

# Geometric Algorithm Visualization, Current Status and Future

D.T. Lee\*

Department of Electrical and Computer Engineering  
Northwestern University  
Evanston, Illinois 60208, USA.  
E-mail: dtlee@ece.nwu.edu

**Abstract.** We give a survey of the current status of geometric algorithm visualization and offer some suggestions regarding geometric software library and future directions for visualization software.

## 1 Introduction

Since its inception two decades ago computational geometry has become a very active research field within theoretical computer science. There are a good number of research publications collected in `pub/geometry/geombib.tar.Z`, available via anonymous **ftp** from `ftp.cs.usask.ca`. Several journals dedicated to computational geometry have been established. The reader is encouraged to visit the Web page on *Geometry in Action* by D. Eppstein at <http://www.ics.uci.edu/~eppstein/geom.html> and computational geometry page by J. Erickson at <http://www.cs.berkeley.edu/~jeffe/compgeom.html> for more information.

Only recently an informal assessment of the impact of the field on other science and engineering disciplines was conducted and the questions of its *relevance* to practice were raised among researchers within the community. See the task force report prepared by Chazelle *et al.*[3]. As a result of these discussions and the need of geometric software, activities concerning development and implementation of geometric algorithms ensued. The annual ACM Symposium on Computational Geometry started to have a video session and entertain communications of experimental results beginning in 1991. The first International Workshop on Computational Geometry Software sponsored by NSF and ONR was held at the Geometry Center, University of Minnesota in January 1995, and effort in collection of geometric software has been underway. See the Web page at <http://www.geom.umn.edu/software/cglist>.

In addition to the LEDA[10] project at the Max-Planck-Institut für Informatik, Saarbrücken, Germany, a concerted effort of building a kernel for geometric software has been initiated in Europe, known as the CGAL project[7].

---

\* Supported by the Office of Naval Research under the Grants No. N00014-93-1-0272 and No. N00014-95-1-1007 and by the National Science Foundation under the Grant CCR-9309743.

There are other projects related to the efforts of building geometric software, including GASP[15], GeoLab[4], GeoMAMOS[8], XYZ Geobench[13], etc. In this article we'll concentrate on geometric algorithm visualization tools, give a summary of their current status and give a *wish list* of what a geometric algorithm visualization tool or environment ought to provide.

## 2 Current Status

Using conventional approaches it is difficult to understand the structures of geometric objects represented in numerical format. Perception of objects is better obtained from visualization, as graphical form is most natural to human eyes, and conveys more information than numerical values or texts. By the same token to better understand the behavior of an algorithm, be it geometric or not, we would make use of visualization. Techniques and tools for algorithm visualization are thus developed in hopes that they not only help describe the algorithm behavior after they are fully implemented, but also help algorithm developers debug the programs when they are being implemented. We describe below briefly projects related to geometric algorithm visualization. There are software tools for *program visualization* or general algorithm animation including Zeus developed by Brown[2], Tango[14] by Stasko and Pavane by Roman *et al.*[12]. The reader is referred to [11] and to a special issue on Visualization (*Computer* Vol. 27, No. 7, July 1994) for more information.

**Alpha-shape**[5] developed at the National Center for Supercomputing Applications, University of Illinois, is a special purpose *shape* modeler and visualization package. It takes a set of points in 3-D space and computes the **alpha**-shape defined by the point set for different values of  $\alpha$ , a control parameter. The computed output for various values of  $\alpha$  is first stored in binary form in a temporary data file, which is later used as input to a graphical visualizer. The user selects a specific value of  $\alpha$  and the pre-computed **alpha**-shape gets displayed. The software is available via anonymous **ftp** from <ftp://ftp.ncsa.uiuc.edu/Visualization/Alpha-shape>.

**GASP**[15] developed at Princeton University is a tool for geometric algorithm animation. It has a library of primitive animation functions. The user stores the needed animation routines in a *style* file, and in the main application program the user will invoke these animation routines at appropriate places. All input and output operations are performed using files. The software written in C is available via anonymous **ftp** from <ftp.cs.princeton.edu:/pub/people/ayt/gasp.tar.Z>.

