FUTURE DIRECTIONS FOR ASTRONOMICAL IMAGE DISPLAY

NASA Grant NAG5-3996

Final Report

For the Period 1 February 1997 through 31 January 2000

Principal Investigator
Eric Mandel

March 2000

Prepared for:

National Aeronautics and Space Administration
Washington, D.C. 20546

Smithsonian Institution
Astrophysical Observatory
Cambridge, Massachusetts 02138

The Smithsonian Astrophysical Observatory
is a member of the
Harvard-Smithsonian Center for Astrophysics

The NASA Technical Officer for this grant is Joe Brederkamp, Code 077.0, NASA Headquarters, Washington, DC 20546-0001.

1 Project Objectives

In the "Future Directions for Astronomical Image Display" project, the Smithsonian Astrophysical Observatory (SAO) and the National Optical Astronomy Observatories (NOAO) evolved our existing image display program into fully extensible, cross-platform image display software. We also devised messaging software to support integration of image display into astronomical analysis systems. Finally, we migrated our software from reliance on Unix and the X Window System to a platform-independent architecture that utilizes the cross-platform Tcl/Tk technology.

2 Accomplishments

Our efforts during the course of this project were concentrated in three areas:

- support for existing image display and messaging software
- development of new messaging software
- development of new image display software

We describe each of these efforts below.

2.1 Support for Existing Software

During the course of this project, we released three versions of our existing SAO R&D software suite (v1.8 in July 1998, v1.9 in May 1999, and v1.9.1 in January 2000), containing the *SAOtng* image display program and the *XPA* point-to-point messaging system. These releases of the SAO R&D package provided much improved support for astronomical image analysis and display. For example, we added support for several new types of *FITS* data, including image extensions, binary tables, n-D images, compressed *FITS* data, and *FITS* files that are on the Web. In addition, *SAOtng* now can display raw event lists (*i.e.*, binary files containing events with a known record structure).

Perhaps the most interesting and significant technical feature of our work on *SAOtng* was the development of a new technique for filtering photon events in *FITS* binary tables: the filter specification is converted into a tiny C program, which then is compiled, linked, and run automatically so that events can be fed to this program and filter results returned to the calling program. The power of the technique lies in these considerations:

- The generated filter program is very small, containing less than 200 lines of code, so that it compiles and links in a second or less on most modern machines.
- Filter checking is performed as part of compiled code, not in the usual interpreted mode. This use of a compiled filter results in an order of magnitude speed improvement over previous techniques, even after the program compilation overhead is added.

- All C syntax and C operators become valid parts of the filter syntax, making available a much wider range of filter possibilities than previously.
- It is easy to replace the standard filter program with a user-specified filter program for more sophisticated applications.

Our "compile on the fly" technique has not only been used successfully in *SAOtng*, but also has given rise to a new research direction (not funded under this grant): development of a "smart" *FITS* library that will support all essential *FITS* access functions (including column and spatial image filtering) with a minimum number of easy-to-use routines. This library is an attempt to build software that makes the best possible assumptions about what a user intends (so that the default behavior is correct for most applications), while also leaving room for less common behavior. Tailoring routines in this way opens the possibility of greatly simplifying many astronomical data analysis programming tasks.

Our SAO R&D software suite is in wide-spread use by astronomers all over the world. Each of our releases was accessed by hundreds of astronomical sites, from Russia to the Navy to the Vatican. We have had the pleasure of working via e-mail with hundreds of users. Major projects such as Chandra and XMM have made heavy use of this software – and of our successor image and messaging software, which also were developed under this grant, and which are described below.

2.2 Development of New Messaging Software

During the course of this project, we re-designed and re-implemented the *XPA* messaging system to move beyond its original conception as an inter-process communication system for X Window System programs. In implementing *XPA* 2.0, we paid close attention to the concept of "minimal software buy-in" that was described at our first PI workshop.

