SANDIA REPORT
SAND98–0233 • UC–705
Unlimited Release
Printed January 1998

Development of an Immersive Environment to Aid in Automatic Mesh Generation

LDRD Final Report

Constantine J. Pavlakos, Jake S. Jones, Scott A. Mitchell

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

SF2900Q(8-81)

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND98-0233
Unlimited Release
Printed January 1998

Distribution
Category UC-705

Development of an Immersive Environment to aid in Automatic Mesh Generation

LDRD Final Report

Constantine J. Pavlakos
Computer Architectures Department

Jake S. Jones
Computer Applications for Manufacturing Department

Scott A. Mitchell
Parallel Computing Sciences Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185

Abstract

The purpose of this work was to explore the use of immersive technologies, such as those used in synthetic environments (commonly referred to as virtual reality, or VR), in enhancing the mesh-generation process for 3-dimensional (3D) engineering models. This work was motivated by the fact that automatic mesh generation systems are still imperfect -- meshing algorithms, particularly in 3D, are sometimes unable to construct a mesh to completion, or they may produce anomalies or undesirable complexities in the resulting mesh. It is important that analysts and meshing code developers be able to study their meshes effectively in order to understand the topology and quality of their meshes. We have implemented prototype capabilities that enable such exploration of meshes in a highly visual and intuitive manner. Since many applications are making use of increasingly large meshes, we have also investigated approaches to handle large meshes while maintaining interactive response. Ideally, it would also be possible to interact with the meshing process, allowing interactive feedback which corrects problems and/or somehow enables proper completion of the meshing process. We have implemented some functionality towards this end -- in doing so, we have explored software architectures that support such an interactive meshing process. This work has incorporated existing technologies developed at Sandia National Laboratories, including the CUBIT mesh generation system, and the EIGEN/VR (previously known as MUSE) and FLIGHT systems, which allow applications to make use of immersive technologies and advanced human computer interfaces.

Acknowledgments

Larry A. Schoof and Philip L. Stanton -- for their contributions to the initial LDRD project proposal.

Tom Anderson -- for FLIGHT development and for his assistance in getting us started on FLIGHT application development.

Introduction

Effective computational analysis is playing an ever increasing role in minimizing the total time and cost to successful completion of a manufacturing process. An important step in computational analysis is the creation of the computational mesh, which discretizes a geometric representation of the three-dimensional (3D) model to be analyzed into finite volumes. In Finite Element Analysis (FEA), which has become the analysis technique of choice for a broad range of numerical simulations, the mesh is of an unstructured form whose many little finite volumes are commonly in the form of hexahedral and/or tetrahedral elements. Because the meshing step for FEA has traditionally been very tedious and time-consuming, significant effort has been and continues to be devoted to develop tools to automate the mesh generation procedure.

Unfortunately, current algorithms for automatic mesh generation are not perfect, in that it cannot be guaranteed that the automatic meshing procedure will yield a quality mesh in all cases. Sometimes, algorithms may produce anomalies and/or undesirable complexities in the resulting mesh, and other times, an algorithm may be overwhelmed by the complexity of a model and be unable to fully complete a mesh. It is important that analysts and meshing code developers be able to examine results of the meshing process, in order to be able to refine the mesh as necessary and/or the algorithm that produced it.

The goal of this project has been to explore the use of immersive technologies, such as those used in synthetic virtual environments (commonly referred to as virtual reality, or VR), towards development of tools, which could enhance the mesh generation process. The objectives have been to enable interaction with meshes, either complete or under construction, in a highly visual and intuitive manner, allowing a much greater understanding of the mesh, as well as potentially allowing interactive feedback to the mesh generation process.

Underlying Technologies

A number of existing technologies developed at Sandia National Laboratories provided a foundation for this work. They are described briefly below.

CUBIT

CUBIT [1,2,3] is the product of an ongoing research and development effort at Sandia in automatic mesh generation. The CUBIT mesh generation/grid generation environment is a two- and three-dimensional finite element mesh generation tool which is being developed to pursue the goal of robust and unattended mesh generation -- effectively automating the generation of quadrilateral and hexahedral elements. It is a solid-modeler based preprocessor that meshes volume and surface solid models for finite element analysis. A combination of techniques including paving, plastering, mapping, sweeping, and various other algorithms being developed are available for discretizing the geometry into a finite element mesh.

Exodus II

Exodus II [4] defines a Finite Element (FE) data model which is used, in common, by various FE applications at Sandia, including mesh generation codes, analysis codes, and visualization codes. Exodus II standardizes a file format and an application programming interface (API) for writing/reading FE data, including mesh geometries. Exodus II provided the file interface used by this project to export data from CUBIT to our immersive visualization prototypes.

EIGEN/VR

EIGEN/VR [5] is a platform for development of synthetic environment applications, which enables exploration of advanced human-computer interfaces. EIGEN/VR is the current incarnation of a system that was once known as the Multi-dimensional User-oriented Synthetic Environment (MUSE) [6]. Using a design approach based on human functionality, EIGEN/VR provides tools for the presentation, exploration, navigation, manipulation, and examination of data, models, and other types of information. The system provides complete device independence and currently supports flat screen, stereo, VR operation, voice recognition, sound synthesis, data sonification, and a variety of commercial interactive devices. Figure 1 shows some of the components of EIGEN/VR in active use for mesh visualization, including the ability to project onto a large wall, which gives a strong sense of immersion when used with stereo viewing.

