



The AudeLA software

A. Klotz^{1,2*}, R. Delmas², D. Marchais², M. Pujol² and C. Jasinski²

¹*IRAP, University of Toulouse, 9 Avenue du colonel Roche, 31028 Toulouse Cedex 4, France*

²*Association des Utilisateurs de détecteur Electroniques, 45 Avenue Reille, 75014 Paris, France*

Abstract. Software is essential to the success of a robotic telescope. Very different tasks must be correctly performed each night: *e.g.* instrument driving, weather monitoring, automation, scheduling, image acquisition. The software of a robotic observatory is therefore complex and, if not designed properly, it can result in many lost nights. AudeLA¹ is an open source astronomical software package designed for image processing and for driving telescopes and cameras. AudeLA is also used successfully in robotic observatories. This paper explains in details the features of the AudeLA software.

Keywords : astronomical software – robotic observatory

1. History of the AudeLA software

AudeLA stands for A.U.D.E. Logiciel d’Astronomie (Astronomy software for A.U.D.E.). A.U.D.E. (Association des Utilisateurs de Détecteurs Electroniques) is a French astronomical association created in 1994 and devoted to the development of digital astronomy. In 1998, a few A.U.D.E. members designed the Audine CCD camera² (Fig. 1). This camera is based on a Kaf-400 Kodak CCD chip. It was sold as a kit with a detailed documentation of educational interest. Audine cameras are also used for teaching CCD technology in Master of Science in Astrophysics at the Toulouse University, and at ISAE (Institut Supérieur d’Aéronautique et de l’espace) as well. A.U.D.E. also created a dedicated software for the Audine camera based on the same open source concept as the camera. A.U.D.E. has remained active since 2000 partic-

*email: aklotz@irap.omp.eu

¹<http://www.audela.org/>

²<http://www.astrosurf.com/audine/>



Figure 1. Audine camera.

ularly in the field of spectroscopy³. The AudeLA software has been developed as a full open source software package for amateur astronomers and now includes many camera and telescope drivers as well as image processing routines. The project is managed by a core team made up by the authors of this paper while many other contributors participate in the development of AudeLA. One characteristics of AudeLA is that any astronomer can easily add its own commands into the software using a plug-in system written either in script language (*e.g.* to design a graphical tool) or in C/C++ code (*e.g.* to write an instrument driver).

Since 2003 AudeLA has been used to manage the TAROT robotic observatories (Klotz et al. 2008). Since 2010 AudeLA controls the Zadko robotic telescope in Australia (Coward et al. 2011). These adaptations consist in a set of additional scripts based on AudeLA functions. The TAROT and Zadko scripts are too specific to be publicly distributed. However, a dedicated plug-in has been included in the last AudeLA version to drive a simple robotic observatory. Finally, AudeLA has also been used since 2010 to drive the T193 telescope at the Observatoire de Haute-Provence in South-East France, providing graphical tools to help support night operators.

AudeLA now has at least two aspects: (i) An intuitive graphical user interface for observers needing a software to drive their observatory devices, (ii) A development platform for astronomers who need to add their own functions to a software.

2. Graphical interface, terminal or daemon ?

Before defining the specifications of AudeLA we need to explain the different running modes of an astronomical program.

Usually, astronomers like programs with a graphical user interface (GUI). For example, the well known SAOImage DS9⁴ is based on such a GUI. In that program functions can be used intuitively thanks to the menu bar and window gadgets (hereafter called widgets). Window forms can be used to display images, get the intensity

³<http://www.astrosurf.com/buil/>

⁴<http://hea-www.harvard.edu/RD/ds9/>

of pixels and to configure image processing parameters. However GUIs have their drawbacks too and the operator must click on buttons all the night to perform the acquisitions. It's a monotonous repeating work, with high probability of human errors. As a consequence, programs based on a GUI are not suited to autonomous robotic observatories. Moreover, software maintenance based on copying the desktop display is not efficient if the distant observatory has a very low bandwidth.

Contrarily to GUIs, a text terminal can be used to write command lines (*e.g* IRAF⁵). Commands are sentences in which words are actions to perform. There are low level commands for basic actions (*e.g* load an image in memory, display an image, start a camera acquisition). If the command language provides programming functions such as conditions and loops it becomes possible to write scripts that describe automatic actions to be carried out during the night. The drawback is that the command syntax must be learned. Moreover, a terminal alone cannot display images. However, the terminal running mode remains a very powerful way to manage a complete night in an automatic observatory. SSH text connections allow to easily launch the terminal running mode from a distant computer even with a low bandwidth.

