1 What is Gaia Sky
  1.1 Main features ................................................. 3

2 Contents
  2.1 Download ......................................................... 5
  2.2 Requirements and Installation ................................ 5
    2.2.1 System requirements .................................. 5
    2.2.2 Installation and uninstallation ....................... 5
    2.2.3 Running from source .................................. 7
  2.3 Running Gaia Sky .............................................. 8
    2.3.1 Running from source .................................. 8
    2.3.2 Running from downloaded package .................... 9
  2.4 Documentation ................................................ 9
    2.4.1 Settings and configuration ............................ 9
    2.4.2 The properties file .................................. 12
    2.4.3 Graphics settings ................................... 13
    2.4.4 Graphics performance ................................ 15
    2.4.5 User Interface ......................................... 18
    2.4.6 Controls ................................................ 21
    2.4.7 Camera modes .......................................... 25
    2.4.8 Stereoscopic (3D) mode ............................... 27
    2.4.9 Planetarium mode ..................................... 29
    2.4.10 360 mode ............................................... 29
    2.4.11 Recording and playing camera paths ............... 31
    2.4.12 Performance .......................................... 32
    2.4.13 Internal reference system ......................... 34
    2.4.14 Data: catalogues and formats ..................... 34
    2.4.15 Scripting ............................................. 44
    2.4.16 Capturing videos .................................... 45
    2.4.17 Taking screenshots .................................. 46
  2.5 Javadoc ....................................................... 47
  2.6 Changelog ..................................................... 47
  2.7 Gaia Sky screenshots ....................................... 47
  2.8 About ........................................................ 50
    2.8.1 Contact ............................................... 50
    2.8.2 Author ................................................ 50
    2.8.3 Acknowledgements ..................................... 50
Note: 11/11/2016 – Version 1.0.1 is out!
CHAPTER 1

What is Gaia Sky

Gaia Sky is a real-time, 3D, astronomy visualisation software that runs on Windows, Linux and MacOS. It lives in the framework of ESA’s Gaia mission to chart about 1 billion stars of our Milky Way Galaxy. It is developed in the Gaia group of the Astronomisches Rechen-Institut (ZAH, Universität Heidelberg).

1.1 Main features

- Free and open source – The application is free and open source and will stay this way. You can even contribute to the development or the translations.
• **From Gaia to the cosmos** – Move freely through the cosmos and descend to the surface of any Planet or do a close-up inspection of Gaia. All movement and transitions are seamless!

• **Gaia** - Observe Gaia in its orbit and discover its movement in the sky and its attitude.

• **3D ready** – With 4 stereoscopic modes: Anaglyphic (red-cyan), VR headset, 3DTV, cross-eye.

• **Planetarium projection mode** – Ready to produce videos for full dome systems.

• **360 mode** – Ready to produce 360 videos.

• **Use your own data** – Comes with HYG and TGAS. Supports VOTable, FITS, CSV and all formats accepted by STIL.

• **Navigate the galaxy** – Support for controllers and gamepads makes navigating the Galaxy a piece of cake.

• **Spacecraft mode** – Jump aboard a spacecraft to navigate the skies!

• **Record and play your camera paths** – Ready to record and play camera paths off-the-shelf.

• **Scriptable and extensible** – Use Python to script and extend the capabilities of Gaia Sky.

• **Internationalised** – Translated so far to English, German, Spanish and Catalan.
2.1 Download

Gaia Sky is available for Linux, OS X and Windows.

- Download the latest version for your OS.
- Download old versions.

2.2 Requirements and Installation

In the sections below is the information on the minimum hardware requirements and on how to install the software.

2.2.1 System requirements

Here are the minimum requirements to run this software:

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Windows 7+ / MacOS X / Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel Core i3 3rd Generation or similar</td>
</tr>
<tr>
<td>GPU</td>
<td>OpenGL 3.0 support / Intel HD 4000 / Nvidia GeForce 8400 GS, 500 MB GRAM</td>
</tr>
<tr>
<td>Memory</td>
<td>3 GB RAM</td>
</tr>
<tr>
<td>Hard drive</td>
<td>230 MB of free space</td>
</tr>
<tr>
<td>Java</td>
<td>On Linux, you need the Java Runtime Environment 7+ installed (openJRE is fine)</td>
</tr>
</tbody>
</table>

2.2.2 Installation and uninstallation

Depending on your system and your personal preferences the installation procedure may vary. Below is a description of the various installation methods available.

- Download Gaia Sky

Windows

Two windows installers are available for 32 and 64-bit systems.

- `gaiasky_windows_<version>.exe` - 32 bit installer.
- `gaiasky_windows-x64_<version>.exe` - 64 bit installer.
To install the Gaia Sky, just double click on the installer and then follow the on-screen instructions. You will need to choose the directory where the application is to be installed.

In order to uninstall the application you can use the Windows Control Panel or you can use the provided uninstaller in the Gaia Sky folder.

**Linux**

We provide 3 packages for Linux systems (**deb** for **Debian**, **Ubuntu** and derivatives, **rpm** for **RedHat**, **Fedora** and derivatives and a **linux installer** which works on all distros) plus an **AUR package** for **Arch** and derivatives.

**deb**

This is the package for Debian-based distros (**Debian**, **Ubuntu**, **Mint**, **SteamOS**, etc.). Download the `gaiasky_<version>.deb` file and run the following command. You will need root privileges to install a **deb** package in your system.

```
$ dpkg -i gaiasky_<version>.deb
```

This will install the application in the `/opt/gaiasky/` folder and it will create the necessary shortcuts and `.desktop` files. The package depends on the `default-jre` package, which will be installed if it is not yet there.

In order to **uninstall**, just type:

```
$ apt-get remove gaiasky
```

**rpm**

This is the package for RPM-based distributions (**Red Hat**, **Fedora**, **Mandriva**, **SUSE**, **CentOS**, etc.) Download the `gaiasky_linux_<version>.rpm` file and run the following command. You will need root privileges to install a **rpm** package in your system.

```
$ yum install gaiasky_linux_<version>.rpm
```

This will install the application in the `/opt/gaiasky/` folder and it will create the necessary shortcuts.

In order to **uninstall**, just type:

```
$ yum remove gaiasky-x86
```

**Install from AUR**

If you have **Arch**, **Manjaro**, **Antergos** or any other Arch Linux derivative, you can install the **package from AUR** using any tool able to install AUR software. For example:

```
$ yauort -S gaiasky
```

This will download the package and install it in the `/opt/gaiasky/` folder. It also creates the necessary shortcuts.

**Unix/Linux installer**

We also provide a **Unix/Linux installer** that will trigger a graphical interface where you can choose the installation location and some other settings. Download the file `gaiasky_unix_<version>.sh` to your disk. Then run the following to start the installation.

```
$ ./gaiasky_unix_[version].sh
```

Follow the on-screen instructions to proceed with the installation.

In order to **uninstall**, just execute the **uninstall** file in the installation folder.
OS X - Mac

For OS X we provide a `gaiasky_macos_<version>.dmg` file here, which is installed by unpacking into the Applications folder. Once unpacked, the installer will come up, you just need to follow its instructions.

Compressed (TGZ) package

A `gaiasky-[version].tgz` package file is also provided here. It will work in all systems but you need to unpack it yourself and create the desired shortcuts. In Windows, use an archiver software (7zip, iZArc, etc.) to unpack it. In Linux and OS X, you can use:

```
$ tar xzvf gaiasky-<version>.tgz
```

### 2.2.3 Running from source

#### Requirements

If you want to compile the source code, you will need the following:

- JDK7 or above (JDK8 recommended)

Please, be aware that only tags are guaranteed to work (here). The master branch holds the development version and the configuration files are possibly messed up and not ready to work out-of-the-box. So remember to use a tag version if you want to run it right away from source.

Also, this guide is for Unix-like systems only. If you are working on Windows, you will need git for windows and Power Shell, even though it has not been tested.

First, clone the repository:

```
$ git clone https://github.com/ari-zah/gaiasky.git
$ cd gaiasky
```

#### Compile and run

To compile the code and run the desktop version of the application:

```
$ gradlew desktop:run
```

#### Package Gaia Sky

To pack the application into a `tar` file:

```
$ gradlew desktop:createTar
```

In order to produce the desktop installers for the various systems you need a licensed version of Install4j.

```
$ gradlew desktop:pack
```

These commands will compile and package the application into a `gaiasky-[version]` folder under the `gaiasky/releases` folder.
2.3 Running Gaia Sky

If you installed the software using an installer or a package manager system (rpm, deb), you just need to use the standard running procedures of your Operating System to run the application.

Windows
In windows, this means clicking on Start and then browsing the start menu folder Gaia Sky. You can run the executable from there.

Linux
Just type gaiasky in a terminal or use your favourite desktop environment search engine to run the Gaia Sky launcher.

OS X
Locate the launcher using the main search.

Code and pakcage
However, if you are a maverick and do not like installers, you can also run the Gaia Sky directly from the source code in GitHub or using the tgz package.

2.3.1 Running from source

Requirements
If you want to compile the source code, you will need the following:

• JDK7 or above (JDK8 recommended)

Please, be aware that only tags are guaranteed to work (here). The master branch holds the development version and the configuration files are possibly messed up and not ready to work out-of-the-box. So remember to use a tag version if you want to run it right away from source.

