Techniques for Visualizing 3D Unstructured Meshes

Carole S. Gitlin
Christopher R. Johnson
Email: cgitlin@cs.utah.edu and crj@cs.utah.edu

UUCS-94-018

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

January 24, 1995

Abstract

We present a computational module for interactively visualizing, large-scale, 3D unstructured meshes. Scientists and engineers routinely solve large-scale computational boundary value problems on unstructured grids. These grids typically range from several hundred thousand elements to millions of elements. With this ability to solve such large-scale problems comes the challenge of viewing the 3D finite element model geometry on a 2D screen. When the mesh is non-uniform, the researcher's ability to grasp the complex spatial relationships of the elements often fails.

The visualization module we have developed for viewing large-scale 3D unstructured finite element meshes includes two novel real-time interactive methods: a clipping "surface" utility and a growth algorithm. Used separately or together, these new utilities allow the user to better explore complicated unstructured polygonal domains.

1 Introduction

Computational solutions of large-scale 3D boundary value problems in science and engineering usually require that the solution domain be partitioned into a set of contiguous polygonal elements. Mesh visualization is important for assessing solution accuracy, checking geometrical integrity (identifying missing and/or distorted elements), and interactive editing. Furthermore, with the advent of adaptive refinement and derefinement methods which alter the structure of the mesh, [HS92, SL94, JM94, MEJ94, YJ94] it is especially helpful to visualize the the changes taking place as an area of the mesh is refined. The computational mesh or grid is usually one of two types: structured or unstructured. With structured 3D meshes, one may visualize the geometry and underlying structure straightforwardly, using traditional graphical techniques. However, unstructured meshes present the more difficult task of grasping spatial relationships of unstructed 3D objects in 2-space. One common technique to assist in viewing these meshes is to use clipping planes [MEJ94]. When this technique is applied to unstructured meshes, however, the clipping plane cuts through portions of elements, leaving line segments on the edges of the mesh.

Speray and Kennon addressed this problem by devising a method of volume probes for interactive data exploration [SK90]. This method cuts sections of grids and displays them on 2D surfaces. However, the volume relationships between elements is lost when this technique is applied to elements displayed as 2D surfaces.

To overcome these problems, we have developed a software package called *MeshView* for visualizing 3D structured and unstructed meshes. MeshView contains a suite of options to provide the user with complete interactive control for mesh visualization. The module is part of a data-flow based computational steering system, SciRun, used for the interactive control of science and engineering boundary value problems. [JP94].

2 Methods

Conventional clipping planes fail when used to view unstructured 3D meshes because they cut across portions of elements. This results in a visual set of incomplete elements and edges. To overcome this problem, we implemented clipping isosurfaces. Instead of cutting off part of an element, this utility draws elements only in their entirety, so that if the plane goes through a particular element, no part of the element is drawn.

As soon as a vertex is found that is outside of the clipping surface, the entire element is discarded, and the algorithm moves on to the next element. The resulting surface is irregular, as the elements are of varying sizes, but all the elements are complete. Moving the clipping isosurface bar back and forth, the user is able to see how entire elements are added.

Another visualization tool we employ uses a growth algorithm. The user selects a starting element, and then can manipulate a slider which "grows" and shrinks the mesh. Growth is determined by elements sharing a face with an element on an "inner" level. The user also has control over which levels are drawn, either drawing all previous levels, or only the current level. Viewing only the current level facilitates visualizing all elements at a certain "distance" away from the seed element. To ease viewing for all the levels, the outermost level is drawn in a brighter color than previous levels. Fig.1 shows four consecutive stages in the growth algorithm. Fig.1a shows the seed tetrahedron. Fig.1b shows the four tetrahedra that share a face with the seed tetrahedron. Fig.1c shows all tetrahedra that share a face with any tetrahedra in Fig.1b, and Fig.1d shows all tetrahedra sharing a face with those in Fig.1c. Fig.2 is the same sequence as in Fig.1, with each of the tetrahedra shaded.

To make the growth algorithm real-time, a preprocessing stage sets up a data structure with the connectivity information of the elements. The algorithm starts with a seed element, and then depending on how many vertices the user has specified to have connected, builds the structure up level by level, using a stack. Once the data structure is filled, it only needs to be modified if the user changes the number of vertices to be connected or the starting element. The user can then interactively grow and shrink the mesh. Preprocessing the level information requires more storage than calculating connectivities on the fly, but it allows for real-time interaction.

