Proceedings of Student/Faculty Research Day, CSIS, Pace University, May 5th, 2006

Astronomy Imaging System

Michael R. Pasacrita
Computer Science Department, Pace University
861 Bedford Road, Pleasantville, New York, 10570 USA
Michael.Pasacrita@fairfin.com

Abstract

Astronomy Image Systems (AIS) are used for display and analysis of astronomy images. Commercial astronomy image systems such as AIPS are available in the market and provide advanced functionality to help astronomy professional's view, analyze and process astronomy images. But they are generally not within the reach of an individual astronomy professional and may not meet the special needs of each individual user. It is our attempt to develop a simple Astronomy Imaging System with the following qualities:
Standalone application (i.e. Runs on a single PC)
Platform Independent (i.e. Windows, Linux, etc...)
Requires limited resources (i.e. A Pentium CPU, with low RAM and small HD will support the software.)
Within reach of an individual user (i.e. Freeware)

Keywords

Astronomy, Image, System, FITS, Viewer, Metadata, Classification

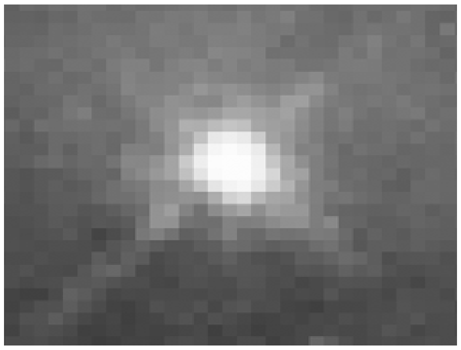
1. Introduction

1.1 What is an image?

An image [5] is an array, or a matrix, of square pixels (picture elements) arranged in columns and rows.

Below is an example:

Figure 1: An image — an array or a matrix of pixels arranged in columns and rows.



In a (8-bit) greyscale image each picture element has an assigned intensity that ranges from 0 to 255. A grey scale image is what people normally call a black and white image, but the name emphasizes that such an image will also include many shades of grey**.** A normal greyscale image has 8 bit color depth = 256 greyscales. A "true color" image has 24 bit color depth = 8 x 8 x 8 bits = 256 x 256 x 256 colors = ~16 million colors.

1.2 Astronomical images

Images of astronomical objects [5] are usually taken with electronic detectors such as a CCD (Charge Coupled Device). Similar detectors are found in normal digital cameras. Telescope images are nearly always greyscale, but nevertheless contain some color information. An astronomical image may be taken through a color filter. Different detectors and telescopes also usually have different sensitivities to different colors (wavelengths).

Below is an example of an image composed from narrow-band exposures. This results in very sharply defined wisps of nebulosity since each exposure separates light from only some very specific physical processes and locations in the nebula.

Figure 2: Example of an Astronomy image constructed from narrow-band exposures

1.2.1 FITS Images

The file format, know as Flexible Image Transfer System (FITS), is the most widely-used file format in Astronomy [1, 2, 3, 4]. The Flexible Image Transport System (FITS) evolved out of the recognition that a standard format was needed for transferring astronomical data from one installation to another. The original form, or Basic FITS, was designed for the transfer of images and consisted of a binary array, usually multidimensional, preceded by an ASCII text header with information describing the organization and contents of the array. The FITS concept was later expanded to accommodate more complex data formats. A new format for image transfer, random groups, was defined in which the data would consist of a series of arrays, with each array accompanied by a set of associated parameters.

FITS is used to store non-image data in conjunction with images of various formats. A major advantage of FITS for scientific data is that the header information is human readable ASCII, this allows an interested user to examine the headers to investigate a file of unknown provenance. Each FITS file consists of one or more headers containing ASCII card images (80 character fixed-length strings) that carry keyword/value pairs, interleaved between data blocks. The keyword/value pairs provide metadata such as size, origin, binary data format, free-form comments, history of the data, and anything else the creator desires: while many keywords are reserved for FITS use, the standard allows arbitrary use of the rest of the name-space (wikipedia).

1.3 The AIS Astronomy Image System (AIS)

*The specific client for this Astronomy Imaging System (AIS) is a Computer Science professional at Pace and a DPS student as well, Matt Ganis. He hopes to pursue a doctoral dissertation in image recognition of solar images. The system will be used as a principal tool to assist him in Astronomy research. Also, he will be provided with the source code to this software, so that he can modify it's capabilities to include his algorithms of image recognition. AIS is designed to provide the user with the ability to search, categorized and manage files of various formats GIF, JPEG and FITS (Flexible Image Transfer System) the most widely-used file format in the Astronomy universe (no pun intended)**.** The system will also help the user to maintain "metadata" and catalog information of astronomical images that the user will be collecting for his astronomy research.*

AIS provides users the ability to store, process and manage astronomy images [1]. Professor Ganis needed a stand-alone, PC-based AIS to maintain a database of images and their associated attributes. The system must also allow for appropriate query functions to be developed.

