The Stellar Command Module

for

Integrating Astronomy and Music

User Guide

Angelo Fraietta

University of New South Wales

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Preface

Stellar Command is a software system that integrates *stellarium* planetarium software with online astronomical data acquisition through *VizieR* database of astronomical catalogues [OBM00]. 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 [R+15, dM15, Lim15]. More recently, composers have used scientific data obtained from individual stars to generate sounds and have created compositions directly correlated to that data [Fra14b, Bri13].

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 multimedia artworks and installations [ZSW17, TIS+13].

Many composers have used the cosmos as inspiration or stimulus to their works, with many using scientific observations or data as input [Fra08b, Fra14b]. The *Quadrivium* linked astronomy, mathematics, geometry and music as a standard part of classical education up until the renaissance [LML+10]. Composers have been mapping mathematics and geometry to music since antiquity [Jam95, AF02]. 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." [RR79, 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" [RR79, 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 [Fra14b]. 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 [DSZ11, Fra14b]. 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 public 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 [Fra14a]. 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 [Bri13].

Although the binocular display has an awesome display–the actual night sky–“few people ventured outside to the astronomical equipment"[Fra14b, 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."[Fra14b, 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 [Fra19c].

Background

Stellarium

Stellarium is a software program designed to enable people to create a virtual planetarium using their home computer [ZW18]. It runs on Windows, OSX and Linux/Unix, including Raspberry Pi [ZW18, 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 [Ber08]. Stellarium has a very high quality graphical display, supporting spherical mirror projection that can be used with a dome [MC09] and is used in many schools and museums because it is both scientifically accurate and visually engaging [Ber08]. 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 [Ber08]. 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 lore[1] [RD76]. 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 [Ash15]. Archaeoastronomers also use the feature to generate an astronomical display from a location for a period sometime in the past [Zot14]. Also, the landscape feature facilitates a connection between landscape and skyscape, so one can map the sky against a landscape for research or aesthetics [ZSW17].

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 [ZW18].

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 [FB19]. The plugin also allows clients to query the status of Stellarium, including the area of sky currently being displayed. The exact celestial 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.

[1] The Tukano tribes are indigenous peoples of the northwestern region of Brazil [Kno76].

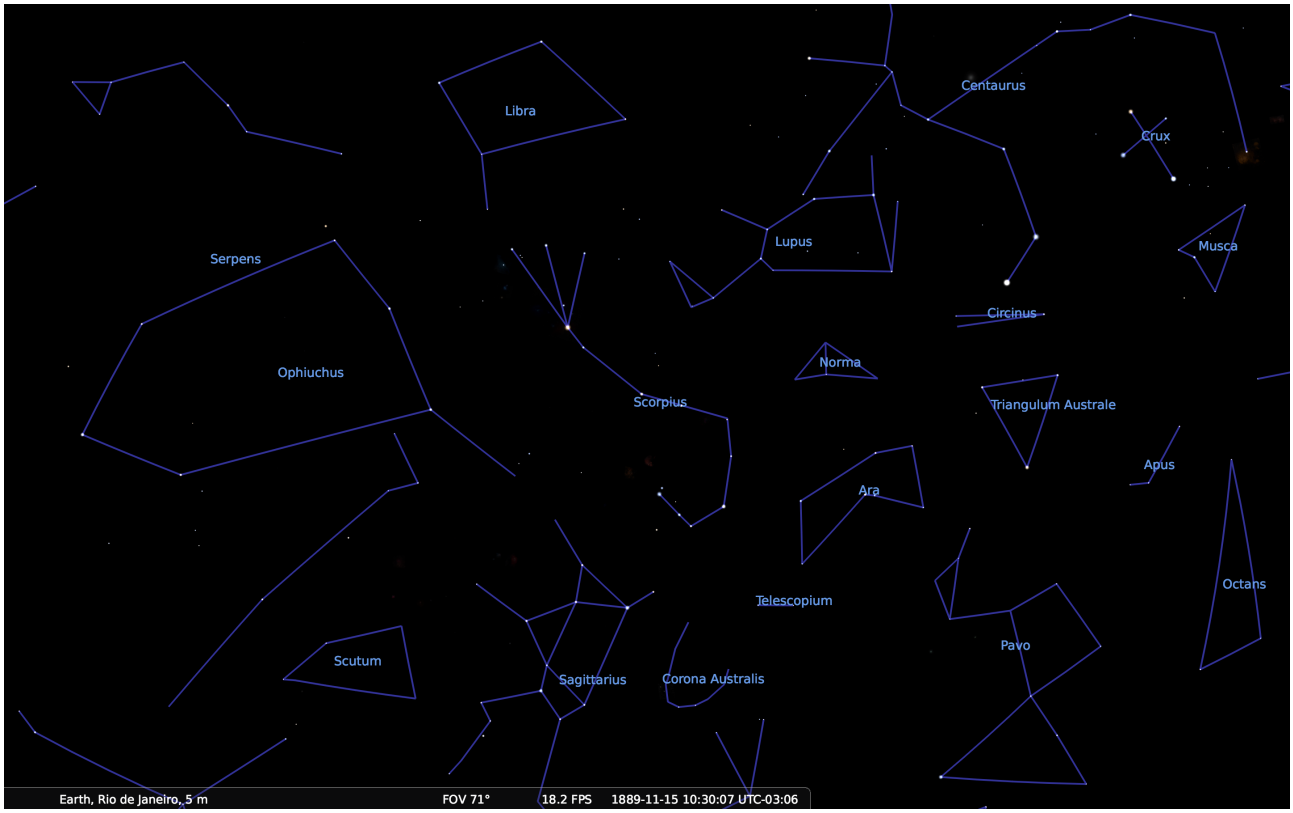

Figure 1: Constellations displayed in Western sky lore.

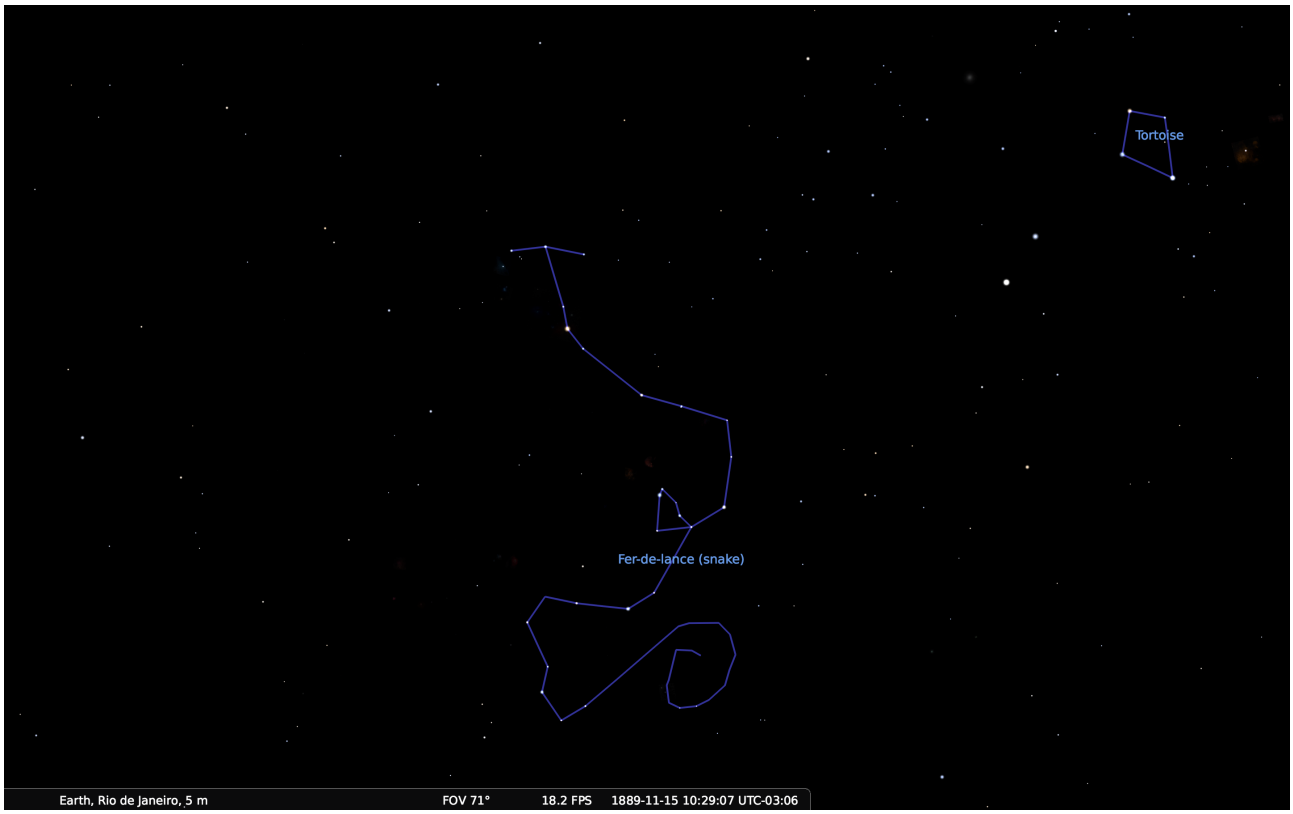

Figure 2: Constellations displayed in Tukano sky lore.

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" [OBM00, 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 [OBM00, p. 24] to currently 18000 [Viz18]. Catalogues can be selected by name or based on the wavelength, mission and astronomy type [Och10]. 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." [KBB+10, 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 equinox[2], and can be defined by object name, RA and Dec., or by IAU-coordinates [viz17].

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 " [VL97, 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.

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) [Lia99].

[2]The default equinox is J2000 [Och10].

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.