Also, this guide is for Unix-like systems only. If you are working on Windows, you will need git for windows and Power Shell, even though it has not been tested.

First, clone the repository:

```bash
$ git clone https://github.com/ari-zah/gaiasky.git
$ cd gaiasky
```

Compile and run
To compile the code and run the desktop version of the application:

```bash
$ gradlew desktop:run
```

Package Gaia Sky
To pack the application into a tgz file:

```bash
$ gradlew desktop:createTar
```

In order to produce the desktop installers for the various systems you need a licensed version of Install4j.
These commands will compile and package the application into a gaiasky-[version] folder under the gaiasky/releases folder.

### 2.3.2 Running from downloaded package

If you prefer to run the application from the tar.gz package, follow the instructions below.

#### Linux

In order to run the application on Linux, open the terminal, uncompress the archive, give execution permissions to the run.sh file and then run it.

```bash
$ tar zxvf gaiasky-[version].tar.gz
$ cd gaiasky-[version]/
$ gaiasky
```

#### Windows

In order to run the application on Windows, open a terminal window (type cmd in the start menu search box) and run the run.bat file.

```bash
cd path_to_gaiasky_folder
gaiasky.bat
```

#### OS X

To run the application on MacOS systems, follow the instructions in the Linux section.

### 2.4 Documentation

#### 2.4.1 Settings and configuration

Gaia Sky can be configured using the provided GUI (Graphical User Interface). However, some functions are not represented in the GUI, so you may need to dive deep into the properties file.

**Graphics settings**

Please refer to the Graphics settings chapter.

**User interface**

The User Interface section allows the user to set the language and the theme of the user interface.

One can select between a choice of languages using the language drop-down. There are currently three visual themes available:
• dark-orange, black and orange theme.
• dark-orange-large, same as dark-orange, but with larger fonts.
• dark-green, black and green theme. The default since v0.800b.
• light-blue, a light theme with a blue tone.
• HiDPI, a version of the dark-orange theme to be used with high density (retina) screens, 4K monitors, etc.

Performance

In the Performance tab you can enable and disable multithreading. Multithreading is still an experimental feature, so use it at your own risk. This allows the program to use more than one CPUs for the processing.

Levels of Detail (LOD)

Certain big datasets use levels of detail to prevent data clutter. Refer to the Levels of Detail (LOD) section for more info.

Controls

You can see the key associations in the Controls tab. Controls are not editable. Check out the Controls documentation to know more.

Screenshot configuration

You can take screenshots anytime when the application is running by pressing F5. There are two screenshot modes available: * Simple, the classic screenshot of what is currently on screen, with the same resolution. * Advanced, where you can define the resolution of the screenshots.

Frame output

There is a feature in the Gaia Sky that enables the output of every frame as an image. This is useful to produce videos. In order to configure the frame output system, use the Frame output tab. There you can select the output folder, the image prefix name, the output image resolution (in case of Advanced mode) and the target frames per second. When the program is in frame output mode, it does not run in real time but it adjusts the internal clock to produce as many frames per second as specified here. You have to take it into account when you later use your favourite video encoder (ffmpeg) to convert the frame images into a video.

Camera recording

Here you can set the desired frames per second to capture the camera paths. If your device is not fast enough in producing the specified frame rate, the application will slow down while recording so that enough frames are captured. Same behaviour will be uploading during camera playback.

360 mode

Here you can define the cube map side resolution for the 360 mode. With this mode a cube map will be rendered (the whole scene in all direction +X, −X, +Y, −Y, +Z, −Z) and then it will be transformed into a flat image using an equirectangular projection. This allows for the creation of 360 (VR) videos.
Fig. 2.1: 360 mode in Gaia Sky
Data

As of version 1.0.0 you can use the Data tab to select the catalogue to load. Gaia Sky ships with two catalogues by default:

- **TGAS** This is based on the Tycho-Gaia Astrometric Solution ([source](#)) and contains a little over 600,000 stars. This catalogue uses levels of detail which can be configured in the Performance tab.

- **HYG** This is the Hipparcos, Gliese and Yale Bright Stars ([home page](#), [GitHub repository](#)) and contains roughly some 100,000 stars.

Gaia

Here you can choose the attitude of the satellite. You can either use the real attitude (takes a while to load but will ensure that Gaia points to where it should) and the NSL, which is an analytical implementation of the nominal attitude of the satellite. It behaves the same as the real thing, but the observation direction is not ensured.

Check for new version

You can always check for a new version by clicking on this button. By default, the application checks for a new version if more than five days have passed since the last check. If a new version is found, you will see the notice here together with a link to the download.

Do not show that again!

If you do not want this configuration dialogue to be displayed again when you launch the Gaia Sky, tick this check box and you are good to go.

2.4.2 The properties file

There is a configuration file which stores most of the properties explained in the previous section and some more. This section is devoted to these properties that are not represented in the GUI but are still configurable. The configuration file is located in $HOME/.gaiasky/global.properties. Here are some of the properties found in this file that are not represented in the GUI.

**Graphics properties**

- `graphics.render.time` - This property gets a boolean (true|false) and indicates whether a timestamp is to be added to screenshots and frames.

**Data properties**

- `data.json.catalog` - This property points to the json file where the catalog(s) to load are defined. The properties `data.json.catalog.*` contain the default catalogs which are shipped with Gaia Sky and are offered in the config dialog.

- `data.json.objects` - Contains the json file where the definition of all the rest of the data is specified. There are three child properties (`data.json.objects.gq.[0..2]`) which contain the default graphics quality options.

- `data.limit.mag` - This contains the limiting magnitude above which stars shall not be loaded.
Scene properties

- `scene.labelfactor` - A real number in $[0..n]$ that controls the number of star labels to display. The larger the number, the more stars will have a label.

- `scene.star.thresholdangle.quad` - This property contains the view angle (in degrees) boundary above which the stars are rendered as quads. Quads are basically 4-vertex quadrilaterals, and they can be rendered as textures (images) or using shaders. They display more detail but are costlier in terms of GPU processing.

- `scene.star.thresholdangle.point` - This property contains the view angle (in degrees) boundary above which the stars are rendered as points. Points are single pixels, so they are not very resource demanding.

- `scene.star.thresholdangle.none` - This property contains the view angle (in degrees) below which the stars are not rendered at all. Usually this is 0 unless you want to cull very distant stars.

- `scene.point.alpha.min` - Contains the minimum alpha value (opacity) in $[0..1]$ for the stars rendered as points. This should in any case be lower than `scene.point.alpha.max`.

- `scene.point.alpha.max` - Contains the maximum alpha value (opacity) in $[0..1]$ for the stars rendered as points. This should in any case be greater than `scene.point.alpha.min`.

- `scene.galaxy.3d` - Contains a boolean. If set to true, the Milky Way will be rendered using a blending of a 2D image with a 3D distribution of stars and nebulae. Otherwise, only the 2D image is used.

Program wide properties

- `program.tutorial` - This gets a boolean (`true|false`) indicating whether the tutorial script should be automatically run at start up.

- `program.tutorial.script` - This points to the tutorial script file.

- `program.debuginfo` - If this property is set to true, some debug information will be shown at the top right of the window. This contains information such as the number of stars rendered as a quad, the number of stars rendered as a point or the frames per second. This can be activated in real time by pressing `CTRL + D`.

- `program.ui.theme` - Specifies the GUI theme. Three themes are available: `bright-blue`, `dark-orange`, `dark-green`, `dark-orange-large` and `HiDPI`. If you have a HiDPI screen (retina, 4K monitor) with a large dots per inch (DPI) number, you should use the HiDPI theme. Since version 0.704b you can also choose the theme by using the `User Interface tab` in the `Preferences dialog`.

2.4.3 Graphics settings

Most of the graphics settings can be adjusted using the `Preferences dialog`.

Resolution and mode

You can find the `Resolution and mode` configuration under the `Graphics tab`. There you can switch between full screen mode and windowed mode. In the case of full screen, you can choose the resolution from a list of supported resolutions in a drop down menu. If you choose windowed mode, you can enter the resolution you want. You can also choose whether the window should be resizable or not. In order to switch from full screen mode to windowed mode during the execution, use the key `F11`. 
Graphics quality

This setting governs the size of the textures, the complexity of the models and also the quality of the graphical effects (light glow, lens flare, etc.). Here are the differences:

- **High** Contains some high-resolution textures (4K) and specular and normal maps for most celestial bodies. Planets and moons have a high vertex count. Graphical effects use a large number of samples to get the best visuals.

- **Normal** contains lower resolution textures (2K when available) and some specular and normal maps are deactivated. The graphical effects use a reasonable amount of quality for nice visuals without compromising the performance too much.

- **Low** Offers a noticeable performance gain on less powerful systems. Same textures and model quality as in the Normal setting. The volumetric light effect is turned off completely and the lens flare effect uses a low number of ghosts.

Antialiasing

In the Graphics tab you can also find the antialiasing configuration. Applying antialiasing removes the jagged edges of the scene and makes it look better. However, it does not come free of cost, and usually has a penalty on the frames per second (FPS). There are four main options, described below.

Find more information on antialiasing in the Antialiasing section.

No Antialiasing

If you choose this no antialiasing will be applied, and therefore you will probably see jagged edges around models. This has no penalty on either the CPU or the GPU. If you want to enable antialiasing with override application settings in your graphics card driver configuration program, you can leave the application antialiasing setting to off.