FLIGHT

FLIGHT, like EIGEN/VR, provides a platform for application development which allows exploration with advanced human-computer interfaces. In particular, FLIGHT integrates a haptic input device known as the PHANToM, from SensAble Technologies, which adds a sense of touch to the human-computer interface. FLIGHT also provides a set of three-dimensional control widgets (in contrast to the two-dimensional widgets provided by standard window environments) which are used to present the basic control interface to FLIGHT, as well as other application-specific controls. Figure 2 shows the desktop FLIGHT environment in active use for mesh visualization.

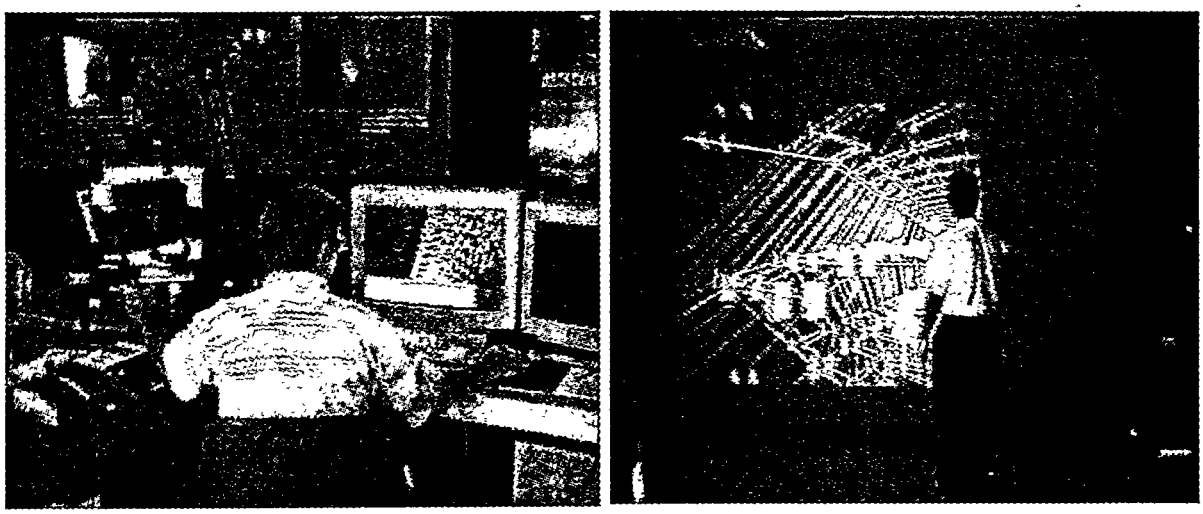

Figure 1. The EIGEN/VR environment.

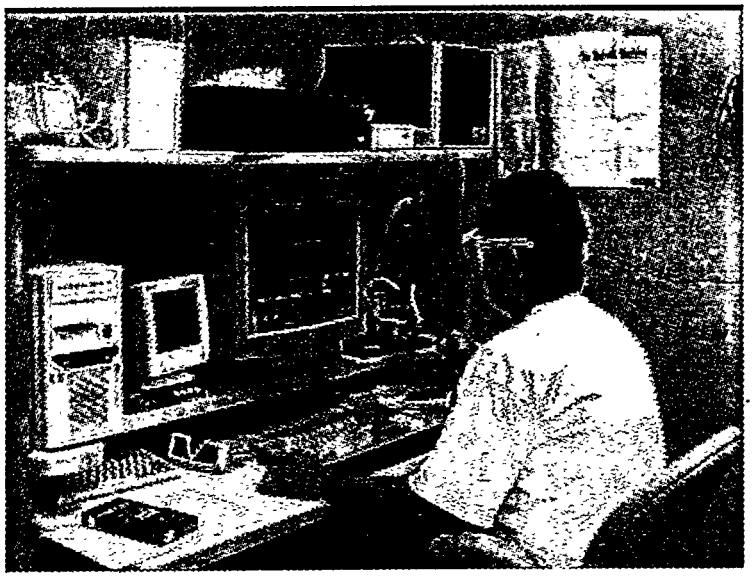

Figure 2. The FLIGHT Environment.

Intentionally Left Blank

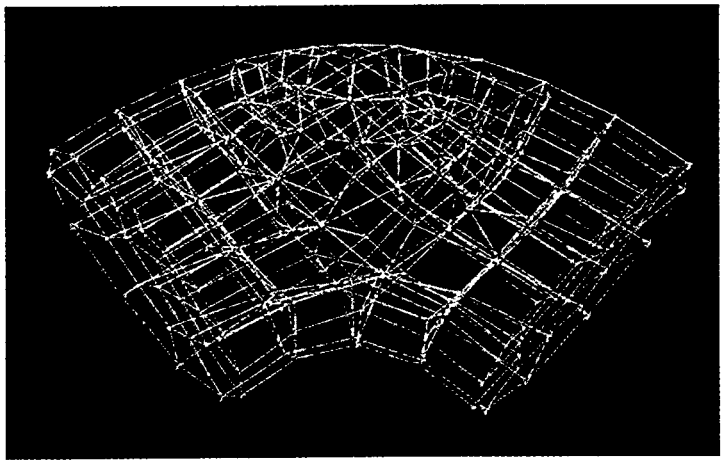

Figure 2. An entire mesh displayed for a "macaroni" object.