One of the most important changes we made to *XPA* 2.0 was to utilize standard sockets rather than the X Selection mechanism. The new *XPA* supports both INET and UNIX sockets, with the choice of method based simply on a single global environment variable. This change not only makes *XPA* 2.0 much faster than the previous implementation, but it also means that *XPA* no longer is dependent on the X libraries and can be added to non-X programs or ported to platforms (such as Windows) where X might not be available. Thus, we now support *XPA* within the Tcl language and in non-graphical Unix programs that process events using the standard select() system call. Other environments can be added in a straight-forward manner.

The new *XPA* 2.0 extends the single point-to-point communication of the original *XPA* by supporting broadcasting. An *XPA* service is registered using both a "class" and a "name" specification. For example, a number of image display programs might register themselves using the class designation "IMDISP" and names such as "saotng-1", "saotng-2", etc. A client can then send or receive messages to/from multiple *XPA* servers by using a class:name template to specify the desired servers. For example, a client can send messages to a specific

SAOtng program by specifying a name only (*e.g.*, "saotng-3") or send to all *SAOtng* programs by specifying a template such as "saotng*" or "IMDISP:*".

Communication between client and server is still direct, *i.e.*, there is no forced need to send a message to an intervening message router that in turn broadcasts the message to individual servers. However, we implemented an optional message routing program that acts as a classical message bus in situations where this is warranted. (For example, it might be desirable to send the currently active *FITS* file to such a message server, where it can be stored for later retrieval by other processes.)

Access to *XPA* services is provided on a host by host basis for each service. By default, users can access an *XPA* service if they are logged onto the same machine. Other default access permissions can be maintained in a simple ASCII file, so that access can be granted to users on other machines for any individual access point. Access permission to *XPA* services also can be changed at run-time (by the access point owner) using simple *XPA* commands. We can extend this security model to use private keys if we ascertain that the astronomical community requires the heightened security.

The user and programming interfaces for *XPA* 2.0 are easy to use but provide a great deal of flexibility and power. An *XPA* service is defined by means of the basic *XPA* server routine XPANew(). Application-specific send and receive callback routines are passed to this routine to implement the specific services. Once a number of *XPA* services are defined using XPANew(), they can be added to an X or Tcl event look with the calls XPAXtAddInput() and XPATclAddInput(), respectively (no such initialization is required for Unix select() loops). The server program then enters its event loop and handles *XPA* requests along with other requests.

On the client side, the *XPA* programming interface parallels the *XPA* programs available to the user at the shell level. For example, to send/receive data to/from an *XPA* service, one can execute xpaset/xpaget at the command line, or call the XPASet()/XPAGet() subroutines inside a program. No other program setup is required at the command line and no other subroutine calls are required within a program (although more advanced support is, of course, available for client applications.) This parallelization of the command-line and programming interfaces allows developers to test their application-specific services easily at the keyboard. (Indeed, we even have taken care to ensure that one connect to *XPA* services via telnet, so that these services can be exercised directly without using client *XPA* programs at all!) We believe *XPA* 2.0 is a major step toward our goal of developing "minimal buy-in" messaging.

XPA 2.0 was ported to Sun/Solaris, Linux, SGI, and Dec/OSF1 (meaning that it works on big-endian and little-endian machines, and on 32-bit and 64-bit architectures). We made three releases of *XPA* during the period of this grant (May 1999 as part of saord 1.9, and two minor releases containing bug fixes). As a result, *XPA* now is part of major astronomical projects such as Chandra, XMM, DEIMOS, and HEASARC/ftools, and is in use by astronomers all over the world.

2.3 Development of New Image Display Software

Along with support for new messaging software, we also sought to move astronomical image display beyond the bounds of any particular program and even beyond the bounds of the Unix operating system that has been so central to astronomical software over the past decade and a half. Towards this end, we developed *SAOTk*, an integrated set of *Tcl/Tk* widgets for astronomical imaging and data visualization (see Table 1).