FXAA - Fast Approximate Antialiasing

This is a post-processing antialiasing which is very fast and produces reasonably good results. It has some impact on the FPS depending on how fast your graphics card is. As it is a post-processing effect, this will work also when you take screenshots or output the frames. Here is more info on FXAA.

NFAA - Normal Field Antialiasing

This is yet another post-processing antialiasing technique. It is based on generating a normal map to detect the edges for later smoothing. It may look better on some devices and the penalty in FPS is small. It will also work for the screenshots and frame outputs.

MSAA - Multi-Sample Antialiasing

As of version 1.0.1 MSAA is not offered anymore. This is implemented by the graphics card and may not always be available. You can choose the number of samples (from 2 to 16, from worse to better) and it has a bigger cost on FPS than the post-processing options. It also looks better. However, this being reliant on a special multisample frame buffer in the graphics card makes it not available for screenshots and frame outputs.

Line style

Whether to render lines with an advanced quad system or using simple GL_LINES. The former will look better at the expense of requiring more processing power in the GPU.
Vertical synchronization (V-sync)

This option limits the frames per second to match your monitor’s refresh rate and prevent screen tearing. It is recommended to leave it enabled unless you want to test how many FPS you can get or you want to fry your card.

2.4.4 Graphics performance

The Gaia Sky uses OpenGL to render advanced graphics and thus its performance may be affected significantly by your graphics card. Below you can find some tips to improve the performance of the application by tweaking or deactivating some graphical effects.

Graphics quality setting

Please see the Graphics quality section.

Antialiasing

Antialiasing is a term to refer to a number of techniques for reducing jagged edges, stairstep-like lines that should be smooth. It reduces the jagged appearance of lines and edges, but it also makes the image smoother. The result are generally better looking images, even though this depends on the resolution display device.

There are several groups of antialiasing techniques, some of them implemented in the Gaia Sky and available for you to choose from the [[preferences dialog|Configuration-interface]]. They all come at a cost, which may vary depending on your system.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Antialiasing</td>
<td>No antialiasing</td>
<td>This has no cost since it does not apply any antialiasing technique.</td>
</tr>
<tr>
<td>FXAA</td>
<td>Post-processing</td>
<td>This has a mild performance cost and produces reasonably good results. If you have a good graphics card, this is super-fast.</td>
</tr>
<tr>
<td>NFAA</td>
<td>Post-processing</td>
<td>Based on the creation of a normal map to identify edges, this is slightly costlier than FXAA but it may produce better results in some devices.</td>
</tr>
<tr>
<td>MSAAx2</td>
<td>MSAA</td>
<td>MSAA is implemented in the graphics card itself and comes at a greater cost than post-processing techniques since it multi-samples the scene and uses its geometry to antialias it. This version uses two samples per pixel.</td>
</tr>
<tr>
<td>MSAAx4</td>
<td>MSAA</td>
<td>Version of MSAA that uses four samples per pixel, therefore it is costlier than MSAAx2.</td>
</tr>
<tr>
<td>MSAAx8</td>
<td>MSAA</td>
<td>Version of MSAA that uses eight samples per pixel, therefore it is costlier than MSAAx4.</td>
</tr>
<tr>
<td>MSAAx16</td>
<td>MSAA</td>
<td>Version of MSAA that uses sixteen samples per pixel, therefore it is costlier than MSAAx8.</td>
</tr>
</tbody>
</table>

Note: Since version 1.0.1 the **MSAA has been removed** due to the lack of support for multisampling frame buffers in libgdx.

Here are some sample images.
<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Antialiasing</td>
<td><img src="image1" alt="Image" /></td>
</tr>
<tr>
<td>FXAA</td>
<td><img src="image2" alt="Image" /></td>
</tr>
<tr>
<td>NFAA</td>
<td><img src="image3" alt="Image" /></td>
</tr>
<tr>
<td>MSAAx2</td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td>MSAAx4</td>
<td><img src="image5" alt="Image" /></td>
</tr>
</tbody>
</table>
Some graphics drivers allow you to override the anti-aliasing settings of applications with some default configuration (usually MSAA or FXAA). You can also use this feature with the Gaia Sky.

Find more information on antialiasing in the Antialiasing section.

**Star brightness**

The star brightness setting has an effect on the graphics performance because it will cause more or less stars to be rendered as quads instead of points, which means multiplying the number of vertices to send to the GPU. Quads are basically flat polygons to which a texture is applied (in this case their appearance is controlled by a shader).

The star brightness can be increased or decreased from the Star brightness slider in the Lighting section of the interface window.

**Hint:** CTRL + D - activate the debug mode to get some information on how many stars are currently being rendered as points and quads as well as the frames per second.

**Model detail**

Some models (mainly spherical planets, planetoids, moons and asteroids) are automatically generated when the Gaia Sky is initialising and accept parameters which tell the loader how many vertices the model should have. These parameters are set in the json data files and can have an impact on devices with low-end graphics processors. Let’s see an example:

```
"model" : {
   "args" : [true],
   "type" : "sphere",
   "params" : {
      "quality" : 150,
      "diameter" : 1.0,
      "flip" : false
   },
   "texture" : {
      "base" : "data/tex/neptune.jpg",
   }
}
```

The quality parameter specifies here the number of both vertical and horizontal divisions that the sphere will have. This number is reduced in the Android port of the Gaia Sky to something between 30 and 50 depending on the object.

Additionally, some other models, such as that of the Gaia spacecraft, come from a binary model file .g3db. These models are created using a 3D modelling software and then exported to either .g3db (bin) or .g3dj (JSON) using fbx-conv. You can create your own low-resolution models and export them to the right format. Then you just need to point the json data file to the right low-res model file. The attribute’s name is model.

```
"model" : {
   "args" : [true],
   "model" : "data/models/gaia/gaia.g3db"
}
```
Bloom, lens flare and light glow

All post-processing algorithms (those algorithms that are applied to the image after it has been rendered) take a toll on the graphics card and can be disabled.

- The **bloom** is not very taxing on the GPU.
- The **lens flare** effect is a bit harder on the GPU, but most modern cards should be able to handle it with no problems.
- The **light glow** effect is far more demanding, and disabling it can result in a significant performance gain in some GPUs. It samples the image around the principal light sources using a spiral pattern and applies a light glow texture which is rather large.

To disable these post-processing effects, find the controls in the UI window, as described in the [[lighting[User-interface#lighting]] section of the [[User interface]] chapter.

Labels

Object labels or names in the Gaia Sky are rendered using a special **shader** which implements distance field fonts. This means that labels look great at all distances but it is costlier than the regular method.

The label factor basically determines the stars for which a label will be rendered if labels are active. It is a real number between 1 and 5, and it will be used to scale the **threshold point angle** (which determines the boundary between rendering as **points** or as **quads**) to select whether a label should be rendered or not.

The label is rendered if the formula below yields true.

\[
\text{viewAngle} > \frac{\text{threshold_angle_point}}{\text{label_factor}}
\]

Currently there is no GUI option for modifying the label factor, so you must directly edit the configuration file in the [[Scene properties|Configuration-files#scene-properties]] section of the [[Configuration files]] chapter.

2.4.5 User Interface

GUI window

The Gaia Sky application has an on-screen user interface designed to be easy to use. It is divided into five sections, *Time, Camera, Objects, Type visibility, Lighting* and *Gaia scan*.

**Time**

You can play and pause the simulation using the **PLAY/PAUSE** button in the **Controls** window to the left. You can also use **SPACE** to play and pause the time. You can also change time warp, which is expressed as a factor. Use **,** and **.** to divide by 2 and double the value of the time warp.

**Camera**

In the camera options pane on the left you can select the type of camera. This can also be done by using the **NUMPAD** 0–4 keys.

There are five camera modes:

- **Free mode** – The camera is not linked to any object and its velocity is exponential with respect to the distance to the origin (Sun).
• **Focus mode** – The camera is linked to a focus object and it rotates and rolls with respect to it.

• **Gaia scene** – Provides an outside view of the Gaia satellite. The camera can not be rotated or translated in this mode.

• **Spacecraft** – Take control of a spacecraft and navigate around at will.

• **Gaia FOV** – The camera simulates either of the fields of view of Gaia, or both.

For more information on the camera modes, see the *Camera modes* section.

Additionally, there are a number of sliders for you to control different parameters of the camera:

• **Field of view**: Controls the field of view angle of the camera. The bigger it is, the larger the portion of the scene represented.

• **Camera speed**: Controls the longitudinal speed of the camera.

• **Rotation speed**: Controls the transversal speed of the camera, how fast it rotates around an object.

• **Turn speed**: Controls the turning speed of the camera.

You can **lock the camera** to the focus when in focus mode. Doing so links the reference system of the camera to that of the object and thus it moves with it.

**Hint**: Lock the camera so that it stays at the same relative position to the focus object.

Finally, we can also **lock the orientation** of the camera to that of the focus so that the same transformation matrix is applied to both.

**Hint**: Lock the orientation so that the camera also rotates with the focus.

Additionally, we can also enable the **crosshair**, which will mark the currently focused object.

**Objects**

There is a list of focus objects that can be selected from the interface. When an object is selected the camera automatically centers it in the view and you can rotate around it or zoom in and out. Objects can also be selected by double-clicking on them directly in the view or by using the search box provided above the list. You can also invoke a search dialogue by pressing **CTRL+F**.

