

The Stellar Command Module
for
Integrating Astronomy and Music
User Guide
Angelo Fraietta
University of New South Wales

TABLE OF CONTENTS

| | |
|--|-------|
| TABLE OF CONTENTS | ii |
| LIST OF FIGURES | v |
| LIST OF TABLES | vi |
| Preface | vii |
| Introduction | ix |
| Background | xi |
| Stellarium | xi |
| VizieR | xiii |
| Stellar Command Module | xiii |
| Terminology | xv |
| Disciplines of research | xv |
| Astronomy | xv |
| Field of View | xv |
| Observation Location | xv |
| Date time - ISO 8601 | xvii |
| Azimuth | xvii |
| Altitude | xvii |
| Celestial Coordinates | xviii |
| Right Ascension | xviii |
| Declination | xviii |
| Magnitude | xviii |
| Atmosphere | xviii |
| Ethnoastronomy | xviii |
| Archaeoastronomy | xviii |
| Astrology | 1 |
| 1 Using Stellar Command with OSC | 1 |
| 1.1 Launching Stellar Command | 1 |
| 1.2 Sending Commands to the Stellar Command Module | 3 |
| 1.2.1 poll | 3 |

| | | |
|----------|--|-----------|
| 1.2.2 | fieldOfView | 3 |
| 1.2.3 | time | 4 |
| 1.2.4 | moveLR | 4 |
| 1.2.5 | moveUD | 4 |
| 1.2.6 | azimuth | 5 |
| 1.2.7 | altitude | 5 |
| 1.2.8 | viewAltAz | 5 |
| 1.2.9 | viewRADec | 5 |
| 1.2.10 | viewObject | 5 |
| 1.2.11 | viewerObservationPoint | 6 |
| 1.2.12 | showGround | 6 |
| 1.2.13 | showAtmosphere | 6 |
| 1.2.14 | showStarLabels | 6 |
| 1.2.15 | showConstellationart | 6 |
| 1.2.16 | timeRate | 7 |
| 1.2.17 | saveTable and loadTable | 7 |
| 1.2.18 | sendStars | 7 |
| 1.2.19 | filter | 8 |
| 2 | Decoding OSC Messages | 9 |
| 2.1 | osc | 9 |
| 2.2 | view | 9 |
| 2.3 | Star Values | 10 |
| 2.3.1 | bundleCount | 10 |
| 2.3.2 | names | 10 |
| 2.3.3 | values | 11 |
| 2.4 | viewerObservationPoint | 11 |
| 2.5 | time | 12 |
| 3 | Using Stellar Command as a Java Library | 13 |
| 3.1 | stellarium | 13 |
| 3.1.1 | StellariumSlave | 13 |
| 3.1.2 | StellariumView | 14 |
| 3.1.3 | StellariumLocation | 14 |
| 3.1.4 | StellariumTime | 14 |
| 3.1.5 | StellariumProperty | 14 |
| 3.2 | vizier | 14 |
| 3.2.1 | VizierQuery | 14 |
| 3.2.2 | StellarDataTable | 15 |
| 3.2.3 | StellarDataRow | 15 |
| 3.2.4 | StellarFilteredTable | 15 |
| 3.2.5 | FilteredData | 15 |
| 3.2.6 | MagnitudeSort | 15 |
| 3.3 | stellarstructures | 15 |
| 3.3.1 | AltAz | 15 |
| 3.3.2 | RaDec | 15 |
| 3.3.3 | ObservationalPoint | 15 |

LIST OF FIGURES

| | | |
|-----|---|------|
| 1 | Constellations displayed in Western sky lore. | xii |
| 2 | Constellations displayed in Tukano sky lore. | xii |
| 3 | stellarium displaying a 60° field of view. | xvi |
| 4 | stellarium displaying a 13° field of view. | xvi |
| 5 | A simulation of Jupiter being viewed with Io as the observation location. | xvii |
| 6 | Midday showing the sky with atmosphere. | 1 |
| 7 | Midday showing the sky with atmosphere. | 2 |
| 8 | Constellation art. | 2 |
| 1.1 | Remote stellarium and OSC Clients. | 2 |

LIST OF TABLES

| | | |
|-----|---|----|
| 2.1 | Sample correlated column names and data | 11 |
|-----|---|----|

Preface

Stellar Command is a software system that integrates *stellarium* planetarium software with online astronomical data acquisition through *VizieR* database of astronomical catalogues [?]. Stellar Command can be used as an interface mechanism for correlating music and sound generation, allowing stellarium to be used as both a direct input interface for performance or composition, and as a remotely controlled derivative multimedia display output.

I hope you will have as much fun creating musical works with it as much as I have.

ANGELO FRAIETTA
Newcastle, AU
June 2019

Introduction

Musical composition and performance inspired or based on astronomy has been used in many cultures for millennia, with many civilizations creating songs and dances based on the astronomical calendar to reinforce the tracking of seasonal activities; such as planting and harvesting of crops, times of trade, and religious or cultural practices [?, ?, ?]. More recently, composers have used scientific data obtained from individual stars to generate sounds and have created compositions directly correlated to that data [?, ?].

The advancement of computing power has made the availability of planetarium software for both desktop and mobile platforms very accessible to many people. These software packages are not only used for scientific and research activities; such as astronomy, education and general stargazing; they have also been used by artists in presenting multi-media artworks and installations [?, ?].

Many composers have used the cosmos as inspiration or stimulus to their works, with many using scientific observations or data as input [?, ?]. The *Quadrivium* linked astronomy, mathematics, geometry and music as a standard part of classical education up until the renaissance [?]. Composers have been mapping mathematics and geometry to music since antiquity [?, ?]. Kepler stated "The heavenly motions are nothing but a continuous song for several voices, to be perceived by the intellect, not by the ear; a music which, through discordant tensions, through syncopations and cadenzas as it were, progresses toward certain pre designed six-voiced cadences, and thereby sets landmarks in the immeasurable flow of time." [?, cited in 286]. This notion inspired Rogers and Ruff to compose *The Harmony of the World* (1979), describing their work as "A Realization for the Ear" [?, p. 286] of Johannes Kepler's Astronomical Data from Harmonices Mundi 1619. More recently, composers have used measurements from online databases as inputs to automata or as stimulus to performers.

