LBL-30412

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Information and Computing Sciences Division

Presented at the Conference on Computing in High Energy Physics,
Tsukuba, Japan, March 11–15, 1991, and to be published in the Proceedings

Visualization Using a Data Flow-Based Paradigm

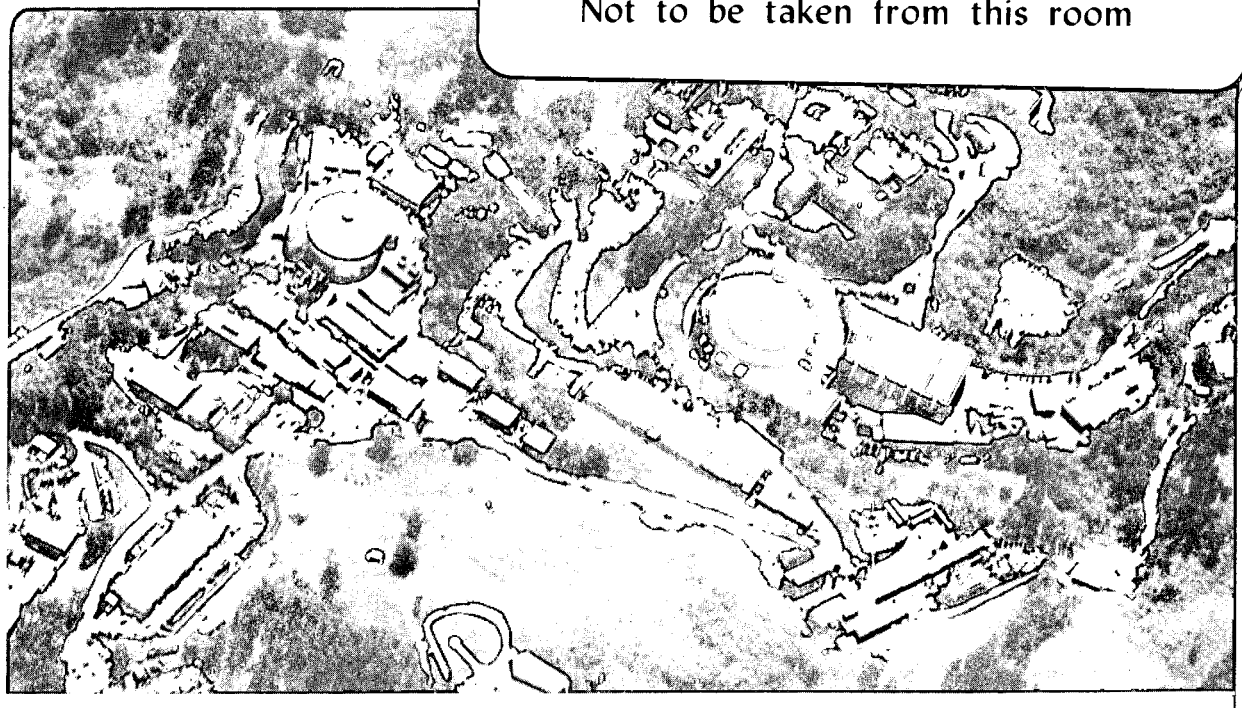
W. Bethel, N. Johnston, and H. Holmes

March 1991

U. C. Lawrence Berkeley Laboratory
Library, Berkeley

FOR REFERENCE

Not to be taken from this room



LBL-30412
Copy 1
Bldg. 50 Library.

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

LBL-30412

Visualization Using a Data Flow-Based Paradigm

Wes Bethel
Nancy Johnston
Harvard Holmes

Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720 USA

This work was supported by the Director, Office of Energy Research,
Office of High Energy and Nuclear Physics, of the
U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Visualization Using a Data Flow-Based Paradigm

Wes Bethel
Nancy Johnston
Harvard Holmes

Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720 USA

1. Introduction

A new computer data analysis and display technology has become available to the scientific community. This technology is the result of the conjunction of recent advances in computer graphics (in surface lighting and shading, object oriented three dimensional hierarchical geometry structures, and volume data rendering techniques); window systems (the wide acceptance of the X window system); user interfaces (the visual programming paradigm); software engineering techniques (the data flow based visualization concept); and affordable, high performance graphics workstations that support this software.

At Lawrence Berkeley Laboratory, we have made extensive use of Stardent Computer's Application Visualization System, a prime example of such a visualization tool, on data from a wide cross-section of scientific disciplines, including Particle Physics, Earth Sciences, Mathematics, Research Medicine and the Human Genome Center.