**Type visibility**

Most graphical elements can be turned off and on using these toggles. For example you can remove the stars from the display by clicking on the **stars** toggle. The object types available are the following:

• Stars

• Planets

• Moons

• Satellites, the spacecrafts

• Asteroids

• Labels, all the text labels
• Equatorial grid
• Ecliptic grid
• Galactic grid
• Orbits, the orbit lines
• Atmospheres, the atmospheres of planets
• Constellations, the constellation lines
• Boundaries, the constellation boundaries
• Milky way
• Others

By checking the **proper motion vectors** checkbox we can enable the representation of star proper motions if the currently loaded catalog provides them. Once proper motions are activated, we can control the number of displayed proper motions and their length by using the two sliders that appear.

**Lighting**

Here are a few options to control the lighting of the scene:

- **Star brightness**: Controls the brightness of stars.
- **Star size**: Controls the size of point-like stars.
- **Min. star opacity**: Sets a minimum opacity for the faintest stars.
- **Ambient light**: Controls the amount of ambient light. This only affects the models such as the planets or satellites.
- **Bloom effect**: Controls the bloom effect.
- **Brightness**: Controls the brightness of the image.
- **Contrast**: Controls the contrast of the image.
- **Motion blur**: Enable or disable the motion blur effect.
- **Lens flare**: Enable or disable the lens flare.
- **Star glow**: Enable or disable star glows. If enabled, the stars are rendered using a glow texture in a post-processing step. This can have a performance hit on some older graphics cards.

**Gaia scan**

You can also enable the real time computation of Gaia observation. To do so, tick the **Enable Gaia scan** checkbox. Keep in mind that this computation is done by interpolating the scan path and calculating what stars fall into Gaia’s both fields of view, so if you set the time pace very high it is going to take a toll on the frames per second. Also, you can choose to colour the stars observed by Gaia according to the number of observations, where purple is 1 and red is 75. To do so, tick the **Colour observed stars** checkbox. Finally, you can decide to only display the stars that have been observed by Gaia at least once. To do so, tick the **Show only observed stars** checkbox.
Music

Since version 0.800b Gaia Sky also offers a music player in its interface. By default it ships with only one *spacey* melody, but you can add your own by dropping them in the folder $HOME/.gaiasky/music.

**Hint:** Drop your mp3, ogg or wav files in the folder $HOME/.gaiasky/music and these will be available during your Gaia Sky sessions to play.

Running scripts

In order to run python scripts, click on the Run script... button at the bottom of the GUI window. A new window will pop up allowing you to select the script you want to run. Once you have selected it, the script will be checked for errors. If no errors were found, you will be notified in the box below and you’ll be able to run the script right away by clicking on the Run button. If the script contains errors, you will be notified in the box below and you will not be able to run the script until these errors are dealt with.

**Hint:** Add your own scripts to the folder $HOME/.gaiasky/scripts so that Gaia Sky can find them.

Preferences window

You can launch the preferences window any time during the execution of the program. To do so, click on the Preferences button at the bottom of the GUI window. For a detailed description of the configuration options refer to the *Configuration Instructions*.

2.4.6 Controls

This section describes the controls of the Gaia Sky.

Keyboard controls

To check the most up-to-date controls go to the Controls tab in the preferences window. Here are the default keyboard controls.

<table>
<thead>
<tr>
<th>Key(s)</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMPAD_0</td>
<td>Free camera</td>
</tr>
<tr>
<td>NUMPAD_1</td>
<td>Focus camera</td>
</tr>
<tr>
<td>NUMPAD_2</td>
<td>Gaia scene camera</td>
</tr>
<tr>
<td>NUMPAD_3</td>
<td>Spacecraft mode</td>
</tr>
<tr>
<td>NUMPAD_4</td>
<td>Gaia FoV 1 camera</td>
</tr>
<tr>
<td>NUMPAD_5</td>
<td>Gaia FoV 2 camera</td>
</tr>
<tr>
<td>NUMPAD_6</td>
<td>Gaia FoV 1 and 2 camera</td>
</tr>
<tr>
<td>L-CTRL + 3</td>
<td>360 mode</td>
</tr>
<tr>
<td>L-CTRL + L-SHIFT + UP</td>
<td>Increase star point size</td>
</tr>
<tr>
<td>L-CTRL + L-SHIFT + DOWN</td>
<td>Decrease star point size</td>
</tr>
<tr>
<td>SPACE</td>
<td>Toggle simulation play/pause</td>
</tr>
<tr>
<td>F5</td>
<td>Take screenshot</td>
</tr>
</tbody>
</table>

Continued on next page
Table 2.1 – continued from previous page

<table>
<thead>
<tr>
<th>Key(s)</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F6</td>
<td>Start/stop frame output mode</td>
</tr>
<tr>
<td>F11</td>
<td>Toggle fullscreen/windowed mode</td>
</tr>
<tr>
<td>L-CTRL + F</td>
<td>Search dialog</td>
</tr>
<tr>
<td>ESCAPE</td>
<td>Quit application</td>
</tr>
<tr>
<td>-</td>
<td>Decrease limiting magnitude</td>
</tr>
<tr>
<td>+</td>
<td>Increase limiting magnitude</td>
</tr>
<tr>
<td>,</td>
<td>Divide time warp by two</td>
</tr>
<tr>
<td>.</td>
<td>Double time warp</td>
</tr>
<tr>
<td>+</td>
<td>Reset limiting magnitude</td>
</tr>
<tr>
<td>L-SHIFT + B</td>
<td>Toggle constellation boundaries</td>
</tr>
<tr>
<td>L-SHIFT + C</td>
<td>Toggle constellation lines</td>
</tr>
<tr>
<td>L-SHIFT + E</td>
<td>Toggle ecliptic grid</td>
</tr>
<tr>
<td>L-SHIFT + G</td>
<td>Toggle galactic grid</td>
</tr>
<tr>
<td>L-SHIFT + L</td>
<td>Toggle labels</td>
</tr>
<tr>
<td>L-SHIFT + M</td>
<td>Toggle moons</td>
</tr>
<tr>
<td>L-SHIFT + O</td>
<td>Toggle orbits</td>
</tr>
<tr>
<td>L-SHIFT + P</td>
<td>Toggle planets</td>
</tr>
<tr>
<td>L-SHIFT + Q</td>
<td>Toggle equatorial grid</td>
</tr>
<tr>
<td>L-SHIFT + S</td>
<td>Toggle stars</td>
</tr>
<tr>
<td>L-SHIFT + T</td>
<td>Toggle satellites</td>
</tr>
<tr>
<td>L-SHIFT + U</td>
<td>Expand/collapse controls window</td>
</tr>
<tr>
<td>L-CTRL + U</td>
<td>Toggle UI completely (hide/show user interface)</td>
</tr>
<tr>
<td>L-CTRL + D</td>
<td>Toggle debug info</td>
</tr>
<tr>
<td>L-CTRL + S</td>
<td>Toggle stereoscopic mode</td>
</tr>
<tr>
<td>L-CTRL + L-SHIFT + S</td>
<td>Switch between stereoscopic profiles</td>
</tr>
</tbody>
</table>

Mouse controls

Here are the default mouse controls for the focus and free Camera modes. The other modes do not have mouse controls.

Focus mode

<table>
<thead>
<tr>
<th>Mouse + keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-MOUSE DOUBLE CLICK</td>
<td>Select focus object</td>
</tr>
<tr>
<td>L-MOUSE SINGLE CLICK</td>
<td>Stop all rotation and translation movement</td>
</tr>
<tr>
<td>L-MOUSE + DRAG</td>
<td>Apply rotation around focus</td>
</tr>
<tr>
<td>L-MOUSE + L-SHIFT + DRAG</td>
<td>Camera roll</td>
</tr>
<tr>
<td>R-MOUSE + DRAG</td>
<td>Pan view freely from focus</td>
</tr>
<tr>
<td>M-MOUSE + DRAG or WHEEL</td>
<td>Move towards/away from focus</td>
</tr>
</tbody>
</table>
Free mode

<table>
<thead>
<tr>
<th>Mouse + keys</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-MOUSE DOUBLE CLICK</td>
<td>Select object as focus (changes to focus mode)</td>
</tr>
<tr>
<td>L-MOUSE SINGLE CLICK</td>
<td>Stop all rotation and translation movement</td>
</tr>
<tr>
<td>L-MOUSE + DRAG</td>
<td>Pan view</td>
</tr>
<tr>
<td>L-MOUSE + L-SHIFT + DRAG</td>
<td>Camera roll</td>
</tr>
<tr>
<td>M-MOUSE + DRAG or WHEEL</td>
<td>Forward/backward movement</td>
</tr>
</tbody>
</table>

Gamepad controls

As of version 0.704b the Gaia Sky supports basic gamepad/controller input. By default, only the Xbox 360 controller is supported, so you may want to map your gamepad differently using software like jtest-gtk –here– on Linux or the Universal Joystick Remapper for Windows. You can also opt for a Joystick-to-keyboard solution such as xpadder –here–, even though this has not been tested with Gaia Sky.