The clipping isosurface and growth algorithm options, with their various parameters, can be combined together, giving the user a powerful interactive control with a high degree of control over the mesh visualization. By allowing the user to explore the mesh in different ways, the program gives the user the information needed to understand the spatial relationships between the 3D elements.

3 Results

Data enters MeshView via SciRun, the computational steering program [JP94]. Setup within SciRun contains a map consisting of an input module, computation mod-

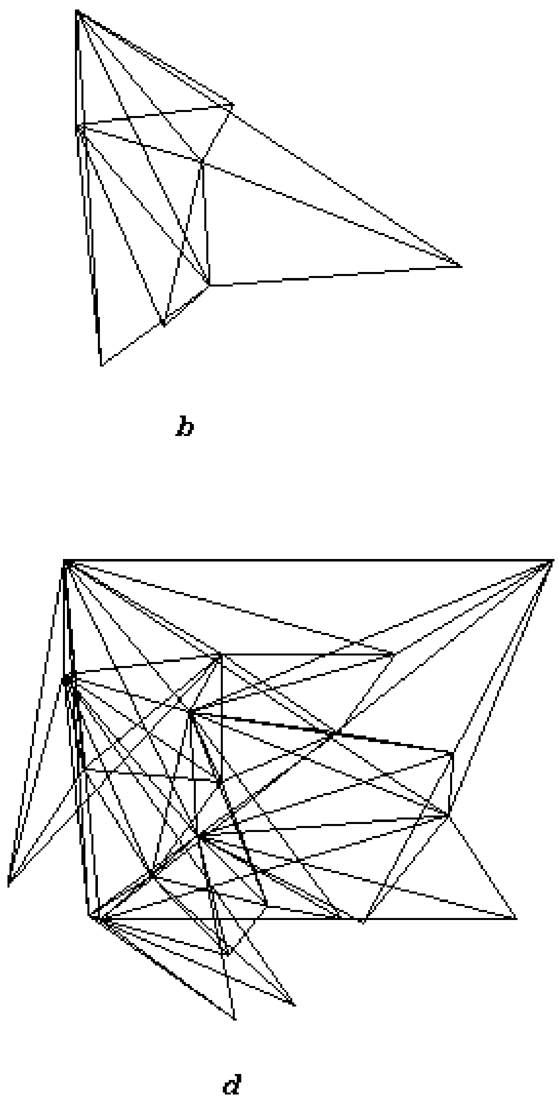

Figure 1: Sequence of four consequtive growths of the mesh. a) Seed tetrahedron. b) All tetrahedra sharing a face with the seed tetrahedron. c) All tetrahedra sharing a face with those in (b). d) All tetrahedra sharing a face with those in (c).

