Astronomical Data Analysis Software and Systems VI
ASP Conference Series, Vol. 125, 1997
Gareth Hunt and H. E. Payne, eds.

Java, Image Browsing, and the NCSA Astronomy Digital Image Library

R. L. Plante,[1] D. Goscha,[1] R. M. Crutcher,[1] J. Plutchak,[2]
R. E. M. McGrath, X. Lu, and M. Folk

National Center for Supercomputing Applications (NCSA), University of Illinois, Urbana, IL 61820, E-mail: rplante@ncsa.uiuc.edu

Abstract. We review our experience, including some lessons learned, with using Java to create data browsing tools for use with the Astronomy Digital Image Library (ADIL) and related digital library projects at NCSA. We give an overview of our Image Data Browser, a generalized tool under development through a collaboration with the NCSA/NASA Project Horizon in support of access to earth and space science data. Emphasis has been placed on a design that can support a variety of data formats and applications both within and outside of astronomy. ADIL will use this tool in a variety of guises to browse and download images from the Library's collection. We see such a tool filling an important niche as a pipeline from a data repository to specialized, native data analysis software (e.g., IRAF, AIPS).

1. Introduction

Digital library technology for accessing earth and space science data has been the focus of a collaborative effort at NCSA known as *Project Horizon,*[3] a cooperative agreement between NASA and the University of Illinois. Project Horizon includes scientists from the fields of astronomy, atmospheric science, and Web server technology as well as researchers from the NCSA HDF group and the University of Illinois Digital Library Initiative. As a participant in Project Horizon, the NCSA Astronomy Digital Image Library (ADIL)[4] has served as a testbed for solutions to accessing large datasets over the Web.

One of the problems Project Horizon has been studying is how one most effectively browses large image datasets over the Web. One early solution included simple static digests of the data which might include a GIF or JPEG preview of the image. More interactivity was brought to the process with the server-side, on-the-fly generation of images driven by inputs from an HTML form. One shortcoming of this latter method is the load put on the server to

---

[1]Astronomy Department, UIUC

[2]Atmospheric Science Department, UIUC

[3]http://www.atmos.uiuc.edu/horizon/

[4]http://imagelib.ncsa.uiuc.edu/imagelib

generate these custom views. This was demonstrated at UI with early versions of the very popular Weather Visualizer[5] which allowed users to download custom visualizations of weather data. Another problem is that there is limit to the level of interactivity that HTML forms can provide.

The advent of Java provides a way to address some of the problems of data browsing. As a network-smart, object-oriented language, Java allows one to develop highly interactive applications that can interact with a server via a Web browser. Java also brings the advantage of platform-independence to application development. For the specific problems of image data browsing, Java allows a server to pass on some of the CPU chores to the individual clients.

Project Horizon's early explorations with Java produced several examples of how Java could be used to interact with scientific data:

- The ADIL[6] uses a Java applet for browsing large FITS images such as those from the NRAO VLA Sky Survey, providing zooming ability and coordinate tracking.
- The Daily Planet uses a Java version of the Weather Visualizer[7] to visualize a variety of weather data as multiple overlayed images which users may turn on and off.
- The NCSA Hierarchical Data Format (HDF) group has released a FITS Data Browser[8] for browsing local FITS images either as a visual image or as a spreadsheet.
- The HDF Data Browser[9] is also a Java application specially suited for browsing the hierarchical structure of HDF files.

With these early successes, it became clear that each of the Java applications contained features that would be useful for browsing all kinds of scientific images, regardless of data format or field of study with which the data is associated. Project Horizon is now working to combine those features into a single package of reusable Java code called the *Horizon Java Image Data Browser Package*.[10]

2. Horizon: a Java Package for Browsing Images

The Horizon Java package will be a collection of image data browsing solutions in the form of reusable Java classes, along with ready-to-use Java applets and applications. We note that the aim of the Horizon package is *not* meant to duplicate or replace the role of visualizing tools such as SAOimage or Aipsview or of image processing packages such AIPS, IRAF, and others. Instead, Horizon focuses specifically on *data browsing*, that stage of data handling before and