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

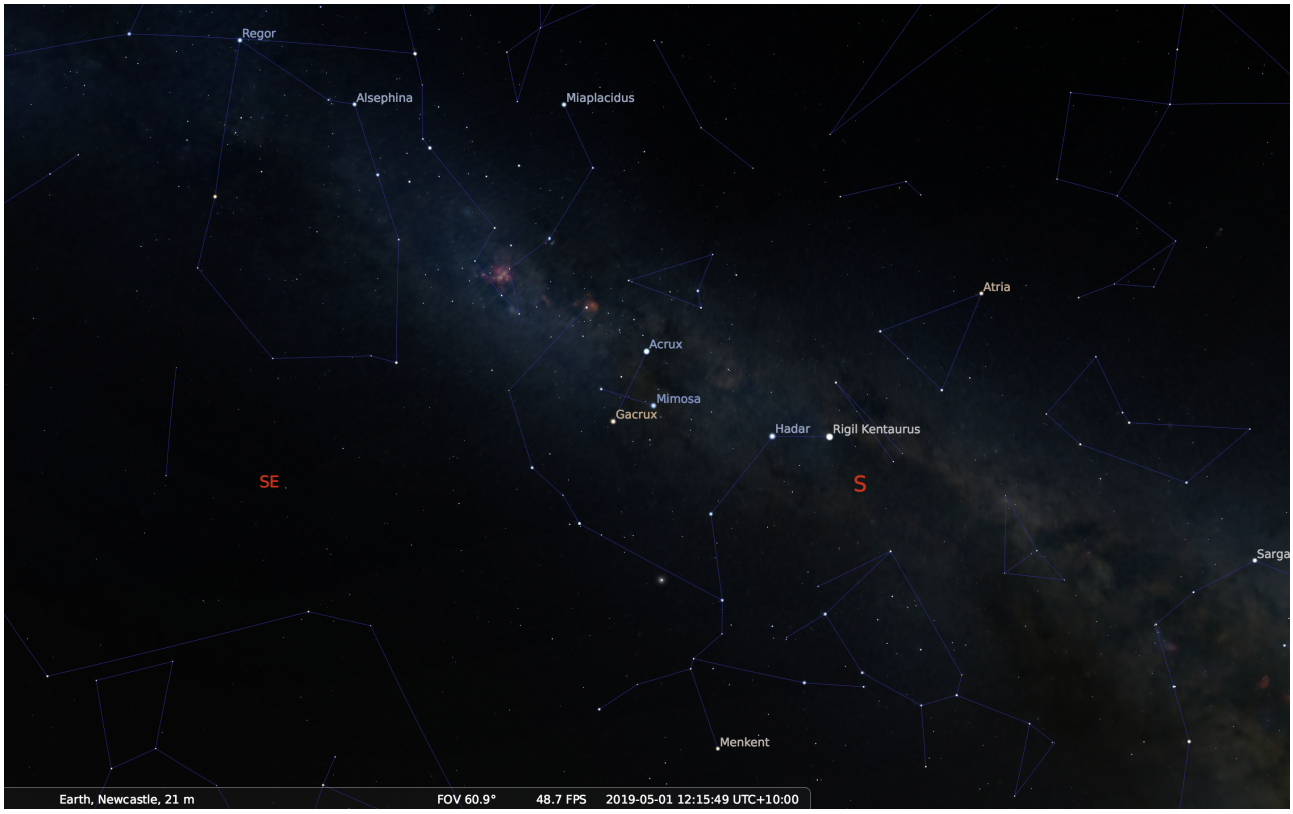

Figure 3: stellarium displaying a 60° field of view.

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

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

(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.

Time Rate

The Stellarium API uses Julian days per second in it's native API, which would mean the normal time rate would return 1.1574074E-5. If you set the time rate to 2 in the Stellarium native API, Stellarium would progress at 2 days per second. I believe that it is more intuitive to present the time rate as a value compared to the normal progression of time. For example, setting the time rate to 2 would simulate running 2x normal speed. Also, this aligns better with Stellarium scripts, which present the time as a rate compared to normal time. Eg, in the Stellarium scripting language, the following would indicate running at 2x normal speed.

```
core.setTimeRate(2);
```

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 affilia-

Figure 6: Midday showing the sky with atmosphere.

tion with Stonehenge" [**?**, p. 263]. Archaeoastronomers have used stellarium to generate an astronomical display from a location for a period sometime in the past [Zot14].

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 no atmosphere.

Figure 8: Constellation art.

Chapter 1

Installing Stellar Command

Installing Stellar Command is completed in two parts: the Stellar Command module and the examples. Additionally, there the option to download the source files from Github.

1.1 Downloading Stellar Command

Download the Stellar Command archive from https://github.com/angelofraietta/StellarCommand/blob/master/build/distributions/StellarCommand.zip

Unzip the archive StellarCommand.zip into a folder. Figure 1.1 displays the contenmts of the archive.

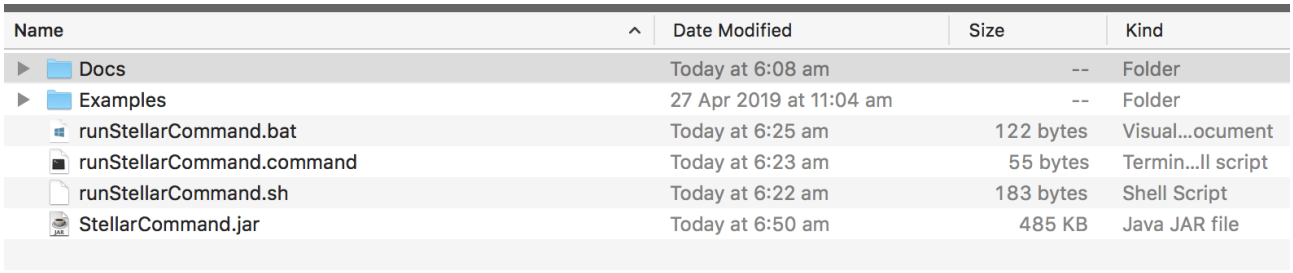

Figure 1.1: Stellar Command contents.

The *Docs* folder contains the documentation files. The *Examples* folder contains examples inside a *HappyBrackets* project. The file *StellarCommand.jar* is the Java archive that you need to include in your project if you are using it as a library. The remaining files are scripts for running in MacOS, Linux or Windows. Instructions for launching Stellar Command are in chapter 2 – *Sending OSC to Stellar Command*, with specifics on scripts in described in section 2.1.2 – *Using Configuration Scripts*.

1.2 Setup Examples

The examples for demonstrating Stellar Command are run using the HappyBrackets creative coding kit. In order to use the kit, you must install Java, IntelliJ and the HappyBrackets IntelliJ plugin. Instructions can be found at https://github.com/orsjb/HappyBrackets/wiki/Getting-Started.

Once you have installed HappyBrackets, open the Examples project by selecting *Open Project* in IntelliJ (Figure 1.2) and then selecting the *Examples* folder and pressing *Open*, as shown in Figure 1.3.

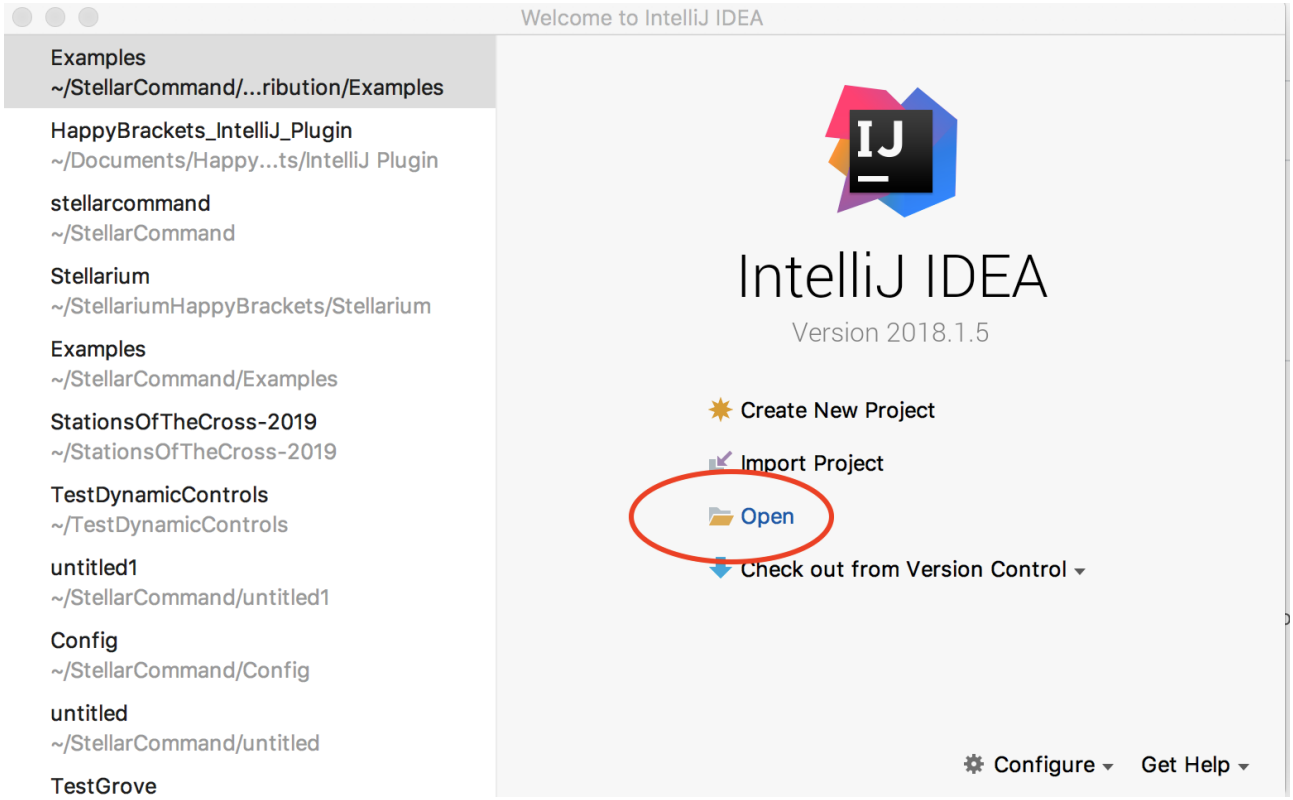

Figure 1.2: Select Open Project in IntelliJ

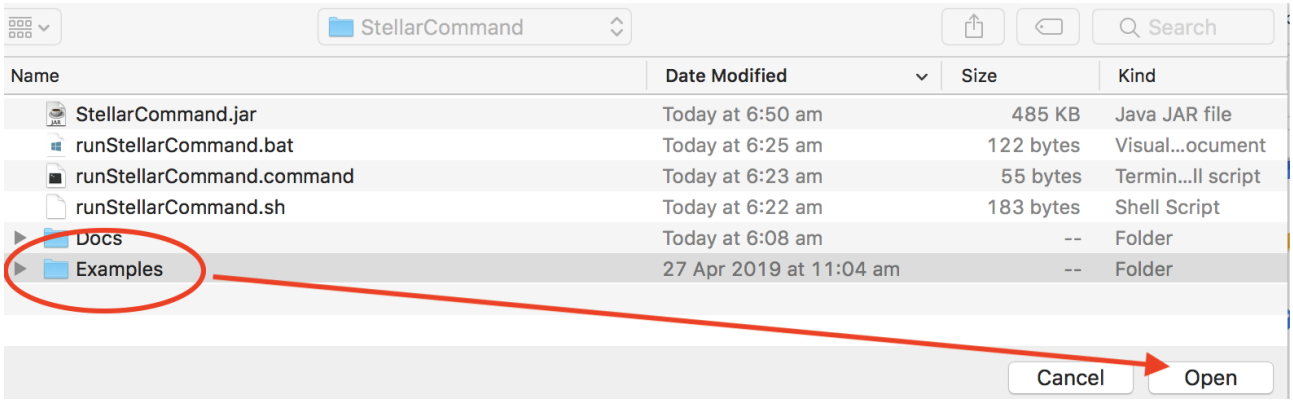

Figure 1.3: Select Examples and press "Open"

DO NOT enter into the *Examples* folder as shown in Figure 1.4 as this will not open the project. If you inadvertently get to this stage, press cancel and try again..