![Xbox 360 controller button configuration](image)

Fig. 2.2: Xbox 360 controller button configuration

The actions mapped to each button or axis depend on the current camera mode (focus, free, spacecraft):
## Focus mode

<table>
<thead>
<tr>
<th>Axis/button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left bumper</td>
<td>Hold to apply 0.5 factor to speed</td>
</tr>
<tr>
<td>Right bumper</td>
<td>Hold to apply 0.1 factor to speed</td>
</tr>
<tr>
<td>Left trigger</td>
<td>Move away from focus</td>
</tr>
<tr>
<td>Right trigger</td>
<td>Move towards focus</td>
</tr>
<tr>
<td>Left stick horizontal</td>
<td>Horizontal rotation around focus</td>
</tr>
<tr>
<td>Left stick vertical</td>
<td>Vertical rotation around focus</td>
</tr>
<tr>
<td>Right stick horizontal</td>
<td>Roll right and left</td>
</tr>
<tr>
<td>Right stick vertical</td>
<td>Move towards or away from focus</td>
</tr>
<tr>
<td>D-pad</td>
<td>None</td>
</tr>
<tr>
<td>A</td>
<td>None</td>
</tr>
<tr>
<td>B</td>
<td>None</td>
</tr>
<tr>
<td>X</td>
<td>None</td>
</tr>
<tr>
<td>Y</td>
<td>None</td>
</tr>
</tbody>
</table>

## Free camera mode

<table>
<thead>
<tr>
<th>Axis/button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left bumper</td>
<td>Hold to apply 0.5 factor to speed</td>
</tr>
<tr>
<td>Right bumper</td>
<td>Hold to apply 0.1 factor to speed</td>
</tr>
<tr>
<td>Left trigger</td>
<td>Move away from focus</td>
</tr>
<tr>
<td>Right trigger</td>
<td>Move towards focus</td>
</tr>
<tr>
<td>Left stick horizontal</td>
<td>Yaw right and left</td>
</tr>
<tr>
<td>Left stick vertical</td>
<td>Pitch up and down</td>
</tr>
<tr>
<td>Right stick horizontal</td>
<td>Move sideways</td>
</tr>
<tr>
<td>Right stick vertical</td>
<td>Move forward and backward</td>
</tr>
<tr>
<td>D-pad</td>
<td>None</td>
</tr>
<tr>
<td>A</td>
<td>None</td>
</tr>
<tr>
<td>B</td>
<td>None</td>
</tr>
<tr>
<td>X</td>
<td>None</td>
</tr>
<tr>
<td>Y</td>
<td>None</td>
</tr>
</tbody>
</table>

## Spacecraft mode

<table>
<thead>
<tr>
<th>Axis/button</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left bumper</td>
<td>Stabilise spacecraft rotations</td>
</tr>
<tr>
<td>Right bumper</td>
<td>Stop spacecraft</td>
</tr>
<tr>
<td>Left trigger</td>
<td>Apply backward thrust</td>
</tr>
<tr>
<td>Right trigger</td>
<td>Apply forward thrust</td>
</tr>
<tr>
<td>Left stick horizontal</td>
<td>Yaw right and left</td>
</tr>
<tr>
<td>Left stick vertical</td>
<td>Pitch up and down</td>
</tr>
<tr>
<td>Right stick horizontal</td>
<td>Roll right and left</td>
</tr>
<tr>
<td>Right stick vertical</td>
<td>None</td>
</tr>
<tr>
<td>D-pad</td>
<td>None</td>
</tr>
<tr>
<td>A</td>
<td>Decrease engine power</td>
</tr>
<tr>
<td>B</td>
<td>None</td>
</tr>
<tr>
<td>X</td>
<td>Increase engine power</td>
</tr>
<tr>
<td>Y</td>
<td>None</td>
</tr>
</tbody>
</table>
**Touch controls**

No mobile version yet.

### 2.4.7 Camera modes

Gaia Sky offers five basic camera modes.

**Focus mode**

This is the default mode. In this mode the camera movement is locked to a focus object, which can be selected by double clicking or by using the find dialog (Ctrl+F).

- *Mouse controls in focus mode*
- *Gamepad controls in focus mode*

**Hint:** NUMPAD_0 – Enter the Focus mode

**Free mode**

This mode does not lock the camera to a focus object but it lets it roam free in space.

- *Mouse controls in free mode*
- *Gamepad controls in free mode*

**Hint:** NUMPAD_1 – Enter the Free mode

**Gaia scene mode**

In this mode the camera can not be controlled. It provides a view of the Gaia satellite from the outside.

**Hint:** NUMPAD_2 – Enter the Gaia scene mode

**Spacecraft mode**

In this mode you take control of a spacecraft. In the spacecraft mode, the GUI changes completely. The Options window disappears and a new user interface is shown in its place at the bottom left of the screen.

- *Attitude indicator* – It is shown as a ball with the horizon and other marks. It represents the current orientation of the spacecraft with respect to the equatorial system.
  - Indicates the direction the spacecraft is currently headed to.
• Indicates direction of the current velocity vector, if any.

• Indicates inverse direction of the current velocity vector, if any.

• **Engine Power** – Current power of the engine. It is a multiplier in steps of powers of ten. Low engine power levels allow for Solar System or planetary travel, whereas high engine power levels are suitable for galactic and intergalactic exploration.

• Stabilises the yaw, pitch and roll angles. If rotation is applied during the stabilisation, the stabilisation is cancelled.

• Stops the spacecraft until its velocity with respect to the Sun is 0. If thrust is applied during the stopping, the stopping is cancelled.

• Return to the focus mode.

• **Gamepad controls in spacecraft mode**

**Hint:** NUMPAD_3 – Enter the Spacecraft mode

Fig. 2.3: Spacecraft mode, with the various controls at the bottom left.
Field of View mode

This mode simulates the Gaia fields of view. You can select FoV1, FoV2 or both.

**Hint:** NUMPAD_4 – Enter Field of View 1 mode
NUMPAD_5 – Enter Field of View 2 mode
NUMPAD_6 – Enter Field of View 1 and 2 mode

2.4.8 Stereoscopic (3D) mode

Gaia Sky includes a stereoscopic mode or 3D mode which outputs two images each intended for each eye, creating the illusion of depth.

**Hint:** LEFT_CTRL + S – Activate the stereoscopic mode
LEFT_CTRL + LEFT_SHIFT + S – Switch between 3D profiles

Stereoscopic profiles

Usually, as the images are placed side by side (even though most 3DTVs also support up and down), the right image is intended for the right eye and the left image is intended for the left eye. This works with 3DTVs and VR head sets (such as the Oculus Rift, Google cardboard, etc.). In 3DTVs, however, the image is distorted because each half of the TV will be stretched back to the whole TV area when the 3D mode is on.

Additionally, there is a technique called cross-eye 3D (you can find some examples here, and here is a very nice video teaching the concept and how to achieve it) which works without any extra equipment and consists on trying to focus your eyes some distance before the actual image so that each eye receives the correct image. In this case the right images goes to the left eye and the left image goes to the right eye.

In order to manage all these parameters, we have created 4 stereoscopic profiles which can be selected by the user and are described below.

- **VR_HEADSET** – The left image goes to the left eye. No distortion is applied.
- **Crosseye** – The left image goes to the right eye. No distortion is applied.
- **3DTV** – The left image goes to the left eye. Distortion is applied.
- **Anaglyph 3D** – To use with red-cyan glasses. Displays both the left and right images at full resolution. Left image contains the red channel, right image contains the green and blue channels.

**Hint:** L-CTRL + L-SHIFT + S – Switch between 3D profiles
<table>
<thead>
<tr>
<th>Profile</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>VR_HEADSET</td>
<td><img src="image1" alt="VR_HEADSET Image" /></td>
</tr>
<tr>
<td>Crosseye</td>
<td><img src="image2" alt="Crosseye Image" /></td>
</tr>
<tr>
<td>DTV</td>
<td><img src="image3" alt="DTV Image" /></td>
</tr>
</tbody>
</table>
2.4.9 Planetarium mode

Gaia Sky includes a planetarium mode which is specially useful to capture videos to be displayed in a full dome. To get the best results, the screen resolution (or the screenshots resolution) should have an aspect ratio of 1:1. In the planetarium mode a few things happen:

- **Fisheye transform** – A fisheye transform is applied to the source image.
- **Field of view** – The field of view is increased to about 180 degrees.
- **View skew** – In focus mode, the view is skewed about 50 degrees downwards.

**Hint:** To activate the planetarium mode, click on the icon in the camera section of the controls window. Exit by clicking again.

2.4.10 360 mode

Gaia Sky includes a 360 mode where the scene is rendered in all directions to a cube map. This cube map is then projected onto a flat image using an equirectangular projection. The final image can be used to create 360 videos with head tracking (see here)

**Hint:** To activate the 360 mode, click on the icon in the camera section of the controls window. Exit by clicking again.

**Hint:** L-CTRL + 3 – Can also be used to toggle the 360 mode.

Configuration

Please, see the 360 mode section.

Creating panorama images

In order to create panorama images that can be viewed with a VR device or simply a 360 viewer, we need to take into consideration a few points.

- Panoramas work best if their **aspect ratio is 2:1**, so a resolution of 5300x2650 or similar should work. (Refer to the Screenshot configuration section to learn how to take screenshots with an arbitrary resolution).

- Some services (like Google) have strong constraints on image properties. For instance, they must be at least 14 megapixels and in jpeg format. Learn more here.