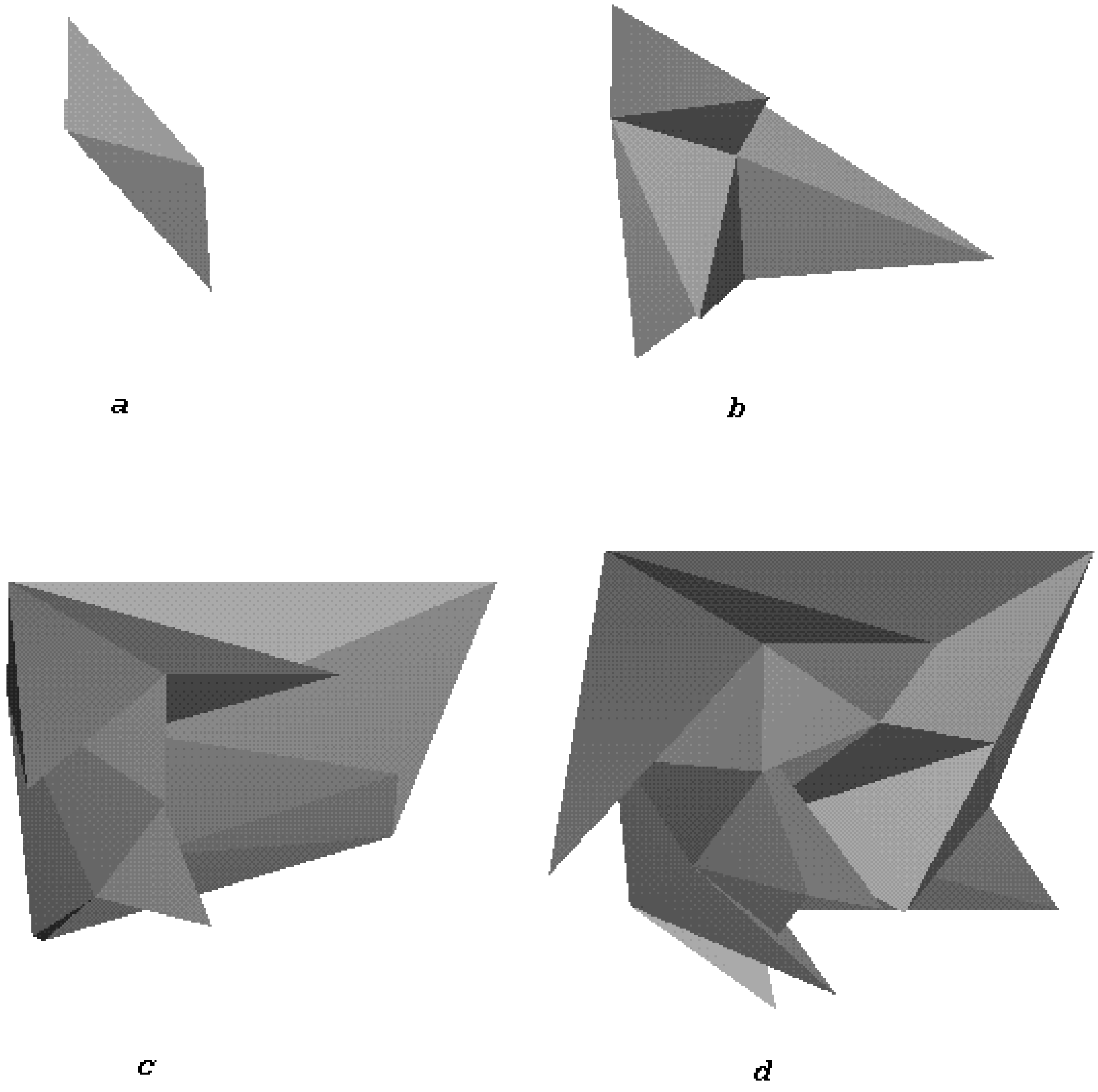

Figure 2: Sequence of four consequtive growths of the mesh, with Phong shading.