I originally developed a system based on astronomical catalogues for musical composition and performance interface using naked eye and binocular astronomy [?]. A specific star was determined by calculating its azimuth and height above the horizon-known as its *altitude*-using accelerometer and magnetometer sensors, and calculating the exact location of the star on the celestial sphere using the sensor data, time and geographical location of the observer. This calculation returns the star's *right ascension (RA)*, which is based on its azimuth at Greenwich Meantime at the vernal equinox, and its *declination (Dec.)*, which is the stars north-south position at the same time [?, ?]. The resultant RA and Dec. are added as input to the VizieR database of online catalogues, returning data about stars-such as brightness and colour-within the defined radius. Various works were created using this interface. In one performance, which was conducted in conjunction with the Newcastle Astronomical Society on one of their field viewing nights, members of the pub-

lic were enticed into viewing the night sky through high powered binoculars, while the sound generated, which was based on data from the stars they were viewing, was played through loudspeakers on the field [?]. Another set of performances was conducted with an improvising ensemble that featured various astronomical photos displayed as a slide show where the astronomical data was mapped as MIDI and functioned as inspirational impetus for the performers [?].

Although the binocular display has an awesome display—the actual night sky—“few people ventured outside to the astronomical equipment”[?, p. 50] because they were required to leave the room to look through binoculars while the ensemble played in a room. Furthermore, the work is severely bound by weather conditions and a clear view of the sky. In one of the performances, “the sky was completely covered with cloud and it rained, so there was nothing to see through the binoculars. The audience, however, enjoyed the ensemble performance with the NASA image slide show with samples fed from stored star tables.”[?, p. 50]. This weather constraint inspired me to use planetarium software as the input and display mechanism as an alternative to binoculars.

Stellar Command enables composers to access astronomical data as input to their software using a common API. It also enables you to provide the audience an impressive planetarium software display that runs on a laptop computer without requiring the audience to leave the room. Furthermore, it facilitates creation of interactive celestial based installations. Stellar Command is available as open source software through GitHub [?].

Background

Stellarium

Stellarium is a software program designed to enable people to create a virtual planetarium using their home computer [?]. It runs on Windows, OSX and Linux/Unix, including Raspberry Pi [?, p. 6]. stellarium calculates positions of the Sun, moon, stars and planets based on the time and location defined by the user, and renders them to the display. stellarium is used by both amateur and professional astronomers, and is used by the European Organisation for Astronomical Research in the Southern Hemisphere to facilitate distribution and sharing of visual data among scientists [?]. stellarium has a very high quality graphical display, supporting spherical mirror projection that can be used with a dome [?] and is used in many schools and museums because it is both scientifically accurate and visually engaging [?]. Moreover, stellarium can display constellations from several different cultures and has labels translated to more than 40 languages, making stellarium both culturally aware and inclusive [?]. For example, Figures 1 and 2 display the same area of sky, however the first presents the constellations using the western sky lore while the latter presents them in the Tukano sky lore¹ [?]. A comparison reveals that *Scorpius* and *Crux* are referred to as the *Fer-de-lance* and the *Tortoise*. This feature can make stellarium an extremely useful tool in facilitating multimedia presentations for ethnomusicology or composing in non-western contexts.

Many astronomers use stellarium to display a prediction of the sky for a future time, such as organising a viewing night or planning an astro-photography session [?]. Archaeoastronomers also use the feature to generate an astronomical display from a location for a period sometime in the past [?]. Also, the landscape feature facilitates a connection between landscape and skyscape, so one can map the sky against a landscape for research or aesthetics [?].

stellarium scripts are written in ECMAScript, also known as Javascript, and enables the programmer to generate and run an automated astronomy presentation, facilitating automation of all the functionality of stellarium [?].

stellarium has a Remote Control plugin that enables third party programs to communicate with stellarium via a REST client. This feature was used by the author to create an interactive spacecraft game using a sonic ball as a preliminary test of stellarium's viability as a responsive performance interface [?]. The plugin also allows clients to query the status of stellarium, including the area of sky currently being displayed. The exact celestial

¹The Tukano tribes are indigenous peoples of the northwestern region of Brazil [?].

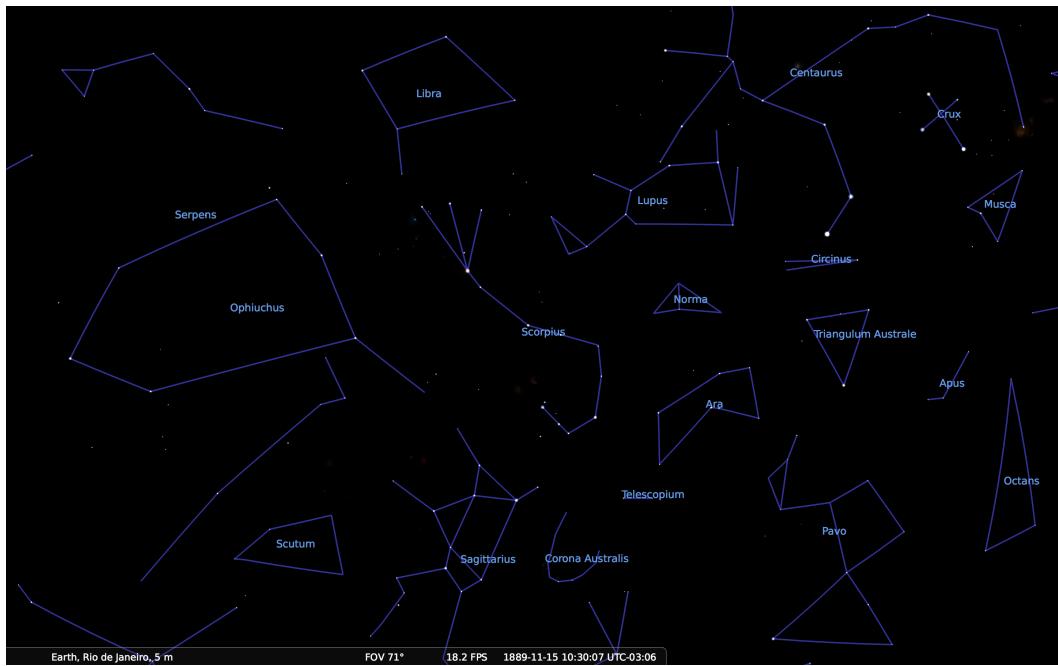


Figure 1: Constellations displayed in Western sky lore.

