Interoperability of Visualization Software and Data Models is Not an Achievable Goal

E. Wes Bethel
ewbethel@lbl.gov
Lawrence Berkeley National Laboratory

1. PANEL DESCRIPTION

The scientific visualization community faces a crisis: there exist many individual tools that can be used to perform visualization, but there is little, if any, hope of being able to use tools from different sources as part of a single application. As a result, our community is fractured, and can be characterized as "islands of capability." The purpose of this panel is to probe the issues that prevent such interoperability, and engage in frank discussion about how our community can rectify these maladies. The issues to be discussed include but are not limited to: (1) lack of "standards" for data storage and modelling of N-dimensional scientific data, similar to those used for raster image files; (2) lack of "standard" interfaces for common visualization tools; (3) the visualization needs of the computational science research community, who are the primary consumers of technology from the visualization community; (4) lack of organization within our community to push for definition and adoption of such "standards;" (5) lack of organization within our community to serve as a "broker" and "promoter" for tools that might conform to even the weakest of standards. The panelist lineup represents a diverse cross-section of expertise and opinions about the panel topic. The panelists themselves are in disagreement about the severity of the problem, and potential solutions. The topic of this panel is highly germane to future growth of visualization as a science, and promises to be highly engaging for panelists and audience members alike.

2. PANELISTS

2.1 Greg Abram, IBM T.J. Watson Research Center

To my mind, much of the failure to develop and adopt broad-based standardization within and between the visualization and computational science communities reduces to the short-sighted belief that the performance of a solution to any particular problem - or, in fact, the performance of any particular component of a particular problem – outweighs the larger goals of maximizing cost-effectiveness and minimizing total time-to-solution across a wide range of problems. Several things foster this attitude. There is the fun and ego-gratification of delivering a heroically built piece of code. There is the performance gain to be had by taking advantage of unique characteristics of a particular problem to provide a tailored, super-tuned piece of code that performs ideally on that problem. There is the need to justify the purchase of hardware with tangible results shown by anecdotes of larger problems being solved in less time. What there isn't is a the recognition of the real problem - maximizing cost-effectiveness and minimizing time-to-solution over all of the tasks that confront an institution, accompanied by anecdotes of problems that arise and are solved satisfactorily, quickly and inexpensively. The problem is compounded once institutional lines are crossed. Proprietary goals make concessions to standardization unpalatable. A lack of concession leads to attempts at standardization that are all things to all people, and are useful to none. Users and management are unwilling to accept costs and inconveniences that they might attribute to being of more benefit to a competitor than themselves.

We attempted to attack this problem when we first developed OpenDX. We saw a visualization community that relied on either domain-specific closed applications, or components tied very closely to specific representations, which were often built for very specific purposes and then tossed into a toolkit. We felt that we could develop a data model that would provide effective representation across a very wide range of applications and then a set of visualization components that operated on any data represented in the model. Since objects in the model provided the ability to describe their representation, we could build high-level components that examine their input and call low-level representation-specific algorithms if they existed, and or use more general-purpose algorithms otherwise. We could "hammer on the highest nail," providing customized algorithms where they proved most valuable, and not waste time developing tuned algorithms for less important cases. We could provide users the ability to create the components that they needed and leverage our vendor-supplied code wherever it would suffice. Finally, we noted the ease-of-use that a self-describing data model would provide, since users building high-level visualization applications could concern themselves with function without regard for representation, up to the point that they needed to address performance issues.

I always hoped that this would serve a larger purpose as well - to subsume both the visualization process and the computation that creates the data for visualization into a single super-application. I believed that the convenience the infrastructure of OpenDX - notably the data model, but also visual programming interface and GUI-builder capabilities, would entice people to incorporate computational science

components into OpenDX, thereby building computational steering and tracking into their application. Alas, this has rarely been the case.

Greg Abram received his PhD at the University of North Carolina at Chapel Hill in 1986. He has been a Research Staff Member at the IBM T.J. Watson Research Center since 1990, working on many aspects of scientific visualization as well as participating in the architecture and implementation of OpenDX.