- Some **metadata** needs to be injected into the image file.
Fig. 2.4: Planetarium mode
Injecting panorama metadata to 360 images

To do so, we can use ExifTool. We will inject the metadata with the following command:

```
exiftool -ProjectionType="equirectangular" -UsePanoramaViewer="True" -
"PoseHeadingDegrees<$exif:GPSImgDirection" -"CroppedAreaImageWidthPixels<$ImageWidth
-"CroppedAreaImageHeightPixels<$ImageHeight" -"FullPanoWidthPixels<$ImageWidth -
"FullPanoHeightPixels<$ImageHeight" -CroppedAreaLeftPixels="0" -
-CroppedAreaTopPixels="0" image_name.jpg
```

Now we can enjoy our image in any 360 panorama viewer like Google Street View app or the Cardboard Camera! Find some examples in this album.

Fig. 2.5: Panorama image captured with Gaia Sky

2.4.11 Recording and playing camera paths

Gaia Sky offers the possibility to record camera paths out of the box in real time and later play them. These camera paths go to a text file in the temp folder of your system.

Camera path file format

The format of the file is pretty straightforward. It consists of a csv file with white spaces as delimiters, each row containing the state of the camera and the time. The state of the camera consists of 9 double-precision floating point numbers, 3 for the position and 3 for the direction vector and 3 for the up vector.

The reference system used is explained in the Internal reference system section. The units are $1 \times 10^{-9} m$.

The format of each row is as follows:

- long - Time as defined by the getTime() function of java.util.Date (here).
- double x3 - Position of the camera.
• double x3 - Direction vector of the camera.
• double x3 - Up vector of the camera.

Recording camera paths

In order to start recording the camera path, click on the REC button next to the Camera section title in the GUI Controls window. The REC button will turn red, which indicates the camera is being recorded.

![Camera path REC and PLAY buttons](image)

Fig. 2.6: Camera path REC and PLAY buttons

In order to stop the recording and write the file, click again on the red REC button. The button will turn grey and a notification will pop up indicating the location of the camera file. Camera files are by default saved in the $HOME/.gaiasky/camera directory.

Playing camera paths

In order to play a camera file, click on the folder icon next to the REC icon. This will prompt a list of available camera files in the $HOME/.gaiasky/camera folder.

2.4.12 Performance

The performance of the application may vary significantly depending on the characteristics of your system. This chapter describes what are the factors that have an impact in a greater or lesser degree in the performance of the Gaia Sky and explains how to tweak them. It is organised in two parts, namely GPU performance (graphics performance) and CPU performance.

Graphics performance

Refer to the Graphics performance chapter.

CPU performance

The CPU also plays an obvious role in updating the scene state (positions, orientations, etc.), managing the input and events, executing the scripts and calling and running the rendering subsystem, which streams all the texturing and geometrical information to the GPU for rendering. This section describes what are the elements that can cause a major impact in CPU performance and explains how to tune them.

Limiting magnitude

You can modify the magnitude limit by setting the property data.limit.mag in the configuration file. This will prevent the loading of stars whose magnitude is higher (they are fainter) than the specified magnitude, thus relieving the CPU of some processing. Also, take a look at the Data properties section.
Levels of Detail (LOD)

These settings apply only when using a catalog with levels of detail like TGAS. We can configure whether we want smooth transitions between the levels (fade-outs and fade-ins) and also the draw distance, which is represented by a range slider. The left knob represents the view angle above which octants are rendered.

![Levels of detail (LOD)](image)

Fig. 2.7: Levels of detail slider in preferences dialog

The right knob only matters if Smooth LOD transitions is checked and sets a higher boundary for the angle for the fade-out and fade-in of octant particles.

- Set the knobs to the right to lower the draw distance and increase performance.
- Set the knobs to the left to higher the draw distance at the expense of performance.

![Octree and levels of detail](image)

Fig. 2.8: Octree and levels of detail. Image: Wikipedia.
2.4.13 **Internal reference system**

The internal cartesian reference system is described as follows: \( XZ \) is the equatorial plane. \( Z \) points towards the vernal equinox point, Aries \( \gamma \). \( Y \) points towards the north celestial pole. \( X \) is perpendicular to both \( Z \) and \( Y \).

![Gaia Sky reference system](image)

Fig. 2.9: Gaia Sky reference system

All the positions and orientations of the entities in the scene are at some point converted to this reference system for representation. The same happens with the orientation sensor data in mobile devices.

2.4.14 **Data: catalogues and formats**

Gaia Sky needs to first load data in order to display it. The internal structure of these data is a *scene graph*, which is basically a *tree with nodes*. The objects that are displayed in a scene are all nodes in this scene graph and are organized in a hierarchical manner depending on their geometrical and spatial relations.

**Hint:** The data nodes in the scene graph are of multiple natures and are *loaded differently depending on their type*. Here we can make the first big distinction in the data nodes depending on where they come from.

The different types of data are:

- **Particle data** – usually stars which come from a star catalogue.
• **Rest of data** – planets, orbits, constellations, grids and everything else qualifies for this category.

Data belonging to either group will be loaded differently into the Gaia Sky. The sections below describe the data format in detail:

### General information on the data loading mechanisms

Gaia Sky uses a flexible data loading mechanism where the correspondences between data loader and files are defined in a couple of JSON files which are specified in the `global.properties` configuration file in `$HOME/.gaiasky/`. The two main files are the catalog file (usually `data/catalog-*.json`) and the objects file (`data/data-low.json`, `data/data-normal.json` and `data/data-high.json` are the default options, which provide the low, normal and high graphics quality settings). See the `properties file` section for more information on this.

#### catalog-*.json example files

```json
{
  "data": [
    {
      "loader": "gaia.cu9.ari.gaiaorbit.data.stars.HYGBinaryLoader",
      "files": [ "data/hygxyz.bin" ]
    }
  ]
}
```

```json
{
  "data": [
    {
      "loader": "gaia.cu9.ari.gaiaorbit.data.stars.OctreeCatalogLoader",
      "files": [ "data/octree/tgas_final_particles.bin", "data/octree/tgas_final_metadata.bin" ]
    }
  ]
}
```

#### data-*.json example file

```json
{
  "data": [
    {
      "loader": "gaia.cu9.ari.gaiaorbit.data.JsonLoader",
    },
    {
      "loader": "gaia.cu9.ari.gaiaorbit.data.constel.ConstellationsLoader",
      "files": [ "data/constel_hip.csv" ]
    }
  ]
}
```
As you see the format in both files is based on specifying Java "loader" classes that will load the list of files under the "files" property. The format should be pretty self-explanatory, but here are some rules:

- The “data” property contains a list of Java classes that implement the ISceneGraphLoader – here interface. Each one of these will load a different kind of data; the JSONLoader – here– loads non-catalog data (planets, satellites, orbits, etc.), the STILCatalogLoader –here– loads VOTables, FITS, CSV and other files through the STIL library, ConstellationsLoader –here– and ConstellationsBoundariesLoader –here– load constellation data and constellation boundary data respectively and so on.

- Then, for each one of these data loaders a list of files is defined. This list will be passed to the loader, which will try to load these files and add them to the scene graph.

### Particle data loading

There are several off-the-shelf options to get data in various formats into Gaia Sky. These options can be organized into two main categories:

- **Local data** – The data to load are files in the local disk.

- **Object server** – The data will be loaded/streamed from an object server in the local machine or over the network.

### Local data

In order to load local data there are a series of default options which can be combined. As described in the [[General data loading|General-information-on-the-data-loading-mechanisms]] section, multiple catalogue loaders can be used at once. Each catalog loader will get a list of files to load. A description of the main local catalog loaders follows.
HYG catalog loaders

These loaders (HYGBinaryLoader –here– and HYGCSVLoader –here–) load the HYG catalog that comes bundled with the Gaia Sky, which may be in csv format or in an arbitrary (not standard) binary –bin– format. Even though they have the HYG– prefix, these can load any file in the same format. The csv and bin formats are described below.

- **CSV format**: This is the csv format as downloaded from the HYG Database site. The first line contains the headers and is skipped. Then, each following row contains a particle (star) with the following columns:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Optional</th>
<th>Ignored</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star ID (pk)</td>
<td>long</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Hipparcos catalog id</td>
<td>long</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Henry Draper catalog id</td>
<td>long</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Harvard Revised catalog id</td>
<td>long</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Gliese catalog id</td>
<td>string</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Bayer / Flamsteed designation</td>
<td>string</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Proper name</td>
<td>string</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Right ascension</td>
<td>float [deg]</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Declination</td>
<td>float [deg]</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Distance</td>
<td>float [pc]</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Magnitude</td>
<td>float [mag]</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Absolute magnitude</td>
<td>float [mag]</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Spectrum type</td>
<td>string</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Color index</td>
<td>float</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

- **BIN format**: The binary format is described in the class comment of HYGBinaryLoader –here–. The meaning of each single bit in this format is described below:

  - 32 bits (int) – The number of stars in the file, starNum repeat the following starNum times (for each star)
– 32 bits (int) – The the length of the name, or nameLength
– 16 bits * “nameLength” (chars) – The name of the star
– 32 bits (float) – Apparent magnitude
– 32 bits (float) – Absolute magnitude
– 32 bits (float) – Color index B-V
– 32 bits (float) – Right ascension [deg]
– 32 bits (float) – Declination [deg]
– 32 bits (float) – Distance [pc * 3.0856775204864006E7]
– 64 bits (long) – Star identifier

There is a utility to convert the csv catalog to the bin format. It is called HYGToBinary –here– and it can easily be adapted to convert any supported format to this binary format.

