Astronomical Data Analysis Software and Systems VII
ASP Conference Series, Vol. 145, 1998
R. Albrecht, R. N. Hook and H. A. Bushouse, eds.

The NCSA Horizon Image Data Browser: a Java Package Supporting Scientific Images

R. L. Plante,[1] W. Xie,

J. Plutchak,[2] R. E. M. McGrath and X. Lu

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

Abstract. The NCSA Horizon Image Data Browser is a Java package which includes ready-to-use applets and applications as well as reusable classes for building new applications for browsing scientific images locally or over the network. We present the current status of the development of this package focusing on three general features: 1) flexible support for metadata through a schema-independent design, 2) the implementation of our metadata model to support world coordinate systems, and 3) support for use of Horizon applications within the NCSA Habanero Collaborative environment.

1. Introduction

The NCSA Horizon Image Data Browser[3] is a Java package for browsing scientific images. It is a collection of ready-to-use applets and applications as well as a toolkit of reusable classes that can be mixed and matched to create new applications. The basic goal of the package is to provide tools that can give a "first-cut" look at image datasets read from the network or local disk and act as a smooth pipeline from data repositories to specialized native software. For many users, particularly non-scientists, it can serve as a cheap, platform-independent tool for visualizing real scientific data. The design of the package is independent of data format allowing for the support of multiple formats. Initially, the package will come with support for FITS, HDF, GIF, and JPEG. The package provides such features as zooming, animation, pixel value and coordinate position display, spreadsheet display, and color fiddling. (For a full discussion of the goals of the package see Plante et al. 1997.)

In this paper, we will focus on three specific design features the Horizon package: the support for metadata, world coordinate systems, and collaboration.

---

[1]Astronomy Department, UIUC

[2]Atmospheric Science Department, UIUC

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

2. Metadata

Metadata—data about data—are an important part of scientific data. They are the ancillary information that accompanies some primary data set, allowing it to be interpreted intelligently by scientists and the software they use. It plays a critical role when data must be transported between different software systems or formats. In the Horizon package, metadata are a set of name-value pairs where the name is of type `java.lang.String` and the value, `java.lang.Object`.

An important assumption that always exists when handling metadata is the agreement between the creators of metadata and its users on the mapping of a metadatum's name to its value type (e.g., `Double`, `String`, etc.) and its conceptual meaning. A set of agreed mappings of names to types and meanings is often referred to as a *schema*. Different formats and different scientific fields often use different schemata to represent metadata.

At the center of Horizon's support for metadata is the `Metadata` class (Plante 1997). Its design is schema-independent and takes into account that there may be multiple schema in use within a single application. Furthermore, metadata can be hierarchical, so the `Metadata` class allows direct access to any individual metadatum in the hierarchy, or whole sections of the hierarchy. Horizon also supports array data as metadata. The following code illustrates both the array and hierarchical features:

```
Metadata mdata;
...
// retrieve all metadata for coordinate axis 0
Metadata axmdata = (Metadata) mdata.getMetadatum("CoordinateSystem.Axes[0]");

// retrieve a specific coordinate axis sub-metadatum directly
String name = (String) mdata.getMetadatum("CoordinateSystem.Axes[0].name");
```

Other features of the `Metadata` class include support for default values that can be overridden but not erased; this feature is used a way of giving read-only access to metadata that are to be shared with many objects. The class also supports on-demand loading of values; if the process of loading metadata is expensive (e.g., they are read from the network), one would prefer that they only be read if the user explicitly requests them.

The Horizon package defines a "slim" metadata schema which it uses to do basic visualization. Part of the job of the format-dependent reader is to convert selected native metadata into the "horizon" schema. Higher-level Horizon classes use this schema to interact with the image data in a format-independent way, including extracting chunks of data and displaying positions in the data's world coordinate system.

3. World Coordinate Systems

A scientist usually wants to know where a piece of data exists in physical or conceptual space–its *World Coordinate System*–rather than its location in some data array. In the Horizon design, data can exist in a data space of arbitrary number of dimensions which maps to a world coordinate space of the same or