2.2 John Shalf, Lawrence Berkeley National Laboratory

Visualization systems will become an essential part of the emerging fabric of Grid services. While there have been many tantalizing demonstrations of the capabilities that this new "Grid" frontier will enable, there is a huge gap between the demonstrations of Grid visualization applications and solutions that could reasonably be deployed in a production environment.

The underlying Grid infrastructure offers a considerable number of challenges to application programmers before it can become a stable substrate for interactive visualization applications. This includes some work that is the traditional domain of visualization researchers like specialized latency tolerant or highly scalable parallel visualization algorithms. However, there are considerably more areas of development that require advances in security models, performance modelling and prediction, dynamic application redeployment, distributed data management, and network protocols that are not the typical domain of visualization researchers. These problems are very complex and will require considerable effort just to build the Grid infrastructure up to a level that can stably support production quality applications.

Unless we make a concerted effort to address the deficiencies in the current Grid and bring it up to a level of abstraction that is sensible for visualization application needs, there will be little progress in this area. However, given the current balkanization of visualization systems and capabilities, every application developer in this community will need to solve the same problems over and over again, thereby continuing this depressing of lack of forward progress in this community. We have failed to rise to the challenge of large data visualization on commodity MPPs in a manner that is widely deployed in production environments. Will we continue with this track record through the Grid revolution as well?

The problems inherent in the Grid are far too complex and varied for us to provide an even minimally useful solution unless we are able to find a way to combine our efforts. The only path to sharing this difficult job of Grid infrastructure is by coming back to the table to address the very difficult problems of interoperability between different visualization frameworks and tool environments that we have abandoned so many times in the past.

John Shalf is a staff scientist at Lawrence Berkeley National Laboratory. He is involved in projects that cover visualization of AMR data, distributed/remote visualization, Grid portal technology, high performance networking, and computer architecture. He has also been a visiting researcher at the at the Albert Einstein Institute/Max Planck Institute in Potsdam Germany and is a member of the Technical Advisory Board for the EU GridLab project that seeks to create application-oriented APIs and frameworks for Grid computing.

2.3 Randy Frank, Lawrence Livermore National Laboratory

Our ability to generate large datasets is rapidly outpacing our ability to manipulate and visualize them. The move to distributed visualization systems to handle these datasets has been slow and strewn with potholes. Moreover, recent leaps in workstation class PC performance has allowed commercial developers to largely avoid dealing with this problem, leaving many of us to roll our own tools from the myriad of available components. Indeed, research into advanced mechanisms for handling these datasets has yielded a number of excellent solutions to these problems, yet seldom do these techniques make their way into production tools for end users. Funding mechanisms and other forces have pushed the community toward large, monolithic implementations that are rarely nimble enough to leverage the latest research results. The complexity of such codes prohibits their adaptation to and timely delivery of vertical visualization solutions that our end users request. A new approach to tool development is needed. There are many great system components available, yet you rarely see them used together to solve a problem. Why?

An approach is needed that encourages independent groups to work together on system components with focused, intelligently designed interfaces that do not require large infrastructure changes to use. Most distributed visualization tools consist of a few basic components that in many cases know nothing about the actual data they are handling. Loosely coupled lightweight components with a relatively narrow focus may be the key to our ability to rapidly prototype and deliver visualization solutions, which leverage the results of novel approaches, our end users desire.

Randall Frank currently serves as the LLNL ASCI VIEWS Visualization Project Lead. He is involved in a number scalable rendering and visualization research projects, targeting interactive visualization of terascale data and distributed clusters. These projects include Chromium, DMX and the TeraScale Browser. Prior to joining LLNL, he worked at Research Systems as a Systems Architect on

their IDL product. He has received BS and MS degrees in Biomedical Engineering from the University of Iowa.