2. Elements of Data Flow-Based Paradigm

The Data Flow paradigm is based on the concept of a software toolbox. The toolbox consists of an aggregation of processing units which are separately compiled and linked, and thus are represented by individual, executable modules. Each computational unit is designed to perform a specific task. For instance, one module could perform a low-pass filtering operation on a scalar field, another could perform a transformation from a vector data item to an iconic representation (such as a three dimensional arrow, with size and direction proportional to the magnitude and direction of each grid element in the vector field) which is described using a collection of graphics primitives.

The only requirement strictly imposed on the computational units is that the data into and out of them is "strongly typed." In other words, there exists a well-defined set of data classes which are used as "templates." Given the strongly-typed data definitions, it is a straightforward to "link" together modules into a "data flow network." Unprocessed, or "raw" data enters at the top of the network, and exits the network after being processed or transformed into a possibly different data representation.

With regard to constructing a data flow network, the simplest approach is to use a collection of UNIX "pipes," where the output of one module becomes the input of another module. Such a network would be restricted to a univalent graph topology, and as such, is relatively restrictive in terms of configurability. Alternately, the modules' input/output facilities could be somewhat more sophisticated by using an interprocess communication channel. A TCP/IP connection would allow modules to share data over a network, possibly executing on drastically different machine architectures, as well as in geographically different locations. This form of input/output management would lend itself to the support of modules passing data to possibly more than a single downstream module. However, the management of connections and actual construction of the data flow network would be a tedious and error-prone task.

The notion of visual programming (VP), applied to the data flow model, vastly simplifies data flow network construction in a user-friendly manner, through the use of graphical user interfaces, and primarily mouse (rather than keyboard) events. Even though users are still required to specify the modules in the data flow network, and their interconnections, this task is performed using mouse generated "point and click" operations. This method is in stark contrast to the non-VP environment in which users must first know that names of all the modules they wish to execute, then type the name of the module along with all associated parameters. Further, the VP environment can allow for heterogeneous network support, permitting users to transparently execute "remote" modules. These modules may exist entirely on remote systems, or the module's internal computational units may be divided across machines. It is expected that as the bandwidth of networks increases, the use of heterogeneous data flow models will proliferate.

It has been our experience that users from a wide variety of computing backgrounds and graphics literacy levels adapt quickly to the VP environment. Scientists are able to quickly produce images from their data, and the VP environment is conducive to user experimentation with different types of imaging techniques. For instance, simply inserting a new computational unit into an existing network may produce dramatically more (or less) desirable results. Regardless of the outcome, the process of experimenting is reduced to a "point and click" operations, rather than a, possibly formidable,

"traditional" programming task.

3. Integrating Existing Applications into this Model

Using the data type definitions, it is a straightforward task to integrate existing codes into a given data flow model. In general, the issues involved are: first, defining and implementing the data mapping from the typed-definitions to that of the structure internal to the ported code, and second, implementing control over a code's parameters in the VP model. The data mapping problem varies in complexity from code to code. A generous number of flexible data-types in the data flow model will facilitate this process. Implementing visual dials and knobs to control a module's parameters is a straightforward task when the VP application is based upon one of the well-known windowing systems (X-windows, for example). Windowing systems typically have an associated toolkit which is used for not only quickly creating graphical user interfaces, but also for standardizing these interfaces across applications.

At LBL, many applications from widely differing environments have been ported into AVS. For example, numerical algorithms for resampling in two and three parametric dimensions are used extensively. The Scry Movie System, a distributed movie-making architecture developed at Lawrence Berkeley Laboratory, has also been integrated into the AVS data flow model, and is used to create the computer-generated movie which accompanies this publication. In both of these cited instances, the native data types were incompatible, to varying degrees, with the format used by AVS. In both of these cases, a minimal amount of effort was required to successfully map from one data type to another. Similarly, a minimal amount of effort was required to provide the graphical user interfaces to the parameters of each of these applications.

4. The Visualization Process at LBL: Case Study

We present an example of the visualization process from research at Lawrence Berkeley Laboratory. The problem is the visualization of a scalar field representing nuclear density during the process of atomic fission. The data is three dimensional, at a given time step. The grid size is 40 by 40 by 80 samples, where each sample is the result of an integration calculation. Each sample occupies approximately 1/2 fermi per side, or 1/8 cubic fermis (where one fermi equals 10e-13 cm). The fourth dimension of the data is the amount of energy applied to the atom, and is represented in the accompanying videotape as the time dimension, so that successive time steps correspond to succesive frame numbers in the animation. The case study illustrates the use of several different visualization techniques for creating images from this single, four-dimensional dataset. The intent is to point out how each of the different visualization techniques has a singular impact on the viewer, and portrays unique kinds of information. We used Stardent Computer's Application Visualization System (a VP, data flow based environment) to produce both the still images and the computer-generated movie which supplements this article.

