Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

Providing a Common GUI to Image Processing Tasks under PCIPS

O. M. Smirnov

Institute of Astronomy of the Russian Academy of Sciences
48 Pyatnitskaya Str., Moscow 109017 Russia

N. E. Piskunov

Joint Institute for Laboratory Astrophysics
University of Colorado, Boulder, CO 80309-0440

1. Introduction

The PCIPS image processing system (Smirnov & Piskunov 1994) has been in development since 1991; the current version is 2.01. PCIPS is a PC-based platform for all kinds of astronomical image processing and data-analysis. Key features of the system include: (1) an intuitive graphical user interface (GUI), (2) all functionality is provided by external application modules, which are easily added to the system as they are developed, (3) there is an application program interface (API) for development of custom applications, and (4) the code runs on practically any Intel-based PC.

As of now, several application packages have been developed under PCIPS. These include: (1) basic image processing operations (mathematics, statistics, filtering, fitting, geometric transforms, export and import of standard data formats, etc.), (2) PCDAOPHOT II, a GUI-equipped version of the DAOPHOT II (Stetson 1991) stellar photometry package (finding objects, aperture photometry, PSF construction, simultaneous PSF fits), (3) CCD/Echelle processing package (flat fielding, cosmic ray hit detection, finding and extracting spectral orders), (4) a spectral analysis package (dispersion curve fitting, continuum normalization, multiple line profile fits), (5) a Fourier analysis package (FFTs, power spectra, filtering in the Fourier domain), and (6) DASHA, a globular cluster/field photometry package (Smirnov & Ipatov 1995).

All of these very different packages have a consistent and powerful GUI that is common to all PCIPS applications. This paper reviews some issues that arose when developing the GUI.

2. Primary Design Goals

Several design goals were formulated at the initial stage of development. First and foremost was *expandability:* the system would not be limited to a set of built-in image processing operations. All image processing functionality was to be provided by external modules (called *applications*), loadable on demand. The upshot of this decision is that the user interface for the applications became the interface of the whole system, since there is not much the user would do in PCIPS apart from running applications. Thus the second design goal emerged—

making the user interface consistent across all applications, both existing ones, and those yet to be written. A third goal was making the interface as simple and easy to use as possible (substituting development time and CPU load for astronomer, or end user, time), while still providing enough features to support a great variety of applications.

A simple solution such as a single (usually huge) form with all the possible parameters, to be filled out by the user when starting an application, was unacceptable. A lot of applications (or, to be exact, their users) would benefit from a truly interactive operation. Finally, a fourth goal was to provide future compatibility with a visual programming metaphor (Smirnov & Piskunov 1993) to be implemented in an advanced version of pcIPS, which requires elements of the user interface to be modular and connectable by pipes.

3. GUI Implementation

Who Provides the GUI? A fundamental design decision was to off-load the task of providing a user interface from individual applications onto the system itself. The decision killed two birds with one stone: interface consistency was assured, since only one common program was handling the GUI, and application development time was cut dramatically by freeing the programmer from the mundane task of baby-sitting the user.

A pcIPS application interacts with the system using a set of AIS (Application Interface Services, the application program interface, or API, of pcIPS) calls. It never communicates with the user directly. Instead, pcIPS translates AIS calls into user interface objects, maintains them, and reports the results back to the application. At the core, AIS is object-oriented. However, we did not want to implement a true object-oriented (OO) application interface in C++, for the sake of compatibility with non-OO languages such as C and FORTRAN. The application programmer works with the more conventional concept of handles, while inside pcIPS the handles are converted into true objects.

Example 1: Viewports. To display images on the screen, an application first makes one AIS call to create a viewport object. Next, it calls another function to display an image in the viewport. pcIPS displays the image, and automatically outfits the viewport with GUI elements that provide visualization tools (zooming, re-coloring, etc.) The application need not know anything of the GUI that goes with a viewport; all it needs are two handles returned to it by AIS, one to a viewport object, and the other to an image object. The functionality of a viewport is completely hidden inside the viewport object.

Example 2: Dialog Boxes. Another example is user input. When the application needs a set of parameters, it calls AIS to create a dialog box object, then requests the necessary parameters, then tells AIS to realize the dialog box. pcIPS decides where and how to display the dialog box, outfits it with standard buttons and tools, allows the user to enter the parameters (providing all sorts of useful gizmos to make the task easier), then reports their values back to the application. All the application has to specify are the names and types of the parameters (and where to store their values), and the name of the dialog box, if necessary. All it needs to know from AIS is the handle of the dialog box object, which it uses in the call to realize the dialog box. It does not need to know what a dialog box looks like, and what the user can do with it.

A big advantage of isolating the user from the application in this manner becomes apparent when we consider the visual programming metaphor, in which individual applications are connected by pipes that move data between them. With AIS, it does not matter to an application whether a parameters was actually requested from the user, or arrived via a pipe from another application. Therefore, all existing packages need little or no modification to adapt to the new metaphor.

4. Future Plans

A completely revised version of PCIPS is currently in development, targeted at UNIX/X11, Windows NT and DOS/Windows95. Some features of the new system are: (1) an unconventional object-oriented database, supporting all sorts of data objects and relations between them. It has a fully extendible data interface, and new data classes will be easily derivable from existing basic classes (i.e., images, arrays, tables, lists, etc.); (2) more visualization capabilities, implemented as properties of each data class (and thus easily extendible as well); (3) an overhauled API, with a separate, truly object-oriented C++ version, with support for more user interface objects and features; and (4) use of the visual programming metaphor.

4.1. Feedback

PCIPS is a commercially distributed product. For details, contact Oleg Smirnov (e-mail: *oms@inasan.rssi.ru*). We welcome any comments and questions both regarding the current version of PCIPS, and the one in development. We are maintaining an e-mail distribution list for the PCIPS Electronic Bulletin (Peb), ask Oleg Smirnov if you want to sign up. Also, visit our anonymous ftp host, *pcips.inasan.rssi.ru* (193.232.30.12). The directory */pcips* contains everything related to the system. A trial version can be picked up from */pcips/trial*.

Acknowledgments. We wish to thank ST ScI for providing the financial support that made this presentation possible. DAOPHOT II was ported with the kind permission of the original author, Dr. P. Stetson.

References

Smirnov, O. M., Piskunov, N. E. 1993, in Astronomical Data Analysis Software and Systems II, ASP Conf. Ser., Vol. 52, eds. R. J. Hanisch, R. J. V. Brissenden, & J. Barnes (San Francisco, ASP), p. 208

Smirnov, O. M., Piskunov, N. E. 1994, in Astronomical Data Analysis Software and Systems III, ASP Conf. Ser., Vol. 61, eds. D. R. Crabtree, R. J. Hanisch, & J. Barnes (San Francisco, ASP), p. 245smirnov2

Stetson, P. B. 1992, in Astronomical Data Analysis Software and Systems I, ASP Conf. Ser., Vol. 25, eds. D. M. Worrall, C. Biemesderfer, & J. Barnes (San Francisco, ASP), p. 297