The third running mode is daemon, also called service in the Windows OS. A daemon is a program that runs with no GUI and no terminal. The commands are usually received using TCP/IP channels, therefore any program can connect to the daemon as a client and send commands. A lot of professional software written for autonomous robotic observatories are based on daemons, *e.g* RTS2 (Kubánek et al. 2008; Kubánek 2010). In such a software, each daemon performs a specific task (*e.g* one for camera acquisition). In the virtual observatory context (Klotz 2010), daemons are often designed as agents. Daemons are simply started by cron or by a dynamic web interface. Their drawbacks are that there are only text logs to debug problems and their configurations are given from a file or by database access.

Each running mode described above has advantages and drawbacks. To sum up, GUIs are well suited for limited and punctual tasks (manual control, maintenance, tests, etc.). Command line terminal mode can bring the power of a script language that can describe how to run an observing night automatically. Daemons run forever and are waiting for basic tasks sent by a central software that organizes the observatory life.

3. Specifications of the AudeLA software and choices

3.1 Specifications

Some specifications were decided when the AudeLA project started in 1999. A few others were added later. Here is the list of them:

⁵<http://iraf.noao.edu/>

- The program must be run under Linux or Windows operating systems (OS). As a result, no OS specific libraries will be used, and wherever possible the source code shall be identical for both platforms. It restricts the compiler to use a standard and cross platform programming language.
- Some very useful astronomical components are OS specific. For example, the Astronomy Common Object Model (ASCOM) and Instrument Neutral Distributed Interface (INDI) protocols are often used to drive telescope mounts, domes and cameras under Windows and Linux respectively. These components must be driven using an OS automatic recognition.
- The script language should be simple, intuitive and easy to use, so astronomers will be able to use it without significant knowledge of computer science. Consequently the syntax should be kept simple and not make use of the abstractions enabled by “object oriented programming”.
- Due to the importance of the script language, a script terminal must be available in addition to the graphical user interface.
- The script language can be extended by astronomers who want to create new functions. Moreover, the script language must include additional libraries that can extend the basic grammar.
- The script language must contain communication functions such as TCP sockets or the use of serial ports. Moreover, high level protocols must be supported by the script language: HTTP, CGI or SOAP. These protocols give an access to astronomical databases and to Virtual Observatory tools at scripting level.
- The graphical user interface must be flexible and should be easily modified. Any astronomer can include new personal features (windows, buttons, forms, skins) in order to adapt AudeLA to its own observatory needs.
- There is a default graphical user interface but it can be replaced by another one written by any user.
- AudeLA can be launched with no graphical user interface allowing to use it in Common Gateway Interface scripts (CGI) or as daemon or service.

3.2 The choice of the C/C++ and Tcl/Tk languages

The OS compatibility restricts programming languages to those able to be compiled with both OSs. We initially had to choice between C/C++ and Java. The problem with Java in 1999 was that it was not associated with a powerful script language. Free script languages linked to C/C++ were mainly Tcl/Tk, Perl, PHP and Python. The choice of the script language was based on the simplicity of the syntax rules and on the tight integration with the C language.

Tcl/Tk appeared to be the best scripting language in 1999 because it can be used inside a C program or it can be extended itself by C language. The syntax rules are basic, and it is often used to quickly develop small applications, drive instruments,

or tie together heterogeneous tools (glue language). Note that the Tcl/Tk language was created by John Ousterhout for his students that want to drive electronic devices⁶. That is exactly what we want to do in a robotic observatory. The installation of Tcl/Tk is also very simple and a program can be distributed as a single executable file. For example, DAOImage DS9 is also based on Tcl/Tk and C/C++ and is distributed as a single binary file. We did not want to do that with AudeLA because we chose to let astronomers the possibility to add personal libraries and modify Tcl scripts.

The Tcl syntax is based on a list of words separated by a space character, like command lines. The first word of the list is actually a command and is followed, if needed, by parameters. For example, to assign 4 to the variable `a`, the corresponding Tcl code list is `set a 4`. The command `set` is followed by two parameters. The first parameter is the symbol of the variable to be assigned and the second one is the value itself. Tcl has only one type of variable: the string. Everything is string in Tcl. To make a calculation, use the command `expr`. For example `expr 1+2!` returns 3 as a result.