The system needs the capability to:

- Add images (together with possible "hand" classifications) to the database
- Display an image from the database
- Delete images from the database
- Query the system to retrieve all images of a certain "hand" classification type

The customer will gather images from various sources. Some of the images will be "hand" classified as having particular attributes (i.e., these are tracking problems - causing streaks, these are out of focus, these are over exposed, these are under exposed, etc). The recognition portion of the system (e.g., neural network, once it is developed) will have the capability to automatically classify images.

Images will likely be in FITS format so a fits-viewer will need to be incorporated. The system should, however, support multiple image types (.jpg, .gif, .bmp). A conversion function from *.fits to *.jpg would be an added option. FITS files also contain several pieces of metadata that should be included in the metadata for fits images to allow for additional query functions. The attribute list for the images will be developed over time, so the list must remain flexible.

In this paper, I describe the technical aspects of our Astronomy Image System. In Section 2, I describe the technologies used to implement the AIS. In Section 3 I explain the methodology used and in Section 4 I project future work on this system. Finally Results and Conclusions are provided in Section 5 followed by References at the end.

2 Technology

2.1 The requirements

The customer requirement is a standalone application for a personal computer. It is also requested to store data pertaining to photo images, some of which can not store this meta-data themselves. Lastly, the program must be user friendly enough that information can be tracked on a large scale for it. For this reason I have chosen the following implementation technologies. The main program itself will be created in Sun's Java J2SE platform. The user-interface will be created using an implementation of Java's "Swing" and "AWT" libraries. Lastly, the data storage will be using a very simple set of tables read in through Java, created as comma delimited, "CSV" files. The reasons for these choices are many.

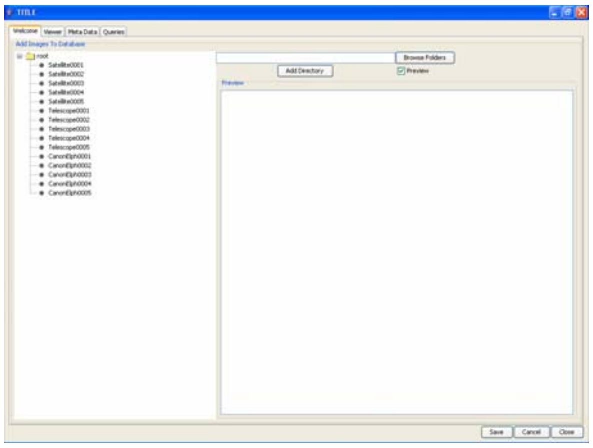
2.2 Sun Java J2SE Platform [10]

There are a couple different versions of Sun's Java platform including J2EE for enterprising solutions, and J2ME for projects of a mobile scope. The J2SE (standard edition) is however has the recommended libraries for the creation of a Standalone desktop application. While other options for this are limitless, ones most accessible to our team would be C++, Visual Basic, or a Scripting language for a graphical environment, but I feel I am the most comfortable and familiar with the Java language. The customer requires that this program eventually be platform independent, so C++ or VB would not be possible, because they are Microsoft developed environments, not applicable to a Linux OS. Also, Java's availability of libraries shared over the World Wide Web will no undoubtedly save time in development. This is due to the fact that the more components that are built "in-house", the longer the time required building these modules, and the greater possibility that there will be a need for code maintenance. This approach of reusable components will be very important in the saving of precious hours of programming, traded in for less the hopefully lower time it takes to find components that I will only need to learn how to implement. Also, Java allows us to use its interfacing solutions to reduce the complication of linking the front-end to the fine print of our coding. For more on this topic, please read the following section on User Interface development in Java's Swing and AWT interfaces.

2.3 Java User Interface – Swing and AWT [9, 10]

Java's interfacing solutions (see previous) are their Swing and AWT packages. With a couple simple import statements, I can begin to start defining elements the user will see. This will also allow me to create the UI first, and then add functionality to it as I develop it. While other UI tools such as ActionScript's "Flash" interface are easier to create nice looking graphics, and allow for animation and color depth that Java does not offer, ActionScript just doesn't give us the same power code-wise and would mean I would have to write hack Scripts to perform many tasks, such as File IO and Database reading.

Fig 3. AIS User Interface – Main Menu



2.4 Images Data store – CSV Files [7,8]

For storing potentially large amounts of data about the locations and attributes of many images, some form of data store is required [7,8]. A relational database engine would be a great solution as it provides several benefits – Security, Data Integrity, Recoverability, Scalability etc.. But considering the special and simple requirements of the customer and user's preference to avoid the task of maintaining a full fledged RDBMS it is felt that the large amount of system resources demanded by an RDBMS is not worth for just a potentially very thin standalone application.