2.4 Jim Ahrens, Los Alamos National Laboratory

I do not believe visualization tool interoperability is a crisis because solutions are available now or will be in the near future. Tools can work together by 1) sharing files and 2) sharing algorithms/user interfaces. Sharing files is fairly simple, but typically hindered by a lack of a common data format. There are current projects working to solve the common data format problem including HDF and the Los Alamos, Livermore and Sandia Scientific Data Management project. In the worse case, a visualization developer can translate from one tool's data format to the other. Sharing algorithms and user interfaces can be achieved by taking code from one tool and porting it for use in another tool. Algorithms can also be shared by using a common architecture. DOE's Common Component Architecture (CCA) project is currently solving this problem. A key question to address is: What is the definition of efficient/effective tool interoperability? A related question is: Who is the interoperability efficient/effective for? Visualization developers or users? I believe the next step for the visualization developer community is to focus on real-world user tool interoperability problems. I suspect many interoperability problems can be solved quickly and easily using the techniques described above.

James Ahrens received a Ph.D. in computer science from the University of Washington in 1996. After graduation he became a staff member at Los Alamos, where he is currently employed. His research interests include scientific visualization and parallel/distributed systems. He has written book chapters, journal and conference papers on these topics. He is leading an open-source software effort to extend the Visualization Toolkit (VTK) and create an end-user tool (ParaView) to visualize extremely large datasets.

2.5 Steve Parker, University of Utah

Visualization systems were one of the earliest adopters of component-based architectures. Numerous systems have been developed over the years that have successfully employed these architectures to form powerful visualization systems, including AVS, IBM Data Explorer, IRIS Explorer, Khoros, Vtk, and SCIRun. However, visualization research is seldom able to take advantage of these environments due to various constraints, both technical and otherwise.

I implore the visualization community to work together to create a standard component-based architecture that will fulfill the needs of visualization researchers and practitioners alike. This system should have the following attributes:

- Based on open standards, including the underlying component technology and the visualization standards themselves.
- Based on a flexible execution model to facilitate a broad range of visualization algorithms.
- Addresses parallelism, both in terms of parallel data and parallel visualization tools.
- Facilitates both flexibility and performance on a broad range of scientific data.
- Facilitates incremental adoption, such that communities can utilize pieces of the standard as they progress towards full adoption.
- Is sufficiently efficient and flexible to work with large-scale visualization tasks.

Many in the community have pieces of this larger puzzle, but putting them all together in an efficient manner is a non-trivial task. Nevertheless, it is an important undertaking that will help to bring to a broader community the myriad of techniques developed by visualization researchers.

Steven Parker is a Research Assistant Professor in the Department of Computer Science at the University of Utah. His research focuses on problem solving environments, which tie together scientific computing, scientific visualization, and computer graphics. He is the principal architect of the SCIRun Software System, which formed the core of his Ph.D. dissertation, and is currently the chief architect of Uintah, a software system designed to simulate accidental fires and explosions using thousands of processors. He was a recipient of the Computational Science Graduate Fellowship from the Department of Energy. He received a B.S. in Electrical Engineering from the University of Oklahoma in 1992, and a Ph.D. from the University Utah in 1999.

2.6 Nagiza Samatova, Oak Ridge National Laboratory

The concept of a "plug-in" is not new. Plug-ins are code modules that literally plug into a computing framework to add capabilities that previously did not exist. Familiar examples include web browser plug-ins for playing live audio files, displaying PDF files, and much more. There is no need for "porting user interfaces" of these numerous applications for use in a Web browser, especially when a typical interface code is tens/hundreds of thousands lines of code. Data format translation – from one format to another – is not practical for terabyte data sets generated by scientific applications – simply use an appropriate plug-in.

We wish life were that simple so that we could take a visualization package(s) and plug it into our data management and data analysis infrastructure, called ASPECT, designed with some special end-to-end and QoS performance requirements. There exist many successful

visualization applications (e.g., OpenDX, ParaView, Terascale Browser) with a variety of complimentary capabilities to choose from. While it is possible to use any one in isolation, using them in a larger context is the ultimate goal. However, incorporating even a single one of them within another application or framework is very time consuming and tedious. Struggling with monolithic frameworks and tightly integrated user interface, system-specific event loops to handle the events for active user interaction, lack of support of various data formats (e.g., HDF, netCDF), or inability to mix and match packages by choosing an algorithm from one that the other doesn't implement are just a few of our frustrations that still make us believe that interoperability of visualization software is still fantasy.