**Octree catalog loader**

This is practically the same format as the binary in the HYGBinaryLoader but adding some metadata to construct an octree in order to cull portions of the catalog that are not visible and to implement a level-of-detail system to reduce the amount of particles in the viewport. This loader needs two files, the particles file and the metadata file. Both files are binary files and their description is below.

• **Particles file**: The actual reading and writing of the particles file is done in the ParticleDataBinaryIO –here–. The format is exactly the same as in the HYG bin format but adding two extra attributes to each star which indicate the pageId (the identifier of the octant) and the particleType, an integer code indicating whether it is a real star or a virtual particle created for a higher LoD (level of detail).

  – 32 bits (int) – The number of stars in the file, starNum repeat the following starNum times (for each star)
  – 32 bits (int) – The the length of the name, or nameLength
  – 16 bits * ‘nameLength’ (chars) – The name of the star
  – 32 bits (float) – Apparent magnitude
  – 32 bits (float) – Absolute magnitude
  – 32 bits (float) – Color index B-V
  – 32 bits (float) – Right ascension [deg]
  – 32 bits (float) – Declination [deg]
  – 32 bits (float) – Distance [pc * 3.0856775204864006E7]
  – 64 bits (long) – Star identifier
  – 64 bits (long) – Page id
  – 32 bits (int) – Particle type

• **Metadata file**: This file contains the information of the Octree, its nodes -octants- and the particles each node contains. The reading and writing is handled by the MetadataBinaryIO –here–. The format is as follows:

  – 32 bits (int) with the number of nodes, nNodes repeat the following nNodes times (for each node)
  – 64 bits (long) – pageId - The page id
  – 64 bits (double) – centreX - The x component of the centre
– 64 bits (double) – centreY - The y component of the centre
– 64 bits (double) – centreZ - The z component of the centre
– 64 bits (double) – sx - The size in x
– 64 bits (double) – sy - The size in y
– 64 bits (double) – sz - The size in z
– 64 bits * 8 (long) – childrenIds - 8 longs with the ids of the children. If no child in the given position, the id is negative.
– 32 bits (int) – depth - The depth of the node
– 32 bits (int) – nObjects - The number of objects of this node and its descendants
– 32 bits (int) – ownObjects - The number of objects of this node
– 32 bits (int) – childCount - The number of children nodes

In order to produce these files from a catalog, one needs to OctreeGenerator –here–. This class will get a list of stars and will produce the Octree according to certain parameters. The class OctreeGeneratorTest –here– may be used to read a catalog from a file, generate the octree and write both the particles and the metadata files back to a file.

**STIL catalog loader**

As of version v0.704 the Gaia Sky supports all formats supported by the STIL library. Since the data held by the formats supported by STIL is not of a unique nature, this catalog loader makes a series of assumptions:

- Positional information exists in the source file (spherical/cartesian equatorial/galactic coordinates are accepted, corresponding to the ucds pos.eq.* and pos.galactic.*, where the * can be ra, dec, glat, glon, x, y and z).
- Apparent magnitude data in at least one filter exists (phot.mag;em.opt.*, where * can be V, B, I or R).
- Absolute magnitude data is not required but always welcome (phys.magAbs;em.opt.*).
- B-V color index is present (corresponding to the ucd phot.color;em.opt.B;em.opt.V). More colors will be supported soon.
- If meta.id and/or meta.id;meta.main are present, they are used as name and identifier of the stars respectively. Otherwise, a random name and identifier are generated and assigned.

**Object server**

Not implemented yet.

**Non-particle data: Planets, Moons, Asteroids, etc.**

Most of the entities and celestial bodies that are not stars in the Gaia Sky scene are defined in a series of json files and are loaded using the JsonLoader –here–. The format is very flexible and loosely matches the underneath data model, which is a scene graph tree.
Top-level objects

All objects in the json files must have at least the following 5 properties:

- **name**: The name of the object. This will translate to the line colour in orbits, to the colour of the point for planets when they are far away and to the colour of the grid in grids.
- **color**: The colour of the object. Valid component types are Stars, Planets, Moons, Satellites, Atmospheres, Constellations, etc.
- **ct**: The ComponentType. This is basically a string that will be matched to the entity type in ComponentType enum. Valid component types are Stars, Planets, Moons, Satellites, Atmospheres, Constellations, etc.
- **impl**: The package and class name of the implementing class.
- **parent**: The name of the parent entity.

Additionally, different types of entities accept different additional parameters which are matched to the model using reflection. Here are some examples of these parameters:

- **size**: The size of the entity, usually the radius in km.
- **appmag**: The apparent magnitude.
- **absmag**: The absolute magnitude.

Below is an example of a simple entity, the equatorial grid:

```json
{
    "name" : "Equatorial grid",
    "color" : [1.0, 0.0, 0.0, 0.5],
    "size" : 1.2e12,
    "ct" : "Equatorial",
    "parent" : "Universe",
    "impl" : "gaia.cu9.ari.gaiaorbit.scenegraph.Grid"
}
```

Planets, moons, asteroids and all rigid bodies

Planets, moons and asteroids all use the model object Planet. This provides a series of utilities that make their json specifications look similar.

Coordinates

Within the coordinates object one specifies how to get the positional data of the entity given a time. This object contains a reference to the implementation class (which must implement IBodyCoordinates) and the necessary parameters to initialize it. There are currently a bunch of implementations that can be of use:

- **OrbitLintCoordinates**: The coordinates of the object are linearly interpolated using the data of its orbit, which is defined in a separated entity. See the [[Orbits|Non-particle-data-loading#orbits]] section for more info. The name of the orbit entity must be given. For instance, the Hygieia moon uses orbit coordinates:

  ```json
  "coordinates" : {
    "impl" : "gaia.cu9.ari.gaiaorbit.util.coord.OrbitLintCoordinates",
    "orbitname" : "Hygieia orbit"
  }
  ```

- **StaticCoordinates**: For entities that never move. A position is required. For instance, the Milky Way object uses static coordinates:
• AbstractVSOP87 – Used for the major planets, these coordinates implement the VSOP87 algorithms. Only the implementation is needed. For instance, the Earth uses these coordinates.

```
json "coordinates": {
    "impl": "gaia.cu9.ari.gaiaorbit.util.coord.vsop87.EarthVSOP87"
}
```

• GaiaCoordinates – Special coordinates for Gaia.

• MoonAACoordinates – Special coordinates for the moon using the algorithm described in the book Astronomical Algorithms by Jean Meeus.

Rotation

The rotation object describes, as you may imagine, the rigid rotation of the body in question. A rotation is described by the following parameters:

- **period** – The rotation period in hours.
- **axialtilt** – The axial tilt is the angle between the equatorial plane of the body and its orbital plane. In degrees.
- **inclination** – The inclination is the angle between the orbital plane and the ecliptic. In degrees.
- **ascendingnode** – The ascending node in degrees.
- **meridianangle** – The meridian angle in degrees.

For instance, the rotation of Mars:

```
"rotation": {
    // In hours
    "period": 24.622962156,
    // Angle between equatorial plane and orbital plane
    "axialtilt": 25.19,
    // Inclination of orbit plane with respect to ecliptic
    "inclination": 1.850,
    "ascendingnode": 47.68143,
    "meridianangle": 176.630
}
```

Model

This object describes the model which must be used to represent the entity. Models can have two origins: - They may come from a 3D model file. In this case, you just need to specify the file.

```
json "model": {
    "args": [true],
    "model": "data/models/gaia/gaia.g3db"
}
```
• They may be generated on the fly. In this case, you need to specify the type of model, a series of parameters and the texture or textures.

```json
"model": {
  "args": [true],
  "type": "sphere",
  "params": {
    "quality": 180,
    "diameter": 1.0,
    "flip": false
  },
  "texture": {
    "base": "data/tex/earth.jpg",
    "specular": "data/tex/earth-specular.jpg",
    "normal": "data/tex/earth-normal-4k.jpg",
    "night": "data/tex/earth-night-2k.jpg"
  }
}
```

• **type** – The type of model. Possible values are *sphere*, *disc*, *cylinder* and *ring*.

• **params** – Parameters of the model. This depends on the type. The **quality** is the number of both horizontal and vertical divisions. The **diameter** is the diameter of the model and **flip** indicates whether the normals should be flipped to face outwards. The **ring** type also accepts **innerradius** and **outerradius**.

• **texture** – Indicates the texture or textures to apply. The **base** texture is the one applied in normal conditions. The **specular** is the specular map to produce specular reflections. The **normal** is a normal map to produce extra detail in the lighting. The **night** is the texture applied to the part of the model in the shade.

### Atmosphere

Planet atmospheres can also be defined using this object. The **atmosphere** object gets a number of physical quantities that are fed in the atmospheric scattering algorithm (Sean O’Neil, GPU Gems).

```json
"atmosphere": {
  "size": 6600.0,
  "wavelengths": [0.650, 0.570, 0.475],
  "m_Kr": 0.0025,
  "m_Km": 0.001,

  "params": {
    "quality": 180,
    // Atmosphere diameters are always 2
    "diameter": 2.0,
    "flip": true
  }
}
```

### Orbits

When we talk about orbits in this context we talk about orbit lines. In the Gaia Sky orbit lines may be created from two different sources. The sources are used by a class implementing the `IOrbitDataProvider` –here– interface, which is also specified in their **orbit** object.