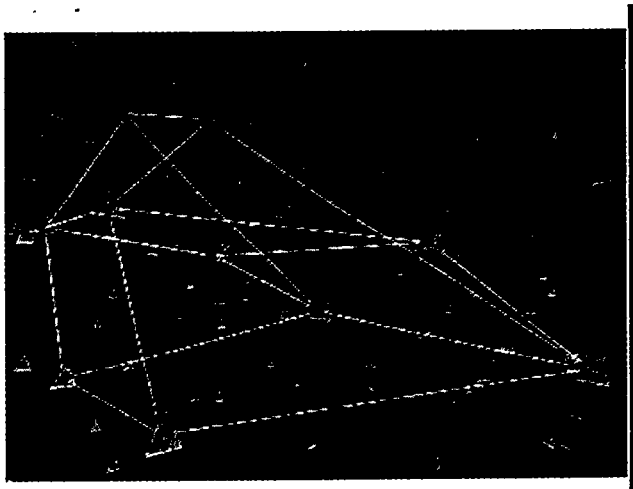

Figure 4. Two elements in the macaroni mesh which share two faces -- an anomaly.

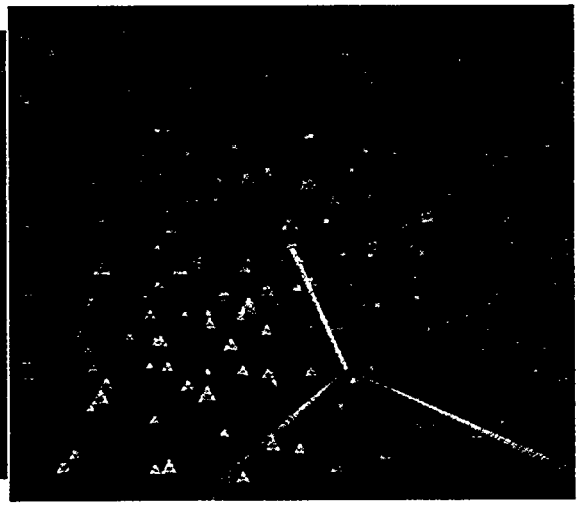

Figure 5."Tethered" to a node -- the elements associated with the node are highlighted -- in this case more than 20 elements contain the same node.

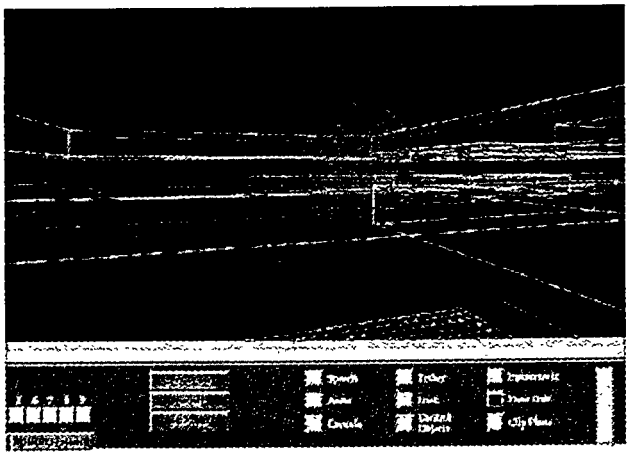

Figure 6. Inside a global view (element block boundaries only) of a 1.5 million element mesh with a "sphere of interest"

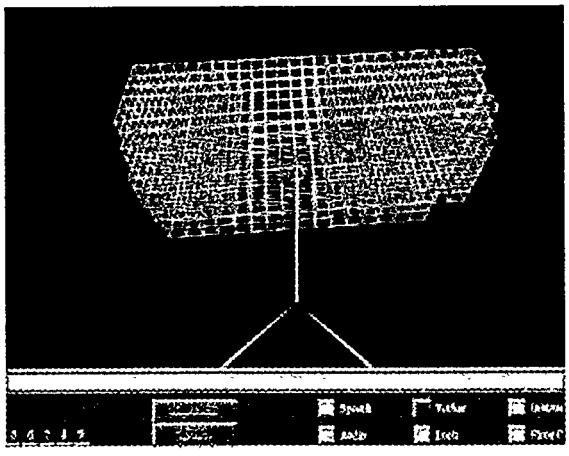

Figure 7. The detailed mesh extracted from the "sphere of interest" in Figure 6.

Intentionally Left Blank

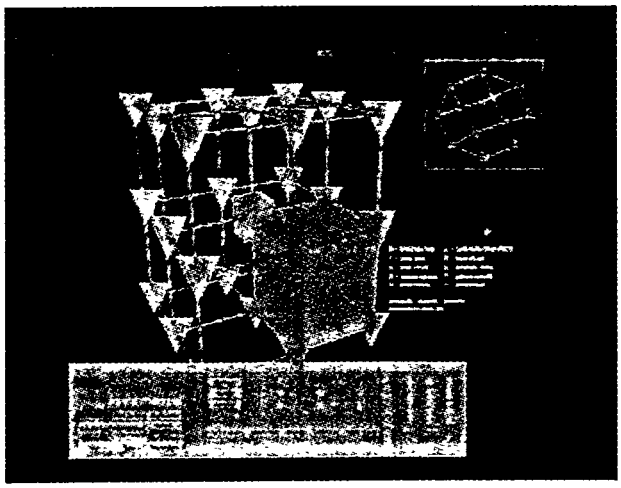

Figure 8. Whisker-weaving sheet diagram displayed with a corresponding mesh for entity association.