**GeoLab**[4], developed at the Universidade Estadual de Campinas is a programming environment for implementation, testing and animation of geometric algorithms. This tool uses shared libraries of algorithms and an incremental approach to aggregating new types of geometric objects, data structures and extensions accessed through dynamic linking. It runs on SparcStations under Sun/OS using

XView graphics library. It is written in C++ and has been used as a research and teaching tool at the Universidade Estadual de Campinas.

**GeomView**[9] developed at the Geometry Center is a general purpose data visualization software. It can be used as a stand-alone *viewer* for geometric objects or as a *display engine* for objects produced by other programs. Specifically it takes as input a geometric data file in a special format and displays the contents in graphic form. Various operations, such as translation, rotations, zoom-in and zoom-out are provided to let the user visualize and manipulate the graphical data. It runs on Silicon Graphics(SGI) IRIS workstations and NeXT workstations.

**GeoSheet**[8] developed at Northwestern University, is an interactive visualization tool designed to simplify geometric algorithm visualization procedures in a distributed environment. It is display device independent, although in the current release the display is based on Xfig (Facility for Interactive Generation of Figures under X11) and is primarily for 2-dimensional objects. An interprocess communication (IPC) mechanism is used for message passing between processes that may be running on different machines. This mechanism allows the user to visualize geometric objects without storing the data into a file. GeoSheet is implemented in C++ running under UNIX and X windows environment. The 3-dimensional version, **3Dsheet**, is under development and runs on SGI machine. The current version contains some geometric algorithm implementations using LEDA. See <http://www.eecs.nwu.edu/~theory/geomamos.html> for more details.

**LEDA**[10] developed at Max Planck Institute für Informatik, provides a sizable collection of data types and algorithms in a form which allows them to be used by non-experts. It is implemented in an object-oriented style with a C++ class library, and emphasizes data structure, algorithm reuse and demonstration.

**Shastra**[1] developed at Purdue University produces several interoperable, collaborationware toolkits for the creation, manipulation and reconstruction of solid models from a variety of sources. These sources include but are not limited to CT/MRI scans, 3D digitized point data and Oceanographic cartography data. These tools provide for the rapid prototyping and analysis of solid objects prior to simulation or manufacture. The applications include solid modelers, algebraic geometry systems, spline surface toolkits and medical image reconstruction systems as well as others. Current work involves tele-collaborative visualization of scalar and vector data from problems in fluid flow, medical imaging and image analysis as well as the development of virtual environments that allow for the simulation and analysis of various physical “worlds”. For details see <http://www.cs.purdue.edu/research/shastra/shastra.html>.

**XYZ GeoBench**[13] developed at ETH, Zurich, Switzerland, is a unified geometric programming environment, providing tools for creating, editing, and

manipulating geometric objects, and demonstrating and animating geometric algorithms. It provides a user interface as well as a library and is implemented in an object-oriented style running on MacIntosh. See <http://wwwjn.inf.ethz.ch/group/projects.html> for more information. A similar work, called Workbench, was developed at Carleton University[6].

### 3 Future Work

Here we briefly give some possible directions for future work in the context of geometric algorithm visualization and software development. It is a *wish-list* of things to have.

**Software Library** Currently we need *more* geometric primitives at all levels. LEDA does provide basic data types for various fundamental data structures, and a collection of graph algorithms. These primitives include routines for computing intersection of convex polyhedra, convex hull computation, Delaunay triangulation in  $d$ -dimensions,  $d \geq 2$ , triangulation of a simple polygon, just to name a few of those fundamental geometric algorithms. It will be useful if a default visualization or animation routine for each of these operations is readily accessible as easily as a function invocation. Such a common software library is desperately needed.

**Rapid Prototyping** To facilitate development of software tools or primitive functions a programming environment that provides facilities for performing unit testing and module integration is desirable. Visualization can be exploited to help algorithm developers visually examine and verify the output of each module. Developing geometric software is not much different from an ordinary software development task. One should follow software design principles and incorporate visualization into the design cycle to grasp the essence of the behavior of the geometric algorithm being developed.