• **An orbit data file.** In this case, the orbit data provider is `OrbitFileDataProvider`.

42 Chapter 2. Contents
• The orbital elements, where the orbit data provider is `OrbitalParametersProvider`.

If the orbit is pre-sampled it comes from an orbit data file. In the Gaia Sky the orbits of all major planets are pre-sampled, as well as the orbit of Gaia. For instance, the orbit of Venus.

```json
{
   "name" : "Venus orbit",
   "color" : [1.0, 1.0, 1.0, 0.55],
   "ct" : "Orbits",
   "parent" : "Sol",
   "impl" : "gaia.cu9.ari.gaiaorbit.scenegraph.Orbit",
   "provider" : "gaia.cu9.ari.gaiaorbit.data.orbit.OrbitFileDataProvider",
   "orbit" : {
      "source" : "data/orb.VENUS.dat",
   }
}
```

If you prefer to define the orbit using the orbital elements, you need to specify these parameters in the `orbit` object. For example, the orbit of Phobos.

```json
{
   "name" : "Phobos orbit",
   "color" : [0.7, 0.7, 1.0, 0.4],
   "ct" : "Orbits",
   "parent" : "Mars",
   "impl" : "gaia.cu9.ari.gaiaorbit.scenegraph.Orbit",
   "provider" : "gaia.cu9.ari.gaiaorbit.data.orbit.OrbitalParametersProvider",
   "orbit" : {
      "period" : 0.31891023,
      "epoch" : 2455198,
      "semimajoraxis" : 9377.2,
      "eccentricity" : 0.0151,
      "inclination" : 1.082,
      "ascendingnode" : 16.946,
      "argofpericenter" : 157.116,
      "meananomaly" : 241.138
   }
}
```

**Grids and other special objects**

There are a last family of objects which do not fall in any of the previous categories. These are grids and other objects such as the Milky Way (inner and outer parts). These objects usually have a special implementation and specific parameters, so they are a good example of how to implement new objects.

```json
{
   "name" : "Galactic grid",
   "color" : [0.3, 0.5, 1.0, 0.5],
   "size" : 1.4e12,
   "ct" : Galactic,
}
```
For example, the grids accept a parameter `transformName`, which specifies the geometric transform to use. In the case of the galactic grid, we need to use the `equatorialToGalactic` transform to have the grid correctly positioned in the celestial sphere.

### Creating your own catalogue loaders

In order to create a loader for your catalogue, one only needs to provide an implementation to the `ISceneGraphLoader` interface.

```java
public interface ISceneGraphLoader {
    public List<? extends SceneGraphNode> loadData() throws FileNotFoundException;
    public void initialize(String[] files) throws RuntimeException;
}
```

The main method to implement is `List<? extends SceneGraphNode> loadData()` - which must return a list of elements that extend `SceneGraphNode`, usually `Stars`.

But how do we know which file to load? You need to create a `catalog-*\.json` file, add your loader there and create the properties you desire. Usually, there is a property called `files` which contains a list of files to load. Once you've done that, implement the `initialize(String[])` - method knowing that all the properties defined in the `catalog-*\.json` file with your catalogue loader as a prefix will be passed in the `Properties p` object without prefix.

Also, you will need to connect this new catalog file with the Gaia Sky configuration so that it is loaded at startup. To do so, locate your `global.properties` file (usually under `$HOME/.gaiasky/`) and update the property `data.json.catalog` with your catalog json file.

Add your implementing jar file to the classpath and you are good to go.

Take a look at already implemented catalogue loaders such as the `OctreeCatalogLoader` - to see how it works.

### Loading data using scripts

Data can also be loaded at any time from a Python script.

TODO

#### 2.4.15 Scripting

Gaia Sky offers the possibility to run Python scripts in the same JVM using Jython.

You can find some example scripts in the `scripts` folder of the project.

**Hint:** Add your own scripts to the folder `$HOME/.gaiasky/scripts` so that Gaia Sky can run them.

An interface is provided in order to encapsulate some complex-behaviour functions and to make scripting easier. This scripting interface is described in the following section.
The scripting interface

The scripting interface is located in the package `gaia.cu9.ari.gaiaorbit.script` (see here).

Description of the functions

Before starting, have a look at the documentation of the scripting interface, which provides extensive descriptions of each function.

Using the scripting interface

In order to import the scripting interface package in your script, you just need to import the default implementation `EventScriptingInterface` from the package `gaia.cu9.ari.gaiaorbit.script`:

```python
# Import scripting interface
from gaia.cu9.ari.gaiaorbit.script import EventScriptingInterface

Then, we need to create the scripting object before start using it.

```python
gs = EventScriptingInterface()
```

Now, we can start executing functions.

```python
# Disable input
gs.disableInput()
gs.cameraStop()
gs.minimizeInterfaceWindow()

# Welcome
gs.setHeadlineMessage("Welcome to the Gaia Sky")
gs.setSubheadMessage("Explore Gaia, the Solar System and the whole Galaxy!")
[...]
```

More examples

You can find more examples by looking at the `scripts` folder in the Gaia Sky package.

How to run scripts

Each script is executed in its own thread in the virtual machine, and runs alongside Gaia Sky. In order to run a script, follow the procedure described in the Running scripts section.

2.4.16 Capturing videos

In order to capture videos there are at least two options which differ significantly.
Frame output system + ffmpeg

This method consists on outputting an image to a file every frame using the Frame output of Gaia Sky to later gather them to create a video using a video encoder software such as ffmpeg, which works on Windows, Linux and OS X.

Once you have recorded the images using the Frame output, you can convert them into a video using the following command:

```
$ ffmpeg -start_number [start_img_num] -i [prefix]%05d.png -vframes [num_images] -s 1280x720 -c:v libx264 -r 25 -preset [slower|veryslow|placebo] -pix_fmt + [out_video_filename].mp4
```

You need to obviously change the prefix and start number, choose the right resolution, frame rate and preset and modify the output format if you need to.

OpenGL context recorder

There are several available options to record OpenGL rendering to videos, in all systems. Below are some of these listed. These methods, however, will only record the OpenGL scene as it is displayed in the screen and are limited to its resolution.

**Linux**

- **glc/glcs** - Command-line interface applications. The documentation and user guides can be found in this wiki.
- **Gamecaster** - Front end to glc.
- **Soul Capture** - Front end to glc.

**Windows**

- **FRAPS** - 3rd party Direct3D and OpenGL recording software.
- **NVIDIA Shadowplay** - Only for Geforce cards.

2.4.17 Taking screenshots

Gaia Sky has an in-built screenshot capturing feature. To take a screenshot press F5 any time during the execution of the program. By default, screenshots are saved in the $HOME/.gaiasky/screenshots folder. The screenshots are in PNG format with high quality settings, so they can grow quite big if the resolution is large.

**Screenshot modes**

- **Simple mode** - This mode saves the current screen buffer to a file. It captures also the GUI and it does so at the current display resolution.
- **Advanced mode** - This mode renders the current scene to an off-screen buffer with an arbitrary resolution. The resolution can be configured in the config dialog, Screenshots tab. The advanced mode will NOT capture the GUI or any additional elements that are not part of the scene.
2.5 Javadoc

You can browse the Gaia Sky javadoc here:

- Gaia Sky javadoc.

2.6 Changelog

- Version history
- Detailed changelog
- Full commit history

2.7 Gaia Sky screenshots

Here are some screenshots.
2.8 About

2.8.1 Contact

If you have doubts or issues you can contact us using one of the following methods.

- Submit an issue to our bug tracking system.
- Drop us a line in tsagrista@ari.uni-heidelberg.de.

Do not forget to visit our Homepage@ARI and also the Gaia Sky product page.

2.8.2 Author

Toni Sagristà Sellés – tonisagrista.com

2.8.3 Acknowledgements

The author would like to acknowledge the following people, or the people behind the following technologies/resources:

- Prof. Stefan Jordan (stefan-jordan.de) for some suggestions and ideas.
- The DLR for financing this project.
- The BWT, Bundesministerium für Wirtschaft und Technologie, also supporting this project.
- My institution, ARI/ZAH.
- Dr. Martin Altmann for providing the Gaia orbit data.
- Libgdx.
- libgdx-contribs-postprocessing.
- HYG catalog.
- PgsLookAndFeel.
- Mark Taylor’s STIL library.
- The Jython Project.
- ernieyu for the Java Swing range slider.
- Nick Risinger for the artist’s conception of the Milky Way.
- Andreas Ressl and Georg Hammershimid for the star glow texture.
• Tom Patterson (shadedrelief) for some textures.
• Solar System Scope for some textures (Mercury, Venus, Earth, Moon).
• Phil Stooke and Grant Hutchison (site) for some of the textures.
• David Navas for the simple spaceship model. Cool stuff!
• Machuca+Arias+Caballero for the music track “Gaia-DR1”.
• Install4j (multi-platform installer builder), for providing a free open source license.
• Bitrock’s InstallBuilder for providing a free open source license.
• And several online resources without which this would have not been possible.

If you think I missed someone, please let me know.