16.3 COLLABORATIVE VISUALIZATION AND COMPUTATION IN THE EARTH SCIENCES USING VisAD

William Hibbard[1]*, Steve Emmerson[2], Curtis Rueden[1], Tom Rink[1], David Glowacki[1], Nick Rasmussen[1], David Fulker[2] and John Anderson[1]

[1]Space Science and Engineering Center
University of Wisconsin - Madison

[2]Unidata Program Office

1. INTRODUCTION

We described our plans for a Java implementation of VisAD at this conference two years ago (Hibbard, Anderson and Paul, 1997) and demonstrated it at this conference last year. Now it is freely available via the world wide web from http://www.ssec.wisc.edu/~billh/visad.html and ready to serve as the foundation for development of interactive meteorology, oceanography and hydrology systems. This paper explains the advantages of using VisAD in your system development efforts. These can be summarized as:

1. Its use of Java will make your systems portable as Java matures.

2. Its general numerical data model enables your systems to add new data sources to integrate with other systems based on VisAD.

3. Its integration of metadata for units, coordinate systems, regular and irregular sampling topologies, missing data indicators and error estimates supports complex computational and visualization functions in your systems.

4. Its support for Java2D, Java3D, direct manipulation and immersive virtual reality helps your systems to adopt these visualization modes.

5. Its use of distributed object via Java RMI makes helps you develop distributed and collaborative systems.

6. Its extensibility enables you to add computational and visualization functions not implemented in VisAD.

7. Its easy installation allows your systems to also be easy to install. Its freely available source code, documentation and application examples provide confidence in long term viability. We have no objection to commercial systems that use VisAD via its application programming interface (API).

In order to give a clear picture, we also summarize the difficulties of using VisAD as a basis for your system developments:

1. Java is not yet mature, meaning that widespread portability and efficient compilers are still under development. VisAD uses Java 1.2 and Java3D, which are expected to come out of beta release only shortly before this conference in January 1999. While these can run in web browsers using Java plug-ins, they will not be standard in web browsers for a while.

2. VisAD achieves its generality through its high level of abstraction, which confronts system developers with a novel API. The API will support almost any function your system needs, but it can be a sort of puzzle to figure out how.

In the next two sections we address these issues in more depth. Then we discuss the basic structure of systems based on VisAD and how VisAD is designed for the future evolution of scientific computing on the Internet.

2. JAVA MATURITY

In order for Java to provide true platform dependence, it must include APIs for every common computational and communication function. This is an enormous task and is taking some time. However, Java 1.2 provides an adequate set of functions for real applications and will be released to Java licensees by the end of 1998. Ports to a wide variety of platforms should be available within three to six months after that. This includes Java3D, used by VisAD for 3-D visualization functions.

At the JavaOne conference in March 1998 James Gosling, inventor of Java, said "this is the year we fix Java performance." He was referring to the HotSpot compiler which will be part of Java 1.2. Furthermore, IBM is applying their considerable

* Corresponding author address: William Hibbard, SSEC, 1225 W. Dayton St., Madison, WI 53706; email address: whibbard@macc.wisc.edu

expertise in compiler design to Java. IBM is reported to have more programmers than Sun working on Java development. HP is also applying considerable effort to Java development as evidenced by their development of an independent Java implementation (required by the Java license to pass compatibility tests) designed to reduce resource use. Furthermore, there are no fundamental reasons why Java performance must remain worse than older languages. We are confident that Java performance will match other languages within a year or two.

Java was first introduced to most of us via its web applets. Even as our use of Java matures to Java distributed objects, the web and its browsers will continue to play an important role for user interface and to define entry points into the network of active, persistent Java distributed objects. However, browsers tend to lag behind Java version releases by about a year, which motivates system developers to use older Java versions. Java plug-ins provide a short term solution to this problem for motivated users. Furthermore, it is a reasonable strategy to direct system development efforts toward technology that will be available in the future, recognizing that technology will mature by the time new systems are ready for end users.

3. USABILITY OF THE VisAD API

VisAD can be puzzling because, for example, it does not include any classes specifically designed for images or grids. Rather, it recognizes that images and grids are instances of functions from n-dimensional spaces to m-dimensional spaces. Generally for images, n = 2 and m = the number of image bands. For grids, n = 2 or 3 and m = the number of atmospheric state variables. The domain coordinates for images are often named line and element whereas the domain coordinates for grids are often named row, column and level. Images and grids include coordinate transforms between their domain coordinates and (latitude, longitude, altitude) coordinates. Image radiances may require calibration, which can be expressed as 1-dimensional coordinate transforms. Grid values such as temperature and pressure may include units such as Kelvin and millibars, and transformed domain coordinates latitude, longitude and altitude may also include units.