A great asset of Tcl is that we can add a new command either using the Tcl command `proc` or using the extension system provided by the Tcl library. The Tcl library provides a comprehensive Application Programming Interface (API) that allows to use Tcl functions in the C/C++ code of a shared library. In particular, the function `Tcl_CreateCommand` registers a word as the name of a new Tcl command and associates a pointer to a C function which will be executed whenever that command is called in a script. This mechanism is useful when time performance is required or to access to device drivers.

The AudeLA platform is a collection of Tcl extensions, written in pure Tcl language or written in C/C++ code in shared binary libraries, providing astronomical functions to the Tcl language.

3.3 Installation of AudeLA

Ready to run packages can be downloaded from the source-forge server⁷ as `.exe`, `.rpm` and `.deb`. For C/C++ developers, the source codes are available using a SVN client. The AudeLA structure folders is: `bin`, `lib`, `gui` and `src`. The `bin` folder contains all executable and shared library files. The `lib` folder contains a lot of additional Tcl public packages providing tools as HTTP, FTP, SOAP, XML, SSL, mime, etc.

The `gui` folder contains two sub-folders: `Audace` and `tutorial`. `Audace` is the name of the standard GUI of AudeLA. It contains all Tcl/Tk files that define the graphical

⁶<http://www.tcl.tk/about/history.html>

⁷<http://sourceforge.net/projects/audela/files/audela/>

appearance but also additional useful Tcl functions for astronomy. The sub-folder tutorial contains another GUI that is a tutorial to test the Audine kit. This tutorial demonstrates that the AudeLA GUI can be rewritten to be entirely different from the default Audace GUI.

The src folder contains C/C++ files spread in a few sub-folders: audela (sources of AudeLA core libraries), contrib (sources of AudeLA personal libraries written by contributors), external (sources of instrument drivers, Sextorator, Fitsio, etc.), libcam and libtel (sources of camera and telescope mount drivers linking Tcl and manufacturer drivers).

3.4 Instrument plug-ins of AudeLA

The libaudela library (C++ code in the audela subfolder) provides a plug-in management system to connect cameras and telescope mounts. Each sub-folder of libcam and libtel contain C/C++ codes to generate shared libraries which will be used by libaudela to create new Tcl commands that give a direct access to the related hardware item. For instance, to drive a SBIG USB camera, simply type :

```
::cam::create sbig USB
```

The second parameter `sbig` loads the libsbig AudeLA library (C code in the libcam subfolder) that creates an instance of a Tcl function related to the SBIG manufacturer driver. The new instantiated command is `cam1` if this is the first camera connected (many cameras can be connected). The source code of libsbig is linked with the vendor specific library and when the entry point is called, (C function `cam_init`) then the camera model is recognized and initialized. The two following Tcl commands set the exposure time to 10 seconds and start an acquisition:

```
cam1 exptime 10
cam1 acq
```

3.5 Running modes of AudeLA

As explained in Section 2, AudeLA can run differently if it is launched as a daemon or as a GUI. The GUI is launched when AudeLA is call from the executable `audela` file, located in the bin folder:

```
./audela
```

To launch AudeLA without graphical interface, use following command:

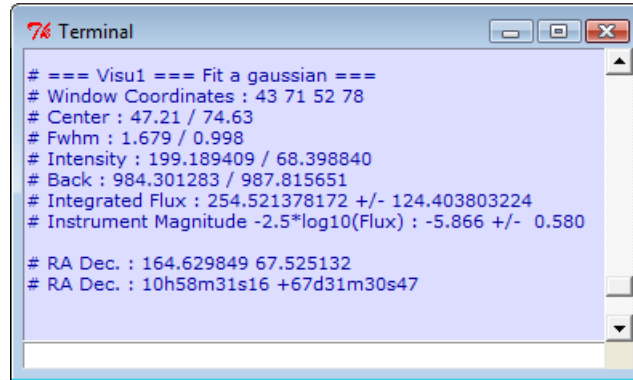


Figure 2. The terminal GUI of AudeLA. The bottom white line is used to write Tcl command lines.

```
./audela --console
```

If used as a daemon the file name of a start-up script to be run can be defined in the command line. This script can be a socket server that waits for an order to execute:

```
./audela --console --file daemon_script.tcl
```

4. AudeLA GUI

The standard AudeLA GUI is called Audace and it is written in pure Tk. Tk is a graphical toolkit that copes with the platform window manager, either the Windows core, or the various window managers on Unix/Linux. The syntax of Tk is Tcl compatible and can be embedded in Tcl scripts. To comply with the requirements defined in Section 3.1, two graphical windows are created. The first window is a terminal where the user can write a Tcl command and read text results (see Fig. 2). The second window (see Fig. 3) contains a menu bar, a 2D space to visualize images, and a bar to set limits of visualizations. An additional tool bar can appear in this window according to the item clicked in the menus. Any user can add a new tool bar using the AudeLA plug-in GUI system defined in the folder `gui/audace/plugin/tool`. Up to now, fifty plug-in GUI tools are available in the standard distribution of AudeLA. Some are used to easily drive cameras and telescope mounts, others are devoted to image processing. In any case, these plug-ins enable the user to customize AudeLA to a specific instrument.

