Data Visualization during Program execution

SunSoft Developer Products

ABSTRACT

Scientific and technical software developers require the ability to "see" their data and do their analyses graphically instead of, or in addition to numerically. Exploration of large datasets, display of simulation results and interactive steering of computation all require data visualization in some way, shape or form.

Application developers in High Performance Computing need flexible methods for displaying large arrays of multivariable information with implicit or explicit geometries.

This paper describes how such functionality is offered with our current product.

1 INTRODUCTION

Visualization of data and information is a vital research and applications frontier that influences the scientific, engineering, medical, business, and entertainment fields. The ability to Visualize data during program execution gives the application developer the opportunity to comprehend large and complex datasets.

The requirements of scientists can be categorized as follows:

1. Data navigation.
   Since the datasets are, as mentioned above, typically quite large, manual inspection of the numeric output is very hard, if not impossible. Visualization combined with navigation allows the scientist to rapidly explore very large datasets.

2. Animation.
   The best way to present time-varying images produced by long running simulations is animation. The ability to store and playback these image sequences is high on the desired list.

3. Interaction.
   The scientist should have the ability to dynamically graph the set of variables. This would allow them to modify or abort computations.

4. Presentation of results.
   Scientists would also like to present their findings using a picture. So, a mechanism to print the graphs should be provided.

The goals for the Software used for visualization include:

1. Ease of Use.
   Scientists want to have a short learning curve. Hence the visualization techniques need to be easy to use and intuitive.

2. Scalability.
   The software should scale well to large datasets.

3. Extensibility.
   The user should not have to unlearn features as new releases become available. Also, the software needs to be extensible to support distributed data when users are working under a distributed paradigm.

The competition has a wide range of product offerings. Silicon Graphics has a product called Iris Explorer™, IBM has DataExplorer™, Thinking Machines has Prism™. There are also a range of third party products that help provide Data Analysis in addition to Data Visualization. These include PV-WAVE™ from Visual Numerics and IDL™ from Research Systems, Inc

The biggest difference between the competition and the scheme detailed in this paper that is used in the Sun WorkShop's Data Visualization is that there is no learning curve to be able to use the functionality. Almost literally, all the user would have to do is point-and-click. In most other systems, there is a command language and the barrier to entry is high.

This paper will first discuss the technology used for the underlying rendering library. After presenting the fundamental design principles used, it will also present the dbx command language.

2 RENDERING LIBRARY

Our requirement was to be able to do Data Visualization from the Debugger to allow the users to visualize arrays and visually determine local maximas and minimas.

We considered the following issues:

- The "Motif"ness of the library was a critical consideration.
- The working set size for applications linked with the library was another issue we considered.
- We did not want to have any run-time licensing.

Given these considerations, we decided to use the XRT technology from KL Group.

3 FUNDAMENTAL DESIGN

The Data Visualization capability is a available from the Sun Workshop. It can be invoked from the Debugger.

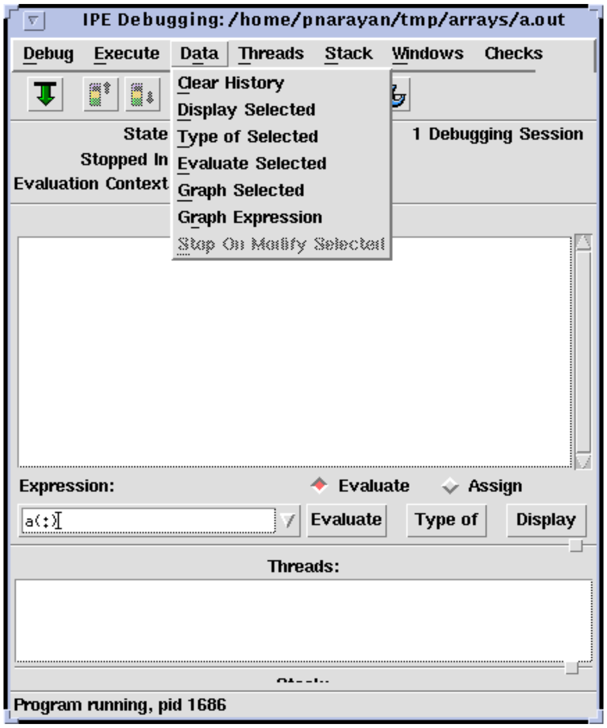

Figure 1 Debugger pane in Sun Workshop

The user selects *Graph Selected* to visualize the selected expression or *Graph Expression* to visualize the expression in the Expression box.