The first step in visualizing this data is to simply import the data into the AVS system. The data was produced by a numerical code which created an ASCII file containing the points at each of the grid locations. A module was written

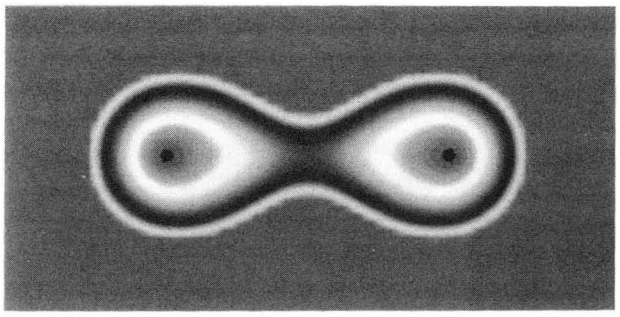


Figure 1

which reads the ASCII file and constructs a "field" structure (the configurable array in the AVS environment).

The first image (Figure 1) was created by first extracting a two-dimensional slice from a single time step, then mapping from the scalar value representing density to a color. This is a simple linear transformation resulting in a two-dimensional image. This image reveals a good deal about the structure of the atom at the energy barrier (the point

when the atom splits).

The second image (Figure 2) is designed to show the same kind of information, but using a three-dimensional representation. Several two-dimensional slices are extracted from the data and displayed in a bounding box. Additionally, an isodensity surface is displayed along with the slices. This image reveals the three dimensional structure of the atom at the energy barrier which is simply not available in a two-dimensional slice of the data.

In the third image (Figure 3), the data is reduced to an octant of the full, three dimensional dataset. Each point in the grid is represented using a sphere, the radius of which is proportional to the value of the scalar field. The use of three dimensional icons is usually applied to the visualization of vector fields. However, visualizing this scalar field using the three dimensional icon illustrates the spatial density of the grid points in one octant of the data, as well as the trends and

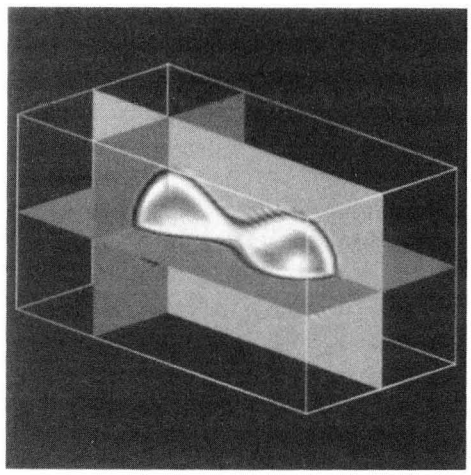


Figure 2

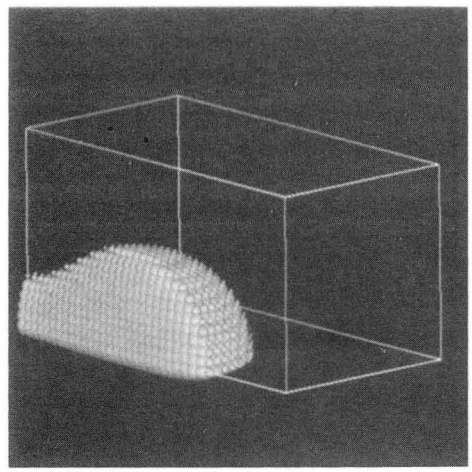


Figure 3

"hotspots" in the data.

Since the complete dataset is a sequence of three dimensional scalar fields, each of which is computed using a different applied energy level, the still images in this paper are produced from the dataset at a single time step. One key feature not present in these images is the dynamic nature of the time dependant data. We have generated animations using each of these visualization techniques, where each frame of the animation represents a different time step in the original data set. Each of these different animation sequences has the same merits as the corresponding still images. When presented in an animation, it is easier for the viewer to gain an intuitive understanding of the dynamic relationship, from time-step to time-step, within the data than is possible with still images.

5. Conclusion

At Lawrence Berkeley Laboratory, we have had overwhelming success using a data flow based visualization system, based on the emerging notion of visual programming. The primary reasons this system has been successful are: high-quality images from scientific data are quickly produced, due to the flexibility of the visualization system and the high-performance hardware available in today's market; the flexible data flow model is designed to permit the use of computational units in an arbitrary ordering;existing applications have been readily converted into modules to fit into the data flow model; the visual programming environment facilitates experimentation with alternative methods of imaging data.

6. Acknowledgement

This work was supported by the Director, Office of Energy Research, Office of High Energy and Nuclear Physics, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
INFORMATION RESOURCES DEPARTMENT
BERKELEY, CALIFORNIA 94720