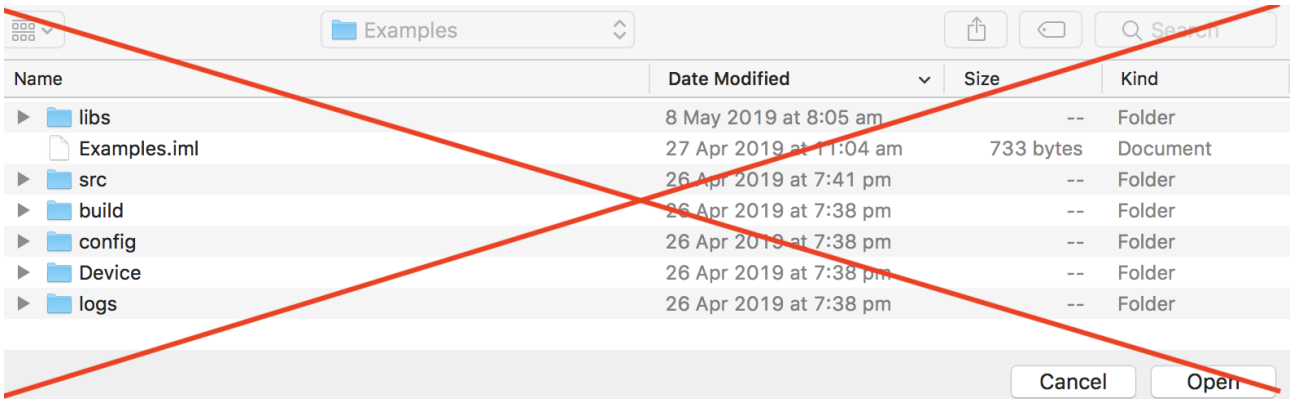

Figure 1.4: Incorrect attempt at opening Example Project

Chapter 2

Sending OSC to Stellar Command

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 5 – *Using Stellar Command as a Java Library*.

This chapter defines sending OSC to Stellar Command. Examples for using OSC can be found at chapter 4 – *OSC Examples*

2.1 Launching Stellar Command

Scripts are provided to save you having to type these parameters each time and is described in section 2.1.2 – *Using Configuration Scripts*. This section, however, will explain in detail what each parameter does. 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` will cause stellarium to sequentially try opening the ports listed, and if these all fail, will then

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

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 2.1, shows three OSC clients and a stellarium server on a LAN, and a remote stellarium server accessible from the internet through *myserver.com*.

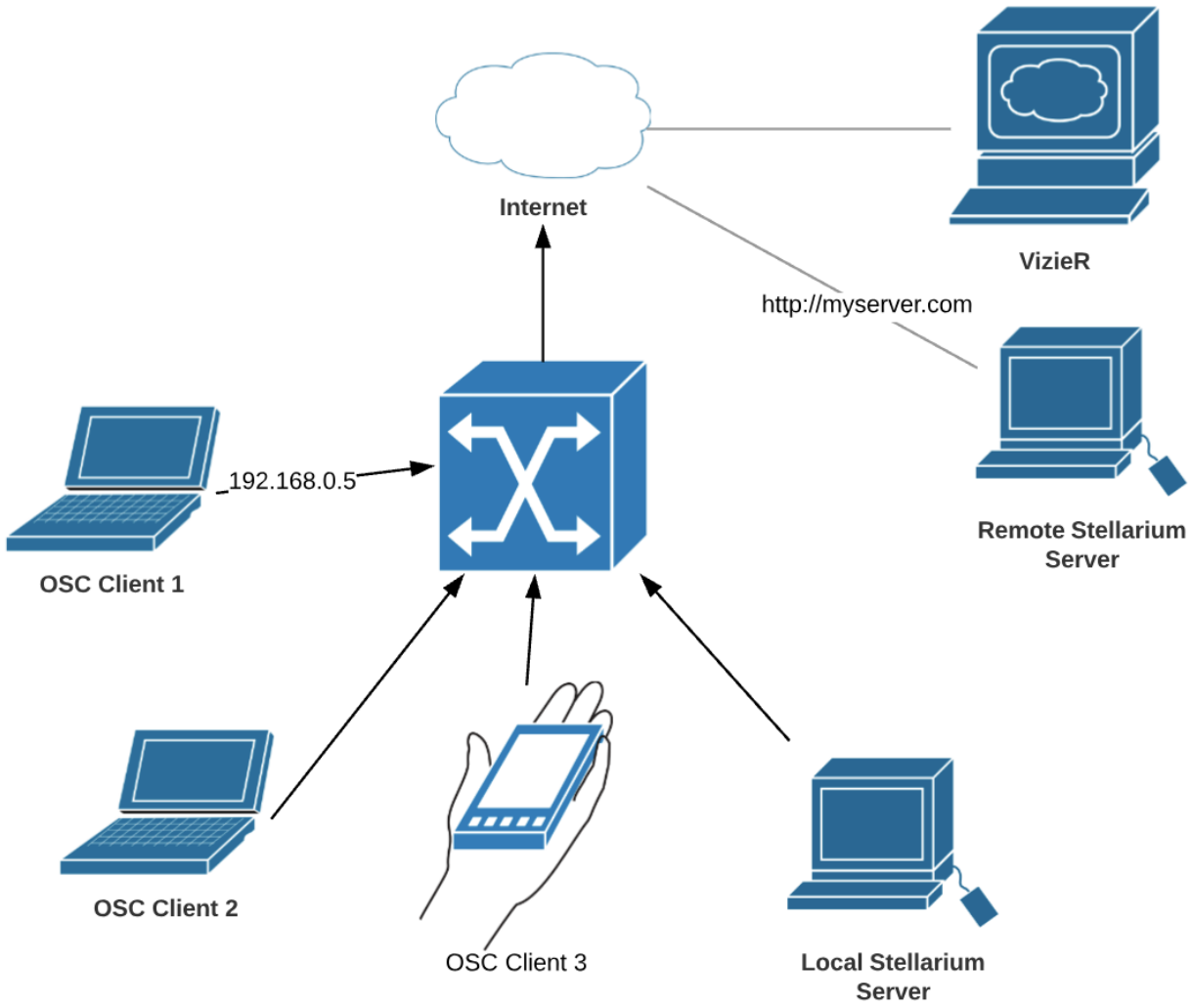

Figure 2.1: Remote Stellarium and OSC Clients.

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 8090[2], 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
```

Stellar Command works by polling Stellarium to detect changes. This is a requirement because Stellarium uses a REST API rather than a persistent socket. By default, the Stellarium will be polled once per second, however, you can change this by adding a parameter *stellariumpoll*, which dictates how often Stellarium is polled in milliseconds. For example, the following would cause Stellarium to be polled approximately every two seconds. Although the parameter is in milliseconds, causing Stellarium to poll more often than once per second can slow your system down significantly.

```
java -jar StellarCommand.jar port=1234 osc=/Stellar \
stellariumpoll=2000
```

If you set a polling time of zero, then Stellarium will not be polled, however, you can request a poll from your own application be sending the *poll* command - described in section 2.2.1 - *poll*.

The 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. You can configure this in Stellar Command by adding the parameter *stellariumport*, which dictates the port Stellar Command will use to attempt to communicate with Stellarium. For example, the following would cause Stellar Command to communicate with Stellarium on port 2000. It is extremely unlikely you would need to change this parameter, however, the facility is there in case you do need to.

```
java -jar StellarCommand.jar port=1234 osc=/Stellar \
stellariumport=2000
```

2.1.1 Launch Without Stellarium

. In some cases, you may want to launch Stellar Command to access the functionality of VizieR without using Stellarium. This may be the case if you are only interested in the astronomical data and are not using Stellarium as an interface device. This is be accomplished by adding the value *disabled* to the *stellarium* parameter. For example, the following command line argument will disable calls to Stellarium.

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

[2]The 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

2.1.2 Using Configuration Scripts

Three scripts are provided to facilitate simple configuration and launching of Stellar Command. The script *runStellarCommand.sh* is used to configure and launch Stellar Command from the commandline in MacOS and Linux. The file *runStellarCommand.bat* is used to configure and launch on Windows . There is an additional file, *runStellarCommand.command*, which allows a person to write the configuration inside the *.sh* file and then launch Stellar Command by double clicking *runStellarCommand.command* with the mouse.

Configuring Stellar Command is done simply by replacing the value in the script with those required.

```
PORT=1234
OSC=/Stellar
TRYPORTS="3333,4444,5555"
CLIENT=
STELLARIUM=
STELLARIUMPOLL=
STELLARIUMPORT=
```

If, for example, you wanted Stellar Command to send it's messages to your client on port 5678, simply change text after *PORT*=.

```
PORT=5678
```

The same applies for the other parameters.

2.2 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.

2.2.1 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. Additionally, the poll message will cause Stellar Command to return the running version of Stellar Command.