A dbx command language is available to affect the items on this list. The dbx command language is not a documented interface. It is used for communication to dbx from the Debugger.

There are, basically, three ways to update the data being visualized:

- Upon demand
- At every break point
- At time intervals

Each item being visualized is specified by an array description and refresh attributes. *Figure 2* shows the resulting graph after the program has started executing.

It displays single dimensioned arrays with two-dimensional graphs [with index in the X-axis and value in the Y-axis] and two-dimensional arrays with 3-D graphs [with indices in the X and Y axis and value in the Z axis]

The user gestures are controlled from the Data Visualization window. *Figure 3* shows the controls for a 2-dimensional graph.

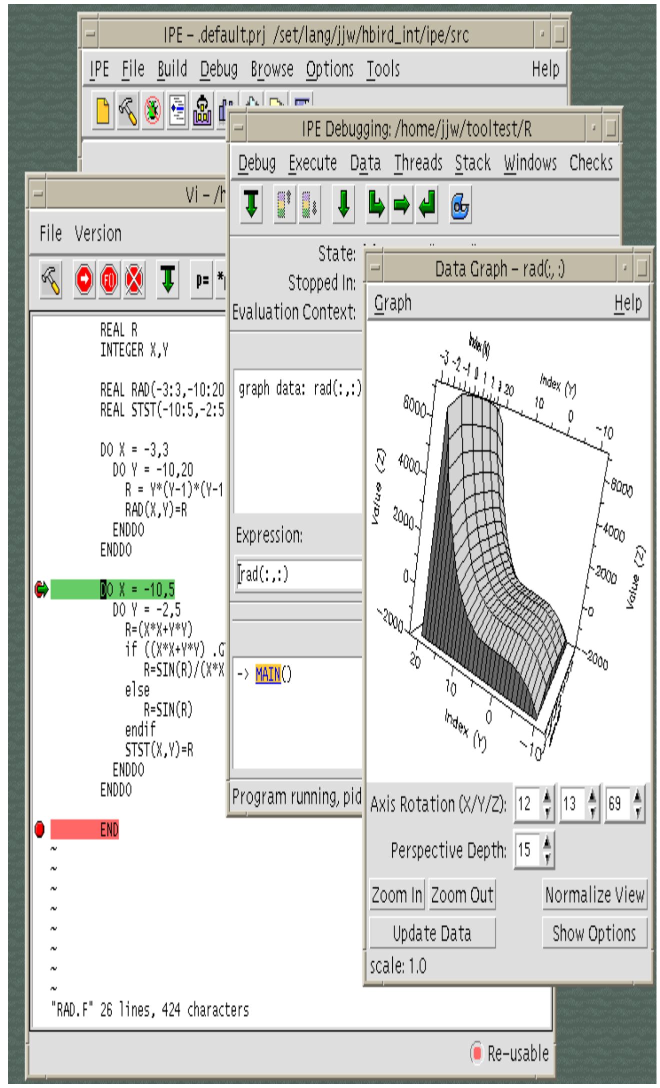

Figure 2 The Data Visualization window

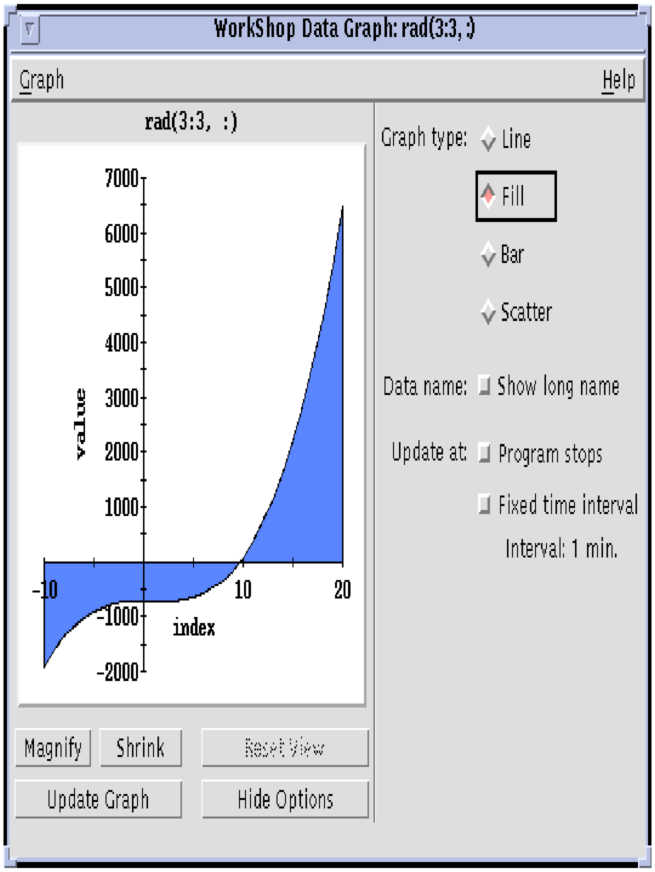

Figure 3 *2-dimensional Fill graph for a single dimension in the array rad*