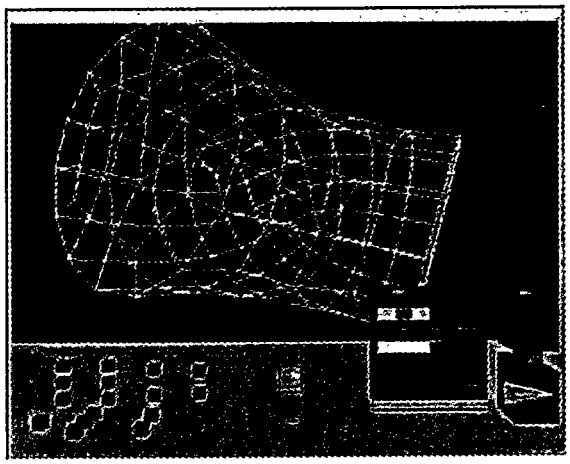

Figure 9. Using FLIGHT to view the external face/ edge representation extracted from a mesh.

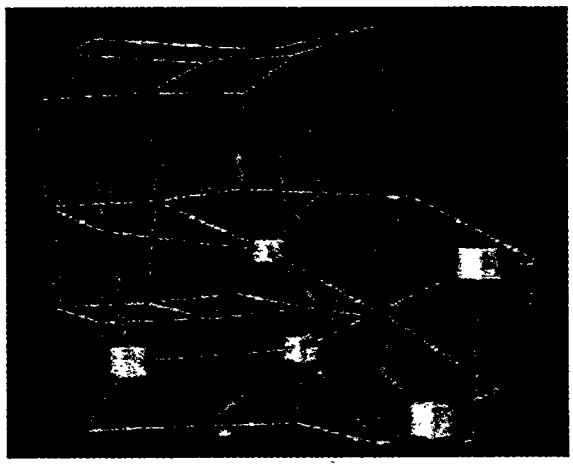

Figure 10. A group of elements, including a bad element, in the vicinity of a "dangling edge".

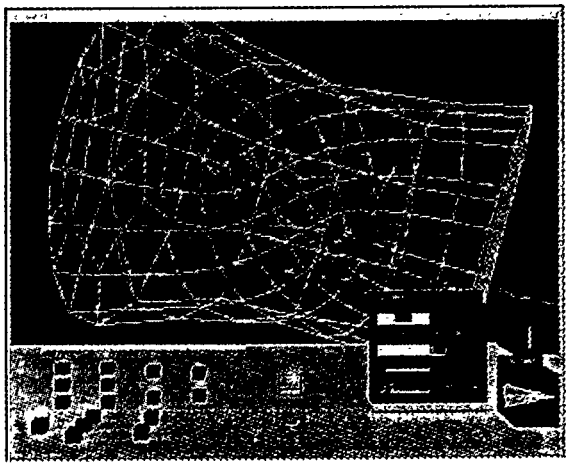

Figure 11. An aspect ratio quality metric visualized together with the mesh external boundary.

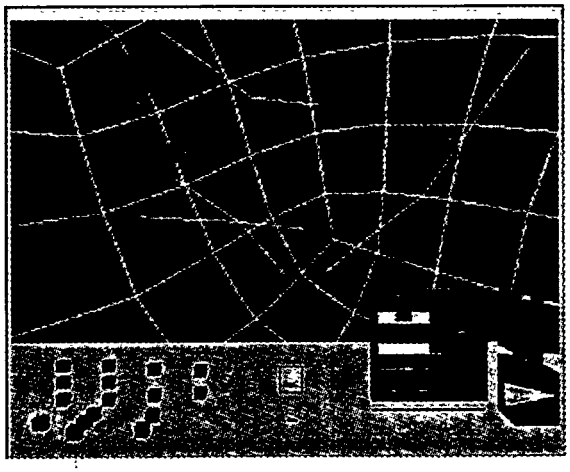

Figure 12. Navigating the inside, after filtering away low aspect values -- note the "dangling edges".

Intentionally Left Blank

Project Accomplishments

The Initial Prototype

Our initial development was based on the EIGEN/VR technology. We implemented a prototype application in a high-end VR laboratory environment, which provided high performance visualization hardware and a rich set of interface devices. Features of the application included:

- The ability to read in CUBIT meshes from Exodus II files.
- The ability to visually differentiate between nodes in the mesh which belong to 8 elements or less and nodes which belong to more than 8 elements, indicating a certain amount of relative complexity in the mesh (note: if a node exists in the interior of a hexahedral mesh, then the ideal is that it belongs to 8 unique elements).
- The ability to turn on/off mesh edges (i.e., to see only nodes, or nodes and edges).
- The ability to display a node id with each of the nodes.
- The ability to highlight specific elements in the mesh for visual scrutiny.
- The ability to "tether" to (i.e., visually focus attention to) a specific node, which also highlights the edges of all of the elements which contain the node for further visual scrutiny.
- The ability to display the surface of an element in shaded mode.
- The ability to grab and move nodes in the mesh, which demonstrates a rudimentary function which one might want to perform if editing a mesh.
- The ability to interact with all of these visual features in a fully stereoscopic environment, supported by the advanced human-computer interface capabilities provided by EIGEN/VR.

Some of these features are illustrated in Figures 3-5. Unfortunately, these still images cannot convey the impact of being able to explore such objects while in full motion and in three-dimensions. This is particularly true of Figures 3 and 5, which are very difficult to understand as flat, two-dimensional wire-frame drawings.

Upon seeing the prototype application, CUBIT's project leader, Tim Tautges, was quoted as follows: "***A capability like this on the desktop would increase our productivity by a factor of four or five for looking at meshes.***" This is corroborated by results from a recent study [7] which measured a three-fold improvement in comprehension when head-tracking and stereo viewing are used for data visualization.

