A Preliminary Evaluation of the VisAD Visualization Toolkit

These materials were developed with the support of the Shodor Education Foundation, 923 Broad Street, Durham, NC 227705 and the National Science Foundation Grant DUE-0127488.

Prepared by:
Mike Murphy
Undergraduate, Department of Computer Science
Clemson University

For:
Shodor Educational Foundation, Inc.

Originally completed as a working draft on 16 June 2003.

Abstract

VisAD is an open-source visualization component toolkit for the Java programming language. This toolkit provides both two- and three-dimensional visualization support, automatic support for panning and zooming images, and a flexible data structure hierarchy. Support for distributed architectures is available "out of the box," as are coordinate frameworks for a number of different visualization types. In addition, the package is designed to be as general as possible, in order to support many different problem domains. Because of the design of the Java language, VisAD applications may be designed to be platform-independent. Thus, VisAD has significant possibilities for designing portable and scalable scientific visualization packages.

Unfortunately, the documentation for VisAD is somewhat nebulous and disconnected, resulting in a very steep learning curve. The large number of Java classes in the package further complicates comprehension, especially when taken in concert with the frequently missing, incomplete, or incorrect component-level documentation. Furthermore, VisAD relies on Sun Microsystems' Java3D technology, which is only supported on a limited subset of Java-enabled platforms. These issues present significant concerns for designing, implementing, and supporting VisAD applications.

Table of Contents

Introduction

The Visualization for Algorithm Development, or VisAD, visualization component package is a collection of Java classes that provide a Java developer with a portable component library to support scientific visualizations. VisAD is an open-source toolkit, released under the Free Software Foundation's GNU Lesser General Public License and available at no cost.

VisAD is one of several potential toolkits available to the Shodor Educational Foundation for use in developing client-side scientific visualization packages for the K-12 and collegiate education communities. Because of the cross-platform nature of Java, VisAD is an exceptionally strong candidate for consideration as a base toolkit for these types of visualizations. The purpose of this evaluation is to provide a preliminary analysis of the issues that will be present in using VisAD as a basis for educational visualizations.

This evaluation focuses on the effort required to understand the VisAD toolkit and implement a visualization solution on top of that toolkit. In addition, this evaluation attempts to determine the ballpark time horizon needed to construct a visualization system on top of VisAD. This evaluation does not provide a conclusive, in-depth evaluation of VisAD, as this is intended to be a preliminary evaluation to determine if further exploration is warranted. In addition, the majority of the technical details of VisAD are beyond the scope of this document.

Background

History of VisAD

VisAD was originally developed from code taken from the open-source Vis5d toolkit, which was itself based on another open-source visualization tool called McIDAS (Hibbard, 2000). Development of VisAD was pioneered by staff of the Space Science and Engineering Center of the University of Wisconsin-Madison. Additional contributions were made by the Unidata Program Center, National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, Australian Bureau of Meteorology, and the National Center for Atmospheric Research (Hibbard, 2003).

VisAD has been released under the GNU Lesser General Public License (LGPL), which means that applications that use it need not be licensed under the GNU General Public Licence (GPL), although the LGPL does have some restrictions of its own.

Technological Basis

The uniqueness of VisAD arises from its basis in the Sun Microsystems Java programming language, which is designed to provide easy cross-platform portability. In addition, VisAD is not an application builder or graphical design tool; rather, it is a suite of Java classes that can be assembled and manipulated to perform various different visualizations (Hibbard, 2003)

VisAD is comprised of code written entirely in Java. The standard packages that ship with Java are designed for 2-dimensional graphics work. To support 3-dimensional visualizations, VisAD relies on the Java3D package from Sun (Hibbard, 2003). At present, the Java3D package is available for the Sun Solaris, Microsoft Windows, Linux, IBM AIX, HP-UX, and SGI IRIX operating systems (Sun, 2003a-b). Notably absent from that list is Apple Mac OS X, although it is widely rumored on the Internet that the Linux version of Java3D runs on OS X. That does appear to be the case, as long as one of the XFree86 ports is installed on top of OS X (Blackdown, 2003).

VisAD Consumers

Several major visualization projects are built on top of VisAD, many by various government and educational sector entities. Included among these projects are the University Centers for Atmospheric Research Integrated Data Viewer, which provides visualizations for a number of meteorological data sets. Also, the University of Wisconsin has released a beta version of VisBio, which is designed to visualize 4-dimensional multispectral data from biological sources. There are numerous other smaller VisAD-based projects available from the official VisAD website (Hibbard, 2003).

Design and Implementation Procedure

Evaluation Objectives

The principal objective in this evaluation experiment was to determine the level of complexity associated with using VisAD as a platform upon which to build a custom visualization system. In order to focus as much attention as possible on VisAD, a simple experimental application was needed.