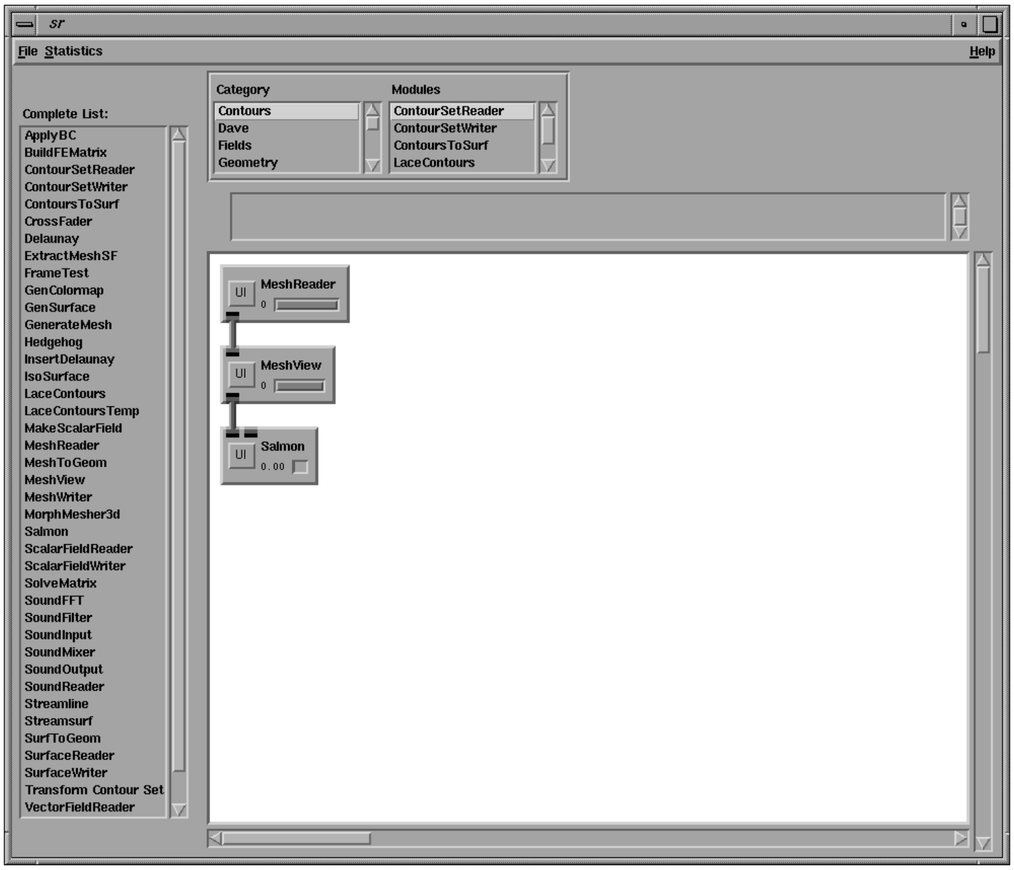

Figure 3: SciRun program and MeshView dataflow map

ule(s), and an output module, Fig. 3. In this case, the input module reads in the data points and connectivities of the mesh. The computation module contains the algorithms developed for MeshView. The output module is the geometry viewer, which displays the mesh. The program is written in C++ using Tk/Tcl [Ous94]. MeshView is currently implemented on SGI workstations, but ports to other platforms are in progress.

3.1 Statistics

The program was tested on a Silicon Graphics Indigo 2 workstation with a 100Mhz R4000 processor, 96 Mbytes main memory, and a 1 Mbyte cache. We tested the program with an unstructured mesh containing 51,125 tetrahedral elements. Initialization took 0.06 seconds. Changing the starting tetrahedron took 1.12 seconds when showing one level, 1.45 seconds when showing thirty levels, and 2.01 seconds when showing all sixty-two levels. For changing the number of levels being shown, it took 0.06 seconds to show one level, 0.16 to show sixteen levels, 0.51 to show thirty-four levels, 0.71 to show 54 levels, and 0.90 seconds to show all levels. The clipping isosurfaces average 0.05 seconds when only one level is shown, 0.50 seconds when thirty-four levels are shown, and 1.15 seconds when all levels are shown.

3.2 User Interface

When the program is started, the main SciRun window is brought up. It includes a large area for creating data-flow maps, as well as lists of all the modules. For running the MeshView program, a map of three modules is needed, an input module called MeshRead, the MeshView module, and a geometry module (called Salmon) for viewing the mesh. As shown in Fig. 3, the modules are connected indicating flow of data.

Each module in the data-flow window has a user interface (UI) button. Clicking on the Salmon UI button opens up a large window for viewing the geometry, Fig. 4. This window also includes buttons for setting the shading technique as well as allowing for rotations, scaling, and translations. When the user clicks on the UI button in the MeshView module, another window is opened up. This window contains the buttons and sliders for the MeshView.

The first two sliders in MeshView allow the user to control aspects of mesh growth. The first slider determines the number of levels drawn, thereby controlling the growth algorithm. Manipulating the slider grows and shrinks the mesh. The next slider determines which tetrahedron is to be used as the seed. The following 6 sliders control the clipping isosurfaces. These sliders permit the user to make a “cube” of clipping isosurfaces and thus able to clip from any orthogonal surface.

The last feature in the MeshView interface is a toggle button that allows the user to specify whether all the levels or only the outermost level will be shown in the growth algorithm.

