DataViewer: A Scene Graph Based Visualization Tool

Randy Paffenroth
Applied and Computational Mathematics
California Institute of Technology,
California, USA.
redrod@acm.org

Dana Vrajitoru
Computer and Information Sciences,
Indiana University at South Bend,
Indiana, USA.
danav@cs.iusb.edu

Thomas Stone
Veridian MRJ Technology Solutions,
Fairfax, Virginia, USA.
tstone@mrj.com

John Maddocks
LCVM^2, EPFL, Switzerland.
John.Maddocks@epfl.ch

Abstract

This article outlines the capabilities of a scientific visualization toolkit called DataViewer, and compares it to analogous software. DataViewer was originally designed for the construction of the visualization part of certain computational steering packages, and consequently it is particularly straightforward to closely couple DataViewer with numerical calculations. Rendering is performed through a high-level scene graph which facilitates the easy construction of complex visualizations. DataViewer differs from other such libraries by allowing complex geometrical objects, which efficiently encapsulate large amounts of data, to be used as nodes in the scene graph. Graphics hardware access is through the OpenGL API.

1 Introduction

DataViewer (*http://lcvmwww.epfl.ch/DV/*) is a scientific visualization library created to allow the easy development of efficient visualization programs for those who are not, and do not wish to become, graphics programming experts. DataViewer comprises a high-level set of routines for rendering geometric objects. It utilizes the OpenGL graphics API [2] for low level rendering. DataViewer has previously been used in several different visualization projects, e.g. most of the visualization modules of the parameter continuation software package VBM (Visualization of Bifurcation Manifolds, [4, 1]) have been developed using it.

The goals of DataViewer are to be computationally efficient, but with a high-level easy-to-use interface, portable (at least over UNIX platforms), and open source, freely available to the mathematical and scientific research communities. A less standard design goal for DataViewer was to facilitate a close coupling between visualization and numerical computations, so as to be able to visualize dynamic data sets easily. By dynamic data sets we here mean data sets that are continually being modified as, for example, in interactively steered codes. Such dynamic data sets require a tighter coupling to the actual computation of the numerical data than is necessary in the more standard mode of visualizing a static, pre-computed data set *a posteriori*.

From all of the available packages for scientific visualization, VTK (*http://www.kitware.com*) represents the software package most closely realizing the DataViewer objectives. In spite of its many advantages, it was not designed for close coupling to numerics, and is in many ways much more sophisticated than our needs. Other software packages like AVS (*http://www.avs.com/*), OpenInventor (*http://oss.sgi.com/projects/inventor/*), and GeomView (*http://www.geomview.org*), have been considered before starting the development of DataViewer, but none of them met our needs closely enough.

Hence in late 1996, we were left with the prospect of starting to create our own visualization software library. DataViewer 2.x was developed by R. Paffenroth, T. Stone, D. Vrajitoru, and A. Ahearn; it is our attempt to create a library which encompasses all of the needs detailed in the Introduction. The package is still being developed along with new applications in research and education.

2 DataViewer 2.x Implementation

DataViewer 2.x is written in C++, and uses many of the language's object oriented features, such as templates and virtual functions. Each graphical object class in DataViewer

2.x inherits from a single base class called DVobject and defines a virtual draw routine that creates a set of OpenGL commands for the graphical representation of the object.

Rendering in DataViewer is performed through a high-level scene graph based structure. The nodes in the graph can be either leaf nodes that contain data for visualization, and container nodes that group together other nodes. Geometry nodes in DataViewer 2.x include sets of lines, cylinders, triangle strips, and others. In DataViewer, properties such as color, translation, and rotation can be either defined by any node or inherited by entire subtrees.

DataViewer differs from other such libraries in that it allows both simple and complex geometrical objects to be used as single nodes in the scene graph. The complex nodes can be used to visualize large amounts of data more efficiently than is possible by using conglomerations of simple nodes, while the simple nodes can be used when flexibility is required. For example, Figure 1 is represented as two leaf nodes, a simple one containing the planar grid, and a more complex one containing a set of line segments rendered as cylinders with an associated ribbon .

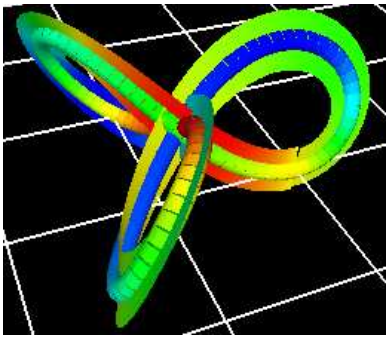

Figure 1. A complex object which is represented in DataViewer by two leaf nodes.

Python gives access to the the Tcl/Tk [3] widget libraries, which DataViewer utilizes for the design of GUIs. For users with only a little familiarity with Tk and Python, it is quite straightforward to create customized DataViewer GUIs for their particular problems. DataViewer 2.x also provides several useful features like stereo viewing and access to six degree-of-freedom controls, such as the Magellan 6D Mouse from Logitech Inc.

DataViewer was designed to be easily linked to numerical programs. This link can be achieved in several ways. First of all, numerical programs written in C++ or FORTRAN can be directly compiled in a DataViewer application can be updated without having to compile the library itself again. Second, more complex stand-alone programs like AUTO [5] can communicate with DataViewer by means of data files. Third, users that are not familiar with programming can use a native geometrical scene file format to visualize their data.

Finally, DataViewer is design to facilitate both flip-book and key-frame animation which can be both achieved through the compiled C++ code or from the interpreted Python GUI.

3 Applications

Several applications of DataViewer have already been implemented. We briefly mention three.

VBM (Visualization of Bifurcation Manifolds) (*http://lcvmwww.epfl.ch/VBM/*) [4, 1] developed by R. Paffenroth, J. Maddocks, R. Manning, D. Vrajitoru, and K. Hoffman, is a software package whose goal is to provide tools for computing, manipulating, and visualizing bifurcation manifolds obtained by parameter continuation. It allows easy access to large data sets using special projections of the bifurcation diagram and provides direct selection of any particular solution, followed by its visualization in a separate window using a data probe (see Figure 1).

The geometrical **Scene File Parser** application (*http://lcvmwww.epfl.ch/DV/Scene_file/*) written by D. Vrajitoru is an application of DataViewer allowing rapid development, modification and visualization of 3D scenes by means of a file parser that can be used both as an input and output mechanism.

Slinky (*http://lcvmwww.epfl.ch/Slinky/*) is a package that was build to interactively explore solutions of an initial value problem for a system of ordinary differential equations describing an elastic ribbon. The ODE system governs the shape of an elastic rod that is used as a model of DNA. The GUI allows the effect of changes in various coefficients and model parameters to be viewed interactively, with no external coding required.

References

[1] J. H. Maddocks, R. Manning, R. Paffenroth, K. Rogers, and J. Warner. Interactive computation, parameter continuation, and visualization. *International Journal of Bifurcation and Chaos*, 7(8):1699–1715, 1997.

[2] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide*. Addison-Wesley, 1993.

[3] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[4] R. Paffenroth. *Mathematical Visualization, Parameter Continuation, and Steered Computations*. PhD thesis, University of Maryland, 1998. `http://www.acm.caltech.edu/~redrod`.

[5] R. Paffenroth. The auto2000 command line user interface. In *Proceedings of the 9th International Python Conference*, pages 233–241, 2001.