Description	Format (unit)	See alos
TimeRatio	Yes	Time ratio	Real number	

For example, the following commands set a visualization time ratio 10x slower than wall-clock time:

```
CMD TIME SetTimeRatio 0.1
```

RevertTime command

The *RevertTime* command reverses the time flow direction. This is equivalent to pressing the corresponding button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME RevertTime
```

TimeModeChangedTo command

The *TimeModeChangedTo* command specifies de time mode currently in use. It can be useful for client to adapt their behavior according the mode.

This command has the following syntax:

```
CMD TIME TimeModeChangedTo <mode>
```

The mode can be “STANDALONE”, “REGULATED”, “SYSTIME” or “RECORDING”.

CMD SERVICE messages

AUTOCLOSE command

The *AUTOCLOSE* command instructs the Broker to close the visualization session. All client applications are disconnected, clients started by the Broker are terminated, and the Broker itself exits.

This command has the following syntax:

```
CMD SERVICE AUTOCLOSE
```

AUTORESTART command

The *AUTORESTART* command instructs the Broker to restart the visualization session. The Broker does the AUTOCLOSE process and then :

- Restarts itself with the same project file if the –externalRestart has not been set
- Closes with a special return code otherwise (i.e. it will be restart by the configuration utility)

This command has the following syntax:

```
CMD SERVICE AUTOCLOSE
```

Request replies

These commands are sent in reply of requests from the Broker. Refer to the *paragraph below* for more information on the various Broker requests.

StoreCommand command

The *StoreCommand* command is sent in reply to a *SaveState request* or *SaveWindow request* context save request. It instructs the Broker to store a view property command for the application into the current scenario state.

This command has the following syntax:

```
CMD SERVICE StoreCommand <ViewPropertyCommand>
```

Parameter	Required	Description	Format (unit)	See also
ViewPropertyCommand	Yes	Command for setting a view property of the application	Character string	CMD PROP commands, CMD STRUCT commands

Commands stored through a *StoreCommand* command will be sent back to the client application when the scenario state they are stored in becomes active. For more information on the project scenario, refer to the *Scenario in VTS* chapter.

SaveStateFinished command

The *SaveStateFinished* command instructs the Broker that all *StoreCommand command* (see above) in reply to a *SaveState request* or *SaveWindow request* context save request have been sent by the client application.

This command has the following syntax:

```
CMD SERVICE SaveStateFinished
```

Synchronized command

The *Synchronized* command is sent by clients in reply to a *SynchroRequested request*, to indicate that the client is synchronized and ready for screen capture.

This command has the following syntax:

```
CMD SERVICE Synchronized
```

The recording of high quality movies in VTS requires client applications to ensure that once a *Synchronized* command has been issued, the visualized scene in the sender will remain frozen until the next *TIME* or *CMD TIME PLAY* message received from the Broker. Refer to the *Recording movies* section in the *Broker user manual* chapter for more information on movies in VTS.

CaptureArea command

The *CaptureArea* command instructs the Broker to take a screenshot of a desktop area.

This command has the following syntax:

```
CMD SERVICE CaptureArea <X> <Y> <Width> <Height> <ImagePath>
```

Parameter	Required	Description	Format (unit)	See also
X	Yes	X coordinate of the area top left corner	Integer	
Y	Yes	Y coordinate of the area top left corner	Integer	
Width	Yes	Width of the captured area	Integer	
Height	Yes	Height of the captured area	Integer	
ImagePath	Yes	Image path	Character string	

StartApplication command

The *StartApplication* command instructs the Broker to start a new instance of the specified client application.

This command has the following syntax:

```
CMD SERVICE StartApplication <ApplicationName> [ApplicationId]
```

Parameter	Required	Description	Format (unit)	See also
Application-Name	Yes	Application name	Character string	
ApplicationId	Optional (default: -1)	Application ID	Positive or null integer	Application IDs in VTS

The provided application name must match that of a client application available to VTS. If the provided application ID is already used amongst already connected clients, the clients is restarted if it is in a sleeping state. If the ID is known and the application is running, the application is not restarted.

StopApplication command

The *StopApplication* command instructs the Broker to stop a running client application.

This command has the following syntax:

```
CMD SERVICE StopApplication <ApplicationId>
```

Parameter	Required	Description	Format (unit)	See also
ApplicationId	Yes (default: -1)	Application ID	Positive or null integer	Application IDs in VTS

The provided application ID must match that of a running client application. If the provided application ID is not used by any application or if the client application is already stopped, the command has no effect.

Please note that external applications (not launched by the Broker) cannot be stopped using this command.

CMD TIMELINE messages

CMD TIMELINE messages allow client applications to add, remove or reload CIC/CCSDS files to the timeline.

LoadFile command

The *LoadFile* command instructs the Broker to load a new CIC/CCSDS file, and display it in the timeline. If the specified file is already present in the timeline, this command has no effect.

This command has the following syntax:

```
CMD TIMELINE LoadFile <FilePath>
```

Parameter	Required	Description	Format (unit)	See also
FilePath	Yes	Absolute or project-relative path to a CIC/CCSDS file in OEM, AEM or MEM format	Character string	

ReloadFile command

The *ReloadFile* command instructs the Broker to reload an already loaded CIC/CCSDS file. If the specified file is not present in the timeline, this command has no effect.

This command has the following syntax:

```
CMD TIMELINE ReloadFile <FilePath>
```

Parameter	Required	Description	Format (unit)	See also
FilePath	Yes	Absolute or project-relative path to a CIC/CCSDS file in OEM, AEM or MEM format	Character string	

UnloadFile command

The *UnloadFile* command instructs the Broker to unload an already loaded CIC/CCSDS file from the timeline.

This command has the following syntax:

```
CMD TIMELINE UnloadFile <FilePath>
```

Parameter	Required	Description	Format (unit)	See also
FilePath	Yes	Absolute or project-relative path to a CIC/CCSDS file in OEM, AEM or MEM format	Character string	

Note that the specified file must have been loaded previously via a `CMD TIMELINE LoadFile` command. Files added in the VTS configuration utility may not be removed via the `CMD TIMELINE UnloadFile` command: they can only be removed from the VTS configuration utility or the Broker itself.

CMD EVENT messages

CMD EVENT messages allow client applications to add, remove or reload CIC/CCSDS mission event files attached to project entities. The Broker will process these commands and forward them to all clients (except the sender client).

LoadFile command

The *LoadFile* command instructs the Broker to load a new CIC/CCSDS event file, and attach it to a project satellite.

This command has the following syntax:

```
CMD EVENT LoadFile <EventFilePath> <SatelliteFullName>
```

Parameter	Required	Description	Format (unit)	See also
EventFilePath	Yes	Absolute or project-relative path to a CIC/CCSDS event file in MEM format	Character string	
SatelliteFullName	Yes	Full name of the satellite to attach the event file to	Character string	Object paths in VTS

ReloadFile command

The *ReloadFile* command instructs the Broker to reload an already loaded CIC/CCSDS event file. If the specified file is not an already loaded event file, this command has no effect.

This command has the following syntax:

```
CMD EVENT ReloadFile <EventFilePath>
```

Parameter	Required	Description	Format (unit)	See also
EventFilePath	Yes	Absolute or project-relative path to a CIC/CCSDS event file in MEM format	Character string	

UnloadFile command

The *UnloadFile* command instructs the Broker to unload and detach an already loaded CIC/CCSDS event file from all project satellites. Events from this file are no longer attached to any satellite.

This command has the following syntax:

```
CMD EVENT UnloadFile <EventFilePath>
```

Parameters	Required	Description	Format (unit)	See also
EventFilePath	Yes	Absolute or project-relative path to a CIC/CCSDS event file in MEM format	Character string	

Note that the specified file must have been loaded previously via a `CMD EVENT LoadFile` command. Event files configured in the VTS configuration utility may not be removed via the `CMD EVENT UnloadFile` command: they can only be removed from the VTS configuration utility itself.

FWD messages

FWD messages provide inter-client communication, by forwarding messages from one client to another.

These messages have the following syntax:

```
FWD <RecipientId> <Message>
```

table

Parameter	Required	Description	Format (unit)	See also
RecipientId	Yes	Application ID of the recipient	ALL character string Positive or null integer	Application IDs in VTS
Message	Yes	Message to forward to the recipient	Character string (may contain spaces)	

The recipient ID must be an existing client application ID for the current visualization. If the ID is not registered in the Broker, the command is bufferised and will be send when an application connect with this exact ID.

Client application IDs are displayed in the *Applications* tab of the Broker. The *ALL* character string may be used instead of a numerical ID to broadcast the message to all clients (except the sender).

The transferred message may contain spaces. It will be transferred as is to the recipient(s).

Below are a few sample forward messages:

- Hide solar arrays in all 3D clients displaying them (the clients must implement the *Visible* structural property for the command to have any effect):

```
FWD ALL CMD STRUCT Visible "Sol/Earth/CubeSat/GS" true
```

- Reset view position and zoom level in client ID 0 (for this command to have any effect, client ID 0 must be a SurfaceView client application):

```
FWD 0 CMD PROP ViewInfos Default
```

DATA messages

DATA messages are described in the *Real-time VTS* section below. They provide data for streamed values.

6.3.4 Common messages received by client applications

Some messages received by clients from the Broker are common across all client applications. This section describes those messages. Application-specific messages are described further below.

Messages received by client applications are categorized as follows:

- *TIME* messages concern time synchronization
- *CMD* messages control various aspects of the visualization
- *DATA* messages provide values for streamed data

Some commands have similar syntax as commands received by the Broker described in the above section.

TIME messages

Time messages synchronize client applications with the current visualization date. Client applications must handle these messages in order to properly synchronize with VTS.

When no *REGULATING* client is connected, the Broker broadcasts the current visualization date to all clients at a rate of 2 Hertz.

These messages have the following syntax:

```
TIME <TimeJD1950> <TimeRatio>
```

Parameters are the same as those described in the **TIME* messages* paragraph of the *Messages received by the Broker* section above, except that the time ratio is always provided.

Client applications which interpolate date between time ticks sent by the Broker may use the date provided in *TIME* messages to synchronize their interpolation loop.

Note that when the visualization is paused, the time ratio is set to 0.

CMD messages

Command messages control various aspects of the visualization for clients. They are divided in several categories:

- *CMD TIME* commands inform about alterations to the flow of time (but do not provide the visualization date or time ratio: refer to the *TIME* messages above for this)
- *CMD SERVICE* commands control various aspects of the visualization unrelated to time
- *CMD EVENT* commands deal with CIC/CCSDS mission event files for the visualization
- *CMD CAMERA* commands define the camera position and target for client applications handling camera commands
- *CMD PROP* commands set new values for specific view properties of client applications
- *CMD STRUCT* commands set new values for structural view properties of client applications

CMD TIME messages

PAUSE command

The *PAUSE* command informs client applications that visualization is paused. Clients may then stop all animation of the visualization scene, while still letting the user interact with their GUI.

This command has the following syntax:

```
CMD TIME PAUSE
```

Visualization date is still sent by the Broker when the visualization is paused. However it remains constant, and the time ratio is set to 0.

PLAY command

The *PLAY* command informs client applications that visualization has resumed from pause.

This command has the following syntax:

```
CMD TIME PLAY
```

CMD SERVICE messages

Initialization commands

Upon connection (after reception of the *INIT* messages), the Broker sends several initialization commands to inform client applications of the main objects for the visualization.

InitCentralBody command

The *InitCentralBody* command informs client applications of a central body used in the visualization. This may be used by client applications to trigger specific initialization for this central body. One such command is sent for each central body of the visualization.

This command has the following syntax:

```
CMD SERVICE InitCentralBody <CentralBodyName> <FullParentPath>
```

Parameter	Description	Format (unit)	See also
CentralBodyName	Name of a central body	Character string	Object paths in VTS
FullParentPath	Full path to the body's parent	Character string	Object paths in VTS

InitSatellite command

The *InitSatellite* command informs client applications of a satellite taking part in the visualization. This may be used by client applications to trigger specific initialization for this satellite. One such command is sent for each satellite of the visualization.

This command has the following syntax:

```
CMD SERVICE InitSatellite <SatelliteName> <FullParentPath>
```

Parameter	Description	Format (unit)	See also
SatelliteName	Name of a satellite	Character string	Object paths in VTS
FullParent-Path	Full path to the satellite's parent (should be a central body)	Character string	Object paths in VTS

Requests

Requests are special commands for which the Broker expects replies from client applications within a defined time frame. Refer to the *Request replies* paragraph above for more information on request replies.

SaveState request

The *SaveState* request informs client applications that a scenario state context save is ongoing, and instructs them to send commands for the view properties declared in their INI files back to the Broker.

This request has the following syntax:

```
CMD SERVICE SaveState
```

Client applications should reply using **StoreCommand* commands* described above. Once all properties have been sent back, a **SaveStateFinished* command* should be issued to inform the Broker that the context save is complete.

Note: Once the *SaveState* request has been issued by the Broker, client applications are expected to send all *StoreCommand* commands and the *SaveStateFinished* command within '*3 seconds*'. Beyond this time frame, the context save will time out for clients which have not responded.

Refer to the *Timeline toolbar* section of the *Scenario in VTS* chapter for more information on scenario state context save actions.

Note that this request does not concern the special *WindowGeometry* property. This property should only be sent in reply to a **SaveWindow* request* (see below).

SaveWindow request

The *SaveWindow* request informs client applications that a window position and geometry context save is ongoing, and instructs them to send back to the Broker the corresponding **StoreCommand* command*.

This request has the following syntax:

```
CMD SERVICE SaveWindow
```

Client applications should reply with a single *StoreCommand* command for the specific *WindowGeometry* property command. The value for this property is application-specific, e.g. *CMD SERVICE StoreCommand CMD PROP WindowGeometry 0 480 640 480*.

Note: Once the *SaveWindow* request has been issued by the Broker, client applications are expected to send back a *StoreCommand* command for the *WindowGeometry* property within **1 second**. Beyond this time frame, the context save will time out for clients which have not responded.

Refer to the *Timeline toolbar* section of the *Scenario in VTS* chapter for more information on window position and geometry context save actions.

SynchroRequested request

The *SynchroRequested* request instructs client applications that they should freeze the visualization at the last transmitted visualization date. This request is transmitted when a movie recording is ongoing in the Broker. Refer to the *Recording movies* section in the *Broker user manual* chapter for more information on movies in VTS.

This request has the following syntax:

```
CMD SERVICE SynchroRequested
```

Client applications should reply with the **Synchronized* command* once they are synchronized with the current visualization date. The Broker will capture a movie frame once all client applications have replied (or timed out).

Note: Once the *SynchroRequested* request has been issued by the Broker, client applications are expected to send back a *Synchronized* command within **500 milliseconds**. Beyond this time frame, the synchronization request will time out for clients which have not responded.

Other commands

ActivateWindow command

The *ActivateWindow* command instructs an application to raise its window if it's minimized and move to the front of other applications. This command should be implemented to enable one of the *Window Geometry Manager* feature.

This command has the following syntax:

```
CMD SERVICE ActivateWindow
```

TakeScreenshot command

The *TakeScreenshot* command instructs client applications to take a screenshot of the visualization scene. This may be used within scripts to save an image of the visualization at specific dates.

This command has the following syntax:

```
CMD SERVICE TakeScreenshot <Filename>
```

Parameter	Description	Format (unit)	See also
Filename	Name of the image file to save the screenshot to	Character string	

The file path must be relative to the project folder.

CMD EVENT messages

CMD EVENT messages instruct client applications to add, remove or reload CIC/CCSDS mission event files attached to project entities.

LoadFile command

The *LoadFile* command instructs client applications to load a new CIC/CCSDS event file, and attach it to a project satellite.

This command has the same syntax and parameters as the **LoadFile* command* received by the Broker. Refer to it for details.

LoadFile commands may be sent upon connection of a new client, to instruct it to load mission event files which were loaded by other client applications before it joined the visualization.

ReloadFile command

The *ReloadFile* command instructs client applications to reload an already loaded CIC/CCSDS event file.

This command has the same syntax and parameters as the **ReloadFile* command* received by the Broker. Refer to it for details.

UnloadFile command

The *UnloadFile* command instructs client applications to unload and detach an already loaded CIC/CCSDS event file from all project satellites. Events from this file should no longer be attached to any satellite.

This command has the same syntax and parameters as the **UnloadFile* command* received by the Broker. Refer to it for details.

CMD CAMERA commands

CMD CAMERA messages instruct client applications to change the visualization camera. Only messages corresponding to cameras declared in the client application's INI file may be received.

These commands have the following syntax:

```
CMD CAMERA <CameraName> <CameraParameters>
```

The list below describes available cameras:

- **Synchronous** Generic camera attached to a reference frame, pointing to the target object from the given frame axis. This is the camera used by VTS for INI camera types Body_Synchronous, Body_Inertial, Satellite_Inertial, Satellite_Sun, Satellite_SatFrame, Satellite_QswFrame, and Satellite_TnwFrame.

  Parameters:

  - **objName**: Celestia full name of the target object (e.g.: "Sol/Earth/CubeSat_ref/CubeSat")
  - **refObjectName**: Celestia full name of the reference frame to attach to (e.g.: "Sol/Earth/CubeSat_ref/CubeSat_Eme2000Axes", "Sol/Earth/CubeSat_ref/CubeSat_SunDir")
  - **direction**: frame axis from which to point to the object (X, -X, Y, -Y, Z, -Z or XYZ)
  - **distanceFactor** (optional): distance factor along the direction (relative to the default distance)

- **Goto** Go to the target object and attach to its reference frame. This is the camera used by VTS for INI camera types Body_Goto and Satellite_Goto.

  Parameters:

  - **objName**: name of the target object (e.g.: "Earth")

- **Center** Point towards the target object. The camera remains at its current location, attached to its current reference frame. This is the camera used by VTS for INI camera types Body_Center and Satellite_Center.

  Parameters:

  - **objName**: name of the target object (e.g.: "CubeSat")

- **CameraOrbitSat** Position the camera along the normal vector of the target object's orbital plane, pointing along that vector, so that the full orbit path of the target object can be seen. This is the camera used by VTS for INI camera type Satellite_Orbit.

  Parameters:

  - **objName**: name of the target object (e.g.: "CubeSat")

- **CameraSensorView** Position the camera from the target sensor's point of view, pointing along its aim direction. The field of view should be adjusted to match the sensor. This is the camera used by VTS for INI camera type Sensor_SensorView.

  Parameters:

  - **sensorName**: Celestia full name of the target sensor (e.g.: "Sol/Earth/CubeSat_ref/CubeSat/Sensor_sens_ref/Sensor")
  - **halfAngleOnX** and **halfAngleOnY** (real number, in radians): aperture half-angles of the target sensor
  - **up** (optional, "X" or "Y"): UP vector for the camera

- **CameraAzimuthElevation** This command is specific to a sensor defined with azimuth and elevation angles constaints alternatively to the CameraSensorView command. It sets the camera from the target sensor's point of view, pointing along its aim direction.

  Parameters:

  - **sensorName**: Celestia full name of the target sensor (e.g.: "Sol/Earth/Station_ref/Station/Sensor_sens_ref")
  - **minAzimuth**, **maxAzimuth**, **minElevation** and **maxElevation** (real number, in radians): aperture angles of the azimuth-elevation sensor
  - **up** (optional, "X" or "Y"): UP vector for the camera

- **WindowSensorView** Same as CameraSensorView, except that the window is resized to fit exactly the sensor.

  Parameters:

  - **sensorName**: Celestia full name of the target sensor (e.g.: "Sol/Earth/CubeSat_ref/CubeSat/Sensor_sens_ref/Sensor")
  - **halfAngleOnX** and **halfAngleOnY** (real number, in radians): aperture half-angles of the target sensor
  - **width** (integer number, in pixels): final width of the window. Height depends on sensor ratio.
  - **up** (optional, "X" or "Y"): UP vector for the camera.

Refer to the *Available cameras* section of the *Description of application properties* chapter for more information on camera capability declarations for client applications.

CMD PROP commands

CMD PROP commands set new values for specific properties of a client application. These properties are declared in the *[SPECIFIC]* section of the application's INI file.

Refer to the *Description of application properties* chapter for more information on application properties. These properties include for example the equatorial grid in Celestia or the zoom level and view position in SurfaceView.

These commands have the following syntax:

```
CMD PROP <PropertyName> <PropertyValue>
```

Parameter	Description	Format (unit)	See also
PropertyName	Property name	Character string	
PropertyValue	Property value	Property-specific (may be composed of several space-separated fields)	Messages received by Celestia and Messages received by SurfaceView

CMD PROP commands are sent upon modification of specific properties in the view properties editor of the Broker, and when a scenario state becomes active and its view properties are restored.

Below are some sample *CMD PROP* commands:

- Enable the equatorial grid in Celestia

```
CMD PROP equatorialgrid true
```

- Reset the view position and zoom level in SurfaceView

```
CMD PROP ViewInfos Default
```

CMD STRUCT commands

CMD STRUCT commands set new values for structural properties of a client application. These properties are declared in the entity sections of the application's INI file.

Refer to the *Description of application properties* chapter for more information on application properties.

These commands have the following syntax:

```
CMD STRUCT <PropertyName> <EntityFullName> <PropertyValue>
```

Parameter	Description	Format (unit)	See also
PropertyName	Property name	Character string	
EntityFullName	Full name of a target entity for the property	Character string	Object paths in VTS
PropertyValue	Property value	Property-specific (may be composed of several space-separated fields)	Messages received by Celestia and Messages received by SurfaceView

Below are some sample *CMD STRUCT* commands:

- Set the scale factor for Earth in Celestia

```
CMD STRUCT BodyScale "Sol/Earth" 0.8
```

- Disable the sensor footprint in SurfaceView

```
CMD STRUCT AimContourVisible "Sol/Earth/CubeSat/Sensor" false
```

6.3.5 Messages received by Celestia

Apart from the common messages described above, Celestia handles the commands corresponding to properties declared in its INI configuration file.

CMD PROP commands

Label Antialiasing Samples	**Name** AntialiasingSamples
Propagation mode INITIAL	**Default value** 1
Parameters format 0/1/2/4/8	
Antialiasing setting for Celestia.	

Label Sensor view camera offset	**Name** SensorCameraOffset
Propagation mode INITIAL	**Default value** 1
Parameters format Positive real number (meters)	
Position offset of the sensor view camera.	

Label Sensor geometry section count	**Name** SensorGeometrySectionCount
Propagation mode INITIAL	**Default value** 128
Parameters format Positive real number	
Number of points making up the outline polygon	

Label Sensor view camera offset	**Name** SensorSwathResolution
Propagation mode INITIAL	**Default value** 1
Parameters format Positive real number(s)	
Interval between two instantaneous sensor aiming surfaces	

Label Window geometry	**Name** WindowGeometry
Propagation mode MANUAL	**Default value** 0,0,640,480
Parameters format 4 integers (pixels)	
Window position and size (x, y, width, height)	

Label Equatorial grid	**Name** equatorialgrid
Propagation mode MANUAL	**Default value** false
Parameters format Boolean	
Display of the celestial equatorial grid	

Label Camera parameters	**Name** CameraDesc
Propagation mode MANUAL	**Default value** Default
Parameters format Character string	
Camera description (target, reference frame, position, field of view)	

Label Selected object	**Name** SelectObject
Propagation mode MANUAL	**Default value** "Sol/Earth"
Parameters format Character string	
Selected object in the 3D view (Celestia full name)	

Label Satellite labels	**Name** SatelliteLabelsVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Visibility of all satellite labels	

Label Solar system scale	**Name** SolarSystemScale
Propagation mode MANUAL	**Default value** 1
Parameters format Positive real number	
Scale factor for the solar system	

Label Window menu	**Name** WindowMenus
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the menu bar	

Label Window text	**Name** WindowText
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of text information on the current object	

Label Ambient Light	**Name** AmbientLight
Propagation mode MANUAL	**Default value** 0.3
Parameters format Real number within [0,1]	
Amount of articifial ambient light	

CMD STRUCT commands

Central body properties

Label Body scale	**Name** BodyScale
Propagation mode MANUAL	**Default value** 1.
Parameters format Positive real number	
Scale factor for the central body	

Label EME2000 inertial frame axes	**Name** Eme2000AxesVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the EME2000 inertial reference frame	

Label Body frame axes	**Name** FrameAxesVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the local body frame	

Label Planetographic grid	**Name** PlanetographicGridVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the planetographic grid	

Label All stations visibility	**Name** AllStationVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of all the ground stations	

Label `All POI visibility`	**Name** `AllPoiVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of all Points of Interest	

Label `All ROI visibility`	**Name** `AllRoiVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of all Regions of Interest	

Satellite and subpart properties

Label `Component visibility`	**Name** `Visible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Visibility of the object	

Satellite properties

Label `Satellite scale`	**Name** `SatelliteScale`
Propagation mode `MANUAL`	**Default value** `1.`
Parameters format `Positive real number`	
Scale factor for the whole satellite	

Label `Orbit path`	**Name** `TrackVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of the satellite's orbit path	

Label Orbit time window	**Name** TrackWindow
Propagation mode MANUAL	**Default value** 2 hours, 2 hours
Parameters format 2 positive real numers (hours) Duration before event, duration after event, in hours	
Duration of the satellite's orbit path before and after the satellite	

Label EME2000 inertial frame axes	**Name** Eme2000AxesVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the EME2000 inertial reference frame	

Label QSW local frame axes	**Name** QswAxesVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the QSW local orbital frame	

Label TNW local frame axes	**Name** TnwAxesVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the TNW local orbital frame	

Label Satellite frame axes	**Name** FrameAxesVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of the local satellite frame	

Label Sun direction	**Name** SunDirectionVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the Sun direction vector	

Label Body direction	**Name** BodyDirectionVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the satellite body direction vector	

Label Velocity vector	**Name** VelocityVectorVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the satellite velocity vector	

Sensor properties

Label Sensor contour	**Name** AimContourVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of the sensor base contour	

Label Sensor volume	**Name** AimVolumeVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of the sensor volume	

Label Sensor axis	**Name** AimAxisVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the sensor aim axis	

Label Sensor swath	**Name** AimTraceVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the sensor swath trace	

POI properties

Label POI visibility	**Name** PoiVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display the point of interest	

Label Text visibility	**Name** PoiTextVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display the point of interest's text	

Label POI icon color	**Name** PoiIconColor
Parameters format Color (i.e. #ffffff)	
Modify the point of interest's color	

The *POI icon color* command is only available via the VTS protocol :

```
CMD STRUCT PoiIconColor <POIFullName> <color>
```

ROI properties

Label ROI visibility	**Name** RoiVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display the region of interest	

Label Text visibility	**Name** RoiTextVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display the region of interest's text	

Label ROI contour color	**Name** RoiContourColor
Parameters format Color (i.e. #ffffff)	
Modify the region of interest's color	

The *ROI contour color* command is only available via the VTS protocol :

```
CMD STRUCT RoiContourColor <ROIFullName> <color>
```

Scriptable *CMD CAMERA* commands

Scriptable *CMD CAMERA* commands are not used by the Broker, but they can be used inside VTS scripts (refer to the *Scripts and macros in VTS* chapter for more information).

These commands have the following syntax:

```
CMD CAMERA <CameraName> <CameraParameters>
```

The table below describes available cameras:

Camera name	Parameters (unit)	Description
CreateMultiView	X Y: where X is the number of views horizontaly and Y the number of views verticaly	Creates multi viewports, each viewport has a scriptable camera.
SelectView	X Y: coordinates of the viewport to control in MultiView mode	Select a viewport, the next command will only affct this viewpot.
SetFov	angle: Field Of View angle in randians	Changes the current camera field of view

6.3.6 Messages received by SurfaceView

Apart from the common messages described above, SurfaceView handles the following commands corresponding to properties declared in its INI configuration file.

CMD PROP initial commands

Label `Scene reference`	**Name** `SceneReference`
Propagation mode `INITIAL`	**Default value** `"Default" (First defined body in the project)`
Parameters format `Character string (internal format)`	
Body name as a scene reference defining which body's surface will be visible in SurfaceView	

Label `Map Projection`	**Name** `MapProjection`
Propagation mode `INITIAL`	**Default value** `Equirectangular`
Parameters format `Character string`	
Projection name amongst the implemented projections	

Label Target framerate	**Name** TargetFramerate
Propagation mode INITIAL	**Default value** 10
Parameters format Positive integer number (Hz)	
Update frequency of the visualization. A high value will produce smoother animation but will consume more CPU time	

Label Track sampling resolution	**Name** TrackSamplingResolution
Propagation mode INITIAL	**Default value** 100
Parameters format Positive integer number (seconds)	
A low value will increase track accuracy at the expense of application's performance.	

Label Sensor geometry section count	**Name** SensorGeometrySectionCount
Propagation mode INITIAL	**Default value** 128
Parameters format Positiveinteger number	
Number of points making up the outline polygon	

Label Sensor coverage significant threshold	**Name** SensorCoverageSignificantThreshold
Propagation mode INITIAL	**Default value** 5.
Parameters format Positive real number (%)	
Minimal percentage of new coverage	

Label Sensor coverage merging threshold	**Name** SensorCoverageMergingThreshold
Propagation mode INITIAL	**Default value** 90.
Parameters format Positive real number (%)	
Minimal percentage for merging polygons	

Label Sensor coverage nadir optimization	**Name** SensorCoverageNadirOptimization
Propagation mode INITIAL	**Default value** .
Parameters format Boolean (false or true)	
Section Coverage swath of the SurfaceView user manual. Enable the sensor coverage nadir optimization	

CMD PROP specific commands

Label `Window geometry`	**Name** `WindowGeometry`
Propagation mode `MANUAL`	**Default value** `0,0,640,480`
Parameters format `4 integers (pixels)`	
Window position and size (x, y, width, height)	

Label `View parameters`	**Name** `ViewInfos`
Propagation mode `MANUAL`	**Default value** `"Default"`
Parameters format `Character string (internal format)`	
View position and zoom level	

Label `Projection center`	**Name** `ProjectionCenter`
Propagation mode `MANUAL`	**Default value** `0 0` or `1 0 0 0`
Parameters format `2 real numbers (latitude longitude)` or `4 real numbers (quaternion for polar projection)`	
Center of the projection	

CMD STRUCT commands

Sun properties

Label `Sub-solar point`	**Name** `SubEntityPointVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of the sub-solar point	

Label `Terminator`	**Name** `VisibilityCircleVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of the day-night terminator	

Label Track visibility	**Name** TrackVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of the Sun track	

Label Track time window	**Name** TrackWindow
Propagation mode MANUAL	**Default value** 2 hours, 2 hours
Parameters format 2 positive real numers (hours)	
Duration of the Sun track before and after the current position of the Sun icon	

Central body properties

Label Sub-entity point	**Name** SubEntityPointVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of the sub-entity point	

Label Visibility circle	**Name** VisibilityCircleVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of the visibility circle of this body on the scene reference (no effect on the scene reference)	

Label Track visibility	**Name** TrackVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of the track (no effect on the scene reference)	

Label Track time window	**Name** TrackWindow
Propagation mode MANUAL	**Default value** 2 hours, 2 hours
Parameters format 2 positive real numers (hours)	
Duration of the track before and after the current position of the sub-object icon	

Label Planetographic grid	**Name** PlanetographicGridVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display of the planetographic grid	

Label Terminator	**Name** TerminatorVisible
Propagation mode MANUAL	**Default value** false
Parameters format Boolean (false or true)	
Display the area where the body is visible on other bodies	

Label All stations visibility	**Name** AllStationVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of all Ground Stations	

Label All sensor station contours	**Name** AllSensorStationContourVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of all sensor station contours	

Label All sensor station surfaces	**Name** AllSensorStationSurfaceVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of all sensor station surfaces	

Label `All POI visibility`	**Name** `AllPoiVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of all Points of Interest	

Label `All POI text visibility`	**Name** `AllPoiTextVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of all POI text	

Label `All ROI visibility`	**Name** `AllRoiVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of all Regionsof Interest	

Label `All ROI text visibility`	**Name** `AllRoiTextVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of all ROI text	

Label `All ROI contour visibility`	**Name** `AllRoiContourVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of all ROI contours	

Label `All ROI fill visibility`	**Name** `AllRoiFillVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of all ROI filled surfaces	

Layer properties

Label Layer visibility	**Name** LayerVisible
Propagation mode MANUAL	**Default value** true
Parameters format Layer name (string) Visibility Boolean (false or true)	
Display of a layer	

Label Layer opacity	**Name** LayerOpacity
Propagation mode MANUAL	**Default value** 1.0
Parameters format Layer name (string) Opacity (EntityRange_t)	
Opacity of a layer	

Satellite properties

Label Track visibility	**Name** TrackVisible
Propagation mode MANUAL	**Default value** true
Parameters format Boolean (false or true)	
Display of the satellite's orbit path	

Label Track time window	**Name** TrackWindow
Propagation mode MANUAL	**Default value** 2 hours, 2 hours
Parameters format 2 positive real numers (hours) Duration before event, duration after event, in hours	
Duration of the satellite's orbit path before and after the satellite	

Label Visible events	**Name** VisibleEvents
Propagation mode MANUAL	**Default value** "*"
Parameters format Character string list	
List of visible event types	

Sensor properties

Label `Sensor footprint`	**Name** `AimContourVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of the sensor footprint (intersection with the central body)	

Label `Sensor swath`	**Name** `AimTraceVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display of the sensor swath trace	

POI properties

Label `POI visibility`	**Name** `PoiVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display the point of interest	

Label `Text visibility`	**Name** `PoiTextVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display the label of the point of interest	

ROI properties

Label `ROI visibility`	**Name** `RoiVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display the region of interest	

Label `Text visibility`	**Name** `RoiTextVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display the label of the region of interest	

Label `Contour visibility`	**Name** `RoiContourVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display the ROI contour	

Label `Fill visibility`	**Name** `RoiFillVisible`
Propagation mode `MANUAL`	**Default value** `true`
Parameters format `Boolean (false or true)`	
Display the ROI fill	

CMD SERVICE messages

CreateROI command

The *CreateROI* command instructs SurfaceView to instantiate a new Region in the specified body, and correpsonding to the specified coordinates.

This command has the following syntax:

```
CMD SERVICE CreateROI <ROIName> <BodyName> <Coordinates>
```

Parameter	Required	Description	Format (unit)
ROIName	Yes	Name given to the new ROI.	Character string
BodyName	Yes	Full name of the receiving body. (ex: “Sol/Earth”)	Character string
Coordinates	Yes	Space-separated list of comma-separated Lat/Long coordinnates. (ex: “15,2 -1,-15 -15,1 -1,15”)	Character string

6.3.7 Real-time VTS

Time synchronization with an external time source

By default, the VTS Broker regulates the flow of visualization time for all client applications. This is the **playback** visualization mode: visualization data is read from CIC/CCSDS data files and time flow can be fully controlled

(play/pause, time ratio, flow direction, etc.).

The **real-time** visualization mode allows VTS to synchronize with an external time source. Upon connection to the Broker, a client wishing to control the visualization time may declare itself as *REGULATING* (as described in the **INIT* message: connection to the Broker* section above). The Broker then switches to real-time mode:

- time control buttons and commands are disabled
- visualization date is received from the regulating client

Note that several regulating clients may not connect to the Broker simultaneously.

The regulating client must send *TIME* messages to the Broker to set the visualization date, as described in the *TIME* messages section above, e.g.:

```
TIME 22105.5468754
```

To ensure good quality of synchronization for the visualization date, a *TIME* message should be sent every second of wall-clock time. Increasing the rate will not necessarily increase the quality of synchronization, due to communication delays and congestion which are better smoothed out by interpolation at a synchronization rate of 1Hz. Decreasing the rate is discouraged as it may lead to misleadingly inaccurate interpolation in client applications.

The VTS Broker estimates the current visualization date based on the *TIME* messages received from the regulating client, and relays the visualization date to all constrained clients, similarly to what happend in *playback* mode.

To ensure correct interpolation of streamed data, the Broker may introduce a delay between the moment it receives the current visualization date from the regulating client and the moment it relays it to constrained clients. This delay is based on the refresh rate of *Stream* data: it is computed as twice the maximum refresh rate of all *INTERPOL* mode streamed data. For example, if all streamed data values from a simulator are broadcasted every second, the Broker will introduce a 2-seconds delay in visualization time. However, if all streamed data are in *DIRECT* mode, no delay is introduced.

Lastly, it should be noted that it is important for the visualization date sent by the regulating client to be recent, i.e. it must be a good representation of the regulating client's clock. The Broker uses the system time at packet reception to relate system time to regulating client time.

Supplying real-time data

Besides visualization date, data values may also be supplied by a client application. This is the purpose of *DATA* messages:

```
DATA <ProviderID> <JD1950 date> <data ID> "<data value>"
```

For example:

```
DATA 123 20447.000174 pos "-6538.3475061863419 2703.5361504162843 197.30707005759857"
DATA 123 20447.000174 quat "0.22339793344197931 0.65359251975872334 0.
↪53489502701655522 0.48661842497202529"
DATA 123 20447.000174 angle "0.175"
```

The sections below describe the parameters to a *DATA* message.

Note that previous versions of VTS required *DATA* messages to provide a protocol version number. Since this version number is now specified in the *INIT* message, they should no longer be provided with *DATA* messages. In order to retain compatibility with previously written clients, the Broker still handles a version number in *DATA* messages.

Data Provider ID

The Provider ID field corresponds to the ID of the client application that provides the data.

This field can be used by data consumers in order to distinguish data sources, as two different providers might provide the same parameters.

Data value date

The <*JD1950 date*> field provides the recording date of the data value. In most cases, this date is the same as the sending date of the message.

Due to the visualization date delay introduced by the Broker, clients receive data messages slightly in advance. This allows them to store the values and interpolate data for the current visualization date (as opposed to extrapolation were no future data values available).

Data identifier

The <*data ID*> field provides the unique identifier of the data. This ID is matched with the *Stream ID* set for streamed data in the VTS configuration utility. Refer to the *Data sources in VTS* chapter for further information on streamed data configuration.

Data value

The <*data value*> field provides the value for the data at the given date. The value must be quoted, no matter its dimension (scalar value, vector, etc.). The unit for the value depends on the data type, e.g.:

- Position of a satellite: kilometers
- Position of a satellite subpart: meters
- Quaternion: no unit
- Angle: degrees

6.3.8 Sample session

This example shows the messages sent by a regulating client defining a square equatorial orbit (don't try this at home!), at 100x wall-clock time:

Sending date	Message
T0	INIT example REGULATING
T0	TIME 23000.000000
T0	DATA 23000.000000 pos “10000 0 0”
T0+1s	TIME 23000.001157
T0+1s	DATA 23000.001157 pos “0 10000 0”
T0+2s	TIME 23000.002315
T0+2s	DATA 23000.002315 pos “-10000 0 0”
T0+3s	TIME 23000.003472
T0+3s	DATA 23000.003472 pos “0 -10000 0”
T0+4s	TIME 23000.004630
T0+4s	DATA 23000.004630 pos “10000 0 0”
T0+5s	TIME 23000.005787
T0+5s	DATA 23000.005787 pos “0 10000 0”
T0+6s	Etc.

6.4 Description of application properties

The VTS toolkit relies on a command mechanism to control client applications from the Broker in a centralized way. This mechanism is used to send view properties commands upon activation of a new scenario state.

Client applications may describe view properties which will be displayed in the *view properties editor* of the VTS configuration toolkit or of the Broker. Initial properties sent to clients upon connection may also be described.

3D client applications may also declare compatible cameras available in the *3D Cameras* tab of the Broker. It is the client application’s responsibility to ensure it is compatible with the cameras it declares to have available.

6.4.1 Application properties file format

The applications properties file must be located in the **doc** subfolder of the application folder with the following name: *<application name>VtsConf.ini*

This file is in standard INI format. It must be composed of sections with uppercase names. Properties in each section are stored in associative arrays with keys prefixed by a number, in order to preserve the order of declarations when displaying the properties in VTS. Array keys must be written in lowercase.

Important: array keys in a single section are numbered from 1 to N. The section must hold the “size = N” statement.

6.4.2 Section list

- **[INITIAL]**: Initial properties for the application. These properties are configured in the application’s pane in the *Structure* tab of the VTS configuration utility. The propagation mode must be set to *INITIAL* for all properties in this section.

- **[SPECIFIC]**: Specific properties for the application.
- **[BODY]**: Structural properties for central bodies.
- **[SUN]**: Specific properties for the Sun.
- **[COMPONENT]**: Structural properties for satellite components. These will be available for the root component of a satellite and for all of its subparts.
- **[SATELLITE]**: Structural properties for satellites.
- **[SENSOR]**: Structural properties for satellite sensors.
- **[STATION]**: Structural properties for ground stations.
- **[CAMERAS]**: Compatible cameras for the application.
- **[ROI]** and **[POI]**: Properties for regions and points of interest.
- **[LAYER]**: Properties for layers.

6.4.3 Property declaration

Each property must be declared with the following required fields:

- **name**: Name of the property. This name must be unique within the current INI section. It will be used as command name in the *VTS synchronization protocol*.
- **type**: Property data type. This may be a basic type, a Qt type, or a VTS type. See below for further details.
- **defaultValue**: Default value for the property.
- **propagation**: Propagation mode for the property. See below for further details.
- **label**: Label of the property, displayed in the view properties editor.

Below is an example of an application properties file:

```
[INITIAL]

; XMLFile
1/name = XMLFile
1/type = DataFile_t
1/defaultValue = "Data/ft.xml"
1/propagation = INITIAL
1/label = Transfer Function File

size = 1


[SPECIFIC]

; WindowGeometry
1/name = WindowGeometry
1/type = QRect
1/defaultValue = @Rect(0 0 640 480)
1/propagation = MANUAL
1/label = Window geometry

size = 1


[SATELLITE]
```

```
; SatelliteScale
1/name = SatelliteScale
1/type = SatelliteScale_t
1/defaultValue = 1.0
1/propagation = MANUAL
1/label = Satellite scale

size = 1
```

6.4.4 Available property types

There are three categories of property types: basic types, VTS types, and Qt types.

Basic types

The table below lists the available basic types:

Type	Description	Example	Editor
bool	Boolean value	true, false	Checkbox
int	Integer number	42	Text box
double	Real number	1.618	Text box

VTS types

VTS types are convenience types handled by VTS.

The table below lists the available VTS types:

Type	Description	Example	Editor
Color_t	RGB color	#00ff00	Color coded on an hexadecimal string, no alpha channel
EntityScale_t	Scale factor for a celestial body (double)	0.5	Slider with a zoom factor of 1000
SatelliteScale_t	Scale factor for a satellite (double)	1000	Slider with a zoom factor of 100
EntityRange_t	Interval of real values [0,1]	0.75	Slider with min value 0, max value 1, default value 30%
DataFile_t	Relative path to a data file in the project folder	"Data/config.xml"	Text field with browse button and copy dialog if the selected file is outside the project folder
ExternalFile_t	Absolute path to a data file (portability of the project may break on other machines)	"C:/Generate/todo.txt"	Text field with browse button allowing absolute paths
CameraDesc_t	Camera description for Celestia	N/A	Specific camera editor
TimeWindow_t	Time distribution before and after a dated event. Durations should be in hours.	5 1	Slider setting the total duration distribution bewteen before and after the event. Changing the "before" or "after" duration modifies the total duration but not the distribution.
Font_t	Font	Arial,12,-1,5,50,0,0,0,0,0	Font described in a string: <family>,<pointSizeF>,<pixelSize>,<styleHint>,<weight>,<style>,<underline>,<
ProjectNamedObject_t	Named object defined in the project	Sol/Earth/Cube1	Drop-down list containing object defined in the project.
CelestiaNamedObject_t	Named object defined in Celestia	Sol/Earth/Cube1	Drop-down list containing object defined in the project. It can also contain a object selected from Celestia, like a star for example.

Qt types

Qt types are those handled by the QVariant class from the Qt library.

The table below lists the available Qt types:

Type	Description	Example	Editor
QString	Character string	"Sol/Earth"	Text box
QStringList	List of character strings (comma-separated)	"SurfaceView", "Celestia"	Text box
QRect	Rectangle (integer coordinates)	@Rect(0 0 640 480)	Spin boxes (X, Y, width, height)

6.4.5 Available propagation modes

The following propagation modes are available:

- **MANUAL**: The property may be set independently for each scenario state. Its value may be propagated across states with the propagation buttons in the view properties editor.
- **AUTO**: The property has a unique value for all scenario states. Its value is automatically propagated across the whole scenario upon modification. Its name is displayed in italics in the view properties editor. This propagation mode should be used in rare cases where the property should remain consistent during the whole project, for example the visibility of a toolbar.
- **INITIAL**: The property is sent to the client application upon startup and may not be altered dynamically nor resent. This propagation mode must be used only for properties in the *INITIAL* section.

6.4.6 Available cameras

The *[CAMERAS]* section lists VTS cameras available in the client application. Refer to the *3D Cameras* tab section in the *Broker user manual* chapter for a description of all VTS cameras. For each available camera, several orientations must be implemented.

```
[CAMERAS]
1/type = Body_Synchronous
2/type = Body_Inertial
size = 2
```

The following VTS cameras may be declared:

Target entity	Camera name	Corresponding Broker cameras
Central body	Body_Synchronous	Fixed in Earth frame: `North pole` and `South pole`
	Body_Inertial	`Inertial`
	Body_Goto	`Goto`
	Body_Center	`Center`
Satellite	Satellite_Inertial	Inertial cameras
	Satellite_Sun	Sun and Body cameras : `View from Sun` and `View toward Sun`
	Satellite_SatFrame	Satellite frame cameras
	Satellite_QswFrame	Sun and Body cameras : `View from Body` and `View toward Body` QSW frame cameras
	Satellite_TnwFrame	TNW frame cameras
	Satellite_Orbit	Miscellaneous cameras : `Orbit`
	Satellite_Goto	Miscellaneous cameras : `Goto`
	Satellite_Center	Miscellaneous cameras : `Center`
Sensor	Sensor_SensorView	`Sensors`

Refer to the **CMD CAMERA* commands* section in the *Synchronization protocol for VTS clients* chapter for more information on the corresponding camera messages sent to client applications.

6.4.7 Usage in the VTS synchronization protocol

Properties and cameras declared in the INI file are used by the VTS synchronization protocol to generate commands for client applications. Properties in the *[INITIAL]* and *[SPECIFIC]* sections are translated into **PROP** commands, while those in entity-specific sections (*[BODY]*, *[COMPONENT]*, *[SATELLITE]*, and *[SENSOR]*) are translated into **STRUCT** commands. Cameras in the *[CAMERAS]* section result in corresponding **CAMERA** commands being sent.

Specific properties:

```
CMD PROP <PropertyName> <PropertyValue>
```

For example:

```
CMD PROP WindowGeometry 0 1 640 480
```

Structural properties: .. code-block:: bash

    CMD STRUCT <PropertyName> <EntityFullName> <PropertyValue>

For example:

```
CMD STRUCT AimContourVisible "Sol/Earth/CubeSat/Sensor" true
```

Cameras:

```
CMD CAMERA <CameraType> <CameraParameters>
```

Note that the *CameraType* parameter is not identical to the camera name listed in the *[CAMERAS]* section of the INI file.

For example:

```
CMD CAMERA CameraSensorView "Sol/Earth/CubeSat_ref/CubeSat/Sensor_sens_ref/Sensor" 0.
↪872665 0.349066
```

6.4.8 Application properties guidelines

For a better integration and taking advantage of all VTS features, here are some recommended properties an application should take into account (property declaration in the ini file and protocol command implementation):

Property	Section	Type	Default value	MANUAL	Description	Received command
WindowGeometry	SPECIFIC	QRect	@Rect(0 0 640 480)	MANUAL	Sets the window geometry of an application	CMD PROP WindowGeometry 0 0 640 480
AlwaysOnTop	SPECIFIC	bool	true/false	MANUAL	Sets the always on top property of an application	CMD PROP AlwaysOnTop false
Frameless	SPECIFIC	bool	true/false	MANUAL	Sets the frameless property of an application	CMD PROP Frameless false

6.5 VTS project file format

A VTS project file describes all the elements of a visualization. It is written in XML format for clarity. Its name must end with the *.vts* extension.

6.5.1 Notation

The sections below are structured as a tree representing the XML tags contained in a VTS project file.

Section names follow this nomenclature:

- Tags written as <*Tag*> may contain other tags.
- Tags written as <*Tag*/> are self-contained (they may not contain other tags).
- Tags written as <**Tag**>* are generic tags which are used in various contexts.

Generic tags are indicated under the sections of tags they may be child of. However, they are only fully described in separate sections outside of the main structure.

6.5.2 File format schema

Definition

Starting from VTS 3.2, each release of VTS comes with its **validation schema** file. Theses files can be found in the *VTS/Docs/Schemas* folder. A validation schema consist of an XML representation of how the VTS project file (also XML) must be organized. VTS validation schemas use the standard **XML Schema Definition** syntax (known as XSD).

Generating VTS projects from the XSD

Since the VTS Schema are described using the standard XML Schema Definition, it is recognized and understood by many software and libraries. In particular, VTS schemas can be used to automatically generate valid VTS projects using almost any modern programming languages. The following chapters describe how to generate a VTS project using *Python* and *C#*.

VTS project generation with Python

In order to generate a VTS project with Python, one can use *PyXB*. PyXB is a set of Python commands capable of generating a Python library from a XSD that:

- gives access to a memory representation of the VTS project file,
- can read a VTS project to this memory representation,
- can write a VTS project from this memory representation.

Using this library one can easily create a project and, for example, add programmatically satellites, components, bodies, layers of any kind, etc.

Installing PyXB

Python 2.7 or 3.x must be available on the computer. We assume the PATH environment variable is set to its directory.

Download PyXB (and unzip if necessary) anywhere on the computer.

PyXB can be found here: https://github.com/pabigot/pyxb (use **git clone** or download it as a ZIP).

Generating a library from the XSD using PyXB

Open a terminal in the PyXB folder and execute **python setup.py install**

Go to the **PyXB/build/scripts*** folder and run **python pyxbgen -m VTSProject -u <path to VTSProject-xxx.xsd>**

This will output the file **VTSProject.py**

Usage example of the generated library

The following example shows how to:

- Open a project "CubeSat.vts",
- Change its Start Time using a MJD Date,
- Create a hundred new satellites with "Sol/Earth" as parent,
- Write a new project file with all these changes.

```
import pyxb
import VTSProject

xml = open("CubeSat.vts").read()
project = VTSProject.CreateFromDocument(xml)

project.General.StartDateTime = VTSProject.MJDDateTimeType("55277 81261.150400")

for i in range(0,100):
    sat = VTSProject.SatelliteType()
    sat.Name = "sat_" + str(i)
    sat.ParentPath = "Sol/Earth"
    project.Entities.Satellite.append(sat)

f = open('CubeSatModified.vts', 'w')
pyxb.RequireValidWhenGenerating(True)
f.write(project.toxml("utf-8"))
```

VTS project generation with C#

Generating a library from the XSD using Visual Studio Community

Install **Visual Studio Community 2017**.

Open the **Developer Command Prompt** and go to the XSD folder.

Enter the command **xsd <name of the XSD file> /classes**.

This will output the source file **<name of this input XSD>.cs**.

Usage example of the generated library

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using System.Xml.Schema;
using System.Xml.Serialization;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            XmlSerializer serializer = new XmlSerializer(typeof(Project));
            Project project;

            XmlSchemaSet sc = new XmlSchemaSet();
            sc.Add(null, @"VTSProject-3.2.xsd");

            XmlReaderSettings settings = new XmlReaderSettings();
            settings.ValidationType = ValidationType.Schema;
            settings.ValidationFlags =
                System.Xml.Schema.XmlSchemaValidationFlags.ReportValidationWarnings |
                System.Xml.Schema.XmlSchemaValidationFlags.ProcessSchemaLocation;
            settings.Schemas = sc;
            settings.ValidationEventHandler += new␣
↪ValidationEventHandler(ValidationCallBack);

            string xml = System.IO.File.ReadAllText(@"CubeSat.vts");
            using (XmlReader reader = XmlReader.Create(new StringReader(xml),␣
↪settings))
            {
                project = (Project)serializer.Deserialize(reader);
            }

            FileStream fout = new FileStream(@"CubeSat.out.vts", FileMode.Create);
            serializer.Serialize(fout, (object)project);
        }

        // Display any validation errors.
        private static void ValidationCallBack(object sender, ValidationEventArgs e)
        {
            Console.WriteLine("Validation Error: {0}", e.Message);
        }
    }
}
```

CHAPTER

SEVEN

DATA GENERATOR INTEGRATION

The VTS toolkit is conceived to allow generic integration of data generator applications to provide CIC/CCSDS data to a VTS visualization. Data generator applications must provide a number of interfaces in order to fully integrate with the toolkit. This chapter describes the interfaces to implement for new data generator applications.

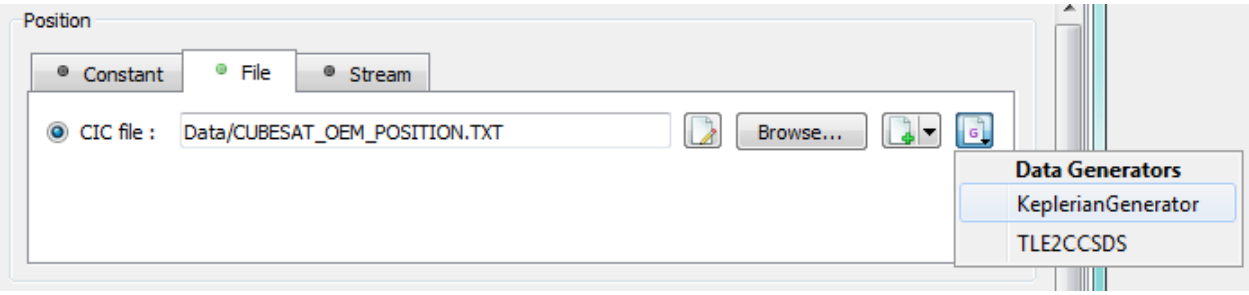


Fig. 7.1: *Position generation*

7.1 Generators folder hierarchy and nomenclature

Data generator applications must follow the following folder hierarchy and nomenclature:

- The generator folder must be located in the **Generators** folder of the VTS toolkit. The new generator folder name must start with an uppercase letter. This name will be used as the generator application's name.
- The generator executable must be located in the **bin** subfolder of the generator folder.
- The generator executable must have the same name as the generator folder (case-insensitive). On a Windows platform the executable must have the *.exe* or *.bat* file extension. On a Linux platform the name can be suffixed by *.sh*, *.lnx* or might not be suffixed.
- The **doc** subfolder of the generator folder may contain a text file with its name starting by the application name and followed by *VtsConf.ini*. If present, this file must contain the generator capacities for generating data.

Sample folder hierarchy for the Keplerian Generator :

Generators	KeplerianGenerator	bin	keplerianGenerator.exe
		doc	keplerianGeneratorVtsConf.ini

If the generator is installed outside the VTS toolkit, a proxy script file (.bat, .sh) file can be used to start the external generator application.

7.1.1 Data generation capacities

If the configuration file (xxxVtsConf.ini) doesn't exist, the generator is handled as a position data generator. If it exists, it can contain the keys listed in the table below (note the default values are false except for the "GenPosition" key which is true by default). A data generator can declare several keys according to its generating capabilities.

Key	Value	Default	Usage
GenPosition	true or false	true	Generate position data
GenQuaternion	true or false	false	Generate quaternion data
GenEuler	true or false	false	Generate euler angle data (see CCSDS documentation)
GenAxis	true or false	false	Generate axis data (X, Y, Z)
GenAngle	true or false	false	Generate angle data (degree or radian, see CCSDS documentation)
GenDirection	true or false	false	Generate direction data (X, Y, Z)
GenAltAz	true or false	false	Generate altitude/azimuth data

7.1.2 Command line parameters

When the data generator is available in VTS, it will be started with the following command line parameters :

- *–vtsgentype [RequestedGenerationType]* with [RequestedGenerationType] equal to "GenPosition", "GenQuaternion", "GenEuler", "GenAxis", "GenAngle", "GenDirection" or "GenAltAz" corresponding to the current generating capability requested
- *–vtsdefaultfile [DefaultOutputFilePath]* with [DefaultOutputFilePath] equal to the current output file path selected in VTS
- *–vtsmjddates [[StartDate] [EndDate]]* with [[StartDate] [EndDate]] as two string parameters, and each date encoded in MJD dates defining the current project dates

7.1.3 Optional standard output reply

When the generating process has ended, the generator may print to standard output a different output file than the provided [DefaultOutputFilePath]. All standard output is processed by VTS seeking a line containing a string starting with "outputvtsfile=" followed by a file path. This file path will be used to update the file entry in the VTS interface. The file path may contain spaces.