Packages should be providers of visualization capabilities, rather than posing barriers to use by scientists or extension by developers. The idea of integration with other tools, frameworks and applications should be present from the start of development, as should the question "what kind of use breaks this package?" But until the mentality of visualization and data management architectures change in a fundamental way, the visualization software "plug-in" concept will remain an elusive desire, rather than a reality.

Nagiza Samatova is a staff scientist at the Oak Ridge National Laboratory. Her research focuses on advanced algorithms for large-scale, distributed and streamline data mining. She is the Project Lead of the ASPECT system developed under the DOE SciDAC Scientific Data Management (SDM) center. She received a Ph.D. in mathematics from Russian Academy of Sciences, Moscow, in 1993 and M.S. in computer science from the University of Tennessee, Knoxville, in 1998.

2.7 Mark Miller, Lawrence Livermore National Laboratory

Interoperability at the level of data models is not only feasible, but it has been achieved on the small and medium scale. Furthermore, we're making slow and steady progress towards large-scale integration. The key is DATA ABSTRACTION!

In the early days of scientific computing, roughly 1950 - 1980, each big simulation code effort included sub-efforts to develop supporting tools for visualization, etc. Developers working in a particular stovepipe designed every piece of software they wrote, simulation code and visualization tools alike, to conform to a common representation for the data. All software in a particular stovepipe was really just one monolithic application held together by a common, binary or ASCII file format. In short, there was no integration.

Between 1980 and 2000 an important innovation emerged, the MENU based I/O library. In fact, two variants emerged each working at a slightly different level of abstraction. One offered a menu of computer science (CS) objects such as arrays, structs and linked lists. The other offered a menu of computational modeling (CM) objects such as structured and unstructured-zoo meshes and zone- and node-centered variables. Examples of the former are CDF and HDF (early 80's). Examples of the latter are EXODUS (1982), Silo (1988) and CDMLib (1998). On the one hand, there are numerous examples of menu-based I/O libraries being used successfully to integrate on the small and medium scale. On the other hand, both classes of menu-based libraries have weaknesses prohibiting integration on the large scale.

By 2000, a new breed of I/O library began to emerge. It is based on modeling the abstract mathematical/physical continuum from which scientific software is ultimately derived. Examples are the Sets and Fields (SAF) and Sheaf scientific data modeling systems. These technologies enable developers to model scientific data in terms of WHAT it represents in a mathematical or physical sense independent of HOW it is represented in an implementation sense. These technologies promise take us into the large scale of integration. Nonetheless, this leap in integration does come at a price. Visualization tool developers must embrace this abstract mathematical world and express their software components in its terms.

Mark C. Miller received his Ph.D. in Electrical Engineering from University of California, Davis where he developed multi-resolution techniques for interactive, terascale terrain visualization. For the past eight years, he has worked on scalable, parallel scientific database technology supporting simulation codes in the Accelerated Strategic Computing Initiative (ASCI).

2.8 Wes Bethel, Lawrence Berkeley National Laboratory

Bethel is a Staff Scientist at Lawrence Berkeley National Laboratory, where he is a Group Leader for the Visualization Group. Bethel's background includes design and implementation of several generations of visualization, virtual reality, and graphics rendering systems and tools. Recent examples include the Visapult application, which was used to win the SC Bandwidth Challenge for three years in a row and OpenRM Scene Graph, an Open Source scene graph API tailored for high performance applications. Bethel's role in the panel will be to provide an historical perspective during introductory remarks, to ignite spirited discussion, and to serve as moderator.

3. ACKNOWLEDGEMENT

This work was supported by the Office of Science, Computational Technology Research, U. S. Department of Energy under Contract No. DE-AC03-76SF00098.