**Debugging** Debugging is always a time-consuming process in any software project. It is more so in debugging geometric software. Traditional source code debuggers are all text-based. A *visual debugger* that can print the content of a geometric object in graphic form will be of tremendous value. Current tools, such as GASP and GeoSheet, for example, can provide assistance in this regard. But much needs to be done in the area of *interactivity* to improve the usage of the tools. See below.

**Interactivity** The visualization tools available to date are mostly data-driven. User interaction with these tools is minimal in the sense that the user can only change the data file and re-execute the visualization routine to visualize the altered data. The tools that allow the user to directly manipulate input data on the display via point-and-click mechanism are lacking. For instance, tools that permit the user to point to an object (point or otherwise) and

modify the position of the object or the information associated with the object are not available as yet. It would be very useful to have this capability so that the user can directly manipulate the objects and observe the effect of these changes on the algorithm as the program is being executed without going through recompilation process.

**Distributed Environment** As the hardware becomes more powerful and accessible, it is recommended that availability of heterogeneous machines be taken into account when designing geometric code and visualization software. Remote execution/access capabilities across the internet will alleviate the burden of installation at the local site of software tools developed at remote sites. The concept of collaborative visualization discussed in [1] may prove useful here.

## References

1. Anupam, V. C. Bajaj, D. Schikore and M. Schikore, "Distributed and Collaborative Visualization," *Computer*, 27,7 July 1994, pp. 37-43.
2. Brown, M. H., "Zeus: A System for Algorithm Animation and Multi-view Editing," *Proc. IEEE Workshop on Visual Languages*, Oct. 1991, pp. 4-9.
3. Chazelle, B, *et al.*, "Application Challenges to Computational Geometry," Tech. Report TR-521-96, Princeton University, April 1996. Also accessible on the Web at <http://www.cs.princeton.edu/~chazelle/taskforce/CGreport.ps>.
4. de Rezende, P. J. and W. R. Jacometti, "GeoLab: An Environment for Development of Algorithms in Computational Geometry," *Proc. Int'l Computational Geometry Software Workshop*, Geometry Center, MN, Jan. 18-20, 1995.
5. Edelsbrunner, H. and E. P. Mücke, "Three-Dimensional Alpha Shapes," *ACM Trans. on Graphics*, 13,1 Jan. 1994, pp. 43-72.
6. Epstein, P., J. Kavanagh, A. Knight, J. May, T. Nguyen, and J.-R. Sack, "A Workbench for Computational Geometry," *Algorithmica*, April 1994, pp. 404-428.
7. Fabri, A. G. Giezeman, L. Kettner, S. Schiia and S. Schönherr, "The CGAL Kernel: A Basis for Geometric Computation," this proceedings.
8. Lee, D. T., S. M. Sheu and C. F. Shen, "GeoSheet, A Distributed Visualization Tool for Geometric Algorithms," Tech. Rep. Department of EECS, Northwestern University, Oct. 1994. *Int'l J. Computational Geometry & Applications*, to appear.
9. Phillips, M., S. Levy and T. Munzner, "Geomview: An Interactive Geometry Viewer," *Notices American Mathematic Society*, 40,8 Oct. 1993, pp. 985-988.
10. Näher, S., "LEDA - A Library of Efficient Data Types and Algorithms", Max-Planck-institut für informatik.
11. Roman, G.-C. and K. C. Cox, "A Taxonomy of Program Visualization Systems," *Computer*, 26,12 Dec. 1993, pp. 11-24.
12. Roman, G.-C., K. C. Cox, C. D. Wilcox, and J. Y. Plun, "Pavane: A System for Declarative Visualization of Concurrent Computations," *J. Visual Languages and Computing*, 32 June 1992, pp. 161-193.
13. Schorn, P., "An Object Oriented Workbench for Experimental Geometric Computation," *Proc. 2nd Canadian Conference in Computational Geometry*, Ottawa, August 6-10, 1990, pp. 172-175.

14. Stasko, J. T. "Tango: A Framework and System for Algorithm Animation," *Computer*, 23,9 Sept. 1990, pp. 27-39.
15. Tal, A. and D. Dobkin, "Visualization of Geometric Algorithms," *IEEE Transactions on Visualization and Computer Graphics*, 1, 2, June 1995, pp. 194-204.