The first time AudeLA is launched, the user can select the language, and customize the setup menu (image folder, the camera connection, observatory location). Fig. 3 shows the items of the setup menu.

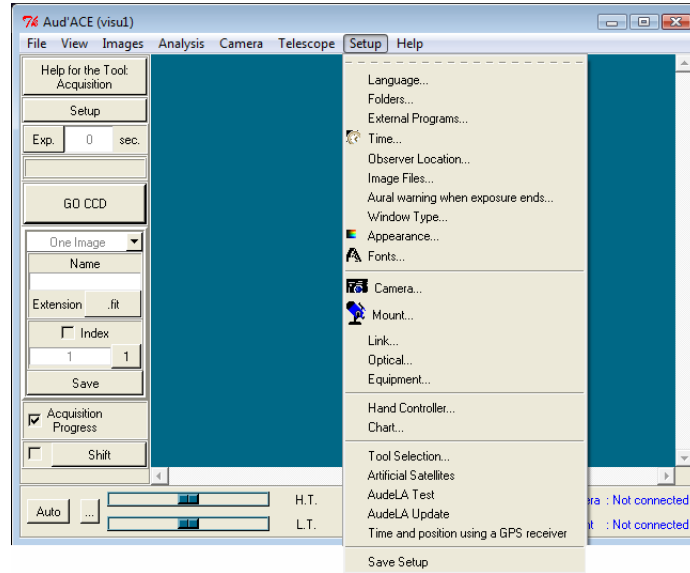


Figure 3. The main GUI of AudeLA. The figure shows the items of the setup menu as well as the left vertical tool bar containing the acquisition plug-in GUI tool.

5. The Robotic observation plug-in GUI tool

Only few commercial software packages provide a GUI to drive robotic observatories (*e.g.* ACP, Prism). Based on the ten years experience with TAROT telescopes, a plug-in GUI tool has been written for AudeLA. Even if the tool is not as complete as the one in RTS2, the goal is to help astronomers during the first stages in robotic mode.

The plug-in GUI tool for robotic observations is available in the file menu. A tool bar contains three buttons: Setup, Planning, Acquisition. They should be used in this sequential order.

The Setup button provides a form window that allows to specify pointing limits, conditions to start observations, folders for images and astrometric catalogs, etc.

The Planning button allows the user to choose an observing strategy. For example “Searching for supernovae” forces the acquisition to observe targets amongst a list of galaxies. A personal scheduler algorithm can be written if the user wants to define a specific observing strategy.

The Acquisition button starts an infinite loop that execute the following sequence: load setup, check if it’s night time, check camera and telescope connections, check safety (rain for example if there is a rain sensor), compute the next stage according to the strategy, slew the telescope (and the dome if any), check setup of instruments

(filter wheel), check if the pointing is OK, perform image acquisition, dark frame acquisition, flat frame corrections, WCS calibration, archiving.

The goal of the acquisition loop is to perform the observations but also the image processing in order to obtain an on-line results (e.g. light curves).

6. Conclusions

AudeLA is an open source software well suited to robotic observatories. A plug-in GUI tool enables beginners to drive a simple robotic observatory. For computer science experts, AudeLA provides enough basic astronomical functions to create complex scripts and drive each specific instrument in a robotic observatory.

Acknowledgment

The authors wish to thank all the AudeLA contributors for the time they spend developing new functions and John Ousterhout who created of the Tcl language in 1988.

References

- Coward D. M., Heary A., Venville G., 2011, *Advances in Space Research* 47, 1922
- Klotz A., Boër M., Eysseric J., et al., 2008, *PASP* 120, 1298
- Klotz A., 2010, *Advances in Astronomy* 2010, 285
- Kubánek P., Jelínek M., French J., 2008, *SPIE* 7019, 92
- Kubánek P., 2010, *Advances in Astronomy*, 2010, 243