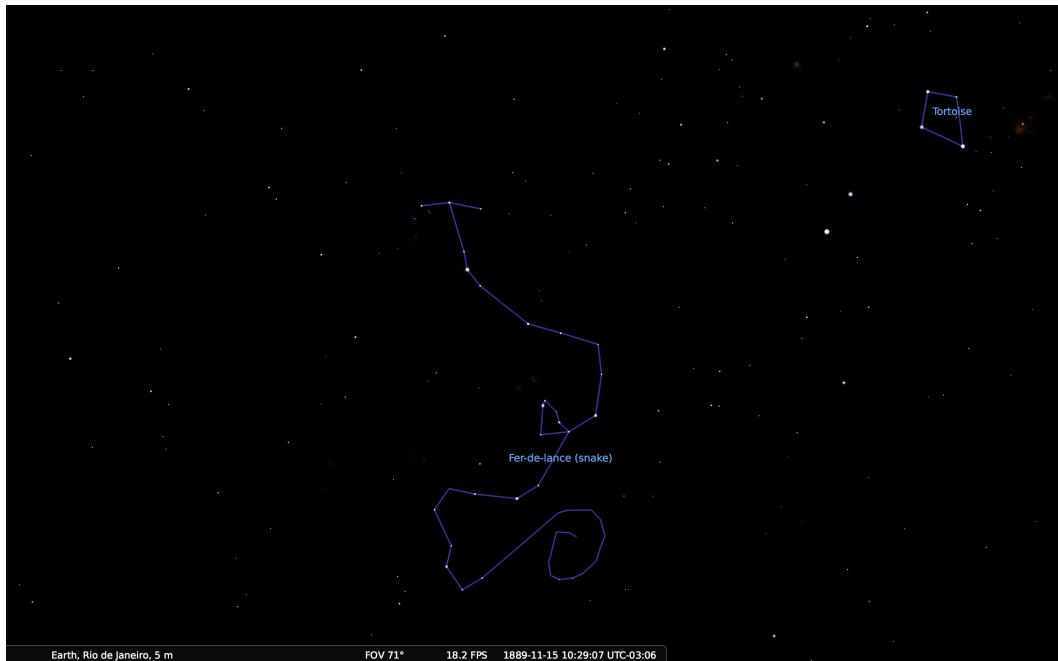


Figure 2: Constellations displayed in Tukano sky lore.

vector and field of view displayed are used as the input to the VizieR server in order to obtain data about stars in that location.

VizieR

VizieR is an online database of high quality astronomical catalogues collected by the Centre de Données astronomiques de Strasbourg (CDS), one of whose main goals "is to promote the usage of the reliable astronomical catalogues to the astronomical community" [?, p. 25]. One of the ways CDS ensures the reliability of their archives and catalogues is to only collect data that has been published or accepted in refereed scientific journals or literature. The number of catalogues available has grown from 3000 in 1999 [?, p. 24] to currently 18000 [?]. Catalogues can be selected by name or based on the wavelength, mission and astronomy type [?]. Wavelength dictates the frequency spectrum of the data in the catalogue, such as radio, infra-red, optical, x-ray, and Gamma-RAY. The mission indicates the purpose for which the catalogue was created or is used. For example, the purpose of the Kepler Mission is to "detect Earth-size planets in the habitable zone ... of solar-like stars..., determine their frequency, and identify their characteristics." [?, p. 2]; and so catalogues created or used within this mission can be specifically targeted. The type of astronomy indicates the subject area of research that the catalogue belongs to; for example, black holes, galaxy clusters, planetary nebulae, red shifts, and photometry.

Once a user has selected the database criteria to search, they must provide VizieR a search window that consists of a target centre and area around that point to search. The target centre is the point on the celestial sphere referenced to a celestial equinox², and can be defined by object name, RA and Dec., or by IAU-coordinates [?].

Stellar Command Module

Although it is possible to communicate directly to stellarium and VizieR directly through the HTTP interface, the stellar position parameters between the two systems are different. stellarium returns its position information as three dimensional spherical points, with a separate query for the field of view; whereas, VizieR requires the data as a two dimensional geometric point with a defined radius. The Stellar Command module abstracts this information from the client software and removes the requirement to perform these calculations by the client. Stellar Command directs VizieR to query the Hipparcos catalogue because it provides a "complete all-sky survey of astrometric and photometric parameters for one million stars down to magnitude 11" [?, p. 201]. An attempt was made using other photometric catalogues, however, it was difficult to obtain consistent results for all sky locations.

The interface required to control both stellarium and VizieR was HTTP based, so a language that provided network functionality as a fundamental core feature was preferred. Also, it was imperative that third party users of the library should not have to learn a particular programming language, but instead, could easily interface with the library using their preferred music package such as Max MSP, SuperCollider or PD. Furthermore, it would be advantageous for users to be able to integrate the library into the programming language of their choice, such as C, C++, Python, Ruby, or Java.

²The default equinox is J2000 [?].

Developing the system as a Java Archive (JAR) fulfilled all these requirements. Using a JAR file allowed instantiation from the command line, running as separate processes and using OSC to communicate with other programs. Additionally, many other programming languages can connect directly to a JAR through the Java Native Interface (JNI) [?].

Terminology

In all areas of research or vocation, there is terminology used that is specific to that field, and astronomy is no exception. Although this is not a treatise on the study of astronomy, it would serve you well to become acquainted with some of these terms.

Disciplines of research

You will hear terms such as *astronomy*, *archaeoastronomy* and *ethnoastronomy*. These terms can sometime become confusing so I will provide short definitions for them.

Astronomy

Astronomy is one of physical sciences and involves the systematic study of celestial objects and phenomena that originate outside the Earth's atmosphere. Astronomic data from various missions are recorded in catalogues for use within the scientific community. Within the field of study, there are various terms used, which are important to know in order to query and interpret the astronomical data you will encounter.

Field of View

The *field of view* is the amount of sky that you are able to see and it is measured in degrees. If we zoom in, we reduce the field of view. If we compare Figures 3 and 4, we see the same point of sky is in the centre, however, the image that looks closer has a lower field of view.

Observation Location

The observer location is the geographical position from where the celestial observation is being made. These are defined using latitude and longitude. For example, if you were viewing the sky from Newcastle Australia, your geographical location would be 32.9283° South, 151.7817° East. Rather than using the terms south and east, northern latitudes are positive and southern are negative. Likewise, east is positive and west is negative. Therefore, the geographical location for Newcastle would be -32.9283, +151.7817. A western longitude can also be positive, however, the value is 360° - the western longitude. Eg, a longitude of -90° is also +270°.