They are also pretty readable, and 100 percent parse-able text viewer as Comma separated values. This means they can be read in at system load-up of our application, and saved at the press of a button. Also, there are libraries available for free on the internet for parsing and querying data from this format. This will make the development time relatively minimal.

CSV parsing methods will check to see if a file called AIS.csv exists. If not, generate that CSV file, if so parse through it and separate which Strings are headings, and which are values. Also, it differentiates between the metadata and the file info by parsing the

2D array twice. Once it just parses the first two columns of each row, bringing in the file name (first column) and file location (second column). Also, it has a method that will parse the heading of the file (metadata tag names) and the second line (default values) which will be useful when adding an image to the data. Also, it has 2 methods to add a line of data to the bottom, and to the far right of the table for adding files and new metadata tags.

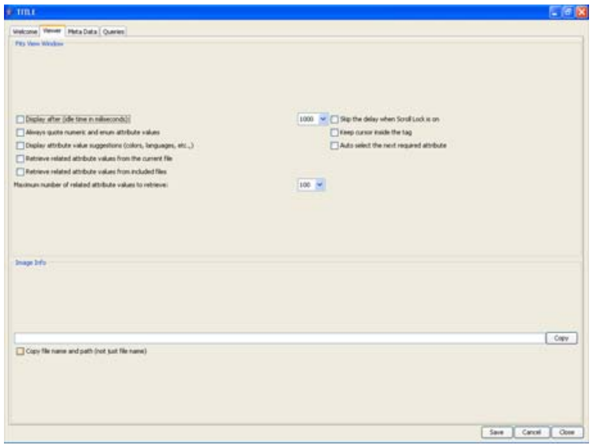
2.5 View Images – FITS Viewer

Java supports, GIF, JPEG, BMP, TIFF and not FITS. So I use Jfits FITS viewer to view FITS images as well. Also the metadata will be separated from the image data.

An application intended to render a FITS [5,6] image for viewing by a user has significantly more responsibility than an application intended to handle other standard image formats (e.g., "jpg" or "gif" images). FITS data arrays contain elements, which typically represent the values of a physical quantity at some coordinate location. Consequently they need not contain any pixel rendering information in the form of transfer functions, and there is no mechanism for color look-up tables. An application should provide this functionality, either statically using a more or less sophisticated algorithm, or interactively allowing a user various degrees of choice. Furthermore, the elements in a FITS data array may be integers or floating point numbers. The dynamic range of the data array values may exceed that of the display medium and the eye, and their distribution may be highly non-uniform. Logarithmic, square root, and quadratic transfer functions along with histogram equalization techniques have proved helpful for rendering FITS data arrays. Some elements of the array may have values which indicate that their data are undefined or invalid; these should be rendered distinctly.

The data array in a FITS image must have a dimensionality between 1 and 999, the boundaries inclusive, indicated by the NAXIS keyword. The extent of any coordinate axis in a FITS data array may, however, consist of only a single element. Hence an algorithm designed to render two-dimensional images will be capable of displaying a three- or four-dimensional FITS array when one or two of the axes consist of a single pixel. Three-dimensional data arrays (NAXIS=3 with multiple elements along each) are of special interest. Inspection of the World Coordinate System (WCS) keywords in an image with NAXIS = 3 or more may indicate that one of the axes is temporal. Writers of viewer applications should consider presenting such an image in a fashion akin to that used for an animated GIF. Even in the absence of WCS indication of a temporal axis this time-lapse display technique can be effective, and application writers should consider offering it for all three-dimensional arrays.A FITS image with NAXIS=1 is a one-dimensional entity such as a spectrum or a time series. Writers of applications intended to handle these one-dimensional FITS images should consider presenting such an image as a graphical plot rather than as a two-dimensional picture with a single row.

Fig 3. AIS User Interface – Main Menu



☐ Display after (idle time in milliseconds): 1000
☐ Skip the delay when Scroll Lock is on
☐ Always quote numeric and enum attribute values
☐ Keep cursor inside the tag
☐ Display attribute value suggestions (colors, languages, etc.,)
☐ Auto select the next required attribute
☐ Retrieve related attribute values from the current file
☐ Retrieve related attribute values from included files
Maximum number of related attribute values to retrieve: 100
☐ Copy file name and path (not just file name)

I integrate and exploit all these simple technologies in our AIS implementation to perform all the customer's requests. Thanks to the portability of Java and Swing, this application will work exactly the same in an NT or Linux environment, and contain all the graphical elements necessary to perform user input and display the required information to the user.

3 Methodology

3.1 AIS objectives

The basic objective of the methodology used is to maintain "metadata" and catalog information of astronomical images that user will be collecting for his research. Basically the user needs a system to track all of the images I receive along with some attributes of the image (all of which I will enter).