```
Stellar/poll
```

[3] Section 2.2.3 – *fieldOfView* shows how to do this.

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

```
/Stellar/osc 3333
```

See sections 3.2 - *version* and 3.1 - *osc*for more detail.

2.2.2 queryVizieR

It is possible to run Stellar Command and make calls to VizieR without running Stellarium, as detailed in section 2.1.1 - *Launch Without Stellarium*. This might be the case if you were only interested in the astronomical data without any correlation to a Stellarium Display, for example, if you were using a graphic programs such as Processing. The call is made by defining a centre radius as a float. If querying using RA and Dec, the second and third arguments are floats. For example, the following call would query a 1°radius at an RaDec of 186.6492, -63.0989

```
/Stellar/osc 1.0 186.6492 -63.0989
```

Additionally, it is possible to query using a star or celestial object instead of RA and Dec. For example, to query a 1°circle around Acrux would be performed as follows:

```
/Stellar/osc 1.0 Acrux
```

Stellar Command will respond by sending Star Values as OSC messages. See section 3.2 - *version* for details on Star Values.

2.2.3 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:

```
/Stellar/fieldOfView 60.0
```

2.2.4 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, using Z for zero time, or -/-HH:MM for a different GMT time shift.

The following will set to 5:51:17pm on 18 April 2019 at GMT.

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

The following will set to 5:51:17pm on 18 April 2019 at GMT + 5 Hours. Note that you must add the "+" or "-" sign and it must be a string value.

```
/Stellar/time 2019 4 18 17 51 17 +05:00
```

If the time zone is blank, the local time at the location will be used. 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
```

You can also set the time by typing the exact string value.

```
/Stellar/time 2019-04-06T12:00:00Z
```

2.2.5 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
```

2.2.6 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
```

2.2.7 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
```

2.2.8 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
```

2.2.9 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
```

2.2.10 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
```

2.2.11 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
```

2.2.12 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
```

2.2.13 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
```

2.2.14 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
```

2.2.15 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
```

2.2.16 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
```

2.2.17 timeRate

It is possible to change the simulated time rate on Stellarium by sending the *timeRate* command followed by the new time rateas 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 2.0, the display would simulate time running at twice normal speed. You can also make time go in reverse by setting the time rate to a negative number. To set the display to simulate time running at twice normal speed, you would send the following:

```
/Stellar/timeRate 2.0
```

See section – *Time Rate* for time rate specifics.

2.2.18 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
```

2.2.19 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
```

2.2.20 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 that 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 filed 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 cleat the Hpmag filters, send the following OSC:

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

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

```
/Stellar/resetFilters
```

2.2.21 script

An extremely powerful feature of Stellarium is the ability to run predefined astronomy shows. These scripts are written in JavaScript and are run within the Stellarium core. You can run, stop or find out the status of the scripting engine in Stellarium with the *script command*.

You can run a specific script by setting the first OSC argument as the name of the script. Sending the following OSC message would cause the *The Jack Bennett Catalog* packaged with Stellarium to run.

```
/Stellar/script bennett.ssc
```

If you choose to keep your scripts separate to those packaged with Stellarium, you will need o pass the full path as the OSC argument. For example,

```
/Stellar/script /Stellar/script /Users/me/myprivatescripts/stellariumscript/bennett.ssc
```

Stopping a script is done by using the word "stop" as the OSC argument.

```
/Stellar/script stop
```

Calling the script command without an argument will cause Stellar Command to send back the scripting engine status. Section 3.7 - *script* details the status of a returned script status message.

Chapter 3

Receiving OSC Messages from Stellar Command

This chapter defines decoding OSC from Stellar Command. Examples for using OSC can be found at chapter 4 – *OSC Examples*

As stated in Section 2.1 – *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.

3.1 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 Stellar Command on. The message is normally received when Stellar Command starts of as a response to a *poll* message. See section 2.2.1 - *poll* for more detail.

3.2 version

This message will indicate the running version of Stellar Command. The three int OSC arguments returned are major, minor, and build. The following received message would indicate the running version of Stellar Command is 1.0.0.

```
/Stellar/version 1 0 0
```

The message is normally received when Stellar Command starts of as a response to a *poll* message. See section 2.2.1 - *poll* for more detail.

3.3 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°.

3.4 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
....
Bundle N of N
Star Column names
Star Values
Star Values
Star Values
...
Star Values
```

Table 3.1: Sample correlated column names and data

OSC Name†	OSC arg	OSC arg	OSC arg	OSC arg	OSC arg	OSC arg
/names	RArad ††	DErad	pmRA	pmDE	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

3.4.1 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
```

3.4.2 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 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

3.4.3 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 3.1 shows how the column names correlate to the astronomical data. [1] [2]

Figure 4.6, shown in section 4.2 – *Receiving OSC in Examples*, shows how the data is received from Stellar Command as OSC.

[1] † */Stellar* was removed from table example for brevity.

[2] ††The actual columns names are *RArad (deg)*, *DErad (deg)*, *pmRA (mas/yr)*, *pmDE (mas/yr)* and *Hpmag (mag)*, however, they have been abbreviated to fit the table.

3.5 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
```

3.6 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 are 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 as a multiplier eg. 1.0

See section – *Time Rate* for time rate specifics.

3.7 script

The *script* message will return the status of the Stellarium scripting engine. A zero value for the OSC argument indicates that no script is running, while a non zero indicates a script is running. For example, the following message would indicate no script is currently running.