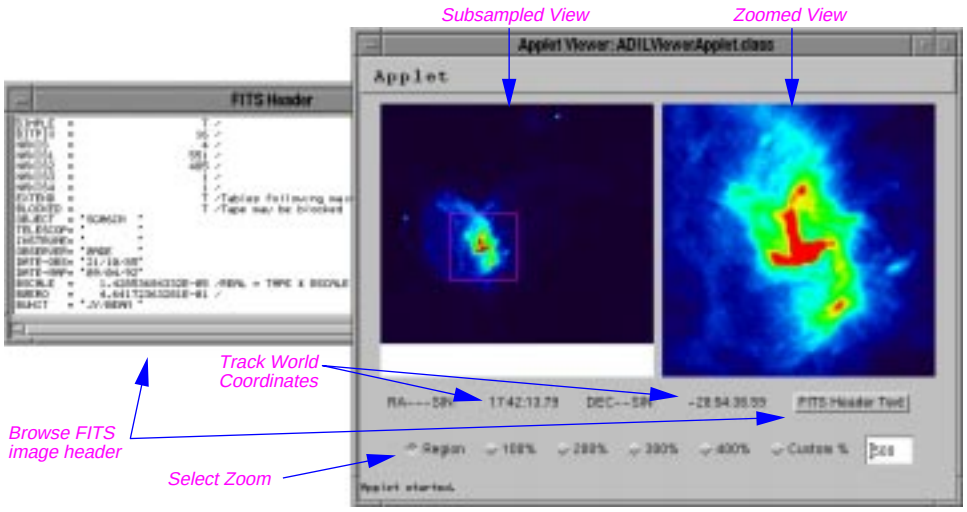

Figure 1. The ADILBrowser Applet: a viewer for browsing images in the NCSA Astronomy Digital Image Library (ADIL).

different number of dimensions. The transformation between data voxels and coordinate positions can be non-linear and even non-reversible.

The two Horizon world coordinate classes used most by the application programmer are `CoordinateSystem` and `CoordPos` (Plante 1997). A `CoordinateSystem` object can be obtained for a particular dataset through its format-independent interface. This object has full knowledge of the dataset's coordinate system and its mapping to the data voxels. Thus, the programmer can give the `CoordinateSystem` a data voxel and get back a `CoordPos` object, an encapsulation of the corresponding coordinate position. The reverse transformation is also supported. One special feature of the `CoordPos` object is that it knows how to print itself; e.g., it knows to print right ascension as *HH:MM:SS.S*, galactic longitude in decimal degrees, and velocity with the appropriate metric units. These default formats can be easily overridden.

The transformation engine within the `CoordinateSystem` class is the `CoordTransform` class, used for converting between two coordinate systems. Horizon comes with a number of specialized `CoordTransforms` that do things like spherical projections, switching between celestial and galactic coordinates, etc. (Some of these have been implemented using FITSWCS, a Java port of the WCS library by Calabretta, see Greisen & Calabretta 1995.) Multiple `CoordTransform` objects can be strung together to produce complex transformations. Thus, every `CoordinateSystem` maintains an internal stack of `CoordTransform` objects through which it can pass data voxels or coordinate positions. Users of the `CoordinateSystem` can attach additional `CoordTransform` objects to the system at any time for on-the-fly switching of coordinate systems.

The combination of `Metadata` and `CoordTransforms` is Horizon's recipe for a smart `CoordinateSystem`. When a `CoordinateSystem` is constructed, it uses `Metadata` gotten from the dataset to configure the needed `CoordTransform` objects. Included in the `Metadata` are `AxisPosFormatter` objects (one for each

axis) that control how coordinate positions are printed out. If the user attaches an additional `CoordTransform` object, say to switch from celestial to galactic coordinates, the `Metadata` are consulted to determine which axes the transformation should be applied to. The `Metadata` are then transformed as well to reflect the new coordinates system; thus, for example, the format can automatically be switched from *DD:MM:SS.S* to decimal degrees.

4. Collaboration

Collaboration is supported within Horizon via the NCSA Habanero package,[4] a collaborative environment written entirely in Java. Habanero allows a group of participants located on different machines to interact with a single tool as a group. (See Crutcher et al. 1998 in this volume for more details on the use of Habanero with astronomical applications.) The Horizon classes will come with special hooks that allow them to be run within Habanero environment in a collaborative way.

Figure 1 from Crutcher et al. (1998, this volume) shows an example of running a Horizon applet, the ADILBrowser (upper right corner; see also Figure 1 of this article) within a Habanero session. Participants use the applet to browse images located in the NCSA Astronomy Digital Image Library.[5] The applet tracks world coordinates and allows zooming into subregions of the image. The remaining windows are part of the Habanero environment which include a session manager, a chat window, and a white board.

Acknowledgments. The Horizon development team thanks Jef Poskanzer, Mark Calabretta, and Tom McGlynn for contributed code. The Horizon Image Data Browser package is supported in part by Project Horizon, a cooperative agreement with NASA, and by the NASA Applied Information Systems Research Program (96-OSS-10).

References

Crutcher, R. M., Plante, R., & Rajlich, P. 1998, this volume.

Greisen and Calabretta 1995, Representations of Celestial Coordinates in FITS[6].

Plante, R. 1997, Supporting Metadata and Coordinate Systems for Scientific Data in the Horizon Java Package[7], white paper.

Plante, R., Goscha, G., Crutcher, R., Plutchak, J., McGrath, R., Lu, X., & Folk, M. 1997, in ASP Conf. Ser., Vol. 125, Astronomical Data Analysis Software and Systems VI, ed. Gareth Hunt & H. E. Payne (San Francisco: ASP), 341.

---

[4]http::://www.ncsa.uiuc.edu/SDG/Software/Habanero

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

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

[7]http://imagelib.ncsa.uiuc.edu/Horizon/docs/articles/CoordsAndMetadata