I present here methodology adopted to view and manage the astronomy images by the user.

3.2 Agile Software and KIS (Keep it Simple) methodolgies

I tried to keep the AIS implementation simple and followed Agile software Desing and Development principles through out. The necessary design and code changes were made as needed by regular interaction among the team members and feedback from the customer. This helped meet the changing requirements efficiently. Simpler options were chosen to the extent possible to meet the requirements of the customer. For example CSV files were chosen in place of a full RDBMS engine to keep it simple for the user to use the system easily and efficiently.

3.3 AIS Methodology – How all components fit together?

The user periodically collects images from various sources for his research work. When he gets images he'll put them in a specific directory on his personal computer (laptop/desktop). The AIS package that includes all the software is pre-loaded into the user's personal computer. The user starts the AIS system whenever he needs it. Upon startup the AIS finds automatically where the images are stored by reading the AIS data store which is a CSV file on the disk.

The AIS system first reads the images data store (CSV file) and then displays a list of images that have not been cataloged on the screen on user's PC. The user can select any of these images for viewing and processing. When the user selects an image from the list, the image is retrieved from the disk and displayed on the screen. A separate window opens and allows the user to put in attributes of the image such as "out of focus", "nebula", "stars", "bloomed", etc. Once the user saves the attributes so entered, this metadata is saved and associated it with the given image in the CSV file Image data store.

If the image is FITS type the system will automatically invoke the FITS viewer and display the image appropriately. In future the AIS will be added the capability to automatically pick up the attributes from the FITS image itself and display to the user for viewing and processing.

4. Results and Conclusions

I provide here a simple standalone Astronomy Image System (AIS) to help Matt Ganis, an individual Astronomy professional Store, Classify and Manage astronomy images that he will collect for his research. KIS (Keep It Simple) principle has been adopted to meet the special needs of an individual user within the time frame. I tried to use Agile Software engineering processes to the extent possible to develop this system and I am glad I did so. I am satisfied with the results. There is lot of scope to make this system even more sophisticated and robust and add new functionality as described above under the projected future work heading. We intend to add this additional functionality in future.

5. Projected Future work

5.1 Relational Database

For now CSV files will be used to store the data. Depending on the volume and complexity of the database requirements a full fledged relational database such as Oracle that runs on all platforms including Linux may be considered in future to store the data more efficiently and provide richer features for data integrity, queries/reports and better performance.

5.2 Dropdown menus

Drop down menus will be provided to help enter or modify the attributes of the images. The menus will be pre-loaded with valid choices to make it more convenient for the user and also ensure data integrity by allowing the user to enter only valid choices from the drop down menu.

5.3 Automatic Classification

Classification of the images will be automated to the extent possible by reading the image attributes automatically from the image. If possible also provide features to automatically measure the quality of the image and classify it appropriately.

At a later point (perhaps spring semester), allow selected images to be fed to a neural network analyzer for automatic classification.

References

1. Astro Images Web site
   http://www.astroimages.org/

2. Greisen, E. W. *FITS: A Remarkable Achievement in Information Exchange* Information Handling in Astronomy - Historical Vistas. Edited by André Heck, Strasbourg Astronomical Observatory, France. *Astrophysics and Space Science Library*, Vol. 285. Dordrecht: Kluwer Academic Publishers, 2003., p.71
http://www.fukuoka-edu.ac.jp/~kanamitu/fits/doc/Greisen.fh.pdf

3. Greisen, E. W., and Calabretta, M. R., *Representations of World Coordinates in FITS*, *Astronomy & Astrophysics*, 395, 1061-1075, 2002

4. Hanisch, R. J., Farris, A., Greisen, E. W., Pence, W. D., Schlesinger, B. M., Teuben, P. J., Thompson, R. W., and Warnock, A., *Definition of the Flexible Image Transport System (FITS), Astronomy & Astrophysics*, 376, 359-380, 2001

5. A short introduction to astronomical image processing – A web article from Hubble European Space Agency Information Centre (HEIC).

http://www.spacetelescope.org/projects/fits_liberator/image_processing.pdf

6. Definition of the Flexible Image Transport System (*FITS)* March 29, 1999 - NASA/Science Office of-Standards and Technology

http://archive.stsci.edu/fits/fits_standard/

7. E.F.Codd: The Relational model for Database management Version 2, Addison-Wesley (1990).

8. C.J. Date: An Introduction to Database Systems, Eighth edition, Addison-Wesley (2004).

9. Kathy Sierra and Bert Bates**:** *Head First Java*, 2nd Edition, O'Reilly Media [2005].

10. Sun Java Developer resource web site

http://java.sun.com/j2se/reference/techart/index.html