To the Desktop

An important objective of this project, which is also conveyed in Tim Tautges' quote above, was to investigate capabilities for the desktop. For this reason, the project acquired a Silicon Graphics Indigo 2 workstation with Impact hardware-assisted graphics. This was a mid-range desktop/deskside workstation at the time, whose capabilities are rapidly becoming available on relatively inexpensive desktop systems. In addition to the workstation, other equipment acquired for the desktop system included Crystal Eyes stereo viewing equipment, a head-tracking unit, and a SpaceBall for 3D navigation.

This part of the project, in which we migrated the initial prototype application onto the desktop equipment, was only moderately successful. In doing the migration, we also ported the application to the standard OpenGL 3D graphics library (versus the old SGI-proprietary GL library) to make use of the emerging version of EIGEN/VR, which was also based on OpenGL. Unfortunately, this new version of EIGEN/VR did not support Crystal Eyes stereo viewing as quickly as we had hoped, and the head-tracking and SpaceBall devices were never enabled.

However, ultimately we did demonstrate the ability to run the prototype application in the desktop environment, with stereo viewing, for modest size meshes (a few thousand elements). Our attempts to run the desktop EIGEN/VR-based prototype on a large mesh (over one million elements) were unsuccessful. This experience certainly contributed to a revelation that we would need to consider other strategies to handle meshes of any significant size.

These initial efforts to develop desktop capability do not tell the whole desktop story. Some of the shortcomings in this stage of our work eventually led us to consider FLIGHT as an additional environment for prototype development, which we will discuss further later in this report.

Linking to CUBIT

Another objective of this project was to investigate the possibility that the immersive visualization system developed for this project could be coupled directly with the meshing system, in such a way that would allow the visualization system to impact the meshing process ("interactive meshing"), for example, by editing the mesh somehow while the mesh was still under construction. This implies considering strategies for interfacing the two systems.

An integration which would have implemented an EIGEN/VR shell around CUBIT was conceptually attractive, since it would enable the use of EIGEN/VR's advanced human-computer interface techniques to interact with CUBIT directly. However, this would have required a complete redesign and rewrite of the graphics functionality already embedded in CUBIT to conform to OpenGL, and this was deemed impractical when considering the relatively short term and limited resources of our project.

Thus, for the sake of prototyping, we chose to implement an architecture that interfaces CUBIT with an EIGEN/VR application, each running as a separate process, communicating with each other across a bidirectional link.

We successfully implemented such a link, and have been able to demonstrate the ability to pass a mesh directly from CUBIT to the visualization system. Also, as noted previously, we have demonstrated the ability to edit a mesh by changing the position of a node in the mesh from within the prototype visualization system.

However, this is the extent to which we have been able to explore interactive meshing. Any further work in this area was relegated by the need to investigate techniques for handling of larger meshes. Additionally, as it turns out, the link architecture we implemented does not work well in practice (see more in "Software Architectures for Interactive Meshing" section).

Large Meshes

In order to provide a useful tool for exploration of meshes, that tool must accommodate meshes that are consistent in size with requirements of day-to-day applications. While the capabilities we developed in early stages of the project were powerful for intuitive comprehension of a mesh, they did not readily support large meshes. During the course of the project, we came upon the opportunity to test our prototype system against a mesh containing about one and one-half million elements. This is certainly a significant-size mesh, although we are already observing the need for applications to do larger, up to about ten million elements today. This data set became a benchmark for our project.

When the initial prototype started, it would load a specified mesh and then attempt to display the mesh in its entirety, displaying all edges and all nodes. When running the prototype against our benchmark mesh on high-end hardware, performance was painfully slow -- much too slow to allow interactive exploration of the mesh. Worse, when trying to run in our desktop environment, the application would fail.

It became clear that we needed to reassess our implementation and take a whole new approach for enabling use with large meshes. Actually, such a new approach was needed regardless of large meshes. Even for smaller meshes, it is not very effective to throw the entire mesh up on the display in wire-frame form. There is simply too much information to absorb. Trying to find local areas in the mesh which may be of special interest is just too difficult when one has no idea where to start looking, much like the proverbial needle in the haystack.

This all suggests that some sort of hierarchical approach to exploring a mesh is needed, both to accomodate interactive graphics response and to alleviate the data overload problem. One approach we took is illustrated in Figures 6 and 7. In Figure 6, a global spatial context for the mesh is portrayed by displaying bounding boxes for each of the element blocks in the mesh (where element blocks are groups of elements into which a mesh may be decomposed). A "sphere of interest" can be arbitrarily positioned and scaled within the global context, upon which a request can be made to extract the detailed mesh within the sphere of interest for display and further scrutiny.

The bounding-box display can be somewhat less than satisfying in terms of visualizing the model represented by the mesh. An alternative we have implemented allows extraction of the external faces of the mesh, eliminating the edges in the interior of each element block. This representation offers a more pleasing display, however, for very large meshes, this data can still be quite large, affecting interactive response. This approach begs for a higher-level representation which accurately represents the model's geometry, while not showing any of the model discretization (e.g., a large flat face would appear as a single polygon rather than hundreds or thousands of quadrilaterals). The meshing system may have access to such a representation, in which case using it would simply require passing this representation along with the mesh.

In combination with higher-level representations of the global context of the mesh, there is still the issue of how to isolate localities in the mesh for further scrutiny. One approach is to use

certain quality metrics which can be computed for a mesh to guide such a search -- this approach was used within our FLIGHT prototype and is described in more detail below.