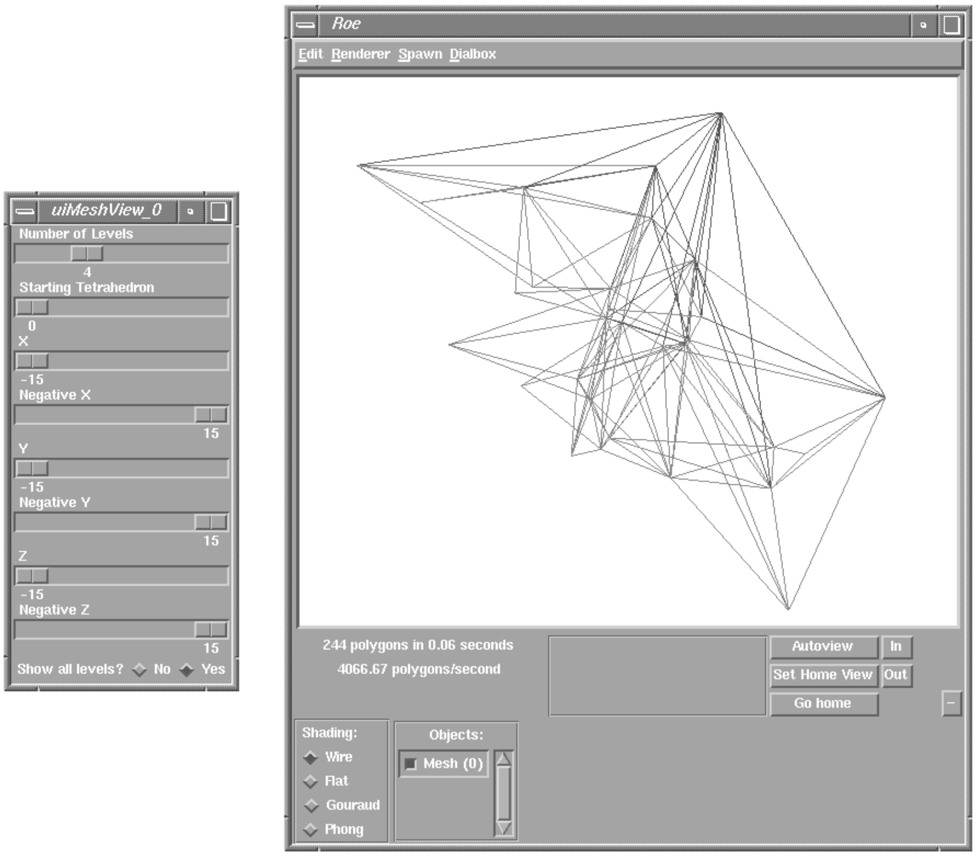

Figure 4: MeshView and Salmon modules, with partial mesh in geometry window

4 Conclusion

We have developed two different techniques for improving visualization of 3D unstructured meshes. Good visualization is imperative for understanding the information displayed in the meshes. Beyond visually assessing the mesh, the researcher may also need to understand regional changes due to adaptive mesh refinements. Traditional methods of mesh visualization have not provided a flexible enough interface for extracting such information. With these new techniques, the user is able to interactively view the meshes to better understand the mesh. Improvements currently being implemented include optimizing the MeshView program and implementing an interactive 3D widget technique to select the starting tetrahedron and clipping isosurfaces instead of using slider bars.

5 Acknowledgements

This work was supported in part by awards from the Whitaker Foundation, the NIH, and the NSF. The authors would like to thank K. Coles, S. Parker, and J. Schmidt for their helpful comments and suggestions.

References

[HS92] S. Hammond and R. Schreiber. Mapping unstructured grid problems to the connection machine. In P. Mehrotra, J. Saltaz, and R. Voigt, editors, *Unstructured Scientific Computation on Scalable Multiprocessors*, pages 11–30. MIT Press, 1992.

[JM94] C.R. Johnson and R.S. MacLeod. Nonuniform spatial mesh adaption using a posteriori error estimate: applications to forward and inverse problems. *Applied Numerical Mathematics*, 14:311–326, 1994.

[JP94] C.R. Johnson and S.G. Parker. A computational steering model applied to problems in medicine. In *Supercomputing '94*, pages 540–549. IEEE Press, 1994.

[MEJ94] R.S. MacLeod, P.R. Ershler, and C.R. Johnson. Map3d: Scientific visualization program for multichannel time series data on unstructured, three-dimensional meshes. program user's guide. Technical Report UUCS -94-016, University of Utah, Department of Computer Science, 1994.

[Ous94] J.K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[SK90] D. Speray and S. Kennon. Volume probes: Interactive data exploration on arbitrary grids. *Computer Graphics*, 24(5), 1990.

[SL94] A. Shostko and R. Lohner. Three-dimensional parallel unstructured grid generation. *AIAA*, 1994.

[YJ94] F. Yu and C.R. Johnson. An automatic adaptive refinement and derefinement method. In *Proceedings of the 14th IMACS World Congress*, pages 1555–1557, 1994.