Design of the Evaluation

A relatively simple input data file format was found in the National Weather Service's "F-6" climate data form. These files were simple to understand and relatively easy to parse. Individual data files from the National Weather Service Forecast Office in Greenville-Spartanburg (NWSFO GSP, 2003) were downloaded and used as input to the experimental application.

In order to accomodate the variable number of days in a month (28-31), as well as the Missing data values from the F-6 file, a linked grid structure was designed. This structure provided a mechanism to support a variable number of data records with a fixed number of fields. To store the actual data into this structure, file parsing code was designed, using standard Java file input components.

From the linked grid structure, the data were available to the VisAD components for rendering. The remainder of the design was completed as the VisAD portion of the project was implemented, closely following the VisAD tutorial examples.

Java Implementation

Implementation of the experimental application began with the implementation and testing of the linked grid structure. Once the linked grid was completed, the file reading component was prepared, and the file data were stored into the grid structure. Implementation then shifted to a combination of design and implementation using the actual VisAD components.

Following the VisAD tutorial online, the VisAD data structure components were assembled from the rather large VisAD component library. Trial and error was used to map the raw data to the proper structural locations, along with appropriate units. Following the data structure layout, several hours of trial and error were needed to link the data to the rendering components.

Following the completion of the data linking, rendering and displaying the resulting visualization was automatic. VisAD

handled the scaling and display of the final graphic with very little programmer intervention.

Results of the Evaluation

The implementation of the experimental application was not completely straightforward. A major issue was encountered during the first attempt to parse the data file, using Java components supplied by Sun Microsystems. After significant troubleshooting, it was determined that the JavaDoc documentation for the component in question, coupled with that component's implementation, presented an ideal model of divergence. Thus, a new data parsing algorithm had to be written from scratch, which resulted in a setback of just over one day.

Apart from Java-related issues, the only other major difficulty in implementing the experimental application was the VisAD object model. Several trial-and-error procedures were required to obtain a functioning visualization, each of which required several hours to achieve relatively small amounts of progress.

Once the application was completed, the result was a space curve plotted from the high and low temperatures for each day of the month. VisAD handled the majority of the rendering of this space curve automatically, although some minor tweaks were needed to adjust the axes and labels. VisAD provided visualization canvas resizing, interactive panning and zooming, and interactive visual position identification; all of these features were available without any additional programming.

Overall, this application required 2 half-time (20 hours) developer weeks, with 1 developer working on the project, to implement. Prior to implementation, another 2 half-time developer weeks were required to study the object model and VisAD tutorials, in order to gain the most fundamental of understanding of VisAD.

Conclusions

Overall Impressions

VisAD is a large and complex tookit, a product of what appears to be an iterative and incremental development process. As such, VisAD provides considerable power at the expense of significant complexity. This visualization toolkit has a number of potential benefits; however it also has a number of issues that present significant concerns.

Design and support issues include the nature of the VisAD toolkit itself, as well as its reliance on the Java3D toolkit, which is not fully supported on all Java-enabled platforms. However, even with these issues, there are some applications that will benefit from the cross-platform capabilities of VisAD on the platforms on which it is supported. Thus, each project will have to be evaluated independently, to determine if VisAD is the appropriate toolkit to satisfy project requirements.

Technological Considerations

The key benefit to VisAD is its Java foundation, which makes VisAD applications cross-platform, as long as development is done in pure Java. This Java requirement presents two key concerns. First, as was demonstrated with this experimental application, there are portions of the Java programming language that behave differently than the documentation would suggest. On top of the documentation issues, truly portable VisAD applications may require that working code in other languages be either re-written in, or mechanically converted to, Java.

Because of the complexities already inherent to VisAD, the additional usage of Java native methods on top of the Java visualization application, would incrase the complexity of the overall visualization significantly. In addition, assuming that the Java native method calls are truly portable across platforms, the usual portability issues of C(++) or Fortran would nullify most of the benefits of using VisAD in the first place. Some languages, especially later Fortran standards, may not even be supported on all platforms. These issues would indicate that VisAD would not be the tool of choice for designing single-platform visualizations from an existing non-Java code base; rather, VisAD would be more suited to purpose-built visualizations to be distributed to consumers with a wide array of hardware platforms and operating systems, such as K-12 educational institutions.

A second issues arises from the large hierarchy of objects that must be assembled in order to create a VisAD application. Trial and error often is needed to determine the exact behavior of VisAD components, because component-level documentation is lacking. Although the tutorials are somewhat helpful, they are often nebulous and require several readings to comprehend. Thus, it is likely that a VisAD-based application will require a significant developer time investment, especially if the developer is using VisAD for the first time.