Once we isolate a place to look, it is possible to have the system display a few elements in that vicinity, or to start with one element and incrementally add layers of neighboring elements (as described in [8]). The full power of systems like our prototype can then be used to visually explore the extracted, relatively simple geometry.

While we have implemented most of the above approaches for exploring large meshes with some success, this work is not complete. Our implementations still lack robustness and there are still issues regarding how to manage the large data on smaller, more memory-constrained systems.

Sheet Diagrams for the Whisker Weaving Developer

While much of the functionality we developed during the course of this project is of general utility to analysts as well as meshing algorithm developers, in one case we demonstrated a capability that was specific to whisker-weaving [9] algorithm development. In whisker-weaving, the algorithm produces a set of two-dimensional graphs, called sheet diagrams, from which the mesh is ultimately constructed. We provided functionality that allows the user of the visualization code to examine sheet-diagrams simultaneously with a corresponding mesh. The user can selectively highlight an entity in the sheet-diagram, which, in turn, results in the associated entity being displayed/highlighted in the mesh. Of course, while traversing entities in the sheet diagram, one can make use of the full 3D visual exploration capabilities of the visualization environment to examine the associated mesh entities. This can help greatly in understanding the relationships between the two-dimensional graph and the three-dimensional mesh, which can otherwise be very difficult to imagine. This capability is shown for a very simple mesh in Figure 8 -- the element associated with the selected node in the sheet diagram is displayed as a shaded object (the sheet diagram is in the upper right of the display).

A Prototype based on FLIGHT

During the course of our project, a new system for computer-human interface exploration was being developed which integrated the sense of touch -- that system was FLIGHT. FLIGHT appealed to us as an application development platform in a variety of ways:

- it is well suited to desktop applications;
- the system offers a SpaceBall navigation/control interface as an alternative to the PHANToM;
- the PHANToM is intriguing as a device for navigation and control, as is the potential for a sense of touch;
- the code itself is compact enough that we thought it would be relatively straightforward to add features such as stereo and head tracking;
- the 3D user interface (i.e. control widgets) could still be manipulated and viewed normally even when stereo viewing is enabled.

The opportunity to experiment with FLIGHT presented itself in the very latter stages of our project -- we began FLIGHT prototyping with only 2-3 months remaining, and even then, some

of the features we made use of were still being completed or refined. However, we were able to demonstrate significant prototype capability, which, when extrapolated with certain functionality we have demonstrated in the EIGEN/VR environment, would provide a powerful desktop environment for mesh visualization.

Figure 9 portrays the use of FLIGHT to view the external face/edge representation extracted from the "macaroni" mesh. The algorithm we implemented to extract external faces/edges is conceptually simple -- if a mesh face occurs in the data only once, it is an external face, and its edges are external edges. Interestingly enough, when this algorithm is applied to the macaroni mesh, it leaves certain "dangling edges" (see Figure 12). These dangling edges are very easy to observe within the FLIGHT-based application. After having observed these artifacts, we went back to the EIGEN/VR environment, imported the same external face/edge extraction algorithm, and used the "sphere-of-interest" capability to extract a small amount of detail in the vicinity of a dangling edge. Sure enough, as Figure 10 shows (towards the lower right), there is a bad element in the mesh at that location. (Note: The "macaroni" sample mesh we used throughout the project was taken from CUBIT in the very early stages of our project -- CUBIT is now able to mesh this object correctly.)

This example identifies only one of many ways in which one might be led to locations in a mesh that are worthy of further scrutiny. An additional approach we implemented within the FLIGHT application made use of certain quality metrics, which we compute for the mesh. An example is depicted in Figure 11. An aspect ratio value computed for each of the hexahedra in the mesh is visualized as red tetrahedra -- the red tetrahedra are merely 3D icons at the center of each hexahedra, with larger ones indicating larger aspect values and smaller ones (or invisible ones) indicating lower aspect values. We also implemented an application control that allows the user to easily filter away lower values, using a 3D slider to change the filter threshold. In Figure 12, we have moved inside the macaroni mesh data, after we have filtered away most of the lower aspect values -- the high-aspect value locations that remain may be areas in the mesh where the user would then like to request more mesh detail for further scrutiny.

We were able to add stereo viewing to the FLIGHT application -- we enabled use of Crystal Eyes. While we had fully expected the 3D graphical user interface to remain functional once stereo was turned on, it was still very satisfying to observe this in actuality, having been accustomed to losing normal viewing of 2D components in traditional windowing environments when the monitor mode was changed to stereo. Due to lack of time, we have not yet enabled head tracking.

The PHANToM has proven to be a real pleasure to use as a 3D navigation and control device, and the haptic sensations already provided by the FLIGHT system when interacting with the user interface have a way of enriching the human experience. We have not used the haptic capability yet with regard to providing force feedback when interacting with the data itself, but we have begun to imagine certain possibilities. For example, the device can be used to emulate magnetic forces -- by applying magnetic forces at locations in the mesh that may be of special interest, it may be possible to draw the user towards such locations as the mesh is being navigated.

We did attempt to use the SpaceBall interface to FLIGHT. It is difficult to make any final observations regarding the utility of this device, since the device interface has not been fully refined. However, it is fair to say that, at this time, it is clumsy at best. Also, it provides no haptic feedback. Finally, even when comparing to the PHANToM for 3D navigation only, we found the PHANToM to be superior.