```
/Stellar/script 0
```

Requesting the script status is done by sending the script request messages, as detailed in section 2.2.21 - *script*.

Chapter 4

OSC Examples

The examples provided use the HappyBrackets creative coding toolkit and are coded in Java. For details on installing HappyBrackets, see section 1.2 – *Setup Examples*

The examples are configured to use port 1234 as the port you will receive OSC from Stellar Command. Additionally, it is configured to have Stellar Command listend on ports 3333, 4444 or 5555. These are the same configuration setting used in the description in section 2.1 – *Launching Stellar Command*. Examples of videos showing OSC messages using with Stellar Command can be found at https://www.youtube.com/playlist?list=PLZvzDOOUIKxO-Vlnt30ciKefTcKxupzOn.

4.0.1 Example Source Location

The source files for the OSC examples are under the *src/stellarcommandexamples/osc* folder, as shown in Figure 4.1.

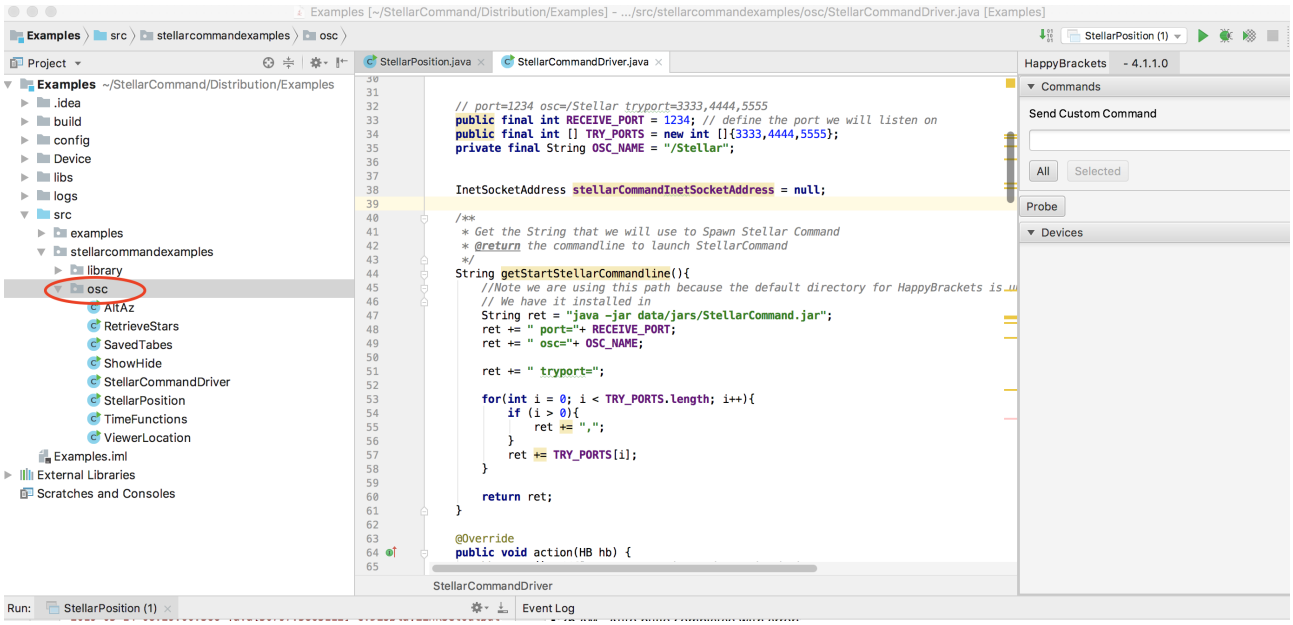

Figure 4.1: OSC examples folder.

4.0.2 Human Readable OSC Messages

OSC messages have bee converted to a human readable string through the function *StellarOSCVocabulary.getOscAsText*. This function has been used to facilitate displaying OSC

messages in both the send and receive OSC examples.

4.0.3 Java Docs

Documentation for the examples is effected through Java docs, which can be accessed by pressing the *F1* key.

Often, constants have been used instead of typing literal string values. For example, Figure 4.2 shows the constant *StellarOSCVocabulary.CommandMessages.ALTITUDE* used instead of typing the literal "altitude". You can view the value of the constant by selecting it in the

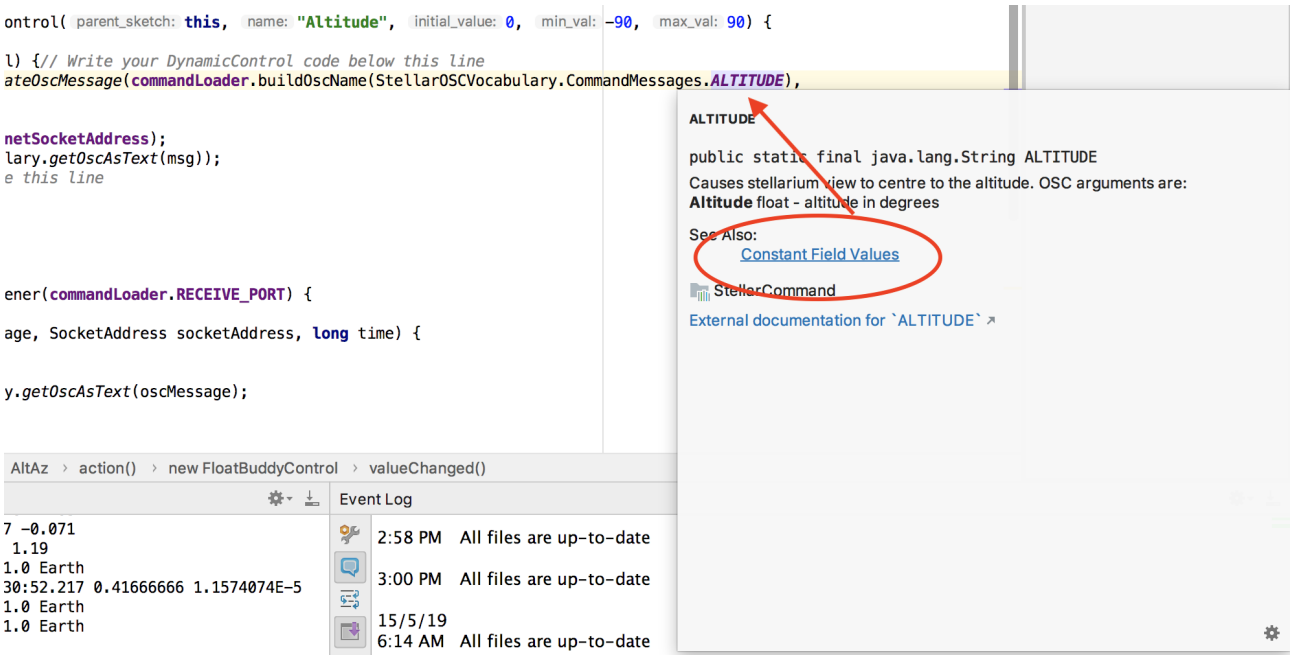

Figure 4.2: OSC examples folder.

editor, pressing the *F1* key and then clicking the *Constant Field Values* link. This will display the Java documentation as shown in Figure 4.3.

constantDisplay

4.1 Sending OSC in Examples

Some of the examples display sending OSC messages to Stellar Command. These examples use HappyBrackets dynamic controls, which are basically text boxes, sliders, checkboxs and buttons, to create OSC messages, which are sent to Stellar Command. For example, Figure 4.4 shows the *AltAz* example for changing the altitude and azimuth. In the image, the azimuth is being changed by the controls in the red box, which call the function *valueChanged*. The OSC message is packed and sent to Stellar Command using the functions *commandLoader.buildOscName* and *OSCMessageBuilder.createOscMessage*. The actual OSC message contents, however, are displayed in the *Diagnostics* text box so you can see what the actual OSC message sent is, which in the case of the Figure 4.4 is */Stellar/azimuth 35.064934* [1]. All the examples will display the OSC message sent in the diagnostic window.

[1] See 2.2.7 – *azimuth* for details of this message.

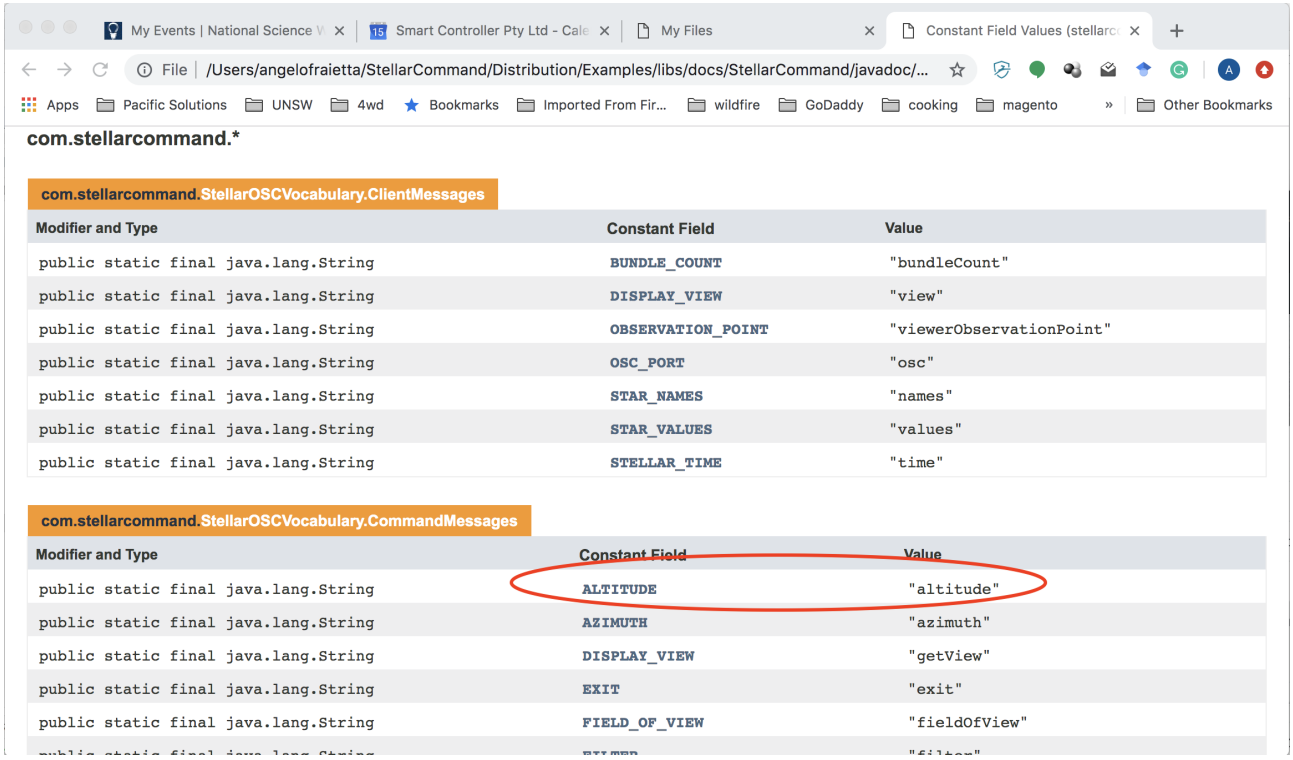

Figure 4.3: OSC examples folder.

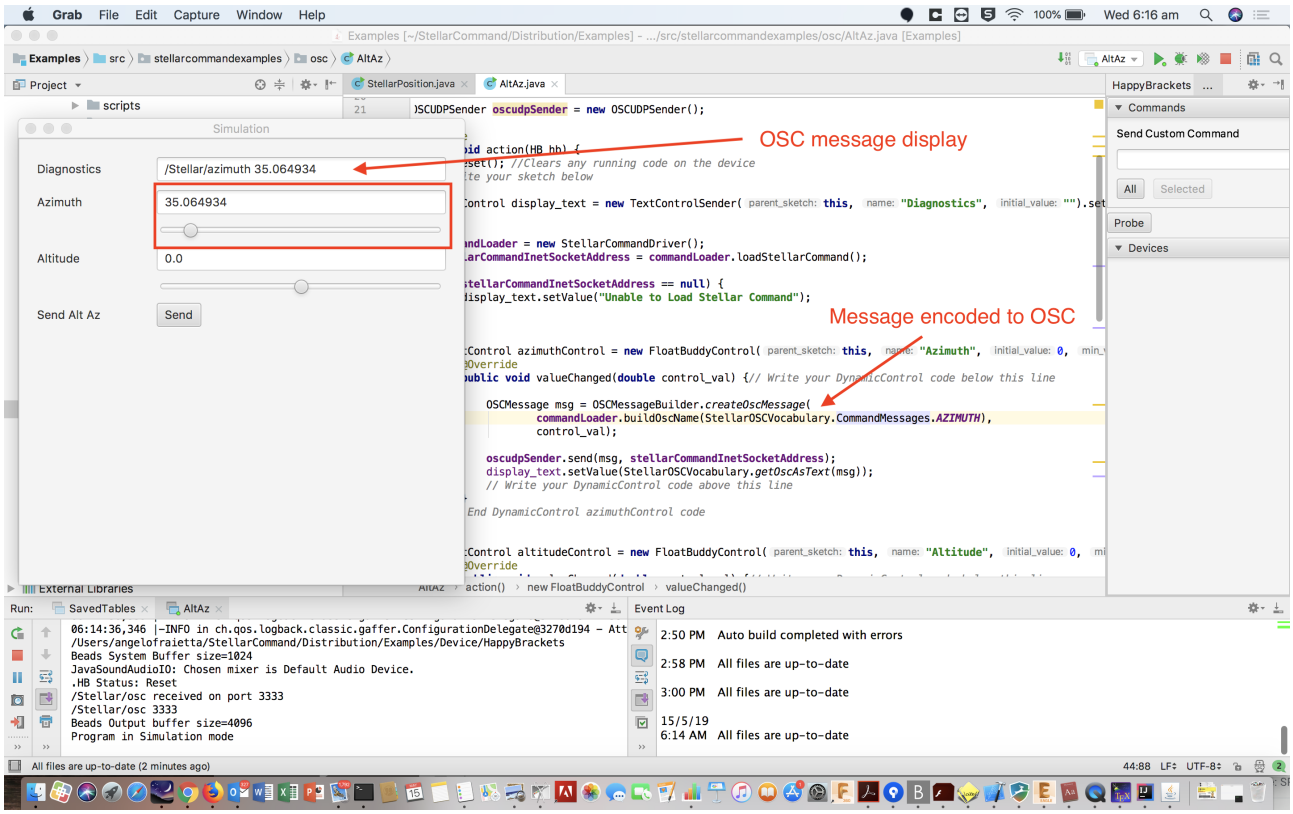

Figure 4.4: AltAz example.

4.2 Receiving OSC in Examples

Some of the examples display OSC messages that have been received from Stellar Command. For example, Figure 4.5 shows OSC received through the *OSCReceived* function. The OSC message is decoded and displayed in the IntelliJ console output, shown inside the red box.

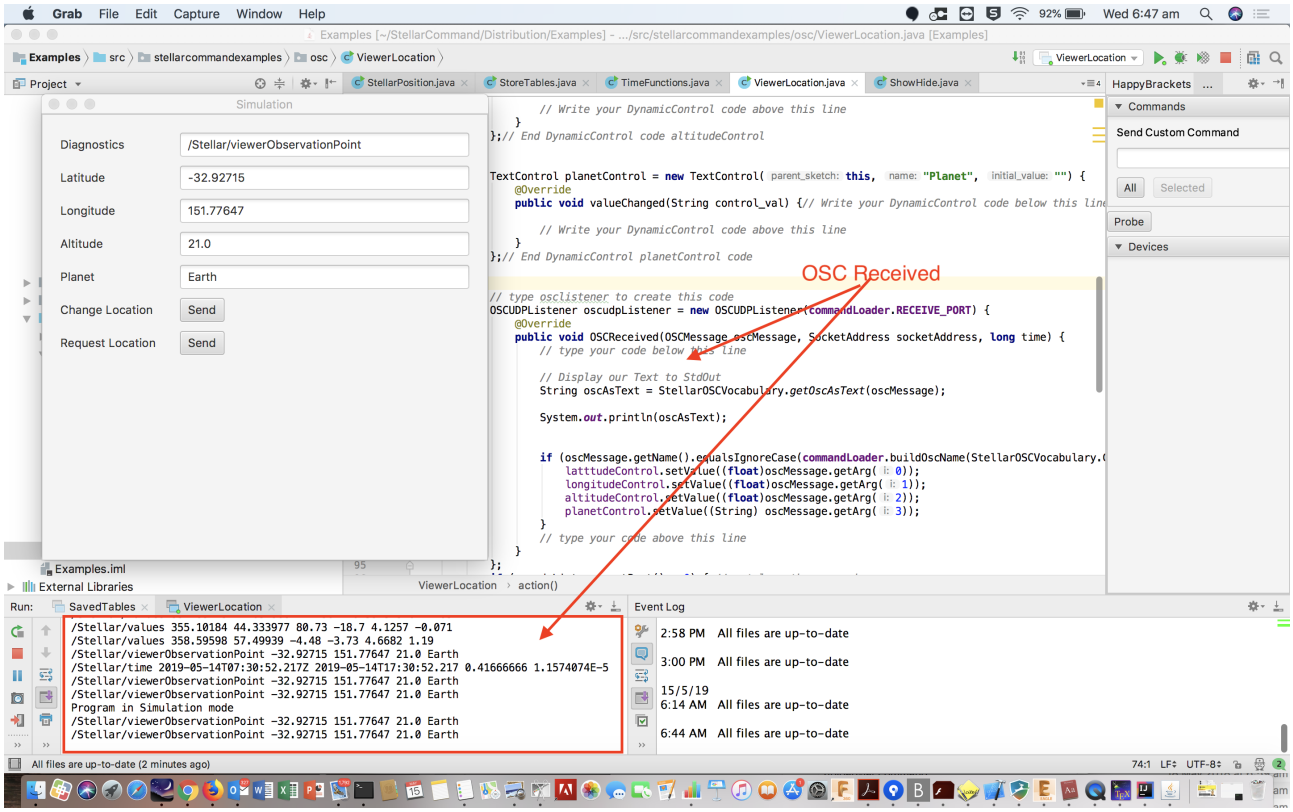

Figure 4.5: View Observation Point Example.

Bundles are separated in the examples, with each message in the bundle processed in sequence, as shown in Figure 4.6.

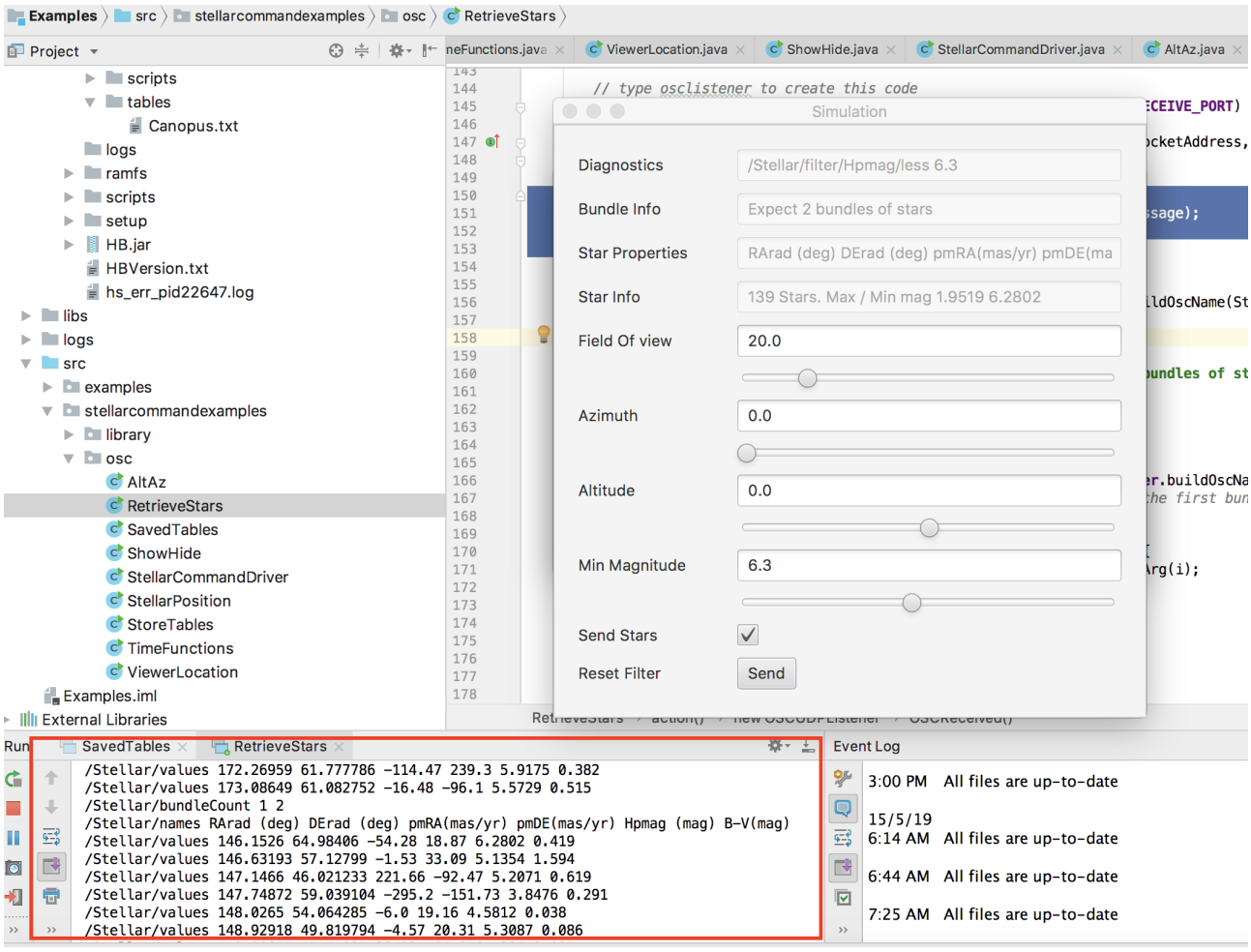

Figure 4.6: Receiving Stellar Data Example.

Chapter 5

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 2 – *Sending OSC to Stellar Command*.

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 as*stellarium*), VizieR query (packaged *vizier*), and complex data conversion (packaged *stellarstructures*).

5.1 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.

5.1.1 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 form 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*.

5.1.2 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 5.3.2 – *RaDec*.

5.1.3 StellariumLocation

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

5.1.4 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.

5.1.5 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.

5.2 vizier

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

5.2.1 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 cn 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 5.3 – *stellarstructures*.

The default catalogue used is catalogue used is I/311/hip2 - Hipparcos, the New Reduction [](StellarCommandManual). 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

5.2.2 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.

5.2.3 StellarDataRow

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

5.2.4 StellarFilteredTable

StellarFilteredTable is a class for sorting a StellarDataTable.

5.2.5 FilteredData

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

5.2.6 MagnitudeSort

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

5.3 stellarstructures

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

5.3.1 AltAz

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

5.3.2 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.

5.3.3 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 RADec of a star to AltAz.

5.3.4 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.

Bibliography

[AAU18] Astronomy archives user group (AAUG). https://www.cosmos.esa.int/web/esdc/archives-user-groups/astronomy, 2018. Accessed: 2019-01-18.

[AF02] Gerard Assayag and Hans G Feichtinger. *Mathematics and music: A Diderot mathematical forum*. Springer Science & Business Media, 2002.

[Ano19] Anonymous. Reference suppressed for anonymity during peer review. 2019.

[AOB14] Samuel Aaron, Dominic Orchard, and Alan F Blackwell. Temporal semantics for a live coding language. In *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design*, pages 37–47. ACM, 2014.

[Ash15] Joseph Ashley. Computers and computer programs. In *Astrophotography on the Go*, pages 151–161. Springer, 2015.

[Bad14] Yusuf Abdullahi Badamasi. The working principle of an Arduino. pages 1–4. IEEE, September 2014.

[Bar16] Steven F. (Steven Frank) Barrett. *Bad to the Bone : crafting electronic systems with BeagleBone Black*. Synthesis digital library of engineering and computer science. Second edition. edition, 2016.

[Ber08] K Berglund. Using free, open source Stellarium software for iya2009. In *Preparing for the 2009 International Year of Astronomy: A Hands-On Symposium*, volume 400, page 483, 2008.