7.2 Orbit propagation with the Keplerian Generator

The Keplerian Generator allows automatic generation of CIC/CCSDS files containing position ephemerides for an elliptical keplerian orbit, in a defined time range. Positions are computed in the EME2000 reference frame.

The Keplerian Generator relies on the Orekit library to compute the ephemerides. Please refer to this library's documentation for further information on how the computations are performed.

Note: Java 8 runtime environment (JRE 1.8.0) must be installed on your system in order to use the Keplerian Generator. Older Java versions (at lease >v1.4) may also work but are not tested. Only the the 32bit JRE is supported

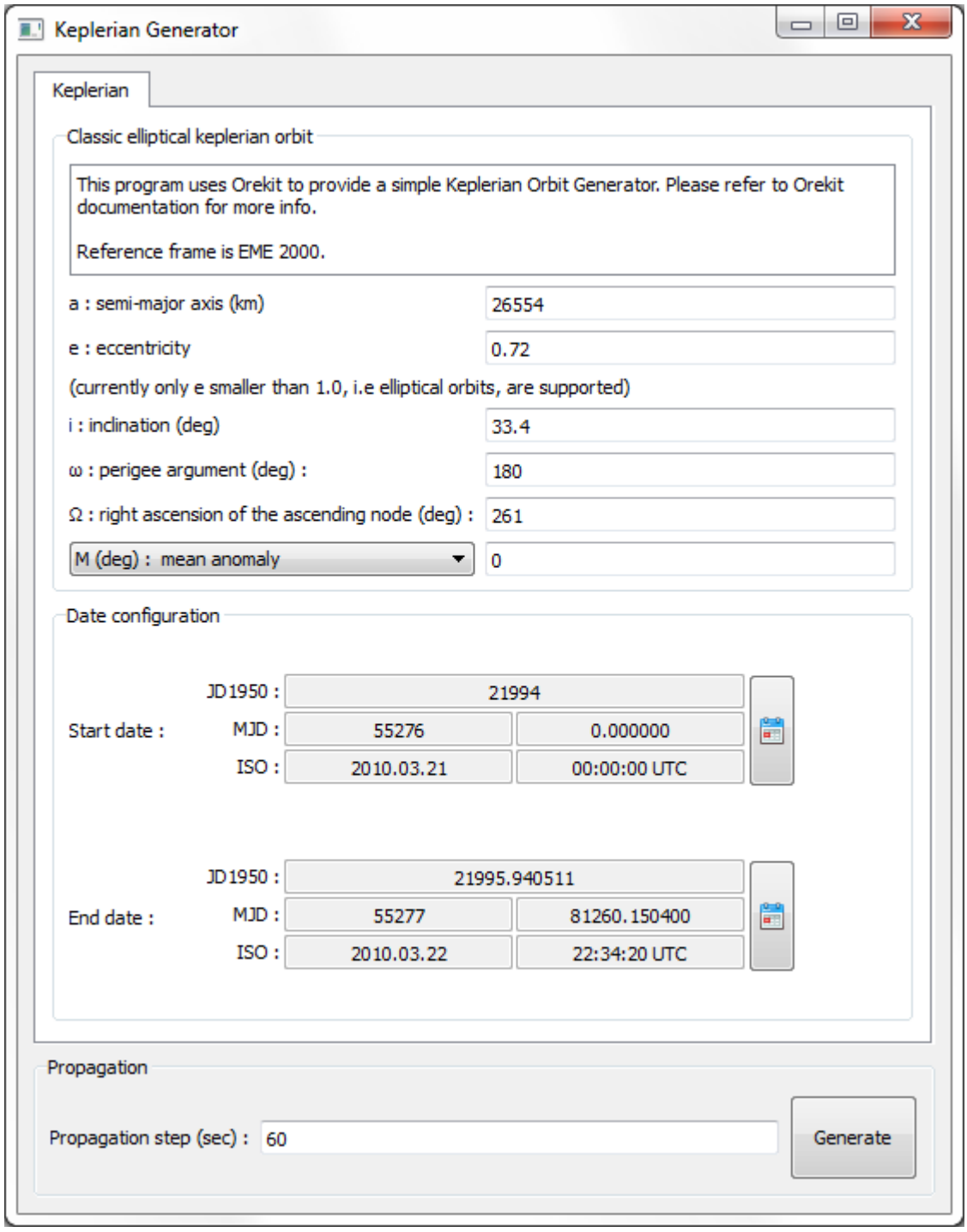


Fig. 7.2: *Keplerian Generator*

on Windows.

7.2.1 Using the Keplerian Generator

To generate an orbit ephemerides file, all orbital parameters in the *Classic elliptical keplerian orbit* area must be filled in. Note that only elliptical orbits are currently supported, i.e. orbits with eccentricity strictly below 1.

The start and end dates for the CIC/CCSDS file must also be defined in the *Date configuration* area, as well as the time interval between two position values (in seconds).

Clicking the **Generate** button prompts the user for the output file location and name, then generates the orbit data. The output data file can then be used to define the position of a satellite in the VTS configuration utility.

7.3 Orbit propagation with the TLE2CCSDS Generator

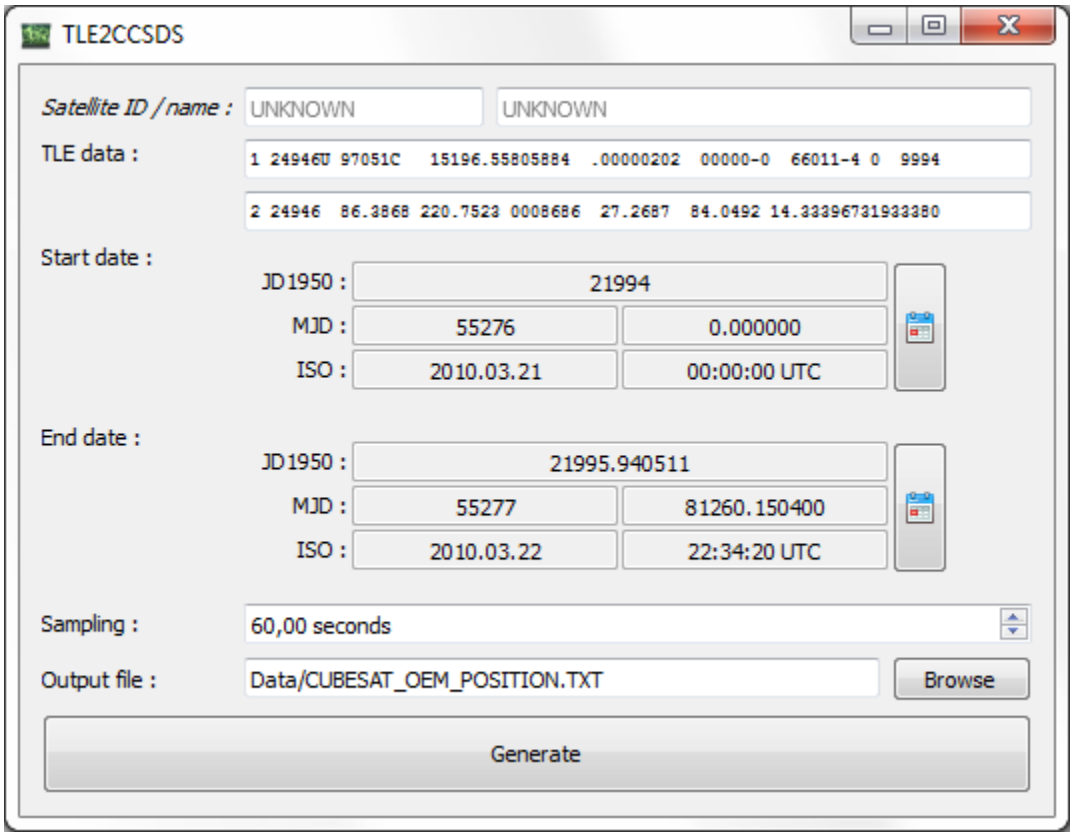


Fig. 7.3: *TLE2CCSDS Generator*

The TLE2CCSDS Generator allows generation of CIC/CCSDS files containing position and velocity ephemerides for a "'two-line element set (TLE)"', in a defined time range. Positions are computed in the EME2000 reference frame. TLE data can be found on http://www.celestrak.com/.

7.3.1 Using the TLE2CCSDS Generator

To generate an orbit ephemerides file, fill the TLE data fields with the orbital elements. A simple way to fill the TLE data fields is to copy the two data line and pasting them into one of the two TLE data fields.

The start and end dates for the CIC/CCSDS file must also be defined in the *Date configuration* area, as well as the time interval between two position values (in seconds).

Select an output file and click the **Generate** button to generate the orbit data. The output data file can then be used to define the position of a satellite in the VTS configuration utility.