Table 1: SAOTk Widgets

Image	FITS viewer for a single image or a mosaic of images
Colorbar	control of imaging color environment
Panner	thumbnail image for position control
Magnifier	magnified view of a section of the image

The Image widget is the central component of the *SAOTk* widget set. It used to display both *FITS* images and binary tables. Multiple extension *FITS* files are also supported. The Image widget is designed specially to support the display of large amounts of data. For images, the rendering time is constant, and is based on the size of the display window, not the size of the data. Thus, the widget is capable of rendering very large images (8k x 8k or larger), or a large mosaic of images (such as 36 images, each 2k x 4k). Since all rendering is done directly from the raw *FITS* data, memory requirements are very small.

For *FITS* binary tables, the rendering time is based on the number of filtered rows or events. Since the image is binned *on the fly* before being rendered to the screen, memory requirements again are very small. Of course, if the data are sorted by position, binning and rendering times are much faster, since the widget can calculate which segments of a table contain events that need to be processed.

The Image widget supports display of a mosaic of images, where each image segment is a separate *FITS* image. These segments may be stored in a single multi-extension *FITS* file, or may be located in a number of different files. The widget uses *FITS* mosaic keywords to place each image in its proper place in the overall image space.

One special feature of the Image widget is the ability to rotate, orient, and zoom an image to align its display to an arbitrary world coordinate system (WCS). This feature is very useful for making comparisons between images from different instruments or telescopes that have different coordinate systems or WCS parameters.

Another important feature of the Image widget is its support for *PostScript* level 1 and level 2 printing. This is not a screen capture process, but a full-featured *PostScript* driver. A variety of document sizes are available, and grayscale and color imaging are supported. Level 2 output is compressed via `RLE` and encoded via `ASCII85`, so the size of the output *PostScript* data is usually very small.

The *SAOTk* widgets are designed to be incorporated easily in any standard *Tcl/Tk* environment to build custom data analysis and visualization applications. We have used

them to develop *DS9*, our code-name for the next version of the *SAOtng* display program. This *Tcl/Tk* application also incorporates the new version of *XPA* (described below) to allow external processes to access and control its data, GUI functions, and algorithms. *DS9* supports all major capabilities in the current version of *SAOtng*, including direct display of *FITS* images and binary tables, multiple frame buffers, region/cursor manipulation, user-defined scale algorithms and colormaps, and easy communication with external analysis tasks. It also supports advanced features such as mosaic images, arbitrary zoom, rotation and pan, and a variety of coordinate systems (including Image, CCD, Detector, and WCS).

DS9 is implemented as a stand-alone application that requires no installation or support files. Binary versions of both *DS9* and *SAOTK* exist for Sun Solaris 2.5/2.6, Linux, SGI, and Alphas, as well as a port to Microsoft Windows. All versions and platforms support a consistent set of capabilities.

Our *DS9* software has not yet been "officially" released, as we still are working on essential support for the IRAF analysis system. However, due to demand from users, we have long been making *DS9* available informally via the WWW and anonymous ftp, with the result that it is already being used heavily within the astronomical community, even in this pre-release form. In fact, *DS9* has been adopted by Chandra and XMM as a replacement for *SAOtng*, and we are working with other groups (such as the DEIMOS Project a Keck/Lick) to tailor *DS9* to their needs. With support for its further development being provided by SAO, we are confident that *DS9* will soon become the *de facto* image display standard in the astronomical community.

3 WWW Access to Project Software

All of the software developed under this grant (along with extensive documentation) is freely available via the World Wide Web. For more information (including down-load instructions), please visit the following sites:

```
# general information about the SAO R\&D group
http://hea-www.harvard.edu/RD/

# information about the SAOtng image display program
http://hea-www.harvard.edu/RD/saotng/index.html

# information about the XPA messaging system
http://hea-www.harvard.edu/RD/xpa/index.html

# information about the ds9 image display program
http://hea-www.harvard.edu/RD/ds9/index.html
```