It is also possible to set the planet or celestial body you are viewing from. For example, Figure 5 displays us looking at Jupiter with our observation location as its moon Io.

Another parameter is the altitude from where the observation is being made.

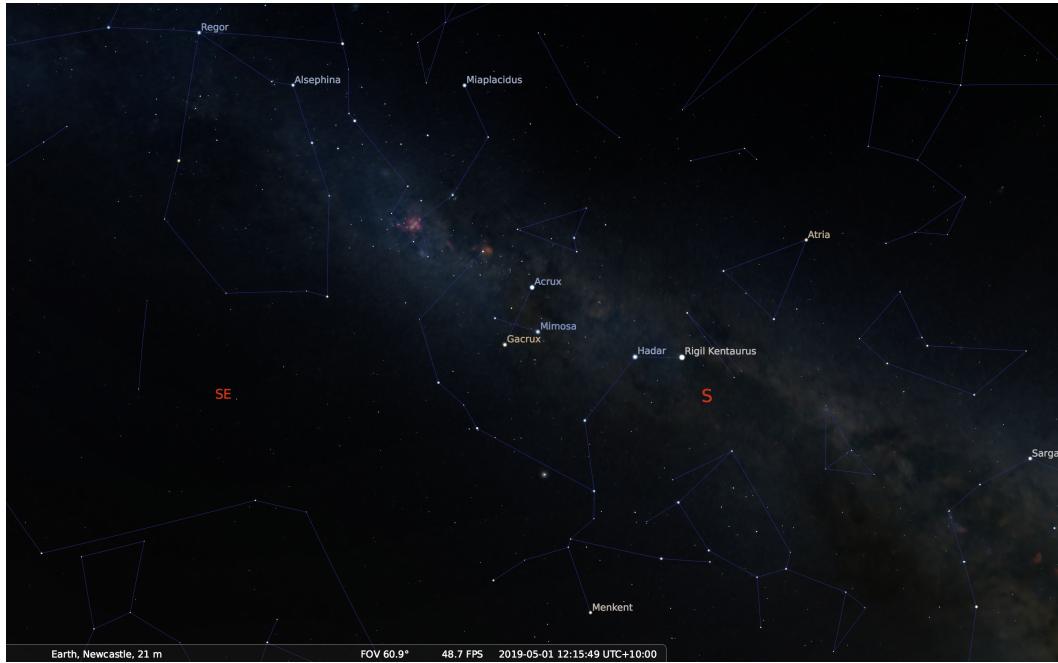


Figure 3: stellarium displaying a 60° field of view.

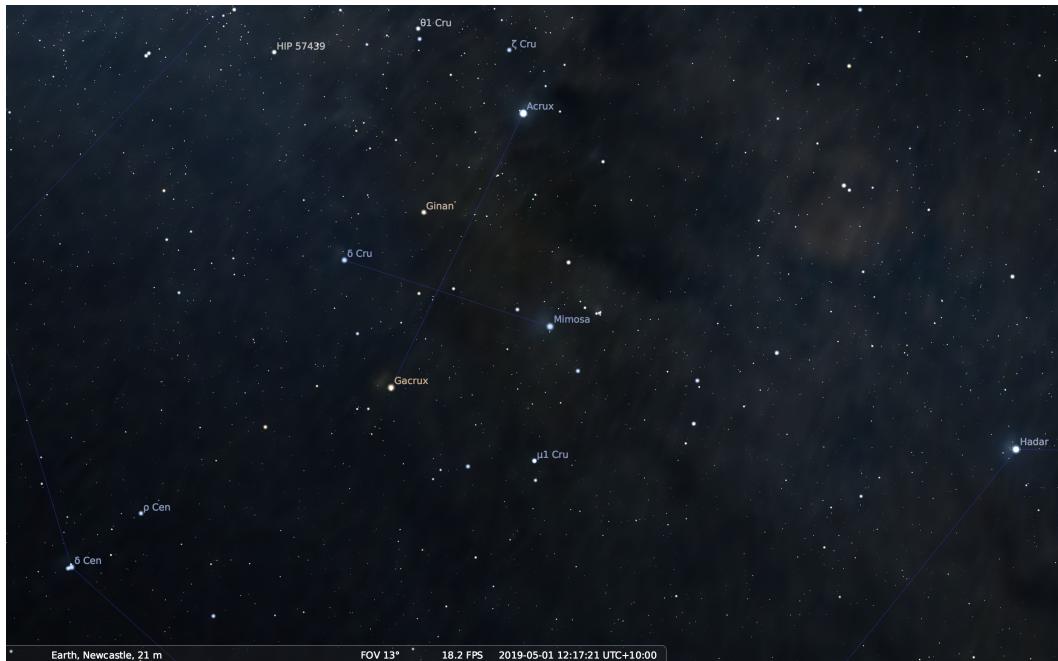


Figure 4: stellarium displaying a 13° field of view.



Figure 5: A simulation of Jupiter being viewed with Io as the observation location.

Date time - ISO 8601

The ISO 8601 date and time format is a standardised way of defining the exact date and time. The time zone is the time difference between local time and Coordinated Universal Time (UTC) and does not change with daylight saving. The exact time is specified by the year, month, day followed by time in hours (24 hour code), minutes, seconds and the time zone. The time zone for UTC is "Z", meaning zero. The time zone is modified by changing the "Z" to hours and minutes, separated by a semicolon. For example, midday on January 1, 2000 at UTC would be defined as "2000-01-01T12:00:00Z". Likewise, "2019-06-01T14:13:00+10:00" would be 1 June 2019 at 2:15pm in Sydney, which is 10 hours ahead of or earlier than UTC time.

Azimuth

Azimuth is the direction with respect to celestial north and is measured in degrees. If, for example, you were looking due south, the azimuth would be 180. East is 90 and west is -90 or +270.

Altitude

The altitude is the angle you are looking up or down at and is measured in degrees. If you were looking directly up, the altitude would be 90 degrees. Similarly, horizontal is zero and down is -90.

Celestial Coordinates

In the same way we can define an exact location on the Earth using latitude and longitude, we define the location of celestial objects using the celestial coordinate system. Instead of longitude and latitude, we use the terms *Right ascension* and *Declination*.