[BF17] Oliver Bown and Sam Ferguson. Creative media+ the internet of things= media multiplicities. *Leonardo*, (Early Access):53–54, 2017.

[BF18] Oliver Bown and Sam Ferguson. Understanding media multiplicities. *Entertainment Computing*, 25:62–70, 2018.

[BFF+19] O. Bown, A. Fraietta, S. Ferguson, L Loke, and L. Bray. Strategies to facilitate rapid creative development with multiple networked devices using Happy-Brackets. In *International Conference on New Interfaces for Musical Expression (NIME-2019)*. Federal University of Rio Grande do Sul, 2019.

[Bin10] A. Binstock. Infoworld review: Top Java programming tools. https://www.infoworld.com/article/2683534/development-environments/infoworld-review--top-java-programming-tools.html, 2010. Accessed: 2018-07-05.

[BL15] Ilias Bergstrom and R Beau Lotto. Code bending: A new creative coding practice. *Leonardo*, 48(1):25–31, 2015.

[BLFR15] Oliver Bown, Lian Loke, Sam Ferguson, and Dagmar Reinhardt. Distributed interactive audio devices: Creative strategies and audience responses to novel musical interaction scenarios. In *International Symposium on Electronic Art*. ISEA, 2015.

[Bri86] Reginald Smith Brindle. *Musical composition*. Oxford University Press, 1986.

[Bri13] Colin Bright. spa-c–e, 2013. Performed live at Colbourne Ave Glebe, Sydney, Australia May 23rd 2013 by The Colin Bright Syzygy Band and Angelo Fraietta.

[BWG+06] Michael Barnett, Heather Wagner, Anne Gatling, Janice Anderson, Meredith Houle, and Alan Kafka. The impact of science fiction film on student understanding of science. *Journal of Science Education and Technology*, 15(2):179–191, 2006.

[BYJ13] Oliver Bown, Miriama Young, and Samuel Johnson. A Java-based remote live coding system for controlling multiple Raspberry Pi units. In *ICMC*, 2013.

[C+03] PC/104 Embedded Consortium et al. Pc/104 specification version 2.5. *San Francisco: PC/104 Embedded Consortium*, 2003.

[Cag07] Nergiz Ercil Cagiltay. Teaching software engineering by means of computer-game development: Challenges and opportunities. *British Journal of Educational Technology*, 38(3):405–415, 2007.

[CMWW11] Jitong Chen, Lingquan Meng, Xiaonan Wang, and Chenhui Wang. An integrated system for astronomical telescope based on Stellarium. In *Advanced Computer Control (ICACC), 2011 3rd International Conference on*, pages 431–434. IEEE, 2011.

[Coi00] Raimundo Olavo Coimbra. *A bandeira do Brasil: raízes histórico-culturais*. Instituto Brasileiro de Geografia e Estatística-IBGE, 2000.

[dBAB+00] Pea de Bernardis, Peter AR Ade, JJ Bock, JR Bond, J Borrill, A Boscaleri, K Coble, BP Crill, G De Gasperis, PC Farese, et al. A flat universe from high-resolution maps of the cosmic microwave background radiation. *Nature*, 404(6781):955, 2000.

[DC90] Stephen E. Deering and David R. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Trans. Comput. Syst.*, 8(2):85–110, May 1990. Accessed: 2018-07-05.

[Dea09] Roger T Dean, editor. *The Oxford handbook of computer music*. OUP USA, 2009.

[DeB00] George E DeBoer. Scientific literacy: Another look at its historical and contemporary meanings and its relationship to science education reform. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, 37(6):582–601, 2000.

[DJ00] GUSTAVO Diaz-Jerez. Algorithmic music: using mathematical models in music composition. *The Manhattan School of Music*, 2000.

[dM15] Flávia Cristina de Mello. *Astronomy and Cosmology of the Guarani of Southern Brazil*, pages 975–980. Springer New York, New York, NY, 2015.

[DM18] Roger T Dean and Alex McLean. *The Oxford Handbook of Algorithmic Music*. Oxford University Press, 2018.

[DSC05] Anderson Faustino Da Silva and Vitor Santos Costa. An experimental evaluation of Java JIT technology. *J. UCS*, 11(7):1291–1309, 2005.

[DSZ11] Peter Duffett-Smith and Jonathan Zwart. Practical astronomy with your calculator or spreadsheet. *Cambridge University Press,*, 2011.

[Dua10] Paulo Araújo Duarte. Astronomia na Bandeira Brasileira. https://web.archive.org/web/20080502120005/http://www.cfh.ufsc.br/~planetar/textos/astroban.htm, 2010. Accessed: 2018-12-21.

[End99] Mica R Endsley. Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics*, 42(3):462–492, 1999.

[Far15] Eleanor Farrington. Parametric equations at the circus: Trochoids and poi flowers. *The College Mathematics Journal*, 46(3):173–177, 2015.

[FB17] Sam Ferguson and Oliver Bown. Creative coding for the Raspberry Pi using the HappyBrackets platform. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, pages 551–553. ACM, 2017.

[FB19] Angelo Fraietta and Oliver Bown. Creating a sonified spacecraft game using HappyBrackets and Stellarium. In *Procedings of the 17th Linux Audio Conference (LAC-19)*, pages 1–7. CCRMA, Stanford University, USA, 2019.

[FM11] Emmanuel Fléty and Côme Maestracci. Latency improvement in sensor wireless transmission using IEEE 802.15. 4. In *New Interfaces for Musical Expression (NIME 2011)*, pages 409–412, 2011.

[Fra05a] Angelo Fraietta. The smart controller workbench. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 46–49. University of British Columbia, Vancouver, 2005.

[Fra05b] Angelo Fraietta. Smart controller/bell garden demo. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 260–261. National University of Singapore, 2005.

[Fra06] Angelo Fraietta. *The Smart Controller – an integrated electronic instrument for real-time performance using programmable logic control*. Phd, Western Sydney University, 2006.

[Fra08a] Angelo Fraietta. Open Sound Control: Constraints and limitations. In *International Conference on New Interfaces for Musical Expression (NIME-2008)*, pages 19–23, 2008.

[Fra08b] A Fraknoi. Music inspired by astronomy: A selected listing for the international year of astronomy. In *Preparing for the 2009 International Year of Astronomy: A Hands-On Symposium*, volume 400, page 514, 2008.

[Fra14a] Angelo Fraietta. Echoes from the fourth day - a segue through the southern night sky for FM synthesiser and binoculars, 2014. Performed in Brickworks Park in collaboration with the Newcastle Astronomical Society.

[Fra14b] Angelo Fraietta. Musical composition with naked eye and binocular astronomy. In *Australasian Computer Music Conference 2014*, pages 47–52. Victorian College of the Arts, 2014.

[Fra19a] A Fraietta. Creating order and progress. In *International Conference on New Interfaces for Musical Expression (NIME-2019)*. Federal University of Rio Grande do Sul, 2019.

[Fra19b] A Fraietta. Stellar command: a planetarium based cosmic performance interface. In *International Conference on New Interfaces for Musical Expression (NIME-2019)*. Federal University of Rio Grande do Sul, 2019.