CHAPTER

EIGHT

ABOUT

8.1 License

The full text of the VTS Software License can be found in the root directory of the VTS Package (see files LICENSE-FR.txt and LICENSE-EN.txt).

```
VTS SOFTWARE License
====================

Please read attentively the provisions of this Licence before downloading the
SOFTWARE. The use of the SOFTWARE by the Licensee means the latter has agreed
with the provisions of this Licence.

The SOFTWARE as described thereof, property of the CNES and named "VTS"
(Visualisation Tool for Space data) has been registered at the "Agence pour la
Protection des Programmes" (119 rue de Flandres, 75019 Paris) on 3rd April 2012
under the number 350009.

CNES grants a non-exclusive free Licence on the SOFTWARE to the Licensee,
who can be a physical or legal person.

LANGUAGE OF THE LICENCE AND APPLICABLE LAW
-----------------------------------------

The Licence is established in French and English. In the case of a dispute,
the French version will prevail.

This Licence is governed by French law. If any dispute should arise,
litigation shall be brought before the applicable courts.

DURATION OF THE LICENCE
-----------------------

The Licence shall last for ten (10) years renewable by tacit recondition.

In any case, this License will not have a total length greater than the legal
protection of copyright.

TERRITORY
---------

The present Licence is granted worldwide. However, due to the sensitive nature
of products for the space industry, the Licensee agrees imperatively to strictly
respect French legislation regarding export licences.
```

```
RIGHTS CONCEDED
---------------

On the SOFTWARE, furnished in executable code with its documentation, CNES
grants to the Licensee a non exclusive right to use the SOFTWARE free of charge.
This non exclusive right to use the SOFTWARE is defined as the execution of the
program to obtain the requested operation.

CNES grants to the Licensee a non exclusive right to grant sub licences free of
charge.

The Licensee is not authorized to make copies of the SOFTWARE, other than a
backup if it is necessary to preserve the use of the SOFTWARE.

LICENSEE'S OBLIGATIONS
----------------------

The rights thereof conceded are submitted to several conditions:

* The Licensee promises to abstain from any commercial use of the SOFTWARE as
  such. However he is authorized to supply services including the SOFTWARE.

* The Licensee has no right to adapt, to transform, to modify or to translate
  the SOFTWARE.

* On each copy of the SOFTWARE must be maintained the mention of the Licence and
  the copyright CNES. In case of publications, communications or presentations,
  the Licensee promises to mention CNES (Centre National d'Etudes Spatiales).

* The Licensee must not use the SOFTWARE in a detrimental way for the image of
  the CNES, especially to its scientific and technical reputation.

* The Licensee promises to enforce all provisions of this License to its
  sub-licensees.

If the Licensee do not respect all these conditions, he would immediately become
a counterfeiter and could be sued for counterfeit.

GUARANTEE
---------

The Licensee uses the SOFTWARE as it is provided herein at his own expense and
risks, without warranty of any kind from CNES.

CNES can not guarantee the SOFTWARE is exempt from bugs, errors, defects or
omissions and has no obligation of fixing anomalies of any kind nor any defects
of the SOFTWARE. CNES can not guarantee that the SOFTWARE meet the Licensee's
needs.

In the same way, CNES is not responsible for any loss of profit, revenue, data
or any direct or indirect damage, resulting from the use of the SOFTWARE or the
inability to use the SOFTWARE. The Licensee accepts CNES can not be sued about
this.

RESILIATION
-----------
```

```
If the licensee doesn't respect agreement's provisions, without prejudice to all
damages on CNES's behalf, this license ends immediately. In that case, the
Licensee will send back the SOFTWARE and all its copies to CNES and the Licensee
will not be able to ask for any kind of compensation.
```

8.2 Qt

VTS is based on Qt 5.12 distributed under the GNU Lesser General Public Licence version 3. The text of the license can be found below and also in the root directory of the VTS package (see file 3rd-party-licences.txt).

8.2.1 Qt source code & re-linking mechansim

To request a copy of the Qt source code, please send us an mail via the following link:

Qt source code request

In order to link VTS to your own version of interface-compatible Qt libraries, simply copy your Qt compiled libraires to the "Apps/Libs" directory in the VTS package.

8.2.2 GNU Lesser General Public Licence version 3

```
                   GNU LESSER GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

  This version of the GNU Lesser General Public License incorporates
the terms and conditions of version 3 of the GNU General Public
License, supplemented by the additional permissions listed below.

  0. Additional Definitions.

  As used herein, "this License" refers to version 3 of the GNU Lesser
General Public License, and the "GNU GPL" refers to version 3 of the GNU
General Public License.

  "The Library" refers to a covered work governed by this License,
other than an Application or a Combined Work as defined below.

  An "Application" is any work that makes use of an interface provided
by the Library, but which is not otherwise based on the Library.
Defining a subclass of a class defined by the Library is deemed a mode
of using an interface provided by the Library.

  A "Combined Work" is a work produced by combining or linking an
Application with the Library.  The particular version of the Library
with which the Combined Work was made is also called the "Linked
Version".

  The "Minimal Corresponding Source" for a Combined Work means the
Corresponding Source for the Combined Work, excluding any source code
```

```
for portions of the Combined Work that, considered in isolation, are
based on the Application, and not on the Linked Version.

  The "Corresponding Application Code" for a Combined Work means the
object code and/or source code for the Application, including any data
and utility programs needed for reproducing the Combined Work from the
Application, but excluding the System Libraries of the Combined Work.

  1. Exception to Section 3 of the GNU GPL.

  You may convey a covered work under sections 3 and 4 of this License
without being bound by section 3 of the GNU GPL.

  2. Conveying Modified Versions.

  If you modify a copy of the Library, and, in your modifications, a
facility refers to a function or data to be supplied by an Application
that uses the facility (other than as an argument passed when the
facility is invoked), then you may convey a copy of the modified
version:

   a) under this License, provided that you make a good faith effort to
   ensure that, in the event an Application does not supply the
   function or data, the facility still operates, and performs
   whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of
   this License applicable to that copy.

  3. Object Code Incorporating Material from Library Header Files.

  The object code form of an Application may incorporate material from
a header file that is part of the Library.  You may convey such object
code under terms of your choice, provided that, if the incorporated
material is not limited to numerical parameters, data structure
layouts and accessors, or small macros, inline functions and templates
(ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the
   Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license
   document.

  4. Combined Works.

  You may convey a Combined Work under terms of your choice that,
taken together, effectively do not restrict modification of the
portions of the Library contained in the Combined Work and reverse
engineering for debugging such modifications, if you also do each of
the following:

   a) Give prominent notice with each copy of the Combined Work that
   the Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the Combined Work with a copy of the GNU GPL and this license
```

```
   document.

   c) For a Combined Work that displays copyright notices during
   execution, include the copyright notice for the Library among
   these notices, as well as a reference directing the user to the
   copies of the GNU GPL and this license document.

   d) Do one of the following:

       0) Convey the Minimal Corresponding Source under the terms of this
       License, and the Corresponding Application Code in a form
       suitable for, and under terms that permit, the user to
       recombine or relink the Application with a modified version of
       the Linked Version to produce a modified Combined Work, in the
       manner specified by section 6 of the GNU GPL for conveying
       Corresponding Source.

       1) Use a suitable shared library mechanism for linking with the
       Library.  A suitable mechanism is one that (a) uses at run time
       a copy of the Library already present on the user's computer
       system, and (b) will operate properly with a modified version
       of the Library that is interface-compatible with the Linked
       Version.

   e) Provide Installation Information, but only if you would otherwise
   be required to provide such information under section 6 of the
   GNU GPL, and only to the extent that such information is
   necessary to install and execute a modified version of the
   Combined Work produced by recombining or relinking the
   Application with a modified version of the Linked Version. (If
   you use option 4d0, the Installation Information must accompany
   the Minimal Corresponding Source and Corresponding Application
   Code. If you use option 4d1, you must provide the Installation
   Information in the manner specified by section 6 of the GNU GPL
   for conveying Corresponding Source.)

  5. Combined Libraries.

  You may place library facilities that are a work based on the
Library side by side in a single library together with other library
facilities that are not Applications and are not covered by this
License, and convey such a combined library under terms of your
choice, if you do both of the following:

   a) Accompany the combined library with a copy of the same work based
   on the Library, uncombined with any other library facilities,
   conveyed under the terms of this License.

   b) Give prominent notice with the combined library that part of it
   is a work based on the Library, and explaining where to find the
   accompanying uncombined form of the same work.

  6. Revised Versions of the GNU Lesser General Public License.

  The Free Software Foundation may publish revised and/or new versions
of the GNU Lesser General Public License from time to time. Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.
```

```
  Each version is given a distinguishing version number. If the
Library as you received it specifies that a certain numbered version
of the GNU Lesser General Public License "or any later version"
applies to it, you have the option of following the terms and
conditions either of that published version or of any later version
published by the Free Software Foundation. If the Library as you
received it does not specify a version number of the GNU Lesser
General Public License, you may choose any version of the GNU Lesser
General Public License ever published by the Free Software Foundation.

  If the Library as you received it specifies that a proxy can decide
whether future versions of the GNU Lesser General Public License shall
apply, that proxy's public statement of acceptance of any version is
permanent authorization for you to choose that version for the
Library.
```

8.3 Programming languages

Several programming languages are used throughout the VTS toolkit:

- **C++/Qt**: VTS main programming language (VTS Configuration utility, VTS Broker, SurfaceView, . . .)
- **C**: CIC/CCSDS library
- **Tcl/Tk**: PrestoPlot, VTS cross-platform toolkit launcher
- **Lua**: VTS plugin for Celestia
- **Java**: Keplerian generator
- **Bash**: Client application launchers, build and deployment scripts
- **DOS Batch**: Client application launchers