Right Ascension

Right Ascension, abbreviated to *RA*, is the angular distance from the east of the First Point of Aries (hereafter FPOA), and is measured in hours, minutes and seconds, and is similar to the longitude on earth, except on the celestial sphere. This is basically how long it takes from the time FPOA is on the meridian (directly north or south) for the RA in question to be overhead. For example, if an object had a Right ascension of 3 Hours, it would mean that it would appear on the meridian three hours after FPOA was. This is similar to the concept of time zones. In order to make calculations easier, we convert the RA to a digital angle by multiplying the number of hours by 15 (24 hours x 15 = 360°). Furthermore, we use decimals instead of minutes and seconds. eg, 90° 30' would be written 90.5°.

Declination

The declination is equivalent to the concept of latitude and is the angular distance a star is north or south of the celestial equator when it is on the meridian (in line with north / south).

Magnitude

The magnitude is how bright a star is. Higher magnitudes mean a lower brightness.

Atmosphere

Atmosphere is the gas that surrounds us and makes the sky appear like daylight. If there was no atmosphere, the sky would appear dark, as it does on the moon, and you would be easily able to see stars during the day. Stellarium gives us the option to hide the atmosphere, enabling us to see stars at any time. For example, Figure 6 shows the midday with normal atmosphere, whereas 7 shows the same sky without the atmosphere.

Ethnoastronomy

Ethnoastronomy is a social science belonging to the discipline of ethnology and examines the way astronomy is practised in the context of a particular culture [?].

Archaeoastronomy

Archaeoastronomy is the study of how ancient civilisations understood the cosmos and in a sense very similar to ethnoastronomy. "Broad public embrace of archaic astronomy probably began in the eighteenth century with awareness of the summer solstice sunrise's affiliation with Stonehenge" [?, p. 263]. Archaeoastronomers have used stellarium to generate an astronomical display from a location for a period sometime in the past [?].



Figure 6: Midday showing the sky with atmosphere.

Astrology

"Astrology, from the Greek, astro-logos, is the assumption that the stars and planets contain meaning and significance for terrestrial affairs....Astrology appears to be a universal feature of human culture and may be understood as a form of cultural astronomy; an important contribution to the understanding of astronomy's cultural uses, applications, uses, and functions; and an indication of society's attitudes to the stars." [?, p. 104]. stellarium has a feature whereby you can display constellation art, as shown in 8, which facilitates representing astrology in your works.



Figure 7: Midday showing the sky with atmosphere.



Figure 8: Constellation art.

Chapter 1

Using Stellar Command with OSC

There are primarily two ways to use Stellar Command. The first is as a separate process that runs on the compute. The second method is to use Stellar Command as a library that you link to directly in your program. If you intend to use Stellar Command as a library, you can skip forward to chapter 3 – [Using Stellar Command as a Java Library](#).

1.1 Launching Stellar Command

The Stellar Command module is instantiated by executing Java with the name of the JAR file and the required program arguments that define communication, such as the network port to send OSC messages to, and the OSC address space. For example, to start the StellarCommand module so it sends OSC messages on UDP port 1234 using an OSC address space of /Stellar,¹ one would execute the following command:

```
java -jar StellarCommand.jar port=1234 osc=/Stellar
```

When the server starts, it will open the first available UDP port, and notify the client of this port. For example, if the command module opened port 4567, it will send an OSC message /Stellar/osc 4567 to the client on the localhost.

```
/Stellar/osc 4567
```

Allowing the command module to find its own port number removes the probability of port clashes as each client furnishes the other with a valid port number for communicating without requiring configuration in the command module. It is, however, possible to request the Stellar Command module try certain ports by adding the argument *tryport* with a comma separated list of ports. For example, the argument *tryport=3333,4444,5555*

¹In this instance, the OSC client and stellarium are on the same computer.

will cause stellarium to sequentially try opening the ports listed, and if these all fail, will then open the first available port.

```
java -jar StellarCommand.jar port=1234 osc=/Stellar tryport=3333,4444,5555
```

The OSC client would receive the following OSC message:

```
/Stellar/osc 3333
```

The OSC client and the stellarium server do not have to be on the same physical computer as the Stellar Command module. For example Figure 1.1, shows three OSC clients and a stellarium server on a LAN, and a remote stellarium server accessible from the internet through *myserver.com*.

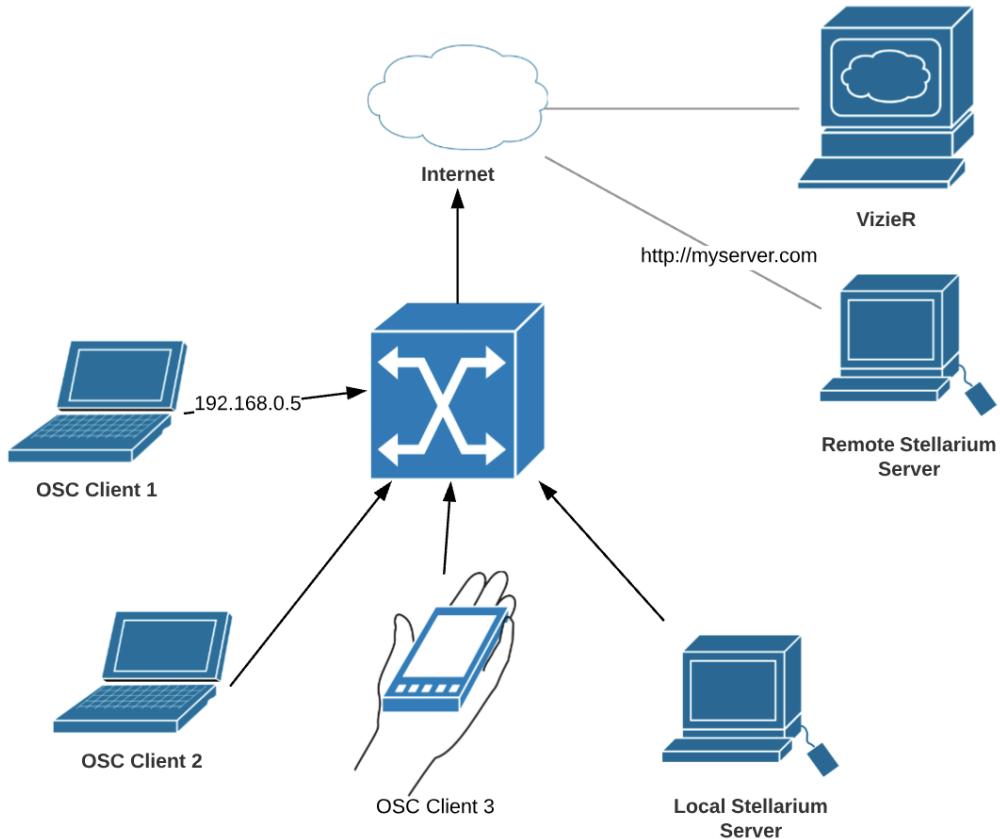


Figure 1.1: Remote stellarium and OSC Clients.