---

[5] http://covis1.atmos.uiuc.edu/covis/visualizer/

[6] http://imagelib.ncsa.uiuc.edu/project/javadoc/96.JC.23.06

[7] http://covis.atmos.uiuc.edu/java/weather0.4/Weather.html

[8] http://hdf.ncsa.uiuc.edu/fits/java

[9] http://hdf.ncsa.uiuc.edu/hdf/java/jhv

[10] http://imagelib.ncsa.uiuc.edu/imagelib/Horizon/

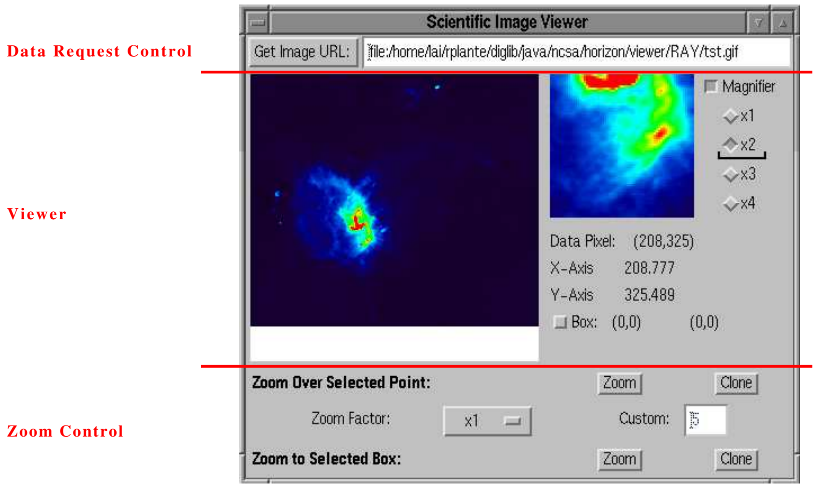

Figure 1. A sample Horizon application for viewing images. Horizontal lines delimit the three Horizon objects that make up the viewer.

after processing and analysis with native software tools. Specifically, the goals of Horizon are: (*i*) to provide a "first-cut" look at image datasets from the network or local disk, (*ii*) to act as a smooth pipeline from data repositories to specialized native software (e.g., IRAF, AIPS, IDL, etc.), (*iii*) to serve as a simple, cheap, and platform-independent image visualizer with basic browsing features for those without access to more sophisticated software, (*iv*) to provide basic image visualization in a collaborative environment, and (*v*) to provide a programming paradigm for adding new features and modifying existing features easily through generalized abstract classes.

Some of the features of the package include: (*i*) basic 2D visualization of multidimensional data, (*ii*) reading/writing data of (in principle) any data format from the network or local disk (initially FITS, HDF, GIF, and JPEG), (*iii*) zooming, sub/super-setting, and animation, (*iv*) pixel value and coordinate position display, (*v*) linking pixels with text-based data, (*v*) spreadsheet browsing, (*vii*) color fiddling, progressive image transmission,[11] (*viii*) overlaying of multiple images, (*ix*) support for collaborative session via the NCSA Habanero Collaboration Tool,[12] and (*x*) easy adaptability and extensibility.

The Horizon package will include applets and application that are ready-to-use or can easily be adapted. For more specific applications, the Horizon classes can be mixed and matched to create new specialized tools.

3. Horizon Design Overview and Implementation Considerations

The Horizon package provides a variety of widgets that can be assembled into a variety of applets and applications. As an example, Figure 1 shows a Viewer ap-

[11] http://www.sal.wisc.edu/~jwp/can.html

[12] http://www.ncsa.uiuc.edu/SDG/Software/Habanero/HabaneroHome.html

plication made up of three Horizon components, a Viewer Panel and two Control Panels. The Control Panels interact with the Viewer Panel (e.g., get selected pixels, request views of subregions, etc.) via public methods of the abstract Viewer class; thus, with very little programming effort, one could remove this particular Viewer and replace it with a specialized version that obeys the same interface. Control Panels will provide specialized functionality such as zoom control, animation, colormap fiddling, and data transfer.

In addition to the GUI components, the Horizon package also provides important classes that work behind the scenes. The most important is the Viewable interface, which serves as the format-independent layer on top of the format-specific reader, providing methods for extracting data and visualizations of the multidimensional data beneath it. Also important is the set of classes that support world coordinate systems associated with the data.

For all its advantages, Java also has limitations which we are trying to keep in mind as we develop the Horizon package, and our experience has suggested some general considerations. For more details on the design of the Horizon package, see the Horizon Design white paper (Plante et al. 1996).

4. Recent Progress

Early fall 1996 saw our first alpha releases. The first was a demo release of classes to illustrate a few of the basic viewers we are currently working on. This was soon followed by the first release of source code in the form of the FITSWCS package. Actually a translation of Mark Calabretta's WCSLIB library, this Java package provides support for the FITS World Coordinate System standard proposed by Greisen & Calabretta (1996). Late fall 1996 will see a major alpha release that will include the basic source code and documentation for building the basic image viewers, including generalized support for coordinates. A full production release is expected by the end of 1997.

In conclusion, we see the Horizon Package filling a variety of niches along the scientific data pipeline through its ability to provide different solutions to different people. Data providers will easily be able to create browsing applets customized for a particular collection of data for a particular audience. Astronomers could use Horizon tools with NCSA Habanero to remotely monitor observations in a collaborative session. Educators and students without access to high-powered graphics workstations could use Horizon applications as a "poor-person's visualizer" of scientific data.

References

Greisen, E., & Calabretta, M. 1995, The draft proposal for WCS syntax in FITS headers[13]

Plante, R., Goscha, D., & Plutchak, J. 1996, Horizon Design White Paper[14]

---

[13] ftp://fits.cv.nrao.edu/fits/documents/wcs/wcs.all.ps.Z

[14] http://imagelib.ncsa.uiuc.edu/imagelib/Horizon/DesignWhitePaper.html