VisAD defines classes for functions from n-dimensional spaces to m-dimensional spaces that include all these coordinate transform and unit metadata. Thus developers who use VisAD as the basis of their systems must translate their concepts of images and grids to these more abstract VisAD classes. This can be a bit puzzling, at least until developers have had some practice using the general numerical data model.

However, the great advantage of using VisAD's abstract data model is that it makes it easy to extend your systems to new data. For example, if you need to overlay your grid displays with the output of analysis algorithms, the abstract data model will have a way to express that output and then displaying it will be straightforward. Similarly, if you need to implement analysis functions that combine regular grids with irregular grids, the abstract data model supports such operations.

Metadata for units, coordinate transforms, sampling geometry and topology, missing data indicators and error estimates are all part of VisAD data objects and integrated into computational and visualization operations on data. This provides high level functionality. For example, when two images are added coordinate transformation and resampling are done implicitly if necessary. While this is unfamiliar to many developers, it allows complex functions to be expressed simply.

Since system developers may find the VisAD metadata unfamiliar and confusing at first, most forms of metadata are optional. Thus, developers can begin by implementing images as simple rectangular arrays of pixels, then add coordinate transforms, units, complex sampling geometries, missing data and error estimates one by one as necessary.

As developers become adept, they will be able to extend VisAD classes in order to define new coordinate transform algorithms and new interpolation algorithms for resampling. They will also be able to extend VisAD classes in order to add new kinds of metadata not included by VisAD.

The VisAD distribution includes a large and growing number of application examples that illustrate how to use VisAD and can serve as starting points for developing similar applications.

4. THE STRUCTURE OF VisAD APPLICATIONS

A program or system based on VisAD is structured as an event-driven network of data, computational, display and user interface objects. These objects may be distributed among different computers on the Internet. When a data object is connected to a display object, a depiction of the data object appears in the display and is updated whenever the value of the data object changes. When a data object is connected to a computational object, the computation is triggered for execution whenever the value of the data object changes. Computational objects can update values of any accessible data objects. Furthermore, when a data object is connected to a display object via what we call *direct manipulation renderers*, users may redraw the depiction of the data object in order to change its value. User interface objects may be connected to data, display or computational objects.

This structure has far-reaching consequences for designers of interactive and collaborative systems. It is relatively easy to build collaborative applications by simply connecting display and user interface objects from different computers to a shared set of data and computational objects. It is also easy to build distributed computations by building a network of data and computational objects residing on different computers. Computational steering techniques can be implemented by connecting a user interface object to a data object that is connected as an input to a computational object.

Very sophisticated interaction techniques can be designed by connecting data objects to display objects via direct manipulation rendering, and then connecting these data objects as inputs to computations as display controls. For examples, this can be used to implement a rubber band box for zooming in to a display sub-region. Or it could be used to implement moveable color slices through a 3-D volume visualization.

5. THE FUTURE OF SCIENTIFIC COMPUTING

Users view the world wide web as a single shared space of web pages and are generally unaware of the identities of individual computers containing web pages. Java extends this to all computing resources, so that users and programs will see the entire network as a single shared space of objects without much attention to the identities of individual computers. Java also allows objects to move around the physical network. Java applets are a simple way to exploit this capability, but much more is possible in a distributed object environment. For example, when data objects move between computers they can include the necessary code for accessing their contents.

This changed way of programming the Internet will be essential as the Internet grows. Attendees of JavaOne 98 already own rings that run Java. In the future Java will run in our telephones, credit cards, televisions, microwave ovens, dog licenses, etc.

In the science world we are already used to the idea that most scientific data is available on the network via ftp or the web. This will evolve into an active network of data, computational, display and user interface objects. Coupled models and other distributed computations will be routinely running. Scientists will be able to:

1. Locate objects in this network via their web browsers.

2. Connect display objects to them in order to visualize their behavior.

3. Connect their own data, computational and user interfaces objects to them in order to analyze their results or control their behavior (subject to access control, of course).

4. Clone the displays and user interface objects of other scientists in order to collaborate.

VisAD is designed with this vision of the future in mind. Perhaps the most compelling reason to use VisAD as the basis of your systems is so they will be ready for this future.

6. ACKNOWLEDGMENTS

We wish to thank Tom Whittaker, Tommy Jasmin, Glen Davis, Russ Rew, John Caron, Tom Yoksas, Don Murray, Steve Pietrowicz, Chris Heistad, Larry Jackson, Bob Rabin, Wilt Sanders and many others for helpful discussions. We wish to thank Joe Bredekamp of NASA (NAG5-3943), Larry Smarr of NCSA and Cliff Jacobs of NSF (Unidata grant) for their support.

7. REFERENCES

Hibbard, W., J. Anderson, and B. Paul, 1997; A Java and World Wide Web implementation of VisAD. Preprints, Conf. Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology. Long Beach, Amer. Meteor. Soc., 174-177.