Creating a connection between *OSC Client 1* and *Remote stellarium Server* is effected by adding adding the arguments

`client=192.168.0.5` and `stellarium=http://myserver.com` to the command line. This will cause the Stellar Command module to send OSC messages to "192.168.0.5" and stellarium commands to `http://myserver.com` on HTP port 8090², effectively acting as a proxy between the two.

```
java -jar StellarCommand.jar port=1234 osc=/Stellar \
client=192.168.0.5 stellarium=http://myserver.com
```

1.2 Sending Commands to the Stellar Command Module

You can provide instructions to Stellar Command in order to control stellarium or to change the amount and type of data you want to receive. For example, you may want the stellarium display to zoom in closer to a particular area of sky. You would do this by decreasing the field of view.³ Likewise, you may want to reduce the amount of astronomical data you are receiving by adding a filter threshold so VizieR will only receive stars within a certain magnitude range.

Message names are not case sensitive. For example, the messages `fieldOfView` and `FieldOfView` could both be used to set the field of view for stellarium.

1.2.1 poll

Sometimes Stellar Command may already be started and configured to send and listen to the ports. You can easily find this out by sending the `poll` command with no arguments, which will cause stellarium to reply on the ports it is configured when started up.

```
Stellar/poll
```

If you were sending OSC on the required port and your cleint was at the correct address, you would receive the following OSC message:

```
/Stellar/osc 3333
```

1.2.2 fieldOfView

In order to send a change in the field of view, send a message using the `fieldOfView` in the namespace, followed by the field of view as a float argument. For example, to set the field of view to 60°, you would send a message as follows:

²The default stellarium Remote Control port is 8090, however, this can be changed inside stellarium. It is assumed that port forwarding when not using a local area network has been configured to send packages to the correct computer hosting stellarium

³Section 1.2.2 – `fieldOfView` shows how to do this.

```
/Stellar/fieldOfView 60.0
```

1.2.3 time

You can set the simulation time by sending a full ISO time stamp as a string message, or by sending each Date time parameter as OSC arguments. For example, you can set the time as a string as follows:

```
/Stellar/time 2019-04-18T17:51:17Z
```

If using specified time, the first three arguments will be Year, Month and Day as integers. The next argument will be the hour as an integer as a 24 hour number. EG, the number 13 will signify 1pm. The next two arguments will be the minutes and seconds as integers. The final optional argument will be the time zone as an alphabetical times zone, or if blank, the local time at the location will be used. the same time could be set by sending the following message.

```
/Stellar/time 2019 4 18 17 51 17 Z
```

Sending the following would set to midday on 18 April 2019 at the Observer location in stellarium.

```
/Stellar/time 2019 4 18 12 0 0
```

1.2.4 moveLR

You can make the stellarium display move left or right by sending the command *moveLR* followed by a floating point number indicating how far to move. A positive value will make it appear that the viewer is turning their head to the right, while a negative number will simulate moving to the left. To move to the right, send the following command:

```
/Stellar/moveLR 1.0
```

1.2.5 moveUD

You can make the stellarium display move up or rdown by sending the command *moveUD* followed by a floating point number indicating how far to move. A positive value will make it appear that the viewer is turning their head up, while a negative number will look lower. To look towards the sky, send the following command:

```
/Stellar/moveUD 1.0
```

1.2.6 azimuth

If you want to change the azimuth that you are looking at, use *azimuth* in the namespace and add the azimuth as a float argument. For example, if you wanted to turn to the east, you would send a message as follows:

```
/Stellar/azimuth 90.0
```

1.2.7 altitude

If you want to change the altitude that you are looking at, use *altitude* in the namespace and add the altitude as a float argument. For example, if you wanted to look 45° up, you would send a message as follows:

```
/Stellar/altitude 45.0
```

1.2.8 viewAltAz

Setting the altitude and the azimuth at the same time is so commonplace that it has been made available as a single OSC message. If you want to change the altitude that you are looking at, use *viewAltAz* in the namespace and add the altitude and then the azimuth as float arguments. The two previous calls could have been done with the single command

```
/Stellar/viewAltAz 45.0 90.0
```

1.2.9 viewRADec

You can move to the RA and Dec to the centre of the sky by using *viewRADec* in the namespace and send the RA and Dec as decimal degrees. If for example, you wanted to centralise an RA of 95.98796° and Dec. of -52.69566° (this is the star Canopus), you would send the following OSC:

```
/Stellar/viewRADec 95.98796 -52.69566
```

1.2.10 viewObject

You may want to select a particular named object in the sky, such as a star, planet or moon. You can do this by using the command *viewObject* followed by the object name. For example, if you wanted to centre the planet Saturn, you would send the following OSC:

```
/Stellar/viewObject Saturn
```

1.2.11 viewerObservationPoint

You can change the latitude, longitude, altitude and planet you are setting as your observation point by using the *viewerObservationPoint* command followed by the latitude, longitude, and altitude as float arguments, and the planet as a string. For example, to set a latitude of 32°, longitude of 151°, and altitude of 21m from Saturn, you would send the following command:

```
/Stellar/viewerObservationPoint 32.0 151.0 21.0 Saturn
```

1.2.12 showGround

It is possible to hide the ground, enabling you to see stars and planets through the earth by sending the *showGround* command, and an integer value of zero to hide the ground or non-zero to show it. To hide the ground, you would send the following OSC message:

```
/Stellar/showGround 0
```

1.2.13 showAtmosphere

It is possible to hide the atmosphere, enabling you to see stars and planets during daylight hours by sending the *showAtmosphere* command, and an integer value of zero to hide the atmosphere or non-zero to show it. To hide the atmosphere, you would send the following OSC message:

```
/Stellar/showAtmosphere 0
```

1.2.14 showStarLabels

It is possible to hide the star labels by sending the *showStarLabels* command, and an integer value of zero to hide the labels or non-zero to show it. To hide the labels, you would send the following OSC message:

```
/Stellar/showStarLabels 0
```

1.2.15 showConstellationart

It is possible to show or hide the constellation art by sending the *showConstellationart* command, and an integer value of zero to hide the art or non-zero to show it. To show the constellation art, you would send the following OSC message:

```
/Stellar/showConstellationart 1
```