[Fra19c] A. Fraietta. Stellar command software module. https://github.com/angelofraietta/StellarCommand, 2019. Accessed: 2019-02-02.

[Gam93] Olympic Games. Music of the spheres. 1993.

[GBL94] Patricia M Greenfield, Craig Brannon, and David Lohr. Two-dimensional representation of movement through three-dimensional space: The role of video game expertise. *Journal of applied developmental psychology*, 15(1):87–103, 1994.

[Gep18] Alexander CT Geppert. *Imagining outer space: European astroculture in the twentieth century*. Springer, 2018.

[Ger05] Lincoln Geraghty. Creating and comparing myth in twentieth-century science fiction:“star trek" and “star wars". *Literature/Film Quarterly*, 33(3):191–200, 2005.

[Gre10] Bruce Gregory. The integration of classical music composition theory with mind-body hypnotherapy. *Australian Journal of Clinical and Experimental Hypnosis (Online)*, 38(1):1, 2010.

[HA07] Jean-Michel Hoc and René Amalberti. Cognitive control dynamics for reaching a satisficing performance in complex dynamic situations. *Journal of cognitive engineering and decision making*, 1(1):22–55, 2007.

[HC02] Cay S Horstmann and Gary Cornell. *Core Java 2: Volume I, Fundamentals*. Pearson Education, 2002.

[Hey03] Paul Heyer. America under attack i: a reassessment of orson welles' 1938 war of the worlds broadcast. *Canadian Journal of Communication*, 28(2):149, 2003.

[Hur58] Paul D Hurd. Science literacy: Its meaning for american schools. *Educational leadership*, 16(1):13–16, 1958.

[Hym07] Mark Hyman. The first mind-body medicine: Bringing shamanism into the 21st century. *Alternative therapies in health and medicine*, 13(5):10–11, 2007.

[Ing17] Simon Ings. Plane speaking: Analogue tech powers a futuristic artwork. *New Scientist Archive*, 235(3133):43, 2017.

[Jam95] Jamie James. *The music of the spheres: Music, science, and the natural order of the universe*. Springer Science & Business Media, 1995.

[KBB+10] David G Koch, William J Borucki, Gibor Basri, Natalie M Batalha, Timothy M Brown, Douglas Caldwell, Jørgen Christensen-Dalsgaard, William D Cochran, Edna DeVore, Edward W Dunham, et al. Kepler mission design, realized photometric performance, and early science. *The Astrophysical Journal Letters*, 713(2):L79, 2010.

[KCC+07] Amit Kapadia, Fabien Chéreau, Lars Lindberg Christensen, Lars Holm Nielsen, Adrienne Gauthier, Robert Hurt, and Ryan Wyatt. Vamp in Stellarium/virgo: A proof of concept. *Proceedings from Communicating Astronomy with the Public*, 2007.

[KG07] Yasmin B Kafai and Michael T Giang. Virtual playgrounds: Children's multiuser virtual environments for playing and learning with science. *Children's learning in a digital world*, pages 196–217, 2007.

[KJ] Petr Kubánek and Martin Jelínek. Rts2–open source observatory manager.

[KK02] James Keogh and Jim Keogh. *J2EE: The complete reference*. McGraw-Hill/Osborne, 2002.

[KK06] Zoltán Kolláth and Jen O Keuler. Stellar acoustics as input for music composition. *Musicae Scientiae*, 10(1_suppl):161–183, 2006.

[Kno76] Francis Knobloch. The Tukano indians and advancing "civilisation". *Mankind Quarterly*, 17(2), 1976.

[Lar02] Craig Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design and the unified process*. Prentice Hall, second edition, 2002.

[LBF+18] L. Loke, O Bown, S Ferguson, L Bray, A Fraietta, and K Packham. Your move sounds so predictable! In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY 2018) Companion Extended Abstracts*, pages 121–125. ACM, 2018.

[Led02] Jim Ledin. Simulation takes off with hardware. *Embedded Systems Programming*, 15(4):19–29, April 2002.

[Lia99] Sheng Liang. *The Java Native Interface: Programmer's Guide and Specification*. Addison-Wesley Professional, 1999.

[Lim15] Flávia Pedroza Lima. *Astronomy in Brazilian Ethnohistory*, pages 945–951. Springer New York, New York, NY, 2015.

[LK06] Jim Lovell and Jeffrey Kluger. *Apollo 13*. Houghton Mifflin Harcourt, 2006.

[LML+10] Miranda Lundy, John Martineau, Miranda Lundy, Daud Sutton, Anthony Ashton, and Jason Martineau. *Quadrivium: The four classical liberal arts of number, geometry, music, & cosmology*. Walker & Company, 2010.

[LYBB14] Tim Lindholm, Frank Yellin, Gilad Bracha, and Alex Buckley. *The Java virtual machine specification*. Pearson Education, 2014.

[LZ12] Yuxi Liu and Guohui Zhou. Key technologies and applications of internet of things. pages 197–200. IEEE, January 2012.

[Mag11] Thor Magnusson. Algorithms as scores: Coding live music. *Leonardo Music Journal*, pages 19–23, 2011.

[Mag14] Thor Magnusson. Scoring with code: Composing with algorithmic notation. *Organised Sound*, 19(3):268–275, 2014.

[Mah05] Michael S Mahoney. The histories of computing (s). *Interdisciplinary Science Reviews*, 30(2):119–135, 2005.

[Mal82] Thomas W Malone. Heuristics for designing enjoyable user interfaces: Lessons from computer games. In *Proceedings of the 1982 conference on Human factors in computing systems*, pages 63–68. ACM, 1982.

[Mar11] Michael Margolis. *Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects*. "O'Reilly Media, Inc.", 2011.

[MC09] Matthew Mc Cool. Touring the cosmos through your computer: a guide to free desktop planetarium software. *CAPjournal,(7)*, pages 21–23, 2009.

[McC18] Mark McCurry. Rtosc-realtime safe open sound control messaging. In *Linux Audio Conference 2018*, page 51, 2018.

[Mir01] Eduardo Miranda. *Composing music with computers*. Focal Press, 2001.

[MJM+16] Andrew P McPherson, Robert H Jack, Giulio Moro, et al. Action-sound latency: Are our tools fast enough? 2016.

[MKC12] Jim Murphy, Ajay Kapur, and Dale Carnegie. Musical robotics in a loudspeaker world: Developments in alternative approaches to localization and spatialization. *Leonardo Music Journal*, pages 41–48, 2012.

[MLF+02] Dieter Mehrholz, L Leushacke, W Flury, R Jehn, H Klinkrad, and M Landgraf. Detecting, tracking and imaging space debris. *ESA Bulletin(0376-4265)*, (109):128–134, 2002.

[MML18] Hicham Medromi, Laila Moussaid, and FAL Laila. Analysis of the allocation of classes, threads and cpu used in embedded systems for Java applications. *Procedia computer science*, 134:334–339, 2018.

[Mor96] Henry M Morris. Meeting user needs keeps board business booming. *Control Engineering*, 43(11):D, 1996.

[MQW10] Steve Massey, Steve Quirk, and Fred Watson. *Atlas of the Southern Night Sky*. New Holland Publishers, 2010.

[OBM00] François Ochsenbein, Patricia Bauer, and James Marcout. The VizieR database of astronomical catalogues. *Astronomy and Astrophysics Supplement Series*, 143(1):23–32, 2000.

[Och10] François Ochsenbein. The "vizquery" program. http://vizier.u-strasbg.fr/vizier/doc/vizquery.htx, 2010. Accessed: 2019-01-18.