Finally, VisAD is a component toolkit, not an application builder. Visualizations are built in the Java programming language by assembling object components provided by the VisAD package. Working at the code level is not intuitive for the developer, and drawing connections between the Java classes and rendered result is extremely difficult at times. This factor is a large part of the reason that VisAD applications will be slow to design and implement.

Support Issues

Apart from the actual design and implementation issues inherent with using VisAD, the package also has several attributes that will complicate end-user technical support. Not least of these issues is reliance on the Java3D toolkit, which has limited platform support. For those platforms for which Java3D is available, it must be installed on top of (and separately from) the regular Java Virtual Machine package. This increases the number of prerequisite installations that the user must perform, and each of these installations is platform-specific, so it will be difficult to provide good generic instructions.

Once the Java base components are in place, the user must also install VisAD (or it must be included with a visualization application) and the visualization application itself. This installation could present a serious support issue because Java requires that the CLASSPATH variable be set properly, or else the Java runtime will complain of missing classes. The only known workaround to this support problem is to provide platform-tailored installers, which would result in a resurfacing of the platform portability problem that Java is supposed to solve.

References

Blackdown, 2003: *Java3D (TM) 1.3 README file for Linux/OpenGL*.
http://www.blackdown.org/java-linux/java2-status/README-3D13

Hibbard, W., 2000: *The VisAD Java Component Library Developers Guide*.
http://www.ssec.wisc.edu/~billh/guide.html

Hibbard, W., 2003: *VisAD Home Page*.
http://www.ssec.wisc.edu/~billh/visad.html

NWSFO GSP, 2003: Past Weather. *National Weather Service Forecast Office, Greenville-Spartanburg, SC*. http://www.erh.noaa.gov/gsp/climate/climate.htm

Sun Microsystems, 2003a: *Java 3D 1.3.1 - Download*.
http://java.sun.com/products/java-media/3D/download.html

Sun Microsystems, 2003b: *Java 3D API on Other Platforms*.
http://java.sun.com/products/java-media/3D/ports.html

Selected VisAD Resources

- VisAD Home Page
- VisAD Tutorial

Inventory of Potential Alternative Toolkits

This partial list of alternative toolkits is copied verbatim from the author's software inventory of 14 December 2002.

```
GMV - General Mesh Viewer
3-D scientific visualization tool for viewing 2-D or 3-D mesh visualizations.
Language: ? Appears to be closed-source
License: Binary freeware
http://laws.lanl.gov/XCM/gmv/GMVHome.html


MeshTV
3-D mesh visualization viewer with user interface components.
Language: not exactly known, but probably C or C++
License: free and source available, but in the COPYRIGHT file included, there
         is a notation that Laurence Livermore National Labs must be notified
         before software can be commercialized.
http://www.llnl.gov/bdiv/meshtv/


GEMPAK
Developed between the NWS and the universities participating in UNIDATA,
GEMPAK provides much of the visualization eventually sought by my project.
The difference is that GEMPAK is closed-source and only available to
universities, although it is free. Also, it appears that some components of
the National Weather Service's AWIPS system have been included in this
system. It may be useful to obtain a copy of this software just to see what
the AWIPS components do.
Language: ? (appears to be closed-source)
License: free but apparently cannot be redistributed
http://www.unidata.ucar.edu/packages/gempak/index.html


Grid Analysis and Display System (GrADS)
This tool has been popularized in the Bulletin of the American Meteorological
Society as a generic visualization engine.
Language: ? (its scripting language is "FORTRAN-like")
License: GrADS Open Source license. Source available but need permission to
         redistribute.
http://grads.iges.org/grads/grads.html


Vis5d+
An open-source visualization tool. Used in some radar research recently
published in the Bulletin of the American Meteorological Society.
Language: Looks to be a mix of C(++) and FORTRAN-77
```

License: GPL v.2
http://vis5d.sourceforge.net/
*** potential problem: Vis5d relies on a licensed underlying codebase with more restrictive licenses. Work is underway to convert Vis5d such that it uses GTK instead; however, we could potentially find ourselves in the position of having to do some of that conversion work (I doubt the Vis5d project people would mind, however).

OpenDX
An open-source visualization tool based on a visualization system originally developed by IBM. Probably needs to be explored further, as they show some pretty pictures on the site. Original IBM system was called Visualization Data Explorer (VDE).
Language: ?
License: IBM Public License
http://www.opendx.org/

Open CASCADE
Another open-source 3D visualization system. This system consists of a set of libraries designed to operate more or less alone, or with other (especially closed source) code.
Language: C++
License: Open CASCADE Public License
http://www.opencascade.com/products/occ/