1.2.16 timeRate

It is possible to change the simulated time rate on stellarium by sending the *timeRate* command followed by the new time rate in days per second as a float. If, for example, you set the time rate to zero, time would be still and the stars would not move through the sky. If, however, you set the time rate to 0.5, the display would simulate time running at one day every two seconds. You can also make time go in reverse by setting the time rate to a negative number. To set the display to simulate a day every two seconds, you would send the following:

```
/Stellar/timeRate 0.5
```

1.2.17 saveTable and loadTable

Retrieving astronomical data from VizieR can sometimes take time, depending on your internet speed and the number of stars in the field of view requested. It is possible to save the current loaded VizieR table of stars to a file so you can use it even when there is no internet. To save the table, send the command *saveTable* followed by the name of the file you want it saved to. You can then use the *loadTable* command later to load the data and have Stellar Command send it to you. To save the current table to the file "Acrux.txt", send the following command:

```
/Stellar/saveTable Acrux.txt
```

To load this table from file and have it send the data to you, send the following command:

```
/Stellar/loadTable Acrux.txt
```

1.2.18 sendStars

It is possible to pause Stellar Command sending stars data back to the client. This may be particularly useful if you are receiving data from thousands of stars and the resources to parse decode all that data in OSC may be unnecessary. Sending the *sendStars* command with a zero will cause Stellar Command to stop querying VizieR, and thus stop sending you updated astronomical data. You will still, however, receive position and field of view change messages. Sending a non-zero argument with the command will resume sending starts. To stop the data being sent, send OSC as follows:

```
/Stellar/sendStars 0
```

1.2.19 filter

Considering how many stars could be within a particular view, it can be beneficial to apply a filter. For example, you may only be interested in stars with a magnitude brighter than 6. Setting a filter actually modifies the query sent to VizieR, thus reducing the amount of time to process the astronomical data. Filtering the data is accomplished by adding the command *filter*, followed by the astronomical data field you want to filter, followed by less or greater and then using the OSC argument as the filter value. for example, to filter the *Hpmag* field so magnitudes brighter (less than) 6.5 are returned, you would send the following OSC command

```
/Stellar/filter/Hpmag/less 6.5
```

You can AND filters together by sending a second command. for example, to filter stars that are between magnitudes 3 and 6, you would send the following OSC.

```
/Stellar/filter/Hpmag/less 6  
/Stellar/filter/Hpmag/greater 3
```

You can filter by star colour (i.e. temperature) by filtering *B-V*. For example, to filter stars with *B-V* values between -1 and 1, you apply the following OSC commands:

```
/Stellar/filter/B-V/less 1  
/Stellar/filter/B-V/greater -1
```

You can reset the specific filter by sending a *reset* command with the filter. Eg, to clear the *Hpmag* filters, send the following OSC:

```
/Stellar/filter/Hpmag/reset
```

You can reset all filters by sending the *resetFilters* command:

```
/Stellar/resetFilters
```

Chapter 2

Decoding OSC Messages

As stated in Section [1.1 – Launching Stellar Command](#), OSC is received from Stellar Command on the port that Stellar Command is configured with alongside the OSC name space. If for example, you had launched Stellar command as follows:

```
java -jar StellarCommand.jar port=1234 osc=/Stellar
```

You would receive all OSC messages from Stellar Command on port 1234 with all messages prefixed with the namespace *Stellar*. We will ignore the leading namespace in these instructions and assume that you will be filtering it.

2.1 osc

This should be the first message you receive from Stellar Command. The *osc* name indicates the UDP port you need to send OSC messages to Stelar Command on.

2.2 view

The *view* message will contain the following OSC arguments:

1. float - Field of view. This is the field of view in decimal degrees.
2. float - the Right Ascension (RA) of the centre point of the display
3. float - the Declination (Dec.) of the centre point of the display

Consider the following received message:

```
/Stellar/view 20.0 96.49851 -52.682182
```

This would indicate a 20 ° field of view, RA of 96.49851° and Dec. of -52.682182°.

2.3 Star Values

In many cases, sending astronomical data for all the stars within a certain radius a point would require create a message that would be too big for a single OSC packet. If we wanted to send astronomical data for say 1000 stars, we would need to send more than one packet. To accommodate this, Stellar Command will send multiple packets of data in bundles, with each packet containing OSC messages with three distinct types of OSC messages - 1 x *bundleCount* message, 1 x *names* message and 0 or more *values* messages.

For example, one complete set of astronomical data might appear as follows:

```
Bundle 1 of N
Star Column names
Star Values
Star Values
Star Values
...
Star Values

Bundle 2 of N
Star Column names
Star Values
Star Values
Star Values
...
Star Values
.....
Star Values
.....
Star Values
Star Values
Star Values
...
Star Values
```

2.3.1 **bundleCount**

The *bundleCount* message will contain the bundle number and the total number of bundles as arguments. The bundle number will be a zero based index, meaning that the first bundle will be bundle zero. For example, the *bundleCount* message for the first bundle of 15 bundles would appear as follows:

```
/Stellar/bundleCount 0 15
```

2.3.2 **names**

The names of the astronomical data columns is returned in the message *names*, with each OSC argument being the name of the data. Stellar Command has configured VizieR to use

Table 2.1: Sample correlated column names and data

| OSC Name | OSC arg |
|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| /names | RArad (deg) | DErad (deg) | pmRA (mas/yr) | pmDE (mas/yr) | Hpmag (mag) | B-V | |
| /values | 000.111046 | -79.061831 | 163.54 | -62.97 | 8.7854 | 0.778 | |
| /values | 001.002533 | -80.39506 | 55.39 | -18.26 | 7.9968 | 0.314 | |
| /values | 001.155615 | -81.345318 | 1.05 | 0.76 | 9.1961 | 0.239 | |
| /values | 001.34229 | -79.253912 | 23.95 | 66.32 | 7.8761 | 1.093 | |
| /values | 001.517384 | -84.09820 | 20.14 | 1.37 | 8.6327 | 1.576 | |
| /values | 001.85792748 | -77.49416 | -2.06 | -9.80 | 8.5621 | 1.014 | |

the Hipparcos catalogue. The column of data returned are as follows:

- RArad (deg) - Right Ascension in ICRS, Ep=1991.25
- DErad (deg) - Declination in ICRS, Ep=1991.25
- pmRA (mas/yr) - Proper motion in Right Ascension
- pmDE (mas/yr) - Proper motion in Declination
- Hpmag (mag) - Hipparcos magnitude
- B-V (mag) - Colour index

2.3.3 values

Each bundle will return zero or more OSC messages that contain the data values that correlate to the column names in the *names* message. Table 2.1 shows how the column names correlate to the astronomical data. Note that */Stellar* was removed from table example for brevity.