[Oui10] Hector Ouilhet. Google sky map: using your phone as an interface. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 419–422. ACM, 2010.

[Pac96] Jozef Pacholczyk. Music and astronomy in the muslim world. *Leonardo*, 29(2):145–150, 1996.

[R+15] Clive LN Ruggles et al. *Handbook of Archaeoastronomy and Ethnoastronomy*. Springer New York, 2015.

[RCL99] Bryan L Riemann, Nancy A Caggiano, and Scott M Lephart. Examination of a clinical method of assessing postural control during a functional performance task. *Journal of Sport Rehabilitation*, 8(3):171–183, 1999.

[RD76] Gerardo Reichel-Dolmatoff. Cosmology as ecological analysis: a view from the rain forest. *Man*, pages 307–318, 1976.

[RML+06] Jukka Rönkkö, Jussi Markkanen, Raimo Launonen, Marinella Ferrino, Enrico Gaia, Valter Basso, Harshada Patel, Mirabelle D'Cruz, and Seppo Laukkanen. Multimodal astronaut virtual training prototype. *International Journal of Human-Computer Studies*, 64(3):182–191, 2006.

[RNC+03] Ricardo Rosas, Miguel Nussbaum, Patricio Cumsille, Vladimir Marianov, Mónica Correa, Patricia Flores, Valeska Grau, Francisca Lagos, Ximena López, Verónica López, et al. Beyond nintendo: design and assessment of educational video games for first and second grade students. *Computers & Education*, 40(1):71–94, 2003.

[RPK+15] Karen Robson, Kirk Plangger, Jan H Kietzmann, Ian McCarthy, and Leyland Pitt. Is it all a game? Understanding the principles of gamification. *Business Horizons*, 58(4):411–420, 2015.

[RPK+16] Karen Robson, Kirk Plangger, Jan H Kietzmann, Ian McCarthy, and Leyland Pitt. Game on: Engaging customers and employees through gamification. *Business horizons*, 59(1):29–36, 2016.

[RR79] John Rodgers and Willie Ruff. Kepler's harmony of the world: A realization for the ear: Three and a half centuries after their conception, Kepler's data plotting the harmonic movement of the planets have been realized in sound with the help of modern astronomical knowledge and a computer-sound synthesizer. *American Scientist*, 67(3):286–292, 1979.

[RT15] Fabienne Reynard and Philippe Terrier. Role of visual input in the control of dynamic balance: variability and instability of gait in treadmill walking while blindfolded. *Experimental brain research*, 233(4):1031–1040, 2015.

[RW12] Matt Richardson and Shawn Wallace. *Getting started with Raspberry Pi*. "O'Reilly Media, Inc.", 2012.

[Sar01] John M Sarkissian. On eagle's wings: The parkes observatory's support of the apollo 11 mission. *Publications of the Astronomical Society of Australia*, 18(3):287–310, 2001.

[spa16] Space debris motion translated into music. https://www.youtube.com/watch?v=PJ8ojV5hiOk, 2016. Accessed: 2019-01-18.

[SPS11] Robert J Stone, Peter B Panfilov, and Valentin E Shukshunov. Evolution of aerospace simulation: From immersive virtual reality to serious games. In *Recent Advances in Space Technologies (RAST), 2011 5th International Conference on*, pages 655–662. IEEE, 2011.

[Tel06] JP Telotte. Lost in space: Television as science fiction icon. *Journal of Popular Film and Television*, 33(4):178–186, 2006.

[TFB17] Luca Turchet, Carlo Fischione, and Mathieu Barthet. Towards the internet of musical things. In *Proceedings of the Sound and Music Computing Conference*, pages 13–20, 2017.

[TFE+18] Luca Turchet, Carlo Fischione, Georg Essl, Damián Keller, and Mathieu Barthet. Internet of musical things: Vision and challenges. *IEEE Access*, 6:61994–62017, 2018.

[TIS+13] Elena Tuveri, Samuel A Iacolina, Fabio Sorrentino, L Davide Spano, and Riccardo Scateni. Controlling a planetarium software with a kinect or in a multitouch table: a comparison. In *Proceedings of the Biannual Conference of the Italian Chapter of SIGCHI*, page 6. ACM, 2013.

[TMF16] Luca Turchet, Andrew McPherson, and Carlo Fischione. Smart instruments: Towards an ecosystem of interoperable devices connecting performers and audiences. In *Proceedings of the Sound and Music Computing Conference*, pages 498–505, 2016.

[Tuf01] Edward R. Tufte. The visual display of quantitative information, 2001.

[vdVdBvO12] JW van der Veen, R de Beer, and D van Ormondt. Utilizing Java concurrent programming, multi-processing and the Java native interface. *Running Native Code in Separate Parallel Processes," Report on behalf of the Marie-Curie Research Training Network FAST*, 2012.

[viz17] Vizier help - faq - tutorial output preferences and constraint specifications. http://vizier.u-strasbg.fr/viz-bin/vizHelp?3.htx#target, 2017. Accessed: 2019-01-18.

[Viz18] VizieR. https://vizier.u-strasbg.fr/viz-bin/VizieR, 2018. Accessed: 2019-01-18.

[VL97] Floor Van Leeuwen. The Hipparcos mission. *Space Science Reviews*, 81(3-4):201–409, 1997.

[VL07] Floor Van Leeuwen. Validation of the new hipparcos reduction. *Astronomy & Astrophysics*, 474(2):653–664, 2007.

[Wal67] D Perkin Walker. Kepler's celestial music. *Journal of the Warburg and Courtauld Institutes*, pages 228–250, 1967.

[WF+97] Matthew Wright, Adrian Freed, et al. Open SoundControl: A new protocol for communicating with sound synthesizers. In *ICMC*, 1997.

[Win95] Todd Winkler. Making motion musical: Gesture mapping strategies for interactive computer music. pages 261–64. The International Computer Music Association, The International Computer Music Association, 1995.

[Wol01] Mark JP Wolf. Genre and the video game. *The medium of the video game*, pages 113–134, 2001.

[ZC11] Gabe Zichermann and Christopher Cunningham. *Gamification by design: Implementing game mechanics in web and mobile apps*. "O'Reilly Media, Inc.", 2011.

[ZFLZ17] Sichen Zhao, Yuan Fang, Wenfeng Li, and Kanglian Zhao. Design and implemention of an emulation node for space network protocol testing. In *International Conference on Machine Learning and Intelligent Communications*, pages 658–667. Springer, 2017.

[ZN12] Georg Zotti and Wolfgang Neubauer. A virtual reconstruction approach for archaeoastronomical research. In *Virtual Systems and Multimedia (VSMM), 2012 18th International Conference on*, pages 33–40. IEEE, 2012.

[Zna02] Alfred Znamierowski. *The world encyclopedia of flags: The definitive guide to international flags, banners, standards and ensigns*. Hermes House, 2002.

[Zot14] Georg Zotti. Towards serious gaming for archaeoastronomical simulation. *Mediterranean Archaeology & Archaeometry*, 14(3), 2014.

[ZSW17] Georg Zotti, Florian Schaukowitsch, and Michael Wimmer. The skyscape planetarium, 2017.

[ZW18] Georg Zotti and Alexander Wolf. stellarium 0.18.0 user guide. 2018.

[ZWC+10] Qian Zhu, Ruicong Wang, Qi Chen, Yan Liu, and Weijun Qin. Iot gateway: Bridgingwireless sensor networks into internet of things. In *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on*, pages 347–352. Ieee, 2010.

Index