Software Architectures for Interactive Meshing

We use the term "interactive meshing" to capture the notion of an interactive mesh visualization system being used, somehow, in direct cooperation with a meshing system to produce a resulting mesh. The role of the visualization component may be to edit a mesh when problems have been detected during mesh construction, or, perhaps, to enable manual decomposition of very complex models into simpler parts which can be meshed more straightforwardly. In any case, such cooperation would require a tight coupling of the visualization and meshing components.

As described previously, we implemented a bidirectional link between CUBIT and our EIGEN/VR-based prototype mesh visualization system which allows passing data and other information back and forth. The main problem with this architecture is that data is copied -- more specifically, in the case of a very large mesh, this would mean duplicating the mesh data within the visualization system. If the visualization system performs any editing, then all or part of the mesh would need to be sent back to the meshing system accordingly. Clearly, this becomes inefficient for large meshes.

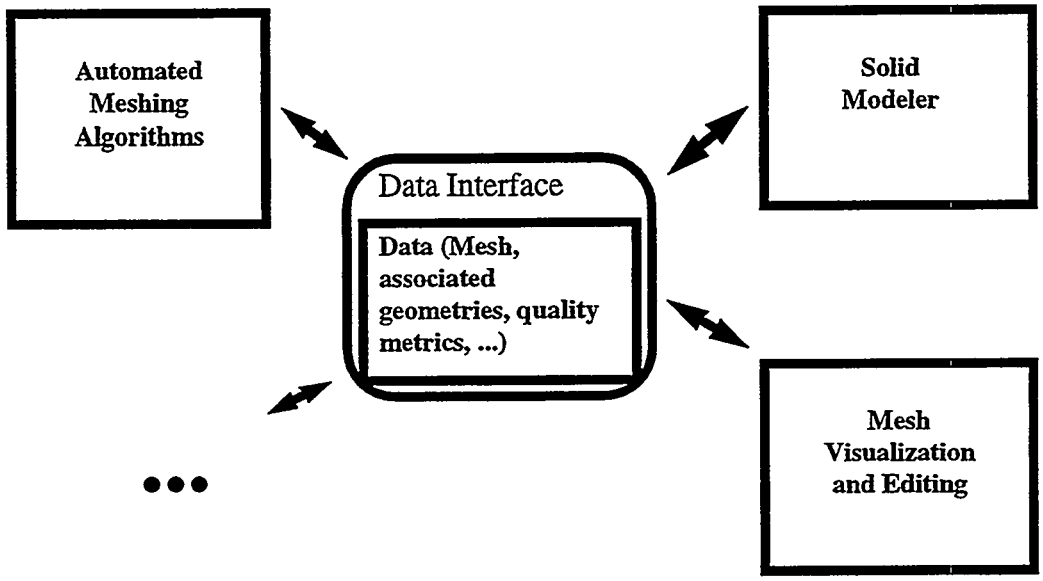

Figure 9. A data-centric approach for program module integration and interoperability

This beckons for a more data-centric approach in which the meshing system and the visualization system could both access the same data, using a standard interface. This can certainly be accomplished by implementing a single monolithic program, embedding all of the visualization capabilities and meshing capabilities within an integrated system. CUBIT, in its current form, actually provides such a framework -- CUBIT is largely implemented as a set of object-oriented, C++ class libraries.

However, embedding the visualization technology within a monolithic system such as CUBIT ultimately limits its availability for use with other meshing systems. Regardless of the quality of internal design, a single monolithic program environment does not encourage broad interoperability with other independently-developed software components. An alternative to this approach is a multi-program environment that provides access to common data via a common data broker and/or data protocol. Shared memory could be used to provide efficient, simultaneous access to the one copy of the data, from multiple programs. Such an architecture is depicted in Figure 13.

Key to all of this is abstracting the data interface and making the data available to multiple, potentially interoperating software components. It is clear that such a data interface is possible for enabling multiple program modules to use a single instantiation of the data, in memory, when the programs are not interacting, much like Exodus II does via files, with similar benefits. However, interaction does complicate things -- it introduces issues such as how to coordinate which program(s) can be modifying the data at any point in time, and whether a set of generic events can be defined which would enable plug-and-play program modules that interact to the desired extent. These issues are resolvable in a confined application environment -- that is, if the program modules are implemented with full knowledge of each other, they can be made to cooperate accordingly. Whether these issues can be resolved or not for a more wide open component software environment is still an open question.

Future Work

We have identified certain areas where this work could be extended and/or refined. They include:

- Integration of the functionality we have demonstrated separately in the EIGEN/VR and FLIGHT environments into a single application. Most likely, we would be inclined to extend the FLIGHT prototype to incorporate select features of the EIGEN/VR prototype.
- Use of other higher-level geometric representations (e.g., untesselated surface boundaries) of the global mesh space, as possible, to improve on graphics rendering performance when viewing the entire global context for large meshes.
- Overall improved robustness for handling of large meshes.
- Investigation of how the haptic feedback provided by the FLIGHT system might be used to help explore meshes (e.g., by using magnetic forces to draw user towards local areas of interest).
- Further investigation of "interactive meshing".

Conclusion

This project has demonstrated an extensive set of prototype capabilities that show a great deal of promise for helping analysts and meshing-code developers understand the quality and topology of their 3D meshes. Indeed, it has been observed that such capabilities could improve one's productivity for looking at meshes by as much as five fold.

We are particularly pleased with the prototype work we have done with the FLIGHT system. We are excited about the potential for extending the prototype we have developed into a powerful end-user tool for exploration of meshes. Our experiences with haptics have convinced us that touch is destined to become a common feature of computer-human interfaces in the next few years.