Figure 4 shows the controls for a 3-dimensional graph. Here, additional controls are available to the user such as Zoom in, Zoom out, Axis Rotation, Perspective Depth, Texture, Shading and Graph Type.

The Array Expression that can be specified in the Expression box is an array specification in the language that is invoking it. One can specify enhanced f90-style extend and stride specification.

eg. C:

arr[] [:] [2:] [:6] [4:9] [::2]

Fortran:

arr (::2, 4:9, :6, 2:, :)

The extent of each dimension is implicit in the array declaration and ':' implies a range specification.

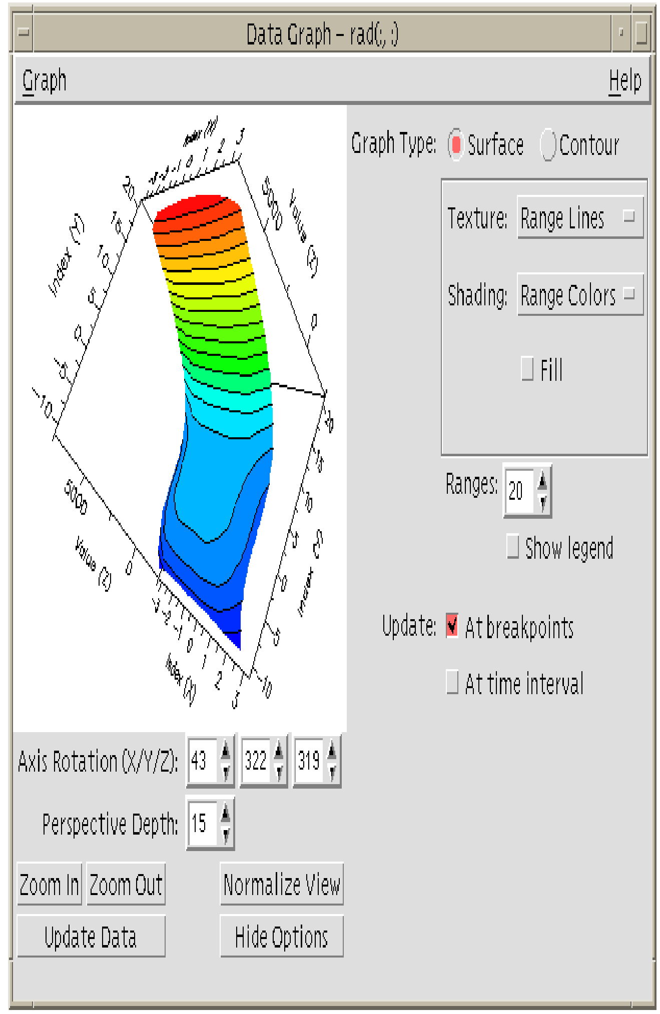

Figure 4 *3-dimensional surface graph for the array rad*

4 CONCLUSION

Our intention was to make the Data Visualization functionality available with this release of the Sun Workshop. There are enhancements that remain to be done. For instance, there is no support for Complex Arrays with Polar plots in this release. This and other enhancements will be made available in future releases. Please send in your comments using the Send Comments dialog box.