2.4 viewerObservationPoint

The *viewerObservationPoint* message will contain information regarding the viewers location within the stellarium simulation. The OSC arguments returned are:

1. float - latitude in degrees.
2. float -longitude in degrees
3. float - altitude in metres
4. string - planet observer is viewing from

For example, the following OSC message would indicate we are viewing from a latitude of 15.824176 South, longitude of 70.520546 East, 930m high from Earth.

| |
|--|
| /Stellar/viewerObservationPoint -15.824176 70.520546 930.0 Earth |
|--|

2.5 time

The *time* message indicates the simulated time being displayed on stellarium. The arguments in the OSC message if we were viewing at midday of 8 June 2019 in Sydney simulating a normal procession of time, would be as follows:

1. string - UTC as ISO formatted String. Eg, .2019-06-08T02:00:00.000Z
2. string - Local time as a string eg. 2019-06-08T12:00:00.000
3. float - GMT time shift in Julian days. eg 0.41666666
4. float - the current time rate in Julian days 1.1574074E-5

Note that time rate is actually a double precision number, however, OSC does not natively support double values. 1.1574074E-5 is approximately the number of days in a second.

Chapter 3

Using Stellar Command as a Java Library

This chapter details how to use Stellar Command as a library that you call directly from within your programming environment. If you intend to use Stellar Command as a standalone server application and communicate to it with Open Sound Control Messages through your preferred music package—such as Max MSP, SuperCollider or PD—you can skip back to chapter 1 – [Using Stellar Command with OSC](#).

The Stellar Command Library is a Java Archive that provides a far more efficient method of controlling Stellarium and accessing the data because the structures can be accessed directly without having to convert them to OSC. In some cases, converting values to OSC can result in data loss when the parameter is a double. Similarly, if a VizieR query contains many thousands of rows of data, these do not need to be encoded into OSC, parsed, and then decoded again. Furthermore, you can easily do efficient functions such as sorting and obtaining altitude and azimuth based on observer location and date and time directly.

The three main packages include in the Jar are Stellarium Control (packed *asstellarium*), VizieR query (packaged *vizier*), and complex data conversion (packaged *stellarstructures*).

3.1 stellarium

The simplest way to access Stellarium is through the *stellarium* package. The package contains various classes to simplify communication with Stellarium. Rather than detailing every function, which you can easily access through the online documentation, the class names and their basic functions will be outlined.

3.1.1 StellariumSlave

StellariumSlave is the class that does the REST communication to Stellarium and provides a common access point. The class contains a synchronised threaded model that enables API calls to Stellarium to complete asynchronously. If, for example, you wanted to set the azimuth of the Stellarium display, you would call *setAzimuth* with the number of degrees and your function would return immediately. The *StellariumSlave* class would then send the API request to Stellarium in a separate thread. This prevents the calling application from having to wait for Stellarium to process the request, which can often take over 100ms. *StellariumSlave* provides the facility to poll Stellarium for changes and provides

StellariumViewListener interfaces for classes *StellariumView*, *StellariumLocation* and *StellariumTime*.

3.1.2 StellariumView

StellariumView contains properties about the display of Stellarium. These include the field of view, and the RaDec. RaDec are collated into a single class, as detailed in section [3.3.2 – RaDec](#).

3.1.3 StellariumLocation

StellariumLocation contains information about the simulated location of the viewer in Stellarium. You can access parameters including latitude, longitude, altitude, planet and landscape.

3.1.4 StellariumTime

StellariumTime provides access to the time parameters of the Stellarium. These include UTC time, local date time and GMT time shift based on the viewer location, the Julian day, and the time rate that Stellarium is simulating.

3.1.5 StellariumProperty

The *StellariumProperty* has properties including the ability to show atmosphere, ground, star labels and constellation art. This is very much in progress and more properties will; be made accessible in time as the need arises.

3.2 vizier

The vizier packages provides the interface to VizieR database of astronomical catalogues.

3.2.1 VizierQuery

The *VizierQuery* class is where you are able to define the catalogue you want to use, outputs that you require, and any filters that you want to apply. You can then perform a read which will provide the astronomical data matching the query as text. This data can be reinterpreted using functions from the *stellarstructures* package, detailed in section [3.3 – stellarstructures](#).

The default catalogue used is catalogue used is I/311/hip2 - Hipparcos, the New Reduction (van Leeuwen, 2007). The default outputs provided are:

- RArad (deg) - Right Ascension in ICRS, Ep=1991.25
- DErad (deg) - Declination in ICRS, Ep=1991.25
- pmRA (mas/yr) - Proper motion in Right Ascension
- pmDE (mas/yr) - Proper motion in Declination

- Hpmag (mag) - Hipparcos magnitude
- B-V (mag) - Colour index

3.2.2 StellarDataTable

StellarDataTable performs analysis and conversion of VizieR data and stores the data into a table structure. The StellarDataTable contains the names of the columns as well as a list of *StellarDataRow* items that contain the astronomical data values.

3.2.3 StellarDataRow

A StellarDataRow contains astronomical data values for one star. The list of values match the column names in the StellarDataTable.

3.2.4 StellarFilteredTable

StellarFilteredTable is a class for sorting a StellarDataTable.

3.2.5 FilteredData

The *FilteredData* class is a utility for sorting StellarDataRow items based on AltAz

3.2.6 MagnitudeSort

the *MagnitudeSort* class is a utility for sorting StellarDataRow items based on magnitude

3.3 stellarstructures

The following classes provide basic encapsulation and some functions on data.

3.3.1 AltAz

Contains a combination of altitude and azimuth as a single structure;

3.3.2 RaDec

Contains the Right Ascension (RA) and the Declination (Dec.) as a single structure. RA is stored as decimal degrees rather than as hours, minutes and seconds.

3.3.3 ObservationalPoint

The *ObservationalPoint* contains information about the UTC date / time, latitude and longitude of a particular point. This information is often required for performing other functions such as converting the RA/Dec of a star to AltAz.

3.3.4 StellarConversions]

The *StellarConversions* is a utility class used for performing functions including converting RaDec to AltAz and converting Stellarium's three dimensional spherical points to RaDec.