While stereo viewing may have limited significance for some 3D applications, we have found it to be very useful for mesh visualization. One important reason is that meshes, by nature, are inherently wire-frame, and stereo helps provide important depth perception when looking at 3D wire-frame geometries. Stereo also helps in understanding depth relationship of multiple objects on the screen -- this is useful when, for example, one is trying to interactively specify a region-of-interest within a global mesh space by positioning an object, such as a sphere, in just the right place.

We have demonstrated approaches for handling large mesh data, but there is still room for improvement. Hierarchical approaches to exploring a large mesh are needed -- it is not practical to look at a full mesh all at once, both because of graphics performance constraints and human data overload. Similarly, we must continue to explore approaches that guide the user toward local areas in a mesh that are worthy of more detailed scrutiny -- one cannot expect to find such areas using a brute-force visual search.

References

[1]Blacker, T.D., et al, "CUBIT Mesh Generation Environment, Volume 1: Users Manual," Sandia National Laboratories Technical Report, SAND94-1100, May 1994.

[2]G. D. Sjaardema, et al," CUBIT Mesh Generation Environment, Volume 2: Developers Manual," Sandia National Laboratories Technical Report, SAND94-1101, 1994.

[3]URL: "http://sass577.endo.sandia.gov/SEACAS/CUBIT/Cubit.html"

[4]Schoof, L.A., Yarberry, V.R., "Exodus II: A Finite Element Data Model," Sandia National Laboratories Technical Report, SAND92-2137, Sep 1994.

[5]URL: "http://www.cs.sandia.gov/SEL"

[6]Maples, C.C., "Multidimensional, User-Oriented Synthetic Environment, A Functionally Based, Human-Computer Interface," The International Journal of Virtual Reality, Vol. 1, Number 1, Winter 1995.

[7]Ware, C., Franck, G., "Evaluating Stereo and Motion Cues for Visualizing Information Nets in Three Dimensions," ACM Transactions on Graphics, Vol. 15, No. 2, pp. 121-140, April 1996.

[8]C. S. Gitlin and C. R. Johnson, "MeshView: A Tool for Exploring 3D Unstructured Tetrahedral Meshes," Proceedings of the 5th International Meshing Roundtable, pp. 333-345, 1996.

[9]T. J. Tautges, T. D. Blacker, and S. A. Mitchell, "Whisker Weaving: a connectivity based method for constructing all-hexahedral finite element meshes," International Journal of Numerical Methods in Engineering, Vol. 39, Number 19, pp. 3327-3350, 1996.

DISTRIBUTION:

1	MS-0188	C. E. Meyers, 4523
1	MS-0188	D. Chavez, 4523
1	MS-0960	J. Q. Searcy, 1400
1	MS-0957	R. C. Reuter Jr., 1401
1	MS-0957	C. S. Leishman, 1401
1	MS-0151	R. D. Skocypec, 9002
1	MS-0841	P. J. Hommert, 9100
1	MS-0828	T. C. Bickel, 9101
1	MS-0828	J. H. Biffle, 9103
1	MS-0826	W. L. Hermina, 9111
1	MS-0826	M. W. Glass, 9111
1	MS-0834	A. C. Ratzel, 9112
1	MS-0835	S. N. Kempka, 9113
1	MS-0827	R. O. Griffith, 9114
1	MS-0825	W. Rutledge, 9115
1	MS-0836	C. W. Peterson, 9116
1	MS-0443	H. S. Morgan, 9117
1	MS-0443	M. L. Blanford, 9117
1	MS-0443	V. L. Porter, 9117
1	MS-0443	G. D. Sjaardema, 9117
1	MS-0443	L. A. Schoof, 9117
1	MS-0437	S. W. Attaway, 9117
1	MS-0321	W. J. Camp, 9200
1	MS-0318	G. S. Davidson, 9215
1	MS-0318	A. Breckenridge, 9215
1	MS-1109	C. F. Diegert, 9215
1	MS-0318	P. Heermann, 9215
1	MS-0318	V. Holmes, 9215
1	MS-0318	D. Zimmerer, 9215
1	MS-1111	S. S. Dosanjh, 9221
1	MS-1111	D. R. Gardner, 9221
1	MS-1111	G. Hennigan, 9221
1	MS-1111	S. Hutchinson, 9221
1	MS-1111	S. Plimpton, 9221
1	MS-1111	J. N. Shadid, 9221
1	MS-1110	D. E. Womble, 9222
1	MS-1110	D. Greenberg, 9223
1	MS-1109	A. L. Hale, 9224
1	MS-1111	G. Heffelfinger, 9225
1	MS-0441	R. W. Leland, 9226
1	MS-0441	T. Tautges, 9226
1	MS-0441	D. R. White, 9226
1	MS-0819	J. Peery, 9231
1	MS-0819	M. A. Christon, 9231
1	MS-0820	P. Yarrington, 9232
1	MS-0439	D. R. Martinez, 9234

10	MS-1110	C. J. Pavlakos, 9215
5	MS-0957	J. S. Jones, 1401
5	MS-0441	S. A. Mitchell, 9226
1	MS-9018	Central Technical Files, 8940-2
5	MS-0899	Technical Library, 4916
2	MS-0619	Review & Approval Desk, 12690 For DOE/OSTI
1	MS-0161	Patent and Licensing Office, 11500
1	MS-0188	LDRD Office, 4523